

Lenguaje C, cuarto bloque: Arrays

José Otero

¹Departamento de informática
Universidad de Oviedo

20 de enero de 2008

Índice

1 Introducción

2 Arrays Unidimensionales

- Concepto
- Declaración
- Uso de Arrays Unidimensionales
- Recorrido de Arrays Unidimensionales
- Paso de Arrays Unidimensionales a funciones

3 Arrays Bidimensionales

- Concepto
- Declaración
- Uso de Arrays Bidimensionales
- Recorrido de Arrays Bidimensionales
- Paso de Arrays Bidimensionales a funciones

Índice

1 Introducción

2 Arrays Unidimensionales

- Concepto
- Declaración
- Uso de Arrays Unidimensionales
- Recorrido de Arrays Unidimensionales
- Paso de Arrays Unidimensionales a funciones

3 Arrays Bidimensionales

- Concepto
- Declaración
- Uso de Arrays Bidimensionales
- Recorrido de Arrays Bidimensionales
- Paso de Arrays Bidimensionales a funciones

Un array es una estructura homogénea:

- Compuesta por varias componentes.
- Todas del *mismo tipo*, `int`, `char`, `float`,...
- Almacenadas *consecutivamente* en memoria.
- Cada componente puede ser accedido directamente por el nombre de la variable array seguido de uno o varios subíndices encerrados entre corchetes. Los subíndices comienzan en cero.
- No existen operaciones definidas sobre arrays *completos*, deben descomponerse en acciones sobre cada componente.
- En el standard que se estudia en este curso, el tamaño es *constante y conocido en tiempo de compilación*: no puede cambiarse durante la ejecución del programa.

Cuando se declara un array se establece:

- Su tipo: `int`, `char`, `float`,...
- Su nombre, un identificador.
- Su número de dimensiones, igual al número de parejas de corchetes que se escriben después del nombre.
- Su extensión (tamaño) en cada una de las dimensiones, se encierra entre cada pareja de corchetes.

Los arrays de una dimensión son la representación de los vectores utilizados en matemáticas o física.

Los arrays de dos dimensiones son la representación de las matrices.

No existe limitación en cuanto al número de dimensiones

Índice

1 Introducción

2 Arrays Unidimensionales

- Concepto
- Declaración
- Uso de Arrays Unidimensionales
- Recorrido de Arrays Unidimensionales
- Paso de Arrays Unidimensionales a funciones

3 Arrays Bidimensionales

- Concepto
- Declaración
- Uso de Arrays Bidimensionales
- Recorrido de Arrays Bidimensionales
- Paso de Arrays Bidimensionales a funciones

Índice

1 Introducción

2 Arrays Unidimensionales

■ Concepto

■ Declaración

■ Uso de Arrays Unidimensionales

■ Recorrido de Arrays Unidimensionales

■ Paso de Arrays Unidimensionales a funciones

3 Arrays Bidimensionales

■ Concepto

■ Declaración

■ Uso de Arrays Bidimensionales

■ Recorrido de Arrays Bidimensionales

■ Paso de Arrays Bidimensionales a funciones

- Un array unidimensional es la representación de un *vector*.
- Cada elemento de un vector se identifica con un índice.
- A todos los efectos, un elemento de un vector se comporta como una variable del tipo con el que se construye el vector.
- No existen operaciones predefinidas, ni de E/S, ni asignación, ni aritméticas, lógicas, relacionales, etc.
- El índice comienza en cero.
- El índice del último elemento es igual al tamaño menos uno.

Índice

1 Introducción

2 Arrays Unidimensionales

- Concepto

- **Declaración**

- Uso de Arrays Unidimensionales

- Recorrido de Arrays Unidimensionales

- Paso de Arrays Unidimensionales a funciones

3 Arrays Bidimensionales

- Concepto

- Declaración

- Uso de Arrays Bidimensionales

- Recorrido de Arrays Bidimensionales

- Paso de Arrays Bidimensionales a funciones

```
tipo nombre[tamano];
```

- `tipo` es uno de los tipos predefinidos en C: `int`, `float`, `char`..
- `nombre` es un identificador.
- `tamano` es una expresión constante, conocida en tiempo de compilación.

Formas de definir el valor del tamaño l.

- Mediante una expresión constante, sin mas:

```
//declara un vector de reales  
//de tamaño 10 y otro de tamaño 45  
float a[10], b[3*15];
```

```
//declara un vector de caracteres  
//de tamaño 25  
char mensaje[25];
```

Formas de definir el valor del tamaño II.

- Mediante una constante definida con `#define`:

```
//Se define el tamaño
//al principio del fuente
//pueden definirse varias ctes
#define tam 10
#define lon 30
...
//en donde se declaren los
//vectores, se escribe tam o lon
//en lugar de 10 o 30
int x[tam],y[lon];
```

El preprocesador sustituye en el fuente `tam` y `lon` por 10 y 30 resp.

Facilita la modificación de los programas y evita errores.

Formas de definir el valor del tamaño III.

■ Mediante una variable constante:

```
//Se define el tamaño como una
//variable constante global
//pueden definirse varias ctes
//esto se escribe fuera de main
const int tam=10;
const int lon=30;
...
//en donde se declaren los
//vectores, se escribe tam o lon
//en lugar de 10 o 30
int x[tam],y[lon];
```

Facilita la modificación de los programas y evita errores.

Índice

1 Introducción

2 Arrays Unidimensionales

- Concepto

- Declaración

- **Uso de Arrays Unidimensionales**

- Recorrido de Arrays Unidimensionales

- Paso de Arrays Unidimensionales a funciones

3 Arrays Bidimensionales

- Concepto

- Declaración

- Uso de Arrays Bidimensionales

- Recorrido de Arrays Bidimensionales

- Paso de Arrays Bidimensionales a funciones

...nombre[expresion]...

- Los [] encierran el índice que especifica un elemento del vector.
- El índice puede ser cualquier expresión de tipo entero que evalúe a 0..tamaño-1.
 - La expresión puede *variar* durante la ejecución del programa.
- Se puede escribir lo anterior en cualquier sitio en el que se pueda escribir una variable del tipo del vector.
- Cada elemento se comporta como una variable del tipo del vector.

```
#include<stdio.h>
int main()
{
int x[5],a=2,b=3; //declaración
x[4]=43;         //diversas asignaciones
x[a]=b*2+1;
a=a+2;
b=x[2]+x[a]*3; //uso en expresiones
//petición por teclado de algún elemento
scanf("%d%d",&x[1],&x[3]);
x[0]=x[1]+x[3];
//mostrar por pantalla algún elemento
printf("%d",x[3]);
}
```


Índice

1 Introducción

2 Arrays Unidimensionales

- Concepto
- Declaración
- Uso de Arrays Unidimensionales
- **Recorrido de Arrays Unidimensionales**
- Paso de Arrays Unidimensionales a funciones

3 Arrays Bidimensionales

- Concepto
- Declaración
- Uso de Arrays Bidimensionales
- Recorrido de Arrays Bidimensionales
- Paso de Arrays Bidimensionales a funciones

- El recorrido de un vector consiste en aplicar una misma operación a cada elemento del mismo, de forma consecutiva.
- Para ello se generan todos los índices válidos del vector en secuencia.
- Una forma cómoda es utilizar un bucle `for`.

```
float a[tam];  
...  
//del primero al ultimo  
for (i=0;i<tam;i++)  
    {  
    //para cada valor de i  
    //hacer algo con a[i]  
    ...a[i]...  
    }
```

```
float a[tam];  
...  
//del ultimo al primero  
for (i=tam-1;i>=0;i--)  
    {  
    //para cada valor de i  
    //hacer algo con a[i]  
    ...a[i]...  
    }
```

```

#include<stdio.h>
const int tam=10;
int main()
{
//a, b, c son vectores
float a[tam],b[tam],c[tam];
//i para el for, n para definir cuantos
//de tam se van a usar
int i,n;

//se pide n asegurando que no sea
//mayor que tam
printf("\nCuantos elementos vas a usar?");
do{
scanf("%d",&n);
}while(n>tam);

//pedir el primer vector
//solo los elementos que
//se van a usar
for (i=0;i<n;i++)
{
//se informa del elemento que se pide
printf("Introduce elemento [%d]=",i);
//se pide el elemento i-esimo
scanf("%f",&a[i]);
}

//pedir el segundo vector
//solo los elementos que
//se van a usar
for (i=0;i<n;i++)
{
printf("Introduce elemento [%d]=",i);
scanf("%f",&b[i]);
}

//calcular la suma de a y b
//cada elemento de c es la
//suma de los elementos correspondientes
//solo se utilizan los elementos
//que se han pedido
for (i=0;i<n;i++)
c[i]=a[i]+b[i];

//mostrar el resultado en una sola linea
//solo los elementos calculados
for (i=0;i<n;i++)
printf("%f ",c[i]);

//mostrar el resultado indicando
//el indice, un valor en cada linea
for (i=0;i<n;i++)
printf("\n[%d]=%f ",i,c[i]);
}

```

Índice

1 Introducción

2 Arrays Unidimensionales

- Concepto
- Declaración
- Uso de Arrays Unidimensionales
- Recorrido de Arrays Unidimensionales
- Paso de Arrays Unidimensionales a funciones

3 Arrays Bidimensionales

- Concepto
- Declaración
- Uso de Arrays Bidimensionales
- Recorrido de Arrays Bidimensionales
- Paso de Arrays Bidimensionales a funciones

- El nombre de un vector, sin `[]`, es un puntero al tipo del vector.
- Su valor es la dirección de memoria en donde se encuentra el primer elemento.
- El contenido de esa dirección de memoria es el primer elemento.
- Los elementos siguientes ocupan posiciones contiguas.
- Si se declara `tipo nombre[tam];` entonces `*(nombre+i)` es lo mismo que `nombre[i]`.
- Cuando se pasa un vector como parámetro a una función, se hace implícitamente por referencia, por lo tanto se puede modificar.
 - El tamaño se puede omitir en la declaración de los parámetros formales.

```

#include<stdio.h>
#define tam 10
//calcula la media de los elementos
float media(float a[],int n)
{
int i;
float s=0;
for (i=0;i<n;i++)
    s=s+a[i];
return s/n;
}

```

```

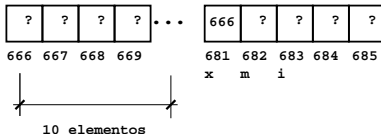
//inicializa los elementos a cero
void inicializa(float a[],int n)
{
int i;
for (i=0;i<n;i++)
    a[i]=0;
}

```

```

int main()
{
float x[tam],m; ←
int i; ←
x[0]=1.2;x[1]=-0.1;x[2]=0.0;x[3]=5.5;
m=media(x,4);
printf("%f\n",m);
inicializa(x,4);
for (i=0;i<4;i++)
    printf("%f ",x[i]);
}

```



```

#include<stdio.h>
#define tam 10
//calcula la media de los elementos
float media(float a[],int n)
{
int i;
float s=0;
for (i=0;i<n;i++)
    s=s+a[i];
return s/n;
}

```

```

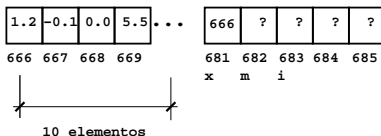
//inicializa los elementos a cero
void inicializa(float a[],int n)
{
int i;
for (i=0;i<n;i++)
    a[i]=0;
}

```

```

int main()
{
float x[tam],m;
int i;
x[0]=1.2;x[1]=-0.1;x[2]=0.0;x[3]=5.5;
m=media(x,4);
printf("%f\n",m);
inicializa(x,4);
for (i=0;i<4;i++)
    printf("%f ",x[i]);
}

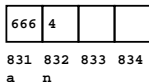
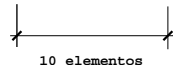
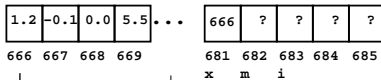
```



```
#include<stdio.h>
#define tam 10
//calcula la media de los elementos
float media(float a[],int n)
{
int i;
float s=0;
for (i=0;i<n;i++)
    s=s+a[i];
return s/n;
}
```

```
//inicializa los elementos a cero
void inicializa(float a[],int n)
{
int i;
for (i=0;i<n;i++)
    a[i]=0;
}
```

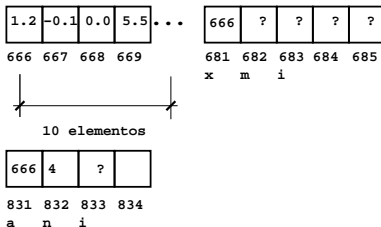
```
int main()
{
float x[tam],m;
int i;
x[0]=1.2;x[1]=-0.1;x[2]=0.0;x[3]=5.5;
m=media(x,4);
printf("%f\n",m);
inicializa(x,4);
for (i=0;i<4;i++)
    printf("%f ",x[i]);
}
```




```
#include<stdio.h>
#define tam 10
//calcula la media de los elementos
float media(float a[],int n)
{
  int i;           ←
  float s=0;
  for (i=0;i<n;i++)
    s=s+a[i];
  return s/n;
}
```

```
//inicializa los elementos a cero
void inicializa(float a[],int n)
{
  int i;
  for (i=0;i<n;i++)
    a[i]=0;
}
```

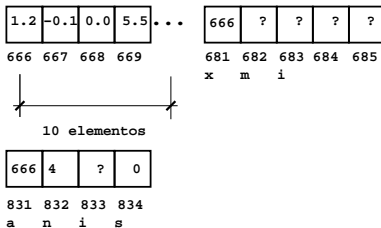
```
int main()
{
  float x[tam],m;
  int i;
  x[0]=1.2;x[1]=-0.1;x[2]=0.0;x[3]=5.5;
  m=media(x,4);
  printf("%f\n",m);
  inicializa(x,4);
  for (i=0;i<4;i++)
    printf("%f ",x[i]);
}
```



```
#include<stdio.h>
#define tam 10
//calcula la media de los elementos
float media(float a[],int n)
{
int i;
float s=0; ←
for (i=0;i<n;i++)
    s=s+a[i];
return s/n;
}
```

```
//inicializa los elementos a cero
void inicializa(float a[],int n)
{
int i;
for (i=0;i<n;i++)
    a[i]=0;
}
```

```
int main()
{
float x[tam],m;
int i;
x[0]=1.2;x[1]=-0.1;x[2]=0.0;x[3]=5.5;
m=media(x,4);
printf("%f\n",m);
inicializa(x,4);
for (i=0;i<4;i++)
    printf("%f ",x[i]);
}
```



```

#include<stdio.h>
#define tam 10
//calcula la media de los elementos
float media(float a[],int n)
{
int i;
float s=0;
for (i=0;i<n;i++)
    s=s+a[i];
return s/n;
}

```

```

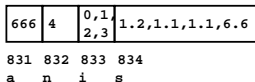
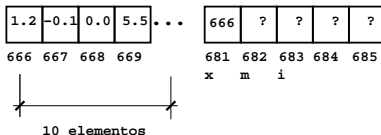
//inicializa los elementos a cero
void inicializa(float a[],int n)
{
int i;
for (i=0;i<n;i++)
    a[i]=0;
}

```

```

int main()
{
float x[tam],m;
int i;
x[0]=1.2;x[1]=-0.1;x[2]=0.0;x[3]=5.5;
m=media(x,4);
printf("%f\n",m);
inicializa(x,4);
for (i=0;i<4;i++)
    printf("%f ",x[i]);
}

```



```

#include<stdio.h>
#define tam 10
//calcula la media de los elementos
float media(float a[],int n)
{
int i;
float s=0;
for (i=0;i<n;i++)
    s=s+a[i];
return s/n;
}

```

```

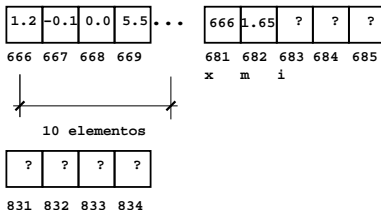
//inicializa los elementos a cero
void inicializa(float a[],int n)
{
int i;
for (i=0;i<n;i++)
    a[i]=0;
}

```

```

int main()
{
float x[tam],m;
int i;
x[0]=1.2;x[1]=-0.1;x[2]=0.0;x[3]=5.5;
m=media(x,4);
printf("%f\n",m);
inicializa(x,4);
for (i=0;i<4;i++)
    printf("%f ",x[i]);
}

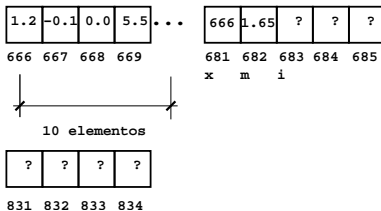
```



```
#include<stdio.h>
#define tam 10
//calcula la media de los elementos
float media(float a[],int n)
{
  int i;
  float s=0;
  for (i=0;i<n;i++)
    s=s+a[i];
  return s/n;
}
```

```
//inicializa los elementos a cero
void inicializa(float a[],int n)
{
  int i;
  for (i=0;i<n;i++)
    a[i]=0;
}
```

```
int main()
{
  float x[tam],m;
  int i;
  x[0]=1.2;x[1]=-0.1;x[2]=0.0;x[3]=5.5;
  m=media(x,4);
  printf("%f\n",m);
  inicializa(x,4);
  for (i=0;i<4;i++)
    printf("%f ",x[i]);
}
```



```

#include<stdio.h>
#define tam 10
//calcula la media de los elementos
float media(float a[],int n)
{
int i;
float s=0;
for (i=0;i<n;i++)
    s=s+a[i];
return s/n;
}

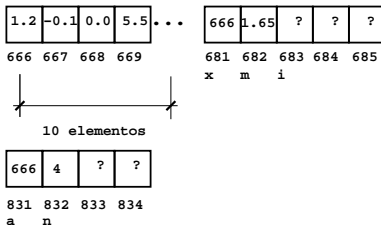
//inicializa los elementos a cero
void inicializa(float a[],int n)
{
int i;
for (i=0;i<n;i++)
    a[i]=0;
}

```

```

int main()
{
float x[tam],m;
int i;
x[0]=1.2;x[1]=-0.1;x[2]=0.0;x[3]=5.5;
m=media(x,4);
printf("%f\n",m);
inicializa(x,4);
for (i=0;i<4;i++)
    printf("%f ",x[i]);
}

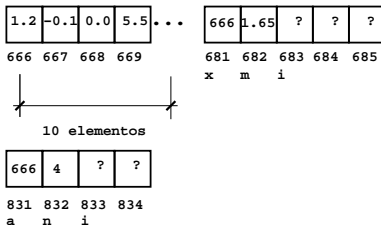
```



```
#include<stdio.h>
#define tam 10
//calcula la media de los elementos
float media(float a[],int n)
{
int i;
float s=0;
for (i=0;i<n;i++)
    s=s+a[i];
return s/n;
}
```

```
//inicializa los elementos a cero
void inicializa(float a[],int n)
{
int i;
for (i=0;i<n;i++)
    a[i]=0;
}
```

```
int main()
{
float x[tam],m;
int i;
x[0]=1.2;x[1]=-0.1;x[2]=0.0;x[3]=5.5;
m=media(x,4);
printf("%f\n",m);
inicializa(x,4);
for (i=0;i<4;i++)
    printf("%f ",x[i]);
}
```



```

#include<stdio.h>
#define tam 10
//calcula la media de los elementos
float media(float a[],int n)
{
int i;
float s=0;
for (i=0;i<n;i++)
    s=s+a[i];
return s/n;
}

```

```

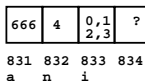
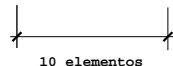
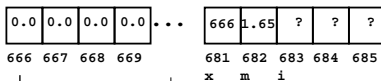
//inicializa los elementos a cero
void inicializa(float a[],int n)
{
int i;
for (i=0;i<n;i++)
    a[i]=0;
}

```

```

int main()
{
float x[tam],m;
int i;
x[0]=1.2;x[1]=-0.1;x[2]=0.0;x[3]=5.5;
m=media(x,4);
printf("%f\n",m);
inicializa(x,4);
for (i=0;i<4;i++)
    printf("%f ",x[i]);
}

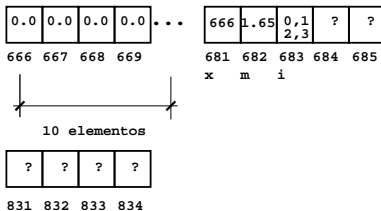
```




```
#include<stdio.h>
#define tam 10
//calcula la media de los elementos
float media(float a[],int n)
{
  int i;
  float s=0;
  for (i=0;i<n;i++)
    s=s+a[i];
  return s/n;
}
```

```
//inicializa los elementos a cero
void inicializa(float a[],int n)
{
  int i;
  for (i=0;i<n;i++)
    a[i]=0;
}
```

```
int main()
{
  float x[tam],m;
  int i;
  x[0]=1.2;x[1]=-0.1;x[2]=0.0;x[3]=5.5;
  m=media(x,4);
  printf("%f\n",m);
  inicializa(x,4);
  for (i=0;i<4;i++)
    printf("%f ",x[i]);
}
```



```
#include<stdio.h>
#define tam 10
//suma a y b devuelve el resultado en c
void suma(float a[],
         float b[],
         float c[],
         int n)
{
    int i;
    for (i=0;i<n;i++)
        c[i]=a[i]+b[i];
}
//pide un vector por el teclado
void lee_vector(float x[],int n)
{
    int i;
    for (i=0;i<n;i++)
        scanf("%f",&x[i]);
}
//muestra un vector por la pantalla
void muestra_vector(float x[],int n)
{
    int i;
    for (i=0;i<n;i++)
        printf("%f ",x[i]);
}
```

```
int main()
{
    int n;
    float x[tam],y[tam],z[tam];
    printf("\nCuantos elementos vas a usar?");
    do{
        scanf("%d",&n);
    }while(n>tam);
    lee_vector(x,n);
    lee_vector(y,n);
    suma(x,y,z,n);
    muestra_vector(z,n);
}
```

Índice

1 Introducción

2 Arrays Unidimensionales

- Concepto
- Declaración
- Uso de Arrays Unidimensionales
- Recorrido de Arrays Unidimensionales
- Paso de Arrays Unidimensionales a funciones

3 Arrays Bidimensionales

- Concepto
- Declaración
- Uso de Arrays Bidimensionales
- Recorrido de Arrays Bidimensionales
- Paso de Arrays Bidimensionales a funciones

Índice

1 Introducción

2 Arrays Unidimensionales

- Concepto
- Declaración
- Uso de Arrays Unidimensionales
- Recorrido de Arrays Unidimensionales
- Paso de Arrays Unidimensionales a funciones

3 Arrays Bidimensionales

- **Concepto**
- Declaración
- Uso de Arrays Bidimensionales
- Recorrido de Arrays Bidimensionales
- Paso de Arrays Bidimensionales a funciones

- Un array bidimensional es la representación de una *matriz*.
- Cada elemento de una matriz se identifica con dos índices, encerrados entre [], convencionalmente el primero representa la fila y el segundo la columna.
- A todos los efectos, un elemento de un array bidimensional se comporta como una variable del tipo con el que se construye el array.
- Los índices comienzan en cero.
- El índice de la última fila o columna es igual al tamaño de esas dimensiones menos uno.
- No existen operaciones predefinidas, ni de E/S, ni asignación, ni aritméticas, lógicas, relacionales, etc.
- No se puede acceder a una fila o columna como si fuese un vector. Utilizar sólo una pareja de [] es un error.

Índice

1 Introducción

2 Arrays Unidimensionales

- Concepto
- Declaración
- Uso de Arrays Unidimensionales
- Recorrido de Arrays Unidimensionales
- Paso de Arrays Unidimensionales a funciones

3 Arrays Bidimensionales

- Concepto
- **Declaración**
- Uso de Arrays Bidimensionales
- Recorrido de Arrays Bidimensionales
- Paso de Arrays Bidimensionales a funciones

```
tipo nombre[filas][columnas];
```

- tipo es uno de los tipos predefinidos en C: int, float, char..
- nombre es un identificador.
- filas y columnas son expresiones constantes, conocidas en tiempo de compilación.
 - Se definen de la misma forma que el tamaño de un vector.

```
...  
#define filas 10  
#define columnas 25  
...  
float a[filas][columnas];  
...
```

```
...  
const int filas=10;  
const int columnas=25;  
...  
float a[filas][columnas];  
...
```

```
...  
float a[10][25];  
...
```

Índice

1 Introducción

2 Arrays Unidimensionales

- Concepto
- Declaración
- Uso de Arrays Unidimensionales
- Recorrido de Arrays Unidimensionales
- Paso de Arrays Unidimensionales a funciones

3 Arrays Bidimensionales

- Concepto
- Declaración
- **Uso de Arrays Bidimensionales**
- Recorrido de Arrays Bidimensionales
- Paso de Arrays Bidimensionales a funciones

...nombre[fila][columna]...

- Los [] encierran los índices que especifican la fila y la columna del elemento.
- Los índices puede ser cualquier expresión de tipo entero que evalúe a 0..filas-1, 0..columnas-1.
 - Las expresiones pueden variar durante la ejecución del programa.
- Se puede escribir lo anterior en cualquier sitio en el que se pueda escribir una variable del tipo del array.
- Cada elemento se comporta como una variable del tipo del array.

```
#include<stdio.h>
int main()
{
int x[2][3],a=2,i=0,j=0; //declaración
x[1][2]=43;           //diversas asignaciones
x[i][j+1]=a*2+1;
a=x[1][2]+x[i][j+1]*3; //uso en expresiones
//petición por teclado de algún elemento
//los índices pueden ser expresiones
scanf("%d%d",&x[i+1][j],&x[i][j]);
x[1][2]=x[1][0]+x[0][0];
i=i+1; //modificación de índices
j=2;
//mostrar por pantalla algún elemento
printf("%d",x[i][j]);
}
```

Índice

- 1 Introducción
- 2 Arrays Unidimensionales
 - Concepto
 - Declaración
 - Uso de Arrays Unidimensionales
 - Recorrido de Arrays Unidimensionales
 - Paso de Arrays Unidimensionales a funciones
- 3 Arrays Bidimensionales**
 - Concepto
 - Declaración
 - Uso de Arrays Bidimensionales
 - Recorrido de Arrays Bidimensionales**
 - Paso de Arrays Bidimensionales a funciones

- El recorrido de un array bidimensional consiste en aplicar una misma operación a cada elemento del mismo.
- Para ello se generan todas las parejas de valores válidos para los índices que denotan filas y columnas.
- Una forma cómoda es utilizar dos bucles `for` anidados.

```
float a[filas][columnas];  
...  
//por filas  
for (i=0;i<filas;i++)  
{  
    for (j=0;j<columnas;j++)  
    {  
        //para cada i j  
        //aplicar op. a[i][j]  
        ...a[i][j]...  
    }  
}
```

```
float a[filas][columnas];  
...  
//por columnas  
for (j=0;j<columnas;j++)  
{  
    for (i=0;i<filas;i++)  
    {  
        //para cada i j  
        //aplicar op. a[i][j]  
        ...a[i][j]...  
    }  
}
```

```

#include<stdio.h>
const int fil=10;
const int col=10;
int main()
{
//a, b, c son matrices
float a[fil][col],b[fil][col],cc[fil][col];
//i j para los for, f y c para definir
//las fil y col que se van a usar
int i,j,f,c;

//se pide f asegurando que no sea
//mayor que fil
printf("\nCuántas filas vas a usar?");
do{
scanf("%d",&f);
}while(f>fil);

//se pide c asegurando que no sea
//mayor que col
printf("\nCuántas columnas vas a usar?");
do{
scanf("%d",&c);
}while(c>col);

//pedir el primer array
for (i=0;i<f;i++)
for (j=0;j<c;j++)
{
printf("Elemento [%d][%d]=",i,j);
scanf("%f",&a[i][j]);
}
//pedir el segundo array
for (i=0;i<f;i++)
for (j=0;j<c;j++)
{
printf("Elemento [%d][%d]=",i,j);
scanf("%f",&b[i][j]);
}
//calcular la suma de a y b
//cada elemento de c es la
//suma de los elementos correspondientes
for (i=0;i<f;i++)
for (j=0;j<c;j++)
cc[i][j]=a[i][j]+b[i][j];
//mostrar el resultado
for (i=0;i<f;i++)
{
for (j=0;j<c;j++)
printf("%f ",cc[i][j]);
printf("\n");
}
}

```

Índice

1 Introducción

2 Arrays Unidimensionales

- Concepto
- Declaración
- Uso de Arrays Unidimensionales
- Recorrido de Arrays Unidimensionales
- Paso de Arrays Unidimensionales a funciones

3 Arrays Bidimensionales

- Concepto
- Declaración
- Uso de Arrays Bidimensionales
- Recorrido de Arrays Bidimensionales
- Paso de Arrays Bidimensionales a funciones

- El nombre de una matriz, sin las dos parejas de `[]`, es un puntero al tipo del vector.
- Su valor es la dirección de memoria en donde se encuentra el primer elemento.
- El contenido de esa dirección de memoria es el primer elemento.
- Los elementos siguientes ocupan posiciones contiguas, por filas, primero la primera fila, después la segunda....
- Si se declara `tipo nombre[fil][col];` entonces `*(nombre+col*i+j)` es lo mismo que `nombre[i][j]`.
- Cuando se pasa una matriz como parámetro a una función, se hace implícitamente por referencia, por lo tanto se puede modificar.
 - El numero de columnas es necesario, el de filas irrelevante.

```

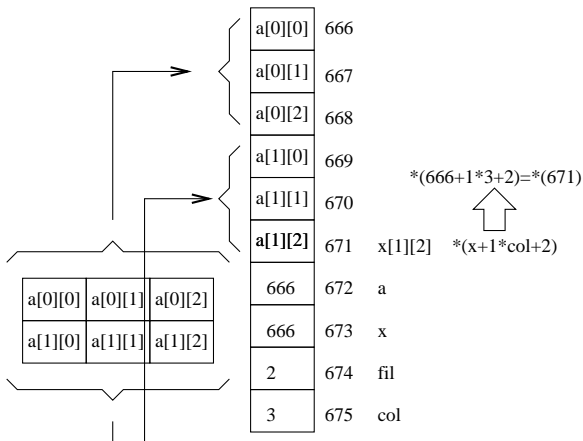
#include<stdio.h>

const int fil=2;
const int col=3;

tipo f(char x[][col])
{
...x[1][2]...
}

int main()
{
char a[fil][col];
...f(a)...
}

```




```

#include<stdio.h>
#define fil 10
#define col 25

//suma a y b devuelve el resultado en c
void suma(float a[][col],
          float b[][col],
          float cc[][col],
          int f,int c)
{
    int i,j;
    for (i=0;i<f;i++)
        for (j=0;j<c;j++)
            cc[i][j]=a[i][j]+b[i][j];
}

//pide una matriz por el teclado
void lee(float x[][col],int f,int c)
{
    int i,j;
    for (i=0;i<f;i++)
        for (j=0;j<c;j++)
            {
                printf("\n[%d][%d]=",i,j);
                scanf("%f",&x[i][j]);
            }
}

//muestra una matriz por la pantalla
void muestra(float x[][col],int f,int c)
{
    int i,j;
    for (i=0;i<f;i++)
        {
            for (j=0;j<c;j++)
                printf("%f ",x[i][j]);
            printf("\n");
        }
}

int main()
{
    int f,c;
    float x[fil][col],y[fil][col],z[fil][col];
    printf("\nIntroduce filas y columnas:");
    do{
        scanf("%d",&f);
    }while(f>fil);

    do{
        scanf("%d",&c);
    }while(c>col);

    lee(x,f,c);
    lee(y,f,c);
    suma(x,y,z,f,c);
    muestra(z,f,c);
}

```