LRParser 설계 및 구현

https://github.com/computerphilosopher/LRParser

1311068 한국어문학부 박범수

1. Parsing Table 설계 – Action Table

- Action 클래스의 2차원 배열로 구현
- Shift, reduce 등의 액션을 enum을 이용해 저장

1. Parsing Table 설계 – Action Table

```
Action actionTable[STATE_COUNT][TERMINAL_COUNT] = {
    //0
    Action(shift, 5), Action(error, ERROR), Action(error, ERROR), Act
    //1
    Action(error, ERROR), Action(shift, 6), Action(error, ERROR), Act
    //2
    Action(error, ERROR), Action(reduce, 2), Action(shift, 7), Actio
    //3
    Action(error, ERROR), Action(reduce,4), Action(reduce, 4), Actio
    //4
    Action(shift, 5), Action(error, ERROR), Action(error, ERROR), Act
    //5
    Action(error, ERROR), Action(reduce, 6), Action(reduce,6), Actio
```

```
class Action {

private:
    int type; //shift or reduce;
    int num; // state or rule number

public:

    Action(int type, int num);

    int GetType();
    string GetTypeString()

    int GetState();

    int GetRuleNumber();

};
```

1. Parsing Table 설계 – GOTO Table

GOTO 테이블		
E	T	F
1	2	3
8	2	3
	9	3
		10

```
const int gotoTable[STATE_COUNT][NONTERMINAL_COUNT] = {
    1, 2, 3,
    ERROR, ERROR, ERROR,
    ERROR, ERROR, ERROR,
    ERROR, ERROR, ERROR,
    8, 2, 3,
    ERROR, ERROR, ERROR,
    ERROR, 9, 3,
    ERROR, ERROR, 10,
    ERROR, ERROR, ERROR,
    ERROR, ERROR, ERROR,
    ERROR, ERROR, ERROR,
    ERROR, ERROR, ERROR,
};
```

2. Stack 설계 - 개요

- Enum을 이용하는 정수 스택으로 구현
- STL 스택을 사용

2. Stack 설계 – 코드

```
enum Terminal {
    ID = 21,
    ADD,
    MUL,
    LEFT_PAREN,
    RIGHT_PAREN,
    END_OF_STRING
};

enum NonTerminal {
    E = 100,
    T,
    F
};
```

```
while (!temp.empty()) {
    int k = temp.top();

    switch (k) {
    case ID:
        ret.append("a");
        break;
    case ADD:
        ret.append("+");
        break;
    case MUL:
        ret.append("*");
        break;
    case LEFT_PAREN:
        ret.append("(");
        break;
    case RIGHT_PAREN:
        ret.append(")");
        break;
```

3. Input 설계

```
while (1) {
    string input;
    cout << "\ninput test string>>";

    cin >> input;

    if (input == "exit" || input == "quit") {
        break;
    }
```

C++ 스트링 클래스 사용

4. Rule 설계 - 개요

```
class Rule {

private:
public:
    Rule();

    static int GetRHSCount(Action action);
    static int GetLHS(Action action);
};
```

RHS의 글자 수를 리턴하는 메소드, LHS의 enum값을 리턴하는 메소드로 구성

4. Rule 설계 – GetRHSCount

Grammar

1. E → E + T
2. E → T
3. T → T * F
4. T → F
5. F → (E)
6. F → id

```
int Rule::GetRHSCount(Action action) {

    int ruleNumber = action.GetRuleNumber();

    switch (ruleNumber) {
    case 1:
        return 3;
        break;
    case 2:
        return 1;
        break;
    case 3:
        return 3;
        break;
```

4. Rule 설계 – GetLHS

Grammar

1. E → E + T
2. E → T
3. T → T * F
4. T → F
5. F → (E)
6. F → id

```
int Rule::GetLHS(Action action) {

    int ruleNumber = action.GetRuleNumber();

    switch (ruleNumber) {

    case 1:
        return E;
        break;
    case 2:
        return E;
        break;
    case 3:
        return T;
        break;
    case 4:
        return T;
        break;
```

2. LR 파서 설계 – 클래스 멤버

- Stack
- Action Table
- GOTO Table
- Input

2. LR 파서 설계 – 루프

```
while (true) {

    int symbol = GetSymbol(input[i]);
    Action action = GetAction(cur_state, input[i]);

    int actionType = action.GetType();

    switch (actionType) {

    case shift:
        cur_state = Shift(action, symbol);
        i++;
        break;

    case reduce:
        cur_state = Reduce(action);
        break;

    case accept:
        cur_state = STATE::ACCEPT;
        break;
```

Stack	Input	Action		
0a5	+a*a	shift	5	
0F3	+a*a	shift	6	
0T2	+a*a	shift	4	
0E1	+a*a	shift	2	
0E1+6	a*a	shift	6	
0E1+6a5	*a	shift	5	
0E1+6F3	*a	shift	6	
0E1+6T9	*a	shift	4	
0E1+6T9*7		a	shift	7
0E1+6T9*7a5			shift	5
0E1+6T9*7F01			shift	6
0E1+6T9		shift	3	
0E1		shift	1	
0E1		accept	<	
Accept!				

2. LR 파서 설계 – Shift

```
int LRParser::Shift(Action action, int symbol) {

    int cur_state = action.GetState();

    parsingStack.push(symbol);
    parsingStack.push(action.GetState());

    return cur_state;
}
```

입력과 상태를 스택에 PUSH

2. LR 파서 설계 – Reduce (POP RHS)

```
int LRParser::Reduce(Action action) {
    int k = Rule::GetRHSCount(action);
    int cur_state = 0;

    for (int j = 0; j < k * 2; j++) {
        parsingStack.pop();
    }
```

RHS의 두 배 만큼 pop 하는 과정

2. LR 파서 설계 – Reduce (Push LHS)

```
int lhs = Rule::GetLHS(action);
parsingStack.push(lhs);
```

2. LR 파서 설계 – Reduce (GOTO)

```
int new_state = gotoTable[cur_state][lhs - E];
parsingStack.push(new_state);
cur_state = new_state;

return cur_state;
```

Reduce 함수에서 GOTO를 함께 처리