컴파일러 과제

LR Parser

제출일자 : 2016.09.19

담당교수 : 유재우 교수님
소속학부 : 컴퓨터학부
학번 : 20142577
이름 : 홍상원

1. source code

```
#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 1000

//parsing table
int actionTable[12][6] = {// [상태번호][심볼] / 양의 정수 : Shift, 0 : 공백, 음의 정수 : Reduce
        {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 gotoTable[12][3] = { //[상태번호][심볼]
        {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}
};

//규칙
//left handside
int produce[7] = {0, EXPRESSION, EXPRESSION, TERM, TERM, FACTOR, FACTOR};
//length of right handside
int produceLen[7] = {0, 3, 1, 3, 1, 3, 1};

//스택 관련
int stack[1000];
int top = -1;
//심볼저장
int symbol;
//값저장
char text[32];
int value[1000];
```

```
int val;

void LRParse();
int Symbol();//심볼 구분 함수
void Push(int);//스택 push 함수
void Reduce(int);
void Shift(int);
void SyntaxError();
void LexicalError();


int main(void) {
	LRParse();

	printf("결과값: %d₩n", value[top]);//계산결과 출력
	return 0;
}

void LRParse() {
	int state;// 상태 저장변수
	//맨 처음 0 을 스택에 저장
	stack[++top] = 0;
	//심볼 구하기
	symbol = Symbol();

	do {
		state = actionTable[stack[top]][symbol - 256];//상태저장
		// state 의 값에 따라 처리
		if(state == ACC)
			printf("success!₩n");
		else if(state > 0)
			Shift(state);
		else if(state < 0)
			Reduce(-state);
		else//state == 0 이면
			SyntaxError();
	} while(state != ACC);
}

int Symbol() {
	static char ch = ' ';//입력값 저장
	int i = 0;

	while(ch == ' ' || ch == '₩t')//공백이 입력되면 다시 입력
		ch = getchar();
```

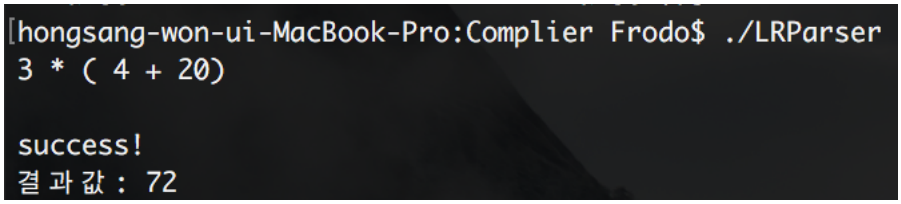
```
	//ch 의 심볼 반환
	if(isdigit(ch)) {
		do {
			text[i++] = ch;
			ch = getchar();
		} while(isdigit(ch));
		text[i] = 0;
		val = atoi(text);
		return NUMBER;
	}
	else if(ch == '+') {
		ch = getchar();
		return PLUS;
	}
	else if(ch == '*') {
		ch = getchar();
		return STAR;
	}
	else if(ch == '(') {
		ch = getchar();
		return LPAREN;
	}
	else if(ch == ')') {
		ch = getchar();
		return RPAREN;
	}
	else if(ch == EOF || ch == '₩n') {
		ch = getchar();
		return END;
	}
	else {
		if(ch != '₩n')
			LexicalError();
	}
}

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

void Shift(int i) {
	Push(i);
	value[top] = val;
	symbol = Symbol();
}
```

```
void Reduce(int i) {
	int exTop;
	top -= produceLen[i];
	exTop = top;
	Push(gotoTable[stack[exTop]][produce[i]]);
	//Reduce 할 때마다 값을 계산
	switch(i) {
		case 1:
			value[top] = value[exTop + 1] + value[exTop + 3];
			break;
		case 2:
			value[top] = value[exTop + 1];
			break;
		case 3:
			value[top] = value[exTop + 1] * value[exTop + 3];
			break;
		case 4:
			value[top] = value[exTop + 1];
			break;
		case 5:
			value[top] = value[exTop + 2];
			break;
		case 6:
			value[top] = value[exTop + 1];
			break;
		default:
			SyntaxError();//Parsing Table Error
			break;
	}
}

void SyntaxError() {
	printf("syntax error!₩n");
	exit(1);
}


void LexicalError() {
	printf("illegal token!₩n");
	exit(1);
}
```

2. 출력 결과

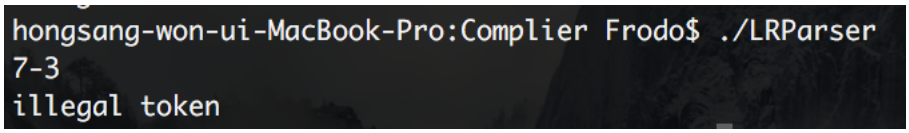
1) 정상적인 경우

- +, *, (,) 연산자 십의 자리 이상 입력 가능 / 십의 자리 이상 음이 아닌 정수 입력 가능



2) 오류가 나는 경우

- 파싱 테이블에 포함되지 않은 연산을 하는 경우 : Lexical Error



- 파싱 테이블에 잘못 접근하는 경우 : Syntax Error