Carleton University
Department of Systems and Computer Engineering
SYSC 2006 - Foundations of Imperative Programming

Lab 1: Updated May 1st

Attendance/Demo

You may do this lab on your own time or in the lab during your regular lab period. This lab **will** be graded by a TA, and successful completion will earn one bonus mark towards your lab mark.

Exercise 1 - Introduction to the Pelles C IDE

Objective

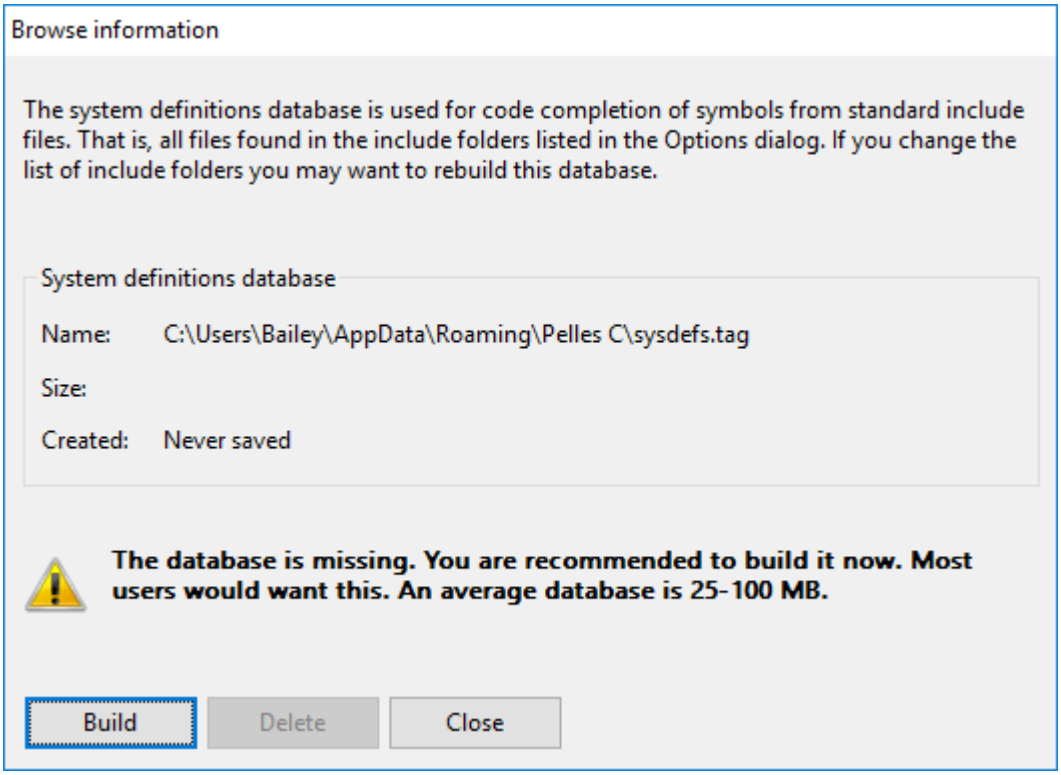
In this part of the lab, you'll learn how to use the Pelles C programming environment to prepare a simple C program. Specifically, you will be able to:

- create a new Pelles C project;
- create a new C source code file, use the editor and add the edited file to the project;
- build the project (create an executable program);
- execute the program.

Step 1

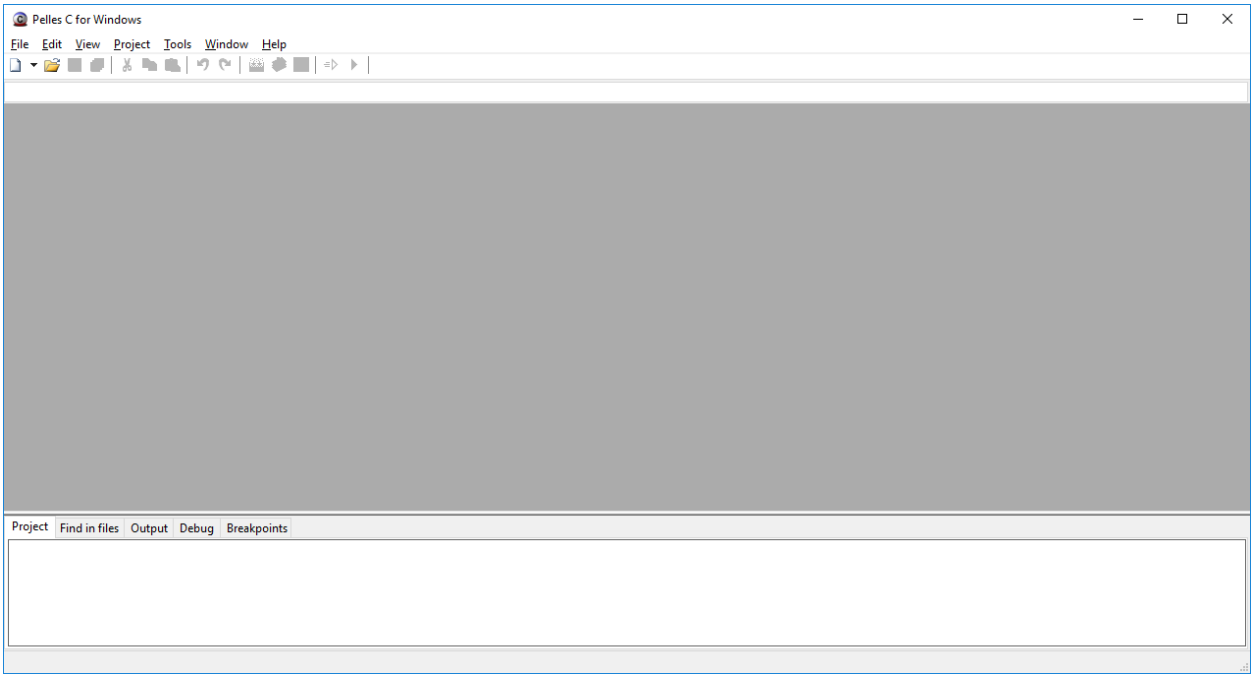
Create a folder named Lab 1.

Launch Pelles C. A Browse information dialogue box may open:



If this box appears, click the Close button.

Pelles C should now look like this. (If a pane labelled Start Page appears, close it by clicking the x to the right of Start Page.)



Step 2

You're now going to create a project named hello, which will contain the classic "Hello, world!" program that's often used as the first example when learning a programming language.

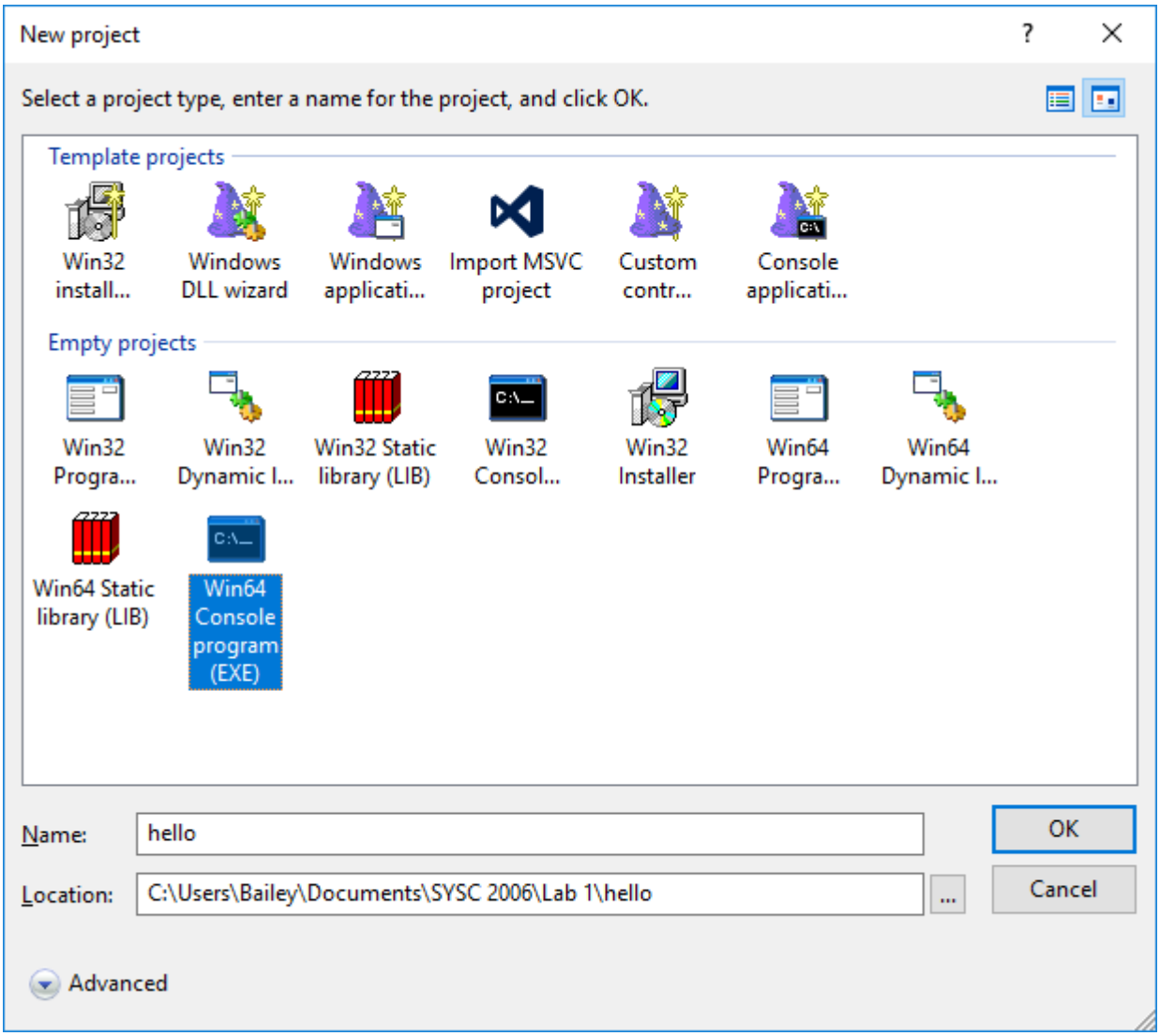
From the menu bar, select File > New > Project... A New project dialogue box will appear.

Clicking the ... button beside the Location: field allows you to navigate to the folder where you'll store your project. Navigate to the Lab 1 folder you created in Step 1. The path will appear in the Location field .

After navigating to that folder, type hello in the Name: field.

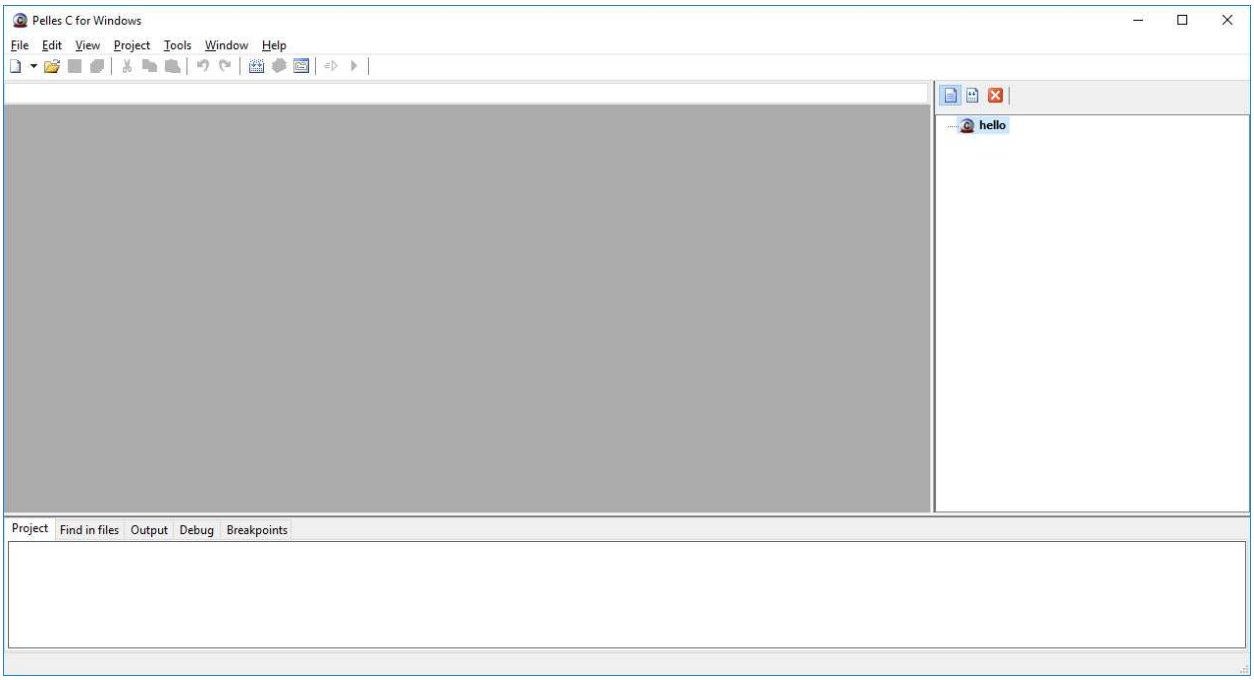
You also need to select the type of project that will be created. If you're using the 64-bit edition of Pelles C, the project type should be Win 64 Console program (EXE). If you're using the 32-bit edition of Pelles C, the project type should be Win32 Console program (EXE). **Do not click Win32 Program (EXE) or Win64 Program (EXE) or any of the other "Empty projects" icons.**

The dialogue box should now look similar to this:



Click the OK button. Pelles C will create a folder named hello inside the Lab 1 folder. The files associated with this project will be stored there.

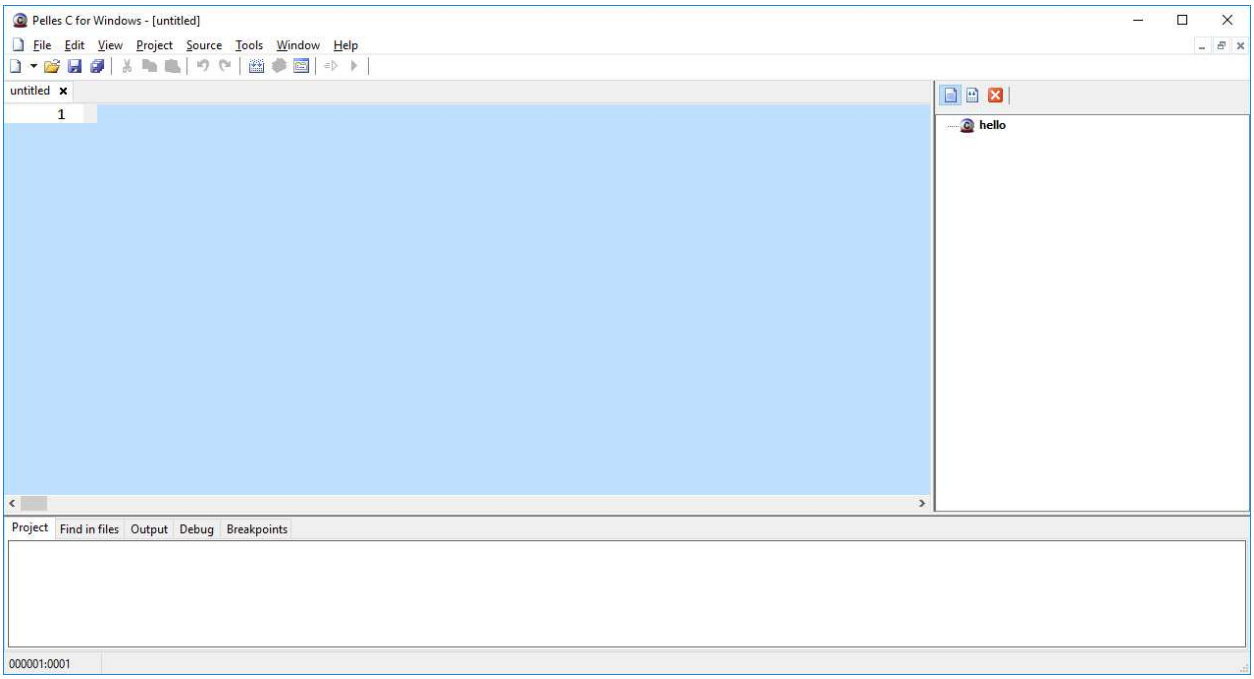
Pelles C should now look like this (notice that the project name, hello, now appears in the right-hand pane):



Step 3

From the menu bar, select File > New > Source code. An empty blue editor pane labelled untitled will appear.

Configure Pelles C to display line numbers by selecting Source > View linenumbers. Pelles C should now look like this:



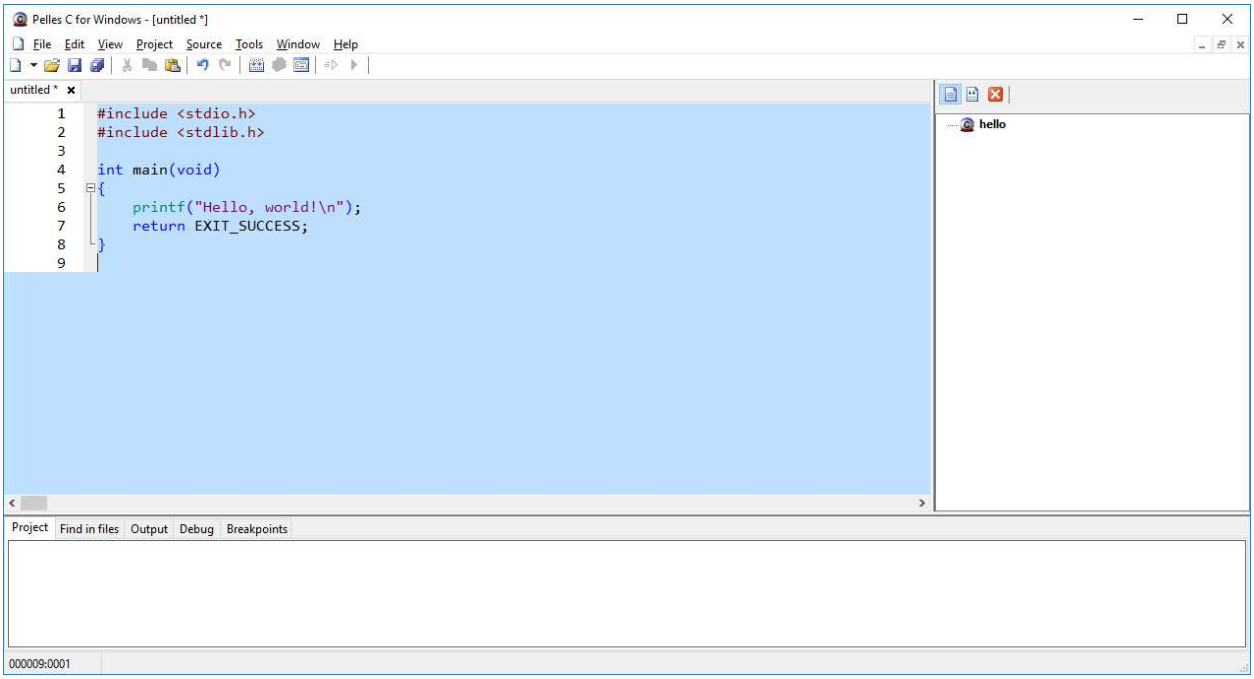
Step 4

Type this C program in the editor pane:

```
#include <stdio.h>
#include <stdlib.h>

int main(void)
{
    printf("Hello, world!\n");
    return EXIT_SUCCESS;
}
```

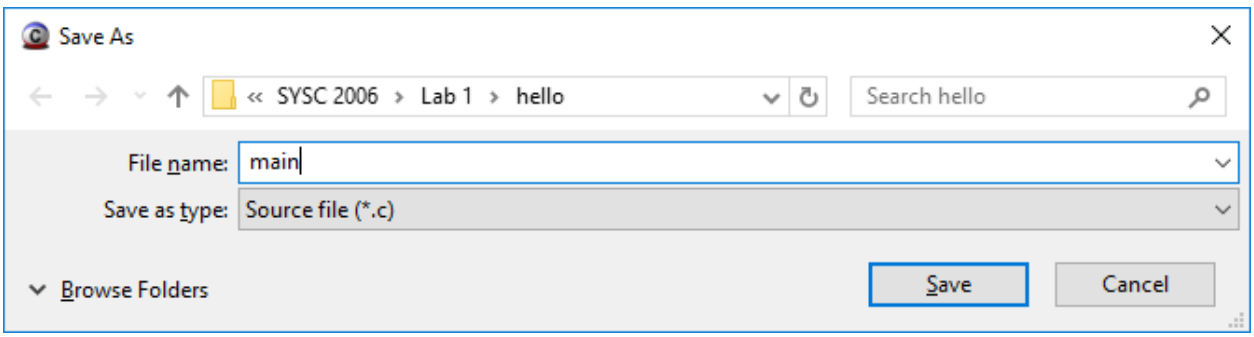
Pelles C should now look like this:



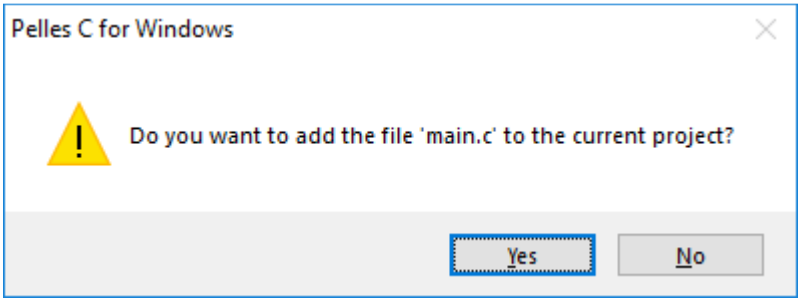
Step 5

You'll now save the source code in a file. Many C programmers always use main.c as the name of the file that contains the main function. We'll follow that convention in this course.

From the menu bar, select File > Save as... (or or click the Save button from the row of icons below the menu bar). A Save As dialogue box will appear. Type the name main in the Filename: field, and ensure that Save as type: is Source file (*.c).

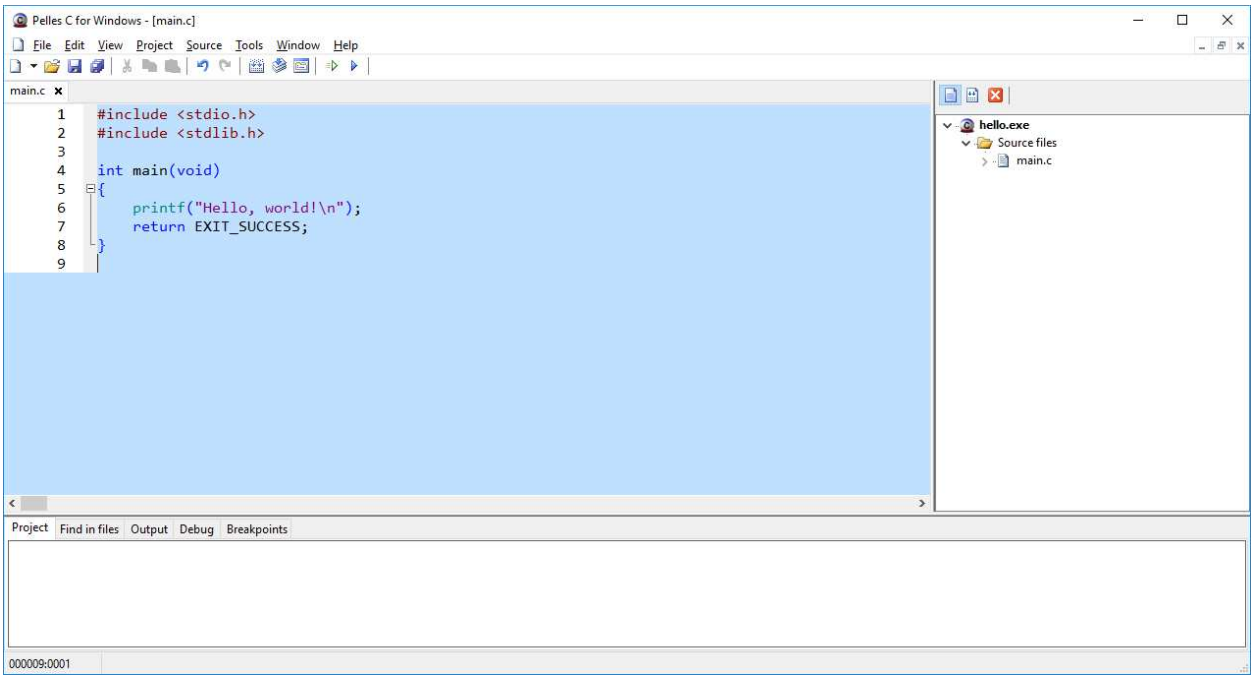


Click Save. You will now be asked if you want to add the source code file to the project:



Click Yes. Pelles C will save the code displayed in the editor pane in a file named main.c and add this file to your project.

Pelles C will now look like this:



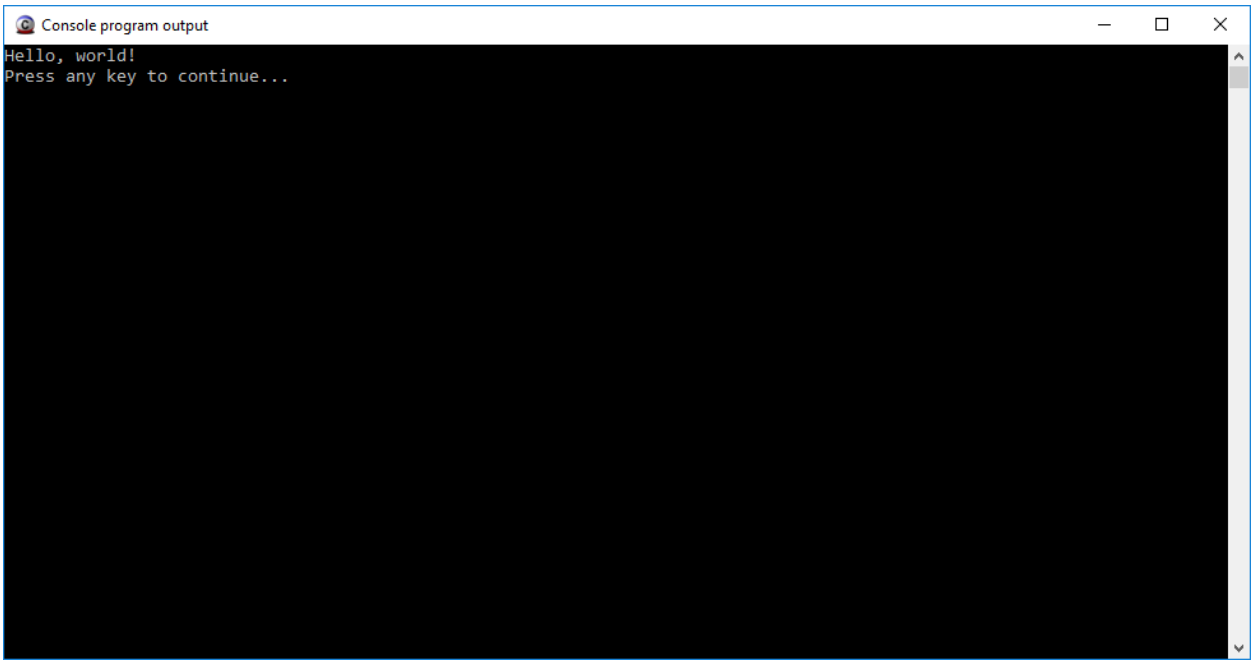
Notice that main.c is now listed as one of the source files in the hello project (see the right-hand pane).

Step 6

From the menu bar, select Project > Build hello.exe (or or click the Build button from the row of icons below the menu bar). Pelles C will compile the code in main.c, and if there are no compilation errors, link the compiled code to the C libraries that the program requires, producing a file named hello.exe file (the executable program).

Step 7

From the menu bar, select Project > Execute hello.exe (or or click the Execute button from the row of icons below the menu bar). A Console window will appear, showing the program's output:



Press any key to close the console window.

Step 8

Edit the program, adding a `printf` statement that outputs `"C programming is fun!"` on a separate line, below `"Hello, world!"`.

You'll need to save the modified source code. From the menu bar, select File > Save (or or click the Save button from the row of icons below the menu bar).

Build and execute the program. It should now display:

```
Hello, world!
C programming is fun!
```

Exercise 2 - Understanding Pelles C Compilation Errors

Edit the "Hello, world!" program you created in Part 1, adding the comment,

```
/* main: generate some simple output. */
```

after the `#include` statements, but before the definition of `main`. Your program should now look like this:

```
#include <stdio.h>
#include <stdlib.h>


/* main: generate some simple output. */


int main(void)
{
    Don't change the body of your main function from Exercise 1.
}
```

Build and execute the program. Did adding the comment have any effect on the output your program displays?

When learning a new programming environment, it is a good idea to write some programs that have syntax errors, so that you can see what error messages the compiler produces. Sometimes the compiler will tell you exactly what is wrong, and all you have to do is fix it. Sometimes, though, the compiler will produce wildly misleading messages, and you will have to figure things out yourself.

After you complete each of part of this exercise, correct the compilation error from that part by "undoing" the change you made to the program, before you start the next part. For example, in part (a), you'll see the error message the compiler produces after you remove a curly-bracket. Correct this error (insert the curly-bracket) before you start part (b). In other words, the changes you make to the program should not be cumulative.

(a) Remove the closing curly-bracket (`}`). Save the modified file and build the program. What error message does the compiler produce?

(b) Remove the opening curly-bracket (`{`). Save the modified file and build the program. What error message does the compiler produce?

(c) Remove the `int` before `main`. Save the modified file and build the program. What error message does the compiler produce?

(d) Change `main` to `mian`. Save the modified file and build the program. What error message does the compiler produce?

(e) Remove the closing `*/` from the comment. Save the modified file and build the program. What error message does the compiler produce?

(f) Change the first occurrence of `printf` to `pintf`. Save the modified file and build the program. What error message does the compiler produce?

(g) Delete one of the parentheses: `(` or `)`. Save the modified file and build the program. What error message does the compiler produce?

(h) Duplicate one of the parentheses: change a `(` to `((` or `)` to `))`. Save the modified file and build the program. What error message does the compiler produce?

(i) Delete the semicolon `;` after the `return` statement.

This exercise was adapted from exercises in *How to Think Like a Computer Scientist - C Version*, by Allen Downey and Thomas Scheffler.

Exercise 3 - Formatting C Code

Objective

In the exercise, you'll learn how to use Pelles C to format your C code so that it adheres to widely-accepted coding conventions.

Step 1

In your Lab 1 folder, create a new project named f_to_c. Remember, the project type should be Win32 Console program (EXE) or Win 64 Console program (EXE), depending on which edition of Pelles C you're using.

After creating the project, you should have a project folder named f_to_c in your Lab 1 folder. Check this. If you do not have a folder named f_to_c, close this project and repeat Step 1.

Step 2

Download file main.c from cuLearn. Move this file into your f_to_c folder.

Step 3

You must also add main.c to your project. To do this, select Project > Add files to project... from the menu bar. In the dialogue box, select main.c, then click Open. An icon labelled main.c will appear in the Pelles C project window.

Step 4

Build the project. It should build without any compilation or linking errors.

Step 5

Execute the project. The program should output a table of Fahrenheit temperatures and their equivalent Celsius temperatures.

Step 6

Open main.c in the editor. The file contains the Fahrenheit to Celsius temperature conversion program that was presented in a lecture. Clearly, the code is poorly formatted. Would you want to attempt to read and understand a source file containing 500 or 1000 lines of code if it was formatted this way?

Step 7

Pelles C makes it easy to reformat your C code so that it adheres to one of two commonly-used formatting conventions (K & R style or BSD/Allman). In addition to placing braces and indenting blocks of code, Pelles C will insert spaces or remove extra spaces when it reformats your code.

To select the formatting style:

- From the menu bar, select Tools > Options... An Options box will appear.
- Click the Tabs tab
- In the C formatting style box, click a radio button to select either Style 1 (for "roughly K & R" style) or Style 2 (which appears to be close to BSD/Allman style).
- Click OK. The Options box will close.

To format the code in your editor window:

- Select Edit > Select all. The code will be highlighted.
- Select Source > Convert to.
- From the submenu, select Formatted C code. The highlighted code will be reformatted to conform to the selected style.

Format the program once using Style 1, then a second time using Style 2. In this course, you can use whichever of these two styles you prefer.