KIT205 Data Structures and Algorithms

Week 2 Tutorial

Creating a new project

We will be using Microsoft Visual Studio in this unit. You should follow the instructions below when creating a new project.

1. Start Microsoft Visual Studio.
2. Choose *New Project…* from the *File* menu.
3. Select the *Visual C++* template and choose *Win32 Console Application*.
4. Enter a name for the project (e.g. *Week2*) and check the location at the bottom of the *New Project* window (you probably want to put it on your network drive in a folder for KIT205). Click *OK*.
5. When the *Win32 Application Wizard* starts hit the *Next>* button.
6. Choose *Empty Project* and uncheck *Security Development Lifecycle (SDL)*, and click *Finish*.
7. From the *Project* menu, choose *Add New Item…*
8. Choose a new *C++ File* and give it a name (e.g. *week2.c*). Make sure that you change the filename extension from *.cpp* to *.c*.
9. The new empty file should now be open. Write a simple program (that is, a *main* method) that displays "hello world" on the console.
10. Run the program without debugging (*Ctrl F5*) so that the console window isn't immediately closed.

Creating and Testing a Linked List

Next we are going to get a linked list up and running. You may find some of the instructions a little light on detail; that is deliberate. If you need extra help, ask your tutor – but try and work it out for yourself first. ☺

11. Add the linked list code from the lectures. The *struct*, *typedef* and function prototypes should go in a new header file called *list.h* and the function implementations should go in a new source file called *list.c*.
12. Test the linked list by modifying your main function so that it creates a list by adding some ints (*insert_at_front*) and then printing the list (*print_list*).
13. Test the other functions (*insert_in_order*, *delete_from_list*, *destroy*) as well.

- You will need to #include <stdlib.h> to be able to use NULL
- *If you declare your list as: List my_list = new_list(); you will need to pass &my_list as the first parameter to the other list functions*
- *Depending on your project settings, you may need to use scanf_s, instead of scanf (check online for more info)*

Testing Infrastructure

In this unit we will be doing a lot of console-based testing. The following steps will make a start on setting up some testing code (either comment your existing code, or create a new project before continuing).

14. Modify your main function (e.g. *week2.c*) so that you:
    a. Create a List
    b. Declare an int for storing input
    c. scanf into the int
    d. start a while loop that continues until your int has the value 0
    e. inside the while loop
        i. add the int to your list
        ii. scanf a new int
    f. destroy the list
15. Test the above version
16. Now modify your main to use a simple menu for selecting functions. The new main should:
    a. Create a List called *my_list*
    b. Create an int called *quit* to that tells the program when to stop and give it the value 0 (false)
    c. Start a while loop that exits if *quit* is true (*while (!quit){}*)
    d. inside the while loop
        i. Create an int called *option*
        ii. Print a prompt and scanf into *option*
        iii. if *option* has the value 0, set *quit* to 1
        iv. if *option* has the value 1, call a new function called *option_insert(&my_list)*
        v. if *option* has the value 2, call a new function called *option_delete(&my_list)*
        vi. if *option* has the value 3, call a new function called *option_print(&my_list)*
    e. destroy the list
17. Create the *option_insert*, *option_delete*, and *option_print* in the same file as main and add/copy code to manipulate the list using your list functions (e.g. *option_insert* will read an int and then call *insert_at_front*)
18. Test the new version

If you have time…

19. Add a list function called *reverse* that takes a list and returns a new list that is the reverse of the given list
20. Add a list function called *merge* that takes two ordered lists as parameters and creates a merged list that is also ordered
    a. See if you can do it without using *insert_in_order* or nested loops