```
#include<iostream>
using namespace std;
class nombre{
  //los atributos suelen ser privados
  tipo1 atributo1;
  tipo2 atributo2;
  //...
  //estos miembros pueden ser accedidos por los usuarios
public:
  //podria haber atributos aqui
  //...

  //este es el constructor por defecto
  //se necesita para declarar objetos sin dar de forma
  //explicita valor a sus atributos
  nombre();
  //este es un constructor a partir de valores del tipo de los
  //atributos
  nombre(tipo1, tipo2,...);
  //puede haber mas constructores
  //...
  //esta funcion puede modificar los atributos de la clase
  tipox nombref1(tipoa,tipob,...);
  //esta funcion NO puede modificar los atributos de la clase
  tipox nombref2(tipoa,tipob,...) const;
  //este es un operador binario interno
  //el parametro es el segundo operando
  //el objeto que llama al operador es el operando izquierdo
  //normalmente no se quiere modificar el operando izquierdo
  nombre operator?(const nombre &) const;
  //esta funcion NO PERTENECE a la clase
  //se define fuera de ella, aqui solo se declara
  //la palabra friend le da permismo para acceder a los
  //atributos de la clase
  friend tipox funcion_amiga(tipoa,tipob,...,nombre,...);

private:
  //esta funcion solo puede ser llamada desde otra funcion miembro
  tipox nombref3(...);
};

  nombre::nombre()
  {
    //...
    //se podria dar un valor por defecto a los atributos
    //...
  }

  nombre::nombre(tipo1 p1, tipo2 p2,...)
  {
    //pueden ser necesarias otras instrucciones
    //...
    atributo1=p1;
    atributo2=p2;
    //...
  }

  tipox nombre::nombref1(tipoa param_a,tipob param_b,...)
  {
    //...
    //se accede a los atributos del objeto que llama
    //a la funcion mediante el nombre de estos:
    ...atributo1...
    ...atributo2...
    //...
  }

  tipox nombre::nombref2(tipoa param_a,tipob param_b,...) const
  {
    //...
    //se accede a los atributos del objeto que llama
    //a la funcion mediante el nombre de estos, pero si se
    //intentan modificar el compilador da un error:
    ...atributo1...
    ...atributo2...
    //puede llamar a funciones publicas o privadas de la propia clase
    ...nombref3(...)...
    //...
  }

  nombre nombre::operator?(const nombre &a) const
  {
    //...
    //atributos del operando izquierdo
    ...atributo1...
    ...atributo2...
    //atributos del operando derecho
    ...a.atributo1...
    ...a.atributo2...
  }

  tipox nombre::nombref3(...)
  {
    //...
  }


//definicion de la funcion friend
tipox funcion_amiga(tipoa a,tipob b,...,nombre x,...)
{
  //...
  //se puede acceder a los atributos de x
  ...x.atributo1...
  ...x.atributo2...
  //...
}

int main()
{
  //declaracion de objetos tipo nombre
  //se llama de forma implicita a los constructores
  //correspondientes
  nombre a,b(valor_tipo1, valor_tipo2,...);

  //tambien se puede llamar al constructor de forma explicita
  //para dar valor a un objeto
  a=nombre(valor_tipo1, valor_tipo2,...);

  //los objetos se pueden asignar si no se usa memoria dinamica
  //mas adelante veremos cuando hay que redefinir =
  a=b;

  //los usuarios pueden llamar a las funciones publicas
  ...a.nombref1(...)...

  //la funcion friend se usa como cq funcion
  ...funcion_amiga(valor_tipoa,valor_tipob,...,a,...)...

  //los operadores como cualquier operador
  ...a?b...
}
```