Lezione 07: Eccezioni

Gestione degli Errori Non Locali

La gestione degli errori non sempre può essere risolta localmente in una singola funzione. In molti casi, è necessario separare la gestione degli errori in **diverse parti** del programma, a causa di:

- **Librerie**: errori possono essere individuati in una libreria, ma non è sempre possibile gestirli lì. La libreria può non conoscere il contesto o la soluzione.
- **Individuazione e Gestione Separata**: un errore può essere identificato da una parte del programma (come una libreria), ma la **gestione** deve avvenire in un altro punto (dove l'errore è più pertinente).

Questo approccio è particolarmente utile per **programmi grandi e complessi**, dove l'esecuzione si prolunga nel tempo.

Le Eccezioni in C++

Le **eccezioni** sono uno strumento che permette di gestire gli errori separando la parte di **individuazione** dell'errore dalla parte di **gestione**.

- Una funzione che non può risolvere un errore **lancia** (throws) un'eccezione.
- Un componente che chiama una funzione **indica** tramite un meccanismo `try-catch` quali eccezioni è in grado di gestire.

Meccanismo Try-Catch

Il meccanismo **try-catch** consente di **lanciare** e **catturare** le eccezioni. Di seguito forniamo un esempio di utilizzo:

```
void taskmaster() {
    try {
        auto result = do_task();  // esegue il task
        // usa i risultati
    }
    catch (const Some_error& e) {
        // gestisci il problema
```

```
        std::cerr << "Errore: " << e.what() << std::endl;
    }
}

int do_task() {
    if (/* in grado di eseguire il task */)
        return result;
    else
        throw Some_error();  // lancia l'errore
}
```

In questo esempio, la funzione `do_task` potrebbe **lanciare** un'eccezione se non riesce a completare il suo compito. La funzione `taskmaster` gestisce l'errore **catturando** l'eccezione tramite il blocco `catch`.

Caratteristiche delle Eccezioni

- Le eccezioni sono **oggetti** che vengono "lanciati" come rappresentazione dell'errore.
- Può essere di qualsiasi tipo **copiabile**, ma si consiglia di usare un tipo personalizzato per evitare conflitti tra le eccezioni generate da diverse librerie.

La libreria standard (`std`) definisce una **gerarchia di eccezioni**, ognuna delle quali rappresenta una diversa tipologia di errore.

Gerarchia delle Eccezioni nella Libreria Standard

- **`logic_error`**: errori che possono essere individuati prima dell'esecuzione o tramite test sugli argomenti delle funzioni.
  - `length_error`
  - `domain_error`
  - `out_of_range`
  - `invalid_argument`
  - `future_error`
- **`runtime_error`**: errori che si verificano durante l'esecuzione del programma.

- `range_error`
- `overflow_error`
- `underflow_error`
- `system_error`
- `bad_exception`
- `bad_alloc`
- `bad_typeid`
- `bad_cast`

Gestione Tradizionale degli Errori Senza Eccezioni

Prima delle eccezioni, le strategie per gestire gli errori includevano vari metodi meno eleganti:

- **Terminare il programma**: interrompere l'esecuzione in caso di errore.
- **Restituire un valore di errore**: alcune funzioni restituivano un valore speciale per indicare l'errore. Tuttavia, questo approccio non sempre è praticabile (ad esempio quando non ci sono valori disponibili da restituire).
- **Stato di errore**: restituire valori legali ma lasciare il programma in uno stato di errore, richiedendo che il programma verifichi continuamente l'errore.
- **Funzioni di gestione degli errori**: chiamare funzioni specifiche per gestire l'errore, ma senza un vero e proprio meccanismo integrato.

Vantaggi delle Eccezioni

- Separano la **logica di individuazione** dell'errore dalla **gestione** dell'errore.
- Permettono di **propagare l'errore** da una funzione all'altra senza dover verificare manualmente l'errore in ogni passaggio.
- Forniscono una **struttura chiara** per il trattamento degli errori in programmi complessi.