Compiler Assignment 1
Design SLR

창의 ICT 공과대학
컴퓨터공학과
20115725
안 준 형

1. Data Structure Explanation

쓰인 자료구조의 형태는 세가지 입니다. 1,2 차원 배열, 연결 리스트, 그리고 스택 입니다.

- 1,2 차원 배열

```
char rule[MAX_RULE_SIZE][MAX_RULE_LENGTH];
```

초기에 입력 받을 rule 을 저장하는 자료구조, MAX_RULE SIZE 는 rule 의 최대 개수를, MAX_RULE LENGTH 는 rule 의 최대 길이를 의미합니다. 저는 각각 30 씩 할당하면 충분하다고 판단하여 30 으로 했습니다.

```
char NonTerminal[MAX_RULE_SIZE];
char Terminal[MAX_RULE_SIZE];
```

Non Terminal 과 Terminal symbol 들을 따로 저장하는 자료구조 입니다.

```
char FIRST_TABLE[MAX_RULE_SIZE][MAX_RULE_LENGTH];
char FOLLOW_TABLE[MAX_RULE_SIZE][MAX_RULE_LENGTH];
```

각각의 TABLE 은 1 행에 E(x 2 행에 T:(x 형태로 FIRST, FOLLOW 에 해당하는 값을 string 으로 저장하고 있습니다.

```
char PARSING_TABLE_SYMBOLS[MAX_RULE_SIZE];
int PARSING_TABLE[MAX_RULE_SIZE][MAX_RULE_LENGTH];
```

PARSING TABLE 은 s1, s2 같은 string 형태의 element 를 가지고, index 를 표시하는 것은 숫자(0,1,2,3,4....)와 문자(x,:$E,A....)가 혼용되어 나오므로 한번에 자료구조로 표현하기에는 복잡하다고 판단했습니다. PARSING TABLE 의 index 를 찾기 위한 PARSING_TABLE_SYMBOLS 를 만들어 PARSING TABLE 의 세로축의 index 를 저장했습니다.

PARSING TABLE 의 계산을 편리하게 하기 위해 s1, s2 같은 shift 연산은 앞에 s 를 생략하고 양의 정수로 표기했습니다. r1,r2 같은 reduce 연산은 -1,-2 처럼 음의 정수로 표기했습니다. TABLE 에서 ACCEPT 는 특별한 integer 인 -10000 으로 나타냈습니다. PARSING TABLE 의 빈 element 는 0 로 표시해 int 형 2 차원 배열로 PARSING TABLE 을 구현하였습니다.

- 연결 리스트

```
index_node* i0_node; //pointing I0,I1....

typedef struct node  {
```

```
    char p_rule[20];
    struct node* next;
}node;

typedef struct index_node {
    int i;
    node* get;
    struct index_node* next;
}index_node;
```

I0, I1... 이나 GOTO(I2,x) 와 같은 연산을 처리하기 위해 여러 개의 string 을 한 행으로 가지고 있는 2 차원 자료구조가 필요했습니다.
그것을 처리하기 위해 연결리스트를 만들고 하나의 node 에는 string 과 행 번호를 나타내는 index 를 갖도록 했습니다.

Index_node 는 행의 index 가 되는 node 입니다. I0,I1...Ix 가 추가 될때마다 동적으로 생성됩니다.

- 스택

```
char stack[MAX_RULE_LENGTH];        // stack for parsing construction
int top;            //
```

스택은 input string 과 PARSING TABLE 을 이용해 ACCEPT 할 것인지 Deny 할 것인지 결정하기 위해 쓰이는 자료구조 입니다.

2. Function Explanation

```
char* first(char nt);
char* follow(char nt);
```

first 와 follow TABLE 에 저장된 값을 가져옵니다. E:x)# 가 저장되어 있으면 x)#를 반환합니다. 구체적으로는 :이후의 메모리를 가리키는 포인터를 return 합니다.

```
int isTerminal(char t);
int isNonTerminal(char t);
int isExist(char*s, char c);
```

Nonterminal 의 기준은 A~Z 입니다 이를 제외한 것은 Terminal symbol 이라고 규정하고 포함되면 1 아니면 0 을 return 합니다. IsExist 함수는 string 과 char 를 입력하면 해당 string 에 char 가 있는지를 판단해 있으면 1, 없으면 0 을 반환합니다.

```
void union_symbol(char* s, char c);
void union_string(char* s1, char* s2, int* f);
void union_string_without_epsilon(char* s1, char* s2);
```
string 과 char, string 과 string 을 합치는 함수입니다. First 와 Follow 를 구하는 함수 내부에서 쓰이는 데, 목적지 string 에 변화되는 바가 없을 때까지 while 문이 돌아가므로 변화를 표시하는 flag 를 추가로 int* 형태로 넘겨줍니다.

```
void add_list(node* np, char* item);
void add_first_list(index_node* i_n,char* item );
void  free_I(index_node* i);
```
C construction 을 하기 위해 I0, I1... 을 구하는 과정에서 연결리스트를 연산하는 함수입니다. add_list 는 노드를 생성하고 연결리스트의 마지막에 추가하는 함수, add_fisrt_list 는 행을 처음 생성하면 index node 에 연결시켜주는 노드를 생성하는 함수입니다. Free_I 는 행 전체를 삭제하는 연산입니다.

```
void eliminate(char* s, char c);
void dot(char*s, char c);
void move_dot(char*s);
char get_after_dot(char* s);
```
eliminate 함수는 string 에 주어진 char 를 제거합니다. dot 함수는 주어진 rule 의 가장 처음에 dot 을 찍습니다. 예를 들면 E>AB 를 넘기면 E>.AB 가 됩니다. move_dot 함수는 rule 의 dot 을 한 symbol 뒤로 옮깁니다. E>.AB 가 파라미터로 주어지면 E>A.B 가 됩니다. get_after_dot 함수는 dot 뒤의 symbol 을 return 합니다.

```
int _goto(index_node* i, char c);
int is_overlap(index_node* i, int* ans);
void  free_I(index_node* i);
```
_goto 함수는 GOTO(I,s)를 처리하기 위한 함수로 _goto 함수 안에서 I1,I2...In 까지 계산이 이루어집니다.

is_overlap 함수는 새로 만들어진 I (E>A.B E>B+.a...)이 기존에 있던 I 와 겹치는지 판단합니다. Free_I 함수는 겹치는 I 가 있으면 새로 만들었던 I 를 제거하기 위해 만들었습니다.

```
int is_dervied_Epsilon(char s);
int is_accepted(char* input);
```
is_dervied_Epsilon 함수는 주어진 symbol 이 rule 을 따라가면 epsilon 으로 되는지를 판단합니다.

Is_accepted 는 input string 이 스택과 PARSING TABLE 을 이용해 accept 가능한지를 판단하고 결과를 반환합니다.

```
void print_list();
void print_first_table();
void print_follow_table();
void print_c_table();
void print_action();
void print_goto();
```

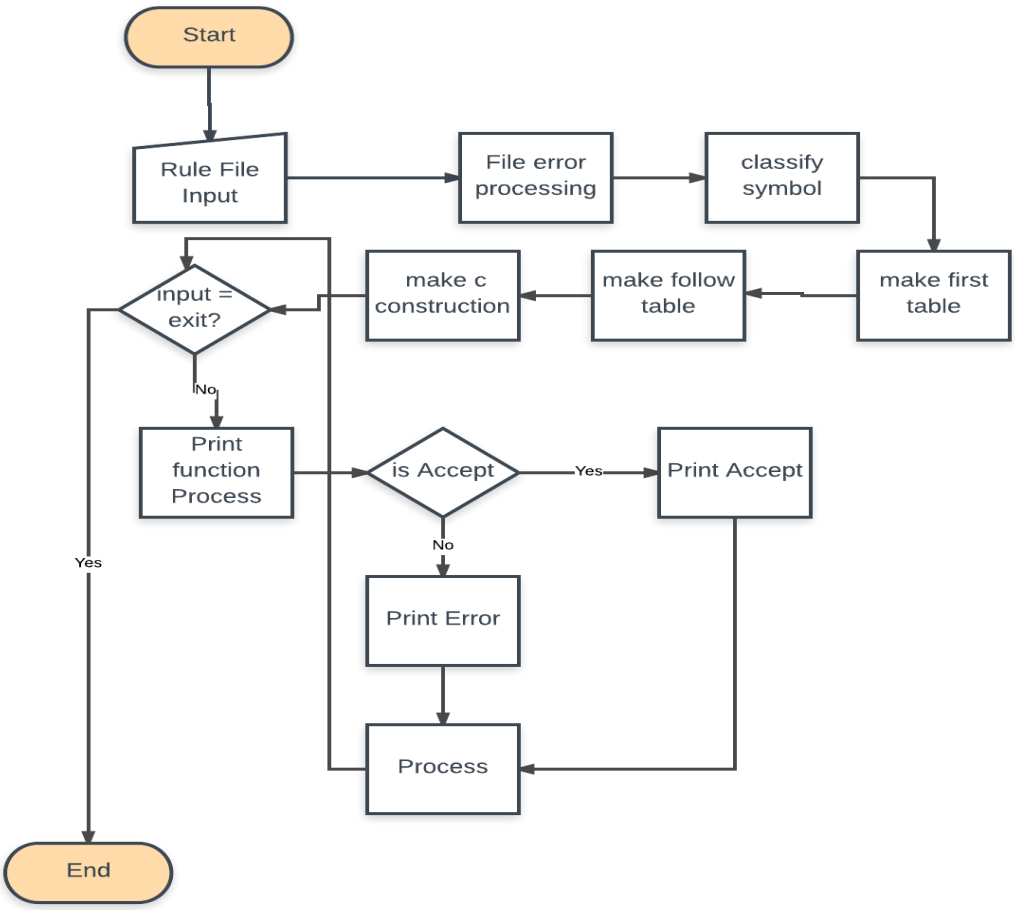
void print_lx(int i);

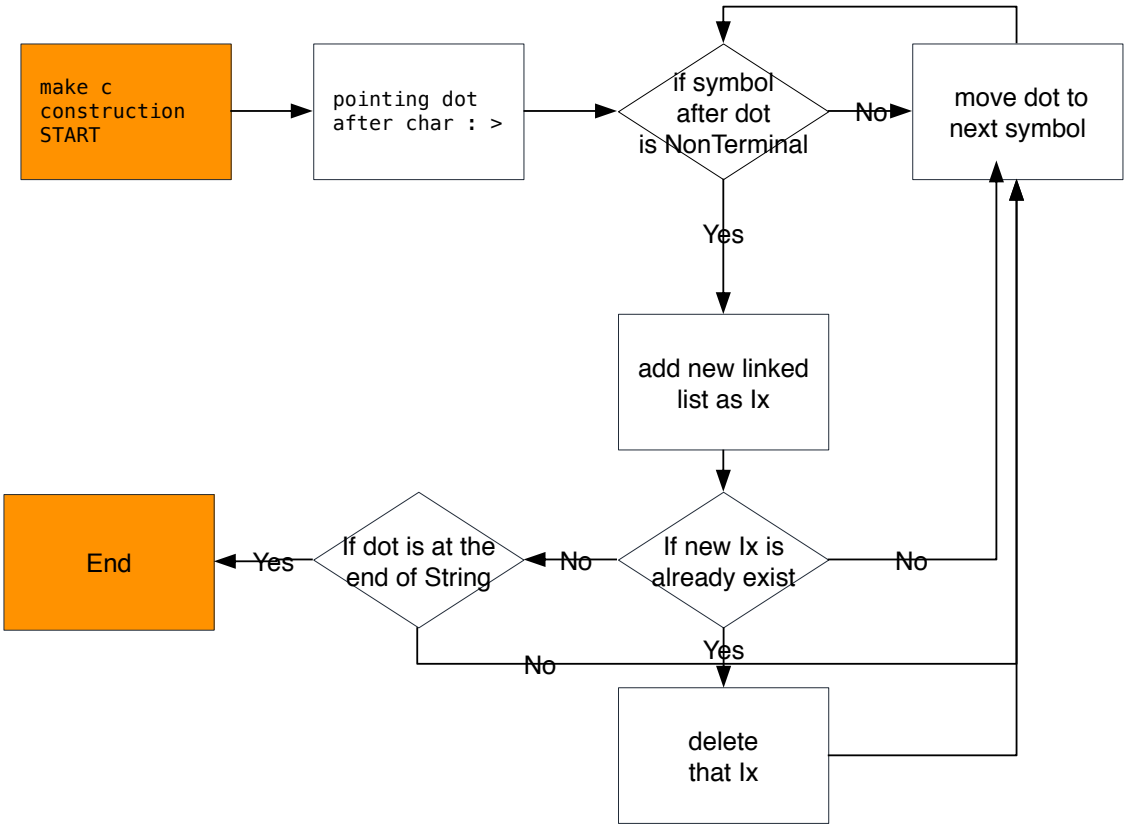
출력 함수입니다.

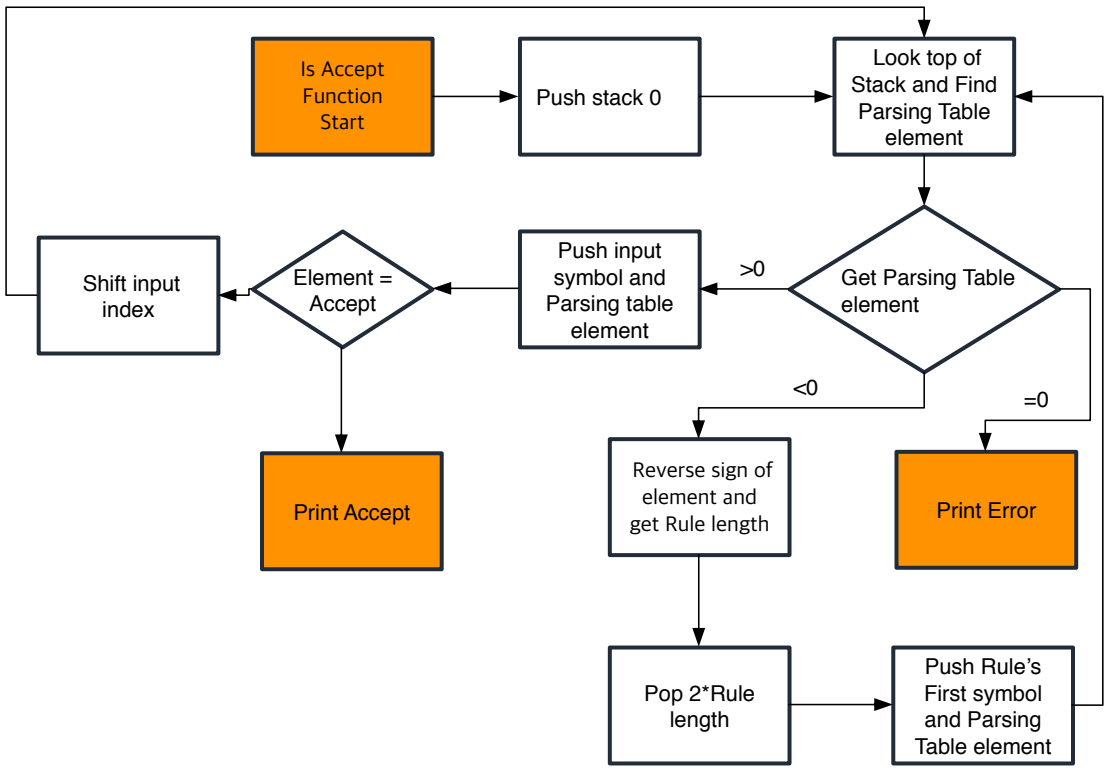
```
void classify_symbol();
void make_first_table();
void make_follow_table();
void make_c_construction();
```

프로그램의 거시적인 flow 를 나타내는 함수들 입니다. Symbol 을 terminal 과 Non Terminal 로 나누고, First,Follow Table 을 만든 후, C0 construction 을 하는 부분을 나눈 함수입니다.

3. Flow Chart




make c construction START
pointing dot after char : >
if symbol after dot is NonTerminal
No
move dot to next symbol
Yes
add new linked list as Ix
End
Yes
If dot is at the end of String
No
If new Ix is already exist
No
No
Yes
delete that Ix


Is Accept Function Start
Push stack 0
Look top of Stack and Find Parsing Table element
Get Parsing Table element
>0
Push input symbol and Parsing table element
Element = Accept
Shift input index
Print Accept
<0
Reverse sign of element and get Rule length
Pop 2*Rule length
Push Rule's First symbol and Parsing Table element
=0
Print Error