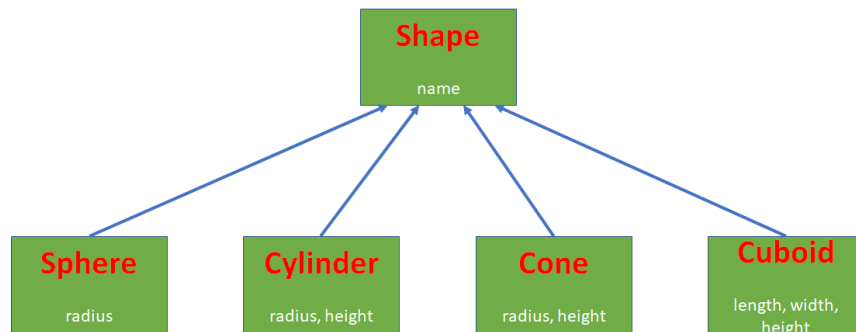


CS 435/535 Assignment #5 – Spring 2021

Project Overview: This project investigates 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`. The `Shape` base class has a 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` (it is supposed 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`, `Cone`, and `Cuboid`. For detailed information on these four particular shapes, please refer to <https://www.mathsisfun.com/geometry/common-3d-shapes.html>. The website also shows how to compute the surface area and the volume of these four shapes. The relationship between the base class and these four derived classes is shown below. These four derived classes have added their own instance variables to describe the details of the corresponding shapes. 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 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, cone or cuboid). The rest will be doubles to represent the dimensions of the shape.

```
<name> sphere <radius>
<name> cylinder <radius> <height>
<name> cone <radius> <height>
<name> cuboid <length> <width> <height>
```

You can download a sample data file named `shapes.dat` from Blackboard for an example. It contains the following shapes.

```
Cube#1 cuboid 1 1 1
Cube#2 cuboid 2 2 2
Cone#1 cone 1 1
Cyl#1 cylinder 1 1
Box#1 cuboid 2 4 6
Box#2 cuboid 10.5 21 10.5
UnitSphere sphere 1
LargeSphere sphere 100
Cone#2 cone 1 2
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 a data file 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 refer to the website mentioned above on how to compute the surface area and the volume. 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 satisfied. 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" 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 to build a list of test conditions.

In this project, you need to create a `makefile` to compile the program. 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** for more testing, but make no changes to the other downloaded files.
- Add four (4) derived classes as required. 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 too 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.** It is not necessary to include the files downloaded from Blackboard.
- 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 Friday, March 19. Late projects are not accepted.

This document including its associated files is for your own personal use only.

**You may not post this document or a portion of this document to a site
such as chegg.com without prior written authorization.**

An assignment shall be completed individually, with no sharing of code or solutions.

All submissions will go through MOSS (Measure Of Software Similarity) for similarity check.

The University of Alabama's Code of Academic Conduct will be rigorously enforced.

An sample execution of the program

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

```
Enter a command: print
```

```
Cuboid: Cube#1, Length=1.00, Width=1.00, Height=1.00  
        Surface Area: 6.00, Volume: 1.00
```

```
Cuboid: Cube#2, Length=2.00, Width=2.00, Height=2.00  
        Surface Area: 24.00, Volume: 8.00
```

```
Cone: Cone#1, Radius=1.00, Height=1.00  
        Surface Area: 7.58, Volume: 1.05
```

```
Cyliner: Cyl#1, Radius=1.00, Height=1.00  
        Surface Area: 12.57, Volume: 3.14
```

```
Cuboid: Box#1, Length=2.00, Width=4.00, Height=6.00  
        Surface Area: 88.00, Volume: 48.00
```

```
Cuboid: Box#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
```

```
Cone: Cone#2, Radius=1.00, Height=2.00  
        Surface Area: 10.17, Volume: 2.09
```

```
Cyliner: Cyl#2, Radius=1.00, Height=2.00  
        Surface Area: 18.85, Volume: 6.28
```

```
Enter a command: print2
```

```
Enter test condition #1: type == cuboid
```

```
Enter test condition #2: area >= 88
```

```
Cuboid: Box#1, Length=2.00, Width=4.00, Height=6.00  
        Surface Area: 88.00, Volume: 48.00
```

```
Cuboid: Box#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 4 shapes.
```

```
Enter a command: count1
```

```
Enter test condition #1: type != cuboid
```

```
There are 6 shapes.
```

```
Enter a command: show
```

```
show: invalid action
```

```
Valid actions: print count min max total avg
```

```
Enter help for help
```

```
Enter a command: quit
```