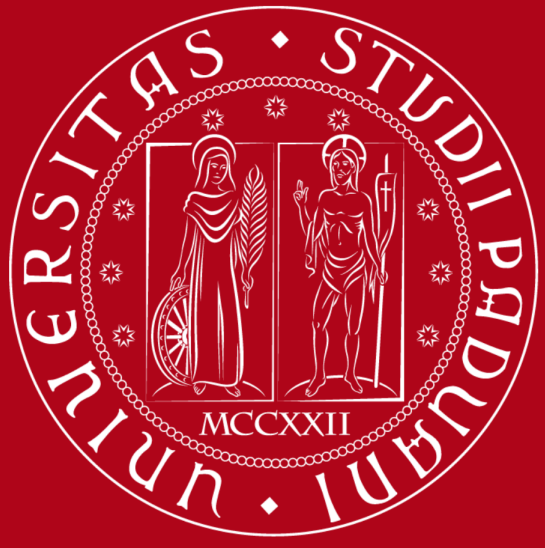


UNIVERSITÀ
DEGLI STUDI
DI PADOVA

Classe Date:
Rilevare gli errori

Stefano Ghidoni

DIPARTIMENTO
DI INGEGNERIA
DELL'INFORMAZIONE

Agenda

- Verifica degli errori
- Eccezioni

Potremmo trovarci in una situazione in cui l'utente non inserisce dati accettabili

Dati non validi

- La classe è più intelligente se è in grado di capire se il suo stato è valido o meno
- Classi ben progettate non entrano mai in uno stato non valido
  - Il rilevamento è il primo passo verso la buona progettazione
- Rilevamento: verifica che **gli invarianti** siano rispettati

Date – 5 (rilevamento dati non validi)

- Creiamo una funzione is_valid() che verifica se i dati sono validi o meno

```
class Date {
public:
    class Invalid {};          // per riportare errori
    Date (int y, int m, int d);      // check e inizializza
    // ...

private:
    int y, m, d;
    bool is_valid();           // ritorna true se la data è
                               // valida
};
```

Funzione is_valid

```
class Date {
public:
    class Invalid {};          // per riportare errori
    Date (int y, int m, int d);      // check e inizializza
    // ...

private:
    int y, m, d;
    bool is_valid();           // ritorna true se la data è
                               // valida
};
```

- is_valid è una funzione separata perché rappresenta un'attività diversa dall'inizializzazione
  - **Può essere utilizzata da tutti i costruttori**

Potremmo avere più costruttori per la stessa classe, non è detto che is_Valid si debba usare per ognuno, in più scriverla dentro ogni costruttore sarebbe replicare codice

Classe Invalid

```
class Date {
public:
    class Invalid {};          // per riportare errori
    Date (int y, int m, int d);     // check e inizializza
    // ...

private:
    int y, m, d;
    bool is_valid();           // ritorna true se la data è
                               // valida
};
```

- Talvolta gli errori sono rappresentati usando una classe dedicata
- La classe Invalid ha il solo scopo di creare un tipo unico che **segnala** questo specifico errore
  - Classe definita in un'altra classe → uno dei pochi casi accettabili

Segnalare un errore

- is_valid() verifica la correttezza dello stato di un oggetto
  - Deve segnalare l'esito della verifica
- Vari modi per segnalare una condizione di errore
  - Usare codici dedicati (return della funzione)
  - Scrivere su una variabile membro
  - …
  - **Lanciare una eccezione**
- La classe Invalid segnala un errore quando *è lanciata* in un'eccezione

→ avere oggetti esistenti ma non validi è rischioso e da evitare nella maggior parte dei casi

Eccezioni – richiamo

- Meccanismo che separa il rilevamento di un errore dalla sua gestione
  - Rilevamento: spesso in una funzione chiamata
  - Gestione: spesso nella funzione chiamante
- Le eccezioni garantiscono che l'errore sia gestito
  - Se non è così, il programma termina

Eccezioni

- Se una funzione rileva un errore che non può gestire, **non ritorna normalmente**
- La funzione chiamata lancia (throw) un'eccezione
- Un qualche chiamante (diretto o indiretto) recepisce (catch) l'eccezione
- Una funzione esprime interesse per le eccezioni usando un blocco try
- Ciò che viene lanciato è un oggetto

Implementazione

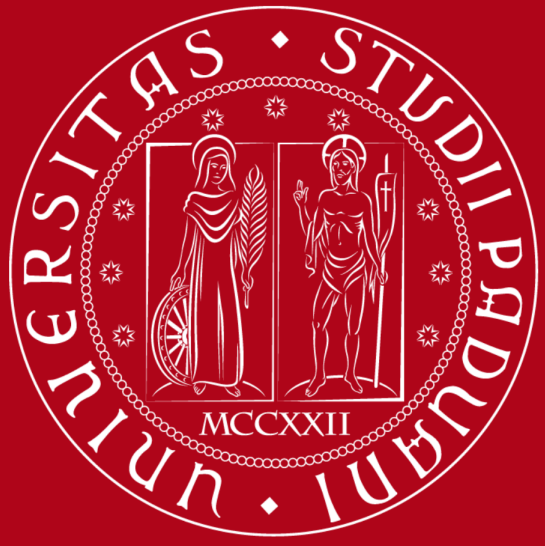
```
Date::Date(int yy, int mm, int dd)
    : y (yy), m(mm), d(dd)
{
    if (!is_valid()) throw Invalid();
}

bool Date::is_valid()
{
    if (m < 1 || m > 12) return false;
    // ...
}
```

- Eccezione lanciata nel costruttore
  - Un'eccezione è l'unico modo in cui possiamo far fallire un costruttore

Eccezioni – try/catch

```
void f(int x, int y)
{
try {
    Date dxy{2004, x, y};
    cout << dxy << '\n';
    dxy.add_day(2);
}
catch(Date::Invalid) {
    cout << "Invalid date!\n";
}
}
```

I catch universali (con i tre punti) esistono, ma in genere li scriveremo per specifiche eccezioni.



UNIVERSITÀ
DEGLI STUDI
DI PADOVA

Classe Date:
Rilevare gli errori

Stefano Ghidoni

DIPARTIMENTO
DI INGEGNERIA
DELL'INFORMAZIONE