

# **An Animation-Based Teaching and Learning Framework for Struggling Introductory Programming Students**

by

**Tlou James Ramabu**

submitted in accordance with the requirements for  
the degree of

**DOCTOR OF PHILOSOPHY**

in the subject

**COMPUTER SCIENCE**

at the  
University of South Africa

Supervisor: Prof Ian Sanders

Co-supervisor: Prof Marthie Schoeman

August 2023

## Declaration

I declare that **AN ANIMATION-BASED TEACHING AND LEARNING FRAMEWORK FOR STRUGGLING INTRODUCTORY PROGRAMMING STUDENTS** is my own work and that all sources used or quoted have been indicated and acknowledged by means of complete references.

I further declare that I submitted this thesis to the appropriate originality detection system which is endorsed by Unisa and that it falls within the accepted requirements for originality.

I further declare that I have not previously submitted this work, or any part of it, for examination at Unisa for another qualification or at any other higher education institution.



---

James Ramabu

16 August 2023

---

Date

## **Acknowledgements**

- I would like to thank my supervisors, Prof Ian Sanders and Prof Marthie Schoeman, for their valuable guidance during this study. Their encouragement and motivation generated positive energy even during the difficult times of Covid-19.
  
- I would like to thank my family for their patience and support throughout the study.
  
- I thank God for the blessing of life and health to complete this study in the name of Jesus Christ. To God be the glory.

## Table of Contents

List of Figures .....	vii
List of Tables.....	xi
Abstract.....	xiii
Summary .....	xv
Summary (Afrikaans language).....	xvi
Summary (Sepedi / Northern Sotho language) .....	xvii
List of publications.....	xviii
List of abbreviations and acronyms .....	xix
Definition of key terms .....	xx
<b>Chapter 1: Introduction .....</b>	<b>1</b>
1.1 Introduction.....	2
1.2 Background to the study.....	2
1.2.1 <i>Struggling Introductory Programming Students</i> .....	2
1.2.2 <i>Teaching and learning of introductory programming concepts</i> .....	3
1.2.3 <i>Action research</i> .....	4
1.2.4 <i>Animation programs</i> .....	4
1.3 Problem statement .....	4
1.4 Research questions and objectives .....	6
1.4.1 <i>Research questions</i> .....	6
1.4.2 <i>Research objectives</i> .....	6
1.4.3 <i>Brief methodology</i> .....	7
1.5 Significance of the study.....	7
1.6 Chapter outline.....	8
1.7 Conclusion .....	9
<b>Chapter 2: Literature review.....</b>	<b>10</b>
2.1 Introduction.....	11
2.2 Purpose of the literature review.....	12
2.3 Contextual overview.....	13
2.3.1 <i>Importance of programming</i> .....	13
2.3.2 <i>Learning to program is difficult</i> .....	13
2.3.3 <i>Computer Science at universities and K12</i> .....	14
2.3.4 <i>Overview of programming education research</i> .....	15
2.4 Challenges in learning to program.....	16
2.4.1 <i>Struggling Introductory Programming Students (SIPS)</i> .....	16

2.4.2 English language.....	17
2.4.3 Mathematics.....	18
2.4.4 Mental models.....	19
2.4.5 Programming jargon .....	20
2.4.6 Problem solving.....	20
2.4.7 Summary .....	21
2.5 Imperative programming approach.....	22
2.5.1 Programming languages.....	22
2.5.2 Programming paradigms.....	23
2.5.3 The structure of procedural programming.....	24
2.5.4 Details of learning challenges .....	28
2.5.5 Summary .....	30
2.6 Approaches to teaching procedural programming .....	31
2.6.1 General teaching challenges in introductory programming.....	31
2.6.2 Traditional methods of teaching and learning to program .....	32
2.6.3 Analogy-like teaching approaches for introductory programming.....	32
2.6.4 Manipulatives in teaching and learning introductory programming.....	33
2.6.5 Using animation programs.....	34
2.6.6 Summary .....	35
2.7 Animation programs .....	36
2.7.1 Block-based programs.....	37
2.7.2 Animation-based programs.....	39
2.7.3 Pedagogical guidelines for uses of animation programs .....	44
2.8 Gap and summary .....	44
<b>Chapter 3: Theoretical framework.....</b>	<b>46</b>
3.1 Introduction.....	47
3.2 Learning theories.....	47
3.2.1 Constructivism .....	48
3.2.2 Piagetian theory.....	48
3.2.3 Neo-Piagetian theories.....	49
3.2.4 Behaviourism .....	51
3.2.5 Cognitivism.....	52
3.2.6 Cognitive theory of multimedia learning.....	52
3.2.7 Conversational framework.....	53
3.3 Learning taxonomies.....	56
3.3.1 Bloom's taxonomy.....	56

3.3.2 <i>SOLO taxonomy</i> .....	57
3.4 Teaching philosophy .....	59
3.5 Constructive alignment .....	60
3.6 Applicable theoretical components of TLPAP .....	61
3.6.1 <i>Overview TLPAP theoretical components</i> .....	61
3.6.2 <i>Adopted learning theories</i> .....	62
3.6.3 <i>TLPAP teaching philosophy</i> .....	65
3.7 Conclusion .....	68
<b>Chapter 4: Methodological approach</b> .....	<b>70</b>
4.1 Introduction .....	71
4.2 Methodological perspective.....	72
4.3 Research paradigm .....	73
4.3.1 <i>Positivism</i> .....	73
4.3.2 <i>Post-positivism</i> .....	73
4.3.3 <i>Interpretivism</i> .....	74
4.3.4 <i>Critical theory</i> .....	74
4.3.5 <i>Pragmatism</i> .....	74
4.3.6 <i>Adopted research paradigm</i> .....	75
4.4 Research design.....	75
4.4.1 <i>Animation programs design</i> .....	76
4.4.2 <i>Action research</i> .....	79
4.4.3 <i>Experimental setup, pre-testing and post-testing of SIPS</i> .....	81
4.4.4 <i>Background of the participants</i> .....	82
4.4.5 <i>Control and experimental groups setup</i> .....	83
4.5 Evaluation strategy .....	84
4.6 Action cycles progression.....	90
4.6.1 <i>Timeline overview</i> .....	90
4.6.2 <i>Cycle 1</i> .....	90
4.6.3 <i>Cycle 2</i> .....	91
4.6.4 <i>Cycle 3</i> .....	91
4.6.5 <i>Cycle 4</i> .....	91
4.6.6 <i>Cycle 5</i> .....	92
4.7 Limitations of the study .....	92
4.8 Ethics statement .....	92
4.9 Conclusion .....	93
<b>Chapter 5: Initial framework – Cycle 1</b> .....	<b>95</b>

5.1 Introduction .....	96
5.2 Planning for cycle 1 .....	96
5.2.1 SIGCSE members' participation .....	96
5.2.2 Identification of manipulatives for introductory concepts .....	97
5.3 Implementation of the manipulatives and results .....	100
5.3.1 Cycle 1 (Concept 1) .....	101
5.3.2 Reflection on cycle 1 (Concept 1).....	104
5.3.3 Cycle 1 (Concept 2) report and reflections.....	105
5.4 Planning for cycle 2 .....	106
5.5 Conclusion .....	107
<b>Chapter 6: Adapted framework – cycle 2.....</b>	<b>108</b>
6.1 Introduction .....	109
6.2 Concept 1 (Assignment) .....	110
6.2.1 Pre-teaching exercises on the use of assignment .....	110
6.2.2 Animation program 1.....	111
6.2.3 Animation program 2.....	112
6.2.4 Animation program 3.....	114
6.2.5 Animation program 4.....	115
6.3 Concept 2 (Decisions/nested decisions).....	116
6.3.1 Problem specification for nested-decider .....	116
6.3.2 Nested-decider animation program .....	117
6.4 Concept 3 (Loops/nested loops) .....	122
6.4.1 Problem specification for loops and nested loops.....	122
6.4.2 Animation program for loop .....	123
6.5 Formulated pedagogical guidelines .....	126
6.6 Experimentation results .....	128
6.6.1 Summary of results (Animation programs on assignment).....	128
6.6.2 Summary of misconceptions found in the algorithms for decisions/nested decisions and loops. ....	131
6.6.3 SOLO-adapted algorithms evaluation - Decisions.....	133
6.6.4 SOLO-adapted algorithms evaluation - Loops.....	135
6.7 Reflections .....	137
6.8 Planning for cycle 3 .....	138
6.9 Conclusion .....	139
<b>Chapter 7: Adapted framework - Cycle 3.....</b>	<b>140</b>
7.1 Introduction .....	141

7.2 Animation programs for introducing new concepts .....	141
7.2.1 <i>Decisions/nested decisions</i> .....	142
7.2.2 <i>Repetitions</i> .....	149
7.2.3 <i>Arrays</i> .....	156
7.2.4 <i>Functions</i> .....	158
7.3 Animation programs with problem specifications .....	166
7.3.1 <i>Repetitions</i> .....	166
7.3.2 <i>Nested loop</i> .....	168
7.3.3 <i>Arrays</i> .....	171
7.3.4 <i>Functions</i> .....	176
7.4 Adapted pedagogical guidelines .....	181
7.5 Experimentation results and analysis .....	184
7.5.1 <i>Results (assignment)</i> .....	185
7.5.2 <i>Results (decisions/nested decisions)</i> .....	186
7.5.3 <i>Results (Loops)</i> .....	191
7.5.4 <i>Results (nested loops)</i> .....	194
7.5.5 <i>Results (one-dimensional array)</i> .....	197
7.5.6 <i>Results (parallel arrays)</i> .....	200
7.5.7 <i>Results (functions)</i> .....	203
7.6 Cycle 4 reflections .....	208
7.7 Planning for cycle 4 .....	211
7.8 Conclusion .....	212
<b>Chapter 8: Adapted framework - Cycles 4 and 5</b> .....	<b>214</b>
8.1 Introduction .....	215
8.2 Additional animation programs .....	216
8.2.1 <i>Combination of decisions and loops</i> .....	216
8.2.2 <i>Additional animation on concept introduction for arrays</i> .....	219
8.3 Experimentation results and analysis .....	222
8.3.1 <i>Results (assignment)</i> .....	222
8.3.2 <i>Results (decisions/nested decisions)</i> .....	223
8.3.3 <i>Results (loops/decisions)</i> .....	227
8.3.4 <i>Results (parallel arrays)</i> .....	231
8.3.5 <i>Results (functions)</i> .....	233
8.4 Summary of reflections on cycle 4 .....	235
8.5 Summary of reflections on cycle 5 .....	236
8.6 Discussions on transitions across cycle 3 to cycle 5 .....	237



8.7 State of the framework.....	241
8.8 Conclusion .....	243
<b>Chapter 9: Conclusion</b> .....	<b>244</b>
9.1 Introduction .....	245
9.2 Structure of the TLPAP framework .....	247
9.3 How the research questions were answered .....	248
9.4 Contributions .....	250
9.5 Validity, generalisation and critical reflection on the findings .....	254
9.6 Limitations.....	257
9.7 Future work.....	257
9.8 Conclusion .....	258
<b>References</b> .....	<b>261</b>
<b>Appendices</b> .....	<b>276</b>
<b>Appendix A: Ethics clearance</b> .....	<b>277</b>
<b>Appendix B: Evaluation sheets for decisions/nested decisions – Cycle 3</b> .....	<b>280</b>
<b>Appendix C: Evaluation sheets for loops – Cycle 3</b> .....	<b>286</b>
<b>Appendix D: Evaluation sheets for nested loops – Cycle 3</b> .....	<b>290</b>
<b>Appendix E: Evaluation sheets for one-dimensional array – Cycle 3</b> .....	<b>294</b>
<b>Appendix F: Evaluation sheets for parallel arrays – Cycle 3</b> .....	<b>299</b>
<b>Appendix G: Evaluation sheets for functions – Cycle 3</b> .....	<b>304</b>
<b>Appendix H: Evaluation sheets for decisions/nested decisions – Cycle 4</b> .....	<b>310</b>
<b>Appendix I: Evaluation sheets for decisions / loops – Cycle 4</b> .....	<b>316</b>
<b>Appendix J: Evaluation sheets for parallel arrays – Cycle 4</b> .....	<b>324</b>
<b>Appendix K: Evaluation sheets for functions – Cycle 4</b> .....	<b>329</b>
<b>Appendix L: A switch button to Java code – Cycle 5</b> .....	<b>335</b>
<b>Appendix M: Transition results– Cycle 5</b> .....	<b>336</b>

# List of Figures

## Chapter 2:

<b>Figure 2.11:</b> Alice program .....	38
<b>Figure 2.12:</b> UUhistle program .....	41
<b>Figure 2.3:</b> ARAWEs animation program .....	42

## Chapter 3:

<b>Figure 3.1:</b> CTML process .....	53
<b>Figure 3.2:</b> Components of a conversational framework .....	55
<b>Figure 3.3:</b> Bloom's taxonomy.....	57
<b>Figure 3.4:</b> Revised Bloom's taxonomy .....	57
<b>Figure 3.5:</b> Constructive alignment framework.....	60
<b>Figure 3.6:</b> TLPAP applicable theoretical components .....	62
<b>Figure 3.7:</b> TLPAP adapted constructive alignment .....	63
<b>Figure 3.8:</b> Computation as interaction illustration.....	66

## Chapter 4:

<b>Figure 4.1:</b> Action research cycles .....	80
<b>Figure 4.2:</b> A process for both experimental and control groups.....	84

## Chapter 5:

<b>Figure 5.1:</b> Sample cups .....	98
<b>Figure 5.2:</b> Components of the manipulative .....	99
<b>Figure 5.3:</b> Program code .....	99
<b>Figure 5.4:</b> Nested-decider .....	99
<b>Figure 5.5:</b> Manipulatives (variables) .....	102
<b>Figure 5.6:</b> Manipulatives (with values) .....	102
<b>Figure 5.7:</b> Manipulatives (three cups) .....	103

## Chapter 6:

<b>Figure 6.1:</b> Pre-teaching exercises.....	110
<b>Figure 6.2:</b> Program 1 (Demo 1).....	111
<b>Figure 6.3:</b> Program 1 (Demo 2).....	111
<b>Figure 6.4:</b> Program 1 (Demo 3).....	112
<b>Figure 6.5:</b> Program 1 (Demo 4).....	112
<b>Figure 6.6:</b> Program 2 (Demo 5).....	112
<b>Figure 6.7:</b> Program 2 (Demo 1).....	113
<b>Figure 6.8:</b> Program 2 (Demo 2).....	113
<b>Figure 6.9:</b> Program 2 (Demo 3).....	113
<b>Figure 6.10:</b> Program 2 (Demo 4).....	113
<b>Figure 6.11:</b> Program 2 (Demo 5).....	114
<b>Figure 6.12:</b> Program 2 (Demo 6).....	114
<b>Figure 6.13:</b> Program 3 (Demo 1).....	114

<b>Figure 6.14:</b> Program 3 (Demo 2).....	114
<b>Figure 6.15:</b> Program 4 (Demo 1).....	115
<b>Figure 6.16:</b> Program 4 (Demo 2).....	115
<b>Figure 6.17:</b> Program 4 (Demo 3).....	115
<b>Figure 6.18:</b> Program 4 (Demo 4).....	115
<b>Figure 6.19:</b> Program 4 (Demo 5).....	116
<b>Figure 6.20:</b> Program 4 (Demo 6).....	116
<b>Figure 6.21:</b> Program 4 (Demo 7).....	116
<b>Figure 6.22:</b> Problem specification (decisions/nested decisions).....	117
<b>Figure 6.23:</b> Nested-decider (Demo 1) .....	118
<b>Figure 6.24:</b> Nested-decider (Demo 2) .....	118
<b>Figure 6.25:</b> Nested-decider (Demo 3) .....	119
<b>Figure 6.26:</b> Nested-decider (Demo 4) .....	119
<b>Figure 6.27:</b> Nested-decider (Demo 5) .....	120
<b>Figure 6.28:</b> Nested-decider (Demo 6) .....	121
<b>Figure 6.29:</b> Nested-decider (Demo 7) .....	121
<b>Figure 6.30:</b> Nested-decider (Demo 8) .....	122
<b>Figure 6.31:</b> Problem specifications – loops .....	122
<b>Figure 6.32:</b> Loops (Demo 1) .....	123
<b>Figure 6.33:</b> Loops (Demo 2) .....	123
<b>Figure 6.34:</b> Loops (Demo 3) .....	124
<b>Figure 6.35:</b> Loops (Demo 4) .....	125
<b>Figure 6.36:</b> Loops (Demo 5) .....	125
<b>Figure 6.37:</b> Pedagogical guidelines .....	126
<b>Figure 6.38:</b> Post-teaching exercises (Assignment) .....	129
<b>Figure 6.39:</b> Post-teaching exercise (decisions/nested decisions) .....	131
<b>Figure 6.40:</b> Post-teaching exercise (loops) .....	132

*Chapter 7:*

<b>Figure 7.1:</b> Simple if-statement (Demo 1) .....	143
<b>Figure 7.2:</b> Simple if-statement (Demo 2) .....	143
<b>Figure 7.3:</b> Simple if-statement (Demo 3) .....	144
<b>Figure 7.4:</b> Simple if-statement (Demo 4) .....	145
<b>Figure 7.5:</b> Simple if-statement (Demo 5) .....	145
<b>Figure 7.6:</b> Simple if-statement (Demo 6) .....	146
<b>Figure 7.7:</b> An if-else statement (Demo 1) .....	147
<b>Figure 7.8:</b> An if-else statement (Demo 2) .....	147
<b>Figure 7.9:</b> An if-else statement (Demo 3) .....	148
<b>Figure 7.10:</b> Nested-if statement (Demo 1).....	148
<b>Figure 7.11:</b> Nested-if statement (Demo 2).....	149
<b>Figure 7.12:</b> For-loop (Demo 1) .....	150
<b>Figure 7.13:</b> For-loop (Demo 2) .....	150
<b>Figure 7.14:</b> For-loop (Demo 3) .....	151
<b>Figure 7.15:</b> For-loop (Demo 4) .....	151
<b>Figure 7.16:</b> For-loop (Demo 5) .....	152
<b>Figure 7.17:</b> For-loop (Demo 6) .....	152
<b>Figure 7.18:</b> While-loop (Demo 1) .....	153

<b>Figure 7.19:</b> Do while-loop (Demo 2).....	153
<b>Figure 7.20:</b> Nested-loop (Demo 1).....	154
<b>Figure 7.21:</b> Nested-loop (Demo 2).....	155
<b>Figure 7.22:</b> Nested-loop (Demo 3).....	156
<b>Figure 7.23:</b> Nested-loop (Demo 4).....	156
<b>Figure 7.24:</b> Arrays (Demo 1).....	157
<b>Figure 7.25:</b> Arrays (Demo 2).....	157
<b>Figure 7.26:</b> Arrays (Demo 3).....	158
<b>Figure 7.27:</b> Arrays (Demo 4).....	158
<b>Figure 7.28:</b> Value-returning function (Demo 1).....	159
<b>Figure 7.29:</b> Value-returning function (Demo 2).....	160
<b>Figure 7.30:</b> Value-returning function (Demo 3).....	160
<b>Figure 7.31:</b> Non-value-returning function (Demo 1).....	161
<b>Figure 7.32:</b> Non-value-returning function (Demo 2).....	161
<b>Figure 7.33:</b> Value parameters (Demo 1).....	162
<b>Figure 7.34:</b> Value parameters (Demo 2).....	163
<b>Figure 7.35:</b> Value parameters (Demo 3).....	163
<b>Figure 7.36:</b> Value parameters (Demo 4).....	164
<b>Figure 7.37:</b> Reference parameters (Demo 1).....	164
<b>Figure 7.38:</b> Reference parameters (Demo 2).....	165
<b>Figure 7.39:</b> Reference parameters (Demo 3).....	165
<b>Figure 7.40:</b> Problem specification – loops.....	166
<b>Figure 7.41:</b> Loops (Demo 1).....	166
<b>Figure 7.42:</b> Loops (Demo 2).....	167
<b>Figure 7.43:</b> Loops (Demo 3).....	167
<b>Figure 7.44:</b> Loops (Demo 4).....	168
<b>Figure 7.45:</b> Loops (Demo 5).....	168
<b>Figure 7.46:</b> Nested loop problem specifications.....	169
<b>Figure 7.47:</b> Nested-loop (Demo 1).....	170
<b>Figure 7.48:</b> Nested-loop (Demo 2).....	170
<b>Figure 7.49:</b> Nested-loop (Demo 3).....	171
<b>Figure 7.50:</b> Problem specifications – arrays.....	171
<b>Figure 7.51:</b> Arrays (Demo 1).....	172
<b>Figure 7.52:</b> Arrays (Demo 2).....	172
<b>Figure 7.53:</b> Arrays (Demo 3).....	173
<b>Figure 7.54:</b> Arrays (Demo 4).....	173
<b>Figure 7.55:</b> Parallel array problem specification.....	173
<b>Figure 7.56:</b> Parallel arrays (Demo 1).....	174
<b>Figure 7.57:</b> Parallel arrays (Demo 2).....	174
<b>Figure 7.58:</b> Parallel arrays (Demo 3).....	175
<b>Figure 7.59:</b> Parallel arrays (Demo 4).....	175
<b>Figure 7.60:</b> Parallel arrays (Demo 4).....	176
<b>Figure 7.61:</b> Parallel arrays (Demo 5).....	176
<b>Figure 7.62:</b> Problem specification – functions.....	177
<b>Figure 7.63:</b> Functions (Demo 1).....	177
<b>Figure 7.64:</b> Functions (Demo 2).....	178
<b>Figure 7.65:</b> Functions (Demo 3).....	178
<b>Figure 7.66:</b> Functions (Demo 4).....	179

<b>Figure 7.67:</b> Functions (Demo 5).....	179
<b>Figure 7.68:</b> Functions (Demo 6).....	180
<b>Figure 7.69:</b> Functions (Demo 7).....	180
<b>Figure 7.70:</b> Functions (Demo 8).....	181
<b>Figure 7.71:</b> Adapted pedagogical guidelines .....	181
<b>Figure 7.72:</b> Pre-teaching algorithm sections on decisions.....	187
<b>Figure 7.73:</b> Post-teaching algorithm sections on decisions .....	187
<b>Figure 7.74:</b> Pre-teaching algorithm sections on loops.....	192
<b>Figure 7.75:</b> Post-teaching algorithm sections on loop .....	192
<b>Figure 7.76:</b> Pre-teaching algorithm sections on nested loops.....	195
<b>Figure 7.77:</b> Problem specification for post-teaching algorithm on nested loops.....	195
<b>Figure 7.78:</b> Post-teaching algorithm sections on nested loops .....	195
<b>Figure 7.79:</b> Pre-teaching algorithm sections on arrays .....	198
<b>Figure 7.80:</b> Problem specification for post-teaching algorithm on one-dimensional array.....	198
<b>Figure 7.81:</b> Post-teaching algorithm sections on one-dimensional array .....	198
<b>Figure 7.82:</b> Pre-teaching algorithm sections on parallel arrays .....	201
<b>Figure 7.83:</b> Problem specification for post-teaching algorithm on parallel arrays .....	201
<b>Figure 7.84:</b> Post-teaching algorithm sections on parallel arrays.....	201
<b>Figure 7.85:</b> Pre-teaching algorithm sections on functions.....	203
<b>Figure 7.86:</b> Problem specification for post-teaching algorithm on functions .....	204
<b>Figure 7.87:</b> Post-teaching algorithm sections on functions.....	205

Chapter 8:

<b>Figure 8.1:</b> Problem specification – decisions and loops.....	216
<b>Figure 8.2:</b> Decisions and loops (Demo 1).....	217
<b>Figure 8.3:</b> Decisions and loops (Demo 2).....	217
<b>Figure 8.4:</b> Decisions and loops (Demo 3).....	218
<b>Figure 8.5:</b> Decisions and loops (Demo 4).....	218
<b>Figure 8.6:</b> Decisions and loops (Demo 5).....	219
<b>Figure 8.7:</b> Decisions and loops (Demo 6).....	219
<b>Figure 8.8:</b> Arrays (Demo 1).....	220
<b>Figure 8.9:</b> Arrays (Demo 2).....	220
<b>Figure 8.10:</b> Arrays (Demo 3).....	221
<b>Figure 8.11:</b> Arrays (Demo 4).....	221
<b>Figure 8.12:</b> Arrays (Demo 5).....	222
<b>Figure 8.13:</b> Problem specification for post-teaching algorithm on decisions/nested decisions.....	224
<b>Figure 8.14:</b> Post-teaching algorithm sections on decisions/nested decisions .....	224
<b>Figure 8.15:</b> Problem specification for post-teaching algorithm on loops/decisions.....	227
<b>Figure 8.16:</b> Pre-teaching algorithm sections on loops/decisions .....	228
<b>Figure 8.17:</b> Post-teaching algorithm sections on loops/decisions .....	228
<b>Figure 8.18:</b> Generic framework .....	242

# List of Tables

## Chapter 2:

<b>Table 2.1:</b> Introductory programming misconceptions.....	30
--	----

## Chapter 3:

<b>Table 3.1:</b> Piaget's cognitive stages .....	49
<b>Table 3.2:</b> Conversational framework teaching media.....	55
<b>Table 3.3:</b> SOLO taxonomy .....	58
<b>Table 3.4:</b> SOLO and Whalley et al. (2006) programming code benchmarking .....	64

## Chapter 4:

<b>Table 4.1:</b> Adapted SOLO by Lister et al. (2006) and Izu, Weerasinghe and Pope (2016)...	85
<b>Table 4.2:</b> SOLO-based evaluation matrix .....	85
<b>Table 4.3:</b> SOLO-based evaluation matrix (with description) .....	86
<b>Table 4.4:</b> Methodological overview .....	93

## Chapter 5:

<b>Table 5.1:</b> Average grades for control and experimental groups .....	103
--	-----

## Chapter 6:

<b>Table 6.1:</b> Results (Assignment) .....	130
<b>Table 6.2:</b> Summary of errors with the algorithms (decisions/nested decisions) .....	132
<b>Table 6.3:</b> Summary of errors with the algorithms (loops) .....	132
<b>Table 6.4:</b> Evaluation sheet (control group on decisions/nested decisions) .....	133
<b>Table 6.5:</b> Evaluation sheet (experimental group on decisions/nested decisions) .....	134
<b>Table 6.6:</b> Evaluation matrix .....	135
<b>Table 6.7:</b> Evaluation sheet (control group on loops) .....	135
<b>Table 6.8:</b> Evaluation sheet (experimental group on loops) .....	136
<b>Table 6.9:</b> Evaluation matrix .....	137

## Chapter 7:

<b>Table 7.1:</b> Results (Assignment) .....	185
<b>Table 7.2:</b> Paired algorithm section (decisions/nested decisions) .....	188
<b>Table 7.3:</b> Average transitions performance on decisions/nested decisions over all pairs..	189
<b>Table 7.4:</b> Paired algorithm sections - loop .....	193
<b>Table 7.5:</b> Average transitions performance on loops over all pairs .....	193
<b>Table 7.6:</b> Pairs algorithm sections – nested loops.....	196
<b>Table 7.7:</b> Average transitions performance on nested loops across pairs .....	196
<b>Table 7.8:</b> Paired algorithm sections – one-dimensional array .....	199
<b>Table 7.9:</b> Average transitions performance on one-dimensional arrays for all pairs.....	199
<b>Table 7.10:</b> Transitions performance on parallel arrays .....	202

<b>Table 7.11:</b> Pairs sections/columns names for functions .....	206
<b>Table 7.12:</b> Average transitions performance on functions for all pairs .....	207

*Chapter 8:*

<b>Table 8.1:</b> Results (Assignment) .....	223
<b>Table 8.2:</b> Paired algorithm section (decisions/nested decisions) .....	225
<b>Table 8.3:</b> Average transitions performance on decisions/nested decisions for all pairs.....	226
<b>Table 8.4:</b> Pairs algorithm sections – nested loops.....	229
<b>Table 8.5:</b> Average transitions performance on decisions/loops for all pairs .....	230
<b>Table 8.6:</b> Transitions performance on parallel arrays .....	232
<b>Table 8.7:</b> Average transitions performance on functions for all pairs .....	234

# Abstract

Introductory programming can be challenging for educators to teach and students to learn. Some students struggle to comprehend introductory programming topics like the uses of *assignment*, *decisions*, *loops*, *functions* and other basic principles of programming. Such struggling students are referred to as Struggling Introductory Programming Students (SIPS) in this study. The struggle to comprehend these basic programming concepts often results in a low success rate and a low pass rate in the module itself. Furthermore, the situation can exacerbate the attrition rate in computing or computing-related courses.

In this study, various methods of teaching and learning to program were investigated. It was found that the use of animations in teaching and learning can have a significant impact on enhancing SIPS comprehension of introductory programming concepts. As a result, we developed a Teaching and Learning Programming with Animation Programs (TLPAP) framework that can aid SIPS in comprehending the basic programming principles in a text-based programming environment. TLPAP consists of two sets of animation programs, namely introductory animation programs for introducing a new concept and animation programs that are accompanied by problem specifications. Furthermore, the TLPAP framework includes a set of compatible pedagogical guidelines for teaching and learning introductory programming with animation programs. The inclusion of the pedagogical guidelines serves as one of the essential elements within the TLPAP framework.

To test the efficacy of the TLPAP framework, we adopted an action research methodology with the actual SIPS from the university participating. The action research methodology made it possible for the researcher to reflect and continually improve the design of the animation programs and pedagogical guidelines within action cycles. The SIPS were divided into two groups: an experimental and a control group. The experimental group was taught by following the TLPAP framework; the control group was taught without following the TLPAP framework. The results were compared through pre-teaching and post-teaching algorithm exercises and the application of qualitative Structure of Observed Learning Outcomes – adapted (SOLO-



adapted) evaluations. During the cycles, patterns of improvements were found in the experimental group as compared to the control group and the TLPAP framework was finalised.

**Key terms:** Teaching, learning, pedagogical guidelines, computer science education, introductory programming, animation programs, struggling students, SOLO.

# Summary

## **An Animation-Based Teaching and Learning Framework for Struggling Introductory Programming Students**

In this study, we investigated the best possible methods for teaching and learning introductory programming concepts (IPC) through an action research methodology. The plan was, specifically, to develop a teaching and learning framework for struggling introductory programming students (SIPS). The use of physical manipulatives, as aiding tools to teach and learn introductory programming, was attempted in the first cycle (2019, semester 2, and 2020, semester 1) of action research. We found that the use of physical manipulatives can aid teaching and learning IPC to a certain extent, but it was limited in other crucial areas, like covering a large class, visibility of fine details to all students, and functioning on online platforms. As a result, in the second cycle (2021, semester 1) of action research, we initiated a framework, named Teaching and Learning Programming with Animation Programs (TLPAP). TLPAP consisted of animated versions of manipulatives and relevant pedagogical guidelines. The pedagogical guidelines are a set of stages or guidelines for the best use of the animation programs.

Through a SOLO (Structure of Observed Learning Outcome) adapted evaluation, we experimented with the animation programs on the actual SIPS, at the university. SIPS were divided into two groups, that is, the control and experimental groups. The control group was taught traditionally through verbal narration, and the experimental group was taught by following TLPAP. The results of cycle 2 helped us to refine the TLPAP framework for cycle 3. As a result, cycle 3 (2021, semester 2) was implemented with improved TLPAP and SOLO-adapted evaluation. The subsequent cycles (cycles 4 and 5 in 2022, semesters 1 and 2) were implemented with even more improved TLPAP. The SOLO evaluation indicated more improvements in the experimental group than in the control group. The conclusion of cycle 5 presented a final TLPAP.

# Summary (Afrikaans language)

## **'n Animasie-gebaseerde Onderrig- en Leerraamwerk vir sukkelende Inleidende Programmeringstudente**

In hierdie studie, het ons die beste moontlike metodes vir die onderrig en leer van inleidende programmeringskonsepte (IPK) deur middel van 'n aksienavorsingsmetodologie ondersoek. Die plan was spesifiek om 'n onderrig- en leerraamwerk vir sukkelende inleidende programmeringstudente (SIPS) te ontwikkel. Die gebruik van fisiese manipulasies as hulpmiddels om inleidende programmering te onderrig en te leer, is in die eerste siklus (2019, semester 2, en 2020, semester 1) van aksienavorsing ingespan. Ons het gevind dat die gebruik van fisiese manipulasies tot 'n sekere mate kan help in die onderrig en leer van IPK, maar dat dit beperk is op ander belangrike gebiede, soos om 'n groot klas te dek, sigbaarheid van fyn besonderhede vir alle studente en funksionering op aanlyn platforms. Gevolglik het ons in die tweede siklus (2021, semester 1) van aksienavorsing 'n raamwerk geïnisieer, naamlik die Program vir Onderrig en Leer van Programmering met Animasie (TLPAP). TLPAP het bestaan uit geanimeerde weergawes van manipulatiewe en relevante pedagogiese riglyne. Die pedagogiese riglyne is 'n stel stadiums of riglyne vir die beste gebruik van die animasieprogramme.

Deur 'n SOLU- (struktuur van waargenome leeruitkoms)- aangepaste evaluering het ons by die universiteit met die animasieprogramme op die werklike SIPS geëksperimenteer. Die SIPS is in twee groepe verdeel, naamlik die kontrole- en eksperimentele groepe. Die kontrolegroep is tradisioneel deur verbale vertelling onderrig, en die eksperimentele groep is onderrig deur TLPAP te volg. Die resultate van siklus 2 het ons gehelp om die TLPAP-raamwerk vir siklus 3 te verfyn. Gevolglik is siklus 3 (2021, semester 2) geïmplementeer met verbeterde TLPAP- en SOLU-aangepaste evaluering. Die daaropvolgende siklusse (siklusse 4 en 5 in 2022, semesters 1 en 2) is met selfs beter TLPAP geïmplementeer. Die SOLU-evaluering het meer verbeterings in die eksperimentele as in die kontrolegroep aangedui. Die afsluiting van siklus 5 het 'n finale TLPAP aangebied.

# Summary (Sepedi / Northern Sotho language)

## **Foreimiweke ya go Ruta le go Ithuta ye e Theilwego godimo ga di Animeišene ya Baithuti bao ba nago le Mathata a Tlhamo ya Mananeo a Tsamaišo ya Khomphutha**

Mo nyakišišong ye, re nyakišišitše mekgwa ye mekaone ye e kgonegago ya go ruta le go ithuta dikgopolo tša tlhamo ya mananeo a khomphuthara (IPC) ka mokgwa wa nyakišišo ya tharollo ya mathata thutong. Leano, gagolo, ebile la go hlama foreimiweke ya go ruta le go ithuta ya baithuti bao ba nago le mathata a tlhamo ya mananeo a tsamaišo ya khomphutha (SIPS). Tšhomišo ya dilo tšeo di ka swarwago bjalo ka didirišwa tša go thuša go ruta le go ithuta tlhamo ya mananeo a tsamaišo ya khomphutha, e lekilwe modikologong wa mathomo (2019, semesetara sa 2, le 2020, semesetara sa 1) wa nyakišišo ya tharollo ya mathata thutong. Re hweditše gore tšhomišo ya dilo tšeo di ka swarwago e ka thuša go ruta le go ithuta IPC go fihla bokgoleng bjo bo itšego, eupša e be e na le magomo ka go dikarolo tše dingwe tše bohlokwa, go swana le go akaretša phapoši ye kgolo, ponagalo ya dikarolwana tše nnyane go baithuti ka moka, le go šoma ga diforamo tša inthaneteng. Bjalo, mo modikologong wa bobedi (2021, semesetara 1) wa nyakišišo ya tharollo ya mathata thutong, re thomile foreimiweke, yeo e bitšwago Tlhamo ya Mananeo a go Ruta le go Ithuta ka dianimeišene (TLPAP). TLPAP e bopilwe ke diphetolelo tša dianimeišene tša dithušapalelo le ditlhahli tša maleba tša thuto. Ditlhahli tša thuto ke sehlopha sa dikgato goba ditlhahli tša tšhomišo ye kaone ya mananeo a animeišene.

Ka tekolo ye e fetoletšwego SOLO (Structure of Observed Learning Outcome), re dirile diteko tša mananeo a dianimeišene go SIPS ya nnete, yunibesithing. SIPS di arotšwe ka dihlopha tše pedi, e lego, dihlopha tša taolo le tša diteko. Sehlopha sa taolo se rutilwe ka mokgwa wa setlwaedi kanegelo ya molomo, gomme sehlopha sa diteko se rutilwe ka go latela TLPAP. Dipelo tša modikologo wa 2 di re thušitše go kaonafatša foreimiweke ya TLPAP ya modikologo wa 3. Bjalo, modikologo wa 3 (2021, semesetara sa 2) o phethagaditšwe ka tekolo ye e kaonafaditšwego ya TLPAP le ye e fetoletšwego SOLO. Modikologo ye e latelago (modikologo ye 4 le 5 ka 2022, disemesetara tša 1 le 2) di phethagaditšwe ka TLPAP ye e kaonafetšego le go feta. Tekolo ya SOLO e bontšhitše dikaonafatšo tše ntši sehlopheng sa diteko go feta sehlopheng sa taolo. Sephetho sa modikologo wa 5 se tšweleditše TLPAP ya mafelelo.

# List of publications

## Conference publications from this research

1. Ramabu, T. J., Sanders, I., & Schoeman, M. (2021, May). Teaching and Learning CS1 with an Assist of Manipulatives. In *2021 IST-Africa Conference (IST-Africa)* (pp. 1-8). IEEE.
2. Ramabu, T., Sanders, I., & Schoeman, M. A. (2021). Manipulatives for Teaching Introductory Programming to Struggling Students: A Case of Nested-decisions. In *CSEDU (1)* (pp. 505-510).

## Journal publications from this research

1. Ramabu, T., Sanders, I., Schoeman, M. (2022). Nested-Decider: An Animation Program for Aiding Teaching and Learning of Decisions/Nested Decisions. In: Barnett, R.J., le Roux, D.B., Parry, D.A., Watson, B.W. (eds) *ICT Education. SACLA 2022. Communications in Computer and Information Science*, vol 1664. Springer, Cham. [https://doi.org/10.1007/978-3-031-21076-1\\_8](https://doi.org/10.1007/978-3-031-21076-1_8).

## List of abbreviations and acronyms

APS	Admission Points Score
CA	Constructive Alignment
CS	Computer Science
CTML	Cognitive Theory on Multimedia Learning
HEI	Higher Education Institution
ICT	Information and Communication Technology
IPC	Introductory Programming Concepts
SA	South Africa
SIPS	Struggling Introductory Programming Students
SOLO	Structure of Observed Learning Outcomes
TLPAP	Teaching and Learning Programming with Animation Programs
TLPM	Teaching and Learning Programming with Manipulatives
TUT	Tshwane University of Technology
LMS	Learning Management System

## Definition of key terms

**Animation programs:** Means computer graphics that animate or make a graphical representation of a programming code to enhance comprehension (Moreno, Sutinen & Joy, 2014).

**Introductory programming concepts:** These are introductory programming concepts in the procedural programming paradigm like decisions, loops and arrays.

**Introductory programming students:** This is a reference to the students who enrolled in an introductory programming course. Sometimes they are referred as novice programmers or Computer Science 1 (CS1) students.

**Learners:** Is a reference to basic education pupils.

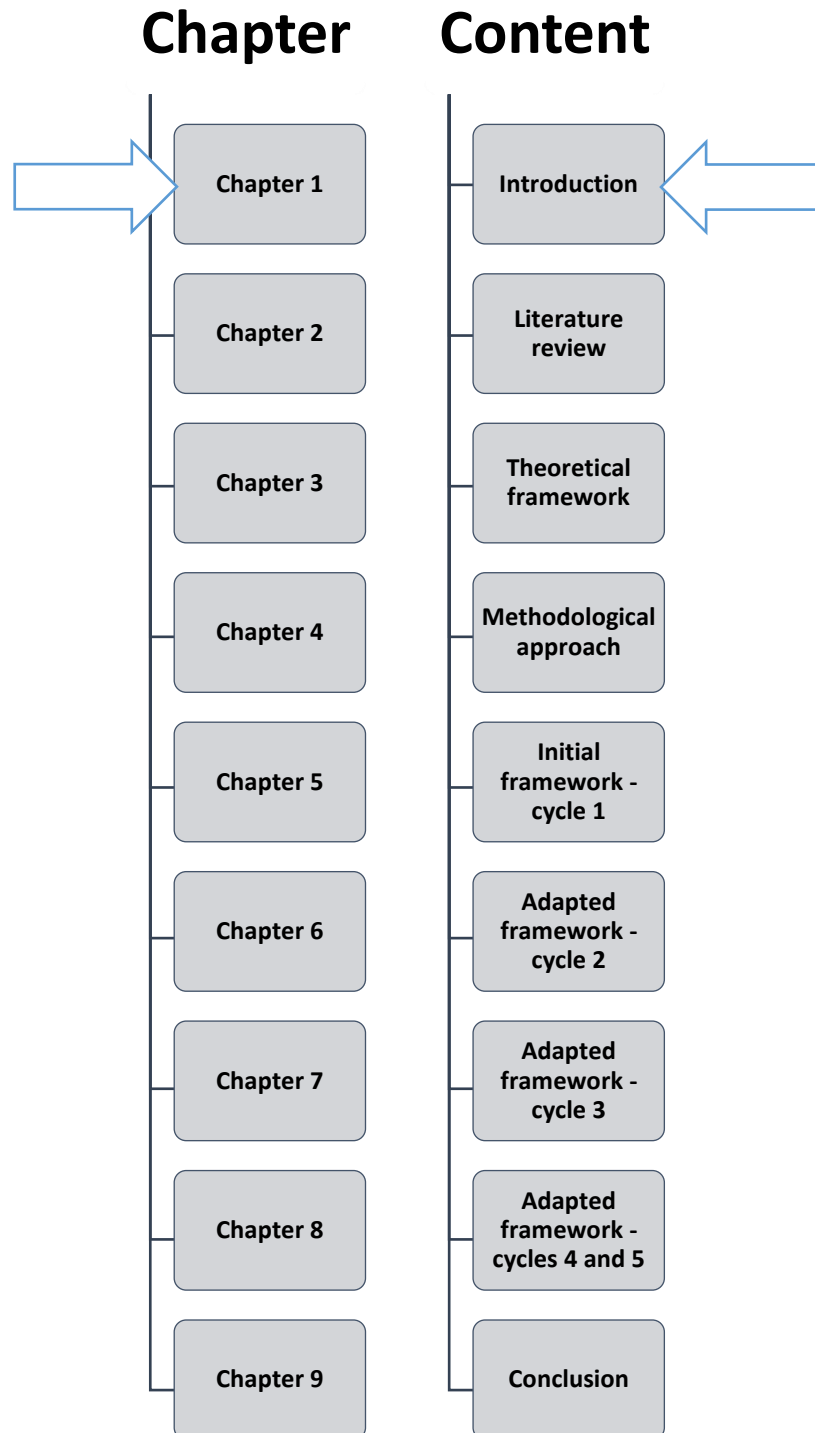
**Objects-later:** This means that procedural programming concepts are presented first, and object-oriented programming concepts are introduced later in the book (introductory programming textbook) or in the curriculum.

**Struggling introductory programming students:** These are students who struggle to comprehend introductory programming concepts.

**Students:** Is a reference to higher education pupils

**Text-based programming:** This means a programming method that involves typing of lines of code to develop an algorithm.

# Chapter 1: Introduction





## 1.1 Introduction

This study investigated various teaching and learning approaches deemed relevant to Struggling Introductory Programming Students (SIPS). The intention was to search, find and improve the potential methods of teaching and learning introductory programming. The relevant teaching and learning methods were subsequently experimented through various cycles of action research methodology. The action research took place at a Higher Education Institution (HEI) with the actual SIPS. At the end of the last cycle (cycle 5), the Teaching and Learning Programming with Animation Programs (TLPAP) framework was validated and finalised. The TLPAP framework has proven to be effective primarily for SIPS and can also be adopted for use as a general framework for non-SIPS.

To get an overview of the context of this study, section 1.2 discusses the background to the study; section 1.3 presents the problem statement; section 1.4 outlines the research questions, objectives and methodology; section 1.5 discusses significance of the study; section 1.6 gives chapter outlines and section 1.7 concludes the chapter.

## 1.2 Background to the study

### *1.2.1 Struggling Introductory Programming Students*

Learning to program can be a challenging task or too abstract for some students (Shuhidan, 2012; Tuparov, Tuparova & Tsarnakova, 2012; Konecki, Lovrenčić & Kaniški, 2016). As a result, computing courses or computing-related courses experience some SIPS. The researcher has been teaching text-based introductory programming and advanced programming since the year 2010 at a South African (SA) university of technology. Therefore, besides the challenges of teaching and learning to program identified in the literature review (discussed in Chapter 2), I have come to know and personally experience SIPS and further have attempted to assist them in the lecture halls and consultation sessions.

Some of the problems I have experienced while assisting SIPS were basic programming principles under the object-later programming curriculum. Object-later programming advocates for teaching and learning of procedural concepts before

Object-Oriented Programming (OOP) concepts can be taught (Ehlert & Schulte, 2009). In my introductory programming class, I came across SIPS that has challenges like incorrect steps in a program code, misunderstanding of the *assignment* operator, ambiguous *decision statements*, inappropriate iterations structures, misunderstanding of arrays and incorrect *function* references and *function* values. At the end, normal pedagogical interventions did not yield positive outcomes. I have also come to experience that the blanket pedagogical approach can only favour non-SIPS, leaving SIPS as the missing middle.

### *1.2.2 Teaching and learning of introductory programming concepts*

Programming is one of the subjects that forms part of a university's computing or computing-related courses. It is also one of the complex major and essential subjects in computing courses, therefore methods of teaching and learning to program ought to take centre stage (Krishnamurthi & Fislser, 2019; Cheah, 2020; Prasad & Chaudhary, 2021). Bart and Shaffer (2016:1) state that "Instructional design is to teaching as software engineering is to programming", implying that one cannot just teach randomly, but proper instructional design can enable smooth teaching and learning in a specific course. Sometimes teaching and learning can have a positive impact on the success or failure of students. This is because teaching is the first activity in contact courses; therefore, it can reduce cognitive load and boost success in comprehension of Introductory Programming Concepts (IPC).

The most common IPC within the late-objects pedagogical approach include variables, *assignment*, data types, strings, arrays, conditional control structures, iterative control structures, functions, methods, procedures and parameter passing (Luxton-Reilly et al., 2018). Some of these concepts are also mentioned in the joint task force and computing curricula (ACM/IEEE-CS Joint Task Force on Computing Curricula, 2013; Kumar & Raj, 2022). In the light of challenges of comprehending these IPC, it is necessary to identify, argue and revolutionise the teaching and learning methods for SIPS. In an attempt to arrive to a relevant solution, an action research methodology was used to carry out the investigation and experimentation towards the formation and finalisation of such a teaching framework.

### *1.2.3 Action research*

The study applied an action research methodology because the solution is in the context of an educational setup. Action research in education allows for a reflective process at the end of each cycle with the aim to revise and improve the experimented teaching and learning methods (Creswell et al., 2007). Action research in the education allows the application of theory and practice (Kearney, Wood & Zuber-Skerritt, 2013), hence the methodology is found to be appropriate for the implementation and continual improvement of the teaching methods and tools. Action research methodology consists of four important phases, that is plan, act, observe and reflect. The process of going through all four phases is referred as action cycles. The researchers can go through a number of action research cycles before the study can be concluded or finalised.

### *1.2.4 Animation programs*

The world of animations has great potential to help in teaching and learning IPC (further explored in Chapter 2, section 2.7). Animation programs make teaching attractive, interesting and encourage students to learn (Weiss, Knowlton & Morrison, 2002; Cevahir, Özdemir & Baturay, 2022). The use of animation programs takes advantage of the increasing connectivity and availability of digital platforms. In the world of teaching and learning to program, animation programs are computer graphics that animate or make a graphical manipulation of a program code with the idea of enhancing comprehension (Moreno, Sutinen & Joy, 2014). Through an action research process, this has realised and uncovered the potential uses of animation programs. As a result, I developed relevant animation programs for teaching and learning IPC specifically for SIPS.

## **1.3 Problem statement**

Text-based programming competency requires full attention to details and syntax which is a challenge for some students (Federici, 2011). As a result, some introductory programming students in Computer Science (CS) or computing-related courses find IPC too complex or challenging to comprehend (Tuparov et al., 2012). Several issues experienced by introductory programming students include misconceptions about

assignment, controls, decisions, tracing, functions, parameters, initialisation and references (Sajaniemi & Kuittinen, 2008; Schoeman, Gelderblom & Muller, 2013; Brown & Altadmri, 2014; Altadmri & Brown, 2015; Islam et al., 2019). The typical consequences of these are primarily poor performance, drop out or academic exclusion (Gomes & Mendes, 2007; Rubio et al., 2015; Brown & Altadmri, 2015; Grover & Basu, 2017, Ahadi et al., 2018; Luxton-Reilly et al., 2018; Cetin, 2020; Cheah, 2020; Prasad & Chaudhary, 2021).

In an effort to alleviate these problems, various pedagogical solutions are adopted and applied in teaching and learning introductory programming (discussed in the literature review section – Chapter 2), but the problems remain unsolved (Cheah, 2020). Among existing pedagogical approaches, animation programs are used to aid the teaching and learning of introductory programming. This is because animation programs are informative, explanatory, explicit, interactive, effective, clearer and consequently can be a learning attraction or motivation for students (Lowe, 2001; Lowe & Schnotz, 2008; Cevahir, Özdemir & Baturay, 2022). However, existing animation programs for teaching and learning introductory programming have these drawbacks:

- They typically contain overloaded graphics and are poorly designed (Lowe & Schnotz, 2008). This can result in additional cognitive load away from the actual learning of programming and can lead to confusion and learning setback.
- They lack introductory animations programs for introducing a new concept, which hinders smooth teaching and learning transitions into more challenging stages of the concept.
- They tend to focus on the generic computing concepts in the K12 stream which are incompatible with a university novice programming student in a text-based programming environment.
- They lack dedicated animation-based solutions specifically designed for SIPS in a text-based programming environment.
- They are not grounded in formalised pedagogical guidelines for best design and use of such animation programs for teaching and learning introductory programming. Perhaps the assumption is that educators and students would figure out how to properly use such animation programs for effective use in teaching and learning. This is a problem, as animation programs can cause

more harm than good in teaching due to lack of applicable pedagogical guidelines.

Based on the stated problems, the thesis statement in this study is formulated as follows:

*A Teaching and Learning Programming with Animation Programs (TLPAP) framework can improve Struggling Introductory Programming Students (SIPS) comprehension of Introductory Programming Concepts (IPC).*

The above thesis statement forms the basis of the research questions and objectives in the subsequent section.

## 1.4 Research questions and objectives

### 1.4.1 Research questions

The thesis statement informed the following research questions:

- (a) What are the leading issues in teaching and learning IPC?
- (b) What are the approaches used in teaching IPC?
- (c) What are the limitations of existing animation programs used for teaching and learning IPC?
- (d) How can we develop a TLPAP framework for teaching and learning IPC?
- (e) How can we prove and evaluate the efficacy of a TLPAP framework?

### 1.4.2 Research objectives

To carry out our thesis statement, the following objectives were established:

- (a) To conduct a literature review on issues of teaching and learning IPC;
- (b) To investigate the approaches used in teaching IPC;
- (c) To investigate animation programs for teaching and learning IPC;
- (d) To develop a TLPAP framework for teaching and learning IPC based on gaps identified in the literature review and by continually improving the framework based on the outcome of the action research cycles;
- (e) To evaluate the efficacy of the TLPAP framework through teaching the actual SIPS by following the TLPAP framework.

### *1.4.3 Brief methodology*

The overall methodological approach followed in this study is action research as briefly highlighted in section 1.2.3 and will further be elaborated in the methodology chapter (Chapter 4). The first three research objectives are achieved with the literature review (Chapter 2). The fourth research objective is achieved through the development of TLPAP. The last research objective is achieved by using an evaluation strategy that uses SOLO-adapted evaluations (see a detailed discussion in Chapter 4).

In the first attempt (cycle 1) of the action research, an investigation into the use of manipulatives for teaching and learning IPC was carried out and experimented with. The target framework was referred to as Teaching and Learning Programming with Manipulatives (TLPM). It was found that the use of manipulatives to teach IPC works to a certain extent, as a result that prompted further investigations and improvements. In the second cycle of the action research, the search continued by exploring other possible methods of teaching and learning IPC. An investigation into the potential of using animation programs to improve comprehension of SIPS was conducted. The animation programs were planned, designed and experimented with. The outcome of the experimentation was successful as compared to the unsatisfactory results in cycle 1. The animation programs were continually improved in the subsequent action cycles which led to the finalisation of TLPAP framework in cycle 5. The details and results of each action cycle are presented in Chapters 5, 6, 7 and 8.

## **1.5 Significance of the study**

This study is investigated within the parameters of teaching and learning, as a result the researcher was exposed to many theories of learning available in the body of knowledge. The theoretical contribution of this study lies in the development of a relevant theoretical framework as a backbone and frame for the study. The process of adopting a theoretical framework, was initiated by studying all relevant teaching and learning theories followed by the synthesis of the applicable theories for a possible contribution in this study.

The final TLPAP is expected to directly benefit the computing students in higher education, learners in basic education that study programming as part of the curriculum and anyone who is studying programming on their own. The final TLPAP

can improve the success rate for teaching and learning to program. Furthermore, the TLPAP can be a relevant framework aiding the academic community, practitioners or curriculum developers. In brief, the study has the following potential contributions:

- (a) Development of TLPAP framework specifically for SIPS in the text-based programming environment.
- (b) Animation programs in two sets or phases, namely the special animation programs for introducing a new concept and animation programs with problem specifications.
- (c) Development of specific pedagogical guidelines that form part of TLPAP. The pedagogical guidelines are to be followed when using TLPAP for effective use of animation programs.
- (d) Theoretical contributions consist of relevant theories that may support the use of TLPAP.
- (e) Methodological contributions consist of SOLO-adapted evaluations for the given algorithms.

## 1.6 Chapter outline

*Chapter 2: Literature review.* The main focus in Chapter 2 are the challenges of learning to program, imperative programming, approaches of teaching procedural programming and existing animation programs for IPC.

*Chapter 3: Theoretical framework.* This chapter discusses the underpinning theories that frame this study. Relevant sections of the chapter are learning theories, learning taxonomies, teaching philosophy and constructive alignment.

*Chapter 4: Methodological approach.* The methodology chapter depicts and substantiates relevant research methods.

*Chapter 5: Initial framework - cycle 1.* Reports on the initial plans and experiments which occurred in cycle 1 of action research.

*Chapter 6: Adapted framework - cycle 2.* Reports on the adapted plans and experiments as occurred in cycle 2 of action research.

*Chapter 7: Adapted framework - cycle 3.* Reports on more adaptation and experiments which occurred in cycle 3 of the action research.

*Chapter 8: Adapted framework - cycles 4 and 5.* Reports on cycles 4 and 5 and a final TLPAP framework.

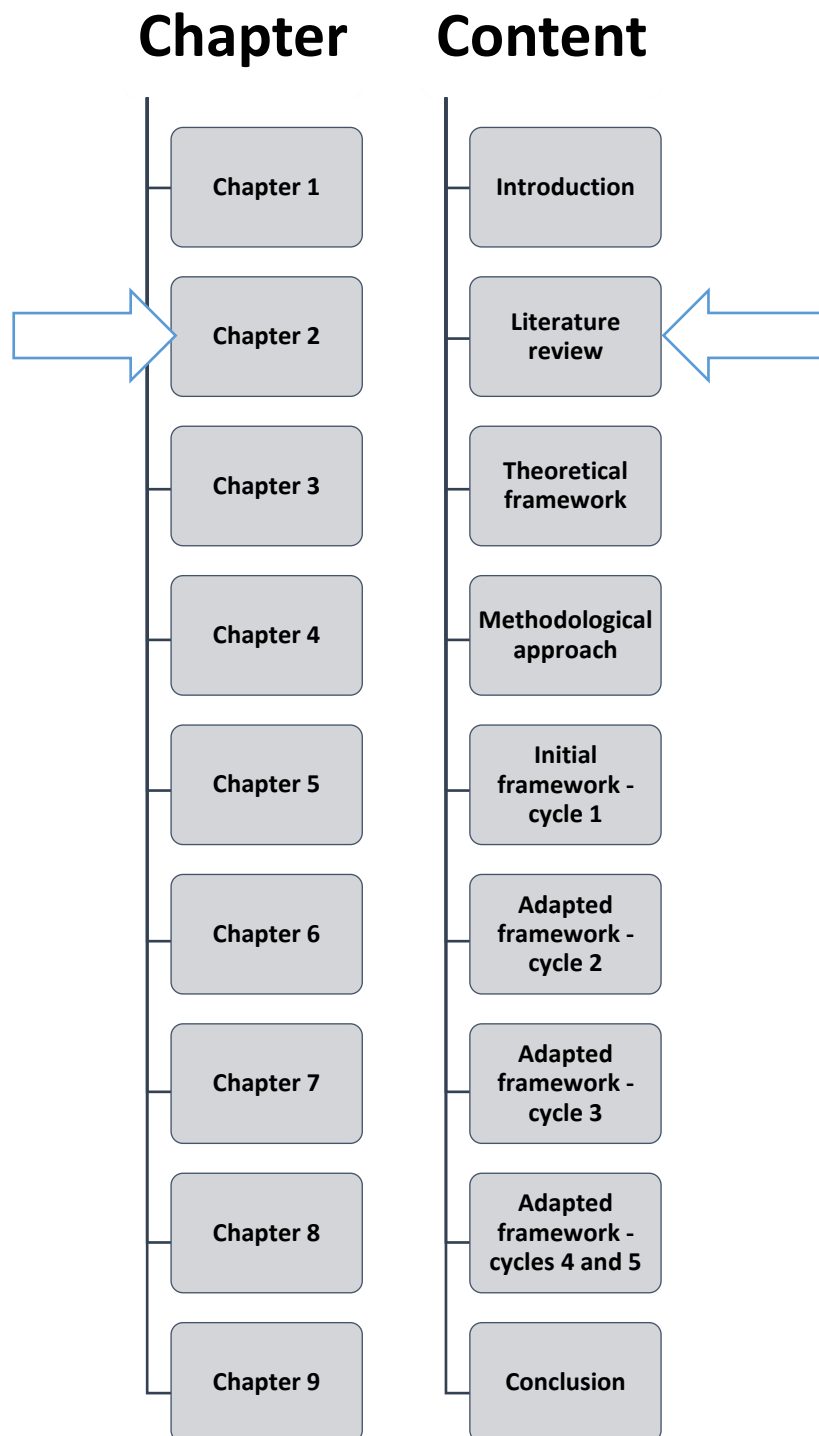
*Chapter 9: Conclusion.* This chapter concludes the study.

## 1.7 Conclusion

The intention of this introductory chapter was to present an overview of the study. The background to the study presented relevant context about the study by giving an overview on SIPS, teaching and learning IPC, action research methodology and animation programs. The problem statement has been addressed and relevant thesis statement has been formulated as: “*A Teaching and Learning Programming with Animation Programs (TLPAP) framework can improve Struggling Introductory Programming Students (SIPS) comprehension of Introductory Programming Concepts (IPC).*” The design, experimentation and finalisation of the TLPAP was subjected to rigorous cycles of action research methodology. The research questions and objectives in this study are carried out sequentially as listed in section 1.4. The first 3 research question and objectives are references to the literature review. A literature review which was conducted prior to the TLPAP realisation, serves as first and foremost guidance and identification of a relevant gap. The literature review is presented in the subsequent chapter (Chapter 2).



# Chapter 2: Literature review



## 2.1 Introduction

The thesis statement of this study says: “*A Teaching and Learning Programming with Animation Programs (TLPAP) framework can improve Struggling Introductory Programming Students (SIPS) comprehension of Introductory Programming Concepts (IPC)*”. In an attempt to present the outcome of this research towards achieving the thesis statement, in Chapter 1, a background to the study, problem statement, research questions and research objectives were presented. The problem is that SIPS find introductory programming concepts difficult to learn. Some of the SIPS’ problems are mostly found within the imperative programming paradigm or late-objects pedagogical approach. The methods of teaching and learning introductory programming have been limited and require further improvements (Rum & Ismail, 2017; Medeiros, Ramalho & Falcão, 2018). In order to address this problem, the study adopts animation-based teaching and learning to improve SIPS’ comprehension of IPC.

To design a suitable solution informed by the fundamental problems and identified gaps, this chapter addresses and presents the literature review based on the three research questions and study objectives. The first three research questions as per section 1.4 in Chapter 1 have been stated as follows:

- (a) What are the leading issues in teaching and learning IPC?
- (b) What are the approaches used in teaching IPC?
- (c) What are the limitations of existing animation programs used for teaching and learning IPC?

The purpose of a literature review is outlined in section 2.2, while section 2.3 provides a contextual overview. To deepen understanding of existing issues, section 2.4 addresses the challenges in learning to program. The imperative programming approach is discussed in section 2.5 to offer perspective of the subject under study. Section 2.6 discusses approaches of teaching procedural programming. Section 2.7 discusses existing animation programs and their limitations. The summary of an identified gap is presented in section 2.8.

## 2.2 Purpose of the literature review

A literature review is a comprehensive investigation of prior research in a specific discipline (Denney & Tewksbury, 2013). A proper literature review should be able to give a reader an overview of what is already known in the area being investigated and what is missing within that discipline (Creswell, 2003). According to Ridley (2008:2), the literature review is “where you identify the theories and previous research which have influenced your choice of the research topic and the methodology you are choosing to adopt”. A well-presented literature review gives the reader an overview of gaps to be filled by discussing weaknesses or limitations of previous studies. Therefore, a literature review compels the authors to acquire comprehensive knowledge about prior studies in the topic under investigation. Moreover, a properly structured literature review gives credibility to the work being presented.

Arguments in the literature review are supported by sources from scholarly empirical articles, journal articles, books, dissertations and occasionally even non-empirical sources (Denney & Tewksbury, 2013). Empirical studies are accumulative; therefore, it is also important for a literature review to have a thorough mix of both older and recently published research.

The literature review in this study consists primarily of research published in the area of computer science education. The specific focus is the key content of programming (imperative programming), teaching methodologies for introductory programming, challenges of learning introductory programming and animation programs for aiding teaching and learning. Google Scholar was mostly used as a search engine to access scholarly indexed literature from various databases. In addition to Google Scholar, the ACM digital library, IEEE digital library, Web of Science, Science Direct and Scopus were directly searched. The search words were “teaching and learning approaches for introductory programming”, “issues of teaching and learning to program”, “challenges in learning to program”, “animation programs”, “animation programs in education”, “animation program for introductory programming”. The subsequent sections in this chapter present the relevant literature review as per the definitions outlined in this section.

## 2.3 Contextual overview

### 2.3.1 *Importance of programming*

Computer programming is one of the major subjects in the Computer Science (CS), information systems or other computing-related courses (Prasad & Chaudhary, 2021). Programming is acknowledged as the core of computing courses and an important 21<sup>st</sup> century skill (Bers, 2019; Krishnamurthi & Fisler, 2019), similarly programming education methods ought to improve and take a leading role.

Computing courses are mostly developed by following the guidelines of the Computing Curriculum 2013 by the joint task force and computing curricula (ACM/IEEE-CS Joint Task Force on Computing Curricula, 2013; Kumar & Raj, 2022). A joint task force on computing curricula was developed by an ad-hoc committee consisting of members from Association of Computing Machinery (ACM) and Institute of Electrical and Electronics Engineers (IEEE) (ACM/IEEE-CS Joint Task Force on Computing Curricula, 2013). The majority of subjects within computing courses are more or less interrelated, overlapping or have emerged from an introductory programming perspective. For example, subjects like algorithms, databases, data structures, software engineering, web computing and scripting rely on the understanding of programming principles and computational thinking ability. This shows the importance of programming and therefore emphasises the need to improve methods of teaching and learning to program. However, the methods in teaching and learning introductory programming have not been adequate to accommodate struggling students (Rum & Ismail, 2017; Medeiros, Ramalho & Falcão, 2018). The challenge is to devise an appropriate method of teaching introductory programming for the struggling students.

### 2.3.2 *Learning to program is difficult*

Certain programming concepts can be too abstract or challenging to learn (Tuparov, Tuparova & Tsarnakova, 2012; Shuhidan, 2012; Konecki, Lovrenčić & Kaniški, 2016). Some novices succeed in programming, while others find it quite difficult to learn (Kelleher & Pausch, 2005). This has led to high failure and attrition rates in computing courses or computing-related courses (Lahtinen et al., 2002; Nikula, Gotel & Kasurinen, 2011; Watson & Li, 2014; Rum & Ismail, 2017; Malik & Coldwell-Neilson, 2017; Prasad & Chaudhary, 2021). While programming can be difficult to learn, it is

likewise difficult to teach (Bennedsen & Caspersen, 2007; Bennedsen & Caspersen, 2019). These challenges continue despite various tools used to assist in teaching and learning to program (Medeiros, Ramalho & Falcão, 2018).

Some of the challenges are found within the concepts of variables, the scope of variables, Booleans, assignment, tracing, decisions, logical operators, comparison operators, loops, arrays, recursion, functions, parameters, initialisation and references (Bayman & Mayer, 1983; Eckerdal & Thuné, 2005; Goldman et al., 2008; Sajaniemi & Kuittinen, 2008; Schoeman, Gelderblom & Muller, 2013; Bruce, 2015; Veerasamy, Souza & Laakso, 2016; Brown & Altadmri, 2017; Grover & Basu, 2017; Luxton-Reilly et al., 2018; Cetin, 2020). Anyango and Suleman (2018), surveyed programming lecturers in Kenya and South Africa (SA). The authors found that students still struggle with recursion, arrays and data types. It is evident that there are continuous issues in the programming education spectrum. Beaubouef and Mason (2005) say another contributing factor for the persistence challenges in learning programming is a lack of algorithm practice due to short semesters or short learning hours. This is further worsened as the students are expected to learn syntax, background theory on programming and problem solving within that short period (Rubio, 2019).

### *2.3.3 Computer Science at universities and K12*

CS courses were initiated around the 1950s and 1960s (Calitz, 2021). The first CS programme was offered in 1953 by the University of Cambridge in the United Kingdom and the first CS department was formed by Purdue University in 1962 (Calitz, 2021). Subsequently, upon global establishment, departments of CS in SA were gradually introduced at universities during the 1960s and 1970s. Some of the SA universities that began offering CS courses during that period include the University of Stellenbosch, University of South Africa, University of Cape Town, Rhodes University and North-West University (Calitz, 2021). In the 1980s, many SA universities were either offering hardcore CS courses or CS-related courses. Recently (in 2022), I have noted in the course information from various international and SA universities that computing courses have been further popularised. However, in SA, the CS-related modules (especially programming) have not been popularised in the K12 stream (SAPeople, 2020; Fares, Fares & Vegas, 2021).

Some of the SA schools lack Information Technology (IT) infrastructure, which is critical for a successful implementation of computing education at primary and high schools. Few schools in urban or high-income areas have adequate internet connectivity, while some are still without enough IT infrastructure to offer CS education (Fares, Fowler & Vegas, 2021). According to Pyott and Sanders (1991: 1), "a problem in teaching CS in the South African context is that many students come from disadvantaged backgrounds and have not been exposed to computer technology". In 2020, only 40% of schools in SA had computer labs (SAPeople, 2020).

### *2.3.4 Overview of programming education research*

There have been various research initiatives around programming or CS education across the globe (Guzdial & du Boulay, 2019). Since the 70s, research has centred more on the technology (software and hardware) part of computing and less on pedagogical strategies (Kandemir, Kalelioğlu & Gülbahar, 2021). Luxton-Reilly et al. (2018) conducted a comprehensive literature review on introductory programming based on 1666 research papers between 2003 and 2017. Papers related to students were 489 in total, curriculum 258, assessment 192 and teaching 905. However, a concern is that only 24 papers out of 905 papers were related to computer program visualisation. The work of Lunn et al. (2021) collected and evaluated computer science education publication data between 2015 and 2020 sourced from high-impact international organisations, namely, Innovation and Technology in Computer Science Education (ITiCSE), ACM International Computing Education Research (ICER) and ACM Transaction on Computing Education (TOCE). The authors identified a growing trend in CS education research output from individuals and universities. Nevertheless, the authors stress that even though research and development seem to be growing globally, a target population or contextualised environment should be examined.

The South African Computer Lecturers' Association (SACLA) conference, formerly known as Computer Science Lecturers' Association (CSLA), is one of the most CS-focused teaching and learning research meetings held in SA. SACLA was established in the 1970s by academics and IBM (International Business Machines), a computer manufacturer based in the United States (Calitz, 2021). SACLA contributed significantly to collaboration with international scholars in growing computing

education research. Most of the time, the thematic areas of research included CS, CS curriculum, information systems and teaching and learning of programming. One of the main subjects identified and mostly researched as noted in the SACLA proceedings is computer programming. The other reputable scientific association similar to SACLA in research for teaching and learning CS concepts is the South African Institute for Computer Scientists and Information Technologists (SAICSIT).

## 2.4 Challenges in learning to program

This section outlines students' challenges with IPC. Outlining the challenges is important because they present better view of the fundamental issues of learning to program, which will further inform better instruction or guide more effective pedagogical solutions (Spohrer & Soloway, 1986).

### *2.4.1 Struggling Introductory Programming Students (SIPS)*

Every year, novices enrol in computing courses in various universities. Some students are informed about the course they have chosen; some are not informed at all. Some students indicate that they selected a certain course simply because it was available for enrolment, or they met the requirements. Beaubouef and Mason (2005) suggest that students are poorly advised when enrolling in CS, thinking that the course is all about computers. It was also found that students have limited career knowledge when it comes to Science, Technology, Engineering and Mathematics (STEM) courses (Hango, 2013). This problem has triggered various solutions in the form of match-making algorithms and course recommender systems (Taha, 2012; Joshi et al., 2012, Ramabu & Oberholzer, 2017). The basic background about computing, the relevance of Mathematics in computing and accurate perceptions about the nature of computing courses are not common knowledge among first-year computing students (Galpin & Sanders, 2007). This can worsen the performance of students in computing courses.

The majority of CS subjects are not introduced in the K12 stream (Grandell, Peltomäki, Back, & Salakoski, 2006), therefore, it is not unusual for computing students at universities to lack adequate prior programming experience. As a result, many first-year students encounter computing subjects for the first time at a university level. This can directly impact the success of computing students at university as they

subsequently struggle to comprehend programming concepts. Ultimately universities have to deal with SIPS. Furthermore, lack of proper incorporation of computational thinking in the K12 curriculum in developing countries (Grover & Pea, 2013; Su & Yang, 2023) can be part of the contributing factors that cause SIPS. While some university students can cope and comprehend computing modules at the first attempt, others do struggle. Ramaano, Ajoodha and Jadhav (2021) compared first-year CS students with prior computer experience and first-year students with no computer experience. The authors found that first-year students with computer experience performed better than students without computer experience. This shows that somehow a basic computer background has a positive effect on comprehension.

### 2.4.2 English language

Computing courses are mostly offered in the English language by universities in SA and globally. English has been deemed the international language and is chosen by many countries as a language of teaching and learning (Veerasamy & Shillabeer, 2014). The language barrier could be experienced by the students who are learning CS courses in the English language (Lei & Allen, 2022). This is because the success of teaching and learning is initially impacted by the understanding of the English language as a medium of instruction. Potentially, native English speakers have an advantage of language clarity and fluency over non-native speakers. Such students stand a chance to avoid language misconceptions on the programming specifications or written exercises to solve a problem through algorithm development. In Figure 2.1, we can see that a low-level English proficiency impacts teaching and learning which further impacts problem specification in a computer programming context.

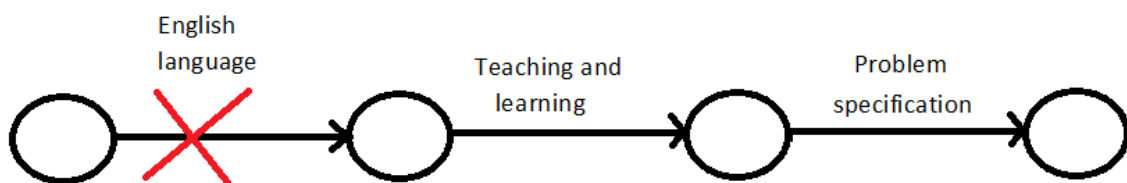


Figure 2.1: Implications of lack of English ability



To select and admit the best students into courses, universities make use of Mathematics (discussed in the subsequent section) and English as requirements to admit students into CS (Werth, 1986; Rauchas et al., 2006). This further limit the chances of prospective students with a low grade subject to be awarded admission into computing courses. Many SA students or international students speak and learn English as a second or additional language (Lei & Allen, 2022). This means that, should such students receive admission into the university, there remains a likelihood that they can struggle with English as the instructional medium.

There have not been significant research initiatives to accommodate students who experience English misconceptions in the CS education stream (Armenti, 2018). Pal (2016) compared native English students and students who used the local language to learn in the K12 stream. The authors taught introductory programming to the three groups by using the post-test scores methodology. Group 1 (previously learned in their non-English local language at K12) was taught introductory programming with the local language. Group 2 (previously learned in their non-English local language at K12) was taught introductory programming by using English language. Group 3 (used English language at K12) was taught introductory programming by using English language. Results demonstrated that generally, students who continue being taught introductory programming in their local language performed better. An earlier similar study by Lau and Yuen (2011) also found that Chinese-instructed students (Chinese as their native language) outperform English-instructed students. Idris and Ammar (2018), surveyed university students and lecturers in Libya, the authors found that programming performance is likely affected by inadequate English proficiency.

The most common programming languages like C, C++, Java and JavaScript also have keywords or commands which are English-like (Veerasingam & Shillabeer, 2014). Therefore, English proficiency is not only helpful for teaching and learning but also within the programming language itself.

### *2.4.3 Mathematics*

Universities make use of mathematics as one of the main benchmark requirements to admit students into CS courses (Werth, 1986; Rauchas et al., 2006). Nothing has

changed so far: as noted in the universities' websites as recently as 2022, most universities still use mathematics as one requirement to admit students into CS programs. This is because mathematics is a major element of CS course (Beaubouef & Mason, 2005; Beaubouef & Mason, 2005; Alpár et al., 2022). CS courses have mathematics as a separate module and other modules are somewhat dependent on it (Alpár et al., 2022). More importantly, mathematics is critical for a better understanding of artificial intelligence, computer graphics and programming (Baldwin et al., 2013). The high attrition rate is also noted among those with low mathematical performances or poor mathematics skills (Konvalina, Wileman & Stephens, 1983; Beaubouef & Mason, 2005). This means that a lack of mathematical knowledge can affect CS students' performance and therefore pedagogical solutions may be necessary to close the gap.

#### *2.4.4 Mental models*

“A mental model is a conceptual representation of an abstract concept or a physical system that provides predictive and explanatory powers to a person in trying to understand the concept or the system and guides their interaction with it ” (Wu, Dale & Bethel, 1998: 292). Mental models constitute the representation of students' mental understanding of how and why certain things work the way they do. According to Wu, Dale and Bethel (1998), educators should have an appropriate conceptual model of teaching which is informed by accurate mental models of the students. Non-experienced students learn differently according to their mental models, experienced students learn differently and possibly SIPS can learn differently by adjusting their mental model accuracy through pedagogical intervention.

One of the challenges in the comprehension of IPC is the incorrect mental models of students (Ramalingam, LaBelle & Wiedenbeck, 2004). In the case of programming education, the mental model is about the knowledge of how a program code and its semantics work. All solutions that strive to help students understand program code better are about adjusting the state of a student's mental models with that of computational or programming thinking. Various methods have been used to measure students' mental models of a program code. For example, Scholtz and Sanders (2010) used the tracing method to measure students' mental models of how recursive functions work. Jimoyiannis (2013) used the structure of observed learning outcomes

(SOLO) taxonomy to measure students' mental models on the understanding of variables and assignments. The viable and non-viable mental models were also investigated on the concepts like variables by Sorva (2008), value-passing and reference-passing (Madison & Gifford, 2002; Ma et al., 2007). This is an important exercise as it is possible for students to produce correct program code while retaining an inaccurate mental model of how the code works (Scholtz & Sanders, 2010). The understanding of program code represents a programmer's mental models, therefore the design of relevant support systems for teaching and learning of programming can lead to the accurate mental models (Storey, Wong & Müller, 2000; Heinonen et al., 2023).

#### *2.4.5 Programming jargon*

Each module in a specific course has special words which can be difficult to master. Such words are referred to as 'jargon'. Similarly, the computing environment has its own jargon. Soloway (1986) says teaching introductory programming students' the syntax and semantics is not enough, but programming terminologies, vocabulary, program goals and plans should be emphasised as well.

Mohamed, Hamilton and Souza (2011), surveyed novice programmers in their second week of learning and found that some students were able to fix basic errors and understood basic jargon while other students struggled with programming jargon. The work of Munasinghe, Bell and Robins (2021) further noted that educators also find some of computing vocabulary difficult to comprehend. The authors stress that sometimes this is caused by a lack of computing background or not being in the context of CS. As a result, this can further aggravate the confusion of teaching and learning to program.

#### *2.4.6 Problem solving*

In everyday life, the cognitive abilities to solve any problem is critical, even in the programming education where algorithms are developed. In programming, an algorithm is written so that it can solve a specific problem when it gets deployed in a computer or any other device. Problem-solving ability is one of the foremost skills that

must be possessed by IPS (Govender, 2021). Lack of problem-solving skills mostly lead to high failure rate in programming (Gomes & Mendes, 2007).

There are various approaches for problem-solving in programming education. The most common approach for problem solving is described by Bransford and Stein (1993) as:

- Problem Identification or formulation,
- Definition of the terms or problem,
- Develop strategies for the solution to the problem,
- Execute the solution and
- Observe, reflect, learn through the effect of the execution.

The above problem-solving steps resonates with the steps indicated by Gomes and Mendes (2007) and Sheth et al., (2016). The work of Ring, Giordan and Ransbottom (2008) present problem-solving steps as “analyse”, “design”, “build” and “test”. The first stage which is “analyse” means understanding and interpretation of the problem. The “design” step means using tools like flowchart to present the model to the solution. The “build” stage is about developing the actual program code using a specific programming language. The last stage (“test”) means using a test case for the experimentation of the program. In an effort to help the students to obtain this important skill, there have been various pedagogical attempts (including the approaches of teaching and learning in section 2.6) to improve teaching and learning to program. Furthermore, the subtopics discussed in this section like English language, Mathematics, mental models and programming jargon have a direct influence on problem solving abilities.

### *2.4.7 Summary*

Various challenges may affect students’ performances in computing courses. Generally, prior disadvantaged schools and lack of connectivity serve as the origin of these challenges. The other challenge is the lack of CS background and programming experience which contribute to the state of SIPS. The lack of accurate mental models about how a program code works, poor English language proficiency, lack of computational thinking and mathematical thinking further contribute to the state of

SIPS. These challenges directly or indirectly affect students' competence and performance in programming and potentially in other computing modules as well.

## 2.5 Imperative programming approach

This section focuses on the issues of learning IPC in the imperative programming paradigm. An imperative programming paradigm is about placement of a sequence of instruction that can be executed at a time by the computer (Pollak, Layka & Sacco, 2022). Researchers need to discuss the fundamental misconceptions in introductory programming which can inform relevant instructional means (Sadler et al., 2013, Altadmri & Brown, 2015). This section, therefore, is important for exploring such fundamental problems of learning to program.

### 2.5.1 *Programming languages*

The choice of a specific programming language to teach first-year students in computing courses has been evolving and debated. We have so many existing programming languages and academia is faced with the dilemma of choosing a relevant language for teaching and learning at first year level. Three decades ago, the Pascal programming language was one of the most prevalent programming languages by the industry for developing software and academia for teaching and learning purposes (Brilliant & Wiseman, 1996). At this point, popular programming languages used in the industry and also adopted by universities for teaching introductory programming include Java, C, C++, Visual Basic, PHP and Python (Siegfried et al., 2012; Rabai, 2015; Aleksić & Ivanović, 2016; Chen et al., 2019).

Various factors can inform the choice of a specific programming language for an introductory programming module. However, no significant differences between the chosen programming languages being studied in introductory programming have been found (Bennedsen & Caspersen, 2007; Watson & Li, 2014). This is true because students learn how to program without necessarily learning a programming language. Programming languages are similar in logic but differ in language specification (syntax). In an attempt to soften the learning process towards programming languages, some K12 curriculum features visual programming which in return gives a university student an experiential advantage when learning textual programming

(Chen et al., 2019). Visual programming makes use of graphics like drag and drop blocks, while textual programming consists of typing in a code (Chen et al., 2019). Some examples of visual programming include Alice (Cooper, Dann & Pausch, 2000) and Scratch (Resnick et al., 2009). Textual programming languages include Java, C++ and JavaScript.

Programming paradigms are another critical factor to be considered when choosing a programming language. The subsequent section discusses programming paradigms.

### *2.5.2 Programming paradigms*

Besides the complexity of selecting an appropriate programming language for industries to develop software or universities to teach, there are programming paradigms to think of. However, each paradigm can be relevant if selected appropriately. According to Krishnamurthi and Fislser (2019), paradigms can be referred to as a class of programming languages that behave similarly from a high-level perspective and have common features. Some programming languages are primarily imperative-based paradigms with procedural and object-oriented programming (OOP) as sub-paradigms (Sim et al., 2012). An OOP paradigm is centred around objects, data in objects and classes, while the procedural paradigm is based on a series of steps, subroutines with conditions and functions (Farell, 2011). For the most part, the procedural programming paradigm is recommended as introductory programming suitable for computing novices at university and for K12 students (Hendrix & Weeks, 2018).

Vilner, Zur and Gal-Ezer (2007) investigated students who took CS1 with OOP and students who took CS1 with a procedural programming approach. The authors found no significant differences between the two groups from their comparison. However, OOP requires a high level of abstraction which sometimes inhibits the proper learning of the fundamentals of programming (Delgado et al., 2016). Some universities in Europe and other continents adopt OOP; however, they still begin teaching novices with a procedural programming approach first (Aleksić & Ivanović, 2016; Luxton-Reilly et al., 2018). This method is generally referred to as the objects-later approach rather than objects-first or objects-early. Dümmel, Westfechtel and Ehmman (2019) contend

that students should be able to code both procedurally and also code through data in objects (OOP). The following section outlines the structure and details of procedural programming content for better understanding and context.

### 2.5.3 The structure of procedural programming

In the objects-later approach where the procedural programming paradigm is learned first, topics covered include problem-solving skills, a brief introduction to programming, arithmetic, decisions, nested decisions, loops, nested loops, arrays, parallel arrays and functions (Gal-Ezer & Harel, 1999). In Figure 2.2, the contents of the C++ textbook by Malik (2018) is presented. As with other similar textbooks, the content of the book by Malik (2018) shows an objects-later approach where procedural programming is learned and later OOP concepts are also learned.

1. An Overview of Computers and Programming Languages	1
2. Basic Elements of C++	27
3. Input/Output	117
4. Control Structures I (Selection)	175
5. Control Structures II (Repetition)	247
6. User-Defined Functions I	319
7. User-Defined Functions II	361
8. User-Defined Simple Data Types, Namespaces, and the <code>string</code> Type	433
9. Arrays and Strings	485
10. Applications of Arrays (Searching and Sorting) and the <code>vector</code> Type	563
11. Records ( <code>structs</code> )	611
12. Classes and Data Abstraction	649
13. Inheritance and Composition	723
14. Pointers, Classes, Virtual Functions, and Abstract Classes	793
15. Overloading and Templates	861
16. Exception Handling	951
17. Recursion	991

Figure 2.2: C++ Programming from program analysis to program design (Malik, 2018) - 1

The rest of the figures (Figure 2.3 to Figure 2.9) have been abstracted from the Malik (2018) textbook to provide a sample description of the content of procedural programming concepts. In Figure 2.3, the content consists of a typical introductory

phase where students are exposed to programming jargon, language syntax, data types and arithmetic.

<b>2</b>	<b>BASIC ELEMENTS OF C++</b>	<b>27</b>
	A C++ Program	28
	The Basics of a C++ Program	31
	Comments	32
	Special Symbols	32
	Reserved Words (Keywords)	33
	Identifiers	33
	Whitespaces	34
	Data Types	35
	Simple Data Types	35
	Floating-Point Data Types	38
	Arithmetic Operators and Operator Precedence	39
	Order of Precedence	43
	Expressions	44
	Mixed Expressions	45
	Type Conversion (Casting)	47
	string Type	49
	Input	50

Figure 2.3: C++ Programming from program analysis to program design (Malik, 2018) – 2

In Figure 2.4, the students learn about input and output through *cin* and *cout* if the programming language is C++. The concept of *assignment* is also emphasised in this type of section.

<b>3</b>	<b>INPUT/OUTPUT</b>	<b>117</b>
	I/O Streams and Standard I/O Devices	118
	cin and the Extraction Operator >>	119
	Using Predefined Functions in a Program	124
	cin and the get Function	127
	cin and the ignore Function	128
	The putback and peek Functions	130
	The Dot Notation between I/O Stream Variables and I/O Functions: A Precaution	132
	Input Failure	133
	The clear Function	135
	Output and Formatting Output	137
	setprecision Manipulator	137
	fixed Manipulator	138
	showpoint Manipulator	139
	setw	142

Figure 2.4: C++ Programming from program analysis to program design (Malik, 2018) - 3



The content of Figure 2.5 consists of control structures part 1 (referred as decisions or selections or if-statements). This section moves to an advanced level where nested decisions and compound statements are learned and constructed.

<b>4</b>	<b>CONTROL STRUCTURES I (SELECTION)</b>	<b>175</b>
	Control Structures	176
	Relational Operators	177
	Relational Operators and Simple Data Types	178
	Comparing Characters	179
	Relational Operators and the <code>string</code> Type	180
	Logical (Boolean) Operators and Logical Expressions	182
	Order of Precedence	184
	<code>int</code> Data Type and Logical (Boolean) Expressions	187
	<code>bool</code> Data Type and Logical (Boolean) Expressions	188
	Selection: <code>if</code> and <code>if...else</code>	188
	One-Way Selection	189
	Two-Way Selection	191
	Compound (Block of) Statements	195
	Multiple Selections: Nested <code>if</code>	195
	Comparing <code>if...else</code> Statements with a Series of <code>if</code> Statements	198
	Short-Circuit Evaluation	199

Figure 2.5: C++ Programming from program analysis to program design (Malik, 2018) - 4

The content of Figure 2.6 consists of control structures part 2 (referred as repetitions or iterations or loops). Advanced concepts include nested loops and a combination of loops and decisions.

<b>5</b>	<b>CONTROL STRUCTURES II (REPETITION)</b>	<b>247</b>
	Why Is Repetition Needed?	248
	<code>while</code> Looping (Repetition) Structure	249
	Designing <code>while</code> Loops	251
	Case 1: Counter-Controlled <code>while</code> Loops	252
	Case 2: Sentinel-Controlled <code>while</code> Loops	255
	Case 3: Flag-Controlled <code>while</code> Loops	259
	Case 4: EOF-Controlled <code>while</code> Loops	263
	<code>eof</code> Function	263
	More on Expressions in <code>while</code> Statements	268
	Programming Example: Fibonacci Number	269
	<code>for</code> Looping (Repetition) Structure	273
	Programming Example: Classifying Numbers	281

Figure 2.6: C++ Programming from program analysis to program design (Malik, 2018) - 5

Figure 2.7 shows how functions are taught. Things like function calls, function return type, passing values into a function and returning a value are learned.

<b>6</b>	<b>USER-DEFINED FUNCTIONS I</b>	<b>319</b>
	Predefined Functions	320
	User-Defined Functions	324
	Value-Returning Functions	324
	Syntax: Value-Returning Functions	326
	Syntax: Formal Parameter List	326
	Function Call	326
	Syntax: Actual Parameter List	327
	return Statement	327
	Syntax: return Statement	327
	Function Prototype	331
	Syntax: Function Prototype	332
	Value-Returning Functions: Some Peculiarity	333
	More Examples of Value-Returning Functions	335
	Flow of Execution	340
	Programming Example: Largest Number	341
	Programming Example: Cable Company	343
	Quick Review	349

Figure 2.7: C++ Programming from program analysis to program design (Malik, 2018) - 6

Figure 2.8 is still on functions, but more advanced concepts like reference parameters are introduced.

<b>7</b>	<b>USER-DEFINED FUNCTIONS II</b>	<b>361</b>
	Void Functions	362
	Value Parameters	367
	Reference Variables as Parameters	368
	Value and Reference Parameters and Memory Allocation	372
	Reference Parameters and Value-Returning Functions	382
	Scope of an Identifier	382
	Global Variables, Named Constants, and Side Effects	386
	Static and Automatic Variables	391
	Debugging: Using Drivers and Stubs	392
	Function Overloading: An Introduction	395
	Functions with Default Parameters	396
	Programming Example: Classify Numbers	399
	Programming Example: Data Comparison	404

Figure 2.8: C++ Programming from program analysis to program design (Malik, 2018) - 7

Figure 2.9 shows how the concept of arrays is taught. Strings are sometimes introduced with arrays because they are manipulated similarly. Advanced concepts like parallel arrays and two-dimensional arrays are also taught.

<b>9</b>	<b>ARRAYS AND STRINGS</b>	<b>485</b>
	<b>Arrays</b>	487
	Accessing Array Components	488
	Processing One-Dimensional Arrays	491
	Array Index Out of Bounds	494
	Array Initialization During Declaration	495
	Partial Initialization of Arrays During Declaration	496
	Some Restrictions on Array Processing	496
	Arrays as Parameters to Functions	497
	Constant Arrays as Formal Parameters	498
	Base Address of an Array and Array in Computer Memory	501
	Functions Cannot Return a Value of the Type Array	503
	Integral Data Type and Array Indices	506
	Other Ways to Declare Arrays	507
	<b>Searching an Array for a Specific Item</b>	507
	<b>C-Strings (Character Arrays)</b>	510
	String Comparison	512
	Reading and Writing Strings	514
	String Input	514
	String Output	515

**Figure 2.9: C++ Programming from program analysis to program design (Malik, 2018) - 8**

Novice programmers are expected to learn the content outlined in this section as part of the introductory programming syllabus. As emphasised previously in section 2.3.2, some novices find learning certain aspects of introductory programming difficult or challenging. The subsequent section discusses details of challenges in learning to program.

### *2.5.4 Details of learning challenges*

Programming challenges can be categorised within three knowledge areas: syntactical, conceptual and strategic knowledge (Qian & Lehman, 2017). Syntactical knowledge refers to the correct usage of syntax or programming language (Brown & Altadmri, 2017). Conceptual knowledge is a broader understanding of a certain concept (Pea, 1986). Strategic knowledge is when a student is capable of combining both syntactical and conceptual knowledge to write expert-level programs (Qian & Lehman, 2017). The promotion of syntactical knowledge is automated within compilers and a student must possess conceptual knowledge before coding can happen. Lack of conceptual understanding leads to a ‘conceptual’ bug (Pea, 1986). According to Pea (1986), a lack of conceptual understanding of certain programming concepts

happens to all primary school to college (university) students. Strategic knowledge can be targeted at introductory programming students, not to achieve an expert level but rather to master an introductory basis which could serve as a steppingstone to expert level.

A comprehensive study by Brown and Altadmri (2015) investigated common programming errors. The authors studied Java compilations of 250 000 novice programs across the world, summarising some of the errors as follows:

- a. Confusing the difference between = and == operators.
- b. Confusing the parentheses, square brackets and other types of brackets.
- c. Confusing short-circuit operators (&& and ||) and conventional operators (| and &).
- d. Forgetting the semicolon at the end of a line.
- e. Putting a semi-colon after the if-statement header, function header or for-loop.
- f. Incorrect separation, in a for-loop.
- g. Forgetting parenthesis when calling a function.
- h. Mixing up >=, <= with =<, =>.
- i. Confusion on how to call non-void and void methods.
- j. Having a function that returns a value but can reach the last line without returning any value. Example:

```
int calcv (int p)  
{  
  If(p < 0)  
    return 0;  
  p++;  
}
```

Some of the errors above are syntactical while some are semantic. Syntactical errors are sometimes unavoidable as they form part of language proficiency exercise. However, semantic errors are prevalent, harder to fix and require special attention (Brown & Altadmri, 2015; Ahadi et al., 2018).

Table 2.1 contains introductory programming misconceptions. The focus is on the errors related to the imperative programming paradigm.

**Table 2.1: Introductory programming misconceptions**

	<b>Misconceptions</b>	<b>Sources</b>
1.	Confusing == and =, misunderstanding the uses of an assignment operator	Du Boulay, 1986, Bruce, 2015; Žanko, Mladenović & Boljat, 2019
2.	Confusion on data types, variable scope, and confusion on whether a variable can store many values at a time	Goldman et al., 2008; Sadi, Halder & Saha, 2014; Bruce, 2015; Grover & Basu, 2017; Kohn, 2017
3.	Confusion on whether <i>if</i> content and the <i>else</i> clauses contents can execute at the same time, confusion on nested <i>if</i> , Booleans and logical operators and <i>if</i> statements in general	Sirkiä & Sorva; 2012; Altadmri & Brown, 2015; Bruce, 2015; Jiang, 2016; Jiang, 2016; Veerasamy, D'Souza & Laakso, 2016; Qian & Lehman, 2017
4.	Confusion between relational and comparison operators like >, =>, <, =<, also comparison like p > x > num	Altadmri & Brown, 2015; Veerasamy, Jiang, 2016, D'Souza & Laakso, 2016;
5.	Not able to trace	Veerasamy, D'Souza & Laakso, 2016
6.	Incorrect function references and values, calling functions with incorrect arguments or parameters, functions with incompatible return value/return type, and recursive functions	Wu, Dale, & Bethel, 1998; Altadmri & Brown, 2015; Veerasamy; Bruce, 2015; D'Souza & Laakso, 2016; Delgado et al., 2016; Malik & Coldwell-Neilson, 2017
8.	Misunderstanding arrays, usage of index in the array	Bruce, 2015; Souza & Laakso, 2016; Veerasamy, D'Souza & Laakso, 2016; Végh, 2016, Corral, 2019
9.	Misconception on loops/nested loops	Veerasamy, D'Souza & Laakso, 2016; Grover & Basu, 2017; Corral, 2019; Cetin, 2020
10.	Incorrect sorting algorithms	Veerasamy, D'Souza & Laakso, 2016

### 2.5.5 Summary

There are many programming languages and paradigms across the software development space. One of the dominating paradigms in the software industry is OOP; however, procedural programming has also been dominant across learning institutions for novice programmers. The imperative programming paradigm approach tends to make use of procedural programming before students advance to OOP. The structure of a procedural programming approach has been presented to shed light on what the content looks like. The details of challenges experienced by students in the procedural programming paradigm have also been demonstrated. The challenges discussed in this section expand our problem domain which informs the basis of investigating approaches of teaching and learning to program in the subsequent section.

## 2.6 Approaches to teaching procedural programming

There are many teaching approaches adopted by introductory programming educators. Some teaching approaches are generic and traditional, while other approaches have been empirically evaluated. This section discusses both empirical and non-empirical teaching approaches used in introductory programming education.

### *2.6.1 General teaching challenges in introductory programming*

Challenges of teaching and learning range from the limitations of educational tools, inadequate methods to keep the students motivated and engaged, the abstract nature of programming, the deficiency of mathematical skills and the unstructured curriculum in general (Medeiros, Ramalho & Falcão, 2018). These teaching and learning challenges are critical for addressing other SIPS' problems outlined in section 2.4. This means that addressing teaching challenges may in return solve many issues experienced by a novice programmer. In my opinion, the most important teaching and learning issues to improve have to be the methods and tools of teaching. This is the first challenge listed by Medeiros, Ramalho and Falcão (2018). The argument is that the correct methods and tools can have a positive influence on the remainder of the challenges learning to program. For example, keeping students motivated and engaged will be affected by the educators' chosen method of teaching or a tool used to aid teaching and learning. If the student does not have a concrete mathematical background or has no prior knowledge of programming, a correct teaching method and necessary tool can accommodate such a student.

Various methods and tools have been developed for teaching and learning introductory programming in an effort to improve teaching and learning. Such methods include using pre-recorded videos, tutorial sessions, tracing approaches, peer teaching, gamification, learning by repeated application, mentorship support, apprenticeship and active learning exercises (Forbes, Malan, Pon-Barry, Reges & Sahami, 2017; Medeiros, Ramalho & Falcão, 2018; Xie et al., 2019). However, these methods of teaching have not been effective or conclusive as the challenges remain (Rum & Ismail, 2017; Medeiros, Ramalho & Falcão, 2018). The subsequent section discusses various teaching approaches in introductory programming education.

### *2.6.2 Traditional methods of teaching and learning to program*

Traditionally, the mode of teaching takes place through verbal, visual or practical communication. As an introductory programming educator in the university, I have noted that traditional teaching and learning offered through verbal concept explanation and demonstration of sample programming code does not improve SIPS' comprehension. In the university or any learning institution, there is no fixed or approved method of teaching; hence, each educator adopts routine methods that do not necessarily accommodate SIPS. The educators' attempts to assist SIPS is primarily through the use of tutorial classes, assigning tutors to students and encouraging the students to make use of consultation hours.

Live coding through broadcast is another teaching approach practiced by some programming educators (Rubin, 2013). I have used this method of teaching several times before and can attest to its effectiveness for *some* students. The good thing about live coding is that students can see how you code, how you diagnose errors and how you fix the errors – this can help the students to understand. However, I still observed increasing numbers of SIPS irrespective of the adoption of live coding. Perhaps the issue with live coding is that students are expected to comprehend many aspects at the same time, with only verbal and type-in coding.

### *2.6.3 Analogy-like teaching approaches for introductory programming*

A comprehensive systematic literature review on introductory programming by Luxton-Reilly (2018) investigated students, teaching, curriculum and assessment. One main finding indicates that there is much more work to be done in the area of analogies and metaphors when teaching programming. Forišek and Steinová (2012) caution that poor metaphors can be misleading and can lead to the wrong conclusion. Rum and Ismail (2017) investigated the productivity of metacognitive scaffolding, self-directed learning, self-questioning, reflective prompts and the use of graphics as pedagogical methods that can aid the learning of computer programming. The authors found that all the listed methods do contribute to effective teaching and learning of computer programming. However, the authors found that the use of graphics helps students develop logical thinking and solves a variety of problems that arise due to the

complexity of learning to program. This is because the use of graphics or animation provides an 'analogy-like' teaching approach.

Students' relational understanding can be invoked by a representative or an analogy (manipulative or program animation) (Anwar et al., 2016). The concept of analogy in teaching programming can be manifested in the form of verbal explanation, use of manipulatives or animations. Educators who adopt the concept of analogy in teaching make use of real-world examples. Real-world examples such as analogy help the student link programming concepts to context (Konecki, Lovrenčić & Kaniški, 2016; Craig, Smith & Petersen, 2017; Moape, Ojo & van Wyk, 2017).

#### *2.6.4 Manipulatives in teaching and learning introductory programming*

The use of manipulatives has been evolving gradually in the field of mathematics education (Clements & McMillen, 1996; Kaminski, Sloutsky & Heckler, 2009; Bouck et al., 2014). There is also evidence of the use of physical manipulatives in teaching introductory programming.

Suzuki and Kato (1995) developed a physical programming language called AlgoBlock. AlgoBlock consists of physical blocks which are connected to a computer. As the user programs physically by arranging the blocks accordingly, a corresponding animated structure reflects on the computer screen. Their work offers a similar solution to Horn and Jacob (2007) who developed a physical programming language as well. Wyeth and Purchase (2002) developed square blocks with lights and sensors on the sides to improve and engage programming thinking in children. Just like Suzuki and Kato (1995), their manipulatives reflect on the screen as they are built and manipulated. Similar work by Hu, Zekelman, Horn and Judd (2015) resulted in a game called Strawbies (from the word strawberry) for children of age five to 10 years to learn to program. The game uses physical wooden tiles in conjunction with a mirror to reflect images on an iPad. An iPad is used as a real-time reflector for a moving object which is channelled to the images of strawberries. Another work by Sullivan, Elkin and Bers (2015) developed physical square blocks for use in the robotics-programming environment.



Aggarwal, Gardner-McCune and Touretzky (2017) use concrete manipulatives (tiles and flashcards) to foster basic programming principles. According to the authors, tiles are puzzle-shaped objects for modelling When-Do (When something happens, then do something) in programming. Flashcards contain rules and instructions that assist students in knowing how to interact and make moves.

In an attempt to find better ways of teaching recursion with manipulatives, Russian “nested” dolls (Wu, Dale, & Bethel, 1998) were suggested in the early 90s. Russian dolls are a set of wooden dolls where each reduced doll is placed into another, forming one doll containing many dolls.

The CS Unplugged organisation developed interesting physical manipulatives as well. CS Unplugged is focused on finding manipulatives relevant to teaching computing-related concepts. Different types of tools developed by this organisation can be found at [www.csunplugged.org.za](http://www.csunplugged.org.za). Concepts like human interface design, communication networks, cryptography, programming and image processing have the corresponding manipulatives on the CS Unplugged website.

Overall, the suggested solution for using manipulatives and unplugged material to teach introductory programming is more suitable for kindergarten or K12 learners. The approach may not be suitable for university students, especially in exhibiting fine details of the program code and may also be incompatible with an online teaching and learning mode.

### *2.6.5 Using animation programs*

According to Muller, Lee and Sharma (2008: 1), the “inclusion of multimedia technologies into the classroom has changed the educational landscape and introduced important changes in the educational system and impact the way students communicate information with each other”. These multimedia technologies can be in the form of animation programs. Sometimes multimedia technologies in education are presented in the form of digitised static pictures. Some of the static pictures or diagrams have been used regularly in books and other paper-based material to improve teaching and learning.

The world has since navigated into digital movable diagrams (referred to as ‘animation’) with the expansion of computer technology. A main feature of a

technology-based learning environment is animation (Lowe & Schnotz, 2008). The use of animation programs in education has proven to be significant, superior, informative, effective and more interactive than the use of static pictures (Lowe, 2001; Lowe & Schnotz, 2008; Ruffini, 2009).

The use of animation in the sphere of teaching and learning has been evolving. The developments around animation in education are accelerated by information technology infrastructure, more specifically, connectivity. The adoption of animation-based teaching and learning is convenient and compatible with various learning management systems (LMS), and as a result, this approach keeps growing and evolving.

The rationale behind animation-based teaching is that it increases interest, motivates, and attracts students to learn, serving as external visualisation that helps internal comprehension (Lowe & Schnotz, 2008). This is because a student can lack internal visualisation of the phenomena under study or can lack accurate perception or can require additional aids for comprehension. Animation programs can reveal some of the complex processes that were blurring visualisation, provide context to the learning outcome and increase learning speed (Brown & Sedgewick, 1984; Clark & Choi, 2005; Lowe & Schnotz, 2008). Moreover, animation programs can bring a powerful dimension to the presentation through colours, sounds, the animation itself and other visual enhancements (Gurka & Citrin, 1996). Teaching and learning with the aid of animation can reduce cognitive load, especially in learning a high demanding cognitive process (Lowe & Schnotz, 2008).

### *2.6.6 Summary*

This section presented various teaching methods to deliver the introductory programming module. The range of methods discussed includes general teaching practices, traditional teaching practices, analogies, manipulatives and animations. While each method has advantages and disadvantages, I however found that animation-based teaching has the potential for enhancing teaching and learning more than the other teaching methodologies. I therefore investigated and advanced the discussion on the existing animation programs for introductory programming under the

procedural programming approach. Section 2.7 below discusses existing animation programs to further identify a gap that this study intends to fill.

## 2.7 Animation programs

Generally, program animation in this study means computer graphics that animate or make a graphical representation of a programming code to enhance comprehension (Moreno, Sutinen & Joy, 2014). Animation is an important educational tool that must be featured in the teaching and learning in the classroom as it can give students a better mental model for program execution (Levy, Ben-Ari & Uronen, 2003).

While animation programs have the potential to enhance programming education, shortcomings associated with them are also reported. The study of Sorva, Karavirta and Malmi (2013) investigated program visualisations for introductory programming. The authors found that program visualisations are short-lived animation programs and may not live up to their expectation. According to Hundhausen, Douglas & Stasko (2002) instructors are unwilling to use algorithm visualisation (animation programs) because of the following perceptions:

- They think they do not have enough time to learn about the algorithm visualisation,
- It can take a lot of time to a point where other class activities are compromised,
- It can take time to develop relevant the algorithm visualisation and
- It may not be educationally effective.

Furthermore, the authors investigated the effectiveness of the algorithm visualisation and found that they can be either ineffective or effective. The authors found that the algorithm visualization just transfer correct mental models to students, it is not necessarily effective. To be effective, algorithm visualization has to be used as vehicle for active learning which incorporates constructivist theory. The incorporation of active learning and engagement into the animation programs is also supported by Sorva, Karavirta and Malmi (2013). In the light of these arguments, this section will thoroughly discuss existing animation programs and their abilities to translate into meaningful learning within the field of introductory programming education. The discussion of animation programs will include block-based programming.

## 2.7.1 Block-based programs

Block-based programming is a programming approach that allows students to drag and drop blocks as a way of building a program (Weintrop & Wilensky 2017). Block-based programming is used increasingly as an introductory module to programming (Weintrop & Wilensky 2017). Some K12 curriculum development practitioners prefer block-based because it lowers barriers to learning programming and motivates learners for possible interest in CS courses (Moors, Luxton-Reilly & Denny, 2018).

### 2.7.1.1 Scratch

One of the most common block-based virtual programming languages which covers topics like variables, decisions and loops is called Scratch (Resnick et al., 2009; Weintrop & Wilensky 2017; Mladenović, Boljat & Žanko 2018). Scratch programming is primarily used in high schools but sometimes also in colleges and universities as a foundation to computing or computing-related courses (Wolz et al., 2009). It has been reported that Scratch is useful for students with no prior programming knowledge (Mishra et al., 2014). The block-based programming approach is similar to pseudocode; hence, it focuses on problem-solving through drag and drop components (as seen in Figure 2.10) without worrying about the syntax of the solution.

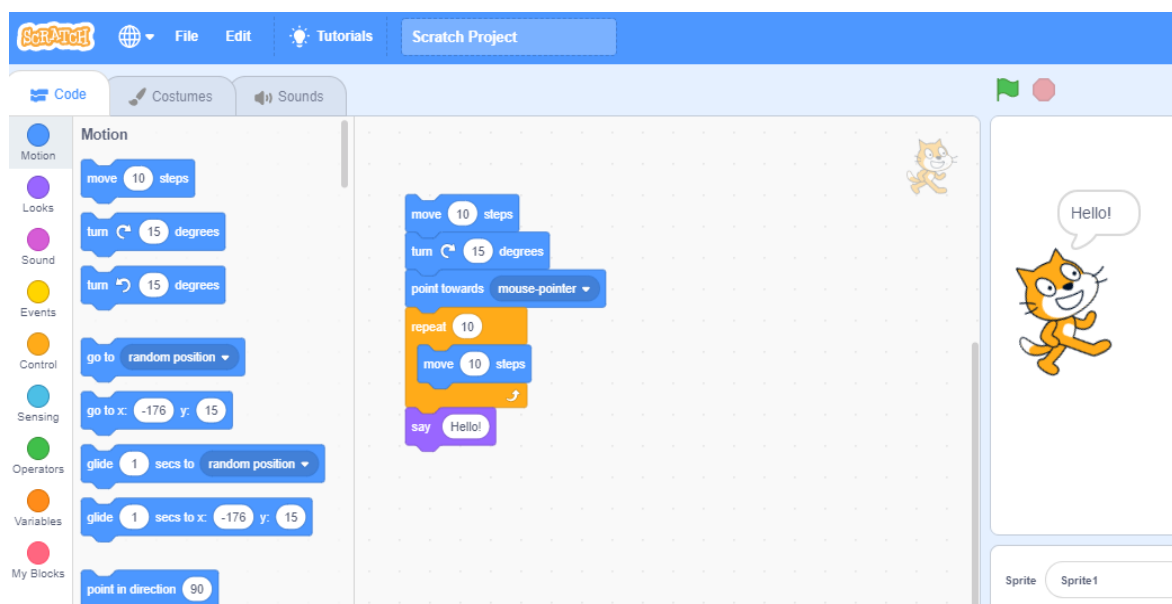


Figure 2.10: Sample scratch program

Sometimes starting students with block-based programming languages may not be effective at a university as it prolongs the duration of the course (Jiang, 2016). The duration of course completion can be further prolonged if there is no transition strategy from block-based to text-based programming. Also, block-based programming can reduce student confidence levels during the transition into the actual text-based programming environment (Moors, Luxton-Reilly & Denny, 2018). It has also been reported that long, complex Scratch programs are challenging to modify and waste time in drag and drop exercises because students have to look up each block in a category with confusing function calls and case statements (Tanrikulu & Schaefer, 2011).

### 2.7.1.2 Alice

Another block-based programming tool is called Alice (based on Python language). Alice provides the development of 3D animation programs (Cooper, Dann & Pausch, 2000). Alice features constructs like state transformation such as move, turn, destroy, decision constructs and looping constructs (see Figure 2.11).

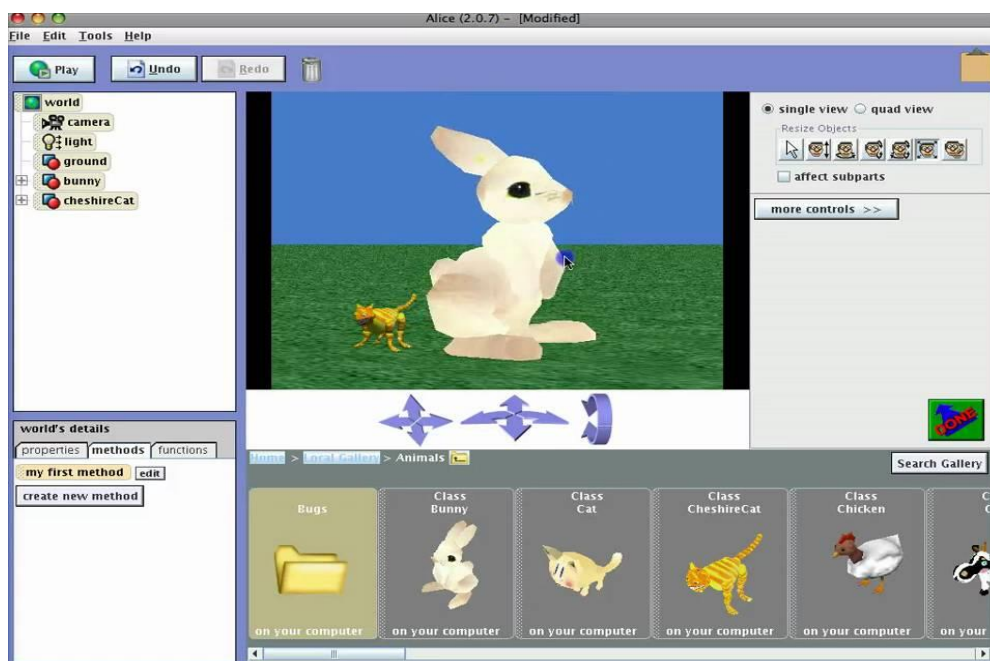


Figure 2.11: Alice program

The Alice-based teaching approach has been tested repeatedly in the K12 computer science education (Rodger et al., 2010). Just like Scratch, Alice as a block-based approach is appropriate for K12, however it can also prolong course duration if

introduced at university level and may not guarantee success in the text-based programming environment. Therefore, SIPS at HEI with or without Alice or Scratch experience from high school may need another pedagogical intervention in a text-based environment.

## *2.7.2 Animation-based programs*

### *2.7.2.1 WinTK-3*

Matsuda and Shindo (2001) worked to create an animation program called WinTK. This animation program allowed students to learn and edit source code and then face a challenge to rewrite, modify or extend it accordingly. There are four kinds of animation within WinTK. The first one is called WinTK-1 which contains basic tools and is based on Rabbit animation. The second one, called WinTK-2, is an interactive paint tool for helping students with keyboard and mouse event handling. The third is WinTK-3 and is based on photo-realistic animation. The last one, WinTK-4, is based on OpenGL and consists of 3D animation programs. The main aim of WinTK is generally reading and rewriting the source code which in return adjusts the outlook of the desired animation. The approach is not relevant or cannot be used as a pedagogical tool in a text-based programming environment at a HEI.

### *2.7.2.2 Jeliot*

Jeliot 1 is a basic program animation aimed primarily at helping high school students learn computer science concepts (Haajanen et al., 1997). Due to the limitations of Jeliot 1, another version was developed and named Jeliot 2000. According to Levy, Ben-Ari and Uronen (2000), some added features in Jeliot 2000 include a more flexible interface, the animated input/output, expressions and animated control decisions. It was found that Jeliot 2000 can improve vocabulary and concrete models for computer science concepts (Levy, Ben-Ari & Uronen, 2000).

Levy, Ben-Ari and Uronen (2000) used input/output concepts, if-statements and loops to test the Jeliot program with control and experimental groups. Even though it improves vocabulary and concrete models for high school students, such benefits are not relevant in a text-based programming environment. This is because computing students would already have been accepted into the courses and would be looking to

enhance their knowledge of the actual programming concepts. The development of both Jeliot 1 and Jeliot 2000 lacked a reference framework that can inform the design and important aspects of program animation which are linked to program code and effective teaching and learning practices.

A new version of Jeliot 2000 was developed, named Jeliot 3 (Moreno et al., 2004). Jeliot 3 has features like the program's ease of use and accommodates a large subset visualisation. However, Jeliot 3 has many sections in the animation space like method frame area, constants area, expression evaluation area and instance area which might be confusing to a novice programmer who needs to understand from the lower level. Another aspect missing in Jeliot 3 is the simplicity of multimedia design as described by Mayer (2005) which might be mistaken as a learning subject itself. Many other similar animation programs for teaching and learning focus on animation itself to a point where it loses its original intention, which is assisting in comprehending a program code.

### *2.7.2.3 UUhistle*

UUhistle is a program visualisation where students predict the program flow by manipulating the graphics components in the program (Sorva & Sirkiä, 2010). In that way students are able to know if the program visualised the way they predicted. This makes UUhistle a student-centred animation program which can be used off campus. However, just like Jeliot 3, the animation program is overloaded with many features which may add confusion or distract a student from learning the actual concept (see Figure 2.12). This restricts UUhistle from compliance with the coherent principle in the principles of multimedia learning by Mayer (2005), which recommends non-extraneous animation programs. Non-extraneous animation means avoiding overloaded features of animations programs that can cause students to deviate from learning the material. Secondly, the program does not feature any motion components to symbolise a program flow or value transfer from one instance to another.

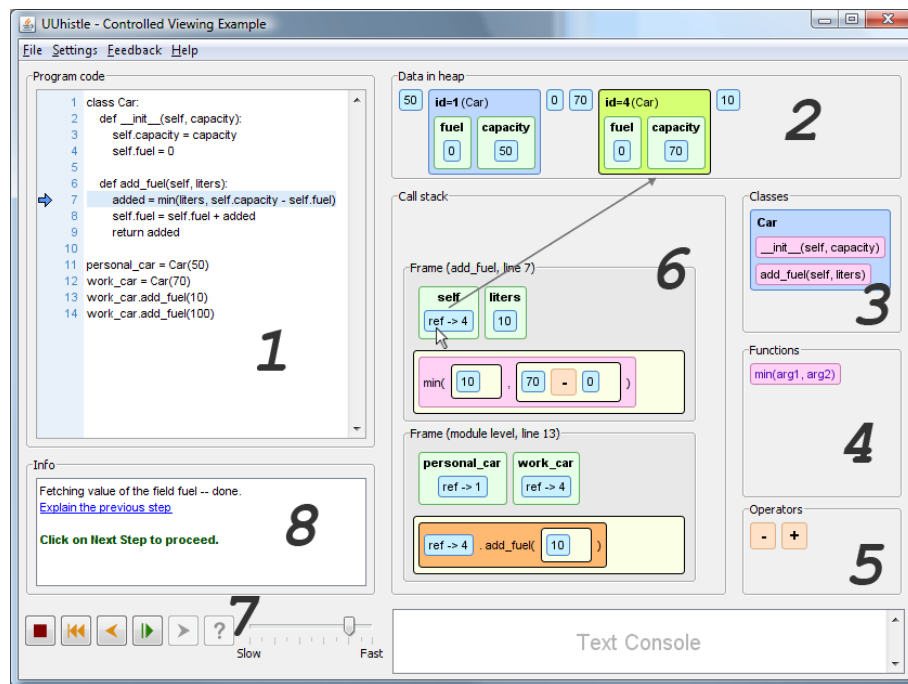


Figure 2.12: UUhistle program

#### 2.7.2.4 ViLLe

ViLLe is a program visualisation that focuses on the abstract level of program code (Rajala et al., 2007). This means that the uses of Ville can usher in a challenge or expand a learning gap for students who struggle with lower-level understanding of program code. The graphic presentation of ViLLe is similar to Jeliot, where graphics or animation are not fully simplified to the lower-level understanding of a novice. Therefore, ViLLe will not be compatible with SIPS.

#### 2.7.2.5 MatrixPro

MatrixPro is another animation program that focuses on data structures and algorithms (Karavirta et al., 2004). The program has drag and drop components which make it similar to block-based programming languages discussed in section 2.7.1. MatrixPro animation orientation is more on the abstraction and conceptual level. However, MatrixPro does not have a record of attempting or extending the animation program to cover IPC, so it may not be compatible with SIPS.

#### 2.7.2.6 Animation-based worked examples

Animation-based Worked Examples (ARAWEs) is an animation program for introductory programming that uses Augmented Reality (AR) technology (Cevahir,



Özdemir & Baturay, 2022). The ARAWEs focus on one concept (loops) and rely more on text for explanation which can further divide learner attention (see Figure 2.3).

**Sample 1:** Write the "for loop" statements that display the counter values between 1 and 3 on the screen.

**Answer:**

```
for (int i=1; i<=3; i++)
{
    Console.WriteLine ("Count = {0}", i);
}
```

#### Step-by-Step Explanations on the Solution

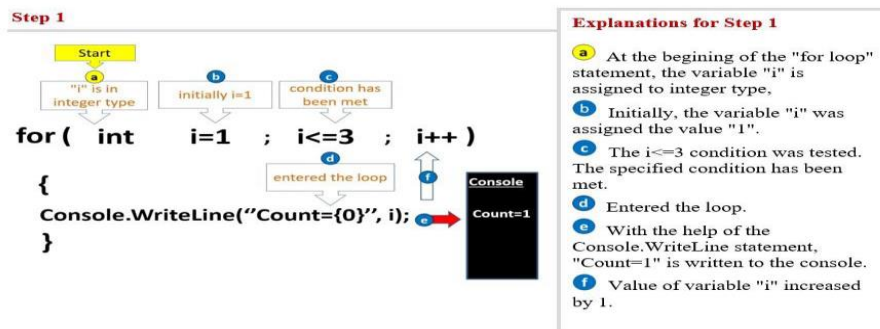


Figure 2.3: ARAWEs animation program

In the case of ARAWEs, the animation program does not comply with one of Mayer's (2005) principles of multimedia learning that cautions against overloaded text on the animation program as it may introduce divided attention or confusion.

#### 2.7.2.7 Other animation programs for introductory programming

Végh and Stoffová (2017) developed an animation card that assists students in understanding sorting. Through pre-test and post-test exercises, the authors tested the efficacy of sorting animation by demonstrating sorting to students. The application was also designed to aid in the teaching and learning of arrays (Végh, 2016). Similar animation programs can be found at <http://algoanim.ide.sk/index.php>. However, the designs also contain loosely connected graphics, which can result in knowledge that is not linked to the knowledge of arrays or sorting. Végh and Stoffová (2017) conceded that their animation card was unable to help students understand sorting in detail.

Osman and Elmusharaf (2014) developed a program animation called Courseware. Courseware uses various blocks of graphics that reflect a program code. The authors also tested the approach by using pre-test and post-test methodology, which included a sorting program and array. The program contains several animation blocks which

are manipulated through inputs of commands. The program is limited to traditional data structures and does not cover basic principles of programming in a text-based programming setup. A similar method is found in the work of Shi, Min and Zhang (2017) wherein a program animation called PlanAni was also tested through control and experimental groups. Sajaniemi and Kuittinen (2003) developed the PlanAni animation program. The program focuses on depicting the roles of variables in a program. The limitation of PlanAni, however, is that it focuses on one aspect of introductory programming. Moreover, PlanAni contains a high number of animated sections that are loosely connected and may confuse a student's attempt to connect the dots. The interface is not simplified: it contains redundant graphics and is overloaded with different shapes of graphics which further contradicts Mayer's (2005) principles of multimedia learning.

Robomind is another animation-based program. Robomind is a text-based programming with a focus on robotics-based building blocks (Faisal, Yuana & Basori, 2017), uses English-like commands to interact with an animation on the right side of the application. This approach is similar to a traditional robotic game, where a physical robot is controlled through commands; however, in this case a visual robotic is used. Robomind is suitable for primary and secondary school children (Yuana & Maryono, 2016) because it is aimed at conceptual understanding and encouragement to pursue computing courses rather than helping students with the actual IPC at HEI.

The work of Ussiph and Seidu (2018) investigated the impact of 3D animation programs in high school. The authors used a combination of Microsoft Basic and the Alice program (discussed previously in section 2.7.1.2) to test the effectiveness of the animation approach. It was found that animation-like programs are capable of helping students with IPC. However, as emphasised in section 2.7.1.2, the approach remains relevant as a K12 program and serves as a motivational tool for the learners to pursue CS at HEI.

Generally, all existing animation programs lack a proper referencing design guide and are not based on rigorously designed guidelines such as the principles of multimedia learning by Mayer (2005).

### *2.7.3 Pedagogical guidelines for uses of animation programs*

The limitations of existing animation programs are not only limited to weaknesses mentioned in sections 2.7.1 and 2.7.2, but also lack rigorously developed concrete pedagogical guidelines. Pedagogical guidelines mean incorporating the best uses of teaching practices or tools for impactful learning. In the case of this study, pedagogical guidelines should mean relevant strategies for Teaching and Learning Programming with Animation Programs. Many teaching and learning materials are developed, but the effective use of such materials is overlooked. As a result, they are part of factors contributing to the difficulties of learning to program (Cheah, 2020).

Animation programs can be powerful in helping students learn a program code and can enhance pedagogical effectiveness (Malone et al., 2009). However, according to Urquiza-Fuentes and Velázquez-Iturbide (2009), the pedagogical effectiveness of existing animation programs is not clear. This means that animation program design should be integral to the overall pedagogical design rather than a standalone teaching tool. Without this, there is a significant possibility that such animation programs can cause more harm than the good purpose they were designed for, simply because they are adopted and used along random pedagogical strategies.

## **2.8 Gap and summary**

I have discussed various teaching and learning challenges and issues regarding SIPS. The challenges range from language issues, incorrect mental models, lack of mathematical proficiency, technical jargon and basic problems experienced by programming students. In an attempt to address these problems, some pedagogical methods have been used or proposed. I have discussed various teaching approaches adopted by educators. Among existing methods, educators can make use of animation programs to enhance SIPS' comprehension of IPC.

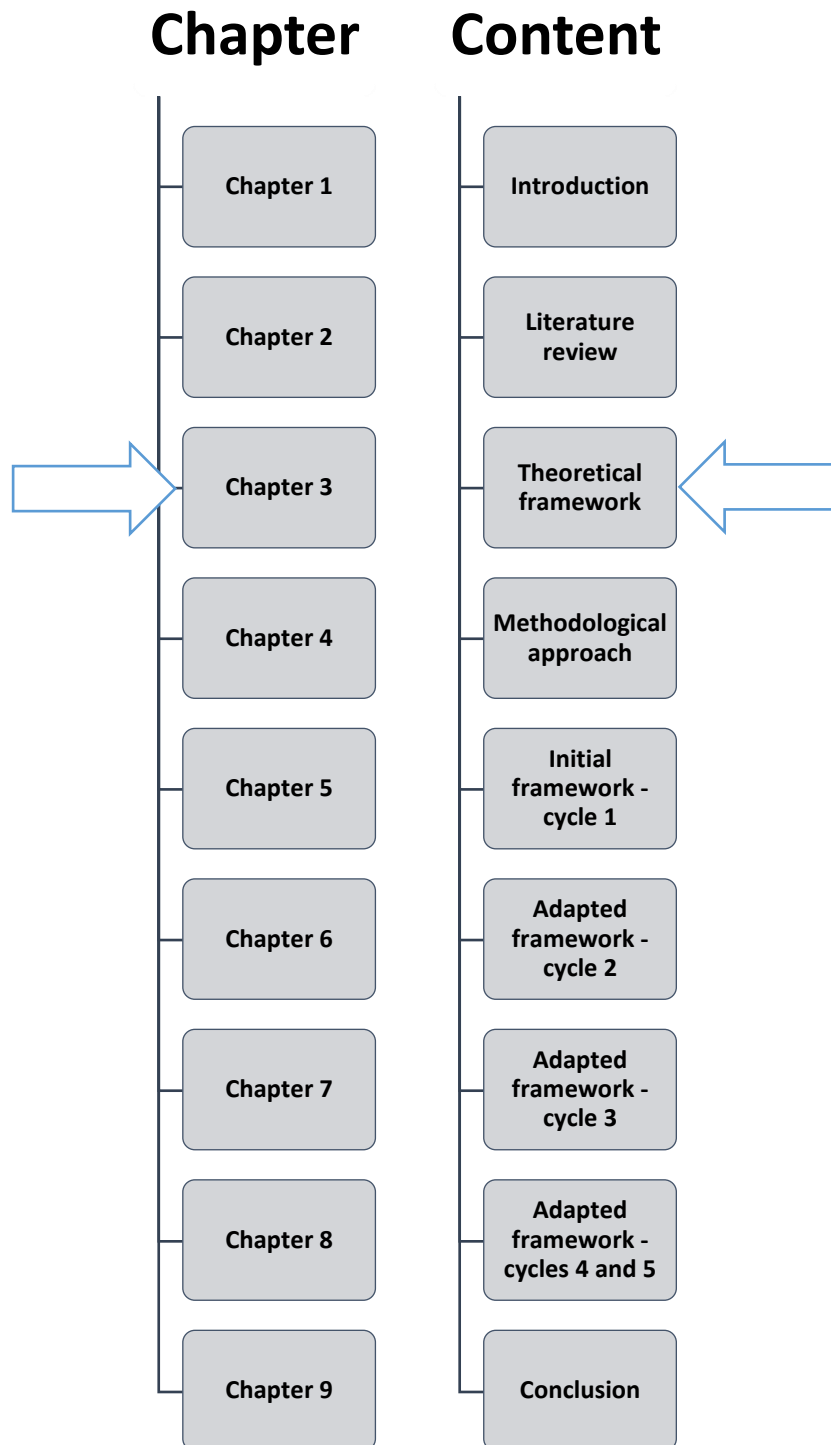
However, a design gap remains in the development of animation programs for introductory programming. Animation programs can do more harm than good if they are not properly designed and accompanied by relevant instruction for effective use. There is also a gap in relation to guidelines that educators can follow or adapt in using animation programs for teaching and learning introductory programming (Osman &

Elmusharaf, 2014). Pedagogical guidelines may be excluded with the assumption that introductory programming educators will know how to use animation programs. Existing animation programs for introductory programming are not adequately designed based on a relevant theoretical basis or guidelines for teaching and learning like the principles of multimedia learning by Mayer (2005).

Animation programs should be designed in a way that they are not an end in themselves but serve as a bridge to comprehend program code. It was also noted that many animation programs have been designed to be suitable for primary and secondary schools, but inadequate for students at HEI doing text-based introductory programming. Some animation programs focus on program details only with an assumption that students understand the basic concepts as important prerequisite knowledge. It was also found that some of the animation programs contain too many blocks animating the same program code, enough to confuse students.

As part of advancing towards our proposed solution, the next chapter (Chapter 3) discusses a theoretical framework in light of the identified gap and proposed solution. A theoretical framework consists of theories that frame the proposed study, serves as the underpinning theories that guides the entire study.

# Chapter 3: Theoretical framework



## 3.1 Introduction

The thesis statement says: “*A Teaching and Learning Programming with Animation Programs (TLPAP) framework can improve Struggling Introductory Programming Students (SIPS) comprehension of Introductory Programming Concepts (IPC)*”. As a way to address the stated thesis statement, the previous two chapters provided an introduction and literature review. In the introduction, it was indicated that programming can be difficult for some students. The misconceptions regarding various IPC and the limitations of existing teaching approaches define the root of the problem. Among various teaching approaches discussed in the literature review, the use of animation programs was noted as having the potential to advance the teaching and learning of IPC. Therefore, the literature review chapter further discussed existing animation programs used in introductory programming. Based on the gap found within the use of animation programs in programming education, the research proposes a solution in the form of an animation-based teaching approach for SIPS.

This chapter serves as a continuation of the literature review; however, the focus is sharpened to the relevant theories that frame the study. This chapter serves as a blueprint and the underpinning theoretical engine of our proposed solution. As there are various closely related theories concerning teaching and learning, it is important to discuss these first before identifying those that will underpin this study. This chapter begins by discussing relevant learning theories in section 3.2. Learning taxonomies as part of important educational resources are discussed in section 3.3. The teaching philosophy and constructive alignment are discussed in sections 3.4 and 3.5, respectively. Having discussed all relevant theories relating to the nature of the study, section 3.6 presents and argues the applicable theoretical components. The chapter is concluded in section 3.7.

## 3.2 Learning theories

Learning theories describe how a human being learns, thus outlining the process or a belief behind the learning process (Muhajirah, 2020). Some of the common and dominant learning theories include behaviourism, cognitivism and constructivism (Schunk, 1991; Mergel, 1998; Ben-Ari, 1998; Nagowah & Nagowah, 2009; Muhajirah,

2020). These theories have been compared in the work of Ertmer and Newby (1993) and found that each is appropriate in various contexts. This section discusses those and other learning theories.

### *3.2.1 Constructivism*

Constructivism is a learning theory and epistemology based on the belief that students learn by constructing their own knowledge rather than receiving and storing ordered information from the educator (Ben-Ari, 1998). It is an approach that allows students to continually reflect on their experiences and further construct new knowledge.

The constructivist theory is in contrast with traditional means of learning where information from books and notes is extracted and received as it is. In the constructivist teaching approach, active learning and participation happen throughout the outlined teaching stages. In a constructivist teaching setup, meaning and symbols cannot merely be passed from an educator to the learner. Instead, active teaching and learning trigger students to construct knowledge on their own.

The constructivist theory serves as a paradigm shift from an educator-centred teaching methodology to a learner-centred teaching methodology. According to Adom, Yeboah and Ankrah (2016), if constructivism is applied properly in the area of teaching and learning, it can supply powerful learning benefits to students. The roots of constructivism can be traced to the work of Jean Piaget (discussed in sections 3.2.2 and 3.2.3) (Perkins, 1991). Sometimes constructivism is also referred to as one of the branches of cognitivism (discussed in section 3.2.5) because both consider learning as a mental activity (Ertmer & Newby, 2013).

### *3.2.2 Piagetian theory*

Jean Piaget, one of the most influential theorists on children's cognitive development and learning, developed and modelled a theory of childhood development in 1920 (Piaget, 1929). One of the key observations within his theory is that he is not interested in whether the answer is correct or not, but in the underlying logic behind the answer. Piaget emphasises that children interpret the world differently throughout the various stages of their lives as their intellectual skills progress. The assumption in this theory

is that children interact with the environment as they grow and subsequently acquire learning.

This theory suggests that humans go through four different cognitive stages as they grow and learn from their childhood stage: Piaget's cognitive stages are sensorimotor, preoperational, concrete operational and formal operational. Each stage is detailed in Table 3.1.

**Table 3.1: Piaget's cognitive stages**

Stage	Age	Description
Sensorimotor	0 - 2	The child develops and understands through interaction with the objects in the world.
Preoperational	2 - 7	At this stage, the child uses acceptable language, grammar and mental images to interact with the world.
Concrete operational	7 - 11	Reasoning skills improve, they start to think dynamically and can categorise objects in order.
Formal operational	11 and older	Can evaluate abstract concepts, can form hypotheses and can apply learned information in a different context.

### 3.2.3 Neo-Piagetian theories

There is literature that connects classical Piagetian theory as discussed in section 3.2.2 to teaching and learning to program. However, the results have not been clear and are somehow disputed (Smith, 1992; Lister, 2011). As a result, Piagetian theory has been adapted by other theorists as a way of seeking a new interpretation or adapting it to various contexts (Demetriou, Shayer & Efklides, 2016). Such theories that emerge from Piagetian theory are referred to as Neo-Piagetian theories. There are several loopholes in the Piagetian theory which have paved way for restructuring in the form of Neo-Piagetian theories (Sevinç, 2019). According to Orlando and Machado (1996), several loopholes can be delineated as follows:

- generalising children's cognitive development;
- confining stages to ages with no confirmation or evidence;
- lacking in-depth details of learning stages development; and
- lacking social consideration.

Some Neo-Piagetian theorists believe in the constructivism learning theory (Sevinç, 2019), which has been discussed in section 3.2.1. According to Lister (2011), the literature indicates a more revolutionised adoption of Neo-Piagetian frameworks than the classical Piagetian theory. Nevertheless, Neo-Piagetian theories often contain



many assumptions of classical Piagetian theory or have an overlap between them. Some of the Neo-Piagetian theories are not bound by age restriction in their assumptions, as the theory applies to any individual at any stage of life.

There are a number of Neo-Piagetian theorists, including Robbie Case, Michael Commons and Kurt Fischer as examples (Case, 1992; Sevinç, 2019). Kurt Fischer argues in his theory that cognitive development cannot progress from one stage to another in a linear process because it ignores the behavioural variation of the learner (Fischer & Bidell, 2006). Just like constructivists, Fischer emphasises that children construct knowledge on their own in a favourable environment and with the assistance of social networks. The theory states that such knowledge construction cannot happen in a structured or ordered linear procedure.

Michael Commons is another Neo-Piagetian theorist who adapted the last stage of the Piagetian cognitive development stage. According to Commons and Ross (2008), the final stage of Piagetian cognitive development should not represent the end-point of cognitive maturity. Alternatively, Commons and Ross (2008) developed post-formal stages (post-formal thought) of cognitive development. Post-formal thought means that the students are curious enough to initiate their own creativity and innovation from time to time. Commons and Ross's (2008) post-formal stages are as follows:

- *Systematic stage*. Ordered relationships between at least two variables are established and systematic coordination takes a central role.
- *Metasystematic stage*. At this level, one would have properly mastered the systematic stage. Two or more variables are evaluated and compared to others in order to come up with an innovative solution.
- *Paradigmatic stage*. This stage is dependent on the previous stages, where the knowledge of the metasystematic stage is used to coordinate different variables to derive or bring about a new paradigm.
- *Cross-paradigmatic stage*. At this stage, the new paradigms created in the previous stage are experimented with and integrated into a new environment.

Another Neo-Piagetian theorist by the name of Robbie Case has developed a theory of executive control and conceptual structures where he bases his formation of

developmental stages (Case, 1984) on three elements, that is: environmental representation, goals representation and strategic representation to achieve the goals. Case (1984) insists that these elements are critical for a child to achieve set goals by employing identified strategies. Case also describes four stages (similar to the original classical Piaget stages of cognitive development) that interact with executive control structures. The stages are described as follows:

- *Sensorimotor stage.* A child learns how to see and hold objects. This happens from birth to 18 months.
- *Inter-relational structure.* A child uses pictures and relates them with action or other objects. This happens from 18 months to five years.
- *Dimensional structures.* A child develops cognitive abilities through representations. This happens from five years to 11 years.
- *Vectorial structure.* At this stage, a child can link up different structures learned in prior stages and can comprehend difficult relationships. This happens from 11 years to 19 years.

Generally, Neo-Piagetian theories present excellent observations and assumptions compared to the original Piagetian theory. Some of the theories base their extension heavily on Piagetian theory while some derive a small portion from it. Educators who are interested in the Neo-Piagetian theories must investigate all theories to adopt the most appropriate ones.

### ***3.2.4 Behaviourism***

The assumption in behaviourism theory is that conditioning through environmental interaction or stimuli channels a learner's behaviour (Quevedo-Torrero, 2009). Such an environment is deliberately constructed to trigger certain behaviours. The three key assumptions in this theory are environmental triggers, instructional repetition and disregard of internal cognitive processes in favour of external stimulus.

Behaviourism focuses on the patterns of behaviours that are repeated continually for mastery purposes (Mergel, 1996). This means that the theory advocates a repetitive instructional activity for mastery and understanding. Behaviourism theory concentrates on observable and measured behaviour as a way to measure if learning

has occurred (Good & Brophy, 1990). This means learning measurements do not focus on the internal thought process of a learner but on the change of behaviour as a result of specific environmental triggers. Therefore, behaviourism is not interested in the cognitive process of getting the answer, but its focus instead is on the type of answer itself.

### ***3.2.5 Cognitivism***

The cognitivism learning theory is based on observed behaviours which are believed to have been influenced by internal cognitive processes (Mergel, 1996). “Cognitive theorists recognise that much learning involves associations established through contiguity and repetition” (Good & Brophy, 1990: 187).

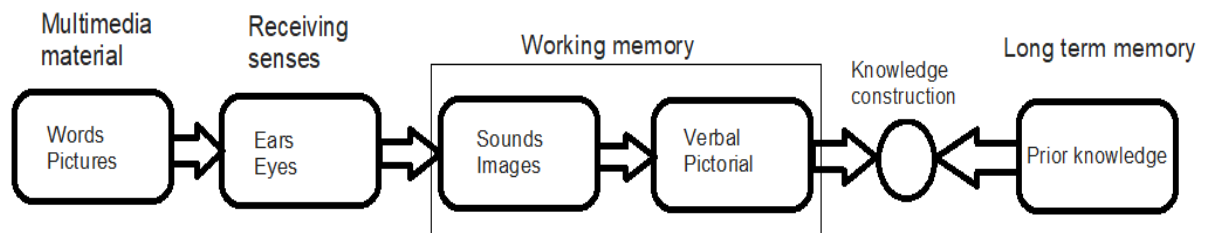
The assumption in cognitivism learning theory is that the learning process is an internal mental process that develops over time for better learning (Ertmer & Newby, 2013). Cognitive theory focuses on the mental structures, state of mind, learning processes, reception of information, storage of information and retrieval of such information from the mind (Ertmer & Newby, 2013). The theory is a direct opposite to behaviourism assumptions which disregard the internal thought processes of a learner. This is because cognitive theory further focuses on what the learner knows and how learning is acquired. The basis of this theory is that existing or prior knowledge must be available for learning to happen (Ertmer & Newby, 2013). This is because it is believed that existing knowledge will make the processing of new information possible through linking and comparison process. According to Schunk (1991), cognitive theory can be used in complex areas of learning more than other theories because of its stand on the evaluation of mental structures.

### ***3.2.6 Cognitive theory of multimedia learning***

Cognitive Theory of Multimedia Learning (CTML) correlates with the constructivist (discussed in section 3.2.1) learning approach because it also makes use of triggers to lead students into constructing their own knowledge.

CTML, was developed by Richard Mayer and is based on the assumption that pictures, words and audio can assist in translating to deep learning (Mayer, 2005). CTML is a

combination of text and pictures in the form of graphics, video or photos (Mayer & Moreno, 1998; Mayer, 2005; Sorden, 2012). Richard Mayer, as a theoretical expert in multimedia learning, suggests that students are not treating pictures, words and audio as separate entities but using them for knowledge construction. CTML is based on cognitive science for multimedia learning which includes dual channels where the brain processes information through visual and auditory channels. Firstly, when CTML is put into practice the visual channel is activated and then processes whatever graphic learning material is presented. Secondly, the auditory channel is activated and processed by the brain separately. The auditory channel is an important supplement to pictures, videos or graphic words and further translates to even deeper learning. The following figure shows the CTML process.



**Figure 3.1: CTML process**

Mayer further talks about the limited capacity assumption (LCA), emphasising caution against cognitive overload. He explains that a human being has a limited information capacity in the brain and that memory is not infinite. As a consequence, educators should avoid situations where information can overflow and be wasted. To avoid that, Mayer advises that we choose only relevant information to dispense.

Another assumption by Mayer is the active processing assumption (APA) which resonates in the working memory of a student. In this case, a learner must actively engage the multimedia learning material in working memory until the material is properly learned, or even pushed towards permanent memory. In this way, Mayer explains, the student is creating an appropriate mental model of information.

### ***3.2.7 Conversational framework***

The conversational framework was “developed by analysing the findings from research on student learning and using these to generate the requirements of the

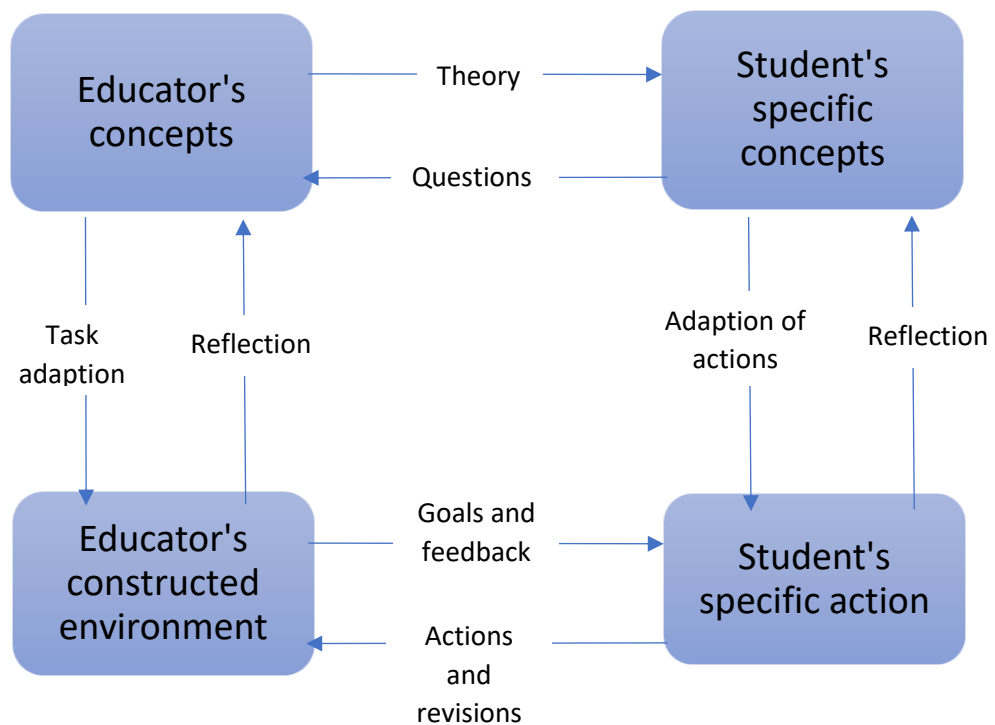
educator who is responsible for designing the learning process for their students” (Laurillard, 2002: 158). The conversational framework was developed by Dianna Laurillard in 1993. The theory emphasises that teaching is a dialogue or a conversation in a learning process and that learning at a higher level must be both theoretical and a practical (Laurillard, 2002; Laurillard, 2013). The framework makes use of the ideas of learning theories like collaborative learning, constructivism, instructions and social learning.

This framework draws on different media formation, educators, students and other relevant pedagogical aspects to support the creation of the learning process, especially between educator and student. The specification in the framework is important as it can help understand how formal learning occurs and how to best develop learning materials.

The theory considers four important components, namely:

- educator's (or tutor's) concepts;
- educator's constructed environment;
- student's specific concepts; and
- student's specific actions.

The relationship between the above components is described in Figure 3.2. Basically, Figure 3.2 shows various activities, interactions and adaptations of concepts that happen between the educator and the student. Such interactions include specific theories from the educator to students followed by questions from the student to the educator. The students adapt the actions in the educator's constructed environment based on the theory and are prompted to reflect based on the experience gained. On the other side, the educator adapts the students' tasks and reflects on the students' actions and feedback.



**Figure 3.2: Components of a conversational framework**

Laurillard further groups the characteristics of different teaching media into the narrative, interactive, adaptive, communicative and productive. The information plotted in Table 3.2 presents the characteristics of such media.

**Table 3.2: Conversational framework teaching media**

Media form	Learning experience	Technology
Narrative	Attending, apprehending	Talk, video, television, broadcast, podcast
Interactive	Investigating, exploring	Library, compact disk, learning objects, web
Adaptive	Experimenting, practicing	Lab, field trip, simulation, virtual environment
Communicative	Discussion, debating	Collaborative environment, online conference
Productive	Articulating, expressing	Talk, print, model, digital files, animation, web page, blog, wiki

Each media form supports a different dimension of teaching and learning; therefore, it may not be possible for one media form to satisfy and support every dimension. However, we have noted that some media forms contain possible attributes which overlap slightly. Therefore, a combination of media forms may be necessary for deep and meaningful teaching and learning.

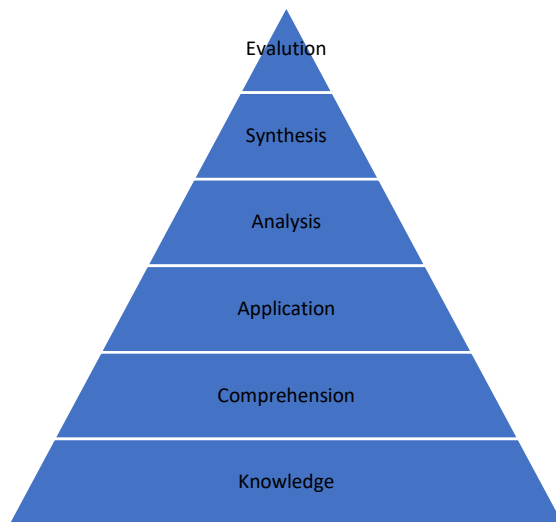
## 3.3 Learning taxonomies

Bloom (1956) explains that while different people in the room can view one window, they may not see the same thing. This view depends on the posing angle of the person, height and overall perspective of the person. A person who has an overall view has the chance to see what is happening in the sky compared to what is happening on the ground. A person with a limited view can rely on the explanations of another person who has a full view. The person being taught has to rely upon and trust the person who is explaining the story. On that note, it is very important to understand what students know and the state of their current cognitive abilities. This can give a narrator (the educator) an idea of how to approach the narration (or, teaching), and this can happen through the lens an educational taxonomy.

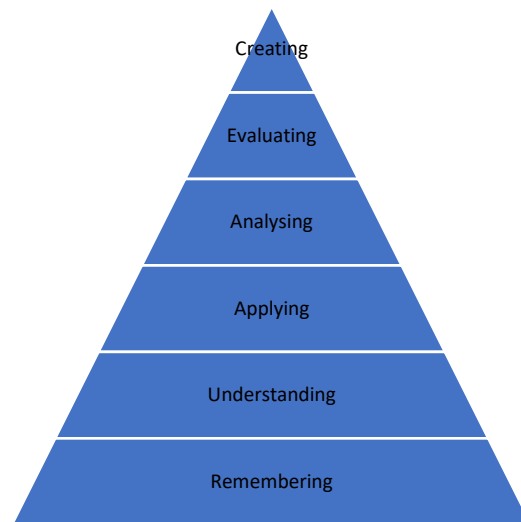
The existence of educational taxonomies helps by measuring and categorising students' intellectual levels, thereby forming an important element in any sort of pedagogical development. If educational taxonomies are properly used, they can help the educator to evaluate students' cognitive levels which can better inform the method of teaching. There are many educational taxonomies proposed and used by various institutions of learning. However, Bloom's taxonomy and Structure of Observed Learning Outcome (SOLO) are popular and preferred by many educational practitioners. The subsequent sections discuss the Bloom and SOLO taxonomies.

### 3.3.1 Bloom's taxonomy

Benjamin Bloom created an educational taxonomy in 1956 to categorise levels of abstraction (Bloom, 1956). According to Azuma, Coallier and Garbajosa (2003), Bloom's taxonomy was first discussed at the 1948 American Psychological Association (APA) in Boston where topics like examining and education were discussed. Bloom categorises learning domains or educational objectives into cognitive, affective and psychomotor. This taxonomy classifies learning or thinking within the cognitive domain into six cognitive levels, from a simple stage to more complex stages (Forehand, 2010). The original version of Bloom's taxonomy is depicted in Figure 3.3. Bloom's revised version (Figure 3.4) of educational taxonomy classifies the lowest stage of learning as *remembering* where students need only to recognise and recall what they have learned.



**Figure 3.3: Bloom's taxonomy**



**Figure 3.4: Revised Bloom's taxonomy**

The revised Bloom's taxonomy by Anderson and Krathwohl (2001) is defined as follows:

- *Remembering*. This stage is about memorising and being able to recall. For example, a question like "What is the capital city of the United States?" requires a learner to remember.
- *Understanding*. At this stage, the students should be able to explain in their own words. This means they must interpret, exemplify, classify, summarise, infer and compare.
- *Applying*. Here the learner should be able to apply knowledge gained in a different context.
- *Analysing*. At this stage, the learner is capable of linking and examining the relationship between various concepts.
- *Evaluating*. The learner can appraise, criticise or justify a situation based on the knowledge gained.
- *Creating*. At this stage, the learner should be capable of identifying the problem and creating an original solution.






### 3.3.2 SOLO taxonomy

SOLO is an educational taxonomy and systematic way of dealing with approaches to learning and the general intellectual development of a learner (Biggs & Collis, 1982; Lister et al., 2006; Biggs & Collis, 2014). SOLO was developed by Biggs and Collis



(1982) and has primarily been used as a hierarchical model to evaluate or classify students' responses to various levels of complexity. Table 3.3 shows the stages of the SOLO taxonomy.

**Table 3.3: SOLO taxonomy**

Prestructural	Unistructural	Multistructural	Relational	Extended abstract
				

Similar to Bloom, the stages of the SOLO taxonomy begin with the lowest stage (prestructural) progressing until the last stage (extended abstract). According to Biggs and Collis (1982), the stages are described as follows:

- a. *The prestructural stage* is when the learner is missing a point and not competent; no learning has taken place and help is needed to initiate learning.
- b. *The unistructural stage* is when the learner is capable of following simple things like naming, following and identifying. A symbol of one bar line in Table 3.3 means a learner learned one thing which is not related to anything nor coordinated with other things.
- c. *The multistructural stage* is when the learner can list, describe or combine. A learner learned several things but cannot relate them yet. A symbol of three bars in Table 3.3 means that they are not connected, indicating loose or scattered knowledge.
- d. *The relational stage* is when the learner can integrate a structure, have a coherent understanding, can analyse, apply, compare, relate, argue and justify. In this stage, the learner can link ideas and relate them to make sense of one coordinated whole.

- e. *The extended abstract* is when the learner can formulate, create, hypothesise and reflect. This means that the learner uses the idea or knowledge gained in the relational stage and extends it.

SOLO has been used and preferred in programming education (Whalley et al., 2006). For example, the work of Malik, Tawafak and Shakir (2021) has emphasised alignment and assessment of teaching approaches to SOLO taxonomy. The main features of SOLO that stands out as compared to Bloom's taxonomy is that it focuses on observable outcomes, the process and the depth of understanding.

### 3.4 Teaching philosophy

Teaching philosophy is a personal or public statement about the beliefs, values, attitudes or culture that is supportive of teaching (Goodyear & Allchin, 1998). The main component of a teaching philosophy is the description of how learning occurs, the intervention strategies, goals, professional development and steps to achieve them (Chism, 1997), "Articulating an individual teaching philosophy provides the foundation by which to clarify goals, to guide behaviour, to seed scholarly dialogue on teaching and to organise evaluation" (Goodyear & Allchin, 1998: 103). A teaching philosophy can serve as a reflection or clarification for the teaching practice (Murray, 1997). There are many statements of teaching philosophies documented by professors and educational organisations with varied compositions and structures (Chism, 1997). Some examples of teaching philosophy can be found in a book by Schmier (1995) entitled, *The humanity of teaching*.

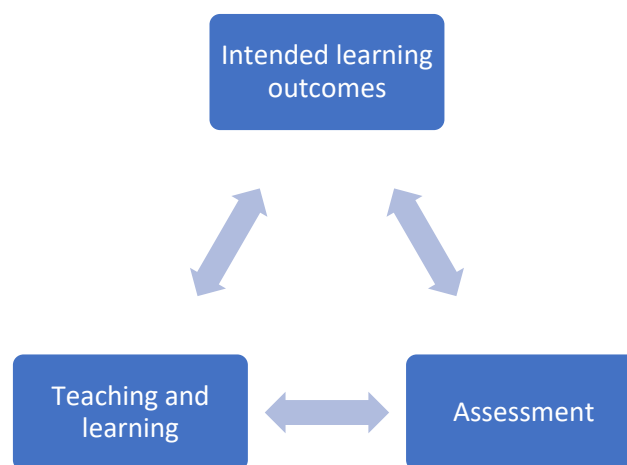
Goodyear and Allchin (1998) suggest that a teaching philosophy can be formulated by reflecting on the experiences, observations, mission, the vision of the faculty and literature on teaching and learning. This means that a teaching philosophy should not be in contradiction with an adopted learning theory. Goodyear and Allchin (1998) further contend that a teaching philosophy should demonstrate creativity and innovation; the implementation of a teaching philosophy is important as the theory is put into practice. The details of such actions or implementation strategies are to be included in the statement of teaching philosophy.

Even though there is no fixed format for an acceptable teaching philosophy, Chism's (1997) advice on the general format of teaching philosophy is as follows:

- A teaching philosophy should be brief, document length guided by context.
- A teaching philosophy should be written with simple terminology.
- A teaching philosophy should use first person in the narrating statements.
- A teaching philosophy should come from someone who is professionally committed to teaching and learning.

### 3.5 Constructive alignment

Constructive alignment (CA) by John Biggs (1996) is an outcome-based approach wherein the intended learning outcomes are designed before teaching and learning can happen. This means that teaching and learning methodologies and the assessments are aligned or shaped to the intended learning outcome (Biggs, 2001; Biggs & Tang, 2010; Wang et al., 2013) (see CA flow in Figure 3.5).



**Figure 3.5: Constructive alignment framework**

CA has been popularised, found to be effective, powerful and is viewed as the most influential model in teaching and learning by many universities (Houghton, 2004; Biggs & Tang, 2010; Kandlbinder, 2014). CA has also been made a policy in some higher education institutions as it is in support of the modern curriculum and validates the quality of the qualifications (Loughlin, Lygo-Baker & Lindberg-Sand, 2021). Teaching and learning that is aligned to the intended learning outcomes bring necessary

consistency across the board, including the assessment (Kandlbinder, 2014). The design strategies of traditional approaches to teaching, learning and assessment are mainly random or consist of loose entities that have no communication links to one another. Moreover, traditional means view teaching as transmitting of information from the educator to the student, so the teaching plan is focused on what the educator does (Biggs & Tang, 2010). Contrary to these traditional means, the CA approach is one in which the educator focuses on the outcome and each entity is in support of another entity.

In the CA, the assessment addresses the described learning outcomes (Kandlbinder, 2014). In this way, the educator sees if the intended learning outcomes have been achieved by the students. Assessment in constructive alignment is rubric-oriented rather than grading through the calculation of marks (Biggs & Tang, 2010). Biggs and Tang (2010) emphasise that assessment in CA should be qualitative rather than quantitative-driven. This means that the evaluation and analysis of the assessment should adopt qualitative methodologies. It is also recommended to relate the question to the bigger picture to resemble the relational stage of SOLO taxonomy (Trigwell & Prosser, 2014).

CA is compatible and aligned with the constructivist learning theory (Biggs, 1996) (discussed in section 3.2.1.). This is because CA is centred on activities that promote active learning (Loughlin, Lygo-Baker & Lindberg-Sand, 2021).

## 3.6 Applicable theoretical components of TLPAP

This section discusses the applicable theoretical components of TLPAP.

### *3.6.1 Overview TLPAP theoretical components*

I have discussed various theories of learning and other related theories in the previous sections of this chapter. The discussion paved a way for us to select and use relevant or applicable theories in this study. The TLPAP is therefore underpinned and framed by CTML, conversational framework, constructivism and the application of constructive alignment (see Figure 3.6).

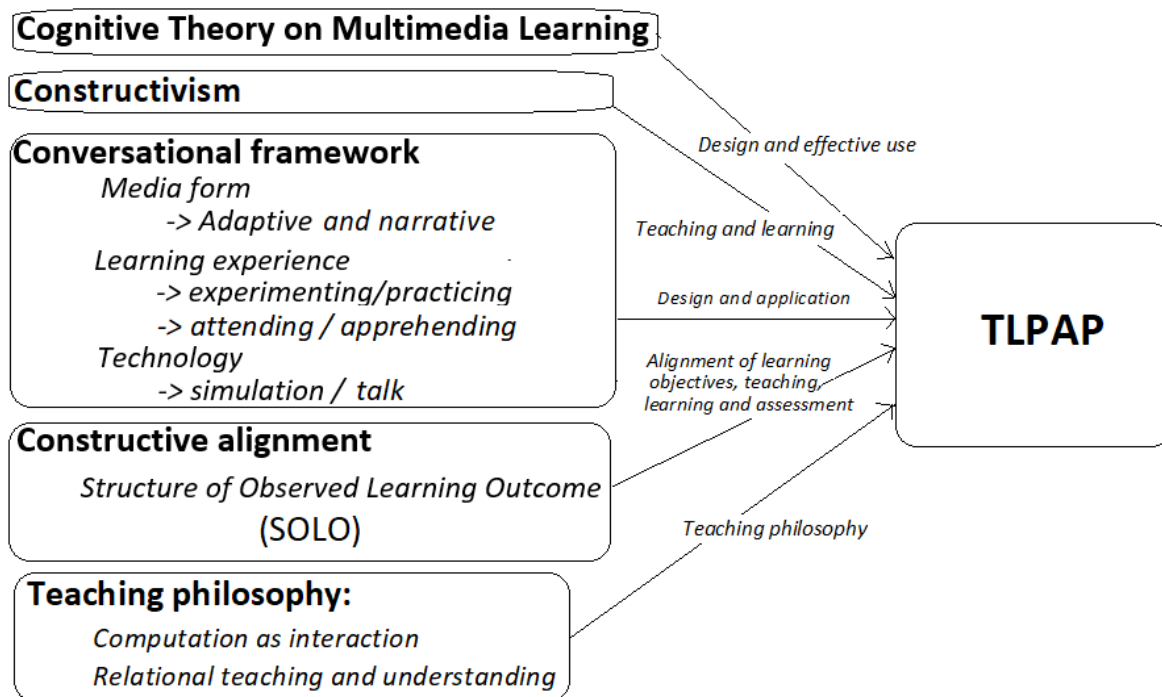


Figure 3.6: TLPAP applicable theoretical components

### 3.6.2 Adopted learning theories

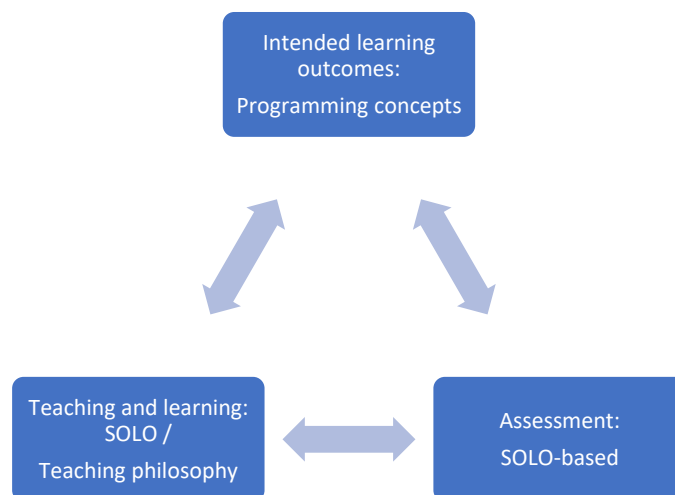
Our study is making use of animation programs to trigger knowledge construction. Therefore, it is vital that the alignment of this research be guided along the lines of relevant theoretical framing, identified as CTML, constructivism and conversational framework.

The CTML assumption is appropriate for TLPAP and is further grounding the TLPAP framework development, animation development and teaching and learning. CTML is relevant because the theory was designed in relation to how a human mind works (Mayer, 2005). The general CTML recommendations and guidance like animation structuring and how human-beings learn with animations are well within the frames of TLPAP.

The conversational framework by Laurillard (section 3.2.7) is adopted based on its relevance to the university setup and multimedia learning. Since TLPAP is designed for students at a higher education level like the university and contains different media formats, the adaption of the conversational framework comes in handy. More importantly, the conversational framework focuses on the student and the educator communication framework when using technology resources like multimedia

resources. There are five media forms (depicted in the previous section, in Table 3.2) in the Laurillard framework and adaptive and narrative media are appropriate to shape the TLPAP. Adaptive media accommodates and promotes experimentation and practicing which is relevant to the nature of programming education. The media form also supports the simulative structure of the TLPAP. I have also found the narrative media form appropriate, which further validates the discussion and explanation during the simulation broadcast of the animation programs.

Teaching and learning to program through the traditional methods (opposite to constructivism) can be incompatible for some students due to the abstract nature of program coding. Therefore, I find the constructivist approach (also compatible with the conversational framework and CTML) appropriate as an application and belief that forms the basis of teaching in the TLPAP framework. The constructivist approach is also compatible with the CA necessary for TLPAP. CA is a constructivist-oriented pedagogical approach that emphasises the alignment of learning, teaching and assessment to the intended learning objectives. In CA, learning outcomes are aligned to the assessment and teaching. I also adopt and apply CA in TLPAP based on its novelty and consistency across teaching, learning, assessment and learning objectives. This is important because teaching does not take place in the absence of learning and assessment. The view of CA within TLPAP is depicted in Figure 3.7.



**Figure 3.7: TLPAP adapted constructive alignment**

In Figure 3.7, the intended learning outcomes are part of programming (in various cycles of action research) for the effective use of TLPAP. Furthermore, such learning outcomes are developed under the premises of the SOLO taxonomy. The use of SOLO further helps in having the appropriate verbs for the intended SOLO level (Biggs & Tang, 2010).

Successful compilation of programming code or an algorithm requires the learner to hold the attributes of stage 4 (relational stage) of the SOLO taxonomy. An algorithm that consists of analysing a scenario, variable declarations, storing values, interacting with values conditionally, storage, sequential steps, iterating necessary programming statements and grouping similar programming statements requires a learner to possess relational stage attributes. This means that anyone who struggles to comprehend questions in the relational level of SOLO is struggling or is categorised as SIPS. Therefore, the new means of teaching is a deliberate effort and novel means to advance students to the relational level of the SOLO taxonomy.

To deal with the assessment/evaluation of feedback, programming code is categorised as per Whalley et al. (2006) in Table 3.4.

**Table 3.4: SOLO and Whalley et al. (2006) programming code benchmarking**

SOLO	Programming code benchmarking
Prestructural	The algorithm is not related to the question.
Unistructural	The student answers one aspect of the question in the algorithm.
Multistructural	The student can put line by line programming statements together and sometimes the summary is included.
Relational	The student can summarise the programming code according to the question.
Extended abstract	<i>No data</i>

We are not able to push SIPS to reach an extended-abstract level of SOLO taxonomy at this stage as we are dealing with a struggling novice programmer; usually that level can only be achieved through overtime learning and repetition. In this way, it substantiates our proposed TLPAP not as a standalone teaching solution but as a coordinated teaching strategy informed and connected to a relevant cognitive level, assessment, learning and taxonomy.

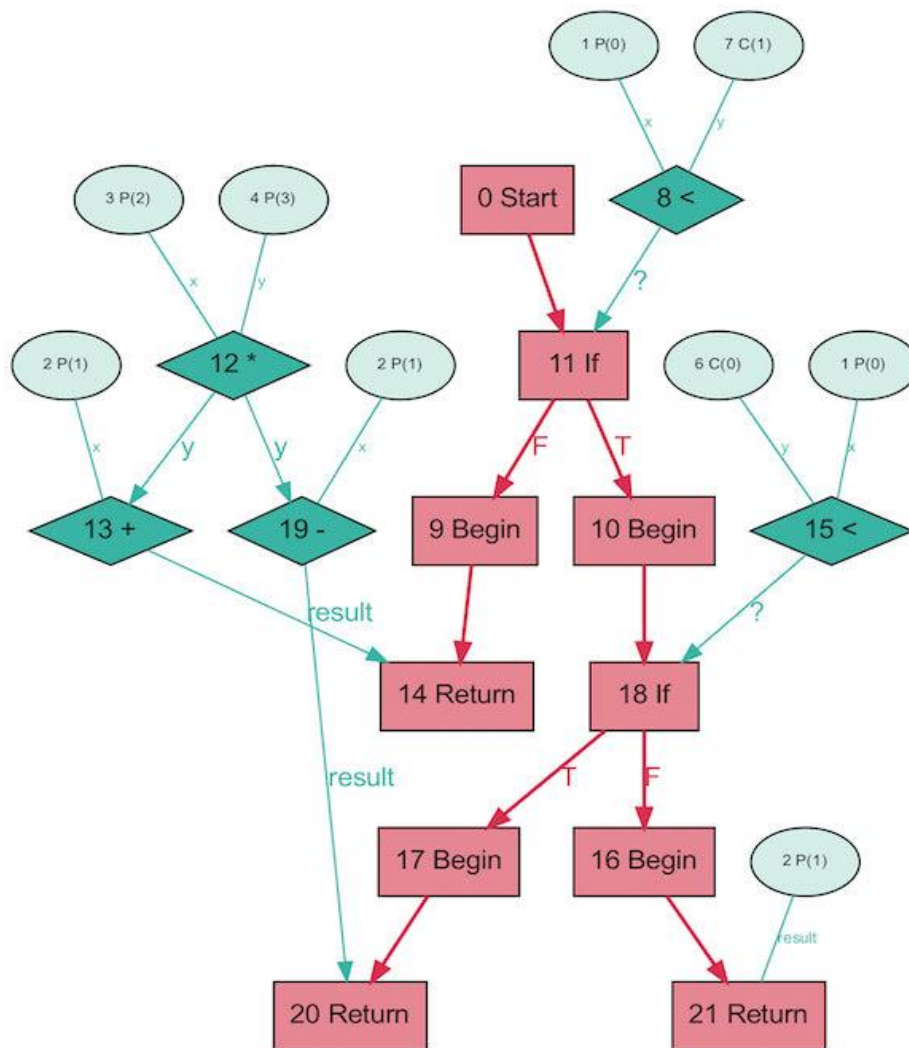
### 3.6.3 TLPAP teaching philosophy

In this section, I give directions and the necessary theoretical basis and parameters for formulating a teaching philosophy compatible with TLPAP or for programming education in general. It is important to note that in this section I am not able to prescribe a specific or fixed teaching philosophy as it is a personal or individualised assumption. While a teaching philosophy differs from person to person, as indicated in section 3.4, sometimes a person is asked to align a philosophy with certain aspects of the organisation or literature in order to avoid contrary statements or practice. Similarly, this section outlines the relevant basis for one to consider formulating a teaching philosophy within the area of programming education or when adopting TLPAP. Section 3.6.3.1 describes the nature of program code with a relevant emphasis using Stein (1998). This is to help the educator understand the structure of programming. Having understood the structure of programming, the subsequent section (section 3.6.3.2) consists of the fundamental basis of teaching to program which is an attempt to innovate the teaching of programming relevant to the relational level of SOLO taxonomy.

#### 3.6.3.1 Computation as interaction

Teaching and learning to program should be viewed from a point of *computation as interaction* rather than computation as calculations (Stein, 1998). Programming should be taught from the introductory programming stage as embedded entities interacting with a dynamic environment just like in video games and robotics. Programming is about putting an ordered set of instructions to the computer to act in a certain way. Such instructions are called an algorithm or program code. A program code must be designed in a desired sequential order so that the interaction of such instructions is not ambiguous. Teaching programming from the knowledge of *computation as an interaction* view can lay a good foundation for successful comprehension from a high-level perspective down to fine details of a program code. Therefore, before one can think of going deeper into developing a teaching philosophy or relating the philosophy to any learning theory, it is important to have a clear understanding of the nature of programming. Figure 3.8, accessed from Seaton (2020), is a sample data structure called sea-of-nodes that shows a novel way of seeing a compilation behind a program code and shows how interactive programming code can be, which further vindicates *computation as an interaction* basis.





**Figure 3.8: Computation as interaction illustration**

The sea-of-nodes in Figure 3.8 presents an overview of how a piece of a program code was compiled and one can analyse the structure of data flow without seeing the actual code. The red arrows in the figure are for control flow; the green arrows mean data flow and boxes (rectangles) are operations. The  $P$  stands for the parameter; therefore,  $P(0)$  will mean a parameter of zero. The  $C$  stands for constant; thus it means  $C(2)$  value. The case is not necessarily to look into the fine details of Figure 3.8, as the main purpose of depicting it is to visualise a sea-of-nodes data structure in a high-level view of program data flow during compilation and program code interactivity. The main argument is that keeping such structure (any type of data structure) in mind should influence a teaching philosophy.

### 3.6.3.2 Relational understanding

A thorough understanding of the nature of program code or compilation of program code as discussed in the previous section should in part inform a relevant teaching philosophy. The study proposes a teaching philosophy that promotes relational thinking and understanding. Therefore, I borrow from Skemp's (1976) theory on relational understanding to further shape a relevant teaching philosophy which is compatible with programming education and TLPAP.

In Mathematics education, Skemp (1976) defines two types of understanding or thinking as *instrumental* and *relational*. Skemp's theory suggests that instrumental understanding means that students know the rules and how to apply them without understanding the reason behind them. Basically, the students apply the rules without substantiated knowledge. For example, a student is taught and learns to get the algebraic question correct, with no real understanding of a reality or what the solution means. It is akin to memorising to learn the correct answers, but then it ends there. Relational understanding is about knowing *why* and *how* the rules work the *way* they do. As it is about being able to relate or connect to other rules, it means that a student has succeeded in acquiring contextual knowledge. However, the prerequisite for relational understanding is instrumental understanding.

According to Skemp, the following are the advantages of relational understanding:

- easily adaptable to other activities;
- easy to recall;
- effective; and
- organic relational schemas.

As part of the TLPAP framework or as a basis for a relevant teaching philosophy, Skemp's instrumental understanding is equated to the knowledge of programming syntax and its application. Syntax is a set of statements written using a certain programming language. Relational understanding is equated with the understanding of how to write down and relate a program code, thereby forming a logically coordinated program code. In mathematics education, a student must learn instrumentally since it is a requirement to initiate relational understanding. Similarly, in

programming education, a student needs the knowledge of syntax (instrumental understanding) before using the same syntax to write a logical algorithm (relational understanding) which will likely have a sea-of-node type of data structure, as explained in section 3.6.3.1. I relate Skemp's relational understanding with programming education because introductory programming students need to deeply know *how* and *why* they write a programming code to deliver a coordinated solution. However, unlike instrumental understanding from the mathematics education side where it is enough to produce the correct answer, in programming education the knowledge of syntax alone is not enough to produce a coordinated algorithm.

As indicated in the issues of teaching and learning to program (Chapter 2), some of the challenges include teaching students to master the syntax, while simultaneously enhancing comprehension to develop a logically coordinated algorithm. Because of that issue, some academic departments emphasise relational understanding more than instrumental understanding by making use of pseudocode. With pseudocode, the syntax is made easy as it is English-like wording, but sequence and logic remain the main part of the pseudocode (Pyott & Sanders, 1991). Similarly, the focus on relational understanding should be aimed at enhancing comprehension and promoting conceptual understanding (Pea, 1986) more than improving syntax knowledge. For example, if the programming code's display statement is incorrect but displays in the correct place or correct order, then it is logically correct. Syntax knowledge should improve over time through repetition to master the language. Additionally, compilers help students to correct syntax errors by detecting and suggesting possible solutions. Skemp's relational teaching and understanding is also compatible with our target SOLO level (relational level), which further emphasises the importance and relevance of this theory to TLPAP or programming education in general.

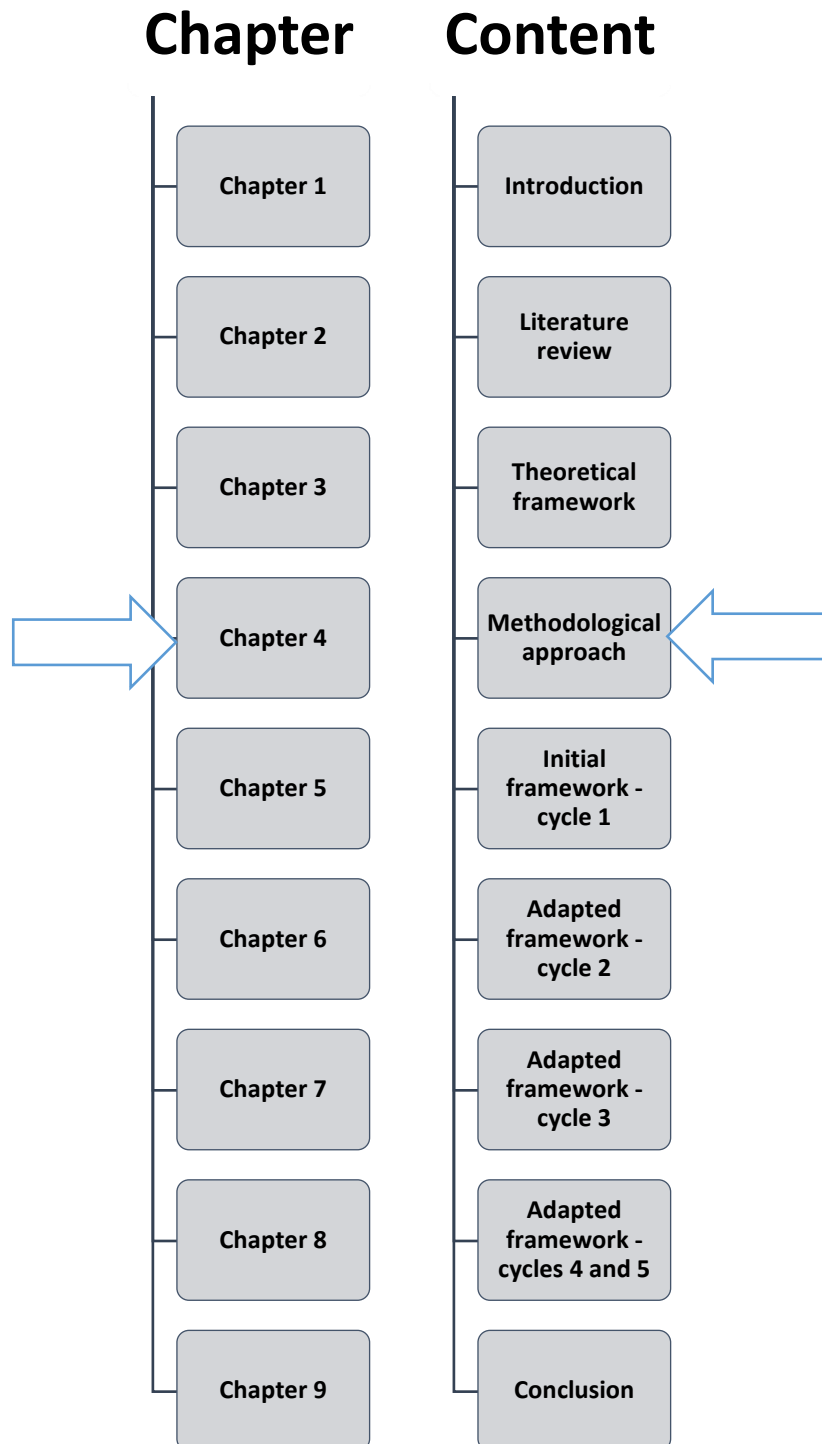
## 3.7 Conclusion

Various learning theories have been discussed in the first section of this chapter. The discussion gave insight into learning theories. As a result, applicable theories were identified that can shape, design and frame TLPAP. CTML, constructivism and conversational framework have been identified as applicable theories that underpin TLPAP. CTML is used as a guide to the design and effective use of animation programs. Conversational framework and constructivism underpin the design and

application of TLPAP. The exact details of how such designs happened are presented in Chapter 4 (Methodological approach).

The constructive alignment (CA) framework has also been identified as a critical component of TLPAP. CA is based on developing and aligning learning outcomes to teaching, learning and assessment. The TLPAP is designed to function within the parameters of CA which further enables the benefits of consistency and quality across teaching, learning, learning outcomes and assessment. This chapter has also presented some background on the nature of program code to guide the formation of a teaching philosophy, a personal or public statement that is supportive of teaching and learning. The subsequent methodological approach (Chapter 4) is rendered compatible under the umbrella of a given theoretical framework.

# Chapter 4: Methodological approach



## 4.1 Introduction

My thesis statement states, “*A Teaching and Learning Programming with Animation Programs (TLPAP) framework can improve Struggling Introductory Programming Students (SIPS) comprehension of Introductory Programming Concepts (IPC)*”. This study has laid out an introduction in Chapter 1 which emphasised the difficulty of learning programming for SIPS by outlining several challenges of learning to program. These challenges are found within the introductory concepts like assignment, decisions, nested decisions, repetitions, nested-repetitions, function parameters, function references and arrays. More details of these introductory programming challenges were addressed in Chapter 2 (a literature review). Chapter 2 also discussed the limitations of the existing approaches of teaching to program, the potential uses of animations and the limitations of existing animation programs. The theoretical framework (Chapter 3) discussed various theories relating to teaching, learning and animation programs. Chapter 3 was concluded with the applicable theories that frame or underpin this study. These theories include constructivism, the conversational framework, Cognitive Theory on Multimedia Learning (CTML), Structure of Observed Learning Outcomes (SOLO) and the Constructive Alignment (CA) framework.

The discussion of this chapter continues by presenting the research methodology in this study. A research investigation is typically undertaken under a specific research methodology. Research methodology is a plan that is theoretically informed to conduct a study (Ladislav & Ellen, 1984), and further informs the choice of specific research methods (Crotty, 1998; Grix, 2004; Rehman, & Alharthi, 2016). A research methodology can be in various dimensions depending on the nature of the study (Kothari, 2024). A well-organised methodology helps in answering research questions (Goddard & Melville, 2004). The research question addressed in this chapter is as follows:

- How can we develop a TLPAP framework for teaching and learning IPC?

This research question is addressed by presenting a research methodology. The chapter begins by discussing the methodological perspective in section 4.2, research paradigms in section 4.3, research design in section 4.4, evaluation strategy in section

4.5, action cycles progression in section 4.6, limitations of the study in section 4.7, ethics statement in section 4.8 and conclude the chapter in section 4.9.

## 4.2 Methodological perspective

There are three main methodological approaches in a scientific investigation: qualitative, quantitative and mixed method approach (Steckler et al., 1992; Gray et al. 2007; Ortiz & Greene, 2007). The quantitative research approach focuses on quantity, statistics or numbers while the qualitative approach focuses on quality (Kothari, 2004). Quantitative research is about quantifying data and analysing variables through statistical means (Apuke, 2017). Data in quantitative research is concerned with mathematical models or statistical analysis while qualitative data is concerned with categorising, summarising or interpreting (Saunders, Lewis & Thornhill, 2007; Apuke, 2017). Furthermore, qualitative research strives to achieve deeper meaning of the phenomena, exploration of a thought process or the process of in-depth insight (Strauss & Corbin, 1998; Creswell, 2016). Examples of qualitative methods include studying documents, interviews, interacting closely with humans or seeking deeper meaning from field notes or documents (Apuke, 2017).

A mixed-method approach is a combination of quantitative and qualitative methods to solve a research problem (Creswell, 2003; Tashakkori & Creswell, 2007). A mixed-method approach is adopted when a problem cannot be fully solved using either a qualitative or quantitative approach separately (Creswell, 2003). This means that the nature of the study should inform whether the study is taking a qualitative route, quantitative route or mixed-method approach.

This study is generally qualitative because it intends to study, understand and interpret students' algorithms. These algorithms are studied to assess the impact of the adopted teaching and learning methods adapted in this study. The other qualitative methods of data collection like interviewing the students and "talk aloud" are not considered as the written exercises (algorithms) reflects what the students thought and understood before and after the intervention.

## 4.3 Research paradigm

Research paradigm refers to the philosophical basis that grounds scientific investigation or a source of the basis for a researcher's worldview or assumptions (Lincoln et al., 2011; Krauss, 2005; Babbie, 2014). All research must be guided by a specific paradigm (Botha & Taylor, 2022). There are three common paradigms that can underpin a scientific investigation within our field of study: that's positivism, interpretivism and critical theory (Rehman & Alharthi, 2016). Post-positivism and pragmatism (emanating from positivism) have also been noted as common paradigms (Guba, & Lincoln, 1994; Kankam, 2019; Kaushik & Walsh, 2019). The following sections explain each research paradigm.

### 4.3.1 *Positivism*

The assumption in positivism is that reality exists on its own, is context-free and is not influenced by humans (Richards, 2003). This means that the ontological assumption of positivists is more of realism because reality is natural (Rehman & Alharthi, 2016). The positivists' epistemological stand is on objectivism, meaning they are anti-subjective in their scientific investigation. They describe phenomena as they are, without interference or intentional conditioning that may lead to disturbance to nature. The positivist approach relies on experiments, observation, deductive analysis and construction of laws. Positivists also rely more on statistical or quantitative approaches (De Villiers, 2005). A positivist researcher is not to be influenced by the objects nor should the objects influence the researcher. Most of the time, the results of a positivist researcher are claimed as true if the procedure can be repeated and yield the same results (Guba & Lincoln, 1994).

### 4.3.2 *Post-positivism*

Post-positivism was formed as an alternative to positivism, which addresses some of its criticisms or rejected philosophical assumptions (Henderson, 2011; Kankam, 2019). The assumptions embedded in the post-positivist paradigm are that knowledge is subjective, is socially or mentally constructed by individuals and is not fixed (Henderson, 2011). Positivists are regarded as realists; however, post-positivists are regarded as critical realists (Guba, 1990). In post-positivist research, the period of data collection can be short, can work with a small sample, and can look for in-depth data analysis and interpretation (Fischer, 1998). Furthermore, in post-positivism, the



quantitative nature is not totally absolved, but it rather advocates for multiple approaches or dimensions of various methods like triangulation (Fischer, 1998; Panhwar, Ansari & Shah, 2017). Post-positivism is regarded as a mixed paradigm, a mixture of positivism and interpretivism (Panhwar, Ansari & Shah, 2017). The interpretivism paradigm is explained in the subsequent section (section 4.3.3).

### *4.3.3 Interpretivism*

The interpretivist assumption is that reality does not exist on its own, but through the influence of the senses. Unlike positivists who accept the natural world as it is, interpretivism accepts constructed realities. The interpretivist epistemological stand is based on subjectivism; hence reality is socially constructed, not found or accepted as it is. Interpretivists gather data for a prolonged duration and seek patterns within that data. Data collection, mainly through a verbal approach rather than statistical methods, means that they collect qualitative data rather than quantitative data (Gall, Borg & Gall, 1996; Roller, 2016). Examples of interpretivists' qualitative data include interviews, observations, field notes and documents (Rehman & Alharthi, 2016). The qualitative methods of data collection are utilised well under the umbrella of interpretivism.

### *4.3.4 Critical theory*

Critical theory, originating from various philosophers, claims that reality is shaped by the interaction of history, culture, politics, gender, ethnicity and religion (Guba & Lincoln, 1994). In critical theory research, the beliefs of the researcher influence the participants or the overall study and the objects under study are interconnected. This means that the mindset of an individual in critical theory research can be adjusted by those in a position of power (Brooke, 2002). Research under critical theory is primarily conducted by those who are in political positions; their research is to be evaluated from a critical viewpoint (Kincheloe & McLaren, 2005). The methodological approach under critical theory is dialogical (Botha & Taylor, 2022).

### *4.3.5 Pragmatism*

The philosophical assumptions in pragmatism state that past experiences, which include beliefs, form part of human action (Kaushik & Walsh, 2019). Pragmatists believe that knowledge cannot be viewed as a reality but is socially constructed through human experience, meaning that assumptions are similar to constructivism

which also relies on linking the past experiences to learn new content. Pragmatism is action-oriented: thought processes always leads to a specific action (Pansiri, 2005; Goldkuhl, 2012). This assumption further states that each action has its own consequences and the meanings can be derived from such consequences. In pragmatism, both reality and world are non-static as action constantly changes everything.

Pragmatism is typically for practical-based research which strives to solve practical, real-world problems (Maxcy, 2003). Moreover, pragmatism offers a flexible methodological approach and research design to conduct a study which is guided by the nature of the scientific investigation.

#### *4.3.6 Adopted research paradigm*

I have discussed what each research paradigm entails. The discussion led to the insight to adopt a relevant research paradigm that can guide and inform this study. There are two paradigms which are close to the qualitative nature of this study, that is the interpretivism and pragmatism. Based on the nature of the study, it is more appropriate to adopt the pragmatism research paradigm because this allows flexibility of adopting a relevant research methodology based on the nature of the research question rather than being restricted to methodologies of an adopted paradigm (Brayman, 2006; Goldkuhl, 2012). The applicability of pragmatism in this study allows the adoption of a qualitative approach while making use of quantitative means if deemed necessary. The study is primarily based on a qualitative approach which focuses on the analysis of SIPS algorithm exercises during action research cycles (discussed in section 4.4.2).

### **4.4 Research design**

Research design concerns how the whole scientific study is logically organised for answering research questions, including research strategies and direction (De Vaus, 2001; Saunders, Lewis & Thornhill, 2007). The following sections present the design flow of this study.

#### 4.4.1 Animation programs design

Chapter 3 (theoretical framework) has discussed and adopted CTML by Mayer (2005). The theory discusses how the human mind works in learning with the assistance of multimedia material or animation. The design of our TLPAP animation programs adopts 12 principles of multimedia as prescribed by Mayer (2005), which are applicable and compatible with the TLPAP framework. The following section explains each principle of multimedia learning, indicating its applicability and implications in the TLPAP animation design.

- a *Coherence principle*. This principle states that the multimedia material should not be *extraneous* as it can distract learning of the main content.

TLPAP applicability: The principle is applied in the design of TLPAP animation programs by ensuring no inclusion of extraneous material that may distract learning within the animation programs. The visuals and text within the TLPAP animation programs are simple. More importantly, the animation design has been deliberately simplified because the target is SIPS.

- b *Signalling principle*. Most of the time, the screen can be populated with various animations; therefore, this principle is about signalling the learner to a specific animation or activity at a time to ensure that students do not get lost during a presentation.

TLPAP applicability: The animation program in the TLPAP draws the attention of the learner by an arrow or highlighting the currently executing line of code with blue colour. Furthermore, an animated scenario is a result of a pointed or highlighted line in a program code section.

- c *Redundancy principle*. The inclusion of visuals, narration and text within the animation can confuse the learner. This must be avoided as it can divide the learner's attention to know whether to focus on narration or text in support of visuals.

TLPAP applicability: To avoid confusion, TLPAP relies on narrative media by the actual educator (also adopted from the conversational framework) during animation program visualisation. Text is included but is minimal, short and part of program code or animation program. The pre-preparation of animation, code explanation and repetition of animation demonstration as prescribed through pedagogical guidelines (discussed in Chapter 7 section 7.4) play important role in text (or program code) clarity.

- d *Spatial contiguity principle.* The space between a text and the corresponding visuals or animation may increase cognitive load and be difficult for some students to decipher. The text and the corresponding visuals must be located strategically next to one another.

TLPAP applicability: All relevant text has been placed next to corresponding visuals to ease comprehension.

- e *Temporal contiguity principle.* This principle emphasises that a human being learns much better when narration and animation happen simultaneously. The method is preferred over having animation first followed by a separate narration about the animation.

TLPAP applicability: The principle is properly applied in the TLPAP. The educator narrates as the animation unfolds or progresses and both are in sync. Occasionally the narration is in the form of an embedded pre-recorded audio; however, for TLPAP, the narration is done live by the educator at that moment in time.

- f *Segmenting principle.* This principle states that visuals should be in segments and should be paced accordingly.

TLPAP applicability: TLPAP segmented animation programs into concepts. For instance, the animation program on introductory concept is separate from the animation program on decision statements. Each concept is further segmented

into other sub-segments like the animation for introducing a concept and the animation for presenting the actual concept. Furthermore, an adjusting option has been placed into the animation program whereby it is possible to set the pace of the animation program to animate slow, very slow or normal.

- g *Pre-training principle.* This principle is about teaching students' the basics first before introducing the main concepts through animations as this can overwhelm students if it happens suddenly. This can include key concepts, terms and even basic information about teaching and learning with animations or visuals.

TLPAP applicability: Each concept is started with simple animation programs for concept introduction which equates to minimal animation. As a result, the students progress smoothly into the main concept animation following the basic prerequisite information on how animations worked on the introductory stage.

- h *Modality principle.* This principle emphasises that a pair of visuals and narration are more productive in learning than a pair of visuals and text. In the case of visuals and narration, the cognitive load is distributed through both verbal and visual channels.

TLPAP applicability: The narration is done during the demonstration of animation program and emphasised the clarity of visuals in the design of our animation programs.

- i *Multimedia principle.* This principle emphasises that pictures and words result in more meaningful learning than words or narration alone.

TLPAP applicability: A combination of relevant animations and text in the TLPAP have been included. However, as indicated previously, text is minimal.

- j *Personalisation principle.* This principle is about adjusting the tone or language of teaching and learning into lower formality as it encourages learner

engagement more. It states that conversational learning or informal learning improves learning and comprehension.

TLPAP applicability: The personalisation principle is applied in TLPAP; hence, the adopted conversational framework (discussed in Chapter 3, section 3.2.7). This is achieved by delivering conversational narration during animation program presentations and between segment breaks.

- k *Voice principle.* According to this principle, a recorded human voice is more effective than computer-generated audio.

TLPAP applicability: A real voice that narrates during the animation program is used. Moreover, a recorded human voice can be included and embedded in the animation program when needed.

- l *Image principle.* A talking head video is not an effective learning experience. It is recommended that learning be incorporated within the text, visuals or audio rather than in a talking head video.

TLPAP applicability: The TLPAP animation programs do not incorporate a talking head video, as it relied on text, visuals and narration.

The above principles have shaped the design of the animation programs and are instrumental for effectiveness of the TLPAP. The same principles have also shaped the formulated set of pedagogical guidelines (in Chapters 6, 7 and 8) which forms part of TLPAP. The guidelines evolved during the action research cycles (from cycle 1 to cycle 5). The following section discusses the action research methodology.

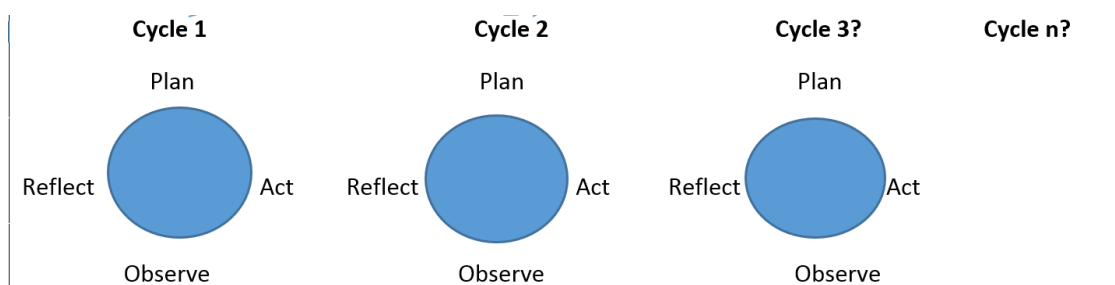
#### ***4.4.2 Action research***

The research methodology followed in this study is an action research approach (Creswell, 2003). The action research approach is a type of qualitative research that also fits in the pragmatism paradigm. Kaushik and Walsh (2019:2) explain that the “focus is on transforming the lives of socially marginalised populations”. Therefore, action research is relevant for this study as teaching and learning will be applied in the

real world to assist SIPS enrolled in the introductory programming modules. Because the solution is applied in the educational setup, theory (research) and practice (intervention in practice), the implementation of the solution in the form of action research is clearly relevant (Kearney, Wood & Zuber-Skerritt, 2013). This will help us to improve the TLPAP teaching and learning guidelines and relevant animation programs towards a fully complete framework.

There are two forms of action research, namely practical action research and participatory action research (Creswell, 2003). Participatory action research focuses on life-enhancing changes. This study, however, takes the practical action research route because it seeks a solution to improve practice by focusing on students, educators and methods of teaching and learning (Creswell, 2003). Practical action research in educational environment allows the educators to reflect on their teaching approaches, to continually improve teaching practices, to explicit about the practical activities and test new solutions (Newton & Burgess, 2008; Salite, 2008; Elliott, 2011). Furthermore, educational action research works along with the constructivism learning theory (Aldridge, Fraser & Sebela, 2004; Zehetmeier et al., 2014). As discussed in Chapter 3, section 3.2.1 and section 3.6, constructivism has been adopted as one of the theories that underpin or frame this study.

Figure 4.1 clarifies the action research process and cycles.



**Figure 4.1: Action research cycles**

At the beginning of each action cycle, the plan is provided for the new cycle of action research. The implementation of the outlined plan is the action in the action research. During action research process, the researchers do observation which is necessary for re-planning and reflections. Reflections is done at the end of each cycle and is based on the activities and overall results obtained during action cycle. When the re-

planning for the new cycle happens, revised reflections are continually put into practice.

#### *4.4.3 Experimental setup, pre-testing and post-testing of SIPS*

To check if the proposed TLPAP yields improved results during implementation, a relevant experimental setup is adopted. Firstly, I invited all students through an invitation letter placed in the learning management system. The letter emphasised that attendance is voluntary but recommended and targeted at the SIPS. The emphasis on targeting SIPS in the recruitment strategy is important because the pre-test and post-test methodology requires the actual SIPS. If the invitation was extended to all students, the risk would be that minimal difference between pre-test and post-test results could have occurred which would fail to indicate anything about the experimental progress of TLPAP. For example, the pre-teaching and post-teaching algorithms of a non-SIPS would have no difference, leaving nothing to analyse or to compare patterns of improvement within the algorithms. However, I have noted within the pre-teaching and post-teaching algorithms that a few non-SIPS still participated in the study. The identification of non-SIPS participants helped us to exclude such patterns in the analysis of results. The exact criteria to filter SIPS and non-SIPS participants is discussed in section 4.5 under the “R→R” transition.

The researcher taught in the experimental group and other introductory programming lecturers taught in the control group. As per educational practical action research methodology, it was important for the researcher to be in the experimental group to experiment, experience and observe a teaching practice which then helped in re-planning and reflection of the subsequent action cycles. Furthermore, the principal researcher was responsible to evaluate both pre-teaching and post-teaching algorithm exercises. As a result, the principal researcher was able to plan and implement each cycle of action research based on the outcome of the previous cycle analysis.

The pre-test (pre-teaching algorithm exercises) approach allowed us to test the state of students understanding before the implementation of a planned intervention. The post-test (post-teaching algorithm exercises) approach allowed us to check the state of students understanding after the intervention was carried out. An algorithm is an alternative reference to ‘program code’ and is sometimes used interchangeably in this



study. The plan behind the teaching algorithm exercises meant to find possible patterns of improvement, non-improvements or deteriorating learning between pre-teaching and post-teaching algorithms. The use of pre-test and post-test methods to qualitatively compare two groups is not new. The work of Levy, Ben-Ari and Uronen (2000); Osman and Elmusharaf (2014); Shi, Min and Zhang (2017); and Végh and Stoffová (2017) have also made use of pre-test and post-test methodological approaches.

The following section discusses the background of the participants.

#### *4.4.4 Background of the participants*

The SIPS were from the Faculty of Information and Communication Technology (ICT) at Tshwane University of Technology (TUT). Some students were in the Information Technology (IT) course (using C++ programming language), some were in the Computer Systems Engineering course (using C++ programming language) and some in the Computer Science (CS) course (using Java programming language). All animation programs were made for C++ programming language since two courses used C++ as an introduction to programming language and few animation programs have the option to demonstrate with C++ code or Java code (and are adaptable to include other programming languages). The few animation programs adapted to Java were limited to the concepts offered as introductory programming in the CS course.

TUT uses certain achievements levels/points of subjects to admit students into courses. If the grade percentage of a subject is in the range of 0% to 29%, the achievement level is 1, a range of 30% to 39% is equated to achievement level 2, a range of 40% to 49% is level 3, a range of 50% to 59% is level 4, a range of 60% to 69% is level 5, a range of 70% to 79% is level 6 and finally a range of 80% to 100% is level 7. The admission requirements for the Computer Systems Engineering diploma at TUT are as follows: the students should have at least level 3 in English, level 5 in mathematics or technical mathematics, level 4 for physical science or technical science and Admission Points Score (APS) of 28. APS score is the sum of the achievement levels of the required subjects to be admitted into a course. The students who do not meet these requirements can qualify for the foundation program. To be admitted into the foundation program a student needs a level 3 in English, level 4 in mathematics or technical mathematics, level 4 in physical science or technical physical

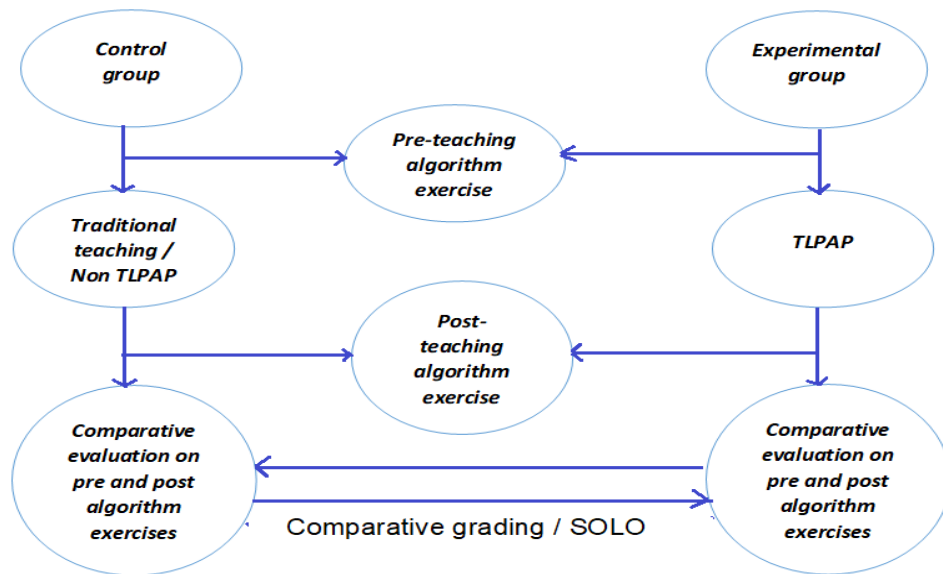
science and an APS of 23. The IT qualification requirements include level 3 in English, level 4 in mathematics or level 5 in mathematics literacy. An APS score of 18 is required with mathematics as part of the subjects and APS of 20 with mathematics literacy as part of the subjects for an admission into an IT qualification.

Note that other background information like age and gender were not considered in this study. The following section discusses the control and experimental group setup.

#### ***4.4.5 Control and experimental groups setup***

The SIPS were randomly and equally divided into two groups to implement the practical action research cycles. In the case where the total participants were odd numbered or some participants had arrived late then one of the groups had more students. The first group is referred to as *experimental* and the second group as *control*. The experimental group was taught by following the TLPAP framework. The control group was taught the same concepts as the experimental group, but without following the TLPAP framework (through the traditional verbal teaching method). Note that the participants (students) would have already been taught by their various lecturers, then the struggling students are invited and divided into two groups (control and experimental) to get additional teaching in this study.

The post-teaching exercise is given to SIPS after the TLPAP has been followed in the experimental group and traditional learning in the control group. The teaching session that occurs between the pre-teaching and post-teaching algorithm exercises in the control and experimental group is based on the pre-teaching algorithm exercises. This is because the study is operating under a CA framework (discussed in section 3.5) which emphasises that teaching, learning, assessments and learning outcomes should be aligned. The teaching sessions in both groups happened on the same day and time of the week. Figure 4.2 demonstrates the overall processes on both experimental and control groups.



**Figure 4.2: A process for both experimental and control groups**

Figure 4.2 shows that the control group writes a pre-teaching algorithm exercise, followed by a teaching session, then writes a post-teaching algorithm exercise. The same process occurs in the experimental group except they are taught by following the TLPAP framework. The SOLO-based evaluation of the pre-teaching and post-teaching algorithm and between control and experimental groups is explained in detail in the following section.

## 4.5 Evaluation strategy

There are many evaluation strategies that can be employed; however, a specific strategy that fits the nature of the study and answers research questions must be determined. In this study, the SOLO taxonomy (Biggs & Collis, 1982) is adopted and adapted to benchmark or evaluate the pre-teaching and post-teaching algorithms. SOLO allows a qualitative evaluation which investigates the whole understanding in the response rather than just assessing a program code line by line (Izu, Weerasinghe & Pope, 2016). Similarly, we adapt SOLO stages to evaluate, analyse and benchmark the algorithm sections. The first stage of SOLO (prestructural) means that the student does not understand at all. The second stage (unistructural) means a student can do one thing which cannot be related to anything else. The third stage (multistructural) means the student can get several things correct but these are still unrelated. The fourth stage (relational) means a student can get several things correct and is able to

relate them. The last stage (extended abstract) is when the students use relational stage knowledge to create or formulate new things. Table 4.1 explains how Lister et al. (2006) and Izu, Weerasinghe and Pope (2016) adopted SOLO into the programming environment to evaluate student outcome.

**Table 4.1: Adapted SOLO by Lister et al. (2006) and Izu, Weerasinghe and Pope (2016)**

SOLO stages	SOLO-based reading and understanding of program code by Lister et al. (2006)	SOLO-based classification of program code by Izu, Weerasinghe and Pope (2016)
Prestructural	Lack of basic programming knowledge	No single concept is related to specifications or unable to select specific design patterns to form a program code
Unistructural	Described one line of code and cannot explain the rest of the program code	Based on the specifications, direct interpretation is applied
Multistructural	Described some of the program code line by line	Based on the specifications, the interpretation is flexible
Relational	Describe what the code does in its entirety and its purpose	Based on the scenario, the program code is well structured accordingly
Extended abstract	<i>Not applicable</i>	Based on the scenario, the program code is well structured and has elements of abstraction beyond the specifications

In this study, I adopt the work of Biggs and Collis (1982) on SOLO to qualitatively evaluate student algorithms. The following SOLO-based evaluation matrix (Table 4.2) has been adapted for a thorough analysis of the pre-teaching and post-teaching algorithms. In the table, the P stands for prestructural, U for unistructural, M for multistructural and R for relational. This matrix is set to evaluate the entire algorithm or a section of an algorithm for a deeper meanings and interpretation.

**Table 4.2: SOLO-based evaluation matrix**

<b>P</b>	P→P	U→P	M→P	R→P
<b>U</b>	P→U	U→U	M→U	R→U
<b>M</b>	P→M	U→M	M→M	R→M
<b>R</b>	P→R	U→R	M→R	R→R
	<b>P</b>	<b>U</b>	<b>M</b>	<b>R</b>

The transition (*example P→U*) in Table 4.2 means that the first value (P) refers to the evaluation status of the pre-teaching algorithm or a section of the pre-teaching algorithm which is placed at the prestructural stage. The first value can also be referred to as the state of SIPS before interventions as it represents the pre-teaching algorithm. The second value (U) in **P→U** means that the status of a post-teaching algorithm or

section of a post-teaching algorithm is placed at the unistructural stage. The following table briefly describes each transition to make the interpretation or feedback more meaningful.

**Table 4.3: SOLO-based evaluation matrix (with description)**

<b>P</b>	<b>P→P</b> <i>Stagnant-no-learning</i>	<b>U→P</b> <i>Lower-level-deterioration</i>	<b>M→P</b> <i>Intermediate-deterioration+</i>	<b>R→P</b> <i>Deteriorated++</i>
<b>U</b>	<b>P→U</b> <i>Lower-level-improvement</i>	<b>U→U</b> <i>Stagnant-at-lower-level</i>	<b>M→U</b> <i>Intermediate-deterioration</i>	<b>R→U</b> <i>Deteriorated+</i>
<b>M</b>	<b>P→M</b> <i>Intermediate-improvement+</i>	<b>U→M</b> <i>Intermediate-improvement</i>	<b>M→M</b> <i>Stagnant-at-intermediate-level</i>	<b>R→M</b> <i>Deteriorated</i>
<b>R</b>	<b>P→R</b> <i>Improved++</i>	<b>U→R</b> <i>Improved+</i>	<b>M→R</b> <i>Improved</i>	<b>R→R</b> <i>Already-know</i>
	<b>P</b>	<b>U</b>	<b>M</b>	<b>R</b>

### **P→P**

*Transition description:* Stagnant-no-learning

*Meaning:* The effects of teaching and learning methods are not realised.

### **P→U**

*Transition description:* Lower-level-improvement

*Meaning:* This means that learning has taken place but at a small scale. The impact of a teaching method is noticeable but with little improvement.

### **P→M**

*Transition description:* Intermediate-improvement+ (The + symbol means, moved 2 stages up)

*Meaning:* This means that learning has taken place at an acceptable rate. The impact of a teaching method is noticeable with good improvements, but still lacks a few things or logical skills to complete the algorithm or a section in an algorithm.

### **P→R**

*Transition description:* Improved++. (The ++ symbol means, moved 3 stages up)

*Meaning:* This means that learning has taken place at a high rate because of the distance between the prestructural and relational stages. The impact of a teaching

method is noticeable with great improvements and the student can form one logical whole in an algorithm.

### **U→P**

*Transition description:* Lower-level-deterioration

*Meaning:* This means that the student is confused or has unlearned; hence, unistructural to prestructural is in the reverse direction. It may also mean that the teaching methods may be doing more harm than good.

### **U→U**

*Transition description:* Stagnant-at-lower-level

*Meaning:* Learning methods used do not make any difference as improvements are not realised through higher stages like U→M or U→R. This may also mean that the teaching method has no effect at all.

### **U→M**

*Transition description:* Intermediate-improvement

*Meaning:* This means that learning has taken place at a good scale. The impact of a teaching method is noticeable with significant improvements. Patterns of existing knowledge about one thing have improved into the upper stage.

### **U→R**

*Transition description:* Improved+ (*The + symbol means, moved 2 stages up*)

*Meaning:* This means that learning has taken place on a great scale. The impact of a teaching method is noticeable with great improvements. Patterns of existing knowledge about one thing have improved two stages up to reach the ultimate level.

### **M→P**

*Transition description:* Intermediate-deterioration+ (*The + symbol means, moved 2 stages down*)

*Meaning:* This means that the student is confused or has unlearned; hence, multistructural to prestructural is in reverse direction (two stages down). It may also mean that the teaching methods may be doing more harm than good.

### **M→U**

*Transition description:* Intermediate-deterioration

*Meaning:* This means that the student is confused or has unlearned; hence, multistructural to prestructural is in reverse mode (one stage down). It may also mean that the teaching methods may be doing more harm than good.

### **M→M**

*Transition description:* Stagnant-at-intermediate-level

*Meaning:* Learning methods do not make any difference as improvements are not realised through higher stages like M→R. This may also mean that the teaching method has no effect at all.

### **M→R**

*Transition description:* Improved

*Meaning:* This means that learning has taken place on a great scale. The impact of a teaching method is noticeable with great improvements. Patterns of existing knowledge about several things have improved into 1 stage up.

### **R→P**

*Transition description:* Deteriorated++ (The ++ symbol means, moved 3 stages down)

*Meaning:* This means that the student is confused or mistakenly unlearned; hence, multistructural to prestructural is in reverse mode (three stages down). It may also mean that the teaching methods may be doing more harm than good. However, this is unlikely transition.

### **R→U**

*Transition description:* Deteriorated+ (The + symbol means, moved 2 stages down)

*Meaning:* This means that the student is confused or mistakenly unlearned; hence, multistructural to prestructural is in reverse mode (two stages down). It may also mean that the teaching methods may be doing more harm than good.

### **R→M**

*Transition description:* Deteriorated

*Meaning:* This means that the student is confused or has unlearned; hence, multistructural to prestructural is in reverse mode (one stage down). It may also mean that the teaching methods may be doing more harm than good.

### **R→R**

*Transition description:* Already-know

*Meaning:* This means that the student was not struggling at all with that concept or all sections of the algorithms (pre-test and post-test) are correct. It also means that the teaching methods were unnecessary as the student was not struggling prior to intervention.

In summary, the desired transitions which signal improvements are  $P \rightarrow U$ ,  $P \rightarrow M$ ,  $U \rightarrow M$ ,  $P \rightarrow R$ ,  $U \rightarrow R$  and  $M \rightarrow R$  and (highlighted in gold in Table 4.3). The transitions which show no improvements prior and after interventions are represented by  $P \rightarrow P$ ,  $U \rightarrow U$ ,  $M \rightarrow M$  (highlighted in grey in Table 4.3). The undesired or negative transitions that mean anti-learning or the teaching method is doing harm to learning are  $U \rightarrow P$ ,  $M \rightarrow P$ ,  $R \rightarrow P$ ,  $M \rightarrow U$ ,  $R \rightarrow U$  and  $R \rightarrow M$  (highlighted in blue in Table 4.3). The  $R \rightarrow R$  means the student knew before and after the teaching and learning intervention, so improvements cannot be linked to the specific intervention.

Upon the qualitative evaluation through the SOLO-based evaluation matrix, the number of algorithms sections or students per transitions are represented quantitatively in the form of a percentage. Example, a total of 20 struggling students wrote the pre-teaching and post-teaching algorithm exercises. In the case of evaluating the loop section in that algorithm, I find that 12 students are in multistructural stage of both the pre-teaching and post-teaching algorithm. This means that 12 students over a total of 20 struggling students are under  $M \rightarrow M$  transition. The outcome of  $12/20$  is 60% when represented as a percentage. Let's assume that 6 students were in the multistructural stage of the pre-teaching exercise and are in the relational stage of the post-teaching exercise. This means the transition will be  $M \rightarrow R$  and 6 students over a total 20 students will be represented by 30%. Lastly, if 2 students are in a relational stage of the pre-teaching and post-teaching exercise, then the transition will be 10% ( $2/20$ ). The rest of the algorithm sections are calculated the same way. At the end, I can see that in the loop section of the algorithm, 60% is  $M \rightarrow M$  transition, 30%  $M \rightarrow R$  transition and 10%  $R \rightarrow R$  transition. The outcome presents both qualitative and quantitative interpretation, because I will know which transitions have emerged, remained the same, deteriorated or improved upon evaluation.



## 4.6 Action cycles progression

This section presents an overview of what happened in each cycle of the action research.

### 4.6.1 *Timeline overview*

In order to provide an overview of the timeline, the information is presented chronologically per semester. The actual implementation of this action research started in 2019 and was concluded in 2022. The following section briefly highlights the activities in each semester or cycle.

**2019 Semester 1:** Identification of manipulatives to teach programming concepts.

**2019 Semester 2:** Implementation of physical manipulatives on IPC. Cycle 1.

**2020 Semester 1:** Implementation of physical manipulatives on IPC. Cycle 2.  
Interrupted by Covid-19 and the limitations of physical manipulatives.

**2020 Semester 2:** Covid-19. No experimental work undertaken.

**2021 Semester 1:** Transition from physical manipulatives to virtual manipulatives (animation programs initially referred to as virtual manipulatives). Cycle 2 continued.

**2021 Semester 2:** Animation programs experimentation. Cycle 3.

**2022 Semester 1:** Animation programs experimentation. Cycle 4.

**2022 Semester 2:** Animation programs experimentation. Cycle 5.

### 4.6.2 *Cycle 1*

The initial plan in this study was based on the use of physical manipulatives for teaching and learning the IPC to the actual affected SIPS. The experimentation with physical manipulatives happened in 2019 semester 2 and 2020 semester 1. Prior to the implementation, the journey was about identifying potential existing manipulatives or designing new manipulatives to help in teaching and learning of IPC. The emphasis in the first cycle was on the fundamentals of introductory programming like the use of *assignment* to demonstrate copying, storage and overwriting. A complete report on

the planning, action, reflection and observation of results is detailed in Chapter 5 (Initial framework – Cycle 1).

### **4.6.3 Cycle 2**

The shortcomings, reflections and observations of cycle 1 helped us to improve the teaching and learning methods that were used for teaching and learning in this cycle. There was success in using physical manipulatives to emphasise storage, copying and overwriting and I had planned to advance into the next topic which was decisions and nested decisions. However, I experienced various shortcomings on the attempt to use physical manipulatives for teaching other IPC. Such shortcomings include incompatibility to teach in a large class, in the online platform and Covid-19 restrictions. This was due to the fact that the cycle was implemented in 2021 semester 1, during which time Covid-19 was still in effect.

As a result, a virtual version of the physical manipulatives were developed by creating an animated program. The animated programs covered IPC (the use of *assignment* to emphasise copying, storage and overwriting), decisions, nested decisions, loops and nested loops. The presentation mode of this cycle was online. The animation programs were designed using a Rapid Application Development C++ Builder 10.4. The details of the shortcomings, reflections, observatory results and improvements made in this cycle are presented in Chapter 6 (Adapted framework – Cycle 2).

### **4.6.4 Cycle 3**

In 2021 semester 2, cycle 3 was implemented with animation programs only and used a set of pedagogical guidelines that were compatible with TLPAP. In this cycle, the animation programs for teaching and learning functions and arrays were included. The improved versions of animation programs for introductory concepts, decisions, nested decisions, loops and functions were experimented on SIPS. The details of the shortcomings, reflections, observatory results and improvements in this cycle are presented in Chapter 7 (Adapted framework – Cycle 3).

### **4.6.5 Cycle 4**

This cycle was implemented with the improved methods or guidelines for TLPAP. The cycle took place in 2022 semester 1 and the details of the shortcomings, reflections,

observatory results and improvements in this cycle are presented in Chapter 8 (Adapted framework – Cycles 4 and 5).

### 4.6.6 Cycle 5

This cycle (2022 semester 2) was also implemented with improved methods based on the reflections and observations of cycle 4. In this cycle, there were extra animation programs with an option to switch to a Java program code. The details of the shortcomings, reflections, observatory results and improvements in this cycle are presented in Chapter 8 (Adapted framework – Cycles 4 and 5). This cycle had minimal adaption, as the framework has attained a satisfactory stage. The final framework is also presented in the Chapter 8.

## 4.7 Limitations of the study

The limitations of this research are as follows:

- The invitation to participate in the study was limited to SIPS; consequently, the sample size was small per cycle. To compensate, I have undertaken multiple cycles and qualitative interpretation of the algorithms.
- The animation programs were designed for the C++ programming language and a few have been extended to cover Java programming language in the last cycle. Other programming languages like Python or C are not included.
- Content of the programming module was limited to what the university (where the experiments took place) included within the course: for example, some module content covers only introduction, decisions and loops.
- We have not made any distinction on the gender disparity of the participants. Other background factors like ethnicity, age and home language were also not taken in to account in this study.

## 4.8 Ethics statement

The ethical clearance to conduct this study was obtained from University of South Africa (UNISA). UNISA is the institution where the PhD programme is registered, and TUT is the institution where the experimentation was conducted. The main point in the ethical clearance certificate is that the identity of the participants is not made public

and participation in this study is voluntary. It also emphasises that the division of SIPS into control and experimental group is random across action cycles. The ethics clearance certificate is attached in Appendix A.

## 4.9 Conclusion

The research methods described in this chapter are synchronous with our research question and research objectives. Table 4.4 overviews the linking and alignment of research questions and objectives to a relevant research method.

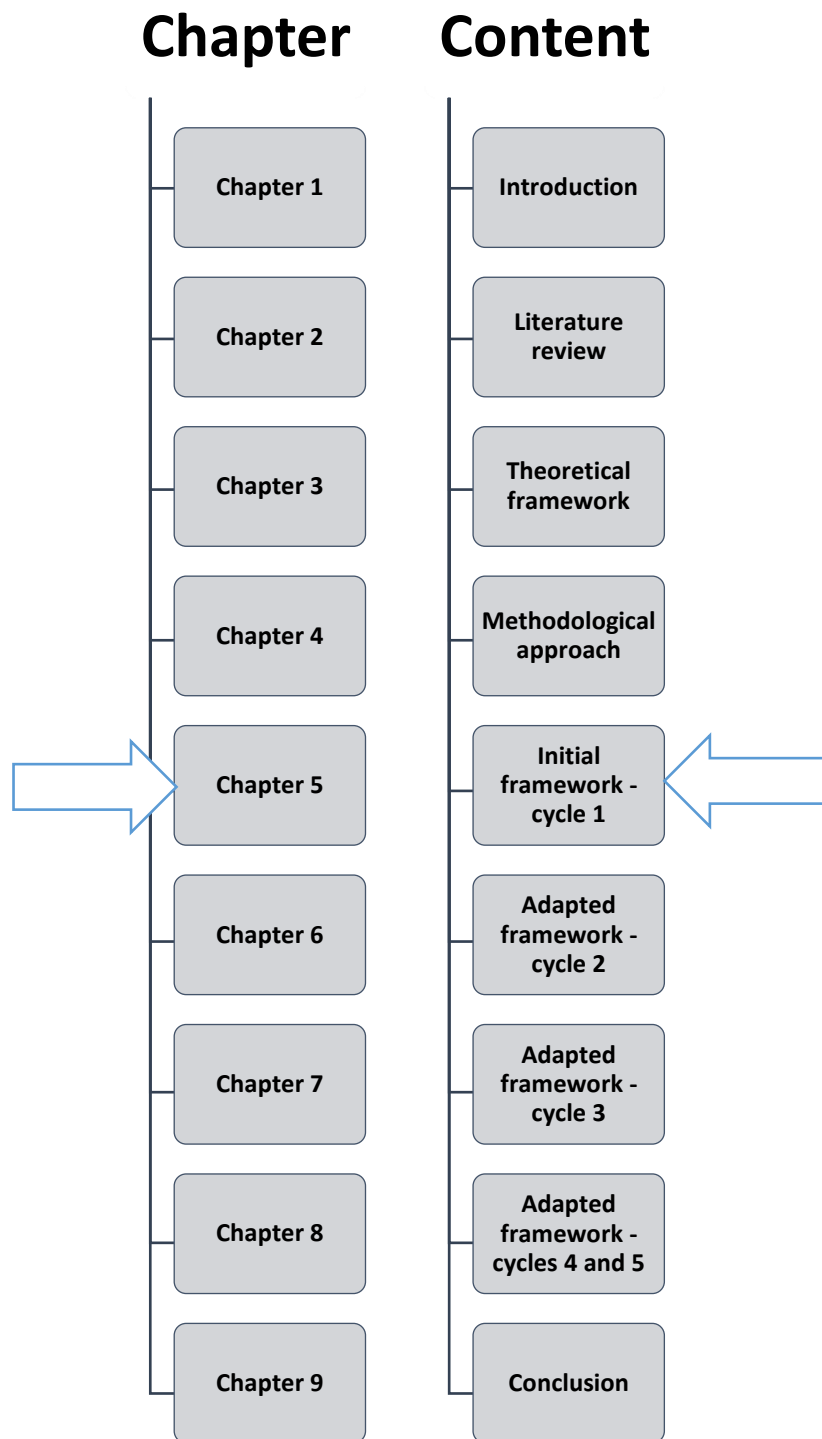
**Table 4.4: Methodological overview**

	<b>Research questions</b>	<b>Research objectives</b>	<b>Method</b>
1.	What are the leading issues in teaching and learning IPC?	To conduct a literature review on the challenges of teaching and learning IPC	Literature review
2.	What are the approaches used in teaching IPC?	To investigate the approaches used in teaching IPC	Literature review
3.	What are the limitations of existing animation programs used for teaching and learning IPC?	To investigate animation programs for teaching and learning IPC	Literature review
4.	How can we develop a TLPAP framework for teaching and learning IPC?	To develop a TLPAP framework for teaching and learning IPC based on gaps identified in the literature review and continually improve the framework based on the outcome of the action research cycles	Theoretical framework/ Action research
5.	How can we prove and evaluate the efficacy of the TLPAP framework?	To evaluate the efficacy of the TLPAP framework through teaching the actual SIPS by following the TLPAP framework	Action research/pre-test and post-test methodology /SOLO adapted evaluations

The TLPAP framework was developed by following the research methodology described in this chapter. The critical steps toward building the TLPAP framework derived from the action research process where I reflected, observed, planned and acted in each cycle. This means that in every cycle, TLPAP was implemented with an improved and revised approach. Because TLPAP consists of pedagogical guidelines and animation programs, the improvements affected both. The exact details of what

was improved or observed in each action cycle are detailed in Chapters 6 to 8. The subsequent chapter (Chapter 5) presents the Initial framework – cycle 1.

# Chapter 5: Initial framework – Cycle 1



## 5.1 Introduction

The aim of this research was to assist Struggling Introductory Programming Students (SIPS) by finding ways to improve and accelerate their understanding of IPC. Initially, the proposed framework was centred around the use of manipulatives to help in teaching and learning of IPC to SIPS. As discussed in Chapter 2 (Literature review, section 2.6.4), physical manipulatives are incorporated into education for teaching and learning purposes. However, there is no concrete, empirical evidence of their reliability in uplifting student comprehension of IPC. The manipulatives can potentially cause more harm than good if their use is not rigorously investigated, tested and justified. I therefore embarked on a research journey to identify possible manipulatives and test their efficacy in helping SIPS learn IPC.

The main goal in this initial cycle was to identify and test existing manipulatives already in use by the educators or design new manipulatives where necessary. This is the first stage of answering the research question which states:

- How can we prove and evaluate the efficacy of the Teaching and Learning Programming with Animation Programs (TLPAP) framework?

The framework in this initial cycle is referred to as Teaching and Learning Programming with Manipulatives (TLPM). The name was adapted as we progressed through the action research cycles and was then modified to TLPAP. In this chapter, I present and explain the manipulatives used and activities of the first cycle of action research and how they led to a transition into animation programs. The remainder of the chapter is outlined as follows: section 5.2 is the planning of cycle 1; section 5.3 is the implementation of cycle 1; section 5.4 presents the planning of cycle 2; and section 5.5 concludes the chapter.

## 5.2 Planning for cycle 1

### *5.2.1 SIGCSE members' participation*

To identify and test appropriate manipulatives, I explored if there are introductory programming educators who are using manipulatives in teaching introductory programming. In February 2019, I posted a question in the Special Interest Group on Computer Science Education (SIGCSE) mailing list to gather evidence and information

on the uses of manipulatives to teach introductory programming. SIGCSE is an Association of Computing Machinery (ACM) sub-group with more than 2 000 members. It consists of Computer Science Education (CSE) researchers, academics, educators, teachers and experts in the field of education and industry. SIGCSE's special focus is on CSE research with researchers from over 60 countries. The following question was posed on the SIGCSE mailing list:

*If anyone has tried using manipulatives in teaching programming concepts (or knows of anyone else who has), could they please email xxxxx@xxxx? Where xxxxx@xxxx was the researcher's address. Posted in February 2019.*

I allowed more than 15 days for responses. In that time, there were 21 concrete responses from academics, CS researchers and former introductory programming educators. Based on the SIGCSE membership, this is a sample of 0.7% because introductory programming is a small portion of the broader computer science education. Some of the responses advised on the type of programming manipulatives that can be used, some responses pertained to manipulatives they were currently using, and other responses recommended manipulatives they had used in the past to teach some IPC. The suggestions ranged from manipulatives that help in sorting, swapping, linked lists, variables, binary searching, arrays, functions, recursion and other interactive activities. For example, cardboard was used to set up manipulatives for demonstrating the use of functions in programming. These results helped us to identify the most suitable manipulatives for experimental purposes with the actual SIPS at a university. The results of such experimentation are reported in section 5.3.

### ***5.2.2 Identification of manipulatives for introductory concepts***

The focus in this section is on teaching and learning the use of *assignment* because the researcher had experimented with the methods and was successful. The focus is also on teaching and learning of *decisions/nested decisions* because the tested methods were tested but not successful.

#### **Concept 1**

One of the misconceptions within IPC identified in the literature review is the use of *assignment*. The manipulatives used in this case were inspired by one of the SIGCSE



respondents who indicated that he was using cups (Figure 5.1) to demonstrate the swapping of values between variables.



**Figure 5.1: Sample cups**

The plan was to test if the idea of demonstrating using cups can yield meaningful learning, especially for SIPS. Therefore, I adapted the idea of cups in emphasising the uses of *assignment* between variables.

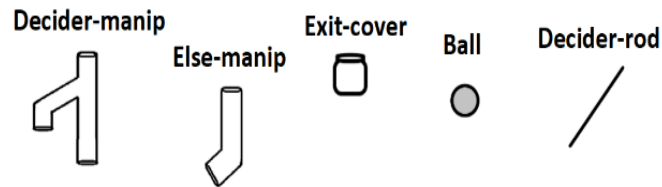
## **Concept 2**

Concept 2 was about identifying the possible manipulative to teach *decisions* and *nested decisions*. In this case, as the responses of SIGCSE members did not link the researcher to any existing manipulative, I designed a new manipulative and named it a nested-decider. The main purpose of the nested-decider manipulative was as follows:

- demonstrate how the `&&` and `||` operators work in a *decision* statement;
- demonstrate how the *else* clause works and when to use it in a *decision* statement;
- ensure SIPS understand that a nested *decision* stops evaluating the rest of the conditions if one condition evaluates to true;
- ensure SIPS understand when to construct a nested *decision* instead of a sequence of separated *decision* statements; and
- enhance SIPS overall conceptual understanding of how the nested *decision* statement works.

The nested-decider is intended to be used primarily by the lecturer to teach decisions and nested decisions to SIPS. The components of nested-decider are

presented in Figure 5.2 to initiate a full understanding of how the manipulative works.



**Figure 5.2: Components of the manipulative**

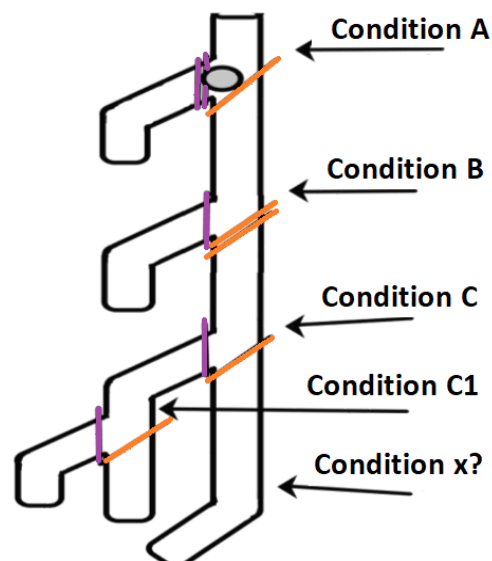
The manipulative should be used with an appropriate scenario and nested decision program code. To understand the overall manipulative functionality, the program code (depicted in Figure 5.3) and the corresponding nested-decider (Figure 5.4) were developed for teaching purposes. For example, condition 1 in the code (Figure 5.3) will be demonstrated by condition A (Figure 5.4) in the nested-decider.

```

1  if (value1 > 0 && value1 < 9) (condition1)
2  {
3      ..code
4  }
5  else if (value1 > 9 || value1 < 21) (condition2)
6  {
7      ..code
8  }
9  else if (value1 < 0) (condition3)
10 {
11     ..code
12     if (value1 == -10) (condition3.1)
13     {
14         ..code
15     }
16 }
17 else (condition x?)
18 {
19     ..code
20 }

```

**Figure 5.3: Program code**



**Figure 5.4: Nested-decider**

An important process for a lecturer to demonstrate a program in Figure 5.3 code with the nested-decider in Figure 5.4 is for the lecturer to unblock the ball path by removing decider-rods based on the conditions. Note that the purple decider-rods are for true conditions and the orange decider-rods are for false conditions. In the first part of the manipulative in Figure 5.4 (condition A), we see that two decider-rods block the tube (Decider-manip) on the left side and one decider-rod blocks the tube on the right side.

All the decider-rods have blocked the tube on the intersection within a decider-manip. When the correct pattern of removing the decider-rods is applied, the ball will automatically take the unblocked path.

*Condition 1* in Figure 5.3 represents *condition A* in Figure 5.4, because two conditions are expected to evaluate to true for the content of an if-statement to execute. Hence,  $true \ \&\& \ true = true$ . If  $value1 = 5$ , then  $value1 > 0?$  is true and the lecturer is expected to remove one of the two purple decider-rods from the left side. The ball will still be trapped on the intersection because of the remaining decider-rods on both sides. On the very same first line in Figure 5.3, the right-side condition says  $value1 < 9?$  This evaluates to true; therefore, the user should remove the second purple decider-rod, then the ball will instantly take a path to the left side. Now the lecturer is expected to be able to explain *condition 1* better with actions taken on the intersection of the manipulative. SIPS are expected to see and understand that the  $\&\&$  short-circuit operator requires both conditions to be true for the contents of the decision statement to execute. Hence the two purple decider rods that represent the true condition on the left side are both removed when the condition becomes true. SIPS should be able to see that it is now impossible for the ball to reverse to get into other tubes. Therefore, the same scenario must be explained in relation to the real programming code. Furthermore, it should be clear that the ball cannot move into the next intersection if the decider-rod on the right (orange decider-rod for false condition) is not removed. A lecturer must also emphasise that a compound decision statement such as (*if* [ $value1 > 0$ ] *if* [ $value2 < 9$ ]) can also be used to represent *condition 1*, and it will not affect how the user interacts with the manipulative. This will help SIPS play around with decision statements and manipulatives for further improvement and understanding.

This nested-decider idea was accepted and published in a CSEDU proceedings and the information about the paper can be accessed in the conference repository<sup>1</sup>.

## 5.3 Implementation of the manipulatives and results

This section reports on the implementation results of concepts 1 and 2.

---

<sup>1</sup> Ramabu, T., Sanders, I., & Schoeman, M. A. (2021). Manipulatives for Teaching Introductory Programming to Struggling Students: A Case of Nested-decisions. In *CSEDU (1)*, 505-510.

### 5.3.1 Cycle 1 (Concept 1)

#### First semester 2019

In this experiment, the focus is on teaching and learning by making use of manipulatives explained in section 5.2.2. The idea is to enhance comprehension on the use of *assignment* which is one of the introductory concepts with which SIPS struggle. To compare control and experimental groups, SIPS were given paper-based pre-teaching and post-teaching exercises. A total of 17 students (experimental group) were taught with manipulatives to alleviate the misconceptions experienced in the previous exercises and 13 students (control group) were taught without the use of manipulatives (verbal teaching without any aiding methods).

Pre-teaching exercise 1:

*Given variables  $x = 7$  and  $p = 12$ , write a C++ statement that copies the value of  $p$  into variable  $x$ . What will be the values of  $x$  and  $p$ ?*

The number of students who indicated that the value of  $x = 7$  was 5/30. Two students from the control group and three students from the experimental group indicated the value of  $x = 7$ . This suggested that they did not understand that  $x$  was assigned a new value that replaced the old value. A total of six students answered that the value of  $x = 19$ . In this case, the misconception is that the statement  $x = p$  means the value of  $x$  got incremented by the value of  $p$ ; hence,  $7 + 12 = 19$ .

Pre-teaching exercise 2:

*If  $x = 8$ ,  $y = 10$ ,  $t = 15$ , what is the value of  $x$  and  $y$  after the execution of the given code below?  $t = x$ ;  $x = y$ ;  $y = t$ ;*

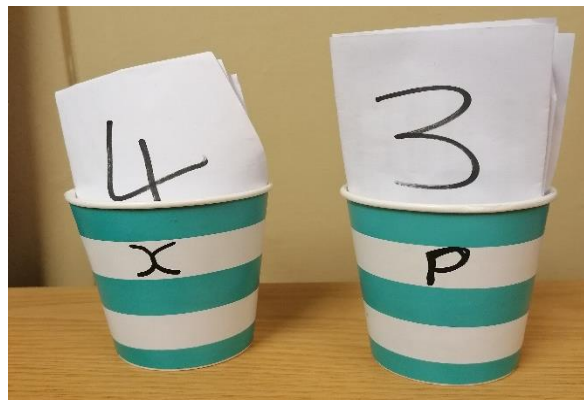
A total of 12/30 indicated in their answers that the value of  $y = 15$ . They did not understand that the value of  $t$  was overwritten with the value of  $x$ .

In the TLPM classroom, I made use of cups and papers as manipulatives to explain the pre-teaching exercise. As seen in Figure 5.5, the names of the variables were written on the side of the cups. The students were taught that this indicates variable declaration and is related to how the memory is reserved in the machine upon declaration.



**Figure 5.5: Manipulatives (variables)**

When a value is assigned into the variable, a piece of paper was used to write a value, then inserted it into the correct cup (see Figure 5.6).



**Figure 5.6: Manipulatives (with values)**

The emphasis on  $x = p$  is achieved by taking a piece of paper, looking at what is inside the variable  $p$  (the cup with  $p$  written on the side), and then writing the same value of  $p$  on the paper as a copy. However, just before I inserted the paper into cup  $x$ , took away the existing paper in cup  $x$  and replaced it with the one copied from cup  $p$ . With this demonstration, I emphasised that the value of  $p$  is copied into  $x$  and that the value in the variable  $x$  gets replaced, not incremented. It was also stressed that the value of  $p$  is not cut or removed from  $p$  as it is a copy. This first demonstration of *assignment* through manipulatives shed light and served as a crucial prerequisite to the subsequent TLPAP exercises.

In the second demonstration, I used three plastic cups to represent variables  $x$ ,  $y$  and  $t$ . I demonstrated how swapping is achieved through an *assignment*. All three cups were initialised with their respective values by writing on the pieces of paper and then inserting these into the cups (see Figure 5.7).



**Figure 5.7: Manipulatives (three cups)**

To measure the improvements between the two groups, post-teaching exercises were designed and completed by both the control and experimental groups.

Post-teaching exercise 1:

*Declare  $x$  and  $y$  as integers. Assign  $x$  with the value 20 and  $y$  with the value 30. Write a code that will store the value of  $x$  into  $y$  and the value of  $y$  into  $x$ .*

Post-teaching exercise 2:

*Students were requested to explain what happens in the memory when the code developed previously (post-teaching exercise 1) is executed. (This is a follow-up question and was asked to assess if post-teaching exercise 1 was computed with understanding.)*

Both pre-teaching and post-teaching exercises were graded. The pre-teaching questions were graded out of four, where each correct answer was counted as one mark. Similarly, the post-teaching exercise was also graded out of four where correct answers counted as a one mark. For the experimental group, the performance was better than the control group on post-teaching exercise 1 and much better on post-teaching exercise 2 (see Table 5.1).

**Table 5.1: Average grades for control and experimental groups**

Pre-teaching algorithm		Post-teaching algorithm	
Control	Experimental	Control	Experimental
Average grade: 1.1	Average grade: 1.3	Average grade: 1.08	Average grade: 2.29

The average grade mark on post-teaching exercise 1 for the experimental group was 2.29 which is higher than the control group grade average (1.08). This attests that teaching and learning using physical manipulatives could have a meaningful impact on teaching and learning, especially for SIPS. The table also shows a big difference between pre-teaching and post-teaching grades in the experimental group, which further indicates a positive increase in understanding in the experimental group. In the control group, the average grade in post-teaching is similar to the average grade in the pre-teaching. This means that the impact of a traditional teaching method is not good enough.

Finally, it is worth noting that there were two students from the experimental group who achieved more than 80% on the pre-teaching exercises and post-teaching exercises. This suggests they did not necessarily need any pedagogical intervention as they solved the exercises correctly before interventions. Therefore, this TLPM is proving to be more meaningful for struggling students.

The results of this cycle were also published in IEEE explore<sup>2</sup>.

### 5.3.2 Reflection on cycle 1 (Concept 1)

#### End of 2019 semester 1

- I noted positive results using cups and papers for teaching and learning the concept of *assignment*.
- Even though the idea of physical manipulatives worked in emphasising the concept of *assignment*, it is limited to a demonstration in a small group. It was observed that those who were seated far away moved closer to clearly see the demonstration.
- The understanding of the uses of *assignment* should be analysed in detail in the cycle 2 experimentation.
- There is a need for additional pre-teaching exercises for even greater understanding.

---

<sup>2</sup> Ramabu, T. J., Sanders, I., & Schoeman, M. (2021, May). Teaching and Learning CS1 with an Assist of Manipulatives. In *2021 IST-Africa Conference (IST-Africa)*, 1-8. IEEE.

- I also noted a frequent change of focus from the demonstration of cups and a program code which was important and required because the code was written on the board and demonstration was at a bit of distance. Therefore, a need for a mechanism to bring the demonstration and program code closer to each other was identified. The solution to the problem was to limit the frequency of changing the focus through head movements.

### *5.3.3 Cycle 1 (Concept 2) report and reflections*

#### **First semester 2021**

It is worth noting that the implementation of concept 2 was anticipated for 2020, but this did not happen because of the closure of learning institutions due to Covid-19. The experiment was supposed to be carried out at a university in a contact session environment with the actual SIPS. Concept 2 was finally attempted for demonstration in March 2021 when the learning institution opened for contact sessions; however, that opening was for a short period as the institutions reverted to online multi-modal sessions when the Coronavirus infection rate rose yet again.

When the institution re-opened, the experimentation of nested-decider was carried out in a contact session, however it did not yield positive results. The following problems were encountered during the demonstration:

- Students indicated that they struggled to pay attention or see the fine details of the manipulatives. This was a common concern as the researcher observed some students in class change seats to be nearer the demonstration point.
- The experimental group had 18 students and the control group had 17 students. Both experimental and control groups were taught without the assistance of manipulatives.
- Pre-teaching and post-teaching exercises were not given to students due to the experimental shortcomings of the nested-decider.
- A need for a solution to make the details of the nested-decider manipulative visible enough was identified.
- All experiments planned to use manipulatives with similar fine details were abandoned and planning to overcome the size problem was initiated.



- I also identified a need to use a real-world problem in the nested-decider program to make the teaching program code more meaningful and relatable to SIPS. The use of examples ignites excitement, confidence, enthusiasm and plays a critical role in learning programming (Malan & Halland, 2004). Real-world examples help students link context to a programming concept for better understanding (Konecki, Lovrenčić & Kaniški, 2016).

## 5.4 Planning for cycle 2

To overcome the limitations experienced in cycle 1, it was apparent that significant changes were required. The first idea was to convert these physical manipulatives into virtual manipulatives. The advantage of virtual manipulatives is their ability to be demonstrated through computer broadcasts where each student can see what happens on their screens. The plan behind virtual manipulatives was also to overcome the distance between the program code and the actual demonstration. Furthermore, virtual manipulatives were regarded as a compatible alternative to an online presentation which was a mode of presentation across learning institutions during that period of Covid-19. Besides being the teaching and learning mode during Covid-19, teaching and learning solutions and innovations continue to advance towards multimedia-based as this is convenient for both students and educators.

I started converting these manipulatives in the middle of 2021 semester 1 and experimented with them towards the end of the same semester. The advantage was that the semester was extended to end in August due to Covid-19 outbreaks at the experimental institution. I therefore had sufficient time to develop and experiment within that extended semester.

To garner a comprehensive theoretical background on virtual manipulatives, I conducted a thorough literature review on the field of virtual manipulatives. In the literature review, I noted that virtual manipulatives were primarily referred to as program visualisation or animation programs. I likewise began referring to this intervention as animation programs instead of virtual manipulatives. A detailed discussion of animation programs and their limitations was presented in the latter section of the literature review (Chapter 2, section 2.7). The development of these animation programs was both inspired by physical manipulatives and the identified

gap on the design and uses of animation programs in the field of introductory programming.

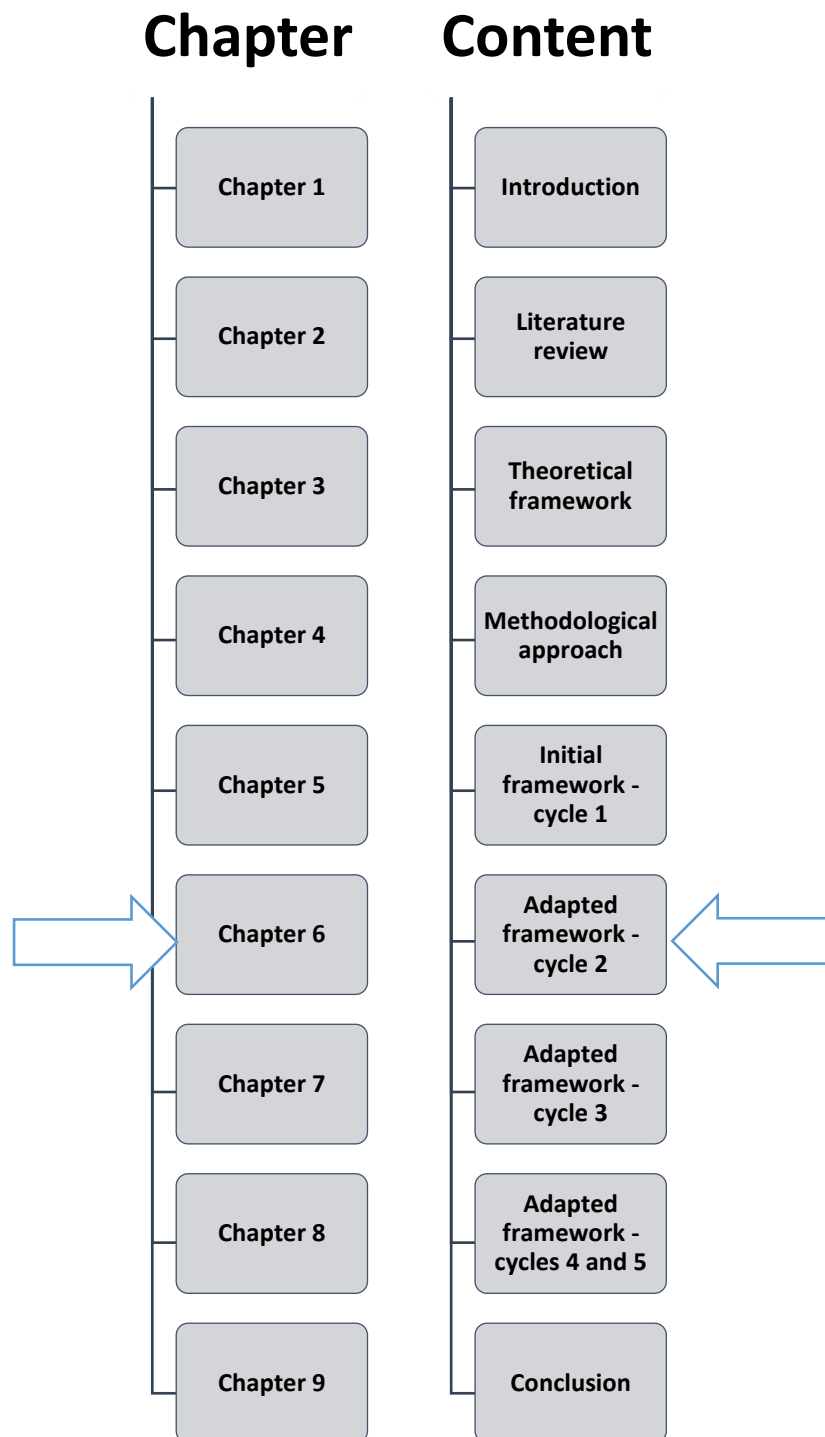
## 5.5 Conclusion

In this cycle, I reported on the use of physical manipulatives to teach introductory programming. Cups were used as a way to emphasise the use of *assignment* in programming and were demonstrated to the actual SIPS at university. It has been noted that the use of cups can indeed improve understanding of how the *assignment* operator works. However, I have noted that the idea of a cups demonstration in a big group is unlikely to be successful.

A manipulative called nested-decider was designed to emphasise *decisions* and *nested decisions* in programming. However, during the experimentation it was found that the nested-decider contained fine details which require students to sit very close to see the demonstration. While this limitation was also identified when using cups to teach the concept of *assignment*, after the demonstration it was clear that everyone was seated closer. Unlike the demonstration with a nested-decider, even after students sat closer, they still struggled to see fine details of the manipulative.

The remainder of the manipulatives that contained fine details were dropped from the experiment. It was necessary to revise the type of intervention used and advance towards a solution that can overcome the stated challenges. The solution in the form of animation programs was realised, planned and experimented. The experimental results of teaching and learning with animation programs are reported in Chapter 6 (Adapted framework – Cycle 2).

# Chapter 6: Adapted framework – cycle 2



## 6.1 Introduction

The aim of this research is to help Struggling Introductory Programming Students (SIPS) by finding ways that can improve their understanding of IPC. To determine the most appropriate and effective teaching and learning methods, I embarked on an action research study to find and conclude on relevant methods. The action research methodology consists of several action cycles. The previous chapter (Chapter 5) presented the activities and the outcomes of cycle 1. The activities in cycle 1 were based on the use of physical manipulatives for teaching and learning IPC. The concept of *assignment* was successfully demonstrated and the findings positive. However, an attempt to demonstrate the concept of *decisions/nested decisions* using manipulatives was unsuccessful and the limitation led to the development of animation programs. One of the projected advantages of animation programs over physical manipulatives includes their compatibility to a big group as visual effects are presented on electronic devices. Electronic devices are important in today's teaching and learning platforms.

Cycle 2 presented in this chapter reports on the animation programs developed and the outcomes of the experimentation with SIPS. The development of the animation programs at this stage was based on the structures of physical manipulatives, reflections in cycle 1 and the animation program gap identified in the literature review.

This cycle serves as a continuation to answer our last research question which is formulated as follows:

- How can we prove and evaluate the efficacy of the Teaching and Learning Programming with Animation Programs (TLPAP) framework?

The framework is now referred to as TLPAP, instead of Teaching and Learning Programming with Manipulatives (TLPM). This chapter begins with an explanation of the designed animation programs for teaching and learning the use of *assignment* (section 6.2), *decisions/nested decisions* (section 6.3) and *loops* (section 6.4). Section 6.5 presents applicable pedagogical guidelines that will be used along the animation programs. Details of experimental results and analysis are given in section 6.6. Section 6.7 contains reflections on this cycle. And section 6.8 presents the planning of cycle 3. Section 6.9 concludes the chapter.

## 6.2 Concept 1 (Assignment)

This section demonstrates the design of animation programs that help SIPS with the concept of *assignment*. The concept of *assignment* is emphasised by focusing on value storage, copying and overwriting.

### 6.2.1 Pre-teaching exercises on the use of assignment

The following questions (in Figure 6.1) were formulated and given to students to answer:

- a. What will be the value of  $x$  after the following algorithm has executed?  
`int x;`  
`x = 3;`  
`x = x + 1;`
- b. What will be the values of  $p$  and  $y$  after the following algorithm is executed?  
`int p, y;`  
`p = 5;`  
`y = 8;`  
`p = y;`
- c. What will be the values of  $k$  and  $t$  after the following algorithm has been executed?  
`int k, t;`  
`k = 1;`  
`t = 2;`  
`k = t;`
- d. What will be the values of  $x$  and  $y$  after the following algorithm has executed?  
`int x, y;`  
`x = 5;`  
`y = 6;`  
  
`x=y;`  
`y=x;`

Figure 6.1: Pre-teaching exercises

The animation program for specification in Figure 6.1 as designed and demonstrated to SIPS as pre-teaching exercises. The following section demonstrates the designed

animation programs and discusses how they are used. The animation programs animate according to the exercises in Figure 6.1 in sequential order.

### 6.2.2 Animation program 1

In Figure 6.2, the algorithm is in a white box and the appropriate animation initiates below the blue line upon button click. In Figure 6.3, a run button has been pressed, the blue highlight in the white box is in line number 1, and the animated container is subsequently generated. The container resembles the variable just like in the physical manipulative (cups).

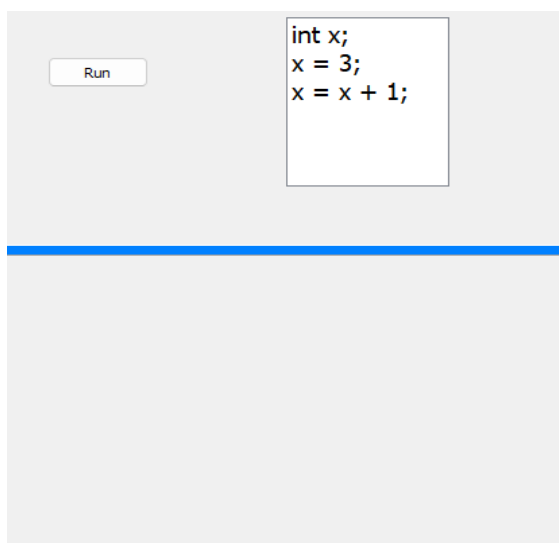


Figure 6.2: Program 1 (Demo 1)

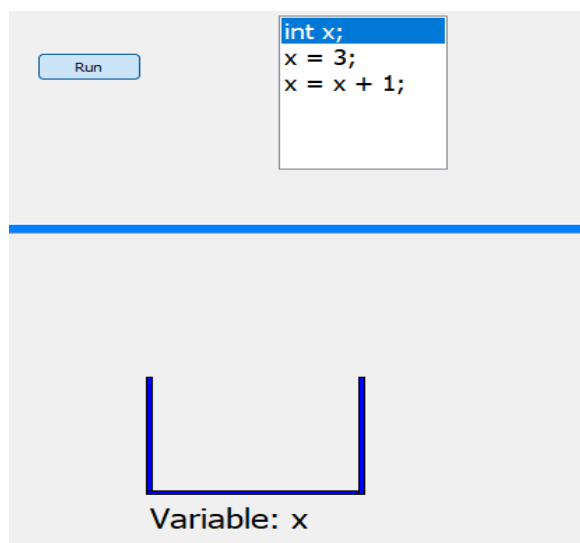


Figure 6.3: Program 1 (Demo 2)

In Figure 6.4, the blue highlight in the program code box moves automatically into line number 2 and value 3 appears in the container to resemble storing of a value. In Figure 6.5,  $x = x + 1$  is demonstrated through a floating number (+1) which indicates that the value of  $x$  is being incremented. The number (+1) appears just below the blue line, and then floats from top to bottom in the direction of the container.

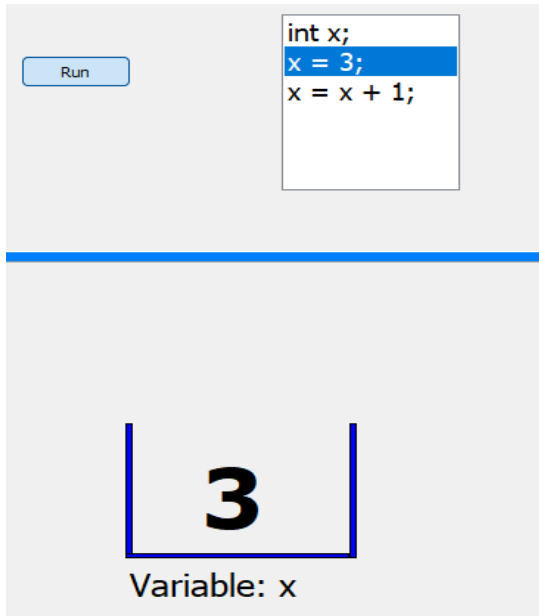


Figure 6.4: Program 1 (Demo 3)

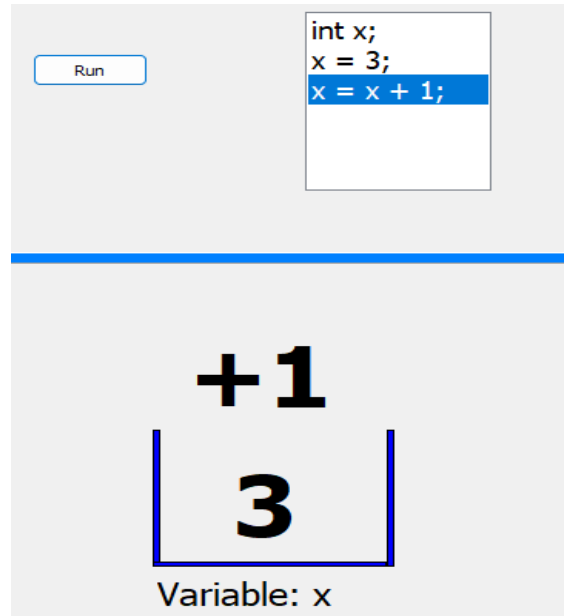


Figure 6.5: Program 1 (Demo 4)

When the floating number (+1) reaches variable x container, the new value replaces the old value (a number 4 replaces a number 3). Figure 6.6 shows a complete executed program and the final state of the animation process.

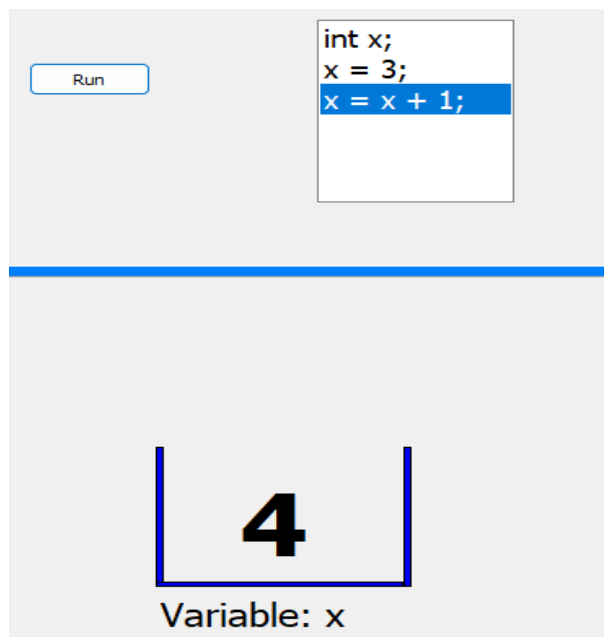


Figure 6.6: Program 2 (Demo 5)

### 6.2.3 Animation program 2

Figure 6.7 contains the program code in the white box. When the blue highlight is at the first line, the two containers appear. The first container represents variable  $p$  and

the second container resembles variable  $y$  (see Figure 6.8). When  $p = 5$  is highlighted, the value 5 appears in cup  $p$  and the same applies when the blue highlight reaches  $y = 8$ .

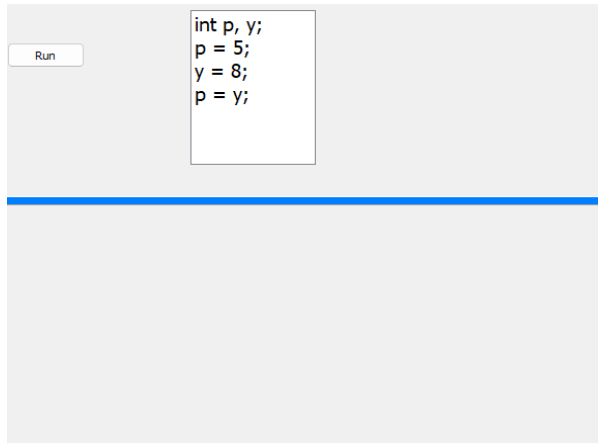


Figure 6.7: Program 2 (Demo 1)

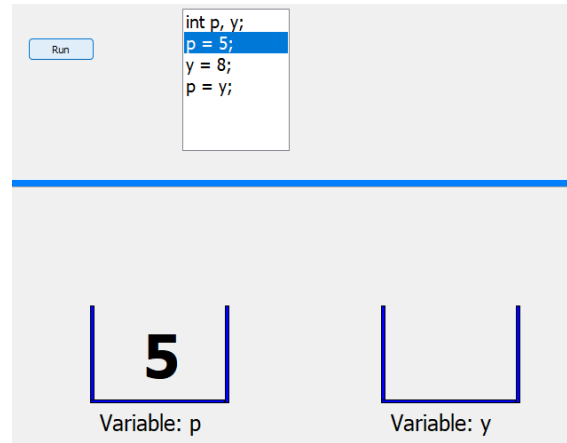


Figure 6.8: Program 2 (Demo 2)

In Figure 6.9, the value is duplicated from variable  $y$ , thereby making an illusion of a copied value. Under the animation section, the text (number 8 as a graphic) moves from 8 towards the variable  $p$  container. The arrow in Figure 6.10 is not part of the animation; it indicates that the value 8 emerged from the container  $y$  and its floating direction.

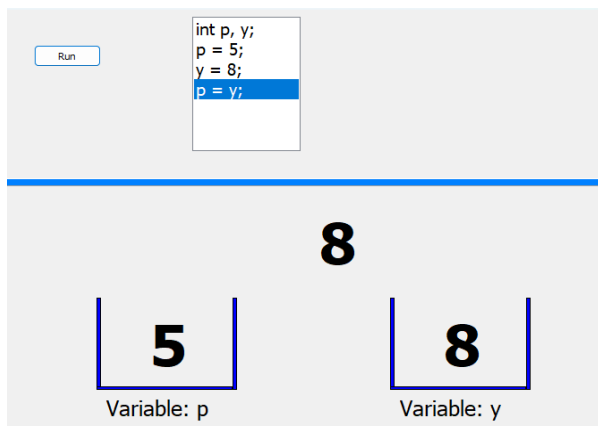


Figure 6.9: Program 2 (Demo 3)

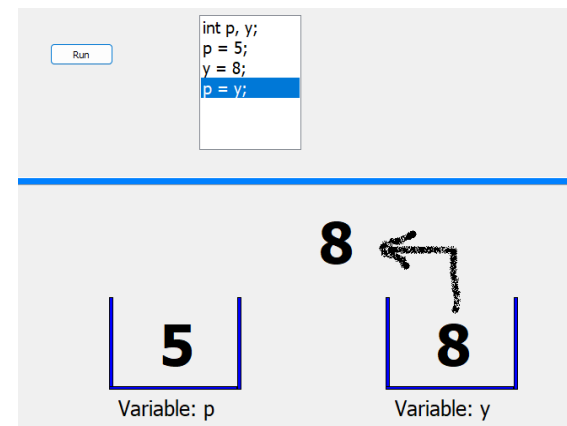


Figure 6.10: Program 2 (Demo 4)

The arrow in Figure 6.11 indicates that the value 8 was copied from variable  $y$  into variable  $p$ . The state of Figure 6.12 is a result of the execution of the last line of the program code and the end of the animation.



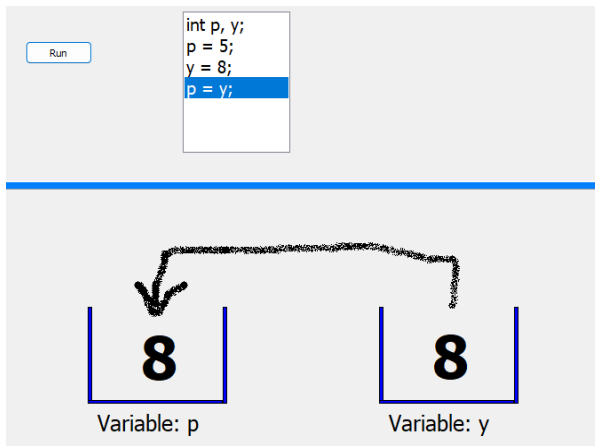


Figure 6.11: Program 2 (Demo 5)

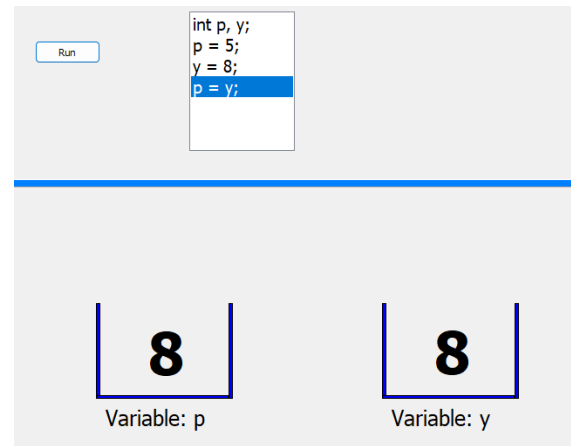


Figure 6.12: Program 2 (Demo 6)

### 6.2.4 Animation program 3

The animation program demonstrated in section 6.2.3 (Animation program 2) and the animation program to be demonstrated in this section are similar, so I have given two figures (Figure 6.13 and Figure 6.14) for sample output. The similarity of the questions was deliberately formulated to give SIPS further understanding of the same concept but different variables, patterns and values. The appearance of Figure 6.13 is the initial state of the animation program and Figure 6.14 indicates the final state of the animation program.

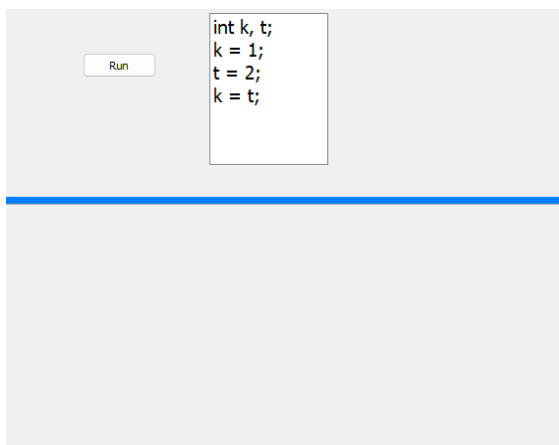


Figure 6.13: Program 3 (Demo 1)

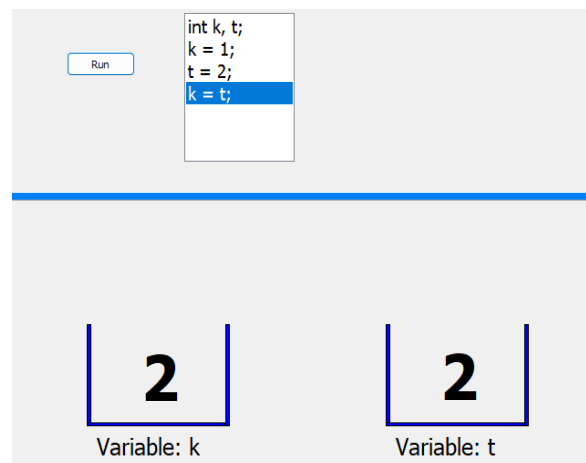


Figure 6.14: Program 3 (Demo 2)

The animation process between Figure 6.12 and Figure 6.13 is the same as demonstrated in section 6.2.3.

### 6.2.5 Animation program 4

The first appearance of the program is depicted in Figure 6.15. The animation progress animates accordingly as was previously indicated as the blue highlight automatically moves line by line. When the program reaches line number 4 (as depicted in Figure 6.16), the value 6 appears inside the variable y container.

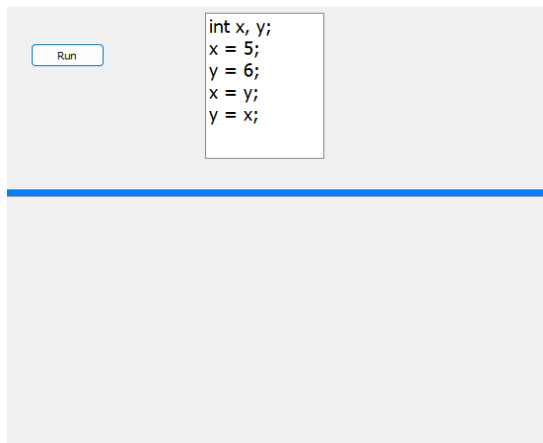


Figure 6.15: Program 4 (Demo 1)

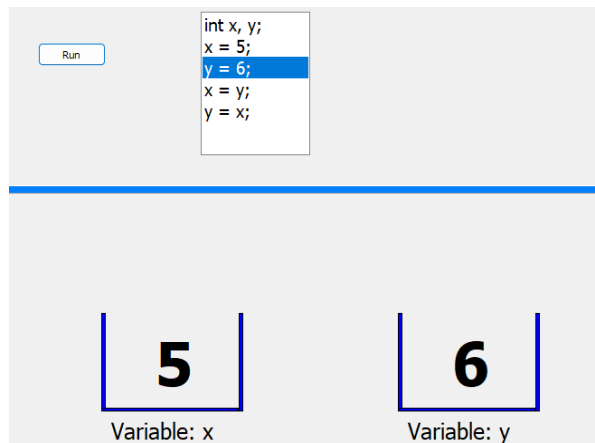


Figure 6.16: Program 4 (Demo 2)

In Figure 6.17, the value 6 animates from variable y to reach and overwrite the content of variable x. The arrow (not part of the animation) in Figure 6.17 demonstrates the floating direction of the number 6. Figure 6.18 is a version without an arrow.

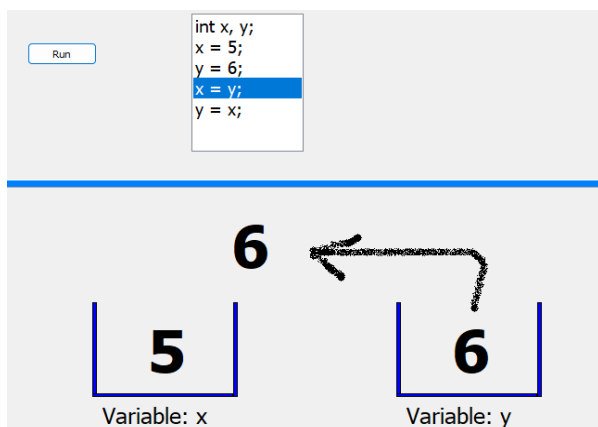


Figure 6.17: Program 4 (Demo 3)

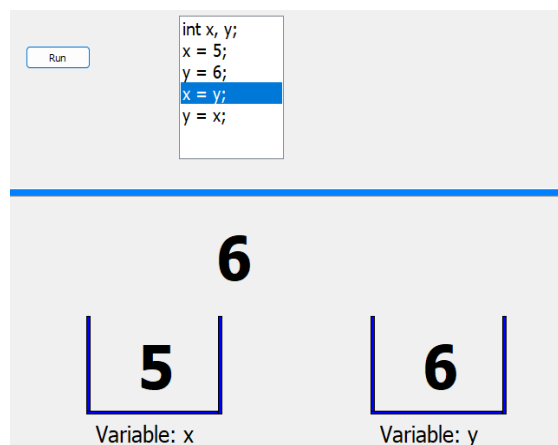


Figure 6.18: Program 4 (Demo 4)

On the last line of the program code, the value of  $x$  floats until it reaches and overwrites the content of variable  $y$  (Figure 6.19). The arrow in Figure 6.19 indicates the floating direction of the number 6. Figure 6.20 is a version without an arrow.

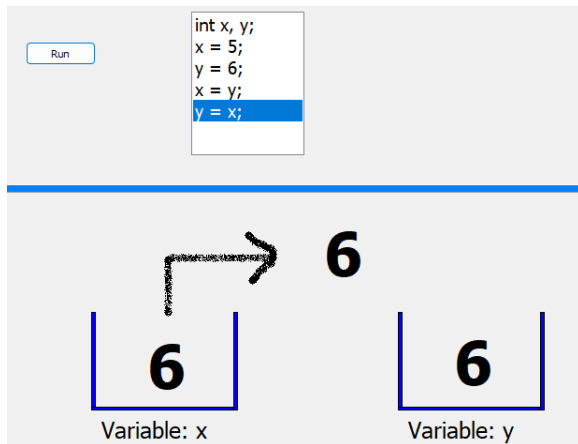


Figure 6.19: Program 4 (Demo 5)

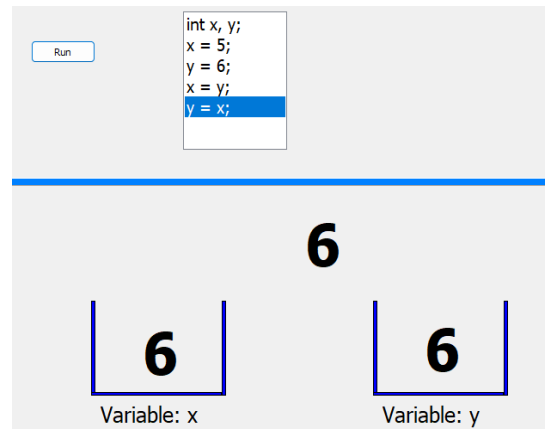


Figure 6.20: Program 4 (Demo 6)

The state of Figure 6.21 is a result of a complete program execution and animation process.

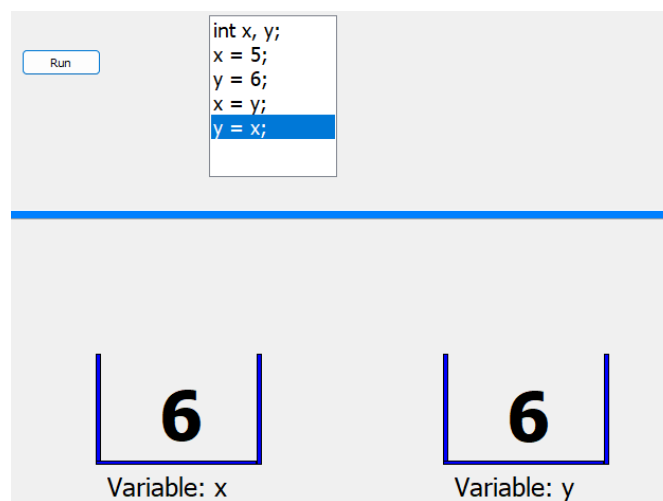


Figure 6.21: Program 4 (Demo 7)

## 6.3 Concept 2 (Decisions/nested decisions)

### 6.3.1 Problem specification for nested-decider

This section demonstrates an animation program called nested-decider. One of the reflection points in cycle 1 was to formulate a real-world problem specification that can

work with nested-decider which can boost SIPS' comprehension on the concept of decisions/nested decisions. The problem specification is described in Figure 6.22.

*You are requested to write an algorithm that shows the process for an employee to get access to the office. An employee must go through a gate authentication, then finally go through office door authentication. At the gate, you need to use a **gate access card** or **Identity document** to get in. If you get access, then in the next stage you need to use both the **office access card** and **office door pin code** for the office door to open.*

*To demonstrate how the authentication from the gate to the office will work, follow the instructions below to complete the algorithm.*

*Prompt the user to enter the letter:*

- *Y or N for a gate access card. (Y means "Yes" and N means "No").*
- *Y or N for an identity document. (Y means "Yes" and N means "No").*

*If the gate access card or identity document contains the value **Y**, then proceed by prompting the user to enter the letter (Y or N), otherwise, show a message "Access into the company not granted".*

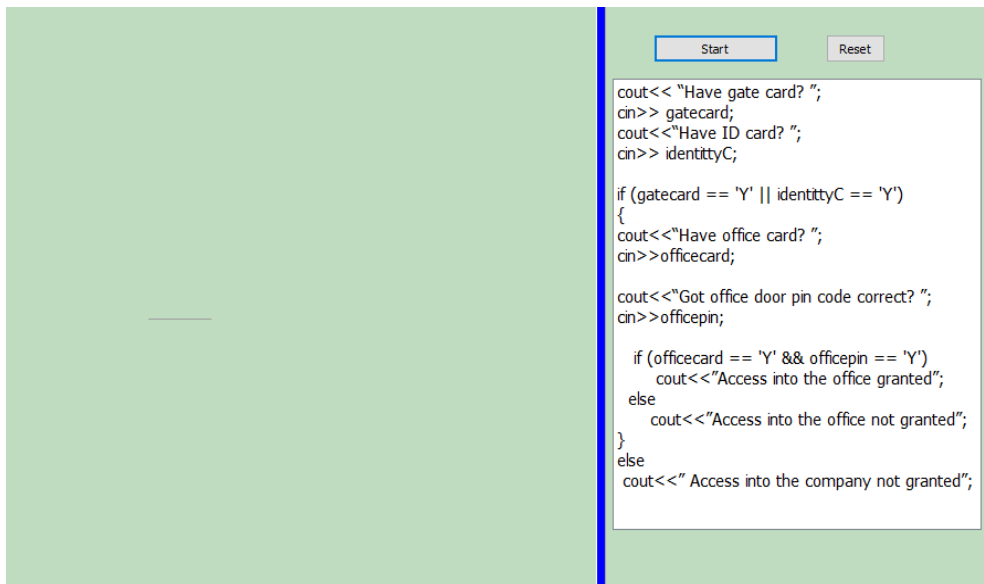
- *Y or N for the office access card. (Y means "Yes" and N means "No").*
- *Y or N for door pin. (Y means "Yes" and N means "No").*

*Check if both office access card and office door pin code contain **Y**, if so, show the message "Access into the office granted", otherwise show the message "Access into the office not granted".*

**Figure 6.22: Problem specification (decisions/nested decisions)**

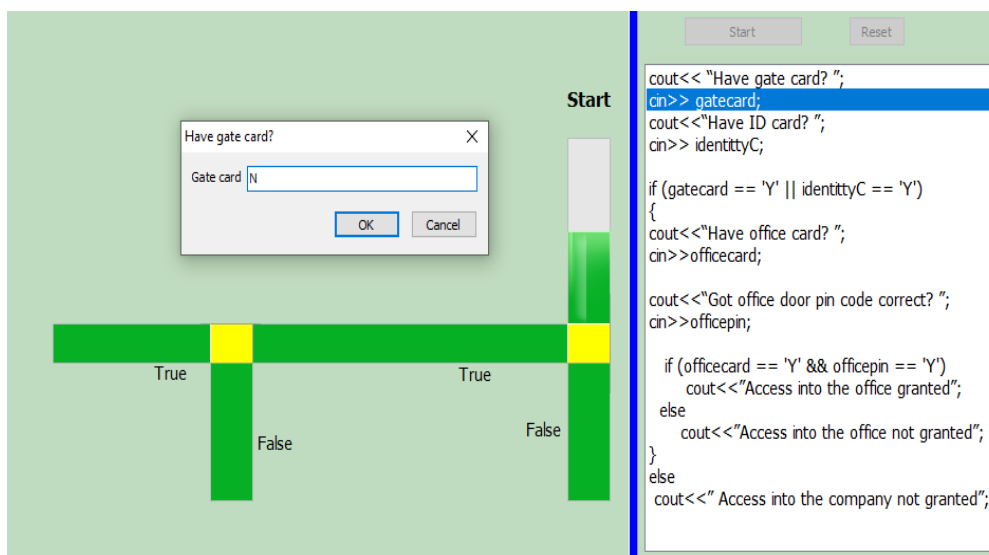
### **6.3.2 Nested-decider animation program**

The program code for problem specification stated in Figure 6.22 is demonstrated in the nested-decider animation program. The nested-decider has two sides: the left side for a program code and the second side reserved for the unfolding of the animation. The code on the right side gets executed after the user presses a start button (see Figure 6.23). The program code and the relevant animation execute synchronously to give SIPS a clearer mental model of a program code and how *decisions/nested decisions* work.



**Figure 6.23: Nested-decider (Demo 1)**

The graphics on the left side start the animation process when the start button is clicked. Figure 6.24 shows the prompting stage of the program (prompts for gate card). The blue highlight on the program code side moves gradually and automatically from one line to another, and the right side animates accordingly based on the currently highlighted line.



**Figure 6.24: Nested-decider (Demo 2)**

In Figure 6.25, the prompt is for the ID card.

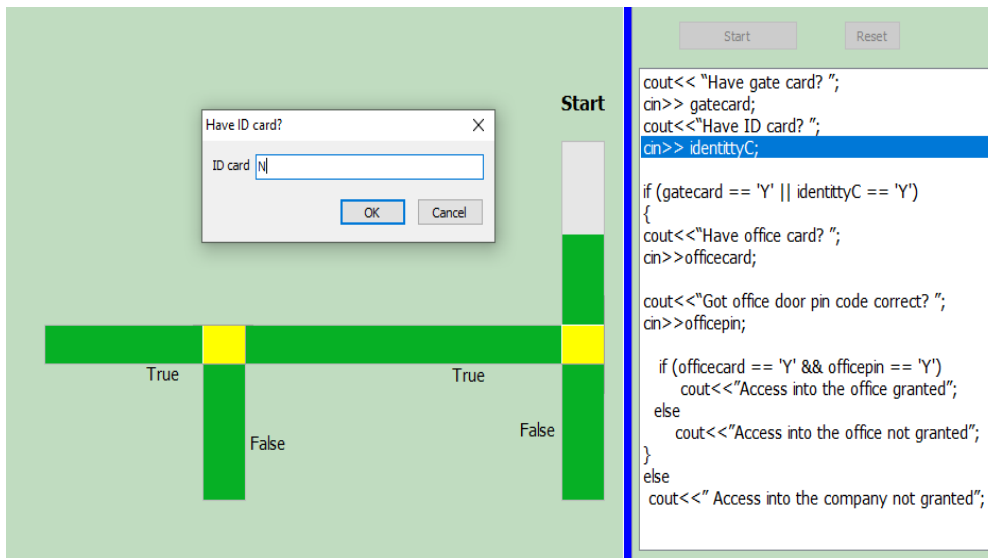


Figure 6.25: Nested-decider (Demo 3)

In Figure 6.26, additional text shows, *Have gate card? = false. Have an ID document? = false. false or false = false*. The appearance of text is important as SIPS may not rely only on the educator's narration but can comprehend independently. The orange square flickers a few seconds and then turns greyish, indicating that the condition has completed the evaluation. The flickering stage, which also serves as a pause, allows the student to read the condition if necessary, then anticipate the direction of the greyish movement before it finally moves. The movement of a greyish bar will either take the bottom direction (false evaluation) or the right direction (true evaluation).

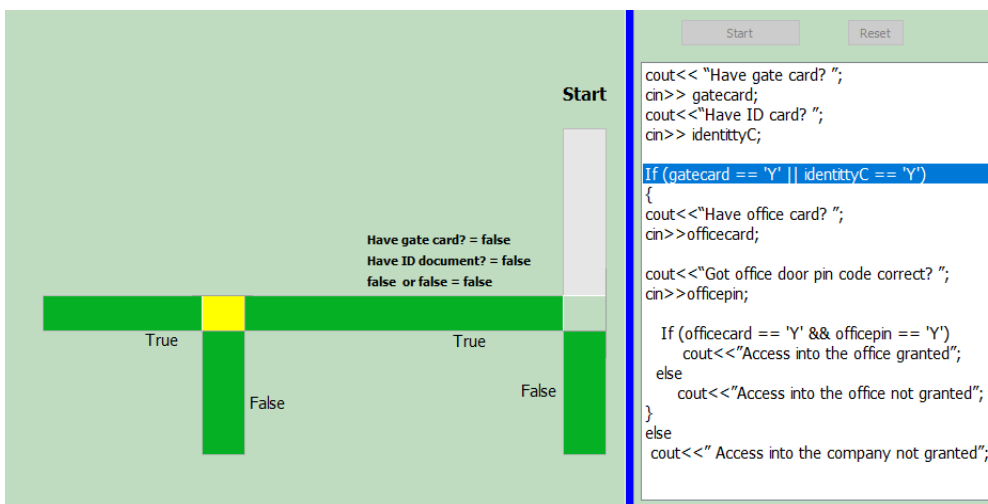


Figure 6.26: Nested-decider (Demo 4)

The state of Figure 6.27 is a result of *false or false* because both values of the gate card and identity card were "N" for No. The critical point SIPS should catch is the

process of understanding the `||` (or short circuit operator) and how a chunk of code is jumped and the last line is executed as a result of an *else* clause. SIPS would have seen that the animating bars on the left remain green which mean that the path was jumped (not executed). Furthermore, when the grey bar completely replaced the green, a message in red font is displayed ("Access into the company not granted") which also corresponds with the currently highlighted line of code.

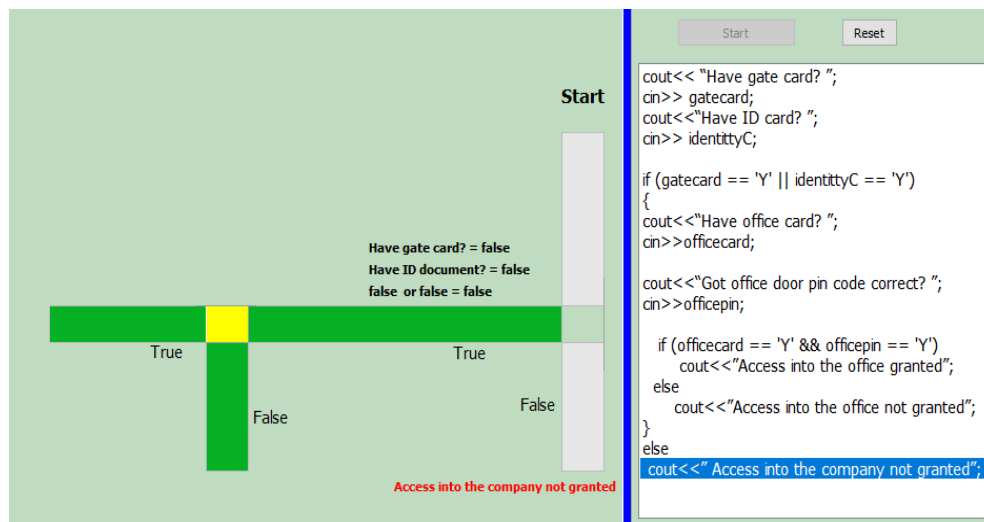


Figure 6.27: Nested-decider (Demo 5)

The state of Figure 6.28 is a result of the gate card and identity document as *false || true* or *true || false* which results in true for any of them. The prompting for the office card and checking if the office pin was correct is done before the greyish animation reaches the yellow box for another condition evaluation.

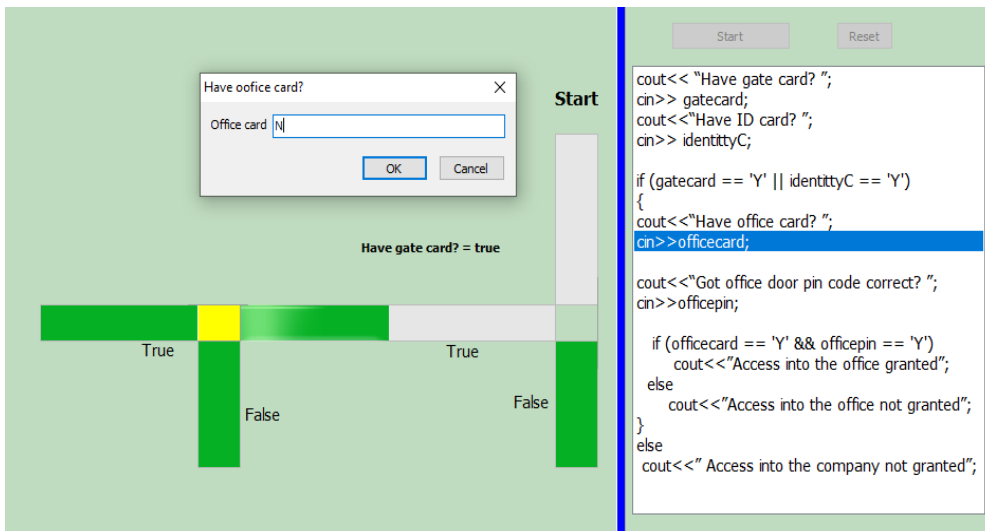


Figure 6.28: Nested-decider (Demo 6)

The state of Figure 6.29 represents a complete executed program where the office card and office pin is *true && false* or *false && true* which gives the results of false. At this stage, SIPS would understand that the true side (green bar) is not greyish.

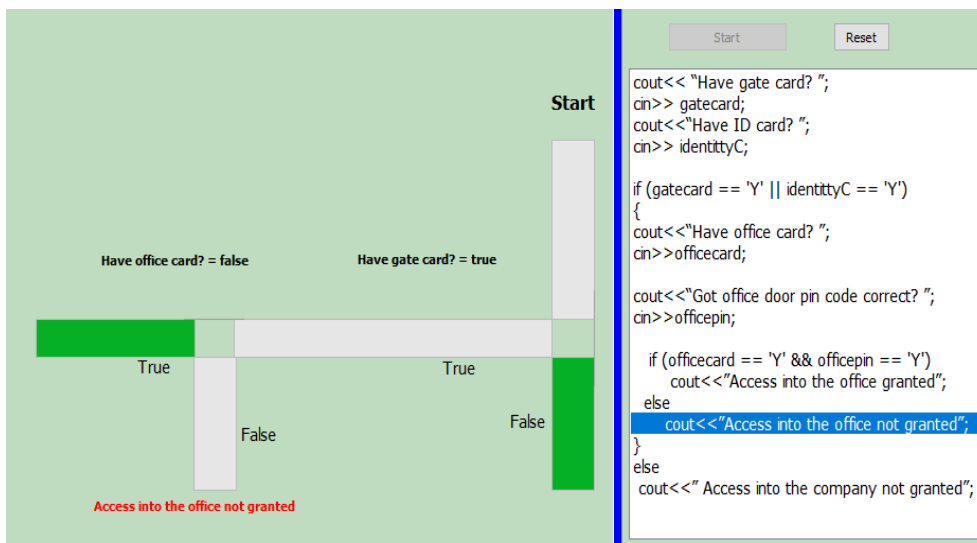
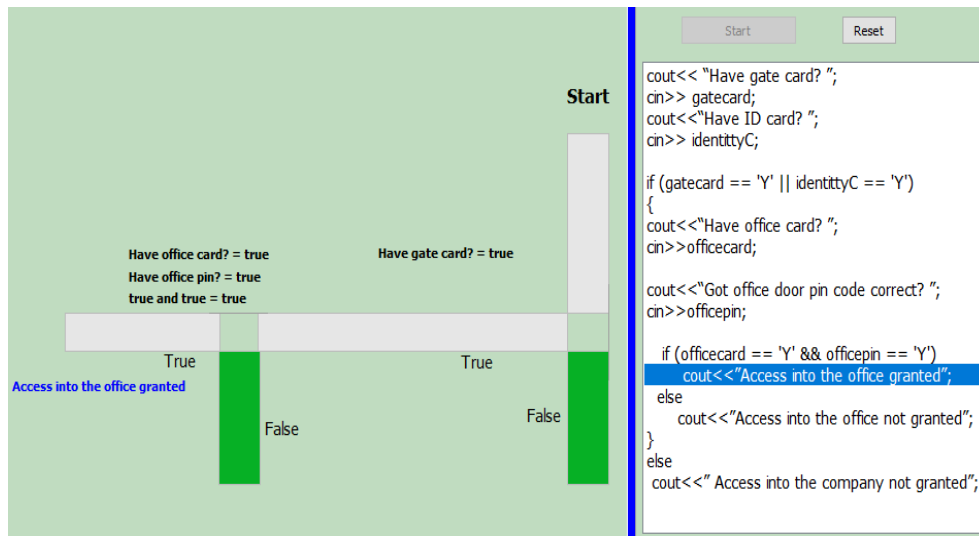


Figure 6.29: Nested-decider (Demo 7)

Finally, Figure 6.30 is a result of an office card and office pin as *true && true* which gives the results of *true*. The corresponding text on the animation is shown as *Have office card? = true. Have office pin? = true. true and true = true*. SIPS would understand that the true direction is executed only when the condition is *true* and *true*.





**Figure 6.30: Nested-decider (Demo 8)**

The Nested-decider animation program was published in a Springer journal<sup>3</sup>.

## 6.4 Concept 3 (Loops/nested loops)

This section demonstrates the animation programs developed for teaching and learning of loops.

### 6.4.1 Problem specification for loops and nested loops

The following figure (Figure 6.31) shows the problem specification designed for loops.

*Write a program that asks a user to enter an integer between 1 and 10. Continue to prompt the user while the value entered does not fall within this range. When the user is successful, display a congratulatory message as many times as the value of the successful number the user entered.*

**Figure 6.31: Problem specifications – loops**

<sup>3</sup> Ramabu, T., Sanders, I., & Schoeman, M. (2022, July). Nested-Decider: An Animation Program for Aiding Teaching and Learning of Decisions/Nested Decisions. In *Annual Conference of the Southern African Computer Lecturers' Association*. Cham: Springer International Publishing, 129 – 148.

## 6.4.2 Animation program for loop

The problem specification in section 6.4.1 (Figure 6.31) has the program code solution and the corresponding animation program as seen in Figure 6.32. The state of Figure 6.32 is before the button is pressed.

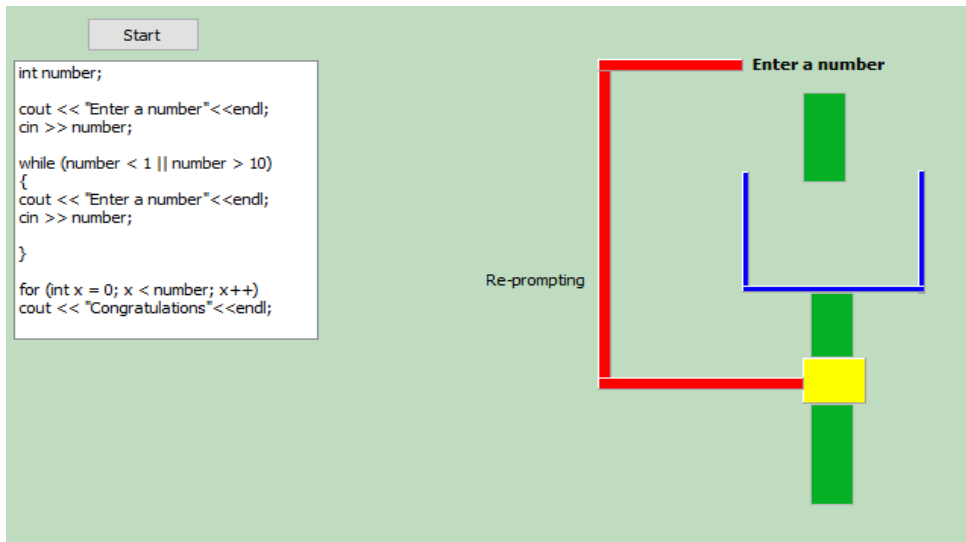


Figure 6.32: Loops (Demo 1)

The prompt to enter a number occurs when the blue highlight reaches `cin>>number` as seen in Figure 6.33.

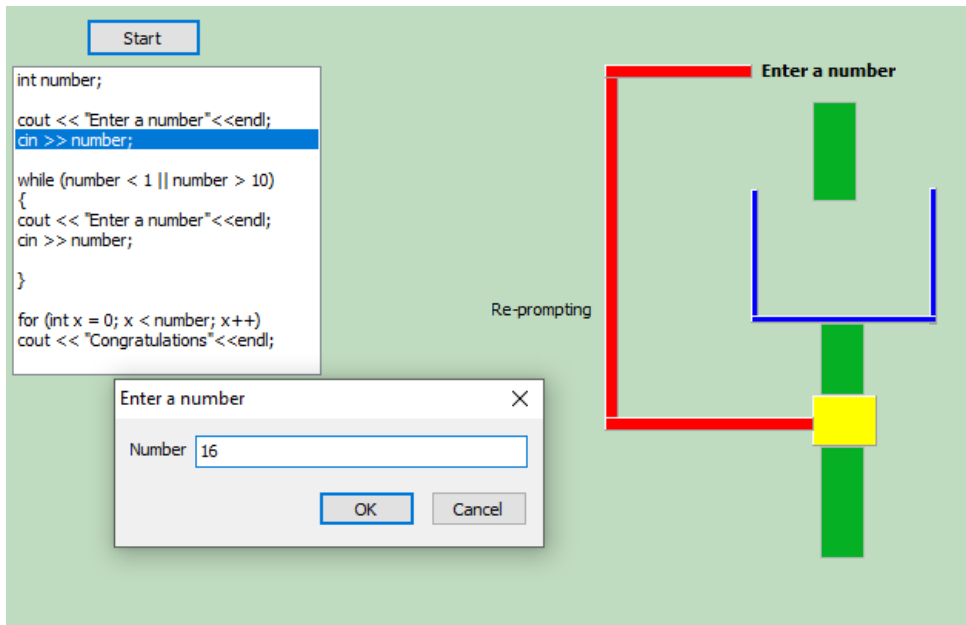


Figure 6.33: Loops (Demo 2)

The first green bar gradually turns greyish from top to bottom after the number has been entered (see Figure 6.34). This resembles a value being stored inside a variable *number* which further reinforces the understanding of *assignment* and storage.

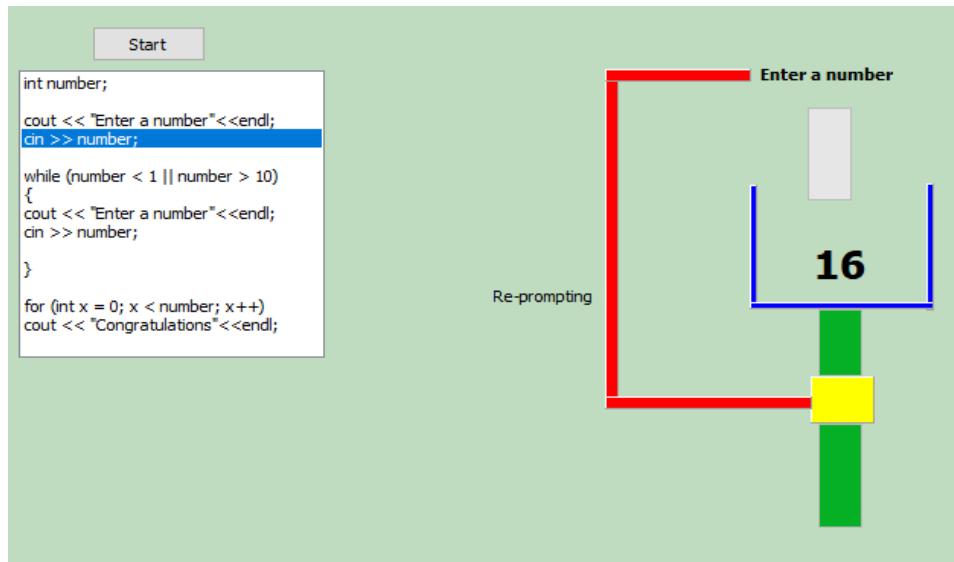


Figure 6.34: Loops (Demo 3)

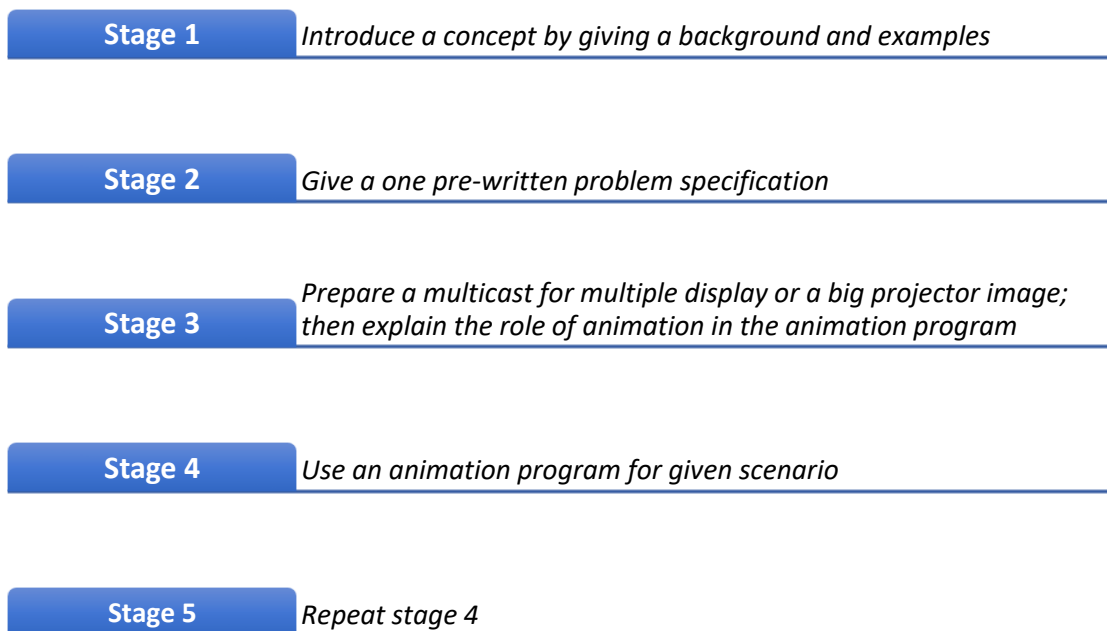
The blue highlight on the program then moves into the line of (*while number < 1 || number > 10*). If a number entered is less than 1 or greater than 10, it means is a true evaluation; then the blue highlight moves inside the loop and re-prompts for a number when *cin>>number* is highlighted (see Figure 6.35). The point in this case is for SIPS to note that a code is repeated by the instruction of the loop condition and what happens when the code is repeated. Once again, when the code is repeated, the concept of overwriting through *assignment* is demonstrated.



congratulations text is printed at every round of the loop. The same experience is realised when the user repeatedly keys in an out-of-range value in the while loop.

## 6.5 Formulated pedagogical guidelines

Animation programs for teaching and learning should not be used in isolation because these programs are only one important piece of the puzzle that must fit into teaching pedagogy. The demonstration of the animation must form part of a bigger picture in the pedagogical sphere. Therefore, these formulated pedagogical guidelines serve as a manual to ensure the effective use and benefits of animation programs for teaching introductory programming with TLPAP. The following guidelines (see Figure 6.37) were followed when demonstrating all animation programs.



**Figure 6.37: Pedagogical guidelines**

The above pedagogical guidelines are substantiated as follows (note that these guidelines will be adapted based on the cycle outcomes):

*Stage 1: Introduce a concept by giving background and examples.*

- Giving thorough background information can help students grasp the overall basic idea about the concept and can also initiate a high-level understanding

before explaining fine details in the form of programming code and animations. This is helpful for avoiding conceptual bugs (Pea, 1986).

- Give an example or metaphor about the concept. The use of examples ignites excitement, confidence, enthusiasm in playing a vital role in learning programming (Malan & Halland, 2004). The use of metaphors or analogies in introductory programming can enhance understanding (Moape, Ojo & van Wyk, 2017).
- Examples should not be too complex or too abstract as this can do more harm than good (Malan & Halland, 2004). According to Malan and Halland (2004), examples like "foo" and "bar" are meaningless to students. Real-world examples help students to link context to a programming concept for better understanding (Konecki, Lovrenčić & Kaniški, 2016). So, in this framework the real-world examples are recommended as they are common and familiar.
- Having given students a background in this stage, that knowledge becomes important for the next stage to link the given example into context. Discussion and interaction should take place throughout as students' reconstruction processes continues.

*Stage 2: Read and explain the problem specification to the students.*

- A program specification refers to the description of what the student should program for solving the problem. The problem specification requires a program code or algorithm as a solution. The problem specification should be in a written or typed format rather than verbal instruction as the students will benefit from reference.
- Do not use difficult terminology when writing a problem specification as these terms could be misinterpreted and subsequently drive students to develop incorrect code. Simplify the language with basic common terminology (Craig, Smith & Petersen, 2017).
- Make sure that students understand this part very well to reduce misconceptions on problem specification, as this is one of a critical barrier to designing a good program code.
- Ask students to identify the inputs and outputs as this will serve as continuous, active learning and engagement throughout the teaching and learning session.

*Stage 3: Run the computer multicast or a big projector image, display the actual first appearance of the animation and then explain the role of animation in the animation program.*

- A multicast can be prepared during a contact session where the controller computer displays the same content on all computers. Alternatively, multicast can be prepared in a standalone platform like Microsoft Teams, Google Meet or Zoom. Another option is a big projector screen visible to all students in the room.
- Explain that the purpose of the animation program is to enhance the programming understanding to ensure that students do not regard the animation program as an independent entity. This will help students to regard the animation programs as a way of learning a program code.

*Stage 4: Use an animation program for a given scenario.*

- At this stage, students should have linked the examples to the introduced concept, grasped the problem specification, noted the inputs and outputs and understand the role of the animation program.
- Let an animation program designed for that specific concept play. The lecturer narrates the programming code in conjunction with the animation unfolding during the process (Végh & Stoffová, 2017).

*Stage 5: Repeat stage 4.*

- The repetition of stage 4 strives to further sharpen program understanding.

## 6.6 Experimentation results

### *6.6.1 Summary of results (Animation programs on assignment)*

The total number of students who participated was 24, with 12 students in the control group and 12 students in the experimental group. This summary is based on the pre-teaching and post-teaching exercises (see Figure 6.38) under the concept of *assignment*.

a. What will be the value of  $t$ ,  $p$  and  $k$  after the following algorithm has executed?

```
int t, p, k;  
t = 5;  
p = 6;  
k = 2;
```

```
t=p;  
p=k;  
k=t;
```

b. What will be the value of  $t$ ,  $p$  and  $k$  after the following algorithm has executed?

```
int t, p, k;  
t = 2;  
p = 4;  
k = 7;
```

```
t=p;  
k=t;  
t=k;
```

Figure 6.38: Post-teaching exercises (Assignment)

Both pre-teaching and post-teaching exercises were graded. Table 6.1 reports on the control and experimental results for both pre-teaching and post-teaching exercises. Note that the pre-teaching exercises are in section 6.2.1. Table 6.1 contains the statistics of the number of students, the percentage of students who answered certain questions correct, and the grade average. For example, in the first column (control group) under pre-teaching exercise 1 there is  $x(8) - 66.6\%$ . In  $x(8) - 66.6\%$ , the number 8 in the brackets means the number of students who got the value of  $x$  correct. The number of students who got the value of  $x$  correct is in the form of a percentage. These calculations apply to all other columns, including the experimental group. The average percentage is recorded at the end of each column.



**Table 6.1: Results (Assignment)**

Pre-teaching exercises		Post-teaching exercises	
Control group	Experimental group	Control group	Experimental group
<b>Pre-teaching exercise 1</b> x (8) - 66.6%	<b>Pre-teaching exercise 1</b> x (9) - 75%	<b>Post-teaching exercise 1</b> t (7) - 58.3%	<b>Post-teaching exercise 1</b> t (9) - 75%
<b>Pre-teaching exercise 2</b> p (8) - 66.6% y (7) - 58.3%	<b>Pre-teaching exercise 2</b> p (8) - 66.6% y (8) - 66.6%	p (6) - 50% k (5) - 41.6%	p (8) - 66.6% k (8) - 66.6%
<b>Pre-teaching exercise 3</b> k (7) - 66.6% t (6) - 50%	<b>Pre-teaching exercise 3</b> k (8) - 66.6% t (7) - 58.3%	<b>Post-teaching exercise 2</b> t (6) - 50%	<b>Post-teaching exercise 2</b> t (10) - 83.3%
<b>Pre-teaching exercise 4</b> x (7) - 58.3% y (7) - 58.3%	<b>Pre-teaching exercise 4</b> x (6) - 58.3% y (4) - 33.3%	p (6) - 50% k (5) - 41.6%	p (9) - 75% k (10) - 83.3%
<b>Average percentage 60.7%</b>	<b>Average percentage 60.6%</b>	<b>Average percentage 48.6%</b>	<b>Average percentage 75%</b>

The control group has an average percentage of 60.7% in the pre-teaching exercise and a grade average of 48.6% in the post-teaching exercise. Surprisingly, the grade average dropped instead of increasing in the post-teaching exercise. This means that the teaching method in the control group did not have a good impact on SIPS. The experimental group had 60.6% in the pre-teaching exercise and 75% in the post-teaching exercise. This increase in the post-teaching exercises is an early indication of the impact of TLPAP on SIPS in the experimental group.

The other notable results is the variation of answers of students who got the value of *k* correct in the first exercise of the post-teaching exercise in both groups. This was caused by the fact that the  $k=t$ ; was supposed to be updated with the overwritten value of *t*, not the initialised value of *t*. The key line of code SIPS were supposed to trace and keep in mind is  $t=p$ ; which changes the value of *t*. However, 6 students from the control group claimed that the value of *k* is 5 instead of 2 in the first pre-teaching exercise. The problem is that the students in the control group did not properly trace the value *k*, as the teaching method was not animation-based and considering that these students are SIPS.

To realise the impact of the adopted teaching methods of both groups, a similar question was given as a post-teaching exercise 2. In the case of post-teaching exercise 2, the tricky part was the value of *t*. What made it tricky was that the value of

$k$  was updated with a value of  $t$ , while the value of  $t$  was also updated with the value of  $k$  and the value of  $t$  was initially updated with the value of  $p$ . In the experimental group more students (83%) got the value of  $t$  correct than in the control group (50%). Generally, the outcome of experimenting with the TLPAP was positive and I will be experimenting again in the next cycle with adapted TLPAP for even more improvement.

### *6.6.2 Summary of misconceptions found in the algorithms for decisions/nested decisions and loops.*

SIPS were able to write both pre-teaching and post-teaching algorithms related to *decisions/nested decisions* and *loops*. The pre-teaching exercise was based on the problem specification stated earlier in Figure 6.21 section 6.3.1. The post-teaching exercise was based on the following problem specification (see Figure 6.39).

*Write a C++ program that determines a final mark, then check if the student has passed or failed. Prompt for a year mark and exam mark. The final mark is the average of the year mark and exam mark. Show a message "You passed" if:*

- *the final mark is 50 or higher and*
- *the exam mark is at least 40.*

*Otherwise, show the message "You failed".*

**Figure 6.39: Post-teaching exercise (decisions/nested decisions)**

A total of 31 students participated in the experiments (15 in the control group and 16 in the experimental group). The test was conducted online through Microsoft Teams. The algorithms were also written and submitted online. Table 6.2 contains the summary of errors found within the algorithms that need *decisions/nested decision* statements. Note that the number in the brackets means the total number of students who committed such error.

**Table 6.2: Summary of errors with the algorithms (decisions/nested decisions)**

Pre-teaching algorithm		Post-teaching algorithm	
Control group	Experimental group	Control group	Experimental group
-Incorrect use of    (3) -Incorrect use of && (3) -Incorrect use of else (0) -Swapped && and    (0) -Not used    (2) -Not used && (2) -Not used else (1) -No if (1) -Disjointed / non-nesting decision (5)	-Incorrect use of    (5) -Incorrect use of && (4) -Incorrect use of else (3) -Swapped && and    (1) -Not used    (2) -Not used && (1) -Not used else (2) -Disjointed / non-nesting decision (6)	-Incorrect use of && (6) - -Incorrect use of else or no else (4) Used    unnecessarily (3)	-Incorrect use of && (3) -Incorrect use of else or no else (3)

The pre-teaching algorithm for loops was based on the problem specification in Figure 6.30 (section 6.4.1). The post-teaching algorithm was based on the following problem specification (see Figure 6.40).

*Write a program that displays the sum of integers from 1 to n value.  
The n value is entered by the user.*

**Figure 6.40: Post-teaching exercise (loops)**

The errors found in the algorithms that needed loops were also summarised. Table 6.3 contains the summary of errors. Note that the number in the brackets means the number of students who committed that error.

**Table 6.3: Summary of errors with the algorithms (loops)**

Pre-teaching algorithm		Post-teaching algorithm	
Control group	Experimental group	Control group	Experimental group
-Incorrect first loop (2) -Incorrect second for loop (6) -Not displayed the right content in the last loop (2) -Incorrectly used nested loop (5)	-Incorrect first loop (2) -Incorrect second for-loop (8) -Not displayed the right content in the last loop (3) -Incorrectly used nested loop (5)	-Incorrect for loop (4) -Incorrect accumulation of sum in the loop (3) -No Display sum outside the loop (3) -Incorrectly used nested loop (4)	-Incorrect for loop (2) -Incorrect accumulation sum in the loop (2) -Not displayed sum outside the loop (2) - Incorrectly used nested loop (3)

### 6.6.3 SOLO-adapted algorithms evaluation - Decisions

The evaluation of pre-teaching and post-teaching algorithms for the concept of decisions / nested decisions are SOLO-adapted as discussed in the methodology chapter (Chapter 4, section 4.5). The teaching and learning method used in the control group is traditional-based infused with verbal or narrative teaching. The second teaching and learning method, for the experimental group, is based entirely on TLPAP. The SOLO-adapted evaluation sheet for the control group is depicted in Table 6.4.

**Table 6.4: Evaluation sheet (control group on decisions/nested decisions)**

		s1	s2	s3	s4	s5	s6	s7	s8	s9	s10	s11	s12	s13	s14	s15
<b>P</b>	Pre										x					
	Post															
<b>U</b>	Pre			x			x					x			x	
	Post										x					
<b>M</b>	Pre		x		x	x		x	x				x	x		x
	Post		x	x	x	x	x	x	x			x		x	x	x
<b>R</b>	Pre	x								x						
	Post	x								x			x			

The evaluation sheet in Table 6.4, s1 means student 1. As there were 15 students in the control group, this is indicated as s1 to s15. In terms of the abbreviations on the first column of the sheet, the P stands for prestructural, U for unistructural, M for multistructural and R for relational. The coordinates form a transition from one stage to another. The SOLO-adapted evaluation sheet for the experimental group is shown in Table 6.5 (based on the concept of decisions / nested decisions).

**Table 6.5: Evaluation sheet (experimental group on decisions/nested decisions)**

		s1	s2	s3	s4	s5	s6	s7	s8	s9	s10	s11	s12	s13	s14	s15	s16
<b>P</b>	Pre				x												
	Post																
<b>U</b>	Pre	x				x		x	x	x	x				x	x	x
	Post																
<b>M</b>	Pre		x	x			x					x	x	x			
	Post	x			x	x			x		x		x		x		x
<b>R</b>	Pre																
	Post		x	x			x	x		x		x		x		x	

The content of Table 6.6 is the evaluation matrix for the control and experimental group. The  $M \rightarrow M$  (*stagnant-at-intermediate-level*) transition has a higher percentage in the control group than the experimental group, which means no impact on the teaching and learning method or simply stagnant at intermediate level. This is the first indicator that the traditional teaching methods only impact to a certain extent, especially on SIPS. The improvements from the lower level ( $P \rightarrow U$ ) were found in one algorithm and the improvements from the higher level ( $M \rightarrow R$ ) were also found in one algorithm. There were two evaluations that transitioned to  $R \rightarrow R$  (*already-know*) which means the teaching methods were not necessary and the success of such transition cannot be linked to a particular teaching intervention. This is because the knowledge was acquired prior to our teaching intervention.

The  $U \rightarrow R$  (*Improved+*) transition had 0 matches in the control group but had 18.7% matches in the experimental group. Even though the  $U \rightarrow R$  recorded a small number, this signals the positive impact of TLPAP over SIPS and is referred to as *improved+*. This is because  $U \rightarrow R$  means that the struggling students were initially far from the relational stage; therefore, it took significant teaching and learning methods to bring them into the relational stage.

The closest the student can reach from the unistructural stage is the multistructural stage which is defined as  $U \rightarrow M$  (*intermediate-improvement*). The performance of  $U \rightarrow M$  (in Figure 6.35) has a marginal difference (26.6% for the control group and 37.5% for the experimental group). This means that both the control and experimental

groups had *intermediate-improvement* status, with the experimental group recording a higher percentage.

**Table 6.6: Evaluation matrix**

P	P→P (Stagnant-no-learning)		U→P (Lower-level-deterioration)		M→P (Intermediate-deterioration+)		R→P (deteriorated++)	
	Con	Exp	Con	Exp	Con	Exp	Con	Exp
	0	0	0	0	0	0	0	0
U	P→U (Lower-level-improvement)		U→U (Stagnant-at-lower-level)		M→U (Intermediate-deterioration)		R→U (deteriorated+)	
	Con	Exp	Con	Exp	Con	Exp	Con	Exp
	6.6%	0	0	0	0	0	0	0
M	P→M (Intermediate-improvement+)		U→M (Intermediate-improvement)		M→M (Stagnant-at-intermediate-level)		R→M (deteriorated)	
	Con	Exp	Con	Exp	Con	Exp	Con	Exp
	0	6.25%	26.7%	37.5%	46.7%	6.3%	0	0
R	P→R (Improved++)		U→R (Improved+)		M→R (Improved)		R→R (Already-improved)	
	Con	Exp	Con	Exp	Con	Exp	Con	Exp
	0	0	0	18.75%	6.6%	31.3%	13.3%	0
	P		U		M		R	

#### 6.6.4 SOLO-adapted algorithms evaluation - Loops

The SOLO-adapted evaluation sheet for the concept of loops for the control group is depicted in Table 6.7.

**Table 6.7: Evaluation sheet (control group on loops)**

		n1	n2	n3	n4	n5	n6	n7	n8	n9	n10	n11	n12	n13	n14
P	Pre				x								x		
	Post												x		
U	Pre		x	x		x	x							x	x
	Post		x		x		x								x
M	Pre	x						x	x	x	x	x			
	Post	x		x		x		x	x		x			x	
R	Pre														
	Post									x		x			

The SOLO-adapted evaluation sheet for the concept of loops for the experimental group is depicted in Table 6.8.

**Table 6.8: Evaluation sheet (experimental group on loops)**

		n1	n2	n3	n4	n5	n6	n7	n8	n9	n10	n11	n12	n13
<b>P</b>	Pre												x	
	Post												x	
<b>U</b>	Pre				x			x		x		x		x
	Post				x									
<b>M</b>	Pre	x	x	x		x	x		x		x			
	Post	x		x			x	x	x	x	x	x		x
<b>R</b>	Pre													
	Post		x			x								

The M→R transition in the control group recorded numbers similar to the experimental group (see Figure 6.9). The M→R indicates improvement; however, it affected very few students in either the control and experimental group. The improvement in the middle level (U→M) affected 30.7% of the experimental students which is a bit more than the U→M impact (21.4%) in the control group. The U→M means *intermediate-improvement*. A small improvement (7.1%) is noticed again under P→U (*lower-level-improvement*) in the control group while the experimental group had 0 algorithms in that transition. The M→M transition (28.6% in the control group and 38.5% in the experimental group) means there were no improvements for struggling students in the middle level. The U→U transition (21.4% in the control group and 7.69% in the experimental group) means there were no improvements for struggling students in the lower level.

**Table 6 9: Evaluation matrix**

P	P→P (Stagnant-no-learning)		U→P (Lower-level-deterioration)		M→P (Intermediate-deterioration+)		R→P (deteriorated++)	
	Con	Exp	Con	Exp	Con	Exp	Con	Exp
	7.1%	0	0	0	0	0	0	0
U	P→U (Lower-level-improvement)		U→U (Stagnant-at-lower-level)		M→U (Intermediate-deterioration)		R→U (deteriorated+)	
	Con	Exp	Con	Exp	Con	Exp	Con	Exp
	7.1%	0	21.4%	7.7%	0	0	0	0
M	P→M (Intermediate-improvement+)		U→M (Intermediate-improvement)		M→M (Stagnant-at-intermediate-level)		R→M (deteriorated)	
	Con	Exp	Con	Exp	Con	Exp	Con	Exp
	0	0	21.4%	30.8%	28.6%	38.5%	0	0
R	P→R (Improved++)		U→R (Improved+)		M→R (Improved)		R→R (Already-improved)	
	Con	Exp	Con	Exp	Con	Exp	Con	Exp
	0	0	0	0	14.3%	15.4%	0	0
	P		U		M		R	

Generally, the performance of the teaching method in the experimental group improved but was not satisfactory, prompting reflection and improvements on both the animations and pedagogical guidelines on loops.

## 6.7 Reflections

Based on this cycle, the reflections are as follows:

- There are persistent errors in the post-teaching algorithms in what is supposed to be basic knowledge. These problems were encountered in the use of *decisions* and even more so in the *loop* conditions.
- Students are not able to identify when to have two or more separate if-statements and when to have a nested if-statement. This was noted in the pre-teaching and post-teaching algorithms of both groups.
- Students are not able to identify when to have two or more separate loops and when to have a nested loop. This was noted in the pre-teaching and post-teaching algorithms of both groups.
- The animation program for the loops requires further design improvements because the results are not good enough compared to the success obtained after using animation programs for teaching and learning *assignment* and



*decisions/nested decisions*. The animation program for loops does not clearly depict two separate loops.

- When animation program for *decisions/nested decisions* was demonstrated, after stage 5 of pedagogical guidelines (which requires a repeat of stage 4), some students requested a repeat of the demonstration again. This was a lesson since the demonstration for the loop animation was repeated three times.
- The need to unify the components in animation program design is identified.
- The animation programs in this cycle were experimented with on Microsoft Teams. Through the Microsoft Teams platform, I was able to note that the animation programs are compatible for online presentation, and I can easily record the session for later access by students. One shortcoming I have experienced was networks disruptions. Another option can be a pre-recorded session which can be placed in the LMS, in this way, students can be directed to a recorded version of the same session in the case of network disruptions, noisy channels or unmuted microphones.

## 6.8 Planning for cycle 3

I am using the reflections of cycle 2 discussed in the previous section (section 6.7) to plan cycle 3. To alleviate the misconceptions about the basics of programming in each concept, the main idea is to develop another set of animation programs specifically for introduction of the new concept. These introductory animation programs are demonstrated before the demonstration of the main animation programs. The introductory animation programs do not need a predefined problem specification as the plan is to give students the conceptual, technical view and basic knowledge before they can solve any problem. Besides identifying the need for animation programs that specialise in concept introduction, a gap has been noted in the literature on the same issue.

Methodologically, the plan is to continue with the same SOLO-based evaluation approach, however in cycle 3 the evaluation will be on various sections of the algorithm, instead of one algorithm as whole. The evaluation of various sections of the algorithm can deepen insight into the analysis of an algorithm. This will help us note the patterns of improvement even in the algorithms which are not 100% correct. The

approach will also help us reflect or plan specifically on a certain section of an algorithm if there are problems.

Having used the animations presented in this cycle in an online platform (MS Teams) and having experienced no incompatibility issues, the demonstration in cycle 3 will be through multicasting in a face-to-face contact session.

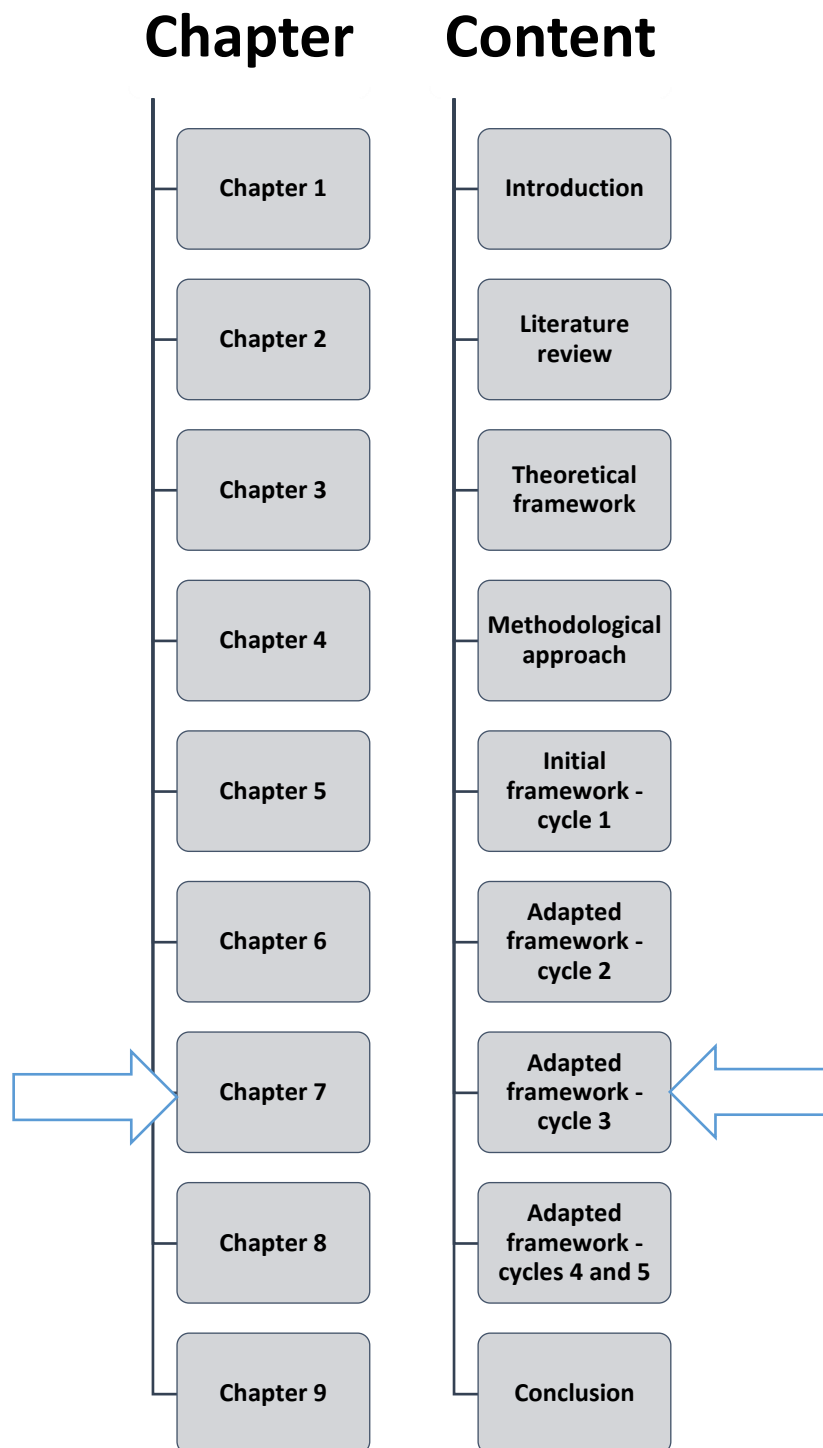
Chapter 7 will focus on the new addition of animation programs in the TLPAP (animation for concept introduction and other new concepts to be covered). The chapter will only repeat the demonstration of the animation programs that are affected by the improvements (animation for loops).

## 6.9 Conclusion

In this study, I have initiated the TLPAP framework for SIPS. This chapter focused on presenting the second cycle of action research where various TLPAP animation programs were outlined and demonstrated. This approach is coupled with guidelines for productive use of animation programs. The process of experimenting, revising and reflecting on the action research methodology allows us to advance toward the final version of the TLPAP framework.

This chapter started with animation programs for the uses of *assignment*, followed by animation programs for the concept of *decisions/nested decisions* and the concept of loops. Through pre-teaching and post-teaching exercises in the control and experimental groups, I measured the impact of the TLPAP framework. The experiments were conducted with university students within an introductory programming module. In the experimental group, the overall improvements were positive in the concept of *assignment* and *decisions/nested decisions*, but more experiments and improvements may be added. The results for the animation for teaching and learning loops were not positive as there was only little improvements or intermediate improvement. As emphasised in the reflection and planning sections (sections 6.7 and 6.8), the plan is to improve the design of the animation for loops and develop separate animation programs for use in the introduction of new concepts. This took place after noting what can be regarded as simple but challenging for SIPS. The full details of cycle 3 are presented in the next chapter (Chapter 7).

# Chapter 7: Adapted framework - Cycle 3



## 7.1 Introduction

As this study is action research, in this chapter I continue to present relevant adapted teaching methods. The action research methodology consists of several action cycles. The previous chapter (Chapter 6) presented the activities and the outcomes of cycle 2 which were based on the animation programs for teaching and learning IPC (*assignment* and *decisions/nested decisions*). One main idea to be demonstrated in this cycle (cycle 3) is the use of a separate set of animation programs specifically for concept introduction. This implies that the animation programs for concept introduction are used first, and thereafter the animation programs with problem specifications. The content of cycle 3 will include animation programs for concepts like *nested loops*, *arrays* and *functions*. The process of developing, testing various animation programs and the adoption of pedagogical guidelines is a way to advance and validate the Teaching and Learning Programming with the Animation Programs (TLPAP) framework.

This cycle serves as a continuation to answer our last research question which is formulated as follows:

- How can we prove and evaluate the efficacy of the teaching and learning programming with the animation programs framework?

The chapter opens by explaining the designed animation programs for introducing a concept (section 7.2), followed by the animation programs with problem specifications (section 7.3), then discussing adapted pedagogical guidelines (section 7.4) and experimental results (section 7.5). Section 7.6 presents the summary of the reflections, with the planning of cycle 4 in section 7.7, while section 7.8 concludes the chapter.

## 7.2 Animation programs for introducing new concepts

This section demonstrates the design of animation programs to be used when introducing a new concept to SIPS. The introductory animation programs are not accompanied by problem specifications as they are short, basic and precise. The students are required to gain the basic knowledge of a certain concept before they can be taught how to solve a problem through a program code. The introductory animation programs focus on the structural aspects of the concept. The design pattern of

introductory animation programs is similar to the animation programs for problem specification which can further reduce animation complexity and ease the transition into the animations for problem specifications. Since the introductory animation programs are short and precise, I found no need to include line indicators (line numbers) on the left side of the program code. The line indicators are included with the animation programs for problem specification because they are likely to be a bit long, so they are helpful in tracing a specific line of code. The uses of these introductory animation programs are incorporated into the pedagogical guidelines used and revised in this cycle (presented in section 7.4). Pedagogical guidelines are followed in efforts to increase the effectiveness of the TLPAP.

### *7.2.1 Decisions/nested decisions*

The introductory animation programs in this section focus on *decisions (if-statements)*. The intended learning outcomes (ILOs) for this concept are for SIPS to construct a proper simple *if-statement*, *if-else statement* and nested *if-statement*. These ILOs are achieved with the introductory animation programs presented in this section and animation programs with problem specification presented in section 7.5.2.

To introduce SIPS to the flow in *decision* statements, the introductory animation programs start with a simple *if-statement*, followed by an *if-else* statement and a *nested if-statement*.

#### *7.2.1.1 Simple if-statement*

The state of Figure 7.1 is the first appearance of the animation program before the button click. On the left of the animation program is a program code and on the right are relevant graphics to animate accordingly. Note that `cout << "Number is less than 10"` line is grey because it is the statement that must execute in the case of true condition; therefore, it catches the attention of SIPS during execution.

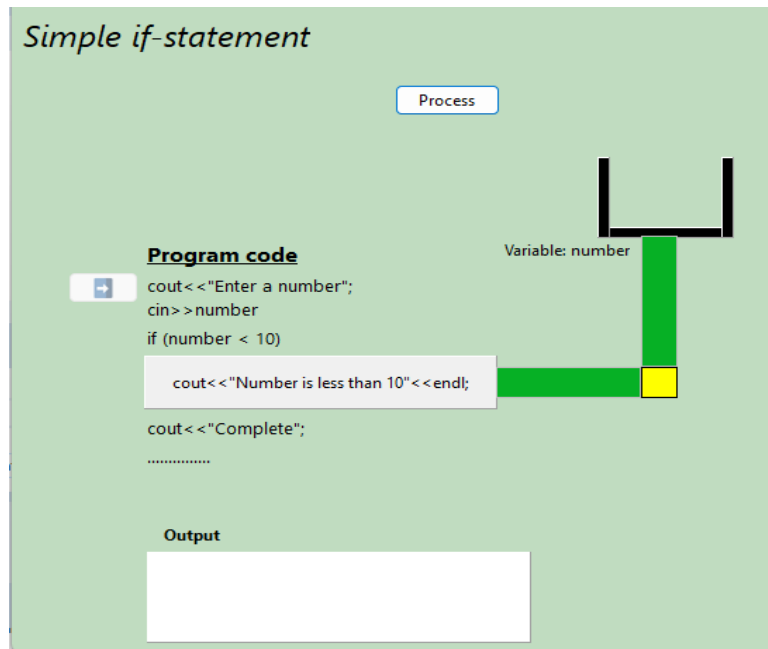


Figure 7.1: Simple if-statement (Demo 1)

An input box pops up after the user presses a *process* button (Figure 7.2). Initially, the arrow on the far left of the program code pointed to first line in Figure 7.1, and then when the arrow progresses to the second line (`cin>>...`) (Figure 7.2), an input box appears.

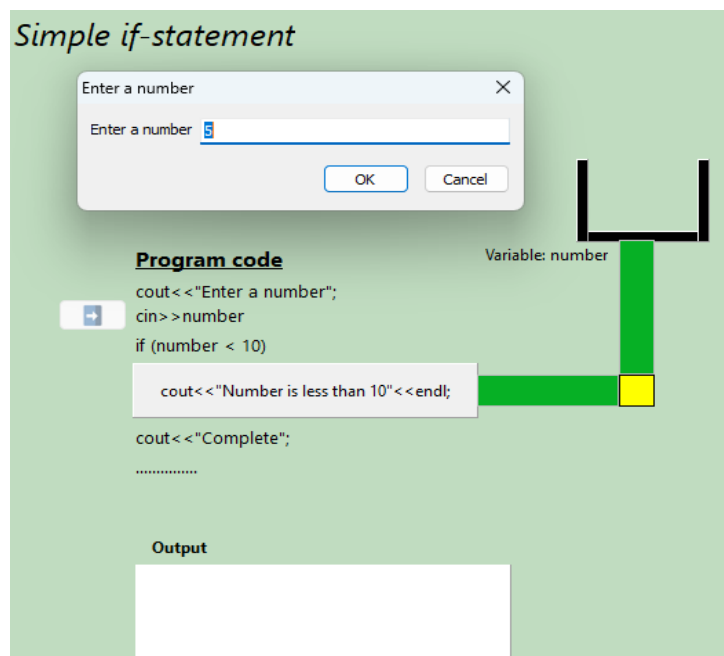
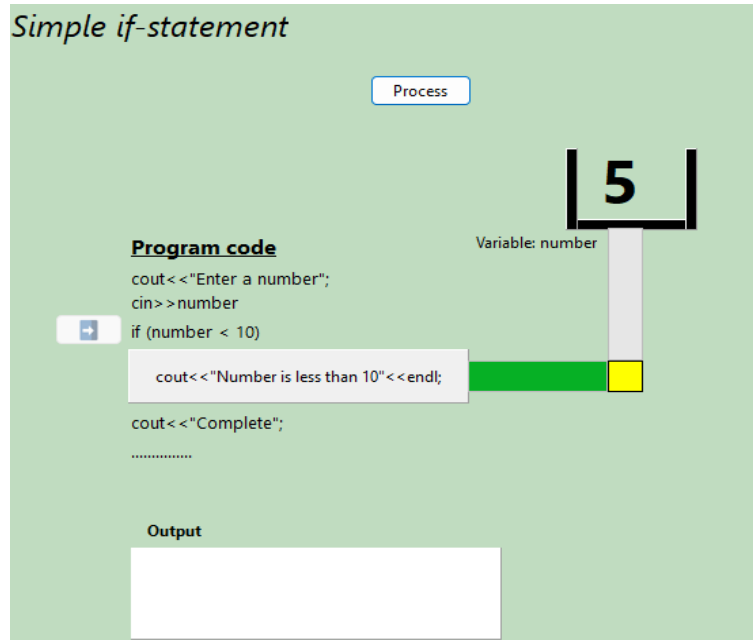


Figure 7 2: Simple if-statement (Demo 2)

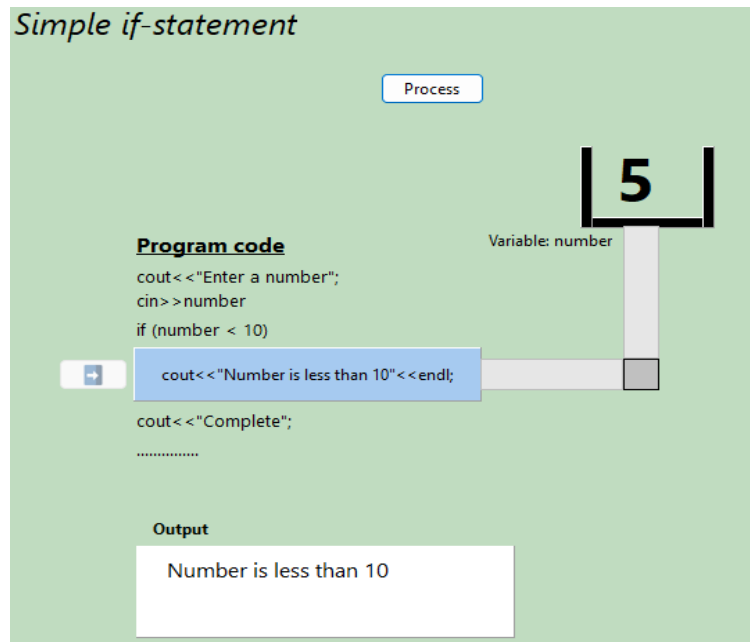
Upon value input, the green vertical bar in the graphic is gradually replaced by the grey bar as a sign of moving to the next line of code. When the move has been

completed and the grey bar has shaded the entire bar, the arrow moves to the next line (see Figure 7.3) which is `if (number < 10)`. While the arrow is being pointed at the if-condition, as shown in Figure 7.3, the yellow square flickers, indicating the process of evaluating a condition.



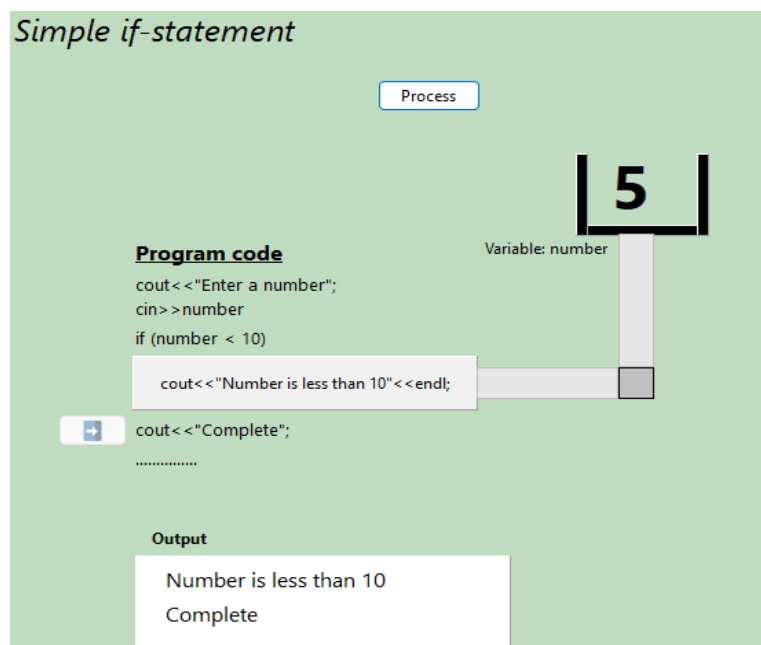
**Figure 7.3: Simple if-statement (Demo 3)**

In Figure 7.4, an arrow is now pointing at the line of code being executed as a result of a true evaluation from the if-condition. That line is shaded with a sky-blue colour to draw SIPS' attention, but it reverts to the original colour when the arrow moves to the next line of code. The output is also printed, as seen at the bottom of Figure 7.4.



**Figure 7.4: Simple if-statement (Demo 4)**

Figure 7.5 shows the arrow pointing at the last line of code; the output is printed at the bottom of the program. At this stage, SIPS should understand that the first output was processed as a result of a true evaluation from the if-condition and the last output gets printed without any conditions.



**Figure 7.5: Simple if-statement (Demo 5)**

To further extend the demonstration and improve understanding, it is important to restart the program and make use of input that will make the if-condition evaluate to



false. In Figure 7.6, the program has been restarted and the value 13 is entered as an input.

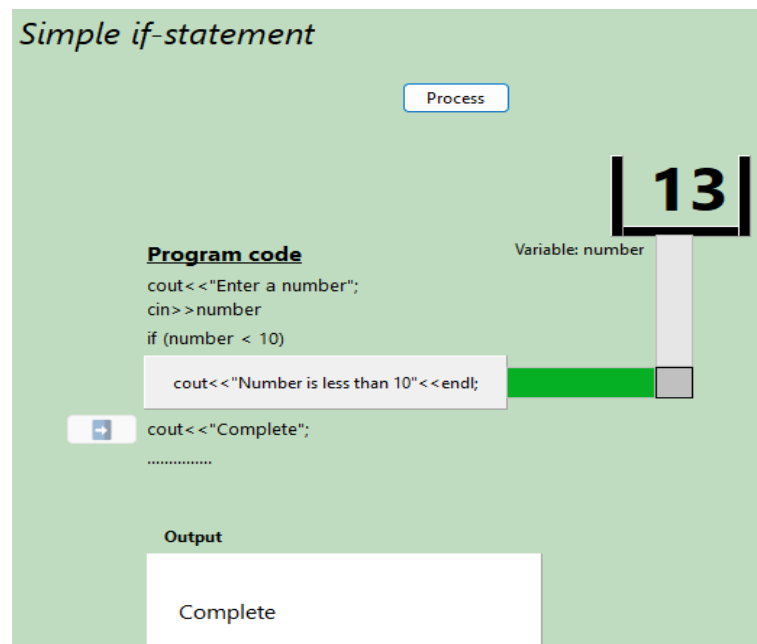


Figure 7.6: Simple if-statement (Demo 6)

The execution of the program is complete and the output is only "Complete". Upon the flickering of the yellow square, the green horizontal bar remains green which means that the condition is false and the relevant line of code is not executed. The arrow will then jump from the if-clause to the `cout<<"Complete"` line of code.

### 7.2.1.2 An if-else statement

The following section explains the program that assists in teaching an *if-else* statement. Figure 7.7 shows the first appearance of the animation program before the *process* button is clicked.

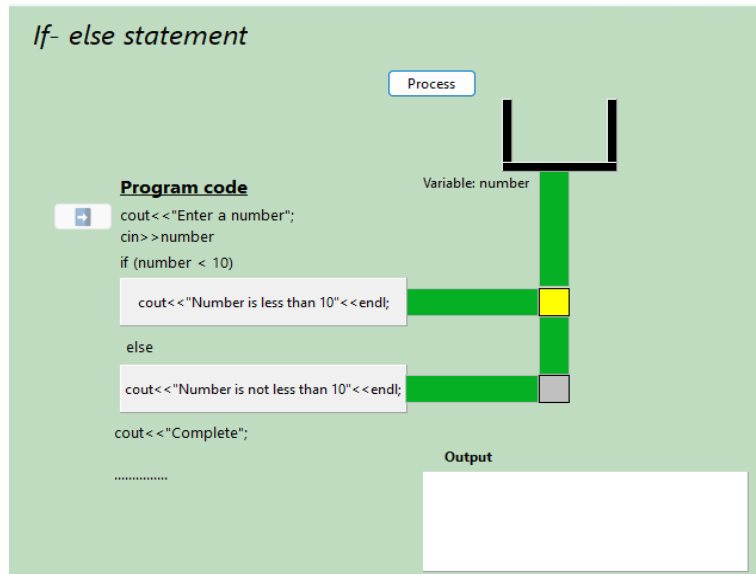


Figure 7.7: An if-else statement (Demo 1)

Figure 7.8 shows a result of a complete animation program when the value 3 is entered. The process of animating and entering a value is the same as explained in the previous section (section 7.2.1.1).

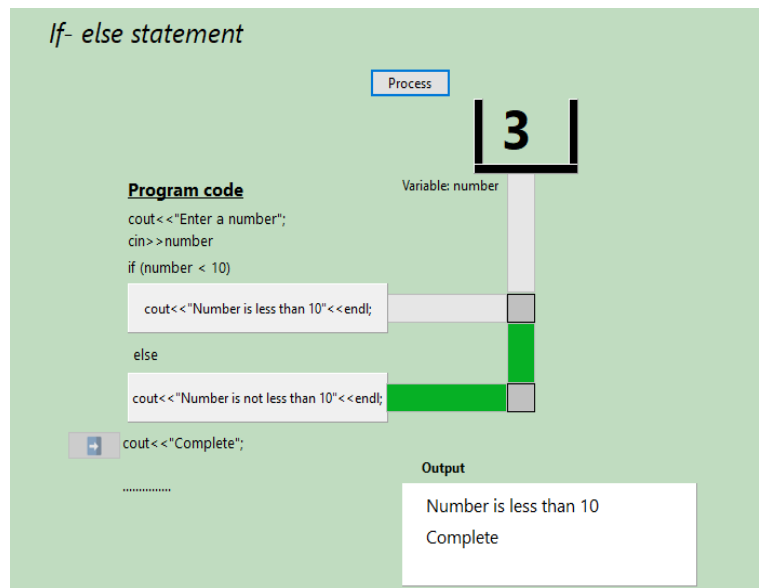
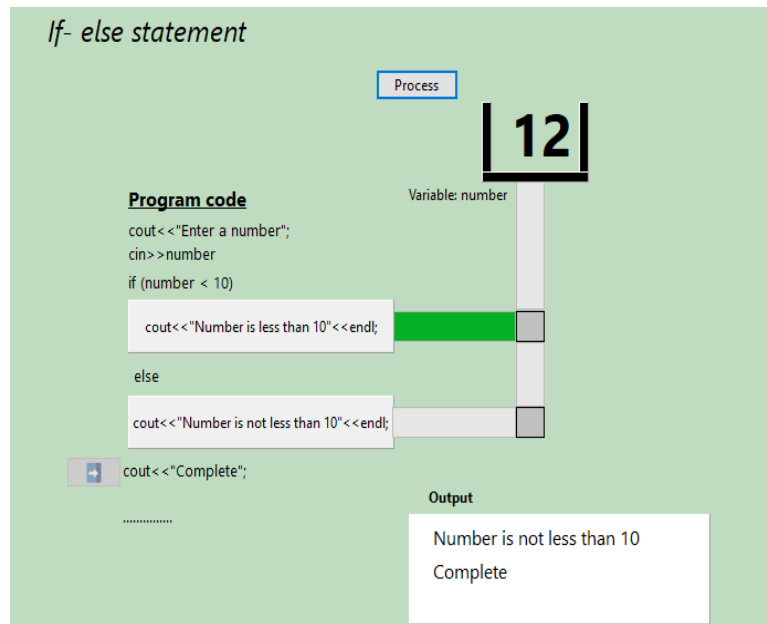


Figure 7.8: An if-else statement (Demo 2)

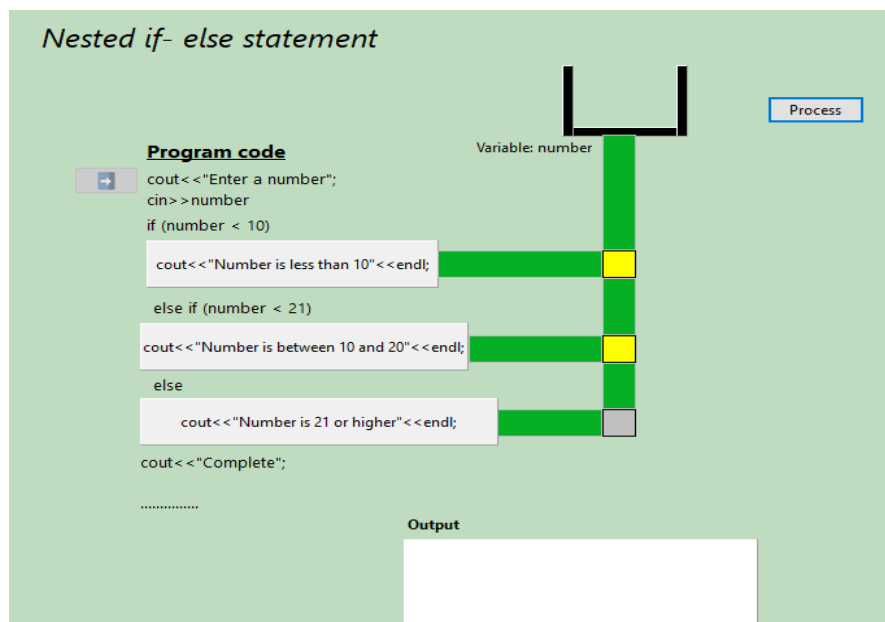
A completed program (see Figure 7.9) shows that an *else* part of the program was executed and SIPS would have seen that the content of the if-condition was jumped due to false evaluation. SIPS would also note the arrow jump from the if-clause to the content of the *else* section, and further note that the bar leading to execute the content of the if-condition remained green.



**Figure 7.9: An if-else statement (Demo 3)**

### 7.2.1.3 Nested-if statement

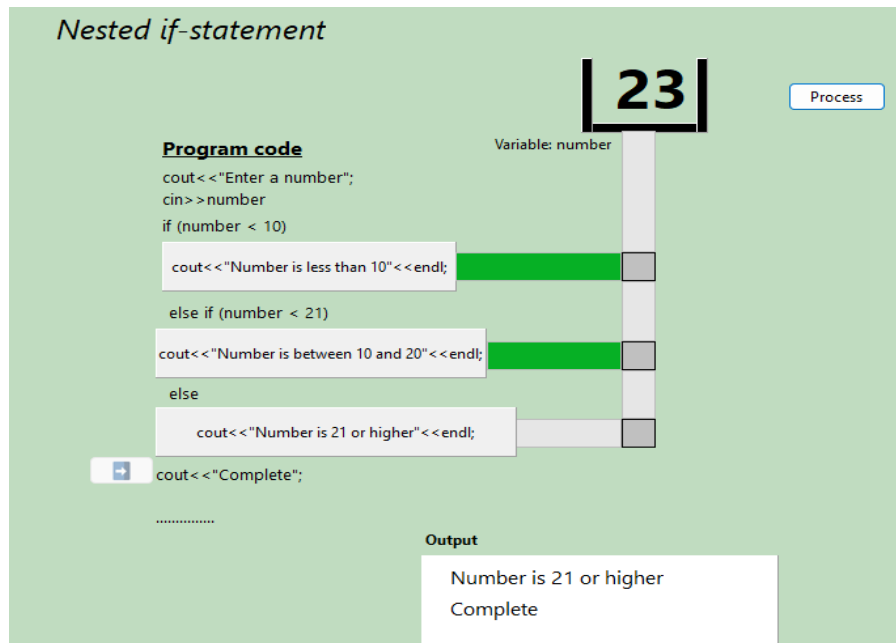
This section demonstrates the nested if-statement. The logic and concept remain the same as the previous two sections, but the demonstration in two figures (Figures 7.10 and 7.11) indicates a significant addition that is part of the nested if-statement. In Figure 7.10, we see another yellow square which represents an additional condition.



**Figure 7.10: Nested-if statement (Demo 1)**

Figure 7.11 shows the program has completed execution. The content of an *else* clause was executed because 23 is not less than 10 and also not less than 21. The

significant learning aspect here is that SIPS would understand that when the condition evaluates to false, the content of that if-condition is jumped (by an arrow) and the horizontal bars would not have changed to grey.



**Figure 7.11: Nested-if statement (Demo 2)**

## 7.2.2 Repetitions

This section demonstrates animation programs that help SIPS with repetitions/loops. The ILO for this concept is for SIPS to construct looping statements which allow a certain portion of a program code to repeat accordingly. The types of loops covered include for-loop, while/do-while loop and *nested loop*. These ILOs are achieved with the introductory animation programs presented in this section and animation programs with problem specification presented in section 7.5.3.

In the animation programs, the important aspects to note for SIPS are the movement of an arrow and what happens to the repeated code. The following section starts with the for-loop.

### 7.2.2.1 For-loop

A sample for-loop code is given in Figure 7.12. The code within a loop gets repeated three times.

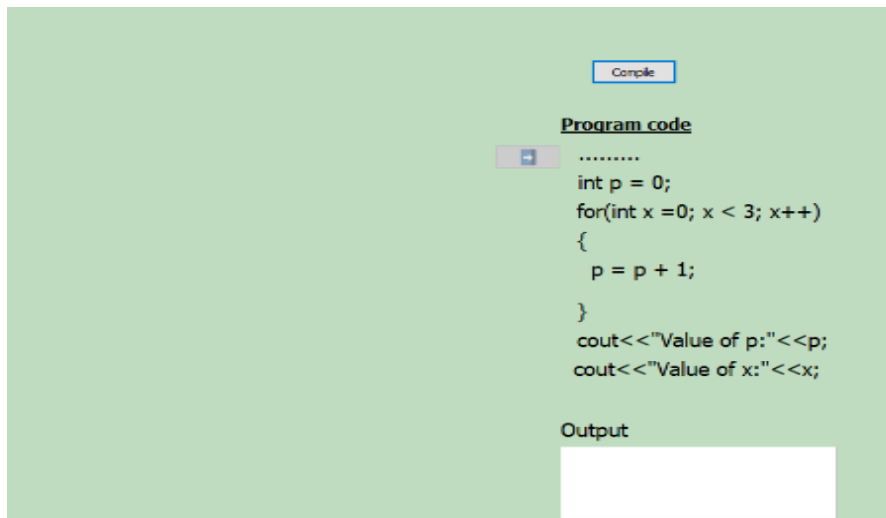


Figure 7.12: For-loop (Demo 1)

When the arrow moves to the first line, a container that represents a variable appears. When the arrow moves to the for-clause, corresponding blue text appears to indicate the current value of  $x$  and whether  $x < 3$ ; is true or false. When the arrow moves to the next line ( $p = p + 1$ ), the green bar gradually turns grey, then executes  $p = p + 1$  (see Figure 7.13).

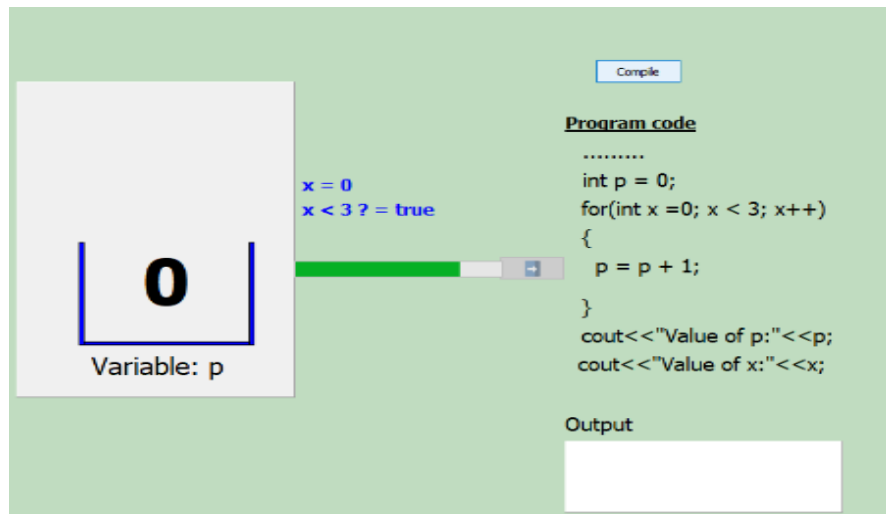


Figure 7.13: For-loop (Demo 2)

In Figure 7.14, the number  $+1$  appears and then gradually moves down until it reaches the location of the value  $0$ . When  $+1$  reaches where  $0$  is, it then changes the value to  $1$ . The arrow moves back to the for-clause, and then the blue text ( $x = 1$  and  $x < 3 = true$ ) is updated accordingly (Figure 7.14).

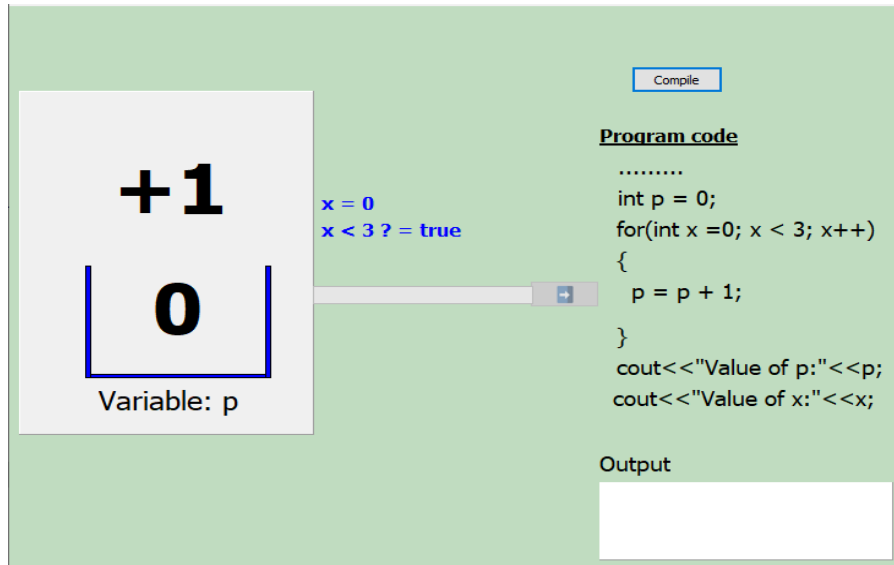


Figure 7.14: For-loop (Demo 3)

In Figure 7.15, the +1 graphic moves again to increment the value of p. When the animating +1 text reaches where the value is, it changes to 2. The arrow will move back to the for-clause again, change x to 2 and x < 3 remains true because the value of x is 2.

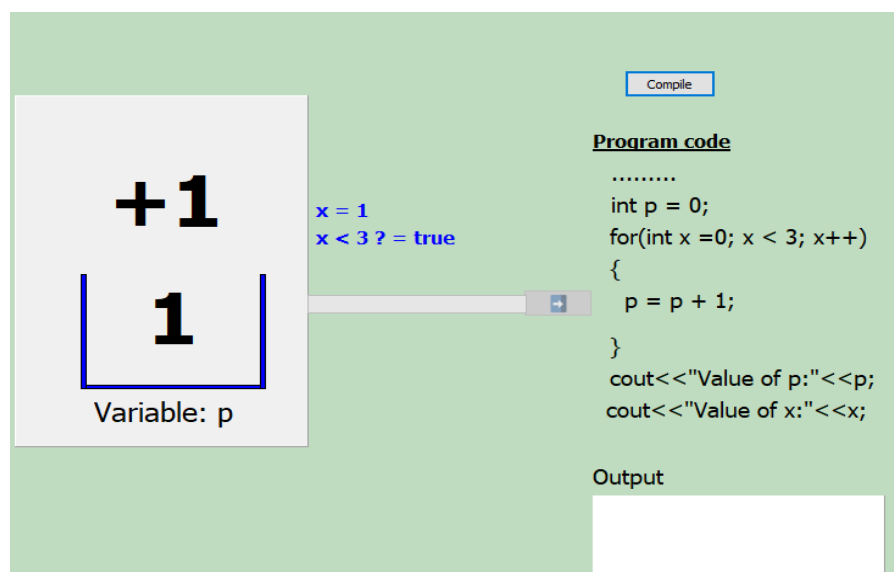


Figure 7.15: For-loop (Demo 4)

In Figure 7.16, the +1 graphic moves to where the value 2 is, then it changes to 3. This indicates the third round of the loop.

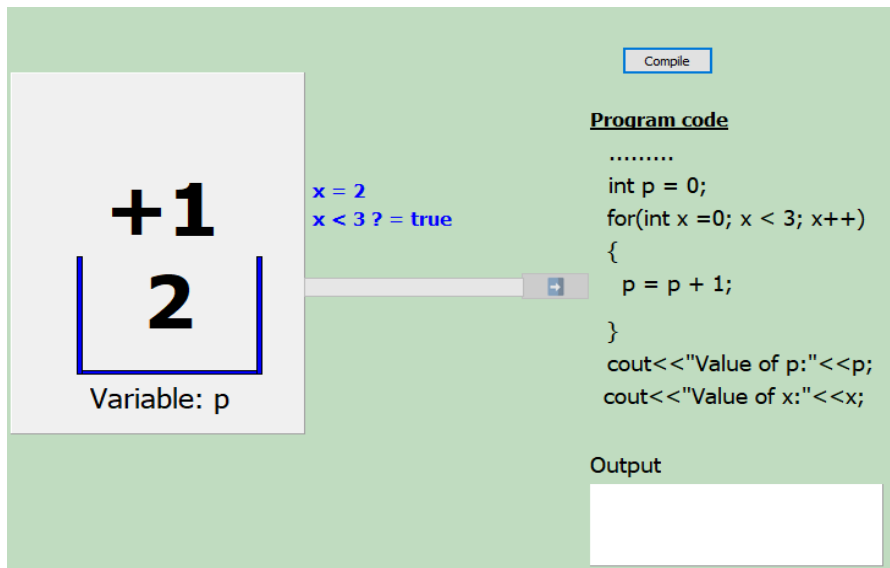


Figure 7.16: For-loop (Demo 5)

On the last round of the loop, an arrow moves back to the for-clause and updates the text accordingly. In this case it still updates  $x = 3$  with blue font and  $x < 3?$  *false* with red font to further draw SIPS' attention. The arrow will not go back to point at  $p = p + 1$  but jump to the first `cout<<` and print the output, then move to the last `cout<<` and print the last output (Figure 7.17). The animation authoring has only been created to accommodate  $p = p + 1$ . Therefore, the emphasis with this demonstration should also be about the similarity of  $x++$  which is part of a loop counter and  $p = p + 1$  which increment 1 inside the loop.

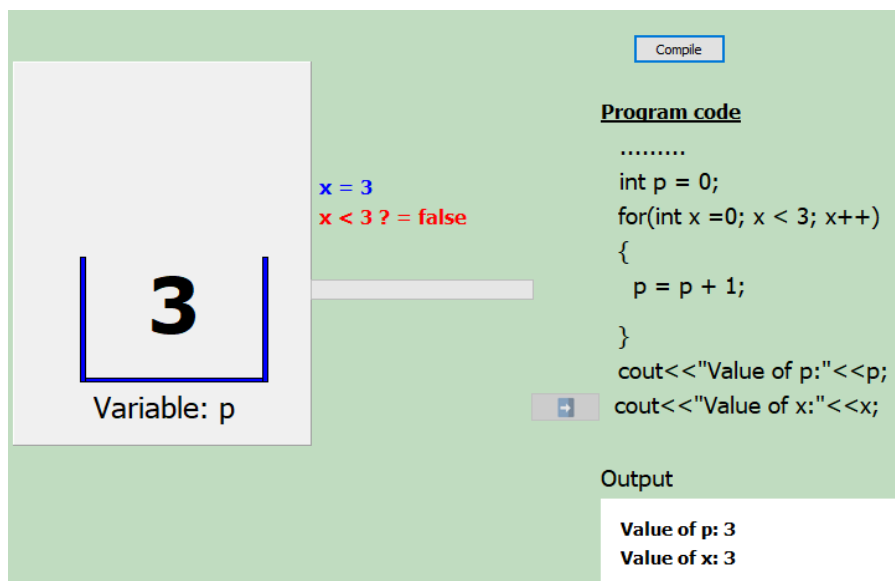


Figure 7.17: For-loop (Demo 6)

### 7.2.2.2 While-loop and Do-while loop

The following animation program (Figures 7.18 and 7.19) does exactly what the previous section did but under a while-loop. Therefore, to avoid repetition, I only show an executed program in the figures. By Figure 7.18, SIPS should be able to contrast differences between for-loop and while-loop which are both types of pre-test loops. This must be emphasised by the educator.

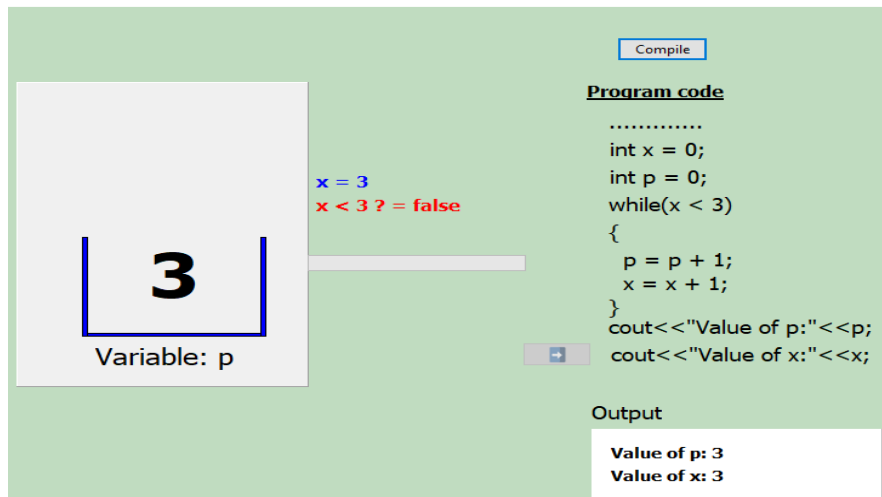


Figure 7.18: While-loop (Demo 1)

By Figure 7.19, SIPS should further comprehend the difference between pre-test and post-test loops, since a do-while is a type of post-test loop (the condition is evaluated at the end of the first round of the loop). This should also be emphasised by the educator as the animation shows how a condition is tested first in the case of a pre-test loop and later in the case of a post-test loop.

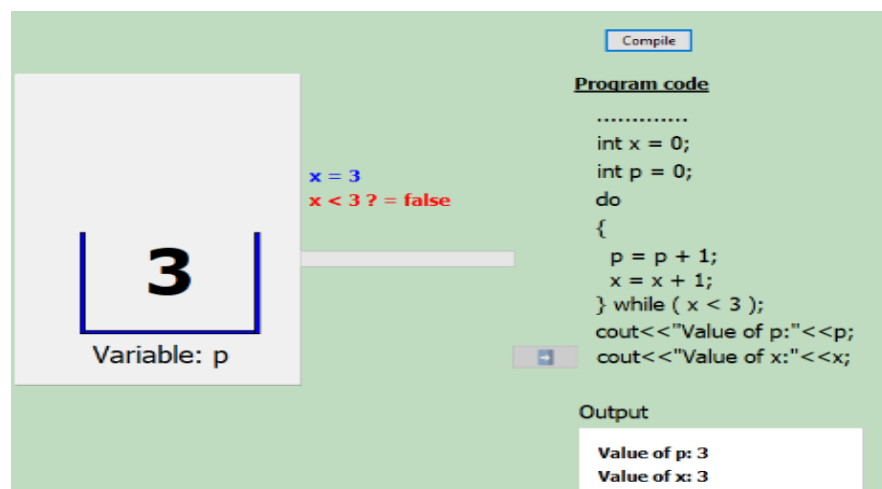


Figure 7.19: Do while-loop (Demo 2)



### 7.2.2.3 Nested loop

In Figure 7.20, the introductory animation program for *nested loop* consists of two rectangular shapes. This is designed in a way to encourage SIPS' understanding of the nature of a *nested loop* and how it happens. The right-side rectangular shape denotes that the code within that shape will be repeated (looped) if necessary. The left side of the shape is reserved for relevant animation and the second half of the shape is the program code. The track bars at the top of each square, part of the animation, are used to trace the state of the loop during a looping process. Likewise, this should reinforce SIPS' understanding of the execution of the inner loop in relation to the outer loop. The nature of the graphics and what they will do are explained by the educator beforehand, as noted in the TLPAP pedagogical guidelines. This introductory animation program differs from other loop animation because the graphics clearly separate the inner loop and outer loop, demonstrating the activities and effects of each loop over the other.

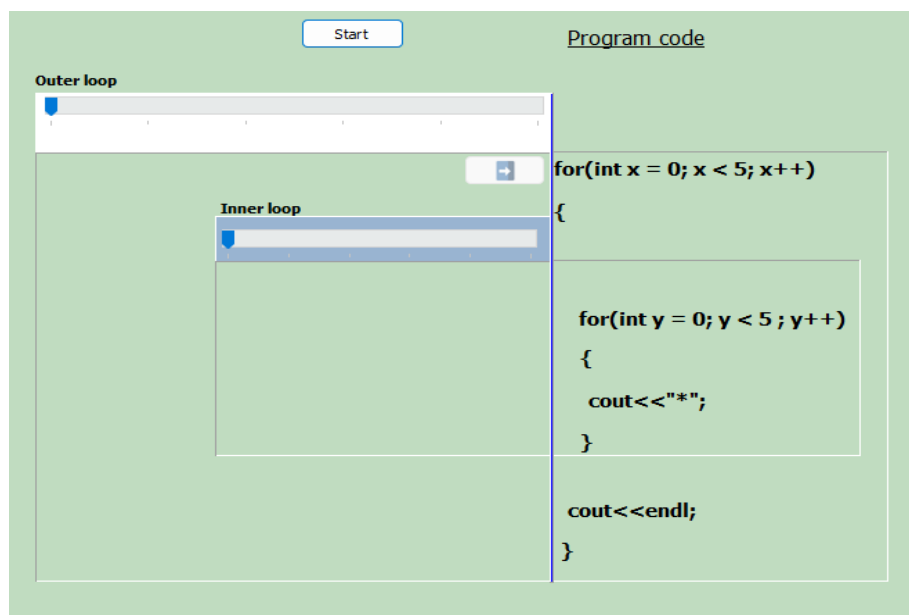


Figure 7.20: Nested-loop (Demo 1)

The trackbars have small spaces between the beginning and the end. For each round of a loop, the blue arrow (pointing down) on the track bar moves step by step (it will have to move five steps before it can reach the end of the bar). Both inner and outer loops are to loop five times, and hence the track bar steps five times. The arrow pointing at the line of a program code indicates the currently executing line of code. The stars (\*), printed gradually during code execution, are printed within the inner

rectangle because that happens through the inner loop. Figure 7.21 reveals the animation program execution which currently is in the third round of the outer loop and fourth round of the inner loop.

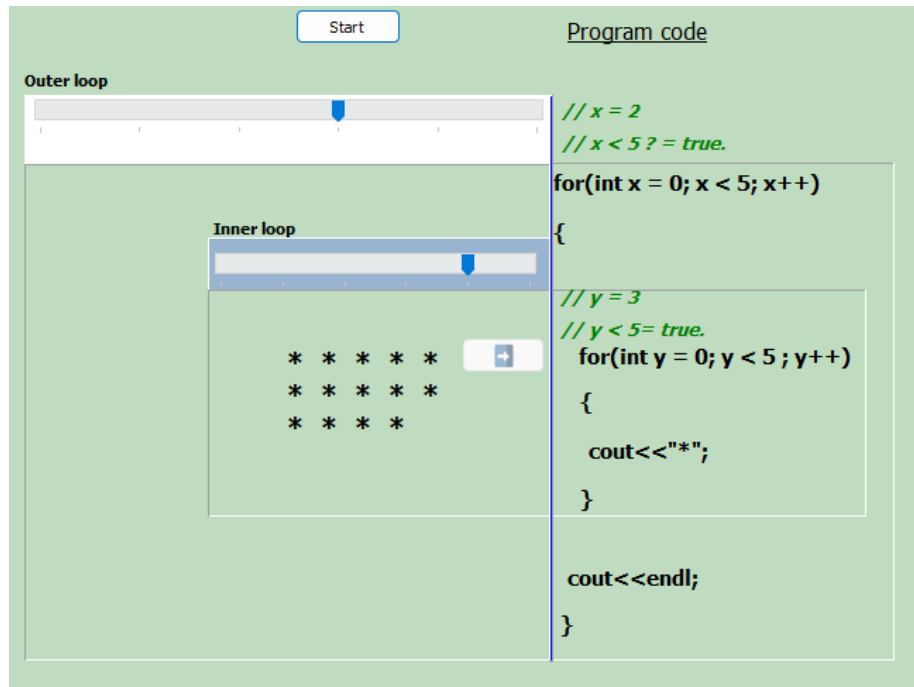


Figure 7.21: Nested-loop (Demo 2)

The additional text at the top of the outer loop indicates the value of  $x$ , the condition and if that condition has been evaluated to true or false. If the condition evaluates to true, the font of the text is green. If the condition evaluates to false, the font of the text turns red. In the case of Figure 7.22, the text at the top of the inner loop is red because  $y = 5$  and  $y$  is not less than 5; therefore, it evaluates to false. Subsequently, the arrow pointed at the `cout<<endl;` indicating the end of the line and the next star to be printed in the new line.

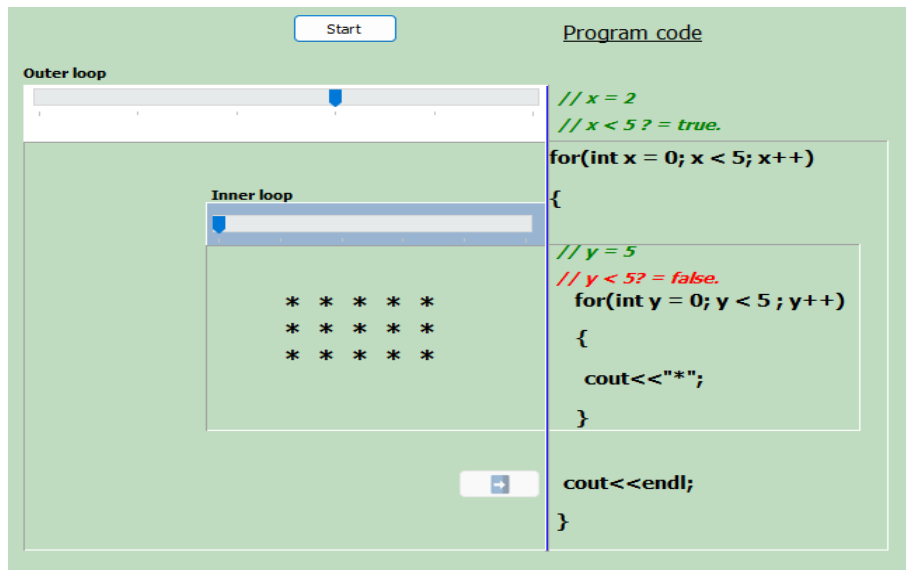


Figure 7.22: Nested-loop (Demo 3)

Figure 7.23 indicates the complete execution of the animation program. In that state, we can see that all stars have been printed (5 x 5) and the conditions of each loop are false (with red font).

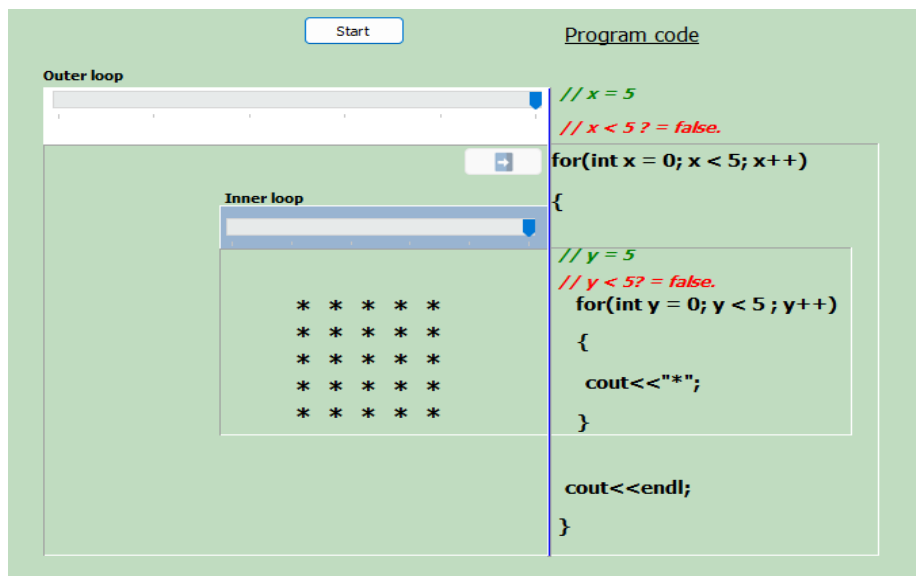


Figure 7.23: Nested-loop (Demo 4)

### 7.2.3 Arrays

The ILOs for the concept of one-dimensional arrays are for SIPS to write algorithms for storing and accessing values in a one-dimensional array. The ILO for parallel arrays is for SIPS to write a one-dimensional array algorithm that can be made parallel or relate to another one-dimensional array.

### 7.2.3.1 One-dimensional array – storing and access

The program in Figure 7.24 consists of a code and corresponding graphics. Graphics are tray-like and will help in storing values.

The screenshot shows a program interface with a green background. At the top left is a 'Start' button. Below it is a tray-like array graphic with six vertical bars. The indices are labeled 'Indices->' and numbered 0 through 5. The values are labeled 'Values->'. To the right is a 'Program code' section with a blue arrow icon pointing to the code. The code is as follows:

```
//storing values in the array
int sum = 0;
int number, arrNumbers[6];
for(int x = 0; x < 6; x++)
{
    cout<<"Enter number"<<endl;
    cin>>arrNumbers[x];
}

//accessing values from the array
for(int x = 0; x < 6; x++)
{
    cout<<arrNumbers[x]<<" - ";
    sum = sum + arrNumbers[x];
}
cout<<"\nSum of array elements "<<sum;
.....
```

Below the code is an empty white box labeled 'Output'.

Figure 7.24: Arrays (Demo 1)

When the arrow reaches the `cin` object, as usual it prompts the user to enter the value. However, in this case, each value is stored in the array as entered (see Figure 7.25).

The screenshot shows a program interface similar to Figure 7.24. It has a 'Start' button. A dialog box is open with the text 'Enter number 6' and 'OK' and 'Cancel' buttons. Below the dialog is the tray-like array graphic. The indices are labeled 'Indices->' and numbered 0 through 5. The values are labeled 'Values->' and show 12, 23, 10, and empty slots for 3, 4, and 5. To the right is a 'Program code' section with a blue arrow icon pointing to the code. The code is as follows:

```
//storing values in the array
int sum = 0;
int number, arrNumbers[6];
for(int x = 0; x < 6; x++)
{
    cout<<"Enter number"<<endl;
    cin>>arrNumbers[x];
}

//accessing values from the array
for(int x = 0; x < 6; x++)
{
    cout<<arrNumbers[x]<<" - ";
    sum = sum + arrNumbers[x];
}
cout<<"\nSum of array elements "<<sum;
.....
```

Below the code is an empty white box labeled 'Output'.

Figure 7.25: Arrays (Demo 2)

When the first loop is complete, the second loop takes over and prints the content of the array. SIPS should be able to see, as the loop's pace is reduced to a point where they can see how the values are printed during the looping process (see Figure 7.26). More importantly, the top of the array graphic contains an animating track bar that points to the currently accessed element in the array. SIPS are also able to see the

index of that currently accessed element at the bottom of the array tray and within the square brackets of `arrNumbers[4]`. The narrator/educator should emphasise this.

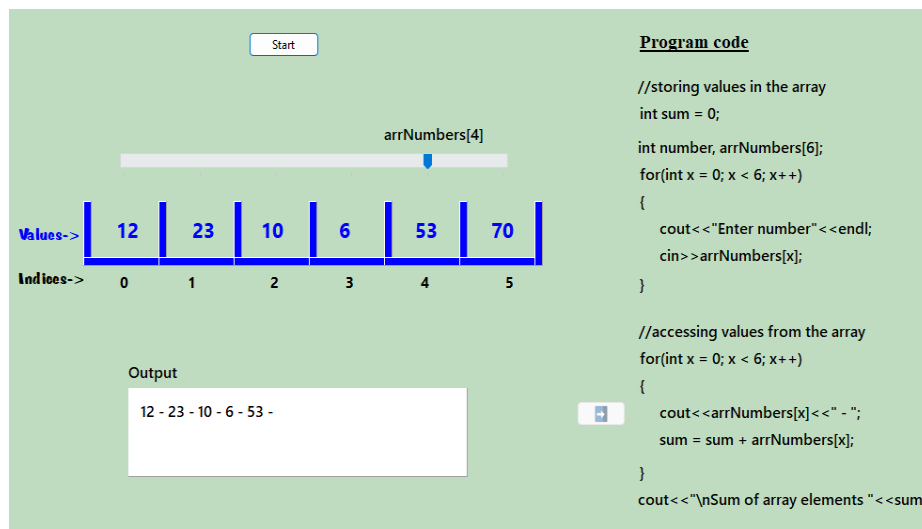


Figure 7.26: Arrays (Demo 3)

The second loop not only prints the content of an array but further determines the sum of array elements. SIPS should be able to see how to retrieve a value from an array (See Figure 7.27).

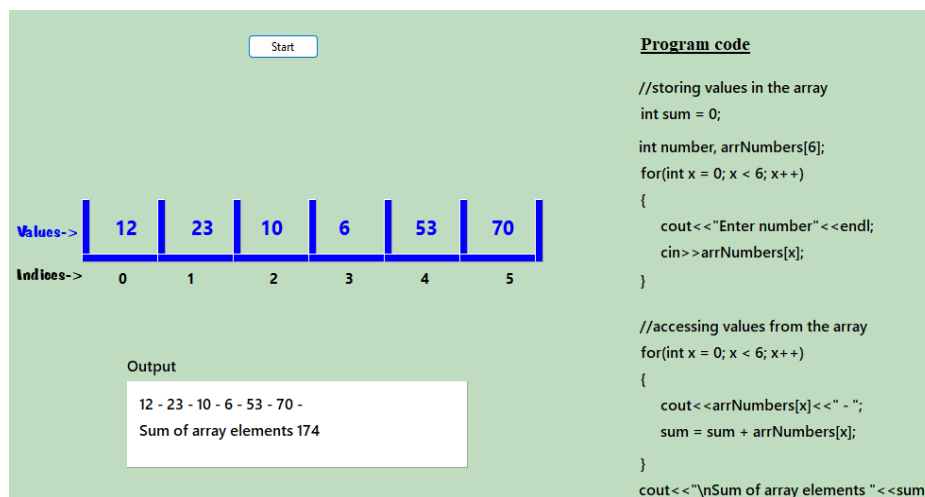


Figure 7.27: Arrays (Demo 4)

## 7.2.4 Functions

This section discusses introductory animation programs for teaching and learning *functions*. The ILOs for the concept of *functions* are for SIPS to construct a line of code

that calls a *function* that returns a value; that does not return a value; pass parameters (values and references) in the *function* calls; and construct proper *function* headers, parameters and *function* content in a way that maps other entities together. These ILOs are achieved with the introductory animation programs presented in this section and animation programs with problem specification presented in section 7.5.5.

### 7.2.4.1 Value returning function

Figure 7.28 shows the animation program for a *function* that returns a value to the calling environment.

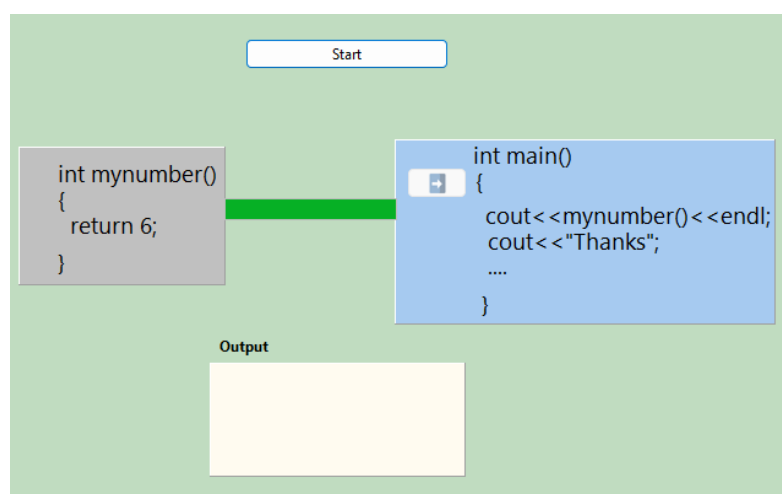
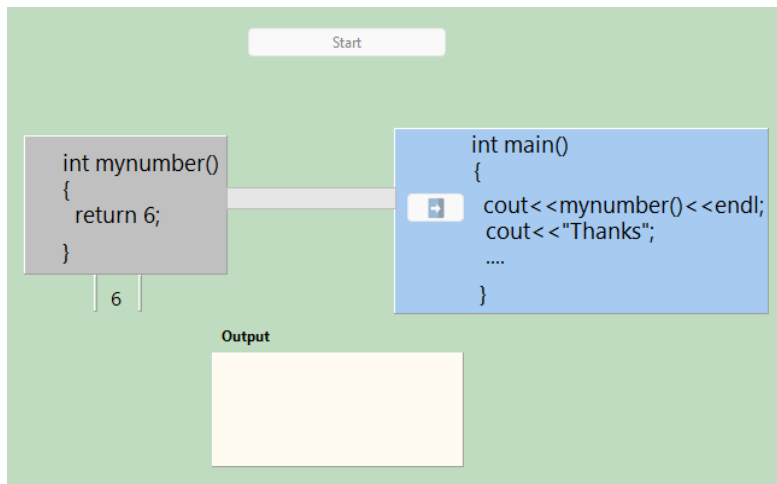


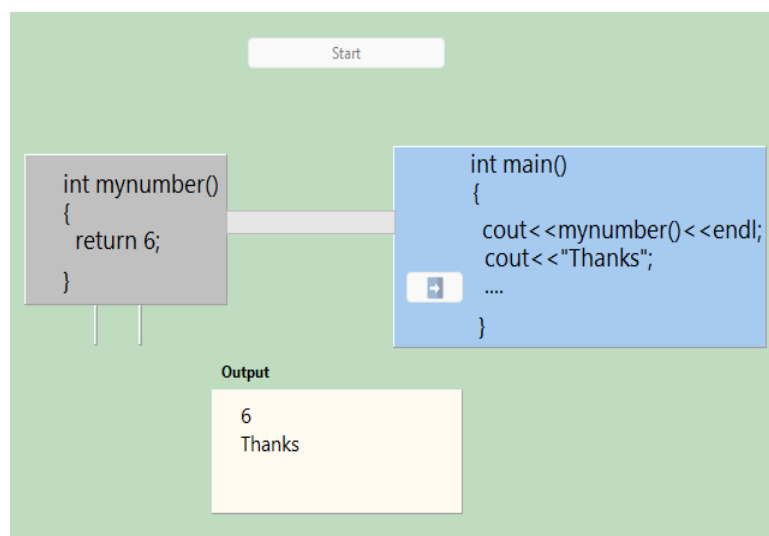
Figure 7.28: Value-returning function (Demo 1)

Upon a start button click, the arrow reaches `cout << mynumber()` then the green bar gets shaded by the grey colour towards `mynumber()` rectangle. The `mynumber()` *function* block gets shaded in a sky-blue colour then reverts to its original colour to catch the attention of the SIPS. This helps SIPS understand that there is a *function* call and the actual *function* gets executed when the *function* is called. Furthermore, if the *function* returns a value, that value is displayed through the `cout` object. To further emphasise that the *function* returns a value, a returned value gets animated as it floats from the bottom of the *function* (See Figure 7.29).



**Figure 7.29: Value-returning function (Demo 2)**

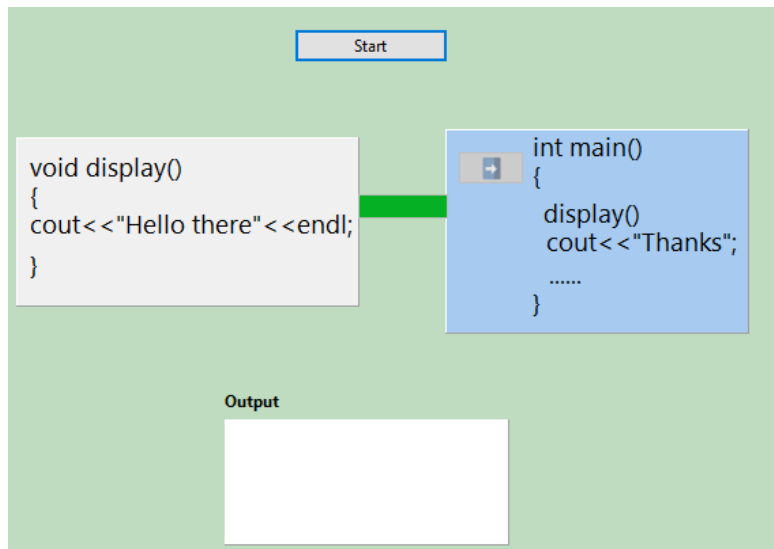
In Figure 7.30, the program has completed the execution and SIPS would have understood how the *function* is called, when is the *function* executed and how the value-returning *function* gets displayed.



**Figure 7.30: Value-returning function (Demo 3)**

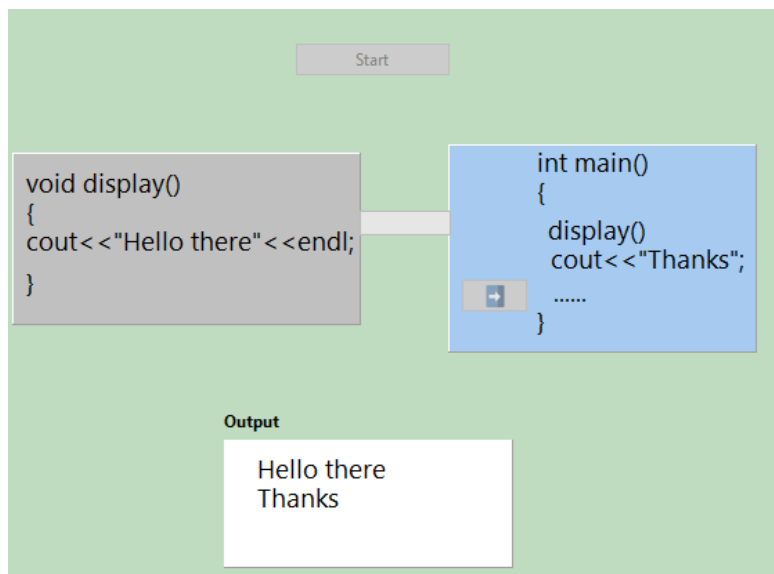
#### 7.2.4.2 Non-value returning function

The design of the animation program in this section is similar to the previous programs, except for changes in the *function* return type and the way the *function* is called. SIPS are expected to contrast between the two ways of calling a *function*. This will further solve the problem of confusion in calling *functions* as discussed in the literature review. The following two figures (See Figures 7.31 & 7.32) are snippets of the program.



**Figure 7.31: Non-value-returning function (Demo 1)**

Figure 7.32 shows a completed execution of the animation program. Note that the bar has gradually turned from green to grey when the arrow was pointing at a line (`display()`) that calls a *function*.

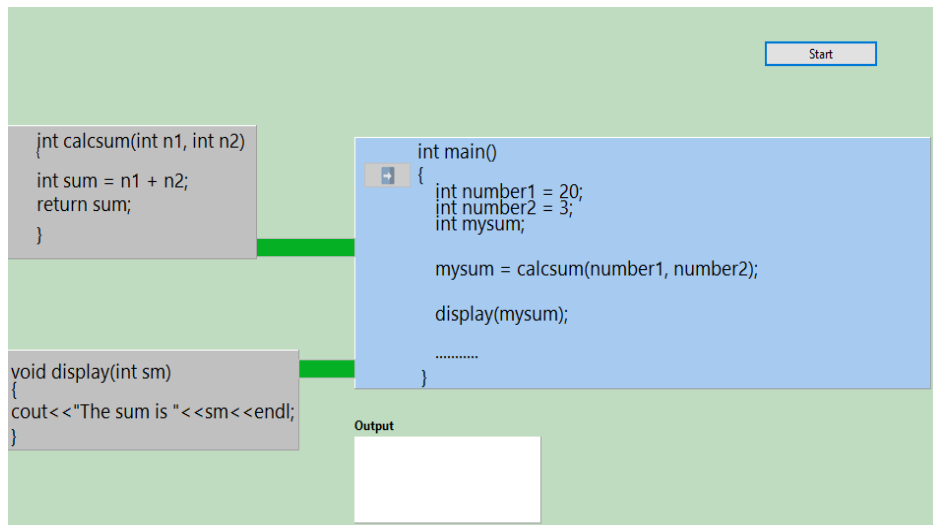


**Figure 7.32: Non-value-returning function (Demo 2)**

### 7.2.4.3 Value parameters

In this section, I demonstrate animation programs for introducing *functions* but with a special focus on value parameters. In Figure 7.33, I show the main *function* and two *functions*. One *function* has parameters and the other is without parameters. SIPS will be able to see what happens when a *function* with parameters is called.





**Figure 7.33: Value parameters (Demo 1)**

As usual, the arrow points line by line in the main *function*. When it points at `mysum = calcsum(number1, number2)` the grey of the bar moves from right to left towards a rectangle containing *function* code. SIPS also benefit from experiencing the alternative way of calling a *function* that returns a value, which is by storing a returned value in a variable. In section 7.2.4.1, the *function* was called in an output statement and in this case, a variable was declared to store a value returned by the *function*. The copies of `number1` and `number2` which are 20 and 3 appear at the top of the `calcsum` *function* (appear as numbers), then gradually move down, creating an animation that indicates values being received/copied into the *function*. When the graphics (20 and 3) reach the variables `n1` and `n2` they disappear, thereby making a demonstration that the values were copied into those variables (See Figure 7.34). The colour of the rectangle temporarily changes to blue to catch the attention of SIPS.

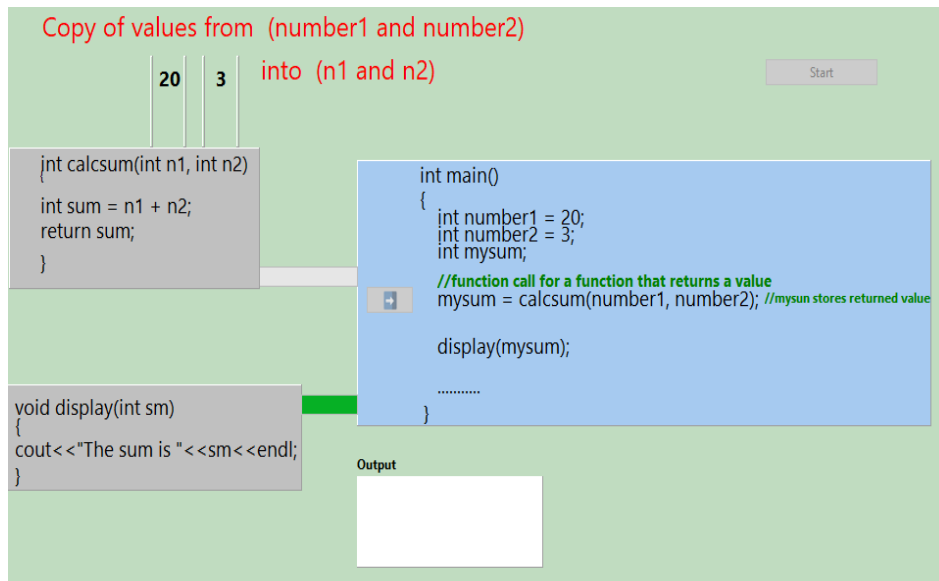


Figure 7.34: Value parameters (Demo 2)

Figure 7.35 is like Figure 7.34. In this case, a value calculated by the `calcsum` function is copied or received by the `display` function. The red font text (copy of `mysum`) further emphasises the value being copied to the `sm` variable.

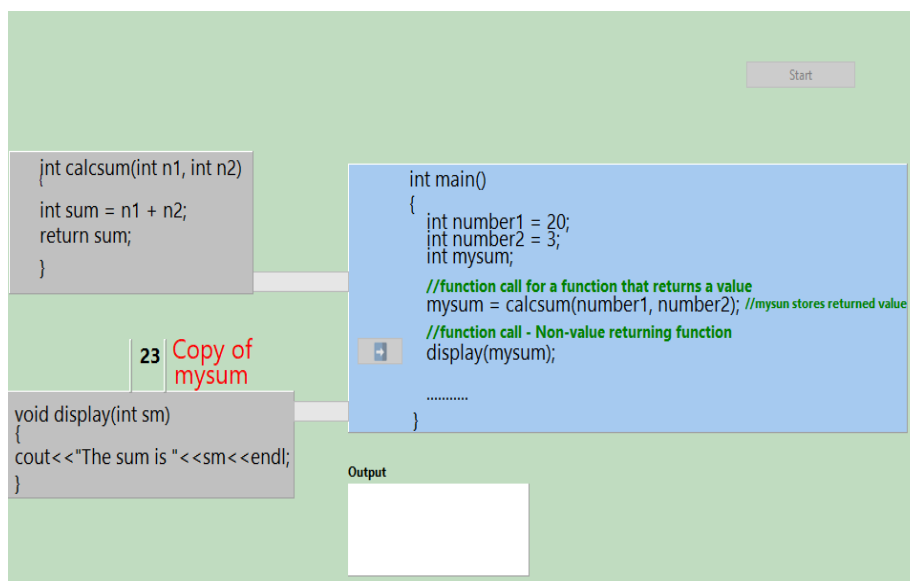


Figure 7.35: Value parameters (Demo 3)

When the animation program execution has been completed (as seen in Figure 7.36), SIPS would understand that the value parameters are copied and sent to be displayed in that function. The green font comments at the top of the function call serve as a notice to function calls.

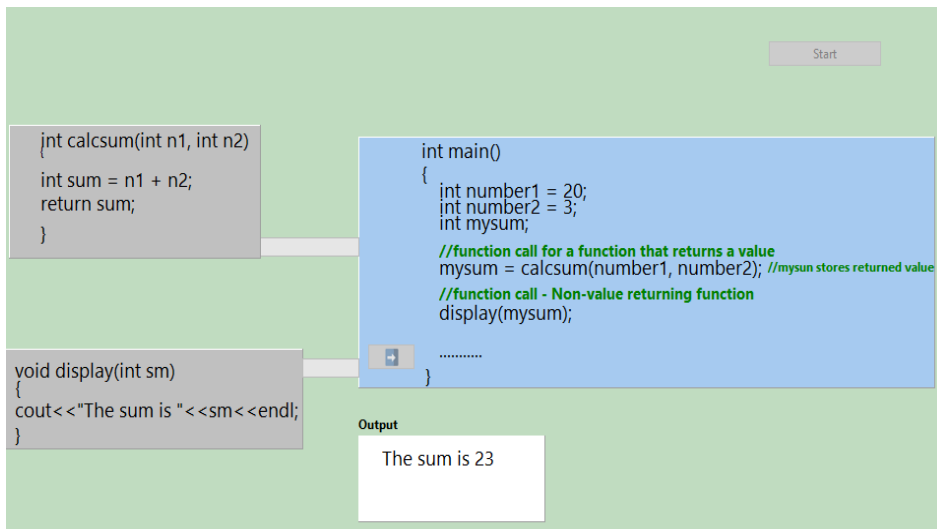


Figure 7.36: Value parameters (Demo 4)

#### 7.2.4.4 Reference parameters

This section demonstrates how reference parameters work. Figure 7.37 shows the main function and a non-value returning function. The function has two value parameters and one reference parameter.

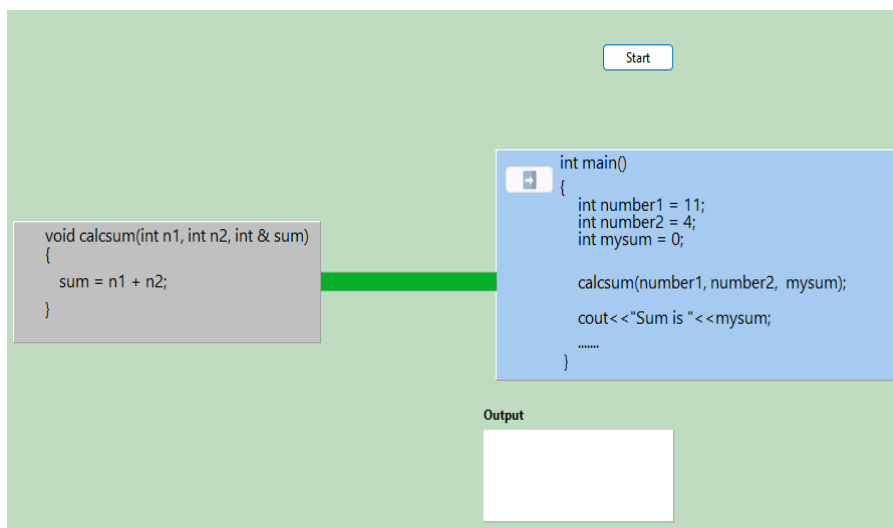


Figure 7.37: Reference parameters (Demo 1)

When the function executes, just like with the previous section, value parameters as graphics move down to be copied into `n1` and `n2`. However, with a reference parameter, the text reference indicates that the reference instead of a value is passed (see Figure 7.38).

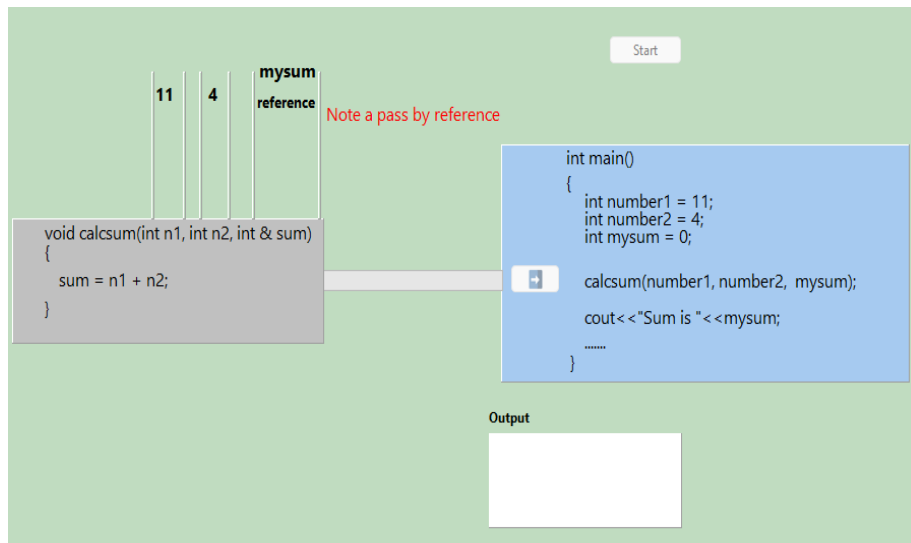


Figure 7.38: Reference parameters (Demo 2)

More importantly, SIPS see that when the sum changes in a *function*, the `mysum` value from the main *function* changes as well. The sum calculated by the `calcsun` *function* floats upwards as a number (15) towards `mysum` (Figure 7.39). When the number (15) reaches `mysum`, the pointing arrow moves to the next line (`cout<<"Sum is "<<mysum`) and the output will be displayed on the output window.

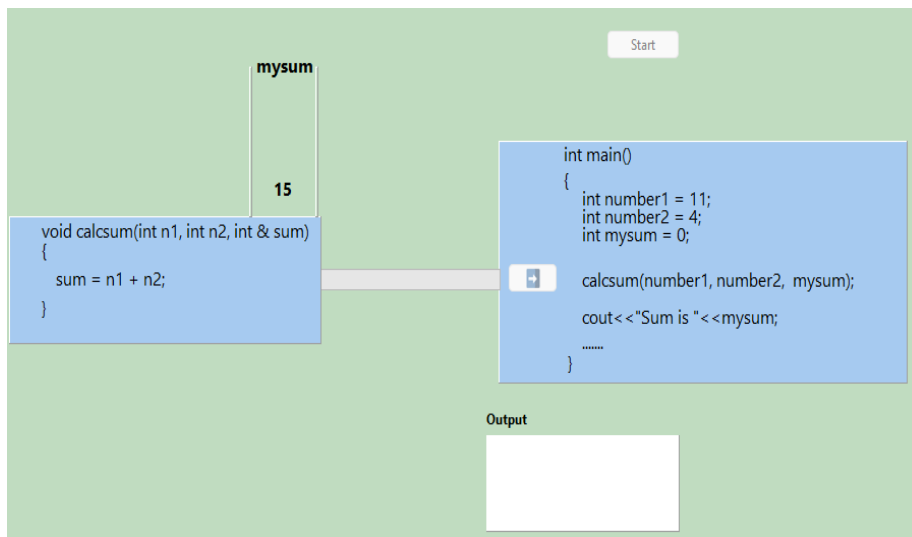


Figure 7.39: Reference parameters (Demo 3)

## 7.3 Animation programs with problem specifications

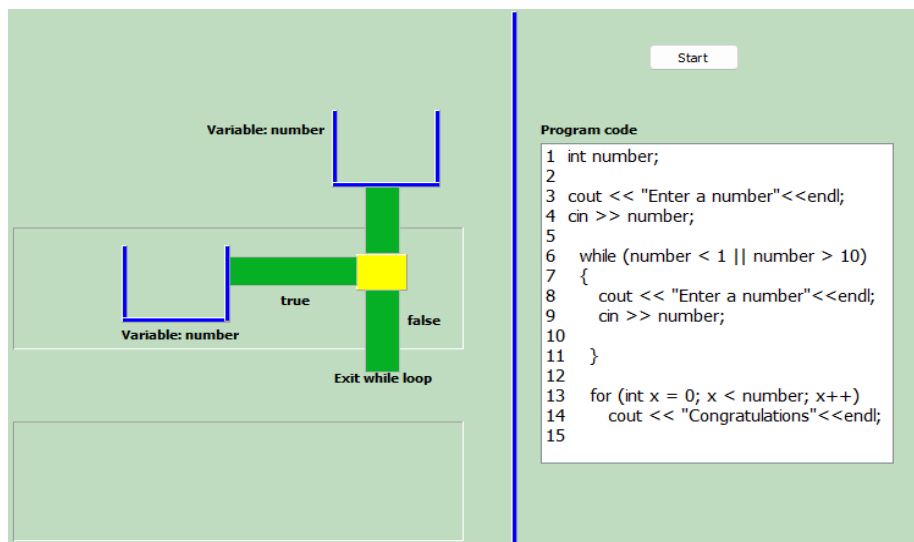
### 7.3.1 Repetitions

The demonstration of the animation program in this section is based on the problem specification in Figure 7.40.

*Write a program that asks a user to enter an integer between 1 and 10. Continue to prompt the user while the value entered does not fall within this range. When the user is successful, display a congratulatory message as many times as the value of the successful number the user entered.*

**Figure 7.40: Problem specification – loops**

Figure 7.41 shows the first appearance of the animation program. The borderline rectangle on the left side indicates a piece of code that is executed inside the loop.



**Figure 7.41: Loops (Demo 1)**

The first prompt is outside the loop; hence, the animated container is outside the rectangular borders. The prompt to enter a number happens at line 4 and in this case, the number 14 was entered (see Figure 7.42).

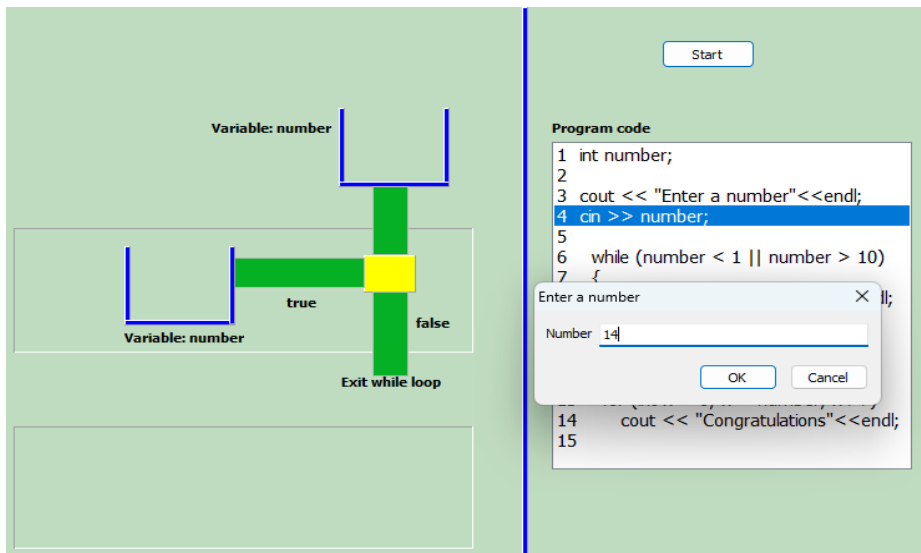


Figure 7.42: Loops (Demo 2)

In Figure 7.43, the blue highlight is in the loop and prompting for the second time is in line 9. This is because 14 is greater than 10 and therefore triggers a re-prompt as the condition evaluates to true (false or true = true). The green text at the top of the rectangle further indicates the true evaluation of the *while*-loop to the student.

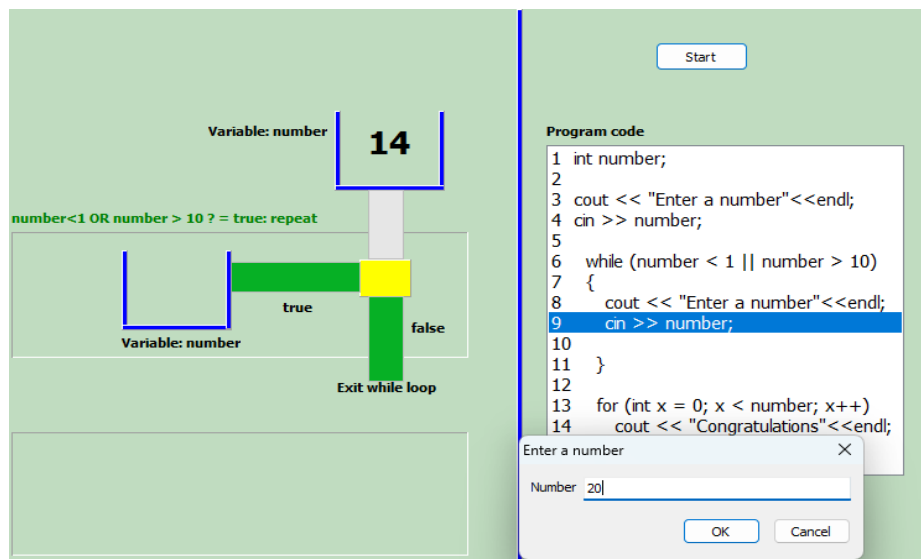


Figure 7.43: Loops (Demo 3)

In Figure 7.44, the number 3 is entered which is expected to evaluate to false because 3 is not less than 1 or greater than 10.

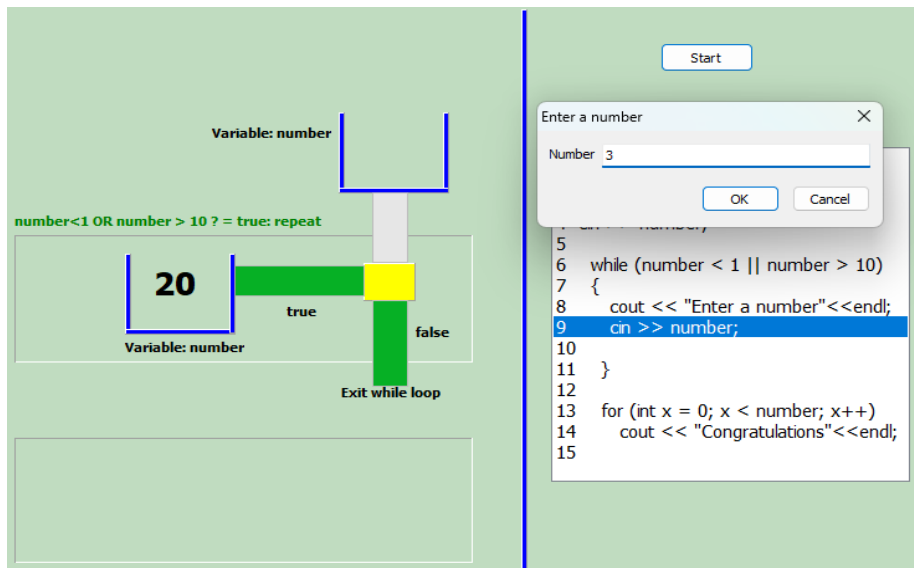


Figure 7.44: Loops (Demo 4)

Figure 7.45 indicates a completed execution of the animation program.

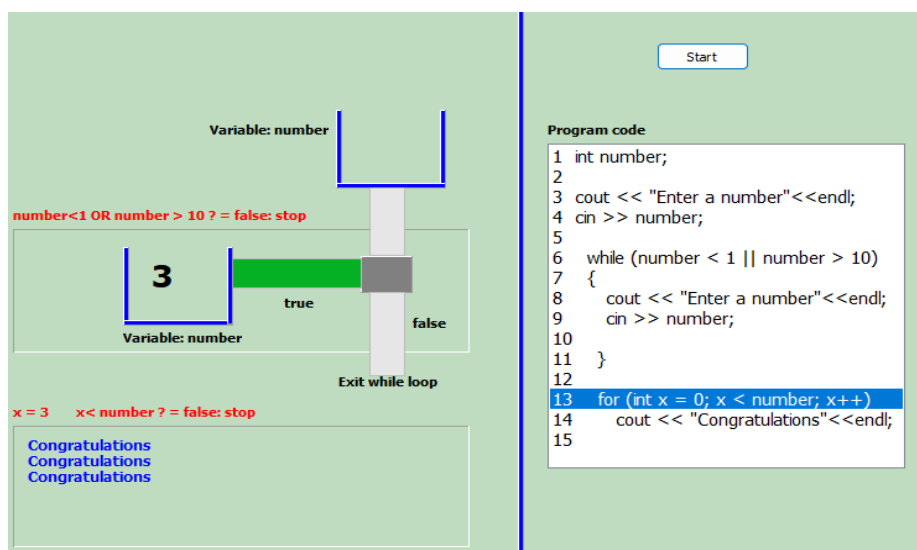


Figure 7.45: Loops (Demo 5)

### 7.3.2 Nested loop

This demonstrates the *nested loop* animation program aligned to problem specifications. The following figure (See Figure 7.46) is the problem specification for the animation program to be demonstrated.

Write a program code to produce the following output (note: Make use of a nested loop: any type of loop):

```
*  
* *  
* * *  
* * * *  
* * * * *  
* * * * * *  
* * * * * * *  
* * * * * * * *  
* * * * * * * * *  
* * * * * * * * * *
```

**Figure 7.46: Nested loop problem specifications**

The animation programs demonstrated through subsequent figures in this section helps SIPS understand that the inner loop reiterates based on the number of times the outer loop reiterates. The program code in Figure 7.47 has a borderline rectangular shape that contains the inner loop code and the big rectangle that covers the outer loop. This layout of the rectangles with borders denotes that a code will be repeated, and its uses should be explained before the animation program demonstration (pedagogical guidelines require the animation to be explained as well before demonstration). This is important to allow SIPS to comprehend the execution of the inner and outer loops as the navigating arrow moves up and down across the program code. The trackbars at the top of each rectangular shape show the status of the loop towards the final repeat. The blue arrow in the trackbar will take a step to the right direction until it reaches the tenth final step during animation program execution.



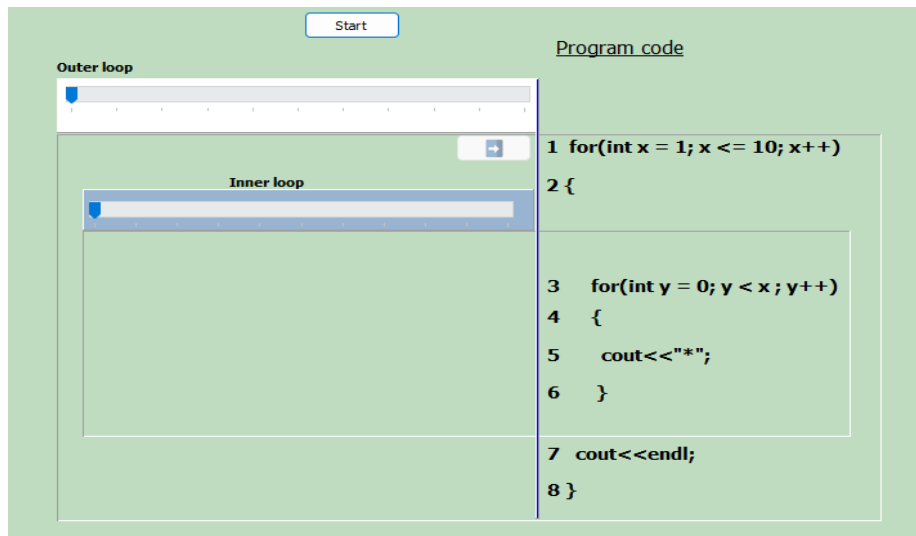


Figure 7.47: Nested-loop (Demo 1)

Upon button click, the pointing arrow starts from the first line which is the for-clause of the outer loop. While the arrow is pointing at the for-clause of the outer loop, the value of  $x$  gets updated with the current value and the condition gets evaluated and the results are printed at the top. In Figure 7.48 the inner loop had just completed looping four times and printed four asterisks on the last line. When the condition of the inner loop evaluates to false, the text at the top of the inner loop changes to a red font and is further updated as a comment at the top of the loop (see red font comment as  $y < x? = \text{false}$  in Figure 7.48).

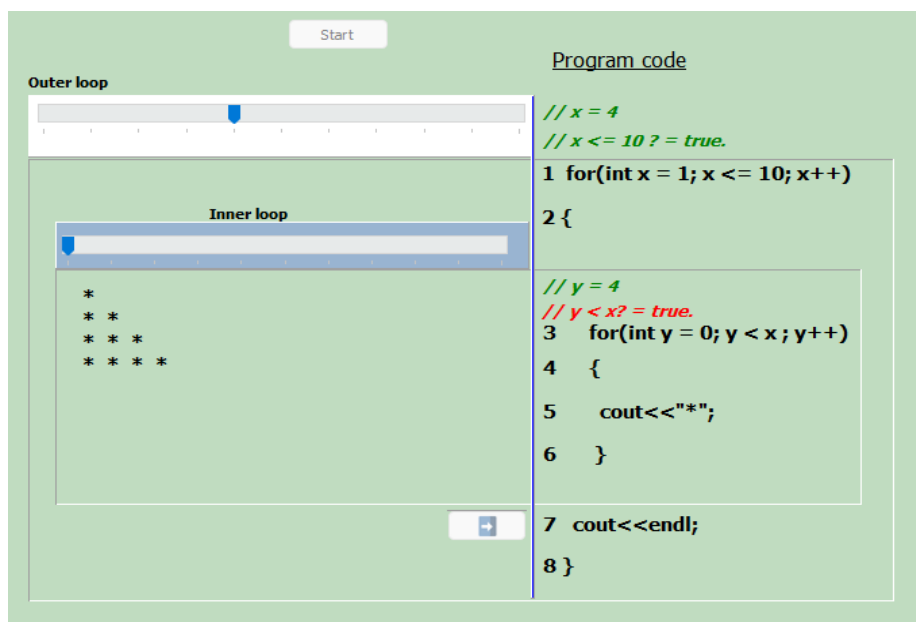


Figure 7.48: Nested-loop (Demo 2)

Figure 7.49 shows an executed animation program. At this stage, SIPS would have a basic understanding of the concept of a *nested loop*.

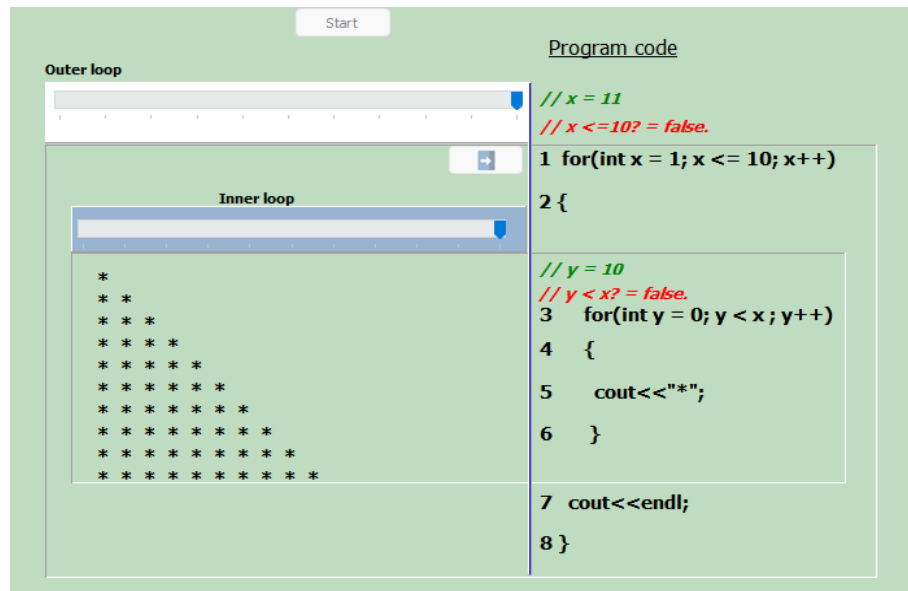


Figure 7.49: Nested-loop (Demo 3)

## 7.3.3 Arrays

### 7.3.3.1 One-dimensional array

The program animations in this section are based on the problem specification in Figure 7.50.

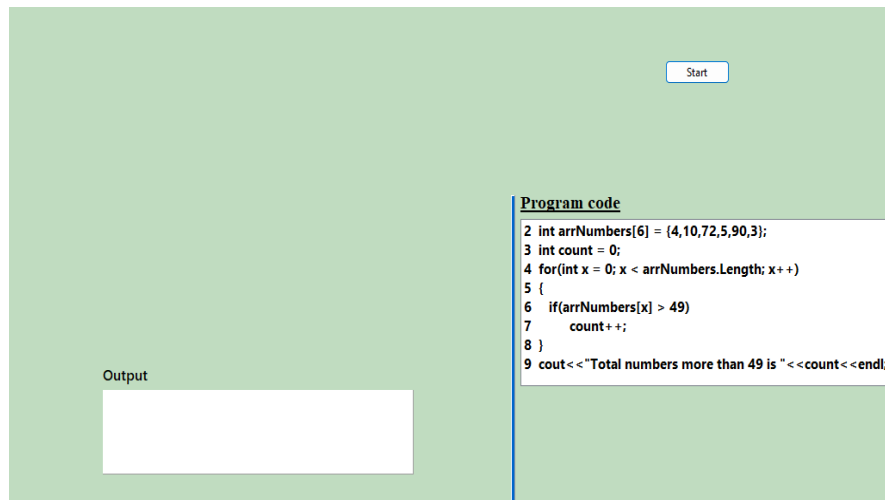
*Declare an array named arrNumbers and initialise it with 4, 10, 72, 5, 90, 3. Count the number of elements in the array that are higher than 49.*

*Sample output:*

*Total numbers more than 49 is 2*

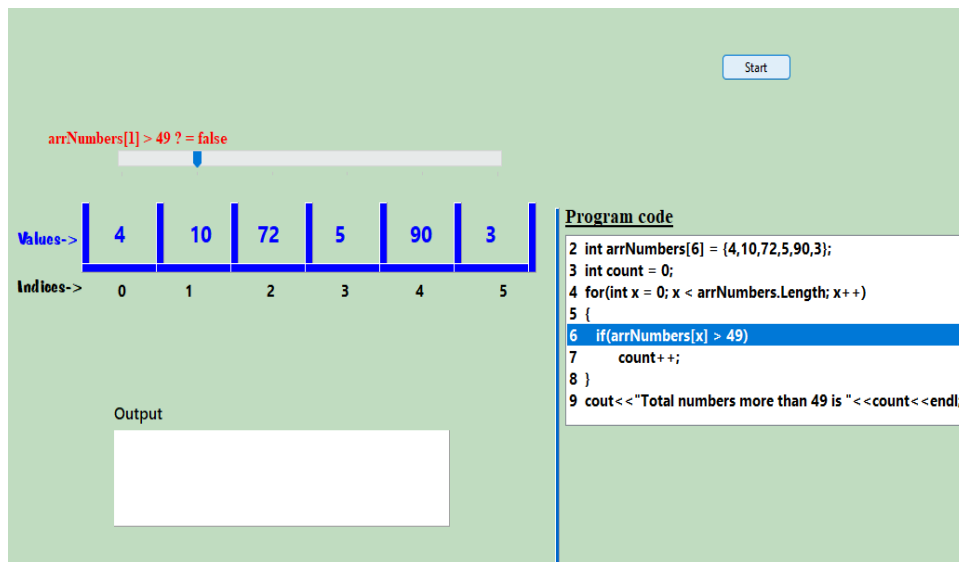
Figure 7.50: Problem specifications – arrays

The following set of figures will demonstrate the animation programs for arrays. Figure 7.51 is the state of the first appearance of the animation program and contains a program code and a display window for the program.



**Figure 7.51: Arrays (Demo 1)**

Figure 7.52 shows the initialised array and relevant graphics on the left. The progress bar at the top indicates the currently accessed element in the array in relation to the currently executed line in the program code. The text above the array animation changes font accordingly: if the condition evaluates to false the font is red, and if the condition evaluates to true the font is green.



**Figure 7.52: Arrays (Demo 2)**

In Figure 7.53, the condition is true at the pointed element and index. Subsequently, the font changes to green.

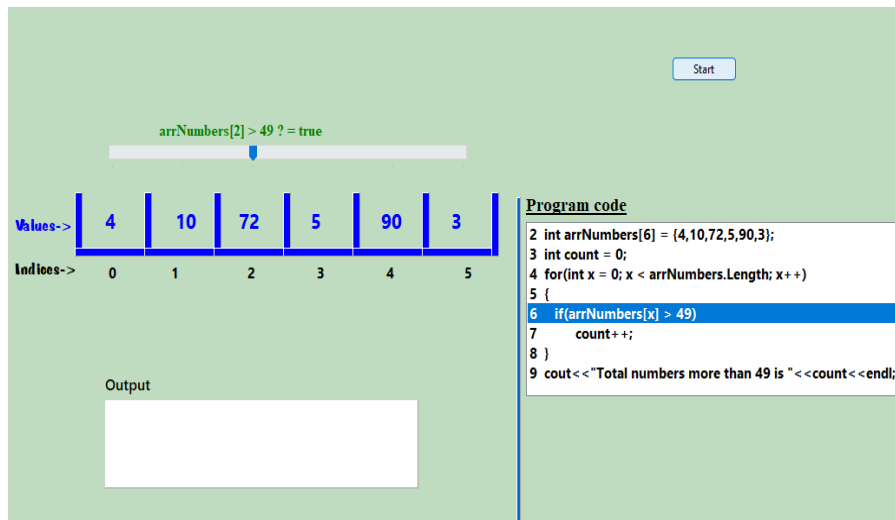


Figure 7.53: Arrays (Demo 3)

Figure 7.54 indicates a complete animation program execution.

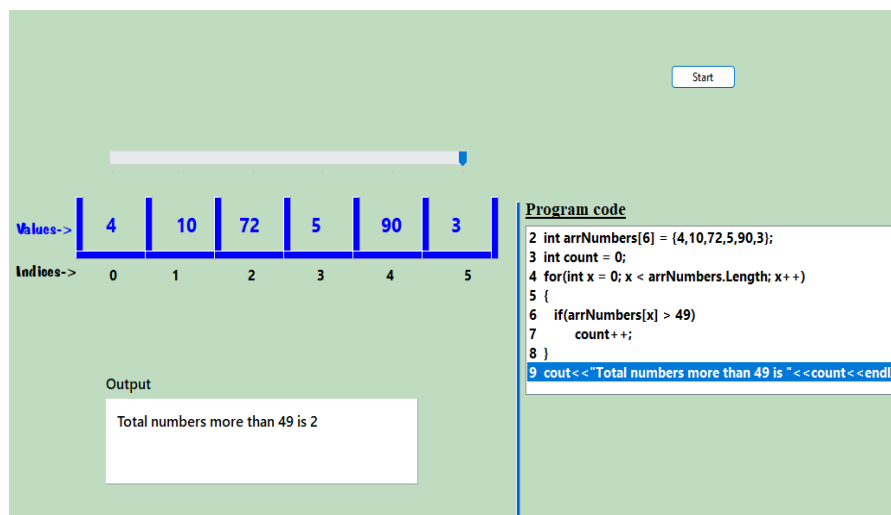


Figure 7.54: Arrays (Demo 4)

### 7.3.3.2 Parallel array

The content of Figure 7.55 is a problem specification for parallel arrays.

*Initialise an array named arrNames with these names: John, Dias, Linda, Obed, Jack and Ouma. Declare another array named arrAges that contains ages of the names in arrNames (parallel array). Initialise arrAges with 22,36,25,20,27,33. Prompt the user to enter the name of a person. If that name exists in the array, then display the age of that person. If that name does not exist in the array, show the message "Name does not exist".*

Figure 7.55: Parallel array problem specification

Figure 7.56 shows the program code and reserved space for the animation visualisation. The purpose is to demonstrate how parallel arrays work.

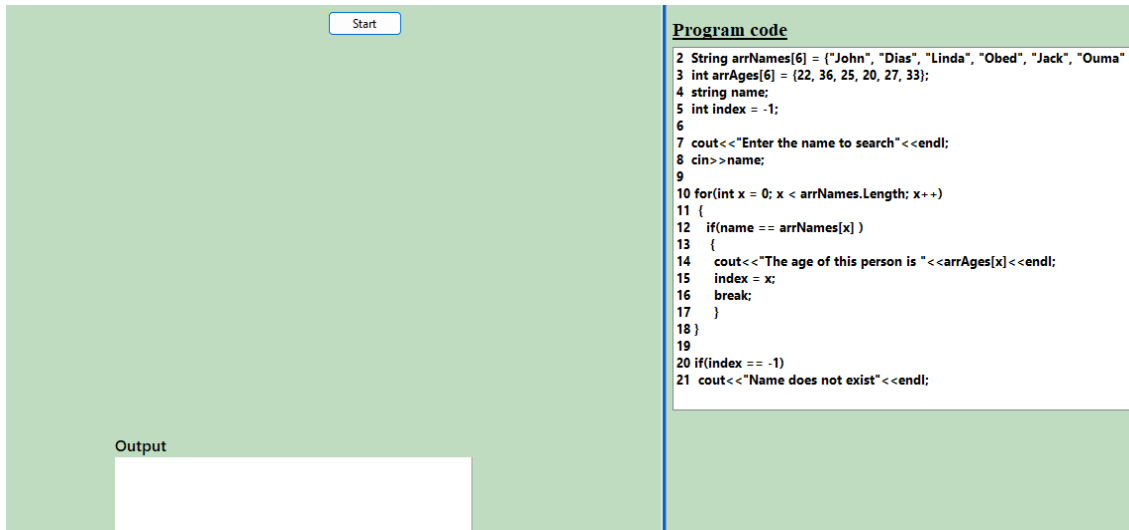


Figure 7.56: Parallel arrays (Demo 1)

Figure 7.57 indicates that a program is prompting for a name (a non-existing name is entered).

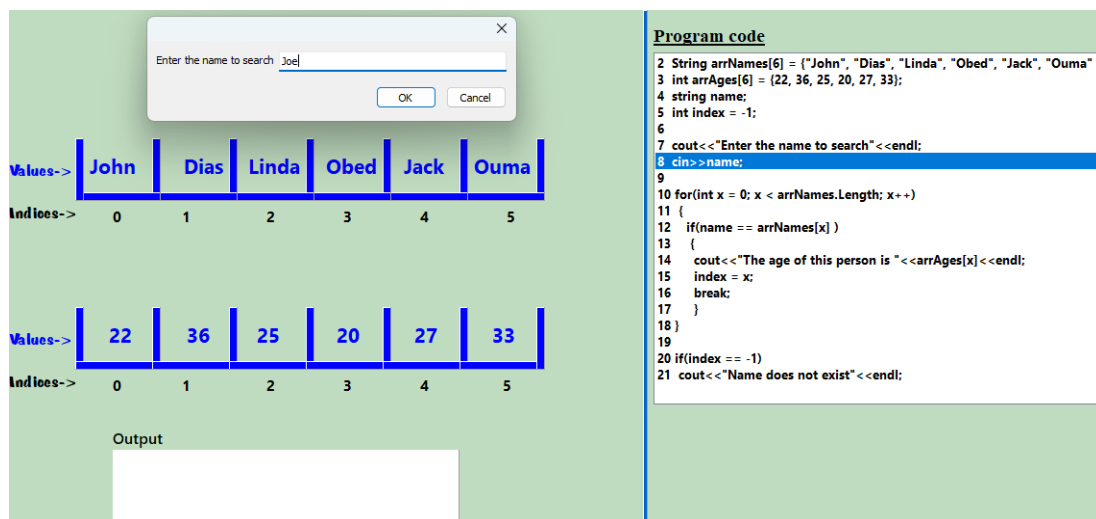


Figure 7.57: Parallel arrays (Demo 2)

Just like the previous section on a one-directional array, the searching happens with the assistance of the track bar (See Figure 7.58).

name == arrNames[i] ? = false

Values-> John | Dias | Linda | Obed | Jack | Ouma

Indices-> 0 | 1 | 2 | 3 | 4 | 5

Values-> 22 | 36 | 25 | 20 | 27 | 33

Indices-> 0 | 1 | 2 | 3 | 4 | 5

Output

Program code

```

2 String arrNames[6] = {"John", "Dias", "Linda", "Obed", "Jack", "Ouma"}
3 int arrAges[6] = {22, 36, 25, 20, 27, 33};
4 string name;
5 int index = -1;
6
7 cout<<"Enter the name to search"<<endl;
8 cin>>name;
9
10 for(int x = 0; x < arrNames.Length; x++)
11 {
12     if(name == arrNames[x] )
13     {
14         cout<<"The age of this person is "<<arrAges[x]<<endl;
15         index = x;
16         break;
17     }
18 }
19
20 if(index == -1)
21     cout<<"Name does not exist"<<endl;

```

Figure 7.58: Parallel arrays (Demo 3)

The name "Joe" does not exist in the array; this means that the condition evaluation of the elements were all false. Figure 7.59 shows that the search has completed and the name does not exist.

name == arrNames[i] ? = false

Values-> John | Dias | Linda | Obed | Jack | Ouma

Indices-> 0 | 1 | 2 | 3 | 4 | 5

Values-> 22 | 36 | 25 | 20 | 27 | 33

Indices-> 0 | 1 | 2 | 3 | 4 | 5

Output

Name does not exist

Program code

```

2 String arrNames[6] = {"John", "Dias", "Linda", "Obed", "Jack", "Ouma"}
3 int arrAges[6] = {22, 36, 25, 20, 27, 33};
4 string name;
5 int index = -1;
6
7 cout<<"Enter the name to search"<<endl;
8 cin>>name;
9
10 for(int x = 0; x < arrNames.Length; x++)
11 {
12     if(name == arrNames[x] )
13     {
14         cout<<"The age of this person is "<<arrAges[x]<<endl;
15         index = x;
16         break;
17     }
18 }
19
20 if(index == -1)
21     cout<<"Name does not exist"<<endl;

```

Figure 7.59: Parallel arrays (Demo 4)

In the case of Figure 7.60, a name that exists is entered.

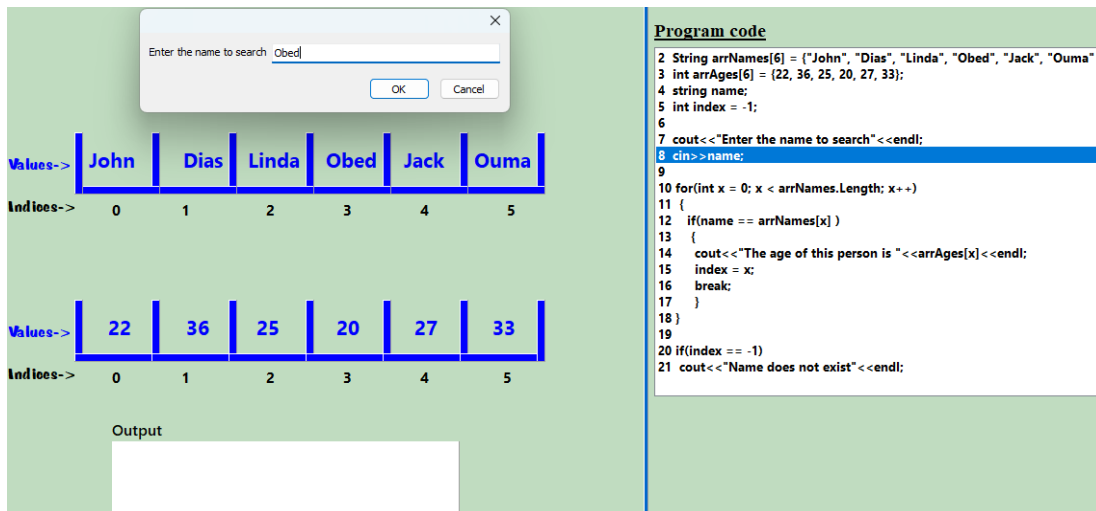


Figure 7.60: Parallel arrays (Demo 4)

The search continues until it reaches the position of "Obed". Subsequently, the text at the top of a track bar changes to `name == arrNames[3] = true` and also changes the font colour to green (see Figure 7.61). The program prints the output and exits the loop due to the break keyword.

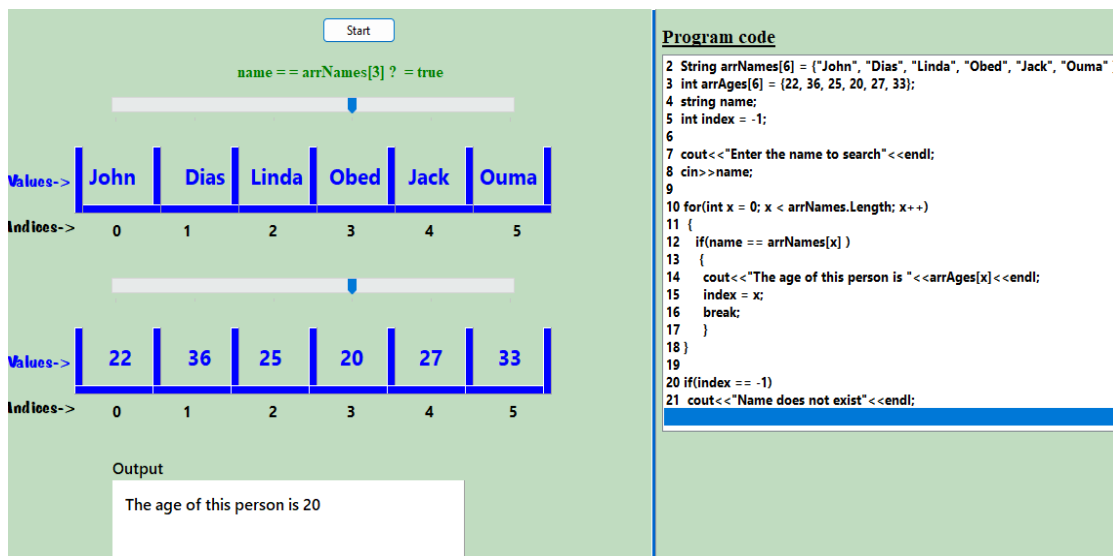


Figure 7.61: Parallel arrays (Demo 5)

### 7.3.4 Functions

The animation program presented in this section is based on the program specifications in Figure 7.62.

Maxwell Fitness Centre is a non-membership fitness centre where anyone can exercise and pay a certain amount per hour. Read and implement the following functions and the main function according to the specifications below.

**Main function**

Prompt the user to enter the name of a person, facility code, and the number of hours to use a certain facility. Note that a facility code is a type of an exercise. Call the necessary functions in the correct order and with relevant parameters where necessary.

**Functions**

- Create a function called **amountdue** that receives facility code and hours worked as parameters then calculates and returns the amount due. Refer to the table below for the calculation of the amount due.

Facility	Facility code	Rates
Squash court	squa	R13.00/hour
Swimming pool	swim	R12.00/hour
Gymnasium	gym	R11.00/hour
Weights	wei	R8.00/hour

- Create a function called **finalamountdue** of a type **void**. The function receives an amount due calculated and returned by the **amountdue** function, then determines the final amount due by adding 14% vat. The final amount due must be accessible to the main function.
- Create a function called **display** to display the name of the person entered earlier and the final amount due.

Figure 7.62: Problem specification – functions

Figure 7.63 shows four *functions*, the main *function* and three other *functions*. The *functions* cover value parameters, reference parameters, value-returning *function* and non-value-returning *functions*.

```

3 int main()
4 {
5     string facility, name;
6     int hrs;
7     int totalamt = 0;
8     double amount, finalamount;
9
10
11     cout << "Enter name of the person" << endl;
12     cin >> name;
13     cout << "Enter facility code" << endl;
14     cin >> facility;
15     cout << "Enter number of hours to use the facility" << endl;
16     cin >> hrs;
17
18     amount = amountdue (facility, hrs);
19     finalamountdue (amount, finalamount);
20     display (finalamount, name);
21     .....
22 }

double amountdue (string fcode,int hrs)
{
    double myamt;
    if(fcode== "squa")
        myamt= 13 * hrs;
    else if(fcode== "swim")
        myamt= 12 * hrs;
    else if(fcode== "gym")
        myamt= 11 * hrs;
    else if(fcode== "wei")
        myamt= 8 * hrs;
    return myamt;
}

void finalamountdue(double amt, double & f_amtdue)
{
    f_amtdue = amt * 1.14;
}

void display (double famount, string name)
{
    cout << name << ": your total amount due is R" << famount << endl;
}
    
```

Figure 7.63: Functions (Demo 1)



In Figure 7.64, the program prompts the user to enter a name, as per the currently highlighted line of code. Subsequently, the facility code (wei) and the number of hours (3) are entered.

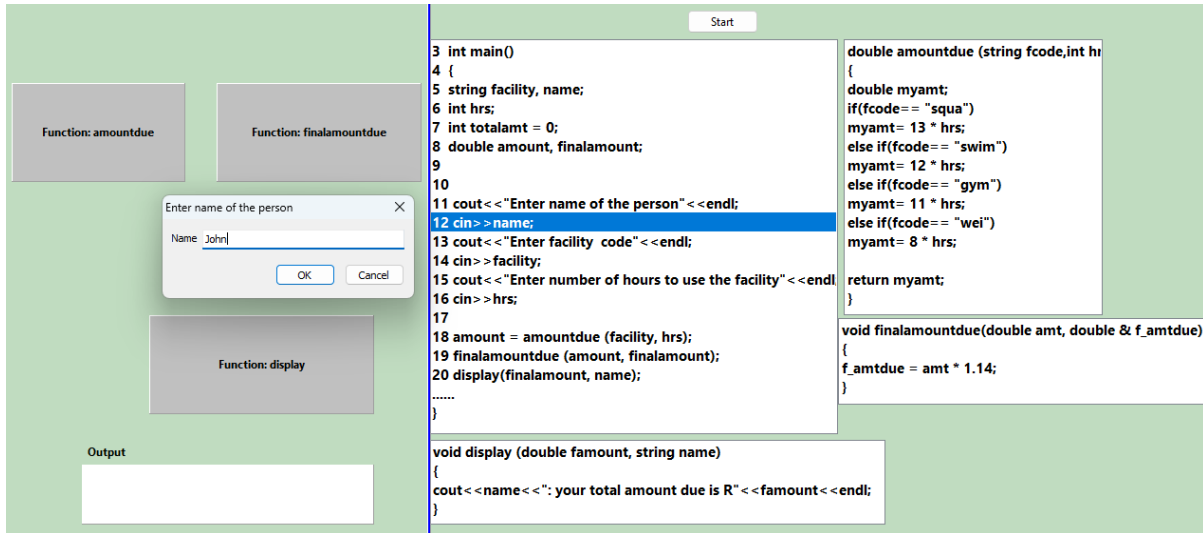


Figure 7.64: Functions (Demo 2)

On the animation side, the grey-coloured rectangle symbolises a function. In Figure 7.65, the blue highlight is on the *function* call, the background colour of the *function* being called changing to sky blue. The actual values sent as parameters appear at the top of the `amountdue` *function* as graphic text, then gradually move down as a sample of copied parameters. When the values ("wei" and "3") reach the parameters ("fcode" and "hrs"), they disappear with an illusion of being copied into those variables.

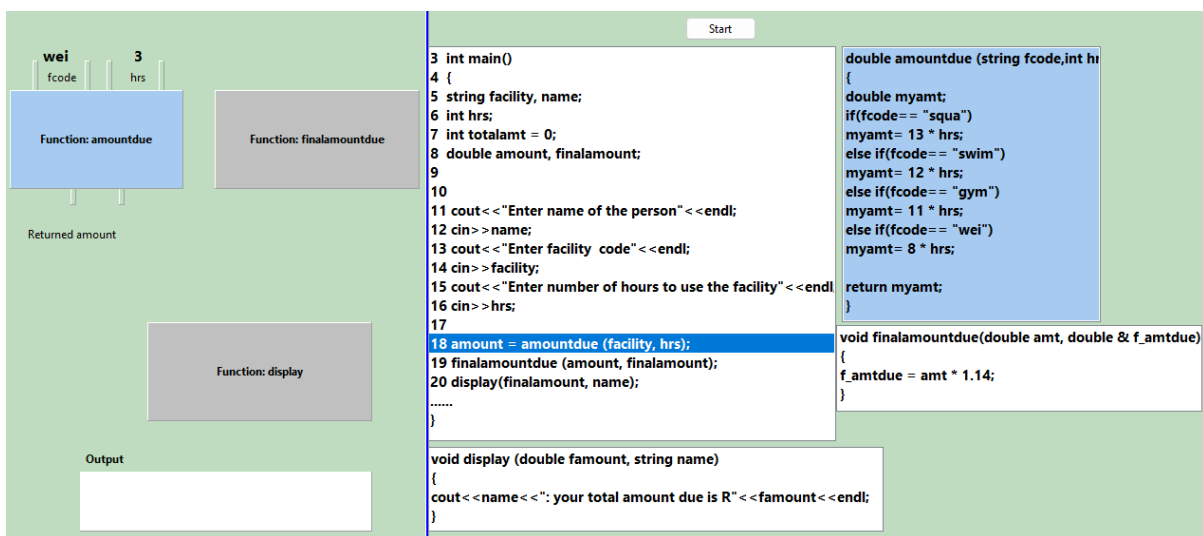


Figure 7.65: Functions (Demo 3)

In Figure 7.66, the value being returned comes out at the bottom of the *function* between the two bars.

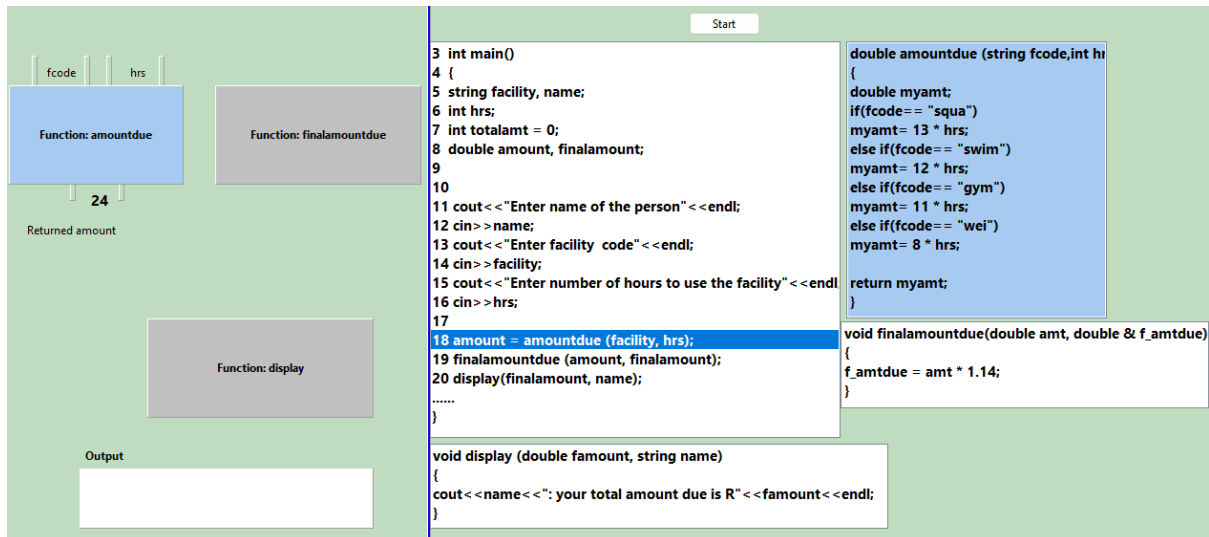


Figure 7.66: Functions (Demo 4)

In Figure 7.67, another *function* call is executed and the program highlighted it in sky blue as well. The `finalamountdue` *function* has two parameters: the value parameter and the reference parameter. The value (24) floats into the `amt` variable just like with the previous *function*.

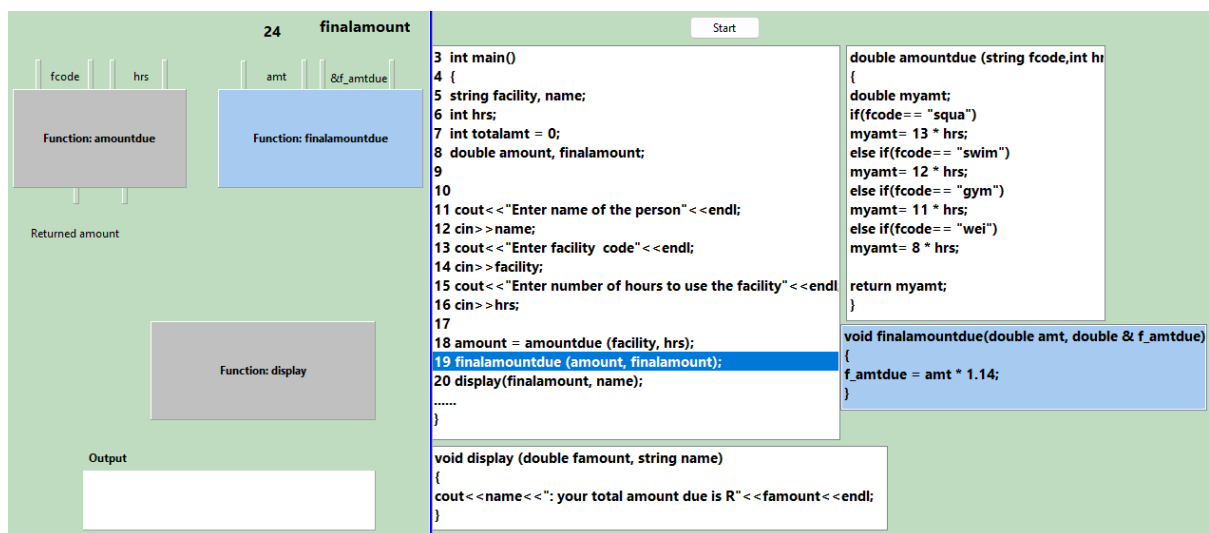


Figure 7.67: Functions (Demo 5)

Figure 7.68 shows the calculated value from the `finalamountdue` function move up and vanish after reaching the `finalamount` variable.

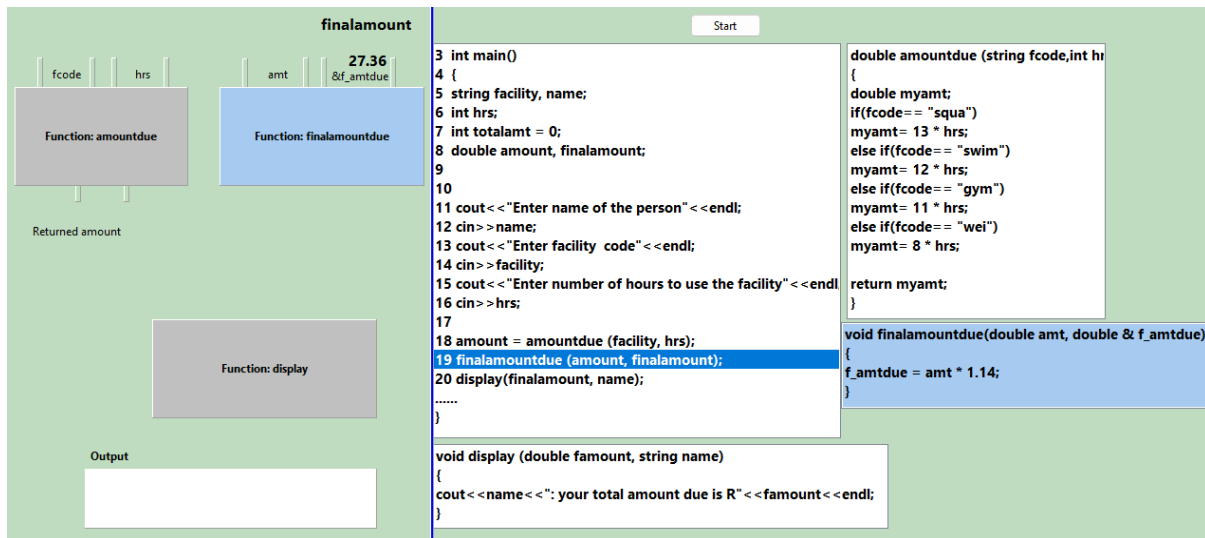


Figure 7.68: Functions (Demo 6)

In Figure 7.69, the last function named `display` receives the values and displays accordingly.

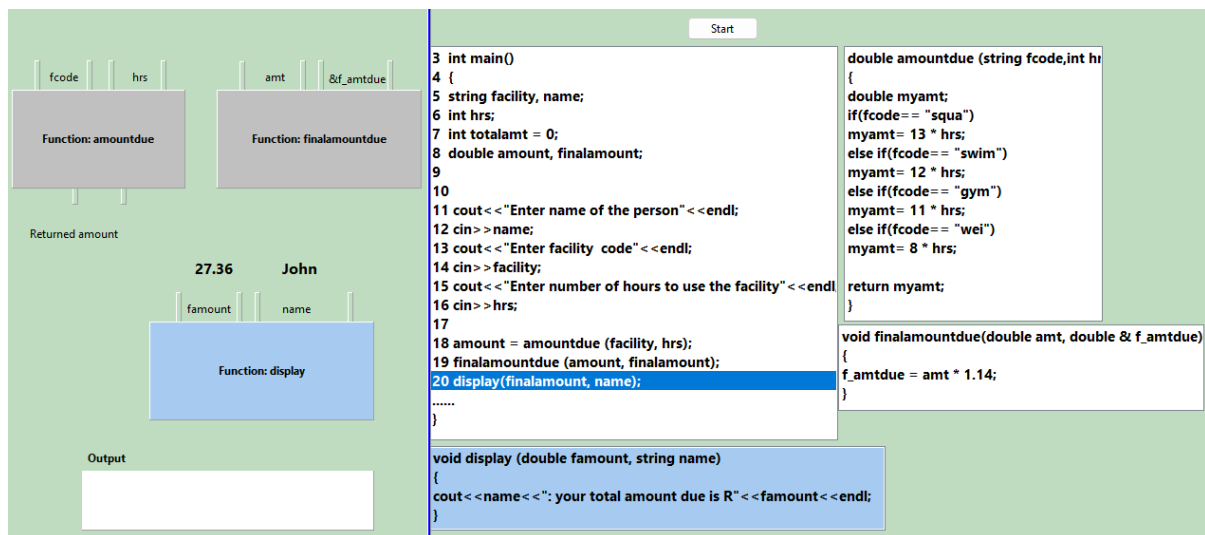


Figure 7.69: Functions (Demo 7)

Figure 7.70 is the state of a completed execution of the animation program.

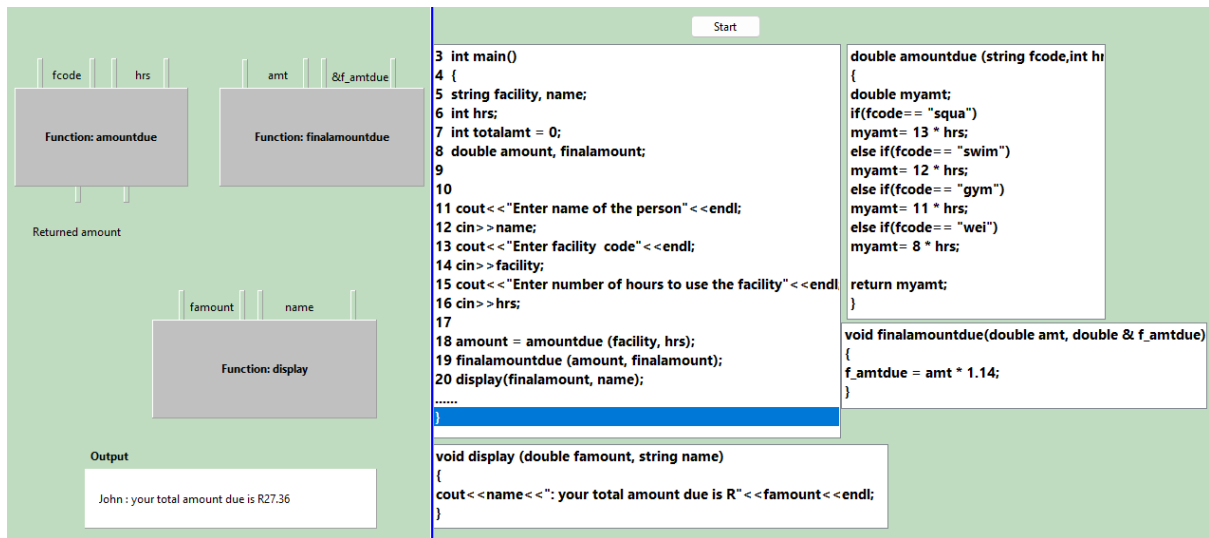


Figure 7.70: Functions (Demo 8)

## 7.4 Adapted pedagogical guidelines

The following guidelines (see Figure 7.71) were revised from cycle 2 (Chapter 6, section 6.5), adapted and followed when all animation programs were demonstrated in this cycle.

- Stage 1** *Introduce a concept by giving a background and examples.*
- Stage 2** *Prepare a computer multicast or a big projecting image then explain the role of animation and its components.*
- Stage 3** *Use an animation program for concept introduction.*
- Stage 4** *Repeat stage 3, then move to stage 5.*
- Stage 5** *Ready the problem specification and its animation program.*
- Stage 6** *Use an animation program with problem specification.*
- Stage 7** *Repeat stage 6 and ask if there are questions*

Figure 7.71: Adapted pedagogical guidelines

The TLPAP animation programs are designed in way that enable the narrator to emphasise certain elements of the concept. The assumption is that these guidelines

should be followed after the narrator (educator) has familiarised himself or herself with how the animation programs work, as discussed in sections 7.2 and 7.3. This will assist in transitioning TLPAP to the SIPS. The pedagogical guidelines in Figure 7.71 are substantiated as follows.

**Stage 1:** Introduce a concept by providing background information and examples.

- Giving a good background helps students grasp the overall basic idea about the concept, it initiates a high-level understanding before the educator can explain fine details of the program code and it is further helpful to avoid conceptual bugs (Pea, 1986).
- Give example or metaphor about the concept. The use of examples ignites excitement, confidence and enthusiasm and plays a vital role in learning programming (Malan & Halland, 2004). The use of metaphors or analogies in introductory programming enhances understanding (Moape, Ojo & van Wyk, 2017).
- Examples should not be too complex or too abstract as they can do more harm than good (Malan & Halland, 2004). According to Malan and Halland (2004) examples like "foo" and "bar" are meaningless to students; therefore, in this framework, we recommend real-world examples which are commonly familiar. Real-world examples help students link the context to a programming concept for better understanding (Konecki, Lovrenčić & Kaniški, 2016).
- Having given students some background in this stage, that knowledge becomes important in the next stages to link the example into context. Discussion and interaction should carry on throughout as students' reconstruction processes continues.

**Stage 2:** Prepare the computer multicast or a big projecting image; explain the role of animation and its components.

- A multicast can be prepared in a contact session where the controller computer displays the same content on all computers. Alternatively, multicast can be prepared in an online platform like Microsoft Teams, Google Meet or Zoom. Another option is a large projector screen visible to all students in the room.

- Explain that the purpose of the animation programs is to enhance programming understanding to ensure that students do not regard the animation program as an independent entity. This will help students regard animation programs as one way of learning and understanding how program code works. Highlight the components of the animation where possible.

**Stage 3:** Use an animation program for concept introduction.

- The first appearance of all our animation programs shows a program code with no animation or graphics at this point. Before the educator can press the start button, a verbal narration of how that program works should be prepared. This is done so that when the narration is presented with animation later, the students can construct their knowledge by linking it with previous knowledge or triggering aspects that were not clear in the program code.
- Secondly, play the animation program and narrate synchronously. Mayer and Anderson (1991), refers to this approach as the 'words-with-pictures' approach, found it to be more effective than the 'words-before-pictures' approach. With words-with-pictures, the narration and the animation occur synchronously. With words-before-pictures, the narration happens first without the involvement of pictures (animation), and later the animation plays without a supporting narration.

**Stage 4:** Repeat stage 3, then move to stage 5.

- The repeat is necessary for SIPS to further grasp the fundamental aspects of the concept introduced. If the program allows different values, a repeat with different values is recommended. If the program gives a varied flow of a program, it is recommended that the number of repeats be made accordingly.

**Stage 5:** Ready the problem specification and its animation programs.

- A program specification refers to the description of what the student should program to solve the problem. The solution to the problem requires a program code or algorithm. This should be in written format rather than verbal instruction as the students will benefit from reference.

- Do not use difficult terminology when writing a problem specification as terms may be misinterpreted and mislead students to develop incorrect program code. Simplify the language with basic, common terminology (Craig, Smith & Petersen, 2017).
- Make sure students understand this part very well as it can reduce misconceptions on problem specification and reduce barriers to designing a good program code.
- Ask the students to identify inputs and outputs as this will serve as continuous active learning and engagement throughout the teaching and learning session.
- Open the animation program for the explained problem specification as the preparation to carry out stage 6.

**Stage 6:** Use an animation program with problem specification.

- At this stage, students should have grasped the problem specification, noted the inputs and outputs and understand the role of that animation program as already introduced through animation programs for concept introduction.
- Let an animation program designed for that specific concept/problem play. The lecturer narrates the program code in conjunction with the animation unfolding during the process (Végh & Stoffová, 2017).

**Stage 7:** Repeat stage 6 two times and then call for questions.

- The three times repeat serves as a minimum; otherwise one can repeat several times accordingly as required. In the case where values are entered, it is recommended that alternative values be used in each repeat.
- The repetition of the stage serves to further sharpen program understanding.

## 7.5 Experimentation results and analysis

This section reports on the experimental results in cycle 3 in 2021 semester 2. The results in section 7.5.1 use the animation programs for *assignment* discussed in cycle 2 (Chapter 6, section 6.2). The results in section 7.5.2 are based on the animation program for problem specification discussed in cycle 2 (Chapter 6, section 6.3). However, in this cycle, the animation program for problem specification is coupled with

the animation program for introducing a concept. The results in section 7.5.3 and 7.5.4 are based on the same problem specification as in cycle 2 (Chapter 6, section 6.4.1), but the animation program for *loops/nested loops* has been updated. The animation programs experimented and reported for the first time are for *nested loops* (section 7.5.4), one-dimensional and parallel arrays (section 7.5.5) and *functions* (section 7.5.6). Therefore, the problem specifications will be explained in those sections.

### 7.5.1 Results (assignment)

The table (Table 7.1) reports on the control and experimental group results for both pre-teaching and post-teaching exercises on the uses of *assignment*. A total of 28 students were in attendance. The students were equally but randomly divided between the groups. The control group had 13 students and the experimental group had 15. In the first column (control group) under pre-teaching exercise 1 the  $x (7) - 66.6\%$ , the number 7 in the brackets means the number of students who got the value of  $x$  correct and the subsequent number represents the same number as a percentage. All other columns have been calculated the same way.

**Table 7.1: Results (Assignment)**

Control group		Experimental group	
Pre-teaching exercises	Post-teaching exercise	Pre-teaching exercises	Post-teaching exercises
<b>Pre-teaching exercise 1</b> $x (7) - 53.8\%$	<b>Post-teaching exercise 1</b> $t (8) - 61.5\%$ $p (7) - 53.8\%$ $k (7) - 53.8\%$	<b>Pre-teaching exercise 1</b> $x (8) - 53.3\%$	<b>Post-teaching exercise 1</b> $t (13) - 86.7\%$ $p (11) - 73.3\%$ $k (12) - 80\%$
<b>Pre-teaching exercise 2</b> $p (9) - 69.2\%$ $y (8) - 61.5\%$	<b>Post-teaching exercise 2</b> $t (8) - 61.5\%$ $p (7) - 53.8\%$ $k (6) - 46.2\%$	<b>Pre-teaching exercise 2</b> $p (8) - 53.3\%$ $y (9) - 60\%$	<b>Post-teaching exercise 2</b> $t (12) - 80\%$ $p (13) - 86.7\%$ $k (11) - 73.3\%$
<b>Pre-teaching exercise 3</b> $k (6) - 46.2\%$ $t (7) - 53.8\%$		<b>Pre-teaching exercise 3</b> $k (8) - 53.3\%$ $t (7) - 46.7\%$	
<b>Pre-teaching exercise 4</b> $x (8) - 61.5\%$ $y (6) - 46.2\%$		<b>Pre-teaching exercise 4</b> $x (7) - 46.7\%$ $y (8) - 53.3\%$	

The pre-teaching and post-teaching exercises emphasised the declaration of a variable, initialisation and overwriting (copying a value from one variable to another) through animation programs. If SIPS get the values correct, the assumption is that



they would have understood initialisation and overwriting which happens using an *assignment* operator.

The experimental group shows a high number of students who got the answers correct in the post-teaching exercise (see the percentage on the value of  $t$ ,  $p$  and  $k$  in table 7.1). Based on the post-teaching exercises, the values of  $t$ ,  $p$  and  $k$  are initialised and changed accordingly through an *assignment* operator. The results of the post-teaching exercise in the experimental group are higher as compared to the pre-teaching exercise and as compared to the post-teaching exercise in the control group. The main aspects of this experiment were to instil proper understanding of how *assignment* works in programming, as this is a significant challenge of learning to program.

### *7.5.2 Results (decisions/nested decisions)*

The nested-decider experiment was repeated during the second semester of 2021. For more accurate analysis and comparison, both pre-teaching and post-teaching algorithms were dissected into comparative sections for SOLO-adapted evaluation purposes. The description of the transitions in the matrix applies as defined in the methodology section (Chapter 4, section 4.5). This SOLO-adapted evaluation also applies to all other experiments in this cycle. There were 21 SIPS participating, 10 SIPS were in the control group and 11 in the experimental group. The following section presents necessary information on the algorithm sections/pairs for *decisions/nested decisions* to ready the SOLO-adapted evaluations.

#### *7.5.2.1 Algorithm sections*

The problem specification and its algorithm for the pre-teaching exercise were explained in Chapter 6, section 6.3.1 in Figure 6.21. The problem specification and its algorithm for the post-teaching exercise were also explained in Chapter 6, section 6.6.2 in Figure 6.38. Each dissected section is given a suitable descriptive name for later references and evaluations (see Figures 7.72 & 7.73).

```

2. cout<< "Have gate card? ";
3. cin>> gatecard;
4. cout<<"Have ID card? ";
5. cin>> identittyC;
6. If (gatecard == "Y" || identittyC == "Y") ← Pre_outer_if_OR
7. {
8.   cout<<"Have office card? ";
9.   cin>>officecard;
10.  cout<<"Got office door pin code correct? ";
11.  cin>>officepin;
12.   if (officecard == "Y" && officepin == "Y") ← Pre_inner_if_AND
13.     cout<<"Access into the office granted"; ← Pre_inner_if_content
14.   else
15.     cout<<"Access into the office not granted"; ← Pre_inner_else_content
16. }
17. else
18. cout<<" Access into the company not granted"; ← Pre_outer_else_content

```

{the structure of nested-if} - Pre\_nested\_if

Pre\_outer\_if\_content

Figure 7.72: Pre-teaching algorithm sections on decisions

```

1. float yearmark, exam_mark, final mark;
2. cout<<"Enter year mark"<<endl;
3. cin>>yearmark;
4. cout<<" Enter exam mark" <<endl;
5. cin>>exam_mark;
6. finalmark = yearmark + exam_mark;
7. finalmark = finalmark/2;
8. if (exam_mark > 39 && finalmark > 49) ← Post_if_AND
9.   cout<<" You passed"; ← Post_if_content
10. else
11. cout<<"You failed"; ← Post_else_content

```

{Structure of if-else} - Post\_if\_else

Figure 7.73: Post-teaching algorithm sections on decisions

The content of Table 7.2 consists of paired algorithm sections to be used for SOLO-adapted evaluation.

**Table 7.2: Paired algorithm section (decisions/nested decisions)**

Pair	Section name	
Pair 1	Pre_nested_if	Represents the correct structure of the if-statement or nested if-statement
	Post_if_else	
Pair 2	Pre_inner_if_AND	Refers to the use the correct structure of the if-clause that has the “&&” operator or uses alternative method
	Post_if_AND	
Pair 3	Pre_outer_if_content	Refers to the correct content in the block following the <i>if-statement</i> condition when it evaluates to true
	Post_if_content	
Pair 4	Pre_inner_if_content	Refers to the correct content in the block following the <i>if-statement</i> condition when it evaluates to true
	Post_if_content	
Pair 5	Pre_outer_else_content	Refers to the correct content in the <i>else</i> part of the <i>if-statement</i> (the code between the blocks)
	Post_else_content	
Pair 6	Pre_inner_else_content	Refers to the correct content of the <i>else</i> part of the <i>if-statement</i> (the code between the blocks)
	Post_else_content	

Since the algorithm has been dissected accordingly, the following section presents the SOLO-adapted evaluation results for the concept.

### 7.5.2.2 Concept and pairs performance

This concept consists of six pairs, that is, pair 1 to pair 6. The references and what each pair means were described in the previous section (section 7.5.1, Table 7.2). Each pair is evaluated through SOLO-adapted evaluation to locate its state before and after a teaching intervention. Just like in cycle 2, the evaluation sheets are used for each pair on both the control and experimental group. The evaluation sheets per pair are presented in Appendix B. Table 7.3 presents the average percentage of transitions across pair 1 to pair 6 which are the algorithm sections on decisions / nested decisions. The average percentage across pairs gives an overall performance of a concept (decisions / nested decisions).

**Table 7.3: Average transitions performance on *decisions/nested decisions* over all pairs**

P	P→P (Stagnant-no-learning)		U→P (Lower-level-deterioration)		M→P (Intermediate-deterioration+)		R→P (deteriorated++)	
	Con	Exp	Con	Exp	Con	Exp	Con	Exp
	0	0	0	0	0	0	0	0
U	P→U (Lower-level-improvement)		U→U (Stagnant-at-lower-level)		M→U (Intermediate-deterioration)		R→U (deteriorated+)	
	Con	Exp	Con	Exp	Con	Exp	Con	Exp
	0	0	3.3%	0	0	0	0	0
M	P→M (Intermediate-improvement+)		U→M (Intermediate-improvement)		M→M (Stagnant-at-intermediate-level)		R→M (deteriorated)	
	Con	Exp	Con	Exp	Con	Exp	Con	Exp
	0	0	6.7%	7.3%	26.7%	7.6%	17%	7.6%
R	P→R (Improved++)		U→R (Improved+)		M→R (Improved)		R→R (Already-improved)	
	Con	Exp	Con	Exp	Con	Exp	Con	Exp
	0	0	3.3%	4.6%	26.7%	48.5%	13.3%	30.3%
	P		U		M		R	

As described in the methodology (section 4.5), the most desired transitions are M→R (*improved*), U→R (*improved+*) and P→R (*improved++*) because such transitions represent an ultimate improvement. The overall comparison across pair 1 to pair 6 in relation to *decisions/nested decisions* shows more of the experimental group under the improved transition (M→R) than the control group.

The average percentage for M→R transition in this concept for the control group is 26.7% and for the experimental group is 48.5%. The recorded percentage of 48.5% in the experimental group is almost double the average percentage in the control group. This means that the TLPAP in the experimental group has been two times more effective for the concept of *decisions/nested decisions* than the normal teaching method in the control group. However, I hope for the same or even more improvements in the experimental group under the M→R transition in the next cycle of the action research.

The performance of the M→R transition could have been higher if the M→M (*stagnant-at-intermediate-level*) transition did not record 26.7% in the control group. Furthermore, the M→M transition outcome means that the teaching method in the control group may be not enough to push more students into the M→R transition. The

average percentage of the M→M transition recorded 7.6% in the experimental group which is less as compared to the control group. There have been instances in both the control and experimental groups that resulted in the R→M transition, which means deteriorated learning; however, both have small average percentages (17% in the control group which is bit higher than the 7.6% in the experimental group). This transition means students are confused and signals additional adjustment of a teaching method. Since the experimental group has a 7.6% average percentage, the effect does not outdo the positives achieved from other transitions and cannot be the indicator of significant adjustments on the adopted teaching approach or tool. The rest of the transitions recorded less than 10% outcome: their difference cannot have major impact on the method of teaching.

In the case of R→R transition which means already-know, I am not able to control how students are taught or gain knowledge outside the experimental setup of this study. As a result, the R→R transition outcome cannot be linked to the teaching intervention neither in the control nor experimental group. This applies to all R→R transition outcomes across the action research cycles in this study. The other possibility that can cause R→R transition is that the student is not struggling with that pair or concept. As a result of these implications, there is no further analysis on the R→R transition outcomes unless deemed necessary.

In the experimental group, there were pairs that recorded 100% M→R (*improved*) transition, meaning that all sections of the algorithm which were in the multistructural category of a pre-test have transitioned into a relational category of the post-test. If there is no 100% M→R transition, it means the transitions have split into one or more of the undesirable transitions which are M→M (*Stagnant-at-intermediate-level*), M→U (*Intermediate-deterioration*) or M→P (*Intermediate-deterioration+*). Such pairs with 100% M→R transition include pair 1 (Pre\_nested\_if and Post\_if\_else) in Appendix B Table B.3; pair 3 (Pre\_outer\_if\_content and Post\_if\_content) in Appendix B Table B.9; pair 4 (Pre\_inner\_if\_content and Post\_if\_content) in Appendix B Table B.12; pair 5 (Pre\_outer\_else\_content and Post\_else\_content) in Appendix B Table B.15; and pair 6 (Pre\_inner\_else\_content and Post\_else\_content) in Appendix B Table B.18. The only pair that has a split transition in the experimental group is pair 2 (Pre\_inner\_if\_AND and Post\_if\_AND) in Appendix B Table B.6. However, pair 2 has a small percentage (9.1%) in the M→M transition of the experimental group as

compared to the higher percentage M→M transition (40%) in the control group. This means that pair 2 is close to 100% M→R transition.

A 100% transition can also occur under the U→R transition or P→R transition. Just like 100% M→R transition, a 100% U→R transition occurs when all sections of the algorithm are at the unistructural stage in the pre-test have resulted into relational stage in the post-test. If there is no 100% U→R transition, it means that the transitions spilt into one or more undesirable transitions which are U→U (*Stagnant-at-lower-level*), U→M (*Intermediate-improvement*) or U→P (*Lower-level-deterioration*) transitions. Similarly, in the case of a 100% P→R transition, split transition will be P→P (*Stagnant-no-learning*), P→U (*Lower-level-improvement*) or P→M (*Intermediate-improvement+*). The P→R transition has no records because it is unlikely; hence, it means knowing nothing (prestructural) to knowing everything (relational). In this concept, only pair 6 in Appendix B Table B.16 has recorded 100% U→R transition in both the control and experimental groups. It is also worth noting that the U→R transition is denoted by *Improved+* because the improving process jumps one stage up (which is multistructural) to reach relational stage. This means that a relevant teaching method would have been more effective to record the U→R transition.

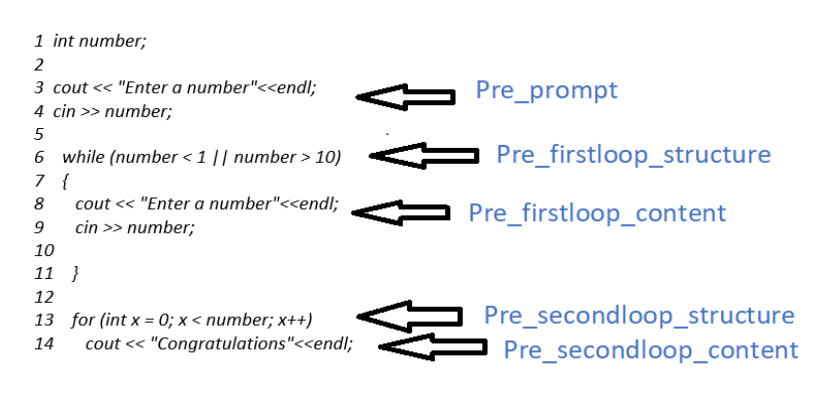
In this concept, the U→M (*intermediate-improvement*) transition has recorded a similar small percentage (6.7% in the control group and 7.3% in the experimental group). This transition is not desirable; however, it is a progress towards the ultimate transition (U→R). The implication of this U→M transition is that the teaching methods are good but not enough. In the case of experimental group, as the percentage is small, the effects do not outdo the overall performance of the TLPAP. There are also records of R→M (*deteriorated*) transition with small average percentage across pair 1 to pair 6. There is a notable record of R→M transition in pair 5 and pair 6 where the control group recorded 30%. The R→M transition in the experimental group recorded 0% in pair 5 and 18.5% in pair 6. The R→M transition implies that the teaching method may be doing more harm than good.

### 7.5.3 Results (Loops)

#### 7.5.3.1 Algorithm sections

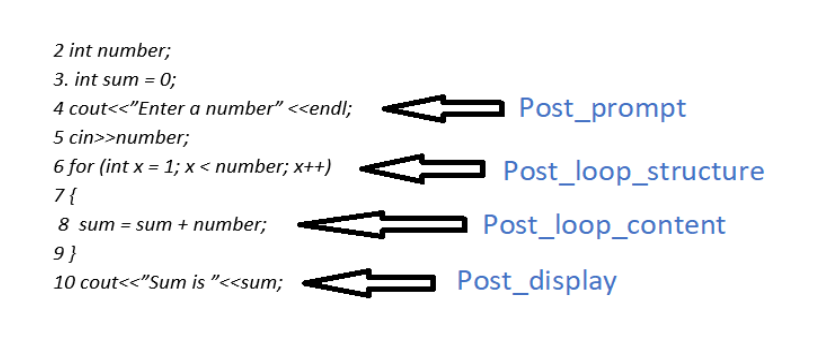
There were 30 SIPS for this experiment, 15 in the control group and 15 in the experimental group. The problem specification for loops on pre-teaching algorithms

was described in Chapter 6, section 6.4.1 in Figure 6.30. The problem specification for loops on post-teaching algorithms was described in Chapter 6, section 6.6.2 in Figure 6.39. The only change in this cycle is the design of the animation and dissected sections for SOLO-adapted evaluations. The following figure (Figure 7.74) indicates the dissected algorithm sections on loops for the pre-teaching algorithm.



**Figure 7.74: Pre-teaching algorithm sections on loops**

Figure 7.75 shows the dissected algorithm sections on loops for the post-teaching algorithm.



**Figure 7.75: Post-teaching algorithm sections on loop**

The content of Table 7.4 consists of a paired section for loops.

**Table 7.4: Paired algorithm sections - loop**

Pair	Section name	
Pair 1	Pre_prompt	Refers to the capability to prompt the values outside the loop
	Post_prompt	
Pair 2	Pre_firstloop_structure	Means the correct structure of the first loop (including the condition).
	Post_loop_structure	
Pair 3	Pre_firstloop_content	Is about the correct content first loop in the pre-teaching algorithm in relation to the loop in the post-teaching algorithm
	Post_loop_content	
Pair 4	Pre_secondloop_structure and Pre_secondloop_content	Means that the output of the program is in the correct place (note that the pre-teaching algorithm display is conditioned by the loop and is part of the display requirement) (see Figure 7.74)
	Post_display	

### 7.5.3.2 Concept and pairs performance

The content of Table 7.5 is the average percentage of transitions across pair 1 to pair 4. The evaluation sheets per pair are presented in Appendix C.

**Table 7.5: Average transitions performance on loops over all pairs**

P	P→P (Stagnant-no-learning)		U→P (Lower-level-deterioration)		M→P (Intermediate-deterioration+)		R→P (deteriorated++)	
	Con	Exp	Con	Exp	Con	Exp	Con	Exp
	0	0	0	0	0	0	0	0
U	P→U (Lower-level-improvement)		U→U (Stagnant-at-lower-level)		M→U (Intermediate-deterioration)		R→U (deteriorated+)	
	Con	Exp	Con	Exp	Con	Exp	Con	Exp
	0	0	1.7%	0	0	0	0	0
M	P→M (Intermediate-improvement+)		U→M (Intermediate-improvement)		M→M (Stagnant-at-intermediate-level)		R→M (deteriorated)	
	Con	Exp	Con	Exp	Con	Exp	Con	Exp
	0	0	18.4%	31.6%	25%	6.7%	1.7%	0
R	P→R (Improved++)		U→R (Improved+)		M→R (Improved)		R→R (Already-improved)	
	Con	Exp	Con	Exp	Con	Exp	Con	Exp
	0	0	1.7%	0	23.3%	41.7%	30%	23.4%
	P		U		M		R	



The M→R (*improved*) transition recorded 41.7% in the experimental and 23.3% in the control group. This M→R transition (for *loops*) is similar to the same transition for the concept of *decisions/nested decisions* (discussed in section 7.5.2.3). This means that the TLPAP in the experimental group has once again been two times more effective than the teaching method in the control group. The other key results which indicate the performance of the teaching method is the M→M (*Stagnant-at-intermediate-level*) transition. The M→M transition resulted in 25% in the control group and 6.7% in the experimental group, providing evidence that teaching methods in the control group need to be improved more than in the experimental group in this concept of loops. The U→M (*Intermediate-improvement*) transition resulted in 18.4% in the control group and 31.6% in the experimental group. The transition means that although progress is apparent, teaching methods may be not enough to result in the U→R transition instead of U→M.

The 100% M→R transition is only found in the experimental group with 60% for pair 4 (Pre\_secondloop\_structure/Pre\_secondloop\_content and Post\_display) as depicted in Appendix C Table C.12. Pair 2 (Pre\_firstloop\_structure and Post\_loop\_structure) in the control group has an M→M transition of 46.7% and the experimental group has 6.7% (see Appendix C, Table C.6). This is the highest recorded result across all pairs in this concept (*loops*) and previous concept (*decisions/nested decisions*). This suggests that the normal teaching method in the control group did not do enough (in pair 2) for teaching the structure of loop as compared to the TLPAP in the experimental group.

## 7.5.4 Results (*nested loops*)

### 7.5.4.1 Algorithm sections

Figure 7.76 indicates the dissected algorithm sections on the *nested loop* for the pre-teaching algorithm.

```

for (int x = 1; x <= 10; x++) ← Pre_outer_loop_structure
{
    for (int y = 0; y < x; y++) ← Pre_inner_loop_structure
    {
        cout << "*" ← Pre_inner_loop_content
    }
    cout << endl; ← Pre_outer_loop_content
}

```

Figure 7.76: Pre-teaching algorithm sections on *nested loops*

Figure 7.77 shows the problem specification for the one-dimensional array.

Write a program to produce the following output: (Use nested loop: Any type of loop)

```

9 8 7 6 5 4 3 2 1
8 7 6 5 4 3 2 1
7 6 5 4 3 2 1
6 5 4 3 2 1
5 4 3 2 1
4 3 2 1
3 2 1
2 1
1

```

Figure 7.77: Problem specification for post-teaching algorithm on *nested loops*

The following figure (Figure 7.78) indicates the dissected algorithm sections on the *nested loop* for the post-teaching algorithm.

```

int num = 9;
for (int x = 0; x < 9; x++) ← Post_outer_loop_structure
{
    for (int y = num; y > 0; y--) ← Post_inner_loop_structure
    {
        cout << y << " " ← Post_inner_loop_content
    }
    num--;
    cout << endl; ← Post_outer_loop_content
}

```

Figure 7.78: Post-teaching algorithm sections on *nested loops*

The content of Table 7.6 consists of a paired section of the *nested loop*.

**Table 7.6: Pairs algorithm sections – nested loops**

Pair	Section name	
Pair 1	Pre_outer_loop_structure	Refers to the correct structure of the outer loop of the <i>nested loop</i>
	Post_outer_loop_structure	
Pair 2	Pre_inner_loop_structure	Requires another loop inside the outer loop with the condition based on the variable of the outer loop or through using an alternative method
	Post_inner_loop_structure	
Pair 3	Pre_inner_loop_content	Refers to the correct content of the inner loop (the code between the blocks)
	Post_inner_loop_content	
Pair 4	Pre_outer_loop_content	Means the correct content of the outer loop (the code between the blocks)
	Post_outer_loop_content	

### 7.5.4.2 Concept and pairs performance

The content of Table 7.7 presents the average percentage of transitions across pair 1 to pair 4. The evaluation sheets per pair are presented in Appendix D.

**Table 7.7: Average transitions performance on *nested loops across pairs***

P	P→P (Stagnant-no-learning)		U→P (Lower-level-deterioration)		M→P (Intermediate-deterioration+)		R→P (deteriorated++)	
	Con	Exp	Con	Exp	Con	Exp	Con	Exp
	0	0	0	0	0	0	0	0
U	P→U (Lower-level-improvement)		U→U (Stagnant-at-lower-level)		M→U (Intermediate-deterioration)		R→U (deteriorated+)	
	Con	Exp	Con	Exp	Con	Exp	Con	Exp
	0	0	0	0	0	0	0	0
M	P→M (Intermediate-improvement+)		U→M (Intermediate-improvement)		M→M (Stagnant-at-intermediate-level)		R→M (deteriorated)	
	Con	Exp	Con	Exp	Con	Exp	Con	Exp
	0	0	15%	25%	43.3%	5%	0	0
R	P→R (Improved++)		U→R (Improved+)		M→R (Improved)		R→R (Already-improved)	
	Con	Exp	Con	Exp	Con	Exp	Con	Exp
	0	0	0	0	20%	43.3%	21.7%	23.4%
	<b>P</b>		<b>U</b>		<b>M</b>		<b>R</b>	

The overall outcome in this concept (*nested loops*) is similar to the concept of loops in all aspects (discussed in section 7.5.3.2) due to the similarity of the concepts in *loops* and *nested loops*.

As discussed in section 7.5.2, the 100% M→R transition means that all students who were in the multistructural category (in the pre-teaching exercises) have transition into relational category (in the post-teaching exercise). There is no 100% M→R transition if some students who were in the multistructural category of the pre-teaching exercise transition into other categories like unistructural, prestructural or remain in the multistructural category of the post-teaching exercise. In this concept of nested loops, the 100% M→R transition is under the pair 2 in the experimental group (Pre\_ouster\_loop\_content and Post\_ouster\_loop\_content) with 40% (see Appendix D, Table D.6 for pair 2). The M→M transition has risen higher (53.3%) in the same pair under the control group. The M→M transition resulted in even higher outcome of 73% in pair 4 (Pre\_ouster\_loop\_content and Post\_ouster\_loop\_content) in the control group while the experimental group has 13.3% (see Appendix D, Table D.12 for pair 4). This suggests that *nested loop* is one of the challenging concepts, difficult for students to construct an accurate nesting loop. In pair 4, the TLPAP has displayed positive effects in the experimental group as the M→R transition resulted at 60% compared to 13.3% for the control group.

### 7.5.5 Results (one-dimensional array)

There were 20 SIPS participating in the experiments, that is 10 in the control group and 10 in the experimental group.

#### 7.5.5.1 Algorithm sections (one dimensional array)

The following figure (Figure 7.79) indicates the dissected algorithm sections on one-dimensional array for the post-teaching algorithm.

```

2 const int arrsize = 6;
3 int arrNumbers[arrsize] = {4,10,72,5,90,3}; ← Pre_array_initialised
4 int count = 0;
5 for (int x = 0; x < arrsize; x++) ← Pre_loop_structure
6 {
7 if(arrNumbers[x] > 49) ← Pre_if_condition
8   count++; ← Pre_if_content
9 }
10 cout<<"Total numbers more than 49 is "<<count<<endl; ← Pre_display

```

**Figure 7.79: Pre-teaching algorithm sections on arrays**

Figure 7.80 shows the problem specification for the one-dimensional array.

*Declare a one-dimensional array and name it arrNumbers. Initialise the arrNumbers array with these numbers: 26, 25, 4, 52, 14, 60, 78, 4. Count the numbers which are odd within an array.*

*Sample output:*

*The odd number: 1*

**Figure 7.80: Problem specification for post-teaching algorithm on one-dimensional array**

The following figure (Figure 7.81) indicates the dissected algorithm sections on one-dimensional array for the post-teaching algorithm.

```
2 const int arrsize = 8;
3 int arrNumbers[arrsize] = { 26, 25, 4, 52, 14, 60, 78, 4 }; ← Post_array_initialised
4 int count = 0;
5
6 for (int x = 0; x < arrsize; x++) ← Post_loop_structure
7 {
8   if (arrNumbers[x] % 2 != 0) ← Post_if_condition
9     count++; ← Post_if_content
10}
11 cout << "Total odd numbers: " << count << endl; ← Post_display
```

**Figure 7.81: Post-teaching algorithm sections on one-dimensional array**

The content of Table 7.8 consists of paired sections for the one-dimensional array.

**Table 7.8: Paired algorithm sections – one-dimensional array**

Pair	Section name	
Pair 1	Pre_array_initialised	Is about the capability to initialise an array
	Post_array_initialised	
Pair 2	Pre_loop_structure	Refers to the structure of the loop which is controlled by the length of the array.
	Post_loop_structure	
Pair 3	Pre_if_condition	Refers to the condition of the <i>if-statement</i> within the loop, which uses an array
	Post_if_condition	
Pair 4	Pre_if_content	Refers to the correct content of the if-statement (the code between the blocks)
	Post_if_content	
Pair 5	Pre_display	Refers to the correct display at the correct place
	Post_display	

### 7.5.5.2 Concept and pairs performance

Table 7.9 presents the average percentage of transitions across pair 1 to pair 5. The evaluation sheets per pair are presented in Appendix E.

**Table 7.9: Average transitions performance on one-dimensional arrays for all pairs**

P	P→P (Stagnant-no-learning)		U→P (Lower-level-deterioration)		M→P (Intermediate-deterioration+)		R→P (deteriorated++)	
	Con	Exp	Con	Exp	Con	Exp	Con	Exp
	0	0	0	0	0	0	0	0
U	P→U (Lower-level-improvement)		U→U (Stagnant-at-lower-level)		M→U (Intermediate-deterioration)		R→U (deteriorated+)	
	Con	Exp	Con	Exp	Con	Exp	Con	Exp
	0	0	0	0	0	0	0	0
M	P→M (Intermediate-improvement+)		U→M (Intermediate-improvement)		M→M (Stagnant-at-intermediate-level)		R→M (deteriorated)	
	Con	Exp	Con	Exp	Con	Exp	Con	Exp
	0	0	12%	10%	12%	2%	0	0
R	P→R (Improved++)		U→R (Improved+)		M→R (Improved)		R→R (Already-improved)	
	Con	Exp	Con	Exp	Con	Exp	Con	Exp
	0	0	0	0	36%	66%	40%	20%
	P		U		M		R	

In the concept of arrays, the M→R (*improved*) transition resulted in 36% for the control group and 66% for the experimental group. The M→R performance in the

experimental group is two-thirds outcome as compared to almost one-third results in the control group. The  $M \rightarrow M$  (*Stagnant-at-intermediate-level*) transition has 12% in the control group and 2% in the experimental group. The difference between the two groups on  $M \rightarrow M$  transition is minimal, as with the  $U \rightarrow M$  transition. The  $M \rightarrow R$  transition is the only major indicator of the TLPAP performance for this concept of one-dimensional arrays.

The  $R \rightarrow R$  (*Already-know*) transition has 40% in the control group and 20% in the experimental group. As emphasised in the earlier section, the  $R \rightarrow R$  transition has no direct implications on the intervention (teaching method) in the control or experimental group. This means there will be little commentary or analysis on the  $R \rightarrow R$  transition result.

In the concept of arrays, the pairs that resulted in 100%  $M \rightarrow R$  transition are pair 1 (Pre\_array\_initialised and Post\_array\_initialised) in Appendix E Table E.3; pair 2 (Pre\_loop\_structure and Post\_loop\_structure) in Appendix E Table E.6; pair 3 (Pre\_if\_condition and Post\_if\_condition) in Appendix E Table E.9; and pair 5 (Pre\_display and Post\_display) in Appendix E Table E.15. The only transition that did not result in 100%  $M \rightarrow R$  transition in the experimental group is pair 4 (Pre\_if\_content and Post\_if\_content) in Appendix E Table E.12. In pair 4, the control group resulted in 100%  $M \rightarrow R$  transition (with 30%) while the experimental group resulted in 70%, which is higher. The TLPAP in the experimental group is once again credited due its capacity to produce more 100%  $M \rightarrow R$  transitions than the control group.

## 7.5.6 Results (parallel arrays)

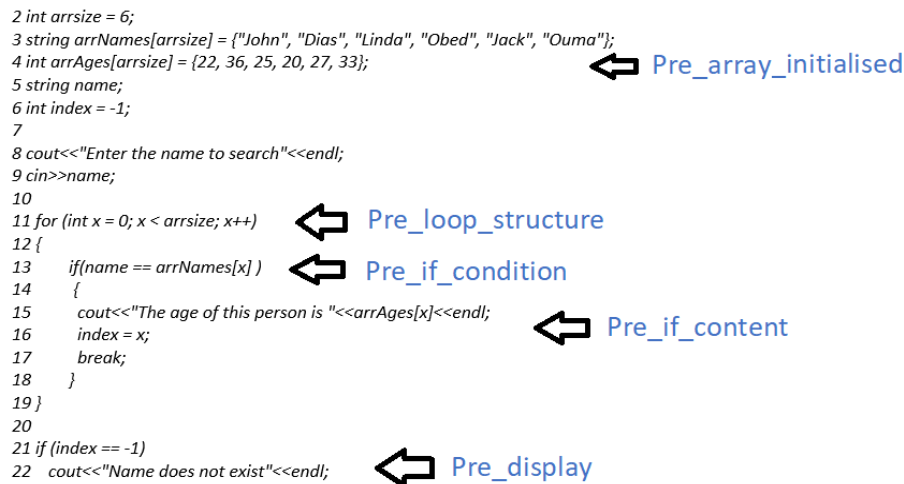
### 7.5.6.1 Algorithm sections (parallel dimensional array)

Figure 7.82 presents the dissected algorithm sections on parallel arrays for the pre-teaching algorithm.

```

2 int arrsize = 6;
3 string arrNames[arrsize] = {"John", "Dias", "Linda", "Obed", "Jack", "Ouma"};
4 int arrAges[arrsize] = {22, 36, 25, 20, 27, 33};
5 string name;
6 int index = -1;
7
8 cout<<"Enter the name to search"<<endl;
9 cin>>name;
10
11 for (int x = 0; x < arrsize; x++)
12 {
13     if(name == arrNames[x])
14     {
15         cout<<"The age of this person is "<<arrAges[x]<<endl;
16         index = x;
17         break;
18     }
19 }
20
21 if (index == -1)
22     cout<<"Name does not exist"<<endl;

```



**Figure 7.82: Pre-teaching algorithm sections on parallel arrays**

Figure 7.83 displays the problem specifications for the post-teaching algorithm.

*Declare an array named arrProducts and that will contain product names. Initialise arrProducts with the product names Apple, Pear, Guava, Banana, Mango and Peach. Declare another array called arrPrices(parallel array to arrProducts), that will contain the prices of arrProducts. Intialise arrPrices with these prices R5, R6, R7, R4, R8, R15.*

- Determine the average price
- Determine and display the name of the most expensive product

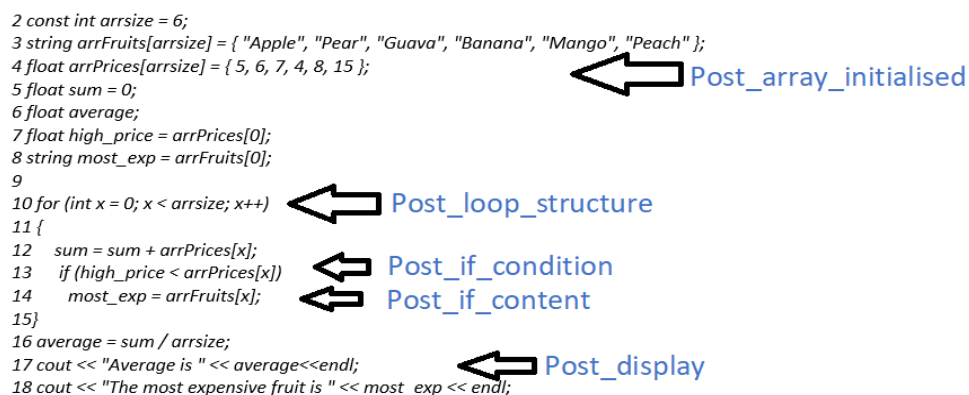
**Figure 7.83: Problem specification for post-teaching algorithm on parallel arrays**

The following figure (Figure 7.84) displays the problem specifications for the post-teaching algorithm on parallel arrays.

```

2 const int arrsize = 6;
3 string arrFruits[arrsize] = {"Apple", "Pear", "Guava", "Banana", "Mango", "Peach"};
4 float arrPrices[arrsize] = { 5, 6, 7, 4, 8, 15 };
5 float sum = 0;
6 float average;
7 float high_price = arrPrices[0];
8 string most_exp = arrFruits[0];
9
10 for (int x = 0; x < arrsize; x++)
11 {
12     sum = sum + arrPrices[x];
13     if (high_price < arrPrices[x])
14         most_exp = arrFruits[x];
15 }
16 average = sum / arrsize;
17 cout << "Average is " << average<<endl;
18 cout << "The most expensive fruit is " << most_exp << endl;

```



**Figure 7.84: Post-teaching algorithm sections on parallel arrays**



Note that the algorithm sections for arrays are the same as for parallel arrays (Table 7.8); however, the interpretation for parallel arrays applies.

### 7.5.6.2 Concept and pairs performance

Table 7.10 presents the average percentage of transitions across pair 1 to pair 5. The evaluation sheets per pair are presented in Appendix F.

**Table 7.10: Transitions performance on parallel arrays**

P	P→P (Stagnant-no-learning)		U→P (Lower-level-deterioration)		M→P (Intermediate-deterioration+)		R→P (deteriorated++)	
	Con	Exp	Con	Exp	Con	Exp	Con	Exp
	0	0	0	0	0	0	0	0
U	P→U (Lower-level-improvement)		U→U (Stagnant-at-lower-level)		M→U (Intermediate-deterioration)		R→U (deteriorated+)	
	Con	Exp	Con	Exp	Con	Exp	Con	Exp
	0	0	0	0	0	0	0	0
M	P→M (Intermediate-improvement+)		U→M (Intermediate-improvement)		M→M (Stagnant-at-intermediate-level)		R→M (deteriorated)	
	Con	Exp	Con	Exp	Con	Exp	Con	Exp
	0	0	10%	12%	16%	0	0	0
R	P→R (Improved++)		U→R (Improved+)		M→R (Improved)		R→R (Already-improved)	
	Con	Exp	Con	Exp	Con	Exp	Con	Exp
	0	0	0	8%	36%	62%	38%	18%
	P		U		M		R	

The performance of the M→R (*Improved*) and M→M (*Stagnant-at-intermediate-level*) transitions in the concept of parallel arrays is similar to the performance of the arrays (presented in the previous section in Table 7.9). The M→R transition continues to double the impact in the experimental group which further credits the capacity of TLPAP in the over-the-normal teaching method.

One of the key performance outcomes in the concept of parallel arrays in the experimental group is its records of 100% M→R transition in all pairs: pair 1 (Pre\_array\_initialised and Post\_array\_initialised) in Appendix F Table F.3; pair 2 (Pre\_loop\_structure and Post\_loop\_structure) in Appendix F Table F.6; pair 3 (Pre\_if\_condition and Post\_if\_condition) in Appendix F Table F.9; pair 4 (Pre\_if\_content and Post\_if\_content) in Appendix F Table F.12; and pair 5 (Pre\_display and Post\_display) in Appendix F Table F.15. The TLPAP in the

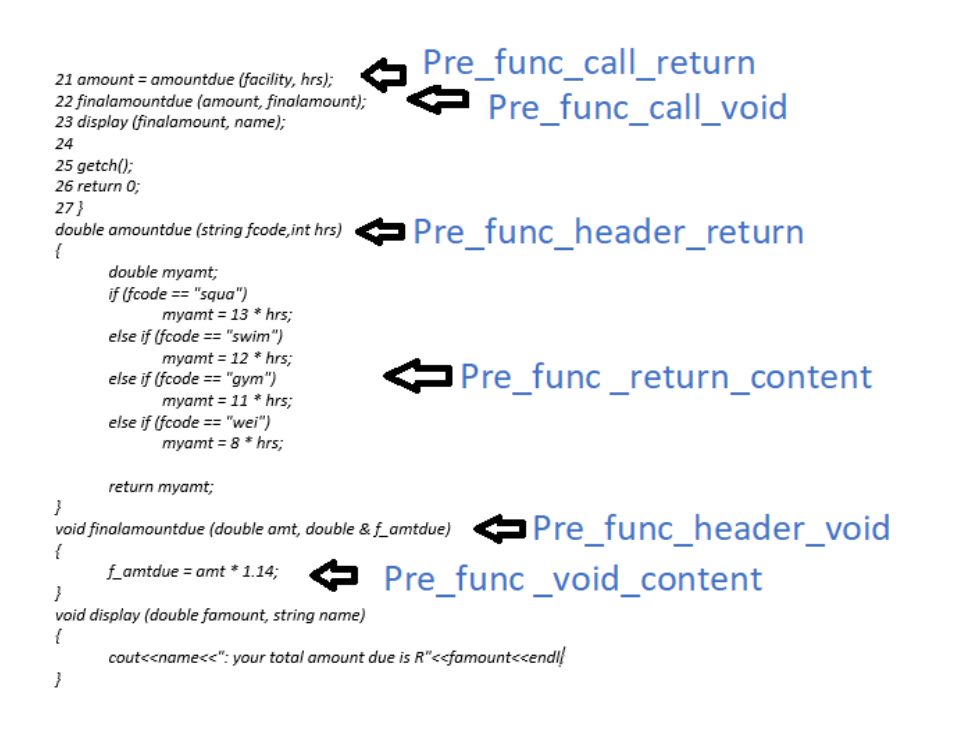
experiment continues to show its capacity through 100% transition outcome in all given pre-teaching sections from the multistructural stage into a relational stage.

### 7.5.7 Results (functions)

There were 32 SIPS participating in this concept, that is 17 in the control group and 15 in the experimental group.

#### 7.5.7.1 Algorithm sections

The following figure (Figure 7.85) indicates the dissected algorithm sections on *functions* for the pre-teaching algorithm.



**Figure 7.85: Pre-teaching algorithm sections on functions**

Figure 7.86 shows the problem specifications for the post-teaching algorithm.

[No global variables must be declared for this program]

People who want to hire a machine to wash their carpets usually go to the *WishyWhishy* company. The company has the three types of machines – type A, B or C for heavy duty, ordinary and light washes respectively. The prices for using the machines are as follows:

Type of machine	Initial cost	Additional cost per hour
A	R50.00	R30.00
B	R62.50	R36.25
C	R74.87	R40.50

The amount must be calculated and returned in a function called **calcAmount** by adding the initial cost to the calculated cost for the time used.

Create another function called **calcTotalAmount** of a type **void** that receives an amount calculated by **calcAmount** as parameter, then calculate vat (14%) and final total amount due. Note that both vat and amount due calculated by **calcTotalAmount** must be accessible to the calling environment.

On the main function, prompt the user to enter the type of machine and time machine used (in minutes). Call all necessary functions implemented to display an amount before tax, a tax amount and an amount after tax.

For more clarity see sample output below.

```
Enter machine type
A
Enter minutes consumed while using the machine
32
I n v o i c e
Amount R1010
Vat R141.4
Total amount due R1151.4
-

Enter machine type
B
Enter minutes consumed while using the machine
50
I n v o i c e
Amount R1875
Vat R262.5
Total amount due R2137.5
-
```

Figure 7.86: Problem specification for post-teaching algorithm on functions

Figure 7.87 displays the dissected algorithm sections on *functions* for the post-teaching algorithm.

```

15 amount = calcAmount(type, minutes);
16 calcTotalAmount(amount, cvat, totalAmount);
17 cout<<"\n! n v o i c e \n";
18 cout<<"\n";
19 cout<<"Amount R "<<amount<<endl;
20 cout<<"Vat R"<<cvat<<endl;
21 cout<<"\n";
22 cout<<"Total amount due R"<<totalAmount<<endl;
23
24 getch();
25 return 0;
26 }
double calcAmount(string tp, int min)
{
    double amount;
    if (tp == "A")
        amount = 50 + (30 * min);
    else
        if (tp == "B")
            amount = 62.50 + (36.25 * min);
        else
            if (tp == "C")
                amount = 74.87 + (40.50 * min);
    return amount;
}
void calcTotalAmount(double amt, double& vat, double& totAmt)
{
    vat = amt * 0.14;
    totAmt = amt + vat;
}

```

← Post\_func\_call\_return  
 ← Post\_func\_call\_void  
 ← Post\_func\_header\_return  
 ← Post\_func\_return\_content  
 ← Post\_func\_header\_void  
 ← Post\_func\_void\_content

**Figure 7.87: Post-teaching algorithm sections on functions**

The content of Table 7.11 consists of paired sections for *functions*.

**Table 7.11: Pairs sections/columns names for functions**

Pair	Sections/columns name	Description
Pair 1	Pre_func_call_return	Refers to the capability to call a <i>function</i> that returns a value
	Post_func_call_return	
Pair 2	Pre_func_call_void	Refers to the capability to call a <i>function</i> that does not return a value ( <i>function</i> of a type <i>void</i> )
	Post_func_call_void	
Pair 3	Pre_func_header_return	Refers to the capability to construct a <i>function</i> header that returns a value with a relevant return type
	Post_func_header_return	
Pair 4	Pre_func_return_content	Refers to the capability to construct the correct content in a <i>function</i> that returns a value
	Post_func_return_content	
Pair 5	Pre_func_header_void	Refers to the capability to construct a <i>function</i> header of a type <i>void</i> that does not return a value
	Post_func_header_void	
Pair 6	Pre_func_void_content	Refers to the capability to provide the correct content in a <i>function</i> that does not return a value
	Post_func_void_content	

### 7.5.7.2 Concept and pairs performance

Table 7.12 presents the average percentage of transitions across pair 1 to pair 6 for functions. The evaluation sheets per pair are presented in Appendix G.

**Table 7.12: Average transitions performance on functions for all pairs**

P	P→P (Stagnant-no-learning)		U→P (Lower-level-deterioration)		M→P (Intermediate-deterioration+)		R→P (deteriorated++)	
	Con	Exp	Con	Exp	Con	Exp	Con	Exp
	0	0	0	0	0	0	0	0
U	P→U (Lower-level-improvement)		U→U (Stagnant-at-lower-level)		M→U (Intermediate-deterioration)		R→U (deteriorated+)	
	Con	Exp	Con	Exp	Con	Exp	Con	Exp
	0	0	3	0	0	0	0	0
M	P→M (Intermediate-improvement+)		U→M (Intermediate-improvement)		M→M (Stagnant-at-intermediate-level)		R→M (deteriorated)	
	Con	Exp	Con	Exp	Con	Exp	Con	Exp
	0	0	18.8%	27.3%	29.4%	4.5%	4%	0
R	P→R (Improved++)		U→R (Improved+)		M→R (Improved)		R→R (Already-improved)	
	Con	Exp	Con	Exp	Con	Exp	Con	Exp
	0	0	0	4.5%	26.5%	62.2%	12.8%	0
	P		U		M		R	

In the concept of *functions*, the M→R (*improved*) transition resulted in 26.5% in the control group and 62.2% in the experimental group. The average percentage of M→R transition in the experimental group is almost a triple what the control group has achieved. Such achievement is once again credited to the capacity of TLPAP. The teaching method in the experimental group (the TLPAP group) has been positively consistent from the first concept to this concept of the experiments. The consistency was primarily indicated by M→R and M→M transitions in both the control and experimental groups.

The M→M (*stagnant-at-intermediate-level*) has continued to be higher in the control group than in the experimental group. As emphasised in the previous sections, the average percentage in the M→M transition in the control group could have been added in the M→R group. This shows that the normal teaching method in the control group is not sufficient to achieve such results. In the experimental group, the M→M transition average percentage has been less than 10% while the M→R transition recorded higher percentages.

The 100% M→R transition in the concept of *functions* are pair 1 (Pre\_func\_call\_return and Post\_func\_call\_return) in Appendix G Table G.3; pair 3 (Pre\_func\_header\_return and Post\_func\_header\_return) in Appendix G Table G.9; and pair 6

(Pre\_func\_void\_content and Post\_func\_void\_content) in Appendix G Table G.18. Pair 2 (Pre\_func\_call\_void and Post\_func\_call\_void) in Appendix G Table G.6 and pair 4 (Pre\_func\_return\_content and Post\_func\_return\_content) in Appendix G Table G.12 did not result in 100% the M→R transition in the experimental group. However, the M→R transition resulted in 73.3% in the experimental group while the control group had 11.8%. In pair 1, there is a notable 23.5% R→M (*deteriorated*) transition in the control group with 0% in comparison to the experimental group. The R→M transition means learning is not improving nor stagnant, but rather in reverse mode from relational to multiscrptual. Pedagogically, this suggests the teaching method requires improvement in the control group due to deteriorating learning.

## 7.6 Cycle 4 reflections

The following reflection emanates from this cycle:

### **Uses of assignment**

- Generally, students in the experimental group performed better than the control group. The emphasis on initialisation, copying of values from one variable to another and overwriting are fundamental skills which must not be overlooked, especially by SIPS. In this cycle, the TLPAP proved to be impactful on SIPS, which is similar to the results reported in cycle 2 (Chapter 6, section 6.6.1).

### **Decisions/nested decisions**

- In the pre-test evaluations of pair 1 to pair 6, there is a notable high number of algorithm sections under the multistructural category in both control and experimental groups. When the algorithm sections are under multistructural category in the pre-test stage, that further confirms that the students are SIPS.
- In the previous cycle when nested-decider was demonstrated, after the demonstration I received a request from students to repeat the animation process again. However, in this cycle when I demonstrated the same nested-decider, I did not receive any request to repeat the animation program. The animation program was repeated based on the number of times stated in the pedagogical guidelines (section 7.4). It is possible that the SIPS are already familiar with the concepts being taught due to the impact of the introductory

animation programs which are demonstrated before the animation program for problem specifications.

- The post-teaching exercise of a nested-decider may be less challenging than the pre-teaching exercise. Changes to the post-teaching exercise could be part of cycle 4 planning.

### **Loops/nested loops**

- Overall, the TLPAP had a greater effect on SIPS in the experimental group as compared to the control group on both the loop and nested-loop.
- I received a suggestion from two students during the question-and-answer session that the rendering of the animation in the program may be too quick for them. I should consider slowing the programs a bit, especially the animation for nested-loop. The plan to slow the pace of the animation rendering was noted and implemented as part of cycle 4 plans. However, I was able to make adjustments on program code to lower the animation rendering on the remaining concepts (*arrays* and *functions*) for this current cycle.
- I have also learned from the algorithm exercises that some few SIPS could not understand when to have a program that requires only a loop or only a *decision* statement or a combination of both. This could be because our animation program did not present a sample combination of a loop and *decisions*. Even though I received positive results on TLPAP in relation to loops and decisions, a plan to design a new animation program that properly requires both loops and *decisions* statements within one algorithm is noted and may be part of cycle 4 planning.

### **One-dimensional and parallel arrays**

- One aspect of one-dimensional arrays that was found missing is an animation that updates elements in the array. The missing aspect is noted and may be incorporated into animation programs for introducing a new concept or animation program with problem specification within the next cycle of this action research.
- As prior knowledge of *decisions* and loops play a critical role in the understanding of arrays and subsequently parallel arrays, if some SIPS still struggle with those concepts, it is unlikely they will succeed with the use of arrays simply because arrays need a loop to populate it with values and further



need the knowledge of *decisions* to filter on conditions or access relevant values in the array. A plan for an additional animation program that requires both loops and *decisions* statements as stated previously is noted.

- Generally, the TLPAP has resulted in more improvements for parallel arrays in the experimental group than in the control group, as is evident in the M→R transition outcome. The concept of parallel arrays resulted in 100% M→R transition for all pairs in the experimental group, signifying the impact of TLPAP as compared to normal, traditional teaching methods.

### Functions

- The overall picture behind teaching and learning *functions* seems more challenging than other concepts. The algorithm sections, where the proper use of reference parameters and passing of arguments were required, have had poor results in the control group compared to the experimental group. Once again, the TLPAP shows consistent improvement, especially in teaching and learning challenging concepts like passing values by references. SIPS in the experimental group were able to clearly see what exactly happens when a reference is passed through these animations.
- The decreased pace of rendering of animation program and narration as per pedagogical guidelines added value toward the success of teaching and learning of this concept. The decreased pace also allowed a more relaxed narration during the rendering of the animation program.

### General view on the cycle

- The *improved* (M→R) pairs in the sections of the algorithm were found mostly in the experimental group and *stagnant-at-lower-level* (M→M) were more in the control group. These two transitions were the main difference between the interventions in the control and experimental groups.
- The *deteriorated* (R→M) pairs in the sections of the algorithm were found mostly in the control group were minimal.
- The *improved+* (U→R) pairs in the sections of the algorithm were found mostly in the experimental group but were minimal.

- Based on the pre-teaching algorithm exercises from both the control and experimental group, most misconceptions are found within the concepts of *nested decisions*, *nested loops*, reference parameters and parallel arrays. Pedagogical guidelines as part of TLPAP were followed when demonstrating the animation programs with no foreseeable revision at this stage.

## 7.7 Planning for cycle 4

The intention was to plan for cycle 4 for further confirmation of the reliability and efficacy of the TLPAP. The following pointers are planned as part of experimenting with cycle 4 with an improved approach.

- Verify that all animation programs comply with the 12 principles of multimedia learning by Mayer, as stated in the methodology section (Chapter 4, section 4.4.1). Apply Mayer's principles of multimedia learning in all areas of the animation programs where required. One change that was done during this cycle was to edit a program code to reduce the speed of the animation program which already complies to one of the recommended principles of multimedia learning. However, the overall solution to be applied in all animation programs is to have a *radio* button where the educator can set the pace of the animation rendering as normal, slow and very slow.
- In the case of a nested-decider, the post-teaching exercise is planned to be adjusted to have a complexity similar to the pre-teaching exercise. Even though this is not a big problem since the pre-teaching exercise is used for teaching, a new problem for the post-teaching exercise is planned. This can further work in our favour to show that the SIPS who have undergone learning through TLPAP can solve any other type of problem thereafter.
- An additional animation program that should be capable of combining a *decision-statement* and a *loop* is planned. This is to give students an additional perspective during teaching and learning on how *decision* statements work with loops.
- Incorporating an animation program for introducing a concept (arrays) that emphasises the update of existing array elements.

- Pedagogical guidelines revised in this cycle (cycle 3) serve as the latest version and are intended to be used as they are in cycle 4. However, I may take a flexible route to adjust the guidelines if deemed necessary at that moment in time.
- The chapter for cycle 4 will only report on the demonstration of the animation programs that were affected by the improvements.

## 7.8 Conclusion

In this chapter, I presented the adapted TLPAP framework where various animation programs for IPC were designed and tested. In addition to the animation programs designed and experimented with previously in cycle 2, there are added animation programs developed and experimented in this cycle (cycle 3). Such animation programs were designed for the concept of loops, *nested loops*, arrays and *functions*. The pedagogical guidelines used in cycle 2 were revised and used along with the animation programs as part of TLPAP.

The pre-teaching and post-teaching algorithms were dissected into sections for further evaluation and analysis. The analysis of students' algorithms was undertaken through SOLO-adapted evaluations. The evaluation outcome indicated that students in the experimental group outperformed students in the control group. This shows the positive impact of TLPAP, which was used in the experimental group, on teaching and learning IPC.

### **State of the framework thus far**

At this stage in our action research, the state of the TLPAP framework can be referred to as **adapted TLPAP framework version 3** pending the outcome of cycle 4 results that could suggest more adaptation for the final framework. The specific items that form part of this adapted TLPAP framework consist of the following:

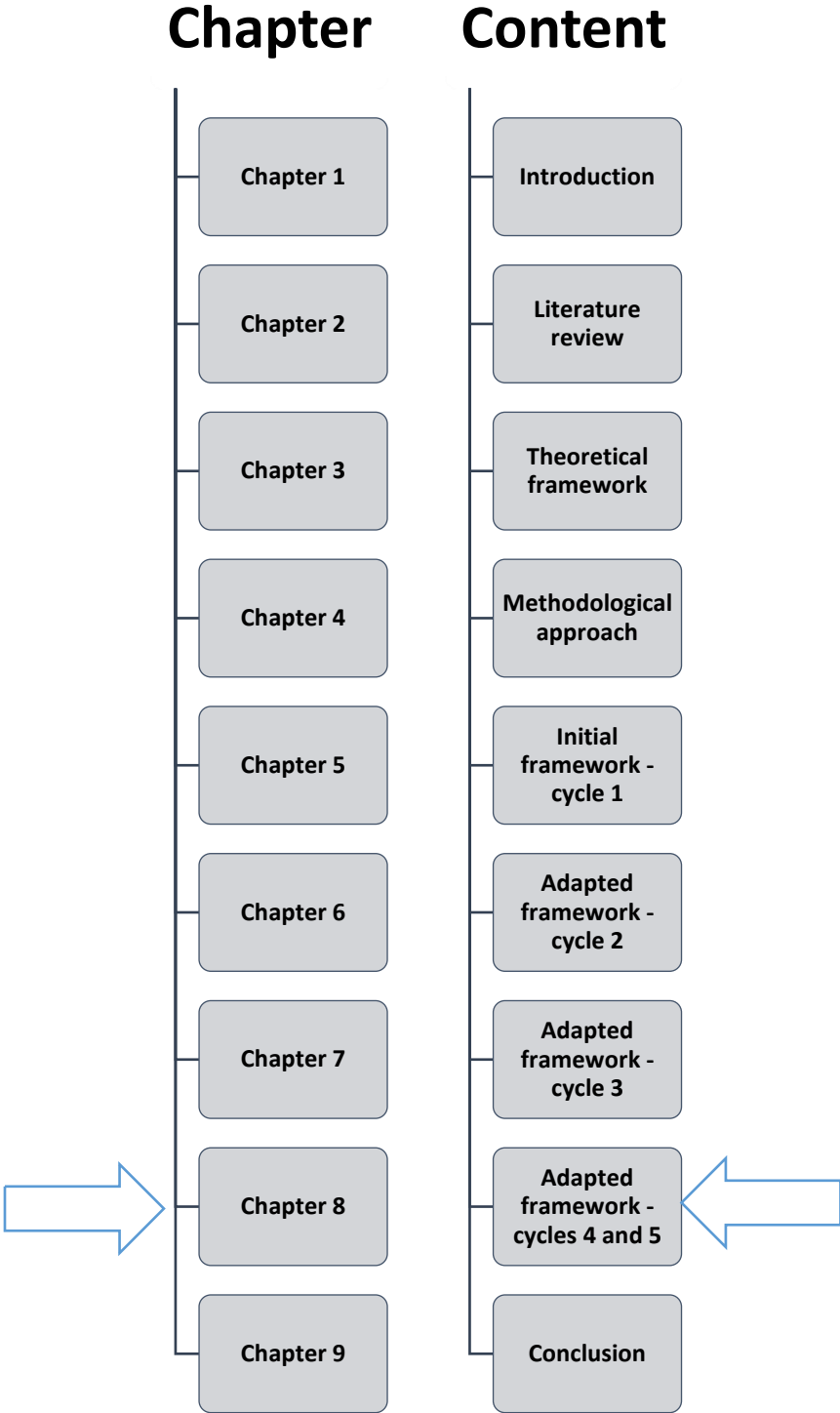
- Animation programs focusing on *assignment* (presented in cycle 2, Chapter 6) as opposed to the manipulatives version (presented in cycle 1, Chapter 5). Even though the use of physical manipulatives yielded positive results, emphasising the uses of *assignment*, the rationale behind this choice is the animation's flexibility to accommodate large classes in a contact teaching

session and its compatibility in an online session. Furthermore, in cycle 3, I repeated the animation program on *assignment* and still had positive findings.

- Another item that is incorporated into this adapted framework is nested-decider animation program presented in cycle 2. The experiments in cycle 2 are positive and the repeat experiments with the same animation program presented in this cycle (cycle 3) are even more positive. However, the experiments in cycle 3 were coupled with an animation program for introducing a concept; therefore, such animations are part of the final framework. The inclusion of animation programs for introducing a concept into the final framework apply to the remaining concepts which are loops, arrays and *functions*.
- The animation programs for loops presented in cycle 2 are not part of the tentative final framework because there were persistent errors like incorrect sentinel, misplaced loop content, incorrect overall looping structure which were found in the experimental group upon presenting the animation programs. The revised animation program for loops presented in this cycle forms part of the final framework.
- The animation programs for arrays and *functions* presented in this cycle (cycle 3) are also incorporated into the adapted framework.
- Finally, the adapted pedagogical guidelines revised in this cycle (cycle 3) are also part of the adapted TLPAP framework pending the outcome of cycle 4.

The subsequent chapter presents the outcome of cycles 4 and 5.

# Chapter 8: Adapted framework - Cycles 4 and 5



## 8.1 Introduction

The idea behind this study, the plan, is to investigate, test and adapt appropriate teaching learning methods relevant for Struggling Introductory Programming Students (SIPS). The aim is to use the adopted method to improve SIPS understanding of IPC and further develop a relevant framework. By this stage, I have already established that the use of animations programs carries the opportunity to advance teaching and learning of SIPS. As a result, the thesis statement has been posed as, “*A Teaching and Learning Programming with Animation Programs (TLPAP) framework, can improve Struggling Introductory Programming Students (SIPS) comprehension of Introductory Programming Concepts (IPC)*”.

To reach the final conclusion on the TLPAP, I embarked on focused action research to be done in several action cycles. The previous chapter (Chapter 7) presented the activities and the outcomes of cycle 3. The activities in cycle 3 were based on the animation programs for teaching and learning IPC (*assignment, decisions/nested decisions, loops, nested loops, one-dimensional arrays, parallel arrays and functions*). The content of cycle 4 includes an additional animation program which is a *combination of loops and decisions*. The other changes implemented in cycle 4 are based on the reflections of cycle 3. Since cycle 5 has minimal adjustment, I am presenting its brief conclusive outcome at the end of this chapter and the actual graphs with exact data are presented in Appendix M. The process of developing and testing various animation programs and the adaption of the pedagogical guidelines are ways to continually advance and validate the teaching and learning programming with the animation programs (TLPAP) framework.

This cycle serves as a continuation to answer our last research question, formulated as follows:

- How can we prove and evaluate the efficacy of teaching and learning programming with the animation programs framework?

The layout of the chapter starts by presenting the additional animation programs in section 8.2 and the experimental results in section 8.3. Section 8.4 is the summary of the reflections on cycle 4 and section 8.5 is the summary of reflections on cycle 5.

Section 8.6 discuss transitions across cycle 3 to cycle 5 and section 8.7 concludes the chapter.

## 8.2 Additional animation programs

### 8.2.1 Combination of decisions and loops

The need for an additional animation program that includes a program code with *loops* and *decision statements* was identified in cycle 3. The design of this additional animation program adhered to all other recommendations stated in cycle 3 and cycle 4, such as compliance with Mayer’s principles of multimedia learning. This additional animation program is based on the problem specification in Figure 8.1.

*You are requested to develop a program that counts the number of males and females from a group of people (there are 6 people). For each person prompt for gender (input in the form of a letter). Letter “M” for male or “F” for female. The inputs are not case sensitive, meaning that a small letter m for male or a small letter f for female should work as well. If the user enters any other letter, show this message “Invalid character, please re-enter the gender”.*

*The program should display the number of males and females in this format (sample output):*  
Total males 2  
Total females 4

**Figure 8.1: Problem specification – decisions and loops**

Figure 8.2 shows the appearance of the animation program which combines *decisions* and *loops*. The borderline rectangle indicates a piece of code executed inside the loop. The steps of execution are the same as previously explained for similar animation programs (nested-decider) in cycle 2. The animation speed can be adjusted to normal, slow or very slow.

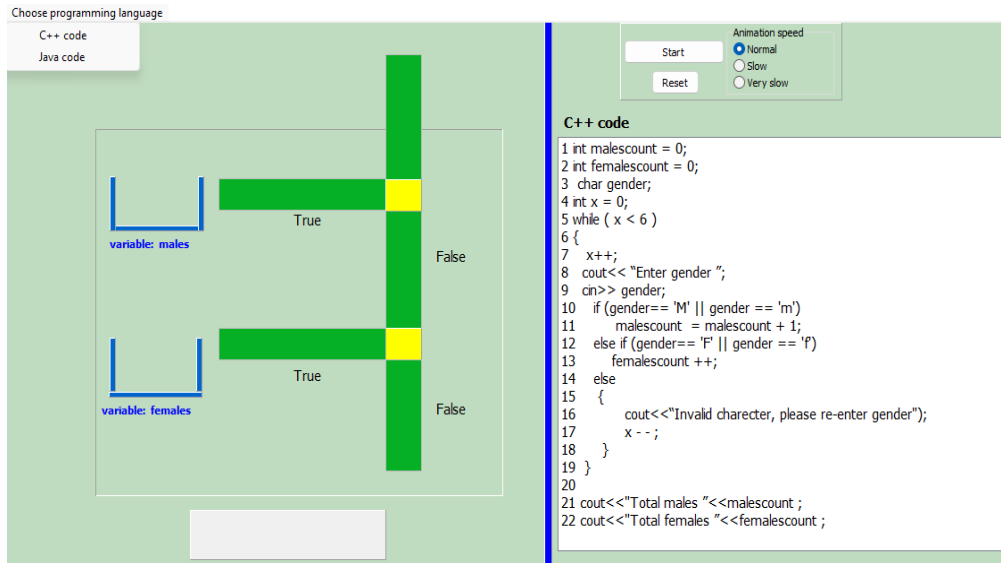


Figure 8.2: Decisions and loops (Demo 1)

Note that when the blue highlight indicating that a statement is executed in the program code side passed line 1 and line 2, the zeros appeared in the two containers on the animation side for the variable males and females (See Figure 8.3). The prompt to enter a gender happens at line 8, and in this case, an “x” was entered.

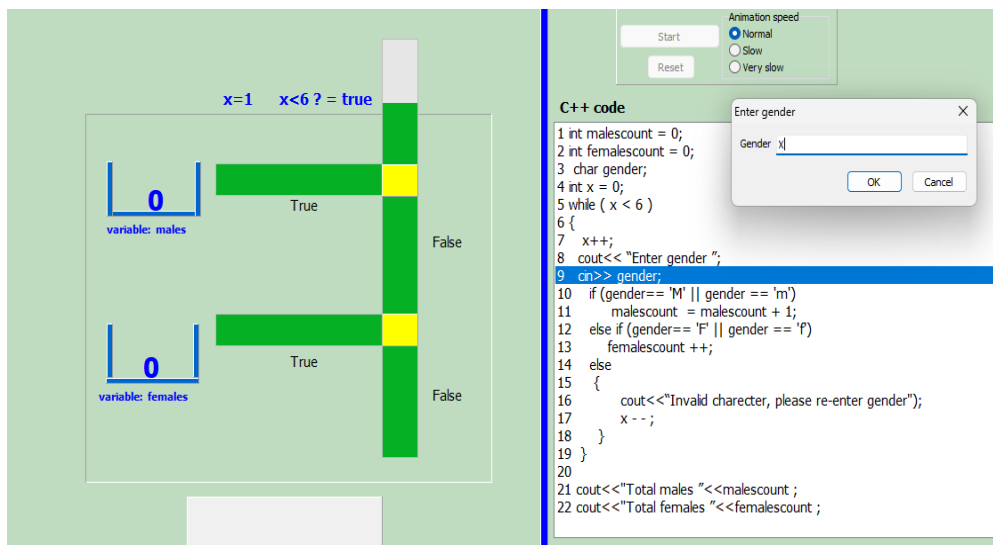


Figure 8.3: Decisions and loops (Demo 2)

Since the value “x” falls under invalid characters, the second else part of the program is executed. However, the key moment is when SIPS have to see the process of advancing towards the last *else* part on the animation side when all the conditions are evaluated first (See Figure 8.4).



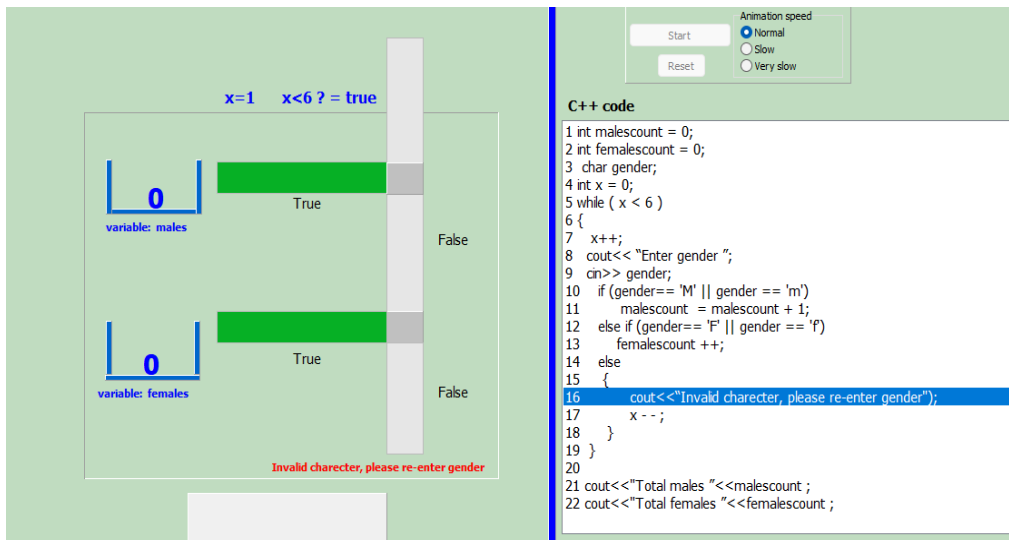


Figure 8.4: Decisions and loops (Demo 3)

Figure 8.5 shows when the correct gender (“M”) has been entered, the variable *males* is incremented by 1. The graphic “+1” floats from top to bottom. When it reaches where “0” is (container labelled *variable: males*), the 0 is replaced by 1 which indicates that it has been added.

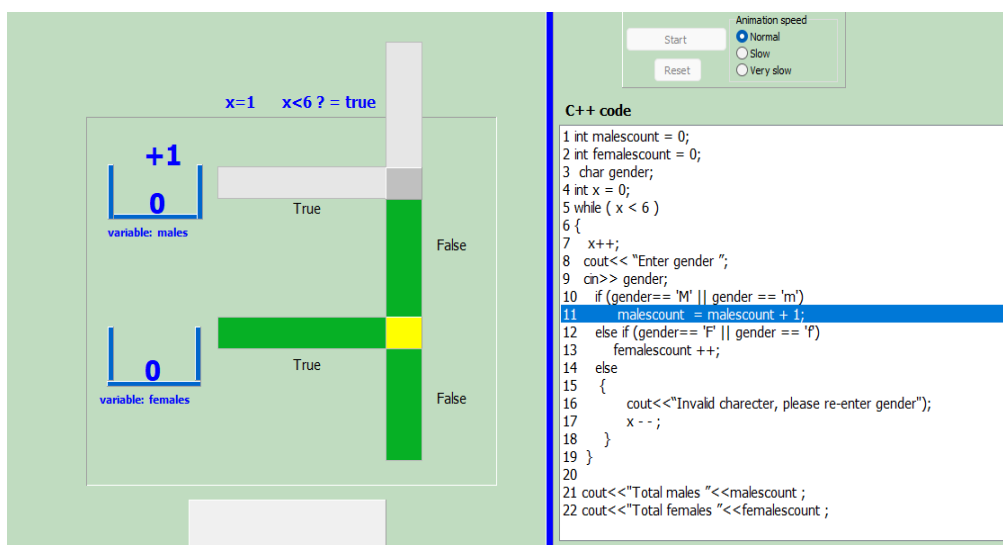


Figure 8.5: Decisions and loops (Demo 4)

Figure 8.6 shows when the correct gender (“F”) has been entered and the variable *females* is incremented by 1. As explained in the previous paragraph, the graphic “+1” floats from top to bottom as a sign of incrementing.

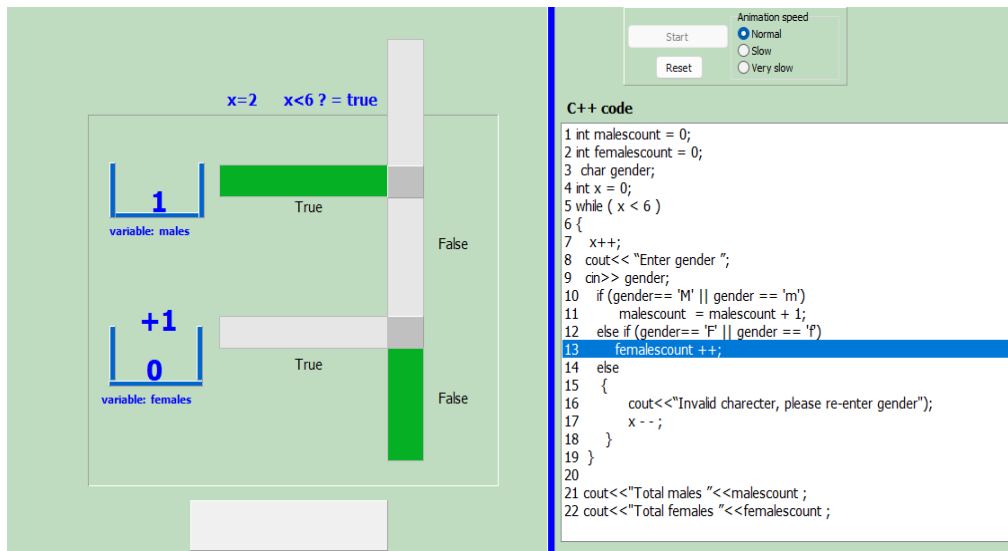


Figure 8.6: Decisions and loops (Demo 5)

Figure 8.7 displays the result of the completed animation process. This means that at this stage the character “M” or “m” was entered twice and the character “F” or “f” was entered four times.

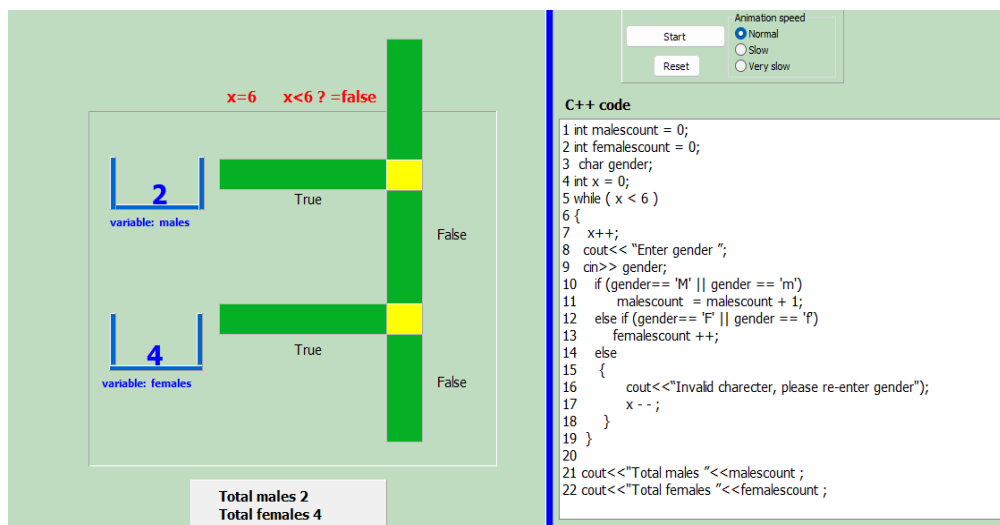


Figure 8.7: Decisions and loops (Demo 6)

### 8.2.2 Additional animation on concept introduction for arrays

The initial ILOs for the concept of one-dimensional arrays (as stated in Chapter 7, section 7.2.3) are for SIPS to write an algorithm that store and access values in a one-dimensional array. Based on the reflection of cycle 3, I have noted that the introductory animation or animation with program specification does not depict to SIPS how the array can be updated. The following animation program will be an add-on (with

emphasis on updating an element in the array) to the introductory animation programs for introducing arrays.

The program in Figure 8.8 consists of a code and corresponding graphics. The blue graphics (tray-like) on the left side represent an array.

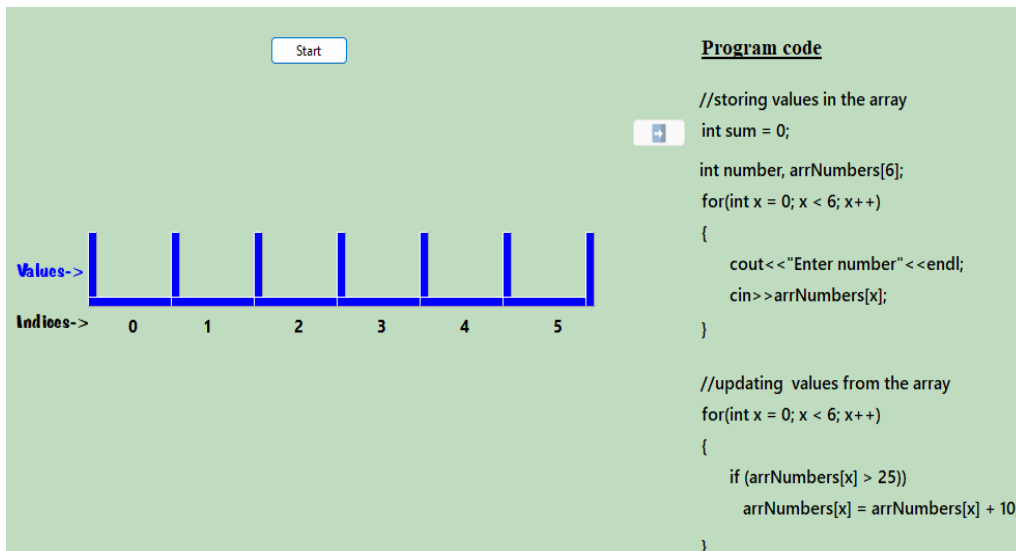


Figure 8.8: Arrays (Demo 1)

When the arrow reaches the *cin* object, as usual it prompts the user to enter the value. In this case, each value is stored in the array as entered (see Figure 8.9).

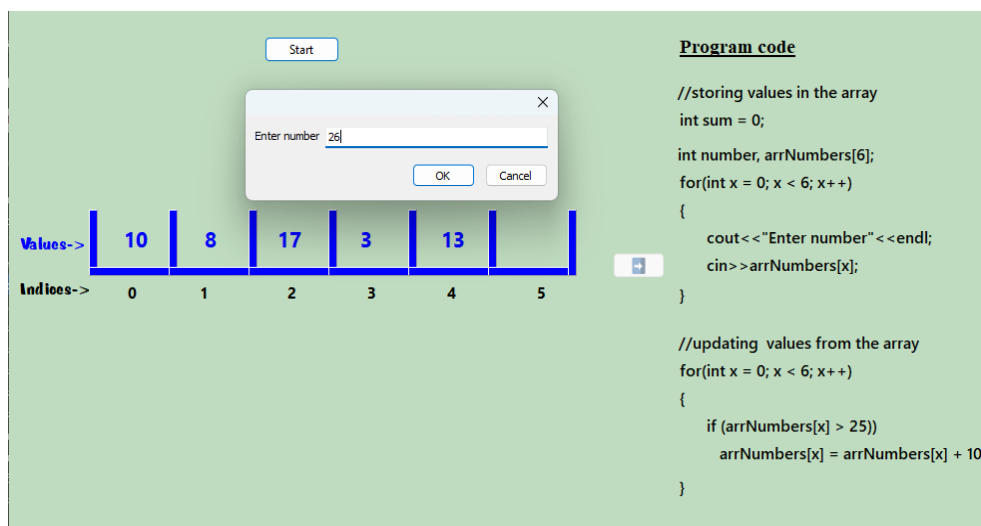
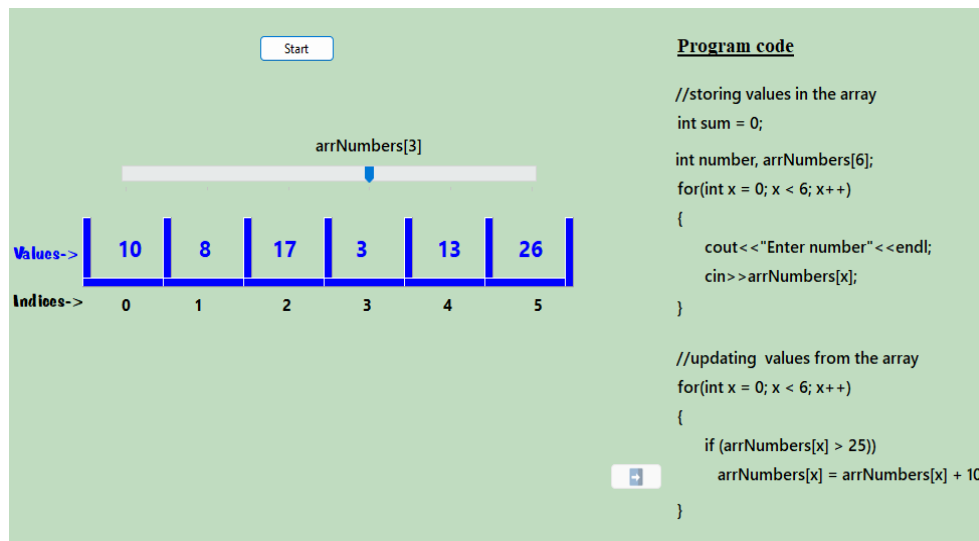


Figure 8.9: Arrays (Demo 2)

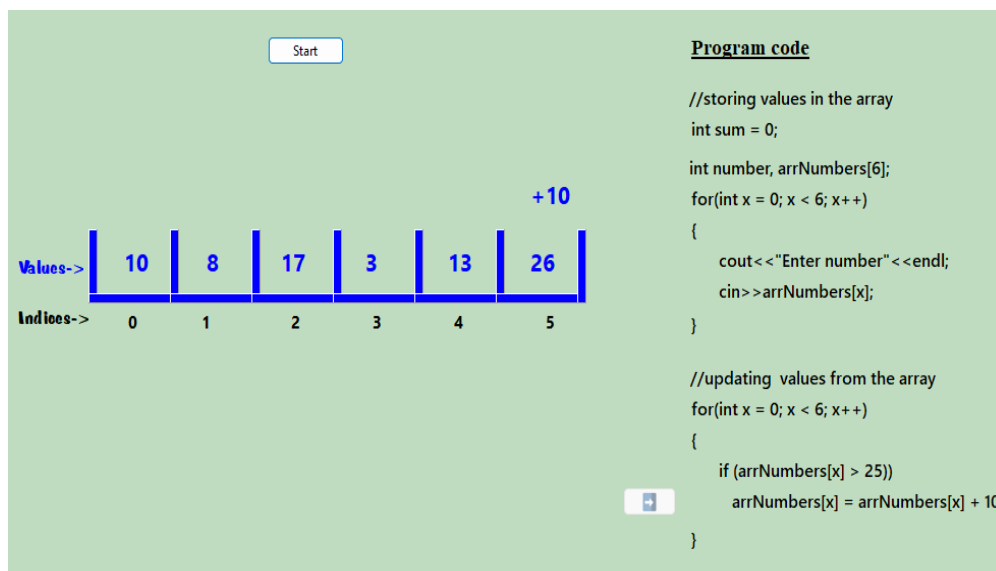
When the first loop is complete, the second loop takes over. There is an animating track bar that points to the currently accessed element in the array. SIPS are able to

see the index of that currently accessed element. In the case of Figure 8.10, each element is accessed to check if is greater than 25.



**Figure 8.10: Arrays (Demo 3)**

The last element in the array is greater than 25, so it gets incremented by 10. The number 10 will float from the top into the last container of the last tray (See Figure 8.11).



**Figure 8.11: Arrays (Demo 4)**

The state of Figure 8.12 is a complete executed animation program; we can see that the last element is now updated.

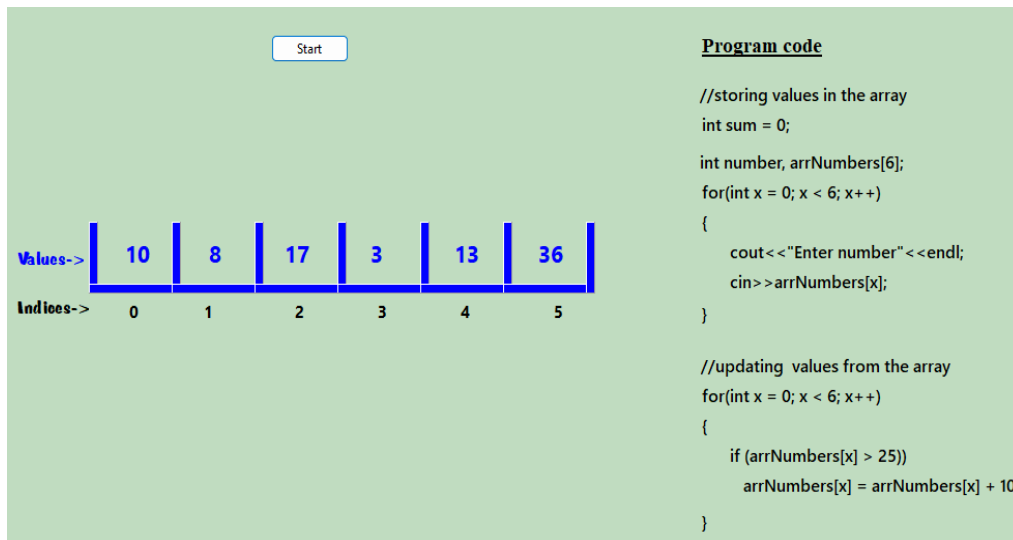


Figure 8.12: Arrays (Demo 5)

## 8.3 Experimentation results and analysis

This section reports on the experimental results in cycle 4 in 2022 semester 1.

### 8.3.1 Results (assignment)

The animation programs for *assignment* discussed in cycle 2 (Chapter 6, section 6.2) produced the results in Table 8.1. The table (Table 8.1) reports on the control and experimental group results for both pre-teaching and post-teaching exercises on the uses of *assignment*. There were 23 students in attendance. The students were divided randomly and as equally as possible between two groups. The control group had 11 students and the experimental group had 12. In the first column (control group) under pre-teaching exercise 1 there is  $x(8) - 72.7\%$ . In an  $x(8) - 72.7\%$ , the number 8 in the brackets means the number of SIPS who got the value of  $x$  correct; the subsequent number represents the same number as a percentage. All other columns have been calculated in the same manner.

**Table 8.1: Results (Assignment)**

Control group		Experimental group	
Pre-teaching exercises	Post-teaching exercise	Pre-teaching exercises	Post-teaching exercises
<b>Pre-teaching exercise 1</b> x (8) – 72.7%	<b>Post-teaching exercise 1</b> t (6) – 54.5% p (5) – 45.5% k (5) – 45.5%	<b>Pre-teaching exercise 1</b> x (7) – 58.3%	<b>Post-teaching exercise 1</b> t (10) – 83.3% p (9) – 75.0% k (9) – 75.0%
<b>Pre-teaching exercise 2</b> p (6) – 54.5% y (7) – 72.7%	<b>Post-teaching exercise 2</b> t (6) – 54.5% p (8) – 72.7% k (6) – 54.5%	<b>Pre-teaching exercise 2</b> p (7) – 58.3% y (8) – 66.7%	<b>Post-teaching exercise 2</b> t (11) – 91.7% p (12) – 100% k (10) – 83.3%
<b>Pre-teaching exercise 3</b> k (6) – 54.5% t (8) – 72.7%		<b>Pre-teaching exercise 3</b> k (6) – 50.0% t (8) – 66.7%	
<b>Pre-teaching exercise 4</b> x (7) – 63.6% y (4) – 36.4%		<b>Pre-teaching exercise 4</b> x (6) – 50.0% y (7) – 58.3%	

The experimental group shows a significant number of students who got the answers correct in the post-teaching exercise (see percentage on the value of *t*, *p* and *k* in Table 8.1). When SIPS get the values of *t*, *p* and *k* in the post-exercise correct, the assumption is that the students now know the uses of *assignment* in initialisation and copying (overwriting). This is because the correct answer includes steps which build on comprehension of initialisation and overwriting. The results of the post-teaching exercise in the experimental group are higher than the pre-teaching exercise in the same experimental group and as compared to the post-teaching exercise in the control group. The main aspects of this experiment were to instil a proper understanding of how *assignment* works in programming.

### 8.3.2 Results (decisions/nested decisions)

There were 25 students participating in this study for this concept, that is 12 students in the control group and 13 students in the experimental group. The experimental group has an extra student because the total number of students is odd.

#### 8.3.2.1 Algorithm sections

The problem specification and its algorithm for the pre-teaching exercise are the same, as explained in Chapter 6, section 6.3, Figure 6.21. The algorithm section for the pre-teaching exercise remains the same as in Chapter 7, section 7.5.2.1, Figure 7.72. The post-teaching algorithm has been changed for cycle 4. In cycle 3 reflections, it was

noted that the post-teaching algorithm missed an additional pair that can be related to the pre-teaching exercise. The problem specification for the new post-teaching exercise is in Figure 8.13.

Write an algorithm that determines a final mark, then check if the student has passed, has a condoned pass, has qualified for a supplementary exam or has failed. Prompt for a year mark and exam mark. Calculate the final mark as the average of the year mark and exam mark.

Condition for a pass:  
If the final mark is 50 or higher, then show the message "Pass".

Condition for a condoned pass:  
If the final mark is either 48 or 49, then reset the final mark to 50 and show the message "Condoned pass".

Condition to qualify for the supplementary exam:  
If the final mark is between 44 and 47, then show the message "You qualify for supplementary exam".

If any of the above conditions are not true, show the message "Failed".

**Figure 8.13: Problem specification for post-teaching algorithm on decisions/nested decisions**

The content of Figure 8.14 is the algorithm sections for the post-teaching exercise.

```

6 finalmark = yearmark + exam_mark;
7 finalmark = finalmark/2;
8
9 if ( finalmark >= 50)
10  cout<<" Pass";
11 else if (finalmark == 48 || finalmark == 49)
12 {
13  finalmark = 50;
14  cout<<" Condoned pass";
15}
16 else if (finalmark >=44 && finalmark < 48)
17  cout<<"You qualify for supplementary exam";
18 else
19  cout<<"You failed";

```

**Figure 8 14: Post-teaching algorithm sections on decisions/nested decisions**

The content of Table 8.2 consists of paired algorithm sections that are to be used for SOLO-adapted evaluation.

**Table 8.2: Paired algorithm section (decisions/nested decisions)**

Pair	Section name	
Pair 1	Pre_nested_if	Represents the correct structure of the nested if-statement
	Post_nested_if	
Pair 2	Pre_inner_if_AND	Refer to using the correct structure of the if-clause that has the “&&” operator or uses alternative method
	Post_if_AND	
Pair 3	Pre_outer_if_OR	Refer to using the correct structure of the if-clause that has the “  ” operator or uses alternative method
	Post_if_OR	
Pair 4	Pre_inner_if_content	Refers to the correct content in the block following the <i>if-statement</i> when it evaluates to true
	Post_if_OR_content	
Pair 5	Pre_outer_else_content	Is about the correct content in the <i>else</i> part of the <i>if-statement</i>
	Post_else_content	
Pair 6	Pre_inner_else_content	Pair 6 refers to the correct content of the <i>else</i> part of the <i>if-statement</i>
	Post_else_content	

### 8.3.2.2 Concept and pairs performance

The content of Table 8.3 is the average percentage of transitions across pair 1 to pair 6. The content of Table 8.3 presents the overall performance of *decisions/nested decisions*. The evaluation sheets per pair are presented in Appendix H. In the previous action cycles, the main indicators of TLPAP’s effect on SIPS were mainly revealed in two transitions, that is, the M→R (*improved*) and M→M (*Stagnant-at-intermediate-level*).



**Table 8.3: Average transitions performance on decisions/nested decisions for all pairs**

P	P→P (Stagnant-no-learning)		U→P (Lower-level-deterioration)		M→P (Intermediate-deterioration+)		R→P (deteriorated++)				
	Con	Exp	Con	Exp	Con	Exp	Con	Exp			
	0	0	0	0	0	0	0	0			
U	P→U (Lower-level-improvement)		U→U (Stagnant-at-lower-level)		M→U (Intermediate-deterioration)		R→U (deteriorated+)				
	Con	Exp	Con	Exp	Con	Exp	Con	Exp			
	0	0	8.3%	0	0	0	0	0			
M	P→M (Intermediate-improvement+)		U→M (Intermediate-improvement)		M→M (Stagnant-at-intermediate-level)		R→M (deteriorated)				
	Con	Exp	Con	Exp	Con	Exp	Con	Exp			
	0	0	18.1%	21.8%	23.6%	9%	2.8%	0			
R	P→R (Improved++)		U→R (Improved+)		M→R (Improved)		R→R (Already-improved)				
	Con	Exp	Con	Exp	Con	Exp	Con	Exp			
	0	0	1.4%	2.6%	30.6%	62.8%	15.3%	5.1%			
<b>P</b>			<b>U</b>			<b>M</b>			<b>R</b>		

Most of the time, in the previous cycle, the M→R transition was higher in the experimental group while the M→M transition was lower in the control group. In this concept of *decisions/nested decision*, the transition outcomes look consistent to cycle 3. For example, in cycle 3 (Chapter 7, section 7.5.2) under the concept of *decisions/nested decisions*, the M→R transition in the control group resulted in an average percentage (26.7%) which is half the average percentage (48.5%) in the experimental group. In this cycle, the same concept results in the M→R transition of 30.6% in the control group and 62.8% in the experimental group. This means that the effect of TLPAP was two times greater than what the normal teaching method achieved.

In the concept of *decisions/nested decision*, the M→M transition continues to be higher in the control group, while the experimental group resulted in 0% or less than 10% within the same transition. As emphasised previously, the M→M transition result should have been in the M→R transition. This means that the higher percentage in the M→M transition is evidence that a relevant teaching method needs further improvements.

There are a few pairs that stand out or have a major influence on the overall performance in understanding a concept. Such pairs include transition outcomes that are much higher than the others or pairs that resulted in 100% transitions. The 100%

M→R transition is found in the experimental group under pair 2 (Pre\_inner\_if\_AND and Post\_if\_AND) and in the control group under pair 3 (Pre\_outer\_if\_OR and Post\_if\_OR). Pair 2 is in Appendix H Table H.6, and Pair 3 is in Appendix H Table H.9. However, in pair 3, the 100% M→R transition in the control group has 25% while the non-100% M→R transition in the experimental group has 69.2%. There is a notable 30.8% in the experimental group under the U→M transition (*intermediate-improvement*) in pair 4 (Pre\_inner\_if\_content and Post\_if\_OR\_content) and pair 5 (Pre\_outer\_else\_content and Post\_else\_content). Pair 4 is in Appendix H Table H.12 and Pair 5 in Appendix H Table H.15. The U→M transition is acceptable since it falls within the category of improving transitions, but as emphasised in the previous cycle, the more desirable transition instead of U→M is U→R (*Improved+*).

The overall comparison across pair 1 to pair 6 in relation to *decisions/nested decisions* shows more of the experimental group under the *improved* category than the control group. The improvements in the experimental group are positive compared to the control group, but also compared to the previous cycles under the concept of *decisions/nested decisions*.

### 8.3.3 Results (loops/decisions)

There were 28 SIPS in participation and they were randomly divided into 14 per group.

#### 8.3.3.1 Algorithm sections

The problem specification in Figure 8.15 is the problem specification for the post-teaching exercise.

You are requested to develop a program that computes students' marks. Prompt the user to enter the exam mark of each student one by one (there are four students in total). If the user enters an exam mark lower than 0 or higher than 100, re-prompt the user to enter the exam mark again.

Calculate and display the average exam mark of all four students.  
Determine and display the number of students who got an exam mark of 75 or higher.

**Figure 8.15: Problem specification for post-teaching algorithm on loops/decisions**

The following figure (Figure 8.16) indicates the dissected algorithm sections on *loops/decisions* for the pre-teaching algorithm.

```

1 int malescount = 0;
2 int femalescount = 0;
3 char gender;
4 int x = 0;
5 while (x < 6)
6 {
7     x++;
8     cout<< "Enter gender ";
9     cin>> gender;
10    if (gender== 'M' || gender == 'm')
11        malescount = malescount + 1;
12    else if (gender== 'F' || gender == 'f')
13        femalescount ++;
14    else
15    {
16        cout<<"Invalid charecter, please re-ente genderr");
17        x --;
18    }
19 }
20
21 cout<<"Total males "<<malescount ;
22 cout<<"Total females "<<femalescount ;

```

Diagram annotations for Figure 8.16:

- Pre\_loop: Arrow pointing to line 5.
- Pre\_loop\_content: Arrow pointing to lines 7-9.
- Pre\_if\_OR: Arrow pointing to line 10.
- Pre\_if\_OR\_content: Arrow pointing to line 11.
- Pre\_if\_OR\_2: Arrow pointing to line 12.
- Pre\_if\_OR\_2\_content: Arrow pointing to line 13.
- Pre\_else: Arrow pointing to line 16.
- Pre\_display: Arrow pointing to line 21.

Figure 8.16: Pre-teaching algorithm sections on loops/decisions

Figure 8.17 displays the dissected algorithm sections on loops/*decisions* for the post-teaching algorithm.

```

1 double exam_mark;
2 double total = 0;
3 double distinction = 0;
4 int x = 0;
5 while (x < 4)
6 {
7     x++;
8     cout<<("Enter an exam mark");
9     cin>>exam_mark;
10
11    if(exam_mark < 0 || exam_mark > 100)
12    {
13        cout<<"Please enter exam mark ranging between 0 and 100"<<endl;
14        x--;
15    }
16    else
17    {
18        total = total + exam_mark;
19        if(exam_mark >= 75)
20            distinction = distinction + 1;
21    }
22 }
23
24 cout<<"Average exam mark is "<<exam_mark;
25 cout<<"Total students with exam mark of 75 or higher is "<< distinction;

```

Diagram annotations for Figure 8.17:

- Post\_loop: Arrow pointing to line 5.
- Post\_loop\_content: Arrow pointing to lines 7-9.
- Post\_if\_OR: Arrow pointing to line 11.
- Post\_if\_OR\_content: Arrow pointing to line 13.
- Post\_else: Arrow pointing to line 18.
- Post\_display: Arrow pointing to line 24.

Figure 8.17: Post-teaching algorithm sections on loops/decisions

The content of Table 8.4 consists of a paired section of the nested loop.

**Table 8.4: Pairs algorithm sections – nested loops**

Pair	Section name	
Pair 1	Pre_loop	Refers to the correct structure of the loop
	Post_loop	
Pair 2	Pre_loop_content	Means the general content of the first loop
	Post_loop_content	
Pair 3	Pre_if_OR	Means the correct format of the condition in an <i>if-statement</i> using the    operator
	Post_if_OR	
Pair 4	Pre_if_OR_content	Refers to the correct content of the if-statement
	Post_if_OR_content	
Pair 5	Pre_if_OR_2	Means the correct format of the condition in the second if statement using the    operator
	Post_if_OR	
Pair 6	Pre_if_OR_2_content	Refers to the correct content of the if-statement
	Post_if_OR_content	
Pair 7	Pre_else	Means the correct content of the <i>else</i> part of the if-statement
	Post_else	
Pair 8	Pre_display	Means correct display at the right place
	Post_display	

### 8.3.3.2 Concept and pairs performance

The content of Table 8.5 is the average percentage of transitions across pair 1 to pair 8. The table shows the overall performance of the *loops/decisions*. The problem specification required a solution that needed both a looping statement and decision statement. The evaluation sheets per pair are presented in Appendix I.

**Table 8.5: Average transitions performance on decisions/loops for all pairs**

P	P→P (Stagnant-no-learning)		U→P (Lower-level-deterioration)		M→P (Intermediate-deterioration+)		R→P (deteriorated++)				
	Con	Exp	Con	Exp	Con	Exp	Con	Exp			
	0	0	0	0	0	0	0	0			
U	P→U (Lower-level-improvement)		U→U (Stagnant-at-lower-level)		M→U (Intermediate-deterioration)		R→U (deteriorated+)				
	Con	Exp	Con	Exp	Con	Exp	Con	Exp			
	0	0	3.6%	0	0	0	0	0			
M	P→M (Intermediate-improvement+)		U→M (Intermediate-improvement)		M→M (Stagnant-at-intermediate-level)		R→M (deteriorated)				
	Con	Exp	Con	Exp	Con	Exp	Con	Exp			
	0	0	20.5%	20.5%	20.5%	2.7%	8%	0			
R	P→R (Improved++)		U→R (Improved+)		M→R (Improved)		R→R (Already-improved)				
	Con	Exp	Con	Exp	Con	Exp	Con	Exp			
	0	0	2.7%	4.5%	33.2%	64.9%	10.1%	7.1%			
P			U			M			R		

An algorithm that requires a combination of loop and decisions was experimented for the first time in this cycle. The overall transition performance is similar to the previous concept (*decisions/nested decisions*) and also to the previous cycles. This further affirms the consistency of TLPAP which performs in a similar pattern regardless of new problem specifications or repeat experiments. For example, the M→R transition still revolves around 30% in the control group and around 60% in the experimental group, just as in previous experiments. Furthermore, the M→M transition continues to retain higher percentages in the control group than the experimental group.

The interesting results are in pair 1 (Pre\_loop and Post\_loop) and pair 2 (Pre\_loop\_content and Post\_loop\_content). Pair 1 is in Appendix H Table H.3 and Pair 2 is in Appendix H Table H.6. In pair 1, the M→R transition in the control was close (42.9%) to the experimental group which resulted in 57.1%. Pair 1 refers to the correct structure of the loops. This means almost half of both the control and experimental groups were able to get the overall structure of the loop correct.

In pair 2, which refers to the correct content of the loop (the code between the blocks), the U→M transition resulted in 42.9% in the control group and 14.2% in the experimental group. This means that the method of teaching for the control group was good for intermediate improvement (U→M transition) specifically for pair 2, but not good enough for U→R (*Improved+*) transition.

The pairs that resulted in 100% M→R transition are pair 1 (Pre\_loop and Post\_loop), pair 2 (Pre\_loop\_content and Post\_loop\_content), pair 4 (Pre\_if\_OR\_content and Post\_if\_OR\_content) in Appendix H Table H.12; pair 6 (Pre\_if\_OR\_2\_content and Post\_if\_OR\_content) in Appendix H Table H.18; and pair 8 (Pre\_display and Post\_display) in Appendix H Table H.24. Overall, the 100% M→R transition continues to complement the effect of the adopted teaching method experimental group.

### ***8.3.4 Results (parallel arrays)***

The total number of students is odd (23), that is 11 students were in the control group and 12 in the experimental group.

#### ***8.3.4.1 Algorithm sections***

The problem specification for parallel arrays remains the same as indicated in Chapter 7, section 7.3.3.2 in Figure 7.55 (pre-teaching exercise) and section 7.5.6.1 in Figure 7.83 (post-teaching exercise). The algorithm pairs remain the same as in Chapter 7, section 7.5.5.1 in Table 7.8. The algorithm section also remains the same for the pre-teaching exercise (Chapter 7, section 7.5.6.1 in Figure 7.82) and the post-teaching exercise (Chapter 7, section 7.5.6.1 in Figure 7.84).

#### ***8.3.4.2 Concept and pairs performance***

The content of Table 8.6 is the average percentage of transitions across pair 1 to pair 6. The content of Table 8.6 presents the overall performance of the parallel arrays. The problem specification required a solution that needed both looping statements and a decision statement. Evaluation sheets per pair are presented in Appendix J.

**Table 8.6: Transitions performance on parallel arrays**

P	P→P (Stagnant-no-learning)		U→P (Lower-level-deterioration)		M→P (Intermediate-deterioration+)		R→P (deteriorated++)	
	Con	Exp	Con	Exp	Con	Exp	Con	Exp
	0	0	0	0	0	0	0	0
U	P→U (Lower-level-improvement)		U→U (Stagnant-at-lower-level)		M→U (Intermediate-deterioration)		R→U (deteriorated+)	
	Con	Exp	Con	Exp	Con	Exp	Con	Exp
	0	0	0	0	0	0	0	0
M	P→M (Intermediate-improvement+)		U→M (Intermediate-improvement)		M→M (Stagnant-at-intermediate-level)		R→M (deteriorated)	
	Con	Exp	Con	Exp	Con	Exp	Con	Exp
	0	0	5.5%	11.7%	34.6%	1.7%	0	0
R	P→R (Improved++)		U→R (Improved+)		M→R (Improved)		R→R (Already-improved)	
	Con	Exp	Con	Exp	Con	Exp	Con	Exp
	0	0	0	8.3%	16.4%	43.3%	38.2%	36.7%
	P		U		M		R	

The average percentage outlook for parallel arrays for M→R (*Improved*) transition remains higher in the experimental group and the M→M (*Stagnant-at-intermediate-level*) transition also results in higher average percentage in the control group.

The pairs that resulted in 100% transition are pair 1 (Pre\_array\_initialised and Post\_array\_initialised) in Appendix J Table J.3; pair 2 (Pre\_loop\_structure and Post\_loop\_structure) in Appendix J Table J.6; pair 3 (Pre\_if\_condition and Post\_if\_condition) in Appendix J Table J.9; and pair 5 (Pre\_display and Post\_display) in Appendix J Table J.15. In cycle 3, this concept of parallel arrays resulted in all pairs into 100% M→R transition. In this cycle in the same concept, only one pair out of five pairs did not result in 100% transition. This is a sign of performance consistency of TLPAP on SIPS.

The R→R (*already-know*) transition resulted to higher percentages in pair 1 for both the control (63.6%) and experimental (66.7%) groups. This affected the number of algorithm sections in the M→R transition as it recorded lower percentages (0% in the control group, 16.7% in the experimental group). The percentages in the R→R transition do not have direct pedagogical implication on either the control or experimental group as I was unable to control what or how students learn outside the parameter of the action research study.

In this concept of parallel arrays, one of the notable results is in pair 3 where the  $M \rightarrow M$  transition resulted in 54.5% in the control group and 0% in the experimental group. Based on the previous  $M \rightarrow M$  (*Stagnant-at-intermediate-level*) transitions across all concepts, this percentage is the highest – setting a record. Perhaps this is caused by the pair requirements which need the correct structure of condition in the if-statement which must make use of parallel array. Irrespective, this is further evidence that the normal teaching method in the control group can only, to a certain extent, help SIPS. In pair 5, the  $U \rightarrow R$  (*Improved+*) transition resulted in 0% in the control group and 25% in the experimental group. The  $U \rightarrow R$  transition is part of improving categories and adds on to the success of the  $M \rightarrow R$  transition generated by TLPAP in the experimental group.

### 8.3.5 Results (*functions*)

There were 22 SIPS participants here, that is 10 students in the control group and 12 students in the experimental group.

#### 8.3.5.1 Algorithm sections

The problem specification for *functions* (pre-teaching exercise) remains the same as indicated in Chapter 7, section 7.3.4 in Figure 7.62. The problem specification for *functions* (post-teaching exercise) remains the same as indicated in Chapter 7, section 7.5.7.1 in Figure 7.86. The algorithm section remains the same for the pre-teaching exercise (section 7.5.7.1 in Figure 7.85) and post-teaching exercise (section 7.5.7.1 in Figure 7.87). The algorithm pairs remain the same as in Chapter 7, section 7.5.7.1, Table 7.11.

#### 8.3.5.2 Concept and pair performance

The content of Table 8.7 shows the average percentage of transitions across pair 1 to pair 6, presenting the overall performance of the *functions*. The evaluation sheets per pair are presented in Appendix K.



**Table 8.7: Average transitions performance on *functions* for all pairs**

P	P→P (Stagnant-no-learning)		U→P (Lower-level-deterioration)		M→P (Intermediate-deterioration+)		R→P (deteriorated++)	
	Con	Exp	Con	Exp	Con	Exp	Con	Exp
	0	0	0	0	0	0	0	0
U	P→U (Lower-level-improvement)		U→U (Stagnant-at-lower-level)		M→U (Intermediate-deterioration)		R→U (deteriorated+)	
	Con	Exp	Con	Exp	Con	Exp	Con	Exp
	0	0	0	0	0	0	0	0
M	P→M (Intermediate-improvement+)		U→M (Intermediate-improvement)		M→M (Stagnant-at-intermediate-level)		R→M (deteriorated)	
	Con	Exp	Con	Exp	Con	Exp	Con	Exp
	0	0	23.3%	8.3%	31.7%	2.8%	10%	0
R	P→R (Improved++)		U→R (Improved+)		M→R (Improved)		R→R (Already-improved)	
	Con	Exp	Con	Exp	Con	Exp	Con	Exp
	0	0	0	7%	26.7%	72.2%	8.3%	9.7%
	P		U		M		R	

The performance of the M→R (*Improved*) transition has served as the main indicator of success of TLPAP in the experimental groups in the precious concepts and cycle. In the concept of *functions*, the M→R transition has 72.2% in the experimental group while the control group has 26.7%. This conceptual performance of *functions* in the M→R transition is not just higher than the previous cycle (cycle 3) which recorded 62.2%, but has set a record, higher than all previous M→R transitions. The pedagogical implication of these results is that TLPAP has continued to heighten its impact in the experimental group as compared to the control group.

The pairs that resulted in 100% M→R transition in the concept of *functions* are pair 1 (Pre\_func\_call\_return and Post\_func\_call\_return) in Appendix K Table K.3; pair 2 (Pre\_func\_call\_void and Post\_func\_call\_void) in Appendix K Table K.6; pair 3 (Pre\_func\_header\_return and Post\_func\_header\_return) in Appendix K Table K.9; pair 4 (Pre\_func\_return\_content and Post\_func\_return\_content) in Appendix K Table K.12; and pair 5 (Pre\_func\_header\_void and Post\_func\_header\_void) in Appendix K Table K.15. The TLPAP is further proving its capability on 100% M→R transition in the experimental group as compared to the control group.

The M→M (*Stagnant-at-lower-level*) transition in the control group has also resulted in higher percentages in pair 2, pair 5 and pair 6. In pair 2, M→M transition has 50% in the control group and the experimental group has 8.3%. In pair 5, M→M transition has 40% in the control group and the experimental group has 0%. In pair 6, M→M transition

has 40% in the control group and 8.3% in the experimental group. This M→M transition outcome has been consistently higher in past concepts and cycles within the control group compared to the experimental group. This is a continuing indication that the teaching method in the control group is less effective, especially on more challenging concept like *functions*. The last notable result is pair 1 which has a 40% U→M (*Intermediate-improvement*) transition in the control group and 8.3% in the experimental group. As emphasised in the earlier concepts, this transition is decent or can be referred as ‘work in progress’ but is not quite enough – the desired transition is U→R instead of U→M.

## 8.4 Summary of reflections on cycle 4

The following reflections emanate from this cycle.

- The uses of *assignment* put emphasis on initialisation, copying of values from one variable to another or overwriting. The performance of the experimental group has been consistently positive from the previous cycles as reported in cycle 2 (Chapter 6, section 6.6.1) and cycle 3 (Chapter 7, section 7.5.1). This serves as a confirmation of the validation of TLPAP as this is a repeat experiment. No changes are foreseen in the next cycle in this experiment.
- Generally, the post-teaching algorithm exercises in the experimental group recorded higher *improved* (M→R) results than the control group.
- The post-teaching algorithm exercises in the experimental group recorded higher *intermediate-improvement* (U→M) than the control group. However, the control group has a reasonable share in the U→M transition. This means that a normal teaching method does help students, but at a slower pace, which resulted in intermediate achievements.
- The *deteriorated* (R→M) pairs in the sections of the algorithm were found mostly in the control group, but they were minimal. This suggests that the normal teaching method does not cause more harm, but it is still not enough to reach the ultimate stage of learning.
- The *improved+* (U→R) pairs in the sections of the algorithm were found mostly in the experimental group, but they were minimal. This means the TLPAP in the

experimental group was good enough to enforce learning for a few students who were really struggling to the ultimate stage of learning.

- Pedagogical guidelines as part of TLPAP were followed (using the adapted version in chapter 7) when demonstrating the animation programs. There are no revisions at this stage.
- The animation speed adjustment on the animation programs had an impactful effect on myself as an educator. This is because my narration was relaxed and not rushed when the animation speed was on slow or very slow. This means the speed adjustment is not only good for the student but for the educators as well.

## 8.5 Summary of reflections on cycle 5

The outcome of cycle 5 is similar to that of cycle 4, with no substantial differences between those cycles. As a result, I only point out reflections of cycle 5 in this section and briefly compare significant differences across cycle 3 to cycle 5 in section 8.6. The intention of proceeding to cycle 5 experimentation was to further confirm the reliability and efficacy of the TLPAP in the experimental group as compared to the control group.

In cycle 5, the animation programs adapted into Java programming language include animation for *decisions* and the combination of *decisions* and *loops* (sample programs in Appendix L). While the animations for the uses of *assignment* were also part of the Java experiments, there was no adaption since they are language-independent. The tutors oversaw the control and experimental groups. The tutors in the experimental group were properly briefed about TLPAP use. The improvements within the post-teaching algorithm sections remained consistent even when tutors were in charge. One suggestion from the tutors included embedding the summary of pedagogical guidelines into the animation programs where they can be accessed with a button click. Other concepts, like *loops*, *nested loops*, *arrays* and *functions* were presented to C++ introductory programming students.

The outcome of teaching and learning the concept of *assignment* produced similar outcomes to the previous cycles. The rest of the concepts presented in cycle 5 also produced similar experimental outcomes. For a thorough overview across cycles, the following section compares the most common transitions across cycle 3 to cycle 5.

## 8.6 Discussions on transitions across cycle 3 to cycle 5

The SOLO-adapted evaluations on cycle 3 to cycle 5 were based on the algorithm pairs/sections while in cycle 2 these were based on the overall algorithm. For a fair comparative view across cycles, the discussion in this section will focus on cycle 3 to cycle 5.

The most common frequently occurring SOLO-adapted transitions between cycle 3 and cycle 5 in both the control and experimental groups were  $M \rightarrow R$  (*improved*),  $U \rightarrow M$  (*intermediate-improvement*),  $M \rightarrow M$  (*stagnant-at-intermediate-level*) and  $R \rightarrow R$  (*already-know*). In the previous sections, the discussion indicated that the  $M \rightarrow R$  and  $M \rightarrow M$  transition had a major implication on the difference between the control group and experimental group. Each concept from cycle 3 to cycle 5 resulted in higher average percentages under the  $M \rightarrow R$  transition and lower under the  $M \rightarrow M$  transition. In the control group, the opposite occurred, that is, a higher  $M \rightarrow M$  transition and lower  $M \rightarrow R$  transition rate. The results of those transitions further confirm the pedagogical implications between the two groups where the TLPAP in the experimental group outperformed the normal teaching method in the control group.

The average percentage for  $U \rightarrow M$  transition in various concepts across cycle 3 to cycle 5 has been lower than 32% in both the control and experimental groups. The highest average percentage under the  $U \rightarrow M$  transition is found in the experimental group with 31.6% for the concept of *loops* (cycle 3). The second highest average percentage is also in the experimental group with 27.5% for the concept of *functions* (cycle 3). The  $U \rightarrow M$  transition is an improving category; however, the outcome can be regarded as semi-improvement since the transition resulted into  $U \rightarrow M$  instead of  $U \rightarrow R$  (*improved+*). The implications of the  $U \rightarrow M$  results in relation to the intervention in both groups is that the teaching method has made a good attempt but not enough for ultimate success.

The transition which occurred most frequently in the control group in a few algorithm sections is the  $R \rightarrow M$  (*deteriorated*) transition. The average percentages for  $R \rightarrow M$  transition in the control group have been 10% or less across cycle 3 to cycle 5, except in the concept of *decisions/nested decisions* in cycle 4 which resulted in 17%. The average percentages for  $R \rightarrow M$  transition in the experimental group have been either 0 or less than 5%. This means that even though the average  $R \rightarrow M$  transitions in the

control group are higher than the experimental group, the teaching methods in both did not lead to severe learning deterioration.

The U→U (*stagnant-at-lower-level*) transition has the majority of 0% in both groups across cycle 3 to cycle 5, meaning that there is nothing to comment about the transition. The average percentage in the R→R (*Already-know*) transition across cycle 3 to cycle 4 has no direct implication on the intervention in either the control or experimental group. This is because the intervention occurred while such pairs were already in the relational stage. Consequently, I was unable to evaluate how they acquired such knowledge or control how they show learning outside the experimental setup of this study. However, the R→R transition serves an important methodological aspect that helps discard the number of algorithm sections that are correct before the intervention. This further assures that the performance of TLPAP is not linked to the R→R transition in anyway.

There are instances where the M→R had a 100% transition. The 100% transition in the M→R category means that all pairs that were categorised in the multistructural stage in the pre-test evaluation of a certain pair have transitioned into the relational stage. If the transition split in the post-test evaluation into 2 or more categories, it is no longer a 100% M→R transition.

**The following sections are pairs that resulted in a 100% transition in the experimental and control groups**

In the concept of *decisions/nested decisions*, the 100% M→R transitions in the group are:

Pair 1 (Pre\_nested\_if and Post\_nested\_if) in cycle 3 and cycle 5

Pair 2 (Pre\_inner\_if\_AND and Post\_if\_AND) in cycle 4 and cycle 5

Pair 3 (Pre\_outer\_if\_OR and Post\_if\_OR) in cycle 3

Pair 4 (Pre\_inner\_if\_content and Post\_if\_OR\_content) in cycle 3

Pair 5 (Pre\_outer\_else\_content and Post\_else\_content) in cycle 3

Pair 6 (Pre\_inner\_else\_content and Post\_else\_content) in cycle 3 and cycle 5

In the concept of a combination of loops and *decisions/nested decisions*, the 100% M→R transitions include:

Pair 1 (Pre\_loop and Post\_loop) in cycle 4

Pair 2 (Pre\_loop\_content and Post\_loop\_content) in cycle 4 and cycle 5  
Pair 3 (Pre\_if\_OR and Post\_if\_OR) in cycle 5  
Pair 4 (Pre\_if\_OR\_content and Post\_if\_OR\_content) in cycle 4 and cycle 5  
Pair 5 (Pre\_if\_OR\_2 and Post\_if\_OR) in cycle 5  
Pair 6 (Pre\_if\_OR\_2\_content and Post\_if\_OR\_content) in cycle 4  
Pair 7 (Pre\_else and Post\_else) in cycle 5  
Pair 8 (Pre\_display and Post\_display) in cycle 4

In the concept of a combination of a *loops* the 100% M→R transitions include:

Pair 2 (Pre\_firstloop\_structure and Post\_loop\_structre) in cycle 5  
Pair 4 (Pre\_secondloop\_structure / Pre\_secondloop\_content and *Post\_display*) in cycle 3

In the concept of a combination of nested-loop, the 100% M→R transitions include:

Pair 1 (Pre\_outer\_loop\_structure and Post\_outer\_loop\_structure) in cycle 5  
Pair 2 (Pre\_inner\_loop\_structure and Post\_inner\_loop\_structure) in cycle 3 and cycle 5  
Pair 3 (Pre\_inner\_loop\_content and Post\_inner\_loop\_content) in cycle 5  
Pair 4 (Pre\_outer\_loop\_content and Post\_outer\_loop\_content) in cycle 5

In the concept of one-dimensional arrays, the 100% M→R transitions include:

Pair 1 (Pre\_array\_initialised and Post\_array\_initialised) in cycle 3 and cycle 5  
Pair 2 (Pre\_loop\_structure and Post\_loop\_structure) in cycle 3  
Pair 3 (Pre\_if\_condition and Post\_if\_condition) in cycle 3 and cycle 5  
Pair 4 (Pre\_if\_content and Post\_if\_content) in cycle 5  
Pair 5 (Pre\_display and Post\_display) in cycle 3 and cycle 5

In the concept of parallel arrays, the 100% M→R transitions include:

Pair 1 (Pre\_array\_initialised and Post\_array\_initialised) in cycle 3 and cycle 4  
Pair 2 (Pre\_loop\_structure and Post\_loop\_structure) in cycle 3 and cycle 4  
Pair 3 (Pre\_if\_condition and Post\_if\_condition) in cycle 3 and cycle 4  
Pair 4 (Pre\_if\_content and Post\_if\_content) in cycle 3

Pair 5 (Pre\_display and Post\_display) in cycle 3 and cycle 4

In the concept of *functions*, the 100% M→R transitions include:

Pair 1 (Pre\_func\_call\_return and Post\_func\_call\_return) in cycle 3, cycle 4 and cycle 5

Pair 2 (Pre\_func\_call\_void and Post\_func\_call\_void) in cycle 5

Pair 3 (Pre\_func\_header\_return and Post\_func\_header\_return) in cycle 3, cycle 4 and cycle 5

Pair 4 (Pre\_func\_return\_content and Post\_func\_return\_content) in cycle 4 and cycle 5

Pair 5 (Pre\_func\_header\_void and Post\_func\_header\_void) in cycle 4 and cycle 5

Pair 6 (Pre\_func\_void\_content and Post\_func\_void\_content) in cycle 3

### **The following sections are pairs that resulted in a 100% M→R transition in the control group**

The control group had three instances where the 100% M→R transition occurred. Pair 3 under *decisions/nested decisions* in cycle 4, pair 1 under the concept of a parallel array in cycle 3 and pair 4 in a one-dimensional array in cycle 3. The description of the pairs is as follows:

Pair 3 (Pre\_outer\_if\_OR and Post\_if\_OR) in cycle 3

Pair 1 (Pre\_array\_initialised and Post\_array\_initialised) in cycle 3

Pair 4 (Pre\_if\_content and Post\_if\_content) in cycle 3

### **Concluding remarks**

In this concept of *decisions/nested decisions*, the M→R 100% transition occurred once in the control group and nine times in the experimental group. In the concept of loops and nested-loops, the 100% transition occurred two times in the experimental group (for loops) and five times (for *nested loops*). The control group had no M→R 100% transitions on either loops or *nested loops*. There were 10 occurrences of 100% M→R transition in the experimental group under the concept of the combination of *decisions* and *loops* and nothing in the control group. The concept of one-dimensional arrays recorded in eight instances of 100% M→R transitions and parallel arrays has

nine 100% M→R transitions in the experimental group. In the control group, the 100% M→R transition occurred once for both one-dimensional array and parallel array. In total, there have been 36 instances of 100% M→R transition in the experimental group across cycle 3 to cycle 5. The control group has three instances of 100% M→R transition across cycle 3 to cycle 5.

In the case where the experimental group did not record 100% M→R transition, the advantage is still in the experimental group due to higher numbers in the M→M transition of the control group. The 100% M→R transition occurrences are an important indicator of TLPAP success within algorithm pairs. Its importance is due to the fact that the transition outcomes all result to improvements. A case where there is no 100% M→R transition is when other transitions like M→M (*stagnant-at-intermediate-level*), M→U (*intermediate-deterioration*) and M→P (*intermediate-deterioration+*) occur in the same algorithm pair.

## 8.7 State of the framework

At the end of cycle 4 of our action research, the state of the TLPAP framework was referred to as **adapted TLPAP framework version 4** pending the outcome of cycle 5 results that could suggest further adaptation towards a final framework. At the end of cycle 5 of the action research, the state of the TLPAP framework was referred to as the **final TLPAP framework**.

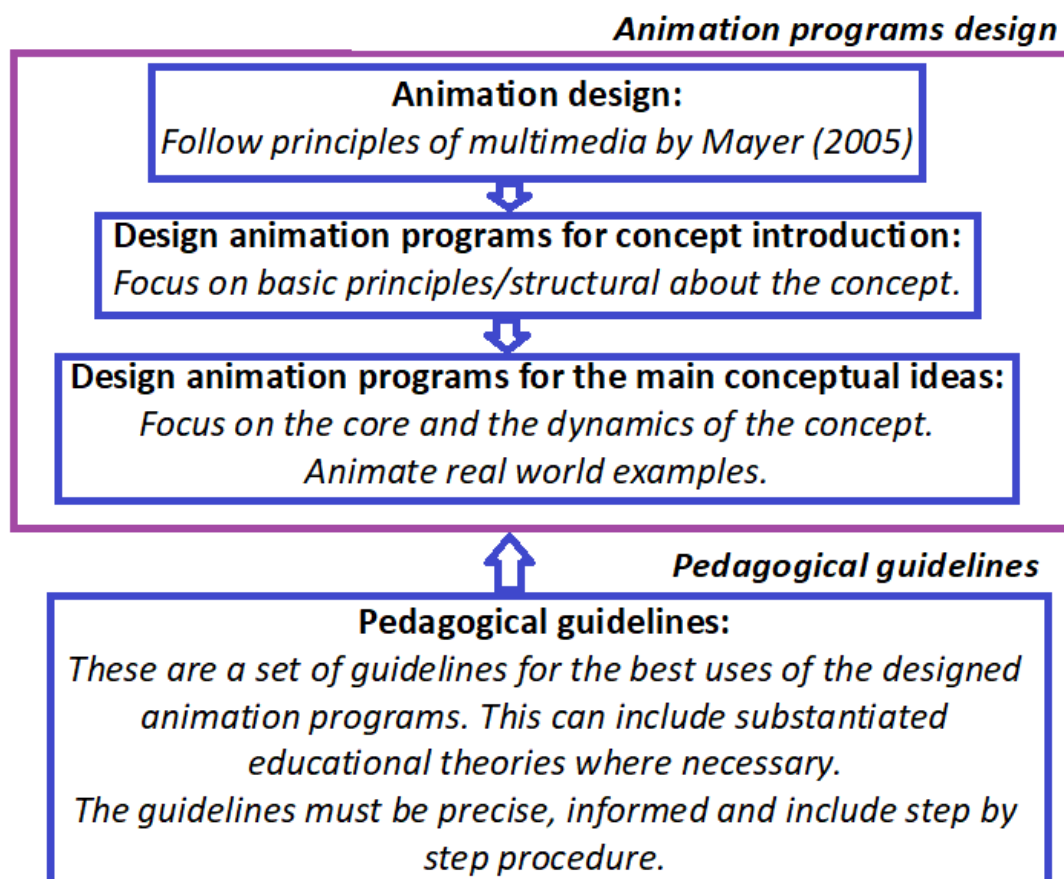
The specific items that form part of this final TLPAP framework consist of the following:

- The animation programs focusing on *assignment* (presented in cycle 2, chapter 6) as opposed to the manipulatives version (presented in cycle 1, chapter 5). Even though the use of physical manipulatives yielded positive results in emphasising the uses of *assignment*, the rationale behind this choice is the animation's flexibility to accommodate large classes in a contact teaching session and its compatibility with an online session.
- All the animation programs for introducing a concept as presented in Chapter 7, section 7.2, as well as an additional introductory animation program presented in Chapter 8, section 8.2.2.
- The nested-decider animation program presented in cycle 2.



- The animation programs for *loops*, *arrays* and *functions* presented in cycle 4.
- The animation program on the combination of *decisions* and loops as presented in cycle 4.
- The adapted pedagogical guidelines revised in cycle 4.

The lessons learned in each cycle of action research has provided insight to further derive a high-level, generic framework for teaching and learning with animation programs. This framework is deemed generic because it is not limited to introductory programming only but can be adapted to develop relevant animation programs for other similar modules or non-programming subjects. See Figure 8.18 for a depicted generic framework.



**Figure 8.18: Generic framework**

The generic framework in Figure 8.18 contains the critical high-level components of the TLPAP framework; that is the animation program design and pedagogical guidelines. In the part of animation program design, the initial step is the application of the principles of multimedia learning by Mayer (2005). These principles must be followed when designing animation programs for educational purposes. Secondly,

animation programs for concept introduction are recommended. This is to instil basic knowledge about the concept before introducing the main aspects of the concept. Thirdly, the main animation programs must focus on the core of subject matter and make use of real-world examples or problems for animating. Teaching and learning through the main animation programs should be less challenging after using the introductory animation programs and further reduce any cognitive overload in the case of complex concepts. The inclusion of introductory animation is crucial as struggling students may not focus on the basics during the main animation programs demonstration but are expected to focus on the main ideas presented at that stage.

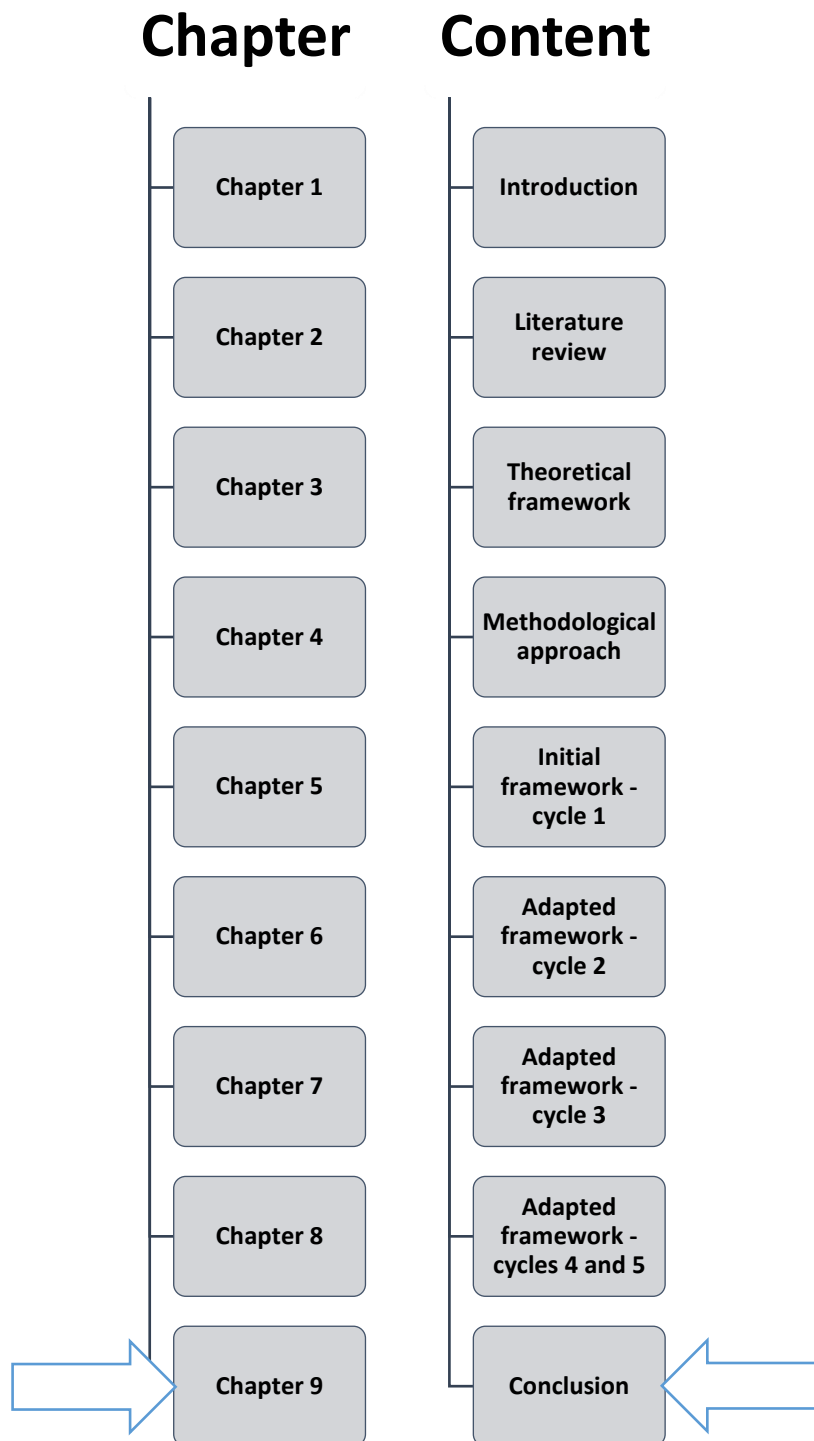
The last part of the generic framework is pedagogical guidelines. Pedagogical guidelines are a crucial component of the framework because they prescribe the best practices for teaching and learning with animation programs in that subject or module. Pedagogical guidelines should be adapted based on the nature of the subject as was done for the TLPAP framework.

## 8.8 Conclusion

In this chapter, I presented the adapted TLPAP framework where various animation programs for IPC were designed and tested. In addition to the animation programs designed and experimented in previous cycles, added animation programs were developed and experimented within cycle 4. Such animation programs were designed under the concept of loops and *decisions* (a combination of both) and additional animation for introducing an array. The pedagogical guidelines adapted in cycle 3 were used in cycle 4 and cycle 5 along with the animation programs as part of TLPAP. Generally, the experiments in these cycles showed consistent improvements in the experimental group across the concepts and previous cycles.

At the end of cycle 5, a final TLPAP framework was concluded and prescribed. The components that form part of TLPAP were all identified and referenced across cycles. Moreover, based on the experience of developing TLPAP, a generic or high-level framework was derived and presented. A generic framework can be adapted and used in other subject areas beyond introductory programming. The next chapter presents the discussions and conclusion of this study.

# Chapter 9: Conclusion



## 9.1 Introduction

The initial aim of this study was to make use of manipulatives for teaching and learning Introductory Programming Concepts (IPC). Through rigorous action research, the study evolved into using animation programs for teaching and learning IPC as each action cycle was reviewed, reflected on and planned accordingly. The core problem as stated in Chapter 1 is that some students find programming too challenging to comprehend. The problem deteriorates into reduced programming performance, a higher dropout rate and academic exclusion in the field of computing.

In the spectrum of pedagogical approaches in teaching and learning programming (discussed in Chapter 2), the use of multimedia tools/animation programs is found to be informative, explanatory, explicit, interactive, effective and clear. The gap within the existing animation programs for teaching and learning IPC is a lack of simplicity which can be key factor to SIPS. Some animation programs contain overloaded graphics which can deviate SIPS' focus or distract from the actual learning. Animation programs for teaching and learning should not be an end to themselves, but rather a way to learn IPC. Existing animation programs do not have a set of pedagogical guidelines compatible with the developed animation programs but rely on the assumption that users can figure out how to properly use the animation programs. Furthermore, the existing animation programs focus more on the K12 stream and block-based programming while there is a dearth of intentional research on struggling text-based programming students at university.

In Chapter 3, theoretical framework for this study was discussed, including constructivism, the conversational framework, teaching philosophy, cognitive theory on multimedia learning and the structure of observed learning outcomes (SOLO). In Chapter 4 the methodology followed in this study was discussed. The discussion in the methodology chapter included the research paradigms and multimedia design guidelines by Mayer (2005). The overall methodology followed is action research in the educational setting. The results of five action research cycles have been recorded and analysed in Chapter 5, Chapter 6, Chapter 7 and Chapter 8.

In cycle 1 (presented in Chapter 5), the teaching and learning activities were based on physical manipulatives as this was the initial plan. In that cycle, the framework was referred to as teaching and learning programming with manipulatives (TLPM). The planned manipulatives were based on teaching and learning the uses of *assignment* and *decisions/nested decisions*. The experimentation was undertaken through pre-test and post-test of algorithms in the control and experimental groups.

The outcome of teaching and learning on the uses of *assignment* was successful, but the outcome of teaching and learning *decisions/nested decisions* was not successful. The main problem encountered in an attempt to use the manipulatives for teaching and learning *decisions/nested decisions* was that the details of the manipulatives were not clearly visible to students. Additionally, the problem was worsened by the Covid-19 pandemic as I had to comply with government regulations on social distancing and higher education opted for online teaching and learning. In cycle 2, the physical manipulatives approach was changed to virtual manipulatives. The virtual manipulatives were referred to as multimedia teaching and learning tools and ultimately referred to as animation programs. Pedagogical guidelines were initiated and followed from this cycle. The framework was referred to as Teaching and Learning Programming with Animation Programs (TLPAP) from cycle 2 onwards.

The development of the animation programs for *assignment* was a direct translation from physical manipulatives to animation programs. The animation programs for *decisions/nested decisions* were also designed based on problem specifications and experimented with in cycle 2. In the case of the animation programs that emphasise the use of *assignment*, experimentation was successful, similar to the outcome of physical manipulatives experimentation in cycle 1. In the case of animation programs for *decisions/nested decisions*, the experimentation was also successful; however, a need to improve the evaluation method was noted and planned for cycle 3.

One challenge encountered by SIPS in cycle 2 (in the concept of *decision/nested decisions*) was persistent error in the basics and simple structural aspects of the concept. The small misalignment between pre-teaching and post-teaching exercises in the concept of *decision/nested decisions* was noted for adjustment in the next cycle.

A plan to improve and design more animation programs for various IPC was noted for cycle 3.

In cycle 3, the introductory animation programs were introduced to focus on the basics and simple structure of the concept before SIPS can be taught with the animation programs which are tied to a problem specification. The experimented animation programs were based on the concept of *assignment*, *decisions/nested decisions*, *loops/nested loops*, *one-dimensional arrays/parallel arrays* and *functions*. The pedagogical guidelines formulated in cycle 2 were adapted and followed in cycle 3. As part of the action research methodology which allows continual improvements per cycle, two additional animation programs were planned for cycle 4.

In cycle 4, the first additional animation program based on a problem specification included the combination of *decisions* and *loops*. The other animation program was an addition to the introductory animation program on the concept of arrays. The framework was then referred to as *TLPAP adapted framework version 4*. In cycle 5, the animation programs were repeated as they were in cycle 4. In the previous cycles, the animation programs were presented along with a C++ program code, but adjustments in cycle 5 included a radio button switch from a C++ code to a Java code. Some of the animations were presented to Java SIPS by tutors and some were still presented to C++ SIPS. At the end of cycle 5, the framework was then referred to as *TLPAP final framework*. The subsequent section discusses the nature of the actual TLPAP.

## 9.2 Structure of the TLPAP framework

The TLPAP framework has been designed specifically for SIPS in a text-based programming environment; however, it can be adopted as a general teaching and learning framework for all introductory programming students. The TLPAP consists of animation programs based on imperative programming concepts (objects-later approach) and the applicable pedagogical guidelines. The pedagogical guidelines are the stages to be followed for proper use of animation programs and effective learning benefits. The pedagogical guidelines can also be seen as a mapping model between TLPAP and other teaching and learning strategies. The animation programs are in two

phases: animation programs for introducing a new concept and animation programs for use after the introduction of a concept. The introductory animation programs consist of simple animations that give SIPS a structural basis about the concept being taught. The animation programs to be used after introductory animation programs are coupled with problem specifications. The idea of these two phases of animation programs is that SIPS should not be struggling with the basics while attempting to solve a problem. For example, if the concept of *decisions/nested decisions* is taught, introductory animation programs based on *decisions/nested decisions* are used. The animation programs with problem specifications should also be based on *decisions/nested decisions*.

The design of the animation programs complied with the 12 principles of multimedia learning by Mayer (2005) to ensure that the animation programs are within the parameters of the tested theoretical basis in teaching and learning. The animation programs' design was simplified for easy comprehension by SIPS. The interface of the animation programs is divided into two: the left side for animation rendering and the right side for program code. The rendering of animations occurs synchronously with program code. When the animation program executes, the program code is highlighted line by line from top to bottom and the corresponding animation occurs as per the currently highlighted line of code. It is recommended that the users of the animation programs familiarise themselves with pedagogical guidelines as this process forms part of TLPAP.

### 9.3 How the research questions were answered

The research questions and objectives formulated in this study emanated from the following thesis statement:

*A Teaching and Learning Programming with Animation Programs (TLPAP) framework can improve Struggling Introductory Programming Students (SIPS) comprehension of Introductory Programming Concepts (IPC).*

The research questions and objectives were developed according to the formulated thesis statement. The following section explains how each research question was answered.

**Research question 1:**

*What are the leading issues in teaching and learning IPC?*

To answer the above research question, the objective was formulated as: *To conduct a literature review on the challenges of teaching and learning IPC.* The literature review on the challenges of teaching and learning to program was conducted and discussed in Chapter 2, section 2.4. In the literature review, I found that there are many issues for teaching and learning to program. The attempt to solve these challenges was addressed in another section of literature review (Chapter 2, section 2.6) which is addressed through the subsequent research question.

**Research question 2:**

*What are the approaches used in teaching IPC?*

To answer the above research question, the objective was formulated as: *To investigate the approaches used in teaching IPC.* The literature review on approaches used in teaching and learning programming concepts was conducted and discussed in Chapter 2, section 2.6. This section discussed various approaches of teaching and learning to program like the use of physical manipulatives and animation programs. The conclusion of this section suggested and substantiated the incorporation of animation programs in the programming education.

**Research question 3:**

*What are the limitations of existing animation programs used for teaching and learning IPC?*

To answer the above research question, the objective was formulated as: *To investigate animation programs for teaching and learning IPC.* The literature review on animation programs for teaching and learning IPC was conducted and discussed in Chapter 2, section 2.7. Various animation programs and their limitations were discussed in this section. The main findings include lack of rigorous reference framework that can be followed to develop and design the animation programs and animation programs that are not accompanied by relevant pedagogical guidelines,



#### **Research question 4:**

*How can we develop a TLPAP framework for teaching and learning IPC?*

To answer the above research question, the objective was formulated as: *To develop a TLPAP framework for teaching and learning IPC based on gaps identified in the literature review and by continually improving the framework based on the outcome of the action research cycles.* The objective was carried out by initially outlining a theoretical framework relevant to TLPAP in Chapter 3 and subsequently a research plan (methodology) as discussed in Chapter 4. The overall research process followed an action research methodology. The study went through five action research cycles from 2019 to 2022 (discussed in Chapters 5, 6, 7 and 8). The development, refinement and finalisation of the solution (TLPAP) was done through reflection, observation and planning within every action cycle.

#### **Research question 5:**

*How can we prove and realise the efficacy of the TLPAP framework?*

To answer the above research question, the objective was formulated as: *To evaluate the efficacy of the TLPAP framework through teaching the actual SIPS by following the TLPAP framework.* This objective was carried out by using the methodology in Chapter 4. The reports on the experiments are in Chapter 5 (action cycle 1), Chapter 6 (action cycle 2), Chapter 7 (action cycle 3), Chapter 8 (action cycles 4 and 5). A SOLO-adapted evaluation was formulated in cycle 2 in order to have meaningful evaluation of the pre-teaching and post-teaching algorithms in both the control and experimental groups. Through the application of a SOLO-adapted evaluation in each cycle, I was able to continually improve the TLPAP. The TLPAP framework was concluded and finalised in cycle 5 of the action research.

## **9.4 Contributions**

This research has resulted in the following contributions.

### **Contribution 1: Physical manipulatives**

The first contribution of this study is the use of physical manipulatives to emphasise the uses of *assignment* in programming. The IPC that can be taught with physical

manipulatives are limited to basic concepts like the uses of *assignment* which emphasise value storage/initialisation and copying of values from one variable to another (overwriting). Through pre-test and post-test methodology, the experimental group (using manipulatives to teach) performed better than the control group (no manipulatives used to teach). The full details of teaching and learning programming with manipulatives is presented in cycle 1 (Chapter 5).

### **Contribution 2: Animation programs**

The second contribution of this study is the development of animation programs that are in two parts, namely introductory animation programs and animation programs with problem specifications. The introductory animation programs, as stressed earlier, are used to introduce a new concept. The animation programs with problem specifications are used after the presentation of introductory animation programs. The animation programs with problem specifications are problem-solving oriented. The design guidelines of all animation programs are based on principles of multimedia learning by Mayer (2005).

We found that the learning gains of the physical manipulatives on *assignment* are similar to the learning gains of using animation programs. However, the advantages of animation programs are their online compatibility and accommodation for a large class for either contact or online presentation modes. The experiments with animation programs were initiated in cycle 2, then proceeded until the last cycle (cycle 5). Cycle 2 (Chapter 6) experiments were performed in an online mode and cycle 3 (Chapter 7) to cycle 5 (Chapter 8) in contact mode.

### **Contribution 3: Pedagogical guidelines**

The third contribution is the use of pedagogical guidelines in the TLPAP. One of the gaps identified in the literature (Chapter 2) is that existing animation programs or other multimedia-related teaching and learning tools were not coupled with relevant pedagogical guidelines. If the animation program for teaching and learning is without compatible guidelines, the assumption is that the users (mostly educators and students) will have to independently determine how to use them properly. This can cause animation programs to do more harm than the good purposes they are designed

for. In this study, the pedagogical guidelines were developed, substantiated and used along with the animation programs during action research cycles. The question of *what, when* or *how* to use animation programs for effective use in teaching and learning IPC have been addressed through prepared pedagogical guidelines. The pedagogical guidelines, in Chapter 7 section 7.4, form part of the final TLPAP along with the animation programs.

#### **Contribution 4: TLPAP framework and generic framework**

The fourth contribution of this study is the TLPAP framework and generic framework. Through SOLO-adapted evaluations, we were able to determine that the use of TLPAP has a positive effect on teaching and learning IPC for SIPS. The TLPAP framework is specifically intended to assist SIPS but can be used as a blanket teaching and learning approach for all students. The actual components that form the final TLPAP are referenced in Chapter 8, section 8.7.

The second part of this contribution is a generic framework. The generic framework has been derived from the completed TLPAP framework and can also be referred to as a high-level view of the TLPAP. The main idea behind the generic framework was to make it adaptable on the development of animation programs for non-programming subjects. The generic framework contains two main recommendations, namely animation programs and pedagogical guidelines. There are four main guidelines for a generic framework. Firstly, the design of the animation programs must follow principles of multimedia learning by Mayer (2005). Secondly, the introductory animation programs must be designed for introducing a specific concept. Thirdly, the main animation programs must be designed to show the core or main aspects of the concepts and must contain real-world examples. Lastly, the informed pedagogical guidelines must be developed in accordance with the nature of the subject, animations and other relevant teaching and learning strategies.

#### **Contribution 5: Methodological contribution**

The fifth contribution of this study is methodological. The adaptation of SOLO to form a SOLO-adapted evaluation was instrumental in validating the efficacy of TLPAP. The learning transitions in the SOLO-adapted evaluation were given a suitable, brief

description for easier cross-referencing between algorithms and meaningful feedback that informed pedagogical adjustments.

In summary, the SOLO-adapted evaluation is able to do the following:

- To qualitatively assess the pre-teaching and post-teaching sections of the algorithm in the control and experimental group. As a result, it was possible to detect the effects of TLPAP in the experimental group compared to the control group. The SOLO-adapted evaluation can also be used by programming educators to monitor learning from one written algorithm to another.
- To detect algorithm sections that have already been mastered through the  $R \rightarrow R$  transition. This transition means *already-improved*, i.e., mastered before the teaching commenced. This can be a benefit for programming teachers to have informed and meaningful feedback on certain areas of the algorithm as it could further prompt some early interventions.
- To detect algorithm sections that broadened the knowledge from one aspect to several aspects and if the students still fell short of mastering the relational stage. Such sections of the algorithms were detected through the  $U \rightarrow M$  transitions which means *Intermediate-improvement*,  $P \rightarrow U$  transition (*lower-level-improvement*) and  $P \rightarrow M$  transition (*Intermediate-improvement+*).
- To note whether the algorithm section moved from bad to worse or showed patterns of deteriorating learning. This can be detected through the  $R \rightarrow M$ ,  $R \rightarrow U$  or  $R \rightarrow P$  transitions.
- To detect whether the teaching intervention has resulted in the ultimate improvements was traced through the  $M \rightarrow R$ ,  $U \rightarrow R$  or  $P \rightarrow R$  transitions.
- To note whether the teaching intervention had no effect at all before and after the teaching intervention. This was detected through the  $P \rightarrow P$ ,  $U \rightarrow U$  or  $M \rightarrow M$  transitions. Programming educators can use these transitions to note sections of the algorithm that are stagnant, meaning that they were neither improving nor deteriorating.

### **Contribution 6: Theoretical contribution**

A theoretical contribution in this study revolves around guidelines recommended for forming a teaching philosophy within the field of CS education, programming education

or when adopting TLPAP. I recommend that a relevant teaching philosophy be associated with the nature of programming (coding) or programming education. Since a teaching philosophy is a personal teaching and learning statement which differs from educator to educator, I am not prescribing a specific teaching philosophy but provide guidelines that can channel the formation of a teaching philosophy with the nature of programming education. There are two main aspects that can help to develop a teaching philosophy relevant for programming education. Firstly, a programming educator must view programming from *computation as interaction*, as supported by Stein (1998), rather than *computation as calculations*, and the teaching approach must promote relational understanding, as borrowed from Skemp's (1976) theory on mathematics education. As emphasised in Chapter 3, section 3.6.3, the knowledge of those two theories can help programming educators formulate a fruitful teaching philosophy for the promotion of teaching and learning in the TLPAP or programming education in general.

## 9.5 Validity, generalisation and critical reflection on the findings

The animation programs in education can be a short-lived achievement, however incorporation of active learning and engagement can have a great impact on its effectiveness (Sorva, Karavirta & Malmi, 2013). In order to incorporate this, this study has relied on active related activities along with the use of animation programs. These active activities were drawn from the ideas of constructivism, conversational framework and principles of multimedia learning.

This study has paved a way to revolutionise the use of animation programs for teaching and learning purposes. This has also demonstrated the best way possible to integrate pedagogical guidelines with animation programs for teaching and learning to program. At the end, the TLPAP was developed and validated through various cycles of an action research methodology.

The validity of the findings in this can be attributed to its rigorous methodological approach which is SOLO-adapted evaluation. Through this methodology, it was possible to qualitatively filter the algorithm sections that were improving, deteriorating, not improving and already improved. The traditional assessment methods are based

on grading an isolated assessment where the results are only quantitative and do not possess meaningful interpretation. The major difference or advantage of SOLO-adapted evaluation is its ability to provide meaningful transitional feedback between two assessments. The SOLO-adapted evaluation can also be adapted to evaluate three or more assessments. I personally recommend this SOLO-adapted evaluation to be applied or used in any study field to qualitatively assess or evaluate students' answers. The SOLO-adapted evaluation can be reliably applied to any relevant field hence the conclusion of this study has made it possible to generalise or derive a generic framework based on the final TLPAP. The generic framework allows subjects like Mathematics, Science and Engineering to apply or reuse the SOLO-adapted evaluation to achieve similar outcomes as in programming education.

The development of animation programs for teaching and learning purposes cannot be based on nothing or on random design. It is recommended to follow the principles of multimedia learning by Mayer (2005) to develop animation programs. The final TLPAP or generic framework recommend that the animation programs must be in two phases, that is the introductory animation programs and main animation programs. The idea behind the separation of the animation programs was realised at the end of the second cycle of our action research. The introductory animation programs demonstrate the basic principles of programming and the main animation programs demonstrate real world problem solving through programming. Similarly, when applying the generic framework in a non-programming environment like Mathematics literacy, the introductory animation can be the basic rules of the mathematical equations while the main animation program contains the real-life experimental problems.

In the process of TLPAP development, there have been challenges and lessons learned. Firstly, the research was supposed to advocate the use physical manipulatives for teaching and learning the IPC. The nature of the action research methodology allowed me to reflect and plan into a different dimension (animation programs) of the research in the subsequent cycles. In other words, the "vehicle" changed to a more convenient "vehicle", but the trip and mission remained the same. In the second cycle of an action research, Covid19 restrictions made things difficult to have face to face experiments. As a result, cycle 2 was conducted online through Microsoft Teams. The Microsoft Teams experiments made me realised that the

animation programs are compatible with online presentation. The development of animation programs was the biggest challenge as I had to code an advanced program to depict the dynamics of introductory programming code. However, through dedication I was able to code and complete all the planned animation programs. The main agenda for future work remains the development of artificial intelligence methods to automatically generate a problem specification, program code and the applicable animation programs.

Reflection is one of the main aspects of an action research methodology. As a result, reflections were done at the end of each action cycle and planning for the next cycles was also done based on the reflections. The main point behind the multiple reflections is a way to improve the teaching methods. At the end of cycle 1, then main reflection was a transition from physical manipulatives to virtual manipulatives. The reflections at the end of cycle 2 included the need for additional animation programs and SOLO-adapted evaluation that need to be applied on the sections of the algorithms. In cycle 5, the main reflection was to do more experiments to improve validity, the inclusion of Java code and animation pace button. The reflections at the end on cycle 4 minimal, as a result additional experiments were carried for the confirmation of the effectiveness of the TLPAP. Furthermore, the additional Java code on the animation program was included in some animation programs. This means such animation programs has an option to display the animation programs in C++ code or Java code.

The overall reflections of this study can be summarised in two main printers, which is the uses of pedagogical guidelines and sample exercises in the animation programs. The uses of compatible pedagogical guidelines have been identified as a gap in this study and deemed as the crucial element of teaching and learning with animation programs in general. The pedagogical guidelines must be used along with the animation programs developed in this study, in order to realise the full benefits of TLPAP. These guidelines outline best practices for teaching and learning from body of knowledge that can be used along with the animation programs. In the case where the generic framework is adapted into a different field of study, pedagogical guidelines can be adapted accordingly to fit the nature of the study.

The sample exercises incorporated in the main animation programs are important, therefore a selection of such examples is also important. Programming is used to solve

real-life problem, as a result the TLPAP used a compatible real-world problem specification that can be solved through an algorithm as a solution. The solution (algorithm) to the problem specification covered the overall aspects of the concept within the main animation programs. I recommend the practice of incorporating real-life examples into the development of animation programs when adapting the generic framework proposed in this study into a different field. In this study, I developed one main animation program per concept, however two or more animation programs per concept can even be more effective as students are exposed to different problem specifications.

## 9.6 Limitations

The design of the animation programs was limited to the concepts offered in the introductory programming module at the university where the experiments took place. The TLPAP covers a computing education approach that advocates for objects later rather than objects early; therefore, there were no OOP (object-oriented programming) concepts. The number of students who participated per cycle was fewer than expected; hence the reliance on qualitative evaluation and multiple cycles. In the  $R \rightarrow R$  (*already-know*) transition, we were not able to trace the transition back to the success of TLPAP because we were not in full control of the students outside TLPAP experimentation.

## 9.7 Future work

The TLPAP framework is a teaching and learning solution for SIPS. The framework is dynamic and animation-based, rendering it comparable with various teaching and learning platforms or LMS. The TLPAP framework was experimented through a multicast broadcast on multiple computers and presented on the online platforms (MS Teams). Therefore, the assumption is it would work in a contact session using a projector screen or any other online platform like Google Meet or Zoom. An additional platform for future use can include a compatible mobile application.

The TLPAP framework has been experimented with in a higher education setup with the actual programming students on computing courses. Since there are schools that introduce and teach introductory programming from high school level, the framework can also be adopted and applied in such environments.



The pedagogical guidelines incorporated within TLPAP framework can be applied and tested with other animation programs for teaching and learning. Pedagogical guidelines for TLPAP are animation-based; as a result, they could be compatible with non-programming animation programs for teaching and learning.

The framework has proven effective in the objects-later programming paradigm. As part of future work, the TLPAP can be adapted into all or challenging aspects of an OOP paradigm. Some of the potential OOP concepts include classes, inheritance, encapsulation and polymorphism. This will broaden the TLPAP for both programming students in the objects-early and objects-later paradigms. Furthermore, the framework can be adapted to other educational settings (non-programming environments like engineering, mathematical courses or any other science-related disciplines).

Artificial intelligence tools like ChatGPT have the potential to impact in an academic sector (Lund & Wang, 2023). As part of future work, the possibility of auto-generating a problem specification that requires a programming solution for teaching and learning purposes can be investigated. Furthermore, such auto-generated problem specifications can auto-generate a short program code and animation programs for teaching and learning purposes. This will expand the animation program by providing various problem specifications and other animations.

## 9.8 Conclusion

In this study, we developed a teaching and learning framework named TLPAP. The TLPAP framework could, however, potentially be suitable to assist struggling students in other topics/ fields as well. The TLPAP consists of animation programs and pedagogical guidelines. The animation programs cover IPC within the imperative programming paradigm (objects-later approach). The concepts include *assignment*, *decisions*, *nested decisions*, *loops*, *nested loops*, *one-dimensional arrays*, *parallel arrays* and *functions*.

Through rigorous and repeated experiments, the TLPAP was thoroughly evaluated in an action research methodology (cycle 2 to cycle 5). The efficacy of the TLPAP was proven through a qualitative SOLO-adapted evaluation method which offered in-depth insight into dissected sections of the algorithms. The pre-test and post-test algorithms were written by both control and experimental groups. The control group was taught

through the traditional verbal teaching method (without TLPAP) and the experimental group was taught by following the TLPAP framework.

Struggling students were recruited to test the efficacy of TLPAP. This was purposeful to validate the TLPAP framework with actual SIPS rather than non-struggling students. Through the pre-test and post-test with SOLO-adapted evaluation, it was possible to detect non-struggling students through patterns of algorithm sections that were correct before and after the intervention (through  $R \rightarrow R$  transition). This further confirms the quality and validity of the SOLO-adapted evaluation method to the study.

The SOLO-adapted evaluations showed that the use of TLPAP can enhance comprehension of SIPS as compared to traditional methods and other existing related solutions discussed in Chapter 2. The SOLO-adapted evaluations outcomes like  $M \rightarrow R$  (*improved*),  $U \rightarrow M$  (*intermediate-improvement*),  $U \rightarrow R$  (*improved+*),  $R \rightarrow M$  (*deteriorated*),  $M \rightarrow M$  (*stagnant-at-intermediate-level*),  $U \rightarrow U$  (*stagnant-at-lower-level*) and  $R \rightarrow R$  (*already-know*) were major indicators in validating the efficacy of TLPAP.

The animation programs with problem specifications use specific problems and animations which are designed for teaching and learning purposes. The assumption is that SIPS should perform better or improve when given a different problem specification from the ones used in teaching and learning. To test this, the pre-teaching consisted of animation programs with certain problem specifications, then the post-test consisting of different problem specifications, while the concept being taught between the pre-test and post-test remains the same. It was proven that SIPS can perform better in different, but similar problem specifications to solve a programming problem, as noted in the experimental group.

A critical element in the TLPAP is the set of pedagogical guidelines. The pedagogical guidelines are compatible stages that have been substantiated and followed when using the designed animation programs. Animation programs are not isolated tools that can be used randomly or rely on an assumption of proper function; hence, the inclusion of these pedagogical guidelines.

In the final stages of the study, a generic framework for teaching and learning was developed based on the final TLPAP. The generic framework (can referred to as high-level view of the TLPAP framework) can be used to develop animation programs for

teaching and learning in subjects related to programming or other non-programming subjects. The main aspects of the generic framework are the design guidelines of the animations and the formation of relevant pedagogical guidelines. The belief is that the generic framework should be adaptable and convenient to adjust in different subject areas. Personally, I hope to see this framework evolve beyond the introductory programming education.

# References

- ACM/IEEE-CS Joint Task Force on Computing Curricula. (2013). *ACM/IEEE Computing Curricula 2013 Final Report*. <http://robotics.stanford.edu/users/sahami/CS2013/ironman-draft/cs2013-ironman-v1.0.pdf>.
- Adom, D., Yeboah, A., & Ankrah, A. K. (2016). Constructivism philosophical paradigm: Implication for research, teaching and learning. *Global journal of arts humanities and social sciences*, 4(10), 1-9.
- Aggarwal, A., Gardner-McCune, C., & Touretzky, D. S. (2017, March). Evaluating the effect of using physical manipulatives to foster computational thinking in elementary school. In *Proceedings of the 2017 ACM SIGCSE Technical Symposium on Computer Science Education*, 9-14.
- Ahadi, A., Lister, R., Lal, S., & Hellas, A. (2018, January). Learning programming, syntax errors and institution-specific factors. In *Proceedings of the 20th Australasian computing education conference*, 90-96.
- Aleksić, V., & Ivanović, M. (2016). Introductory programming subject in European higher education. *Informatics in Education*, 15(2), 163-182.
- Aldridge, J. M., Fraser, B. J., & Sebela, M. P. (2004). Using teacher action research to promote constructivist learning environments in South Africa. *South African Journal of Education*, 24(4), 245-253.
- Alpár, G., Yeni, S., Aivaloglou, E., & Hermans, F. (2022, March). Can Math Be a Bottleneck? Exploring the Mathematics Perceptions of Computer Science Students. In *2022 IEEE Global Engineering Education Conference (EDUCON)*, 217-225. IEEE.
- Altadmri, A., & Brown, N. C. (2015, February). 37 million compilations: Investigating novice programming mistakes in large-scale student data. In *Proceedings of the 46th ACM Technical Symposium on Computer Science Education*, 522-527. ACM.
- Anderson, L. W., & Krathwohl, D.R (2001). Anderson and Krathwohl Bloom's Taxonomy Revised Understanding the New Version of Bloom's Taxonomy.
- Anwar, R. B., Yuwono, I., As'ari, A. R., & Rahmawati, D. (2016). Mathematical representation by students in building relational understanding on concepts of area and perimeter of rectangle. *Educational Research and Reviews*, 11(21), 2002-2008.
- Anyango, J. T., & Suleman, H. (2018, November). Teaching Programming in Kenya and South Africa: What is difficult and is it universal? In *Proceedings of the 18th Koli Calling International Conference on Computing Education Research*, 1-2.
- Apuke, O. D. (2017). Quantitative research methods: A synopsis approach. *Kuwait Chapter of Arabian Journal of Business and Management Review*, 33(5471), 1-8.
- Armenti, S. M. (2018). *Computer Science Education with English Learners*. University of Rhode Island.
- Azuma, M., Coallier, F., & Garbajosa, J. (2003, September). How to apply the Bloom taxonomy to software engineering. In *Eleventh annual international workshop on software technology and engineering practice*, 117-122. IEEE.
- Babbie, E. (2014). *The Practice of Social Research*. Boston, Cengage Learning.
- Baldwin, D., Walker, H. M., & Henderson, P. B. (2013). The roles of mathematics in computer science. *ACM Inroads*, 4(4), 74-80.

- Bayman, P., & Mayer, R. E. (1983). A diagnosis of beginning programmers' misconceptions of BASIC programming statements. *Communications of the ACM*, 26(9), 677-679.
- Beaubouef, T. (2002). Why computer science students need math. *SIGCSE Bulletin*, 34(4), 57-59.
- Beaubouef, T., & Mason, J. (2005). Why the high attrition rate for computer science students: some thoughts and observations. *ACM SIGCSE Bulletin*, 37(2), 103-106.
- Ben-Ari, M. (1998). Constructivism in computer science education. *ACM SIGCSE bulletin*, 30(1), 257-261.
- Bennedsen, J., & Caspersen, M. E. (2007). Failure rates in introductory programming. *ACM SIGCSE Bulletin*, 39(2), 32-36.
- Bennedsen, J., & Caspersen, M. E. (2019). Failure rates in introductory programming: 12 years later. *ACM Inroads*, 10(2), 30-36.
- Bers, M. U. (2019). Coding as another language: A pedagogical approach for teaching computer science in early childhood. *Journal of Computers in Education*, 6(4), 499-528.
- Biggs, J. (1996). Enhancing teaching through constructive alignment. *Higher education*, 32(3), 347-364.
- Biggs, J. (2001). Constructive alignment. In *Background notes to support a seminar given by Professor John Biggs*. jbiggs@bigpond.com *Assessing language or content*.
- Biggs, J. B., & Collis, K. F. (1982). The psychological structure of creative writing. *Australian Journal of Education*, 26(1), 59-70.
- Biggs, J. B., & Collis, K. F. (2014). *Evaluating the quality of learning: The SOLO taxonomy (Structure of the Observed Learning Outcome)*. Academic Press.
- Biggs, J., & Tang, C. (2010, February). Applying constructive alignment to outcomes-based teaching and learning. In *Training material for "quality teaching for learning in higher education" workshop for master trainers, Ministry of Higher Education, Kuala Lumpur*, 23-25.
- Bloom, B. S. (1956). Taxonomy of educational objectives: The classification of educational goals. *Cognitive domain*.
- Botha, L., & Taylor, E. (2022). Information systems research: determining the correct research paradigm for a specific research problem.
- Bouck, E. C., Satsangi, R., Doughty, T. T., & Courtney, W. T. (2014). Virtual and concrete manipulatives: A comparison of approaches for solving mathematics problems for students with autism spectrum disorder. *Journal of Autism and developmental disorders*, 44, 180-193.
- Bransford, J. D., & Stein, B. S. (1993). The IDEAL problem solver.
- Brooke, C. (2002). What does it mean to be 'critical' in IS research?. *Journal of Information Technology*, 17(2), 49-57.
- Brown, M. H., & Sedgewick, R. (1984, January). A system for algorithm animation. In *Proceedings of the 11th annual conference on Computer graphics and interactive techniques*, 177-186.
- Brown, N. C., & Altadmri, A. (2014, July). Investigating novice programming mistakes: educator beliefs vs. student data. In *Proceedings of the tenth annual conference on International computing education research*, 43-50. ACM.
- Brown, N. C., & Altadmri, A. (2017). Novice Java programming mistakes: Large-scale data vs. educator beliefs. *ACM Transactions on Computing Education (TOCE)*, 17(2), 1-21.
- Bruce. (2015). <https://edwinbruce.wordpress.com/2015/07/08/programming-misconceptions-2/>

- Bryman, A. (2006). Paradigm peace and the implications for quality. *International journal of social research methodology*, 9(2), 111-126.
- Calitz, A. P. (2021, July). The 50 Year History of SACLA and Computer Science Departments in South Africa. In *Annual Conference of the Southern African Computer Lecturers' Association*, 3-23. Springer, Cham.
- Case, R. (1984). *Intellectual Development: a systematic reinterpretation*. New York, Academic Press.
- Case, R. (1992). Neo-Piagetian theories of child development. *Intellectual development*, 161-196.
- Cetin, I. (2020). Teaching loops concept through visualization construction. *Informatics in Education- An International Journal*, 19(4), 589-609.
- Cevahir, H., Özdemir, M., & Baturay, M. H. (2022). The Effect of Animation-Based Worked Examples Supported with Augmented Reality on the Academic Achievement, Attitude and Motivation of Students towards Learning Programming. *Participatory Educational Research*, 9(3), 226-247.
- Cheah, C. S. (2020). Factors contributing to the difficulties in teaching and learning of computer programming: A literature review. *Contemporary Educational Technology*, 12(2), 272.
- Chen, C., Haduong, P., Brennan, K., Sonnert, G., & Sadler, P. (2019). The effects of first programming language on college students' computing attitude and achievement: a comparison of graphical and textual languages. *Computer Science Education*, 29(1), 23-48.
- Chism, N. V. (1997). Developing a philosophy of teaching statement, Essays on teaching excellence. *Toward the best in the academy*, 9(3). 1-2.
- Clark, R. E., & Choi, S. (2005). Five design principles for experiments on the effects of animated pedagogical agents. *Journal of Educational Computing Research*, 32(3), 209-225.
- Clements, D. H., & McMillen, S. (1996). Rethinking "concrete" manipulatives. *Teaching children mathematics*, 2(5), 270-279.
- Commons, M. L., & Ross, S. N. (2008). What postformal thought is, and why it matters. *World Futures*, 64(5-7), 321-329.
- Cooper, S., Dann, W., & Pausch, R. (2000). Alice: a 3-D tool for IPC. *Journal of computing sciences in colleges*, 15(5), 107-116.
- Corral, J. M. R. (2019). Multimedia System for Self-learning C/C++ Programming Language. *Innovation in Information Systems and Technologies to Support Learning Research: Proceedings of EMENA-ISTL 2019*, 7, 55.
- Craig, M., Smith, J., & Petersen, A. (2017, November). Familiar contexts and the difficulty of programming problems. In *Proceedings of the 17th Koli Calling International Conference on Computing Education Research*, 123-127. ACM.
- Creswell, J. W. (2003). *Research design*. Thousand Oaks, 155-179. CA: Sage publications.
- Creswell, J. W., & Poth, C. N. (2016). *Qualitative inquiry and research design: Choosing among five approaches*. Sage publications.
- Creswell, J. W., Hanson, W. E., Clark Plano, V. L., & Morales, A. (2007). Qualitative research designs: Selection and implementation. *The counseling psychologist*, 35(2), 236-264.
- Crotty, M. J. (1998). The foundations of social research: Meaning and perspective in the research process. *The foundations of social research*, 1-256.
- De Vaus, D. (2001). Research design in social research. *Research design in social research*, 1-296.

- Delgado, C. A., da Silva, J. C., Mascarenhas, F., & Duboc, A. L. (2016, July). The teaching of functions as the first step to learn imperative programming. In *Anais do XXIV Workshop sobre Educação em Computação*, 2393-2402. SBC.
- Demetriou, A., Shayer, M., & Efklides, A. (2016). *Neo-Piagetian theories of cognitive development: Implications and applications for education*. Routledge.
- Denney, A. S., & Tewksbury, R. (2013). How to write a literature review. *Journal of criminal justice education*, 24(2), 218-234.
- Du Boulay, B. (1986). Some difficulties of learning to program. *Journal of Educational Computing Research*, 2(1), 57-73.
- Dümmel, N., Westfechtel, B., & Ehmann, M. (2019, April). Work in progress: Gathering requirements and developing an educational programming language. In *2019 IEEE Global Engineering Education Conference (EDUCON)*, 1-4. IEEE.
- Eckerdal, A., Thuné, M., & Berglund, A. (2005, October). What does it take to learn 'programming thinking'?. In *Proceedings of the first international workshop on Computing education research*. 135-142.
- Ehlert, A., & Schulte, C. (2009, August). Empirical comparison of objects-first and objects-later. In *Proceedings of the fifth international workshop on Computing education research workshop* (pp. 15-26).
- Elliott, J. (2011). Educational action research and the teacher. *Action Researcher in Education*, 1(1), 1-3.
- Ertmer, P. A., & Newby, T. J. (2013). Behaviorism, cognitivism, constructivism: Comparing critical features from an instructional design perspective. *Performance improvement quarterly*, 26(2), 43-71.
- Faisal, M., Yuana, R., & Basori, M. (2017, October). Comparative study between robomind and scratch as programming assistance tool in improving understanding of the basic programming concepts. In *International Conference on Teacher Training and Education 2017 (ICTTE 2017)* (pp. 787-797). Atlantis Press.
- Fares, K., Fowler, B., & Vegas, E. (2021). *How South Africa implemented its computer science education program*.
- Farrell, J. (2008). *Object-oriented programming using C++*. Cengage Learning.
- Fischer, B. F. (1998). Postpositivism: the critique of empiricism. *Policy Studies Journal*, 26(1), 129-146.
- Fischer, K. W., & Bidell, T. R. (2006). Dynamic development of action and thought. *Handbook of child psychology*, 1, 313-399.
- Forbes, J., Malan, D. J., Pon-Barry, H., Reges, S., & Sahami, M. (2017, March). Scaling introductory courses using undergraduate teaching assistants. In *Proceedings of the 2017 ACM SIGCSE Technical Symposium on computer science education*, 657-658.
- Forehand, M. (2010). Bloom's Taxonomy from Emerging Perspectives on Learning. *Teaching and Technology*, 26.
- Forišek, M., & Steinová, M. (2012, February). Metaphors and analogies for teaching algorithms. In *Proceedings of the 43rd ACM technical symposium on Computer Science Education*, 15-20.
- Gal-Ezer, J., & Harel, D. (1999). Curriculum and course syllabi for a high-school CS program. *Computer Science Education*, 9(2), 114-147.

- Gall, M. D., Borg, W. R., & Gall, J. P. (1996). *Educational research: An introduction*. Longman Publishing.
- Galpin, V. C., & Sanders, I. D. (2007). Perceptions of computer science at a South African university. *Computers & Education*, 49(4), 1330-1356.
- Goddard, W., & Melville, S. (2004). *Research methodology: An introduction*. Juta and Company Ltd.
- Goldkuhl, G. (2012). Pragmatism vs interpretivism in qualitative information systems research. *European journal of information systems*, 21(2), 135-146.
- Goldman, K., Gross, P., Heeren, C., Herman, G., Kaczmarczyk, L., Loui, M. C., & Zilles, C. (2008). Identifying important and difficult concepts in introductory computing courses using a delphi process. *ACM SIGCSE Bulletin*, 40(1), 256-260.
- Gomes, A., & Mendes, A. J. (2007, September). Learning to program-difficulties and solutions. In *International Conference on Engineering Education–ICEE* (7).
- Gomes, A., & Mendes, A. J. (2007). Problem Solving in Programming. In *PPIG* (p. 18).
- Good, T. L., & Brophy, J. E. (1990). *Educational psychology: A realistic approach*. Longman/Addison Wesley Longman.
- Goodyear, G. E., & Allchin, D. (1998). Statements of teaching philosophy. *To improve the academy*, 17(1), 103-121.
- Govender, I. (2021). Towards understanding information systems students' experience of learning introductory programming: A phenomenographic approach. *Journal of Information Technology Education. Innovations in Practice*, 20, 81.
- Gray, P. S., Williamson, J. B., Karp, D. A., & Dalphin, J. R. (2007). *The research imagination: An introduction to qualitative and quantitative methods*. Cambridge University Press.
- Grix, J. (2018). *The foundations of research*. Bloomsbury Publishing.
- Grover, S., & Basu, S. (2017, March). Measuring student learning in introductory block-based programming: Examining misconceptions of loops, variables, and boolean logic. In *Proceedings of the 2017 ACM SIGCSE technical symposium on computer science education*, 267-272.
- Grover, S., & Pea, R. (2013). Computational thinking in K–12: A review of the state of the field. *Educational researcher*, 42(1), 38-43.
- Guba, E. G. (1990). The paradigm dialog. In *Alternative paradigms conference, Mar, 1989, Indiana U, School of Education, San Francisco, ca, us*. Sage Publications, Inc.
- Guba, E. G., & Lincoln, Y. S. (1994). Competing paradigms in qualitative research. *Handbook of qualitative research*, 2(105), 163-194,.
- Gurka, J. S., & Citrin, W. (1996, September). Testing effectiveness of algorithm animation. In *Proceedings 1996 IEEE Symposium on Visual Languages*, 182-189. IEEE.
- Guzdial, M., & du Boulay, B. (2019). The History of Computing. *The Cambridge handbook of computing education research (2019)*, 11.
- Haajanen, J., Pesonius, M., Sutinen, E., Tarhio, J., Terasvirta, T., & Vanninen, P. (1997, September). Animation of user algorithms on the Web. In *Proceedings. 1997 IEEE Symposium on Visual Languages (Cat. No. 97TB100180)*, 356-363. IEEE.
- Hango, D. W. (2013). *Gender differences in science, technology, engineering, mathematics and computer science (STEM) programs at university*. Ottawa: Statistics Canada= Statistique Canada.



- Henderson, K. A. (2011). Post-positivism and the pragmatics of leisure research. *Leisure Sciences*, 33(4), 341-346.
- Heinonen, A., Lehtelä, B., Hellas, A., & Fagerholm, F. (2023). Synthesizing research on programmers' mental models of programs, tasks and concepts—A systematic literature review. *Information and Software Technology*, 107300.
- Hendrix, R., & Weeks, M. (2018, March). First programming language for high school students. In *Society for Information Technology & Teacher Education International Conference*. Association for the Advancement of Computing in Education (AACE), 1896-1903.
- Horn, M. S., & Jacob, R. J. (2007, February). Designing tangible programming languages for classroom use. In *Proceedings of the 1st international conference on Tangible and embedded interaction*, 159-162.
- Houghton, W. (2004). *Engineering subject centre guide: Learning and teaching theory for engineering academics*. Loughborough University.
- Hu, F., Zekelman, A., Horn, M., & Judd, F. (2015, June). Strawbies: explorations in tangible programming. In *Proceedings of the 14th International Conference on Interaction Design and Children*, 410-413.
- Hundhausen, C. D., Douglas, S. A., & Stasko, J. T. (2002). A meta-study of algorithm visualization effectiveness. *Journal of Visual Languages & Computing*, 13(3), 259-290.
- Hundhausen, C. D., Brown, J. L., & Farley, S. (2006, September). Adding procedures and pointers to the ALVIS algorithm visualization software: a preliminary design. In *Proceedings of the 2006 ACM symposium on Software visualization*, 155-156.
- Idris, M. B., & Ammar, H. (2018). The correlation between Arabic student's English proficiency and their computer programming ability at the university level. *USA International Journal of Managing Public Sector Information and Communication Technologies (IJMPICT)*.
- Islam, N., Shafi Sheikh, G., Fatima, R., & Alvi, F. (2019). A Study of Difficulties of Students in Learning Programming. *Journal of Education & Social Sciences*, 7(2), 38-46.
- Izu, C., Weerasinghe, A., & Pope, C. (2016, August). A study of code design skills in novice programmers using the SOLO taxonomy. In *Proceedings of the 2016 ACM Conference on International Computing Education Research*, 251-259.
- Jiang, Z. (2016). Uncertainty Avoidance—A New Teaching/Learning Method for an Introductory Programming Course. *Computer Education*, 4(4), 52.
- Jimoyiannis, A. (2013). Using SOLO taxonomy to explore students' mental models of the programming variable and the assignment statement. *Themes in Science and Technology Education*, 4(2), 53-74.
- Joshi, M., Manoj, L., Oludayo, O. O., Modiba, M. M., & Virendrakumar, C. B. (2012, September). Automated matchmaking of student skills and academic course requisites. In *Proceedings of the CUBE International Information Technology Conference*, 514-519.
- Kaminski, J. A., Sloutsky, V. M., & Heckler, A. F. (2009). Research Commentary: Concrete Instantiations of Mathematics: A Double-Edged Sword. *Journal for Research in Mathematics Education*, 40(2), 90-93.
- Kandemir, C. M., Kalelioğlu, F., & Gülbahar, Y. (2021). Pedagogy of teaching introductory text-based programming in terms of computational thinking concepts and practices. *Computer Applications in Engineering Education*, 29(1), 29-45.
- Kandlbinder, P. (2014). Constructive alignment in university teaching. *HERDSA News*, 36(3), 5-6.

- Kankam, P. K. (2019). The use of paradigms in information research. *Library & Information Science Research*, 41(2), 85-92.
- Karavirta, V., Korhonen, A., Malmi, L., & Stalnacke, K. (2004, August). Matrixpro-a tool for demonstrating data structures and algorithms ex tempore. In *IEEE International Conference on Advanced Learning Technologies, 2004. Proceedings*, 892-893. IEEE.
- Kaushik, V., & Walsh, C. A. (2019). Pragmatism as a research paradigm and its implications for social work research. *Social sciences*, 8(9), 255.
- Kearney, J., Wood, L., & Zuber-Skerritt, O. (2013). Community-university partnerships: Using participatory action learning and action research (PALAR). *Gateways: International Journal of Community Research and Engagement*, 6, 113-130.
- Kelleher, C., & Pausch, R. (2005). Lowering the barriers to programming: A taxonomy of programming environments and languages for novice programmers. *ACM Computing Surveys (CSUR)*, 37(2), 83-137.
- Kincheloe, J. L., & McLaren, P. (2005). Rethinking critical theory and qualitative research. In N. K. Denzin & Y. S. Lincoln (Eds.), *The Sage handbook of qualitative research*, 303-342. CA: Sage.
- Kohn, T. (2017, March). Variable evaluation: An exploration of novice programmers' understanding and common misconceptions. In *Proceedings of the 2017 ACM SIGCSE Technical Symposium on Computer Science Education*, 345-350.
- Konecki, M., Lovrenčić, S., & Kaniški, M. (2016, May). Using real projects as motivators in programming education. In *2016 39th International Convention on Information and Communication Technology, Electronics and Microelectronics (MIPRO)*, 883-886. IEEE.
- Konvalina, J., Wileman, S. A., & Stephens, L. J. (1983). Math proficiency: A key to success for computer science students. *Communications of the ACM*, 26(5), 377-382.
- Kothari, C. R. (2004). *Research methodology: Methods and techniques*. New Age International.
- Krauss, S. E. (2005). Research paradigms and meaning making: A primer. *The qualitative report*, 10(4), 758-770.
- Krishnamurthi, S., & Fisler, K. (2019). 13 Programming Paradigms and Beyond. *The Cambridge handbook of computing education research*, 377.
- Kumar, A. N., & Raj, R. K. (2022, March). Computer Science Curricula 2023 (CS2023) Community Engagement by the ACM/IEEE-CS/AAAI Joint Task Force. In *Proceedings of the 54th ACM Technical Symposium on Computer Science Education (2)*, 1212-1213.
- Ladislav, H. O. L. Y., & Ellen, R. (1984). Theory, methodology and the research process. *Ethnographic research: A guide to general conduct*, 11-34.
- Lahtinen, E., Ala-Mutka, K., & Järvinen, H.M. (2005). *A study of the difficulties of novice programmers*. Proceeding of ITiCSE'05, ACM: Monte de Caparica.
- Lau, W. W., & Yuen, A. H. (2011). The impact of the medium of instruction: The case of teaching and learning of computer programming. *Education and Information Technologies*, 16(2), 183-201.
- Laurillard, D. (2002, December). Design tools for e-learning. In *Ascilite*, 3-4.
- Laurillard, D. (2013). *Rethinking university teaching: A conversational framework for the effective use of learning technologies*. Routledge.

- Lei, Y., & Allen, M. (2022, February). English Language Learners in Computer Science Education: A Scoping Review. In *Proceedings of the 53rd ACM Technical Symposium on Computer Science Education V. 1*, 57-63.
- Levy, R. B. B., Ben-Ari, M., & Uronen, P. A. (2000, July). An extended experiment with Jeliot 2000. In *Proceedings of the First International Program Visualization Workshop, Porvoo, Finland*, 131-140.
- Levy, R. B. B., Ben-Ari, M., & Uronen, P. A. (2003). The Jeliot 2000 program animation system. *Computers & Education*, 40(1), 1-15.
- Lincoln, Y. S., Lynham, S. A., & Guba, E. G. (2011). Paradigmatic controversies, contradictions, and emerging confluences, revisited. *The Sage handbook of qualitative research*, 4(2), 97-128.
- Lister, R. (2011, December). Concrete and other neo-Piagetian forms of reasoning in the novice programmer. In *Conferences in research and practice in information technology series*.
- Lister, R., Simon, B., Thompson, E., Whalley, J. L., & Prasad, C. (2006). Not seeing the forest for the trees: novice programmers and the SOLO taxonomy. *ACM SIGCSE Bulletin*, 38(3), 118-122.
- Loughlin, C., Lygo-Baker, S., & Lindberg-Sand, Å. (2021). Reclaiming constructive alignment. *European Journal of Higher Education*, 11(2), 119-136.
- Lowe, R. (2001). Beyond “eye-candy”: improving learning with animations. *Apple University Consortium*.
- Lowe, R., & Schnotz, W. (Eds.). (2008). *Learning with animation: Research implications for design*. Cambridge University Press.
- Lund, B. D., & Wang, T. (2023). Chatting about ChatGPT: how may AI and GPT impact academia and libraries? *Library Hi Tech News*.
- Lunn, S., Marques Samary, M., & Peterfreund, A. (2021, March). Where is Computer Science Education Research Happening? In *Proceedings of the 52nd ACM Technical Symposium on Computer Science Education*, 288-294.
- Luxton-Reilly, A., Albluwi, I., Becker, B. A., Giannakos, M., Kumar, A. N., Ott, L., ... & Szabo, C. (2018, July). Introductory programming: a systematic literature review. In *Proceedings Companion of the 23rd Annual ACM Conference on Innovation and Technology in Computer Science Education*, 55-106. ACM.
- Madison, S., & Gifford, J. (2002). Modular programming: novice misconceptions. *Journal of Research on Technology in Education*, 34(3), 217-229.
- Malan, K., & Halland, K. (2004, October). Examples that can do harm in learning programming. In *Companion to the 19th annual ACM SIGPLAN conference on Object-oriented programming systems, languages, and applications*, 83-87.
- Malik, DS. (2018) *C++ Programming, from program analysis to program design*, 5<sup>th</sup> Ed, Cengage learning.
- Malik, S. I., & Coldwell-Neilson, J. (2017). A model for teaching an introductory programming course using ADRI. *Education and Information Technologies*, 22(3), 1089-1120.
- Malik, S. I., Tawafak, R. M., & Shakir, M. (2021). Aligning and assessing teaching approaches with solo taxonomy in a computer programming course. *International Journal of Information and Communication Technology Education (IJICTE)*, 17(4), 1-15.
- Malone, B., Atkison, T., Kosa, M., & Hadlock, F. (2009, October). Pedagogically effective effortless algorithm visualization with a PCIL. In *2009 39th IEEE Frontiers in Education Conference*, 1-6. IEEE.

- Matsuda, H., & Shindo, Y. (2001, August). Effect of using computer graphics animation in programming education. In *Proceedings IEEE International Conference on Advanced Learning Technologies*, 164-165. IEEE.
- Maxcy, S. J. (2003). Pragmatic threads in mixed methods research in the social sciences: The search for multiple modes of inquiry and the end of the philosophy of formalism. *Handbook of mixed methods in social and behavioral research*, 51-89.
- Mayer, R. E. (Ed.). (2005). *The Cambridge handbook of multimedia learning*. Cambridge University Press.
- Mayer, R. E., & Anderson, R. B. (1991). Animations need narrations: An experimental test of a dual-coding hypothesis. *Journal of educational psychology*, 83(4), 484.
- Mayer, R. E., & Moreno, R. (1998). A cognitive theory of multimedia learning: Implications for design principles. *Journal of educational psychology*, 91(2), 358-368.
- Medeiros, R. P., Ramalho, G. L., & Falcão, T. P. (2018). A systematic literature review on teaching and learning introductory programming in higher education. *IEEE Transactions on Education*, 62(2), 77-90.
- Mergel, B. (1998). Instructional design and learning theory. *Educational Communications and Technology*
- Mishra, S., Balan, S., Iyer, S., & Murthy, S. (2014, June). Effect of a 2-week scratch intervention in CS1 on learners with varying prior knowledge. In *Proceedings of the 2014 conference on Innovation & technology in computer science education*, 45-50.
- Mladenović, M., Boljat, I., & Žanko, Ž. (2018). Comparing loops misconceptions in block-based and text-based programming languages at the K-12 level. *Education and Information Technologies*, 23(4), 1483-1500.
- Moape, T. G., Ojo, S. O., & van Wyk, E. A. (2017, October). A framework for appropriate pedagogical use of conceptual metaphor in computing. In *2017 International Conference on Computing Networking and Informatics (ICCNi)*, 1-9. IEEE.
- Mohamed Shuhidan, S., Hamilton, M., & D'Souza, D. (2011, June). Understanding novice programmer difficulties via guided learning. In *Proceedings of the 16th annual joint conference on Innovation and technology in computer science education*, 213-217.
- Moors, L., Luxton-Reilly, A., & Denny, P. (2018, April). Transitioning from block-based to text-based programming languages. In *2018 International Conference on Learning and Teaching in Computing and Engineering (LaTICE)*, 57-64. IEEE.
- Moreno, A., Myller, N., Sutinen, E., & Ben-Ari, M. (2004, May). Visualizing programs with Jeliot 3. In *Proceedings of the working conference on Advanced visual interfaces*, 373-376.
- Moreno, A., Sutinen, E., & Joy, M. (2014, March). Defining and evaluating conflictive animations for programming education: The case of Jeliot ConAn. In *Proceedings of the 45th ACM technical symposium on Computer science education*, 629-634.
- Muhajirah, M. (2020). Basic of Learning Theory:(Behaviorism, Cognitivism, Constructivism, and Humanism). *International Journal of Asian Education*, 1(1), 37-42.
- Muller, D. A., Lee, K. J., & Sharma, M. D. (2008). Coherence or interest: Which is most important in online multimedia learning?. *Australasian Journal of Educational Technology*, 24(2).
- Munasinghe, B., Bell, T., & Robins, A. (2021, February). Teachers' understanding of technical terms in a Computational Thinking curriculum. In *Australasian Computing Education Conference*, 106-114.

- Murray, J. P. (1997). *Successful Faculty Development and Evaluation: The Complete Teaching Portfolio*. ASHE-ERIC Higher Education Report No. 8, 1995. ERIC Clearinghouse on Higher Education, Graduate School of Education and Human Development, George Washington University, One Dupont Circle, Suite 630, Washington DC, 10036-1183.
- Nagowah, L., & Nagowah, S. (2009). A Reflection on the Dominant Learning Theories: Behaviourism, Cognitivism and Constructivism. *International Journal of Learning*, 16(2).
- Newton, P., & Burgess, D. (2008). Exploring types of educational action research: Implications for research validity. *International journal of qualitative methods*, 7(4), 18-30.
- Orlando, L., & Machado, A. (1996). In defence of Piaget's theory: A reply to 10 common criticisms. *Psychological Reviews*, 103, 143-164.
- Ortiz, D., & Greene, J. (2007). Research design: qualitative, quantitative, and mixed methods approaches. *Qualitative Research Journal*, 6(2), 205-208.
- Osman, W. I., & Elmusharaf, M. M. (2014). Effectiveness of combining algorithm and program animation: A case study with data structure course. *Issues in Informing Science and Information Technology*, 11, 155-168.
- Pal, Y. (2016). A framework for scaffolding to teach programming to vernacular medium learners. *Diss. Indian Institute of Technology Bombay*.
- Panhwar, A. H., Ansari, S., & Shah, A. A. (2017). Post-positivism: An effective paradigm for social and educational research. *International Research Journal of Arts & Humanities (IRJAH)*, 45(45).
- Pansiri, J. (2005). Pragmatism: A methodological approach to researching strategic alliances in tourism. *Tourism and Hospitality Planning & Development*, 2(3), 191-206.
- Pea, R. D. (1986). Language-independent conceptual "bugs" in novice programming. *Journal of educational computing research*, 2(1), 25-36.
- Perkins, D. N. (1991). Technology meets constructivism: Do they make a marriage?. In *Constructivism and the technology of instruction*, 45-55. Routledge. Source: Educational Technology, May 1991, 31(5), 18-23. Educational Technology Publications, Inc.
- Piaget, J. (1929). *The child's conception of the world*. London: Kegan Paul, Trench & Trubner.
- Prasad, A., & Chaudhary, K. (2021). Interactive animation and affective teaching and learning in programming courses. In *Advances in Computer, Communication and Computational Sciences*, 613-623. Springer, Singapore.
- Pyott, S., & Sanders, I. (1991). ALEX: an aid to teaching algorithms. *ACM SIGCSE Bulletin*, 23(3), 36-44.
- Qian, Y., & Lehman, J. (2017). Students' misconceptions and other difficulties in introductory programming: A literature review. *ACM Transactions on Computing Education (TOCE)*, 18(1), 1-24.
- Quevedo-Torrero, J. U. (2009, April). Learning theories in computer science education. In *2009 Sixth International Conference on Information Technology: New Generations*, 1634-1635. IEEE.
- Rabai, B. A. (2015). Programming Language Use in US Academia and Industry. *Informatics in Education*, 14(2), 143-160.
- Rajala, T., Laakso, M. J., Kaila, E., & Salakoski, T. (2007, November). VILLE: a language-independent program visualization tool. In *Proceedings of the Seventh Baltic Sea Conference on Computing Education Research*, 88, 151-159.

- Ramaano, T., Ajoodha, R., & Jadhav, A. (2021). Different models relating prior computer experience with performance in first year computer science. *National Research Foundation of South Africa*.
- Ramabu, T. J., & Oberholzer, H. J. (2017, May). Designing and exploring study field recommender system for prospective students. In *2017 IST-Africa Week Conference (IST-Africa)*, 1-8. IEEE.
- Ramabu, T., Sanders, I., & Schoeman, M. A. (2021). Manipulatives for Teaching Introductory Programming to Struggling Students: A Case of Nested-decisions. In *CSEDU (1)*, 505-510.
- Ramabu, T. J., Sanders, I., & Schoeman, M. (2021, May). Teaching and Learning CS1 with an Assist of Manipulatives. In *2021 IST-Africa Conference (IST-Africa)*, 1-8. IEEE.
- Ramabu, T., Sanders, I., & Schoeman, M. (2022, July). Nested-Decider: An Animation Program for Aiding Teaching and Learning of Decisions/Nested Decisions. In *Annual Conference of the Southern African Computer Lecturers' Association*. Cham: Springer International Publishing, 129 – 148.
- Ramalingam, V., LaBelle, D., & Wiedenbeck, S. (2004, June). Self-efficacy and mental models in learning to program. In *Proceedings of the 9th annual SIGCSE conference on Innovation and technology in computer science education*, 171-175.
- Rauchas, S., Rosman, B., Konidaris, G., & Sanders, I. (2006). Language performance at high school and success in first year computer science. *ACM SIGCSE Bulletin*, 38(1), 398-402.
- Rehman, A. A., & Alharthi, K. (2016). An introduction to research paradigms. *International Journal of Educational Investigations*, 3(8), 51-59.
- Resnick, M., Maloney, J., Monroy-Hernández, A., Rusk, N., Eastmond, E., Brennan, K., ... & Kafai, Y. (2009). Scratch: programming for all. *Communications of the ACM*, 52(11), 60-67.
- Richards, K. (2003). *Qualitative inquiry in TESOL*. Springer.
- Ridley, D. (2008). *The literature review: A step-by-step guide for students*. Sage publication.
- Ring, B. A., Giordan, J., & Ransbottom, J. S. (2008). Problem solving through programming: motivating the non-programmer. *Journal of Computing Sciences in Colleges*, 23(3), 61-67.
- Rodger, S. H., Bashford, M., Dyck, L., Hayes, J., Liang, L., Nelson, D., & Qin, H. (2010, June). Enhancing K-12 education with alice programming adventures. In *Proceedings of the fifteenth annual conference on Innovation and technology in computer science education*, 234-238.
- Roller, M. R. (2016). Qualitative Data Analysis.  
<https://www.rollerresearch.com/MRR%20WORKING%20PAPERS/Qualitative%20Data%20Analysis%20May%202020.pdf>
- Rubin, M. J. (2013, March). The effectiveness of live-coding to teach introductory programming. In *Proceeding of the 44th ACM technical symposium on Computer science education*, 651-656. ACM.
- Rubio, M. A. (2019, May). Automatic Categorization of Introductory Programming Students. In *International Joint Conference: 12th International Conference on Computational Intelligence in Security for Information Systems (CISIS 2019) and 10th International Conference on European Transnational Education (ICEUTE 2019)*, 302-311. Springer, Cham.
- Rubio, M. A., Romero-Zaliz, R., Mañoso, C., & de Madrid, A. P. (2015). Closing the gender gap in an introductory programming course. *Computers & Education*, 82, 409-420.

- Ruffini, M. F. (2009). Creating animations in PowerPoint to support student learning and engagement. *EDUCAUSE Quarterly Magazine*, 32(4).
- Rum, S. N. M., & Ismail, M. A. (2017). Metacognitive support accelerates computer assisted learning for novice programmers. *Journal of Educational Technology & Society*, 20(3), 170-181.
- Sadi, M. S., Halder, L., & Saha, S. (2014, February). Variable dependency analysis of a computer program. In *2013 International Conference on Electrical Information and Communication Technology (EICT)*, 1-5. IEEE.
- Sadler, P. M., Sonnert, G., Coyle, H. P., Cook-Smith, N., & Miller, J. L. (2013). The influence of teachers' knowledge on student learning in middle school physical science classrooms. *American Educational Research Journal*, 50(5), 1020-1049.
- Sajaniemi, J., & Kuittinen, M. (2003, June). Program animation based on the roles of variables. In *Proceedings of the 2003 ACM symposium on Software visualization*, 7-17.
- Sajaniemi, J., & Kuittinen, M. (2008). From procedures to objects: A research agenda for the psychology of object-oriented programming education. *Human Technology: An Interdisciplinary Journal on Humans in ICT Environments*.
- Salite, I. (2008). Educational action research for sustainability: Constructing a vision for the future in teacher education. *Journal of Teacher Education for Sustainability*, 10(2008), 5-16.
- SAPeople. (2020, February 10). *South African Coding Project is Heading for Paris for UN Global Event*. Accessed on 23/09/2022 at: <https://www.sapeople.com/2020/02/10/south-african-coding-project-is-heading-for-paris-for-un-global-event/>
- Saunders, M., Lewis & Thornhill (2007). Research methods. *Business Students 4th edition Pearson Education Limited, England*.
- Schmier, L. (1995). *Random Thoughts: The Humanity of Teaching*. Atwood Publishing, PO Box 3185, Madison, WI 53704.
- Schoeman, M., Gelderblom, H., & Muller, H. (2013). Investigating the effect of program visualization on introductory programming in a distance learning environment. *African Journal of Research in Mathematics, Science and Technology Education*, 17(1\_2), 139-151.
- Scholtz, T. L., & Sanders, I. (2010, June). Mental models of recursion: investigating students' understanding of recursion. In *Proceedings of the fifteenth annual conference on Innovation and technology in computer science education*, 103-107.
- Schunk, D. H. (1991). *Learning theories: An educational perspective*. New York: Merrill.
- Seaton, C. (2020). *Understanding Programs Using Graphs*. Accessed on 20/08/2022 at: <https://shopify.engineering/understanding-programs-using-graphs>
- Sevinç, G. (2019). A review on the neo-Piagetian theory of cognitive development. *Ankara University Journal of Faculty of Educational Sciences (JFES)*, 52(2), 611-631.
- Sheth, S., Murphy, C., Ross, K. A., & Shasha, D. (2016, February). A course on programming and problem solving. In *Proceedings of the 47th ACM Technical Symposium on Computing Science Education* (pp. 323-328).
- Shi, N., Min, Z., & Zhang, P. (2017). Effects of visualizing roles of variables with animation and IDE in novice program construction. *Telematics and Informatics*, 34(5), 743-754.
- Shuhidan, S.M. (2012). *Probing the minds of novice programmers through guided learning*, PhD thesis, retrieved July 2013, RMIT University: Australia.
- Siegfried, R. M., Greco, D., Miceli, N., & Siegfried, J. (2012). Whatever happened to Richard Reid's list of first programming languages? *Information Systems Education Journal*, 10(4), 24.

- Sim, J. M., Tacla, C. A., Stadzisz, P. C., & Banaszewski, R. F. (2012). *Notification oriented paradigm (nop) and imperative paradigm: A comparative study*.
- Skemp, R. R. (1976). Relational understanding and instrumental understanding. *Mathematics teaching*, 77(1), 20-26.
- Smith, L. (1992). *Jean Piaget Critical Assessments*. London, New York: Routledge.
- Soloway, E. (1986). Learning to program= learning to construct mechanisms and explanations. *Communications of the ACM*, 29(9), 850-858.
- Sorden, S. D. (2012). The cognitive theory of multimedia learning. *Handbook of educational theories*, 1(2012), 1-22.
- Sorva, J. (2008, November). The same but different students' understandings of primitive and object variables. In *Proceedings of the 8th International Conference on Computing Education Research*, 5-15.
- Sorva, J., & Sirkiä, T. (2010, October). UUhistle: a software tool for visual program simulation. In *Proceedings of the 10th Koli Calling International Conference on Computing Education Research*, 49-54.
- Sorva, J., Karavirta, V., & Malmi, L. (2013). A review of generic program visualization systems for introductory programming education. *ACM Transactions on Computing Education (TOCE)*, 13(4), 1-64.
- Spohrer, J. C., & Soloway, E. (1986). Novice mistakes: Are the folk wisdoms correct? *Communications of the ACM*, 29(7), 624-632.
- Steckler, A., McLeroy, K. R., Goodman, R. M., Bird, S. T., & McCormick, L. (1992). Toward integrating qualitative and quantitative methods: an introduction. *Health education quarterly*, 19(1), 1-8.
- Stein, L. A. (1998). What We Swept Under the Rug: Radically Rethinking CS 1. *Computer Science Education*, 8(2), 118-129.
- Storey, M. A., Wong, K., & Müller, H. A. (2000). How do program understanding tools affect how programmers understand programs?. *Science of Computer Programming*, 36(2-3), 183-207.
- Strauss, A., & Corbin, J. (1998). *Basics of qualitative research techniques*. Sage publications.
- Su, J., & Yang, W. (2023). A systematic review of integrating computational thinking in early childhood education. *Computers and Education Open*, 100122.
- Sullivan, A., Elkin, M., & Bers, M. U. (2015, June). KIBO robot demo: engaging young children in programming and engineering. In *Proceedings of the 14th international conference on interaction design and children*, 418-421.
- Suzuki, H., & Kata, H. (1995). *Interaction-level support for collaborative learning: AlgoBlock—an open programming language*.
- Taha, K. (2012, October). Automatic academic advisor. In *8th International Conference on Collaborative Computing: Networking, Applications and Worksharing (CollaborateCom)*, 262-268. IEEE.
- Tanrikulu, E., & Schaefer, B. C. (2011). The users who touched the ceiling of scratch. *Procedia-Social and Behavioral Sciences*, 28, 764-769.
- Tashakkori, A., & Creswell, J. W. (2007). The new era of mixed methods. *Journal of mixed methods research*, 1(1), 3-7.
- Trigwell, K., & Prosser, M. (2014). Qualitative variation in constructive alignment in curriculum design. *Higher Education*, 67(2), 141-154.



- Tuparov, G., Tuparova, D., & Tsarnakova, A. (2012). Using interactive simulation-based learning objects in introductory course of programming. *Procedia-Social and Behavioral Sciences*, 46, 2276-2280.
- Urquiza-Fuentes, J., & Velázquez-Iturbide, J. Á. (2009). A survey of successful evaluations of program visualization and algorithm animation systems. *ACM Transactions on Computing Education (TOCE)*, 9(2), 1-21.
- Ussiph, N., & Seidu, H. K. (2018). The impact of using 3D interactive animation tool in teaching computer programming at the senior high school level. *Circulation Comput. Sci*, 3(2), 18-28.
- Veerasamy, A. K., & Shillabeer, A. (2014). Teaching English based programming courses to English language learners/non-native speakers of English. *International Proceedings of Economics Development and Research*, 70, 17.
- Veerasamy, A. K., D'Souza, D., & Laakso, M. J. (2016). Identifying novice student programming misconceptions and errors from summative assessments. *Journal of Educational Technology Systems*, 45(1), 50-73.
- Végh, L. (2016). Javascript Library for Developing Interactive Micro-Level Animations for Teaching and Learning Algorithms on One-Dimensional Arrays. *Acta Didactica Napocensia*, 9(2), 23-32.
- Végh, L., & Stoffová, V. (2017). Algorithm animations for teaching and learning the main ideas of basic sortings. *Informatics in Education*, 16(1), 121-140.
- Vilner, T., Zur, E., & Gal-Ezer, J. (2007). Fundamental concepts of CS1: procedural vs. object-oriented paradigm-a case study. *ACM SIGCSE Bulletin*, 39(3), 171-175.
- Wang, P., Bednarik, R., & Moreno, A. (2012, November). During automatic program animation, explanations after animations have greater impact than before animations. In *Proceedings of the 12th Koli calling international conference on computing education research*, 100-109.
- Wang, X., Su, Y., Cheung, S., Wong, E., & Kwong, T. (2013). An exploration of Biggs' constructive alignment in course design and its impact on students' learning approaches. *Assessment & Evaluation in Higher Education*, 38(4), 477-491.
- Watson, C., & Li, F. W. (2014, June). Failure rates in introductory programming revisited. In *Proceedings of the 2014 conference on Innovation & technology in computer science education*, 39-44.
- Weintrop, D., & Wilensky, U. (2017). Comparing block-based and text-based programming in high school computer science classrooms. *ACM Transactions on Computing Education (TOCE)*, 18(1), 1-25.
- Weiss, R. E., Knowlton, D. S., & Morrison, G. R. (2002). Principles for using animation in computer-based instruction: Theoretical heuristics for effective design. *Computers in Human Behavior*, 18(4), 465-477.
- Werth, L. H. (1986). Predicting student performance in a beginning computer science class. *ACM SIGCSE Bulletin*, 18(1), 138-143.
- Whalley, J., Lister, R., Thompson, E., Clear, T., Robbins, P., Kumar, P. A., & Prasad, C. (2006). An Australasian study of reading and comprehension skills in novice programmers, using the Bloom and SOLO taxonomies. In *8th Australasian Computing Education Conference (ACE2006)*, *Australian Computer Science Communications*, 28, 243-252. Australian Computer Society.
- Wolz, U., Leitner, H. H., Malan, D. J., & Maloney, J. (2009, March). Starting with scratch in CS 1. In *Proceedings of the 40th ACM technical symposium on Computer science education*, 2-3.

- Wu, C. C., Dale, N. B., & Bethel, L. J. (1998, March). Conceptual models and cognitive learning styles in teaching recursion. In *Proceedings of the twenty-ninth SIGCSE technical symposium on Computer science education*, 292-296.
- Wyeth, P., & Purchase, H. C. (2002, April). Tangible programming elements for young children. In *CHI'02 extended abstracts on Human factors in computing systems*, 774-775.
- Xie, B., Loksa, D., Nelson, G. L., Davidson, M. J., Dong, D., Kwik, H., ... & Ko, A. J. (2019). A theory of instruction for introductory programming skills. *Computer Science Education*, 29(2-3), 205-253.
- Yuana, R. A., & Maryono, D. (2016, January). Robomind utilization to improve student motivation and concept in learning programming. In *Proceeding of International Conference on Teacher Training and Education 1* (1).
- Žanko, Ž., Mladenović, M., & Boljat, I. (2019). Misconceptions about variables at the K-12 level. *Education and Information Technologies*, 24(2), 1251-1268.
- Zehetmeier, S., Andreitz, I., Erlacher, W., & Rauch, F. (2015). Researching the impact of teacher professional development programmes based on action research, constructivism, and systems theory. *Educational action research*, 23(2), 162-177.

# Appendices

## Appendix A: Ethics clearance

---



### UNISA COLLEGE OF SCIENCE, ENGINEERING AND TECHNOLOGY'S (CSET) ETHICS REVIEW COMMITTEE

2023/01/25

Dear Mr. Tiou James Ramabu

ERC Reference #: **19/TJR/2019/CSET\_SOC**

Name: Tiou James Ramabu

Student #: 55402844

Staff #:

**Decision: Ethics Approval from  
2023/01/25 to 2028/01/25  
Humans involved.**

**This certificate is an amendment  
to the certificate issued On 24  
May 2019.**

**Researcher(s):** Tiou James Ramabu  
48166723@mylife.unisa.ac.za, 012 337 1401

**Supervisor (s):** Prof. I Sanders  
[sandeid@unisa.ac.za](mailto:sandeid@unisa.ac.za)  
Prof. Martha Anna Schoeman  
[schoema@unisa.ac.za](mailto:schoema@unisa.ac.za), 011 670 9178

**Working title of research:**

**An Animation-based Teaching and Learning Framework for Struggling  
Introductory Programming Students.**

**Qualification:** Phd.

Thank you for the application for research ethics clearance by the Unisa College of Science, Engineering and Technology's (CSET) Ethics Review Committee for the above mentioned research. Ethics approval is granted for 5 years.

*The **low risk application** was expedited by the College of Science, Engineering and Technology's (CSET) Ethics Review Committee on 2023/01/25 in compliance with the Unisa Policy on Research Ethics and the Standard Operating Procedure on Research Ethics Risk*



University of South Africa  
Preller Street, Muckleneuk Ridge, City of Tshwane  
PO Box 392 UNISA 0003 South Africa  
Telephone: +27 12 429 3111 Facsimile: +27 12 429 4150  
[www.unisa.ac.za](http://www.unisa.ac.za)

*Assessment. The decision will be tabled at the next Committee meeting for ratification.*

The proposed research may now commence with the provisions that:

1. The researcher will ensure that the research project adheres to the relevant guidelines set out in the Unisa COVID-19 position statement on research ethics attached.
2. The researcher(s) will ensure that the research project adheres to the values and principles expressed in the UNISA Policy on Research Ethics.
3. Any adverse circumstance arising in the undertaking of the research project that is relevant to the ethicality of the study should be communicated in writing to the College of Science, Engineering and Technology's (CSET) Ethics Review Committee.
4. The researcher(s) will conduct the study according to the methods and procedures set out in the approved application.
5. Any changes that can affect the study-related risks for the research participants, particularly in terms of assurances made with regards to the protection of participants' privacy and the confidentiality of the data, should be reported to the Committee in writing, accompanied by a progress report.
6. The researcher will ensure that the research project adheres to any applicable national legislation, professional codes of conduct, institutional guidelines and scientific standards relevant to the specific field of study. Adherence to the following South African legislation is important, if applicable: Protection of Personal Information Act, no 4 of 2013; Children's act no 38 of 2005 and the National Health Act, no 61 of 2003.
7. Only de-identified research data may be used for secondary research purposes in future on condition that the research objectives are similar to those of the original research. Secondary use of identifiable human research data require additional ethics clearance.
8. No field work activities may continue after the expiry date 2028/01/25. Submission of a completed research ethics progress report will constitute an application for renewal of Ethics Research Committee approval.



URERC 25.04.17 - Decision template (V2) - Approve

University of South Africa  
Preller Street, Muckleneuk Ridge, City of Tshwane  
PO Box 392 UNISA 0003 South Africa  
Telephone: +27 12 429 3111 Facsimile: +27 12 429 4150  
[www.unisa.ac.za](http://www.unisa.ac.za)

*Note*

*The reference number 2022/CSET/SOC/031 should be clearly indicated on all forms of communication with the intended research participants, as well as with the Committee.*

Yours sincerely,



---

Dr D Bisschoff  
Chair of School of Computing Ethics Review Subcommittee  
College of Science, Engineering and Technology (CSET)  
E-mail: dbischof@unisa.ac.za  
Tel: (011) 471-2109



---

Dr KT Masumboka  
Acting Director: School of  
Computing College of Science  
Engineering and Technology  
(CSET)  
E-mail: mnkane@unisa.ac.za  
Tel: (011) 670 9104



---

Prof. B Mamba  
Executive Dean  
College of Science Engineering and  
Technology (CSET)  
E-mail: mambabb@unisa.ac.za  
Tel: (011) 670 9230

## Appendix B: Evaluation sheets for decisions/nested decisions – Cycle 3

### Pair 1 (Pre\_nested\_if and Post\_if\_else)

The evaluation outcome for the control group is in Table B.1 and for the experimental group in Table B.2. The content of Table B.3 is the evaluation matrix outcome of the control and the experimental group based on pair 1. Pair 1 represents the correct structure of the if-statement or nested if-statement.

**Table B.1: Evaluation sheet - control group - pair 1**

		s1	s2	s3	s4	s5	s6	s7	s8	s9	s10
<b>P</b>	Pre										
	Post										
<b>U</b>	Pre						x				
	Post						x				
<b>M</b>	Pre	x	x	x		x		x	x		x
	Post			x	x				x		
<b>R</b>	Pre				x					x	
	Post	x	x			x		x		x	x

**Table B.2: Evaluation sheet - experimental group - pair 1**

		s1	s2	s3	s4	s5	s6	s7	s8	s9	s10	s11
<b>P</b>	Pre											
	Post											
<b>U</b>	Pre											x
	Post											
<b>M</b>	Pre	x		x	x		x	x		x		
	Post											x
<b>R</b>	Pre		x			x			x		x	
	Post	x	x	x	x	x	x	x	x	x	x	

**Table B.3: Pair 1 matrix (Pre\_nested\_if and Post\_if\_else)**

<b>P</b>	<b>P→P</b> (Stagnant-no-learning)		<b>U→P</b> (Lower-level-deterioration)		<b>M→P</b> (Intermediate-deterioration+)		<b>R→P</b> (deteriorated++)	
	Con	Exp	Con	Exp	Con	Exp	Con	Exp
	0	0	0	0	0	0	0	0
<b>U</b>	<b>P→U</b> (Lower-level-improvement)		<b>U→U</b> (Stagnant-at-lower-level)		<b>M→U</b> (Intermediate-deterioration)		<b>R→U</b> (deteriorated+)	
	Con	Exp	Con	Exp	Con	Exp	Con	Exp
	0	0	10%	0	0	0	0	0
<b>M</b>	<b>P→M</b> (Intermediate-improvement+)		<b>U→M</b> (Intermediate-improvement)		<b>M→M</b> (Stagnant-at-intermediate-level)		<b>R→M</b> (deteriorated)	
	Con	Exp	Con	Exp	Con	Exp	Con	Exp
	0	0	0	9.1%	20%	0	10%	0
<b>R</b>	<b>P→R</b> (Improved++)		<b>U→R</b> (Improved+)		<b>M→R</b> (Improved)		<b>R→R</b> (Already-improved)	
	Con	Exp	Con	Exp	Con	Exp	Con	Exp
	0	0	0	0	50%	54.5%	10%	36.4%
	<b>P</b>		<b>U</b>		<b>M</b>		<b>R</b>	

## Pair 2 (Pre\_inner\_if\_AND and Post\_if\_AND)

The evaluation outcome for the control group is in Table B.4 and for the experimental group in Table B.5. The content of Table B.6 is the evaluation matrix outcome of the control and the experimental group based on pair 2. Pair 2 refers to using the correct structure of the if-clause that has the “&&” operator or uses alternative method.

**Table B.4: Evaluation sheet - control group - pair 2**

		s1	s2	s3	s4	s5	s6	s7	s8	s9	s10
<b>P</b>	Pre										
	Post										
<b>U</b>	Pre		x			x					
	Post										
<b>M</b>	Pre			x	x		x	x			x
	Post	x	x	x	x			x		x	x
<b>R</b>	Pre	x							x	x	
	Post					x	x		x		

**Table B.5: Evaluation sheet - experimental group- pair 2**

		s1	s2	s3	s4	s5	s6	s7	s8	s9	s10	s11
<b>P</b>	Pre											
	Post											
<b>U</b>	Pre											
	Post											
<b>M</b>	Pre		x	x		x	x			x		x
	Post		x						x			
<b>R</b>	Pre	x			x			x	x		x	
	Post	x		x	x	x	x		x	x	x	x

**Table B.6: Pair 2 matrix (Pre\_inner\_if\_AND and Post\_if\_AND)**

<b>P</b>	<b>P→P</b> (Stagnant-no-learning)		<b>U→P</b> (Lower-level-deterioration)		<b>M→P</b> (Intermediate-deterioration+)		<b>R→P</b> (deteriorated++)	
	<b>Con</b>	<b>Exp</b>	<b>Con</b>	<b>Exp</b>	<b>Con</b>	<b>Exp</b>	<b>Con</b>	<b>Exp</b>
	0	0	0	0	0	0	0	0
<b>U</b>	<b>P→U</b> (Lower-level-improvement)		<b>U→U</b> (Stagnant-at-lower-level)		<b>M→U</b> (Intermediate-deterioration)		<b>R→U</b> (deteriorated+)	
	<b>Con</b>	<b>Exp</b>	<b>Con</b>	<b>Exp</b>	<b>Con</b>	<b>Exp</b>	<b>Con</b>	<b>Exp</b>
	0	0	0	0	0	0	0	0
<b>M</b>	<b>P→M</b> (Intermediate-improvement+)		<b>U→M</b> (Intermediate-improvement)		<b>M→M</b> (Stagnant-at-intermediate-level)		<b>R→M</b> (deteriorated)	
	<b>Con</b>	<b>Exp</b>	<b>Con</b>	<b>Exp</b>	<b>Con</b>	<b>Exp</b>	<b>Con</b>	<b>Exp</b>
	0	0	10%	0	40%	9.1%	20%	9.1%
<b>R</b>	<b>P→R</b> (Improved++)		<b>U→R</b> (Improved+)		<b>M→R</b> (Improved)		<b>R→R</b> (Already-improved)	
	<b>Con</b>	<b>Exp</b>	<b>Con</b>	<b>Exp</b>	<b>Con</b>	<b>Exp</b>	<b>Con</b>	<b>Exp</b>
	0	0	10%	0	10%	45.5%	10%	36.4%
	<b>P</b>		<b>U</b>		<b>M</b>		<b>R</b>	



### Pair 3 (Pre\_outer\_if\_content and Post\_if\_content)

The evaluation outcome for the control group is in Table B.7 and for the experimental group in Table B.8. The content of Table B.9 is the evaluation matrix outcome of the control and the experimental group based on pair 3. Pair 3 means the students' capabilities of placing the right content in the block for when the if-statement evaluates to true.

**Table B.7: Evaluation sheet - control group - pair 3**

		s1	s2	s3	s4	s5	s6	s7	s8	s9	s10
<b>P</b>	Pre										
	Post										
<b>U</b>	Pre					x					x
	Post										
<b>M</b>	Pre	x		x	x		x		x	x	
	Post	x			x	x	x				x
<b>R</b>	Pre		x					x			
	Post		x	x				x	x	x	

**Table B.8: Evaluation sheet - experimental group - pair 3**

		s1	s2	s3	s4	s5	s6	s7	s8	s9	s10	s11
<b>P</b>	Pre											
	Post											
<b>U</b>	Pre				x				x			
	Post											
<b>M</b>	Pre	x	x			x		x			x	
	Post				x							x
<b>R</b>	Pre			x			x			x		x
	Post	x	x	x		x	x	x	x	x	x	

**Table B.9: Pair 3 matrix (Pre\_outer\_if\_content and Post\_if\_content)**

<b>P</b>	<b>P→P</b> (Stagnant-no-learning)	<b>U→P</b> (Lower-level-deterioration)		<b>M→P</b> (Intermediate-deterioration+)		<b>R→P</b> (deteriorated++)		
		Con	Exp	Con	Exp	Con	Exp	Con
	0	0	0	0	0	0	0	0
<b>U</b>	<b>P→U</b> (Lower-level-improvement)	<b>U→U</b> (Stagnant-at-lower-level)		<b>M→U</b> (Intermediate-deterioration)		<b>R→U</b> (deteriorated+)		
		Con	Exp	Con	Exp	Con	Exp	Con
	0	0	0	0	0	0	0	0
<b>M</b>	<b>P→M</b> (Intermediate-improvement+)	<b>U→M</b> (Intermediate-improvement)		<b>M→M</b> (Stagnant-at-intermediate-level)		<b>R→M</b> (deteriorated)		
		Con	Exp	Con	Exp	Con	Exp	Con
	0	0	20%	9%	30%	0%	0	9%
<b>R</b>	<b>P→R</b> (Improved++)	<b>U→R</b> (Improved+)		<b>M→R</b> (Improved)		<b>R→R</b> (Already-improved)		
		Con	Exp	Con	Exp	Con	Exp	Con
	0	0	0	9%	30%	45.5%	20%	27.3%
	<b>P</b>	<b>U</b>		<b>M</b>		<b>R</b>		

### Pair 4 (Pre\_inner\_if\_content and Post\_if\_content)

The evaluation outcome for the control group is in Table B.10 and for the experimental group in Table B.11. The content of Table B.12 is the evaluation matrix outcome of the control and the experimental group based on pair 4. Pair 4 refers to the correct content in the block following the *if-statement* when it evaluates to true.

**Table B.10: Evaluation sheet - control group - pair 4**

		s1	s2	s3	s4	s5	s6	s7	s8	s9	s10
<b>P</b>	Pre										
	Post										
<b>U</b>	Pre										
	Post										
<b>M</b>	Pre		x		x	x	x		x	x	x
	Post	x	x				x	x		x	
<b>R</b>	Pre	x		x				x			
	Post			x	x	x			x		x

**Table B.11: Evaluation sheet - experimental group - pair 4**

		s1	s2	s3	s4	s5	s6	s7	s8	s9	s10	s11
<b>P</b>	Pre											
	Post											
<b>U</b>	Pre							x				
	Post											
<b>M</b>	Pre	x		x	x		x		x		x	x
	Post					x		x				
<b>R</b>	Pre		x			x				x		
	Post	x	x	x	x		x		x	x	x	x

**Table B.12: Pair 4 matrix (Pre\_inner\_if\_content and Post\_if\_content)**

<b>P</b>	<b>P→P</b> (Stagnant-no-learning)		<b>U→P</b> (Lower-level-deterioration)		<b>M→P</b> (Intermediate-deterioration+)		<b>R→P</b> (deteriorated++)	
	<b>Con</b>	<b>Exp</b>	<b>Con</b>	<b>Exp</b>	<b>Con</b>	<b>Exp</b>	<b>Con</b>	<b>Exp</b>
	0	0	0	0	0	0	0	0
<b>U</b>	<b>P→U</b> (Lower-level-improvement)		<b>U→U</b> (Stagnant-at-lower-level)		<b>M→U</b> (Intermediate-deterioration)		<b>R→U</b> (deteriorated+)	
	<b>Con</b>	<b>Exp</b>	<b>Con</b>	<b>Exp</b>	<b>Con</b>	<b>Exp</b>	<b>Con</b>	<b>Exp</b>
	0	0	0	0	0	0	0	0
<b>M</b>	<b>P→M</b> (Intermediate-improvement+)		<b>U→M</b> (Intermediate-improvement)		<b>M→M</b> (Stagnant-at-intermediate-level)		<b>R→M</b> (deteriorated)	
	<b>Con</b>	<b>Exp</b>	<b>Con</b>	<b>Exp</b>	<b>Con</b>	<b>Exp</b>	<b>Con</b>	<b>Exp</b>
	0	0	0	9%	30%	0	20%	9%
<b>R</b>	<b>P→R</b> (Improved++)		<b>U→R</b> (Improved+)		<b>M→R</b> (Improved)		<b>R→R</b> (Already-improved)	
	<b>Con</b>	<b>Exp</b>	<b>Con</b>	<b>Exp</b>	<b>Con</b>	<b>Exp</b>	<b>Con</b>	<b>Exp</b>
	0	0	0	0	40%	63.6%	10%	18.2%
	<b>P</b>		<b>U</b>		<b>M</b>		<b>R</b>	

### Pair 5 (Pre\_outer\_else\_content and Post\_else\_content)

The evaluation outcome for the control group is in Table B.13 and for the experimental group in Table B.14. The content of Table B.15 is the evaluation matrix outcome of the control and the experimental group based on pair 5. Pair 5 is about the correct content in the *else* part of the *if-statement*.

**Table B.13: Evaluation sheet - control group - pair 5**

		s1	s2	s3	s4	s5	s6	s7	s8	s9	s10
<b>P</b>	Pre										
	Post										
<b>U</b>	Pre	x				x					
	Post	x									
<b>M</b>	Pre			x	x				x		
	Post		x	x		x		x			x
<b>R</b>	Pre		x				x	x		x	x
	Post				x		x		x	x	

**Table B.14: Evaluation sheet - experimental group - pair 5**

		s1	s2	s3	s4	s5	s6	s7	s8	s9	s10	s11
<b>P</b>	Pre											
	Post											
<b>U</b>	Pre		x						x			
	Post											
<b>M</b>	Pre	x			x		x			x		x
	Post		x						x			
<b>R</b>	Pre			x		x		x			x	
	Post	x		x	x	x	x	x		x	x	x

**Table B.15: Pair 5 matrix (Pre\_outer\_else\_content and Post\_else\_content)**

<b>P</b>	<b>P→P</b> (Stagnant-no-learning)		<b>U→P</b> (Lower-level-deterioration)		<b>M→P</b> (Intermediate-deterioration+)		<b>R→P</b> (deteriorated++)	
	<b>Con</b>	<b>Exp</b>	<b>Con</b>	<b>Exp</b>	<b>Con</b>	<b>Exp</b>	<b>Con</b>	<b>Exp</b>
	0	0	0	0	0	0	0	0
<b>U</b>	<b>P→U</b> (Lower-level-improvement)		<b>U→U</b> (Stagnant-at-lower-level)		<b>M→U</b> (Intermediate-deterioration)		<b>R→U</b> (deteriorated+)	
	<b>Con</b>	<b>Exp</b>	<b>Con</b>	<b>Exp</b>	<b>Con</b>	<b>Exp</b>	<b>Con</b>	<b>Exp</b>
	0	0	10%	0	0	0	0	0
<b>M</b>	<b>P→M</b> (Intermediate-improvement+)		<b>U→M</b> (Intermediate-improvement)		<b>M→M</b> (Stagnant-at-intermediate-level)		<b>R→M</b> (deteriorated)	
	<b>Con</b>	<b>Exp</b>	<b>Con</b>	<b>Exp</b>	<b>Con</b>	<b>Exp</b>	<b>Con</b>	<b>Exp</b>
	0	0	10%	18.2%	10%	0	30%	0
<b>R</b>	<b>P→R</b> (Improved++)		<b>U→R</b> (Improved+)		<b>M→R</b> (Improved)		<b>R→R</b> (Already-improved)	
	<b>Con</b>	<b>Exp</b>	<b>Con</b>	<b>Exp</b>	<b>Con</b>	<b>Exp</b>	<b>Con</b>	<b>Exp</b>
	0	0	0	0	20%	45.5%	20%	36.4%
	<b>P</b>		<b>U</b>		<b>M</b>		<b>R</b>	

### Pair 6 (Pre\_inner\_else\_content and Post\_else\_content)

The evaluation outcome for the control group is in Table B.16 and for the experimental group in Table B.17. The content of Table B.18 is the evaluation matrix outcome of the control and the experimental group based on pair 6. Pair 6 refers to the correct content of the *else* part of the *if-statement* (in this case an inner *else* part of the pre-teaching exercise).

**Table B.16: Evaluation sheet - control group - pair 6**

		s1	s2	s3	s4	s5	s6	s7	s8	s9	s10
<b>P</b>	Pre										
	Post										
<b>U</b>	Pre		x								
	Post										
<b>M</b>	Pre	x		x			x		x	x	
	Post	x				x	x	x	x		x
<b>R</b>	Pre				x	x		x			x
	Post		x	x	x					x	

**Table B.17: Evaluation sheet - experimental group - pair 6**

		s1	s2	s3	s4	s5	s6	s7	s8	s9	s10	s11
<b>P</b>	Pre											
	Post											
<b>U</b>	Pre				x							x
	Post											
<b>M</b>	Pre		x			x			x		x	
	Post						x			x		
<b>R</b>	Pre	x		x			x	x		x		
	Post	x	x	x	x	x		x	x		x	x

**Table B.18: Pair 6 matrix (Pre\_inner\_else\_content and Post\_else\_content)**

<b>P</b>	<b>P→P</b> (Stagnant-no-learning)		<b>U→P</b> (Lower-level-deterioration)		<b>M→P</b> (Intermediate-deterioration+)		<b>R→P</b> (deteriorated++)	
	Con	Exp	Con	Exp	Con	Exp	Con	Exp
	0	0	0	0	0	0	0	0
<b>U</b>	<b>P→U</b> (Lower-level-improvement)		<b>U→U</b> (Stagnant-at-lower-level)		<b>M→U</b> (Intermediate-deterioration)		<b>R→U</b> (deteriorated+)	
	Con	Exp	Con	Exp	Con	Exp	Con	Exp
	0	0	0	0	0	0	0	0
<b>M</b>	<b>P→M</b> (Intermediate-improvement+)		<b>U→M</b> (Intermediate-improvement)		<b>M→M</b> (Stagnant-at-intermediate-level)		<b>R→M</b> (deteriorated)	
	Con	Exp	Con	Exp	Con	Exp	Con	Exp
	0	0	0	0	30%	0	30%	18.2%
<b>R</b>	<b>P→R</b> (Improved++)		<b>U→R</b> (Improved+)		<b>M→R</b> (Improved)		<b>R→R</b> (Already-improved)	
	Con	Exp	Con	Exp	Con	Exp	Con	Exp
	0	0	10%	18.2%	20%	36.4%	10%	27.3%
	<b>P</b>		<b>U</b>		<b>M</b>		<b>R</b>	

## Appendix C: Evaluation sheets for loops – Cycle 3

### Pair 1 (Pre\_prompt and Post\_prompt)

The evaluation outcome for the control group is in Table C.1 and for the experimental group in Table C.2. The content of Table C.3 is the evaluation matrix outcome of the control and the experimental group based on pair 1. Pair 1 refers to the capability to prompt the values outside the loop.

**Table C.1: Evaluation sheet - control group - pair 1**

		s1	s2	s3	s4	s5	s6	s7	s8	s9	s10	s11	s12	s13	s14	s15
<b>P</b>	Pre															
	Post															
<b>U</b>	Pre							x			x		x			
	Post							x								
<b>M</b>	Pre			x	x											
	Post			x							x					
<b>R</b>	Pre	x	x			x	x		x	x		x		x	x	x
	Post	x	x		x	x	x		x	x		x	x	x	x	x

**Table C.2: Evaluation sheet - experimental group - pair 1**

		s1	s2	s3	s4	s5	s6	s7	s8	s9	s10	s11	s12	s13	s14	s15
<b>P</b>	Pre															
	Post															
<b>U</b>	Pre		x					x				x				
	Post															
<b>M</b>	Pre					x										
	Post		x			x		x				x				
<b>R</b>	Pre	x		x	x		x		x	x	x		x	x	x	x
	Post	x		x	x		x		x	x	x		x	x	x	x

**Table C.3: Pair 1 matrix (Pre\_prompt and Post\_prompt)**

<b>P</b>	<b>P→P</b> (Stagnant-no-learning)		<b>U→P</b> (Lower-level-deterioration)		<b>M→P</b> (Intermediate-deterioration+)		<b>R→P</b> (deteriorated++)	
	Con	Exp	Con	Exp	Con	Exp	Con	Exp
	0	0	0	0	0	0	0	0
<b>U</b>	<b>P→U</b> (Lower-level-improvement)		<b>U→U</b> (Stagnant-at-lower-level)		<b>M→U</b> (Intermediate-deterioration)		<b>R→U</b> (deteriorated+)	
	Con	Exp	Con	Exp	Con	Exp	Con	Exp
	0	0	6.7%	0	0	0	0	0
<b>M</b>	<b>P→M</b> (Intermediate-improvement+)		<b>U→M</b> (Intermediate-improvement)		<b>M→M</b> (Stagnant-at-intermediate-level)		<b>R→M</b> (deteriorated)	
	Con	Exp	Con	Exp	Con	Exp	Con	Exp
	0	0	6.7%	20%	6.7%	6.7%	0	0
<b>R</b>	<b>P→R</b> (Improved++)		<b>U→R</b> (Improved+)		<b>M→R</b> (Improved)		<b>R→R</b> (Already-improved)	
	Con	Exp	Con	Exp	Con	Exp	Con	Exp
	0	0	6.7%	0	6.7%	0	73.3%	80%
	<b>P</b>		<b>U</b>		<b>M</b>		<b>R</b>	

## Pair 2 (Pre\_firstloop\_structure and Post\_loop\_structre)

The evaluation outcome for the control group is in Table C.4 and for the experimental group in Table C.5. The content of Table C.6 is the evaluation matrix outcome of the control and the experimental group based on pair 2. Pair 2 means the correct structure of the first loop (including the condition).

**Table C.4: Evaluation sheet - control group - pair 2**

		s1	s2	s3	s4	s5	s6	s7	s8	s9	s10	s11	s12	s13	s14	s15
<b>P</b>	Pre															
	Post															
<b>U</b>	Pre		x						x		x					
	Post															
<b>M</b>	Pre	x		x	x		x	x		x		x		x	x	x
	Post	x	x	x		x	x		x	x	x	x		x	x	
<b>R</b>	Pre					x								x		
	Post				x			x						x		x

**Table C.5: Evaluation sheet - experimental group - pair 2**

		s1	s2	s3	s4	s5	s6	s7	s8	s9	s10	s11	s12	s13	s14	s15
<b>P</b>	Pre															
	Post															
<b>U</b>	Pre	x			x			x	x	x			x		x	
	Post															
<b>M</b>	Pre		x	x		x	x				x	x		x		x
	Post	x		x	x			x	x	x		x	x		x	
<b>R</b>	Pre															
	Post		x			x	x				x			x		x

**Table C.6: Pair 2 matrix (Pre\_firstloop\_structure and Post\_loop\_structre)**

<b>P</b>	<b>P→P</b> (Stagnant-no-learning)	<b>U→P</b> (Lower-level-deterioration)		<b>M→P</b> (Intermediate-deterioration)		<b>R→P</b> (deteriorated++)			
		Con	Exp	Con	Exp	Con	Exp	Con	Exp
		0	0	0	0	0	0	0	0
<b>U</b>	<b>P→U</b> (Lower-level-improvement)	<b>U→U</b> (Stagnant-at-lower-level)		<b>M→U</b> (Intermediate-deterioration)		<b>R→U</b> (deteriorated+)			
		Con	Exp	Con	Exp	Con	Exp	Con	Exp
		0	0	0	0	0	0	0	0
<b>M</b>	<b>P→M</b> (Intermediate-improvement+)	<b>U→M</b> (Intermediate-improvement)		<b>M→M</b> (Stagnant-at-intermediate-level)		<b>R→M</b> (deteriorated)			
		Con	Exp	Con	Exp	Con	Exp	Con	Exp
		0	0	20%	53.3%	46.7%	6.7%	6.7%	0
<b>R</b>	<b>P→R</b> (Improved++)	<b>U→R</b> (Improved+)		<b>M→R</b> (Improved)		<b>R→R</b> (Already-improved)			
		Con	Exp	Con	Exp	Con	Exp	Con	Exp
		0	0	0	0	20%	40%	6.7%	6.7%
		<b>P</b>		<b>U</b>		<b>M</b>		<b>R</b>	

### Pair 3 (Pre\_firstloop\_content and Post\_loop\_content)

The evaluation outcome for the control group is in Table C.7 and for the experimental group in Table C.8. The content of Table C.9 is the evaluation matrix outcome of the control and the experimental group based on pair 3. Pair 3 means the first loop in the pre-teaching algorithm in relation to the loop in the post-teaching algorithm has the correct content.

**Table C.7: Evaluation sheet - control group - pair 3**

		s1	s2	s3	s4	s5	s6	s7	s8	s9	s10	s11	s12	s13	s14	s15
<b>P</b>	Pre															
	Post															
<b>U</b>	Pre	x			x								x			
	Post															
<b>M</b>	Pre		x			x		x	x	x	x			x	x	
	Post	x			x	x		x		x			x			
<b>R</b>	Pre			x			x					x				x
	Post		x	x			x		x		x	x		x	x	x

**Table C.8: Evaluation sheet - experimental group - pair 3**

		s1	s2	s3	s4	s5	s6	s7	s8	s9	s10	s11	s12	s13	s14	s15
<b>P</b>	Pre															
	Post															
<b>U</b>	Pre		x		x								x			
	Post															
<b>M</b>	Pre	x		x		x	x	x	x	x	x	x		x	x	x
	Post		x		x			x			x		x			
<b>R</b>	Pre															
	Post	x		x		x	x		x	x		x		x	x	x

**Table C.9: Pair 3 matrix (Pre\_firstloop\_content and Post\_loop\_content)**

<b>P</b>	<b>P→P</b> (Stagnant-no-learning)		<b>U→P</b> (Lower-level-deterioration)		<b>M→P</b> (Intermediate-deterioration+)		<b>R→P</b> (deteriorated++)		
	Con	Exp	Con	Exp	Con	Exp	Con	Exp	
	0	0	0	0	0	0	0	0	
<b>U</b>	<b>P→U</b> (Lower-level-improvement)		<b>U→U</b> (Stagnant-at-lower-level)		<b>M→U</b> (Intermediate-deterioration)		<b>R→U</b> (deteriorated+)		
	Con	Exp	Con	Exp	Con	Exp	Con	Exp	
	0	0	0	0	0	0	0	0	
<b>M</b>	<b>P→M</b> (Intermediate-improvement+)		<b>U→M</b> (Intermediate-improvement)		<b>M→M</b> (Stagnant-at-intermediate-level)		<b>R→M</b> (deteriorated)		
	Con	Exp	Con	Exp	Con	Exp	Con	Exp	
	0	0	20%	20%	20%	13.3%	0	0	
<b>R</b>	<b>P→R</b> (Improved++)		<b>U→R</b> (Improved+)		<b>M→R</b> (Improved)		<b>R→R</b> (Already-improved)		
	Con	Exp	Con	Exp	Con	Exp	Con	Exp	
	0	0	0	0	33.3%	66.7%	26.7%	0	
		<b>P</b>		<b>U</b>		<b>M</b>		<b>R</b>	

**Pair 4 (Pre\_secondloop\_structure /Pre\_secondloop\_content and Post\_display)**

The evaluation outcome for the control group is in Table C.10 and for the experimental group in Table C.11. The content of Table C.12 is the evaluation matrix outcome of the control and the experimental group based on pair 4. Pair 4 means that the output of the program is in the correct place.

**Table C.10: Evaluation sheet - control group - pair 4**

		s1	s2	s3	s4	s5	s6	s7	s8	s9	s10	s11	s12	s13	s14	s15
<b>P</b>	Pre															
	Post															
<b>U</b>	Pre		x	x				x						x		
	Post															
<b>M</b>	Pre	x			x	x	x		x		x		x		x	x
	Post	x	x	x	x			x	x					x	x	
<b>R</b>	Pre									x		x				
	Post					x	x			x	x	x	x			x

**Table C.11: Evaluation sheet - experimental group - pair 4**

		s1	s2	s3	s4	s5	s6	s7	s8	s9	s10	s11	s12	s13	s14	s15
<b>P</b>	Pre															
	Post															
<b>U</b>	Pre			x		x			x		x			x		
	Post															
<b>M</b>	Pre		x		x		x	x		x		x	x		x	x
	Post			x		x			x		x			x		
<b>R</b>	Pre	x														
	Post	x	x		x		x	x		x		x	x		x	x

**Table C.12: Pair 4 matrix (Pre\_secondloop\_structure / Pre\_secondloop\_content and Post\_display)**

<b>P</b>	<b>P→P</b> (Stagnant-no-learning)		<b>U→P</b> (Lower-level-deterioration)		<b>M→P</b> (Intermediate-deterioration+)		<b>R→P</b> (deteriorated++)	
	Con	Exp	Con	Exp	Con	Exp	Con	Exp
	0	0	0	0	0	0	0	0
<b>U</b>	<b>P→U</b> (Lower-level-improvement)		<b>U→U</b> (Stagnant-at-lower-level)		<b>M→U</b> (Intermediate-deterioration)		<b>R→U</b> (deteriorated+)	
	Con	Exp	Con	Exp	Con	Exp	Con	Exp
	0	0	0	0	0	0	0	0
<b>M</b>	<b>P→M</b> (Intermediate-improvement+)		<b>U→M</b> (Intermediate-improvement)		<b>M→M</b> (Stagnant-at-intermediate-level)		<b>R→M</b> (deteriorated)	
	Con	Exp	Con	Exp	Con	Exp	Con	Exp
	0	0	26.7%	33.3%	26.7%	0	0	0
<b>R</b>	<b>P→R</b> (Improved++)		<b>U→R</b> (Improved+)		<b>M→R</b> (Improved)		<b>R→R</b> (Already-improved)	
	Con	Exp	Con	Exp	Con	Exp	Con	Exp
	0	0	0	0	33.3%	60%	13.3%	6.7%
	<b>P</b>		<b>U</b>		<b>M</b>		<b>R</b>	



## Appendix D: Evaluation sheets for nested loops – Cycle 3

### Pair 1 (Pre\_outer\_loop\_structure and Post\_outer\_loop\_structure)

The evaluation outcome for the control group is in Table D.1 and for the experimental group in Table D.2. The content of Table D.3 is the evaluation matrix outcome of the control and the experimental group based on pair 1. Pair 1 refers to the correct structure of the outer loop of the nested loop.

**Table D.1: Evaluation sheet - control group - pair 1**

		s1	s2	s3	s4	s5	s6	s7	s8	s9	s10	s11	s12	s13	s14	s15
<b>P</b>	Pre															
	Post															
<b>U</b>	Pre															
	Post															
<b>M</b>	Pre		x					x		x				x		
	Post		x							x				x		
<b>R</b>	Pre	x		x	x	x	x		x		x	x	x		x	x
	Post	x		x	x	x	x	x	x		x	x	x		x	x

**Table D.2: Evaluation sheet - experimental group - pair 1**

		s1	s2	s3	s4	s5	s6	s7	s8	s9	s10	s11	s12	s13	s14	s15
<b>P</b>	Pre															
	Post															
<b>U</b>	Pre				x									x		
	Post															
<b>M</b>	Pre															
	Post				x									x		
<b>R</b>	Pre	x	x	x		x	x	x	x	x	x	x	x		x	x
	Post	x	x	x		x	x	x	x	x	x	x	x		x	x

**Table D.3: Pair 1 matrix (Pre\_outer\_loop\_structure and Post\_outer\_loop\_structure)**

<b>P</b>	<b>P→P</b> (Stagnant-no-learning)		<b>U→P</b> (Lower-level-deterioration)		<b>M→P</b> (Intermediate-deterioration+)		<b>R→P</b> (deteriorated++)	
	Con	Exp	Con	Exp	Con	Exp	Con	Exp
	0	0	0	0	0	0	0	0
<b>U</b>	<b>P→U</b> (Lower-level-improvement)		<b>U→U</b> (Stagnant-at-lower-level)		<b>M→U</b> (Intermediate-deterioration)		<b>R→U</b> (deteriorated+)	
	Con	Exp	Con	Exp	Con	Exp	Con	Exp
	0	0	0	0	0	0	0	0
<b>M</b>	<b>P→M</b> (Intermediate-improvement+)		<b>U→M</b> (Intermediate-improvement)		<b>M→M</b> (Stagnant-at-intermediate-level)		<b>R→M</b> (deteriorated)	
	Con	Exp	Con	Exp	Con	Exp	Con	Exp
	0	0	0	13.3%	20%	0	0	0
<b>R</b>	<b>P→R</b> (Improved++)		<b>U→R</b> (Improved+)		<b>M→R</b> (Improved)		<b>R→R</b> (Already-improved)	
	Con	Exp	Con	Exp	Con	Exp	Con	Exp
	0	0	0	0	6.5%	0	73.3%	86.7%
	<b>P</b>		<b>U</b>		<b>M</b>		<b>R</b>	

## Pair 2 (Pre\_inner\_loop\_structure and Post\_inner\_loop\_structure)

The evaluation outcome for the control group is in Table D.4 and for the experimental group in Table D.5. The content of Table D.6 is the evaluation matrix outcome of the control and the experimental group based on pair 2. Pair 2 requires another loop inside the outer loop with the condition based on the variable of the outer loop or by using an alternative method.

**Table D.4: Evaluation sheet - control group - pair 2**

		s1	s2	s3	s4	s5	s6	s7	s8	s9	s10	s11	s12	s13	s14	s15
<b>P</b>	Pre															
	Post															
<b>U</b>	Pre	x		x		x						x				
	Post															
<b>M</b>	Pre		x		x		x	x	x	x	x		x	x	x	x
	Post	x	x	x	x	x		x	x		x	x	x		x	x
<b>R</b>	Pre															
	Post						x			x				x		

**Table D.5: Evaluation sheet - experimental group - pair 2**

		s1	s2	s3	s4	s5	s6	s7	s8	s9	s10	s11	s12	s13	s14	s15
<b>P</b>	Pre															
	Post															
<b>U</b>	Pre		x	x	x	x			x	x	x			x	x	
	Post															
<b>M</b>	Pre	x					x	x				x	x			x
	Post		x	x	x	x			x	x	x			x	x	
<b>R</b>	Pre															
	Post	x					x	x				x	x			x

**Table D.6: Pair 2 matrix (Pre\_inner\_loop\_structure and Post\_inner\_loop\_structure)**

<b>P</b>	<b>P→P</b> (Stagnant-no-learning)		<b>U→P</b> (Lower-level-deterioration)		<b>M→P</b> (Intermediate-deterioration+)		<b>R→P</b> (deteriorated++)	
	Con	Exp	Con	Exp	Con	Exp	Con	Exp
	0	0	0	0	0	0	0	0
<b>U</b>	<b>P→U</b> (Lower-level-improvement)		<b>U→U</b> (Stagnant-at-lower-level)		<b>M→U</b> (Intermediate-deterioration)		<b>R→U</b> (deteriorated+)	
	Con	Exp	Con	Exp	Con	Exp	Con	Exp
	0	0	0	0	0	0	0	0
<b>M</b>	<b>P→M</b> (Intermediate-improvement+)		<b>U→M</b> (Intermediate-improvement)		<b>M→M</b> (Stagnant-at-intermediate-level)		<b>R→M</b> (deteriorated)	
	Con	Exp	Con	Exp	Con	Exp	Con	Exp
	0	0	26.7%	60%	53.3%	0	0	0
<b>R</b>	<b>P→R</b> (Improved++)		<b>U→R</b> (Improved+)		<b>M→R</b> (Improved)		<b>R→R</b> (Already-improved)	
	Con	Exp	Con	Exp	Con	Exp	Con	Exp
	0	0	0	0	20%	40%	0	0
	<b>P</b>		<b>U</b>		<b>M</b>		<b>R</b>	

### Pair 3 (Pre\_inner\_loop\_content and Post\_inner\_loop\_content)

The evaluation outcome for the control group is in Table D.7 and for the experimental group in Table D.8. The content of Table D.9 is the evaluation matrix outcome of the control and the experimental group based on pair 3. Pair 3 refers to the correct content of the inner loop.

**Table D.7: Evaluation sheet - control group - pair 3**

		s1	s2	s3	s4	s5	s6	s7	s8	s9	s10	s11	s12	s13	s14	s15
<b>P</b>	Pre															
	Post															
<b>U</b>	Pre	x			x				x					x		
	Post															
<b>M</b>	Pre		x	x		x	x	x			x	x	x		x	x
	Post	x			x		x	x	x		x			x	x	
<b>R</b>	Pre									x						
	Post		x	x		x				x		x	x			x

**Table D.8: Evaluation sheet - experimental group - pair 3**

		s1	s2	s3	s4	s5	s6	s7	s8	s9	s10	s11	s12	s13	s14	s15
<b>P</b>	Pre															
	Post															
<b>U</b>	Pre				x		x			x				x		
	Post															
<b>M</b>	Pre	x	x	x		x		x	x		x	x			x	x
	Post				x		x			x				x		
<b>R</b>	Pre												x			
	Post	x	x	x		x		x	x		x	x	x		x	x

**Table D.9: Pair 3 matrix (Pre\_inner\_loop\_content and Post\_inner\_loop\_content)**

<b>P</b>	<b>P→P</b> (Stagnant-no-learning)		<b>U→P</b> (Lower-level-deterioration)		<b>M→P</b> (Intermediate-deterioration+)		<b>R→P</b> (deteriorated++)	
	Con	Exp	Con	Exp	Con	Exp	Con	Exp
	0	0	0	0	0	0	0	0
<b>U</b>	<b>P→U</b> (Lower-level-improvement)		<b>U→U</b> (Stagnant-at-lower-level)		<b>M→U</b> (Intermediate-deterioration)		<b>R→U</b> (deteriorated+)	
	Con	Exp	Con	Exp	Con	Exp	Con	Exp
	0	0	0	0	0	0	0	0
<b>M</b>	<b>P→M</b> (Intermediate-improvement+)		<b>U→M</b> (Intermediate-improvement)		<b>M→M</b> (Stagnant-at-intermediate-level)		<b>R→M</b> (deteriorated)	
	Con	Exp	Con	Exp	Con	Exp	Con	Exp
	0	0	26.7%	13.3%	26.7%	6.5%	0	0
<b>R</b>	<b>P→R</b> (Improved++)		<b>U→R</b> (Improved+)		<b>M→R</b> (Improved)		<b>R→R</b> (Already-improved)	
	Con	Exp	Con	Exp	Con	Exp	Con	Exp
	0	0	0	0	40%	66.7%	6.5%	0
	<b>P</b>		<b>U</b>		<b>M</b>		<b>R</b>	

### Pair 4 (Pre\_outer\_loop\_content and Post\_outer\_loop\_content)

The evaluation outcome for the control group is in Table D.10 and for the experimental group in Table D.11. The content of Table D.12 is the evaluation matrix outcome of the control and the experimental group based on pair 4. Pair 4 means the correct content of the outer loop (the code between the blocks).

**Table D.10: Evaluation sheet - control group - pair 4**

		s1	s2	s3	s4	s5	s6	s7	s8	s9	s10	s11	s12	s13	s14	s15
<b>P</b>	Pre															
	Post															
<b>U</b>	Pre															
	Post															
<b>M</b>	Pre	x	x	x	x	x		x	x	x	x		x	x	x	
	Post	x		x	x	x	x	x	x	x	x		x	x	x	x
<b>R</b>	Pre						x					x				x
	Post		x									x				

**Table D.11: Evaluation sheet - experimental group - pair 4**

		s1	s2	s3	s4	s5	s6	s7	s8	s9	s10	s11	s12	s13	s14	s15
<b>P</b>	Pre															
	Post															
<b>U</b>	Pre				x			x								x
	Post															
<b>M</b>	Pre	x	x	x		x	x			x	x	x	x	x	x	
	Post				x			x			x		x			x
<b>R</b>	Pre								x							
	Post	x	x	x		x	x		x	x		x		x	x	

**Table D.12: Pair 4 matrix (Pre\_outer\_loop\_content and Post\_outer\_loop\_content)**

<b>P</b>	<b>P→P</b> (Stagnant-no-learning)		<b>U→P</b> (Lower-level-deterioration)		<b>M→P</b> (Intermediate-deterioration+)		<b>R→P</b> (deteriorated++)	
	Con	Exp	Con	Exp	Con	Exp	Con	Exp
	0	0	0	0	0	0	0	0
<b>U</b>	<b>P→U</b> (Lower-level-improvement)		<b>U→U</b> (Stagnant-at-lower-level)		<b>M→U</b> (Intermediate-deterioration)		<b>R→U</b> (deteriorated+)	
	Con	Exp	Con	Exp	Con	Exp	Con	Exp
	0	0	0	0	0	0	0	0
<b>M</b>	<b>P→M</b> (Intermediate-improvement+)		<b>U→M</b> (Intermediate-improvement)		<b>M→M</b> (Stagnant-at-intermediate-level)		<b>R→M</b> (deteriorated)	
	Con	Exp	Con	Exp	Con	Exp	Con	Exp
	0	0	6.5%	20%	73%	13.3%	0	0
<b>R</b>	<b>P→R</b> (Improved++)		<b>U→R</b> (Improved+)		<b>M→R</b> (Improved)		<b>R→R</b> (Already-improved)	
	Con	Exp	Con	Exp	Con	Exp	Con	Exp
	0	0	0	0	13.3%	60%	6.5%	6.5%
	<b>P</b>		<b>U</b>		<b>M</b>		<b>R</b>	

## Appendix E: Evaluation sheets for one-dimensional array – Cycle 3

### Pair 1 (Pre\_array\_initialised and Post\_array\_initialised)

The evaluation outcome for the control group is in Table E.1 and for the experimental group in Table E.2. The content of Table E.3 is the evaluation matrix outcome of the control and the experimental group based on pair 1. Pair 1 is about the capability to initialise an array.

**Table E.1: Evaluation sheet - control group - pair 1**

		s1	s2	s3	s4	s5	s6	s7	s8	s9	s10
<b>P</b>	Pre										
	Post										
<b>U</b>	Pre							x			
	Post										
<b>M</b>	Pre	x							x		
	Post	x						x			
<b>R</b>	Pre		x	x	x	x	x			x	x
	Post		x	x	x	x	x		x	x	x

**Table E.2: Evaluation sheet - experimental group - pair 1**

		s1	s2	s3	s4	s5	s6	s7	s8	s9	s10
<b>P</b>	Pre										
	Post										
<b>U</b>	Pre										
	Post										
<b>M</b>	Pre		x				x				
	Post										
<b>R</b>	Pre	x		x	x	x		x	x	x	x
	Post	x	x	x	x	x	x	x	x	x	x

**Table E.3: Pair 1 matrix (Pre\_array\_initialised and Post\_array\_initialised)**

<b>P</b>	<b>P→P</b> (Stagnant-no-learning)		<b>U→P</b> (Lower-level-deterioration)		<b>M→P</b> (Intermediate-deterioration+)		<b>R→P</b> (deteriorated++)	
	Con	Exp	Con	Exp	Con	Exp	Con	Exp
	0	0	0	0	0	0	0	0
<b>U</b>	<b>P→U</b> (Lower-level-improvement)		<b>U→U</b> (Stagnant-at-lower-level)		<b>M→U</b> (Intermediate-deterioration)		<b>R→U</b> (deteriorated+)	
	Con	Exp	Con	Exp	Con	Exp	Con	Exp
	0	0	0	0	0	0	0	0
<b>M</b>	<b>P→M</b> (Intermediate-improvement+)		<b>U→M</b> (Intermediate-improvement)		<b>M→M</b> (Stagnant-at-intermediate-level)		<b>R→M</b> (deteriorated)	
	Con	Exp	Con	Exp	Con	Exp	Con	Exp
	0	0	10%	0	10%	0	0	0
<b>R</b>	<b>P→R</b> (Improved++)		<b>U→R</b> (Improved+)		<b>M→R</b> (Improved)		<b>R→R</b> (Already-improved)	
	Con	Exp	Con	Exp	Con	Exp	Con	Exp
	0	0	0	0	10%	20%	70%	80%
	<b>P</b>		<b>U</b>		<b>M</b>		<b>R</b>	

## Pair 2 (Pre\_loop\_structure and Post\_loop\_structure)

The evaluation outcome for the control group is in Table E.4 and for the experimental group in Table E.5. The content of Table E.6 is the evaluation matrix outcome of the control and the experimental group based on pair 2. Pair 2 refers to the structure of the loop which is controlled by the length of the array.

**Table E.4: Evaluation sheet - control group - pair 2**

		s1	s2	s3	s4	s5	s6	s7	s8	s9	s10
<b>P</b>	Pre										
	Post										
<b>U</b>	Pre										
	Post										
<b>M</b>	Pre		x		x			x	x		x
	Post							x			
<b>R</b>	Pre	x		x		x	x			x	
	Post	x	x	x	x	x	x		x	x	x

**Table E.5: Evaluation sheet - experimental group - pair 2**

		s1	s2	s3	s4	s5	s6	s7	s8	s9	s10
<b>P</b>	Pre										
	Post										
<b>U</b>	Pre				x		x				
	Post										
<b>M</b>	Pre	x	x	x		x		x	x	x	
	Post				x		x				
<b>R</b>	Pre										x
	Post	x	x	x		x		x	x	x	x

**Table E.6: Pair 2 matrix (Pre\_loop\_structure and Post\_loop\_structure)**

<b>P</b>	<b>P→P</b> (Stagnant-no-learning)		<b>U→P</b> (Lower-level-deterioration)		<b>M→P</b> (Intermediate-deterioration+)		<b>R→P</b> (deteriorated++)	
	Con	Exp	Con	Exp	Con	Exp	Con	Exp
	0	0	0	0	0	0	0	0
<b>U</b>	<b>P→U</b> (Lower-level-improvement)		<b>U→U</b> (Stagnant-at-lower-level)		<b>M→U</b> (Intermediate-deterioration)		<b>R→U</b> (deteriorated+)	
	Con	Exp	Con	Exp	Con	Exp	Con	Exp
	0	0	0	0	0	0	0	0
<b>M</b>	<b>P→M</b> (Intermediate-improvement+)		<b>U→M</b> (Intermediate-improvement)		<b>M→M</b> (Stagnant-at-intermediate-level)		<b>R→M</b> (deteriorated)	
	Con	Exp	Con	Exp	Con	Exp	Con	Exp
	0	0	0	20%	10%	0	0	0
<b>R</b>	<b>P→R</b> (Improved++)		<b>U→R</b> (Improved+)		<b>M→R</b> (Improved)		<b>R→R</b> (Already-improved)	
	Con	Exp	Con	Exp	Con	Exp	Con	Exp
	0	0	0	0	40%	70%	50%	10%
	<b>P</b>		<b>U</b>		<b>M</b>		<b>R</b>	

### Pair 3 (Pre\_if\_condition and Post\_if\_condition)

The evaluation outcome for the control group is in Table E.7 and for the experimental group in Table E.8. The content of Table E.9 is the evaluation matrix outcome of the control and the experimental group based on pair 3. Pair 3 refers to the condition of the *if-statement* within the loop, which uses an array.

**Table E.7: Evaluation sheet - control group - pair 3**

		s1	s2	s3	s4	s5	s6	s7	s8	s9	s10
<b>P</b>	Pre										
	Post										
<b>U</b>	Pre						x		x		
	Post										
<b>M</b>	Pre	x		x	x	x		x		x	x
	Post						x		x	x	x
<b>R</b>	Pre		x								
	Post	x	x	x	x	x		x			

**Table E.8: Evaluation sheet - experimental group - pair 3**

		s1	s2	s3	s4	s5	s6	s7	s8	s9	s10
<b>P</b>	Pre										
	Post										
<b>U</b>	Pre							x			
	Post										
<b>M</b>	Pre	x	x	x	x		x		x	x	x
	Post							x			
<b>R</b>	Pre					x					
	Post	x	x	x	x	x	x		x	x	x

**Table E.9: Pair 3 matrix (Pre\_if\_condition and Post\_if\_condition)**

<b>P</b>	<b>P→P</b> (Stagnant-no-learning)		<b>U→P</b> (Lower-level-deterioration)		<b>M→P</b> (Intermediate-deterioration+)		<b>R→P</b> (deteriorated++)	
	Con	Exp	Con	Exp	Con	Exp	Con	Exp
	0	0	0	0	0	0	0	0
<b>U</b>	<b>P→U</b> (Lower-level-improvement)		<b>U→U</b> (Stagnant-at-lower-level)		<b>M→U</b> (Intermediate-deterioration)		<b>R→U</b> (deteriorated+)	
	Con	Exp	Con	Exp	Con	Exp	Con	Exp
	0	0	0	0	0	0	0	0
<b>M</b>	<b>P→M</b> (Intermediate-improvement+)		<b>U→M</b> (Intermediate-improvement)		<b>M→M</b> (Stagnant-at-intermediate-level)		<b>R→M</b> (deteriorated)	
	Con	Exp	Con	Exp	Con	Exp	Con	Exp
	0	0	20%	10%	(20%)	0	0	0
<b>R</b>	<b>P→R</b> (Improved++)		<b>U→R</b> (Improved+)		<b>M→R</b> (Improved)		<b>R→R</b> (Already-improved)	
	Con	Exp	Con	Exp	Con	Exp	Con	Exp
	0	0	0	0	50%	80%	10%	10%
	<b>P</b>		<b>U</b>		<b>M</b>		<b>R</b>	

### Pair 4 (Pre\_if\_content and Post\_if\_content)

The evaluation outcome for the control group is in Table E.10 and for the experimental group in Table E.11. The content of Table E.12 is the evaluation matrix outcome of the control and the experimental group based on pair 4. Pair 4 refers to the correct content of the if-statement.

**Table E.10: Evaluation sheet - control group - pair 4**

		s1	s2	s3	s4	s5	s6	s7	s8	s9	s10
<b>P</b>	Pre										
	Post										
<b>U</b>	Pre								x		x
	Post										
<b>M</b>	Pre			x		x		x			
	Post										
<b>R</b>	Pre	x	x		x		x			x	
	Post	x	x	x	x	x	x	x	x	x	x

**Table E.11: Evaluation sheet - experimental group - pair 4**

		s1	s2	s3	s4	s5	s6	s7	s8	s9	s10
<b>P</b>	Pre										
	Post										
<b>U</b>	Pre					x			x		
	Post										
<b>M</b>	Pre	x	x	x	x		x	x		x	x
	Post					x			x	x	
<b>R</b>	Pre										
	Post	x	x	x	x		x	x			x

**Table E.12: Pair 4 matrix (Pre\_if\_content and Post\_if\_content)**

<b>P</b>	<b>P→P</b> (Stagnant-no-learning)		<b>U→P</b> (Lower-level-deterioration)		<b>M→P</b> (Intermediate-deterioration+)		<b>R→P</b> (deteriorated++)	
	Con	Exp	Con	Exp	Con	Exp	Con	Exp
	0	0	0	0	0	0	0	0
<b>U</b>	<b>P→U</b> (Lower-level-improvement)		<b>U→U</b> (Stagnant-at-lower-level)		<b>M→U</b> (Intermediate-deterioration)		<b>R→U</b> (deteriorated+)	
	Con	Exp	Con	Exp	Con	Exp	Con	Exp
	0	0	0	0	0	0	0	0
<b>M</b>	<b>P→M</b> (Intermediate-improvement+)		<b>U→M</b> (Intermediate-improvement)		<b>M→M</b> (Stagnant-at-intermediate-level)		<b>R→M</b> (deteriorated)	
	Con	Exp	Con	Exp	Con	Exp	Con	Exp
	0	0	20%	20%	0	10%	0	0
<b>R</b>	<b>P→R</b> (Improved++)		<b>U→R</b> (Improved+)		<b>M→R</b> (Improved)		<b>R→R</b> (Already-improved)	
	Con	Exp	Con	Exp	Con	Exp	Con	Exp
	0	0	0	0	30%	70%	50%	0
	<b>P</b>		<b>U</b>		<b>M</b>		<b>R</b>	



### Pair 5 (Pre\_display and Post\_display)

The evaluation outcome for the control group is in Table E.13 and for the experimental group in Table E.14. The content of Table E.15 is the evaluation matrix outcome of the control and the experimental group based on pair 5. Pair 5 refers to the correct display at the correct place.

**Table E.13: Evaluation sheet - control group - pair 5**

		s1	s2	s3	s4	s5	s6	s7	s8	s9	s10
<b>P</b>	Pre										
	Post										
<b>U</b>	Pre	x									
	Post										
<b>M</b>	Pre		x	x	x	x	x	x	x	x	
	Post	x			x		x	x			
<b>R</b>	Pre										x
	Post		x	x		x			x	x	x

**Table E.14: Evaluation sheet - experimental group - pair 5**

		s1	s2	s3	s4	s5	s6	s7	s8	s9	s10
<b>P</b>	Pre										
	Post										
<b>U</b>	Pre										
	Post										
<b>M</b>	Pre	x	x		x	x	x	x	x	x	x
	Post										
<b>R</b>	Pre			x							
	Post	x	x	x	x	x	x	x	x	x	x

**Table E.15: Pair 5 matrix (Pre\_display and Post\_display)**

<b>P</b>	<b>P→P</b> (Stagnant-no-learning)		<b>U→P</b> (Lower-level-deterioration)		<b>M→P</b> (Intermediate-deterioration+)		<b>R→P</b> (deteriorated++)		
	Con	Exp	Con	Exp	Con	Exp	Con	Exp	
	0	0	0	0	0	0	0	0	
<b>U</b>	<b>P→U</b> (Lower-level-improvement)		<b>U→U</b> (Stagnant-at-lower-level)		<b>M→U</b> (Intermediate-deterioration)		<b>R→U</b> (deteriorated+)		
	Con	Exp	Con	Exp	Con	Exp	Con	Exp	
	0	0	0	0	0	0	0	0	
<b>M</b>	<b>P→M</b> (Intermediate-improvement+)		<b>U→M</b> (Intermediate-improvement)		<b>M→M</b> (Stagnant-at-intermediate-level)		<b>R→M</b> (deteriorated)		
	Con	Exp	Con	Exp	Con	Exp	Con	Exp	
	0	0	10%	0	30%	0	0	0	
<b>R</b>	<b>P→R</b> (Improved++)		<b>U→R</b> (Improved+)		<b>M→R</b> (Improved)		<b>R→R</b> (Already-improved)		
	Con	Exp	Con	Exp	Con	Exp	Con	Exp	
	0	0	0	0	50%	90%	20%	0	
		<b>P</b>		<b>U</b>		<b>M</b>		<b>R</b>	

## Appendix F: Evaluation sheets for parallel arrays – Cycle 3

### Pair 1 (Pre\_array\_initialised and Post\_array\_initialised)

The evaluation outcome for the control group is in Table F.1 and for the experimental group in Table F.2. The content of Table F.3 is the evaluation matrix outcome of the control and the experimental group based on pair 1. Pair 1 is about the capability to initialise an array.

**Table F.1: Evaluation sheet - control group - pair 1**

		s1	s2	s3	s4	s5	s6	s7	s8	s9	s10
<b>P</b>	Pre										
	Post										
<b>U</b>	Pre										
	Post										
<b>M</b>	Pre			x							
	Post										
<b>R</b>	Pre	x	x		x	x	x	x	x	x	x
	Post	x	x	x	x	x	x	x	x	x	x

**Table F.2: Evaluation sheet - experimental group - pair 1**

		s1	s2	s3	s4	s5	s6	s7	s8	s9	s10
<b>P</b>	Pre										
	Post										
<b>U</b>	Pre					x					
	Post										
<b>M</b>	Pre									x	
	Post					x					
<b>R</b>	Pre	x	x	x	x		x	x	x		x
	Post	x	x	x	x		x	x	x	x	x

**Table F.3: Pair 1 matrix (Pre\_array\_initialised and Post\_array\_initialised)**

<b>P</b>	<b>P→P</b> (Stagnant-no-learning)		<b>U→P</b> (Lower-level-deterioration)		<b>M→P</b> (Intermediate-deterioration+)		<b>R→P</b> (deteriorated++)	
	Con	Exp	Con	Exp	Con	Exp	Con	Exp
	0	0	0	0	0	0	0	0
<b>U</b>	<b>P→U</b> (Lower-level-improvement)		<b>U→U</b> (Stagnant-at-lower-level)		<b>M→U</b> (Intermediate-deterioration)		<b>R→U</b> (deteriorated+)	
	Con	Exp	Con	Exp	Con	Exp	Con	Exp
	0	0	0	0	0	0	0	0
<b>M</b>	<b>P→M</b> (Intermediate-improvement+)		<b>U→M</b> (Intermediate-improvement)		<b>M→M</b> (Stagnant-at-intermediate-level)		<b>R→M</b> (deteriorated)	
	Con	Exp	Con	Exp	Con	Exp	Con	Exp
	0	0	0	10%	0	0	0	0
<b>R</b>	<b>P→R</b> (Improved++)		<b>U→R</b> (Improved+)		<b>M→R</b> (Improved)		<b>R→R</b> (Already-improved)	
	Con	Exp	Con	Exp	Con	Exp	Con	Exp
	0	0	0	0	10%	10%	90%	80%
	<b>P</b>		<b>U</b>		<b>M</b>		<b>R</b>	

### Pair 2 (Pre\_loop\_structure and Post\_loop\_structure)

The evaluation outcome for the control group is in Table F.4 and for the experimental group in Table F.5. The content of Table F.6 is the evaluation matrix outcome of the control and the experimental group based on pair 2. Pair 2 refers to the structure of the loop which is controlled by the length of the array.

**Table F.3: Evaluation sheet - control group - pair 2**

		s1	s2	s3	s4	s5	s6	s7	s8	s9	s10
<b>P</b>	Pre										
	Post										
<b>U</b>	Pre						x				
	Post										
<b>M</b>	Pre	x		x		x			x	x	x
	Post						x				x
<b>R</b>	Pre		x		x			x			
	Post	x	x	x	x	x		x	x	x	

**Table F.4: Evaluation sheet - experimental group - pair 2**

		s1	s2	s3	s4	s5	s6	s7	s8	s9	s10
<b>P</b>	Pre										
	Post										
<b>U</b>	Pre				x		x		x		
	Post										
<b>M</b>	Pre	x	x	x		x		x		x	x
	Post				x				x		
<b>R</b>	Pre										
	Post	x	x	x		x	x	x		x	x

**Table F.6: Pair 2 matrix (Pre\_loop\_structure and Post\_loop\_structure)**

<b>P</b>	<b>P→P</b> (Stagnant-no-learning)		<b>U→P</b> (Lower-level-deterioration)		<b>M→P</b> (Intermediate-deterioration+)		<b>R→P</b> (deteriorated++)	
	Con	Exp	Con	Exp	Con	Exp	Con	Exp
	0	0	0	0	0	0	0	0
<b>U</b>	<b>P→U</b> (Lower-level-improvement)		<b>U→U</b> (Stagnant-at-lower-level)		<b>M→U</b> (Intermediate-deterioration)		<b>R→U</b> (deteriorated+)	
	Con	Exp	Con	Exp	Con	Exp	Con	Exp
	0	0	0	0	0	0	0	0
<b>M</b>	<b>P→M</b> (Intermediate-improvement+)		<b>U→M</b> (Intermediate-improvement)		<b>M→M</b> (Stagnant-at-intermediate-level)		<b>R→M</b> (deteriorated)	
	Con	Exp	Con	Exp	Con	Exp	Con	Exp
	0	0	10%	20%	10%	0	0	0
<b>R</b>	<b>P→R</b> (Improved++)		<b>U→R</b> (Improved+)		<b>M→R</b> (Improved)		<b>R→R</b> (Already-improved)	
	Con	Exp	Con	Exp	Con	Exp	Con	Exp
	0	0	0	10%	50%	70%	30%	0
	<b>P</b>		<b>U</b>		<b>M</b>		<b>R</b>	

### Pair 3 (Pre\_if\_condition and Post\_if\_condition)

The evaluation outcome for the control group is in Table F.7 and for the experimental group in Table F.8. The content of Table F.9 is the evaluation matrix outcome of the control and the experimental group based on pair 3. Pair 3 refers to the condition of the *if-statement* within the loop, which uses an array.

**Table F.7: Evaluation sheet - control group - pair 3**

		s1	s2	s3	s4	s5	s6	s7	s8	s9	s10
<b>P</b>	Pre										
	Post										
<b>U</b>	Pre	x		x							
	Post										
<b>M</b>	Pre		x		x		x		x	x	x
	Post	x	x	x					x		x
<b>R</b>	Pre					x		x			
	Post				x	x	x	x		x	

**Table F.8: Evaluation sheet - experimental group - pair 3**

		s1	s2	s3	s4	s5	s6	s7	s8	s9	s10
<b>P</b>	Pre										
	Post										
<b>U</b>	Pre		x					x			
	Post										
<b>M</b>	Pre	x		x		x	x		x	x	x
	Post										
<b>R</b>	Pre				x						
	Post	x	x	x	x	x	x	x	x	x	x

**Table F.9: Pair 3 matrix (Pre\_if\_condition and Post\_if\_condition)**

<b>P</b>	<b>P→P</b> (Stagnant-no-learning)		<b>U→P</b> (Lower-level-deterioration)		<b>M→P</b> (Intermediate-deterioration+)		<b>R→P</b> (deteriorated++)	
	Con	Exp	Con	Exp	Con	Exp	Con	Exp
	0	0	0	0	0	0	0	0
<b>U</b>	<b>P→U</b> (Lower-level-improvement)		<b>U→U</b> (Stagnant-at-lower-level)		<b>M→U</b> (Intermediate-deterioration)		<b>R→U</b> (deteriorated+)	
	Con	Exp	Con	Exp	Con	Exp	Con	Exp
	0	0	0	0	0	0	0	0
<b>M</b>	<b>P→M</b> (Intermediate-improvement+)		<b>U→M</b> (Intermediate-improvement)		<b>M→M</b> (Stagnant-at-intermediate-level)		<b>R→M</b> (deteriorated)	
	Con	Exp	Con	Exp	Con	Exp	Con	Exp
	0	0	20%	0	30%	0	0	0
<b>R</b>	<b>P→R</b> (Improved++)		<b>U→R</b> (Improved+)		<b>M→R</b> (Improved)		<b>R→R</b> (Already-improved)	
	Con	Exp	Con	Exp	Con	Exp	Con	Exp
	0	0	0	20%	30%	70%	20%	10%
	<b>P</b>		<b>U</b>		<b>M</b>		<b>R</b>	

**Pair 4 (Pre\_if\_content and Post\_if\_content)**

The evaluation outcome for the control group is in Table F.10 and for the experimental group in Table F.11. The content of Table F.12 is the evaluation matrix outcome of the control and the experimental group based on pair 4. Pair 4 refers to the correct content of the if-statement.

**Table F.10: Evaluation sheet - control group - pair 4**

		s1	s2	s3	s4	s5	s6	s7	s8	s9	s10
<b>P</b>	Pre										
	Post										
<b>U</b>	Pre			x							x
	Post										
<b>M</b>	Pre	x	x			x		x		x	
	Post			x							x
<b>R</b>	Pre				x		x		x		
	Post	x	x		x	x	x	x	x	x	

**Table F.11: Evaluation sheet - experimental group - pair 4**

		s1	s2	s3	s4	s5	s6	s7	s8	s9	s10
<b>P</b>	Pre										
	Post										
<b>U</b>	Pre		x			x	x				
	Post										
<b>M</b>	Pre	x		x	x			x	x	x	x
	Post		x				x				
<b>R</b>	Pre										
	Post	x		x	x	x		x	x	x	x

**Table F.12: Pair 4 matrix (Pre\_if\_content and Post\_if\_content)**

<b>P</b>	<b>P→P</b> (Stagnant-no-learning)		<b>U→P</b> (Lower-level-deterioration)		<b>M→P</b> (Intermediate-deterioration+)		<b>R→P</b> (deteriorated++)	
	Con 0	Exp 0	Con 0	Exp 0	Con 0	Exp 0	Con 0	Exp 0
<b>U</b>	<b>P→U</b> (Lower-level-improvement)		<b>U→U</b> (Stagnant-at-lower-level)		<b>M→U</b> (Intermediate-deterioration)		<b>R→U</b> (deteriorated+)	
	Con 0	Exp 0	Con 0	Exp 0	Con 0	Exp 0	Con 0	Exp 0
<b>M</b>	<b>P→M</b> (Intermediate-improvement+)		<b>U→M</b> (Intermediate-improvement)		<b>M→M</b> (Stagnant-at-intermediate-level)		<b>R→M</b> (deteriorated)	
	Con 0	Exp 0	Con 10%	Exp 20%	Con 10%	Exp 0	Con 0	Exp 0
<b>R</b>	<b>P→R</b> (Improved++)		<b>U→R</b> (Improved+)		<b>M→R</b> (Improved)		<b>R→R</b> (Already-improved)	
	Con 0	Exp 0	Con 0	Exp 10%	Con 50%	Exp 70%	Con 30%	Exp 0
	<b>P</b>		<b>U</b>		<b>M</b>		<b>R</b>	

### Pair 5 (Pre\_display and Post\_display)

The evaluation outcome for the control group is in Table F.13 and for the experimental group in Table F.14. The content of Table F.15 is the evaluation matrix outcome of the control and the experimental group based on pair 5. Pair 5 refers to the correct display at the correct place.

**Table F.13: Evaluation sheet - control group - pair 5**

		s1	s2	s3	s4	s5	s6	s7	s8	s9	s10
<b>P</b>	Pre										
	Post										
<b>U</b>	Pre	x									
	Post										
<b>M</b>	Pre		x	x		x	x	x		x	x
	Post	x	x								x
<b>R</b>	Pre				x				x		
	Post			x	x	x	x	x	x	x	

**Table F.14: Evaluation sheet - experimental group - pair 5**

		s1	s2	s3	s4	s5	s6	s7	s8	s9	s10
<b>P</b>	Pre										
	Post										
<b>U</b>	Pre							x			
	Post										
<b>M</b>	Pre	x	x	x	x	x	x		x	x	x
	Post							x			
<b>R</b>	Pre										
	Post	x	x	x	x	x	x		x	x	x

**Table F.15: Pair 5 matrix (Pre\_display and Post\_display)**

<b>P</b>	<b>P→P</b> (Stagnant-no-learning)		<b>U→P</b> (Lower-level-deterioration)		<b>M→P</b> (Intermediate-deterioration+)		<b>R→P</b> (deteriorated++)	
	Con	Exp	Con	Exp	Con	Exp	Con	Exp
	0	0	0	0	0	0	0	0
<b>U</b>	<b>P→U</b> (Lower-level-improvement)		<b>U→U</b> (Stagnant-at-lower-level)		<b>M→U</b> (Intermediate-deterioration)		<b>R→U</b> (deteriorated+)	
	Con	Exp	Con	Exp	Con	Exp	Con	Exp
	0	0	0	0	0	0	0	0
<b>M</b>	<b>P→M</b> (Intermediate-improvement+)		<b>U→M</b> (Intermediate-improvement)		<b>M→M</b> (Stagnant-at-intermediate-level)		<b>R→M</b> (deteriorated)	
	Con	Exp	Con	Exp	Con	Exp	Con	Exp
	0	0	10%	10%	20%	0	0	0
<b>R</b>	<b>P→R</b> (Improved++)		<b>U→R</b> (Improved+)		<b>M→R</b> (Improved)		<b>R→R</b> (Already-improved)	
	Con	Exp	Con	Exp	Con	Exp	Con	Exp
	0	0	0	0	50%	90%	20	0
	<b>P</b>		<b>U</b>		<b>M</b>		<b>R</b>	

## Appendix G: Evaluation sheets for functions – Cycle 3

### Pair 1 (Pre\_func\_call\_return and Post\_func\_call\_return)

The evaluation outcome for the control group is in Table G.1 and for the experimental group in Table G.2. The content of Table G.3 is the evaluation matrix outcome of the control and the experimental group based on pair 1. Pair 1 refers to the capability to call a function that returns a value.

**Table G.1: Evaluation sheet - control group - pair 1**

		s1	s2	s3	s4	s5	s6	s7	s8	s9	s10	s11	s12	s13	s14	s15	s16	s17
<b>P</b>	Pre																	
	Post																	
<b>U</b>	Pre	x																
	Post																	
<b>M</b>	Pre		x		x	x	x	x			x	x		x		x		x
	Post	x			x		x		x		x		x		x		x	x
<b>R</b>	Pre			x					x	x			x		x		x	
	Post		x	x		x		x		x		x		x		x		

**Table G.2: Evaluation sheet - experimental group - pair 1**

		s1	s2	s3	s4	s5	s6	s7	s8	s9	s10	s11	s12	s13	s14	s15
<b>P</b>	Pre															
	Post															
<b>U</b>	Pre	x		x		x				x			x			x
	Post															
<b>M</b>	Pre		x		x		x	x	x		x	x		x	x	
	Post			x		x							x			x
<b>R</b>	Pre															
	Post	x	x		x		x	x	x	x	x	x		x	x	

**Table G.3: Pair 1 matrix (Pre\_func\_call\_return and Post\_func\_call\_return)**

<b>P</b>	<b>P→P</b> (Stagnant-no-learning)		<b>U→P</b> (Lower-level-deterioration)		<b>M→P</b> (Intermediate-deterioration+)		<b>R→P</b> (deteriorated++)	
	Con	Exp	Con	Exp	Con	Exp	Con	Exp
	0	0	0	0	0	0	0	0
<b>U</b>	<b>P→U</b> (Lower-level-improvement)		<b>U→U</b> (Stagnant-at-lower-level)		<b>M→U</b> (Intermediate-deterioration)		<b>R→U</b> (deteriorated+)	
	Con	Exp	Con	Exp	Con	Exp	Con	Exp
	0	0	0	0	0	0	0	0
<b>M</b>	<b>P→M</b> (Intermediate-improvement+)		<b>U→M</b> (Intermediate-improvement)		<b>M→M</b> (Stagnant-at-intermediate-level)		<b>R→M</b> (deteriorated)	
	Con	Exp	Con	Exp	Con	Exp	Con	Exp
	0	0	6.7%	(3.5%)	23.5%	0	23.5%	0
<b>R</b>	<b>P→R</b> (Improved++)		<b>U→R</b> (Improved+)		<b>M→R</b> (Improved)		<b>R→R</b> (Already-improved)	
	Con	Exp	Con	Exp	Con	Exp	Con	Exp
	0	0	0	13.3%	35.3%	60%	11.8%	0
	<b>P</b>		<b>U</b>		<b>M</b>		<b>R</b>	

## Pair 2 (Pre\_func\_call\_void and Post\_func\_call\_void)

The evaluation outcome for the control group is in Table G.4 and for the experimental group in Table G.5. The content of Table G.6 is the evaluation matrix outcome of the control and the experimental group based on pair 2. Pair 2 refers to the capability to call a function that does not return a value (function of a type *void*).

**Table G.4: Evaluation sheet - control group - pair 2**

		s1	s2	s3	s4	s5	s6	s7	s8	s9	s10	s11	s12	s13	s14	s15	s16	s17
<b>P</b>	Pre																	
	Post																	
<b>U</b>	Pre		x		x						x					x		
	Post																	
<b>M</b>	Pre	x		x		x	x	x	x				x		x		x	x
	Post	x	x	x	x		x	x	x		x		x		x	x	x	
<b>R</b>	Pre											x		x				
	Post					x						x		x				x

**Table G.5: Evaluation sheet - experimental group - pair 2**

		s1	s2	s3	s4	s5	s6	s7	s8	s9	s10	s11	s12	s13	s14	s15
<b>P</b>	Pre															
	Post															
<b>U</b>	Pre	x							x							
	Post															
<b>M</b>	Pre		x	x	x	x	x	x		x	x	x	x	x	x	x
	Post	x				x			x				x			
<b>R</b>	Pre															
	Post		x	x	x		x	x		x	x	x		x	x	x

**Table G.6: Pair 2 matrix (Pre\_func\_call\_void and Post\_func\_call\_void)**

<b>P</b>	<b>P→P</b> (Stagnant-no-learning)		<b>U→P</b> (Lower-level-deterioration)		<b>M→P</b> (Intermediate-deterioration+)		<b>R→P</b> (deteriorated++)	
	Con	Exp	Con	Exp	Con	Exp	Con	Exp
	0	0	0	0	0	0	0	0
<b>U</b>	<b>P→U</b> (Lower-level-improvement)		<b>U→U</b> (Stagnant-at-lower-level)		<b>M→U</b> (Intermediate-deterioration)		<b>R→U</b> (deteriorated+)	
	Con	Exp	Con	Exp	Con	Exp	Con	Exp
	0	0	(5.9%)	0	0	0	0	0
<b>M</b>	<b>P→M</b> (Intermediate-improvement+)		<b>U→M</b> (Intermediate-improvement)		<b>M→M</b> (Stagnant-at-intermediate-level)		<b>R→M</b> (deteriorated)	
	Con	Exp	Con	Exp	Con	Exp	Con	Exp
	0	0	23.3%	13.3%	47.1%	13.3%	0	0
<b>R</b>	<b>P→R</b> (Improved++)		<b>U→R</b> (Improved+)		<b>M→R</b> (Improved)		<b>R→R</b> (Already-improved)	
	Con	Exp	Con	Exp	Con	Exp	Con	Exp
	0	0	0	0	11.8%	73.3%	11.8%	0
	<b>P</b>		<b>U</b>		<b>M</b>		<b>R</b>	



### Pair 3 (Pre\_func\_header\_return and Post\_func\_header\_return)

The evaluation outcome for the control group is in Table G.7 and for the experimental group in Table G.8. The content of Table G.9 is the evaluation matrix outcome of the control and the experimental group based on pair 3. Pair 3 refers to the capability to construct a function header that returns a value with a relevant return type.

**Table G.7: Evaluation sheet - control group - pair 3**

		s1	s2	s3	s4	s5	s6	s7	s8	s9	s10	s11	s12	s13	s14	s15	s16	s17
<b>P</b>	Pre																	
	Post																	
<b>U</b>	Pre			x				x				x						x
	Post																	
<b>M</b>	Pre	x	x			x	x			x	x			x		x		
	Post			x		x		x		x		x						x
<b>R</b>	Pre				x				x				x		x		x	
	Post	x	x		x		x		x		x		x	x	x	x	x	

**Table G.8: Evaluation sheet - experimental group - pair 3**

		s1	s2	s3	s4	s5	s6	s7	s8	s9	s10	s11	s12	s13	s14	s15
<b>P</b>	Pre															
	Post															
<b>U</b>	Pre		x				x		x	x			x		x	
	Post															
<b>M</b>	Pre	x		x	x	x		x			x	x		x		x
	Post		x				x		x	x			x		x	
<b>R</b>	Pre															
	Post	x		x	x	x		x			x	x		x		x

**Table G.9: Pair 3 matrix (Pre\_func\_header\_return and Post\_func\_header\_return)**

<b>P</b>	<b>P→P</b> (Stagnant-no-learning)		<b>U→P</b> (Lower-level-deterioration)		<b>M→P</b> (Intermediate-deterioration+)		<b>R→P</b> (deteriorated++)	
	Con	Exp	Con	Exp	Con	Exp	Con	Exp
	0	0	0	0	0	0	0	0
<b>U</b>	<b>P→U</b> (Lower-level-improvement)		<b>U→U</b> (Stagnant-at-lower-level)		<b>M→U</b> (Intermediate-deterioration)		<b>R→U</b> (deteriorated+)	
	Con	Exp	Con	Exp	Con	Exp	Con	Exp
	0	0	0	0	0	0	0	0
<b>M</b>	<b>P→M</b> (Intermediate-improvement+)		<b>U→M</b> (Intermediate-improvement)		<b>M→M</b> (Stagnant-at-intermediate-level)		<b>R→M</b> (deteriorated)	
	Con	Exp	Con	Exp	Con	Exp	Con	Exp
	0	0	23.5%	40%	11.8%	0	0	0
<b>R</b>	<b>P→R</b> (Improved++)		<b>U→R</b> (Improved+)		<b>M→R</b> (Improved)		<b>R→R</b> (Already-improved)	
	Con	Exp	Con	Exp	Con	Exp	Con	Exp
	0	0	0	0	35.3%	60%	29.4%	0
	<b>P</b>		<b>U</b>		<b>M</b>		<b>R</b>	

### Pair 4 (Pre\_func\_return\_content and Post\_func\_return\_content)

The evaluation outcome for the control group is in Table G.10 and for the experimental group in Table G.11. The content of Table G.12 is the evaluation matrix outcome of the control and the experimental group based on pair 4. Pair 4 refers to the capability to construct the correct content in a function that returns a value.

**Table G.10: Evaluation sheet - control group - pair 4**

		s1	s2	s3	s4	s5	s6	s7	s8	s9	s10	s11	s12	s13	s14	s15	s16	s17
<b>P</b>	Pre																	
	Post																	
<b>U</b>	Pre					x	x					x					x	
	Post																	
<b>M</b>	Pre	x	x	x				x	x	x			x	x	x	x		x
	Post		x	x		x	x	x		x		x		x	x		x	
<b>R</b>	Pre				x						x							
	Post	x			x				x		x		x			x		x

**Table G.11: Evaluation sheet - experimental group - pair 4**

		s1	s2	s3	s4	s5	s6	s7	s8	s9	s10	s11	s12	s13	s14	s15
<b>P</b>	Pre															
	Post															
<b>U</b>	Pre				x		x				x					x
	Post															
<b>M</b>	Pre	x	x	x		x		x	x	x		x	x	x	x	
	Post				x						x			x		x
<b>R</b>	Pre															
	Post	x	x	x		x	x	x	x	x		x	x		x	

**Table G.12: Pair 4 matrix (Pre\_func\_return\_content and Post\_func\_return\_content)**

<b>P</b>	<b>P→P</b> (Stagnant-no-learning)		<b>U→P</b> (Lower-level-deterioration)		<b>M→P</b> (Intermediate-deterioration+)		<b>R→P</b> (deteriorated++)	
	Con	Exp	Con	Exp	Con	Exp	Con	Exp
	0	0	0	0	0	0	0	0
<b>U</b>	<b>P→U</b> (Lower-level-improvement)		<b>U→U</b> (Stagnant-at-lower-level)		<b>M→U</b> (Intermediate-deterioration)		<b>R→U</b> (deteriorated+)	
	Con	Exp	Con	Exp	Con	Exp	Con	Exp
	0	0	0	0	0	0	0	0
<b>M</b>	<b>P→M</b> (Intermediate-improvement+)		<b>U→M</b> (Intermediate-improvement)		<b>M→M</b> (Stagnant-at-intermediate-level)		<b>R→M</b> (deteriorated)	
	Con	Exp	Con	Exp	Con	Exp	Con	Exp
	0	0	23.5%	20%	35.3%	6.7%	0	0
<b>R</b>	<b>P→R</b> (Improved++)		<b>U→R</b> (Improved+)		<b>M→R</b> (Improved)		<b>R→R</b> (Already-improved)	
	Con	Exp	Con	Exp	Con	Exp	Con	Exp
	0	0	0	6.7%	29.4%	66.7%	0	0
	<b>P</b>		<b>U</b>		<b>M</b>		<b>R</b>	

**Pair 5 (Pre\_func\_header\_void and Post\_func\_header\_void)**

The evaluation outcome for the control group is in Table G.13 and for the experimental group in Table G.14. The content of Table G.15 is the evaluation matrix outcome of the control and the experimental group based on pair 5. Pair 5 refers to the capability to construct a function header of a type *void* that does not return a value.

**Table G.13: Evaluation sheet - control group - pair 5**

		s1	s2	s3	s4	s5	s6	s7	s8	s9	s10	s11	s12	s13	s14	s15	s16	s17
<b>P</b>	Pre																	
	Post																	
<b>U</b>	Pre			x		x						x	x		x			x
	Post			x								x						
<b>M</b>	Pre	x	x				x	x		x						x	x	
	Post		x			x		x		x			x		x	x		x
<b>R</b>	Pre				x				x		x			x				
	Post	x			x		x		x		x			x			x	

**Table G.14: Evaluation sheet - experimental group - pair 5**

		s1	s2	s3	s4	s5	s6	s7	s8	s9	s10	s11	s12	s13	s14	s15
<b>P</b>	Pre															
	Post															
<b>U</b>	Pre		x	x	x			x			x		x			x
	Post					x										
<b>M</b>	Pre	x					x		x	x		x		x	x	
	Post		x	x	x			x		x	x		x			x
<b>R</b>	Pre															
	Post	x				x	x		x			x		x	x	

**Table G.15: Pair 5 matrix (Pre\_func\_header\_void and Post\_func\_header\_void)**

<b>P</b>	<b>P→P</b> (Stagnant-no-learning)		<b>U→P</b> (Lower-level-deterioration)		<b>M→P</b> (Intermediate-deterioration+)		<b>R→P</b> (deteriorated++)	
	Con	Exp	Con	Exp	Con	Exp	Con	Exp
	0	0	0	0	0	0	0	0
<b>U</b>	<b>P→U</b> (Lower-level-improvement)		<b>U→U</b> (Stagnant-at-lower-level)		<b>M→U</b> (Intermediate-deterioration)		<b>R→U</b> (deteriorated+)	
	Con	Exp	Con	Exp	Con	Exp	Con	Exp
	0	0	11.8%	0	0	0	0	0
<b>M</b>	<b>P→M</b> (Intermediate-improvement+)		<b>U→M</b> (Intermediate-improvement)		<b>M→M</b> (Stagnant-at-intermediate-level)		<b>R→M</b> (deteriorated)	
	Con	Exp	Con	Exp	Con	Exp	Con	Exp
	0	0	23.5%	40%	23.5%	6.7%	0	0
<b>R</b>	<b>P→R</b> (Improved++)		<b>U→R</b> (Improved+)		<b>M→R</b> (Improved)		<b>R→R</b> (Already-improved)	
	Con	Exp	Con	Exp	Con	Exp	Con	Exp
	0	0	0	6.7%	11.8%	40%	23.5%	0
	<b>P</b>		<b>U</b>		<b>M</b>		<b>R</b>	

### Pair 6 (Pre\_func\_void\_content and Post\_func\_void\_content)

The evaluation outcome for the control group is in Table G.16 and for the experimental group in Table G.17. The content of Table G.18 is the evaluation matrix outcome of the control and the experimental group based on pair 6. Pair 6 refers to the capability to provide the correct content in a function that does not return a value.

**Table G.16: Evaluation sheet - control group - pair 6**

		s1	s2	s3	s4	s5	s6	s7	s8	s9	s10	s11	s12	s13	s14	s15	s16	s17
<b>P</b>	Pre																	
	Post																	
<b>U</b>	Pre	x										x						
	Post																	
<b>M</b>	Pre		x	x	x		x		x		x		x	x	x	x	x	x
	Post	x	x		x		x				x	x			x		x	
<b>R</b>	Pre					x		x		x								
	Post			x		x		x	x	x			x	x		x		x

**Table G.17: Evaluation sheet - experimental group - pair 6**

		s1	s2	s3	s4	s5	s6	s7	s8	s9	s10	s11	s12	s13	s14	s15
<b>P</b>	Pre															
	Post															
<b>U</b>	Pre	x			x		x					x				
	Post															
<b>M</b>	Pre		x	x		x		x	x	x	x		x	x	x	x
	Post	x			x		x					x				
<b>R</b>	Pre															
	Post		x	x		x		x	x	x	x		x	x	x	x

**Table G.18: Pair 6 matrix (Pre\_func\_void\_content and Post\_func\_void\_content)**

<b>P</b>	<b>P→P</b> (Stagnant-no-learning)		<b>U→P</b> (Lower-level-deterioration)		<b>M→P</b> (Intermediate-deterioration+)		<b>R→P</b> (deteriorated++)	
	Con	Exp	Con	Exp	Con	Exp	Con	Exp
	0	0	0	0	0	0	0	0
<b>U</b>	<b>P→U</b> (Lower-level-improvement)		<b>U→U</b> (Stagnant-at-lower-level)		<b>M→U</b> (Intermediate-deterioration)		<b>R→U</b> (deteriorated+)	
	Con	Exp	Con	Exp	Con	Exp	Con	Exp
	0	0	0	0	0	0	0	0
<b>M</b>	<b>P→M</b> (Intermediate-improvement+)		<b>U→M</b> (Intermediate-improvement)		<b>M→M</b> (Stagnant-at-intermediate-level)		<b>R→M</b> (deteriorated)	
	Con	Exp	Con	Exp	Con	Exp	Con	Exp
	0	0	11.8%	26.7%	35.3%	0	0	0
<b>R</b>	<b>P→R</b> (Improved++)		<b>U→R</b> (Improved+)		<b>M→R</b> (Improved)		<b>R→R</b> (Already-improved)	
	Con	Exp	Con	Exp	Con	Exp	Con	Exp
	0	0	0	0	35.3%	73.3%	0	0
	<b>P</b>		<b>U</b>		<b>M</b>		<b>R</b>	

## Appendix H: Evaluation sheets for decisions/nested decisions – Cycle 4

### Pair 1 (Pre\_nested\_if and Post\_nested\_if)

The evaluation outcome for the control group is in Table H.1 and for the experimental group in Table H.2. The content of Table H.3 is the evaluation matrix outcome of the control and the experimental group based on pair 1. Pair 1 refers to the correct structure of the nested *if-statement*.

**Table H.1: Evaluation sheet - control group - pair 1**

		s1	s2	s3	s4	s5	s6	s7	s8	s9	s10	s11	s12
<b>P</b>	Pre												
	Post												
<b>U</b>	Pre								x				x
	Post												
<b>M</b>	Pre	x	x		x	x				x	x		
	Post		x					x	x	x			x
<b>R</b>	Pre			x			x	x				x	
	Post	x		x	x	x	x				x	x	

**Table H.2: Evaluation sheet - experimental group - pair 1**

		s1	s2	s3	s4	s5	s6	s7	s8	s9	s10	s11	s12	s13
<b>P</b>	Pre													
	Post													
<b>U</b>	Pre	x			x		x			x				
	Post													
<b>M</b>	Pre		x	x		x		x	x		x	x	x	x
	Post	x			x		x		x	x			x	
<b>R</b>	Pre													
	Post		x	x		x		x			x	x		x

**Table H.3: Pair 1 matrix (Pre\_nested\_if and Post\_nested\_if)**

<b>P</b>	<b>P→P</b> (Stagnant-no-learning)		<b>U→P</b> (Lower-level-deterioration)		<b>M→P</b> (Intermediate-deterioration+)		<b>R→P</b> (deteriorated++)	
	<b>Con</b>	<b>Exp</b>	<b>Con</b>	<b>Exp</b>	<b>Con</b>	<b>Exp</b>	<b>Con</b>	<b>Exp</b>
	0	0	0	0	0	0	0	0
<b>U</b>	<b>P→U</b> (Lower-level-improvement)		<b>U→U</b> (Stagnant-at-lower-level)		<b>M→U</b> (Intermediate-deterioration)		<b>R→U</b> (deteriorated+)	
	<b>Con</b>	<b>Exp</b>	<b>Con</b>	<b>Exp</b>	<b>Con</b>	<b>Exp</b>	<b>Con</b>	<b>Exp</b>
	0	0	0	0	0	0	0	0
<b>M</b>	<b>P→M</b> (Intermediate-improvement+)		<b>U→M</b> (Intermediate-improvement)		<b>M→M</b> (Stagnant-at-intermediate-level)		<b>R→M</b> (deteriorated)	
	<b>Con</b>	<b>Exp</b>	<b>Con</b>	<b>Exp</b>	<b>Con</b>	<b>Exp</b>	<b>Con</b>	<b>Exp</b>
	0	0	16.7%	30.8%	16.7%	15.4%	8.3%	0
<b>R</b>	<b>P→R</b> (Improved++)		<b>U→R</b> (Improved+)		<b>M→R</b> (Improved)		<b>R→R</b> (Already-improved)	
	<b>Con</b>	<b>Exp</b>	<b>Con</b>	<b>Exp</b>	<b>Con</b>	<b>Exp</b>	<b>Con</b>	<b>Exp</b>
	0	0	0	0	33.3%	53.8%	25%	0
	<b>P</b>		<b>U</b>		<b>M</b>		<b>R</b>	

## Pair 2 (Pre\_inner\_if\_AND and Post\_if\_AND)

The evaluation outcome for the control group is in Table H.4 and for the experimental group in Table H.5. The content of Table H.6 is the evaluation matrix outcome of the control and the experimental group based on pair 2. Pair 2 refers to the correct structure of the if-clause using the "&&" operator or using an alternative method.

**Table H.4: Evaluation sheet - control group - pair 2**

		s1	s2	s3	s4	s5	s6	s7	s8	s9	s10	s11	s12
<b>P</b>	Pre												
	Post												
<b>U</b>	Pre	x										x	
	Post												
<b>M</b>	Pre		x		x		x	x		x	x		x
	Post	x	x					x			x	x	x
<b>R</b>	Pre			x		x			x				
	Post			x	x	x	x		x	x			

**Table H.5: Evaluation sheet - experimental group - pair 2**

		s1	s2	s3	s4	s5	s6	s7	s8	s9	s10	s11	s12	s13
<b>P</b>	Pre													
	Post													
<b>U</b>	Pre			x				x						x
	Post													
<b>M</b>	Pre	x	x		x	x	x			x		x	x	
	Post			x				x						x
<b>R</b>	Pre								x		x			
	Post	x	x		x	x	x		x	x	x	x	x	

**Table H.6: Pair 2 matrix (Pre\_inner\_if\_AND and Post\_if\_AND)**

<b>P</b>	<b>P→P</b> (Stagnant-no-learning)		<b>U→P</b> (Lower-level-deterioration)		<b>M→P</b> (Intermediate-deterioration+)		<b>R→P</b> (deteriorated++)	
	<b>Con</b>	<b>Exp</b>	<b>Con</b>	<b>Exp</b>	<b>Con</b>	<b>Exp</b>	<b>Con</b>	<b>Exp</b>
	0	0	0	0	0	0	0	0
<b>U</b>	<b>P→U</b> (Lower-level-improvement)		<b>U→U</b> (Stagnant-at-lower-level)		<b>M→U</b> (Intermediate-deterioration)		<b>R→U</b> (deteriorated+)	
	<b>Con</b>	<b>Exp</b>	<b>Con</b>	<b>Exp</b>	<b>Con</b>	<b>Exp</b>	<b>Con</b>	<b>Exp</b>
	0	0	0	0	0	0	0	0
<b>M</b>	<b>P→M</b> (Intermediate-improvement+)		<b>U→M</b> (Intermediate-improvement)		<b>M→M</b> (Stagnant-at-intermediate-level)		<b>R→M</b> (deteriorated)	
	<b>Con</b>	<b>Exp</b>	<b>Con</b>	<b>Exp</b>	<b>Con</b>	<b>Exp</b>	<b>Con</b>	<b>Exp</b>
	0	0	16.7%	23.1%	33.3%	0	0	0
<b>R</b>	<b>P→R</b> (Improved++)		<b>U→R</b> (Improved+)		<b>M→R</b> (Improved)		<b>R→R</b> (Already-improved)	
	<b>Con</b>	<b>Exp</b>	<b>Con</b>	<b>Exp</b>	<b>Con</b>	<b>Exp</b>	<b>Con</b>	<b>Exp</b>
	0	0	0	0	25%	61.5%	25%	15.4%
	<b>P</b>		<b>U</b>		<b>M</b>		<b>R</b>	

### Pair 3 (Pre\_outer\_if\_OR and Post\_if\_OR)

The evaluation outcome for the control group is in Table H.7 and for the experimental group in Table H.8. The content of Table H.9 is the evaluation matrix outcome of the control and the experimental group based on pair 3. Pair 3 means the students' ability to use the correct format of comparison in the *if-statement* with the || operator.

**Table H.7: Evaluation sheet - control group - pair 3**

		s1	s2	s3	s4	s5	s6	s7	s8	s9	s10	s11	s12
<b>P</b>	Pre												
	Post												
<b>U</b>	Pre	x	x			x	x			x		x	
	Post	x					x						
<b>M</b>	Pre				x				x				x
	Post		x			x				x		x	
<b>R</b>	Pre			x				x			x		
	Post			x	x			x	x		x		x

**Table H.8: Evaluation sheet - experimental group - pair 3**

		s1	s2	s3	s4	s5	s6	s7	s8	s9	s10	s11	s12	s13
<b>P</b>	Pre													
	Post													
<b>U</b>	Pre				x									
	Post													
<b>M</b>	Pre	x		x		x	x	x	x	x	x	x	x	x
	Post				x					x			x	
<b>R</b>	Pre		x											
	Post	x	x	x		x	x	x	x		x	x		x

**Table H.9: Pair 3 matrix (Pre\_outer\_if\_OR and Post\_if\_OR)**

<b>P</b>	<b>P→P</b> (Stagnant-no-learning)		<b>U→P</b> (Lower-level-deterioration)		<b>M→P</b> (Intermediate-deterioration+)		<b>R→P</b> (deteriorated++)	
	Con	Exp	Con	Exp	Con	Exp	Con	Exp
	0	0	0	0	0	0	0	0
<b>U</b>	<b>P→U</b> (Lower-level-improvement)		<b>U→U</b> (Stagnant-at-lower-level)		<b>M→U</b> (Intermediate-deterioration)		<b>R→U</b> (deteriorated+)	
	Con	Exp	Con	Exp	Con	Exp	Con	Exp
	0	0	16.7%	0	0	0	0	0
<b>M</b>	<b>P→M</b> (Intermediate-improvement+)		<b>U→M</b> (Intermediate-improvement)		<b>M→M</b> (Stagnant-at-intermediate-level)		<b>R→M</b> (deteriorated)	
	Con	Exp	Con	Exp	Con	Exp	Con	Exp
	0	0	33.3%	7.7%	0	15.4%	0	0
<b>R</b>	<b>P→R</b> (Improved++)		<b>U→R</b> (Improved+)		<b>M→R</b> (Improved)		<b>R→R</b> (Already-improved)	
	Con	Exp	Con	Exp	Con	Exp	Con	Exp
	0	0	0	0	25%	69.2%	25%	7.7%
	<b>P</b>		<b>U</b>		<b>M</b>		<b>R</b>	

### Pair 4 (Pre\_inner\_if\_content and Post\_if\_OR\_content)

The evaluation outcome for the control group is in Table H.10 and for the experimental group in Table H.11. The content of Table H.12 is the evaluation matrix outcome of the control and the experimental group based on pair 4. Pair 4 refers to the correct content in the *if-statement* (the code between the blocks).

**Table H.10: Evaluation sheet - control group - pair 4**

		s1	s2	s3	s4	s5	s6	s7	s8	s9	s10	s11	s12
<b>P</b>	Pre												
	Post												
<b>U</b>	Pre	x	x		x	x	x			x		x	
	Post	x	x				x			x			
<b>M</b>	Pre			x				x	x		x		x
	Post				x							x	
<b>R</b>	Pre												
	Post			x		x		x	x		x		x

**Table H.11: Evaluation sheet - experimental group - pair 4**

		s1	s2	s3	s4	s5	s6	s7	s8	s9	s10	s11	s12	s13
<b>P</b>	Pre													
	Post													
<b>U</b>	Pre		x				x				x		x	
	Post													
<b>M</b>	Pre	x		x	x	x		x	x	x		x		x
	Post		x	x			x				x		x	
<b>R</b>	Pre													
	Post	x			x	x		x	x	x		x		x

**Table H.12: Pair 4 matrix (Pre\_inner\_if\_content and Post\_if\_OR\_content)**

<b>P</b>	<b>P→P</b> (Stagnant-no-learning)		<b>U→P</b> (Lower-level-deterioration)		<b>M→P</b> (Intermediate-deterioration+)		<b>R→P</b> (deteriorated++)	
	Con	Exp	Con	Exp	Con	Exp	Con	Exp
	0	0	0	0	0	0	0	0
<b>U</b>	<b>P→U</b> (Lower-level-improvement)		<b>U→U</b> (Stagnant-at-lower-level)		<b>M→U</b> (Intermediate-deterioration)		<b>R→U</b> (deteriorated+)	
	Con	Exp	Con	Exp	Con	Exp	Con	Exp
	0	0	0	0	0	0	0	0
<b>M</b>	<b>P→M</b> (Intermediate-improvement+)		<b>U→M</b> (Intermediate-improvement)		<b>M→M</b> (Stagnant-at-intermediate-level)		<b>R→M</b> (deteriorated)	
	Con	Exp	Con	Exp	Con	Exp	Con	Exp
	0	0	16.7%	30.8%	33.3%	7.7%	0	0
<b>R</b>	<b>P→R</b> (Improved++)		<b>U→R</b> (Improved+)		<b>M→R</b> (Improved)		<b>R→R</b> (Already-improved)	
	Con	Exp	Con	Exp	Con	Exp	Con	Exp
	0	0	8.3%	0	41.7%	61.5%	0	0
	<b>P</b>		<b>U</b>		<b>M</b>		<b>R</b>	



### Pair 5 (Pre\_outer\_else\_content and Post\_else\_content)

The evaluation outcome for the control group is in Table H.13 and for the experimental group in Table H.14. The content of Table H.15 is the evaluation matrix outcome of the control and the experimental group based on pair 5. Pair 5 is about comparing whether the content of the outer *else* part of the *if-statement* is correct.

**Table H.13: Evaluation sheet - control group - pair 5**

		s1	s2	s3	s4	s5	s6	s7	s8	s9	s10	s11	s12
<b>P</b>	Pre												
	Post												
<b>U</b>	Pre	x				x					x	x	
	Post												
<b>M</b>	Pre			x	x			x	x	x			x
	Post	x		x		x		x		x	x	x	
<b>R</b>	Pre		x				x						
	Post		x		x		x		x				x

**Table H.14: Evaluation sheet - experimental group - pair 5**

		s1	s2	s3	s4	s5	s6	s7	s8	s9	s10	s11	s12	s13
<b>P</b>	Pre													
	Post													
<b>U</b>	Pre		x		x					x				x
	Post													
<b>M</b>	Pre	x		x		x	x	x	x		x	x	x	
	Post		x		x	x				x				x
<b>R</b>	Pre													
	Post	x		x			x	x	x		x	x	x	

**Table H.15: Pair 5 matrix (Pre\_outer\_else\_content and Post\_else\_content)**

<b>P</b>	<b>P→P</b> (Stagnant-no-learning)		<b>U→P</b> (Lower-level-deterioration)		<b>M→P</b> (Intermediate-deterioration+)		<b>R→P</b> (deteriorated++)	
	Con	Exp	Con	Exp	Con	Exp	Con	Exp
	0	0	0	0	0	0	0	0
<b>U</b>	<b>P→U</b> (Lower-level-improvement)		<b>U→U</b> (Stagnant-at-lower-level)		<b>M→U</b> (Intermediate-deterioration)		<b>R→U</b> (deteriorated+)	
	Con	Exp	Con	Exp	Con	Exp	Con	Exp
	0	0	33.3%	0	0	0	0	0
<b>M</b>	<b>P→M</b> (Intermediate-improvement+)		<b>U→M</b> (Intermediate-improvement)		<b>M→M</b> (Stagnant-at-intermediate-level)		<b>R→M</b> (deteriorated)	
	Con	Exp	Con	Exp	Con	Exp	Con	Exp
	0	0	0	30.8%	25%	7.7%	0	0
<b>R</b>	<b>P→R</b> (Improved++)		<b>U→R</b> (Improved+)		<b>M→R</b> (Improved)		<b>R→R</b> (Already-improved)	
	Con	Exp	Con	Exp	Con	Exp	Con	Exp
	0	0	0	0	25%	61.5%	16.7%	0
	<b>P</b>		<b>U</b>		<b>M</b>		<b>R</b>	

### Pair 6 (Pre\_inner\_else\_content and Post\_else\_content)

The evaluation outcome for the control group is in Table H.16 and for the experimental group in Table H.17. The content of Table H.18 is the evaluation matrix outcome of the control and the experimental group based on pair 6. Pair 6 is about comparing whether the content of the inner *else* part of the *if-statement* is correct or not.

**Table H.16: Evaluation sheet - control group - pair 6**

		s1	s2	s3	s4	s5	s6	s7	s8	s9	s10	s11	s12
<b>P</b>	Pre												
	Post												
<b>U</b>	Pre				x					x		x	
	Post												
<b>M</b>	Pre	x		x		x	x	x	x		x		x
	Post	x	x		x		x			x	x	x	x
<b>R</b>	Pre		x										
	Post			x		x		x	x				

**Table H.17: Evaluation sheet - experimental group - pair 6**

		s1	s2	s3	s4	s5	s6	s7	s8	s9	s10	s11	s12	s13
<b>P</b>	Pre													
	Post													
<b>U</b>	Pre	x		x							x			
	Post													
<b>M</b>	Pre		x		x	x	x	x	x	x		x	x	x
	Post			x				x						
<b>R</b>	Pre													
	Post	x	x		x	x	x		x	x	x	x	x	x

**Table H.18: Pair 6 matrix (Pre\_inner\_else\_content and Post\_else\_content)**

<b>P</b>	<b>P→P</b> (Stagnant-no-learning)	<b>U→P</b> (Lower-level-deterioration)		<b>M→P</b> (Intermediate-deterioration+)		<b>R→P</b> (deteriorated++)			
		<b>Con</b>	<b>Exp</b>	<b>Con</b>	<b>Exp</b>	<b>Con</b>	<b>Exp</b>	<b>Con</b>	<b>Exp</b>
		0	0	0	0	0	0	0	0
<b>U</b>	<b>P→U</b> (Lower-level-improvement)	<b>U→U</b> (Stagnant-at-lower-level)		<b>M→U</b> (Intermediate-deterioration)		<b>R→U</b> (deteriorated+)			
		<b>Con</b>	<b>Exp</b>	<b>Con</b>	<b>Exp</b>	<b>Con</b>	<b>Exp</b>	<b>Con</b>	<b>Exp</b>
		0	0		0	0	0	0	0
<b>M</b>	<b>P→M</b> (Intermediate-improvement+)	<b>U→M</b> (Intermediate-improvement)		<b>M→M</b> (Stagnant-at-intermediate-level)		<b>R→M</b> (deteriorated)			
		<b>Con</b>	<b>Exp</b>	<b>Con</b>	<b>Exp</b>	<b>Con</b>	<b>Exp</b>	<b>Con</b>	<b>Exp</b>
		0	0	25%	7.7%	33.3%	7.7%	8.3%	0
<b>R</b>	<b>P→R</b> (Improved++)	<b>U→R</b> (Improved+)		<b>M→R</b> (Improved)		<b>R→R</b> (Already-improved)			
		<b>Con</b>	<b>Exp</b>	<b>Con</b>	<b>Exp</b>	<b>Con</b>	<b>Exp</b>	<b>Con</b>	<b>Exp</b>
		0	0	0	15.4%	33.3%	69.2%	0	7.7%
		<b>P</b>		<b>U</b>		<b>M</b>		<b>R</b>	

# Appendix I: Evaluation sheets for decisions / loops – Cycle 4

## Pair 1 matrix (Pre\_loop and Post\_loop)

The evaluation outcome for the control group is in Table I.1 and for the experimental group in Table I.2. The content of Table I.3 is the evaluation matrix outcome of the control and the experimental group based on pair 1. Pair 1 refers to the correct structure of the loop.

**Table I.1: Evaluation sheet - control group - pair 1**

		s1	s2	s3	s4	s5	s6	s7	s8	s9	s10	s11	s12	s13	s14
<b>P</b>	Pre														
	Post														
<b>U</b>	Pre	x											x		
	Post														
<b>M</b>	Pre			x	x		x		x			x		x	x
	Post	x			x			x		x			x		
<b>R</b>	Pre		x			x		x		x	x				
	Post		x	x		x	x		x		x	x		x	x

**Table I.2: Evaluation sheet - experimental group - pair 1**

		s1	s2	s3	s4	s5	s6	s7	s8	s9	s10	s11	s12	s13	s14
<b>P</b>	Pre														
	Post														
<b>U</b>	Pre			x										x	x
	Post														
<b>M</b>	Pre	x	x		x		x	x	x		x		x		
	Post			x										x	x
<b>R</b>	Pre					x				x		x			
	Post	x	x		x	x	x	x	x	x	x	x	x		

**Table I.3: Pair 1 matrix (Pre\_loop and Post\_loop)**

<b>P</b>	<b>P→P</b> (Stagnant-no-learning)		<b>U→P</b> (Lower-level-deterioration)		<b>M→P</b> (Intermediate-deterioration+)		<b>R→P</b> (deteriorated++)	
	Con	Exp	Con	Exp	Con	Exp	Con	Exp
	0	0	0	0	0	0	0	0
<b>U</b>	<b>P→U</b> (Lower-level-improvement)		<b>U→U</b> (Stagnant-at-lower-level)		<b>M→U</b> (Intermediate-deterioration)		<b>R→U</b> (deteriorated+)	
	Con	Exp	Con	Exp	Con	Exp	Con	Exp
	0	0	0	0	0	0	0	0
<b>M</b>	<b>P→M</b> (Intermediate-improvement+)		<b>U→M</b> (Intermediate-improvement)		<b>M→M</b> (Stagnant-at-intermediate-level)		<b>R→M</b> (deteriorated)	
	Con	Exp	Con	Exp	Con	Exp	Con	Exp
	0	0	14.3%	21.4%	7.1%	0	14.3%	0
<b>R</b>	<b>P→R</b> (Improved++)		<b>U→R</b> (Improved+)		<b>M→R</b> (Improved)		<b>R→R</b> (Already-improved)	
	Con	Exp	Con	Exp	Con	Exp	Con	Exp
	0	0	0	0	42.9%	57.1%	21.4%	21.4%
	<b>P</b>		<b>U</b>		<b>M</b>		<b>R</b>	

## Pair 2 (Pre\_loop\_content and Post\_loop\_content)

The evaluation outcome for the control group is in Table I.4 and for the experimental group in Table I.5. The content of Table I.6 is the evaluation matrix outcome of the control and the experimental group based on pair 2. Pair 2 means the general content of the first loop (the code between the blocks).

**Table I.4: Evaluation sheet - control group - pair 2**

		s1	s2	s3	s4	s5	s6	s7	s8	s9	s10	s11	s12	s13	s14
<b>P</b>	Pre														
	Post														
<b>U</b>	Pre		x		x			x			x		x	x	
	Post														
<b>M</b>	Pre	x		x		x	x		x	x		x			
	Post	x	x		x	x		x		x	x		x	x	
<b>R</b>	Pre														x
	Post			x			x		x			x			x

**Table I.5: Evaluation sheet - experimental group - pair 2**

		s1	s2	s3	s4	s5	s6	s7	s8	s9	s10	s11	s12	s13	s14
<b>P</b>	Pre														
	Post														
<b>U</b>	Pre					x			x	x					
	Post														
<b>M</b>	Pre	x	x	x	x		x	x			x	x	x		x
	Post					x			x						
<b>R</b>	Pre													x	
	Post	x	x	x	x		x	x		x	x	x	x	x	x

**Table I.6: Pair 2 matrix (Pre\_loop\_content and Post\_loop\_content)**

<b>P</b>	<b>P→P</b> (Stagnant-no-learning)		<b>U→P</b> (Lower-level-deterioration)		<b>M→P</b> (Intermediate-deterioration+)		<b>R→P</b> (deteriorated++)	
	<b>Con</b>	<b>Exp</b>	<b>Con</b>	<b>Exp</b>	<b>Con</b>	<b>Exp</b>	<b>Con</b>	<b>Exp</b>
	0	0	0	0	0	0	0	0
<b>U</b>	<b>P→U</b> (Lower-level-improvement)		<b>U→U</b> (Stagnant-at-lower-level)		<b>M→U</b> (Intermediate-deterioration)		<b>R→U</b> (deteriorated+)	
	<b>Con</b>	<b>Exp</b>	<b>Con</b>	<b>Exp</b>	<b>Con</b>	<b>Exp</b>	<b>Con</b>	<b>Exp</b>
	0	0	0	0	0	0	0	0
<b>M</b>	<b>P→M</b> (Intermediate-improvement+)		<b>U→M</b> (Intermediate-improvement)		<b>M→M</b> (Stagnant-at-intermediate-level)		<b>R→M</b> (deteriorated)	
	<b>Con</b>	<b>Exp</b>	<b>Con</b>	<b>Exp</b>	<b>Con</b>	<b>Exp</b>	<b>Con</b>	<b>Exp</b>
	0	0	42.9%	14.3%	21.4%	0	0	0
<b>R</b>	<b>P→R</b> (Improved++)		<b>U→R</b> (Improved+)		<b>M→R</b> (Improved)		<b>R→R</b> (Already-improved)	
	<b>Con</b>	<b>Exp</b>	<b>Con</b>	<b>Exp</b>	<b>Con</b>	<b>Exp</b>	<b>Con</b>	<b>Exp</b>
	0	0	0	7.1%	28.6%	71.4%	7.1%	7.1%
	<b>P</b>		<b>U</b>		<b>M</b>		<b>R</b>	

### Pair 3 (Pre\_if\_OR and Post\_if\_OR)

The evaluation outcome for the control group is in Table I.7 and for the experimental group in Table I.8. The content of Table I.9 is the evaluation matrix outcome of the control and the experimental group based on pair 3. Pair 3 refers to the correct format of the condition in an *if-statement* using the || operator.

**Table I.7: Evaluation sheet - control group - pair 3**

		s1	s2	s3	s4	s5	s6	s7	s8	s9	s10	s11	s12	s13	s14
<b>P</b>	Pre														
	Post														
<b>U</b>	Pre			x					x						
	Post														
<b>M</b>	Pre	x				x	x	x		x	x			x	x
	Post	x	x	x	x		x	x	x	x			x	x	x
<b>R</b>	Pre		x		x							x	x		
	Post					x					x	x			

**Table I.8: Evaluation sheet - experimental group - pair 3**

		s1	s2	s3	s4	s5	s6	s7	s8	s9	s10	s11	s12	s13	s14
<b>P</b>	Pre														
	Post														
<b>U</b>	Pre		x		x	x						x			x
	Post														
<b>M</b>	Pre			x			x	x	x	x	x		x	x	
	Post		x		x	x	x					x			x
<b>R</b>	Pre	x													
	Post	x		x				x	x	x	x		x	x	

**Table I.9: Pair 3 matrix (Pre\_if\_OR and Post\_if\_OR)**

<b>P</b>	<b>P→P</b> (Stagnant-no-learning)		<b>U→P</b> (Lower-level-deterioration)		<b>M→P</b> (Intermediate-deterioration+)		<b>R→P</b> (deteriorated++)	
	<b>Con</b>	<b>Exp</b>	<b>Con</b>	<b>Exp</b>	<b>Con</b>	<b>Exp</b>	<b>Con</b>	<b>Exp</b>
	0	0	0	0	0	0	0	0
<b>U</b>	<b>P→U</b> (Lower-level-improvement)		<b>U→U</b> (Stagnant-at-lower-level)		<b>M→U</b> (Intermediate-deterioration)		<b>R→U</b> (deteriorated+)	
	<b>Con</b>	<b>Exp</b>	<b>Con</b>	<b>Exp</b>	<b>Con</b>	<b>Exp</b>	<b>Con</b>	<b>Exp</b>
	0	0	0	0	0	0	0	0
<b>M</b>	<b>P→M</b> (Intermediate-improvement+)		<b>U→M</b> (Intermediate-improvement)		<b>M→M</b> (Stagnant-at-intermediate-level)		<b>R→M</b> (deteriorated)	
	<b>Con</b>	<b>Exp</b>	<b>Con</b>	<b>Exp</b>	<b>Con</b>	<b>Exp</b>	<b>Con</b>	<b>Exp</b>
	0	0	14.3%	35.7%	42.9%	7.1%	21.4%	0
<b>R</b>	<b>P→R</b> (Improved++)		<b>U→R</b> (Improved+)		<b>M→R</b> (Improved)		<b>R→R</b> (Already-improved)	
	<b>Con</b>	<b>Exp</b>	<b>Con</b>	<b>Exp</b>	<b>Con</b>	<b>Exp</b>	<b>Con</b>	<b>Exp</b>
	0	0	0	0	14.3%	50%	7.1%	7.1%
	<b>P</b>		<b>U</b>		<b>M</b>		<b>R</b>	

### Pair 4 (Pre\_if\_OR\_content and Post\_if\_OR\_content)

The evaluation outcome for the control group is in Table I.10 and for the experimental group in Table I.11. The content of Table I.12 is the evaluation matrix outcome of the control and the experimental group based on pair 4. Pair 4 refers to the correct content of the *if-statement*.

**Table I.10: Evaluation sheet - control group - pair 4**

		s1	s2	s3	s4	s5	s6	s7	s8	s9	s10	s11	s12	s13	s14
<b>P</b>	Pre														
	Post														
<b>U</b>	Pre										x				
	Post														
<b>M</b>	Pre	x	x	x	x	x	x	x	x	x				x	
	Post			x				x			x				x
<b>R</b>	Pre											x	x		x
	Post	x	x		x	x	x		x	x		x	x	x	

**Table I.11: Evaluation sheet - experimental group - pair 4**

		s1	s2	s3	s4	s5	s6	s7	s8	s9	s10	s11	s12	s13	s14
<b>P</b>	Pre														
	Post														
<b>U</b>	Pre						x								
	Post														
<b>M</b>	Pre	x	x	x	x	x		x	x	x	x		x	x	x
	Post						x								
<b>R</b>	Pre											x			
	Post	x	x	x	x	x		x	x	x	x	x	x	x	x

**Table I.12: Pair 4 matrix (Pre\_if\_OR\_content and Post\_if\_OR\_content)**

<b>P</b>	<b>P→P</b> (Stagnant-no-learning)		<b>U→P</b> (Lower-level-deterioration)		<b>M→P</b> (Intermediate-deterioration+)		<b>R→P</b> (deteriorated++)	
	<b>Con</b>	<b>Exp</b>	<b>Con</b>	<b>Exp</b>	<b>Con</b>	<b>Exp</b>	<b>Con</b>	<b>Exp</b>
	0	0	0	0	0	0	0	0
<b>U</b>	<b>P→U</b> (Lower-level-improvement)		<b>U→U</b> (Stagnant-at-lower-level)		<b>M→U</b> (Intermediate-deterioration)		<b>R→U</b> (deteriorated+)	
	<b>Con</b>	<b>Exp</b>	<b>Con</b>	<b>Exp</b>	<b>Con</b>	<b>Exp</b>	<b>Con</b>	<b>Exp</b>
	0	0	0	0	0	0	0	0
<b>M</b>	<b>P→M</b> (Intermediate-improvement+)		<b>U→M</b> (Intermediate-improvement)		<b>M→M</b> (Stagnant-at-intermediate-level)		<b>R→M</b> (deteriorated)	
	<b>Con</b>	<b>Exp</b>	<b>Con</b>	<b>Exp</b>	<b>Con</b>	<b>Exp</b>	<b>Con</b>	<b>Exp</b>
	0	0	7.1%	7.1%	14.3%	0	7.1%	0
<b>R</b>	<b>P→R</b> (Improved++)		<b>U→R</b> (Improved+)		<b>M→R</b> (Improved)		<b>R→R</b> (Already-improved)	
	<b>Con</b>	<b>Exp</b>	<b>Con</b>	<b>Exp</b>	<b>Con</b>	<b>Exp</b>	<b>Con</b>	<b>Exp</b>
	0	0	0	0	57.1%	85.7%	14.3%	7.1%
	<b>P</b>		<b>U</b>		<b>M</b>		<b>R</b>	

### Pair 5 (Pre\_if\_OR\_2 and Post\_if\_OR)

The evaluation outcome for the control group is in Table I.13 and for the experimental group in Table I.14. The content of Table I.15 is the evaluation matrix outcome of the control and the experimental group based on pair 5. Pair 5 means the correct format of the condition in the second *if-statement* using the || operator.

**Table I.13: Evaluation sheet - control group - pair 5**

		s1	s2	s3	s4	s5	s6	s7	s8	s9	s10	s11	s12	s13	s14
<b>P</b>	Pre														
	Post														
<b>U</b>	Pre						x				x	x		x	
	Post												x		
<b>M</b>	Pre	x		x	x	x		x		x			x		x
	Post		x	x	x		x	x		x	x	x		x	
<b>R</b>	Pre		x						x						
	Post	x				x			x						x

**Table I.14: Evaluation sheet - experimental group - pair 5**

		s1	s2	s3	s4	s5	s6	s7	s8	s9	s10	s11	s12	s13	s14
<b>P</b>	Pre														
	Post														
<b>U</b>	Pre		x	x			x			x					x
	Post														
<b>M</b>	Pre	x			x	x		x	x		x	x	x	x	
	Post		x	x	x		x			x					x
<b>R</b>	Pre														
	Post	x				x		x	x		x	x	x	x	

**Table I.15: Pair 5 matrix (Pre\_if\_OR\_2 and Post\_if\_OR)**

<b>P</b>	<b>P→P</b> (Stagnant-no-learning)	<b>U→P</b> (Lower-level-deterioration)		<b>M→P</b> (Intermediate-deterioration+)		<b>R→P</b> (deteriorated++)			
		<b>Con</b>	<b>Exp</b>	<b>Con</b>	<b>Exp</b>	<b>Con</b>	<b>Exp</b>	<b>Con</b>	<b>Exp</b>
		0	0	0	0	0	0	0	0
<b>U</b>	<b>P→U</b> (Lower-level-improvement)	<b>U→U</b> (Stagnant-at-lower-level)		<b>M→U</b> (Intermediate-deterioration)		<b>R→U</b> (deteriorated+)			
		<b>Con</b>	<b>Exp</b>	<b>Con</b>	<b>Exp</b>	<b>Con</b>	<b>Exp</b>	<b>Con</b>	<b>Exp</b>
		0	0	0	0	7.1%	0	0	0
<b>M</b>	<b>P→M</b> (Intermediate-improvement+)	<b>U→M</b> (Intermediate-improvement)		<b>M→M</b> (Stagnant-at-intermediate-level)		<b>R→M</b> (deteriorated)			
		<b>Con</b>	<b>Exp</b>	<b>Con</b>	<b>Exp</b>	<b>Con</b>	<b>Exp</b>	<b>Con</b>	<b>Exp</b>
		0	0	28.6%	35.7%	28.6%	7.1%	7.1%	0
<b>R</b>	<b>P→R</b> (Improved++)	<b>U→R</b> (Improved+)		<b>M→R</b> (Improved)		<b>R→R</b> (Already-improved)			
		<b>Con</b>	<b>Exp</b>	<b>Con</b>	<b>Exp</b>	<b>Con</b>	<b>Exp</b>	<b>Con</b>	<b>Exp</b>
		0	0	0	0	21.4%	57.1%	7.1%	0
		<b>P</b>		<b>U</b>		<b>M</b>		<b>R</b>	

**Pair 6 (Pre\_if\_OR\_2\_content and Post\_if\_OR\_content)**

The evaluation outcome for the control group is in Table I.16 and for the experimental group in Table I.17. The content of Table I.18 is the evaluation matrix outcome of the control and the experimental group based on pair 6. Pair 6 refers to the correct content of the *if-statement*.

**Table I.16: Evaluation sheet - control group - pair 6**

		s1	s2	s3	s4	s5	s6	s7	s8	s9	s10	s11	s12	s13	s14
<b>P</b>	Pre														
	Post														
<b>U</b>	Pre	x	x		x				x				x		x
	Post														
<b>M</b>	Pre			x		x	x	x		x	x	x		x	
	Post	x	x				x		x						
<b>R</b>	Pre														
	Post			x	x	x		x		x	x	x	x	x	x

**Table I.17: Evaluation sheet - experimental group - pair 6**

		s1	s2	s3	s4	s5	s6	s7	s8	s9	s10	s11	s12	s13	s14
<b>P</b>	Pre														
	Post														
<b>U</b>	Pre					x							x		
	Post														
<b>M</b>	Pre	x	x	x	x		x	x	x	x	x	x		x	x
	Post												x		
<b>R</b>	Pre														
	Post	x	x	x	x	x	x	x	x	x	x	x		x	x

**Table I.18: Pair 6 matrix (Pre\_if\_OR\_2\_content and Post\_if\_OR\_content)**

<b>P</b>	<b>P→P</b> (Stagnant-no-learning)		<b>U→P</b> (Lower-level-deterioration)		<b>M→P</b> (Intermediate-deterioration+)		<b>R→P</b> (deteriorated++)	
	Con	Exp	Con	Exp	Con	Exp	Con	Exp
	0	0	0	0	0	0	0	0
<b>U</b>	<b>P→U</b> (Lower-level-improvement)		<b>U→U</b> (Stagnant-at-lower-level)		<b>M→U</b> (Intermediate-deterioration)		<b>R→U</b> (deteriorated+)	
	Con	Exp	Con	Exp	Con	Exp	Con	Exp
	0	0	0	0	0	0	0	0
<b>M</b>	<b>P→M</b> (Intermediate-improvement+)		<b>U→M</b> (Intermediate-improvement)		<b>M→M</b> (Stagnant-at-intermediate-level)		<b>R→M</b> (deteriorated)	
	Con	Exp	Con	Exp	Con	Exp	Con	Exp
	0	0	21.4%	7.1%	7.1%	0	0	0
<b>R</b>	<b>P→R</b> (Improved++)		<b>U→R</b> (Improved+)		<b>M→R</b> (Improved)		<b>R→R</b> (Already-improved)	
	Con	Exp	Con	Exp	Con	Exp	Con	Exp
	0	0	21.4%	7.1%	50%	85.7%	0	0
	<b>P</b>		<b>U</b>		<b>M</b>		<b>R</b>	



### Pair 7 (Pre\_else and Post\_else)

The evaluation outcome for the control group is in Table I.19 and for the experimental group in Table I.20. The content of Table I.21 is the evaluation matrix outcome of the control and the experimental group based on pair 7. Pair 7 means the correct content of the *else* part of the *if-statement*.

**Table I.19: Evaluation sheet - control group - pair 7**

		s1	s2	s3	s4	s5	s6	s7	s8	s9	s10	s11	s12	s13	s14
<b>P</b>	Pre														
	Post														
<b>U</b>	Pre				x			x		x	x	x		x	
	Post				x					x	x	x			
<b>M</b>	Pre	x		x		x			x				x		
	Post	x	x	x			x	x			x		x	x	
<b>R</b>	Pre		x				x								x
	Post					x			x						x

**Table I.20: Evaluation sheet - experimental group - pair 7**

		s1	s2	s3	s4	s5	s6	s7	s8	s9	s10	s11	s12	s13	s14
<b>P</b>	Pre														
	Post														
<b>U</b>	Pre			x		x	x		x			x			x
	Post														
<b>M</b>	Pre	x	x		x			x		x	x		x	x	
	Post			x			x		x		x				x
<b>R</b>	Pre														
	Post	x	x		x	x		x		x		x	x	x	

**Table I.21: Pair 7 matrix (Pre\_else and Post\_else)**

<b>P</b>	<b>P→P</b> (Stagnant-no-learning)		<b>U→P</b> (Lower-level-deterioration)		<b>M→P</b> (Intermediate-deterioration+)		<b>R→P</b> (deteriorated++)	
	Con	Exp	Con	Exp	Con	Exp	Con	Exp
	0	0	0	0	0	0	0	0
<b>U</b>	<b>P→U</b> (Lower-level-improvement)		<b>U→U</b> (Stagnant-at-lower-level)		<b>M→U</b> (Intermediate-deterioration)		<b>R→U</b> (deteriorated+)	
	Con	Exp	Con	Exp	Con	Exp	Con	Exp
	0	0	21.4%	0	0	0	0	0
<b>M</b>	<b>P→M</b> (Intermediate-improvement+)		<b>U→M</b> (Intermediate-improvement)		<b>M→M</b> (Stagnant-at-intermediate-level)		<b>R→M</b> (deteriorated)	
	Con	Exp	Con	Exp	Con	Exp	Con	Exp
	0	0	21.4%	28.6%	21.4%	7.1%	14.3%	0
<b>R</b>	<b>P→R</b> (Improved++)		<b>U→R</b> (Improved+)		<b>M→R</b> (Improved)		<b>R→R</b> (Already-improved)	
	Con	Exp	Con	Exp	Con	Exp	Con	Exp
	0	0	0	14.3%	14.3%	50%	7.1%	0
	<b>P</b>		<b>U</b>		<b>M</b>		<b>R</b>	

### Pair 8 (Pre\_display and Post\_display)

The evaluation outcome for the control group is in Table I.22 and for the experimental group in Table I.23. The content of Table I.24 is the evaluation matrix outcome of the control and the experimental group based on pair 8. Pair 8 means students demonstrate the ability to display the correct output at the right place in the code.

**Table I.22: Evaluation sheet - control group - pair 8**

		s1	s2	s3	s4	s5	s6	s7	s8	s9	s10	s11	s12	s13	s14
<b>P</b>	Pre														
	Post														
<b>U</b>	Pre		x			x						x			
	Post					x									
<b>M</b>	Pre			x			x	x	x	x	x		x		x
	Post		x					x		x		x	x		
<b>R</b>	Pre	x			x									x	
	Post	x		x	x		x		x		x			x	x

**Table I.23: Evaluation sheet - experimental group - pair 8**

		s1	s2	s3	s4	s5	s6	s7	s8	s9	s10	s11	s12	s13	s14
<b>P</b>	Pre														
	Post														
<b>U</b>	Pre			x					x				x		
	Post														
<b>M</b>	Pre	x	x		x		x	x		x	x	x		x	
	Post								x				x		
<b>R</b>	Pre					x									x
	Post	x	x	x	x	x	x	x		x	x	x		x	x

**Table I.24: Pair 8 matrix (Pre\_display and Post\_display)**

<b>P</b>	<b>P→P</b> (Stagnant-no-learning)		<b>U→P</b> (Lower-level-deterioration)		<b>M→P</b> (Intermediate-deterioration+)		<b>R→P</b> (deteriorated++)		
	Con	Exp	Con	Exp	Con	Exp	Con	Exp	
	0	0	0	0	0	0	0	0	
<b>U</b>	<b>P→U</b> (Lower-level-improvement)		<b>U→U</b> (Stagnant-at-lower-level)		<b>M→U</b> (Intermediate-deterioration)		<b>R→U</b> (deteriorated+)		
	Con	Exp	Con	Exp	Con	Exp	Con	Exp	
	0	0	7.1%	0	0	0	0	0	0
<b>M</b>	<b>P→M</b> (Intermediate-improvement+)		<b>U→M</b> (Intermediate-improvement)		<b>M→M</b> (Stagnant-at-intermediate-level)		<b>R→M</b> (deteriorated)		
	Con	Exp	Con	Exp	Con	Exp	Con	Exp	
	0	0	14.3%	14.3%	21.4%	0	0	0	
<b>R</b>	<b>P→R</b> (Improved++)		<b>U→R</b> (Improved+)		<b>M→R</b> (Improved)		<b>R→R</b> (Already-improved)		
	Con	Exp	Con	Exp	Con	Exp	Con	Exp	
	0	0	0	7.1%	35.7%	62.3%	21.4%	14.3%	
		<b>P</b>		<b>U</b>		<b>M</b>		<b>R</b>	

## Appendix J: Evaluation sheets for parallel arrays – Cycle 4

### Pair 1 (Pre\_array\_initialised and Post\_array\_initialised)

The evaluation outcome for the control group is in Table J.1 and for the experimental group in Table J.2. The content of Table J.3 is the evaluation matrix outcome of the control and the experimental group based on pair 1. Pair 1 refers to the capability to initialise parallel arrays.

**Table J.1: Evaluation sheet - control group - pair 1**

		s1	s2	s3	s4	s5	s6	s7	s8	s9	s10	s11
<b>P</b>	Pre											
	Post											
<b>U</b>	Pre								x			
	Post											
<b>M</b>	Pre	x				x		x				
	Post	x				x		x	x			
<b>R</b>	Pre		x	x	x		x			x	x	x
	Post		x	x	x		x			x	x	x

**Table J.2: Evaluation sheet - experimental group - pair 1**

		s1	s2	s3	s4	s5	s6	s7	s8	s9	s10	s11	s12
<b>P</b>	Pre												
	Post												
<b>U</b>	Pre					x		x					
	Post												
<b>M</b>	Pre			x									x
	Post												
<b>R</b>	Pre	x	x		x		x		x	x	x	x	
	Post	x	x	x	x	x	x	x	x	x	x	x	x

**Table J.3: Pair 1 matrix (Pre\_array\_initialised and Post\_array\_initialised)**

<b>P</b>	<b>P→P</b> (Stagnant-no-learning)		<b>U→P</b> (Lower-level-deterioration)		<b>M→P</b> (Intermediate-deterioration+)		<b>R→P</b> (deteriorated++)	
	<b>Con</b>	<b>Exp</b>	<b>Con</b>	<b>Exp</b>	<b>Con</b>	<b>Exp</b>	<b>Con</b>	<b>Exp</b>
	0	0	0	0	0	0	0	0
<b>U</b>	<b>P→U</b> (Lower-level-improvement)		<b>U→U</b> (Stagnant-at-lower-level)		<b>M→U</b> (Intermediate-deterioration)		<b>R→U</b> (deteriorated+)	
	<b>Con</b>	<b>Exp</b>	<b>Con</b>	<b>Exp</b>	<b>Con</b>	<b>Exp</b>	<b>Con</b>	<b>Exp</b>
	0	0	0	0	0	0	0	0
<b>M</b>	<b>P→M</b> (Intermediate-improvement+)		<b>U→M</b> (Intermediate-improvement)		<b>M→M</b> (Stagnant-at-intermediate-level)		<b>R→M</b> (deteriorated)	
	<b>Con</b>	<b>Exp</b>	<b>Con</b>	<b>Exp</b>	<b>Con</b>	<b>Exp</b>	<b>Con</b>	<b>Exp</b>
	0	0	9.1%	0	27.3%	0	0	0
<b>R</b>	<b>P→R</b> (Improved++)		<b>U→R</b> (Improved+)		<b>M→R</b> (Improved)		<b>R→R</b> (Already-improved)	
	<b>Con</b>	<b>Exp</b>	<b>Con</b>	<b>Exp</b>	<b>Con</b>	<b>Exp</b>	<b>Con</b>	<b>Exp</b>
	0	0	0	16.7%	0	16.7%	63.6%	66.7%
	<b>P</b>		<b>U</b>		<b>M</b>		<b>R</b>	

## Pair 2 (Pre\_loop\_structure and Post\_loop\_structure)

The evaluation outcome for the control group is in Table J.4 and for the experimental group in Table J.5. The content of Table J.6 is the evaluation matrix outcome of the control and the experimental group based on pair 2. Pair 2 refers to the structure of the loop which is controlled by the length of the array as a delimiter.

**Table J.4: Evaluation sheet - control group - pair 2**

		s1	s2	s3	s4	s5	s6	s7	s8	s9	s10	s11
<b>P</b>	Pre											
	Post											
<b>U</b>	Pre											
	Post											
<b>M</b>	Pre	x		x	x				x		x	
	Post	x			x				x			
<b>R</b>	Pre		x			x	x	x		x		x
	Post		x	x		x	x	x		x	x	x

**Table J.5: Evaluation sheet - experimental group - pair 2**

		s1	s2	s3	s4	s5	s6	s7	s8	s9	s10	s11	s12
<b>P</b>	Pre												
	Post												
<b>U</b>	Pre						x			x			
	Post												
<b>M</b>	Pre	x		x	x						x	x	x
	Post						x			x			
<b>R</b>	Pre		x			x		x	x				
	Post	x	x	x	x	x		x	x		x	x	x

**Table J.6: Pair 2 matrix (Pre\_loop\_structure and Post\_loop\_structure)**

<b>P</b>	<b>P→P</b> (Stagnant-no-learning)		<b>U→P</b> (Lower-level-deterioration)		<b>M→P</b> (Intermediate-deterioration+)		<b>R→P</b> (deteriorated++)	
	Con	Exp	Con	Exp	Con	Exp	Con	Exp
	0	0	0	0	0	0	0	0
<b>U</b>	<b>P→U</b> (Lower-level-improvement)		<b>U→U</b> (Stagnant-at-lower-level)		<b>M→U</b> (Intermediate-deterioration)		<b>R→U</b> (deteriorated+)	
	Con	Exp	Con	Exp	Con	Exp	Con	Exp
	0	0	0	0	0	0	0	0
<b>M</b>	<b>P→M</b> (Intermediate-improvement+)		<b>U→M</b> (Intermediate-improvement)		<b>M→M</b> (Stagnant-at-intermediate-level)		<b>R→M</b> (deteriorated)	
	Con	Exp	Con	Exp	Con	Exp	Con	Exp
	0	0	0	16.7%	27.3%	0	0	0
<b>R</b>	<b>P→R</b> (Improved++)		<b>U→R</b> (Improved+)		<b>M→R</b> (Improved)		<b>R→R</b> (Already-improved)	
	Con	Exp	Con	Exp	Con	Exp	Con	Exp
	0	0	0	0	18.2%	50%	63.6%	33.3%
	<b>P</b>		<b>U</b>		<b>M</b>		<b>R</b>	

### Pair 3 matrix (Pre\_if\_condition and Post\_if\_condition)

The evaluation outcome for the control group is in Table J.7 and for the experimental group in Table J.8. The content of Table J.9 is the evaluation matrix outcome of the control and the experimental group based on pair 3. Pair 3 refers to the structure of condition in the *if-statement* which must make use of the parallel arrays.

**Table J.7: Evaluation sheet - control group - pair 3**

		s1	s2	s3	s4	s5	s6	s7	s8	s9	s10	s11
<b>P</b>	Pre											
	Post											
<b>U</b>	Pre		x			x						
	Post											
<b>M</b>	Pre	x		x				x	x	x	x	
	Post	x	x	x		x		x	x	x	x	
<b>R</b>	Pre				x		x					x
	Post				x		x					x

**Table J.8: Evaluation sheet - experimental group - pair 3**

		s1	s2	s3	s4	s5	s6	s7	s8	s9	s10	s11	s12
<b>P</b>	Pre												
	Post												
<b>U</b>	Pre						x						
	Post												
<b>M</b>	Pre	x			x			x				x	
	Post						x						
<b>R</b>	Pre		x	x		x			x	x	x		x
	Post	x	x	x	x	x		x	x	x	x	x	x

**Table J.9: Pair 3 matrix (Pre\_if\_condition and Post\_if\_condition)**

<b>P</b>	<b>P→P</b> (Stagnant-no-learning)		<b>U→P</b> (Lower-level-deterioration)		<b>M→P</b> (Intermediate-deterioration+)		<b>R→P</b> (deteriorated++)	
	<b>Con</b>	<b>Exp</b>	<b>Con</b>	<b>Exp</b>	<b>Con</b>	<b>Exp</b>	<b>Con</b>	<b>Exp</b>
	0	0	0	0	0	0	0	0
<b>U</b>	<b>P→U</b> (Lower-level-improvement)		<b>U→U</b> (Stagnant-at-lower-level)		<b>M→U</b> (Intermediate-deterioration)		<b>R→U</b> (deteriorated+)	
	<b>Con</b>	<b>Exp</b>	<b>Con</b>	<b>Exp</b>	<b>Con</b>	<b>Exp</b>	<b>Con</b>	<b>Exp</b>
	0	0	0	0	0	0	0	0
<b>M</b>	<b>P→M</b> (Intermediate-improvement+)		<b>U→M</b> (Intermediate-improvement)		<b>M→M</b> (Stagnant-at-intermediate-level)		<b>R→M</b> (deteriorated)	
	<b>Con</b>	<b>Exp</b>	<b>Con</b>	<b>Exp</b>	<b>Con</b>	<b>Exp</b>	<b>Con</b>	<b>Exp</b>
	0	0	0	16.7%	54.5%	0	0	0
<b>R</b>	<b>P→R</b> (Improved++)		<b>U→R</b> (Improved+)		<b>M→R</b> (Improved)		<b>R→R</b> (Already-improved)	
	<b>Con</b>	<b>Exp</b>	<b>Con</b>	<b>Exp</b>	<b>Con</b>	<b>Exp</b>	<b>Con</b>	<b>Exp</b>
	0	0	0	0	18.2%	33.3%	27.3%	58.3%
	<b>P</b>		<b>U</b>		<b>M</b>		<b>R</b>	

### Pair 4 (Pre\_if\_content and Post\_if\_content)

The evaluation outcome for the control group is in Table J.10 and for the experimental group in Table J.11. The content of Table J.12 is the evaluation matrix outcome of the control and the experimental group based on pair 4. Pair 4 refers to the content of the *if-statement*, i.e., the block of code to be executed should the condition in the *if-statement* evaluate to true.

**Table J.10: Evaluation sheet - control group - pair 4**

		s1	s2	s3	s4	s5	s6	s7	s8	s9	s10	s11
<b>P</b>	Pre											
	Post											
<b>U</b>	Pre	x										
	Post											
<b>M</b>	Pre		x		x	x		x	x		x	x
	Post		x		x			x				x
<b>R</b>	Pre			x						x		
	Post	x		x		x			x	x	x	

**Table J.11: Evaluation sheet - experimental group - pair 4**

		s1	s2	s3	s4	s5	s6	s7	s8	s9	s10	s11	s12
<b>P</b>	Pre												
	Post												
<b>U</b>	Pre								x				x
	Post												
<b>M</b>	Pre	x	x	x	x		x			x	x	x	
	Post			x					x				x
<b>R</b>	Pre					x		x					
	Post	x	x		x	x	x	x		x	x	x	

**Table J.12: Pair 4 matrix (Pre\_if\_content and Post\_if\_content)**

<b>P</b>	<b>P→P</b> (Stagnant-no-learning)		<b>U→P</b> (Lower-level-deterioration)		<b>M→P</b> (Intermediate-deterioration+)		<b>R→P</b> (deteriorated++)	
	Con	Exp	Con	Exp	Con	Exp	Con	Exp
	0	0	0	0	0	0	0	0
<b>U</b>	<b>P→U</b> (Lower-level-improvement)		<b>U→U</b> (Stagnant-at-lower-level)		<b>M→U</b> (Intermediate-deterioration)		<b>R→U</b> (deteriorated+)	
	Con	Exp	Con	Exp	Con	Exp	Con	Exp
	0	0	9.1%	0	0	0	0	0
<b>M</b>	<b>P→M</b> (Intermediate-improvement+)		<b>U→M</b> (Intermediate-improvement)		<b>M→M</b> (Stagnant-at-intermediate-level)		<b>R→M</b> (deteriorated)	
	Con	Exp	Con	Exp	Con	Exp	Con	Exp
	0	0	9.1%	16.7%	36.4%	8.3%	0	0
<b>R</b>	<b>P→R</b> (Improved++)		<b>U→R</b> (Improved+)		<b>M→R</b> (Improved)		<b>R→R</b> (Already-improved)	
	Con	Exp	Con	Exp	Con	Exp	Con	Exp
	0	0	0	0	27.3%	58.3%	18.2%	16.7%
	<b>P</b>		<b>U</b>		<b>M</b>		<b>R</b>	

### Pair 5 (Pre\_display and Post\_display)

The evaluation outcome for the control group is in Table J.13 and for the experimental group in Table J.14. The content of Table J.15 is the evaluation matrix outcome of the control and the experimental group based on pair 5. Pair 5 refers to the capability to display output at the right position in the code.

**Table J.13: Evaluation sheet - control group - pair 5**

		s1	s2	s3	s4	s5	s6	s7	s8	s9	s10	s11
<b>P</b>	Pre											
	Post											
<b>U</b>	Pre	x		x					x			x
	Post			x			x		x		x	x
<b>M</b>	Pre				x	x	x			x	x	
	Post	x			x							
<b>R</b>	Pre		x					x				
	Post		x			x		x		x		

**Table J.14: Evaluation sheet - experimental group - pair 5**

		s1	s2	s3	s4	s5	s6	s7	s8	s9	s10	s11	s12
<b>P</b>	Pre												
	Post												
<b>U</b>	Pre		x		x		x		x				
	Post												
<b>M</b>	Pre	x		x		x				x	x	x	x
	Post				x								
<b>R</b>	Pre							x					
	Post	x	x	x		x	x	x	x	x	x	x	x

**Table J.15: Pair 5 matrix (Pre\_display and Post\_display)**

<b>P</b>	<b>P→P</b> (Stagnant-no-learning)		<b>U→P</b> (Lower-level-deterioration)		<b>M→P</b> (Intermediate-deterioration+)		<b>R→P</b> (deteriorated++)	
	<b>Con</b>	<b>Exp</b>	<b>Con</b>	<b>Exp</b>	<b>Con</b>	<b>Exp</b>	<b>Con</b>	<b>Exp</b>
	0	0	0	0	0	0	0	0
<b>U</b>	<b>P→U</b> (Lower-level-improvement)		<b>U→U</b> (Stagnant-at-lower-level)		<b>M→U</b> (Intermediate-deterioration)		<b>R→U</b> (deteriorated+)	
	<b>Con</b>	<b>Exp</b>	<b>Con</b>	<b>Exp</b>	<b>Con</b>	<b>Exp</b>	<b>Con</b>	<b>Exp</b>
	0	0	27.3%	0	0	0	0	0
<b>M</b>	<b>P→M</b> (Intermediate-improvement+)		<b>U→M</b> (Intermediate-improvement)		<b>M→M</b> (Stagnant-at-intermediate-level)		<b>R→M</b> (deteriorated)	
	<b>Con</b>	<b>Exp</b>	<b>Con</b>	<b>Exp</b>	<b>Con</b>	<b>Exp</b>	<b>Con</b>	<b>Exp</b>
	0	0	9.1%	8.3%	27.3%	0	0	0
<b>R</b>	<b>P→R</b> (Improved++)		<b>U→R</b> (Improved+)		<b>M→R</b> (Improved)		<b>R→R</b> (Already-improved)	
	<b>Con</b>	<b>Exp</b>	<b>Con</b>	<b>Exp</b>	<b>Con</b>	<b>Exp</b>	<b>Con</b>	<b>Exp</b>
	0	0	0	25%	18.2%	58.3%	18.2%	8.3%
	<b>P</b>		<b>U</b>		<b>M</b>		<b>R</b>	

## Appendix K: Evaluation sheets for functions – Cycle 4

### Pair 1 (Pre\_func\_call\_return and Post\_func\_call\_return)

The evaluation outcome for the control group is in Table K.1 and for the experimental group in Table K.2. The content of Table K.3 is the evaluation matrix outcome of the control and the experimental group based on pair 1. Pair 1 refers to the capability to call a function that returns a value.

**Table K.1: Evaluation sheet - control group - pair 1**

		s1	s2	s3	s4	s5	s6	s7	s8	s9	s10
<b>P</b>	Pre										
	Post										
<b>U</b>	Pre			x		x		x			x
	Post										
<b>M</b>	Pre	x	x				x			x	
	Post		x	x		x		x	x	x	x
<b>R</b>	Pre				x				x		
	Post	x			x		x				

**Table K.2: Evaluation sheet - experimental group - pair 1**

		s1	s2	s3	s4	s5	s6	s7	s8	s9	s10	s11	s12
<b>P</b>	Pre												
	Post												
<b>U</b>	Pre	x											
	Post												
<b>M</b>	Pre		x	x	x	x	x		x	x		x	x
	Post	x											
<b>R</b>	Pre							x			x		
	Post		x	x	x	x	x	x	x	x	x	x	x

**Table K.3: Pair 1 matrix (Pre\_func\_call\_return and Post\_func\_call\_return)**

<b>P</b>	<b>P→P</b> (Stagnant-no-learning)		<b>U→P</b> (Lower-level-deterioration)		<b>M→P</b> (Intermediate-deterioration+)		<b>R→P</b> (deteriorated++)	
	<b>Con</b>	<b>Exp</b>	<b>Con</b>	<b>Exp</b>	<b>Con</b>	<b>Exp</b>	<b>Con</b>	<b>Exp</b>
	0	0	0	0	0	0	0	0
<b>U</b>	<b>P→U</b> (Lower-level-improvement)		<b>U→U</b> (Stagnant-at-lower-level)		<b>M→U</b> (Intermediate-deterioration)		<b>R→U</b> (deteriorated+)	
	<b>Con</b>	<b>Exp</b>	<b>Con</b>	<b>Exp</b>	<b>Con</b>	<b>Exp</b>	<b>Con</b>	<b>Exp</b>
	0	0	0	0	0	0	0	0
<b>M</b>	<b>P→M</b> (Intermediate-improvement+)		<b>U→M</b> (Intermediate-improvement)		<b>M→M</b> (Stagnant-at-intermediate-level)		<b>R→M</b> (deteriorated)	
	<b>Con</b>	<b>Exp</b>	<b>Con</b>	<b>Exp</b>	<b>Con</b>	<b>Exp</b>	<b>Con</b>	<b>Exp</b>
	0	0	40%	8.3%	20%	0	10%	0
<b>R</b>	<b>P→R</b> (Improved++)		<b>U→R</b> (Improved+)		<b>M→R</b> (Improved)		<b>R→R</b> (Already-improved)	
	<b>Con</b>	<b>Exp</b>	<b>Con</b>	<b>Exp</b>	<b>Con</b>	<b>Exp</b>	<b>Con</b>	<b>Exp</b>
	0	0	0	0	20%	75%	10%	16.7%
	<b>P</b>		<b>U</b>		<b>M</b>		<b>R</b>	



### Pair 2 (Pre\_func\_call\_void and Post\_func\_call\_void)

The evaluation outcome for the control group is in Table K.4 and for the experimental group in Table K.5. The content of Table K.6 is the evaluation matrix outcome of the control and the experimental group based on pair 2. Pair 2 refers to the capability to call a function that does not return a value.

**Table K.4: Evaluation sheet - control group - pair 2**

		s1	s2	s3	s4	s5	s6	s7	s8	s9	s10
<b>P</b>	Pre										
	Post										
<b>U</b>	Pre			x							
	Post										
<b>M</b>	Pre	x	x			x	x	x	x	x	x
	Post	x	x	x	x		x		x		x
<b>R</b>	Pre				x						
	Post					x		x		x	

**Table K.5: Evaluation sheet - experimental group - pair 2**

		s1	s2	s3	s4	s5	s6	s7	s8	s9	s10	s11	s12
<b>P</b>	Pre												
	Post												
<b>U</b>	Pre												
	Post												
<b>M</b>	Pre		x	x	x	x	x	x	x	x	x	x	x
	Post					x							
<b>R</b>	Pre	x											
	Post	x	x	x	x		x	x	x	x	x	x	x

**Table K.6: Pair 2 matrix (Pre\_func\_call\_void and Post\_func\_call\_void)**

<b>P</b>	<b>P→P</b> (Stagnant-no-learning)		<b>U→P</b> (Lower-level-deterioration)		<b>M→P</b> (Intermediate-deterioration+)		<b>R→P</b> (deteriorated++)	
	<b>Con</b>	<b>Exp</b>	<b>Con</b>	<b>Exp</b>	<b>Con</b>	<b>Exp</b>	<b>Con</b>	<b>Exp</b>
	0	0	0	0	0	0	0	0
<b>U</b>	<b>P→U</b> (Lower-level-improvement)		<b>U→U</b> (Stagnant-at-lower-level)		<b>M→U</b> (Intermediate-deterioration)		<b>R→U</b> (deteriorated+)	
	<b>Con</b>	<b>Exp</b>	<b>Con</b>	<b>Exp</b>	<b>Con</b>	<b>Exp</b>	<b>Con</b>	<b>Exp</b>
	0	0	0	0	0	0	0	0
<b>M</b>	<b>P→M</b> (Intermediate-improvement+)		<b>U→M</b> (Intermediate-improvement)		<b>M→M</b> (Stagnant-at-intermediate-level)		<b>R→M</b> (deteriorated)	
	<b>Con</b>	<b>Exp</b>	<b>Con</b>	<b>Exp</b>	<b>Con</b>	<b>Exp</b>	<b>Con</b>	<b>Exp</b>
	0	0	10%	0	50%	8.3%	10%	0
<b>R</b>	<b>P→R</b> (Improved++)		<b>U→R</b> (Improved+)		<b>M→R</b> (Improved)		<b>R→R</b> (Already-improved)	
	<b>Con</b>	<b>Exp</b>	<b>Con</b>	<b>Exp</b>	<b>Con</b>	<b>Exp</b>	<b>Con</b>	<b>Exp</b>
	0	0	0	0	30%	83.3%	0	8.3%
	<b>P</b>		<b>U</b>		<b>M</b>		<b>R</b>	

**Pair 3 (Pre\_func\_header\_return and Post\_func\_header\_return)**

The evaluation outcome for the control group is in Table K.7 and for the experimental group in Table K.8. The content of Table K.9 is the evaluation matrix outcome of the control and the experimental group based on pair 3. Pair 3 refers to the capability to construct a function header for a function that returns a value with a relevant return type.

**Table K.7: Evaluation sheet - control group - pair 3**

		s1	s2	s3	s4	s5	s6	s7	s8	s9	s10
<b>P</b>	Pre										
	Post										
<b>U</b>	Pre		x			x				x	
	Post										
<b>M</b>	Pre	x		x	x		x	x	x		
	Post		x	x		x	x			x	
<b>R</b>	Pre										x
	Post	x			x			x	x		x

**Table K.8: Evaluation sheet - experimental group - pair 3**

		s1	s2	s3	s4	s5	s6	s7	s8	s9	s10	s11	s12
<b>P</b>	Pre												
	Post												
<b>U</b>	Pre								x		x		
	Post												
<b>M</b>	Pre	x	x	x	x		x	x		x		x	x
	Post										x		
<b>R</b>	Pre					x							
	Post	x	x	x	x	x	x	x	x	x		x	x

**Table K.9: Pair 3 matrix (Pre\_func\_header\_return and Post\_func\_header\_return)**

<b>P</b>	<b>P→P</b> (Stagnant-no-learning)		<b>U→P</b> (Lower-level-deterioration)		<b>M→P</b> (Intermediate-deterioration+)		<b>R→P</b> (deteriorated++)	
	<b>Con</b>	<b>Exp</b>	<b>Con</b>	<b>Exp</b>	<b>Con</b>	<b>Exp</b>	<b>Con</b>	<b>Exp</b>
	0	0	0	0	0	0	0	0
<b>U</b>	<b>P→U</b> (Lower-level-improvement)		<b>U→U</b> (Stagnant-at-lower-level)		<b>M→U</b> (Intermediate-deterioration)		<b>R→U</b> (deteriorated+)	
	<b>Con</b>	<b>Exp</b>	<b>Con</b>	<b>Exp</b>	<b>Con</b>	<b>Exp</b>	<b>Con</b>	<b>Exp</b>
	0	0	0	0	0	0	0	0
<b>M</b>	<b>P→M</b> (Intermediate-improvement+)		<b>U→M</b> (Intermediate-improvement)		<b>M→M</b> (Stagnant-at-intermediate-level)		<b>R→M</b> (deteriorated)	
	<b>Con</b>	<b>Exp</b>	<b>Con</b>	<b>Exp</b>	<b>Con</b>	<b>Exp</b>	<b>Con</b>	<b>Exp</b>
	0	0	30%	8.3%	20%	0	0	0
<b>R</b>	<b>P→R</b> (Improved++)		<b>U→R</b> (Improved+)		<b>M→R</b> (Improved)		<b>R→R</b> (Already-improved)	
	<b>Con</b>	<b>Exp</b>	<b>Con</b>	<b>Exp</b>	<b>Con</b>	<b>Exp</b>	<b>Con</b>	<b>Exp</b>
	0	0	0	8.3%	40%	75%	10%	8.3%
	<b>P</b>		<b>U</b>		<b>M</b>		<b>R</b>	

### Pair 4 (Pre\_func\_return\_content and Post\_func\_return\_content)

The evaluation outcome for the control group is in Table K.10 and for the experimental group in Table K.11. The content of Table K.12 is the evaluation matrix outcome of the control and the experimental group based on pair 4. Pair 4 refers to the capability to code the correct content for the body of a function that returns a value.

**Table K.10: Evaluation sheet - control group - pair 4**

		s1	s2	s3	s4	s5	s6	s7	s8	s9	s10
<b>P</b>	Pre										
	Post										
<b>U</b>	Pre	x							x		
	Post										
<b>M</b>	Pre			x	x	x		x		x	
	Post	x	x	x		x			x		x
<b>R</b>	Pre		x				x				x
	Post				x		x	x		x	

**Table K.11: Evaluation sheet - experimental group - pair 4**

		s1	s2	s3	s4	s5	s6	s7	s8	s9	s10	s11	s12
<b>P</b>	Pre												
	Post												
<b>U</b>	Pre			x		x					x		
	Post												
<b>M</b>	Pre	x	x		x			x	x	x		x	x
	Post			x									
<b>R</b>	Pre						x						
	Post	x	x		x	x	x	x	x	x	x	x	x

**Table K.12: Pair 4 matrix (Pre\_func\_return\_content and Post\_func\_return\_content)**

<b>P</b>	<b>P→P</b> (Stagnant-no-learning)		<b>U→P</b> (Lower-level-deterioration)		<b>M→P</b> (Intermediate-deterioration+)		<b>R→P</b> (deteriorated++)	
	<b>Con</b>	<b>Exp</b>	<b>Con</b>	<b>Exp</b>	<b>Con</b>	<b>Exp</b>	<b>Con</b>	<b>Exp</b>
	0	0	0	0	0	0	0	0
<b>U</b>	<b>P→U</b> (Lower-level-improvement)		<b>U→U</b> (Stagnant-at-lower-level)		<b>M→U</b> (Intermediate-deterioration)		<b>R→U</b> (deteriorated+)	
	<b>Con</b>	<b>Exp</b>	<b>Con</b>	<b>Exp</b>	<b>Con</b>	<b>Exp</b>	<b>Con</b>	<b>Exp</b>
	0	0	0	0	0	0	0	0
<b>M</b>	<b>P→M</b> (Intermediate-improvement+)		<b>U→M</b> (Intermediate-improvement)		<b>M→M</b> (Stagnant-at-intermediate-level)		<b>R→M</b> (deteriorated)	
	<b>Con</b>	<b>Exp</b>	<b>Con</b>	<b>Exp</b>	<b>Con</b>	<b>Exp</b>	<b>Con</b>	<b>Exp</b>
	0	0	20%	8.3%	20%	0	20%	0
<b>R</b>	<b>P→R</b> (Improved++)		<b>U→R</b> (Improved+)		<b>M→R</b> (Improved)		<b>R→R</b> (Already-improved)	
	<b>Con</b>	<b>Exp</b>	<b>Con</b>	<b>Exp</b>	<b>Con</b>	<b>Exp</b>	<b>Con</b>	<b>Exp</b>
	0	0	0	16.7%	30%	66.7%	10%	8.3%
	<b>P</b>		<b>U</b>		<b>M</b>		<b>R</b>	

### Pair 5 (Pre\_func\_header\_void and Post\_func\_header\_void)

The evaluation outcome for the control group is in Table K.13 and for the experimental group in Table K.14. The content of Table K.15 is the evaluation matrix outcome of the control and the experimental group based on pair 5. Pair 5 refers to the capability to construct a function header for a function of a type *void* that does not return a value.

**Table K.13: Evaluation sheet - control group - pair 5**

		s1	s2	s3	s4	s5	s6	s7	s8	s9	s10
<b>P</b>	Pre										
	Post										
<b>U</b>	Pre				x						
	Post										
<b>M</b>	Pre	x	x	x			x		x		x
	Post	x		x	x	x	x			x	x
<b>R</b>	Pre					x		x		x	
	Post		x					x	x		

**Table K.14: Evaluation sheet - experimental group - pair 5**

		s1	s2	s3	s4	s5	s6	s7	s8	s9	s10	s11	s12
<b>P</b>	Pre												
	Post												
<b>U</b>	Pre	x							x		x		
	Post												
<b>M</b>	Pre		x		x	x	x	x		x			x
	Post								x				
<b>R</b>	Pre			x								x	
	Post	x	x	x	x	x	x	x		x	x	x	x

**Table K.15: Pair 5 matrix (Pre\_func\_header\_void and Post\_func\_header\_void)**

<b>P</b>	<b>P→P</b> (Stagnant-no-learning)		<b>U→P</b> (Lower-level-deterioration)		<b>M→P</b> (Intermediate-deterioration+)		<b>R→P</b> (deteriorated++)	
	Con	Exp	Con	Exp	Con	Exp	Con	Exp
	0	0	0	0	0	0	0	0
<b>U</b>	<b>P→U</b> (Lower-level-improvement)		<b>U→U</b> (Stagnant-at-lower-level)		<b>M→U</b> (Intermediate-deterioration)		<b>R→U</b> (deteriorated+)	
	Con	Exp	Con	Exp	Con	Exp	Con	Exp
	0	0	0	0	0	0	0	0
<b>M</b>	<b>P→M</b> (Intermediate-improvement+)		<b>U→M</b> (Intermediate-improvement)		<b>M→M</b> (Stagnant-at-intermediate-level)		<b>R→M</b> (deteriorated)	
	Con	Exp	Con	Exp	Con	Exp	Con	Exp
	0	0	10%	8.3%	40%	0	20%	0
<b>R</b>	<b>P→R</b> (Improved++)		<b>U→R</b> (Improved+)		<b>M→R</b> (Improved)		<b>R→R</b> (Already-improved)	
	Con	Exp	Con	Exp	Con	Exp	Con	Exp
	0	0	0	16.7%	20%	58.3%	10%	16.7%
	<b>P</b>		<b>U</b>		<b>M</b>		<b>R</b>	

### Pair 6 matrix (Pre\_func\_header\_void and Post\_func\_header\_void)

The evaluation outcome for the control group is in Table K.16 and for the experimental group in Table K.17. The content of Table K.18 is the evaluation matrix outcome of the control and the experimental group based on pair 6. Pair 6 refers to the capability to code the correct content for the body of a function that does not return a value.

**Table K.16: Evaluation sheet - control group - pair 6**

		s1	s2	s3	s4	s5	s6	s7	s8	s9	s10
<b>P</b>	Pre										
	Post										
<b>U</b>	Pre		x		x					x	
	Post										
<b>M</b>	Pre	x		x		x	x	x	x		
	Post	x	x		x	x	x		x	x	
<b>R</b>	Pre										x
	Post			x				x			x

**Table K.17: Evaluation sheet - experimental group - pair 6**

		s1	s2	s3	s4	s5	s6	s7	s8	s9	s10	s11	s12
<b>P</b>	Pre												
	Post												
<b>U</b>	Pre			x							x		
	Post												
<b>M</b>	Pre	x	x		x	x	x	x	x	x		x	x
	Post			x			x				x		
<b>R</b>	Pre												
	Post	x	x		x	x		x	x	x		x	x

**Table K.18: Pair 6 matrix (Pre\_func\_header\_void and Post\_func\_header\_void)**

<b>P</b>	<b>P→P</b> (Stagnant-no-learning)		<b>U→P</b> (Lower-level-deterioration)		<b>M→P</b> (Intermediate-deterioration+)		<b>R→P</b> (deteriorated++)	
	Con	Exp	Con	Exp	Con	Exp	Con	Exp
	0	0	0	0	0	0	0	0
<b>U</b>	<b>P→U</b> (Lower-level-improvement)		<b>U→U</b> (Stagnant-at-lower-level)		<b>M→U</b> (Intermediate-deterioration)		<b>R→U</b> (deteriorated+)	
	Con	Exp	Con	Exp	Con	Exp	Con	Exp
	0	0	0	0	0	0	0	0
<b>M</b>	<b>P→M</b> (Intermediate-improvement+)		<b>U→M</b> (Intermediate-improvement)		<b>M→M</b> (Stagnant-at-intermediate-level)		<b>R→M</b> (deteriorated)	
	Con	Exp	Con	Exp	Con	Exp	Con	Exp
	0	0	30%	16.7%	40%	8.3%	0	0
<b>R</b>	<b>P→R</b> (Improved++)		<b>U→R</b> (Improved+)		<b>M→R</b> (Improved)		<b>R→R</b> (Already-improved)	
	Con	Exp	Con	Exp	Con	Exp	Con	Exp
	0	0	0	0	20%	75%	10%	0
	<b>P</b>		<b>U</b>		<b>M</b>		<b>R</b>	

## Appendix L: A switch button to Java code – Cycle 5

In this cycle, the animation programs for uses of *assignment*, nested-decider and the animation program on the combination of *decisions* and *loops* have been incorporated with the Java code. These animation programs have the option to switch to a Java code.

The screenshot shows an animation interface with a green background. On the left, there is a diagram of a nested-decider. It consists of a horizontal bar with a yellow segment in the middle. Below this bar, there are two vertical bars. The left vertical bar is labeled 'True' and the right vertical bar is labeled 'False'. The horizontal bar is labeled 'True' on the left and 'False' on the right. A 'Start' button is located at the top right of the diagram area. On the right side of the interface, there is a control panel with a 'Start' button, a 'Reset' button, and an 'Animation speed' section with three radio buttons: 'Normal' (selected), 'Slow', and 'Very slow'. Below the control panel is a 'Java code' section with a text area containing the following code:

```

1. System.out.println ("Have gate card?");
2. gatecard = keyboard.next().charAt(0);
3. System.out.println ("Have ID card?");
4. identityC = keyboard.next().charAt(0);
5
6 if (gatecard == 'Y' || identityC == 'Y')
7 {
8. System.out.println ("Have office card?");
9. officecard = keyboard.next().charAt(0);
10
11. System.out.println ("Got office door pin code correct?");
12. officepin = keyboard.next().charAt(0);
13
14 if (officecard == 'Y' && officepin == 'Y')
15. System.out.println ("Access into the office granted");
16 else
17. System.out.println ("Access into the office not granted");
18 }
19 else
20. System.out.println ("Access into the company not granted");

```

Figure K.1: Switching nested-decider to a Java code

The screenshot shows an animation interface with a green background. On the left, there is a diagram of a loop. It consists of a horizontal bar with a yellow segment in the middle. Below this bar, there are two vertical bars. The left vertical bar is labeled 'True' and the right vertical bar is labeled 'False'. The horizontal bar is labeled 'True' on the left and 'False' on the right. A 'Start' button is located at the top right of the diagram area. On the right side of the interface, there is a control panel with a 'Start' button, a 'Reset' button, and an 'Animation speed' section with three radio buttons: 'Normal' (selected), 'Slow', and 'Very slow'. Below the control panel is a 'Java code' section with a text area containing the following code:

```

1 int malescount = 0;
2 int femalescount = 0;
3 char gender;
4. Scanner keyboard = new Scanner(System.in);
5 for (int x = 0; x < 6; x++)
6 {
7. System.out.println ("Enter gender?");
8. gender = keyboard.next().charAt(0);
9
10 if (gender == 'M' || gender == 'm')
11. malescount = malescount + 1;
12 else if (gender == 'F' || gender == 'f')
13. femalescount++;
14 else
15 {
16. System.out.println ("Invalid character, please re-enter gender?");
17. x--;
18. }
19 }
20
21. System.out.println ("Total males " + malescount);
22. System.out.println ("Total females " + femalescount);

```

Figure K.2: Switching *decisions/loops* to a Java code

## Appendix M: Transition results– Cycle 5

Table M.1 displays the average percentages for the improving transitions in cycle 5.

**Table M.1: Improving transition results**

<b>Group</b>	<b>U→M (Intermediate- improvement)</b>		<b>M→R (Improved)</b>		<b>U→R (Improved+)</b>	
	<b>Con</b>	<b>Exp</b>	<b>Con</b>	<b>Exp</b>	<b>Con</b>	<b>Exp</b>
<b>Decisions/ nested decisions</b>	21.7%	10.5%	46%	72.2%	0	5.5%
<b>Decisions/ loops</b>	16.4%	13.3%	42.2%	70.8%	0	6.7%
<b>Loops</b>	7.5%	2.3%	32.5%	50%	0	2.3%
<b>Nested loops</b>	15%	9.1%	32.5%	72.5%	0	2.3%
<b>One dimensional array</b>	18%	10.9%	18%	45.5%	2%	9.1%
<b>Functions</b>	14%	6.1%	32.9%	72.5%	0	6.7%

Table M.2 displays the average percentages for the non-improving transitions and already improved transitions in cycle 5.

**Table L.2: Non-improving and already-improved transitions results**

<b>Group</b>	<b>U→U (Stagnant- at-lower- level)</b>		<b>M→M (Stagnant-at- intermediate -level)</b>		<b>R→M (Deteriorated)</b>		<b>R→R (Already- improved)</b>	
	<b>Con</b>	<b>Exp</b>	<b>Con</b>	<b>Exp</b>	<b>Con</b>	<b>Exp</b>	<b>Con</b>	<b>Exp</b>
<b>Decisions/ nested decisions</b>	0	0	26.7%	4.8%	1.7%	0	14.2%	11.1%
<b>Decisions/ loops</b>	1.6%	0	24.2%	2.5%	3.9%	0	13.4%	6.7%
<b>Loops</b>	0	0	30%	4.6%	0	0	27.5%	41.1%
<b>Nested loops</b>	0	0	27.5%	0	2.5%	0	22.5%	15.9%
<b>One dimensional array</b>	0	0	32%	1.8%	0	0	22%	32.7%
<b>Functions</b>	3.9%	0	32.3%	1.4%	1.3%	0	15.8%	11.7%