과제 – 1 (수식의 값 계산)

컴퓨터학부 20160548 이승현

1. 문제 해결 방법

이 과제의 목표는 곱셈, 덧셈 기호와 정수, 실수로 이루어진 수식을 입력으로 받고 그 수식의 결과 값을 구하는 프로그램을 작성하는 것이다. 교재에 있는 미완성의 소스코드를 이용하면 되기 때문에 실제로 해결해야 하는 문제는 get_token() 함수를 구현하는 것 한 가지뿐이다.

get_token() 함수는 한 문자씩 읽어 토큰으로 분류하고, 어떤 종류의 토큰인지 알아내 전역 변수 token에 저장한다. 그리고 해당 토큰이 숫자인 경우에는 숫자 문자열을 정수형 또는 실수형으로 변환하여 저장해야 한다.

위 기능을 다음과 같이 get_token() 내에서 구현하였다. 일단 한 문자를 읽는다. ' '과 '₩t'이 들어왔으면 의미 없는 공백이므로 바로 다음 문자를 읽는다. 읽은 문자를 토큰으로 분류하는 기능은 if, else if, else 문으로 구현하였다. '+', '*', '(', ')', '₩n' 문자가 들어오면 token에 해당되는 enum 값을 넣고 함수를 종료한다. 여기에서 '₩n' 문자가 입력되면 입력의 끝 이므로 수식의 끝이라고 생각하고 token에 END를 넣는다.

읽은 문자가 숫자라면 숫자 문자열을 실수형으로 변환하는 작업을 시작한다. 다음 문자를 계속해서 읽어 들여 연속된 숫자들을 number_string 배열에 넣는다. 이 때 실수도 입력될 수 있으므로 '.' 문자도 배열에 넣는다. 계속 다음 문자를 읽다가 숫자나 '.'이 아닌 다른 문자가 들어오면 숫자 입력이 끝난 것으로 생각하고 읽었던 문자를 다시 ungetc()로 입력 버퍼에 돌려놓는다. 그 뒤 number_string에 들어있던 숫자 문자열을 atof()를 이용하여 실수형으로 변환하고, 그 값을 전역 변수인 num에 저장해 수식의 값 계산에 이용할 수 있도록 한다. 이렇게 숫자를 읽는 과정에서 '.' 문자가 두 번 들어왔다면 잘못된 숫자 형식이므로 error() 함수를 호출하도록 했다.

또한 정수형 수식은 정수 값으로 출력하고, 실수형이나 혼합형은 실수 값으로 출력하는 기능은 전역변수 is_float을 이용하여 구현하였다. is_float 변수를 0으로 초기화 해 놓은 뒤, get_token()에서 숫자로 된 문자열을 실수형으로 변환하는 과정에서 '.' 문자가 입력됐다면 is_float의 값을 1로 바꾼다. 이렇게 해서 수식이 정수로만 이루어져 있다면 is_float의 값은 0이 되고, 실수가 하나라도 포함되어 있다면 is_float의 값은 1이 된다. 수식을 모두 계산한 뒤 결과를 출력할 때 is_float이 0이라면 결과 값을 int형으로 변환하여 정수로 출력, 그렇지 않다면 실수형 그대로 출력하도록 했다.

수식에 오류가 있는 경우 오류가 발생한 위치를 표시하는 기능도 구현하였다. 이 기능은 전역변수인 character_index를 이용하여 구현하였다. 입력 버퍼에서 문자를 하나씩 읽을 때마다 character_index를 1씩 증가시켜 수식의 오류가 발생한 경우 그 값을 이용하여 오류가 발생한 위치를 표시하고, 오류 내용을 출력하도록 하였다.

위 내용을 구현한 소스코드는 보고서의 '3.소스코드'에 첨부하였다.

2. 실행 결과

```
shlee@shlee-virtual-machine:~/workspace/ssucompiler/project1$ ./a.out
3+4*12
51
```

⇒ 정수형만 있는 수식

```
shlee@shlee-virtual-machine:~/workspace/ssucompiler/project1$ ./a.out
3 + 4  *12
51
```

⇒ 공백문자가 포함된 수식

```
shlee@shlee-virtual-machine:~/workspace/ssucompiler/project1$ ./a.out
3.0 + 4 * 12
51.000000
```

⇒ 실수형이 섞여 있는 혼합형 수식

```
shlee@shlee-virtual-machine:~/workspace/ssucompiler/project1$ ./a.out
6.11 * 2.14
13.075401
```

⇒ 실수형 수식

```
shlee@shlee-virtual-machine:~/workspace/ssucompiler/project1$ ./a.out
1*2+2*3+3*4+4*5+5*6
70
```

⇒ 긴 수식

```
shlee@shlee-virtual-machine:~/workspace/ssucompiler/project1$ ./a.out
(3+2)*10
50
```

⇒ 괄호가 있는 수식

```
shlee@shlee-virtual-machine:~/workspace/ssucompiler/project1$ ./a.out
1+2*
    ^
There must be a number or a left parenthesis.
```

⇒ 연산자가 잘못 사용된 수식

```
shlee@shlee-virtual-machine:~/workspace/ssucompiler/project1$ ./a.out
3++2
  ^
There must be a number or a left parenthesis.
```

⇒ 연산자가 잘못 사용된 수식

```
shlee@shlee-virtual-machine:~/workspace/ssucompiler/project1$ ./a.out
(3+2*10
       ^
Missing right parenthesis.
```

⇒ 닫는 괄호가 없는 수식

```
shlee@shlee-virtual-machine:~/workspace/ssucompiler/project1$ ./a.out
2*(22+33)3
         ^
It must be empty at '^'
```

⇒ 제대로 끝나지 않은 수식

```
shlee@shlee-virtual-machine:~/workspace/ssucompiler/project1$ ./a.out
22.45+12.3.4*2
          ^
Invalid number format.
```

⇒ 잘못된 형식의 수가 있는 수식

3. 소스코드

```
#include <stdio.h>
#include <stdlib.h>
#include <ctype.h>

#define NUMBER_STRING_LENGTH 100
int is_float = 0;
int character_index = 0;
float num;
enum {null, NUMBER, PLUS, STAR, LPAREN, RPAREN, END} token;

void get_token();
float expression();
float term();
float factor();
void error(int i);

int main() {
	float result;
	get_token();
	result = expression();
	if (token != END)
		error(3);
	else
		if (is_float) // 실수형이나 혼합형인 경우
			printf("%f₩n", result); // 실수로 결과 출력
		else // 정수형인 경우
			printf("%d₩n", (int)result); // 정수로 결과 출력
```

```
}

void get_token(){
	// next token --> token
	// number value --> num
	char next_character = getchar();
	++character_index;
	while (next_character == ' ' || next_character == '₩t') {
		next_character = getchar();
		++character_index;
	}
	if (next_character == '+') {
		token = PLUS;
	} else if (next_character == '*') {
		token = STAR;
	} else if (next_character == '(') {
		token = LPAREN;
	} else if (next_character == ')') {
		token = RPAREN;
	} else if (next_character == '₩n') {
		token = END;
	} else if (isdigit(next_character)){ // 숫자인 경우
		char number_string[NUMBER_STRING_LENGTH]; // float 데이터 형으로 바꾸기 전에 문자열 형태로 숫자를 저장해 둘 배열
		int i = 0;
		int dot_count = 0;
		token = NUMBER;
		number_string[i++] = next_character;
		while (1) { // 여러 자리로 된 숫자를 읽는다
```

```
                next_character = getchar(); // 다시 한 문자를 읽는다
                ++character_index;
                if (isdigit(next_character) || next_character == '.') {
                        number_string[i++] = next_character;
                        if (next_character == '.') {
                                if (++dot_count > 1) error(4); // 한 숫자 내에서 '.'이 2개 이상 나오면 에러
                                is_float = 1;
                        }
                } else { // 읽은 문자가 실수를 이루는 문자가 아니라면
                        ungetc(next_character, stdin); // 읽었던 문자를 다시 입력 버퍼에 돌려놓는다
                        --character_index;
                        break; // 숫자 모두 읽은 것 이므로 반복 종료
                }
        }
        number_string[i] = '₩0';
        num = atof(number_string); // 문자열로 된 숫자를 float형으로 변환함
    } else {
        token = null;
    }
    return;
}

float expression() {
    float result;
    result = term();
    while (token == PLUS) {
        get_token();
        result += term();
```

```
        }
        return result;
}

float term() {
        float result;
        result = factor();
        while (token == STAR) {
                get_token();
                result *= factor();
        }
        return result;
}

float factor() {
        float result;
        if (token == NUMBER) {
                result = num;
                get_token();
        } else if (token == LPAREN) {
                get_token();
                result = expression();
                if (token == RPAREN)
                        get_token();
                else
                        error(2);
        } else {
                error(1);
```

```
	}
	return result;
}

void error(int i) {
	while(character_index-- > 1) printf(" ");
	printf("^₩n");
	switch(i) {
		case 1:
			printf("There must be a number or a left parenthesis.₩n");
			break;
		case 2:
			printf("Missing right parenthesis.₩n");
			break;
		case 3:
			printf("It must be empty at '^'₩n");
			break;
		case 4:
			printf("Invalid number format.₩n");
			break;
	}
	exit(1);
}
```