

OUTPUT, VARIABLES AND ASSIGNMENT

Today

- Our first C++ program
- Output
- The software development cycle
- Variables
- Assignment and mathematical operators

Our first C++ program

“hello world”

- Typical first program in any language
- Output only (no input)

The source code

```
//-----  
//  hello.cpp  
//  
//  This program demonstrates output in C++  
//  
//  Simon Parsons  
//  2nd September 2008  
//-----  
  
#include <iostream>  
using namespace std;  
  
int main()  
{  
    cout << "This is my c++ world\n";  
    cout << "Hello from inside of it!\n";  
}
```

Line by line

- The lines that begin with `//` are *comments*.
- They describe what the program does, they don't do anything
 - The computer ignores them.
- The key part of the program is the `cout`.
- This tells the computer to print something on the screen.
- The “something” that is printed is the thing inside the `" "`.
 - That is the meaning of the `>>`

- `#include <iostream>` tells the compiler we will be doing some output (or input)
- It prepares the computer to handle `cout`.
- `int main()` marks the start of the program.
 - You can think of it as saying “here’s the main part of the program”.
- We read `{` as “begin” and `}` as “end”.
- So, how do we read the whole program?

Output

- *Methods*

`cout`

- *Arguments*

- Also called *parameters*
- Those things that follow `cout`
- `<<` followed by a *string*, i.e., text in double quotes (")
- Escape sequences: `\n`, `\t`

- Example

```
cout << "Are Macs better than PCs?\n";  
cout << "Are Macs better than PCs?" << endl;
```

Things to notice

- C++ is CASE sensitive
- Punctuation is really important!
- *Whitespace* doesn't matter for compilation
- *BUT* whitespace DOES matter for readability
 - and your grade!
- In general, file name is same as class name.
- For now, file name is the same as project name.

Let's try it: the software development cycle

1. Open up a *text editor* or an *IDE*
2. Type in the *source code* and save it as a *text file*
3. *Compile* the source code,
using the *g++* command or a menu option on the IDE
4. *Execute* the program, from the command line or from within the IDE


Data storage

- Think of the computer's memory as a bunch of boxes
- Inside each box, there is a number
- You give each box a name
⇒ defining a *variable*
- Example:

Program code:

```
int x;
```

Computer's memory:

x → 

Variables

- Variables have:
 - name
 - type
 - value
- Naming rules:
 - names may contain letters and/or numbers
 - but cannot begin with a number
 - names may also contain underscore (_)
 - can be of any length
 - cannot use C++ keywords (also called *identifiers*)
 - C++ is *case-sensitive!!*

Assignment

- = is the assignment operator
- Example:

Program code:

```
int x;  
// declaration  
x = 19;  
// assignment
```

or

```
int x = 19;
```

Computer's memory:

x → 19

Mathematical operators.

+	unary plus
−	unary minus
+	addition
−	subtraction
*	multiplication
/	division
%	modulo

Example:

```
int x, y;  
x = -5;  
y = x * 7;  
y = y + 3;  
x = x * -2;  
y = x / 19;
```

What are x and y equal to?

Modulo means “remainder after integer division”

Increment and decrement operators

- We are always increasing and decreasing values by one, so there are shortcuts.

- Increment: ++

`i++;`

is the same as:

`i = i + 1;`

- Decrement: --

`i--;`

is the same as:

`i = i - 1;`

Assignment operators.

- There are shorthand ways of doing other combinations of arithmetic and assignment.

+=

`i += 3;` is the same as: `i = i + 3;`

-=

`i -= 3;` is the same as: `i = i - 3;`

***=**

`i *= 3;` is the same as: `i = i * 3;`

- Also:

`/=`

`i /= 3 ;` is the same as: `i = i / 3 ;`

`%=`

`i %= 3 ;` is the same as: `i = i % 3 ;`

Summary

- This lecture covered writing a first C++ program.
- We also sat down, wrote an initial program and ran it.
- We later discussed data and variables
- With the idea of a variable under our belts, we could start to think about arithmetic and assignment.
- This makes it possible to write more interesting programs.