Universidad Nacional Autónoma de México
Facultad de Ingeniería
Compiladores
Práctica Analizador Sintáctico Descendente LL(1)
Profesor: Adrián Ulises Mercado Martínez
Ayudante: Carlos Gerardo Acosta

Analizador Sintáctico Descendente

Objetivo:

El estudiante elaborara un analizador sintáctico descendente del tipo LL(1) para una gramática.

Desarrollo:

1. Seleccione alguna de las siguientes gramáticas
   (a) S → S ∨ S | S* | S+ | S? | (S) | SS | a | b
   (b) E → E + E | E - E | E * E | E / E | E % E | (E) | num
2. Realice las modificaciones necesarias para que la gramática sea LL(1).
3. Represente los símbolos de la gramática utilizando un arreglo de la siguiente estructura

```
#define TERM 1
#define NON_TERM 2
#define EPSILON 3

typedef struct _symbol{
  short type;
  char name[15];
  short pos;
} sym;
```

   - type indica el tipo de símbolo que es.
   - name es el nombre del símbolo, que puede ser opcional
   - pos es la posición que ocupa el símbolo dentro de la tabla de análisis, ya sea su renglón o su columna dependiendo si es no terminal o terminal.

4. Represente las producciones de la gramática utilizando un arreglo de la siguiente estructura.

```
typede struct _production{
  short head;
  char *body;
  short num;
} prod;
```

   donde:

   - head es el encabezado de la producción y es el indice que ocupa en el arreglo de símbolos
   - body es una cadena de caracteres que contiene los indices de los símbolos que conforman el cuerpo de la producción. Separados por espacios en blanco o por /. Ejemplo body = "1/2/3" suponiendo que esta producción esta compuesta de los símbolos cuyos indices son el 1, 2 y 3.
   - num es el número de símbolos que componen el cuerpo de la producción. En el ejemplo anterior num = 3

5. Utilizando el arreglo arreglo de producciones para manejar indices llene la tabla de análisis sintáctico.

```
int matriz[NUM_NON_TERM][NUM_TERM];
```

6. Elabore para la gramática de su selección, el analizador léxico utilizando lex, recuerde que el entero a devolver debe corresponder con las posiciones de los terminales. **Sugerencia:** utilice <<EOF>> para cambiar el valor de retorno predeterminado de lex cuando encuentra el fin de archivo.

7. Con las siguientes estructuras:

```
typedef struct _node node;

struct _node{
  sym info;
  node *next;
};


struct _stack{
  node *root;
  int num;
}
```

   Elabore una pila que será utilizada para el análisis sintáctico.

8. Implemente el algoritmo de análisis sintáctico utilizando la pila y la tabla de análisis sintáctico en conjunto con el programa generado por lex.

9. Escriba sus conclusiones

Algoritmo de análisis sintáctico

Entrada: Una cadena de entrada, token el apuntador a la entrada, una pila
Resultado: Si la cadena fue aceptada o no.

```
token = yylex();
Hacer X la cima de la pila;
mientras X ≠ $ hacer
    si X = a entonces
        pop();
        token = yylex();
    si no, si  X ∈ Σ y X ≠ token entonces
        error();
    si no, si M[X, token] = X → Y1Y2Y3...Yk entonces
        pop();
        para i = k, i¿k, 1 hacer
            push(Yi);
        fin
    si no, si M[X, token] = error entonces
        error();
fin
si token = $ entonces
    Escribir "La cadena es aceptada";
fin
en otro caso
    Escribir "La cadena no es aceptada";
fin
```