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;

}