**COP 2334 – Programming Project 5**

# See Assignment for due date

# Project Outcomes

Develop a C++ program that:

* uses C++ functions for procedural abstraction,
* reads input from file,
* creates a user-defined class,
* uses C-style arrays, including partially filled arrays
* uses selection (if and if else) statements,
* uses iteration (for, while, and/or do-while loops), and
* follows standard programming practices.

# Project Requirements

1. Write a program that
   1. Reads a series of sequences of characters from a file,
   2. Stores each sequence of characters in an object of a user-defined MyString class.
   3. Stores each MyString object in an Array of MyString objects.
   4. Prints each MyString object to the screen.
2. Use top-down design and procedural abstraction to identify the functions to use in the program. The program should include functions for reading from a file and writing the results to the screen or to a file.
3. The program should **prompt the user for** **the input file name** that contains the sequences of characters. The format of the prompts is at your discretion
4. The sequences of characters are in a text file with the following format (see sample included with the assignment):
   1. A series of sequences of characters with each sequence separated by a newline.
   2. Example of the input file format ([newline] represents the newline character):

abcdefghijklmnopqrstuvwxyz

zyxvuwtsqr

acegilnoqsuwy

abcdefghijklmnopqrstuvwxyzabcdefghijklmnopqrstuvwxyz

z

[more sequences of characters...]

* 1. A sample input file is available with the assignment in Canvas.

1. The user-defined MyString class is a simple implementation of a string class similar to that in the C++ *string* library.
   1. The MyString class must have the following private member variables
      1. content
         1. a C-style array of type char.
         2. Created to hold a maximum of 32 characters
      2. length, an int that represents the number of characters in content array. The partially filled array value, not the size of the array.
   2. The MyString class must have the following public member functions:
      1. default constructor, which initializes the content and length member variables to appropriate values;
      2. getLength, which has no parameters and returns the value of the length member variable;
      3. pushBack,
         1. a single parameter of type char
         2. test if array is full
            1. if the array is not full

adds the char to the array at the length position.

Updates the length member variable

returns true - indicates the char was successfully added to the content array;

* + - * 1. if the array is full - returns false if array is full and indicates the char cannot be added.
    1. Print
       1. a reference parameter of type ostream (note that the parameter is type ostream and not o**f**stream, so that the print member function can write to either the screen or a file.
       2. Prints out the contents array to the ostream object.

1. A suggested plan to develop the program follows.
   1. Before the main function, declare the MyString class with the appropriate public member functions and private member variables.
   2. After the main function, implement the MyString class member functions.
   3. After the MyString class declaration, but before the main function, declare two functions for reading the input file and writing the MyString objects.
      1. The function that reads the input file should have parameters for an ifstream (reference parameter), an array of base type MyString, and an integer for the maximum size of the array. The function should return an integer that represents the actual number of valid MyString objects in the array.
      2. The function that writes the output should have parameters for an ostream (reference parameter), an array of base type MyString, and an integer that represents the actual number of valid MyString objects in the array.
   4. After the main function, write the definitions for the functions that read the input file and write the output.
      1. The read function should read, from the input file, each character in the sequence of characters **up to but not including the newline character (‘\n’)**. As each character in the sequence is read from the file, it should be stored in a MyString object by calling the MyString pushBack member function. Again, the newline character should not be stored in the MyString object.
      2. The write function should iterate through the array of MyString objects and write each object to the screen by calling the MyString print member function.
   5. In the main function
      1. declare an array of base type MyString with a size of 16, using a declared constant for the size;
      2. declare an ifstream variable for reading the input file;
      3. prompt the user to enter the name of the input file, open the input file and test that the file was opened successfully, exiting the program if it was not;
      4. call the read function, passing the ifstream, array of MyString objects, and the maximum size as arguments, storing the return value that represents that actual number of objects read from the file; and
      5. call the write function, passing cout as the ostream argument (to write to the screen), the array of MyString objects, and the actual number of valid objects in the array.
      6. The output to the screen should be a series of sequences of characters, just like that in the input file.

# Implémentation Notes

1. **Do not use the C++ array class in the program.**
2. You may assume that the input file is in correct format.
3. At first, test your program with simple input to verify that it is working properly. Then test with more complex input to verify that it will work with files with more than 16 sequences of characters or sequences with more than 32 character

# Important Notes

1. Projects will be graded on whether they correctly solve the problem and whether they adhere to good programming practices.
2. Projects must be submitted via Canvas/eLearning by the time specified on the due date. Projects submitted after that time will receive a grade of zero.
3. This is an individual project. Please review UWF's academic conduct policy that was described in the syllabus. Note that viewing another student’s solution, whether in whole or in part, is considered academic dishonesty. Further, allowing another student to view your code is considered academic dishonesty. Finally, submitting code obtained via the internet or other sources, whether in whole or in part, is considered academic dishonesty. All programs submitted will be reviewed for evidence of academic dishonesty, and all violations will be handled accordingly.