2.3 – Istruzioni, dichiarazioni, definizioni

(*ioni)

Libro di testo:

- Capitolo 4.4
- Capitolo 8.2

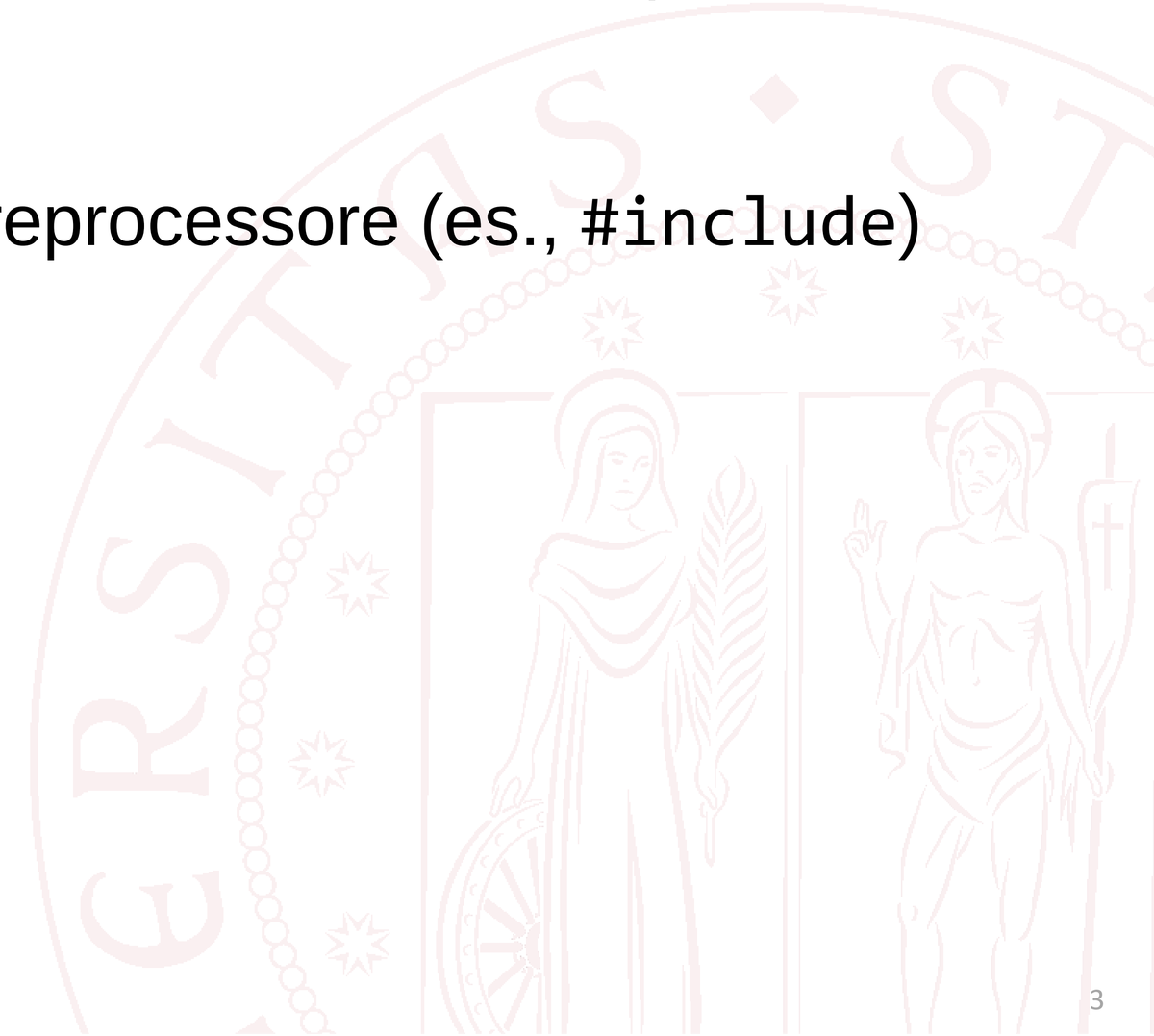
DIPARTIMENTO
DI INGEGNERIA
DELL'INFORMAZIONE

Agenda

- Istruzioni
- Dichiarazioni e definizioni
- Inizializzazione

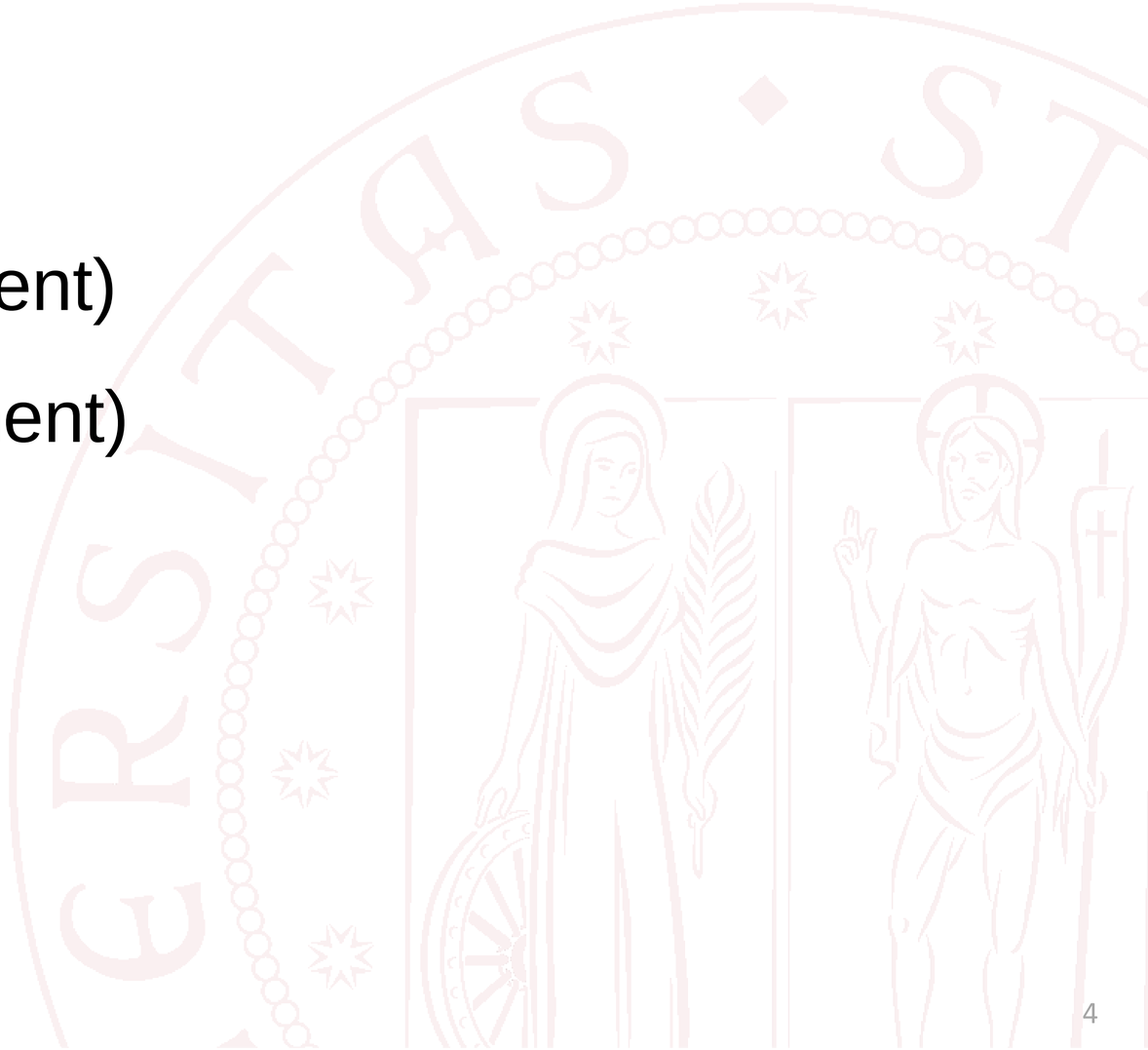
Statements

- Un'istruzione o *statement* è una parte di codice C++ che specifica un'azione
  - Non sono istruzioni le direttive del preprocessore (es., `#include`)
- Termina con **`;`**
  - Necessario per interpretare il codice
- Vari tipi di istruzioni
  - Qualche esempio?



Tipi di istruzioni

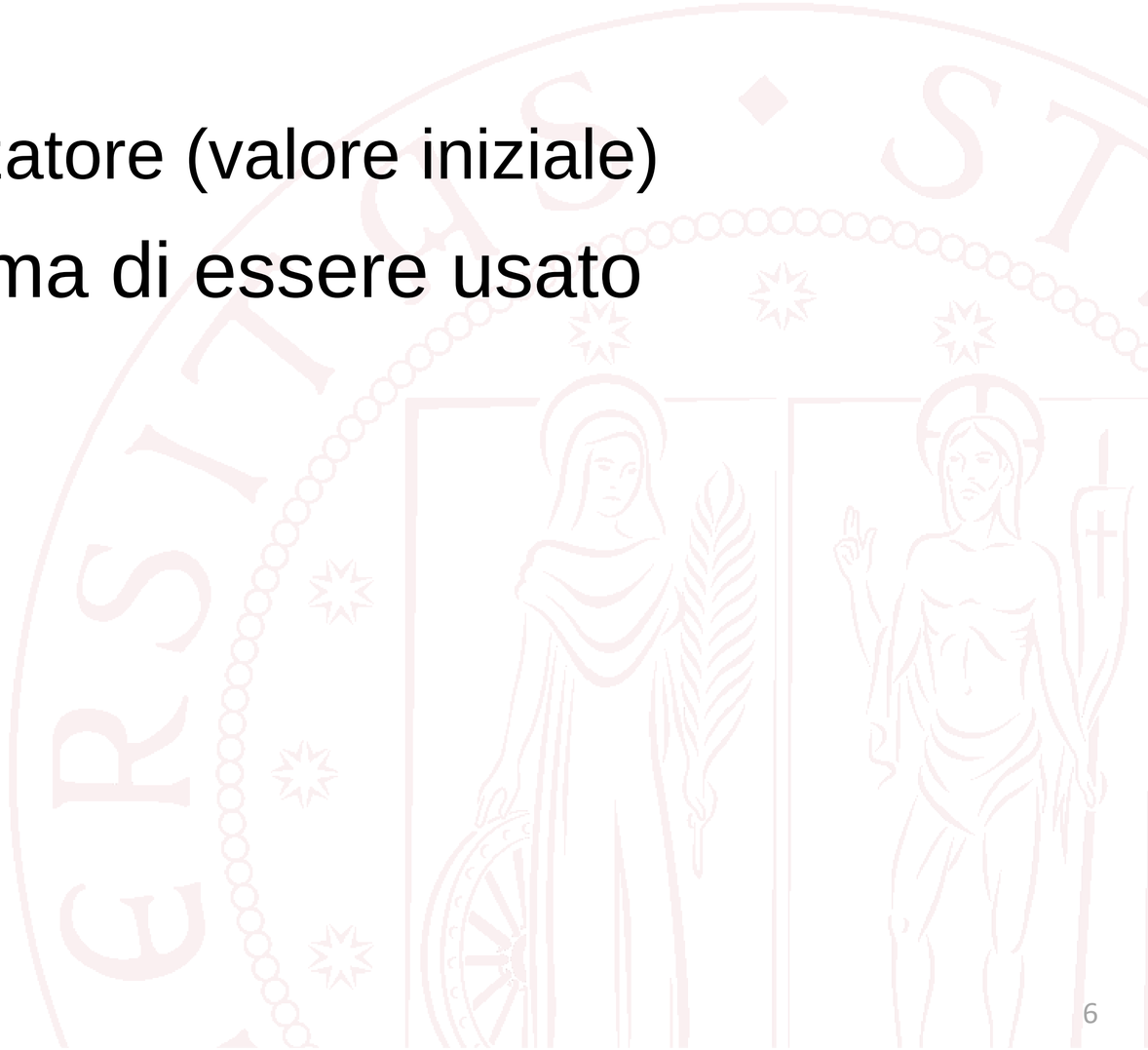
- Expression statements
- Dichiarazioni
- Selezione (if-statement, switch-statement)
- Iterazione (while-statement, for-statement)
- …



Istruzioni per gestire le variabili (e non solo): dichiarazioni e definizioni

Dichiarazione

- Dichiarazione: istruzione che introduce un nome in uno scope
  - Specifica il nome
  - Specifica il tipo
  - Opzionalmente, specifica un inizializzatore (valore iniziale)
- Un nome deve essere dichiarato prima di essere usato
- Dichiarazione di:
  - Variabili
  - Funzioni
  - ... (molto altro)



Dichiarazione

- Esempi di dichiarazioni – per ciascuna delle seguenti, verificate:
  - Specifica il nome?
  - Specifica il tipo?
  - Opzionalmente, specifica un inizializzatore (valore iniziale)?

```
int a = 7;
const double cd = 8.7;
double sqrt (double);     // una funzione con un
                          // argomento double
std::vector<Token> v;     // vettore di variabili Token
```

Dichiarazioni

- Una dichiarazione ha effetto sulla memoria?

```
int a = 7;
const double cd = 8.7;
double sqrt (double);     // una funzione con un
                          // argomento double
vector<Token> v;          // vettore di variabili Token
```

Definizioni

- Definizione: una dichiarazione che specifica completamente l'entità dichiarata
  - Una definizione è anche una dichiarazione

```
int a = 7;
vector<double> v;
double sqrt (double d) { /* ... */ }
```

- Alcune dichiarazioni ***non*** sono definizioni

```
double sqrt (double);
extern int a;
```

Definizione

- «Fornisce tutte le informazioni necessarie per creare l'entità nella sua interezza» (BS)
- Può avere un significato diverso a seconda dell'entità definita:
  - variabile → riservare la memoria
  - funzione → fornire il corpo della funzione
  - classe → fornire tutti i membri e le funzioni membro della classe
- **Quando un'entità è definita allora, per forza, è anche dichiarata**

Dichiarazioni vs definizioni

- In alcuni contesti (**non in questo corso**), dichiarazione e definizione sono considerati mutuamente esclusivi
  - Dichiarazione intesa come *dichiarazione che non è anche una definizione*
- Una dichiarazione spiega come usare un nome, è ***un'interfaccia***, non alloca spazio in memoria né specifica il corpo di una funzione

Dichiarazioni vs definizioni

- **Variabili**
  - Una dichiarazione fornisce il nome della variabile e il tipo (nulla è detto sulla memoria)
  - Una definizione fornisce un oggetto (riserva memoria) con un determinato nome e tipo

- **Funzioni**
  - Una dichiarazione fornisce solo il tipo (degli argomenti e del valore ritornato)
  - Una definizione fornisce il corpo
    - Il corpo occupa spazio in memoria!

Definizione

```
int func();                 // dichiarazione della funzione func

int main()
{
    int x = func();         // N.B. Utilizzo prima della definizione
}

int func()                  // definizione della funzione func
{
    return 2;
}
```

Dichiarazioni vs definizioni

- Perché esistono dichiarazione e definizione?
- Per poter dichiarare entità prima di fornirne la definizione
- Per gestire file sorgente multipli (ad esempio file headers)
  - La dichiarazione permette a tutte le parti del programma di «conoscere» l'entità senza bisogno di sapere come essa è stata definita

- Dichiarazione vs definizione riflette interfaccia vs implementazione
  - Interfaccia: di cosa abbiamo bisogno per usare qualcosa
  - Implementazione: di cosa abbiamo bisogno affinché qualcosa faccia ciò che deve

Dichiarazioni vs definizioni

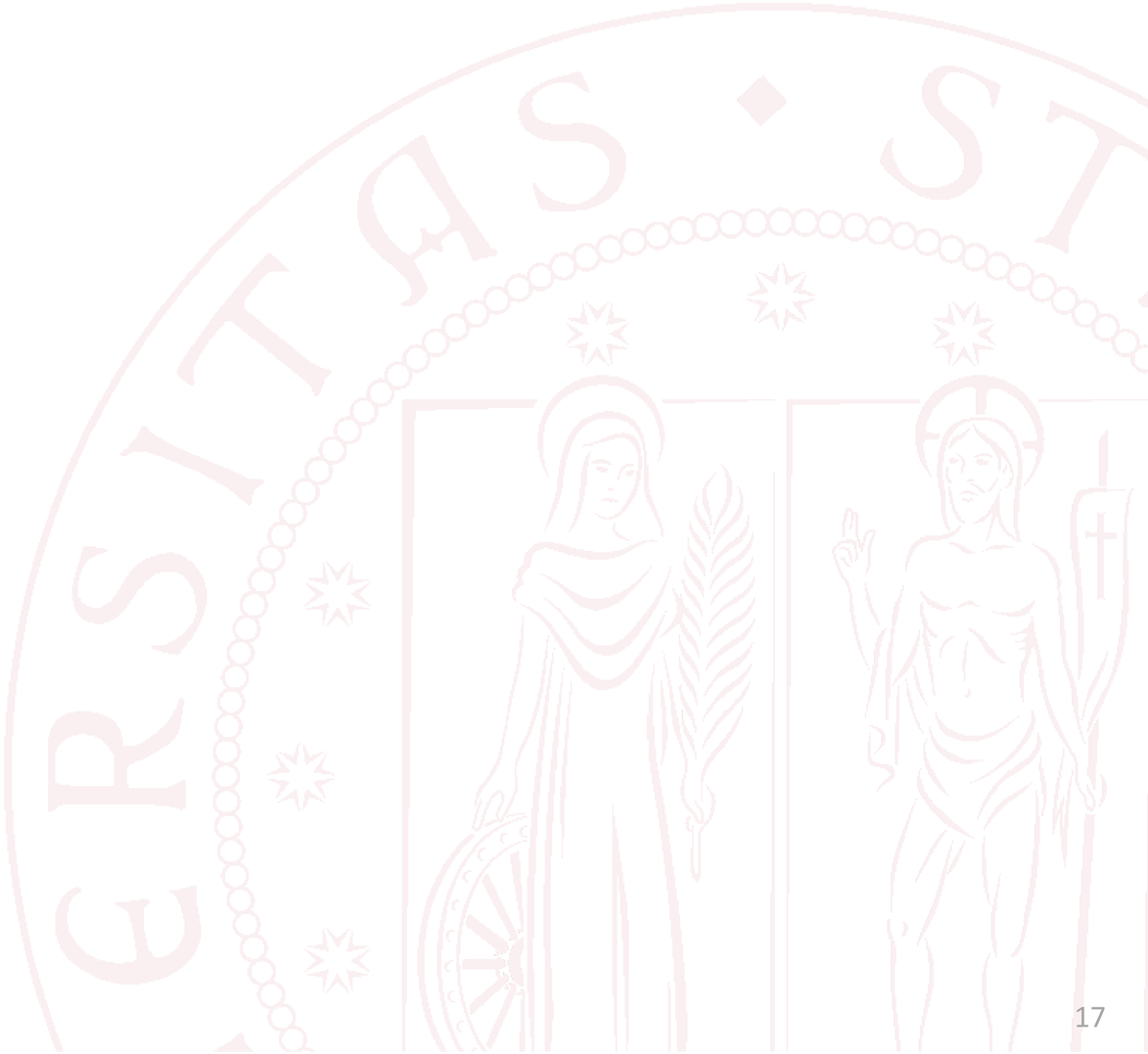
- La sola dichiarazione ci permette di riferirci a un'entità (variabile, funzione, …)
- Spesso vogliamo la definizione altrove
  - Più sotto nel file
  - In un altro file
- Dichiarazioni raccolte in un header file permettono di condividere le funzioni

Esempi di dichiarazioni e definizioni

```
int x = 7;                                          // ?
```

Esempi di dichiarazioni e definizioni

```
int x = 7;                                  // definizione
extern int x;                               // ?
```



Esempi di dichiarazioni e definizioni

```
int x = 7;                          // definizione
extern int x;                       // dichiarazione
extern int x;                       // ?
```

- `extern` specifica che la dichiarazione non è una definizione
  - Poco usato, poco utile

Esempi di dichiarazioni e definizioni

```
int x = 7;                              // definizione
extern int x;                           // dichiarazione
extern int x;                           // un’altra dichiarazione

double sqrt(double);                    // ?
```

- Extern specifica che la dichiarazione non è una definizione
  - Poco usato, poco utile

Esempi di dichiarazioni e definizioni

```
int x = 7;                                  // definizione
extern int x;                               // dichiarazione
extern int x;                               // un’altra dichiarazione

double sqrt(double);                        // dichiarazione
double sqrt(double d) {/* ... */}           // ?
```

- Extern specifica che la dichiarazione non è una definizione
  - Poco usato, poco utile

Esempi di dichiarazioni e definizioni

```
int x = 7;                                  // definizione
extern int x;                               // dichiarazione
extern int x;                               // un’altra dichiarazione

double sqrt(double);                        // dichiarazione
double sqrt(double d) {/* ... */}           // definizione
double sqrt(double);                        // ?
```

- Extern specifica che la dichiarazione non è una definizione
  - Poco usato, poco utile

Esempi di dichiarazioni e definizioni

```
int x = 7;                                  // definizione
extern int x;                               // dichiarazione
extern int x;                               // un’altra dichiarazione

double sqrt(double);                        // dichiarazione
double sqrt(double d) {/* ... */}           // definizione
double sqrt(double);                        // altra dichiarazione
double sqrt(double);                        // ?
```

- Extern specifica che la dichiarazione non è una definizione
  - Poco usato, poco utile

Esempi di dichiarazioni e definizioni

```
int x = 7;                                  // definizione
extern int x;                               // dichiarazione
extern int x;                               // un’altra dichiarazione

double sqrt(double);                        // dichiarazione
double sqrt(double d) {/* ... */}           // definizione
double sqrt(double);                        // altra dichiarazione
double sqrt(double);                        // altra dichiarazione

int sqrt(double);                           // ?
```

- Extern specifica che la dichiarazione non è una definizione
  - Poco usato, poco utile

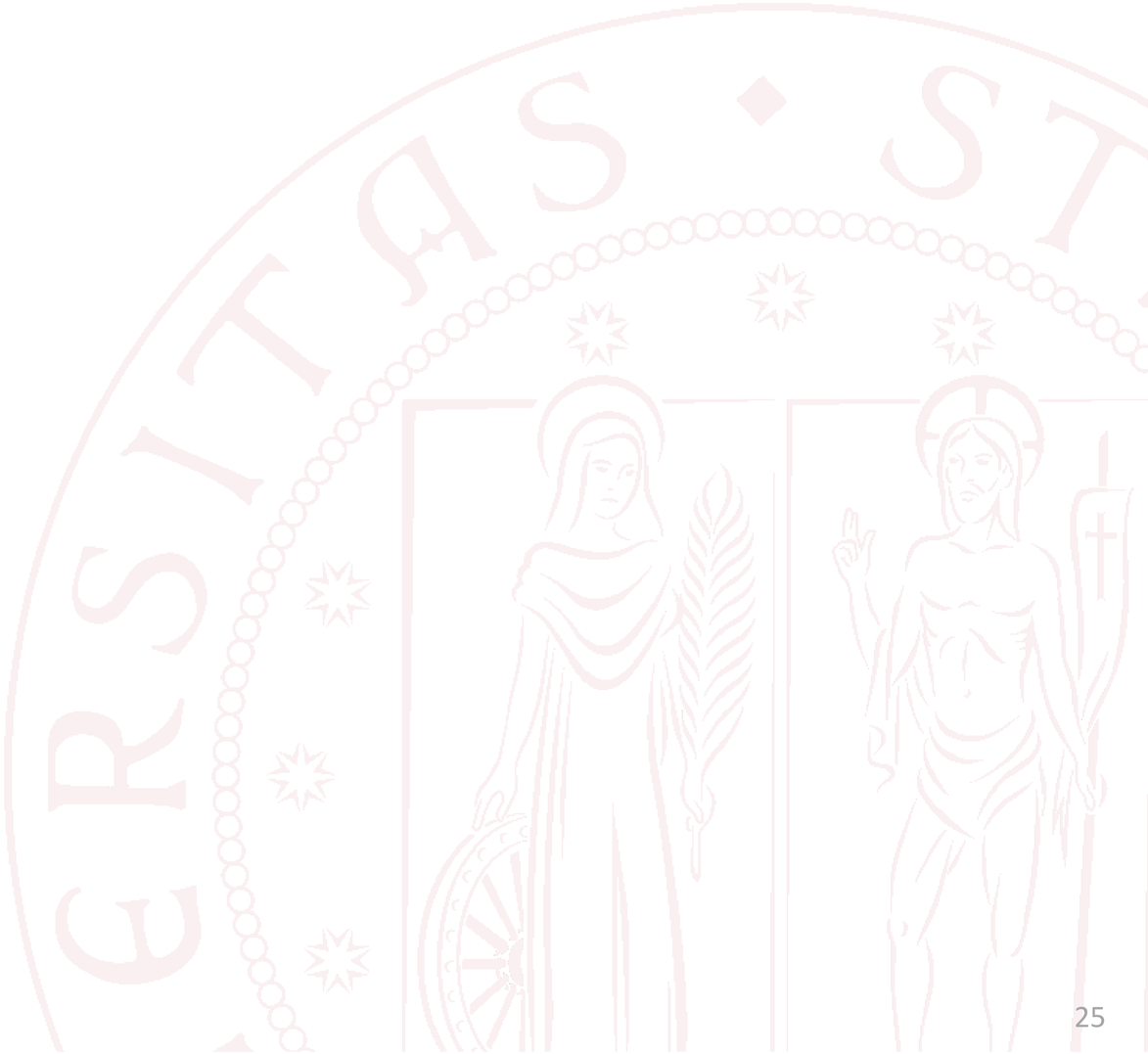
Esempi di dichiarazioni e definizioni

```
int x = 7;                                  // definizione
extern int x;                               // dichiarazione
extern int x;                               // un'altra dichiarazione

double sqrt(double);                        // dichiarazione
double sqrt(double d) {/* ... */}           // definizione
double sqrt(double);                        // altra dichiarazione
double sqrt(double);                        // altra dichiarazione

int sqrt(double);                           // errore: dichiarazione
                                            // incoerente
```

- `extern` specifica che la dichiarazione non è una definizione
  - Poco usato, poco utile

Definizioni

- In C++ si possono definire
  - Variabili
  - Costanti
  - Funzioni
  - Namespace
  - Tipi (es., classi, enum)
  - Template



Inizializzazione

Dichiarazione, definizioni e inizializzazioni

```
int a;
double d = 7;
vector<int> vi(10);                 // inizializzatore con sintassi ()
vector<int> vi2 {1, 2, 3, 4};       // inizializzatore con sintassi {}

const int x = 7;                    // inizializzatore obbligatorio
const int x2 {9};                   // inizializzatore con sintassi {}
const int y;                        // errore: nessun inizializzatore
```

- L'inizializzazione è spesso presente nelle dichiarazioni e definizioni
- La sintassi {} è preferibile per l'inizializzazione
  - È più esplicita
  - Ancora non molto diffusa
- L'operatore = è usato per inizializzazioni semplici

Variabili non inizializzate

- Che succede se non inizializziamo le variabili?
- "La ricetta per bug oscuri…" (BS)

```
void f(int z)
{
    int x; // non inizializzato
    // ... Nessun asssegnamento su x
    x = 7;
    // ...
}
```

- Sembra codice accettabile, ma può essere pericoloso!
  - Dipende dal codice nel primo ...

Variabili non inizializzate

- Esempio di ... problematico

```
void f(int z)
{
    int x;                  // non inizializzato
                            // ... Nessun asssegnamento su x
    if (x > z)
    {
                            // ...
    }
                            // ...
    x = 7;
                            // ...
}
```

come si comporta il programma?