SENG1000 – C/C++ PROGRAMMING

FOCUSED ASSIGNMENT 8 - ABOUT THE LAST MAJOR TOPICS

OVERVIEW

Write a program to give you practice at working with C-style strings, files, command-line arguments, and structs.

GENERAL COURSE OBJECTIVES ADDRESSED IN THIS ASSIGNMENT

- Use C-style strings and C-style string functions.
- Use file I/O.
- Use argc and argv.
- Use structs.
- Use best practices regarding Magic Numbers.

ACADEMIC INTEGRITY AND LATE PENALTIES

- Link to Academic Integrity Information
- Link to Late Policy

EVALUATION

- The evaluation of this assignment will be done as detailed in the Marking lecture from Week 2.

PREPARATION

- View Week 6, 10, and 11 videos.

REQUIREMENTS

Command-line Arguments

- When the program runs, it should have exactly four arguments on the command line.
    - Contents:
        - argv[1] contains a positive integer.
        - argv[2] contains a string that is no longer than 20 characters in length.
        - argv[3] contains a string that is no longer than 20 characters in length.
        - argv[4] contains a string that is no longer than 20 characters in length.

    - It is possible that the program might be tested with too many or too few arguments. In that case, display an appropriate error message and quit the program immediately.
        - When I mark, if the correct number of arguments are there, they will be valid values as described above.

Struct

- You must create a struct called MyData. It must contain the following fields:
    - an int that contains the value found in argv[1]
        - call this field howMany
    - a C-style string (i.e. array of char of size 21) that contains the value found in argv[2]
        - call this field theText
    - a C-style string that contains the value found in argv[3]
        - this contains information about the folder/directory in which you are to put the output file
        - call this field directoryPath
    - a C-style string that contains the value found in argv[4]
        - this contains the filename for the output file
        - call this field filename
- You must declare a variable called myArgs. This variable's data type is struct MyData.

Text File

- Write code to create a text file.
    - All lines in the text file will contain the string stored in the theText field of the args variable.
        - All lines must end with the \n character.
    - The number of lines in the text file will be equal to the number stored in the howMany field of the args variable
- The path and name of the text file is obtained from the directoryPath and filename fields of the args variable.
    - You must create a separate C-style string variable to contain the full location of the file. The full location is made up of the directoryPath, followed by a backslash, followed by the filename.
        - This must be of an appropriate size.
        - For simplicity, you can assume that the directoryPath found on the command line will never end in a backslash.

Other Requirements:

- There is **no** user input in this assignment. This does not include the command-line arguments, of course.

- You must use C-style text file I/O functions as mentioned in the File I/O lecture to create the text file.
    - You will have to use the #pragma that you used for getNum(), due to the compiler not liking fopen().
- All string handling will be done using C-style strings. Do not use the C++ string class.
- You must use error checking on **all** file I/O functions. If there is an error associated with any of the functions, display an appropriate error message indicating which function call failed and quit the program.
    - Although it would be normal to try to close the file upon an error writing lines to the file, you are not responsible for doing so.
    - Testing to try to make file I/O errors can be sometimes difficult. I will be testing for failure on opening the file and will be looking at the source code for other file I/O errors.
- You must use best practices for Magic Numbers.
    - When creating the constant for the length of the full location of the file, use the other constant(s) in the definition.
- Doing any of the following will result in an **automatic mark of 0** with no further feedback:
    - using C++ strings
    - using C++ file I/O
    - failing to use argc and argv
    - having a hardcoded filename (i.e. one that doesn't come from the command line but instead is explicitly mentioned within your code)
    - having user input besides the arguments on the command line
- All other usual course requirements (e.g. file header comment, initialize variables when declared) apply.

CHECKLIST REQUIREMENTS

- Create a requirements checklist. This should contain the specific requirements from this assignment as well as any relevant requirements that have been covered in lecture or that are found in the SET Coding Standards or SET Submission Standards. Do it in whatever form you wish. Hand in your completed checklist in PDF form as checklist.pdf. Not having this checklist will result in a cap of 80 on your mark.

GIT REQUIREMENTS

- You must use GitHub under GitHub Classroom for revision control of your source file. The details were mentioned in an eConestoga announcement earlier. You must click on the link for the Focused 8 Assignment in GitHub Classroom (mentioned in an eConestoga Announcement).
- Make git commits every time that you have something working or otherwise substantial.
- It is expected that you will make **regular** git commits with **meaningful** commit comments (describing the changes that you successfully made since the last commit).

  - On an assignment of this size, I would expect at least 5 distinct and meaningful commits. They should be done as you work on your code, so that they are separated by a reasonable amount of time (e.g. it would not be likely that you would complete this assignment in 10 minutes).
  - Take this requirement seriously.
- Failing to use GitHub Classroom will cap your mark at 60.

FILE NAMING REQUIREMENTS

- You must call your source file f8.cpp.
- You must call your checklist checklist.pdf.

SUBMISSION REQUIREMENTS

- Do not hand in any other source files besides those mentioned in the File Naming Requirements.
- Follow the instructions in the SET Submission Standards and the lecture on Submitting Assignments to submit your program. Submit both files to the correct Assignment folder.
- Once you have submitted your files, make sure that you've received the eConestoga e-mail confirming your submission. Do not submit that e-mail (simply keep it for your own records until you get your mark).

REMINDER

- Remember, you can use the Project Properties within Visual Studio to set up command-line arguments for testing. If you don't recall this, check the lecture.