CS 435/535 Assignment #5 – Spring 2023

Project Overview: This project is to investigate polymorphism in C++. In C++, polymorphism allows the `main` function to be written before an actual class is defined. In this project, you will download from Blackboard the file `main.cpp` that contains the `main` function and the two files (`Shape.h` and `Shape.cpp`) that define a base class named `Shape` to represent a 3D shape. The `Shape` base class has a shape name represented by a private instance variable called `name`, but it does not have any other details on the shape. However, the base class has four (4) pure virtual functions: `getArea` (to compute the surface area of the shape), `getVolume` (to compute the volume of the shape), `test` (to test whether the shape satisfies a list of conditions), and `getInfo` (to form a string that describes the shape). You are asked to write four (4) derived classes: `Sphere`, `Cylinder`, `Torus`, and `Box` to represent four actual 3D shapes.

- A sphere is defined by its radius.
- A cylinder is defined by the radius of its base and its height.
- A torus (aka donut) has two radii, small and big. A torus is formed by rotating a small circle (defined by the small radius) about an axis on the same plane. The center of the rotating circle forms a bigger circle (defined by the big radius). (To compute the surface area and the volume correctly, the small radius will not be bigger than the big radius.)
- A box has length, width, and height.

For each derived class, you need to have one header file (with the `.h` extension) and one implementation file (with the `.cpp` extension). However, you can use any name for the file, any name for the class, and any names for the instance variables. The `main` function does not need to know the class names you will use. The only requirement is that each derived class must be derived from the `Shape` base class, which will force you to implement the four virtual functions in the derived class. *Polymorphism enables the `main` function to invoke a function of a derived class in the future by just calling the corresponding virtual function of the base class.*

To test the program, we assume that the shapes will be stored in a data file that will be passed to the program as a command line argument. Each shape will occupy one line in the data file. Each line starts with the `<name>` that is a string, followed by the type (`sphere`, `cylinder`, `torus` or `box`). The rest will be doubles to represent the dimensions of the shape.

```
<name> sphere <radius>
<name> cylinder <radius> <height>
<name> torus <small_radius> <big_radius>
<name> box <length> <width> <height>
```

In principle, you can use any file format as long as you can present a list of shapes to the `main` function. However, in order to test everyone's program using the same data set, you have to prepare the data files using the above format. You can download a sample data file named `shapes.dat` from Blackboard for an example. It contains the following shapes.

```
Cube#1 box 1 1 1
Cube#2 box 2 2 2
Donut#1 torus 1 1
Cyl#1 cylinder 1 1
Case#1 box 2 4 6
Case#2 box 10.5 21 10.5
UnitSphere sphere 1
LargeSphere sphere 100
Donut#2 torus 3 7
Cyl#2 cylinder 1 2
```

You need to write a function named `readFrom` that will return a pointer to a vector of shapes read from a data file with its name given as a parameter to the function. The function has the following signature.

```
std::vector<Shape*> *readFrom(std::string);
```

The signature is stored in a file named `reading.h` that you can also download from Blackboard. You need to create a `.cpp` file to implement the function. You can assume the data files to test your program will always follow the correct format as described above. However, if the file can't be opened for reading, you shall return a pointer to an empty vector.

As mentioned earlier, you need to implement four virtual functions in each of four derived classes. Please search to find out how to compute the surface area and the volume of each shape. For the `getInfo` function, please see the sample execution at the end on how to format the output string. The `test` function is to test whether the shape satisfies a list of conditions. The list of conditions is passed to the function as a parameter, and it is a vector of strings. The number of strings in the vector will always be a multiple of 3. Three consecutive strings form a test condition in the `<name> <op> <value>` form. The `<name>` string will be "type", "area" or "volume". The `<op>` string is one of the six relational operators ("==", "!=", ">=", "<=", ">", and "<"). The `<value>` string is the reference value in the string format to be compared with. For the `test` function to return `true`, all the test conditions have to be true. For example, if the string vector contains 9 strings: "type" ">" "cyl" "area" "<=" "1000" "volume" ">" "100.5", the function returns `true` if the shape type is greater than "cyl" (i.e. any shape type but box) and the surface area is less than or equal to 1000 and the volume is greater than 100.5. If the vector is empty (containing no testing conditions), the function will always return `true`. Please see the `getTestConditions` function in `main.cpp` to see how a list of test conditions is built.

In this project, you need to create a `makefile` to compile your program as we do not know the file names you will use. Please use the `-Wall -std=c++11` options when compiling your program with g++. The executable of this program shall be named as `a.out` regardless of the operating system you are using (using the `-o a.out` option with g++). Your `makefile` shall support the commands "`make`" to produce `a.out`, and "`make clean`" to remove `a.out` and any object files (with the `.o` extension).

What You Need To Do

- Create a directory named **`project5`** on your machine. Download **`main.cpp`**, **`Shape.h`**, **`Shape.cpp`** **`reading.h,`** and **`shapes.dat`** from Blackboard to the **`project5`** directory. You can make changes to **`shapes.dat`** to add more shapes, but make no changes to the other C++ files. (In reality, you will be given the object files with the .o extension instead of the source code. For simplicity, you are given the source code, and your job is to compile them into the object files based on your operating system as if you were given the object files. Therefore, do not change any downloaded C++files.)
- Add four (4) derived classes as requested. Each class shall have its own header and implementation. *You can add more derived classes for additional shapes and the `main` function will work with them correctly without any changes.*
- Create a `.cpp` file to implement the `readFrom` function, based on its signature in **`reading.h`**.
- Create a **`makefile`** that supports the commands "`make`" to produce `a.out`, and "`make clean`" to remove `a.out` and any object files. Your `makefile` will be used to compile your program to produce `a.out`, and `a.out` will be used to test your program.
- When you are ready to submit your project, compress your **`project5`** directory into a single (compressed) zip file, **`project5.zip`**. **No other compressed files will be accepted**.
- Once you have a compressed zip file named **`project5.zip`**, submit that zip file to Blackboard.
- Your submission will be graded on `cs-parallel.ua.edu`. Make sure to test it on that machine before submission.
- Make sure to follow the above instructions exactly. Otherwise, we may not be able to grade your submission.

Assignment #5 is due at 11:59pm on Wednesday, March 22. Late projects are not accepted.

A sample execution of the program

```
$ ./a.out shapes.dat

Enter a command: print
Box: Cube#1, Length=1.00, Width=1.00, Height=1.00
        Surface Area: 6.00, Volume: 1.00
Box: Cube#2, Length=2.00, Width=2.00, Height=2.00
        Surface Area: 24.00, Volume: 8.00
Torus: Donut#1, Small Radius=1.00, Big Radius=1.00
        Surface Area: 39.48, Volume: 19.74
Cylinder: Cyl#1, Radius=1.00, Height=1.00
        Surface Area: 12.57, Volume: 3.14
Box: Case#1, Length=2.00, Width=4.00, Height=6.00
        Surface Area: 88.00, Volume: 48.00
Box: Case#2, Length=10.50, Width=21.00, Height=10.50
        Surface Area: 1102.50, Volume: 2315.25
Sphere: UnitSphere, Radius=1.00
        Surface Area: 12.57, Volume: 4.19
Sphere: LargeSphere, Radius=100.00
        Surface Area: 125663.71, Volume: 4188790.20
Torus: Donut#2, Small Radius=3.00, Big Radius=7.00
        Surface Area: 829.05, Volume: 1243.57
Cylinder: Cyl#2, Radius=1.00, Height=2.00
        Surface Area: 18.85, Volume: 6.28

Enter a command: print2
Enter test condition #1: type == box
Enter test condition #2: area >= 88
Box: Case#1, Length=2.00, Width=4.00, Height=6.00
        Surface Area: 88.00, Volume: 48.00
Box: Case#2, Length=10.50, Width=21.00, Height=10.50
        Surface Area: 1102.50, Volume: 2315.25

Enter a command: count1
Enter test condition #1: type > cyl
There are 6 shapes.

Enter a command: count1
Enter test condition #1: type == box
There are 4 shapes.

Enter a command: show
show: invalid action
Valid actions: print count min max total avg
Enter help for help

Enter a command: quit
```