2021 자료구조 Report

Stack을 이용한 사칙계산기

학년/반 : 2/B

학번: 201844050

이름 : 박승민

목차

1. 문제설명
2. 요구사항 분석
3. 요구사항 명세서
4. 설계
5. Program 작성
6. 출력 Result
7. 참고 문헌
8. 검토

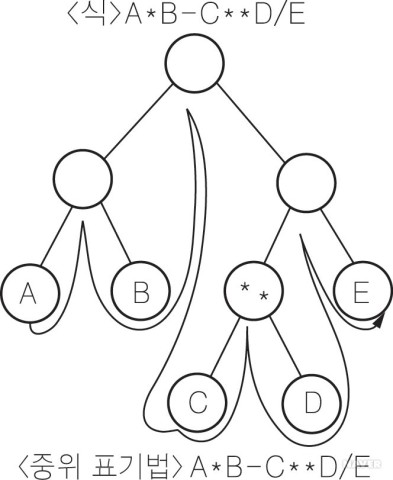
1. 문제설명

- 중위(infix)표기법으로 작성된 수식을 입력 받아 주어진 처리 조건에 맞게 이를 후위(postfix)표기법으로 변환하는 과정과, 후위(postfix)표기법 변환된 수식을 계산하여 결과를 출력하는 과정에 대한 알고리즘을 제시하고, C언어 또는 Java언어로 실행 과정을 구현하는 문제이다.

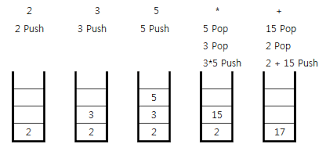
2. 요구사항 분석

중위(infix)표기법/후위(postfix)표기법이 무엇인가?

①중위(infix)표기법이란 연산자를 앞에 표기하고 그다음에 피연산자(ex)+)를 표기하는 기법으로 연산자의 우선순위 규칙에 지배되며 괄호와 같은 단락 기호를 사용하는 수학 상의 식을 구성하는 방법입니다. 각 연산자는 인접하는 피연산자 또는 중간 결과에 대하여 행해지는 연산을 표시하는 것입니다. 예를 들어 A와B를 곱한 값에 C를 더한다고 한다면 AxB+C 라는 식으로 표현하는 것입니다.



②중위 표기법이 우리 인간들에게 익숙하고 편한 수식이라고 하면 컴퓨터를 위한 수식을 표기하는 방법입니다. 기본적인 방식은 연산자를 연산 대상 즉, 피연산자뒤에 쓰는 연산 표기법입니다. 예를 들어 AxB+C의 중위표기식을 AB\*C+라는 방식으로 표기하는 것입니다. 이러한 수식은 우리들은 굉장히 낯설고 계산할 때 어려움을 겪을 수 있지만 컴퓨터입장에서는 식을 추가적으로 변환할 필요없이 수식을 앞에서부터 읽어가면서 스택에 저장하여 계산할 때 용이합니다. 그리고 추가적으로 연산에 우선순위가 모호한 과정에서 괄호를 표기해야하지만 후위 표기법에는 그러한 과정이 발생하지 않습니다.



3. 요구사항 명세

1)사용할 수 있는 연산자는 “+, -, \*, /, %”로 한정되어있다.

2)피연산자는 “0 ~ 9” 사이의 정수만을 허용한다.

3)수식에 괄호”( )”를 사용한다.

4)중위표기법(infix)를 후위표기법(postfix)로 변환하는 과정을 출력하고 결과를 제시한다.

예) infix -> (2+3)\*5

|  |  |  |
| --- | --- | --- |
| symbol | 연산자 stack | postfix 표기법 |
| ( | ( |  |
| 2 | ( | 2 |
| + | (+ | 2 |
| 3 | (+ | 23 |
| ) |  | 23+ |
| \* | \* | 23+ |
| 5 | \* | 23+5 |
| \0 |  | 23+5\* |

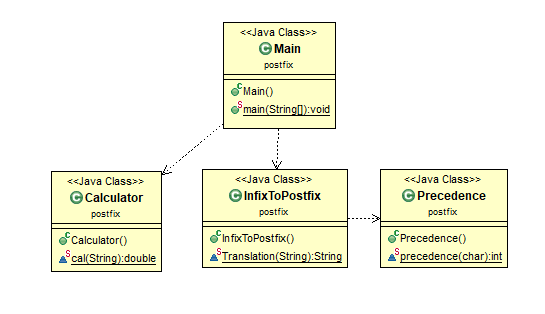
postfix -> 23+5\*

5)후위표기법(postfix)으로 제시된 수식 표기법을 계산하는 과정을 출력하고 결과를 제시한다.

예) 23+5\*

|  |  |  |
| --- | --- | --- |
| symbol | 연 산 | 피연산자 stack |
| 2 |  | 2 |
| 3 |  | 23 |
| + | 2+3 | 5 |
| 5 |  | 55 |
| \* | 5\*5 | 25 |
| \0 |  |  |

계산결과 -> 25

4. 설계

클래스

1.Main

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
| 클래스명 | 구분 | | | |
| Main | 사용변수 | 자료형 | 변수명 | 비고 |
| String | s | 사용자 입력값저장 |
| String | postfix | 후위표기법 변환값 저장 |
| 호출클래스 | 클래스명 | | 비고 |
| BufferedReader |  | 사용자로부터 값 입력받음 |
| BufferedWriter |  | 특정값을 출력 |

전체적인 프로그램의 동작을 담당하는 부분으로 사용자로부터 입력을 받고 그 값을 각각의 클래스를 호출하여 값을 처리하고 최종적으로 처리가 끝나 도출된 값을 출력하여 사용자에게 결과를 보여줍니다. BufferedReader를 통해 사용자로부터 값을 입력받고trim()메소드를 사용하여 사용자의 실수로 인해 수식 사이에 공백이 포함되었을 경우 공백을 제거하여 다시 프로그램을 실행하여 입력하는 경우를 제외해줍니다. 그리고 BufferedWriter를 사용하여 최종 결과값을 출력해줍니다.

2.InfixToPostfix

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
| 클래스명 | 구분 | | | |
| InfixToPostifx | 사용변수 | 자료형 | 변수명 | 비고 |
| String | s | 매개변수 |
| String | postfix | 후위표기법 변환값 저장 |
| Stack | oper | 연산자 저장 |
| 호출클래스 | 클래스명 | | 비고 |
| Precedence | | 연산자 우선순위 비교 |

사용자에게 입력받은 중위표기법의 식이 알맞게 입력되었는지 판단하여 연산자 우선순위에따라 후위표기법으로 변환하는 역할을 수행합니다. 이 과정에서 연산자의 우선순위를 판별하기위해 Precedence클래스를 호출합니다. 만약 제대로된 후위표기법으로 변환이 불가능할 경우 에러를 반환하여 사용자에게 제대로된 값을 입력하길 권고하고 프로그램을 종료시킵니다. Char타입의infix변수를 사용하여 사용자가 입력한 값을 판별하여 조건에 맞는 정수 또는 연산자가 아닐경우, 제대로된 수식이 입력되지 않았을경우 에러메세지를 출력하며 프로그램을 종료시킵니다. oper라는 변수의 Stack구조를 사용하여 연산자를 관리합니다. String타입의 postfix변수를 사용하여 후위표기법으로 변환된 수식을 저장합니다.

3.Precedence

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
| 클래스명 | 구분 | | | |
| Precedence | 사용변수 | 자료형 | 변수명 | 비고 |
| Char | ch | 매개변수 |

연산자를 파라미터로 받아 연산자의 종류에 따라 특정 값을 매겨 연산자의 우선순위를 매깁니다.

|  |  |
| --- | --- |
| 연산자 순위 | 연산자 종류 |
| 1 | (, ) |
| 2 | \*, /, % |
| 3 | +, - |

- 괄호 안에 있는 연산자부터 먼저 수행하며, 우선순위가 높은 연산자부터 수행하되 만약 연산자의 우선순위가 같을 경우 왼쪽부터 즉, 먼저 나온 연산자부터 수행합니다. x+y+z와 같은 식은 x+(y\*z)와 같이 계산됩니다. 이것은 \* 연산자의 우선순위가 + 연산자의 우선순위보다 높기 때문입니다.

4.Calculator

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
| 클래스명 | 구분 | | | |
| Calculator | 사용변수 | 자료형 | 변수명 | 비고 |
| String | postfix | 매개변수 |
| Double | result | 연산후 결과값 저장 |
| Char | token | 후위표기법 값 추출 |
| Stack | test | 정수저장후 연산자 만날 때 정수 반환하여 연산 |

최종적으로 사용자가 입력한 중위표기법이 후위표기법으로 변환이 재대로 이루어졌을 때 수식을 분석해 계산을 하여 결과를 도출해내는 클래스입니다. 후위표기법으로 변환된 식을 앞에서부터 검사해 0 ~ 9 사이의 정수일 경우 스택에 저장하고 연산자일 경우 그동안 스택에 저장해 두었던 정수를 2개를 꺼내어 연산자에 맞춰 값을 계산하여 나온 값을 다시 스택에 저장합니다. 이 과정을 식의 끝에 도달할 때까지 반복하며 끝에 도달하였을 때 스택에 저장되어있는 최종 결과값을 반환해 줍니다.

**알고리즘 설계**

1.수식에서 첫 번째 위치한 값을 추출한다.

2.추출한 값이 피연산자이면 그래도 출력후 postfix변수에 누적해준다

3.추출한 값이 “(” 이면 스택에 push하여 저장한후 다음 위치에 값을 추출한다.

4.추출한 값이 연산자라면 스택에 저장한다. 이때 스택에 값이 존재할 경우 스택의 top에 위치한 값과 우선순위가 어떤 것이 높은지 비교한다.

4-1.만약 추출한 연산자가 기존 top의 연산자보다 우선순위가 높으면 그대로 스택에 저장한다.

4-2.그렇지않고 top의 연산자가 우선순위가 높은경우 top의 연산자를 pop 하여 출력하고 postfix값에 누적해준다.

5.추출한 값이 “)” 라면 저장해 두었던 “(”이 스택에 나올때까지 스택에 저장되어있는 값들을 pop하여 postfix에 누적한다.

**입출력 설계**

**1) 괄호가 없는 경우**

1.1 우선순위가 같을 경우

1.2 n/0으로 나누는 경우 (에러발생)

1.3 9보다 큰 수가 입력 될 시

1.4 9보다 작은 수가 입력 될 시

1.5 연산자가 연달아 나오는 경우 ex) ++

1.6 피연산자가 연달아 나오는 경우 ex) 33+

1.7 Null인 경우

1.8 연산자 뒤에 피연산자가 없는 경우 ex) 2+3\*

1.9) 지정 연산자 외에 연산자가 나올 경우 ex) >>, <<

1.0) -9~9 , 5개 연산자 외에 다른 문자가 오는 경우

**2) 괄호가 있는 경우**

2.1 우선순위가 같을 경우

2.2 n/0으로 나누는 경우 (에러발생)

2.3 9보다 큰 수가 입력 될 시

2.4 -9보다 작은 수가 입력 될 시

2.5 연산자가 연달아 나오는 경우 ex) ++

2.6 피연산자가 연달아 나오는 경우 ex) 33+

2.7 Null인 경우

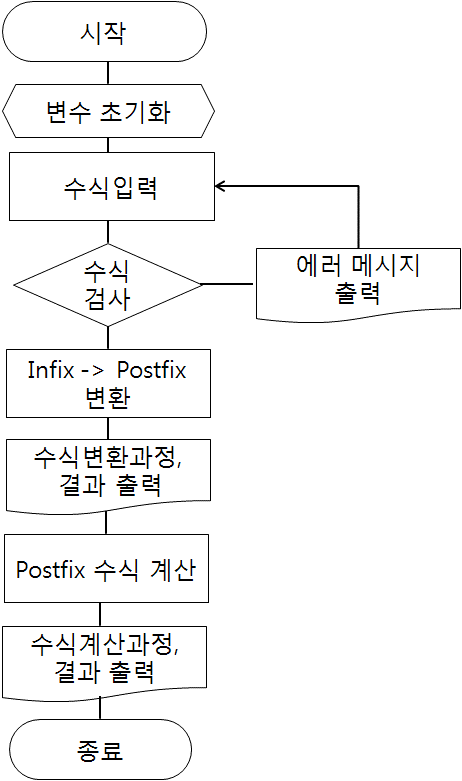
2.8 ‘(’없이 ‘)’만 나오는 경우

2.9) ‘)’가 없는 경우

2. 연산자 뒤에 피연산자가 없는 경우 ex) 2+3\*

2. 지정 연산자 외에 연산자가 나올 경우 ex) >>, <<

2. 0~9 , 5개 연산자 외에 다른 문자가 오는 경우

2. -()인 경우 -1\*를 해준다

5. Program 작성

**package** postfix;

**import** java.io.\*;

**public** **class** Main {

**public** **static** **void** main(String[] args) **throws** IOException {

BufferedReader br = **new** BufferedReader(**new** InputStreamReader(System.***in***));

BufferedWriter bw = **new** BufferedWriter(**new** OutputStreamWriter(System.***out***));

String s = br.readLine().trim();

String postfix = InfixToPostfix.*Translation*(s);

bw.write("계산 결과 : "+String.*valueOf*(Calculator.*cal*(postfix)));

bw.flush();

bw.close();

br.close();

}

}

**package** postfix;

**public** **class** Precedence {

**static** **int** precedence(**char** ch) {

**if** (ch == '(') **return** 2;

**if** (ch == '+' || ch == '-') **return** 1;

**else** **return** 0;

}

}

**package** postfix;

**import** java.util.Stack;

**public** **class** Calculator {

**static** **double** cal(String postfix) {

**double** result = 0;

Stack<Double> test = **new** Stack<>();

**for** (**char** token : postfix.toCharArray()) {

**if** ('1' <= token && token <= '9') {

token -= 48;

test.push((**double**) token);

} **else** {

**if** (token == '\*') {

**double** x = test.pop();

**double** y = test.pop();

result = y \* x;

test.push(result);

} **else** **if** (token == '/') {

**double** x = test.pop();

**double** y = test.pop();

result = y / x;

test.push(result);

} **else** **if** (token == '%') {

**double** x = test.pop();

**double** y = test.pop();

result = y % x;

test.push(result);

}**else** **if** (token == '-') {

**double** x = test.pop();

**double** y = test.pop();

result = y - x;

test.push(result);

} **else** **if** (token == '+') {

**double** x = test.pop();

**double** y = test.pop();

result = y + x;

test.push(result);

}

}

System.***out***.println("스택상태 : "+ test + ", 토큰상태 : "+ token + ", 계산결과 : "+ result);

}

**return** result;

}

}

**package** postfix;

**import** java.util.Stack;

**public** **class** InfixToPostfix {

**static** String Translation(String s){

String postfix = "";

Stack<Character> oper = **new** Stack<>();

**for**(**char** infix : s.toCharArray()){

**if**('1' <= infix && infix <= '9'){

postfix += infix;

}

**else** **if**(infix == '('){

oper.push(infix);

}

**else** **if**(infix == ')'){

**while**(!oper.isEmpty()){

**if**(oper.peek() == '('){

oper.pop();

**break**;

}

postfix += oper.pop();

}

}

**else**{

**while**(!oper.isEmpty() && Precedence.*precedence*(oper.peek()) <= Precedence.*precedence*(infix)){

postfix += oper.pop();

}

oper.push(infix);

}

System.***out***.println("전위표기식 -> 후위 표기식 : "+postfix);

}

**while** (!oper.isEmpty()) {

postfix += oper.pop();

System.***out***.println("전위표기식 -> 후위 표기식 : "+postfix);

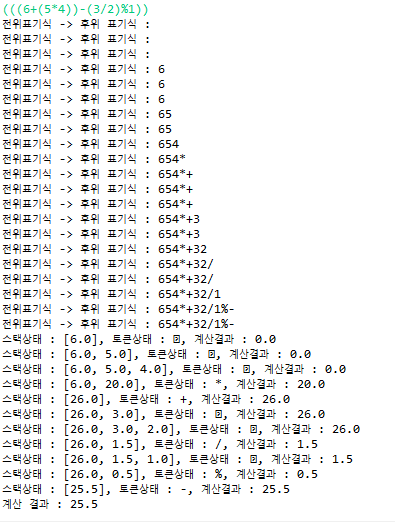
}

**return** postfix;

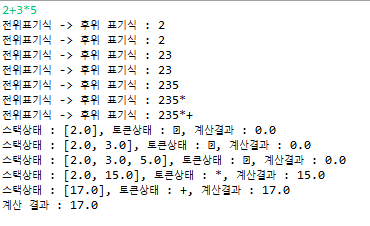
}

}

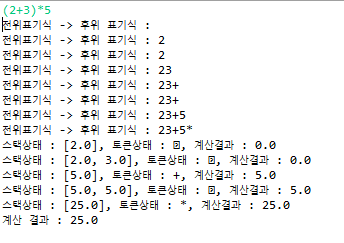
6. 출력 결과



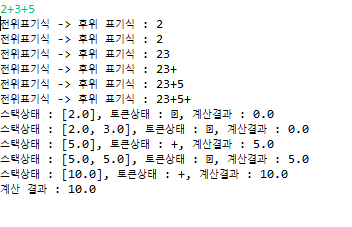
괄호가 없는 경우



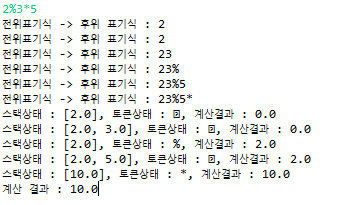
괄호가 있는경우



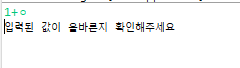
복수의 연산자들이 우선순위가 같을 경우



복수의 연산자들이 우선순위가 다른 경우



피연산자의 값이 올바르지 않을 경우



연산자가 연속적으로 입력될 경우



괄호가 잘못되었을 경우



7. 참고 문헌

|  |  |  |
| --- | --- | --- |
| 자료이름 | 저자 | 출판사 |
| 자료구조 | 윤경섭 | 정익사 |
| 네이버, 구글 등 각종 포털사이트 검색엔진 활용 | | |
| 자료구조 강의시간 수업자료 pptx 참고 | | |

8. 검토

과제를 처음 들었을 때 스택이라는 구조를 교수님 수업을 들으면서 전부 이해했다고 생각하여 굉장히 쉽게 여겨 기간이 3주인 것은 과제에 부담 갖지 말고 여유롭게 진행하라는 뜻이라고 생각했습니다. 그러나 막상 과제를 시작하니 생각해야 할 부분도 굉장히 많고 수업 시간에 이해하는 것과 스스로 직접 사용해보고 설계하는 것은 완전히 다른 수준의 문제라는 것을 깨달았습니다. 그리고 계산기도 굉장히 친숙한 프로그램이라 간단할 거라고 생각했지만 연산자의 우선 수위, 괄호 등 신경 써야 할 부분이 너무 많았습니다. 코드를 작성했다가 다시 갈아엎기를 여러 번 반복하였고 “연산자가 왜 제대로 출력이 되지 않을까”, “중간에 괄호를 어떤 식으로 처리할까”, “스택에서 에러는 어떻게 처리하지”등 여러 가지 고민을 많이 했습니다. 후위 표기법으로 변환하는 코드를 다 완성했다고 생각하여 코드 검증을 위해서 백준이라는 사이트(<https://www.acmicpc.net/problem/1918>)에서 같은 예제를 발견하여 코드를 검증하였는데 미처 고려하지 못한 예시들을 만나 다시 좌절을 겪고 해가 뜰 때까지 코드를 수정해보는 경험도 했습니다. 이번 과제를 수행하면서 구글이나 네이버등을 통해서 정말 많은 자료들을 찾아보고 강의 교안도 계속 읽어보고, 강의도 다시 듣기를 반복하였습니다. 자료구조시간에 배우는 지식들이 얼마나 중요한지 깨닫게 되었고 중위표기법과 후위표기법, 스택의 정확한 개념과 구조 그리고 활용법, 후위표기법에서 스택이 쓰이는 이유 등 과 같은 것들에 대해 충분한 이해를 갖게 되었습니다.