INSTITUTO POLITÉCNICO NACIONAL
UNIDAD PROFESIONAL INTERDISCIPLINARIA EN INGENIERÍA Y TECNOLOGÍAS AVANZADAS

Práctica No. 3. Manejo de arreglos unidimensionales y multidimensionales.

Unidad Temática: II Estructuras de control y Aplicaciones con Arreglos

Lugar de realización: Laboratorio de Cómputo **Duración:** 12.5 hrs.

Objetivo

Desarrollar programas aplicando el método de la burbuja y resolver problemas con arreglos bidimensionales.

Resultados Esperados

- Trabajo de investigación logrado en el pre-reporte
- Solución de casos prácticos en Lenguaje C haciendo uso de arreglos unidimensionales y multidimensionales para mostrar su uso y aplicación.
- Solución casos prácticos relacionados con operaciones matemáticas utilizando matrices.
- Solución problemas de ordenación con el método de la Burbuja.

Pre-reporte.

Para el buen desarrollo de la práctica el alumno entregará un reporte que incluya los siguientes elementos:

1. Investigar sintaxis y un ejemplo de vectores.
2. Investigar sintaxis y un ejemplo de matrices.
3. Investigar operaciones que se pueden hacer con matrices.

"Las oportunidades no son producto de la casualidad,
más bien son resultado del trabajo. "
(Tonatíhu)

Material y Equipo.

- IDE Dev-Cpp o compatible.
- Computadora

Introducción.

Los arreglos pueden ser de dos tipos:

- Unidimensionales (vector)
- Multidimensionales

Arreglos o vectores

Un **vector** es una secuencia de datos del mismo tipo. Los tipos de datos pueden ser de cualquier tipo, como: `int`, `float`, `char`, estructuras, etc.

Tiene un límite inferior que es 0 y un límite superior que es *n-1*, donde *n* es el número de elementos almacenados en el vector.

Declaración de un vector:

```
<tipo> <nombre_arreglo>[<número_de_elementos>];
```

Ejemplo:

int numeros[10];

Para acceder a cada elemento se usa su índice:

```
for(i = 0; i<10; i++)
      numeros[i]= 0;
```

Un vector se ilustra gráficamente de la siguiente manera:

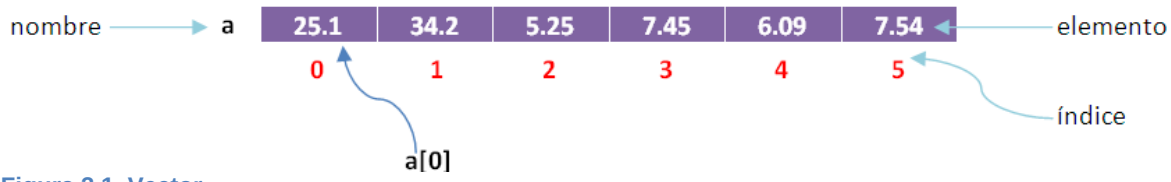


Figura 3.1. Vector

Arreglos multidimensionales

Los **arreglos multidimensionales** son aquellos que tienen más de una dimensión y por lo tanto más de un índice.

Los más usuales son los de dos dimensiones también conocidos como: **tablas** o **matrices.**

Declaración de una matriz:

```
<tipo de dato><nombre arreglo> [<número_filas>][<número_columnas>]
```

El primer índice permite acceder a las filas y el segundo a las columnas, los índices de igual manera inician en 0.

Una matriz se ilustra gráficamente de la siguiente manera:

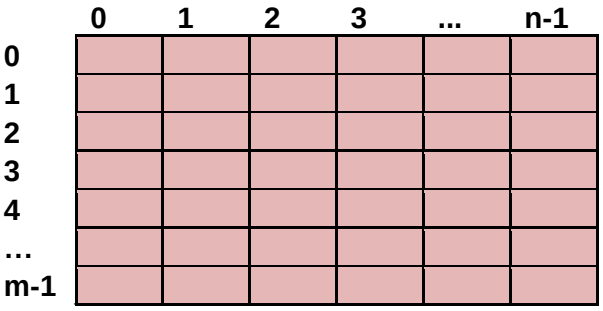


Figura 3.2. Matriz

Donde el número de elementos de la matriz está dado por: (m)*(n)

Ejemplo para insertar datos en una matriz:

```
float cd[2][4];
for(i = 0; i < 2; i++){
      for(j = 0; j < 4; j++)
            scanf("%d", &cd[i][j]);
}
```

A continuación se muestra la salida del código anterior:

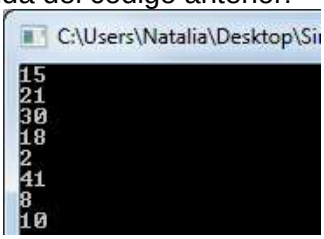


Figura 3.3. Resultado de datos en matriz

Desarrollo.

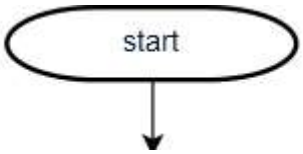
Ejemplo:

Llenar un vector de 6 elementos con datos del teclado y pedir al usuario el número a buscar, utilizando búsqueda secuencial.

Solución:

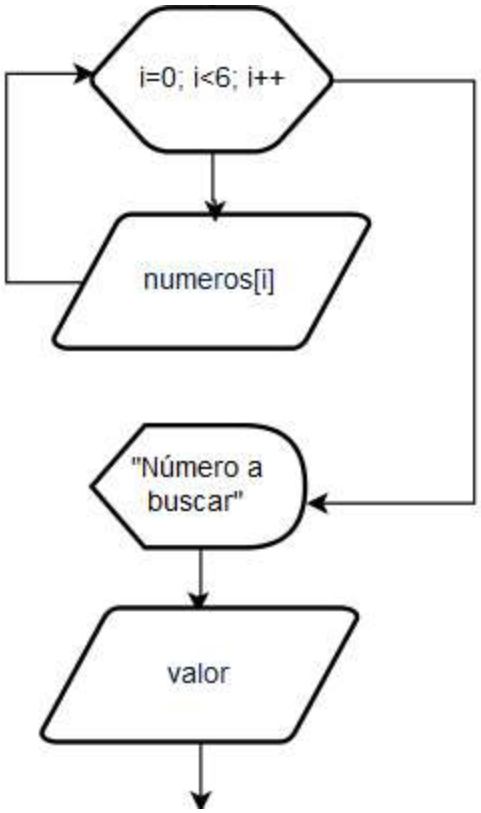
Diagrama de Flujo

1. Inicio del programa

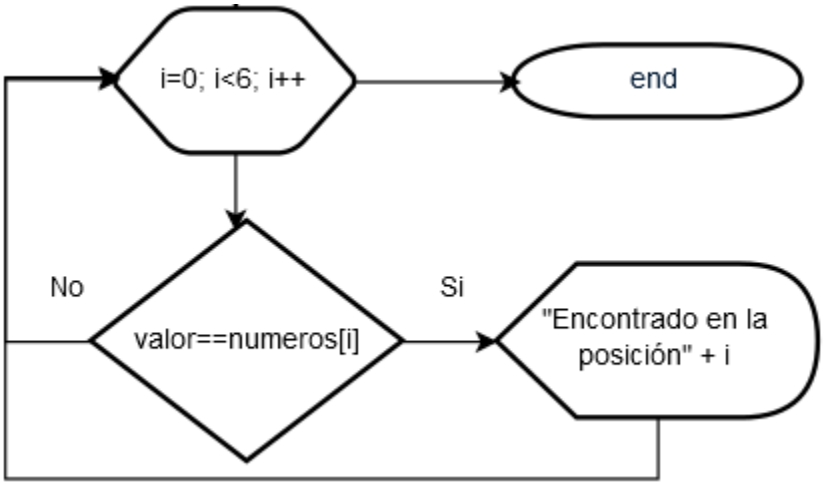


2. Identificación de entradas:

Los valores de entrada son los elementos del vector: `números` y el número a buscar



3. Realizar condición:



4. Identificación de salida
   La salida será las posiciones o índices del valor buscado
5. Fin

Programa en C *Ver código 3arreglos.c anexo en disco*

1. Inicio del programa

```
#include <stdio.h>
int main(){
```

2. Identificación de entradas:

Declarar las variables a utilizar en el programa

```
int numeros[6], i, valor;
```

Solicitar los elementos del arreglo y el valor a buscar:

```
for(i=0; i<6; i++)
     scanf("%d", &numeros[i]);
printf("Numero a buscar:\n");
scanf("%d", &valor);
```

3. Realizar condición:

```
for(i=0; i<6; i++){
     if(valor==numeros[i])
          printf("Encontrado en la posicion: %d\n", i);
}
```

4. Fin

```
}
```

Programas

Realice los siguientes ejercicios:

1. Programa que genere un histograma de una lista de 100 valores, es decir, el programa deberá generar valores aleatorios entre 1 y 10 y guardarlos en un arreglo unidimensional de 100 elementos, posteriormente deberá mostrar la frecuencia de cada valor guardado en dicho arreglo, para ello se recomienda usar otro arreglo unidimensional de 10 elementos ya que los valores aleatorios estarán comprendidos entre 1 y 10.

Entrada:

Arreglo de 100 números aleatorios.

Procedimiento:

1) **Declarar dos arreglos unidimensionales, uno de 100 elementos y otro de 10 elementos.**
2) **Generar 100 valores aleatorios entre 1 y 10 y guardarlos en el arreglo de 100 elementos.**
3) **Contar la frecuencia de cada valor (1 al 10) en el arreglo de 100 elementos, los resultados se pueden guardar en el otro arreglo de 10 elementos.**
4) **Mostrar los resultados de las frecuencias de cada valor.**

Salida:

Suponga que el conjunto aleatorio inicial de 100 elementos es:

```
9     9     4     6     2     8     1     10    3     7
10    8     5     1     6     4     10    4     8     7
7     9     6     2     8     5     2     6     3     7
5     6     5     3     2     8     5     4     2     7
9     9     6     4     8     7     6     4     3     5
4     1     5     4     3     9     3     8     8     6
10    4     2     5     1     10    9     8     10    7
7     1     9     7     5     3     7     6     2     5
4     3     4     7     1     10    2     2     7     2
1     6     10    5     6     4     5     10    9     2
```

El resultado para el ejemplo del conjunto inicial sería:

```
1:    7           6:    11
2:    11          7:    12
3:    8           8:    9
```

```
4:    12          9:    9
5:    12          10:   9

ó

1:   *******
2:   ***********
3:   ********
4:   ************
5:   ************
6:   ***********
7:   ************
8:   *********
9:   *********
10:  *********
```

2. Programa que guarde en un arreglo de 10 elementos enteros, valores introducidos por el teclado y muestre como resultado el mismo arreglo pero sin los elementos repetidos.

Entrada:

Diez valores enteros guardados en el arreglo.

Procedimiento:

1) **Declarar el arreglo de 10 elementos.**
2) **Introducir valores del teclado y guardarlos en el arreglo.**
3) **Recorrer el arreglo en busca de elementos repetidos.**
4) **En caso de que los elementos estén repetidos asignar un 0 a esa posición del arreglo o recorrer los elementos.**
5) **Mostrar los valores del arreglo resultante.**

Salida:

Suponga los valores de entrada del arreglo:

10 4 9 11 4 7 10 30 11 10

El resultado sería:

10 4 9 11 0 7 0 30 0 0

3. En un arreglo bidimensional se almacenan las lluvias mensuales registradas en centímetros, en 10 estados de México (Sonora, Aguascalientes, Oaxaca, Colima, Chihuahua, San Luis Potosí, Durango, Veracruz, Yucatán y Tabasco), del último año. Escribir un programa en C que permita resolver lo siguiente:
   a) El estado que tuvo el mayor registro de precipitación pluvial durante el último año.
   b) El estado que tuvo menor registro de lluvias durante el último año.
   c) El mes que tuvo mayor registro de lluvias del estado de Aguascalientes del último año.

Entrada:

-

Procedimiento:

1) **Declarar el arreglo bidimensional de 10 filas y 12 columnas, cada fila representará un estado y cada columna un mes del año.**
2) **Inicializar la matriz con valores para prueba.**

3) **Realizar las sumatorias correspondientes para obtener el inciso a y b.**
4) **Realizar las comparaciones correspondientes para obtener el inciso c.**
5) **Mostrar los resultados.**

Salida:

Suponga el conjunto inicial para el arreglo bidimensional:

90	90	40	60	20	80	10	10	30	70	50	25
100	80	50	10	60	40	100	40	80	70	100	56
70	90	60	20	80	50	20	60	30	70	110	60
50	60	50	30	20	80	50	40	20	70	200	70
90	90	60	40	80	70	60	40	30	50	10	40
40	10	50	40	30	90	30	80	80	60	90	86
100	40	20	50	10	100	90	80	100	70	130	120
70	10	90	70	50	30	70	60	20	50	40	89
40	30	40	70	10	100	20	20	70	20	90	66
10	60	100	50	60	40	50	100	90	20	15	30

El resultado sería:

```
a) Resultado = Durango
b) Resultado = Sonora
c) Resultado = Enero, Julio, Noviembre
```

4. Escribir un programa que resuelva el siguiente laberinto. La entrada es la coordenada (0,1) y la salida es la coordenada (0,5), los unos representan los obstáculos y los espacios libres se podrían representar por ceros. El usuario introduce “A” para avanzar arriba, “B” para avanzar abajo, “D” para avanzar a la derecha e “I” para avanzar a la izquierda.

   Nota: mostrar cada movimiento.

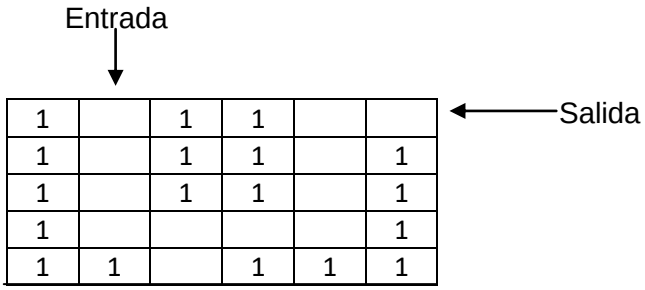


Figura 3.4. Laberinto

Entrada:

Caracteres: A, B, D o I .

Procedimiento:

1) **Declarar una matriz de 5 filas y 6 columnas para el laberinto.**
2) **Inicializar la matriz con los valores respectivos de ceros y unos.**
3) **Introducir del teclado los caracteres A, B, D o I para indicar el movimiento en el laberinto.**
4) **Mostrar la matriz en cada movimiento.**

Salida:

Al llegar a la salida del laberinto el programa mostraría lo siguiente (suponiendo que el valor 2 es el usuario):

1	0	1	1	0	2
1	0	1	1	0	1
1	0	1	1	0	1
1	0	0	0	0	1
1	1	0	1	1	1

```
Haz llegado a la salida, bien hecho.
```

5. Entre los ejemplos de programas que muestran la aplicación de los arreglos se encuentran los siguientes:

   a) Método de la burbuja: Este método permite ordenar de menor a mayor los valores guardados en un arreglo unidimensional, haciendo comparaciones e intercambios durante su ejecución.
   b) Transpuesta en matrices: operación de una matriz que consiste en intercambiar ordenadamente las filas por las columnas.
   c) Determinantes con matrices: el determinante determina la unicidad de la solución de un sistema de ecuaciones lineales, dependiendo el orden de la matriz son las reglas que se utilizan para calcular el determinante.
   d) Inversa en matrices: La matriz inversa de una matriz *A*, es aquella matriz denotada por A^{-1}, tal que: $A \cdot A^{-1} = A^{-1} \cdot A = 1$, es decir, el producto de la matriz *A* y la matriz inversa tanto por la izquierda como por la derecha debe de ser igual a la matriz identidad, existen varios métodos para calcularla como: el método de Gauss-Jordan y el método de la matriz de adjuntos.

A continuación se presenta el código de los primeros dos, usted deberá:
- Probarlos e imprimir la pantalla de ejecución.
- Realizar los programas del inciso c y d.

a) Método de la burbuja (ordena los valores de menor a mayor)

```
#include<stdio.h>
#include<conio.h>
void main()
{
      int a[10],x,y,aux;
      clrscr();
      printf("Introduce los elementos de tu arreglo\n");

      for(x=0;x<10;x++)
            scanf("%d",&a[x]);

      for(y=0;y<9;y++)
      {
            if(a[y]>a[y+1])
            {
                  aux=a[y];
                  a[y]=a[y+1];
                  a[y+1]=aux;
            }
      }

      printf("\n");
```

```
      for(x=0;x<10;x++)
            printf("%d\t",a[x]);
      getch();
}
```

b) Transpuesta en matrices (calcula la transpuesta de una matriz)

```
#include <stdio.h>
#include <conio.h>
void main()
{
int ma[3][3];
int a, b;

for(a=0;a<3;a++)
{
  for(b=0;b<3;b++)
  {
     printf("Introduce el valor de %d");
     scanf("%d",&ma[a][b]);
    }
  }
}

printf("La inversa de la matriz a es..: ");

for(a=0;a<3;a++)
{
  for(b=0;b<3;b++)
  {
     printf("  %d  ",ma[b][a]);
    }
     printf("\n");
  }
}
getch();
}
```

Problema de Aplicación

Haciendo uso de matrices, analice, diseñe e implemente el siguiente problema:

Detección de Incendios. Suponga que se cuenta con una planta de oficinas como la que se muestra en la Figura 3.1. Cada cubículo cuenta con un detector de humo y una bocina. En caso de detectar incendio en una oficina se activarán los rociadores de agua de ese cubículo y los que lo rodean, los cubículos junto a aquellos que tienen activas los rociadores recibirán una alarma sonora en su oficina.

Por ejemplo si se detecta humo en el cubículo A7, se

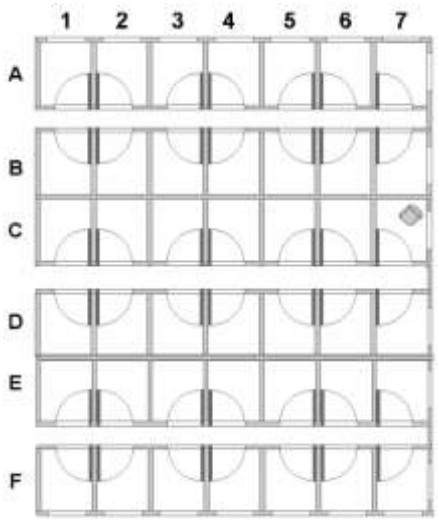


Figura 3.5. Control de incendios en oficinas

activan los rociadores de A6, A7, B6 y B7, así como la alarma de A5, B5, C5, C6 y C7.

Restricciones.

- Se le pregunta al usuario cuántas oficinas y cuáles se están incendiando
- El programa no debe tener programadas todas las condiciones, programe de manera genérica, de tal forma que si el tamaño de la matriz cambia no se tenga que modificar las condiciones.

Sugerencias

- Puede usar la notación de la figura para manipular fácilmente

Proyectos (opcionales).

1. Codificar un programa que implemente el método de burbuja para ordenar un vector de n elementos.
2. Codificar un programa que simule un juego de ajedrez para dos jugadores, donde el tablero esté implementado por una matriz.
3. Codificar un programa que lea una cadena y la muestre al revés.
4. Codificar un programa que encripte una cadena sumándole un 3 al código ASCII de cada uno de sus caracteres.

Ponderación de la Práctica.

Sección	Elemento a Evaluar
Pre-reporte	Ejemplos de arreglos
Ejercicio 1	Uso y aplicación de arreglos unidimensionales
Ejercicio 2	Manejo de arreglos unidimensionales
Ejercicio 3	Uso y aplicación de arreglos bidimensionales
Ejercicio 4	Manejo de arreglos bidimensionales
Ejercicio 5	Aplicación de arreglos en algoritmos como: método de ordenación y operaciones con matrices
Problema	Interpretación del problema, análisis y diseño de la solución e implementación

Bibliografía.

Cairo Osvaldo, Metodología de la programación, Ed. Alfaomega 3ª Impresión, México 2005, ISBN: 970-15-0057-01, Págs: 3- 476.

Joyanes Aguilar Luis , Fundamentos de Programación, Ed Mc Graw Hill, 4ª Impresión, España, 2008, ISBN: 8448161114, Págs: 47-73, 74-75, 76-101, 113-141 y 151 -534.

Guardati, Silvia, Estructura de Datos Orientado a Objetos con C ++, Ed.Prentice, México 2007, ISBN: 9702607922, Págs: 1–183.

Cairó Battistuti, Fundamentos de Programación. Piensa en C, Ed. Prentice Hall, México 2006, ISBN: 970-26-0810-4

Antonakos James, Mansfiel Kenneth, Programación Estructurada en C, Ed. Prentice Hall, Madrid 1997, ISBN: 0-13-520487-9

NOTA: Presentar el reporte en un documento (PDF o DOC), el código fuente y la impresión de la pantalla de ejecución.

"Puedes apoyarte de tus compañeros y profesor para aclarar tus dudas"