자료구조 실습과제 #6

중위식 -> 후위식 변환 후 계산하는 프로그램

5671144-강범창

1. 중위식을 입력 받습니다.

```
printf("Enter an infix expression: ");
scanf("%[^\n]s", inputTerm);
```

전위식을 입력 받는 코드입니다. 공백 도 같이 받아드립니다.

2. 입력 받은 중위식을 후위식으로 변환합니다.

```
strcpy(currntInput, infixToPostfix(inputTerm));
printf("Postfix expression: %s\n", currntInput);
```

입력 받은 중위식을 후위식으로 변환 후 그 변환된 문자열을 계산하기에 옳바른 값이라 생각하고 current Input 변수에 복사를 합니다. 그후 출력을 합니다. 아래는 후위식으로 변환하는 코드입니다.

```
StackType* stack = create(MAX_STACK_SIZE);
char topOp;
char num[100] = "";
int numIndex = 0;
int idx = 0; // 결과값을 저장하기 위해서 idx 를 사용합니다.
char result[300] = "";
```

우선 후위식으로 변환하기 전에 후위식으로 변경을 하기위해 stack 과 괄호의 시작을 알수있는 topOp, 223 같은 붙어있는 숫자를 저장하기 위한 num 배열, 그리고 그 길이를 알기 위해 numIndex 변수를 선언합니다.
그리고 결과를 return 하기 위해 result 배열을 선언해줍니다.
이때 result 배열의 인덱스로 idx 라는 변수를 선언해주었습니다.
이후 입력 받은 값이 문제가 있는지 없는지 error check 를 하게됩니다.

에러 체크의 경우 3 가지 경우로 처리를 하였습니다. 괄호가 닫쳐 있지 않은 경우, 중위식이 아닌 경우, 잘못된 문자열 이 입력된 경우로 하였습니다. 우선 괄호가 닫쳐 있지 않은 경우는 열려 있는 경우만 체크하여, 열려 있다면 닫쳐 있지 않은 것 이므로 에러, 닫는 것만 있을 때도 마찬가지로 처리하여 주었습니다, 두번째로 값이 이상하다면, 즉 + 로 시작하거나 ++ 의 경우를 막기위해 prev 이전의 값을 확인하여 숫자나 괄호가 아니라면 에러가 나오도록 하였습니다. 그리고 이상한 문자가 올 경우는 숫자가 아니고 , 정해진 연산자가 아닐 경우에 에러를 출력하도록 하였습니다.

```
errorCheck(exp);
--------------------------------------------------------------------------------
char prev = ' ';
        bool isOperOpend = false;

        for (int ei = 0; ei < strlen(exp); ei++)
        {
                char ch = exp[ei];
                if (!isspace(ch)) // 공백인지 확인.
                {
                        if (isdigit(ch)) {
                                prev = '1';
                                continue;
                        }
                        if (ch == '+' || ch == '-' || ch == '*' || ch == '/' || ch == '('
|| ch == ')') {
                                if (ch == '(') {
                                        isOperOpend = true;
                                        prev = '(';
                                        continue;
                                }
                                if (ch == ')') {
                                        if (!isOperOpend) {
                                                fprintf(stderr, "Error : Mismatched
parentheses");
                                                exit(1);
                                        }
                                        isOperOpend = false;

                                        prev = ')';
                                        continue;
                                }
                                // 중위식이기에 오퍼레이터의 이전에는 괄호나 숫자만이
존재할수 있습니다.
                                if (prev != ')' && prev != '(' && !isdigit(prev)) {
                                        fprintf(stderr, "Error : Invalid expression.");
                                        exit(1);
                                }
```

```
                                else {
                                        prev = ' ';
                                }
                        }
                        else {
                                fprintf(stderr, "Error : Invalid character");
                                exit(1);
                        }
                }

        }
        if (isOperOpend) {
                fprintf(stderr, "Error : Mismatched parentheses");
                exit(1);
        }
```

2. 후위식으로 변경된 값들을 계산합니다.

이후로는 후위식으로 변경된 중위식을 계산을 하는 코드입니다.

```
strcpy(currntInput, infixToPostfix(inputTerm));
        printf("Postfix expression: %s₩n", currntInput);
        /// 계산 ////////////////////////////////
        // currntInput => 222 110 1 + / 3 *

        for (int i = 0; i < strlen(currntInput); i++) {
                if (currntInput[i] != ' ') {
                        ch = currntInput[i];
                        if (isdigit(ch))
                        {
                                num[numIndex++] = ch;
                        }
                        else {

                                op2 = pop(stack);
                                op1 = pop(stack);
                                switch (ch)
                                {
                                case '+':
                                        push(stack, op1 + op2);
                                        break;
                                case '-':
                                        push(stack, op1 - op2);
                                        break;
                                case '*':
                                        push(stack, op1 * op2);
                                        break;
                                case '/':
                                        push(stack, op1 / op2);
                                        break;
                                }
                        }
                }
```

```
                else {
                        if (strlen(num) != 0) {
                                value = atoi(num); // atoi -> 문자열에 있는값을 정수로
바꿔줍니다!
                                push(stack, value);

                                memset(num, 0, 100); // 문자 를 저장하던 배열을 초기화
합니다.
                                numIndex = 0;
                        }

                }
        }
        printf("Result: %d₩n", stack->data[0]);

        free(stack->data);
        free(stack);

        return 0;
```