Compiladores

- Práctica 06: Ciclo for -

Grupo 3CM7

Vargas Romero Erick Efraín
Prof. Tecla Parra Roberto

Instituto Politécnico Nacional
Escuela Superior de Cómputo
Juan de Dios Bátiz, nueva industrial Vallejo
07738 ciudad de México

Chapter 1

Práctica 06

1.1 Ciclo FOR

1.1.1 Descripción

En esta sextra práctica se han añadido los ciclos for.

1.1.2 Ejemplos

A continuación muestro una captura de pantalla, la cual muestra la compilación del código en yacc, y también la compilación del código que es generado en c y finalmente la ejecución del programa.

Figure 1.1: Ciclos FOR

```
[erick@erick-pc Práctica 06]$ ./a.out
a = [0 0 0]
b = [20 0 0]
for(a = [0 0 0]; a < b; a = a + [1 0 0]){ print a}
forcode[ 0.000000 0.000000 0.000000 ]
[ 1.000000 0.000000 0.000000 ]
[ 2.000000 0.000000 0.000000 ]
[ 3.000000 0.000000 0.000000 ]
[ 4.000000 0.000000 0.000000 ]
[ 5.000000 0.000000 0.000000 ]
[ 6.000000 0.000000 0.000000 ]
[ 7.000000 0.000000 0.000000 ]
[ 8.000000 0.000000 0.000000 ]
[ 9.000000 0.000000 0.000000 ]
[ 10.000000 0.000000 0.000000 ]
[ 11.000000 0.000000 0.000000 ]
[ 12.000000 0.000000 0.000000 ]
[ 13.000000 0.000000 0.000000 ]
[ 14.000000 0.000000 0.000000 ]
[ 15.000000 0.000000 0.000000 ]
[ 16.000000 0.000000 0.000000 ]
[ 17.000000 0.000000 0.000000 ]
[ 18.000000 0.000000 0.000000 ]
[ 19.000000 0.000000 0.000000 ]
```

1.1.3 Código

Nuevamente se ha modificado la gramática, se han añadido mpas símbolos gramaticales, esto se muestra a continuación

```
%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 ísmbolos gramaticales para la áprctica 6
%token<sym>     FOR
%type<inst>     for exprn
/**íJerarqua de operadores**/

//Para áprctica 3
%right '='
//Para la áprctica 5
%left OR AND
%left GT GE LT LE EQ NE
//íSmbolos gramaticales de la áprctica 1
//Suma y resta de vectores
%left '+' '-'
//Escalar por un vector
%left '*'
//Producto cruz y producto punto
%left '#' '.' '|'
//Para la áprctica 5
%left UNARYMINUS NOT

/**áGramtica**/
%%

    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
```

```
50        | BLTIN '(' exp ')'    {$$ = $3; code2(bltin, (Inst)$1 -> u.ptr);}
51        | exp '+' exp          {code(add);}
52        | exp '-' exp          {code(sub);}
53        | escalar '*' exp      {code(escalar);}
54        | exp '*' escalar      {code(escalar);}
55        | exp '#' exp          {code(producto_cruz);}
56        | exp GT exp           {code(mayor);}
57        | exp LT exp           {code(menor);}
58        | exp GE exp           {code(mayorIgual);}
59        | exp LE exp           {code(menorIgual);}
60        | exp EQ exp           {code(igual);}
61        | exp NE exp           {code(diferente);}
62        | exp OR exp           {code(or);}
63        | exp AND exp          {code(and);}
64        | NOT exp              {$$ = $2; code(not);}
65    ;
66
67    escalar: number            {code2(constpushd, (Inst)$1);}
68        | exp '.' exp          {code(producto_punto);}
69        | '|' exp '|'          {code(magnitud);}
70        ;
71
72    vector: '[' NUMBER NUMBER NUMBER ']'    {  Vector* v = creaVector(3);
73                                               v -> vec[0] = $2;
74                                               v -> vec[1] = $3;
75                                               v -> vec[2] = $4;
76                                               $$ = install("", VECTOR, v)
                                                    ;}
77        ;
78
79
80    number: NUMBER                        {$$ = installd("", NUMB, $1);}
81    ;
82
83    stmt: exp                                                  {code(pop);}
84    |   PRINT exp                                              {code(print); $$
          = $2;}
85    |   while cond stmt end                                    {($1)[1] = (Inst
          )$3;
86                                                                ($1)[2] = (Inst
                                                                    )$4;}
87
88    | if cond stmt end                                         {($1)[1] = (Inst
          )$3;
89                                                                ($1)[3] = (Inst
                                                                    )$4;}
90
91    | if cond stmt end ELSE stmt end                           {($1)[1] = (Inst
          )$3;
92                                                                ($1)[2] = (Inst
                                                                    )$6;
93                                                                ($1)[3] = (Inst
                                                                    )$7;}
```

```
94
95      | for '(' exprn ';' exprn ';' exprn ')' stmt end     {($1)[1] = (Inst
            )$5;
96                                                           ($1)[2] = (Inst
                                                                 )$7;
97                                                           ($1)[3] = (Inst
                                                                 )$9;
98                                                           ($1)[4] = (Inst
                                                                 )$10;}
99      | '{' stmtlst '}'                                  {$$ = $2;}
100     ;
101
102     cond: '(' exp ')'                       {code(STOP); $$ = $2;}
103     ;
104
105     while: WHILE                            {$$ = code3(whilecode, STOP,
            STOP);}
106     ;
107
108     if: IF                                  {$$ = code(ifcode);
109                                              code3(STOP, STOP, STOP);}
110     ;
111
112     end: /* NADA */                         {code(STOP); $$ = progp;}
113     ;
114
115     stmtlst: /* NADA */                     {$$ = progp;}
116     |   stmtlst '\n'
117     |   stmtlst stmt
118     ;
119
120     //ÁPRCTICA 6
121     for: FOR                                {$$ = code(forcode); code3(STOP,
            STOP, STOP); code(STOP);}
122     ;
123
124     exprn: exp                              {$$ = $1; code(STOP);}
125     |   '{' stmtlst '}'                     {$$ = $2;}
126     ;
127
128 %%
```

Finalmente se ha añadido en code c una función que ejecuta esta instrucción

```
1  /********* ÁPRCTICA 6 *********/
2  void forcode(){
3      Datum d;
4      Inst* savepc = pc;
5      execute(savepc + 4);
6      execute(*((Inst **)(savepc)));
7      //Se saca la óinstruccin
8      d = pop();
9      while(d.val){
10         execute(* ( (Inst **)(savepc + 2)));    /* Cuerpo del ciclo*/
```

```
11          execute(* ( (Inst **)(savepc + 1)));    // Último campo
12          pop();
13          execute(*((Inst **)(savepc)));           /* CONDICION */
14          d = pop();
15      }
16      pc = *((Inst **)(savepc + 3)); /*Vamos a la siguiente posicion*/
17  }
```