Polimorfismo

Alberto Ferrari

Polimorfismo e Funzioni Virtuali

- **Polimorfismo** e **funzioni virtuali** sono un meccanismo fondamentale per realizzare sistemi estensibili
- Consentono di trattare gli oggetti di tutte le classi di una gerarchia come se fossero oggetti della classe base
- Risultato: scrittura di programmi più semplici, in cui viene favorito il testing ed il mantenimento del codice

Funzioni Virtuali

- Quando una funzione membro di una classe base è dichiarata **virtuale**
  - Se la funzione è chiamata tramite un oggetto, la risoluzione del riferimento avviene a tempo di compilazione (**static binding**) e la funzione chiamata è quella della classe dell'oggetto
  - Se per chiamare la funzione viene utilizzato un puntatore o un riferimento, il programma sceglie a tempo di esecuzione la funzione della classe appropriata (**late binding o dynamic binding**)

Alberto Ferrari

Funzioni Virtuali

- Per specificare che una funzione è virtuale si include il modificatore **virtual** nella dichiarazione (non nella definizione)
- Quando la definizione di una funzione virtuale viene cambiata in una classe derivata si parla di **overriding**
- La proprietà di essere una funzione virtuale viene ereditata lungo tutta la gerarchia

Alberto Ferrari

Polimorfismo

- Capacità di oggetti appartenenti a classi che derivano da una classe base comune di rispondere in modi diversi alla chiamata di una certa funzione
- Si implementa tramite le funzioni virtuali: alla chiamata di una funzione virtuale tramite un puntatore o un riferimento alla classe base, il programma sceglie la ridefinizione corretta della funzione nella classe derivata appropriata

Polimorfismo (2)

- Aumento di generalità: è il runtime a doversi occupare delle specificità, non il programmatore
- Estendibilità: il codice è scritto indipendentemente dai tipi derivati, nuovi tipi possono essere aggiunti senza dover apportare modifiche a quanto già sviluppato

Polimorfismo (3)

- Se una funzione non virtuale è ridefinita in una classe derivata **non si ha polimorfismo**: se viene chiamata tramite un puntatore alla classe base, è utilizzata la versione della classe base, se viene chiamata tramite un puntatore alla classe derivata, viene utilizzata la versione della classe derivata

Alberto Ferrari

The Slicing Problem

- Assegnando un oggetto di una classe derivata a una variabile della sua classe base, le variabili e funzioni membro aggiunte dalla classe derivata vengono perse
- Il problema può essere risolto usando funzioni virtuali e puntatori

Alberto Ferrari

Overriding e Overloading

- Il meccanismo di **overriding** è concettualmente molto diverso da quello di **overloading**, e non deve essere confuso con esso
- L'overloading consente di definire in una stessa classe più metodi aventi lo stesso nome, ma che differiscano nella *signature*. L'overriding, invece, consente di ridefinire un metodo in una sottoclasse: il metodo originale e quello che lo ridefinisce hanno necessariamente la stessa firma, e solo a tempo di esecuzione si determina quale dei due deve essere eseguito

Alberto Ferrari

Implementazione delle Funzioni Virtuali

- Se una classe ha una o più funzioni membro virtuali il compilatore crea una tabella che per ogni funzione virtuale contiene l’indirizzo in memoria del codice della funzione
- Quando viene creato un oggetto della classe, la sua descrizione contiene un puntatore alla tabella delle funzioni virtuali
- Quando una funzione virtuale viene chiamata usando un puntatore all’oggetto, il sistema runtime usa la tabella invece del tipo del puntatore per decidere quale definizione della funzione usare
- Le funzioni virtuali introducono **overhead**

Alberto Ferrari

Classi Astratte e Funzioni Virtuali Pure

- Se una funzione è **virtuale pura** la sua definizione non è necessaria
- Una classe con una o più funzioni virtuali pure è detta **astratta**
- Una classe astratta può essere usata solo come classe base per derivare altre classi, non si possono creare oggetti

Alberto Ferrari

Ereditarietà di Interfaccia e di Implementazione

- Con l'ereditarietà pubblica si può ereditare:
  - La sola interfaccia (funzioni virtuali pure)
  - L'interfaccia più
    - Un'implementazione di default (funzioni virtuali)
    - Un'implementazione obbligatoria (funzioni non virtuali)
- È bene non ridefinire le funzioni ereditate non virtuali

Alberto Ferrari

Distruttori delle Classi Base

- Se si dealloca un oggetto di una classe derivata attraverso un puntatore alla classe base e la classe base ha un distruttore non virtuale, il risultato è indefinito
- **È bene dichiarare virtuali i distruttori delle classi base**

Alberto Ferrari

Funzioni Virtuali e Parametri di Default

- Quando si eredita una funzione virtuale con un parametro di default, è bene non cambiarne il valore
- Motivazione
  - Le funzioni virtuali usano dynamic binding
  - I parametri di default usano static binding

Alberto Ferrari

Upcasting e Downcasting

- Il casting da un tipo discendente a un tipo antenato è detto **upcasting** ed è sicuro
- Il casting da un tipo antenato a un tipo discendente è detto **downcasting** ed è pericoloso

Alberto Ferrari

Dynamic Cast

- Consente di fare un downcasting sicuro
- Va usato su puntatori
- Se il cast riesce (cioè se il tipo dinamico corrisponde al tipo che si vuole ottenere), restituisce un puntatore al nuovo tipo, altrimenti restituisce **NULL**

Alberto Ferrari