자료구조응용

10. 스택 응용 - 후위 표기 수식 (15점)

2025.10.1.

1. [후위 표기 수식 계산] 후위 표기법(postfix notation)으로 표현된 하나의 수식을 파일로부터 입력받아 그 계산결과를 화면에 출력하는 프로그램을 작성하시오. (5점)

[프로그램 설명]

입력파일(**input.txt**) : 62/3-42*+
사용되는 연산자 : +, -, *, /, %
사용되는 피연산자 : 1~9 사이의 한 자리 정수
'(', ')' 연산자는 입력되지 않음
divide by zero에 대한 테스트 및 처리는 구현하지 않음
입력수식의 문자열 길이는 최대 80으로 함

```
int stack[MAX_STACK_SIZE];
int top = -1;
```

```
typedef enum {lparen, rparen, plus, minus, times, divide,
              mod, eos, operand} precedence;
```

```
int eval(void)
{/* evaluate a postfix expression, expr, maintained as a
    global variable. '\0' is the the end of the expression.
    The stack and top of the stack are global variables.
    getToken is used to return the token type and
    the character symbol. Operands are assumed to be single
    character digits */
  precedence token;
  char symbol;
  int op1, op2;
  int n = 0; /* counter for the expression string */
  top = -1;
  token = getToken(&symbol, &n);
  while (token != eos) {
    if (token == operand)
      push(symbol-'0'); /* stack insert */
    else {
      /* pop two operands, perform operation, and
         push result to the stack */
      op2 = pop(); /* stack delete */
      op1 = pop();
      switch(token) {
        case plus: push(op1+op2);
                   break;
        case minus: push(op1-op2);
                    break;
        case times: push(op1*op2);
                    break;
        case divide: push(op1/op2);
                     break;
        case mod: push(op1%op2);
      }
    }
    token = getToken(&symbol, &n);
  }
  return pop(); /* return result */
}
```

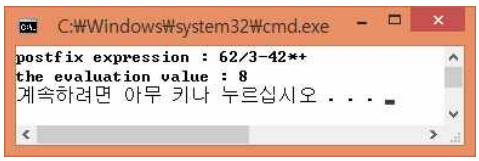
Program 3.13: Function to evaluate a postfix expression

```
precedence getToken(char *symbol, int *n)
{/* get the next token, symbol is the character
    representation, which is returned, the token is
    represented by its enumerated value, which
    is returned in the function name */
   *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 '\0' : return eos;
      default  : return operand; /* no error checking,
                                    default is operand */
   }
}
```

Program 3.14: Function to get a token from the input string

※ '(', ')'의 경우 본 문제에서는 사용되지 않음

[실행 예]



[문자열 파일 입력 시 주의 사항]

```
#define MAX_EXPR_SIZE 80
char expr[MAX_EXPR_SIZE];  // postfix expression
int main(void)
{              ⋮
        FILE *fp;
        int i = 0;
        if((fp = fopen("input.txt", "r"))  == NULL )
        {       fprintf(stderr, "cannot open the file");
                exit(EXIT_FAILURE);
        }

        // <<<<<<<<<<<<<<<<<< 문자열 파일입력 방법 1~4 >>>>>>>>>>>>>>>>>>>>>>>>
        // 파일 끝의 white space(엔터키, 탭, 스페이스바 공백 등) 유무에 상관없이
        // 마지막 문자(1번 사례의 경우 '+')가 제대로 입력됨

        // 입력방법 1
        fscanf(fp, "%s", expr);

        // 입력방법 2
        // 주의: %c 다음에 공백 있어야 파일끝 엔터키 여부에 상관없이 제대로 입력함
        /*
        fscanf(fp, "%c ", &expr[i++]);
        while( !feof(fp))
                fscanf(fp, "%c ", &expr[i++]);
        */
```

```
        // 입력방법 3
        /*
        int ret = fscanf(fp, "%c ", &expr[i++]);
        while ( ret != EOF)
                ret = fscanf(fp, "%c ", &expr[i++]);
        */

        // 입력방법 4
        /*
        while (fscanf(fp, "%c ", &expr[i++]) != EOF)
                ;
        */

        // 입력방법 5
        // 주의: 파일 끝에 엔터키 사용하지 말 것. 파일 끝 엔터키가 그대로 expr 배열로 입력됨
        // fgets(expr, MAX_EXPR_LEN, fp);
          ⋮
        return 0;
}
```

2. [중위 표기 수식에서 후위 표기 수식으로 변환] 중위 표기법(infix notation)으로 표현된 하나의 수식을 파일로부터 입력받아 후위 표기법(postfix notation)으로 변환하여 화면 및 파일에 동시에 출력하는 프로그램을 작성하시오. 또한 실행 예에 대해 입출력 및 스택 상태를 보여주는 테이블을 작성하시오. (구현 5점, 표 그리기 5점)

[프로그램 설명]

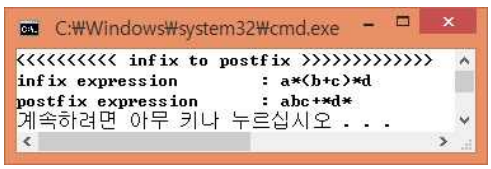
입력 파일("input.txt") : a*(b+c)*d 화면 출력 & 파일 출력("output.txt") : abc+*d* ※ 입력 수식의 문자열 길이는 최대 80으로 함
사용되는 연산자 : +, -, *, /, %, (,) 사용되는 피연산자 : 알파벳 소문자, 1~9 사이의 한 자리 정수 ※ 피연산자가 모두 1~9의 한 자리 정수면, 출력 결과를 1번 문제의 입력으로 사용 가능
posfix(), printToken() 함수의 화면출력 부분에 파일출력 추가 void printToken(precedence); 직접 구현

```
typedef enum {lparen, rparen, plus, minus, times, divide,
                        mod, eos, operand} precedence;

/* isp and icp arrays -- index is value of precedence
   lparen, rparen, plus, minus, times, divide, mod, eos */
int isp[] = {0,19,12,12,13,13,13,0};
int icp[] = {20,19,12,12,13,13,13,0};

precedence stack[MAX_STACK_SIZE];
top = -1;
```

```
void postfix(void)
{/* output the postfix of the expression. The expression
    string, the stack, and top are global */
  char symbol;
  precedence token;
  int n = 0;
  top = 0;    /* place eos on stack */
  stack[0] = eos;
  for (token = getToken(&symbol, &n); token != eos;
                          token = getToken(&symbol,&n)) {
    if (token == operand)
      printf("%c", symbol);
    else if (token == rparen) {
      /* unstack tokens until left parenthesis */
      while (stack[top] != lparen)
        printToken(pop());
      pop();  /* discard the left parenthesis */
    }
    else {
      /* remove and print symbols whose isp is greater
         than or equal to the current token's icp */
      while(isp[stack[top]] >= icp[token])
        printToken(pop());
      push(token);
    }
  }
  while ( (token = pop()) != eos)
    printToken(token);
  printf("\n");
}
```

Program 3.15: Function to convert from infix to postfix

[실행 예]



Token	Stack [0]	[1]	[2]	[3]	Top	Output
a	*eos*				0	*a*
*	*eos*	*			1	*a*
(	*eos*	*	(		2	*a*
b	*eos*	*	(		2	*ab*
+	*eos*	*	(	+	3	*ab*
c	*eos*	*	(	+	3	*abc*
)	*eos*	*			1	*abc* +
*	*eos*	*			1	*abc* +*
d	*eos*	*			1	*abc* +**d*
eos					-1	*abc* +**d**

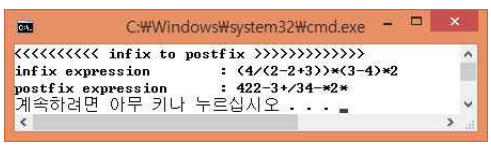


표 그리기 (2점)

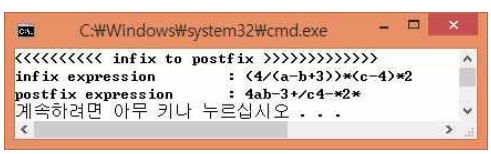


표 그리기 (3점)

■ 제출 형식

- 솔루션 이름　: DS 10
- 프로젝트 이름 : 1, 2
- 각 소스파일에 주석처리

  “학번 이름”

  “본인은 이 소스파일을 다른 사람의 소스를 복사하지 않고 직접 작성하였습니다.”
- 제출 파일

  ① 소스코드와 실행 결과가 보이도록 화면을 캡쳐한 **보고서** 파일(**“학번.pdf”**)

  ※ 한글 [파일 → pdf로 저장하기...] 메뉴 사용

  ② **C 소스 파일**을 하나의 디렉터리에 모아 압축한 파일 (**“학번.zip”**)

  ※ “학번.pdf”와 “학번.zip”을 하나로 압축하지 말고 별도 파일로 제출

■ **주의**

- 소스 복사로는 실력향상을 기대할 수 없습니다!!!
- 1차 마감 : 수업일 자정
- 2차 마감 : 수업 익일 자정(만점의 60%, 반올림)
- 문항 별로 1차 2차 나눠서 제출할 수 없으며, 최종 제출 시간에 따라 1차, 2차로 구분함