

CSC2059 Data Structures and Algorithms

Practical 5: Lists

Wednesday 3rd November 2021

Launch Visual Studio. Create a new Solution Practical5 and name the first project List1.

Project 1. Linked List class expansion

In this project, we will add new member functions into the List class to expand this class. Download the provided template classes ListNode.h and List.h, and add these classes into the project List1.

(1) Inside the List class, add a new public function

```
void print();
```

which prints the data items of the list nodes in a list object, from the first node to the last node.

(2) Inside the List class, add a new public function

```
bool remove(int p1, int p2);
```

This function is an extension of the member function `int remove(int p);` which removes a node in a list at position p. The new function will remove a range of nodes from position p1 to position p2 (inclusive). It returns true for successful removal and false for illegal parameters p1 and/or p2.

(3) Create a test1.cpp in the project and use the following main function to test the above new member functions. The expected outputs are shown below the test function.

```
int main()
{
    // create an alphabet list
    List<char> chlist;
    for (int i = 0; i < 26; i++)
        chlist.insert_end(97 + i);

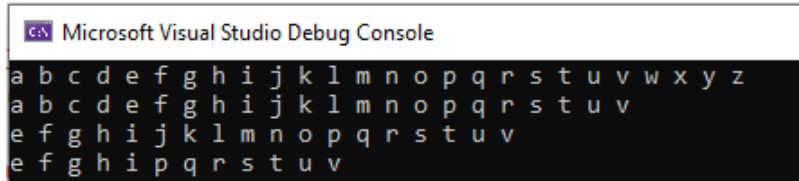
    // print chlist
    chlist.print();

    // remove nodes from 22-25
    if (chlist.remove(22, 25))
        chlist.print();

    // remove nodes from 0-3
    if(chlist.remove(0, 3))
        chlist.print();

    // remove nodes from 5-10
    if(chlist.remove(5, 10))
        chlist.print();

    return 0;
}
```



```
Microsoft Visual Studio Debug Console
a b c d e f g h i j k l m n o p q r s t u v w x y z
a b c d e f g h i j k l m n o p q r s t u v
e f g h i j k l m n o p q r s t u v
e f g h i p q r s t u v
```

Project 2. Linked List for storage of data with an unknown size

In Solution Practical5 add a new project List2, in which create a source file test2.cpp. Download the provided corpus.txt file and place it into the List2 folder along with the test2.cpp file. Add the template classes ListNode.h and List.h into the project.

The following shows an incomplete program, showing the use of a linked list (an object of our List class) for loading data of an unknown size – the corpus file mentioned in our lecture - into the program. The program uses C++ file I/O classes for opening the corpus file for reading the data into a linked list, each list node storing a word. To use the class you need to #include <fstream>.

As you can notice, the corpus contains many word items which include some non-alphabetic character(s), e.g., \', \-, \., \., \", \&, \%, \; There is also a sentence code (e.g., 011a0101) at the beginning of each sentence which is not a word. You should ignore all these items and only store the **proper** words, in which each character is an alphabetic letter, in the list.

You are asked to complete this program, in test2.cpp, by completing the missing parts indicated by points 1-4 in the program. After execution, your program should print out the number of the proper words in this corpus file.

Hit: You will find it useful to write a function for testing for proper words (and non-proper words):

```
bool isWord(string item);
```

The incomplete program with missing parts indicated by 1-4

```
int main()
{
    // allocate a string-type list pointed to by pc, used to store all
    // words in the corpus
    List<string>* pc = new List<string>;

    // open the corpus.txt file for input
    ifstream fi("corpus.txt");
    string item;

    // read each item in the file separated by space until the end of the file
    while (fi >> item) {
        // 1. test if the read item is a word - with only letters and
        // no other characters

        // 2. if yes insert the word into the list pointed by pc
    }
    // close the file
    fi.close();

    // 3. print the size of the list
    // 4. free the memory used to store the corpus

    return 0;
}
```

Project 3. Timing operations for array list and linked list

In Solution Practical5 add a new project List3, in which add a source file test3.cpp. Copy the corpus.txt file into the List3 folder with test3.cpp for convenience of its reading into the program. Add the template classes ListNode.h and List.h into the project.

- (a) In our lecture video and code, we showed an example of comparing the timings of the insertion operation between the array list and linked list, to compare their performance. Following the lecture example, in this project we first time the deletion operations for the array list and linked list. The following shows two functions for array list insertion and deletion operations. This is followed by a main function, which outlines a plan to time the deletion operations for an array list of a pre-defined Max_ListSize, and a linked list, assuming both lists holding the same data of corpus.txt.

You are asked to convert the testing plan into testing programs in test3.cpp. Then run the tests for variable conditions to compare the performance of the two list data structures for deletion operations.

```
// maximum size of array list e.g. 2^20
constexpr int Max_ListSize = 262144;

// insert an item at position p, into an array list with size items pointed by List
template<typename T>
T* insert(T* list, int& size, int p, T item)
{
    if (p < 0 || p > size) {
        cout << "Unable to insert at " << p << endl << endl;
        return NULL;
    }
    else if (size >= Max_ListSize) {
        cout << "List full" << endl << endl;
        return NULL;
    }

    for (int i = size - 1; i >= p; i--)
        list[i + 1] = list[i];
    list[p] = item;
    size++;

    return list;
}

// remove an item at position p, from an array list with size items pointed by List
template<typename T>
T remove(T* list, int& size, int p)
{
    if (p >= 0 && p < size) {
        T item = list[p];
        for (int i = p + 1; i < size; i++)
            list[i - 1] = list[i];
        size--;
        return item;
    }
    else {
        cout << "Unable remove at position " << p << endl;
        exit(1);
    }
}
```

```

int main()
{
    /* array list */
    // 1. generate a string-type array list, alist, of Max_ListSize
    //     initialise size of alist, size, to 0

    // 2. open the corpus file: ifstream fi("corpus.txt");
    //     read each item in the file & insert each proper word at the end of alist
    //     close the file: fi.close();

    // 3. time deletion operations for alist
    //     e.g. remove 100 words at the front, middle or end of alist

    /* linked list */
    // 1. declare a string-type List object llist

    // 2. open the corpus file: fi.open("corpus.txt");
    //     read each item in the file & insert each proper word at the end of llist
    //     close the file: fi.close();

    // 3. time deletion operations for llist
    //     e.g. remove 100 words at the front, middle or end of llist

    return 0;
}

```

- (b) Next, we time the operations of traversing a linked list. Rename the above main function to main1, and copy the following main function into test3.cpp. Part 1 of the main is to be completed by you by copying your code from task (a) above.

Part 2 and Part 3 show two methods to traverse the list, from the first node to the last node, to find the longest proper word in the list (you will use the isWord function defined in Project 2). The first method (Part 2) uses the member function **get(int p)** to get the item value at each position p, for p = 0 to listSize – 1. The second method (Part 3) first sets the pointer pCurrentNode to pFirstNode, then steps through the list by calling the member function **get_next()**, each call returns the item value pointed to by pCurrentNode and advances pCurrentNode to pCurrentNode->pNextNode. At the end of each part, the code prints out the longest word found in the corpus.

Time the operations of Part 2 and Part 3 to compare how much time will be needed by each method to find the longest word.

Warning - Part 2 may take a long time to finish but Part 3 is fast!

Make sure you understand what causes the big speed difference between these two methods.

When you run the Part 2 code, you should comment the Part 3 code, and vice versa.

```
int main()
{
    // 1. use a linked list - llist, to store corpus.txt
    /* To be completed by you */

    // 2. traverse llist using get(int p) to each position p in the list to
    // find the longest word
    string word;
    unsigned int wordLen = 0;
    for (int p = 0; p < llist.size(); p++) {
        string item = llist.get(p);
        if (isWord(item) && item.size() > wordLen) {
            wordLen = item.size();
            word = item;
        }
    }
    cout << "The longest word is: " << word << endl;

    // 3. traverse llist using set_first & then get_next to each position in the
    // list to find the longest word
    string word;
    unsigned int wordLen = 0;
    llist.setp_first();
    for (int p = 0; p < llist.size(); p++) {
        string item = llist.get_next();
        if (isWord(item) && item.size() > wordLen) {
            wordLen = item.size();
            word = item;
        }
    }
    cout << "The longest word is: " << word << endl;

    return 0;
}
```

[A mock test for Practical Test 1](#)

Practical Test 1

Practical Test 1 will take place from the timetabled Practical session time on Wednesday 10th November. The test will cover the taught material from Lecture 1 to 5, and practical material from Practical 1 to 5. More details about the test arrangement can be found on CSC2059 Canvas.

Take the following as an exercise. Using this Practical 5 as a mock test, practice how your test will be timed and how to upload your submission at the end of the test to Canvas.

On the Canvas Page for CSC2059, click Assignments and then click CSC2059 Practical 5 (the mock test). Click **Take the quiz** to start the mock test. Carefully read the instructions in the test.

At the end of the test document, you will see instructions for how to submit your work to Canvas for marking. For Practical 5, the instructions are:

Submission Notes

- 1) Save your projects.
- 2) Make sure that you have the following FOUR source files ready for submission:
test1.cpp, List.h
test2.cpp
test3.cpp
- 3) ZIP the above 4 files, and DO NOT INCLUDE ANY OTHER FILES.
- 4) Name the zip file to XXXXXXXX.zip, where XXXXXXXX is your student number.
- 5) Check that your .zip file contains the above 4 source files (double click on the .zip file. You do not need to extract the files again).
- 6) In case something goes wrong with your submission, first make a backup of your zip file on to a USB drive, and/or email it to yourself.
- 7) On the Canvas Page for CSC2059, click Assignments and then click CSC2059 Practical 5. Go to Question 2 'Upload your work here'. Click Choose a file and then locate the zip. Click Open to upload it. Finally click **Submit quiz** to see your file on Canvas.

The mock test submission won't marked.

If you have any questions, ask a demonstrator or a teaching staff face to face in the labs, or online by using MS Teams, during the timetabled practical session time. At any other time outside the timetabled session time, you can send your questions by using the Ticketing System, or by using emails.