Il programmatore che c'e' in noi – Lezione 9– Strutture, Unioni, Enumerazioni

Tutti abbiamo un carta d'identita'. (la mia e' scaduta, tra l'altro…)

La nostra carta di identita' contiene un insieme di dati di diverso tipo e di diverse dimensioni.
Sono presenti dati quali Cognome (stringa o array di caratteri) Nome (stringa) Comune di Nascita (stringa) Data di nascita (immaginiamola divisa in tre interi, anno, mese, giorno) Altezza (double) ecc.

La cosa veramente importante e' che in un solo tipo di dato (la carta di identita') sono state raggruppate diverse informazioni (dati) ma che sono in qualche modo "logicamente" collegate tra di loro.

Pensate se invece di avere un unico documento con tutti i dati identificativi, avessimo un documento per ciascun dato (una carta che specifica il Nome, una il Cognome, una dove siamo Nati …) mentre e' estremamente comodo gestire un solo documento che contenga tutte le informazioni necessarie per l'identificazione di una persona.

Bene, la carta di identita' e' una STRUTTURA DATI.

Una struttura e' una collezione di variabili, anche di tipo e dimensioni differenti, che possono essere gestite tramite un solo nome.
In modo da garantire un modo semplice ed immediato il legame logico che i dati, presenti nelle singole variabili, hanno tra di loro e per semplificarne la gestione.

Per definire una struttura dati, si utilizza la keyword **struct** propria del linguaggio C.

Le sintassi sono:

```
//DICHIARAZIONE DELLA STRUTTURA (come e' composta, ma non crea nessuna variabile)
struct NOMESTRUTTURA {
TIPO NOMEVARIABILE;
TIPO NOMEVARIABILE;
…
};

//DICHIARAZIONE DI UNA VARIABILE CHE USA LA STRUTTURA
struct NOMESTRUTTURA NOMEVARIABILE;
```

In alternativa e' possibile dichiarare sia la struttura sia la/le variabili che saranno di tale tipo in una sola sintassi

```
struct NOMESTRUTTURA {
TIPO NOMEVARIABILE;
TIPO NOMEVARIABILE;
…
}ELENCONOMIVARIABILI; //GLI ELENCHI SONO SEPARATI DA VIRGOLA
```

Se, abbiamo bisogno di gestire 100 carte di identita', conviene usare un ARRAY DI STRUTTURA

```
struct NOMESTRUTTURA {
```

TIPO NOMEVARIABILE;
TIPO NOMEVARIABILE;
…
}NOMEARRAY[DIMENSIONE]; //DICHIARO un certo numero (DIMENSIONE) di strutture accessibili tramite indice dell'array NOMEARRAY,

Facciamo un po' di esempi .

Creiamo ad esempio la struttura per "simulare" la nostra carta d'identita'.

ESEMPIO (1)

```
//Dichiaro come e' costituita la struttura
struct cartaIdentita{
  char nome[101]; //stringa per contenere al massimo 100 caratteri (+1 = NUL)
  char cognome[101];
  char comuneResidenza[51];
  int nAnnoNascita; //Anno aaaa
  int nMeseNascita; // Mese mm
  int nGiornoNascita; //Giorno gg
  double nAltezza; //Altezza in metri,centimetri
};

//Mi serve una carta di identita', creo la variabile
struct cartaIdentita CartaIdentitaFranco;
```

Ogni volta che nel programma avro' bisogno di una nuova carta di identita' da gestire, sara' sufficiente scrivere

```
struct cartaIdentita CartaIdentitaMarco;
```

In alternativa, se so gia' prima che gestiro' solo un certo numero di carte di identita', posso dichiarare tutto in una sola volta,quindi

ESEMPIO (2)

```
//Dichiaro come e' costituita la struttura e due variabili di tale tipo di dato (cartaIdentita)
struct cartaIdentita{
  char nome[101]; //stringa per contenere al massimo 100 caratteri (+1 = NUL)
  char cognome[101];
  char comuneResidenza[51];
  int nAnnoNascita; //Anno aaaa
  int nMeseNascita; // Mese mm
  int nGiornoNascita; //Giorno gg
  double nAltezza; //Altezza in metri,centimetri
} CartaIdentitaFranco, CartaIdentitaMarco;
```

Nel caso in cui, abbiamo bisogno di gestire 1000 carte di identita', dichiaro un array

ESEMPIO (2)

```
//Dichiaro un array di strutture, oltre alla definizione della struttura stessa
struct cartaIdentita{
  char nome[101]; //stringa per contenere al massimo 100 caratteri (+1 = NUL)
  char cognome[101];
  char comuneResidenza[51];
  int nAnnoNascita; //Anno aaaa
  int nMeseNascita; // Mese mm
  int nGiornoNascita; //Giorno gg
  double nAltezza; //Altezza in metri,centimetri
} carteIdentitaClienti[1000];  //Dichiaro un array di 1000 strutture di tipo cartaIdentita
```

Rappresentazione in memoria della nostra variabile di tipo struttura.
Ogni variabile quando viene stanziata (allocata) per poter essere utilizzata all'interno di un programma,occupa una determinata quantita' di RAM (in funzione del tipo di dato) e viene posizionata in una specifica locazione della nostra memoria ram.

Cosa avviene quando definisco in dato di tipo struct e creo un istanza di tale variabile:
Dove e' posizionata in memoria ? Quanta memoria occupa ?

Analizziamo l'ESEMPIO (1).

La variabile CartaIdentitaFranco sara' posizionata in memoria in una qualche locazione (possiamo sapere quale usando l'operatore & per farci restituire l'indirizzo di memoria)

Quindi per conoscere la locazione di memoria in cui si trova la nostra struttura in questo modo
NSLog("Indirizzo della variabile : %p",& CartaIdentitaFranco);

Per conoscere la dimensione, uso sizeof()
NSLog("Dimensione della variabile: %d", sizeof(CartaIdentitaFranco));

Accesso alle variabili contenute nella struttura (**VARIABILI MEMBRO**)

Per accedere ai membri della struttura (le variabili che la compongono) si utilizza l'operatore . (punto)

NOMEVARIABILESTRUTTURA.NOMEVARIABILEMEMBRO

Esempio
Per compilare i dati della struttura CartaIdentitaFranco

Dato che alcuni membri della struttura sono delle stringhe C (array di caratteri), uso la funzione standard C per assegnare un testo ad una variabile stringa ossia
strcpy(char *strDestinazione,char *strOrigine)

strcpy(CartaIdentitaFranco.nome,"Francesco"); //Assegno la stringa Francesco alla variabile membro nome, per accedere alla variabile nome uso operatore .

strcpy(CartaIdentitaFranco.cognome,"Germinara");
strcpy(CartaIdentitaFranco.comuneResidenza,"Pinerolo");

CartaIdentitaFranco.nAnnoNascita=1966; //Assegno il valore 1966 alla variabile nAnnoNascita di tipo INT
CartaIdentitaFranco.nMeseNascita=2; //Assegno il mese
CartaIdentitaFranco. nGiornoNascita =20; //Assegno il giorno
CartaIdentitaFranco.nAltezza=1.72; //Assegno altezza

Per visualizzare il contenuto della variabile CartaIdentitaFranco

NSLog("Carta di identita': Cognome: '%s' Nome: '%s' Residente: '%s' Nato il: %02d/%02d%04d Alto: %6.2f", CartaIdentitaFranco.nome, CartaIdentitaFranco.cognome, CartaIdentitaFranco. comuneResidenza, CartaIdentitaFranco. nGiornoNascita, CartaIdentitaFranco. nMeseNascita, CartaIdentitaFranco. nAnnoNascita, CartaIdentitaFranco. nAltezza);

Per accedere alle variabili di un array di struttura, e' sufficiente indicare l'indice dell'array tra parentesi quadre, prima di usare l'operatore .

Esempio
#define MAX_PERSONE 1200 //definisco una costante numerica
struct cartaIdentita elencoVotanti[MAX_PERSONE]; //Istanzio un array di strutture cartaIdentita

//Per accedere ai dati della prima carta di identita' uso indice 0, quindi
strcpy(elencoVotanti[0].nome,"Marco");
strcpy(elencoVotanti[0].nome,"Coisson"); //spero non me ne voglia Marco per aver usato i dati
                                         //in queso esempio
elencoVotanti[0]. nAltezza = 1.74; //Non ne ho idea… invento… J
…

//Per accedere ai dati dell'ultima carta di identità la 1199 esima.
strcpy(elencoVotanti[MAX_PERSONE-1].nome,"Franco");
