과제 – 2 (LR 파서)

컴퓨터학부 20160548 이승현

1. 문제 해결 방법

이 과제의 목표는 곱셈, 덧셈 기호와 정수, 실수로 이루어진 수식을 입력으로 받아 LR 파서를 이용하여 그 수식의 문법이 제대로 되어있는지를 판단하고, 수식의 결과 값을 구하는 프로그램을 작성하는 것이다.

이를 구현하기 위하여 먼저 실수형과 정수형을 모두 저장할 수 있는 구조체 TT를 아래와 같이 정의하였다. 이 구조체는 공용체로 되어있는 멤버 변수 V를 갖고 있다. V에 정수형 또는 실수형 값을 저장한다. V에 저장된 값의 형식은 TT의 또 다른 멤버변수인 T를 이용해 알 수 있다. T에 INT 또는 FLOAT을 넣어 어떤 형식의 값을 저장하고 있는지 표시한다. 그리고 정수형과 실수형 사이에 연산이 일어나는 경우에 해당 위치에 경고 메시지를 출력하는 기능을 위해 token_start_index라는 멤버 변수를 추가하였다. 이 변수에는 수식 내에서 해당 피연산자가 시작되는 위치를 저장한다.

```
typedef enum {INT, FLOAT} type;
typedef union {
        int dValue; // 정수는 여기에 저장
        float fValue; // 실수는 여기에 저장
} value;
typedef struct {
        type T; // 피연산자의 형식 저장
        value V; // 피연산자의 값 저장
        int token_start_index; // 수식에서 피연산자의 시작 위치 저장
} TT;
```

에러, 경고메시지 출력을 위하여 int형 배열 ew_list, int형 변수 ew_last와 conversion_warning_flag를 추가하였다. ew_list는 에러가 일어난 위치나 정수와 실수간 연산이 일어난 곳의 위치를 저장하는 배열이다. ew_last는 ew_list배열의 마지막 원소의 인덱스를 갖고 있는 변수이다. conversion_warning_flag는 정수와 실수 사이에 연산이 일어난 경우에는 1, 그렇지 않은 경우에는 0을 값으로 갖는 플래그 변수이다.

수식에서 실수 값을 읽기 위해 주어진 코드의 yylex()함수를 수정하였다. 해당 함수는 입력된 수식을 분석하여 토큰을 알아내고, 토큰이 숫자인 경우에는 그 값을 저장한다. 교재에 있던 기존의 코드는 정수 값만 변환할 수 있었다. 이를 수정하여 실수 값 또한 변환하여 저장할 수 있도록 했다. 숫자들을 읽을 때 '.' 문자가 있으면 실수, 그렇지 않으면 정수라 판단하고, 실수인 경우 atof()를 사용하여 변환하고, 정수인 경우 atoi()를 이용해 변환하여 저장하도록 했다. 경고메시지 출력에 사용할 숫자의 위치도 시작 위치도 저장한다.

구조체 TT를 이용한 수식 계산을 위해 reduce() 함수를 수정하였다. 두 피연산자의 자료형이 모두 int형이면 그냥 연

산을 한 뒤 다시 정수형으로 저장한다. 그 외의 경우에는 결과가 무조건 float형이 된다. 이 때에는 두 피연산자를 꺼내서 연산을 한 뒤 실수형으로 저장한다. 두 피연산자의 자료형이 서로 다른 경우에는 해당 연산이 일어나는 위치에 경고 메시지를 출력하기 위해 뒤쪽 피연산자의 시작 위치를 ew_list에 추가하고, conversion_warning_flag에 1을 넣어 실수와 정수 사이에 연산이 일어났음을 표시한다.

프로그램의 마지막 부분에서는 계산한 결과를 자료형에 맞게 출력하고, 정수와 실수 사이에 연산이 일어났다면 해당 위치를 표시하고, 경고 메시지를 출력하도록 했다.

2. 실행 결과

```
shlee@shlee-virtual-machine:~/workspace/ssucompiler/project2$ ./a.out
3+4*12

51
```

⇒ 정수형 수식

```
shlee@shlee-virtual-machine:~/workspace/ssucompiler/project2$ ./a.out
3 + 4    * 12

51
```

⇒ 공백문자가 포함된 수식

```
shlee@shlee-virtual-machine:~/workspace/ssucompiler/project2$ ./a.out
3.0 + 4 * 12
      ^
WARNING: conversion from 'int' to 'float', possible loss of data
51.000000
```

⇒ 실수형이 섞여 있는 혼합형 수식

```
shlee@shlee-virtual-machine:~/workspace/ssucompiler/project2$ ./a.out
6.11 * 2.134

13.038741
```

⇒ 실수형 수식

```
shlee@shlee-virtual-machine:~/workspace/ssucompiler/project2$ ./a.out
1*2+2*3+3*4+4*5+5*6

70
```

⇒ 긴 수식

```
shlee@shlee-virtual-machine:~/workspace/ssucompiler/project2$ ./a.out
3 + 4 + 2.0 + 3 + 1.0 + 3 + 3 + 2.0
        ^     ^         ^   ^
WARNING: conversion from 'int' to 'float', possible loss of data
21.000000
```

⇒ 긴 혼합형 수식

```
shlee@shlee-virtual-machine:~/workspace/ssucompiler/project2$ ./a.out
(3+2)*10

50
```

⇒ 괄호가 있는 수식

```
shlee@shlee-virtual-machine:~/workspace/ssucompiler/project2$ ./a.out
1+2*
     ^
syntax error
```

⇒ 연산자가 잘못 사용된 수식

```
shlee@shlee-virtual-machine:~/workspace/ssucompiler/project2$ ./a.out
3++2
    ^
syntax error
```

⇒ 연산자가 잘못 사용된 수식

```
shlee@shlee-virtual-machine:~/workspace/ssucompiler/project2$ ./a.out
2*(22+33
        ^
syntax error
```

⇒ 닫는 괄호가 없는 수식

```
shlee@shlee-virtual-machine:~/workspace/ssucompiler/project2$ ./a.out
2*(22+33)3.0
           ^
syntax error
```

⇒ 제대로 끝나지 않은 수식

```
shlee@shlee-virtual-machine:~/workspace/ssucompiler/project2$ ./a.out
22.45+12.3.4*2
          ^
illegal token
```

⇒ 잘못된 형식의 수가 있는 수식

3. 소스코드

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

#define NUMBER 256
#define PLUS 257
#define STAR 258
#define LPAREN 259
#define RPAREN 260
#define END 261
#define EXPRESSION 0
#define TERM 1
#define FACTOR 2
#define ACC 999

int action[12][6] = {
        {5, 0, 0, 4, 0, 0}, {0, 6, 0, 0, 0, ACC}, {0, -2, 7, 0, -2, -2},
        {0, -4, -4, 0, -4, -4}, {5, 0, 0, 4, 0, 0}, {0, -6, -6, 0, -6, -6},
        {5, 0, 0, 4, 0, 0}, {5, 0, 0, 4, 0, 0}, {0, 6, 0, 0, 11, 0},
        {0, -1, 7, 0, -1, -1}, {0, -3, -3, 0, -3, -3}, {0, -5, -5, 0, -5, -5} };

int go_to[12][3] = {
        {1, 2, 3}, {0, 0, 0}, {0, 0, 0}, {0, 0, 0}, {8, 2, 3}, {0, 0, 0},
        {0, 9, 3}, {0, 0, 10}, {0, 0, 0}, {0, 0, 0}, {0, 0, 0}, {0, 0, 0} };

int prod_left[7] = {0, EXPRESSION, EXPRESSION, TERM, TERM, FACTOR, FACTOR};
int prod_length[7] = {0, 3, 1, 3, 1, 3, 1};
```

```
// 피연산자 저장에 사용하는 enum, struct
typedef enum{INT, FLOAT}type;
typedef union {
	int dValue; // 정수는 여기에 저장
	float fValue; // 실수는 여기에 저장
} value;
typedef struct {
	type T; // 피연산자의 형식 저장
	value V; // 피연산자의 값 저장
	int token_start_index; // 수식에서 피연산자의 시작 위치 저장
} TT;

// 에러 처리 관련
int command_index = 0; // 입력된 수식의 인덱스
int ew_list[1000]; // 에러, 경고 표시할 부분의 인덱스를 저장할 배열
int ew_last = -1; // ew_list의 마지막 원소의 인덱스
int conversion_warning_flag = 0; // 정수형 실수형이 혼합된 식일 경우 경고해주기 위한 플래그
void push_to_ew_list(int); // ew_list에 원소 추가하는 함수
void print_ew_points(); // 에러, 경고 위치 표시하는 함수
int compare_int(const void *_a, const void *_b); // qsort에 사용하는 정수 크기 비교 함수

// 원래 있던 변수들
int stack[1000];
TT values[1000];
int top = -1;
int sym;
char yytext[32];
TT yylval;
```

```
// 원래 있던 함수들
void push(int);
void shift(int);
void reduce(int);
void yyerror();
TT yyparse();
int yylex();
void lex_error();

int main() {
	TT result;
	result = yyparse();
	print_ew_points();
	if (result.T == INT) {
		printf("%d₩n", result.V.dValue);
	} else {
		printf("%f₩n", (float)result.V.fValue);
	}

	return 0;
}

TT yyparse() {
	int i;
	stack[++top] = 0; // initial state
	sym = yylex();
	do {
```

```
                i = action[stack[top]][sym - 256]; // get relation
                if (i == ACC) {
                        //printf("success!₩n");
                } else if (i > 0) {
                        shift(i);
                } else if (i < 0) {
                        reduce(-i);
                } else {
                        yyerror();
                }
        } while (i != ACC);

        return values[top];
}

void push(int i) {
        top++;
        stack[top] = i;
}

void shift(int i) {
        push(i);
        values[top] = yylval;
        sym = yylex();
}

void reduce(int i) {
        int old_top;
```

```
	top -= prod_length[i];
	old_top = top;
	push(go_to[stack[old_top]][prod_left[i]]);

	// value
	switch(i) {
		case 1:
			if (values[old_top + 1].T != values[old_top + 3].T) { // 두 숫자의 형식이 다른 경우
				// warning
				push_to_ew_list(values[old_top + 3].token_start_index); // 경고 표시할 위치 저장
				conversion_warning_flag = 1; // 플래그 1로 바꿔 정수형->실수형 변환이 일어났음을 표시함
			}

			if (values[old_top + 1].T == INT && values[old_top + 3].T == INT) { // 두 숫자 모두 정수 일때 - 연산 결과가 정수인 경우
				values[top].V.dValue = values[old_top + 1].V.dValue + values[old_top + 3].V.dValue; // 두 수를 더해서 저장
				values[top].T = INT; // 연산 결과가 정수형임을 저장
			} else { // 연산 결과가 실수인 경우
				float value1, value2;
				// 첫번째 피연산자를 꺼내서 value1에 넣는다
				if (values[old_top + 1].T == FLOAT) {
					value1 = values[old_top + 1].V.fValue;
				} else {
					value1 = values[old_top + 1].V.dValue;
				}

				// 두번째 피연산자를 꺼내서 value2에 넣는다
```

```
                if (values[old_top + 3].T == FLOAT) {
                        value2 = values[old_top + 3].V.fValue;
                } else {
                        value2 = values[old_top + 3].V.dValue;
                }

                values[top].V.fValue = value1 + value2; // 연산 결과를 저장
                values[top].T = FLOAT; // 연산 결과가 실수형임을 저장
        }
        break;
case 2:
        values[top] = values[old_top + 1];
        break;
case 3:
        if (values[old_top + 1].T != values[old_top + 3].T) {
                // warning
                push_to_ew_list(values[old_top + 3].token_start_index); // 경고 표시할 위치 저장
                conversion_warning_flag = 1; // 플래그 1로 바꿔 정수형->실수형 변환이 일어났음을 표시함
        }

        if (values[old_top + 1].T == INT && values[old_top + 3].T == INT) { // 두 숫자 모두 정수 일때 - 연산 결과가 정수인 경우
                values[top].V.dValue = values[old_top + 1].V.dValue * values[old_top + 3].V.dValue; // 두 수를 곱해서 저장
                values[top].T = INT; // 연산 결과가 정수형임을 저장
        } else { // 연산 결과가 실수인 경우
                float value1, value2;
                // 첫번째 피연산자를 꺼내서 value1에 넣는다
```

```
                if (values[old_top + 1].T == FLOAT) {
                        value1 = values[old_top + 1].V.fValue;
                } else {
                        value1 = values[old_top + 1].V.dValue;
                }

                // 두번째 피연산자를 꺼내서 value1에 넣는다
                if (values[old_top + 3].T == FLOAT) {
                        value2 = values[old_top + 3].V.fValue;
                } else {
                        value2 = values[old_top + 3].V.dValue;
                }

                values[top].V.fValue = value1 * value2; // 연산 결과를 저장
                values[top].T = FLOAT; // 연산 결과가 실수형임을 저장
        }
        break;
case 4:
        values[top] = values[old_top + 1];
        break;
case 5:
        values[top] = values[old_top + 2];
        break;
case 6:
        values[top] = values[old_top + 1];
        break;
default:
        yyerror(/*"parsing table error"*/);
```

```
                        break;
        }
}

void yyerror() {
        push_to_ew_list(command_index);
        print_ew_points();
        printf("syntax error₩n");
        exit(1);
}

int yylex() {
        static char ch = ' ';
        int i = 0;
        int is_dot_included = 0;
        while (ch == ' ' || ch == '₩t') {
                ch = getchar();
                ++command_index;
        }
        if (isdigit(ch)) {
                int token_start_index = command_index - 1;
                do {
                        if (ch == '.') {
                                if (is_dot_included) { // 앞에 이미 '.'이 입력되었다면
                                        lex_error(); // 한 숫자 내에 '.'이 두개 이상 있는 것은 잘못된 숫자 형식이므로
에러
                                } else { // '.'이 처음 입력됐을 때
                                        is_dot_included = 1; // '.'이 입력되었음을 표시 -> 실수형임을 표시
                                }
```

```
                }
                yytext[i++] = ch;
                ch = getchar();
                ++command_index;
        } while (isdigit(ch) || ch == '.');
        yytext[i] = '₩0';

        TT new_value;
        if (is_dot_included) { // 실수
                new_value.V.fValue = atof(yytext); // 문자열을 float형으로 변환
                new_value.T = FLOAT; // float 자료형임을 저장
        } else { // 정수
                new_value.V.dValue = atoi(yytext); // 문자열을 int형으로 변환
                new_value.T = INT; // int 자료형임을 저장
        }
        new_value.token_start_index = token_start_index; // 수식 내에서 이 숫자의 시작 위치를 저장 - 에러, 경고 위치 표시에 사용
        yylval = new_value;

        return(NUMBER);
    } else if (ch == '₩n') {
        return(END);
    }else if (ch == '+') {
        ch = getchar();
        ++command_index;
        return(PLUS);
    } else if (ch == '*') {
        ch = getchar();
        ++command_index;
```

```
                return(STAR);
        } else if (ch == '(') {
                ch = getchar();
                ++command_index;
                return(LPAREN);
        } else if (ch == ')') {
                ch = getchar();
                ++command_index;
                return(RPAREN);
        } else {
                lex_error(END);
                return 0;
        }
}

void lex_error() {
        push_to_ew_list(command_index);
        print_ew_points();
        printf("illegal token₩n");
        exit(1);
}

// ew_list에 원소 추가하는 함수
void push_to_ew_list(int i){
        if (ew_last + 1 < 1000) { // 배열이 꽉찼으면 저장하지 않음
                ew_list[++ew_last] = i;
        }
}
```

```
// 에러, 경고 위치 표시하는 함수
void print_ew_points(){
	int list_i;
	int command_i = 0;
	qsort((void*)ew_list, ew_last + 1, sizeof(int), compare_int); // 앞쪽부터 순차적으로 출력하기 위해 정렬함
	for (list_i = 0; list_i <= ew_last; ++list_i) {
		while (command_i++ < ew_list[list_i]) { // 에러, 경고 출력할 위치까지 이동
			printf(" ");
		}
		printf("^"); // 에러, 경고해야할 위치에 '^'문자 출력
	}
	printf("₩n");
	if (conversion_warning_flag == 1) {
		printf("WARNING: conversion from 'int' to 'float', possible loss of data₩n"); // 경고 메시지 출력
	}
}

// qsort에 사용하는 정수 크기 비교 함수
int compare_int(const void *_a, const void *_b) {
	int *a = (int *)_a;
	int *b = (int *)_b;

	return *a - *b;
}
```