CSC2040 Assessed Practical 2

Wednesday 13th November 2019

Before you begin this test please read carefully and follow instructions. As you write programs for the test, saving as you go, the following procedure will ensure that if your computer crashes or if you upload the incorrect files your work will not be lost and can be restored.

1 Log on to the machine in the usual way.

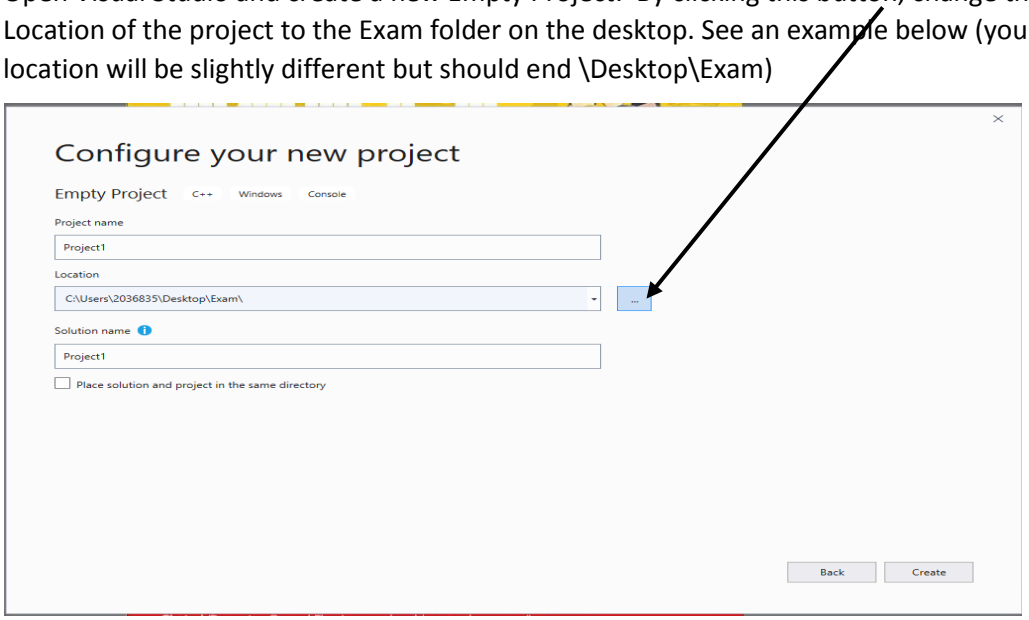
2 Using File Explorer  locate the S:\ drive (public_shares (\\nas02.eeecs.qub.c.uk).

3 In the folder CSC2040 Exam 1920 you will find an application:

1 Create_Exam_Folder

4 Run Create_Exam_Folder by double clicking on it and then clicking Run. An exam folder appears on the desktop. Press any key to exit the blue screen.

5 Open Visual Studio and create a new Empty Project. By clicking this button, change the Location of the project to the Exam folder on the desktop. See an example below (your location will be slightly different but should end \Desktop\Exam)



Name the Solution Test2 and the first program Project1.

For the questions below, look for two files fragments.txt and Shape.h. The files includes the various fragments of supplied code mentioned in the questions below, to save you typing them out.

Project 1 Copy Constructor (25 marks)

Inside Project1 follow Steps 1-3 below to create a C++ program:

1. Create a header file "RandArray.h", in which you declare a class `RandArray` (see fragments.txt):

```
class RandArray
{
public:
	// constructor - allocate data array with length = len, and fill in the array
	// with random numbers within the range [min, max]
	RandArray(int len, int min, int max);
	// destructor - free data array
	~RandArray();
```

```
        // print - print values in data array
        void print();

private:
        int* data;    // data array
        int length;   // array length
};
```

2. Create a source file “RandArray.cpp”, in which you type the code below to implement this class (see fragments.txt):

```
RandArray::RandArray(int len, int min, int max)
        : length(len)
{
        if (len < 0) {
                data = NULL;
                return;
        }
        else {
                data = new int[length];
                for (int i = 0; i < length; i++)
                        data[i] = rand() % (max - min + 1) + min;
        }
}

RandArray::~RandArray()
{
        if (data) delete[] data;
}

void RandArray::print()
{
        if (data) {
                for (int i = 0; i < length; i++)
                        cout << data[i] << " ";
                cout << endl;
        }
}
```

3. In Project1 create a source file “Project1.cpp” to include the program below for testing the class:

```
int main()
{
        RandArray x(10, -5, 5);
        // print x three times
        for (int n = 0; n < 3; n++) {
                RandArray y = x;
                y.print();
        }

        return 0;
}
```

Problem:
Run the above program and you will see that it prints out x not three times but only once and then crashes. You are asked to add a copy constructor for the class to fix this problem.

Project 1 (continued) Operator Overloading (25 marks)

Further, in the given `RandClass` add a new operator *= function. This function is applied to two objects of the class to perform, e.g., `x *= z`, which multiplies the `data` array of object `x` by the

`data` array of object `z`, element by element till the end of the arrays. If the two `data` arrays have different lengths, then the shorter `length` is used to stop the loop of the element-to-element multiplication.

To test your function, in the above main function before `return 0` type the following code:

```
x *= x;
x.print();
```

Make sure the program produces the results as you have expected.

Project 2 Template Function (25 marks)

The following shows a C++ function `remove` (see fragments.cpp) that is a modified function from Practical 2. The function takes an integer `array` with `array_len` elements and removes the element at the position `pos`. Upon finishing the operation, the function returns the pointer to the value removed, and the calling arguments `array` and `array_len` will have been modified accordingly.

```
int* remove(int*& array, int& array_len, int pos)
{
	// test
	if (!array || array_len < 1) {
		cout << "invalid array" << endl;
		return NULL;
	}
	if (pos < 0 || pos >= array_len) {
		cout << "pos is out of range" << endl;
		return NULL;
	}

	// value to be removed
	int* value_removed = new int;
	*value_removed = array[pos];

	// new array length after removal
	array_len--;
	// new array
	int* temp = new int[array_len];
	for (int i = 0; i < pos; i++) temp[i] = array[i];
	for (int i = pos; i < array_len; i++) temp[i] = array[i + 1];

	// delete original array
	delete[] array;
	// set the new array
	array = temp;

	return value_removed;
}
```

Problem:
You are asked to convert the above function into a template function so that it can be used to remove element for other types of arrays. Follow these instructions for the conversion: (1) in Solution Test2 add a new project Project2 and create a new source file Project2.cpp, (2) type or copy the above code into Project2.cpp and start conversion, (3) create a main function below the template function to test it:

1. Declare an `int` variable `array_len` and initialize it to a value of 5.
2. Allocate a dynamic `string` array `array` of size `array_len`, initialise the array with the following elements, and print all the values in this initial array:

A `string` array with 5 elements

abc
def
ghi
jkl
mno

3. Call the template function `Remove` to remove the value at `pos = 3`. Use the returned pointer to print the value removed. Finally print all the values in the `array` after removal.
4. Free the memories allocated in above operations as appropriate, to avoid memory leak.

Project 3 Inheritance and Virtual Functions (25 marks)

In Solution Test2 add a new project Project3. Download the provided file `Shape.h`, place it into the Project3 folder and add it into Project3. `Shape.h` combines class inheritance and virtual functions with the aim to implement a form of "one interface, multiple methods" - polymorphism, for calculating the areas and perimeters of different geometric shapes. It currently supports the calculations of three different shapes: rectangle, circle and right triangle.

Problem:
In Project3, create a Project3.cpp in which you write a main function to demonstrate how to use `Shape.h`. In the main function, you can use the following statements as the interface with the user to read shape parameter data from the keyboard, to calculate the shape area and perimeter with the given data, and to print out the results, where `base_ptr` is a base-class pointer. These statements remain the same for different shapes to calculate and hence we achieve the so-called polymorphism.

```
base_ptr->read_shape_data();
base_ptr->compute_area();
base_ptr->compute_perimeter();
base_ptr->print_result();
```

Your program should allow the user to choose which shape to calculate from the keyboard.

Submission Notes

1. Save your project.
2. Make sure you have the following **FIVE** source files ready for submission:

   RandArray.h, RandArray.cpp, Project1.cpp
   Project2.cpp
   Project3.cpp
3. Locate your Solution folder Test2 containing all your source files. ZIP this folder (by right clicking the folder and choosing 'Send To' and selecting Compressed (Zipped) folder). Rename the zip file to XXXXXXX.zip, where XXXXXXX is your student number.
4. Check that your .zip file contains your source files (double click on the .zip file. You do not need to extract the files again).
5. In case something goes wrong with your submission, go to the folder CSC2040 Exam 1920 and find the application:

   2 Backup_Exam_Folder

   Double click on it. You can make a backup of your zip file on to a USB drive, and/or email it to yourself.
6. On the course's Assignments page for Practical Test 2 on Canvas choose Submit Assignment, followed by File Upload. Click Choose File and locate the zip file you have just created. Click Submit Assignment again.