Lab 3 [ECE 2620 (C++, Data Structures & Algorithms)]

C++ Classes, Constructors & Templates

Problem Description and Design

In lab exercise 2, you wrote a C++ program using the `circle` class, and its member functions. In this lab exercise, you will refine the same program, using templates and a class constructor.

You will use templates to redesign the class circle so that its four **`private`** data members `x1`, `y1`, `x2`, and `y2` can be of any numeric type (namely, `int`, `float` or `double`). In addition, the **`protected`** member function `distance`, and the **`public`** member functions `radius`, `circumference` and `area` need to be rewritten as well using templates so that they can now work with any numeric data type. Regardless of the input data type, the computed results i.e. distance, radius, circumference and area are always double precision floating point quantities. In addition, you will write a default class constructor (with all default parameters), using templates, to initialize the `x1`, `y1`, `x2` and `y2` class variables. All data members of the class object *should be set to zero* if the programmer does not pass explicit values for them when the object is first declared. Note that you will still need to retain the **`public`** member function `populate_classobj`. since you will want to modify the contents of the data object during run-time with user-supplied values.

Your job is to do the following:

1. Define the class `circle` with templates as described above.
2. Write the member functions using templates as described above.
3. In `main()`, declare two objects belonging to the class `circle` called `my_obj1` and `my_obj2`.

   `my_obj1` has integer data members while `my_obj2` has double precision floating point data members.

   In the declaration of `my_obj1`, set its `x1` and `y1` values to 1 and 3, respectively. Leave the `x2` and `y2` values unspecified.

   When declaring `my_obj2`, set its `x1`, `y1`, `x2`, `y2` values to -1.5, -6.65, -0.5, 10.0, respectively.
4. A sample run of your program will consist of providing the user the following options:

   (1) Compute the radii of the two circles
   (2) Compute the circumferences of the two circles
   (3) Compute the areas of the two circles
   (4) Enter new x1, y1, x2, y2 values for Object 1
   (5) Enter new x1, y1, x2, y2 values for Object 2
   (6) Exit

   Option (4) prompts the user to enter the new ***x*** and ***y*** integer co-ordinates of the center of the circle, and a point on that circle. Store the co-ordinates in `my_obj1`.

   Option (5) prompts the user to enter the new ***x*** and ***y*** double precision floating point co-ordinates of the center of the circle, and a point on that circle. Store the co-ordinates in `my_obj2`.

   Recall that you need to use the member function `populate_classobj`.
6. Once a response is received, the program must then proceed appropriately and display the above menu again (unless the user wishes to exit the program).

NOTES:

1. For the above program, note that the member functions `distance`, `radius`, `circumference` and `area` must return double precision floating point values even when `x1`, `y1`, `x2`, and `y2` are all integers.
2. The syntax for defining a member function using templates is:

```
template<class T>
return_type class_name<T>::function_name(..)
  {
    ...
  }
```

   where,

   return_type can be `T` or `double` (see note 1).

3. Document your code well. By this, I do not mean that you should write comments for every line of code. Any time you write a piece of code or a function that does something of significance, use comments to explain what that bit of code does, logically (conceptually) speaking.
4. Keep your source program neat and modular. Use functions, instead of writing a monolithic program.
5. Start every program (main.cc) with

```
//Your name here
//Course + section number
//  Anything else you may wish to tell me
```

   These comments at the very beginning of your program, help us to identify you, and see if you have anything to say to me such as instructions on how to run the program, anything that does not work in your program, anything that does work(!), etc.