# 자료구조

과제 3

물리학과 20182326 이선민

# 문제 1번. 계산기

```
[~/2022_1/Data_structure/Assignment/assignment_3]$ gcc calculator.c
[~/2022_1/Data_structure/Assignment/assignment_3]$ ./a.out
2+3*4*4-1
= 49
[~/2022_1/Data_structure/Assignment/assignment_3]$ ./a.out
2+(3*4)*2-12
= 14
[~/2022_1/Data_structure/Assignment/assignment_3]$ ./a.out
2+(3*4*2-12
          ^ 이 위치에 오류가 있습니다.
[~/2022_1/Data_structure/Assignment/assignment_3]$ ./a.out     *[master]
(11+3)*2*3-12
= 72
[~/2022_1/Data_structure/Assignment/assignment_3]$ ./a.out     *[master]
2+(3+*4^2-12
     ^ 이 위치에 오류가 있습니다.
[~/2022_1/Data_structure/Assignment/assignment_3]$ ./a.out
(11.3+6.7)*4.5/2.1-9.8
= 28.771429
```

- 우선 문제에서 중요한 포인트는 세가지 이다.
- 1. 실수 계산 가능
- 2. 괄호 계산 가능
- 3. 잘못된 수식 걸러내기

# 문제 1번 계산기

- 나는 우선 식을 입력받아 check_formula라는 함수를 만들어 여기서 잘못된 수식을 걸러내 주었다. 이 함수는 수식이 정상적일 경우 -1을 반환하고 비정상적일 경우 잘못된 부분의 인덱스를 반환한다. 수식이 잘못되었을 경우 인덱스 부분에 수식이 잘못되었음을 알려주고 프로그램을 종료한다.

- 스택에 get_top이라는 함수를 추가하였다.
- 가장 위에 있는 데이터를 반환해준다. (꺼내는 것은 아니다.)

# 문제 1번 계산기

- 정상적일 경우 calculate라는 함수에서 계산이 이루어 진다.
- 우선 calculate함수에서 to_postfix함수를 호출하여 수식을 후위 계산식으로 바꾼다. To_postfix함수에서는 괄호만 잘 신경써주면서 스택을 활용하여 바꿔주었다.
- 후위 계산식으로 계산할 때도 새로운 스택을 만들어 거기에 숫자를 담았고, 연산자가 있으면 두개를 꺼내 계산해주었다.
- 두개씩 계산할 때는 실수 계산 이기 때문에 get_result라는 함수를 만들어 거기에서 계산을 해주었다.

# 문제 1번 계산기

- Get_result함수는 숫자 1과 2를 담은 배열과, 계산 결과를 담을 스택, 그리고 연산자를 매개변수로 받아서 계산한다. 이때 숫자 1과 2는 저장할 당시에 스택에서 꺼내 바로 저장하기 때문에 거꾸로 뒤집혀져서 저장되어 있어서 다시 원상태로 뒤집어 주는 과정을 추가 해주었다. 그리고 stdlib라이브러리에 있는 atof메소드를 활용해 문자열 형을 실수 형으로 바꾼 후 연산해주고
- 스택에 다시 넣어주기 위해 stdio 라이브러리에 있는 sprintf메소드를 활용해 다시 결과값을 문자열로 바꿔준후 스택에 넣어주었다.
- 이때 뒤에 붙는 불필요한 0들은 반복문을 통해 제거해주었다.
- 각각의 숫자를 꺼낼때 구분할 수 있게 하기 위해 스택에 숫자를 넣을 때, 숫자 사이사이에 공백을 추가해 주었다.

# 문제 1번. 계산기

```
[~/2022_1/Data_structure/Assignment/assignment_3]$ gcc calculator.c
[~/2022_1/Data_structure/Assignment/assignment_3]$ ./a.out
2+3*4*4-1
= 49
[~/2022_1/Data_structure/Assignment/assignment_3]$ ./a.out
2+(3*4)*2-12
= 14
[~/2022_1/Data_structure/Assignment/assignment_3]$ ./a.out
2+(3*4*2-12
           ^ 이 위치에 오류가 있습니다.
[~/2022_1/Data_structure/Assignment/assignment_3]$ ./a.out     *[master]
(11+3)*2*3-12
= 72
[~/2022_1/Data_structure/Assignment/assignment_3]$ ./a.out     *[master]
2+(3+*4^2-12
     ^ 이 위치에 오류가 있습니다.
[~/2022_1/Data_structure/Assignment/assignment_3]$ ./a.out
(11.3+6.7)*4.5/2.1-9.8
= 28.771429
```

# 문제 2번 열차 재배치

- 계산기에서와 마찬가지로 스택에 get_top함수를 추가하여 가장 위에있는 데이터를 확인하는 용도로 사용하였다.
- 스택이 2개일때와 3개일때 호출하는 함수를 분리했다.
- 전체적인 아이디어는 스택의 개수에 따라 9개의 숫자를 각각 오름차순으로 집어넣었다.
- 예를 들어 스택이 3개면 첫번째 스택에 234, 두번째에 567, 세번째에 89를 넣었다.
- 1의 경우 나오면 즉시 out되도록 하였다.

# 문제 2번 열차 재배치

- 횟수를 최대한 줄이기 위해 recent_out이라는 변수를 두어 가장 최근에 나간 숫자를 저장시켰다. 만약 최근에 나간 숫자가 2인데 현재 3이 들어왔다면 3은 스택에 저장 안되고 바로 나갈 수 있도록 하였다.
- 그리고 만약 3이 바로 나갔는데 스택중 하나에 가장 위에 4가 있다면 4도 바로 pop하여 out시켰다.
- 숫자를 스택에 집어 넣을 때는 정렬을 하면서 집어넣었다. 가장 작은 숫자가 맨 위로, 가장 큰 수 가 가장 밑으로 가도록 정렬하며 집어넣었다.

# 문제 2번 열차 재배치

- 들어오는 것들을 전부 처리한 이후에는 넣을 때 이미 정렬하며 스택에 넣었으므로 나머지는 그냥 pop하고 out 시키면 된다.

# 문제 2번 열차 재배치

```
[~/2022_1/Data_structure/Assignment/assignment_3]$ gcc train.c
[~/2022_1/Data_structure/Assignment/assignment_3]$ ./a.out
스 택 의  개 수 : 2
열 차  번 호  입 력 : 397182645
01 : IN(3)
02 : PUSH(1, 3)
03 : IN(9)
04 : PUSH(2, 9)
05 : IN(7)
06 : PUSH(2, 7)
07 : IN(1)
08 : OUT(1)
09 : IN(8)
10 : POP(2)
11 : PUSH(1, 7)
12 : PUSH(2, 8)
13 : POP(1)
14 : PUSH(2, 7)
15 : IN(2)
16 : OUT(2)
17 : POP(1)
18 : OUT(3)
19 : IN(6)
20 : PUSH(2, 6)
21 : IN(4)
22 : OUT(4)
23 : IN(5)
24 : OUT(5)
25 : POP(2)
26 : OUT(6)
27 : POP(2)
28 : OUT(7)
29 : POP(2)
30 : OUT(8)
31 : POP(2)
32 : OUT(9)
종 료 (총  32회 )
```

# 문제 2번 열차 재배치

```
[~/2022_1/Data_structure/Assignment/assignment_3]$ ./a.out
스택의 개수 : 3
열차 번호 입력 : 397182645
01 : IN(3)
02 : PUSH(1, 3)
03 : IN(9)
04 : PUSH(3, 9)
05 : IN(7)
06 : PUSH(2, 7)
07 : IN(1)
08 : OUT(1)
09 : IN(8)
10 : PUSH(3, 8)
11 : IN(2)
12 : OUT(2)
13 : POP(1)
14 : OUT(3)
15 : IN(6)
16 : PUSH(2, 6)
17 : IN(4)
18 : OUT(4)
19 : IN(5)
20 : OUT(5)
21 : POP(2)
22 : OUT(6)
23 : POP(2)
24 : OUT(7)
25 : POP(2)
26 : OUT(8)
27 : POP(2)
28 : OUT(9)
종료(총 28회)
```