12.4 – Algoritmi STL

Associative container

Capitoli:

- 21.6

DIPARTIMENTO
DI INGEGNERIA
DELL'INFORMAZIONE

Agenda

- Associative container
  - `std::map`
  - `std::set`
- Gestione di una coppia: `std::pair`

Associative container – lista

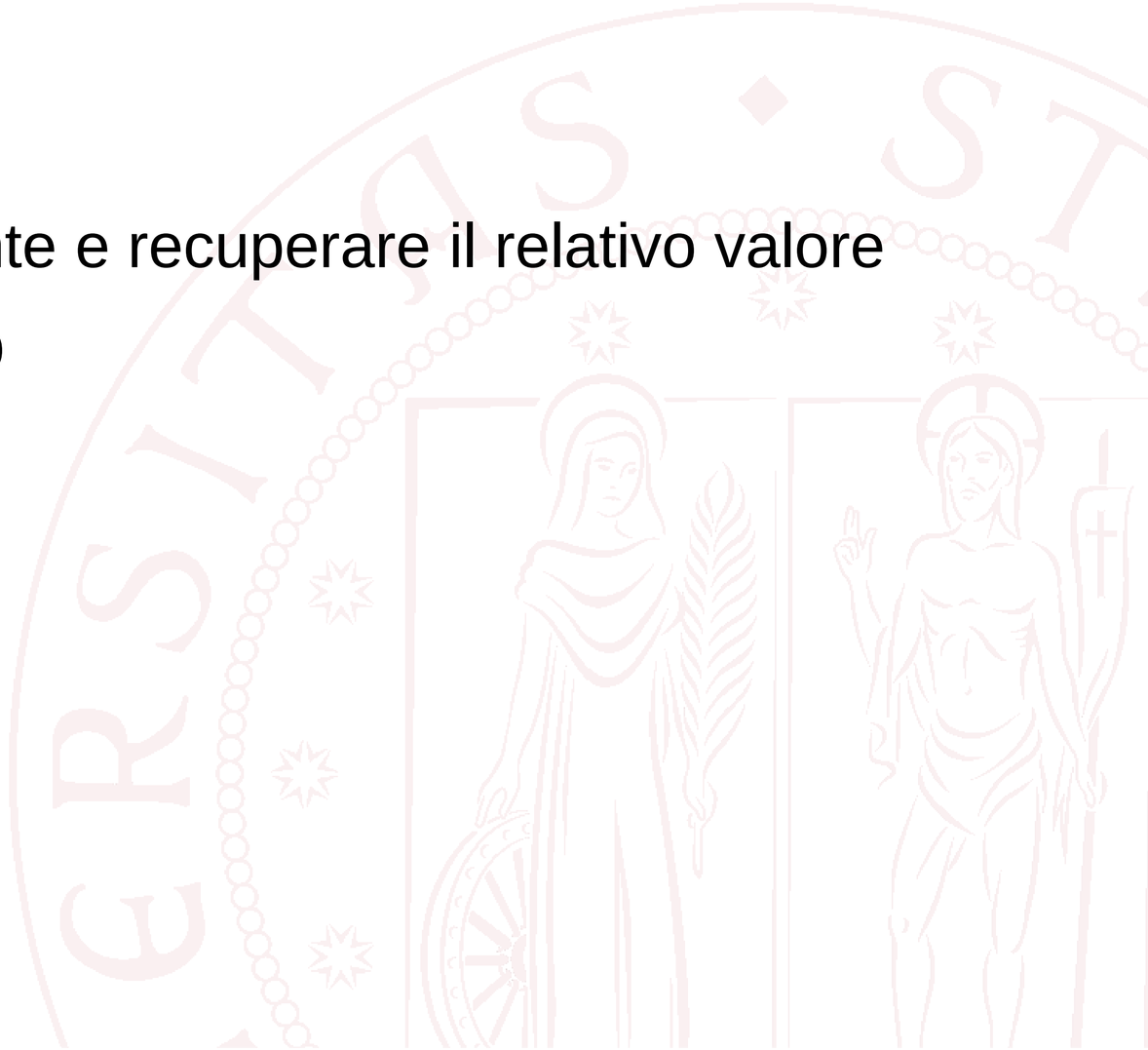
Contenitore	Significato
`std::map`	Contenitore ordinato di coppie (chiave, valore)
`std::set`	Contenitore ordinato di chiavi
`std::unordered_map`	Contenitore non ordinato di coppie (chiave, valore)
`std::unordered_set`	Contenitore non ordinato di chiavi
`std::multimap`	Una `std::map` in cui una chiave può ricorrere più di una volta
`std::multiset`	Un `std::set` in cui una chiave può ricorrere più di una volta
`std::unordered_multimap`	`std::unordere_map` in cui una chiave può ricorrere più di una volta
`std::unordered_multiset`	`std::unordered_set` in cui una chiave può ricorrere più di una volta

std::map

DIPARTIMENTO
DI INGEGNERIA
DELL'INFORMAZIONE

`std::map`

- Gestisce coppie chiave-valore
- Contenitore ordinato
- Ottimizzato per ricerche dalla chiave
  - È veloce verificare se una chiave è presente e recuperare il relativo valore
- Rappresentato con un albero bilanciato
- Header: `<map>`



`std::map` – esempio

- Esempio: creare un istogramma delle parole in un testo

```
int main()
{
    std::map<std::string, int> words;

    for(std::string s; cin >> s; )
        ++words[s];

    for(const auto& p : words)
        std::cout << p.first << ": " << p.second << '\n';

    return 0;
}
```

Contenitore indicizzato con stringhe

Ad ogni stringa corrisponde un intero:

key [std::string]	value [int]
Ciao,	...
come	...
stai?	...

Elemento cruciale

Ciclo sugli elementi di `words`

Indicizzare `map`

```
for(std::string s; cin >> s; )
    ++words[s];
```

- Indicizzazione con una `std::string`
- A partire da `std::string`, la `std::map` fornisce il relativo `int`
- Se la chiave non esiste nella `std::map`, è creata un elemento con quella chiave
  - Il corrispondente `int` è inizializzato a `0` – costruttore di default di `int`

std::pair

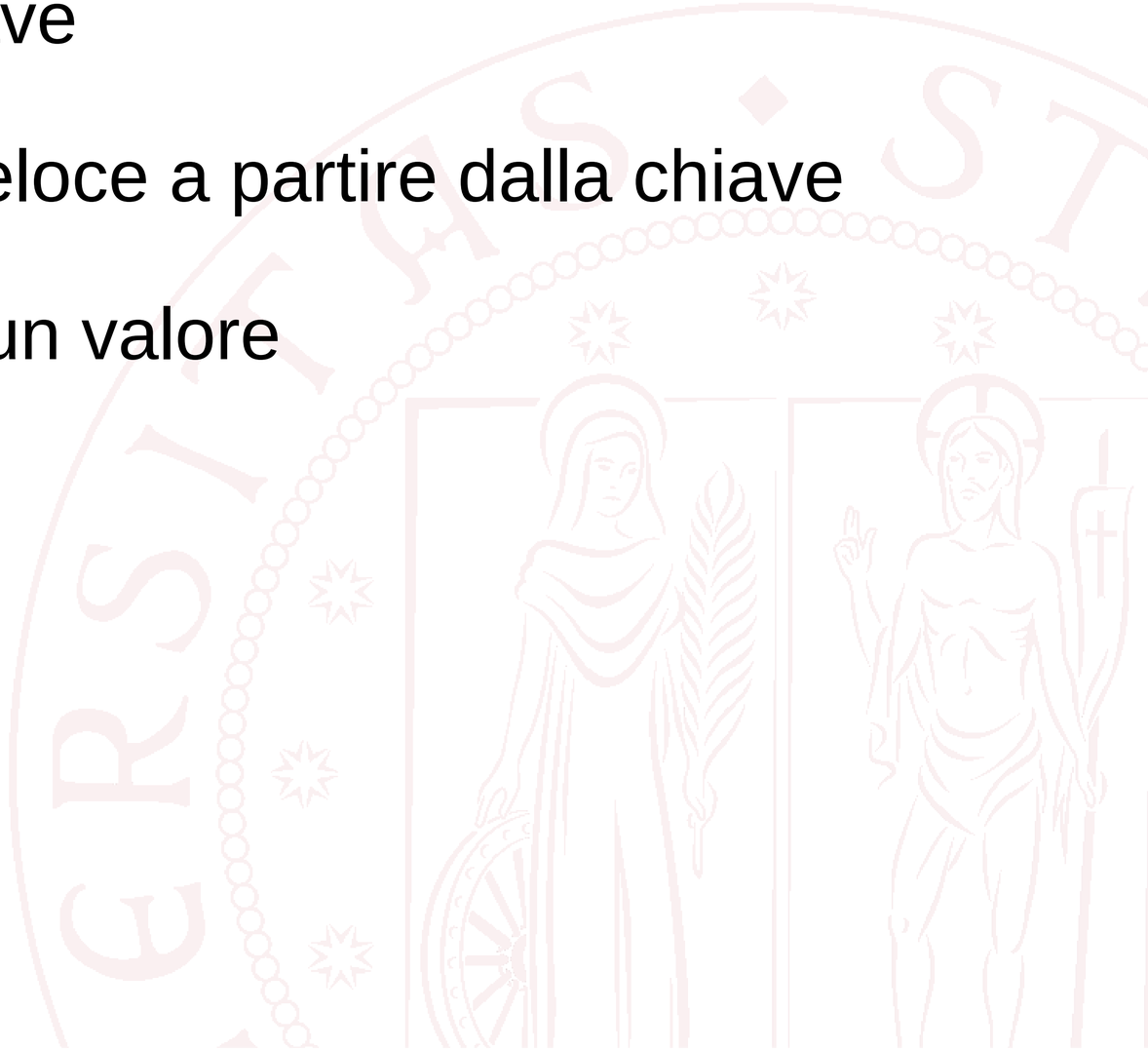
- Gli elementi della `std::map` sono coppie chiave-valore
  - Gestite come `std::pair<std::string, int>`
  - Elementi di una `std::pair`: `first` e `second`

- Per creare una `std::pair`: `make_pair()`

```
std::cout << p.first << ": " << p.second << '\n';
```

`std::map` vs `std::vector`

- `std::map` ordina i dati rispetto alla chiave
- `std::map` offre una ricerca molto più veloce a partire dalla chiave
- Associazione esplicita di una chiave a un valore

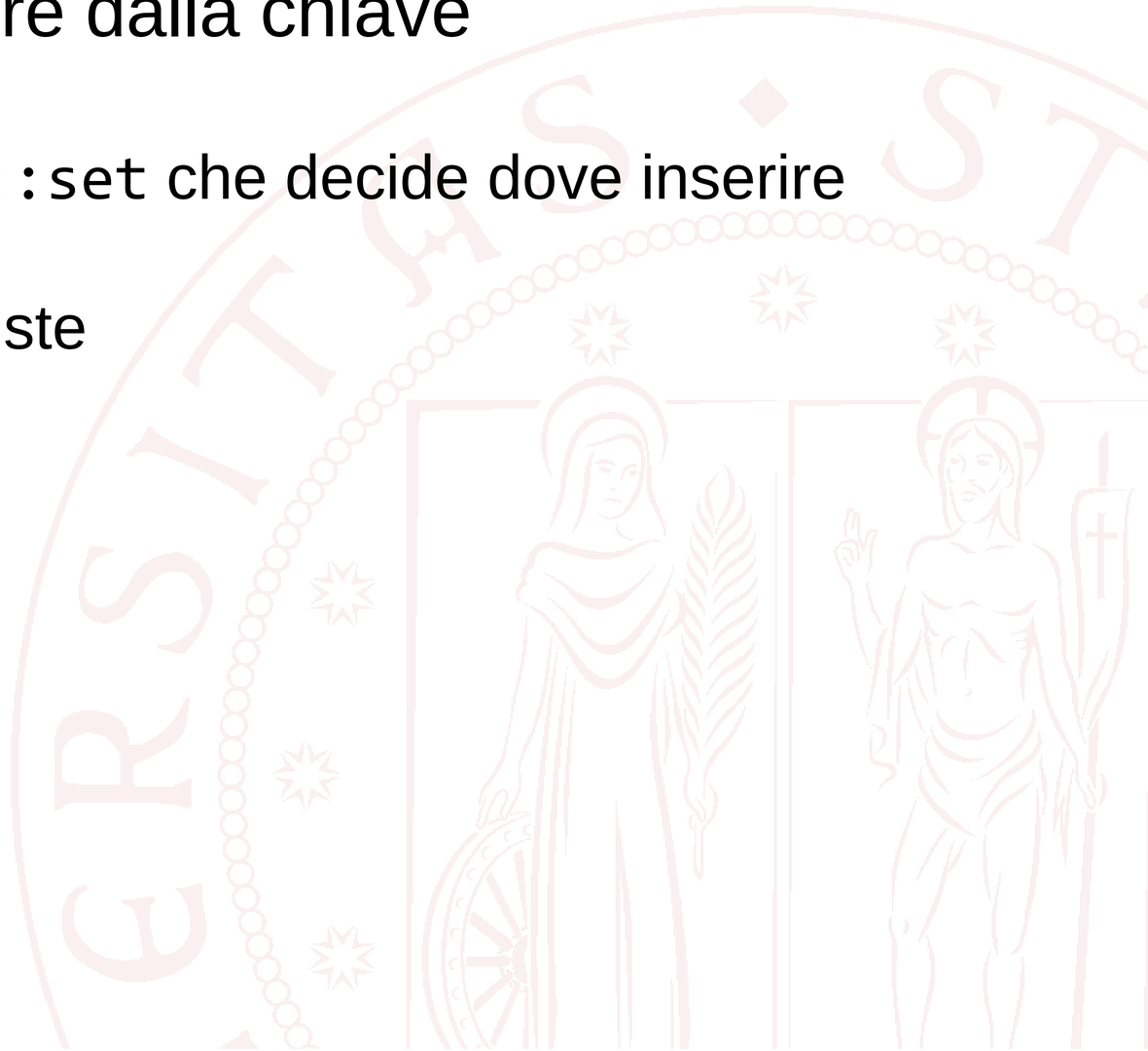


std::set

DIPARTIMENTO
DI INGEGNERIA
DELL'INFORMAZIONE

`std::set`

- Un `std::set` è una `std::map` senza i valori
- Viene meno la ricerca del valore a partire dalla chiave
  - Non è disponibile `operator[]`
  - Non è disponibile `push_back()` – è il `std::set` che decide dove inserire l'elemento
  - Gestito tramite le operazioni tipiche delle liste
  - `insert()` ed `erase()`
- Header: `<set>`



`std::set`

- `std::set` è un contenitore ordinato
- Deve essere definita una funzione d'ordine per ciascun tipo contenuto
  - Inclusi UDT, ovviamente

`std::set` – esempio

- Es: se vogliamo costruire un `std::set` di elementi `Fruit`:

```
struct Fruit{
    std::string name;
    int count;
    double unit_price;
    Date last_sale_date;
    // ...
};
```

`std::set` – esempio

- È necessario definire una funzione d'ordine:

```
struct Fruit_order {
    bool operator()(const Fruit& a, const Fruit& b) const
    {
        return a.name < b.name;
    }
};
```

- È ora possibile creare:

```
std::set<Fruit, Fruit_order> inventory;
// Nota: Fruit_order serve per confrontare Fruit
```

`std::set` – esempio

- È possibile popolare il `std::set` così:

```
inventory.insert(Fruit{"Quince", 5});
inventory.insert(Fruit{"Apple", 200, 0.37});
```

- E scorrerlo / stamparlo così:

```
for(auto p = inventory.begin(); p != inventory.end(); ++p)
    cout << *p << '\n';
```

- `std::set` non contiene `pair`, si accede al dato direttamente con *

Spunti di approfondimento

- `std::unordered_map`
  - Implementano le hash table
- Stream iterator

Recap

- Uso di `std::map` per la gestione di coppie chiave-valore
- Uso di `std::set` per la gestione ordinata di elementi
- Funzioni d'ordine
- Inserimento di oggetti in container ordinati