알고리즘 과제-2

156번

20162518

컴퓨터학부

최승서

**4. 스택을 사용하는 정수 계산기**

**a) 소스 코드**

#include<stdio.h>

#include<stdlib.h>

#include<string.h>

#include<math.h>

typedef double operator;

typedef struct stack\_node {

operator data;

struct stack\_node\* link;

}stack\_node;

stack\_node\* top;

void topMakeNull() {

stack\_node\* tmp;

while (top != NULL) {

tmp = top;

top = top->link;

free(tmp);

}

}

void push(operator ch) {

stack\_node\* temp = (stack\_node\*)malloc(sizeof(stack\_node));

temp->data = ch;

temp->link = top;

top = temp;

}

operator pop() {

operator ch;

stack\_node\* temp = top;

if (top == NULL) {

// printf("\n stack is empty! \n");

return 0;

}

else {

ch = temp->data;

top = temp->link;

free(temp);

return ch;

}

}

void postfix\_priority\_1(char\* postfix, char op, int\* p) {

char temp;

while (1)

{

if (top == NULL)

{

break;

}

temp = (char)pop(); // 스택에서 하나를 꺼냄

if (temp == '+' || temp == '-' || temp == '\*' || temp == '/' || temp == '%' || temp == '^') // 연산기호면 배열에 저장

{

postfix[(\*p)++] = temp;

postfix[(\*p)++] = ' ';

}

else // 아니면 다시 스택에 푸시

{

push(temp);

break;

}

}

push(op); // 현재 연산 기호 저장

}

void postfix\_priority\_2(char\* postfix, char op, int\* p) {

char temp;

while (1) {

if (top == NULL) {

break;

}

temp = (char)pop(); // 스택에서 하나를 꺼냄

if (temp == '\*' || temp == '/' || temp == '%' || temp == '^') { // 우선순위가 같은 연산 기호라면

postfix[(\*p)++] = temp;

postfix[(\*p)++] = ' ';

}

else { //아니면 다시 스택에 푸시

push(temp);

break;

}

}

push(op); //현재 연산기호 저장;

}

void postfix\_priority\_3(char\* postfix, char op, int\* p) {

char temp;

while (1) {

if (top == NULL) {

break;

}

temp = (char)pop(); // 스택에서 하나를 꺼냄

if (temp == '^') {

push(temp);

break;

}

else { //아니면 다시 스택에 푸시

push(temp);

break;

}

}

push(op); //현재 연산기호 저장;

}

void postfix\_priority\_default(char\* postfix, char op, char\* exp, int\* p, int i) {

postfix[(\*p)++] = op;

if (exp[i + 1] == '+' || exp[i + 1] == '-' || exp[i + 1] == '\*' || exp[i + 1] == '/' || exp[i + 1] == '%' || exp[i + 1] == ')' || exp[i + 1] == '^' || exp[i + 1] == '\0')

postfix[(\*p)++] = ' ';

}

void postfix\_bloack(char\* postfix, int\* p) {

char temp;

while (1) {

temp = (char)pop(); // 스택에서 하나를 꺼냄

if (temp != '(') { // 열림 괄호가 아니라면

postfix[(\*p)++] = temp; // 문자 배열에 저장

postfix[(\*p)++] = ' ';

}

else

break;

}

}

char\* infix\_to\_postfix(char\* infix) {

topMakeNull();

int i = 0, p = 0;

int len = strlen(infix);

char op;

char\* postfix = (char\*)malloc(len \* 2);

char temp;

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

op = infix[i];

switch (op) {

case '(':

push(op);

break;

case ')':

postfix\_bloack(postfix, &p);

break;

case '\*':

case '/':

case '%':

postfix\_priority\_2(postfix, op, &p);

break;

case '+':

case '-':

postfix\_priority\_1(postfix, op, &p);

break;

case '^':

postfix\_priority\_3(postfix, op, &p);

break;

default:

postfix\_priority\_default(postfix, op, infix, &p, i);

break;

}

}

while (top) { //스택에 남아있는 값을 문자배열에 저장

postfix[p++] = (char)pop();

postfix[p++] = ' ';

}

postfix[p] = '\0'; //문자배열 마지막 값을 널값으로 초기화

return postfix;

}

operator calc\_postfix(char\* postfix) {

int opr1, opr2;

int result, i = 0, cnt = 0;

int len = strlen(postfix);

char op;

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

op = postfix[i];

if (op >= '0' && op <= '9') {

while (postfix[i + cnt] != ' ')

cnt++;

result = atoi(&postfix[i]);

i += cnt;

cnt = 0;

push(result);

}

else {

if (op != ' ') {

opr2 = pop();

opr1 = pop();

switch (op) {

case '^':

push(pow(opr1, opr2));

break;

case '%':

push(opr1 % opr2);

break;

case '\*':

push(opr1 \* opr2);

break;

case '/':

push(opr1 / opr2);

break;

case '+':

push(opr1 + opr2);

break;

case '-':

push(opr1 - opr2);

break;

}

}

}

}

return pop();

}

int error\_check(char\* infix)

{

int len = strlen(infix);

int i = 0;

int cnt = 0;

char symbol, opr;

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

{

symbol = infix[i];

switch (symbol)

{

case '\*':

case '/':

case '^':

opr = infix[i + 1];

if (opr == '+' || opr == '-' || opr == '\*' || opr == '/' || opr == '^')

{

printf("\nERROR\n");

return -1;

}

break;

case '+':

case '-':

opr = infix[i + 1];

if (opr == '+' || opr == '\*' || opr == '/' || opr == '^')

{

printf("\nERROR\N");

return -1;

}

break;

case '(':

push(symbol);

break;

case ')':

if (top == NULL)

{

printf("' ) ' Syntax error!!\n");

return -1;

}

else

{

if (pop() != '(')

{

printf("' ( ' Syntax error!!\n");

return -1;

}

}

break;

default:

if (symbol == ' ') {

break;

}

else if ((symbol < '0' || symbol > '9'))

{

printf("All not number!!\n");

return -1;

}

else

{

cnt++;

}

break;

}

}

if (!cnt || cnt == 1)

{

printf("Nmber is empty or unavailable!!");

return -1;

}

while (top)

{

opr = (char)pop();

if (opr == '(' || opr == '{' || opr == '[' ||

opr == '+' || opr == '-' || opr == '\*' || opr == '/')

{

printf("` %c ' syntax error!!\n", opr);

return -1;

}

}

return 0;

}

int main() {

char infix[200];

char\* postfix;

operator result;

while (1) {

printf("infix expression 입력:");

gets\_s(infix, 200);

fflush(stdin);

if(infix[0] == 'q')

break;

if (error\_check(infix) == -1) {

getchar();

continue;

}

postfix = infix\_to\_postfix(infix);

result = calc\_postfix(postfix);

printf("\npostfix expression 출력: %s", postfix);

printf("\n계산 결과 : %.lf\n", result);

getchar();

}

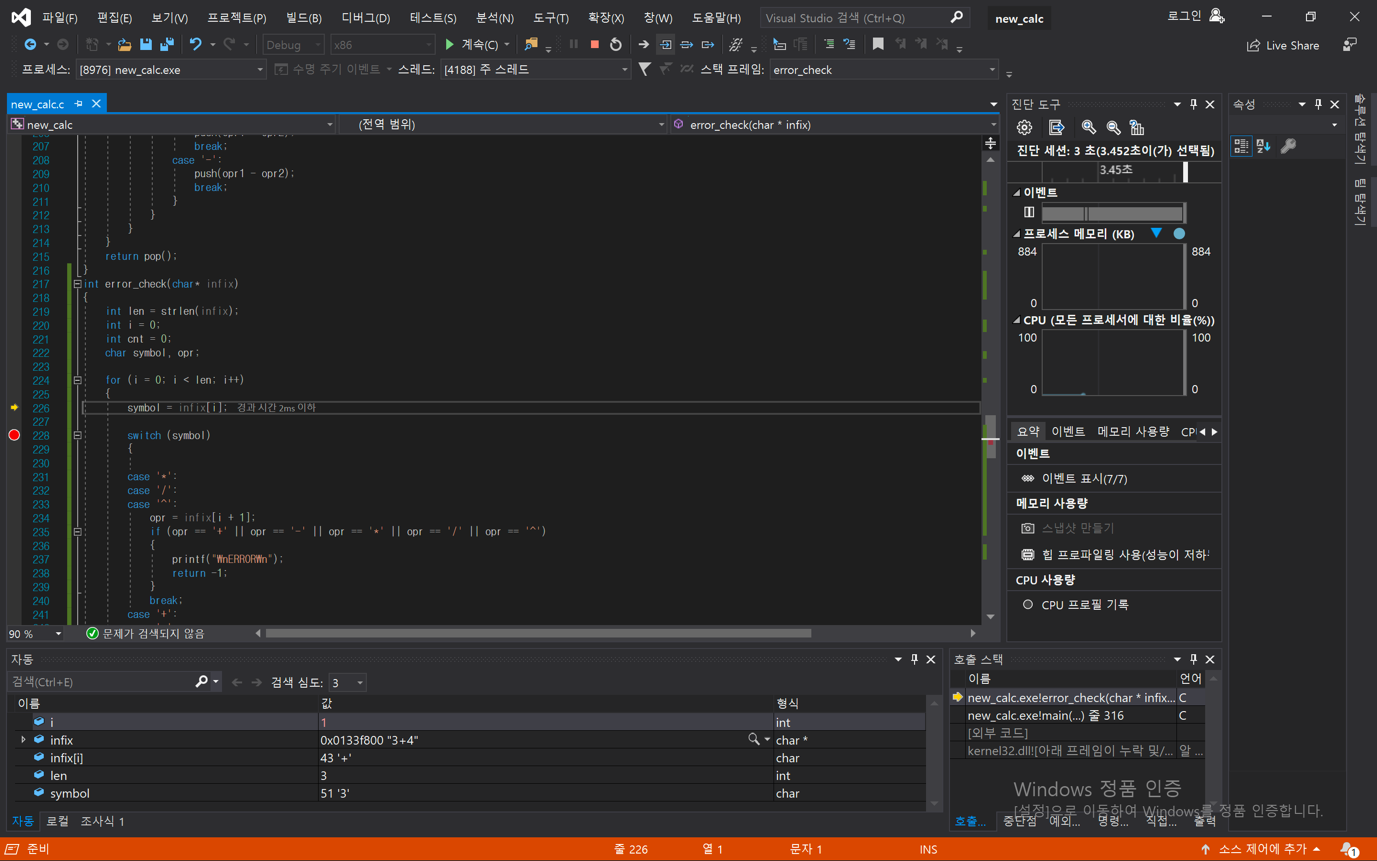
printf(" 프로그램 종료\n");

getchar();

return 0;

}

**b)디버거 사용화면**



**c) 프로그램 실행화면**

