REPORT

성결대학교
SUNGKYUL UNIVERSITY

과목명 | 데이터구조(5-7)

담당교수 | 김인겸 교수님

학과 | 정보통신공학과

학년 | 2학년

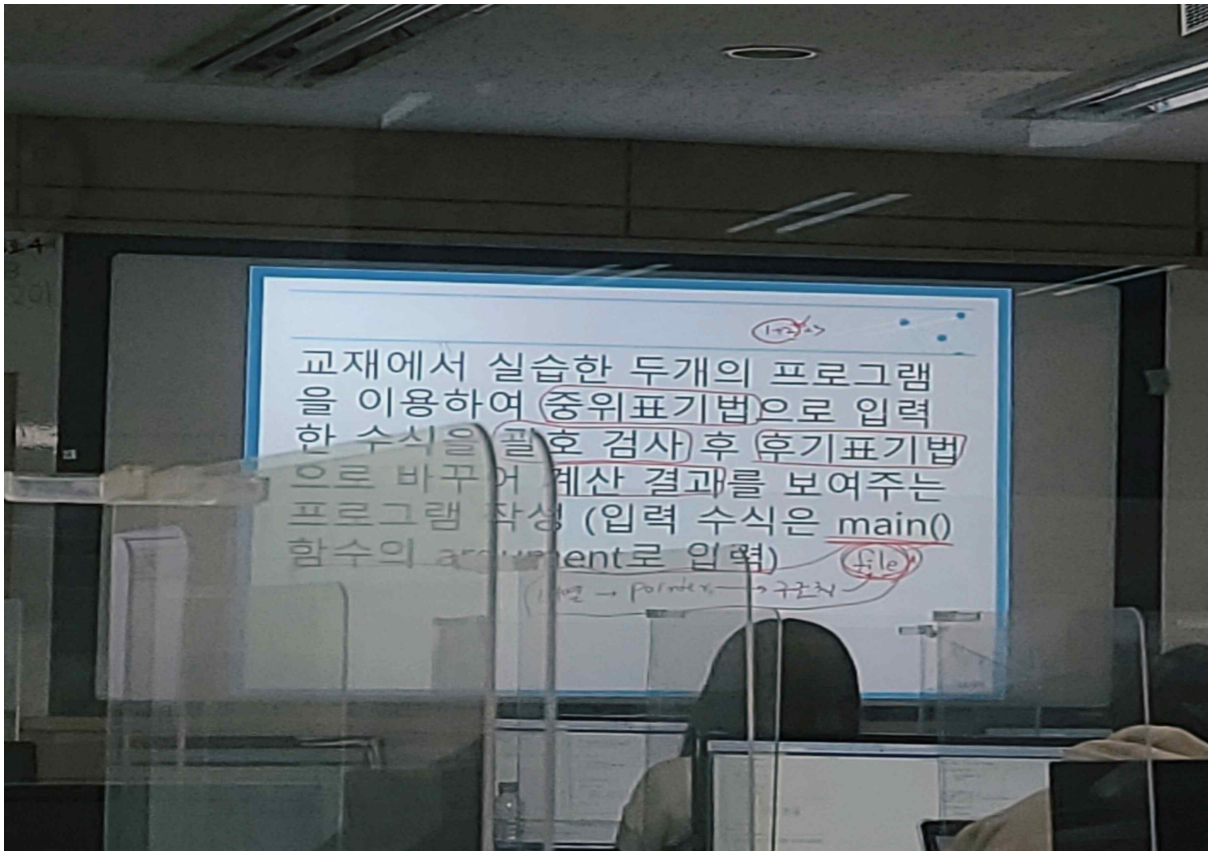
학번 | 20190954

이름 | 허진환

제출일 | 22.04.10

목차

1. 문제 정의



- 프로그램 요구사항 : main함수에서 argument로 입력받은 중위표기수식을 후위표기수식으로 바꾸어 출력되게 하고, 이 후위표기수식를 계산하여 결과도 출력되게 하는 프로그램을 작성한다.

2. 소스 코드

```
#define _CRT_SECURE_NO_WARNINGS
#include <stdio.h>
#include <stdlib.h>

#define MAX_STACK_SIZE 100 // 스택의 크기 100으로 기호상수 선언

typedef double Element; // 스택의 요소 자료형 변환 용이하도록 Element 정의

Element data[MAX_STACK_SIZE]; // 스택 선언

int top; // 스택 맨 위 변수 선언(스택 위치)

/*error 출력 함수*/
void error(char str[])

{
	printf("%s₩n", str);
	exit(1);
}

/*스택 초기화 함수*/
void init_stack() { top = -1; }

/*스택 공백 확인 함수, 스택 맨 위가 -1이면 공백이므로 안됨(이 경우 1 반환)*/
int is_empty() { return top == -1; }

/*스택 포화 확인 함수, 스택 맨위 상태가 스택 크기보다 클 경우 포화상태이면 1 반환*/
int is_full() { return top == MAX_STACK_SIZE - 1; }

/*스택 안에 요소가 얼마나 있는지 확인*/
int size() { return top + 1; }

/*스택에 요소 push*/
void push(Element e)
{
	if (is_full()) // 스택이 포화상태이면
		error("스택 포화 에러₩n");

	data[++top] = e; // 스택 맨위를 선증가시켜 스택 맨위에 요소 push
}

/*스택에 요소 pop*/
Element pop()
{
	if (is_empty()) // 만약 공백상태라면
		error("스택 공백 에러₩n");

	return data[top--]; // 스택 맨 위 값을 반환한 후 스택 맨위 위치를 후감소시킴
}

/*스택에 요소 peek(pop과 달리 스택 맨위 요소 유지)*/
Element peek()
{
```

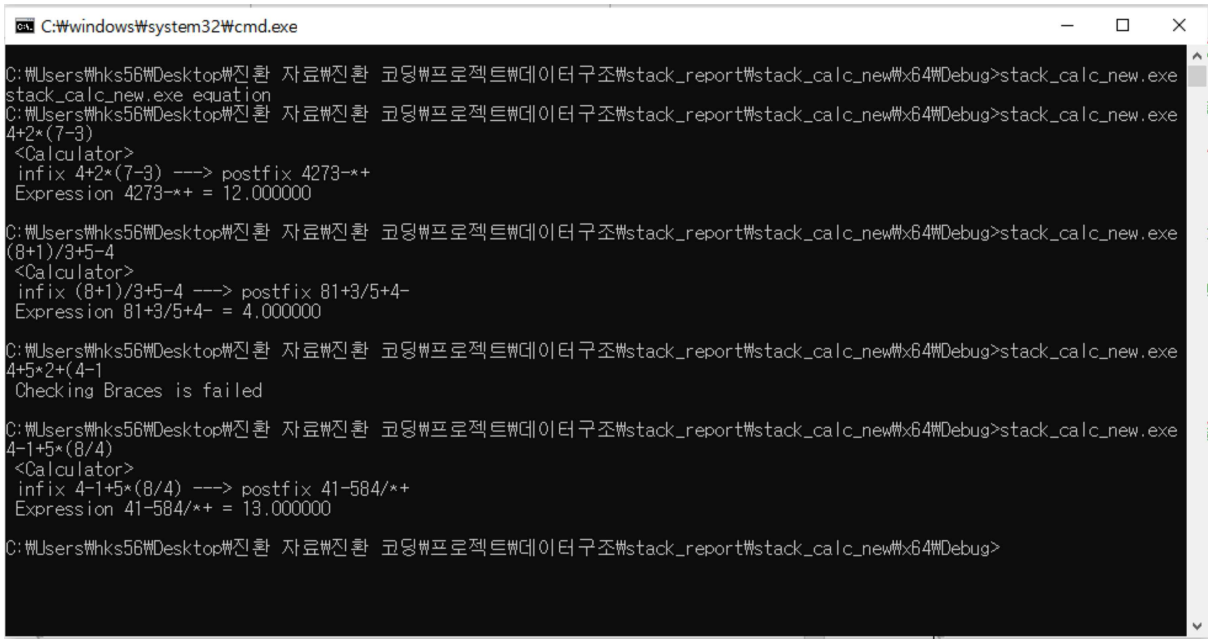
```
        if (is_empty()) // 만약 공백상태라면
                error("스택 공백 에러₩n");

        return data[top]; // 스택 맨 위 값 반환만 시킴
}

/*괄호검사 함수, 인수로 문자열을 받아 조건에 따라 괄호에 오류가 없으면 0반환, 오류 발생 시
조건에 따른 int형 숫자 반환*/
int check_matching(char expr[])
{
        int i = 0, prev; // 함수의 인수 expr[]문자열의 요소 하나하나 읽기 위해 int형 변수 i=0
초기화
        char ch; // 요소 대입용 char 변수 ch선언

        init_stack();

        while (expr[i] != '₩0') { // 함수의 인수 expr[]문자열의 요소 끝까지 읽을때까지 반복
                ch = expr[i++]; // ch에 expr문자열 요소 대입 후 후증가로 expr문자열 요소 모두
읽기

                if (ch == '[' || ch == '(' || ch == '{') // 만약 왼쪽 괄호일 경우
                        push(ch); // 일단 스택안으로

                else if (ch == ']' || ch == ')' || ch == '}') { // 오른쪽 괄호를 만날 경우
                        if (is_empty())
                                return 2; // 조건 2 위반 : 오른쪽괄호부터 나와버림

                        prev = pop(); // 스택 공백X라면 스택 맨 위 요소인 괄호 prev에 대입

                        if ((ch == '[' && prev != ']')

                                || (ch == '(' && prev != ')')

                                || (ch == '{' && prev != '}')) { // 만약 괄호의 쌍이 맞지
않는다면

                                return 3; // 조건 3 위반

                        }
                }
        }

        if (is_empty() == 0) return 1; // 스택 공백상태가 아니라면 조건 1 위반(괄호 쌍이 모두
지어지지 않음)

        return 0; // 조건 1, 2, 3 모두 충족 시 오류 없으므로 0 반환
}

/*후위표기수식 계산 함수
        함수의 인수로 받은 문자열 expr[]에 담긴 후위표기수식을 왼쪽에서 오른쪽으로 스캔
        피연산자 만날 시 바로 스택 저장, 연산자 만날 시 pop 두 번(연산자 하나에 대응되는
피연산자 2개)
        연산한 후 스택에 다시 push하여 스택에 연산결과 넣음
        문자열 expr[]을 모두 읽어 연산이 종료되면 그 값 스택에 push하여 스택에는 연산값만
```

```
        연산값 pop하여 반환시킴
*/
double calc_postfix(char expr[])
{
        char c;
        int i = 0;

        double val, val1, val2; // 피연산자값 소수이므로 double형 선언

        init_stack();

        while (expr[i] != '₩0') {
                c = expr[i++];

                if (c >= '0' && c <= '9') { // 0보다크고 9보다 작은 피연산자일경우
                        val = c - '0'; // 이 숫자를 val에 삽입하고 스택에 push
                        push(val);
                }
                else if (c == '+' || c == '-' || c == '*' || c == '/') { // 연산자일 경우
                        val2 = pop(); // 스택 맨 위 피연산자 pop
                        val1 = pop(); // 그 다음 피연산자 pop
                        switch (c) {
                        case '+': push(val1 + val2); break;
                        case '-': push(val1 - val2); break;
                        case '*': push(val1 * val2); break;
                        case '/': push(val1 / val2); break;
                        }
                }
        }
        return pop();
}

/*연산자 우선순위를 반환하는 함수로 반환되는 숫자가 크면 우선순위 높음*/
int precedence(char op)
{
        switch (op) {
        case '(': case ')': return 0;

        case '+': case '-': return 1;

        case '*': case '/': return 2;
        }

        return -1;
}

/*중위표기수식 -> 후위표기수식으로 변환하는 함수
        중위표기수식 문자열 expr[]과 변환된 후위표기수식 postexpr[] 문자열을 인수로 받음
        expr[] 문자열 요소를 모두 스캔하여 후위표기수식에 맞추어 postexpr[]에 요소가 하나씩
들어가도록 함
        피연산자(숫자)를 만날 시 바로 후위표기수식 postexpr[]문자열에 저장되도록 함
        연산자(괄호)를 만날 시 먼저 계산해야 하므로 왼쪽 괄호라면 스택에 저장하고, 오른쪽
괄호라면 짝이 맞는 왼쪽 연산자를 찾을 때까지 스택에서 먼저 들어가있던 사칙연산자를 pop하여
후위표기수식 postexpr[]문자열에 들어갈 수 있도록 함.
        연산자(사칙연산자)를 만날 시 스택 안에 들어가있는 연산자를 유지하면서 비교하기 위해
```

```
peek한다.
        precedence()함수를 사용하여 스택 안에 들어가있던 연산자와 중위표기수식에서 만난 연산자 간의 우선순위를 비교한다.
        만약 스택 안에 먼저 들어가있던 연산자가 우선순위가 높거나 같다면 postexpr[]문자열에 들어갈 수 있도록 하고, 이 값을 pop하여 스택에서 없앤다.
        만약 스택 안에 먼저 들어가있던 연산자의 우선순위가 낮다면 스택안으로 사칙연산자가 들어가도록 함.
        그 이후 스택에 들어가있던 연산자를 모두 postexpr[]문자열로 들어가도록 한다.
*/
void infix_to_postfix(char expr[], char postexpr[])
{
        int i = 0, j = 0;
        char c, op; // char형 변수 c에는 함수의 인수로 전달받은 문자열인 expr[]의 각 요소값이 차례대로 대입될 것이고, op에는 스택에 저장될 연산자가 pop되어 대입된다.

        init_stack();

        while (expr[i] != '₩0') {
                c = expr[i++];

                if ((c >= '0' && c <= '9'))
                        postexpr[j++] = c; // c에 대입된 값이 피연산자(0~9까지 정수)일 경우, 그대로 postexpr에 일단 저장 (배열에 저장되는 것 = 그대로 출력되는 것과 같은 의미)

                else if (c == '(')
                        push(c);// c에 대입된 값이 왼쪽괄호 '('일 경우 스택 안에 일단 저장

                else if (c == ')') {
                        while (is_empty() == 0) {
                                op = pop(); // c에 대입된 값이 오른쪽괄호 ')'일 경우 스택 안에 저장되어 있는 값을 pop 후 op에 대입

                                if (op == '(') break; // 계속 스택에서 빼면서 스택 안에 먼저 들어가 있었던 '('가 나올 경우 멈춤 () 짝을 찾았기 때문.

                                else postexpr[j++] = op; // '('연산자 나올 때까지 op에 대입된 값을(스택에서 pop된 값을) postexpr에 저장(그대로 출력되는 것과 같은 의미)
                        }
                }

                else if (c == '+' || c == '-' || c == '*' || c == '/') { // char형 변수 c에 대입된 값이 사칙연산자일 때
                        while (is_empty() == 0) {

                                op = peek(); // 스택 안에 저장되어 있는 값을 peek 후 op에 대입(top 값은 유지되도록)

                                if (precedence(c) <= precedence(op)) {
                                        postexpr[j++] = op; // op에 대입된 연산자(스택 안)가 c에 대입된 연산자보다 우선순위 높을 경우 postexpr에 저장

                                        pop(); // 이후 pop하여 스택 top에서 op 제거
                                }
                                else break;
                        }
```

```
                        push(c); // expr[]문자열에서 읽어들인 사칙연산자 c는 스택안으로 저장되어
계속 비교되어 결국 postexpr[j]에 대입되도록 함
                }
        }

        while (is_empty() == 0)
                postexpr[j++] = pop(); //printf("%c ", pop());
}

int main(int argc, char* argv[]) // main함수에 파라미터 전달 / argc = 전달할 정보의 개수, argv =
메인함수에 전달되는 실질적인 정보

{
        char* expr; // 수식을 얼마나 입력받을지 모르므로 char형 포인터 변수 expr 선언
        char postexpr[80] = { "₩0" }; // 후위표기수식이 담길 문자열 postexpr[80]선언 후, 널 문자
넣어줌으로써 초기화

        if (argc != 2) // 메인함수에 전달될 정보가 2개가 아니라면
        {
                printf("%s equation", argv[0]);

                exit(1);
        }

        expr = argv[1]; // 중위표기수식에 프로그램 실행시 사용자에 의해 입력되어 메인함수에
전달되는 argv값을 넣어줌

        if (!check_matching(expr)) // 만약 입력받은 중위표기수식이 괄호검사를 통과했다면
정상적으로 연산 시작
        {
                init_stack();

                printf(" <Calculator>₩n");

                infix_to_postfix(expr, postexpr); // infix_to_postfix(expr[1], postexpr[1]); 과 같음
(선행 코드인 expr = argv[1]; 때문에)

                printf(" infix %s ---> postfix %s₩n Expression %s = %lf₩n", expr, postexpr,
postexpr, calc_postfix(postexpr));

        }
        else
                printf(" Checking Braces is failed ₩n");

        return 0;

}
```

3. 실행 결과



- 메인 함수의 argument가 충분히 입력되지 않을 경우 equation이 출력되고, 괄호가 제대로 닫히지 않았을 경우 Checking Braces is failed가 출력된다. 입력값에 이상이 없는 경우 후위표기수식으로 맞게 변환되어 출력되고, 이에 따라 계산 결과도 맞게 출력된다.