

# CS 135 - Computer Science I

## Design Assignment 8 (DA8-11/16)

## Programming Assignment 9 (PA9-11/19)

As specified in your syllabus, you must turn your assignments in by 6:00 pm on the due date specified. If it is turned in late, but prior to 12:00 midnight the day it is due, credit will be reduced by 50% of the earned score. Any laboratories turned in more than 6 hours late will not earn any credit.

### Objectives:

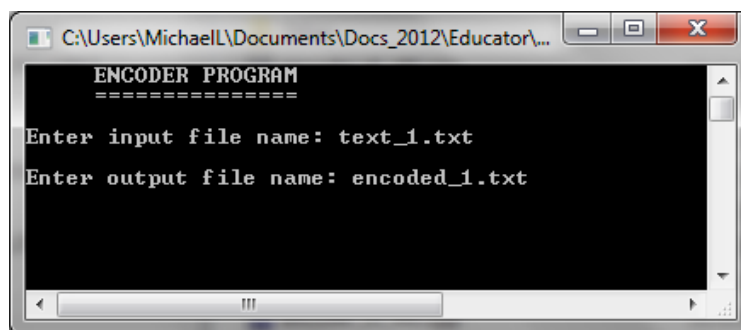
- 1) You will use a design procedure to develop a well-organized and modular program
- 2) You will use reference parameters to develop functions with multiple return quantities
- 3) You will use file input and output (I/O) to access and process data from a file
- 4) You will use simple one-dimensional arrays to read data from a file, store data for processing, and display results of a process
- 5) You will use and manipulate c-style strings; **string class/object values are no longer allowed**
- 6) You will use simple loops to read from a file, process data, and write to a file
- 7) You will use a function to conduct some specific file I/O operations

### Tasks:

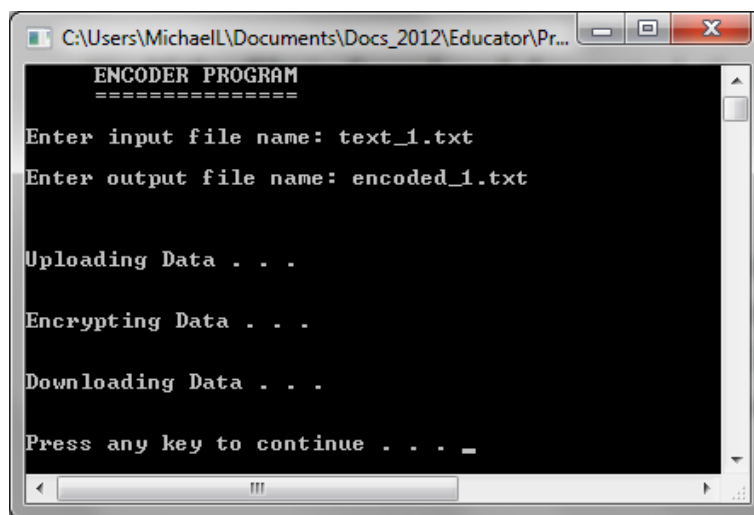
#### *Encoding Encrypted Files*

- 1) Create your program folder as needed. Your program name will be `encoder.cpp`. You must use `iostream` I/O operations such as `cout` or `cin` for this program.
- 2) You will also need to download a series of data files from `text_1.txt` up to and including `text_6.txt`. These files are the original data for last week's program. Again remember, if you download these as text and paste the data into a file, make sure you add an extra ENTER or two after the last row. This will be necessary for your file input operations.
- 3) You will now be developing the encoder for this problem. You now have the original text and you will need to encrypt it in such a way that your decoder program will decrypt it without significant modification. In fact the only modification that can be made to your decoder program is to make the size of your data array larger as needed (see Task #17). This should only require the changing of one global constant at the beginning of the program.

- 4) As before, the data is to be encrypted in two ways. First, the program must convert all the ASCII values from the input file to five or six digit *even* numbers for the two or three digit ASCII values respectively. Next, the program must randomly write approximately 3 five or six digit *odd* numbers between 33 and 126 (the ASCII range) for every one of the encoded even numbers. In other words, for every four numbers you write, one of them should be the even encoded ASCII value and the other three should be the odd dummy values. You must use a random process to do this, so the quantity of each written group of numbers won't be exact.
- 5) The program will prompt the user for the input file name (e.g., one of the six written files), and then the output file name (e.g., the new file you will be creating with your output). An example of the program operation is shown here:

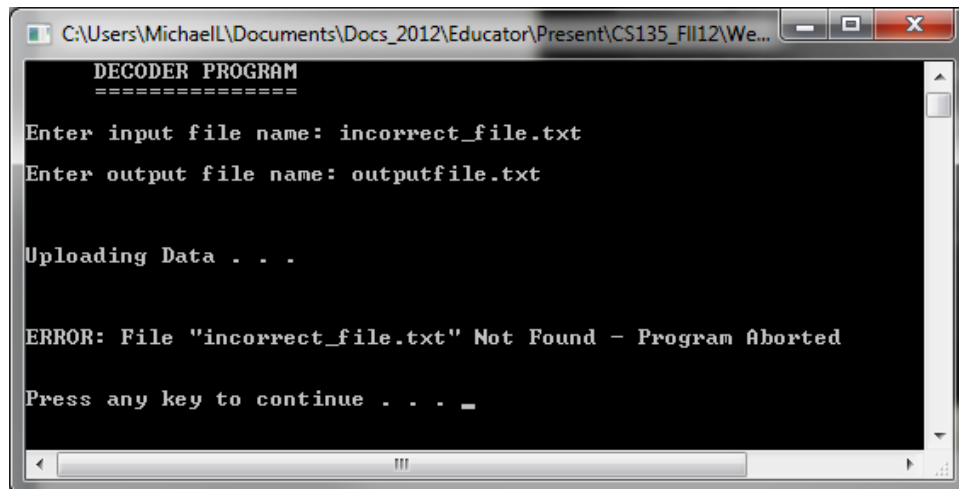


- 6) If the file is found and encrypted, the following screen must be displayed.



- 7) Note the difference in process reports during the program run. Each of these reports must be associated with the three operations specified later in this document. As before, they must only be displayed when the specified action is being conducted; they cannot be displayed together as a group of two or three messages.

- 8) If the input file is not found, the following screen must be displayed and the program must be shut down without conducting any further processing. It is acceptable for the reading process to be attempted (as shown in the display). However, if the reading process fails, the error message must be displayed and no other processing may occur after that point in the program. In addition, a failure message must be displayed and the program stopped if the input file has greater than 1,000 characters.



```
DECODER PROGRAM
=====
Enter input file name: incorrect_file.txt
Enter output file name: outputfile.txt

Uploading Data . . .

ERROR: File "incorrect_file.txt" Not Found - Program Aborted

Press any key to continue . . . _
```

- 9) Your program must first read the data from the file into a large c-style string. Then it must iterate through the string and convert the ASCII characters to 5 or 6 digit numbers as specified previously in this document. At the same time the process is converting these values, it must also be adding dummy odd-numbered values to the output array (again, as specified previously in this document).

- 10) After the data has been read and processed, it must write the integer values to the array in a formatted form, as show here.

82190	96011	101122	85001	48167
97824	115292	111342	117511	90447
119341	110990	32678	70693	78152
68637	74203	117356	48445	96667

- 11) Note the formatting of the numbers. Each value is a 5 or 6 digit number right justified in an eight character space. This must be accomplished by converting each number from an integer to a c-style string with the padding characters leading the digits. For example, using the first value shown above, the string must contain “space space space eight two one nine zero”; the next value would be “space space space nine six zero one one”; the next value would be “space space one zero one one two two”; and so on. The writing process should apply this conversion as it outputs the values.

- 12) You do not need to concern yourself with end line or space quantities as they will be converted and passed into the file. However, you are required to place the values in the text file as shown, being five values per row.

13) There are several opportunities to create functions in this program, so you must create and use at least ten functions. It is important that you develop them appropriately and effectively, with consideration for the following:

- a) you will use the **rand( )** function in your program, but it should never be found in problem solving code. Instead, it should be used to create functions that get you what you need, such as a random number between two limits, or an even random number, an odd percentage, and so on. Also, remember that to get reasonable randomness, you *must* place the **srand( )** function in the initialization part of the **main** function as follows:

```
srand( time( NULL ) );
```

- b) in addition to this, you should have a function that provides percentage odds, meaning it should return **true** for a possibility of **x** out of 100. For example a parameter of 25 would randomly return **true** 25 times out of a 100, a parameter of 60 would randomly return **true** 60 times out of 100, and so on. You will use this function to help you write approximately three odd numbered dummy values for every even number encrypted value
  - c) finally, for this program, you may not use any c-style string management functions such as **strlen**, **strcpy**, etc. You may also not use **getline**, although you may use **get**; you can find references to this function online; as a note, **get** is also explained and used in the laboratory activity
  - d) finally (finally), remember that ***you may not use any string class/objects anywhere in your program.*** To be clear, this means that if the identifier **string** is found *anywhere* in your code, you will lose credit, as specified in the General Usage Rubric
- 17) As before, you will need to know how large to create your arrays. You may assume that no string data set to be encrypted is larger than 1,000 characters, and remember that this includes spaces and punctuation. Any other array capacities in your program, such as your integer array capacities, must be adjusted with the 1,000 character capacity in mind. Use your probability values with an extra margin of safety to calculate the integer array capacities. As a note, since you may not always have control over the file used as input to your program, you must make sure that the program will exit elegantly if an input file size exceeds your input string capacity. It must stop, report the error, pause for the user to read it, and end the program without conducting any other operations.
- 18) Create a screen shot of each of the three conditions: 1) correctly input file name, processing, and completed program action, 2) an incorrectly input file name and the error message and program shut down, and 3) an attempt to encrypt a text file of more than 1,000 characters.
- 19) Also provide at least three encrypted files with your assignment folder. This is explained later in this document.

## ***Turning in your Design Assignment:***

### **Information:**

Week: 12

Laboratory: 12

Design Assignment: 8

Due Date: 11/16, 6:00 pm

### **To turn in:**

The first five steps of the Six Step Programming Process, including:

1. encoder\_s1.cpp
2. encoder\_s2.cpp
3. encoder\_s3.cpp
4. encoder\_s4.cpp
5. encoder\_s5.cpp

***Upload these as separate files.*** Note that following the instructions for uploading your work is critical; you will lose points if you do not follow these instructions. Do not upload any files or in any format other than that specified here.

For information on how to turn in Design Assignments, refer to the "How to Turn in Design Assignments" in the "General Course Information" folder

Important grading note: Due to the large number of students and the small number of people grading assignments, a fraction of the Design Assignments will be graded each week. If your DA is not graded that week, you will receive full credit; if it is graded, you will receive feedback and the appropriate grade. The random process will ensure that all students will be graded an equal number of times across the semester, and it also provides the possibility of getting graded more than one week in a row. You must do your best on each DA and assume you will be graded each week.

Important DA feedback note: If your DA does not get graded in one particular week, you should do the following:

1. At least look at the provided sample DA when it is uncovered on the Tuesday after the PA is due; make sure your structure, organization, and problem-solving strategies are comparable
2. Stop by one of the Instruction Team offices (i.e., Instructors, TAs, Tutors, etc.) and ask them to help you review your DA so you can be sure to do well when you are graded

# Turning in your Programming Assignment:

## Information:

Week: 12

Laboratory: 12

Programming Assignment: 9

Due Date: 11/19, 6:00 pm

## To turn in:

1. The Word file containing the following:
  - a. There should be at least three screen shots for the programs as specified in item #18 previously in this document
  - b. Remember to clearly annotate every displayed result
2. The executable file:
  - a. encoder.exe (encoder\_s6.exe is also acceptable)
3. The source code file:
  - a. encoder\_6.cpp
4. At least three encoded files produced from the six decoded input files:
  - a. encoded\_1.txt
  - b. encoded\_2.txt
  - c. encoded\_3.txt
  - d. encoded\_4.txt
  - e. encoded\_5.txt
  - f. encoded\_6.txt

These files must be compressed and uploaded as one zip file. To do this, select all of the required files, right click on them, and select “Send To”, then select “Compressed (zipped) Folder”.

Once the folder is created, it will be placed in the same folder in which you are working. Change the name of the zipped folder to “LastnameFirstname\_PAX” (where ‘X’ is the number of the Programming Assignment) as shown in the following example: “LeveringtonMichael\_PA3” (no quotes). After you have renamed the zipped folder, double click on it to verify that it has all the files it is supposed to have.

Note that following the instructions for uploading your work is critical; you will lose points if you do not follow these instructions. Do not upload any files or in any format other than that specified here.

For information on how to turn in Programming Assignments, refer to the "How to Turn in Programming Assignments" in the "General Course Information" folder