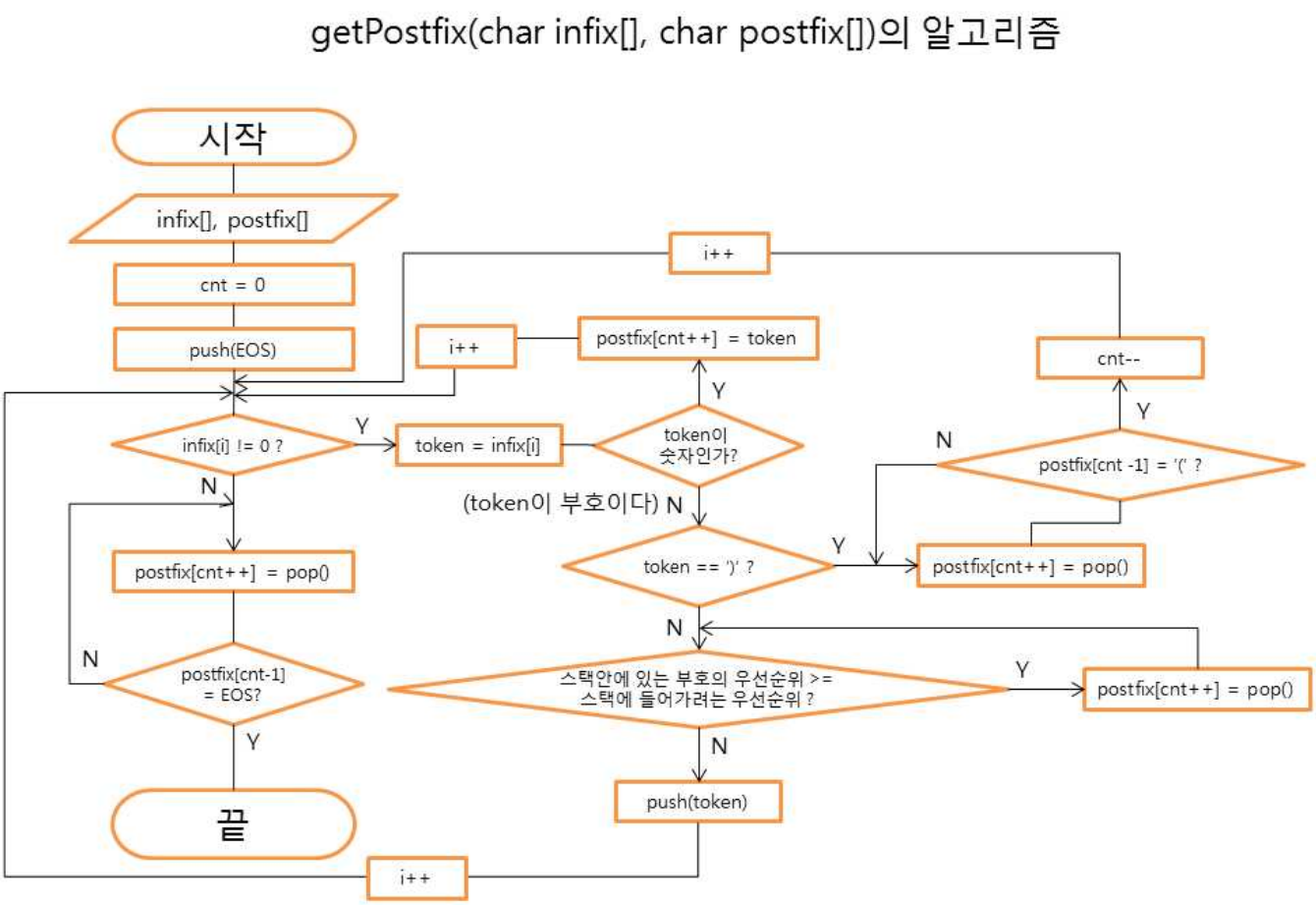
3. 스택을 이용한 postfix와 evaluation

21410785 황 희

1. 개요

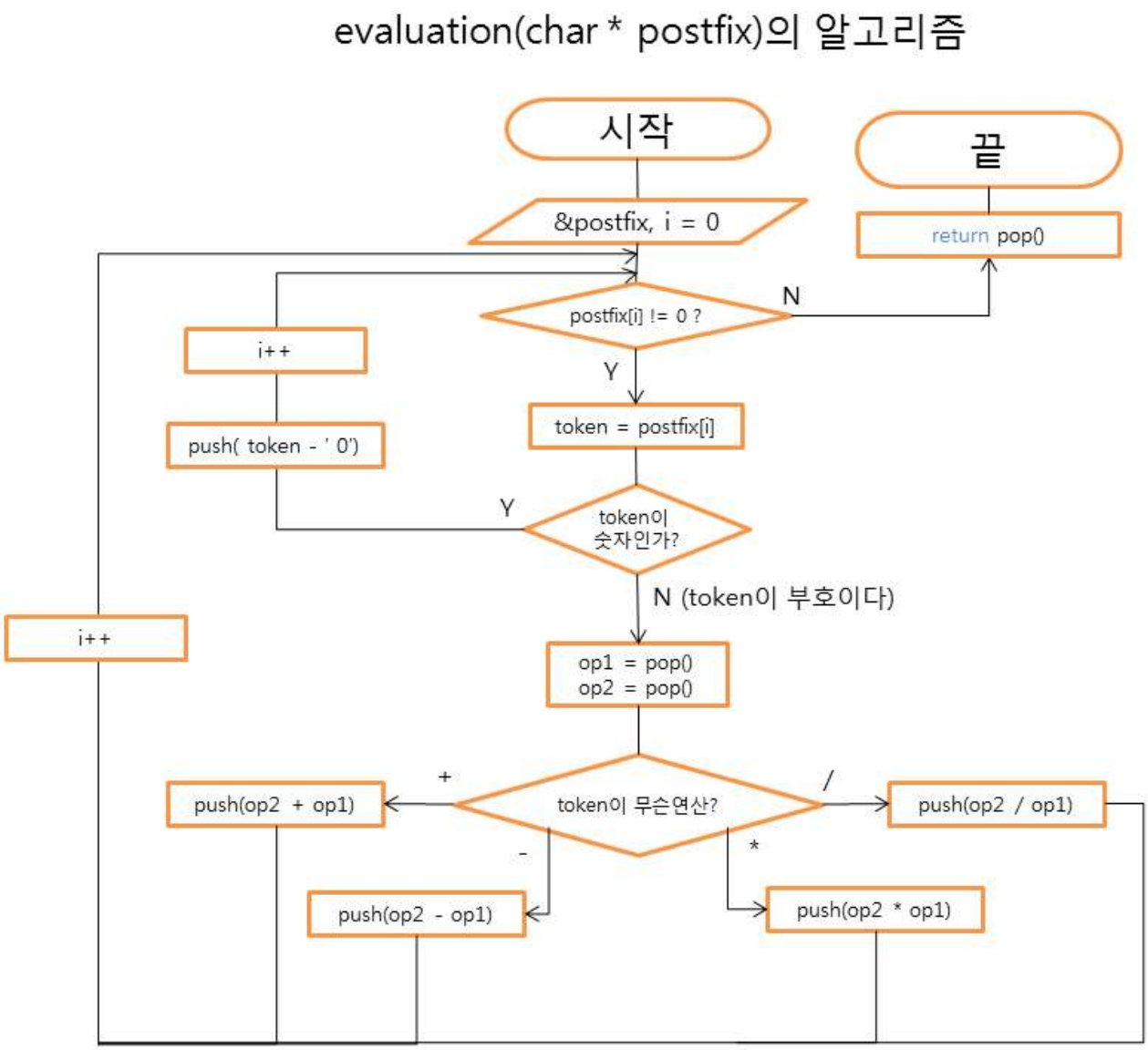
- getPostfix(char infix[], char postfix[])의 알고리즘



getPostfix는 infix(즉, 3*(2+4)와 같이 사람이 계산하는 계산식)을 postfix(324+*와 같이 컴퓨터가 계산하기 편리한 계산식)로 바꿔주는 함수이다. +와 -만 있다면 굳이 postfix로 바꿀 필요는 없지만 *와 /,그리고 괄호가 있기 때문에 우선순위를 생각해 계산하여야 하므로 infix를 postfix로 변환 하는 것이다. 먼저, 메인에서 입력받아둔 계산식 infix[]와 postfix로 변환한 계산식을 넣을 postfix[]를 매개변수로 받는다. infix를 postfix로 변환할 때 임시공간이 필요하므로 전역변수로 스택선언을 해두는데, 스택의 젤 처음에 EndOfStack이라고 스택의 마지막위치를 알려주는 변수를 넣는다. 그리고, infix의 내용을 확인하는데 infix[i]에 내용이 존재한다면 token이라는 임시변수에 infix[i]의 내용을 삽입한다. 그리고 token이 숫자인지 부호인지 체크를 한다. 만약 숫자라면 postfix[cnt]에 숫자를 넣고, cnt에 1을 더한다. 만약 token이 부호라면 부호가 오른쪽괄호')' 인지 확인하고 오른쪽괄호가 아니라면 다시 현재 스택에 있는 부호의 우선순위와 스택에 들어가려는 부호의 우선순위를 비교해준다. 들어가려는 부호의 우선순위가 크다면 스택에 token을 쌓아주고, 원래 있던 부호의 우선순위가 더 크다면 스택에 쌓여있는 부호를 꺼내서 우선순위를 비교하는 작업을 스택에 쌓이려는 부호의 우선순위가

더 클 때까지 반복해준다. token이 오른쪽 괄호인지 확인하는 작업에서 만약 token이 오른쪽 괄호')'라면 왼쪽괄호'('가 있다는 뜻이므로 스택에서 왼쪽괄호'('가 나올 때까지 스택에서 부호를 꺼내준다. 왼쪽 괄호가 나온다면 cnt에 1을 빼주고, infix[i]에 값이 있는지 확인하는 작업으로 다시 되돌아간다. 만약 infix[i]에 값이 없다면 스택에 쌓여있는 나머지 부호를 모두 빼서 postfix에 넣어준 후, 마지막으로 EOS가 postfix에 들어오면 작업을 종료해준다.

- evaluation(char * postfix)의 알고리즘



evaluation은 infix를 컴퓨터가 계산하기 쉽게 바꾼 postfix로 계산을 실행하도록 하는 함수이다. 먼저, postfix의 주소값을 받는다. 그다음, postfix[i]에 값이 있는지 확인해주고, 있다면 token에 postfix[i]의 내용을 넣는다. 그리고 token이 숫자인지 확인하고, 숫자라면 스택에 숫자를 넣어주고 i에 1을 더해 다시 postfix[i]에 값이 있는지 확인한다. 그리고 만약 token이 연산자라면 op1에는 스택에 있는 숫자를 꺼내고(연산자 뒤에 있는 숫자),op2에는 그 다음 수를 스택에서 꺼낸다(연산자 앞에 있는 숫자) 그다음 token이 무슨 연산자인지 확인하고, 그

연산자에 맞게 (op2 (연산) op1)을 하여 스택에 쌓아준다. 이를 postfix의 마지막 인덱스까지 반복해주면 최종 연산결과만 스택에 남게 되므로 i가 postfix의 제일 마지막 배열의 위치라면 스택에 있는 값을 꺼내 리턴하면 최종 연산결과가 된다.

2. 본문

- 코드(설명은 주석으로 대체)

```
#include<stdio.h>
#define MAXSTACK 100 // 최대스택의 크기를 100으로 제한
#define EOS 0 // 자료의 끝 EOS를 0으로 상수화
char stack[MAXSTACK]; // 스택을 사이즈가 MAXSTACK인 배열로 생성
int top = -1; // 제일 꼭대기 스택의 위치를 나타내는 변수
char pop() // 스택에서 제일 꼭대기 값을 꺼내는 함수
{
	if (top == -1) // 스택안에 아무것도 들어있지않다면
	{
		printf("empty");
		return top;
	}
	return stack[top--]; // 스택안에 내용이 들어있다면 꼭대기의 내용 리턴
}
char push(char data) // 스택 꼭대기에 값을 쌓는 함수 매개변수로 data를 받음
{
	if (MAXSTACK - 1 == top) // 스택이 꽉 찼다면
		return 0; // 0을 리턴
	return stack[++top] = data; // 스택에 data값을 쌓음
}
int isdisit(char token) // token이 숫자인지 부호인지 구별하는 함수
{
	if ('0' <= token && token <= '9') // 토큰이 숫자라면
		return 1; // 1을 리턴
	else // 부호라면
		return 0; // 0을 리턴
}

int evaluation(char * postfix) // postfix를 받아 계산하는 함수
{
	char i, op1, op2;
	char token;
	for (i = 0; postfix[i] != 0; i++) // postfix[i]에 값이 있는 동안 계산해주는 반복문
	{
		token = postfix[i]; // postfix[i]의 값을 token에 넣는다
		if (isdisit(token)) // token이 숫자라면
			push(token - '0'); // 스택에 숫자를 쌓는다
		else // token이 부호라면
		{
```

```
                        op1 = pop(); op2 = pop();
                        // op1과 op2에 계산될 숫자를 꺼내서 넣어준다
                        switch (token) // 부호별 계산해주는 switch문
                        {
                        case '-': push(op2 - op1); break;
                        case '+': push(op2 + op1); break;
                        case '*': push(op2 * op1); break;
                        case '/': push(op2 / op1); break;
                        }
                }
        } // 스택에 있는 모든 값이 계산 될때까지 반복
        return pop(); // 최종결과를 꺼낸다
}

int getPriority(char token) // 부호를 스택에 넣을 때의 우선순위를 반환해주는 함수
{
        switch (token)
        {
        case '(':                                       //숫자는 우선순위를 나타냄
        case ')': return 10;            //괄호가 제일 우선순위가높음
        case '*':
        case '/': return 5;
        case '+':
        case '-': return 2;
        }
        return 0;
}

int getInStackPriority(char token)    //스텍내의 연산자 우선순위를 숫자로 보내줌.
{
        switch (token)
        {
        case '(': return 1;
        case ')': return 10; // 오른쪽 괄호는 등장과 동시에 괄호안의 부호를 모두 꺼내야하므로
        case '*':           // 우선순위가 제일 크다
        case '/': return 5;
        case '+':
        case '-': return 2;
        }
        return 0;
}

void getPostfix(char infix[], char postfix[]) // infix를 postfix로 변환해주는 함수
{
        int i, cnt = 0;
        char temp;
```

```
	push(EOS); // 스택의 제일 바닥을 EOS로 표시해준다
	for (i = 0; infix[i] != 0; i++) // infix에 값이 있는동안 반복
	{
		char token = infix[i]; // infix[i]의 값을 token에 입력
		if (isdisit(token))                                  // token이 숫자라면
			postfix[cnt++] = token;
			// postfix[cnt]에 token을 넣어주고, cnt를 1증가시킨다
		else // token이 부호라면
		{
			if (token == ')') // token이 ')'라면
			{
				do {
					postfix[cnt++] = pop();
					// postfix[cnt++]에 스택에서 팝한 값을 넣는다.
				} while (postfix[cnt - 1] != '(');
					// postfix[cnt - 1]이 '('일 때까지
				cnt--; // cnt -1
				continue; // 작업을 계속 반복함
			}
			while (getInStackPriority(stack[top]) >= getPriority(token))
	// 스택에 있는 부호의 우선순위와 스택에 집어넣으려는 부호의 우선순위를 비교하여
		// 스택내에 있는 부호의 우선순위가 더 크다면
				postfix[cnt++] = pop();
			// postfix[cnt++]에 부호가 쌓여있는 스택의 꼭대기값을 집어넣는다
			push(token); // token을 스택에 쌓아준다.
		}

	}
	do {
		postfix[cnt++] = pop(); // postfix[cnt++]에 스택의 값을 꺼내서 넣어준다
	} while (postfix[cnt - 1] != EOS); // 스택에 쌓여있는 값이 없어질때까지
}
int main()
{
	char infix[20], postfix[20];
	gets_s(infix); // infix를 입력받는다
	getPostfix(infix, postfix); // infix를 주고 postfix를 받는다
	printf("postfix : %s\n", postfix);
	printf("evaluation reust : %d\n", evaluation(postfix)); // postfix의 계산결과 출력
	return 0;
}
```

- 결과창

연산식을 infix형식(3+2*(6-2)/3)으로 입력하자 getPostfix함수에서 연산자의 우선순위에 따라 postfix형식(3263-*3/+)으로 바꾸어 postfix형식의 연산식을 출력해주었고, 이 postfix를 가지고 evaluation함수로 계산하여 최종 결과는 5라고 나오게 되었다.

3. 결론

컴퓨터는 연산자의 우선순위를 지정해주지 않으면 모르므로 우리가 평상시에 사용하는 수학 연산식을 계산하라고 주면 우리가 아는 답과는 다른 값이 나온다. 연산자의 우선순위에 따라 계산을 하도록 하려면 각 연산자에게 우선순위를 숫자로 지정하고, 스택이 존재하여야 한다. 스택의 FIFO(First In First Out)특징을 이용하여 계산을 해주는 것인데, 그 전에 FIFO식으로 계산을 하려면 우리가 사용하는 계산식을 컴퓨터가 이해하기 쉽도록 바꿔주는 작업이 필요하다. 이에 사용하는 함수가 getPostfix함수이고, 결과로 나오는 postfix를 계산하기위해 evaluation함수를 사용하였다. 이 과정에는 전역변수로 선언된 스택배열이 필요하다. 이 코드를 통해 스택배열이 이해가 되었고, 연산식에서 스택배열의 중요성을 알 수 있게 되었다.

2017-12-09

황 희