COESIONE ITALIA 21-27 PIEMONTE

Cofinanziato dall'Unione europea

REGIONE PIEMONTE

WEB DEVELOPER
Fondamenti di Programmazione

Massimo PAPA

LE TABELLE

LE TABELLE

- Abbiamo visto che la struttura ci permette di gestire più elementi non omogenei.
- Mentre l'array ci permette di gestire elementi omogenei tra di loro.
- Supponiamo di avere più record definiti da una stessa struct.
- Potremmo allora pensare di memorizzare tali record in un'array.

LE TABELLE

- Andremo a definire quindi un'array di struct
- Un'array di struct viene solitamente chiamato **tabella**.
- Infatti si può schematizzare come una tabella con tante colonne quanti sono i campi della struttura e un numero di righe pari al numero degli elementi dell'array.
- Nella prossima slide vediamo un esempio di strutture dati per la gestione di una biblioteca.

LE TABELLE - Es. BIBLIOTECA

- Dobbiamo gestire i libri di una biblioteca.
- Potremmo pensare di memorizzare le informazioni dei libri della biblioteca in un array.
- Il libro ha queste caratteristiche peculiari:

  ➔ Titolo
  ➔ Autori
  ➔ Anno Edizione
  ➔ Luogo Edizione
  ➔ Casa Editrice
  ➔ Data di acquisto

LE TABELLE - Es. BIBLIOTECA

- Queste caratteristiche peculiari “scaturiscono” dall’analisi dello specifico problema che ci accingiamo a risolvere
- Se p.e. andassimo a modellizzare il libro di un venditore di libri usati, potremmo pensare che caratteristiche fondamentale siano lo stato del libro, il prezzo, etc,etc
- Per quanto riguarda la data abbiamo già visto che può essere a sua volta una struct.
- Andiamo a definire in C++ tutte le strutture dati che ci servono.

LE TABELLE - Es. BIBLIOTECA

```
struct Data {
    int gg;
    int mm;
    int aa;
};

struct Libro {
    string titolo;
    string autori[3];
    int annoEd;
    string luogoEd;
    string casaEd;
    Data dataAcquisto;
};

#define NMAXLIBRI 1000

Libro biblioteca[NMAXLIBRI]
```

Struttura per gestire le date

Campo composto da un array di 3 elementi (gli autori possono essere più di uno)

Campo avente il tipo Data definito nella precedente struttura

L'intera bibliotea è definita come un array di NMAXLIBRI elementi, ciascuno dei quali ha la struttura di *Libro*

LE TABELLE - Es. BIBLIOTECA

- Con queste definizioni di dati, nel programma è consentito l'uso delle seguenti istruzioni:

```
biblioteca[10].autori[0] = "Paperino";

biblioteca[40].AnnoEd = 2004;

biblioteca[5].dataAcquisto.mese = 7;
```

- La prima istruzione assegna il nome del primo autore al libro in posizione 10.
- La seconda assegna l'anno di edizione al 41^{esimo} libro.
- La terza assegna al sesto libro il mese di acquisto luglio.

LE TABELLE - Es. DIPENDENTE

- La struttura più adatta per rappresentare le informazioni di un dipendente di una ditta che percepisce 13 mensilità, potrebbe essere:

```
struct Persona{
   string nome;
   string indirizzo;
   int livello;
   float stipendio[13];
};
```

- Per rappresentare le informazioni relative a tutti i dipendenti che la ditta ha assunto, si può utilizzare un array di elementi `Persona`

```
#define NMAXDIPENDENTI 200
Persona dipendente[NMAXDIPENDENTI];
```

LE TABELLE - Es. Gara Campestre

- Affrontiamo un semplice problema.
- Si vuole gestire la classifica di una gara campestre di corsa.
- Alla fine di una gara campestre vengono memorizzati, per ogni partecipante, il numero di pettorale, il nome e il tempo impiegato espresso in ore, minuti e secondi. Si vuole la stampa dei partecipanti ordinati rispetto al tempo impiegato.
- Il problema si divide in tre parti:
    1. acquisizione dei dati
    2. ordinamento
    3. stampa

Es. Gara Campestre - Input dati

- L'acquisizione dei dati prevede la memorizzazione di questi in una tabella.
- Cioè in un array di strutture: ogni riga della tabella è composta da una struttura che contiene i dati di un partecipante, vale a dire:
  - ➔ il numero di pettorale
  - ➔ il nome
  - ➔ il tempo impiegato

Es. Gara Campestre - Input dati

- A sua volta il tempo impiegato è definito da una struttura contenente i seguenti campi:
  - ➔ ore (hh)
  - ➔ minuti (mm)
  - ➔ secondi (ss)
  - ➔ totale secondi

- Il totale secondi ottenuto dalla valutazione dell'espressione:

  totsecondi = hh*3600+mm*60+ss

Es. Gara Campestre - Stampa

- La stampa dei dati consiste nel trasferire in output i dati della tabella ordinata
- Il dispositivo di output è lo standard output
- La stampa avviene mediante un ciclo che scansiona tutti gli elementi dell'array, dalla prima all'ultima.
- Ad ogni elemento letto dell'array vengono estratte le informazioni della struttura che si desidera stampare.

Es. Gara Campestre - Le strutture

```
//definizione delle strutture dati
struct Tempo{
   int hh;
   int mm;
   int ss;
   int totSecondi;
};

struct Partecipante{
   int pettorale;
   string nome;
   Tempo tempoImpiegato;
};

#define NMAXPARTECIPANTI 200
Partecipante tabella[NMAXPARTECIPANTI];
```

Es. Gara Campestre - Function prototype

```
//definizione dei prototipi delle funzioni

//Input del numero dei partecipanti alla gara
int chiediNumPartecipanti();

//Input delle informazioni dei partecipanti
void caricaTabella(Partecipante tab[], int N);

//Funzione che effettua l'ordinamento
//a secondo del tempo di arrivo
void ordina(Partecipante tab[], int N);

//Stampa a video dei risultati della gara
void stampa(Partecipante tab[], int N);
```

Es. Gara Campestre - main function

```
//Funzione principale
int main(){
   int N; //Numero di partecipanti
   // Tabella dei partecipanti alla gara
   Partecipante tabella[NMAXPARTECIPANTI];

   //Inserimento dati
   N = chiediNumPartecipanti();
   caricaTabella(tabella, N);

   //Ordinamento
   ordina(tabella, N);

   //Stampa
   stampa(tabella, N);

   return 0;
}
```

Es. Gara Campestre - Il codice completo

- Ora esaminiamo il codice completo