UNIDAD 4.- CONCEPTOS DE DATOS ESTRUCTURADOS - ARREGLOS UNIDIMENSIONALES

1. Método de ordenamiento con un vector y con vectores paralelos.
2. Métodos de búsqueda.

1. MÉTODOS DE ORDENAMIENTO

Aunque a lo largo de los años se han desarrollado muchos algoritmos de ordenamiento, los usados con más frecuencia son: método de la burbuja, de selección, de la burbuja mejorada, shell. Todos ellos realizan el ordenamiento en memoria RAM, por ello se denominan métodos de ordenamiento interno.

PRINCIPIOS GENERALES DEL ORDENAMIENTO

Los algoritmos que se presentan comparan elementos de un mismo vector, y, si los dos elementos están desordenados se realiza una transferencia bidireccional entre estos valores.

```
if  (A [ i ] > A[i+1] )
 {
        AUX=A[i];
        A[i] = A [i+1];
        A[i+1]= AUX;
}
```

En la condición anterior, se debe incluir el operador de relación:

- **>** si quiere realizar un ordenamiento en forma ascendente.
- **<** si quiere realizar un ordenamiento en forma descendente.

MÉTODO DE LA BURBUJA

El método es muy clásico y sencillo, pero poco eficiente.

Se basa en la comparación de elementos adyacentes en el vector, e intercambiando sus valores si están desordenados.

Análisis:

- Comienza de izquierda a derecha, comparando a[0] con a[1], si están desordenados se intercambian entre sí.
- A continuación se compara a[1] con a[2] y así sucesivamente.

Programa en Lenguaje C para ordenar un vector de N elementos con Método de la Burbuja

```
#include<stdio.h>

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

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

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

	do
	{
		b=0;
		for (i=0;i<n-1;i++)
		{
			if (a[i]>a[i+1])
			{
				aux=a[i];
				a[i]=a[i+1];
				a[i+1]=aux;
				b=1;
			}
		}
	}
	while (b==1);

	printf("\nEl vector ordenado es:\n");

	for(i=0;i<n;i++)
	{
		printf("\n a[%d]= %d\n",i,a[i]);
	}
}
```

ORDENAMIENTO EN VECTORES PARALELOS

Para poder ordenar los vectores paralelos, se debe en primer lugar, ingresar todos los vectores, en segundo lugar, se debe tener en claro por cuál o cuáles de los vectores se realizará el ordenamiento; siendo el primer vector la clave principal de ordenamiento y los demás se denominan claves secundarias.

Ejemplo:

Ingresar 2 vectores de orden N correspondientes a los datos de legajo y promedio de los alumnos. A partir de él mostrar un listado de los legajos con sus correspondientes promedios ordenados en forma descendente por promedio.

Solución:

Los pasos por seguir para poder realizar este trabajo son:

- Ingresar todos los vectores: la lectura puede realizarse dentro del mismo ciclo de repetición ya que todos son del mismo orden.
- Establecer cuál es el vector clave del ordenamiento, ya que por él se realizará la comparación del elemento que se encuentra en la posición [i], con el que se encuentra en la posición [i+1].
- Establecer si el ordenamiento será ascendente o descendente a fin de determinar el operador de relación (< o >) que comparará los elementos de las posiciones [i] e [i+1].

Programa en Lenguaje C para ordenar vectores paralelos de N elementos con Método de la Burbuja

```
#include<stdio.h>

main ()
{

	int n,i,leg[50],b,auxl;
	float pro[50],auxp;

	printf("Ingrese la cantidad de alumnos: ");
	scanf("%d",&n);

	for (i=0;i<n;i++)
	{
		printf("\n leg[%d]= ",i);
		scanf("%d",&leg[i]);
		printf("   pro[%d]= ",i);
		scanf("%f",&pro[i]);
	}

	do
	{
		b=0;
		for (i=0;i<n-1;i++)
		{
			if (pro[i]<pro[i+1])
			{
				auxl=leg[i];
				leg[i]=leg[i+1];
				leg[i+1]=auxl;

				auxp=pro[i];
				pro[i]=pro[i+1];
```

```
                                pro[i+1]=auxp;

                                b=1;
                        }
                }
        }
        while (b==1);

        printf("\nEl vector ordenado es\n");

        for(i=0;i<n;i++)
        {
                printf("\n leg[%d]= %d\n",i,leg[i]);
                printf("   pro[%d]= %.2f\n",i,pro[i]);
        }
}
```

ORDENAMIENTO CON DISTINTAS CLAVES DE ORDENAMIENTO

Si se poseen vectores paralelos, se puede realizar el ordenamiento por uno o más de ellos. Se debe establecer cuál será la clave primaria y los otros vectores pueden llegar a ser claves secundarias.

Ejemplo:
Ingresar 2 vectores con la información de equipos de fútbol, uno de ellos contiene los puntos obtenidos por el equipo, y otro que contiene la diferencia de goles.
Ambos vectores son de orden N.
A partir de los datos ingresados se debe mostrar la tabla de posiciones.

Solución:

Para poder dar a conocer la tabla de posiciones se deben establecer cuáles serán las claves del ordenamiento.

En este caso se conoce que las tablas de posiciones se ordenan primero por los puntos (clave primaria), si hay dos equipos que tengan igual cantidad de puntos, se ordenan por la diferencia de goles (clave secundaria).

Los pasos por seguir para poder realizar este trabajo son:

- Ingresar todos los vectores: la lectura puede realizarse dentro del mismo ciclo de repetición ya que todos son del mismo orden.
- Establecer cuál es vector clave primaria o principal del ordenamiento, ya que por él se realizará la comparación del elemento que se encuentra en la posición [i], con el que se encuentra en la posición [i+1]. (Clave primaria: Puntos).
- Establecer si el ordenamiento será ascendente o descendente a fin de determinar el operador de relación (< o >) que comparará los elementos de las posiciones [i] e [i+1]. En este caso el ordenamiento debe ser descendente: el operador será de <.

- Si al comparar los elementos de la posición [i] con el de la posición [i+1] fueran iguales, se debe preguntar por la clave secundaria por la rama negativa de la estructura de selección, que será por la diferencia de goles

Programa en Lenguaje C para ordenar un vector de N elementos con Método de la Burbuja con clave de ordenamiento

```
#include<stdio.h>
main ()
{
	int n,i,puntos[50],dife[50],b,auxp,auxd;

	printf("Ingrese la cantidad de equipos:");
	scanf("%d",&n);

	for (i=0;i<n;i++)
	{
		printf("\n puntos[%d]= ",i);
		scanf("%d",&puntos[i]);
		printf("   dife[%d]= ",i);
		scanf("%d",&dife[i]);
	}

	do
	{
		b=0;
		for (i=0;i<n-1;i++)
		{
			if (puntos[i]<puntos[i+1])
			{
				auxp=puntos[i];
				puntos[i]=puntos[i+1];
				puntos[i+1]=auxp;

				auxd=dife[i];
				dife[i]=dife[i+1];
				dife[i+1]=auxd;

				b=1;
			}
			else
			{
				if (puntos[i]==puntos[i+1])
				{
					if (dife[i]<dife[i+1])
					{
						auxp=puntos[i];
						puntos[i]=puntos[i+1];
						puntos[i+1]=auxp;

						auxd=dife[i];
						dife[i]=dife[i+1];
```

```
                                        dife[i+1]=auxd;

                                        b=1;
                                    }
                                }
                            }
                        }
                }
        while (b==1);

        printf("\nEl vector ordenado es\n");
        for (i=0;i<n;i++)
        {
                printf("\n puntos[%d]= %d\n",i,puntos[i]);
                printf("   dife[%d]= %d\n",i,dife[i]);
        }
}
```

2. MÉTODOS DE BÚSQUEDA

Los métodos de búsquedas tratan de determinar si un determinado elemento se encuentra dentro de un vector o no.

La búsqueda de un elemento dentro de un array es una de las operaciones más importantes en el procesamiento de la información, y permite la recuperación de datos previamente almacenados.

El tipo de búsqueda se puede clasificar como interna o externa, según el lugar en el que esté almacenada la información (en memoria o en dispositivos externos). Todos los algoritmos de búsqueda tienen dos finalidades:

- Determinar si el elemento buscado se encuentra en el conjunto en el que se busca.
- Si el elemento está en el conjunto, hallar la posición en la que se encuentra.

MÉTODOS DE BÚSQUEDA SECUENCIAL

El método de búsqueda en un vector desordenado trata de determinar si el elemento k está o no en el vector, para lo cual debe recorrer indefectiblemente el vector completo para saber si no está.

En este método se utiliza una variable bandera (b) a la cual se le definen dos valores posibles, en este caso b=0 significa que el elemento k no está en el vector; en cambio b=1 significa que k está.

Programa en Lenguaje C para buscar el valor K dentro en un vector

```
#include<stdio.h>

main (void)
{

	int n,i,v[50],k,b=0;

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

	printf("Ingrese el elemento k que desea buscar: ");
	scanf("%d",&k);

	for (i=0;i<n;i++)
	{
		printf("\n v[%d]= ",i);
		scanf("%d",&v[i]);
	}

	for(i=0;i<n;i++)
	{
		if (v[i]==k)
		{
			printf("\n %d esta en la posicion %d",k,i);
			b=1;
		}
	}

	if (b==0)
	{
		printf("\n%d no esta en el vector",k);
	}


}
```

MÉTODOS DE BÚSQUEDA BINARIA

La búsqueda binaria consiste en dividir el array por su elemento medio en dos subarrays más pequeños, y comparar el elemento con el del centro. Si coinciden, la búsqueda termina. Si el elemento es menor, debe estar (si está) en el primer subarray, y si es mayor está en el segundo; se descarta la mitad en la que no está y nuevamente en la mitad en la que se puede encontrar el valor k se encuentra el valor medio y se repite el trabajo.

Programa en Lenguaje C para buscar el valor K dentro en un vector con Búsqueda Binaria

```
#include<stdio.h>

main (void)
```

```
{

	int n,i,v[50],k,iz,de,c,b;

	printf("Ingrese el orden del vector: ");
	scanf("%d",&n);
	printf("Ingrese el elemento k que desea buscar: ");
	scanf("%d",&k);

	for (i=0;i<n;i++)
	{
		printf("\n v[%d]= ",i);
		scanf("%d",&v[i]);
	}

	iz=0;
	de=n-1;
	b=0;
	while (iz<=de && b==0)
	{
		c=(iz+de)/2;
		if (v[c]==k)
		{
			b=1;
		}
		else
		{
			if (k>v[c])
			{
				iz=c+1;
			}
			else
			{
				de=c-1;
			}
		}

	}

	if (b==1)
	{
		printf("\nEl elemento se encontro en la posicion %d",c);
	}
	else
	{
		printf("\nEl elemento no se encontro",c);
	}
}
```