

Basics of C++ Programming

Larry Caretto
Computer Science 106

Computing in Engineering and Science

February 2, 2006

California State University
Northridge

Outline

- Review last lecture
- Basic structure of a C++ program
 - Comments and statements
 - White space not part of program
 - Variables are case sensitive
 - Collection of functions with execution starting in a function called main
- Screen output and keyboard input

California State University
Northridge

2

Review

- <http://www.csun.edu/~lcaretto/comp106>
 - Structure of computer
- Programming in higher level language like C++
 - Compile, link, execute, debug
- Integrated development environment
- Design, code, test
 - Program correctness essential

California State University
Northridge

3

Basic C++ Programs

- A C++ program is a collection of functions
 - Execution starts in the main function
 - Functions may be in more than one file
 - Have comments and code
 - Must include libraries
- Initial assignments will have only one function, called main

California State University
Northridge

4

A Note on Syntax

- When we describe programs we distinguish between items that are written as stated and descriptions of other items that are required or optional
- Items written as stated will be in a non-proportional font like `this one`
- Descriptions of other items will be in bold italic in brackets `<like this>`
- Example => Name: `<your name>`

California State University
Northridge

5

Basic C++ Program Structure

```
#include <iostream>
using namespace std;
int main()
{
    <your program statements>
    return EXIT_SUCCESS;
}
```

- May have other #include statements

California State University
Northridge

6

Purpose of Basic Structure

- The `#include <iostream>` statement invokes the basic input/output library
- The using namespace `std;` statement defines standard names so you do not have to declare them in your code
- The `int main()` declares the start of the main function and declares it to be of type `int` (more on types next week)

Purpose of Basic Structure II

- The opening and closing braces, `{` and `}`, mark the start and end of the main function
- `EXIT_SUCCESS` is a symbolic constant defined to have a value of zero
- The return statement passes a value to the calling function – in this case the operating system

C++ Statements

- C++ statements can have multiple lines
- The end of a C++ statement is marked with a semicolon (`;`)
- White spaces are ignored in C++
- `w = x + y;` is the same as `w=x+y;`
- C++ statements use program variables that represent computer memory locations

Program Variables

- Look like mathematical variables but they not
- Program variables describe operations on memory locations, e. g.,
 - Take the value from memory location 121
 - Add to this the value from location 237
 - Store the result in location 142
- C++ writes this as `w = x + y;`

Programming vs. Math

- Consider the following operations
 - `y = 2;`
 - `x = y;`
 - `y = 3;`
- What is `x` at the end of this sequence?
 - In mathematics `x` would equal `y`, which is 3
 - In programming, we set the value of `x` to the value of `y` when `y` was 2; thus `x` is 2

Programming vs. Math II

- What the statements on the previous slide mean
 - `y = 2;` // assign a value 2 to the variable named `y` representing a memory location
 - `x = y;` // find the value of the variable named `y` and assign that value to `x`
 - `y = 3;` // assign a value 3 to `y`
- Once `x` is set to the value of `y` (2), it is not changed (in this sequence)

More on Variables

- Rules for variable names
 - Can have up to 31 characters (actually more, but only first 31 are recognized)
 - Allowed characters are A-Z, a-z, 0-9 and _
 - Variable name cannot start with 0-9
 - Variables names are case sensitive
 - Answer is not the same as answer
- Variable names refer to memory locations whose values you can change

Guidelines for Variables

- Use meaningful variable names
 - innerRadius, outerRadius, rInner, rOuter, rIn, rOut, are good examples
 - Referring to an inner radius and an outer radius as x and y is not a good idea
 - In lecture we will often use single letter variables for simplicity
 - This is reasonable when we represent common physical abbreviations: KE = $mV^2/2$ could use variable m for mass, V for velocity and KE

Variables and Data Types

- Variables must be declared as having a certain data type before they are used
- Variables should be declared as close to the point of their first use as possible
- Variables are usually assigned a value when they are first declared
- We will cover data types in detail next week

Changing Variable Values

- Can change a value by input statements and assignment statements
- Use = as the assignment operator
 - $x = 2$; assigns the value 2 to x
 - $x = y$; assigns the value of y to x
 - $x = x + 2$; takes the existing value of x, adds 2 to it, and assigns this to x
 - What is x after $x = 1$; $y = 2$; $x = x + y$?

The value of x is 3

Input and Output

- We use the following commands and operators for input and output
 - Use the command cout for screen output
 - Use the command cin for keyboard input
 - Use the output (insertion) operator << between output items for screen and files
 - Use the input (extraction) operator >> between input items from keyboard and files

Output using cout

- `cout << "<string>";` writes the string between the quotation marks to the screen
- `cout << x;` writes the value of the variable, x, to the screen.
- Can have one or more output (<<) operators in a single cout command

Code	Screen output
<code>cout << "Name"</code>	Name
<code>int x = 2; cout << x;</code>	2
<code>cout << " x = " << x;</code>	x = 2

Input using cin

- Input prompt tells user what to input
- Enter several variables with a single cin command
 - Separate entries by a space and press <enter> (the Enter key) after last entry

Code	Actions
cout << "Enter x: "; cin >> x;	Type 2.3 <enter> for x = 2.3
cout << "Enter x, y, and z: "; cin >> x >> y >> z;	Type 1.4 -3.2 12.7 <enter> to set x = 1.4, y = -3.2 and z = 12.7

Output spacing

- cout does not provide any spacing between output or new lines
- E.g., x = 13.2; y = 12.6; cout << x << y; (or cout << x; cout << y;) would print 13.212.6
- You can put a string of blanks in your cout commands: cout << x << " " << y; would print 13.2 12.6

Escape sequences

- Special characters to control printing entered in strings
 - \n for new line (can also use endl)
 - \t for tab
 - \" for quotation mark
 - \\ for backslash
- E.g cout << x << "\t" << y << "\n";
- Or cout << x << "\t" << y << endl;

Equivalent Statements

```
cout << "\n radius = " << r <<
"\ndiameter = " << d <<
"\n area = " << a;
```

```
cout << "\n radius = " << r
<< "\ndiameter = " << d
<< "\n area = " << a;
```

```
cout << endl;
cout << " radius = " << r << endl;
cout << "diameter = " << d << endl;
cout << " area = " << a;
```

Equivalent Statements II

```
• What is the output from these statements?
double r = 1, PI = 3.14159265358979;
double d = 2 * r;
double a = PI * r * r;
cout << "\n radius = " << r
<< "\ndiameter = " << d
<< "\n area = " << a;
```

radius = 1 Default output is six
diameter = 2 significant figures
area = 3.14159 with no trailing zeros

Output Errors

- What is wrong with these statements?
 - cout << "Enter x: "; cin >> y;
 - cout << "x = " << y;
 - cout << "x divided by y is " << y / x;
 - cout << " x minus y is " << y - y;
- Note that the above statements are **not** syntax errors, but they are errors that could lead to very confusing results.

Output Quiz

- What is the output from these statements?

```
double x = 3.1, y = 2.2;  
cout << "x = " << y; x = 2.2  
cout << x << y; 3.12.2  
cout << x << "x = "; 3.1x =  
cout << "My name is << x";  
My name is << x  
cout << "My name is \n" << x;
```