**8th assignment**

2013011491

컴퓨터전공

안찬영

char\* postfix\_to\_infix(char expression[])

{

StackType s;

char token,infix[20];

int i,op1,op2;

int j=0;

init(&s);

for(i=0;expression[i]!='\0';i++)

{

token=expression[i];

if(isalnum(token))

push(&s,token);

else{

if(j==0){

op2=pop(&s);

op1=pop(&s);

infix[j++]='(';

infix[j++]=(char)op1;

infix[j++]=token;

infix[j++]=(char)op2;

infix[j++]=')';

infix[j++]='\0';

}else{

op2=pop(&s);

memmove(infix+1,infix,sizeof(infix)-1);

infix[j++]=token;

infix[j++]=op2;

infix[j++]=')';

infix[j++]='\0';

}

}}

printf("\ninfix\_to\_postfix expression:\n");

printf("%s",infix);

return expression;

}

후위표기를 중위표기로 바꾸는 함수 입니다. 후위표기 되어 있는 식을 앞부터 한 문자 씩 scan해서 숫자일 경우에는 스택에 push합니다. 연산자일 경우에는 첫 연산일 경우 스택에서 operand 2개를 pop하여 infix배열에 operand1, 연산자,operand2 순서로 입력합니다. 첫 연산자가 아닐 경우에, 먼저 memmove로 배열을 왼쪽부터 한칸씩 이동하여 맨 앞에 ’(‘를 만들어 주고, infix배열에 연산자와 operand를 추가 입력합니다. 그후 infix배열을 출력합니다.

int postfixEval(char exp[])

{

StackType s;

char token;

int i,op1,op2;

init(&s);

for(i=0;exp[i]!='\0';i++){

token=exp[i];

if(isalnum(token))

push(&s,token-'0');

else{

op2=pop(&s);

op1=pop(&s);

switch(token)

{

case '+' :

push(&s,op1+op2);

break;

case '-' :

push(&s,op1-op2);

break;

case '\*' :

push(&s,op1\*op2);

break;

case '/' :

push(&s,op1/op2);

break;

}

}

}

return pop(&s);

}

후위표기로 되있는 식을 계산하는 함수입니다.

후위 표기로부터 앞 문자부터 스캔합니다. 숫자일 경우에 스택에 push 해주고, operator일 경우에, stack으로부터 두 개의 operand를 pop하고 switch문을 통해 해당 연산을 수행 후에 다시 그 값을 stack에 push합니다. 그리고 최종값을 pop하여 return합니다.

<결과>

