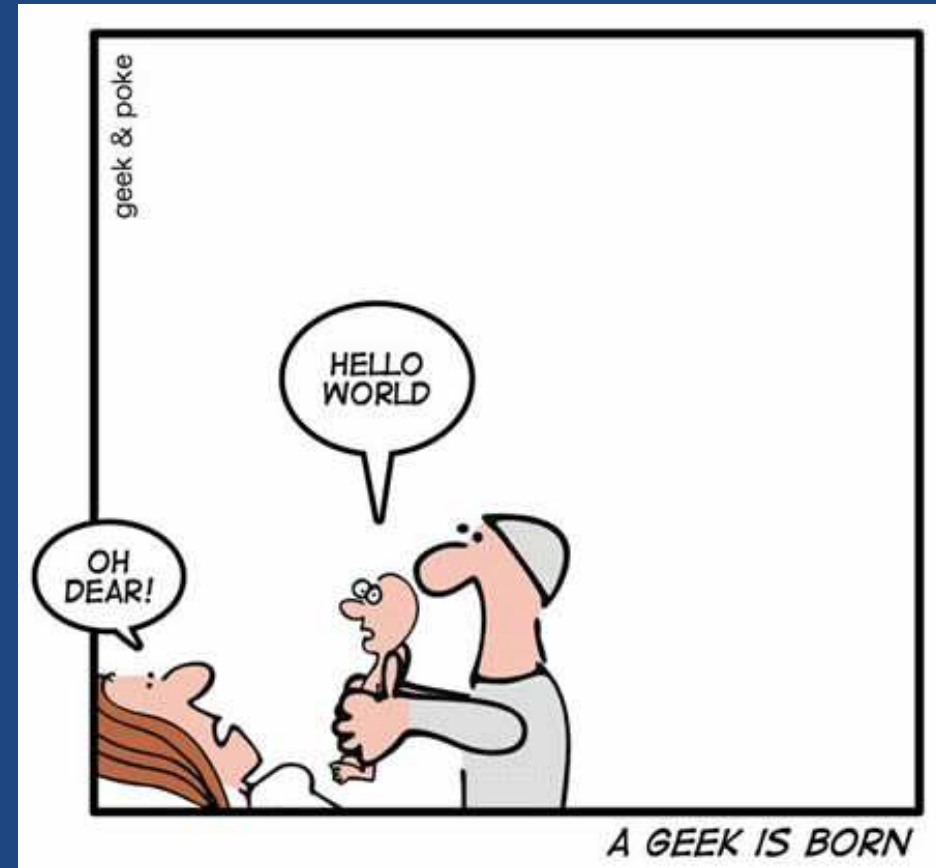


Introduction to C++

Chapter 1.3-1.4

Slidedeck #2
CMPT 130
© Brian Fraser



Topics

- 1) What does a **simple C++ program** look like?
- 2) How can we **output text** to the screen?
- 3) What kind of **errors** will we see?

Hello World!

A simple C++ program.

Simple C++ Program

```
// A simple C++ program.  
#include <iostream>  
using namespace std;  
  
int main () {  
    cout << "Hello world";  
    return 0;  
}
```

Output



Simple C++ Program

```
// A simple C++ program.
```

```
#include <iostream>
```

```
using namespace std;
```

```
int main () {
```

```
    cout << "Hello world";
```

```
    return 0;
```

```
}
```

Comments:

All text on a line after a `//` is a comment.

These are notes to the programmer;..

Simple C++ Program

```
// A simple C++ program.
```

```
#include <iostream>
```

```
using namespace std;
```

```
int main () {  
    cout << "Hello world";  
    return 0;  
}
```

#include:

Tells the compiler that we may use the keyboard or the screen in our program.

This is called a ..

The file *iostream* is included for our use.

Simple C++ Program

```
// A simple C++ program.  
#include <iostream>  
using namespace std;  
  
int main () {  
    cout << "Hello world";  
    return 0;  
}
```

using namespace:

All **identifiers** (such as variable and function names) are inside a **namespace**.

Basically, this states that we want to use identifiers in the **std** namespace.

..

Simple C++ Program

```
// A simple C++ program.  
#include <iostream>  
using namespace std;
```

```
int main () {  
    cout << "Hello world";  
    return 0;  
}
```

int main() { ... }:

Creates the `main()` function.

The `main()` function is..

Functions are named
collection of statements.

Note: C++ is case sensitive!
`main()` is different than `Main()` or `MAIN()`!

Simple C++ Program

```
// A simple C++ program.  
#include <iostream>  
using namespace std;  
  
int main () {  
    cout << "Hello world";  
    return 0;  
}
```

`int` is the **return type**.

In this case, an integer.
It is the type of information the program "returns" to the OS.

`main` is the name of our function.
Each program we create must..

The `()` indicates this is...

The `{ ... }` indicates a **block**.
In this case, a block of statements associated with the `main()` function.

Simple C++ Program

```
// A simple C++ program.  
#include <iostream>  
using namespace std;  
  
int main () {  
    cout << "Hello world";  
    return 0;  
}
```

cout...

Think of << as sending the string to **cout**.
(**cout** = character out).

Completed statements end with a semicolon.
We will later get a better feel for this.
For now, just concentrate on the parts of the program.

Simple C++ Program

```
// A simple C++ program.  
#include <iostream>  
using namespace std;  
  
int main () {  
    cout << "Hello world";  
    return 0;  
}
```

return:

The **return** statement in the **main()** function returns a value to the operating system.

Returning 0 to the OS indicates success (by convention).

..

Review

- 1) What C++ statement prints "I love programming" to the screen?
- 2) What part of a C++ program is first to be executed?
- 3) What is wrong with this C++ statement?
`cOut >> "Hello!"`

Tools

How to build
an executable.



Build Process

- We write C++ code; computer runs machine code.

C++ Source Code

myfile.cpp



```
graph LR; A[myfile.cpp] --> B[myfile (.exe)];
```

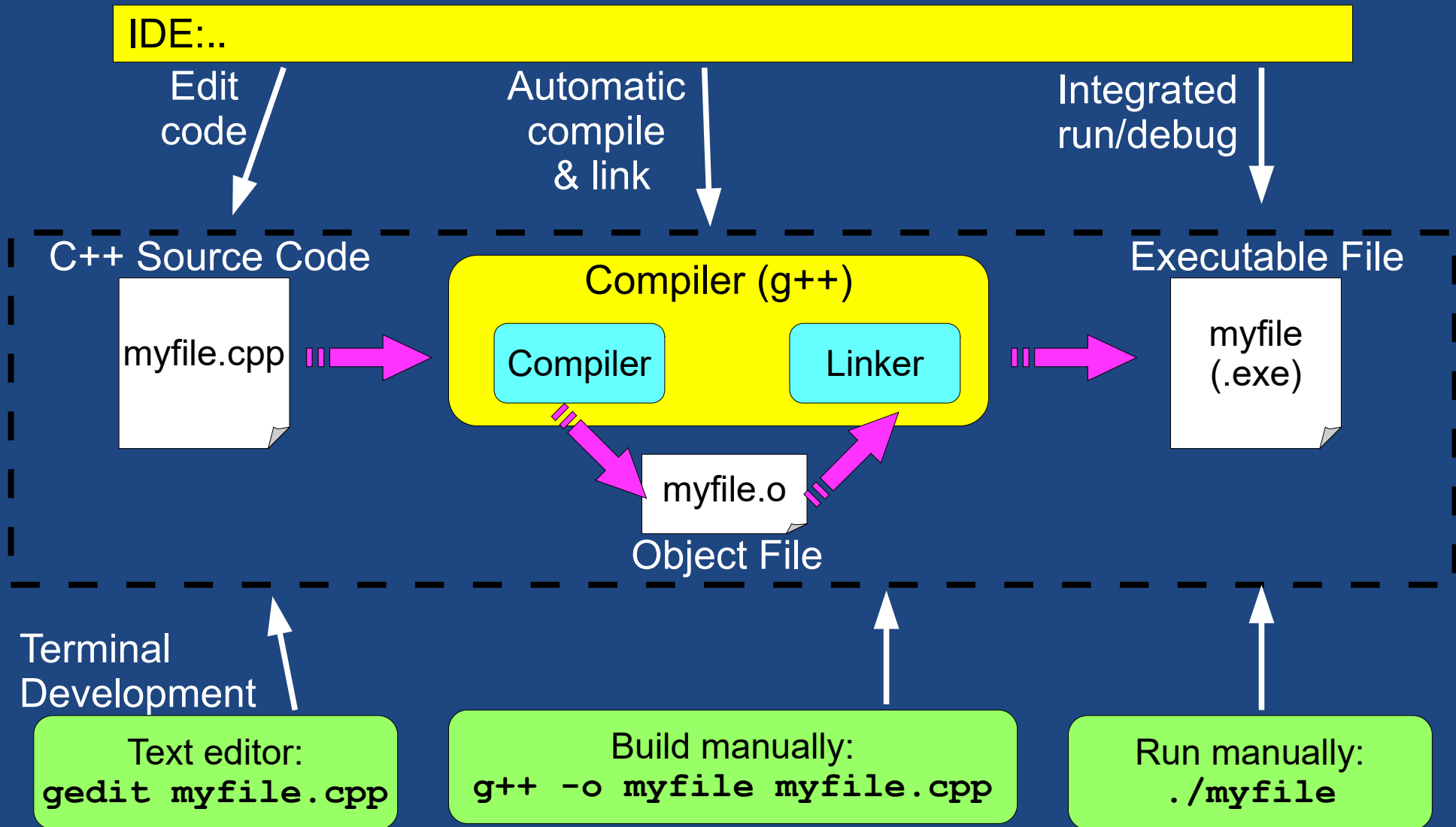
The diagram illustrates the build process. On the left, under the heading 'C++ Source Code', there is a white document icon with a folded corner containing the text 'myfile.cpp'. A thick pink arrow points from this icon to the right. On the right, under the heading 'Executable File', there is another white document icon with a folded corner containing the text 'myfile (.exe)'. A second thick pink arrow points from the first icon to the second, indicating the transformation of source code into an executable file.

Executable File

myfile
(.exe)

- **Build...**
- **Tool Options**
 - **IDE...**
All tasks done through graphical user interface (UI)
 - **Terminal Development:**
All tasks done through a command prompt.

Tools Options



Visual Studio Code

The screenshot shows the Visual Studio Code interface with the following components:

- Window Title:** hello.cpp - cmpt130 - Visual Studio Code
- Menu Bar:** File, Edit, Selection, View, Go, Run, Terminal, Help
- Explorer:** Shows a project structure for 'CMPT130' with subfolders '.vscode' and 'firstDayDemo'. The 'firstDayDemo' folder contains files 'gcd', 'gcd.cpp', 'hello', 'hello.cpp', 'helloworld', and 'helloworld.cpp'. The 'hello.cpp' file is selected.
- Editor:** Displays the content of 'hello.cpp' with the following code:

```
1 // A simple C++ program.
2 #include <iostream>
3 using namespace std;
4
5 int main () {
6     cout << "Hello world";
7     return 0;
8 }
```
- Terminal:** Shows the output of running the program. The output is 'Hello world[1] + Done'. The terminal prompt is 'brian@Ubuntu20:~/cmpt130/firstDayDemo\$'.

The `cout` Object

cout

- **cout** (Pronounced “C Out”, not “kout”)
 - Think of it as **character out**, or **console out**.
- **cout** is a **stream object**:
 - It operates on a **stream** (sequence) of characters.
- **<<** is the **stream-insertion operator**:
 - Use it to push text into **cout**
cout << "Wow! Programming is fun!";
 - Think of **<<** as an arrow point to the left:



Multiple Strings

- You can send multiple different strings to `cout`:

```
// Displaying multiple strings.  
#include <iostream>  
using namespace std;  
  
int main () {  
    cout << "Programming is " << "great fun ";  
    cout << "all the time!";  
  
    return 0;  
}
```

Notice all the strings are run together, even though they are from separate statements.

Common Problem

- What is the problem with the following?

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

```
int main () {
    // Demonstrate a common problem
    cout << "My favourite numbers are: ";
    cout << "0";
    cout << "42";
    cout << "73";
    return 0;
}
```

```
..
```

Line Feeds

- Can put line feeds in with either:
 - End Line Stream Manipulator: `endl`
`cout << "First line." << endl;`
`cout << "Second." << endl << "Third.";`
 - New Line Character: `"\n"`
`cout << "First line.\n";`
`cout << "Second.\n" << "Third.";`

..



Special Characters

- **Escape Sequences:**

..

- New line: "One \n on \n top"
- Tabs (line up): "Age: \t"
- A \ character: "Up \\ down"
- A ' character: "'\m lovin\' programming!"
- A " character: "I said, \"Yes!\" too"

- Note that the escape sequence must be inside a string, whereas **endl** must **not** be in the string.

Escape Sequence Example

```
// Demonstrate escape sequences and endl
```

```
#include <iostream>
```

```
using namespace std;
```

```
int main () {
```

```
    cout << "Movie Lineup\n";
```

```
    cout << "7:30\tSpace Balls" << endl;
```

```
    cout << "10:40\tIt's a Wonderful Life" << endl;
```

```
    cout << "12:30\tGone with the Wind"<<endl<<endl;
```

```
    cout << "He'll say, \"They're great!\"\n";
```

```
    return 0;
```

```
}
```



```
C:\Users\Brian Fraser\Documents\Visual Studio 2010\Projects\Example_IntroCpp\Debug\MultipleStrings-C...
Movie Lineup
7:30      Space Balls
10:40     It's a Wonderful Life
12:30     Gone with the Wind

He'll say, "They're great!"
```

Spot the Mistakes

```
// Show some common mistakes.
```

```
#include <iostream>
```

```
using namespace std;
```

```
int main () {
```

```
    // Spot the mistakes:
```

```
    cout << "C++ is fun! endl";
```

```
    cout << "Computers are awesome!" << \n;
```

```
    cout << "Amazing stuff!/n";
```

```
    cout << "I say "Yeah!"" >> endl;
```

```
    return 0;
```

```
}
```


Review

- 1) Write one or more C++ statements which output the following (including tabs, and line-feeds):

Name: "Brian"
Fav-Colour: Green

To err is human,
but to really foul things up
you need a computer.

Paul Ehrlich



1982: Bug in software
controlling Soviet pipeline
causes largest man-made non-
nuclear explosion in history.

Errors

- **Compile Error**
 -
 - Syntax errors, such as forgetting a ;
 - Semantic errors, such as invalid type casting.
- **Run-time Error**
 - Errors causing...
such as an un-checked divide by zero (exceptions).
- **Logical Error**
 -
 - Caused by programmer error (bug).

Debugging

- Most (all?) large programs have bugs.
 - You'll spend a large amount of time debugging!
- QA is Quality Assurance
 - The task of showing the program is..
 - Cannot reasonably **prove** that there are no bugs:
 - Can show it works for..
Ex: square root of 16?
 - Can show it works for..
Ex: square root of 0? -1? 4 billion?

Summary

- Simple program: "Hello world!"
- Output to the console with `cout`.
 - `cout << "One " << "Two";`
 - `cout << "With 2 line feeds\n" << endl;`
- Compile C++ code to machine code.
- Escape Sequence: `\n`, `\t`, `\\`, `\'`, `\"`
- 3 types of errors:
 - `Compile, run-time, logical.`