**Proyecto Individual para Diseño de Compiladores Agosto-Diciembre 2022**

**Lenguaje Estadístico MIR**

Las características generales del lenguaje a desarrollar serán las siguientes, teniendo un enfoque a un lenguaje sencillo de fundamentos de programación, manejando datos simples con ciertas funciones estáticas de estadística y demás operaciones básicas de un lenguaje.

La estructura general es la siguiente:

**Program** Nom\_prog;

<Declaración de variables Globales> *Opcionales*

<Declaración de Funciones> *Opcionales, y solamente hay funciones*

**main**()

**{**

<Estatutos> *Opcionales*

**}**

Las palabras y símbolos en bold son reservadas, con comentarios en itálicas.

Sintaxis general para declaración de variables:

**VARS**

tipo: lista\_ids;

<tipo: lista\_ids; > ….

Donde los tipos de variables solo pueden ser int, flot y char; y las lista de ids son los identificadores separados por comas, pudiendo ser definidas como vectores, es decir, con una sola dimensión: id[N], donde N es un numero entero y se convierte a un arreglo indexable de 1 a N).

Ejemplo: **int:** id1, id2, id3[5]; *Dos variables y un vector de 5 casillas*.

Sintaxis general para la declaración de Funciones (siendo posibles 0 o más funciones):

**function** <tipo-retorno> nombre\_modulo (<Parámetros>);

<Declaración de Variables Locales>

**{**

<Estatutos> *(Llamadas recursivas son posibles)*

**}**

Los parámetros siguen la sintaxis de la declaración de variables simples y solo son de entrada. Tipo retorno son los tres tipos soportados, o void si la función no regresa valor.

Descripción de Estatutos

La sintaxis descrita de los estatutos soportados para el lenguaje Mir son los siguientes:

Asignacion:

ID **=** Expresión; alternativamente ID[Exp] = Expresión;

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

ID = Nombre\_Funcion((<arg1>,<arg2>,…..); *siempre los argumentos(parámetros actuales son expresiones.*

A un identificador se le asigna el valor de retorno de una función

También se incluye algo como:

ID= Nombre\_Funcion(<args>….)+ ID2[i+2]-cte

Donde se le puede asignar el resultado de una expresión en donde se invoca una función.

Llamada a funciones void

Nombre\_Funcion(<args>….);

Se llama una función que no regresa valor.

Retorno de una función

**return**(exp); *Estatuto especial que solamente se encuentra adentro funciones e indica el valor de retorno, si no es void.*

Lectura

**read** (id, id[j-3],….);

Se puede leer uno o mas identificadores, separados por comas.

Escritura

**write**(<“letrero” o expresión>, <“letrero” o expresión>…);

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

Estatus de Decisión

**if** (expresión) **then**

**{** <Estatutos>; **}**

<else> *Opcional el else*

**{** <Estatuos>; **}**

Estatutos de Repetición Condicionales y No-Condicionales

Condicional

**while** (expresión) **do** *Repite los estatutos mientras la expresión es verdadera*

**{** <Estatutos>; **}**

No-Condicional

**for** ID<dimensiones>=exp **to** exp **do** *Repite desde i hasta j brincando 1 en 1*

**{** <Estatutos>; **}**

Expresiones

Las expresiones en MIR son las tradicionales del lenguaje C y Java, con operadores aritméticos, lógicos y relacionales: +,-,\*,/, && o and, || o or, <,>,<=,>=, == y demás. Se manejan las prioridades tradicionales, con el uso de paréntesis para alteración de ese orden.

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

Funciones especiales Simples

El uso de funciones directas especiales con palabras reservadas para dar ciertas demonstraciones de estadística simple:

Media

Moda

Varianza

PlotXY

Con parámetros posiblemente dinámicos, ejemplo Media(Arreglo), etcétera.

Ejemplo de Programa:

Program MIR;

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);

}

main()

{

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;

}

}