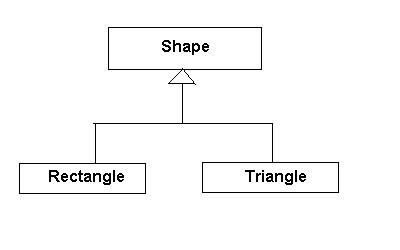
Lab 9: Polymorphism  
In this lab you will:

* Learn how to use C++ virtual functions to implement polymorphism

Basic Structure  
C++ supports polymorphism through virtual functions. The selection of an appropriate function is done at run time, which is called late or dynamic binding. The particular function to invoke is determined at run-time based on the specific type of object. This mechanism is possible only through the use of pointers to objects.

In this lab, you have to implement the following class hierarchy.   


Shape is the base class. It only provides the interfaces, which the derived classes, i.e. Rectangle and Triangle should implement. The three functions which are common to the Rectangle and Triangle class are

* GetArea(): A pure virtual function
* Draw(): A virtual function

In the following steps, you will find all the information you need about the Shape, Rectangle and Triangle classes in order to implement the virtual functions.

Getting the Files  
If you've ssh’ed into the GL servers, issue the following command at the command prompt while you’re inside your lab9 directory:

cp /afs/umbc.edu/users/c/m/cmarron/pub/cmsc202/lab9.zip .

unzip lab9.zip

(Note the '.' at the end of the command — that is very important.)

You should have the following files:

1. [Shape.h:](http://www.csee.umbc.edu/courses/undergraduate/202/fall15_marron/labs/lab09/Shape.h) The Shape class definition.
2. [Shape.cpp:](http://www.csee.umbc.edu/courses/undergraduate/202/fall15_marron/labs/lab09/Shape.cpp) The Shape class implementation.
3. [Rectangle.h:](http://www.csee.umbc.edu/courses/undergraduate/202/fall15_marron/labs/lab09/Rectangle.h) The Rectangle class definition
4. [Rectangle.cpp:](http://www.csee.umbc.edu/courses/undergraduate/202/fall15_marron/labs/lab09/Rectangle.cpp) The Rectangle class implementation
5. [Triangle.h:](http://www.csee.umbc.edu/courses/undergraduate/202/fall15_marron/labs/lab09/Triangle.h) The Triangle class definition
6. [Triangle.cpp:](http://www.csee.umbc.edu/courses/undergraduate/202/fall15_marron/labs/lab09/Triangle.cpp) The Triangle class implementation
7. [Lab9.cpp:](http://www.csee.umbc.edu/courses/undergraduate/202/fall15_marron/labs/lab09/Lab9.cpp) This will contain the driver program.

The Shape class  
The Shape class contains two virtual functions — GetArea() and Draw(). TheGetArea() function is a pure virtual function. This makes Shape an abstract base class, meaning you cannot instantiate it. The function prototypes are as follows:

1. virtual int GetArea() const = 0;
2. virtual void Draw() const;

In addition to the above virtual functions, Shape class has a virtual destructor. Shapehas no private data members.

The Rectangle class  
The Rectangle class publicly derives from the Shape class. The Rectangle class will have two private, integer data members: length and width. The class must implement the following functions:

1. Rectangle(int length=0, int width=0): A Rectangle class constructor.
2. virtual int GetArea() const: An accessor that returns the area of theRectangle.
3. virtual void Draw() const: This function should output a string so that we can identify when the Rectangle's Draw function is called. E.g., "Drawing a rectangle ..."
4. virtual ~Rectangle(): A virtual destructor

The Triangle class  
The Triangle class publicly derives from Shape class. The Triangle class will have two private, integer data members: length and height. The class must implement the following functions:

1. Triangle(int length=0, int height=0): A triangle class constructor
2. virtual int GetArea() const: An accessor that returns the area of theTriangle.
3. virtual void Draw() const: This function should output a string so that we can identify when the Triangle's Draw function is called. E.g., "Drawing a triangle ..."
4. virtual ~Triangle(): A virtual destructor

The Driver Program: Lab9.cpp  
Lab9.cpp should contain the code that will actually do the late binding and test the implementation of virtual functions. You will be given a skeleton of the main()function in Lab9.cpp`. The skeleton creates a pointer to the base class (i.e. a Shapeclass pointer). It also creates a Rectangle and a Triangle object. You need to write the code that performs dynamic binding:

1. Make the base class pointer point to the Rectangle object (You can use '&' to get the object's address)
2. Invoke the functions for area and draw, e.g. ptr->GetArea(), and verify that you are getting the expected results.
3. Repeat steps 1 and 2 for a Triangle object.

Compiling and Testing  
**Compiling**  
g++ -Wall -ansi Lab9.cpp Shape.cpp Rectangle.cpp Triangle.cpp -o lab9   
  
  
**Run**  
linux3[22]% ./lab9