

Lab 4 Computing Systems

Compilation with C++

In this lab you will explore the C++ compilation process. This will highlight some of the steps that occur during the compilation process. The example program also highlights problems that can occur when using floating point numbers – and that can cause problems if not anticipated.

1. Download and save the C++ Project

Download the file **CS-compiler-lab.zip** from Moodle. If you are working on the university computers, I recommend saving the file to the C:/Temp folder, otherwise save in your normal documents folder.

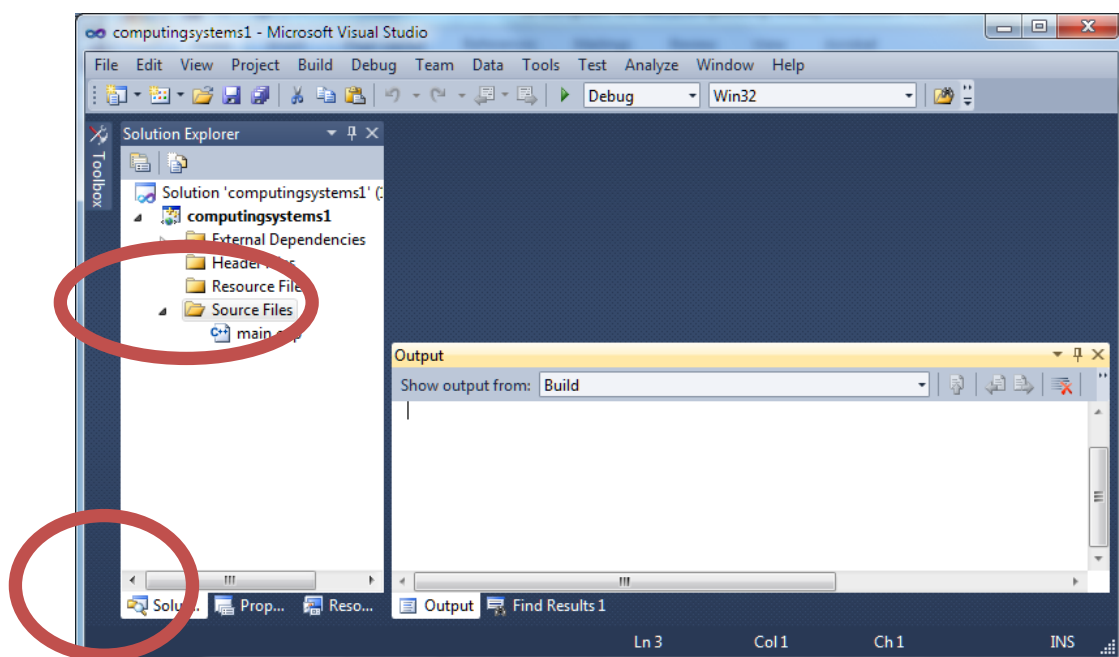
2. Explore the project files

Unzip the file and open the project folder. Explore the folder contents – you should find a Visual Studio project file called *computingsystems1*, and a sub-folder with the same name. In the sub-folder is another project file (again with the name *computingsystems1*), and a single source code file *main.cpp*. *Main.cpp* is the only program source code files – the other files are simply used by Visual Studio to store project information and settings

3. Open the Project

If you are using a computer that has Visual Studio 2010, you can open the project by double clicking on the *computingsystems1.sln* file. *If the computer you are using has a C++ compiler, but not Visual Studio 2010, you should still be able to compile and run the program by creating a new project using the main.cpp source code file. There is no room to provide instructions on this here. If you do not have Visual Studio 2010, notes have been placed on Moodle on how to obtain legitimate free copies of the software through academic licensing programs.*

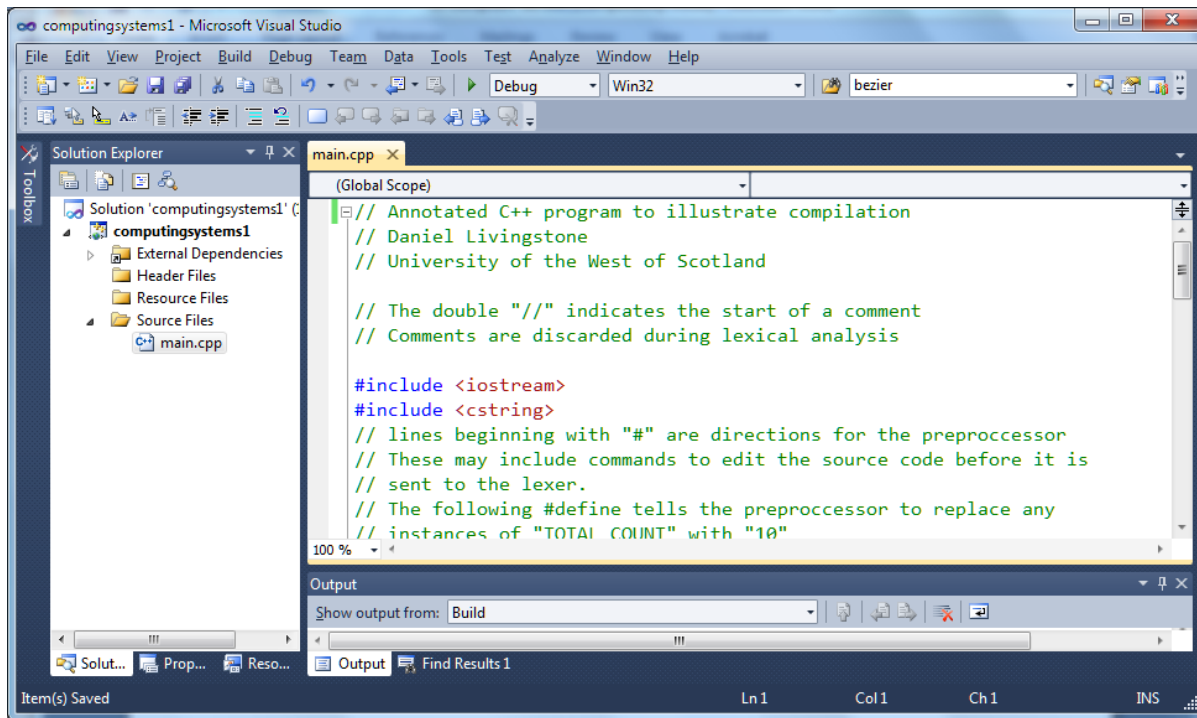
DO NOT COMPILE OR RUN THE PROJECT YET!



Lab 4 Computing Systems

4. Visual Studio Quick Start

The Visual Studio IDE (Integrated Development Environment) should look something like the image shown above. You may need to use the **Solution Explorer** (highlighted) to see the list of source files (just `main.cpp` in this case). A lab demonstrator can help here if you have not previously used an IDE. Double clicking on `main.cpp` will open the source code window:



Take some time to read through the source code of the program. *Can you tell what the program does?*
The source code is repeated below for convenience.

```
// Annotated C++ program to illustrate compilation
// Daniel Livingstone
// University of the West of Scotland

// The double "/" indicates the start of a comment
// Comments are discarded during lexical analysis

#include <iostream>
#include <cstring>
// lines beginning with "#" are directions for the preprocessor
// These may include commands to edit the source code before it is
// sent to the lexer.
// The following #define tells the preprocessor to replace any
// instances of "TOTAL_COUNT" with "10"
#define TOTAL_COUNT 10

using namespace std;
```

```
void hexOut(float f); // Forward declaration of a function
// The hexOut function is actually at the end of the program.

int main() { // C++ expects a function named "main" as the program start
    // In the following two lines, three variables are declared
    unsigned int i; // an unsigned binary integer
    float a = 0.0f, b = 1.0f; // two floating point numbers

    for (i = 0; i < TOTAL_COUNT ; i++) { // This loop will repeat 10 times
        a = a + 0.1f; // increment a by 0.1 each time
        cout << "Step " << i << ": a= " << a << endl;
    }

    cout << "After loop, a = " << a << ", b = " << b << endl;

    if ( a == b ) // is a equal to b?
        cout << endl << "a is equal to b" << endl;
    else
        cout << endl << "a is NOT equal to b" << endl;

    // Print the hex of the two float representations
    hexOut(a);
    hexOut(b);

    cout << "Press return to continue..." << endl;
    getchar(); // This line waits for the keyboard to be pressed
               // Just here to stop the console window closing before
               // you have time to read the program output!
}

// This is an example of a C/C++ procedure, is called from the main program
// and prints out the hexadecimal of a floating point number
void hexOut(float f) {
    unsigned u;
    memcpy(&u, &f, sizeof f);
    cout << hex << u << endl;
}
```

5. What does the program do?

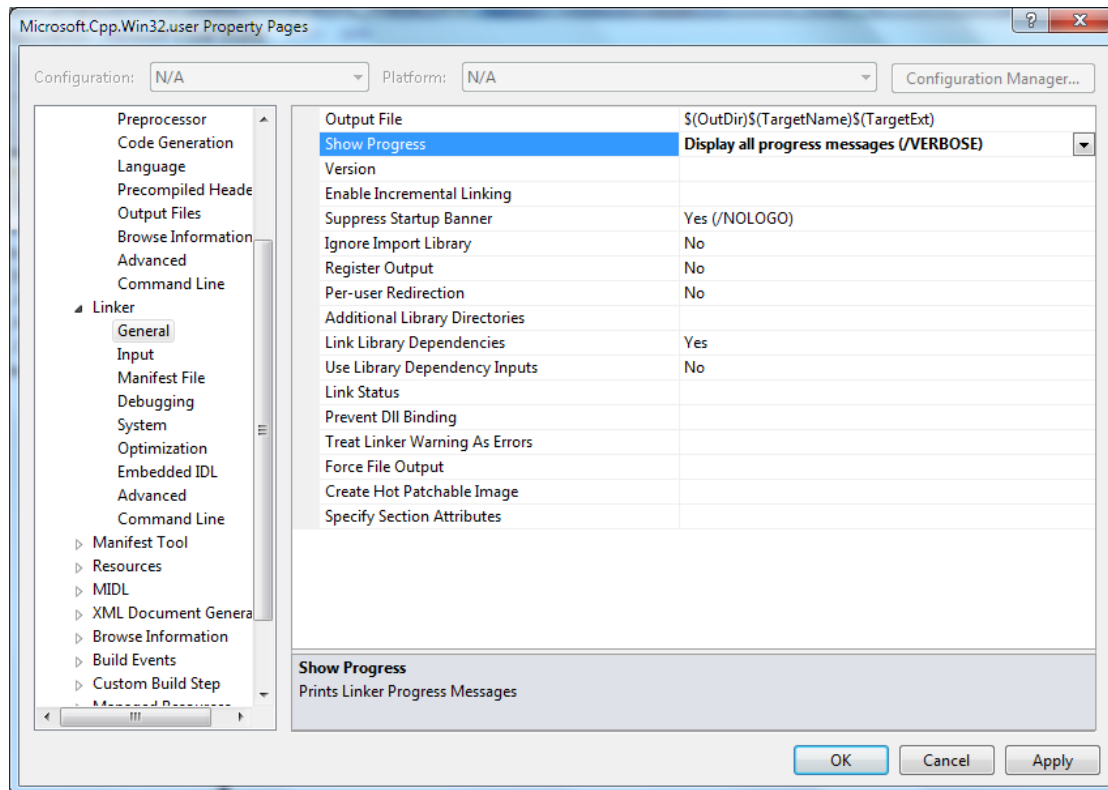
Read through the program source code again. Are you able to work out any details of what the program does? In particular, think about what happens in the *for loop* – this is a block of code that is going to repeat a set number of times, performing the same calculation each time.

Write down what you think the program does, and what you expect the output to be, before moving to the next step in the exercise.

Lab 4 Computing Systems

6. Compiling the Program

Before you compile the program, you should change one of the Visual Studio default settings so that you can see more detail on what is involved in the build process. Hold down the **ALT** and **F7** keys to open the project properties manager (or select **Properties** from the **Project** menu). In the dialog that appears, open the **Configuration Properties, Linker, General** tab, as shown below:



Change the **Show Progress** option to **Display all progress messages**. Click OK to close the dialog.

Now you can compile the program by pressing the **F7** function key, or selecting **Build -> Build Solution** from the program menu. As the project compiles and links, an output window (below the window that shows the project source code) should detail the progress. The output from this is too long to consider in detail, but should start something like this (you may see some additional lines at the start depending on your local configuration):

```
1>----- Rebuild All started: Project: computingsystems1, Configuration: Debug Win32 -----
1> main.cpp
1>
1> Starting pass 1
1> Processed /DEFAULTLIB:msvcprtd
1> Processed /DEFAULTLIB:MSVCRTD
1> Processed /DEFAULTLIB:OLDNAMES
1>
1> Searching libraries
1>   Searching C:\Program Files\Microsoft SDKs\Windows\v7.0A\lib\kernel32.lib:
1>   Searching C:\Program Files\Microsoft SDKs\Windows\v7.0A\lib\user32.lib:
1>   Searching C:\Program Files\Microsoft SDKs\Windows\v7.0A\lib\gdi32.lib:
1>   Searching C:\Program Files\Microsoft SDKs\Windows\v7.0A\lib\winspool.lib:
```

Lab 4 Computing Systems

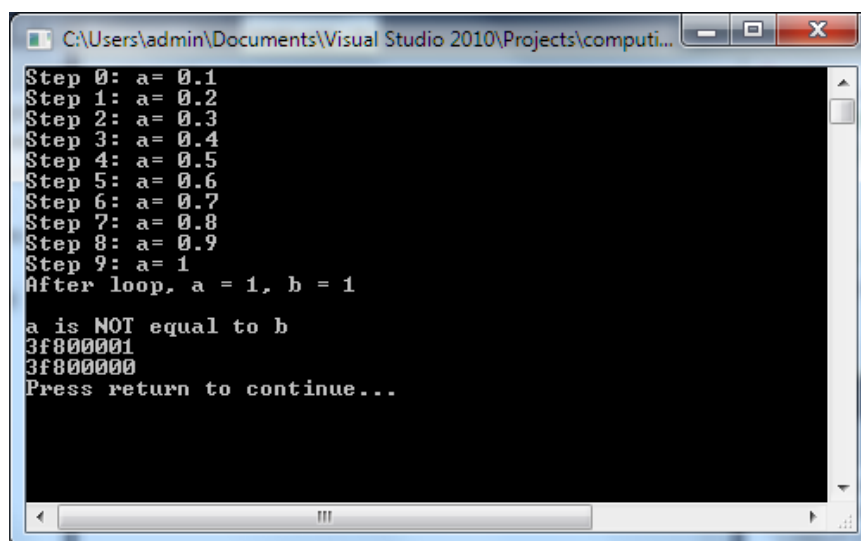
There may be a number of lines that refer to different default libraries - either to be included or excluded (/DEFAULTLIB or /NODEFAULTLIB), before the first pass is begun. Compilation in C++ is a two-pass process, with each pass performing part of the compilation. One example of why this is needed can be seen with the function **hexOut** – this is called from inside the main function, but the code for the function is at the end of the program. During the first pass, the compiler does not know where in memory the hexOut function would be – so it does not have sufficient information to complete compilation until after it has completed its first pass through the source code. Multi-pass compilation also allows compilers to perform additional optimisations.

The following lines refer to different library (.lib) files. These files contain useful functions that a programmer may use – and that have already been compiled. If required, the code from these will be linked into our project.

7. Running the Project

You can compile and run the program quickly just by pressing the **F5** function key on the keyboard. (Alternatively, click on the green 'play' arrow on the menu bar, or choose the menu option **Debug -> Start Debugging**)

If all is well, the program will compile and run and you will see the following appear in a new console window:

A screenshot of a Visual Studio 2010 console window. The title bar shows the file path: C:\Users\admin\Documents\Visual Studio 2010\Projects\computi... The console output is as follows:

```
Step 0: a= 0.1
Step 1: a= 0.2
Step 2: a= 0.3
Step 3: a= 0.4
Step 4: a= 0.5
Step 5: a= 0.6
Step 6: a= 0.7
Step 7: a= 0.8
Step 8: a= 0.9
Step 9: a= 1
After loop, a = 1, b = 1
a is NOT equal to b
3f800001
3f800000
Press return to continue...
```

The program initialises a variable **a** with the value 0. Then it iterates ten times through a loop, adding 0.1 to **a** each time. After completing the loop the program prints a line to show that **a** now has the value 1, and that variable **b** also has the value 1.

Finally, it compares the variables **a** and **b**, and declares that they are NOT equal in value.

This might be unexpected, but is due to the way in which computers use a binary floating point representation to store real numbers – as mentioned in class, not all values can be represented precisely.

The final two lines of program output show the hexadecimal of the floating point representations for **a** and **b** (a binary print out would have required 32 0's and 1's, hex is much briefer!). It is immediately obvious that the two values are NOT equal. But what are they?

Lab 4 Computing Systems

Check the values at <http://www.h-schmidt.net/FloatApplet/IEEE754.html> - enter a value on the line for the hexadecimal representation, and press return to see the binary and decimal equivalents.

8. Explore the project files again

For the final exercise, explore the project folders once more. There is a small self-test in the module materials section on Moodle – answer the questions there about your project.