



# UNIVERSITÀ DEGLI STUDI DI PADOVA

## Standard exception

Stefano Ghidoni



- Recap sulle eccezioni
- Standard exceptions
- Derivare le standard exceptions



# Eccezioni con Date

```
class Date {  
    public:  
        Date (int yy, int mm, int dd);  
        bool is_valid();  
        class Invalid{};  
        // ...  
    private:  
        int y, m, d;  
};
```

```
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;  
    // ...  
}
```



# Eccezioni con Date

```
class Date {  
    public:  
        Date (int yy, int mm, int dd);  
        bool is_valid();  
        class Invalid{};  
        // ...  
    private:  
        int y, m, d;  
};
```

Tipo che segnala  
una data non valida

```
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;  
    // ...  
}
```



- Classe dedicata
  - Design comune
  - Potenziali problemi?



- La classe Invalid è stata creata per segnalare un'eccezione
    - Non è standard
    - Eventuali catch devono conoscere l'esistenza di Invalid
- Sono potenziali problemi



- Come standardizzare le eccezioni?
- La libreria standard contiene una classe base che descrive il concetto di eccezione
  - `std::exception`
  - Tutte le altre derivano da `std::exception`

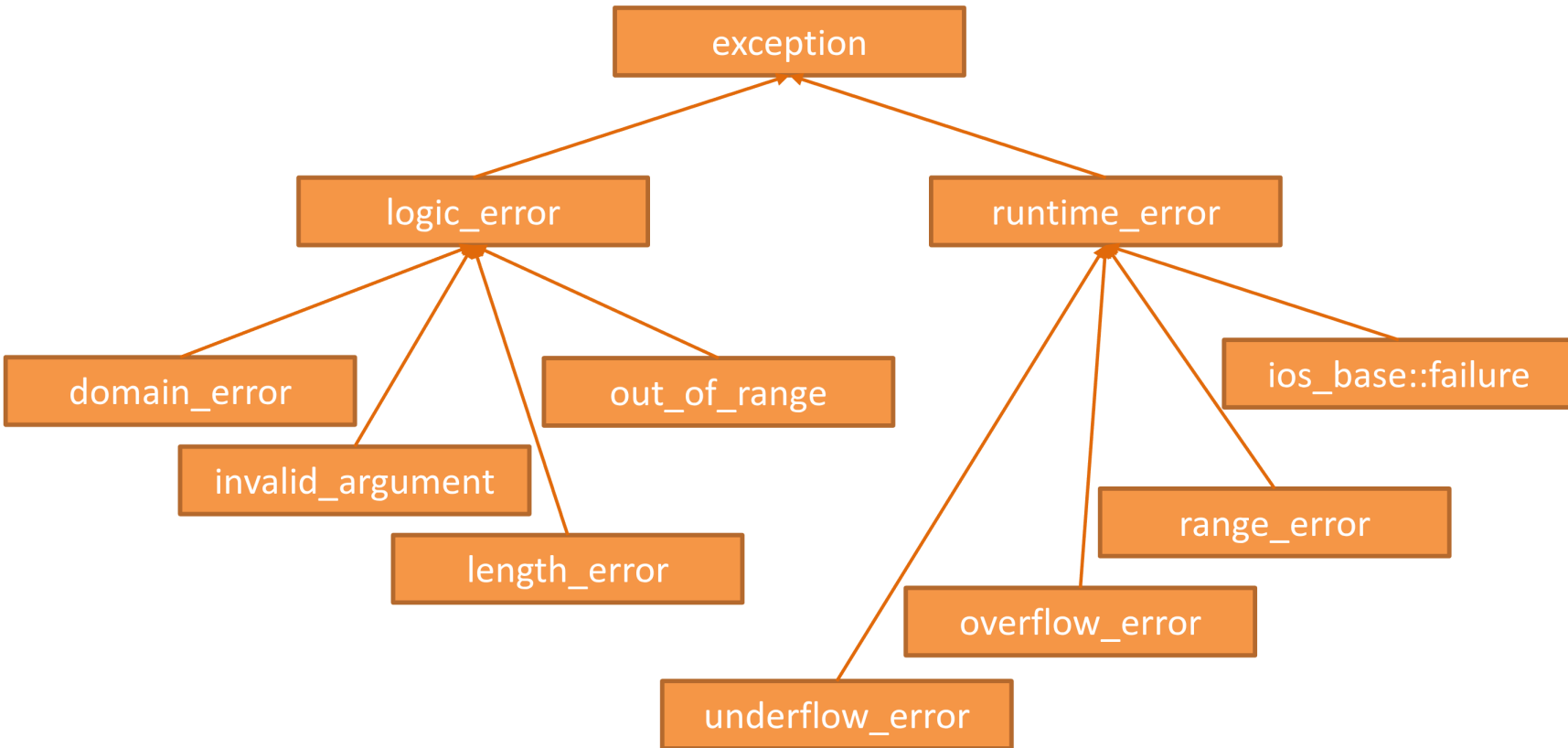


- Due famiglie di eccezioni
  - **`std::logic_error`**
    - Riporta un errore che consegue una logica difettata come la violazione di pre-condizioni o gli invarianti della classe
  - **`std::runtime_error`**
    - Riporta errori dovuti a eventi esterni al programma stesso che sono difficilmente prevedibili





# Eccezioni standard





- Classe exception:

```
class exception {  
    public:  
        exception();  
        exception(const exception&);  
        exception& operator=(const exception&);  
        virtual ~exception();  
        virtual const char* what() const;  
};
```

- Costruttore e assegnamento di copia disponibili
- Funzione what() per ottenere una stringa di descrizione



- Tipico pattern di utilizzo

```
// ...  
  
try {  
    // ...  
    // ...  
} catch (std::exception& e) {  
    std::cerr << e.what() << std::endl;  
}  
  
// ...
```

- L'eccezione rilevata ora è standard



- È anche possibile creare eccezioni derivate

```
class My_error : runtime_error {  
    public:  
        My_error(int x) : error_value{x} {}  
        int error_value;  
        const char* what() const override { return "My_error"; }  
};
```

- Le eccezioni derivate permettono di personalizzare `what()`

In genere derivare dalla libreria standard è sconsigliato, ma nel caso delle eccezioni non è un problema



- Eccezioni dedicate
- Eccezioni standard
- Eccezioni derivate di `std::exception`



# UNIVERSITÀ DEGLI STUDI DI PADOVA

## Standard exception

Stefano Ghidoni