G52CPP
C++ Programming
Lecture 1
Extra Material

Dr Jason Atkin

E-Mail: jaa@cs.nott.ac.uk

"Hello World"

A simple C++ (and C) program

Since you have had 2 semesters to forget what you did in G51PRG

The “Hello World” Program

```c
#include <stdio.h> /* C file */

int main(int argc, char* argv[])
{
  printf("Hello world!\n");
  return 0;
}
```

C version

```cpp
#include <cstdio> /* C++ file */

int main(int argc, char* argv[])
{
  printf("Hello world!\n");
  return 0;
}
```

C++ version

Compiling as C++

- Name the files **`.cpp`** instead of **`.c`**
- Can still include header files
- C preprocessor still exists
  - Can still use **`#define`**
    - Templates often make macros unnecessary
  - Can still use **`#include`**
    - But see next slide
- Conditional compilation is still needed in order to avoid multiple header file inclusion
  - i.e. **`#ifndef  #define  #endif`**

Using the C library functions

- The **standard C library** is still available
- **Header files have changed names**
  - Add a **`c`** at the beginning and remove the **`.h`**
  - e.g. **`#include <cstdio>`**
- C++ header files MAY differ from the C versions
  - But provide the same functionality
  - *In C++ they may not actually be files*
- Functions are in the **`std`** namespace
  - We will consider namespaces later
- But, also available in global namespace
  - So you can use them as global functions
- Need to link to the **`libstdc++`** library in gcc
  - e.g. **`gcc test.cpp –lstdc++ -o test`**
  - or **`g++ test.cpp –o test`**

#include

```
#include <cstdio>
int main( int argc, char* argv[] )
{
  printf("Hello world!\n");
  return 0;
}
```

- Pre-processor command – processed before compilation
- It means: “Replace the statement “#include <…>” by the contents of the file(?) called ‘**cstdio**’ BEFORE compiling”
- **stdio.h/cstdio** declare many STanDard Input and Output functions, some other functions and some constants

int main(int argc, char* argv[])

```
#include <cstdio>
int main( int argc, char* argv[] )
{
  printf("Hello world!\n");
  return 0;
}
```

- Define a function called **`main()`**, which returns a value of type '**`int`**' and has two parameters called **`argc`** and **`argv`**
- Your program will start with a call to your '**`main`**' function
- **`argc`** and **`argv`** specify the command line arguments
- **`argc`** is of type '**`int`**' and is the count of arguments

char* and argv

```
int main( int argc, char* argv[] )
```

- **`argv`** is of type '**`char*[ ]`**',
  - an array of '**`char*`**'s, or C-style strings
  - The elements of the array are the command line arguments
- Remember:
  - **`char*`** is a **pointer** to a character
  - In this case, a pointer to an array of characters
  - With a value 0 at the end
- This is the only type of string available in C, but C++ provides us with more possibilities

printf("Hello world!\n");

```
#include <cstdio>
int main( int argc, char* argv[] )
{
  printf("Hello world!\n");
  return 0;
}
```

- Send the string “Hello World” to the output stream
- This will usually be displayed in the output window
- Similar to the Java: **`System.out.print()`**
- BUT: The ‘f’ at the end of **`printf`** stands for ‘formatted’
- **`printf`** is a powerful function for formatting output