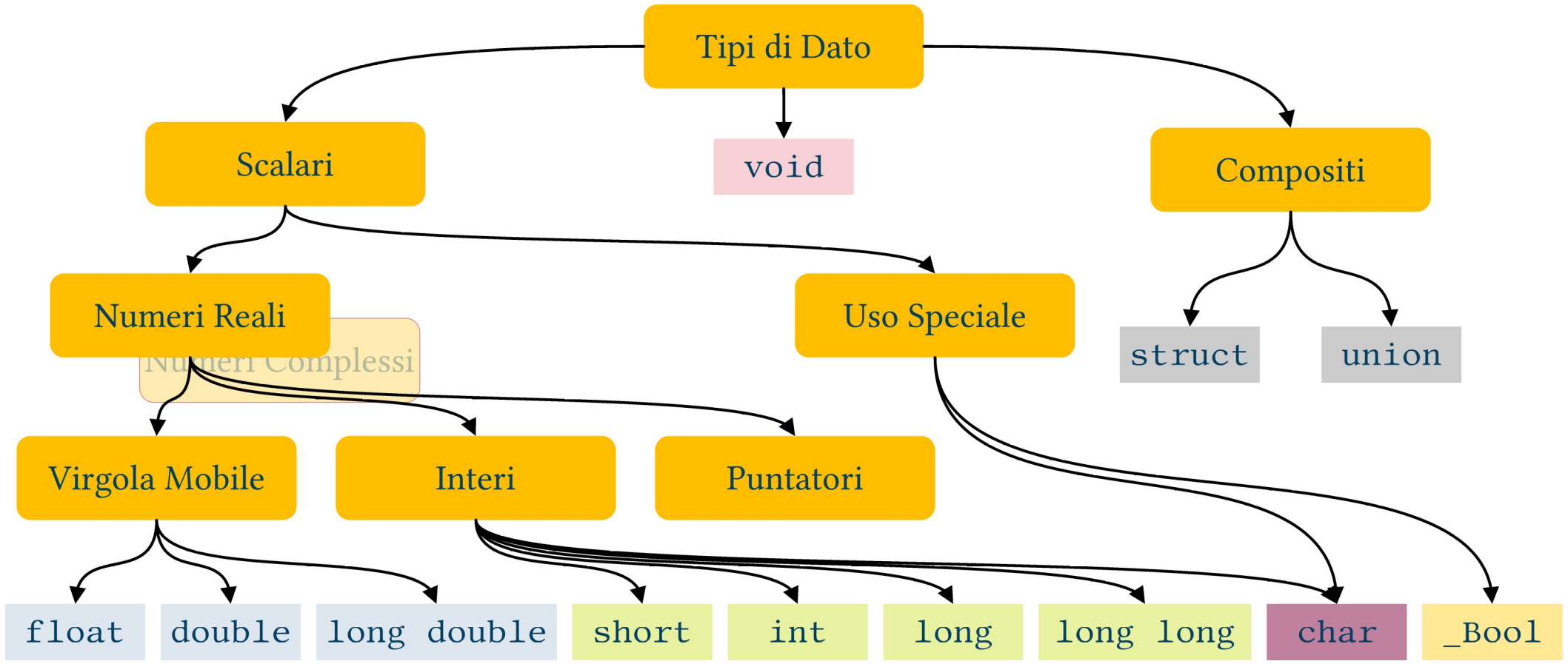
Sommario

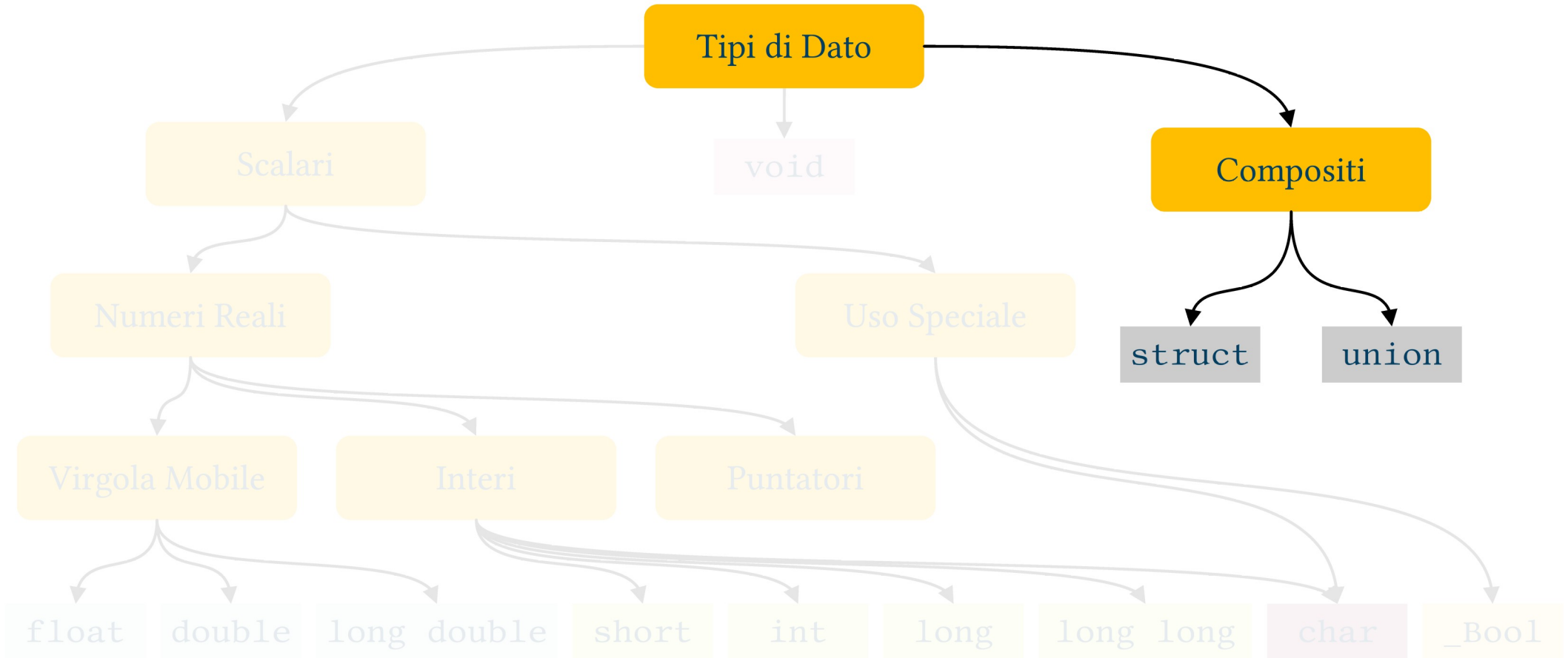
UNIVERSITÀ DI PARMA

- Struct
- ~~Union~~
- ~~Enum~~

C99 Albero dei Tipi di Dato



C99 Albero dei Tipi di Dato



Perché i dati compositi?

UNIVERSITÀ DI PARMA

- Fino ad ora:
  - Numeri di differenti tipologie e intervalli
  - Caratteri e sequenze di caratteri
- Inefficiente per modellare mondo reale
  - Esempio: anagrafica studenti, mi servono:
    - Nome e cognome ⟶ stringhe
    - Matricola ⟶ numero intero
    - ...

struct

UNIVERSITÀ DI PARMA

- Il costrutto struct permette di
  - Aggregare dati anche di tipo differente
  - Definire nuove tipologie di dato
- Esempio:

```
struct studente{
   char nome[100], cognome[100];
   unsigned long matricola;
};
```

struct

UNIVERSITÀ DI PARMA

- In questo esempio sto aggregando dati di tipo differente
- 2 array di char e un dato di tipo numerico intero
  - I singoli componenti si chiamano “elementi” o “membri”

```
struct studente{
   char nome[100], cognome[100];
   unsigned long matricola;
};
```

struct

UNIVERSITÀ DI PARMA

- La "`struct studente`" è un nuovo tipo di dato
  - NON è una variabile
- Posso usarla per definire variabili

```
struct studente{
   char nome[100], cognome[100];
   unsigned long matricola;
};

struct studente a, b, *c;
```

- Come accedo ai singoli elementi?
  - `.` per le variabili
  - `->` per i puntatori

```
struct studente a, b, *c;

a.matricola  = 12345;
c->matricola = 67890;
```

- È possibile inizializzare le variabili di tipo “struct”?
- Approccio simile agli array
- Devo mantenere ordine elementi
- Inizializzazioni parziali ⟶ 0

```
struct studente a={“Gino”, “Pilotino”, 12345};
```

- Il C99 introduce anche altro tipo di sintassi
  - Indico nome elementi
  - Ordine non obbligatorio

```
struct studente a={
.nome=“Gino”,
.cognome=“Pilotino”,
.matricola=12345};
```

Strutture nidificate

UNIVERSITÀ DI PARMA

- Le struct possono contenere altre struct

```
struct anagrafica{
  char nome [100], cognome[100];
  int giorno, anno, mese;
};

struct studente{
   struct anagrafica dati;
   char matricola[12];
};
```

Strutture nidificate

UNIVERSITÀ DI PARMA

- Particolare importanza rivestono le struct che contengono puntatori allo stesso tipo di dato
  - Liste, alberi, grafi ecc.
  - Non le vedremo

```
struct nodo{
  char nome [100], cognome[100];
  struct nodo *successivo;
};
```

Struct e occupazione memoria

UNIVERSITÀ DI PARMA

- Gli elementi/membri di una struct sono memorizzati nella stessa area di memoria
- Ma non necessariamente in maniera contigua
- Padding
  - Permette uso piú efficiente memoria
  - Tipicamente allineo elementi struct ad indirizzi multipli di un determinato numero di byte (2, 4 ecc.)

union

- Stessa sintassi struct
- Ma: elementi tutti sovrapposti in memoria
  - Sistemi embedded
  - Gestione HW
  - Flessibilità

```
union nodo{
  int    intero;
  double floating;
} mixed_array[100];
```

enum

- Costrutto per definire tipicamente elenchi di valori costanti
  - Di fatto di tipo “int”
- Spesso usato come argomento funzioni

```
enum stagioni{primavera, estate, autunno, inverno};

enum stagioni a;

a = primavera;
```