Grupo Miércoles 10:00-12:00 semanas A

-Práctica 3-

Autor: Hector Toral Pallas

NIP: 798095

Preguntas:

¿Qué es un token? Fragmento de la entrada que corresponde a un solo símbolo terminal.

¿Qué son $$,$1,$2?

-$$ es la pseudo-variable que representa el valor semántico para la agrupación de la regla que va a construir.

-$1 y $2 medio por el cual se accede al valor semántico de las reglas.

¿Qué es yylval? Lugar donde se almacena el valor semántico del token.

¿Qué es el “start symbol”? Es el primer símbolo no terminal que se encuentra en la especificación de la gramática.

¿Cómo se define en bison? `%start symbol`

¿Cómo escribirías en bison la siguiente gramática?

S -> aS | T

T -> bTb | vacio

```
S : a S
    |   T
    ;
T : b T b;
    /* vacio */
```

Ejercicio 1:

La gramática que corresponde a calcOrig.y es:

exp -> exp ADD factor | exp SUB factor | factor

factor -> exp MUL factor | exp DIV factor | term

term -> OP exp CP | NUMBER

Al compilar los diferentes fuentes estos son los mensajes que he obtenido:

```
┌──(hec7or㉿kali)-[~/Documentos/practica3]
└─$ bison -yd calcOrig.y
calcOrig.y: aviso: 16 conflictos reducción/reducción [-Wconflicts-rr]
calcOrig.y: note: rerun with option '-Wcounterexamples' to generate conflict counterexamples

┌──(hec7or㉿kali)-[~/Documentos/practica3]
└─$ flex calcOrig.l

┌──(hec7or㉿kali)-[~/Documentos/practica3]
└─$ gcc lex.yy.c y.tab.c -lfl -o calcOrig
y.tab.c: In function ‘yyparse’:
y.tab.c:1039:16: warning: implicit declaration of function ‘yylex’ [-Wimplicit-function-declaration]
 1039 |       yychar = yylex ();
      |                ^~~~~
y.tab.c:1216:7: warning: implicit declaration of function ‘yyerror’; did you mean ‘yyerrok’? [-Wimplicit-function-declaration]
 1216 |       yyerror (YY_("syntax error"));
      |       ^~~~~~
      |       yyerrok
```

Errores de calOrig:

-1+5: syntax error

5+5+5+5+5: syntax error

5/0: floating point exception

Errores de calMejor:

-1+5: syntax error

5/0: floating point exception

5+5+5+5+5 ahora funciona por su definición recursiva.

Ejercicio 2.1:

1. Resumen

He añadido una nueva regla que detecta valores numéricos [0,1] seguidos del carácter “b” de tal manera que ese número sea interpretado como un número binario para pasarlo y operar con él como un entero.

2. Pruebas

Para probar mi nueva regla funciona medianamente bien he realizado las siguientes pruebas:

10+3
10b+3
-5+5
-5 + 10
-5+10b

Ejercicio 2.2:

1. Resumen

He añadido dos nuevas regla, una que detecta el “;” y otra que detecta “;b”, la segunda lo que hace es implementar la nueva acción de interpretar el resultado en binario.

2. Pruebas

Para probar que las nuevas reglas funcionan medianamente bien he realizado las siguientes pruebas:

10+3;
10+3;b
-5+5;
-5+5;b
-5 + 10;
-5+10;b

Ejercicio 2.3:

1. Resumen

He realizado 2 modificaciones en el “.l” y en el “.y” de tal manera que he añadido la regla de acumular “acum” para la variable que acumula y la de asignación “:=” par realizar asignaciones a la variable “acum”. Y por último en el “.y” he añadido las reglas gramaticales para poder añadir la variable acum.

2. Pruebas

Para probar que las nuevas reglas funcionan medianamente bien he realizado las siguientes pruebas:

acum := 5
acum+5
acum :=-10
acum+30

Ejercicio 3:

1. Resumen

Simplemente he traducido la gramática a la sintaxis de bison, y he creado las respectivas reglas en flex para poder detectar los tokens correspondientes para interpretarlos en bison.

2. Pruebas

Para probar que funciona bien he realizado las siguientes pruebas:
./nombreEjecutable <test.txt> resultados.txt como estaba vacio pues esta correcto

3. Lenguaje que reconoce

Ni idea