UNIVERSIDAD NACIONAL DE LOMAS DE ZAMORA

FACULTAD DE INGENIERÍA

INGENIERÍA MECATRÓNICA

FUNDAMENTOS DE LOS COMPUTADORES DIGITALES

MÓDULO 5 - ARREGLO

CURSO 2024

AUTORES

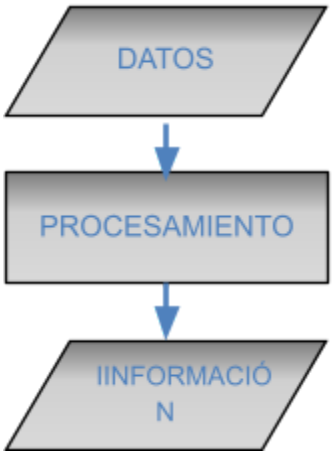
Diego Servetto - Licenciado en Informática (UNLZ)
Ezequiel Blanca - Ingeniero en Electrónica (UTN)
Néstor Maceda - Licenciado en Informática Educativa (UNLa)
Kenneth Gonzalez - Estudiante (UNLZ)
Bianca Rodríguez L. - Estudiante (UNLZ)
Ramiro Gazillón - Estudiante (UNLZ)

ÍNDICE

TEORÍA DE ARREGLOS

Introducción al concepto

En la **unidad1,** se realizó una introducción a los algoritmos y su representación. Y de todos los tipos de algoritmos se focalizó sobre los **algoritmos computaciones** y su representación en diagramas de flujo y pseudocódigo. También se estudió, que los algoritmos computacionales estaban estructurados principalmente en tres etapas: Recolección de **datos**, **procesamiento** de operaciones y finalmente exhibición de la **información**.



Sobre estas tres etapas (datos, proceso e información), la ciencia de computación, trabaja para obtener el mayor rendimiento en los recursos de los sistemas computacionales, y así brindar a las distintas disciplinas, información automática y eficiente que servirá para la obtención y construcción de conocimiento.

En función a dicha labor, se han creado una infinidad de **lenguajes de programación** orientados a los distintos recursos o bloques del sistema. Por ejemplo hay lenguajes de programación orientados a la **información** (su recuperación, su consulta, su categorización y clasificación) muy empleado por archivistas, bibliotecólogos, museólogos, etc que trabajan con grandes cantidades de información. Otros lenguajes de programación están orientados a favorecer el gran manejo de **datos** (aplicados a Data Mning, Big Data, Bussines, Ciencia, etc.) y finalmente otros, están orientados al bloque de **procesamiento** (por ejemplo el conocido lenguaje **Fortran**, un lenguaje creado para matemática e ingeniería que podía realizar casi todo tipo de cálculo en forma rápida y eficiente)

Por otro lado, hay lenguajes orientados al hardware y sus procesos, como el lenguaje ensamblador o **lenguaje C de multipropósito,** que no poseen herramientas avanzadas para el tratamiento automático de **datos** o gestión de la **información**.

Organización de datos

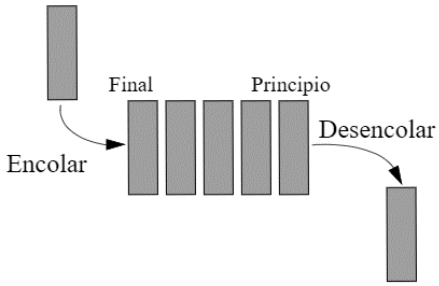
En informática, para trabajar con grandes cantidades de **conjuntos de datos**, utiliza distintos tipos de técnicas que permiten arreglarlos o estructurarlos[1]. Esta organización de datos, se clasifica principalmente en dos formas: lineales o no lineales.

En forma lineal, es en forma de listas, armando una fila de datos. Ejemplo de estos son las **colas**, las **pilas** y los **arreglos**. Y en forma no lineal de organizar los datos son las demás

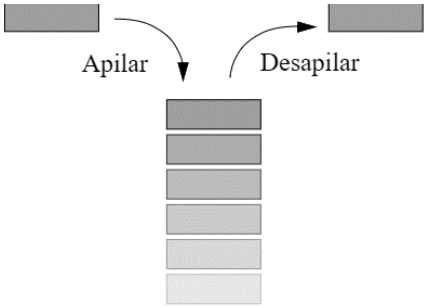
[1] Estructura de datos, no es un contenido de la presente cátedra. Solo se hace una introducción a los efectos de poner en contexto al lector para una mejor comprensión de los temas objetivo.

formas que no organizan los datos en una línea o fila, estas son: las **matrices**, los **árboles** y los **grafos**.

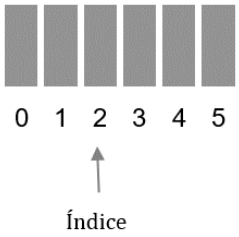
A modo de ejemplo, se describen las formas lineales para comprender como se estructuran los datos de grandes cantidades. La lista tipo **cola**, es una estructura de longitud fija la cual se denomina **FIFO** (First Input First Output) el primro en entrar es el primero en salir. Los datos entran por un extremo y salen por el otro extremo. Es el caso típico de una fila de personas en un comercio cuando vamos a pagar las cuentas, el primero en entrar será el primero en ser atendido y los que van ingresando se acomodan al final de fila.



En cambio en una lista tipo **pila**, también de longitud o tamaño fijo, lo elementos se van apilando uno arriba del otro. Su nombre **LIFO** (Last Input First Output) último en entrar primero en salir.Es el caso típico de un vagón de tren. Las primeras personas que entraron van sentadas y las últimas que entraron irán paradas pero serán las primeras en salir. Las personas sentadas que entraron primero no podrán salir hasta que las paradas salgan primero.



Finalmente están las listas tipo **arreglo**, es el caso de una estantería. Los estantes ya están presentes y los datos se van acomodando en los mismos. Cada estante se ubica por una posición denominado índice.



Estas son formas de organizar y trabajar con grandes cantidades de datos en programación.

Organización de datos en C

¿Cómo haríamos en lenguaje C, para manejar grandes cantidades de datos?

Por suerte, en general, todos los lenguajes de programación, cuentan con más o menos herramientas fundamentales para el tratamiento de cantidades de datos. Por ejemplo, una herramienta específica y común empleada en los lenguajes de programación (estudiada incluso en algoritmos) es el **ARREGLO.** Si bien, el **ARREGLO**, es una herramienta que nos permite organizar y trabajar con cantidades de datos, en lenguaje C, sus funciones no están totalmente automatizadas, siendo el programador, el que deberá buscar la mejor estrategia, para su tratamiento eficiente.

¿Qué es un ARREGLO?

El **ARREGLO**, como herramienta en la creación de algoritmos, es una colección (conjunto o grupo) de datos de un mismo tipo y de una cantidad fija, organizados en forma lineal, cuya posición en la lista de datos implica un número de orden (es una lista ordenada en función a su posición).

Un ejemplo, dado el **conjunto**[2] de datos A = {**28**, 37, 45, 53, 7, 31, **9**}, se puede considerar un ARREGLO porque tiene una longitud fija (siete elementos) y todos sus elementos constitutivos son del mismo tipo. Para referirnos a cada uno de ellos lo hacemos en forma ordenada, A = {a_1, a_2, a_3, a_4, a_5, a_6, a_7}, siendo a_1 el **28** y finalmente a_7 representa al **9**.

Otro ejemplo podría ser A = {'a', 'r', 'd', 's', 'p'}. Si es un ARREGLO porque todos los elementos son caracteres, es de una longitud fija y también para referirnos a cada elemento tenemos en cuenta su posición en la lista. A = {a_1, a_2, a_3, a_4, a_5}.

	A
1	Nombre
2	Sonia
3	Ignacio
4	Mauricio
5	Roberto
6	Lorena
7	Jorge
8	Brenda
9	Teresa

[2] Los conjuntos de elementos se representan entre llaves '{', '}'

Una clara representación de ARREGLO (como muestra la figura), puede ser una lista de nombres en **una columna**, donde cada nombre tiene una posición asignada.

En contraposición, hay conjuntos de datos que no pueden denominarse ARREGLO si no cumplen con las condiciones preestablecidas, ejemplo: A = {2, 3, False, 4,'a', 3,'b', True, "hola"}, no es un ARREGLO puesto que sus elementos no son del mismo tipo. Otro ejemplo podría ser: A = {1, 2, 3,…}, no es un ARREGLO porque no tiene una cantidad finita de elementos.

Definición de arreglo

Un ARREGLO (del inglés formación) es una tipo de dato estructurado, constituida por una colección de elementos que se caracteriza por:

- **Finito**: Es una colección que tiene un número definido de elementos
- **Homogéneo**: Todos sus elementos son del mismo tipo.
- **Ordenado**: Se encuentran almacenados en forma secuencial.

Por su analogía con la **matemática** un ARREGLO es una **matriz lineal**[3]: A = ($a_{(1)}$, $a_{(2)}$, $a_{(3)}$, $a_{(4)}$), independientemente si esta es una matriz columna o matriz fila.

$$A = \begin{pmatrix} 10 \\ 7 \\ 8 \\ 15 \end{pmatrix} \qquad A = \begin{pmatrix} 1 & 0 & -4 & 9 \end{pmatrix}$$

El término arreglo

Debido a la cantidad de aplicaciones que tiene un ARREGLO en todas las disciplinas científicas y tecnológicas, se observa una infinidad de formas respecto a su denominación y naturalmente también en su definición. Encontramos los siguientes términos análogos a ARREGLOS:

- **ARRAY** = Término del inglés empleado para denominar al ARREGLO, ARRAY es una organización o formación ordenada de elementos.
- **ARRAYS** = es el plural de ARRAY en inglés y se utiliza principalmente como la traducción de matriz. Depende de los autores algunos definen ARREGLO como array y otros en su forma plural de ARRAYS
- **MATRIX** = palabra de matriz en inglés.

[3] Las matrices como conjunto de elementos ordenados se representa entre paréntesis '(',')'

- **MATRIZ** = término empleado en el **álgebra lineal**.
- **PLANILLA** = empleado en gestión o administración.
- **LISTA** = empleado en gestión o administración.
- **VECTOR** = Término que proviene de física, ingeniería mecánica o de tensores en ingeniería civil.

¿Por qué lo denominamos ARREGLO?

Lo denominamos arreglo porque así lo denomina nuestra **bibliografía** de referencia en la presente cátedra: ***"El lenguaje de programación ANISI C de Brian W. Kernighan y Dennis M. Ritchie".*** Por otra parte ARREGLO hace referencia a arreglar, ordenar, organizar un conjunto de datos. Algunos autores, comentan que ARREGLO es una traducción errónea de ARRAY.
Sin embargo, es común entre los programadores emplear el término ARREGLO.

Operaciones elementales en arreglos

Suponiendo un ARREGLO de 6 elementos A = (31, 52, 18. 73. 20. 79). Las operaciones fundamentales que se puede realizar con un arreglo son:

Vaciar un arreglo

Es un algoritmo que pone todos los elementos del arreglo en un valor nulo. Por ejemplo en cero si se trata de un arreglo de enteros.

A = (31, 52, 18. 73. 20**.** 79) //arreglo original
A = (0, 0, 0. 0. 0**.** 0) //arreglo inicializado

Insertar un elemento

Es el algoritmo que agrega un valor a una posición determinada del arreglo.

A = (0, 0, 0, 0, 0) //arreglo original.
$a_{(2)}$ = 45 //en la posición 2 se asigna el valor 45.
A = (0, 45, 0, 0, 0) //como ya tenía en la posición 2 el valor 52 se sobre escribe el 45.

En el caso que dicha posición ya tenga un valor se sobrescribe.

A = (31, **52**, 18. 73. 20. 79) //arreglo original.
$a_{(2)}$ = 45 //en la posición 2 se asigna el valor 45.

A = (31, **45**, 18. 73. 20. 79) //como ya tenía en la posición 2 el valor 52 se sobre escribe el 45.

Invertir un arreglo

Es un algoritmo que se encarga de poner los últimos elementos al principio y los que están al principio los coloca al final.

A = (31, 52, 18. 73. 20**.** 79) //arreglo original
A = (79, 20, 73. 18, 52, 31) //arreglo invertido

Borrar un elemento

Borrar un elemento del arreglo en realidad es sobre escribirlo con algún valor que consideremos nulo. Por ejemplo borramos el valor en la posición 5.

A = (31, 52, 18. 73. **20.** 79) //arreglo original
$a_{(5)}$ = 0 //a la posición 5 se le asigna un valor nulo.
A = (31, 52, 18. 73. **0**. 79) //posición borrada.

Localizar un elemento

Consiste en un algoritmo que busque un contendió determinado y al encontrarlo me devuelva en la posición que se encuentra. Si hay más de uno con el mismo valor devuelve el primero que encuentra. Si no encuentra el valor devuelve un valor negativo.

A = (31, 52, 18. 73. 20**.** 79) //arreglo original

Buscar el 18 //busca el valor 18
Respuesta = 3 //devuelve la posición en la que se encuentra
Buscar el 67 //busca el valor 67
Respuesta = -1 //al no encontrarse el valor en el arreglo devuelve -1

Ordenar arreglo

Es el algoritmo que ordena internamente los elementos de un arreglo. Este se puede realizar en forma ascendente o descendente.

A = (31, 52, 18. 73. 20**.** 79) //arreglo original
A = (18, 20, 31. 52, 73, 79) //arreglo ordenado en forma ascendente

Clasificación de arreglos

Según su contenido

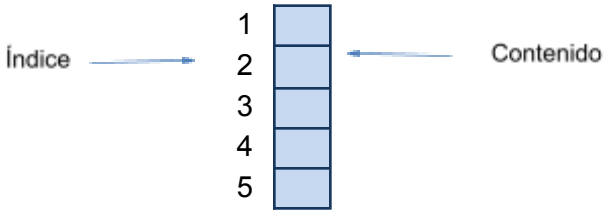
- **Constantes**
  Los elementos definidos no pueden modificarse en todo el algoritmo donde se utilice el arreglo.
- **Variables**
  Los elementos definidos se pueden modificar durante proceso del algoritmo.

Según su tamaño

- **Estáticos**
  Son arreglos de tamaño fijo.
- **Dinámicos**
  Son arreglos que pueden modificar su tamaño durante el transcurso del programa.

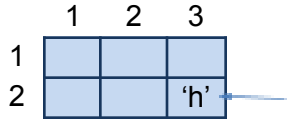
Según su dimensión

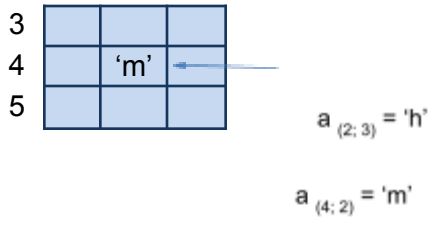
- **Unidimensional**
  Son los arreglos estudiados, conocidos como lineales o de una dimensión. Estos también se denominan **vectores**.



- **Multidimensional**

Son arreglos de dos o más dimensiones. Estos son conocidos como **matrices**. Por ejemplo se ilustra un arreglo de dos dimensiones. Constituido de 5 filas y 3 columnas. Para referirnos a un elemento se indica primero la fija y luego la columna

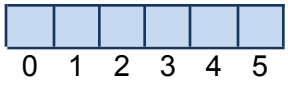




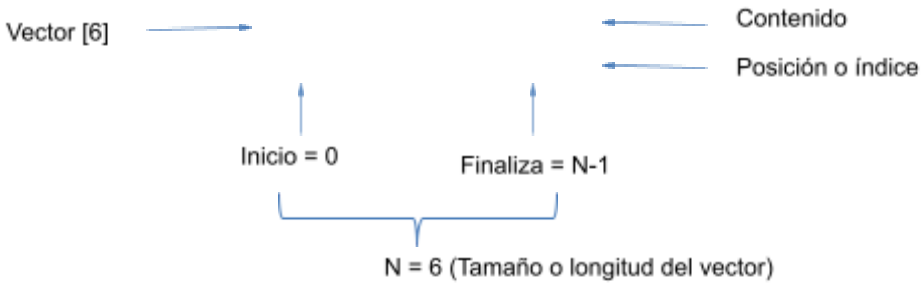
ARREGLOS EN C

Vectores

Estructura del dato



En lenguaje C, los vectores o arreglos unidimensionales se almacenan en memoria forma contigua y el índice o posición inicial es el valor cero (0). Por lo tanto, un vector de N elementos irá desde la posición 0 hasta la N-1. Y el índice no se marca indica entre corchetes [i].
Por ejemplo, dado el vector que contiene N = 6 elementos, comienza en 0 y finaliza en N-1 = 5



Declaración de un vector

La declaración de un arreglo unidimensional o vector en lenguaje C, se encuentra constituido de la siguiente forma:

<Tipo><Nombre> [Tamaño];

Donde:

<Tipo>: es el tipo de dato de los datos que va a contener el arreglo, por ejemplo: int, float, double, char, etc.
<Nombre>: Es un nombre de variable único que va a representar al conjunto de datos.

[Tamaño]: indica la cantidad de elemento que va contener el vector.

Ejemplo:

```
int edades [30];        //vector para guardar las edades de 30 personas del  0 al 29
char sexo[2];           //para indicar el sexo 'm' de masculino y 'f' de femenino.
float precio [20];      //para almacenar la lista de precios de 20 productos
```

Inicialización de un vector

En la declaración

La inicialización de un vector se puede realizar en la misma declaración, seguido de un signo igual. Por ejemplo:

```
char sexo[2] = {'m','f'};
char vocales[5] = {'a', 'e', 'i', 'o', 'u'};
float notas[3] = {6.50, 8.20, 10.00};
```

En el programa

O bien durante el cuerpo del programa asignado a cada posición su valor

```
…
sexo [0] = 'm';
sexo [1] = 'f';
…
```

También se puede inicializar ingresando los datos manualmente desde el teclado

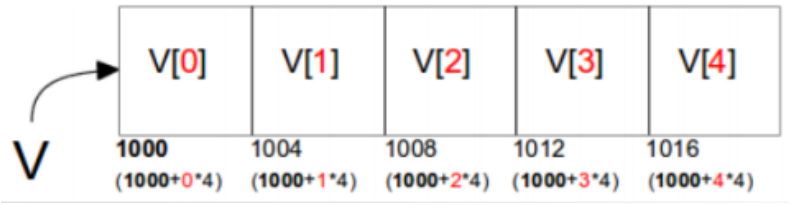
```
…
Printf ("\n Ingrese el carácter para masculino");
Scanf ("%c", &caracter);
sexo [0] = caracter;
printf ("\n Ingrese el carácter para masculino");
scanf ("%c", &caracter);
sexo [1] = caracter;
…
```

O mediante un ciclo **for**:

```
…
//Borrar el vector
for (i=0, i<=10, i++)
{
        vector[i] = 0;
}
…
```

Almacenamiento de un vector

Para lenguaje C un arreglo no es más que un sector de memoria contigua con capacidad para guardar la cantidad de elementos declarados. Así si declaro int v[5] guarda lugar en memoria para almacenar 5 **int** consecutivos.



Esto explica, porqué el primer subíndice es el cero, ya que el primer elemento es el que está desplazado cero “elementos” con respecto al inicio del vector.

Vector constante

Es un vector que no se puede modificar los valores asignados una vez realizada su declaración. Para lograr esto se emplea la palabra reservada const de constante.
Por ejemplo:

```
const int vec[3] = {35,3,12};
…
…
Vec[2] = 7 //da error el todo intento de modificar un valor en el vector.
…
```

Aplicación

Si debiera, por algún motivo, almacenar la edad de 100 personas requeriría por lógica, 100 variables. Una por cada nombre.

```
int edad1;
int edad2;
int edad3;
int edad4;
…
…
int edad100;
```

Naturalmente, esta forma de programar es poco práctica. Imagina que si debiera almacenar en memoria unos 500 datos para realizar cálculos estadísticos. ¿Debería declarar 500 variables?, ciertamente sí.
Incluso, no solo declarar es complicado, sino también asignar los valores a cada una de las variables

```
edad1 = 34
edad2= 21
edad3 = 19
…
…
edad100 = 40
…
…
edad500 = 23
```

El vector ofrece una solución práctica para el manejo de datos en gran escala. El vector, es un tipo de variable que permite con un solo nombre, definir todo el conjunto de datos.

```
int edad[500];
```

y la carga se puede realizar con un simple ciclo **for**:

```
…
        //Cargar un vector con 100 nombres
        for (i=0, i<=99, i++)
        {
                printf (“\n Ingrese el nombre[%d]: ”,i);
                scanf(“%c”, nombre[i]);
```

```
    }
    …
```

Ejemplo

Se presenta un código en C, donde se observan las operaciones básicas con un vector

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

#define TAM_VECTOR 10

int main()
{
   int opcion_menu;
   double resultado_suma=0;
   double vnum[TAM_VECTOR];
   enum opciones {CARGAR=1, IMPRIMIR, SUMAR, SALIR};

   do
   {
      printf("\nMenu de Opciones:\n");
      printf("1) Cargar datos\n");
      printf("2) Imprimir datos \n");
      printf("3) Sumar Valores \n");
      printf("4) Salir \n");
      printf("\nElija una opcion: ");
      scanf("%d",&opcion_menu);

      switch (opcion_menu)
      {
      case CARGAR:
```

```
            for (int pos=0; pos<TAM_VECTOR; pos++)
            {
               printf("Ingresar datos para la posicion %d: ", pos);
               scanf("%lf", &vnum[pos]);
            }
            break;
         case IMPRIMIR:
            for (int pos=0; pos<TAM_VECTOR; pos++)
            {
               printf("La posicion: %d tiene el valor: %lf \n", pos, vnum[pos]);
            }
            break;
         case SUMAR:
            for (int pos=0; pos<TAM_VECTOR; pos++)
            {
                     // idem resultado_sumatoria = resultado_sumatoria +   vector[pos]
                      resultado_suma += vnum[pos];
            }
            printf("La suma de todos los elementos es: %lf", resultado_suma);
            break;
         case SALIR:
            printf("Saliendo...\n\n");
            break;
         default:
            printf("Opcion incorrecta, vuelva a intentar\n\n");
         }
      }
      while (opcion_menu!=4);

      return 0;
}
```

Matriz

Estructura del dato

En lenguaje C, las matrices o arreglos bidimensionales, representados como planillas, o tablas, no existen como tal. Sino que es un vector de vector. Tal como se muestra en la figura:

Tenemos un Vector de tamaño tres que a su vez internamente tiene otro vector de tamaño 5. Pero para los fines prácticos, es aceptable realizar una abstracción del mismo y considerarlo como una tabla compuesta por filas y columnas.

Declaración de una matriz

La declaración de un arreglo bidimensional o matriz en lenguaje C, se encuentra constituido de la siguiente forma:

<Tipo><Nombre> [Fila][Columna];

Donde:

<Tipo>: es el tipo de dato de los datos que va a contener el arreglo, por ejemplo: int, float, double, char, etc.
<Nombre>: Es un nombre de variable único que va a representar al conjunto de datos.
[Fila]: indica la cantidad de filas, líneas o renglones de la matriz.
[Columna]: Indica el número de columnas de la matriz, tabla o planilla
.

Ejemplo:

```
int numeros [3][5];      //matriz de enteros de 3 filas y 5 columnas
```

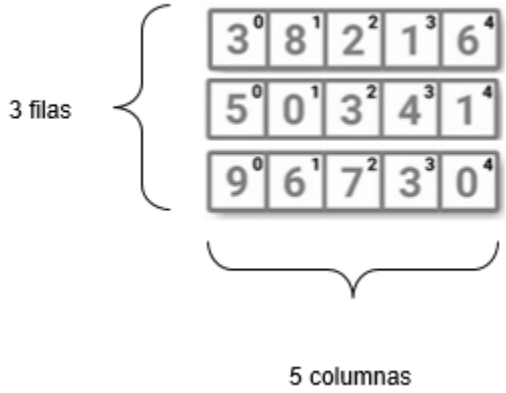
Inicialización de un vector

En la declaración

- Forma 1: **int** numeros [3][2] = {{3, 8, 2, 1, 6},{5, 0, 3, 4, 1},{9, 6, 7, 3, 0}};
- Forma 2: **int** numeros [3][2] = {3, 8, 2, 1, 6, 5, 0, 3, 4, 1, 9, 6, 7, 3, 0};

En el programa

```
numeros [0][0] = 3;
numeros [0][1] = 8;
numeros [0][2] = 2;
numeros [0][3] = 1;
numeros [0][4] = 6;
…
…
numeros [2][3] =3;
numeros [2][4] =0;
…
```



Naturalmente este ingreso también se puede realizar con el ciclo for.

Ejemplo

```
int main(){
    int matriz[2][3] = {51,52,53 , 54,55,56};
    int filas,columnas;

    for(filas=0;filas<2;filas++){
        for(columnas=0;columnas<3;columnas++){
            printf("%i ",matriz[filas][columnas]);
        }
        printf("\n");
    }



    getch();
    return 0;
```