UNIDAD 4.- CONCEPTOS DE DATOS ESTRUCTURADOS

OBJETIVOS:
Que el alumno:

- Comprenda el concepto de dato estructurado, los seleccione adecuadamente y los organice en forma estructurada.
- Represente adecuadamente las estructuras de arreglo y registro.

TEMAS:

1. Definición de dato estructurado.
2. **Arreglos unidimensionales**: definición, lectura e impresión, operaciones, vectores paralelos, métodos de búsqueda, método de ordenamiento con un vector y con vectores paralelos. Representación en un lenguaje C.
3. **Arreglos bidimensionales**: definición, lectura e impresión, operaciones (suma, resta, multiplicación de un escalar por una matriz, multiplicación de matrices), operaciones por fila, operaciones por columna, búsqueda, ordenamiento, tipos de matrices, elementos característicos de una matriz, representación en lenguaje C.
4. **Arreglos Multidimensionales**: definición, lectura e impresión, operaciones, representación en lenguaje C.
5. **Cadenas de Caracteres**: definición, lectura e impresión, representación en lenguaje C y funciones definidas en él.
6. **Registros**: definición, lectura e impresión, representación en lenguaje C. Registros jerarquizados, array de registros y registros de array.

1. DEFINICIÓN DE DATO ESTRUCTURADO

Se llama estructura de datos o tipo de dato estructurado a los tipos de datos construidos a partir de otros tipos de datos. Es un conjunto de datos.

Pueden realizarse diferentes clasificaciones. Atendiendo al tipo de los datos que la componen.

Desde el punto de vista de la gestión de memoria las estructuras de datos se clasifican en:

- ***Estáticas***: si poseen un número fijo de elementos. Los ejemplos más típicos son los arrays y registros. Su mayor desventaja es la necesidad de tener que definir el número máximo de elementos que podrá tener la estructura. Su mayor ventaja es la rapidez de acceso a cada elemento individual de la estructura. Estas estructuras reservan la gestionan la memoria en tiempo de compilación del programa.
- ***Dinámicas:*** si el número de elementos que contienen puede variar durante la ejecución del programa. Su principal inconveniente es la lentitud en el acceso, ya que normalmente se realiza de forma secuencial. La ventaja es sin embargo importante, la posibilidad de aumentar o disminuir en tiempo de ejecución el número de elementos que componen la estructura. La gestión de memoria se realiza en tiempo de ejecución del programa.

2. ARREGLOS UNIDIMENSIONALES

El array es la estructura de datos más usual y extendida en los lenguajes de programación. Un array es una estructura de datos formada por una cantidad fija de datos de un **mismo tipo**, cada uno de los cuales tiene asociado uno o más índices que determinan de forma unívoca la posición del dato en el array.

El array es una estructura de datos estática y homogénea. Al definir un array en un programa se especifica su tamaño (número de elementos que la constituyen) con lo cual el compilador del lenguaje reserva memoria para el array. Dentro de la memoria, el array se almacena de forma contigua.

Una de las características principales de los arrays es que permiten el acceso directo o aleatorio a sus elementos. En un array unidimensional para alcanzar una determinada posición i, basta con dar ese valor al índice.

Las operaciones básicas son la lectura y escritura de datos en las distintas posiciones.

Prácticamente todos los lenguajes de alto nivel permiten la utilización de variables con subíndice.

Ejemplo de asignación de valor: variable simple B=1

Ejemplo de asignación de valor: variable con subíndice A[5]=1

Un **arreglo** es un tipo de dato estructurado, en el cual todos los elementos son de la misma naturaleza, es decir que son homogéneos, además presenta la característica de que sus

elementos se almacenan en memoria RAM; por lo que al apagar la computadora estos datos se pierden.

Un arreglo unidimensional se denomina **vector** y posee una sola dimensión.

Aquí es fundamental poder distinguir:

- **Orden del Vector**: indica la cantidad de elementos del mismo.
- **Nombre del vector**: indica la forma de distinguir un vector de otro; por lo que dos vectores serán distintos si poseen nombres diferentes y no subíndices diferentes.
  Ejemplo:
  - Vector A
  - Vector B
- **Subíndice**: indica la posición de un elemento.
  - El subíndice puede ser una constante entera. Ejemplo A[31]
  - El subíndice puede ser una variable entera, A[k]
  - El subíndice puede ser una expresión aritmética cuyo resultado sea un valor entero positivo Ejemplo A[k+1+2*N]

  Por ejemplo:
  A[i] representa el elemento i-ésimo del vector A.
  A[2] representa el elemento de posición 2 del vector A.
  A[i+1] representa el i-ésimo +1 elemento.

  Por ejemplo, si tenemos un vector como el siguiente:

  V **N**=4

2	15	7	78

  - N=4 es la cantidad de elementos del vector.
  - V es el nombre del vector.
  - Con i se indicará el subíndice.
  - V[0]=2 es el primer elemento
  - V[N-1]=78 es el elemento que se encuentra en la posición N-1.

DEFINICIÓN, LECTURA E IMPRESIÓN

Un concepto para tener en claro es que al ser un vector un conjunto de datos, tanto su lectura como su impresión se realizan con una estructura de iteración cuyo corte de control será i < N (orden del vector), siempre que se utilice una estructura while y la variable contadora correspondiente al subíndice se inicialice con el valor 0.

Sólo se podrá realizar la impresión o lectura fuera del ciclo de repetición si lo que se pretende leer o imprimir es un único valor (correspondiente a un elemento del vector).

El siguiente programa en C, muestra la definición de un vector de 100 elementos de tipo entero, y la lectura de un vector de orden N.

PROGRAMA PARA DEFINICIÓN – LECTURA E IMPRESIÓN

```
#include<stdio.h>

main (void)
{
        int i,n;
        int v[100];
        printf ("Ingrese la cantidad de elementos");
        scanf("%d",&n);
        for(i=1;i<=n;i++)
        {
                printf("Ingrese un elemento: ");
                scanf("%d",&v[i]);

                printf("%d",v[i]);
        }
        getch();
}
```

Realiza la definición del vector

Ingresa cada uno de los elementos del vector

Muestra cada uno de los elementos del vector

EJEMPLO DE APLICACIÓN

```
/*************************************************************************************
Se tienen las notas de examen de 4 alumnos. Se quiere hacer un programa que
determine cuantos alumnos sacaron una nota mayor al promedio
*************************************************************************************/
```

```
#include<stdio.h>

main (void)
{
        int i,c;
        float nota[4],suma;
        float promedio;
        suma=0;
        c=0;
        for (i=0;i<4;i++)
        {
                printf("\n Nota[%d]= ",i);
                scanf("%f",&nota[i]);
                suma=suma+nota[i];
        }
        promedio=suma/4;
        pintf("\nPromedio= %.2f",promedio);

        for (i=0;i<4;i++)
        {
                If (nota[i]>promedio)
                        c=c+1;
        }
        printf("\nSon %d los alumnos que tienen una nota mayor que el promedio",c);
        printf("\n\n");
}
```

OPERACIONES CON VECTORES

Con vectores se pueden realizar las siguientes operaciones:

1. Suma de dos o más vectores.
2. Resta de dos o más vectores.
3. Multiplicación de dos o más vectores.
4. División de dos vectores.
5. Multiplicación de un vector por un escalar.

1. SUMA DE DOS O MÁS VECTORES

Para sumar dos o más vectores, los mismos deben ser del mismo orden, es decir deben tener la misma cantidad de elementos.

Supongamos dos vectores A y B de orden N, se ingresan los elementos del vector A y y del vector B, luego se realiza la suma en el correspondiente elemento del vector C.

En forma genérica: C[i]=A[i]+B[i]

Vector A

5	7	10

Vector B

3	8	6

Vector C

8	15	16

Programa en Lenguaje C para ingresar y sumar dos vectores

```
#include<stdio.h>

main (void)
{
        int n,i,a[50],b[50],c[50];
        printf("Ingrese el orden del vector: ");
        scanf("%d",&n);

        for (i=0;i<n;i++)
        {
                printf("\n A[%d]= ",i);
                scanf("%d",&a[i]);
                printf("\n B[%d]= ",i);
                scanf("%d",&b[i]);
                c[i]=a[i]+b[i];
        }

        pintf("\nLos valores del vector C son:\n");
        for (i=0;i<n;i++)
        {
                printf("\n C[%d]=%d",i,c[i]);
        }
        getch();
}
```

2. RESTA DE DOS O MÁS VECTORES

Para restar dos o más vectores, los mismos deben ser del mismo orden, es decir deben tener la misma cantidad de elementos.

Supongamos dos vectores A y B de orden N, se ingresan los elementos del vector A y y del vector B, posteriormente se realiza la resta en el correspondiente elemento del vector C.

En forma genérica: C[i]=A[i]-B[i]

Vector A

5	7	10

Vector B

3	8	6

Vector C

2	-1	4

Programa en Lenguaje C para ingresar y restar dos vectores

```
#include<stdio.h>

main (void)
{
        int n,i,a[50],b[50],c[50];

        printf("Ingrese el orden del vector: ");
        scanf("%d",&n);

        for(i=0;i<n;i++)
        {
                printf("\n A[%d]= ",i);
                scanf("%d",&a[i]);
                printf("\n B[%d]= ",i);
                scanf("%d",&b[i]);
                c[i]=a[i]-b[i];
        }

        pintf("\nLos valores del vector C son:\n");
        for (i=0;i<n;i++)
        {
                printf("\n C[%d]=%d",i,c[i]);
        }
}
```

3. MULTIPLICACIÓN DE DOS O MÁS VECTORES

Para multiplicar dos o más vectores, los mismos deben ser del mismo orden, es decir deben tener la misma cantidad de elementos.

Supongamos dos vectores A y B de orden N, se ingresan los elementos del vector A y y del vector B, a continuación, se realiza el producto en el correspondiente elemento del vector C.

En forma genérica: C[i]=A[i]*B[i]

Vector A

5	2	3

Vector B

3	8	6

Vector C

15	16	18

Programa en Lenguaje C para ingresar y multiplicar dos vectores

```
#include<stdio.h>

main (void)
{
        int n,i,a[50],b[50],c[50];
        printf("Ingrese el orden del vector: ");
        scanf("%d",&n);

        for (i=0;i<n;i++)
        {
                printf("\n A[%d]= ",i);
                scanf("%d",&a[i]);
                printf("\n B[%d]= ",i);
                scanf("%d",&b[i]);
                c[i]=a[i]*b[i];
        }

        pintf("\nLos valores del vector C son:\n");
        for (i=0;i<n;i++)
        {
                printf("\n C[%d]=%d",i,c[i]);
        }
        getch();
}
```

4. DIVISIÓN O COCIENTE ENTRE DOS VECTORES

Para dividir dos vectores, los mismos deben ser del mismo orden, es decir deben tener la misma cantidad de elementos, y es condición necesaria que el vector B no tenga elementos nulos, ya que no es posible la división por cero.

Supongamos dos vectores A y B de orden N, se ingresan los elementos del vector A y del vector B, a continuación, se realiza el cociente en el correspondiente elemento del vector C.

En forma genérica: C[i]=A[i]/B[i]

Vector A

9	10	20

Vector B

3	2	4

Vector C

3	5	5

Programa en Lenguaje C para ingresar y dividir dos vectores

```
#include<stdio.h>

main (void)
{
        int n,i,a[50],b[50],c[50];

        printf("Ingrese el orden del vector: ");
        scanf("%d",&n);

        for (i=0;i<n;i++)
        {
                printf("\n A[%d]= ",i);
                scanf("%d",&a[i]);
                printf("\n B[%d]= ",i);
                do
                {
                        scanf("%d",&b[i]);
                }
                while (b[i]==0);
                c[i]=a[i]/b[i];
        }

        pintf("\nLos valores del vector C son:\n");
        for (i=0;i<n;i++)
```

```
        {
                printf("\n C[%d]=%d",i,c[i]);
        }
}
```

5. MULTIPLICACIÓN DE UN ESCALAR POR UN VECTOR

Para multiplicar un vector A por un escalar k, se debe realizar el producto de cada elemento del vector A por el escalar k.

En forma genérica: C[i]=A[i]*k

Vector A

5	7	10

Escalar

K=3

Vector C

15	21	30

Programa en Lenguaje C para ingresar y multiplicar un vector por un escalar

```
#include<stdio.h>
main (void)
{
        int k,n,i,a[50],c[50];
        printf("Ingrese el valor de k: ");
        scanf("%d",&k);
        printf("Ingrese el orden del vector: ");
        scanf("%d",&n);

        for (i=0;i<n;i++)
        {
                printf("\n A[%d]= ",i);
                scanf("%d",&a[i]);
                c[i]=a[i]*k;
        }

        pintf("\nLos valores del vector C son:\n");
        for (i=0;i<n;i++)
        {
                printf("\n C[%d]=%d",i,c[i]);
        }
}
```

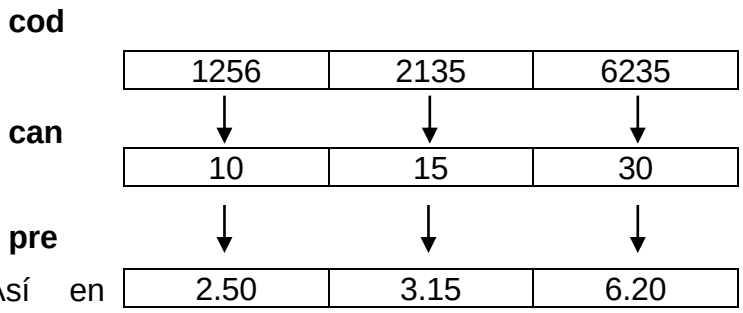
VECTORES PARALELOS

Dos o más arreglos que utilizan el mismo subíndice para acceder a elementos de distintos arreglos, se denominan arreglos paralelos. Estos arreglos pueden procesarse simultáneamente.

En los vectores paralelos la información se corresponde por posición y todos tienen el mismo orden.

Por ejemplo, supongamos que se registra la información de los productos en 3 vectores:

1. Vector **cod:** contiene los códigos de los productos.
2. Vector **can**: contiene la cantidad que hay en stock de cada producto.
3. Vector **pre**: contiene el precio de cada producto.



Así en el ejemplo anterior el código 1256, se corresponde con cantidad igual a 10 y precio igual a 2.50.

Se deben ingresar todos los elementos de los vectores y luego realizar el trabajo solicitado con ellos.

Ejercicio

Realizar la codificación en Lenguaje C, a fin poder ingresar los vectores cod, can y pre, todos de orden N, e indicar cuál es código del producto que posee el precio más alto.

Programa en Lenguaje C

```
#include<stdio.h>

main (void)
{
        int n,i,cod[50],can[50];
        float pre[50];

        printf("Ingrese el orden del vector: ");
        scanf("%d",&n);
        may=0;

        for (i=0;i<n;i++)
```

```
        {
                printf("\n cod[%d]= ",i);
                scanf("%d",&cod[i]);

                printf("\n can[%d]= ",i);
                scanf("%d",&can[i]);

                printf("\n pre[%d]= ",i);
                scanf("%f",&pre[i]);

                if (pre[i]>may)
                {
                      may=pre;
                      pos=i;
                 }
        }
        pintf("\nEl producto con el precio más alto es %d:\n",cod[pos]);
}
```

PROBLEMAS EJEMPLOS

En esta sección se pretende que el alumno tenga las herramientas mínimas necesarias para poder encarar cualquier solución de un problema informático que comprenda arreglos unidimensionales (vectores).

Generalmente se conoce el orden del vector y por ello se trabaja con un ciclo de repetición (estructura while o for).

En todos los casos se presenta la solución en términos de Diagrama de Flujo y Codificación en Lenguaje C.

Ejercicio Nº 1.
Introduzca un vector de N números enteros.

Solución

En este caso particular se poseen datos, pero no resultados, ya que no se especifica el trabajo a realizar con los elementos del vector.

Datos
N : orden del vector
V : nombre del vector
i : subíndice de los elementos del vector
V[i] : cada uno de los elementos del vector

Codificación en Lenguaje C

```
#include<stdio.h>

main (void)
{
      int n,i,v[50];
      printf("Ingrese el orden del vector: ");
      scanf("%d",&n);

      i=0;
      while (i<n)
      {
            printf("\n Ingrese elemento: ");
            scanf("%d",&v[i]);
            i=i+1;
      }
}
```

Ejercicio Nº 2.
Introduzca un vector de N números enteros y muestre la suma de estos.

Solución

Datos
N : orden del vector
V : nombre del vector
i : subíndice de los elementos del vector
V[i] : cada uno de los elementos del vector

Resultado
S: identificador que indica la suma de los valores introducidos en el vector.

Codificación en Lenguaje C

```
#include<stdio.h>

main (void)
{
      int n,i,v[50],s;

      printf("Ingrese el orden del vector: ");
      scanf("%d",&n);
      i=0;
      s=0;
      while (i<n)
      {
```

```
            printf("\n v[%d]= ",i);
            scanf("%d",&v[i]);
            s=s+v[i];
            i=i+1;
  }
  printf("\nSuma de los valores: %d",s);
}
```