MA400 Computer Session - Your Programming Environment

Mark Baltovic

September 18, 2009

1 The Dev-C++ environment

1.1 Saving your work

Always save your work to your H: drive.
Since you will be working with a number of small programs (and source files), it is advisable to create a folder in your H: drive for your work.

1.2 Accessing your work from Dev-C++

Dev-C++ is particular about how it access your files. If you access your H: drive using the "My Documents" shortcut, then you will get warning messages about UNC paths whenever you execute your compiled programs.[1]
You should therefore save and access your files via:

My Computer ⟶ H:

where your H: drive will be labelled along the lines of

```
Surname on 'Student\A_users' (H:)
```

2 Hello, World

1. Start Dev-C++ from the Start menu:

   Programs ⟶ Teaching ⟶ Bloodshed Dev-C++ ⟶ Dev-C++

2. Start a new source file by:

   - selecting **File ⟶ New ⟶ Source File**,
   - using the keyboard shortcut **Ctrl+N**, or
   - using the icon (third button from the left on the first row of icons).

   Now you should have an editor window within a larger window. This is where you type your program.

3. Type in the following program:

```
// Prints "Hello, world!"
#include <iostream>
#include <cstdlib>     // For EXIT_SUCCESS
int main()
```

[1] Dev-C++ does not support network paths.

```
{
  std::cout << "Hello, world!\n";
  system("PAUSE");
  return(EXIT_SUCCESS);
}
```

4. Save this source file in your H: drive work folder.
   Be sure you avoid using "My Documents", and access this via **My Computer → H:** (see above).

5. Compile your program, by:

   - selecting **Execute ⟶ Compile**,
   - using the keyboard shortcut **Ctrl+F9**, or
   - using the compile icon (first button from the left on the second row of icons).

   If no errors are reported after compilation, execute the program by:

   - selecting **Execute ⟶ Run**,
   - using the keyboard shortcut **Ctrl+F10**, or
   - using the run icon (second button from the left on the second row of icons).

   Alternatively, both steps of compilation and execution can be done in a single step:

   - selecting **Execute ⟶ Compile and Run**,
   - using the keyboard shortcut **F9**, or
   - using the icon (third button from the left on the second row of icons).

6. You should be greeted by your program in a newly opened window.
   It waits for you to press any key to close the window (due to the `system(''PAUSE'')` function call.).

   Remark Note that your program is colour-coded as you type it. For example, different colours are used to indicate:

   - language keywords such as `int`,
   - constants such as "Hello, world!\n", and
   - compiler directives such as `include`.

3 Using the command line

At some point in the course, you will need to run your compiled executable from the command line (in order to pass it arguments). Here we shall briefly go over what it takes to both compile and run programs using the DOS command line.

3.1 Running `cmd`

To run the `cmd` command, you need to call up the **Run** window. This can be done by:

- selecting **Start ⟶ Run...**, or
- using the keyboard shortcut **Win+R**.

Once this is up, type cmd and hit **Enter**.

Once the window has opened, you should see the following prompt:

```
H:\>_
```

where the underscore (cursor) is blinking. Anything you type will start to appear after the `>`.

This indicates that you are in your H: drive. If you are somewhere else, e.g. you see C instead of H, you jump to your H: drive by simply typing in `H:` at the prompt and hitting Enter.

Once in your H: drive, you can navigate to your folder using the `cd` command. Thus, if your work folder is called **CPP** and is in the top level of your H: drive, then enter the following at the prompt (hit Enter after each line).

```
cd CPP
dir
```

This should then give you a listing of the files in your directory CPP.[2]

To move to the parent directory (which in this case is the H: drive itself) type

```
cd ..
```

and then hit enter.

3.2 Tab completion

Note that the DOS prompt allows **tab completion**. If, at any point in your typing a directory or file name, you hit the **Tab** button, the prompt will complete what you have written, subject to certain rules:

1. if there are many ways to complete what you have written, then it will cycle through all the possibilities in alphabetical order: each hit of **Tab** will advance through the list;
2. in particular, if you start hitting **Tab** without having typed anything, then it will start to cycle through all the files in the current directory.

Try this now.

3.3 Compiling your program from the command line

Suppose we have written a source file called hello.cpp and wish to compile it from the command line; that is, without using the Dev-C++ IDE. This can be done typing the following command at the command prompt

```
C:\Dev-Cpp\bin\g++ -o hello hello.cpp
```

(Note that the `C:\` part *is* required.) Tab completion may help to speed this up a little (but be sure to supply it some initial letters – there are a great deal of files in these directories to cycle through). In this case, you will then get

```
"C:\Dev-Cpp\bin\g++.exe"
```

(including the double quotes).

The result of running this command is that the source file hello.cpp is compiled, and you will then see a hello.exe file appear in the directory.
What happens if you omit the `-o hello` part?[3]

[2] Note that DOS is case insensitive here, so `cd cpp` would work just as well.

[3] In this case, you will get an executable with the default name a.exe.

3.4 Running your program from the command line

You can now run the executable from the command line by simply typing the name of the executable in at the DOS prompt and then hitting enter:

```
hello.exe
```

In this case, the program output will occur in the same window as your activities above.

Once again, tab completion can help you quickly select the file you want if your directory contains many objects.

4 Appendix - A basic template

The lecture slides do not list complete programs, but the relevant code fragments to illustrate various ideas. To see them in practice, you will need to expand them into complete programs.

Below is a template for a C++ program which should give you enough to start experimenting with these code fragments. (Of course, you may still have to add certain things like variable definitions to get them to work.)

```
#include <iostream>
#include <cstdlib>       // For EXIT_SUCCESS
using namespace std;     // Make all std names global

int main()
{


  // Insert your code here


  system("PAUSE");       // "Press any key to continue..."
  return(EXIT_SUCCESS);
}
```