TIPI ENUMERATIVI

Un ***tipo enumerativo*** è un tipo di dato che consiste in un insieme ristretto di valori.

Ogni elemento appartenente al tipo ha un proprio nome (identificatore).

I nomi associati agli elementi del tipo sono trattati come costanti, chiamate costanti enumerative.

I tipi enumerativi sono compatibili con un dato intero.

La loro funzione è semplicemente quella di rendere più leggibile il codice.

TIPI ENUMERATIVI

Un ***tipo enumerativo*** viene specificato tramite ***l'elenco dei valori*** che i dati di quel tipo possono assumere

Schema generale:

```
typedef enum {
     a1, a2, a3, ... , aN } EnumType;
```

Il compilatore associa a ciascun "identificativo di valore" a_1, .., a_N un *numero naturale* (0,1,...), che viene usato nella valutazione di espressioni che coinvolgono il nuovo tipo

TIPI ENUMERATIVI

Gli "identificativi di valore" $a_1, \ldots, a_N$
sono a tutti gli effetti delle ***nuove costanti***

Esempi:

```
typedef enum {
  lu, ma, me, gi, ve, sa, dom} Giorni;
typedef enum {
  cuori, picche, quadri, fiori} Carte;
Carte   C1, C2, C3, C4, C5;
Giorni  Giorno;
if (Giorno == dom) /* giorno festivo */
else /* giorno feriale */
```

TIPI ENUMERATIVI

Analogamente si può utilizzare un tag:

```
enum Giorni{
  lu, ma, me, gi, ve, sa, dom};
enum Carte {
  cuori, picche, quadri, fiori};
enum Carte      C1, C2, C3, C4, C5;
enum Giorni     Giorno;
```

Oppure direttamente la definizione delle variabili:

```
enum {
  lu, ma, me, gi, ve, sa, dom } Giorno;
enum {
  cuori, picche, quadri, fiori} C1, C2, C3, C4, C5;
```

TIPI ENUMERATIVI

Un "identificativo di valore" può comparire *una sola volta* nella definizione di *un solo tipo*, nello stesso scope di visibilità, altrimenti si ha ambiguità

Esempio:

```
typedef   enum {
  lu, ma, me, gi, ve, sa, dom} Giorni;

typedef enum { lu, ma, me} PrimiGiorni;
```

La definizione del secondo tipo enumerativo è ***scorretta***, perché gli identificatori `lu, ma, me` sono già stati usati altrove.

TIPI ENUMERATIVI

Un tipo enumerativo è *totalmente ordinato:*
vale l'ordine con cui gli identificativi di valore sono stati elencati nella definizione

Esempio:

```
typedef enum {
  lu, ma, me, gi, ve, sa, dom} Giorni;
```

Data questa definizione,

`lu < ma` *è vera*

`lu >= sa` *è falsa*

in quanto `lu` ↔ 0, `ma` ↔ 1, `me` ↔ 2, ...

TIPI ENUMERATIVI

Poiché un tipo enumerativo è, *per la macchina C,* indistinguibile da un intero, è possibile, anche se sconsigliato, ***mescolare interi e tipi enumerativi***

Esempio:

```
typedef enum {
   lu, ma, me, gi, ve, sa, dom} Giorni;
Giorni g;
g = 5;          /* equivale a g = sa */
```