

TABLE OF CONTENTS

CHAPTER ONE	1
SOFTWARE COMPLEXITY AND OBJECT-ORIENTED	
.1
PROGRAMMING CONCEPTS.	1
LEARNING OBJECTIVES	2
PROGRAMMING PROBLEM SOLUTION METHODS .3	
 THE PROBLEM EVOLUTION3
 IN COMPARISON TO AN INTELLIGENT SYSTEM. . 3	
COMPLEXITY	6
LARGENESS	8
 INPUT STIMULI AND A SYSTEM'S RESPONSE . . . 8	
 MAINTAINABILITY	8
 GENERICITY	9
 ABSTRACTION	9
 ENCAPSULATION	9
OBJECT ORIENTED COMPUTING.	11
 INTRODUCTION	12
ENCAPSULATION, INHERITANCE, AND	
POLYMORPHISM	12
 PROGRAM STRUCTURE	13

OBJECT ORIENTED CONCEPTS	18
OBJECT INTERACTION	20
SELF REFERENCE	21
SUMMARY	21
CLASSES	22
CREATING INSTANCES OF AN OBJECT	22
SUMMARY	23
INHERITANCE24
POLYMORPHISM26
WHAT ARE OBJECT ORIENTED SYSTEMS?27
A GENERAL MODEL OF OBJECT-ORIENTED	COMPUTING . . .30
ENCAPSULATION	30
CLASSIFICATION31
a) Sets	32
b) Abstract data types	32
c) Classes (Defined data types)	32
d) Objects (Prototypes from class templates)	32
POLYMORPHISM.	33
a) Implicitly.	33
b) Explicitly.	34
(1) Overloading	34
(2) Parametric Polymorphism	34

INTERPRETATION	34
ACTUAL IMPLEMENTATION	36
CHAPTER TWO.	37
A COMPREHENSIVE REVIEW OF C	37
LEARNING OBJECTIVES	37
CONVENTIONS USED IN TRAINING MANUAL	38
INTRODUCTION	40
ELEMENTS OF A C PROGRAM	41
LANGUAGE TRANSLATORS	43
HOW DOES A COMPILER WORK?	44
STAGES DURING COMPILATION.	45
ANALYSIS OF THE SOURCE CODE	46
1. LEXICAL (or LINEAR) ANALYSIS	46
2. SYNTACTIC (or HIERARCHICAL) ANALYSIS	46
3. SEMANTICS ANALYSIS.	46,48
THE SYMBOL TABLE MANAGER	48
PRE-PROCESSORS	48
LOADER AND LINK EDITOR.	49
FUNCTIONS	50
OVERVIEW OF FUNCTIONS	50
THE FUNCTION PROTOTYPE	50

PARADIGMN OF THE C FUNCTION54
THE MAIN CONTROL FUNCTION IN A C	
PROGRAM	55
ARGUMENTS PASS TO MAIN	55
HANDS-ON EXERCISE 2-156
THE BORLAND TURBO C++INTEGRATED DEVELOPMENT ENVIRONMENT (IDE)	56
(1) UNDERSTANDING BORLAND TURBO C++ IDE.56
CREATING A NEW SOURCE FILE58
THE FUNCTION DEFINITION59
REUSABLE C FUNCTIONS.	60
FUNCTION CALLS	62
STORAGE CLASS SCOPE AND VISIBILITY OF A	
FUNCTION65
DATA TYPES AND DATA STRUCTURES	65
TYPE CONVERSION67
TYPE CASTING67
DATA STRUCTURES	68
PROGRAM DEBUGGING/EDITING EXERCISE 2-2.69
OUTPUT70
ARRAYS IN C	71
MULTI-DIMENSIONAL ARRAYS	72
POINTERS	72

PROGRAMMING EXERCISE 2-378
ADDITIONAL EXERCISE78
ADDITIONAL TURBO C++ HANDS-ON EXERCISES 2-2.	79
MULTIFUNCTION AND MULTIFILE PROGRAMS .	.79
ADDITIONAL HANDS-ON EXERCISES 2-380
ADDING ITEMS TO THE PROJECT FILE	88
CREATING THE EXECUTABLE FILE	90
HEADER FILES, FUNCTIONS AND LIBRARIES91
CHARACTER STRINGS	92
CONSTANTS.	93
PROGRAMMING EXERCISE 2-4.	93
POINTERS TO FUNCTIONS	94
THE SCOPE OF A VARIABLE95
THE DURATION OF A VARIABLE97
STORAGE CLASS SPECIFIERS97
AUTOMATIC VARIABLES99
STATIC VARIABLES.99
REGISTER VARIABLES99
EXPRESSIONS AND OPERATORS	99
HOW EXPRESSIONS ARE EVALUATED102
THE ASSIGNMENT OPERATOR.103
THE ARITHMETIC OPERATORS104
THE INCREMENT AND DECREMENT OPERATORS . .	.104
THE LOGICAL RELATIONAL OPERATORS.105

THE BITWISE OPERATORS106
BIT SHIFTS.107
BITWISE LOGICAL OPERATORS108
THE CONDITIONAL OPERATOR.109
THE COMMA OPERATOR109
IMPLEMENTATION-DEFINED BEHAVIOUR.109
PROGRAM STATEMENTS109
LOOPS110
PROGRAM PROBLEM EXAMPLE.114
THE PROBLEM SOLUTION.	114
PROGRAMMING EXERCISE 2- 5.	117
THE BREAK, CONTINUE & RETURN STATEMENTS .	118
THE CASE STRUCTURE121
A GENERAL NESTED CASE STRUCTURE.	122
PROGRAM EXERCISE 2-6.	125
THE SWICH CASE STRUCTURE126
PROGRAMMING EXERCISE 2-7130
RECURSIONS AND ITERATIONS.131
RECURSION131
PROBLEM SOLUTION TECHNIQUES133
AN EXAMPLE PROGRAMMING PROBLEM	134
THE BOOK STOR PROGRAM134
INSTRUCTION134

INTRODUCTION.152
INPUT TEST DATA.152
PROGRAM CONTROL LOGIC153
ADDITIONAL EXERCISES 2-2154
ADDITIONAL EXERCISES 2-3154
ADDITIONAL EXERCISES 2-4155
ADDITIONAL EXERCISES 2-5155
ADDITIONAL EXERCISES 2-6156
INPUT TEST DATA.156
OUTPUT.156
PROGRAM LOGIC DESIGN.157
THE PROGRAM FLOWCHART158
ADDITIONAL EXERCISES 2-7159
INPUT TEST DATA.159
OUTPUT.159
ADDITIONAL EXERCISES 2-8160
INPUT TEST DATA.160
ADDITIONAL EXERCISES 2-9161
ADDITIONAL EXERCISES 2-10162
INSTRUCTIONS.162
INPUT TEST DATA162
OUTPUT162
THE PROGRAM DESIGN.163
THE PROGRAM CONTROL LOGIC.164

ADDITIONAL EXERCISES 2-11.	.165
INPUT TEST DATA.	.165
OUTPUT	.165
THE PROGRAM DESIGN.	.166
THE PROGRAM LOGIC FLOWCHART.	.167
ADDITIONAL EXERCISES.	.168
INSTRUCTIONS.	.168
ADDITIONAL EXERCISES 2-12.	.168
ADDITIONAL EXERCISES 2-13.	.168
ADDITIONAL EXERCISES 2-14.	.170
Prompts and User Input	.170
THE PROGRAM FLOWCHART.	.171
ADDITIONAL EXERCISES 2-15.	.173
INSTRUCTION.	.173
OUTPUT AND SCREEN FORMATS	.173
THE PROGRAM DESIGN.	.174
CHAPTER THREE	.175
INTRODUCTION TO C++ OBJECTS AND IDEAS	.175
LEARNING OBJECTIVES	.175
INTRODUCTION	.176
THE BASIC STRUCTURE OF A C++ PROGRAM.	.177
SOURCE CODE LAYOUT.	.179

PROGRAM EDIT EXERCISE 3-1	182
HOW C++ IMPROVES ON C	183
* COMMENTS	183
* CONSTANTS	183
* TYPE DEFINITIONS AND STRUCTURES	183
* MEMBER ACCESS CONTROL	184
* STRUCTURE DEFINITION.	184
* CALL BY REFERENCE	185
* MEMORY ALLOCATION AND POINTERS	186
* DYNAMIC DECLARATION OF VARIABLES.	187
* EXCEPTION HANDLING	188
* OTHER EXTENSIONS.	188
POSTFIX OPERATORS.	189
HANDS-ON EXERCISES 3-1	191
ANSI C++ PROGRAMMING AND INTEGRATED	191
DEVELOPMENT ENVIRONMENT (IDE)	191
THE MULTIPLICATION PROGRAM	191
THE PROGRAM SPECIFICATIONS	191
THE PROGRAM LOGIC DESIGN	192
THE PROGRAM CONTROL LOGIC	193
CREATING AND COMPILING A SOURCE PROGRAM.	196
THE MAIN CONTROL FUNCTION.	198
STAGES IN THE CREATION OF A C/C++ EXECUTABLE	201
PROGRAM	201

COMPILING THE EXAMPLE PROGRAM	202
LINK TO CREATE A LIBRARY OR EXECUTABLE.	205
FILE	205
RUNING THE EXECUTABLE PROGRAM.	208
USING SAMPLE PROGRAMS ON THE SYSTEM.	209
PROGRAM EXERCISE 3-1	211
THE MICROSOFT VISUAL C++ INTEGRATED.	213
DEVELOPMENT ENVIRONMEMNT	213
UNDERSTANDING THE MS VISUAL C++ IDE.	214
HANDS-ON EXERCISES 3-2	215
NAVIGATING TO CHECK THE CONTENT OF A.	224
PROJECT.	224
ADDITIONAL MSVC HANDS-ON EXERCISES 3-3	227
OBJECT ORIENTED PROGRAMMING WITH	229
ANSI C++.	229
A PRACTICAL UNDERSTANDING OF THE KEY	229
CONCEPTS.	229
USING CLASSES TO BUILD A GRAPHIC SYSTEM.	231
CLASSES	232
CLASS MEMBER ACCESS CONTROL	233
HANDS-ON PROGRAMMING EXERCISE 3-4	236
CREATING THE PROJECT	243
DYNAMIC OBJECT.	244
OBJECTS AND POINTERS	244
MEMORY ALLOCATION	244

MEMORY DEALLOCATION	245
OBJECT SELF POINTER	245
PROGRAMMING EXERCISE 3-2	247
ADDITIONAL EXERCISE	248
ENCAPSULATION AND CLASSES.	249
DECLARING OBJECT INSTANCES OF A CLASS. . .	250
INLINE FUNCTIONS.	251
DURATION.	253
DURATION OF STATIC OBJECTS.	253
CALLING A CLASS MEMBER FUNCTION	254
CONSTRUCTORS AND DESTRUCTORS.	255
OVERLOADED CONSTRUCTORS	257
HOW CLASS IMPLEMENT INFORMATION HIDING.	260
CLASS MEMBER ACCESS CONTROL	261
CLASS PRIVATE SECTION.	261
CLASS PROTECTED SECTION.	261
CLASS PUBLIC SECTION.	261
FRIEND FUNCTIONS AND FRIEND CLASSES . . .	263
PROGRAMMING EXERCISE 3-3	264
THROWING EXCEPTIONS	267
TRY BLOCKS.	267
THE THROW STATEMENT	267
EXCEPTION HANDLERS	268
EXCEPTION SPECIFICATION.	268

DEFAULT EXCEPTION HANDLER	269
OBJECT SELF REFERENCE	269
THE this OPERATOR	269
NESTED CLASS DECLARATION	270
PROGRAMMING EXERCISE 3-4	272
PROGRAMMING EXERCISE 3-5	273
ADDITIONAL PROGRAMMING EXERCISES . . .	274
ADDITIONAL EXERCISES 3-1	274
INTRODUCTION	274
INPUT TEST DATA	274
OUTPUT	275
ADDITIONAL EXERCISES 3-2	275
INPUT TEST DATA	275
THE PROGRAM CONTROL LOGIC	276
ADDITIONAL EXERCISES 3-3	276
ADDITIONAL EXERCISES 3-4	277
ADDITIONAL EXERCISES 3-5	277
ADDITIONAL EXERCISES 3-6	277
ADDITIONAL EXERCISES 3-7	278
INPUT TEST DATA	278
OUTPUT	278
PROGRAM LOGIC DESIGN	279
THE PROGRAM FLOWCHART	280

ADDITIONAL EXERCISES 3-8	281
ADDITIONAL EXERCISES 3-9.	282
ADDITIONAL EXERCISES 3-10	282
ENUMERATION	283
CHAPTER FOUR.	284
INHERITANCE AND POLYMORPHISM IN C++. . .	.284
LEARNING OBJECTIVES	284
INTRODUCTION	285
OOP TERMINOLOGY	287
Abstraction	287
Class	287
Classification	287
Encapsulation	287
Inheritance	287
Instance	287
Instantiation	287
Interface	287
Message	287
Object	287
Polymorphism	288
SENDING MESSAGES	288
INHERITANCE	289

A PRACTICAL EXAMPLE OF INHERITANCE	290
THE POINT CLASS	290
INHERITANCE AND MEMBER ACCESS CONTROL .	292
DEFINING A DERIVED CLASS	292
TURBO C++ IDE HANDS-ON EXERCISE	298
THE PROBLEM SPECIFICATION AND CLASS DEFINITION .	298
THE IMPLEMENTATION SOURCE CODE	300
HEADER FILE CONTENT.	300
THE KEYBOARD AS THE USER INPUT INTERFACE .	301
ENUMERATION OF MENU TYPES	302
DEFINITION AND USE OF THE MENU RECORD STRUCTURE .	303
THE MENU ITEM LIST	304
MENU CLASS DEFINITION MEMBER FUNCTIONS .	304
FUNCTIONS TO GET AND SET INTERNAL DATA .	307
VALUES.	307
ADDITIONAL GLOBAL FUNCTIONS	307
SAMPLE USE OF THE MENU SYSTEM	308
TEST DATA USED IN THE MAIN FUNCTION	309
THE USE OF GLOBAL FUNCTIONS	310
FUNCTION TO INITIALISE THE MENU TABLE	313
FUNCTION TO INITIALISE THE GRAPHIC SYSTEM. .	316
SET OF GLOBAL MENU HANDLING FUNCTIONS .	318
FUNCTION TO CLEAR ALL ACTIVE MENUS.	318
FUNCTION TO DRAW ALL ACTIVE PULL-DOWN MENUS . .	319

FUNCTION TO DRAW AN ACTIVE PULL-DOWN MENU . . .	319
FUNCTION TO DETERMINE THE HIGHEST ACTIVE	
PULL-DOWN MENU	320
KEYBOARD EVENT HANDLING FUNCTIONS . . .	321
GET AND DISPATCH KEYBOARD MESSAGE FUNCTION. . .	321
HANDLING INPUTS FOR EXTENDED KEYS . . .	325
CONTROL LOGIC TO MANAGE THE UP/DOWN	
ARROW KEY PRESS	329
THE ESCAPE OR DELETE KEY INPUT	331
DISPATCHING MESSAGE TO MENU AND DOCUMENT OBJECTS.	
.	331
FUNCTION TO DISPATCH POP-UP MESSAGES	333
USE OF GLOBAL VARIABLES.	334
SYSTEM STARTUP CONSTANTS	334
MAIN CONTROL FUNCTION IN THE TEST PROGRAM. . .	335
THE MAIN CONTROL MESSAGE CHECKING LOOP	336
HANDS-ON EXERCISE WITH THE BORLAND TURBO .	
C++ IDE.	337
PROGRAM EDITING EXERCISE	338
ADDITIONAL PROGRAMMING EXERCISE 4-1 . . .	342
PROGRAMMING EXERCISE 4-2	343
EXTENDING CLASSES BY INHERITANCE	348
HANDS-ON EXERCISE WITH MSVC IDE	352
LEARNING OBJECTIVES	352
USING THE MICROSOFT FOUNDATION CLASS	

LIBRARY	353
THE FOUNDATION CLASS LIBRARIES	354
THE PROGRAM SPECIFICATION	354
HANDS-ON INSTRUCTIONS	354
PLACING STATIC TEXT CONTROL ITEMS IN THE	
DIALOG WINDOW	367
PLACING TEXT EDIT CONTROLS IN THE DIALOG	
WINDOW	371
ADDING A COMMAND BUTTON CONTROL	374
SETTING CONTROL PROPERTIES	375
ADDING A HEADER FILE TO THE PROJECT.	383
EDITING THE APPLICATION DIALOG CLASS	
CREATED BY MSVC IDE.	385
UNDERSTANDING MSVC GENERATED CLASS	
FILE	386
UNDERSTANDING FORMATTED COMMENTS	387
Message Map comment.	387
Data Map comment	388
Virtual Function comment	388
Active Dispatch Map comment	389
PASSING DATA BETWEEN DIALOG CONTROLS AND	
SOURCE CODE	391
DIALOG DATA EXCHANGE	394
DIALOG DATA VALIDATION	396
ADDING EVENT HANDLERS TO CONTROLS	397
DEFAULT EVENT HANDLERS.	397

EDITING THE CONSTRUCTOR FUNCTION	401
EDITING THE DIALOG INITIALISATION FUNCTION .	404
PROGRAM EDITING EXERCISE	410
PROGRAMMING EXERCISE 4-2	411
MULTIPLE INHERITANCES	412
PROGRAMMING EXERCISE 4-3	416
MEMBER ACCESS RESOLUTION	416
POLYMORPHISM	417
OVERLOADING	417
OVERLOAD RESOLUTION.	418
CLASS OVERLOADED FUNCTIONS	422
OVERLOADING OPERATORS IN C++	422
DEFINING ANSI C++ SYMBOL OPERATOR.	423
VIRTUAL FUNCTIONS.	425
EXAMPLE VIRTUAL FUNCTION	427
PROGRAMMING EXERCISE 4-4.	430
PROGRAMMING EXERCISE 4-5	432
MORE VIRTUAL FUNCTION EXAMPLES	432
IMPLEMENTING VIRTUAL FUNCTIONS	433
USING VIRTUAL FUNCTIONS IN A PROGRAM. .	434
PROGRAMMING EXERCISE 4-6	435
FRIEND FUNCTIONS	436
GENERICITY IN OOP SYSTEMS	438

TEMPLATES	439
FUNCTION TEMPLATES	440
GENERIC CLASSES	441
EXAMPLE GENERIC CLASSES	442
One-Way Linked List.	443
IMPLEMENTATION OF A GENERIC CLASS . . .	444
MORE EXAMPLE OF OPERATOR OVERLOADING.	446
CHAPTER FIVE.	448
C++ IOSTREAM LIBRARY FUNCTIONS	448
LEARNING OBJECTIVES.	448
INTRODUCTION	448
THE IOSTREAM LIBRARY	450
THE ISTREAM AND OSTREAM CLASSES	451
THE STANDARD I/O STREAMS	451
INPUT AND OUTPUT.	451
OUTPUT	451
OVERLOADING THE PUT-TO OPERATOR.	454
SYMBOL "<<"	454
PROGRAMMING EXERCISE 5-1.	455
INPUT	455
PROGRAMMING EXERCISE 5-2.	458
FORMATTED OUTPUT	460

FORMAT STATE	461
MANIPULATORS	463
FILE INPUT/OUTPUT	466
THE CONDITION STATES.	470
PROGRAMMING EXERCISE 5-3	470
THE OUTPUT FORMAT.	471
THE CONTROL BREAK LOGIC.	471
ADDITIONAL PROGRAMMING EXERCISE 5-1 .	474
INDEX OF TERMS.	474

INDEX OF TERMS

0

Object-Oriented · 28

A

abort() function · 188, 269
absolute machine instructions · 44
abstract · 11-18, 26, 32, 33, 53, 144, 176, 192, 249, 285, 342, 438, 441, 444
Abstract · 4, 32
abstract class · 438
abstract data type · 32, 249, 438
abstract stream · 450
Abstracted behaviour · 21
abstracted procedure · 16
abstraction · 3-18, 24-27, 40, 44, 60, 193, 249, 260, 285, 289
Abstraction · 4-11, 287
accelerators · 400
access control · 233-242, 261, 262, 284, 285, 292-294, 348, 350, 363
access control attribute · 233, 261
access level · 233, 292, 416
access modifier · 262, 292, 293, 343, 351
access right · 272, 351, 416
access rules · 249, 251, 271, 272, 289, 386, 422
access scope · 350
access specification · 184, 263, 264, 291, 292, 363, 386
access specifications · 263, 292
access specifiers · 249, 263, 292
accessibility · 97, 99, 192, 233, 261, 292, 422
accumulators · 104
Ada · 28, 34
addition operator · 46, 212
address value · 72-75, 92, 319, 335
algorithmic solution · 1, 193-195
algorithms · 1, 20-24, 31, 133, 193, 246, 247
allocated memory · 73, 74, 97, 99, 244, 245, 253, 336
allocated storage · *See* memory allocation
alphanumeric characters · 42
alternative processing · *See* case structure.
angled brackets · 196, 222, 308, 439
ANSI · 21, 35-42, 50-72, 78, 89-114, 124, 133, 134, 151, 175-191, 219-233, 244, 246, 252, 257, 264-274, 284-292, 298, 300, 405, 417, 423-427
ANSI C · 21, 35-42, 50-72, 78, 89-114, 124, 133, 134, 151, 175-191, 224-233, 246, 252,

257, 264-274, 284-292, 298, 300, 405, 417, 423-427
APL · 43
append mode · 466, 468
application configuration · 358-360, 385
AppWizard · 352-362, 385, 386
argc · 55, 220
argument · 48-55, 63-66, 89, 109, 132, 188, 211, 220, 252-260, 267, 268, 294, 304, 416-430, 441, 453-469
argument list · 64, 109, 257
arguments · 14, 48-53, 63, 94, 109, 211, 256, 257, 265, 294, 418-425, 440, 441
argv · 55, 220, 469
arithmetic expression · 71, 72, 104
array · 12, 30, 48, 55, 68-93, 101, 109, 134, 144-148, 174, 187, 189, 195, 237-251, 266, 272, 273, 286, 303-316, 352, 390, 403-410, 420, 426, 441, 442, 474
array name · 71, 74, 189
array of characters · 92, 307
array of records · 71, 237, 272, 403
array subscripts · 74
arrays · 37, 68-74, 92, 94, 173, 183, 187, 195
ASCII value · 93, 301, 302, 321, 325, 336
assembler · 43
assembly language program · 44
assert() function · 188
assertion statements · 219
assignment operator · 46, 99-104, 452
associativity · 99-103, 109
asynchronous operation · 31
atof() function · 197
attributes of objects · 32
authorized interface · 184, 233, 261
automatic · 97, 99, 266
automatic conversion · 472
automatic deallocation · 256

B

backing storage · 273
base class · 229, 289-295, 348, 360, 413, 451
base constructor · 294
BASIC · 43, 62, 177, 417
binary · 100-104, 148, 149, 218, 356, 438, 441, 453-460
binary form · 453
binary operators · 103
binary search · 148, 149, 410
binding · 26, 34, 35, 242, 284, 417, 426, 427
biological systems · 1, 3, 4, 5, 9
BIOS · 44, 301
bit field · 462

bit wise operation ·106
bitwise operators ·106, 108
Borland IDE ·56, 224, 290
borland turbo c++ ·56, 337
bottom-up design ·33, 37, 133, 151, 229
break statement ·110, 118, 129
bubble sort algorithm ·143, 148
buffer ·451-454
Button control ·374

C

called function ·53, 62-64, 109, 418
calling function ·53, 54, 62-64, 82, 120, 121, 181, 195, 312, 321, 333
Caption ·376, 379
case statement ·129, 321-327
case structure ·60, 109, 111, 121-129, 140, 323-326, 332
ceil() function ·347
cerr ·268, 451, 452
character array ·304
character constant ·93
character conversions ·452
character pointer ·92, 443
check box ·89, 358, 376, 377, 395
child class ·289-294. *See derived class*
cin function ·181
cin operator ·456
Circumstantial Abstraction ·10
class ·*See template*
class definition ·23, 223, 235, 274, 290-292, 298, 363, 412, 418, 446
class definition file ·235, 274
class hierarchies ·285
class hierarchy ·292, 295, 450
class libraries ·364
class members ·271
class object ·453-455, 466
class template ·11, 22, 23, 32, 184, 232, 249, 250, 284, 441
class type ·454, 455
Class View ·224, 225
classes ·22-45, 93, 94, 155, 178, 217, 229-234, 249-277, 287-297, 348, 359-364, 385, 412-416, 425, 428, 435-441
classification ·22, 30-35, 181, 230, 283-287
client edge ·374, 377
clog ·451, 452
close() function ·468
clrscr() function ·197, 225
Clu ·28, 34
code generation ·48
code reusability ·60
code re-use ·27
coded information ·44
comma operator ·109, 211
command button ·352, 371, 373, 383, 392, 410
command line ·55, 210
command prompt ·55, 56
command statements ·38, 126, 129, 137, 194, 196
common attributes ·285
common behaviour ·23, 32, 33
common properties ·*See classification.*
compile ·35-44, 56-61, 69, 77-79, 85-93, 117, 130, 148, 175-178, 184, 191, 202-210, 222-226, 235, 243, 244, 253, 264, 283, 337, 343, 356, 359, 365, 406-411, 426-440
compile time ·35, 43-48, 61, 69, 77, 235, 244, 253, 283, 356, 359, 426, 427, 440
compiler ·40-77, 91-109, 177-186, 195-204, 224-244, 250-259, 269, 284, 294, 305, 310, 401, 402, 416-418, 425-432, 440, 441
complexity ·1-9, 16, 19, 120, 133, 139, 151, 192, 193, 287, 289
component field ·69
component members ·184
component select operator ·21, 235, 259
compound statement ·124
concatenated ·424, 453
concatenation ·92, 417, 423, 424, 446
concise representation ·9
condition flag ·470
conditional compilation ·*See Rational pre-processing*
conditional operator ·101, 109
configuration ·216, 352-363
configuration options ·216, 357
conformance ·27
console application ·214-217, 227, 228
console output ·449
constants ·41-45, 84, 180, 334
constraint ·31
constrea ·449, 450
constructor ·175, 240-266, 273, 293-296, 304-307, 343, 386, 394, 401, 403, 414
constructor function ·256, 344, 394, 403, 452, 455
context resolution ·26
continue statement ·119, 120
control break logic ·470-472
control expression ·458
control field ·*See key field.*
control format ·70
control ID ·398, 400
control item ·367-382, 392-398, 405, 410
control logic ·1-14, 60, 81, 109-150, 179, 192-198, 221, 274, 310-316, 329, 342, 343
control specification ·262, 363
control statement ·69, 81, 102-109, 139
control structure ·37, 78, 82, 110, 112, 118-126, 144-151, 229, 322, 332, 338

control structures · 11, 37, 63, 109, 110
Control-Oriented · 11
conversion · 67, 114, 117, 224, 273, 452, 460, 464
conversion function · 273
converted type · 460
Copy function · 221, 369
correctness · 35, 127, 139
counters · 104
cout function · 178-182, 197, 203, 223, 451-464
CPU · 99, 104

D

data declaration · 181, 254
data hiding · See information hiding
data management · 41, 109
data member · 73, 183, 231, 232, 239-242, 259, 260, 293, 294, 303-316, 342-350, 387, 405, 414, 436, 455
data structure · 9, 19-24, 30, 37, 40, 68, 74, 186, 438-443
data type · 9, 32-42, 62-68, 92, 93, 99, 102, 107, 108, 183, 184, 193, 195, 232, 249, 250, 258, 260, 300, 422, 438-444
debug · 56, 57, 88-93, 148, 191, 208, 212, 218, 219, 435
debug mode · 458
decimal · 93, 107, 181, 191, 410, 460-464
declaration · 50-52, 59-65, 71, 74, 80, 81, 90-99, 109, 181-189, 211, 230-232, 244, 251-257, 263-271, 283, 294, 301, 383-390, 417-429, 439
decrement operator · 189
default · 55, 59, 95, 99, 129, 181, 200-206, 215-218, 231-233, 240, 241, 255-262, 269, 273, 292, 294, 304, 313, 316, 323, 334, 348-356, 374-387, 397-406, 420, 421, 451, 459-461, 468
default access modifier · 292
default arguments · 255
default constructor · 256, 294
default event · 397
default file name · 59
default statement · 129
default values · 202, 240, 257, 273, 304, 381
defined interface · 7, 10, 16-21, 30
definition file · 363, 388-390
delegation · 27
Delimiter · 41
dereference · 73, 189
dereference operation · See *indirection*
derived class · 261, 291-294
derived classes · 285-292
destructor function · 245, 255-258
device context · 270, 353

dialog base application · 286, 352-357, 385
dialog box · 83, 89, 200-207, 354, 379, 393-410
dialog components · 352
dialog control · 368, 383-397, 404-406
Dialog Data Exchange · 394
Dialog Data Validation · 394, 396
dialog initialisation · 404
dialog item · 366-368, 375-378, 391
dialog resource · 362, 366, 391
dialog window · 38, 82-89, 202-228, 352-409
directory system · 83
display system · 56, 93, 114, 143, 177, 231, 232, 268, 270, 289, 290, 304, 309, 323, 353, 425
document view · 298-306, 312-323, 329, 331, 342-346
document window · 58, 59, 196, 203, 343, 357-398, 403
do-while loop · 113, 195
duration · 53, 65, 97, 187, 245, 253, 254, 277, 403, 408
dynamic · 6, 7, 72, 73, 186, 249-256, 287, 342, 343, 355, 426
dynamic behaviour · 6, 7
Dynamic binding · 35, 426
dynamic structures · 186
dynamically allocated · 245

E

early binding · 426
else statement · 81, 121-123, 328, 406
encapsulate · 8, 62, 184, 232, 249, 289, 308, 313, 321, 338, 339
encapsulated · 1, 8-12, 19, 30-35, 235, 249, 287, 290, 320
encapsulating · 260, 290
encapsulation · 3-19, 28, 30, 63, 230, 249, 250, 263, 289, 338
Encapsulation · 9-12, 18, 30, 35, 53, 175, 229, 249, 250, 285, 287
enclosed system · 8
end-of-file value · 459
entry address · 42, 94, 230
enum · 55, 67, 185
enumerated mnemonics · 461
enumerated type · 125, 185, 232, 300, 302, 345, 374, 384, 420, 468. See *enum*
enumerators · 271, 272
error message · 95, 148, 173, 197, 247, 268, 273, 407, 410
escape sequence · 41, 42, 89, 93, 178
Esquel · 48
event handler · 397-400, 407
event handlers · 371-375, 397-400, 411
Event handling · 410

exception · 27, 57, 175, 188, 237, 248, 267, 269
exception specification · 268, 269
exe file · 56, 82, 220
executable program · 43, 55, 73, 79, 87, 88, 148, 182, 201-214, 220, 221, 243, 298
executable programs · 56, 58, 175, 176, 213
exit statements · 81
exit() function · 188, 268, 311
explicit initialisation · 97, 254
explicitly delete · 245
expressions · 46-48, 65, 75, 99-106, 190
extended keyboard · 312, 321
Extended Keys · 325
Extended Styles · 377
extern · 55, 91-97, 254, 272
external behaviour · 31, 32
external interface · 22, 30, 32
External static variables · 99
external variable · 73, 95, 96, 254
extract from · 448, 456
extraction operator · 451-460

F

factorial · 131
fgets() function · 78
field width · 460-464
File inclusion · 48
file path · 200
file scope · 253
File View · 225, 226
fill() function · 461
flexibility · 17, 26, 27, 35, 356
flexible sharing · 30
flow of control · 6, 15, 81, 105, 111, 126, 139
flow of Information · 5
flowchart · 11, 15, 37, 111-117, 127, 133, 139-144, 150-157, 163-166, 187, 194-198, 275, 279, 310, 315, 342, 353
Flowcharting Software system · *See Logic Coder.*
flushing · 453
for · 471
formal parameters · 49
format flags · 461-463
format state flags · 461
formatted · 78, 387, 388, 450-452
formatted comment · 387, 388
formatted output · 450
for-next loop · 60, 70, 314
for-next statement · 110
foundation class libraries · 354, 360
foundation classes · 361, 364
fprintf() function · 78
free memory · 73, 247

friend · 175, 184, 233, 261-264, 270, 272, 308, 350, 429, 436-438
friend access specifier · 233, 263
friend class · 437
friend function · 263, 272, 308, 350, 436, 455
friend specification · 264
fscanf() function · 78
function · 4-15, 36-109, 120, 128-137, 143-150, 175-204, 211-275, 287-294, 301-369, 378-389, 395-445
function body · *See function definition.*
function call · 60, 64, 71, 99, 104, 131, 189-197, 212, 223, 224, 242, 251, 310-323, 331, 333, 405, 416, 418, 427, 441
Function call · 37, 101
function call operator · 189
function definition · 37, 50-65, 179, 187, 197, 223, 242, 245, 252, 257, 389, 405, 416, 425, 427, 436
function dispatcher · 303
function identifiers · 41
function main · 55, 321
function name · *See function identifier.*
function parameter · 53, 95, 104
function prototype · 37, 148, 232, 249. *See prototype.*
function signature · *See function prototype.*
functional components · 230
functional decomposition · 16, 37, 133, 134
functions · 5, 16, 19, 30-42, 50-54, 62-73, 79-85, 91-99, 130, 139, 140, 150, 175, 178, 184-188, 195-214, 224, 225, 232-274, 284-321, 332-354, 362, 364, 370, 373, 386, 394-400, 406-447

G

gcount() function · 459, 460
generalisation · 5, 6, 16, 139
generalise function · 136
Generic · 4, 193, 441, 444
generic class · 438, 441
generic control structure · 126
generic data types · 439
generic description · 34
generic modules · 1
generic reusable function · 191
generic solution · 1
generic system · 9, 349, 438
genericity · 4, 9, 27, 28, 34, 68, 193, 284, 417, 438
get() function · 443, 458, 459
getch() function · 87, 130, 197, 301, 336, 458
get-from · 455

getline() function · 458, 459
global · 48, 51, 60, 73, 79-81, 91-99, 148, 187,
196, 219, 254, 255, 271, 300, 307, 313-326,
334, 335, 390, 403, 410
global functions · 300
global macros · 300
global scope · 97
global variables · 73, 80, 81, 96, 97, 313, 316,
326, 334, 403
graphic code · 79
graphic device · 298
graphic system · 79, 82, 231, 290, 311-317,
334, 335
Graphical User Interface · 353 *See GUI*
graphics function · 87
graphics hardware · 81
graphics library routines · 87
graphics mode · 85, 87
GUI · 6, 7, 56-59, 214, 217

H

handle · 188, 247, 249, 267, 268, 285, 366,
367
handler code · 352
header file · 69, 84, 148, 177, 197, 235-238,
266, 273, 291, 293, 300-308, 316, 337, 343-
345, 362, 363, 383-388, 401, 416, 426, 430,
451, 457-466
heap memory · *See the heap*
hexadecimal · 42, 93, 457-464
hierarchical decomposition · 13
hierarchy chart · 14-16, 133-144, 150, 415
hierarchy of interdependency · 5
high-level language · 43
Highly Specialised · 4

I

I/O stream · 448
IDE · 36-39, 56-59, 69, 79-91, 175, 176, 191-
229, 236, 243, 264, 265, 285-290, 298, 308,
337, 347, 352-366, 371-376, 382-391, 400-
409
identifier · 41, 47, 212, 456, 468.
if statement · 49, 81, 106, 120-126, 132, 327,
328, 336, 406
if-else structure · *See case structure.*
ifstream object · 467
illegal assignment · 66
implementation · 1, 7-40, 56, 66, 81, 107-
127, 133-144, 150, 151, 175, 177, 192-199,
213, 219-260, 266, 273, 284-294, 300, 309,
313, 327, 337-339, 344-352, 362-366, 387-
389, 395-408, 416, 423-426, 433-446
implementation code · 237, 238

implementation file · 235, 237, 274, 364,
400
include directive · 90
include file · 61, 178, 197, 221, 235, 344, 345,
402, 435
inclusion polymorphism · 33-35
increment operator · 189
incremental solution · 260
indentation · 123
indented · *See indentation*
index · 48, 71, 92, 101, 146, 187, 189, 312,
319-323, 410, 441
indirection · 73
information hiding · 7, 19, 21, 31, 184, 233,
243, 250, 260-263, 289, 436
inheritance · 11, 12, 24-28, 36, 175, 229, 231,
237, 244, 248, 284-292, 298, 309, 338, 342,
348, 352, 385, 412, 415, 432
inherited class · 261
initialisation · 304, 310, 316, 342, 401, 414
initialisers · 69
initializing values · 241, 257
inline · 175, 235, 237, 251-256, 263, 271, 273,
293
inline definition · 235
inline expansion · 252
input argument · 50
input mode · 467
input parameter · 51-55, 65, 66, 94, 104, 131,
186, 195, 197, 211, 240-251, 313, 331, 405,
417, 433, 441
input stream · 178, 450, 451, 459
insertion · 92, 144, 451-455
insertion operator · 454
instances · 23, 63, 122, 178-185, 224-236,
250, 251, 266, 287, 295, 308, 335, 414, 428
instantiation · 22, 23, 184, 287, 340, 341
instantiation process · 23
Integrated Development Environment ·
See IDE.
intelligent software · 4
intelligent system · 3, 5
interdependencies · 30
interface · 5-31, 43, 53, 54, 178, 195, 211, 215,
233-243, 249, 250, 270, 300, 301, 321, 340,
357-365, 371, 449, 451
Interface · 5, 7, 23, 30, 56, 57, 214, 287
intermediate files · 220, 228
internal details · 20, 133
internal documentation · 180, 308
internal state · 7, 30, 31
interpretation · 6, 30-35, 179, 289, 312
interpreter · 43, 55
interrelated components · 5
invocation methods · 20
ios · 450-452, 460-468
iostream · 70, 177-181, 448-470

iostream library ·450, 451
istream ·450-460, 466
iteration ·119, 241

J

justification ·461

K

kbhit() function ·301, 336
keyword ·45, 66, 84, 96, 99, 184-186, 231-237, 251-254, 261, 262, 269, 270, 419, 429, 430
Keyword ·41-45, 55, 233, 261, 420
keyword const ·183

L

labels ·13, 45, 376
language alphabet ·45
Language extension ·48
largeness ·1-8, 14, 133, 151
left shift ·107
left shift operation ·107
level of precedence ·99
lexical ·30, 45, 46
lexical analyser ·45
Lexical analysis ·47
lib file ·56, 205, 206, 234, 300, 337
lib files ·202
library routines ·202
linear analysis ·46
link editing ·49
link list ·342, 437-443
linked list ·68, 186, 246-248, 269, 273, 305, 342, 441, 443
link-editor ·49
linker ·56, 205, 243, 337
LISP ·43
List ·9, 55, 68, 89, 137, 354, 397, 398, 443
lists handling ·194
loader ·49
local ·*See local variable.*
local class ·272
local copy ·63
local variable ·95
log file ·268
Logic Coder ·6, 106, 187, 192, 199
logic of a system ·5
logical operator ·103, 106
logical relational operators ·103, 106
LogicCoder ·11, 129, 141, 310, 315, 353
longjmp() function ·188
loop logic structure ·110-113, 119, 121, 151

Loops ·109

M

machine code ·43, 49, 54, 208, 219, 257, 426
machine executable file ·202, 205
machine language ·43
machine level ·7, 40, 41, 42, 54, 94, 110
macro definition ·48, 49, 180, 183, 440
Macro processing ·48
macroprocessor ·49
main memory ·471
main() function ·95, 194, 256, 312, 319
maintainability ·1-8, 35, 40, 62, 133, 179, 229
maintenance ·3-8, 37, 187
malloc() function ·72, 78
manipulation operators ·103
manipulator ·418, 453-465
manipulators ·448, 454, 463, 464
member data ·183, 184, 233, 251-261, 292, 303, 306, 312, 339, 340, 350, 351, 364, 385, 386, 390, 403
member function ·181, 235, 242, 244, 252-257, 264, 266, 272, 273, 327, 338-344, 350, 351, 363, 364, 387, 397-400, 416, 425-429
member functions ·184, 233-237, 249, 255, 261, 263, 293, 300, 308, 313, 340, 350, 351, 385, 422, 429, 432
member select operator ·189, 254
member variables ·272, 304, 351, 366, 388, 389, 401
memory allocation ·66, 72, 186, 247, 253-258, 403
memory buffered input ·449
memory reference ·100, 102, 186, 237
menu driven program ·134
menu items ·58, 215, 303-308, 314, 316, 332, 338, 342, 400
menu system ·285, 290, 298-322, 331-344, 438
menus ·12, 38, 218, 290, 298-303, 309-321, 327-331, 342, 384
message ·11, 12, 20-23, 48, 53, 73, 79, 87, 95, 114, 134, 144-154, 160, 173-178, 191-202, 212, 223-229, 243, 247, 268, 288, 299-312, 321-325, 331-342, 357, 386, 389, 396-398, 407, 410
message box ·311, 410
message dispatch ·299, 335, 397
message dispatcher ·307, 313, 335
message handler ·383, 397
message handles ·249
message handling ·383, 387, 397
message ID ·398, 399
message map ·387-398
message passing ·20, 242, 274

messages ·11-20, 90, 203, 229, 230, 238, 243,
249, 285-290, 312, 320, 323, 332-342, 438
method binding ·24, 26
Microsoft Foundation class ·243
Microsoft Visual C++ ·*See MSVC.*
Microsoft Visual C++ IDE ·36, 37, 56, 175,
176. *See MSVC IDE*
minimisation ·144
modular system ·1
modularity ·21, 27, 35
modulus operator ·104
mouse pointer ·12, 215, 224, 225, 366-372,
408
MSVC ·56, 214-229, 236, 243, 285, 290, 352-
376, 382-391, 403-409
MSVC IDE ·220-224, 357
multi-file projects ·56, 175
multiple assignments ·104
multiple inheritance ·284-294, 348, 428, 451
multiple return ·81
multiplication operator ·46

N

name binding ·272
nested case structure ·123, 124, 126, 324
nested class ·270, 271, 272
nested if-else ·121, 322
nested loop ·110
nested loops ·121, 144
nested pointer ·76
nested while loop ·336
new() function ·244
new operator ·352
newline character ·70, 178, 454, 459
node ·46, 442-444
non-machine instruction ·43
null character ·42, 93, 456-459
null pointer ·187
numeric constant ·46
numeric values ·136, 410

O

obj file ·56, 202-208, 237, 243, 356
object ·2-45, 56, 65, 69, 77, 97, 133, 178-189,
202-206, 215-273, 285-312, 318-349, 361,
362, 386-390, 401-408, 414, 416, 422-436,
442
object instance ·242
object classification ·22
object constructor ·240
object interaction ·20, 21
object module ·92
Object Oriented Computing ·1

Object Oriented Programming ·1, 11, 18,
36, 40, 68, 151, 175, 176, 183, 195, 229, 238,
260, 274, 284, 285, 298, 354
object-oriented ·2, 3, 11-35, 133
object-oriented system ·3, 11, 17, 20, 26-33
octal ·42, 93, 460-464
ofstream object ·466, 468
on-line information ·84
OOP ·11, 40, 56, 175, 176, 188, 229-234, 243,
250-260, 284-292, 349, 417, 438, 445
operands ·48, 101
operating system ·54, 55, 79, 80, 188, 191,
213-220, 237, 244, 245, 253, 268, 269, 301,
311, 317, 335, 361, 365, 387, 389, 397
operational interface ·19, 21, 31
operator ·21, 41-48, 70, 99-109, 122, 175-189,
204, 212, 244, 253-259, 266-272, 284, 352,
416-427, 433, 441, 447-461, 468, 470
operator symbols ·45, 422
optimization ·202
optimized ·194, 219, 220
optimized algorithms ·194
order of precedence ·102-104, 212
orderly evolution ·289
ostream ·450-459, 464-467
outer class ·272
Output formatting ·461
output mode ·466
output stream ·178, 268, 448-453, 467
output streams ·450
overload ·257, 417-428, 446-454, 460
overload resolution ·417-422
overloaded ·26, 34, 187, 257, 284, 417-424,
440, 446, 447
overloaded definition ·452
overloaded function ·417, 418
overloading ·26, 34, 35, 417-424, 446
Overloading ·34, 188, 284, 417

P

padding ·72
parameter declarations ·419-421
parameter list ·51-59, 64, 195, 197, 418, 420
parameter variable ·186, 240, 304, 326
parameterised type ·439
parameters ·20, 49-59, 86, 87, 144, 186, 195,
240, 251, 311, 418, 421, 440
parametric polymorphism ·34
parent class ·292, 294. *See inherited class*
parse tree ·46
parsing ·*See syntax analysis*
Paste ·369-373
perceptual view ·18
pixel ·231, 290, 303, 314, 334
pointer ·12, 65-78, 94, 101, 186-189, 197,
224, 237-254, 266-269, 301, 303, 311-329,

335, 342, 373, 390, 403, 405, 420-428, 436, 442-444

pointer variable · 75, 76, 186, 190, 245

pointers · 30, 37, 55, 68, 73-76, 94, 146, 186, 187, 195, 197, 246, 251, 269, 271, 319, 334, 420, 425, 437-443

polymorphic · 12, 26, 33-35, 285, 349, 356, 416, 428

polymorphic objects · 12, 34

polymorphism · 11, 12, 24-35, 244, 175, 229, 231, 284-290, 349, 417, 438, 449

pop-up menu · 298-303, 312, 315, 322-336, 375, 392, 398

Portability Issues · 109

postfix notation · 104, 105

post-increment · 105

precision · 40, 66, 67, 180, 181, 204, 223, 461-464

precision() function · 181, 461, 462

predefined process · 117

predicate · *See sets.*

prefix notation · 104

pre-increment · 105

pre-process · *See pre-processor.*

pre-process directive · 401, 457

pre-processor · 48, 76, 401

printf · 41, 70-77, 95, 105, 140

printf() function · 95, 105

private · 165, 175, 184, 231-234, 242, 259, 261-266, 292-294, 308, 340, 348-351, 386-390, 436, 437

private data · 184, 436, 455

procedural - programming · 13, 14

procedural language · 133

procedural statements · 13

procedures · 12-16, 30, 31, 46

program code · 70, 76, 80, 148, 180, 187, 232, 243, 351, 426, 457, 458, 464

program control · 81, 82

program development cycle · 151

program editing · 163, 290, 338, 410

Program Logic Design · 157, 192, 279

program reliability · 62

program specification · 114, 116, 126, 130, 152-158, 191-194, 274-279, 286, 300, 337, 361, 364

project configuration · 243

project file · 82-88, 130, 216, 243, 337, 343, 352, 356

prompt · 12, 114, 135, 144, 170, 173, 181, 191-197, 210, 211

Properties dialog · 375-378

protected · 14, 30, 175, 184, 233, 249, 261, 263, 292, 293, 308, 350, 351, 362

protected interface · 30

prototype · 50-55, 63, 64, 80, 92, 120, 148, 178, 195, 197, 211, 219, 224, 225, 241, 250, 405, 417, 450, 453, 459, 462, 468

public · 175, 184, 231-234, 242, 245, 255-262, 268, 291-293, 307, 343-351, 363, 386-389, 405, 416

public member functions · 233

pull-down menu · 58, 59, 82, 84, 200-215, 227, 228, 298-308, 312-336, 343, 354, 356, 409

Punctuator · 41, 139, 179

put() function · 453, 459

put-to · 204, 448-460

Q

quality assurance · 8

R

raise an exception · 188

raised exception · 267

RAM · 7, 94

Rational pre-processing · 48

read() function · 459, 460

readability · 124, 179

real number · 48

record · 471, 472

record structure · 68-71, 125, 183, 236, 237, 272, 301, 303, 352, 383, 384, 401, 402

Records · 68, 388

recursion · 47, 119, 131

recursive call · *See recursion.*

Recursive looping · 131

redirect input · 449

redundant coding · 62

reference operator · 78, 186

reference parameter · 185

reference variables · 185, 187

register variable · 99

register variables · 97, 99, 110

registers · 99, 104

Relativistic Abstraction · 10

Release version · 220

relocatable addresses · 49

relocatable code · 94

relocatable machine instructions · 44, 49

reserved words · 55, 184

resource · 3-9, 41, 226, 258, 355-367, 391

return · 20, 42, 48-66, 79-82, 100-107, 120, 131, 144, 178-181, 195, 197, 211, 220-224, 245-257, 266-273, 306, 307, 321-323, 333, 338, 365, 405, 407, 417, 418, 425-433, 440-444

return statement · 53, 62, 81, 82, 120, 222, 268, 307, 405

return value ·104
 reusability ·1, 144, 193-197, 229
 reusability of components ·1
 reusable function ·62, 117
 rich text edit ·371

 right shift operation ·107, 108
 ROM ·94, 310, 337
 row vector ·72
 rule of precedence ·101, 102
 run time ·35, 43, 44, 50, 73, 244, 253, 356,
 359, 375, 379, 417, 426-431
 runtime binding ·426
 run-time error ·269

S

scope ·37, 48-53, 60, 65, 95, 97, 184, 233,
 242, 255-263, 270-272, 390, 403, 414-427,
 433
 Scope ·65
 scope resolution operator ·253, 259, 270,
 416, 425, 427, 433
 scope rule ·37
 scoping rules ·270, 272
 scroll bar ·352, 405
 search ·92, 134, 137, 148-150, 194, 247, 410
 secondary memory ·474
 seekg() function ·468
 seekp() function ·468
 self reference ·21, 269
 semantic ·13, 28-32, 40, 48, 77, 176
 sending a message ·266
 sequential file ·471
 setf() function ·461, 462
 setjmp() function ·188
 sets ·31, 32, 89, 101, 316, 358
 setw() function ·456, 474
 shortcut toolbar ·220-226, 369-373, 385
 Simula ·28
 single entry point ·110, 118-126
 single exit point ·110, 118-126
 single exit point principle ·118, 119
 sizeof operator ·76, 187
 sizeof() function ·319, 403
 sizeof() operator ·77, 352
 Small Talk ·12, 28, 36, 43, 184
 software engineering ·1, 27, 238
 software system ·3-15, 40, 44, 133, 138, 175,
 229, 298, 310
 software systems ·1-17, 28, 30, 36, 40, 127,
 128, 139, 144, 151, 176, 195, 197, 213, 238
 sort ·143-149, 194, 410
 sorting function ·148
 source files ·56, 89, 96, 353, 360

 source program ·37-50, 56-65, 78, 81, 94-96,
 106-110, 117-129, 139, 179-211, 219-230,
 236, 238, 265, 274, 313, 315, 392-397
 specialisation ·5, 25, 144, 285, 289
 specialised function ·5, 139
 spin button ·352
 sqrt() function ·347
 stack ·30-34, 68, 119, 131, 438-441
 stack overhead ·251
 standard control structures ·110
 standard error handling ·268
 standard input ·181, 449, 456, 466, 467
 standard input device ·181
 standard libraries ·449
 standard library files ·202
 standard output device ·178, 181, 457, 459,
 466
 standard output stream ·178, 452
 starting address ·94
 state variables ·22-28, 460
 statement delimiter ·46
 statements ·13, 37, 46, 48, 70, 81, 82, 105-
 113, 119-132, 139, 177, 178, 211, 238, 252,
 267, 335, 406
 static ·6, 7, 55, 97, 99, 253, 254, 271, 272, 287,
 342, 366-377, 422-429
 Static behaviour ·6
 static binding ·426
 static description ·6, 7
 static edge ·374-378
 static objects ·*See static.*
 static text ·367-371
 Static Text control ·369-382, 405
 static variable ·97, 99, 253, 254, 272, 342
 stepwise refinement ·13
 strcmp() function ·92
 stream ·448-474
 stream library ·450
 stream objects ·448, 451, 469, 474
 stream operators ·448
 stream output ·452
 stream reference ·463
 string ·41-49, 55, 70, 92, 136, 144, 148, 178,
 181, 197, 202, 218, 224, 258, 268, 301-315,
 395, 396, 410-424, 446, 447
 string constant ·93
 string handling ·194
 string input ·449
 string variable ·92
 String-literal ·41
 Strings ·68
 strlen() function ·197, 224
 struct ·55, 68, 69, 184, 185, 231-237, 254,
 261-266, 292, 301, 336, 390, 440
 structure component ·69
 structure definition ·68, 125, 183-185, 231,
 237, 246, 350, 385

structure members · 233
structure tag · 68, 184, 185
structure types · 37
structuring techniques · 13
Styles property · 376, 395
sub-routine · 41, 42, 94
subroutines · 191, 205
subscript · *See index.*
subsidiary menu · 38
swap function · 186
switch-case structure · 126, 321, 322, 331, 332, 345
symbol table · 48
symbolic constant · 180
syntactic meaning · 177
syntax · 13, 20, 32-48, 67, 117, 179, 187, 252, 254, 269, 348, 416, 428-439, 450, 464
syntax analyser · 46
syntax analysis · 45
synthesise · 45
systems memory · *See heap*
System's Response · 8

T

tab control sequence · 70
target · *See object.*
tellg() function · 468
template · 22, 28, 32, 49, 68, 187, 197, 232, 233, 250, 439-445
template declaration · 439
templates · 22, 23, 32, 438-441
temporary memory locations · *See register*
terminal display · 178, 449
test data · 78, 125, 134, 152-283, 309, 310, 352-354, 403
text document · 58, 80, 86, 196, 215, 385
text edit · 200, 215, 352, 371, 379, 381, 399-410
Text Edit control · 372-374, 382, 392, 395, 410
text editor · 12, 56, 57, 69, 85, 90, 93, 191, 200-203, 210-222, 264, 401
text file · 78-85, 191, 212, 220, 221, 237, 239, 264, 272, 337
text property · 352
the heap · 187, 244-247, 253, 272, 319, 336, 352, 403, 410
this pointer · 244, 245, 269
throw expression · 188, 267-269
throw statement · 267
token · 41-48
toolbar buttons · 400
toolbox · 366-374, 397
top-down design · 33, 37, 133, 134, 192
translator · 43
tree algorithms · 194

Trees · 9, 68
trigger · 31
try block · 188, 267, 273
try statement · 267
Type · 45, 48, 65-67, 80, 85, 91, 101, 164-168, 199-204, 216, 222, 224, 337, 350, 385, 393-402, 408, 441, 442
type cast · 67, 75, 195, 224, 404
Type Casting · 67
type checking · 34, 35, 48, 284, 417, 426, 440
type conversion · 67, 103, 188
type declarator · 230
type definition · 283, 287
type identifiers · 41
type information · 48
type list · 269
type modifiers · 66
typedef · 55, 184, 185, 390, 419

U

understandability · 179
union · 55, 159, 189, 232-234, 255, 281
unsetf() function · 461
user interface · 3, 19, 62, 352, 353, 367, 385

V

validation checks · 195
validation function · 144
validation loop · 117, 140, 150
variable name · 471
variable names · 45, 63, 64, 186, 226, 261, 394
variables classification · 181
virtual base class · 294, 295
virtual child class · *See virtual base class*
virtual destructors · 255
virtual function · 388, 389, 400, 425-429
virtual keyword · 425
virtual Print() function · 432
virtual specification · 362
virtual specifier · 429
visibility · 31, 95, 260
visibility interface · *See external behaviour*

W

well design · 193
while loop · 69-71, 113, 118, 204, 313, 314, 336, 456, 458
while statement · 110, 140
white space · 179, 454-464
white space characters · 179
width() function 70, 461

windows compatible · 353, 354
windows mode · 57
Windows mode · 57, 217-221
write() function · 453, 459

