



Metodología de la Programación C++

Tema 3.- Sobrecarga de operadores

- El operador de asignación**
- La clase mínima**
- Operadores aritméticos**
- Operadores << y >>**
- Operador []**



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] 1



El operador de asignación C++
Ejemplo

Dados dos objetos de tipo Polinomio p1 y p2 ...
... ¿Es posible ejecutar una asignación del tipo p1=p2?

La respuesta es que **sí**: se hace una asignación, uno a uno, entre los miembros de p2 y los de p1.

*Surge un problema similar al que había con el paso por valor.
Además, en el constructor de copia se está creando el objeto destino mientras que en la asignación ya existe el objeto destino.*

La solución es construir un nuevo método para realizar la asignación correctamente.

```
class Polinomio {
    private:
        ...
    public:
        ...
        void Asignar(const Polinomio & p1);
};
```

por eficiencia

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

El operador de asignación C++

Ejemplo



```

void Polinomio::Asignar(const Polinomio &p)
{
    if (ncoef>0) delete [] this->p; // Si tiene memo. liberamos
    if (p.ncoef>0) {
        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();
}

```

Coincide con el código del constructor de copia

Polinomio p1, p2;
...
p1.Asignar(p2);

¿Podemos hacer que la sintaxis de la asignación sea la misma que en el resto de asignaciones?

p1=p2

Sí, sobreponiendo el operador de asignación.

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

El operador de asignación C++

operator=



```

TIPO & operator= (TIPO &dest, const TIPO &orig)
{
    // Copia de orig en dest ...
}
...
TIPO obj1, obj2;
obj1 = obj2;

```

Si el operador es miembro de una clase, el primer parámetro se omite y se considera que el objeto a la izquierda del = es el objeto que hace la llamada:

```

CLASE & CLASE::operator= (const CLASE &orig)
{
    // Copia de orig en el objeto que hace la llamada (*this)...
}
...
CLASE obj1, obj2;
obj1 = obj2;

```

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

**El operador de asignación C++
operator=**



```
Polinomio & Polinomio::operator=(const Polinomio &p)
{
    if (ncoef>0) delete [] this->p; // Si tiene memo. liberamos
    if (p.ncoef>0) {
        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();
}
```

```
Polinomio p1, p2;
...
p1 = p2;           → Equivalentes
p1.operator=(p2);
```



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

**El operador de asignación C++
operator=**



¿Qué devuelve el operator=?

Una asignación devuelve una referencia al valor que ha sido asignado

```
int x, y, z;
x = y = z = 3;
```

En nuestro caso devolveremos una referencia del objeto que hemos asignado, es decir: ***this**

```
Polinomio & Polinomio::operator=(const Polinomio &p)
{
    if (ncoef>0) delete [] this->p; // Si tiene memo. liberamos
    if (p.ncoef>0) {
        ...
        return *this;
    }
}
```

Esto permite hacer asignaciones en cadena de polinomios.

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

 **El operador de asignación C++
operator=**

¿Qué ocurre en esta situación?

```
Polinomio p1;
... // Asignamos algún valor a p1
p1 = p1;
```

Solución:

```
Polinomio & Polinomio::operator=(const Polinomio &p)
{
    if (&p!=this) {
        if (ncoef>0) delete [] this->p;
        if (p.ncoef>0) {
            ...
        }
        return *this;
}
```

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

 **El operador de asignación C++
operator=**

Esquema genérico para construir la sobrecarga de operator= en una clase que tiene memoria dinámica:

```
CLASE & CLASE::operator=(const CLASE &p)
{
    if (&p!=this) { // Si no es el mismo objeto
        // Si *this tiene memoria dinámica -> liberarla
        // Copiar p en *this (reservar nueva memoria y copiar)
    }
    return *this; // Devolver referencia a *this
}
```

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

El operador de asignación C++

Ejemplo

```

class Polinomio {
    ...
public:
    ...
    Polinomio & operator=(const Polinomio &p);
};

Polinomio & Polinomio::operator=(const Polinomio &p) {
    if (&p!=this) {
        if (ncoef>0) delete [] this->p; // Si tiene memo. liberamos
        if (p.ncoef>0) {
            this->p = new float [p.ncoef]; // Nueva memoria
            this->grado = p.grado;
            this->ncoef = p.ncoef;
            for (int i=0; i<p.ncoef; i++)
                this->p[i] = p.p[i];
        } else
            PonerACero();
    }
    return *this;
}

```

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

El operador de asignación C++

Ejemplo (ejemplo_asignacion.cpp)

```

class Ejemplo {           // No hay sobrecarga de operator=
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 (un entero)
    *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) [Navigation icons] 10

El operador de asignación C++

Ejemplo (ejemplo_asignacion.cpp)

```

int main(int argc, char *argv[]) {
    cout << "Creamos a" << endl;
    Ejemplo a;
    cout << "Modificamos a" << endl;
    a.set(4,5);
    cout << "Mostramos a: ";
    a.print();
    cout << "Abrimos bloque" << endl;
    {

        Ejemplo b;
        cout << "Mostramos b: ";
        b.print();
        cout << "Asignamos b=a";
        b=a;
        cout << "Mostramos a: ";
        a.print();
    }
    cout << "Mostramos b: ";
    b.print();
    cout << "Cerramos bloque" << endl;
}
cout << "Mostramos a: ";
a.print();
cout << "Fin" << endl;
}

```

¿Qué resultado produce el siguiente código?

El operador de asignación C++

Ejemplo (ejemplo_asignacion.cpp)

¿Y si añadimos la sobrecarga de operator=?

```

class Ejemplo {
    ...
public:
    ...
    Ejemplo & operator=(const Ejemplo &x);
};

Ejemplo& Ejemplo::operator=(const Ejemplo &x)
{
    cout << " Operador = " << endl;
    *p = *(x.p);           // Copiamos *p y z
    z = x.z;
    return *this;          // Devolvemos la referencia a *this
}

```

¿Por qué no es necesario comprobar si `this!=&x`?
 ¿Por qué no es necesario liberar la memoria de `*this`?

El operador de asignación C++

Ejemplo (ejemplo_return.cpp)



```

class Ejemplo {
private:
    int *p;
    int z;
public:
    Ejemplo();                                // Constructor por defecto
    Ejemplo(const Ejemplo &x);                // Constructor de copia
    ~Ejemplo();                               // Destructor
    Ejemplo & operator=(const Ejemplo &x);
    void get(int &p, int &z);
    void set(int p, int z) {
        void print() { cout << " ";
    }
};

Ejemplo func_devuelve()
{
    Ejemplo x;
    cout << "Funcion" << endl;
    x.set(5,6);
    return x;
}

int main()
{
    cout << "Comenzamos" << endl;
    Ejemplo a;
    cout << "Mostramos a: ";
    a.print();
    cout << "Llamada func" << endl;
    a = func_devuelve();
    cout << "Mostramos a: ";
    a.print();
    cout << "Fin" << endl;
}

```

¿Qué ocurre al hacer return de un objeto de la clase Ejemplo?

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

Metodología de la Programación C++



Tema 7.- Sobrecarga de operadores

- El operador de asignación**
- La clase mínima**
- Operadores aritméticos**
- Operadores << y >>**
- Operador []**




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] 14

 **La clase mínima C++**

¿Qué debería tener una clase mínima?

Al construir una clase, debemos tener en cuenta las siguientes normas:

- La representación de los datos *suele* ser **private**.
- La interfaz *suele* ser **public**.
- Normalmente, construiremos un **constructor por defecto**.
- Debemos construir el **destructor**.
- Debemos construir el **constructor de copia**.
- Debemos sobrecargar el **operador de asignación**.

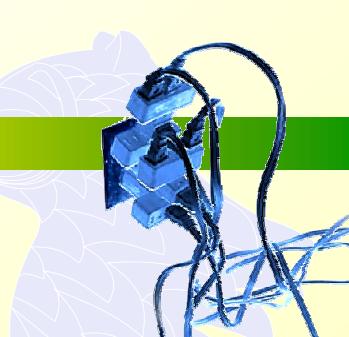
Obligatorios en caso de usar memoria dinámica.
Si no los definimos el compilador añade una versión por defecto.

Metodología de la Programación - Javier Martínez Baena/Joaquín Fdez-Valdivia - Dpto. Ciencias de la Computación e I.A. (Univ. Granada) |◀|◀|▶|▶|◀|▶| 15

 **Metodología de la Programación C++**

Tema 7.- Sobrecarga de operadores

- El operador de asignación*
- La clase mínima*
- Operadores aritméticos*
- Operadores << y >>*
- Operador []*



 **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) |◀|◀|▶|▶|◀|▶| 16

Operadores aritméticos C++

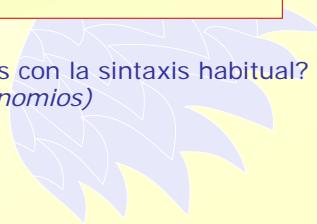
Ejemplo

Construyamos una función para sumar dos polinomios:

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

¿Podríamos hacer algo para sumar polinomios con la sintaxis habitual?
 $p1 = p2 + p3;$ (siendo $p1$, $p2$ y $p3$ polinomios)

Sí, sobrecargando el operador +



Metodología de la Programación - Javier Martínez Baena/Joaquín Fdez-Valdivia - Dpto. Ciencias de la Computación e I.A. (Univ. Granada) | Back | Forward | Home | Help | 17

Operadores aritméticos C++

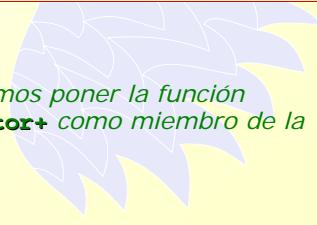
operator+

Sobrecarga del operador +:

```
// Sumamos dos polinomios
Polinomio operator+ (const Polinomio &p1, const Polinomio &p2)
{
    Polinomio res;
    int gmax = (p1.GetGrado()>p2.GetGrado()) ?
               p1.GetGrado() : p2.GetGrado();
    for (int i=0; i<=gmax; ++i)
        res.SetCoef(i, p1.GetCoef(i) + p2.GetCoef(i));
    return res;
}
```

```
Polinomio p1, p2, p3;
... // Damos valor a p2 y p3
p1 = p2 + p3;
p1 = operator+(p2,p3);
```

¿Podemos poner la función **operator+** como miembro de la clase?



Metodología de la Programación - Javier Martínez Baena/Joaquín Fdez-Valdivia - Dpto. Ciencias de la Computación e I.A. (Univ. Granada) | Back | Forward | Home | Help | 18

Operadores aritméticos C++ operator+



Sobrecarga del operador + como miembro de la clase:

```
// Sumamos dos polinomios
Polinomio Polinomio::operator+ (const Polinomio &p2) const
{
    Polinomio res;
    int gmax = (GetGrado()>p2.GetGrado()) ?
                GetGrado() : p2.GetGrado();
    for (int i=0; i<=gmax; ++i)
        res.SetCoef(i, GetCoef(i) + p2.GetCoef(i));
    return res;
}
```

```
Polinomio p1, p2, p3;
... // Damos valor a p2 y p3

p1 = p2 + p3;

p1 = p2.operator+(p3);
```



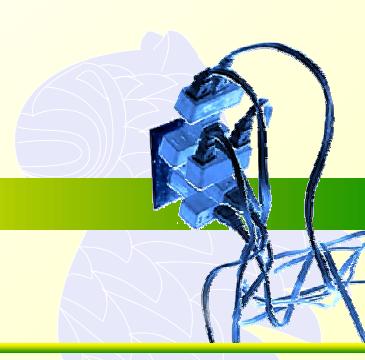
Metodología de la Programación - Javier Martínez Baena/Joaquín Fdez-Valdivia - Dpto. Ciencias de la Computación e I.A. (Univ. Granada) [navigation icons] 19

Metodología de la Programación C++



Tema 7.- Sobrecarga de operadores

- El operador de asignación*
- La clase mínima*
- Operadores aritméticos*
- Operadores << y >>*
- Operador []*




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] 20

Operadores << y >> C++

Sobrecarga del operador << (salida de datos):

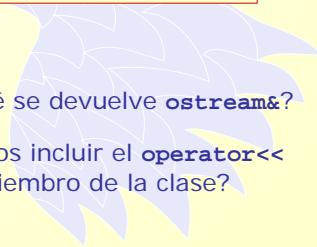
```
// Mostramos un polinomio en pantalla
ostream & operator<< (ostream &flujo, const Polinomio &p)
{
    flujo << p.GetCoef(0) ;
    for (int i=1; i<=p.GetGrado(); ++i)
        flujo << " + " << p.GetCoef(i) << "*x^" << i;
    flujo << endl;
    return flujo;
}
```

```
Polinomio p1, p2;
... // Damos valor a p1 y p2

cout << p1;
cout << p1 << p2;
```

¿Por qué se devuelve ostream&?

¿Podemos incluir el operator<< como miembro de la clase?



Operadores << y >> C++

Sobrecarga del operador >> (entrada de datos):

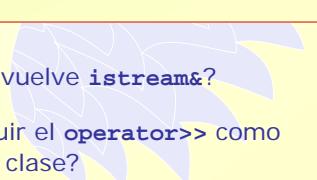
```
// Leemos un polinomio por teclado
istream & operator>> (istream &flujo, Polinomio &p) {
    int g=0;
    float v=0;
    while (g>=0) {
        // Pedimos los coeficientes de la forma grado y valor
        // Si grado es negativo acabamos la introducción de datos
        flujo >> g >> v;
        p.SetCoef(g,v);
    }
    return flujo;
}
```

```
Polinomio p1, p2;

cin >> p1;
cin >> p1 >> p2;
```

¿Por qué se devuelve istream&?

¿Podemos incluir el operator>> como miembro de la clase?



 Operadores << y >> C++
Ejemplo

Sobrecarga del operador << (salida de datos):

```
class Polinomio {
    ...
public:
    ...
    friend ostream & operator<<(ostream &flujo,
                                    const Polinomio &p);
};

// Mostramos un polinomio en pantalla
ostream & operator<< (ostream &flujo, const Polinomio &p)
{
    flujo << p.p[0];
    for (int i=1; i<=p.grado; ++i)
        flujo << " + " << p.p[i] << "*x^" << i;
    flujo << endl;
    return flujo;
}
```

La función operator<< NO pertenece a la clase, sólo es amiga

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

 Metodología de la Programación C++

Tema 7.- Sobrecarga de operadores

- El operador de asignación*
- La clase mínima*
- Operadores aritméticos*
- Operadores << y >>*
- Operador []*



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

Operador [] C++ operator[]

¿Es posible cambiar la sintaxis ... `x = p.GetCoef(i);`
... por esta otra? `x = p[i];`

```
float Polinomio::operator[](int i)
{
    assert(i>=0);
    assert(i<=grado);
    return p[i];
};
```

¿Es posible cambiar la sintaxis ... `p.SetCoef(i,x);`
... por esta otra? `p[i] = x;`

```
float & Polinomio::operator[](int i)
{
    assert(i>=0);
    assert(i<=grado);
    return p[i];
};
```

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

Operador [] C++ operator[]

¿Qué ocurre con la siguiente llamada?

```
void funcion(const Polinomio p)
{
    ...
    x = p[i];
    ...
};
```

Sobrecargamos de nuevo el operador []

```
const float & Polinomio::operator[](int i) const
{
    assert(i>=0);
    assert(i<=grado);
    return p[i];
};
```

Imaginemos que tenemos una clase matriz (2D) ... ¿podríamos usar la sintaxis del doble corchete [][]?

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