Object Oriented Programming by C++

C++ Basic (2/2)

Variables, Constants, Standard I/O, Expression, and Operators

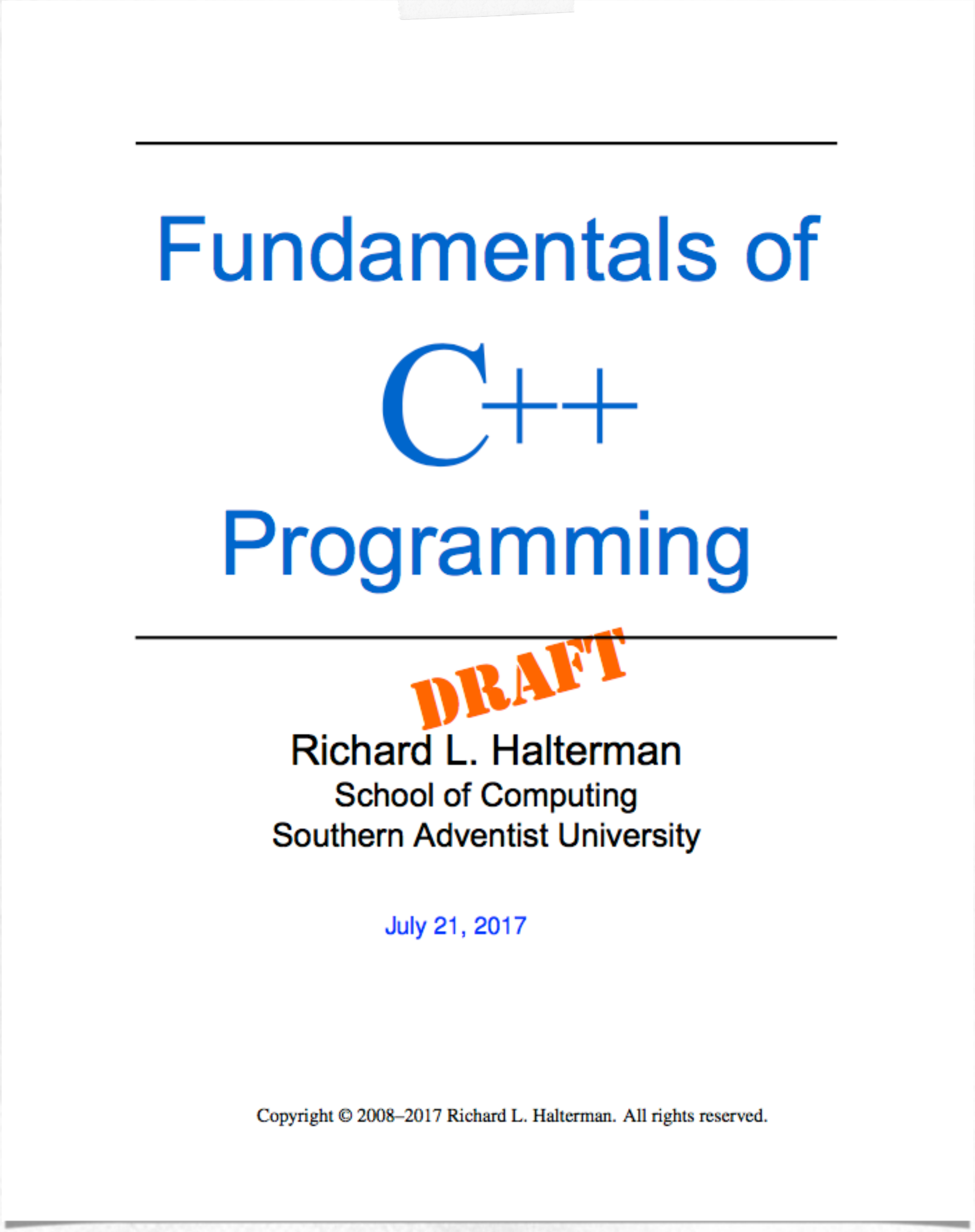
2017. 8.

Sungwon Lee / Professor

Email: drsungwon@khu.ac.kr
Web: http://mobilelab.khu.ac.kr/

Textbook & Copyright

- Textbook: http://python.cs.southern.edu/cppbook/progcpp.pdf
- Sample Codes: https://github.com/halterman/CppBook-SourceCode



Fundamentals of

C++

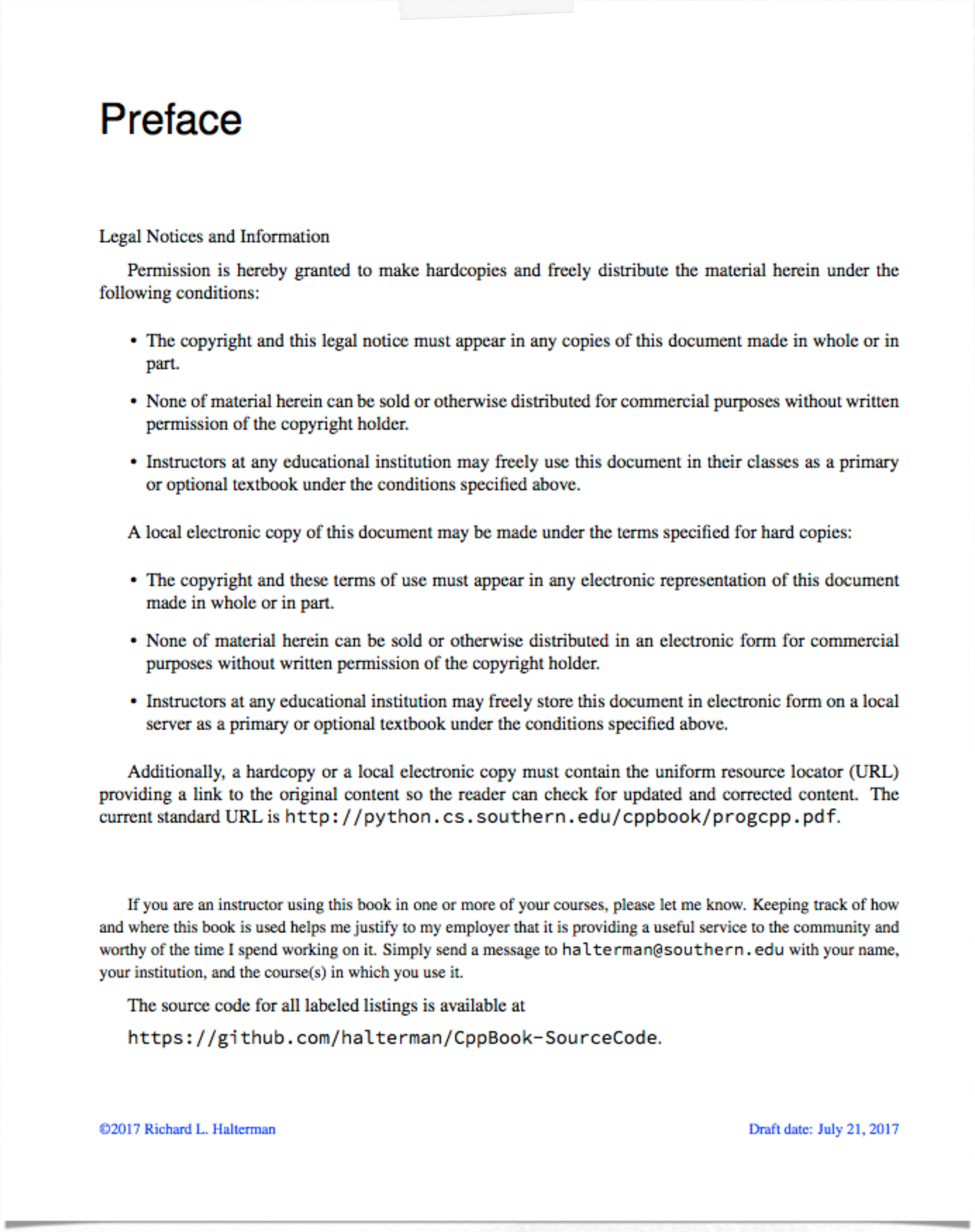
Programming

DRAFT

Richard L. Halterman
School of Computing
Southern Adventist University

July 21, 2017

Copyright © 2008–2017 Richard L. Halterman. All rights reserved.



Preface

Legal Notices and Information

Permission is hereby granted to make hardcopies and freely distribute the material herein under the following conditions:

- The copyright and this legal notice must appear in any copies of this document made in whole or in part.
- None of material herein can be sold or otherwise distributed for commercial purposes without written permission of the copyright holder.
- Instructors at any educational institution may freely use this document in their classes as a primary or optional textbook under the conditions specified above.

A local electronic copy of this document may be made under the terms specified for hard copies:

- The copyright and these terms of use must appear in any electronic representation of this document made in whole or in part.
- None of material herein can be sold or otherwise distributed in an electronic form for commercial purposes without written permission of the copyright holder.
- Instructors at any educational institution may freely store this document in electronic form on a local server as a primary or optional textbook under the conditions specified above.

Additionally, a hardcopy or a local electronic copy must contain the uniform resource locator (URL) providing a link to the original content so the reader can check for updated and corrected content. The current standard URL is `http://python.cs.southern.edu/cppbook/progcpp.pdf`.

If you are an instructor using this book in one or more of your courses, please let me know. Keeping track of how and where this book is used helps me justify to my employer that it is providing a useful service to the community and worthy of the time I spend working on it. Simply send a message to `halterman@southern.edu` with your name, your institution, and the course(s) in which you use it.

The source code for all labeled listings is available at

`https://github.com/halterman/CppBook-SourceCode`.

©2017 Richard L. Halterman Draft date: July 21, 2017

Contents

- Identifier
- Variables
- Constants
- Standard Input & Output
- Operators

Identifier

Naming for things (Variable, Constants, etc.,)

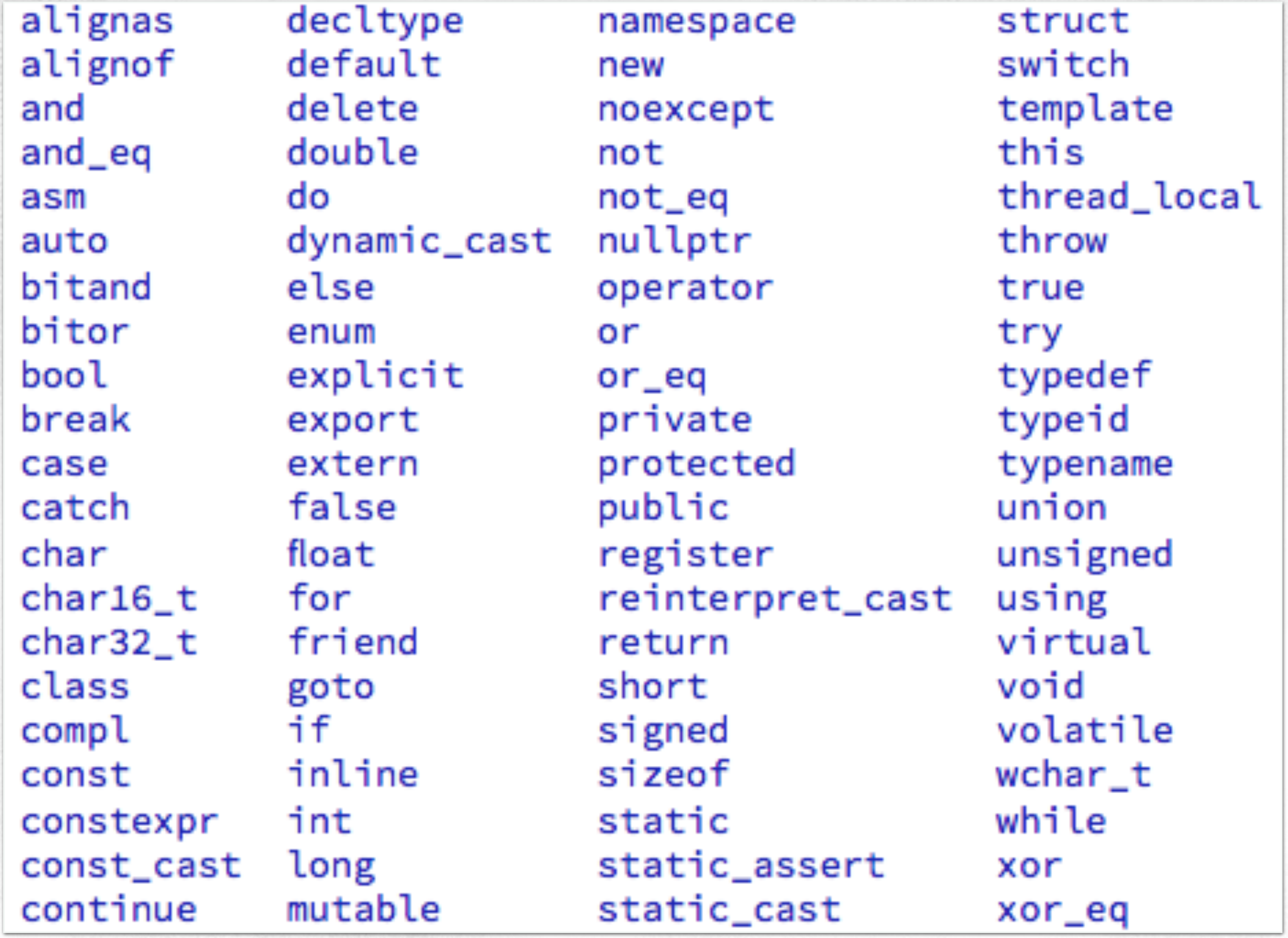
- While mathematicians are content with giving their variables one-letter names like *x*, programmers should use longer, more descriptive variable names.
- A variable name is one example of an identifier.
- C++ has strict rules for variable names:
  - Identifiers must contain at least one character.
  - The first character must be an alphabetic letter (upper or lower case) or the underscore

    *ABCDEFGHIJKLMNOPQRSTUVWXYZabcdefghijklmnopqrstuvwxyz_*
  - The remaining characters (if any) may be alphabetic characters (upper or lower case), the under score, or a digit

    ABCDEFGHIJKLMNOPQRSTUVWXYZabcdefghijklmnopqrstuvwxyz_0123456789
  - No other characters (including spaces) are permitted in identifiers.
  - C++ is a case-sensitive language.
  - A reserved word cannot be used as an identifier.

Identifier

C++ Reserved Words Example



alignas
alignof
and
and_eq
asm
auto
bitand
bitor
bool
break
case
catch
char
char16_t
char32_t
class
compl
const
constexpr
const_cast
continue
decltype
default
delete
double
do
dynamic_cast
else
enum
explicit
export
extern
false
float
for
friend
goto
if
inline
int
long
mutable
namespace
new
noexcept
not
not_eq
nullptr
operator
or
or_eq
private
protected
public
register
reinterpret_cast
return
short
signed
sizeof
static
static_assert
static_cast
struct
switch
template
this
thread_local
throw
true
try
typedef
typeid
typename
union
unsigned
using
virtual
void
volatile
wchar_t
while
xor
xor_eq

Integer

- Line 04:
  - *Declaration* statement.
  - All variables in a C++ program must be declared.
  - A declaration specifies the type of a variable.
  - The word int indicates that the variable is an integer.
  - The name of the integer variable is x.
  - We say that variable x has type int.

```
01: #include <iostream>
02:
03: int main() {
04:     int x;
05:     x = 10;
06: }
```

Variables

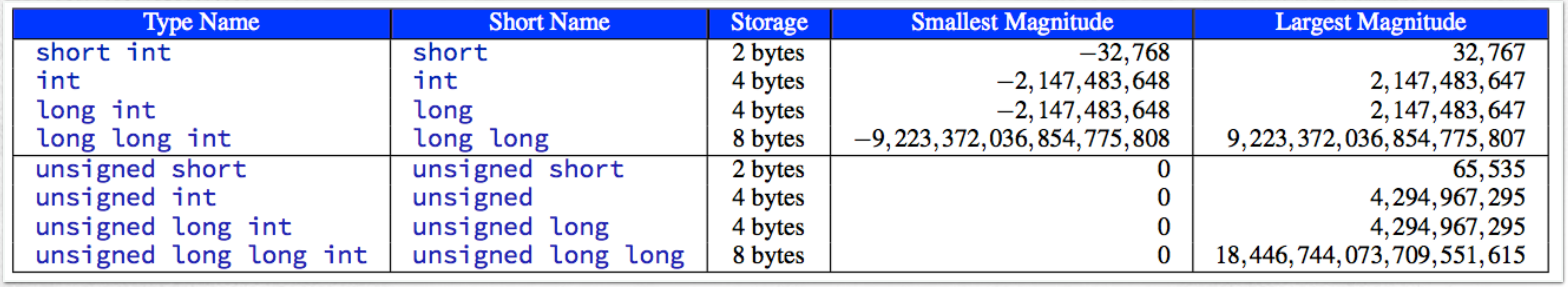
Integer

- Line 05:
  - *Assignment* statement.
  - An assignment statement associates a value with a variable.
  - The key to an assignment statement is the symbol ‘=’ which is known as the assignment operator.
  - Here the value 10 is being assigned to the variable x.
  - This means the value 10 will be stored in the memory location the compiler has reserved for the variable named x.

```
01: #include <iostream>
02:
03: int main() {
04:     int x;
05:     x = 10;
06: }
```

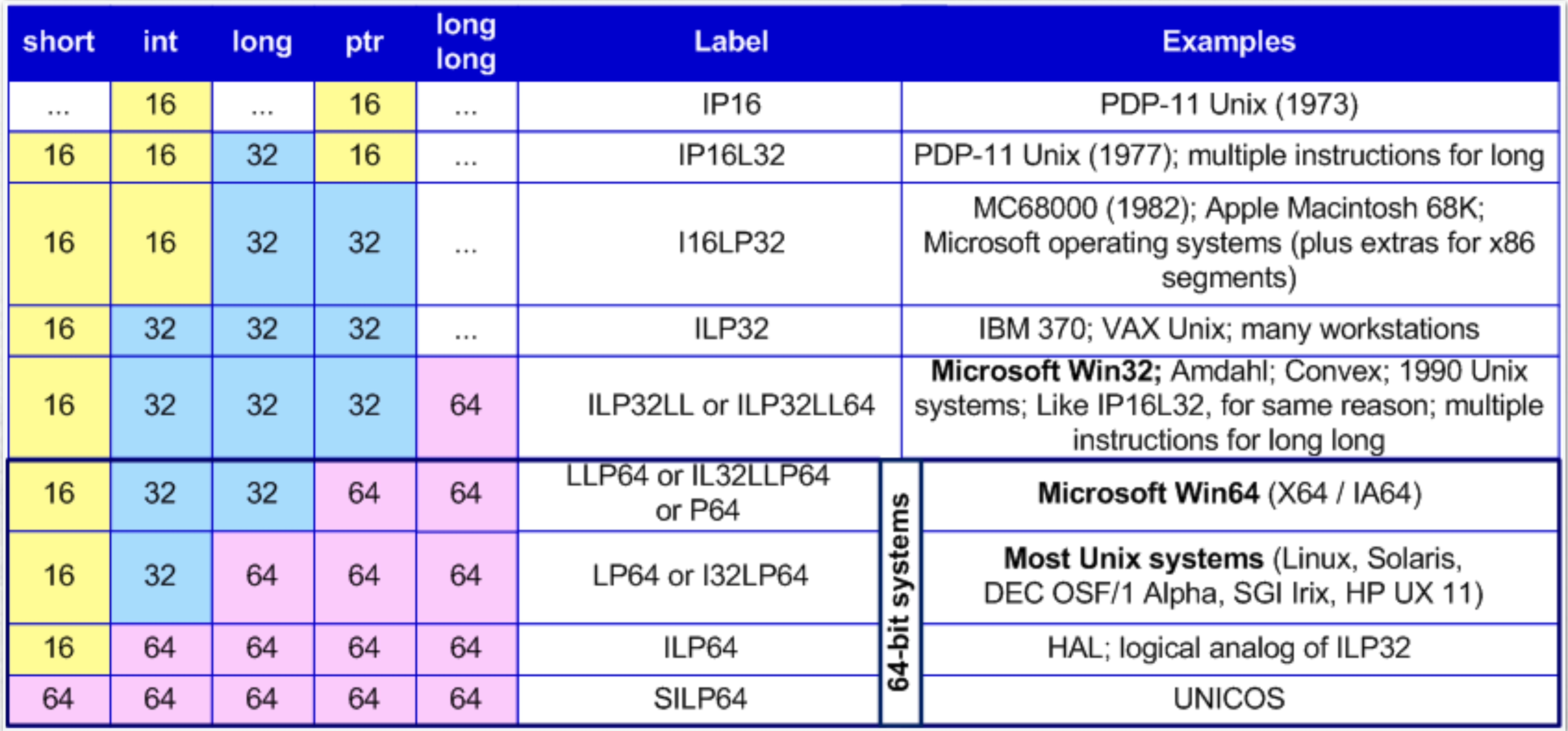
Integer

- Additional Integer Types
  - Example: 32 bit computer system case (Microsoft Visual C++)



Type Name	Short Name	Storage	Smallest Magnitude	Largest Magnitude
`short int`	`short`	2 bytes	−32,768	32,767
`int`	`int`	4 bytes	−2,147,483,648	2,147,483,647
`long int`	`long`	4 bytes	−2,147,483,648	2,147,483,647
`long long int`	`long long`	8 bytes	−9,223,372,036,854,775,808	9,223,372,036,854,775,807
`unsigned short`	`unsigned short`	2 bytes	0	65,535
`unsigned int`	`unsigned`	4 bytes	0	4,294,967,295
`unsigned long int`	`unsigned long`	4 bytes	0	4,294,967,295
`unsigned long long int`	`unsigned long long`	8 bytes	0	18,446,744,073,709,551,615

  - Example: Integer size dependency on machines



short	int	long	ptr	long long	Label		Examples
...	16	...	16	...	IP16		PDP-11 Unix (1973)
16	16	32	16	...	IP16L32		PDP-11 Unix (1977); multiple instructions for long
16	16	32	32	...	I16LP32		MC68000 (1982); Apple Macintosh 68K; Microsoft operating systems (plus extras for x86 segments)
16	32	32	32	...	ILP32		IBM 370; VAX Unix; many workstations
16	32	32	32	64	ILP32LL or ILP32LL64		**Microsoft Win32;** Amdahl; Convex; 1990 Unix systems; Like IP16L32, for same reason; multiple instructions for long long
16	32	32	64	64	LLP64 or IL32LLP64 or P64	64-bit systems	**Microsoft Win64** (X64 / IA64)
16	32	64	64	64	LP64 or I32LP64		**Most Unix systems** (Linux, Solaris, DEC OSF/1 Alpha, SGI Irix, HP UX 11)
16	64	64	64	64	ILP64		HAL; logical analog of ILP32
64	64	64	64	64	SILP64		UNICOS

Reference: https://www.viva64.com/en/t/0012/

Variables

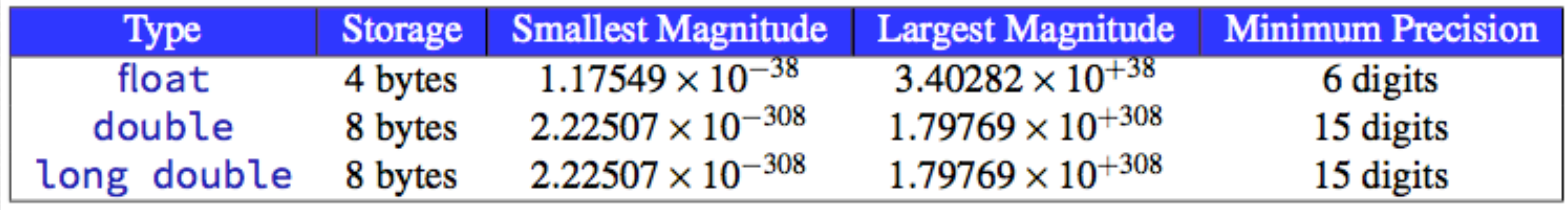
Integer

- Various types and Initialization

```
#include <iostream>

int main() {
    int x1 = 10;
    int x2, y1, z1;
    int x3 = 0, y2, z2 = 5;
    long x4 = 4456;
}
```

Floating-Point

- C++ supports such non-integer numbers, and they are called floating-point numbers.
- The name comes from the fact that during mathematical calculations the decimal point can move or "float" to various positions within the number to maintain the proper number of significant digits.
- Example: 32 bit computer system case (Microsoft Visual C++)



Type	Storage	Smallest Magnitude	Largest Magnitude	Minimum Precision
float	4 bytes	1.17549×10^{-38}	$3.40282 \times 10^{+38}$	6 digits
double	8 bytes	2.22507×10^{-308}	$1.79769 \times 10^{+308}$	15 digits
long double	8 bytes	2.22507×10^{-308}	$1.79769 \times 10^{+308}$	15 digits

Variables

Floating-Point

- Floating-point variable example

```
#include <iostream>

int main() {
   double pi = 3.14159;
}
```

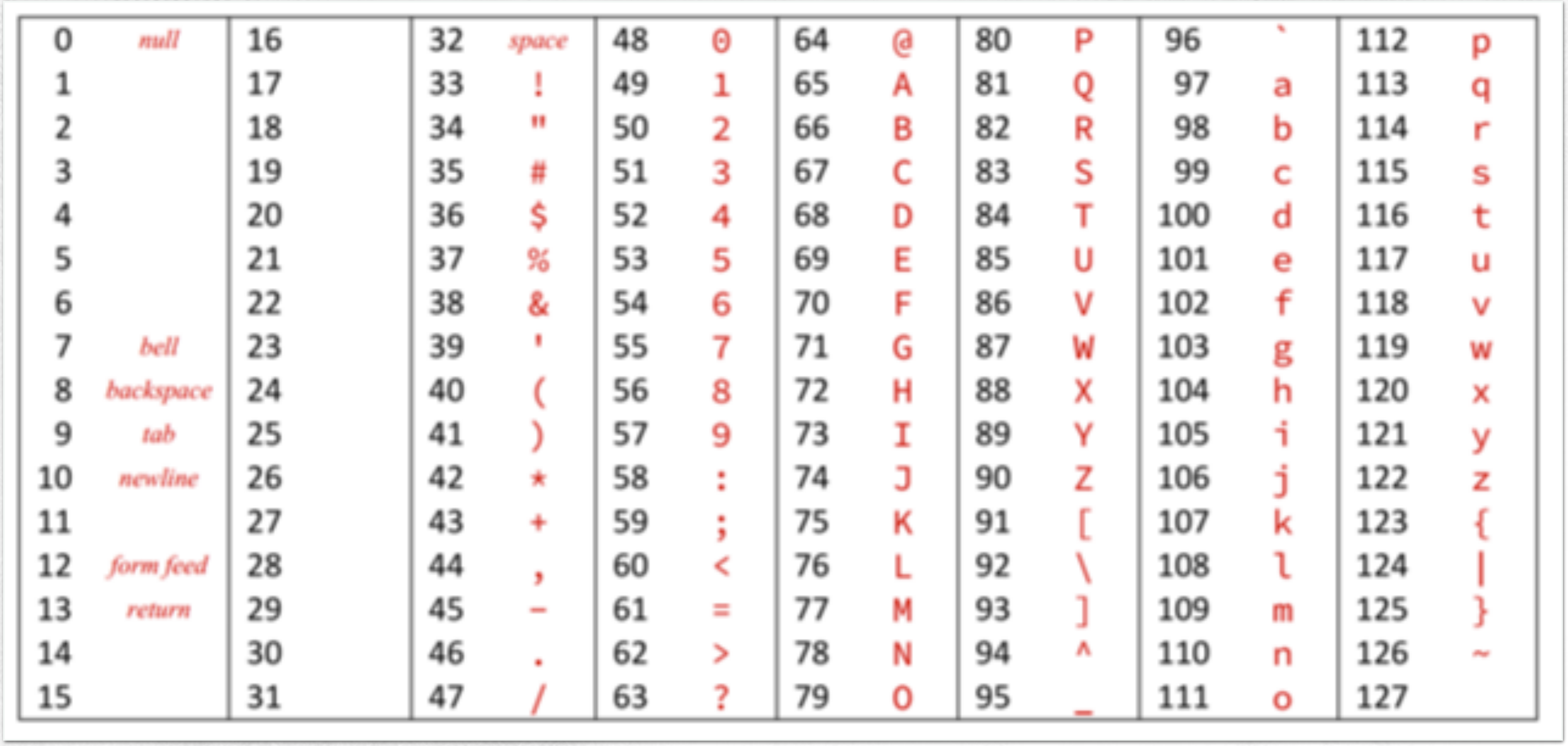
Variables

Character

- The *char* data type is used to represent single characters: letters of the alphabet (both upper and lower case), digits, punctuation, and control characters (like newline and tab characters).
- Most systems support the American Standard Code for Information Interchange (ASCII) character set.

```
#include <iostream>

int main() {
   char ch1, ch2;

   /* ASCII code number */
   ch1 = 65;

   /* ASCII code */
   ch2 = ‘A’;
}
```

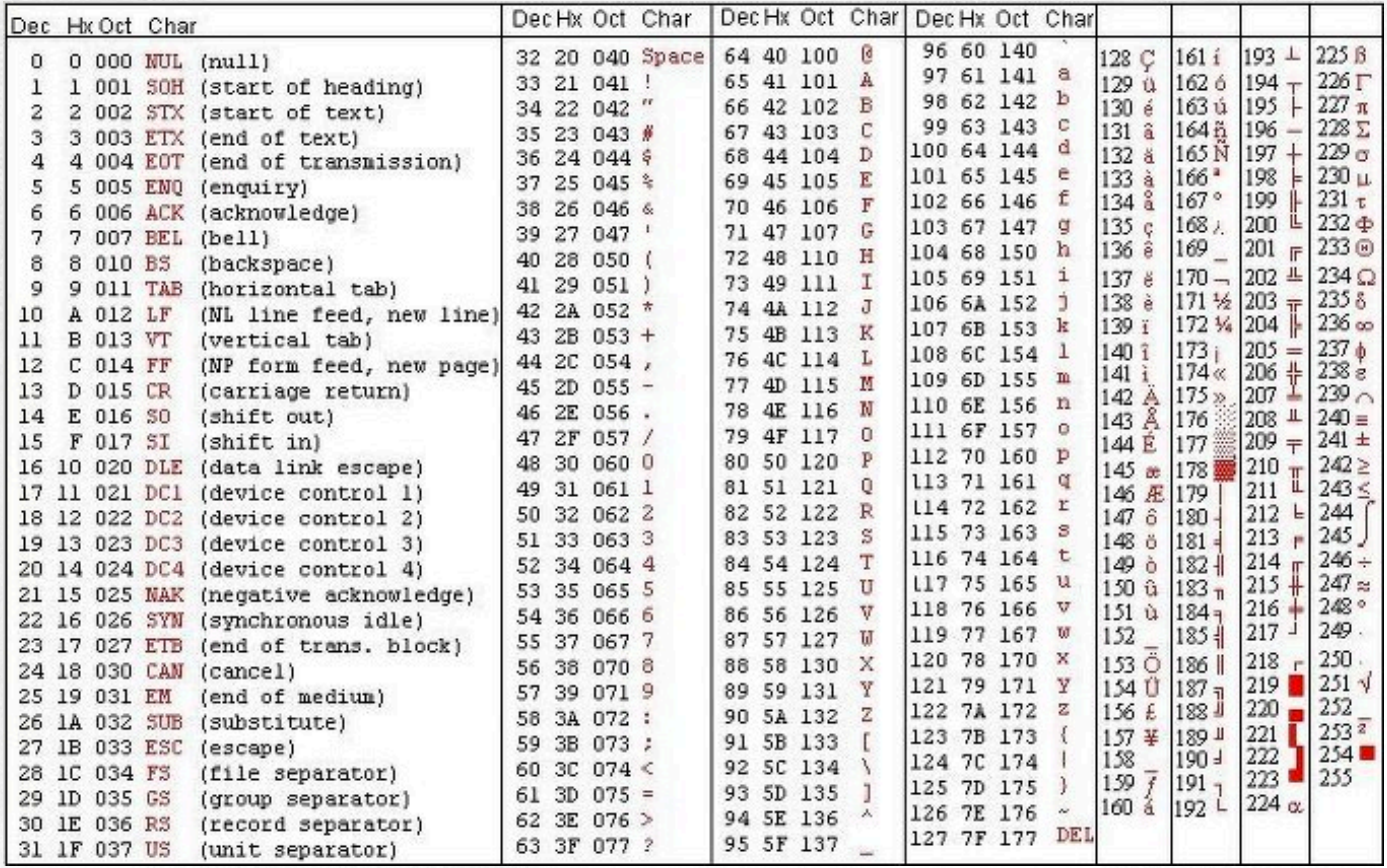
ASCII Code



0	*null*	16		32	*space*	48	0	64	@	80	P	96	`	112	p
1		17		33	!	49	1	65	A	81	Q	97	a	113	q
2		18		34	"	50	2	66	B	82	R	98	b	114	r
3		19		35	#	51	3	67	C	83	S	99	c	115	s
4		20		36	$	52	4	68	D	84	T	100	d	116	t
5		21		37	%	53	5	69	E	85	U	101	e	117	u
6		22		38	&	54	6	70	F	86	V	102	f	118	v
7	*bell*	23		39	'	55	7	71	G	87	W	103	g	119	w
8	*backspace*	24		40	(	56	8	72	H	88	X	104	h	120	x
9	*tab*	25		41	)	57	9	73	I	89	Y	105	i	121	y
10	*newline*	26		42	*	58	:	74	J	90	Z	106	j	122	z
11		27		43	+	59	;	75	K	91	[	107	k	123	{
12	*form feed*	28		44	,	60	<	76	L	92	\	108	l	124	|
13	*return*	29		45	-	61	=	77	M	93	]	109	m	125	}
14		30		46	.	62	>	78	N	94	^	110	n	126	~
15		31		47	/	63	?	79	O	95	_	111	o	127	

Variables

Extended ASCII Code



Dec	Hx	Oct	Char	Dec	Hx	Oct	Char	Dec	Hx	Oct	Char	Dec	Hx	Oct	Char								
0	0	000	NUL (null)	32	20	040	Space	64	40	100	@	96	60	140	`	128	Ç	161	í	193	┴	225	ß
1	1	001	SOH (start of heading)	33	21	041	!	65	41	101	A	97	61	141	a	129	ü	162	ó	194	┬	226	Γ
2	2	002	STX (start of text)	34	22	042	"	66	42	102	B	98	62	142	b	130	é	163	ú	195	├	227	π
3	3	003	ETX (end of text)	35	23	043	#	67	43	103	C	99	63	143	c	131	â	164	ñ	196	─	228	Σ
4	4	004	EOT (end of transmission)	36	24	044	$	68	44	104	D	100	64	144	d	132	ä	165	Ñ	197	┼	229	σ
5	5	005	ENQ (enquiry)	37	25	045	%	69	45	105	E	101	65	145	e	133	à	166	ª	198	╞	230	µ
6	6	006	ACK (acknowledge)	38	26	046	&	70	46	106	F	102	66	146	f	134	å	167	º	199	╟	231	τ
7	7	007	BEL (bell)	39	27	047	'	71	47	107	G	103	67	147	g	135	ç	168	¿	200	╚	232	Φ
8	8	010	BS (backspace)	40	28	050	(	72	48	110	H	104	68	150	h	136	ê	169	_	201	╔	233	Θ
9	9	011	TAB (horizontal tab)	41	29	051	)	73	49	111	I	105	69	151	i	137	ë	170	¬	202	╩	234	Ω
10	A	012	LF (NL line feed, new line)	42	2A	052	*	74	4A	112	J	106	6A	152	j	138	è	171	½	203	╦	235	δ
11	B	013	VT (vertical tab)	43	2B	053	+	75	4B	113	K	107	6B	153	k	139	ï	172	¼	204	╠	236	∞
12	C	014	FF (NP form feed, new page)	44	2C	054	,	76	4C	114	L	108	6C	154	l	140	î	173	¡	205	═	237	φ
13	D	015	CR (carriage return)	45	2D	055	-	77	4D	115	M	109	6D	155	m	141	ì	174	«	206	╬	238	ε
14	E	016	SO (shift out)	46	2E	056	.	78	4E	116	N	110	6E	156	n	142	Ä	175	»	207	╧	239	∩
15	F	017	SI (shift in)	47	2F	057	/	79	4F	117	O	111	6F	157	o	143	Å	176	░	208	╨	240	≡
16	10	020	DLE (data link escape)	48	30	060	0	80	50	120	P	112	70	160	p	144	É	177	▒	209	╤	241	±
17	11	021	DC1 (device control 1)	49	31	061	1	81	51	121	Q	113	71	161	q	145	æ	178	▓	210	╥	242	≥
18	12	022	DC2 (device control 2)	50	32	062	2	82	52	122	R	114	72	162	r	146	Æ	179	│	211	╙	243	≤
19	13	023	DC3 (device control 3)	51	33	063	3	83	53	123	S	115	73	163	s	147	ô	180	┤	212	╘	244	⌠
20	14	024	DC4 (device control 4)	52	34	064	4	84	54	124	T	116	74	164	t	148	ö	181	╡	213	╒	245	⌡
21	15	025	NAK (negative acknowledge)	53	35	065	5	85	55	125	U	117	75	165	u	149	ò	182	╢	214	╓	246	÷
22	16	026	SYN (synchronous idle)	54	36	066	6	86	56	126	V	118	76	166	v	150	û	183	╖	215	╫	247	≈
23	17	027	ETB (end of trans. block)	55	37	067	7	87	57	127	W	119	77	167	w	151	ù	184	╕	216	╪	248	°
24	18	030	CAN (cancel)	56	38	070	8	88	58	130	X	120	78	170	x	152		185	╣	217	┘	249	·
25	19	031	EM (end of medium)	57	39	071	9	89	59	131	Y	121	79	171	y	153	Ö	186	║	218	┌	250	·
26	1A	032	SUB (substitute)	58	3A	072	:	90	5A	132	Z	122	7A	172	z	154	Ü	187	╗	219	█	251	√
27	1B	033	ESC (escape)	59	3B	073	;	91	5B	133	[	123	7B	173	{	156	£	188	╝	220	▄	252	
28	1C	034	FS (file separator)	60	3C	074	<	92	5C	134	\	124	7C	174	|	157	¥	189	╜	221	▌	253	²
29	1D	035	GS (group separator)	61	3D	075	=	93	5D	135	]	125	7D	175	}	158		190	╛	222	▐	254	■
30	1E	036	RS (record separator)	62	3E	076	>	94	5E	136	^	126	7E	176	~	159	ƒ	191	┐	223	▀	255	
31	1F	037	US (unit separator)	63	3F	077	?	95	5F	137	_	127	7F	177	DEL	160	á	192	└	224	α		

Reference: https://en.wikipedia.org/wiki/ASCII ; http://zairo.tistory.com/entry/Webhackingkr-Chellange-Level-26

Constants

Named Constant

- Avogadro’s number (PI = 3.141592) and the speed of light are scientific constants; that is, to the degree of precision to which they have been measured and/or calculated, they do not vary.
- C++ supports named constants. Constants are declared like variables with the addition of the *const* keyword:

```
const double PI = 3.14159;
```

- Once declared and initialized, a constant can be used like a variable in all but one way—a constant may not be reassigned. It is illegal for a constant to appear on the left side of the assignment operator (=) outside its declaration statement.

```
PI = 2.5; (compile error)
```

- Since it is illegal to assign a constant outside of its declaration statement, all constants must initialized where they are declared.
- Generally express constant names in all capital letters; in this way, within the source code a human reader can distinguish a constant quickly from a variable.

Variables

Named Constant

- Named constant example

```
#include <iostream>

int main() {
   const double pi = 3.14159;
   double temp = 0;

   temp = pi;
}
```

Standard Input & Output

Output Stream

- *cout << “Please enter two integer values: “;*
  - This statement prompts the user to enter some information.
  - This statement is our usual print statement.

Standard Input & Output

Input Stream

- *cin >> value1;*
  - This statement causes the program's execution to stop until the user types single numbers on the key- board and then presses enter.
  - Once the user presses the enter key, the value entered is assigned to the variable.
- *cin >> value1 >> value2;*
  - This statement causes the program's execution to stop until the user types two numbers on the key- board and then presses enter.
  - The first number entered will be assigned to value1, and the second number entered will be assigned to value2.
  - The user may choose to type one number, press enter, type the second num- ber, and press enter again.
  - Instead, the user may enter both numbers separated by one of more spaces and then press enter only once.
  - The program will not proceed until the user enters two numbers.

Most Famous C++ Software

```
#include <iostream>

using namespace std;

int main()
{
   cout << “Hello World”;
}
```

Input & Output Stream Example

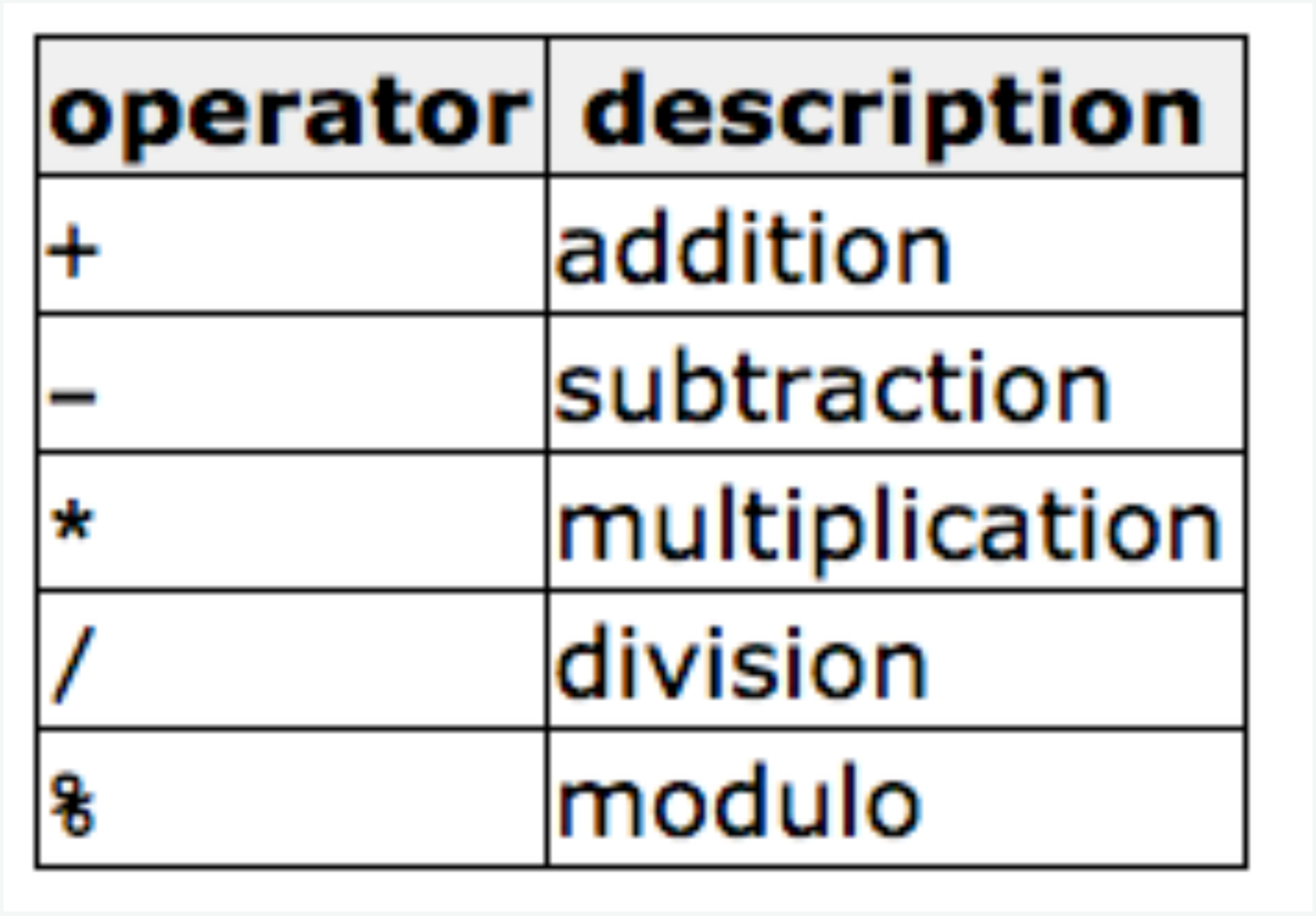
```
#include <iostream>
using namespace std;
int main() {
   int value1, value2, sum;
   cout << "Please enter two integer values: ";
   cin >> value1 >> value2;
   sum = value1 + value2;
   cout << value1 << " + " << value2 << " = " << sum << '\n';
}
```

```
#include <iostream>
int main() {
   int value1, value2, sum;
   std::cout << "Please enter two integer values: ";
   std::cin >> value1 >> value2;
   sum = value1 + value2;
   std::cout << value1 << " + " << value2 << " = " << sum << '\n';
}
```

Operators

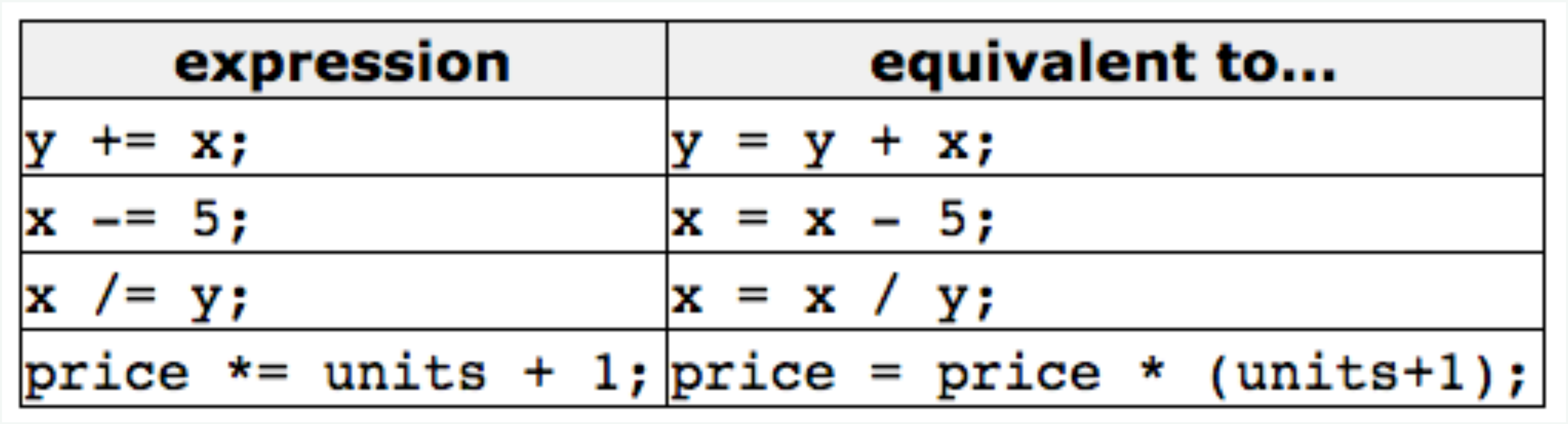
Arithmetic Operators

- Arithmetic Operators



operator	description
+	addition
-	subtraction
*	multiplication
/	division
%	modulo

- Compound Assignments



expression	equivalent to...
`y += x;`	`y = y + x;`
`x -= 5;`	`x = x - 5;`
`x /= y;`	`x = x / y;`
`price *= units + 1;`	`price = price * (units+1);`

Reference: http://www.cplusplus.com/doc/tutorial/operators/

Operators

Arithmetic Operators

- Increase Operator

++x; {or} x++;

x += 1;

x = x + 1;

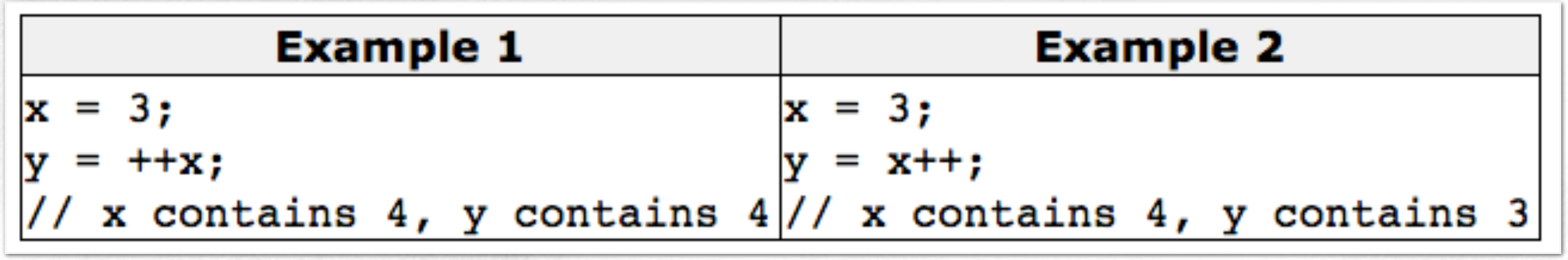
- Decrease Operator

--x; {or} x--;

x -= 1;

x = x - 1;

- Prefix or Suffix Operation (Compound Operation)



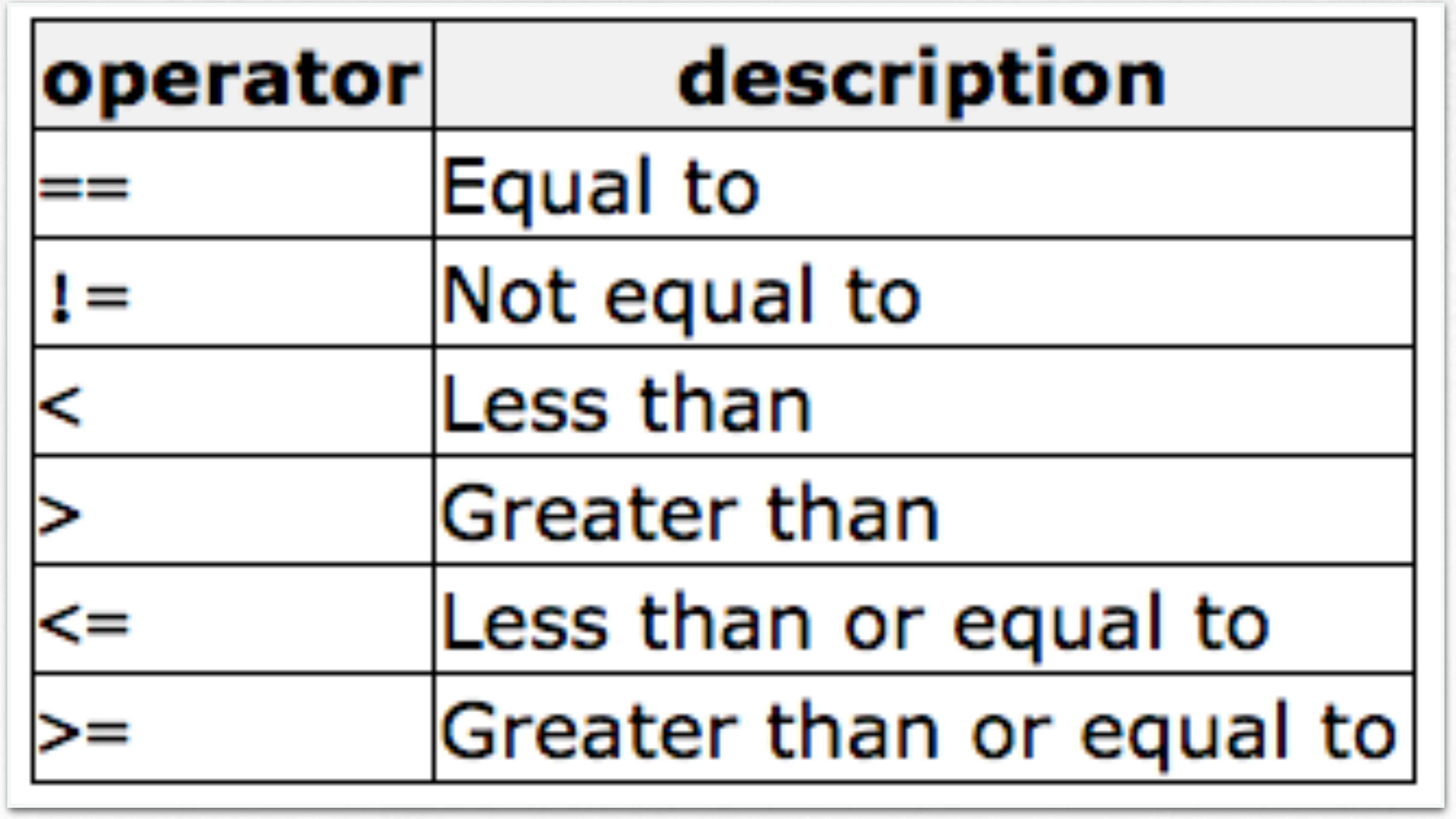
Example 1	Example 2

`x = 3;`
`y = ++x;`
`// x contains 4, y contains 4`
	`x = 3;`
	`y = x++;`
	`// x contains 4, y contains 3`

Reference: http://www.cplusplus.com/doc/tutorial/operators/

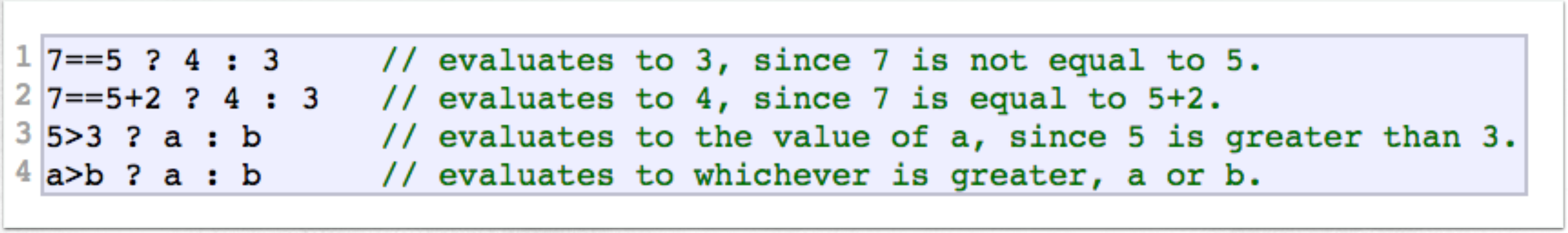
Relational Operators

- Relational Operators



operator	description
==	Equal to
!=	Not equal to
<	Less than
>	Greater than
<=	Less than or equal to
>=	Greater than or equal to

- Conditional Ternary Operator
  - The conditional operator evaluates an expression,
  - returning one value if that expression evaluates to true, and a different one if the expression evaluates as false. Its syntax is:



```
7==5 ? 4 : 3      // evaluates to 3, since 7 is not equal to 5.
7==5+2 ? 4 : 3    // evaluates to 4, since 7 is equal to 5+2.
5>3 ? a : b       // evaluates to the value of a, since 5 is greater than 3.
a>b ? a : b       // evaluates to whichever is greater, a or b.
```

Reference: http://www.cplusplus.com/doc/tutorial/operators/

Operators

Examples

```
// assignment operator
#include <iostream>
using namespace std;

int main ()
{
  int a, b;         // a:?,  b:?
  a = 10;           // a:10, b:?
  b = 4;            // a:10, b:4
  a = b;            // a:4,  b:4
  b = 7;            // a:4,  b:7

  cout << "a:";
  cout << a;
  cout << " b:";
  cout << b;
}
```

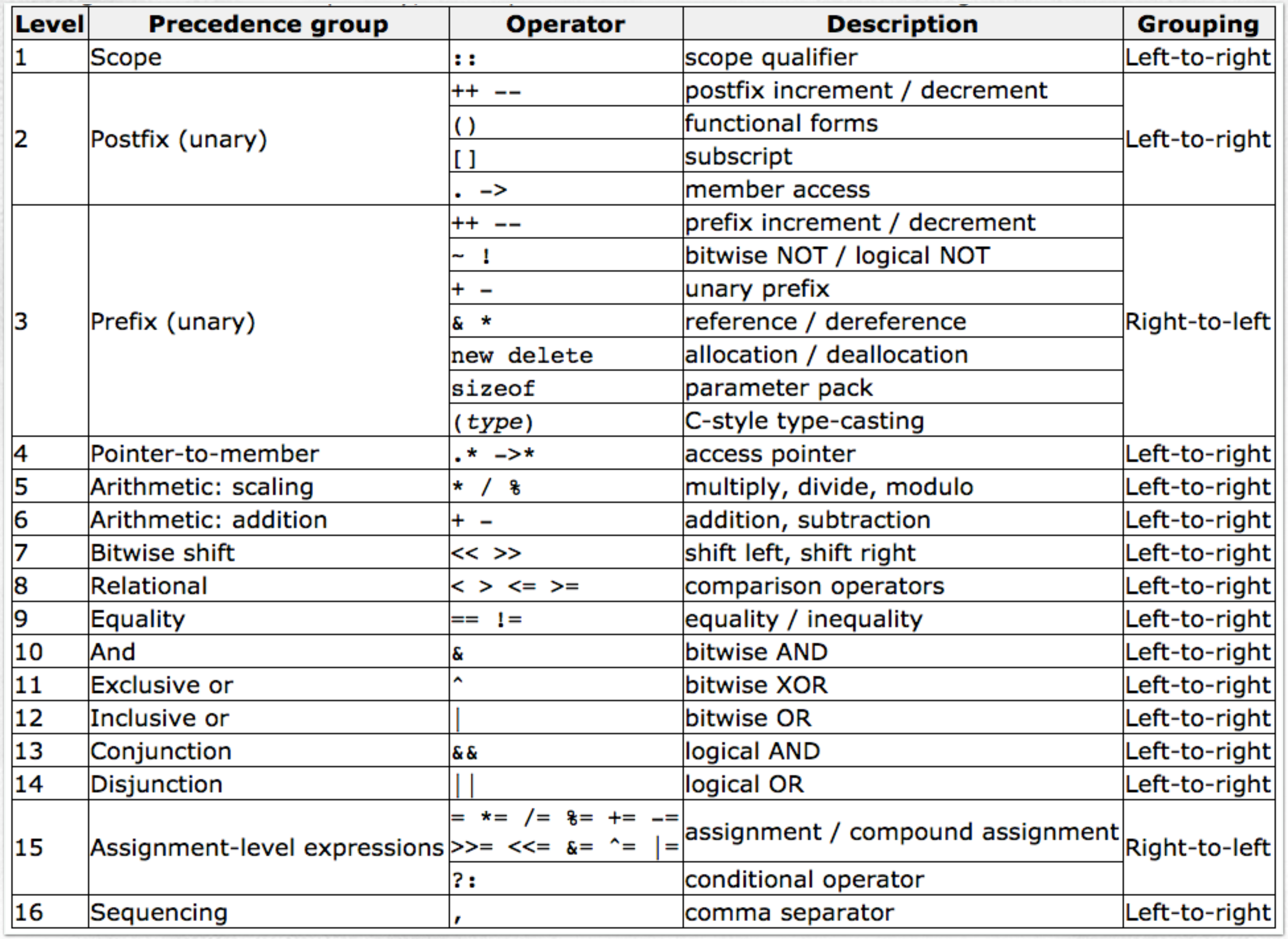
```
// compound assignment operators
#include <iostream>
using namespace std;

int main ()
{
  int a, b=3;
  a = b;
  a+=2;      // equivalent to a=a+2
  cout << a;
}
```

```
// conditional operator
#include <iostream>
using namespace std;

int main ()
{
  int a,b,c;

  a=2;
  b=7;
  c = (a>b) ? a : b;
  cout << c << '\n';
}
```

Reference: http://www.cplusplus.com/doc/tutorial/operators/

Operators

Precedence Priority



Level	Precedence group	Operator	Description	Grouping
1	Scope	`::`	scope qualifier	Left-to-right
2	Postfix (unary)	`++ --`	postfix increment / decrement	Left-to-right
		`()`	functional forms	
		`[]`	subscript	
		`. ->`	member access	
3	Prefix (unary)	`++ --`	prefix increment / decrement	Right-to-left
		`~ !`	bitwise NOT / logical NOT	
		`+ -`	unary prefix	
		`& *`	reference / dereference	
		`new delete`	allocation / deallocation	
		`sizeof`	parameter pack	
		`(type)`	C-style type-casting	
4	Pointer-to-member	`.* ->*`	access pointer	Left-to-right
5	Arithmetic: scaling	`* / %`	multiply, divide, modulo	Left-to-right
6	Arithmetic: addition	`+ -`	addition, subtraction	Left-to-right
7	Bitwise shift	`<< >>`	shift left, shift right	Left-to-right
8	Relational	`< > <= >=`	comparison operators	Left-to-right
9	Equality	`== !=`	equality / inequality	Left-to-right
10	And	`&`	bitwise AND	Left-to-right
11	Exclusive or	`^`	bitwise XOR	Left-to-right
12	Inclusive or	|	bitwise OR	Left-to-right
13	Conjunction	`&&`	logical AND	Left-to-right
14	Disjunction	||	logical OR	Left-to-right
15	Assignment-level expressions	`= *= /= %= += -=` `>>= <<= &= ^=` |=	assignment / compound assignment	Right-to-left
		`?:`	conditional operator	
16	Sequencing	`,`	comma separator	Left-to-right

Reference: http://www.cplusplus.com/doc/tutorial/operators/

Comments

Enhance Readability

- Ignored by compiler
- Only useful for programmers

```
//  single-line comment
```

```
/*
 *  multi-line comment
 */
```

Object Oriented Programming by C++

Sungwon Lee / Professor

Email: drsungwon@khu.ac.kr
Web: http://mobilelab.khu.ac.kr/