스택_괄호 검사 문제_배열

stackAry.h

```
#ifndef STACK_H
#define STACK_H

#include <string.h>
#include <stdio.h>

#define MAX_CAPACITY 100

char stack[MAX_CAPACITY];
int top = -1;

int is_full();
int is_empty();

void push(char ch)
{
	if (is_full())
		return;
	top++;
	stack[top] = ch;
}

char pop()
{
	if (is_empty())
		return 'A';
	char tmp = stack[top];
	top--;
	return tmp;
}

char peek()
{
	return stack[top];
}

int is_empty()
{
	return top == -1;
}

int is_full()
{
	return top == MAX_CAPACITY - 1;
}

#endif
```

stackAry.cpp

```
#include "stackAry.h"

#define MAX_LENGTH 100

char OPEN[] = "([{";
char CLOSE[] = ")]}";

int is_balanced(char* expr);
int is_open(char ch);
int is_close(char ch);

int main()
{
	char expr[MAX_LENGTH];
	scanf_s("%s", expr, MAX_LENGTH);
	if (is_balanced(expr))
		printf("%s: balanced.₩n", expr);
	else
		printf("%s: unbalanced.₩n", expr);
}

int is_balanced(char* expr)
{
	int balanced = 1;
	int index = 0;
	while (balanced && index < strlen(expr)) {
		char ch = expr[index];
		if (is_open(ch) > -1)
			push(ch);
		else if (is_close(ch) > -1) {
			if (is_empty()) {
				balanced = 0;
				break;
			}
			char top_ch = pop();
			if (is_open(top_ch) != is_close(ch)) {
				balanced = 0;
			}
		}
		index++;
	}
	return (balanced == 1 && is_empty() == 1);
}

int is_open(char ch)
{
	for (int i = 0; i < strlen(OPEN); i++)
		if (OPEN[i] == ch)
			return i;
	return -1;
}

int is_close(char ch)
```

```
{
        for (int i = 0; i < strlen(CLOSE); i++)
                if (CLOSE[i] == ch)
                        return i;
        return -1;
}
```

스택_괄호 검사 문제_연결리스트

stackList.h

```
#ifndef STACK_H
#define STACK_H

#include <string.h>
#include <stdio.h>
#include <stdlib.h>

struct node {
        char* data;
        struct node* next;
};
typedef struct node Node;

Node* top = NULL;

void push(char* item) {
        Node* p = (Node*)malloc(sizeof(Node));
        p->data = item;
        p->next = top;
        top = p;
}

char* pop() {
        if (is_empty())
                return NULL;
        char* result = top->data;
        top = top->next;
        return result;
}

char* peek() {
        if (is_empty())
                return NULL;
        return top->data;
}

int is_empty() {
        return top == NULL;
}

#endif
```

스택ADT_괄호 검사 문제_배열

stackADTAry.h

```
#ifndef STACKADTARY_H
#define STACKADTARY_H

#include <stdbool.h>

typedef int Item; //정수를 사용하는 stack이라서 별명 item으로 관리
// 나중에 데이터 타입이 바뀌면 여기만 수정하면 된다. 코드의 재사용에 유리
// ex) typedef float Item

typedef struct stack_type* Stack;
// stack_type* 타입을 Stack으로 별명 설정

Stack create();
void destory(Stack s);
void make_empty(Stack s);
bool is_empty(Stack s);
void push(Stack s, Item i);
bool is_full(Stack s);
Item pop(Stack s);
Item peek(Stack s);
void reallocate(Stack s);

#endif
```

stackADTAry.cpp

```
#include <stdio.h>
#include <stdlib.h>
#include "stackADTAry.h"

#define INIT_CAPACITY 100

struct stack_type {
        Item* contents; //stack 역할을 하는 배열
        int top;
        int size; // 배열의 크기
};

static void terminate(const char* message)
{
        printf("%s₩n", message);
        exit(1);
}

Stack create()
{
        Stack s = (Stack)malloc(sizeof(struct stack_type));
        if (s == NULL) {
                terminate("Error in creawte: stack could not be created.");
```

```
        }
        s->contents = (Item*)malloc(INIT_CAPACITY * sizeof(Item));
        if (s->contents == NULL) {
                free(s);
                terminate("Error in create: stack could not be created.");
        }
        s->top = -1;
        s->size = INIT_CAPACITY;
        return s;
}

void destory(Stack s)
{
        free(s->contents);
        free(s);
}

void make_empty(Stack s)
{
        s->top = -1; //실제로 배열 내용을 없앤게 아니지만 top1 -1로 사용상 비웠다고 볼 수
있다.
}

bool is_empty(Stack s)
{
        return s->top == -1; // top이 -1이면 비어있다는 의미
}

void push(Stack s, Item i)
{
        if (is_full(s)) {
                reallocate(s);
        }
        s->top++;
        s->contents[s->top] = i;
}

Item pop(Stack s)
{
        if (is_empty(s))
                terminate("Error in pop: stack is empty");
        s->top--;
        return s->contents[s->top + 1];
}

Item peek(Stack s)
{
        if (is_empty(s)) {
                terminate("Error in peek: stack is empty.");
        }
        return s->contents[s->top];
}

void reallocate(Stack s)
{
```

```
        Item* tmp = (Item*)malloc(2 * s->size * sizeof(Item));
        if (tmp == NULL) {
                terminate("Error in create: stack could not be created.");
        }
        for (int i = 0; i < s->size; i++) {
                tmp[i] = s->contents[i];
        }
        free(s->contents);
        s->contents = tmp;
        s->size *= 2;
}

bool is_full(Stack s)
{
        return s->top == s->size - 1;
}
```

main.cpp

```
#include "stackADTAry.h"
#include <stdio.h>

int main()
{
        Stack s1 = create();
        Stack s2 = create();

        push(s1, 10);
        printf("%d", pop(s1));

        return 0;
}
```

스택ADT_괄호 검사 문제_연결리스트

stackADTList.h

```
#ifndef STACKADTList_H
#define STACKADTList_H

#include <stdbool.h>

typedef int Item; //정수를 사용하는 stack이라서 별명 item으로 관리
// 나중에 데이터 타입이 바뀌면 여기만 수정하면 된다. 코드의 재사용에 유리
// ex) typedef float Item

typedef struct stack_type* Stack;
// stack_type* 타입을 Stack으로 별명 설정

Stack create();
void destory(Stack s);
void make_empty(Stack s);
```

```
bool is_empty(Stack s);
void push(Stack s, Item i);
Item pop(Stack s);
Item peek(Stack s);

#endif
```

stackADTList.cpp

```
#include <stdio.h>
#include <stdlib.h>
#include "stackADTList.h"

#define INIT_CAPACITY 100

struct node {
        Item data;
        struct node* next;
};

struct stack_type {
        struct node* top;
};

static void terminate(const char* message)
{
        printf("%s₩n", message);
        exit(EXIT_FAILURE);
}

Stack create()
{
        Stack s = (Stack)malloc(sizeof(struct stack_type));
        if (s == NULL) {
                terminate("Error in creawte: stack could not be created.");
        }
        s->top = NULL;
        return s;
}

void destory(Stack s)
{
        make_empty(s);
        free(s);
}

void make_empty(Stack s)
{
        while (!is_empty(s))
                pop(s);
}

bool is_empty(Stack s)
```

```
{
        return s->top == NULL; // top이 -1이면 비어있다는 의미
}

void push(Stack s, Item i)
{
        struct node* new_node = (struct node *)malloc(sizeof(struct node));
        if (new_node == NULL) {
                terminate("Error in push: stack is full.");
        }
        new_node->data = i;
        new_node->next = s->top;
        s->top = new_node;
}

Item pop(Stack s)
{
        struct node* old_top;
        Item i;
        if (is_empty(s))
                terminate("Error in pop: stack is empty");
        old_top = s->top;
        i = old_top->data;
        s->top = old_top->next;
        free(old_top);
        return i;
}

Item peek(Stack s)
{
        if (is_empty(s)) {
                terminate("Error in peek: stack is empty.");
        }
        return s->top->data;
}
```

main.cpp

```
#include "stackADTList.h"
#include <stdio.h>

int main()
{
        Stack s1 = create();
        Stack s2 = create();

        push(s1, 10);
        printf("%d", pop(s1));

        return 0;
}
```