EE273 - Engineering Design for Software Development 2

Week 9, Semester 1

Learning Outcomes:

- Further experience in defining and using appropriate project structures and good code layout
- Further experience in organising programs into multiple implementation and header files
- Ability to define custom data types and structures
- Further experience in defining functions
- Further experience in writing and calling functions within a C/C++ program and passing parameters by value, pointer and reference
- Further experience in passing and returning structure pointers
- Further experience in reading/writing from/to files
- Ability to understand and use linked lists.

Preface

- The lab sheets are released one week in advance of the appropriate laboratory session.
- It is key that students prepare in advance of the actual laboratory sessions; reading through the lab sheets, mapping out draft solutions and indeed trying out sample code.
- Students are expected at arrive at each lab session fully prepared for the lab exercises; the lab exercises are not designed to be completed solely within the lab sessions but are designed to be completed by the end of each lab session if appropriate preparation takes place.
- Students are expected to maintain a logbook as part of the laboratory work – taking notes, recording code and reflective comments about their solutions and progress to-date.

Part 1: Define a Person linked list structure in header file

In a header file that can be included in modules of source code, define a structure for representation of a Person. Define a structure that has in it a `string` and a pointer to another instance of the structure you are defining. (Hint: you will need to use a `typedef` statement somewhere!).

Do not forget to include the header file "boilerplate" directives `#ifndef …, #define …, #endif.`

Part 2: Read from a file and create a list (linked-list)

Create a text file that includes, on each line of the file, the name of a person. (Include at least 10 names).

Write a program to read in each of the names written in the above text file. Use `new` to create instances of the structure you defined above to store each of the names. Put these instances in a single linked list. (There is an example similar to this in earlier lecture).

Extend the Person module to declare and define a function that 'walks' through the linked list and write the names out to the screen.

Note: the function definition shall be implemented in a separate `cpp` file with the same name as the header file. For example if your header file is `Person.hpp` then the function definition should be in `Person.cpp`.

- Compile, Link and Run the program

Part 3: Sorting the List

In this last part, you will need to think a little bit about the way in which you could achieve the ordering before you start coding it. For example, how will you test whether one name stored in a string is alphabetically before or after another name stored in another string? Are there any functions in the string class that would help? (Look them up in the `strings` lecture, a textbook or online). Or do you need to extract the `char` array from the string object using the `c_str()` function and compare the first few elements of the `char` array in which the text is really stored?

Modify your program either

- so that the names are read from the file and added into a linked list in alphabetical order, or
- so that a linked list that has already been created is sorted into alphabetical order.

Use the function you wrote for walking through the list to show that the list has indeed been sorted into alphabetical order. (It, of course, should not already be alphabetical order in the text file!)

- Compile, Link and Run the program
- Try with data files from a colleague.

Recording Progress

- Print out listings of your programs and put in your logbook. Make a note in your logbook of anything that you found difficult but learned how to do (so that you can look it up again later in case you forget it). Make a note of the answers to all key questions.
- Demonstrate the operation of your code with a colleague and test the program. How well does your program perform compared with theirs?

Write some general reflective comments about the laboratory and speak to the teaching staff to discuss your progress.