Contents

Foreword v

Preface vii

Part I Introduction 1

1 The concept of the C++ Standard Template Library 3

1.1 Genericity of components 4
1.2 Abstract and implicit data types 4
1.3 The fundamental concept 5
 1.3.1 Containers 5
 1.3.2 Iterators 5
 1.3.3 Algorithms 6
 1.3.4 Interplay 6
1.4 Internal functioning 9
1.5 Complexity 14
 1.5.1 O notation 15
 1.5.2 Ω notation 18
1.6 Auxiliary classes and functions 19
 1.6.1 Pairs 19
 1.6.2 Comparison operators 20
 1.6.3 Function objects 21
 1.6.4 Function adapters 24
1.7 Some conventions 27
 1.7.1 Namespaces 27
 1.7.2 Header files 28
 1.7.3 Allocators 28

2 Iterators 29

2.1 Iterator properties 30
 2.1.1 States 30
 2.1.2 Standard iterator and traits classes 30
 2.1.3 Distances 32
 2.1.4 Categories 33

2.1.5 Reverse iterators . . . 35
2.1.6 Const iterators . . . 36
2.1.7 Tag classes . . . 36
2.2 Stream iterators . . . 37
2.2.1 Istream iterator . . . 37
2.2.2 Ostream iterator . . . 40

3 Containers 45
3.1 Data type interface . . . 45
3.2 Container methods . . . 46
3.2.1 Reversible containers . . . 46
3.3 Sequences . . . 47
3.3.1 Vector . . . 49
3.3.2 List . . . 52
3.3.3 Deque . . . 56
3.3.4 showSequence . . . 56
3.4 Iterator categories and containers . . . 58
3.4.1 Derivation of value and distance types . . . 61
3.4.2 Inheriting iterator properties . . . 63
3.5 Iterators for insertion into containers . . . 63

4 Abstract data types 69
4.1 Stack . . . 69
4.2 Queue . . . 70
4.3 Priority queue . . . 72
4.4 Sorted associative containers . . . 73
4.4.1 Set . . . 74
4.4.2 Multiset . . . 78
4.4.3 Map . . . 78
4.4.4 Multimap . . . 81

Part II Algorithms 83

5 Standard algorithms 85
5.1 Copying algorithms . . . 85
5.2 Algorithms with predicates . . . 86
5.2.1 Algorithms with binary predicates . . . 87
5.3 Nonmutating sequence operations . . . 87
5.3.1 for_each . . . 87
5.3.2 find and find_if . . . 89
5.3.3 find_end . . . 90
5.3.4 find_first_of . . . 92
5.3.5 adjacent_find . . . 93
5.3.6 count and count_if . . . 94
5.3.7 mismatch . . . 95

5.3.8 equal 98
5.3.9 search 99
5.3.10 search_n 101
5.4 Mutating sequence operations 101
5.4.1 iota 101
5.4.2 copy and copy_backward 102
5.4.3 copy_if 104
5.4.4 swap, iter_swap, and swap_ranges 105
5.4.5 transform 107
5.4.6 replace and variants 109
5.4.7 fill and fill_n 111
5.4.8 generate and generate_n 112
5.4.9 remove and variants 113
5.4.10 unique 115
5.4.11 reverse 116
5.4.12 rotate 117
5.4.13 random_shuffle 119
5.4.14 partition 121
5.5 Sorting, merging, and related operations 122
5.5.1 sort 122
5.5.2 nth_element 126
5.5.3 Binary search 127
5.5.4 Merging 130
5.6 Set operations on sorted structures 134
5.6.1 includes 134
5.6.2 set_union 135
5.6.3 set_intersection 136
5.6.4 set_difference 137
5.6.5 set_symmetric_difference 138
5.6.6 Conditions and limitations 139
5.7 Heap algorithms 141
5.7.1 pop_heap 143
5.7.2 push_heap 145
5.7.3 make_heap 147
5.7.4 sort_heap 148
5.8 Minimum and maximum 150
5.9 Lexicographical comparison 151
5.10 Permutations 152
5.11 Numeric algorithms 153
5.11.1 accumulate 153
5.11.2 inner_product 154
5.11.3 partial_sum 156
5.11.4 adjacent_difference 157

Part III Beyond the STL: components and applications 159

6 Set operations on associative containers 161
6.1 Subset relation . . . 162
6.2 Union . . . 162
6.3 Intersection . . . 163
6.4 Difference . . . 163
6.5 Symmetric difference . . . 164
6.6 Example . . . 165

7 Fast associative containers 169
7.1 Fundamentals . . . 169
7.1.1 Collision handling . . . 170
7.2 Map . . . 171
7.2.1 Example . . . 180
7.3 Set . . . 181
7.4 Overloaded operators for sets . . . 182
7.4.1 Union . . . 182
7.4.2 Intersection . . . 183
7.4.3 Difference . . . 183
7.4.4 Symmetric difference . . . 183
7.4.5 Example . . . 184

8 Various applications 185
8.1 Cross-reference . . . 185
8.2 Permuted index . . . 187
8.3 Thesaurus . . . 190

9 Vectors and matrices 195
9.1 Checked vectors . . . 195
9.2 Matrices as nested containers . . . 197
9.2.1 Two-dimensional matrices . . . 198
9.2.2 Three-dimensional matrix . . . 201
9.2.3 Generalization . . . 204
9.3 Matrices for different memory models . . . 204
9.3.1 C memory layout . . . 207
9.3.2 FORTRAN memory layout . . . 208
9.3.3 Memory layout for symmetric matrices . . . 209
9.4 Sparse matrices . . . 210
9.4.1 Index operator and assignment . . . 213
9.4.2 Hash function for index pairs . . . 214
9.4.3 Class MatrixElement . . . 215
9.4.4 Class sparseMatrix . . . 217
9.4.5 Run time measurements . . . 220

10 External sorting **221**
10.1 External sorting by merging . . . 221
10.2 External sorting with accelerator . . . 228

11 Graphs **233**
11.1 Class Graph . . . 236
11.1.1 Insertion of vertices and edges . . . 238
11.1.2 Analysis of a graph . . . 239
11.1.3 Input and output tools . . . 243
11.2 Dynamic priority queue . . . 245
11.2.1 Data structure . . . 246
11.2.2 Class dynamic_priority_queue . . . 246
11.3 Graph algorithms . . . 252
11.3.1 Shortest paths . . . 254
11.3.2 Topological sorting of a graph . . . 258

A Appendix **265**
A.1 Auxiliary programs . . . 265
A.1.1 Reading the thesaurus file roget.dat . . . 265
A.1.2 Reading a graph file . . . 266
A.1.3 Creation of vertices with random coordinates . . . 267
A.1.4 Connecting neighboring vertices . . . 268
A.1.5 Creating a LaTeX file . . . 269
A.2 Sources and comments . . . 271
A.3 Solutions to selected exercises . . . 271
A.4 Overview of the sample files . . . 279
A.4.1 Files in the include directory . . . 279
A.4.2 Files for the introductory examples . . . 279
A.4.3 Files for the standard algorithms . . . 279
A.4.4 Files for applications and extensions . . . 280

References **283**

Index **285**