Proyecto Especial de Diseño de Compiladores Covid19: Individual

Lenguaje ForeverAlone

A continuación se describen las características generales del lenguaje que se deberá desarrollar.

La estructura general de un programa escrito en ForeverAlone es:

Programa Nombre_prog ;
<Declaración de Variables Globales>
<Definición de Funciones> **%%** Sólo hay funciones

%% Procedimiento Principal .... comentario
principal()
{
<Estatutos>
}

*** Las secciones en itálicas son opcionales (pudiera o no venir).**
*** Las palabras y símbolos en bold son Reservadas y el %% indica comentario.**

Para la <u>Declaración de Variables</u>: *(hay globales y locales)*
sintaxis:

var %%Palabra reservada
tipo **:** lista_ids**;**
<*tipo : lista_ids;* **>** *etc...*

donde
tipo =(solo tiene) **int, float y char**.
lista_ids = identificadores separados por comas, tienen máximo una dimensión [N] de 0 a N-1.
Ej: **int** : id1[cte-entera], id2, id3;
con lo que se definen tres variables enteras, donde la primera tiene una dimensión de tamaño N.

Para la <u>Declaración de Funciones</u>: *(se pueden definir 0 ó más funciones)*
sintaxis:
funcion <*tipo-retorno*> nombre_módulo **(** <*Parámetros*> **)** ;
<Declaración de Variables Locales>
{
<Estatutos> ***%% El lenguaje soporta llamadas recursivas*.**
}
Los parámetros siguen la sintaxis de la declaración de <u>variables simples (no dimensionadas)</u> y únicamente son de entrada.
tipo-retorno puede ser de cualquier tipo soportado o bien void (si no regresa valor)

Para los Estatutos:

La sintaxis básica de cada uno de los estatutos en el lenguaje **ForeverAlone** es:

ASIGNACION

Id<*dimension*> **=** Expresión**;**

A un identificador (que pudiera ser simple o ser una casilla de un elemento dimensionado) se le asigna el valor de una expresión. Cabe aclarar que siempre, a excepción de en la declaración, las Dimensiones son Expresiones aritméticas.

Id<*dimension*> **=** Nombre_Módulo**(**(<*param1*>, (<*param2*>,…**);** %%siempre los parámetros actuales son Expresiones

A un identificador, se le asigna el valor que regresa una función.

O bien, pudiera ser algo como: Id<*dimension*> = Nombre_Módulo(<*param1*>,..) + Id<*dimension*> – cte

A un identificador se le puede asignar el resultado de una expresión en donde se invoca a una función.

LLAMADA A UNA FUNCIÓN VOID

Nombre_Módulo (<*param1*>,..**);**

Se manda llamar una función que no regresa valor (caso de funciones *void*).

RETORNO DE UNA FUNCIÓN

regresa(exp **)** %%Este estatuto va dentro de las funciones e indica el valor de retorno (si no es void)

LECTURA

lee (id<*dimension*> , *id*<*dimension*> >....**);**

Se puede leer uno ó más identificadores (con o sin dimensiones) separados por comas.

ESCRITURA

escribe ("letrero" ó expresión<*, "letrero" ó expresión*>....**);**

Se pueden escribir letreros y/ó resultados de expresiones separadas por comas.

ESTATUTO DE DECISION (puede o no venir un "sino")

si (expresión**) entonces** **%%** típica decisión doble
{ <*Estatutos*>*;* **}**
<***sino***
{ <*Estatutos*>*;* ***}***>

ESTATUTOS DE REPETICION

CONDICIONAL

mientras (expresión**) haz** **%%** Repite los estatutos mientras la expresión sea verdadera
{ <*Estatutos*>*;* **}**

NO-CONDICIONAL

desde Id<*dimensiones*>**=** exp **hasta** exp **hacer**
{ <*Estatutos*>*;* **}** **%%** Repite desde N hasta M brincando de 1 en 1

EXPRESIONES

Las expresiones en **ForeverAlone** son las tradicionales (como en C y en Java). Existen los operadores aritméticos, lógicos y relacionales: **+, - , *, / , &**(and) **, |** (or)**, < , >, ==, etc.** Se manejan las prioridades tradicionales, se pueden emplear paréntesis para alterarla.

En **ForeverAlone** existen identificadores, palabras reservadas, constantes enteras, constantes flotantes, constantes char y constantes string (letreros).

%% Se anexa ejemplo

```
programa foreveralone;
var
   int i, j, p;
   int  Arreglo[10] , OtroArreglo[10] ;
   float  valor;

funcion int fact (int j)
var int i;
   { i= j + (p - j*2+j) ;
   si ( j == 1) entonces
      { regresa ( j ); }
   sino
      { regresa ( j * fact( j-1); }
 }

funcion void inicia (int y)
var int x;
 { x= 0;
  mientras ( x < 11) haz
     {Arreglo[x] = y * x;
      x = x+1;}
 }

principal ( )
{   lee (p) ;  j =p *2;
    inicia ( p * j  -  5) ;
    desde i=0 hasta 9 hacer
            { Arreglo [ i ] = Arreglo [ i ] * fact (Arreglo [ i ] - p) ; }
    desde i=0 hasta 9 hacer
            { OtroArreglo [ i ] = Arreglo [ i ]  - p) ; }
    mientras ( i >= 0)
            {  escribe ("resultado" , Arreglo [ i ] ,   fact ( i +2) * valor ) ;
               i = i - 1;
            }
    mientras ( i < 10)
            {  escribe ("Otros datos" , OtroArreglo [ i ] , p, i + OtroArreglo[i] ) ;
               i = i + 1;
            }
}
```