과제 – 4 (신택스 트리)

컴퓨터학부 20160548 이승현

1. 문제 해결 방법

이 과제의 목표는 곱셈, 덧셈 기호와 정수로 이루어진 수식을 입력으로 받아 LR 파서를 이용하여 그 수식의 문법이 제대로 되어있는지를 판단하고, 수식에 대한 신택스 트리를 생성하고 출력하는 프로그램을 작성하는 것이다.

이를 구현하기 위하여 먼저 신택스 트리의 노드 역할을 할 구조체 NODE를 다음과 같이 정의한다.

```
typedef struct node {
        NODE_NAME name;
        int val;
        struct node *llink;
        struct node *rlink;
} NODE;
```

yylex() 함수에서 정수를 읽어 들인 뒤 shift() 함수에서 해당 정수를 값으로 갖는 NODE 구조체를 만들어 values[top]에 넣는다. reduce() 함수에서는 합, 곱 연산자를 reduce 할 때 ADD, MUL을 값으로 갖는 NODE 구조체를 만들고 해당 NODE의 자식 노드를 가리키는 llink, rlink가 두 피연산자를 가리키도록 한다. 이 과정을 반복하며 LR parsing을 한다. 입력된 수식이 제대로 된 수식이었다면(accept) 이 과정이 끝났을 때 value[1]이 신택스 트리의 루트 노드가 된다. parsing이 끝난 뒤 value[1]을 이용하여 신택스 트리의 내용을 중위순회 하며 출력한다. 출력한 내용을 이용하여 트리가 잘 만들어 졌는지 확인할 수 있다. 출력을 끝내면 프로그램을 종료한다.

수식 에러 표시는 과제2와 동일한 방법을 사용했다. 다만 과제4의 프로그램은 정수형 수식만 입력되므로, 혼합형 수식에서 정수와 실수 사이에 연산이 일어날 때 경고를 출력하는 기능은 삭제하였다.

2. 실행 결과

생성된 신택스 트리를 중위 순회하면서 출력한 결과

```
shlee@shlee-virtual-machine:~/workspace/ssucompiler/project4$ ./a.out
3+4*12
INT_VAL, value: 3
ADD
INT_VAL, value: 4
MUL
INT_VAL, value: 12
```

⇒ 정수형 수식

```
shlee@shlee-virtual-machine:~/workspace/ssucompiler/project4$ ./a.out
3  + 4    * 12
INT_VAL, value: 3
ADD
INT_VAL, value: 4
MUL
INT_VAL, value: 12
```

⇒ 공백문자가 포함된 수식

```
shlee@shlee-virtual-machine:~/workspace/ssucompiler/project4$ ./a.out
3.0+4*12
  ^
illegal token
```

⇒ 실수형이 섞여 있는 혼합형 수식

```
shlee@shlee-virtual-machine:~/workspace/ssucompiler/project4$ ./a.out
1*2+2*3+3*4+4*5+5*6
INT_VAL, value: 1
MUL
INT_VAL, value: 2
ADD
INT_VAL, value: 2
MUL
INT_VAL, value: 3
ADD
INT_VAL, value: 3
MUL
INT_VAL, value: 4
ADD
INT_VAL, value: 4
MUL
INT_VAL, value: 5
ADD
INT_VAL, value: 5
MUL
INT_VAL, value: 6
```

⇒ 긴 수식

```
shlee@shlee-virtual-machine:~/workspace/ssucompiler/project4$ ./a.out
(3+2)*10
INT_VAL, value: 3
ADD
INT_VAL, value: 2
MUL
INT_VAL, value: 10
```

⇒ 괄호가 있는 수식

```
shlee@shlee-virtual-machine:~/workspace/ssucompiler/project4$ ./a.out
1+2*
   ^
syntax error
```

⇒ 연산자가 잘못 사용된 수식

```
shlee@shlee-virtual-machine:~/workspace/ssucompiler/project4$ ./a.out
3++2
  ^
syntax error
```

⇒ 연산자가 잘못 사용된 수식

```
shlee@shlee-virtual-machine:~/workspace/ssucompiler/project4$ ./a.out
2*(22+33
      ^
syntax error
```

⇒ 닫는 괄호가 없는 수식

```
shlee@shlee-virtual-machine:~/workspace/ssucompiler/project4$ ./a.out
2*(22+33)
INT_VAL, value: 2
MUL
INT_VAL, value: 22
ADD
INT_VAL, value: 33
```

⇒ 닫는 괄호가 있는 수식 2

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

typedef enum {INT_VAL, ADD, MUL} NODE_NAME;
typedef struct node {
        NODE_NAME name;
        int val;
        struct node *llink;
        struct node *rlink;
} NODE;

NODE *values[1000];
NODE *makenode(NODE_NAME name, int v, NODE *p, NODE *q) {
        NODE *n;
        n = (NODE *)malloc(sizeof(NODE));
```

```
	n->name = name;
	n->val = v;
	n->llink = p;
	n->rlink = q;

	return n;
}

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};

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

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

// 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
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


void print_values(NODE *node); // 신택스 트리를 중위순회하며 출력하는 함수
void print_node(NODE *node); // NODE의 내용을 출력하는 함수

int main() {
	yyparse();

	print_values(values[1]);

	return 0;
}
```

```
int 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);
}

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

void shift(int i) {
	push(i);
	if (sym == NUMBER) {
		values[top] = makenode(INT_VAL, yyval, NULL, NULL);
```

```
        }
        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:
                        values[top] = makenode(ADD, 0, values[old_top + 1], values[old_top + 3]); // ADD NODE 생성
                        break;
                case 2:
                        values[top] = values[old_top + 1];
                        break;
                case 3:
                        values[top] = makenode(MUL, 0, values[old_top + 1], values[old_top + 3]); // MUL NODE 생성
                        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)) {
        do { // 연속된 숫자들을 계속 읽어들인다
            yytext[i++] = ch;
            ch = getchar();
            ++command_index;
```

```
                } while (isdigit(ch));
                yytext[i] = '₩0';
                yyval = atoi(yytext); // 문자열을 int형으로 변환
                return(NUMBER);
        } else if (ch == '₩n') {
                return(END);
        } else if (ch == '+') {
                ch = getchar();
                return(PLUS);
        } else if (ch == '*') {
                ch = getchar();
                return(STAR);
        } else if (ch == '(') {
                ch = getchar();
                return(LPAREN);
        } else if (ch == ')') {
                ch = getchar();
                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);
}

// NODE의 정보를 출력하는 함수
void print_node(NODE *node) {
        switch(node->name) {
                case INT_VAL:
                        printf("INT_VAL, value: %d₩n", node->val);
                        break;
                case ADD:
                        printf("ADD₩n");
                        break;
                case MUL:
                        printf("MUL₩n");
                        break;
        }
}

// 신택스 트리를 중위순회하면서 출력하는 함수
void print_values(NODE *node) {
        if (node == NULL) yyerror();
        if (node->llink != NULL) {
                print_values(node->llink);
        }
        print_node(node);
        if (node->rlink != NULL) {
                print_values(node->rlink);
        }
```

```
        return;
}

// 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");
}

// 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;
}
```