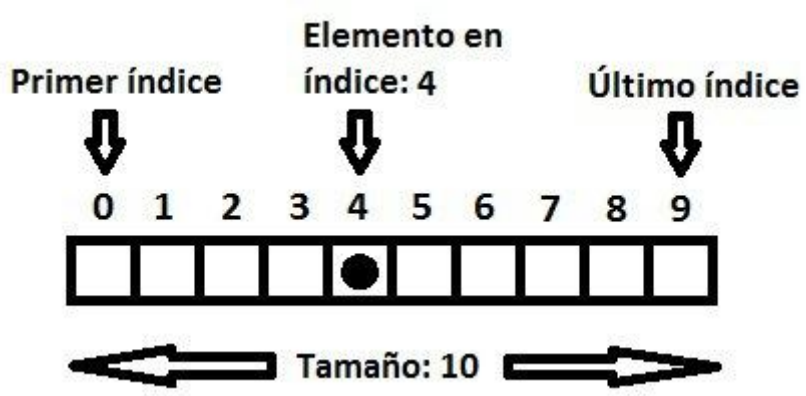
UNIDAD CURRICULAR: ALGORITMICA Y PROGRAMACIÓN

UNIDAD VII. ARREGLOS

CONTENIDO:



UNIDAD VII

ARREGLOS

Un arreglo es una secuencia de posiciones de la memoria a las que se puede acceder directamente, que contienen datos del mismo tipo que pueden ser seleccionados individualmente mediante el uso de subíndices.

Clasificación de los arreglos

- Arreglos unidimensionales o vectores
- Arreglos multidimensionales.

Arreglos Unidimensionales

Un arreglo unidimensional, array, vector o tabla es un tipo de datos estructurado que está formado de una colección finita y ordenada de datos del mismo tipo.

El tipo de acceso a los arreglos unidimensionales es el acceso directo, es decir, podemos acceder a cualquier elemento del arreglo sin tener que consultar a elementos anteriores o posteriores, esto mediante el uso de un índice para cada elemento del arreglo que nos da su posición relativa.

Para implementar arreglos unidimensionales se debe reservar espacio en memoria, y se debe proporcionar la dirección base del arreglo, la cota superior y la inferior.

Declaración de un arreglo

tipo_base nombre_del arreglo [*número_de_posiciones*]

tipo_base: es el tipo de dato que guardará el arreglo (int, float, char, entre otros).

nombre_del_arreglo: Corresponde al nombre del arreglo o vector.

número_de_posiciones: Cantidad de elementos o posiciones que tendrá el arreglo.

Ejemplo:

int arreglo [10];

En este caso, el arreglo tiene 10 posiciones, es decir, tiene capacidad para guardar 10 elementos de tipo entero.
También se puede definir una constante para el número máximo de posiciones del arreglo y esta es la manera mas recomendada:

Ejemplo:

#define N 10

int arreglo [N];

En este ejemplo se define una constante N con el valor de 10, por lo tanto el arreglo es de 10 posiciones.
Si luego se desea cambiar el número de posiciones del arreglo, basta con cambiar el valor de la constante N.

Representación de un arreglo en la memoria

Un arreglo ocupa tantas celdas de memoria como el cardinal de su tipo índice:

Ejemplo:

int arregloedad [9];

Se puede decir que la variable ***arregloedad*** tiene 9 celdas.

Representación gráfica del *arregloedad:*

Contenido del arreglo →	18	20	15	25	22	19	21	17	16
Subíndice →	0	1	2	3	4	5	6	7	8

El contenido del arreglo corresponde a los valores de cada uno de los elementos del arreglo, mientras que el subíndice corresponde a la posición de cada elemento, puede comenzar desde 0 o desde 1, según el lenguaje de programación que se utilice. En el caso de C++ comienza desde 0.

Accediendo un arreglo

Cada celda del arreglo se puede acceder como una variable independiente. En C++ se comienza a enumerar las celdas desde 0. Para acceder a un elemento del arreglo hay que indicar el nombre del arreglo y entre corchetes el subíndice o posición de dicho elemento:

arreglo[posición_o_subíndice]

Ejemplo:

arregloedad[2]

En este caso se está haciendo referencia al tercer elemento del arreglo arregloedad, que contiene el valor 15.

Operaciones básicas de los arreglos:

- Asignación
- Lectura
- Escritura
- Recorrido(acceso secuencial)
- Actualizar (agregar, modificar, eliminar)
- Ordenamiento
- Búsqueda

Asignación: La asignación de valores a un elemento del vector se realizará con el operador de asignación(=).Si se desea asignar un valor a alguna posición del arreglo, lo que se hace es indicar el nombre del arreglo y entre corchetes el subíndice o la posición donde se va a guardar el valor.

Nombre_del_arreglo [subíndice] = valor;

Ejemplo:

```
arregloedad[3]= 25;
```

En este ejemplo se le asigna el valor 25 al elemento que se encuentra en el subíndice 3 (al cuarto elemento) del arreglo ***arregloedad***.

Lectura: consiste en "Cargar" o asignar valores a los elementos del arreglo. Generalmente estos valores los asigna el usuario a través de la instrucción de lectura y utilizando estructuras repetitivas o por asignación directa.

Ejemplo:

Siguiendo con el ejemplo anterior del arreglo ***arregloedad:***

18	20	15	25	22	19	21	17	16
0	1	2	3	4	5	6	7	8

Al realizar las siguientes asignaciones:

arregloedad[1]= 12;
arregloedad [5]= 28;

el arreglo quedaría de la siguiente manera:

18	12	15	25	22	28	21	17	16
0	1	2	3	4	5	6	7	8

Nótese que el elemento de subíndice 1 tenía el valor de 20 y al realizar la asignación ***arregloedad[1]= 12***, el valor 20 es sustituido por el 12. Lo mismo ocurre con el elemento que está en la posición 5.

Si lo que se desea es que el usuario sea el que asigne valores al arreglo, se debe utilizar la instrucción de lectura de datos:

cin >>arreglo[posición_o_subíndice];

Ejemplo:

cout <<"Ingrese el valor de la posición 0:";
cin >>arregloedad[0];

Con estas instrucciones, a la posición 0 (es decir a la primera posición) del arreglo se le asignará el valor que el usuario suministre.

En caso de asignar valores al resto de las posiciones se tendría que pedir al usuario los ocho datos restantes:

```
cout <<"Ingrese el valor de la posición 1:";
cin >>arregloedad[1];
cout <<"Ingrese el valor de la posición 2:";
cin >>arregloedad[2];
cout <<"Ingrese el valor de la posición 3:";
cin >>arregloedad[3];
cout <<"Ingrese el valor de la posición 4:";
cin >>arregloedad[4];
cout <<"Ingrese el valor de la posición 5:";
cin >>arregloedad[5];
cout <<"Ingrese el valor de la posición 6:";
cin >>arregloedad[6];
cout <<"Ingrese el valor de la posición 7:";
cin >>arregloedad[7];
cout <<"Ingrese el valor de la posición 8:";
cin >>arregloedad[8];
```

Esta sería una forma muy rudimentaria, pues si el arreglo tiene una cantidad mayor de posiciones, sería muy largo el programa. Es por ello que para las operaciones con arreglos se hace uso de las estructuras repetitivas, (sobre todo se utiliza el for).

Escritura: corresponde a mostrar el contenido de un arreglo, a través de la instrucción de escritura e indicando el arreglo y la posición donde se encuentra el elemento a visualizar.

cout<<arreglo[posición];

Ejemplo:

Siguiendo con el ejemplo del arreglo ***arregloedad,*** si se desea mostrar la edad que está almacenada en la octava posición, la instrucción sería la siguiente:

cout <<"El contenido de la octava edad es: " <<arregloedad[7];

En este caso mostrará el elemento que se encuentra en el subíndice 7, es decir, el valor 17.

Recorrido: Se puede acceder a los elementos de un arreglo para introducir datos en él (leer), o bien para visualizar su contenido (escribir). A la operación de efectuar una acción general sobre todos los elementos de un arreglo, se le denomina recorrida del arreglo. Para ello se utilizan estructuras repetitivas (haga desde) donde la variable de control puede ser utilizada como subíndice del arreglo la cual se va a ir incrementando permitiendo acceder sucesivamente cada elemento del mismo

Ejemplo:

Cargar el arregloedad por medio del suministro de datos del usuario.

```
#define N 9
for (k=0; k<N k++)
{ cout<<"ingresa la edad: ";
  cin>> arregloedad[k];
}
```

La variable control del for (k) servirá de subíndice del arreglo, por esta razón se inicializa en 0 de manera que se guarden los datos en el arreglo desde la posición 0.

N corresponde al número de posiciones del arreglo , y al colocar la condición k<N el ciclo se repetirá mientras k sea menor que el número de total de posiciones del arreglo. Por lo tanto se realizarán 9 iteraciones (desde k=0 hasta 8).

La escritura de arreglos también se representa de manera similar:

Ejemplo:

```
Mostrar el contenido del arregloedad:
for (k=0; k<N k++)
{ cout<<"ingresa la edad: "<< arregloedad[k];
}
```

Actualizar (agregar, modificar, eliminar): La operación de agregar o incluir consiste en añadir valores a los elementos de un arreglo siempre y cuando haya espacio en él. Una vez cargado el arreglo se pueden modificar los valores de los elementos del arreglo, tal como se realizó en ejemplos anteriores. La operación de borrar un elemento del arreglo consiste en desplazar los elementos del arreglo que están después del elemento a eliminar a la posición anterior. Estas operaciones se visualizaran más adelante con un ejemplo práctico.

MÉTODOS DE ORDENAMIENTO

El ordenamiento de un arreglo es un procedimiento mediante el cual se disponen los elementos de un arreglo en un orden específico: alfabética o numéricamente, y ascendente o descendentemente.

Existen varios métodos de ordenamiento, tales como, burbuja, quick sort, heap sort, selección, inserción, entre otros. Sin embargo, a continuación se explicará el de selección.

Método de ordenamiento de selección: Consiste en determinar el elemento mas pequeño del arreglo y colocarlo en la primera posición. Luego se busca el mas pequeño del resto de los elementos y se intercambia con la segunda posición y así se repite el mismo proceso con el resto de los elementos del arreglo. A continuación se presenta el código del método:

```
void orden_seleccion()
 { int i,j,aux;
   for (i=0; i<N-1;i++)              //N es el número de posiciones del arreglo
      for (j=i+1; j<N; j++)
         if (arreglo[ j ]< arreglo[ i ])
            { aux = arreglo[ i ];
```

```
                arreglo[ i ]= arreglo[ j ];
                arreglo[ j ]= aux;
              }
        }
```

MÉTODOS DE BÚSQUEDA

La búsqueda de un elemento dado en un arreglo es una aplicación muy utilizada. Existen dos tipos de búsqueda básicas que realizan esta actividad: la búsqueda secuencial y la búsqueda binaria.

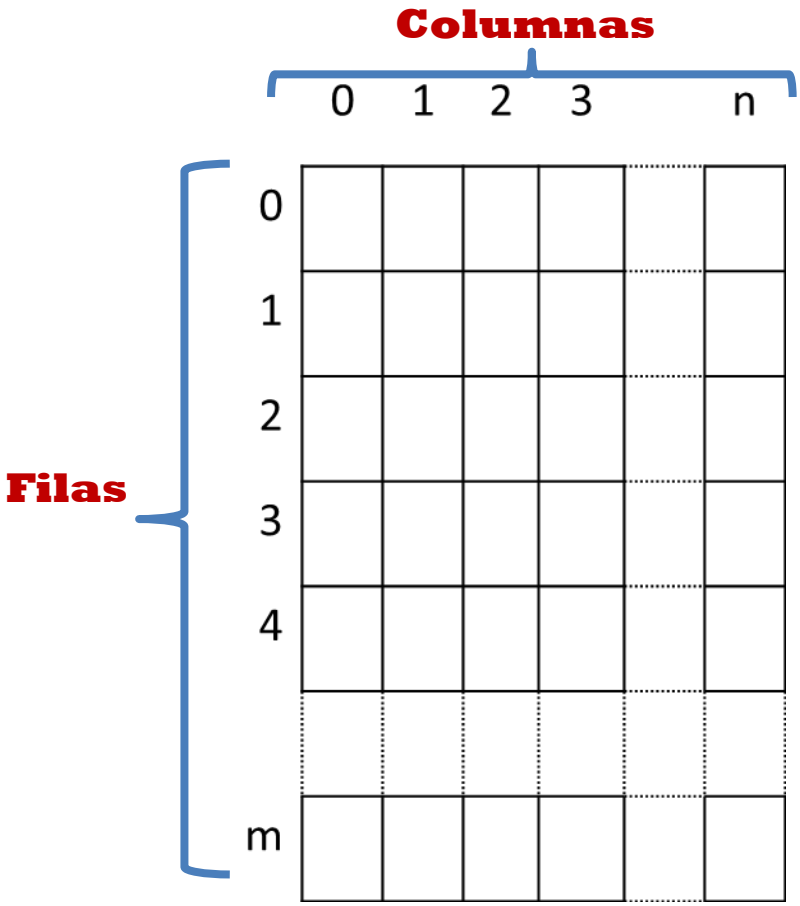
Búsqueda secuencial: busca un elemento determinado recorriendo secuencialmente el arreglo desde la primera posición y se detiene cuando se encuentra el elemento buscado o se alcanza el fin del arreglo. Este método es utilizado para arreglos cuyos contenidos no están ordenados.

Búsqueda binaria: utiliza un método de "divide y vencerás" para localizar el valor buscado y requiere que los elementos del arreglo estén ordenados. Este método consiste en examinar primero el elemento central de la lista; si este es el elemento buscado, entonces la búsqueda ha terminado. En caso contrario, se determina si el elemento buscado está en la primera o en la segunda mitad del arreglo y, a continuación, se repite este proceso, examinando el elemento central del subarreglo.

Arreglos multidimensionales

Son arreglos que tienen mas de una dimensión, y en consecuencia, mas de un índice. Los arreglos mas usuales son los de dos dimensiones, conocidos también como arreglos bidimensionales, tablas o matrices. Sin embargo, es posible crear arreglos de tantas dimensiones como se requiera.

A continuación se presenta la estructura de un arreglo bidimensional con ***m*** filas y ***n*** columnas.



Estructura de una arreglo bidimensional

EJERCICIOS RESUELTOS

a) Declare un arreglo que permita guardar los sueldos de 100 empleados. Una forma de declarar sería la siguiente:

float sueldos[100];

Sin embargo es mas conveniente realizar la declaración utilizando una constante:

#define E 100

float sueldos[E];

b) Realice una función que permita guardar en el arreglo anterior, los sueldos de los empleados suministrados por teclado

```
void cargar_arreglo_sueldos()
{ int j;
  for (j=0; j<E; j++)
   { cout<< "ingresa el sueldo: ";
      cin>> sueldos[j]; }
}
```

c) Realice una función que retorne el monto total a pagar en la nómina de empleados.

```
float total_nomina()
{ int j;
  float acum_sueldos=0;
  for (j=0; j<E; j++)
     {acum_sueldos=acum_sueldos+sueldos[j];}
 return acum_sueldos;
}
```

d) Realice una función que retorne el número de empleados que ganan menos de 10000 Bs.

```
float total_empleados()
{ int j, cont_empl=0;
  for (j=0; j<E; j++)
     if (sueldos[j]<10000)
        cont_empl=cont_empl+1;
   return cont_empl;
}
```

e) Usando la función ***total_nomina***, calcule el promedio de sueldos.
promedio=total_nomina()/E;

REFERENCIAS BIBLIOGRÁFICAS

Bassard, G y Bratley, P. (2010). Fundamentos de algoritmia. Prentice-Hall.

Joyanes, L. (2008). Fundamentos de programación. Algoritmos , Estructuras de datos y objetos. Mc Graw Hill. Tercera edición.

Joyanes, L. y Zahonero, I. (2005). Programación en C. Metodología, algoritmos y Estructura de datos. Mc Graw Hill. Segunda Edición

Martí, N. y Ortega, Y. (2004). Estructuras de datos y Métodos Algorítmicos. Ejercicios Resueltos. Pearson Education.