CISC 1600/1610 Computer Science I

Programming in C++

Professor Daniel Leeds
dleeds@fordham.edu
JMH 328A

Introduction to programming with C++

Learn

- Fundamental programming concepts
- Key techniques
- Basic C++ facilities

By the end of the course, you will be able to:

- Write small C++ programs
- Read much larger programs
- Learn the basics of many other languages
- Proceed to advanced C++ courses

2

Requirements

- Lectures and lab sessions
- Labs assignments – roughly 5 across semester
- Final project
- Exams – 1 midterm, 1 final

- Academic integrity – discuss assignments with your classmates, but DO NOT copy assignments

3

How to succeed in class

Ask questions

- In class
- In office hours, tutor room
- Study together and discuss assignments with each other (without plagiarizing!)

Textbook

- Read and re-read the material
- Complete practice problems

Start coding and studying early

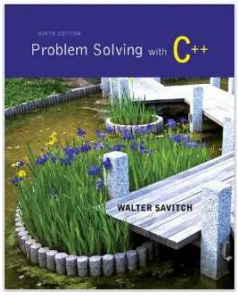
4

Course textbook

Problem Solving With C++

Ninth Edition

Walter Savitch



5

Course website

http://storm.cis.fordham.edu/leeds/cisc1600

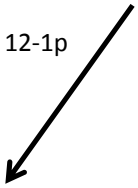
Go online for

- Lecture slides
- Assignments
- Course materials/handouts
- Announcements

6

Instructor

Prof. Daniel Leeds
dleeds@fordham.edu
Office hours: Tues 2-3p, Wed 12-1p
Office: JMH 328A



7

A program provides a computer with a set of simple instructions to achieve a goal

8

Programs are everywhere

On your computer:

- Web browser
  - Request and display information from distant sites
- Word processor
  - Record text, change appearance, save to disk
- Music player
  - Organize mp3's, select time in song, play, stop

9

Programs are everywhere

In the dining hall:

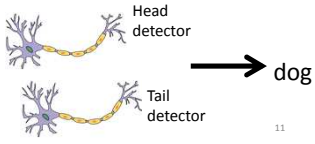
- Cashier
  - Compute price of food purchase, charge payment to account, (if pay cash: compute change)
- HVAC
  - Monitor temperature, adjust A/C or heating
- Electronic signs
  - Display menus and prices, load and display university news

10

Programs are everywhere

In humans:

- Sports
  - When to run, where to run; when to pass, who to pass to; when to shoot
- The brain
  - Neurons working together to combine information about an image to recognize a dog or a car



11

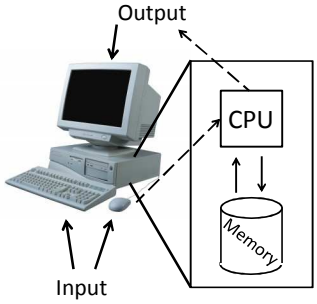
Computer system structure

Central processing unit (CPU) – performs all the instructions

Memory – stores data and instructions for CPU

Input – collects information from the world

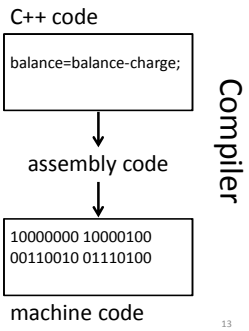
Output – provides information to the world



12

C++ – high-level language

- High-level language
  - Uses words to describe instructions
  - More intuitive to people
  - Computer-independent
- Machine-language
  - Uses binary to describe instructions
  - Less intuitive to people
  - Computer-dependent



13

Why C++?

- Popular modern programming language
- In use since 1980's
- Similar structure to many/most other popular languages (Java, C#, Perl, Python)

14

Why C++?

Some programming history:

- C++ developed as improvement on C
- C developed as improvement on B
- B developed as improvement on …
- BCPL – Basic Computer Programming Language
- Various languages before BCPL – ADA, COBOL, FORTRAN

15

Course outline

- Programming basics, input/output, arithmetic
- Conditional statements
- Loops
- Modularity – functions
- Complex data – arrays, vectors strings, and classes

Throughout the semester:

- Proper programming style

16

Programming basics

- Program structure and components
- Output text
- Variables
- Input information
- Perform arithmetic
- Type safety

17

Our first program: "Hello world!"

```
// include library of standard input and output commands
#include <iostream>
using namespace std;

int main()
{ // Begin main function
    cout << "Hello world!\n";  // output "Hello world!"

    return 0;                 /* indicate successful
                                 program completion */
} // End main function
```

```
> ./myProgram
Hello world!
>
```

18

The components of "Hello world!"

- Comments `//`, `/*  */`
- `main` function
- Preprocessor directives `#include`

Using comments

```
// include library of standard input and output commands
#include <iostream>
using namespace std;

int main()
{ // Begin main function
    cout << "Hello world!\n";  // output "Hello world!"

    return 0;                 /* indicate successful
                                 program completion */
} // End main function
```

- Explain programs to other programmers
- Ignored by compiler
- Syntax:

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


Preprocessor directives

```
// include library of standard input and output commands
#include <iostream>
using namespace std;

int main()
{ // Begin main function
    cout << "Hello world!\n";  // output "Hello world!"

    return 0;                 /* indicate successful
                                 program completion */
} // End main function
```

- Lines beginning with #
- Executed before compiling the program

`main` function

```
// include library of standard input and output commands
#include <iostream>
using namespace std;

int main()
{ // Begin main function
    cout << "Hello world!\n";  // output "Hello world!"

    return 0;                 /* indicate successful
                                 program completion */
} // End main function
```

Every C++ program has the function `int main()`

- `main` contains the instructions to be executed by the program
- The instructions included in the "body" of main are placed between curly braces `{ }`

Statements

```
// include library of standard input and output commands
#include <iostream>
using namespace std;

int main()
{ // Begin main function
    cout << "Hello world!\n";  // output "Hello world!"

    return 0;                 /* indicate successful
                                 program completion */
} // End main function
```

- Instructions to be performed when the program is run
- Each statement is completed with a `;`

Using "white spaces"

```
// include library of standard input and output commands
#include <iostream>
using namespace std;

int main()
{ // Begin main function
    cout << "Hello world!\n";  // output "Hello world!"

    return 0;                 /* indicate successful
                                 program completion */
} // End main function
```

- "White spaces" are blank lines, space characters, and tabs
- White spaces are ignored by the compiler
- Use indentation to group pieces of code together

Output command

```
cout << "Hello world!\n";
```

- `cout << "text";` outputs the specified text to the screen
- `cout` is the output stream object
- The text is delimited by double-quotes " "
  - **Only** use simple quotes (") not curly quotes (“”)
- << is the “stream insertion operator” directing the text into `cout`

Terminology:

A “character” is any single letter or symbol. E.g.: 'b', '?', '&'

A collection of characters is called a “string.” E.g.: "Hello world", "afe094n", "C++ is fun! "

25

Output command, part 2

```
cout << "Hello world!\n";
```

```
> ./myProgram
Hello world!
>
```

- Escape character: backslash \
- Escape sequence: backslash followed by another character
  - New line: \n
  - Tab: \t

```
cout << "Hello\n world!\n";
```

```
> ./myProgram
```

26

Output command, part 3

```
cout << "Hello world!\n";
```

```
> ./myProgram
Hello world!
>
```

- We can place multiple stream insertion operators in a sequence.

```
cout << "Hello" << " world.";
cout << "How are \nyou today?";
```

```
> ./myProgram
```

27

User input: “Hello ____!”

```
// include library of standard input and output commands
#include <iostream>
using namespace std;

int main()
{ // Begin main function
    string name;            // create variable called name
    cout << "What is your name?";
    cin >> name;            // get name from user
    cout << "Hello ";       // output "Hello "
    cout << name << "!\n"; // output "<name>!"
    return 0;              // end program
} // End main function
```

```
> ./myProgram
What is your name? Alice
Hello Alice!
>
```

28

Variables

Variables store information

`char`	single character ('a', 'Q')
`int`	integers (-4, 82)
`bool`	logic (true or false)
`float`	real numbers (1.3, -0.45)
`vector`	sequence of values ({16,5}, {-2.3,3.4,-0.4})
`string`	text ("Hello", "reload")

29

Variable declaration

```
// include library of standard input and output commands
#include <iostream>
using namespace std;

int main()
{ // Begin main function
    string name;            // create variable called name
    cout << "What is your name?";
    cin >> name;            // get name from user
    cout << "Hello ";       // output "Hello "
    cout << name << "!\n"; // output "<name>!"
    return 0;              // end program
} // End main function
```

Updated for September 11, 2014

“Declare” new variable by writing type followed by variable name.

More examples:

```
int age, weight; // multiple declarations
```

30

Variable declaration and initialization

- All variables must be declared before they are used

```
int cost;   // declare variable
```

- Variables are initialized with the first assignment statement

```
cost = 25;  // initialize variable
```

- Declaration and initialization can be performed in one line

```
int weight = 140;
```

31

"Constant" variables

- The value of a variable ordinarily can be changed throughout the program
- `const` fixes variable value after initialization

```
const float healthyTemp = 98.6;
```

32

Variable names

- A variable name is any valid identifier that is not a keyword
  - Starts with a letter, contains letters, digits, and underscores (_) only
  - Cannot begin with a digit
  - Case sensitive: `username≠userName≠UserName`

33

Variable names, part 2

Choose meaningful names

- Avoid acronyms
- Avoid lengthy names

- Good:
  `age`, `size`, `address`, `count`, `sumData`
  `x`, `y`, `i` – single letters as counting variables
- Bad:
  `rbi`, `lda`, `xZ25`, `neuron_response_magnitude`

34

Keywords

Also known as: "Reserved names"

- Examples
  - `cout`, `return`, `string`, `int`
- Must be used as they are defined in the programming language
- Cannot be used as variable names

35

Variable assignment

- Typically, variables are assigned values with the = operator

```
string weather;
weather = "sunny";
cout << "The weather today is ";
cout << weather << endl;
```

- The variable to be changed is always to the left of the = operator
- The value assigned from the right of the = operator
  - Constants: `weight = 140;`
  - Variables: `ageErica = ageJen;`
  - Expressions: `balance = balance – cost;`

36

Input command

```
// include library of standard input and output commands
#include <iostream>
using namespace std;

int main()
{ // Begin main function
    string name;            // create variable called name
    cout << "What is your name?";
    cin >> name;            // get name from user
    cout << "Hello ";       // output "Hello "
    cout << name << "!\n";  // output "<name>!"
    return 0;               // end program
} // End main function
```

- `cin >> varName;` receives input from keyboard saves into the `varName`

37

Arithmetic in C++

Operators

- Addition: `5 + 2` evaluates to `7`
- Subtraction: `5 - 2` evaluates to `3`
- Multiplication: `5 * 2` evaluates to `10`
- Division: `4 / 2` evaluates to `2`
- Modulo: `5 % 2` evaluates to `1` (only integers)

38

Order of operations

- First: Parentheses
- Second: Multiplication, Division, Modulo
- Third: Add, Subtract

- Evaluate from Left to Right
- Evaluate inner-most parentheses before outer ones

```
int a = ( 4 * ( 5 + 2 ) - 4 ) / 4;
```

39

Assignment operators

```
int a = 6;
```

- Standard assignment: `a = 3;`
- Other assignments:
  - `a += 3;  // a = a + 3;`
  - `a -= 3;  // a = a - 3;`
  - `a *= 3;  // a = a * 3;`
  - `a /= 3;  // a = a / 3;`
  - `a %= 3;  // a = a % 3;`

40

Increment and decrement

```
int c = 12;
```

- Increment by 1: `c++` evaluates to `c + 1`

- Decrement by 1: `c--` evaluates to `c - 1`

41

What does this program do?

```
#include <iostream>
using namespace std;

int main()
{
    int dollars, coins;
    cout << "How many dollars do you have? ";
    cin >> dollars;
    coins = dollars*4;
    cout << "I will give you " << coins;
    cout << " coins.\n";
    return 0;
}
```

42

The binary representation

- `int age = 65;` assigns a binary code to memory: `00000000000000000000000001000001`
- `char grade = 'A';` assigns a binary code to memory: **`01000001`**
- Every variable value is a number in binary, C++ interprets the binary number based on the variable type

43

From numbers to symbols: the ASCII table

Dec	Hx	Oct	Char	Dec	Hx	Oct	Html	Chr	Dec	Hx	Oct	Html	Chr	Dec	Hx	Oct	Html	Chr
0	0	000	NUL (null)	32	20	040		Space	64	40	100	@	@	96	60	140	`	`
1	1	001	SOH (start of heading)	33	21	041	!	!	65	41	101	A	A	97	61	141	a	a
2	2	002	STX (start of text)	34	22	042	"	"	66	42	102	B	B	98	62	142	b	b
3	3	003	ETX (end of text)	35	23	043	#	#	67	43	103	C	C	99	63	143	c	c
4	4	004	EOT (end of transmission)	36	24	044	$	$	68	44	104	D	D	100	64	144	d	d
5	5	005	ENQ (enquiry)	37	25	045	%	%	69	45	105	E	E	101	65	145	e	e
6	6	006	ACK (acknowledge)	38	26	046	&	&	70	46	106	F	F	102	66	146	f	f
7	7	007	BEL (bell)	39	27	047	'	'	71	47	107	G	G	103	67	147	g	g
8	8	010	BS (backspace)	40	28	050	(	(	72	48	110	H	H	104	68	150	h	h
9	9	011	TAB (horizontal tab)	41	29	051	)	)	73	49	111	I	I	105	69	151	i	i
10	A	012	LF (NL line feed, new line)	42	2A	052	*	*	74	4A	112	J	J	106	6A	152	j	j
11	B	013	VT (vertical tab)	43	2B	053	+	+	75	4B	113	K	K	107	6B	153	k	k
12	C	014	FF (NP form feed, new page)	44	2C	054	,	,	76	4C	114	L	L	108	6C	154	l	l
13	D	015	CR (carriage return)	45	2D	055	-	-	77	4D	115	M	M	109	6D	155	m	m
14	E	016	SO (shift out)	46	2E	056	.	.	78	4E	116	N	N	110	6E	156	n	n
15	F	017	SI (shift in)	47	2F	057	/	/	79	4F	117	O	O	111	6F	157	o	o
16	10	020	DLE (data link escape)	48	30	060	0	0	80	50	120	P	P	112	70	160	p	p
17	11	021	DC1 (device control 1)	49	31	061	1	1	81	51	121	Q	Q	113	71	161	q	q
18	12	022	DC2 (device control 2)	50	32	062	2	2	82	52	122	R	R	114	72	162	r	r
19	13	023	DC3 (device control 3)	51	33	063	3	3	83	53	123	S	S	115	73	163	s	s
20	14	024	DC4 (device control 4)	52	34	064	4	4	84	54	124	T	T	116	74	164	t	t
21	15	025	NAK (negative acknowledge)	53	35	065	5	5	85	55	125	U	U	117	75	165	u	u
22	16	026	SYN (synchronous idle)	54	36	066	6	6	86	56	126	V	V	118	76	166	v	v
23	17	027	ETB (end of trans. block)	55	37	067	7	7	87	57	127	W	W	119	77	167	w	w
24	18	030	CAN (cancel)	56	38	070	8	8	88	58	130	X	X	120	78	170	x	x
25	19	031	EM (end of medium)	57	39	071	9	9	89	59	131	Y	Y	121	79	171	y	y
26	1A	032	SUB (substitute)	58	3A	072	:	:	90	5A	132	Z	Z	122	7A	172	z	z
27	1B	033	ESC (escape)	59	3B	073	;	;	91	5B	133	[	[	123	7B	173	{	{
28	1C	034	FS (file separator)	60	3C	074	<	<	92	5C	134	\	\	124	7C	174			|
29	1D	035	GS (group separator)	61	3D	075	=	=	93	5D	135	]	]	125	7D	175	}	}
30	1E	036	RS (record separator)	62	3E	076	>	>	94	5E	136	^	^	126	7E	176	~	~
31	1F	037	US (unit separator)	63	3F	077	?	?	95	5F	137	_	_	127	7F	177	&#127;	DEL

Source: www.LookupTables.com

44

Variable types, revisited

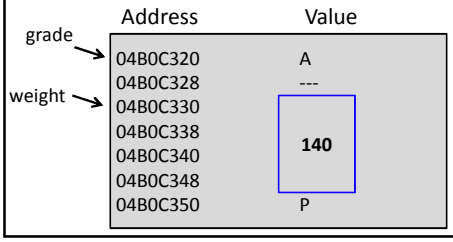
`char`	single character ('a', 'Q')	1 byte
`int`	integers (-4, 82)	4 bytes
`bool`	logic (true or false)	1 byte
`float`	real numbers (1.3, -0.45)	4 bytes
`vector`	sequence of values ({16,5}, {-2.3,3.4,-0.4})	? bytes
`string`	text ("Hello", "reload")	? bytes

- Each variable is represented by a certain number of 0s and 1s
- Each 0-or-1 is a bit
- 8 bits in a row is a byte

45

Variables – locations in memory

- Each variable indicates a location in memory
- Each location holds a value
- Value can change as program progresses
- Variable value exists before initialization



46

Assigning between types

```
int x = 5;
float y = -2.5;
float z = x * y;
int g = y - x;
```

47

Assigning between types

- `int` vs `float`
  - If compiler permits, floats will be rounded to nearest integer and ints will be expanded to a precision float
- `int` vs `char`
  - If compiler permits, char will be converted to integer ASCII code and int will be converted to corresponding ASCII character
- `int` vs `bool`
  - If compiler permits, bool will be converted to 0 (if false) or 1 (if true) and int will be converted to false (of 0) or 1 (if not 0)

```
int x = 5;
float y = -2.5;
float z = x * y;
int g = y - x;
```

48