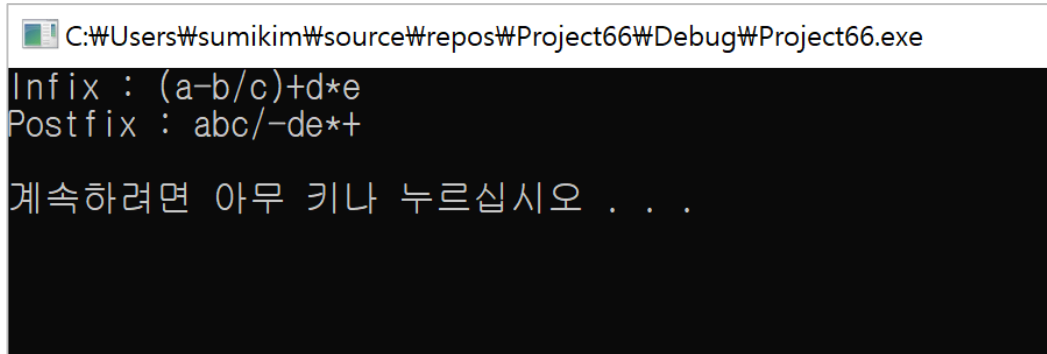


## 오픈랩 과제 #4 : Stacks & Queues

2020. 10. 16 (금) / 20181202 김수미

### 1. 코드 실행 결과



```
C:\Users\sumikim\source\repos\Project66\Debug\Project66.exe
Infix : (a-b/c)+d*e
Postfix : abc/-de*+

계속하려면 아무 키나 누르십시오 . . .
```

### 2. 코드 및 알고리즘 설명

교재에는 일반 배열로 구현된 Stack을 이용해 Infix to Postfix 알고리즘을 구현했는데, 해당 코드를 변형하여 Linked List로 구현된 Stack을 사용하려 하니 너무 헛갈리는게 많아 교재 내용을 일부만 참고해 처음부터 코드를 다시 작성했다.

```
1  #include <stdio.h>
2  #include <stdlib.h>
3  #include <string.h>
4  #define MAX_EXPR_SIZE 100
5  char expr[MAX_EXPR_SIZE];
6
7  // linked list stack을 구성하는 node
8  typedef struct node {
9      char data;
10     struct node *next;
11 } node;
12 node *top = NULL; // top of stack
13 node *pstart = NULL; // point token
```

먼저 linked list stack의 각 node가 되는 구조체 node를 선언해주었다. 해당 node는 다항식의 token값을 저장하는 char data와 다음 node로 연결해주는 struct node \*next로 구성되어 있다.

node \*top은 스택의 top을, node \*pstart는 매번 stack을 훑어나가면서 해당 값을 다른 인자에 넘겨주거나 출력하기 위한 용도로 생성되었다.

```

15 // insert token to pstart
16 void insert(char a) {
17     struct node *t, *child;
18     child = (struct node *)malloc(sizeof(struct node));
19     child->next = NULL;
20     child->data = a;
21
22     t = pstart;
23     if (pstart == NULL) pstart = child;
24     else {
25         while (t->next != NULL) t = t->next;
26         t->next = child;
27     }
28 }

```

void insert 함수는 pstart 에 노드 정보를 채워 pstart값을 활용할 수 있게 해주는 함수이다. data 정보와 어느 노드에 링크를 연결해야 하는지를 해당 함수를 통해 전달받게 된다.

```

30 // stack push operation
31 void push(char token) {
32     struct node *p;
33     p = (struct node *)malloc(sizeof(struct node));
34     p->data = token;
35
36     if (top == NULL) {
37         top = p;
38         p->next = NULL; }
39     else { p->next = top; top = p; }
40 }
41
42 // stack pop operation
43 char pop() {
44     struct node *x;
45     char k;
46     if (top == NULL) {
47         printf("stack underflow\n");
48         return 0; }
49     else {
50         x = top;
51         top = top->next; k = x->data;
52         free(x);
53         x = NULL;
54         return k; }
55 }

```

void push 함수와 void pop 함수는 linked list stack에서 값을 push, pop하는 함수이다. 말 그대로 push 함수는 스택에 값을 집어 넣고, pop 함수는 스택의 값을 반환한다.

```

57      // print postfix expression
58      void printpost() {
59          struct node *to;
60          if (pstart == NULL)
61              printf("");
62          else {
63              to = pstart;
64              while (to != NULL) {
65                  printf("%c", to->data);
66                  to = to->next;
67              }
68          }
69

```

void printpost 함수는 Posfix 수식을 출력하는데 사용되는 함수이다. 뒤에 postfix함수에서 스택 내부의 token값의 우선순위를 하나씩 체크하면서 출력해 줄 때 사용된다.

```

70      // precedence selector
71      int precedence(char ch) {
72          if (ch == '*' || ch == '/' || ch == '%')
73              return (4);
74          else if (ch == '+' || ch == '-')
75              return (3);
76          else
77              return (2);
78      }
79

```

int precedence 함수는 Infix 수식으로부터 입력 받은 token값을 Postfix형식으로 출력하기 위해 값들 사이의 우선순위를 비교하는 함수이다. 연산자는 +, -, \*, /, % 만 포함하기 때문에 +, - 의 우선순위를 동등하게, \*, /, % 의 우선순위를 동등하게 맞춰 주되 \*, /, %의 우선순위를 +, - 보다 높게 잡아주었다.

```

82 //infix to postfix conversion
83 void postfix(char infix[]) {
84     int len, index = 0;
85     char token, temp;
86     len = strlen(infix);
87
88     while (len > index) {
89         token = infix[index];
90         switch (token) {
91             case '(':
92                 push(token);
93                 break;
94             case ')':
95                 temp = pop();
96                 while (temp != '(') {
97                     insert(temp);
98                     temp = pop();
99                 }
100                 break;
101             case '^':
102             case '+':
103             case '-':
104             case '*':
105             case '%':
106             case '/':
107                 if (top == NULL) push(token);
108                 else {
109                     while (top != NULL && (precedence(top->data) >= precedence(token))) {
110                         temp = pop();
111                         insert(temp);
112                     } push(token);
113                 } break;
114             default:
115                 insert(token);
116                 index++;
117         }
118     }
119     while (top != NULL) {
120         temp = pop();
121         insert(temp);
122     }
123     printpost();
124 }

```

void postfix함수는 본격적으로 Infix다항식의 구성요소들을 순서대로 비교하며 postfix로 출력해주는 핵심 함수이다. 수식을 처음부터 차례대로 읽어가면서 등장하는 연산기호들의 우선순위를 비교해 스택에 저장하고, 순서에 맞게 pop시키며 출력한다.

```

127  int main() {
128      // open expr.txt file
129      FILE* fp;
130      fp = fopen("expr.txt", "r");
131      if (fp == NULL) {
132          fprintf(stderr, "file open error \n");
133          exit(1); }
134
135      // save expr.txt in expr[]
136      int index = 0;
137      while (!feof(fp)) {
138          fscanf(fp, "%s", &expr[index]);
139          index++; }
140      fclose(fp);
141
142      // print Infix and Postfix
143      printf("Infix : %s\n", expr);
144      printf("Postfix : ");
145      postfix(expr);
146      printf("\n\n");
147
148      system("pause");
149      return 0;
150  }

```

함수를 호출해 코드 실행 결과를 확인하는 main함수이다. expr.txt파일을 오픈해 텍스트 파일에 입력 되어있는 Infix 수식을 char expr[ ]에 저장해 주었고, postfix 함수를 호출해 해당 수식의 Postfix Expression을 출력시켜 결과를 확인할 수 있다.