# [알고리즘 스터디] 자료구조 복습



imhyun · 방금 전 · 비공개



알고리즘 자료구조

# 풍선 터뜨리기 - 원형 연결 리스트

https://www.acmicpc.net/problem/2346

> 문제 핵심 : *N번 풍선의 오른쪽에는 1번 풍선이* 있다고 하였다. 연결 리스트 중 마지막 노드가 첫 번째 노드를 가르켜, 연결의 형태가 원을 이루는 '**원형 연결 리스트**'의 형태로 풀 수 있다. 풍선을 터뜨린 후, 풍선 안 종이를 꺼내어 그 종이에 적혀있는 값만큼 이동하여 터뜨린다고 하였다. 이 때 풍선을 터뜨린 다는 것은 ***노드의 삭제***를 의미하고, 종이에 적혀있는 값만큼 이동한다는 것은 노드의 탐색을 의미한다. 양수라면 오른쪽으로 이동하니 next 주소를 기억해야 하고, 음수라면 왼쪽으로 이동하니 prev 주소를 기억해야 한다. 따라서 최종적으로는 '**원형 이중 연결 리스트**'를 구현해야 한다. (음수 부분을 생각 안 하고 원형 단일 연결 리스트로 푸는 바람에 다시 코드를 갈아 엎었다,,,ㅎㅎ)

## 원형 연결 리스트 개념

원형 연결 리스트에는 **head를 설정**하는 방법, **tail을 설정**하는 방법이 있다. 우리는 문제 해석을 통해 InsertLast(입력한 후 노드를 추가하기), DeleteNode (번호만큼 이동해 삭제하기) 두 가지 ADT를 만들어야 한다.

이 때 head를 설정한다면 size만큼 반복하여 끝 노드를 찾아야 하기 때문에, **차라리 주소를 알아 바로 Insert할 수 있는 tail을 설정**하기로 하였다.

이 때 주의해야 할 것은 Insert에서 init하고 첫번째, 두번째 삽입을 진행할 때의 경우이다. 특히 InsertLast이므로, ***L->T의 연결, 삽입 node의 연결을 잘 생각해주어야 한다***.

# 실제 코드

```
#include <stdio.h>
#include <stdlib.h>
#pragma warning (disable:4996)

typedef struct ListNode {
    int elem;
    int num;
    struct ListNode* next;
    struct ListNode* prev;
}ListNode;

typedef struct ListType {
    ListNode* T;
}ListType;

void init(ListType *);
void InsertLast(ListType* ,int,int);
void DeleteNode(ListType* );
void print(ListType *);

int main() {
    ListType L;
    int N;
    int i;
    int e;

    init(&L);

    scanf("%d", &N);

    //풍선 더미들 만듦.
    for (i = 0; i < N; i++) {
        scanf("%d", &e);
        InsertLast(&L, e,i);
    }

    DeleteNode(&L);

    return 0;
}

void init(ListType * L) {
    L->T = NULL;
}

void InsertLast(ListType* L, int e,int i) {
    //노드 한 개 생성
    ListNode* node = (ListNode*)malloc(sizeof(ListNode));
    node->elem = e;
```

```
        node->num = i + 1;
        node->next = NULL;
        node->prev = NULL;

        //Insert
        if (L->T == NULL) {
            L->T = node;
            L->T->prev = node;
            L->T->next=node;
        }

        else {

            node->next = L->T->next;
            node->prev = L->T;

            L->T->next = node;
            if (L->T->prev == L->T)
                L->T->prev = node;

            L->T = node;
        }

    }

    void DeleteNode(ListType* L) {
        ListNode* p = L->T;
        ListNode* deleted = L->T->next; //첫번째 풍선
        int i;

        while (L->T!=NULL)
        {
            printf("%d ", deleted->num);

            //삭제를 할 거야.
            p->next = deleted->next;
            deleted->next->prev = p;

            if (deleted->elem > 0) {
                //이동을 할 거야. (양수일 때)
                for (i = 0; i < deleted->elem - 1; i++)
                    p = p->next;
            }

            if (deleted->elem < 0) {
                //이동을 할 거야. (음수일 때)
                for (i = deleted->elem; i<0; i++)
                    p = p->prev;
            }

            free(deleted);

            deleted = p->next;

            if (deleted == L->T)
                L->T = p;
        }
    }
```

```
void print(ListType* L) {
    ListNode* p = L->T->next;
    for (; p != L->T; p = p->next) {
        printf(" %d", p->elem);
    }
    printf(" %d", p->elem);

}
```

하지만,,, **~~메모리 초과~~** 문제가....
메모리 초과 문제는 '스택 오버플로우'가 발생했기 때문이라고 한다. 크게
**1. 너무 많은 변수를 배열에 저장하거나**
**2. 재귀적 호출을 통해 너무 많은 함수를 호출할 때**
두 가지 케이스가 가장 크다고 한다....

# 스택

https://www.acmicpc.net/problem/10828

> 문제 핵심 : 단순한 스택 **개념 문제**이다. 스택은 선형 자료구조로, **한쪽은 막히고 한쪽만 뚫려 있는** 프링글스 통에 비유할 수 있다. (~~참고로 저는 프링글스 한 통 다 먹을 수 있어요,,,,~~) 따라서 "먼저 들어간 것이 나중에 나온다!" 즉, **후입 선출**의 특성을 지닌다.

# 스택 특징

- 삽입과 삭제는 **후입 선출**이다.
- 삽입과 삭제는 **top에서만** 진행된다.

# 실제 코드

```
#include <stdio.h>
#include <stdlib.h>
#pragma warning (disable:4996)

typedef struct ListNode {
    int elem;
    struct ListNode* prev;
}ListNode;
```

```
typedef struct ListType {
    ListNode* T;
}ListType;

void init(ListType *);
void push(ListType*, int);
void pop(ListType*);
void size(ListType*);
void empty(ListType*);
void top(ListType*);


int main() {
    ListType L;
    int N, i;
    char type[6] = "";
    int elem;

    init(&L);

    scanf("%d", &N);

    //풍선 더미들 만듦.
    for (i = 0; i < N; i++) {
        scanf("%s", &type);

        if (strcmp(type, "push") == 0) {
            scanf("%d", &elem);
            push(&L, elem);
        }

        else if (strcmp(type, "pop") == 0)
            pop(&L);

        else if (strcmp(type, "size") == 0)
            size(&L);

        else if (strcmp(type, "empty") == 0)
            empty(&L);

        else if (strcmp(type, "top") == 0)
            top(&L);

    }

    return 0;
}

void init(ListType * L) {
    L->T = NULL;
}
void push(ListType* L, int e) {
    //노드 생성
    ListNode* node = (ListNode*)malloc(sizeof(ListNode));
    node->elem = e;
    node->prev = NULL;

    //Top에서의 연결
```

```
        if(L->T !=NULL)
            node->prev= L->T;

        L->T = node;

    }
    void pop(ListType* L) {
        if (L->T == NULL) //비어있지 않으면
            printf("-1\n");
        else {        //비어 있으면
            ListNode* p = L->T;

            printf("%d\n", L->T->elem);
            L->T = p->prev;
            free(p);
        }


    }

    void size(ListType* L) {
        ListNode* p = L->T;
        int num=0;

        for (; p != NULL; p = p->prev)
            num++;

        printf("%d\n", num);
    }

    void empty(ListType* L) {
        if (L->T != NULL) //비어있지 않으면
            printf("0\n");
        else //비어 있으면
            printf("1\n");
    }

    void top(ListType* L) {
        if (L->T != NULL)
            printf("%d\n", L->T->elem);
        else
            printf("-1\n");

    }
```

여기서 또한 **어떤 데이터 구조를 사용할 것인지 정하고 문제에서 만들고자하는 기능을 생각**했어야 했는데 그러지 못했다. Top에서만 pop,push가 진행되어야 하는데 단순히 next 주소를 구조체로 만들고 삽입하는 node의 next는 없는 형태로 Top으로 만들어주니 push할 때 이전 노드에 접근하지 못 했다.

> 오늘의 최종 결론 : 어떤 데이터 구조 쓸지 정했으면, **구체적으로 어떻게 접근하여 기능을 구현**할지 생각하고 코드 작성하기!!!

**박시현**

이전 포스트
[시계열 분석]Prophet 모델로 제품 판매량 예측하기

## 0개의 댓글

댓글을 작성하세요

댓글 작성