Figure

Mostrare a video le seguenti figure (sfruttare loop e condizioni):

1.1

```
x   x   x   x   x

x               x

x               x

x               x

x   x   x   x   x
```

1.2

```
x       x       x

    x   x   x

x   x   x   x   x

    x   x   x

x       x       x
```

1.3

```
o   H   H   H   o

-   o   H   o   H

-   -   o   H   H

-   o   -   o   H

o   -   -   -   o
```

Cicli

2

Scrivere un programma che, dato un numero in ingresso, ne stampi tutti i suoi divisori.

3

Scrivere un programma che, presi in ingresso due numeri A e B diversi entrambi pari o entrambi dispari, verifichi che il valore assoluto della differenza tra la somma dei numeri pari tra A e B esclusi e la somma dei numeri dispari, sia uguale al valore medio di A e B.

Esempio, dati A=3 e B=9, facciamo la somma dei valori dispari tra A e B: 5+7=12 e la somma dei valori pari tra A e B: 4+6+8=18, la differenza è 6 che è la media dei due valori.

Funzioni

4.a

Scrivere una funzione per la somma, una per la sottrazione, una per la moltiplicazione e una per la divisione. Scrivere poi una funzione `calcolatrice` che, passati come parametri un carattere e due numeri, decida in base al carattere che funzione chiamare e ne ritorni il risultato:

- 'a': addizione
- 's': sottrazione
- 'm': moltiplicazione
- 'd': divisione

4.b

Modificare la funzione `calcolatrice` così che possa prendere un argomento in più e che salvi il valore in tale parametro piuttosto di ritornarlo a fine esecuzione.

Provare questo esercizio sia con i riferimenti che con i puntatori.

5

La libreria `cstdlib` contiene due funzioni per generare numeri randomici. La prima funzione è `void srand(unsigned int)` che prende in ingresso un "seed", ossia un valore di inizializzazione e deve essere chiamata una sola volta all'inizio del main. Per il nostro scopo possiamo scegliere un qualsiasi valore di inizializzazione, diciamo 13. La seconda funzione è `int rand()` che genera un numero casuale. Per avere un numero casuale limitato in un certo range $[A, B)$, possiamo scrivere `rand() % (B - A + 1) + A`.

```
#include<cstdlib>

// Vostra funzione che conterrà rand()

int main(){
  srand(13);
```

```
  return 0;
}
```

Scrivere una funzione, che presi come argomenti due valori A e B, ritorni numeri casuali pari nell'intervallo e stampare il valore di ritorno nel main.

Riavviare più volte il programma. Cosa notate? Cosa succede se si cambia il valore di inizializzazione di `srand()`?

6.a

Scrivere un programmino che simuli una partita a sasso carta forbice su tre turni tra due giocatori. A ogni turno estrarre in maniera casuale un numero tra 1 e 3 compresi:

- 1 -> sasso
- 2 -> carta
- 3 -> forbice

Usare un enum per dichiare `sasso`, `carta`, `forbice`.

Usare la funzione precedentemente scritta per farsi tornare il valore tra 1 e 3.

Fare poi i controlli per valutare quale dei due giocatori ha vinto.

Aggiungere un ciclo while che chieda se si vuole continuare a giocare.

6.b

Modificare l'esercizio precedente in modo che il progamma chieda all'inizio se si vuole simulare la partita tra due computer, tra un giocatore e un computer, o se tra due giocatori. Nel caso uno dei due sia un giocatore, prendere in input un valore tra 1 e 3 compresi, e fare i dovuti controlli sul valore in ingresso.

Puntatori e riferimenti

7

Dati in input prezzo e sconto, scrivere una funzione che calcoli il costo scontato. Scrivere la funzione con passaggio per valore, per indirizzo e per riferimento (che differenze ci sono?).

8

Scrivere una funzione `void mySwap (int*, int*)` che esegua lo scambio dei valori di due variabili tipo `int`.

9

Scrivere una funzione `void genericSwap (void*, void*, bool)` che esegua un controllo sul parametro booleano e decida se eseguire uno scambio tra puntatori di tipo `int` o di tipo `char`.

Ricorsione e iterazione

10

Scrivere una funzione ricorsiva che calcoli la somma tra due numeri interi **positivi**.

11

Scrivere una funzione ricorsiva che calcoli il fattoriale di un qualsiasi numero intero. Riscrivere poi la sua versione iterativa (quale delle due è più efficiente?).

Bonus: Usare la libreria std::chrono::steady_clock per misurare quanto ogni funzione impiega. Per ottenere un tempo attuale è possibile usare:

```
chrono::time_point<std::chrono::steady_clock> start = chrono::steady_clock::now();
```

Prendendo il tempo prima e dopo la funzione che si vuole misurare (tempo1 e tempo2) è possibile sottrarli (tempo2 - tempo1) e ottenere la durata