자료구조 실습과제

과일 리스트 관리 프로그램

5671144 강범창

1. 과일 리스트 프로그램을 작성.

1.1 단순 연결리스트를 메뉴로 조작하는 프로그램을 작성하기위해 리스트에 사용할 구조체와 element를 선언합니다.

```
typedef char* element;

typedef struct ListNode
{
	element data;
	struct ListNode* link;
} ListNode;
```

1.2 단순 연결리스트를 구현합니다.

insert first 함수입니다. 연결리스트의 head에 값을 넣어 가장 앞에 넣는 것을 구현합니다.

```
ListNode* insertFirst(ListNode* head, int item)
{
	ListNode* node = (ListNode*)malloc(sizeof(ListNode));
	node->data = item;
	node->link = head;
	head = node;
	return head;
}
```

Insert 함수입니다. Head 와 이전의 node추가할 값을 받아와서 사용하며, 이전의 node 에 새로 만든 node 를 넣어 이전에 바꾸자 한 node에 위치에 값을 넣어줍니다.

```
ListNode* insert(ListNode* head, ListNode* pre, element item)
{
	ListNode* node = (ListNode*)malloc(sizeof(ListNode));

	node->data = item;
	node->link = pre->link;
	pre->link = node;
	return head;
}
```

InsertLast 함수입니다. Head 와 item을 받으며 리스트의 마지막에 있는 값을 찾아 거기에 새로운 값으로 만든 node를 가리키게 하여 마지막에 node를 넣는 것을 구현합니다.

```
ListNode* insertLast(ListNode* head, element item)
{
	ListNode* node = (ListNode*)malloc(sizeof(ListNode));
	node->data = item;
	node->link = NULL;

	if (head == NULL)
	{
		head = node;
	}
	else
	{
		ListNode* current = head;
		while (current->link != NULL)
		{
			current = current->link;
		}
		current->link = node;
	}
	return head;
}
```

Delete first 함수입니다. Head를 받아 가장 앞에 있는 node 를 삭제하고 그 다음 연결되어 있는 link를 넣어둬 리스트의 가장 앞에 것을 삭제하는 것을 구현합니다.

```
ListNode* deleteFirst(ListNode* head)
{
	ListNode* removed;
	if (head == NULL)
	{
		return NULL;
	}
	removed = head;
	head = removed->link;
	free(removed);
	return head;
}
```

DeleteListNode 함수입니다. Node를 삭제하는 함수입니다. 입력으로 head, tartget(지울 노드) , 지워진 노드들을 확인하기위해 DeletedList 를 입력으로 받습니다.

```
ListNode* deleteListNode(ListNode* head, ListNode* target, DeletedList* delList)
{
	ListNode* pre = NULL;
	ListNode* current = head;

	while (current != NULL && current != target)
	{
		pre = current;
		current = current->link;
	}

	if (pre == NULL)
	{
		head = current->link;
```

```
        }
        else
        {
                pre->link = current->link;
        }

        delList->data[delList->length] = current->data;
        delList->length++;

        free(current);
        return head;
}
```

SearchList 함수입니다. Head와 element 를 받으며 연결 리스트에서 element 를 찾는데 사용됩니다. 문자열 확인을 위해 strcmp 를 사용하였습니다.

```
ListNode* searchList(ListNode* head, element x)
{
        ListNode* node = head;
        while (node != NULL)
        {
                if (strcmp(node->data, x) == 0)
                        return node;
                node = node->link;
        }
        return NULL;
}
```

printList 함수입니다. Head 를 받아 head에서부터 마지막 노드까지 출력을 합니다.

```
void printList(ListNode* head)
{
	printf("₩nFruit list: ₩n");
	for (ListNode* p = head; p != NULL; p = p->link)
	{
		printf("%s->", p->data);
	}
	printf("NULL₩n");
}
```

1.3 메인에서 입력을 받아 처리하는 부분입니다.

우선 리스트에 최초 과일리스트를 insertLast함수를 통해 집어넣어 줍니다. 이후 입력을 받아 4라면 프로그램을 종료하고, 아니라면 그에 맞는 메뉴를 조건에 맞춰 실행시켜줍니다.

1의 경우 – 과일 이름을 입력 받으며, 이것으로 리스트에서 검색을 하여 있는 경우에는 있다는 메시지와 현재 리스트에 있는 값들을 출력해줍니다. 만약 검색해서 없는 값이다 하면, 리스트의 마지막에 값을 넣고 넣었다는 메시지와 현재 리스트를 출력해줍니다.

2의 경우 – 삭제할 과일의 이름을 받고, 이것으로 검색을 하여 리스트에 이미 있는지 없는지를 확인합니다. 이후 값이 존재한다면 그 값을 삭제를 하고 삭제가 되었다는 메시지와 삭제된 아이템을 보여줍니다. 아니라면 리스트에 값이 없다는 뜻이기에, 없다는 메시지를 출력해주고 현재 리스트를 보여줍니다.

3의 경우 – 2에서 삭제를 성공한 아이템들을 모아둔 리스트를 출력합니다. 0이라면 삭제된 리스트가 없다 출력을 하고 아니라면 삭제된 아이템들을 출력합니다.

4의 경우 – 프로그램을 종료한다는 메시지와 함께 프로그램을 종료합니다.

나머지의 경우 – 따로 처리를 하지 않고 다시 메뉴를 출력합니다.

```
int main(void)
{
	int menu;

	ListNode* head = NULL;
	DeletedList delList;
	delList.length = 0; // Initialize delList
	char input_string[30];
	char* fruitList[10] = {
		"Mango",
		"Orange",
		"Apple",
		"Grape",
		"Cherry",
		"Plum",
		"Guava",
		"Raspberry",
		"Banana",
		"Peach",
	};
	for (int i = 0; i < 10; i++)
	{
		head = insertLast(head, fruitList[i]);
	}

	while (1)
	{
		char* input_string = (char*)malloc(30);
```

```
		printf("==== Menu ====₩n");
		printf("1. Input ₩n");
		printf("2. Delete₩n");
		printf("3. Print ₩n");
		printf("4. Exit₩n");
		printf("Select number: ");
		scanf("%d", &menu);
		printf("₩n");
		switch (menu)
		{
		case 1:
			printf("Fruit name to add: ");
			scanf(" %[^₩n]s", input_string);

			ListNode* insertNode = searchList(head, input_string);
			if (insertNode){
				printf("%s already exists.₩n", insertNode->data);
				printList(head);

			}
			else{
				head = insertLast(head, input_string);
				printf("%s has been added.₩n", input_string);
				printList(head);
			}
			break;
		case 2:
			printf("Fruit name to delete: ");
			scanf("%s", input_string); // 표준 입력을 받아서 배열 형태의
문자열에 저장

			ListNode* deleteNode = searchList(head, input_string);

			if (deleteNode)
			{
```

```
                        head = deleteListNode(head, deleteNode, &delList);
                        printf("%s has been deleted.₩n",
delList.data[delList.length - 1]);
                }
                else
                {
                        printf("%s is not on the list.₩n", input_string);
                }
                printList(head);
                break;
        case 3:
                printf("List of the deleted fruits : ");

                if (delList.length == 0)
                {
                        printf("NULL₩n");
                        printList(head);
                        break;
                }

                for (int i = 0; i < delList.length; i++) {
                        ;
                        printf("%s->", delList.data[i]);
                }
                printf("NULL");
                printf("₩n");
                break;
        case 4:
                printf("Exit the program.₩n");
                exit(0);
                break;
        default:
                break;
        }
        printf("₩n");
```

```
        }
        return 0;
}
```

아래는 실행 결과입니다.

스크린샷으로 담기에는 너무 보기가 힘들어져, 영상으로 대체하였습니다.

https://youtu.be/CQ5P9k6gdS0