

C++ Fourth Edition The Complete Reference

The Definitive Guide to C++

Master programmer and best-selling author Herb Schildt has updated and expanded his classic reference to C++. Using expertly crafted explanation, insider tips, and hundreds of examples, he explains and demonstrates every aspect of C++. You'll find details on the entire C++ language, including its keywords, operators, preprocessor directives,

and libraries. There is even a synopsis of the extended keywords used for .NET programming. Of course, everything is presented in the clear, crisp, uncompromising style that has made Herb Schildt the choice of millions. Whether you're a beginning programmer or a seasoned pro, the answers to all your C++ questions can be found in this lasting resource.

ed coverage includes:

• Data types and operators

• Control statements

• Functions

• Classes and objects

• Constructors and destructors

• Function and operator overloading

• Inheritance

• Virtual functions

• Namespaces

- Templates
- Exception handling
- The I/O library
- The Standard Template Library (STL)
- Containers, algorithms, and iterators
- Principles of object-oriented programming (OOP)
- Runtime type ID (RTTI)
- The preprocessor
- Much, much more

Herbert Schildt is the world's leading programming author. He is an authority on the C, C++, Java, and C# programming languages, and a master Windows programmer. He was a member of the ANSI/ISO committee that standardized C++. His programming books have sold more than three million copies worldwide and have been translated into all major foreign languages.



Professional

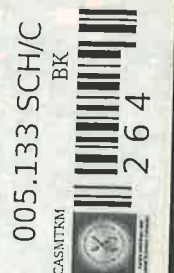
Schildt's classic
C++ reference—
updated and expanded

Schildt's classic
C++ reference—
updated and expanded

FREE
CODE
ONLINE

The
Complete
Reference

Fourth Edition



Schildt

Mc
Graw
Hill
Education

INDIAN
EDITION

The Complete Reference

Schildt's classic C++
reference—updated
and expanded

For sale in
India, Pakistan,
Nepal, Bangladesh
Sri Lanka and Bhutan
only

C++

Fourth Edition

Covers the International
Standard for C++,
including keywords,
syntax, and libraries

Includes advanced features such
as overloading, inheritance,
virtual functions, namespaces,
templates, the STL, and RTTI

Works with all
C++ compilers,
including Visual C++



Herbert Schildt

The world's leading programming author with more than 3 million books sold!

FREE
CODE
ONLINE

264

COLLEGE OF AVIAN SCIENCES & MANAGEMENT
KVASU CAMPUS, THIRUVAZHAMKUNNU

Acc. No. 264

Call No.



This book should be returned on or before the
date last given below.

--	--	--

++:
ence,
dition





McGraw Hill Education (India) Private Limited

C++: The Complete Reference, Fourth Edition

Copyright © 2003 by The McGraw-Hill Companies, Inc.

All rights reserved. No part of this publication may be reproduced or distributed in any form or by any means, or stored in a database or retrieval system, without the prior written permission of the publisher, with the exception that the program listings may be entered, stored, and executed in a computer system, but they may not be reproduced for publication.

McGraw Hill Education (India) Edition 2003

33rd reprint 2014
DDCQCRUDLABD

Reprinted in India by arrangement with The McGraw-Hill Companies, Inc., New York

Sales territories: India, Pakistan, Nepal, Bangladesh, Sri Lanka and Bhutan.

ISBN (13 digit): 978-0-07-053246-5

ISBN (10 digit): 0-07-053246-X

Published by McGraw Hill Education (India) Private Limited,
P-24, Green Park Extension, New Delhi 110 016, and printed at
Rajkamal Electric Press, Kundli 131 028, Haryana

Contents at a Glance

Part I The Foundation of C++: The C Subset

1	An Overview of C	3
2	Expressions	13
3	Statements	57
4	Arrays and Null-Terminated Strings	89
5	Pointers	113
6	Functions	137
7	Structures, Unions, Enumerations, and User-Defined Types	161
8	C-Style Console I/O	187
9	File I/O	211
10	The Preprocessor and Comments	237

Part II C++

11	An Overview of C++	255
12	Classes and Objects	289

13	Arrays, Pointers, References, and the Dynamic Allocation Operators	325
14	Function Overloading, Copy Constructors, and Default Arguments	359
15	Operator Overloading	383
16	Inheritance	417
17	Virtual Functions and Polymorphism	443
18	Templates	459
19	Exception Handling	487
20	The C++ I/O System Basics	509
21	C++ File I/O	539
22	Run-Time Type ID and the Casting Operators	567
23	Namespaces, Conversion Functions, and Other Advanced Topics	591
24	Introducing the Standard Template Library	629

Part III The Standard Function Library

25	The C-Based I/O Functions	699
26	The String and Character Functions	723
27	The Mathematical Functions	737
28	Time, Date, and Localization Functions	747
29	The Dynamic Allocation Functions	757
30	Utility Functions	761
31	The Wide-Character Functions	775

Part IV The Standard C++ Class Library

32	The Standard C++ I/O Classes	787
33	The STL Container Classes	811
34	The STL Algorithms	839
35	STL Iterators, Allocators, and Function Objects	861
36	The String Class	881
37	The Numeric Classes	897
38	Exception Handling and Miscellaneous Classes	925

Part V Applying C++

39	Integrating New Classes: A Custom String Class	935
40	Parsing Expressions	963
A	The .NET Managed Extensions to C++	999
B	C++ and the Robotics Age	1005
	Index	1009



Contents

Introduction	xxix
--------------------	------

Part I

The Foundation of C++: The C Subset

1 An Overview of C	3
The Origins and History of C	4
C Is a Middle-Level Language	5
C Is a Structured Language	6
C Is a Programmer's Language	8
The Form of a C Program	9
The Library and Linking	10
Separate Compilation	12
Understanding the .C and .CPP File Extensions	12
2 Expressions	13
The Five Basic Data Types	14
Modifying the Basic Types	15
Identifier Names	16
Variables	17
Where Variables Are Declared	18
Local Variables	18

Formal Parameters	21
Global Variables	21
The const and volatile Qualifiers	23
const	23
volatile	24
Storage Class Specifiers	25
extern	25
static Variables	28
register Variables	30
Variable Initializations	31
Constants	32
Hexadecimal and Octal Constants	33
String Constants	33
Backslash Character Constants	33
Operators	35
The Assignment Operator	35
Type Conversion in Assignments	35
Multiple Assignments	37
Arithmetic Operators	37
Increment and Decrement	38
Relational and Logical Operators	40
Bitwise Operators	42
The ? Operator	47
The & and * Pointer Operators	48
The Compile-Time Operator sizeof	50
The Comma Operator	50
The Dot (.) and Arrow (->) Operators	51
The [] and () Operators	51
Precedence Summary	52
Expressions	52
Order of Evaluation	52
Type Conversion in Expressions	53
Casts	54
Spacing and Parentheses	55
Compound Assignments	56
3 Statements	57
True and False in C and C++	58
Selection Statements	59
if	59
Nested ifs	60
The if-else-if Ladder	62
The ? Alternative	63
The Conditional Expression	66

switch	67
Nested switch Statements	70
Iteration Statements	70
The for Loop	70
for Loop Variations	72
The Infinite Loop	76
for Loops with No Bodies	77
The while Loop	77
The do-while Loop	79
Declaring Variables Within Selection and Iteration Statements	81
Jump Statements	82
The return Statement	82
The goto Statement	83
The break Statement	83
The exit() Function	85
The continue Statement	86
Expression Statements	88
Block Statements	88
4 Arrays and Null-Terminated Strings	89
Single-Dimension Arrays	90
Generating a Pointer to an Array	92
Passing Single-Dimension Arrays to Functions	92
Null-Terminated Strings	94
Two-Dimensional Arrays	96
Arrays of Strings	100
Multidimensional Arrays	101
Indexing Pointers	102
Array Initialization	105
Unsize Array Initializations	106
A Tic-Tac-Toe Example	108
5 Pointers	113
What Are Pointers?	114
Pointer Variables	115
The Pointer Operators	115
Pointer Expressions	116
Pointer Assignments	117
Pointer Arithmetic	117
Pointer Comparisons	119
Pointers and Arrays	121
Arrays of Pointers	122
Multiple Indirection	123
Initializing Pointers	125
Pointers to Functions	126

C's Dynamic Allocation Functions	129
Problems with Pointers	131
6 Functions	137
The General Form of a Function	138
Scope Rules of Functions	138
Function Arguments	139
Call by Value, Call by Reference	139
Creating a Call by Reference	140
Calling Functions with Arrays	142
argc and argv—Arguments to main()	144
The return Statement	147
Returning from a Function	148
Returning Values	149
Returning Pointers	151
Functions of Type void	152
What Does main() Return?	153
Recursion	153
Function Prototypes	155
Standard Library Function Prototypes	157
Declaring Variable-Length Parameter Lists	158
Old-Style Versus Modern FunctionParameter Declarations	158
7 Structures, Unions, Enumerations, and User-Defined Types	161
Structures	162
Accessing Structure Members	165
Structure Assignments	165
Arrays of Structures	166
Passing Structures to Functions	167
Passing Structure Members to Functions	167
Passing Entire Structures to Functions	168
Structure Pointers	169
Declaring a Structure Pointer	170
Using Structure Pointers	170
Arrays and Structures Within Structures	173
Bit-Fields	174
Unions	176
Enumerations	180
Using sizeof to Ensure Portability	183
typedef	184
8 C-Style Console I/O	187
An Important Application Note	188

Reading and Writing Characters	189
A Problem with getchar()	190
Alternatives to getchar()	190
Reading and Writing Strings	192
Formatted Console I/O	195
printf()	195
Printing Characters	196
Printing Numbers	196
Displaying an Address	198
The %n Specifier	198
Format Modifiers	199
The Minimum Field Width Specifier	199
The Precision Specifier	200
Justifying Output	201
Handling Other Data Types	202
The * and # Modifiers	202
scanf()	203
Format Specifiers	203
Inputting Numbers	203
Inputting Unsigned Integers	205
Reading Individual Characters Using scanf()	205
Reading Strings	205
Inputting an Address	206
The %n Specifier	206
Using a Scanset	206
Discarding Unwanted White Space	207
Non-White-Space Characters in the Control String	208
You Must Pass scanf() Addresses	208
Format Modifiers	208
Suppressing Input	209
9 File I/O	211
C Versus C++ File I/O	212
Streams and Files	212
Streams	212
Text Streams	213
Binary Streams	213
Files	213
File System Basics	214
The File Pointer	215
Opening a File	215
Closing a File	217
Writing a Character	217
Reading a Character	218

Using fopen(), getc(), putc(), and fclose()	218
Using feof()	220
Working with Strings: fputs() and fgets()	222
rewind()	223
ferror()	224
Erasing Files	226
Flushing a Stream	226
fread() and fwrite()	227
Using fread() and fwrite()	227
fseek() and Random-Access I/O	229
fprintf() and fscanf()	230
The Standard Streams	232
The Console I/O Connection	233
Using freopen() to Redirect the Standard Streams	234
10 The Preprocessor and Comments	237
The Preprocessor	238
#define	238
Defining Function-like Macros	240
#error	241
#include	242
Conditional Compilation Directives	242
#if, #else, #elif, and #endif	243
#ifdef and #ifndef	245
#undef	246
Using defined	247
#line	248
#pragma	248
The # and ## Preprocessor Operators	248
Predefined Macro Names	250
Comments	250
Single-Line Comments	252

Part II

C++

11 An Overview of C++	255
The Origins of C++	256
What Is Object-Oriented Programming?	257
Encapsulation	258
Polymorphism	258
Inheritance	259
Some C++ Fundamentals	259
A Sample C++ Program	260

A Closer Look at the I/O Operators	263
Declaring Local Variables	264
No Default to int	265
The bool Data Type	266
Old-Style vs. Modern C++	267
The New C++ Headers	268
Namespaces	269
Working with an Old Compiler	270
Introducing C++ Classes	270
Function Overloading	275
Operator Overloading	278
Inheritance	278
Constructors and Destructors	283
The C++ Keywords	287
The General Form of a C++ Program	288
12 Classes and Objects	289
Classes	290
Structures and Classes Are Related	293
Unions and Classes Are Related	295
Anonymous Unions	296
Friend Functions	297
Friend Classes	302
Inline Functions	303
Defining Inline Functions Within a Class	306
Parameterized Constructors	307
Constructors with One Parameter: A Special Case	309
Static Class Members	310
Static Data Members	310
Static Member Functions	315
When Constructors and Destructors Are Executed	317
The Scope Resolution Operator	319
Nested Classes	319
Local Classes	320
Passing Objects to Functions	320
Returning Objects	323
Object Assignment	324
13 Arrays, Pointers, References, and the Dynamic	325
Allocation Operators	325
Arrays of Objects	326
Creating Initialized vs. Uninitialized Arrays	328
Pointers to Objects	329
Type Checking C++ Pointers	332

The this Pointer	332
Pointers to Derived Types	334
Pointers to Class Members	337
References	339
Reference Parameters	339
Passing References to Objects	343
Returning References	344
Independent References	345
References to Derived Types	346
Restrictions to References	347
A Matter of Style	347
C++'s Dynamic Allocation Operators	347
Initializing Allocated Memory	349
Allocating Arrays	350
Allocating Objects	351
The nothrow Alternative	356
The Placement Form of new	357
14 Function Overloading, Copy Constructors, and Default Arguments	359
Function Overloading	360
Overloading Constructors	362
Overloading a Constructor to Gain Flexibility	362
Allowing Both Initialized and Uninitialized Objects	364
Copy Constructors	366
Finding the Address of an Overloaded Function	370
The overload Anachronism	371
Default Function Arguments	371
Default Arguments vs. Overloading	376
Using Default Arguments Correctly	377
Function Overloading and Ambiguity	378
15 Operator Overloading	383
Creating a Member Operator Function	384
Creating Prefix and Postfix Forms	389
of the Increment and Decrement Operators	390
Overloading the Shorthand Operators	390
Operator Overloading Restrictions	391
Operator Overloading Using a Friend Function	393
Using a Friend to Overload ++ or --	396
Friend Operator Functions Add Flexibility	398
Overloading new and delete	403
Overloading new and delete for Arrays	406
Overloading the nothrow Version of new and delete	

Overloading Some Special Operators	407
Overloading []	407
Overloading ()	411
Overloading ->	413
Overloading the Comma Operator	414
16 Inheritance	417
Base-Class Access Control	418
Inheritance and protected Members	420
Protected Base-Class Inheritance	424
Inheriting Multiple Base Classes	425
Constructors, Destructors, and Inheritance	426
When Constructors and Destructors Are Executed	426
Passing Parameters to Base-Class Constructors	430
Granting Access	434
Virtual Base Classes	437
17 Virtual Functions and Polymorphism	443
Virtual Functions	444
Calling a Virtual Function Through a Base	447
Class Reference	448
The Virtual Attribute Is Inherited	450
Virtual Functions Are Hierarchical	453
Pure Virtual Functions	455
Abstract Classes	455
Using Virtual Functions	458
Early vs. Late Binding	
18 Templates	459
Generic Functions	460
A Function with Two Generic Types	463
Explicitly Overloading a Generic Function	466
Overloading a Function Template	466
Using Standard Parameters with Template Functions	467
Generic Function Restrictions	468
Applying Generic Functions	469
A Generic Sort	470
Compacting an Array	472
Generic Classes	476
An Example with Two Generic Data Types	477
Applying Template Classes: A Generic Array Class	479
Using Non-Type Arguments with Generic Classes	481
Using Default Arguments with Template Classes	483
Explicit Class Specializations	

	The typename and export Keywords	484		More get() Functions	551
	The Power of Templates	485		getline()	551
19	Exception Handling	487		Detecting EOF	553
	Exception Handling Fundamentals	488		The ignore() Function	555
	Catching Class Types	494		peek() and putback()	556
	Using Multiple catch Statements	495		flush()	556
	Handling Derived-Class Exceptions	497		Random Access	557
	Exception Handling Options	498		Obtaining the Current File Position	561
	Catching All Exceptions	498		I/O Status	561
	Restricting Exceptions	500		Customized I/O and Files	563
	Rethrowing an Exception	502	22	Run-Time Type ID and the Casting Operators	567
	Understanding terminate() and unexpected()	503		Run-Time Type Identification (RTTI)	568
	Setting the Terminate and Unexpected Handlers	504		The Casting Operators	578
	The uncaught_exception() Function	505		dynamic_cast	578
	The exception and bad_exception Classes	506	23	Namespaces, Conversion Functions, and Other Advanced Topics	591
	Applying Exception Handling	506		Namespaces	592
20	The C++ I/O System Basics	509		Namespace Fundamentals	592
	Old vs. Modern C++ I/O	510		using	596
	C++ Streams	511		Unnamed Namespaces	598
	The C++ Stream Classes	511		Some Namespace Options	599
	C++'s Predefined Streams	512		The std Namespace	601
	Formatted I/O	513		Creating Conversion Functions	603
	Formatting Using the ios Members	513		const Member Functions and mutable	607
	Setting the Format Flags	514		Volatile Member Functions	609
	Clearing Format Flags	515		Explicit Constructors	610
	An Overloaded Form of setf()	516		The Member Initialization Syntax	611
	Examining the Formatting Flags	518		Using the asm Keyword	616
	Setting All Flags	519		Linkage Specification	617
	Using width(), precision(), and fill()	520		Array-Based I/O	618
	Using Manipulators to Format I/O	522		The Array-Based Classes	619
	Overloading << and >>	526		Creating an Array-Based Output Stream	619
	Creating Your Own Inserters	526		Using an Array as Input	621
	Creating Your Own Extractors	532		Input/Output Array-Based Streams	623
	Creating Your Own Manipulator Functions	535		Using Dynamic Arrays	624
				Using Binary I/O with Array-Based Streams	625
21	C++ File I/O	539		Summarizing the Differences Between C and C++	626
	<fstream> and the File Classes	540	24	Introducing the Standard Template Library	629
	Opening and Closing a File	540		An Overview of the STL	630
	Reading and Writing Text Files	543		Containers	630
	Unformatted and Binary I/O	545			
	Characters vs. Bytes	545			
	put() and get()	546			
	read() and write()	548			

Algorithms	631
Iterators	631
Other STL Elements	632
The Container Classes	633
General Theory of Operation	634
Vectors	635
Accessing a Vector Through an Iterator	639
Inserting and Deleting Elements in a Vector	641
Storing Class Objects in a Vector	643
Lists	645
Understanding end()	649
push_front() vs. push_back()	651
Sort a List	652
Merging One List with Another	653
Storing Class Objects in a List	655
Maps	658
Storing Class Objects in a Map	662
Algorithms	664
Counting	664
Removing and Replacing Elements	670
Reversing a Sequence	672
Transforming a Sequence	673
Using Function Objects	675
Unary and Binary Function Objects	675
Using the Built-in Function Objects	675
Creating a Function Object	678
Using Binders	680
The string Class	683
Some string Member Functions	687
Strings Are Containers	693
Putting Strings into Other Containers	694
Final Thoughts on the STL	695

Part III

The Standard Function Library

25 The C-Based I/O Functions	699
clearerr	700
fclose	701
feof	701
ferror	701
fflush	701
fgetc	702
fgetpos	702

fgets	703
fopen	703
fprintf	705
fputc	705
fputs	706
fread	706
freopen	706
fscanf	707
fseek	707
fsetpos	708
ftell	708
fwrite	709
getc	709
getchar	710
gets	710
perror	710
printf	711
putc	714
putchar	714
puts	714
remove	715
rename	715
rewind	715
scanf	719
setbuf	719
setvbuf	720
sprintf	720
sscanf	720
tmpfile	721
tmpnam	721
ungetc	721
vprintf, vfprintf, and vsprintf	722

26 The String and Character Functions	723
isalnum	724
isalpha	724
iscntrl	725
isdigit	725
isgraph	725
islower	725
isprint	726
ispunct	726
isspace	726
isupper	727

isxdigit	727
memchr	727
memcmp	727
memcpy	727
memmove	728
memset	728
strcat	729
strchr	729
strcmp	729
strcoll	730
strcpy	730
strcspn	731
strerror	731
strlen	731
strncat	731
strncmp	732
strncpy	732
strpbrk	733
strrchr	733
strspn	733
strstr	734
strtok	734
strxfrm	734
tolower	735
toupper	735

27 The Mathematical Functions

acos	737
asin	738
atan	738
atan2	739
ceil	739
cos	739
cosh	740
exp	740
fabs	740
floor	741
fmod	741
frexp	741
ldexp	741
log	742
log10	742
modf	742
pow	743
sin	743

sinh	744
sqrt	744
tan	744
tanh	745

28 Time, Date, and Localization Functions

asctime	748
clock	749
ctime	749
difftime	750
gmtime	750
localeconv	750
localtime	752
mktime	752
setlocale	752
strftime	753
time	754

29 The Dynamic Allocation Functions

calloc	758
free	758
malloc	759
realloc	759

30 Utility Functions

abort	762
abs	762
assert	763
atexit	763
atof	763
atoi	764
atol	764
bsearch	764
div	765
exit	766
getenv	766
labs	766
ldiv	767
longjmp	767
mblen	767
mbstowcs	768
mbtowc	768
qsort	768
raise	769

rand	770
setjmp	770
signal	770
srand	770
strtod	771
strtol	771
strtoul	772
system	772
va_arg, va_start, and va_end	773
wcstombs	773
wctomb	774

31	The Wide-Character Functions	775
	The Wide-Character Classification Functions	776
	The Wide-Character I/O Functions	779
	The Wide-Character String Functions	779
	Wide-Character String Conversion Functions	779
	Wide-Character Array Functions	782
	Multibyte/Wide-Character Conversion Functions	783

Part IV**The Standard C++ Class Library**

32	The Standard C++ I/O Classes	787
	The I/O Classes	788
	The I/O Headers	790
	The Format Flags and I/O Manipulators	791
	Several Data Types	793
	The streamsize and streamoff Types	793
	The streampos and wstreampos Types	793
	The pos_type and off_type Types	793
	The openmode Type	793
	The iostate Type	794
	The seekdir Type	794
	The failure Class	794
	Overload << and >> Operators	794
	The General-Purpose I/O Functions	795
	bad	795
	clear	795
	eof	795
	exceptions	795
	fail	796
	fill	796
	flags	796
	flags*	797

flush	797
fstream, ifstream, and ofstream	797
gcount	798
get	798
getline	799
good	800
ignore	800
open	800
peek	801
precision	802
put	802
putback	802
rdstate	802
read	803
readsome	803
seekg and seekp	804
setf	805
setstate	805
str	806
stringstream, istringstream, ostringstream	806
sync_with_stdio	807
tellg and tellp	808
unsetf	808
width	808
write	809

33	The STL Container Classes	811
	The Container Classes	812
	bitset	814
	deque	816
	list	819
	map	822
	multimap	824
	multiset	827
	queue	829
	priority_queue	830
	set	831
	stack	833
	vector	834

34	The STL Algorithms	839
	adjacent_find	840
	binary_search	840
	copy	841

copy_backward	841
count	841
count_if	842
equal	842
equal_range	842
fill and fill_n	842
find	843
find_end	843
find_first_of	843
find_if	843
for_each	844
generate and generate_n	844
includes	844
inplace_merge	845
iter_swap	845
lexicographical_compare	845
lower_bound	846
make_heap	846
max	846
max_element	847
merge	847
min	847
min_element	848
mismatch	848
next_permutation	848
nth_element	849
partial_sort	849
partial_sort_copy	849
partition	850
pop_heap	850
prev_permutation	850
push_heap	851
random_shuffle	851
remove, remove_if, remove_copy, and remove_copy_if	851
replace, replace_copy, replace_if, and replace_copy_if	852
reverse and reverse_copy	853
rotate and rotate_copy	853
search	853
search_n	854
set_difference	854
set_intersection	854
set_symmetric_difference	855
set_union	855
sort	856
sort_heap	856

stable_partition	857
stable_sort	857
swap	857
swap_ranges	858
transform	858
unique and unique_copy	858
upper_bound	859
35 STL Iterators, Allocators, and Function Objects	861
Iterators	862
The Basic Iterator Types	862
The Low-Level Iterator Classes	863
The Predefined Iterators	864
Two Iterator Functions	872
Function Objects	872
Function Objects	873
Binders	874
Negators	875
Adaptors	876
Allocators	879
36 The String Class	881
The basic_string Class	882
The char_traits Class	894
37 The Numeric Classes	897
The complex Class	898
The valarray Class	902
The slice and gslice Classes	917
The Helper Classes	920
The Numeric Algorithms	920
accumulate	920
adjacent_difference	921
inner_product	922
partial_sum	923
38 Exception Handling and Miscellaneous Classes	925
Exceptions	926
<exception>	926
<stdexcept>	927
auto_ptr	928
The pair Class	930
Localization	931
Other Classes of Interest	931

Part V

Applying C++

39	Integrating New Classes: A Custom String Class	935
	The StrType Class	936
	The Constructors and Destructors	938
	I/O on Strings	939
	The Assignment Functions	941
	Concatenation	942
	Substring Subtraction	945
	The Relational Operators	947
	Miscellaneous String Functions	948
	The Entire StrType Class	949
	Using the StrType Class	958
	Creating and Integrating New Types in General	961
	A Challenge	961
40	Parsing Expressions	963
	Expressions	964
	Parsing Expressions: The Problem	965
	Parsing an Expression	966
	The Parser Class	968
	Dissecting an Expression	969
	A Simple Expression Parser	971
	Understanding the Parser	977
	Adding Variables to the Parser	978
	Syntax Checking in a Recursive-Descent Parser	988
	Building a Generic Parser	989
	Some Things to Try	997
A	The .NET Managed Extensions to C++	999
	The .NET Keyword Extensions	1000
	Preprocessor Extensions	1002
	The attribute Attribute	1003
	Compiling Managed C++	1003
B	C++ and the Robotics Age	1005
	Index	1009

Introduction

If there is one language that defines modern programming, it is C++. Its syntax, style, and philosophy have set the standard by which all other languages are judged. Furthermore, C++ is the universal language of programming. When an algorithm or technique is described, it is usually done so using the C++ syntax. The long-term success of C++ has also left a lasting impression on computer language development. For example, both Java and C# are descended from C++. Frankly, to be a professional programmer implies proficiency in C++. It is the one language that no programmer can afford to ignore.

This is the fourth edition of *C++: The Complete Reference*. It fully describes and demonstrates the keywords, syntax, functions, classes, and features that define the C++ language. More specifically, this book fully describes Standard C++. This is the version of C++ defined by the ANSI/ISO Standard for C++ and it is the version of C++ that is supported by all major compilers, including Microsoft's Visual C++ and Borland's C++ Builder. Thus, the information in this book is applicable to all modern programming environments.

In the time that has passed since the previous edition of this book, there have been no changes to the C++ language. There have, however, been big changes to the computing environment. For example, a new standard for C, called C99, was created, Java became the dominant language for Web programming, the .NET Framework was

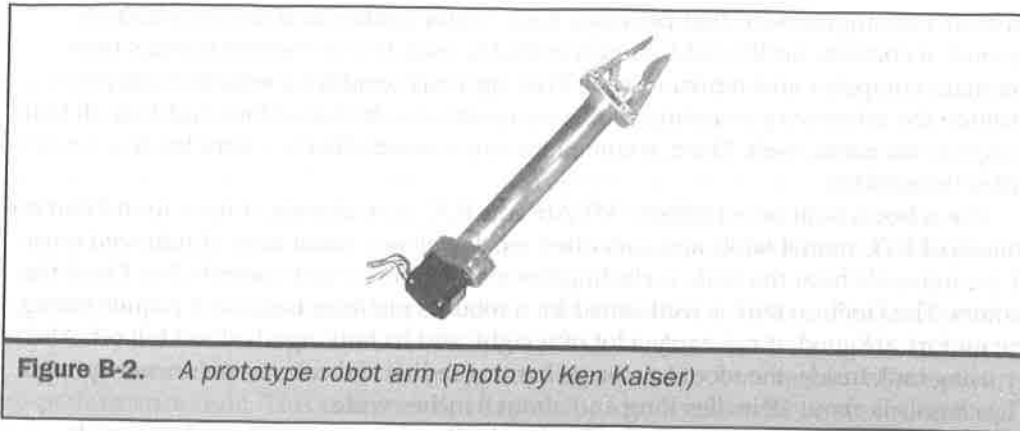


Figure B-2. A prototype robot arm. (Photo by Ken Kaiser)

Although the main robotic control code will always remain in C++, I am experimenting with migrating a couple of subsystems, including the RS-232 communication routines, to C#. C# offers a convenient interface to IP data transfers and being able to control the robot from a remote location via the Internet is a tantalizing thought.

Index

- & (bitwise operator), 43, 44
- & (pointer operator), 48, 49, 115-116, 141, 167, 170, 262, 347
- & (reference parameter), 340-341, 347
- &&, 40, 41, 42
- < >, 242, 268, 465, 483
- >, 51, 171, 176, 178, 329, 344
 - overloading, 407, 413-414
- >* (pointer-to-member operator), 337, 338, 339
- * (multiplication operator), 37, 38, 39
- * (pointer operator), 48-49, 115-116, 124, 347
- * (printf() placeholder), 202-203
 - l, 43, 44
 - ll, 40, 41, 42
- [], 51-52, 90, 145, 207, 350, 351, 356
 - overloading, 407-411
- ^, 43, 45, 207
- :, 47, 271, 613
- :: (scope resolution operator), 272, 319, 438-439
- , (comma operator), 50-51
 - overloading, 414-416
- { }, 8, 18, 58, 88
- . (dot operator), 51, 165, 176, 178, 272, 293; 344
- .* (pointer-to-member operator), 337, 338
- !, 40, 41, 42
- !=, 41, 42
- =, 31, 35
- =, 41, 42
- <, 41, 42
- << (left shift), 43, 45-46
- << (output operator), 262-264
 - overloading, 526-532, 794-795
- <=, 41, 42
- , 37, 38, 39
- -, 38-39, 389-390, 393-395
- () used for clarification in expressions, 55-56
- () function operator, 138
 - overloading, 407, 411-413
- () precedence operator, 40, 42, 51
- % (format specifier), 195
- % (modulus operator), 37, 38, 39
- +, 37, 38, 39
- ++, 38-39, 389-390, 393-395
- # (preprocessor directive), 238
- # (preprocessor operator), 248-249, 250
- # (printf() modifier), 202-203
- ## (preprocessor operator), 248, 249-250
- ?, 47-48, 63-66
- >, 41, 42
- >> (right shift), 43, 45-46
- >> (input operator), 262, 263-264
 - overloading, 526, 532-535, 794-795
- >=, 41, 42
- ;(semicolon), 88, 163
- /, 37, 38, 39
- /**/, 250-251
- //, 252, 262
- ~, 43, 46-48, 285

A

- abort(), 489, 490, 500, 503, 504, 762
- abs(), 762-763
- _abstract keyword, 1000
- Access declarations, 434-437
- Access specifiers, 290, 418-425
- accumulate() algorithm, 920-921
- acos(), 738
- Ada, 5
- Adaptor(s), 633, 876-878
- Address, memory
 - & operator used to return, 48, 115-116
 - pointer as, 48, 115
 - relocatable format, 12

adjacent_difference()
 algorithm, 921-922
 adjacent_find() algorithm, 840
 adjustfield format flag, 514
 advance(), 872
 Aggregate data type, 162
 ALGOL, 6, 8
 <algorithm> header, 664
 Algorithms, 631, 635, 664-674,
 840-859
 table of STL, 665-667
 Allocation, dynamic. *See*
 Dynamic allocation
 allocator class, 632, 879-880
 member functions,
 table of, 880
 Allocators, 632, 879-880
 AND
 & bitwise operator, 43, 44
 && logical operator, 40,
 41, 42
 ANSI/ISO
 C standard, 2, 4
 C++ standard, 4,
 256-257, 267
 app, 793
 append(), 688
 argc, 144-145, 147
 Arguments, function
 call by reference passing
 convention, 140-141, 170,
 339-343
 call by value passing
 convention, 139-140
 command line, 144-147
 default, 371-378, 380
 passing arrays as, 92-93, 98,
 102, 142-144
 passing functions as,
 126-129
 argv, 123, 144-147
 Arithmetic operators, 37-40
 precedence of, 39
 Array(s)
 allocating with new, 350-351
 bounds checking on, 6, 91,
 367, 410
 compacting, 470-472
 definition of, 90

generating pointer to, 92
 indexing versus pointer
 arithmetic, 121-122
 initialization, 105-107
 multidimensional, 101-102
 of objects, 326-329, 354,
 364-366
 to functions, passing, 92-93,
 142-144
 of pointers, 122-123
 pointers to access, 103-104,
 121-122
 safe, creating, 367-369,
 410-411
 single-dimension, 90-91
 sorting, 469-470
 square brackets as operator
 for indexing, 51-52
 of strings, 100-101
 of structures, 166
 within structures, 173
 two-dimensional, 96-101
 unsized, 106-107
 variable-length, 4
 vector as dynamic, 635
 Array-based I/O, 618-626
 and binary data, 625-626
 using dynamic arrays and,
 624-625
 using ios member functions
 with, 619
 Arrow operator (->), 51, 171,
 176, 178, 329, 344
 overloading, 407, 413-414
 asctime(), 748-749
 asin(), 738-739
 asm statement, 616-617
 Assembly language, 5, 8-9
 using asm to embed,
 616-617
 C used in place of, 8, 42-43
 assert(), 763
 assign(), 687-688
 Assignment
 compound, 56
 functions used in, 150,
 344-345
 multiple, 37
 object, 324

operation for C++ classes,
 default, 389
 operator, 35
 pointer, 117, 332
 shorthand, 56, 390
 structure, 165-166
 type conversion in, 35-37
 atan(), 739
 atan2(), 739
 ate, 793
 atexit(), 763
 atof(), 763-764
 atoi(), 146, 764
 atol(), 764
 attribute, 1003
 auto keyword, 18
 auto_ptr class, 928-930

B

B language, 4
 back_insert_iterator class,
 866, 867
 Backslash character constants,
 33-34
 bad(), 563, 795
 bad_alloc class, 348, 926
 bad_cast, 578, 927
 bad_exception class, 506, 926
 bad_typeid, 572, 926
 badbit, 561, 563, 794, 803
 Base class
 access control, 418-424
 constructors, passing
 parameters to, 430-434
 definition of, 278, 418
 general form for inheriting,
 279-280, 418
 inheritance, protected,
 424-425
 virtual, 437-441
 base(), 868
 basefield format flag, 514
 BASIC, 5, 6, 7, 8
 basic_filebuf class, 788, 789
 basic_fstream class, 512,
 788, 789
 basic_ifstream class, 512,
 788, 789
 basic_ios class, 511, 512, 788, 789

basic_iostream class, 511, 512,
 788, 789
 basic_istream class, 511, 512,
 788, 789
 basic_istreambuf class,
 788, 789
 basic_ofstream class, 512,
 788, 789
 basic_ostream class, 511, 512,
 788, 789
 basic_ostringstream class,
 788, 789
 basic_streambuf class, 511, 512,
 788, 789
 basic_string class, 683, 882-894
 constructors, 882
 member functions, table of,
 884-894
 basic_stringbuf class, 788, 789
 basic_stringstream class, 788,
 789
 BCPL language, 4
 before(), 568-569
 beg, 794
 begin(), 635, 636, 637, 641
 Biliter, 632, 812
 binary, 793
 binary_function class, 679,
 873, 874
 binary_negate class, 875-876
 binary_search() algorithm,
 840-841
 bind1st() binder, 680-681,
 874-875
 bind2nd() binder, 680-681,
 874-875
 Binders, 633, 680-682, 874-875
 BinPred type, 632, 812
 Bit shift operators (>> and <<),
 43, 45-46
 Bit-fields, 162, 174-176
 bitset container, 633, 634, 812,
 814-816
 member functions, table of,
 815-816
 <bitset> header, 633, 812
 Bitwise operation,
 definition of, 43
 Bitwise operators, 42-47

table of, 43
 Block statements, 58, 88
 bool data type, 14, 40, 58,
 59, 266
 _Bool data type, 59, 266
 boolalpha
 format flag, 514
 manipulator, 522,
 525-526, 791
 _box keyword, 1001
 break statement, 67, 68, 69-70,
 76, 83-85
 Broken-down time, 748
 bsearch(), 764-765
 BUFSIZ macro, 719

C

.C file extension, 12
 C Programming Language, *The*
 (Kernighan & Ritchie), 4
 C standard, ANSI/ISO, 2, 4
 C89, 4-5
 C99, 4-5
 C++
 differences between C and,
 130, 626-627
 differences between
 old-style and modern,
 267-270
 origins of, 256-257
 and robotics, 1006, 1008
 sample program, 260-263
 Standard, 4, 257, 267
 c_str(), 692
 Calendar time, 748
 Call by reference, 140-141
 automatic, using reference
 parameter, 339-343
 structure pointers
 used for, 170
 Call by value, 139-140
 calloc(), 758
 Case sensitivity, 10, 17, 287
 case statement, 67, 69-70
 <cassert> header, 763
 Casts, 54-55
 and base class pointers,
 334, 335
 C++ operators for, 55,
 578-589
 and pointers, 116, 130-131
 catch statement, 488-500, 502
 catch(...) form of, 498-500
 and derived-class
 exceptions, 497
 general form of, 488
 using multiple, 495-496
 <cctype> header, 724
 ceil(), 739-740
 cerr, 512
 <cerrno> header, 738
 char data type, 14, 15
 char_traits class, 789, 882,
 894-896
 member functions, table of,
 895-896
 Character(s)
 ASCII, 14, 547
 in C console I/O, 189-191
 constants, 32, 626
 constants, backslash, 33-34
 control, 724
 printable, 724
 set, extended, 547-548
 wide. *See* Wide character(s)
 Character translations
 in C I/O, 213, 217, 706
 in C++ file I/O, 545, 557
 cin, 262, 512
 Class(es)
 abstract, 455
 access specifications,
 290-292
 base. *See* Base class
 creating conversion
 functions for, 603-607
 declaration, general form
 for, 271-272, 290
 defining functions within,
 306-307, 309
 derived. *See* Derived class
 forward declaration of, 300
 friend, 302-303
 generic. *See* Generic class
 instance of, 271
 libraries, 458
 local, 320-321

nested, 319
 overview of, 270-274
 stream, 511-512
 structures and, 293-295
 unions and, 295-297
 class keyword, 270, 290, 295
 Class member(s)
 accessing, public, 272-273, 293
 definition of, 271, 292
 initialization syntax, 611-616
 pointers to, 337-339
 restrictions on, 292-293
 static, 310-317
 clear(), 795
 clearerr(), 700
 <climits> header, 14
 <locale> header, 748
 clock(), 749
 clock_t type, 748
 CLOCKS_PER_SEC, 748, 749
 clog, 512
 close(), 542
 <cmath> header, 738
 COBOL, 5, 7, 8
 Code
 block, 8, 18-20
 compartmentalization of, 6-7, 8
 managed (.NET Framework), compiling, 1000, 1003
 Comma operator, 50-51
 overloading, 414-416
 Command line arguments, 144-147
 Comments
 in C89, 250-251
 in C++ and C99, 4, 252, 262
 Common Language Runtime (CLR), 1000, 1003
 Comp type, 632, 812
 compare(), 692
 Compilation
 conditional, 242-246
 separate, 12, 27

Compilers
 compiling C programs with C++, 12
 working with older C++, 270
 complex class, 898-901
 functions defined for, table of, 900-901
 <complex> header, 898
 Compound data types, 162
 Compound statements, 58, 88
 Conditional expression, 58, 66-67
 Conglomerate data type, 162
 conio.h header file, 191
 const qualifier, 23-24, 25
 Constants, 32-34
 const_cast, 586-588
 Constructor(s), 283-287
 copy, 322-323, 324, 366-369
 and default arguments, 375-376
 execution order of, 317-318, 426-430
 explicit, 610-611
 and inheritance, 426-434
 and member initialization syntax, 613-616
 overloading, 362-369
 parameterized, 307-310
 passing parameters to base class, 430-434
 container protected variable, 867
 Containers, 630-631, 812-837
 defined by STL, table of, 633, 812-813
 string class and, 693-695
 typedef names for, 634, 813-814
 continue statement, 86-87
 copy() algorithm, 841
 copy_backward() algorithm, 841
 cos(), 740
 cosh(), 740
 count(), algorithm, 664, 665, 668-669, 841
 count_if() algorithm, 664, 665, 668, 669-670, 842
 cout, 262, 512

_cplusplus predefined macro, 250
 .CPP file extension, 12
 <csetjump> header, 767, 770
 <cstdarg> header, 722, 774
 <cstdio> header, 188, 214-215, 700, 719, 721
 <cstdlib> header, 59, 758, 762
 <csignal> header, 769, 771
 <cstring> header, 94, 724
 ctime(), 749
 <ctime> header, 748
 ctype.h header file, 724
 cur, 794
 current protected member, 868
 <wchar> header, 776, 779, 782, 783
 <wctype> header, 776

D

Data
 compartmentalization of, 67
 in expressions, 14
 Data type(s)
 aggregate, 162, 173
 ANSI/ISO C, table of, 15
 basic, 14-15
 class as, 271
 conversion of, in assignments, 35-37
 conversion of, in expressions, 6, 53-54
 creating and integrating new C++, 961
 definition of, 6
 modifiers for basic, 15-16
 DATE predefined macro, 250
 dec
 format flag, 514, 791
 manipulator, 522, 791
 Decrement operator (--), 38-39
 overloading for prefix and postfix, 389-390, 393-395
 default statement, 67
 #define directive, 238-241
 and function-like macros, 240-241

and preprocessor operators
 # and ##, 248-250
 defined compile-time operator, 247-248
 __delegate keyword, 1001
 delete dynamic allocation operator, 347-357, 758
 and arrays, 350-351
 and arrays of objects, 356
 overloading, 398-403, 406-407
 overloading for arrays, 403-406
 deque container, 630, 633, 634-635, 812, 816-818
 member functions, table of, 817-818
 <deque> header, 633, 812
 Derived class
 access declaration within, 434-436
 creating, 279
 definition of, 278, 418
 inheriting multiple base classes, 425-426
 objects, base class pointers to, 334-336
 Destructors, 284-287
 execution order of, 317-318, 426-430
 and inheritance, 426-430
 difference_type, 863
 difftime(), 750
 distance(), 872
 div(), 762, 765-766
 div_t type, 762, 765
 divides() function object, 677-678
 do-while loop, 7, 79-81
 domain_error exception, 928
 Dot operator (.), 51, 165, 176, 178, 272, 293, 344
 double data type, 14, 15
 Dynamic allocation, 129-131
 functions for, 130-131, 758-759
 operators for, 347-357, 758
 dynamic_cast, 578-586

E

Early binding, 458
 EDOM, 738
 #elif directive, 243, 244-245, 248
 else, 59
 #else directive, 243-245
 empty(), 646, 655
 Encapsulation, 258, 265
 class as basic unit of, 290
 and global variables, 315
 how to achieve, 271, 292
 end, 794
 end(), 635, 636, 637, 641, 649-651
 #endif, 243-245
 endl manipulator, 522, 791
 ends manipulator, 522, 791
 enum keyword, 180
 Enumerations, 162, 180-182
 EOF macro, 189, 215, 220, 700
 eof(), 553-555, 563, 625, 795-796
 eofbit, 794, 803
 equal()
 algorithm, 842
 member function, 870
 equal_range() algorithm, 842
 ERANGE, 738, 772, 773
 erase(), 634, 636, 637, 641, 688-689
 errno, 700, 738, 772, 773
 errno.h header file, 738
 Error checking, run-time, 6
 #error directive, 241
 Errors
 pointer, 131-135
 See also Exception handling
 Escape sequences, 33
 __event keyword, 1001
 exception class, 506, 794, 926-927
 Exception handling, 348, 488-507
 applying, 506-507
 and catching all exceptions, 498-500
 classes, 926-928
 fundamentals, 488-497
 and restricting exceptions, 500-502
 and rethrowing exceptions, 502-503
 <exception> header, 503, 504, 506, 926-927
 exceptions(), 796
 Exclusive OR. See XOR
 exit(), 85-86, 150, 490, 766
 EXIT_FAILURE, 263, 762, 766
 EXIT_SUCCESS, 263, 762, 766
 exp(), 740
 explicit specifier, 610-611
 export keyword, 485
 Expression(s), 52-56
 conditional, 58, 66-67
 definition of, 14, 52
 evaluation order, 52, 65
 function calls used in, 64-65, 149-150
 parser, 964-997
 pointer, 116-120
 production rules of, 966-967
 statements, 58, 88
 tokens, 969
 type conversion in, 53-54
 extern storage class specifier, 25-27, 618
 Extractors, creating, 526, 532-535

F

fabs(), 741
 facet class, 931
 fail(), 563, 796
 failbit, 561, 563, 794, 803
 failed(), 872
 failure class, 794
 false, 40, 58, 266
 fclose(), 150, 217, 218-220, 701
 feof(), 220-221, 701
 ferrord(), 224-225, 701-702
 fflush(), 226-227, 702
 fgetc(), 218, 702
 fgetpos(), 702-703
 fgets(), 192, 222, 232-233, 703
 File(s), C
 in C I/O system, 213-214
 closing, 217
 control structure, 213
 erasing, 226

opening, 215-217, 703-705
 pointer, 215, 216, 217
 File(s), C++
 closing, 542
 get pointer, 557, 561
 opening, 540-542
 put pointer, 557, 561
 reading and writing
 unformatted and binary, 545-553
 reading and writing text, 543-545
 FILE data type, 213, 215, 700
 File position indicator, 213, 702-703, 708
 resetting, 223-224, 715
 setting, 229-230
 FILE predefined macro, 248, 250
 filebuf class, 540, 790
 fill()
 algorithm, 843
 member function, 521-522, 796
 fill_n() algorithm, 843
 _finally keyword, 1001
 find()
 algorithm, 843
 member function, 635, 659, 662, 690-692
 find_end() algorithm, 843
 find_first_of() algorithm, 843-844
 find_if() algorithm, 844
 fixed
 format flag, 514, 791
 manipulator, 522, 791
 flags(), 518-520, 797
 float data type, 14, 15
 floatfield format flag, 514, 791
 Floating-point constants, 32-33
 floor(), 741
 flush manipulator, 522, 791
 flush(), 556-557, 797
 fmod(), 741
 fmtflags enumeration, 513, 791
 fopen(), 215-217, 218-220, 233, 703-705
 FOPEN_MAX macro, 215, 217
 for loop, 7, 70-77
 declaring variable within, 81
 general form of, 70-71
 infinite, 76
 variations of, 72-76
 with no body, 77
 for_each() algorithm, 844
 FortIter, 632, 812
 Formal parameters. *See* Parameters, formal
 Format flags, 513-520, 791
 Forth, 5
 FORTRAN, 5, 7, 257
 Forward declaration, 300
 fpos_t data type, 215, 700
 fprintf(), 230-232, 544, 705
 fputc(), 217, 705
 fputs(), 222, 705
 fread(), 227-228, 235, 706
 free(), 130-131, 348, 349, 758-759
 freeze(), 625
 freopen(), 234-235, 706-707
 frexp(), 741-742
 Friend classes, 302-303
 Friend functions, 297-302
 and the this pointer, 334, 391
 friend keyword, 298
 front_insert_iterator class, 866, 867
 fscanf(), 230-232, 544, 707
 fseek(), 215, 229-230, 707-708
 fsetpos(), 708
 fstream class, 512, 540, 790
 <fstream> header, 540, 790, 791
 fstream(), 797-798
 ftell(), 230, 708-709
 Function(s), 7-8, 10
 arguments. *See* Arguments, function
 conversion, creating, 603-607
 formal parameters of. *See* Parameters, formal
 friend. *See* Friend functions
 general form of, 138

generic. *See* Generic function(s)
 inline. *See* Inline functions
 library, 10-12, 697-698
 main(). *See* main()
 member. *See* Member functions
 objects. *See* Function objects
 passing multidimensional arrays to, 98, 102
 passing objects to, 320-323
 passing single-dimension arrays to, 92-93
 passing structures to, 167-169
 pointers to, 126-129
 prototypes, 155-157, 626
 predicate, 632
 recursive, 153-155
 return type, default to int, 266
 returning from, 148-149
 returning objects from, 323-324
 returning pointers from, 151-152
 returning references from, 344-345
 returning values from, 149-151
 scope rules of, 138-139
 stand-alone, 7-8, 28
 used in assignment statement, 344-345
 used in expressions, 64-65, 149-150
 virtual. *See* Virtual functions
 void, 150, 152-153
 Function objects, 632-633, 675-682, 872-878
 built-in, list of, 632, 675, 873
 using binders with, 680-682, 874-875
 creating, 678-680, 873
 Function overloading, 275-278, 360-371
 and ambiguity, 378-381
 and constructors, 362-369

and function pointers, 370-371
 versus default arguments, 376-377
 <functional> header, 632-633, 675, 872
 fwide(), 779
 fwrite(), 179, 227-228, 235, 709

G

Garbage collection, 1001
 _gc keyword, 1001
 gcount(), 550-551, 798
 generate() algorithm, 844-845
 generate_n() algorithm, 844-845
 Generated function, 462
 Generic class, 472-484
 creating safe array with, 477-480
 default arguments and, using, 481-483
 explicit specialization of, 483-484
 general form of, 473
 using non-type arguments in, 479-480
 typeid and, 576-578
 Generic function(s), 460-472
 applying, 468-472
 explicitly overloading, 463-465
 general forms of, 460, 462
 restrictions, 467-468
 get(), 546-547, 625, 798-799
 member function of auto_ptr, 929-930
 overloaded forms of, 551
 Get pointer, 557, 804, 808
 getch(), 218, 219-220, 709
 getch(), 190-191, 193
 getchar(), 189-191, 192, 193, 205, 710
 getche(), 190-191, 193
 getenv(), 766
 getline(), 551-553, 799-800
 gets(), 143, 144, 192, 193, 205, 222, 232, 710
 gmtime(), 750

good(), 563, 800
 goodbit, 561, 563, 794, 803
 goto statement, 7, 83, 138
 greater() function object, 680-682
 gslice class, 917, 919-920
 gslice_array class, 920

H

Headers and header files, 27, 157, 242, 261, 268-269, 603
 Heap, 130, 131, 758
 hex
 format flag, 514, 791
 manipulator, 522, 791
 Hexadecimal constants, 33
 Hierarchical classifications, 418
 Hoare, C.A.R., 769
 HUGE_VAL macro, 738, 772

I

_identifier keyword, 1001
 Identifiers, 16-17, 626
 #if directive, 243-244, 245, 247
 if statement
 declaring variable within, 81-82
 general form of, 59
 if-else-if ladder, 62-63
 nested, 60-62
 ? as alternative to, 63-66
 #ifdef directive, 245-246
 #ifndef directive, 245, 246
 ifstream class, 512, 540, 790
 ifstream(), 797-798
 ignore(), 555-556, 800
 imag(), 899
 in, 793
 #include directive, 242, 268, 269
 includes() algorithm, 845
 Increment operator (++), 38-39
 overloading for prefix and postfix, 389-390, 393-395
 indirect_array class, 920
 Inheritance, 259, 278-283, 418-441
 access declaration in, 434-437

access specifiers and, 418-425
 constructors, destructors, and, 426-434
 multiple base class, 425-426
 and virtual base classes, 437-441
 and virtual attribute of functions, 448-450
 InIter, 632, 812
 Inline functions, 303-307
 within a class, defining, 305-306
 inline keyword, 241, 303, 306
 and C99, 305
 inner_product() algorithm, 922-923
 inplace_merge() algorithm, 845
 Input operator (>>), 262, 263-264
 overloading, 526, 532-535, 794-795
 insert(), 634, 636, 637, 641, 643, 646, 648, 688, 689
 insert_iterator class, 864-866
 inserter(), 865
 Inserters, creating, 526-532
 int data type, 14, 15
 default to, 265-266, 627
 unsigned versus signed, 16
 Integers
 signed vs. unsigned, 16
 size of, 14
 _interface keyword, 1001
 internal
 format flag, 513, 791
 manipulator, 522, 792
 int_type data type, 555
 invalid_argument exception, 928
 I/O, C-style, 188
 functions, 700-722, 779, 780
 I/O, C console, 188-209
 basic functions for, table of, 193
 and characters, 189-191
 formatted, 195-209
 and strings, 191-194
 I/O, C file, 212-235
 common functions for, table of, 214

connection with console I/O, 233-234
 files in, 212, 213-214. *See also* File(s), C
 formatted, with `fprintf()` and `fscanf()`, 230-232
 random-access, 229-230
 reading and writing a character in, 217-218
 reading and writing blocks of data in, 227-228
 streams in, 212-214.
See also Streams and strings, 222
 I/O, C++
 array-based. *See* Array-based I/O
 byte-oriented vs. character-oriented, 545-546
 formatted, 513-526
 functions, 795-809
 headers, 790-791
 manipulators. *See* Manipulators
 old vs. modern, 510
 operators. *See* Operators, I/O
 predefined streams, 512-513
 streams, 511
 template classes, 511-512, 788-790
 I/O, C++ file, 540-566
 customized I/O and, 563-566
 flushing buffers in, 556-557
 random access, 557-561, 814
 status, obtaining, 561-563
See also Files, C++
 _IOFBF, 719
 _IOLBF, 719
 <iomanip> header, 523, 790, 791, 792
 _IONBF, 719
 ios class, 512, 540, 619, 789
 <ios> header, 790
 ios::app, 541, 801

ios::ate, 541, 801
 ios::badbit, 561
 ios::beg, 557, 804
 ios::binary, 541, 545, 801
 ios::cur, 557, 804
 ios::end, 557, 804
 ios::eofbit, 561
 ios::failbit, 561
 ios::goodbit, 561
 ios::in, 541, 801
 ios::out, 541, 801
 ios::trunc, 541, 801
 ios_base class, 511, 788, 789
 <iosfwd> header, 790, 793
 iostate enumeration, 561, 794
 istream class, 512, 619, 789
 <istream> header, 260, 278, 510, 512, 790, 791
 istream.h header file, 510
 is_open(), 542
 isalnum(), 724
 isalpha(), 724
 iscntrl(), 725
 isdigit(), 725
 isgraph(), 725
 islower(), 725-726
 isprint(), 726
 ispunct(), 726
 isspace(), 726
 istream class, 512, 532, 619, 789
 <istream> header, 790
 istream_iterator class, 868-869
 istream_type type, 869, 870
 istreambuf_iterator class, 869
 istream class, 790
 istreambuf_iterator class, 806-807
 istrstream class, 619, 621, 622
 isupper(), 727
 iswctype(), 776-778
 isxdigit(), 727
 Iteration statements, 58, 70-81
 declaring variables within, 81
 Iterator(s), 631, 635, 639-641, 812, 862-872
 functions, 872
 predefined, 864-872

iterator
 class, 863
 type, 631, 641, 863
 <iterator> header, 862, 863
 iterator_category type, 863
 iterator_traits class, 864
 iter_swap() algorithm, 845

J

Java, 5, 7
 jmp_buf type, 767
 Jump statements, 58, 68, 82-87

K

kbhit(), 84
 Kernighan, Brian, 4
 Keywords
 C, 6, 8
 C89, table of, 10
 C++, table of, 287-288
 extended, common, 10
 .NET Framework extensions to C++, 1000-1002

L

L_tmpnam, 721
 Label
 identifier for goto statement, 83
 statements, 58, 67
 labs(), 766
 Language(s), computer
 block-structured, 6, 139
 high-level, 5-6
 middle-level, C as, 5-6
 programmer's, C as, 8-9
 structured, C as, 6-8, 257-258
 Late binding, 458
 LC_ALL, 753
 LC_COLLATE, 753
 LC_CTYPE, 753
 LC_MONETARY, 753
 LC_NUMERIC, 753
 LC_TIME, 753

lconv structure, 750-751
 ldexp(), 742
 ldiv(), 762, 767
 ldiv_t structure, 762, 765-766, 767
 left
 format flag, 513, 791
 manipulator, 522, 792
 length_error exception, 928
 less() function object, 632, 633, 658
 lexicographical_compare() algorithm, 846
 Library
 class, 11, 458, 785-786, 898
 standard function, 10-12, 697-698
 standard template. *See* Standard template library (STL)
 <limits> header, 931
 #line directive, 248
 __LINE__ predefined macro, 248, 250
 Line-buffered input, 190
 Linkage
 categories of, 25
 specification, 617-618
 Linker, 12, 27
 list container, 630, 633, 634-635, 645-657, 812, 819-822
 member functions, table of, 646-647, 819-822
 <list> header, 633
 Literals, 32
 locale class, 931
 <locale> header, 931
 locale.h header file, 748
 localeconv(), 750-752
 Localization
 C functions for, 748, 750-751, 752-754
 class library, 931
 localtime(), 752
 log(), 742
 log10(), 742-743
 logic_error class, 928

Logical operators, 40-42
 truth table for, 40
 long modifier, 15-16
 LONG_MAX, 772
 LONG_MIN, 772
 longjmp(), 507, 767, 770
 Loops
 do-while, 7, 79-81
 for. *See* for loop
 infinite, 76
 and structured languages, 7
 time delay, 77
 while, 7, 77-79
 with no bodies, 77
 lower_bound() algorithm, 846
 lvalue, 35

M

Macro
 function-like, 240-241
 name, 238
 predefined, 250
 replacement, 238
 main(), 10, 156, 627
 argc and argv as arguments to, 144-147
 return value from, 153
 using void in parameter list of, 147
 make_heap() algorithm, 846-847
 make_pair(), 660-661, 662, 930-931
 malloc(), 130-131, 348, 349, 356, 758, 759
 managed pragma, 1002
 Manipulators
 creating custom, 535-538
 to format I/O, using, 522-526
 table of C++, 522-523, 791-792
 map container, 631, 633, 634, 658-664, 813, 822-824
 member functions, table of, 659-660, 823-824
 <map> header, 633, 813
 mask_array class, 920
 math.h header file, 738
 max() algorithm, 847
 max_element() algorithm, 847
 MB_CUR_MAX, 762, 774
 mblen(), 767-768
 mbstate_t type, 776, 783
 mbstowcs(), 768
 mbtowc(), 768
 mem_fun() adaptor, 877
 mem_fun_ref() adaptor, 878
 mem_fun_ref_t class, 878
 mem_fun_t class, 877-878
 mem_fun1() adaptor, 877
 mem_fun1_ref() adaptor, 878
 mem_fun1_ref_t class, 878
 mem_fun1_t class, 877-878
 Member functions, 271, 272, 292
 const, 607-609
 and scope resolution operator, 272
 static, 315-317
 and the this pointer, 332-334
 volatile, 609
 within class, defining, 306-307, 309
 Member variables, 271, 272, 292, 293
 initialization syntax for, 611-616
 static, 310-315
 memchr(), 727
 memcmp(), 727-728
 memcpy(), 728
 memmove(), 728
 <memory> header, 879, 928, 931
 memset(), 729
 merge(), 647, 648, 653-655
 algorithm, 847-848
 Metadata, 1002
 Microsoft Intermediate Language (MSIL), 1000
 min() algorithm, 848
 min_element() algorithm, 848
 mismatch() algorithm, 848-849
 mktime(), 752
 modf(), 743
 Modula-2, 5, 6, 7, 8
 <mscorlib.dll>, 1002

_property keyword, 1002
 protected access specifier, 290,
 418, 420
 effects of, 420-425
 Prototypes, function,
 155-157, 626
 ptr_fun() adaptor, 876-877
 ptrdiff_t type, 664, 842, 863
 public access specifier, 271,
 290, 418
 effects of, 418-419
 push_back(), 634, 636, 637, 646,
 647, 649, 651-652, 867
 push_front(), 635, 647, 648,
 651-652, 867
 push_heap() algorithm, 851
 Put pointer, 557, 804, 808
 put(), 546, 547-548, 626, 802
 putback(), 556, 802
 putc(), 217-219, 714
 putchar(), 189-190, 193, 232, 714
 puts(), 192-194, 714

Q

qsort(), 768-769
 queue container, 633, 813,
 829-830
 member functions, table of,
 829-830
 <queue> header, 633, 813
 Quicksort, 155, 769

R

raise(), 769
 rand(), 59, 574, 770
 RAND_MAX, 59, 762, 770
 RandIter, 632, 812
 Random-access I/O, 229-230,
 557-561, 804
 random_shuffle() algorithm,
 851-642
 range_error exception, 928
 raw_storage_iterator class, 931
 rdstate(), 561-563, 802-803
 read(), 548-551, 626, 803
 readsome(), 803-804

real(), 899
 realloc(), 758, 759
 Recursion, 153-155
 Reference(s)
 definition of, 339
 independent, 345-346
 to objects, passing, 343-344
 parameters, 339-343, 346,
 393-394
 restrictions, 347
 returning, 344-345
 typeid and, 572-573
 and virtual functions,
 447-448
 reference type, 863
 register storage class specifier,
 30-31, 627
 reinterpret_cast, 588-589
 Relational operators, 40-42
 release(), 929
 Relocatable format, 12
 remove(), 226, 715
 algorithm, 852
 remove_copy() algorithm, 666,
 670-672, 852
 remove_copy_if() algorithm,
 852
 remove_if() algorithm, 666,
 681-682, 852
 rename(), 715
 replace(), 688, 689
 algorithm, 852-853
 replace_copy() algorithm, 666,
 670-672, 852-853
 replace_copy_if() algorithm,
 852-853
 replace_if() algorithm, 852-853
 resetiosflags manipulator,
 523, 792
 restrict qualifier, 4
 return statement, 82
 using to return from
 a function, 148-149
 using to return a value, 149
 reverse() algorithm, 667,
 672-673, 853
 reverse_copy() algorithm, 853
 reverse_iterator class, 868

rewind(), 223-224, 715
 rfind(), 690-692
 Richards, Martin, 4
 right
 format flag, 513, 791
 manipulator, 523, 792
 Ritchie, Dennis, 4
 Robotics, 1006-1008
 rotate() algorithm, 853
 rotate_copy() algorithm,
 853-854
 Run-time type identification
 (RTTI), 568-578
 runtime_error class, 928
 rvalue, 35

S

Scalar, 59
 scanf(), 195, 203-209, 262,
 715-719
 format specifiers, table of,
 204, 716
 Scanset, 206-207, 718-719
 scientific
 format flag, 514, 791
 manipulator, 523, 792
 Scope resolution operator (::),
 272, 319, 438-439
 Scope rules, 138
 __sealed keyword, 1002
 search() algorithm, 854
 search_n() algorithm, 854
 SEEK_CUR macro, 215, 229, 708
 SEEK_END macro, 215, 229, 708
 SEEK_SET macro, 215, 229,
 230, 708
 seekdir enumeration, 557,
 794, 804
 seekg(), 557-560, 804
 seekp(), 557-560, 804
 Selection statements, 58, 59-70
 declaring variables within,
 81-82
 set container, 633, 634, 813,
 831-833
 member functions, table of,
 832-833

<set> header, 633, 813
 set_difference() algorithm,
 854-855
 set_intersection() algorithm, 855
 set_symmetric_difference()
 algorithm, 855-856
 set_terminate(), 504
 set_unexpected(), 504
 set_union() algorithm, 856
 setbase manipulator, 523, 792
 setbuf(), 719
 setf(), 514-515, 805
 overloaded, 516-518
 setfill manipulator, 523, 792
 setiosflags() manipulator, 523,
 525, 792
 setjmp(), 507, 767, 770
 setlocale(), 752-753
 setprecision manipulator,
 523, 792
 setvbuf(), 719-720
 setw manipulator, 523, 792
 short modifier, 15-16
 Shorthand assignment
 operators, 56, 390
 showbase
 format flag, 514, 791
 manipulator, 523, 792
 showpoint
 format flag, 514, 791
 manipulator, 523, 792
 showpos
 format flag, 514, 791
 manipulator, 523, 792
 SIG_DFL, 771
 SIG_ERR, 771
 SIG_IGN, 771
 SIGABRT, 769
 SIGFPE, 769
 SIGILL, 769
 SIGINT, 770
 Sign flag, 16
 signal(), 770-771
 signed modifier, 15-16
 SIGSEGV, 770
 SIGTERM, 770
 sin(), 150, 743
 sinh(), 744
 size(), 636, 637
 sizeof operator, 50, 131, 183-184

size_t data type, 50, 130, 215,
 398, 700, 724, 762, 776
 skipws
 format flag, 513, 791
 manipulator, 523, 792
 slice class, 917-919
 slice_array class, 920
 sort(), 647, 652-653
 algorithm, 856
 sort_heap() algorithm, 856-857
 splice(), 647, 648
 sprintf(), 720
 sqrt(), 150, 744
 sscanf(), 720
 sqrt(), 150, 744
 srand(), 771
 <sstream> header, 790
 stable_partition() algorithm, 857
 stable_sort() algorithm, 857
 Stack
 and local variables, 20
 and recursive routines, 154
 stack container, 633, 813,
 833-834
 member functions,
 table of, 834
 <stack> header, 633, 813
 Standard C++, 4, 257, 267
 Standard template library
 (STL), 11, 256-257, 630-695
 elements of, 630-634
 general theory of operation,
 634-635
 Statements, 57-88
 static storage class specifier,
 28-30, 310
 static_cast, 588
 std namespace, 261, 269-270,
 510, 592, 601-603
 stdarg.h header file, 722
 <stdbool.h> header, 266
 __STDC__ predefined
 macro, 250
 stderr standard stream, 232-233
 <stdexcept> header, 926, 927-928
 stdin standard stream, 232-234
 stdio.h header file, 188,
 214-215, 700
 stdlib.h header file, 59, 759, 762
 stdout standard stream, 232-234

Stepanov, Alexander, 256
 Storage class specifiers, 25-31
 str(), 624, 806
 strcat(), 94-95, 685, 729
 strchr(), 94-95, 729
 strcmp(), 74, 94-95, 128, 730
 strcoll(), 730
 strcpy(), 94-95, 684, 685, 731
 strcspn(), 731
 Stream(s)
 binary, 212, 213
 C++, 510
 for C++ array-based I/O,
 619-623
 for C++ file I/O, 540
 classes, 510-512, 540,
 618-619
 flushing, 226-227
 predefined C++, 512-513
 redirecting standard, 232,
 233-235, 512
 standard C, 232-233
 text, 212-213
 streambuf class, 512, 790
 <streambuf> header, 791
 streambuf_type, 872
 streamoff data type, 793
 streampos data type, 793
 streamsize data type, 548, 793
 strerror(), 731
 strptime(), 753-754
 Stride, 918
 String(s)
 as arrays, 90, 94-95
 arrays of, 100-101
 class, creation of custom,
 936-961
 classes, Standard C++, 94,
 630, 683-695, 882-896
 in console I/O, 192-194
 constant, 33, 94, 126
 in C file I/O, 222
 limitations of
 null-terminated, 683-684
 manipulation functions,
 94-95
 substring subtraction from,
 936, 945-946
 table, 126

string class, 94, 630, 683-695, 882, 884
 and containers, 693-695
 dynamic aspect of, 687
 member functions, 687-692
 operators defined for, 685
 <string> header, 684
 string.h header file, 724
 stringbuf class, 790
 stringstream class, 790
 stringstream(), 806-807
 strlen(), 24, 79, 94-95, 731-732
 strncat(), 732
 strncmp(), 732
 strncpy(), 733
 Stroustrup, Bjarne, 256, 267
 strpbrk(), 733
 strchr(), 733
 strspn(), 734
 strstr(), 94-95, 734
 stringstream class, 619, 623, 625
 <sstream> header, 618
 strtod(), 771-772
 strtok(), 734
 strtol(), 772
 strtoul(), 772-773
 struct keyword, 162, 163, 295
 Structure(s), 162-174
 arrays of, 166
 arrays and structures
 within, 173-174
 assignments, 165-166
 and classes, 293-295
 declaration, 162-163,
 164-165
 members, accessing, 165,
 171, 173
 passing to functions,
 167-169
 pointers, 169-173
 size of, determining, 183
 variable, declaring, 163-164
 strxfrm(), 735
 swap() algorithm, 857
 swap_ranges() algorithm, 858
 switch statement, 67-70
 declaring variable within, 81

sync_with_stdio(), 807
 system(), 773
 Systems program, 9

T

tan(), 744
 tanh(), 744
 tellg(), 561, 808
 tellp(), 561, 808
 Template function. *See* Generic
 function(s)
 template keyword, 460
 template<> syntax, 465, 483
 Templates, 460-485
 advantages to using, 485
 definition of, 460
 See also Generic class;
 Generic function(s)
 terminate(), 489, 503-505
 terminate_handler type, 504, 927
 Ternary operator (?), 47-48, 63-66
 this pointer, 315, 332-334,
 386, 391
 Thompson, Ken, 4
 throw statement, 489-490,
 502-503
 throw() clause, 500
 Time and date functions,
 748-755
 Time delay loops, 77
 time.h header file, 748
 time(), 754-755
 TIME predefined
 macro, 250
 time_t data type, 748
 tm structure, 748
 TMP_MAX, 721
 tmpfile(), 720-721
 tmpnam(), 721
 Token, 969
 tolower(), 735
 toupper(), 735
 towctrans(), 778-779
 transform() algorithm, 667,
 673-674, 676-678, 858
 true, 40, 58, 266

True and false in C and C++,
 40, 58
 trunc, 793
 try statement, 488-493, 501-502
 _try_cast keyword, 1002
 Two's complement, 16
 Type checking and C++
 pointers, 549
 Type conversion
 and ambiguity in function
 overloading, 378-380
 in assignments, 35-37
 in expressions, 53-54
 using unions for
 nonstandard, 178-179
 Type promotion, 53
 type_info class, 568, 931
 typedef statement, 162, 184-185
 typeid, 568-578, 926-927
 using dynamic_cast to
 replace, 582-584
 <typeinfo> header, 568, 931
 typename keyword, 460,
 484-485
 _typeof keyword, 1002
 Types. *See* Data types

U

ULONG_MAX, 773
 unary_function class, 678,
 873-874
 unary_negate class, 875, 876
 uncaught_exception(), 505
 #undef directive, 246-247
 underflow_error exception, 928
 unexpected(), 500, 501,
 503-504, 926
 unexpected_handler type,
 504, 927
 ungetc(), 721-722
 union keyword, 177
 Unions, 162, 176-179, 184
 anonymous, 179, 296-297
 and classes, 295-297
 for nonstandard type
 conversions, 178-179
 size of, determining, 184

unique() algorithm, 858-859
 unique_copy() algorithm,
 858-859
 unitbuf
 format flag, 514, 791
 manipulator, 523, 792
 UNIX, 4
 unmanaged pragma, 1002
 UnPred type, 632, 812
 unsetf(), 515-516, 808
 unsigned modifier, 14-16
 upper_bound() algorithm, 859
 uppercase
 format flag, 514, 791
 manipulator, 523, 792
 #using preprocessor
 directive, 1002
 using statement, 261, 270, 434,
 437, 596-598
 <utility> header, 632, 931

V

va_arg(), 773-774
 va_end(), 773-774
 va_list type, 722, 774
 va_start(), 773-774
 valarray class, 902-920
 member functions, table of,
 903-907
 nonmember operator
 functions defined for,
 table of, 908-913
 transcendental functions
 defined for, table of,
 914-915
 <valarray> header, 902
 _value keyword, 1002
 Variables, 17-23
 automatic, 18
 declaration vs.
 definition of, 26
 declaring, 17-18, 81-82
 as formal parameters, 21
 initializing, 31-32
 instance, 292

member. *See* Member
 variables
 placement in memory, 133
 pointer, 48-49, 115
 reference. *See* Reference(s)
 register, 30-31
 storage class specifiers for,
 25-31
 structure, 163-164
 Variables, global, 7, 21-23
 declarations, difference
 between C and C++, 626
 and encapsulation, 315
 extern used with, 25-27
 static, 29-30, 598-599
 Variables, local, 6-7, 18-20,
 21, 22
 declarations, differences
 between C and C++,
 19-20, 264-265, 626
 initializing, 20
 static used with, 20, 28-29,
 30, 139
 vector container, 630, 633, 634,
 635-645, 813, 834-837
 member functions, table of,
 637, 835-837
 <vector> header, 633, 813
 vfprintf(), 722
 Virtual functions, 444-458
 and base-class references,
 447-448
 and class libraries, 458
 hierarchical nature of,
 450-453
 and inheritance of virtual
 attribute, 448-450
 and late binding, 458
 overloading versus
 overriding and, 446-447
 pure, 453-455
 using, 455-458
 virtual keyword, 440, 444, 445
 Visual C++ compiler, 191
 void data type, 14, 15, 153,
 261, 626

W

volatile access modifier,
 24-25, 609
 vprintf(), 722
 vsprintf(), 722
 wchar.h header file, 776
 WCHAR_MAX, 776
 WCHAR_MIN, 776
 wchar_t data type, 14, 32, 513,
 698, 776
 wcstombs(), 774
 wctomb(), 774
 wctrans(), 778-779
 wctrans_t type, 776
 wctype.h header file, 776
 wctype(), 776-778
 wctype_t type, 776
 WEOF macro, 776
 werr, 513
 what(), 794, 926
 while loop, 7, 77-79
 declaring variable within, 81
 Wide character(s), 511, 513, 545
 functions, 776-784
 I/O classes, 512, 789-790
 width(), 520, 521-522, 808-809
 win, 513
 wint_t type, 776
 wlog, 513
 Word alignment, 183
 wout, 513
 write(), 548-551, 626, 809
 ws manipulator, 523, 792
 wstreampos type, 793
 wstring class, 683, 862, 864

X

XOR
 bitwise operator (^), 43, 45
 logical operation, 40-42