Appendix A

C++ Keywords

The following lists the keywords and reserved words of the C++ language:

asm	To declare that a block of code is to be passed to the assembler
auto	A storage class specifier that is used to define objects in a block
bool	Boolean false–true type that can hold either the **false** or **true** literals
break	Terminates a **switch** statement or a loop
case	Used specifically within a **switch** statement to specify a match for the statement's expression
catch	Specifies actions taken when an exception occurs
char	Fundamental data type that defines character objects
class	To declare a user-defined type that encapsulates data members and operations or member functions
const	To define objects whose value will not alter throughout the lifetime of program execution
continue	Transfers control to the start of a loop
default	Handles expression values in a **switch** statement that are not handled by **case**
delete	Memory deallocation operator
do	Indicates the start of a **do-while** statement in which the sub-statement is executed repeatedly until the value of the expression is logical-false
double	Fundamental data type used to define a floating-point number
else	Used specifically in **if-else** statement
enum	To declare a user-defined enumeration data type
explicit	To declare an explicit constructor
export	Allows a template definition to be *accessible from another translation unit.*
extern	An identifier specified as **extern** has external linkage to the block
false	Boolean literal of value zero
float	Fundamental data type used to define a floating-point number
for	Indicates the start of a **for** statement to achieve repetitive control
friend	A **class** or operation whose implementation can access the **private** data members of a **class**
goto	Transfer control to a specified label
if	Indicate start of an **if** statement to achieve selective control
inline	A function specifier that indicates to the compiler that inline substitution of the function body is to be preferred to the usual function call implementation

int	Fundamental data type used to define integer objects
long	A data type modifier that defines a 32-bit **int** or an extended **double**
mutable	Allows an object member to override **const**ness
namespace	Defines a scope
new	Memory allocation operator
operator	Overloads a C++ operator with a new declaration
private	Declares **class** members which are not visible outside the **class**
protected	Declares **class** members which are **private** except to derived classes
public	Declares **class** members which are visible outside the **class**
register	A storage class specifier that is an **auto** specifier, but which also indicates to the compiler that an object will be frequently used and should therefore be kept in a register
return	Returns an object to a function's caller
short	A data type modifier that defines a 16-bit **int** number
signed	A data type modifier that indicates an object's sign is to be stored in the high-order bit
sizeof	Returns the size of an object in bytes
static	The lifetime of an object defined **static** exists throughout the lifetime of program execution
struct	To declare new types that encapsulate both data and member functions
switch	Switch statement
template	Parametrised or generic type
this	A **class** pointer which points to an object or instance of the **class**
throw	Generate an exception
true	Boolean literal of value one
try	Indicates start of a block of exception handlers
typedef	Synonym for another integral or user-defined type
typeid	The **typeid** *( )* operator returns the type of its operand
typename	Within a **template** **typename** indicates that a qualified name denotes a type
union	Similar to a structure, **struct**, in that it can hold different types of data, but a union can hold only one of its members at a given time.
unsigned	A data type modifier that indicates the high-order bit is to be used for an object
using	**using** declaration and **using** directive
virtual	A function specifier that declares a member function of a **class** which will be redefined by a derived **class**
void	Absent of a type or function parameter list
volatile	Define an object which may vary in value in a way that is undetectable to the compiler
wchar_t	Wide character type
while	Start of a **while** statement and end of a **do-while** statement

Appendix **B**

ASCII Character Set

It appears customary to include the ASCII character set in an Appendix of a book on computer programming, so here it is[1]:

Decimal	*Hexadecimal*	*Octal*	*Binary*	*Key*	*ASCII character*
0	00	0	00000000	CTRL+2	null
1	01	1	00000001	CTRL+A	☺
2	02	2	00000010	CTRL+B	☻
3	03	3	00000011	CTRL+C	♡
4	04	4	00000100	CTRL+D	♢
5	05	5	00000101	CTRL+E	♧
6	06	6	00000110	CTRL+F	♤
7	07	7	00000111	beep	•
8	08	10	00001000	backspace	◘
9	09	11	00001001	tab	
10	0a	12	00001010	newline	■
11	0b	13	00001011	CTRL+K	♂
12	0c	14	00001100	CTRL+L	♀
13	0d	15	00001101	enter	♪
14	0e	16	00001110	CTRL+N	♫
15	0f	17	00001111	CTRL+O	¤
16	10	20	00010000	CTRL+P	.
17	11	21	00010001	CTRL+Q	,
18	12	22	00010010	CTRL+R	↕
19	13	23	00010011	CTRL+S	!!
20	14	24	00010100	CTRL+T	¶
21	15	25	00010101	CTRL+U	§
22	16	26	00010110	CTRL+V	■
23	17	27	00010111	CTRL+W	↨
24	18	30	00011000	CTRL+X	↑
25	19	31	00011001	CTRL+Y	↓
26	1a	32	00011010	CTRL+Z	→
27	1b	33	00011011	esc	←
28	1c	34	00011100	CTRL+\	@
29	1d	35	00011101	CTRL+]	↔

1 Note that characters 0–31 (inclusive) and 127 are control characters, 32–126 represent keys on the keyboard and 127–255 are the IBM extended ASCII character set, which can be displayed by pressing the ALT key and typing the decimal code of the character on the numeric keypad. The IBM extended character set is not part of the ASCII character set and as a result may not be fully portable.

Decimal	*Hexadecimal*	*Octal*	*Binary*	*Key*	*ASCII character*
30	1e	36	00011110	CTRL+6	▲
31	1f	37	00011111	CTRL+-	▼
32	20	40	00100000	spacebar	
33	21	41	00100001	!	!
34	22	42	00100010	"	"
35	23	43	00100011	#	#
36	24	44	00100100	$	$
37	25	45	00100101	%	%
38	26	46	00100110	&	&
39	27	47	00100111	'	'
40	28	50	00101000	(	(
41	29	51	00101001	)	)
42	2a	52	00101010	*	*
43	2b	53	00101011	+	+
44	2c	54	00101100	,	,
45	2d	55	00101101	-	-
46	2e	56	00101110	.	.
47	2f	57	00101111	/	/
48	30	60	00110000	0	0
49	31	61	00110001	1	1
50	32	62	00110010	2	2
51	33	63	00110011	3	3
52	34	64	00110100	4	4
53	35	65	00110101	5	5
54	36	66	00110110	6	6
55	37	67	00110111	7	7
56	38	70	00111000	8	8
57	39	71	00111001	9	9
58	3a	72	00111010	:	:
59	3b	73	00111011	;	;
60	3c	74	00111100	<	<
61	3d	75	00111101	=	=
62	3e	76	00111110	>	>
63	3f	77	00111111	?	?
64	40	100	01000000	@	@
65	41	101	01000001	A	A
66	42	102	01000010	B	B
67	43	103	01000011	C	C
68	44	104	01000100	D	D
69	45	105	01000101	E	E
70	46	106	01000110	F	F
71	47	107	01000111	G	G
72	48	110	01001000	H	H
73	49	111	01001001	I	I
74	4a	112	01001010	J	J
75	4b	113	01001011	K	K
76	4c	114	01001100	L	L
77	4d	115	01001101	M	M
78	4e	116	01001110	N	N
79	4f	117	01001111	O	O
80	50	120	01010000	P	P
81	51	121	01010001	Q	Q
82	52	122	01010010	R	R
83	53	123	01010011	S	S
84	54	124	01010100	T	T
85	55	125	01010101	U	U
86	56	126	01010110	V	V

Decimal	*Hexadecimal*	*Octal*	*Binary*	*Key*	*ASCII character*
87	57	127	01010111	W	W
88	58	130	01011000	X	X
89	59	131	01011001	Y	Y
90	5a	132	01011010	Z	Z
91	5b	133	01011011	[	[
92	5c	134	01011100	\	\
93	5d	135	01011101	]	]
94	5e	136	01011110	^	^
95	5f	137	01011111	_	_
96	60	140	01100000	'	'
97	61	141	01100001	a	a
98	62	142	01100010	b	b
99	63	143	01100011	c	c
100	64	144	01100100	d	d
101	65	145	01100101	e	e
102	66	146	01100110	f	f
103	67	147	01100111	g	g
104	68	150	01101000	h	h
105	69	151	01101001	i	i
106	6a	152	01101010	j	j
107	6b	153	01101011	k	k
108	6c	154	01101100	l	l
109	6d	155	01101101	m	m
110	6e	156	01101110	n	n
111	6f	157	01101111	o	o
112	70	160	01110000	p	p
113	71	161	01110001	q	q
114	72	162	01110010	r	r
115	73	163	01110011	s	s
116	74	164	01110100	t	t
117	75	165	01110101	u	u
118	76	166	01110110	v	v
119	77	167	01110111	w	w
120	78	170	01111000	x	x
121	79	171	01111001	y	y
122	7a	172	01111010	z	z
123	7b	173	01111011	{	{
124	7c	174	01111100		
125	7d	175	01111101	}	}
126	7e	176	01111110	~	~
127	7f	177	01111111	CTRL+¬	Δ
128	80	200	10000000	ALT+128	Ç
129	81	201	10000001	ALT+129	ü
130	82	202	10000010	ALT+130	é
131	83	203	10000011	ALT+131	â
132	84	204	10000100	ALT+132	ä
133	85	205	10000101	ALT+133	à
134	86	206	10000110	ALT+134	å
135	87	207	10000111	ALT+135	ç
136	88	210	10001000	ALT+136	ê
137	89	211	10001001	ALT+137	ë
138	8a	212	10001010	ALT+138	è
139	8b	213	10001011	ALT+139	ï
140	8c	214	10001100	ALT+140	î
141	8d	215	10001101	ALT+141	ì
142	8e	216	10001110	ALT+142	Ä
143	8f	217	10001111	ALT+143	Å

Decimal	*Hexadecimal*	*Octal*	*Binary*	*Key*	*ASCII character*
144	90	220	10010000	ALT+144	É
145	91	221	10010001	ALT+145	æ
146	92	222	10010010	ALT+146	Æ
147	93	223	10010011	ALT+147	ô
148	94	224	10010100	ALT+148	ö
149	95	225	10010101	ALT+149	ò
150	96	226	10010110	ALT+150	û
151	97	227	10010111	ALT+151	ù
152	98	230	10011000	ALT+152	
153	99	231	10011001	ALT+153	Ö
154	9a	232	10011010	ALT+154	Ü
155	9b	233	10011011	ALT+155	¢
156	9c	234	10011100	ALT+156	£
157	9d	235	10011101	ALT+157	¥
158	9e	236	10011110	ALT+158	₧
159	9f	237	10011111	ALT+159	ƒ
160	a0	240	10100000	ALT+160	á
161	a1	241	10100001	ALT+161	í
162	a2	242	10100010	ALT+162	ó
163	a3	243	10100011	ALT+163	ú
164	a4	244	10100100	ALT+164	ñ
165	a5	245	10100101	ALT+165	Ñ
166	a6	246	10100110	ALT+166	a
167	a7	247	10100111	ALT+167	o
168	a8	250	10101000	ALT+168	¿
169	a9	251	10101001	ALT+169	⌐
170	aa	252	10101010	ALT+170	¬
171	ab	253	10101011	ALT+171	½
172	ac	254	10101100	ALT+172	¼
173	ad	255	10101101	ALT+173	¡
174	ae	256	10101110	ALT+174	«
175	af	257	10101111	ALT+175	»
176	b0	260	10110000	ALT+176	░
177	b1	261	10110001	ALT+177	▒
178	b2	262	10110010	ALT+178	▓
179	b3	263	10110011	ALT+179	│
180	b4	264	10110100	ALT+180	┤
181	b5	265	10110101	ALT+181	╡
182	b6	266	10110110	ALT+182	╢
183	b7	267	10110111	ALT+183	╖
184	b8	270	10111000	ALT+184	╕
185	b9	271	10111001	ALT+185	╣
186	ba	272	10111010	ALT+186	║
187	bb	273	10111011	ALT+187	╗
188	bc	274	10111100	ALT+188	╝
189	bd	275	10111101	ALT+189	╜
190	be	276	10111110	ALT+190	╛
191	bf	277	10111111	ALT+191	┐
192	c0	300	11000000	ALT+192	└
193	c1	301	11000001	ALT+193	┴
194	c2	302	11000010	ALT+194	┬
195	c3	303	11000011	ALT+195	├
196	c4	304	11000100	ALT+196	─
197	c5	305	11000101	ALT+197	┼
198	c6	306	11000110	ALT+198	╞
199	c7	307	11000111	ALT+199	╟
200	c8	310	11001000	ALT+200	╚

Decimal	*Hexadecimal*	*Octal*	*Binary*	*Key*	*ASCII character*
201	c9	311	11001001	ALT+201	╔
202	ca	312	11001010	ALT+202	╩
203	cb	313	11001011	ALT+203	╦
204	cc	314	11001100	ALT+204	╠
205	cd	315	11001101	ALT+205	═
206	ce	316	11001110	ALT+206	╬
207	cf	317	11001111	ALT+207	╧
208	d0	320	11010000	ALT+208	╨
209	d1	321	11010001	ALT+209	╤
210	d2	322	11010010	ALT+210	╥
211	d3	323	11010011	ALT+211	╙
212	d4	324	11010100	ALT+212	╘
213	d5	325	11010101	ALT+213	╒
214	d6	326	11010110	ALT+214	╓
215	d7	327	11010111	ALT+215	╫
216	d8	330	11011000	ALT+216	╪
217	d9	331	11011001	ALT+217	┘
218	da	332	11011010	ALT+218	┌
219	db	333	11011011	ALT+219	█
220	dc	334	11011100	ALT+220	▄
221	dd	335	11011101	ALT+221	▌
222	de	336	11011110	ALT+222	▐
223	df	337	11011111	ALT+223	▀
224	e0	340	11100000	ALT+224	α
225	e1	341	11100001	ALT+225	ß
226	e2	342	11100010	ALT+226	Γ
227	e3	343	11100011	ALT+227	π
228	e4	344	11100100	ALT+228	∑
229	e5	345	11100101	ALT+229	σ
230	e6	346	11100110	ALT+230	µ
231	e7	347	11100111	ALT+231	τ
232	e8	350	11101000	ALT+232	Φ
233	e9	351	11101001	ALT+233	Θ
234	e9	351	11101010	ALT+234	Ω
235	eb	353	11101011	ALT+235	δ
236	ec	354	11101100	ALT+236	∞
237	ed	355	11101101	ALT+237	ø
238	ee	356	11101110	ALT+238	ε
239	ef	357	11101111	ALT+239	∩
240	f0	360	11110000	ALT+240	≡
241	f1	361	11110001	ALT+241	±
242	f2	362	11110010	ALT+242	≥
243	f3	363	11110011	ALT+243	≤
244	f4	364	11110100	ALT+244	⌠
245	f5	365	11110101	ALT+245	⌡
246	f6	366	11110110	ALT+246	÷
247	f7	367	11110111	ALT+247	≈
248	f8	370	11111000	ALT+248	°
249	f9	371	11111001	ALT+249	•
250	fa	372	11111010	ALT+250	·
251	fb	373	11111011	ALT+251	√
252	fc	374	11111100	ALT+252	^{n}
253	fd	375	11111101	ALT+253	^{2}
254	fe	376	11111110	ALT+254	■
255	ff	377	11111111	ALT+255	blank

Appendix **C**

Operators: Precedence, Associativity and Arity

The following list shows all of the operators in C++. The precedence, associativity and arity of each operator is given. Note that **`sizeof`**, **`new`** and **`delete`** are operators. All of the operators listed can be overloaded except for `::`, `.`, **`sizeof`**, `.*` and `?:`.

Key: L=Left, R=Right, U=Unary, B=Binary, T=Ternary, N/A=Not applicable

Operator	*Precedence*	*Associativity*	*Arity*	*Description*
`::`	17	R	U	Global scope
`::`		L	B	Class scope
`()`	16	L	N/A	Function call
`()`		L	N/A	Type construction
`[]`		L	B	Array index
`->`		L	B	Indirect member access
`.`		L	B	Direct member access
`sizeof`	15	R	U	Object size in bytes
`new`		R	U	Memory allocator
`delete`		R	U	Memory deallocator
`()`		R	B	cast
`-`		R	U	Minus sign
`+`		R	U	Plus sign
`*`		R	U	Pointer dereference
`&`		R	U	reference
`!`		R	U	Logical NOT
`~`		R	U	Bitwise NOT
`++`		R	U	increment
`--`		R	U	decrement
`.*`	14	L	B	Direct member pointer access
`->*`		L	B	Indirect member pointer access
`/`	13	L	B	division
`*`		L	B	multiplication
`%`		L	B	modulus
`+`	12	L	B	addition
`-`		L	B	subtraction
`<<`	11	L	B	Bit-shift left

Operator	*Precedence*	*Associativity*	*Arity*	*Description*
>>		L	B	Bit-shift right
<	10	L	B	Less than
>		L	B	Greater than
<=		L	B	Less than or equal to
>=		L	B	Greater than or equal to
==	9	L	B	equality
!=		L	B	inequality
&	8	L	B	Bitwise AND
^	7	L	B	Bitwise XOR
|	6	L	B	Bitwise OR
&&	5	L	B	Logical AND
||	4	L	B	Logical OR
?:	3	L	T	conditional
=	2	R	B	assignment
+=		R	B	Addition and assignment
-=		R	B	Subtraction and assignment
/=		R	B	Division and assignment
*=		R	B	Multiplication and assignment
%=		R	B	Modulus and assignment
&=		R	B	Bitwise AND and assignment
|=		R	B	Bitwise OR and assignment
^=		R	B	Bitwise XOR and assignment
<<=		R	B	Bit-shift left and assignment
>>=		R	B	Bit-shift right and assignment
,	1	L	B	Comma (sequencing)

Appendix D

Glossary

abstraction	The essential characteristics of an object which distinguish it from other objects.
array	Group of identical data types or elements.
base class	A **`class`** from which another class can inherit. Also known as a superclass.
character	Such as /, *, ?, a, A, b, ... Refer to Appendix B for a complete listing of the ASCII character set.
`class`	A **`class`** encapsulates data and functions which operate on the **`class`**'s data, defining a given structure and behaviour. A **`class`** is a type and a scope with an exact specification specified in the **`class`** declaration.
comment	A user-defined string of information placed within program code that does not influence compilation.
compile	Convert a program source code file in to an object file which contains machine language instructions.
concrete class	A **`class`** that possesses the functionality of C++'s built-in data types, such as **`int`** and **`double`**.
concurrency	The action of different objects operating simultaneously.
constant	Something that does not change.
declaration	Specifies an identifier's name and type.
definition	A declaration that allocates memory.
derived	A **`class`** that has inherited from one or more alternative classes. Also known as a subclass.
encapsulation	The formulation of separating the structure and behaviour of an abstraction into cells.
`enum`	An enumeration data type which provides mnemonic identifiers to a set of integer constants.

exception	A situation in which a program encounters an abnormal situation for which it was not designed. In C++ you can transfer control (**throw** an exception) to another part of a program that is designed to deal explicitly with exceptions.
execution	The process of running a program.
friend	A **class** or function that has access to the **private** sections of another **class**.
function	A standalone module that can have input or output or both.
identifier	A name given to either a constant or a variable object.
inheritance	A hierarchy among classes.
instance	An instance does things, and we can do things to an instance by sending messages. Messages are sent via member functions. An instance is also referred to as an *object*.
interface	A set of functions within a module.
keyword	Such as **int**, **double** or **new**. A word that the compiler already has a definition for. A keyword (or reserved word) cannot be used as the name of a data object. Keywords in C++ are in lowercase. For a list of keywords refer to Appendix A.
link	*Linking* is the process of combining object files into a single executable program.
make	Combined process of compiling and linking to generate an executable program.
member function	An operation on an object. Member functions are called *methods* or *operations* in other object-oriented programming languages.
module	A piece of program code which forms part of a larger software system. A module has a clearly defined interface. A source file is an example of a module.
object	An object does things, and we can do things to an object by sending messages. Messages are sent via member functions. An object is also referred to as an *instance*.
persistence	An object is referred to as persistent if it outlives its creator or program execution and/or an object survives a move from its original memory address.
polymorphism	A name that is attached to a variable which denotes objects of several different classes that are related through a common base **class**. Polymorphous – assuming many forms.
pragmatics	Rules for the good and proper use of a language.
preprocessor	The compiler's preprocessor. Preprocessor directives are prefixed with the number (or hash) sign, #.
primitives	Primary subsystems of a complex system such as base classes.

private	If a member of **class** is **private** it can only be accessed by member functions and **friend**s of the **class** in which it is declared.
protected	If a member of a **class** is **protected** it can only be accessed by member functions and **friend**s of the **class** in which it is declared or a **class** that is *derived* from this **class**.
public	If a member of a **class** is **public** it can be accessed by any function, member or not.
run-time type information	A mechanism which allows information to be obtained about an object at run-time.
scope	The visibility, availability and life of an object within a program. There are generally six different types of scope: global, local, **class**, function, **namespace** and file.
semantics	The meaning of syntactically correct constructs of a language.
source file	A file containing declarations, functions and objects. Frequently referred to as a translation unit.
STL	The C++ Standard Template Library.
structure	A user-defined declaration that encapsulates data members and member functions or operations.
syntax	The set of rules for the correct use of a language.
template	A generic **class**. Often referred to as a *parametrised type*.
translation unit	A module.
type	A **class** implements a type. A type determines what values and operations can be performed on an object. All names or objects must have a type in C++.
type checking	All objects are created with a specific type, such as **int**, **double** and `Complex*`. Type checking involves checking that the use of an object is consistent with its type. C++ is often referred to as *strongly typed*. If violations of type are detected at compilation, a language is referred to as *strongly statically typed*. Alternatively, if violations of type are allowed to pass compilation and type checking is postponed until run-time then a language is referred to as *dynamically typed*.
union	Similar to a *structure*, with the restriction that a **union** can hold only one of its members at any given time.
variable	As the name suggests, something that changes. The name of a storage location that depends on the type and size of the data that it contains.
virtual function	A base **class** member function that can be redefined by each derived **class**.

References

Ada Reference Manual (1983) *Reference Manual for the Ada Programming Language*, February 1983, Washington DC: Department of Defence, Ada Joint Program Office.

Adams, J., Leestma, S. and Nyhoff, L. (1995) *C++: An Introduction to Computing*, Prentice Hall, Englewood Cliffs NJ.

Ammerau, L. (1997) *STL for C++ Programmers*, John Wiley & Sons, Chichester, England.

Anton, H. (1991) *Elementary Linear Algebra*, 6th edn, John Wiley & Sons, New York.

Anuff, E. (1996) *The Java Sourcebook*, John Wiley & Sons, New York.

Austen, M. H. (1999) *Generic Programming and the STL: Using and Extending the C++ Standard Template Library*, Addison-Wesley, Reading, MA.

Barton, J. J. and Nackman, L. R. (1994) *Scientific and Engineering C++: An Introduction with Advanced Techniques and Examples*, Addison-Wesley, Reading MA.

Bashein, G. and Detmer, P. R. (1994) Centroid of a polygon, in *Graphics Gems IV* (ed. Heckbert, P. S.), Academic Press, London.

Bauby, J.-D. (1998) *The Diving-Bell and the Butterfly*, Fourth Estate, London.

Bobrow, D., DeMichiel, L., Gabriel, R., Keene, S., Kiczales, G. and Moon, D. (1988) *Common Lisp Object System Specification*, X3J13 Document 88-002R, September 1988, *SIGPLAN Notices* **23**.

Booch, G. (1994) *Object-Oriented Analysis and Design with Applications*, 2nd edn, Benjamin/Cummings, Redwood City CA.

Booch, G., Jacobson, I. and Rumbaugh, J. (1999) *The Unified Modelling Language User Guide*, Addison-Wesley, Reading MA.

Bowyer, A. (1995) *SVLIS Set-Theoretic Kernel Modeller: Introduction and User Manual*, 2nd edn, Information Geometers, Winchester. See also The SVLIS Web page: `http://www.bath.ac.uk/~ensab/G_mod/Svlis/svlis.html`.

Bowyer, A. and Woodwark, J. (1983) *A Programmer's Geometry*, Butterworths, London.

Bowyer, A. and Woodwark, J. (1993) *Introduction to Computing with Geometry*, Information Geometers, Winchester.

Budd, T. A. (1987) *A Little Smalltalk*, Addison-Wesley, Reading MA.

Budd, T. A. (1994) *Classic Data Structures in C++*, Addison-Wesley, Reading MA.

Calvert, C. (1997) *C++Builder*, Sams Publishing, Indianpolis.

Coad, P. and Nicola, J. (1993) *Object-Oriented Programming*, Yourdon Press, New Jersey.

Coplien, J. O. (1992) *Advanced C++ Programming Styles and Idioms*, Addison-Wesley, Reading MA.

Cormen, T. H., Leiserson, C. E. and Rivest, R. L. (1990) *Introduction to Algorithms*, MIT Press and McGraw-Hill, New York.

Corney, J. (1997) *3D Modelling with the ACIS Kernel and Toolkit*, John Wiley & Sons, Chichester.

Dahl, O.-J., Myrhaug, B. and Nygaard, K. (1970) *SIMULA Common Base Language*, Norwegian Computing Centre S-22, Oslo.

Devlin, K. (2000) *The Maths Gene: Why Everyone has it, but most People don't use it*, Weidenfeld & Nicolson, London.

Ellis, M. A. and Stroustrup, B. (1990) *The Annotated C++ Reference Manual*, Addison-Wesley, Reading MA.

Ford, W. and Topp, W. (1996) *Data Structures in C++*, Prentice Hall, Englewood Cliffs NJ.

Gifford, B. (1990) *Wild at Heart*, Grafton, London.

Glaeser, G. (1994) *Fast Algorithms for 3D-Graphics*, Springer-Verlag, London.

Goldberg, A. and Robson, D. (1983) *Smalltalk-80: The Language and its Implementation*, Addison-Wesley, Reading MA .

Golub, G. H. and Van Loan, C. F. (1989) *Matrix Computations*, 2nd edn, Johns Hopkins University Press, Baltimore MD.

Graham, R. L. (1972) *An Efficient Algorithm for Determining the Convex Hull of a Finite Planar Set, Info. Proc. Lett.*, 1, 132–3.

Hearn, D. and Baker, M. P. (1994) *Computer Graphics*, 2nd edn, Prentice Hall, Englewood Cliffs NJ.

Horowitz, E., Sahni, S. and Mehta, D. (1995) *Fundamentals of Data Structures in C++*, Computer Science Press, New York.

Jacobson, I., Booch, G. and Rumbaugh, J. (1999) *The Unified Software Development Process*, Addison-Wesley, Reading MA.

Jennings, A. and McKeown, J. J. (1992) *Matrix Computation*, 2nd edn, John Wiley & Sons, New York.

KAI (2000) *The KAI Web page*, `http://www.kai.com/`.

Kay, A. C. (1993) *The Early History of Smalltalk*, ACM SIGPLAN Second History of Programming Languages Conference, 20–23 April 1993, ACM, pp. 69–95.

Kernighan B. and Ritchie D. (1978) *The C Programming Language*, 1st edn, Prentice Hall, Englewood Cliffs NJ.

Kernighan B. and Ritchie D. (1988) *The C Programming Language*, 2nd edn, Prentice Hall, Englewood Cliffs NJ.

Kirkerud, B. (1989) *Object-Oriented Programming with Simula*, Addison-Wesley, Reading MA.

Lafore, R. (1991) *Object-Oriented Programming in Turbo C++*, Waite Group Press, Mill Valley CA.

Lafore, R. (1993) *Lafore's Windows Programming Made Easy*, Waite Group Press, Mill Valley CA.

Laszlo, M. J. (1996) *Computational Geometry and Computer Graphics in C++*, Prentice Hall, Englewood Cliffs NJ.

Lippman, S. B. (1991) *C++ Primer*, 2nd edn, Addison-Wesley, Reading MA.

Luse, M. (1993) *Bitmapped Graphics Programming in C++*, Addison-Wesley, Reading MA.

Masters, T. (1993) *Practical Neural Network Recipes in C++*, Academic Press, London.

Meyer, B. (1991) *Eiffel: The Language*, Prentice Hall, Englewood Cliffs NJ.

Meyer, B. (1995) *Object Success: A Manager's Guide to Object Orientation, its Impact on the Corporation, and its Use for Reengineering the Software Process*, Prentice Hall, Englewood Cliffs NJ.

Mortenson, M. E. (1989) *Computer Graphics: An Introduction to the Mathematics and Geometry*, Industrial Press, New York.

Mössenböck, H. (1993) *Object-Oriented Programming in Oberon-2*, Springer-Verlag, London.

Musser, D. and Saini, A. (1996) *STL Tutorial and Reference Guide: C++ Programming with the Standatd Template Library*, Addison-Wesley, Reading, MA.

Negroponte, N. (1995) *being digital*, Hodder & Stoughton, London.

O'Rourke, J. (1994) *Computational Geometry in C*, Cambridge University Press, Cambridge.

OpenGL (2000) *The OpenGL Web page*, `http://www.opengl.org/`.

Peitgen, H.-O., Jurgens, H. and Saupe, D. (1992) *Chaos and Fractals, New Fractals of Science*, Springer-Verlag, London.

Petzold, C. (1992) *Programming Windows 3.1*, 3rd edn, Microsoft Press, Redmond WA.

Phong, B. T. (1975) *Illumination for Computer-Generated Pictures*, *Commun. ACM*, **18**(6), 311–17.

Pinson, L. J. and Wiener, R. S. (1988) *An Introduction to Object-Oriented Programming and Smalltalk*, Addison-Wesley, Reading MA.

Plastock, R. A. and Kalley, G. (1986) *Theory and Problems of Computer Graphics*, Schaum's Outline Series, McGraw-Hill, New York.

Plauger, P. J. (1995) *The Draft Standard C++ Library*, Prentice Hall, Englewood Cliffs NJ.

Porter, A. (1993) *C++ Programming for Windows*, Osborne McGraw-Hill, Maidenhead.

Preparata, F. P. and Shamos, M. I. (1985) *Computational Geometry: An Introduction*, Springer-Verlag, London.

Pugh, W. (1989) *Skip lists: a probabilistic alternative to balanced trees*, in *Proc. Workshop Algorithms and Data Structures*, Ottawa Canada, August 1989.

Python (2000) *The Python web page*, `http://www.python.org/`.

Rao, V. B. and Rao, H. V. (1993) *C++ Neural Networks and Fuzzy Logic*, MIS Press, New York.

Reisdorph, K. and Henderson, K. (1997) *Teach Yourself Borland C++Builder in 21 Days*, Sams Publishing, Indianapolis.

Reiser, M. and Wirth, N. (1992) *Programming in Oberon – Steps Beyond Pascal and Modula-2*, Addison-Wesley, Reading MA.

Richards, M. and Whitney-Strevens, C. (1979) *BCPL: The Language and its Compiler*, Cambridge University Press, Cambridge.

Ritchie, D. M. (1993) *The Development of the C Language*, ACM SIGPLAN Second History of Programming Languages Conference, 20–23 April 1993, ACM, pp. 201–8.

Robson, R. (1997) *Using the STL: the C++ Standard Template Library*, Springer-Verlag, New York.

Rumbaugh, J., Jacobson, I. and Booch, G. (1999) *The Unifed Modelling Language Reference Manual*, Addison-Wesley, Reading MA.

Saunders, J. (1989) *A Survey of Object-Oriented Programming Languages*, Journal of Object-Oriented Programming, **1** (6).

Schildt, H. (1994) *C++ from the Ground Up*, Osborne McGraw-Hill, Maidenhead.

Schildt, H. (1995) *C++: The Complete Reference*, 2nd edn, Osborne McGraw-Hill, Maidenhead.

Sedgewick, R. (1988) *Algorithms*, 2nd edn, Addison-Wesley, Reading MA.

Sedgewick, R. (1992) *Algorithms in C++*, Addison-Wesley, Reading MA.

Snyder, J. M. and Barr, A. H. (1987) *Raytracing Complex Models Containing Surface Tessellations*, SIGGRAPH 87, pp. 119–28.

Standard (1998) *Programming Languages-C++* (ISO/IEC 14882:1998). The C++ standard can be obtained on-line from the American National Standards Institute (ANSI) at `http://www.ansi.org/`.

Stevens, T. (1992) *Fractal Programming and Ray Tracing with C++*, Prentice Hall, Englewood Cliffs NJ.

Stevens, R. T. (1994) *Object-Oriented Graphics Programming in C++*, Academic Press, London.

Stroustrup, B. (1982) *Classes: An Abstract Data Type Facility for the C Language*, ACM SIGPLAN Notices.

Stroustrup, B. (1986) *The C++ Programming Language*, Addison-Wesley, Reading MA.

Stroustrup, B. (1991) *The C++ Programming Language*, 2nd edn, Addison-Wesley, Reading MA.
Stroustrup, B. (1993a) *A History of C++: 1979–1991*, ACM SIGPLAN Second History of Programming Languages Conference, 20–23 April 1993, ACM, pp. 271–7.
Stroustrup, B. (1993b) Why consider language extensions?, in *Proc. European C++ User Group Technical Conference*, 7 March–9 July 1993, pp. 50–63.
Stroustrup, B. (1994) *The Design and Evolution of C++*, Addison-Wesley, Reading MA.
Stroustrup, B. (1997) *The C++ Programming Language*, 3rd edn, Addison-Wesley, Reading MA.
Sun (2000) *The Sun Java web page*, `http://www.java.sun.com/`.
Swan, T. (1991) *Tom Swan's GNU C++ for Linux*, MacMillan, New York.
Sykes, C. (1994) *No Ordinary Genius, The Illustrated Richard Feynman*, Weidenfield & Nicolson, London.
Szyperski, C. (1998) *Component Software: Beyond Object-Oriented Programming*, Addison-Wesley, Reading MA.
Thomas, P. and Weedon, R. (1995) *Object-Oriented Programming in Eiffel*, Addison-Wesley, Reading MA.
USENIX (1987) Sante Fe NM, C++ Workshop, November.
USENIX (1988) Denver CO, C++ Conference, October.
USENIX (1990) San Francisco CA, C++ Conference, April.
USENIX (1991) Washington DC, C++ Conference, April.
USENIX (1992) Portland OR, C++ Technical Conference, August.
USENIX (1993) Munich, European C++ User Group Technical Conference, July.
Watt, A. and Watt, M. (1992) *Advanced Animation and Rendering Techniques: Theory and Practice*, ACM Press, New York.
Weiss, M. A. (1996) *Algorithms, Data Structures and Problem Solving with C++*, Addison-Wesley, Reading MA.
Wells, D. and Young, C. (1993) *Raytracing Creations: Generate 3D Photo-Realistic Images on the PC* (with a chapter by Farmer, D.), Waite Group Press, Mill Valley CA.
Wilt, N. (1994) *Object-Oriented Ray Tracing in C++*, John Wiley & Sons, New York.
Yao, P. (1994) *Borland C++ 4.0 Programming for Windows*, Borland Press, Scotts Valley CA.

Index

The order of the operators coincides with the precedence levels of Appendix C.