

Laurea in Informatica – Programmazione ad Oggetti – Appello d'Esame 25/01/2024

Nome..... Cognome..... Matricola.....

Esercizio Cosa Stampa

```
class A {
public:
    A() {cout<< " A() ";}
    ~A() {cout<< " ~A ";}
    A(const A& x) {cout<< " Ac ";}
    virtual const A* j() {cout<<" A::j "; return this;}
    virtual void k() {cout <<" A::k "; m();}
    void m() {cout <<" A::m "; j();}
};

class C: virtual public B {
public:
    C() {cout<< " C() ";}
    ~C() {cout<< " ~C ";}
    void g() const {cout <<" C::g ";}
    void k() override {cout <<" C::k "; B::n();}
    virtual void m() {cout <<" C::m "; g(); j();}
    B& n() override {cout <<" C::n "; return *this;}
};

class E: public C, public D {
public:
    E() {cout<< " E() ";}
    ~E() {cout<< " ~E ";}
    E(const E& x) {cout<< " Ec ";}
    virtual void g() const {cout <<" E::g ";}
    const E* j() {cout <<" E::j "; return this;}
    void m() {cout <<" E::m "; g(); j();}
    D& n() final {cout <<" E::n "; return *this;}
};

A* p1 = new E(); B* p2 = new C(); A* p3 = new D(); B* p4 = new E();
const A* p5 = new D(); const B* p6 = new E(); const E* p7 = new E();
```

```
class B: virtual public A {
public:
    B() {cout<< " B() ";}
    virtual ~B() {cout<< " ~B ";}
    virtual void g() const {cout <<" B::g ";}
    virtual const B* j() {cout <<" B::j "; n(); return this;}
    void k() {cout <<" B::k "; j(); m();}
    void m() {cout <<" B::m "; g(); j();}
    virtual A& n() {cout <<" B::n "; return *this;}
};

class D: virtual public B {
public:
    D() {cout<< " D() ";}
    ~D() {cout<< " ~D ";}
    virtual void g() {cout <<" D::g ";}
    const B* j() {cout <<" D::j "; return this;}
    void k() const {cout <<" D::k "; k();}
    void m() {cout <<" D::m "; g(); j();}
};
```

NON CONS
 ↓
 p1 → j() → k() (X)
 NC = THIS CONS
 p1 NON CONS

Le precedenti definizioni compilano correttamente. Per ognuna delle seguenti istruzioni scrivere nell'apposito spazio:

- **NON COMPILA** se la compilazione dell'istruzione provoca un errore;
- **UNDEFINED** se lo statement compila correttamente ma la sua esecuzione provoca un undefined behaviour o un errore run-time;
- se l'istruzione compila correttamente e non provoca errori a run-time allora si scriva la stampa che l'esecuzione produce in output su cout; se non provoca alcuna stampa allora si scriva **NESSUNA STAMPA**.

```
(p1->j())->k(); .....
(dynamic_cast<const E*>(p1->j()))->g(); .....
p2->m(); .....
(p2->j())->g(); .....
p3->k(); .....
(p4->n()).m(); .....
((dynamic_cast<D*>(p4))->n()).k(); .....
(dynamic_cast<E*>(p5))->j(); .....
(dynamic_cast<E*>(const_cast<B*>(p6)))>k(); .....
new E(*p7); .....
delete p1; .....
delete p4; .....
```

Laurea in Informatica – Programmazione ad Oggetti – Appello d'Esame 25/01/2024

Nome..... Cognome..... Matricola.....

Esercizio Cosa Stampa

```
class A {
public:
    A() {cout<< " A() ";}
    ~A() {cout<< " ~A ";}
    A(const A& x) {cout<< " Ac ";}
    virtual const A* j() {cout<< " A::j "; return this;}
    virtual void k() {cout<< " A::k "; m();}
    void m() {cout<< " A::m "; j();}
};

class C: virtual public B {
public:
    C() {cout<< " C() ";}
    ~C() {cout<< " ~C ";}
    void g() const {cout<< " C::g ";}
    void k() override {cout<< " C::k "; B::n();}
    virtual void m() {cout<< " C::m "; g(); j();}
    B& n() override {cout<< " C::n "; return *this;}
};

class E: public C, public D {
public:
    E() {cout<< " E() ";}
    ~E() {cout<< " ~E ";}
    E(const E& x) {cout<< " Ec ";}
    virtual void g() const {cout<< " E::g ";}
    const E* j() {cout<< " E::j "; return this;}
    void m() {cout<< " E::m "; g(); j();}
    D& n() final {cout<< " E::n "; return *this;}
};

class B: virtual public A {
public:
    B() {cout<< " B() ";}
    virtual ~B() {cout<< " ~B ";}
    virtual void g() const {cout<< " B::g ";}
    virtual const B* j() {cout<< " B::j "; n(); return this;}
    void k() {cout<< " B::k "; j(); m();}
    void m() {cout<< " B::m "; g(); j();}
    virtual A& n() {cout<< " B::n "; return *this;}
};

class D: virtual public B {
public:
    D() {cout<< " D() ";}
    ~D() {cout<< " ~D ";}
    virtual void g() {cout<< " D::g ";}
    const B* j() {cout<< " D::j "; return this;}
    void k() const {cout<< " D::k "; k();}
    void m() {cout<< " D::m "; g(); j();}
};

A* p1 = new E(); B* p2 = new C(); A* p3 = new D(); B* p4 = new E();
const A* p5 = new D(); const B* p6 = new E(); const E* p7 = new E();
```

(dynamic_cast<const E*>(p1->j()))->g();

(2°)

RETURN THIS (1°)

Le precedenti definizioni compilano correttamente. Per ognuna delle seguenti istruzioni scrivere nell'apposito spazio:

- **NON COMPILA** se la compilazione dell'istruzione provoca un errore;
- **UNDEFINED** se lo statement compila correttamente ma la sua esecuzione provoca un undefined behaviour o un errore run-time;
- se l'istruzione compila correttamente e non provoca errori a run-time allora si scriva la stampa che l'esecuzione produce in output su cout; se non provoca alcuna stampa allora si scriva **NESSUNA STAMPA**.

```
(p1->j())->k(); .....
(dynamic_cast<const E*>(p1->j()))->g(); .....
p2->m(); .....
(p2->j())->g(); .....
p3->k(); .....
(p4->n()).m(); .....
((dynamic_cast<D*>(p4))->n()).k(); .....
(dynamic_cast<E*>(p5))->j(); .....
(dynamic_cast<E*>(const_cast<B*>(p6)))>k(); .....
new E(*p7); .....
delete p1; .....
delete p4; .....
```

Laurea in Informatica – Programmazione ad Oggetti – Appello d'Esame 25/01/2024

Nome..... Cognome..... Matricola.....

Esercizio Cosa Stampa

```

class A {
public:
    A() {cout<< " A() ";}
    ~A() {cout<< " ~A ";}
    A(const A& x) {cout<< " Ac ";}
    virtual const A* j() {cout<<" A::j "; return this;}
    virtual void k() {cout <<" A::k "; m();}
    void m() {cout <<" A::m "; j();}
};

class C: virtual public B {
public:
    C() {cout<< " C() ";}
    ~C() {cout<< " ~C ";}
    void g() const {cout <<" C::g ";}
    void k() override {cout <<" C::k "; B::n();}
    virtual void m() {cout <<" C::m "; g(); j();}
    B& n() override {cout <<" C::n "; return *this;}
};

class E: public C, public D {
public:
    E() {cout<< " E() ";}
    ~E() {cout<< " ~E ";}
    E(const E& x) {cout<< " Ec ";}
    virtual void g() const {cout <<" E::g ";}
    const E* j() {cout <<" E::j "; return this;}
    void m() {cout <<" E::m "; g(); j();}
    D& n() final {cout <<" E::n "; return *this;}
};

A* p1 = new E(); B* p2 = new C(); A* p3 = new D(); B* p4 = new E();
const A* p5 = new D(); const B* p6 = new E(); const E* p7 = new E();
    
```

Handwritten notes:

- NC (Non-Const) with an arrow pointing to the `const E* j()` line.
- NON CONST with an arrow pointing to the `dynamic_cast<E*>(p5) -> j()` expression.
- THIS = CONST with an arrow pointing to the `dynamic_cast<E*>(p5) -> j()` expression.

Le precedenti definizioni compilano correttamente. Per ognuna delle seguenti istruzioni scrivere nell'apposito spazio:

- **NON COMPILA** se la compilazione dell'istruzione provoca un errore;
- **UNDEFINED** se lo statement compila correttamente ma la sua esecuzione provoca un undefined behaviour o un errore run-time;
- se l'istruzione compila correttamente e non provoca errori a run-time allora si scriva la stampa che l'esecuzione produce in output su cout; se non provoca alcuna stampa allora si scriva **NESSUNA STAMPA**.

```

(p1->j())->k(); .....
(dynamic_cast<const E*>(p1->j()))->g(); .....
p2->m(); .....
(p2->j())->g(); .....
p3->k(); .....
(p4->n()).m(); .....
((dynamic_cast<D*>(p4)->n()).k(); .....
(dynamic_cast<E*>(p5)->j()); .....
(dynamic_cast<E*>(const_cast<B*>(p6)))->k(); .....
new E(*p7); .....
delete p1; .....
delete p4; .....
    
```

Laurea in Informatica – Programmazione ad Oggetti – Appello d'Esame 25/01/2024

Nome..... Cognome..... Matricola.....

Esercizio Cosa Stampa

```
class A {
public:
    A() {cout<< "A() ";}
    ~A() {cout<< " ~A ";}
    A(const A& x) {cout<< " Ac ";}
    virtual const A* j() {cout<<" A::j "; return this;}
    virtual void k() {cout <<" A::k "; m();}
    void m() {cout <<" A::m "; j();}
};

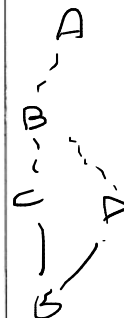
class C: virtual public B {
public:
    C() {cout<< " C() ";}
    ~C() {cout<< " ~C ";}
    void g() const {cout <<" C::g ";}
    void k() override {cout <<" C::k "; B::n();}
    virtual void m() {cout <<" C::m "; g(); j();}
    B& n() override {cout <<" C::n "; return *this;}
};

class E: public C, public D {
public:
    E() {cout<< " E() ";}
    ~E() {cout<< " ~E ";}
    E(const E& x) {cout<< " Ec ";}
    virtual void g() const {cout <<" E::g ";}
    const E* j() {cout <<" E::j "; return this;}
    void m() {cout <<" E::m "; g(); j();}
    D& n() final {cout <<" E::n "; return *this;}
};

class B: virtual public A {
public:
    B() {cout<< "B() ";}
    virtual ~B() {cout<< " ~B ";}
    virtual void g() const {cout <<" B::g ";}
    virtual const B* j() {cout <<" B::j "; n(); return this;}
    void k() {cout <<" B::k "; j(); m();}
    void m() {cout <<" B::m "; g(); j();}
    virtual A& n() {cout <<" B::n "; return *this;}
};

class D: virtual public B {
public:
    D() {cout<< "D() ";}
    ~D() {cout<< " ~D ";}
    virtual void g() {cout <<" D::g ";}
    const B* j() {cout <<" D::j "; return this;}
    void k() const {cout <<" D::k "; k();}
    void m() {cout <<" D::m "; g(); j();}
};

A* p1 = new E(); B* p2 = new C(); A* p3 = new D(); B* p4 = new E();
const A* p5 = new D(); const B* p6 = new E(); const E* p7 = new E();
```



new E(*p7) cout << endl;

A C) B C) C C) D C) C C

Le precedenti definizioni compilano correttamente. Per ognuna delle seguenti istruzioni scrivere nell'apposito spazio:

- **NON COMPILA** se la compilazione dell'istruzione provoca un errore;
- **UNDEFINED** se lo statement compila correttamente ma la sua esecuzione provoca un undefined behaviour o un errore run-time;
- se l'istruzione compila correttamente e non provoca errori a run-time allora si scriva la stampa che l'esecuzione produce in output su cout; se non provoca alcuna stampa allora si scriva **NESSUNA STAMPA**.

```
(p1->j())->k(); .....
(dynamic_cast<const E*>(p1->j()))->g(); .....
p2->m(); .....
(p2->j())->g(); .....
p3->k(); .....
(p4->n()).m(); .....
((dynamic_cast<D*>(p4))->n()).k(); .....
(dynamic_cast<E*>(p5))->j(); .....
(dynamic_cast<E*>(const_cast<B*>(p6)))>k(); .....
new E(*p7); .....
delete p1; .....
delete p4; .....
```

Laurea in Informatica – Programmazione ad Oggetti – Appello d'Esame 25/01/2024

Nome..... Cognome..... Matricola.....

Esercizio Cosa Stampa

```
class A {
public:
    A() {cout<< " A() ";}
    ~A() {cout<< " ~A ";}
    A(const A& x) {cout<< " Ac ";}
    virtual const A* j() {cout<<" A::j "; return this;}
    virtual void k() {cout <<" A::k "; m();}
    void m() {cout <<" A::m "; j();}
};

class C: virtual public B {
public:
    C() {cout<< " C() ";}
    ~C() {cout<< " ~C ";}
    void g() const {cout <<" C::g ";}
    void k() override {cout <<" C::k "; B::n();}
    virtual void m() {cout <<" C::m "; g(); j();}
    B& n() override {cout <<" C::n "; return *this;}
};

class E: public C, public D {
public:
    E() {cout<< " E() ";}
    ~E() {cout<< " ~E ";}
    E(const E& x) {cout<< " Ec ";}
    virtual void g() const {cout <<" E::g ";}
    const E* j() {cout <<" E::j "; return this;}
    void m() {cout <<" E::m "; g(); j();}
    D& n() final {cout <<" E::n "; return *this;}
};

A* p1 = new E(); B* p2 = new C(); A* p3 = new D(); B* p4 = new E();
const A* p5 = new D(); const B* p6 = new E(); const E* p7 = new E();

class B: virtual public A {
public:
    B() {cout<< " B() ";}
    virtual ~B() {cout<< " ~B ";}
    virtual void g() const {cout <<" B::g ";}
    virtual const B* j() {cout <<" B::j "; n(); return this;}
    void k() {cout <<" B::k "; j(); m();}
    void m() {cout <<" B::m "; g(); j();}
    virtual A& n() {cout <<" B::n "; return *this;}
};

class D: virtual public B {
public:
    D() {cout<< " D() ";}
    ~D() {cout<< " ~D ";}
    virtual void g() {cout <<" D::g ";}
    const B* j() {cout <<" D::j "; return this;}
    void k() const {cout <<" D::k "; k();}
    void m() {cout <<" D::m "; g(); j();}
};
```

delete p1; A -> ~A() [stop]
delete p4; B -> ~B ~D ~C
~B ~A

Le precedenti definizioni compilano correttamente. Per ognuna delle seguenti istruzioni scrivere nell'apposito spazio:

- **NON COMPILA** se la compilazione dell'istruzione provoca un errore;
- **UNDEFINED** se lo statement compila correttamente ma la sua esecuzione provoca un undefined behaviour o un errore run-time;
- se l'istruzione compila correttamente e non provoca errori a run-time allora si scriva la stampa che l'esecuzione produce in output su cout; se non provoca alcuna stampa allora si scriva **NESSUNA STAMPA**.

```
(p1->j())->k(); .....
(dynamic_cast<const E*>(p1->j()))->g(); .....
p2->m(); .....
(p2->j())->g(); .....
p3->k(); .....
(p4->n()).m(); .....
((dynamic_cast<D*>(p4))->n()).k(); .....
(dynamic_cast<E*>(p5))->j(); .....
(dynamic_cast<E*>(const_cast<B*>(p6)))->k(); .....
new E(*p7); .....
delete p1; .....
delete p4; .....
```

Laurea in Informatica – Programmazione ad Oggetti – Appello d'Esame 25/01/2024

Nome..... Cognome..... Matricola.....

Esercizio Cosa Stampa

```

class A {
public:
    A() {cout<< " A() ";}
    ~A() {cout<< " ~A ";}
    A(const A& x) {cout<< " Ac ";}
    virtual const A* j() {cout<<" A::j "; return this;}
    virtual void k() {cout <<" A::k "; m();}
    void m() {cout <<" A::m "; j();}
};

class C: virtual public B {
public:
    C() {cout<< " C() ";}
    ~C() {cout<< " ~C ";}
    void g() const {cout <<" C::g ";}
    void k() override {cout <<" C::k "; B::n();}
    virtual void m() {cout <<" C::m "; g(); j();}
    B& n() override {cout <<" C::n "; return *this;}
};

class E: public C, public D {
public:
    E() {cout<< " E() ";}
    ~E() {cout<< " ~E ";}
    E(const E& x) {cout<< " Ec ";}
    virtual void g() const {cout <<" E::g ";}
    const E* j() {cout <<" E::j "; return this;}
    void m() {cout <<" E::m "; g(); j();}
    D& n() final {cout <<" E::n "; return *this;}
};

class B: virtual public A {
public:
    B() {cout<< " B() ";}
    virtual ~B() {cout<< " ~B ";}
    virtual void g() const {cout <<" B::g ";}
    virtual const B* j() {cout <<" B::j "; n(); return this;}
    void k() {cout <<" B::k "; j(); m();}
    void m() {cout <<" B::m "; g(); j();}
    virtual A& n() {cout <<" B::n "; return *this;}
};

class D: virtual public B {
public:
    D() {cout<< " D() ";}
    ~D() {cout<< " ~D ";}
    virtual void g() {cout <<" D::g ";}
    const B* j() {cout <<" D::j "; return this;}
    void k() const {cout <<" D::k "; k();}
    void m() {cout <<" D::m "; g(); j();}
};
    
```

STAMP C B::B1 = 10.

```

A* p1 = new E(); B* p2 = new C(); A* p3 = new D(); B* p4 = new E();
const A* p5 = new D(); const B* p6 = new E(); const E* p7 = new E();
    
```

Le precedenti definizioni compilano correttamente. Per ognuna delle seguenti istruzioni scrivere nell'apposito spazio:

- **NON COMPILA** se la compilazione dell'istruzione provoca un errore;
- **UNDEFINED** se lo statement compila correttamente ma la sua esecuzione provoca un undefined behaviour o un errore run-time;
- se l'istruzione compila correttamente e non provoca errori a run-time allora si scriva la stampa che l'esecuzione produce in output su cout; se non provoca alcuna stampa allora si scriva **NESSUNA STAMPA**.

```

(p1->j())->k(); .....
(dynamic_cast<const E*>(p1->j()))->g(); .....
p2->m(); .....
(p2->j())->g(); .....
p3->k(); .....
(p4->n()).m(); .....
((dynamic_cast<D*>(p4))->n()).k(); .....
(dynamic_cast<E*>(p5))->j(); .....
(dynamic_cast<E*>(const_cast<B*>(p6)))->k(); .....
new E(*p7); .....
delete p1; .....
delete p4; .....
    
```