PUNTEROS

PROGRAMACIÓN (HOMOLOGADA)

FACULTAD POLITECNICA

Punteros

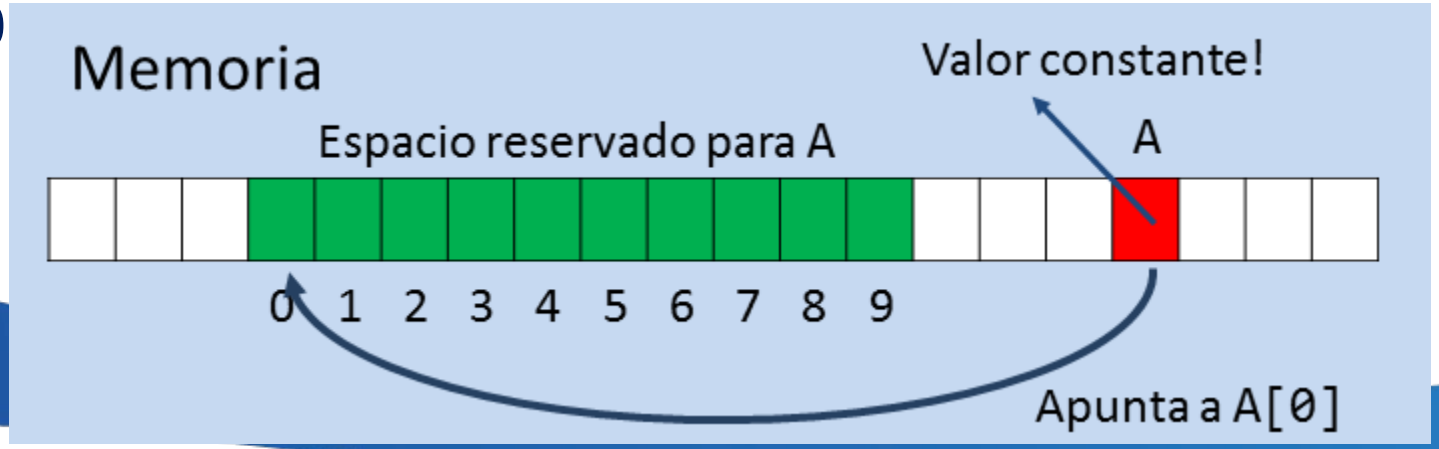
- Punteros y Array

LA RELACIÓN ENTRE PUNTEROS Y ARREGLOS

Los arreglos y los punteros están estrechamente relacionados

- El nombre del arreglo es como un puntero constante
- Los punteros pueden hacer operaciones de suscripción de arreglos

La declaración `int A[10];` define un bloque de 10 elementos consecutivos llamados `A[0`



LA RELACIÓN ENTRE PUNTEROS Y ARREGLOS/2

- *Declare* ***un arreglo*** `A[5]` ***y un puntero*** `p`

  `p = A;`

  El nombre del arreglo es en realidad una dirección del primer elemento

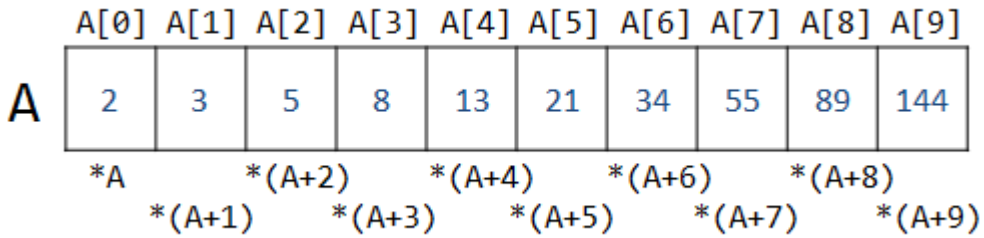
  O

  `p = &A[0]`

  Asigna explícitamente a **p** la dirección del primer elemento

LA RELACIÓN ENTRE PUNTEROS Y ARREGLOS/3

- ***Elemento A[n]***



  - puede ser acezado por **`*( p + n )`**
  - **`n`** - desplazamientos (puntero/notación desplazamiento)
  - Arreglo mismo puede usar aritmética de punteros.
    `A[3]` igual como **`*(A + 3)`**
  - Punteros pueden ser suscrito (puntero/notación índice)
    `p[3]` igual como **`p[3]`**

LA RELACIÓN ENTRE PUNTEROS Y ARREGLOS/4

demás, las siguientes notaciones son equivalen

A[i] *(A+i)

&A[i] a+i

Si ***pa*** es un puntero: ***pa***[i] *(***pa***+i)

Punteros y vectores

Como los nombres de los arreglos son **punteros constantes** que apuntan al primer elemento del arreglo, su valor no se puede modificar. Por lo tanto, hay ciertas operaciones que no están permitidas.

```
int a=5, b[]={1,2,3,4};
int *p;
p = &a;
```

Operaciones permitidas

```
p = b;
p = &b[3];
a = *(b+1);
p = b+2;
```

Operaciones inválidas

```
b++;
b = b+3;
b = &a;
b = p
```

PUNTEROS Y FUNCIONES

```
include<stdio.h>
void copia(char *, char *);
int main(){
    char cad1[30];
    copia(cad1,"hola chau");
    printf("%s\n",cad1);
    return 0;
}
void copia(char *c1, char *c2){
    int i=0;
    while(*(c2+i)!='\0'){
        *(c1+i)=*(c2+i);
        i++;
    }
    *(c1+i)='\0';
}
```

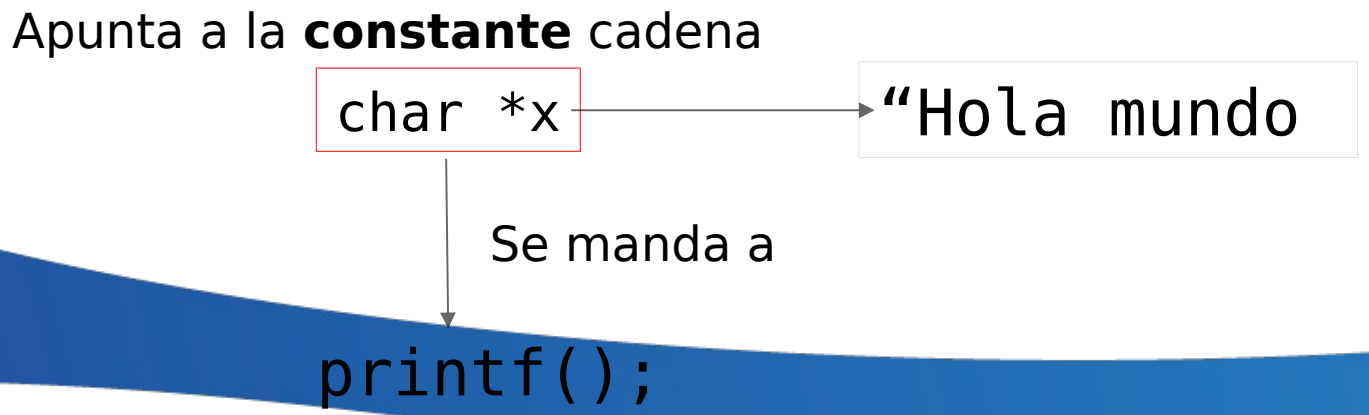
$x \qquad iq$

Punteros a caracter y funciones

Una **constante cadena**, escrita como: “Hola”, es un arreglo de caracteres. Estas constantes pueden usarse como parámetros de funciones. Una forma común de usarlas es a través de la función `printf`:

```
printf("Hola mundo\n");
```

Cuando una constante cadena aparece en un programa, lo hace a través de un puntero a carácter; y `printf` recibe un puntero que apunta al inicio del arreglo de caracteres.



PUNTEROS A CARACTER Y FUNCIONES

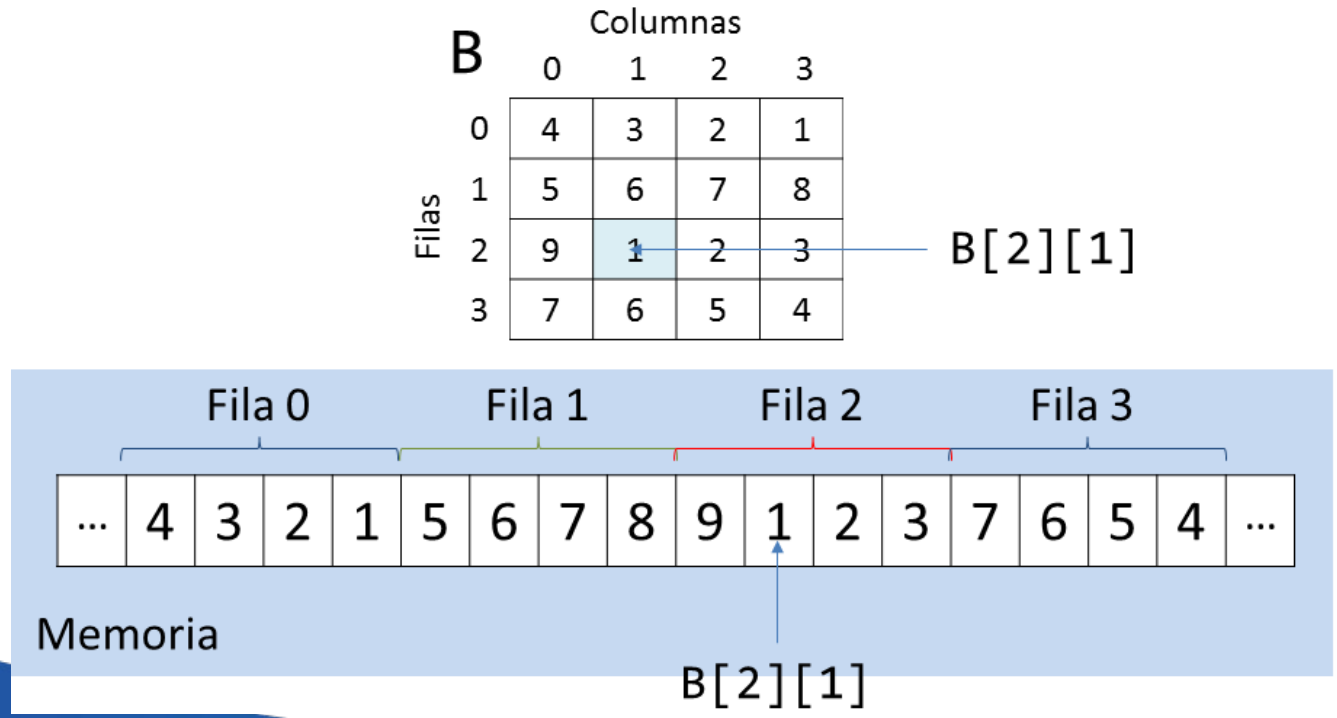
Existe una diferencia importante entre estas definiciones:

•char amesage[] = "now is the time"; /*un vector de

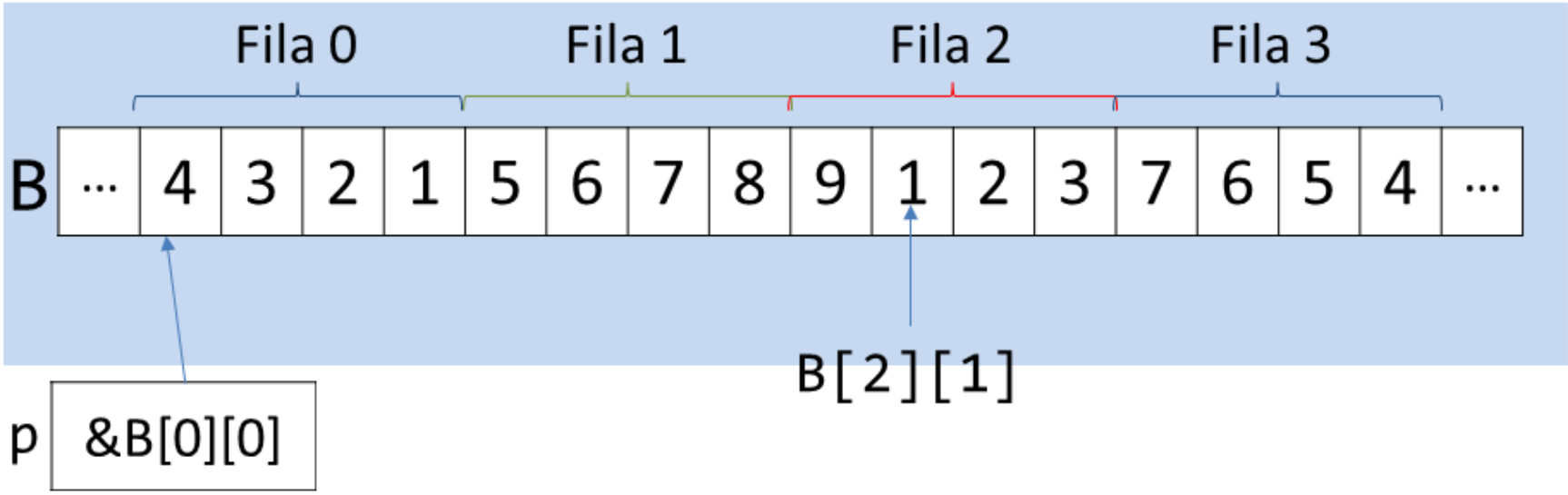
caracteres */
•char *pmessage = "now is the time"; /* un puntero */

PUNTEROS Y MATRIZ

Las matrices pueden considerarse como un vector, donde cada elemento es otro vector. Por lo tanto, una representación esquemática de su almacenamiento en memoria es la siguiente:



PUNTEROS Y MATRIZ/2

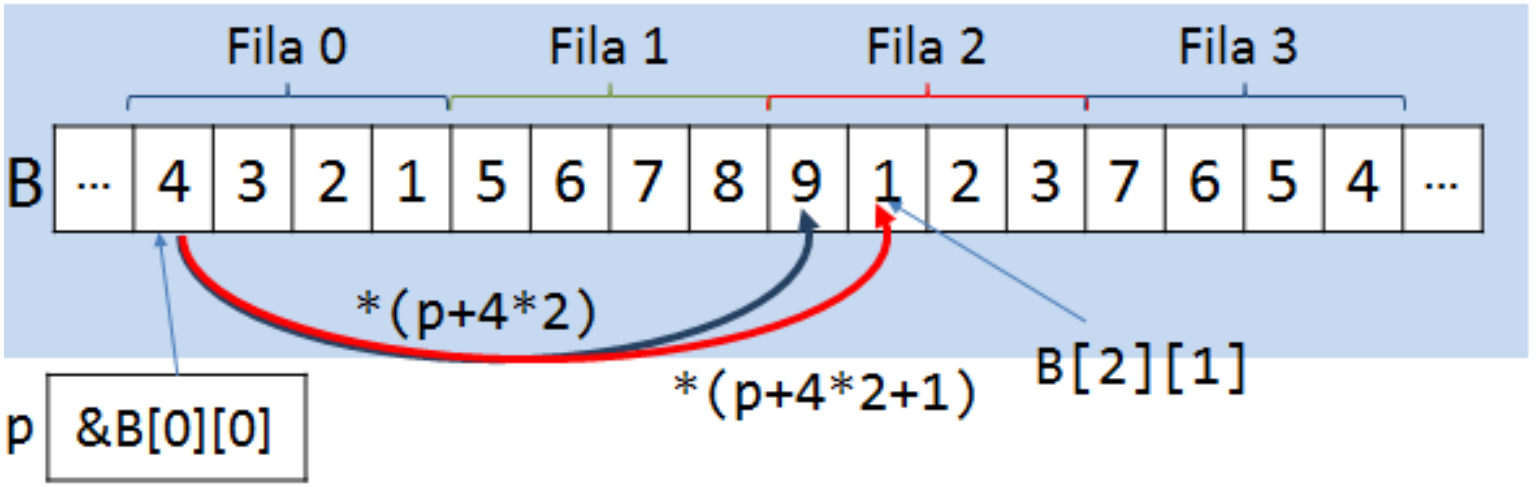


Entonces, si hacemos lo siguiente:

```
int *p;
p=&B[0][0];
```

¿Cómo podríamos acceder al elemento B[2][1] mediante p?

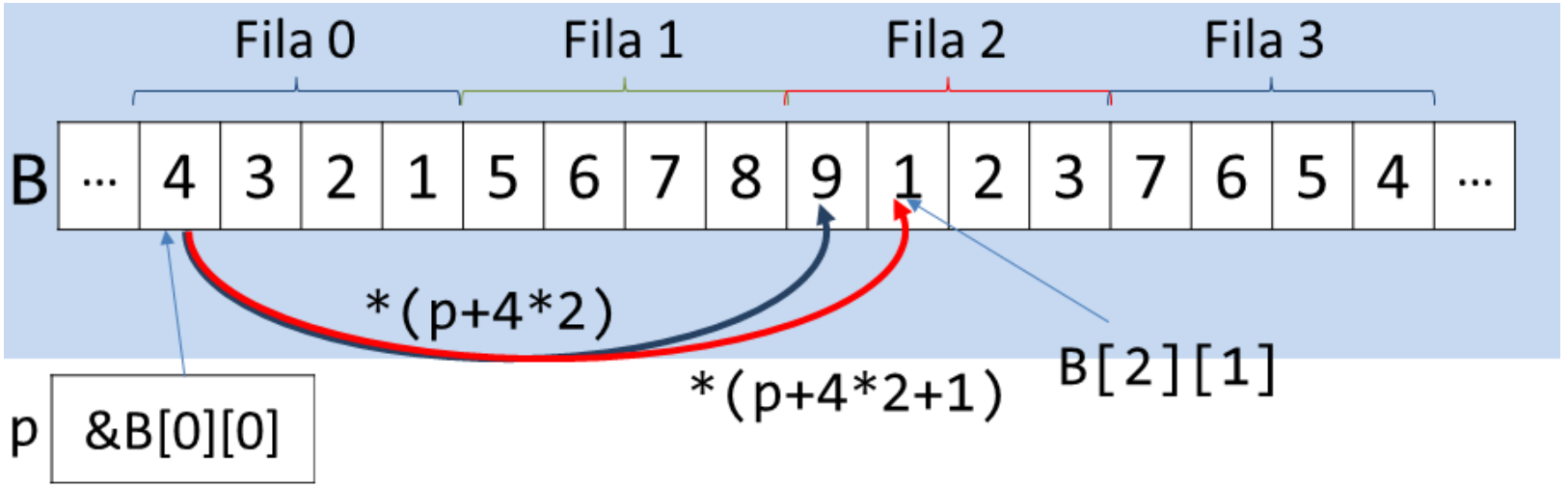
PUNTEROS Y MATRIZ/3



Se tiene que:

B[2][1] ⟷ *(p+4*2+1)

PUNTEROS Y MATRIZ/4



En general, para acceder al elemento C[i][j] de una matriz C de N columnas:

`C[i][j]` ⟷ `*(p+i*N+j)`

Fórmula de direccionamiento

PUNTEROS Y MATRIZ/4

```
float mat[2][4];          //declaración del arreglo
```

Col 0	Col 1	Col 2	Col 3
1.45	-23.5	-14.08	17.3
20	2.95	0.082	6.023

Con subíndices	Con **APUNTADORES**	Valor
mat[0][0]	*(*(mat+0)+0)	1.45
mat[0][1]	*(*(mat+0)+1)	-23.5
mat[0][2]	*(*(mat+0)+2)	-14.08
mat[0][3]	*(*(mat+0)+3)	17.3
mat[1][0]	*(*(mat+1)+0)	20
mat[1][2]	*(*(mat+1)+1)	2.95
mat[1][3]	*(*(mat+1)+2)	0.082
mat[1][4]	*(*(mat+1)+3)	6.023

PUNTEROS Y MATRIZ/4

Se debe prestar especial atención a lo siguiente: una matriz es un **vector de vectores**. Recuérdese también que, por la relación entre vectores y punteros, `(mat+i)` apunta a `mat[i]`.

Si la matriz tiene `N` columnas y si se hace `q = &mat[0][0]` (dirección base de la matriz), el elemento `mat[i][j]` puede ser accedido de varias formas. Basta recordar que dicho elemento tiene por delante `i` filas completas, y `j` elementos de su fila:

Formas de referencia a elementos de una matriz

*(q + N*i + j)	fórmula de direccionamiento
*(mat[i] + j)	primer elemento fila i desplazado j elementos
(*(mat + i))[j]	[j] equivale a sumar j a un puntero
(((mat + i)) + j)	Acceso a valores a través del nombre de la matriz

MATRIZ Y FUNCIONES

Así, para enviar una matriz a una determinada función, podemos proporcionar la dirección del primer elemento, y la cantidad de filas y columnas de la matriz. Aquí se muestra una forma de hacerlo.

```
void imprimirMatriz(int *p, int m, int n){
    int i,j;
    printf("La matriz es:\n");
    for(i=0;i<m;i++){
        for(j=0;j<n;j++){
            printf("%d\t",*(p+i*n+j));
        }
        printf("\n");
    }
    printf("\n");
}
```

```
int main(){
    int b[4][3]={{1,2,3},{4,5,6},
    {7,8,9},{10,11,12}};
    imprimirMatriz(&b[0][0],4,3);
    return 0;
}
```

GRACIAS POR LA ATENCIÓN