G52CPP, 2017/18, Coursework 2, C++ Hangman, v1.0

Overview

Coursework 2 builds upon Coursework 1. You need to complete Coursework 1 first, and you will them modify your answer to form Coursework 2. This coursework aims to give you some initial experience with container classes, strings and file access in the standard C++ library – i.e. the C++ class way of doing things.

Requirements: (see the hints on the next page!)

1. Change your command line argument handling to accept zero or one command line argument. Change your code so that the random number seed is always initialised from the time rather than being passed in from the command line. If a single single parameter is passed on the command line then assume it is the filename of words to load (see requirement 2). If no command line arguments are provided, use the name "wordlist.txt" as the filename to load.
2. Replace the array of (string literal) words in your C version of the program by a vector of strings in your code. Create a text file called wordlist.txt, with a number of words in it, one word per line.
3. Change your program to load the words into your vector rather than using hard-coded strings, using an ifstream. Ensure that your program still works correctly if words are added to or deleted from the file (e.g. for marking).
4. Use string objects rather than char arrays or pointers to malloc'ed memory for the various strings in the program (e.g. the word being guessed, the list of available letters, the letters identified so far, etc). I suggest to go through each of these variables and change the type from char* or char array to string, then fix the code. Hint: changing the variable name at the same time will help you to find all occurrences.
5. Change from using the C-style output functions (e.g. printf()) to instead use the cout object instead.
6. Replace the console input function (e.g. getchar()) by the cin object so that user input is read using the cin stream object rather than the C-style functions.

Note: you may still use tolower() if you wish (I think it's probably the easiest way to do that conversion).

Additional considerations/requirements

- Use the C++ classes where appropriate, not the old C-style functions. (You get no marks for the coursework if you don't do this.) Note that I permitted use of the tolower() function if you wish.
- You should not need most of the C header files (see hints for what I needed)
- Your program should work as for coursework part 1, except that it now has different command line arguments and loads words from a file – see the demo version of CW2 that I created. If it does not work properly your mark will be severely penalised – hence why I suggest in the hints to change one requirement at a time so you maintain a working version in case of any issues.
- Ensure that the character checks are case-insensitive, as for part 1.
- You should not need the "`#define _CRT_SECURE_NO_WARNINGS`" at the top of your code any more.
- Ensure that you use new not malloc() if you need to do this, and that you delete all objects that you create using new (if you actually need to do this, see hints!)
- Ensure that your program does not crash and works well before you demo it

Submission and Marking criteria

This coursework is worth 6% of the module mark. You should be able to get at least most of it done in the lab sessions. There are 6 criteria above (for 6 marks, 1% each), so you should ensure that when you demo your program you can demonstrate that all 6 criteria were met in order to get the marks for the coursework.

In general your program will follow the same structure as for part 1. The six criteria above will be tested for the marking – so if it isn't mentioned it's not a marking criterion.

Submit your working code via moodle and have it marked in one of the labs. Note that the key thing is to get it marked (for instant feedback to you about your mark and what was wrong, if anything) – the moodle submission is a double-check in case of any issues later on, and a proof that you did the coursework.

The following text is hints and help, not requirements, so you can safely ignore it if you already know what you are doing.

Hints:

I suggest to start at your original hangman program – back up the initial version first then take the requirements one at a time and keep a copy after you have finished each one.

You will want to look at the following classes: vector, string, ifstream. I suggest to check which methods are available for these before commencing your coding. You may also want to check what operators can be used on a string object.

Consider all of the constructors for strings, and consider what you can do with strings – e.g. adding, changing, or erasing characters in them. Don't assume that a string in C++ is as limited as the String class in Java – particularly check what operators you can apply to it, and use this opportunity to experiment with it.

For requirement 3, create the vector, open the ifstream to load the file, read one word (or line, see getline()) at a time and store the word in the vector. Note the comments in lectures about arrays/vectors of objects rather than arrays/vectors of pointers – this will make it a lot easier for you.

For requirement 4: I renamed the variables when I did this, e.g. from arrAvailable (array of chars) to strAvailable (string), and pCurrentWord (pointer) to strCurrentWord (string) so that the compiler told me which code I needed to change (via errors, which Visual Studio underlines in red for me). You don't have to do so, but may find it helps you to find errors. Then it's just a case of creating/initialising the variables correctly and going through each error that Visual Studio underlines for you, changing the code appropriately.

I needed the following C++ class library header files: vector, string, fstream, iostream

I still needed the ctime header file, for time(), but none of the other C-library headers. I also included cctype, for tolower(), since although it still compiled without it, you should always include the relevant header files for functions that you call.

I didn't need the malloc and free or new and delete in my implementation so don't worry if you don't either.