Object Oriented Programming

Project

Submission date: 2018-06-17

학 과: 컴퓨터정보공학부

담당교수: 김영민 교수님

실 습: 수강/수요일

학 번: 2017202082 성 명: 류 화 랑

1. Introduction

This project is to create an address book that contains information about students. All of the students ' information consists of linked lists. The department node and the name node are inherited from the node class. The composition between the node class and the department node and the name node uses polymorphism. Students ' information is also linked to the right, left, and front nodes of the name node. The department nodes are a circular list, and the name nodes are connected in both directions. You can read commands from text files to change, save, and manage student information.

1. Flowchart
   1. Department List, Department student name down List (2D Linked List)



* 1. Name List



* 1. Contact List



* 1. Grade List



* 1. StudentID List



* 1. Main function



* 1. Manager LOAD



* 1. Manager ADD



* 1. Manager UPDATE



* 1. Manager MODIFY\_DEPARTMENY



* 1. Manager DELETE\_DEPARTMENT



* 1. Manager PRINT



* 1. Manager PRINT\_STUDENT\_ID



* 1. Manager PRINT\_CONTACT



* 1. Manager FIND\_NAME



* 1. Manager SAVE



1. Algorithm
   1. Department List, Department student name down List (2D Linked List)

First, the new student's information is dynamically allocated to the new node. Then connect the student's information to the right node of the name node, the left node, and the front node. If there is no head node, set up the new department as the head node and associate it with the name node. If there is a head node, divide the cases to link the department nodes in ascending order.

Repeat loop by size of department linked list.

If the new department is larger, search the following department nodes. At this time, if the department you are searching for is the last node, insert a new node at the end of the list.

If the new department is smaller, a new node is inserted in the middle. At this time, if the department node you are searching is the head node, set the new node as the head node and insert it at the top of the list. If it is not a head node, insert it literally in the middle.

If a node with the same department name comes in, students should be added under that node.

At this time, the case is divided into three.

If the new name is larger, continue to explore the next node. At this time, if you need to add at the end of the name, add a new name node to the end of the name.

If the new name is smaller, it should be inserted in the middle. At this time, the smallest name in the department is added to the top, and if it is necessary to insert the name node in the middle of the course, insert it in the middle.

If the new name is the same as the new one, the new one is treated as if it were small.

When the loop is finished, search the last node and configure a circular list.

* 1. Name List

If there is no head node, set the head node. Otherwise, continue to add the name node in ascending order. Repeat loop by size of department list.  
If the new name is larger than the name being searched, continue to search for the next node. At this point, create a node at the end of the name list if you need to insert it at the end.

If the new name is smaller, it should be inserted in the middle. If it is necessary to insert directly in front of the head node, set the new name node as the head node, and, otherwise, insert it in the middle.

If there are same names, the newly introduced name node will be treated the same as if it were smaller.

* 1. Contact List

If you do not have a head node, set the contact number of the person in front of the name node as the head node. Otherwise, repeat the name linked list to the end and continue inserting it into the next node of the contact linked list.

* 1. Grade List

If you do not have a head node, set the Grade of the person in front of the name node as the head node. Otherwise, repeat the name linked list to the end and continue inserting it into the next node of the Grade linked list.

* 1. StudentID List

If you do not have a head node, set the StudentID of the person in front of the name node as the head node. Otherwise, repeat the name linked list to the end and continue inserting it into the next node of the StudentID linked list.

* 1. Main function

Opens Command text file in a read format and the LOG text file in a write format. Repeat loop until end of file in command text file. It generates tokens based on spaces by reading the Command text file one line. Then, if the token and command match, execute the functions of the manager class. If it is necessary to separate more tokens, separate them again and pass them to the function parameter.

* 1. Manager LOAD

Open the information text file in a read format, move the cursor to the end of the file, and open it in a write format. If the information text file has not been opened, print the error code 100 and exit.  
Repeat the loop to the end of the file, read one line after another, and detach all tokens. Comprises the linked list by dividing the separated tokens into parameters in the insert function of the department and linked list. After you finish the loop, configure the Grade, studentID, and contact linkedlist.

* 1. Manager ADD

Open the Add\_information text file in a read format, move the cursor to the end of the file, and open the LOG text file in a write format. If the information text file has not been opened, print the error code 200 and exit.  
Repeat the loop to the end of the file, read one line after another, and detach all tokens. At this time, if the number of students or contact numbers you have read are duplicated, an error code 200-1 is printed and the student is not added. Comprises the linked list by dividing the separated tokens into factors in the insertion function of the department and linked list. After you finish the loop, configure the Grade, studentID, and contact linkedlist.

* 1. Manager UPDATE

Open the Update\_information text file in a read format, move the cursor to the end of the file, and open the LOG text file in a write format. If the information text file has not been opened, print the error code 300 and exit.  
Repeat the loop to the end of the file, read one line after another, and detach all tokens. At this time, if the number of students or contact numbers you have read are duplicated, an error code 300-1 is printed and the student is not added. Then, if the same name is found over and over again, delete the student and update the student again. If the student does not exist in the existing information, the error code 300-2 is output. After you finish the loop, configure the Grade, studentID, and contact linkedlist.

* 1. Manager MODIFY\_DEPARTMENY

Move the cursor to the end of the file and open it in a write format. If the department entered does not exist, the error code 400 is printed.  
If a matched department exists while repeating the loop, disconnect it. Then, move people from the department to the department of interest and organize the linked list. It also deletes the remaining information of the students to reduce memory leakage. After you finish the loop, configure the Grade, studentID, and contact linkedlist.

* 1. Manager DELETE\_DEPARTMENT

Move the cursor to the end of the file and open it in a write format. If the department entered does not exist, an error code 500 is printed.  
If a matched department exists while repeating the loop, delete the student information of the department and finally delete the corresponding department node to reduce memory leakage. After you finish the loop, configure the Grade, studentID, and contact linkedlist.

* 1. Manager PRINT

Move the cursor to the end of the file and open it in a write format. If the linked list does not exist, the error code 600-1 is output.  
If a course is not entered, repeat the loop and print all student information. If a department is entered, print out the information of the student when it matches the one that is repeated and entered. If the department entered and the department does not exist, the error code 600-2 is printed.

* 1. Manager PRINT\_STUDENT\_ID

Move the cursor to the end of the file and open it in a write format. If the linked list does not exist, the error code 700 is printed.  
Store all student information in an array Then measure the length of the array and arrange it in ascending order by the studentID of insertion sort. When insertion sort, one value was extracted from an unsorting place and inserted into an sorting array. Finally, print out student information.

* 1. Manager PRINT\_CONTACT

Move the cursor to the end of the file and open it in a write format. If the linked list does not exist, the error code 800 is printed.  
Store all student information in an array Then measure the length of the array and arrange it in ascending order by using the selection sort by the length of the array. Finally, print out student information.  
When making a selection sort, a value is drawn from an sorting place and inserted if a value smaller than that value is found in an unsorting place.

* 1. Manager FIND\_NAME

Move the cursor to the end of the file and open it in a write format.  
Store all student information in an array Then measure the length of the array and arrange it in ascending order by name using bubble alignment by the length of the array.  
The bubble arrangement was done in the form of swapping while scrubbing from the front.

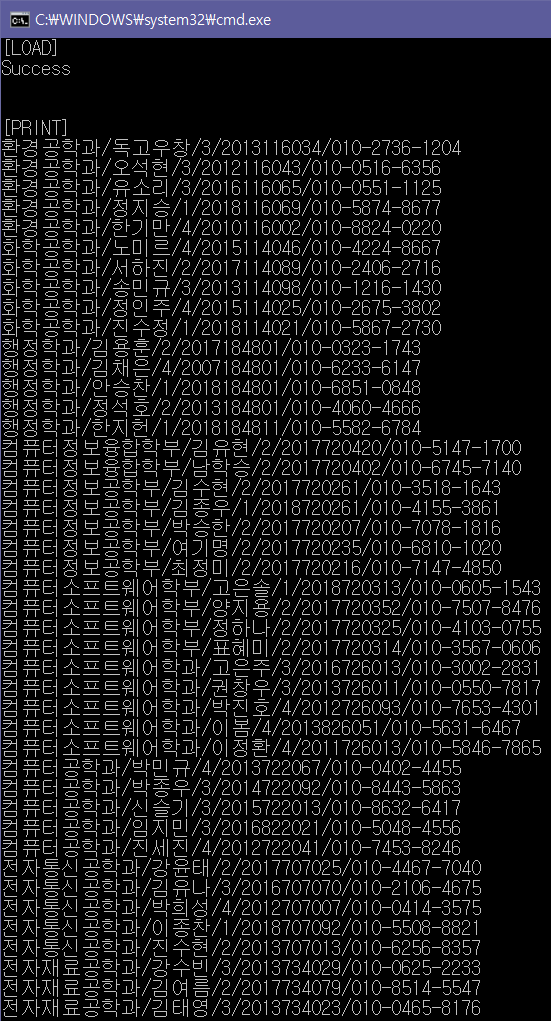
If the entire name is entered, repeat the loop and print information if only the first name is found. If only the last name is entered, repeat the loop, and then print all people with the last name.  
However, if the name or last name does not exist, the error code 900 is printed.

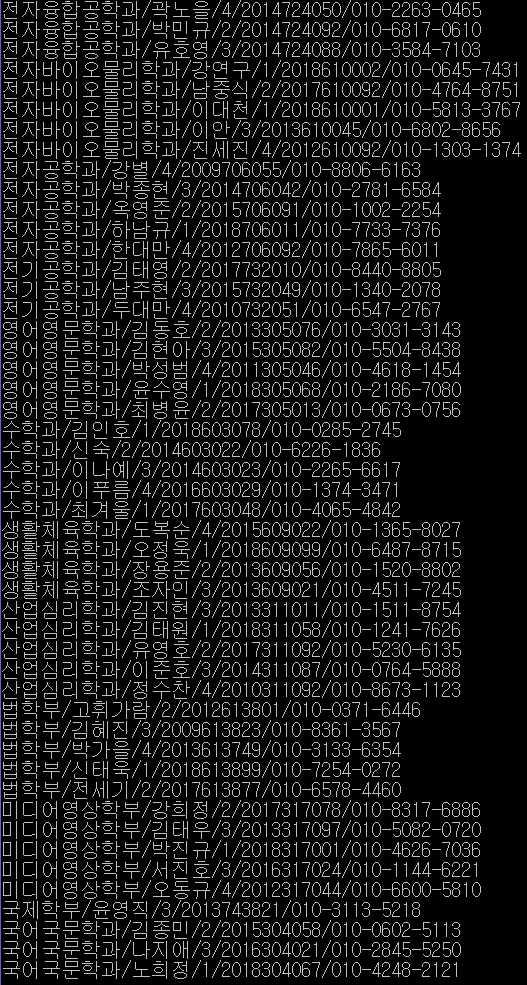
* 1. Manager SAVE

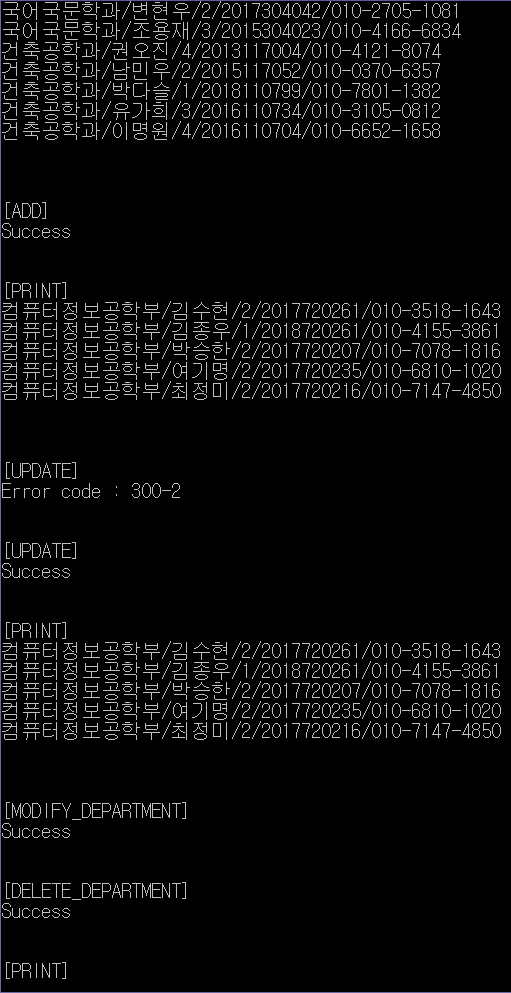
If a file name factor is entered, the file is created with the file name entered and opened. Then enter all of the linked list information into the file. If the file name factor is not entered, the error code 1000 is printed.

1. Result Screen

Console result







An error code 300-2 comes out once because one person was not originally on the linked list.

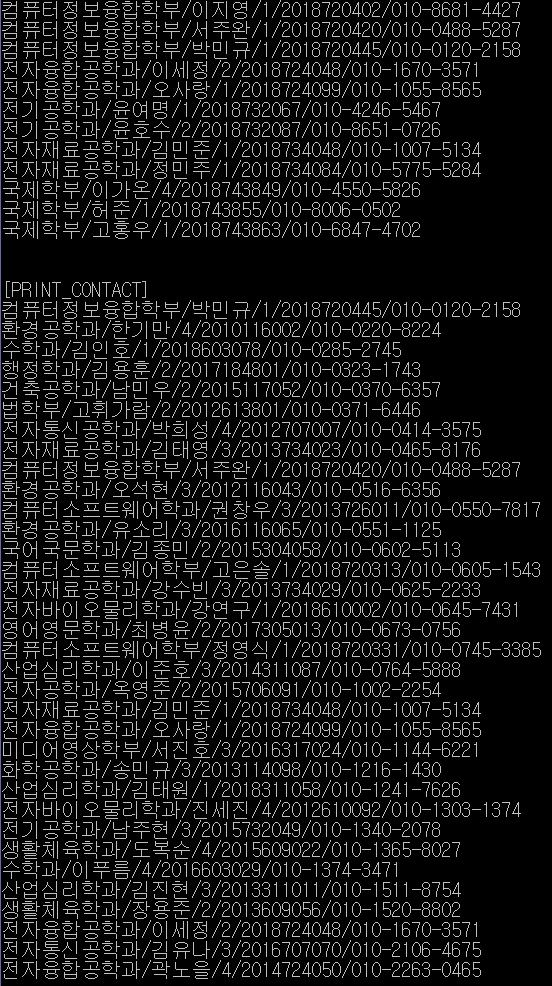


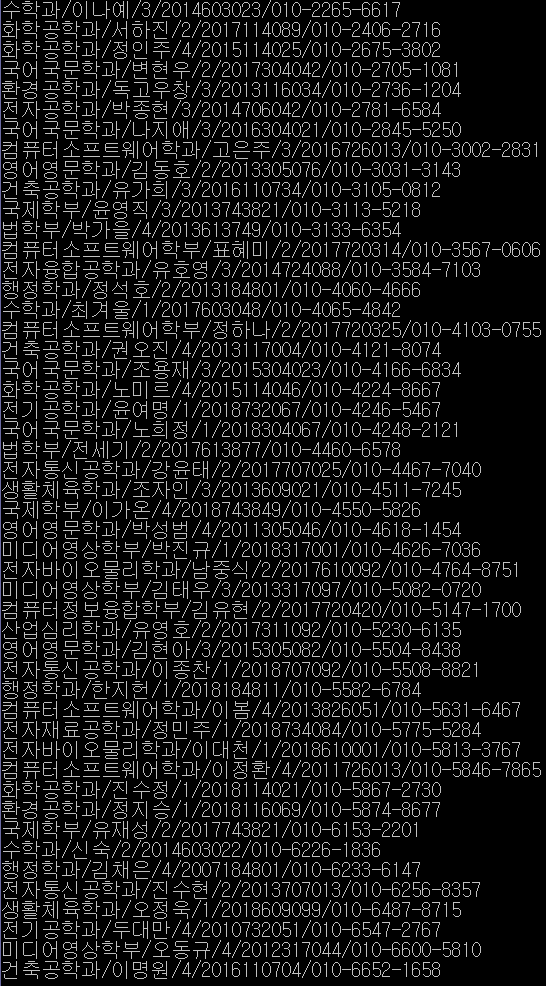


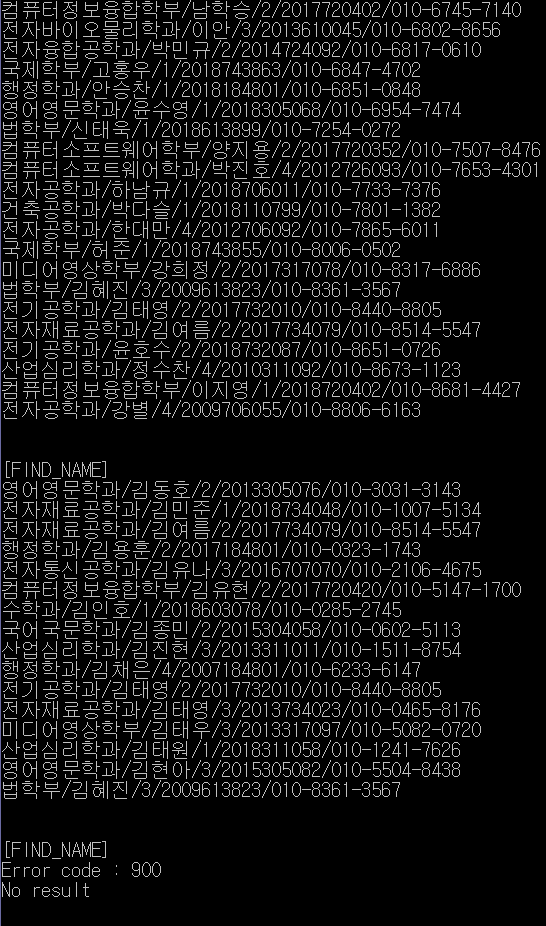




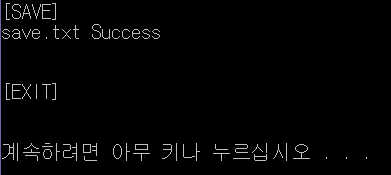






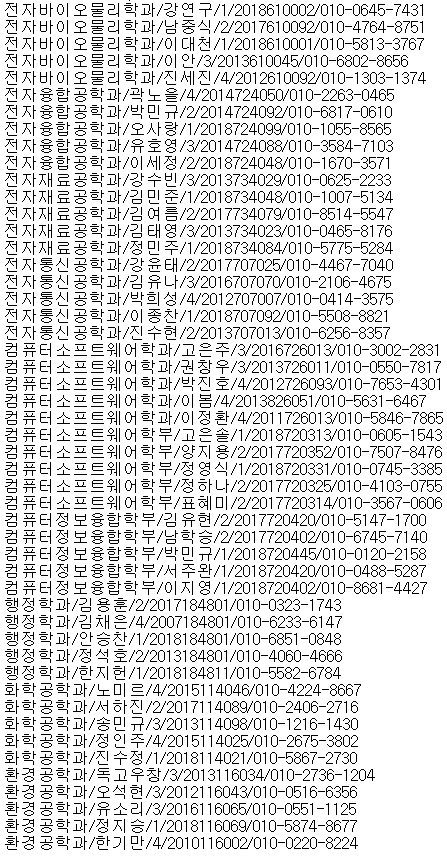


The student was on the node of the computer information engineering department, which was deleted, so we can see the error code 900 is running well.



Save.txt file





LOG file is same with console result

1. Consideration

Until now, it was the most difficult program to code. Once the linked list was connected in many directions and errors were generated in one place, it was very difficult to fix them. And since it was the first time to make a program like this, it was hard to start. But I learned a lot from making this program. I got used to a lot of things such as how to use various functions, file I/O, linked lists, and exception processing.