

 Metodología de la Programación C++

## Tema 3.- Constructores y destructores

- El constructor por defecto*
- Otros constructores*
- El destructor*
- El constructor de copias*
- Clases como atributos de otras clases*
- Llamadas explícitas a los constructores*



UGR 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) [navigation icons]

 El constructor por defecto C++  
Una clase polinomio

**Ejemplo:** vamos a construir una clase para trabajar con polinomios.

```
class Polinomio {
    // La representación interna es privada
private:
    float *p;      // Vector de coeficientes
    int ncoef;    // Número de coeficientes en p
    int grado;    // Número de coef. utilizados

    // La interfaz es pública
public:
    void SetCoef(const int coef, const float val);
    float GetCoef(const int coef) const;
    int GetGrado(void) const;
};
```

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

*El constructor por defecto C++*

### Una clase polinomio: la implementación

```

int Polinomio::GetGrado() const
{
    return grado;
}

float Polinomio::GetCoef(const int coef) const
{
    return ((coef>=0) && (coef<=grado)) ? p[coef] : 0;
}

```

Estos dos métodos podrían ser **inline**:

- Son muy **cortos**: no supone una gran ineficiencia en espacio.
- Previsiblemente, serán usados con mucha **frecuencia**: ganamos en eficiencia temporal.

Metodología de la Programación · Javier Martínez Baena/Joaquín Fdez-Valdivia · Dpto. Ciencias de la Computación e I.A. (Univ. Granada) [Navigation icons]

*El constructor por defecto C++*

### Una clase polinomio: la implementación

```

void Polinomio::SetCoef(const int coef, const float val) {
    if (coef>=0) {                                // Si coef es válido ...
        if (coef>=ncoef) {                         // Aumentar memoria si hace falta
            float *aux = new float[coef+1];          // Reservamos nueva memoria
            if (ncoef>0) {                          // Si había coeficientes almacenados:
                for (int i=0; i<ncoef; i++) // 1.- Copiamos los datos a la
                    aux[i] = p[i];                  //     nueva memoria
                delete [] p;                      // 2.- Liberamos memoria antigua
            }
            for (int i=ncoef; i<coef; i++) // Ponemos a 0 la nueva memo.
                aux[i] = 0;
            p = aux;                           // Reasignamos puntero con coeficientes
            ncoef = coef+1;                  // Asignamos el nuevo número de coeficientes
        }
        p[coef] = val;                         // Asignamos el nuevo coeficiente
        if (val==0) {                          // Si el coeficiente es cero:
            for (int i=coef; i>0 && p[i]; i--) // Calculamos nuevo grado
                grado--;
        } else if (coef>grado)           // Calculamos el nuevo grado
            grado = coef;
    }
}

```

Metodología de la Programación · Javier Martínez Baena/Joaquín Fdez-Valdivia · Dpto. Ciencias de la Computación e I.A. (Univ. Granada) [Navigation icons]

 **El constructor por defecto C++**  
**Usando la clase polinomio**

```
int main(int argc, char *argv[])
{
    Polinomio p1;
    p1.SetCoef(3,4.5);
}
```

En el momento de hacer la llamada a `SetCoef` ... ¿Cuánto valen los atributos de `p1`?

En la declaración únicamente se le reservó memoria al objeto `p1` pero:  
→ No se reservó memoria para el miembro `p`.  
→ No se inicializó ninguno de los otros atributos (`grado`, `ncoef`).

Necesitamos un método que **inicialice** un objeto de tipo `Polinomio`.

```
void Polinomio::Inicializar(void)
```

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

 **El constructor por defecto C++**  
**Usando la clase polinomio**

```
void Polinomio::Inicializar(void)
{
    p = 0;
    grado = 0;
    ncoef = 0;
}
```

Debemos llamar a este método **antes** de hacer ninguna otra cosa con los objetos de tipo `Polinomio`.



```
int main(int argc, char *argv[])
{
    Polinomio p1;
    p1.Inicializar(); // Inicializamos p1
    p1.SetCoef(3,4.5);
}
```

*Si se nos olvida → El programa no funciona.*

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

 **El constructor por defecto** C++  
**El constructor por defecto**

Podemos automatizar este proceso usando el **constructor por defecto**.

*El constructor por defecto es un **método con el mismo nombre que la clase** que es ejecutado en el momento de la creación de los objetos de esa clase (de forma automática).*

El constructor por defecto	No tiene ningún parámetro. No devuelve nada (ni siquiera <code>void</code> ). Generalmente es <code>public</code> .
----------------------------	---

```
class Polinomio {
    private:
        ...
    public:
        ...
        Polinomio();
};
```

```
Polinomio::Polinomio()
{
    p = 0;
    grado = 0;
    ncoef = 0;
}

int main(int argc, char *argv[])
{
    Polinomio p1; // Aquí se llama a C.D.
    p1.SetCoef(3,4.5);
}
```

Metodología de la Programación · Javier Martínez Baena/Joaquín Fdez-Valdivia · Dpto. Ciencias de la Computación e I.A. (Univ. Granada) [Navigation icons] 7

 **El constructor por defecto** C++  
**El constructor por defecto**

*Ejemplo de uso:*  
Crear el polinomio  
 $3x^3 + 4x^2 - 3x + 2$

```
int main(int argc, char *argv[])
{
    Polinomio p1;
    p1.SetCoef(0,2);
    p1.SetCoef(1,-3);
    p1.SetCoef(2,4);
    p1.SetCoef(3,3);
}
```

Cada vez que añadimos un coeficiente reservamos de nuevo memoria.  
El constructor podría reservar algo de memoria por defecto:

```
Polinomio::Polinomio()
{
    p = new float[100];
    for (int i=0; i<100; i++) p[i]=0;
    grado = 0;
    ncoef = 100;
}
```

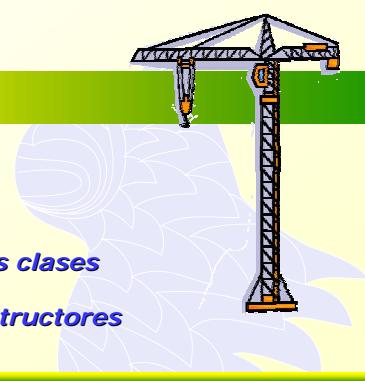
¿Pero ... porqué 100 y no 1000 ó 10?

Metodología de la Programación · Javier Martínez Baena/Joaquín Fdez-Valdivia · Dpto. Ciencias de la Computación e I.A. (Univ. Granada) [Navigation icons] 8

 Metodología de la Programación C++

## Tema 6.- Constructores y destructores

- El constructor por defecto*
- Otros constructores*
- El destructor*
- El constructor de copias*
- Clases como atributos de otras clases*
- Llamadas explícitas a los constructores*



Metodología de la Programación - Javier Martínez Baena/Joaquín Fdez-Valdivia - Dpto. Ciencias de la Computación e I.A. (Univ. Granada) [navigation icons] 9

 Otros constructores C++

### Constructores con parámetros

¿Podemos decirle al constructor el tamaño inicial que queremos?

↓

Podemos crear **otro** constructor con un parámetro.

```
class Polinomio {
    private:
    ...
public:
    ...
    Polinomio();
    Polinomio(int n);
};

...
Polinomio p(10);
Polinomio r;
Polinomio t(0);
...
```

```
Polinomio::Polinomio(int n)
{
    if (n>0) {
        p = new float[n];
        for (int i=0; i<n; i++) p[i]=0;
        grado = 0;
        ncoef = n;
    } else {
        p = 0;
        grado = 0;
        ncoef = 0;
    }
}
```

Estamos repitiendo el código del constructor por defecto.

Metodología de la Programación - Javier Martínez Baena/Joaquín Fdez-Valdivia - Dpto. Ciencias de la Computación e I.A. (Univ. Granada) [navigation icons] 10

**Otros constructores C++**

Para evitar duplicar código:

```
class Polinomio {
    private:
        ...
        void PonerACero(void);
    public:
        ...
        Polinomio();
        Polinomio(int n);
};
```

Llamada equivalente  
this->PonerACero();

¿Podemos resumir los dos constructores en uno solo?

```
void Polinomio::PonerACero(void)
{
    p = 0;
    grado = 0;
    ncoef = 0;
}

Polinomio::Polinomio()
{
    PonerACero();
}

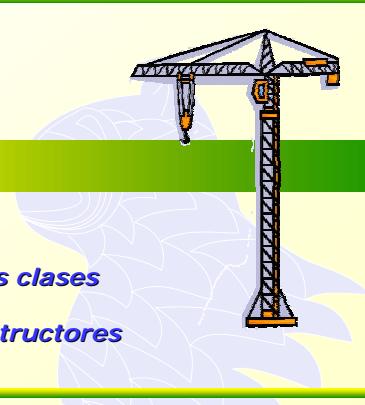
Polinomio::Polinomio(int n)
{
    if (n>0) {
        p = new float[n];
        for (int i=0; i<n; i++) p[i]=0;
        grado = 0;
        ncoef = n;
    } else
        PonerACero();
}
```

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

**Metodología de la Programación C++**

## Tema 6.- Constructores y destructores

- El constructor por defecto**
- Otros constructores**
- El destructor**
- El constructor de copias**
- Clases como atributos de otras clases**
- Llamadas explícitas a los constructores**



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

**El destructor C++**

### Liberando la memoria del polinomio

¿Qué ocurre con el siguiente código?

```
int main() {
    ...
    func();
}

void func() {
    Polinomio p1(20);
    ...
}
```

p1 es una variable local: se destruye al finalizar la función.

Diagram illustrating memory deallocation:

- A local variable **p1** is shown in a stack frame. It contains pointers **p**, **grado**, and **coef**, along with some padding (...). A red bracket below the frame indicates "Esta memoria se libera" (This memory is freed).
- A pointer **p** points to a dynamically allocated array of coefficients. This array is shown as a sequence of boxes, with the last one being deallocated. A red bracket below the array indicates "Esta memoria no se libera" (This memory is not freed).

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

**El destructor C++**

### Liberando la memoria del polinomio

Necesitamos un método que permita liberar la memoria:

```
class Polinomio {
private:
    ...
public:
    ...
    void Liberar(void);
};

void Polinomio::Liberar(void) {
    if (ncoef>0)
        delete [] p;
    ncoef = 0;
    p = 0;
    grado = 0;
}
```

void func() {
 Polinomio p1;
 ...
 p1.LiberarMemoria()
}

*El método debe llamarse antes de que se destruya el objeto.*

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

**El destructor C++**

### Liberando la memoria del polinomio: el destructor

Podemos automatizar el proceso de destrucción implementando un método especial denominado **destructor**.

- El destructor es **único** y no lleva parámetros ni devuelve nada.
- Se ejecuta, de forma automática, en el momento de destruir cada objeto de la clase.

```

class Polinomio {
    private:
        ...
    public:
        ...
        ~Polinomio();
    };
    ...
    Polinomio::~Polinomio()
    {
        if (ncoef>0)
            delete [] p;
        // El resto no es necesario
    }

```

void func() {  
 Polinomio p1(20);  
 ...  
}

*El método se ejecuta automáticamente justo antes de destruir el objeto p1.*

Metodología de la Programación · Javier Martínez Baena/Joaquín Fdez-Valdivia · Dpto. Ciencias de la Computación e I.A. (Univ. Granada) [Navigation icons] 15

**El destructor C++**

### Ejemplo (ejemplo\_condes.cpp)

¿Qué resultado produce el siguiente código?

```

class Prueba {
public:
    Prueba();
    ~Prueba();
};

Prueba::Prueba() {
    cout << "Constructor" << endl;
}

Prueba::~Prueba() {
    cout << "Destructor" << endl;
}

```

```

void funcion() {
    Prueba local;
    cout << "Funcion" << endl;
}

Prueba varglobal;

int main() {
    Prueba ppal;
    cout << "main 1" << endl;
    funcion();
    cout << "main 2" << endl;
}

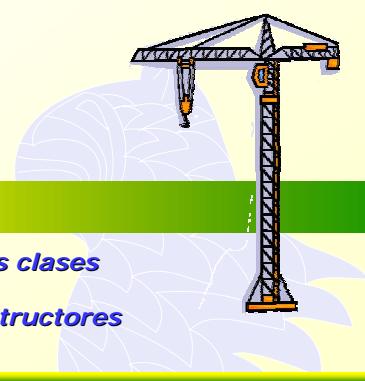
```

Metodología de la Programación · Javier Martínez Baena/Joaquín Fdez-Valdivia · Dpto. Ciencias de la Computación e I.A. (Univ. Granada) [Navigation icons] 16

 Metodología de la Programación C++

## Tema 6.- Constructores y destructores

- El constructor por defecto*
- Otros constructores*
- El destructor*
- El constructor de copias*
- Clases como atributos de otras clases*
- Llamadas explícitas a los constructores*



Metodología de la Programación - Javier Martínez Baena/Joaquín Fdez-Valdivia - Dpto. Ciencias de la Computación e I.A. (Univ. Granada) [navigation icons] 17

 El constructor de copias C++  
El constructor de copia por defecto

Construir una función (*externa a la clase*) que sume dos polinomios

```
void Sumar(const Polinomio p1, const Polinomio p2,
           Polinomio &res)
{
    // Determinamos el grado mayor
    int gmax = (p1.GetGrado()>p2.GetGrado()) ?
                p1.GetGrado() : p2.GetGrado();
    // Sumamos coeficientes
    for (int i=0; i<=gmax; ++i)
        res.SetCoef(i, p1.GetCoef(i) + p2.GetCoef(i));
}
```

¿Qué ocurre con la memoria (parámetros) en la siguiente llamada?

```
Polinomio a, b, r;
...
Sumar(a, b, r);
```

Se usa el **constructor de copia por defecto** (*copia bit a bit*).

Metodología de la Programación - Javier Martínez Baena/Joaquín Fdez-Valdivia - Dpto. Ciencias de la Computación e I.A. (Univ. Granada) [navigation icons] 18

 **El constructor de copias C++**

### ¿Podemos evitar la copia incorrecta?

Podemos evitar que se haga la copia usando un paso por referencia:

```
void Sumar(const Polinomio &p1, const Polinomio &p2,
           Polinomio &res)
{
    // Determinamos el grado mayor
    int gmax = (p1.GetGrado() > p2.GetGrado()) ?
                p1.GetGrado() : p2.GetGrado();
    // Sumamos coeficientes
    for (int i=0; i<=gmax; ++i)
        res.SetCoef(i, p1.GetCoef(i) + p2.GetCoef(i));
}
```

... aunque **lo correcto es** indicar como se haría una copia de forma adecuada mediante la definición de un **constructor de copia** propio para la clase.

Metodología de la Programación · Javier Martínez Baena/Joaquín Fdez-Valdivia · Dpto. Ciencias de la Computación e I.A. (Univ. Granada)  19

 **El constructor de copias C++**

### Creando un constructor de copia

Es posible crear un método (constructor de copia) que haga una copia correcta de un objeto de la clase en otro.

- Al ser un constructor, *se llama igual que la clase*.
- No devuelve nada y tiene como único parámetro, *constante y por referencia*, a un objeto de la clase.
- Lo que hace es copiar el objeto que se pasa como parámetro en el objeto que hace la llamada al constructor.
- Se ejecutará cuando se esté creando el objeto que hace la llamada.
- *Se llama automáticamente* al hacer un *paso por valor* para copiar el parámetro actual en el parámetro formal.

```
class Polinomio {
private:
    ...
public:
    ...
    Polinomio(const Polinomio &p);
};
```

¿Qué sentido tiene que pasemos el parámetro como **const** y por referencia?

Metodología de la Programación · Javier Martínez Baena/Joaquín Fdez-Valdivia · Dpto. Ciencias de la Computación e I.A. (Univ. Granada)  20

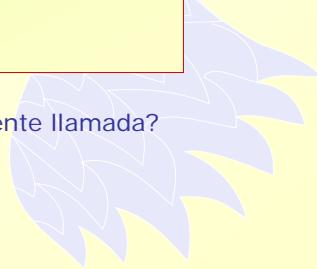
 **El constructor de copias C++**

**Ejemplo**

```
Polinomio::Polinomio(const Polinomio &p)
{
    if (p.ncoef>0) {
        // Usamos this para no confundir p
        this->p = new float [p.ncoef];
        this->grado = p.grado;
        this->ncoef = p.ncoef;
        for (int i=0; i<p.ncoef; i++)
            this->p[i] = p.p[i];
    } else
        PonerACero();
}
```

¿Qué ocurre con la memoria en la siguiente llamada?

```
Polinomio a, b, r;
...
Sumar(a, b, r);
```



Metodología de la Programación · Javier Martínez Baena/Joaquín Fdez-Valdivia · Dpto. Ciencias de la Computación e I.A. (Univ. Granada)  21

 **El constructor de copias C++**

**Ejemplo (ejemplo\_constcop.cpp)**

```
class Ejemplo { // Sin constructor de copia
private:
    int *p; // La clase usa memoria dinámica
    int z; // y también tiene un miembro estático
public:
    Ejemplo(); // Constructor por defecto
    ~Ejemplo(); // Destructor
    void get(int &p, int &z) { p=*(this->p); z=this->z; }
    void set(int p, int z) { *(this->p) = p; this->z=z; }
    void print() { cout << "*p=" << *p << "z=" << z << endl; }
};
Ejemplo::Ejemplo() {
    cout << " Constructor " << endl;
    p = new int; // Reservamos memoria
    *p = 2; // Iniciamos *p y z con el valor 2
    z = 2;
}
Ejemplo::~Ejemplo() {
    cout << " Destructor " << endl;
    delete p; // Liberamos memoria dinámica
}
```

Metodología de la Programación · Javier Martínez Baena/Joaquín Fdez-Valdivia · Dpto. Ciencias de la Computación e I.A. (Univ. Granada)  22

 **El constructor de copias C++**

**Ejemplo (ejemplo\_constcop.cpp)**

¿Qué hace el siguiente código?

```

int main() {
    cout << "Creamos a" << endl;
    Ejemplo a;
    cout << "Mostramos valores de a: ";
    a.print();
    cout << "Llamamos a la función func_param_ref()" << endl;
    func_param_ref(a);
    cout << "Mostramos valores de a: ";
    a.print();
    cout << "Llamamos a la función func_param_valor()" << endl;
    func_param_valor(a);
    cout << "Mostramos valores de a: ";
    a.print();
    cout << "Fin" << endl;
}

```

Metodología de la Programación · Javier Martínez Baena/Joaquín Fdez-Valdivia · Dpto. Ciencias de la Computación e I.A. (Univ. Granada) [Navigation icons] 23

 **El constructor de copias C++**

**Ejemplo (ejemplo\_constcop.cpp)**

¿Qué ocurre si añadimos el constructor de copia?

```

class Ejemplo {                                // Añadimos constructor de copia
    ...
public:
    ...
    Ejemplo(const Ejemplo &x);      // Constructor de copia
};

Ejemplo::Ejemplo(const Ejemplo &x) {
    cout << " Constructor de copia " << endl;
    p = new int;        // reservamos memoria para la copia
    *p = *(x.p);       // Copiamos valores de *p y z
    z = x.z;
}

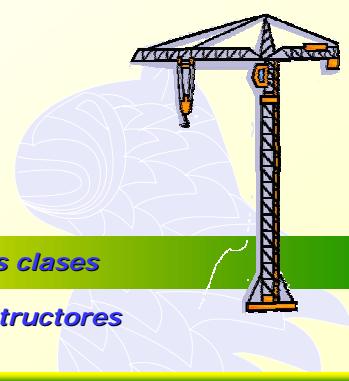
```

Metodología de la Programación · Javier Martínez Baena/Joaquín Fdez-Valdivia · Dpto. Ciencias de la Computación e I.A. (Univ. Granada) [Navigation icons] 24

*Metodología de la Programación C++*

## Tema 6.- Constructores y destructores

- El constructor por defecto*
- Otros constructores*
- El destructor*
- El constructor de copias*
- Clases como atributos de otras clases*
- Llamadas explícitas a los constructores*



Metodología de la Programación - Javier Martínez Baena/Joaquín Fdez-Valdivia - Dpto. Ciencias de la Computación e I.A. (Univ. Granada) [navigation icons] 25

*Clases como atributos de otras clases C++ Ejemplo*

Un miembro de una clase puede ser de cualquier clase

```

class Punto2D {
private:
    double x,y;
public:
    Punto2D() { x=y=0.0; }
    Punto2D(double x, double y) { Set(x,y); }
    ~Punto2D() { };
    void Set(double x, double y) { this->x=x; this->y=y; }
    void Get(double &x, double &y) const { x=this->x; y=this->y; }
};

class Linea2D {
private:
    Punto2D p1, p2; // p1 y p2 son de tipo Punto2D
public:
    Linea2D();
    ~Linea2D();
};

```

Metodología de la Programación - Javier Martínez Baena/Joaquín Fdez-Valdivia - Dpto. Ciencias de la Computación e I.A. (Univ. Granada) [navigation icons] 26

**Clases como atributos de otras clases C++ Ejemplo**



```

Linea2D::Linea2D()
{
    // <-- En este punto se crean p1 y p2
    p1.Set(-1,-1);    // una vez creados les asignamos valores
    p2.Set(1,1);      // (-1,-1) y (1,1) respectivamente
}

Linea2D::~Linea2D()
{
    // <-- En este punto se destruyen p1 y p2
}

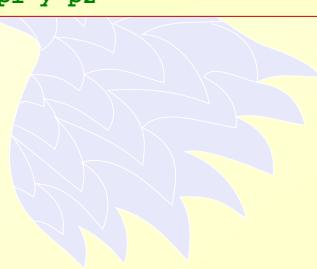
```

¿Qué ocurre con este código?

```

int main(int argc, char *argv[])
{
    Linea2D lin;
}

```



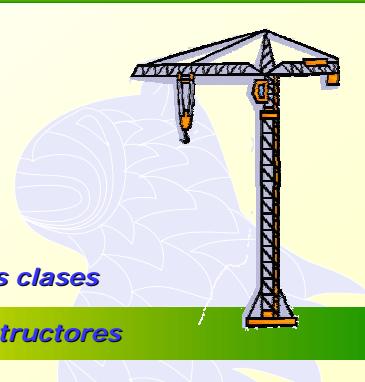
Metodología de la Programación - Javier Martínez Baena/Joaquín Fdez-Valdivia - Dpto. Ciencias de la Computación e I.A. (Univ. Granada) [Navigation icons] 27

**Metodología de la Programación C++**



## Tema 6.- Constructores y destructores

- El constructor por defecto*
- Otros constructores*
- El destructor*
- El constructor de copias*
- Clases como atributos de otras clases*
- Llamadas explícitas a los constructores*



Metodología de la Programación - Javier Martínez Baena/Joaquín Fdez-Valdivia - Dpto. Ciencias de la Computación e I.A. (Univ. Granada) [Navigation icons] 28

**Llamadas explícitas a los constructores C++**

**Otros usos de los constructores**



Podemos hacer llamadas explícitas a los métodos constructores:

```

Polinomio p; → Llamada al constructor de copia para crear
Polinomio z(p); el objeto z a partir del objeto p

Polinomio p, x; → Llamada al constructor por defecto y al
p = Polinomio(); constructor con parámetros
x = Polinomio(10);

```

**¿Es posible asignar dos objetos de una clase?**

Si, por defecto se hace la asignación elemento a elemento.  
*Más adelante veremos como hacerlo correctamente.*

Metodología de la Programación - Javier Martínez Baena/Joaquín Fdez-Valdivia - Dpto. Ciencias de la Computación e I.A. (Univ. Granada) [Navigation icons] 29

**Llamadas explícitas a los constructores C++**

**Constructores como iniciadores**



```

class Punto2D {
    private:
        double x,y;
    public:
        Punto2D() { x=y=0.0; }
        Punto2D(double x, double y) { Set(x,y); }
        ~Punto2D()
        void Set(double x, double y) { this->x=x; this->y=y; }
        double GetX() const { return x; }
        double GetY() const { return y; }
};

class Linea2D { // Le añadimos un nuevo constructor
    private:
        Punto2D p1, p2;
    public:
        Linea2D();
        Linea2D(const Punto2D &p1, const Punto2D &p2);
        ~Linea2D();
};

```

Metodología de la Programación - Javier Martínez Baena/Joaquín Fdez-Valdivia - Dpto. Ciencias de la Computación e I.A. (Univ. Granada) [Navigation icons] 30



## Llamadas explícitas a los constructores C++ Constructores como iniciadores

La implementación del nuevo constructor podría ser:

```
Linea2D::Linea2D(const Punto2D &pun1, const Punto2D &pun2)
{
    // Aquí se crean p1 y p2
    // A continuación se les da el valor inicial
    p1.Set(pun1.GetX(),pun1.GetY());    // Se inicia p1
    p2.Set(pun2.GetX(),pun2.GetY());    // Se inicia p2
}
```

Alternativamente podemos usar una lista de iniciadores para crear y asignar un valor inicial a p1 y p2:

```
Linea2D::Linea2D(const Punto2D &pun1, const Punto2D &pun2)
    : p1(pun1), p2(pun2)
// p1 y p2 se han creado llamando directamente al
// constructor deseado
{
```

¿Qué ventajas o inconvenientes tiene esta alternativa?