

C++ for Numerical Methods

Session 1 - Introduction

In this first session, we will run and configure the Dev-C++ development environment which will be used in this course. We will then learn the process of compiling programs using some example code.

Objectives for this Session

By the end of this session you should have completed the following:

- Logged into a workstation and run the Dev-C++ Integrated Development Environment;
- Learnt how to create a new C++ project in Dev-C++, and be able to save them and open them again;
- Learnt how to compile a C++ project using Dev-C++ and be able to run the executable code which is created.
- Learnt how to read and understand some basic errors reported by the compiler.

Preparation for the next session

Before the next session you should have completed the following:

- Completed the Exercises on this sheet.
- Read all five sections of the *Basics of C++* chapter from the tutorial website.

Exercises

1. (a) Log in to the lab computer using the username/password combination supplied by your department.
(b) Locate the Dev-C++ 4.9.9.2 program and run it. If this is the first time the program has been run, you may need to configure some settings. The default options should be sufficient.
(c) Download the *session01_hello.cpp* program from the course website, save it to your user area (H: drive) and open it using the Dev-C++ program.
(d) Find the *Compile and run* toolbar button (or press *F9*) to compile the program. After compilation completes, it should run automatically. The program prints the words *"Hello World"* to the terminal and waits. Press a key to terminate the program.
2. The aim of this exercise is to introduce you to the compiler. It is important not to be put off by the sometimes overly verbose output the compiler produces. Its purpose is to help you fix your code, so before panicking you should take a moment to carefully read what it says!

- (a) Download the *session01_error1.cpp* source code from the course website, save it to your user area, and open it using Dev-C++. Compile this code like you did in the previous exercise. Note that this time, the compiler does not complete successfully. It produces an error similar to:

```
    session01_error1.cpp    In function 'int main(int, char**)':  
5  session01_error1.cpp    expected ';' before 'system'
```

Note: The wording the compiler uses may vary slightly according to the version being used.

The first line tells us which part of the code the error is in - it is not the actual error which is preventing the code from compiling. The second line has the form:

```
<line> <file> <error>
```

The compiler is telling us that we are missing a semi-colon. You should note that although the error is indicated as line 5, the semicolon should be placed at the end of line 4 (i.e. "before" the word *system*). For this reason you may need to look at the lines surrounding the line the compiler reports, to find the real error. If you add a semicolon to the end of line 4, the code should now compile and run.

Tip: You can identify the current line you are on by looking at the numbers in the bottom left of the Dev-C++ window.

- (b) Download the *session01_error2.cpp* source code from the course website, save it to your user area, and open it using Dev-C++. Compile the code and examine the error messages. They should look similar to:

```
    session01_error2.cpp    In function 'int main(int, char**)':  
5  session01_error2.cpp    expected ';' before 'for'  
5  session01_error2.cpp    'i' undeclared  
5  session01_error2.cpp    expected ';' before ')' token  
8  session01_error2.cpp    expected '}' at end of input
```

The purpose of this part is to highlight the importance of fixing errors in order. The errors reported on the third and fourth line are not actually errors; they are a consequence of a previous error (on the second line), and it will break your code further if you try and fix them! Add the semicolon again to the end of line 4 of the code and recompile. The compiler now reports

```
    session01_error2.cpp    In function 'int main(int, char**)':  
8  session01_error2.cpp    expected '}' at end of input
```

and our false errors are no longer reported.

C++ is known as a block-structured language; that is, code is contained within blocks. If a block is longer than one line long it must be surrounded by a matching pair of curly braces ({, }). The remaining error is a common error when the closing brace is missed out. In the code, insert a new line after line 6 and place a closing curly brace (}). The code should now compile successfully and run, printing "Hello World" and the numbers 0 to 9.

3. You should now begin reading the *Basics of C++* section of the tutorial website and try some of the example programs they provide.

Tips

Some things to check when your code doesn't compile:

- Have you forgotten a semi-colon at the end of a statement near the line-number at which the compiler reports an error?
- Do all curly braces match correctly?
- Are variable names spelt correctly? Remember C++ is *case-sensitive* and all keywords are written in lowercase.