CS 11 Data Structures and Algorithms

Assignment 5: Dynamic Memory in Classes

Skip to Main Content

Learning Objectives

After the successful completion of this learning unit, you will be able to:

- Define syntactically correct classes that use dynamic memory in accordance with good programming practice
- Explain the "Big-Four" member functions that are required when a class uses dyamic memory.

Assignment 5.1 [15 points]

This assignment will not be graded for style issues. If your code works correctly, you'll get 15 points. Style will be graded next week.

See also **client program** and **correct output**.

Write a string class. To avoid conflicts with other similarly named classes, we will call our version MyString. This object is designed to make working with sequences of characters a little more convenient and less error-prone than handling raw c-strings, (although it will be implemented as a c-string behind the scenes). The MyString class will handle constructing strings, reading/printing, and accessing characters. In addition, the MyString object will have the ability to make a full deep-copy of itself when copied.

Your class must have only one data member, a c-string implemented as a dynamic array. In particular, you must not use a data member to keep track of the size or length of the MyString.

This is the first part of a two part assignment. In the next assignment you will be making some refinements to the class that you create in this assignment. For example, no documentation is required this week, but full documentation will be required next week.

Here is a list of the operations this class must support:

- A length member function that returns the number of characters in the string. Use strlen().

- Construction of a MyString from a const c-string. You should copy the string data, not just store a pointer to an argument passed to the constructor. Constructing a MyString with no arguments creates an empty MyString object (i.e. ""). A MyString object should be implemented efficiently (space-wise) which is to say you should not have a fixed-size buffer of chars, but instead allocate space for chars on an as-needed basis. Use strcpy().

- Printing a MyString to a stream using an overloaded << (insertion) operator, which should simply print out its characters. Use <<.

- Your MyString object should overload the square brackets [ ] operator to allow direct access to the individual characters of the string. This operation should range-check and assert if the index is out of bounds. You will write two versions of the [ ] operator, a const version that allows read access to the chars, and a non-const version that returns the client a reference to the char so they can change the value.

- All six of the relational operators (<, <=, >, >=, ==, !=) should be supported. They should be able to compare MyString objects to other MyStrings as well as MyStrings to c-strings. The ordering will be based on ASCII values. You can think of this as essentially alphabetical order; however, because of the way that ASCII values are defined, uppercase letters will always come before lowercase letters, and punctuation will make things even more complicated. Confused? You don't need to worry about any of this: just use the results of calling the strcmp() function. MyStrings or c-strings should be able to appear on either side of the comparison operator.

- Don't forget to include the big-3.

You may use all of the c-string functionality provided by C++. This will include the strlen(), strcmp(), and strcpy() functions, along with the overloaded insertion operator for c-strings. These functions are all covered in detail in the text. When you use strcpy() treat it as a void function despite the fact that it has a return value. Do not use strncpy(), strncat(), or strncmp() since they are not implemented in all versions of C++. You may NOT use anything from the C++ string class!!

Unfortunately, Visual C++ will, under its default settings, report an error when you try to use strcpy() or strcat(), even though they are standard C++. You can prevent this by adding this line as the first line in your file:

```
#pragma warning(disable:4996)
```

You must place your header file and implementation file in a namespace. Normally one would call a namespace something more likely to be unique, but for purposes of convenience we will call our namespace "cs_mystring".

Submit Your Work

Name your source code file(s) mystring.cpp and mystring.h. Execute the given client program and copy/paste the output into the bottom of the implementation file, making it into a comment. Use the Assignment Submission link to submit the source file(s). When you submit your assignment there will be a text field in which you can add a note to me (called a "comment", but don't confuse it with a C++ comment). In this "comments" section of the submission page let me know whether the program(s) work as required.