

**Metodología de la Programación**

## Tema 1.- La memoria dinámica

**La memoria dinámica**

**Ejemplos de estructuras con memoria dinámica**

**Vectores dinámicos**

**Matrices dinámicas**

Metodología de la Programación - Javier Martínez Baena/Joaquín Fdez-Valdivia - Dpto. Ciencias de la Computación e I.A. (Univ. Granada) | 1

**La memoria dinámica**

**Estructura de la memoria**

Metodología de la Programación - Javier Martínez Baena/Joaquín Fdez-Valdivia - Dpto. Ciencias de la Computación e I.A. (Univ. Granada) | 2

 **La memoria dinámica** 

### El operador new

**C: malloc, free**  
**C++: new, delete**

**NO MEZCLAR !!!**

El operador **new** permite reservar nueva memoria.

```
tipo_de_dato *ptr;
...
ptr = new tipo_de_dato;
```

Devuelve un puntero a la zona de memoria reservada.

```
int *p1;      float *p2;
p1 = new int;
p2 = new float;
```

También se puede usar de esta forma (**NO SE RECOMIENDA**):

```
tipo_de_dato *ptr;
...
ptr = new (nothrow) tipo_de_dato
```

Idem. En caso de que no se haya podido hacer la reserva devuelve el puntero nulo (0).

Metodología de la Programación - Javier Martínez Baena/Joaquín Fdez-Valdivia - Dpto. Ciencias de la Computación e I.A. (Univ. Granada)  3

 **La memoria dinámica** 

### La instrucción delete

La instrucción **delete** permite liberar memoria que *previamente* ha sido reservada con **new**.

```
delete ptr; Libera la memoria apuntada por el puntero ptr.
```

Para poder liberar un puntero, primero se debe haber hecho un **new** con éxito.

¿Qué ocurre?

```
int *p1, *p2;
int x=7;

p1 = &x;
p2 = new int;

*p1 = 8;
*p2 = 9;

delete p2;
delete p1;
```

```
struct Celda {
    int x;
    Celda *p;
};

Celda *ptr;

ptr = new Celda;
ptr->p = new Celda;
ptr->p->p = new Celda;
ptr->p->p->p = ptr;

delete ptr->p->p;
delete ptr->p;
delete ptr;
```

¿Importa el orden?

```
delete ptr;
delete ptr->p;
delete ptr->p->p;
```

Metodología de la Programación - Javier Martínez Baena/Joaquín Fdez-Valdivia - Dpto. Ciencias de la Computación e I.A. (Univ. Granada)  4

**Metodología de la Programación**

## Tema 3.- La memoria dinámica

**La memoria dinámica**

**Ejemplos de estructuras con memoria dinámica**

**Vectores dinámicos**

**Matrices dinámicas**

Metodología de la Programación - Javier Martínez Baena/Joaquín Fdez-Valdivia - Dpto. Ciencias de la Computación e I.A. (Univ. Granada) | 5

**Ejemplos de estructuras con memoria dinámica**

**Listas enlazadas**

Construye un programa que permita **crear** las siguientes estructuras para almacenar una lista de números enteros:

Implementa un programa que haga un **listado** de ambas.

Implementa un trozo de programa que **inserte** un elemento en una posición arbitraria.

Implementa un trozo de programa que **borre** un elemento de una posición arbitraria.

Implementa un programa que haga un **listado en orden inverso** de ambas.

Metodología de la Programación - Javier Martínez Baena/Joaquín Fdez-Valdivia - Dpto. Ciencias de la Computación e I.A. (Univ. Granada) | 6

**Ejemplos de estructuras con memoria dinámica**

**Árboles**

Los árboles son estructuras de datos que se usan, por ejemplo, para representar expresiones aritméticas. Están formados por nodos enlazados entre sí que se disponen de forma jerárquica (en forma de árbol).

Cada nodo suele ser de la siguiente forma:

```
struct nodo {
    string dato;      // Información
    nodo *hijo_izq, *hijo_der, *padre;
};
```

Dada la siguiente expresión:  
 $(a+b*4)-7*y$

Implementa el trozo de código que construye un árbol con el siguiente esquema:

A continuación libera la memoria de la estructura

Metodología de la Programación - Javier Martínez Baena/Joaquín Fdez-Valdivia - Dpto. Ciencias de la Computación e I.A. (Univ. Granada) | Back | Next | Previous | Home | 7

**Metodología de la Programación**

## Tema 3.- La memoria dinámica

**La memoria dinámica**

**Ejemplos de estructuras con memoria dinámica**

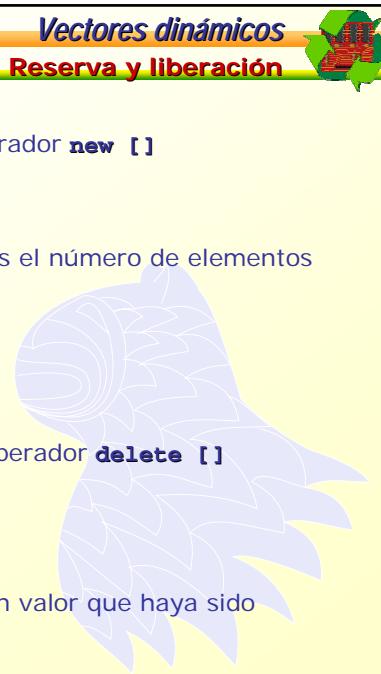
**Vectores dinámicos**

**Matrices dinámicas**

Metodología de la Programación - Javier Martínez Baena/Joaquín Fdez-Valdivia - Dpto. Ciencias de la Computación e I.A. (Univ. Granada) | Back | Next | Previous | Home | 8

**Vectores dinámicos**

**Reserva y liberación**



La reserva de memoria se hace con el operador **new []**

```
tipo_de_dato *ptr;
...
ptr = new tipo_de_dato [N];
```

N es el número de elementos

```
int *p1;
p1 = new int[10];
```

La liberación de memoria se hace con el operador **delete []**

```
delete [] ptr;
```

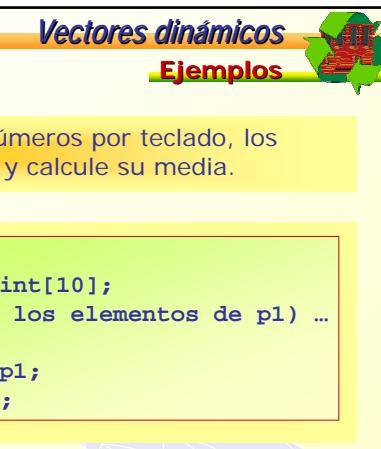
```
delete [] p1;
```

Sólo podemos hacer **delete[]** sobre un valor que haya sido devuelto por un **new[]**

Metodología de la Programación - Javier Martínez Baena/Joaquín Fdez-Valdivia - Dpto. Ciencias de la Computación e I.A. (Univ. Granada) | < < < > > > | 9

**Vectores dinámicos**

**Ejemplos**



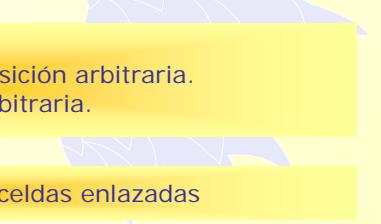
Haz un programa que lea una serie de números por teclado, los almacene en memoria dinámica (vector) y calcule su media.

¿Qué ocurre en este código?

```
int *p1;
p1 = new int[10];
... (leemos los elementos de p1) ...
p1++;
cout << *p1;
delete p1;
```

Dado el vector del ejercicio anterior:

- Inserta un nuevo elemento en una posición arbitraria.
- Borra un elemento de una posición arbitraria.



Implementa el ejercicio anterior usando celdas enlazadas

Metodología de la Programación - Javier Martínez Baena/Joaquín Fdez-Valdivia - Dpto. Ciencias de la Computación e I.A. (Univ. Granada) | < < < > > > | 10

**Metodología de la Programación**

## Tema 3.- La memoria dinámica

**La memoria dinámica**

**Ejemplos de estructuras con memoria dinámica**

**Vectores dinámicos**

**Matrices dinámicas**

Universidad de Granada DECSAI Universidad de Granada

Metodología de la Programación - Javier Martínez Baena/Joaquín Fdez-Valdivia - Dpto. Ciencias de la Computación e I.A. (Univ. Granada) 11

**Matrices dinámicas**

**Implementación 1**

No es posible hacer una reserva directa de memoria para matrices 2D  
Hay que ingeníárselas usando vectores (1D).

**Una posible implementación: Usando un único vector 1D**

La matriz:

1	2	3
4	5	6
7	8	9
10	11	12

La estructura para almacenar la matriz:

```
int *m;
int fil, col;
m = new int[fil*col];
```

Operaciones:

- { ¿Acceso al elemento (i,j)?
- ¿Insertar o borrar filas y columnas?

Metodología de la Programación - Javier Martínez Baena/Joaquín Fdez-Valdivia - Dpto. Ciencias de la Computación e I.A. (Univ. Granada) 12

**Matrices dinámicas**  
**Implementación 2**

Otra posible implementación: Usando un vector 1D de punteros a vectores 1D

La matriz

1	2	3
4	5	6
7	8	9
10	11	12

La estructura para almacenar la matriz:

m → [ fila 1: 1 2 3 ]  
       [ fila 2: 4 5 6 ]  
       [ fila 3: 7 8 9 ]  
       [ fila 4: 10 11 12 ]

```
int **m;
int fil, col;
m = new int*[fil];
for (int i=0; i<col; i++)
    m[i] = new int[col];
```

¿Acceso al elemento (i,j)?  
 ¿Insertar o borrar filas y columnas?  
 ¿Reserva y liberación de memoria?

Metodología de la Programación - Javier Martínez Baena/Joaquín Fdez-Valdivia - Dpto. Ciencias de la Computación e I.A. (Univ. Granada) | 13

**Matrices dinámicas**  
**Implementación 3**

Otra posible implementación: Usando un vector 1D de punteros a vectores 1D

La matriz

1	2	3
4	5	6
7	8	9
10	11	12

La estructura para almacenar la matriz:

m → [ 1 2 3 4 5 6 7 8 9 10 11 12 ]

```
int **m;
int fil, col;
m = new int*[fil];
m[0] = new int[fil*col];
for (int i=1; i<fil; i++)
    m[i] = m[0]+i*col;
```

¿Acceso al elemento (i,j)?  
 ¿Insertar o borrar filas y columnas?  
 ¿Reserva y liberación de memoria?

Metodología de la Programación - Javier Martínez Baena/Joaquín Fdez-Valdivia - Dpto. Ciencias de la Computación e I.A. (Univ. Granada) | 14