**자료구조 실습보고서**

**실습 4-2. 괄호 검사**

**2016년 3월 23일**

**학번: 201404051**

**이름: 정 용 석**

**1. 실습 문제 소개**

실습 4-1 과정에서 작성한 스택 구조체와 이를 응용한 함수들을 사용하여 주어진 수식 문자열에 대하여 세 가지의 괄호 (), {}. []의 짝이 맞는지 확인하는 프로그램을 괄호 검사 알고리즘에 맞게 작성한다.

**2. 소스 코드**

#include "myStack.h"

#include "stdio.h"

#include "string.h"

#pragma warning (disable : 4996)

#define MAX\_LEN 101

typedef char Expression[MAX\_LEN]; //char[MAX\_LEN] 형식을 가진 Expression 자료형 선언

//입력: 문자열 str1, 최대 길이 MAX\_LEN

//출력: 문자열 str1의 역순 문자열 str2

char \*strRev(char \*str1, char \*str2)

{

STACK s;

int len;

int i;

initStack(&s);

len = strlen(str1);

for (i = 0; i < len; i++)

push(&s, str1[i]);

for (i = 0; i < len; i++)

str2[i] = pop(&s);

str2[i] = '\0'; // "\0"

return str2;

}

//입력: Expression exp, 최대 길이 MAX\_LEN

//출력: Expression exp의 현재 주소 값을 int로 반환

int getSymbol(Expression exp)

{

return \*exp;

}

//입력: Expression exp, 최대 길이 MAX\_LEN

//출력: void

void testPair(Expression exp) //입력한 수식어의 괄호 검사

{

STACK s;

int symbol; //현재 괄호 저장 변수

int exit = 1; //while문 탈출 변수

int left\_sym; //가장 마지막으로 저장된 괄호 시작 기호

int i = 0;

initStack(&s);

while (exit) //exit = 0이면 탈출

{

//while문이 다시 호출될 때마다 문자열의 다음 문자를 가르킨다.

symbol = getSymbol(exp+i++);

switch (symbol){

case '(':

push(&s, symbol); //스택에 저장

break;

case '{':

push(&s, symbol); //스택에 저장

break;

case '[':

push(&s, symbol); //스택에 저장

break;

case ')':

if (isEmpty(s) == -1){ //스택이 비어있을 때(시작괄호가 없을때)

printf("invalid ')'\n");

exit = 0;

break;

}

else if (isEmpty(s) == 0){ //스택에 정보가 있을때

left\_sym = pop(&s);

switch (left\_sym) { //스택에서 꺼내 괄호가 현재 괄호와 불일치시

case '{':

printf("Missing '%c' before ')'\n", (left\_sym + 2));

exit = 0;

break;

case '[':

printf("Missing '%c' before ')'\n", (left\_sym + 2));

exit = 0;

break;

}

}

break;

case '}': //스택이 비어있을 때(시작괄호가 없을때)

if (isEmpty(s) == -1){

printf("invalid '}'\n");

exit = 0;

break;

}

else if (isEmpty(s) == 0){ //스택에 정보가 있을때

left\_sym = pop(&s);

switch (left\_sym) { //스택에서 꺼내 괄호가 현재 괄호와 불일치시

case '(':

printf("Missing '%c' before '}'\n", (left\_sym + 1));

exit = 0;

break;

case '[':

printf("Missing '%c' before '}'\n", (left\_sym + 2));

exit = 0;

break;

}

}

break;

case ']':

if (isEmpty(s) == -1){ //스택이 비어있을 때(시작괄호가 없을때)

printf("invalid ']'\n");

exit = 0;

break;

}

else if (isEmpty(s) == 0){ //스택에 정보가 있을때

left\_sym = pop(&s);

switch (left\_sym) { //스택에서 꺼낸 괄호가 현재 괄호와 불일치시

case '{':

printf("Missing '%c' before ']'\n", (left\_sym + 2));

exit = 0;

break;

case '(':

printf("Missing '%c' before ']'\n", (left\_sym + 1));

exit = 0;

break;

}

}

break;

case 0: //문자열의 끝에 다달았을때,

if (isEmpty(s) == -1){ //스택이 비어있으면 Correct

printf("Correct!\n");

exit = 0;

break;

}

else if (isEmpty(s) == 0){ //스택에 무언가 있다면 오류 출력

left\_sym = pop(&s);

switch (left\_sym) {

case '{':

printf("Missing '%c' \n", (left\_sym + 2));

exit = 0;

break;

case '(':

printf("Missing '%c' \n", (left\_sym + 1));

exit = 0;

break;

case '[':

printf("Missing '%c' \n", (left\_sym + 2));

exit = 0;

break;

}

}

break;

}

}

}

int main()

{

Expression exp;

while (1) {

printf("Type in Expression: ");

scanf("%s", exp);

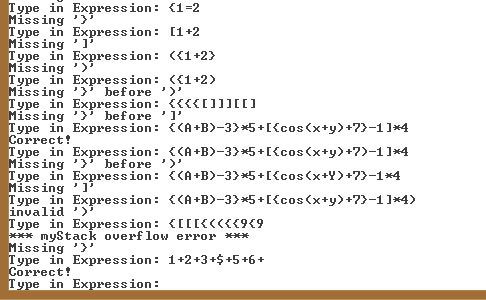
testPair(exp);

}

return 0;

}

3. 테스트 결과



4. 작성자 코멘트

실습 4-1에서 작성한 스택 구조체와 기능 함수들의 정의까지는 실습 안내에 따라 작성하는데 큰 무리는 없었다. 하지만 조금 의아 하면서 헷갈렸던 점이 있는데, 첫번째가 isFull이나 isEmpty 함수 같은 경우 변환 값을 boolean으로 설정하지 않은 게 의아했다. 사실상 별 차이가 없지만, 실습 4-2를 진행하면서 계속 어문이 true, false가 아니어서 굉장히 혼란스러웠었다. 또한, pop함수 같은 경우에도 int를 반환하면서 사실상 false일 때의 리턴 값이 존재 하지 않아 현재는 문제가 없지만, 추후의 implementation 과정에서 문제가 될 것 같다는 생각이 들었다.

막상 실습 4-2를 진행하기에 앞서 알고리즘을 확인할 때, 이론적으로는 전혀 어렵지가 않아서 Pseudo Code 그대로 진행했지만, 많은 오류가 있다는 점을 깨달았다. 하지만 막상 코드 분석 및 디버깅 과정에서 어려움이 많았는데, 일단 switch 문에 대한 경험이 많이 없어서 다시 한 번 복습을 진행해야 했다. 또한 전달 인자의 자료 형 또한 새로 정의해주는 점과 결과값이 true 혹은 false라는점 때문에 한참 헤맸다. 위에서 말했듯이, isEmpty나 isFull의 반환 값이 Boolean이 아닌 int형이라서 전혀 통일이 되지 않은 상황이라 코드 작성에 어려움이 많았다. 또한 Pseudo Code에는 설명되어 있지 않고, 실행 예제 문에만 있는 예외 문들이 문제가 됐었다. 예를 들면, 스택 empty인데 수식 끝에 닫는 괄호가 존재할 경우 같은 경우는 언급되지 않았었다.

위의 문제점들을 어느정도는 해결을 했지만, testPair 함수의 반환 값을 Boolean으로 했을 때이 구현 방법을 행하지 않았다. 반환형이 Boolean일 때는 오류에 대한 출력이 힘들다는 점을 감안해서 함수 내에서 오류 문을 출력 및 모든 과정을 집어넣었다. 현재 작성된 프로그램은 모든 예외 처리가 포함되어 있지만, 한가지는 빠져있다. 오류가 1개 이상일 때의 모든 오류를 테스트 해주지는 않는다는 것이다. 예를 들면 빠져있는 괄호가 2개 이상일 시에는, 가장 마지막으로 열린 괄호의 짝 만을 찾아준다. 이 점들 또한 수정이 가능 하지만, 구체적인 지시가 없었기에 생략하였다.