Raccolta di Esercizi di

Programmazione

# Algoritmi e strutture di controllo

1. Data l’area del cerchio, stampare la circonferenza
2. Data la base e l’altezza di un triangolo, scrivere l’area
3. Somma di due numeri (v1 - usando una terza variabile)
4. Dati due numeri calcolare il loro quoziente se il divisore è != 0, ritornare “impossibile” se il divisore = 0
5. Dato in input un numero n, stampa la prima potenza di 2 maggiore o uguale a n
6. Moltiplicazione di due numeri senza usare l’operatore \*
7. Divisione di due numeri interi senza usare l’operatore di divisione
8. dati dall'utente due numeri interi n1 e n2 il programma deve stampare:
   1. tutti i numeri dal più piccolo dei due al più grande dei due
   2. tutti i numeri dal più grande dei due al più piccolo dei due
   3. tutti i numeri da n1 a n2 (comunque siano n1 e n2 quindi potresti dover andare in ordine crescente o decrescente)
   4. tutti i numeri da n2 a n1 (comunque siano n1 e n2 quindi potresti dover andare in ordine crescente o decrescente)
9. Aggiungi all’esercizio sulla divisione tra due numeri senza usare l’operazione diviso anche la stampa del resto
10. Inserire n numeri != 0 (0 per finire), contare quanti sono i numeri inseriti
    1. Variante avanzata: i numeri sono voti che devono essere validi e alla fine ne va calcolata la media.
11. Moltiplicazione di due numeri senza usare l’operatore di moltiplicazione
12. Dati tre numeri reali dire che tipo di triangolo essi formano (classificazione dei triangoli in base ai lati).
13. Data una sequenza di n numeri interi (valore di n dato dall’utente), calcolare la somma dei pari ed il prodotto dei dispari
14. Con un opportuno ciclo chiedere all’utente di inserire due numeri n1 e n2 compresi tra 1 e 100. Se i numeri non fossero corretti, rimanere nel ciclo e ripetere la richiesta. Stampare quanti sono i numeri pari compresi tra i due numeri
15. Data una sequenza di numeri terminata dal numero 0 (leggo numeri finchè non mi viene dato il numero 0), stampo il maggiore e il minore
16. Letto un numero intero positivo n stampare il fattoriale: n! = 1 · 2 · 3 · . . . ·n
17. Dati in input 3 numeri, stamparli in ordine crescente
18. Realizzare un programma che, presi in input 2 operandi reali e un operatore (+, -, \*, /), esegue l’operazione stampandone il risultato
19. Progettare un algoritmo che legga da terminale una sequenza di interi positivi e negativi terminati dal valore 0 (uno su ogni linea) e stampi il prodotto degli interi positivi e la somma dei negativi.
    1. Progettare un algoritmo che legga da terminale una sequenza di interi positivi e negativi fintanto che l’utente dice di volerne inserire ancora, e stampi il prodotto degli interi positivi e la somma dei negativi.
20. Esercizio di pag 377 n 14
21. Dati due numeri, determinare il maggiore (verificare anche se sono uguali)
22. Data una sequenza di numeri terminata da 0, dire quanti sono i numeri inseriti (diversi da 0)
23. Dati due numeri in input b ed e, calcola e scrivi be
24. Dati in ingresso 4 numeri, che rappresentano gli orari in cui avvengono due diversi eventi della giornata, in modo che i primi 2 numeri siano ore e minuti del primo orario e gli altri 2 numeri siano ore e minuti del secondo orario, stabilire quale evento è avvenuto prima. Inserisci dei controlli durante l’input in modo che le ore possano andare da 0 a 23 e i minuti da 0 a 59.
25. Dato un numero di minuti e un numero di secondi rappresentare il conto alla rovescia.
26. Dato in input un numero intero n, stampa l’ennesimo numero della successione di fibonacci. La successione di fibonacci è quella successione di numeri in cui ogni numero è la somma dei due numeri precedenti. I primi due numeri sono 1 1. La successione inizia così: 1 1 2 3 5 8 13 21 …
27. Dato in input un numero intero, conta da quante cifre è composto.
28. Dato in input un numero intero n di 3 cifre (in questo caso sarebbe utile mettere un controllo sull’input, cioè continuare a richiedere il numero fintanto che il numero dato non è di 3 cifre), stampa separatamente le sue cifre. Consiglio: usa divisioni per 10 e resti della divisione per 10. Puoi provare anche a generalizzare il programma in modo che funzioni anche con numeri di dimensione diversa da 3 cifre.
    1. Generalizza l’esercizio in modo che possa funzionare con un numero qualsiasi.
29. Dato in input un numero intero n, stabilisci se è primo
30. Dato in input un numero intero n, stampa la sua scomposizione in fattori primi
31. Data una sequenza di n numeri (n dato dall’utente) stabilire qual è il numero più grande, qual è il più piccolo e calcolare la media.
32. Data una sequenza di n numeri (n dato dall’utente) stabilire qual è il secondo numero più grande, qual è il secondo più piccolo.
33. Data una sequenza di numeri positivi (fai il controllo sull’input) terminati da 0, scrivi qual è la maggior differenza tra due numeri dati consecutivamente
34. Data una sequenza di numeri positivi (fai il controllo sull’input) terminati da 0, scrivi la somma dei numeri divisibili per 3
35. Dato un numero n scrivi somma e prodotto di tutti i numeri minori o uguali a n.
36. Dato un numero n stabilire quante volte è possibile dividerlo per 2. (esempio 20 è divisibile per 2, 2 volte)
37. Dato un numero n, decidere se è primo
38. Data una sequenza di prezzi di prodotti, calcolare la spesa totale sapendo che se un prodotto costa meno di 100€ lo devo scontare del 10% altrimenti del 5%. Decidi tu il metodo per capire quando terminare la lettura della sequenza di prezzi.
39. Data una sequenza di 5 numeri che rappresentano i voti presi nelle diverse materie, stabilire se lo studente sarà promosso, bocciato o rimandato a settembre. Lo studente è promosso se non ha insufficienze, è bocciato se ha almeno 3 insufficienze, altrimenti è rimandato. Ricordati di controllare i valori dei voti in input che devono essere voti validi.
40. Leggi 3 parole e stampale tutte insieme in un'unica volta (solo un parallelogramma di scrittura).
41. Dato un numero n positivo stampa tutti i numeri da 1 a n, i primi n multipli di 2 (2 compreso che consideri il primo multiplo) e i primi n multipli di 3
42. Dati tre numeri positivi verificare che questi tre numeri siano una terna pitagorica ( una terna pitagorica è un insieme di 3 numeri per cui la somma del quadrato di due numeri sia uguale al quadrato del terzo numero, in altre parole sono le lunghezze dei lati di un triangolo rettangolo)
43. Scrivere l’algoritmo per il pagamento della spesa che consiste nel chiedere inizialmente se si è in possesso della carta fedeltà, poi chiedere tutti i prezzi dei prodotti acquistati terminando la sequenza con un prezzo uguale a zero. Il programma deve sommare i prezzi, e se si è in possesso della carta, scontare del 10% i prezzi minori di 50 e del 5% i prezzi maggiori di 50
44. Un libro deve essere restituito in biblioteca dopo 15 giorni di prestito altrimenti si è multati di 0,80€ al giorno di ritardo. Ricevuto in ingresso il numero di giorni di un prestito, visualizza se il socio deve essere multato per il ritardo e a quanto ammonta la multa da pagare.
45. Calcolare il costo della bolletta telefonica sapendo che i primi 30 scatti costano 20 centesimi l’uno, gli scatti dal 31 al 100 costano 15 centesimi l’uno, mentre gli ulteriori scatti costano 10 centesimi l’uno. Aggiungere infine una tassa fissa di 2,50€ per le spese telefoniche. In input al programma è dato il numero di scatti effettuati.
46. Costruire uno schema di flusso che rappresenti l’algoritmo per il seguente gioco: il computer genera un numero segreto (usa la funzione random(100) che genera un numero casuale tra 0 e 99) e il giocatore deve individuarlo seguendo le indicazioni fornite dal computer (ti dice se il numero da trovare è più grande o più piccolo di quello che hai provato); una volta trovato il numero segreto, il numero di tentativi effettuati è visualizzato a video.
47. Calcolare il quoziente e il resto della divisione intera di due numeri interi positivi forniti in ingresso chiamati dividendo e divisore, applicando il metodo delle sottrazioni successive. Per esempio, se dividendo=13 e divisore=5, il programma dovrà restituire Quoziente=2 e Resto=3, calcolati sottraendo successivamente il valore di divisore dal valore di dividendo
48. Visualizza i termini della successione di Fibonacci compresi nell’intervallo tra due interi positivi N1 e N2, entrambi forniti in ingresso, con N2>N1 (controlla bene gli input)
49. Simula il lancio di un dado per N volte (con N intero e positivo in ingresso e usando la funzione random(7) per il lancio) e verifica che effettivamente la probabilità che esca 6 è 1/6. Per farlo fai tanti tiri e vedi se esce 6 un sesto delle volte (più tiri fai più dovresti avere un risultato positivo).
50. Simula il lancio di due dadi e verifica le probabilità che escano: una coppia di 6, un valore totale uguale a 7 (due diverse probabilità)
51. Dati in ingresso due numeri positivi x e y, visualizza in ordine decrescente la sequenza di numeri interi compresi tra x e y che sono divisibili per il minore tra x e y. Ad esempio, se x = 7 e y = 35, la sequenza è 35 28 21 14 7.
52. Simula una serie di lanci di un dado a 6 facce (d6) e un dado da 20 facce (d20). Calcola il valore medio che si ottiene lanciando il d6 e quello ottenuto lanciando il d20.
53. Valuta se è più probabile ottenere 7 oppure ottenere 8 lanciando 2 dadi da 6 e sommando i 2 valori ottenuti.
54. Voglio valutare la probabilità che tirando due dadi da 6 esca il valore 2. Il valore calcolato deve essere abbastanza preciso e decido di calcolarlo misurando la media di tanti tiri di dado e fermandomi solo quando vedo che facendo uscire un nuovo 2 valore calcolato non varia di più di 0,01%
55. Vengono dati in input i valori delle altezze di n persone (n chiesto all'utente). Per ogni persona oltre all'altezza viene indicato anche il sesso. Deve essere stampato il valore medio di altezza separatamente per i maschi e per le femmine. L’utente può decidere di non indicare né maschio né femmina, in quel caso non contare quella persona.
56. Scrivi il codice di un gioco per 2 persone basato sul lancio di dadi. All’inizio del gioco si decide che tipo di dado usare, si possono scegliere i dadi da 4, 6, 8 o 20 facce. Il gioco consiste nel tirare uno per volta il dado e sommando ogni volta i numeri usciti finchè uno dei due giocatori non supera un valore limite che è 4 volte il valore massimo rappresentato sul dado scelto (ad esempio se si sceglie il dado da 6 il limite è 24). Per rendere il gioco equo si può fare in modo che i due giocatori debbano tirare per forza lo stesso numero di tiri e che entrambi i giocatori possano superare il limite facendo finire il gioco in parità
57. Scrivi una variante del gioco precedente in cui l’obiettivo è avvicinarsi il più possibile al valore limite senza superarlo. Chi si avvicina di più vince. In ogni momento un giocatore può decidere di accontentarsi del valore ottenuto e smettere di tirare.
58. Leggi un numero positivo e controlla se esso è divisibile per 2 o divide 3 e non è compreso tra 10 e 20
59. Leggi un numero positivo e controlla se esso è divisibile per 2 o è un multiplo di 3 ed è compreso tra 10 e 20
60. dato un numero n stampa tutti i numeri da 1 a n che sono divisibili da 2 o 3 o 5 ma non sono divisibili per 4
61. Dato un numero positivo n disegna le figure geometriche riportate di seguito:

per n = 5

|  |  |  |
| --- | --- | --- |
| xxxxx  xxxx  xxx  xx  x | xxxxx  xxxxx  xxxxx  xxxxx  xxxxx | x  xx  xxx  xxxx  xxxxx |

1. Esegui un programma che abbia il seguente output (chiaramente usando dei cicli):

1 2 3 4 5 6

2 3 4 5 6 +

3 4 5 6 + +

4 5 6 + + +

5 6 + + + +

6 + + + + +

1. Simula l’esecuzione di un gioco di dadi in cui due persone si scontrano lanciando ognuno un dado. Il gioco è diviso in round. Al primo round i due giocatori lanciano ognuno un dado da venti facce e vince chi fa il numero più alto, in caso di pareggio ritirano i dadi finchè uno dei due non vince. Chi perde il round deve affrontare i round successivi con un dado con una faccia in meno. Perde la partita chi arriva ad avere un dado con 0 facce. Mostra ad ogni round quante facce hanno i dadi dei due giocatori e mostra il risultato di ogni lancio dei dadi. Dichiara infine il vincitore.
2. Scrivi un programma che dato un numero n stampi una piramide come nel seguente esempio:

Con n = 3

1

121

12321

1. Scrivere un programma che legga due interi n e m con valori compresi tra 1 e 9, i cui prodotto sia inferiore a 35, e stampi m piramidi di altezza n. L’esempio si riferisce al caso n=3 e m=4.

1 1 1 1

121 121 121 121

12321123211232112321

1. Scrivi un programma che permetta di controllare i dati di input immessi dall’utente:
   1. se l’utente inserisce un intero N compreso tra 1 e 10, il programma deve stampare a video il valore NN,
   2. se l’intero N e’ compreso tra 11 e 20, il programma deve stampare a video la somma 1 + 2 + 3 +… + N
   3. altrimenti deve dare un segnale di errore.
2. Si realizzi un programma che legga un intero N da tastiera, e stampi a video il risultato della seguente sommatoria:



Una volta calcolato e stampato il valore a video, il programma deve chiedere un nuovo numero all’utente e ripetere il calcolo. Il programma deve terminare solo qualora l’utente inserisca un valore negativo.

1. Si progetti in C un programma che legge un float, rappresentante un ammontare di euro; di seguito il programma deve leggere un tasso d’interesse (in percentuale), ed un numero di anni. Il programma deve stampare, in uscita, per ogni anno, come l’ammontare cresce con gli interessi. Si ricordi che l’interesse si calcola con la seguente formula:

Cfin = Cin ( 1 + (r/100))N

dove Cfin è il capitale finale, Cin è quello iniziale, r è l’interesse, e N rappresenta il numero di anni in cui si applicano gli interessi.

Ad esempio supponiamo che il capitale iniziale sia di 1000.0 €, con un tasso del 3%, per un periodo di 3 anni. L’output stampato deve avere all’incirca questo aspetto:

Capitale iniziale: 1000.00€

Dopo 1 anno: 1030.00 €

Dopo 2 anni: 1060.90 €

Dopo 3 anni: 1092.73 €

1. Realizzare un programma che legga da input un carattere dell’alfabeto e stampi a video il carattere stesso ed il suo valore ASCII. Il programma deve controllare che il carattere inserito sia compreso tra ‘a’ e ‘z’ o tra ‘A’ e ‘Z’ (in caso contrario si stampi un messaggio di errore). Dopo la stampa, il programma deve continuare a chiedere nuovi caratteri, finché l’utente non inserisce il carattere corrispondente al numero zero ( ‘0’ ): in tal caso il programma termina.
2. Realizzare un programma che prende in input una sequenza di caratteri ‘0’ e ‘1’ e conta la lunghezza della più lunga sotto-sequenza di ‘0’ di fila L’inserimento della sequenza termina quando si inserisce un carattere diverso da ‘0’ e ‘1’ A quel punto, si stampa a video il valore trovato
3. Realizzare un programma che prenda in input una sequenza di cifre (tra 1 e 9) e calcoli la somma massima fra le sottosequenze di cifre non decrescenti Il programma termina quando viene inserito lo 0

Esempio:



# Array, stringhe e matrici

## Array numerici

1. Leggi n voti, caricali in un array, calcolane la media, poi stampa tutti i numeri e la media.
2. Continua l’esercizio precedente calcolando e stampando la media dei soli voti sufficienti.
3. Carica un array di numeri casuali, chiedi poi all’utente un numero e digli se questo numero è presente nell’array.
4. Carica un array, di dimensione a piacere, di numeri casuali compresi in un range di numeri deciso dall’utente. Esegui poi le seguenti operazioni:
   1. inverti l’ordine dei numeri contenuti nell’array,
   2. Date due posizioni a e b dell’array (chiaramente controlla che a sia minore di b e che siano compresi tra 0 e N-1) stampa prima i numeri dell’array dalla posizione a alla posizione b poi i numeri dalla posizione b alla posizione a
   3. Dato un numero positivo x qualsiasi stampa tutti i numeri dell’array dalla posizione 0 alla posizione x. Se x > N-1, dopo aver stampato tutto l’array ricomincia a stampare dalla posizione 0 e continua fino ad aver stampato x+1 numeri.
5. Scrivi un programma che ordini un array con l’algoritmo selection sort, prima in ordine crescente poi in ordine decrescente.
6. Scrivi un programma che ordini un array con l’algoritmo bubble sort, prima in ordine crescente poi in ordine decrescente.
7. Scrivi un programma che dato un array di dimensione n e due valori x y tali che x<=y<=n, ordini l’array dalla posizione x alla posizione y comprese.
8. Riempi un array di numeri casuali poi rappresenta l’array come un istogramma.

Ad es.

1: x

0:

4: xxxx

6: xxxxxx

3: xxx

9: xxxxxxxxx

1. Riempi un array di numeri casuali tra 0 e 9 e calcola la somma massima fra le sottosequenze di cifre non decrescenti. Il valore deve essere calcolato da un’apposita funzione.

Esempio:



la prima riga rappresenta l'array e la seconda i vari valori calcolati (viene restituito 13 che è il più grande)

1. Scrivi un programma che riempia un array di numeri casuali di una cifra in modo tale che ogni numero sia sempre maggiore o minore di entrambi i numeri adiacenti (ad esempio 1 8 3 5 2 7 4 5 3). L’array generato deve essere stampato poi ordinato e infine ristampato. Per implementare le funzionalità descritte realizza due funzioni: una che riceve un array e lo riempie di numeri casuali come descritto, una che riceve un array e un indicazione che dica se ordinare l’array in ordine crescente o decrescente (ad esempio una variabile booleana).

## Stringhe

1. Leggi una stringa da input e poi comunica all’utente di quante lettere è composta la stringa. Il conteggio deve essere fatto in due modi diversi, usando la funzione strlen e contando i caratteri uno alla volta.
2. Dato un array di numeri interi casuali, calcola e stampa il valore massimo e il valore minimo contenuti nell’array
3. Dato un array di numeri interi casuali, calcola e stampa media e mediana dell’array
4. Dato un array di numeri interi casuali, stampa tutti i numeri superiori alla media partendo dal fondo
5. Leggi una stringa da input e poi stampala in due modi diversi, la prima volta tutta insieme poi stampando una lettera per volta separando le lettere con uno spazio.
6. Leggi una stringa da input e poi stampala al contrario, cioè partendo dal fondo ma non modificandone il contenuto.
7. Leggi una stringa da input e poi modificala in modo da ottenere la stringa inversa (ad esempio “ciao” diventa “oaic”). Stampa infine la stringa.
8. Dichiara e inizializza due stringhe con valori a piacere, stampale. Scambia il contenuto delle due stringhe in modo che la prima contenga il contenuto della seconda e viceversa, poi ristampale.
9. Leggi da input due stringhe, se le due stringhe sono uguali comunica che sono uguali altrimenti scrivi quella che viene prima in ordine alfabetico
10. Leggi usando un’unica funzione scanf tre parole. Indica poi qual è la parola più lunga, la parola che viene prima in ordine alfabetico e la parola che contiene più vocali.
11. Scrivi e utilizza una funzione che riceve una stringa e un carattere e che stampi tutto il contenuto della stringa andando a capo ogni volta che trova il carattere dato (il carattere rappresenta un separatore della stringa in sottostringhe).
12. Scrivi un programma che contenga in un array la seguente stringa “Ciao a tutti, mi chiamo Francesco\nOggi vi voglio parlare delle stringhe, siete d’accordo?” e che la stampi in modo tale che quando trova una virgola va a capo(oltre a stampare la virgola) e quando trova un a capo, va a capo due volte.
13. Scrivi due funzioni che implementino il cifrario di Cesare cioè un cifrario che per cifrare un testo trasforma ogni lettera nella lettera che si trova tre posizioni più avanti nell’alfabeto. Le due funzioni sono le funzioni “cifra” e “decifra”. Scrivi infine u programma che permetta di inserire un testo da cifrare o decifrare a scelta.
14. Scrivi e utilizza una funzione che riceve due stringhe e sia in grado di dire se la stringa più piccola è una sottostringa di quella più grande. Nel tuo programma il main deve creare due stringhe di lunghezza a piacere contenenti lettere casuali (solo lettere maiuscole e minuscole senza altri simboli o cifre; la funzione non cambia)

Variante: invece di dire se la stringa è contenuta nell’altra stringa restituire la posizione di partenza della sottostringa trovata.

Ad esempio “mi c” è sottostringa di “Ciao a tutti mi chiamo Francesco”

1. Scrivi un programma che chieda all'utente nome e cognome e poi:

* Verifichi che il nome e il cognome siano stati scritti con le iniziali maiuscole (devi controllare se la prima lettera ha i valori corretti secondo la tabella ASCII, i caratteri sono come numeri)
* crei una stringa che contenga le generalità dell'utente (nome e cognome insieme)
* stampi le generalità al contrario.
* chieda nuovamente nome e cognome e verifichi se sono state riscritte le stesse cose di prima

## Matrici di numeri

1. Creare una matrice che contiene le tabelline dei numeri dall' 1 al 10. Alla fine stamparla.
2. Scrivi un programma che istanzia una matrice n x m di interi casuali (n e m scelti da te). Oltre a stampare l'intera matrice, chiedi all'utente quale riga o quale colonna stampare e stampala sullo schermo
3. Scrivere un programma che riempia due matrici A e B di dimensione n x m e calcoli la matrice C = A + B e la matrice D = A x B (in realtà è un finto prodotto di matrici). Stampa le matrici ottenute. Per il calcolo della somma calcola normalmente C[i][j] = A[i][j] + B[i][j] mentre per la moltiplicazione hai due possibilità:

* Versione semplice se non riesci proprio a fare la versione corretta: C[i][j] = A[i][j] x B[i][j].
* Versione complicata: vai a guardare su internet come si calcola e scrivi l’algoritmo per farlo.

1. Scrivi un programma che data una matrice n x m di interi, scambia le righe pari con quelle dispari  
   variante: le dimensioni della matrice vengono scelte dall'utente tra i valori massimi 10 x 20.
2. Scrivi un programma che data una matrice n x n di interi, con n scelto dall'utente, scambia le righe con le colonne
3. Scrivi un programma che riempia una matrice n x m di numeri interi a caso, n e m sono scelti dall’utente e il range dei numeri è da 0 a x scelto anch’esso dall’utente. Determina poi il numero più frequente generato nella matrice. Il riempimento delle matrici e la ricerca del numero più frequente devono essere svolti da due apposite funzioni, mentre l’interazione con l’utente dev’essere lasciata al main.
4. Scrivi un programma che riempia una matrice n x m di numeri interi a caso, n e m sono scelti dall’utente e il range dei numeri è da 0 a x scelto anch’esso dall’utente. Determina poi il la riga o la colonna con la somma dei numeri più grande. Il riempimento delle matrici e la ricerca della riga o della colonna devono essere svolti da apposite funzioni, mentre l’interazione con l’utente dev’essere lasciata al main.
5. Scrivi un programma che:

* crei e riempia di valori casuali una matrice di dimensioni a tua scelta,
* ne stampi il contenuto sullo schermo,
* chieda all'utente un numero e cerchi se il numero è presente nella matrice
* se il numero è presente si stampino le coordinate della posizione nella matrice

1. Scrivi un programma che istanzia una matrice n x m di interi casuali (n e m scelti da te). Oltre a stampare l'intera matrice, chiedi all'utente quale riga o quale colonna stampare e stampala sullo schermo
2. data una matrice n x m di numeri casuali (n e m scelti da te), stampa l'indice della riga con la massima somma dei numeri contenuti. Stampa anche l'indice della colonna con la massima somma dei numeri.
3. Scrivi un programma che per mezzo di un’apposita funzione riempia una matrice di numeri casuali. Stampa poi il numero di numeri pari contenuti in ogni riga della matrice (riga per riga separatamente) e il numero di numeri dispari contenuti in ogni colonna della matrice. Anche in questo caso definisci una o due funzioni per farlo (a scelta).

## Array di stringhe o matrici di caratteri

1. Scrivi un programma che chieda all'utente una serie di nomi che devono essere tutti memorizzati in un array di stringhe (e sarà quindi una matrice, ad esempio char nomi[n][m] con n numero dei nomi e m numero massimo di caratteri per stringa). Il numero di nomi da inserire deve essere chiesto all'utente. Quando viene inserito un nome bisogna controllare che inizi con una lettera e se questa è minuscola deve essere fatta diventare maiuscola. Alla fine stampa tutte le stringhe.
2. Continua l’esercizio precedente aggiungendo le seguenti funzionalità:

* Cerca e stampa il nome più lungo
* Cerca e stampa il nome che in ordine alfabetico è il primo
* Chiedi all'utente un nome e digli se il nome è presente nell'elenco dei nomi (usa strcmp).

1. Dato un array di stringhe (quindi una matrice), riempito come preferisci, stampa la stringa più lunga, la più corta e la lunghezza media delle stringhe. I tre risultati devono essere ottenuti per mezzo di apposite funzioni.
2. Scrivi e usa una funzione che dato un array di stringhe (quindi una matrice), restituisca il numero di parole palindrome contenute nell’array. Riempi l’array come preferisci.
3. Scrivi un programma che permetta di rappresentare il campo di gioco di “Campo minato”. Il campo deve essere formato da una superficie suddivisa in caselle quadrate, quindi avrai una grande tabella (ad esempio di dimensioni 8x12 ma puoi scegliere diversamente). Sul campo devono essere disposte a caso una serie di mine (suggerisco 16 ma puoi cambiare). Fai poi in modo che ogni casella che non sia una mina abbia un numero che rappresenti il numero di mine adiacenti (anche in diagonale quindi un numero da 1 a 8) lasciando vuote le celle che non hanno mine adiacenti (non scrivere 0)
4. Crea un programma di gestione di una agenda. L’agenda deve permettere di visualizzare e inserire i dati riguardanti i propri contatti. Per ogni persona devono essere memorizzati nome cognome e numero di telefono. La visualizzazione dei dati deve avvenire in ordine alfabetico per nome o cognome (scegli tu). Aggiungi poi una funzione per cancellare un contatto in una specifica posizione dell’agenda. (Non avendo ancora fatto la gestione dei files ogni volta l’agenda partirà da uno stato predefinito che puoi decidere tu). Aggiungi anche un’interfaccia utilizzabile.
5. Scrivi un programma che chieda all’utente di scrivere una frase e le legga con la funzione gets o fgets. Le parole della frase devono poi essere stampate una per volta andando sempre a capo tra una parola e l’altra.
6. Scrivi un programma che chieda all’utente di scrivere una serie di parole lette una ad una. Queste parole devono essere unite a formare un’unica stringa che deve essere infine stampata.
7. Scrivi un programma che chieda all’utente di scrivere una frase e le legga con la funzione gets o fgets (otterrai quindi un’unica stringa). Le parole della frase devono poi essere ordinate e stampate in ordine alfabetico. (consiglio: prendi le parole dalla stringa, mettile in un array e ordina l’array)

## Algoritmi di ordinamento

1. Carica un array di numeri casuali e utilizza il selection sort per ordinarlo. Deve essere fatta una stampa dell’array prima e dopo l’ordinamento
2. Carica un array di numeri casuali e utilizza il bubble sort per ordinarlo. Deve essere fatta una stampa dell’array prima e dopo l’ordinamento
3. Carica un array di numeri casuali e utilizza la funzione sort della libreria standard per ordinarlo. Deve essere fatta una stampa dell’array prima e dopo l’ordinamento
4. Scrivi un programma che dato un array di numeri interi, sia in grado di:

* controllare se l'array è palindromo,
* invertire l'ordine dei numeri dell'array,
* ordinare in ordine decrescente l'array.

1. Dato un array di stringhe (quindi una matrice di caratteri), che puoi riempire come preferisci, ordina l’array in ordine alfabetico prima crescente e poi decrescente. Devono essere inserite le stampe dell’intero array prima e dopo ogni ordinamento.
2. Crea un programma di gestione di una agenda. L’agenda deve permettere di visualizzare e inserire i dati riguardanti i propri contatti. Per ogni persona devono essere memorizzati nome cognome e numero di telefono. La visualizzazione dei dati deve avvenire in ordine alfabetico per nome o cognome (scegli tu). (Non avendo ancora fatto la gestione dei files ogni volta l’agenda partirà da uno stato predefinito che puoi decidere tu).

# Funzioni

1. Scrivi e utilizza una funzione per calcolare e restituire la somma tra due numeri
2. Scrivi e utilizza 4 funzioni che effettuano le 4 operazioni e restituiscono il risultato.
3. Scrivi e utilizza una funzione che dati due valori e un simbolo di una delle 4 operazioni effettua quella operazione e restituisce il risultato
4. Scrivi e utilizza le seguenti funzioni:

* raddoppia un numero passando parametri per valore,
* raddoppia un numero passando parametri per indirizzo,
* raddoppia due numeri contemporaneamente.

1. Scrivi e utilizza una funzione che restituisca il prodotto di due numeri e alla fine dell’esecuzione lasci modificati i due numeri in modo che valgano il doppio di prima.
2. Funzione che date le coordinate di due punti nel piano, calcolino e restituiscano le coordinate del punto medio tra i due (senza approssimazioni).
3. Funzione che dato il tempo di caduta di un oggetto, calcola l'altezza da cui è caduto l'oggetto e la velocità  di impatto. Le formule che ti servono sono: h = 1/2 \* g \* t^2, con g=9,81 e v=g\*t. Almeno uno dei due valori deve essere restituito normalmente
4. Funzione che prevede se un oggetto si rompe cadendo da una certa altezza. Quello che ti serve sapere è quindi l’altezza da cui cade l’oggetto e la velocità massima di impatto che l’oggetto può sopportare. Le formule che ti servono sono: h = 1/2 \* g \* t^2, con g=9,81 e v=g\*t.

* Modifica l’esercizio facendo in modo che la funzione restituisca anche la velocità di impatto.

1. Scrivi ed utilizza una funzione che dati due numeri li aumenta entrambi di un certo valore passato come parametro.
2. Scrivi e utilizza una funzione che dato un array di numeri, trova e restituisce il massimo e il minimo numero presenti.
3. Scrivi e utilizza una funzione che, data una stringa, restituisce true se la stringa inizia con una lettera maiuscola.
4. Funzione che data una stringa, la stampa separando ogni carattere della stringa con uno spazio.
5. Scrivi e utilizza una funzione che riceve come parametri due stringhe e restituisce una stringa che è la concatenazione delle altre due.
6. Scrivi un programma che, dati n voti in un array, li passi ad una funzione che restituisca la media dei voti escludendo il voto più alto e il voto più basso.
7. Scrivi e utilizza una funzione che stampa una stringa ricevuta come parametro.
8. Scrivi un programma che utilizzi una funzione che, ricevuto un array di numeri interi, restituisce il numero che compare con più frequenza (in caso di parità restituisce il primo numero trovato con la frequenza maggiore).
9. Scrivi una funzione che scambia il contenuto di due variabili intere (i valori devono rimanere scambiati per chi chiama la funzione).
10. Scrivi un programma che sappia riordinare un array. Per farlo dividi le seguenti funzionalità in diverse funzioni:

* funzione che scambia il valore di due variabili
* funzione che riceve un array e lo riempie di numeri casuali
* funzione che stampa un array
* funzione che riceve un array e lo ordina. Lo scambio del valore delle variabili deve essere fatto usando la funzione descritta sopra
* main che dichiara un array e poi usando le funzioni descritte sopra lo riempie di numeri casuali, lo stampa, lo ordina e poi lo ristampa.

1. Scrivi e utilizza una funzione che dato un array restituisce i valori massimi e minimi dell'array
2. Scrivi e utilizza una funzione che calcola il quoziente tra due numeri interi e restituisce, oltre al quoziente anche il resto della divisione.
3. Scrivi e utilizza una funzione che data una stringa, fa in modo che la prima lettera sia maiuscola e tutte le altre minuscole. i caratteri che non sono lettere non devono essere modificati.
4. Scrivi una funzione che calcola il massimo comune divisore tra due numeri.   
   Scrivi ed utilizza poi una funzione che dati due valori numeratore e denominatore, semplifica la frazione, usando la funzione mcd appena definita.
5. Scrivi ed utilizza una funzione che riceva come parametri tre numeri e che restituisca true se questi numeri possono essere le misure di un triangolo. In questo caso la funzione deve anche restituire una stringa in cui c'è scritto il tipo di triangolo (equilatero, isoscele, scaleno).
6. Scrivi e utilizza le versioni iterativa e ricorsiva di una funzione che dato un array di interi, calcola e restituisce la massima differenza tra due numeri consecutivi.

## Funzioni ricorsive

1. Scrivi ed utilizza una funzione iterativa che calcola il fattoriale di un numero
2. Scrivi ed utilizza una funzione ricorsiva che calcola il fattoriale di un numero
3. Scrivi una funzione iterativa che calcola l'ennesimo numero della successione di Fibonacci.
4. Scrivi una funzione ricorsiva che calcola l'ennesimo numero della successione di Fibonacci.
5. Scrivi una funzione ricorsiva che effettui la moltiplicazione tra due numeri
6. Scrivi una funzione che calcola ricorsivamente il quoziente tra due numeri utilizzando le sottrazioni.
7. Scrivi una funzione ricorsiva che stampi un array di numeri.
8. Scrivi una funzione ricorsiva che stampi al contrario un array di numeri.
9. Scrivi e utilizza le versioni iterativa e ricorsiva di una funzione che riceve un array e inverte l'ordine dei numeri in esso contenuti.
10. Funzione ricorsiva per trovare i valori massimi e minimi di un vettore.
11. Funzione ricorsiva per cercare un valore all’interno di un vettore non ordinato (ricerca lineare). Risultato: –1 se non lo trova, altrimenti l’indice della posizione dove è stato trovato.
12. Scrivere una funzione ricorsiva per cercare un valore all’interno di un vettore ordinato (ricerca dicotomica). Risultato: – 1 o l’indice.   
    Indicazioni: se il vettore è ordinato, iniziate a controllare il numero a metà del vettore, se il numero controllato è più piccolo di quello da trovare, cercherò nella parte del vettore a sinistra (richiamo la funzione indicando come intervallo in cui cercare solo la parte a sinistra), se è più piccolo dovrò cercare a destra, se è uguale l'ho trovato.
13. Scrivere una funzione ricorsiva che riceve un array di interi e restituisce true se gli elementi sono tutti dispari e false se non lo sono.
14. Scrivere una funzione ricorsiva che riceve un array di interi e restituisce il prodotto degli elementi negativi.
15. Scrivi una funzione di ordinamento ricorsiva.

# Struct

1. Scrivi un programma che definisca un punto e poi dati due punti scelti da te sia in grado, per mezzo di due apposite funzioni, di calcolare la distanza fra due punti e trovare il punto medio.
2. Scrivi un programma per gestire un corso. Al corso sono iscritte delle persone, massimo 30. Per ogni persona dovete sapere nome e cognome e ad ognuno alla fine del corso viene assegnato un voto da 1 a 10. Alla fine stampa tutti i dati di ogni studente, compresi i voti.
3. Scrivi e utilizza un insieme di funzioni da utilizzare per fare calcoli con le frazioni. In pratica devono essere definite le funzioni per fare almeno le 4 operazioni (magari anche altre operazioni). Deve anche essere fatta una funzione per effettuare la semplificazione delle frazioni (puoi decidere se creare e usare anche un'altra funzione mcd o utilizzare un altro metodo). Scrivi tutte queste funzioni in un file separato e poi utilizza queste funzioni per fare dei calcoli con le frazioni in un programma.   
   In una prima versione del programma considera numeratore e denominatore come due variabili intere separate. In una seconda versione definisci invece una struct frazione che contiene numeratore e denominatore. Le funzioni dovranno essere modificate di conseguenza.
4. Definisci una struct punto. Scrivi ed utilizza poi le funzioni:

* Funzione per stampare una rappresentazione del punto (es. A(1,2))
* Funzione per calcolare il punto medio
* Funzione per calcolare la retta passante tra due punti. Dovrai anche definire la struct retta che contiene le informazioni necessarie ad identificare una retta e definire una funzione per stamparla in modo standard (y=mx+c)
* Funzione che verifica se tre punti sono allineati.

1. Completa il seguente programma in cui è stato dichiarato un array di stringhe in cui sono scritti nomi, cognomi e date di nascita e morte di esploratori famosi. Devi definire le strutture dati adeguate a contenere queste informazioni: una struct esploratore e un array di esploratori. Usa la funzione sscanf per leggere i dati dalla stringa dati e riempi le strutture dati da te definite. scrivi poi una serie di funzioni che lavorino sulle tue strutture dati:

* una funzione che stampa l'array di esploratori.
* una funzione che ordina l'array in ordine alfabetico secondo il nome degli esploratori
* una funzione che ordina l'array in ordine crescente secondo le date di nascita degli esploratori

Consigli e indicazioni extra:

* - La funzione sscanf l'abbiamo usata pochissimo in passato ma trovate facilmente online indicazioni su come si usa.
* - Alcuni potrebbero trovare comodo definire una struttura dati contenente l'array degli esploratori insieme al numero di esploratori contenuti nell'array.

int main(){

char dati[15][50] = {

"Marco Polo 1254 1324",

"Cristoforo Colombo 1451 1506",

"Amerigo Vespucci 1454 1512",

"Francisco Pizarro 1475 1541",

"Ferdinando Magellano 1480 1521",

"Hernan Cortez 1485 1547",

"Walter Raleigh 1552 1618",

"Henry Hudson 1570 1611",

"James Cook 1728 1779",

"Charles Darwin 1809 1882",

"Kit Carson 1809 1866",

"David Livingstone 1813 1873",

"Charles Foucauld 1858 1916",

"Ronald Amundsen 1872 1928",

"Ernest Shakleton 1874 1922",

};

}

# Files in C

1. Scrivi un programma che scriva su un file un elenco di nomi, chiuda il file, lo riapra in lettura lo legga tutto e stampi ciò che ha letto.
2. Scrivi un programma che scriva su un file un elenco di nomi (o numeri se preferisci), chieda poi all'utente un nome da cercare nel file e stampi sullo schermo l'esito della ricerca (trovato o no). Le operazioni di scrittura su file devono essere eseguite da una funzione che riceve come parametro la stringa contenente il nome del file. Le operazioni di ricerca devono essere svolte da una funzione che riceve come parametro la stringa contenente il nome del file e la stringa da cercare.

* Continua l'esercizio aggiungendo una funzione che dato il nome del file e il numero di riga, cerca, se esiste, la riga e la restituisce.
* Aggiungi anche la possibilità di chiedere all'utente se vuole aggiungere dei nomi a quelli già presenti (aprendo poi il file in modalità append e aggiungendo i nomi dati dall'utente)

1. Scrivi un programma che permetta di leggere e scrivere informazioni in un file di testo che deve essere già esistente. Le operazioni che l'utente può scegliere devono essere:

* inserire un nome e se esso non è presente nel file, aggiungerlo.
* visualizzare il numero di righe del file
* inserire un nome e se esso è presente nel file, cancellare la riga dal file (non fare diventare quella riga vuota ma modificare proprio tutto il file. ti conviene rileggere tutti i nomi e riscriverli da capo)

1. Scrivi un programma che permetta di leggere e scrivere informazioni in un file di testo che deve essere già esistente (crealo tu secondo le indicazioni di seguito). Le informazioni riguardano i voti di alcuni studenti. Le operazioni che l'utente può scegliere devono essere:

* leggere i voti degli studenti e salvarli in array o matrici
* calcolare la media per ogni studente e scrivere in un altro file gli esiti.   
  Es.  
   Francesco 6 Promosso   
   Paolo 5,6 Bocciato   
   Emanuele 7,2 Promosso   
   ...
* dato il nome di uno studente, cercare i suoi voti,

calcolarne la media e stampare su schermo voti e media

Per iniziare in modo semplice, il file conterrà un numero fisso di voti

per ogni studente, 3.

Es:

Francesco 6 7.5 6

Paolo 5 5.5 6

Emanuele 7 5 5.5

Come variante (che prevede l’utilizzo degli stringstream) ogni riga contiene, oltre al nome,

un numero variabile di voti.

Es:

Francesco 6 6

Paolo 5 5.5 6 6

Emanuele 7 5 5.5

1. Scrivi una funzione parametrica in grado di modificare un file di testo di nome "miofile.txt" letto da disco in modo tale che, se l'ultimo elemento della linea è una virgola, la linea successiva venga eliminata
2. Scrivi una funzione che conta e restituisce il numero di righe di un file.   
   Scrivi poi un'altra funzione che calcola per ogni riga il numero di parole contenute nella riga. I valori vanno inseriti in un array o un vettore da restituire al main.
3. Scrivi un programma che legga un file di testo e che scriva in un secondo file, lo stesso testo modificato in modo che tutte le lettere siano minuscole. Il file di testo iniziale generalo tu come vuoi.
4. Scrivi un programma che legga un file di testo e che aggiunga alla fine di tale file, dopo un paio di righe vuote, lo stesso testo modificato in modo che tutte le lettere siano minuscole. Il file di testo iniziale generalo tu come vuoi. Attento a fare in modo che se il programma viene eseguito più volte, le righe da considerare sono solo quelle iniziali.

### Che richiedono anche le strutture dati

1. Scrivi un programma che legga i dati contenuti in un file di testo (testo riportato di seguito) e li inserisca in opportune strutture dati (richiesta la conoscenza delle classi). Di questi dati il programma deve fare le seguenti cose:

* stamparli sullo schermo
* ordinarli in ordine decrescente di voto
* stamparli nuovamente in ordine
* scriverli in ordine in un secondo file che puoi nominare come vuoi

Dati da mettere nel file di testo:

Amici 7.00

Biella 9.00

Brescia 6.00

Carolla 8.00

Cunegatti 7.00

DeBella 4.00

DeVecchi 9.00

Fumagalli 7.00

Galimberti 9.00

Germanò 9.00

Gubellini 9.00

Lepore 5.00

Maconi 6.00

Mariani 8.00

Mattavelli 9.00

Oggiano 4.00

Passoni 5.00

Pastori 4.00

Pirovano 7.00

Rudi 10.00

Russell 6.00

Sogos 4.00

Tezza 6.00

Varisco 8.00

1. Scrivi un programma che legga i dati contenuti in un file di testo (testo riportato di seguito) e li inserisca in opportune strutture dati che rappresentino il registro dei voti per una materia. Il programma deve poi calcolare la media dei voti per ogni singolo studente e aggiungerla alla struttura dati (che quindi conterrà anche una variabile media). Il programma deve mostrare tutti i dati raccolti e calcolati sullo schermo e salvarli in un secondo file di testo.

Amici 7.00 8.00 7.00

Biella 8.00 9.50 7.00

Brescia 5.00 7.50 9.00

Carolla 7.00 8.50 8.50

Cunegatti 7.00 7.00 5.50

DeBella 4.00 4.00 4.50

DeVecchi 8.00 10.00 6.00

Fumagalli 8.50 6.00 4.50

Galimberti 9.00 9.00 9.00

Germanò 8.50 9.00 7.50

Gubellini 8.00 10.00 7.00

Lepore 5.50 4.00 3.00

Maconi 7.50 5.00 7.50

Mariani 8.00 8.00 7.00

Mattavelli 9.00 9.50 8.50

Oggiano 5.00 3.00 3.00

Passoni 5.00 6.00 6.00

Pastori 3.50 5.00 7.00

Pirovano 6.50 7.50 7.50

Rudi 9.50 10.00 10.00

Russell 7.50 4.50 5.50

Sogos 4.00 3.50 3.50

Tezza 7.00 5.50 5.50

Varisco 8.00 9.00 8.50

1. Scrivi un programma che legga i dati contenuti in un file di testo (testo riportato di seguito) e li inserisca in opportune strutture dati che rappresentino il registro dei voti per una materia. Il programma deve poi calcolare la media dei voti per ogni singolo studente e aggiungerla alla struttura dati (che quindi conterrà anche una variabile media). Il programma deve mostrare tutti i dati raccolti e calcolati sullo schermo e salvarli in un secondo file di testo. Questa versione del programma è più difficile della precedente perché si può notare che i cognomi possono essere formati da più parole e sono separati dai voti da un “:”, inoltre il numero di voti varia per ogni studente.

Amici: 7.00 8.00 7.00

Biella: 8.00 9.50 7.00

Brescia: 5.00 7.50 9.00

Carolla: 7.00 8.50 8.50

Cunegatti: 7.00 7.00

De Bella: 4.00 4.00 4.50 5.00

De Vecchi: 8.00 10.00 6.00

Fumagalli: 8.50 6.00 4.50

Galimberti: 9.00 9.00 9.00

Germanò: 8.50 9.00 7.50

Gubellini: 8.00 10.00 7.00

Lepore: 5.50 4.00 3.00 5.00

Maconi: 7.50 5.00

Mariani: 8.00 8.00 7.00

Mattavelli: 9.00 9.50 8.50

Oggiano: 5.00 3.00 3.00 2.00

Passoni: 5.00 6.00 6.00 7.50

Pastori: 3.50 5.00 7.00 2.00

Pirovano: 6.50 7.50 7.50

Rudi: 9.50 10.00 10.00

Russell: 7.50 4.50

Sogos: 4.00 3.50 3.50 2.00

Tezza: 7.00 5.50 5.50

Varisco: 8.00 9.00 8.50

Pirovano 6.50 7.50 7.50

Rudi 9.50 10.00 10.00

Russell 7.50 4.50 5.50

Sogos 4.00 3.50 3.50

Tezza 7.00 5.50 5.50

Varisco 8.00 9.00 8.50

1. Scrivi un programma che legga i dati contenuti in un file di testo (testo riportato di seguito) e li inserisca in opportune strutture dati. Osservando il file possiamo notare alcune cose:

* Il cognome è separato dal resto della riga da un simbolo ':'.
* I cognomi possono essere formati da più di una parola.
* il numero di voti per ogni studente non è costante.

Per poter leggere il file dovrai utilizzare correttamente la funzione getline. Ti consiglierei di leggere una riga per volta e poi lavorare con uno stringstream sulla riga. Dovrai utilizzare la versione completa di getline per leggere i cognomi: getline(fs, riga, ':')

Per gestire i dati dovrai anche modificare la classe studente che dovrà contenere invece che un singolo voto, un vettore di voti. Aggiungi anche le funzionalità per calcolare le medie per ogni studente (e magari salvarla in un attributo dello studente) e la media totale dei voti medi degli studenti.

Dati da mettere nel file di testo:

Amici: 7.00 8.00 7.00

Biella: 8.00 9.50 7.00

Brescia: 5.00 7.50 9.00

Carolla: 7.00 8.50 8.50

Cunegatti: 7.00 7.00 5.50

De Bella: 4.00 4.00 4.50 5.00

De Vecchi: 8.00 10.00 6.00

Fumagalli: 8.50 6.00 4.50

Galimberti: 9.00 9.00 9.00

Germanò: 8.50 9.00 7.50

Gubellini: 8.00 10.00 7.00

Lepore: 5.50 4.00 3.00 5.00

Maconi: 7.50 5.00 7.50

Mariani: 8.00 8.00 7.00

Mattavelli: 9.00 9.50 8.50

Oggiano: 5.00 3.00 3.00 2.00

Passoni: 5.00 6.00 6.00 7.50

Pastori: 3.50 5.00 7.00 2.00

Pirovano: 6.50 7.50 7.50

Rudi: 9.50 10.00 10.00

Russell: 7.50 4.50 5.50

Sogos: 4.00 3.50 3.50 2.00

Tezza: 7.00 5.50 5.50

Varisco: 8.00 9.00 8.50

1. Leggi le informazioni riguardanti una serie di libri dal file allegato "09-scrittori.txt" e:
   1. carica i dati in un array di libri (definisci un'apposita struttura libro)
   2. ordina i libri per data di scrittura e riscrivili in un altro file nello stesso formato del file originale
   3. scrivi e usa una funzione che riceve i dati già caricati e che restituisce il libro più recente
   4. scrivi e usa una funzione che restituisca il secolo in cui sono stati scritti più libri

Divina Commedia;Dante Alighieri;1321

Promessi Sposi;Alessandro Manzoni;1840

Il Decamerone;Alessandro Boccaccio;1350

L'Orlando Furioso;Ludovico Ariosto;1516

Il fu Mattia Pascal;Luigi Pirandello;1904

Ultime lettere di Jacopo Ortis,Ugo Foscolo;1802

Se questo è un uomo;Primo Levi;1947

Il barone rampante;Italo Calvino;1957

I Malavoglia;Giovanni Verga;1881

Il Principe; Niccolò Macchiavelli;1532

La Gerusalemme Liberata;Torquato Tasso;1581

Cuore;Edmondo De Amicis;1886

Il deserto dei Tartari;Dino Buzzati;1940

La coscienza di Zeno;Italo Svevo;1923

Pinocchio;Carlo Collodi;1883

Il piacere;Gabriele D'Annunzio;1889

Myricae;Giovanni Pascoli;1891

Canti;Giacomo Leopardi;1831

Uno, nessuno e centomila;Luigi Pirandello;1926

Il nome della rosa;Umberto Eco;1980

La casa in collina;Cesare Pavese;1948

Il gattopardo;Giuseppe Tomasi di Lampedusa;1958

I Vicerè;Federico De Roberto;1894

Mastro don Gesualdo;Giovanni Verga;1889

1. Scrivi un programma che sia in grado di leggere da file i dati riportati di seguito e salvarli in opportune strutture dati (punto e triangolo). Il programma deve poi stampare sia su schermo che in un altro file i dati riguardanti ogni triangolo e se esso è scaleno, isoscele o equilatero. Per decidere il tipo di triangolo devono essere usate tre diverse funzioni che controllano ognuna se il triangolo passato come parametro è di uno dei tre tipi richiesti.

Input

(1, 0) (3, 3) (-0.6, 3.23)

(1, 0) (1, 4) (-2.46, 2)

(2, 0) (5, 3) (2, 6)

(3, 0) (4.4, 3.3) (2.35, 4.8)

(2, 2) (4, 2) (4, 6)

Output

(1.000000, 0.000000) (3.000000, 3.000000) (-0.600000, 3.230000) - equilatero

(1.000000, 0.000000) (1.000000, 4.000000) (-2.460000, 2.000000) - equilatero

(2.000000, 0.000000) (5.000000, 3.000000) (2.000000, 6.000000) - isoscele

(3.000000, 0.000000) (4.400000, 3.300000) (2.350000, 4.800000) - scaleno

(2.000000, 2.000000) (4.000000, 2.000000) (4.000000, 6.000000) - scaleno

# Divisione programmi in progetti di più files

Per apprendere questo argomento è possibile modificare gli esercizi contenenti funzioni in modo da spostare la dichiarazione e l’implementazione delle funzioni in file diversi da quelli contenenti il main.

# Elementi fondamentali del C++

## IO, String e Vector

Per studiare questi argomenti è consigliato provare a risolvere gli esercizi proposti per imparare ad utilizzare stringhe e array in C.

Ho raccolto comunque altri esercizi svolti per questi argomenti.

1. Chiedi all’utente di scrivere una parola, memorizzala in una stringa, conta da quante lettere è composta in due diversi modi e poi comunica i valori calcolati all’utente. Il primo metodo è contare ad una ad una le lettere, il secondo è usare la funzione size della stringa.
2. Dichiara un vettore e aggiungi ad esso 5 numeri casuali. Stampa il vettore prima dall’inizio alla fine poi dalla fine all’inizio.
3. Dichiara un vettore di stringhe e riempilo di stringhe date dall’utente in input. Decidi un meccanismo per far scegliere all’utente quando terminare
4. Utilizza la funzione sort per ordinare e stampare i vettori definiti nei due esercizi precedenti.
5. Dichiara un vettore e aggiungi ad esso 5 numeri casuali. Scrivi ed utilizza una funzione che restituisce la media degli elementi del vettore.

# Streams

Per risolvere gli esercizi riguardanti i seguenti argomenti: streams e classi, è necessario in molti casi conoscere entrambi gli argomenti, con l’eccezione dei primi esercizi di base che ti permetteranno di imparare un argomento senza ancora aver studiato l’altro.

1. Scrivi un programma che chieda all’utente di scrivere una frase e le legga con la funzione getline. Le parole della frase devono poi essere stampate una per volta andando sempre a capo tra una parola e l’altra.
2. Scrivi un programma che chieda all’utente di scrivere una serie di parole lette una ad una. Queste parole devono essere unite a formare un’unica stringa che deve essere infine stampata.
3. Scrivi un programma che chieda all’utente di scrivere una frase e le legga con la funzione getline (otterrai quindi un’unica stringa). Le parole della frase devono poi essere ordinate e stampate in ordine alfabetico. (consiglio: prendi le parole dalla stringa, mettile in un vettore e ordina il vettore)
4. Scrivi un programma che scriva su un file un elenco di nomi, chiuda il file, lo riapra in lettura lo legga tutto e stampi ciò che ha letto.
5. Scrivi un programma che scriva su un file un elenco di nomi (o numeri se preferisci), chieda poi all'utente un nome da cercare nel file e stampi sullo schermo l'esito della ricerca (trovato o no). Le operazioni di scrittura su file devono essere eseguite da una funzione che riceve come parametro la stringa contenente il nome del file. Le operazioni di ricerca devono essere svolte da una funzione che riceve come parametro la stringa contenente il nome del file e la stringa da cercare.

* Continua l'esercizio aggiungendo una funzione che dato il nome del file e il numero di riga, cerca, se esiste, la riga e la restituisce.
* Aggiungi anche la possibilità di chiedere all'utente se vuole aggiungere dei nomi a quelli già presenti (aprendo poi il file in modalità append e aggiungendo i nomi dati dall'utente)

1. Scrivi un programma che permetta di leggere e scrivere informazioni in un file di testo che deve essere già esistente. Le operazioni che l'utente può scegliere devono essere:

* inserire un nome e se esso non è presente nel file, aggiungerlo.
* visualizzare il numero di righe del file
* inserire un nome e se esso è presente nel file, cancellare la riga dal file (non fare diventare quella riga vuota ma modificare proprio tutto il file. ti conviene rileggere tutti i nomi e riscriverli da capo)

1. Scrivi un programma che permetta di leggere e scrivere informazioni in un file di testo che deve essere già esistente (crealo tu secondo le indicazioni di seguito). Le informazioni riguardano i voti di alcuni studenti. Le operazioni che l'utente può scegliere devono essere:

* leggere i voti degli studenti e salvarli in array o matrici
* calcolare la media per ogni studente e scrivere in un altro file gli esiti.   
  Es.  
   Francesco 6 Promosso   
   Paolo 5,6 Bocciato   
   Emanuele 7,2 Promosso   
   ...
* dato il nome di uno studente, cercare i suoi voti,

calcolarne la media e stampare su schermo voti e media

Per iniziare in modo semplice, il file conterrà un numero fisso di voti

per ogni studente, 3.

Es:

Francesco 6 7.5 6

Paolo 5 5.5 6

Emanuele 7 5 5.5

Come variante (che prevede l’utilizzo degli stringstream) ogni riga contiene, oltre al nome,

un numero variabile di voti.

Es:

Francesco 6 6

Paolo 5 5.5 6 6

Emanuele 7 5 5.5

1. Scrivi una funzione parametrica in grado di modificare un file di testo di nome "miofile.txt" letto da disco in modo tale che, se l'ultimo elemento della linea è una virgola, la linea successiva venga eliminata
2. Scrivi una funzione che conta e restituisce il numero di righe di un file.   
   Scrivi poi un'altra funzione che calcola per ogni riga il numero di parole contenute nella riga. I valori vanno inseriti in un array o un vettore da restituire al main.
3. Scrivi un programma che legga un file di testo e che scriva in un secondo file, lo stesso testo modificato in modo che tutte le lettere siano minuscole. Il file di testo iniziale generalo tu come vuoi.
4. Scrivi un programma che legga un file di testo e che aggiunga alla fine di tale file, dopo un paio di righe vuote, lo stesso testo modificato in modo che tutte le lettere siano minuscole. Il file di testo iniziale generalo tu come vuoi. Attento a fare in modo che se il programma viene eseguito più volte, le righe da considerare sono solo quelle iniziali.

### Che richiedono anche le classi

1. Scrivi un programma che legga i dati contenuti in un file di testo (testo riportato di seguito) e li inserisca in opportune strutture dati (richiesta la conoscenza delle classi). Di questi dati il programma deve fare le seguenti cose:

* stamparli sullo schermo
* ordinarli in ordine decrescente di voto
* stamparli nuovamente in ordine
* scriverli in ordine in un secondo file che puoi nominare come vuoi

Dati da mettere nel file di testo:

Amici 7.00

Biella 9.00

Brescia 6.00

Carolla 8.00

Cunegatti 7.00

DeBella 4.00

DeVecchi 9.00

Fumagalli 7.00

Galimberti 9.00

Germanò 9.00

Gubellini 9.00

Lepore 5.00

Maconi 6.00

Mariani 8.00

Mattavelli 9.00

Oggiano 4.00

Passoni 5.00

Pastori 4.00

Pirovano 7.00

Rudi 10.00

Russell 6.00

Sogos 4.00

Tezza 6.00

Varisco 8.00

1. Scrivi un programma che legga i dati contenuti in un file di testo (testo riportato di seguito) e li inserisca in opportune strutture dati che rappresentino il registro dei voti per una materia. Il programma deve poi calcolare la media dei voti per ogni singolo studente e aggiungerla alla struttura dati (che quindi conterrà anche una variabile media). Il programma deve mostrare tutti i dati raccolti e calcolati sullo schermo e salvarli in un secondo file di testo.

Amici 7.00 8.00 7.00

Biella 8.00 9.50 7.00

Brescia 5.00 7.50 9.00

Carolla 7.00 8.50 8.50

Cunegatti 7.00 7.00 5.50

DeBella 4.00 4.00 4.50

DeVecchi 8.00 10.00 6.00

Fumagalli 8.50 6.00 4.50

Galimberti 9.00 9.00 9.00

Germanò 8.50 9.00 7.50

Gubellini 8.00 10.00 7.00

Lepore 5.50 4.00 3.00 5.00

Maconi 7.50 5.00 7.50

Mariani 8.00 8.00 7.00

Mattavelli 9.00 9.50 8.50

Oggiano 5.00 3.00 3.00

Passoni 5.00 6.00 6.00

Pastori 3.50 5.00 7.00

Pirovano 6.50 7.50 7.50

Rudi 9.50 10.00 10.00

Russell 7.50 4.50 5.50

Sogos 4.00 3.50 3.50

Tezza 7.00 5.50 5.50

Varisco 8.00 9.00 8.50

1. Scrivi un programma che legga i dati contenuti in un file di testo (testo riportato di seguito) e li inserisca in opportune strutture dati che rappresentino il registro dei voti per una materia. Il programma deve poi calcolare la media dei voti per ogni singolo studente e aggiungerla alla struttura dati (che quindi conterrà anche una variabile media). Il programma deve mostrare tutti i dati raccolti e calcolati sullo schermo e salvarli in un secondo file di testo. Questa versione del programma è più difficile della precedente perché si può notare che i cognomi possono essere formati da più parole e sono separati dai voti da un “:”, inoltre il numero di voti varia per ogni studente.

Amici: 7.00 8.00 7.00

Biella: 8.00 9.50 7.00

Brescia: 5.00 7.50 9.00

Carolla: 7.00 8.50 8.50

Cunegatti: 7.00 7.00

De Bella: 4.00 4.00 4.50 5.00

De Vecchi: 8.00 10.00 6.00

Fumagalli: 8.50 6.00 4.50

Galimberti: 9.00 9.00 9.00

Germanò: 8.50 9.00 7.50

Gubellini: 8.00 10.00 7.00

Lepore: 5.50 4.00 3.00 5.00

Maconi: 7.50 5.00

Mariani: 8.00 8.00 7.00

Mattavelli: 9.00 9.50 8.50

Oggiano: 5.00 3.00 3.00 2.00

Passoni: 5.00 6.00 6.00 7.50

Pastori: 3.50 5.00 7.00 2.00

Pirovano: 6.50 7.50 7.50

Rudi: 9.50 10.00 10.00

Russell: 7.50 4.50

Sogos: 4.00 3.50 3.50 2.00

Tezza: 7.00 5.50 5.50

Varisco: 8.00 9.00 8.50

Pirovano 6.50 7.50 7.50

Rudi 9.50 10.00 10.00

Russell 7.50 4.50 5.50

Sogos 4.00 3.50 3.50

Tezza 7.00 5.50 5.50

Varisco 8.00 9.00 8.50

1. Scrivi un programma che legga i dati contenuti in un file di testo (testo riportato di seguito) e li inserisca in opportune strutture dati. Osservando il file possiamo notare alcune cose:

* Il cognome è separato dal resto della riga da un simbolo ':'.
* I cognomi possono essere formati da più di una parola.
* il numero di voti per ogni studente non è costante.

Per poter leggere il file dovrai utilizzare correttamente la funzione getline. Ti consiglierei di leggere una riga per volta e poi lavorare con uno stringstream sulla riga. Dovrai utilizzare la versione completa di getline per leggere i cognomi: getline(fs, riga, ':')

Per gestire i dati dovrai anche modificare la classe studente che dovrà contenere invece che un singolo voto, un vettore di voti. Aggiungi anche le funzionalità per calcolare le medie per ogni studente (e magari salvarla in un attributo dello studente) e la media totale dei voti medi degli studenti.

Dati da mettere nel file di testo:

Amici: 7.00 8.00 7.00

Biella: 8.00 9.50 7.00

Brescia: 5.00 7.50 9.00

Carolla: 7.00 8.50 8.50

Cunegatti: 7.00 7.00 5.50

De Bella: 4.00 4.00 4.50 5.00

De Vecchi: 8.00 10.00 6.00

Fumagalli: 8.50 6.00 4.50

Galimberti: 9.00 9.00 9.00

Germanò: 8.50 9.00 7.50

Gubellini: 8.00 10.00 7.00

Lepore: 5.50 4.00 3.00 5.00

Maconi: 7.50 5.00 7.50

Mariani: 8.00 8.00 7.00

Mattavelli: 9.00 9.50 8.50

Oggiano: 5.00 3.00 3.00 2.00

Passoni: 5.00 6.00 6.00 7.50

Pastori: 3.50 5.00 7.00 2.00

Pirovano: 6.50 7.50 7.50

Rudi: 9.50 10.00 10.00

Russell: 7.50 4.50 5.50

Sogos: 4.00 3.50 3.50 2.00

Tezza: 7.00 5.50 5.50

Varisco: 8.00 9.00 8.50

1. Leggi le informazioni riguardanti una serie di libri dal file allegato "09-scrittori.txt" e:
   1. carica i dati in un array di libri (definisci un'apposita struttura libro)
   2. ordina i libri per data di scrittura e riscrivili in un altro file nello stesso formato del file originale
   3. scrivi e usa una funzione che riceve i dati già caricati e che restituisce il libro più recente
   4. scrivi e usa una funzione che restituisca il secolo in cui sono stati scritti più libri

Divina Commedia;Dante Alighieri;1321

Promessi Sposi;Alessandro Manzoni;1840

Il Decamerone;Alessandro Boccaccio;1350

L'Orlando Furioso;Ludovico Ariosto;1516

Il fu Mattia Pascal;Luigi Pirandello;1904

Ultime lettere di Jacopo Ortis,Ugo Foscolo;1802

Se questo è un uomo;Primo Levi;1947

Il barone rampante;Italo Calvino;1957

I Malavoglia;Giovanni Verga;1881

Il Principe; Niccolò Macchiavelli;1532

La Gerusalemme Liberata;Torquato Tasso;1581

Cuore;Edmondo De Amicis;1886

Il deserto dei Tartari;Dino Buzzati;1940

La coscienza di Zeno;Italo Svevo;1923

Pinocchio;Carlo Collodi;1883

Il piacere;Gabriele D'Annunzio;1889

Myricae;Giovanni Pascoli;1891

Canti;Giacomo Leopardi;1831

Uno, nessuno e centomila;Luigi Pirandello;1926

Il nome della rosa;Umberto Eco;1980

La casa in collina;Cesare Pavese;1948

Il gattopardo;Giuseppe Tomasi di Lampedusa;1958

I Vicerè;Federico De Roberto;1894

Mastro don Gesualdo;Giovanni Verga;1889

# Classi

Per risolvere gli esercizi riguardanti i seguenti argomenti: streams e classi, è necessario in molti casi conoscere entrambi gli argomenti, con l’eccezione dei primi esercizi di base che ti permetteranno di imparare un argomento senza ancora aver studiato l’altro.

Per imparare nello specifico le classi si possono riproporre tutti gli esercizi fatti per le struct, aggiungendo tutti gli aspetti della programmazione ad oggetti. In particolare è opportuno:

1. valutare se rendere o meno privati gli attributi della classe efornire eventualmente metodi get e set
2. Definire i costruttori.
3. Definire una funzione toString (richiede l’utilizzo degli stringstream).
4. Definisci una classe rettangolo che ha come attributi pubblici le misure di base e altezza. Definisci anche gli opportuni costruttori e le funzioni per calcolare area e perimetro del rettangolo
5. Definisci una classe punto che abbia come attributi le coordinate reali nel piano. Oltre a fornire gli opportuni costruttori, fornisci e seguenti metodi:

* Funzione toString per fornire una rappresentazione del punto (es. A(1,2))
* Funzione per calcolare il punto medio
* Funzione per calcolare la retta passante tra due punti. Dovrai anche definire la classe retta
* Funzione che verifica se tre punti sono allineati. Questa funzione deve essere di tipo statico o essere definita esternamente alla classe (L’ideale è realizzare entrambe le versioni come esercizio)

1. Definisci una classe rettangolo che sia definita come insieme di 4 punti. Il costruttore dovrà verificare se i 4 punti formano un rettangolo per poterli accettare se no costruisce il rettangolo di default che è un quadrato degenere di lato 0 con tutti i vertici nell’origine. Per poter preservare il rettangolo rendi privati i punti del rettangolo e fornisci tutti i metodi get e set. Dovrai fornire anche la funzione toString oltre ai metodi per il calcolo di perimetro e area.
2. Definisci una classe frazione che offra possibilità di istanziare frazioni e effettuare tutte le operazioni matematiche fondamentali con le frazioni. Le operazioni dovranno essere accessibili per mezzo di operatori oltre che per mezzo di funzioni normali. Dovranno essere fornite tutte le funzionalità fondamentali: costruttori, toString, semplificazione della frazione funzionale alle altre operazioni e al costruttore.
3. Scrivi un programma in grado di fare operazioni con vettori nel piano. Dovrai definire la classe vettore e per essa gli operatori per effettuare somme e differenze di vettori. Aggiungi tutte le funzionalità necessarie (costruttori, toString, gestione accessibilità…).
4. Scrivi un programma in cui siano definite una serie di classi:

* Classe locomotiva che ha come attributi il peso, la forza con cui riesce a spingere, la forza con cui i freni riescono a frenare.
* Classe vagone che ha come attributi i posti disponibili il peso del vagone vuoto (magari in futuro gestiremo anche i passeggeri), la forza dei freni, il numero di passeggeri che trasporta e il peso totale compresi i passeggeri (considera che i passeggeri pesano sempre 65kg).
* Classe treno che deve contenere una locomotiva, un vettore di vagoni e la velocità corrente del treno (e volendo anche il peso corrente del treno in modo da non doverlo calcolare da 0 ogni volta che serve, ma solo ogni volta che un passeggero sale o scende dal treno).

Aggiungi alle classi, oltre agli opportuni costruttori, i metodi (funzioni della classe) necessari a svolgere le seguenti azioni:

* aggiungere o rimuovere un vagone dal treno (magari fai in modo che questo si possa fare solo quando il treno è fermo)
* far accelerare il treno (ogni volta che la funzione per accelerare viene chiamata devi calcolare la variazione di velocità secondo la relazione F=m\*a, considerando uguali massa e peso) passando come parametro il tempo per cui si vuole accelerare. (Siccome nel nostro mondo immaginario gli attriti (tranne quelli dei freni) non esistono e la relatività di Einstein neppure, il nostro treno può raggiungere qualsiasi velocità)
* far frenare il treno con le stesse regole dell'accelerazione considerando però che devi sommare le forze di frenata di locomotiva e vagoni. Quando freni ovviamente non puoi ottenere una velocità negativa.
* Far salire o scendere un passeggero da un determinato vagone, quando è possibile (treno fermo e posti disponibili o passeggeri finiti).

Scegli opportunamente se rendere pubblici o privati determinati attributi e metodi.

Il programma deve creare un treno, farlo muovere accelerando e frenando, e far salire e scendere passeggeri, stampando lo stato del treno ogni volta che succede qualcosa. (crea le opportune funzioni toString).

1. Scrivi un programma che gestisca le informazioni riguardanti la stagione di Formula 1. Le strutture dati fondamentali riguardano:

* Piloti di cui sappiamo nome e cognome e la scuderia di appartenenza
* Scuderia di cui sappiamo il nome e l’elenco
* Autodromo di cui sappiamo il nome, la città e la nazione di appartenenza.
* Gara di cui sappiamo in che autodromo si svolge, la data e l’ordine di arrivo dei piloti (che consiste in un vettore di piloti).
* Stagione di cui sappiamo l’anno e che contiene la classifica dei piloti e la classifica delle scuderie

Il programma deve essere in grado di fornire all’utente le stampe delle classifiche.

Se hai già studiato la lettura e scrittura da file, il programma deve leggere da file separati gli elenchi dei piloti e delle scuderie e i risultati delle gare. Per ogni gara il programma legge un diverso file che contiene tutti i dati della gara. Il programma deve poi scrivere su due file separati le classifiche piloti e costruttori (scuderie).

1. Leggi da due file le informazioni riguardanti un campionato di calcio. Il programma deve essere in grado di stampare la classifica del campionato sia su schermo che su file. Per raggiungere l’obiettivo devi definire le opportune strutture dati e gestire correttamente input e output da file.

# Strutture dati dinamiche in C++

## Liste, pile e code nella libreria standard

1. Carica da un file di testo, il cui contenuto è riportato di seguito, l'elenco dei voti degli studenti identificati dal loro cognome e i voti ad essi associati. Questi dati devono essere gestiti con una mappa.  
   Le funzionalità da implementare sono:

* Stampa tutto l'elenco dei voti
* Stampa il voto di uno specifico studente dato il suo cognome
* Stampa tutti i nomi degli studenti con voti sufficienti.
* Stampa la media dei voti della classe.
* Crea una nuova mappa che associa ad ogni voto dato il numero degli studenti che ha quel voto.

Contenuto del file di testo:

Bergonzi 3

Bna 7

Borghesan 4

Carrettiero 7

Casanova 6

Compagnoni 6

DiValentin 6

Faustini 5

Grigoli 5

Lintas 6

Manzotti 5

Mazzoleni 4

Monzani 6

Origo 6

Pagano 3

Perani 7

Perego 4

Puertas 3

Russo 6

Sala 6

Simone 7

Stucchi 7

Tuscano 3

Wu 8

Zaninello 8

1. Carica da un file di testo, il cui contenuto è riportato di seguito, l'elenco dei voti degli studenti identificati dal loro cognome e i voti ad essi associati. Questi dati devono essere gestiti con una mappa.   
   Le funzionalità da implementare sono:

* Stampa tutto l'elenco dei voti
* Stampa il voto di uno specifico studente dato il suo cognome
* Stampa tutti i nomi degli studenti con voti sufficienti.
* Stampa la media dei voti della classe.
* Crea una nuova mappa che associa ad ogni voto dato il numero degli studenti che ha quel voto.

Bergonzi 3 6 5 7

Bna 7 6 6 7

Borghesan 4 5 6

Carrettiero 7 6 6 7 5

Casanova 6 6 8 6

Compagnoni 6 7 7

DiValentin 6 5 6 6 6

Faustini 5 5 5 6 5

Grigoli 5 6

Lintas 6

Manzotti 5 5 6

Mazzoleni 4 5 6

Monzani 6 6

Origo 6 6

Pagano 3 3 4

Perani 7 6 7 6

Perego 4 3 5 6 6

Puertas 3 4 7 5

Russo 6 6

Sala 6 5 7

Simone 7 7

Stucchi 7

Tuscano 3 5 5

Wu 8 7 9

Zaninello 8 7

1. Scrivi un programma che carichi in una mappa i dati contenuti nel file input.txt che riguardano la classifica della stagione di F1 2019. Il programma deve poi riuscire a stampare la classifica non in ordine alfabetico per scuderia ma in ordine di punti fatti, in pratica deve stampare la classifica finale. Puoi usare il metodo che preferisci.   
   Il metodo che ti consiglio di provare, richiede l'utilizzo di una struttura dati che non abbiamo visto espressamente insieme ma che è piuttosto semplice e che indirettamente hai già usato: pair<string, int>. Questo tipo di dato è il singolo elemento di una mappa come map<string, int>. Potresti creare un vector di pair: vector<pair<string, int> e poi utilizzare opportunamente la funzione sort per ordinarlo come desideri. per farlo dovrai anche scrivere la funzione di confronto tra due pair. Purtroppo tutto questo richiede che tu ricordi qualcosa sui vettori e su come si ordinano, cosa richiesta nel progetto per pasqua.   
   Se trovi difficoltà a leggere i dati dal file puoi semplificare la cosa modificando il file, anche se sarebbe bene che imparassi a leggere come si deve i file.   
   Contenuto del file input.txt:

Mercedes, 612

Ferrari, 435

Red Bull, 323

McLaren, 111

Renault, 68

Toro Rosso, 62

Racing Point, 58

Alfa Romeo, 35

Haas, 28

Williams, 1

1. Scrivi un programma che utilizzi le tre nuove strutture dati viste. Crea e riempi una lista di numeri casuali. Ordina poi la lista e stampala. Crea quindi una coda e una pila che devono essere riempiti con i numeri della lista. Quando inserisci i valori, fai in modo che alla fine i numeri vengano stampati nello stesso ordine della lista ordinata.
2. Scrivere la funzione string inverti(string s) che prende in input una stringa s e restituisce in output la stringa ottenuta invertendo s. La funzione deve usare come unica struttura dati ausiliaria uno stack.
3. Scrivere la funzione bool checkParentheses(string s). La funzione prende in input una stringa che rappresenta un’espressione contenente sia parentesi tonde che parentesi quadre e restituisce true se e solo se le parentesi sono correttamente annidate.
4. Scrivere la funzione public bool isPalindrome(string s). La funzione prende in input una stringa e restituisce true se e solo se la stringa è palindroma. Ad esempio, la funzione restituisce true se riceve in input la stringa “anna” o la stringa “abcecba” e restituisce false se riceve in input la stringa “abc” o la stringa “abcdba”.
5. Scrivere la funzione int extract(queue<int> Q, int k) che restituisce in output l’elemento di Q che si trova in k-esima posizione a partire dal front. Il metodo non deve utilizzare strutture dati ausiliare e deve lasciare inalterato il contenuto di Q. Se Q contiene meno di k elementi, allora la funzione deve stampare un messaggio di errore e restituire 0.
6. Scrivere la funzione void reverse(Queue<int> Q) che inverte il contenuto della coda Q.