

C Programming

Learning Package 1

Introduction to C Programming

Dr Scott Turner

School of Science and Technology

University of Northampton



**Introduction**

Before we look at the specific commands for programming in C, we will look at programming as a form of problem solving. Later in this Learning Package, the basis of C programs will be discussed, including what is the basic structure of the C programs.

**Learning outcomes**

At the end of this Learning Package, you should be able to:

* Discuss why programming and problem solving are related.
* Discuss what a life cycle is.
* Type in, compile, and use a C program.
* Describe some of the basic features of a C program.

**Study guide**

Session 1:

Section 1.1 Programming and problem solving SAQ 1.1

SAQ 1.2

Section 1.2 Worked Example of Psuedo-Code Task 1.1

Session 2:

Section 1.3 Installing the software Section 1.4 Introducing C

Section 1.5 Functions Section 1.6 Code to Results Task 1.2

Session 3:

Exercises SAQ 1.3

# 1.1 Programming and problem solving

Read pages 1-15 and pages 21-24 of the module textbook.

SAQ 1.1

Why is programming about problem solving?

SAQ 1.2

Fill in the missing words.

Stepwise refinement is the process of **b ing** a **p m** into **s r** and **s r** 'chunks'. The process continues until a **s\_t** of **e ily-s d** sub-**p s** has been arrived at.

(Hint: Look at page 4 of the module textbook).

**1.2 Worked example of Psuedo-Code**

Using psuedo-code to start of an algorithm describing how to open a bottle of lemonade.

First attempt:

* Pick up bottle
* Turn bottle top

The descriptions in the first attempt were not detailed enough, for example what is involved in ‘Turn bottle top’?

Second attempt:

* Lift bottle off the surface it was on.
* Hold bottle firmly in one hand.
* Grasp the bottle top firmly in the other hand.
* Twist the bottle top in anticlockwise direction.

The second attempt has more detail, but some of the instructions are still not clear. For example ‘Lift bottle off the surface it was on’. What steps are involved in this?

Third attempt:

* Hold bottle firmly in left hand.
* Lift bottle off the surface it was on.
* Hold the bottle top firmly in the thumb and fingers of the right hand.
* Twist the bottle top in anticlockwise direction
* If bottle top does not move then apply more force in the twisting action until bottle top moves.
* Continue twisting the bottle top until the bottle top is free of the bottle.

This process can be developed further to consider issues such as:

* Is this the correct bottle?
* What happens if the bottle top is stuck?
* What happens if the bottle is wet?
* How much force is needed to lift the bottle of the surface?

In the third attempt, we saw different types of instruction. An instruction such as “Twist the bottle top in anticlockwise direction,” describes an action that will be carried out without a decision to be made. An instruction such as “If bottle top does not move then apply more force in the twisting action until bottle top moves” includes a decision, and what follows next depends on that decision.

# Task 1.1

1. Can you expand these steps in the example above so what they do is explained in such a way that it is unambiguous?
2. Is this the only algorithm that could have been written to do this?
   1. **Installing the software**
      * A possible compiler for this material can be found at this rather long link: [http://sourceforge.net/projects/dev-cpp/files/Binaries/Dev-C++ 4.9.9.2/devcpp-4.9.9.2\_setup.exe/download?use\_mirror=switch](http://sourceforge.net/projects/dev-cpp/files/Binaries/Dev-C%2B%2B%204.9.9.2/devcpp-4.9.9.2_setup.exe/download?use_mirror=switch). Other C compilers could be equally used. (Disclaimer: **The software is produced by a third party not connected with The University of Northampton and is only a suggestion of a possible compiler. Therefore the University of Northampton or the author cannot accepted any responsibility for any damage due to downloading and installing the software. If you are unsure of the safety of the software please use another appropriate piece of software.**)
      * It should download a set-up file.
      * Once this is downloaded then run the file devcpp-4.9.9.2\_setup.exe and it should start the installation process.
      * Warning message may come up saying

“Welcome to Dev-C++ install program. Please do not install this version of Dev-C++ over an existing installation.”

* + - Press ok
    - You must agree to the licence if you want to continue. Read it and if you are happy with the terms, press the agree button.
    - It is best to press next on the next screen and install all the components.
    - On the next screen, press the install button for the installation process to continue.
    - You might be asked if you want to install this for all the users. This is up to you.
    - If it is your own personal machine usually press yes.
    - Close the installation window and the program will start.
    - A message titled Beta Version Notice will appear (see Figure 1.1). Read it and if you are happy press OK.

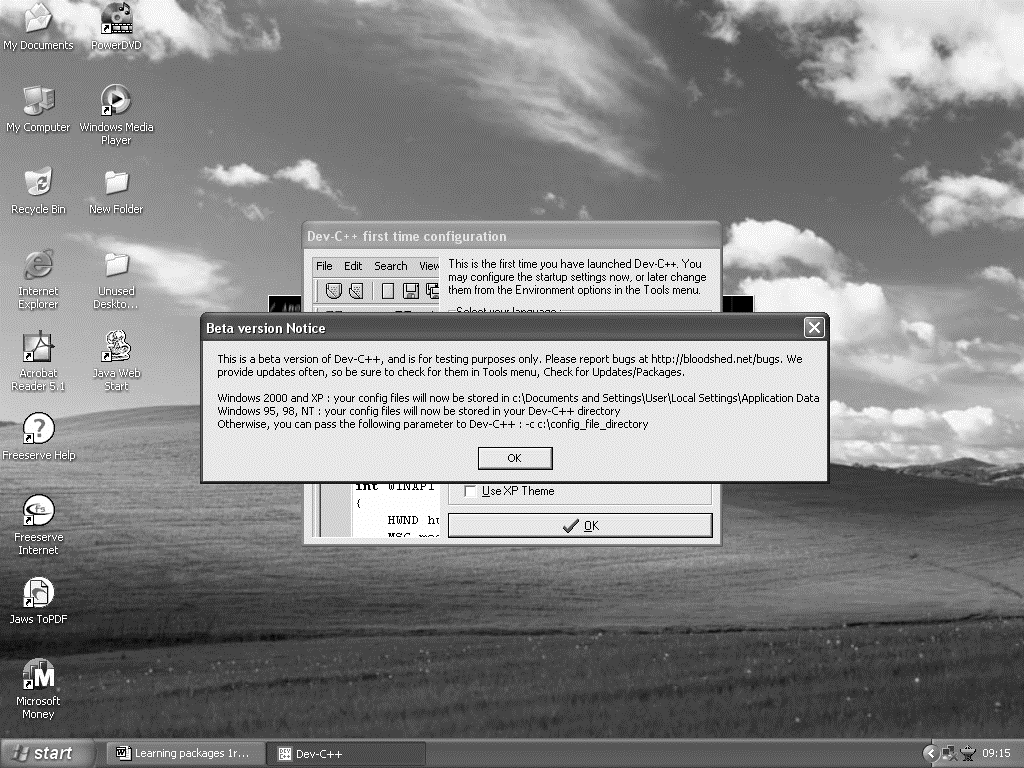


Figure 1.1

* + - Depending on your operating system, you might get a window with various options should appear next, here you can choose the way the program window will appear and the language in which you want massages to be displayed. Press ok.
    - A new window should automatically open, this is where you will be writing your programs (see Figure 1.2).

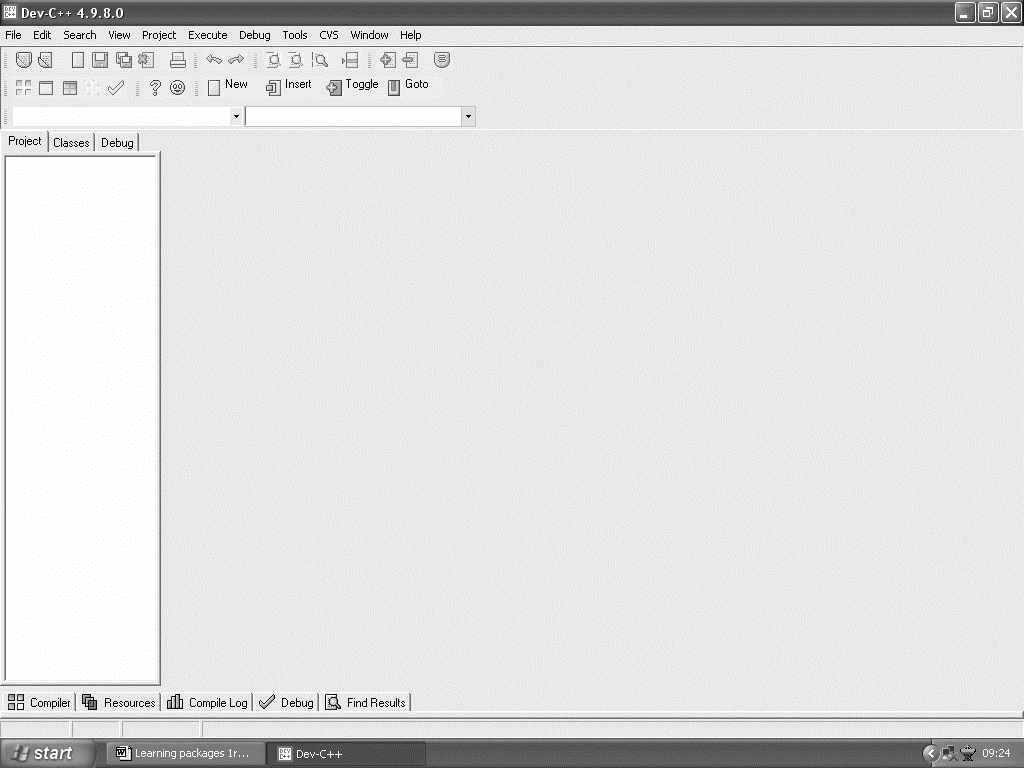


Figure 1.2

# Introduction to C

Read sections 2.1-2.3 on pages 27- 33 of the module textbook. This section discussed some of the basics of C programming, and explained some of the features.

# Functions

Read section 2.4 of the module textbook on pages 33-37. The program 2.3 in the module textbook showed the main() function being the first function defined in the program followed by the new function main\_menu(). We will use a different convention of defining all the new functions first, with the last function defined being the main() function.

#include <stdio.h> void main\_menu ()

{

printf("\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\n"); printf("\* Bridge Tutor \n"); printf("\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\n\n"); printf("1. shuffle\n");

printf("2. deal\n");

printf("3. display the hands\n"); printf("4. count the points\n");

printf("5. bid\n");

printf("6. play\n"); printf("q. quit\n\n\n");

printf("Press a number (1-6) or q to quit\n"); system("PAUSE");

}

void main()

{

main menu();

}

# Code to results

Read section 2.5 of the module textbook (pages 37-40).

Task 1.2

Now you have read about a C program, let us look at some of the commands and functions, and see how you can put what you just read about into action.

Task 1.2.1 Entering a program

* + - Open DevC++ 4

# File->New Project

* + - Click on Console application
    - Click on C project

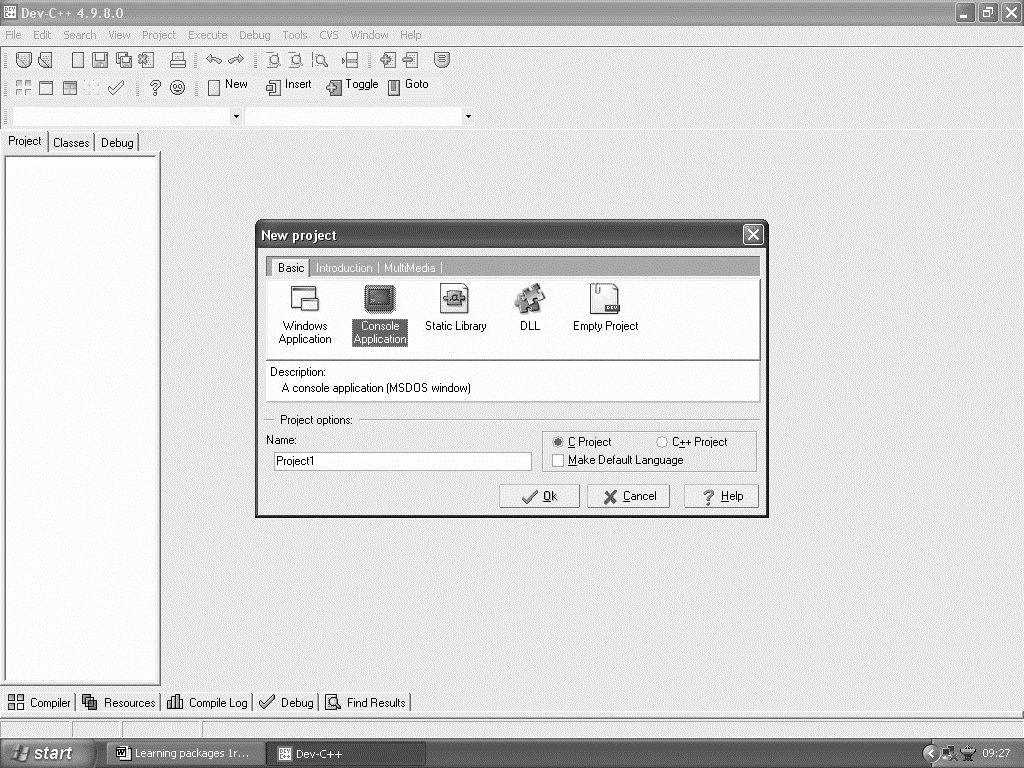


Figure 1.3

* + - Change the Project name to **program1**, then click on OK.

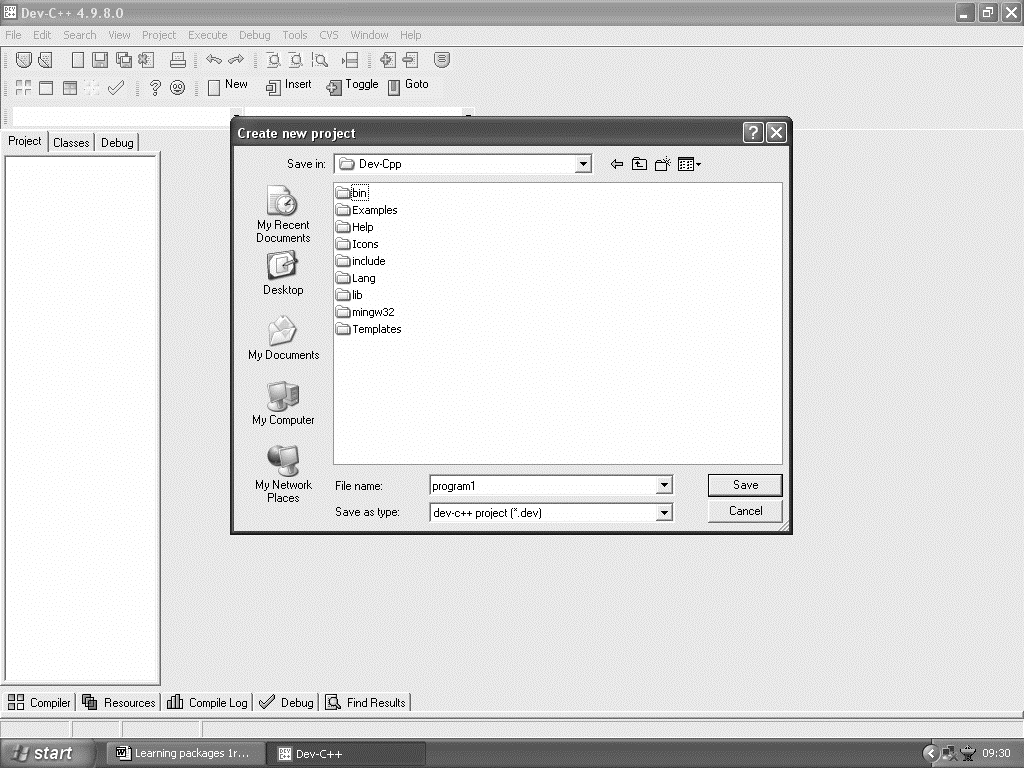


Figure 1.4

* + - Click on Save
    - Delete what is automatically produced for the moment, and type in the code below (a modified version of program 2.1 in the book)

# /\*greetings.c\*/

**/\*the first program – greetings\*/**

**#include <stdio.h> main ()**

**{**

**printf(“Welcome to “); printf(“the world of C.\n“);**

**}**

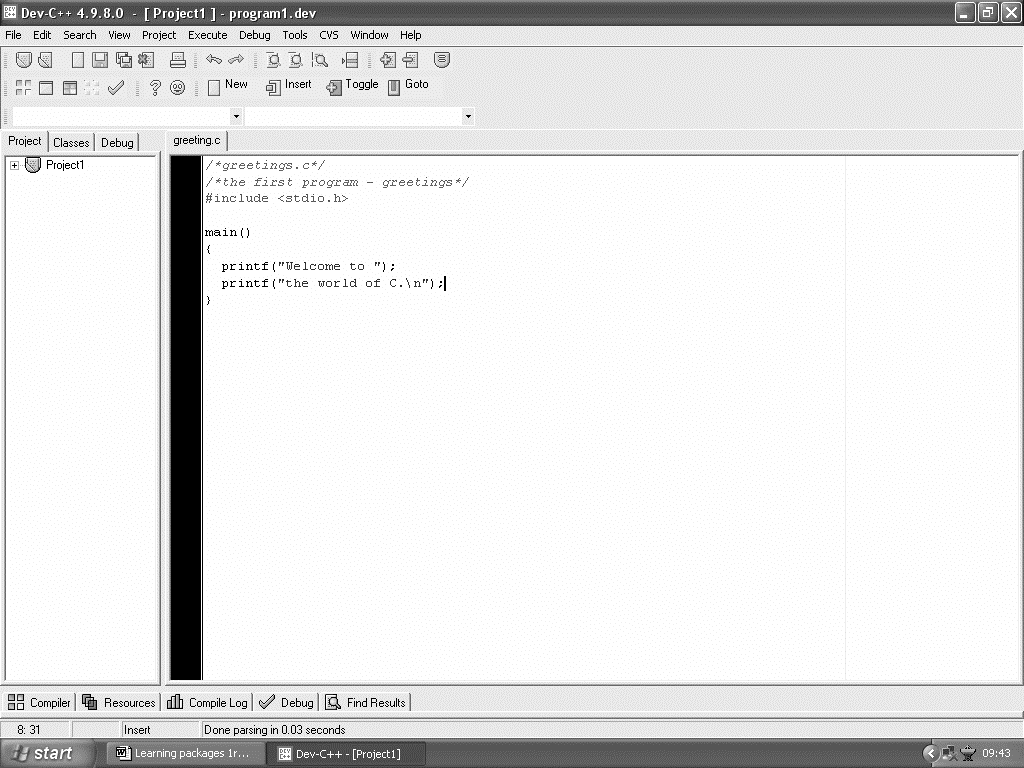


Figure 1.5

Task 1.2.2 Compiling a program

# execute->Compile

* type in **program1**, and click on save
* (on Windows XP) Program will start the compilation process, when it finishes it display a box on the screen saying Compilation complete.
* (On windows 98) Program will start the compilation process, when it finishes it display a box on the screen saying Done in a small window.
  + If there were no errors detected, click on the finish (or close) button on the screen, and follow the instructions under Using your program (section 1.2.3).
  + If errors were detected, then click on the finish (or close) button on the screen. Firstly, do not worry. The problems are likely to be relatively quick to fix, and spotting errors is good practice for when you write your own programs. With the aid of suggestions, (see end of the learning package), correct the program and repeat the instructions in this section, to compile the program.

Task 1.2.3 Using the program you have created

* minimise the DevC++ window

# program->MS-Dos Prompt

(on XP: All programsAccessoriesCommand prompt)

* change directory using **cd c:\Dev-Cpp**
* type in **program1** and press Enter

The program should now have been performed on the screen **Welcome to the world of C**.

Task 1.2.4 Using the Dev C++ environment

We can do execute programs from within the DevC++ program.

* Close the DOS window by click on the X in the top right corner.
* Bring back the DevC++ window.
* Go to ExecuteRun
* What happened the screen flash and it was difficult to see what happened.
* Alter the program (as shown below)

# /\*greetings.c\*/

**/\*the first program – greetings\*/**

**#include <stdio.h> main ()**

**{**

**printf(“Welcome to “); printf(“the world of C.\n“); system(“PAUSE”);**

**}**

* In the book there is a not a line system(“PAUSE”); this is add to programs in when using this compiler so when after the program you are writing has been performed the computer will wait for the user to press a key. This lets the user see what the program has done. We will consider this system () function later in the module
* Now recompile the code using **execute****compile and run**. This both compiles the code and executes it. If an error occurs whilst the code is being compiled, the program will not be executed.

# Exercises

Do exercises 1, 2, 3, 5, 6, 7, 8 on pages 41-42 from the module textbook.

* 1. **Summary**

See the summary on page 40 of the module textbook.

SAQ 1.3

What would you expect the following statements in a working C program to do?

* + 1. printf(“Hello\n\nWorld);
    2. /\*message\*/
    3. #include <stdio.h>
    4. menu\_1();

**Exercise answers**

1. /\*greetings.c\***/**

/\*the first program – greetings\*/

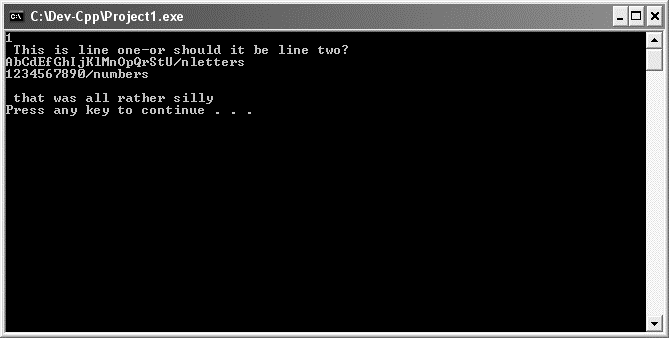
#include **<** stdio.h> ma**i**n ()

{

printf(“Welcome to “)**;** printf(“the world of C.\n“);

}

2.



3. The code as shown on page 41 of the module textbook will not compile. See the end of this learning package for help with spotting some of the errors in the code. Corrected code shown below

#include <stdio.h> main ()

{

printf("This is another very simple\n"); printf(" P\n");

printf(" R\n");

printf(" O"); printf(" GRAM\n");

}

1. With the Dev C++ package you can get away with missing out

#include<stdio.h>. This is not true for all versions of C compilers.

1. Example answer with my initials

/\*greetings.c\*/

/\*the first program - greetings\*/

#include <stdio.h> main ()

{

printf("\*\*\*\* \*\*\*\*\*\n");

printf("\* \*\n"); printf("\*\*\*\* \*\n"); printf(" \* \*\n"); printf("\*\*\*\* \*\n");

}

1. Example answer

/\*greetings.c\*/

/\*the first program - greetings\*/

#include <stdio.h> main ()

{

printf("\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\n"); printf("\*Scott Turner \*\n"); printf("\*University College Northampton\*\n"); printf("\*Avenue Campus \*\n"); printf("\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\n"); system("PAUSE");

}

1. Example

#include **<** stdio.h> void main\_menu ()

{

printf("\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\n"); printf("\* Bridge Tutor \n"); printf("\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\n\n"); printf("1. shuffle\n");

printf("2. deal\n");

printf("3. display the hands\n"); printf("4. count the points\n"); printf("5. bid\n");

printf("6. play\n"); printf("q. quit\n\n\n");

printf("Press a number (1-6) or q to quit\n");

}

void main()

{

system("cls"); main\_menu();

}

**Answers to SAQ**

SAQ 1.1: Why is programming about problem solving?

An example answer: A program is written to solve a problem. To write a program the problem needs to be understood, a solution needs to be devised, a solution needs to be built, and the final results to be looked at to see if it is a solution to a particular problem.

SAQ 1.2

Stepwise refinement is process of **breaking** a **problem** into **smaller** and **smaller** 'chunks'. The process continues until a **set** of **easily solved** sub- **problems** has been arrived at.

SAQ 1.3: What would you expect the following statements in a working C program to do?

* + 1. This prints Hello on the screen then prints a blank line and then puts World on the screen.
    2. It is a comment so the C compiler ignores it, but it is useful for explaining what the program does.
    3. Links the program that was typed in with the file stdio.h, which provides some of the functions that were used, by the programs typed in.
    4. This would be a call to a function menu\_1.

# General steps in compiling a program

Entering a program

* Open DevC++ 4
* File->New Project
* Click on Console application
* Click on C project
* Click on OK
* Change the Project name to a new name, then click on OK
* Change the directory to Dev-Cpp
* Click on Save
* type in your code

Compiling a program

* execute->Compile
* type in a new name for your program (this name must not be the same as the project name), and click on save
* Program will start the compilation process, when it finishes it display a box on the screen saying Compilation complete (or Done).
  + If the total error were zero, click on the finish (or Close) button on the screen, and follow the instructions under using your program (section 2.4.3).
  + If the total error were not zero, then click on the finish button on the screen. Firstly, do not worry the problems are likely to be relatively quick to fix, and spotting errors is good practice for when you write your own programs. With the aid of suggestions in the section, likely errors (see end of learning package). Correct the program and repeat the instructions in this section, to compile the program.

Using your program

* minimise DevC++ window
* **program->MS-Dos** Prompt
* change directory using **cd c:\Dev-Cpp**
* type in **program1**

**Likely errors**

Some common errors in C programming:

* Missing a semicolon (**;**) at the end of a statement.
* Using a bracket ( **(** or **)** ) instead of ( **{** or **}**) when defining the function.
* Missing off **#** in the line **#include <stdio.h>**
* Using brackets in **#include (stdio.h)** instead of **<** and **>** as in

# #include <stdio.h>

* Missing speech marks when using the **printf** function or using single quote marks when speech marks should be used.
* Typing **print** instead of **printf**.
* \ is a special character if it is not followed by another character such as n then the program may not work as you expect.
* Using /n instead of \n.