프로그래밍과제 #2

보고서

프로그래밍언어(나)

글로벌미디어학부

20182705 빙기범

[-] 처리 구현함.

1. 전역 변수 선언 및 초기화

- Java에서는 class의 변수

- token: 현재 검사하고 있는 입력의 lexem

- num: <number>와 <digit>에서 얻은 수 저장

- input(input_data): 입력 받은 수식 저장

- input_count: input에서 처리를 받지 못한 원소를 차례대로 가리킴.

- paren: ')'로 닫지 못한 '('의 수

C++

```cpp
char token;
int num = 0;
string input;
int input_count = 0;
int paren = 0;
```

Java

```java
static char token;
static int num = 0;
static String input;
static int input_count = 0;
static int paren = 0;
```

Python

```python
token = 0
num = 0
input_data = 0
input_count = 0
paren = 0
```

2. error 함수

- 문법에 맞지 않는 수식이 입력된 경우 'Syntax error!!'를 출력하고 프로그램 종료

C++

```cpp
void error(){
    cout << "syntax error!!" << endl;
    exit(1);
}
```

Java

```java
static void error(){
    System.out.println("syntax error!!")
    System.exit( status: 1);
}
```

Python

```python
def error():
    print("syntax error!!\n")
    exit(1)
```

3. lex() 함수 구현

- 만약 token에 숫자가 들어오면 자리수를 하나 올리고, 일의 자리 수에 숫자를 추가하여, 피연산자를 조합하여 integer로 만듦.(C++, Java에서는 문자열을 integer로 바꾸기 위해 48을 빼줌)

- token에 연산자가 들어오면 피연산자를 0으로 초기화.

- terminal이 들어오면 다음 token에 다음 입력 요소를 집어넣음.

- 띄어쓰기 무시함.

- 입력한 피연산자의 최상위 자리의 수가 0일 경우 에러.

C++

```cpp
void lex(){
    if(isdigit(token)){
        if(num==0 && token=='0')
            error();
        num = num * 10 + token - 48;
    }
    else if(token=='*' || token=='/' || token=='+' || token=='-')
        num = 0;

    token = input[input_count++];

    while(isspace(token)){
        token = input[input_count++];
    }
}
```

Java

```java
static void lex(){
    if(Character.isDigit(token)) {
        if (num == 0 && token == '0')
            error();
        num = num * 10 + token - 48;
    }
    else if(token=='*' || token=='/' || token=='+' || token=='-')
        num = 0;

    if(input.length()>input_count) {
        token = input.charAt(input_count++);
        while (token == ' ') {
            if (input.length() > input_count)
                token = input.charAt(input_count++);
        }
    }
}
```

Python

```python
def lex():
    global token
    global input_data
    global input_count
    global num
    if token.isdigit():
        if num == 0 and token == '0':
            error()
        num = num * 10 + int(token)
    elif token=='*' or token=='/' or token=='+' or token=='-':
        num = 0

    if(len(input_data)>input_count):
        token = input_data[input_count]
        input_count += 1
        while token == ' ':
            if (len(input_data) > input_count):
                token = input_data[input_count]
                input_count += 1
```

4. expr() 함수 구현

- <expr> → <term> {+ <term>| - <term>}

- 중괄호는 while로 구현

- terminal은 lex()를 실행.

C++

```cpp
float expr() {
    float value = term();
    while(token == '+' || token == '-'){
        switch(token) {
        case '+':
            lex();
            value += term();
            break;
        case '-':
            lex();
            value -= term();
            break;
        default:
            error();
        }
    }
    return value;
}
```

Java

```java
static float expr() {
    float value = term();
    while(token == '+' || token == '-'){
        switch(token) {
            case '+':
                lex();
                value += term();
                break;
            case '-':
                lex();
                value -= term();
                break;
            default:
                error();
        }
    }
    return value;
}
```

Python

```python
def expr():
    global token
    value = term()
    while token == '+' or token == '-':
        if token == '+':
            lex()
            value += term()
        elif token == '-':
            lex()
            value -= term()
        else:
            error()
    return value
```

5. term() 함수 구현

- <term>→ <factor> {*<factor>| /<factor> }

- 중괄호는 while로 구현

- terminal은 lex()를 실행.

C++

```cpp
float term(){
    float value = factor();
    while(token == '*' || token == '/'){
        switch(token) {
        case '*':
            lex();
            value *= factor();
            break;
        case '/':
            lex();
            value /= factor();
            break;
        default:
            error();
        }
    }
    return value;
}
```

Java

```java
static float term(){
    float value = factor();
    while(token == '*' || token == '/'){
        switch(token) {
            case '*':
                lex();
                value *= factor();
                break;
            case '/':
                lex();
                value /= factor();
                break;
            default:
                error();
        }
    }
    return value;
}
```

Python

```python
def term():
    global token
    value = factor()
    while token == '*' or token == '/':
        if token == '*':
            lex()
            value *= factor()
        elif token == '/':
            lex()
            value /= factor()
        else:
            error()
    return value
```

6. factor() 함수 구현

- <factor> → [-] (<numbers> | (<expr>))

- terminal은 lex()를 실행.

- [-]가 있을 경우, minus값을 -1로 하고, 최종 value에 곱해 return 함.

C++

```cpp
float factor(){
    float value;
    float minus = 1;

    if(token == '-'){
        lex();
        minus = -1;
    }
    if(token == '('){
        paren++;
        lex();
        value = expr();
        if(token == ')'){
            paren--;
            lex();
        }
        else
            error();
    }
    else
        value = number();

    return minus * value;
}
```

Java

```java
static float factor(){
    float value = 0;
    float minus = 1;
    if(token == '-'){
        lex();
        minus = -1;
    }
    if(token == '('){
        paren++;
        lex();
        value = expr();
        if(token == ')') {
            paren--;
            lex();
        }
        else
            error();
    }
    else
        value = number();

    return minus * value;
}
```

Python

```python
def factor():
    global token
    global paren
    minus = 1

    if token == '-':
        lex()
        minus = -1
    if token == '(':
        paren += 1
        lex()
        value = expr()
        if token == ')':
            paren -= 1
            lex()
        else:
            error()
    else:
        value = number()
    return minus * value
```

7. number() 함수 구현

- digit() 이후 token값이 숫자나 연산자나 ')', 널 값이 아니면 error() 실행

- digit() 이후 token값이 이전에 '(' 없이 ')'이면 error() 실행

- digit() 이후 token값이 숫자(0~9)가 오면 digit() 실행

C++

```cpp
int number(){
    digit();
    if(isdigit(token) || token=='+' || token=='-' || token=='*' || token=='/' || token==')' || token=='\0'){
        if(token==')' && paren<1)
            error();
        while(isdigit(token))
            digit();
    }
    else
        error();

    return num;
}
```

Java

```java
static int number(){
    digit();
    if(Character.isDigit(token) || token=='+' || token=='-' || token=='*' || token=='/' || token==')' || token=='\0') {
        if (token == ')' && paren < 1)
            error();
        while (Character.isDigit(token))
            digit();
    }
    else
        error();

    return num;
}
```

Python

```python
def number():
    global token
    global num
    global paren
    digit()
    if token.isdigit() or token=='+' or token=='-' or token=='*' or token=='/' or token==')' or token=='\0':
        if token==')' and paren<1:
            error()
        while token.isdigit():
            digit()
    else:
        error()
    return num
```

8. digit() 함수 구현

- <digit>→ 0 | 1 | 2 | 3 | 4 | 5 | 6 | 7 | 8 | 9

- token이 0-9값이면 lex() 실행, 이외의 값이 오면 에러 출력

C++

```cpp
int digit(){
    if(isdigit(token))
        lex();
    else
        error();

    return num;
}
```

Java

```java
static int digit(){
    if(Character.isDigit(token))
        lex();
    else
        error();

    return num;
}
```

Python

```python
def digit():
    global token
    global num
    if token.isdigit():
        lex()
    else:
        error()
```

9. main 함수 구현

- 엔터키가 입력되면 하나의 수식이 입력된 것으로 처리함.

- 입력된 수식을 input(input_data)에 저장함.

- input의 가장 첫번째 값을 token에 넣음.

- result에 RD parser 및 계산 결과를 저장 후 출력 후, 다음 수식을 입력 받음.

C++

```cpp
int main(void) {
    while(true){
        getline(cin, input);
        token = input[input_count++];
        float result = expr();
        cout << result << endl;

        token = 0;
        num = 0;
        input_count = 0;
    }
    return 0;
}
```

Java

```java
public static void main(String[] args) {
    while(true){
        Scanner sc = new Scanner(System.in);
        input = sc.nextLine();
        token = input.charAt(input_count++);
        input = input + '\0';
        float result = expr();
        if(result%1.0==0)
            System.out.println((int)result);
        else
            System.out.println(result);
        token = 0;
        num = 0;
        input_count = 0;
    }
}
```

Python

```python
if __name__ == '__main__':
    while True:
        input_data = input()
        input_data = input_data + '\0'
        token = input_data[input_count]
        input_count += 1
        result = expr()
        print(result)

        token = 0
        num = 0
        input_data = 0
        input_count = 0
```

10. 실행 결과

C++

```
4531 + ( 32 * 5 ) + (86+5)
4782
(69*8)          +911
1463
(74          +88)
162
((32*4)
syntax error!!
```

Java

```
4531 + ( 32 * 5 ) + (86+5)
4782
(69*8)          +911
1463
(74          +88)
162
((32*4)
syntax error!!
```

Python

```
4531 + ( 32 * 5 ) + (86+5)
4782
(69*8)          +911
1463
(74          +88)
162
((32*4)
syntax error!!
```