NSTITUTO POLITÉCNICO NACIONAL
ESCUELA SUPERIOR DE COMPUTO

PRACTICA 6 CICLO FOR

Opción calculadora de vectores

COMPILADORES

Grupo: 3CM17

ALUMNO: MORA GUZMAN JOSE ANTONIO

FECHA ENTREGA: Miercoles 3 noviembre 2021

Descripción

En esta practica se añaden los ciclos for, que son ciclos que tienen 3 condiciones las cuales son: inicio, final y cantidad en que va aumentar la variable que controla el ciclo

Codigo

Se modifico la gramatica para añadir mas simbolos gramaticales

```
/**Creación de símbolos terminales y no terminales**/
%token<comp> NUMBER
%type<comp> escalar

%token<sym> VAR INDEF VECTOR NUMB
%type<sym> vector number

%type<inst> exp asgn

%token<sym>     PRINT WHILE IF ELSE BLTIN
%type<inst>     stmt stmtlst cond while if end

//Nuevos símbolos gramaticales para la práctica 6
%token<sym>     FOR
%type<inst>     for exprn
/**Jerarquía de operadores**/
```

```
/**Gramática**/
%%

    list:
      | list '\n'
      | list asgn '\n'    {code2(pop, STOP); return 1;}
      | list stmt '\n'    {code(STOP); return 1;}
      | list exp '\n'     {code2(print, STOP); return 1;}
      | list escalar '\n' {code2(printd, STOP); return 1;}
      | list error '\n'   {yyerror;}
      ;

    asgn: VAR '=' exp   {$$ = $3; code3(varpush, (Inst)$1, assign);}
    ;

    exp: vector             {$$ = code2(constpush, (Inst)$1);}
      | VAR                 {$$ = code3(varpush, (Inst)$1, eval);}
      | asgn
      | BLTIN '(' exp ')'   {$$ = $3; code2(bltin, (Inst)$1 -> u.ptr);}
      | exp '+' exp         {code(add);}
      | exp '-' exp         {code(sub);}
      | escalar '*' exp     {code(escalar);}
      | exp '*' escalar     {code(escalar);}
      | exp '#' exp         {code(producto_cruz);}
      | exp GT exp          {code(mayor);}
      | exp LT exp          {code(menor);}
      | exp GE exp          {code(mayorIgual);}
      | exp LE exp          {code(menorIgual);}
      | exp EQ exp          {code(igual);}
      | exp NE exp          {code(diferente);}
      | exp OR exp          {code(or);}
      | exp AND exp         {code(and);}
      | NOT exp             {$$ = $2; code(not);}
    ;

```

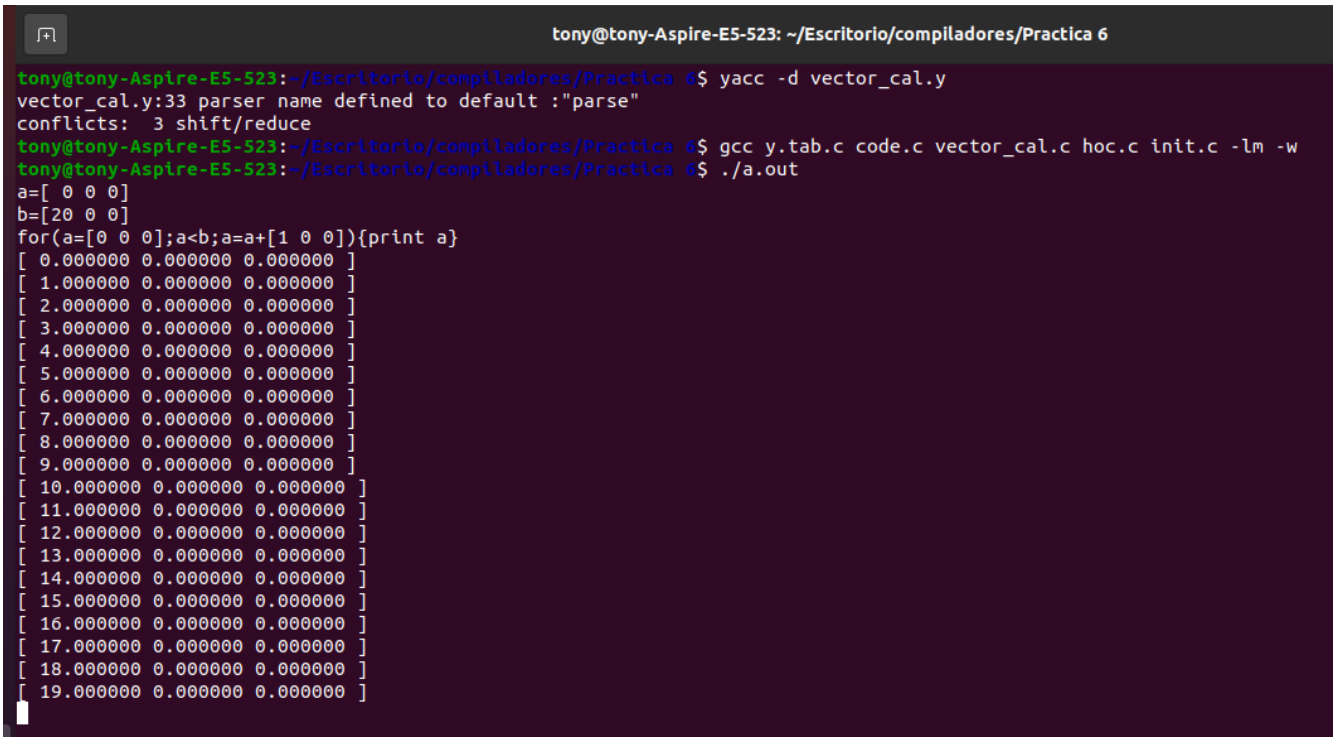
```
118     stmt: exp                                               {code(pop);}
119     |   PRINT exp                                          {code(print); $$ = $2;}
120     |   while cond stmt end                                 {($1)[1] = (Inst)$3;
121                                                              ($1)[2] = (Inst)$4;}
122
123     | if cond stmt end                                       {($1)[1] = (Inst)$3;
124                                                              ($1)[3] = (Inst)$4;}
125
126     | if cond stmt end ELSE stmt end                         {($1)[1] = (Inst)$3;
127                                                              ($1)[2] = (Inst)$6;
128                                                              ($1)[3] = (Inst)$7;}
129
130     | for '(' exprn ';' exprn ';' exprn ')' stmt end         {($1)[1] = (Inst)$5;
131                                                              ($1)[2] = (Inst)$7;
132                                                              ($1)[3] = (Inst)$9;
133                                                              ($1)[4] = (Inst)$10;}
134     | '{' stmtlst '}'                                       {$$ = $2;}
135     ;
```

```
155     //PRÁCTICA 6
156     for: FOR                        {$$ = code(forcode); code3(STOP, STOP, STOP); code(STOP);}
157     ;
158
159     exprn: exp                      {$$ = $1; code(STOP);}
160     |   '{' stmtlst '}'             {$$ = $2;}
161     ;
162
163 %%
164
```

Y tambien se añadio en code.c una funcion que ejecuta la instrucción

```
269 /********* PRÁCTICA 6 *********/
270 void forcode(){
271     Datum d;
272     Inst* savepc = pc;
273     execute(savepc + 4);
274     execute(*((Inst **)(savepc)));
275     //Se saca la instrucción
276     d = pop();
277     while(d.val){
278         execute(* ( (Inst **)(savepc + 2)));    /* Cuerpo del ciclo*/
279         execute(* ( (Inst **)(savepc + 1)));    // Último campo
280         pop();
281         execute(*((Inst **)(savepc)));           /* CONDICION */
282         d = pop();
283     }
284     pc = *((Inst **)(savepc + 3)); /*Vamos a la siguiente posicion*/
285 }
```

Pruebas del programa



Conclusiones

Al igual que la practica anterior esta al ser una continuacion se vuelve cada vez mas sencillo el trabajar en las practicas y esta vez toco agregar un ciclo que personalmente yo uso demasiado y es el ciclo for y se le puede ver muchisima mas forma a nuestro compilador puesto que ya tiene muchas funcionalidades

Link del video

https://youtu.be/oT1kFkwsZc0