

Ejemplo de clase

1. Ejemplo de clase : La clase Cuenta
2. Uso de la clase Cuenta
3. Métodos y objetos receptores de mensajes (Importante)

Ejemplo de clase

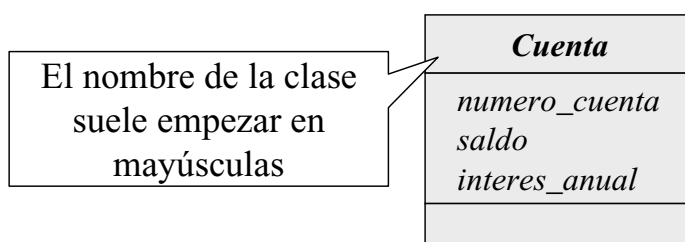
Una clase para cuentas de un banco

Vamos a modelar con una clase, un nuevo tipo de datos , donde los elementos de la clase, los objetos, son cuentas bancarias. El nombre de la clase que vamos a modelar es *Cuenta*.

- ✓ La información asociada a cada una de las cuentas:

Nº de cuenta, saldo de la cuenta, interés anual

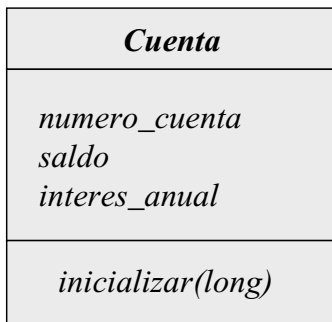
Establecemos un nombre y un tipo para cada uno de los atributos, y los colocamos en la parte privada.



```
#include <iostream.h>
class Cuenta
{
private:
    long int numero_cuenta;
    float saldo;
    float interes_anual;
};
```

Ejemplo de clase

Ahora tenemos que pensar en las operaciones necesarias para la gestión de cuentas bancarias. Por ejemplo, necesitaremos un método que se encargue de la inicialización de los objetos de la clase *Cuenta*.



El atributo *numero_cuenta* se actualiza en éste método, y nunca se puede cambiar

```
#include <iostream.h>
class Cuenta
{
private:
    long int numero_cuenta;
    float saldo;
    float interes_anual;
public:
    void inicializar( long int num);
};

void Cuenta:: inicializar( long int num)
{
    numero_cuenta = num;
    saldo = 0;
    interes_anual = 0;
}
```

Ejemplo de clase

¿Se nos ocurren más operaciones?

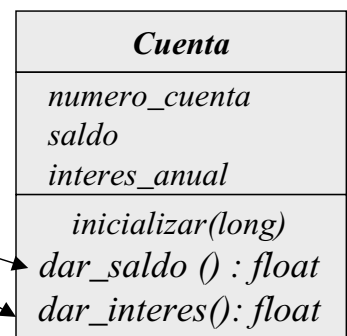
Puede ser útil diseñar métodos que devuelvan el contenido de los atributos:

```
#include <iostream.h>
class Cuenta
{
private:
    long int numero_cuenta;
    float saldo;
    float interes_anual;
public:
    void inicializar( long int num);
    float dar_saldo();
    float dar_interes();
};
```

```
void Cuenta:: inicializar( long int num)
{
    numero_cuenta = num;
    saldo = 0;
    interes_anual = 0;
}

float Cuenta:: dar_saldo()
{ return saldo; }

float Cuenta:: dar_interes()
{ return interes_anual ; }
```



Ejemplo de clase

¿Se nos ocurren más operaciones?

```
#include <iostream.h>
class Cuenta
{
private:
    long int numero_cuenta;
    float saldo;
    float interes_anual;
public:
    void inicializar( long int num);
    float dar_saldo();
    float dar_interes();
    void mod_saldo (float s);
    void mod_interes( float i );
    void ingreso ( float cantidad);
    bool reintegro ( float r);
    void mostrar_datos ();
};
```

<i>Cuenta</i>
<i>numero_cuenta</i>
<i>saldo</i>
<i>interes_anual</i>
<i>inicializar(long)</i>
<i>dar_saldo () : float</i>
<i>dar_interes(): float</i>
<i>mod_saldo (float);</i>
<i>mod_interes(float);</i>
<i>ingreso (float);</i>
<i>reintegro (float): bool</i>
<i>mostrar_datos ();</i>

Para usar la clase, basta con conocer su nombre y la forma de los métodos

Ejemplo de clase

Más operaciones ...

```
void Cuenta :: mod_saldo (float s)
{ saldo = s ; }

void Cuenta :: mod_interes( float i )
{ interes_anual = i ; }

void Cuenta :: ingreso ( float cantidad)
{ saldo = saldo + cantidad ; }

bool Cuenta :: reintegro ( float r)
{
    if ( r > saldo ) return false;
    else
    { saldo = saldo - r;
      return true;
    }
}
```

Actualiza el saldo de la cuenta

Actualiza el interés anual

Devuelve *true* si hay saldo suficiente y resta la cantidad.
Devuelve *false* si no hay saldo suficiente

```
void Cuenta :: mostrar_datos ()
{
    cout << "Nº de cuenta : " << numero_cuenta << endl ;
    cout << "Saldo: " << saldo << endl ;
}
```

Ejemplo de clase

Uso de la clase **Cuenta**

```
#include <iostream.h>
class Cuenta
{
    ...
};
...
void main();
{
    Cuenta cc; // cc es un objeto de la clase cuenta
    cc.inicializar( 24316622 );
    cc.mod_saldo(10000);
    cc.mod_interes(2);
    cc.mostrar_datos( );
    cc.ingreso( 12000 );
    cc.mostrar_datos( );
    bool b;
    b = cc.reintegro(10000);
    if (b==false) cout << "No hay saldo";
    cc.mostrar_datos( );
}
```

Nº de cuenta: 24316622
Saldo: 10000

Nº de cuenta: 24316622
Saldo: 22000

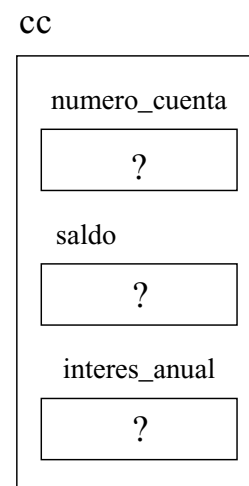
Nº de cuenta: 24316622
Saldo: 12000

Ejemplo de clase

¿Qué ocurre durante la ejecución?

```
#include <iostream.h>
class Cuenta
{
    ...
};
...
void main();
{
    Cuenta cc;
    cc.inicializar( 24316622 );
    cc.mod_saldo(10000);
    cc.mod_interes(2);
    cc.mostrar_datos( );
    cc.ingreso( 12000 );
    cc.mostrar_datos( );
    bool b;
    b = cc.reintegro(10000);
    if (b==false) cout << "No hay saldo";
    cc.mostrar_datos( );
}
```

Se crea un objeto llamado **cc** de la clase **Cuenta**



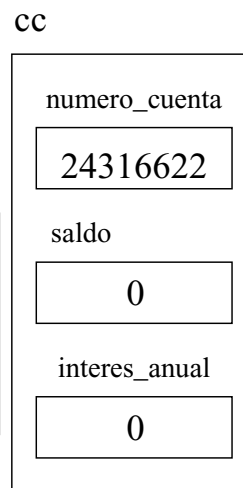
Ejemplo de clase

¿Qué ocurre durante la ejecución?

```
#include <iostream.h>
class Cuenta
{
    ...
};
...
void main();
{
    Cuenta cc;
    cc.inicializar( 24316622 );
    cc.mod_saldo(10000);
    cc.mod_interes(2);
    cc.mostrar_datos( );
    cc.ingreso( 12000 );
    cc.mostrar_datos( );
    bool b;
    b = cc.reintegro(10000);
    if (b==false) cout << "fallo" << endl;
    cc.mostrar_datos( );
}
```

Se envía el mensaje inicializar al objeto cc proporcionando el argumento 24316622

```
void Cuenta:: inicializar( long int num)
{
    numero_cuenta = num;
    saldo = 0;
    interes_anual = 0;
}
```



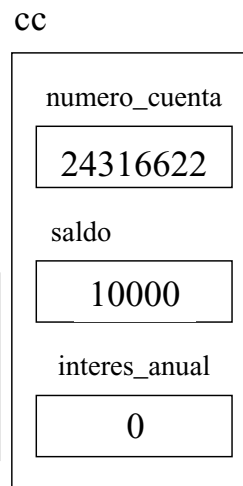
Ejemplo de clase

¿Qué ocurre durante la ejecución?

```
#include <iostream.h>
class Cuenta
{
    ...
};
...
void main();
{
    Cuenta cc;
    cc.inicializar( 24316622 );
    cc.mod_saldo(10000);
    cc.mod_interes(2);
    cc.mostrar_datos( );
    cc.ingreso( 12000 );
    cc.mostrar_datos( );
    bool b;
    b = cc.reintegro(10000);
    if (b==false) cout << "fallo" << endl;
    cc.mostrar_datos( );
}
```

Se envía el mensaje mod_saldo al objeto cc proporcionando el valor 10000 como argumento

```
void Cuenta :: mod_saldo (float s)
{ saldo = s ; }
```



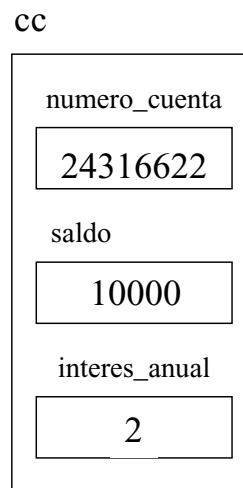
Ejemplo de clase

¿Qué ocurre durante la ejecución?

```
#include <iostream.h>
class Cuenta
{
    ...
};
...
void main();
{
    Cuenta cc;
    cc.inicializar( 24316622 );
    cc.mod_saldo(10000);
    cc.mod_interes(2);
    cc.mostrar_datos( );
    cc.ingreso( 12000 );
    cc.mostrar_datos( );
    bool b;
    b = cc.reintegro(1000);
    if (b==false) cout << "Error";
    cc.mostrar_datos( );
}
```

Se ejecuta el método `mod_interes` de la clase `Cuenta`, sobre el objeto `cc`, modificándose el valor del atributo `interes_anual`.

```
void Cuenta :: mod_interes( float i )
{ interes_anual = i ; }
```



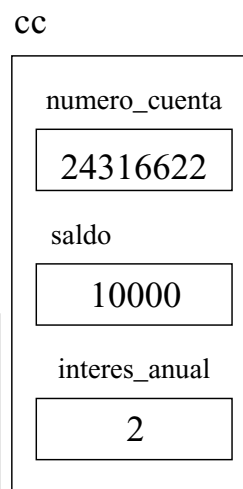
Ejemplo de clase

¿Qué ocurre durante la ejecución?

```
#include <iostream.h>
class Cuenta
{
    ...
};
...
void main();
{
    Cuenta cc;
    cc.inicializar( 24316622 );
    cc.mod_saldo(10000);
    cc.mod_interes(2);
    cc.mostrar_datos( );
    cc.ingreso( 12000 );
    cc.mostrar_datos( );
    bool b;
    b = cc.reintegro(1000);
    if (b==false) cout << "Error";
    cc.mostrar_datos( );
}
```

Se envía el mensaje `mostrar_datos` al objeto `cc`. Se ejecuta el método `mostrar_datos` sobre el objeto `cc`. Se visualizan por pantalla los valores de los atributos `numero_cuenta` y `saldo`.

```
void Cuenta :: mostrar_datos ( )
{
    cout << "Nº de cuenta : " << numero_cuenta;
    cout << "Saldo: " << saldo << endl ;
}
```



Ejemplo de clase

¿Qué ocurre durante la ejecución?

```
#include <iostream.h>
class Cuenta
{
    ...
};
...
void main();
{
    Cuenta cc;
    cc.inicializar( 24316622 ) ;
    cc.mod_saldo(10000);
    cc.mod_interes(2);
    cc.mostrar_datos( );
    cc.ingreso( 12000 ) ;
    cc.mostrar_datos( );
    bool b;
    b = cc.reintegro(10000);
    if (b==false) cout << "No hay saldo";
    cc.mostrar_datos( );
}
```

Se envía el mensaje ingreso al objeto cc proporcionando el valor 12000 como argumento. Se ejecuta el método ingreso de la clase Cuenta sobre el objeto cc, modificándose el valor del atributo saldo.

```
void Cuenta :: ingreso ( float cantidad)
{
    saldo = saldo + cantidad ;
}
```

cc

numero_cuenta

24316622

saldo

22000

interes_anual

2

Ejemplo de clase

¿Qué ocurre durante la ejecución?

```
#include <iostream.h>
class Cuenta
{
    ...
};
...
void main();
{
    Cuenta cc;
    cc.inicializar( 24316622 ) ;
    cc.mod_saldo(10000);
    cc.mod_interes(2);
    cc.mostrar_datos( );
    cc.ingreso( 12000 ) ;
    cc.mostrar_datos( );
    bool b;
    b = cc.reintegro(10000);
    if (b==false) cout << "No hay saldo";
    cc.mostrar_datos( );
}
```

Se envía el mensaje reintegro al objeto cc con el valor 10000 como argumento. Se ejecuta el método reintegro de la clase Cuenta sobre el objeto cc.

cc

numero_cuenta

24316622

saldo

22000

interes_anual

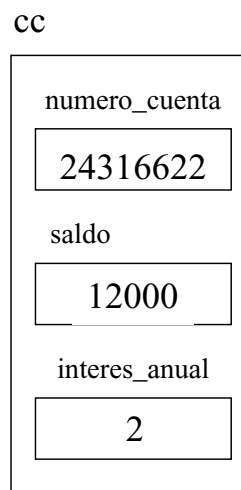
2

¿Qué ocurre durante la ejecución?

```
...  
...  
void main();  
{  
    Cuenta cc;  
    cc.inicializar( 24316622 );  
    cc.mod_saldo(10000);  
    cc.mod_interes(2);  
    cc.mostrar_datos( );  
    cc.ingreso( 12000 );  
    cc.mostrar_datos( );  
    bool b;  
    b = cc.reintegro(10000);  
    if (b==false) cout << "No h  
    cc.mostrar_datos( );  
}
```

Como la cantidad 10000 no es mayor que el atributo `saldo`, se le resta a ese atributo la cantidad y se devuelve `true` como resultado de éste método.

```
bool Cuenta :: reintegro ( float r )  
{  
    if ( r > saldo ) return false;  
    else  
        { saldo = saldo - r;  
          return true;  
        }  
}
```



Métodos y objetos receptores de mensajes

¡ Importante !

Ya sabemos que un método se ejecuta cuando se pasa el correspondiente mensaje al objeto.

El código del método se ejecuta sobre el objeto que ha recibido el mensaje:



Los atributos que se utilizan en el método, se refieren a los atributos del objeto receptor del mensaje.


```
void Cuenta :: ingreso ( float cantidad )  
{  
    saldo = saldo + cantidad ;  
}
```

```
...  
cc.saldo(10000);  
...
```

Se refiere al atributo `saldo` del objeto `cc`

Métodos y objetos receptores de mensajes

¡ Muy Importante !

 Sin embargo, es posible que dentro de un método se llame a otro método. En éste caso, se dice que el objeto se envía un mensaje a sí mismo.

Por ejemplo, podemos escribir un método para permitir el abono de intereses que se produce a final de año:

```
void Cuenta :: abono_intereses ( )  
{  
    float cantidad;  
    cantidad = (saldo * interes_anual ) / 100 ;  
    ingreso (cantidad);  
}
```

Se calculan los intereses en base al interés anual y se incrementa el saldo.

Cuando no figura delante ningún objeto receptor del mensaje, éste se envía a uno mismo

```
...  
cc.abono_intereses();  
...
```

El objeto cc se envía el mensaje ingreso a sí mismo