Proyecto Especial de Diseño de Compiladores Covid19 AD20: Individual

Lenguaje MeMySelf

A continuación se describen las características generales del lenguaje que se deberá desarrollar. Es un lenguaje orientado a jóvenes que buscan aprender los fundamentos de la programación gráfica.

La estructura general de un programa escrito en MeMySelf es:

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

%% Procedimiento Principal .... comentario
main()
{
<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.
Ej: **int** : id1, id2, id3; %%con lo que se definen tres variables enteras.

Para la <u>Declaración de Funciones</u>: *(se pueden definir 0 ó más funciones)*
sintaxis:
<tipo-retorno> **module** 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</u> y únicamente son de entrada.
tipo-retorno puede ser de cualquier tipo soportado o bien void (si no regresa valor)

Para los <u>Estatutos</u>:

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

ASIGNACION
Id **=** Expresión**;**

A un identificador se le asigna el valor de una expresión.

Id = 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 = Nombre_Módulo(*<param1>*,..) + Id – cte

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

LLAMADA A UN MÓDULO VOID

Nombre_Módulo (<*param1*>,..**);**

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

RETORNO DE UNA FUNCIÓN

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

LECTURA

read (id*, id*....**);**

Se puede leer uno ó más identificadores separados por comas.

ESCRITURA

write ("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")

if (expresión**) then** **%%** típica decisión doble
{ <*Estatutos*>*;* **}**
<***else***
{ <*Estatutos*>*;* ***}***>

ESTATUTOS DE REPETICION

CONDICIONAL

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

NO-CONDICIONAL

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

EXPRESIONES

Las expresiones en **MeMySelf** 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 **MeMySelf** existen identificadores, palabras reservadas, constantes enteras, constantes flotantes, constantes char y constantes string (letreros).

FUNCINES ESPECIALES

LINE, POINT, CIRCLE, ARC, PENUP, PENDOWN, COLOR, SIZE, CLEAR, etc

Cada función especial tendrá la parametrización apropiada, ej: POINT(x,y), CIRCLE (RADIO), etc..
Line, Circle, Arc: Pintan una línea, un círculo y un arco respectivamente.
PENUP, PENDOWN levanta la pluma (no pintar), baja la pluma (pintar).
COLOR, SIZE: cambia el color y el grosor al pintar.
Etc.

%% Se anexa ejemplo

```
program MeMyself;
var
   int i, j, p;
   float  valor;

int module fact (int j)
var int i;
   { i= j + (p - j*2+j) ;
   if ( j == 1) then
       { return ( j ); }
  else
       { return ( j * fact( j-1); }
 }

void module  pinta (int y)
var int x;
 { x= 1;
  while ( x < 11) do
     {Circle(y + x*5);;
     Color(x + 10);
     Size (10 - x);
      x = x+1;}
 }

main ( )
 {   read (p) ;  j =p *2;  Point( 0, 0);
    i =  fact ( p) ;
     from i=0 to 9 do
               { pinta(i * j) ; }
     while ( i < 10) do
               {  write ("HelloWorld", fact(i)) ;
                   i = i + 1;
               }
 }
```