Lab Session – Introduction to C Programming

Objectives.

- Write simple programs in C

Aim

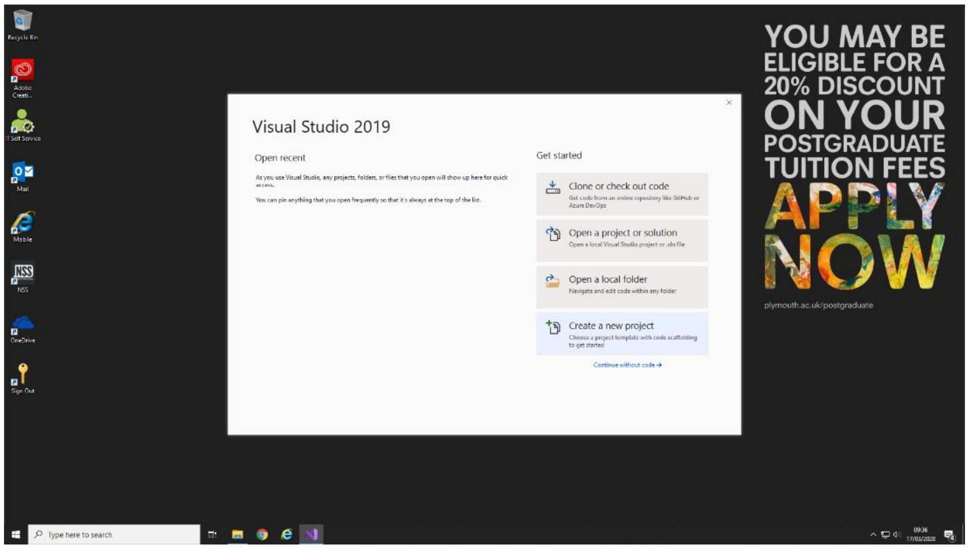
The **aim** of this lab session is to provide you with an introduction to C Language. For the rest of this module, we will be using C, as it is the dominant language for low-level programming and Operating Systems. The aim is not to become experts in C but get a basic understanding. C is highly portable and is a major part of Windows, UNIX, and Linux operating systems. C programs are much faster than equivalent programs in other languages such as C#, Java and Python.

Section 1 - How to Create a C Project in Visual Studio

Visual Studio does not support a separate template for C. **Note that C++ is a superset of C**. So, C++ supports everything that comes with C plus extra features such as Object-Oriented Programming, Exception Handling and a rich C++ Library. **So, we will create a C++ project and write C code**.

Open Visual studio and select ‘create new project’ as in Fig.1. Then, type ‘C++’ in the search area in the top and select the ‘empty project’ option (Fig.2). Specify the project’s name and directory and click ‘create’ button (Fig.3). Right click on the ‘source files’ and select ‘add’-> ’new item’ (Fig.4). Then select ‘C++ File (.cpp)’ and click ‘Add‘ button (Fig.5). You will be asked to give a name to the file.

Press F7 to build the code (this option is under the Build tab). Then, press Ctrl+F5 to run the program without debugging or press F5 to run with debugging.


Visual Studio 2019

Open recent

As you use Visual Studio, any projects, folders, or files that you open will show up here for quick access.

You can pin anything that you open frequently so that it's always at the top of the list.

Get started

Clone or check out code
Get code from an online repository like GitHub or Azure DevOps

Open a project or solution
Open a local Visual Studio project or .sln file

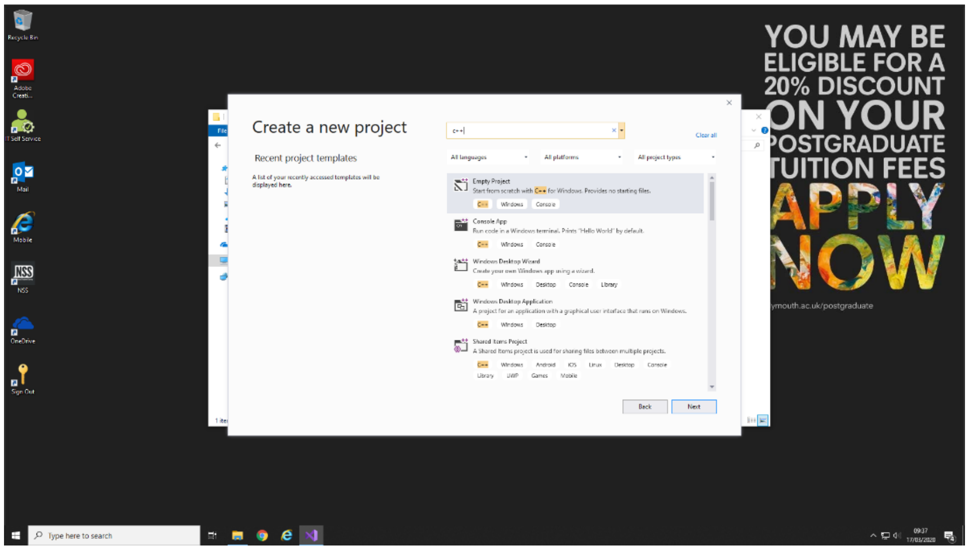
Open a local folder
Navigate and edit code within any folder

Create a new project
Choose a project template with code scaffolding to get started

Continue without code →

YOU MAY BE ELIGIBLE FOR A 20% DISCOUNT ON YOUR POSTGRADUATE TUITION FEES
APPLY NOW
plymouth.ac.uk/postgraduate
Type here to search

Fig.1. How to create a C++ Project Step1


Create a new project

Recent project templates

A list of your recently accessed templates will be displayed here.

c++

Clear all

All languages | All platforms | All project types

Empty Project
Start from scratch with C++ for Windows. Provides no starting files.
C++ Windows Console

Console App
Run code in a Windows terminal. Prints "Hello World" by default.
C++ Windows Console

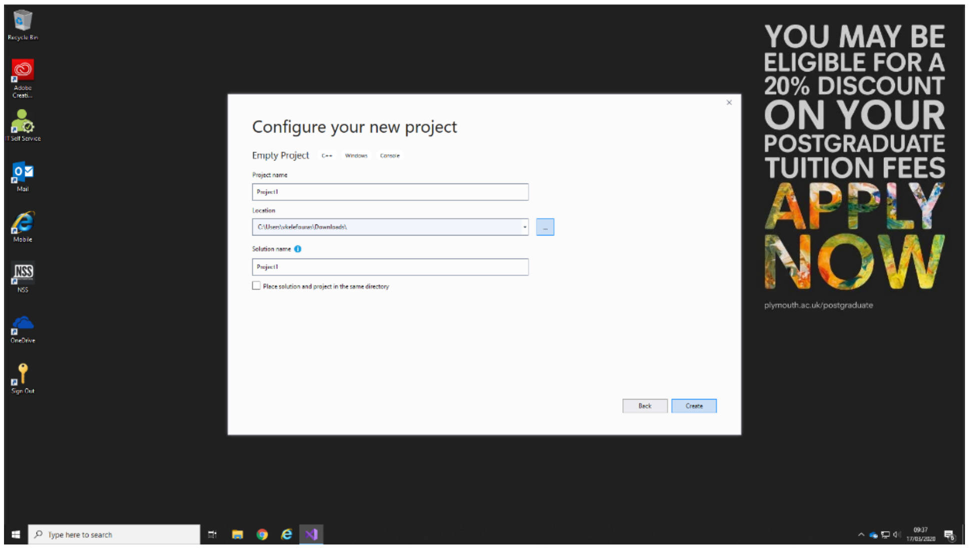
Windows Desktop Wizard
Create your own Windows app using a wizard.
C++ Windows Desktop Console Library

Windows Desktop Application
A project for an application with a graphical user interface that runs on Windows.
C++ Windows Desktop

Shared Items Project
A Shared Items project is used for sharing files between multiple projects.
C++ Windows Android iOS Linux Desktop Console Library UWP Games Mobile

Back
Next
YOU MAY BE ELIGIBLE FOR A 20% DISCOUNT ON YOUR POSTGRADUATE TUITION FEES
APPLY NOW
Type here to search

Fig.2. How to create a C++ Project Step2


YOU MAY BE ELIGIBLE FOR A 20% DISCOUNT ON YOUR POSTGRADUATE TUITION FEES
APPLY NOW
plymouth.ac.uk/postgraduate
Back
Create

Configure your new project

Empty Project C++ Windows Console

Project name: Project1

Location: C:\Users\vkelefouras\Downloads\

Solution name: Project1

☐ Place solution and project in the same directory

Fig.3. How to create a C++ Project Step3

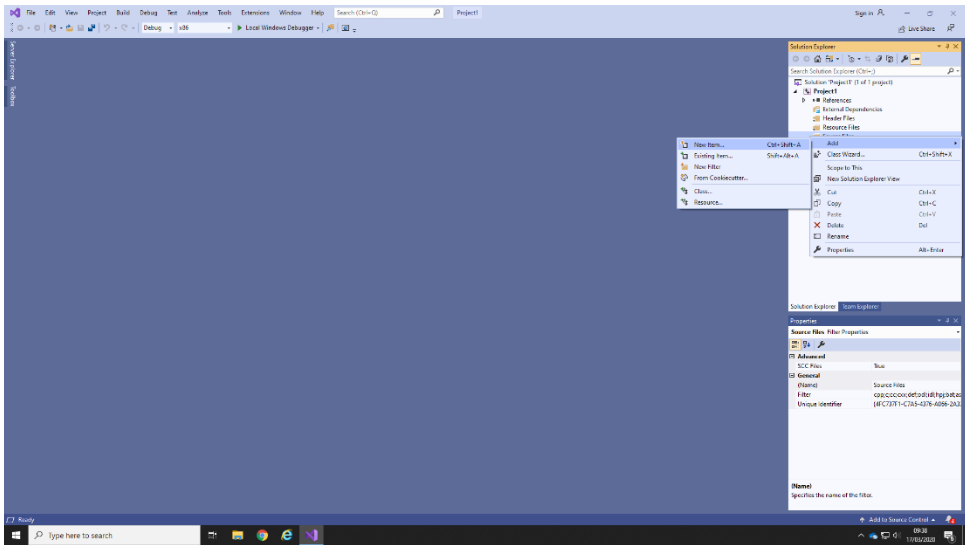
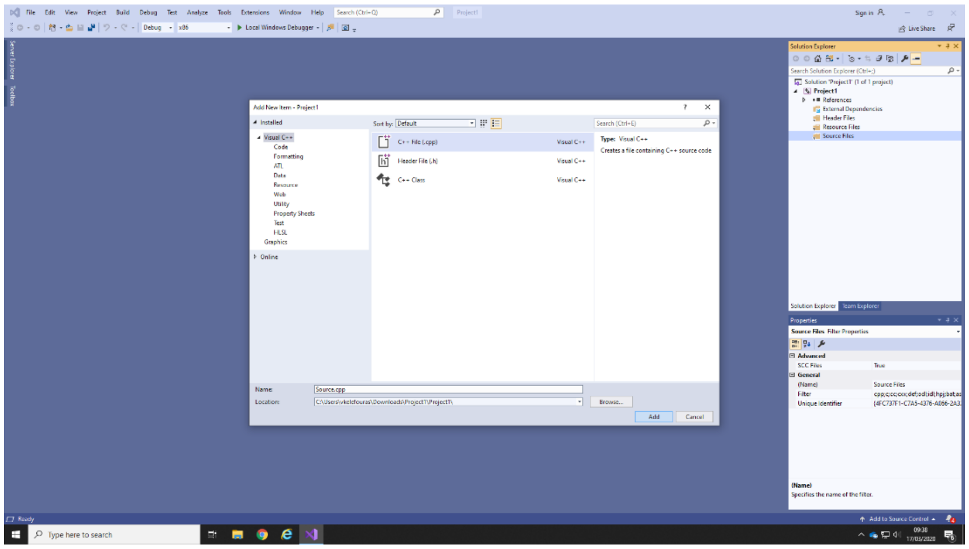


Fig.4. How to create a C++ Project Step4



Add New Item - Project1

Type: Visual C++
Creates a file containing C++ source code

Name: Source.cpp
Location: C:\Users\vkelefouras\Downloads\Project1\Project1\

Fig.5. How to create a C++ Project Step5

Section 2 – C Examples

Task1 (Data Types). Compile and run the ‘***datatypes.cpp’*** file. Make sure you understand every line of the code.

Task2. **Array Addition**

Write a C program which adds two arrays and stores the result into another array, and print the results. A solution is found in ‘***vector_addition.cpp’*** file.

Task3 (Reading from input using scanf_s function). In this task, you will be using the program in ‘***scanf.cpp’*** file.
In C programming, a string is a sequence of characters terminated with a null character ‘\0’. For example:

char A[] = "Hi there";

When the compiler encounters a sequence of characters enclosed in the double quotation marks, it appends a null character ‘\0’ at the end by default. So, the A[] array’s elements are : A[0]=’H’, A[1]=’i’, …, A[8]=’e’, A[9]=’\0’.

Scanf() is perhaps the easiest way to read a string. The scanf() function reads the sequence of characters until it encounters whitespace (space, newline, tab, etc.). Scanf function is not a safe function (note hackers take advantage of such unsafe functions) and this is why Visual Studio does not allow us to use it. Instead Visual Studio allows us to use its safe equivalent which is scanf_s. You will learn more about this topic next year.

The scanf_s command is as follows

scanf_s(“%19s”, name, sizeof(name));

The first operand specifies the type of the expected input. %19s means that we are expecting a string, up to 19 characters long. The second operand is the memory location where the input string will be stored to and the third operand is the maximum size of the input. If we enter more characters than this they will be rejected.

Task4: Find the frequency of a character into a string.

You can find this program in ‘***frequency.cpp’*** file. The ‘string’ array is a large input array which is initialized by the user using scanf_s function (see previous task). The ‘pattern’ array is an array which contains two characters. In C, the last character of an array is always the zero character, therefore this array contains just a single character. This program finds how many times this character occurs in the ‘string’ array. Given that the ‘pattern’ array contains just one character, there is no need to use an array here; a char variable could have been used instead.

In line 18, the ‘string’ array is read, while in line 21, the ‘pattern’ array is read from the keyboard. Note that the ‘pattern’ array consists of a single character (the 2nd is always the zero character); this is why the 1st operand of scanf_s is %1s. Thus, if we type more than one characters, they will be discarded.

In line 23, there is a for loop which compares all the ‘string’ characters with the ‘pattern’ character. If the string character is the pattern character, then the counter variable increases its value.

Task5. Write a C program which calculates the length of a string. The solution is provided in ‘array_length.cpp’ file.

Task6. Print the array’s memory addresses

Every variable is stored into a memory location and every memory location has a memory address. A memory address can be accessed by using the ampersand (&) operator. Consider the following example

```
#include <stdio.h>

 int main () {

   int  var1=4;

   printf("The memory address of %p contains %d \n", &var1, var1  );

   return 0;

}
```

The code above prints:

The memory address of 0x7ffdd32f4b9c contains 4.

Keep in mind that the memory address that var1 is stored might be different every time you compile the program. To print the memory address of a variable, the ‘%p’ format specifier is used. ‘%p’ is a format specifier which is used if we want to print data of type (void *) or in simple words address of a variable in this case. The output is displayed in hexadecimal value. However, the memory addresses can be printed in integer format too, if we use ‘%d’ instead of ‘%p’ ; in this case, a warning will be shown ‘*warning: format ‘%d’ expects argument of type ‘int’, but argument 4 has type ‘int *’, which can be ignored*.

The ‘***print_array_memory_addresses.cpp***’ file prints the memory addresses of an array. The memory addresses are printed in hex format. How many bytes are being allocated for each element? The answer is 4bytes, as the data type is of type ‘int’ which is four bytes. As you can observe, the array’s elements are stored into consecutive memory locations. Change the data type of the array from ‘int’ to ‘short int’ and then into ‘long long int’. Are the memory addresses different now? Why? Given that ‘short int’ occupies 2 bytes, each array element needs 2 bytes.

Task7. Write a program that finds the maximum element of an array. A code solution is found on ‘***max_value_solution.cpp’*** file.

Task8. Write a program that finds the maximum and minimum elements of an array. A code solution is found on ‘***maxmin_value_solution.cpp’*** file.

Further Reading

If you want to learn more about C programming you can study the following links:

1) Tim Bailey, 2005, An Introduction to the C Programming Language and Software Design, available at: http://www-personal.acfr.usyd.edu.au/tbailey/ctext/ctext.pdf
2) C examples, Programiz, available at https://www.programiz.com/c-programming/examples
3) C Programming examples with Output, Beginners book, available at https://beginnersbook.com/2015/02/simple-c-programs/
4) C Programming Tutorial, from tutorialspoint.com, available at https://www.unf.edu/~wkloster/2220/ppts/cprogramming_tutorial.pdf