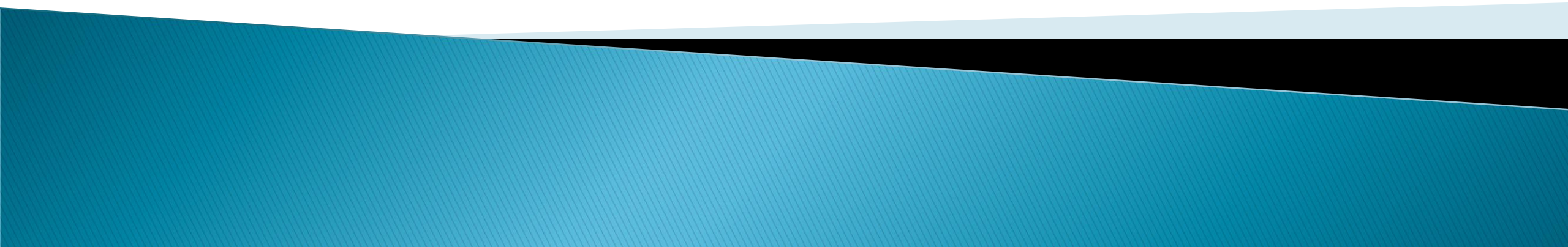


Asignación dinámica de memoria

Estructuras dinámicas



Matrices Dinámicas

- ▶ Es posible reservar memoria para una matriz de dos formas distintas
 - Como un único bloque homogéneo.
 - En este caso se accede a los elementos utilizando un desplazamiento a partir de la dirección inicial.
 - Como un vector de punteros a filas
 - Primero se reserva memoria para la dirección inicial de cada fila y luego se aloca cada fila específicamente.
 - En este caso se puede acceder a los elementos mediante índices.

Ejercicio 1 (para completar ...)

```
#include <stdio.h>
#include <string.h>
#include <stdlib.h>
#define FIL 3
#define COL 5
void VerMatriz(int *, int f, int c);
int main()
{   int M[FIL][COL] = {{1,2,3,4,5},
                        {6,7,8,9,10},
                        {11,12,13,14,15}};

    int *Dinamica=(int *) malloc(FIL*COL*sizeof(int));

    memcpy(Dinamica, M, sizeof(M));

    VerMatriz(Dinamica, FIL, COL);
    return(0);
}
```

← FALTA

Matrices dinámicas

- ▶ Ahora veremos como reservar memoria dinámicamente para una matriz de manera que sus elementos puedan ser accedidos a través de índices.
- ▶ La declaración y el acceso sería los siguientes

```
double ** matriz;
```

para trabajar con matrices con dos índices declaramos matrices dinámicas

```
...
```

```
matriz[2][5] = 123.45;
```

Veamos como
hacerlo

```

/* Programa asignar una matriz dinámica */
#include <stdlib.h>
int main(void)
{
    const int f=3; /* número de filas */
    const int c=7; /* número de columnas */
    double **mat; /* puntero de puntero */
    int i; /* contador */

    /* Asigno memoria (falta el control de errores)*/ debería preguntar if mat == NULL
    ➡ mat=(double**) calloc(f, sizeof(double*)); reserva X punteros a double

    for(i=0; i<f; i++) {
        mat[i]=(double*) calloc(c, sizeof(double));
    }

    < ... sigue ... >

```

f	3
c	7
mat	???
i	???

```

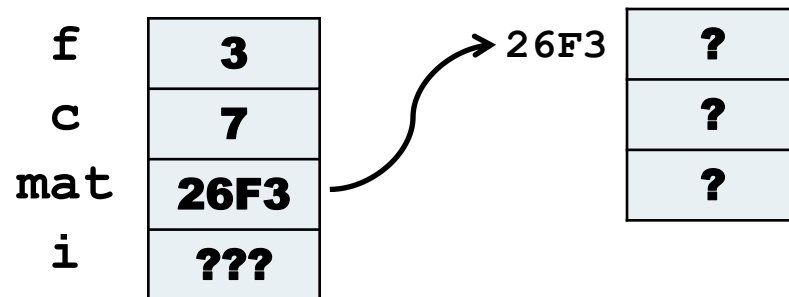
/* Programa asignar una matriz dinámica */
#include <stdlib.h>
int main(void)
{
    const int f=3; /* número de filas */
    const int c=7; /* número de columnas */
    double **mat; /* puntero de puntero */
    int i; /* contador */

    /* Asigno memoria (falta el control de errores)*/
    mat=(double**) calloc(f, sizeof(double*));

    ➡ for(i=0; i<f; i++) {
        mat[i]=(double*) calloc(c, sizeof(double));
    }

    < ... sigue ... >

```



```

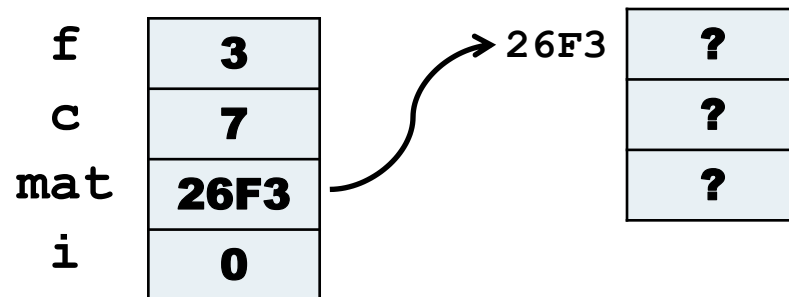
/* Programa asignar una matriz dinámica */
#include <stdlib.h>
int main(void)
{
    const int f=3; /* número de filas */
    const int c=7; /* número de columnas */
    double **mat; /* puntero de puntero */
    int i; /* contador */

    /* Asigno memoria (falta el control de errores)*/
    mat=(double**)calloc(f,sizeof(double*));

    ➡ for(i=0; i<f; i++) {
        mat[i]=(double*)calloc(c,sizeof(double));
    }

    < ... sigue ... >

```



```

/* Programa asignar una matriz dinámica */
#include <stdlib.h>
int main(void)
{
    const int f=3; /* número de filas */
    const int c=7; /* número de columnas */
    double **mat; /* puntero de puntero */
    int i; /* contador */

    /* Asigno memoria (falta el control de errores)*/
    mat=(double**) calloc(f, sizeof(double*));

```

```

    for(i=0; i<f; i++) {
        mat[i]=(double*) calloc(c, sizeof(double));
    }

```



< ... sigue ... >

f
c
mat
i

3
7
26F3
0

mat

→ 26F3

3A64
?
?

→ 3A64

c: num de filas

fila 0 de la matriz -> es otro vector en otro lado


```

/* Programa asignar una matriz dinámica */
#include <stdlib.h>
int main(void)
{
    const int f=3; /* número de filas */
    const int c=7; /* número de columnas */
    double **mat; /* puntero de puntero */
    int i; /* contador */

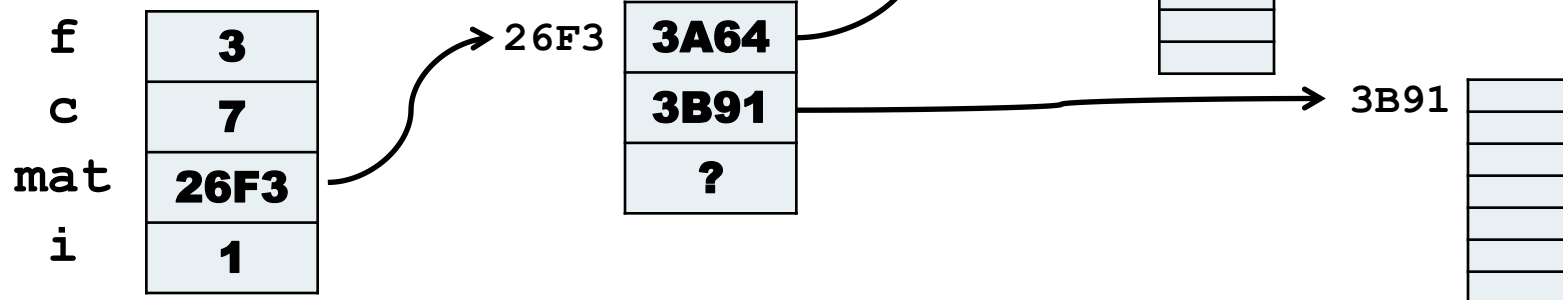
    /* Asigno memoria (falta el control de errores)*/
    mat=(double**) calloc(f, sizeof(double*));

    for(i=0; i<f; i++) {
        mat[i]=(double*) calloc(c, sizeof(double));
    }

```



< ... sigue ... >



```

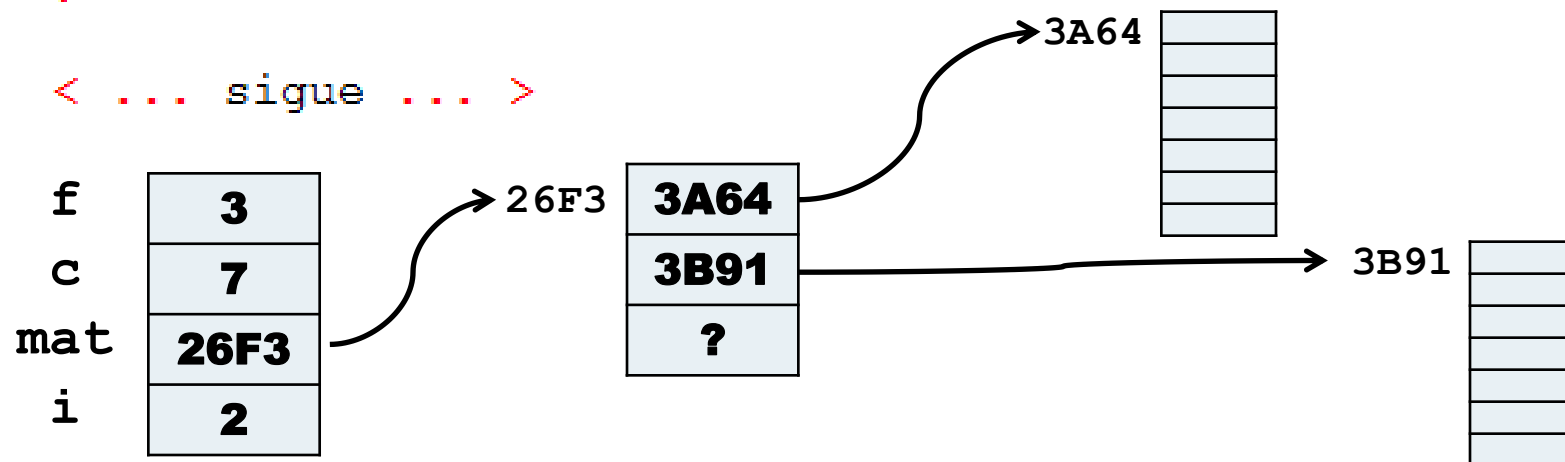
/* Programa asignar una matriz dinámica */
#include <stdlib.h>
int main(void)
{
    const int f=3; /* número de filas */
    const int c=7; /* número de columnas */
    double **mat; /* puntero de puntero */
    int i; /* contador */

    /* Asigno memoria (falta el control de errores)*/
    mat=(double**) calloc(f, sizeof(double*));

    ➡ for(i=0; i<f; i++) {
        mat[i]=(double*) calloc(c, sizeof(double));
    }

    < ... sigue ... >

```



```

/* Programa asignar una matriz dinámica */
#include <stdlib.h>
int main(void)
{
    const int f=3; /* número de filas */
    const int c=7; /* número de columnas */
    double **mat; /* puntero de puntero */
    int i; /* contador */

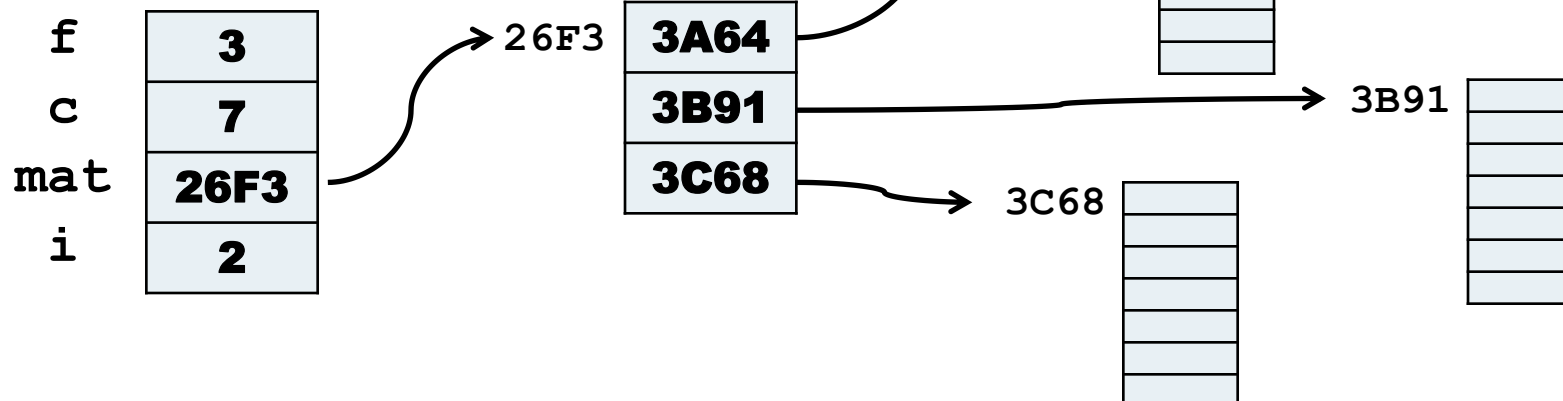
    /* Asigno memoria (falta el control de errores)*/
    mat=(double**) calloc(f, sizeof(double*));

    for(i=0; i<f; i++) {
        mat[i]=(double*) calloc(c, sizeof(double));
    }

```

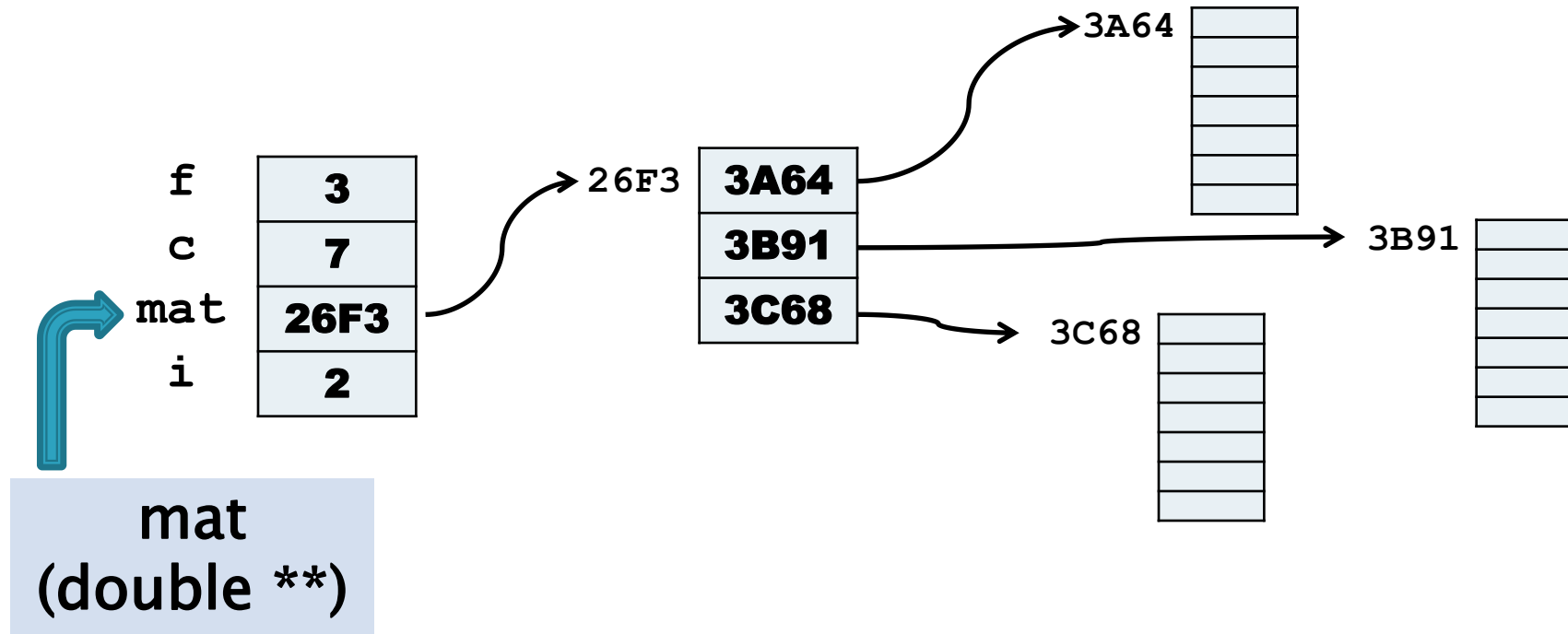


< ... sigue ... >



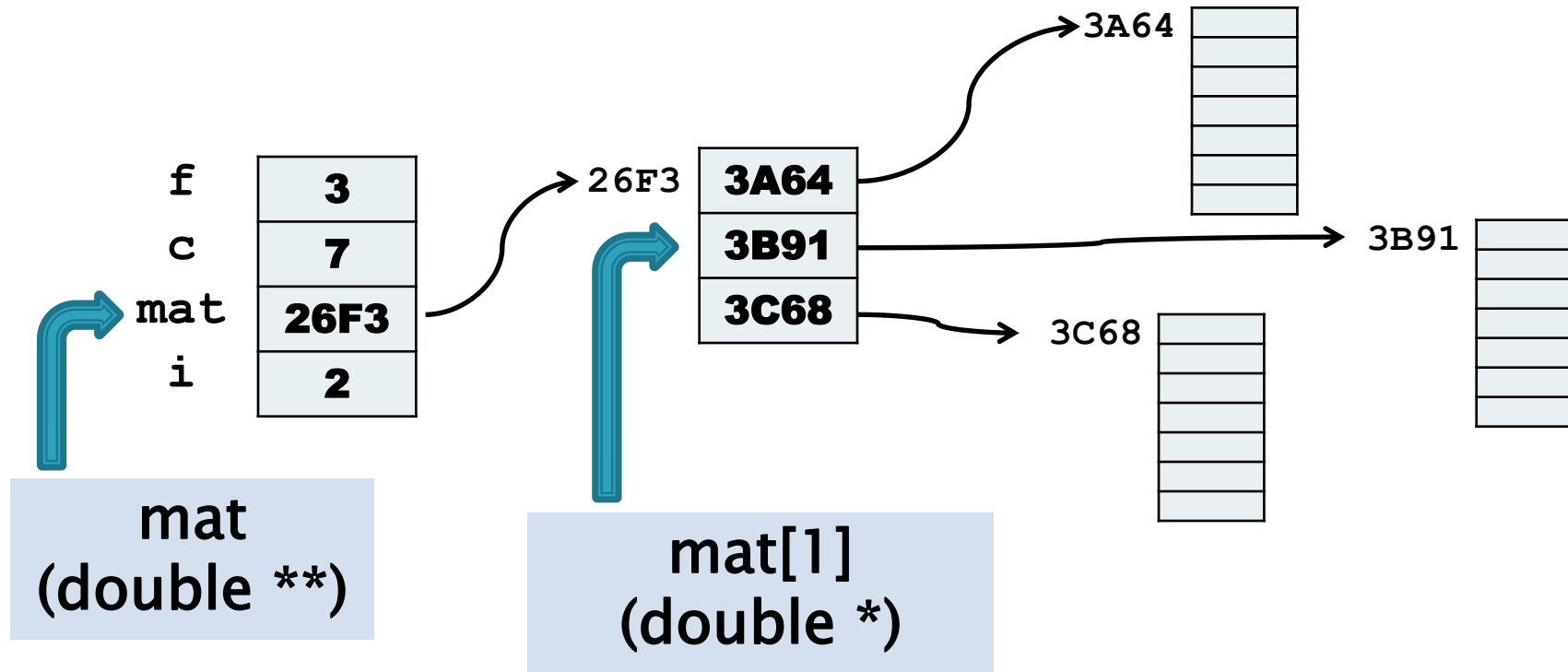
Accediendo a la matriz

`mat[1][4] = 23.4;`



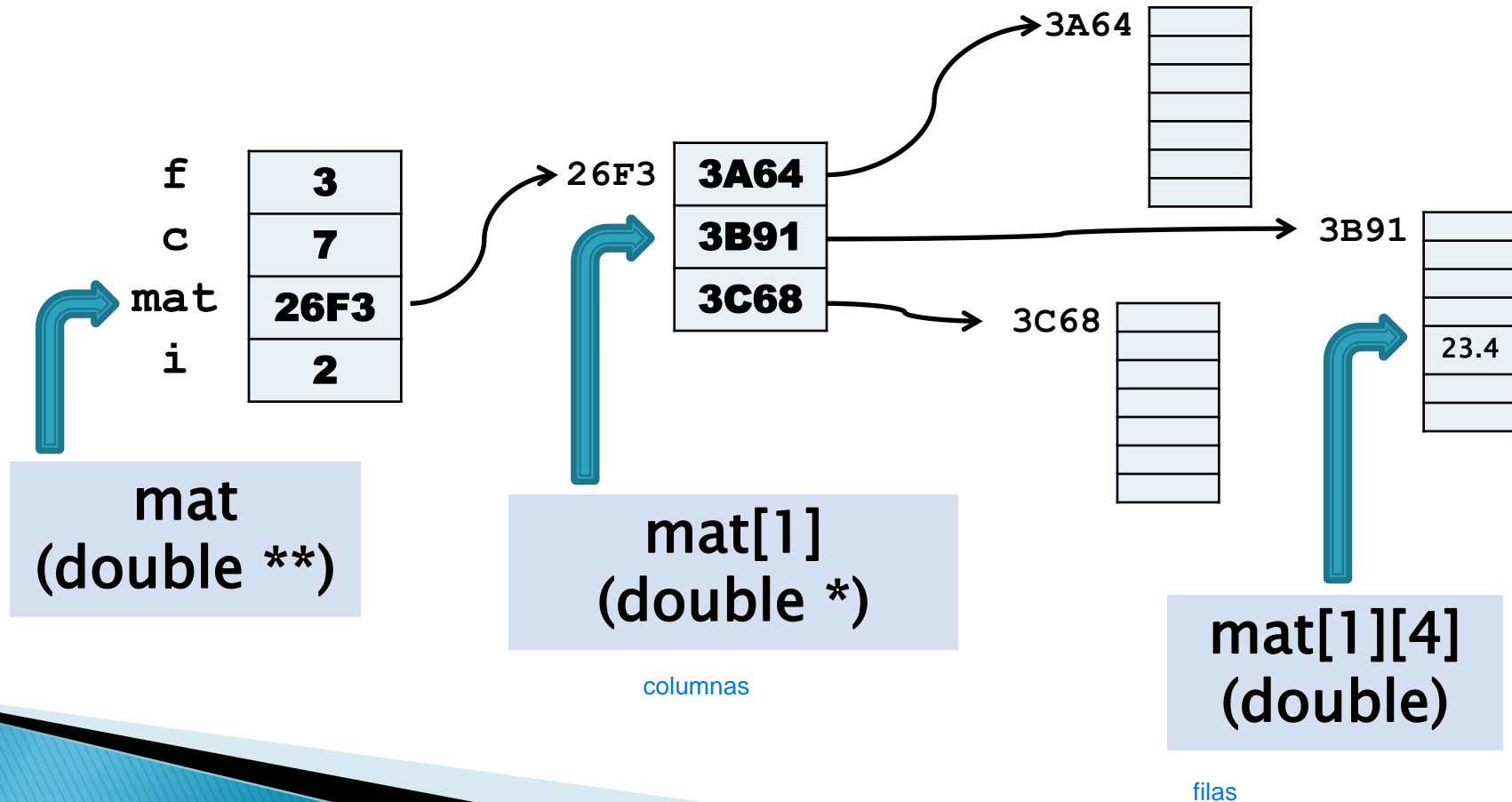
Accediendo a la matriz

`mat[1][4] = 23.4;`



Accediendo a la matriz

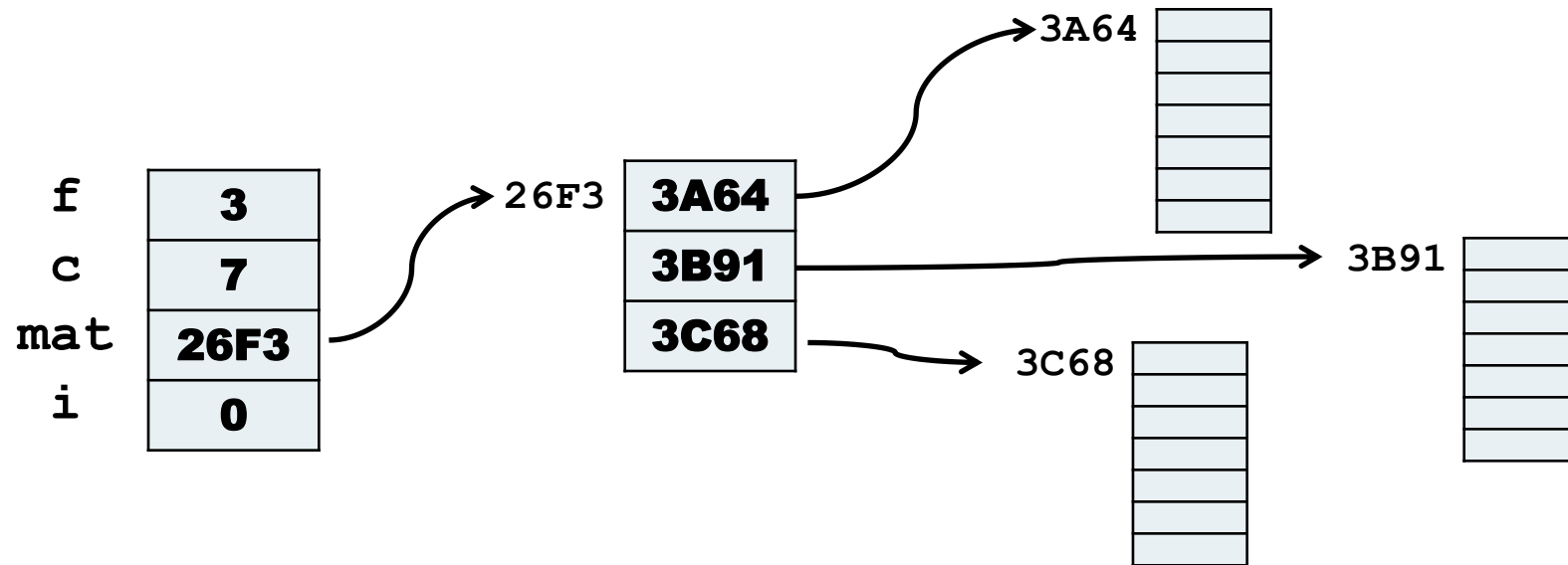
`mat[1][4] = 23.4;`



Liberando memoria

de atras para adelante

```
/* Liberar memoria */  
for(i=0; i<f; i++) {  
    free( mat[i] );  
}  
free(mat);
```

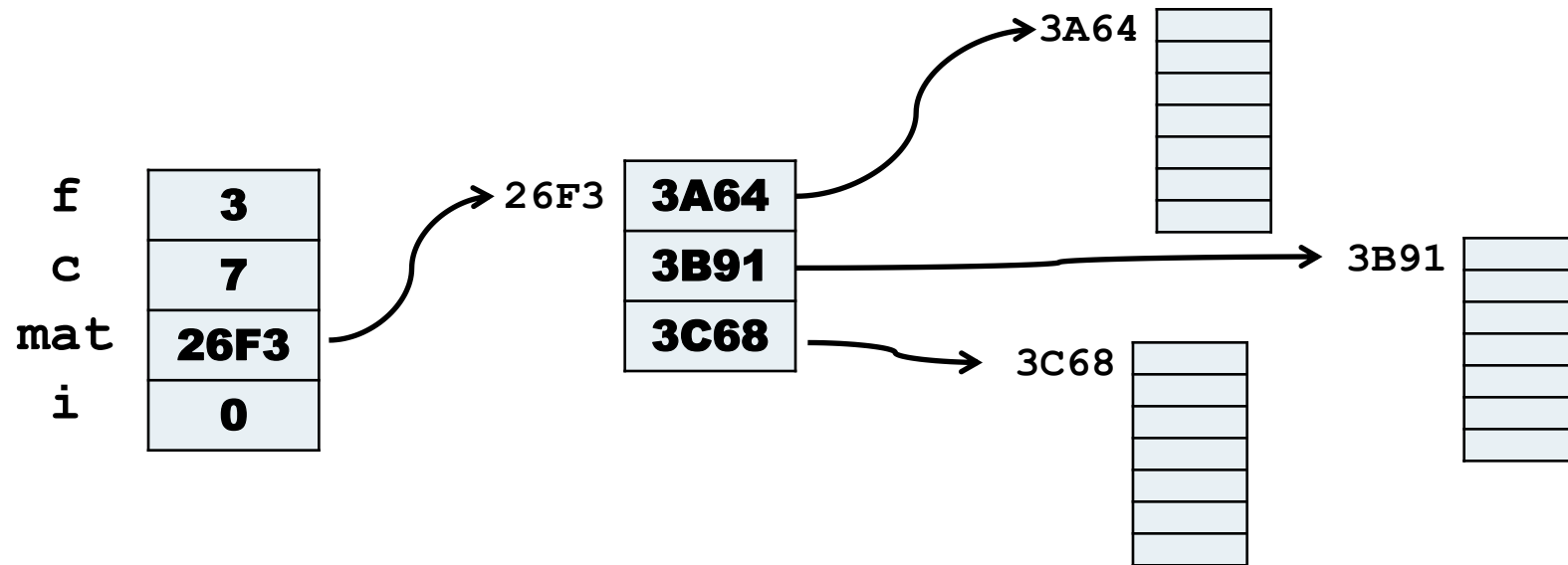


Liberando memoria

→

```
/* Liberar memoria */  
for(i=0; i<f; i++) {  
    free(mat[i]);  
}  
free(mat);
```

3A64

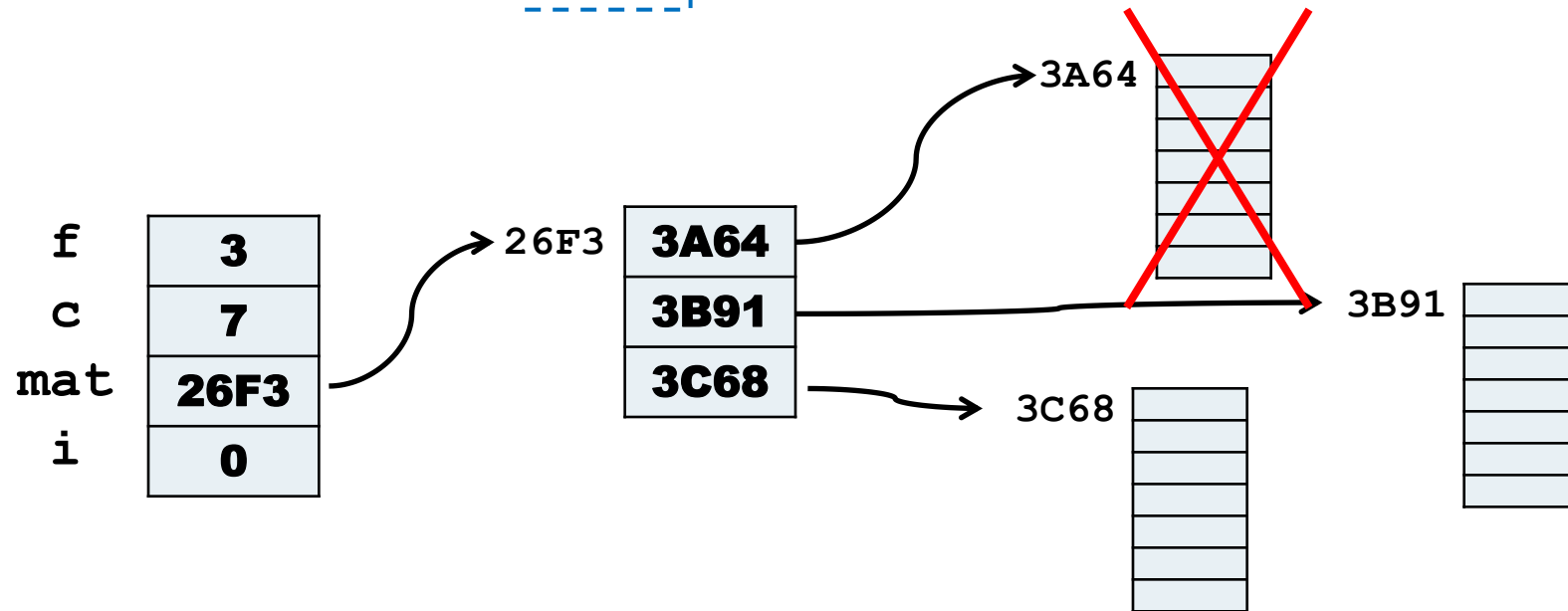


Liberando memoria

```
/* Liberar memoria */  
for(i=0; i<f; i++) {  
    free(mat[i]);  
}  
free(mat);
```

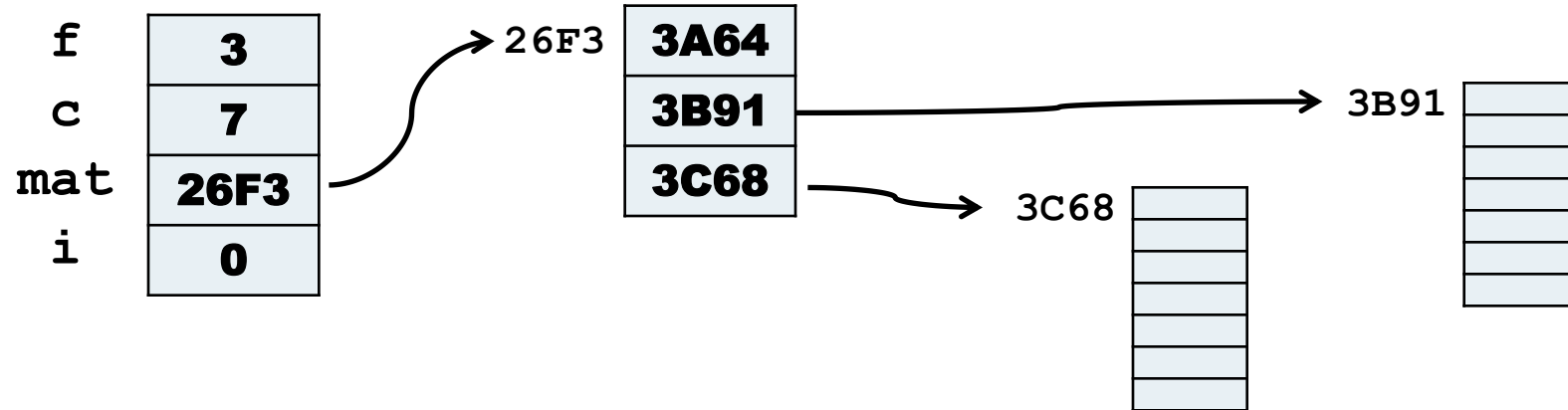
→

3A64



Liberando memoria

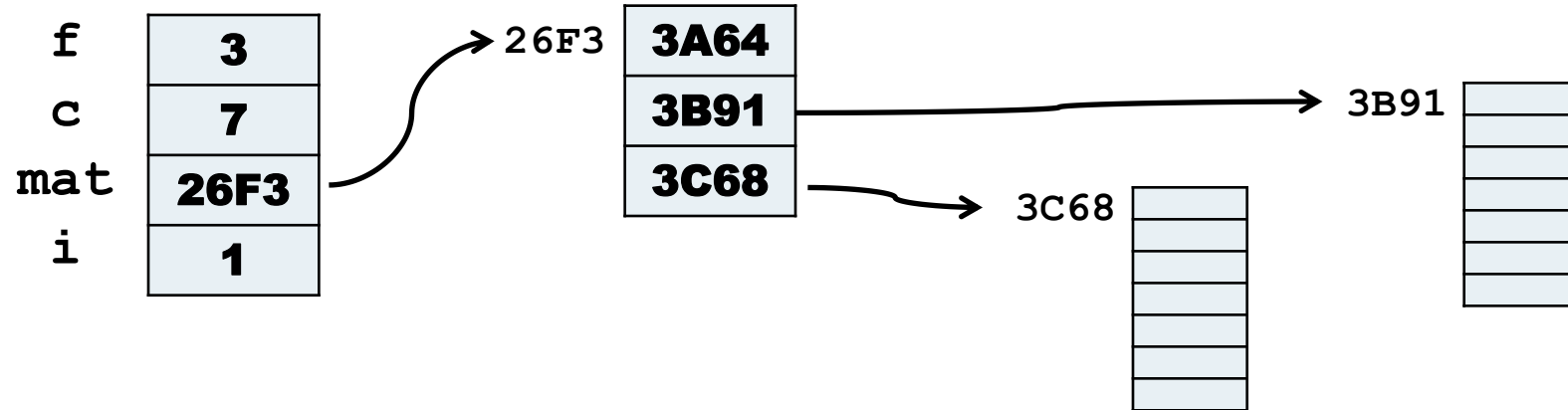
```
/* Liberar memoria */  
for(i=0; i<f; i++) {  
    free( mat[i] );  
}  
free(mat);
```



Liberando memoria



```
/* Liberar memoria */  
for(i=0; i<f; i++) {  
    free( mat[i] );  
}  
free(mat);
```

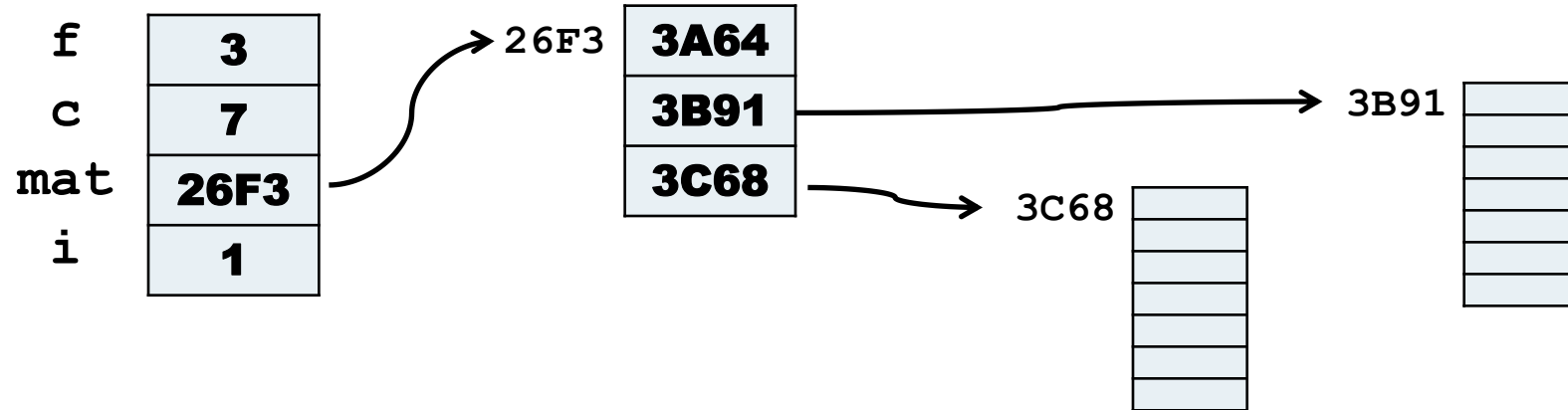


Liberando memoria

→

```
/* Liberar memoria */  
for(i=0; i<f; i++) {  
    free(mat[i]);  
}  
free(mat);
```

3B91

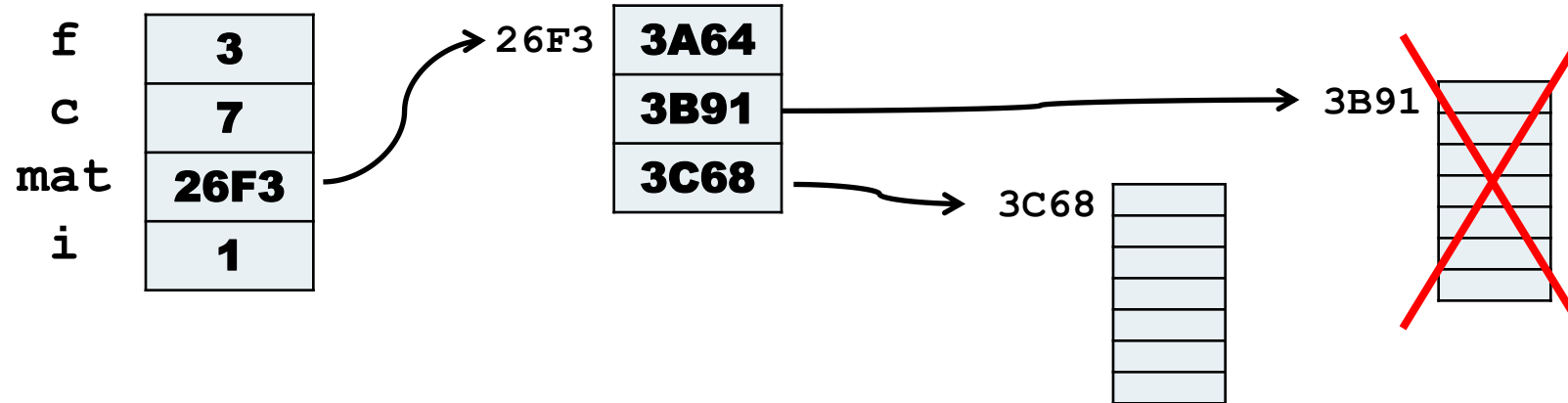


Liberando memoria

➡

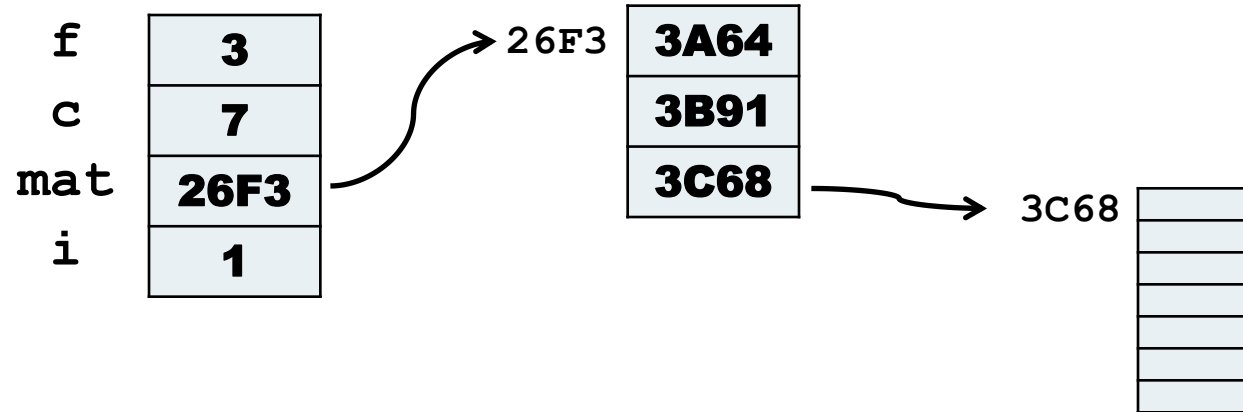
```
/* Liberar memoria */  
for(i=0; i<f; i++) {  
    free(mat[i]);  
}  
free(mat);
```

3B91



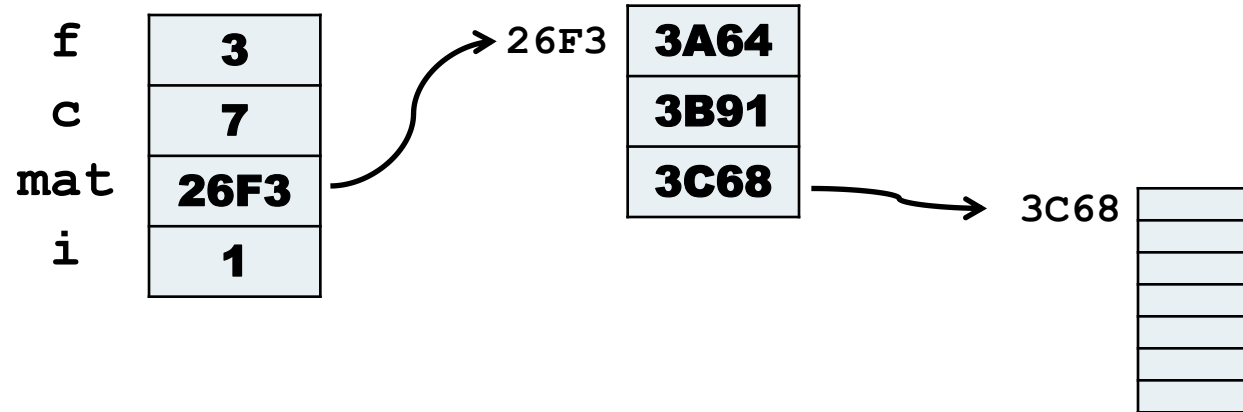
Liberando memoria

```
/* Liberar memoria */  
for(i=0; i<f; i++) {  
    free( mat[i] );  
}  
free(mat);
```



Liberando memoria

```
/* Liberar memoria */  
for(i=0; i<f; i++) {  
    free(mat[i]);  
}  
free(mat);
```

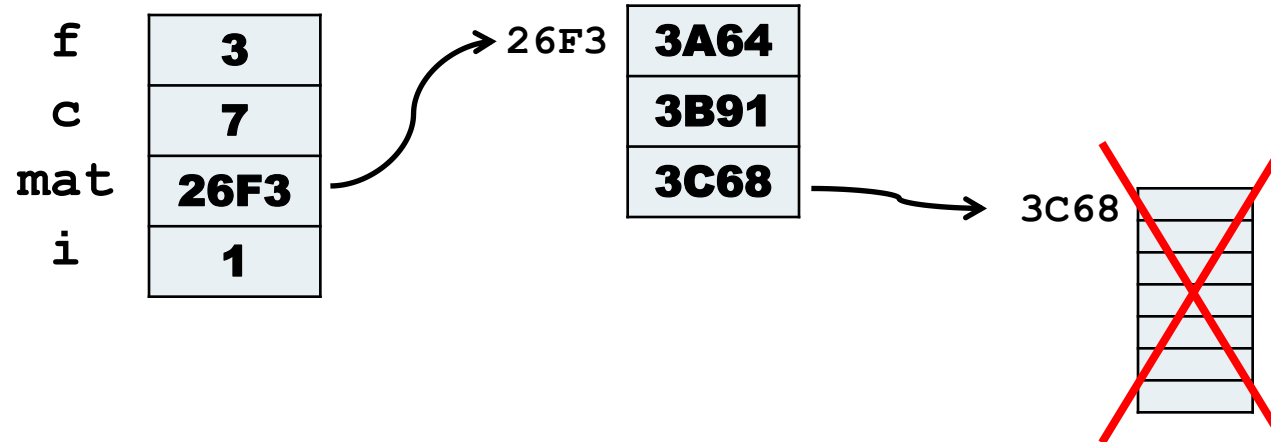


Liberando memoria

➡

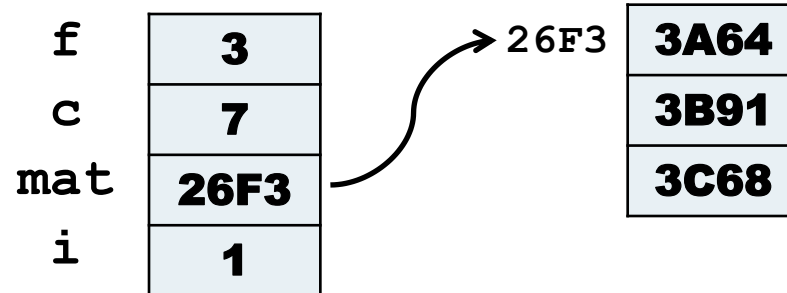
```
/* Liberar memoria */  
for(i=0; i<f; i++) {  
    free(mat[i]);  
}  
free(mat);
```

3B91



Liberando memoria

```
/* Liberar memoria */  
for(i=0; i<f; i++) {  
    free( mat[i] );  
}  
free(mat);
```

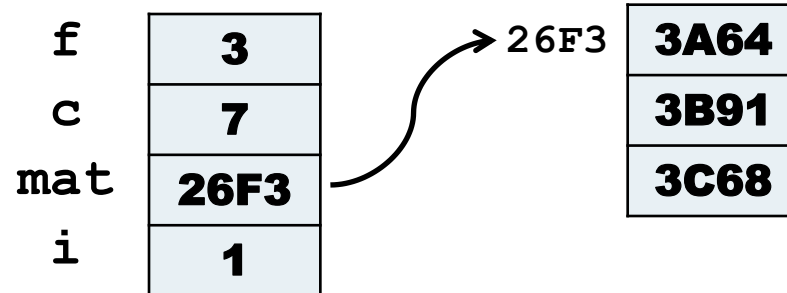


Liberando memoria

```
/* Liberar memoria */  
for(i=0; i<f; i++) {  
    free( mat[i] );  
}
```



```
free(mat);
```



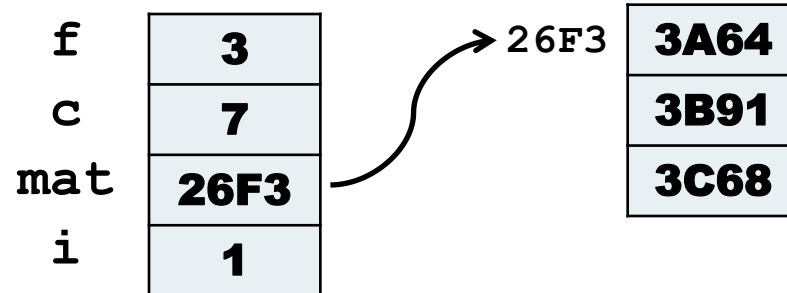
Liberando memoria

```
/* Liberar memoria */  
for(i=0; i<f; i++) {  
    free( mat[i] );  
}
```



```
free(mat);
```

26F3



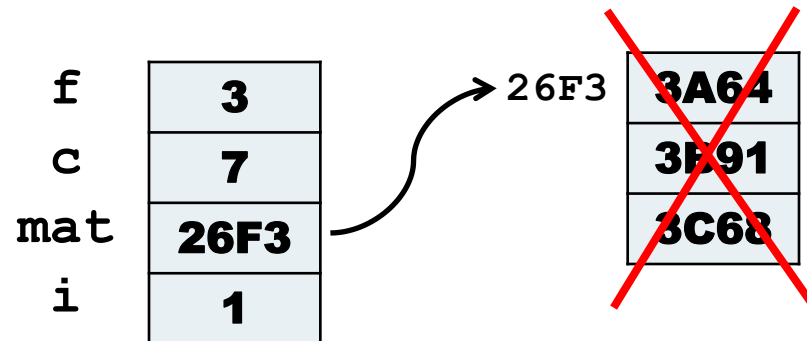
Liberando memoria

```
/* Liberar memoria */  
for(i=0; i<f; i++) {  
    free( mat[i] );  
}
```



```
free(mat);
```

26F3



Liberando memoria

```
/* Liberar memoria */  
for(i=0; i<f; i++) {  
    free( mat[i] );  
}  
free(mat);
```



f	3
c	7
mat	26F3
i	1

```
#include <stdio.h>
#include <stdlib.h>
#define FIL 3
#define COL 5
void VerMatriz(int **, int , int);
void Liberar(int **, int);
int main()
{   int f;
    int M[FIL][COL]={1,2,3,4,5},
                    {6,7,8,9,10},
                    {11,12,13,14,15}};

    int **Dinamica = (int **) malloc(FIL*sizeof(int *));

    for (f=0; f<FIL; f++){
        Dinamica[f]=(int *) malloc(COL*sizeof(int));
        // memcpy(Dinamica[f], M[f], sizeof(M[f])); // es correcto
        memcpy(Dinamica[f], M[f], COL*sizeof(int));
    }
    VerMatriz(Dinamica, FIL, COL);
    Liberar(Dinamica, f); //falta implementar
    return(0);
}
```

Diagrama de anotaciones:


- Una flecha roja apunta desde el comentario `// es correcto` hacia la línea `memcpy(Dinamica[f], M[f], COL*sizeof(int));`.
- Una flecha azul apunta desde el comentario `comienzo de la fila` hacia el índice `f` en `M[f]`.
- Una flecha azul apunta desde el comentario `cantidad q tiene q copiar y el tamaño` hacia el argumento `COL*sizeof(int)`.

- Note que sólo puede acceder a los elementos a través de los índices


```
void VerMatriz(int **M, int fil, int col)
{ int f,c;

  for (f=0; f<fil; f++) {
    for(c=0; c<col; c++)
      // printf("%2d ", *(M+col*f+c) );    // es INCORRECTO
      // printf("%2d ", *M++ );           // es INCORRECTO
      printf("%2d ", M[f][c] );           // es CORRECTO
    printf("\n");
  }
}
```


Ejercicio 2

- ▶ Declare y reserve memoria para una matriz de enteros de N filas y M columnas. Los valores de N y M se leen de teclado.
 - ▶ Luego, cargue la matriz.
 - ▶ Recorra la matriz e imprima la suma de los valores de cada fila.
 - ▶ Resuelva el ejercicio de dos formas distintas
 - Utilizando un único bloque homogéneo.
 - Utilizando un vector de punteros a filas.
- 

Ejercicio 3

- ▶ En un torneo se inscriben 8 equipos de 5 integrantes cada uno. De cada persona se registra nombre (hasta 20 caracteres) y edad (número entero).
 - ▶ Declare y reserve memoria para una matriz dinámica que le permita cargar la información.
 - ▶ Defina una función para leer desde teclado la información de una persona. Es decir que recibe un puntero a una estructura y la completa con el nombre y la edad.
 - ▶ Utilice la función anterior para cargar la información en la matriz.
- 

Ejercicio 4

- ▶ Se lee una secuencia de números enteros terminada en 0 (cero).
 - ▶ Al finalizar la lectura se deberá imprimir en pantalla, los primeros 3 números ingresados que terminen en cero, luego los primeros 3 que terminen en 1 y así siguiendo.
 - ▶ Si tales números no existen, no debe imprimirse nada.
 - ▶ Vea el ejemplo en la transparencia siguiente
- 

Ingrese nros (0 = salir)

12

65

7002

985

1

34

202

11

42

52

0

Los 3 primeros de c/digito son:

1 11

12 7002 202

34

65 985

Ejercicio 4

- Resolver este ejercicio utilizando
 - `int matriz[10][3], cant[10];`
 - `int * matriz, cant[10];`
 - `int ** matriz, *cant;`
 - `struct { int nros[3];
 int cant; } matriz[10];`
 - `struct { int *nros;
 int cant; } * matriz;`