Estructuras de datos: Arreglos (array)

Estructura de datos

Hasta ahora, para hacer referencia a un dato utilizábamos una variable. El problema se plantea cuando tenemos gran cantidad de datos que guardan entre sí una relación. No podemos utilizar una variable para cada dato.

Para resolver estas dificultades se agrupan los datos en un mismo conjunto, estos conjuntos reciben el nombre de **estructura de datos**.

Arreglos o arrays

Un array (o arreglo) es una estructura de datos con elementos homogéneos, del mismo tipo, numérico o alfanumérico, reconocidos por un nombre en común. Para referirnos a cada elemento del array usaremos un índice (empezamos a contar por 0).

Declaración de arrays

Para declarar un array tenemos que ejecutar dos instrucción:

1. En primer lugar debemos declara el tipo de datos de la variable, con `Definir`.
2. Debemos indicar el número de elementos que va a tener el array, para ello utilizamos la instrucción `Dimension`:

   ```
   Dimension <identificador> [<maxl>,...,<maxN>];
   ```

   Esta instrucción define un arreglo con el nombre indicado en y N dimensiones. Los N parámetros indican la cantidad de dimensiones y el valor máximo de cada una de ellas. La cantidad de dimensiones puede ser una o más, y la máxima cantidad de elementos debe ser una expresión numérica positiva.

Por ejemplo definimos un array de una dimensión (también llamado **vector**) de 10 elementos enteros.

```
Definir vector como Entero;
Dimension vector[10];
```

Otro ejemplo, definir una array de dos dimensiones (también llamado **matriz** o **tabla**) de 3 filas y cuatro columnas de cadenas.

```
Definir tabla como Cadenas;
Dimension tabla[3,4];
```

Para acceder a un elemento de un array se utiliza un índice. El primer elemento está en la posición 1.

Para asignar un valor a un elemento del vector:

```
vector[1]<-10;
```

Para mostrar el primer elemento del vector:

```
Escribir vector[1];
```

Otro ejemplo asignamos y mostramos el segundo elemento de la segunda fila de la tabla:

```
tabla[2,2] <- "Hola";
Escribir tabla[2,2];
```

Arreglos unidimensionales: Vectores

Un **vector** es una array unidimensional. Para declarar un vector de 10 enteros:

```
Definir vector como Entero;
Dimension vector[10];
```

Para acceder a cada uno de los elementos del vector utilizamos un índice. el primer elemento se accede con el índice 0. Podemos trabajar individualmente con cada uno de los elementos:

```
vector[1]<-10;
Escribir vector[0];
```

El acceso a un elemento que no existe producirá un error, por ejemplo:

```
vector[10]<-10;
```

Recorrido de un vector

Vamos a inicializar todos los elementos de un vector. Para ello vamos a **recorrer** el vector e inicializar cada elemento con un valor ,por ejemplo lo vamos a inicializar a 0. Para recorrer un vector utilizamos un bucle `Para`:

```
Para i<-1 hasta 10 Hacer
        array[i]<-0;
FinPara
```

Podríamos recorrer el vector para mostrar el valor de los elementos:

```
Para i<-1 hasta 10 Hacer
        Escribir array[i];
FinPara
```

Ejemplo

Inicializar un vector de 5 cadenas a partir de los datos pedidos por teclado y posterior mostrarlos en pantalla en mayúsculas.

```
Proceso VectorCadenas
        Definir i Como Entero;
        Definir vector Como Caracter;
        Dimension vector[5];
        Para i<-1 hasta 5 Hacer
                Escribir Sin Saltar "Dime la cadena número ",i+1,":";
                Leer vector[i];
        FinPara
        Escribir "Las cadenas en mayúsculas";
```

```
        Para i<-1 hasta 5 Hacer
                Escribir Sin Saltar Mayusculas(vector[i])," ";
        FinPara
FinProceso
```

Arreglos multidimensionales: Tablas

Una **tabla** en un array bidimensional. La primera dimensión indica el número de filas y el segundo el número de columnas.

```
Definir tabla como Entero;
Dimension tabla[3,4];
```

Hemos definido una tabla de enteras con 3 filas y 4 columnas, por tanto tenemos 12 elementos.

Para acceder a cada uno de lo elemento tenemos que indicar la fila y la columna en la que se encuentra, siempre empezando por el 0. Por ejemplo para inicializar el elemento que esta en la primera fila y la segunda columna sería:

```
tabla[1,2] <- 10;
```

El acceso a un elemento que no existe producirá un error.

Recorrido de una tabla

Para recorrer todos los elementos de una tabla necesitamos utilizar dos bucles anidados. Normalmente el exterior nos va a permitir recorrer las filas y el interior las columnas. Por ejemplo para inicializar todos los elementos a 0, quedaría:

```
Para filas<-1 hasta 3 Hacer
        Para columnas<-1 hasta 4 Hacer
                tabla[filas,columnas]<-0;
        FinPara
FinPara
```

De forma similar podríamos recorrer la tabla para mostrar los elementos:

```
Para filas<-1 hasta 3 Hacer
        Para columnas<-1 hasta 4 Hacer
                Escribir tabla[filas,columnas];
        FinPara
FinPara
```

Ejemplo

Inicializar una tabla con los números del 1 al 5, sus cuadrados y sus cubos. Por lo tanto tenemos que definir una tabla con 5 filas y 3 columnas. Muestra los datos:

```
Proceso CuadradoCubos
        Definir tabla Como Entero;
        Definir filas,columnas Como Entero;
        Dimension tabla[5,3];
        Para filas<-1 hasta 5 Hacer
                tabla[filas,1]<-filas+1;
                tabla[filas,2]<-(filas+1)^2;
                tabla[filas,3]<-(filas+1)^3;
```

```
        FinPara

        Para filas<-1 hasta 5 Hacer
                Para columnas<-1 hasta 3 Hacer
                        Escribir Sin Saltar tabla[filas,columnas]," ";
                FinPara
                Escribir "";
        FinPara
FinProceso
```

Arrays multidimensionales

Los arrays pueden ter las dimensiones que deseemos, por ejemplo podemos tener una array de tres dimensiones:

```
Definir tabla como Entero;
Dimension tabla[4,4,4];
```

Y podríamos inicializar el primer elemento como:

```
tabla[1,1,1]<-10;
```

Necesitaríamos tres bucles para recorrer un array de tres dimensiones:

```
Para i<-1 hasta 3 Hacer
        Para j<-1 hasta 3 Hacer
                Para k<-1 hasta 3 Hacer
                        tabla[i,j,k]<-0;
                FinPara
        FinPara
FinPara
```