Sabancı University

Faculty of Engineering and Natural Sciences
CS204 Advanced Programming
Fall 2021

Homework 7 – Geometrical shapes
(inheritance and polymorphism)
Due: 07/01/2022 Friday), 12:30 pm (around noon)

PLEASE NOTE:

Your program should be a robust one such that you have to consider all relevant user mistakes and extreme cases; you are expected to take actions accordingly!

You can NOT collaborate with your friends and discuss solutions. You have to write down the code on your own. Plagiarism will not be tolerated!

1. Introduction

The aim of this assignment is to make you familiar with **inheritance** (multi-inheritance) and **polymorphism** issues. You will deal with multi-inheritance through implementing, from scratch, several small classes, some of which inherit a class that already inherited another class. The content and a brief description of each the classes will be given to you in section two of this document.

Polymorphism will be done in the main() function. The main() function will be given to you in package with your assignment (fig.1), and you are asked to implement a function called in main() that is related to polymorphism. **You are not allowed to make any changes inside the main function**, but you can change the main file. Your polymorphism implementation part should be done in such a way that it will be consistent with the program flow (i.e. have exactly the same sample output) as it is described in section three (fig.2).

Basically, the main idea of this homework is finding, displaying and comparing the perimeter, area and the volume of both 2D (concretely, rectangles) as well as 3D figures (right hexahedrons - boxes) with each-other. Of course, all of this should be done by incorporating some inheritance and polymorphism logics behind, as we shall see it below.

2. The classes to be implemented

In all, you will have to implement five small classes from scratch. Below we will give a brief description of which member functions and variables each class should have, but you should have in mind that some of those should not be defined or implemented at all; rather they are simply inherited from their base class. It is up to you to decide which are inherited, and which not.

1. **Class shape**: this is the base class for all other classes. It is an abstract class (thus, some if its functions are purely virtual).It should have the following properties:
   - **Member variables:** string myName: keeps the name of the shape.
   - **Member methods (functions):** 1. *perimeter()* 2. *area()* 3. *volume()* 4. *getName()*: returns the name of the shape (i.e. returns myName). You should put constructors and destructors as needed.
2. **Class TwoDShape:** inherits class shape. This is also an abstract class. Since TwoDShape inherits class shape, it should also have the same member variables and methods as its base class, of course adjusted (adopted). For this class, subsequently for all those that inherit it, the **volume is defined as being zero**. You can put constructors and destructors as needed, which in turn (in order to avoid rewriting things) can call the constructor(s) of the base class for parts of the implementation(s).
3. **Class ThreeDShape:** inherits class shape. It is also an abstract class. It should have the same member variables and methods as its base class has, again adjusted (adopted) for this class. Namely, for this class, subsequently for all those that inherit it, **the perimeter is defined as being zero**. You can put constructors and destructors as needed, which in turn can call the base class constructor(s) for parts of the implementation(s).
4. **Class rectangle**: inherits TwoDShape. Adds two new member variables called *width* and *length*, both of type float. The perimeter here is defined as 2*(width + length), while the area is width*length.
5. **Class box**: inherits class ThreeDShape. Furthermore, it adds three member variable called *width*, *length* and *height*, all of type float. Also defines area as 2*(width*length+width*height+length*height) and volume as width*length*height.

It is up to you to decide which (or whether) certain member functions will be virtual or not and if they are virtual, would they be purely virtual or not. You should do as much of the job as you can in the base classes, thus avoiding repeating the same things in the inherited classes (actually this was one of the main reasons for using inheritance – avoiding repetitions). Furthermore, you should decide about access-specifier (public, protected or private) for both class variables and methods. The inheritance access-specifier (public, protected or private) is also left to you. Finally, you should decide which functions will be overridden by the inherited classes (if needed), and which will remain the same as the base class during the inheritance process.

While you are not obliged to do so, yet we encourage you to have three headers in your solution project, namely: 1. Shape.h (here only class shape will be declared and defined), 2. TwoDShapes.h (for classes TwoDShape and rectangle), and 3. ThreeDShapes.h (classes ThreeDShape and box). For the last two headers you might use separate .cpp files and "include" them properly.

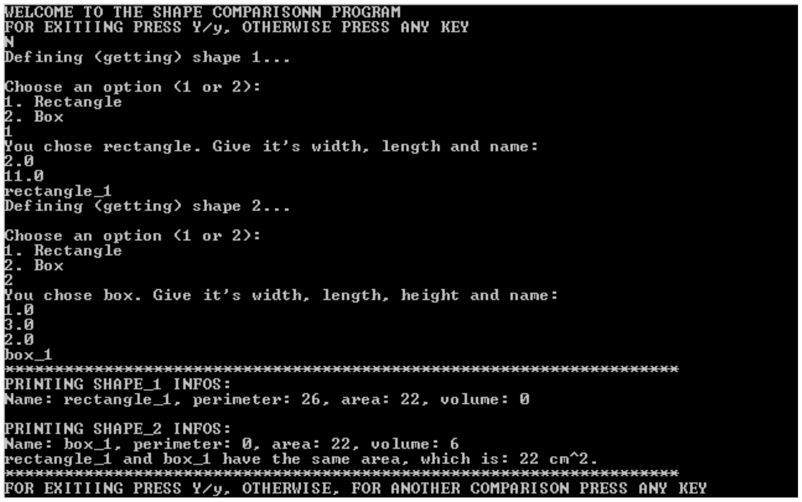
3. Program flow

Main function is given in fig.1. You should implement the getShape() function used inside main() in such a way that the output will be consistent with the sample run given in fig.2. During the process, inside getShape(), you may use other helper functions, of course if you decide it is needed to do so.

```
int main()
{
   cout<<"WELCOME TO THE SHAPE COMPARISONN PROGRAM"<<endl;
   cout<<"FOR EXITIING PRESS Y/y, OTHERWISE PRESS ANY KEY"<<endl;
   /* define here two variables, shape_1 and shape_2, of the class shape.*/
   /* what should they be in order to enable proper (expected) polymorphism usage? */
  char c;
   while (tolower(c = getchar())!='y')
   {
      cout<<"Defining (getting) shape 1..."<<endl;
      shape_1 = getShape();
      cout<<"Defining (getting) shape 2..."<<endl;
      shape_2 = getShape();
      cout<<"********************************************************************"<<endl;
      cout<<"PRINTING SHAPE_1 INFOS:"<<endl<<"Name: "<<shape_1->getName()<<", perimeter: "
        <<shape_1->perimeter()<<", area: "<<shape_1->area()<<", volume: "<<shape_1->volume()<<endl<<endl;
      cout<<"PRINTING SHAPE_2 INFOS:"<<endl<<"Name: "<<shape_2->getName()<<", perimeter: "
        <<shape_2->perimeter()<<", area: "<<shape_2->area()<<", volume: "<<shape_2->volume()<<endl;
      bool nothingInCommon=true; //to check whether they have anything in common (perimeter, area, volume)
      if(shape_1->perimeter()==shape_2->perimeter())
      {
         nothingInCommon = false;
         cout<<shape_1->getName()<<" and "<<shape_2->getName()<<" have a same perimeter, which is: "
                                <<shape_1->perimeter()<<" cm."<<endl;
      }
      if(shape_1->area()==shape_2->area())
      {
         nothingInCommon = false;
         cout<<shape_1->getName()<<" and "<<shape_2->getName()<<" have a same area, which is: "
                                <<shape_1->area()<<" cm^2."<<endl;
      }
      if(shape_1->volume()==shape_2->volume())
      {
         nothingInCommon = false;
         cout<<shape_1->getName()<<" and "<<shape_2->getName()<<" have a same volume, which is: "
                                <<shape_1->volume()<<" cm^3."<<endl;
      }
      if (nothingInCommon)
        cout<<shape_1->getName()<<" and "<<shape_2->getName()<<" don't have anything in common."<<endl;

      cout<<"********************************************************************"<<endl;
      cout<<"FOR EXITIING PRESS Y/y, OTHERWISE, FOR ANOTHER COMPARISON PRESS ANY KEY"<<endl<<endl;
      cin.ignore();//flushing the buffer for remaining character(s), in order getchar() to work
   }//while(tolower(c = getchar())!='y')
   cout<<"PROGRAM EXITING. THANKS FOR USING IT."<<endl;
system("pause");
   return 0;
}
```

Fig.1. Main function

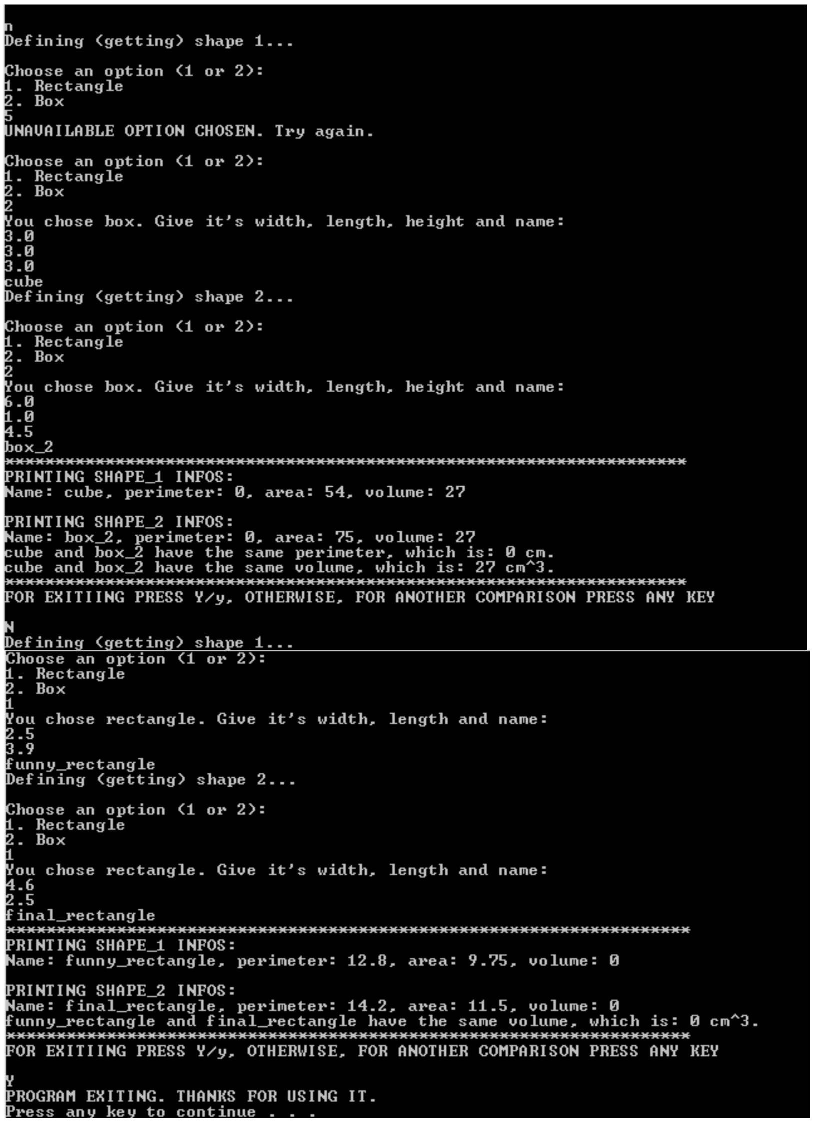


```
WELCOME TO THE SHAPE COMPARISONN PROGRAM
FOR EXITIING PRESS Y/y, OTHERWISE PRESS ANY KEY
N
Defining (getting) shape 1...

Choose an option (1 or 2):
1. Rectangle
2. Box
1
You chose rectangle. Give it's width, length and name:
2.0
11.0
rectangle_1
Defining (getting) shape 2...

Choose an option (1 or 2):
1. Rectangle
2. Box
2
You chose box. Give it's width, length, height and name:
1.0
3.0
2.0
box_1
********************************************************************
PRINTING SHAPE_1 INFOS:
Name: rectangle_1, perimeter: 26, area: 22, volume: 0

PRINTING SHAPE_2 INFOS:
Name: box_1, perimeter: 0, area: 22, volume: 6
rectangle_1 and box_1 have the same area, which is: 22 cm^2.
********************************************************************
FOR EXITIING PRESS Y/y, OTHERWISE, FOR ANOTHER COMPARISON PRESS ANY KEY
```



```
n
Defining (getting) shape 1...

Choose an option (1 or 2):
1. Rectangle
2. Box
5
UNAVAILABLE OPTION CHOSEN. Try again.

Choose an option (1 or 2):
1. Rectangle
2. Box
2
You chose box. Give it's width, length, height and name:
3.0
3.0
3.0
cube
Defining (getting) shape 2...

Choose an option (1 or 2):
1. Rectangle
2. Box
2
You chose box. Give it's width, length, height and name:
6.0
1.0
4.5
box_2
*****************************************************************
PRINTING SHAPE_1 INFOS:
Name: cube, perimeter: 0, area: 54, volume: 27

PRINTING SHAPE_2 INFOS:
Name: box_2, perimeter: 0, area: 75, volume: 27
cube and box_2 have the same perimeter, which is: 0 cm.
cube and box_2 have the same volume, which is: 27 cm^3.
*****************************************************************
FOR EXITIING PRESS Y/y, OTHERWISE, FOR ANOTHER COMPARISON PRESS ANY KEY

N
Defining (getting) shape 1...
Choose an option (1 or 2):
1. Rectangle
2. Box
1
You chose rectangle. Give it's width, length and name:
2.5
3.9
funny_rectangle
Defining (getting) shape 2...

Choose an option (1 or 2):
1. Rectangle
2. Box
1
You chose rectangle. Give it's width, length and name:
4.6
2.5
final_rectangle
*****************************************************************
PRINTING SHAPE_1 INFOS:
Name: funny_rectangle, perimeter: 12.8, area: 9.75, volume: 0

PRINTING SHAPE_2 INFOS:
Name: final_rectangle, perimeter: 14.2, area: 11.5, volume: 0
funny_rectangle and final_rectangle have the same volume, which is: 0 cm^3.
*****************************************************************
FOR EXITIING PRESS Y/y, OTHERWISE, FOR ANOTHER COMPARISON PRESS ANY KEY

Y
PROGRAM EXITING. THANKS FOR USING IT.
Press any key to continue . . .
```

Fig.2. Sample run

Finally, you should not be concerned about the inputs and you may assume that they are always given correctly (as expected), e.g. if a float number is expected, the input will be float. Also, in order to avoid using getline() function, the name of the shape shouldn't contain empty spaces (e.g. empty space, tab, …) . Other, unspecified, issues should be handled in a way you think is a proper one of dealing with them.

Some Important Rules

Although some of the information is given below, first, please read the homework submission and grading policies in the course webpage and lecture notes of the first week. In order to get a full credit, your programs must be efficient and well commented and indented. Presence of any redundant computation or bad indentation, or missing, irrelevant comments may decrease your grades if we detect them. You also have to use understandable identifier names, informative introduction and prompts. Modularity is also important; you have to use functions wherever needed and appropriate.

When we grade your homework we pay attention to these issues. Moreover, in order to observe the real performance of your codes, we are going to run your programs in *Release* mode and **we may test your programs with very large test cases**.

What and where to submit (PLEASE READ, IMPORTANT)

You should prepare (or at least test) your program using MS Visual Studio 2012 C++. We will use the standard C++ compiler and libraries of the abovementioned platform while testing your homework. You need to place your first and last name in the program (as a comment line of course).

Submissions guidelines are below. Some parts of the grading process are automatic. Students are expected to strictly follow these guidelines in order to have a smooth grading process. If you do not follow these guidelines, depending on the severity of the problem created during the grading process, 5 or more penalty points are to be deducted from the grade.

Name your cpp file that contains your program as follows:

"SUCourseUserName_YourLastname_YourName_HWnumber.cpp"

Your SUCourse user name is actually your SUNet user name which is used for checking sabanciuniv e-mails. Do NOT use any spaces, non-ASCII and Turkish characters in the file name. For example, if your SUCourse user name is cago, name is Çağlayan, and last name is Özbugsızkodyazaroğlu, then the file name must be :

Cago_Ozbugsizkodyazaroglu_Caglayan_hw7.cpp

Do not add any other character or phrase to the file name. Make sure that this file is the latest version of your homework program. Compress this cpp file using WINZIP or WINRAR programs. Please use "zip" compression. "rar" or another compression mechanism is NOT allowed. Our homework processing system works only with zip files. Therefore, make sure that the resulting compressed file has a zip extension. Check that your compressed file opens up correctly and it contains your cpp file.

You will receive no credits if your compressed zip file does not expand or it does not contain the correct file. The naming convention of the zip file is the same as the cpp file (except the extension of the file of course). The name of the zip file should be as follows:

You will receive no credits if your compressed zip file does not expand or it does not contain the correct file. The naming convention of the zip file is the same as the cpp file (except the extension of the file of course). The name of the zip file should be as follows:

SUCourseUserName_YourLastname_YourName_HWnumber.zip

For example zubzipler_Zipleroglu_Zubeyir_hw1.zip is a valid name, but

hw1_hoz_HasanOz.zip, HasanOzHoz.zip

are **NOT** valid names.

Submit via SUCourse ONLY! You will receive no credits if you submit by other means (email, paper, etc.).

Successful submission is one of the requirements of the homework. If, for some reason, you cannot successfully submit your homework and we cannot grade it, your grade will be 0.

Note: For consistency, the questions related to HW7 should be asked to your TA Rana Kalkan (ranakalkan@sabanciuniv.edu). Of course, technical help regarding HW7 can be obtained from the TAs/LAs during their OH.

Good Luck!
CS204 Team (Artrim Kjamilji, Rana Kalkan)