CS 11 Data Structures and Algorithms

Assignment 6: Dynamic Memory in Classes 2

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-three" member functions that are required when a class uses dyamic memory.

Assignment 6.1 [75 points]

See also **client program**, **data file**, and **correct output**.

This week you'll be making the following refinements to the class that you wrote in the last assignment. All the requirements from that class are still in force. For example, all MyStrings must always be stored in a dynamic array that is exactly the correct size to store the string. Your score on this assignment will take into consideration your work on both the previous assignment and this assignment.

1. Extraction Operator

Just like the >> operator that reads C-strings, your >> operator should skip any leading spaces and then read characters into the string up to the first whitespace character.

For reasons of convenience, we will impose a limit of 127 on the number of characters this function will read. This is so you can temporarily read into a non-dynamic array and then copy what you need into your data member, which will be a dynamic array. Note that this does not mean that all MyStrings will always have a maximum of 127 characters. For example, you might get a MyString with more than 127 characters by using the MyString constructor or by concatenating two MyStrings.

Hint: Don't try to read character by character in a loop. Use the extraction operator to do the reading of the input into a non-dynamic array, then use strcpy() to copy it into your data member. Make sure to allocate the correct amount of memory.

Hint: if you use the extraction operator as suggested above, you will not have to skip leading whitespace, because the extraction operator does that for you.

2. A read() function

The read() function will allow the client programmer to specify the delimiting character (the one to stop at instead of the first whitespace character). It should work just like the getline() function works for c-strings; that is, it should place everything up to but not including the delimiting character into the calling object, and it should also consume (and discard) the delimiting character. This will be a void function that will take two arguments, a stream and the delimiting character. It should not skip leading spaces. The limit of 127 characters imposed on the >> function above also applies to this function.

Hint: Don't try to read character by character in a loop. Use the in.getline() function to do the reading of the input into a non-dynamic array, then use strcpy() to copy it into your data member.

3. Concatenation Operator

Overload the + operator to do MyString concatenation. The operator must be able to handle either MyString objects or C-strings on either side of the operator. Be careful with the memory management here. You'll have to allocate enough memory to hold the new MyString. I suggest using strcpy() to get the left operand into the result MyString, and then strcat() to append the right operand. Both strcpy() and strcat() should be used as if they are void, even though they do have return values.

4. Combined Concatenation/Assignment Operator

Overload the shorthand += to combine concatenation and assignment. Only MyStrings can be on the left-hand side of a += operation, but either MyStrings or C-strings may appear on the right side. If you pay close attention to the += operator from the feetInches class, these may be the easiest points of the semester.

5. Add Documentation

Hints about reading input files:

I suggest that you copy the text from the input file webpage(s) and paste it into a file that you have created using your IDE. The files you create when you type in your IDE are always text files (even if they don't end with .txt). If you're using Windows, you could also use Notepad, but there's no reason to open another application when you are already working in your IDE. I strongly suggest that you don't use TextEdit (Mac) or Word, because these do not store files as text files by default.

Where to save your input file so that your IDE can find it:

In Visual C++, right click on the name of the project in the solution explorer. It appears in bold there. Unless you chose a different name for the project, it will be ConsoleApplication1. From the drop-down menu, choose "show folder in windows explorer". The folder that opens up will be the correct place to save your file.

In Xcode, while on the project navigator tab (looks like a folder) you'll see a yellow folder called Products. click the expansion triangle next to it and your project executable should show up. right click it and choose show in finder. The folder it's in pops up and you can add your input file to that folder.

Here is a tutorial video created by a former student about **creating and placing input files in the right spot for Xcode to find them**. It's an alternate method. I would suggest watching this video only if the suggestion above doesn't seem to be working for you. In order for these instructions to work, you must have executed a program in your project at least once. Otherwise the folder named "debug" that the video refers to will not exist.

Extensions are part of the file name. If you want to count the words in a file named "myfile.txt", typing "myfile" or "myfile.cpp" won't work.

For Windows users, before you start this assignment, I suggest that you make sure that Windows is showing you the complete file name of your files, including the extensions. Windows hides this from you by default. In Windows 7 and 8 the procedure is as follows:

1. Choose "control panel" from the start menu
2. Choose "Folder Options" from the list of control panel items. In Windows 8 you may have to type "Folder Options" into the control panel search bar.
3. Choose the "view" tab from the Folder Options window
4. Find the checkbox that says "Hide extensions for known file types".
5. Uncheck that checkbox.

I've received two different suggestions from student about how to do this in Windows 10. One student says that the procedure is exactly the same as that outlined above, except that the "Folder Options" window is now the "File Explorer Options" window. The other student says this: go to file explorer and click "View" on the toolbar, then go to "Options" on the furthest right, click on the drop down arrow and choose "Change folder and search options". Click "View" on the top and uncheck "Hide extensions for known file types".

Please let me know if you find that this doesn't work. Also let me know if you think that one of the two Windows 10 options is clearly better.

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.