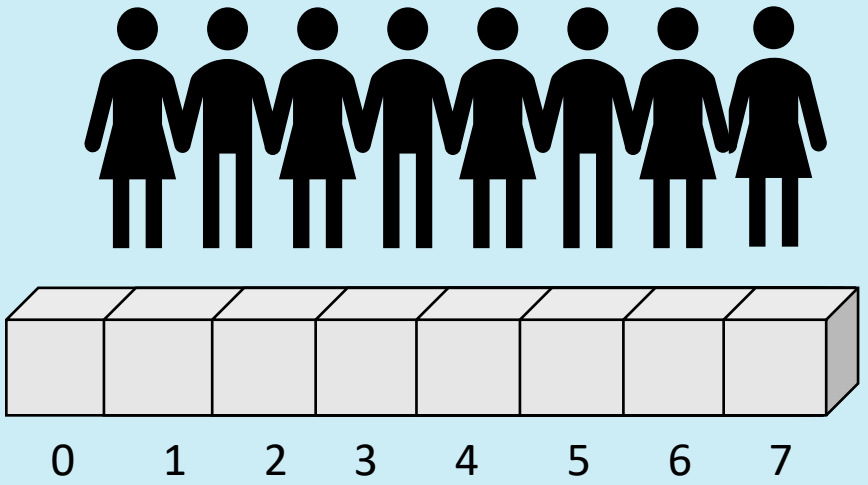
UNIVERSIDAD NACIONAL DE TRUJILLO

UNT

ARREGLOS

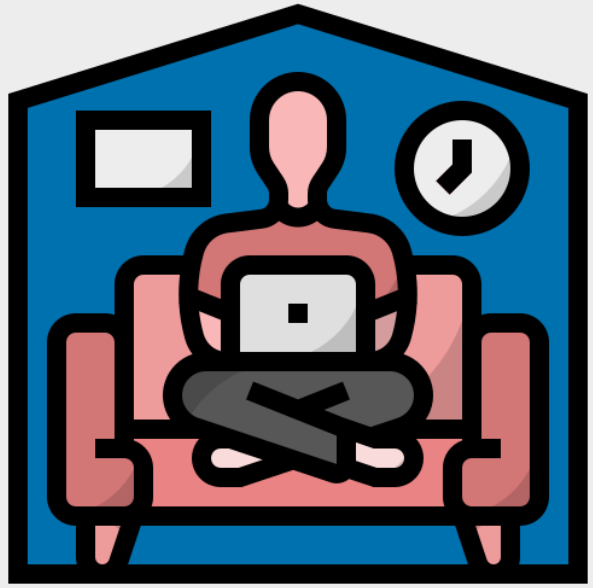


Mg. Marcelino Torres Villanueva

DEFINICIÓN

Un arreglo es un tipo de dato estructurado que almacena en una sola variable un conjunto limitado de datos o elementos del mismo tipo.

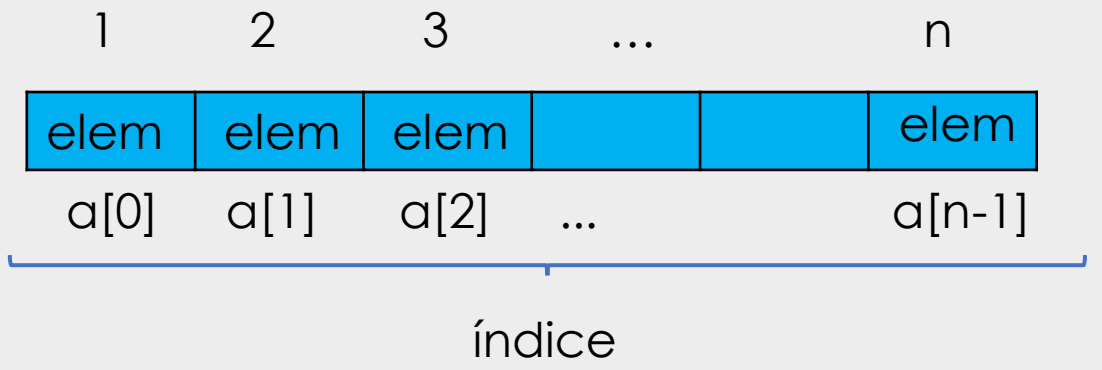
Asimismo, es un conjunto de localidades de memoria contiguas donde la dirección más baja corresponde al primer elemento y la dirección más alta al último. Por sí mismo, el nombre del arreglo apunta a la dirección del primer elemento del arreglo.



Los datos se llaman elementos del arreglo y su posición se numera consecutivamente: 1, 2, 3…n.

Un arreglo en C++ inicia en la posición cero, por lo tanto el i-ésimo elemento está en la posición i-1, es decir si el arreglo llamado a tiene n elementos, sus nombres son: a[0], a[1], ..., a[n-1].

El tipo de elementos almacenados en el arreglo puede ser cualquier tipo de dato.



Para acceder a un elemento específico de un arreglo se usa un índice o subíndice.

CARACTERISTICAS

- ✓ Tener un único nombre de variable que representa a todos los elementos y éstos se diferencian por un índice o subíndice.
- ✓ Almacenar los elementos del arreglo en memoria contigua.
- ✓ Acceder de manera directa o aleatoria a los elementos individuales del arreglo, por el nombre del arreglo y el índice o subíndice.
- ✓ Ser una lista de un número finito de n elementos del mismo tipo.

Importancia de declarar arreglos de tamaño adecuado

Al igual que cualquier variable, los arreglos ocupan espacio en memoria. El programador especifica el tipo de dato y el total de elementos requerido por el arreglo de tal forma que la computadora pueda reservar la cantidad apropiada de memoria.

Si el programador declara un arreglo de 100 elementos de tipo entero y sólo utiliza 10 espacios, desperdicia 90 en memoria para datos de tipo entero. Por lo contrario, si se declara un arreglo de 50 elementos y se quieren manejar 100, faltarán 50 espacios; sin embargo, no se presentará mensaje de error en el tiempo de compilación o ejecución, sino hasta que el sistema operativo se dé cuenta y por lo tanto surja la falla en el programa.

Se pueden declarar varios arreglos en una sola instrucción y de esta forma reservar la memoria necesaria.

Para reservar 100 elementos para el arreglo a y 50 elementos para el arreglo x, ambos de tipo entero, se puede utilizar la siguiente declaración:

PSEUDOCODIGO	CODIFICACION
entero a[100], x[50]	**int a[100], x[50];**

ARREGLOS UNIDIMENSIONALES (VECTORES O LISTAS)

Un arreglo unidimensional es un conjunto de n elementos del mismo tipo almacenados en memoria continua en un vector o lista. Para acceder a cada elemento del arreglo se requiere de un solo índice o subíndice, el cual representa la posición en la que se encuentra.

Forma de declarar un arreglo unidimensional

```
tipo_dato identif_arreglo[tam_arreglo];
```

```
int notas[7];
```

Donde:

- **tipo_dato** se refiere al tipo de dato de cada elemento del arreglo; puede ser entero, real, carácter, etc.
- **identif_arreglo** es el nombre que representa a todo el arreglo.
- **tam_arreglo** es la cantidad de elementos que contiene el arreglo.

Si declaramos un arreglo de tipo entero llamado lista, así será su representación en memoria:

Posición en memoria	1000	1001	1002	1003	1004	1005	1006	1007
lista	0		1		2		3	

Los enteros requieren de dos bytes para almacenarse en memoria; como se muestra, por cada posición se requiere de dos localidades de memoria, por ejemplo el 0 ocupa la posición 1000 y 1001.

La cantidad de arreglos que se pueden declarar dependerá de la memoria libre, que comúnmente es de 64 Kbytes; esta cantidad puede variar e incluso se puede utilizar más memoria disponible siempre y cuando la computadora cuente con ella.

Inicialización de un vector

Cuando se declara un arreglo, sus valores se pueden inicializar de la siguiente manera:

```
int lista[5]= {10, 2, 1, 20, 60}
```

Una Característica importante de los arreglos en C++ es que no se pueden modificar los limites superior e inferior (y por tanto el rango) durante el programa. El límite inferior se fija siempre en 0 y el superior lo fija el programador, es decir:

```
int lista[5]= {10, 2, 1, 20, 60}
```

Posición → 0 1 2 3 4 = (5 posiciones)

Acceso a los elementos de un vector

Cada valor dentro de un vector se conoce como elemento del arreglo. Para acceder a un elemento de un arreglo especifique el nombre del vector con el índice del elemento entre corchetes [].

Ejemplo:

```
int numeros[] = {10, 2, 1, 20, 60};
for(int  indice = 0; indice < 5 ; indice++)
cout<<numeros[indice]<<endl;
```

EJERCICIOS RESUELTOS

Ejercicios resueltos

01. Hacer un programa para ingresar n valores reales en un arreglo y los muestre en la pantalla, además reportar el mayor, el menor y el promedio.

PSEUDOCODIGO

```
numDatos (entero n(R))

    Hacer
        Escribir "Número de elementos del
        Vector: "
            Leer n
    mientras n<=0  o n>MAX

fin_numDatos
```

PSEUDOCODIGO

ingresoVector (real v[], entero n)

```
entero i
Para i ← 0 hasta n-1 inc 1 hacer
        escribir "v[", i, "]:"
        leer v[i]
fin_para
```

fin_ingresoVector

PSEUDOCODIGO

reporteVector (real v[], entero n)

```
entero i
para i ← 0 hasta n-1 inc 1 hacer
        escribir v[i]
fin_para
```

fin_reporteVector

PSEUDOCODIGO

```
real mayor (real v[], entero n)

    entero i
    real may
    may ← v[0]
    para i ←0 hasta n-1 inc 1 hacer
            si v[i] > may entonces
                    may ← v[i]
            fin_si
    fin_para
    retornar may

fin_mayor
```

PSEUDOCODIGO

```
real menor (real v[], entero n)

    entero i
    real men
    men ← v[0]
    para i←0 hasta n-1 inc 1 hacer
        si v[i] < men entonces
            men ← v[i]
        fin_si
    fin_para
    retornar men

fin_menor
```

PSEUDOCODIGO

real promedio (real v[], entero n)

```
entero i
real s
s ← 0
para i←0 hasta n-1 inc 1 hacer
        s ← s + v[i]
fin_para
retornar s/n
```

fin_promedido

PSEUDOCODIGO

Algoritmo arreglos_01

```
    real x[100]
    entero n
    numDatos(n)
    ingresoVector(x,n)
    escribir "Datos ingresados "
    reporteVector(x,n)
    escribir "El mayor es : ", mayor(v,n)
    escribir "El menor es : ", menor(v,n)
    escribir "El promedio es : ",
promedio(v,n)
```

fin_Algoritmo

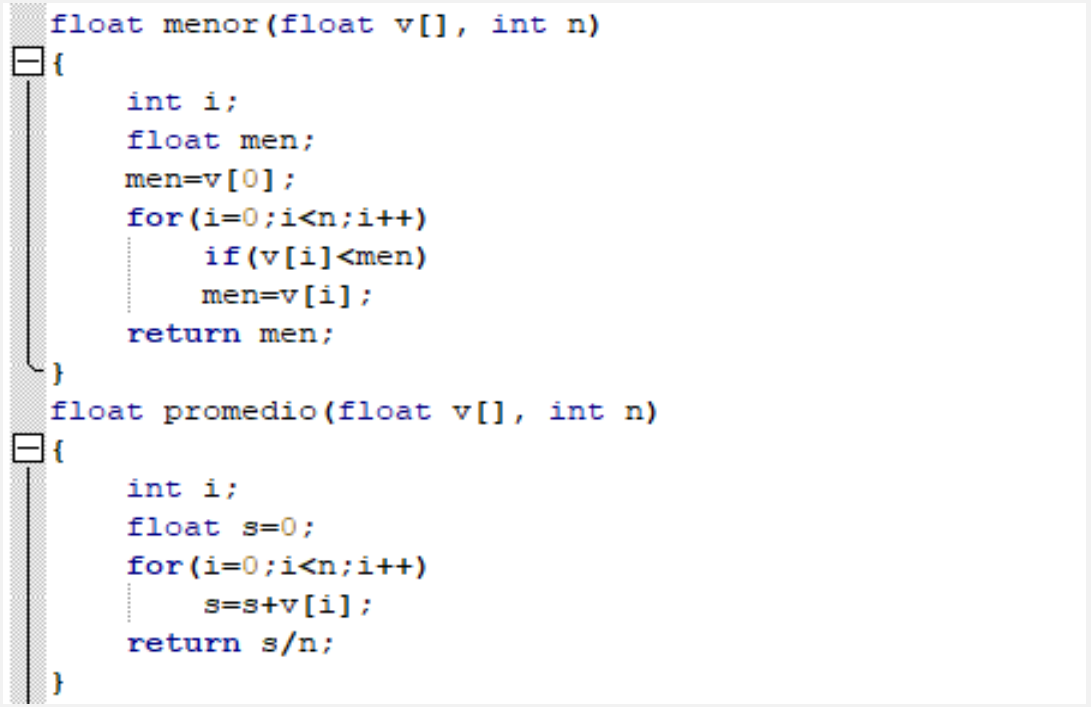
CODIFICACION

```cpp
#include<iostream>
#include <cstdlib>
using namespace std;
#define MAX 100
void numDatos(int &n);
void ingresoVector(float v[], int n);
void reporteVector(float v[], int n);
float mayor(float v[], int n);
float menor(float v[], int n);
float promedio(float v[], int n);
int main(int argc, char *argv[])
{
    float x[MAX];
    int n;
    numDatos(n);
    cout<<"Ingreso de datos del Vector "<<endl;
    ingresoVector(x,n);
    cout<<"Vector Ingresado"<<endl;
    reporteVector(x,n);
    cout<<"El mayor : "<<mayor(x,n)<<endl;
    cout<<"El menor : "<<menor(x,n)<<endl;
    cout<<"El promedio es : "<<promedio(x,n)<<endl;
    return 0;
}
void numDatos(int &n)
{
    do{
        cout<<"Número de elementos del Vector : ";
        cin>>n;
    }while(n<=0 || n>MAX);
}
```

```cpp
void numDatos(int &n)
{
    do{
        cout<<"Número de elementos del Vector : ";
        cin>>n;
    }while(n<=0 || n>MAX);
}
void ingresoVector(float v[], int n)
{
    int i;

    for(i=0;i<n;i++)
    {
        cout<<"v["<<i<<"]:";
        cin>>v[i];
    }
}
void reporteVector(float v[], int n)
{
    int i;

    for(i=0;i<n;i++)
        cout<<v[i]<<endl;
}
float mayor(float v[], int n)
{
    int i;
    float may;
    may=v[0];
    for(i=0;i<n;i++)
        if(v[i]>may)
        may=v[i];
    return may;
}
```

CODIFICACION	SALIDA DE PANTALLA
	Número de elementos del Vector : 5 Ingreso de datos del Vector v[0]:1 v[1]:2 v[2]:3 v[3]:4 v[4]:5 Vector Ingresado 1 2 3 4 5 El mayor : 5 El menor : 1 El promedio es : 3

```
float menor(float v[], int n)
{
    int i;
    float men;
    men=v[0];
    for(i=0;i<n;i++)
        if(v[i]<men)
        men=v[i];
    return men;
}
float promedio(float v[], int n)
{
    int i;
    float s=0;
    for(i=0;i<n;i++)
        s=s+v[i];
    return s/n;
}
```

02. Programa para ingresar n valores reales en un arreglo y calcular la desviación standard.

ds= $$\sqrt{\frac{\sum_{i=1}^{n}(x_i-\overline{x})^2}{n-1}}$$

y = $$\overline{x}=\frac{1}{N}\sum_{i=1}^{N}x_i$$

PSEUDOCODIGO

real desviacionStandard(real v[], entero n)

```
entero i
real p,suma,ds
p ← promedio(v,n)
suma ← 0
para i ← 0 hasta n-1 inc 1 hacer
        suma ← suma + (v[i] – p)2
fin_para
ds ← raizCuadrada(suma/(n-1))
retornar ds
```

fin_desviacionStandard

PSEUDOCODIGO

Algoritmo arreglos_02

```
real x[100]
entero n
numDatos(n)
ingresoVector(x,n)
escribir "Datos ingresados "
reporteVector(x,n)
escribir "La desviación Standard es: ",
desviacionStandard(v,n)
```

fin_Algoritmo

CODIFICACION

```cpp
#include<iostream>
#include <cmath>
using namespace std;

#define MAX 100
void numDatos(int &n);
void ingresoVector(float v[], int n);
void reporteVector(float v[], int n);
float promedio(float v[], int n);
float desviacionStandard(float v[], int n);

int main(int argc, char *argv[])
{
    float x[MAX];
    int n;
    numDatos(n);
    cout<<"Ingreso de datos del Vector "<<endl;
    ingresoVector(x,n);
    cout<<"Vector Ingresado"<<endl;
    reporteVector(x,n);
    cout<<"La desviación standard es : "<<desviacionStandard(x,n)<<endl;
    return 0;
}
void numDatos(int &n)
{
    do{
        cout<<"Número de elementos del Vector : ";
        cin>>n;
    }while(n<=0 || n>MAX);
}
```

```cpp
void ingresoVector(float v[], int n)
{
    int i;

    for(i=0;i<n;i++)
    {
        cout<<"v["<<i<<"]:";
        cin>>v[i];
    }
}
void reporteVector(float v[], int n)
{
    int i;

    for(i=0;i<n;i++)
        cout<<v[i]<<endl;
}
float promedio(float v[], int n)
{
    int i;
    float s=0;
    for(i=0;i<n;i++)
        s=s+v[i];
    return s/n;
}

float desviacionStandard(float v[], int n)
{
    int i;
    float p=promedio(v,n),suma=0,ds;

    for(i=0;i<n;i++)
        suma=suma + pow(v[i]-p,2);
    ds=sqrt(suma/(n-1));
    return ds;
}
```

SALIDA DE PANTALLA

```
Número de elementos del Vector : 5
Ingreso de datos del Vector
v[0]:21
v[1]:32
v[2]:10
v[3]:5
v[4]:12
Vector Ingresado
21
32
10
5
12
La desviación standard es: 10.6536
```

03. Programa para ingresar n valores reales en un vector y luego invierta el vector.

PSEUDOCODIGO

```
invierteVector(real v[], entero n)

    entero i,j
    real temp
    j ← n-1
    para i←0 hasta entero(n/2)-1 inc 1 hacer
        temp ← v[i]
        v[i] ← v[j]
        v[j] ← temp
        j ← j -1
    fin_para

fin_invierteVector
```

PSEUDOCODIGO

Algoritmo arreglos_03

```
real x[100]
entero n
numDatos(n)
ingresoVector(x,n)
escribir "Datos ingresados "
reporte vector(x,n)
invierteVector(x,n)
escribir "Vector invertido", reporteVector(x,n)
```

fin_Algoritmo

CODIFICACION

```cpp
#include<iostream>
#include <cmath>
using namespace std;

#define MAX 100
void numDatos(int &n);
void ingresoVector(float v[], int n);
void reporteVector(float v[], int n);
void invierteVector(float v[], int n);

int main()
{
    float x[MAX];
    int n;
    numDatos(n);
    cout<<"Ingreso de datos del Vector "<<endl;
    ingresoVector(x,n);
    cout<<"Vector Ingresado"<<endl;
    reporteVector(x,n);
    invierteVector(x,n);
    cout<<"Vector invertido "<<endl;
    reporteVector(x,n);

    return 0;
}
void numDatos(int &n)
{
    do{
        cout<<"Número de elementos del Vector : ";
        cin>>n;
    }while(n<=0 || n>MAX);
}
```

```cpp
void ingresoVector(float v[], int n)
{
    int i;

    for(i=0;i<n;i++)
    {
        cout<<"v["<<i<<"]:";
        cin>>v[i];
    }
}
void reporteVector(float v[], int n)
{
    int i;

    for(i=0;i<n;i++)
        cout<<v[i]<<endl;
}
void invierteVector(float v[], int n)
{
    int i,j;
    float temp;

    for(i=0,j=n-1;i<n/2;i++,j--)
    {
        temp=v[i];
        v[i]=v[j];
        v[j]=temp;
    }
}
```

SALIDA DE PANTALLA

```
Número de elementos del Vector : 6
Ingreso de datos del Vector
v[0]:12
v[1]:13
v[2]:10
v[3]:5
v[4]:11
v[5]:8
Vector Ingresado
12
13
10
5
11
8
Vector invertido
8
11
5
10
13
12
```

Ejercicios resueltos

04. Programa para ingresar 2 vectores de n elementos reales cada uno y reportar el producto escalar de ellos.

PSEUDOCODIGO

```
real productoEscalar(real x[], real y[], entero n)

    entero i
    real pe
    pe ← 0
    para i←0 hasta n-1 inc 1 hacer
            pe ← pe + x[i] * y[i]
    fin_para
    retornar pe

fin_promedido
```

PSEUDOCODIGO

Algoritmo arreglos_04

```
real x[100], y[100]
entero n
numDatos(n)
escribir "Ingreso de datos del primer Vector",
ingresoVector(x,n)
escribir "Ingreso de datos del segundo
Vector",
ingresoVector(y,n)
escribir "El producto escalar es: " ,
productoEscalar(x,y.n)
```

fin_Algoritmo

CODIFICACION

```cpp
#include<iostream>
using namespace std;

#define MAX 100
void numDatos(int &n);
void ingresoVector(float v[], int n);
void reporteVector(float v[], int n);
float productoEscalar(float x[], float y[], int n);
int main()
{
    float x[MAX],y[MAX];
    int n;
    numDatos(n);
    cout<<"Ingreso de Datos del Primer Vector"<<endl;
    ingresoVector(x,n);
    cout<<"Ingreso de Datos del Segundo Vector"<<endl;
    ingresoVector(y,n);
    cout<<"El producto escalar es : "<<productoEscalar(x,y,n)<<endl;
    return 0;
}
void numDatos(int &n)
{
    do{
        cout<<"Número de elementos de los vectores : ";
        cin>>n;
    }while(n<=0 || n>MAX);
}
```

```cpp
void ingresoVector(float v[], int n)
{
    int i;

    for(i=0;i<n;i++)
    {
        cout<<"v["<<i<<"]:";
        cin>>v[i];
    }
}
void reporteVector(float v[], int n)
{
    int i;

    for(i=0;i<n;i++)
        cout<<v[i]<<endl;
}
float productoEscalar(float x[], float y[], int n)
{
    int i;
    float pe=0;
    for(i=0;i<n;i++)
        pe=pe+x[i]*y[i];
    return pe;
}
```

SALIDA DE PANTALLA

```
Número de elementos de los vectores : 5
Ingreso de Datos del Primer Vector
v[0]:10
v[1]:2
v[2]:15
v[3]:4
v[4]:8
Ingreso de Datos del Segundo Vector
v[0]:12
v[1]:34
v[2]:20
v[3]:4
v[4]:2
El producto escalar es : 520
```

05. Hacer un programa usando un vector de Reales donde se haga lo siguiente: Ingresar un elemento, consultar un elemento, modificar un elemento, eliminar un elemento, insertar un elemento y ordenar los elementos. Todo esto en un menú:

Vector
[1] Ingresar Elemento
[2] Consultar Elemento
[3] Modificar elemento
[4] Eliminar elemento
[5] Insertar elemento
[6] Ordenar elementos
[7] Mostrar elementos
[8] Salir

1. Búsqueda

ARREGLO INICIAL

12	20	8	10	5	VALOR
0	1	2	3	4	POSICION

Dato a buscar: 10

X[0]= 10? **No**

X[1]= 10? **No**

X[2]= 10? **No**

X[3]= 10? **Si**

Devuelve el índice 3

Dato a buscar: 27

X[0]= 27? **No**

X[1]= 27? **No**

X[2]= 27? **No**

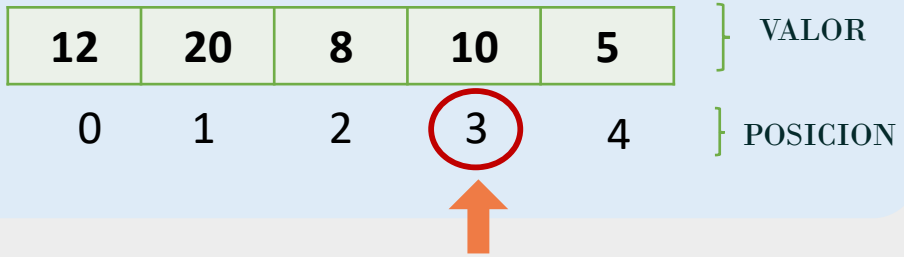
X[3]= 27? **No**

X[4]= 27? **No**

Como no lo encontró devuelve -1

2. Modificar un elemento que se encuentra en la posición p

ARREGLO INICIAL



Posición del elemento a modificar p=3

X[3] =7

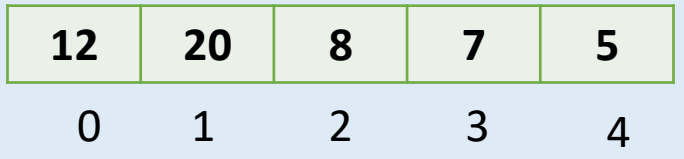
Nuevo valor : 7

ARREGLO FINAL

12	20	8	7	5
0	1	2	3	4

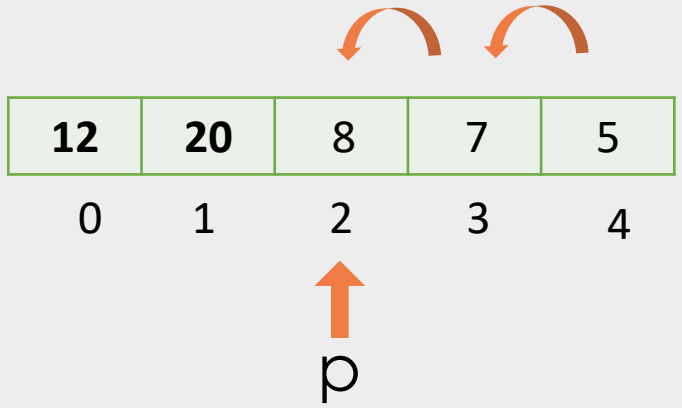
3. Eliminar un elemento que se encuentra en la posición p

ARREGLO INICIAL



n=5

Si p=2

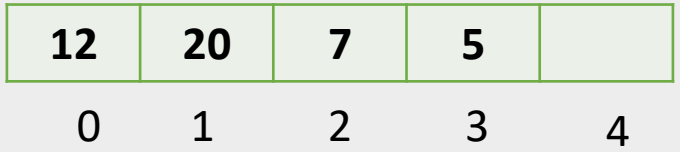


X[2] = X[3]

X[3] = X[4]

n= n-1=5-1=4

ARREGLO FINAL



n=4

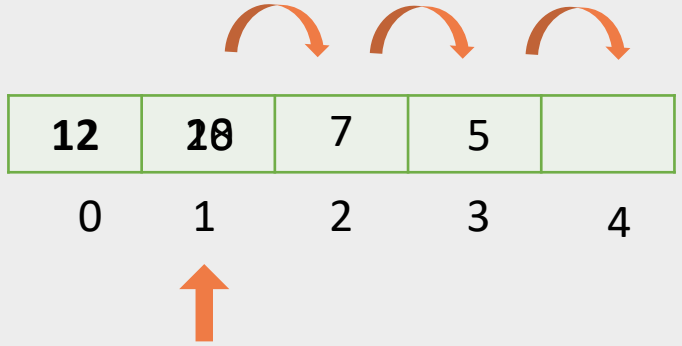
4. Insertar un nuevo elemento en una posición p

ARREGLO INICIAL

12	20	7	5	
0	1	2	3	4

n=4

Dato a insertar : 18 Posición donde se desea insertar p=1



X[4] = X[3]

X[3] = X[2]

X[2] = X[1]

n= n+1=4+1=5

ARREGLO FINAL

12	18	20	7	5
0	1	2	3	4

n=5

5. Ordenamiento por intercambio

ARREGLO INICIAL

20	12	6	18	9

i=0

j=1

20	12	6	18	9
0	1	2	3	4

V[0] >V[1]

SE REALIZA INTERCAMBIO

20	12	6	18	9
0	1	2	3	4

j=2

12	20	6	18	9
0	1	2	3	4

V[0] > V[2]

SE REALIZA INTERCAMBIO

12	20	6	18	9
0	1	2	3	4

j=3

6	20	12	18	9
0	1	2	3	4

V[0] > V[3]

NO SE REALIZA INTERCAMBIO

6	20	12	18	9
0	1	2	3	4

j=4

6	20	12	18	9
0	1	2	3	4

V[0] > V[4]

NO SE REALIZA INTERCAMBIO

6	20	12	18	9
0	1	2	3	4

i=1

j=2

6	20	12	18	9
0	1	2	3	4

V[1] >V[2]

SE REALIZA INTERCAMBIO

6	20	12	18	9
0	1	2	3	4

j=3

6	12	20	18	9
0	1	2	3	4

V[1] > V[3]

NO SE REALIZA INTERCAMBIO

6	12	20	18	9
0	1	2	3	4

j=4

6	12	20	18	9
0	1	2	3	4

V[1] > V[4]

SE REALIZA INTERCAMBIO

6	12	20	18	9
0	1	2	3	4

i=2

j=3

6	9	20	18	12
0	1	2	3	4

V[2] >V[3] SI

SE REALIZA INTERCAMBIO

6	9	20	18	12
0	1	2	3	4

j=4

6	9	18	20	12
0	1	2	3	4

V[2] > V[4] SI

SE REALIZA INTERCAMBIO

6	9	18	20	12
0	1	2	3	4

i=3

j=4

6	9	12	20	18
0	1	2	3	4

V[3] > V[4] SI

SE REALIZA INTERCAMBIO

6	9	12	20	18
0	1	2	3	4

PSEUDOCODIGO

```
ingresarElementos(real v[], entero n(R))

    escribir "Ingrese número: "
    leer v[n]
    n ← n + 1

fin_ingresarElemento


entero búsqueda(real v[], entero n, real dato)
    entero i
    para i ←0 hasta n-1 inc 1 hacer
        si v[i] = dato entonces
            retornar i
        fin_si
    fin_para
    retornar -1
fin_busqueda
```

PSEUDOCODIGO

```
consultarElemento(real v[], entero n)

    real dato
    entero p
    escribir "Dato a buscar : "
    leer dato
    p ← búsqueda(v,n,dato)
    si p ≠ -1 entonces
            escribir "El dato se encuentra en la posición ", p
    sino
            escribir "El dato no se Encuentra"
   fin_si

fin_consultarElemento
```

PSEUDOCODIGO

modificarElemento(real v[], entero n)

```
real dato
entero p
escribir "Dato a modificar: "
leer dato
p ← búsqueda(v,n,dato)
si p ≠ -1 entonces
      escribir "Nuevo dato: "
      leer v[p]
sino
          escribir "El dato no se encuentra"
fin_si
```

fin_modificarElemento

PSEUDOCODIGO

eliminar(real v[], entero n(R) , entero p)

```
entero i
para i←p hasta n -2 inc 1 hacer
        v[i] ← v[i+1]
fin_para
n ← n – 1
```

fin_eliminar

PSEUDOCODIGO

eliminarElemento(real v[], entero n(R))

```
real dato
entero p
escribir "Dato a eliminar: "
leer dato
p ← búsqueda(v,n,dato)
si p ≠ -1 entonces
    eliminar(v,n,p)
    escribir "Dato eliminado"
sino
        escribir "El dato no se encuentra"
fin_si
```

fin_eliminarElemento

PSEUDOCODIGO

insertar(real v[], entero n(R) , real dato, entero p)

```
entero i
para i←n-1 hasta p inc -1 hacer
        v[i+1] ← v[i]
fin_para
v[p] ← dato
n ← n + 1
```

fin_insertar

PSEUDOCODIGO

insertarElemento(real v[], entero n(R))

```
real dato
entero p
escribir "Dato a insertar: "
leer dato
Hacer
        escribir "Posición donde desea insertar:"
        leer p
mientras p<0 o p>n
insertar(v,n,dato,p)
```

fin_insertarElemento

PSEUDOCODIGO

```
ordenar(real v[], entero n)

    entero i,j
    real temp

    para i ←0 hasta n - 2 inc 1 hacer
        para j ← i+1 hasta n – 1 inc 1 hacer
            si v[i] > v[j] entonces
                temp ← v[i]
                v[i] ← v[j]
                v[j] ← temp
            fin_si
        fin_para
    fin_para

fin_ordenar
```

PSEUDOCODIGO

reporteVector(real v[], entero n)

```
entero i
para i ←0 hasta n - 1 inc 1 hacer
        escribir v[i]
fin_para
```

fin_reporteVector

PSEUDOCODIGO

ordenarElementos(real v[], entero n)

ordenar(v,n)
escribir "Datos ordenados"
reporteVector(v,n)

fin_ordenarElementos

PSEUDOCODIGO

mostrarElementos(real v[], entero n)

escribir "Elementos del Vector"
reporteVector(v,n)

fin_mostrarElementos

PSEUDOCODIGO

```
Algoritmo Operaciones_con_Vectores
    real x[100]
    entero n, op
    n ← 0
    Hacer
        escribir " MENU "
        escribir "[1] Ingresar elemento"
        escribir "[2] Consultar elemento"
        escribir "[3] Modificar elemento"
        escribir "[4] Eliminar elemento"
        escribir "[5] Insertar elemento"
        escribir "[6] Ordenar elementos"
        escribir "[7] Mostrar elementos"
        escribir "[8] Salir "
        escribir "Ingrese opcion (1-8):"
        leer op
        según_sea op hacer
                caso 1: ingresarElemento(x,n)
                caso 2: consultarElemento(x,n)
                caso 3: modificarElemento(x,n)
                caso 4: eliminarElemento(x,n)
                caso 5: insertarElemento(x,n)
                case 6: ordenarElementos(x,n)
                caso 7: mostrarElementos(x,n);
        fin_segun_sea
    mientras op ≠ 8
fin_algoritmo
```

CODIFICACION

```
#include<iostream>
#include <cstdlib>
using namespace std;
#define MAX 100

void ingresarElemento(float v[], int &n);
int busqueda(float v[], int n, float dato);
void consultarElemento(float v[], int n);
void modificarElemento(float v[], int n);
void eliminar(float v[], int &n,int p);
void eliminarElemento(float v[], int &n);
void insertar(float v[], int &n, float dato,int p);
void insertarElemento(float v[], int &n);
void ordenar(float v[], int n);
void ordenarElementos(float v[], int n);
void reporteVector(float v[], int n);
void mostrarElementos(float v[], int n);

int main(int argc, char *argv[])
{
    float x[MAX];
    int n=0,op;
    do{
        system("cls");
        cout<<"MENU VECTORES"<<endl;
        cout<<"[1] Ingresar elemento"<<endl;
        cout<<"[2] Consultar elemento"<<endl;
        cout<<"[3] Modificar elemento"<<endl;
        cout<<"[4] Eliminar elemento"<<endl;
        cout<<"[5] Insertar elemento"<<endl;
        cout<<"[6] Ordenar elementos"<<endl;
        cout<<"[7] Mostrar elementos"<<endl;
        cout<<"[8] Salir "<<endl;
        cout<<"Ingrese opcion (1-8):";
        cin>>op;
        switch(op){
        case 1: ingresarElemento(x,n);break;
        case 2: consultarElemento(x,n);break;
        case 3: modificarElemento(x,n);break;
        case 4: eliminarElemento(x,n);break;
        case 5: insertarElemento(x,n);break;
        case 6: ordenarElementos(x,n);break;
        case 7: mostrarElementos(x,n); break;
        }
    }while(op!=8);
}
void ingresarElemento(float v[], int &n)
{
    system("cls");
    cout<<"Ingrese número : ";
    cin>>v[n];
    n++;
}
int busqueda(float v[], int n, float dato)
{
    int i;
    for(i=0;i<n;i++)
        if(v[i]==dato) return i;
    return -1;
}
```

CODIFICACION

```
void consultarElemento(float v[], int n)
{
    float dato;
    int p;
    system("cls");
    cout<<"dato a buscar: ";
    cin>>dato;
    p=busqueda(v,n,dato);
    if(p!=-1)
        cout<<"El dato se encuentra en la posición: "<<p<<endl;
    else
        cout<<"El dato no se encuentra"<<endl;
    system("pause");
}
void modificarElemento(float v[], int n)
{
    float dato;
    int p;
    system("cls");
    cout<<"dato a modificar: ";
    cin>>dato;
    p=busqueda(v,n,dato);
    if(p!=-1)
    {
        cout<<"Nuevo dato : ";
        cin>>v[p];
    }
    else
       cout<<"El dato no se encuentra"<<endl;
    system("pause");
}
```

```
void eliminar(float v[], int &n,int p)
{
    int i;
    for(i=p;i<n-1;i++)
        v[i]=v[i+1];
    n=n-1;
}
void eliminarElemento(float v[], int &n)
{
    float dato;
    int p;
    system("cls");
    cout<<"dato a eliminar : ";
    cin>>dato;
    p=busqueda(v,n,dato);
    if(p!=-1)
    {
        eliminar(v,n,p);
        cout<<"Dato eliminado"<<endl;
    }
    else
       cout<<"El dato no se encuentra"<<endl;
    system("pause");
}
void insertar(float v[], int &n, float dato,int p)
{
    int i;
    for(i=n-1;i>=p;i--)
        v[i+1]=v[i];
    v[p]=dato;
    n=n+1;
}
```

CODIFICACION

```
void insertarElemento(float v[], int &n)
{
    float dato;
    int p;
    system("cls");
    cout<<"dato a insertar: ";
    cin>>dato;
    do{
        cout<<"Posición donde desea insertar : ";
        cin>>p;
    }while(p<0|| p>n);
    insertar(v,n,dato,p);
}
void ordenar(float v[], int n)
{
    int i,j;
    float temp;
    for(i=0;i<n-1;i++)
        for(j=i+1;j<n;j++)
        if(v[i]>v[j])
    {
        temp=v[i];
        v[i]=v[j];
        v[j]=temp;
    }
}
void reporteVector(float v[], int n)
{
    int i;

    for(i=0;i<n;i++)
        cout<<v[i]<<endl;
}
```

```
void ordenarElementos(float v[], int n)
{
    system("cls");
    ordenar(v,n);
    cout<<"Datos Ordenados "<<endl;
    reporteVector(v,n);
    system("pause");
}
void mostrarElementos(float v[], int n)
{
    system("cls");
    cout<<"Elementos del Vector "<<endl;
    reporteVector(v,n);
    system("pause");
}
```

SALIDA DE PANTALLA

```
MENU VECTORES
[1] Ingresar elemento
[2] Consultar elemento
[3] Modificar elemento
[4] Eliminar elemento
[5] Insertar elemento
[6] Ordenar elementos
[7] Mostrar elementos
[8] Salir
Ingrese opcion (1-8):1

Ingrese número : 10
Ingrese número : 23

Presione una tecla para continuar . . .

dato a buscar: 10
El dato se encuentra en la posición: 0

Presione una tecla para continuar . . .

dato a modificar: 23
Nuevo dato : 100
```

```
Presione una tecla para continuar . . .

dato a eliminar : 23
El dato no se encuentra

Presione una tecla para continuar . . .

dato a insertar: 15
Posición donde desea insertar : 1

Datos Ordenados
10
15
100
Presione una tecla para continuar . . .

Elementos del Vector
10
15
100
Presione una tecla para continuar . . .
```