



UNIVERSITÀ
DEGLI STUDI
DI PADOVA

Tipi definiti dall'utente – intro

Stefano Ghidoni



DIPARTIMENTO
DI INGEGNERIA
DELL'INFORMAZIONE



Agenda

- Introduzione alle classi
- Accesso ai dati
- Interfaccia e implementazione



Tipi

- C++: forte focus sui tipi
- Tipi built-in
 - int
 - char
 - double
 - ...
- Tipi definiti dall'utente / User Defined Types (UDT)
 - Classi definite in std
 - string
 - vector
 - ostream
 - Classi definite dal programmatore
 - Struct definite dal programmatore



- `vector`, `string`, `list`, `ostream`, ... Sono considerati UDT
 - Costruiti con le stesse tecniche delle classi definite dal programmatore
 - È possibile costruire i contenitori con gli strumenti messi a disposizione dal C++
 - Sono comunque parte del linguaggio come i tipi built-in
 - I programmatori della libreria standard non hanno "superpoteri" ☺



Perché le classi?

- La programmazione orientata agli oggetti porta alla creazione di nuovi tipi
- "Types are good for directly representing ideas in code" (BS)
- Due elementi chiave:
 - Rappresentazione (representation): un tipo deve sapere come rappresentare i dati necessari in un oggetto
 - Concetto di valore corrente o di **stato**
 - Operazioni (operations): un tipo deve sapere che operazioni possono essere applicate a un oggetto (cioè ai dati che lo rappresentano)



- Quasi tutte le macchine hanno uno stato
- Che stato potremmo definire nei seguenti casi?
 - Macchinetta del caffè bool spenta/accesa, bool in attesa/creando bevanda, ...
 - Tazza di caffè (!!?) bool piena/vuota, bool intera/rotta, bool pulita/sporca, bool risposta/in uso, double temperatura, ...
 - Motore di un'auto
 - Auto a combustione
 - Auto elettrica



Strumenti per definire tipi

- Classe
 - Rappresenta direttamente un concetto in un programma
 - Specifica:
 - Come sono rappresentati i relativi oggetti
 - Attenzione al significato di *oggetto*!
 - Come questi oggetti possono essere creati
 - Come questi oggetti possono essere usati
 - Come questi oggetti possono essere **distrutti**
 - Se pensiamo a qualcosa come a un'entità separata, probabilmente può essere mappata in una classe
- Enumerazioni

Classi



- Una classe è composta da:
 - Tipi built-in
 - Altri UDT
 - Funzioni
- Le entità utilizzate per definire la classe sono chiamati **membri**
 - Data member (dato membro)
 - Function member (funzione membro)



- Un esempio semplice

```
class X
{
public:
    int m;
    int mf(int v) { int old = m; m = v; return old; }
};
```



Accesso ai membri

- È possibile accedere ai membri con la notazione:
oggetto.membro
- Le funzioni membro non devono usare questa notazione

```
x var;  
  
var.m = 7;  
  
int x = var.mf(9);
```



Interfaccia vs implementazione

- Paradigma: interfaccia vs implementazione
- Naturale implementazione con le classi
- **Interfaccia (**public**)**
 - Punto di vista dell'utente
 - Accesibile agli utenti
- **Implementazione (**private**)**
 - Punto di vista dell'implementatore
 - Non direttamente accessibile
 - Lettura/scrittura solo tramite l'interfaccia
 - Permette il controllo dello stato (es., coerenza)
- Default: i membri di una class sono privati



Public & private

```
class X
{
public:
    // membri pubblici:
    // interfaccia utente
    // funzioni
    // tipi
    // eventuali dati (più probabilmente privati)

private:
    // membri privati
    // dettagli implementativi
    // funzioni
    // tipi
    // dati
};
```

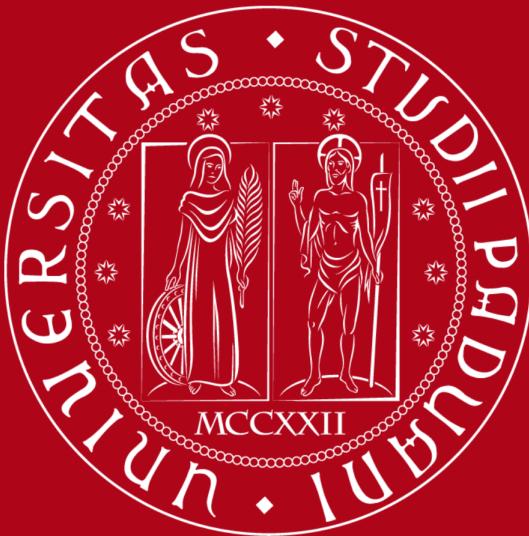


- Le struct sono varianti delle class
- Unica differenza tecnica: i membri di default sono pubblici
- Solitamente usate quando ci sono dati ma non funzioni
- Eredità del C



Progettazione di una classe

- Il C++ supporta la programmazione a oggetti con una serie di strumenti potenti
 - Funzioni membro
 - Costruttori
 - Protezione dei dati
 - Overloading degli operatori
 - Gestione della copia dei dati
 - (Helper function)



UNIVERSITÀ
DEGLI STUDI
DI PADOVA

Tipi definiti dall'utente – intro

Stefano Ghidoni