Workshop 7

STL Algorithms

In this workshop, you use the Algorithm category of the Standard Template Library.

LEARNING OUTCOMES

Upon successful completion of this workshop, you will have demonstrated the abilities to

- Copy data from a file into a sequential container
- Use the numeric library to accumulate data values
- Use a lambda expression to specify an operation on each value in a data set
- Use the algorithm library to sort data values

SUBMISSION POLICY

The *in-lab* section is to be completed during your assigned lab section. It is to be completed and submitted by the end of the workshop period. If you attend the lab period and cannot complete the *in-lab* portion of the workshop during that period, ask your instructor for permission to complete the *in-lab* portion after the period. If you do not attend the workshop, you can submit the *in-lab* section along with your *at-home* section (see penalties below). The *at-home* portion of the lab is due on the day that is four days after your scheduled in-lab workshop (23:59:59) (even if that day is a holiday).

All your work (all the files you create or modify) must contain your name, Seneca email and student number.

You are responsible to back up your work regularly.

Late Submission Penalties:

- *In-lab* portion submitted late, with *at-home* portion: **0** for *in-lab*. Maximum of 7/10 for the entire workshop.
- If any of *in-lab*, *at-home* or *reflection* portions is missing, the mark for the workshop will be **0**/10.

SPECIFICATIONS – IN LAB

Introduction to Statistical Analysis

Statistical analysis uses standard measures to make general predictions about data based on a small sample of the actual data:

- sample mean -- the average of all values in the sample
- sample standard deviation -- the spread of the numbers away from their mean
- sample median -- the middle number in the sorted set of the values (that is, the value separating the lower and upper halves of the data in a sorted set)

The formula for the sample mean is

$$z_{mean} = \frac{\Sigma_i z_i}{n}$$

The symbol Σ denotes *sum of*, i refers to the index of the element in the sample set and n refers to the number of elements in the sample set.

The formula for the sample standard deviation is

$$ssd = \left[\frac{\Sigma_i (z_i - z_{mean})^2}{n-1}\right]^{1/2}$$

A regression line provides a simple form of predicting outcomes based on sample data. The line relates a set of independent (x) values to a corresponding set of dependent (y) values. The number of values in each set is the same; that is, each value in the independent set has one corresponding value in the dependent set.

The regression line is the best fit of a line to the pairs of data values. It is the line that passes through the data points drawn on a two-dimensional (x, y) system of coordinates as closely as possible to the data points. The line's coefficients are:

- slope – the slope of the line in the x-y plane
- y_intercept – the y value of the line where it crosses the y-axis in the x-y plane

The formulas for these coefficients are:

$$slope = \frac{n(\Sigma_i x_i y_i) - \Sigma_i x_i \Sigma_i y_i}{n(\Sigma_i x_i^2) - (\Sigma_i x_i)^2}$$

$$y_{intercept} = \frac{\Sigma_i y_i - slope * \Sigma_i x_i}{n}$$

You can find a regression calculator here. The linear regression of a set of points is displayed here in graphical form. The dots represent the x-y coordinates of the points in the data set and the red line is the regression line calculated with the formulas above. You can find more information here.

Detail Specifications

The solution to the in-lab part of this workshop consists of two modules:

- **w7** (supplied)
- **DataTable**

Enclose your source code within the **sict namespace** and include the necessary header file guard. The output from your executable running Visual Studio with the following command line argument should look like

```
Command Line : C:\Users\...\Debug\in_lab.exe Simple.dat

****************************************
*** Processing file [Simple.dat]
****************************************

Data Values
------------
       x       y
  2.1000  8.0000
  2.5000 12.0000
  4.0000 14.0000
  3.6000 10.0000

Statistics
----------
  y mean    =  11.0000
  y sigma   =   2.5820
```

The input for testing your solution is stored in a user-prepared file. The name of the file is specified on the command line as shown in red above. The file is supplied with this workshop. The contents of this file are

```
2.1 8
2.5 12
4.0 14
3.6 10
```

Each record in the file consists of a product number and its price.

DataTable Module

Design and code a class template named **DataTable** for holding and processing statistical data. Your interface to the hierarchy uses the field width (**int FW**) defined outside the translation unit and specifies the following public member function requirements:

- A one-argument constructor that receive a reference to an std::ifstream object, read all records from the object and stores them in the most appropriate STL container.
- **void displayData(std::ostream& os) const** – a query that displays the x-y data in the format shown above. Use the field width and precision specified in **w7**.

- **void displayStatistics(std::ostream& os) const** – a query that displays the statistics for the current object in the format shown above. Use the field width and precision specified in **w7**.

In-Lab Submission (30%)

To test and demonstrate execution of your program use the same data as shown in the output example above.

Upload your source code to your `matrix` account. Compile and run your code using the latest version of the gcc compiler and make sure that everything works properly.

Then, run the following command from your account: (replace `profname.proflastname` with your professor's Seneca userid)

```
~profname.proflastname/submit 345XXX_w7_lab<ENTER>
```

and follow the instructions. Replace **XXX** with the section letter(s) specified by your instructor.

SPECIFICATIONS – AT HOME

The at-home part of this workshop adds median and linear regression analysis to your in-lab solution.

The output from your executable running Visual Studio with the following command line argument should look like

```
Command Line : C:\Users\...\Debug\at_home.exe Simple.dat HS_College_GPA.dat

****************************************
*** Processing file [Simple.dat]
****************************************

Data Values
------------
       x       y
  2.1000  8.0000
  2.5000 12.0000
  4.0000 14.0000
  3.6000 10.0000

Statistics
----------
  y mean    =  11.0000
  y sigma   =   2.5820
  y median  =  12.0000
  slope     =   1.9087
  intercept =   5.1784

****************************************
*** Processing file [HS_College_GPA.dat]
****************************************

Data Values
------------
       x       y
  3.4500  3.7600
  2.7800  2.8700
  2.5200  2.5400
  3.6700  3.8300
  3.2400  3.2900
  2.1000  2.6400
  2.8200  2.8600
  2.3600  2.0300
  2.4200  2.8100
  3.5100  3.4100
  3.4800  3.6100
  2.1400  2.4800
  2.5900  3.2100
  3.4600  3.5200
  3.5100  3.4100
  3.6800  3.5200
  3.9100  3.8400
  3.7200  3.6400
  2.1500  2.1400
  2.4800  2.2100
  3.0900  3.1700
  2.7100  3.0100
  2.4600  3.1700
  3.3200  3.0100
  3.6100  3.7200
  3.8200  3.7800
  2.6400  2.5100
  2.1900  2.1000
  3.3400  3.2100
  3.4800  3.6800
  3.5600  3.4800
  3.8100  3.7100
  3.9200  3.8100
  2.5200  2.0900
  2.7100  2.1700
  3.1500  2.9800
  3.2200  3.2800
  2.2900  2.7400
  2.0300  2.1900
  3.1400  3.2800
  3.5200  3.6800
  2.9100  3.1700
  2.8300  3.1700
  2.6500  3.3100
  2.4100  3.0700
  2.5400  2.3800
  2.6600  2.9400
```

```
    3.2100    2.8400
    3.3400    3.1700
    3.6800    3.7200
    2.8400    2.1700
    2.7400    2.4200
    2.7100    2.4900
    2.2400    3.3800
    2.4800    2.0700
    3.1400    3.2200
    2.8300    2.7100
    3.4400    3.3100
    2.8900    3.2800
    2.6700    3.1900
    3.2400    3.2400
    3.2900    3.5300
    3.8700    3.7200
    3.9400    3.9800
    3.4200    3.0900
    3.5200    3.4200
    2.2400    2.0700
    3.2900    3.1700
    3.4100    3.5100
    3.5600    3.4900
    3.6100    3.5100
    3.2800    3.4000
    3.2100    3.3800
    3.4800    3.5400
    3.6200    3.4800
    2.9200    3.0900
    2.8100    3.1400
    3.1100    3.2800
    3.2800    3.4100
    2.7000    3.0200
    2.6200    2.9700
    3.7200    4.0000
    3.4200    3.3400
    3.5100    3.2800
    3.2800    3.3200
    3.4200    3.5100
    3.9000    3.6800
    3.1200    3.0700
    2.8300    2.7800
    2.0900    3.6800
    3.1700    3.3000
    3.2800    3.3400
    3.0200    3.1700
    3.4200    3.0700
    3.0600    3.1900
    2.7600    2.1500
    3.1900    3.1100
    2.2300    2.1700
    2.4800    2.1400
    3.7600    3.7400
    3.4900    3.2700
    3.0700    3.1900
    2.1900    2.9800
    3.4600    3.2800
```

```
Statistics
----------
  y mean    =   3.1212
  y sigma   =   0.5066
  y median  =   3.2100
  slope     =   0.7802
  intercept =   0.7279
```

The input for testing your solution is stored in supplied user-prepared files. The names of these files are specified on the command line as shown in red above.

DataTable Module

Update the constructor for your **DataTable** template to calculate the median and regression coefficients for the data received. Update the **displayStatistics** query to include output of the median, slope and y-intercept as shown above.

Reflection

Study your final solution, reread the related parts of the course notes, and make sure that you have understood the concepts covered by this workshop. Take your time. Explain in your own words what you have learned in completing this workshop. Include in your explanation but do not limit it to the following points (40%):

- The reason for using the vector container rather than any other available in the STL.
- List the STL template functions that you used in your solution.
- Identify where you used lambda expressions.
- Comment on the ease of programming associated with the STL.

To avoid deductions, refer to code in your solution as examples to support your explanations.

Include all corrections to the Quiz(zes) you have received (30%).

At-Home Submission (70%)

To test and demonstrate execution of your program use the same data as shown in the output example above.

Upload your source code to your `matrix` account. Compile and run your code using the latest version of the gcc compiler and make sure that everything works properly.

Then, run the following command from your account: (replace `profname.proflastname` with your professor's Seneca userid)

```
~profname.proflastname/submit 345XXX_w7_home<ENTER>
```

and follow the instructions. Replace **XXX** with the section letter(s) specified by your instructor.