CSC2040 Assessed Practical 1

Wednesday 23rd October 2019

For the questions below, you are given a file "fragments.txt". The file includes the various fragments of supplied code mentioned in the questions below, to save you typing them out.

In Visual Studio create a new Empty Project. Name the Solution Test1 and the first program Project1. Find a suitable Location to save your program.

Project 1 Functions and parameters (30 marks)

In Project1, using a header file (`my_functions.h`) for the declaration and a source file (`my_functions.cpp`) for the implementation develop two functions, detailed below. Additionally, create a source file `Project1.cpp` to host the main function of the project for testing.

1. Write a function `most_least_diff(double, a, double b, double c)`. The function take three `double` numbers `a`, `b` and `c` as parameters. It will find, among the three given numbers, the pair of numbers that are most different, and the pair of numbers that are least different, respectively. The difference between two numbers `a`, `b` is measured by their absolute difference `fabs(a-b)`, a system function available by `#include <iostream>`. Your function should print out the two pairs of numbers found along with their corresponding differences, for example (see "fragments.txt"):

   ```
   cout << "Most different: " << max_diff_num1 << " " << max_diff_num2 << " with
   difference = " << max_diff << endl;
   cout << "Least different: " << min_diff_num1 << " " << min_diff_num2 << " with
   difference = " << min_diff << endl;
   ```

   where (`max_diff_num1, max_diff_num2`) and (`min_diff_num1, min_diff_num2`) are the two pairs of numbers found, with maximum difference `max_diff` and minimum difference `min_diff`, respectively.

   In Project1.cpp run some tests on your function.

2. Write a function `admission` that automatically examines and charges credit points to grant or deny some admission. The function returns a Boolean value, true for admitted and false for denial. The credit points (an integer value) are passed to the function as a parameter. Assume each admission costs 3 credit points, and admission will be denied without sufficient credit points. Run the following test code to show that with 7 credit points one can gain two admissions, and have 1 point left afterwards (see "fragment.txt").

   ```
   int credits = 7;
   int num_admissions = 0;
   while(admission(credits)) num_admissions++;
   cout << "number of admissions " << num_admissions << endl;
   cout << "credit points left " << credits << endl;
   ```

Project 2 Arrays, Pointers and Dynamic Allocation (35 marks)

In the Solution Test1 add a new project Project2, in which create a new source file `Project2.cpp` to hold your code solutions to the following questions.

1. In C++, the name of an array is a pointer to its first element. Therefore a function having an integer array of `size` elements as the parameters can be declared in either of the two forms:

```
void array_op(int array[], int size)
void array_op(int* array, int size)
```

Because of this array-pointer equivalence, one can access the array elements by using array indexing, or alternatively by using pointer arithmetic.

In `Project2.cpp`, type the following `array_op` function with operations (1)-(2) completed (see "fragments.txt"). Complete the remaining operations marked by (3)-(5).

```
void array_op(int* array, int size)
{
   // (1) test
   if (array == 0 || size <= 0) return;

   // (2) use array indexing to cycle through the elements of array to assign each
   // element a value
   for (int n = 0; n < size; n++)
      array[n] = n;

   // (3) use array indexing to cycle through the elements of array to print out the
   // value of each element

   // (4) use pointer arithmetic to cycle through the elements of array to double the
   // value in each element

   // (5) use pointer arithmetic to print out the value of the element that is
   // 1/3-size away from the beginning
}
```

Following the above `array_op` function write a short main function with the following code to test the `array_op` function:

```
int array[100];
array_op(array, 100);
```

2. In the same Project2.cpp, rename the above `main()` function to `main1()`, and then create a new `main()` for the following test.

Write code into `main()` to implement the following operations:

(a) Declare a `double` variable `pi`, assign it the value 3.1415926, and then declare a pointer `ppi` to point to `pi`.

(b) Declare a `double` array `array`[40], and declare a pointer `parray` to point to `array`[40]. Then, use `parray` to cycle through the elements of `array`[40] to set all the elements to the value pointed to by `ppi`. In the last assignment operation, only `parray` and `ppi` can be used.

(c) Allocate a double array `array2` of 40 elements by using the `new` operator, and copy the values of `array` to `array2`, element by element. Then, print out all the values in `array2`.

(d) To finish, to avoid memory leak, use `delete` or `delete[]` as appropriate to delete the appropriate pointer(s) to free the unmanaged memories used in the above operations.

Project 3 Classes (35 marks)

In the Solution Test1 add a new project Project3. In the project create a C++ class named `RandArray`. The class has the following structure of member data and functions (see "fragments.cpp"):

```
class RandArray {
private:
   int* data;    // private data, an integer array
   int length;   // private data, length of the data array

public:
   // constructor, which sets length to the given len, allocates memory
   // for the data array to the given length, and sets all elements of the
   // array to random numbers between 0 and max_num-1 inclusive, by calling
   // function rand() % max_num available by #include <iostream>
   RandArray(int len, int max_num);
   // destructor, to free the memory allocated for the data array
   ~RandArray();

   // function, to print the data array
   void print_data();
};
```

Create the class in two separate files, and then create a test main function, as detailed below.

1. First, create a header file `RandArray.h` for the class, which includes the declaration of the class, as above.
2. Second, create a source file `RandArray.cpp` for the class, which includes a suitable implementation of all the member functions of the class, including: (1) the constructor, (2) the destructor, and (3) the `print_data` function, as defined above.
3. Finally, in Project3 create a test program `Project3.cpp` to include the main function for the following test:

   (a) Declare an object of the `RandArray` class. The object has a data array with length = 100 to hold 100 random numbers between 0 and 25 inclusive. Print the random numbers held by this object by applying the `print_data` function to this object.

   (b) Use `new` to declare a pointer to create a dynamic object of the class, which has a data array with length = 50 to hold 50 random numbers between 0 and 10 inclusive. Use the pointer to access the `print_data` function of the object to print out its data. After this operation, delete the pointer to free the memory used by the object.

   (c) Further in the main function declare two arrays of the `RandArray` objects such as the following:

   ```
   RandArray array1[5];
   RandArray* array2 = new RandArray[5]; // a dynamic array created using new
   ```

   You will receive compilation errors. Solve this problem by adding an appropriate constructor into the above RandArray class definition (RandArray.h and RandArray.cpp).

Submission Notes

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

   Project1.cpp, my_functions.h, my_functions.cpp
   Project2.cpp
   Project3.cpp, RandArray.h, RandArray.cpp
3. Locate your Solution folder Test1 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, first 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 1 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.