

Proyecto Especial para Diseño de Compiladores AD23: Individual

Lenguaje MyR language

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, a través del manejo y manipulación de conjuntos de datos simples para realizar análisis estadístico básico, o bien a través del manejo de elementos gráficos sencillos.

La estructura general de un programa escrito en MyR 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 (pudieran o no venir).
- * Las palabras y símbolos en **bold** son Reservadas y el %% indica comentario.

Para la Declaración de Variables: (*hay globales y locales*)

sintaxis:

```
VARs %%Palabra reservada  
    tipo: lista_ids;  
<tipo: lista_ids; > etc...
```

donde

tipo =(solo puede ser) **int, flot y char**.

lista_ids = identificadores separados por comas, pudiendo definir UNA dimensión

(id[N] donde N es un número entero y se convierte en un arreglo indexable de 1 a N).

Ej: **int:** id1, id2, id3[6]; %%con lo que se definen dos variables y un vector de 6 enteros (1 a 6).

Para la Declaración de Funciones: (*se pueden definir 0 ó más funciones*)

sintaxis:

```
function <tipo-retorno> nombre_módulo ( <Parámetros> ) ;  
    <Declaración de Variables Locales>  
    {  
        <Estatutos>                               %% El lenguaje soporta llamadas recursivas.  
    }
```

Los parámetros siguen la sintaxis de la declaración de variables simples 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 **MyR** es:

ASIGNACION

Id = Expresión; ó Id[exp] = Expresión;

A un identificador (o a una casilla) se le asigna el valor de una expresión.

Id = Nombre_Función(<arg1>, (<arg2>, ...); %%siempre los argumentos (parámetros actuales) son Expresiones

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

O bien, pudiera ser algo como: $Id = \text{Nombre_Función}(<arg1>, \dots) + Id2[i+2] - cte$

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_Función (<arg1>,...);

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[j-3],...);

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

```
while (expresión) do %% 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 **MyR** 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 **MyR** existen identificadores, palabras reservadas, constantes enteras, constantes flotantes, constantes char y constantes string (letreros).

FUNCIONES ESPECIALES

OpciónA: *Funciones como: Media, Moda, Varianza, Regresión Simple, PlotXY,, etc*
(matemático)

Cada función especial tendrá la parametrización apropiada, ej: Media(Arreglo), etc..

OpciónB: *LINEA, PUNTO, CIRCULO, ARCO, PENUP, PENDOWN, COLOR, GROSOR, LIMPIAR, etc*
(gráfico)

Cada función especial tendrá la parametrización apropiada, ej: PUNTO(x,y), CIRCULO (RADIO), etc..

LINEA, CIRCULO, ARCO: Pintan una línea, un círculo y un arco respectivamente.

PENUP, PENDOWN levanta la pluma (no pintar), baja la pluma (pintar).

COLOR, GROSOR: cambia el color y el grosor al pintar.

Etc.

%% Se anexan ejemplos

OPCIÓN A:

```

Program MyRlike;
VARs
  int: i, j, p , arreglo[10];
  float: valor;

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

function void calcula (int y)
VARs int x;
{ x= 1;
  while ( x < 11) do
    { y = y * arreglo[x]
      x = x+1;}
  write ( arreglo[x] )
}
write ("acumulado", y);
}

principal ( )
{  read (p) ; j =p *2;
  i = fact ( p) ;
  for i=1 to 10 do
    { arreglo[i] = p + i ; }
  p = Media (arreglo);
  while ( i > 0) do
  { calcula (p-i)
    j = fact(arreglo[i]);
    write( j , i);
    i = i + 1;
  }
}

```

OPCIÓN B:

```

program MyRGraphic;
VARs
  int i, j, p;
  float valor;

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

function void pinta (int y)
VARs int x;
{ x= 1;
  while ( x < 11) do
    {Circulo(y + x*5);;
    Color(x + 10);
    Grosor (10 - x);
    x = x+1;}
}

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

```