**CSCI 1730 - Programming Assignment 4 - 100 pts.**

**Due Date: Apr 24, 2019**

1. Write a C++ program that will repeatedly perform the following text file analysis tasks.

* It will prompt the user to enter a text file name and then open the file for reading.
* It will count the number of words in the open text file and display this count along with the average word length (average number of characters per word), the total number of word characters, the total number of punctuation characters, the length of the shortest word, and the length of the longest word.
* It will list all of the shortest words found in the text file.
* It will list all of the longest words found in the text file.
* It will search for a word specified by the user and report how many times the word occurs in the text file.

The program will display all of the results on the screen as well as writing the results to an output report text file. The main function of your program should first ask the user for the name of the output report file and open the file for writing. Then it should repeatedly offer the user a menu of the above tasks. Each of the above tasks should be implemented in your program as separate functions.

**Note:** A word is defined to be a string of symbols that is preceded and followed by whitespace or the beginning of the line or the end of the line, or a portion of such a string. If such a string begins and ends with non-punctuation characters, then the string is a word. If such a string begins or ends with punctuation characters, the word embedded in the string will be the string with preceding and ending punctuation characters removed. In either case, punctuation characters in the middle of the string should not be removed and should be considered part of the word. So, the punctuation count should include those punctuation characters before or after a word, but not include punctuation characters embedded in a word.

**Note:** The length of a word followed by a punctuation character does not include the punctuation character.

**File handling notes:**

* An input file must be open before the user selects any of the above options other than the first one. So, the functions that handle the other tasks need to check to see if an input file is open before doing their task.
* To determine if the input file is currently open, you can use this function call which returns true if the file associated with fin is open:

fin.is\_open()

* If the user selects the first option and an input file is already open, the function handling that task should first close the input file before attempting to open a new one.
* Each function that processes the input file will need to have the input file positioned to read at its beginning. Include this code to accomplish this task (assume that fin is the ifstream object that refers to your input file in the program):

fin.clear();

fin.seekg(0);

* ifstream and ofstream function parameters must be passed by reference.

You can copy two text files ([S:\Coursework\Liu\story.txt](file:///S:\Coursework\Liu\story.txt) and [S:\Coursework\Liu\story2.txt](file:///S:\Coursework\Liu\story2.txt)) for testing your program.

To see how this program will work, copy [S:\Coursework\Liu\a4-1.exe](file:///S:\Coursework\Liu\a5-1.exe) and [S:\Coursework\Liu\story.txt](file:///S:\Coursework\Liu\story.txt) or [S:\Coursework\Liu\story2.txt](file:///S:\Coursework\Liu\story2.txt) to your own folder to run the program. The program will first prompt the user to enter an output file name, you can enter a file name of your choice, with a file ending .txt. If you choose choice 1, it will ask you to enter the input file name, you can enter story.txt or story2.txt.

Note: if you use story.txt as input file, the following output (both on screen and output file) would be a sample run of the program :

\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_

Filename: story.txt

Total number of words = 179

Average word length = 4.9162 characters.

Shortest word length = 1

Longest word length = 13

Filename: story.txt

Shortest words in file:

a

a

a

I

a

a

a

a

a

Filename: story.txt

Longest words in file:

disappointing

participation

Massachusetts

Filename: story.txt

Search word: hello

This word appears 0 time(s) in the file

Filename: story.txt

Search word: disappointing

This word appears 1 time(s) in the file

\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_

2. **Vectors** are math quantities that have both magnitude and direction. This problem involves the use of two-dimensional vectors – here are the math details:

* A **two-dimensional vector** is a directed line segment from the origin of the *xy*-plane to a point, (*a*,*b*). This vector is denoted <*a*,*b*>.
* **Vector addition/subtraction:** Two vectors can be added or subtracted resulting in a vector:

<*a*,*b*> + <*c*,*d*> = <*a*+*c*,*b*+*d*> or <*a*,*b*> – <*c*,*d*> = <*a*–*c*,*b*–*d*>

* **Vector dot product:** Two vectors can be multiplied resulting in a number – this operation is called the dot product of two vectors: <*a*,*b*> \* <*c*,*d*> = *ac*+*bd*
* **Vector/scalar multiplication:** A vector can be multiplied by a scalar (a number) resulting in a vector**;** for any scalar *r*, <*a*,*b*>\**r* = <*ra*, *rb*>
* **Equality of vectors:** <*a*,*b*> = <*c*,*d*> means *a* = *c* and *b* = *d*.

In the last two assignments, you wrote programs that modeled **complex numbers**. Once again, here are the math details:

* A **complex number** is a number of the form *a* + *bi* where *a* and *b* are real numbers and *i* is the imaginary unit, .
* **Complex number addition/subtraction:** 
* **Complex number multiplication:** 
* **Complex number division:** 
* **Equality of complex numbers:**  means *a* = *c* and *b* = *d*.

Notice that both two-dimensional vectors and complex numbers **essentially are defined by two numbers**. Some of the arithmetic performed on each are the same (addition and subtraction), as is the meaning of equality; the other arithmetic operations are different.

Write a C++ program that will model both two-dimensional vectors and complex numbers using **derived classes**. Start with a base class, Pairs, which should model a pair of floating point numbers and include overloaded addition, subtraction, and is-equal-to operators in addition to appropriate constructors. Then, include two derived classes, Complex and Vect. Both classes will inherit their data from the base Pairs class.

* The Complex class should include overloaded multiplication and division operators in addition to appropriate constructors. It should also include overloaded **friend** insertion and extraction operators. (Note: You can use the Complex class you wrote for the last assignment and modify it slightly for this assignment.)
* The Vect class should include overloaded multiplication operators in addition to appropriate constructors. Note that it will need **two overloaded multiplication operators** – one to handle the dot product and a second to handle scalar multiplication. It should also include overloaded friend insertion and extraction operators.

**Note:** The assignment operator of your base Pairs class will not be inherited in your derived classes. To avoid problems that might develop because of this,

**Either:** include these statements within your Complex and Vect class definitions:

using Pairs::operator =;

**or:** overload the assignment operator = in your derived classes.

To exercise your new collection of classes, use this main program:

int main()

{

int sel;

do {

cout << "Select an option - (1) perform complex number actions\n";

cout << " (2) perform vector actions\n";

cout << " (3) exit\n";

cin >> sel;

if(sel == 1)

cActions();

else if(sel == 2)

vActions();

else if (sel == 3)

cout << "Bye...\n";

else

cout << "Invalid input - try again\n";

} while (sel != 3);

char ch; cin >> ch;

return 0;

}

You will need to write the stand-alone functions cActions and vActions – here is what each does:

* cActions – this function contains an array of 26 complex numbers and will repeatedly prompt the user to select one of four options:
  1. Enter a complex number
  2. Display a complex number
  3. Perform arithmetic or equality comparisons of complex numbers
  4. Exit the function

When the user selects options 1 or 2, the function asks where the number is to be stored or read from in the array, and then completes the selected action using that array location.

When the user selects option 3, the function will ask for the operation to be used. If the

operation selected is an arithmetic operator, the function will next ask for the array locations

of the numbers to be combined using the operator, as well as the array location of where the

result should be stored.

If the operation selected is the is-equal-to operator, the function will only ask for the array

locations of the numbers to be compared and will display the result of the comparison.

In all cases, the chosen operation is then performed.

* vActions – this function contains an array of 26 vectors and will repeatedly prompt the user to select one of four options:

1. Enter a vector
2. Display a vector
3. Display a scalar
4. Perform arithmetic or equality comparisons of vectors
5. Exit the function

When the user selects options 1 or 2, the function asks where the vector is to be stored or read from in the array, and then completes the selected action using that array location.

When the user selects option 3, the function will display the value of a scalar variable that is used to hold the result of a dot product, or the previous value of a scalar used to perform a vector-scalar product.

When the user selects option 4, the function will ask for the operation to be used.

If the operation selected is addition or subtraction, the function will next ask for the array

locations of the vectors to be combined using the operator, as well as the array location of

where the result should be stored.

If the operation selected is dot product, the function will next ask for the array locations of

the vectors to be combined using the operator, but will not ask for the result location, but will store the result in the scalar variable.

If the operator selected is vector-scalar multiplication, the function will ask for the array

location of the vector to be multiplied and the scalar value and the location of the result

vector.

In all cases, the chosen operation is then performed.

To see how this program will work, copy [S:\Coursework\Liu\a5-2.exe](file:///S:\Coursework\Liu\a5-2.exe) or the overloaded assignment operator version [S:\Coursework\Liu\a5-2 overload=.exe](file:///S:\Coursework\Liu\a5-2%20overload=.exe) to your own folder to run the program.

A sample run of the program:

Select an option - (1) perform complex number actions

(2) perform vector actions

(3) exit

1

Select an option:

(1) Enter a complex number

(2) Display a complex number

(3) Perform arithmetic or equality check

(4) Exit

1

Enter a complex number in a+bi or a-bi form: 1+2i

Where do you want to store this (enter A-Z): A

Select an option:

(1) Enter a complex number

(2) Display a complex number

(3) Perform arithmetic or equality check

(4) Exit

2

Which one do you want to display (enter A-Z): A

1+2i

Select an option:

(1) Enter a complex number

(2) Display a complex number

(3) Perform arithmetic or equality check

(4) Exit

1

Enter a complex number in a+bi or a-bi form: 3+4i

Where do you want to store this (enter A-Z): B

Select an option:

(1) Enter a complex number

(2) Display a complex number

(3) Perform arithmetic or equality check

(4) Exit

3

Enter an operation: + - \* / = \*

Enter first number (enter A-Z): A

Enter second number (enter A-Z): B

Enter result location (enter A-Z): C

Select an option:

(1) Enter a complex number

(2) Display a complex number

(3) Perform arithmetic or equality check

(4) Exit

2

Which one do you want to display (enter A-Z): C

-5+10i

Select an option:

(1) Enter a complex number

(2) Display a complex number

(3) Perform arithmetic or equality check

(4) Exit

3

Enter an operation: + - \* / = /

Enter first number (enter A-Z): A

Enter second number (enter A-Z): B

Enter result location (enter A-Z): C

Select an option:

(1) Enter a complex number

(2) Display a complex number

(3) Perform arithmetic or equality check

(4) Exit

2

Which one do you want to display (enter A-Z): C

0.44+0.08i

Select an option:

(1) Enter a complex number

(2) Display a complex number

(3) Perform arithmetic or equality check

(4) Exit

3

Enter an operation: + - \* / = =

Enter first number (enter A-Z): A

Enter second number (enter A-Z): B

1+2i does not equal 3+4i

Select an option:

(1) Enter a complex number

(2) Display a complex number

(3) Perform arithmetic or equality check

(4) Exit

4

Bye...

Select an option - (1) perform complex number actions

(2) perform vector actions

(3) exit

2

Select an option:

(1) Enter a vector

(2) Display a vector

(3) Display scalar

(4) Perform arithmetic or equality check

(5) Exit

1

Enter vector in <a,b> form: <1,2>

Where do you want to store this (enter A-Z): A

Select an option:

(1) Enter a vector

(2) Display a vector

(3) Display scalar

(4) Perform arithmetic or equality check

(5) Exit

2

Which one do you want to display (enter A-Z): A

<1,2>

Select an option:

(1) Enter a vector

(2) Display a vector

(3) Display scalar

(4) Perform arithmetic or equality check

(5) Exit

1

Enter vector in <a,b> form: <3,4>

Where do you want to store this (enter A-Z): B

Select an option:

(1) Enter a vector

(2) Display a vector

(3) Display scalar

(4) Perform arithmetic or equality check

(5) Exit

4

Enter an operation: + - . (dot prod) \* (scalar prod) = -

Enter first vector (enter A-Z): A

Enter second vector (enter A-Z): B

Enter result location (enter A-Z): C

Select an option:

(1) Enter a vector

(2) Display a vector

(3) Display scalar

(4) Perform arithmetic or equality check

(5) Exit

2

Which one do you want to display (enter A-Z): C

<-2,-2>

Select an option:

(1) Enter a vector

(2) Display a vector

(3) Display scalar

(4) Perform arithmetic or equality check

(5) Exit

4

Enter an operation: + - . (dot prod) \* (scalar prod) = .

Enter first vector (enter A-Z): A

Enter second vector (enter A-Z): B

Select an option:

(1) Enter a vector

(2) Display a vector

(3) Display scalar

(4) Perform arithmetic or equality check

(5) Exit

3

Scalar = 11

Select an option:

(1) Enter a vector

(2) Display a vector

(3) Display scalar

(4) Perform arithmetic or equality check

(5) Exit

4

Enter an operation: + - . (dot prod) \* (scalar prod) = =

Enter first vector (enter A-Z): A

Enter second vector (enter A-Z): B

<1,2> does not equal <3,4>

Select an option:

(1) Enter a vector

(2) Display a vector

(3) Display scalar

(4) Perform arithmetic or equality check

(5) Exit

5

Bye...

Select an option - (1) perform complex number actions

(2) perform vector actions

(3) exit

3

Bye...

3. Write a C++ program that uses virtual functions to input, calculate, and display some quantities associated with a variety of 2- and 3-dimensional objects. Design an **abstract base class** Shape from which you will derive classes Rectangle, Circle, Triangle, Box, Can, Cone, and Ball. The class Shape should contain **pure virtual functions** Display, GetDimensions, Area,Perimeter, and Volume. For each of the **derived classes**, write member functions Display, GetDimensions, Area,Perimeter, and Volume that will do the following:

1. Display - output the type and dimensions for an object of the class.
2. GetDimensions - get the dimensions for an object of the class.
3. Perimeter- Calculate and display the perimeter of an object of a 2-dimensional shape class; do nothing for an object of a 3-dimensional shape class.
4. Area - Calculate and display the area of an object of a 2-dimensional shape class; calculate and display the surface area for an object of a 3-dimensional shape class.
5. Volume - Calculate and display the volume of an object of a 3-dimensional shape class; do nothing for an object of a 2-dimensional shape class.

Include **appropriate constructors** and **any other member functions** that you think are necessary.

Write a program driver that will allocate an array of 20 pointers to class Shape and then repeatedly prompt the user, up to a maximum 20 times, to enter a shape type, **dynamically allocate memory** for an appropriate shape object, and then prompt for and read the appropriate dimensions for the shape. After the input of shapes is complete, the program should then **loop through** **the pointer array contents** and display the shape type, dimensions, area, perimeter and/or volume for each of the input shape objects.

**Formulas for the cone calculations:**

Dimensions: Height and base radius (*h* and *r*); Surface area = ; Volume = .

If you need any other formulas for the other shapes, let me know.

To see how this program will work, copy [S:\Coursework\Liu\a5-3.exe](file:///S:\Coursework\Liu\a5-3.exe) to your own folder to run the program.

**A sample run** (user input is in **bold**):

It is time to enter your shape selection and dimensions.

Enter the number of the shape type:

1 - Rectangle

2 - Circle

3 - Triangle

4 - Box

5 - Can

6 - Cone

7 - Ball

=>

**1**

Input for Rectangle - enter length and width (separated by space):

**1 3**

Select another shape? (y or n):

**y**

It is time to enter your shape selection and dimensions.

Enter the number of the shape type:

1 - Rectangle

2 - Circle

3 - Triangle

4 - Box

5 - Can

6 - Cone

7 - Ball

=>

**4**

Input for Box - enter length, width, and height (separated by space):

**5 2 1**

Select another shape? (y or n):

**y**

It is time to enter your shape selection and dimensions.

Enter the number of the shape type:

1 - Rectangle

2 - Circle

3 - Triangle

4 - Box

5 - Can

6 - Cone

7 - Ball

=>

**7**

Input for Ball - enter radius:

**3**

Select another shape? (y or n):

**n**

Next, a display of your selected shapes, dimensions. and other data

Shape = Rectangle Length = 1.0 Width = 3.0

Area = 3.0

Perimeter = 8.0

Shape = Box Length = 5.0 Width = 2.0

Height = 1.0

Surface area = 34.0

Volume = 10.0

Shape = Ball Radius = 3.0

Surface area = 113.097335544

Volume = 113.097335544

**What you need to turn in:**

1. **Source code listing:** A printed copy of the source code for each problem. Remember to include the name of each group member in a comment at the top of your source code. Be sure to follow the “Code Style Guidelines” specified in class.
2. **Source code files:** E-mail me your source code as attachments.
3. **Working in Groups:** Every student will work with two other students in our class on this assignment; all members of the group must contribute to the solution. **Turn in only one copy of the solution** – **clearly identify the name of each member on everything that you turn in.**
4. **Late Assignments:** Assignments are due before class on the specified due date (both the paper copies and the e-mail copies). If you wish to turn in the paper copy of an assignment after class, place them under my office door. Assignments turned in late will be assessed a 20% penalty per class day late.