Compiladores

- Práctica 03: Variables -

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 03

1.1 Tabla de símbolos

1.1.1 Descripción

Para esta tercer práctica se ha añadido la posibilidad de definir variables en la calculadora de vectores, esto se realiza de forma simple, si el usuario teclea algo como variablea = [111] se añade esa variable a la entrada en la tabla de símbolos del programa. Para esto se ha añadido como es evidente la tabla de símbolos y se ha modificado la gramática y se han añadido un par de funciones extra.

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 java y finalmente la ejecución del programa.

Figure 1.1: Ejemplo

```
[erick@erick-pc Práctica 03]$ ./a.out
variablea = [1 1 1]
variableb = [2 2 2]
variablea

variablea = [ 1.000000 1.000000 1.000000 ]
variableb

variableb = [ 2.000000 2.000000 2.000000 ]
variablea + variableb

variablea =
variableb = [ 3.000000 3.000000 3.000000 ]
variablea - variableb

variablea =
variableb = [ -1.000000 -1.000000 -1.000000 ]
variablea . variableb

variablea =
variableb =     6.000000
variablea # variableb

variablea =
variableb = [ 0.000000 0.000000 0.000000 ]
```

1.1.3 Código

A continuación se incluye el código que ha sido modificado, empezamos con los símbolos gramaticales y los elementos que se han añadido a la pila de YACC

```
1  //óDefinicin de tipos de dato de la pila de yacc
2  %union{
3      double comp;
4      Vector* vec;
5      //ñAadida en la áprctica 3
6      Symbol* sym;
7  }
8
9  /**óCreacin de ísmbolos terminales y no terminales**/
10 %token<comp>     NUMBER      //íSmbolo terminal
11 %type<vec>       exp         //íSmbolo no terminal
12 %type<vec>       vect        //íSmbolo no terminal
13 %type<comp>      number      //íSmbolo no terminal
14 //NUEVOS ÍSMBOLOS GRAMATICALES PARA LA ÁPRCTICA 3
15 %token<sym>      VAR BLTIN   //íSmbolo terminal
16 %token<sym>      UNDEF       //íSmbolo terminal
17 %type<vec>       asgn        //íSmbolo no terminal
```

Después la gramática

```
1      asgn: VAR '=' exp       {$$ = $1 -> u.vec = $3;
2                                $1 -> type = VAR;}
3      ;
4
5      exp: vect               {$$ = $1;}
6      //La óexpresin es una variable
```

```
7         | VAR                {printf("\n%s = ", $1 -> name);
8                               if($1 -> type == UNDEF) //Verificamos si existe
                                    la variable
9                                 printf("Variable no definida %s\n", $1 -> name
                                      );
10                              $$ = $1 -> u.vec;}
11      //La óexpresin es una óasignacin
12        | asgn
```

yylex

```
1     int yylex(){
2   int c;
3   while((c = getchar()) == ' '|| c == '\t')
4   /**Salta blancos**/;
5   if(c == EOF)
6     return 0;
7   if(isdigit(c)){
8     ungetc(c, stdin);
9     scanf("%lf", &yylval.comp);
10    return NUMBER;
11  }
12
13  if(isalpha(c)){
14    Symbol* s;
15    char sbuf[200];
16    char* p = sbuf;
17    do{
18      *p++ = c;
19    } while((c = getchar()) != EOF && isalnum(c));
20
21    ungetc(c, stdin);
22    *p = '\0';
23    if((s = lookup(sbuf)) == (Symbol* )NULL)
24      s = install(sbuf, UNDEF, NULL);
25    yylval.sym = s;
26
27    if(s -> type == UNDEF)
28      return VAR;
29    else
30      return s -> type;
31  }
32  if(c == '\n')
33    lineno++;
34  return c;
35 }
```

Se ha añadido symbol.c en nuestro caso se renombró como symbol

```
1     #include "hoc.h"
2 #include "y.tab.h"
3 #include <string.h>
4 #include <stdlib.h>
5
```

```
6  static Symbol *symlist=0;     /* tabla de simbolos: lista ligada */
7
8  Symbol *lookup(char *s)     /* encontrar s en la tabla de simbolos */
9  {
10 Symbol  *sp;
11   for (sp = symlist; sp != (Symbol *)0; sp = sp->next)
12     if (strcmp(sp->name, s)== 0)
13       return sp;
14   return 0;       /* 0 ==> no se encontro */
15 }
16
17 Symbol *install(char *s,int t, Vector *vec) /* instalar s en la tabla de
       simbolos */
18 {
19   Symbol *sp;
20   char *emalloc();
21   sp = (Symbol *) emalloc(sizeof(Symbol));
22   sp->name = emalloc(strlen(s)+ 1) ; /* +1 para '\0' */
23   strcpy(sp->name, s);
24   sp->type = t;
25   sp->u.vec= vec;
26   sp->next  =  symlist;   /*  poner al frente de la lista   */
27   symlist =  sp;
28     return sp;
29 }
30
31 char  *emalloc(unsigned n)  /*   revisar el regreso desde malloc  */
32 {
33   void *p;
34   p = malloc(n);
35   return p;
36 }
```

Con su respectivo header

```
1     #include "vector_cal.h"
2  typedef struct Symbol { /* entrada de la tabla de simbolos */
3    char    *name;
4    short    type;    /* VAR, BLTIN, UNDEF */
5    union {
6      double comp;          /* si es VAR */
7      double  (*ptr)();       /* si es BLTIN */
8      Vector* vec;
9    } u;
10   struct Symbol   *next;  /* para ligarse a otro */
11 } Symbol;
12
13 Symbol *install(char *s,int t, Vector *vec), *lookup(char *s);
```