// OOP-Punto.cpp: definizione di una classe Punto (nel main viene richiamato **il costruttore di copia**)

```
#include <iostream>
using namespace std;

// --- Definizione della classe Punto
class Punto
{private:
   // Coordinate del punto
   double x;
   double y;
 public:
   // Costruttori
   Punto();                          // 1° costruttore
   Punto(double s, double t);      // 2° costruttore
   // Visualizza le coordinate del punto
   void printXY();
   // Imposta il valore delle coordinate del punto
   void setPoint(double s, double t);
};

// --- Sezione implementativa della classe
// Costruttore del 1° costruttore della classe Punto
Punto::Punto()
{   x = y = 0;
}
// Definizione del 2° costruttore della classe Punto
// Definizione del metodo printXY della classe Punto
void Punto::printXY()                //scope resolution ::
{   cout << "x = " << x << '\t' << "y = " << y << endl;
}
// Definizione del metodo SetPoint della classe Punto
void Punto::setPoint(double s, double t)
{   // Controllo della correttezza delle coordinate
   x = s < 0 ? 0 : (s > 100 ? 100 : s);
   y = t < 0 ? 0 : (t > 100 ? 100 : t);
}
```

Il **costruttore di copia di default** non è visibile nel listato della classe (è implicito) ed ha la seguente dichiarazione:

```
Punto::Punto(const Punto &s)
{   x = s.x;
    y = s.y;
}
```

```
int main()
{  Punto P, Q(12.5,10.7);          // P e Q sono oggetti (istanze) della classe Punto
   cout<<"P: ";
   P.printXY();  // visualizza le coordinate
   P.setPoint(20,40);

   Punto P1=P;          //Richiama il Costruttore di copia    NON è un assegnamento
   Punto Q1(Q);         // altro modo per richiamare il costruttore di copia
   Punto Q2=Punto(Q);   // altro modo per richiamare il costruttore di copia

   cout<<"P: ";
   P.printXY();    // visualizza le coordinate
   cout<<"P1: ";
   P1.printXY();
   cout<<"Q: ";
   Q.printXY();
   cout<<"Q1: ";
   Q1.printXY();
  cout<<"Q2: ";
  Q2.printXY();
system("pause");  }
```

Si può definire un **costruttore di copia di personalizzato** nella classe, ad esempio:

```
Punto::Punto(const Punto &s)
{   x = s.x + 5;
    y = s.y – 10;
}
```

RIASSUMENDO

- **COSTRUTTORE DI COPIA DI DEFAULT**

Esempio:

```
class NomeClasse
 { …        // attributi
   …
   …
 };
```

Il **costruttore di copia di default** non è visibile nel listato della classe (è implicito) ed ha la seguente dichiarazione:

```
NomeClasse:: NomeClasse (const NomeClasse &a)
{   … = a. ……;
    … = a. ……;
    …
}
```

Chiamata di un costruttore copia

```
NomeClasse A;    // dichiarazione di una istanza di nome A

NomeClasse  copia1diA=A;        // copia

NomeClasse  copia2diA(A);     // copia

NomeClasse  copia3diA = NomeClasse(A);     // copia
```

- **COSTRUTTORE DI COPIA PERSONALIZZATO (quando è necessario?)**

Nella maggior parte dei casi è sufficiente il costruttore di copia di default.

Tuttavia, **se la classe possiede membri puntatori** la copia di default **copia i puntatori**, ma **non le aree puntate** (vedi esempio nella pag. successiva)**.**

Per questo si avrebbero due oggetti i cui rispettivi membri puntatori puntano alla stessa area. Questo può essere pericoloso perché se viene chiamato il distruttore di uno dei due oggetti, l'altro punterà ad un'area che non è più accessibile.

Esempio:

```
class Prova
 { int *p;      // attributi
   float  q;
   …
 };

int main()
{Prova  X;
 Prova  Y=X;
….
```

Il **costruttore**:

```
Prova:: Prova ()
{   p = new int (100);
    q = 15.8;
    …
}
```

Costruttore di copia personalizzato:

```
Prova:: Prova (  const Prova &a)
{   p = new int;
    *p = *a.p;
    q  =a.q;
    …
}
```

// OOP-Prova.cpp: definizione di una classe Punto con **Costruttore di copia personalizzato**
// caso in cui nella classe è presente un puntatore tra gli attributi

```
#include <iostream>
using namespace std;
// --- Definizione della classe Prova
class Prova
{private:
  // Attributi
  int *p;
  float q;
 public:
   // Costruttori
  Prova();                        // costruttore
  Prova(const Prova &);      // costruttore di copia     Provare a commentare con //
                                                        e vedere cosa viene visualizzato dal programma
  // Visualizza valori degli attributi
  void print();
  // Imposta il valore dell'area puntata da p
  void setP(int);
};

// --- Sezione implementativa della classe
// Costruttore
Prova::Prova()
{  p = new int(100);
   q = 15.4;
}
```

Provare a commentare la parte nel riquadro e vedere cosa viene visualizzato dal programma

```
// Definizione costruttore di copia personalizzato
Prova::Prova(const Prova &a)
{
  p = new int;
 *p = *a.p;        // cioè  *(a.p) che è il contenuto dell'area puntata da  a.p
  q = a.q;
}
```

```
// Definizione del metodo di visualizzazione
void Prova::print()                //scope resolution ::
{   cout << "valore puntato da p = " << *p << '\t' << "q = " << q << endl;
}
void Prova::setP(int v)
{  *p = v;
 }

int main()
  { Prova X;
    Prova Y=X;

    cout<<"X: ";
    X.print( );
    cout<<"Y: ";
    Y.print( );

    X.setP(200);

    cout<<"X: ";
    X.print( );
    cout<<"Y: ";
    Y.print( );
   system("pause");
   }
```