
FACULTAD DE INFORMÁTICA

# Introducción a la POO
# Clases y objetos

## RESUMEN

Programación orientada a objetos — Unidad 1

Autor: Luis Hernández Yáñez

FdI UCM

## *Orientado a objetos*

¿Qué se puede calificar como *orientado a objetos*?

La orientación a objetos se refiere a una metodología de desarrollo de software.

Un poco de historia

Años 60 Lenguaje Simula-67 (Noruega).

Proyecto Dynabook de Alan Kay (primera computadora personal)

Smalltalk-80 --> Apple

Años 80 Resurge con fuerza el interés por la OO

Eiffel, C++, Turbo Pascal

Años 90 Gran expansión

Java



FdI UCM

## Evolución de los enfoques de programación

Programación procedimental

*Decidir qué procedimientos se quieren;*
*utilizar los mejores algoritmos que se pueda.*

Procedimientos (funciones, rutinas, ...) Paso de argumentos

Fortran, C, Pascal.

Programación modular

*Decidir qué módulos se quieren; partir el programa de forma que los datos estén ocultos en los módulos.*

Ocultamiento de datos Interfaz
Implementación variable Compilación separada

C, Modula-2.



FdI UCM

## Programación modular

Programación modular: una pila de enteros en C++

```
// PILA.H - Interfaz del módulo
int pop();
void push(int);
const max = 100;
```

Interfaz que permanece

```
// PILA.CPP - Implementación del módulo
#include <PILA.H>
static int v[max];
static int* p = v;
       // pila inicializada a vacía

int pop()
{ ... }

void push(int i)
{ ... }
```

Implementación intercambiable

¿Qué ocurre si necesitamos dos o más pilas?

## Programación con tipos abstractos de datos (TAD)

### Abstracción de datos

*Decidir qué tipos se quieren; proporcionar un conjunto completo de operaciones para cada tipo.*

Encapsulación de código y datos

Modula-2, Ada, C++.

Ejemplo: Matrices cuadradas de enteros de 10 x 10.

Interfaz (vista externa):
Nombre del tipo: **Matriz**
Operaciones:
**+**: **Matriz** x **Matriz** → **Matriz**
**-**: **Matriz** x **Matriz** → **Matriz**
**Invertir**: **Matriz** → **Matriz** ...

Única información que necesita un programador para usar el TAD.

Vista interna (implementación):
Datos: array bidimensional de N x N elementos enteros.
Código de todas y cada una de las operaciones.





## Programación con tipos abstractos de datos (TAD)

Interfaz (vista externa)

```
class Matriz {
public:
  friend Matriz operator+(Matriz, Matriz);
  friend Matriz operator-(Matriz, Matriz);
  friend Matriz Invertir(Matriz);
  // ...
```

Vista interna (implementación)

```
private:
  int matriz[10][10];
};

Matriz operator+(Matriz A, Matriz B)
{
  // ...
}
Matriz operator-(Matriz A, Matriz B)
{
  // ...
```





## Programación con tipos abstractos de datos (TAD)

TAD identificado por un nombre: define un conjunto de operaciones que se pueden aplicar a los ejemplares del tipo.

Ejemplares del TAD: variables de ese tipo.

El TAD encapsula los datos que conforman los ejemplares del tipo y el código de las operaciones establecidas.

- ✓ Los datos y la implementación de las operaciones se ocultan.
- ✓ A los utilizadores del tipo se les proporciona una visión externa constituida por la interfaz (nombre del tipo y forma de uso de las operaciones).

Todos los ejemplares tienen el mismo comportamiento.

Cada ejemplar tiene su propio estado (datos propios).

PROBLEMA:
La adaptación o especialización no se maneja adecuadamente.





## Programación orientada a objetos

### Programación orientada a objetos

*Decidir qué clases se quieren; proporcionar un conjunto completo de operaciones en cada clase; hacer explícitas las semejanzas mediante la herencia.*

Herencia Vinculación dinámica y polimorfismo

Smalltalk, C++, Eiffel.

*Una nueva forma de ver las cosas*

| | Programación con TAD | POO |
|---|---|---|
| Definición | TAD | Clase |
| Estado | Datos | Atributos |
| Comportamiento | Operaciones | Métodos |
| Ejemplares | Variables | Objetos |

## Programación orientada a objetos

En POO, la definición de los ejemplares (estado y comportamiento) se encuentra en la **clase**.

La clase encapsula el código y los datos.

Los ejemplares de las clases son los **objetos**.

La clase define **atributos** (datos) y **métodos** (operaciones).

La vista externa de la clase (su interfaz) viene dada por su nombre y la forma de uso de los métodos.

Los atributos han de ocultarse (ocultamiento de la información).

Cada objeto (ejemplar) contiene su propio conjunto de atributos.

Cada método declarado en la clase constituye un servicio que proporcionan todos los objetos de esa clase.

Todos los objetos de una misma clase proporcionan los mismos servicios.





## Clases en C++

Público y privado

```
class Punto {
public:
  void inicializa();
  void x(double);
  void y(double);
  double x();
  double y();
  void dibujate();
private:
  double _x, _y;
};
```

PROTOTIPOS

- `inicializa()` → Método inicializador
- `x(double)`, `y(double)` → Métodos mutadores
- `x()`, `y()` → Métodos accedentes
- `dibujate()` → Método visualizador

Métodos (funciones miembro)

Atributos

Atributos no objetos

¡¡¡ Atributos privados !!!

```
void Punto::inicializa() {
  _x = 0;
  _y = 0;
}
```

Métodos implementados fuera de la estructura `class`

Convenios de denominación

. . . / . . .





## Clases en C++

```
void Punto::x(double f) {
  _x = f;
}

void Punto::y(double f) {
  _y = f;
}

double Punto::x() {
  return _x;
}

double Punto::y() {
  return _y;
}

void Punto::dibujate() {
  ...
}
```

El código de los métodos puede acceder directamente a todo (público y privado).

Sobrecarga de funciones

| Punto |
|---|
| _x<br>_y |
| inicializa()<br>x():double<br>x(double)<br>y():double<br>y(double)<br>dibujate() |

. . . / . . .





## Objetos

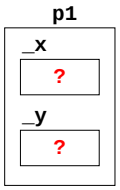
```
int main() {
  Punto p1;
  p1.inicializa();
  p1.x(12);
  p1.y(7);
  p1.dibujate();
  Punto p2;
  p2.inicializa();
  p2 = p1;
  p2.y(23);
  p2.dibujate();

  return 0;
}
```

¡¡¡ Sin acceso a los atributos !!! (acceso sólo a lo público)

Pasos de mensajes

## Objetos

```
int main() {
  Punto p1;
  p1.inicializa();
  p1.x(12);
  p1.y(7);
  p1.dibujate();
  Punto p2;
  p2.inicializa();
  p2 = p1;
  p2.y(23);
  p2.dibujate();

  return 0;
}
```

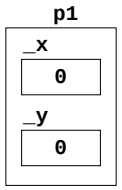







## Objetos

```
int main() {
  Punto p1;
  p1.inicializa();
  p1.x(12);
  p1.y(7);
  p1.dibujate();
  Punto p2;
  p2.inicializa();
  p2 = p1;
  p2.y(23);
  p2.dibujate();

  return 0;
}
```

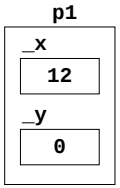







## Objetos

```
int main() {
  Punto p1;
  p1.inicializa();
  p1.x(12);
  p1.y(7);
  p1.dibujate();
  Punto p2;
  p2.inicializa();
  p2 = p1;
  p2.y(23);
  p2.dibujate();

  return 0;
}
```

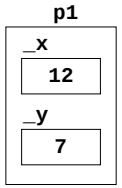







## Objetos

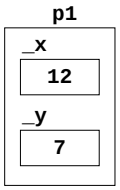
```
int main() {
  Punto p1;
  p1.inicializa();
  p1.x(12);
  p1.y(7);
  p1.dibujate();
  Punto p2;
  p2.inicializa();
  p2 = p1;
  p2.y(23);
  p2.dibujate();

  return 0;
}
```



## Objetos

```
int main() {
  Punto p1;
  p1.inicializa();
  p1.x(12);
  p1.y(7);
  p1.dibujate();
  Punto p2;
  p2.inicializa();
  p2 = p1;
  p2.y(23);
  p2.dibujate();

  return 0;
}
```

Se verá el punto en la pantalla.

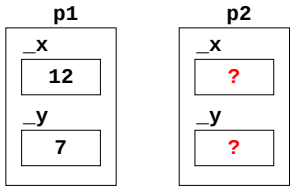







## Objetos

```
int main() {
  Punto p1;
  p1.inicializa();
  p1.x(12);
  p1.y(7);
  p1.dibujate();
  Punto p2;
  p2.inicializa();
  p2 = p1;
  p2.y(23);
  p2.dibujate();

  return 0;
}
```

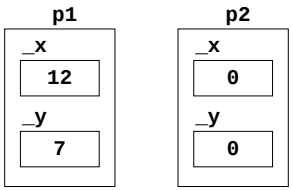







## Objetos

```
int main() {
  Punto p1;
  p1.inicializa();
  p1.x(12);
  p1.y(7);
  p1.dibujate();
  Punto p2;
  p2.inicializa();
  p2 = p1;
  p2.y(23);
  p2.dibujate();

  return 0;
}
```

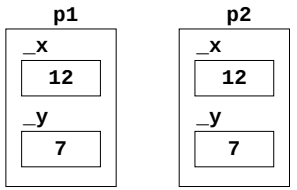






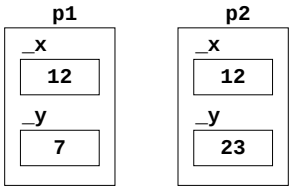

## Objetos

```
int main() {
  Punto p1;
  p1.inicializa();
  p1.x(12);
  p1.y(7);
  p1.dibujate();
  Punto p2;
  p2.inicializa();
  p2 = p1;
  p2.y(23);
  p2.dibujate();

  return 0;
}
```



## Objetos

```
int main() {
  Punto p1;
  p1.inicializa();
  p1.x(12);
  p1.y(7);
  p1.dibujate();
  Punto p2;
  p2.inicializa();
  p2 = p1;
  p2.y(23);
  p2.dibujate();

  return 0;
}
```

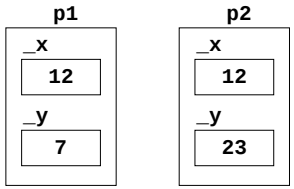







## Objetos

```
int main() {
  Punto p1;
  p1.inicializa();
  p1.x(12);
  p1.y(7);
  p1.dibujate();
  Punto p2;
  p2.inicializa();
  p2 = p1;
  p2.y(23);
  p2.dibujate();

  return 0;
}
```




Dos objetos:
Mismo comportamiento pero diferente estado.





## Clientelismo

```
class Circulo {
public:
  void inicializa();
  void centro(Punto);
  void radio(double);
  Punto centro();
  double radio();
  void dibujate();
private:
  Punto _centro;
  double _radio;
};

void Circulo::inicializa() {
  _radio = 0;
  _centro.inicializa();
}
```

La clase **Circulo** es cliente de la clase **Punto**.

Atributo objeto de otra clase (**Punto**).

Atributos del objeto receptor del mensaje.

Paso de mensaje al objeto-atributo.

. . . / . . .



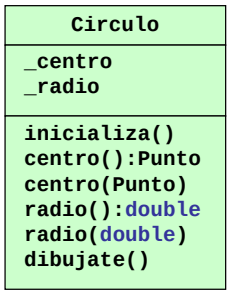


## Clientelismo

```
void Circulo::centro(Punto p) {
  _centro = p;
}

void Circulo::radio(double f) {
  _radio = f;
}

Punto Circulo::centro() {
  return _centro;
}

double Circulo::radio() {
  return _radio;
}

void Circulo::dibujate()
{ ... }
```

Objeto-parámetro

Devolución de un objeto




## Clientelismo

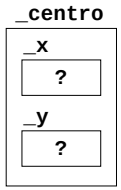
```
#include "punto.h"
#include "circulo.h"

int main() {
  Circulo c1, c2;
  Punto p;
  p.inicializa();
  p.x(2);
  p.y(7);
  c1.inicializa();
  c1.centro(p);
  c1.radio(15);
  c1.dibujate();
  c2.inicializa();
  p.x(5);
  c2.centro(p);
  c2.radio(9);
  c2.dibujate();

  return 0;
}
```





## Clientelismo

```
#include "punto.h"
#include "circulo.h"

int main() {
  Circulo c1, c2;
  Punto p;
  p.inicializa();
  p.x(2);
  p.y(7);
  c1.inicializa();
  c1.centro(p);
  c1.radio(15);
  c1.dibujate();
  c2.inicializa();
  p.x(5);
  c2.centro(p);
  c2.radio(9);
  c2.dibujate();

  return 0;
}
```

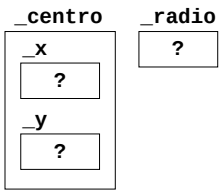







## Clientelismo

```
#include "punto.h"
#include "circulo.h"

int main() {
  Circulo c1, c2;
  Punto p;
  p.inicializa();
  p.x(2);
  p.y(7);
  c1.inicializa();
  c1.centro(p);
  c1.radio(15);
  c1.dibujate();
  c2.inicializa();
  p.x(5);
  c2.centro(p);
  c2.radio(9);
  c2.dibujate();

  return 0;
}
```

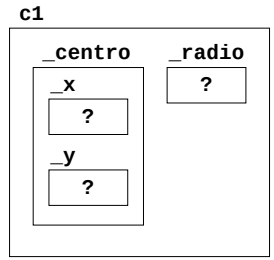






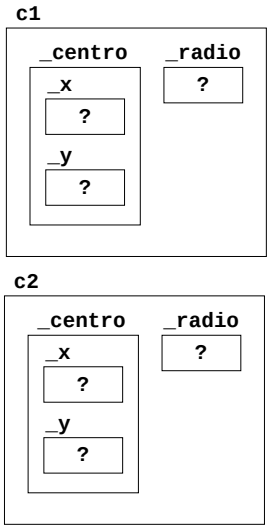

## Clientelismo

```
#include "punto.h"
#include "circulo.h"

int main() {
  Circulo c1, c2;
  Punto p;
  p.inicializa();
  p.x(2);
  p.y(7);
  c1.inicializa();
  c1.centro(p);
  c1.radio(15);
  c1.dibujate();
  c2.inicializa();
  p.x(5);
  c2.centro(p);
  c2.radio(9);
  c2.dibujate();

  return 0;
}
```



## Clientelismo

```
#include "punto.h"
#include "circulo.h"

int main() {
  Circulo c1, c2;
  Punto p;
  p.inicializa();
  p.x(2);
  p.y(7);
  c1.inicializa();
  c1.centro(p);
  c1.radio(15);
  c1.dibujate();
  c2.inicializa();
  p.x(5);
  c2.centro(p);
  c2.radio(9);
  c2.dibujate();

  return 0;
}
```

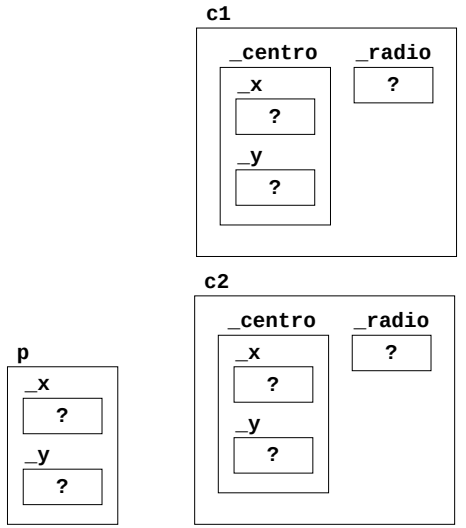







## Clientelismo

```
#include "punto.h"
#include "circulo.h"

int main() {
  Circulo c1, c2;
  Punto p;
  p.inicializa();
  p.x(2);
  p.y(7);
  c1.inicializa();
  c1.centro(p);
  c1.radio(15);
  c1.dibujate();
  c2.inicializa();
  p.x(5);
  c2.centro(p);
  c2.radio(9);
  c2.dibujate();

  return 0;
}
```

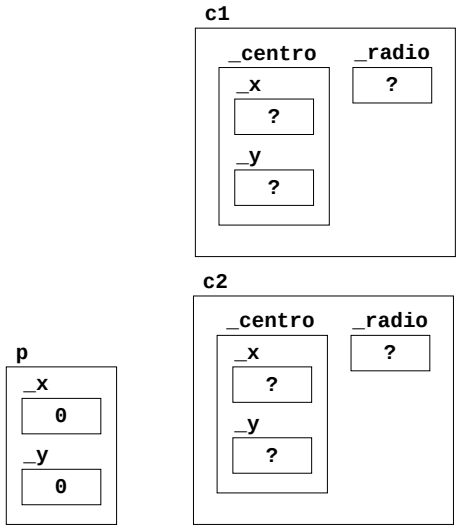







## Clientelismo

```
#include "punto.h"
#include "circulo.h"

int main() {
  Circulo c1, c2;
  Punto p;
  p.inicializa();
  p.x(2);
  p.y(7);
  c1.inicializa();
  c1.centro(p);
  c1.radio(15);
  c1.dibujate();
  c2.inicializa();
  p.x(5);
  c2.centro(p);
  c2.radio(9);
  c2.dibujate();

  return 0;
}
```

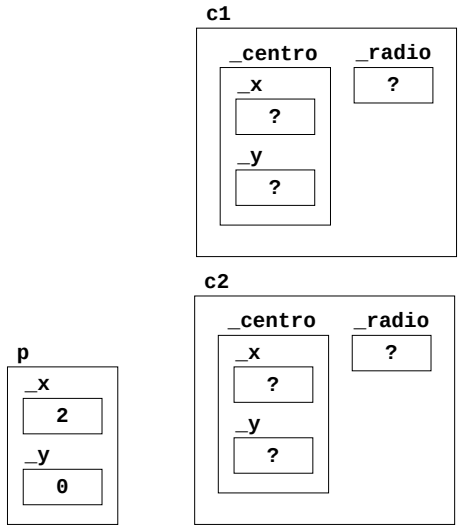






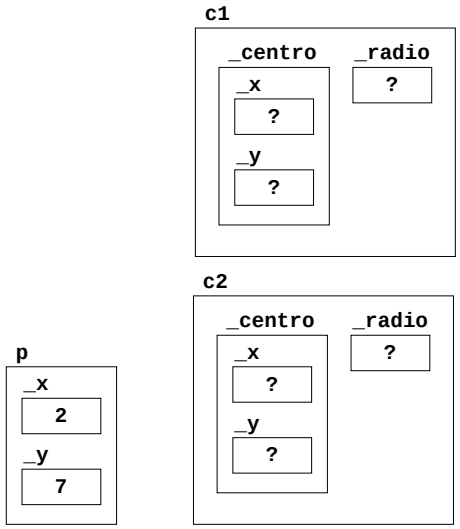

## Clientelismo

```
#include "punto.h"
#include "circulo.h"

int main() {
  Circulo c1, c2;
  Punto p;
  p.inicializa();
  p.x(2);
  p.y(7);
  c1.inicializa();
  c1.centro(p);
  c1.radio(15);
  c1.dibujate();
  c2.inicializa();
  p.x(5);
  c2.centro(p);
  c2.radio(9);
  c2.dibujate();

  return 0;
}
```



## Clientelismo

```
#include "punto.h"
#include "circulo.h"

int main() {
  Circulo c1, c2;
  Punto p;
  p.inicializa();
  p.x(2);
  p.y(7);
  c1.inicializa();
  c1.centro(p);
  c1.radio(15);
  c1.dibujate();
  c2.inicializa();
  p.x(5);
  c2.centro(p);
  c2.radio(9);
  c2.dibujate();

  return 0;
}
```

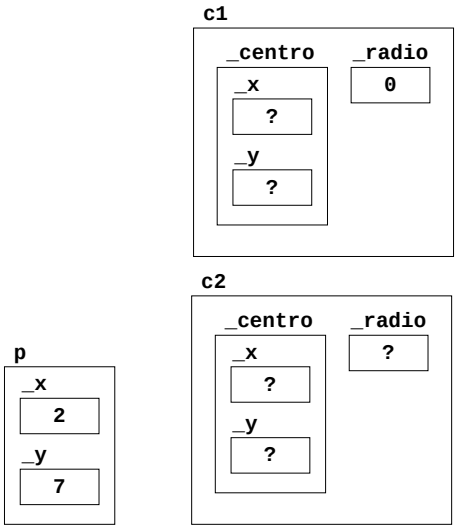







## Clientelismo

```
#include "punto.h"
#include "circulo.h"

int main() {
  Circulo c1, c2;
  Punto p;
  p.inicializa();
  p.x(2);
  p.y(7);
  c1.inicializa();
  c1.centro(p);
  c1.radio(15);
  c1.dibujate();
  c2.inicializa();
  p.x(5);
  c2.centro(p);
  c2.radio(9);
  c2.dibujate();

  return 0;
}
```

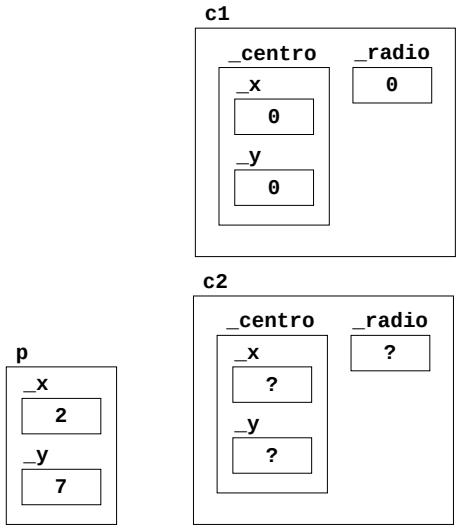







## Clientelismo

```
#include "punto.h"
#include "circulo.h"

int main() {
  Circulo c1, c2;
  Punto p;
  p.inicializa();
  p.x(2);
  p.y(7);
  c1.inicializa();
  c1.centro(p);
  c1.radio(15);
  c1.dibujate();
  c2.inicializa();
  p.x(5);
  c2.centro(p);
  c2.radio(9);
  c2.dibujate();

  return 0;
}
```

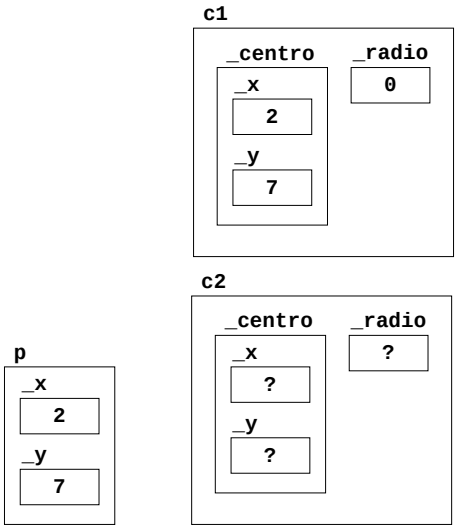




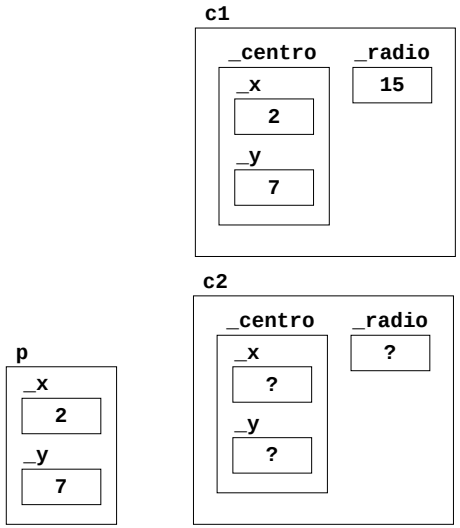



## Clientelismo

```
#include "punto.h"
#include "circulo.h"

int main() {
  Circulo c1, c2;
  Punto p;
  p.inicializa();
  p.x(2);
  p.y(7);
  c1.inicializa();
  c1.centro(p);
  c1.radio(15);
  c1.dibujate();
  c2.inicializa();
  p.x(5);
  c2.centro(p);
  c2.radio(9);
  c2.dibujate();

  return 0;
}
```



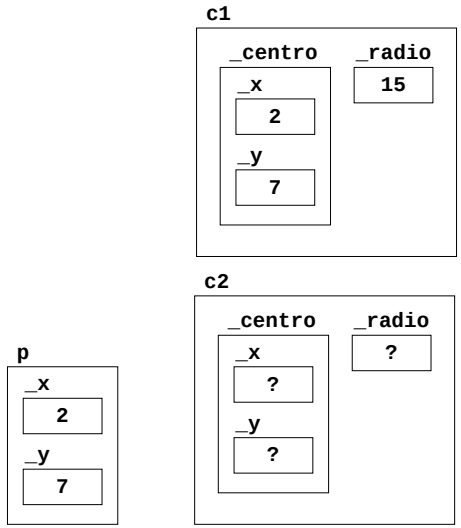
```
#include "punto.h"
#include "circulo.h"

int main() {
  Circulo c1, c2;
  Punto p;
  p.inicializa();
  p.x(2);
  p.y(7);
  c1.inicializa();
  c1.centro(p);
  c1.radio(15);
  c1.dibujate();
  c2.inicializa();
  p.x(5);
  c2.centro(p);
  c2.radio(9);
  c2.dibujate();

  return 0;
}
```







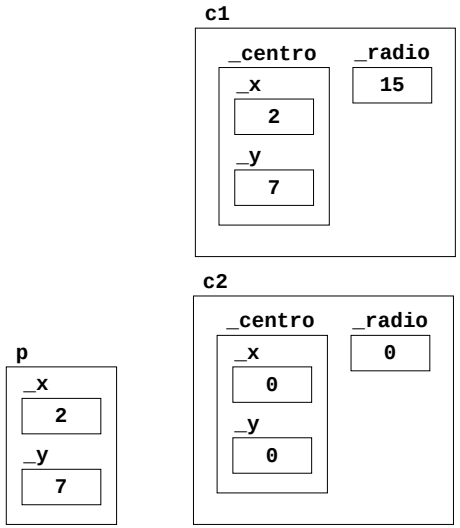

```
#include "punto.h"
#include "circulo.h"

int main() {
  Circulo c1, c2;
  Punto p;
  p.inicializa();
  p.x(2);
  p.y(7);
  c1.inicializa();
  c1.centro(p);
  c1.radio(15);
  c1.dibujate();
  c2.inicializa();
  p.x(5);
  c2.centro(p);
  c2.radio(9);
  c2.dibujate();

  return 0;
}
```







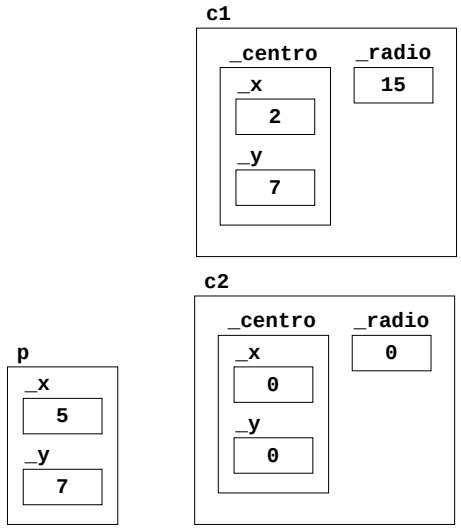

```
#include "punto.h"
#include "circulo.h"

int main() {
  Circulo c1, c2;
  Punto p;
  p.inicializa();
  p.x(2);
  p.y(7);
  c1.inicializa();
  c1.centro(p);
  c1.radio(15);
  c1.dibujate();
  c2.inicializa();
  p.x(5);
  c2.centro(p);
  c2.radio(9);
  c2.dibujate();

  return 0;
}
```





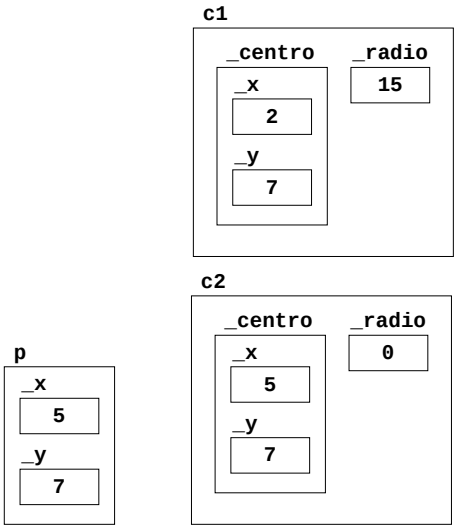



```
#include "punto.h"
#include "circulo.h"

int main() {
  Circulo c1, c2;
  Punto p;
  p.inicializa();
  p.x(2);
  p.y(7);
  c1.inicializa();
  c1.centro(p);
  c1.radio(15);
  c1.dibujate();
  c2.inicializa();
  p.x(5);
  c2.centro(p);
  c2.radio(9);
  c2.dibujate();

  return 0;
}
```



## Clientelismo

```
#include "punto.h"
#include "circulo.h"

int main() {
  Circulo c1, c2;
  Punto p;
  p.inicializa();
  p.x(2);
  p.y(7);
  c1.inicializa();
  c1.centro(p);
  c1.radio(15);
  c1.dibujate();
  c2.inicializa();
  p.x(5);
  c2.centro(p);
  c2.radio(9);
  c2.dibujate();

  return 0;
}
```

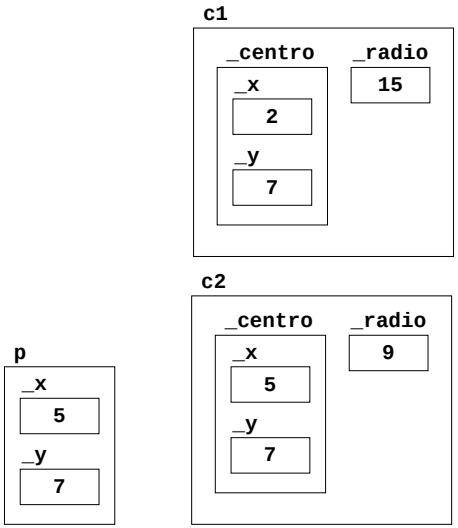







## Clientelismo

```
#include "punto.h"
#include "circulo.h"

int main() {
  Circulo c1, c2;
  Punto p;
  p.inicializa();
  p.x(2);
  p.y(7);
  c1.inicializa();
  c1.centro(p);
  c1.radio(15);
  c1.dibujate();
  c2.inicializa();
  p.x(5);
  c2.centro(p);
  c2.radio(9);
  c2.dibujate();

  return 0;
}
```



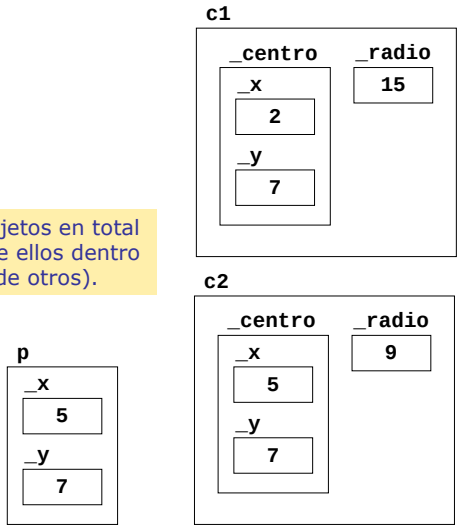





## Clientelismo

```
#include "punto.h"
#include "circulo.h"

int main() {
  Circulo c1, c2;
  Punto p;
  p.inicializa();
  p.x(2);
  p.y(7);
  c1.inicializa();
  c1.centro(p);
  c1.radio(15);
  c1.dibujate();
  c2.inicializa();
  p.x(5);
  c2.centro(p);
  c2.radio(9);
  c2.dibujate();

  return 0;
}
```

5 objetos en total (2 de ellos dentro de otros).








## Una clase **Cuenta**

### Una clase para cuentas de un banco

Atributos: la información ➔ Número de cuenta, saldo e interés anual.

```
long int _numero;
double _saldo;
double _interes;
```

El número de cuenta se establecerá al inicializarla:

```
void inicializa(long int num)
```

Resto de servicios (métodos).-

Accedentes: `double saldo()` `double interes()`

Mutadores: `void saldo(double s)` `void interes(double i)`

Visualizador: `void mostrar()`

Otros:
```
void ingreso(double cantidad)
bool reintegro(double)
void abonoIntereses()
```

## Una clase **Cuenta**

```
#include <iostream> // Entrada/salida por consola
using namespace std; // Funcionalidad estándar

class Cuenta {
public:
  void inicializa(long int);
  double saldo();
  double interes();
  void saldo(double);
  void interes(double);
  void ingreso(double);
  bool reintegro(double);
  void abonoIntereses();
  void mostrar();
private:
  long int _numero;
  double _saldo;
  double _interes; // Interés anual (porcentaje)
}; // Fin de la definición de la clase Cuenta
```

| Cuenta |
| --- |
| |
| inicializa(long int)<br>saldo():double<br>saldo(double)<br>interes():double<br>interes(double)<br>ingreso(double)<br>reintegro(double)<br>abonoIntereses()<br>mostrar() |

. . . / . . .



## Una clase **Cuenta**

```
void Cuenta::inicializa(long int num) {
  _numero = num; _saldo = 0; _interes = 0;
}

double Cuenta::saldo() { return _saldo; }

double Cuenta::interes() { return _interes; }

void Cuenta::saldo(double s) { _saldo = s; }

void Cuenta::interes(double i) { _interes = i; }

void Cuenta::ingreso(double cantidad) {
  _saldo += cantidad;
}
```

. . . / . . .



## Una clase **Cuenta**

```
bool Cuenta::reintegro(double cantidad) {
// Devuelve true si hay saldo suficiente (y resta
// la cantidad); false si no hay saldo suficiente
  if(cantidad > _saldo) return false;
  _saldo -= cantidad;
  return true;
}
void Cuenta::abonoIntereses() {
// abono mensual de intereses
  ingreso(_saldo * _interes / 100 / 12);
}
// El objeto se pasa un mensaje ingreso a sí mismo

void Cuenta::mostrar() {
  cout << endl;
  cout << "Número de cuenta: " << _numero << endl;
  cout << "Saldo: " << _saldo << endl;
}
```

. . . / . . .



## Una clase **Cuenta**

```
int main()
{
  Cuenta cc;
  cc.inicializa(24316534);
  cc.saldo(100000);
  cc.interes(10);
  cc.mostrar();
  cc.ingreso(26000);
  cc.mostrar();
  cc.abonoIntereses();
  cc.mostrar();
  if(!cc.reintegro(10000))
    cout << "No hay saldo";
  cc.mostrar();

  return 0;
}
```

```
Número de cuenta: 24316534
Saldo: 100000

Número de cuenta: 24316534
Saldo: 126000

Número de cuenta: 24316534
Saldo: 127050

Número de cuenta: 24316534
Saldo: 117050
```

>>> Prog0101