



INSTITUTO POLITÉCNICO NACIONAL
ESCUELA SUPERIOR DE CÓMPUTO

Práctica número 1:
Yacc básico

Calculadora para vectores

2 de enero de 2021

Grupo: 3CM7

Nombre del alumno:
Ramos Mesas Edgar Alain

Número de boleta:
2013090243

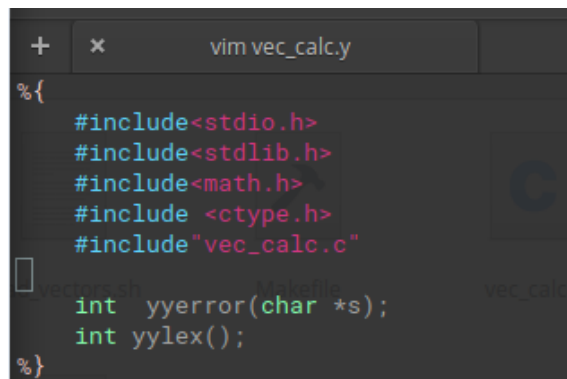
MATERIA: COMPILADORES

1. Introducción

Los programas yacc y lex son herramientas de gran utilidad, compatibles y necesarias entre sí, para un diseñador de compiladores. Muchos compiladores se han construido utilizando estas herramientas (p.ej., el compilador de C de GNU (gcc)) o versiones más avanzadas. Los programas bison y flex son las versiones más modernas (no comerciales) de yacc y lex, y se distribuyen bajo licencia GPL con cualquier distribución de Linux (y también están disponibles para muchos otros UNIX). El programa lex genera analizadores léxicos a partir de una especificación de los componentes léxicos en términos de expresiones regulares (en el estilo de UNIX); lex toma como entrada un fichero (con la extensión.l) y produce un fichero en C (llamado "lex.yy.c") que contiene el analizador léxico. Yacc es un programa para generar analizadores sintácticos. Las siglas del nombre significan Yet Another Compiler-Compiler, es decir, . Otro generador de compiladores más". Genera un analizador sintáctico (la parte de un compilador que comprueba que la estructura del código fuente se ajusta a la especificación sintáctica del lenguaje) basado en una gramática analítica escrita en una notación similar a la BNF. Yacc fue desarrollado por Stephen C. Johnson en AT&T para el sistema operativo Unix. Después se escribieron programas compatibles, por ejemplo Berkeley Yacc, GNU bison, MKS yacc y Abraxas yacc. Cada uno ofrece mejoras leves y características adicionales sobre el Yacc original, pero el concepto ha seguido siendo igual. Yacc también se ha reescrito para otros lenguajes, incluyendo Ratfor, EFL, ML, Ada, Java, y Limbo.

2. Desarrollo

La práctica consistió básicamente en desarrollar el archivo vec_calc.y (modificando el archivo hoc1.y) para poder realizar la calculadora de manera correcta. Debido a que el profesor proporcionó las funciones necesarias para las operaciones con los vectores, simplemente fue necesario copiarlas en un archivo llamado vec_calc.c e incluirlo en la sección de encabezados del archivo vec_calc.y.



```
+ x vim vec_calc.y
%{
#include<stdio.h>
#include<stdlib.h>
#include<math.h>
#include <ctype.h>
#include"vec_calc.c"

int yyerror(char *s);
int yylex();
%}
```

Tras agregar los archivos de cabecera necesarios se agregaron las reglas gramaticales necesarias, así como las acciones semanticas (encerradas entre llaves a lado de las reglas gramaticales) para la calculadora. Además fue importante recordar que el tipo de yylval es el tipo de los elementos de la pila de yacc (en este caso, se trataba de una unión). También fue necesario borrar la

definición que se tenía de `yylex()` en el archivo brindado por el profesor y modificar el tipo de retorno lo cual permitió crear los vectores y valores resultantes de las operaciones.

```

/**Gramática**/
inputString:
| inputString list;

list: '\n'
| exp '\n' {imprimeVector($1);}
| number '\n' {printf("%lf\n", $1);}
;

exp: vector
| exp '+' exp {$$ = sumaVector($1, $3);} //Suma de vectores
| exp '-' exp {$$ = restaVector($1, $3);} //Resta de vectores
| exp '*' NUMBER {$$ = escalarVector($3, $1);} //Multiplicación por escalar
| NUMBER '*' exp {$$ = escalarVector($1, $3);} //Mult por escalar V2
| exp 'x' exp {$$ = productoCruz($1, $3);} //Producto Cruz
;

number: NUMBER
| vector '[' vector {$$ = productoPunto($1, $3);} //Producto Punto
| '[' vector '[' {$$ = vectorMagnitud($2);} //Magnitud
;

/**
 * vector -> NUMBER NUMBER NUMBER
 */
vector: '[' NUMBER NUMBER NUMBER ']' {Vector *v = creaVector(3);
    v -> vec[0] = $2;
    v -> vec[1] = $3;
    v -> vec[2] = $4;
    $$ = v;}
;

```

Posteriormente solo fue necesario compilar el archivo con `yacc` para generar los archivos que serían compiladores después con `gcc` y así finalmente generar el archivo ejecutable. Para ello se utilizó un archivo `Makefile` en el que se incluyeron los pasos de compilación necesarios para poder obtener el archivo ejecutable llamado `calc`.

```

+ x Prac1_1:make
edrasen@edrasen-HP-Pavilion-Laptop-15-cw1xxx:~/Documentos/Compiladores/Prac1_1$ cat Makefile
Gram=y.tab.c y.tab.h
all: $(Gram)
    @gcc -o calc y.tab.c -lm
    @echo Compiled
    @echo un archivo Makefile en el
    $(Gram): vec_calc.y
    calc. @yacc -d vec_calc.y
clean:
    @rm -f *.out *.tab.* calc
    @echo Clean
edrasen@edrasen-HP-Pavilion-Laptop-15-cw1xxx:~/Documentos/Compiladores/Prac1_1$ make
Compiled

```

Una vez que se obtuvo el archivo ejecutable se genero un script llamado load_vectors.sh con un comando para ejecutar el archivo calc pasandole como datos de entrada un conjunto de operaciones escritas en un archivo de texto llamado input_val.txt, el cual contiene los vectores de prueba que se muestran en la siguiente imagen.

```

+ * Prac1_1: ./load_vectors.sh
edrasen@edrasen-HP-Pavilion-Laptop-15-cw1xxx:~/Documentos/Compiladores/Prac1_1$ cat load_vectors.sh
#!/bin/bash
./calc < input_val.txt
edrasen@edrasen-HP-Pavilion-Laptop-15-cw1xxx:~/Documentos/Compiladores/Prac1_1$ cat input_val.txt
[1 2 3] + [4 5 6]
[1 1 1] - [1 1 1]
[1 2 3] . [1 1 1]
[1 1 1] x [1 1 1]
|[3 3 3]|
edrasen@edrasen-HP-Pavilion-Laptop-15-cw1xxx:~/Documentos/Compiladores/Prac1_1$ ./load_vectors.sh
[ 5.000000 7.000000 9.000000 ]
[ 0.000000 0.000000 0.000000 ]
6.000000
[ 0.000000 0.000000 0.000000 ]
5.196152
edrasen@edrasen-HP-Pavilion-Laptop-15-cw1xxx:~/Documentos/Compiladores/Prac1_1$

```

3. Conclusiones

Yacc es un programa que permite generar analizadores sintácticos basándose en una gramática analítica escrita, mientras que lex genera analizadores léxicos por lo que es necesario utilizar ambos para poder crear programas que simulen el proceso de compilación de otros programas. En la presente práctica pudimos comprobar el uso y efectividad de yacc, en conjunto con lex, en sus versiones respectivas: bison y flex, para Ubuntu, a la hora de crear una calculadora para vectores que comprueba el orden y sintaxis de lo que se escriben como entrada y da un resultado de acuerdo al análisis de los operadores y números escritos.