INFORMATICA A

A.A. 2019-20

Laboratorio n°4

Dott. Michele Zanella

Ing. Gian Enrico Conti

HEAP LAB

Funzioni

- Hanno una "firma" (*signature*):
  - Tipo dell'output
  - Nome univoco
  - Parametri di input

```
int my_function(int a)
```

- Generalmente terminano con la restituzione, al *chiamante*, del risultato della computazione, attraverso l'istruzione **`return`**.
  - Esistono anche funzioni che non restituiscono nulla: quelle di tipo **`void`**

```
int somma(int a, int b){
    int s = a + b;
    return s;
}
```

- Vengono utilizzate per riutilizzare parti di codice e per migliorarne la leggibilità.
- **In C non è possibile utilizzare una funzione prima che sia dichiarata!**

Funzioni e puntatori

- Posso passare i parametri ad una funzione in due modi:
  - Per *copia*: copio il valore del parametro passato nella variabile locale (parametro)

```
int somma(int a, int b){
    int s = a + b;
    return s;
}
```

  - Per *referenza*: passo il puntatore (indirizzo) del parametro passato. **Se modifico il puntatore deferenziato modifico anche il valore del parametro all'esterno della funzione!!**

```
void somma(int a, int b, int* ris){
    *ris = a + b; //modifico direttamente il valore di ris
}
```

Header files

- Vengono utilizzati per spostare le definizioni delle variabili globali e dei tipi in un file esterno:
  - Possibilità di riutilizzare strutture dati in più file (es., librerie)
  - Migliorano la leggibilità del codice (meno righe nei sorgenti .c)
- Vengono salvati con un formato specifico: `my_header.h`
- Vengono inclusi nei file .c tramite la `#include "my_header.h"`

- Per il progetto creiamo un file `define_and_typedefs.h`

- Negli header files posso anche inserire dei **prototipi** delle funzioni, ossia una loro dichiarazione fatta senza specificare il corpo della funzione stessa.

File I/O

- I file rappresentano una sequenza di byte, siano essi *file di testo* o *file binari*.
- La fine del file è espressa dal carattere speciale **EOF**.
- Vengono rappresentati tramite una struttura **FILE**
- Nella manipolazione dei file si utilizza il puntatore a tale struttura: `FILE* fp;`
- La libreria <stdlib.h> mette a disposizione una serie di funzioni per operare sui file:
  - `FILE* fopen(const char* filename, const char* mode)`: crea o apre un file, specificando il nome del file e la modalità di apertura.
    - **r:** lettura
    - **w:** scrittura
    - **a:** writing appending mode
  - `int fputc(int c, FILE* fp)`: scrive il caratter *c* nel file puntato da *fp* tramite un *output stream*
  - `int fputs(const char* s, FILE *fp)`: scrive una stringa *s* nel file puntato da *fp*
  - Altre simili a quelle di console: `fwrite`, `fprintf`...

File I/O

- Altre funzioni
  - `int fgetc(FILE *fp)`: leggere un carattere dal file puntato da *fp*
  - `char* fgets(char* buf, int n, FILE *fp)`: legge fino a *n-1* caratteri dal file puntato da *fp*. La funzione si ferma se incontra un '\n' o la fine del file.
  - Alte simili a quelle di console: `fread`, `fscanf`…
  - `int fclose(FILE *fp)`: Chiude un file ed esegue un flush sui buffer di scrittura/lettura.

- La sequenza per operare sui file è:
  1. fopen(…)
  2. freaed/fgets oppure fwrite/fputs
  3. fclose
- In caso di problemi in apertura il puntatore *fp* = NULL

Approccio per la soluzione dei problemi

1) Analisi dei requisiti, definizione del modello e dei tipi necessari
2) Stesura del Flow chart (o degli step) ad alto livello della soluzione
3) Definizione delle funzioni necessarie e organizzazione del codice
4) Implementazione e compilazione passo-passo (non aspetto alla fine di tutto!)
5) Testing

Esercizio 4.1: Somma vettori con puntatori

Scrivere una funzione che calcoli la somma di due vettori passati come puntatori e salvi il risultato in una variabile passata come puntatore.

```
void sommavp(int* v1, int* v2, int* ris);
```

Hint:

Considerate la dimensione dei vettori come prefissata

Esercizio 4.2: Stringa Inversa

Si implementi una funzione `reverse` con prototipo:

```
void reverse(char* inversa,
             char* sorgente);
```

che riempia la stringa puntata da "inversa" con i caratteri di "sorgente" nell'ordine opposto.

- Esempio: HELLO -> OLLEH

Esercizio 4.3: File.txt

Si implementi un programma che scriva più linee inserite dall'utente nel file *test.txt* e ne successivamente ne legga il contenuto.

- Attenzione ad aprire e chiudere il file nelle due modalità!

Esercizio 4.4: Mastermind

Scrivere un programma per giocare a **Mastermind**.

http://www.webgamesonline.com/mastermind/

http://www.webgamesonline.com/mastermind/rules.php

Regole di base:

- 2 giocatori (CPU e utente)
- CPU sceglie una sequenza di colori non nota all'utente
- L'utente deve indovinare l'esatta sequenza (posizione e colore)
- Ad ogni round l'utente inserisce una possibile sequenza
- CPU risponde in questo modo:
    - Per ogni posizione e colore indovinato nella sequenza, mostra un colore nero a destra (=black peg)
    - Per ogni colore indovinato (ma posizione sbagliata) nella sequenza, mostra una colore bianco (=white peg)

Esercizio 4.4: Mastermind (cont'd)

Altre regole:

- L'utente definisce la lunghezza della sequenza da indovinare
- Nella sequenza i colori si possono ripetere
- L'utente definisce il numero massimo di round in cui indovinare la sequenza
- L'utente vince se indovina l'esatta sequenza entro il numero massimo di round, altrimenti vince CPU.

Esercizio 4.4: Mastermind (cont'd)

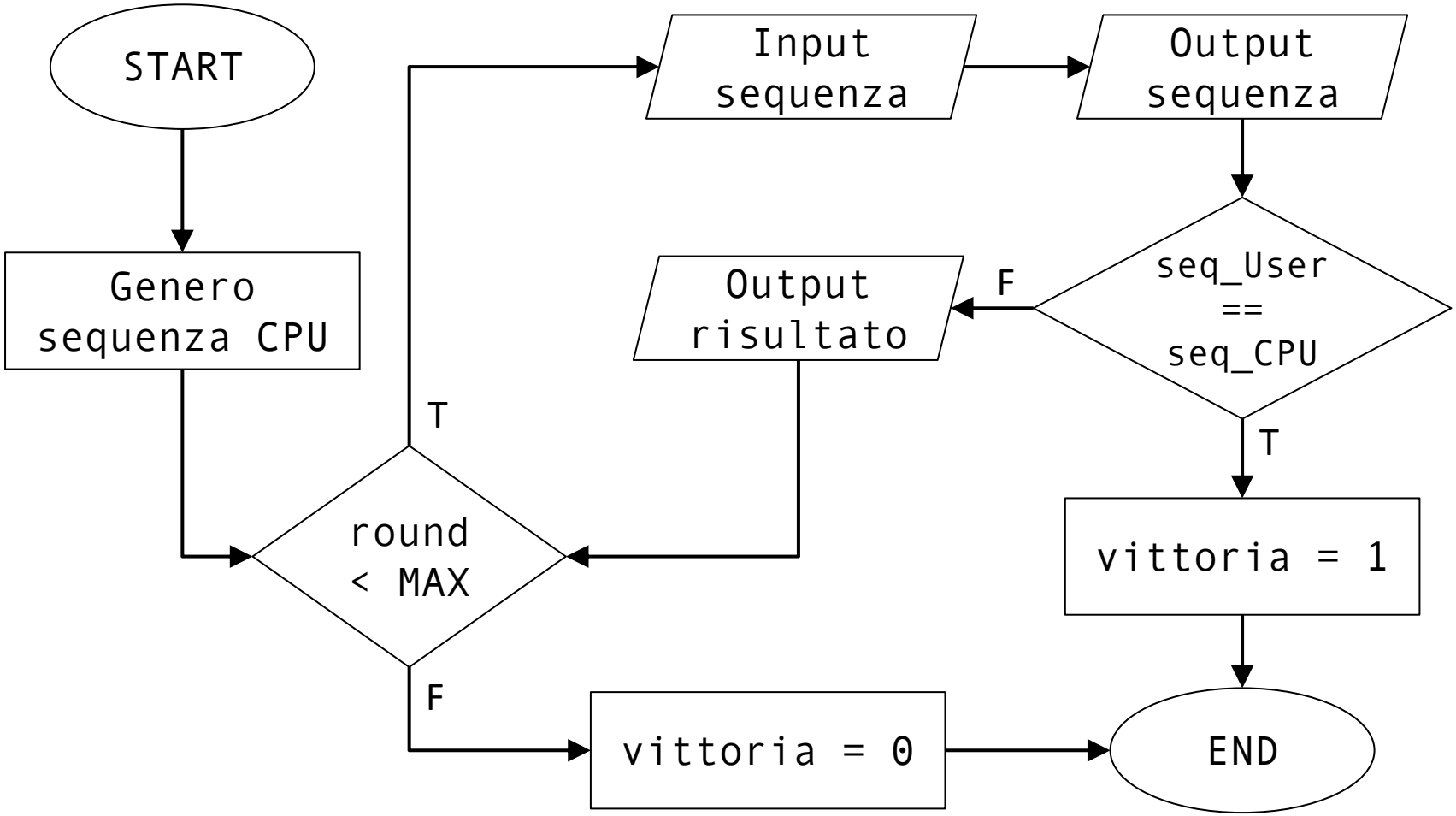
Requisiti:

- L'utente deve definire le impostazioni di gioco (lunghezza sequenza, numero massimo di round)
- CPU deve generare una sequenza casuale di colori
- L'utente deve poter inserire una possibile sequenza ad ogni round
- CPU deve mostrare la qualità del risultato del round (colori bianchi e/o neri)

Hints:

- Utilizzare solo testo: i colori vengono scritti come caratteri testuali (e.g., w = "pallino bianco")
- Definire un carattere per ogni colore per l'inserimento (e.g., 'y' = "giallo")
- 6 colori: rosso, giallo, verde, arancione, nero e bianco

Esercizio 4.4: Mastermind (Flowchart generale)



Esercizio 4.4: Mastermind (cont'd)

Altri hints:

- Utilizzare la funzione `rand()` della librearia `stdlib.h` per generare il codice CPU casuale.
- Definire ed implementare le seguenti funzioni (oltre a quelle per stampare risultato e codice):

```
void genera_codice(char* secret_code);
```

```
void indovina_codice(char* guess_code);
```

```
int controlla_codice(char* secret_code,
                     char* guess_code,
                     int* white_pegs,
                     int* black_pegs);
```

```
void stampa_codice(char* code);
```

Esercizio a casa 4.1: Gara di danza

Scrivere un programma che gestisca i punteggi di una gara di danza. calcoli il punteggio dell'esecuzione per ogni atleta di una gara di danza.

Requisiti:

- Ogni atleta è caratterizzato da un codice alfanumerico e dal suo punteggio finale
- Possibilità di inserire il codice degli atleti da input
- Possibilità di inserire i punteggi dei singoli giudici da input e calcolare il risultato finale
- Possibilità di stampare tutti gli atleti con i relativi punteggi finali

Hints:

- Utilizzare le funzioni della `stdio`
- Definire numero di giudici e atleti
- Definire i tipi e le funzioni necessarie

Esercizio a casa 4.2: Data.csv

Si implementi un programma che legga dal file *data.csv* le temperature massime e minime di alcune città italiane.
Poi trovi la più calda e la più fredda (considerando rispettivamente la massima e la minima).
Infine chieda all'utente di aggiungere gli stessi dati per la città nella quale si trova attualmente.

- Dichiarare le struct, typedef e define necessarie
- Prototipare le funzioni necessarie

Esercizio a casa 4.3: Salvataggio partite Mastermind con File I/O

Aggiungere al gioco Mastermind la possibilità di salvare la partita e riprendere il gioco.

Requisiti:

- Il gioco deve salvare in un file di testo (`.txt`) ogni round di una partita
- L'utente sceglie se riprendere l'esecuzione di una partita o se iniziarne una nuova
- Il gioco deve poter riprendere l'esecuzione di una partita dall'ultimo round giocato salvato su un file di testo (`.txt`)

Hints:

- Se la partita termina (vittoria o sconfitta) svuotare il file di testo
- Aprire e CHIUDERE il file nelle modalità corrette!
- Se si inizia una partita nuova, sovrascrivere il file precedente
- Salvo solo l'ultimo round giocato (ulteriore esercizio: salvare tutta la partita fino all'ultimo round giocato)