Lab6: Object-oriented Design and Array Data Structures

Objective

The objective of this lab is to learn to design and implement modular programs using functions:

- Design solution using C++ classes, instantiate objects and invoke functions on the objects.
- Use array data structures to store data.
- Reinforce all the programming concepts learned in Labs 1- 5.

Description

For this lab, you will revisit the two (2) problems from Lab 5:

1. **Projectile**: Design a class ***Projectile*** with functions ***height*** and ***distance*** functions that compute the distance and height of a projectile as a function of time as specified in Lab 3. The initial velocity, initial height and launch angle will be specified as private data (properties) of the class ***Projectile***. You will write another program with main function to test the ***Projectile*** class.

2. **Stick Figure**: Define and use three function ***drawTrianlgle, drawRectangle,*** and ***drawHollowRectangle*** to draw a stick figure as specified in Lab 3 and another fun figure of your choice using the same principles. Instead of printing out or saving the output in a file, the two figures will be saved in a two-dimensional array of characters and passed back to main program. After both figures have been captured into the array, it will be printed out and saved in an output file.

We will use a combination of files, standard input (keyboard) and standard output (monitor screen). For each solution you will develop an algorithm, code the algorithm in C++, develop and test it in the Visual C++ environment.

Visual C++ Environment

A **project** in the VC++ environment is a program. A **workspace** is a folder in which all project-related information is stored. When you create a project you may create it in a new workspace or add it to an existing workspace. Refer to the handout “Working with VC++ programs” for more details. Do the following:

1. Create a workspace called Lab6.
2. Create a project for each program and add it to the Lab5 workspace. Create project names as follows:

    a. Projectile6
    b. StickFigures6

Template for Program Header

Place the following code at the beginning of every source code file that you submit for this class.

```
/************************************************************
* NAME: your name                                           *
* PERSON NUMBER: your person number                         *
* PROGRAM:  Lab name                                        *
* PURPOSE: 1-2 line summary of the purpose of the lab       *
* DATE: Date of last update                                 *
* PLATFORM: Microsoft Visual C++ 6.0 Pro                    *
* Course & Section:                                         *
************************************************************/
```

On-line submission of your code

All source code (.cpp) and data files created by you and generated by the programs that you implement will need to be submitted using the on-line command which is available on-line on the course website (www.cse.buffalo.edu/~terrill/eas230). A summary of the procedure for submission is as follows:

For example, from your command-prompt, do the following:

- ssh unix.eng.buffalo.edu
  - enter your username and password
- Change directories to where your files are located as shown in the example below,
  - cd ~/eas230/Lab6/Projectile6
- Run the following command for each file
  - `submit_eas230 <file>`
    - where `<file>` is the name of the file you need to submit.

Problem 1: Projectile Functions 60%

We will modify the projectile problem solved in Lab 3 to design and implement a ***class Projectile*** that encapsulates the properties and operations (functions) associated with a projectile.

- You will implement the class using a header file and a corresponding source: ***Projectile.h and Projectile.cpp***
- ***Projectile.h*** will include the private data and the function to operate on the data.
- Some of the functions are as given below:
  Projectile(double angle, double height, double velocity); // constructor
  double distance (double time);
  double height (double time);
- ***Projectile.cpp*** will include the definition of the function specified in the ***Projectile.h***
- To test the class Projectile, you will write an application ***ProjectileTester.cpp*** that will instantiate an object ***happyObj*** of class Projectile and test its functions.
  - You will input values of alpha (in degrees), v0 and z0 from standard input (keyboard) and initialize the ***happyObj***.
  - The main function will call the object's functions for computing height and distance for various time values. It is good programming style to do all the input/output from the main function.
  - For each input set, the program will compute a set of r(t) and z(t) and output to the screen as well as to an output file "**Plot6.out**". It will have three columns time, distance and height. Each set of output will have heading that shows the angle of launch. A sample data set is shown below:

Projectile Motion with launch angle 15 degrees.

```
Time   Distance   Height
……......
0.3    1.88356    10.5632
0.31    1.94634   10.5501
0.32    2.00913   10.5361
0.33    2.07191   10.521
……......
```

Projectile Motion with launch angle 30 degrees.

```
Time   Distance   Height
……......
0.53    2.98346 10.8447
0.54    3.03975 10.8247
0.55    3.09604 10.8037
0.56    3.15233 10.7818
……..
```

The projectile problem from Labs 3 and 4 is repeated below for your reference:

This problem deals with equations dealing with motion of objects in two dimensions. In particular we will look at projectile motion as discussed in the web resource: ***http://www.glenbrook.k12.il.us/gbssci/phys/Class/vectors/u3l2b.html***. We are interested in the height and distance of the projectile with reference to time. The formulae for these two are readily available at another web resource: **http://www.egwald.com/geometry/projectile3d.php** and are repeated below. Assume that the projectile is fired from a gun that is mounted on a cliff above the sea.

z_0 = height of the gun above the x-y plane in meters.
v_0 = muzzle velocity = initial velocity of the projectile in meters/sec.
alpha = the angle between the horizontal (the x-y plane) and the muzzle of the gun in radians.
z(t) = the height of the projectile t seconds after being fired (meters).
r(t) = the distance of the projectile from the gun after t seconds (meters).
g = pull of gravity (9.81 meters/sec).
Then the parametric equations of motion of (r(t), z(t)) are:

$$r(t) = v_0 * \cos(alpha) * t$$
$$z(t) = -1/2 * g * t^{\wedge 2} + v_0 * \sin(alpha) * t + z_0$$

You will input z_o, v_o, and alpha and compute distance and height of the at various time intervals until the projectile reaches ground. Your program should repeatedly compute the distance and height and output them in a tabular form. Use a while loop control structure. Choose your intervals appropriately after executing the correct version of the program couple of times. Your output will contain the data computed by the equation for the distance and height against time values.

Code

Create a new project called **Projectile6** and add it to the workspace **Lab6**.

1. You will create a header file ***Projectile.h*** that specified the Projectile class. (The C++ code for this will be given to you during lecture.)
2. You will create a source file ***Projectile.cpp*** that defines the functions specified in ***Projectile.h***
3. Then create the ***ProjectileTester.cpp*** that will input launch angle, initial velocity and initial height and instantiate (construct) the ***happyObj*** with these initial values. Then it will invoke the ***distance (double time) and height(double time)*** to compute these values in a loop until the projectile hits the ground. It will output the values to a file "***Plot6.out***"

Submission

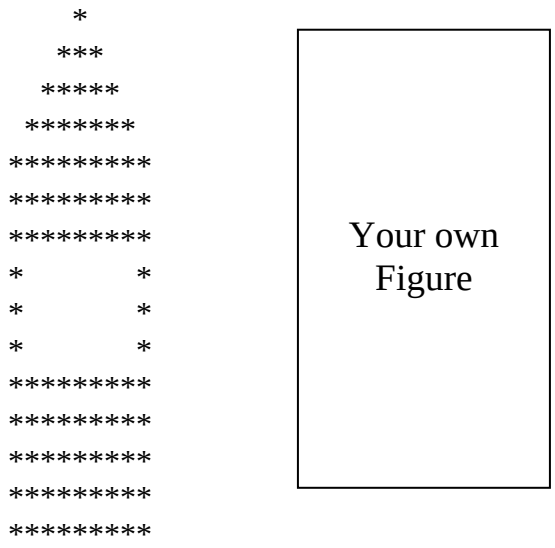
Submit the files **Projectile.h, Projectile.cpp, ProjectileTester.cpp, and Plot6.out**.

Problem 2: Stick Figure with Two-dimensional Arrays 40%

The problem is same as Lab3. The output from the functions will be written into a two dimensional array of characters large enough to hold the figure given below and another one of your choice. **It is very important that both figures are stored in a single two dimensional array. This array will be declared in the main function and passed as a parameter to the drawing functions, as shown in the prototypes given below.**

The main function will save array into an output file ***Stick6.out***.

Observe the "stick" figure given below that is made of asterisk symbols. Use functions and write modular and reusable code to save it into an array. Reuse the functions and draw another meaningful figure of your choice.



Use functions for building triangle, rectangle and hollow rectangle. The figures can be filled up in the array one-after another or side-by-side. (The latter may be slightly difficult and may need the use of the *parameter offset*.) The function prototypes are given below:

void drawTriangle (int length, int width, int offset,int startRow, char board[][]);
void drawRectangle (int length, int width, int offset, int startRow,char board[][]);
void drawHollowRectangle (int length, int width, int offset, int startRow, char board[][]);

Code

Create a single project called **StickFig6** and add it to the workspace **Lab6**. Create C++ source files named **StickFig6.cpp**. Main function will output the array containing the two stick figures into a file Stick6.out.

Submission

Submit the files **StickFig6.cpp** and **Stick6.out.**

LATE POLICY:

NO LATE LAB SUBMISSION WILL BE ACCEPTED FOR THIS LAB AND ANY FUTURE LABS.