Objected-oriented Programming Report

Assignment #3

|  |
| --- |
| **Professor**  김영민 교수님 |
| **Department**  Computer engineering |
| **Student ID** 2013722004 |
| **Name** 최민기 |
| **Class** 설계 : 1(Mon) , 2(Wed) / 실습 : 3, 4 (Thu) |
| **Date**  2017. 4. 28 (Fri) |

1. Read the input.txt file and make a list of all the words in this file without duplicates. Each word is stored as an Item object in an Item object array table. For each word, a list of all the words that have appeared immediately after the word and the number of appearances are stored in the head of the Item object that expresses each word in a linked list of follower objects. For example, suppose the contents of a data file looks like this:

**Result Screen)**

|  |
| --- |
| **[input.txt]** |
|  |
| You can see the result of outputiing a string of data length, copied string, and delimeter. |

**Explanation)**

This program reads the "input.txt" file and constructs a single link list with the following words for each word. First, read Notepad and use strtok to find the word and create a linked list corresponding to the word. Thus, the Item object array consists of the head of the Item and the following words.

**Consideration)**

At first I knew that the List data structure was important, but I had not tried it. I was embarrassed by the fact that there was a list starting with task # 1, but since it was kindly presented in the lecture material, I learned how to use the List data structure. It was not difficult to solve the problem of creating a list for each word and connecting it to a single word.

1. Write a Mobile phone sales list management program using a Circular Doubly Linked List. The attributes of the managed product are the product name, the brand, and the price. The management functions of this program are as follows:

Insert product at beginning

Insert product at last

Insert product at position

Delete product

Update product

Search product

Sort by price in ascending order

Display product list

Reverse product list

**Please refer to the examples for detailed implementation of the function. Result Screen)**

|  |
| --- |
| **[Page1]** |
| **[Page2]** |
| **[Page3]** |
| Depending on the result of the execution, you can see that the result is in the List according to position |
|  |
| You can see the message that “Galaxy Note5” located at Position 1 is deleted. |
|  |
| You can see that Price has been changed with Search command by modifying Price with Update command. |
|  |
| You can also see that the output of the list has been modified according to the intermediate command. |
|  |
| You can see that the List is reserved according to the Reserve command. |
|  |
| You can also see that they are ascending by price in Sort command. |

**Explanation)**

This program used the List data structure. Unlike Problem # 1, unlike Singly Linked List, it is composed of bidirectional Linked List. Because it is connected in both directions, Next and Prev exist when searching Node. You can point to the head position and point to the tail position. Because it is composed of circular Linked List, searching is fast because it can search backward and forward.

**Consideration)**

The first task was easy to do by connecting Node and Node in one direction without previous search. However, task # 2 was difficult because it had to be connected in both directions and set head and tail. As I was drawing a picture with A4 paper, I was able to understand it easily by picking up one List connection structure one by one. I was able to catch up on the List as I went through Problems 1 and 2.

1. Write a land occupation program. The representation of each land is as follows:

The land of Korean soldiers is 1

The land of Japanese soldiers is 0

Empty land is -1

Japanese soldiers can not occupy the land of Korean soldiers, and they are only occupied. After a day, Korean soldiers occupy land in east, west, south, north, and diagonal directions. The occupied land is changed to 1 and the empty land is always -1. Find the date when Korean soldiers occupies all the land of Japan. The date when Korean soldiers occupied all the land of Japanese soldiers:

**Result Screen)**

|  |
| --- |
| Input  Output |
|  |
| Since the input takes 4 days to be occupied, you can see that the output is 4 days. |
|  |
| [한국어](javascript:void(0))에서 번역  의 번역 표시  [해당 Input은 점령하는데 6일이 걸리기 때문에 출력결과가 6일인 것을 확인할 수 있다](javascript:void(0)) 대신 번역  이것을 찾으셨나요?  Because the input takes 6 days to be occupied, you can see that the output is 6 days. |
|  |
| It displays a result screen indicating that it can not be occupied because the Korean land does not exist at all. |
| You can see the result of the exception processing because the Korean land is clogged to a place that can not be occupied. |
|  |
| As all lands are occupied by Korea, you can see that there is no land to occupy. |

**Explanation)**

This program has a land of ROK 1, a land that can not be occupied by -1, and a Japanese land of 0. After a day, the Korean army occupies the Japanese land one by one in the direction of the four. When the occupation is completed, it displays the day of occupation. The size of the ground is input from the user and the shape of the ground is inputted from the user. Because there is a land that can not be occupied, an exception should be treated accordingly.

**Consideration)**

At first I had a lot of ideas on how to implement this program. I thought it would be easy to implement using a two-dimensional array, but that's not the intent of the problem. It took me a long time to learn about the STL Queue and to figure out how to implement it in the Queue. At this time, I thought of the “bfs” algorithm. The “bfs” algorithm is the algorithm used in the maze search algorithm. I realized that the land I could not occupy was a wall, so I could easily implement it. I have never used STL functions while coding, but I think I have more knowledge by using this opportunity.

1. Write a program that automatically deallocates an instance which is dynamic allocated when any variable refers to the instance. For the automatic deallocation, write the Ptr<T> class. The Ptr<T> class should print the reference count when the reference count is changed. Note that **1)**The Ptr<T> class should be written as a template class; 2**)**The Ptr<T> class has a public static method, **Create()**,which dynamic allocates for type T; **3)**Do not change the steps of the main function, then the program would show the results which are the same as the [**example results**];

**Result Screen)**

|  |
| --- |
|  |

**Explanation)**

This program is a program that automatically dynamically allocates and deallocates when a reference is made. Reference counting occurs automatically when a reference is made. I use the concept of templates and the concept of operator overloading because we are using a class with a specific type.

**Consideration)**

Problem # 4 took a lot of time to understand. The concept of operator overloading and the notion of templates were unfamiliar. Also, it is hard to understand how the program works even if you look at the example result screen. In reference counting, it took a long time to understand the concept of accurately counting the number of pointers to the same object. Also, it is difficult to implement a part that automatically releases the dynamic allocation when the reference count becomes 0. I can also see that using this algorithm has the effect of reducing memory usage.

1. Write a program that manages the information for addresses. The program reads the data for the people and the houses from **input.txt** at the first. For the management, write Person and House classes. The Person class should have a name, and a house where the person lives in. The House class should have an address and the people who live in the house. At the last, write AddrManager class to manage the information. Note that **1)**The Person, House, and AddrManager classes should be declared in **Person.h**, **House.h**, and **AddrManager.h**, respectively; **2)**A Person should have a house; **3)**A House should manage the people who live in the house using std::list<T>; **4)**The AddrManager class should have a list of people and a list of houses as member variables using std::map<TKey, TValue> to find a person and house with name and address, respectively; **5)**Do not make more than one instance of the Person or the House with the same name or the same address, respectively; **6)**Handle all exception which can be occurred due to user input excluding the exceptions which can be occurred due to input file.

**Result Screen)**

|  |
| --- |
| **[input.txt]** |
|  |
| With the Show People function, you can see that you are outputting a house that lives with everyone. |
|  |
| You can see the output of people living with all the houses through the Show House function |
|  |
| You can confirm that you have changed the name using the Change Name function. |
|  |
| You can confirm that you have changed the name of your house using Change Address function. |

**Explanation)** This program used a structure that connects people living at home using the map and list in the STL. If there is a map for House and Person, and there are several people in a house, use Person map as List. Since map has a key value, you can quickly access the node you are searching for.

**Consideration)** I used STL map and list for the first time through task # 5

I have been told that when I insert data into the map, it is automatically sorted, but I have confirmed that it has not been sorted. The reason is that it was not sorted using a chracter type array without using a string datatype. As we proceeded with this assignment, I found that STL was very convenient. Also, in implementing the function # 3 and # 4, I can see that the key value in the map can not be changed in the data modification part, so we have to use the method of having the temporary address value and putting it in the map again. I was able to find that it was convenient to use STL while doing task 5.

1. Write a program for the SUDOKU game with Microsoft Foundation Class (MFC) framework. The program reads **input.txt** to initialize the SUDOKU board at the first. The 0 in the input file refers to an enable block which user can write a number. If mouse double click event is occurred at an enable block, the block is selected and the color of the selected block is changed. If key down event is occurred after an enable block is selected, there can be three cases; **1)**When key down event is occurred with correct number (1 ~ 9), the number should be marked at the block; **2)** When key down event is occurred with Delete key or ESC key, the number which has already been marked at the block should be gone; **3)**When key down event is occurred with any other keys, the block should be unselected; If there are duplicated numbers in vertical lines, horizontal lines, and the 3x3 blocks, the numbers should be showed in red. When the game is over, the message box for Game Over should be shown.

**Result Screen)**

|  |  |
| --- | --- |
|  | Initial screen  Sdoku Game Start |
|  | When mouse double click occurs, it turns green. |
|  | If there are duplicate numbers, you can see that they are numbered in red. |
|  | Press the ESC button to erase the entered number |
|  | If the Sudoku board is full of duplicate algorithms, you will see a popup message. |

**Explanation)** This program uses MFC to create a Sdoku Game. Read the file "input.txt" and print the number on the Sdoku board, which consists of 9x9 spaces. Also, if there are duplicate numbers in the vertical lines, they are treated as red. Sdoku Game will end when the number is entered according to the rules of the game.

**Consideration)** It was too awkward to use MFC for the first time. I did not even know how to use the MFC classes. Checking duplicate numbers is only 1, and the Sudoku algorithm rule is not applied properly. I still understood this through the use of this MFC. After finishing this lesson, I would like to finish if it is later.

**Reference**)

Assignment\_lecture\_1 ( 김영민 교수님, 한만기 조교님, 고은정 조교님 )

Assignment\_lecture\_2 ( 김영민 교수님, 한만기 조교님, 고은정 조교님 )

Assignment\_lecture\_3 ( 김영민 교수님, 한만기 조교님, 고은정 조교님 )