

# Contents

|  |           |
|--|-----------|
| <b>Preface</b>   | <b>v</b>  |
| <b>I A++ - The Language</b>  | <b>1</b>  |
| <b>1 Educational Programming Languages</b>   | <b>3</b>  |
| 1.1 Introduction . . . . .   | 3         |
| 1.2 Pascal . . . . .   | 3         |
| 1.3 Scheme . . . . .   | 3         |
| 1.4 Logo . . . . .   | 4         |
| 1.5 A++ . . . . .  | 4         |
| <b>2 Introduction to A++</b>   | <b>5</b>  |
| 2.1 Purpose of A++ and Origin . . . . .  | 5         |
| 2.1.1 Purpose . . . . .  | 5         |
| 2.1.2 Origin . . . . .   | 5         |
| 2.1.3 ARS — <i>Generalization of the Lambda-Calculus</i> . . . . .   | 5         |
| 2.1.4 Name of the language . . . . .   | 6         |
| 2.2 Motivations for the development of A++ . . . . .   | 6         |
| 2.2.1 To support an alternate method of teaching the principles of programming                             | 6         |
| 2.2.2 To provide a learning tool for exploring the fundamentals of programming                             | 7         |
| 2.2.3 To support a method teaching powerful programming patterns applicable<br>to most languages . . . . . | 7         |
| 2.2.4 To open a new view of programming for many programmers: . . . . .                                    | 7         |
| 2.3 Features of A++ . . . . .  | 7         |
| 2.3.1 Programming paradigms supported: . . . . .   | 7         |
| 2.3.2 Constitutive Principles of A++ . . . . .   | 7         |
| 2.3.3 Closure . . . . .  | 8         |
| 2.3.4 Basic abstractions derived from ARS . . . . .  | 9         |
| 2.3.5 Development of Applications with A++ . . . . .   | 9         |
| 2.4 Internal Architecture of A++ . . . . .   | 10        |
| 2.4.1 Definition of A++ as Programming Language . . . . .  | 10        |
| 2.4.2 Examples of A++ - Syntax . . . . .   | 11        |
| <b>3 Linking Logic with the Physical World</b>   | <b>13</b> |
| 3.1 Syntax of A++ including pre-defined primitive abstractions . . . . .                                   | 14        |
| 3.2 Examples using pre-defined primitives of A++ . . . . .   | 14        |
| 3.2.1 Pre-defined primitive abstractions in A++ . . . . .  | 15        |
| 3.2.2 A++ including pre-defined primitives . . . . .   | 16        |
| 3.2.3 A++ Overview . . . . .   | 17        |
| <b>4 General Programming Patterns and A++</b>  | <b>19</b> |

|          |  |           |
|----------|--|-----------|
| 4.1      | Closure Pattern . . . . .                                      | 19        |
| 4.2      | CLAM Pattern . . . . .   | 20        |
| 4.3      | List Pattern . . . . .   | 20        |
| 4.4      | Dictionary Pattern . . . . .                                   | 21        |
| 4.5      | Set Pattern . . . . .  | 22        |
| 4.6      | Recursion Pattern . . . . .                                    | 22        |
| 4.7      | Higher Order Function Pattern . . . . .                        | 23        |
| 4.8      | Message Passing Pattern . . . . .                              | 23        |
| 4.8.1    | Classes of objects . . . . .                                   | 23        |
| 4.8.2    | Instance of a class . . . . .                                  | 24        |
| 4.8.3    | Constructors . . . . .   | 24        |
| 4.8.4    | Creating instances of a class . . . . .                        | 25        |
| 4.8.5    | Sending messages . . . . .                                     | 26        |
| 4.8.6    | Executing methods . . . . .                                    | 27        |
| 4.8.7    | Essential features of object oriented programming . . . . .    | 28        |
| 4.8.8    | Relation between classes . . . . .                             | 29        |
| 4.9      | Meta Object Protocol Pattern . . . . .                         | 29        |
| <b>5</b> | <b>Discovering the Power of A++</b> . . . . .                  | <b>31</b> |
| 5.1      | Basic Abstractions . . . . .                                   | 31        |
| 5.1.1    | The IF- Abstraction . . . . .                                  | 31        |
| 5.1.2    | Extended Logical Abstractions . . . . .                        | 32        |
| 5.1.3    | Application of extended logical abstractions . . . . .         | 33        |
| 5.2      | Numeric Abstractions . . . . .                                 | 33        |
| 5.2.1    | Natural Numbers . . . . .                                      | 33        |
| 5.2.2    | Arithmetic Operations . . . . .                                | 35        |
| 5.2.3    | Application of numeric abstractions . . . . .                  | 36        |
| 5.3      | Collections of Data . . . . .                                  | 36        |
| 5.3.1    | Basic abstractions for pairs . . . . .                         | 37        |
| 5.3.2    | Basic utility abstractions for lists . . . . .                 | 38        |
| 5.4      | Extended Numerical Abstractions . . . . .                      | 39        |
| 5.4.1    | Abstraction ‘zeropair’ . . . . .                               | 39        |
| 5.4.2    | Decrementing a number: ‘pred’ . . . . .                        | 39        |
| 5.4.3    | Subtracting a number: ‘sub’ . . . . .                          | 40        |
| 5.5      | Relational Abstractions . . . . .                              | 40        |
| 5.5.1    | Comparing two numbers: ‘equaln’ . . . . .                      | 40        |
| 5.5.2    | Comparing two numbers: ‘gtp’ . . . . .                         | 40        |
| 5.5.3    | Comparing two numbers: ‘ltp’ . . . . .                         | 40        |
| 5.5.4    | Comparing two numbers: ‘gep’ . . . . .                         | 41        |
| 5.6      | Examples for recursion . . . . .                               | 41        |
| 5.6.1    | Calculating the faculty of a number . . . . .                  | 41        |
| 5.6.2    | Calculating the sum of elements of a list . . . . .            | 41        |
| 5.6.3    | Insertion Sort . . . . .                                       | 41        |
| 5.7      | Higher Order Functions . . . . .                               | 42        |
| 5.7.1    | Creating a new function by composition: ‘compose’ . . . . .    | 42        |
| 5.7.2    | Changing the arity of a function: ‘curry’ . . . . .            | 42        |
| 5.7.3    | Converting all elements in a list: ‘map’ . . . . .             | 42        |
| 5.7.4    | Converting the ‘map’ function: ‘mapc’ . . . . .                | 43        |
| 5.7.5    | Selecting elements from a given list: ‘filter’ . . . . .       | 44        |
| 5.7.6    | Searching for an element in a given list: ‘locate’ . . . . .   | 44        |
| 5.7.7    | Iterating through all elements of a list: ‘for-each’ . . . . . | 44        |
| 5.8      | Set Operations . . . . .                                       | 45        |

|           |  |           |
|-----------|--|-----------|
| 5.8.1     | Checking for a member in a set: 'memberp'            | 45        |
| 5.8.2     | Adding an element to a set: 'addelt'                 | 45        |
| 5.8.3     | Combining two sets: 'union'                          | 46        |
| 5.9       | Associative Lists in A++                             | 46        |
| 5.9.1     | Abstractions for associative lists                   | 47        |
| 5.9.2     | Application of associative lists                     | 47        |
| 5.10      | Miscellaneous abstractions                           | 48        |
| 5.10.1    | Testing the basic abstractions                       | 49        |
| 5.11      | Object Oriented Programming in A++                   | 50        |
| 5.11.1    | First example of object oriented programming in A++  | 50        |
| 5.11.2    | Second example of object oriented programming in A++ | 51        |
| 5.11.3    | Third example of object oriented programming in A++  | 57        |
| 5.12      | Imperative Programming in A++                        | 65        |
| <b>6</b>  | <b>Computer Resources for A++</b>                    | <b>67</b> |
| 6.1       | Support Functions                                    | 67        |
| 6.1.1     | Abstraction for displaying a number                  | 67        |
| 6.1.2     | Abstraction for displaying a boolean value           | 67        |
| 6.1.3     | Abstraction for displaying lists                     | 67        |
| 6.2       | A++ Interpreters                                     | 67        |
| 6.2.1     | A++ Interpreter written in Perl                      | 68        |
| 6.2.2     | A++ Interpreter written in C                         | 72        |
| 6.3       | Initializing the A++ Interpreter                     | 74        |
| 6.3.1     | Initializing the A++ Interpreter part 1              | 74        |
| 6.3.2     | Initializing the A++ Interpreter part 2              | 74        |
| 6.3.3     | Initializing the A++ Interpreter part 3              | 74        |
| 6.3.4     | Initializing the A++ Interpreter part 4              | 75        |
| 6.3.5     | Initializing the A++ Interpreter part 5              | 75        |
| 6.3.6     | Initializing the A++ Interpreter part 6              | 76        |
| 6.3.7     | Initializing the A++ Interpreter part 7              | 76        |
| 6.4       | WWW Links  | 77        |
| <b>7</b>  | <b>Extending A++</b>                                 | <b>79</b> |
| 7.1       | ARS++  | 79        |
| 7.2       | ARSAPI   | 80        |
| <b>II</b> | <b>A++ - The Implementation</b>                      | <b>81</b> |
| <b>1</b>  | <b>General Introduction</b>                          | <b>83</b> |
| <b>2</b>  | <b>A++ Implementation in Perl</b>                    | <b>85</b> |
| 2.1       | ARS and Perl   | 85        |
| 2.1.1     | Syntax of Perl                                       | 85        |
| 2.1.2     | Syntax of ARS  | 86        |
| 2.1.3     | Basic Abstractions                                   | 86        |
| 2.1.4     | Numeric Abstractions                                 | 88        |
| 2.1.5     | Arithmetic Operations                                | 88        |
| 2.1.6     | Application of numeric abstractions                  | 88        |
| 2.1.7     | Collections of Data                                  | 88        |
| 2.1.8     | Basic utility abstractions for lists                 | 90        |
| 2.1.9     | Relational Abstractions                              | 92        |
| 2.1.10    | Examples for recursion                               | 92        |

|          |   |            |
|----------|---|------------|
| 2.1.11   | Higher Order Functions . . . . .                                    | 93         |
| 2.1.12   | Set Operations . . . . .  | 95         |
| 2.1.13   | Associative lists . . . . .   | 97         |
| 2.1.14   | Object Oriented Programming in Perl . . . . .                       | 101        |
| 2.1.15   | Regular expressions . . . . .                                       | 104        |
| 2.2      | A++ Interpreter . . . . .   | 105        |
| 2.2.1    | Main Program: Command Line Mode . . . . .                           | 105        |
| 2.2.2    | Main Program: WEB-Based Application . . . . .                       | 105        |
| 2.2.3    | Expression Definition Module (EXP) . . . . .                        | 107        |
| 2.2.4    | List Module (PAIR) . . . . .  | 108        |
| 2.2.5    | Value Module (VALUE) . . . . .                                      | 110        |
| 2.2.6    | Name Management Module (NAME) . . . . .                             | 111        |
| 2.2.7    | Environment Module (ENV) . . . . .                                  | 112        |
| 2.2.8    | Parser Module (ARSP) . . . . .                                      | 116        |
| 2.2.9    | Expression Evaluation Module (EVAL) . . . . .                       | 120        |
| <b>3</b> | <b>A++ Implementation in C</b> . . . . .                            | <b>129</b> |
| 3.1      | ARS and C . . . . .   | 129        |
| 3.1.1    | Mechanisms for Abstraction . . . . .                                | 129        |
| 3.1.2    | Operations within functions . . . . .                               | 130        |
| 3.1.3    | Simple and complex expressions . . . . .                            | 136        |
| 3.1.4    | Modularization . . . . .  | 136        |
| 3.1.5    | Variables and pointers . . . . .                                    | 137        |
| 3.1.6    | Structures in C . . . . .   | 139        |
| 3.1.7    | Special control operations: setjmp/longjmp . . . . .                | 140        |
| 3.2      | A++ Interpreter in C . . . . .                                      | 141        |
| 3.2.1    | Introduction . . . . .  | 141        |
| 3.2.2    | ADT's used by the the interpreter . . . . .                         | 141        |
| 3.2.3    | Functions used by the interpreter . . . . .                         | 146        |
| 3.2.4    | Source code of the A++ Interpreter . . . . .                        | 156        |
| 3.3      | ARS based programming in C . . . . .                                | 178        |
| 3.3.1    | The 'clam' as a key for ARS based programming in C . . . . .        | 178        |
| 3.3.2    | Datatypes . . . . .   | 179        |
| 3.3.3    | Mapping ARS to C . . . . .  | 185        |
| 3.3.4    | Simple example . . . . .  | 185        |
| 3.3.5    | Basic demo of ARSAPI for C . . . . .                                | 186        |
| 3.3.6    | Advanced demo of ARSAPI for C: menu system 'genfui' . . . . .       | 187        |
| 3.3.7    | Source code of 'ARSAPI for C' demos . . . . .                       | 187        |
| 3.4      | External tools . . . . .  | 207        |
|          | <b>Appendices</b> . . . . .   | <b>211</b> |
| <b>A</b> | <b>Detailed Discussion of Addition and Multiplication</b> . . . . . | <b>211</b> |
| A.1      | Addition of the numbers 'two' and 'three' . . . . .                 | 211        |
| A.2      | Multiplication of the numbers 'two' and 'three' . . . . .           | 213        |
| <b>B</b> | <b>The Lambda Calculus</b> . . . . .                                | <b>217</b> |
| B.1      | Introduction . . . . .  | 217        |
| B.1.1    | Origin . . . . .  | 217        |
| B.1.2    | Definition . . . . .  | 217        |
| B.1.3    | Literature . . . . .  | 217        |
| B.2      | Syntax of Lambda Expressions . . . . .                              | 217        |

|          |  |            |
|----------|--|------------|
| B.3      | Basic Rules for Lambda Conversions . . . . .           | 218        |
| B.3.1    | Notation used in Conversion Rules . . . . .            | 218        |
| B.3.2    | Alpha Conversion . . . . .                             | 218        |
| B.3.3    | Beta Conversion . . . . .                              | 219        |
| B.3.4    | Eta Conversion . . . . .                               | 220        |
| B.3.5    | Rules of Associativity . . . . .                       | 221        |
| B.3.6    | Y-Combinator . . . . .                                 | 222        |
| <b>C</b> | <b>Testing the Y-Combinator in A++</b>                 | <b>227</b> |
| C.1      | A++ Source code of Y-Combinator test program . . . . . | 227        |
| C.2      | Running the Y-Combinator test program . . . . .        | 227        |
| C.3      | Comments on the Y-Combinator test program . . . . .    | 228        |
| <b>D</b> | <b>Background of the Author</b>                        | <b>229</b> |
|          | <b>Index</b>   | <b>233</b> |