Eccezioni

Alberto Ferrari

Gestione delle Eccezioni

- C++ fornisce strumenti per gestire situazioni eccezionali
- Terminologia
  - Sollevare un'eccezione **(to throw an exception)** = segnalare una situazione eccezionale
  - Intercettare l'eccezione **(to catch or handle the exception)** = catturare o gestire la situazione eccezionale

Costrutto try-throw-catch

```
try
{
  Some_Statements
  if (Exceptional_Case)
    throw exception;
  Some_More_Statements
}
catch(Type e)
{
  Code to be performed if a value of type Type is thrown in the try
    block
}
```

Alberto Ferrari

Blocco try

- Contiene il codice per gestire le situazioni normali
- Può riconoscere e segnalare situazioni speciali sollevando eccezioni
- Se non si verificano eccezioni, l'esecuzione del blocco try è quella standard
- E' buona norma inserire in un blocco try il solo codice che potenzialmente può sollevare un'eccezione

Alberto Ferrari

Istruzione throw

- Usata per “lanciare” un valore detto eccezione
- Il valore lanciato può essere di qualsiasi tipo
- L’esecuzione del blocco **try** termina e il controllo passa a un blocco **catch**

Alberto Ferrari

Blocco catch

- Contiene il codice per gestire la situazione eccezionale
- Ha un parametro che
  - Specifica quale tipo di valore può essere intercettato dal blocco
  - Consente di utilizzare il valore intercettato all’interno del blocco
- Se non viene sollevata nessuna eccezione l’esecuzione del blocco **try** viene completata e il blocco **catch** viene ignorato
- Una volta completato il blocco **catch**, viene eseguito il codice che segue
- Un blocco **catch** risponde solo a un blocco **try** immediatamente precedente
- Se non c’è un blocco **catch** del tipo opportuno il programma termina

Alberto Ferrari

Classi di Eccezioni

- Sono classi i cui oggetti contengono l'informazione che si vuole lanciare al blocco **catch**
- In questo modo si ottiene un diverso tipo per ogni possibile situazione eccezionale
- Può essere utile definire una gerarchia di classi di eccezioni

Alberto Ferrari

Eccezioni Multiple

- Un blocco **try** può potenzialmente sollevare più eccezioni di tipi diversi
- In ogni esecuzione verrà sollevata al massimo una eccezione
- Ogni blocco **catch** può intercettare valori di un solo tipo
- Si possono avere più blocchi **catch** dopo un blocco **try** per gestire eccezioni di tipo diverso

Alberto Ferrari

Blocco catch di Default

- Quando in un blocco **try** viene sollevata un'eccezione, i blocchi **catch** che seguono sono considerati in ordine, viene eseguito il primo che intercetta quel tipo di eccezione
- Blocco **catch** speciale che intercetta ogni tipo di eccezione, da usare come default

```
catch(…) { cout << "Unexplained exception";}
```

Alberto Ferrari

Sollevare un'Eccezione in una Funzione

- Spesso è utile ritardare la gestione di un'eccezione
- Una funzione può sollevare un'eccezione e non intercettarla
- Sarà il programma che usa la funzione a gestire l'eccezione
- Il programma metterà la chiamata della funzione in un blocco **try** seguito da un blocco **catch** che intercetta l'eccezione

Alberto Ferrari

Specifica delle Eccezioni

- Elenca le eccezioni che possono essere sollevate da una funzione e non vengono da essa intercettate
- Deve apparire sia nella dichiarazione che nella definizione della funzione
- Se viene sollevata un'eccezione che non viene intercettata e non compare nella specifica, viene chiamata la funzione **unexpected** che per default termina il programma, ma può essere ridefinita

Specifica delle Eccezioni ed Ereditarietà

- Se B->D, un'eccezione di tipo D viene gestita come se fosse di tipo B, anche rispetto alla specifica
- Quando si dà una nuova definizione o si fa overriding di una funzione ereditata, non si possono aggiungere eccezioni alla specifica, ma se ne possono eliminare

Quando Sollevare un'Eccezione

- Se la funzione è in grado di gestire in modo semplice il caso speciale, non deve sollevare l'eccezione
- Se invece il modo in cui il caso speciale va gestito dipende da dove la funzione è usata, si delega la gestione al livello superiore (le eccezioni non intercettate risalgono di scope)
- Non usare le eccezioni nei distruttori

Alberto Ferrari

Esempio: Allocazione Dinamica

```
#include <new>
using std::bad_alloc;

try
{
  int *p = new int[100];
}
catch(bad_alloc)
{
  cout << "Cannot alloc p";
   …
}
```

Alberto Ferrari

Abuso delle Eccezioni

- La gestione delle eccezioni genera overhead sia temporale che spaziale
- Le istruzioni **throw** rendono contorto il flusso di controllo (come le istruzioni di salto)
- La gestione delle eccezioni va usata con moderazione

Alberto Ferrari

Vantaggi della Gestione delle Eccezioni

- In un linguaggio che non la supporta si può restituire un codice di errore
  - Occorre controllarlo ogni volta
  - Il programma può ignorarlo
  - Alcune funzioni non possono restituire un codice di errore
- In C++
  - Gestione uniforme delle eccezioni per tutte le funzioni
  - L'eccezione non può essere ignorata
  - Non si mescola la gestione del caso speciale e dei casi normali
  - Migliora la gestione delle variabili locali

Alberto Ferrari