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

- Attendance and participation
- Lectures and lab sessions
- Labs assignments – roughly 6-8 across semester
- Quizzes – each 15 minutes, 5 across semester
- Final project
- Exams – 1 midterm, 1 final

- Academic integrity – may discuss assignments with your classmates, but you MUST write all your code and all your answers yourself

3

How to succeed in class

Ask questions

- In class
- In office hours JMH 328A, tutor room JMH 301
- 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

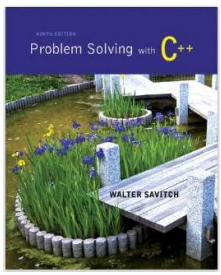
4

Course textbook

Problem Solving With C++

Ninth Edition

Walter Savitch



5

Course website

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

Go online for

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

6

Instructor

Prof. Daniel Leeds
dleeds@fordham.edu
Office hours: Tues 11a-12p, Thu 12:30-1:30p
Office: JMH 328A

7

"Two courses in one": CISC 1600 and CISC 1610

CISC 1600 and 1610 cover lecture and lab

- Lecture room: JMH 342
- Lab room: JMH 331

- We may work in lab during lecture time
- We may have lecture during lab time

Listen for announcements during class/online!

8

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

9

Programs are everywhere

On your computer:

- Web browser
  - Request and display information from distant sites
- Word processor
  - Record text, change appearance, save to disk

10

Programs are everywhere

In the dining hall:

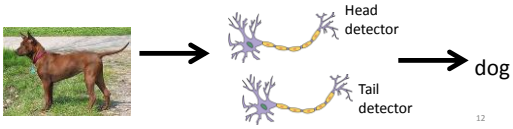
- Cashier
  - Compute price of food purchase, charge payment to account, (if pay cash: compute change)
- HVAC
  - Monitor temperature, adjust A/C or heating

11

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



12

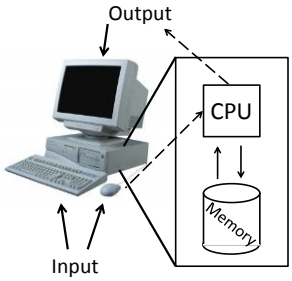
Computer system structure

Central processing unit (CPU) – performs all the instructions

Memory – stores data and instructions for CPU

Input – collects information from the world

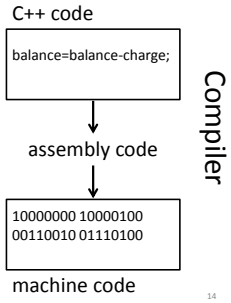
Output – provides information to the world



13

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



14

Why C++?

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

15

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

16

Course outline

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

Throughout the semester:

- Proper programming style

17

Programming basics

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

18

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!
>
```

19

The components of "Hello world!"

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

20

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 */
```

21

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

22

`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 { }

23

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 ;

24

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

25

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! "

26

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
```

27

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
```

28