Lab3: Selection and Repetition Control Structures

Objective

The objective of this lab is to learn to solve problem that require selection and repetition in their solution. We will focus on these issues:

- Engineering problem solving for motion in two dimensions
- Selection control structures: *if* and *if..else*
- Repetition control structures: *while, do while* and *for*
- Software engineering concept: incremental development
- Input data validation

Description

For this lab, you will solve three (3) problems:

1. Compute the sine and cosine of angles using the Taylor series, number of terms used limited by the accuracy specified by the user (20%).
2. Compute the distance and height of the project as a function time (40%).
3. Draw stick figure according to specification given by a diagram (40%).

For each solution you will develop an algorithm, code the algorithm in C++, develop and test it 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 Lab3.
2. Create a project for each program and add it to the Lab2 workspace. Create project names as follows:
    a. Taylor2
    b. Projectile
    c. StickFigure

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) will need to be submitted using the on-line command which available on-line on the course website (www.cse.buffalo.edu/~terrill/eas230). A summary of the procedure for submission is as follows:

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/Lab3/Taylor2
- Run the following command for each file
  - `submit_eas230 <file>`
    - where `<file>` is the name of the file you need to submit.

Problem 1: Taylor Series 20%

You will work on *incremental development* of the Taylor series discussed in Lab2. In Lab2 we limited the number of terms to 3. In this lab the number of terms will be determined by the accuracy of result. You will design your program so that it is accurate to 6 decimal places. If we use the approximation given by Taylor Series for sine and cosine, for accuracy to 6 decimal places, we stop iterating when the term is insignificant as quantified by the following expression:

$$\left| \frac{x^n}{n!} \right| < 0.0000005$$

Your computation will "iteratively" add terms after making sure it is significant using the expression given above. For iterating we suggest using "while" control structure.

The Taylor series approximation for sine and cosine are given below:

$$\sin x = x - (x^3/3!) + (x^5/5!) - (x^7/7!) + (x^9/9!) \ldots..$$

$$\cos x = 1 - (x^2/2!) + (x^4/4!) - (x^6/6!) + (x^8/8!) \ldots..$$

Algorithm

Write an algorithm and include it as a "preamble" comment in your program.

Code

Create a new project called **Taylor2** and add it to the workspace **Lab3**. Create a C++ source file named **Taylor2.cpp**.

Submission

Submit the code for Taylor2.cpp.

Problem 2: Motion in Two Dimensions 40%

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:

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

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 you intervals appropriately after executing the correct version of the program couple of times. Your output will contain

- Title of the problem
- Tiles for columns “Time” , “Distance” and “Height”
- The data computed by the equation for the distance and height against time values.

Algorithm

Write an algorithm and include it as a “preamble” comment in your program.

Code

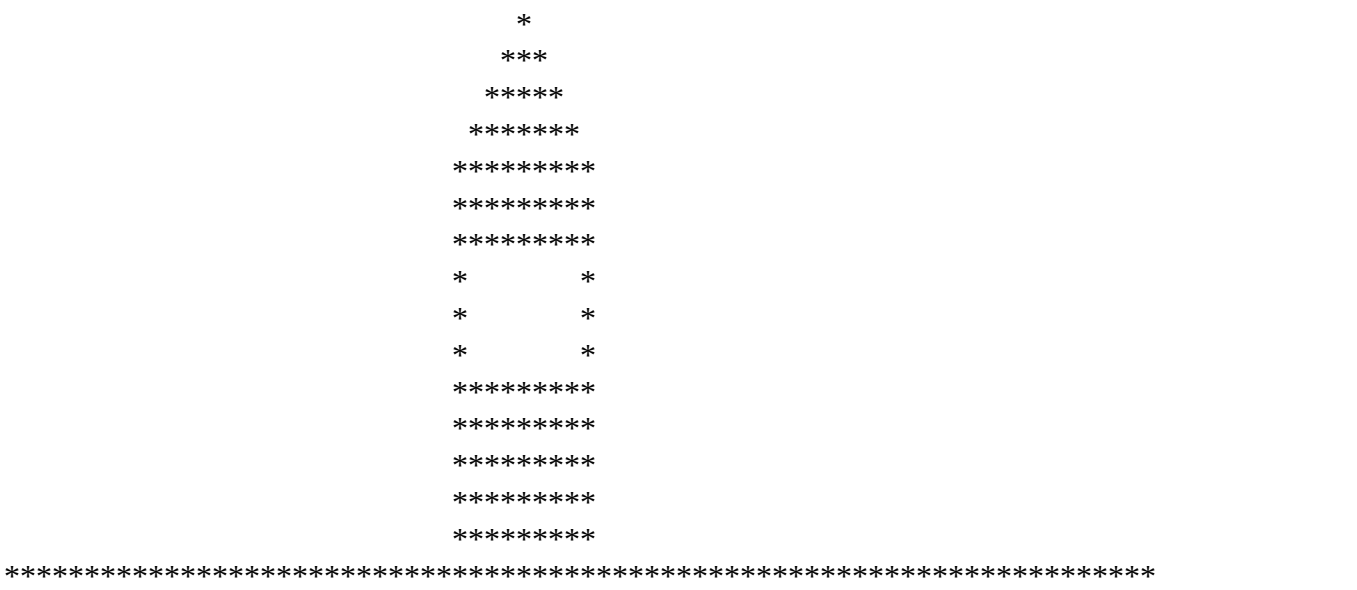
Create a new project called **Projectile** and add it to the workspace **Lab3**. Create a C++ source file named **Projectile.cpp**.

Submission

Submit the code for **Projectile.cpp**.

Problem 3: Stick Figure 40%

Observe the “stick” figure given below that is made of asterisk symbols. Use “for” loops and write modular and reusable code to print that diagram. Reuse the “for” loops and draw another meaningful figure of your choice.



Algorithm

Write an algorithm and include it as a “preamble” comment in your program.

Code

Create a new project called **StickFig** and add it to the workspace **Lab3**. Create a C++ source file named **StickFig.cpp**.

Submission

Submit the code for **StickFig.cpp**.