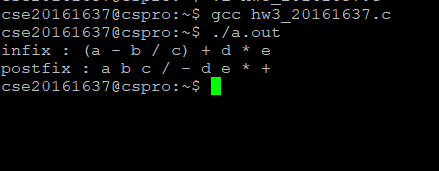
**자료구조 오픈랩 3**

**20161637 장호영**



#define \_CRT\_SECURE\_NO\_WARNINGS

#include <stdio.h>

#include <stdlib.h>

#define MAX\_EXPR\_SIZE 100

typedef enum { lparen, rparen, plus, minus, times, divide, mod, eos, operand, end }

precedence;

char expr[MAX\_EXPR\_SIZE];

int isp[] = { 0, 19, 12, 12, 13, 13, 13, 0,-1 };

int icp[] = { 20, 19, 12, 12, 13, 13, 13, 0,-1 };

//링크드 리스트 선언

typedef struct Node {

precedence data;

struct Node\* next;

}Node;

//스택선언

typedef struct Stack

{

Node\* top;

} Stack;

Stack\* stack;

precedence getToken(char\* symbol, int\* n);

void InitStack();

int IsEmpty();

void Push(precedence data);

precedence Pop();

//스택 시작함수

void InitStack()

{

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

stack->top = NULL;

}

//스택 비었는지 확인

int IsEmpty()

{

return stack->top == NULL;

}

//스택에 데이터 추가

void Push(precedence data)

{

Node\* now = (Node\*)malloc(sizeof(Node));

now->data = data;

now->next = stack->top;

stack->top = now;

}

//스택에서 데이터 꺼내기

precedence Pop()

{

Node\* now;

precedence result;

if (IsEmpty())

{

return eos;

}

now = stack->top;

result = now->data;

stack->top = now->next;

free(now);

return result;

}

// operator 출력함수(스택에서 꺼내면서 출력할 때 씁니다)

void Print\_Operator(int n) {

switch (n) {

case 0: printf("( "); break;

case 1: printf(") "); break;

case 2: printf("+ "); break;

case 3: printf("- "); break;

case 4: printf("\* "); break;

case 5: printf("/ "); break;

case 6: printf("%% "); break;

case 7: printf(" "); break;

}

}

//infix를 postfix로 바꾸기

void postfix() {

char symbol;

int n = 0;

int top = 0;

//마지막 나올때 알려주려고 eos 넣기

Push(eos);

precedence token;

//토큰을 하나씩 읽으면서

for (token = getToken(&symbol, &n); token != end; token = getToken(&symbol, &n)) {

if (token == eos) {

continue;

}

else if (token == operand) //숫자면 바로 꺼냄

printf("%c ", symbol);

else if (token == rparen) { //오른쪽 괄호면 왼쪽 괄호 나올때까지 계속 출력

/\* unstack tokens until left paranthesis \*/

while (stack->top->data != lparen)

{

Print\_Operator(Pop());

}

Pop(); //왼쪽 괄호는 그냥 출력없이 꺼내줌

}

else {

//연산자 우선순위 파악해서 pop 출력

while (isp[stack->top->data] >= icp[token])

{

Print\_Operator(Pop());

}

//스택에 토큰 push

Push(token);

}

}

//이제 문자열 다 봤으면 스택에 남은거 다 출력해주기

while ((token = Pop()) != eos) {

Print\_Operator(token);

}

printf("\n");

}

//토큰 얻는 함수

precedence getToken(char\* symbol, int\* n) {

\*symbol = expr[(\*n)++];

switch (\*symbol) {

case '(': return lparen;

case ')': return rparen;

case '+': return plus;

case '-': return minus;

case '/': return divide;

case '\*': return times;

case '%': return mod;

case ' ': return eos;

case '\n': return end;

default: return operand;

}

}

//파일포인터로 txt파일 읽어서 expr 전역변수에 할당하기

void rExpr(FILE\* fp) {

fgets(expr, sizeof(expr), fp);

printf("infix : %s", expr);

}

int main()

{

InitStack();// 스택초기화

FILE\* fp = fopen("expr.txt", "r");

if(fp==NULL) //에러 체크

return 0;

rExpr(fp);

printf("postfix : ");

postfix();

return 0;

}

함수에 대한 설명은 주석에 자세히 달았으므로 알고리즘에 대해 조금 더 설명하겠습니다.

이 코드는 Linked List로 구현된 Stack으로 Infix를 Postfix로 바꾸어주는 코드입니다.

1. 스택의 모습은 링크드 리스트가 새로 들어오면 맨 앞으로 가게 되고 그 맨 앞을 포인팅하는 것이 stack의 top이 되겠습니다. (마치 배열의 0번째 index느낌입니다)
2. 우선 void rExpr(FILE\* fp)로 원하는 infix 문자열을 받아옵니다.
3. void postfix()를 호출합니다.
4. Expr을 읽어가면서 token을 얻고 그 토큰이 숫자라면 바로 출력합니다.
5. 읽은 토큰이 operator라면 스택에서 가장 상위에 있는 연산자와 우선순위를 비교하여 우선순위가 작으면 출력해주고 아니라면 그냥 스택에 넣어줍니다.
6. 이러는 도중 )를 만나면 (전까지 그냥 다 출력해버리고 (는 출력하지 말고 스택에서 꺼냅니다.
7. 반복하다가 expr이 끝에 도달하면 스택에 남아있는 연산자들을 다 꺼내주고 끝냅니다.