INFORMÁTICA INDUSTRIAL

Sobrecarta y operadores en C++.

M. Abderrahim, A. Castro, J. C. Castillo
Departamento de Ingeniería de Sistemas y Automática

uc3m | Universidad Carlos III de Madrid

AGENDA

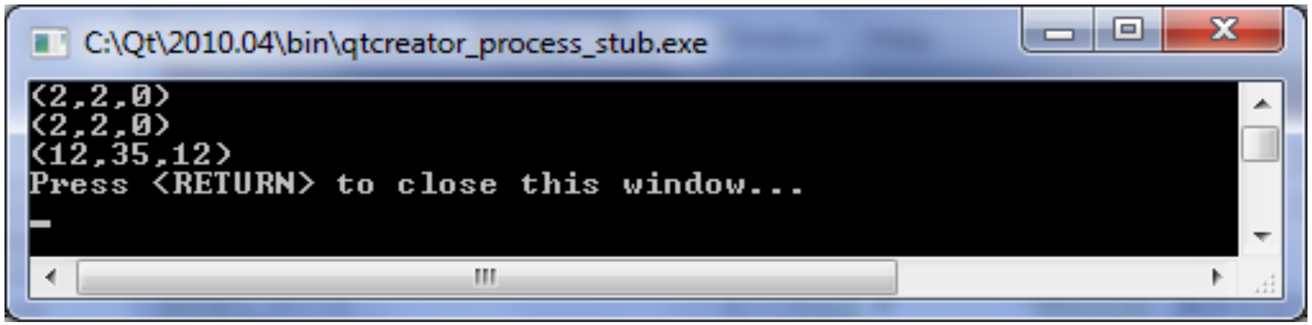
- Metodos Sobrecargados
- Operadores Sobrecargados

Métodos Sobrecargados

- En C++ es posible escribir varias funciones que teniendo el mismo nombre **se diferencia por el tipo de parámetros utilizados**.
- Lo mismo podemos hacer con los métodos de una clase.
- Como se muestra en el siguiente ejemplo:

Ejemplo: Métodos Sobrecargados

```cpp
using namespace std;
class punto{
public:
    punto(float xx=0, float yy=0, float zz=0): x(xx), y(yy), z(zz) {}
    void Asignar(float xi, float yi, float zi){
         x = xi;
         y = yi;
         z = zi;
    }
    void Asignar(punto p){
         Asignar(p.x, p.y, p.z);
    }
    void Ver(){
         cout << "(" << x << "," << y << "," << z << ")" << endl;
    }
private: float x, y, z;
};
int main(void){
    punto Z, P = punto(2,2);
    P.Ver();
    Z.Asignar(P);   Z.Ver();
    Z.Asignar(12,35,12);
    Z.Ver();
}
```



Operadores Sobrecargados

- En C++ los operadores pasan a ser funciones como otra cualquiera, pero que permiten para su ejecución una notación especial.
- Todos tienen asignada una función por defecto.
- En C++ el programador puede sobrecargar casi todos los operadores para adaptarlos a su propio uso.
- Para ello se dispone de una sintaxis específica que permite definirlos o declararlos.

```
Prototipo para los operadores:
<tipo> operator <el operador>
(<argumentos>);

Definición para los operadores:
<tipo> operator <el operador>
(<argumentos>)
{
    <sentencias>;
}
```

Operadores Sobrecargados

- Limitaciones
  - Se pueden sobrecargar todos los operadores excepto ".", "," , ".*" , "::" ,"?:".
  - Los operadores "=", "[ ]", "->", "new" y "delete", sólo pueden ser sobrecargados cuando se definen como miembros de una clase.
  - Los argumentos para los operadores externos deben ser tipos enumerados o estructurados: struct, union o class.

Ejemplo de sobrecarga de operadores

```cpp
#include <iostream>
using namespace std;
typedef struct _complex{
    float real;
    float imag;
}complex;

complex operator +(complex x,complex y)
{
    complex z;
    z.real = x.real+y.real;
    z.imag = x.imag+y.imag;
    return z;
}
int main(){
    complex a={5.0F,3.0F},b={3.0F,1.2F},c;
    c=a+b;
    cout<<c.real<<"+"<<c.imag<<"i"<<endl;
}
```

```
WJRG-MacBookPro:Ejemplos wladimir$ g++ complex.cpp -o complex
WJRG-MacBookPro:Ejemplos wladimir$ ./complex
8+4.2i
WJRG-MacBookPro:Ejemplos wladimir$
```

Sobrecarga de Operadores Binarios

- Los operadores binarios son aquellos que requieren de dos operandos para calcular el valor de la operación.

- Cuando un operador binario es miembro de una clase, se asume que el primer operando es el propio objeto de la clase donde se define el operador.

- Cuando dentro de una clase se sobrecargan operadores binarios sólo será necesario especificar un operando (el segundo), el otro (el primero) es el propio objeto.

Sobrecarga de Operadores Binarios

- En la declaración de la clase se incluirá una línea con la siguiente sintaxis.

```
<tipo> operator <operador binario>(<tipo> <identificador>);
```

- Habitualmente el tipo será el mismo de la clase, pero podría ser diferente.

Ejemplo: Operador Binarios

```cpp
#include <iostream>
using namespace std;
class Tiempo {
public:
   Tiempo(int h=0, int m=0) : hora(h), minuto(m) {}
   void Mostrar(){
       cout << hora << ":" << minuto << endl;};
   Tiempo operator+(Tiempo h);
private:
   int hora; int minuto;
};
Tiempo Tiempo::operator+(Tiempo h)
{
   Tiempo temp;
   temp.minuto = minuto + h.minuto;
   temp.hora = hora + h.hora;
   if(temp.minuto >= 60)
   {
       temp.minuto -= 60;
       temp.hora++;
   }
   return temp;
}
```

```cpp
int main(void)
{
    Tiempo Ahora(12,24), T1(4,45);
    T1 = Ahora + T1;
    T1.Mostrar();
    (Ahora + Tiempo(4,45)).Mostrar();
}
```

```
WJRG-MacBookPro:Ejemplos wladimir$ g++ Tiempo.cpp -o tiempo
WJRG-MacBookPro:Ejemplos wladimir$ ./tiempo
17:9
17:9
WJRG-MacBookPro:Ejemplos wladimir$
```

Sobrecarga de Operadores Binarios

- En el próximo ejemplo agregaremos otro operador de suma a nuestra clase Tiempo

- Pero en vez de pasarle un objeto tiempo le vamos a pasar un valor entero que representa los minutos que le queremos sumar a los objetos de la clase Tiempo.

Ejemplo: Operador Binarios

```
#include <iostream>
using namespace std;
class Tiempo {
public:
   Tiempo(int h=0, int m=0) : hora(h), minuto(m) {}
   void Mostrar(){
       cout << hora << ":" << minuto << endl;};
   Tiempo operator+(Tiempo h);
       Tiempo operator+(int mins);
private:
   int hora; int minuto;
};
Tiempo Tiempo::operator+(Tiempo h){
   Tiempo temp;
   temp.minuto = minuto + h.minuto;
   temp.hora = hora + h.hora;
   if(temp.minuto >= 60)
   {
       temp.minuto -= 60;
       temp.hora++;
   }
   return temp;
}
```

```
Tiempo Tiempo::operator +(int mins) {
    Tiempo temp;
    temp.minuto=minuto+mins;
    temp.hora=hora+temp.minuto/60;
    temp.minuto=temp.minuto%60;
    return temp;
}
```

```
int main(void)
{
    Tiempo Ahora(12,24), T1(4,45);
    T1 = Ahora + T1;
    T1.Mostrar();
    (Ahora + 45).Mostrar();
}
```

```
WJRG-MacBookPro:Ejemplos wladimir$ g++ Tiempo2.cpp -o tiempo2
WJRG-MacBookPro:Ejemplos wladimir$ ./tiempo2
17:9
13:9
WJRG-MacBookPro:Ejemplos wladimir$
```

Sobrecarga del Operador =

- Es uno de los operadores que es común sobrecargar.

- Si no se sobrecarga se usaría el operador por defecto que hace una copia exacta del objeto.

- Esto es **válido siempre y cuando** no se este utilizando memoria dinámica.

- En el ejemplo a continuación se muestra los problemas que puede causar el operador por defecto, cuando se usa con clases donde exista asignación dinámica.

Ejemplo: Sobrecarga del Operador =

```cpp
#include <iostream>
#include <cstring>
using namespace std;
class Cadena
{
public:
    Cadena(const char *cad);
    Cadena() { cadena=NULL;} ;
    ~Cadena() { delete[] cadena; };
    void Mostrar(){cout << cadena << endl;};
    void RellenarDeGuiones();
private:
    char *cadena;
};
```

```cpp
Cadena::Cadena(const char *cad)
{
    cadena = new char[strlen(cad)+1];
    strcpy(cadena, cad);
}
```

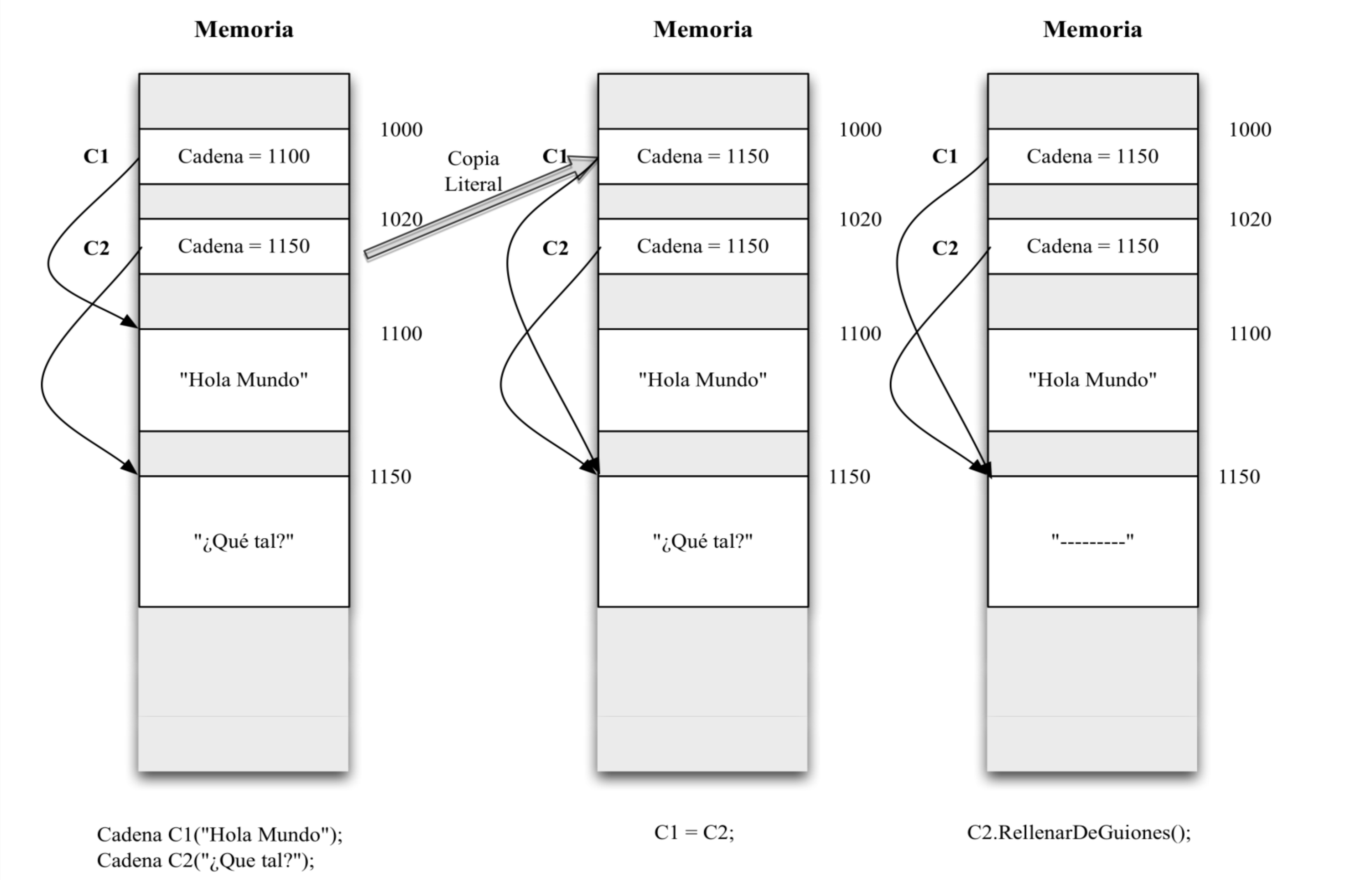
```cpp
void Cadena::RellenarDeGuiones(){
    for(int i=0;i<strlen(cadena);i++)
    cadena[i]='-';
}
```

```cpp
int main(void)
{
    Cadena C1("Hola Mundo"), C2("¿Qué tal?");
    C1.Mostrar();
    C2.Mostrar();
    C1=C2;
    C2.RellenarDeGuiones();
    C1.Mostrar();
    C2.Mostrar();
}
```

```
WJRG-MacBookPro:Ejemplos wladimir$ g++ Cadena.cpp
WJRG-MacBookPro:Ejemplos wladimir$ ./a.out
Hola Mundo
¿Qué tal?
------------
------------
a.out(12460) malloc: *** error for object 0x100100090:
pointer being freed was not allocated
*** set a breakpoint in malloc_error_break to debug
Abort trap
WJRG-MacBookPro:Ejemplos wladimir$
```

Ejemplo: Sobrecarga del Operador =

Esto es lo que pasa en la memoria!!



Ejemplo: Sobrecarga del Operador =

- Solución, sobrecargar el operador =

```
Cadena & Cadena::operator=(const Cadena &c)
{
      if (this != &c){
            delete [] cadena;  // del objeto donde copia si ….
            if (c.cadena){
                    cadena = new char[strlen(c.cadena)+1];
            strcpy(cadena, c.cadena);
             }else
                    cadena = NULL;
        }
        return *this;
}
```

```
WJRG-MacBookPro:Ejemplos wladimir$ g++ Cadena2.cpp
WJRG-MacBookPro:Ejemplos wladimir$ ./a.out
Hola Mundo
¿Qué tal?
¿Qué tal?
------------
WJRG-MacBookPro:Ejemplos wladimir$
```

Sobrecarga Operadores Unitarios

- Los operadores unitarios son aquellos que sólo requieren un operando, como el incremento.
- Cuando se sobrecargan operadores unitarios en una clase el operando es el propio objeto de la clase donde se define el operador. Por lo tanto los operadores unitarios dentro de las clases no requieren parámetros. Sintaxis:

```
<tipo> operator <el operador unitario>();
```

- Normalmente el <tipo> es la clase para la que estamos sobrecargando el operador.

Ej. sobrecarga operador unitario ++

```
class Tiempo {
    ...
    Tiempo operator++(); //forma prefija
    ...
};
Tiempo Tiempo::operator++() {
    minuto++;
    while(minuto >= 60) {
        minuto -= 60;
        hora++;
    }
    return *this;
}
...
Tiempo T1(4,45);
T1.Mostrar();
++T1;
T1.Mostrar();
...
```

```
WJRG-MacBookPro:Ejemplos wladimir$ g++ Tiempo3.cpp
WJRG-MacBookPro:Ejemplos wladimir$ ./a.out
4:45
4:46
WJRG-MacBookPro:Ejemplos wladimir$
```

Ej. Sobrecarga operadores unitarios ++ (prefijo) y ++ (sufijo)

```cpp
class Tiempo {
    ...
    Tiempo operator++(); //Forma prefija
    Tiempo operator++(int); //Forma sufija
    ...
};
Tiempo Tiempo::operator++() {
    minuto++;
    while(minuto >= 60) {
        minuto -= 60;
        hora++;
    }
    return *this;
}
Tiempo Tiempo::operator++(int) {
    Tiempo temp(*this);//guardamos el valor inicial copiando el valor
    minuto++;
    while(minuto >= 60) {
        minuto -= 60;
        hora++;
    }
    return temp;
}
```

```cpp
int main(void)
{
    Tiempo T1(4,45);
    T1.Mostrar();
    (T1++).Mostrar();
    T1.Mostrar();
    (++T1).Mostrar();
    T1.Mostrar();
    Return 0
}
```

```
WJRG-MacBookPro:Ejemplos wladimir$ g++ Tiempo4.cpp
WJRG-MacBookPro:Ejemplos wladimir$ ./a.out
4:45
4:45
4:46
4:47
4:47
WJRG-MacBookPro:Ejemplos wladimir$
```

uc3m | Universidad Carlos III de Madrid