

Operación de Inserción

- 1) Verificar que el arreglo tenga posiciones disponibles para insertar un nuevo elemento.
- 2) Si existe posiciones disponibles proceder a realizar la inserción.
- 3) Si el arreglo esta ordenado se debe realizar la inserción en la posición que corresponda para que quede ordenado después de realizar la misma (al inicio, al medio o al final).
- 4) Si el arreglo no esta ordenado se puede optar por insertar el nuevo elemento al final del arreglo.
- 5) Para realizar la inserción , se debe mover los elementos una posición hacia la derecha. Es necesario conocer la posición del último elemento del arreglo.

Tipos de Inserción que se pueden realizar en arreglos con las siguientes condiciones:

Operación de Inserción

- Arreglo ordenado sin elementos repetidos
- Arreglo ordenado con elementos repetidos.
- Arreglo desordenado sin elementos repetidos.
- Arreglo desordenado con elementos repetidos.

Operación de Inserción

Arreglo Original, desordenado

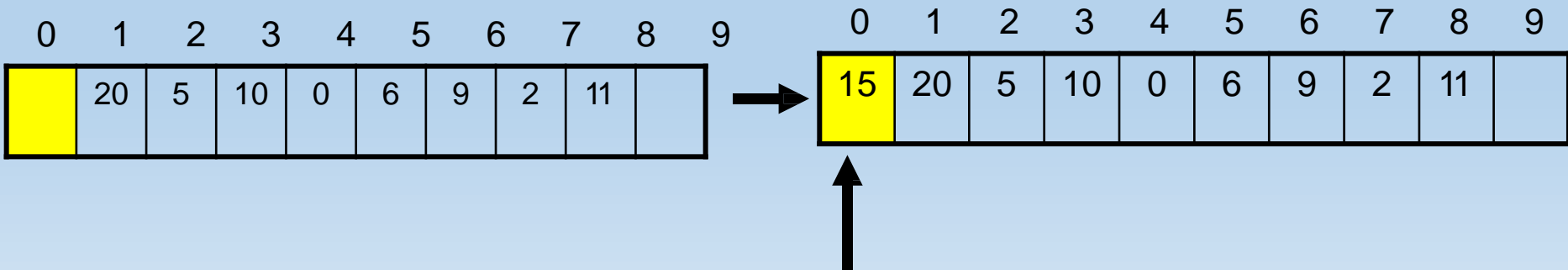
0	1	2	3	4	5	6	7	8	9
20	5	10	0	6	9	2	11		

Tiene libre las posiciones 8 y 9.

Tiene libre las posiciones 8 y 9. Por Ejemplo Insertar el elemento 15 en la posición 8. El nuevo arreglo después de la inserción es:

20	5	10	0	6	9	2	11	15	
----	---	----	---	---	---	---	----	----	--

También se puede optar por insertar en la posición 1. En este caso se deben mover los elementos una posición hacia la derecha, dejar libre la posición 1 e insertar el nuevo elemento. Por Ejemplo Insertar el elemento 15 en la posición 0. El nuevo arreglo después de la inserción es:



Operación de Inserción

Arreglo Original , ordenado sin elemento repetidos

0	1	2	3	4	5	6	7	8	9
5	7	10	12	13	16	18	20		

Tiene libre las posiciones 8 y 9.

Tiene libre las posiciones 8 y 9, por lo tanto hay lugares disponibles para insertar.

El elemento que se ingresa se debe colocar en la posición que corresponda para que el arreglo continúe ordenado después de realizar la inserción del nuevo elemento.

Una vez que se ingresa el elemento a insertar, se debe preguntar si la inserción es:

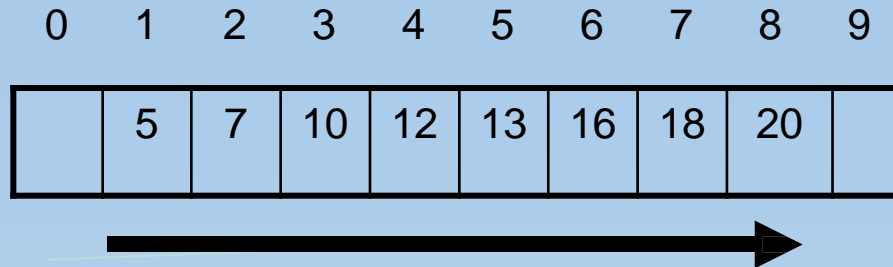
- Al inicio ,antes del primer elemento
- Al final del arreglo
- Al medio del arreglo

Insertar al inicio del arreglo

Por ejemplo insertar el número 3.

$3 < 5$, entonces puedo realizar la inserción.

Desplazo los elementos del arreglo una posición a la derecha, quedando la posición 0 disponible para insertar 3.



*/*Desplazar los elementos del arreglo una posición hacia la derecha */*
for (j=i; j<ult;j++)
a[j+1]= a[j];

Insertar el número 3_



Insertar al final del arreglo

Por ejemplo insertar el número 22.

$20 < 22$, entonces puedo realizar la inserción al final del arreglo.


0	1	2	3	4	5	6	7	8	9
5	7	10	12	13	16	18	20	22	

Insertar al medio del arreglo

Por ejemplo insertar el número 11.

**Tengo que recorrer el arreglo hasta encontrar el primer numero mayor, en este caso 12.
A partir de 12 tengo que desplazar los elementos una posición hacia la derecha, para
dejar un lugar libre en la posición 3**

0	1	2	3	4	5	6	7	8	9
5	7	10		12	13	16	18	20	



*/*Desplazar los elementos del arreglo una posición hacia la derecha */*
for (j=ult; j>i;j--)
a[j+1]= a[j];

Ahora se puede insertar el número 11 en la posición 3.

0	1	2	3	4	5	6	7	8	9
5	7	10	11	12	13	16	18	20	

Pregunta:

¿Existe alguna función en C que me permita saber si el arreglo esta con todas sus posiciones ocupadas?




```
#include <stdio.h>

#include <conio.h>

int main()

{

    int a[]={0, 4, 78, 5, 32};

    int n,t,cant,c,num;

    n= sizeof(a);

    printf ("Tamaño en bytes: %d\n",n);

    t = sizeof(a[0]);

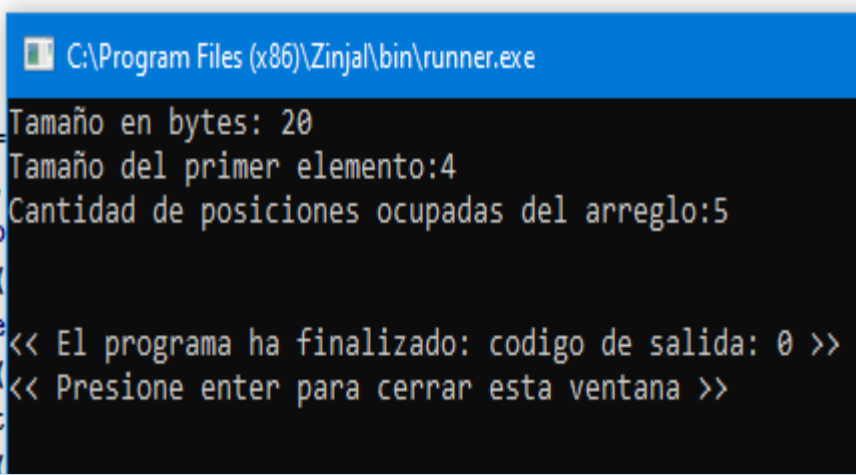
    printf ("Tamaño del primer elemento:%d\n",t);

    cant=n/t;

    printf ("Cantidad de posiciones ocupadas del arreglo:%d\n",cant);

    return 0;

}
```



```
C:\Program Files (x86)\Zinjal\bin\runner.exe

Tamaño en bytes: 20
Tamaño del primer elemento:4
Cantidad de posiciones ocupadas del arreglo:5

<< El programa ha finalizado: codigo de salida: 0 >>
<< Presione enter para cerrar esta ventana >>
```

```
#include <stdio.h>
```

```
#include <conio.h>
```

```
int main()
```

```
{
```

```
    int a[]={0, 4, 78, 5, 32};
```

```
    int n,t,cant,c,num;
```

```
    n= sizeof(a);
```

```
    printf ("Tamaño en bytes: %d\n",n);
```

```
    t = sizeof(a[0]);
```

```
    printf ("Tamaño del primer elemento:%d\n",t);
```

```
    cant=n/t;
```

```
    printf ("Cantidad de posiciones ocupadas del arreglo:%d\n",cant);
```

```
    printf("ingrese un elemento a insertar\n");
```

```
    scanf("%d",&num);
```

```
    a[cant]=num;
```

```
    for(c=0; c<=cant;c++) printf("elemento %d:%d \n", c,a[c]);
```

```
    return 0;
```

C:\Program Files (x86)\Zinjal\bin\runner.exe

```
Tamaño en bytes: 20
Tamaño del primer elemento:4
Cantidad de posiciones ocupadas del arreglo:5
ingrese un elemento a insertar
10
elemento 10:3792896
elemento 0:0
elemento 1:4
elemento 2:78
elemento 3:5
elemento 4:32
elemento 5:10
elemento 6:4
elemento 7:20
elemento 8:8
elemento 9:6422284
```

Pregunta:

¿Es un error de codificación?

El lenguaje ¿permite hacer esta operación?

¿Pueden contestar estas preguntas?



Operación de Eliminación

- 1) Verificar que el arreglo se encuentre el elemento a eliminar.
- 2) Si existe el elemento en el arreglo proceder a realizar la eliminación.
- 3) La eliminación se realiza moviendo los elementos una posición hacia la izquierda los elementos del arreglo, quedando una posición libre. Para esto es necesario identificar la posición en la que se encuentra el elemento a eliminar .
- 4) Luego de realizar la eliminación el arreglo cuenta con un elemento menos.

Arreglo Original

0	1	2	3	4	5	6	7	8	9
20	5	10	0	6	9	2	11	7	3

Eliminar el elemento 9.

Al eliminar el elemento 9 en la posición 5. El nuevo arreglo después de la eliminación es:

20	5	10	0	6	2	11	7	3	
----	---	----	---	---	---	----	---	---	--

Al eliminar el elemento 9, queda libre la última posición

Tipos de Eliminación se pueden realizar en arreglos con las siguientes condiciones:

Operación de Eliminación

- **Arreglo ordenado sin elementos repetidos.** En este caso se puede utilizar la búsqueda binaria para localizar el elemento a eliminar.
- **Arreglo ordenado con elementos repetidos.**
- **Arreglo desordenado sin elementos repetidos.**
- **Arreglo desordenado con elementos repetidos.**

```

#include <stdio.h>#include <conio.h>
#define max 9
int main()
{
    int a[max]={0, 4, 78, 5, 32, 9, 77, 1, 23};
    int n,i,j;

    printf("Operación de eliminación en un arreglo desordenado sin elementos repetidos:\n ");
    printf("Ingresar elemento a eliminar:\n ");
    scanf("%d",&n);

    /*Buscar elemento en el arreglo

    i=0;
    while (n!=a[i] && i<= max)
        i++;

    if (a[i]== n)
    {
        /*Eliminar el elemento n del arreglo */
        for (j=i; j<max;j++)
            a[j]= a[j+1];
        /*Mostrar el arreglo despues de la eliminación*/
        for(i = 0; i < max-1; i++)
            printf("Digito %d:%d\n",i,a[i]);
    }
    else printf ("Elemento no está en el arreglo\n");

    return 0;
    getch();
}

```

```
#include <stdio.h>
#include <conio.h>
#define max 9
```

```
int main()
{
```

```
    int a[max]= {0, 4, 78, 0, 32, 9, 77, 4, 23};
    int n,i,j,ult,c;
```

```
    printf("Operación de eliminación en un arreglo desordenado con elementos repetidos:\n ");
    ult=max;
    printf("Arreglo original\n");
    for(j = 0; j < ult; j++)
        printf("Digito %d:%d\n",j,a[j]);
```

```
    printf("Ingresar elemento a eliminar:\n ");
    scanf("%d",&n);
    i=0;
    c=0;
    while (i<= ult)
    {
        if (a[i]== n)
        {
            /*Eliminar el elemento n del arreglo */
            for (j=i; j<ult;j++)
                a[j]= a[j+1];
            /*Mostrar el arreglo despues de la eliminación*/
            c++;
            printf("eliminación nro.%d\n",c);
            ult=ult-1;
            for(j = 0; j < ult; j++)
                printf("Digito %d:%d\n",j,a[j]);
        }
        i++;
    }
    if (ult==max)printf( "Elemento no encontrado en el arreglo\n");

    return 0;
    getch();
```

```
}
```

C:\Program Files (x86)\Zinjal\bin\runner.exe

```
Digito 2:78
Digito 3:0
Digito 4:32
Digito 5:9
Digito 6:77
Digito 7:4
Digito 8:23
Ingresar elemento a eliminar:
4
eliminación nro.1
Digito 0:0
Digito 1:78
Digito 2:0
Digito 3:32
Digito 4:9
Digito 5:77
Digito 6:4
Digito 7:23
eliminación nro.2
Digito 0:0
Digito 1:78
Digito 2:0
Digito 3:32
Digito 4:9
Digito 5:77
Digito 6:23

<< El programa ha finalizado: codigo de salida: 0 >>
<< Presione enter para cerrar esta ventana >>_
```

Realizar las correcciones al siguiente programa.

C:\Program Files (x86)\Zinjal\bin\runner.exe

```
Ingresar elemento a eliminar:
0
eliminación nro.1
Digito 0:4
Digito 1:78
Digito 2:0
Digito 3:32
Digito 4:9
Digito 5:77
Digito 6:4
Digito 7:23
eliminación nro.2
Digito 0:4
Digito 1:78
Digito 2:32
Digito 3:9
Digito 4:77
Digito 5:4
Digito 6:23
eliminación nro.3
Digito 0:4
Digito 1:78
Digito 2:32
Digito 3:9
Digito 4:77
Digito 5:4

<< El programa ha finalizado: codigo de salida: 0 >>
<< Presione enter para cerrar esta ventana >>
```



```
#include <stdio.h>
#define max 9
int sumoarreglo(int a[],int n);
```

/* Funcion sumoarreglo, devuelve la suma de un arreglo de enteros */

```
int sumoarreglo(int a[],int n)
{
    int suma=0;
    for (int i=0;i<n;i++)
        suma += a[i];
    return suma;
}
```

No se ha especificado el tamaño de a. El ordenador contará los elementos que hemos puesto entre llaves y reservará espacio para ellos. De esta forma siempre habrá el espacio necesario,

```
int main()
{
    int b[max]= {0, 4, 78, 5, 32, 9, 77, 1, 23};
    int z;
    z=0;
    z= sumoarreglo(b,max);
    printf ("La suma es: %d",z);

    return 0;
}
```

Agregar la siguiente función al programa anterior.

```
/* Funcion promedio
float promedio(int a[],int n)
{
    int suma=0;
    float prom;
    prom=0;
    for (int i=0;i<n;i++)
        suma += a[i];
    prom=suma/n;
    return prom;
}
```