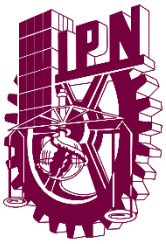


NSTITUTO POLITÉCNICO NACIONAL
ESCUELA SUPERIOR DE COMPUTO

PRACTICA 1 YACC BÁSICO
OPCIÓN CALCULADORA DE VECTORES

COMPILADORES

GRUPO: 3CM17

ALUMNO: MORA GUZMAN JOSE ANTONIO

FECHA ENTREGA: VIERNES 8 OCTUBRE 2021

DESCRIPCIÓN

Para esta practica se eligió la opción de calculadora de vectores donde se debe de escribir una especificación en yacc para evaluar expresiones que involucren operaciones con vectores como la suma, la resta, producto punto, producto punto cruz, multiplicacion por un escalar, y la magnitud.

CÓDIGO

Se muestra el archivo vector_cal.y

y la primera parte es la creación de símbolos terminales y no terminales

```
/**Creación de símbolos terminales y no terminales**/
%token<val>       NUMBER      //Símbolo terminal

%type<vector>    exp         //Simbolo no terminal
%type<vector>    vector    //Símbolo no terminal
%type<val>          number      //Símbolo no terminal

/**Jerarquía de operadores**/
//Suma y resta de vectores
%left '+' '-'
//Escalar por un vector
%left '*'
//Producto cruz y producto punto
%left 'x' '.'
```

A continuación se muestra la gramática

```
exp: vector
//Suma de vectores
  | exp '+' exp     {$$ = sumaVector($1, $3);}
//Resta de vectores
  | exp '-' exp     {$$ = restaVector($1, $3);}
//Multiplicación por un escalar caso 1
  | exp '*' NUMBER     {$$ = escalarVector($3, $1);}
//Multiplicación por un escalar caso 2
  | NUMBER '*' exp  {$$ = escalarVector($1, $3);}
//Producto cruz
  | exp 'x' exp     {$$ = productoCruz($1, $3);}
;
/**
  * number -> NUMBER

**/
number: NUMBER
  | vector '.' vector {$$ = productoPunto($1, $3);}
  | '|' vector '|' {$$ = vectorMagnitud($2);}
  ;

/**
  * 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;}
  ;
```

Y tambien el YYLEX

```
/**Código en C**/

void main(){
  yyparse();
}

int yylex (){
    int c;
    while ((c = getchar ()) == ' ' || c == '\t')
      ;
  if (c == EOF)
        return 0;
  if(isdigit(c)){
      ungetc(c, stdin);
      scanf("%lf", &yylval.val);
      return NUMBER;
  }
  return c;
}

int yyerror(char *s){
  printf("%s\n", s);
  return 0;
}
```

a continuacion se muestra el vector_cal.c

```c
#include <stdio.h>
#include "vector_cal.h"
#include <ctype.h>
#include <stdlib.h>
#include <math.h>

Vector *creaVector(int n){
   Vector *vec;
   int i;
   vec=(Vector *)malloc(sizeof(Vector));
   vec->n = n;
   vec->vec = (double *)malloc(sizeof(double)*n);
   return vec;
}

void imprimeVector(Vector *v){
   int i;
   printf("[ ");
   for(i=0; i< v->n; i++)
      printf("%f ", v->vec[i]);
   printf("]\n");
}

Vector *copiaVector(Vector *v){
   int i;
   Vector *copy=creaVector(v->n);
   for(i = 0; i< v->n; i++)
      copy->vec[i]=v->vec[i];
   return copy;
}

Vector *sumaVector(Vector *a, Vector *b){
Vector *c;
int i;
   c=creaVector(a->n);
   for(i=0; i< a->n;i++)
      c->vec[i]=a->vec[i]+b->vec[i];
   return c;
}
Vector *restaVector(Vector *a, Vector *b){
Vector *c;
int i;
   c=creaVector(a->n);
   for(i=0; i< a->n;i++)
      c->vec[i]=a->vec[i]-b->vec[i];
   return c;
}

Vector  *escalarVector(double c, Vector *v){

    Vector *r_vector = creaVector(v -> n);
    int i;
    for(i = 0; i < v -> n; i++){
        r_vector -> vec[i] = c * v->vec[i];
    }
  return r_vector;
}

Vector *productoCruz(Vector *a, Vector *b){
  Vector *r;
  r = creaVector(a -> n);
  if(a-> n == 2){
        r -> vec[0] = a -> vec[0] * b ->vec[1];
        r -> vec[1] -= a -> vec[1] * b -> vec[0];
    }
    else if(a -> n == 3){
        r -> vec[0] = a -> vec[1] * b -> vec[2]
```

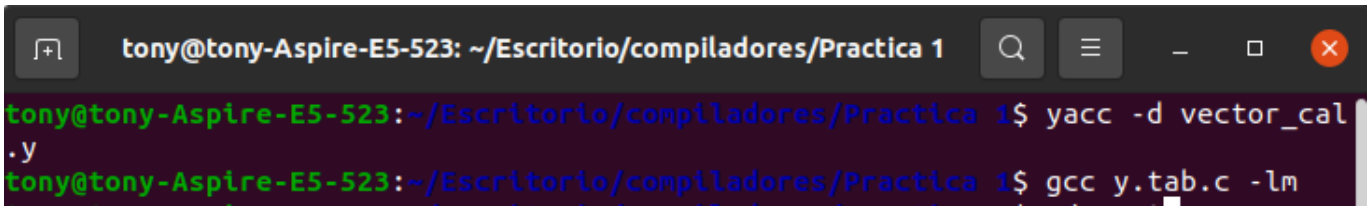
```
66      else if(a -> n == 3){
67          r -> vec[0] = a -> vec[1] * b -> vec[2]
68              - a -> vec[2] * b -> vec[1];
69
70          r -> vec[1] = a -> vec[2] * b -> vec[0]
71              - a -> vec[0] * b -> vec[2];
72
73          r -> vec[2] = a -> vec[0] * b -> vec[1]
74              - a->vec[1] * b -> vec[0];
75      }
76      return r;
77  }
78
79  double productoPunto(Vector *a, Vector *b){
80      double resultado = 0.0f;
81      int i;
82      for(i = 0; i < a->n; i++){
83          //Acumulamos el resultado del producto de cada componente
84          resultado += ( a -> vec[i] * b->vec[i] );
85      }
86      return resultado;
87  }
88
89  double vectorMagnitud(Vector *a){
90      double resultado = 0.0f;
91      int i;
92      for(i = 0; i < a->n; i++){
93          resultado += ( a -> vec[i] * a -> vec[i] );
94      }
95      resultado = sqrt(resultado);
96      return resultado;
97  }
```

Y finalmente se muestra el header

```c
struct vector {
        char name;
        int n;
        double *vec;
};
typedef struct vector Vector;
//Creación de un vector
Vector *creaVector(int n);
//Impresión de un vector
void imprimeVector(Vector *a);
//Copiado de vectores
Vector *copiaVector(Vector *a);
//Suma de vectores
Vector *sumaVector(Vector *a, Vector *b);
//Resta de vectores
Vector *restaVector(Vector *a, Vector *b);
//Multiplica un vector por un escalar
Vector *escalarVector(double c, Vector *v);
//Producto cruz entre dos vectores
Vector *productoCruz(Vector *a, Vector *b);
//Producto punto entre vectores
double productoPunto(Vector *a, Vector *b);
//Cálculo de la magnitud de un vector
double vectorMagnitud(Vector *a);
```

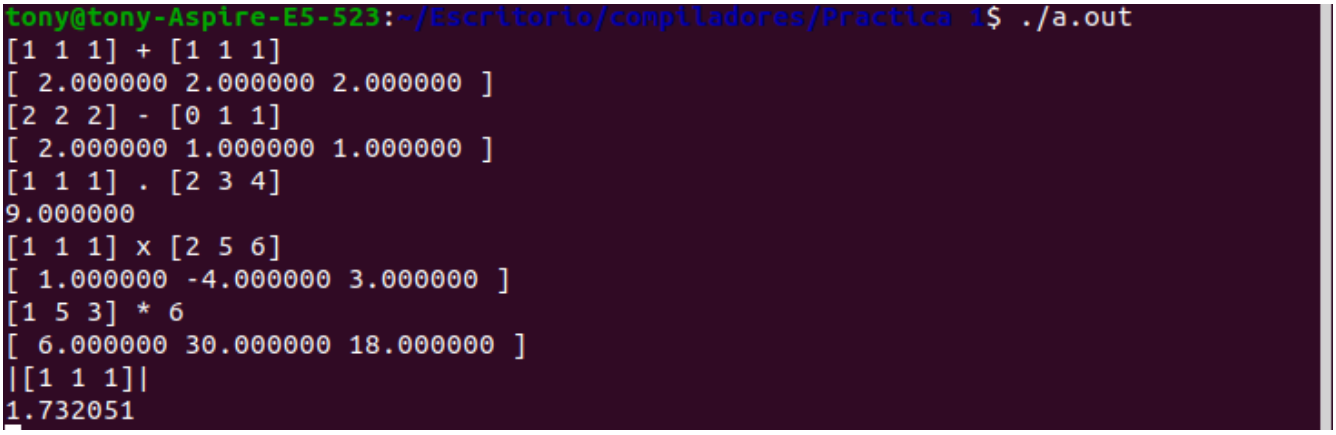
PRUEBAS DEL PROGRAMA

En la practica se utilizo yacc en el sistema operativo de ubuntu y una vez teniendo los archivos .c, .h, .y se tuvo que compilar el programa con los siguientes comandos en la terminal de ubuntu:



Después se tiene que ejecutar el archivo a.out con el siguiente comando

ya ahi estamos ejecutando la calculadora y podemos probar la calculadora con algunos ejemplos:



CONCLUSIÓN

Esta practica sirvió para empezar a practicar las cosas básicas de yacc, desde el como hacer el código hasta compilar, al principio es complicado pero una vez que vas avanzando en la practica se va haciendo mas sencillo.