2 - Dichiarazioni e Definizioni

Dichiarazione: costrutto che introduce (dichiara) un nome per una entità. Informa il compilatore del tipo e delle proprietà di una variabile o di una funzione, ma senza riservare spazio in memoria o fornire dettagli sull'implementazione.
Definizioni: dichiarazione che, oltre al nome, fornisce ulteriori elementi per caratterizzare l'entità. Oltre a dichiarare, assegna un valore nel caso di una variabile, o un'implementazione nel caso di una funzione e alloca memoria.

Tutte le definizioni sono anche dichiarazioni, ma non tutte le dichiarazioni sono definizioni, in questo caso vengono chiamate **dichiarazioni pure**.

Esempi:

1. *Tipi di dato*
   `Struct S;`
   dichiarazione pura del tipo S, non possiamo creare oggetti di questo tipo ma può tornare utile per la definizione di puntatori o riferimenti del tipo S, senza dover conoscere il tipo ⇒ **puntatori opachi**.
   `Struct T { int a; };`
   definizione del tipo T, possiamo creare oggetti di questo tipo.
2. *Variabili*:
   `extern int a`
   dichiarazione pura di variabile globale, il compilatore sa della sua esistenza ma la creazione verrà fatta successivamente
   `extern int d = 2`
   definizione, perché è inizializzata
3. *Funzioni*:
   `void foo (int a);`
   dichiarazione pura di funzione, è presente solo la signature della funzione
   `void foo(int a) { std::cout << a; }`
   definizione di funzione, viene fornita l'implementazione della funzione
4. *Template*:

```
template <typename T> struct S; // dichiarazione pura di template di classe
template <typename T>
struct S {
        T t;
}; // definizione di template di classe

template <typename T>
T add(T t1, T t2); // dichiarazione pura di template di funzione
template <typename T>
T add(T t1, T t2) {
```

```
            return t1 + t2;
        }
```