Proyecto Especial de Diseño de Compiladores COVID19: En Pareja

Lenguaje Covid19--

A continuación, se describen las características generales del lenguaje imperativo que se deberá desarrollar durante el curso.

La estructura general de un programa escrito en Covid19—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 =(tipos atómicos) **int, float, string y char**. Además tiene un tipo No-atómico **Dataframe**
lista_ids = identificadores separados por comas,
pudieran tener un máximo dos dimensiones [N] [M] de 0 a N-1 y 0 a M-1.

Ej: **int** : id1, id2[4], id3[2][3];
Dataframe SourceData;
con lo que se define una variable entera (id1), un vector de 4 enteros (id2[0]... id[3]) y una matriz de 6 enteros (id3[0][0] ... id3[1][2]). %%la declaración de la dimensión siempre es una CTE-Entera
la variable SourceData servirá para cargar de un archivo datos en formato Matricial, no se especifica tamaño porque éste depende de la información contenida en un archivo en tiepo de ejecución.

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 simple soportado o bien void (si no regresa valor)

Para los Estatutos:

La sintaxis básica de cada uno de los estatutos en el lenguaje Covid19-- es:

ASIGNACION

Id<*dimensiones*> **=** 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<*dimensiones*> **=** 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<*dimensiones*> = Nombre_Módulo(<*param1*>,..) + Id<*dimensiones*> – 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<*dimensiones*> , *id*<*dimensiones*> >....**);**

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.

CARGA_DATOS

CargaArchivo (id , “ruta de acceso al archivo”, #deVariables, #deRenglones**)**

Se cargarán los datos contenidos en el archivo sobre una variable de tipo Dataframe. La carga contabiliza la cantidad de variables presentes (columnas) y la cantidad de datos para c/var (renglones).

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 Covid19-- 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 Covid19-- existen identificadores, palabras reservadas, constantes enteras, constantes flotantes, constantes char y constantes string (letreros).

CARACTERISTICA ESPECIAL

Esta versión de proyecto permitirá trabajar y manipular (estadísticamente) los datos contenidos en un archivo texto. La carga de dicho archivo permitirá reconocer/interpretar cada encabezado del archivo como una variable (ej: File1.score). Se debe incluir una función que genere un arreglo con las "variables" detectadas en el archivo (por simplicidad, pueden suponer que habría un máximo de 100 variables y que cada variable tendría máximo 1000 datos -hacia abajo-). Adicionalmente, el lenguaje debe proporcionar un conjunto de funciones estadísticas básicas que se pudieran aplicar sobre ese dataframe (ejemplo: calcular medias, modas, varianzas para alguna de las variables, calcular la distribución presente en los datos, etc). También debe considerar alguna(s) función que permita correlacionar las variables presentes en el archivo. Debe incluir al menos un par de funciones que permitan graficar o visualizar los datos, tendencia, distribución, etc. ***%%Sean creativos con sus funciones "extras"***

%% Se anexa Ejemplo

```
programa Covid19;
var
   int i, j, p, maxVariables, maxRenglones;
   int valor,  Arreglo[10] , OtroArreglo[10] ;
   dataframe DatosCovid;
   string  VarCodiv[100];

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;                                   %%solo se ejemplifican algunas de las posibles funciones especiales
    inicia ( p * j  -  5) ;
    desde i=0 hasta 9 hacer
            { Arreglo [ i ] = Arreglo [ i ] * fact (Arreglo [ i ] - p) ; }
    cargaArchivo (DatosCovid , "C://Documentos/Covid.txt , maxVariables, maxRenglones );
    Variables (DatosCodiv, VarCovid, maxVariables);          %%genera el arreglo de Variables de acuerdo al número máximo contabilizado al cargar archivo
    desde k=0 hasta MaxVariables hacer
            { valor = Media (DatosCovid, Variables[k] ) ;
              escribe ("Media para la Variable" , Variables[k] ,   valor ) ; }      %%calcula y despliega las medias para cada una de las Variables
   escribe ("Indice de correlacion entre Variable 1 y Variable 5 de la muestra" ,  Correlaciona (DatosCovid, Variables[1] , Variables[5]) ) ;
    mientras ( i >= 0)
            {  escribe ("resultado" , Arreglo [ i ] ,   fact ( i +2) * valor ) ;
               i = i - 1;
            }
}
```