Laboratory: Starting out with DEV C++

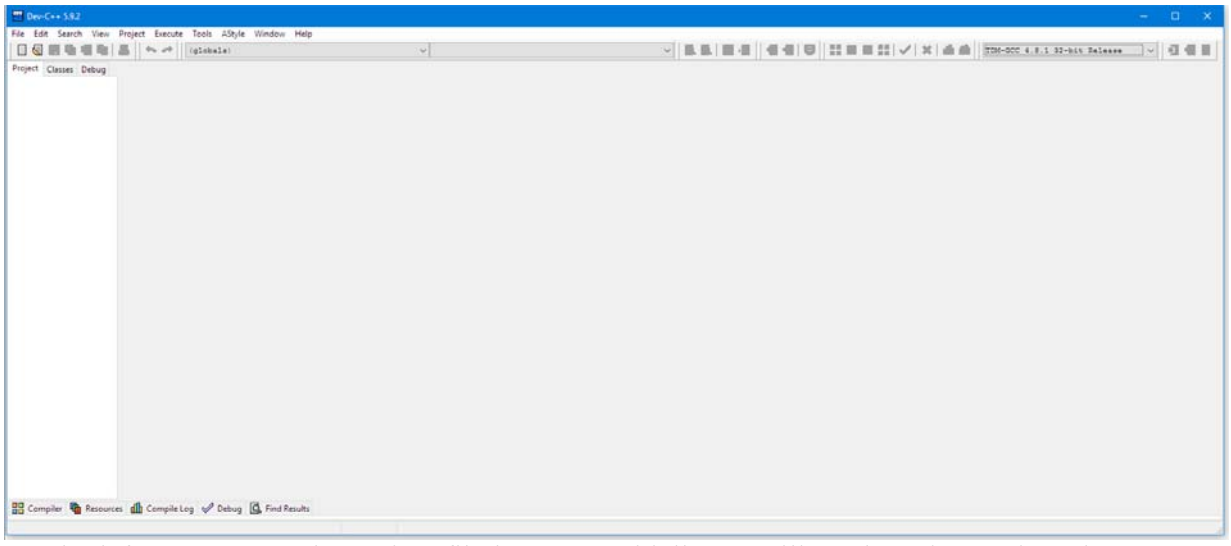
1. Introduction

Welcome to the wonderful world of programming. In this laboratory you will be learning how to develop programs and implement algorithms. To do this you have to write a text file with instruction in the language of your choice and then run them through a compiler. The compiler takes your text file and translates it into machine code (according to the compiler options you set) that can be executed by your computer. In order to do this the compiler also needs additional information about what object files and libraries your program might be using. Configuring all this by hand is not difficult but it is cumbersome and requires you (the human) to do a lot of repetitive work. To save you the trouble of doing all this and to help you write better code a new class of programs was developed: Integrated Development Environments, also known as IDEs.

IDEs include a text editor with syntax highlighting (displays source code text in different colors and fonts, according to the category of terms) and an interface for integrating the compiler in the same environment in which the user writes the program. Additionally IDEs usually include other useful features such as debugging (step by step execution, variable monitoring etc.), code completion and refactoring tools (code refactoring is the process of restructuring existing computer code "changing the factoring" without changing its external behavior).

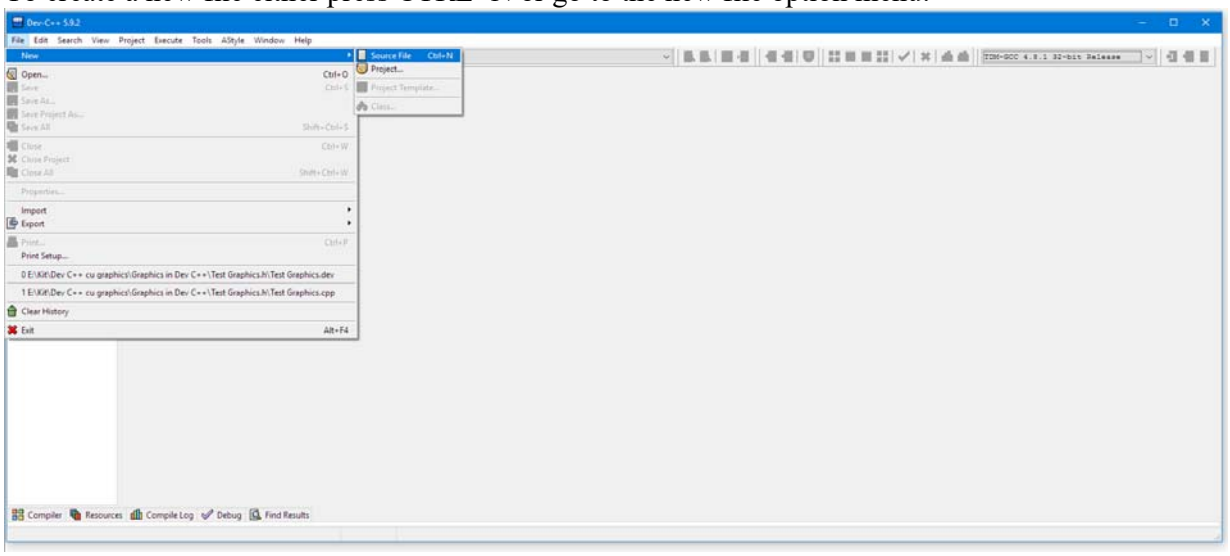
2. Workflow

The chosen IDE for this laboratory is DevC++ and it integrates several compilers for C and C++ that can be selected from a dropdown list.
First let's look at the main window



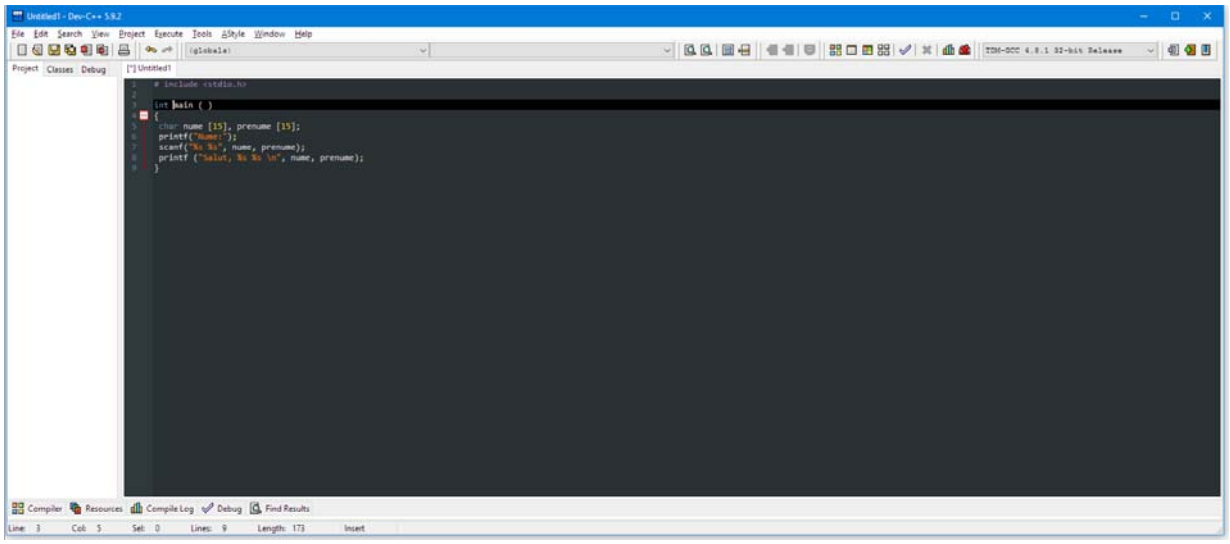
On the left you can see the project file browser. Initially we will not be using projects because our programs will be very simple but this window will allow you to switch between files when you are using a project.
In the top right you can see the compiler dropdown list.
On the bottom you will see the windows for Compiler output, Resources, Compile logs, Debug and the Results explorer. These will autoexpand when there is content available to display. The large gray area is where you will write your source code but to do that you need to create a new file.

To create a new file either press CTRL+N or go to the new file option menu:



Please note that we will not be opening a new project at this time.

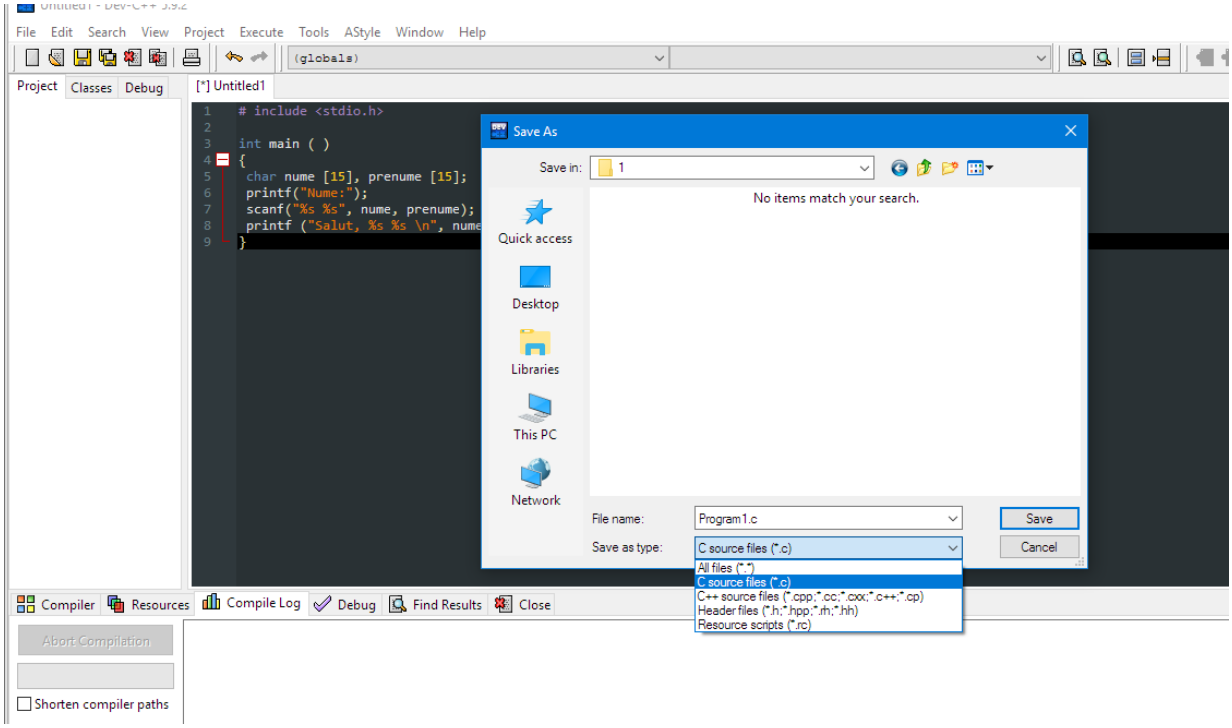
Once a new file has been opened you can start writing your code just like you would write any other text.



This is a very basic “Hello World” type program.
Note a few things: the syntax highlighting changes the color of the words to help you better vizualise the program, the bottom of the screen contains information on the cursor position in the file (line, column etc.), the lines are numbered to the left of each line and every section delimited with {} can be collapsed simply by clicking on the red minus sign next to it.

After writing your program you have to save it.

To save a program you need to press the “save” button shaped like diskette or press CTRL+S or go to File-> Save As. Once you have called the save command you will be prompted for a filename and save destination:

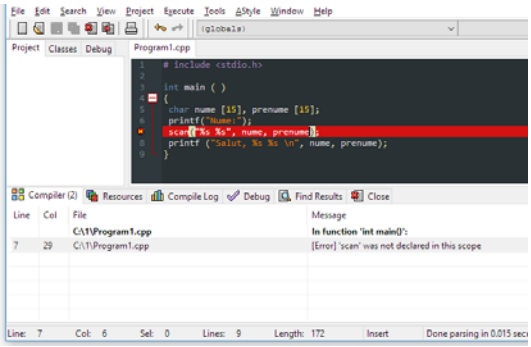


Pay very close attention to the filetype you are saving in! If you use C++ features in a .c file your compile might fail! The reverse is also true. Wrong file extensions can be the source of very strange errors that are hard to debug.

This basic demonstration program was written in C so you should save it as a “C source file (*.c)” and make sure it is saved that way.

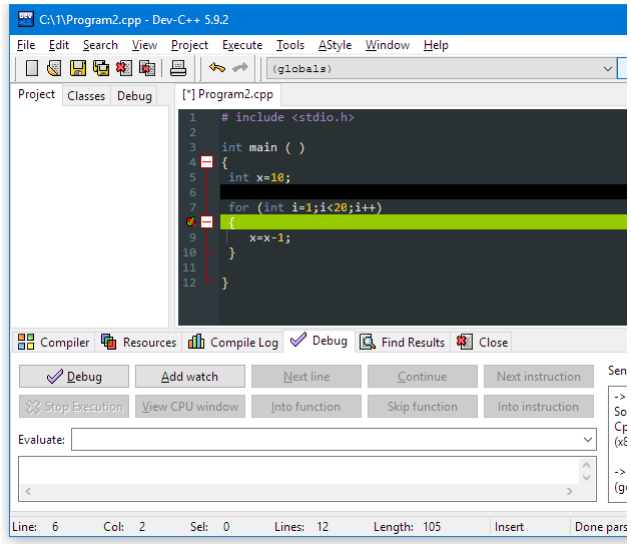
After it has been saved it can be compiled and run. To compile a program go to the “Execute” menu in the top menu bar. There you will have the option to Compile, Run, Compile&Run and Rebuild all. Click compile or compile and run. Alternatively if you prefer to use the keyboard you can use the F11 hotkey.

Your program should be compiled. If you have not run it then run it now.



If you have any errors they will be displayed in the compiler output area in the bottom of the screen.
For example in this case the user wrote scan instead of scanf and the compilation failed.
This is a typical syntax error that appears frequently but is easily fixed. Just remember that sometimes the error may be caused by an unfinished instruction above the line highlighted by the IDE like a missing ; or a missing paranthesis.

On the other hand your progam may be compiling correctly but you are unsatisfied by the output. For these cases the DEBUG mode was created. It allows you to track the functioning of the program step by step and inspect all the variables as they are loaded in the memory.
To start debug mode first compile your program to make sure there are no compiler errors.
Then go to the “Execute” menu and select debug (or press F5 or the purple checkmark button).
If prompted to enable debug mode in the compiler select “Yes”.
Add a breakpoint by clicking the line number:



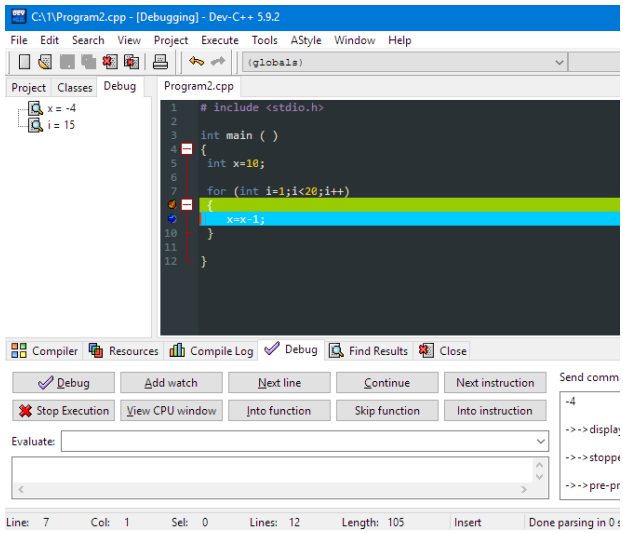
Clicking again will remove the breakpoint.
Breakpoints are simply places where program execution will pause.

With the breakpoint added press the Debug button. Your program should run until it encounters the breakpoint.
With the program paused try pressing the “Next line” button. Each time you press it the program advances one line.
WARNING: If you reach a line with and instruction that requests user input like “scanf” the program will wait for user input before continuing regardless of pressing the next line button.

Now that you know how to run your program line by line use this new ability: insert printf instructions in the code and see how the information is printed when you click “next line”.

There is one more useful function you should be aware of in Debug mode: variable watch. This allows you to view the values of variables as they change during the execution of the program.

To add a variable watch first stop the the execution of the program using the STOP EXECUTION button. Then click the add watch button and write the variable name. In the case of the program from the image we want to watch the **i** and **x** variables. Set these variable watches and then execute the program line by line. Do not forget about the breakpoint!



The variable values will be visible in the Debug tab on the left.

Click “Next line” and see them change.

The debugger is a very powerful tool so play around with it until you are comfortable using it. Try to find out what the other buttons in the debugger do.