

[HW10] Conference Enroll Management Program

I. Introduction:

This is a report of my implement of conference enroll management program written in C which utilizes struct pointers to create a linked list which uses a method called self-referencing structure and struct pointers to create a list of data of the same type.

Outline of implementation:

- a. In main.c file: get user's input and use a switch to browse through the options.
- b. In process.h header file: declare a selection function for main.c to call from process.c.
- c. In process.c: declaration of selection, enroll, list, search and delete function.

II. Implementation:

How it works:

First off, *Struct person* is initialized in process.h header file.

```
C process.h > ...
1  #ifndef PROCESS_FILE
2  #define PROCESS_FILE
3
4  typedef struct node{
5      char name[20];
6      char phone[15];
7      struct node *next;
8  } person, *personPtr;
9
10 extern void selection();
11
12 #endif
```

Process.h header file.

Then the user's input is taken to *option*, passed to switch to decide which function to run. Which passes *option* to the switch in process.c.

From here, the user has 5 choices:

1. Enroll: the user input name and phone number separated by a space to enroll. If the name already exists, enroll will not be possible.

2. Listing: the program lists all enrolled participants including phone number.
3. Search: the user input a name to search, if the name is in the enrolled list, the program will print out that name along with its registered phone number. If not, it will simply return.
4. Delete: the user input a name to delete, if there is such name exists, the program will delete that name from the linked list, relink the list if necessary. Otherwise, the program will simply return.
5. Quit: quit the program.

How each function works:

1. *Main*: takes user input option, initializes linked list header, and then passes both of it to Selection function.

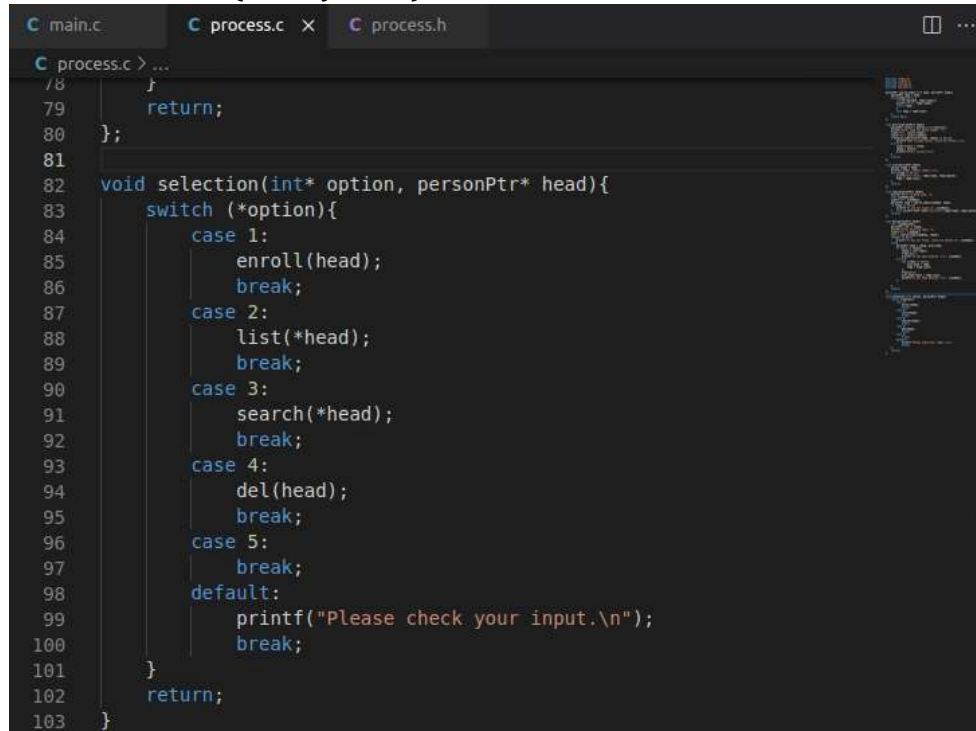
```

C main.c > main(int, char **)
1  #include <stdio.h>
2  #include <stdlib.h>
3  #include <string.h>
4  #include "process.h"
5
6  int main(int argc, char** argv){
7      int option = 0;
8      personPtr head = NULL;
9      while(1){
10         printf("---Enrollment Management System---\n");
11         printf("[1] Enroll\t [2] List\t [3] Search\t [4] Delete\t\n");
12         printf("[5] Quit\n");
13         printf("Selection: ");
14         scanf("%d", &option);
15         switch(option){
16             case 5:
17                 break;
18             default:
19                 selection(&option, &head);
20                 break;
21         }
22         if(option == 5){
23             printf("Quitting...\n");
24             break;
25         }
26         printf("-----\n\n");
27     }
28     return 0;

```

Main.c

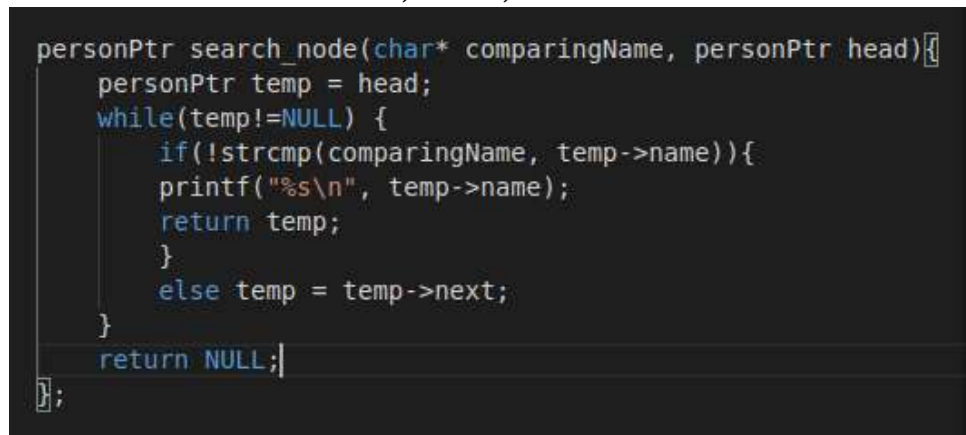
2. *Selection*: switch receives option address and the head address declared in the main function and passes each necessary argument to functions called (mainly head).

A screenshot of a code editor with three tabs: 'main.c', 'process.c', and 'process.h'. The 'process.c' tab is active, showing a C function named 'selection'. The function takes two arguments: 'int* option' and 'personPtr* head'. It uses a switch statement to handle five cases: case 1 calls 'enroll(head)', case 2 calls 'list(*head)', case 3 calls 'search(*head)', case 4 calls 'del(head)', and case 5 calls 'break'. A default case prints 'Please check your input.\n' and also calls 'break'. The function ends with a 'return;' statement. Line numbers 78 to 103 are visible on the left margin.

```
78     }
79     return;
80 };
81
82 void selection(int* option, personPtr* head){
83     switch (*option){
84         case 1:
85             enroll(head);
86             break;
87         case 2:
88             list(*head);
89             break;
90         case 3:
91             search(*head);
92             break;
93         case 4:
94             del(head);
95             break;
96         case 5:
97             break;
98         default:
99             printf("Please check your input.\n");
100             break;
101     }
102     return;
103 }
```

Switch function in process.c

3. *Search_node*: this function is similar to List but instead of printing out nodes' information, search_node takes in a string to compare, cycles through the node list and check for matched results. If there is a match, it will return the matched result's pointer. Return NULL otherwise. This function is used across *enroll*, *search*, and *delete* function.

A screenshot of a code editor showing a C function named 'search_node'. The function takes two arguments: 'char* comparingName' and 'personPtr head'. It declares a 'personPtr temp' and sets it to 'head'. A while loop 'while(temp!=NULL)' iterates through the list. Inside the loop, an 'if(!strcmp(comparingName, temp->name))' checks for a match. If true, it prints the name and returns 'temp'. If false, it moves to the next node with 'temp = temp->next;'. After the loop, it returns 'NULL'. The function is enclosed in a block of curly braces. Line numbers are not visible on the left margin.

```
personPtr search_node(char* comparingName, personPtr head){
    personPtr temp = head;
    while(temp!=NULL) {
        if(!strcmp(comparingName, temp->name)){
            printf("%s\n", temp->name);
            return temp;
        }
        else temp = temp->next;
    }
    return NULL;
};
```

Search_node function in process.c

4. *Enroll*:

```
void enroll(personPtr* head){
    personPtr insert = malloc(sizeof(person));
    printf("Enter name and phone number: ");
    scanf("%s", insert->name);
    scanf("%s", insert->phone);
    if(search_node(insert->name, *head) != NULL){
        printf("Name already exists! Could not enroll.\n");
    } else {
        insert->next = *head;
        *head = insert;
        printf("Enroll success!\n");
    }
    return;
};
```

Enroll function.

When option 1 enroll is called, switch passes linked list header pointer pointed to a pointer to enroll function for easy header changes. New pointer (*insert*) is then created which then take the user's input and store it. Enroll function then proceeds to pass the input name to the function *search_node* to check for duplicate names. If there are not any duplicate names, *insert*'s next pointer will be head, and *insert* becomes head element. In other words, the newly initialized element will become the new head element.

5. *List*: takes the head pointer and assign it to *temp*, which then print out the linked list data while cycling through each node from *temp*.

```
void list(personPtr head){
    person *temp = head;
    printf("\nName \tPhone number:\n");
    while(temp != NULL){
        printf("%s\t %s\n", temp->name, temp->phone);
        temp = temp->next;
    }
    return;
};
```

List function.

6. *Search*: this function starts off by asking for a name to search, which then calls *search_node* function, passing the header to it. If there is no such name, the function will return. Otherwise, the name and phone number are printed out following by a return.

```

void search(personPtr head){
    printf("Enter finding name: ");
    char scanName[20];
    scanf("%s", scanName);
    personPtr temp = search_node(scanName, head);
    if (temp == NULL){
        printf("%s was not found.\n", scanName);
        return;
    } else printf("NAME FOUND:\n%s\t%s\n",temp->name, temp->phone);
    return;
};

```

Search function.

7. *Del*: receives a pointer pointed to a head pointer, takes in the input for the deleting name. Then proceeds to call *search_node* to find the declared name. If that name is not registered, function will notify it and return. Otherwise, if the specified name is also the header, the header will become the next pointer it points to, and then it will be deleted. If it is in the middle of the list, a *temp* node is created and the function loops through the list again to register the previous node, save it to *temp*, free the *curr* (deleting node) and point the *previous node* to the next (*temp*) node.

```

void del(personPtr* head){
    char scanName[20];
    personPtr curr = *head;
    printf("Enter deleting name: ");
    scanf("%s", scanName);
    curr = search_node(scanName, *head);
    if(curr == NULL)
        printf("%s was not found, could not delete.\n", scanName);
    else{
        personPtr temp = *head, prev_node;
        if (curr == *head){
            *head = curr->next;
            free(curr);
            printf("%s has been deleted :(\n", scanName);
        } else{
            while(temp != curr){
                prev_node = temp;
                temp = temp->next;
            };
            free(curr);
            prev_node->next = temp->next;
            printf("%s has been deleted :(\n", scanName);
        }
    }
    return;
};

```

Del (delete) function.

Results:

Enroll:

```
ubuntu@dang-nhat-tuan:/mnt/hgfs/shared/HW10_201924619$ make
gcc -std=c11 -Wall -o obj/process.o -c /mnt/hgfs/shared/HW10_201924619/process.c
gcc -std=c11 -Wall -o main obj/main.o obj/process.o
ubuntu@dang-nhat-tuan:/mnt/hgfs/shared/HW10_201924619$ ./main
---Enrollment Management System---
[1] Enroll      [2] List      [3] Search      [4] Delete      [5] Quit
Selection: 1
Enter name: Seosu
Enter phone: 01099998888
Enroll success!
-----

---Enrollment Management System---
[1] Enroll      [2] List      [3] Search      [4] Delete      [5] Quit
Selection: 1
Enter name: Yeoju
Enter phone: 01088887777
Enroll success!
-----

---Enrollment Management System---
[1] Enroll      [2] List      [3] Search      [4] Delete      [5] Quit
Selection: 3
Enter finding name: Seosu
Seosu
NAME FOUND:
Seosu  01099998888
-----
```

Listing:

```
---Enrollment Management System---
[1] Enroll      [2] List      [3] Search      [4] Delete      [5] Quit
Selection: 2

Name    Phone number:
Yeoju   01088887777
Seosu   01099998888
-----

---Enrollment Management System---
[1] Enroll      [2] List      [3] Search      [4] Delete      [5] Quit
Selection: █
```


Search:

```
---Enrollment Management System---
[1] Enroll      [2] List      [3] Search      [4] Delete      [5] Quit
Selection: 3
Enter finding name: Seosu
NAME FOUND:
Seosu  01099998888
-----

---Enrollment Management System---
[1] Enroll      [2] List      [3] Search      [4] Delete      [5] Quit
Selection: 3
Enter finding name: Yamco
Yamco was not found.
-----
```

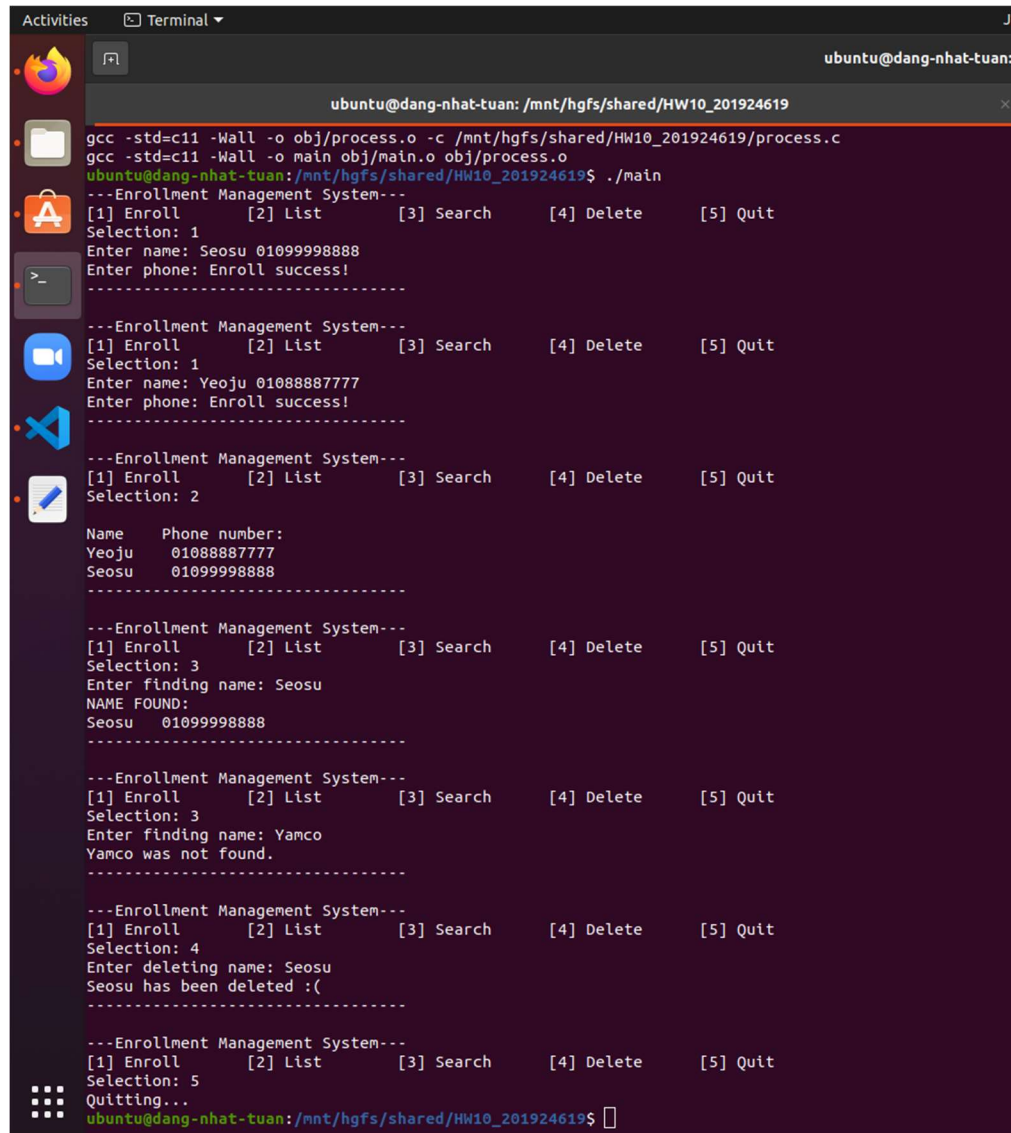
Deletion:

```
---Enrollment Management System---
[1] Enroll      [2] List      [3] Search      [4] Delete      [5] Quit
Selection: 4
Enter deleting name: Seosu
Seosu has been deleted :(
-----
```

Quitting:

```
---Enrollment Management System---
[1] Enroll      [2] List      [3] Search      [4] Delete      [5] Quit
Selection: 5
Quitting...
ubuntu@dang-nhat-tuan:/mnt/hgfs/shared/HW10_201924619$
```

All:



```
ubuntu@dang-nhat-tuan: /mnt/hgfs/shared/HW10_201924619
gcc -std=c11 -Wall -o obj/process.o -c /mnt/hgfs/shared/HW10_201924619/process.c
gcc -std=c11 -Wall -o main obj/main.o obj/process.o
ubuntu@dang-nhat-tuan:/mnt/hgfs/shared/HW10_201924619$ ./main
---Enrollment Management System---
[1] Enroll    [2] List    [3] Search    [4] Delete    [5] Quit
Selection: 1
Enter name: Seosu 01099998888
Enter phone: Enroll success!
-----
---Enrollment Management System---
[1] Enroll    [2] List    [3] Search    [4] Delete    [5] Quit
Selection: 1
Enter name: Yeoju 01088887777
Enter phone: Enroll success!
-----
---Enrollment Management System---
[1] Enroll    [2] List    [3] Search    [4] Delete    [5] Quit
Selection: 2
Name    Phone number:
Yeoju    01088887777
Seosu    01099998888
-----
---Enrollment Management System---
[1] Enroll    [2] List    [3] Search    [4] Delete    [5] Quit
Selection: 3
Enter finding name: Seosu
NAME FOUND:
Seosu    01099998888
-----
---Enrollment Management System---
[1] Enroll    [2] List    [3] Search    [4] Delete    [5] Quit
Selection: 3
Enter finding name: Yamco
Yamco was not found.
-----
---Enrollment Management System---
[1] Enroll    [2] List    [3] Search    [4] Delete    [5] Quit
Selection: 4
Enter deleting name: Seosu
Seosu has been deleted :(
-----
---Enrollment Management System---
[1] Enroll    [2] List    [3] Search    [4] Delete    [5] Quit
Selection: 5
Quitting...
ubuntu@dang-nhat-tuan:/mnt/hgfs/shared/HW10_201924619$
```

III. Conclusion:

Linked list is a superior function comparing to array or classes if correctly used thanks to its flexibility to delete, add, and modify elements anywhere in the list. However, its flexibility is quite difficult to master especially for beginner programmers as it requires knowledge about pointers, structs and memory management in C/C++.

-----End of report-----