Unit 3

Constants, Expressions, and Variables

C++ Output (with 'cout')

Unit Objectives

- List the various C data types
- Identify what type a constant is
- Know how to write constants in the appropriate C++ syntax
- Know the C++ operators and their order of operations
- Write basic output statements of text and constants using cout

C/C++ Program Format/Structure

- Comments
  - Anywhere in the code
  - C-Style => "/*" and "*/"
  - C++ Style => "//"
- Compiler Directives
  - #includes tell compiler what other library functions you plan on using
  - 'using namespace std;' -- Just do it for now!
- main() function
  - Starting point of execution for the program
  - All code/statements in C must be inside a function
  - Statements execute one after the next and end with a semicolon (;)
  - Ends with a 'return 0;' statement
- Other functions
  - printName() is a function that can be "called"/"invoked" from main or any other function

```
/* Anything between slash-star and
star-slash is ignored even across
multiple lines of text or code */

// Anything after "//" is ignored on a line

// #includes allow access to library functions
#include <iostream>
#include <cmath>
using namespace std;

// Code is organized into units called functions
void printName()
{
  cout << "Tommy Trojan" << endl;
}

// Execution always starts at the main() function
int main()
{
  cout << "Hello: " << endl;
  printName();
  printName();
  return 0;
}
```

```
Hello:
Tommy Trojan
Tommy Trojan
```

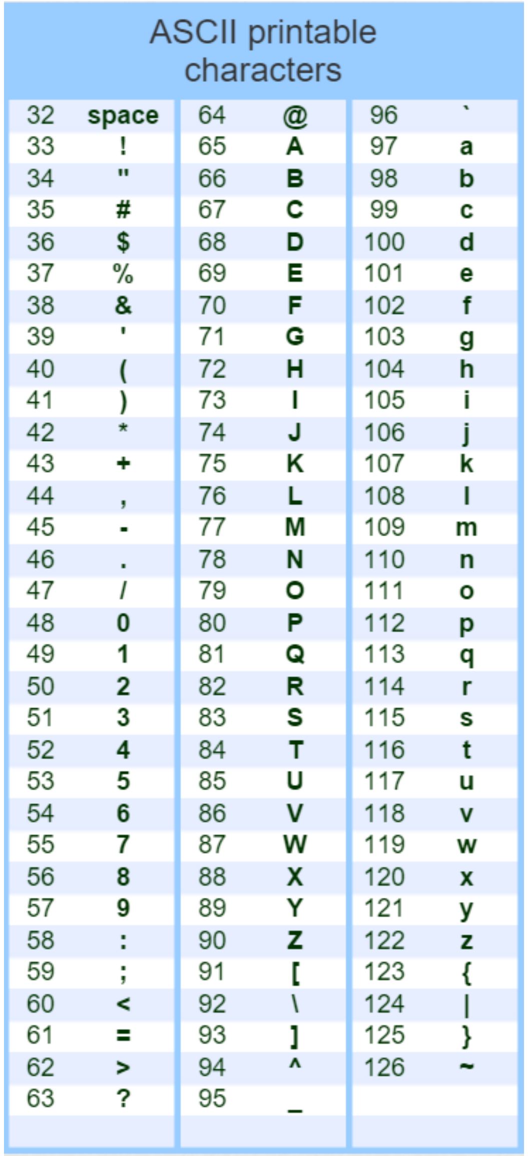
Review C Integer Data Types

- Integer Types (signed by default… unsigned with optional leading keyword)

C Type (Signed)	C Type (Unsigned)	Bytes	Bits	Signed Range	Unsigned Range
char	unsigned char	1	8	-128 to +127	0 to 255
short	unsigned short	2	16	-32768 to +32767	0 to 65535
int	unsigned int	4	32	-2 billion to +2 billion	0 to 4 billion
long	unsigned long	8	64	$-8*10^{18}$ to $+8*10^{18}$	0 to $16*10^{18}$

Review Text Rep.

- Text characters are usually represented with some kind of binary code (mapping of character to a binary number such as 'a' = 01100001 bin = 97 dec)
- ASCII = Traditionally an 8-bit code
  - How many combinations (i.e. characters)?
  - English only
- UNICODE = 16-bit code
  - How many combinations?
  - Most languages w/ an alphabet
- In C/C++ a single printing/text character must appear between single-quotes (')
  - Example: 'a', '!', 'Z'



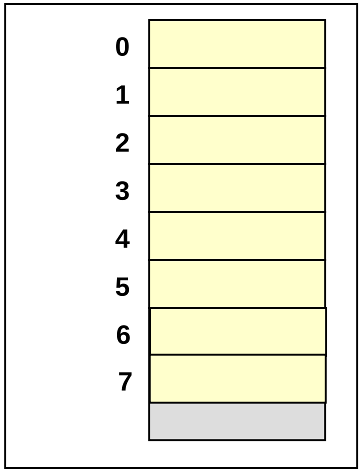
ASCII printable characters

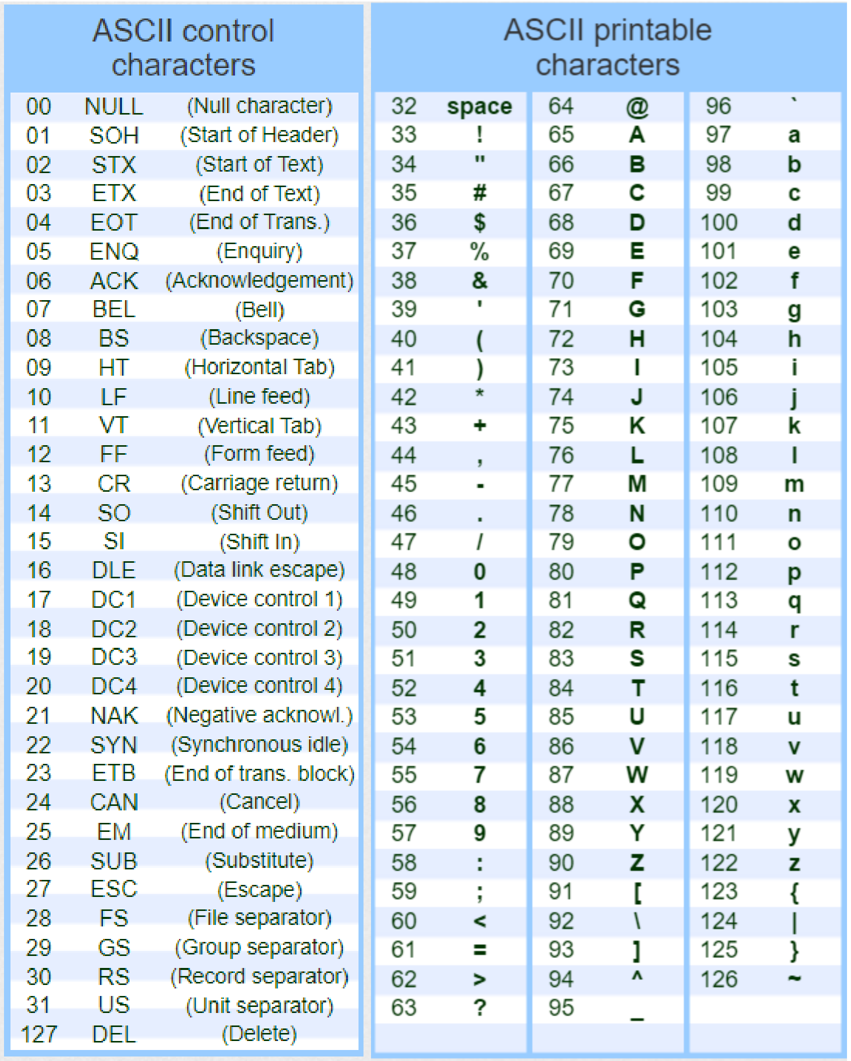
32	space	64	@	96	`
33	!	65	A	97	a
34	"	66	B	98	b
35	#	67	C	99	c
36	$	68	D	100	d
37	%	69	E	101	e
38	&	70	F	102	f
39	'	71	G	103	g
40	(	72	H	104	h
41	)	73	I	105	i
42	*	74	J	106	j
43	+	75	K	107	k
44	,	76	L	108	l
45	-	77	M	109	m
46	.	78	N	110	n
47	/	79	O	111	o
48	0	80	P	112	p
49	1	81	Q	113	q
50	2	82	R	114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

http://www.theasciicode.com.ar/

Review

- Show how "Hi!\n" would be stored in the memory below
  - Use decimal to represent each byte
  - Remember how we terminate a string



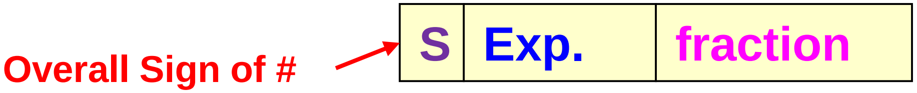


ASCII control characters		
00	NULL	(Null character)
01	SOH	(Start of Header)
02	STX	(Start of Text)
03	ETX	(End of Text)
04	EOT	(End of Trans.)
05	ENQ	(Enquiry)
06	ACK	(Acknowledgement)
07	BEL	(Bell)
08	BS	(Backspace)
09	HT	(Horizontal Tab)
10	LF	(Line feed)
11	VT	(Vertical Tab)
12	FF	(Form feed)
13	CR	(Carriage return)
14	SO	(Shift Out)
15	SI	(Shift In)
16	DLE	(Data link escape)
17	DC1	(Device control 1)
18	DC2	(Device control 2)
19	DC3	(Device control 3)
20	DC4	(Device control 4)
21	NAK	(Negative acknowl.)
22	SYN	(Synchronous idle)
23	ETB	(End of trans. block)
24	CAN	(Cancel)
25	EM	(End of medium)
26	SUB	(Substitute)
27	ESC	(Escape)
28	FS	(File separator)
29	GS	(Group separator)
30	RS	(Record separator)
31	US	(Unit separator)
127	DEL	(Delete)

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

What About Rational/Real #'s

- Previous binary system assumed binary point was fixed at the far right of the number, so we can't represent decimals
  - 10010. *(implied binary point)*
- Consider scientific notation:
  - Avogadro's Number: $+6.0247 * 10^{23}$
  - Planck's Constant: $+6.6254 * 10^{-27}$
- Can one representation scheme represent such a wide range?
  - Yes! Floating Point
  - Represents the sign, significant digits (fraction), exponent as separate bit fields
- Decimal: $\pm D.DDD * 10^{\pm exp}$
- Binary: $\pm b.bbbb * 2^{\pm exp}$



Overall Sign of #

S | Exp. | fraction

C Floating Point Types

- float and double types:

C Type	Bytes	Bits	Range
float	4	32	±7 significant digits * $10^{+/-38}$
double	8	64	±16 significant digits * $10^{+/-308}$

- Prefer double over float
  - Many compilers will upgrade floats to doubles anyhow
- Don't use floating-point if you don't need to
  - It suffers from rounding error
  - Some additional time overhead to perform arithmetic operations

C CONSTANTS & DATA TYPES

Constants

- Integer: `496, 10005, -234`
- Double: `12.0, -16., 0.23, -2.5E-1, 4e-2`
- Characters (char type): enclosed in single quotes
  - Printing characters: `'a', '5', 'B', '!'`
  - Non-printing special characters use "escape" sequence (i.e. preceded by a \ ): `'\n'` (newline/enter), `'\t'` (tab) , `'\\'` (slash), `'\''` (apostrophe)
- C-Strings
  - 0 or more characters between double quotes

    `"hi1\n", "12345", "b", "\tAns. is %d"`
  - Ends with a `'\0'`=NULL character added as the last byte/character to allow code to delimit the end of the string
- Boolean (C++ only): `true, false`
  - Physical representation: 0 = false, (Non-zero) = true

0	104	‘h’
1	105	‘i’
2	49	‘1’
3	10	‘\n’
4	00	Null
5	17	
6	59	
7	c3	
	...	

String Example

(Memory Layout)

You're Just My Type

- Indicate which constants are matched with the correct type.

Constant	Type	Right / Wrong
4.0	int	
5	int	
'a'	string	
"abc"	string	
5.	double	
5	char	
"5.0"	double	
'5'	int	

Solutions are provided at the end of the slide packet.

EXPRESSIONS

Arithmetic Operators

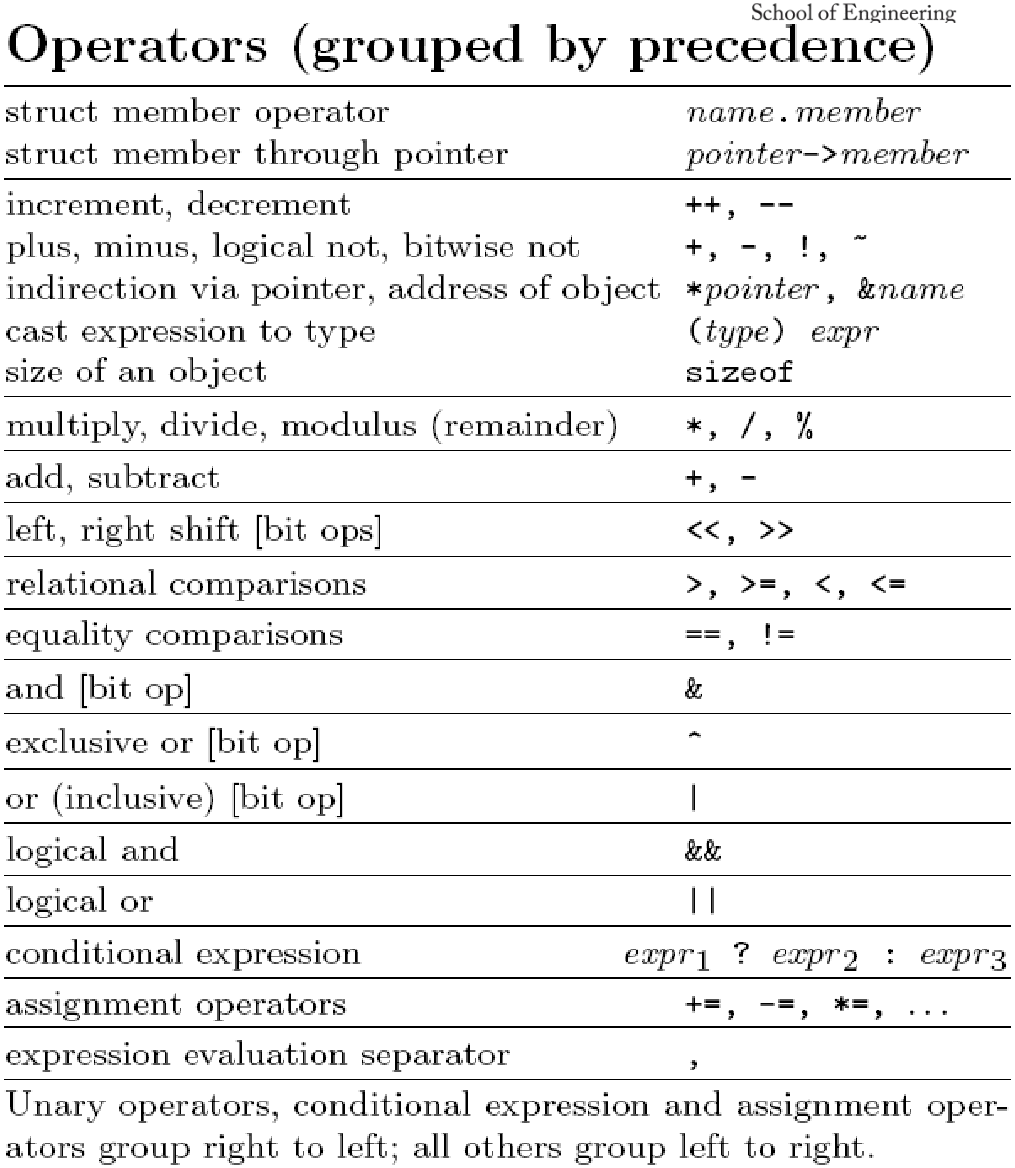
- Addition, subtraction, multiplication work as expected for both integer and floating point types
- Modulus is only defined for integers

Operator	Operation
+	Addition
-	Subtraction
*	Multiplication
/	Division (Integer vs. Double division)
%	Modulus (remainder) [for integers only]

```
10 % 3 = __
17 % 10 = __
```

Precedence

- Order of operations/ evaluation of an expression
- Top Priority = highest (done first)
- Notice operations with the same level or precedence usually are evaluated left to right (explained at bottom)
- Evaluate:
  - `2*-4-3+5%2;`
- Tips:
  - Use parenthesis to add clarity
  - Add a space between literals `(2 * -4) - 3 + (5 % 2)`



Operators (grouped by precedence)

struct member operator	*name*.*member*
struct member through pointer	*pointer*->*member*
increment, decrement	++, --
plus, minus, logical not, bitwise not	+, -, !, ~
indirection via pointer, address of object	**pointer*, &*name*
cast expression to type	(*type*) *expr*
size of an object	sizeof
multiply, divide, modulus (remainder)	*, /, %
add, subtract	+, -
left, right shift [bit ops]	<<, >>
relational comparisons	>, >=, <, <=
equality comparisons	==, !=
and [bit op]	&
exclusive or [bit op]	^
or (inclusive) [bit op]	|
logical and	&&
logical or	||
conditional expression	$expr_1$? $expr_2$: $expr_3$
assignment operators	+=, -=, *=, ...
expression evaluation separator	,

Unary operators, conditional expression and assignment operators group right to left; all others group left to right.

Division

- Computers perform division differently based on the type of values used as inputs
- **Integer** Division:
  - When dividing two integral values, the result will also be an integer (any remainder/fraction will be dropped)
  - 10 / 4 = 2 52 / 10 = 5 6 / 7 = 0
- **Floating-point** (Double) & Mixed Division
  - 10.0 / 4.0 = 2.5 52.0 / 10 = 5.2 6 / 7.0 = 0.8571
  - Note: If one input is a double, the other will be promoted temporarily to compute the result as a double

Exercise Review

- Evaluate the following:

```
25 / 3
20 - 12 / 4 * 2
3 - 15 % 7
18.0 / 4
28 - 5 / 2.0
```

Exercises from: D.S. Malik, C++ Programming, 5th Ed., Ch. 2, Q6.

Using 'cout'…

SIMPLE C++ OUTPUT

Output From Your Program

- To see the output in C++ we need to explicitly tell the computer to output the value using 'cout'
  - So what happens to the result of 12*3 on the first line?
- Note: 'endl' stands for **end-line** and causes the cursor to move to the next line of the screen

Performing computation is like having a thought. No output is generated unless you explicitly write it down.

To output a result to the screen in C++ (i.e. "write it down") we use the 'cout' command

```
// iostream allows access to 'cout'
#include <iostream>
using namespace std;

// Execution always starts at the main() function
int main()
{
  12 * 3;                 // No result printed

  cout << 12 * 3 << endl; // 36 printed

  return 0;
}
```

Printing Different Values & Types

- 'cout' requires appropriate use of separators between consecutive values or different types of values
- 'cout' does not add spaces between consecutive values; you must do so explicitly
  - Since text strings are a different value we must separate it with the '<<' operator
- Generally good practice to give some descriptive text with your numeric output
  - Note: You may divide up output over multiple 'cout' statements. Unless an 'endl' or '\n' is used, the next 'cout' statement will resume where the last one left off

```
// iostream allows access to 'cout'
#include <iostream>
using namespace std;


// Execution always starts at the main() function
int main()
{
  cout << 345  754 << endl; // Bad
  cout << 345 << 754 << endl; // Better, but no spaces
  cout << 345 << " " << 754 << endl; // Best
  return 0;
}
```

```
// iostream allows access to 'cout'
#include <iostream>
using namespace std;

// Execution always starts at the main() function
int main()
{
  cout << "3 dozen is " << 3*12 << " items." << endl;

  cout << "There are " << 60*24*365 << " minutes";
  cout << " in a year." << endl;
  return 0;
}
```

Output:

```
3 dozen is 36 items.
There are 525600 minutes in a year.
```

SOLUTIONS

You're Just My Type

- Indicate which constants are matched with the correct type.

Constant	Type	Right / Wrong
4.0	int	double (.0)
5	int	int
'a'	string	char
"abc"	string	C-string
5.	double	float/double (. = non-integer)
5	char	Int…but if you store 5 in a char variable it'd be okay (char = some number that fits in 8-bits/1-byte
"5.0"	double	C-string
'5'	int	char

Exercise Review

- Evaluate the following:
  - `25 / 3 = 8`
  - `20 - 12 / 4 * 2 = 14`
  - `3 - 15 % 7 = 2`
  - `18.0 / 4 = 2.5`
  - `28 - 5 / 2.0 = 25.5`

Exercises from: D.S. Malik, C++ Programming, 5th Ed., Ch. 2, Q6.