

# 자료구조응용

## 10. 연결리스트 (15점)

2020.4.15. (수)

1. 다음과 같이 정렬되지 않는 정수 데이터를 입력하면서 정렬된 Linked List를 만들고 실행 예와 같이 수행되는 프로그램을 작성하라. (10점)

### (1) 실행 순서

① 입력파일("input.txt")로 부터 데이터를 입력받으면서 정렬된 Linked List를 만든다. 입력 데이터는 정렬되지 않은 값으로 중복 가능하다.

input.txt										
50	80	30	20	19	90	30	55	77	30	
99	45	55	89	91	10	20	66	38	59	
22	55	88	22	66	29	50	95	78	83	

- ② Linked List의 처음부터 끝까지 노드의 데이터를 출력한다.
- ③ 성적이 50점 이하인 노드를 Linked List에서 삭제한다.
- ④ Linked List의 처음부터 끝까지 노드의 데이터를 출력한다.
- ⑤ Linked List를 모두 삭제한다.

### (2) 구현 세부사항

① 구조체 정의문은 다음과 같다.

```
typedef struct listNode *listPointer;
typedef struct listNode {
    int data;
    listPointer link;
} listNode;
listPointer first = NULL;
```

### ② 함수

- find : insert 위치를 찾는 함수를 새롭게 정의
- insert : 교재의 Program 4.2는 empty list가 아닌 경우 첫 노드로 추가할 때 에러가 발생한다. 제공되는 수정된 insert 함수를 참고하거나 본인이 생각하는 새로운 함수를 정의
- printList : Program 4.4를 참고하여 수정
- delete : Program 4.3을 그대로 사용

※delete 함수명이 C++의 예약어와 같아서 에러로 표시되나 실행에 문제는 없음. 다른 이름으로 정의해도 됨.

---

```

void insert(listPointer *first, listPointer x, int data)
{ /* 체인의 x 노드 바로 뒤에 data 값을 가지는 새 노드를 추가 */

    /* 새 노드 할당 및 데이터 저장 */
    listPointer temp;
    MALLOC(temp, sizeof(*temp));
    temp->data = data;

    if(*first == NULL)
    { /* (a) 빈 리스트에 노드 추가 */
        _____
        _____
    }
    else
    { /* 비어 있지 않은 리스트에 추가 */
        if ( x == NULL )
        { // (b) 첫 노드로 추가
            _____
            _____
        }
        else
        { // (c) 두 번째 이상 위치에 추가
            _____
            _____
        }
    }
}

```

---

Program 4.2의 수정 코드

---

```

void delete(listPointer *first, listPointer trail,
            listPointer x)
{ /* delete x from the list, trail is the preceding node
   and *first is the front of the list */
    if (trail)
        trail->link = x->link;
    else
        *first = (*first)->link;
    free(x);
}

```

---

**Program 4.3:** Deletion from a list

---

```

void printList(listPointer first)
{
    printf("The list contains: ");
    for (; first; first = first->link)
        printf("%4d", first->data);
    printf("\n");
}

```

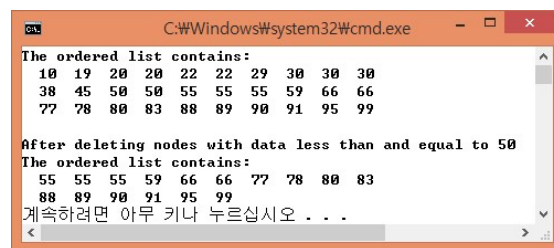
---

**Program 4.4:** Printing a list

### ③ 데이터 입력 및 정렬된 리스트 만들기

```
fscanf(...);
while( !feof(fp) )
{
    find(...)          // find insert position x
    insert(...);       // insert data first after node x.
    fscanf(...);
}
```

### (3) 실행 예



```
C:\Windows\system32\cmd.exe
The ordered list contains:
10 19 20 20 22 22 29 30 30 30
38 45 50 50 55 55 55 59 66 66
77 78 80 83 88 89 90 91 95 99

After deleting nodes with data less than and equal to 50
The ordered list contains:
55 55 55 59 66 66 77 78 80 83
88 89 90 91 95 99
계속하려면 아무 키나 누르십시오 . . .
```

2. [Linked Stacks] 다음과 같은 스택을 생성하고 실행하는 프로그램을 작성하라. 이를 위해, push, pop, stackEmpty 함수를 구현하여야 한다. (5점)

#### (1) 실행 순서

① 입력파일("input.txt")로 부터 학번 순으로 미리 정렬된 데이터를 입력받으면서 순서대로 Linked Stack을 구현한다. (과목번호, 학번, 성적)의 쌍으로 데이터들이 입력되며 각 과목별로 스택에 저장된다.

0	1	95
1	1	80
2	1	89
0	2	45
1	2	81
0	3	45
1	3	12
2	3	33
0	4	99
1	4	94
2	4	91
0	5	67
2	5	49

② 각 과목 별로 학번의 역순으로 노드의 데이터(학번, 성적)를 출력하라.

## (2) 구현 세부사항

```
#define MAX_STACKS 3
typedef struct {
    int id;          //학번
    int grade;       //성적
} element;
typedef struct stack *stackPointer;
typedef struct stack {
    element data;
    stackPointer link;
}Node;
stackPointer top[MAX_STACKS];
/*****
top[i] = NULL, 0 ≤ i < MAX_STACKS // initial condition
top[i] = NULL, iff the ith stack is empty
*****/
```

---

```
void push(int i, element item)
{ /* add item to the ith stack */
    stackPointer temp;
    MALLOC(temp, sizeof(*temp));
    temp→data = item;
    temp→link = top[i];
    top[i] = temp;
}
```

---

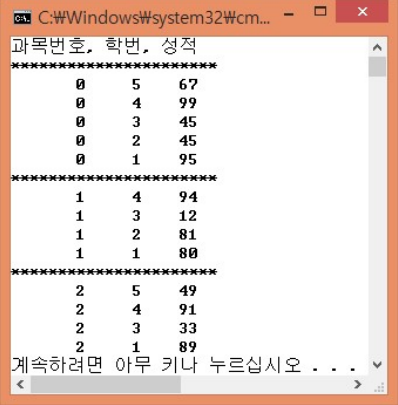
### **Program 4.5:** Add to a linked stack

```
element pop(int i)
{ /* remove top element from the ith stack */
    stackPointer temp = top[i];
    element item;
    if (!temp)
        return stackEmpty();
    item = temp→data;
    top[i] = temp→link;
    free(temp);
    return item;
}
```

---

### **Program 4.6:** Delete from a linked stack

### (3) 실행 예



A screenshot of a Windows command prompt window. The title bar shows the path 'C:\Windows\system32\cmd...'. The window contains a table of scores with three columns: '과목번호' (Subject Number), '학번' (Student ID), and '성적' (Score). The table is divided into three sections by lines of asterisks. The first section has 5 rows for subject 0, the second for subject 1, and the third for subject 2. At the bottom, there is a prompt '계속하려면 아무 키나 누르십시오 . . . '.

과목번호	학번	성적
0	5	67
0	4	99
0	3	45
0	2	45
0	1	95
1	4	94
1	3	12
1	2	81
1	1	80
2	5	49
2	4	91
2	3	33
2	1	89

#### ■ 제출 형식

- 솔루션 이름 : DS 10
- 프로젝트 이름 : 1, 2
- 각 소스파일에 주석처리  
“학번 이름”  
“본인은 이 소스파일을 복사 없이 직접 작성하였습니다.”
- 솔루션 정리 후 솔루션 폴더를 “학번.zip”으로 압축하여 과제 게시판에 제출

#### ■ 주의

- 소스 복사로는 실력향상을 기대할 수 없습니다!!!
- 마감 : 4월 19일(일) 자정