**11° lezione programmazione II**

**Ereditarietà in C++**

**L’OOP** offre la possibilità di estendere le classi già esistenti. In sostanza attraverso le classi definite in precedenza si possono creare nuove classi mantenendo gli attributi e i metodi delle classi definite in precedenza(classe generatrice). **La classe che viene ereditata si chiama sottoclasse. La classe generatrice si chiama sovra classe.** Queste relazioni individuano una gerarchia che si può descrivere usando un grafo di gerarchia.

Immagine che contiene testo, schermata, Carattere, linea

Descrizione generata automaticamente

La nuova classe si differenzia dalla sopra classe in due modi:

• **Per estensione**: aggiungendo nuovi attributi e metodi

**• Per ridefinizione**: modificando i metodi ereditati,specificando una implementazione diversa di un metodo (override(modificare il corpo della funzione già definita in precedenza), overload(aggingere parametri alle funzioni già definite in precedenza)).

Immagine che contiene testo, linea, schermata, diagramma

Descrizione generata automaticamente

Vi sono 2 tipi di ereditarietà:

1. **Eredità singola**

Immagine che contiene testo, schermata, Carattere, linea

Descrizione generata automaticamente

1. **Eredità doppia**

Immagine che contiene testo, schermata, Carattere, linea

Descrizione generata automaticamente

Le sottoclassi hanno la possibilità di ridefinire i metodi ereditati (mantenendo lo stesso nome) oppure lasciarli inalterati perché già soddisfacenti.

**Cos’è il polimorfismo?**

**Il polimorfismo** indica la possibilità per i metodi di assumere implementazioni, diverse all’interno della gerarchia delle classi. **Esempio:** tutti i veicoli a motore possiedono il metodo ‘’accelera’’. Le sottoclassi ‘’automobile’’ e ‘’moto’’ è probabile che lo ridefiniscano per adeguarlo alle particolari esigenze (es. pedale vs. manopola).

Se si considera l’esempio di sopra in run-time un’istanza della classe ‘’veicoli a motore’’ può rappresentare sia un’ ‘’automobile’’ che una ‘’moto’’, ma quando viene richiesta l’attivazione del metodo ‘’accelera’’ è **importante garantire che, tra tutte le implementazioni, venga scelta quella corretta**.

**Qual è lo strumento che si utilizza per il polimorfismo?**

**Il collegamento dinamico** è lo strumento utilizzato per la realizzazione del polimorfismo. È dinamico perché l’associazione tra l’oggetto e il metodo corretto da eseguire è effettuata a run-time.

**È importante sottolineare che se una classe A eredita da una classe B, possiamo anche dire che una classe derivata B estende le funzionalità della classe base A**.

**Se una classe derivata non fornisce nessuna estensione allora probabilmente abbiamo un errore nel design della classe.**

**Come si dichiara una classe derivata?**

**Immagine che contiene testo, schermata, Carattere, bianco

Descrizione generata automaticamente**

Lo specificatore può essere:

* **private(privato)**
* **public(pubblico)**
* **protected (protetto)**

**Livelli di accesso**

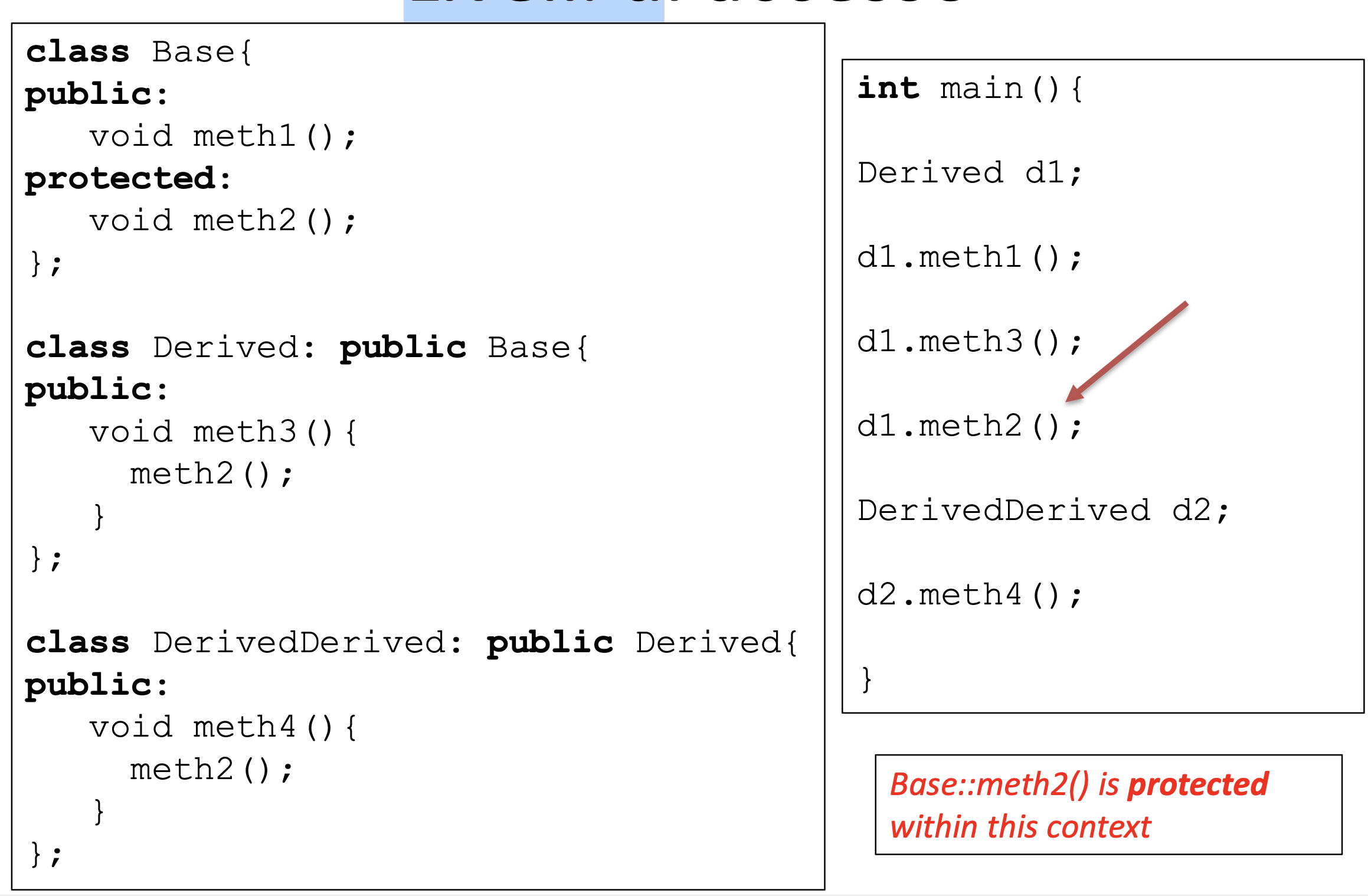
Si può dire che una classe ha due interfacce diverse per due categorie di classi:

* Ha una interfaccia **public** per fornire servizi a classi non collegate mediante ereditarietà.
* Ha una interfaccia **protected** per fornire servizi alle classi derivate. **In questo caso gli attributi con protected: sono visibili alla sottoclasse**
* una classe marcata come final (posto dopo il nome della classe) non può essere derivata.
* Una classe derivata in modo **public** eredita **gli attributi e i metodi pubblici e protetti della classe e ne mantiene tale livello di accesso;**
* Una classe derivata in modo **protected** eredita **gli attributi e i metodi pubblici e protetti della classe e li espone con un livello di accesso protected**;
* Una classe derivata in modo **private** **eredita i metodi e gli attributi pubblici e protetti della classe e li espone con un livello di accesso private**, per cui non li rende fruibili al di fuori della classe stessa**.**

**Immagine che contiene testo, schermata, Carattere, numero

Descrizione generata automaticamente**

In generale, è bene fare si che i **dati membro della classe base siano privati** ma con **metodi pubblici per accedervi**. Se vogliamo che i **membri di una classe siano visibili ai metodi di una classe derivata** **dobbiamo dichiararli protetti**. **In questo modo** **questi membri protetti** **rimarranno accessibili nella gerarchia, a meno che l’ereditarietà non sia privata.**

****

**Costruttori delle classi derivate**

Quando si **crea un oggetto** da una **classe derivata i costruttori di ogni classe da cui si deriva sono invocati in sequenza**, **fino ad arrivare al costruttore della classe più derivata**.

Immagine che contiene testo, Carattere, schermata, numero

Descrizione generata automaticamenteImmagine che contiene testo, Carattere, schermata, bianco

Descrizione generata automaticamente

Siccome D eredità da B è come se si sta chiamando due volte B.

**Distruttore delle classi derivate**

**Per i distruttori avviene l’invocazione di tutti i distruttori delle classi all’interno della gerarchia ma in ordine inverso**. **Quindi il distruttore della classe derivata è il primo, a seguire tutti i distruttori delle classi da cui si deriva.** I distruttori non possono essere sovraccaricati, quindi non c’è nessun problema nell’identificare quale metodo deve essere chiamato.

**Quante tipologie di polimorfismo esistono in C++?**

Ci sono diverse tipologie di polimorfismo in C++:

**1. Over loading.**

**2. Subtyping.**

**3. Programmazione generica.**

**Il metodo più utilizzato nella OOP è il secondo**. **Ci si può riferire ad un’istanza di una classe derivata come se fosse l’istanza della sua super-classe (base), ma ogni oggetto risponde alle chiamate ai metodi come specificato dal suo vero tipo.**

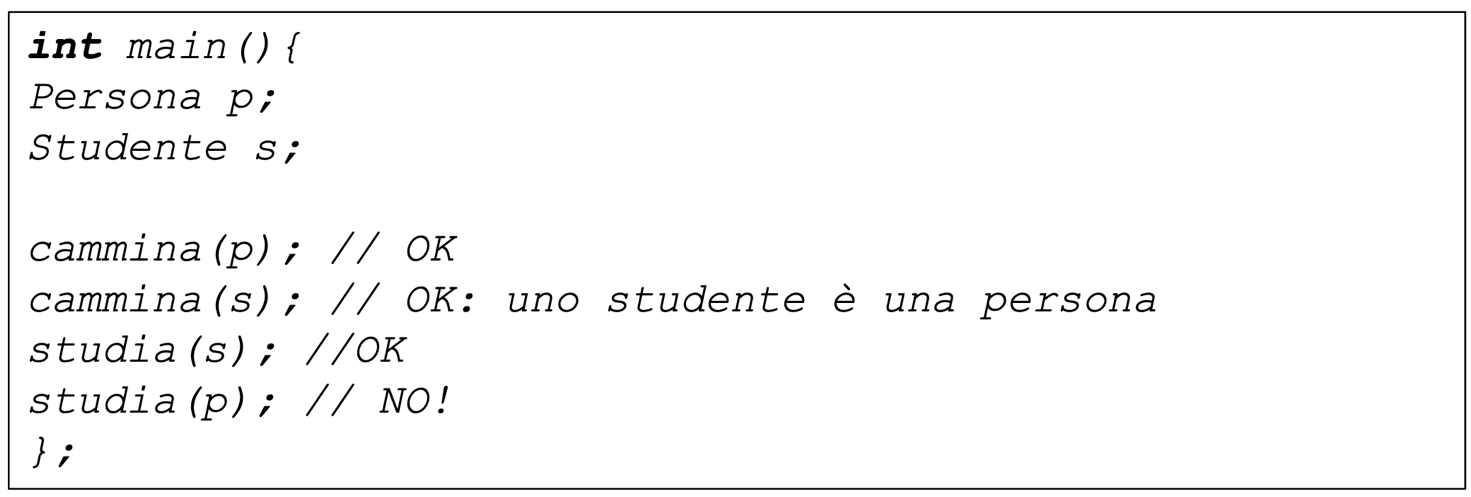
**Il principio di sostituibilità di Liskov (Liskov Substitution Principle - LSP))**

è stato formulato da Barbara Liskov e Jeannette Wing. Esso stabilisce una regola fondamentale per l'ereditarietà e il polimorfismo, sottolineando la necessità che le classi derivate siano sostituibili senza alterare il corretto funzionamento del programma.

In modo intuitivo, il principio afferma che **se S è un sotto-tipo di T, allora gli oggetti di tipo T possono essere sostituiti da oggetti di tipo S senza influire sulla correttezza del programma.** **Questo significa che un oggetto istanziato da una classe derivata dovrebbe essere in grado di sostituire senza problemi un oggetto istanziato dalla classe base in tutti i contesti in cui la classe base è utilizzata, senza che il programma si comporti in modo imprevisto o sbagliato**.

Immagine che contiene testo, schermata, Carattere, linea

Descrizione generata automaticamente



Cos’è il binding

**ll "binding**" in programmazione si riferisce **all'associazione tra identificatori (come nomi di variabili o funzioni) e le relative entità (come variabili, funzioni o metodi).** Esistono due tipi principali di binding:

* **Binding statico:** Il collegamento avviene in fase di compilazione del programma. Ciò significa che **le associazioni tra gli identificatori e le entità vengono stabilite a priori, durante la compilazione, e rimangono fisse durante l'esecuzione del programma**. Questo tipo di binding è tipico dei linguaggi di programmazione **staticamente tipizzati, come C++ e Java**. **Ad esempio**, quando si chiama una funzione in C++, **il compilatore determina quale funzione chiamare basandosi sul tipo statico dei parametri passati**.
* **Binding dinamico: La connessione avviene durante l'esecuzione del programma**. **In questo caso, il codice da eseguire viene determinato solo al momento della chiamata effettiva della funzione o del metodo**. Questo tipo di binding è tipico dei linguaggi di programmazione orientati agli oggetti e supporta il polimorfismo. Ad esempio, in C++ con l'utilizzo di puntatori o riferimenti polimorfi, il binding dinamico avviene quando si chiama un metodo su un oggetto tramite un puntatore o un riferimento alla classe base, e la decisione su quale metodo chiamare viene presa in base al tipo reale dell'oggetto durante l'esecuzione.

**Vantaggi e svantaggi del binding**

* **Vantaggio principa**le del binding dinamico: offre un alto grado di flessibilità e praticità nella gestione delle gerarchie di classi
* **Svantaggio principale:** è meno efficiente di quello statico

I linguaggi più strettamente OOP offrono solo binding dinamico, in C++ il binding per default è quello statico. Per specificare il binding dinamico si fa precedere la dichiarazione della funzione dalla parola riservata **virtual.**

**Funzioni virtuali**

Immagine che contiene testo, Carattere, schermata, linea

Descrizione generata automaticamente

**virtual** anteposto alla dichiarazione di una funzione indica al compilatore che questa può essere definita in una classe derivata.

Ogni classe derivata **deve definire le sue proprie versioni delle funzioni dichiarate virtuali nella classe base**: **se le classi Cerchio e Rettangolo derivano dalla classe Figura, debbono entrambe definire le funzioni membro calcolare\_area() e disegnare().**

Immagine che contiene testo, schermata, Carattere

Descrizione generata automaticamente

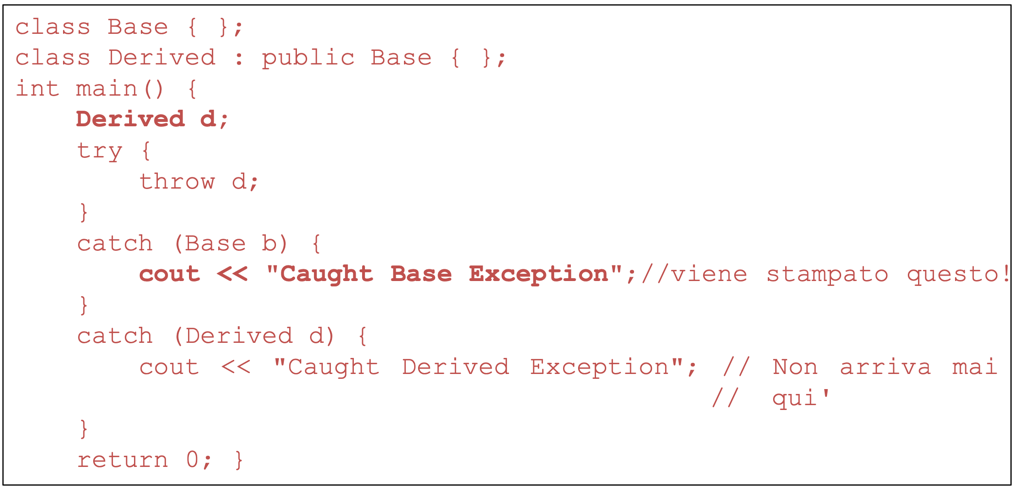
Linee guida da seguire quando si definisce una classe polimorfica (cioè che ha almeno un metodo virtuale):

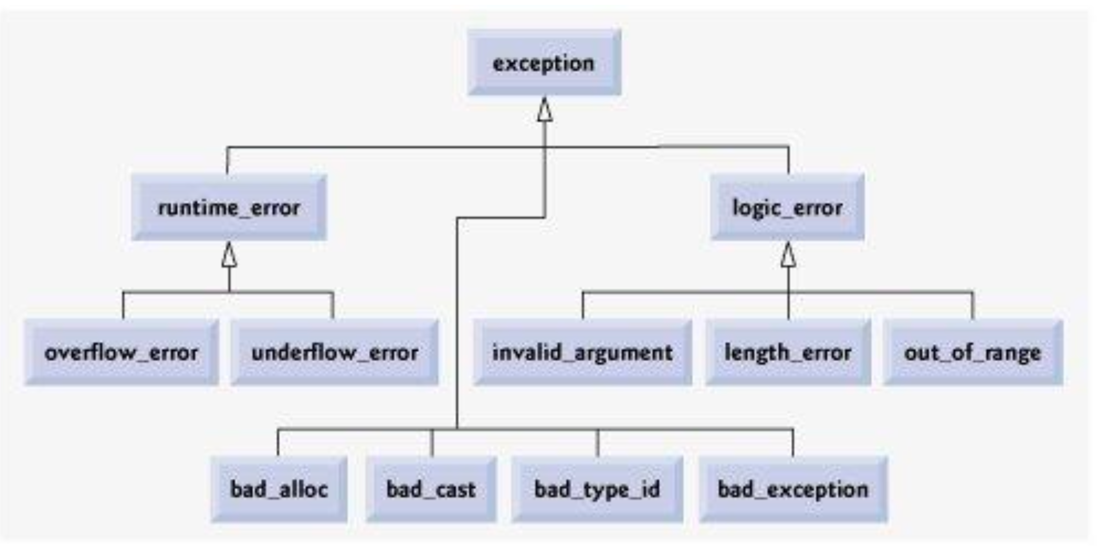
* **Almeno un metodo virtuale deve essere indicato come tale nella classe base.**
* **virtual si usa solo nella dichiarazione del metodo, non nella definizione.**
* **Una funzione top-level (una funzione non associata ad una classe) non può essere virtuale.**
* **Un metodo statico non può essere virtuale.**
* **final indica che un metodo non può essere riscritto nelle classi derivate.**
* **Il distruttore di una classe base dovrebbe sempre essere dichiaratoImmagine che contiene testo, schermata, linea, Carattere

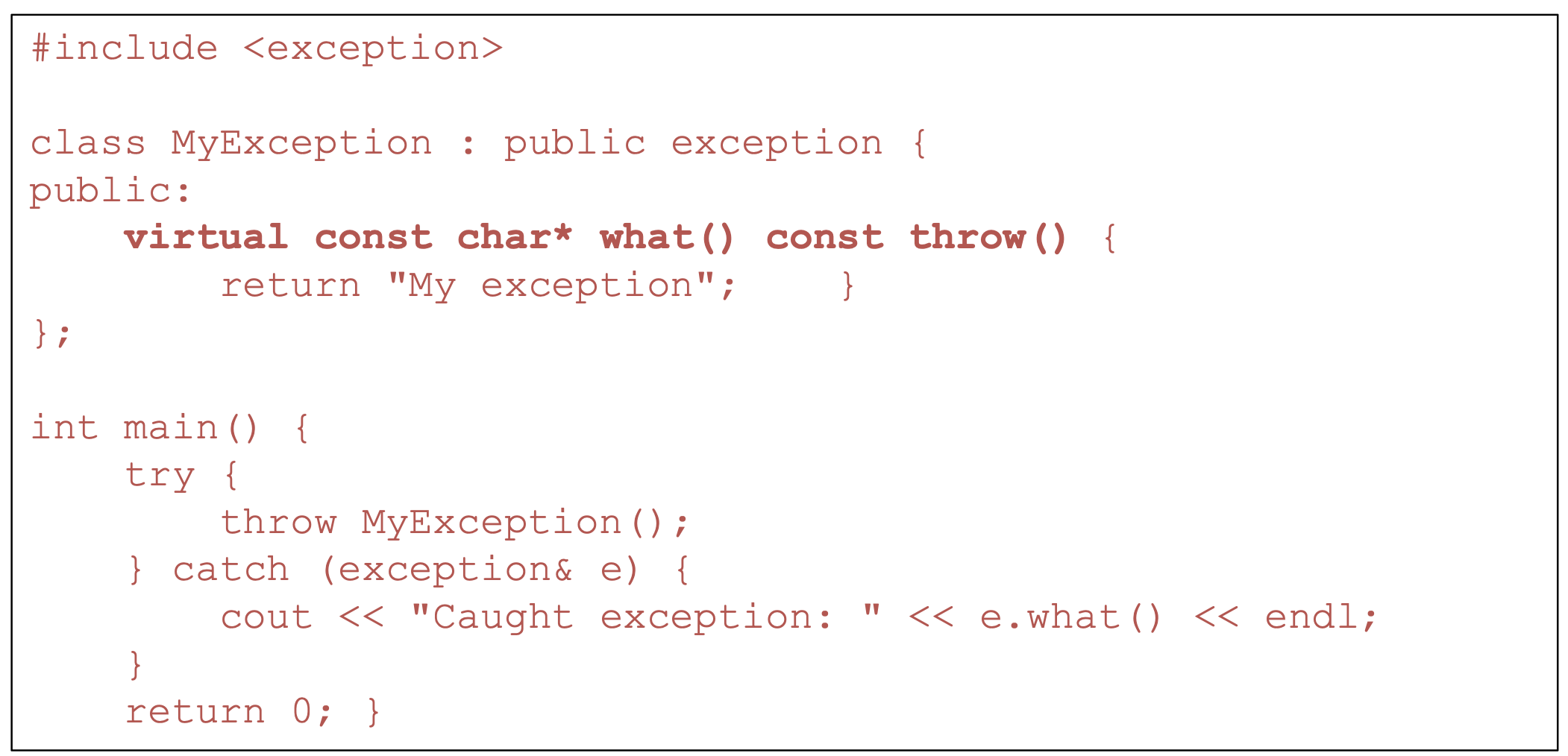
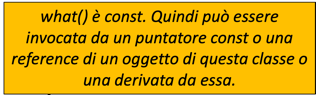
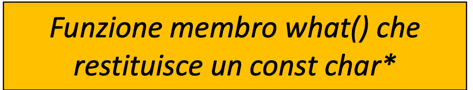
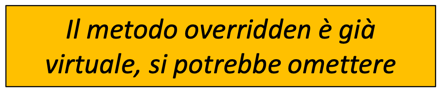
  Descrizione generata automaticamente**

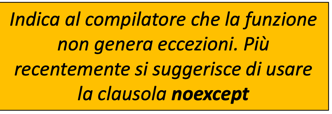
**Gestione delle eccezioni**

Se mettiamo in cascata più clausole catch che tengono conto di una gerarchia di classi bisogna specificare prima quelle derivate, altrimenti verrà sempre catturata la ‘’versione’’ base dell’eccezione/classe.









**Classi astratte**

**In questo modo si definisce una funzione puramente virtuale**



Una classe **astratta** è una classe **che contiene almeno un metodo puramente virtuale**. Come conseguenza, **non possiamo istanziare oggetti di una classe astratta**. Queste infatti **vengono usate come strumenti di specifica delle funzionalità richieste all’interno di una gerarchia di classi (cioè per definire interfacce).**

**Una classe derivata da una classe astratta deve necessariamente fare l’override di tutti i metodi puramente virtuali ereditati per poter essere istanziata**.

**Ereditarietà multipla**

Immagine che contiene diagramma, testo, linea, Piano

Descrizione generata automaticamente

**Una classe può ereditare attributi e comportamento di più di una classe base.**

Immagine che contiene diagramma, linea

Descrizione generata automaticamente **Problema del diamante**

In questo caso B e C ereditano dalla classe A e la classe D eredita sia da B che da C.

Se un metodo in D chiama un metodo definito in A, da quale classe viene ereditato?

**Soluzione**

* **C++, per default, segue ogni percorso** (di ereditarietà) **separatamente; quindi, D conterrà due** (separati) **oggetti di A**
* **Soluzione:** se le ereditarietà da A ad B (e da A ad C) sono “virtual”, C++ crea un solo oggetto A
* L’effetto della parola chiave virtual in una clausola di derivazione è quello di **forzare il compilatore a includere la base virtuale una sola volta nella definizione degli oggetti derivati, anche se essa appare più volte nella catena di derivazione.** **In questo modo si ottimizza l’uso delle risorse, e si risolvono a monte eventuali conflitti di nomi**.

Immagine che contiene testo, schermata, Carattere, numero

Descrizione generata automaticamente