Costruttore di Copia Standard

Q: Come si implementa correttamente un costruttore di copia standard?

A: Il costruttore di copia standard deve:

1. Richiamare i costruttori di copia delle classi base:

```
class Derived: public Base1, public Base2 {
public:
    Derived(const Derived& d):
        Base1(d),  // Richiama costruttore copia di Base1
        Base2(d),  // Richiama costruttore copia di Base2
        member(d.member)  // Copia dei membri
    {}
};
```

2. Con ereditarietà virtuale:

```
class E: public virtual Base {
public:
    E(const E& e):
        Base(e),    // Richiama costruttore della base virtuale
        member(e.member)
    {}
};
```

3. Con puntatori:

```
class F {
private:
    T* ptr;
public:
    // Copia superficiale
    F(const F& f): ptr(f.ptr) {}

    // Copia profonda
    F(const F& f):
        ptr(f.ptr != nullptr ? new T(*f.ptr) : nullptr)
    {}
};
```

Operatore di Assegnazione Standard

Q: Come si implementa correttamente l'operatore di assegnazione standard?

A: L'operatore di assegnazione standard deve:

1. Pattern base:

```
Class& operator=(const Class& other) {
    if(this != &other) {  // Controllo auto-assegnazione
        // Assegnazione membri
    }
    return *this;
}
```

2. Con ereditarietà (NB - NON NECESSARIO da soluzioni controllare this != something):

```
Derived& operator=(const Derived& d) {
        Base1::operator=(d);  // Assegnazione base classes
        Base2::operator=(d);
        member = d.member;    // Assegnazione membri
    return *this;
}
```

3. Con ereditarietà virtuale (assicura la costruzione degli oggetti UNA VOLTA SOLA):

```
E& operator=(const E& e) {
        Base::operator=(e);  // Base virtuale
        member = e.member;
    return *this;
}
```

Metodo di Clonazione

Q: Come si implementa correttamente un metodo di clonazione?

A: Il metodo di clonazione deve:

1. Pattern base:

```
class Base {
public:
    virtual Base* clone() const {
```

```
        return new Base(*this);
    }
};
```

2. Con ereditarietà:

```
class Derived: public Base {
public:
    virtual Derived* clone() const override {
        return new Derived(*this);
    }
};
```

3. Con copia profonda:

```
class F {
public:
    virtual F* clone() const {
        // Usa il costruttore di copia profonda
        return new F(*this);
    }
};
```

Best Practices

1. Costruttore di Copia:
    - Copiare tutti i membri della classe
    - Gestire correttamente i puntatori (shallow vs deep copy)
    - Richiamare i costruttori delle classi base nell'ordine corretto
2. Operatore di Assegnazione:
    - Proteggere da auto-assegnazione
    - Deallocare memoria esistente se necessario
    - Copiare tutti i membri
    - Restituire reference a *this
3. Clonazione:
    - Rendere il metodo virtuale
    - Usare covariant return type nelle classi derivate
    - Garantire una copia completa dell'oggetto
    - Considerare se serve copia profonda o superficiale