13.1 – Standard exceptions

Capitoli libro:

- B.2.1

DIPARTIMENTO
DI INGEGNERIA
DELL'INFORMAZIONE

Agenda

- Recap sulle eccezioni
- Standard exceptions
- Derivare standard exceptions

La classe Date

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

Abbiamo creato un tipo specifico per segnalare una data invalida

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

Lanciamo questo tipo in caso di eccezione

La classe `Date`

Qual è un possibile problema del tipo `Invalid()`?

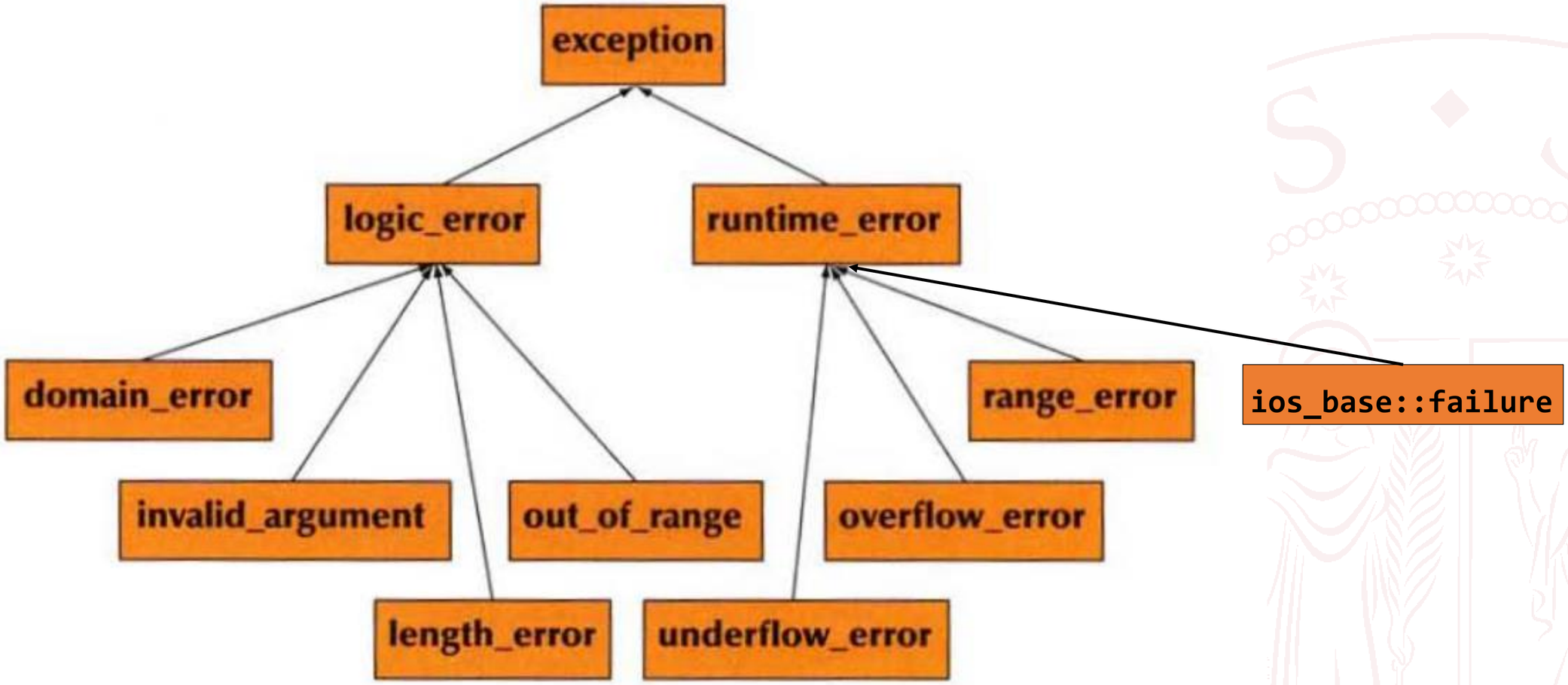
```
class Date {
    public:
        Date (int yy, int mm, int dd);
        bool is_valid();
        class Invalid{};
        // ...
    private:
        int y, m, d;
};
```

- Non è standard
- Lo conosciamo solo noi che abbiamo disegnato la classe
- E se qualcuno usa la nostra classe `Date` all'interno del suo programma?

`std::exception`

- La standard library fornisce una varietà di eccezioni
- Tutte le eccezioni nella standard library derivano da `std::exception`
- Si possono dividere in due famiglie:
  - **`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

Alcune classi di eccezioni nella standard library

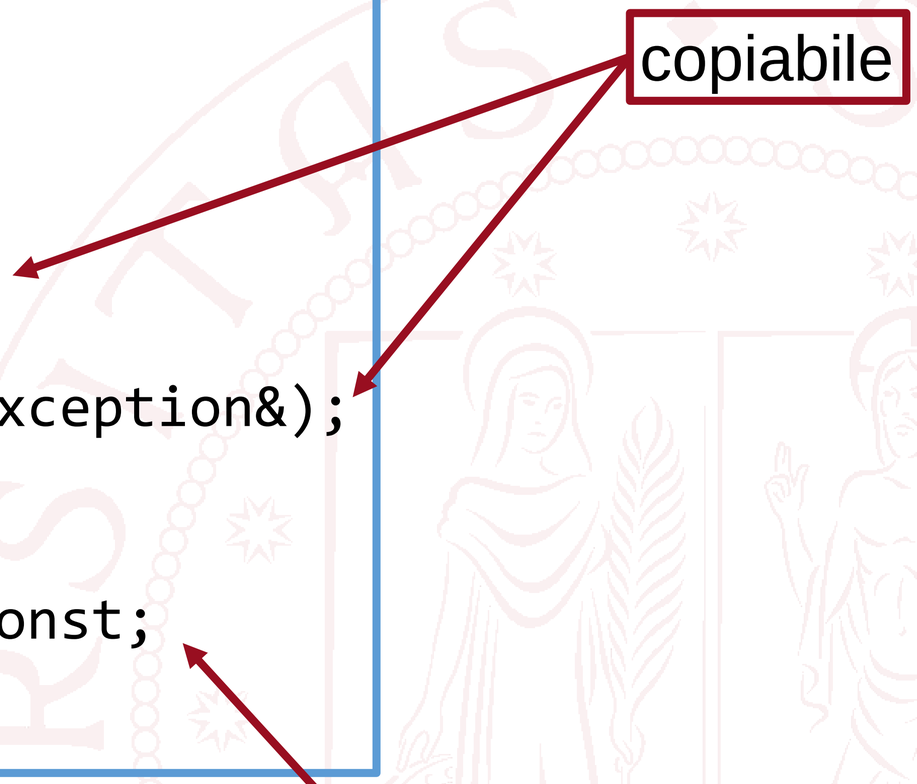


https://en.cppreference.com/w/cpp/error/exception

Design della classe `std::exception`

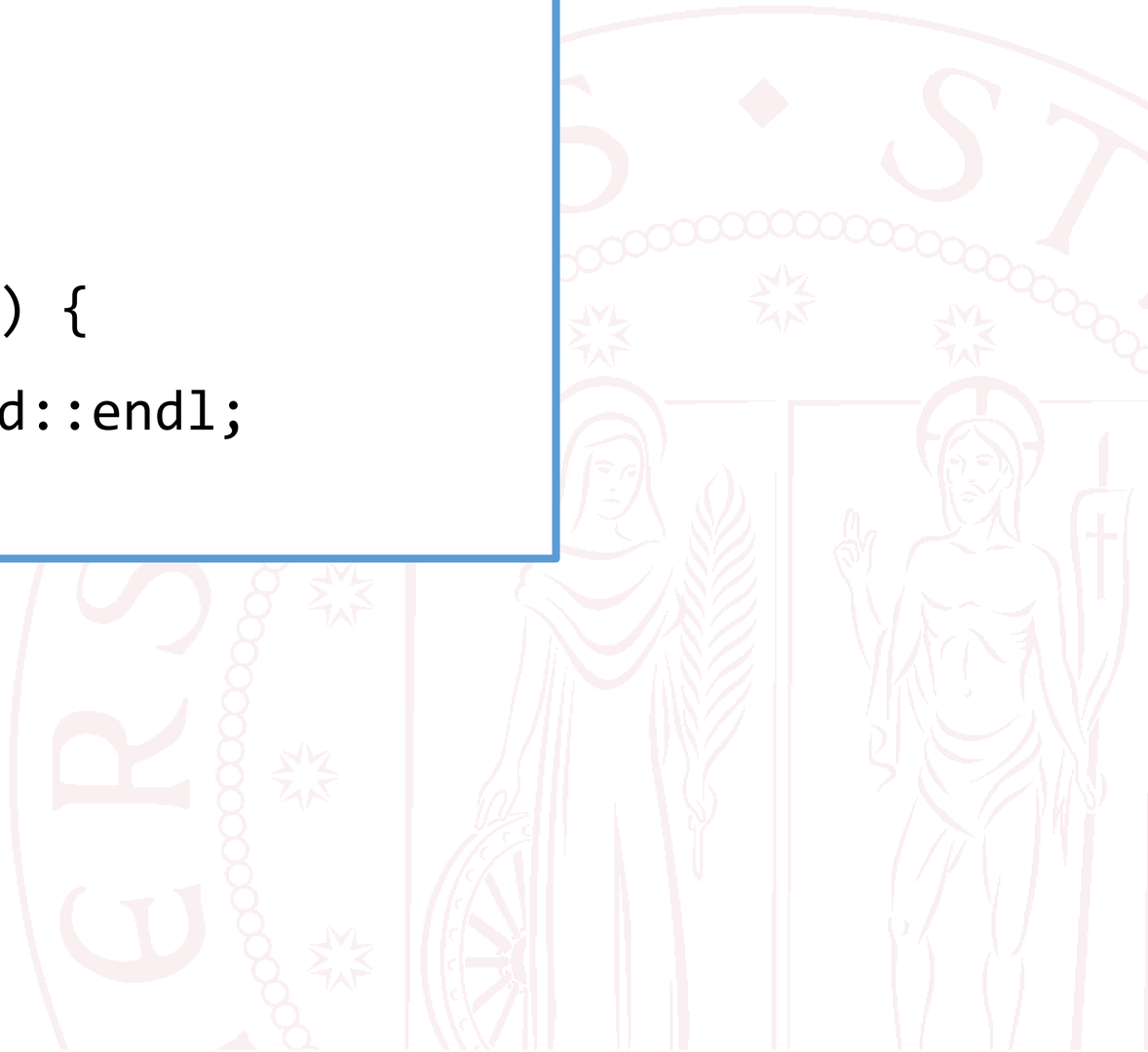
- Inclusa in `<exception>`

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



Utilizzo della classe `std::exception`

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



Derivare dalle standard exceptions

- Piuttosto che creare una propria classe per l'eccezioni, è dunque meglio:
  - Utilizzare la libreria standard

    (`std::runtime_error`, `std::out_of_range`…)
  - Creare una classe derivata dalla libreria standard

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

Recap

- Genericità delle eccezioni
- Standard exceptions
- Design standard exceptions
- Derivare le standard exceptions