CE221 ASSIGNMENT 1 2015

Set by: Mike Sanderson

Credit: 20% of total module mark

Deadline: 11.59.59, Monday 16 November

Submission of this assignment will be via FASer; your programs will be tested during your lab in week 8 or week 9.

It is expected that marks and feedback will be returned by the beginning of week 11.

You should refer to pages 41-43 of the Undergraduate Students' Handbook for details of the departmental policy regarding late submission and plagiarism; the work handed in must be entirely your own.

Introduction

This assignment comprises two exercises: exercise 1 contributes 60% of the mark and exercise 2 contributes 40%.

Files for use in this assignment will be available online in a zip file.

Exercise 1 (60%)

The folder `ex1` in the zip file contains the `Time` class from part 4 of the lecture slides. Some extra operator functions have been declared in the header file. You are required to implement these in the `.cpp` file.

The six comparison operators should return boolean results (i.e. either `true` or `false`). A time `t1` should be regarded as being smaller than another time `t2` if it occurs earlier in the day. Note that some of these operators have been declared as member functions whereas others have been declared as friends of the class; this has been done arbitrarily to require you to implement both kinds of function. You must ***not*** make any changes to the header file.

It is recommended that you implement the == operator and one of the < and > operators directly and use those in the implementation of the other operators.

The `operator-` member function and the `-=` and `--` operators should behave in a similar way to the `+`, `+=` and `++` operators (but `t-n` should be the time `n` seconds before `t` instead of `n` seconds after `t`). If you choose to implement `operator-` directly and call it in the other operators the code for the other three will be almost identical to those in the lecture slides (apart from the fact that two of the functions have been declared as friends).

The `operator-` friend function with two `Time` arguments should return the elapsed time between its 2 arguments, in ***minutes***. For example 3.30pm-10.00am should have the value 330 (5 hours and 30 minutes is 330 minutes).

The first operand denotes the finish time and the second operand the start time so if the second argument is later in the day than the first you should assume that it refers to the previous day and hence 1.00am-11.30pm should have the value 90 (1.00 am is 90 minutes after 11.30pm). The answer should be given to the nearest minute if the elapsed time involves fractions of a minute.

A file containing a main function to test your class will be provided for the lab demonstrations – this will not be made available before the deadline, so it is strongly recommended that you write in a separate file a main function to test your code; this should not be submitted.

In order to allow my test class to load successfully, if you fail to complete any of the required functions you must instead provide dummy versions that return default values (e.g. `false` for the comparison operators and 0 for the `-` operator).

Exercise 2 (40%)

This exercise involves write a program to analyse the text of a play. The program will read the play, generate some information about the play, and provide the output both to the screen ***and to a file*** (whose name should be provided by the user).

The information to be provided by the program is

1 the total number of words in the script; and
2 the number of occurrences in the script of each word from a list of words that will be provided in a file.

The program should ask the user for the names of the file containing the play script, the file containing the search words and the output file. The program should terminate cleanly with an appropriate error message if any of the files cannot be opened.

For the purpose of this assignment "word" should be regarded as a sequence of non-white-space characters containing at least one letter. Any punctuation marks at the beginning or end of a word should be removed; punctuation marks in the middle of a word are to be regarded as part of the word. When comparing words you should ignore case, so `IT`, `it` and `it.` are all occurrences of the same word.

A file containing a list of words to be searched for will be provided for testing purposes.

The Play Script

The Gutenberg Foundation provides a large range of books in text form. They are usually out of print. They are provided by volunteers and are free. The text we are using is a Shakespeare play. The text is preceded by the "contract" between the user and the Gutenberg Foundation, which essentially says that the user will not use the text for financial gain. A précis of the contract is inserted several times into the text of the play, and once at the end.

Program Structure and Development

Your program must contain classes called `ReadWords` and `Writer`, each with a `.cpp` and a `.h` file, and a main function in a separate `.cpp` file.

Header files for these classes are provided in the `ex1` folder in the zip file; you must ***not*** make any changes to `ReadWords.h`. You must ***not*** change any of the existing material in `Writer.h`, but you may add extra content to this file.

It is suggested that you develop the program incrementally. First develop a simple version of the `ReadWords` class that does no punctuation removal and write a main function to test it using the file containing words to be searched for, displaying each word that is read. Next replace this main function with one that uses the play script looping through the words, counting the words in the script. You should then add code to remove punctuation from the beginning and end of words, using the `ispunct` function (which is declared in the header file `<cctype>`) to check whether a character is punctuation. Note that a word could contain more than one leading or trailing punctuation character, e.g `"Help!"`.

Next you should develop the `Writer` class which should be used for writing the information generated by the program to the output file and modify the main function so that it asks the user for the name of the output file and uses the `Writer` class to output the word count to the file as well as to the screen. (All output to the file must be done using calls to methods from the `Writer` class.)

You should then develop code to input the words to be searched for and to count the occurrences of these words. You may, if you wish, use an extra class for this.

A different word list will be provided for the lab demonstration; this list will not contain more than 10 words.

Deliverables

You should submit a single zip file containing two folders, `ex1` and `ex2`. The only file formats acceptable are `.zip` , `.7z` or a linux gzipped tar file. If you submit a file in any other format you will earn no marks for this assignment.

The folder `ex1` should contain the `Time.h` file (even if you have not modified the one that was provided) and your `Time.cpp` file (and nothing else).

The folder `ex2` should contain all of the `.h` and `.cpp` files for the exercise 1, along with a `.txt` file that was produced as output by your program using the word list provided.

Lab Demonstration

You should be ready to demonstrate your programs in the lab in week 8. However, due to the limited time available it is possible that not all of the demonstrations will be completed in that lab, so some may take place in week 9.

If you fail to demonstrate your programs without good cause 20% will be deducted from your mark. If you are unable to attend both labs due to extenuating circumstances you must notify me by the end of Friday of week 9.

Marking Scheme

20% of the marks for this assignment will be awarded for programming style and comments (10% for each). Your comments should say precisely what each function does and what each non-local variable represents; within function bodies you should use only brief comments to say what groups of lines do – you must ***not*** say what every statement does.

The total mark available for style and comments will be proportional to the amount of the assignment that you have attempted, e.g. if you attempt only about 60% of the assignment the mark available for style and comments would be about 12%.

Of the remaining 80%, 48 marks will be awarded for exercise 1 and 32 for exercise 2.

For exercise 1 about 18 marks will be awarded for the `operator-` member function and the `-=` and `--` operators, about 15 for the `operator-` friend function and the remainder for the comparison operators.

For exercise 2 about 18 of the marks will be awarded for correct word extraction and the total word count and the remainder for the occurrence counts of the search words.