



UNIVERSITÀ  
degli STUDI  
di CATANIA

# Creazione, copia e distruzione di oggetti

Corso di programmazione I AA 2021/22

Corso di Laurea Triennale in Informatica

---

Prof. Giovanni Maria Farinella

Web: <http://www.dmi.unict.it/farinella>

Email: [gfarinella@dmi.unict.it](mailto:gfarinella@dmi.unict.it)

Dipartimento di Matematica e Informatica

1. Inizializzazione di oggetti
2. Copia di oggetti
3. Distruzione di oggetti

## Inizializzazione di oggetti

---

## Inizializzazione senza costruttore

```
1  class ClasseX{
2      public:
3          int x; float y;
4  }
5  ClasseX a {1, 2.45};
6  ClasseX b = {3, 4.45}; //memberwise initialization
7  ClasseX c = {3}; //OK, y == 0.0
8  ClasseX d = a; //copy initialization
```

Inizializzazione membro a membro (memberwise), da **lista di inizializzatori** oppure da **altro oggetto** (linea 8).

NB: **No costruttore**, valori di inizializzatori copiati all'interno di attributi (public!) della classe, nell'ordine in cui sono dichiarati.

## Esempi svolti

27\_01.cpp – memberwise initialization

27\_02.cpp – memberwise initialization (copy)

## Inizializzazione con costruttore

Un costruttore rappresenta un modo di **inizializzare automaticamente gli oggetti**.

```
1 | class X{
2   |   int x, y, z;
3   |   public:
4   |   | X(int a, int b, int c){
5   |       x=a; y=b; z=c;
6   |   }
7   | }
8   | //...
9   | X obj(1,2,3); //invocaz. autom. costruttore!
```

### Quali operazioni andrebbero compiute all'interno di un costruttore?

- Inizializzazione di funzioni membro.
- Allocazione (dinamica) di memoria
- ...

L'invocazione del costruttore **avverrà contestualmente alla fase di creazione dell'oggetto.**

Il costruttore **deve avere lo stesso nome della classe.**

Al costruttore **non è associato un tipo di ritorno, quindi non restituisce alcun dato.**

Come qualunque altro metodo:

- Può avere parametri standard.
- Può essere “overloaded”.

## Inizializzazione con costruttore

Un costruttore, se definito, obbligherà il programmatore ad inserire gli inizializzatori previsti da quest'ultimo in fase di creazione dell'oggetto.

```
1  class X{
2      int x, y;
3      public:
4          X(int , int );
5  }
6  //...
7  X obj = {2}; //ERR del comp.! ←
8  X obj1 = {4,5} // OK →
```

## Inizializzazione con costruttore

NB: La notazione con parentesi tonde necessita di un costruttore!

```
1  class X{
2      int x, y;
3      public:
4          X(int, int);
5  }
6  class Y{
7      int x, y;
8  }
9  //...
10 X x1 = {1,2}; //OK, Costruttore
11 X x2 = (4,5); //OK, Costruttore
12 Y y1 = {1,2}; //OK, memberwise initialization
13 Y y2 = (4,5); //Comp. ERR
```

## Inizializzazione con costruttore

Costruttore non overloaded. Nessun costruttore di default.

```
1  class X{
2      int x, y;
3      public:
4          X(int, int);
5  }
6  //...
7  X x1; //ERR del comp.!
8  X x2 {}; //ERR del comp.!
9  X x3 {1,2}; // OK
10 X x4 {1,2,3,4,5}; // ERR del comp!
11 X x5 {x3}; // OK, copia da x3
```

# Inizializzazione con costruttore

**Overloading.** NB: Costruttore di default fornito dal programmatore.

```
1  class X{
2      int x, y;
3      float z;
4      public:
5          X(int , int , float );
6          X(int , int );
7          X(int );
8          X(); ←
9  }
10 X x1 {1 , 1 , 4.5}; // OK
11 X x2 {1 , 1}; // OK
12 X x3 {1}; // OK
13 X x4; // OK
```

## Esempi svolti

27\_03.cpp

27\_04.cpp

# Inizializzazione con costruttore

## Argomenti standard.

```
1  class X{
2      int x, y;
3      float z;
4      public:
5          X(int=0, int=0, float=0.0);
6  }
7  X x1 {1, 1, 4.5}; // OK
8  X x2 {1, 1}; // OK
9  X x3 {1}; // OK
10 X x4; // OK
```

NB: Costruttore di default è senza parametri, in questo caso linea 10 usa argomenti standard.

## Esempi svolti

27\_05.cpp

27\_06.cpp

# Lista di inizializzazione dei membri

Nella definizione del costruttore si può inserire del codice (lista di inizializzazione dei membri) che rappresenta **chiamate ad altri costruttori**:

- ogni chiamata a costruttore rappresenta inizializzazione membro dell'oggetto
- ordine di esecuzione è quello della dichiarazione dei membri stessi.
- nel caso di classe composta, se un oggetto non contiene alcun costruttore, esso non sarà inizializzato

Inizializzazione tipi primitivi.

```
1 class X{
2   int x;
3   float y;
4   public:
5     X(int a, float b) : x(a), y(b){
6       // ...
7     }
8 }
```

## Lista di inizializzazione dei membri

Oggetti composti.

```
1  class Y{
2      float z ;
3      public :
4          Y( float );
5  }
6  class X{
7      int x;
8      Y obj;
9      public :
10         X( int a ) : x(a), obj(0.5) {
11             // ...
12         }
13 }
```

## Lista di inizializzazione dei membri

Membri che rappresentano istanze di classi **senza alcun costruttore di default vanno** necessariamente **inizializzati mediante lista di inizializzazione** (linea 9: errore del compilatore!)

```
1  class Y{
2      float z;
3      public:
4          Y(float );
5  }
6  class X{
7      int x; Y obj;
8      public:
9      X(int a) : x(a) {...} //Comp. ERR
10 }
```

## Lista di inizializzazione dei membri

Variabili **costanti** e **reference** vanno **necessariamente** **inizializzati**.

```
1 class X{
2     float &f;
3     const short k;
4     int x;
5     public:
6     X(int a) : x(a) {...} //COMP. Err
7 }
```

Linea 6: errore del compilatore (f e k vanno inizializzati).

### Esempi svolti

27\_07.cpp

27\_08.cpp

27\_09.cpp

## Copia di oggetti

---

## Costruttori di copia

un **Costruttore di copia** è un costruttore speciale che permette di **creare un oggetto inizializzandolo** con i dati di un **altro oggetto** esistente della stessa classe.

Il costruttore di copia di **default** esegue una copia **membro a membro**.

Viene automaticamente invocato:

1. nelle dichiarazioni con inizializzazione esplicita;
2. nelle chiamate a funzioni con **parametro oggetto passato per valore** (vedi 27\_13.cpp.);
3. quando una **funzione restituisce un oggetto** (vedi 27\_13.cpp).

# Costruttori di copia

Un costruttore di copia presenta un unico parametro formale,  
ovvero un riferimento a costante dello stesso tipo della classe.

```
class X{  
    X(int , float );  
    X (const X &); ←  
}  
X x1 {1, 4.5};  
X x2 {x1}; //invocaz. costruttore di copia  
↑ ↑
```

## Costruttori di copia

```
class X{
    X(int , float );
    X (const X &);
}
X x1 {1, 4.5};
X x2 {x1}; //costruttore di copia
X x3 = x1; //costruttore di copia
x3 = x2; //inizializzazione memberwise
```

- **Si rende necessario** nei casi di allocazione dinamica per i dati della classe;
- si ricordi che viene chiamato **solo nelle inizializzazioni**, non negli assegnamenti;

## Esempi svolti

27\_10.cpp – costruttore di copia

27\_11.cpp – costruttore di copia, oggetti composti

27\_12.cpp – costruttore di copia vs free store

27\_13.cpp – costruttore di copia vs passaggio di parametri

## **Distruzione di oggetti**

---

Una funzione distruttore è una funzione membro:

- viene **invocata automaticamente** al momento della distruzione di un oggetto:
  - a seguito di chiamata a delete;
  - perchè variabile istanza non più visibile (fuori scope);
- è unico all'interno di una classe, e viene denotato dal nome della classe, **anteponendo il simbolo** ~
- A differenza del costruttore, si può invocare esplicitamente.

## Esempi svolti

27\_14.cpp – distruttore

27\_15.cpp – distruttore

FINE