



PROGRAMACIÓN

ARREGLOS Y CADENAS DE CARACTERES

[Array]

Departamento de Informática - Facultad Politécnica

¿Qué veremos hoy?

- Necesidad de contar con estructuras de datos
- Arreglos unidimensionales (vectores)
 - Manejo de cadenas
- Arreglos multidimensionales
- Cadenas de caracteres

Necesidad de estructuras de datos

Las variables simples únicamente pueden almacenar un dato. En ejercicios anteriores, hemos trabajado con listas de números (calificaciones de alumnos de una misma clase, trabajadores de una empresa, etc.); y su manejo con variables simples puede ser un poco impráctico, además de no considerar el almacenamiento de las entradas.

Pregunta 1: ¿Cómo podríamos ordenar N enteros utilizando variables simples?

```
int a = 5;  
double b = 4;  
char c = 'a';  
float d = 3.2;
```

{ 5 , 1 , 8 , 7 , 3 , 4 , ... }

Pregunta 2: ¿Cómo calcularíamos la cantidad de alumnos que obtuvieron una nota inferior al promedio del curso en cierta materia?

Estructuras de datos

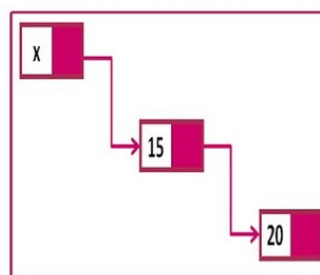
Para salvar estas situaciones, la mayoría de los lenguajes de programación incluyen características de **estructuras de datos**. Una estructura de datos es una colección de datos que pueden ser caracterizados por su organización y las operaciones que se definen en ella (*acceso, inserción, borrado*).

Las estructuras de datos se dividen en:

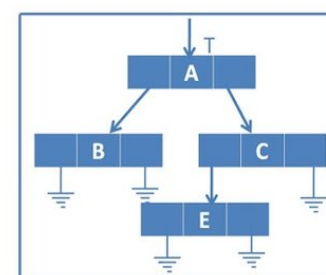
- **Estáticas** (tamaño definido de antemano): arreglos, registros, cadenas
- **Dinámicas** (no tiene limitaciones de tamaño): listas(pilas/colas), listas enlazadas, árboles, grafos.



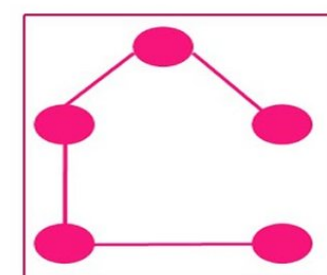
Sorting



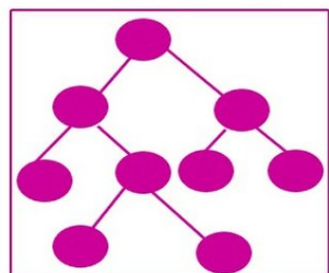
Link list



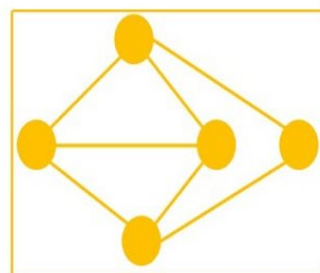
list



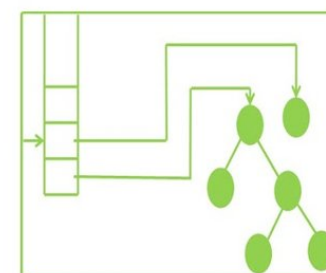
spanning tree



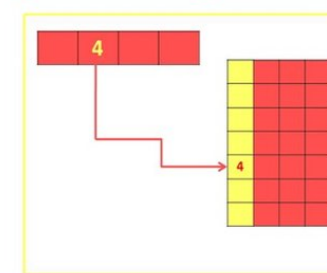
Tree



Graph

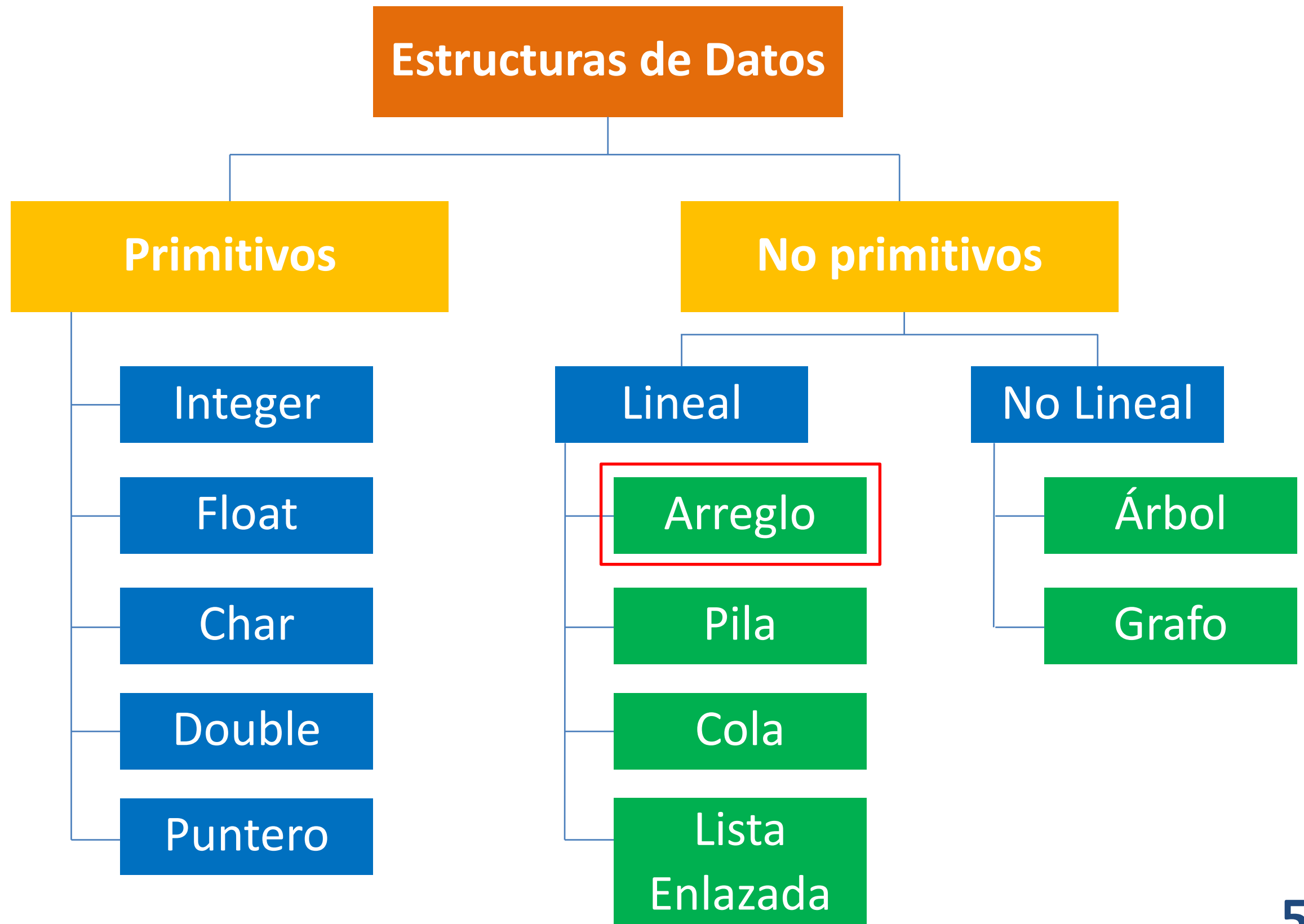


Stack



Hashing

Estructuras de datos



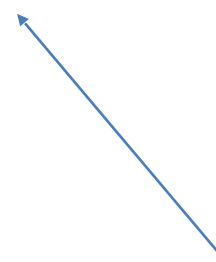
Arreglos

Las **estructuras de datos básicas** que soportan la mayoría de los lenguajes son los **arreglos** (arrays) (siendo el **vector** un arreglo de una dimensión, y la **matriz** uno de dos dimensiones).

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

Ejemplo de vector: nota (notas de una clase de n alumnos)

nota 0	nota 1	nota 2	...	nota	...	nota -1
--------	--------	--------	-----	------	-----	---------

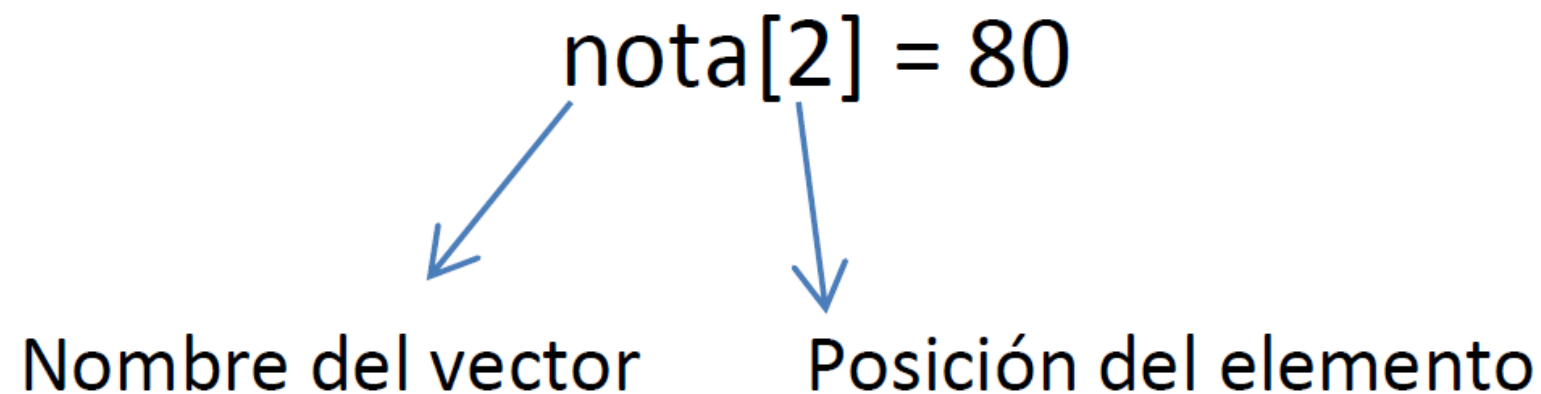


nota[2]

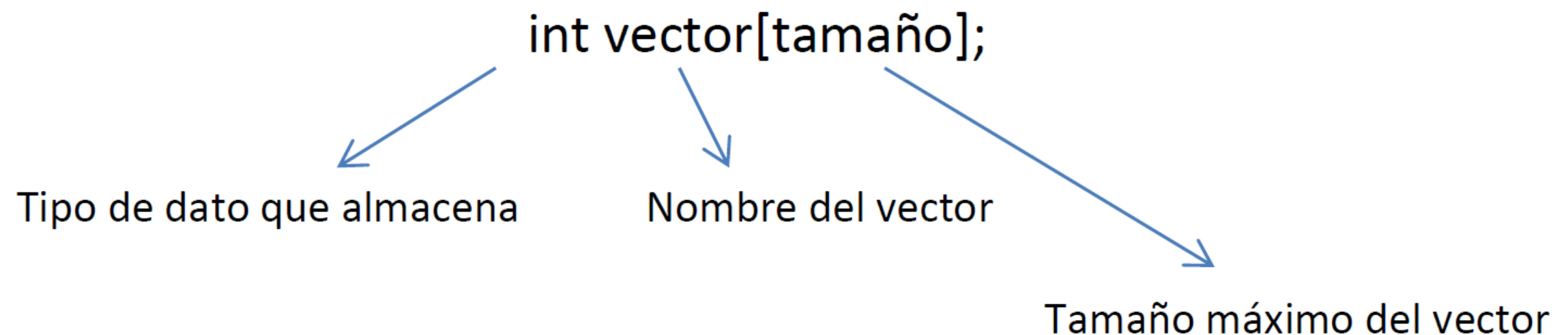
Observación importante: en C, el primer elemento se representa por el índice 0. Si el tamaño del vector es n , entonces el último elemento tiene un índice $n - 1$.

Arreglos unidimensionales o vectores

El vector `nota` tiene subíndices o índices de sus elementos $(0, 1, 2, \dots, i, \dots, n-1)$ que indican la posición de un elemento particular dentro del arreglo (de tamaño n). Por ejemplo, si se desea modificar el tercer elemento de un vector de tipo numérico:



La declaración de un vector en C se hace de la siguiente manera:



Vectores – Declaración

Declaración de vectores

```
tipo nombre_vector[tam];
```

Es muy similar a la declaración de las variables simples, pero el tamaño del vector va en corchetes luego del nombre del mismo. El tamaño tam debe tener valores asignados previamente (en el caso de que sean variables), o bien ser constantes.

Si los elementos del vector están definidos, puede hacerse la siguiente declaración:

```
int a[5]={1,2,3,4,5};
```


Vectores – Lectura e impresión

```
#include<stdio.h>

int main(){
    int n;
    printf("Ingrese el tamanho del vector:
");
    scanf("%d",&n);
    int a[n];
    int i;
    for(i=0;i<n;i++){
        printf("Ingrese el elemento %d: ",i);
        scanf("%d",&a[i]);
    }
    printf("\nEl vector es:\n");
    for(i=0;i<n;i++) printf("%d\t",a[i]);
    return 0;
}
```

Ejemplo 1

Escribir un programa que permita leer un vector de n números enteros introducidos por teclado y que calcule e imprima por pantalla la suma de todos sus elementos, el promedio, el mayor y el menor.

Ejemplo 1

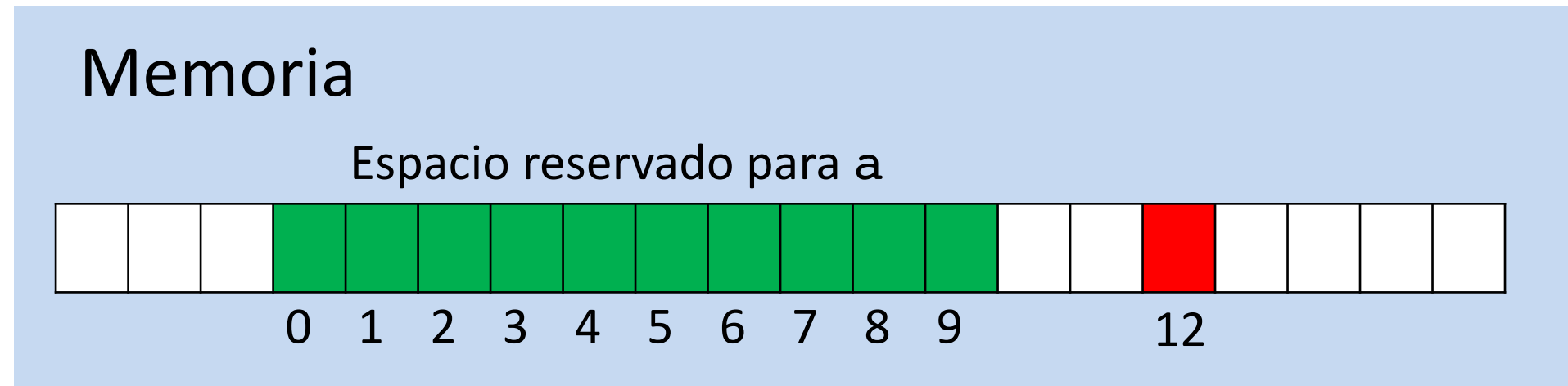
```
#include<stdio.h>
int main(){
    int i,n,max,min,sum=0;
    float p;
    printf("Ingrese la cantidad de elementos del vector: ");
    scanf("%d",&n);
    int vec[n]; //Se dimensiona el vector
    for(i=0;i<n;i++) scanf("%d",&vec[i]);
    max=vec[0]; min=vec[0]; sum+=vec[0];
    for(i=1;i<n;i++){
        if(min>vec[i]) min=vec[i];
        if(max<vec[i]) max=vec[i];
        sum+=vec[i];
    }
    printf("\nEl menor es: %d",min);
    printf("\nEl mayor es: %d",max);
    printf("\nEl promedio es: %.2f",(1.0*sum/n));
    return 0;
}
```

Algunas observaciones

- Si escribimos:

```
int a[10];
```

```
a[12]=8;
```



Esto no *necesariamente* marcará un error; pero como se almacena el 8 en un lugar no reservado, puede sobre-escribirse otra variable (o hasta dar un error en el sistema operativo). Cuando veamos punteros analizaremos esto con mayor detalle.

- `int b[];` //Da error de compilación
- `int c[]={1,2,3}` //Es correcto, y se reservan 3 espacios en la memoria
- Tener cuidado al dimensionar un vector con una variable (asegurarse de que esa variable tenga un valor asignado *anteriormente*).


Vectores y funciones

Se puede pasar un vector como argumento de una función. Un ejemplo es el siguiente:

No hace falta colocar el tamaño aquí

```
void imprimirVector(int x[], int n){  
    int i;  
    printf("El vector es:\n");  
    for(i=0;i<n;i++) printf("%d\t",x[i]);  
    printf("\n");  
}
```

```
int main(){  
    int a[]={1,2,3,4,5};  
    imprimirVector(a,5);  
    return 0;  
}
```

 C:\Users\oche_\Downloads\Docencia\Computacion\C

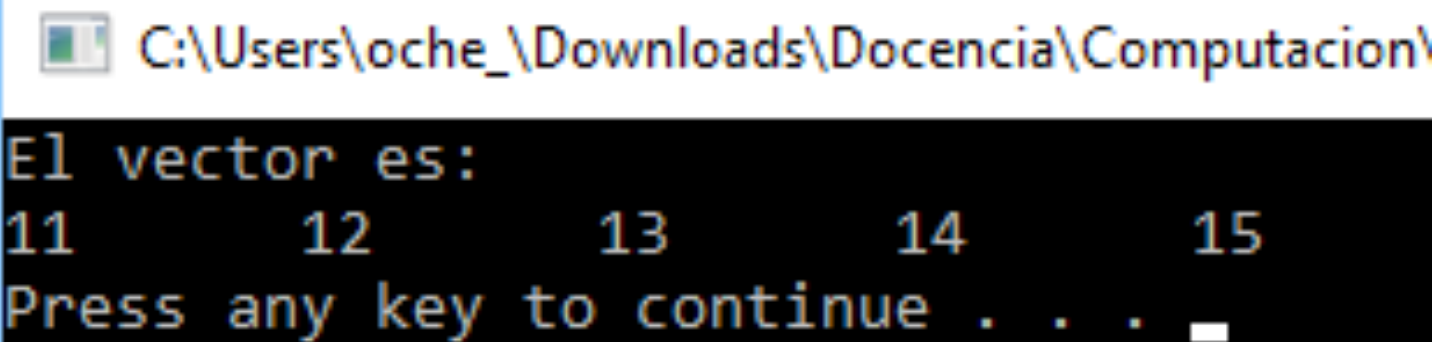
```
El vector es:  
1      2      3      4      5  
Press any key to continue . . .
```

Vectores y funciones

Cuando un vector es el argumento de una función, se pasa la **referencia*** del vector al llamar a dicha función. Por ello, si se hacen cambios dentro de la función, **éstos se reflejarán en el vector original**.

```
void sumar10(int x[], int n){  
    int i;  
    for(i=0;i<n;i++) x[i]+=10;  
}
```

```
int main(){  
    int a[]={1,2,3,4,5};  
    sumar10(a,5);  
    imprimirVector(a,5);  
    return 0;  
}
```



```
C:\Users\oche_\Downloads\Docencia\Computacion\  
El vector es:  
11      12      13      14      15  
Press any key to continue . . .
```

*En la clase de punteros veremos por qué ocurre esto.

Cadenas en C

En C, no existe el tipo de dato “cadena” (ó string). Pero existe una convención en la forma de representar una cadena, y es a través de un **arreglo de caracteres**. El fin de una cadena se indica con el carácter especial `'\0'`.

<code>'H'</code>	<code>'e'</code>	<code>'l'</code>	<code>'l'</code>	<code>'o'</code>	<code>'\n'</code>	<code>'\0'</code>
------------------	------------------	------------------	------------------	------------------	-------------------	-------------------

Esta cadena se declara de las siguientes maneras:

```
char a[7]={'h','e','l','l','o','\n','\0'};  
char b[7]="hello\n";
```

Cadenas en C - Lectura

Formas de leer una cadena por teclado:

```
char cad[20];
```

- `gets(cad);` → se encuentra en `stdio.h`
- `scanf("%s", cad);` → fijarse que no se usa `&` (por tratarse de un vector), y se coloca `%s`

La diferencia entre ambas es que `gets()` lee los espacios en blanco, mientras que `scanf()` no.

Formas de imprimir una cadena en pantalla:

- `printf("%s", cad);` → imprime la cadena y el cursor queda al lado.
- `puts(cad);` → imprime la cadena y da un salto de línea.

Cadenas en C – Funciones

Algunas funciones útiles de cadenas

```
#include<string.h>
```

- `strcpy(cad1, cad2);` → copia la cadena cad2 a la cad1.
- `strcat(cad1, cad2);` → concatena las cadenas cad1 y cad2 (el resultado queda en cad1).
- `int a=strlen(cad1);` → devuelve la cantidad de caracteres de la cadena cad1 (sin contar el `\0`).
- `int a=strcmp(cad1, cad2);` → devuelve el valor que representa la diferencia entre los primeros caracteres distintos.

Ejercicios: escribir funciones que implementen cada una de las presentadas (llamarlas de otra forma).

Obs: con cadenas no puede hacerse lo siguiente:

```
char cad[10];
```

```
cad = "hola"; //error!! No puede asignarse directamente
```

Cadenas en C – Funciones

```
void copia(char cad1[], char cad2[]){
    int i=0;
    while(cad2[i]!='\0'){
        cad1[i]=cad2[i];
        i++;
    }
    cad1[i]='\0';
}
```

```
void concatenar(char cad1[], char cad2[]){
    int i=0;
    while(cad1[i]!='\0') i++;
    int j=0;
    while(cad2[j]!='\0'){
        cad1[i]=cad2[j];
        i++;j++;
    }
    cad1[i]='\0';
}
```

Cadenas en C – Funciones

```
int longitud(char cad1[]){  
    int i=0;  
    while(cad1[i]!='\0') i++;  
    return i;  
}
```

```
int comparar(char cad1[], char cad2[]){  
    int i=0,a;  
    while((cad1[i]!='\0') && (cad1[i]==cad2[i])) i++;  
    a=cad1[i]-cad2[i];  
    return a;  
}
```

Cadenas en C – Conversión

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

- Cadena a entero (**atoi**):

```
char cad[] = "12345";  
int a = atoi(cad); // a <-- 12345
```

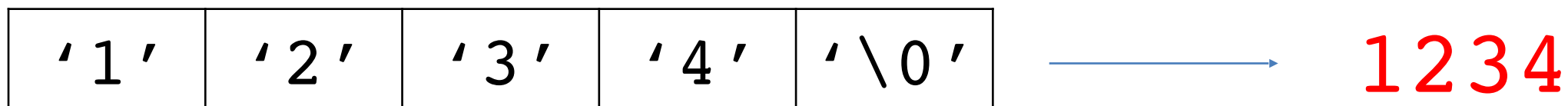
- Entero a cadena (**itoa**):

```
int a = 12345;  
char cad[10];  
itoa(a, cad, 10); //cad <-- "12345"
```

Obs: lleva el 10 porque es en base decimal

Cadenas en C – Conversión

```
int cadAnum(char s[])
{
    int i=0, n=0, signo=1;
    while(s[i]==' ') i++; //se eliminan los espacios en blanco
    if ((s[i]=='+' || (s[i]=='-')) { //se toma el signo
        if(s[i] == '-') signo=-1;
        i++;
    }
    while(s[i]!='\0'){
        n=10*n +(s[i]-'0');
        i++;
    }
    return (signo*n);
}
```



Cadenas en C – Conversión

```
void numAcad(char s[], int n){
    int i=0,d=1,j;
    if(n<0){
        s[i]='-';
        i++;
        n=n*(-1);
    }
    int aux=n;
    while(aux>=10){ //se cuenta el numero de cifras
        d++;
        aux/=10;
    }
    for(j=(d+i-1);j>=i;j--){
        s[j]=(n%10)+'0';
        n/=10;
    }
    s[i+d]='\0';
}
```

1234



'1'	'2'	'3'	'4'	'\0'
-----	-----	-----	-----	------

Cadenas en C – Conversión

Tarea: convertir una cadena a un punto flotante (número decimal) y viceversa.

Entrada

'1'	'2'	'.'	'3'	'4'	'\0'
-----	-----	-----	-----	-----	------

Salida

12.34

Entrada

12.34

Salida

'1'	'2'	'.'	'3'	'4'	'\0'
-----	-----	-----	-----	-----	------

Arreglos multidimensionales

Hasta el momento, hemos considerado sólo los arreglos unidimensionales (vectores), y en ellos cada elemento se define o referencia por un índice o subíndice. Estos vectores son elementos de datos escritos en una secuencia.

Sin embargo, existen grupos de datos que son mejor representados en forma de tabla o matriz con dos o más subíndices. Por ejemplo: tablas kilométricas entre ciudades, informes de ventas periódicas, etc.

Cuadro de Distancia en Km / Distance chart in Km											
CIUDADES / CITIES	ANTOFAGASTA	ARICA	CALAMA	IQUIQUE	LA SERENA	PUERTO MONTT	PUNTA ARENAS	SANTIAGO	TEMUCO	VALDIVIA	VIÑA DEL MAR
ANTOFAGASTA	***	701	213	492	887	2377	4451	1361	2038	2202	1311
ARICA	701	***	614	316	1588	3078	5152	2062	2739	2903	2012
CALAMA	213	614	***	410	1104	2590	5152	1574	2251	2415	1532
CONCEPCION	1880	2581	2093	2372	993	626	2700	519	287	452	639
IQUIQUE	492	316	410	***	1377	2869	4943	1853	2530	2694	1811
LA SERENA	887	1588	1104	1377	***	1490	3564	474	1151	1315	432
OSORNO	2274	2975	2487	2766	1387	109	2177	913	236	107	1033
PUERTO NATALES	4464	5165	4677	4956	3577	2299	254	3103	2424	2295	3223
PUERTO VARAS	2357	3058	2570	2849	1470	20	2266	996	319	190	1116
PUERTO MONTT	2377	3078	2590	2869	1490	***	2286	1016	339	210	1136
PUNTA ARENAS	4451	5152	4664	4943	3564	2286	***	3090	2413	2284	3210
SANTIAGO	1361	2062	1574	1853	474	1016	3090	***	677	841	120
TEMUCO	2038	2739	2251	2530	1151	339	2413	677	***	162	797
VALDIVIA	2202	2903	2415	2694	1313	210	2177	839	162	***	959

Arreglos multidimensionales

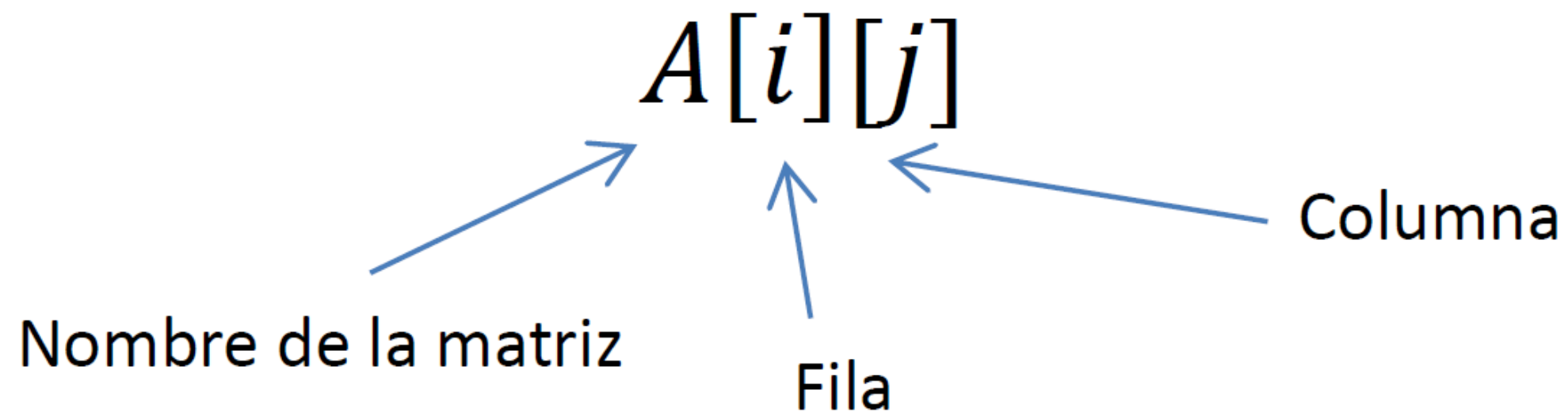
Se pueden definir **tablas** o **matrices** como **arreglos multidimensionales**, cuyos elementos se pueden referenciar por dos, tres o más subíndices. El término **matriz** se asocia generalmente a un **arreglo bidimensional**. Como en el caso del vector o arreglo unidimensional, sus elementos son del mismo tipo (`int`, `char`, `float`, etc).

		Columnas				
A		0	1	2	3	4
Filas	0					
	1					
	2					
	3					
	4					

$A[2][3]$

Matrices

La matriz puede verse como un **vector de vectores**. Por ello, se necesita especificar dos subíndices para poder identificar cada uno de sus elementos.



Observación: *un vector de cadenas puede verse como una matriz de caracteres, donde se tiene una palabra en cada fila.*

Palabras

'h'	'o'	'l'	'a'	'\0'
'q'	'u'	'e'	'\0'	
't'	'a'	'l'	'\0'	

Matrices

La declaración de una matriz o arreglo bidimensional es:

```
tipo nombre_matriz[cant_filas][cant_columnas];
```

Se deben tener los mismos cuidados que los mencionados para el caso de los vectores en relación al dimensionamiento de la matriz (cantidad de filas y columnas).

Se puede declarar una matriz con sus elementos:

```
int matriz[3][6] = {{16, 21, 8, 3, -7, 9}, {-3, 11, 0, 5, 9, 7}, {13, 7, -64, 19, 14, 2}}
```

	0	1	2	3	4	5
0	16	21	8	3	-7	9
1	-3	11	0	5	9	7
2	13	7	-64	19	14	2

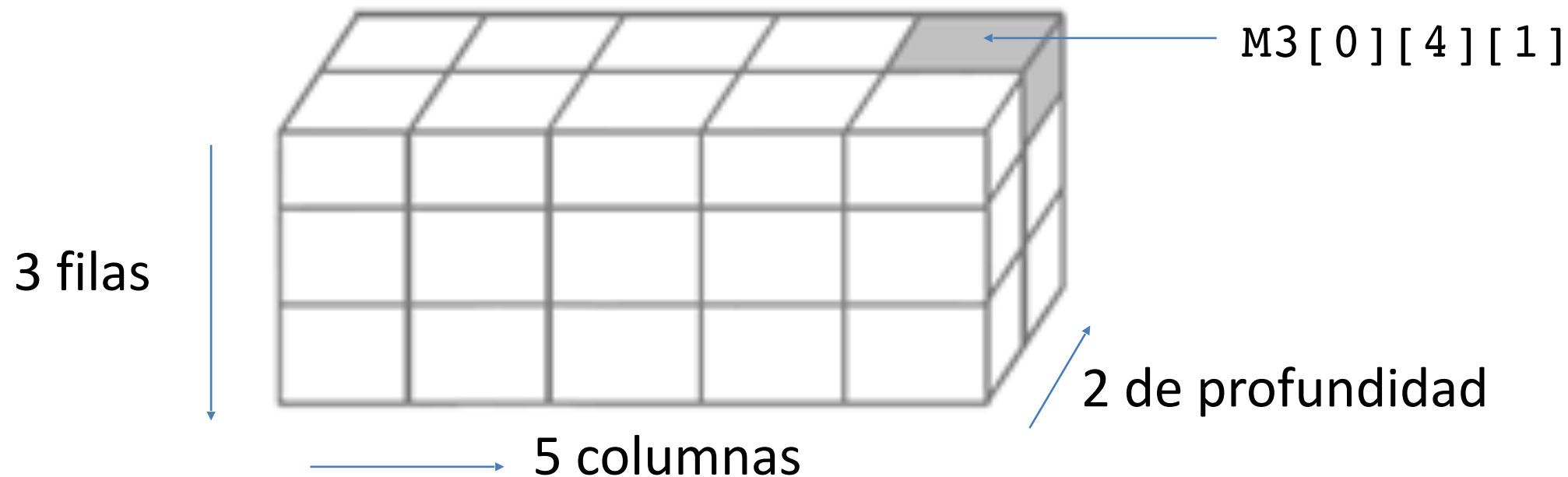
Arreglos multidimensionales

Las dimensiones de los arreglos van aumentando a medida que se incluyen más corchetes en la declaración de los arreglos.

Por ejemplo:

```
int M3[3][5][2] //define un arreglo tridimensional
```

Matriz de tres dimensiones



Matrices – Lectura e impresión

Lectura de datos de una matriz

```
for(i=0;i<cant_filas;i++){  
    for(j=0;j<cant_columnas;j++){  
        scanf("%d",&A[i][j]);  
    }  
}
```

Impresión en pantalla de los datos de una matriz

```
for(i=0;i<cant_filas;i++){  
    for(j=0;j<cant_columnas;j++){  
        printf("%d\t",A[i][j]);  
    }  
    printf("\n");  
}
```

Matrices – Ejemplo 1

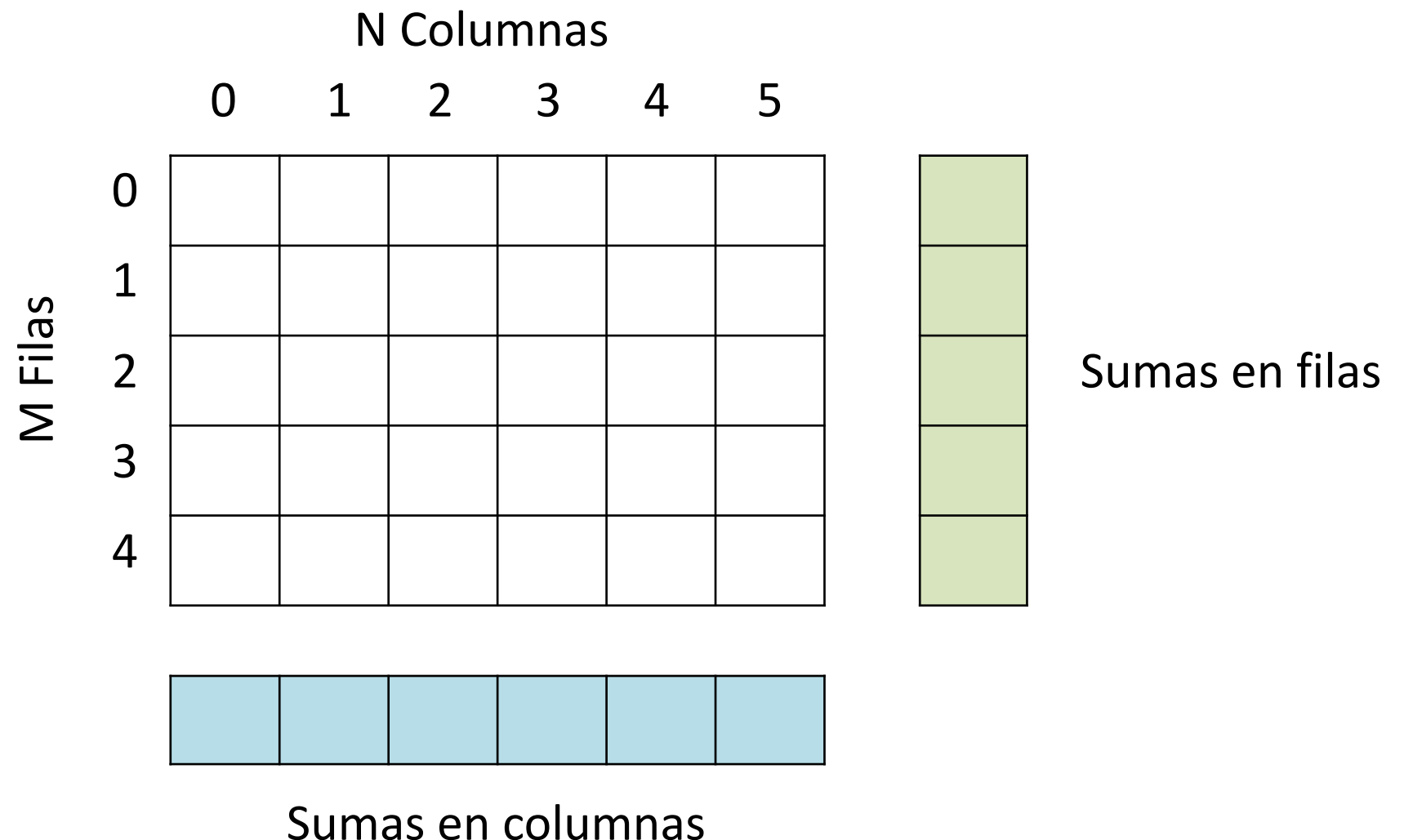
Escribir un programa que permita leer una matriz de $F \times C$ elementos, de números enteros introducidos por teclado. Calcular e imprimir el siguiente mensaje dependiendo de la suma de los elementos:

- “Filas mágicas”, si todas las sumas de las filas dan el mismo resultado y,
- “Columnas mágicas” si todas las sumas de las columnas dan el mismo resultado.

Matrices – Ejemplo 1 – Idea

Calcular e imprimir el siguiente mensaje dependiendo de la suma de los elementos:

- “Filas mágicas”, si todas las sumas de las filas dan el mismo resultado y,
- “Columnas mágicas” si todas las sumas de las columnas dan el mismo resultado.



Matrices – Ejemplo 1

```
#include<stdio.h>
int main(){
    int i,j,M,N,suma_ref_fil=0, suma_ref_col=0;
    printf("Ingrese el numero de filas: ");scanf("%d",&M);
    printf("Ingrese el numero de columnas: ");scanf("%d",&N);
    int A[M][N];
    for(i=0;i<M;i++){
        for(j=0;j<N;j++){
            printf("Ingrese A[%d][%d]: ",i,j);
            scanf("%d",&A[i][j]);}}
    for(i=0;i<N;i++) suma_ref_fil+=A[0][i];
    for(i=0;i<M;i++) suma_ref_col+=A[i][0];
    //Se revisan las filas
    int fil=1,sum; //bandera que indica si es o no magica en cuanto a filas
    for(i=1;i<M;i++){
        sum=0;
        for(j=0;j<N;j++) sum+=A[i][j];
        if(sum!=suma_ref_fil){fil=0;break;}}
    int col=1; //bandera que indica si es o no magica en cuanto a columnas
    for(i=1;i<N;i++){
        sum=0;
        for(j=0;j<M;j++) sum+=A[j][i];
        if(sum!=suma_ref_col){col=0;break;}}
    if(fil) printf("\nFilas magicas!!");
    else printf("\nLas filas no son magicas!!");
    if(col) printf("\nColumnas magicas!!");
    else printf("\nLas columnas no son magicas!!");
}
```


Ejercicios! (Vectores)

- 1- Calcular la cantidad de alumnos que obtuvieron nota inferior al promedio del curso en cierta materia. Hay N alumnos (donde N es un valor entero positivo ingresado por teclado), y todos rindieron. Las notas son números enteros que van del 0 al 100 (se asume que todas las notas ingresadas son correctas).
- 2- Diseñar un programa que calcule el producto escalar de dos vectores A y B de N elementos (números reales). Este programa debe contar con una función `escalar` que reciba dos vectores (y el tamaño de ambos) y devuelva el producto escalar.

Ejercicios! (Vectores)

3- Escribir una función que reciba un vector de enteros y su tamaño, y que pase cada elemento a la posición derecha. El elemento final del vector pasa al inicio.

1	2	3	4	5	6	7	8	9
---	---	---	---	---	---	---	---	---



9	1	2	3	4	5	6	7	8
---	---	---	---	---	---	---	---	---

Ejercicios! (Vectores)

4- Se tiene un vector de números binarios de tamaño n (siendo el mismo un múltiplo de 3). Un ejemplo es el siguiente:

1	0	1	0	0	0	1	0	0
---	---	---	---	---	---	---	---	---

Diseñar un algoritmo que cree nuevo vector a partir del vector de entrada, donde después de cada 3 elementos del vector original, se agregue un elemento que indique la cantidad de 1's en esos tres elementos. En nuestro caso, la salida sería:

1	0	1	2	0	0	0	0	1	0	0	1
---	---	---	---	---	---	---	---	---	---	---	---

Ejercicios! (Matrices)

1. Dada una matriz **A** de **MxN** números enteros, obtener la matriz transpuesta **B**, e imprimir ambas matrices.
2. Dada una matriz **A** de **NxN** números enteros, indicar si la misma es simétrica ($A[i][j]$ debe ser igual a $A[j][i]$ para $0 \leq i, j < N$).
3. Dada una matriz cuadrada **C** de **X** filas (con números reales), obtener las sumas de los elementos de la diagonal principal y los de la diagonal secundaria.

Ejercicios! (Matrices) - “Desafío”

Escribir un programa que lea un conjunto de n cadenas (cada cadena con 100 caracteres como máximo), y las imprima en orden alfabético.

Entrada

haremos
aca
una
prueba

Salida

ordenar.exe
aca
haremos
prueba
una

Gracias por la atención

