5645640 제갈건 0511

1. 실행코드

#define \_CRT\_SECURE\_NO\_WARNINGS

#include <stdio.h>

#include <stdlib.h>

#include <string.h>

//top, size, full, empty, swap, printf

/\* 산술연산

\* prefix, infix, postfix

\* postfix - stack

\* 퀴즈 있음

\*/

/\*

\* 1. 현재 스택의 element의 개수를 구하는 문제

\* 2.

\*/

#define max\_size 100

typedef struct ListStack {

char data;

struct ListStack\* under;

} Stack;

bool IsEmptyStack(Stack\* Top) {

if (Top == NULL) {

//printf("Stack is Empty\n");

return true;

}

return false;

}

bool isFullStack(Stack\* Top) {

int size = 0;

if (Top == NULL) return false;

while (Top != NULL && size <= max\_size) {

size++;

Top = Top->under;

}

if (size >= max\_size) {

printf("Stack is Full\n");

return true;

}

return false;

}

void printStack(Stack\* Top) {

if (IsEmptyStack(Top)) return;

//printf("Stack :TOP");

do {

printf(" %c", Top->data);

Top = Top->under;

} while (Top != NULL);

printf("\n");

if (isFullStack(Top)) return;

}

void push(Stack\*\* Top, char data) {

Stack\* newNode = NULL;

newNode = (Stack\*)malloc(sizeof(Stack));

if (isFullStack(\*Top)) return;

//리스트 맨 앞에 새로운 노드를 추가합니다.

newNode->data = data;

newNode->under = \*Top;

\*Top = newNode;

//printf("%d pushed to stack\n", data);

}

int pop(Stack\*\* Top) {

Stack\* temp = NULL;

int data = 0;

if (IsEmptyStack(\*Top)) return INT\_MIN;

else

{ //Top 위치(리스트 맨 앞)에 있는 노드를 제거합니다

temp = \*Top;

data = (\*Top)->data;

\*Top = (\*Top)->under;

free(temp); // 노드 제거

}

return data;

}

int peek(Stack\* Top) {//peek함수

if (IsEmptyStack(Top)) return INT\_MIN;

printf("\npeek\_top: %d\n", Top->data);

return Top->data;

}

int size(Stack\* Top) { // 현재 스택의 개수 리턴

if (isFullStack(Top)) return max\_size;

int size = 0;

while (Top != NULL) {

size++;

Top = Top->under;

}

return size;

}

bool check\_matching(Stack\*\* Top, const char\* expr) {

/\*

\* operator : 연산자

\* operand : 피연산자

\*/

int i = 0;

char ch = 0;

char open\_ch;

while (expr[i] != NULL) { //입력 expr의 끝으로 갈 때까지 순회한다.

ch = expr[i]; //ch ← expr의 다음 글자

switch (ch) {

case '(': case '[': case'{':

push(Top, ch); //ch를 스택에 삽입

break;

case ')': case ']': case'}':

// 스택이 비어 있으면(no open bracket) false

if (IsEmptyStack((\*Top))) {

printStack(\*Top);

printf("no open bracket");

return false;

}

else { //스택에서 open\_ch를 꺼낸다

open\_ch = pop(Top);

// 만약 pop한 문자가 pair가 맞지 않다면(wrong pair) false

if ((open\_ch == '(' && ch != ')') || (open\_ch == '[' && ch != ']') || (open\_ch == '{' && ch != '}')) {

printStack(\*Top);

printf("wrong pair");

return false;

}

break;

}

}

i++;

}

printStack(\*Top);

if (IsEmptyStack((\*Top))) {

printf("정상 출력");

return true; // 비어있다면 ture

}

//스택이 비어 있지 않으면(no close bracket) false

else {

printf("no close bracket");

return false;

}

}

int main(void)

{

Stack\* S = NULL;

const char\* exp = "[({2+3}\*4)+9]";

//const char\* exp = "[({2+3\*4)}+9]"; //worng pair

//const char\* exp = "{(2+3\*4)}+9]"; // no open bracket

//const char\* exp = "[({2+3}\*4)+9"; // no close bracket

printf("\nExpression => %s\n", exp);

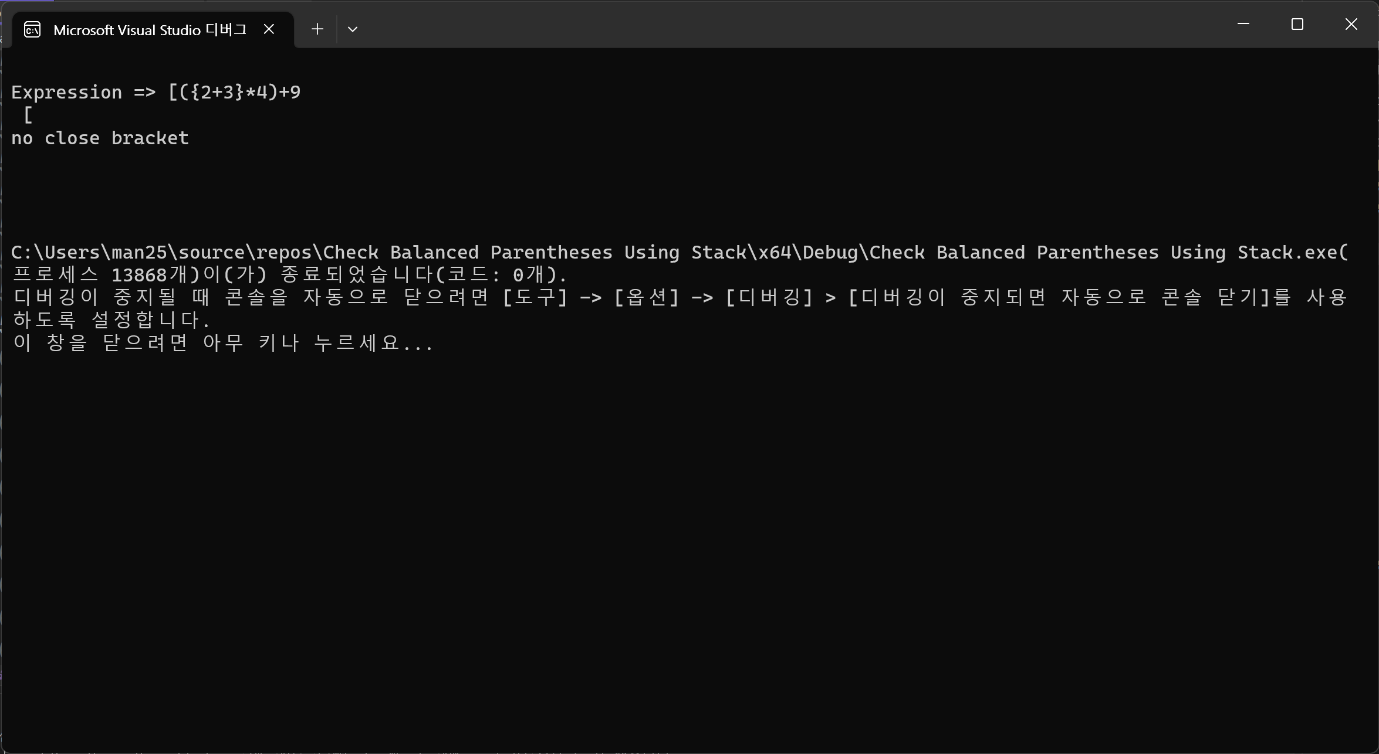
check\_matching(&S, exp);

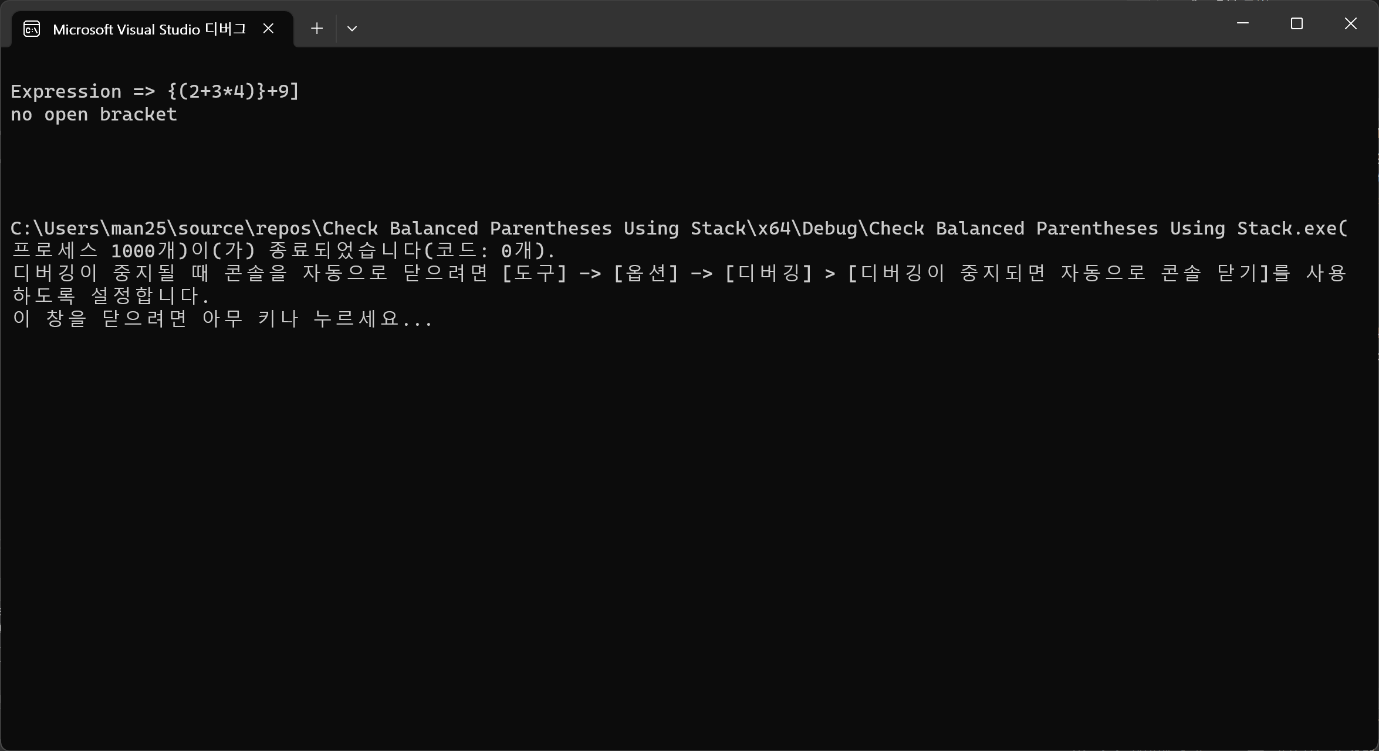
free(S);

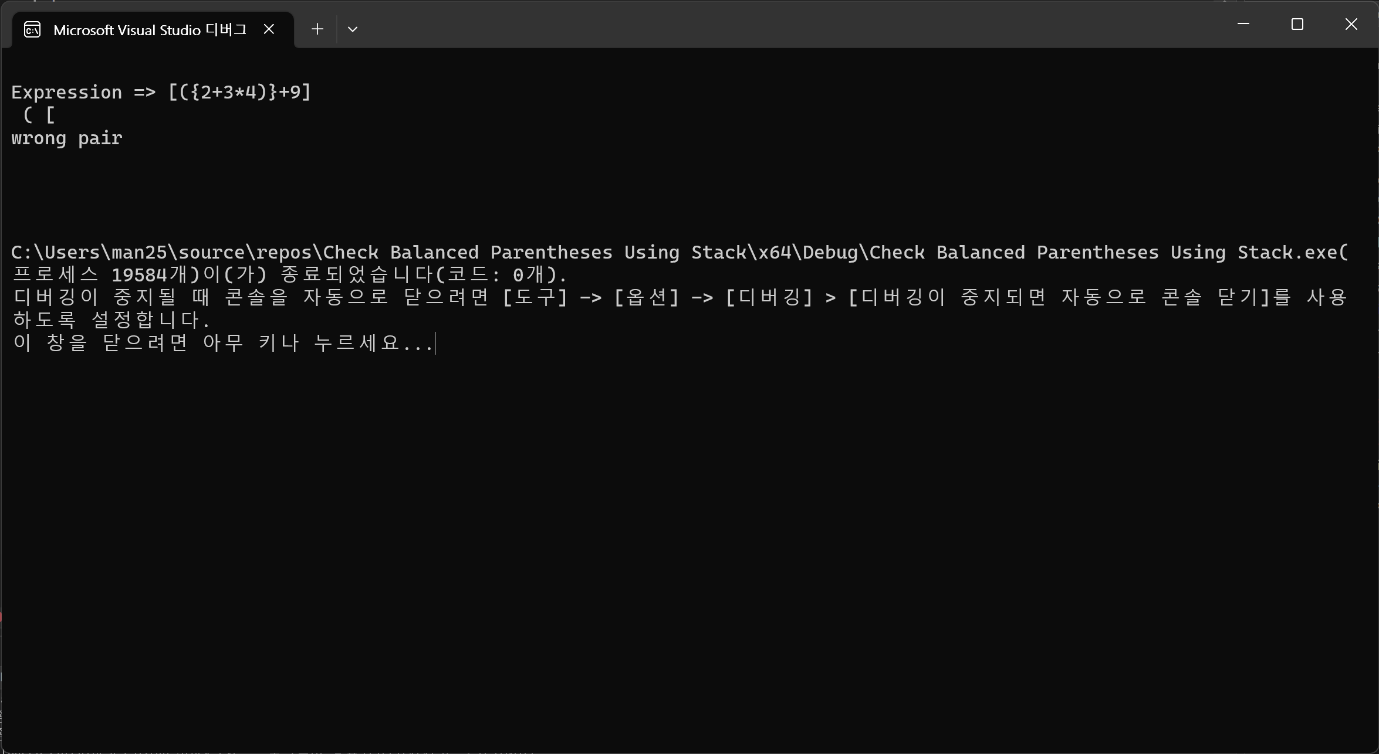
return 0;

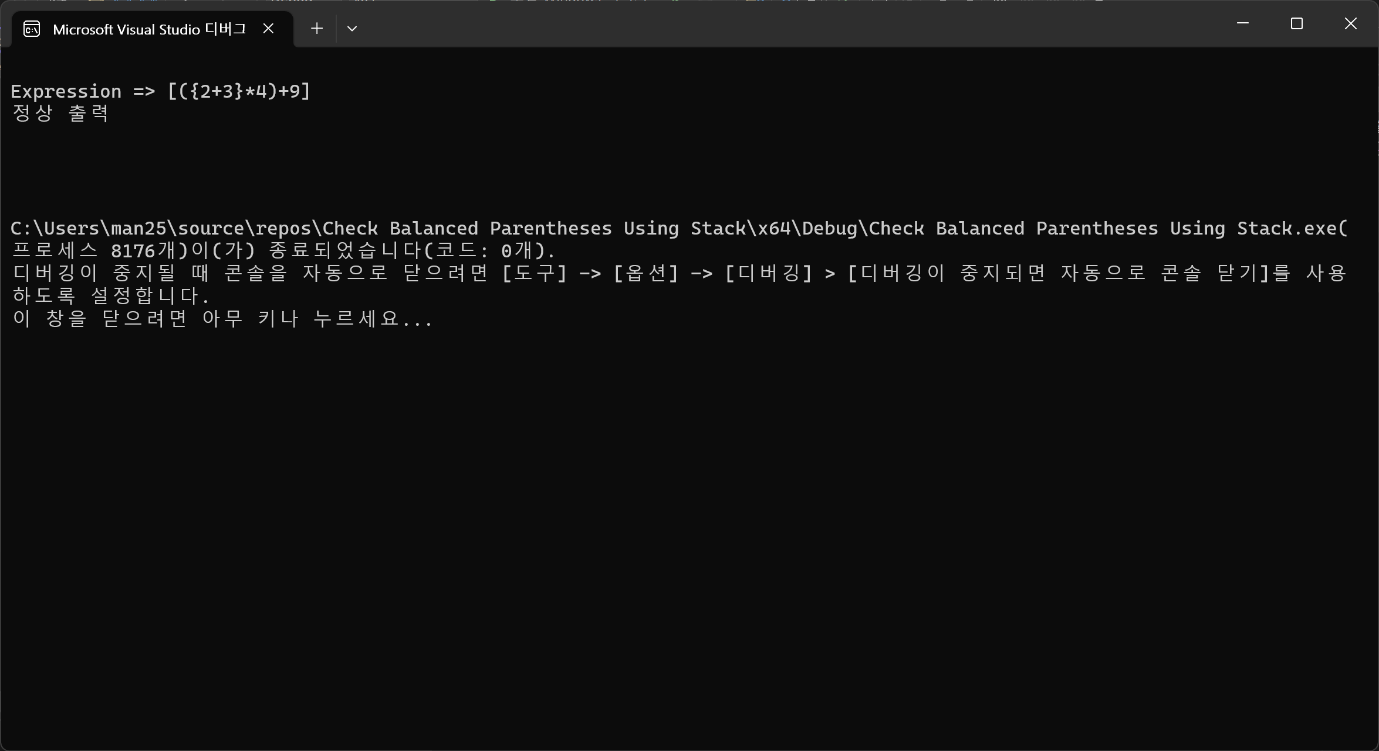
}

1. 실행화면









1. 고찰

이번 과제를 하면서 확실히 문자열에 약하다는 것을 깨달았습니다. Char을 사용하면서 가장 많이 오류(버퍼나 스트림 관련 오류)를 발견했고 해결하는 데 어려움을 느꼈습니다. 개인적으로 문자열에 관한 것(버퍼나 스트림)을 추가적으로 공부해야 할 것 같습니다.