TIPI DI DATO IN C

Tipicamente un elaboratore è capace di trattare domini di dati **di tipi primitivi**

- numeri naturali, interi, reali
- caratteri e stringhe di caratteri

e quasi sempre anche collezioni di oggetti, mediante la definizione **di tipi strutturati**

- array, strutture

Spesso un linguaggio di programmazione permette di introdurre **altri tipi definiti dall'utente**

TIPI DEFINITI DALL'UTENTE

In C, l'utente può introdurre nuovi tipi tramite **una definizione di tipo**

La definizione associa a un identificatore (nome del tipo) un tipo di dato

- aumenta la leggibilità del programma
- consente di ragionare per astrazioni

Linguaggio C consente in particolare di:

- ridefinire tipi già esistenti
- definire dei nuovi tipi enumerativi
- definire dei nuovi tipi strutturati

TIPI RIDEFINITI

Un nuovo identificatore di tipo viene dichiarato identico a un tipo già esistente

Schema generale:

typedef TipoEsistente NuovoTipo;

Esempio:

```
typedef int MioIntero;
MioIntero x,y,z;
int w;
```

TIPI ENUMERAIVI

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

Schema generale:

typedef enum { a_1, a_2, a_3, ... , a_N } 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.

Gli "identificativi di valore" a_1,... , a_N sono a tutti gli effetti delle nuove costanti

Esempio:

```
typedef enum { lu, ma, me, gi, ve, sa, dom} Giorni;
typedef enum { cuori, picche, quadri, fiori} Carte;

Carte c1, c2, c3, c4, c5;
Giorni g;

if (g == dom) /* giorno festivo */
……
else /* giorno feriale */
…..
```

Un "identificativo di valore" può comparire una sola volta nella definizione di un solo tipo, 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

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, etc. etc.

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 */

È anche possibile specificare esplicitamente i valori naturali cui associare i simboli $a_1,\ldots,a_N$

• qui, lu = 0, ma = 1, me = 2, etc. etc.
typedef enum { lu, ma, me, gi, ve, sa, dom} Giorni;

• qui, invece, lu = 1, ma = 2, me = 3, etc. etc.
typedef enum { lu=1, ma, me, gi, ve, sa, dom} Giorni;

• qui, infine, l'associazione è data caso per caso typedef enum { lu=1, ma, me=7, gi, ve, sa, dom} Giorni;
N.B. quindi ma = 2 mentre gi = 8, ve = 9, etc. etc.

Il tipo booleano non esiste in C, ma si può facilmente definire in termini di tipo enumerativo