Basi di programmazione in C
Eserciziario

Rel. 1.7

7.10.2020

Luigi Ferrari

Indice

Licenza

1. Prerequisiti e flowchart

1. **Scelte e iterazioni:** Rappresenta graficamente con schemi di iterazione o di selezione le seguenti frasi

```
1) Se fa caldo
      allora apri la finestra
2) Finché fa caldo
      lascia la finestra aperta
3) Finché ci sono fogli nel cassetto
      prendi un foglio
4) Se ci sono quaderni nel cassetto
      allora prendi un quaderno
      altrimenti vai a comprarne uno
5) Ripeti
      lava un piatto
   finché i piatti non sono tutti puliti
6) Finché c'è un piatto sporco
      lava un piatto
7) Finché hai un voto insufficiente
      studia per recuperare
8) Ripeti
      studia di più
   finché non prendi un voto sufficiente
9) Finché non hai capito cosa è l'iterazione
      fai un esercizio sull'iterazione
10) Se ho capito la spiegazione
       riordino gli appunti
       altrimenti chiedo chiarimenti
11) Se c'è spazio nella scheda di memoria del cellulare
       finché non finisce
          scatto fotografie
       altrimenti compro delle cartoline
```

2. **Diagrammi di flusso:** Rappresentare con un flow-chart i seguenti procedimenti.

   1) Travasare il vino contenuto in una damigiana da 20 litri in bottiglie da 3/4.

   2) Calcolare il numero di fazzoletti 40x40 cm ricavabili da una stoffa di dimensioni date.

   3) Eseguire la somma tra due numeri di due cifre ciascuno.

   4) Eseguire la differenza tra due numeri di due cifre ciascuno.

   5) Eseguire la moltiplicazione tra due numeri di due cifre ciascuno.

   6) Eseguire la divisione tra due numeri di due cifre ciascuno.

   7) Calcolare la lunghezza dell'ipotenusa di un triangolo rettangolo di cui si conoscono i cateti.

   8) Prepararsi e fare una verifica di elettronica.

   9) Fare il flow chart di un procedimento.

   10) Vincere al totocalcio.

3. **Orario:** Descrivere mediante un diagramma di flusso un programma in grado di calcolare quanti secondi sono passati tra la mezzanotte e un'ora indicata mediante ora (intero), minuti (intero), secondi (intero) e un numero che rappresenta l'essere di mattino o di pomeriggio (0 per il mattino e 1 per il pomeriggio).

4. **Cedolino:** Descrivere mediante un diagramma di flusso un programma in grado di stampare il ruolino di paga settimanale di un lavoratore inglese.

   Allo stipendio lordo va dedotto il 5% per la liquidazione, il 25% di cio' che rimane come imposta sul reddito e una deduzione fissa di 14.50 sterline alla settimana per l'assistenza medica.

   Nel cedolino vanno indicati il lordo, il netto e tutte le deduzioni previste.

5. **Anni bisestili:** Descrivere mediante un diagramma di flusso un programma in grado di calcolare se un certo anno e' bisestile o no. Un anno è bisestile se il suo numero è divisibile per 4, con l'eccezione degli anni secolari (quelli divisibili per 100) che non sono divisibili per 400.

   Suggerimento: utilizzare il resto della divisione intera.

6. **Somma di ore:** Descrivere mediante un diagramma di flusso un programma in grado di fare la somma tra due ore, presentate nella forma hh:mm:ss.

   Il programma deve tener conto che se si superano le ore 23:59:59 si deve ripartire da 00:00:00.

7. **Rappresentazione dei numeri nel sistema binario:** Copiare sul quaderno e completare:

```
Il più grande numero intero positivo rappresentabile con

4  cifre binarie è  ............    10 cifre binarie è  ............

16 cifre binarie è  ............    20 cifre binarie è  ............

Convertire in decimale 1 0 1 1 0 0 1 0   ..............

Convertire in binario  120               ..............

convertire da decimale a binario (compl. a 2, 8 bit) i numeri
   -20     ............
   -100    ............

eseguire la moltiplicazione dei due numeri binari    1 1 1 0 0 x
                                                         1 1 0
                                                    -----------
```

8. **Numerazione esadecimale:** Tradurre in binario e in esadecimale, sul quaderno, i seguenti valori interi espressi in base decimale: 13, 56 208, 120, -4, -1, -100. Utilizzare 8 bit per ogni valore.

9. **Numerazione esadecimale:** Trasformare, sul quaderno, in binario e in decimale i seguenti numeri esadecimali che rappresentano valori interi con 8 bit: 30, 4A, D9, FF

10. **Calcolo del resto:** Scrivere il flowchart e fare la trace-table di almeno due casi significativi del seguente algoritmo:

```
1. prendi due numeri A e B
2. se A non è maggiore di B scambiali
3. togli B da A
4. se A è positivo torna a 3.
5. se A è diverso da zero aggiungigli B
```

```
6. stampa A
```

11. **Potenza:** Scrivere il flowchart e fare la trace-table di almeno due casi significativi del seguente algoritmo:

```
1. prendi due numeri A e B
2. assegna a C il valore 1
3. assegna a C il valore di C per A
4. diminuisci B di una unità
4. se B è diverso da zero torna a 3.
6. stampa C
```

Proporre eventuali miglioramenti o correzioni. Suggerimento: cosa succede se l'esponente è zero o se è un numero negativo?

12. **Radice quadrata:** Scrivere il flowchart e fare la trace-table di almeno due casi significativi del seguente algoritmo:

```
 1. prendi un numero N
 2. assegna a R il valore 1
 3. assegna a X il valore 1
 4. calcola D come differenza tra R*R e N
 5. se D/N e' inferiore a 0,01 vai a 12.
 6. se D e' maggiore di zero
 7. togli X a R
 8. vai a 10.
 9. assegna a X il valore di X/10
10. aggiungi X a R
11. vai a 3.
12. stampa R
```

2. Basi linguaggio C

Tipi di dato

1. **Tipi di dato:** Utilizzando le risorse a disposizione, completare la tabella (prof. Mario Janin, 1996).

Numeri interi

interi	char	int	long	unsigned	unsigned long
bytes occupati					
valore max					
valore min					
specificatore di formato					

Numeri reali

virgola mobile	float	double	long double
bytes occupati			
valore max			
valore min			
specificatore di formato			

2. **Dichiarazione e assegnazione:** Scrivere un piccolo programma che dichiara delle variabili: una intera, una reale, una di tipo carattere. Stamparne il valore di ognuna (una per riga). Il programma assegna POI ad ognuna di loro un valore e ripete la stampa. Infine, aggiunge 2 ad ognuna delle tre variabili e ne stampa di nuovo il valore. Fare delle osservazioni riguardo a cosa si ottiene.

3. **Conversioni implicite:** Dicharare tre variabili: i di tipo intero, r di tipo reale e c di tipo carattere. Verificare quali assegnazioni di una variabile all'altra sono possibili (ad esempio: i = r; oppure r = c;). Indicare in quali casi il programma viene compilato serza errori e qualisono invece le assegnazioni vietate. Provare poi a assegnare dei valori alle

variabili, a fare le assegnazioni consentite e verificare cosa contengono le varibili dopo le assegnazioni.

Utilizzare printf() con gli opportuni specificatori di formato.

Input e output

4. **Disegno:** Scrivere un programma che stampi a video la seguente figura:

```
****
*
***
*
****
```

5. **Clip Art:** Disegnare una casetta utilzzando le lettere.

6. **Somma:** Si scriva un programma in linguaggio C che legga due valori interi e visualizzi la loro somma. Ripetere con l'uso di numeri reali.
7. **predecessore e successore:** Si scriva un programma in linguaggio C che legga un valore intero e visualizzi il valore intero precedente e il successivo.
8. **Aree:** Si scriva un programma in linguaggio C che, dato un numero reale D immesso da tastiera, calcoli e stampi:
   1. l'area del quadrato di lato D.
   2. l'area del cerchio di diametro D.
   3. l'area del triangolo equilatero di lato D.
9. **Far di conto:** Si scriva un programma in linguaggio C capace di compiere le 4 operazioni (somma, sottrazione, moltiplicazione e divisione) tra due numeri reali inseriti da tastiera. Dopo che sono stati inseriti i due numeri, detti a e b, il programma dovrà visualizzare i quattro valori a+b, a-b, a*b, a/b. Si ipotizzi che b sia sempre diverso da 0.
10. **Cartellino del prezzo:** Scrivere un programma che richieda all'utente il prezzo di un oggetto e la percentuale di sconto ad esso applicata. Con i dati letti mostra a video il cartellino con il prezzo (originale e scontato).

    Esempio:

```
Inserisci i dati:
Prezzo: 25
Sconto: 10
```

```
Cartellino:
              Prezzo:  25.00 euro
Sconto applicato 10%:   2.50 euro
       Prezzo finale:  22.50 euro
```

Scelte e alternative

NOTA Questi esercizi qui riportati non fanno uso delle funzioni.

11. **Valore assoluto:** Si realizzi un programma in linguaggio C che acquisisca da tastiera un numero e stampi il valore assoluto di tale numero.
12. **Analisi:** Si scriva un programma in linguaggio C che legga due numeri da tastiera, detti a e b, e determini le seguenti informazioni, stampandole a video:

    1. determini se b è un numero positivo o negativo.
    2. determini se a è un numero pari o dispari.
    3. calcoli il valore di a + b.
    4. determini quale scelta tra le operazioni +, -, * e / applicate ad a e b porta al risultato massimo, e quale è questo valore massimo.

13. **Equazione di primo grado:** Data l'equazione:

    $$ax + b = 0$$

    con a e b inseriti da tastiera, scrivere un programma in linguaggio C per determinare il valore di x, se esiste, che risolve l'equazione.
14. **Mesi:** Dato un numero intero tra 1 e 12, che rappresenta il mese corrente, stampare il nome del mese per esteso ("Gennaio" ... "Dicembre").
15. **Semplice calcolatrice:** Scrivere un programma che legga da tastiera 2 numeri reali, da intendersi come operandi. Il programma poi legge un carattere che specifica l'operazione da effettuare. ATTENZIONE: per la lettura dell'operazione utilizzare, ad esempio, la funzione getch(), che restituisce un solo carattere, ma prima di essa scrivere

    ```
    fflush(stdin);
    ```

    per evitare strani malfunzionamenti.

    In particolare, se l'operazione vale '+', allora occorre calcolare e stampare la somma dei due operandi. Se l'operazione vale '-', occorre calcolare e stampare la differenza dei due operandi. In tutti gli altri casi stampare un messaggio di errore.

    Esempio

    ```
    Operando 1: 7
    Operando 2: 9
    Operazione: +
    La somma vale 16
    Operando 1: 7
    Operando 2: 9
    Operazione: -
    La differenza vale -2
    Operando 1: 7
    Operando 2: 9
    ```

```
Operazione: a
Operazione non valida
```

16. **Diversi tipi di incremento:** Provare a prevedere l'output di questo programma e poi verificarlo compilandolo ed eseguendolo. Spiegare il motivo dei risultati ottenuti.

```
 int main(){
   int a, b;
   a = 5;
   b = a+1;
   printf("1. a=%3d b=%3d\n", a, b);
   b = a++;
   printf("2. a=%3d b=%3d\n", a, b);
   b= ++a;
   printf("3. a=%3d b=%3d\n", a, b);
   return 0;
}
```

17. **Concerto:** Il servizio di rivendita di biglietti percepisce una provvigione sul prezzo del biglietto. La provvigione è pari al 15% del prezzo del biglietto, ma in ogni caso è pari ad almeno 5 Euro.

    Scrivere un programma che, dato il prezzo di un biglietto, calcoli e stampi:

    1. il valore della provvigione.
    2. il prezzo finale del biglietto.

18. **Tipo di frazione:** Scrivere un programma che data una frazione dica se è propria, impropria o apparente. Le frazioni PROPRIE sono quelle nelle quali il NUMERATORE è MINORE rispetto al DENOMINATORE. Le frazioni IMPROPRIE sono quelle nelle quali il NUMERATORE è MAGGIORE rispetto al DENOMINATORE. Le frazioni APPARENTI sono quelle nelle quali il NUMERATORE è UGUALE o MULTIPLO rispetto al DENOMINATORE.

19. **Re e Regina:** Su una scacchiera 8x8 sono posizionati due pezzi: il Re bianco e la Regina nera. Si scriva un programma in linguaggio C che, acquisite le posizioni del Re e della Regina, determini se la Regina è in posizione tale da poter mangiare il Re. Le posizioni dei due pezzi sono identificate da mediante la riga e la colonna su cui si trovano, espresse come numeri interi tra 1 e 8.

Cicli

NOTA Questi esercizi qui riportati non fanno uso delle funzioni.

20. **Tabella di esecuzione:** Eseguire a mano, passo dopo passo, il seguente frammento di codice, mostrando i valori assunti da n e le informazioni visualizzate in una tabella di esecuzione (o trace table). Indicare alla fine cosa compare esattamente sul terminale.

```
int n;
n = 5;
while (n>= 0) {
   n=n-1;
   printf("%d", n);
}
```

21. **Tabella di esecuzione:** Eseguire a mano, passo dopo passo, il seguente frammento di codice, mostrando i valori assunti da n e le informazioni visualizzate in una tabella di esecuzione (o trace table). Indicare alla fine cosa compare esattamente sul terminale.

```
int n;
n = 2;
while (n <= 5) {
   printf("%d, ", n);
  n = n+1;  // alternativa: n++;
}
```

22. **Tabella di esecuzione:** Eseguire a mano, passo dopo passo, il seguente frammento di codice, mostrando i valori assunti da r e di i in una tabella di esecuzione (o trace table). Supporre che a valga 2 e n valga 4.

```
int i, r;
i = 1;
r = 1;
while (i <= n) {
   r = r * a;
  i = i + 1;    // alternativa: i++;
}
```

Successivamente indicare quale potrebbe essere lo scopo dell'algoritmo nel caso di a e n qualunque.

23. **Tabella di esecuzione:** Eseguire a mano, passo dopo passo, il seguente frammento di codice. Che errore si riscontra?

```
int n;
n = 1;
while (n != 50) {
   printf("%d", n);
  n = n + 10;
}
```

24. **Correzione e traduzione:** Il seguente algoritmo, espresso in pseudocodice, dovrebbe contare il numero di cifre presenti in un numero n:

```
contatore = 1
temp = n
while (temp > 10)
  incrementa contatore
 dividi temp per 10.0
mostra il valore di contatore
```

Verificarne il funzionamento con una tabella di esecuzione per i casi n = 123 e n = 100. Quale errore viene trovato? Come lo si può correggere? Scrivere il corrispondente codice in linguaggio C.

25. **Indentazione ed esecuzione:** Scrivere le tabelle di esecuzione per i seguenti frammenti di codice, supponendo tutte le variabili intere, dopo avere riscritto il ciclo con la corretta indentazione:

a) i=0; j=9; n=0; while (i<j) { i++; j--; n++; }

b) i=1; j=0; n=0; while (i<10) { i++; n= n+i+j; j++; }

c) i=2; j=10; n=1; while (i!=j) { i=i+2; j=j-2; n=n+2; }

d) i=6; j=0; n=0; while (i>0) { i--; n=n+i-j; }

26. **Somma di N valori:** Si scriva un programma in linguaggio C per calcolare la somma di un insieme di N numeri inseriti da tastiera. Il programma deve leggere inizialmente il valore di N. In seguito il programma legge gli N numeri e infine ne visualizza la somma.

27. **Max di un numero indefinito di valori:** Si scriva un programma in linguaggio C per calcolare la somma di un insieme di numeri inseriti da tastiera. Il programma deve leggere una sequenza di numeri, fermandosi non appena viene inserito il numero 0; a questo punto visualizza il valore corrispondente al massimo dei valori introdotti.

28. **Conteggio dei valori inseriti:** Un utente inserisce da tastiera una serie di numeri interi positivi, ed il termine della serie è indicato dall'inserimento del valore -1. Il programma, al termine dell'inserimento, deve stampare quanti numeri pari l'utente ha inserito, e quanti numeri in totale sono stati inseriti.

29. **Terminazione al compiersi di una condizione:** Si scriva un programma in linguaggio C che legga da tastiera una serie di numeri interi fino a quando la somma di tutti i numeri introdotti fino a quel momento non supera il valore 1000. A quel punto, il programma stampa quanti numeri sono stati inseriti.

30. **Disequazione e potenza:** Dato un numero reale positivo y immesso da tastiera, si scriva un programma in linguaggio C che determini qual è il massimo numero intero positivo x tale per cui sia valida la relazione " x elevato alla x è minore o uguale a y".

31. **Fattoriale:** Si scriva un programma in linguaggio C che acquisisca un numero intero positivo N da tastiera e stampi il valore del fattoriale di N.

    Suggerimento. Si ricorda che il fattoriale di un numero è il prodotto di tutti i numeri compresi tra 1 ed N.

    $$N! = 1 * 2 * 3 * ... * (N-1) * N$$

    Inoltre 0! = 1.

32. **Sequenza:** Si scriva un programma in linguaggio C per poter analizzare una sequenza di numeri: il programma stampa se la sequenza dei valori inseriti è strettamente crescente, strettamente decrescente oppure ne' strettamente crescente ne' strettamente decrescente.

33. **Quadrato di asterischi:** Si realizzi un programma che legga da tastiera un valore intero n, compreso tra 1 e 10, e stampi a video un "quadrato di asterischi" di lato n.

    Esempio:

```
Inserisci n: 5
*****
*****
*****
*****
*****
```

34. **Quadrato vuoto di asterischi:** Si realizzi un programma che legga da tastiera un valore intero n, compreso tra 1 e 10, e stampi a video un "quadrato vuoto di asterischi" di lato n.

    Esempio:

```
Inserisci n: 5
*****
*   *
*   *
*   *
*****
```

35. **Quadrati perfetti:** Scrivere un programma che visualizzi tutti i numeri interi minori di n che sono quadrati perfetti (ad esempio, se n vale 100 visualizza 0, 1, 4, 9, 16, 25, 36, 49, 64, 81).
36. **Potenze di 2:** Scrivere un programma che visualizza tutti i numeri interi che sono potenze di 2 inferiori ad un numero dato (ad esempio se viene fornito in input 50 mostra 1, 2, 4, 8, 16, 32)
37. **Somma delle cifre dispari di un numero **:** Scrivere un programma che somma tutte le cifre dispari di un numero n fornito (ad esempio se il numero è 14391 la somma è 1+3+9+1=14).
38. **Triangolo di Floyd:** Scrivere un programma in linguaggio C per la rappresentazione del triangolo di Floyd. Il triangolo di Floyd è un triangolo rettangolo che contiene numeri naturali, definito riempiendo le righe del triangolo con numeri consecutivi e partendo da 1 nell'angolo in alto a sinistra.

    Si consideri ad esempio il caso N=5. Il triangolo di Floyd e' il seguente:

```
1
2 3
4 5 6
7 8 9 10
11 12 13 14 15
```

    Il programma riceve da tastiera un numero intero n. Il programma visualizza le prime n righe del triangolo di Floyd.

    Suggerimento. Si osserva che il numero di valori in ogni riga corrisponde all'indice della riga: 1 valore sulla prima riga, 2 sulla seconda, 3 sulla terza.

3. Funzioni

ATTENZIONE Per ogni funzione che viene richiesta devono essere eseguiti i seguenti passi, che comprendono l'analisi, la codifica e il test.

1. la scelta del nome
2. la comprensione di quanto viene richiesto
3. l'identificazione del numero, tipo e significato di ognuno dei parametri in ingresso (se presenti)
4. l'dentificazione del tipo e del significato del valore di ritorno (se presente)
5. la scrittura di un commento con i quattro punti precedenti
6. la dichiarazione con il prototipo
7. la definizione della funzione (il codice)
8. il main di prova della funzione (se possibile automatico, cioè che non richiede input all'utente)

Esercizi su analisi, sintassi e prototipi

1. **Sommatore banale:** Somma di due numeri reali
2. **Sconto:** Calcolo prezzo scontato di un articolo

3. **Pitagora:** Calcolo area di un triangolo rettangolo
4. **Area di un triangolo:** Calcolo area di un triangolo dati i lati (ma è davvero un triangolo?)
5. **Cambiavalute:** Conversione di un importo in Euro nel corrispondente in dollari
6. **Trading:** Le azioni dell'azienda Peia passano da x Euro a y Euro nel corso di un anno. Calcolare il guadagno o la perdita percentuale.
7. **Sprint:** Dato il tempo sui 100 metri di un atleta calcolare la velocità media in km/h.

8. **BitOn:** Dato un numero intero positivo trovare il numero di bit a 1 di cui è composto.
9. **Numero di cifre:** Dato un numero intero positivo trovare il numero di cifre di cui è composto (ad es. 244 ha 3 cifre).

10. **Codifica dei caratteri:** Dato un carattere mostrare sul video il carattere stesso e il suo codice ASCII in tre formati: in decimale, in ottale ed in esadecimale.
11. **Velocità media:** Data una distanza e il tempo impiegato per percorrerla, trovare la velocità media.
12. **Piastrellista:** Realizzare una funzione che date le misure del pavimento di una stanza rettangolare, la dimensione delle piastrelle e il numero di piastrelle di una scatola trova il numero di scatole di piastrelle necessario per coprirlo tutto.

Algoritmi sequenziali, input ed output

13. **On the road:** Un ragazzo fa un viaggio in moto, partendo con il serbatoio pieno. Durante il viaggio si ferma più volte a fare benzina e ogni volta mette nel serbatoio 8 litri. Alla fine del viaggio rifà il pieno (in modo da avere il serbatoio come alla partenza) e prende nota dei chilometri percorsi (il contachilometri non riporta le frazioni).

    Scrivere una funzione che calcoli il consumo medio di carburante espresso in chilometri per ogni litro.

Immagine tratta dal film "I diari della motocicletta"

14. **Preventivo:** Scrivere un programma che consenta ad una impresa edile di fare dei preventivi di spesa sui lavori. I dati che devono essere forniti al programma sono il costo del materiale, il numero di giorni di lavoro previsti e il numero di operai da dedicarvi.

    Una giornata di un operaio costa 120 Euro. Al totale deve essere aggiunta una quota pari al 4% di assicurazione. Sul totale complessivo deve essere calcolata l'IVA al 20%.

    Alla fine deve essere visualizzato un prospetto il più possibile simile al seguente:

```
Impresa Massacan
Via Muratori 99
16100 Genova
```

```
Preventivo

Costo materiali                              1500,00
Mano d'opera 3 operai per 10 giorni          3600,00
                                           ---------
                              SubTotale      5100,00 Euro
Assicurazione (4%)                            204,00 Euro
                                           ---------
                              SubTotale      5304,00 Euro
IVA (20%)                                    1060,80 Euro
                                           ---------
                                 Totale      6464,80 Euro
```

Passaggio di argomenti modificabili (passaggio di puntatori)

15. **Incremento:** Scrivere una funzione che riceve l'indirizzo di una variabile intera e ne incrementa il valore contenuto.
16. **Scambio:** Scrivere una funzione che scambia il contenuto di due variabili reali.
17. **Indirizzo:** Scrivere una funzione che visualizza l'indirizzo e il valore contenuto in una variabile reale.
18. **A caso:** Scrivere una funzione che riceve l'indirizzo di una variabile intera x e un valore intero n; la funzione modifica il valore di x aggiungendogli n e restituisce il valore della differenza tra x e n.
19. **Velocità media:** Data una distanza e il tempo impiegato per percorrerla, trovare la velocità media in km/h e in m/s.

Variabili statiche e variabili globali

20. **Protocollo:** Un timbro del protocollo deve fornire ad ogni utilizzo un numero appartenente ad una sequenza crescente. Realizzare una funzione int timbra() che ad ogni chiamata restituisce un numero ogni volta diverso (la prima volta 1, la seconda 2...). Utilizzare una variabile statica.
21. **Condivisione:** Dichiarare una variabile intera globale e verificare che due funzioni possono accedervi. In particolare una la incrementa, una seconda la stampa e una terza la azzera.
22. **Condivisione tra file diversi:** Realizzare un costituito da più moduli, ognuno dei quali può accedere as una variabile globale errorCode intera. Questa variabile vale 0 se non ci sono errori, un codice di errore se è stato riscontrato un errore (ad esempio un tentativo di divisione per zero). Suggerimento: un modulo contiene e il main con la dichiarazione della variabile, uno la funzione dividi che può provocare errore, un terzo una funzione che a seconda del valore del codice di errore ne visualizza la descrizione estesa.
23. **Variabili statiche:** Verificare con un programma che una variabile globale dichiarata static è visibile da tutte le funzioni del file dove è dichiarata ma NON dalle funzioni di altri files.

4. Controllo di flusso

ATTENZIONE Per ogni funzione che viene richiesta devono essere eseguiti i seguenti passi, che comprendono l'analisi, la codifica e il test.

1. la scelta del nome
2. la comprensione di quanto viene richiesto
3. l'identificazione del numero, tipo e significato di ognuno dei parametri in ingresso (se presenti)
4. l'dentificazione del tipo e del significato del valore di ritorno (se presente)
5. la scrittura di un commento con i quattro punti precedenti
6. la dichiarazione con il prototipo
7. la definizione della funzione (il codice)
8. il main di prova della funzione (se possibile automatico, cioè che non richiede input all'utente)

4.1. Test

1. **Massimo:** Dati tre numeri restituire il massimo.
2. **Visualizzazione ordinata:** Dati tre numeri visualizzarli in ordine crescente.
3. **Ordinamento:** Dati tre numeri scabiarne il contenuto in modo che il primo contenga in valore minore, il secondo quello intermedio e l'ultimo il valore massimo.
4. **Controllo input:** Dati un valore minimo e un valore massimo chiedere un numero reale in input all'utente e verficare che sia un numero compreso tra i due valori.
5. **Somma di tempi:** Mostra a video, nel formato mostrato nell'esempio seguente, la somma di due ore espresse in ore, minuti e secondi.

   Esempio

   ```
    h  m  s
   22 14 30 +
    3 50 27 =
    -------
    2 04 57
   ```

   Il programma deve tener conto che se si superano le ore 23:59:59 si deve ripartire da 0:00:00.
6. **Divisibilità:** Indicare se un numero intero e' divisibile per un altro.
7. **Stagione:** forniti il giorno e il mese determinare la stagione e stamparne a schermo il nome (primavera, estate, autunno, inverno)
8. **Interi distinti:** Dati tre numeri interi restituire quanti di essi sono distinti (ad esempio se i tre numeri sono 8, 3, 3 i numeri distinti sono 2).
9. **Tipo di carattere:** dato un carattere restituire 1 se è una cifra, 2 se è una vocale, 3 se è una consonante, 0 se non è alcuna di esse. Attenzione che le lettere possono essere maiuscole o minuscole.
10. **Valore assoluto:** Dato un numero reale restituire il valore assoluto del numero stesso.

11. **Triangolo rettangolo:** Dati tre numeri reali a, b, c verificare se a è l'ipotenusa di un triangolo con i cateti b e c; versione potenziata: l'ipotenusa può essere uno qualunque tra a, b, c.
12. **La superstrada:** Una superstrada è divisa in tronchi; ogni tronco corrisponde ad un casello di ingresso/uscita. La percorrenza del primo tratto costa un Euro, se si continua a percorrere la superstrada i successivi tronchi costano 0.50 Euro. Determinare il prezzo da pagare all'uscita noti il numero della stazione di ingresso (un numero intero da 1 a N) e quello della stazione di uscita. Attenzione: la superstrada può essere percorsa in entrambi i sensi. (Ad esempio, entrando al casello 6 e uscendo al casello 3 il prezzo da pagare è 2 Euro: 1 euro per il tratto dal 6 al 5, 50 centesimi per il tratto dal 5 al 4 e altri 50 per il tratto dal 4 al 3).
13. **Verifica orario:** Dato un orario costituito da tre numeri interi h, m, s verifica se si tratta di un orario valido.
14. **FizzBuzz:** si fa riferimento ad un gioco per bambini, un cui ogni bambino dice il numero successivo a quello detto dal bambino prima di lui (partendo da 1), ma sostituendo i numeri divisibili per 3 con "Fizz", quelli divisibili per 5 con "Buzz" e quelli divisibili per 3 e per 5 con "FizzBuzz". La versiione semplificata da realizzare riceve un numero intero (maggiore o uguale ad 1) e mostra sullo schermo le parole "Fizz", "Buzz", "FizzBuzz" o il numero stesso.
15. **Or esclusivo:** Date due variabili booleane intere (in C qualunque numero diverso da 0 è vero) determinare l'or esclusivo di esse, restituendo 1 o 0 a seconda dei casi.
16. **Parchimetro:** Un parchimetro di un comune rivierasco determina il prezzo in base alla durata della sosta: fino ad un'ora è di 1.50 Euro, per le altre tre ore successive (o frazione di esse) è di 1 euro ciascuna. Attenzione: se si superano le 4 ore si continua a 1 Euro l'ora ma il totale viene raddoppiato. Suggerimento: fare prima degli esempi con diverse durate (espresse in minuti) calcolando a mano l'importo. Usare gli esempi come "casi di prova" del programma.
17. **Somma di angoli:** sommare due angoli espressi in gradi, primi e secondi. Per angoli maggiori di 360 gradi occorre ricondursi a valori minori togliendo un opportuno multiplo di 360 gradi.
18. **Numero di soluzioni di equazione di secondo grado:** dati i termini a, b e c di una equazione di secondo grado calcolare il numero di soluzioni reali distinte.
19. **Soluzioni equazione di secondo grado:** dati i termini a, b, e c di una equazione di secondo grado calcolare le soluzioni reali e mostrarle a video.
20. **Vinaio:** Un vinaio propone uno sconto a chi compra una certa quantità minima di vino. Si visualizza quanto l'acquirente spende, in base alla quantità ed al prezzo non scontato di un litro di vino.
21. **Funzioni trigonometriche:** dato tipo di funzione (seno, coseno, tangente) e il valore dell'angolo in radianti, calcola se la funzione e' positiva in quel punto.

    Nota: utilizzare le funzioni matematiche del linguaggio C (da cercare sulle risorse a disposizione).
22. **Calcolo del resto:** A fronte di un pagamento con una banconota da 100 euro, assumendo che sia stato fatto un acquisto per un importo inferiore a 100 euro, calcolare il resto da

fornire usando il numero minimo di monete o banconote dei tagli esistenti. Mostrare a video il numero di monete o banconote da restituire per ciascun taglio.

4.2. Algoritmi iterativi (cicli)

1. **Triangolo:** Visualizzare su video un triangolo di N righe formate da asterischi. Esempio con N=4:

```
*
**
***
****
```

   La funzione restituisce il numero totale di asterischi stampati.

2. **Quadrato:** Visualizzare su video un quadrato di lato N tutto pieno di asterischi. Esempio con N=4:

```
****
****
****
****
```

3. **Quadrato vuoto:** Visualizzare su video un quadrato di lato N. vuoto, di asterischi. Esempio con N=4:

```
****
*  *
*  *
****
```

   Evoluzione: il quadrato deve essere centrato orizzontalmente nello schermo (80 colonne).

4. **No diagonale:** Visualizzare su video un quadrato di lato N tutto pieno di asterischi tranne che sulla diagonale principale. Esempio con N=10:

```
 *********
* ********
** *******
*** ******
**** *****
***** ****
****** ***
******* **
******** *
*********
```

5. **Albero di Natale:** Data l'altezza in numero di righe N disegnare sullo schermo un albero di Natale di altezza N. Scegliere liberamente il disegno, che deve comprendere il tronco, la chioma ed eventuali palline.

6. **Tabella dei codici ASCII:** Stampare a video la tabella dei codici ASCII, escludendo i primi 32 caratteri (codici ASCII da 0 a 31).

7. **Inserimento di una sequenza:** richiedere l'introduzione di più numeri interi fino a che non viene introdotto il valore 0. A quel punto deve restituire quanti numeri positivi sono stati introdotti.

8. **Sequenza crescente:** richiedere l'introduzione di più numeri interi fino a che non sono più in sequenza strettamente crescente. Restituire la somma dei valori crescenti introdotti.
9. **Calcolo della media:** Forniti dall'utente un numero N e N numeri reali calcolare la media
10. **Calcolo del Fattoriale:** Dato N, numero intero, calcolare il fattoriale (4! = 1*2*3*4 = 24)
11. **Somma dei pari:** dato un numero N determinare la somma di tutti i numeri pari compresi tra 1 e N (ad esempio: se N è 5 la somma è 2+4=6).
12. **Prodotto:** Dati due numeri interi calcolarne il prodotto mediante successive somme. Ottimizzare l'algoritmo.
13. **Multipli:** Dati tre numeri interi positivi contare quanti numeri compresi tra il primo e il secondo sono multipli del terzo valore.
14. **Sommatore:** calcola la somma dei numeri dispari compresi tra due valori passati in input. Attenzione: potrebbero non essere inseriti in ordine. Esempio: se i due numeri sono 12 e 15 restituisce 13+15=28.
15. **Somma delle cifre:** Dato un numero intero qualunque ricavare la somma delle cifre di cui è composto (ad esempio: con 708 restituisce 7+0+8 = 15)
16. **Somma con soli incrementi e decrementi:** Calcolare la somma di due numeri utilizzando solo operazioni di incremento (ad esempio n++;) e decremento (ad esempio num--;). Suggerimento: la somma dei due numero può essere ottenuta aggiungendo al primo numero tante unità quante se ne possono togliere al secondo).
17. **Quoziente della divisione intera:** Dati due interi N1 e N2 calcolare il quoziente della divisione senza usare gli operatori \ e % ma facendo la divisione per sottrazioni successive. Fare un insieme di test significativi.
18. **Resto della divisione tra interi:** Dati due interi N1 e N2 calcolare il resto della divisione senza usare gli operatori \ e % ma facendo la divisione per sottrazioni successive.
19. **Lancio del dado:** simulare il lancio di un dado con un numero casuale compreso tra 1 e 6. Quando esce il numero 6 restituisce quanti lanci sono stati effettuati.
20. **Il problema dell'accattone:** Un accattone al mattino non ha neppure un centesimo. Chiede una moneta ad ogni passante fino a che non ha 5 euro, così può andare a fare colazione. Realizzare l'algoritmo individuando input e output.

    Evoluzione: quando ha raccolto i 5 euro, è interessato anche a sapere QUANTE monete ha raccolto. Modificare l'algoritmo.
21. **Deposito di capitale dopo N anni:** Indicare, in assenza di prelievi e versamenti l'importo dopo N anni con tasso dato e interessi reinvestiti.
22. **Raddoppio del capitale investito:** Deposito di capitale: calcolare quanti anni sono necessari per raddoppiare il capitale iniziale con tasso dato e interessi reinvestiti.
23. **Calcolo del quadrato di un numero con le somme:** Sapendo che il quadrato di un numero intero positivo n è la somma dei primi n numeri dispari, realizzare la funzione e utilizzarla per il calcolo dei quadrati dei primi 100 numeri naturali.

24. **Tabella di conversione della temperatura:** creare una tabella di conversione tra le varie unita' di misura della temperatura (Celsius, Farheneit e Kelvin). La tabella deve prendere in considerazione valori di temperatura in Celsius compresi tra Tmin e Tmax. I successivi valori della temperatura a partire dall'estremo inferiore Tmin si ottengono incrementando ogni volta, con passo predefinito deltaT, il valore della temperatura fino al valore dell'estremo superiore Tmax.

25. **Durata di un mutuo:** Per l'acquisto di una casa si vuole chiedere un mutuo di 100.000 Euro. La banca concede il mutuo all'interesse del 4% e richiede il versamento di una quota trimestrale di 2.000 Euro. Si calcoli in quanto tempo il mutuo verra' estinto.

    Osservazione. Sapendo che l'interesse annuo richiesto dalla banca è del 4% si ragiona nel seguente modo: ogni anno si deve pagare l'interesse sul capitale residuo, più una quota di capitale per ridurre il debito. Dato che le quote annue pagate sono costanti, risulta che nei primi anni le quote sono composte soprattutto di interessi mentre negli ultimi, in cui l'interesse è più basso perchè la maggior parte del capitale è stato restituito, le quote sono composte soprattutto di capitale da restituire.

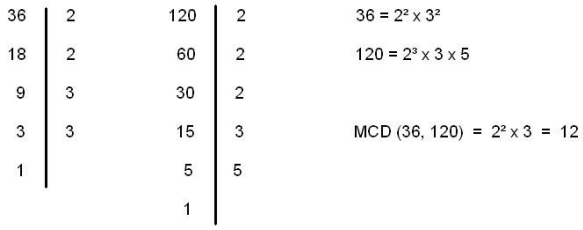
26. **Divisori di un numero:** Dato un numero N intero positivo mostrare a video i suoi divisori.

27. **Conversione decimale-binario:** Realizzare un algoritmo che consenta la conversione da decimale a binario di un numero intero qualsiasi. Il risultato viene mostrato a video, in ordine inverso (ad esempio, 6 viene mostrato come 011). Sfida: trovare il modo di visualizzare il risultato nell'ordine corretto.

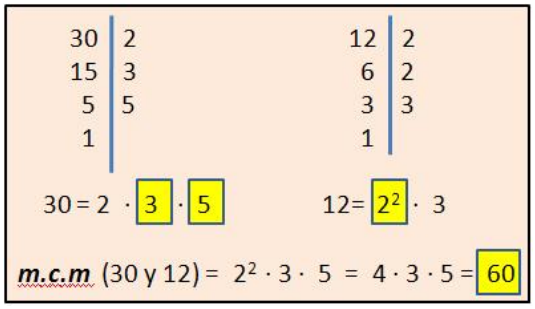
28. **Tabella di dati cinematici:** Costruire una tabella che, secondo per secondo, fornisca i dati di posizione e velocità relativi al moto di un carrello trascinato su una rotaia orizzontale dalla caduta libera di un corpo collegato al carrello tramite una fune e una carrucola posta su una estremità della rotaia. La rotaia è del tipo a cuscino d'aria e quindi l'attrito è pressochè nullo. Dati: MassaCorpo=0.4 Kg, LunghezzaRotaia=2 m.

    Si ricorda: F=ma, a=variazione di velocità/variazione di tempo.

29. **Massimo comun divisore:** Dati due numeri interi positivi calcolare il MCD.



30. **Minimo comune multiplo:** Dati due numeri interi positivi calcolare il mcm.



31. **Calcolo del seno per serie:** Il valore del seno di un angolo (espresso in radianti) può essere ottenuto con la formula seguente (^ = elevamento a potenza):

```
sin(x) = x - x^3/3! + x^5/5! - ... (-1)^n  x^(2n+1)/(2n+1) + ...
```

Calcolare quanti termini sono necessari per ottenere una approssimazione al valore di sin(x) pari al P per cento.

Suggerimento: per ogni valore ottenuto si calcola la variazione (assoluta, cioè senza tenere conto del segno) rispetto al ciclo precedente. Se il valore ottenuto, espresso in percentuale, è inferiore al valore di P ci si ferma.

32. **Espansione termica:** La lunghezza L di una barra metallica ad una temperatura T (espressa in gradi centigradi) è data dall'equazione:

```
L = Lo + E*T*Lo
```

dove Lo è la lunghezza a 0 gradi ed E è il coefficiente di espansione.

Produrre una tabella che riporti la lunghezza di una barra a diverse temperature comprese tra 0 e 100 gradi, supponendo che, alla temperatura di 20 gradi, la barra sia lunga esattamente 1 metro.

Il programma deve essere in grado di acquisire il coefficiente di espansione della barra da prendere in considerazione.

33. **Tabella di Pitagora:** Stampare la tabellina pitagorica.

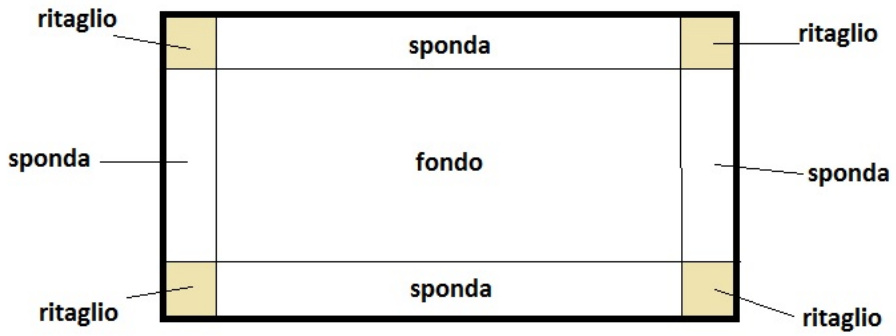
	1	2	3	4	5	6	7	8	9	10
1	1	2	3	4	5	6	7	8	9	10
2	2	4	6	8	10	12	14	16	18	20
3	3	6	9	12	15	18	21	24	27	30
4	4	8	12	16	20	24	28	32	36	40
5	5	10	15	20	25	30	35	40	45	50
6	6	12	18	24	30	36	42	48	54	60
7	7	14	21	28	35	42	49	56	63	70
8	8	16	24	32	40	48	56	64	72	80
9	9	18	27	36	45	54	63	72	81	90
10	10	20	30	40	50	60	70	80	90	100

34. **Successione di Fibonacci:** Calcolare i primi N numeri della successione di Fibonacci. La successione di Fibonacci è quella in cui il numero successivo si ottiene per somma dei due elementi precedenti. La successione deve essere inizializzata con i numeri 1 e 1.

    Esempio: 1 1 2 3 5 8 13 21 34 55 ....

35. **La vasca - ricerca del massimo volume con superficie limitata:** Per realizzare un esperimento di idraulica in laboratorio è necessario costruire un serbatoio. In magazzino sono disponibili alcuni fogli rettangolari di lamierino di ottone, di misure diverse. Bisogna poter valutare velocemente il massimo volume ottenibile con i diversi fogli, pensando di realizzare un serbatoio a forma di parallelepipedo, scoperto. Chi ha analizzato il problema ha già scartato l'ipotesi di risolverlo studiando una funzione di terzo grado quale è, in questo caso, il volume del prisma rispetto a uno spigolo.

    Definire l'algoritmo per acquisire le misure di un foglio e produrre per via enumerativa un valore approssimato del massimo volume ottenibile, pensando di piegare il foglio come illustrato:



    [Verifica: la soluzione, per un foglio di 50 cm per 80 cm, vale 10 cm]

36. **Allevamento bovino:** Un allevamento bovino è costituito da un numero iniziale di 1500 capi. Di essi il 67% sono mucche e l'88% delle mucche dà alla luce nel corso dell'anno un vitello. Il 16% dei capi viene macellato o muore ogni anno per cause naturali. Produrre una tabella che mostri l'andamento dell'allevamento nei prossimi 10 anni.

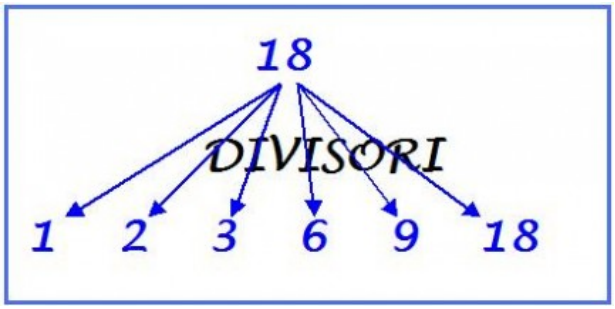
37. **Calcolatrice da tavolo:** Realizzare una calcolatrice da tavolo: chiede un operatore (+,-,*,/) e due operandi e mostra il risultato dell'operazione. Il programma deve consentire l'esecuzione di un qualsiasi numero di operazioni consecutive senza uscire.

38. **Conversioni di istruzioni di ciclo:** Dato il seguente algoritmo disegnare il flow-chart e riscriverlo trasformando i cicli for in while e in do..while.

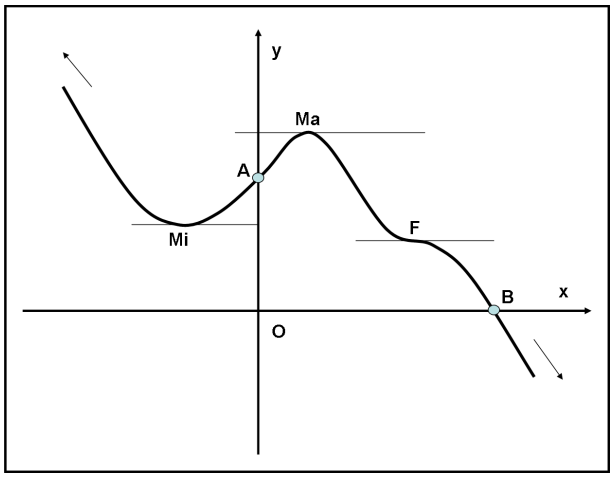
```
void mistero(){
   int conta;
   char lettera;
   for (conta=1; conta<=5; conta++){
      printf("%d ", conta);
      for(lettera = 'A'; lettera <= 'Z'; lettera++)
```

```
            printf("%c", lettera);
         printf("\n");
         }
      }
   }
   void main(){
      mistero();
      getch();
   }
```

39. **Quadratura del cerchio:** Determinare per tentativi la dimensione del lato di un quadrato che abbia la stessa area di un cerchio di raggio dato. Non si deve utilizzare la funzione sqrt().
40. **Minimo numero reale:** Trovare per tentativi il più piccolo numero reale rappresentabile in linguaggio C. (suggerimento: x è troppo piccolo se 1+x=1; la precisione viene determinata decidendo come far decrescere x).
41. **Primalità:** Determinare se un numero intero è primo. Note: il valore 1 non è primo.
42. **Divisori consecutivi:** Trovare tutti i numeri inferiori a 10000 che hanno come divisori tre numeri consecutivi. La visualizzazione deve essere in ordine crescente.



43. **Tabelle di funzioni trigonometriche:** Visualizzare a scelta, anche più volte, le tabelle delle funzioni sin(x), cos(x) e tg(x) con passo di un grado e nell'intervallo di un periodo.
44. **Tabelle di funzioni:** dati i valori di a, b e c reali e xmin e xmax sempre reali, visualizzare una tabella con due colonne, x e f(x) dove f(x) = a*x*x + b * x + c. Il valore x parte da xmin e arriva a xmax in modo che nella tabella ci siano 20 valori diversi di x.



45. **Calendario:** Realizzare una funzione che consenta di stampare un calendario mensile, con lo stesso formato di quello indicato qui sotto:

```
Lu Ma Me Gi Ve Sa Do
       1  2  3  4  5
 6  7  8  9 10 11 12
13 14 15 16 17 18 19
20 21 22 23 24 25 26
27 28 29 30 31
```

    Individuare con attenzione i parametri della funzione.

46. **Switch case:** Scrivere una funzione che prende in input un numero intero e una lettera e fornisce le seguenti informazioni sullo schermo a seconda della lettera: a) e' positivo o no; b) è pari o dispari; c) è divisibile per 3; d) è divisibile per cinque; e) è divisibile per 7.

    Restituisce la lettera maiuscola della scelta se la risposta è positiva, la lettera minuscola se la risposta è negativa. Se la lettera non è compresa tra quelle mostra a video "input errato, la lettera x non è valida." e restituise il carattere '?'.

47. **Gioco di Nim:** Si tratta di un gioco ben conosciuto, con un certo numero di varianti. Utilizzeremo una versione che ha una strategia interessante per arrivare alla vittoria. Due giocatori prendono a turno le biglie da un mucchio. In ciascun turno, il giocatore sceglie quante biglie levare: deve prenderne almeno una, ma non oltre la metà del mucchio. Quindi tocca all'altro giocatore. Perde chi rimane con l'ultima biglia.

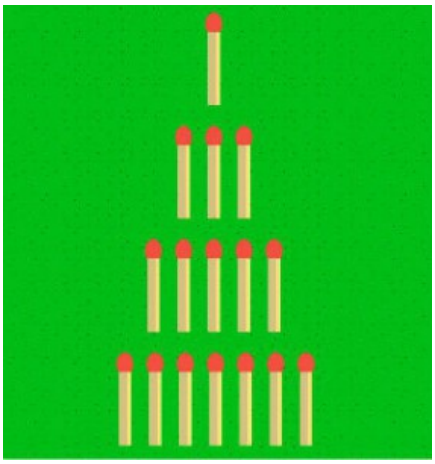
    Scrivete un programma completo in cui il computer gioca contro un avversario umano. Generate un numero intero casuale, compreso fra 10 e 100, per indicare il numero iniziale di biglie. Generate un altro numero intero casuale, compreso fra zero e uno, per decidere se la prima mossa tocca al computer o al giocatore. Generate un numero intero casuale, compreso fra zero e uno, per stabilire se il computer giocherà in modo intelligente o stupido. Nel modo stupido, quando tocca il suo turno, il computer si limita a sottrarre dal mucchio un numero casuale di biglie, purché sia una quantità ammessa (compresa fra 1 e n/2). Nel modo intelligente, il computer leva il numero di biglie sufficiente affinché il numero di quelle rimanenti sia uguale a una potenza di due, meno uno, ovvero 3, 7, 15, 31 o 63. E' sempre una mossa valida, eccetto quando il numero delle biglie è inferiore di un'unità a una potenza di due. In questo caso, il computer preleverà una quantità casuale, purché ammessa.

    Noterete che, nel modo intelligente, il computer non si può battere quando ha la prima mossa, a meno che il mucchio non contenga 15, 31 o 63 biglie. Naturalmente, un giocatore umano che ha la prima mossa, e che conosca la strategia vincente, puo' vincere contro il computer.

    (Horstmann, P6.17).

48. **Gioco dei fiammiferi:** Sul tavolo sono presenti 11 fiammiferi. Ogni giocatore a turno ne può prendere 1, 2 o 3 a sua scelta. Perde chi prende l'ultimo. Trovare la strategia vincente e realizzare un algoritmo che la applichi giocando con voi.

49. **Gioco di Marienbad:** Simile al gioco di Nim, si gioca con i fiammiferi.



```
Le regole del gioco sono le seguenti:
a) I due giocatori muovono a turno, togliendo fiammiferi dalla tavola.
b) A chi tocchi la prima mossa si decide per sorteggio, o per accordo.
c) Si possono prendere tanti fiammiferi quanti si vogliono (almeno uno)
ma da una sola riga.
d) Chi resta con l'ultimo fiammifero ha perso.
Supponiamo che la partita si giochi tra due giocatori "perfetti", che eseguono
sempre le mosse migliori, senza commettere errori.
Vengono poste le seguenti domande:
a. Il giocatore perfetto che incomincia vince per forza?
b. Il giocatore perfetto che incomincia perde per forza?
c. Il giocatore perfetto che incomincia vince o perde secondo
   gli sviluppi casuali del gioco?
```

Una volta risposto alle domande realizzare il programma che consente di giocare contro il computer.

50. **Calcolo del pi greco con le gocce:** Si espone alla pioggia un foglio di carta quadrato con inscritto un cerchio di diametro pari al lato del foglio. Il valore di pi greco è ricavabile considerando il numero delle gocce cadute nelle due figure (quadrato e cerchio) proporzionale alle aree.

    Suggerimento: considerare il cerchio centrato in 0,0 e il quadrato con angoli nei punti 1,1 e -1,-1.

51. **Stampa libretto:** Alcune stampanti prevedono la possibilità di stampa in formato "libretto": vengono stampate due pagine per ogni foglio, sul fronte e sul retro, in modo che piegando a metà il fascicolo di fogli si ottenga un opuscolo di dimensione A5. Praticamente: se sono da stampare fino a 8 pagine il primo foglio contiene su un lato (verso) le pagine 1 e 8 e dall'altro (retro) la 2 e la 7; il secondo foglio sul verso la 3 e la 6 e sul retro la 4 e la 5 (fare le prove manualmente). Scrivere una funzione che, dato un numero di pagine, mostri la lista dei fogli da utilizzare con i numeri di pagina sul verso e sul retro, con questo formato:

input = 6

```
  foglio 1: V 1, bianca; R 2, bianca
  foglio 2: V 3, 6 R 4, 5
```

Scrivere un programma di prova per i casi di 9 e 15 pagine che consenta di verificare se la stampa è quella attesa.

52. **Onda quadra:** Disegnare un'onda quadra per 10 periodi, ognuno di 6 caratteri come dimensione, centrata nello schermo, con valore picco-picco di 10 linee e valor medio 0.

    Evoluzione: fornire la possibilità di cambiare tutti i parametri compreso il duty-cycle.

53. **Onda triangolare:** Disegnare un'onda triangolare per k periodi, ognuno di nc caratteri come dimensione, centrata nello schermo, con valore picco-picco di vpp linee e valor medio vm.

54. **Numeri perfetti:** Un numero perfetto è un numero uguale alla somma dei suoi divisori escludendo il numero stesso e includendo l'unità. Per esempio i divisori di 28 sono: 1, 2, 4, 7, 14 e 28 stesso; escludendo 28 si ha che 1 + 2 + 4 + 7 + 14 = 28 perciò 28 è un numero perfetto.

    Progettare una funzione che verifichi se un numero N è perfetto o meno (suggerimento: sommare tutti i divisori del numero e verificare se la somma è uguale al numero stesso).

    I primi numeri perfetti sono 6, 28, 496, 8128, 33550336).

55. **Numeri primi gemelli:** Due numeri primi sono detti gemelli se distano 2. Quindi sono quelli presenti nelle coppie 3 e 5, 5 e 7, 11 e 13 e così via. Realizzare un programma che conta quante sono le coppie di numeri primi gemelli inferiori a 1000.

56. **Decadimento radioattivo:** Ogni anno un materiale radioattivo perde per disintegrazione atomica una certa percentuale p della propria massa m secondo la seguente formula:

    ```
              m * p
    m = m - -------
              100
    ```

    Scrivere una funzione che, a partire dalla massa m espressa in grammi, dal numero di anni N e dalla percentuale di disintegrazione p, determini la massa residua di un materiale radioattivo.

    Scrivere una seconda funzione che determina il numero di anni necessario per avere la metà della massa di partenza.

57. **Morra cinese:** Realizzare il gioco della Morra cinese: il programma fa scegliere all'utente quante partite giocare. Il computer sceglie a caso una delle tre possibilità e la confronta con la scelta dell'utente, aggiornando il punteggio.

    Alla fine comunica se ha vinto l'utente o il computer e con quale punteggio.

58. **Gioco d'azzardo:** Il computer determina in modo pseudocasuale una terna di simboli,, ciascuno dei quali potrà essere " $ " oppure " ? ". Ad esempio, un esito possibile è

    ```
    $ ? $
    ```

    Ogni partita, il giocatore investe 1 euro.

    Se escono tutti " $ ", egli incassa 7 euro (si riprende quindi l'euro puntato, più 6 euro di vincita netta), altrimenti l'euro investito è perso.

Si effettuano n partite, con n deciso dal giocatore, e viene determinata la situazione finanziaria del giocatore dopo ciascuna partita e quindi anche al termine della sequenza di partite (converrà utilizzare l'istruzione "delay" e mandare di volta in volta sempre in output la terna estratta).

Tutto ciò permetterà, tra l'altro, di stabilire se il gioco è equo: basterà far effettuare un numero molto elevato di partite ed osservare se il giocatore tende complessivamente a essere in perdita o in vincita.

59. **Sucessore con stessi bit a 1:** dato un numero intero positivo >0, trovare il minore tra i valori più grandi di lui che abbia lo stesso numero di bit a 1.

    Esempi:

```
Input: 129  => Output: 130 (10000001 => 10000010)
Input: 127 => Output: 191 (01111111 => 10111111)
Input: 1 => Output: 2 (01 => 10)
Input: 323423 => Output: 323439 (10011101111101011111 => 10011101111101101111)
```

    https://www.codewars.com/kata/56bdd0aec5dc03d7780010a5 (Codewars)

60. **Divisori:** Letto in ingresso un intero n, il programma ne elenca i divisori e li conta, tenendo conto anche dei divisori "impropri" (che sono il numero stesso e l'unità).

61. **Riduzione ai minimi termini:** Letti in input i due termini a, b di una frazione, ridurre la frazione data ai minimi termini. Ad esempio, se l'input è 180 e 450, l'output sarà 2/5.

62. **Giorno della settimana:** Scrivere una funzione int GiornoSettimana(int gg, int mm, int aa) che calcoli in che giorno cade una certa data. Restituisce 0 per domenica, 1 per lunedì e così via. Partire da una data di cui si conosce il giorno in cui cade, calcolare la differenza in giorni e da qui il risultato. Attenzione: poiché l'attuale calendario (Gregoriano) ha avuto inizio l'11 Novembre 1582, la funzione non darà risultati attendibili per date precedenti a questa.

63. **Successione di Syracuse:** La congettura di Collatz riguarda il seguente algoritmo: si prenda un intero positivo n.

    Se n = 1, l'algoritmo termina. Se n è pari, si divida per due; altrimenti si moltiplichi per 3 e si aggiunga 1. Per esempio, iniziando con n = 6, otteniamo la successione 6, 3, 10, 5, 16, 8, 4, 2, 1.

    La congettura di Collatz asserisce che questo algoritmo giunge sempre a termine, cioè al valore 1, indipendentemente dal valore di partenza; finora risulta vera per qualunque numero fino a 1,003 per 10 alla 20. Nel nostro caso vogliamo una funzione che, per un qualunque numero naturale, stampa a video i valori ottenuti via via e restituisce il loro numero. Il programma verifica che la congettura è corretta per tutti i valori inferiori a un milione e evidenzia il valore (o i valori) che genera la sequenza più lunga.

64. **Numeri narcisistici:** I numeri narcisistici (a numeridi Amstrong, o numeri piuccheperfetti) sono i numeri il nui valore corrisponde alla somma delle cifre elevate alla potenza corrsipondente al nmero di cifre del numero stesso. Ad esempio il numero 370 è narcisistico perchè ha tre cifre e 3 elevato alla terza più sette elevato alla terza più 0 elevato alla terza fa 370.

Scrivere una funzione che verifica se un numero è narcisistico e cercare i numeri narcisistici inferiori a 10000.

Suggerimento: lo sono i numeri tra 0 e 9, l'ultimo dovrebbe essere 8208).

65. **Spartizione del bottino:** Il famoso brigante Totò le Mokò ha ideato un metodo per spartire il bottino con il suo complice Gegè: se ci sono N monete da spartire, comincia a tenerne una per sè dicendo "Una a me" e una la dà al complice "Una a te"; poi ne prende due per sè dicendo "Due a me" e ne dà un'altra al complice "Due a te", poi tre per sè e una terza per il complice, fino a che ci sono monete. Comincia sempre a dare la prima a sè stesso.

    Scrivere una funzione che dato il numero di monete restituisce quante ne prende Totò. Scrivere il main che chiede il numero di monete e stampa, dopo aver chiamato la funzione, le monete di Totò e del complice.

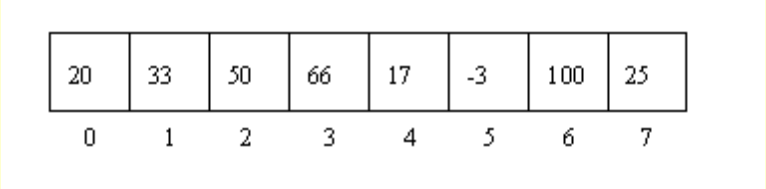
    Versione avanzata: prevedere un numero qualunque di complici, ugnuno dei quali viene trattato con lo stesso criterio. Modificare opportunamente la funzione. Anche qui stampare il numero di monete prese da ciascuno.

    (Adattato da una prova delle Olimpiadi di Informatica 2016, selezioni territoriali).

5. Array e stringhe

5.1. Array monodimensionali o vettori

Per array o vettore si intende una sequenza di elementi memorizzatoi uno di seguito all'altro, accessibili mediante la loro posizione (indice). L'indice parte da 0 (primo elemento del vettore v è v[0]) e arriva alla dimensione del vettore meno 1 (l'ultimo elemento del vettore v di dimensione DIM è v[DIM-1]).



Un generico array

Array parzialmente riempiti

Per array parzialmente riempito si intende un array dove non tutte le celle sono utilizzate.

Esistono almeno tre modi per realizzare un array parzialmente riempito:

- **con valore neutro:** se viene indicato un valore "neutro" sono da considerare solo gli elementi DIVERSI dal valore neutro, cioè è come se i valori della sequenza con valore neutro fossero vuoti (ad esempio se il valore neutro è il valore 0 e la sequenza contiene i valori 3, 0, 6, 7, 9, 0, 0 ,2 è da considerare la sottosequenza 3, 6, 7, 9, 2). E' come se ci fossero delle posizioni "vuote", dei buchi, nel vettore. Richiede esistano uno o più valori non significativi (neutri) per il problema.
- **con sentinella o terminatore:** utilizzando un valore speciale, denominato "terminatore" o "flag" o "sentinella", che segnala la fine degli elementi validi. Le celle successive contengono valori non significativi. Richiede che esista un valore particolare da usare come terminatore (ad esempio: se la sentinella è il valore -1 e la sequenza contiene i valori 3, 0, 6, -1, 4, 3, -1, 7 ... sono da considerare solo gli elementi della sottosequenza iniziale 3, 0, 6).
- **con indicatore di elementi validi:** si utilizza una varabile intera che contiene il numero di elementi iniziali validi: le due variabili, quella che contiene l'array vero e proprio e il contatore, sono da utilizzare come un tuttuno. Quando si modifica l'array aggiungendo o togliendo un elemento va aggiornato sempre anche il numero di elementi contenuti.

  Una variante, utilizzata in casi particolari di array di valori interi, prevede che, anziché usare una variabile apposita, si utilizzi la prima cella per contenere il numero di eleementi validi nel seguito.

  Esempio Se un array contiene la sequenza di valori 3, 0, 6, 7, 9, 0, 0, 2 e il numero di elementi validi è 4, sono da prendere in considerazione gli elementi, iniziali, 3, 0, 6, 7. Se si vuole eliminare il secondo elemento, i successivi DEVONO essere spostati indietro e il contatore deve essere decrementato.

```
3 0 6 7 9 0 0 2   cont 4
  ^
```

  Eliminando il secondo valore otteniamo:

```
3 6 7 7 9 0 0 2   cont 3
```

Si noti che si spostano, per ragioni di efficienza, solo i valori significativi dell'array.

Osservazione: la soluzione con contatore degli elementi è quella più generale.

1. **riempimento:** dato un vettore di DIM numeri interi inizializzarlo (cioè riempire le celle) con valori tutti pari a 0.
2. **riempimento con indice:** dato un vettore di DIM numeri interi inizializzarlo (cioè riempire le celle) con valori ognuno pari all'indice della cella.
3. **riempimento con multipli:** dato un vettore di DIM numeri interi e un numero intero "start", inizializzare il vettore inserendo "start" nella prima cella e nelle celle successive i multipli di "start" non divisibili per 7.
4. **serie crescente:** dato un vettore di numeri interi di dimensione DIM riempirlo con i numeri naturali partendo da un valore fornito in input.
5. **non consecutivi:** dato un vettore di numeri interi successivi trovare la posizione del primo elemento che non rispetta la regola. Ad esempio per il vettore 3, 4, 5, 6, 8, 9 il primo non consecutivo è 8. Attenzione al caso che siano tutti consecutivi.
6. **ricerca:** ricercare in un vettore di dimensione DIM la presenza di un valore e restituire la posizione della prima occorrenza.
7. **ricerca occorrenze:** ricercare in un vettore di dimensione DIM la presenza di un valore e restituire il numero di occorrenze nel vettore.
8. **vettore parzialmente riempito con flag:** ricercare, in un vettore di numeri interi parzialmente riempito con flag pari a -1, la presenza di valori naturali pari e restituirne il numero.
9. **vettore parzialmente riempito con buchi:** ricercare, in un vettore di numeri interi parzialmente riempito con valore "neutro" pari a -1, la presenza di valori naturali dispari e restituirne il numero.
10. **vettore di caratteri:** riempire un vettore di caratteri, di dimensione 26, con le lettere dell'alfabeto maiuscole.
11. **ricerca in un vettore di caratteri:** dato un vettore di caratteri di dimensione DIM cercare la sequenza più lunga di caratteri minuscoli dell'alfabeto in ordine, a partire da 'a' (ad esempio se nel file c'è una 'a', seguita da una 'b' nella successiva locazione e da una 'c' in quella ancora dopo, abbiamo una sequenza di 3). La funzione restituisce la posizione iniziale (cioè della 'a'), oppure -1 se non viene trovata.
12. **copia di un vettore:** dati due vettori di caratteri di dimensione DIM copiare il primo sul secondo cambiando tutti i caratteri che non sono lettere con un carattere spazio (' ').
13. **centratura di un vettore:** dato un vettore di caratteri di dimensione DIM1 e un secondo vettore di dimensione DIM2 (con DIM2 > DIM1), copiare il primo vettore sul secondo, avendo cura di "centrarlo" inserendo un numero adeguato di spazi prima e dopo.
14. **vettore con sentinella:** dato un vettore di caratteri di dimensione qualunque, gestito con un valore sentinella corrispondente al carattere spazio (' '), determinare il numero di

elementi che contiene (cioè quanti caratteri sono diversi dal valore sentinella a partire dalla prima posizione in avanti).

15. **minimo:** Elemento più piccolo in un array di numero reali pieno di dimensione DIM.
16. **posMinimo:** Posizione dell'elemento più piccolo in un array di numeri reali parzialmente riempito con indicatore del numero di elementi validi.
17. **posMax:** Posizione dell'elemento più grande in un array di numeri naturali parzialmente riempito con dimensione totale fornita e elementi non validi che contengono valori negativi.
18. **maggioriMedia:** Numero di elementi maggiori della media in un array di numeri interi parzialmente riempito con flag di terminazione che vale -9999.
19. **isInOrdine:** verifica se un vettore di numeri reali ha gli elementi in ordine strettamente crescente.
20. **filtro:** copia gli elementi di un vettore di numeri reali in un secondo vettore, solo se sono strettamente maggiori di un valore fornito in ingresso. Restituisce il numero di elementi copiati.
21. **compattatore:** Dato un vettore di numeri interi positivi, sposta gli elementi pari a zero in fondo al vettore. Restituisce il numero di elementi diversi da zero.
22. **maxDifferenzaAdiacenti:** dato un vettore di numeri reali restituisce il valore della massima differenza, in valore assoluto, tra due elementi in posizione consecutiva.
23. **maxDifferenza:** dato un vettore di numeri reali restituisce il valore della massima differenza, in valore assoluto, tra due elementi qualsiasi.
24. **numSup:** restituisce il numero di elementi di un vettore che sono superiori alla media dei valori contenuti.
25. **maggiorProdotto:** trova il maggiore tra i prodotti ottenuti moltiplicando due valori adiacenti di un vettore di numeri reali di dimensione data.
26. **minR:** Elemento più piccolo in un array di numero reali pieno di dimensione DIM, versione ricorsiva L'idea è che tale valore è il minimo fra:

    l'n-esimo elemento;

    il minimo elemento contenuto nel sottovettore costituito dai primi n-1 elementi (definizione ricorsiva);

    caso base: il minimo di un vettore di 1 elemento è l'elemento stesso.
27. **livellatore:** sostituisce tutti i valori negativi in un vettore di numeri reali con il valore 0. Restituisce il numero di elementi sostituiti.
28. **zeriConsecutivi:** Restituisce il numero massimo di zeri consecutivi presenti in un vettore.
29. **sequenzaCrescente:** restituisce la dimensione della più lunga sequenza di elementi strettamente crescenti.
30. **vicinoMedia:** restituisce la posizione dell'elemento più vicino alla media dei valori reali contenuti.

31. **stampaPositivi:** Stampa, uno per riga, la posizione e il valore degli elementi positivi contenuti in un array di valore reali.
32. **ordinamentoBitOn:** ordinare un vettore, composto di numeri interi e di dimensione data, in ordine ascendente per numero di bit a uno di ciascuno dei suoi elementi.
33. **Concatenazione:** Dati tre vettori, v1 di dimensione dim1, v2 di dimensione dim2 e v3 di dimensione ALMENO dim1+dim2 riempire v3 con tutti gli elementi di v1 seguiti da quelli di v2. Restituisce il numero di elementi inseriti in v3.
34. **Unione:** Dati tre vettori, v1 di dimensione dim1, v2 di dimensione dim2 e v3 di dimensione ALMENO dim1+dim2 riempire v3 con tutti gli elementi DISTINTI contenuti in v1 oppure in v2. Restituisce il numero di elementi inseriti in v3.
35. **Intersezione:** Dati tre vettori, v1 di dimensione dim1, v2 di dimensione dim2 e v3 di dimensione ALMENO dim1+dim2 riempire v3 con tutti gli elementi DISTINTI contenuti sia in v1 che in v2. Restituisce il numero di elementi inseriti in v3.
36. **Merge:** Sono dati tre vettori, v1 di dimensione dim1, v2 di dimensione dim2 e v3 di dimensione ALMENO max(dim1,dim2). I vettori v1 e v2 sono ordinati in modo crescente: riempire v3 con tutti gli elementi di v1 e v2 in disposti in ordine crescente. Restituisce il numero di elementi inseriti in v3.
37. **divisoriDistinti:** Dato un numero costruisce un vettore con i suoi divisori distinti.
38. **divisoriConOccorrenze:** Dato un numero costruisce due vettori: uno con i suoi fattori primi distinti e un secondo, nella posizione corrispondente, quante volte compare nella scomposizione (ad esempio: 24= 2*2*2*3: il 2 tre volte, il 3 una volta).
39. **multipli:** memorizzare in un array di 1000 elementi i primi 1000 multipli di N dato.
40. **separatore:** un array contiene numeri reali. E' parzialemnte riempito con flag pari a 0. Visualizzare su due colonne i valori positivi e i vaori negativi in esso contenuti.

```
Esempio: se i valori contenuti sono
     -11.7 1 3.05 -5.9 -0.4 0 5.6 -11.5 90.44 0 1.8
le due colonne saranno mostrate cosi:
                     Positivi  Negativi
                        1.00    -11.70
                        3.05     -5.90
                                 -0.40
```

41. **maggioritari:** verifica quanti dei valori reali, in un vettore di dimensione data, soddisfano la condizone di essere maggiori di tutti i successivi.
42. **quanti maggiori:** si chiedono all'utente 10 valori reali e si mostrano uno per riga, indicando a fianco quanti valori sono maggiori di esso.
43. **treDivisoriConsecutivi:** memorizzare in un array tutti i numeri inferiori a 10000 che hanno (almeno) tre divisori consecutivi.
44. **sommaDue:** Dato un vettore di N numeri interi ed un valore intero somma, elencare tutte le coppie di elementi che, sommati, forniscono quel valore.
45. **sommaN**:** Dato un vettore di N numeri interi e un valore intero x, stampare, una per riga, tutti i valori del vettore tali che la somma dei loro valori è uguale a x.

46. **convertitore di ore:** dato un vettore di caratteri di 5 elementi riempirlo con ore e minuti separati da un carattere ':'. Esempio: se fornisco 8 e 34 il vettore conterrà i cinque caratteri '0', '8', ':', '3', '4'. Gestire il caso di input non corretto (come?).
47. **sommaAltri*:** Dato un vettore di N numeri interi stampare tutti i valori v[i] del vettore tali che la somma degli altri valori (escluso cioè quello considerato) sia uguale a v[i].

    Ne esistono due versioni: la prima usa due cicli, una seconda altra con un solo ciclo.

    Realizzarle entrambe.
48. **stesse cifre:** Scrivere la funzione nextBigger(int n) che, dato un numero intero positivo n, trova il più piccolo valore successivo che ha le stesse cifre, -1 se non esiste. Esempi:

```
12   ==> 21
513  ==> 531
2017 ==> 2071
9    ==> -1
111  ==> -1
531  ==> -1
```

    https://www.codewars.com/kata/next-bigger-number-with-the-same-digits (Codewars)
    Variante: trovare il più grande valore inferiore con lo stesse cifre, senza considerare quelli con zeri iniziali.

    https://www.codewars.com/kata/5659c6d896bc135c4c00021e (Codewars)
49. **calcolaTempo:** si richiedono 40 misure di distanze (crescenti) percorse da un'auto dopo 1, 2, 3... 40 secondi. Dopo si richiede ciclicamente, fino a che l'operatore non introduce il valore 0, una distanza e il calcolatore indica quanti secondi sono necessari per percorrerla.

    (es. se le misure dopo 21 e 22 secondi sono rispettivamente 390.60 e 426.35 metri e viene richiesta la distanza di 400 metri il calcolatore risponde con il valore di 22 secondi).
50. **La sarta:** Si richiedono cinque misure di lunghezza, che sono le dimensioni di altrettanti scampoli di stoffa, e la lunghezza richiesta per la confezione di un abito. Si fornisce il numero di abiti confezionabili con gli scampoli.

    Uno scampolo consente la confezione di tanti abiti quante le volte che la misura dell'abito sta nello scampolo (es. con 8 metri si possono confezionare due abiti che richiedono 3.5 metri e 1 solo che ne richiede 5).
51. **La ditta di confezioni **:** Date cinque misure di lunghezza, che sono le dimensioni di altrettanti scampoli di stoffa, e altre due lunghezze richieste per la confezione di due abiti. Si richiede il numero massimo di abiti (che possono essere di una sola delle due misure o in parte di una e parte dell'altra misura) confezionabili con gli scampoli ma MINIMIZZANDO gli scarti. Uno scampolo consente la confezione di tanti abiti quante le

volte che la misura dell'abito sta nello scampolo (es. con 8 metri si possono confezionare due abiti che richiedono 3.5 metri e 1 solo che ne richiede 5).

52. **lamiere:** si acquisiscono due serie di 10 numeri reali che rappresentano una le lunghezze e l'altra le larghezze di 10 pezzi di lamiera. Il programma deve stampare il numero delle lamiere (compreso tra 0 e 9) che hanno area maggiore di un valore dato.
53. **troncaSequenze:** Definire una funzione che, ricevuti:

    un array di interi i cui elementi contengano esclusivamente valori 0 e 1,

    il numero n di elementi di tale array

    un altro intero k (con 1 <= k <= n),

    modifichi eventualmente il contenuto di alcuni elementi da 0 in 1, in modo tale che almeno un elemento ogni k consecutivi abbia valore 1. La funzione restituisce il numero di elementi modificati. Attenzione: il numero di elementi modificati dalla procedura, affinché sia garantita la proprietà desiderata, deve essere il minore possibile.
54. **Automi cellulari:** Consultare i siti che parlano di automi cellulari (ad esempio wikipedia) e realizzare un automa cellulare lineare che applichi la regola 28. Utilizzare un vettore di 77 caratteri contenente un solo 1 in posizione centrale, rappresentare gli uni e gli zeri con caratteri a scelta.

    Fase 2: chiedere all'utente il numero della regola e effettuare una simulazione.
55. **Triangolo di Tartaglia:** Realizzare una procedura che dato un interno n piccolo mostri sullo schermo le prime n riche del triangolo di Tartaglia (vedere, ad esempio, http://www.oilproject.org/lezione/triangolo-tartaglia-formula-di-newton-spiegazione-potenza-binomio-13066.html).

```
                1
              1   1
            1   2   1
          1   3   3   1
        1   4   6   4   1
      1   5   10  10  5   1
    1   6   15  20  15  6   1
```

56. **sommaFurba:** Esercizio proposto alle selezioni delle Olimpiadi di Informatica. Sia dato un insieme di numeri interi positivi di cui si vuole calcolare la somma totale. Potete

sommare due numeri alla volta, inserendo poi il risultato nell'insieme di numeri (al posto dei due sommati), fino ad avere un solo numero, pari alla somma totale dei numeri iniziali. Stabiliamo che il "costo" di una somma sia pari al valore della somma stessa; ad esempio, sommando 5 e 7 "paghiamo" 12.

Il particolare problema proposto consiste nel trovare il costo minimo quando l'insieme è costituito inizialmente dai seguenti otto numeri: 2, 5, 6, 8, 10, 12, 20, 27.

Come si deve procedere, in generale, per pagare il meno possibile? Un'idea è questa (ma sarà quella giusta?): ad ogni passo, si sommano i due numeri più piccoli presenti attualmente nell'insieme. La si realizzi mediante una funzione che riceva un array (contenente i numeri da sommare) e il numero n dei suoi elementi, e restituisca il "prezzo" pagato per eseguire le n-1 somme necessarie.

57. **Calcolo del resto potenziato:** Simulare il pagamento di uno o più beni in un negozio, avvenuto in contanti. A fronte di un pagamento con banconote o monete di cui sono dati taglio e numero, e assumendo che sia stato fatto un acquisto per un importo inferiore ai soldi versati, calcolare il resto da fornire utilizzando le banconote e monete disponibili in cassa, utilizzando il numero minimo di esse. Esempio: per pagare un importo di 86,60 Euro sono state usate due banconote da 50 Euro. In cassa ci sono 3 banconote da 50 euro, 3 da 20 euro, 5 da 10 Euro... 8 da 1 centesimo di euro. Per il resto si userà una banconota da 10 euro e...

58. **Confronto:** dati due vettori di uguale dimensione contenenti numeri interi stabilire nel più breve tempo possibile se contengono esattamente gli stessi elementi.

59. **Shut the box *:** E' un gioco tra un numero qualunque di contendenti (solitamente tra 2 e 4), che iniziano con nessuna penalità. In una scatola ci sono nove tessere, numerate da 1 a 9. A turno, ciascun giocatore lancia più volte due dadi. Con il valore ottenuto può togliere una o più tessere, per un totale esattamente pari alla somma dei punti dei due dadi. Se sono rimaste tessere per una somma inferiore a 7, il giocatore può scegliere di lanciare un solo dado anziché due. Se riesce a togliere tutte le tessere ha "chiuso la scatola" e la sua penalità è zero, altrimenti la sua penalità è la somma dei valori sulle tessere rimaste nella scatola. A questo punto tocca al secondo giocatore che comincia da capo. Vince chi, dopo tre turni, ha meno penalità accumulate. Scrivere un programma che giochi una partita tra l'elaboratore e un utente, mostrando a video come il gioco procede, simulando il lancio del o dei dadi.

60. **Shut the box in solitario:** Versione semplificata dell'esercizio precedente: il gioco viene svolto "in solitario", simulando il lancio dei dadi (o del dado, se le tessere 7, 8 e 9 sono state tolte e l'utente sceglie di lanciare un solo dado), controllando la correttezza delle azioni dell'utente.

61. **Shut the box con il domino:** Anzichè utilizzare i dadi si usano le tessere del domino (da simulare con un vettore di coppie, mescolate), che vengono via via scartate dopo l'utilizzo.

    Altre varianti possono essere trovate su wikipedia cercando la voce "Shut the Box".

62. **Crivello di Eratostene:** Dato un vettore di N elementi interi, utilizzarlo per realizzare il crivello di Eratostene. Il vettore al termine dell'algoritmo sarà riempito secondo queste convenzioni: l'indice del vettore rappresenta i numeri naturali; il contenuto di ogni elemento del vettore contiene 1 se il valore dell'indice è un un numero primo e 0 altrimenti. L'algoritmo prevede che venga inserito il valore 0 nei primi due elementi del vettore (0 e 1 non sono numeri primi) e 1 in tutte le celle successive.

    A questo punto l'algoritmo prevede di cercare il primo valore a 1 e "cancellare" (mettere a 0) tutti i suoi multipli inferiori a N. Si cerca poi il successivo valore a 1 e si cancellano i multipli, continuando fino a che non si arriva all'ultima cella. A questo punto solo nelle posizioni corrspondenti ai numeri primi ci sarà 1.

    Verificare il risultato stampando il vettore su due colonne: la prima con l'indice del vettore e la seconda con la dicitura "primo" se il vettore contiene 1, nulla altrimenti.

63. **Terorema sui numeri primi tra x e 2x:** Verificare, utilizzando il crivello di Eratostene dell'esercizio precedente, che per un qualunque numero x maggiore di 2, tra x e il suo doppio c'e' almeno un numero primo.

64. **Bagni pubblici:** é scientificamente accertato che, in un bagno pubblico, gli uomini generalmente preferiscono rendere massima la distanza tra sè e i servizi già occupati, occupando il servizio che si trova al centro della più lunga sequenza di servizi liberi.

    Ad esempio, esaminiamo questa situazione, in cui sono presenti dieci servizi, tutti liberi.

    ```
    _ _ _ _ _ _ _ _ _ _
    ```

    Il primo avventore occuperà una delle due posizioni centrali:

    ```
    _ _ _ _ _ X _ _ _ _
    ```

    Il successivo sceglierà il servizio che si trova al centro dell'area libera sulla sinistra:

    ```
    _ _ X _ _ X _ _ _ _
    ```

    Scrivete un programma che legga inizialmente il numero di servizi disponibili e visualizzi diagrammi nel formato appena descritto, uno per volta, man mano che i servizi vengono occupati. Suggerimento: usare un array di valori booleani (0 e 1) per indicare se un servizio è occupato oppure no.

65. **Solitario bulgaro:** Il gioco inizia con 45 carte (non è necessario che siano carte da gioco, bastano dei cartoncini impilabili senza segni di distinzione), che vanno divise in un certo numero di pile di dimensione casuale: si può iniziare, ad esempio, con pile di dimensioni pari a 20, 5, 1, 9 e 10. Ad ogni turno si prende una carta da ciascuna pila, formando con esse una nuova pila: la configurazione iniziale dell'esempio viene trasformata in un insieme di pile aventi dimensioni pari a 19, 4, 8, 9 e 5. Il solitario termina quando sono presenti, in qualsiasi ordine, le pile aventi dimensioni pari a 1, 2, 3, 4, 5, 6, 7, 8 e 9 (e si può dimostrare che accade sempre).

Il programma usa una funzione per creare generare una configurazione casuale di N pile, con N compreseo tra 1 e 5) e una funzione che riceve questa configurazione e stampa l'evoluzione delle pile turno dopo turno.

66. **Valutazione del quiz:** Il primo array in ingresso contiene le risposte corrette a un quiz, indicate ognuna con una stringa, come ad esempio ["a", "a", "b", "d"]. Il secondo array contiene le risposte dello studente. I due array sono non vuoti e della stessa lunghezza. Scrivere un funzione che restituisce il punteggio, ottenuto assegnando +4 ad ogni risposta corretta, -1 per ogni risposta errata, 0 per ogni risposta non data (stringa vuota) e sommando i valori ottenuti. Se il punteggio è negativo restituisce 0.

67. **Identificatore di geni:** Tratto da:

    http://introcs.cs.princeton.edu/java/31datatype/GeneFind.java.html Il gene e' l'unità funzionale dell'ereditarietà biologica, ossia l'unità di informazione genetica, capace di replicarsi, mutare, trasferirsi da una generazione all'altra, esprimersi, adattarsi all'ambiente e partecipare al processo evolutivo.

    Un esempio per chiarificare il concetto di genoma (in questo caso umano), è paragonarlo alle informazioni contenute in un libro:

    Il genoma, cioè il libro, contiene 23 cromosomi, ossia i capitoli.

    Ciascun capitolo contiene tra i 48 e i 250 milioni di lettere (cioè i nucleotidi A,C,G,T) senza spazi che le separino.

    Questo libro è contenuto nel nucleo della cellula.

    Il libro è contenuto nella sua interezza in ciascuna cellula dell'organismo, con pochissime eccezioni (i globuli rossi, per esempio, nell'uomo sono privi di nucleo).

    Nella sintesi delle proteine, una delle funzioni del DNA, un gene è individuato, all'interno di un genoma, da una sequenza introduttiva (codone di partenza, costituito da tre nucleotidi) e termina con un altro codone di altri tre nucleotidi. All'interno tra i due codoni i nucleotidi devono essere in numero multiplo di 3.

    Esempio: se il genoma fosse ATAGATGCATAGCGCATAGCTAGATGTGCTAGCAT, il codone di partenza ATG e il codone di fine TAG il nostro programma di analisi deve mostrare a video due geni:

```
Input: ATAGATGCATAGCGCATAGCTAGATGTGCTAGCAT
Start: ATG
Stop : TAG
Trovati 2 geni.
gene  1: CATAGCGCA
 gene  2: TGC
```

68. **cercaChiManca:** Viene fornito un array che contiene tutti i numeri interi da 0 a 100 in maniera disordinata, eccetto uno. Scrivere una funzione che restituisce il numero mancante.

5.2. Algoritmi di ricerca e ordinamento

1. **Ricerca sequenziale iterativa:** Dato un vettore e un valore da ricercare realizzare la funzione con l'algoritmo che scadisce il vettore fino a che non trova l'elemento.

Analisi: cosa restituire? cosa succede se il dato da cercare non è presente? come si comporta la funzione se ci sono più valori che corrispondono al dato cercato? fare una stima del tempo di ricerca:

se moltiplichiamo la dimensione per 100 di quanto cambia il tempo? Verificare i risultati in ognuno dei casi e tempi con prove adeguate mediante programma di test.

2. **Ricerca dicotomica:** Dato un vettore ORDINATO e un valore da ricercare realizzare la funzione con l'algoritmo che usa la ricerca dicotomica per trovare il dato cercato: confronta l'elemento centrale del vettore con il dato: se sono uguali ha trovato il dato, altrimenti considera la prima metà del vettore se il dato da cercare é minore di qeullo centrale o la seconda metà nell'altro caso.

   Analisi: cosa restituire? cosa succede se il dato da cercare non è presente? come si comporta la funzione se ci sono più valori che corrispondono al dato cercato? fare una stima del tempo di ricerca:

   se moltiplichiamo la dimensione per 100 di quanto cambia il tempo? Verificare i risultati in ognuno dei casi e tempi con prove adeguate mediante programma di test.

3. **Ricerca sequenziale ricorsiva:** Dato un vettore e un valore da ricercare realizzare la funzione ricorsiva che verifica se il dato è nell'ultima posizione; se non lo è richiama sé stessa sul sottovettore che escude l'ultimo elemento. Se la dimensione arriva a zero vuol dire che il dato cercato non é presente.

   Eseguire le stesse prove degli esercizi precedenti.

4. **Ricerca dicotomica ricorsiva:** Dato un vettore e un valore da ricercare realizzare la funzione ricorsiva che ordina il vettore (magari ricorsivamente...) e poi esegue la ricerca dicotomica in maniera ricorsiva: confronta il dato cercato con l'elemento in posizione centrale, se non sono uguali chiama sè stessa nella prima metà o nella seconda metà del vettore. Attenzione: non deve riordinare....

   Eseguire le stesse prove degli esercizi precedenti.

5. **Ordinamento su altro vettore:** realizzare una funzione che dati due vettori di numeri interi e la loro dimensione riempie il secondo con i valori contenuti nel primo messi in ordine crescente.

   Versione 1: senza duplicati.

   Versione 2: con possibile presenza di duplicati.

   In entrambi i casi non deve essere modificato in alcun modo il vettore di partenza.

6. **Ordinamento con Bubble sort:** realizzare una funzione che dato un vettore di numeri interi e la sua dimensione lo ordina con l'algoritmo del bubble sort.

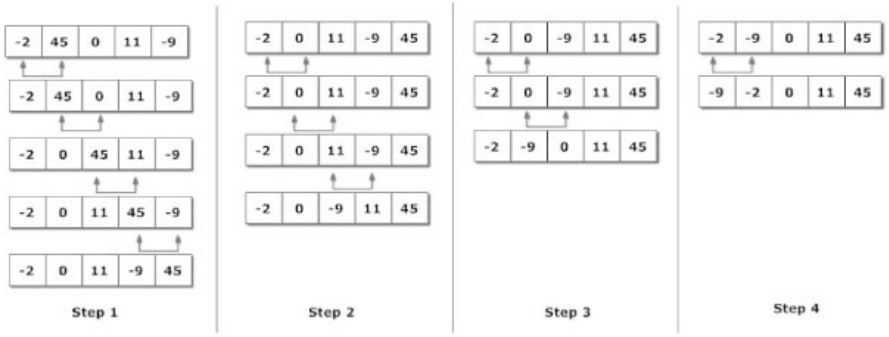


Figure: Working of Bubble sort algorithm

7. **Ordinamento con insertion sort:** realizzare una funzione che dato un vettore di numeri interi e la sua dimensione lo ordina con l'algoritmo dell'insertion sort.

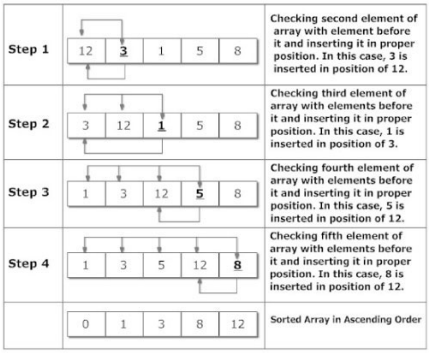


Figure: Sorting Array in Ascending Order Using Insertion Sort Algorithm

8. **Ordinamento con selection sort:** realizzare una funzione che dato un vettore di numeri interi e la sua dimensione lo ordina con l'algoritmo del selection sort.

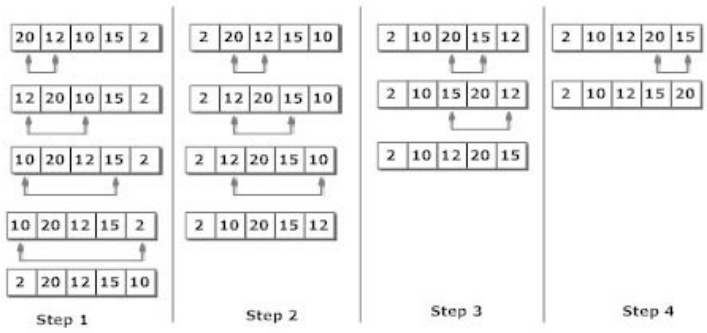
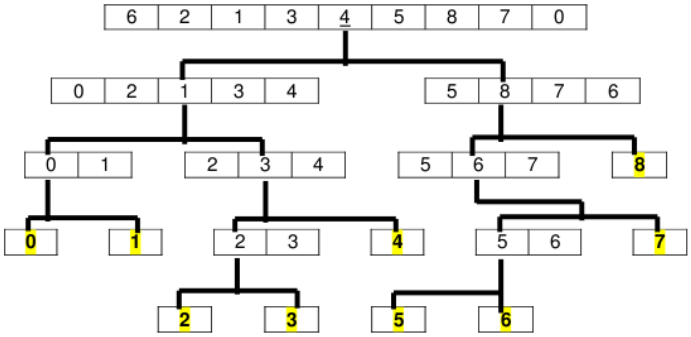
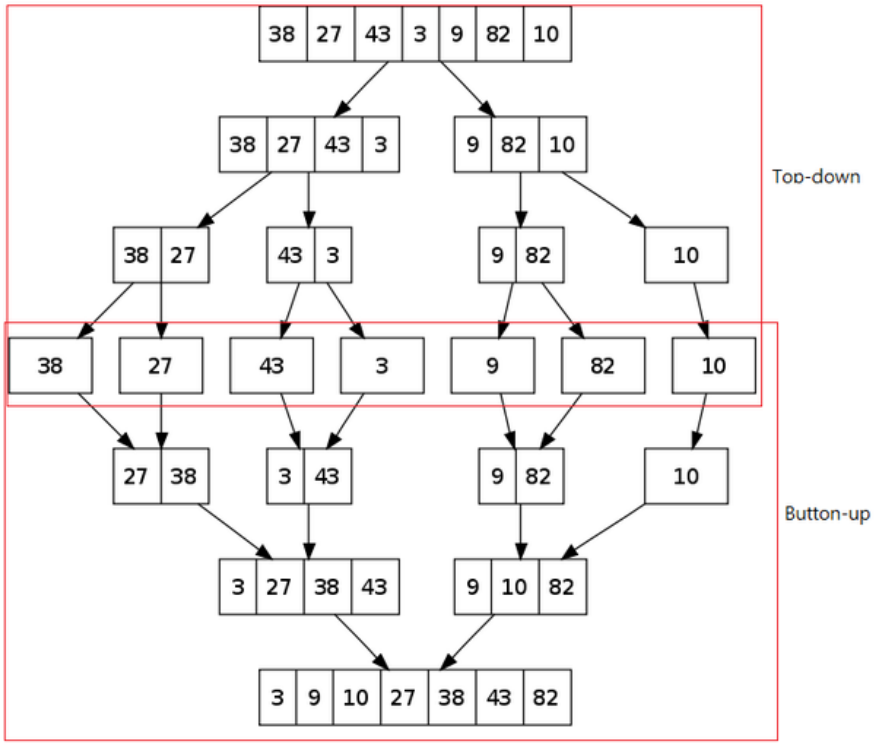


Figure: Selection Sort

9. **Ordinamento con quick sort **:** realizzare una funzione che dato un vettore di numeri interi e la sua dimensione lo ordina con l'algoritmo RICORSIVO del quick sort.

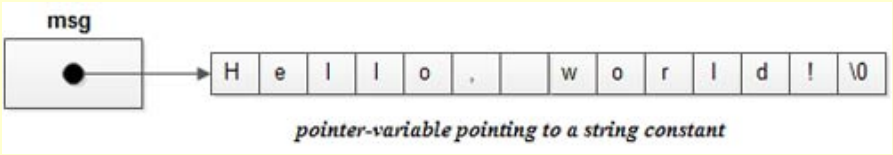


10. **Ordinamento con merge sort **:** realizzare una funzione che dato un vettore di numeri interi e la sua dimensione lo ordina con l'algoritmo del merge sort.



11. **Ricerca del valore più vicino:** Dato un vettore di numeri interi interi crescenti, trovare la posizione dell'elemento più vicino ad un dato valore. (Questa funzione è il nucleo che consente la soluzione del problema "Filiali" delle olimpiadi di informatica a squadre OIS 2017, Practice del 21.10.2016).

5.3. Stringhe



Distinzione:

- **array di caratteri:** sono array di quelli soliti, gestiti secondo le regole che il programmatore definisce per sue esigenze.
- **stringhe:** sono array di caratteri che vanno considerati come un unico elemento. Per scelta dei progettisti del linguaggio di programmazione hanno una serie di facilitazioni. In alcuni linguaggi sono array di caratteri gestiti con sentinella, in altri con contatoe degli elementi, in altri ancora sono tipi di dato predefiniti. Anche la codifica dei singoli caratteri può differire da un linguaggio all'altro.

Le stringhe, nel linguaggio C, sono array parzialmente riempiti gestiti che il sistema della sentinella. Il carattere di fine stringa è il carattere '\0' ("barra zero" o "slash zero").

Dichiarazione e uso di una stringa:

```
// stringa di massimo 99 caratteri (più '\0' finale)
char frase[100];

// stringa con valore iniziale, dimensione 12+1 caratteri
char fraseConInizializzazione[]="Ciao a tutti";
```

Osservazioni:

- NON viene fatto alcun controllo sull'accesso oltre la fine della stringa, nè in lettura, nè in scrittura.
- Per le stringhe in C non vale l'operatore di assegnazione (non si può scrivere s1="Ciao"; e neppure s2=s1).
- Si possono dichiarare senza specificare le dimensioni (char stringa[]="ciao ciao" definisce un array di 10 caratteri.
- esistono decine di funzioni predefinite per la loro gestione
- printf e scanf possono leggerle direttamente, ma scanf dà molti problemi perchè gestisce gli spazi come terminatori. Anche gets() è stata eliminata dallo standard del C (consentiva di leggere stringhe di lunghezza qualunque ma senza controllare che non superassero la dimensione del vettore in cui inserire quanto digitato). Conviene l'utilizzo di fgets che richiede di indicare il numero massimo di caratteri da leggere. Rimangono due problemi: viene letto il anche il '\n' e cosa fare del resto dei caratteri eventualmente digitati oltre quelli richiesti.

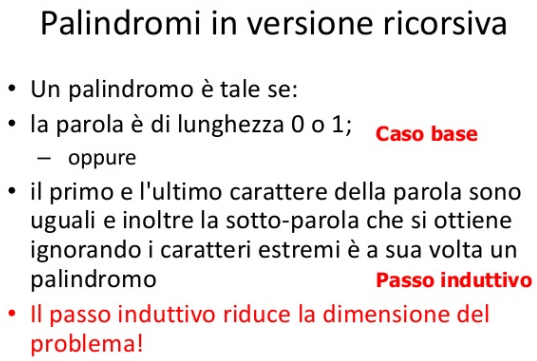
1. **lunghezza:** scrivere una funzione che calcoli la lunghezza di una stringa
2. **contaLettera:** scrivere una funzione che dica quante lettere 'a' ci sono in una stringa
3. **contaParole:** scrivere una funzione che dica quante parole ci sono in una stringa
4. **copiatrice:** scrivere una funzione che copi una stringa su un'altra
5. **contaDoppie:** conta il numero di doppie presenti in una stringa
6. **contaVocali:** numero di vocali presenti in una stringa
7. **contaLettere:** Data una stringa di lunghezza massima 80 caratteri contare il numero di lettere comprese tra 'A' e 'G' che contiene. Supporre i caratteri tutti maiuscoli.
8. **cambiaVocali:** sostituire tutte le vocali di una stringa con la lettera 'a'
9. **ugualiPer:** calcola quanti caratteri iniziali sono in comune tra due stringhe date.
10. **mescolatore:** comporre una stringa s3 prendendo alternativamente i caratteri dalle stringhe s1 e s2.
11. **presenze:** data una stringa restituire una seconda stringa, di lunghezza 26, composta solo di valori 0 o 1, a seconda che la lettera corrispondente sia presente o non nella stringa di

partenza. Ad esempio, se la stringa di parteza contine almeno una lettera 'a' o una lettera 'A', nella prima posizione della stringa in output ci sarà un '1', altrimenti uno '0'. Lo stesso con la lettera 'b' o 'B' per la seconda posizione e così via. Esempio: con

```
   "a   **&  cZ"  ===>  "10100000000000000000000001"
```

12. **rovescio:** invertire il contenuto di una stringa ("Ciao" diventa "oaiC").
13. **isAnagramma:** date due stringhe comunicare se una è l'anagramma dell'altra
14. **isPalindroma:** data una parola comunicare se è palindroma ("anna" e' palindroma)
15. **isFrasePalindroma:** data una frase comunicare se è palindroma (Sono palindrome, ad esempio: "I topi non avevano nipoti", "E nonno dai! Ci porti ai tropici a donnone?", "E' la' Roma, attici da Mafia. Hai fama di citta' amorale", "Era d'uso sudare", "Ai lati d'Isabella e Alle basi d'Italia").

    Nella figura la versione ricorsiva dell'algoritmo. Provare a realizzarla.



16. **Conta frequenze:** Realizzare un programma organizzato a menù che, date più stringhe di max DIM caratteri ognuna, presenta le seguenti voci:

```
a) inserire nuovo input.
b) frequenze di ogni lettera dell'alfabeto fino a questo momento inserita,
su cinque colonne.
c) lettera o lettere con più occorrenze.
d) lettere presenti in ordine di frequenza.
e) mostra ultima stringa inserita.
f) frequenze relative di ogni lettera dell'alfabeto.
g) numero di frasi fino a questo momento inserite.
x) fine programma.
```

17. **Conversione decimale-binario:** Dato un numero naturale produrre una stringa con l'equivalente valore binario. Ogni 4 cifre binarie deve essere indicato uno spazio. La stringa può contenere zeri iniziali per avere sempre un multiplo di 4 cifre binarie.

    Ad esempio 45 produce "0010 1101".

18. **Conversione decimale-binario 2:** Dato un numero intero (anche negativo) compreso tra -32768 e 32768 produrre una stringa con l'equivalente valore binario in complemento a 2, a 16 bit. Ogni 4 cifre binarie deve essere indicato uno spazio.

19. **expTab:** data una stringa la modifica sostituendo tutti i caratteri di tabulazione '\t' con tre spazi.
20. **contiene:** verificare se una stringa s1 contiene la stringa s2. restituisce la posizione oppure -1 se non è presente.
21. **sostituzione:** sostituire nella stringa s1 la stringa s2 con la stringa s3 (se presente). Restituisce: 1 se la sostituzione è avvenuta; -1 se s2 non è stata trovata in s1; -2 se s2 e s3 hanno lunghezza diversa (ed s1 non è stata modificata).
22. **mescolatoreFrasi:** date due stringhe produrne una terza prendendo una parola dalla prima e una dalla seconda alternativamente.
23. **sostituzionePlus:** sostituire nella stringa s1 la stringa s2 con la stringa s3 (se presente). Restituisce: 1 se la sostituzione è avvenuta; -1 se s2 non è stata trovata in s1. Attenzione: s2 e s3 possono avere lunghezze diverse.
24. **concat:** concatena s1 e s2 inserendo in mezzo il carattere c. I parametri sono s1, s2 e c.
25. **nomeCognome:** riceve due stringhe in ingresso, il nome e il cognome di una persona, e produce una terza stringa costituita dalle iniziali del nome o dei nomi seguite dal cognome o dai cognomi.

    Esempio: nome "Pier Domenico" cognome "Dell'Acqua" produce "P. D. Dell'Acqua".

    Realizzare un programma di test automatico della funzione con almeno quattro casi significativi.
26. **checksum:** Il codice internazionale per i libri (International Standard Book Number, ISBN) è un codice a 13 cifre, in cui il codice di controllo è calcolato con questo algoritmo:

    si moltiplica ogni cifra per un peso che assume in modo alternato i valori 1 e 3, partendo dalla prima cifra a sinistra che ha peso 1;

    si sommano i risultati

    si divide la somma per 10 e si prende il resto della divisione

    si sottrae il risultato da 10: questa cifra è la cifra di controllo; se quest'ultimo risultato fosse proprio 10, lo si sostituirà con 0.

    Esempio: dato un codice 978-88-430-2534

    ```
    9*1+7*3+8*1+8*3+8*1+4*3+3*1+0*3+2*1+5*3+3*1+4*3 = 117
    117 / 10 = 11 con resto 7
    10 - 7 = 3
    ```

    La prova della correttezza di un codice ISBN-13 si ottiene rieseguendo il calcolo iniziale, stavolta con tutte e 13 le cifre. Il risultato dell'addizione dovrà essere divisibile per 10.

    Realizzare due funzioni: una che verifica se un codice ISBN è corretto e una che date le prime 12 lettere di un codice ISBN calcola la l'ultima e la aggiunge in fondo.
27. **converteData:** Realizzare una funzione che accetta una stringa nel formato "dd/mm/yyyy", come ad esempio "24/06/2016", e la converte in "gg mese yyyy", nell'esempio "24 Giugno 2016".

28. **difettatore:** Realizzare una funzione che modifica una frase in ingresso aggiungendo una lettera 'f' dopo ogni vocale. Se la vocale è maiuscola anche la lettera aggiunta sarà maiuscola.

    Se la vocale è seguita da un'altra vocale va saltata. La funzione restituisce il numero di aggiunte che sono state fatte.

    Realizzare un programma di test che chiama la funzione con almeno tre casi significativi e ne verifica automaticamente la corretta esecuzione.

    Realizzare un programma che legge la frase da tastiera e visualizza la frase ottenuta e il numero di aggiunte fatte.

29. **inverteFrase:** Scrivere una funzione C che riceve in ingresso una stringa contenente una frase (parole separate soltanto da spazi) e una seconda stringa. Questa viene ottenuta invertendo l'ordine delle lettere di ogni singola parola della prima stringa.

    Esempio: "Qual ramo del Lago di Como" diventa "leuQ omar led ogaL id omoC" Creare poi un programma di test che verifichi automaticamente almeno quatttro casi significativi.

30. **maiuscolatore:** Realizzare una funzone che ricevuta una stringa composta da parole separate da spazi ne rende maiuscole tutte le iniziali e minuscole tutte le altre lettere.

31. **censura:** Realizzare una funzione che riceve in ingresso una frase e un vettore di frasi da censurare. La funzione modifica la prima stringa sostituendo tutte le occorrenze delle parole contenute nel vettore con un corrispondente numero di asterischi. Se le parole da censurare sono parte di una parola viene sostituita solo la parte da censurare.

32. **generatore di password:** Realizzare due funzioni che possano essere utilizzate per creare un insieme di password uniche e sicure per i dipendenti di una azienda. La prima funzione, generaPassword, produce una stringa contenente una password con le seguenti caratteristiche: 8 lettere alfabetiche, costituite da 4 coppie consonante-vocale, almeno una delle 4 consonanti deve essere maiuscola; ad esse segue un carattere speciale compreso nell'insieme [! ? . : _ = * + - $ &]; infine sono scelte tre cifre. Ad esempio potrebbe generare la password "ceToGuka=094".

    La seconda funzione generaInsiemeDiPassword riceve un vettore di stringhe e la sua dimensione: il suo scopo è riempire il vettore di password DISTINTE utilizzando la funzione generaPassword.

33. **Conversione di un numero in parola:** Scrivere una funzione, che prende in input un numero intero di al più tre cifre (tra 0 e 999), e una stringa che riempie con lo stesso numero scritto in parole (per esempio: "zero", "novecentonovantanove").

    Suggerimento. Usare gli operatori che fanno la divisione tra interi e ne danno il quoziente e il resto. Per esempio: (n%10) = cifra unità, ((n/100) % 10) = cifra decine, ecc.

    SFIDA: accettare un numero qualunque di cifre compreso tra 1 e 7.

34. **Controllo parentesi:** Data una stringa che rappresenta una espressione algebrica con eventuali parentesi tonde '(' e ')' per stabilire le priorità, scrivere una funzione che restituisce 1 se le parentesi sono ben messe (cioè ad ogni parentesi aperta ne corrisponde una chiusa), 0 altrimenti. Esempi:

```
"(2x +  (1-x)) + 3(dy/dx)+ 4 = 0" restituisce 1
```

```
   ")(4 + x )( = (dy/dx)"            restituisce 0
```

35. **Ricerca sottostringa iniziale:** scrivere la funzione strcspn che, date due stringhe s1 e s2 restituisce il numero di caratteri iniziali della stringa s1 che non contengono alcun carattere di s2.

    Versione inglese: calculates the length of the number of characters before the 1st occurrence of character present in both the string.

36. **Like:** Quando vengono assegnati dei "like" a un post su un social network sotto il post viene di solito indicata una lista di persone che hanno espresso la loro approvazione.

    Se il like è uno solo compare "Piace a Emanuele", se sono due la forma è "Piace a Federica e Lorenza", se sono più di due "Piace a Marco, Lorenzo e a altri 7". Se nessuno ha espresso gradimento compare "Nessun giudizio". Scrivere una funzione che mostra a schermo la frase corretta sulla base di una stringa contenente i nomi di coloro che hanno espresso il loro like, separati da virgole e zero o più spazi.

37. **Regola dei divisori di 13:** Quando si dividone le potenze di 10 successive (1, 10, 100, 1000...) per 13 si ottengono i valori 1, 10, 9, 12, 3, 4. Poi la sequenza si ripete.

    Da qui il seguente metodo: moltiplicare la cifra più a destra del numero da dividere per 13 con il numero più a sinistra nella sequenza indicata sopra, la seconda cifra più a destra alla seconda cifra più a sinistra del numero nella sequenza. Il ciclo continua, sommando via via tutti questi prodotti. Ripetere questo processo fino a quando la sequenza di somme è stazionaria, cioè fornisce sempre lo stesso valore.

    Esempio

    Quale è il resto della divisione di 1234567 con 13?

    7×1 + 6×10 + 5×9 + 4×12 + 3×3 + 2×4 + 1×1 = 178 Ripetiamo lo stesso procedimento con 178:

    8x1 + 7x10 + 1x9 = 87 e di nuovo con 87:

    7x1 + 8x10 = 87

    D'ora in poi la sequenza è stazionaria e il resto di 1234567 per 13 è lo stesso del resto di 87 per 13, cioè 9.

    Chiamare **tredici** la funzione che elabora questa sequenza di operazioni su un numero intero n (> = 0). tredici restituirà il numero fisso.

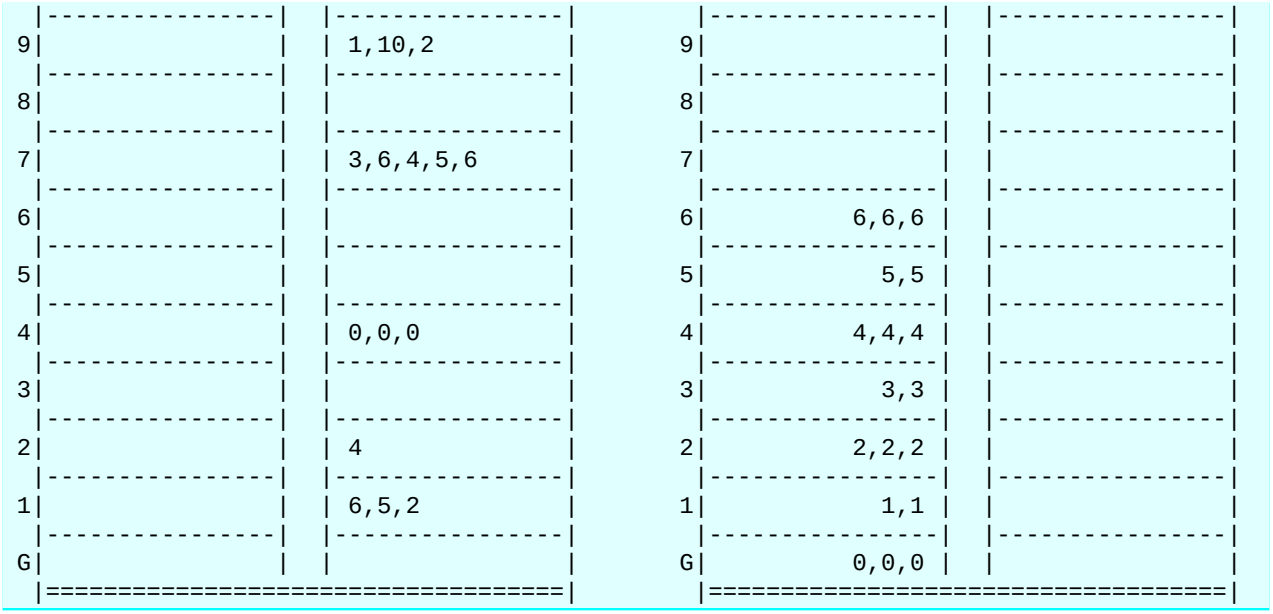
    tredici(1234567) calcola 178, quindi 87, quindi 87 e restituisce 87.

    tredici (321) calcola 48, 48 e restituisce 48.

    (Fonte *https://www.codewars.com/kata/a-rule-of-divisibility-by-13*)

38. **L'ascensore **:** Un edificio a più piani ha un ascensore. Le persone sono in coda su piani diversi in attesa dell'ascensore. Alcune persone vogliono salire. Alcune persone vogliono andare giù. Il piano a cui vogliono andare è rappresentato da un numero (quando entrano nell'ascensore lo scelgono premendo il bottone).

```
      PRIMA (persone in coda)                DOPO (persone arrivate da destinazione)
                    +--+                                    +--+
  /-----------------|  |-----------------\       /-----------------|  |-----------------\
10|                 |  | 1,4,3,2         |      10|             10 |  |                 |
```



Regole dell'ascensore.

L'ascensore va solo su o giù! Ogni piano ha sia i pulsanti SU che GIÙ (tranne i piani terminali che hanno solo GIÙ e SU). L'ascensore non cambia mai direzione finché non ci sono più persone che vogliono salire / scendere nella direzione in cui sta già viaggiando.

Quando è vuoto, l'ascensore cerca di essere intelligente. Per esempio, se stava salendo, potrebbe continuare per andare a raccogliere la persona al piano più alto che vuole scendere. Se stava scendendo, potrebbe continuare per andare a raccogliere la persona al piano più basso che desidera salire.

L'ascensore ha una capacità massima di persone.

Quando viene chiamato, l'ascensore si ferma a un piano anche se è pieno, anche se, a meno che qualcuno non scenda, nessun altro può salire! Se l'ascensore è vuoto e nessuna persona è in attesa, tornerà al piano terra.

Regole per le persone.

Le persone sono in "file" che rappresentano il loro ordine di arrivo per aspettare l'ascensore.

Tutte le persone possono premere i pulsanti SU / GIÙ di chiamata di sollevamento.

Solo le persone che vanno nella stessa direzione dell'ascensore possono accedervi.

L'entrata è secondo l'ordine "in coda", ma quelli che non sono in grado di entrare non bloccano quelli che possono dietro di loro.

Se una persona non è in grado di accedere a un ascensore completo, premerà nuovamente il pulsante SU / GIÙ Chiamata Lift dopo che è partito senza di essa.

Sfida: portare tutte le persone ai piani a cui vogliono andare obbedendo alle regole dell'ascensore e alle regole delle persone.

Lo scopo è restituire un elenco di tutti i piani in cui l'ascensore si è fermato (nell'ordine visitato!).

NOTA: l'ascensore si avvia sempre al piano terra e le persone in attesa al piano terra possono entrare immediatamente.

Analisi: Dati in ingresso:

Per ogni piano c'è un elenco di persone in coda.

L'altezza dell'edificio può essere qualunque valore intero >0.

0 = piano terra.

Non tutti i piani hanno code.

Per ogni coda di persone l'indice 0 corrisponde al primo della coda.

I numeri indicano a quale piano la persona vuole andare.

E' definita la capacità (numero massimo di persone ammesse) dell'ascensore.

Convalida dei parametri: tutti i parametri di input possono essere considerati corretti. Non c'è bisogno di controllare cose come: a) le persone che vogliono andare ai piani che non esistono; b) le persone che vogliono prendere l'ascensore per il piano su cui sono già; c) edifici con <2 piani; d) scantinati...

Output:

L'elenco di tutti i piani in cui l'ascensore si è fermato (nell'ordine visitato!).

(Fonte *https://www.codewars.com/kata/58905bfa1decb981da00009e*)

5.4. Array multidimensionali

Altre idee: http://impararegiocando.com/matematica.

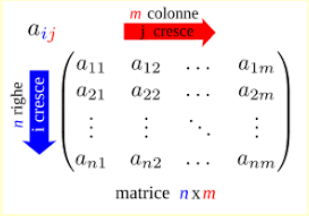
Array a due dimensioni: sono detti anche matrici.

Dichiarazione e uso:

```
int tab[4][3]; // 4 righe e tre colonne)
tab[0][2] = 5; // assegnazione
```

Somma del contenuto di una matrice

```
int i, j, somma =0;
int tab[4][3];
for(i=0; i<4; i++){
   for(j=0; j<4; j++){
      somma += tab[i][j];
   }
}
```



Nota: negli esercizi seguenti per "diagonale principale" si intende l'insieme delle celle con stesso valore per l'indice di riga e di colonna (0,0), (1,1)...

1. **Inizializzazione:** Data una matrice di numeri interi NxM riempirla di numeri casuali tra 0 e 100.
2. **Somma elementi:** Data una matrice di numeri interi NxM determinare la somma di tutti gli elementi contenuti.
3. **Ricerca elementi positivi:** Data una matrice di numeri interi NxM determinare quanti tra gli elementi contenuti nelle righe pari sono positivi. Stampare il contenuto della matrice e il risultato.
4. **Triangolare:** Data una matrice di numeri interi NxN determinare se è triangolare, cioè se tutti gli elementi al di sotto della diagonale principale sono zero.
5. **Simmetria:** Data una matrice di numeri interi NxN determinare se è simmetrica rispetto alla diagonale principale.
6. **Prodotto:** Data una matrice di numeri interi NxM e un vettre di M elementi determinare il prodotto della matrice per il vettore, che crea un vettore di N elementi (cercare definizione ad esempio su wikipedia).
7. **Distanze:** Considerare le distanze tra N località di una grande città e inserirle in una matrice quadrata NxN di numeri interi. Sulla diagonale principale ci saranno tutti zeri, nelle altre caselle la distanza tra la località corrispondente alla riga e la località corrispondente alla colonna. I valori potrebbero non essere gli stessi nelle due direzioni per via dei sensi unici. Realizzare una funzione inizializza che riempie la matrice con valori fissi e una seconda funzione che date due località (partenza e arrivo) restituisce la distanza tra esse. NOTA: fare ipotesi su come identificare le località).
8. **Verificatore di sudoku:** Dato uno schema risolto del sudoku, scrivere una funzione che verifica che non siano state violate le regole (sono semplici e ricavabili dalla rete internet).
9. **Temperature:** Realizzare un programma in grado di gestire una tabella che rappresenta le temperature di venti località in tutti i giorni di un mese. Il valore -999 indica un valore assente da non considerare.

   Deve consentire:

   a) di inserire i valori.

   b) di inserire valori casuali.

   c) di eliminare tutti i valori.

   d) di stampare il contenuto.

e) di indicare il valore massimo, il valore minino e la media dei valori presenti.

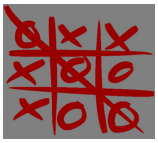
f) di trovare la località con la temperatura media più elevata.

g) di mostrare il giorno mediamente più caldo.

h) di cancellare tutti i valori relativi ad una località.

x) uscire dal programma.

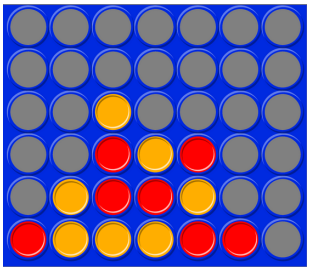
10. **Tris o filetto:** Realizzare il gioco del tris in cui uno dei giocatori è il calcolatore.



Versione 1: il calcolatore gioca a caso.

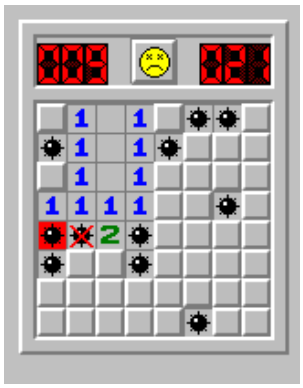
Versione 2: il calcolatore cerca di vincere. Se gioca per primo non perde.

11. **Forza 4:** I giocatori decidono tramite conta chi dei due inizierà il gioco. A turno faranno quindi scivolare la propria pedina nei binari prestabiliti dalla scacchiera cercando di allineare quattro pedine in fila orizzontale, verticale o obliqua. Nel frattempo devono stare attenti alle mosse dell'avversario per evitare che anch'esso riesca a fare quattro impilando le proprie. Vince la partita chi allinea per primo in orizzontale, verticale o obliquo quattro pedine del proprio colore. Nel caso nessuno dei due partecipanti al gioco riuscisse a fare "quattro" la partita finisce in parità e se ne comincia un'altra.

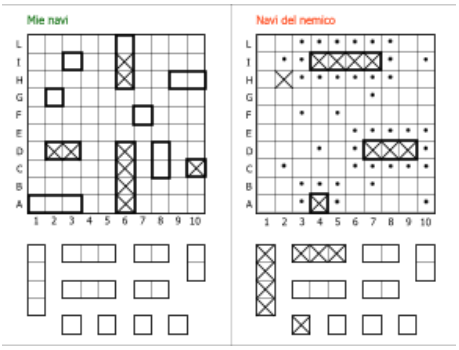


12. **Gara di bellezza:** In un concorso di bellezza, N giudici esprimono il loro giudizio su K candidati o candidate. Il giudizio è un valore numerico tra 0 e 5. Realizzare quattro funzioni che, ricevuta la matrice con i giudizi, restituiscano rispettivamente: la posizione del giudice più severo; il giudizio medio del/della concorrente più bello/a; la posizione del/della concorrente più bello/a, il giudice che ha assegnato più 5.

13. **Campo minato:** Realizzare il solitario del campo minato (ora prato fiorito...) in forma testuale.



14. **Battaglia navale:** Realizzare il gioco della battaglia navale.



Versione 2: uno dei giocatori è il calcolatore.

15. **Gioco del 15:** consiste nel cercare di riordinare in sequenza crescente le tessere numerate da 1 a 15 in una tabellina di forma quadrata di 4 righe e 4 colonne. Può essere spostata nella casella vuota una tra le tessere ad essa vicine.

16. **Sottomatrici a somma nulla:** Determinare se in una matrice NxM acquisita da tastiera vi sono sottomatrici quadrate in cui la somma degli elementi è nulla.

17. **Il viaggio del topolino:** Simulare il movimento di un topolino in un'area quadrata non piana. Si dispone di una matrice Q, di dimensione N * N, che descrive le caratteristiche di una certa area: in ciascuna locazione Q(x, y) della matrice vi è la quota del quadratino di superficie posto alle coordinate (x, y). A partire da una posizione iniziale del topolino, (x0, y0), si stampino le coordinate di tutti i quadratini toccati dal topolino se esso segue le seguenti regole di movimento:

    a) il topolino si sposta ad ogni passo di un solo quadratino, nelle 8 direzioni possibili.

    b) il topolino sceglie il quadratino su cui muoversi determinando il quadratino di quota massima tra gli 8 adiacenti.

c) se tutti i quadratini adiacenti sono ad una quota inferiore rispetto al quadratino attuale, il topolino si ferma.

Deve essere possibile scegliere la posizione iniziale del topolino.

Visualizzare il percorso del topo per una matrice 80 x 20.

18. **Cruciverba:** Data una matrice rettangolare contenente lo schema di un cruciverba e una parola verificare se la parola è presente nel cruciverba e indicarne la posizione di partenza (riga e colonna, partendo da 1) e la direzione (orizzontale o verticale).

    Esempio di cruciverba, dove i quadretti neri codificarli con asterischi.

MAROCCO si trova nella posizione 4,6 verticale.

Sfida: rappresentare inqualche modo anche i numerini interni allo schema e restituire quel valore al posto delle coordinate.

19. **Ricerca in matrice:** Realizzare una funzione che, data una matrice NxM di interi, trovi l'elemento per cui la media degli elementi ad esso adiacenti sia massima. Si visualizzino a video le coordinate di tale elemento ed il suo valore. Si considerino come adiacenti a ciascun elemento i quattro elementi nelle quattro direzioni cardinali. Si tratti inoltre l'ultima colonna come adiacente alla prima, e l'ultima riga come adiacente alla prima. Si supponga che N ed M possano variare tra 1 e 100.

    Proporre casi di test significativi.

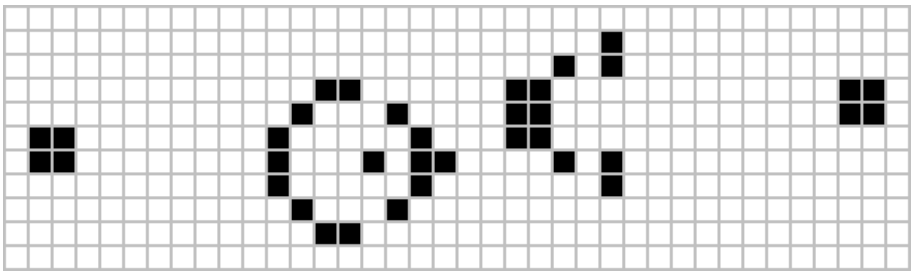
20. **Sottomatrici a somma nulla:** Realizzare una funzione in grado di determinare se in una matrice NxM, fornita in input, vi sono sottomatrici quadrate (di dimensione almeno 2x2) in cui la somma degli elementi è nulla. Restituisce il numero di matrici a somma nulla trovate.

21. **Gioco della Vita **:** Si tratta in realtà di un gioco senza giocatori, intendendo che la sua evoluzione è determinata dal suo stato iniziale, senza necessità di alcun input da parte di giocatori umani. Si svolge su una griglia di caselle quadrate (celle) che si estende all'infinito in tutte le direzioni; questa griglia è detta mondo. Ogni cella ha 8 vicini, che sono le celle ad essa adiacenti, includendo quelle in senso diagonale. Ogni cella può trovarsi in due stati: viva o morta (o accesa e spenta, on e off). Lo stato della griglia evolve in intervalli di tempo discreti, cioè scanditi in maniera netta. Gli stati di tutte le celle in un dato istante sono usati per calcolare lo stato delle celle all'istante successivo. Tutte le celle del mondo vengono

quindi aggiornate simultaneamente nel passaggio da un istante a quello successivo: passa così una generazione.

Le transizioni dipendono unicamente dallo stato delle celle vicine in quella generazione:

1. Qualsiasi cella viva con meno di due celle vive adiacenti muore, come per effetto d'isolamento; 2. Qualsiasi cella viva con due o tre celle vive adiacenti sopravvive alla generazione successiva; 3. Qualsiasi cella viva con più di tre celle vive adiacenti muore, come per effetto di sovrappopolazione; 4. Qualsiasi cella morta con esattamente tre celle vive adiacenti diventa una cella viva, come per effetto di riproduzione.



Cannone di Gosper, configurazione che produce un aliante

22. **Il problema di Giuseppe Flavio:** Nel cerchio del suicidio, dove ci sono N soldati numerati da 1 a N, ognuno fa fuori il compagno alla sinistra, in quale posizione dovete mettervi per essere l'ultimo sopravvissuto? Un rompicapo che spiega come il soldato Giuseppe evitò di togliersi la vita nel suicidio di massa durante l'assedio di Iotapata (67 d.C.).

    Fonte *https://www.dagospia.com/rubrica-2/media_e_tv/problema-flavio-giuseppe-cerchio-suicidio-dove-ognuno-fa-fuori-135948.htm*

23. **Il problema di Giuseppe complicato:** Con riferimento all'esercizio precedente, far variare oltre al numero di soggetti messi in cerchio anche la distanza tra chi uccide e la sua vittima: ad esempio, con distanza 3, il numero 1 ne salta 3 (2,3,4) e uccide il numero 5, il numero 2 ne salta 3 (3,4,6) e uccide il 7...

    https://www.codewars.com/kata/5550d638a99ddb113e0000a2

5.5. Passaggio di argomenti al main

1. **Sommatore al volo:** Realizzare un semplice main che in base al valore di argc esegue la somma dei numeri passati in input nel vettore argv. Attenzione: si tratta di stringhe...
2. **Array parzialmente riempito su linea di comando:** Scrivere un programma che legge N numeri interi da linea di comando (N<=100) e riempie un vettore con essi, poi stampa il vettore ordinato.
3. *** Comandi Unix:** Realizzare un programma che è in grado di ricevere delle opzioni sulla linea di comando, in stile unix: ad esempio "-h" e "-?" mostrano l'help del programma, "-v" mostra la versione, "-o filename" manda l'output sul file indicato...

5.6. Puntatori

I puntatori non sono altro che variabili contenenti dei riferimenti ad altre variabili.

Sono utilizzati per diversi scopi:

- per simulare il passaggio per riferimento dei parametri alle funzioni, in modo che al loro interno possano modificare il contenuto di variabili;
- per creare vettori di dimensione determinabile a tempo di esecuzione e in genere per realizzare l'allocazione dinamica della memoria;
- per realizzare strutture dati dinamiche quali liste, grafi, e alberi.

Per dichiarare un puntatore è sufficiente aggiungere un asterisco (scelta infelice) tra il tipo e il nome della variabile puntatore.

```
int * pInt;
float *pFloat, *pFloat2, unFloat;
char *ps;
```

Sebbene gli spazi siano ininfluenti conviene scrivere l'asterisco vicino al nome della variabile per il caso in cui ci siano più variabili nella stessa riga: nel caso dei float ci sono due puntatori (pFloat1 e pFloat2) e una variabile reale unFloat.

I puntatori possono assumere il valore 0, il valore NULL oppure quello di un riferimento ad una cella di memoria. In questo caso di usa l'operatore &, come nell'esempio seguente. Per ottenere il contenuto della cella riferita da una variabile puntatore si utilizza l'operatore * (di nuovo, altra pessima scelta del simbolo!).

```
int a, b;
int *p;
a = 7;
p = &a;      // p contiene un riferimento alla ("indirizzo della") variabile a
b = *p;     // b contiene il valore il cui riferimento (indirizzo) è p, di tipo int
            // quindi anche b contiene 7.
```

Attenzione: il tipo serve a determinare la dimensione del dato, cioè quanti byte occupano le variabili puntate dal puntatore.

Attenzione: non si possono assegnare riferimenti a variabili register (perchè non sono memorizzate nella RAM).

E' possibile fare operazioni sui puntatori, tenendo conto che operano in modo diverso a seconda del loro tipo: nei vettori il nome della variabile è in realtà un puntatore alla prima cella del vettore, ma si possono utilizzare anche i puntatori.

```
float a[10];
float *pa;
pa=&a[0];  // pa contiene l'indirizzo della locazione di memoria che contiene a[0]
```

Pertanto si può accedere alle locazioni del vettore a[0], a[1], a[2], anche mediante *pa, *(pa+1), *(pa+2).

La dicitura (pa+2) significa che viene calcolato il riferimento alla seconda cella successiva a quella riferita da pa.

Osservazione: nei prototipi delle funzioni, spesso per indicare un vettore si usa un puntatore, ma sarebbe da preferire la notazione con le parentesi quadre in quanto rendono evidente che si tratta di un vettore e non di una variabile scalare passata per riferimento.

```
int somma(int v[], int s);   // v è un vettore, s la dimensione
int somma(int *v, int s);    // v potrebbe essere un vettore oppure
                             // un riferimento a variabile
int somma(int *, int);       // equivalente alla precedente, meno chiara

...

int somma( int *v, int s){          int somma(int *v, int s){
    int tot = 0 , i;                     int tot;
    for(i=0; i<s; i++){                  tot = *v + s;
        tot += *(v+i);                   *v = tot;
    }                                    // non possibile v = &tot;
```

```
        return tot;                             return tot;
    }                                       }
```

1. Siano date tre variabili float a, b, s. Dovete, usando SOLO delle variabili puntatore che "puntano" a queste tra variabili, assegnare alle prime due dei valori qualunque e nella terza ottenere la somma. Infine stampare il risultato ottenuto. ATTENZIONE: a, b e s possono essere usate UNA VOLTA SOLA per ottenerne l'indirizzo.
2. Sia data una funzione che opera su puntatore a char (una stringa, in effetti). Contare quante vocali contiene sfruttando solo puntatori. Prototipo:

```
int contaVocali(char *ps);
```

6. Strutture dati complesse e allocazione dinamica

6.1. Struct e typedef

Definizione di un nuovo tipo di dato:

```
typedef struct libro {
   char titolo[50];
   char * autore;  // puntatore
   int pagine;
   float prezzo;
} Libro;
```

Dichiarazione e assegnazione di variabili del tipo della struttura

```
Libro testo = { "Il nome della Rosa", "Umberto Eco", 356, 9.80};
Libro copia;

copia = testo;  //assegnazione di una struttura ad un'altra
```

Accesso ai campi:

```
testo.pagine = 358;
strcpy(testo.titolo, "Il nome della Rosa");
testo.autore = "Umberto Eco";
```

Struct e funzioni: il passaggio come parametro avviene per VALORE (cioè viene fatta una copia della struct); è possibile il passaggio per riferimento, ma occorre fare attenzione alle parentesi o utilizzare la notazione "freccia" (->) per accedere ai campi.

```
void aumentaPrezzo(Libro * l){
   (*l).prezzo = (*l).prezzo + 2.0; // aumenta di 2 Euro
   l->prezzo = l->prezzo + 2.0;   // alternativa (notazione "freccia")
}
```

Funzioni che restituiscono delle struct: è possibile restituire una struttura ed assegnarla ad una variabile dello stesso tipo.

```
Libro creaLibro(char titolo[], char autore[], int numpag){
   Libro l;
   strcpy(l.titolo, titolo;
   l.autore = autore;
   l.pagine = numpag;
   l.prezzo = 10.0;
   return l;
}

main(){
   Libro testo;
   testo = creaLibro("Avere o essere?", "Erich Fromm", 266);
   printf("Il testo di %s e' intitolato %-20s\n", testo.autore, testo.titolo);
}
```

1. **Persona:** definire un nuovo tipo Persona in grado di contenere i campi nome, cognome, età di una persona. Dichiarare due varibili di tipo Persona, assegnare ai loro campi dei dati fissi e stampare poi le due variabili, una per riga, con il seguente formato:

"Nome e cognome: Giobatta Bignone eta': 89"

2. **Automobile:** definire un nuovo tipo Automobile in grado di contenere i campi targa, marca, modello, potenza, velocità massima, prezzo, optionals (un array di stringhe). Creare una variabile catorcio che contine i dati "AA 123 BB", "Fiat", "Panda 4x4", 50 kW, 125 km/h, 8500 Euro e ha come optional la vernice metallizzata e i cerchi in lega. Stampare il contenuto della variabile in modo ordinato.
3. **Garage:** Con riferimento all'esercizio precedente definire una variabile

```
Automobile garage[10];
```

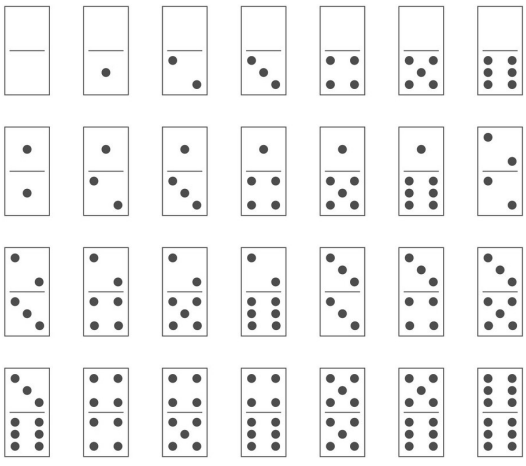
gestita come array parzialmente riempito con buchi.

Scrivere una funzione entra(Automobile a) che mette una automobile nel primo posto libero e una funzione esce(int posizione) che toglie l'auto nella posizione indicata. Gestire in qualche modo i casi di errore.

Scrive un main di prova con dati fissi che prova le funzione e stampa il contenuto del garage dopo ogni operazione.

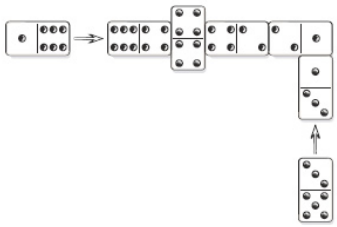
Completare poi la gestione del garage con funzioni che a) conta il numero di auto presenti b) fa uscire tutte le auto

4. **Orario:** definire una struttura per memorizzare un orario (ore, minuti, secondi) e scrivere le funzioni necessarie per assegnare un valore, aggiungere un intervallo di tempo, restituire la differenza, in secondi, tra due orari.
5. **Domino:** Realizzare una struct che rappresenta una tessera del domino. Definire poi un vettore contenente tutte le tessere del domino e realizzare delle funzioni utili a giocare con esse Ad esempio: mescolare il mucchio delle tessere coperte, prenderne (scoprirne) una a caso, calcolare quante ne rimangono coperte, verificare se una tessera può "attaccarsi" con un'altra (cioè almeno uno dei due valori di ciascuna è uguale ad uno dell'altra)...



6. **Gioco del domino:** Realizzare uno dei giochi possibili del domino: vengono messe le tessere, coperte, sul tavolo. Una di esse, a caso, viene scoperta. Via via che il gioco procede le altre tessere potranno essere "attaccate" ad uno dei due lati se corrispondono come valore, allungando la catena. I due giocatori (elaboratore e utente) prendono una tessera a caso tra quelle coperte e cercano di "attaccarla" a una delle due estremità della

catena sul tavolo. Chi non ci riesce prende una penalità e rimette la tessera, di nuovo nascosta, sul tavolo. Se le tessere sono terminate o il gioco è bloccato, vince chi ha meno penalità.



Le regole si possono trovare più in dettaglio, ad esempio, a questo indirizzo *https://www.alexcramer.com/how-to-play* (vengono proposte diverse versioni, tra le quali la nostra).

7. **Orario delle lezioni:** realizzare una struttura dati in grado di memorizzare l'orario delle lezioni di una classe (giorno, ora, materia, docenti). Realizzare un programma a menù in grado di stampare, ad esempio, l'orario di un certo giorno di una classe, oppure quello settimanale di un docente.
8. **Archivio studenti:** Realizzare una funzione che ricerca per numero di matricola gli studenti memorizzati in un array che contiene informazioni relative ad ognuno di essi. Se l'elemento cercato si trova nell'array, la funzione restituisce l'indirizzo della cella corrispondente. Si supponga che l'array su cui si effettua la ricerca sia ordinato in modo crescente. Si definiscano le strutture dati necessarie per lo sviluppo della funzione.

Prototipo della funzione

```
Studente * ricerca(ArchivioStudenti *a, int matr);
```

9. **Registro:** Scrivere una funzione che riceve un vettore con i dati degli alunni di una classe (ogni alunno ha almeno nome, cognome e numero di matricola). Il vettore contiene i dati degli alunni in ordine di iscrizione (e quindi di matricola). La funzione visualizza sullo schermo l'elenco degli alunni, uno per riga, in ordine alfabetico. A inizio di ogni riga c'è la posizione nel registro.
10. **Divisori quadrati:** I divisori di 42 sono: 1, 2, 3, 6, 7, 14, 21, 42. Questi divisori al quadrato sono: 1, 4, 9, 36, 49, 196, 441, 1764.

    La somma di questi divisori al quadrato è 2500, che è 50 * 50, cioè un quadrato! Dati due numeri interi m, n (1 <= m <= n) vogliamo trovare tutti i numeri naturali compresi tra m e n la cui somma dei divisori al quadrato è essa stessa un quadrato. 42 è uno di questi numeri.

    Il risultato deve essere un array di tuple (realizzate con struct) che contiene in ogni elemento: il numero, la lista dei suoi divisori (nel formato di una stringa), la somma dei divisori al quadrato.

    Esempi:

```
listaQuadrati(1, 200) --> [[1, "1", 1], [42, "1, 2, 3, 6, 7, 14, 21, 42", 2500]]
```

11. **Indice:** Definire una struct denominata Prodotto per dei prodotti di profumeria, in cui sono indicati il codice, il nome, il prezzo, la quantità di pezzi in magazzino.

    Sia dato un array di Prodotto, parzialmente riempito con contatore. Realizzare un secondo array che contiene le posizioni dei prodotti nell'array principale in ordine di codice crescente. Scrivere infine una funzione cerca(int codice) che trova il prodotto con il codice fornito e restituisce una copia della struttura o null se non lo ha trovato. Deve utilizzare la ricerca dicotomica sull'indice, con cui va poi a prendere i dati nell'array principale.

12. **Aggiornamento dell'indice:** Sulla struttura dati (array e indice) dell'esercizio precedente scrivere due funzioni per aggiungere un prodotto, per modificare la quantità e per eliminare un prodotto.

13. **Treno:** Definire una struttura in grado di memorizzare le informazioni riguardanti un treno sugli orari delle stazioni (numero, stazione di partenza, stazione di arrivo, ora di partenza e di arrivo, se viaggia tutti i giorni oppure è feriale o festivo...).

    Definire poi una variabile di tipo Treno e riempirla con dei dati a scelta e una struttura dati in grado di memorizzare i dati che partono ogni giorno da Savona (massimo 50 treni).

6.2. Allocazione dinamica

```
void *malloc(size_t size)
```

Alloca size byte nello heap. La memoria non viene inizializzata.

La funzione restituisce il puntatore alla zona di memoria allocata in caso di successo e NULL in caso di fallimento, nel qual caso la variabile di sistema errno assumera' il valore ENOMEM. La funzione strerror(errno) fornisce la descrizione del problema.

```
#include <stdlib.h>
void *calloc(size_t nitems, size_t size)
```

Alloca nitems elementi, ciascuno di size byte nello heap. La memoria viene inizializzata a 0.

La funzione restituisce il puntatore alla zona di memoria allocata in caso di successo e NULL in caso di fallimento, nel qual caso errno assumera' il valore ENOMEM.

```
void *realloc(void *ptr, size_t size)
```

Cambia la dimensione del blocco allocato all'indirizzo ptr portandola a size.

La funzione restituisce il puntatore alla zona di memoria allocata in caso di successo e NULL in caso di fallimento, nel qual caso errno assumera' il valore ENOMEM.

```
void free(void *ptr)
```

Disalloca lo spazio di memoria puntato da ptr.

La funzione non ritorna nulla e non riporta errori.

1. **Vettori allocati dinamicamente:** Scrivere una funzione che a seconda di una lettera che identifica il tipo (i=int, f= float, c=char, s=stringa) e un numero che rappresenta la dimensione, crea un vettore del tipo scelto e della dimensione proposta e lo riempie di

altrettanti valori casuali (per le stringhe mette parole con un numero casuale di lettere alfabetiche casuali compreso tra 5 e 10).

2. **Aritmetica dei puntatori:** Usando l'aritmetica dei puntatori per scorrere il vettore, dati un vettore di float e la sua dimensione cambiare ogni elemento negativo con il valore 0. Restituisce il numero di elementi cambiati.
3. **Puntatori:** Si dichiari in C un vettore di 100 numeri reali (float), di nome dati. Si dichiarino successivamente 3 variabili puntatore, di nome p0, p1, p2, da inizializzare (direttamente nella dichiarazione), con i valori dei puntatori alla prima, all'ultima e alla casella di indice 50 nel vettore. Riempire il vettore con i numeri interi a partire da 63 in avanti e, utilizzando i puntatori, stampare il valore contenuto nelle celle puntate. Scrivere poi un ciclo che usi il primo puntatore (p0) per stampare tutti gli elementi del vettore, incrementando il valore di p0 ogni volta. Quale sarà il valore di p0 al termine del ciclo?
4. **Reciproci:** Scrivere una funzione che crea un vettore di numeri float della dimensione passata in input e vi inserisce i valori dei reciproci dei valori naturali a partire da 1 (1, 1/2, 1/3, 1/4...), fino a riempirlo.
5. **Sottostringa:** Scrivere una funzione

   ```
   char * trovaESostituisci(const char *s1, const char * s2, const *s3)
   ```

   che cerca in una stringa una sottostringa e costruisce una nuova stringa in cui ogni occorrenza di s2 in s1 è sostituita con s3. Se si volesse cambiare il valore direttamente in s1 come si potrebbe procedere? Fare una analisi dettagliata.
6. **Array infinito:** realizzare un array di dimensioni virtualmente infinite: quando viene aggiunto un elemento si verifica che non sia già pieno. Se cose fosse, si crea un nuovo array di dimensione maggiore (ad esempio di 10 elementi più grande), si ricopiano tutti gli elementi presenti nel primo vettore nel secondo, si scambiano i puntatori, si dealloca il vecchio array e, infine, si aggiunge il nuovo elemento al vettore. Inglobare tutto l'algoritmo in una funzione add() con opportuni parametri e eventuale valore di ritorno.

   Verificare approfonditamente che il meccanismo funzioni in tutte le situazioni. Ad esempio: se il primo vettore NON è stato allocato dinamicamente funziona?
7. **A rovescio:** Scrivere due versioni di un programma che legge una sequenza di interi inserita dall'utente e la stampa al contrario, allocando la memoria necessaria in modo dinamico attraverso l'uso della funzione malloc.

   a) L'input è dato da un intero n e da una sequenza di n numeri; basta una sola chiamata di malloc per allocare un vettore di dimensione n.

   b) L'input è dato da una sequenza di numeri terminata da 0; non potendo prevedere quanti numeri verranno inseriti, il vettore andrà ridimensionato man mano: partire da una dimensione fissata piccola (es: 5) e raddoppiare la lunghezza del vettore ogni volta che questo si riempie.
8. **Costruttore:** Data una struct Ora, realizzare una funzione costruisciOra che, forniti i valori di ore, minuti e secondi crea una struct Ora e ne restituisce l'indirizzo dopo aver inserito in essa i valori passati.

Realizzare una seconda funzione costruisceOraVuota restituisce una struct Ora con valori di ora, minuti e secondi pari a zero.

9. **Lettore da terminale:** Scrivere due funzioni che leggano da standard input una sequenza di caratteri e la memorizzino in una stringa di dimensione opportuna allocata dinamicamente:

   a) char *read_line(void) deve leggere una riga terminata da \n; b) char *read_word(void) deve leggere una parola di caratteri alfanumerici (nota:

   se il primo carattere letto non è alfanumerico, la stringa restituita sarà la stringa vuota).

   Entrambe le funzioni devono restituire l'indirizzo del primo carattere della stringa memorizzata o il puntatore NULL in caso di errore.

   Attenzione: la stringa dovrà essere della minima dimensione possibile, cioè un carattere in più di quelli contenuti nella stringa.

10. **Matrice 2D:** Si scriva una funzione malloc2d, in grado di allocare una matrice rettangolare di numeri reali (tipo float), le cui dimensioni sono ricevute come parametri. La matrice viene inizializzata azzerando tutte le caselle. Scrivere poi una funzione print2d che stampa il contenuto di una qualunque matrice 2D di numeri reali (Sono necessari dei parametri? Se sì, quali?).

11. **Codifica a spirale:** La codifica a spirale di una stringa si ottiene con i seguenti passaggi: . Si forma un quadrato abbastanza grande da contenere tutti i caratteri di stringa.

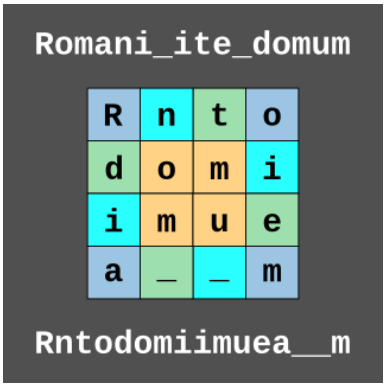
    . A partire dall'angolo in alto a sinistra, si posizionano i caratteri della stringa nelle celle degli angoli, muovendosi in senso orario.

    . Al termine del primo ciclo, si continua a posizionare i caratteri nelle celle nella posizione successiva nella rispettiva riga / colonna.

    . Quando le celle della cornice esterna sono piene, si ripete con la cornice immediatamente successiva (vedere l'esempio nella figura seguente per ulteriori chiarimenti).

    . Tutte le celle non utilizzate sono riempite con un carattere spazio.

    La frase codificata è costitituita dall'unione delle righe.



Input Una stringa composta da qualsiasi combinazione di caratteri alfabetici, il carattere spazio e uno dei seguenti caratteri _!@#$%^&()[]{}+-*/='"<>,.?:;.

Gli argomenti passati al metodo non avranno mai spazi finali.

Ouput Il metodo restituisce il messaggio codificato come stringa senza gli spazi finali.

Suggerimenti si possono usare funzioni di allocazione dinamica della matrice e della stringa risultato.

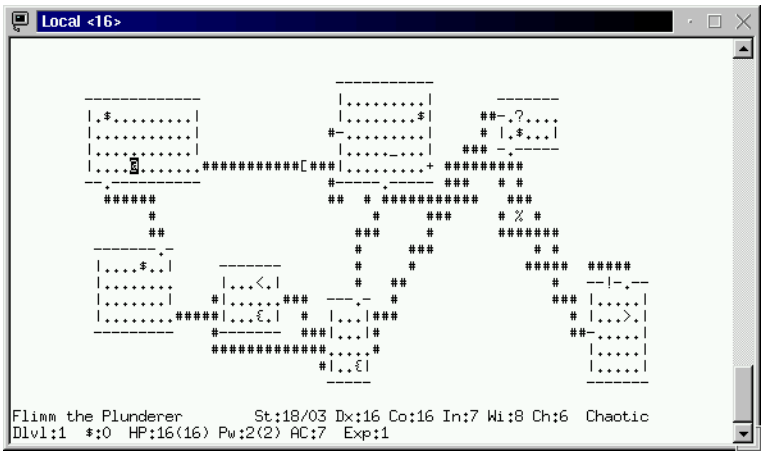
Esempi di test

```
   char s[] = "Romani ite domum";
   char * ris;
   ris=codifica(s); // "Rntodomiimuea  m"

   char s2[] = "Stsgiriuar i ninmd l otac";
   ris = codifica(s2); // "Sic transit gloria mundi"

   /* Encoding sequence for a 5 x 5 square:
   [ 1  5  9 13  2]
   [16 17 21 18  6]
   [12 24 25 22 10]
   [ 8 20 23 19 14]
   [ 4 15 11  7  3]
*/
```

(Fonte *https://www.codewars.com/kata/5a24a35a837545ab04001614*)

6.3. Esercizi di riepilogo

1. **Gioco di ruolo:** Realizzare un gioco di ruolo in cui i diversi personaggi si muovono in un labirinto (cercare rogue o nethack su intenet per degli esempi).



Una schermata di NetHack

2. **Briscola:** Realizzare un programma che consenta di giocare a briscola con il calcolatore.
3. **Battaglia navale - refactoring:** Migliorare il gioco della battaglia navale realizzato con gli array multidimensionali. Realizzare il gioco della battaglia navale utilizzando, ad esempio, le strutture Nave, CampoDiBattaglia, Giocatore.
4. **Stazione Metereologica:** Una stazione metereologica londinese rileva, per ogni giorno dell'anno, le seguenti informazioni:

   Temperatura minima, media e massima; Indice di inquinamento (misurato come mg di piombo e altre sostanze nocive in un metro cubo di aria); Umidità percentuale; Velocità del vento; Direzione del vento; Precipitazioni (in mm di acqua); Tipo di precipitazione (RAIN, SNOW, SLEET, HAIL, FREEZING RAIN); Acidita' della precipitazione (in pratica il ph).

Dopo aver immaginato un sistema software per la memorizzazione e la elaborazione su PC dei dati rilevati, si risponda ai seguenti punti:

a) Descrivere, con una struct di nome 'Weather', come sono tenuti in memoria i dati relativi ad un dato giorno.

b) Scrivere una funzione che permette di stampare in forma tabellare le informazioni contenute in una variabile di tipo Weather.

c) Si immagini di voler effettuare una elaborazione su tutti i dati di un anno. E' possibile portare tutti i dati nella memoria principale? (giustificare la risposta).

In ogni caso fornire qualche dettaglio su come potrebbe essere condotta una elaborazione di questo tipo (Mauro Salvemini, 1991).

7. Files

Argomenti:

- **File binari:** funzioni di utilità, ricerca, aggiunta, cancellazione di valori.
- **File di testo:** confronto con i file binari per la memorizzazione di valori numerici; frequenza delle lettere e delle parole.

Esercizi di base

1. **Dimensione di un file:** Realizzare una funzione che dato il nome di un file restituisce la dimensione in byte.
2. **Informazioni sui file:** Realizzare una funzione che dato il nome di un file stampa le sue caratteristiche: se esiste, nome, percorso assoluto (path), dimensione, se è una direcotry; se è una directory la lista dei file (e directory) che contiene.
3. **Dimensioni su disco:** Scrivere un programma che calcola la dimensione su disco della directory corrente. Se viene passato un argomento al programma calcola la dimensione su disco della directory (se esiste) il cui nome è l'argomento passato; con un ulteriore argomento "-r" prima della directory calcola la dimensione anche di tutte le sottodirectory; con l'opzione "-h" stampa un help su come adoperare il programma e il suo scopo.
4. **Tree:** Realizzare un programma che visualizza tutti i file contenuti nella directory corrente e di tutte le sue sottodir; una versione più avanzata accetta su linea di comando a directory di cui si vuole estrarre la lista.
5. **Search:** Scrivere un programma che cerca un file in un disco, con eventuale selezione della directory di partenza (i.e. "C:\\Utenti").
6. **File binari:** Realizzare un programa che consente due operazioni: a) scrive su un file binario "interi.dat" una sequenza di numeri interi (ad esempio i numeri da 1 a 100, oppure 100 numeri casuali); b) legge lo stesso file visualizzando sul monitor i numeri letti, uno per riga, e al termine il numero di elementi letti. Se il file non è stato riempito l'operazione di lettura indica "file vuoto".
7. **File binari con numeri reali:** Duplicare l'esercizio precedente utilizzando numeri reali anzichè interi. La dimensione del file cambia? Perché?
8. **Ricerca valore:** Realizzare una funzione che cerca un valore in un file contenente un elenco di numeri reali. La funzione restituisce la posizione del valore (0 se nella prima posizione, 1 se nella seconda e così via), oppure -1 se il valore non è presente nel file.
9. **Aggiunta di un valore:** Realizzare una funzione che cerca un valore in un file contenente un elenco di numeri reali e, se non viene trovato, lo aggiunge in fondo al file. La funzione restituisce 0 se il valore è stato aggiunto o -1 altrimenti.
10. **Cancellazione di valore:** Realizzare una funzione che cerca un valore in un file contenente un elenco di numeri reali e, una volta trovato, lo elimina dal file. La funzione restituisce 0 se il valore è stato tolto o -1 altrimenti.

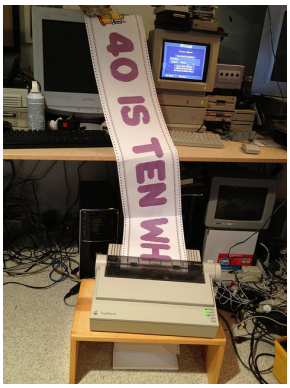
SUGGERIMENTO: forse occorre riscrivere completamente il file.

11. **File di testo:** Realizzare un programa che consente due operazioni: a) scrive su un file di testo "interi.txt" un l'elenco di valori interi (casuali, oppure letti da input), separati tra loro da uno spazio; b) legge lo stesso file visualizzando sul monitor una tabella di due colonne, con intestazione "pari" e "dispari", contenente i numeri letti, uno per riga, nella colonna corretta. Sotto la tabella indica il numero di elementi letti.
12. **File di testo e file binari:** Confrontare le dimensioni tra un file binario contenente un dato numero di valori interi e un file di testo in cui gli stessi numeri sono memorizzati separati da uno spazio. Quale dei due è più grande? Cambiando solo i valori memorizzati le dimensioni dei file cambiano a loro volta?
13. **Parole in un file:** Realizzare una funzione che dato il nome di un file di testo conta il numero di parole presenti nel file.
14. **Top Ten:** Molti giochi on line consentono di indicare il proprio nome e al termine di ogni partita visualizzano i 10 migliori punteggi ottenuti con vicino il nome del giocatore. La "top ten" è solitamente memorizzata in un file. Fare una analisi approfondita del problema e presentare due soluzioni: una che prevede l'uso di file binari e una che prevede l'uso di un file di testo. Realizzarle entrambe e discutere sui pregi e i difetti di ognuna di esse.
15. **Frequenze delle lettere:** Scrivere un programma che riceve in input un file di testo e conta le occorrenze di ognuna delle lettere dell'alfabeto. Alla fine mostra due tabelle con tre colonne: lettera, frequenza assoluta e frequenza relativa (numero di occorrenze sul totale delle lettere). Le due tabelle sono in ordine alfabetico la prima e in ordine di frequenza la seconda.
16. **Conta occorrenze:** Scrivere un programma che riceve in input un file di testo e costruisce un secondo file che contiene tutte le parole presenti nel primo con vicino il numero di occorrenze di ognuna di esse.

Esercizi complessi

17. **Prototipatore:** L'applicazione legge un file sorgente scritto in linguaggio C e produce un file con lo stesso nome e suffisso ".h" contenente i prototipi di tutte le funzioni trovate nel file sorgente.
18. **Documentatore:** L'applicazione legge un file sorgente scritto in linguaggio C e produce un file con lo stesso nome e suffisso ".html" contenente i commenti che precedono ciascuna funzione e il loro prototipo in modo che il file visualizzato da un browser sia di aspetto gradevole. Supporre che i commenti che precedono le funzioni comincino con "/**" invece che con "/*".

    Se una funzione non ha commento iniziale viene indicato il tipo ritornato seguito dal nome della funzione e poi i nomi dei parametri e il loro tipo, uno per riga.
19. **Banner:** Negli anni '80 era frequente realizzare dei banner cartacei con la carta a modulo continuo delle stampanti ad aghi. Per fare ciò, scelta la frase da stampare, ogni lettera veniva stampata ruotata e molto in grande utilizzando più volte la stessa lettera.



Realizzare un programma che consente all'utente di inserire il testo e scrive su un file il banner, supponendo siano disponibili 80 caratteri per ogni riga (cosa comune a quei tempi).

```
  AAA    SSSSS   CCCCC  IIIII IIIII       TTTTTTT              tt
 AAAAA  SS      CC    C  III   III           TTT     eee  xx  xx tt
AA   AA  SSSSS  CC       III   III           TTT   ee   e   xx   tttt
AAAAAAA      SS CC    C  III   III           TTT   eeeee     xx   tt
AA   AA  SSSSS   CCCCC  IIIII IIIII          TTT    eeeee xx  xx  tttt
```

20. **Generatore di parole con i numeri telefonici:** Le tastiere telefoniche standard dei telefoni cellulari contengono le cifre da zero a nove. A ognuna delle cifre da 2 a 9 sono associate tre o quattro lettere (vedi tabella).

```
2    A B C        6    M N O
3    D E F        7    P Q R S
4    G H I        8    T U V
5    J K L        9    W X Y Z
```

Per molte persone la memorizzazione dei numeri di telefono è una operazione difficile, per cui spesso utilizzano questa corrispondenza per sviluppare parole di sette caratteri che corrispondano ai propri numeri telefonici (ciò vale negli Stati Uniti, ma può essere adattato anche da noi). Per esempio, una persona con numero di telefono 468-6734 potrebbe usare la tabella e individuare la parola "INUMERI".

Le aziende cercano per questo di avere numeri di telefono, dalle ditte di telefonia, che possano essere facilmente ricordati dai propri clienti. Scrivere un programma che, dato un numero di telefono a sette cifre, scriva su un file tutte le possibili combinazioni di parole ottenibili con quel numero. Al massimo sono 16384 (perché?).

Evitare i numeri di telefono con le cifre zero e uno.

21. **Generatore versione plus:** Usando un dizionario completo, modificare l'esercizio precedente in modo che mostri solo le parole presenti nel dizionario.

22. **Analizzatore di file:** Scrivere un programma che estragga da un file HTML tutti i tag e li mostri sull'output. Un tag inizia con '<' e termina con '>'.

23. **Superfici e volumi:** Si desidera sviluppare un programma per il calcolo delle superfici e dei volumi di un edificio.

Il programma riceve sulla riga di comando due parametri: il primo è il nome del file che contiene le dimensioni dell'edificio mentre il secondo è il numero di piani di cui è composto l'edificio.

La struttura dell'edificio è descritta in un file di testo così organizzato: per ogni piano è presente una prima riga contenente due valori interi: il numero di stanze presenti nel piano e l'altezza del piano. Tale riga è seguita da tante righe quante sono le stanze, ognuna contenente due valori che rappresentano le dimensioni della stanza. Tutte le stanze sono di forma rettangolare, tutte le dimensioni sono espresse in centimetri e sono date come numeri interi positivi.

Il programma deve calcolare e presentare sull'unità di output standard:

a) la superficie totale di tutte le stanze dell'edificio, espressa in metri quadri.

b) il volume totale di tutte le stanze dell'edificio, espresso in metri cubi.

Ad esempio, se il programma, supposto chiamarsi dimimmo, venisse attivato con la seguente riga di comando:

dimimmo CASA.TXT 2

(ovvero l'edificio è composto da due piani e le relative dimensioni si trovano nel file CASA.TXT) ed il file CASA.TXT contenesse i seguenti dati:

```
2 300
200 200
200 400
1 200
200 300
```

(ovvero il primo piano è alto 300 cm e consiste di due stanze rispettivamente di 200 cm per 200 cm e 200 cm per 400 cm, mentre il secondo piano è alto 200 cm e consiste di un'unica stanza di 200 cm per 300 cm) allora il programma dovrebbe produrre il seguente output:

```
Superficie totale dell'edificio:   18.00 metri quadri
    Volume totale dell'edificio:   48.00 metri cubi
```

24. **Ricette di Suor Germana:** Suor Germana vuole realizzare una versione elettronica delle sue famose ricette di cucina, sotto forma di un programma. In particolare, si vuole che il programma identifichi quali sono le ricette cucinabili, dato il contenuto attuale del frigorifero di una famiglia.

    Il programma accede a due file:

    1. un file di testo (denominato Germana.txt) contenente gli ingredienti necessari per tutte le ricette di Suor Germana secondo il seguente formato:

    a) ogni riga è nella forma ricetta ingrediente quantità.

    b) ricetta è una stringa (max 20 caratteri, senza spazi) che indica il nome della ricetta.

    c) ingrediente è una stringa (max 20 caratteri, senza spazi) che indica il nome di un ingrediente.

    d) quantità è un numero reale che indica la quantità di tale ingrediente nella ricetta corrispondente.

e) sia ricetta, sia ingrediente sono ripetuti più volte nel file, ma sempre in associazione a ingredienti o ricette diversi.

f) non è noto a priori il numero di righe del file, né è specificato alcun ordinamento noto per il file.

2. un file di testo (il cui nome è passato come primo parametro sulla linea di comando) rappresentante il contenuto attuale del frigorifero secondo il seguente formato:

ogni riga è nella forma ingrediente quantità a) ingrediente corrisponde ad uno degli ingredienti presenti nel file delle ricette.

b) quantità è un numero reale che identifica la quantità presente di tale ingrediente nel frigorifero.

c) ogni ingrediente è presente una sola volta in questo file.

d) non è noto a priori il numero di righe del file, né è specificato alcun ordinamento noto per il file.

Il programma riceve come argomenti sulla linea di comando il nome del file contenente le disponibilità del frigorifero ed il nome di una ricetta, e deve fornire in output l'elenco degli ingredienti della ricetta (con l'indicazione se ciascuno di essi è disponibile o meno) e la conclusione finale se la ricetta scelta può essere preparata.

Ad esempio se i file Germana.txt e frigo.txt contenessero i seguenti dati:

```
(Germana.txt)                          (frigo.txt)
padellino uovo 1                       uovo 1
frittata olio 0.3                      olio 0.5
padellino olio 0.2                     parmigiano 0.1
frittata uovo 1
coque uovo 1
frittata parmigiano 0.2
```

ed il programma (denominato cerca) venisse attivato con la riga di comando;

```
cerca frigo.txt frittata
```

allora dovrebbe produrre il seguente risultato:

```
Ingredienti:
- olio: OK
- uovo: OK
- parmigiano: richiesto 0.2, disponibile 0.1
Ricetta "frittata" impossibile
```

25. **Lunghezze delle parole:** Si scriva un programma in grado di analizzare il contenuto di un file di testo e di calcolare la distribuzione di frequenza della lunghezza delle varie parole in esso contenute.

    Per le finalità del presente programma, si definisce "parola" una sequenza di caratteri alfanumerici.

    Il programma riceve sulla linea di comando il nome del file da analizzare e produce in uscita una tabella con le frequenze, espresse in valore assoluto (non in percentuale).

    Esempio:

Si supponga che il file diario.txt contenga il seguente testo:

```
C'era una volta... "Un Re", diranno i miei piccoli lettori.
No, c'era una volta un pezzo di legno!
```

e che il programma (denominato freqlett.c) venga invocato con il seguente comando:

```
freqlett diario.txt
```

Il programma dovrà produrre in uscita:

```
Frequenza delle lunghezze delle parole
Parole lunghe 1 caratteri:   3
Parole lunghe 2 caratteri:   5
Parole lunghe 3 caratteri:   4
Parole lunghe 4 caratteri:   1
Parole lunghe 5 caratteri:   4
Parole lunghe 7 caratteri:   3
```

Infatti le parole di 1 carattere sono "C i c", quelle di 2 caratteri sono "Un Re No un di", quelle di 3 caratteri sono "era una era una", quelle di 4 caratteri sono "miei", quelle di 5 caratteri sono "volta volta pezzo legno" e quelle di 7 caratteri sono "diranno piccoli lettori".

26. **Codifica di Huffman:** Realizzare un programma che determina la codifica di Huffman (cercare su internet la sua definizione) a partire da un testo sufficientemente lungo (ad esempio il libro "I promessi sposi"). Stampare la codifica di ognuno dei caratteri alfabetici espresso nella codifica, usando 0 e 1.

27. **Tassi di cambio:** Realizzare un programma che sa utilzzabile per analizzare l'andamento dei tassi di cambi di alcune valute. Il programma utilizza dei file di testo (denominati, ad esempio, tassi20170717.txt, tassi20170718.txt etc) contenenti, uno per riga, i tassi di cambio dell'euro con un'altra valuta di cui è fornita la sigla e la descrizione.

Esempio di file dei tassi

```
USD, Dollaro USA, 1.15
YPN, Yen Giapponese, 128.45
DKK, Corona danese, 7.44
```

Il programma deve consentire di visualizzare, ad esempio, la variazione di tasso di una valuta tra due date (in valori assoluti e percentuali), il grafico dei tassi di una valuta, il valore in euro di una somma di denaro di altra valuta con applicato il tasso di cambio più recente.

28. **Albero Genealogico:** Realizzare una applicazione che consente di memorizzare un albero genealogico di una famiglia.

    Il formato del file è il seguente (i nomi delle persone sono sostituiti dalle descrizioni):

```
trisnonno, 1810-1866; trisnonna, 1815-1878
 bisnonno1, 1831-1880; bisnonna1, 1833-1890
  nonna1, 1855-1913; nonno1, 1849-1900
   padre1, 1870-1950; madre1, 1872-1951
    figlio1, 1901-1966
   figlia1, 1903-1977; figlio2, 1901-1980
    nipote1, 1935-2005
```

```
    nipote2, 1936-1945
  nonna2, 1857-1920; nonno2, 1855-1889
 bisnonna2, 1832-1865
 bisnonna3, 1833-1833
 bisnonno4, 1835, 1840
 bisnonno5, 1836-1931; bisnonna4, 1840-1913
  nonno3, 1867-1950, nonna3, 1871-1958
...
```

in cui gli spazi a inizio riga determinao la "generazione" a partire da una coppia di "capostipiti" (nel caso dell'esempio, trisnonno e trisnonna). I coniugi sono indicati a fianco, separati da un punto e virgola, i figli sono a capo, indentati di un carattere in più. Nel caso di più matrimoni si continua sulla stessa riga.

Realizzare una serie di funzioni di utilità che consentano, ad esempio: di caricare tutto l'albero genealogico in memoria RAM; di apportare modifiche alla versione in memoria (aggiunta di un nominativo, modifica di un nominativo...); di salvare la versione in RAM su disco, creando una nuova versione del file; di mostrare a video, creando un opportuno file in formato HTML, per una data persona, le indicazioni della sua famiglia (coniuge ed elenco dei figli) oppure di tutto l'albero genealogico.

Fase successiva: consentire l'aggiunta di informazioni testuali e di una fotografia.

8. Argomenti avanzati

Argomenti:

- **Reti di Petri:**
- **Automi:**
- **Macchina di turing:**

Reti di Petri.

Esempio su http://www.dacrema.com/Informatica/petri_bancomat.htm

1. Studiare una struttura dati per memorizzare una rete di Petri di qualunque dimensione. Definire poi delle funzioni per definire una rete di Petri qualsiasi.
2. Realizzare un simulatore per le reti di Petri.

Esercizi sugli automi.

Per informazioni cercare "Automa a stati finiti".

1. Automa di Mealy

Esercizi su macchine di turing (MdT).

Materiale utile, con sintassi e esercizi svolti, su *http://iissanluri.edu.it/attachments/article/1167/CapitoloTuring.pdf* .

Testi delle gare dell'Università di Pisa *http://mdt.di.unipi.it/TestiGara/IndiceTesti.aspx* .

Simulatore (esempio) *http://kangourou.di.unimi.it/2012/turing/myturing.html*

1. Costruire una MdT che data una stringa scritta con un alfabeto di sole A e B mi faccia comparire una parola di sole A. (cioè le A restano A e le B diventano A)
2. Costruire una MdT che data una stringa di A, B e C. Sostituisca ad ogni A la lettera C, a ogni C la lettera A e le B restino B.
3. Costruire una MdT che data un numero scriva P se è pari e D se è dispari (es: input 123 output 123D)
4. Costruire una MdT che dato un numero mi inverta le cifre 1 con 2 (cioè dove c'è scritto 1 scriverà 2 e dove c'è scritto 2 scriverà 1) e le altre cifre rimarranno invariate.
5. Dato un numero in base dieci si scriva il numero sommato a 2.
6. Dato un numero in base dieci si scriva il numero sommato a 3.
7. Costruire una MdT che data una stringa di A e B mi scriva quante A sono presenti (si suppone che non ci siano più di 5 A)
8. Dato un numero intero sottrarre 1
9. Dato un numero intero moltiplicare per 2
10. Data una stringa di 0 e 1 invertire gli 0 con gli 1 e viceversa

11. Data una stringa di 0 e 1 fare in modo che diventino tutti 1
12. Data una stringa di 0 e 1 cancellare la stringa
13. Data una stringa di 0 e 1 di aggiunga un 1 alla fine della stringa.
14. Dato un numero in base dieci si scriva uno 0 alla fine del numero (input 123 output 1230)
15. Dato un numero in base dieci si cancelli tutto il numero
16. Dato un numero in base dieci si cancelli tutto il numero tranne la prima (input 123 output 1)
17. Dato un numero in base dieci si sommi 1 (input 123 output 124)
18. Dato un numero in base dieci si sommi 2 (input 123 output 125)

9. Algoritmi da conoscere assolutamente

Elenco minimale degli algoritmi significativi da conoscere senza incertezze al termine del corso:

Algoritmi numerici:

1. Scomposizione in fattori primi di un numero intero
2. Fattoriale di un numero (eventualmente anche con versione ricorsiva)
3. Elevamento a potenza intera di un numero reale
4. MCD tra due numeri (con diverse soluzioni)
5. Calcolo del mcm tra due numeri (con diverse soluzioni)
6. Primalità di un numero intero (N.B. 1 non e' considerato numero primo, si parte da due)
7. Stampa dei primi N numeri primi
8. Calcolo dell'n-esimo numero primo
9. Serie di Fibonacci (versione iterativa e versione ricorsiva)
10. Tabellina pitagorica
11. Calcolo della radice quadrata con il metodo dicotomico
12. Scambio del contenuto di due variabili

Argoritmi sui vettori parzialmente ordinati delle diverse tipologie (sentinella, contatore, valore nullo):

1. Copia di un vettore su un altro
2. Ricerca del max/min tra gli elementi
3. Somma degli elementi di un vettore (accumulazione); calcolo della media
4. Aggiunta di un elemento
5. Ricerca di un elemento (sia per vettore ordinato che non ordinato)
6. Calcolo del numero di occorrenze di un elemento
7. Ricerca del valore o dei valori più frequenti (moda)
8. Ordinamento di un vettore (diversi algoritmi)
9. Inserimento di un elemento in un vettore ordinato
10. Ricerca di un sottovettore in un vettore (esempio: ricerca sottostringa in una stringa)
11. Unione di due vettori ordinati
12. Unione di due vettori non ordinati (aggiunta in fondo)
13. Eliminazione di un elemento
14. Eliminazione dei duplicati
15. Intersezione tra due vettori di elementi distinti (insiemi)
16. Unione tra due vettori di elementi distinti (insiemi)
17. Crivello di Eratostene per determinare i numeri primi inferiori a 1000

Algoritmi sulla manipolazione di stringhe:

1. Ricerca di una sottostringa in una stringa
2. Concatenazione di una stringa con un'altra
3. Verificare se una stringa è un anagramma di una seconda
4. Contare le parole presenti in una stringa (separate da uno o più spazi)
5. Ordinare i caratteri in una stringa
6. Sostituire tutti i caratteri di tabulazione ('\t') in una stringa con tre spazi

Strutture dati astratte

1. Realizzare una lista (creazione, aggiunta e rimozione di elementi)
2. Realizzazione di una lista circolare
3. Realizzare una coda
4. Realizzare un albero binario (aggiunta, rimozione, attraversamento)
5. Realizzazione di una mappa

Quesiti simpatici che possono essere richiesti:

1. Scambiare il valore di due variabili intere SENZA usare altre variabili
2. Muretto di mattoni (http://codingbat.com/doc/practice/makebricks-introduction.html)
3. Barretta d cioccolato (http://codingbat.com/doc/practice/makebricks-introduction.html)

10. Schede sintattiche

10.1. Primo programma in C

Struttura di un file sorgente in C

```
/* programma: NomeFile.c
 * autore: NomeAutoreDelProgramma
 * BreveDescrizioneDelProgramma
 */
/* Inclusione delle librerie */
#include <stdio.h>
#include <stdlib.h>
#include <math.h>

int main(void) {
   /* Definizione delle variabili */
   . . .
   /* algoritmo */
   . . .
   return 0;
}
```

Nota:

Quando il programma riceve degli argomenti sulla linea di comando, allora la definizione della funzione main deve essere modificata come:

```
int main(int argc, char* argv[])
```

Librerie principali

```
#include <stdio.h>    // Lettura/scrittura su terminale e su file
#include <stdlib.h>   // Interazione con sistema operativo
#include <math.h>     // Funzioni matematiche
#include <string.h>   // Elaborazione di testi e stringhe
#include <ctype.h>    // Analisi del tipo di caratteri
#include <limits.h>   // Valori minimi e massimi
```

10.2. Variabili e tipi di dato principali

Definizione delle variabili

```
int i, j, numero;              // variabili intere
float raggio, dimensioneLato;  // variabili reali
char car;                      // un carattere in codice ASCII,
                               //    ad esempio 'g'
```

I tipi interi

	<limits.h>		compilatore gcc		
Tipo	**Min**	**Max**	**N.bit**	**Min**	**Max**
char	CHAR_MIN	CHAR_MAX	8	−128	127
int	INT_MIN	INT_MAX	32	−2 147 483 648	2 147 483 647
short int	SHRT_MIN	SHRT_MAX	16	−32 768	32 767
long int	LONG_MIN	LONG_MAX	32	−2 147 483 648	2 147 483 647
unsigned int	0	UINT_MAX	32	0	4 294 967 295
unsigned short int	0	USHRT_MAX	16	0	65 535
unsigned long int	0	ULONG_MAX	32	0	4 294 967 295

I tipi reali

Tipo	**N.bit**	**Mantissa**	**Esponente**	**Min/Max**	**Epsilon**
float	32	23 bit	8 bit	$\pm 3.402 \cdot 10^{+38}$	$\pm 1.175 \cdot 10^{-38}$
double	64	53 bit	10 bit	$\pm 1.797 \cdot 10^{+308}$	$\pm 2.225 \cdot 10^{-308}$

Conversioni di tipo esplicite

Tra tipi scalari	`(nuovotipo)espressione`

Da stringa a numero	`gets(line) ;`
	`x = atoi(line) ; /* int */`
	`x = atol(line) ; /* long */`
	`x = atof(line) ; /* float o double */`

Istruzioni di base

Assegnazione

```
a = 87;
c = a+b;
b = b + 65;    // oppure: b += 65;
c = c + 1;     // oppure c += 1; oppure c++;
e = (b * b - 4) * k + 4 * a;
x1 = ( -b + sqrt(delta) ) / (2 * a );
nomeVariabile = funzioneQualunque(valore1, 24) ;
```

Lettura da input di numeri interi e reali

```
scanf("%d", &c);
scanf("%f", &numero);
```

Scrittura su output di numeri interi, reali

```
printf("Il risultato vale %d", ris);
printf("area = %f", area);
printf("area = %6.2f", area); // 5.658 diventa ..5.66 (due spazi iniziali)
printf("\n"); // va a capo
```

Espressioni aritmetiche

```
+ - * /            // le 4 operazioni
%                  //  modulo della divisione tra interi
((a+b)*(c/(d-e)))  // espressioni
(int)x             // valore intero della variabile reale x
(float)n           // valore reale della variabile intera n
```

Funzioni definite in math.h

Funzione	Descrizione	Esempio
`sqrt(x)`	radice quadrata di *x*	`sqrt(900.0)` è `30,0` `sqrt(9.0)` è `3,0`
`exp(x)`	funzione esponenziale *e*	`exp(1.0)` è `2,718282` `exp(2.0)` è `7,389056`
`log(x)`	logaritmo naturale di *x* (in base *e*)	`log(2.718282)` è `1,0` `log(7.389056)` è `2,0`

`log10(x)`	logaritmo di *x* (in base 10)	`log10(1.0)` è `0,0`
		`log10(10.0)` è `1,0`
		`log10(100.0)` è `2,0`

`fabs(x)`	valore assoluto di *x*	`fabs(5.0)` è `5.0`
		`fabs(0.0)` è `0,0`
		`fabs(-5.0)` è `-5,0`

`ceil(x)`	arrotonda *x* all'intero più piccolo non minore di *x*	`ceil(9.2)` è `10,0` `ceil(-9.8)` è `-9,0`
`floor(x)`	arrotonda *x* all'intero più grande non maggiore di *x*	`floor(9.2)` è `9,0` `floor(-9.8)` è `-10,0`
`pow(x, y)`	*x* elevato alla potenza *y* ()	`pow(2, 7)` è `128,0` `pow(9, .5)` è `3,0`
`fmod(x, y)`	resto di *x*/*y* in virgola mobile	`fmod(13.657, 2.333)` è `1,992`
`sin(x)`	seno trigonometrico di *x* (*x* è espressa in radianti)	`sin(0.0)` è `0,0`
`cos(x)`	coseno trigonometrico di *x* (*x* è espressa in radianti)	`cos(0.0)` è `1,0`
`tan(x)`	tangente trigonometrica di *x* (*x* è espressa in radianti)	`tan(0.0)` è `0,0`

Variabili di tipo carattere

```
char c, lettera;   // definizione
c = 'h';           // assegnazione, codice ASCII
lettera = 72;      // codice ASCII di 'h'
scanf("%c", &c);   // lettura da input, richiede Enter
lettera = getch(); // lettura da input
printf("%c", c);   // stampa
putc(lettera);     // stampa
```

`'\n'`	A capo
`'\t'`	Tabulazione
`'\b'`	Backspace (cancella ultimo carattere)
`'\a'`	Campanello (*alert*)
`'\r'`	Ritorno carrello sulla stessa riga
`'\\'`	Carattere di *backslash* \
`'\''`	Carattere di singolo apice '
`'\"'`	Carattere di doppio apice "
`'\xNN'`	Carattere il cui codice ASCII vale `NN` (in base 16)

Funzioni della libreria `<ctype.h>`

Nome	Parametri	Restituisce	Descrizione	Esempi
`isalpha`	`char ch`	vero/falso	Lettera maiuscola o minuscola (`A...Z`, `a...z`)	`if(isalpha(ch)) { ... }`
`isupper`	`char ch`	vero/falso	Lettera maiuscola (`A...Z`)	`if(isupper(ch)) { ... }`
`islower`	`char ch`	vero/falso	Lettera minuscola (`a...z`)	`if(islower(ch)) { ... }`
`isdigit`	`char ch`	vero/falso	Cifra numerica (`0...9`)	`if(isdigit(ch)) { ... }`
`isalnum`	`char ch`	vero/falso	Lettera oppure cifra numerica: `isalpha(ch)||isdigit(ch)`	`if(isalnum(ch)) { ... }`
`isxdigit`	`char ch`	vero/falso	Cifra numerica oppure lettera valida in base 16 (`a...f`, `A...F`)	`if(isxdigit(ch)) { ... }`
`ispunct`	`char ch`	vero/falso	Simbolo di punteggiatura (`!"#$%&'()*+,-./:;<=>?@[\]^_'{|}~`)	`if(ispunct(ch)) { ... }`
`isgraph`	`char ch`	vero/falso	Qualsiasi simbolo visibile (lettera, cifra, punteggiatura)	`if(isgraph(ch)) { ... }`
`isprint`	`char ch`	vero/falso	Qualsiasi simbolo visibile o spazio	`if(isprint(ch)) { ... }`
`isspace`	`char ch`	vero/falso	Spazio, tab o a capo	`if(isspace(ch)) { ... }`
`iscntrl`	`char ch`	vero/falso	Qualsiasi carattere di controllo	`if(iscntrl(ch)) { ... }`
`toupper`	`char ch`	`char`	Ritorna la versione maiuscola di `ch`	`for(i=0; s[i]!=0; i++) s[i] = toupper(s[i]) ;`
`iscntrl`	`char ch`	`char`	Ritorna la versione minuscola di `ch`	`for(i=0; s[i]!=0; i++) s[i] = tolower(s[i]) ;`

Espressione condizionali

- **Confronto di uguaglianza:** == (ATTENZIONE: mai usare =)
- **Confronto di disuguaglianza:** !=
- **Confronto di ordine:** < <= > >=
- **Congiunzione AND:** (a>0) && (b>0)

- **Disgiunzione OR:** (a>0) || (b>0)
- **Negazione NOT:** !(a+b < c)
- **Appartenenza ad intervalli:** x compreso tra a e b, estremi inclusi (x in [a, b]):

  a<=x && x<=b

  (NOTA: mai usare a<=x<=b)
- **Esclusione da intervalli:** x non compreso tra a e b (x NOT in [a, b]):

  x<a || x>b

  oppure

  !(a<=x && x<=b)

Costanti

define

```
#define MAX 100
```

il **precompilatore** sostituisce in ogni punto del file le occorrenze di MAX con 100.

- Solitamente in cima al file, subito dopo i #include
- senza =
- senza ;
- senza tipo di dato
- #define parte dalla prima colonna (non ci possono essere spazi)

const

```
const int MAX = 100 ;
```

E' una normale variabile il cui contenuto non è modificabile.

- Solitamente dentro il main o la funzione che la adopera.
- necessario =, il ; e la specificazione del tipo (int, float...)

10.3. Controllo del flusso

Istruzioni per il test (scelta)

- **if con else:**

```
if (condizione) {
   istruzioni 1 ;
}else{
   istruzioni 2 ;
}
```

- **if senza alternativa else:**

```
if (condizione) {
   istruzioni 1 ;
}
```

- **if annidati:**

```
if (condizione) {
   istruzioni 1 ;
   if (condizione 2){
      istruzioni 3;
   }
}else{
   istruzioni 4;
   if (condizione) {
      istruzioni 5 ;
   }else{
      istruzioni 6 ;
   }
}
```

- **scelta multipla con if:**

```
if (condizione) {
   istruzioni 1 ;
}else if (condizione2){
   istruzioni 2 ;
}else if (condizione3){
   istruzioni 3;
}else{
   istruzioni 4;
}
```

- **scelta multipla con switch case:**

```
switch (espressione) {
   case 2:
      istruzioni 1 ;
      break ;
   case 20:
      istruzioni 2 ;
      break ;
   case 210:
      istruzioni 3 ;
      break ;
   default:
      istruzioni di default ;
}
```

NOTA: se si dimentica il break vengono eseguite le istruzioni del case successivo.

NOTA: l'espressione deve fornire un intero o un carattere. Per i caratteri nel "case" si indica il valore tra apici (ad esempio: case 'd':).

Cicli

Struttura di un ciclo

1. Inizializzazione. Assegnazione del valore iniziale a tutte le variabili che vengono lette durante il ciclo (nella condizione o nel corpo).
2. Condizione di ripetizione. Condizione, di solito inizialmente vera, che al termine del ciclo diventerà falsa. Deve dipendere da variabili che saranno modificate all'interno del ciclo (nel corpo o nell'aggiornamento).
3. Corpo del ciclo. Le istruzioni che effettivamente occorre ripetere: sono lo scopo per cui il ciclo viene realizzato. Si possono usare e modificare le variabili inizializzate.

4. Aggiornamento. Modifica di una o più variabili in grado di aggiornare il valore della condizione di ripetizione (rendendola, prima o poi, falsa). Tengono "traccia" del progresso dell'iterazione.

Costrutti per i cicli

- **while:**

```
while(condizione) {
   corpo ;
}
```

- **for:** Equivale in tutto e per tutto ad un while

```
for(inizializzazione; condizione; incremento) {
   corpo ;
}
```

Equivale a:

```
inizializzazione;
while(condizione) {
   corpo;
  incremento;
}
```

- **do... while:**

```
do{
   corpo ;
}while(condizione) {
```

Istruzioni break e continue

break

```
while ( condizione ) {
  Blocco1 ;
  if ( condizioneUscita ) /* condizione uscita */
    break ;
  Blocco2 ;
}
/* se condizioneUscita e' vera, salta immediatamente qui,
ed interrompe il ciclo anche se condizione e' ancora vera.
In tal caso, Blocco2 non viene eseguito. */
```

continue

```
while ( condizione ) {
   Blocco1 ;
   if ( condizioneUscita )
     continue ;
   Blocco2 ;
   /* se condizioneUscita e' vera, salta immediatamente qui,
      poi riprende la prossima iterazione. In tal caso,
      Blocco2 non viene eseguito. */
}
```

10.4. Funzioni

Definizione di funzioni

- **Prototipo:** dichiarazione del nome, dell'interfaccia e del tipo delle variabili. Ricorda: finisce con un punto-e-virgola!

```
int leggi(int min, int max) ;
```

- **Definizione:** dichiarazione dell'interfaccia e definizione del corpo effettivo della funzione. Nessun punto-e-virgola! Ricorda: è necessaria l'istruzione return.

```
int leggi(int min, int max){
   int val ;
   /* ... codice del corpo della funzione ... */
   return val ;
}
```

- **Chiamata:** utilizzo della funzione all'interno di un'altra funzione

```
int main() {
  int x, a, b ;
  . . .
  x = leggi(a, b) ;
  . . .
}
```

Parametri delle funzioni

```
//Tipi scalari, passati by value
int funz(int x, double f);

// Vettori, passati by reference
int funz(int v[]);

// Tipi scalari, passati by reference
int funz(int *x, double *f) {
  *x = 1 ;
  *f = 2.3 ;
}
. . .
funz( &intero, &reale);
```

10.5. Vettori e stringhe

Definizione

```
int valori[100];
```

dove:

- **int:** tipo di dato dei valori contenuti nel vettore. Puoò essere uno qualunque dei tipi predefiniti o anche dei tip definiti dall'utente (struct ...)
- **valori:** nome arbitrario. E' anche l'indirizzo (riferimento) del primo elemento.
- **100:** dimensione del vettore. Solitamente si preferisce utilizzare una costante:

```
#define DIM 100
...
```

```
int valori[DIM];
```

- l'indice va da 0 alla dimensione -1. Nel caso dell'esempio sono validi

```
valori[0] = 7;              // primo elemento
k = 1;
valori[k] = -45;            // secondo elemento
valori[DIM-1] = valori[0];  // copia il primo nell'ultimo elemento
for(i=0; i<DIM; i++)
   valori[i] = 0;           // inserisce 0 in tutte le posizioni
```

Vettori parzialmente riempiti Non tutte le posizioni contengono dati significativi.

- con terminatore: ad esempio il '/0' nelle stringhe;
- con contatore: solo le prime posizioni sono occupate. Quante è indicato da una variabile.

```
int valori[DIM], scelta;
int numValori = 0;

do{
   printf("valore da inserire [0 per terminare]:");
   scanf("%d", &scelta);
   if (scelta!=0){
      valori[numValori++]=scelta;
  }
}while(scelta !=0);
```

- con "buchi": serve un valore neutro (ad esempio 0 oppure -999). Gli elementi con il valore neutro non sono considerati, gli altri hanno valori significativi.

Stringhe

Definizione

```
char s[SIZE];             // SIZE-1 caratteri al massimo
char s[]="Ciao a tutti";  // 12 caratteri al massimo
```

Assegnazione

```
strcpy(s, "ciao") ;
strcpy(s, s2) ;
```

Lettura

```
scanf("%s", s);   // NON USARE: comportamenti strani
gets(s);
```

Stampa

```
printf("%s", s);
printf("%20s", s);      // in 20 caratteri se più corta, allineata a destra
printf("%-20.20s", s);  // usa non meno e non più di 20 caratteri,
                        // allineata a sinistra
puts(s) ;
```

Funzioni della libreria `<string.h>`

Nome	Parametri	Restituisce	Descrizione	Esempi
`strlen`	`char s[N]`	`int`	Lunghezza della stringa	`lun = strlen(s) ;`

`strcpy`	`char dst[N],`
	`char src[M]`
			Copia il contenuto di `src` all'interno di `dst`	`strcpy(s1, s2) ;`
				`strcpy(s, "") ;`
				`strcpy(s1, "ciao") ;`

`strncpy`	`char dst[N],`
	`char src[M],`
	`int nc`
			Copia il contenuto di `src` (max `nc` caratteri) all'interno di `dst`	`strncpy(s1, s2, 20) ;`
				`strncpy(s1, s2, MAX) ;`

`strcat`	`char dst[N],` `char src[N]`		Accoda il contenuto di `src` alla fine di `dst`	`strcat(s1, s2) ;` `strcat(s1, "␣") ;`

`strncat`	`char dst[N],`
	`char src[M],`
	`int nc`
			Accoda il contenuto di `src` (max `nc` caratteri) alla fine di `dst`	`strncat(s1, s2, 50) ;`

`strcmp`	`char s1[N],` `char s2[M]`	`int`	Risultato `<0` se `s1` precede `s2`, `==0` se `s1` è uguale a `s2`, `>0` se `s1` segue `s2`	`if(strcmp(s, r)==0)` `while(strcmp(r,"*")!=0)`

`strncmp`	`char s1[N],`
	`char s2[M],`
	`int n`
		`int`	Come `strcmp`, ma confronta solo i primi `n` caratteri	`if(strncmp(r,`
				`"buon", 4)==0)`

`strchr`	`char s[N],`
	`char ch`
		`==NULL`
		o
		`!=NULL`
			Risultato `!=NULL` se il carattere `ch` compare nella stringa, `==NULL` se non compare.	`if(strchr(s, '.')!=NULL)`
				`if(strchr(s, ch)==NULL)`

`strstr`	`char s[N],`
	`char r[N]`
		`==NULL`
		o
		`!=NULL`
			Risultato `!=NULL` se la sotto-stringa `r` compare nella stringa `s`, `==NULL` se non compare.	`if(strstr(s, "xy")!=NULL)`
				`if(strstr(s, s1)==NULL)`

`strspn`	`char s[N],` `char r[N]`	`int`	Restituisce la lunghezza della parte iniziale di `s` che è composta esclusivamente dei caratteri presenti in `r` (in qualsiasi ordine).	`lun = strspn(s, "␣") ;` `lun = strspn(s, ".,") ;`
`strcspn`	`char s[N],` `char r[N]`	`int`	Restituisce la lunghezza della parte iniziale di `s` che è composta esclusivamente dei caratteri *non* presenti in `r`.	`lun = strspn(s, "␣") ;` `lun = strspn(s, ".,") ;`

10.6. Struct

Definizione di un nuovo tipo di dati e di variabili di quel tipo:

```
typedef struct Libro {
   char titolo[50];
   char * autore;  // puntatore
   int pagine;
   float prezzo;
} Libro;

Libro testo = { "il nome della Rosa", "Umberto Eco", 356, 9.80};
```

Accesso ai campi della struttura (notazione punto):

```
  testo.pagine = 358;
  strcpy(testo.titolo, "Il nome della Rosa");
  testo.autore = "Umberto Eco";
```

Assegnazione di una struttura ad un'altra:

```
  Libro testo = { "Il nome della rosa", "Umberto Eco", 356, 9.80};
  Libro testoCopia;
  testoCopia = testo;
  testoCopia.prezzo = 8.00; // versione scontata.
```

Il passaggio delle strutture alle funzioni come parametro avviene per VALORE (cioè viene fatta una copia della struct); altrimenti usare i puntatori.

```
  void aumentaPrezzo(Libro * l){
     (*l).prezzo = (*l).prezzo*2.0; // aumenta di 2 Euro
    // alternativa (notazione "freccia")
    l->prezzo = l->prezzo +2.0;
  }
```

E' possibile restituire una struttura:

```
  Libro creaLibro(char titolo[], char autore[], int numpag){
     Libro l;
    strcpy(l.titolo, titolo;
    l.autore = autore;
    l.pagine = numpag;
    l.prezzo = 10.0;
    return l;
  }

  main(){
    Libro testo;
   testo = creaLibro("Avere o essere?", "Erich Fromm", 266);
   printf("Il testo di %s e' intitolato %-20s\n", testo.autore, testo.titolo);
 }
```

10.7. Files

Apertura di un file

E' preferibile dichiarare una variabile di tipo puntatore a FILE (struttura predefinita) ed effettuare le operazioni su file passando il puntatore in modo da non essere vincolati dal nome del file (che può così essere definito in esecuzione) Esempio: FILE *inputFile, *outputFile; Per l'apertura si utilizza la funzione fopen. Sintassi:

fopen (nomeFile, modalità);

Dove modalità è una stringa che definisce il modo di apertura del file e può essere:

- **r:** sola lettura (il file deve già esistere)
- **r+:** lettura/scrittura (il file deve già esistere)
- **w:** sola scrittura (se il file non esiste, lo crea)
- **w+:** lettura/scrittura (se il file non esiste, viene creato)
- **a:** aggiunge in coda al file, se non esiste lo crea.

Per i file binari si deve posporre "b" alla lettera che indica la modalità; ad esempio la modalità di apertura in lettura/scrittura sarà "rb+".

Operazioni sui file di testo

- fclose(f) chiude il file, DA FARE SEMPRE AL TERMINE DELL'ESECUZIONE
- while (!feof(f)) {...} verifica se il file è terminato (condizione da usare nelle letture)
- ch = fgetc(f) legge un carattere
- fputc(c, f) scrive un carattere
- fgest(f), fputs(s, f) legge o scrive una stringa
- fscanf(f, "%d\n", &n) lettura formattata; NON USARE MAI
- fprintf(f, "%s %d\n", s, 44) scrittura formattata

Operazioni sui file binari

E' possibile accedere in lettura o scrittura ai dati di un file operando su un intero blocco di dati testuali o binari di qualsiasi dimensione. Le funzioni utilizzati sono

- int fread (void *puntBlocco , int dimDato, int numDati, FILE *pf);
- int fwrite (void *puntBlocco, int dimDato, int numDati, FILE *pf);
- int fseek(FILE *fp, long diQuantiByte, int aPartireDa)
- int ftell(FILE *fp)
- int fflush(FILE *fp)
- int remove(char * nomeFIle)