TEMA 6: ESTRUCTURAS DE DATOS (Arrays).

CONTENIDO:

6.1.- Introducción a las estructuras 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. Para cada uno de estos datos se debería utilizar una variable distinta, lo que acarrea una gran laboriosidad a la hora de construir el programa, unida a la cantidad de variables a usar.

Para resolver estas dificultades se agrupan los datos en un mismo conjunto, estos conjuntos reciben el nombre de **estructura de datos**. Podemos decir que una estructura de datos es una colección de datos que pueden ser caracterizados por su organización y por las operaciones que se definen en ella.

6.1.1.- Tipos de datos más frecuentes:

- Datos Simples
  - Estándar: pueden ser: entero, real , carácter, lógico.
  - Definidos por el programador: subrango, enumerativo.
- Datos Estructurados
  - Estáticos: tablas, registros, archivo, conjunto, cadena.
  - Dinámico: listas (pilas o colas), listas enlazadas, arboles, grafos.

6.2.- Arrays unidimensionales: los vectores.

Un array es una estructura interna de datos con elementos homogéneos, del mismo tipo, numérico o alfanumérico, reconocidos por un nombre en común, que residen en la memoria del ordenador.

Cuando dicho array es de una sola dimensión, se denomina vector.

Para hacer referencia a los elementos de un vector se utiliza lo que se denomina un indice. Se suele simbolizar por las letras I, J... sucesivamente, como veremos más adelante en los ejemplos.

Ej: I← 7
V(I)← "HOLA"

En este ejemplo, asignamos la cadena "HOLA" en el vector **V** en la posición marcada por el índice I que es 7.

Ej: Ejemplo de un vector de caracteres

V(1)	**V(2)**	**V(3)**	**V(4)**
"esto"	"es"	"un"	"vector"

Ej: Vamos ahora a realizar un pseudocódigo que lea una cadena y que la introduzca en la posición 1 de la tabla, lea otra cadena y la introduzca en la 2 y así sucesivamente hasta 10.

Para I← 1 hasta 10 **hacer**
Leer(cadena)
V(I)← cadena
Fin_para

Como puede comprobarse es muy sencillo, se nos pedirá por pantalla una cadena, la escribiremos y se almacenará en la posición que vaya indicando el indice que recorre el vector que no es otro que I.

Este es sólo un ejemplo de las operaciones que podemos realizar con vectores, vamos a comentar ahora qué otras operaciones podemos realizar

6.3.- Operaciones con vectores.

- Asignación
  Ejemplo:

```
para i ← 1 hasta 10 hacer
        DAI (i) ←— 8
fin_para
```

- Lectura / escritura

  Ejemplo:

```
leer ( DAI ( I ))
escribir ( DAI ( k ))
```

- Acceso o recorrido de un vector

Comentario: Tabla va ser la palabra reservada que vamos a utilizar para definir un tipo Tabla, al igual que tenemos tipo entero, carácter, etc.

Ejemplo: Lectura de 20 valores enteros de un vector denominado lista.

```
Programa Leer_vector
  Entorno
    tipo
        vector: tabla (1..20) de entero
    var
        lista: vector
        i: entero
  Inicio
    para i ← 1 hasta 20 hacer
        escribir ("Introducir elemento: " , i)
        leer (lista(i))
    fin_para
  fin
```

Ejemplo: Hacer una tabla de puntuación realizando lo siguiente:

a. Lectura de la tabla de 40 elementos.
b. Calculo de la suma de los valores de la tabla.
c. Calculo de la media de los valores de la tabla.

```
Programa Notas
Entorno
        const
            limite = 40
        tipo
            vector: tabla (1…limite) Real
        var
            Puntuación: vector
            Total, Media: real
            I: entero
    Inicio
```

```
        Total ←—— 0
        Media ←——  0
        para I ←——  1 hasta limite hacer
            escribir (“Introduzca puntuación: “, I)
            leer (Puntuación (I))
            Total ←—— Total + Puntuación (I )
        fin_para
        Media ←—— Total / limite
        escribir( “El total de los 40 nº es:  “, Total)
        escribir (“La media de los 40 nº es: “; Media)
fin
```

Ejemplo: Realizar un algoritmo que permita realizar el cuadrado de los cien primeros números enteros y a continuación escribir la tabla que contenga los 100 nº cuadrados.

```
Programa Cuadrado
Entorno
    const
        limite =100
    tipo
        vector: tabla (1…limite) de entero
    var
        numero: vector
        I, cuadrado: entero

Inicio
    para I ←——  1 hasta limite hacer
        cuadrado ←— I * I
        numero(I) ←— cuadrado
    fin_para
escribir (“El cuadrado de los cien primeros numeros es: “)
    para I ←——  1 hasta limite hacer
        escribir (numero (I))
    fin_para
fin
```

Ejemplo: Programa que lee las calificaciones de un alumno con 10 asignaturas, las almacena en un vector y las calcule y saca la media.

```
Programa Notas
Entorno
    const
        limite = 10
    tipo
        vector: tabla (1…limite)de real
    var
```

```
                    nota: vector
                    media, total: real
                    I: entero
Inicio
        total ←——— 0
        para I ←——— 0 hasta limite hacer
                escribir (“Introducir numero: “,I)
                leer (nota (I))
        fin_para
        para I ←——— hasta limite hacer
                total ←——— total + nota (I)
        fin_para
                media ←——— total / limite
                escribir (“Nota media:  “,media)
    fin
```

Ejemplo: Programa que lee una secuencia de 50 nº enteros y los almacena en un vector y luego lo saca en orden inverso al de entrada

```
    Programa Inverso
Entorno
        const
                limite = 50
        tipo
                numero: tabla (1..limite) de real
        var
                I : entero
    Inicio
        para I ←——— 1 hasta limite hacer
                escribir (“Introducir numero: “, I)
                leer (numero (I))
        fin_para
        para I ←——— limite hasta 1 con incremento
-1 hacer
                escribir (numero(I))
        fin_para
  fin
```

6.4.- Arrays Bidimensionales (Matrices o tablas).

Un vector es una tabla de una sola dimensión, como ya hemos visto. Vamos a estudiar ahora las tablas de dos dimensiones (Fila, Columna).

Para direccionarla lo haremos de la siguiente forma:

Nombre(Fila,Columna)

Ej: T(2,3) . Tabla de 2 filas y 3 columnas.

Cómo se dimensiona una tabla:

E*jemplo:*

```
Programa dimensionar
Entorno
    const
        M=30
        N=20
    tipo
        matriz: tabla [1…M,1…N] de real
    var
        alumnos: matriz
Inicio
    .
    .
    .
fin
```

Tratamiento de una tabla bidimensional o matriz

Se hace con el anidamiento de dos bucles **Para**, el primero para las filas (indice I) y el segundo para las columnas (indice J). Con esto recorremos nuestra tabla por filas ya que el bucle externo es el de las filas, si queremos recorrer por columnas el bucle externo irá con el indice J.

Ej 1: Tenemos la matriz M, con 2 filas y 3 columnas. Vamos a tratarla o recorrerla por filas.

```
Para I←1 hasta 2 hacer
    Para J←1 hasta 3 hacer
        Operar(M(I,J)) (*siendo operar cualquier operación*)
    Fin_para
Fin_para
```

Fila 1 → Recorremos Columnas 1,2,3

Fila 2 → Recorremos Columnas 1,2,3

I=Fila, J=Columna

Ej 2: Tenemos la matriz M, con 2 filas y 3 columnas. Vamos a tratarla o recorrerla por columnas.

```
Para J←1 hasta 3 hacer
    Para I←1 hasta 2 hacer
        Operar(M(I,J)) (*siendo operar cualquier operación*)
    Fin_para
Fin_para
```

Columna 1 → Recorremos Filas 1,2

Columna 2 → Recorremos Filas 1,2

Columna 3 → Recorremos Filas 1,2

I=Fila, J=Columna

*Ejemplo: Tenemos una tabla de 2 dimensiones 50*20 de tipo real y vamos a calcular la suma de todos sus elementos*

```
Programa suma
Entorno
    const
        M=50
        N=20
    tipo
        matriz: tabla[1…M,1…N] de real
    var
        numero: matriz
        I, J: entero
        suma: real
    Inicio
        para I ←1 hasta M hacer
            para J ← 1 hasta N hacer
                escribir (“Introduzca valor: “, I, J)
                leer (numero(I,J))
                suma ← suma + numero(I,J)
            fin_para
        fin_para
        escribir(“la suma es: “; suma)
    fin
```

Ejemplo: Realizar la suma de 2 matrices bidimensionales de un nº entero, la matriz será 3 2.*

```
Programa suma_matrices
Entorno
    const
        M=3
```

```
                    N=2
            tipo
                    matriz: tabla [1…M,1…N] de real
            var
                    tabR, tab1, tab2 : matriz
                    I, J, valor: entero
        Inicio
            escribir (“Introduzca valores de la tabla 1: “)
            para I ←——1 hasta M hacer
                para J ← 1 hasta N hacer
                    escribir ( “valor: “,I, J)
                    leer (tab1, (I, J))
                fin_para
            fin_para
            para I ←—— 1 hasta M hacer
                para J ← 1 hasta N hacer
                    tabR(I,J) ←— tab1 (I, J) + tab2 (I, J)
                fin_para
            fin_para
              escribir (“la tabla resultado de la suma es:
“, tabR(I,J))
        fin
```

*Ejemplo: Escribir un algoritmo que permita sumar los elementos positivos de una tabla y los elementos negativos, siendo la tabla real de 50 * 20.*

```
        Programa Suma(+,-)
        Entorno
            const
                    M=50
                    N=20
            tipo
                    matriz: tabla [1…M, 1….N] de real
            var
                    numero: matriz
                    suma+, suma- : entero
                    I, J : entero
        Inicio
            escribir (“ Introduzca valores de la
matriz: ←—— “)
            para I 1 hasta M hacer
                para J ← 1 hasta N hacer
                    escribir ( “numero: “ , I, J)
                    leer (numero (I, J))
                    si numero >=0 entonces
                        suma+ ←— suma+ + numero
(I, J)
                    sino
                        suma- ←— suma- + numero
(I,J)
```

```
                finsi
            fin_para
        fin_para
    escribir (“ El resultado de los elementos (+): “, suma+)   escribir (“ El resultado de los elementos (-): “, suma-)
    fin
```

Ejemplo: Tenemos una matriz de 6 filas y 8 columnas que contiene el nº de alumnos matriculados en cada grupo de un centro docente de cada asignatura (grupo: filas; asignaturas: columnas). Hay un vector asignatura tipo cadena.

1: Metodologia	*5: Visual Basic*
2: Análisis	*6: Lenguaje C*
3: Novell	*7: Cobol*
4: Ms_Dos	*8: Pascal*

Realizar un programa que carge desde teclado dicha tabla y a continuación calcule e imprime el total de alumnos matriculados por asignaturas.

```
Program Matriculación
Entorno
    Const
        F=6
        C=8
    Tipo
        T1:Tabla(1..F,1..C) de  entero
        T2:Tabla(1..8) de  cadena
    Var
        Matricula: T1
        asignatura : T2
        I,J,Suma : entero

Inicio

    asignatura (1)  <——  “ Metodologia ”
    asignatura (2)  <——  “ Analisis “
    asignatura (3)  <——  “ Novell “
    asignatura (4)  <——  “ Ms_Dos “
    asignatura (5)  <——  “ Visual Basic “
    asignatura (6)  <——  “ Lenguaje C ”
```

```
asignatura (7)  ←——  " Cobol "
asignatura (8)  ←——  " Pascal "

(*recorre por filas*)
Para I←1 hasta F hacer
        Para J←1 hasta C hacer
                Escribir("Grupo",I,"Asignatura",Asignatura(J))
                Leer(Matricula(I,J)
        Fin_para
Fin_para

Escribir("Alumnos matriculados ")
Escribir("Asignaturas.....Numero de alumnos")

(*recorre por columnas*)
Para J←1 hasta C hacer  (*I es el indice para las columnas*)
        Para I←1 hasta F hacer (*J es el indice para las filas*)
                Escribir("Grupo",I,"Asignatura",Asignatura(J))
                Leer(Matricula(J,I)
        Fin_para
        Escribir(Asignatura(I))
Fin_para
Fin
```

6.5.- Almacenamiento de arrays en memoria.

Debido a la importancia de los arrays, casi todos los lenguajes de programación de alto nivel proporcionan medios eficaces para almacenar y acceder a los elementos de los arrays, de modo que el programador no tenga que preocuparse sobre los detalles específicos de almacenamiento. Sin embargo, el almacenamiento en el ordenador está dispuesto en secuencia contínua, de modo que cada acceso a una matriz o tabla la máquina debe realizar la tarea de convertir la posición dentro del array en una posición perteneciente a una línea.

Representaciones gráficas de arrays de una (vector) y dos (tabla) dimensiones:

Vector

A(1)
A(2)

Tabla

A(1,1)	A(1,2)	A(1,3)	A(1,4)
A(2,1)	A(2,2)	A(2,3)	A(2,4)
A(3,1)	A(3,2)	A(3,3)	A(3,4

A(i)
A(n)

6.5.1.- Almacenamiento de un vector.

El almacenamiento de un vector en memoria se realiza en celdas o posiciones secuenciales. Así, en el caso de un vector A con un subíndice de rango 1 a n.

Posición B	A(1)
Posición B+1	A(2)
.	A(3)
.	
.	A(i)
Posición B+n-1	A(n)

Si cada elemento del array ocupa S bytes (1 byte = 8 bits), y B es la dirección inicial de la memoria central del ordenador, la dirección inicial del elemento i-ésimo sería:

B+(I-1)*S

En general, el elemento N[I] de un array definido como N(L:U) tiene la dirección inicial:

B+(I-L)*S