

# ① COSA-STAMPA

COSTRUZIONE → LEFTMOST (C)

DISTRUZIONE → RIGHTMOST (D)

## Esercizio Cosa Stampa

```
class Z {
public: Z(int x) {}
};

class A {
public:
    A() {cout << "A() "; }
    ~A() {cout << "~A "; }
};

class B: public A {
public:
    void f(int) {cout << "B::f(int) "; f(3.14); }
    virtual void f(double) {cout << "B::f(double) "; }
    virtual B* f(Z) {cout << "B::f(Z) "; return this; }
    B() {cout << "B() "; }
    ~B() {cout << "~B "; }
};

class C: virtual public B {
public:
    virtual void f(const int&) {cout<< "C::f(const int&) ";}
    virtual C* f(Z) {cout << "C::f(Z) "; return this;}
    C() {cout << "C() "; }
    virtual ~C() {cout << "~C "; }
};

A* pa = new F; D* pd = new D; E* pe = new E; F* pf = new F; B *pbl=pd, *pb3=pf; C* pc=pf;
```

*Handwritten notes:* F\* puntF=newF; A() B() C() D() C() F()

```
class D: virtual public B {
public:
    D* f(Z) {cout << "D::f(Z) "; f(3.14); return this;}
    virtual void f(double) {cout << "D::f(double) "; }
    D() {cout << "D() "; }
    ~D() {cout << "~D "; }
};

class E: public C {
public:
    virtual void f() {cout << "E::f() "; C::f(Z(1));}
    C* f(Z) {cout << "E::f(Z) "; f(); return this;}
    E() {cout << "E() "; }
    E(const E& e) {cout << "Ec "; }
    ~E() {cout << "~E "; }
};

class F: public E, public D {
public:
    void f() const {cout << "F::f() "; }
    F* f(Z) {cout << "F::f(Z) "; return this;}
    void f(double) {cout << "F::f(double) "; }
    F() {cout << "F() "; }
    ~F() {cout << "~F "; }
};
```

## Esercizio Cosa Stampa

```
class Z {
public: Z(int x) {}
};

class A {
public:
    A() {cout << "A() "; }
    ~A() {cout << "~A "; }
};

class B: public A {
public:
    void f(int) {cout << "B::f(int) "; f(3.14); }
    virtual void f(double) {cout << "B::f(double) "; }
    virtual B* f(Z) {cout << "B::f(Z) "; return this; }
    B() {cout << "B() "; }
    ~B() {cout << "~B "; }
};

class C: virtual public B {
public:
    virtual void f(const int&) {cout<< "C::f(const int&) ";}
    virtual C* f(Z) {cout << "C::f(Z) "; return this;}
    C() {cout << "C() "; }
    virtual ~C() {cout << "~C "; }
};

A* pa = new F; D* pd = new D; E* pe = new E; F* pf = new F; B *pbl=pd, *pb3=pf; C* pc=pf;
```

*Handwritten notes:* COPY A, E\* puntE = new E(\*pe), A() B() C() C()

```
class D: virtual public B {
public:
    D* f(Z) {cout << "D::f(Z) "; f(3.14); return this;}
    virtual void f(double) {cout << "D::f(double) "; }
    D() {cout << "D() "; }
    ~D() {cout << "~D "; }
};

class E: public C {
public:
    virtual void f() {cout << "E::f() "; C::f(Z(1));}
    C* f(Z) {cout << "E::f(Z) "; f(); return this;}
    E() {cout << "E() "; }
    E(const E& e) {cout << "Ec "; }
    ~E() {cout << "~E "; }
};

class F: public E, public D {
public:
    void f() const {cout << "F::f() "; }
    F* f(Z) {cout << "F::f(Z) "; return this;}
    void f(double) {cout << "F::f(double) "; }
    F() {cout << "F() "; }
    ~F() {cout << "~F "; }
};
```

N.B!

## Esercizio Cosa Stampa

```
class Z {
public: Z(int x) {}
};

class A {
public:
    A() {cout << "A() "; }
    ~A() {cout << "~A "; }
};

class B: public A {
public:
    void f(int) {cout << "B::f(int) "; f(3.14); }
    virtual void f(double) {cout << "B::f(double) "; }
    virtual B* f(Z) {cout << "B::f(Z) "; return this; }
    B() {cout << "B() "; }
    ~B() {cout << "~B "; }
};

class C: virtual public B {
public:
    virtual void f(const int&) {cout<< "C::f(const int&) ";}
    virtual C* f(Z) {cout << "C::f(Z) "; return this;}
    C() {cout << "C() "; }
    virtual ~C() {cout << "~C "; }
};

A* pa = new F; D* pd = new D; E* pe = new E; F* pf = new F; B *pbl=pd, *pb3=pf; C* pc=pf;
```

*Handwritten notes:* F → A() B() C() D() C() F(), For operations: D:: operation, C:: operation, STANDARDS

```
class D: virtual public B {
public:
    D* f(Z) {cout << "D::f(Z) "; f(3.14); return this;}
    virtual void f(double) {cout << "D::f(double) "; }
    D() {cout << "D() "; }
    ~D() {cout << "~D "; }
};

class E: public C {
public:
    virtual void f() {cout << "E::f() "; C::f(Z(1));}
    C* f(Z) {cout << "E::f(Z) "; f(); return this;}
    E() {cout << "E() "; }
    E(const E& e) {cout << "Ec "; }
    ~E() {cout << "~E "; }
};

class F: public E, public D {
public:
    void f() const {cout << "F::f() "; }
    F* f(Z) {cout << "F::f(Z) "; return this;}
    void f(double) {cout << "F::f(double) "; }
    F() {cout << "F() "; }
    ~F() {cout << "~F "; }
};
```

### Esercizio Cosa Stampa

```

class Z {
public: Z(int x) {}
};

class A {
public:
    A() {cout << "A() "; }
    ~A() {cout << "~A "; }
};

class B: public A {
public:
    void f(int) {cout << "B::f(int) "; f(3.14); }
    virtual void f(double) {cout << "B::f(double) "; }
    virtual B* f(Z) {cout << "B::f(Z) "; return this; }
    B() {cout << "B() "; }
    ~B() {cout << "~B "; }
};

class C: virtual public B {
public:
    virtual void f(const int&) {cout<< "C::f(const int&) "; }
    virtual C* f(Z) {cout << "C::f(Z) "; return this;}
    C() {cout << "C() "; }
    virtual ~C() {cout << "~C "; }
};

class D: virtual public B {
public:
    D* f(Z) {cout << "D::f(Z) "; f(3.14); return this;}
    virtual void f(double) {cout << "D::f(double) "; }
    D() {cout << "D() "; }
    ~D() {cout << "~D "; }
};

class E: public C {
public:
    virtual void f() {cout << "E::f() "; C::f(Z(1)); }
    C* f(Z) {cout << "E::f(Z) "; f(); return this;}
    E() {cout << "E() "; }
    E(const E& e) {cout << "Ec "; }
    ~E() {cout << "~E "; }
};

class F: public E, public D {
public:
    void f() const {cout << "F::f() "; }
    F* f(Z) {cout << "F::f(Z) "; return this;}
    void f(double) {cout << "F::f(double) "; }
    F() {cout << "F() "; }
    ~F() {cout << "~F "; }
};

A* pa = new F; D* pd = new D; E* pe = new E; F* pf = new F; B *pb1=pd, *pb3=pf; C* pc=pf;
    
```

Handwritten notes and diagram:

- `pb3->f(3);` with `TS(B)` and `TOLE` written next to it.
- A call graph diagram showing the execution flow: `A` (root) calls `B`, which calls `C`, which calls `D`, which calls `E`, which calls `F`. Arrows indicate the sequence of calls.
- Arrows point from the code to the diagram, indicating which part of the code is being executed.

### Esercizio Cosa Stampa

```

class Z {
public: Z(int x) {}
};

class A {
public:
    A() {cout << "A() "; }
    ~A() {cout << "~A "; }
};

class B: public A {
public:
    void f(int) {cout << "B::f(int) "; f(3.14); }
    virtual void f(double) {cout << "B::f(double) "; }
    virtual B* f(Z) {cout << "B::f(Z) "; return this; }
    B() {cout << "B() "; }
    ~B() {cout << "~B "; }
};

class C: virtual public B {
public:
    virtual void f(const int&) {cout<< "C::f(const int&) "; }
    virtual C* f(Z) {cout << "C::f(Z) "; return this;}
    C() {cout << "C() "; }
    virtual ~C() {cout << "~C "; }
};

class D: virtual public B {
public:
    D* f(Z) {cout << "D::f(Z) "; f(3.14); return this;}
    virtual void f(double) {cout << "D::f(double) "; }
    D() {cout << "D() "; }
    ~D() {cout << "~D "; }
};

class E: public C {
public:
    virtual void f() {cout << "E::f() "; C::f(Z(1)); }
    C* f(Z) {cout << "E::f(Z) "; f(); return this;}
    E() {cout << "E() "; }
    E(const E& e) {cout << "Ec "; }
    ~E() {cout << "~E "; }
};

class F: public E, public D {
public:
    void f() const {cout << "F::f() "; }
    F* f(Z) {cout << "F::f(Z) "; return this;}
    void f(double) {cout << "F::f(double) "; }
    F() {cout << "F() "; }
    ~F() {cout << "~F "; }
};

A* pa = new F; D* pd = new D; E* pe = new E; F* pf = new F; B *pb1=pd, *pb3=pf; C* pc=pf;
    
```

Handwritten notes and diagram:

- `pa->f(1.2);` with `pa` written next to it.
- A call graph diagram showing the execution flow: `A` (root) calls `B`, which calls `C`, which calls `D`, which calls `E`, which calls `F`. Arrows indicate the sequence of calls.
- Arrows point from the code to the diagram, indicating which part of the code is being executed.

### Esercizio Cosa Stampa

```

class Z {
public: Z(int x) {}
};

class A {
public:
    A() {cout << "A() "; }
    ~A() {cout << "~A "; }
};

class B: public A {
public:
    void f(int) {cout << "B::f(int) "; f(3.14); }
    virtual void f(double) {cout << "B::f(double) "; }
    virtual B* f(Z) {cout << "B::f(Z) "; return this; }
    B() {cout << "B() "; }
    ~B() {cout << "~B "; }
};

class C: virtual public B {
public:
    virtual void f(const int&) {cout<< "C::f(const int&) "; }
    virtual C* f(Z) {cout << "C::f(Z) "; return this;}
    C() {cout << "C() "; }
    virtual ~C() {cout << "~C "; }
};

class D: virtual public B {
public:
    D* f(Z) {cout << "D::f(Z) "; f(3.14); return this;}
    virtual void f(double) {cout << "D::f(double) "; }
    D() {cout << "D() "; }
    ~D() {cout << "~D "; }
};

class E: public C {
public:
    virtual void f() {cout << "E::f() "; C::f(Z(1)); }
    C* f(Z) {cout << "E::f(Z) "; f(); return this;}
    E() {cout << "E() "; }
    E(const E& e) {cout << "Ec "; }
    ~E() {cout << "~E "; }
};

class F: public E, public D {
public:
    void f() const {cout << "F::f() "; }
    F* f(Z) {cout << "F::f(Z) "; return this;}
    void f(double) {cout << "F::f(double) "; }
    F() {cout << "F() "; }
    ~F() {cout << "~F "; }
};

A* pa = new F; D* pd = new D; E* pe = new E; F* pf = new F; B *pb1=pd, *pb3=pf; C* pc=pf;
    
```

Handwritten notes and diagram:

- `pb1->f(Z(2));` with `pb1` written next to it.
- A call graph diagram showing the execution flow: `A` (root) calls `B`, which calls `C`, which calls `D`, which calls `E`, which calls `F`. Arrows indicate the sequence of calls.
- Arrows point from the code to the diagram, indicating which part of the code is being executed.

### Esercizio Cosa Stampa

```

class Z {
public: Z(int x) {}
};

class A {
public:
    A() {cout << "A() "; }
    ~A() {cout << "~A "; }
};

class B: public A {
public:
    void f(int) {cout << "B::f(int) "; f(3.14); }
    virtual void f(double) {cout << "B::f(double) "; }
    virtual B* f(Z) {cout << "B::f(Z) "; return this; }
    B() {cout << "B() "; }
    ~B() {cout << "~B "; }
};

class C: virtual public B {
public:
    virtual void f(const int&) {cout<< "C::f(const int&) "; }
    virtual C* f(Z) {cout << "C::f(Z) "; return this;}
    C() {cout << "C() "; }
    virtual ~C() {cout << "~C "; }
};

class D: virtual public B {
public:
    D* f(Z) {cout << "D::f(Z) "; f(3.14); return this;}
    virtual void f(double) {cout << "D::f(double) "; }
    D() {cout << "D() "; }
    ~D() {cout << "~D "; }
};

class E: public C {
public:
    virtual void f() {cout << "E::f() "; C::f(Z(1));}
    C* f(Z) {cout << "E::f(Z) "; f(); return this;}
    E() {cout << "E() "; }
    E(const E& e) {cout << "Ec "; }
    ~E() {cout << "~E "; }
};

class F: public E, public D {
public:
    void f() const {cout << "F::f() "; }
    F* f(Z) {cout << "F::f(Z) "; return this;}
    void f(double) {cout << "F::f(double) "; }
    F() {cout << "F() "; }
    ~F() {cout << "~F "; }
};

A* pa = new F; D* pd = new D; E* pe = new E; F* pf = new F; B *pbl=pd, *pb3=pf; C* pc=pf;
    
```

**if(typeid(pb3)==typeid(F)) pb3->f(Z(2));**

**F = c F**

**F::f(Z)**

### Esercizio Cosa Stampa

```

class Z {
public: Z(int x) {}
};

class A {
public:
    A() {cout << "A() "; }
    ~A() {cout << "~A "; }
};

class B: public A {
public:
    void f(int) {cout << "B::f(int) "; f(3.14); }
    virtual void f(double) {cout << "B::f(double) "; }
    virtual B* f(Z) {cout << "B::f(Z) "; return this; }
    B() {cout << "B() "; }
    ~B() {cout << "~B "; }
};

class C: virtual public B {
public:
    virtual void f(const int&) {cout<< "C::f(const int&) "; }
    virtual C* f(Z) {cout << "C::f(Z) "; return this;}
    C() {cout << "C() "; }
    virtual ~C() {cout << "~C "; }
};

class D: virtual public B {
public:
    D* f(Z) {cout << "D::f(Z) "; f(3.14); return this;}
    virtual void f(double) {cout << "D::f(double) "; }
    D() {cout << "D() "; }
    ~D() {cout << "~D "; }
};

class E: public C {
public:
    virtual void f() {cout << "E::f() "; C::f(Z(1));}
    C* f(Z) {cout << "E::f(Z) "; f(); return this;}
    E() {cout << "E() "; }
    E(const E& e) {cout << "Ec "; }
    ~E() {cout << "~E "; }
};

class F: public E, public D {
public:
    void f() const {cout << "F::f() "; }
    F* f(Z) {cout << "F::f(Z) "; return this;}
    void f(double) {cout << "F::f(double) "; }
    F() {cout << "F() "; }
    ~F() {cout << "~F "; }
};

A* pa = new F; D* pd = new D; E* pe = new E; F* pf = new F; B *pbl=pd, *pb3=pf; C* pc=pf;
    
```

**C\* pc = new F();**

**static\_cast<E\*>(pc)->f();**

**B \*PC = new F();**

### Esercizio Cosa Stampa

```

class Z {
public: Z(int x) {}
};

class A {
public:
    A() {cout << "A() "; }
    ~A() {cout << "~A "; }
};

class B: public A {
public:
    void f(int) {cout << "B::f(int) "; f(3.14); }
    virtual void f(double) {cout << "B::f(double) "; }
    virtual B* f(Z) {cout << "B::f(Z) "; return this; }
    B() {cout << "B() "; }
    ~B() {cout << "~B "; }
};

class C: virtual public B {
public:
    virtual void f(const int&) {cout<< "C::f(const int&) "; }
    virtual C* f(Z) {cout << "C::f(Z) "; return this;}
    C() {cout << "C() "; }
    virtual ~C() {cout << "~C "; }
};

class D: virtual public B {
public:
    D* f(Z) {cout << "D::f(Z) "; f(3.14); return this;}
    virtual void f(double) {cout << "D::f(double) "; }
    D() {cout << "D() "; }
    ~D() {cout << "~D "; }
};

class E: public C {
public:
    virtual void f() {cout << "E::f() "; C::f(Z(1));}
    C* f(Z) {cout << "E::f(Z) "; f(); return this;}
    E() {cout << "E() "; }
    E(const E& e) {cout << "Ec "; }
    ~E() {cout << "~E "; }
};

class F: public E, public D {
public:
    void f() const {cout << "F::f() "; }
    F* f(Z) {cout << "F::f(Z) "; return this;}
    void f(double) {cout << "F::f(double) "; }
    F() {cout << "F() "; }
    ~F() {cout << "~F "; }
};

A* pa = new F; D* pd = new D; E* pe = new E; F* pf = new F; B *pbl=pd, *pb3=pf; C* pc=pf;
    
```

**B -> f(4) -> B::f(WF)**

**(pb3->f(Z(3)))->f(4);**

**NOVA IN B (RASS)**