Laboratorio di Programmazione 1

Docente: dr. Damiano Macedonio

Lezione 15 – 10/03/2014

Le Strutture

- La *struttura* è un costrutto del linguaggio C che permette di raggruppare elementi di diverso tipo in un'unica entità logica.
  - Gli array sono uno strumento analogo, ma permettono di raggruppare solo elementi dello *stesso tipo*.

- Le strutture definiscono un nuovo tipo di dati: si possono dichiarare variabili di tipo `struct` *`nome_struttura`*.

- La definizione di una struttura:
  - Inizia con la parola chiave `struct`, seguita dal nome della struttura e da una lista di campi tra parentesi graffe.
  - Ciascun campo viene specificato tramite un tipo ed un nome.
  - I campi sono separati da un punto e virgola.

Le Strutture: Esempio

Le Strutture

- Per accedere ad un campo di una struttura, occorre specificare il nome della variabile, seguito da un punto e dal nome del campo.

```
struct date today;
today.day = 10;
today.month = 3;
today.year = 2014;
if (today.day == 1 && today.month == 1)
    printf("Buon anno!\n");
```

impostazione valore

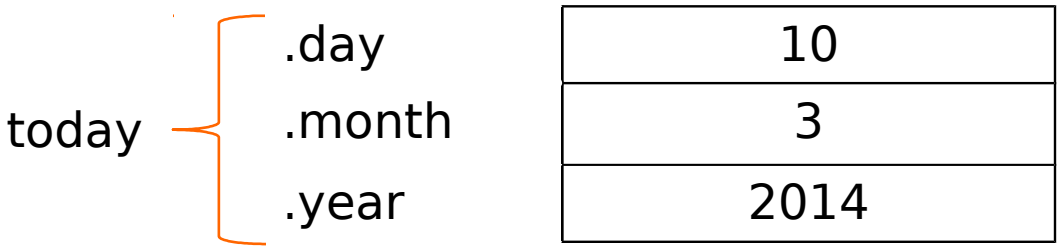
lettura valore

- I campi di una struttura possono essere usati nelle espressioni come una qualsiasi altra variabile semplice.

```
tomorrow.day = today.day + 1;
```

Le Strutture: Allocazione di Memoria

- La *dichiarazione di una struttura* comunica al compilatore come è fatta la struttura.
- La *dichiarazione di una variabile di tipo struttura* alloca in memoria lo spazio necessario a memorizzare i campi della struttura.



Le Strutture: Visibilità

- Le strutture definite prima e all'esterno di qualsiasi funzione sono dette *globali*.
  - La definizione di una struttura globale permette a tutte le variabili che sono definite nel programma di essere dichiarate con il tipo della struttura.

- Le strutture definite all'interno di una particolare funzione sono dette *locali*.
  - Le strutture locali sono visibili e posso essere utilizzate solo all'interno della funzione che le dichiara.

Le Strutture: Visibilità

```
struct date { //struttura globale
  int day;
  int month;
  int year;
}


int main(void) {
  struct date today;
  ...
}
```

```
int function(void) {
  struct time { //struttura locale
    int hour;
    int minutes;
  }
  struct date tomorrow;
  struct time now;
}
```

Le Strutture: Argomenti di Funzioni

- Le strutture possono essere utilizzate come *argomenti* di funzioni:
  - `int numberOfDays(struct date d);`
  - La funzione `numberOfDays` restituisce un valore intero e riceve come unico argomento una variabile di tipo `struct date`.

- Qualsiasi modifica apportata dalla funzione ai valori contenuti in un campo della struttura *non* ha effetto sulla struttura originale.
  - Tali modifiche hanno effetto *solo* sulla copia della struttura che viene creata alla chiamata di funzione.
  - Quando passate come argomento di una funzione le strutture si comporatano come variabili ordinarie/semplici e non come array.

Le Strutture: Argomenti di Funzioni

```
void changeDate(struct date d) {
  d.day = 24;
  d.month = 6;
}

int main(void) {
  struct date d;
  d.day = 10;
  d.month = 3;
  d.year = 2014;

  changeDate(d);
  printf("Data di oggi: %i/%2i/%i",
            d.day, d.month, d.year)
}
```

Output
Data di oggi: 10/03/2014

Le Strutture: Valore di Ritorno di Funzioni

- Le strutture possono essere utilizzate come *valori di ritorno* di funzioni.

```
struct date changeDate(struct date d) {
    struct date result;
    result.day = d.day + 1;
    result.month = d.month + 1;
    result.year = d.year + 2;
    return result;
}

struct d1 = ....;
struct d2 = changeDate(d1);
```

Le Strutture: Inizializzazione

- Le strutture possono essere inizializzate in modo simile agli array: gli elementi vengono elencati tra parentesi graffe e separati da una virgola.

```
struct date d;
d.day = 10;
d.month = 3;
d.year = 2014
```

```
struct date d = {10, 3, 2014}
```

- I valori iniziali indicati tra le parentesi graffe devono essere *espressioni costanti*.

Le Strutture: Inizializzazione

- Quando si utilizza l'inizializzazione tramite parentesi graffe, è possibile specificare meno valori di quelli contenuti nelle struttura (come per gli array!).
  - `struct date d = {10, 3};`
  - In questo caso `d.day = 10`, `d.month = 3`, mentre `d.year` non viene inizializzato.
- È possibile anche specificare i nomi dei campi nella lista di inizializzazione, usando la notazione `.nome_campo = valore`
  - `struct date d = {.month = 3, .year = 2014};`
  - In questo modo è possibile inizializzare i campi in qualsiasi ordine o inizializzare campi specifici.

Le Strutture: Assegnamento Composto

- La notazione con parentesi graffe può essere utilizzata non solo per l'inizializzazione di una struttura, ma anche per assegnare (successivamente) uno o più valori ad una struttura usando un'unica istruzione.

```
struct date today;
...
today = (struct date) {10, 3, 2014};
```

- `{10, 3, 2014}` è detto *letterale composto*.
- L'operatore di cast `(struct date)` è necessario per indicare al compilatore il tipo dell'espressione.
- È possibile usare anche la notazione `.`*`campo`*.