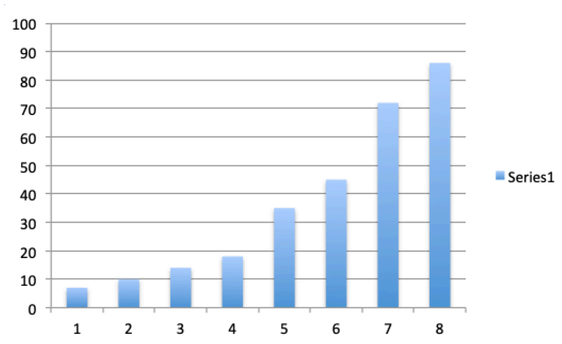
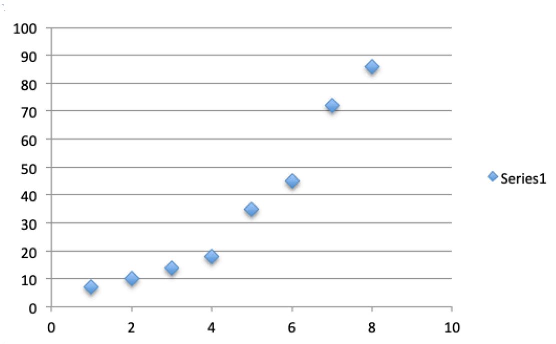
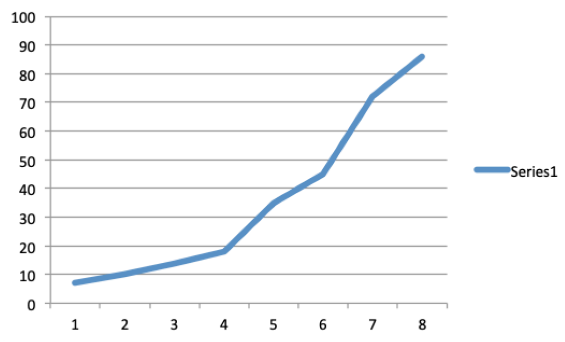
CSCE 4813 – Programming Project #1
Due Date – 01/31/2020 at 11:59pm
REVISED

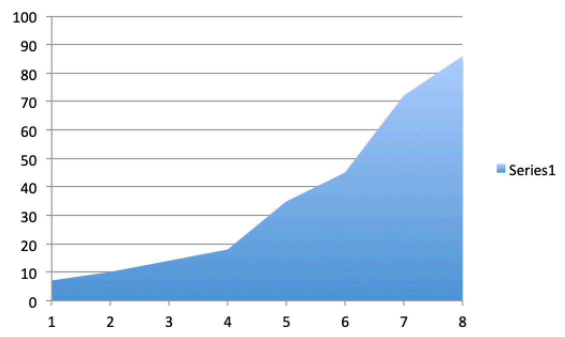
1. Problem Statement:

When spreadsheet software was introduced in the 1980's it dramatically changed how people could store, manipulate and visualize data. The spreadsheet's simple user interface enabled users to easily enter rows and columns of data, do arithmetic calculations, and most importantly display the results in easily understood charts.









In this programming project, you will be developing a collection of functions that will let users create and display data using the four types of charts above.

If we look closely at these four charts, you will see that the background information for all four of these charts are the same.

- They all display a series of evenly spaced horizontal lines.
- They all display evenly spaced tic marks on the X and Y axis.
- They all display numerical values adjacent to the tic marks on the axes.
- They all display the name of the data field being displayed

The major difference between these four charts is how they convey information about data values using different geometric objects.

- For a column chart, the height of each column is determined by the data value relative to the height of the chart. The width of each column can be determined by dividing the width of the chart by the number of data values being displayed.

- For a point chart, data values are displayed using points represented by squares, diamonds, or circles. In this case, the X and Y locations of points are determined by the magnitude of the data value relative to the height and width of the chart.

- For a line chart, data is displayed using a sequence of line segments that connect the point locations of the data. In this case, the width of the line is chosen to be larger than the scale lines so the line segments are visible.

- For the area chart, data is displayed by filling in the area under the line chart with a solid color. This area can be defined using a collection of polygons where each polygon has one line segment going through two data points and three line segments that mark the right, bottom, and left sides of the area being enclosed.

1.1 Chart Creation

For this programming assignment, your first task is to implement a C++ program that prompts the user to enter the name of their input data file, and the type of chart they would like to create (column, point, line, area). Your program should then read in the input data file, do the necessary calculations to scale this data to the size of the output window, and then output a sequence of graphics commands to create the specified chart.

Your output charts should look as similar as possible to the four examples above. To simplify this project, you are NOT required to output the numbers that are adjacent to the tic marks in the charts or the name data field being displayed. You only need to draw lines in the background and the geometric objects representing data values.

The four graphics commands we will be using for this project are:

- set_color #R #G #B

  This command sets the color of the "pen" in the graphics program. All object drawn after this command will be displayed with this color until another set_color command is executed. The color values for R, G, B should be floating point values between 0 and 1, where 0 = black, and 1 = full color.

- draw_point #size #x #y

  This command draws a point of the specified size at the specified location. The size value should be an integer between [1..10]. The (x,y) coordinates should be integers between [0..width-1] and [0..height-1] respectively.

- draw_line #width #x1 #y1 #x2 #y2

  This command draws a line of the specified width from (x1,y1) to (x2,y2). The width value should be an integer between [1..10]. The (x,y) coordinates should be integers between [0..width-1] and [0..height-1] respectively.

- draw_polygon #N #x1 #y1 #x2 #y2 ... #xN #yN

  This command fills in a convex polygon region that is defined by a sequence of N points. The value N should be an integer greater than or equal to 3 that specifies the number of points on the polygon boundary. The (x,y) coordinates of each point should be integers between [0..width-1] and [0..height-1] respectively.

1.2 Chart Display

Your second task is to implement an OpenGL program that reads a sequence of commands in the format above, and calls the appropriate OpenGL commands to display the corresponding chart on the screen. You will be given a skeleton OpenGL program that demonstrates how to draw each of the graphics primitives above.

To demonstrate the correctness of your program, you should run your first program to four times to generate four sequences of output commands, and save these commands in four data files. Then you should run your OpenGL program four times to generate images of these charts. When your program is working, do screen captures of your four charts and include these images into your project report.

2. Design:

For each of the chart types above, the major design task is to work out what sequence of graphics commands are needed, and exactly what x_pos y_pos values to use when displaying these geometric objects. The key here is to work out the scale factors you need to use to convert the raw data values into coordinates that are within the chart. For example, if the data values go from 1..100 and the height of the chart is 500 pixels, then the scale factor should be 500/100 = 5. You will notice that there is small border around the outside of the drawing area. Your program will need to take this border into account when displaying graphics objects.

3. Implementation:

This semester we will be using C++ and OpenGL to implement all of our programming projects. If you are using a Mac with Xcode installed, then you can download the src.tar file and compile the sample graphics code using the enclosed Makefile. If you are using a PC, then your best option would be to download and install a Linux VM from the department's website. The instructions for doing this are posted in README file the "Source Code" page of the class website. Once you have Linux and OpenGL installed, you can compile your graphics program using "g++ -Wall show_chart.cpp -o show_chart -lGL -lGLU -lglut".

You are encouraged to look at sample OpenGL programs to see how the "main" function and the "display" function are normally implemented. As always, you should break the code into appropriate functions, and then add code incrementally writing comments, adding code, compiling, debugging, a little bit at a time.

Remember to use good programming style when creating your program. Choose good names for variables and constants, use proper indenting for loops and conditionals, and include clear comments in your code. Also, be sure to save backup copies of your program somewhere safe. Otherwise, you may end up retyping your whole program if something goes wrong.

4. Testing:

Test your program with different random number generator seeds until you get some images that look fun/interesting. Take a screen shot of these images to include in your project report. You may also want to show some bad/ugly images that illustrate what happens if there is a problem somewhere (e.g. too many lines, lines too short, etc.). You can discuss how you corrected these problems in your project report.

5. Documentation:

When you have completed your C++ program, write a short report using the project report template describing what the objectives were, what you did, and the status of the program. Be sure to include several output images. Finally, describe any known problems and/or your ideas on how to improve your program. Save this report to be submitted electronically via Blackboard.

6. Project Submission:

In this class, we will be using electronic project submission to make sure that all students hand their programming projects and labs on time, and to perform automatic plagiarism analysis of all programs that are submitted. When you have completed the tasks above go to Blackboard to upload your documentation (a single docx or pdf file), and all of your C++ program files. Do NOT upload an executable version of your program.

The dates on your electronic submission will be used to verify that you met the due date above. All late projects will receive reduced credit:

- 10% off if less than 1 day late,
- 20% off if less than 2 days late,
- 30% off if less than 3 days late,
- no credit if more than 3 days late.

You will receive partial credit for all programs that compile even if they do not meet all program requirements, so handing projects in on time is highly recommended.

7. Academic Honesty Statement:

Students are expected to submit their own work on all programming projects, unless group projects have been explicitly assigned. Students are NOT allowed to distribute code to each other, or copy code from another individual or website. Students ARE allowed to use any materials on the class website, or in the textbook, or ask the instructor and/or GTAs for assistance.

This course will be using highly effective program comparison software to calculate the similarity of all programs to each other, and to homework assignments from previous semesters. Please do not be tempted to plagiarize from another student.

Violations of the policies above will be reported to the Provost's office and may result in a ZERO on the programming project, an F in the class, or suspension from the university, depending on the severity of the violation and any history of prior violations.