Object Oriented Programming

Mini Project One

Session: Fall 2020

Faculty of Information Technology
University of Central Punjab
Lahore, Pakistan

Text Processor

You will be developing a Text Processor in C++. The tasks need to be performed are given below:

1. Reads a paragraph from a file, using the insertion operator >>

2. Write the paragraph to a file, using the extraction operator <<

3. Display the paragraph on screen, using the extraction operator <<

4. Checks if two paragraphs are equal, using the operator ==

5. Convert all characters of the paragraph to UPPERCASE or lowercase, must have a Boolean flag that determines if the text is in lowercase or UPPERCASE. Overload unary operator '+' for UPPERCASE and '-' for lowercase conversion.

6. Count the number of English and Punctuation characters in the paragraph

7. Count number of characters in every word of every sentence.

8. Count number of words in every paragraph, in every sentence, a word is terminated on a space character

9. Count number of sentences in the whole text, in every paragraph as well. A sentence is terminated on a full stop '.'

10. Encrypt and Decrypt the paragraph, must have a Boolean flag that determines if the text is in plain text or encrypted. Encryption is done based on key contained in the paragraph. For Encryption, this key is added to the ASCII value of each English character and Punctuation. English character at the end the rotated so that z with a key of 2 will become b. Similarly for Decryption the key is subtracted from each encrypted character. Overload unary operator % for Encryption and * for Decryption conversion.

11. Spellcheck words of the paragraph against a given dictionary. The dictionary is just a text file containing a list of words. To spellcheck, each word of the paragraph is checked if that word is present in the dictionary. All words that are not present in the dictionary are returned as a list of words.

12. Add two Paragraphs in to a single Paragraph, using binary operator +.

13. Compress and decompress the given text using Hoffman coding. Simplest way is to calculate frequency of each word in the text, assign a code to each word and replace each word in the text with the corresponding code. Coding is assigned such a way is that the word with the highest frequency is assigned the smallest number (code) and word with the lowest frequency is assigned the smallest number (code). Other the English character including punctuation remain as it. Decompression is done the same way in reverse i.e. replace each code with the corresponding word. You must have a Boolean flag that determines if the text is plain text or compressed. Also you will need to provide compression ratio after the compression is completed. Compression ratio = (number of character in original text)/(number of characters in compressed text). Overload unary operator ! for Compression and ^ for Decryption conversion

Task 1:

At first we will use a single class named **Paragraph** to implement the text processor. The very first task is to write the class definition for Paragraph using the following standard artifacts:

- A constructor that takes default parameters for each private variable, other than those that can be assigned or calculated
- Copy constructor
- Getter and Setter for each private variable
- **Don't provide function where you can overload operators, therefore must provide the following operators**:
    - Assignment operator =
    - Equal operator ==
    - Not equal operator !=
    - Insertion operator >> for reading from a file

  - Extract operator >> for writing in a file and writing on console as well (displaying on screen)
  - Unary operators as indicated above

Restrictions:

- You are not allowed to use getline to read the paragraph
- Instead of using a single character array for storing paragraph text, please use 4 dimensional character Array. First dimension to store a single character, second to store a word, third to store a sentence and fourth to store the paragraph.

Task 2:

Now instead of using a single class, we will use more than one class and see how this problem becomes more easy to manage. Now come up with the final solution which have separate class for every separate entity. Write the class definitions (.h files) for each class.

Task 3:

Provide the implementation (.cpp files) of the classes declared in task 2 and write a driver programme to test them.

Task 4: (Output file)

Your task is to provide output file named "Frequency.txt" which should contain frequency of paragraphs in whole text, sentences in each paragraph and frequency of words in each sentence of each paragraph.

For eg:

Paragraph number: 1

Sentences in Paragraph 1: 2

Words in Sentence no 1:5

Characters in word no 1 of sentence no 1: 3

Characters in word no 2 of sentence no 1: 2

Characters in word no 3 of sentence no 1: 4

Characters in word no 4 of sentence no 1: 5

Characters in word no 5 of sentence no 1: 3

Words in Sentence no 2:4

Characters in word no 1 of sentence no 2: 2

Characters in word no 2 of sentence no 2: 3

Characters in word no 3 of sentence no 2: 1

Characters in word no 4 of sentence no 2: 5