UNIVERSITY OF TRENTO - Italy
Department of Information
and Communication Technology

Laboratorio 8

Pierluigi Roberti
Carmelo Ferrante

DISI – aa 2024-25
Università degli Studi di Trento
pierluigi.roberti@unitn.it

Generazione valori casuali

Ricordarsi di re-impostare il generatore dei numeri casuali:

```
srand(time(0));
```

Se richiesto includere la libreria

```
#include <time.h>
```

- **Interi** => esempio per intervallo [18,30]

```
int max=30, min=18;
int valInt = rand()%(max-min+1)+min;
```

- **Caratteri** => esempio per intervallo [‘d’ – ‘w’]

```
char max='w', min='d';
char valCar = rand()%(max-min+1)+min;
```

- **Floating point** => esempio per intervallo [2.00 – 25.00]

```
int max=2500, min=200;
float valFlo = (float)( rand()%(max-min+1)+min )/100;
```

Uso parola chiave typedef

Creazione di un alias di tipo (iniziale maiuscola)

Sintassi:

```
typedef tipo NomeNuovoTipo;
```

Esempio

```
typedef int Lunghezza;
typedef char Stringa[30];
```

Dichiarazione variabili basate sul nuovo tipo

```
Lunghezza l;
Stringa nome, cognome;
```

Uso di typedef e struct

Dichiarazione tipo di dato basato su struttura NON anonima

```
typedef struct NomeTipo {
  Tipo1 nomeCampo1;
  ...
  TipoN nomeCampoN;
} NomeTipo;
```

Il **nuovo tipo** di dato viene messo **prima del main** in modo che sia «visibile/usabile» in ogni punto del programma!

Dichiarazione variabile basata sul nuovo tipo di dato definito

```
NomeTipo s1;
s1.nomeCampo1 = ...;
```

Esercizio 1-parte1

- Definire con typedef un tipo di dato strutturato denominato ***Tacquirente*** contenente:
  - nome array di **MAX_STR** caratteri
  - cognome array di **MAX_STR** caratteri
- Definire con typedef un tipo di dato strutturato denominato ***Tvettura*** contenente :
  - marca array di **MAX_STR** caratteri
  - cilindrata di tipo int
  - anno di tipo int (anno di immatricolazione)
  - acquirente di tipo ***Tacquirente***
- Definire con typedef un tipo di dato strutturato denominato ***Tsalone*** contenente :
  - n_vetture di tipo int (numero effetivo di vetture nel salone)
  - vetture array di **MAX_AUTO** di tipo ***Tvettura***

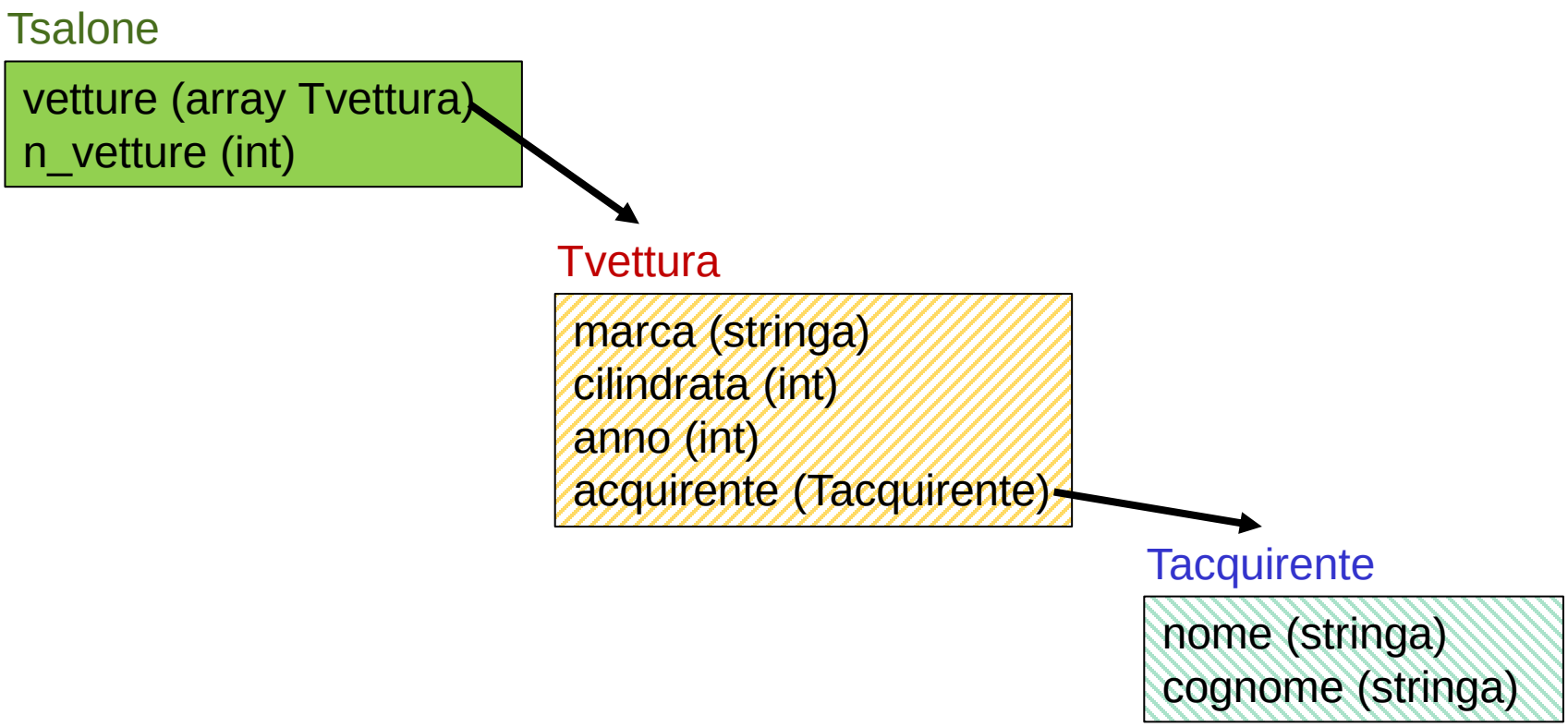
- Definire inoltre le seguenti “costanti” (**#define**):

MAX_AUTO pari a 100 (Numero massimo di vetture)

MAX_STR pari a 21 (nomi di lunghezza massima di 20 caratteri)

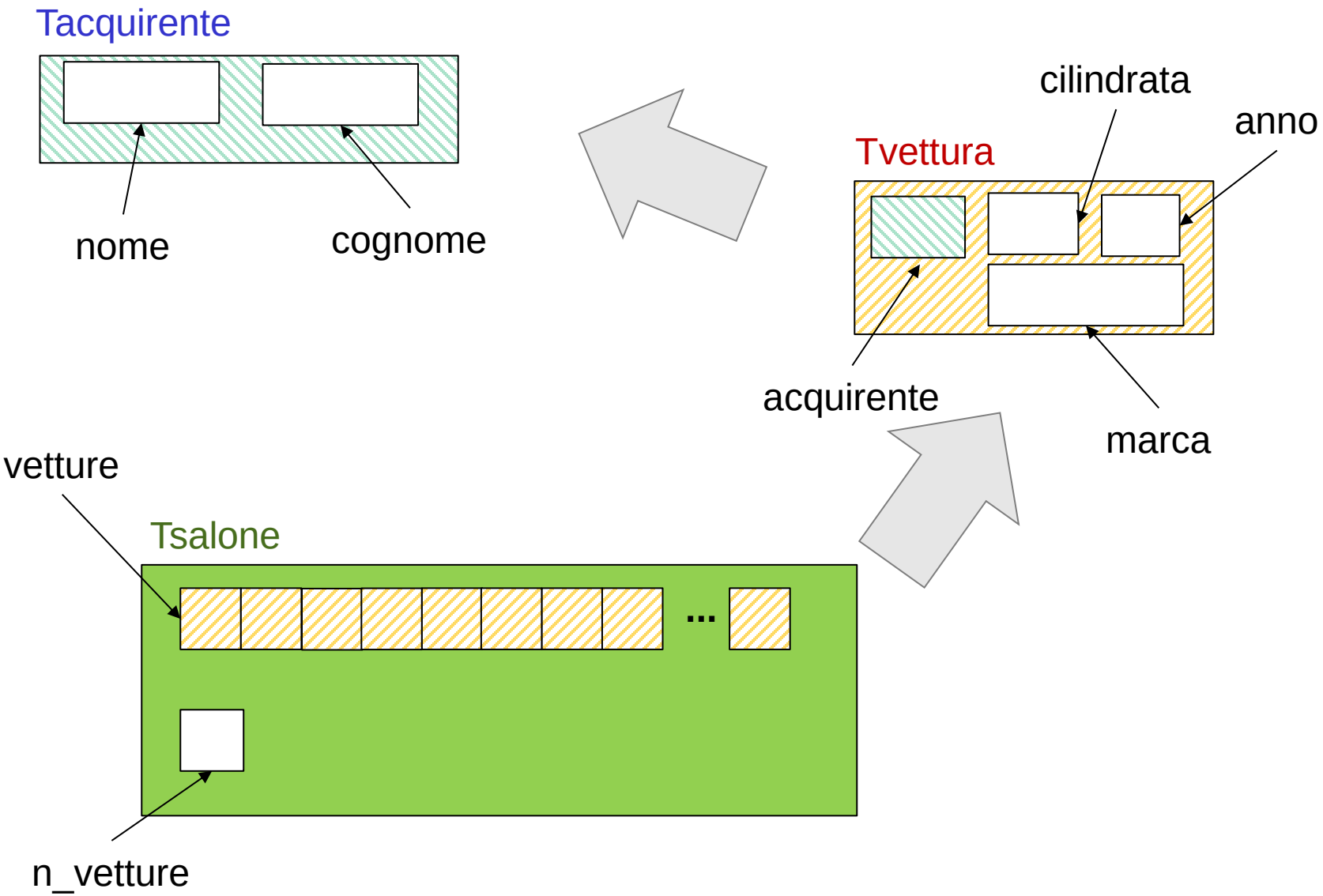
Esercizio 1

In pratica si deve fare un array di struct che contengono altre struct (l'acquirente è un campo della vettura).



- All'interno del main, dichiarare una variabile **salone** di tipo ***Tsalone***

Esercizio 1


Tacquirente
nome
cognome
cilindrata
anno
Tvettura
acquirente
marca
vetture
Tsalone
...
n_vetture

Esercizio 1-parte2

- Inizializzazione delle strutture:
  - Chiedere all'utente il numero di vetture (modificare il campo **`n_vetture`**)
    - Controllo input: dato compreso tra 3 e MAX_AUTO
  - Per ogni vettura da considerare nella variabile `salone`
    - → 1 ciclo FOR
    - Inizializzare campi marca, cilindrata, anno, nome, cognome
    - Inizializzare con valori casuali oppure chiedendo all'utente i dati

- Calcolare e visualizzare
  - Il solo cognome degli acquirenti di auto di cilindrata superiore a 1500
  - Il numero totale di auto che sono state immatricolate nell'anno 2000
  - Il cognome dell'acquirente con l'auto di cilindrata massima
  - → 1 ciclo FOR

Esercizio 2-parte1

- Definire con typedef un tipo di dato strutturato denominato ***Telemento*** contenente:
  - c di tipo carattere
  - val di tipo float
- Definire con typedef un tipo di dato strutturato denominato ***Tdato*** contenente:
  - m di tipo array NR x NC di tipo ***Telemento***
- Definire inoltre le seguenti “costanti”:

NR pari a 25 (Numero massimo di righe)

NC pari a 10 (Numero massimo di colonne)

W pari a 90 (Numero elementi da inizializzare nella matrice)

Esercizio 2

```
#define NR 25
#define NC 10
```

In questo caso è possibile aggiungere altre informazioni (campi) alla struttura Tdato

```
typedef struct Telemento{
    char c;
    int val;
}Telemento;
typedef struct Tdato{
    Telemento m[NC][NC];
}Tdato;

// Nella funzione main
Tdato mat;

// esempio accesso ai campi
mat.m[0][5].c = 'k';
mat.m[0][5].val = 12;
```

E8.3 Esercizio2- struct_con_matrice.c

```
typedef struct Telemento{
    char c;
    int val;
}Telemento;
typedef Telemento Tdato[NR][NC];

// Nella funzione main
Tdato mat;
// equivale a
// Telemento mat[NR][NC];

// esempio accesso ai campi
mat[0][5].c = 'k';
mat[0][5].val = 12;
```

E8.3 Esercizio2-matrice_di_struct.c

Esercizio 2-parte2

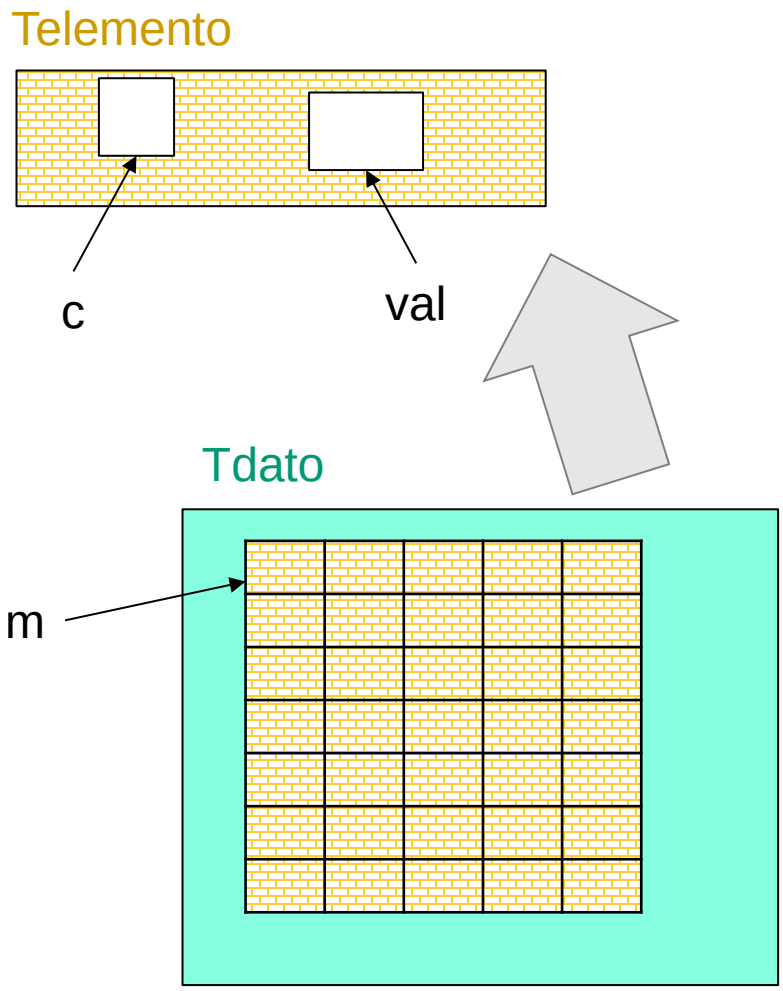
- Definire nel main una variabile **mat** di tipo ***Tdato***
- Inizializzare gli elementi della matrice impostando il campo **`c`** pari a ‘!’
- Inizializzare W elementi “validi” in modo casuale:
  - cercare una posizione libera:
    - generazione casuale posizione i, j fino a che il campo **`c`** è pari a ‘!’
  - nella posizione trovata
    - inizializzare **`c`** con un carattere casuale [‘c’, ‘x’]
    - inizializzare **`val`** con un numero casuale [-10.00, 15.00]

Esercizio 2-parte3

- Stampare a video la matrice inizializzata
  - Se l'elemento non e' "valido" (**`c`** contiene '!') → stampare #
  - Altrimenti → stampare il contenuto di **`c`** e **`val`** nel formato: [**`c val`**] (separati da uno spazio)
- [1] Chiedere all'utente un carattere e calcolare la somma di tutti i valori **`val`** degli elementi che hanno **`c`** pari al carattere inserito.
- [2] Stampare a video la **posizione** ed il carattere **`c`** associato ai valori **`val`** minimo.
- [3] Stampare a video la **posizione** ed il carattere **`c`** associato ai valori **`val`** massimo.
- [4] Per ogni riga della matrice stampare la media dei valori "validi" presenti.

NOTA: [1], [2], [3] e [4] possono essere eseguiti con un unico ciclo

Esercizio 2




```
typedef struct Telemento{
    char c;
    int val;
}Telemento;

typedef struct Tdato{
    Telemento m[NR][NC];
}Tdato;
```

Esercizio 3

Si consideri un programma per gestire gli appartamenti di un condominio.

- Si definisca un tipo struct ***Tpersona*** che contiene il nome, cognome, data di nascita.
- Si definisca un tipo struct ***Tvano*** che contiene una descrizione (array di char) ed i metri quadri.
- Si definisca un tipo struct ***Tappartamento*** che contiene una o più persone (al massimo MAX_PERSONE di tipo*Tpersona*), l'insieme dei vani da cui è composto (al massimo MAX_VANI di tipo *Tvano*).
- Si definisca un tipo struct ***Tcondominio*** che contiene l'indirizzo (rappresentato per semplicità da una stringa), il nome dell'amministratore (*Tpersona*), gli appartamenti (*Tappartamento*) che lo compongono organizzati secondo piani e numero di interno relativo al piano (matrice). Si assuma un numero massimo di piani MAX_PIANI ed un numero massimo di interni/appartamenti per piano MAX_APP. (Stesso numero di interni per ogni piano)

Se necessario, definire ulteriori e opportuni tipi ausiliari.

Nel main dichiarare una variabile **cond** di tipo ***Tcondominio***.

Definire le strutture dati e le variabili necessarie per rappresentare le informazioni definite sopra.

Inizializzare i dati in modo casuale e stamparne il contenuto a video

Esercizio 3

- Costanti (attribuire il valore che si preferisce ma maggiore di 3)
  - MAX_PERSONE, MAX_VANI, MAX_PIANI, MAX_APP
- ***Tpersona***
  - nome → stringa
  - cognome → stringa
  - data di nascita → giorno, mese, anno (3 interi o struct Tdata opportuna)
- ***Tvano***
  - descrizione → stringa
  - metri quadri → numero con vigola (2 cifre decimali)
- ***Tappartamento***
  - persone → array di tipo Tpersona, dimensione MAX_PERSONE
  - vani → array di tipo Tvano, dimensione MAX_VANI
  - n_persone → intero → **numero effettivo di persone presenti**
  - n_vani → intero → **numero effettivo di vani presenti**
- ***Tcondominio***
  - indirizzo → stringa
  - amministratore → stringa
  - appartamenti → matrice di tipo Tappartamento di dimensione MAX_PIANI x MAX_APP
  - n_piani → intero → **numero effetivo di piani**
  - n_app → intero → **numero effettivo di appartamenti (lo stesso per ogni piano)**

Esercizio 4

- L'**allenatore (*Tallenatore*)** è una persona caratterizzata da
  i) nome, ii) cognome, iii) numero di coppe vinte (un intero >=0).
- La singola **squadra** (***Tsquadra***) è identificata dal
  i) nome, ii) colore (della casacca), iii) punteggio (corrente intero >=0), iv) l'allenatore.
- Definire con typedef la struttura dati ***Tfantacalcio***: un insieme di NUM_SQUADRE squadre (di tipo *Tsquadra*) con il relativo allenatore (di tipo *Tallenatore*).
- Definire nel main una variabile **fantacalcio** di tipo ***Tfantacalcio***
- Inizializzare i dati con valori casuali
- Stampare il contenuto della variabile **fantacalcio**
- Calcolare (**con un ciclo**) e visualizzare
  - il solo cognome degli allenatori di squadre che hanno più di 30 punti in classifica
  - il numero totale di allenatori che hanno vinto almeno una coppa.

Esercizio 4

- Vincoli e parametri per inizializzazione casuale
- NUM_SQUADRE → 30 → define
- ***Tallenatore***
  - nome → stringa *(3 caratteri casuali)*
  - cognome→ stringa *(4 caratteri casuali)*
  - coppe → intero *(casuale [0, 16])*
- ***Tsquadra***
  - nome → stringa *(3 caratteri casuali)*
  - colore → stringa *(2 caratteri casuali)*
  - punteggio → intero *(casuale [0, 50])*
  - allenatore → *Tallenatore*
- ***Tfantacalcio***
  - Array di tipo *Tsquadra* di dimensione NUM_SQUADRE
  - **NOTA: Tutti gli elementi dell'array sono utilizzati**
    - **Non** serve variabile (tipo int) per considerare numero effettivo di elementi in array

Esercizio 4

```
...

typedef Tsquadra Tfantacalcio [NUM_SQUADRE];

// In funzione main
Tfantacalcio fantacalcio;

// esempio accesso ai campi
fantacalcio[0].punteggio = 0;
```

```
...

typedef struct Tfantacalcio{
    Tsquadra squadre[NUM_SQUADRE];
} Tfantacalcio;

// In funzione main
Tfantacalcio fantacalcio;

// esempio accesso ai campi
fantacalcio.squadre[0].punteggio = 0;
```

Esercizio 5

Si consideri un programma per gestire le partite del campionato di calcio.

- Si definisca un tipo struct ***Tgiocatore*** che contiene il nome, cognome e numero del giocatore.
- Si definisca un tipo struct ***Tsquadra*** che contiene il nome della squadra e l'insieme dei nomi dei giocatori (N_GIOCATORI di tipo *Tgiocatore*).
- Si definisca un tipo struct ***Tpartita*** che contiene le squadre che hanno disputato una partita (di tipo *Tsquadra*), la data in cui è stata disputata, il numero della giornata, il girone (andata o ritorno), e il risultato.
- Si definisca un tipo struct ***Tcampionato*** che contiene le partite (N_PARTITE di tipo *Tpartita*), la serie a cui queste partite sono riferite (A, B, ..).

Se necessario, definire ulteriori e opportuni tipi ausiliari.

Il programma deve prevedere la gestione di due campionati.

Definire le strutture dati e le variabili necessarie per rappresentare le informazioni definite sopra.

Inizializzare i dati in modo casuale e stamparne il contenuto a video.

Esercizio 5

- ***Tgiocatore***
  - nome → stringa
  - cognome → stringa
  - numero del giocatore → intero
- ***Tsquadra***
  - nome della squadra → stringa
  - elenco dei nomi dei giocatori → array di tipo *Tgiocatore* di dimensione N_GIOCATORI
- ***Tpartita***
  - le squadre → 2 squadre: squadraA e squadraB (di tipo *Tsquadra*)
  - data → giorno, mese, anno (3 interi oppure struct opportuna)
  - numero della giornata → intro
  - girone → 0 oppure 1 (andata o ritorno)
  - reti squadraA → intero
  - reti squadraB → intero
- ***Tcampionato***
  - partite → array di tipo *Tpartita* di dimensione N_PARTITE
  - serie → carattere
- Variabili da definire nel main:
  - **campionato1**, **campionato2** di tipo *Tcampionato*

Esercizio 5

- **Tgiocatore**
  - nome → stringa
  - cognome → stringa
  - numero del giocatore → intero
- **Tsquadra**
  - nome della squadra → stringa
  - elenco dei nomi dei giocatori → N_GIOCATORI
- **Tpartita**
  - le squadre → 2 squadre: squadraA e squadraB
  - data → giorno, mese, anno (3 interi oppure struct opportuna)
  - numero della giornata → intro
  - girone → 0 oppure 1 (andata o ritorno)
  - reti squadraA → intero
  - reti squadraB → intero
- **Tcampionato**
  - partite → array di tipo Tpartita di dimensione N_PARTITE
  - serie → carattere
- Variabili
  - Tcampionato campionato1, campionato2;

Tsquadra squadraA;
Tsquadra squadraB;

Soluzione non ottimale!

Per ogni partita dovrei definire/copire i dati delle squadre: molte informazioni duplicate e spreco di spazio

Esercizio 5b

- ***Tgiocatore***
  - nome → stringa
  - cognome → stringa
  - numero del giocatore → intero
- ***Tsquadra***
  - nome della squadra → stringa
  - elenco dei nomi dei giocatori → array di tipo *Tgiocatore* di dimensione N_GIOCATORI
- ***Tpartita***
  - **squadre Array di 2 elementi di tipo *Tsquadra* → 2 indici che identificano la squadra in vettore squadre**
  - data → giorno, mese, anno (3 interi oppure struct opportuna)
  - numero della giornata → intro
  - girone → 0 oppure 1 (andata o ritorno)
  - reti squadraA → intero
  - reti squadraB → intero
- ***Tcampionato***
  - partite → array di tipo *Tpartita* di dimensione N_PARTITE
  - serie → carattere
- Variabili da definire nel main:
  - *Tsquadra* **squadre [**N_SQUADRE**];**
  - **campionato1**, **campionato2** di tipo *Tcampionato*