1 Esercizi in pseudocodice

Questa dispensa propone esercizi sulla scrittura di algoritmi in un linguaggio semi–formale, utile all'acquisizione delle abilità essenziali per implementare algoritmi in qualsiasi linguaggio di programmazione.

1.1 Algoritmi ed esecutori

Dato un problema ed un opportuno metodo risolutivo, la *risoluzione* di tale problema è quel processo che trasforma i *dati in ingresso* nei corrispondenti *dati finali*. Affinché la risoluzione di un problema possa essere realizzata attraverso l'uso di un calcolatore, il metodo risolutivo deve poter essere definito come una sequenza di azioni o istruzioni elementari, ovvero occorre codificare la risoluzione del problema in un algoritmo. Si dice *esecutore* quella macchina astratta capace eseguire le istruzioni specificate dall'algoritmo.

Un *algoritmo* è una sequenza finita di istruzioni, definita con precisione, che portano alla risoluzione di un compito. Un algoritmo è tale se possiede le seguenti proprietà:

- **Eseguibilità**: ogni istruzione deve essere eseguibile dal calcolatore in tempo finito;
- **Non–ambiguità**: ogni istruzione deve essere univocamente interpretabile dal calcolatore;
- **Finitezza**: il numero totale di azioni da eseguire, per ogni insieme di dati in ingresso, deve essere finito.

1.2 Il linguaggio "SIMITA"

Introduciamo uno pseudolinguaggio di programmazione, che chiameremo SIMITA[1]. Vogliamo che questo linguaggio sia in grado di gestire:

- Dati singoli (sia quelli in ingresso, sia quelli ottenuti durante l'esecuzione del programma, sia quelli finali);
- Scelte tra alternative;
- Ripetizioni;
- Gruppi di dati.

Immaginiamo di poter gestire i valori dei dati su dei "foglietti". Sui di essi possiamo *scrivere*, *leggere*, *cancellare* e *riscrivere* (come nella memoria dei calcolatori). Per indicare scrittura di un numero N su foglietto f:

$f \leftarrow N$

Allo stesso modo indichiamo la scrittura del contenuto del foglietto g nel foglietto f come:

$f \leftarrow g$

In entrambi i casi l'operazione di scrittura cancellerà tutto ciò che era scritto sul foglietto f. Inoltre, disponiamo di altre espressioni per codificare delle funzionalità di cui dispone l'esecutore (nel nostro caso l'elaboratore):

- $leggi(f)$: indica l'operazione di lettura di un valore introdotto dall'utente (ad esempio, da tastiera) e la scrittura di tale valore nel foglietto f;
- $stampa(f)$: indica l'operazione di lettura del valore contenuto nel foglietto f e la stampa (ad esempio, a video) di tale valore. L'istruzione $stampa$ permette anche di stampare dei caratteri, ad esempio $stampa$("ciao come stai") stampa a video i caratteri "ciao come stai".

Gestiamo nel linguaggio SIMITA il costrutto condizionale, capace di esprimere il concetto di scelta tra alternative, nel seguente modo:

```
Istruzione 0

Se condizione allora
    Istruzione 1
```

[1] Se invece vi volete divertire con un linguaggio differente `https://github.com/esseks/monicelli`

```
    Istruzione 2
Altrimenti
    Istruzione 3
    Istruzione 4
chiudi Se

Istruzione 5
```

Nell'esempio precedente l'algoritmo prescrive al calcolatore di eseguire innanzitutto "Istruzione 0". Successivamente, esso valuta "condizione": se "condizione" risulta vera, allora vengono eseguite "Istruzione 1" ed "Istruzione 2", in caso contrario vengono eseguite "Istruzione 3" ed "Istruzione 4". Al termine viene eseguita "Istruzione 5". Il blocco "**Altrimenti**" è opzionale: se assente, nel caso in cui "condizione" risulti falsa l'esecuzione passa direttamente ad "Istruzione 5". Non vi è limite al numero di istruzioni specificate in ogni blocco del costrutto condizionale.

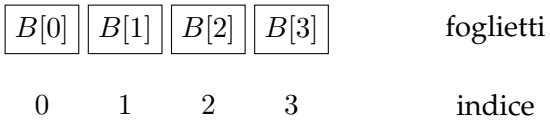
Per poter eseguire più volte una stessa operazione, in SIMITA, abbiamo un costrutto ciclico (o iterativo), capace di esprimere il concetto di ripetitività:

```
Istruzione 0        //questo è un commento

Finché condizione esegui
    Istruzione 1
    Istruzione 2
chiudi Finché

Istruzione 3
```

Nell'esempio precedente la sequenza di istruzioni 1 e 2 viene eseguita un numero *finito* di volte. L' "Istruzione 0" viene eseguita una sola volta. Dopodiché viene valutata "condizione": se risulta vera, allora vengono eseguite "Istruzione 1" ed "Istruzione 2". Viene poi valutata nuovamente "condizione": se vera, si ripete l'esecuzione di "Istruzione 1" ed "Istruzione 2". L'iterazione *termina*, e quindi non vengono più eseguite "Istruzione 1" ed "Istruzione 2", solo quando "condizione" diventa falsa. Quando questo avviene, si esce dal ciclo e si esegue "Istruzione 3". Solitamente, al fine di evitare che il costrutto ciclico non abbia fine, è buona norma che le istruzioni all'interno del ciclo modifichino i foglietti che vengono valutati in "condizione".

Sempre nell'esempio precedente è stata inserita una porzione di istruzione, detta *commento*, che non verrà considerata dall'esecutore, ossia "questo è un commento". Solitamente un commento viene inserito per descrivere in linguaggio naturale ciò che le istruzioni specificano e quindi per chiarificare a chi legge le istruzioni il compito svolto dall'algoritmo.

SIMITA ha anche la nozione di *blocchi di foglietti*:



$B[0]$ indica il primo foglietto del blocco;

$B[1]$ indica il secondo foglietto del blocco;

...

$B[n-1]$ indica l'n-simo foglietto del blocco.

Si noti che ogni foglietto $B[n]$ è, a tutti gli effetti, un normale foglietto. Quindi, ad esempio, le seguenti istruzioni sono valide:

```
i ← 1
f ← B[i]        //scrive il contenuto del secondo foglietto
                //B[1] in f

B[i] ← 3        //scrive 3 nel secondo foglietto B[1]

stampa(B[4])      //stampa il contenuto del quinto foglietto

leggi(B[i])      //legge un dato
                  //e lo scrive nel secondo foglietto
```

1.3 Esercizi

Esercizio 1.1

Scrivere in linguaggio SIMITA l'algoritmo per descrivere il problema di svegliarsi ed uscire di casa, sapendo che:

- Una volta sveglio si deve spegnere la sveglia (`spegniSveglia()`);
- Viviamo in una casa di 3 piani e prima di uscire dobbiamo chiudere tutte le finestre (se sono aperte) ad ogni piano;
- Dormiamo al secondo piano, dove c'è il bagno, in cui ci dobbiamo fare la doccia (`doccia()`);
- Al primo piano c'è la cucina dove dobbiamo fare colazione (`colazione()`), se è domenica facciamo una colazione sostanziosa (`colazioneSostanziosa()`);
- Al piano terreno c'è la porta di uscita che dobbiamo aprire e chiudere.

L'utente ci comunica all'inizio quali sono le finestre aperte e se è domenica. Ogni volta che eseguiamo un'azione dobbiamo comunicarla all'utente stampandola a video.

Esercizio 1.2

Calcolare il prodotto di due numeri interi positivi e stamparlo a video. L'esecutore è in grado di eseguire solamente somme e sottrazioni.

Esercizio 1.3

Calcolare il fattoriale di un numero intero e positivo e stamparlo a video. L'esecutore è in grado di eseguire moltiplicazioni.

Esercizio 1.4

1. Scrivere un algoritmo per valutare e stampare a schermo se un numero intero positivo è pari o dispari utilizzando solo le 4 operazioni fondamentali.
2. Scrivere il medesimo algoritmo supponendo di avere un operatore div che restituisce il risultato intero della divisione (ad esempio, $7\ div\ 3$ ha come risultato 2).

Esercizio 1.5

Determinare se una funzione continua ha uno zero nell'intervallo $[a, b]$ (i cui estremi sono forniti dall'utente). Si faccia ricorso al seguente teorema (teorema di Bolzano):

- Ipotesi: $f : [a, b] \mapsto \mathbb{R}$, continua in $[a, b] \subseteq \mathbb{R}$, tale che $f(a) \cdot f(b) < 0$
- Tesi: $\exists c \in [a, b]$ tale che $f(c) = 0$ (zero di f)

Per valutare la funzione f in un punto generico a potete utilizzare l'operatore

$$g \leftarrow calcola(f, a)$$

che scrive il risultato di $f(a)$ sul foglietto g.

Esercizio 1.6

1. Scrivere un programma che data una sequenza di numeri (positivi) in ingresso restituisce il maggiore e la sua posizione nella sequenza. Supponiamo che la sequenza abbia 10 numeri.

2. Si provi ad implementare un algoritmo analogo a quello dell'esercizio precedente, utilizzando un solo ciclo ed eliminando l'assunzione che i numeri inseriti siano positivi.

3. Si estenda l'algoritmo per calcolare la media m dei valori inseriti.

4. Si estenda l'algoritmo per calcolare e stampare a video la differenza $N[i] - m$ tra ogni elemento della sequenza e la media della sequenza.

5. È possibile implementare questi algoritmi senza l'uso di blocchi di foglietti?

6. Si estenda l'algoritmo per calcolare anche la deviazione standard della sequenza, supponendo di avere a disposizione una funzione che calcola la radice quadrata di un numero (`sqrt()`). È possibile risolvere questo problema senza ricorrere ai blocchi di fogli?

Esercizio 1.7

Determinare almeno uno zero di una funzione continua f nell'intervallo $[a, b]$.

Soluzioni

Soluzione dell'esercizio 1.1

Risoluzione del problema: prima di tutto dobbiamo acquisire e memorizzare i dati relativi alle finestre e al fatto che sia o meno domenica. Dopodichè segnaleremo all'utente che abbiamo iniziato la procedura di uscita e spegneremo la sveglia. Per ogni piano controlleremo se le finestre sono chiuse e, solo se sono aperte, le chiuderemo. Ad ogni piano faremo un'azione differente (doccia al secondo piano, colazione al primo, aprire e chiudere la porta al pian terreno). Per quanto riguarda l'azione della colazione dovremo controllare se è domenica e, solo in quel caso, fare una colazione abbondante. Non appena usciti, comunicheremo all'utente che abbiamo finito la procedura di uscita.

```
leggi(domenica)        //1 se domenica, 0 altrimenti
nPiano ← 2
Finché nPiano > 0 esegui
    leggi(fin[nPiano])        //1 se aperte, 1 se chiuse
    nPiano ← nPiano − 1
chiudi Finché
porta = 0        //0 se chiusa, 1 se aperta
stampa("Inizio procedura uscita")
spegniSveglia()
nPiano ← 2
Finché nPiano > 0 esegui
    Se fin[nPiano] = 1 allora
        fin[nPiano] = 0
        stampa("Chiuse le finestre al piano nPiano")
    chiudi Se
    Se nPiano = 2 allora
        doccia()
        stampa("Doccia fatta")
    chiudi Se
    Se nPiano = 1 allora
        Se domenica = 1 allora
            colazioneSostanziosa()
            stampa("Colazione sostanziosa fatta")
        Altrimenti
            colazione()
            stampa('Colazione fatta')
        chiudi Se
    chiudi Se
    Se nPiano = 0 allora
        porta = 1
        stampa("Porta aperta")
        porta = 0
        stampa("Porta chiusa")
    chiudi Se
    nPiano ← nPiano − 1
chiudi Finché
stampa("Fine procedura uscita")
```

Soluzione dell'esercizio 1.2

Risoluzione del problema: dopo aver letto e memorizzato i due numeri, calcoliamo il prodotto in modo cumulativo utilizzando un ciclo. Infine stampiamo il risultato a video.

Soluzione 1

```
leggi(f)        //leggi il primo numero
leggi(g)        //leggi il secondo numero

risultato ← 0       //inizializzo il risultato

Finché f > 0 esegui
    risultato ← risultato + g       //aggiungi g per f volte
    f ← f − 1
chiudi Finché

stampa(risultato)
```

Soluzione 2

```
leggi(f)        //leggi il primo numero
leggi(g)        //leggi il secondo numero

Se g < f allora       //assicuriamoci che f < g
    tmp ← f       //metto f su un foglietto temporaneo
    f ← g       //g contiene il numero minore
    g ← tmp
Altrimenti
    tmp ← g       //l'altro lo scrivo in g
chiudi Se

Finché f > 1 esegui       //ripeti la somma f − 1 volte
    tmp ← tmp + g       //aggiungi g per f volte
    f ← f − 1
chiudi Finché

stampa(tmp)
```

In alcuni casi (ad esempio, $f = 10000$ e $g = 2$). questo algoritmo è più efficiente del primo in quanto, se eseguito, userebbe meno istruzioni in totale. L'efficienza di un algoritmo rispetto ad un altro avviene rispetto a due caratteristiche:

- **Complessità temporale**: uso meno istruzioni in totale;
- **Complessità spaziale**: uso meno foglietti in totale;

In questo caso entrambi gli algoritmi hanno la stessa complessità spaziale, in quanto usano entrambi 3 foglietti in tutto. Il secondo algoritmo "spreca" 4 istruzioni all'inizio per poter poi risparmiarne nel caso in cui $f \gg g$.

Soluzione dell'esercizio 1.3

Risoluzione del problema: si legge e memorizza il numero dato. Si inizializza il risultato al numero dato. Si esegue un ciclo moltiplicando il risultato per numeri descrescenti partendo dal dato meno 1 fino ad 1. Si stampa il risultato a video.

```
leggi(f)
fattoriale ← f

Finché f > 2 esegui
    f ← f − 1
    fattoriale ← fattoriale ∗ f
chiudi Finché

stampa(fattoriale)
```

Soluzione dell'esercizio 1.4

1. Risoluzione del problema: dopo aver letto e memorizzato il numero, sottraiamo 2 finché il risultato non è 1 o 0. Se il risultato è 1, il numero di partenza è dispari, altrimenti è pari.

```
leggi(f)        //leggi il numero

Finché f > 2 esegui
    f ← f − 2
chiudi Finché

Se f = 1 allora
    stampa(dispari)
Altrimenti
    stampa(pari)
chiudi Se
```

2. Risoluzione del problema: dopo aver letto e memorizzato il numero, dividiamo per 2 usando l'operatore div e successivamente moltiplichiamo per 2. Se il risultato è uguale al numero di partenza, il numero è pari, altrimenti è dispari.

```
leggi(f)        //leggi il numero

g ← 2 * (f div 2)

Se f = g allora
    stampa(pari)
Altrimenti
    stampa(dispari)
chiudi Se
```

Soluzione dell'esercizio 1.5

Risoluzione del problema: leggiamo e memorizziamo gli estremi dell'intervallo forniti dall'utente.

```
leggi(a)        //lettura estremo inferiore
leggi(b)        //lettura estremo superiore

A ← calcola(f, a)
B ← calcola(f, b)

Se A * B < 0 allora
    stampa("esiste almeno uno zero")
Altrimenti
    stampa("potrebbe non esistere alcuno zero")
chiudi Se
```

Soluzione dell'esercizio 1.6

1. Risoluzione del problema: innanzitutto procediamo all'acquisizione dei 10 numeri. All'inizio il massimo è zero e la sua posizione è meno 1. Confrontiamo questo massimo con il primo numero della sequenza dei numeri acquisiti. Se il numero della sequenza è maggiore lo sostituiamo al massimo e registriamo la sua posizione. Ripetiamo l'operazione con tutti i numeri della sequenza e infine stampiamo a video valore e posizione del massimo.

```
max ← 0
i ← 0
pos ← −1

Finché i < 10 esegui
    leggi(N[i])
    i ← i + 1
chiudi Finché

i ← 0
Finché i < 10 esegui
    Se N[i] > max allora
        max ← N[i]
        p ← i
    chiudi Se
    i ← i + 1
chiudi Finché

stampa("il valore massimo è:")
stampa(max)

stampa("ed è in posizione:")
stampa(p)
```

2. Risoluzione del problema: prima di tutto notiamo che il primo ciclo, quello di acquisizione, memorizza i numeri nel blocco di foglietti. Tuttavia, il secondo ciclo, non fa altro che esaminarli (nello stesso ordine). Quindi, è possibile eliminare del tutto la necessità di memorizzare i numeri, e svolgere le istruzioni per il controllo "maggiore di" subito dopo l'acquisizione.

   Inoltre, eliminando l'assunzione di numeri positivi, non abbiamo un estremo inferiore (zero). Perciò è necessario inizializzare il massimo al primo valore incontrato (che potrebbe essere inferiore a zero).

```
i ← 0
p ← i
n ← 0      //foglietto per il numero letto

Finché i < 10 esegui
    leggi(n)

    Se i = 0 ∨ n > max allora      //primo numero o maggiore
        max ← n      //salvo il valore massimo
        p ← i      //salvo la posizione
    chiudi Se

    i ← i + 1      //incremento il contatore
chiudi Finché

stampa("il valore massimo è:")
stampa(max)

stampa("ed è in posizione:")
stampa(p)
```

3. Risoluzione del problema: il calcolo della media richiede che, durante l'acquisizione, si calcoli anche la somma incrementale di tutti i numeri letti. Al termine del ciclo, tale somma verrà divisa per il numero di input immessi.

```
i ← 0
p ← i
media ← 0
n ← 0      //foglietto per il numero letto

Finché i < 10 esegui
    leggi(n)

    Se i = 0 ∨ n > max allora      //primo numero o maggiore
        max ← n      //salvo il valore massimo
        p ← i      //salvo la posizione
    chiudi Se

    media ← media + n      //somma incrementale
    i ← i + 1      //incremento il contatore
chiudi Finché

media ← media/(i + 1)

stampa("il valore massimo è:")
stampa(max)

stampa("ed è in posizione:")
stampa(p)

stampa("il valore medio è:")
stampa(media)
```

4. Risoluzione del problema: per calcolare lo scarto dalla media di ogni elemento è necessario memorizzare tutti gli elementi, perché la media sarà calcolabile solo dopo aver letto tutti i numeri. Quindi:

```
i ← 0
p ← i
media ← 0
n ← 0      //foglietto per il numero letto

Finché i < 10 esegui
    leggi(n)

    Se i = 0 ∨ n > max allora      //primo numero o maggiore
        max ← n      //salvo il valore massimo
        p ← i      //salvo la posizione
    chiudi Se

    N[i] ← n      //memorizzazione elemento i-esimo
    media ← media + n      //somma incrementale
    i ← i + 1      //incremento il contatore
chiudi Finché

media ← media/(i + 1)
i ← 0

Finché i < 10 esegui      //ciclo su tutti i foglietti
    stampa("lo scarto dalla media di ")
    stampa(N[i])
    stampa("è: ")
    stampa(N[i] − media)      //scarto dalla media
    i ← i + 1
chiudi Finché

stampa("il valore massimo è:")
stampa(max)

stampa("ed è in posizione:")
stampa(p)

stampa("il valore medio è:")
stampa(media)
```

5. Risoluzione del problema: per il calcolo del massimo e della media non è necessario memorizzare i valori inseriti, quindi non sono necessario blocchi di foglietti. Per quanto riguarda lo scarto della media, non è possibile implementare un algoritmo incrementale, poiché il valore della media è noto solo alla fine dell'acquisizione dei valori.

6. Risoluzione del problema: la deviazione standard è definita come la media degli scarti quadratici dalla media, ovvero:

$$\mathrm{devstd}(\{x_1, \ldots, x_M\}) = \sqrt{\frac{\sum_{i=1}^{M}(x_i - m)^2}{M}}$$

si deve quindi calcolare la somma degli scarti quadratici $N[i] - m$, dividere tale numero per il contatore ed estrarne la radice quadrata.

```
i ← 0
p ← i
media ← 0
devstd ← 0
n ← 0      //foglietto per il numero letto

Finché i < 10 esegui
    leggi(n)

    Se i = 0 ∨ n > max allora      //primo numero o maggiore
        max ← n      //salvo il valore massimo
        p ← i      //salvo la posizione
    chiudi Se

    N[i] ← n      //memorizzazione elemento i-esimo
    media ← media + n      //somma incrementale
    i ← i + 1      //incremento il contatore
chiudi Finché

media ← media/(i + 1)

i ← 0
Finché i < 10 esegui      //ciclo su tutti i foglietti
    scarto ← (N[i] − media)
    scarto ← scarto ∗ scarto      //scarto quadratico
    devstd ← devstd + scarto      //somma scarti
    i ← i + 1
chiudi Finché

devstd ← devstd/(i + 1)      //media somma scarti
devstd ← sqrt(devstd)      //deviazione standard

stampa("il valore massimo è:")
stampa(max)

stampa("ed è in posizione:")
stampa(p)

stampa("il valore medio è:")
stampa(media)

stampa("la deviazione standard è:")
stampa(devstd)
```

Diversamente dal caso degli scarti esiste una formula per il calcolo incrementale della deviazione standard, che può essere derivato analiticamente dalla sua definizione. Nell'algoritmo precedentemente descritto non ha senso utilizzare tale formula in quanto vogliamo poi stampare gli scarti.

Soluzione dell'esercizio 1.7

```
leggi(a)        //lettura estremo inferiore
leggi(b)        //lettura estremo superiore

A ← calcola(f, a)
B ← calcola(f, b)

Se A * B > 0 allora
    stampa("potrebbe non esistere alcuno zero")
Altrimenti
    K ← 1

    Finché K ≠ 0 esegui
        c ← (b + a)/2
        K ← calcola(f, c)

        Se A * K < 0 allora
            b ← c
            B ← K
        chiudi Se
        Se B * K < 0 allora
            a ← c
            A ← K
        chiudi Se

    chiudi Finché
chiudi Se

stampa(c)
```

Attenzione: Il programma potrebbe non terminare nel caso in cui lo zero non sia in un valore razionale ($c \in \mathbb{R} \setminus \mathbb{Q}$), poiché K è sempre un razionale ($K \in \mathbb{Q}$).

Inoltre, si osservi che il programma non gestisce il caso in cui lo zero sia $c = a$ o $c = b$.

2 Operatori matematici e costrutto `if`

Questa dispensa propone esercizi sulla scrittura di algoritmi, in linguaggio C, utili alla comprensione delle operazioni tra numeri e del costrutto condizionale `if`. Si introducono anche le due funzioni principali della libreria `stdio.h`: `scanf` e `printf`.

Figura 2.1: Un consiglio da seguire.

Nel caso in cui ci fossero delle domande riguardanti il funzionamento dei costrutti di base si fa riferimento alla Figura 2.1. Nello specifico, si può consultare come riferimento:

- Il manuale del corso: "Informatica: arte e mestiere", D. Mandrioli, S. Ceri, L. Sbattella, P. Cremonesi, G. Cugola, McGraw-Hill Education;
- Un manuale di C on–line: `http://www.cplusplus.com/`.

Si assume una conoscenza di base sul linguaggio C, tale da permettere al lettore di comprendere il significato del seguente frammento di codice.

```
#include <stdio.h>

void main() {

    /* Corpo del programma */

    getchar();
```

```
}
```

Ai fini del corso, è ininfluente la scelta di dichiarare il `main` come

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

  /* Corpo del programma */

  getchar();
  return 0;
}
```

oppure come

```
void main() {

  /* Corpo del programma */

  getchar();
}
```

Tuttavia, si richiede che il lettore comprenda la differenza tra le due alternative. La riga di codice `Corpo del programma` non viene considerata dal compilatore, in quanto circondata dai caratteri `/*` e `*/`, che indicano l'apertura e la chiusura di un commento, rispettivamente. Nel caso in cui si voglia commentare da un punto fino alla fine della riga, può essere anche usata l'espressione `//`. L'istruzione `getchar()` non fa parte della soluzione. Si tratta di un'istruzione bloccante per mettere l'elaboratore in attesa di un carattere da tastiera. Senza questa istruzione, o istruzioni equivalenti (e.g., la `system("PAUSE")`), l'esecuzione del programma termina immediatamente senza permettere all'utente di visualizzare l'output a video.

2.1 Operazioni matematiche

Si assume che il lettore sia familiare con i tipi di dato numerici previsti dal C (e.g., `int`, `float`) e con i rispettivi di caratteri che di specifica del formato (e.g., `"%d"`, `"%f"`). Inoltre, le operazioni matematiche essenziali necessarie alla comprensione degli esercizi proposti in questa sezione sono:

```
#include <stdio.h>

void main() {
    printf("Addizione: 1+2 = %d\n", 1+2);

    printf("Moltiplicazione: 1*2 = %d\n", 1*2);

    printf("Sottrazione: 1-2 = %d\n", 1-2);

    printf("Divisione: 8/3 = %d (%f)\n", 8/3, 8.0/3.0);

    printf("Resto della divisione intera: 8 mod 3 = %d", 8 % 3);

    getchar();
}
```

Prima di procedere oltre, il lettore deve aver compreso il significato di questo frammento di codice (e.g., provando a compilarlo e ad eseguirlo).

2.2 Costrutto `if` e condizioni

Il costrutto `if` codifica un ramo condizionale. Il linguaggio C segue la seguente sintassi:

```
if (condizione)
  statement;
[else statement;]
```

dove le parentesi quadre indicano che la parte **`else`** `statement;` è opzionale. Come per tutti gli altri costrutti in C, se uno `statement` è una sola istruzione terminata da punto e virgola, non serve altro. Se invece uno `statement` è composto da più istruzioni terminate da punto e virgola, sarà necessario racchiuderlo tra parentesi graffe, ossia:

```
if (condizione) {
  istruzione1;
  istruzione2;
  ...
}
```

La `condizione` è un'espressione booleana, ovvero un'istruzione che, quando valutata, risulta sempre in un valore pari a zero (0, falso) o uno (1, vero). Per comporre espressioni booleane complesse si utilizzano i seguenti operatori:

Operatori relazionali valutano relazioni binarie tra i due operandi:

- `<` minore di
- `<=` minore di o uguale uguale a
- `>` maggiore di
- `>=` maggiore di o uguale a
- `==` uguale a
- `!=` non uguale a (diverso)

Operatori booleani valutano condizioni di verità tra i due operandi

- `&&` AND (congiunzione logica)
- `||` OR (disgiunzione logica)

Attenzione: si osservi che in C, l'operazione di assegnamento `a = 3` è diversa dall'operazione di confronto `a == 3`. La prima è sempre valutata come vera (1, uno), mentre

la seconda, ovviamente, dipende dal valore memorizzato in `a`. Perciò:

```
#include <stdio.h>

void main() {
    int a;

    scanf("%d", &a); //leggi(a)

    if (a == 4) //confronto
        printf("La variabile 'a' contiene il valore 4\n");
    else
        printf("La variabile 'a' NON contiene il valore 4\n");

    if (a = 4) //assegnamento
        printf("Questo ramo viene sempre eseguito.\n");
    else
        printf("Questo ramo NON viene mai eseguito.\n");

    getchar();
}
```

2.2.1 Esercizi

Esercizio 2.1

Scrivere un programma che esegua la differenza di due numeri interi inseriti da tastiera.

Esercizio 2.2

Scrivere un programma che riceve in ingresso un prezzo (numero razionale) ed uno sconto (intero tra 0 e 100) da applicare, e restituisce il prezzo scontato e il risparmio ottenuto.

Esercizio 2.3

Scrivere un programma che prende in ingresso un tempo espresso in ore, minuti e secondi e ne restituisce l'equivalente in secondi.

Esercizio 2.4

Scrivere un programma che prende in ingresso un tempo espresso in secondi e ne restituisce l'equivalente nel formato ore, minuti, secondi.

Esercizio 2.5

Scrivere un programma che calcoli la distanza tra due punti, a e b, interi sulla retta $y = 0$.

Esercizio 2.6

Scrivere un programma che calcoli la distanza tra due punti, a e b, interi su un retta. Potete utilizzare la funzione `abs()` della libreria `math.h`, che calcola il valore assoluto di un numero intero.

```
printf("abs(1-2) = %d", abs(1-2));
//output: abs(1-2) = 1
```

Esercizio 2.7

Scrivere un programma che legga da input un numero intero e stampi su output:

- la stringa `basso` se il numero è compreso tra 0 e 3;
- la stringa `MediO` se il numero è compreso tra 4 e 8;
- la stringa `ALTO!` se il numero è compreso tra 9 e 10;
- la stringa `Numero non valido` altrimenti.

Esercizio 2.8

1. Scrivere un programma che dati tre interi positivi valuti se essi possono essere i lati di un triangolo
2. Nel caso di risposta positiva al punto precedente si comunichi anche il tipo di triangolo (scaleno, isoscele, equilatero, rettangolo)

Esercizio 2.9

Scrivere un programma che legga da tastiera un numero intero che rappresenta un anno (e.g., 2012) e che determini poi se tale anno è bisestile o meno. Si può assumere che il numero intero letto da tastiera sia sempre valido (e.g., di 4 cifre, positivo).

Un anno è bisestile se è multiplo di 4 ma non di 100, oppure se è multiplo di 400.

Soluzioni

Soluzione dell'esercizio 2.1

Risoluzione del problema: dovremo innanzitutto leggere e memorizzare i due numeri. Una volta eseguita l'operazione di differenza dovremo stampare a video il risultato dell'operazione.

```
#include <stdio.h>

// inizio della procedura principale, detta "main"
void main() {
    /*  Dichiarazione delle variabili, equivalenti ai "foglietti". */
    int a;
    int b;
    int differenza;

    /* Stampa a video della stringa formattata. Il carattere speciale
         "\n"
     * manda a capo. */

    printf("Inserisci il primo numero \n"); // stampa("...")

    /* Legge un valore di tipo intero (i.e., "%d") e lo scrive nella
       cella di memoria della variabile "a". L'indirizzo di memoria
       di tale cella  specifocato con l'operatore "&", ovvero "
       indirizzo di" */

    scanf("%d", &a); //leggi(a)

    // Come sopra
    printf("Inserisci il secondo numero \n"); //stampa("...")
    scanf("%d", &b); //leggi(b)

    /* Calcola la differenza tra il valore memorizzato nella
       variabile "a" ed
     * il valore memorizzato nella variabile "b". Il risultato dell'
       operazione
     *  scritto nella variabile "differenza". */
    differenza = a - b;

    /* Stampa a video la stringa formattata. Al posto dello
       specificatore di
     * formato "%d" verr stampato il contenuto della variabile "
       differenza",
     * formattato come intero. */
    printf("La differenza  %d\n", differenza);
}
```

Soluzione dell'esercizio 2.2

Risoluzione del problema: leggiamo e memorizziamo la cifra e lo sconto. Dopodichè calcoliamo il prezzo scontato e, facendone la differenza con il prezzo iniziale, calcoliamo l'ammontare dello sconto. Infine stampiamo a schermo i due valori ottenuti.

```
#include <stdio.h>

void main()
{
    float prezzo;
    float sconto;
    float finale;
    float risparmio;

    printf("Inserisci il prezzo originale:\n");
    scanf("%f", &prezzo);

    printf("Inserisci lo sconto da applicare:\n");
    scanf("%f", &sconto);

    risparmio = prezzo * (sconto / 100);
    finale = prezzo - risparmio;

    printf("Il prezzo scontato : %f\n", finale);
    printf("Il risparmio : %f\n", risparmio);
}
```

Soluzione dell'esercizio 2.3

Risoluzione del problema: leggiamo i tre dati in ingresso e li memorizziamo. Per calcolare i secondi innanzitutto trasformiamo le ore in minuti (moltiplicando per 60) e successivamente i minuti in ore.

```
#import <stdio.h>

void main() {
    int secondi;
    int sec;
    int min;
    int ore;

    printf("Inserisci il numero di ore:\n");
    scanf("%d", &ore);
    printf("Inserisci il numero di minuti:\n");
```

```
    scanf("%d", &min);
    printf("Inserisci il numero di secondi:\n");
    scanf("%d", &sec);

    secondi = (ore * 60 + min) * 60 + sec;

    printf("%d ore, %d minuti e %d secondi equivalgono a %d secondi\n
        ",
            ore, min, sec, secondi);
}
```

Soluzione dell'esercizio 2.4

Risoluzione del problema: leggiamo e memorizziamo i secondi da convertire. Dividendo (per intero) per 60 calcoliamo il numero di minuti e, calcolando il resto, otteniamo i secondi nel secondo formato. Ripetendo l'operazione con i minuti riusciamo a trovare il numero di ore e minuti che ci occorrono nel secondo formato. Stampiamo a schermo ore, minuti e secondi.

```
#import <stdio.h>

void main() {
    int secondi;
    int sec;
    int min;
    int ore;

    printf("Inserisci il numero di secondi:\n");
    scanf("%d", &secondi);

    min = secondi / 60;
    sec = secondi - min * 60;

    ore = min / 60;
    min = min - ore * 60;

    printf("%d secondi equivalgono a (h:m:s) %d:%d:%d",
            secondi, ore, min, sec);
}
```

Oppure in maniera più compatta:

```
#import <stdio.h>

void main() {
    int secondi;
    int sec;
    int min;
```

```
    int ore;

    printf("Inserisci il numero di secondi:\n");
    scanf("%d", &secondi);

    ore = secondi / 3600;
    min = (secondi - ore * 3600) / 60;
    sec = secondi - ore * 3600 - min * 60;

    printf("%d secondi equivalgono a (h:m:s) %d:%d:%d",
            secondi, ore, min, sec);
}
```

Soluzione dell'esercizio 2.5

```
#include <stdio.h>

void main() {
    int a, b, lunghezza;

    printf("Inserisci il primo punto: ");
    scanf("%d", &a);

    printf("Inserisci il secondo punto: ");
    scanf("%d", &b);

    lunghezza = a - b;

    if (lunghezza == 0) {
        printf("I due punti coincidono.\n");
    } else {
        if (lunghezza < 0) {
            lunghezza = -lunghezza;
        }

        printf("La lunghezza del segmento : %d\n", lunghezza);
    }

    getchar();
}
```

Soluzione dell'esercizio 2.6

```
#include <stdio.h>
#include <math.h>

void main() {
```

```
    int a;
    int b;
    int lunghezza;

    printf("Inserisci il primo punto: ");
    scanf("%d", &a);

    printf("Inserisci il secondo punto: ");
    scanf("%d", &b);

    lunghezza = abs(a - b);

    printf("La lunghezza del segmento : %d\n", lunghezza);

    getchar();
}
```

Soluzione dell'esercizio 2.7

```
#include <stdio.h>

void main(){

int numero;

printf("Inserire un numero da 0 a 10: ");
scanf("%d",&numero);

if (numero >= 0 && numero <= 3)
    printf("basso\n");
if (numero >= 4 && numero <= 8)
    printf("MediO\n");
if (numero >= 9 && numero <= 10)
    printf("Alto!\n");
if  (numero < 0 || numero > 10)
    printf("Numero non valido\n");

}
```

Oppure nella versione più ottimizzata:

```
#include <stdio.h>

void main(){

int numero;

printf("Inserire un numero da 0 a 10: ");
```

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

if (numero >= 0 && numero <= 3)
    printf("basso\n");
else {
    if (numero >= 4 && numero <= 8)
       printf("MediO\n");
    else {
        if (numero >= 9 && numero <= 10)
            printf("Alto!\n");
        else
            printf("Numero non valido\n");
    }
}

}
```

Soluzione dell'esercizio 2.8

```
#include <stdio.h>

void main(){

int a,b,c;
int somma_a, somma_b, somma_c;
int diff_a, diff_b, diff_c;

scanf("%d",&a);
scanf("%d",&b);
scanf("%d",&c);

/* somma minore del terzo*/
somma_a = b+c > a;
somma_b = a+c > b;
somma_c = a+b > c;

/* differenza maggiore del terzo*/
diff_a = b+c > a;
diff_b = a+c > b;
diff_c = a+b > c;

printf("I tre numeri");
if (!(somma_a && somma_b && somma_c && diff_a && diff_b && diff_c))
     printf(" non");
printf(" sono i lati di un triangolo\n");

}
```

```
#include <stdio.h>

void main(){

int a,b,c;
int somma_a, somma_b, somma_c;
int diff_a, diff_b, diff_c;
int is_triangolo;
int ipotenusa, somma;

scanf("%d",&a);
scanf("%d",&b);
scanf("%d",&c);

/* somma minore del terzo*/
somma_a = b+c > a;
somma_b = a+c > b;
somma_c = a+b > c;

/* differenza maggiore del terzo*/
diff_a = b+c > a;
diff_b = a+c > b;
diff_c = a+b > c;

is_triangolo = somma_a && somma_b && somma_c && diff_a && diff_b &&
   diff_c;
printf("I tre numeri");
if (!is_triangolo)
     printf(" non");
printf(" sono i lati di un triangolo");

if (is_triangolo) {
    if (a == b && b == c)
        printf(" equilatero");
    else {
        if (a != b && a!= c && c != b)
            printf(" scaleno");
        else
            printf(" isoscele");
        // controllo se  rettangolo
        if (a >= b && a >= c) {
            ipotenusa = a * a;
            somma = b*b + c*c;
        }
        if (b > a && b >= c) {
            ipotenusa = b * b;
            somma = a*a + c*c;
        }
        if (c >= a && c >= b) {
```

```
            ipotenusa = c * c;
            somma = a*a + b*b;
        }
        if (ipotenusa == somma)
            printf(" (rettangolo)");
    }

}
printf("\n");
}
```

Soluzione dell'esercizio 2.9

```
#include <stdio.h>

void main() {
    int anno;
    int bisestile;

    printf("\nInserisci il numero dell'anno: ");
    scanf("%d", &anno);

                if (anno % 4 == 0) {
                                bisestile = 1;
                                if (anno % 100 == 0 && anno % 400 !=
                                    0)
                                                bisestile = 0;
                else
                                bisestile = 0;
                }

    printf("L'anno %d ", anno);
    if (!bisestile)
        printf("non ");
    printf("e' bisestile\n");

}
```

oppure

```
#include <stdio.h>

void main() {
int anno;
int d4,d100,d400;

scanf("%d",&anno);
```

```
d4 = anno % 4 == 0;
d100 = anno % 100 == 0;
d400 = anno % 400 == 0;

if ((d4 && !d100) || d400)
    printf("E' bisestile\n");
else
    printf("Non e' bisestile\n");

}
```

3 Costrutti `while`, `for` e `switch`

Questa dispensa propone esercizi sulla scrittura di algoritmi, in linguaggio C, utili alla comprensione dei costrutti `while` e `for` e `switch`.

I costrutti per costruire cicli in C sono il `while` (e la variante `do...while` e il `for`.

```
inizializzazione; //opzionale

while (condizione) {
  corpo;

  incremento; //opzionale
}
```

La `condizione` è valutata prima di ogni iterazione, inclusa la prima; quindi il corpo del ciclo (e quindi anche l'eventuale istruzione di `incremento`) potrebbero non eseguire mai, nel caso in cui la condizione sia falsa dall'inizio.

```
inizializzazione; //opzionale

do {
  corpo;

  incremento; //opzionale
} while (condizione);
```

Invece, nella variante `do...while`, il corpo e l'eventuale istruzione di incremento sono eseguiti almeno una volta prima di valutare la condizione.

Il `for` è equivalente, ma ha una sintassi più compatta:

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

Si noti che le parentesi sono necessarie solo nel caso di corpo con più di un'istruzione. Inoltre, l'espressione di inizializzazione e di incremento sono opzionali. Di fatto, un ciclo `while` può essere scritto in modo equivalente con un costrutto `for` nel seguente modo:

```
inizializzazione;

for (; condizione; ) {
  corpo;

  incremento;
}
```

Il costrutto del C che permette di scegliere tra alternative multiple (più di 2) è lo `switch`.

```
inizializzazione;

switch (variabile) {
  case valore1:
      istruzione1;
      break;
  case valore2:
      istruzione2;
      break;
  default:
      istruzione3;
      break;
}
```

Il `break` tra un case e l'altro permette che venga eseguito solo il codice relativo al case in cui si è capitati. Il `break` finale non è necessario ma consigliato (nel caso in cui si dovesse immettere nuovi case).

3.0.1 Esercizi

Esercizio 3.1

Scrivere un programma che dato un numero positivo ne restituisca la radice intera

Esercizio 3.2

1. Scrivere un programma che dato un numero reale positivo l e un intero positivo n restituisca l'area del poligono regolare con n lati di lunghezza l. Si implementi una soluzione per che gestisca i casi $n = 3, 4, 5, 6$ e che sia espandibile agevolmente. Suggerimento: l'area del pentagono regolare é:

$$A = l^2 \frac{5}{2} \sqrt{\frac{\sqrt{5}}{10} + \frac{1}{4}}$$

2. Supporre ora che l'utente possa inserire valori negativi per l e n. Si ripeta l'acquisizione dei dati finché l'utente non inserisce dei valori accettabili per i due parametri.

Esercizio 3.3

Scrivere un programma che richiede all'utente un intero positivo e ne stampa a schermo tutti i divisori.

Esercizio 3.4

Scrivere un programma che richiede all'utente un intero positivo e determina se è primo o meno. Il programma deve continuare a chiedere il numero fino a che l'utente non ne inserisce uno positivo.

Esercizio 3.5

Scrivere un programma che richiede all'utente un intero positivo N e stampa a schermo i primi N numeri primi. Ad esempio: con $N = 7$ a schermo avremo 2 3 5 7 11 13 17.

Esercizio 3.6

Scrivere un programma che richiede all'utente un indovinare un numero casuale (generato con la funzione `rand`) tra 1 e 10.

Esercizio 3.7

Dati due numeri m e n questi sono numeri amicali se la somma dei divisori di m è uguale a n, e viceversa (per esempio 220 e 284, 1184 e 1210, 2620 e 2924, 5020 e 5564, 6232 e 6368, 17296 e 18416). Scrivere un programma che ricevuto in ingresso due numeri interi restituisce 1 se i numeri sono amicali, 0 altrimenti.

Soluzioni

Soluzione dell'esercizio 3.1

Abbiamo già visto in un'esercitazione precedente lo pseudocodice simita per questo algoritmo.

Iniziamo con la soluzione utilizzando il costrutto `while`

```
#include <stdio.h>

void main() {

int n; //numero inserito dall'utente
int radice; //radice intera di n

printf("Inserire un numero intero positivo: ");
scanf("%d",&n);

radice = n / 2;

while (radice * radice > n)
    radice--;

printf("La radice intera di %d e': %d",n,radice);

}
```

Alternativamente si può utilizzare anche il ciclo `for`

```
#include <stdio.h>

void main() {

int n; //numero inserito dall'utente
int radice; //radice intera di n
printf("Inserire un numero intero positivo: ");
scanf("%d",&n);

for(radice = n/2; radice*radice > n; radice--) {

}

printf("La radice intera di %d e': %d",n,radice);

}
```

Attenzione In questa soluzione il ciclo non è vuoto, ma semplicemente l'unica operazione è il decremento della variabile che viene eseguito nella terza espressione del ciclo `for`

Soluzione dell'esercizio 3.2

1.
```c
#include <stdio.h>
#include <math.h>

void main(){

float l; //lato del poligono
int n; //numero di lati
float area; //area del poligono regolare con n lati lunghi l

printf("Inserire la lunghezza del lato ");
scanf("%f",&l);
printf("Inserire il numero di lati ");
scanf("%d",&n);

switch (n) {
case 1:
case 2:
    printf("Il poligono e' degenere");
    break;
case 3:
    area = l * l * sqrt(3) / 2;
    break;
case 4:
    area = l * l;
    break;
case 5:
    area = l * l * 5 / 2 * sqrt(sqrt(5)/10 + 0.25);
    break;
case 6:
    area = l * l * sqrt(3) * 3;
    break;
default:
    printf("Poligono non supportato");
}

if (n > 2 && n < 7)
printf("L'area del poligono regolare con %d lati e': %f",n,area)
   ;

}
```

2.
```c
#include <stdio.h>
#include <math.h>
```

```
void main(){

float l; //lato del poligono
int n; //numero di lati
float area; //area del poligono regolare con n lati lunghi l

do {
    printf("Inserire la lunghezza del lato ");
    scanf("%f",&l);
    printf("Inserire il numero di lati ");
    scanf("%d",&n);
    if (l <= 0 || n < 1)
        printf("Immissione scorretta. Inserire due numeri
           positivi\n");
} while (l <= 0 || n < 1);

switch (n) {
case 1:
case 2:
    printf("Il poligono e' degenere");
    break;
case 3:
    area = l * l * sqrt(3) / 2;
    break;
case 4:
    area = l * l;
    break;
case 5:
    area = l * l * 5 / 2 * sqrt(sqrt(5)/10 + 0.25);
    break;
case 6:
    area = l * l * sqrt(3) * 3;
    break;
default:
    printf("Poligono non supportato");
}

if (n > 2 && n < 7)
printf("L'area del poligono regolare con %d lati e': %f",n,area)
   ;

}
```

Soluzione dell'esercizio 3.3

```
#include <stdio.h>
```

```c
void main() {
    int n, i;

    do {
        printf("Inserire un intero ");
        scanf("%d" , &n);
    } while(n < 1);

    // soluzione con il ciclo while

    i = 1;

    while(i <= n / 2) {
        if(n % i == 0)
            printf(" %d " , i);

        i++;
    }
}
```

```c
#include <stdio.h>

void main() {
    int n, i;

    do {
        printf("Inserire un intero ");
        scanf("%d" , &n);
    } while(n < 1);

    for(i = 1; i <= n / 2; i++)
        if(n % i == 0)
            printf(" %d ", i);

}
```

Attenzione L'`init_expr i = 1;` e la `loop_expr i++;` fanno parte della prima riga nel ciclo. Nel ciclo `while` erano prima del ciclo e all'interno rispettivamente.

Soluzione dell'esercizio 3.4

```c
#include <stdio.h>

void main()
{
    int n, i, primo;
```

```
    do {
        printf("Inserire un intero: ");
        scanf("%d" , &n);
    } while(n < 1);

    /*
      Variabile di flag: il suo valore cambia da 1 a 0 quando trovo
          un
      divisore. Nell'inizializzazione della variabile, assumiamo che
      il numero inserito sia primo.
    */
    primo = 1;

    /*
      Occorre escludere 1 e n perche' dobbiamo considerare solo i
      divisori propri. Quindi iniziamo da 2.
     */
    i = 2;

    /*
      Inutile controllare i numeri da n/2 + 1 a n: non contengono
         divisori.

      Arresto il ciclo quando la variabile di flag primo diventa 0: è
      inutile cercare altri divisori
    */
    while (i <= n/2 && primo == 1) {
        if (n % i == 0) //se divisore trovato -> numero non primo!
            primo = 0;

        i++;
    }

    /*
      Qui, al termine del ciclo, se primo == 1 vuol dire che non si è
      mai verificato n % i == 0, quindi non esistono divisori propri
      ed n è primo.
    */

    printf("\n%d", n);

    if (primo == 0)
        printf(" non "); // il corpo dell'if è solo quest'istruzione
    printf(" è primo\n");
}
```

```
#include <stdio.h>

void main() {
```

```
    int n, i, trovato;

    do {
        printf("Inserire un intero: ");
        scanf("%d" , &n);
    } while(n < 1);

    for (i = n/2; i > 1 && n % i != 0; i--);

    /* //In alternativa

       i = n/2;

       while (i > 1 && n % i != 0)
           i--;

     */

    if (i > 1)
                printf("Divisore %d trovato: numero non primo.", i);
    else //i == 1
                                printf("Divisore non trovato: numero
                                   primo.");

    getchar();
}
```

Soluzione dell'esercizio 3.5

```
#include <stdio.h>

void main(){

int n_primi = -2;
int n = 2;
int i;
int divisore;
int flag = 1;

while (n_primi < 1) {
    printf("Numero positivo: ");
    scanf("%d",&n_primi);
}

i = 1;
while (i <= n_primi) {
    flag = 1;
    for(divisore = 2; divisore <= n / 2 && flag == 1; divisore++) {
        if (n % divisore == 0) {
```

```
            flag = 0;
        }
    }

    if (flag == 1) {
        printf("%d, ",n);
        i++;
    }
    n++;
}
}
```

Soluzione dell'esercizio 3.6

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

int main ()
{
  int segreto;
  int tentativo;

  srand(time(NULL));

  segreto = rand() % 10 + 1;

  do {
    printf ("Indovina il numero (tra 1 e 10): ");
    scanf ("%d",&tentativo);
    if (segreto < tentativo)
        printf("In numero e' piu' basso\n");
    else {
        if (segreto > tentativo)
            printf("In numero e' piu' alto\n");
    }
  } while (segreto!=tentativo);

printf("Esatto!");
}
```

Soluzione dell'esercizio 3.7

```
#include <stdio.h>

void main(){
```

```
int n,m;
int sum_n, sum_m;
int i;
scanf("%d",&n);
scanf("%d",&m);

printf("Divisori di %d: ",n);
sum_n = 0;
for (i = 1; i <= n / 2; i++) {
    if (n % i == 0) {
        printf("%d,",i);
        sum_n += i;
    }
}
printf("\n");

printf("Divisori di %d: ",m);
sum_m = 0;
for (i = 1; i <= m / 2; i++) {
    if (m % i == 0) {
        printf("%d,",i);
        sum_m += i;
    }
}
printf("\n");

if (sum_n == m && sum_m == n)
    printf("I due numeri sono amicali\n");
else
    printf("I due numeri non sono amicali\n");

}
```

Attenzione L'algoritmo che viene presentato non è efficiente. Volendo si potrebbe minimizzare il numero delle iterazioni considerate.

4 Array numerici

Questa dispensa propone esercizi sulla scrittura di algoritmi, in linguaggio C, utili alla comprensione dei vettori (*ingl.*, array). La dichiarazione di un vettore di elementi omogenei in C avviene grazie alla seguente dichiarazione:

```
void main(){
tipo_variabile nome_variabile[numero_elementi];

istruzioni;
}
```

dove `tipo_variabile` è il tipo degli elementi del vettore e `numero_elementi` è la lunghezza del vettore (considerata una costante da qui in poi). Solitamente la lunghezza del vettore viene dichiarato come una costante tramite la parola chiave `#define` per migliorare la leggibilità del codice.

Per accedere a singoli elementi dell'array utilizzeremo degli indici che varieranno tra 0 e `numero_elementi`-1. Per accedere ad un elemento del vettore si utilizzano le parentesi quadre, ad esempio:

```
void main(){
int vettore[10];

vettore[1] = 3;
}
```

il precedente programma dichiara un vettore di interi di dimensione 10 e successivamente inserisce il valore 4 nel secondo elemento del vettore.

Un array ha dimensioni fisse. Esse devono essere specificate alla dichiarazione del programma. Non è possibile modificare le dimensioni di un array durante l'esecuzione del programma.

4.1 Esercizi

Esercizio 4.1

Scrivere un programma che acquisisca un certo numero di valori interi e positivi stabilito dall'utente (massimo 10), e poi stampi a video un vettore contenente tutti i numeri pari della sequenza e un vettore contente tutti i numeri dispari.

Esercizio 4.2

Scrivere un programma che acquisisca un numero stabilito dall'utente di voti di esami e si assicuri che tutti i voti inseriti siano nell'intervallo $[18, 30]$. Il numero massimo di voti inseribili è 20, ma l'utente potrebbe decidere di inserire meno voti. Il programma dovrà poi stampare a video un sommario simile a quello di seguito. La media troncata è la media calcolata non contando gli estremi (voti minimo e massimo).

```
STATISTICHE VOTI:

Numero esami sostenuti: ...
Media: ...
Media troncata: ...
Varianza: ...
Deviazione standard: ...
```

Per il calcolo la varianza, $\hat{\sigma}^2$, seguire la seguente formula:

$$\hat{\sigma}^2 = \frac{\sum_{i=0}^{N}(v_i - \hat{v})^2}{N-1} \tag{4.1}$$

dove v_i è l'i-esimo voto e $\hat{v}$ è la media campionaria dei voti. La deviazione standard è semplicemente la radice quadrata della varianza.

Esercizio 4.3

Scrivere un progrmma che, acquisita una sequenza di al massimo 50 numeri interi tra 1 e 100, stampi a video l'istogramma dei divisori. L'istogramma deve avere tutti i valori da 2 al valore massimo immesso diviso per 2. Il programma deve considerare solo i divisori propri, dove un divisore positivo di n diverso da n stesso è chiamato divisore proprio.

```
Quanti numeri vuoi considerare: 5

Inserire il 1o numero: 6
Inserire il 2o numero: 4
Inserire il 3o numero: 3
Inserire il 4o numero: 9
Inserire il 5o numero: 12
2               | ***
3               | ***
```

```
4               | *
5               |
6               | *
```

Esercizio 4.4

Scrivere un programma che chieda all'utente di inserire un numero variabile (al massimo 10) di valori reali. Il programma dovrà assicurarsi che l'utente non possa inserire più di 10 valori.

Il programma dovrà cercare, durante l'acquisizione, la posizione del minimo e del massimo.

Infine, il programma dovrà stampare un sommario dei valori inseriti, con l'indicazione della posizione del minimo (-) e del massimo (+), il tutto formattato come indicato di seguito.

```
Quanti elementi vuoi inserire (max: 10)? 100
Il valore inserito è troppo grande!
Quanti elementi vuoi inserire (max: 10)? 4
Inserire il 1o valore: 100
Inserire il 2o valore: 20
Inserire il 3o valore: -1
Inserire il 4o valore: 5.3
  +| 100.00
   | 20.00
-  | -1.00
   | 5.30
```

Esercizio 4.5

Scrivere un programma che acquisisca un certo numero di valore stabilito dall'utente (massimo 10), e poi acquisisca un eguale quantità di indici, che specificano l'ordine in cui il programma dovrà stampare i valori acquisiti in precedenza. Per esempio:

```
Quanti elementi vuoi inserire (max: 10)? 5
Inserire il 1o valore: 10
Inserire il 2o valore: 20
Inserire il 3o valore: 30
Inserire il 4o valore: 40
Inserire il 5o valore: 50
Qual è il 1o valore che vuoi stampare? 3
Qual è il 2o valore che vuoi stampare? 2
Qual è il 3o valore che vuoi stampare? 1
Qual è il 4o valore che vuoi stampare? 4
Qual è il 5o valore che vuoi stampare? 5
30
20
10
40
50
```

Esercizio 4.6

Scrivere un programma che acquisisce un numero variabile di interi, con un massimo di 100. Il programma dovrà costruire un insieme (ossia controllare che non ci siano elementi ripetuti) a partire dagli elementi inseriti.

Esercizio 4.7

Implementare le operazioni di intersezione, unione, e differenza insiemistica tra due array acquisiti da tastiera, supponendo siano insiemi (ossia non abbiano elementi ripetuti).

Soluzioni

Soluzione dell'esercizio 4.1

```
#include <stdio.h>

#define MAX_LEN 10

void main(){

int pari[MAX_LEN];
int dispari[MAX_LEN];
int n_pari = 0, n_disp = 0;

int valore,i, n_val = -1;

while (n_val < 1){
    printf("Inserire il numero di valori da considerare: ");
    scanf("%d",&n_val);
}

for (i = 0; i < n_val; i++) {
    printf("Inserire il %do valore: ",i+1);
    scanf("%d",&valore);
    if (valore % 2 == 0) {
        pari[n_pari] = valore;
        n_pari++;
    }
    else {
        dispari[n_disp] = valore;
        n_disp++;
    }
}

printf("Numeri pari: ");
for (i = 0; i < n_pari; i++) {
    printf("%d ",pari[i]);
}
printf("\n");

printf("Numeri dispari: ");
for (i = 0; i < n_disp; i++) {
    printf("%d ",dispari[i]);
}

}
```

Soluzione dell'esercizio 4.2

```
#include <stdio.h>
#include <math.h>

#define MAX_LEN 20

void main(){

int n;
```

```
int voti[MAX_LEN];
int i;
float sum;
float media, media_troncata;
int voto_minimo = 33;
int voto_massimo = 10;
float dev, varianza, deviazione_standard;

do {
    printf("Inserire il numero degli esami: ");
    scanf("%d",&n);
} while ( n < 3 || n > 20);

sum = 0;
for (i = 0; i < n; i++) {
    do {
        printf("Inserire il voto %d: ",i+1);
        scanf("%d",&voti[i]);
    } while ( voti[i] < 18 || voti[i] > 30);

    if (voti[i] > voto_massimo)
        voto_massimo = voti[i];

    if (voti[i] < voto_minimo)
        voto_minimo = voti[i];

    sum += voti[i];
}

media = sum / n;
media_troncata = (sum - voto_massimo - voto_minimo) / (n-2);

dev = 0;
for (i = 0; i < n; i++)
    dev += (voti[i] - media) * (voti[i] - media);

//pow(voti[i] - media,2);
varianza = dev / (n - 1);
deviazione_standard = sqrt(varianza);

printf("STATISTICHE VOTI\n\n");
printf("Il numero degli esami: %d\n",n);
printf("La media e': %f\n",media);
printf("La media troncata e': %f\n",media_troncata);
printf("La varianza e': %f\n",varianza);
printf("La deviazione standard e': %f\n",deviazione_standard);


}
```

Soluzione dell'esercizio 4.3

```
#include <stdio.h>

#define INT_MAX 100
#define LEN_DIVI 50
#define N_MAX 50

void main(){
```

```
int numeri[N_MAX];
int divisori[LEN_DIVI];
int maxi = -1;
int n_val = -1;
int i,j;

while (n_val < 1 || n_val > N_MAX){
    printf("Inserire il numero di valori da considerare: ");
    scanf("%d",&n_val);
}

// Inizializzazione divisori
for (i = 0; i < LEN_DIVI; i++)
    divisori[i] = 0;

for (i = 0; i < n_val; i++) {
    //Acquisizione dati
    do {
    printf("Inserire il %do valore: ",i+1);
    scanf("%d",&numeri[i]);
    } while (numeri[i] <= 0);

    // Calcolo massimo
    if (numeri[i] > maxi)
        maxi = numeri[i];

    //Calcolo istogramma
    for (j = numeri[i] / 2; j > 1; j--)
        if (numeri[i] % j == 0)
            divisori[j-1]++;
}

for (i = 1; i < maxi / 2; i++) {
    printf("%d\t| ",i+1);
    for (j = 0; j < divisori[i]; j++)
        printf("*");
    printf("\n");
}

}
```

Soluzione dell'esercizio 4.4

```
#define DIM 10

#include <stdio.h>

int main()
{
    //dichiarazione variabili
    int i;
    int dim;
    int min_idx = 0; //assumiamo che min sia il primo elemento
    int max_idx = 0; //assumiamo che max sia il primo elemento

    float valori[DIM];

    //acquisizione della dimensione
    do {
        printf("Quanti elementi vuoi inserire (max: %d)? ", DIM);
```

```
        scanf("%d", &dim);

        if (dim > DIM)
            printf("Il valore inserito e' troppo grande!\n");
    } while (dim > DIM);

    //acquisizione valori con ricerca di min max
    for (i = 0; i < dim; i++) {
        printf("Inserire il %do valore: ", i+1);
        scanf("%f", &valori[i]);

        if (valori[i] < valori[min_idx])
            min_idx = i;

        if (valori[i] > valori[max_idx])
            max_idx = i;
    }

    //stampa dei valori con indicazione del massimo e del minimo
    for (i = 0; i < dim; i++) {
        if (i == min_idx && i == max_idx)
            printf("- +|");
        else {
            if (i == min_idx)
                printf("-  |");

            if (i == max_idx)
                printf("  +|");
        }

        if (i != min_idx && i != max_idx)
            printf("   |");

        printf(" %.2f\n", valori[i]);
    }
}
```

Soluzione dell'esercizio 4.5

```
#define DIM 10

#include <stdio.h>

int main()
{
    int i;
    int j; //indice usato solo per chiarezza, ma non necessario

    int indice[DIM];
    int valori[DIM];
    int dim;

    //acquisizione della dimensione
    do {
        printf("Quanti elementi vuoi inserire (max: %d)? ", DIM);
        scanf("%d", &dim);

        if (dim > DIM)
            printf("Il valore inserito e' troppo grande!\n");
    } while (dim > DIM);
```

```
    //acquisizione valori
    for (i = 0; i < dim; i++) {
        printf("Inserire il %do valore: ", i+1);
        scanf("%d", &valori[i]);
    }

    //acquisizione indice
    for (i = 0; i < dim; i++) {
        printf("Qual e' il %do valore che vuoi stampare? ", i+1);

        //tutti i valori dell'indice devono puntare ad elementi validi
        // quindi all'interno dei limiti 0,dim
        do {
            scanf("%d", &indice[i]);

            if (indice[i] > dim || indice[i] < 0)
                printf("Il valore inserito non e' in [0, %d]", dim);
        } while (indice[i] > dim || indice[i] < 0);

        //si assume che l'utente che vuole indicare il primo elemento (ad
        // esempio), inserisca il numero 1, quindi dobbiamo decrementare
        // tutto di 1: 1o, 2o, 3o... -> 0, 1, 2
        indice[i] = indice[i] - 1;
    }

    //stampa in ordine stabilito dall'indice
    for (i = 0; i < dim; i++)
    {
        j = indice[i];

        printf("%d\n", valori[j]);
    }
}
```

Soluzione dell'esercizio 4.6

```
#include <stdio.h>
#include <math.h>

#define MAX_LEN 100

void main(){

int n;
int lista[MAX_LEN];
int insieme[MAX_LEN];
int lungh_ins;
int i,j;
int trovato;

do {
    printf("Inserire il numero degli elementi da inserire: ");
    scanf("%d",&n);
} while ( n < 0 || n > 100);

for (i = 0; i < n; i++) {
        printf("Inserire il %do numero : ",i+1);
        scanf("%d",&lista[i]);
}
```

```
lungh_ins = 1;
insieme[0] = lista[0];
for (i = 1; i < n; i++) {
    trovato = 0;

    for (j = 0; j < lungh_ins; j++)
        if (lista[i] == insieme[j])
            trovato = 1;

    if (!trovato) {
        insieme[lungh_ins] = lista[i];
        lungh_ins++;
    }
}
printf("[ ");
for (i = 0; i < lungh_ins;i++)
    printf("%d ",insieme[i]);
printf(" ]");

}
```

Soluzione dell'esercizio 4.7

```
#define DIM 30

#include <stdio.h>

int main()
{
    // indici per scansione
    int i, j;

    // dimensione effettiva degli array
    int dim_a, dim_b;

    // flag utile per le ricerche di elementi
    int trovato;

    // array e insiemi A e B
    int lista[DIM];
    int A[DIM];
    int B[DIM];

    // dimensione effettiva degli insiemi
    int len_a = 0;
    int len_b = 0;
    int len_u = 0; //dimensione unione
    int len_i = 0; //dimensione intersezione
    int len_d = 0; //dimensione differenza

    // altri insiemi
    int unione[2*DIM];    // A u B
    int intersezione[DIM];// A ^ B
    int differenza[DIM];  // B \ A

    /*
     * ACQUISIZIONE DELLA DIMENSIONE DELLA PRIMA LISTA
     */
    do {
```

```
    printf("Quanti elementi vuoi inserire nella prima lista (max: %d)? ",
           DIM);
    scanf("%d", &dim_a);

    if (dim_a > DIM)
        printf("Il valore inserito  troppo grande!\n");
} while (dim_a > DIM);

/*
 * ACQUISIZIONE DELLA PRIMA LISTA DI ELEMENTI
 */
for (i = 0; i < dim_a; i++) {
    printf("Inserire il %do elemento: ", i+1);
    scanf("%d", &lista[i]);
}

/*
 * CONVERSIONE IN INSIEME A
 */
for (i = 0; i < dim_a; i++) {
    trovato = 0;

    //ricerco il valore i-esimo nella lista
    for (j = 0; !trovato && j < len_a; j++)
        trovato = (lista[i] == A[j]);

    //se non trovato, lo inserisco e incremento l'indice
    if (!trovato) {
        //gi che ci sono, lo inserisco anche nell'unione
        unione[len_a] = lista[i];

        A[len_a] = lista[i];
        len_a++;
    }
}

/*
 * A QUESTO PUNTO A COINCIDE CON L'INSIEME UNIONE, AL QUALE DOVR
 * AGGIUNGERE GLI ELEMENTI DI B SENZA RIPETIZIONI
 */
len_u = len_a;

// stampa l'insieme A
printf ("Insieme A = {");
for (i = 0; i < len_a; i++)
    printf("%d ", A[i]);
printf ("}\n");

/*
 * ACQUISIZIONE DELLA DIMENSIONE DELLA SECONDA LISTA
 */
do {
    printf("Quanti elementi vuoi inserire nella seconda lista (max: %d)? ",
           DIM);
    scanf("%d", &dim_b);

    if (dim_b > DIM)
        printf("Il valore inserito  troppo grande!\n");
} while (dim_b > DIM);

/*
 * ACQUISIZIONE DELLA SECONDA LISTA DI ELEMENTI
```

```
     */
    for (i = 0; i < dim_b; i++) {
        printf("Inserire il %do elemento: ", i+1);
        scanf("%d", &lista[i]);
    }

    /*
     * CONVERSIONE IN INSIEME B E CALCOLO UNIONE, INTERSEZIONE,
     * DIFFERENZA
     */
    for (i = 0; i < dim_b; i++) {
        trovato = 0;

        //ricerco il valore i-esimo nella lista
        for (j = 0; !trovato && j < len_b; j++)
            trovato = (lista[i] == B[j]);

        //se non trovato, lo inserisco e incremento l'indice
        if (!trovato) {
            B[len_b] = lista[i];
            len_b++;
        }

        /*
         * UNIONE: Tutti gli elementi in A e tutti gli elementi in B
         * senza ripetizioni.
         */
        trovato = 0;

        //ricerco il valore i-esimo nella lista
        for (j = 0; !trovato && j < len_u; j++)
            trovato = (unione[j] == lista[i]);

        //se non trovato, allora va aggiunto all'unione
        if (!trovato) {
            unione[len_u] = lista[i];
            len_u++;
        }

        /*
         * INTERSEZIONE: Tutti gli elementi che sono sia in A che in
         * B, senza ripetizioni.
         */
        trovato = 0;

        //uso len_b-1 perch len_b  gi stato incrementato
        for (j = 0; !trovato && j < len_a; j++)
            trovato = (A[j] == B[len_b-1]);

        //trovato anche in A: allora fa parte dell'intersezione
        if (trovato) {
            trovato = 0;

            //controllo per non inserire duplicati in intersezione
            for (j = 0; !trovato && j < len_i; j++)
                trovato = (intersezione[j] == B[len_b-1]);

            if (!trovato) {
                intersezione[len_i] = B[len_b-1];
                len_i++;
            }
        } else { //non trovato in A: allora fa parte della differenza
```

```
            /*
             * DIFFERENZA: Tutti gli elementi che sono in B, tranne
             * quelli che sono in A.
             */
            trovato = 0;

            //controllo per non inserire duplicati in differenza
            for (j = 0; !trovato && j < len_d; j++)
                trovato = (differenza[j] == B[len_b-1]);

            if (!trovato) {
                differenza[len_d] = B[len_b-1];
                len_d++;
            }
        }
    }

    // stampa l'insieme B
    printf ("B = {");
    for (i = 0; i < len_b; i++)
        printf("%d ", B[i]);
    printf ("}\n");

    // stampa l'insieme unione
    printf ("unione = {");
    for (i = 0; i < len_u; i++)
        printf("%d ", unione[i]);
    printf ("}\n");

    // stampa l'insieme intersezione
    printf ("intersezione = {");
    for (i = 0; i < len_i; i++)
        printf("%d ", intersezione[i]);
    printf ("}\n");

    // stampa l'insieme differenza
    printf ("differenza = {");
    for (i = 0; i < len_d; i++)
        printf("%d ", differenza[i]);
    printf ("}\n");
}
```

5 Stringhe

Le stringhe di caratteri sono gestite in C come dei vettori di `char` con alla fine un "tappo" dato dal carattere `'\0'`. E' possibile acquisire un'intera stringa di caratteri in una sola istruzione grazie all'istruzione `scanf('%s',variabile);` oppure `gets(variabile)`. Entrambi devono essere seguiti dall'istruzione `fflush(stdin);` che serve ad evitare errori nella successiva acquisizione di stringhe. Mentre l'istruzione `scanf('%s',variabile);` ferma la sua acquisizione al primo spazio, `gets(variabile)` inserisce nella variable tutto l'input fino al primo invio.

Poichè le stringhe necessitano di un tappo, ogni volta che ne dichiaro una devo aggiungere un elemento per il tappo, ossia se voglio una stringa lunga `N_MAX` dovrò dichiarare un array di `N_MAX+1` elementi.

Esiste una libreria di C che gestisce le stringhe `string.h` essa ci permette di:

- ottenere la lunghezza di una stringa con la funzione `strlen(stringa);`
- confrontare due stringhe `strcmp(stringa1,stringa2)` e restituisce zero se sono uguali, un numero negativo se `stringa1` viene prima in ordine lessicografico di `stringa2` e un numero positivo se `stringa2` viene prima in ordine lessicografico di `stringa1` (se i caratteri sono tutti maiuscoli o tutti minuscoli);
- copiare una stringa `stringa2` in un'altra stringa `stringa1` con la funzione `strcpy(stringa1,stringa2);`
- concatenare due stringhe con la funzione `strcat(stringa1,stringa2);`

5.1 Esercizi

Esercizio 5.1

Implementare la stessa funzionalità della `strcat()` senza ovviamente utilizzare la funzione stessa e senza utilizzare la `strlen()`. Considerare delle stringhe in ingresso di al massimo 20 caratteri.

Esercizio 5.2

1. Scrivere un programma che determini se una frase acquisita da tastiera è palindroma. Controllare se sono palindrome le seguenti frasi (senza spazi e maiuscole):
   - Eri un nano non annuire
   - Ad una vera pia donna dei simili fili misi e annodai: pareva nuda
   - O mordo tua nuora, o ari un autodromo
   - Occorre portar aratro per Rocco
2. (Bonus) Modificare il programma precedente per considerare stringhe in input contenenti anche caratteri speciali e spazi.

Esercizio 5.3

Implementare la seguente variante del cifrario di Cesare[1]. Dopo aver acquisito una messaggio di massimo 160 caratteri (alfabetici minuscoli), il programma dovrà chiedere all'utente una chiave (numero intero K, $1 < K < 25$).

Il programma stamperà il messaggio cifrato, ottenuto traslando ogni lettera di K posizioni in avanti (dove K è la chiave).

Dopo aver stampato il messaggio cifrato, il programma chiede all'utente di inserire un messaggio (che si assume sia stato cifrato con lo stesso algoritmo di cui sopra). Tale messaggio sarà dunque decifrato dal programma, il quale dovrà svolgere l'operazione inversa.

Infine, il messaggio decifrato sarà stampato a video.

Esercizio 5.4

[1] `http://it.wikipedia.org/wiki/Cifrario_di_Cesare`

1. Scrivere un programma che converta una stringa in alfabeto farfallino. Nell'alfabeto farfallino, ogni vocale è raddoppiata, e tra le due vocali è inserita la lettera 'f'. Ad esempio, "ciao" diventa "cifiafaofo".

2. (bonus) Scrivere il medesimo programma utilizzando soltanto una variabile per memorizzare la stringa tradotta (i.e., senza copiare la traduzione in una nuova variabile).

Esercizio 5.5

Scrivere un programma che considera, da una stringa letta da tastiera, solo le lettere maiuscole e minuscole. Il programma dovrà poi calcolare le occorrenze di ogni carattere nella stringa. Le occorrenze saranno poi stampate a video a video come istogramma con gli asterischi. Si assuma che la stringa possa avere una dimensione massima di 256 elementi.

Si veda l'esempio seguente.

```
str = Ciao Come Stai? Non c'e' male GRAZIE! Mi trovo molto bene in questa citta'.

a |
b | *
c | **
d |
e | ******
f |
g |
h |
i | *****
j |
k |
l | **
m | ***
n | ***
o | *******
p |
q | *
r | *
s | *
t | ******
u | *
v | *
w |
x |
y |
A |
B |
C | **
D |
E | *
F |
G | *
H |
I | *
J |
K |
```

```
L |
M | *
N | *
O |
P |
Q |
R | *
S | *
T |
U |
V |
W |
X |
Y |
```

Esercizio 5.6

Scrivere un programma che determini se due parole acquisite da tastiera sono una l'anagramma dell'altra.

Soluzioni

Soluzione dell'esercizio 5.1

```
#include <stdio.h>
#include <string.h>

#define MAX_LEN 100

void main(){

char str1[MAX_LEN+1],str2[MAX_LEN+1];
char str12[2*MAX_LEN+1];
int lungh1,lungh2;
int i;

printf("Inserire la prima stringa: ");
gets(str1);
fflush(stdin);

printf("Inserire la seconda stringa: ");
gets(str2);
fflush(stdin);

//for (lungh1 = 0; str1[lungh1] == '/0'; lungh1++)
lungh1 = 0;
for (i = 0; str1[i] != '\0'; i++)
    lungh1++;

lungh2 = 0;
for (i = 0; str2[i] != '\0'; i++)
    lungh2++;

//printf("%d   %d",lungh1,lungh2);


for (i = 0; i <= lungh1; i++)
    str12[i] = str1[i];

for (i = 0; i < lungh2; i++)
    str12[i+lungh1+1] = str2[i];

str12[lungh1+lungh2] = '\0';

printf("%s",str12);


}
```

Soluzione dell'esercizio 5.2

```
#include <stdio.h>
#include <string.h>

#define LEN 256

void main(){
```

```
char parola[LEN+1];
int i,j,palindromo;
int n;

printf("Inserire la frase: ");
gets(parola);
fflush(stdin);

n_car = strlen(parola),

palindromo = 1;
j = n_car - 1;
for (i = 0; (i < n_car/2) && (palindromo == 1); i++) {
    if (parola[i] != parola[j])
        palindromo = 0;
        j--;
}

if (palindromo == 1)
    printf("E' palindromo");
else
    printf("Non e' palindromo");

}
```

Soluzione dell'esercizio 5.3

```
#include <stdio.h>
#include <string.h>

#define MAX_LEN 160


void main(){

char messaggio[MAX_LEN+1;
char cifrato[MAX_LEN+1];

int lungh;
int chiave;
int i;

printf("Inserire un messaggio: ");
gets(messaggio);
fflush(stdin);
lungh = strlen(messaggio);

do {
    printf("Inserire una chiave: ");
    scanf("%d",&chiave);
} while ( chiave < 1 || chiave > 25 );
fflush(stdin);

for (i = 0; i <= lungh; i++) {
    if (messaggio[i] > 96 && messaggio[i] < 123)
        cifrato[i] = (messaggio[i] - 97 + chiave) % 26 + 97;
        cifrato[i] = messaggio[i] + chiave;
    else
        cifrato[i] = messaggio[i];
```

```
}

printf("Messaggio cifrato: ");
printf("%s\n",cifrato);

printf("Inserire un messaggio cifrato ");
gets(cifrato);
fflush(stdin);

for (i = 0; i <= lungh; i++) {
    if (cifrato[i] > 96 && cifrato[i] < 123)
        messaggio[i] = (cifrato[i] - 97 + (26 - chiave) ) % 26 + 97;
    else
        messaggio[i] = cifrato[i];

}

printf("Messaggio in chiaro: ");
printf("%s\n",messaggio);

}
```

Soluzione dell'esercizio 5.4

1.

```
#include <stdio.h>
#include <string.h>

#define LEN 256


void main(){

char parola[LEN+1];
char parola_tradotta[LEN+1];
int n,i,j;
int accettabile;

//Acquisizione
printf("Inserire la parola da tradurre");
scanf("%s",parola);

n = strlen(parola);

j = 0;
accettabile = 1;

//Traduzione
for (i = 0; i <= n; i++) {
    // Copio lettera
    if (j < LEN+1) {
        parola_tradotta[j] = parola[i];
        if (parola[i] == 'a' || parola[i] == 'e' || parola[i] == 'i' || parola[i
            ] == 'o' || parola[i] == 'u') {
            parola_tradotta[j+1] = 'f';
            parola_tradotta[j+2] = parola[i];
            j += 3;
        }
        else
            j++;
```

```
    }
    else
        accettabile = 0;

}
if (accettabile == 1)
    printf("%s",parola_tradotta);
else
    printf("Parola troppo lunga da tradurre");

}
```

2.

```
#define LEN 256

#include <stdio.h>
#include <string.h>

int main () {

    int i, k, len;
    char str[LEN];

    /* Acquisizione */
    do {
        printf("Inserire una frase da tradurre: ");
        gets(str);
        len = strlen(str);
    } while (len > LEN);

    /* Lettura da sx a dx */
    for (i = 0; i <= len; i++) {
        if (str[i] == 'a'||
            str[i] == 'e'||
            str[i] == 'i'||
            str[i] == 'o'||
            str[i] == 'u') {

            /* Shift a dx di 2 posizioni */
            for(k = len; k > i ; k--)
                str[k + 2] = str[k];

            str[i+1] = 'f';
            str[i+2] = str[i];

            i = i + 2;
            len = len + 2;
        }
    }

    printf("%s\n", str);
}
```

Soluzione dell'esercizio 5.5

```
#define LEN 256

#include <stdio.h>
#include <string.h>
```

```
/*
  Tabella ascii:

  0 nul    1 soh    2 stx    3 etx    4 eot    5 enq    6 ack    7 bel
  8 bs     9 ht    10 nl    11 vt    12 np    13 cr    14 so    15 si
 16 dle   17 dc1   18 dc2   19 dc3   20 dc4   21 nak   22 syn   23 etb
 24 can   25 em    26 sub   27 esc   28 fs    29 gs    30 rs    31 us
 32 sp    33  !    34  "    35  #    36  $    37  %    38  &    39  '
 40  (    41  )    42  *    43  +    44  ,    45  -    46  .    47  /
 48  0    49  1    50  2    51  3    52  4    53  5    54  6    55  7
 56  8    57  9    58  :    59  ;    60  <    61  =    62  >    63  ?
 64  @    65  A    66  B    67  C    68  D    69  E    70  F    71  G
 72  H    73  I    74  J    75  K    76  L    77  M    78  N    79  O
 80  P    81  Q    82  R    83  S    84  T    85  U    86  V    87  W
 88  X    89  Y    90  Z    91  [    92  \    93  ]    94  ^    95  _
 96  `    97  a    98  b    99  c   100  d   101  e   102  f   103  g
104  h   105  i   106  j   107  k   108  l   109  m   110  n   111  o
112  p   113  q   114  r   115  s   116  t   117  u   118  v   119  w
120  x   121  y   122  z   123  {   124  |   125  }   126  ~   127 del
*/

int main()
{
    int i, j;
    int ast;

    int hist[25]; //z-a -> 90-65 -> 25
    int HIST[25]; //Z-A -> 122-97 -> 25

    char lettera;
    char str[LEN];

    //inizializzazione dell'istogramma
    for (i = 0; i < 25; i++)
        hist[i] = HIST[i] = 0;

    //acquisizione stringa
    printf("str = ");
    gets(str);

    printf("\n");

    /*
     * Esempio:
     * str[3] = 'd'
     *
     * Bisogna incrementare l'istogramma in posizione 4
     * hist['d'-'a'] = hist[100-97] = hist[3]
     */
    for (i = 0; i < strlen(str); i++) {
        if (str[i] > 'a' && str[i] < 'z')
            hist[str[i]-'a']++;

        if (str[i] > 'A' && str[i] < 'Z')
            HIST[str[i]-'A']++;
    }

    //per ogni lettera
    for (i = 0; i < 25*2; i++)
    {
        //stampa il giusto numero di asterischi
        if (i < 25) {
```

```
            ast = hist[i];
            lettera = 'a' + i;
        } else {
            ast = HIST[i-25];
            lettera = 'A' + i - 25;
        }

        printf("%c | ", lettera);

        for (j = 0; j < ast; j++)
            printf("*");

        printf("\n");
    }
}
```

Soluzione dell'esercizio 5.6

```
#include <stdio.h>
#include <string.h>

#define MAX_LEN 100
void main(){

char stringa1[MAX_LEN];
char stringa2[MAX_LEN];

int i,j;
int trovato, uguali;

printf("Inserire la prima stringa: ");
scanf("%s",&stringa1);
fflush(stdin);
printf("Inserire la seconda stringa: ");
scanf("%s",&stringa2);
fflush(stdin);

uguali = 1;
if (strlen(stringa1) != strlen(stringa2))
    uguali = 0;

for (i = 0; i < strlen(stringa1) && uguali; i++) {
    trovato = 0;
    for (j = 0; j < strlen(stringa2) && !trovato; j++)
        if (stringa1[i] == stringa2[j]){
            stringa2[j] = '-';
            trovato = 1;
        }

    if (trovato == 0)
        uguali = 0;
}

if (uguali)
    printf("Le due stringhe sono una l'anagramma dell'altra");
else
    printf("No");
}
```

6 Typedef, matrici e codifica

Questa dispensa propone esercizi sulla scrittura di algoritmi, in linguaggio C, utili alla comprensione della definizione di matrici, strutture e di tipi, oltre ad un riepilogo sulla codifica dei numeri con complemento a due e IEEE standard.

I tipi di dato strutturato sono dichiarabili in C tramite la parola chiave `typedef`

```
#define LEN n_elementi

void main(){

struct {
        tipo_campo1 nome_campo1;
        tipo_campo2 nome_campo2;
        ...
} nome_struttura;

istruzioni;
}
```

A questo punto potremo utilizzare `nome_struttura` come se fosse una variabile. Per accedere ai campi della struttura si utilizza la sintassi `nome_struttura.nome_campo`. Grazie ad esso possiamo accedere (leggere o scrivere) nei campi della struttura. L'assegnamento tra strutture è possibile solo se le strutture sono dello stesso tipo (stessi campi, per nome e tipo, con stesso ordine di dichiarazione). Non è possibile fare confronti tra strutture in maniera automatica.

Oltre ad i tipi nativi del C (ad esempio `int`, `char` o `float`) è possibile dichiarare dei tipi definiti dall'utente tramite la parola chiave `typedef`. Ad esempio, volendo dichiarare dei vettori della stessa dimensione `n_elementi`:

```
#define LEN n_elementi

void main(){

typedef tipo_variabile nome_variabile[LEN];


istruzioni;
}
```

E' buona norma dichiarare nuovi tipi con nomi che inizino con una lettera maiuscola, con un prefisso o con un suffisso scelto.

Le matrici in C vengono definite come vettori di vettori. Ad esempio per dichiarare una matrice bidimensionale dovremo scrivere:

```
#define DIM1 n_righe
#define DIM2 n_colonne

void main(){

tipo_variabile nome_variabile[DIM1][DIM2];

istruzioni;
}
```

dove `n_righe` e `n_colonne` sono degli interi che specificano il numero di righe e colonne della matrice. Nel caso di dimensioni superiori a 2, dovremo mettere tante parentesi quadre e specificare le grandezze massime per ogni dimensione.

La tecnica della dichiarazione di strutture si può combinare con la dichiarazione di matrici. Ad esempio:

```
#define LEN n_elementi

void main(){

typedef tipo_variabile Nome_tipo[LEN][LEN];

typedef struct {
        tipo_campo1 nome_campo1;
        tipo_campo2 nome_campo2;
        ...
} Nome_tipo_struttura;

istruzioni;
}
```

La definizione di un tipo non implica l'istanziazione di una variabile di quel tipo. La dichiarazione della variabile avverrà di seguito.

Riepilogo: CP2 Non cambia la codifica dei numeri positivi. La cifra più significativa ha significato -2^{m-1} se il numero è rappresentato da m bit.

Decimale -> CP2

- Controllo se il numero è rappresentabile con m bit

- Se è positivo, converto il numero in binario con l'algoritmo delle divisioni successive
- Se è negativo, considero il suo opposto, lo codifico in binario, inverto ogni bit, sommo 1

CP2 -> Decimale

- Se è positivo, converto il numero da binario a decimale (somma di esponenti di 2)
- Se è negativo, inverto ogni bit, aggiungo 1, converto in decimale, cambio di segno

La somma di numeri espressi in complemento a due si esegue normalmente, si ignora il carry (cifra significativa che esce dagli m bit). Abbiamo overflow se c'è inconsistenza nell'operazione (per esempio se la somma di negativi dà un numero positivo).

Riepilogo: IEEE 754-1985 standard Decimale -> IEEE standard

- Definisco il bit di segno $S = 0$ per positivo e $S = 1$ per negativo
- Codifico in virgola fissa in base 2, parte frazionaria e parte intera
- Porto il numero binario in forma normalizzata in base 2
- Definisco a 23 bit come la mantissa M senza il primo bit (sempre 1)
- Calcolo l'esponente E (aggiungo 127 a quello che ho trovato durante la normalizzazione)
- Compongo il numero $S/E/M$

IEEE standard -> Decimale

- Decompongo il numero $S/E/M$
- Calcolo l'esponente E e sottraggo 127
- Sposto la virgola in base all'esponente alla mantissa M
- Converto in decimale la parte intera
- Converto in decimale la parte frazionaria
- Definisco il bit di segno $S = 0$ per positivo e $S = 1$ per negativo

6.1 Esercizi

Esercizio 6.1

Scrivere un programma che permetta all'utente di gestire un libretto di voti di esami, per un massimo di 20 esami. Ogni esame sostenuto ha i seguenti attributi:

- nome del corso (stringa di massimo 256 caratteri)
- voto (minimo 18, massimo 30)
- data (in formato gg/mm/aaaa)
- codice del corso (codice di massimo 6 cifre)

Il programma permette all'utente di svolgere le seguenti operazioni:

inserimento: inserisce un esame in fondo alla lista degli esami eventualmente presenti.

stampa: stampa tutti gli esami presenti, con i dettagli di cui sopra.

ricerca: chiede all'utente di inserire un codice e cerca nel libretto la presenza di un esame corrispondente a quel codice. Se presente, stampa tale esame.

uscita: esce dal programma.

tramite un menù di scelta di questo tipo:

```
[i] inserimento nuovo esame
[s] stampa del libretto
[r] ricerca per codice
[x] uscita
```

Il programma deve continuare a presentare tale menù, fino a quando l'utente non preme il tasto per uscire.

Esercizio 6.2

Scrivere un programma che chieda all'utente la dimensione effettiva di una matrice quadrata, che dovrà poi essere acquisita dal programma.

Successivamente, il programma dovrà controllare se la matrice è diagonale (i.e., nessuno zero sulla diagonale principale e solo zeri fuori da essa).

Estendere il programma per controllare anche se la matrice inserita è simmetrica.

Esercizio 6.3

(TdE Febbraio 2012, variante) Data m una matrice di dimensione $n \times n$ (costante simbolica) di numeri interi nell'intervallo $[0, 9]$:

1. si scriva un frammento di programma in linguaggio C (con le relative dichiarazioni di variabili necessarie al corretto funzionamento), che trovi il numero più frequente (si ipotizzi che tale numero sia unico). Il programma deve stampare a schermo tutti i numeri presenti nella matrice con valore minore del valore più frequente e successivamente tutti quelli con valore maggiore di quello più frequente;
2. Aggiungere al programma di cui sopra un frammento di codice che legga tutti e soli i valori memorizzati nel secondo dei due vettori (che potrebbero essere meno della lunghezza massima del vettore) e stampi a video se sono o meno monotoni crescenti, ovvero se tutti gli elementi adiacenti sono ordinati $a_i \leq a_{i+1}$.

Esercizio 6.4

(a) Si dica qual è l'intervallo di valori interi rappresentabile con la codifica in complemento a due a 9 bit.

(b) Con riferimento a tale codifica indicare, giustificando brevemente le risposte, quali delle seguenti operazioni possono essere effettuate correttamente:

- $-254 - 255$
- $+254 - 253$
- $-18 + 236$
- $+217 + 182$

(c) Mostrare in dettaglio come avviene il calcolo delle operazioni (i) e (ii), evidenziando il bit di riporto e il bit di overflow così ottenuti.

Esercizio 6.5

(TdE November 2012)

1. Si determini la codifica binaria dei numeri -12.625 e 16.65 secondo lo Standard IEEE 754-1985 a precisione singola, riportando i calcoli effettuati.

2. L'uso della rappresentazione IEEE 754-1985 a precisione singola ha comportato delle approssimazioni?

3. L'uso della rappresentazione IEEE 754-1985 a precisione doppia avrebbe portato ad una rappresentazione più precisa dei due numeri considerati?

Si ricorda che lo standard IEEE 754-1985 a precisione singola ha le seguenti caratteristiche: 1 bit per il segno, 23 bit per la mantissa, 8 per l'esponente (K=127) Si ricorda che lo standard IEEE 754-1985 a precisione doppia invece ha le seguenti caratteristiche: 1 bit per il segno, 52 bit per la mantissa, 11 per l'esponente (K=1023)

Esercizio 6.6

Scrivere un programma che converta un numero intero positivo da decimale a binario avendo a disposizione 10 bit, se possibile.

Soluzioni

Soluzione dell'esercizio 6.1

```
#include <stdio.h>

#define DIM_LIBR 20
#define DIM_NOME 256
//#define MIN_VOTO 18
//#define MAX_VOTO 30
#define DIM_DATA 10
#define DIM_CODI 6

void main(){

char menu;
int n_esami = 0;
int i,j;

typedef struct{
    char nome_corso[DIM_NOME+1];
    int voto;
    char data[DIM_DATA+1];
    char codice[DIM_CODI+1];
} Esame;

Esame libretto[DIM_LIBR];

do {
    system("cls"); //WINDOWS
    //system("clear"); //UNIX
    printf("Scegliere una delle seguenti opzioni:\n");
    printf("[i] inserimento nuovo esame\n");
    printf("[s] stampa del libretto\n");
    printf("[r] ricerca per codice\n");
    printf("[x] uscita\n");
    scanf("%c",&menu);
    fflush(stdin);

    switch (menu) {
        case 'i':
            //Istruzioni
            printf("Inserisci nome esame: ");
            scanf("%s",libretto[n_esami].nome_corso);
            printf("Inserisci voto: ");
            scanf("%d",&libretto[n_esami].voto);
            printf("Inserisci data: ");
            scanf("%s",libretto[n_esami].data);
            printf("Inserisci codice: ");
            scanf("%s",libretto[n_esami].codice);
            n_esami++;

            break;
        case 's':
            //Istruzioni
            for (i = 0, i < n_esami, i++)
                printf() ....

            break;
        case 'r':
```

```
            //Istruzioni
            printf("Inserisci il codice da controllare: ");
            scanf("%s",codice);
            for (i = 0; i < n_esami; i++) {
                //libretto[i].codice == codice;
                is_equal = 1;
                for (j = 0; j < DIM_CODI; j++)
                    if (libretto[i].codice[j] != codice[j])
                        is_equal = 0;

                if (is_equal)
                    //Stampare esame
            }


            break;
        default:
            break;
    }
} while (menu != 'x');

}
```

Soluzione dell'esercizio 6.2

```
#include <stdio.h>

#define MAX_DIM 20


void main(){

typedef enum{falso, vero} Booleano;

int matrice[MAX_DIM][MAX_DIM];
int dim_matr;
int i,j;

Booleano is_diagonale, is_simmetrica;

do {
    printf("Inserire la dimensione della matrice quadrata: ");
    scanf("%d",&dim_matr);
} while (dim_matr < 0 || dim_matr > MAX_DIM);

for(i = 0; i < dim_matr; i++)
    for(j = 0; j < dim_matr; j++) {
        printf("Inserire l'elemento in posizione [%d][%d]: ",i+1,j+1);
        scanf("%d",&matrice[i][j]);
    }

is_diagonale = 1;

for (i = 0; i < dim_matr; i++) {
    if (matrice[i][i] == 0)
        is_diagonale = 0;
    }

if (is_diagonale) {
    for(i = 0; i < dim_matr; i++)
```

```
        for(j = 0; j < dim_matr; j++) {
            if (i != j && matrice[i][j] != 0)
                is_diagonale = 0;
        }
}
is_simmetrica = 1;
for(i = 0; i < dim_matr; i++) {
    for(j = i+1; j < dim_matr; j++) {
        if (matrice[i][j] != matrice[j][i])
            is_simmetrica = 0;
    }
}

is_diagonale = -2;

if (is_diagonale)
    printf("La matrice e' diagonale e simetrica\n");
else {
    if (is_simmetrica)
        printf("La matrice non e' diagonale, ma simmetrica\n");
    else
        printf("La matrice non e' diagonale, ne' simmetrica\n");
}


}
```

Soluzione dell'esercizio 6.3

```
#define N 5       // dimensione matrice
#define RANGE 10  // da 0 a 9

#include <stdio.h>

int main () {
    int m[N][N];
    int v1[N*N], v2[N*N];
    int freq[RANGE];
    int fmax = 0;
    int valmax;

    int u, z;

    int i, j;

    //inizializzo matrice con valori crescenti (non necessario
    // se non per evitare di inserire a mano i valori)
    for (i = 0; i < N; i++) {
        for (j = 0; j < N; j++) {
            m[i][j] = i * j;
            printf("%2d ", m[i][j]);
        }
        printf("\n");
    }

    //inizializzo vettore frequenze
    for (i = 0; i < RANGE; i++)
        freq[i] = 0;

    for (i = 0; i < N; i++) {
```

```
        for (j = 0; j < N; j++) {
            freq[m[i][j]]++;
        }
    }

    //ricerco valore piu frequente
    for (i = 0; i < RANGE; i++)
        if (i == 0 || freq[i] > fmax) {
            valmax = i;    //valore
            fmax = freq[i]; //frequenza
        }

    //spostamento valori
    u = z = 0;

    for (i = 0; i < N; i++) {
        for (j = 0; j < N; j++) {
            if (m[i][j] < valmax) {
                v1[u] = m[i][j];
                u++;
            }

            if (m[i][j] > valmax) {
                v2[z] = m[i][j];
                z++;
            }
        }
    }

    //stampa valori
    printf("fmax = %d, valmax = %d\n", fmax, valmax);
    for (i = 0; i < u; i++)
        printf("%d ", v1[i]);

    printf(" (%d elementi)\n", u);

    for (i = 0; i < z; i++)
        printf("%d ", v2[i]);

    printf(" (%d elementi)\n", z);

    //controllo se v2 e' monotono
    i = 0;
    while (i < z-1 && v2[i] <= v2[i+1])
        i++;

    //uscita prematura dal ciclo?
    if (i < z-1)
        printf("Vettore NON monotono crescente.");
    else
        printf("Vettore monotono crescente.");
}
```

Soluzione dell'esercizio 6.4

(a) I valori rappresentabili vanno da $-2^{m-1} = -256$ a $2^{m-1} - 1 = +255$ compresi.

(b) Le soluzioni delle operazioni sono:

- $-254-255$ NO si ottiene un valore negativo troppo grande in valore assoluto
- $+254-253$ SI si ottiene un valore piccolo in valore assoluto
- $-18+236$ SI si ottiene un valore positivo, grande in valore assoluto ma nei limiti
- $+217+182$ NO si ottiene un valore positivo troppo grande in valore assoluto

(c)

- In complemento a 2 a $m = 9$ bit, $(-254)_{10} = (100000010)_{CP2}$ (perchè $(254)_{10} = (011111110)_2$). Da questo possiamo ricavare che $(-255)_(10) = (100000001)_{CP2}$ essendo $-255 = -254 - 1$. La somma diventa:

		(1)								
		1	0	0	0	0	0	0	1	0	+
		1	0	0	0	0	0	0	0	1	=
[1]	(1)	0	0	0	0	0	0	0	1	1	

- 254 in complemento a due è $(011111110)_{CP2} = (011111110)_2$ essendo positivo. invece $-253 = -254 + 1 = (100000011)_{CP2}$, quindi:

		(1)	(1)	(1)	(1)	(1)	(1)	(1)	(1)		
		0	1	1	1	1	1	1	1	0	+
		1	0	0	0	0	0	0	1	1	=
[0]	(1)	0	0	0	0	0	0	0	0	1	

ignorando il carry abbiamo il risultato esatto $(000000001)_{CP2} = (1)_{10}$.

Soluzione dell'esercizio 6.5

Per quanto riguarda -12.625

- Definisco il bit di segno: $S = 1$
- Codifico in virgola fissa in base 2, parte frazionaria $(0.625)_{10} = (0.101)_2$ e parte intera $(12)_{10} = (1100)_2$ ovvero 1100.101
- Porto il numero binario in forma normalizzata: 1.100101×2^3
- Definisco a 23 bit come la mantissa $M = 10010100000000000000000$
- Calcolo l'esponente $E = 127 + 3 = (130)_{10} = (010000010)_2$
- Compongo il numero 1 010000010 10010100000000000000000

Il numero è già rappresentato in maniera esatta in precisione singola. La precisione doppia non cambierà il numero (non aumenterà la precisione della rappresentazione).

Per quanto riguarda 16.65

- Definisco il bit di segno: $S = 0$
- Codifico in virgola fissa in base 2, parte frazionaria $(0.65)_{10} = (0.10\overline{1001})_2$ e parte intera $(16)_{10} = (10000)_2$ ovvero $10000.10\overline{1001}$
- Porto il numero binario in forma normalizzata: $1.00001010\overline{1001} \times 2^5$
- Definisco a 23 bit come la mantissa $M = 00001010011001100110011$
- Calcolo l'esponente $E = 127 + 5 = (132)_{10} = (010000100)_2$
- Compongo il numero $0\ 010000100\ 00001010011001100110011$

Il numero non è rappresentato in maniera esatta in precisione singola, ed essendo periodico non lo può essere neanche in precisione doppia. La precisione doppia avrà una rappresentazione più precisa del numero.

Soluzione dell'esercizio 6.6

```
#include <stdio.h>
#include <math.h>

#define MAX_DIGIT 10

void main() {

int n_10, temp, i, nDigit, maxVal;
int n_2[MAX_DIGIT];

maxVal = pow(2 , MAX_DIGIT) - 1;
do
{
    printf("Inserire il nr base 10: ");
    scanf("%d" , &n_10);
}while(n_10 > maxVal || n_10 <0);

temp = n_10;
nDigit = 0;

while(temp > 0)
{
    n_2[nDigit] = temp % 2;
    temp = temp / 2;
    nDigit++;
}

printf("%d in base 2 diventa: " , n_10);
for(i = nDigit - 1 ; i >= 0 ; i--)
    printf("%d", n_2[i]);

}
```

7 Riepilogo

Questa dispensa propone esercizi riepilogativi sui concetti visti finora ovvero:

- costrutti condizionali (`if`,`switch`);
- costrutti iterativi (`for`,`while`);
- dichiarazione di vettori e matrici;
- dichiarazione di dati strutturati (`struct`);
- dichiarazione di nuovi tipi (`typedef`, `enum`);

7.1 Esercizi

Esercizio 7.1

(TdE Novembre 2006) Le seguenti dichiarazioni definiscono un tipo di dato che rappresenta una matrice quadrata di dimensione `DIM` (non indicata nel codice proposto, ma che deve essere precisata per ottenere del codice compilabile) e una variabile `m` di quel tipo.

```
#define DIM ... /* dimensione della matrice, da precisare */

typedef int MatriceQuadrata [DIM][DIM];
MatriceQuadrata m;
int s,p;
```

Scrivere un frammento di codice che permetta di calcolare nelle variabili:

- `s` la somma dei prodotti degli elementi delle due diagonali della matrice, presi ordinatamente;
- `p` il prodotto delle somme degli elementi delle due diagonali della matrice, presi ordinatamente;

Per esempio, se `DIM` avesse valore 5 e la matrice `m` fosse la seguente:

3	2	1	5	8
2	5	1	6	4
12	4	2	6	7
5	2	13	6	8
7	3	1	4	1

allora:

$\texttt{s} = 3 \cdot 8 + 5 \cdot 6 + 2 \cdot 2 + 6 \cdot 2 + 1 \cdot 7$

$\texttt{p} = (3 + 8) \cdot (5 + 6) \cdot (2 + 2) \cdot (6 + 2) \cdot (1 + 7)$

Esercizio 7.2

(TdE November 2007) Le seguenti dichiarazioni definiscono tipi di dati relativi alla categoria degli impiegati di un'azienda (gli impiegati possono essere di prima, seconda, ..., quinta categoria), agli uffici occupati da tali impiegati, all'edificio che ospita tali uffici (edificio diviso in 20 piani ognuno con 40 uffici).

```
/* definizioni dei tipi */
```

```
typedef struct {
  char nome[20], cognome[20];
  int cat; // contiene valori tra 1 e 5
  int stipendio;
} Impiegato;

typedef enum{
  nord, nordEst, est, sudEst, sud, sudOvest, ovest, nordOvest
} Esposizione;

typedef struct {
  int superficie; /* in m^2 */
  Esposizione esp;
  Impiegato occupante;
} Ufficio;

/* definizioni delle variabili */
Ufficio torre[20][40];

/* rappresenta un edificio di 20 piani con 40 uffici per piano */
```

1. Si scriva un frammento di codice (che includa eventualmente anche le dichiarazioni di ulteriori variabili) che stampi il cognome, lo stipendio e la categoria di tutte e sole le persone che occupano un ufficio orientato a sud oppure a sudEst e avente una superficie compresa tra 20 e 30 metri quadri;

2. Visualizzi a schermo i piani che non hanno neanche un ufficio esposto a nord;

3. Stampi lo stipendio medio degli impiegati in questi piani che si chiamano "Giacomo" di nome.

Esercizio 7.3

Assumendo che `c1` e `c2` siano due variabili di tipo char, che memorizzano rispettivamente i valori `'e'` ed `'m'` indicare, per ognuna delle espressioni logiche:

1. se l'espressione è vera o falsa (per i valori delle variabili sopra indicati);

2. se è sempre vera per qualsiasi valore che le due variabili possono assumere;

3. se è sempre falsa per qualsiasi valore che le due variabili possono assumere.

Si fornisca una giustificazione per ogni risposta inserita nella tabella (risposte senza giustificazione potranno essere considerate nulle).

1. `((c1 != 'e') && (c2 == 'm')) || ((c1 != 'h') && (c2 == 'm'))`

2. `(c1 < 'g') || !((c1 <= 'g') && (c1 != 'g'))`
3. `(c1 <= 'm') || ((c2 > 'm') || !(c2 > c1))`

Esercizio 7.4

Scrivere un programma che inizializzi una matrice 10×10 con valori crescenti, in modo tale che $m[i][j] > m[i-1][j-1]$.

Dopodiché, il programma dovrà stampare la matrice seguendo un percorso a spirale in senso orario, a partire dalla cella $m[0][0]$.

Per semplicità si può assumere la matrice quadrata.

Esercizio 7.5

Scrivere un programma che inizializzi una matrice m di dimensione `DIM` $\times$ `DIM` fissata nel programma, `DIM`, con valori crescenti, in modo tale che $m[i][j] > m[i-1][j-1]$.

Dopodiché, il programma dovrà acquisire una matrice s di dimensione `DIMS` $\times$ `DIMS` fissate nel programma, `DIMS` $<$ `DIM`.

1. Scrivere un'opportuno frammento di codice che determini se s è una sottomatrice di m. Il codice deve essere parametrico rispetto a `DIM` e `DIMS`; ovvero cambiando i valori di `DIM` e `DIMS` il codice deve rimanere invariato.
2. (TODO) stampare la posizione i, j dove la sottomatrice viene trovata.

Esercizio 7.6

(TdE November 2012) Si considerino le seguenti dichiarazioni di tipi e variabili che definiscono le strutture dati per rappresentare informazioni relative alle tessere fedeltà dei clienti di una compagnia aerea:

```
#define MAXVIAGGI 100

typedef char Stringa[15];

typedef struct {
  Stringa aeroportoPartenza;
  Stringa aeroportoArrivo;
  float distanza; /* distanza in chilometri (lunghezza del volo) */
} Viaggio;

typedef struct{
  char codiceTessera[10];
  Stringa nome;
  Stringa cognome;
  Stringa nazionalita;
  int numViaggiEffettuati;
```

```
  Viaggio elencoViaggi[MAXVIAGGI];
} Cliente;
```

1. Definire, usando il linguaggio C, un'appropriata variabile per memorizzare le informazioni relative a 50 clienti. Si chiami tale variabile `elencoClienti`;

2. Scrivere in linguaggio C, aggiungendo eventualmente opportune dichiarazioni di variabili, un frammento di codice che permetta di visualizzare a video, per ogni cliente che ha effettuato almeno 10 viaggi, nome, cognome, numero totale di chilometri percorsi e lunghezza media dei voli. Si supponga che l'elenco dei clienti sia memorizzato nella variabile `elencoClienti` definita al punto 1 e che essa sia già stata inizializzata con le informazioni relative a 50 clienti.

Esercizio 7.7

1. Si definiscano le seguenti variabili che specificano le strutture dati per rappresentare i messaggi scambiati attraverso Facebook e ai profili degli utenti:

   - una struttura che definisca un'utente, con nome, cognome e e-mail, tutti composti da caratteri alfanumerici;

   - una struttura che definisca un messaggio, con un contenuto alfanumerico, un mittente scelto tra gli utenti, un numero fissato di destinatari e i destinatari, anch'essi scelti tra gli utenti (fino ad un massimo di 256).

2. Definire, usando il linguaggio C, un'appropriata variabile per memorizzare le informazioni relative a 10 utenti e 10 messaggi. Si chiamino tali variabili `utenti_facebook` e `messaggi_mandati`.

3. Scrivere in linguaggio C, aggiungendo eventualmente opportune dichiarazioni di variabili, un frammento di codice che permetta di visualizzare a video l'indirizzo email di tutti gli utenti che hanno spedito almeno un messaggio e che non siano tra i destinatari di tale messaggio. Si presupponga che la variabile `messaggi_inviati` sia inizializzata correttamente con le informazioni relative a 10 messaggi. Si presupponga inoltre che l'indirizzo email sia identificatore univoco di un utente: pertanto si può utilizzare l'indirizzo email per verificare se due utenti sono lo stesso utente.

Esercizio 7.8

Scrivere un programma che, letta una stringa inserita da tastiera, di lunghezza massima 256. Dopo la lettura il programma deve ordinare i caratteri presenti nella stringa in ordine lessicografico (e.g., `'a' < 'b' < ... < 'z'`) secondo la tabella ASCII.

Suggerimento: pensare al caso limite di una stringa composta da due soli caratteri oltre al terminatore (e.g., `"zc\0"`).

Esercizio 7.9

(TdE Novembre 2010) Si considerino le seguenti dichiarazioni

```
typedef char Stringa[30];

typedef char Matricola[10];

typedef struct {
  Stringa cognome, nome;
  Matricola m;
} DatiStudente;

typedef struct {
  DatiStudente stud;

  /* presenza e voto delle 2 prove intermedie */
  int pres1, pres2; //0 se non presente, !=0 altrimenti
  int voto1, voto2;
} DatiProveStudente;

typedef struct {
  DatiProveStudente s[300];
  int nStud; //numero studenti effettivamente inclusi nel registro
} RegistroProveInt;

registroproveInt r;
```

Durante ogni corso sono previste due prove scritte in itinere non obbligatorie: gli studenti possono partecipare, a loro scelta, a una o a entrambe. Se entrambe le prove sono valide e se la somma dei punteggi è almeno 18, lo studente ha superato l'esame del corso senza dover sostenere altre prove. Ogni prova in itinere assegna al massimo 17 punti, e la prova in itinere è valida solo se il voto è di almeno 8 punti.

1. Assumendo che la variabile `r` sia inizializzata, si scriva un frammento di codice, dichiarando eventuali variabili aggiuntive, che stampi a schermo la matricola e i punti ottenuti dagli studenti che hanno presenziato a una sola delle due prove in itinere, ma non ad entrambe.

2. Con riferimento alle ulteriori dichiarazioni di seguito:

```
typedef struct {
    matricola m[300];
    int punti[300];
    int nStud; //come sopra, studenti effettivamente in elenco
} RegistroEsiti;

RegistroEsiti neg;
```

si scriva una variante del codice precedente che, invece di stampare matricole e punteggi, li inserisca nella variabile `neg` senza lasciare buchi e aggiornando opportunamente il valore di `nStud`.

Esercizio 7.10

(TdE Luglio 2011) Si considerino le seguenti dichiarazioni

```
typedef struct {
    int p1, p2;
} Pari;

Pari p;
```

1. Si scriva un frammento che legga da tastiera un numero intero ed inserisca in `p1` e `p2` i due numeri pari più vicini a quello letto da tastiera e minori di esso. Se ad esempio l'utente inserisce 15, la variabile `p` deve contenere `p.p1 = 14` e `p.p2 = 12;`

2. Data la seguente ulteriore dichiarazione

   ```
   Pari arrayCoppiePari[100];
   ```

   si scriva un nuovo frammento di codice che, letto da tastiera un numero `n > 0`, trovi, a partire da 0, le prime `n` coppie di numeri pari e le memorizzi nell'array. Il frammento di codice deve verificare anche che il valore `n` sia positivo e compatibile con le dimensioni dell'array dichiarato.

Esercizio 7.11

(TdE Settembre 2011) Si considerino le seguenti dichiarazioni:

```
typedef struct {
    float x;
    float y;
} Punto;

typedef struct {
  Punto a;
  Punto b;
} Segmento;

Segmento dati[100];
Segmento s;
Segmento ris[100];
int num_coincidenti;
```

dove il tipo `punto` rappresenta un punto nel piano cartesiano (x, y), il tipo `segmento` rappresenta un segmento con i punti `a` e `b` come estremi, e l'array `dati` contiene le informazioni relative a `100` segmenti nel piano cartesiano.

Si assuma che l'array `dati` sia opportunamente inizializzato. Scrivere un frammento di codice in linguaggio C che:

1. acquisisca da tastiera e memorizzi in `s` le informazioni relative ad un segmento;
2. inserisca nell'array `ris` tutti i segmenti presenti in `dati` che sono coincidenti con quello appena letto e scritto in `s`; si ricorda che due segmenti si dicono coincidenti quando entrambi i loro punti estremi, indipendentemente dall'ordine, hanno le stesse coordinate cartesiane;
3. assegni alla variabile `num_coincidenti` il numero di segmenti coincidenti trovati.
4. (bonus) cercare in `dati` tutte le coppie di segmenti adiacenti che risultano formare delle spezzate e stampare a video i punti in sequenza.

Esercizio 7.12

Scrivere un programma che calcoli elabori una matrice di interi di almeno 3x3 elementi, nel seguente modo:

```
      0 1 2 3 4 5

  0   a b c d e f
  1   g h i j k l
  2   m n o p q r
  3   s t u u v x
```

nell'elemento 0,0 (in questo caso esemplificato con 'a' per brevità) il programma dovrà scrivere la somma degli elementi della sottomatrice 3x3 "centrata" in 0,0. Quando questa sottomatrice non è completamente definita, come nel caso di 0,0, il programma dovra sommare solo gli elementi esistenti.

Ad esempio, al posto di 'a' il programma scriverà $a + b + h + g$;

```
+-----+
|     |
|  a b| c d e f
|  g h| i j k l
+-----+
   m n  o p q r
   s t  u u v x
```

al posto di 'i' il programma scriverà b+c+d+h+i+j+n+o+p.

```
      +-----+
    a |b c d| e f
    g |h i j| k l
    m |n o p| q r
      +-----+
    s  t u u  v x
```

Soluzioni

Soluzione dell'esercizio 7.1

```
/* prima variante */

s = 0;
p = 1;

for(i = 0; i < DIM ; i++) {
    s += m[i][i] * m[i][DIM - i -1];
    p *= m[i][i] + m[i][DIM - i -1];
 }

/* seconda variante con (due indici) */
s = 0;
p = 1;

for(i = 0, j = DIM-1; i < DIM, j > 0; i++, j--) {
    s += m[i][k] * m[i][j];
    p *= m[i][j] + m[i][j];
 }
```

Soluzione dell'esercizio 7.2

```
int p, u; /* indice di piano nelledificio e di ufficio nel piano */

for (p = 0; p < 20; p++)
    for (u = 0; u < 40; u++)
        if ((torre[p][u].esp == sudEst || torre[p][u].esp == sud) &&
            (torre[p][u].superficie >=20 && torre[p][u].superficie<=30)) {
            printf("\n il Signor %s è impiegato di categoria %d",
                   torre[p][u].occupante.cognome,
                   torre[p][u].occupante.cat);
            printf("e ha uno stipendio pari a %d euro \n",
              torre[p][u].occupante.stipendio);
        }
```

```
int uffNord; /* uffNord fa da flag*/

for (p = 0; p < 20; p++) {
    uffNord = 0; //per ogni piano assumo 0
    for (u = 0; u < 40 && !uffNord; u++)
        if(torre[p][u].esposizione == nord)
            uffNord = 1;

    /* se qui vale ancora 0 vuol dire che non ci sono uffici a nord*/
    if (!uffNord)
        printf("il piano %d  non ha edifici esposti a nord", p);
 }

/* è corretto anche senza !uffNord nel for(), anche se non è efficiente. */
```

```
for(i = 0; i < 20; i++) { //scorre i piani
    noUfficiNord = 1; //versione con flag inversa
```

```
    for(j = 0; j < 40; j++) //senza flag arriva fino a 39
        if(torre[i][j].esposizione == nord)
            noUfficiNord = 0;

    if(noUfficiNord) {
        printf("il piano %d non ha uffici a nord", i);
        stipendioMedio = 0;
        cnt = 0;
        for(j = 0; j < 40; j++)
            if(strcmp(torre[i][j].occupante.nome, "Giacomo") == 0){
                stipendioMedio += torre[i][j].occupante.stipendio;
                cnt++;
            }
        printf("Lo stipendio medio dei Giacomo nel "
               "piano è %f ", stipendioMedio / cnt);
    }
}
```

Soluzione dell'esercizio 7.3

1. È vera perché è vero il secondo disgiunto (di cui è vero sia il primo congiunto, in quanto `c1` vale `'e'`, che il secondo congiunto, in quanto `c2` vale `'m'`). L'espressione, essendo vera, ovviamente non è sempre falsa. Inoltre non è sempre vera, per esempio sarebbe falsa se `c2` valesse `'k'` o qualsiasi altro valore diverso da `'m'`.

2. È sempre vera perché, in base alla legge di De Morgan, è equivalente a

   ```
   (c1 <'g') || (c1 > 'g') || (c1 == 'g')
   ```

   quindi la formula non è sempre falsa.

3. È sempre vera perché equivalente alla formula

   ```
   (c1 <= 'm') || (c2 > 'm') || (c2 <= c1)
   ```

   che è identicamente vera: infatti sarebbe falsa solo se fossero falsi tutti e tre i suoi disgiunti, cosa che non può essere, perché se sono falsi i primi due (cioè se `c1 > 'm'` e `c2 <= 'm'`) allora `c2 < c1`, quindi il terzo è vero. Quindi la formula non è sempre falsa.

Soluzione dell'esercizio 7.4

```
#define DIM 10

#include <stdio.h>

int main() {
    int i, j, inf, sup;

    //definizione tipo matrice
    typedef int matrice_t[DIM][DIM];
```

```
    //una matrice
    matrice_t m;

    //inizializzazione della matrice con valori crescenti
    for (i = 0; i < DIM; i++) {

        for (j = 0; j < DIM; j++) {
            m[i][j] = i*j;
            printf("%2d ", m[i][j]);
        }

        printf("\n");
    }

    //stampa a spirale

    //limite inferiore (e superiore)
    for (inf = 0; inf < DIM/2; inf++) {
        sup = DIM - inf - 1;

        printf("\n");

        //da sinistra a destra
        for (j = inf; j < sup; j++)
            printf("%2d ", m[inf][j]);

        printf("\n");

        //dall'alto verso il basso
        for (i = inf; i <= sup; i++)
            printf("%2d ", m[i][sup]);

        printf("\n");

        //da destra a sinistra
        for (j = sup-1; j >= inf; j--)
            printf("%2d ", m[sup][j]);

        printf("\n");

        //dal basso verso l'alto
        for (i = sup-1; i > inf; i--)
            printf("%2d ", m[i][inf]);

        printf("\n");
    }
}
```

Soluzione dell'esercizio 7.5

```
#define DIM 10
#define DIMS 3

#include <stdio.h>

int main() {
    int i, j, si, sj, uguali;

    //una matrice
```

```
    int m[DIM][DIM];

    //una sottomatrice
    int s[DIMS][DIMS];

    //inizializzazione della matrice con valori crescenti
    for (i = 0; i < DIM; i++) {
        for (j = 0; j < DIM; j++) {
            m[i][j] = i*j;
            printf("%2d ", m[i][j]);
        }

        printf("\n");
    }

    //inizializzazione della sottomatrice
    /*
      Test 1: 36 42 48 42 49 56 48 56 64
      Test 2: 1  2  3 2  4  6 3  6  9
      Test 3: 36 42 48 42 49 56 48 56 65
     */
    for (i = 0; i < DIMS; i++) {
        for (j = 0; j < DIMS; j++) {
            scanf("%d", &s[i][j]);
            printf("%2d ", s[i][j]);
        }

        printf("\n");
    }

    //indici per scorrere la sottomatrice
    si = sj = 0;
    uguali = 0;

    for (i = 0; i < DIM-DIMS && !uguali; i++) {
        for (j = 0; j < DIM-DIMS && !uguali; j++) {
            uguali = 1;

            for (si = 0; si < DIMS && uguali; si++)
                for (sj = 0; sj < DIMS && uguali; sj++) {
                    uguali = (m[i+si][j+sj] == s[si][sj]);
                    printf("%d %d %d %d\n", i, j, si, sj);
                }
        }
    }

    //se ha completato i 2 cicli interni e 'uguali == 1'
    if (!uguali)
        printf("NON ");
    printf("trovata");
}
```

Soluzione dell'esercizio 7.6

1. Definizione: `Clienti elencoClienti[50]`
2. Nome, cognome e chilometri totali percorsi e lunghezza media dei voli per i clienti che hanno effettuato almeno 10 viaggi:

```
void main () {
  float somma_distanze;
  int i, j;
  Cliente elencoClienti[50];

  //inizializzazione [...]

  for (i = 0; i < 50; i++) {
    if (elencoClienti[i].numViaggiEffettuati >= 10) {
      somma_distanze = 0.0;
      for (j = 0; j < elencoClienti[i].numViaggiEffettuati; j++)
        somma_distanze += elencoClienti[i].elencoViaggi[j].distanza;

      printf("%s %s %f %f",
             elencoClienti[i].nome,
             elencoClienti[i].cognome,
             somma_distanze,
             somma_distanze/elencoClienti[i].numViaggiEffettuati);
    }
}
```

Soluzione dell'esercizio 7.7

1.
```
typedef struct {
  char nome[24];
  char cognome[24];
  char email[64];
} Utente;

typedef struct {
  char contenuto[256];
  Utente mittente;
  int numDestinatari;
  Utente destinatari[256];
} Messaggio;
```

2.
```
Utente utenti_facebook[10];
Messaggio messaggi_inviati[10];
```

3. Gli utenti che hanno spedito almeno un messaggio sono necessariamente presenti come mittenti nella variabile `messaggi_inviati`:

```
void main () {
  int i, j;
  Utente utenti_facebook[10];
  Messaggio messaggi_inviati[10];

  for (i = 0; i < 10; i++) {
    j = 0;

    //ricerca destinatario.email != mittente.email con strcmp()
    while (j < Messaggi[i].numDestinatari &&
              strcmp(Messaggi[i].mittente.email,
                     Messaggi[i].destinatari[j].email) != 0) {
      j++;
    }
```

```
    if (j == Messaggi[i].numDestinatari) //non trovato
      printf("%s", Messaggi[i].mittente.email);
}
```

Soluzione dell'esercizio 7.8

Per ordinare una stringa di due caratteri si scambiano tali caratteri di posizione solo se questi non sono già ordinati. In una stringa di lunghezza maggiore di due caratteri si procede a scambiare tutti i caratteri adiacenti fino a che non ci sono più scambi da effettuare. Arrivati a tale condizione la stringa è ordinata.

```
#include <stdio.h>

#define LEN 256

int main () {
    char str[LEN+1];
    char tmp;
    int passi = 0;
    int scambi;
    int i;

    printf("Inserire una stringa: ");
    gets(str);

    do {
        scambi = 0; //non ho scambiato
        for (i = 0; str[i+1] != '\0'; i++) {
            if (str[i] > str[i+1]) { //se non ordinati
                //scambio
                tmp = str[i];
                str[i] = str[i+1];
                str[i+1] = tmp;
                scambi = 1; //ho dovuto scambiare
                printf("  %d: %s\n", passi, str);
            }
        }

        printf("%d: %s\n", passi, str);

        passi++;
    } while (scambi); //stop quando scambi non vale 1

    return 0;
}
```

Questo algoritmo di ordinamento, noto anche con il nome di *bubble sort*. Alternativamente si può usare un algoritmo meno efficiente che cerchi iterativamente l'elemento da mettere all'inizio della stringa ordinata:

```
#include <stdio.h>
#include <string.h>

#define LEN 256
```

```
void main () {
char str[LEN+1];
char ordered_str[LEN+1];

char min_char;
int ind_min_char, n;
int i,j;

printf("Inserire una stringa: ");
gets(str);
n = strlen(str);

for (i = 0; i < n; i++) {
    //Find first letter
    min_char = str[0];
    ind_min_char = 0;
    for (j = 1; j < n - i; j++) {
        if (str[j] < min_char){
            min_char = str[j];
            ind_min_char = j;
        }
    }

    //Insert letter in ordered_str
    ordered_str[i] = min_char;

    //Shift letter after the minimum
    for (j = ind_min_char; j < n - i; j++)
        str[j] = str[j+1];


}

ordered_str[n] = '\0';

printf("La parola ordinata e': %s",ordered_str);

}
```

Soluzione dell'esercizio 7.9

1. Si scorre il vettore `s` fino a `nStud` e si stampano i dati solo se `pres1` XOR `pres2` hanno valori diversi da zero.

```
int i;

for (i = 0; i < r.nStud; i++)
  if (!(r.s[i].pres1 && r.s[i].pres2) &&
      (r.s[i].pres1 || r.s[i].pres1)) {
          printf("%s ", r.s[i].stud.m);

          if (r.s[i].pres1)
              printf("%d\n", r.s[i].voto1);
          else
              printf("%d\n", r.s[i].voto2);
      }
```

2. Al posto della stampa effettuo una copia valore per valore e aggiorno il contatore `nStud`:

```
int i;
neg.nStud = 0;

for (i = 0; i < r.nStud; i++)
  if (!(r.s[i].pres1 && r.s[i].pres2) &&    // non entrambe
      (r.s[i].pres1 || r.s[i].pres1)) {     // una delle due
          //neg[neg.nStud].m = r.s[i].stud.m; ERRATO!
          strcpy(neg[neg.nStud].m, r.s[i].stud.m);

          if (r.s[i].pres1)
              neg[nStud].punti = r.s[i].stud.voto1;
          else
              neg[nStud].punti = r.s[i].stud.voto2;

          neg.nStud++;
      }
```

Soluzione dell'esercizio 7.10

1. Letto il numero, se è dispari, si decrementa di 1 e si ottiene un numero pari; altrimenti il numero stesso era già pari (`p1`). L'altro numero (`p2`) si ottiene decrementando `p1` di 2.

```
//...

int n;

scanf("%d", &n);

if (n > 3) {
  if (n % 2 != 0) //dispari
      p.p1 = n - 1; //pari
  else
      p.p1 = n;     //pari

  p.p2 = p.p1 - 2;  //pari
}
```

2. Si procede ciclicamente da 0.

```
#define MAX 100

#include <stdio.h>

int main () {
    typedef struct {
        int p1, p2;
    } pari;

    int i, n;
    pari arrayCoppiePari[MAX];

    do {
        scanf("%d", &n);
```

```
    } while(n <= 0 || n > MAX);

    //primo numero pari == 2
    arrayCoppiePari[0].p1 = 2;

    for (i = 0; i < n; i++) {
        arrayCoppiePari[i+1].p1 = arrayCoppiePari[i].p2 = arrayCoppiePari[i].p1 +
             2;
        printf("<p1: %d, p2: %d>\n", arrayCoppiePari[i].p1, arrayCoppiePari[i].p2
            );
    }

    return 0;
}
```

Soluzione dell'esercizio 7.11

```
#include <stdio.h>

int main () {
    typedef struct {
        float x;
        float y;
    } punto;

    typedef struct {
        punto a;
        punto b;
    } segmento;

    segmento dati[100];
    segmento s;
    segmento ris[100];
    int num_coincidenti;
    int i;

    //[...] inizializzazione della variabile dati

    //1
    printf("Inserire coordinata x del primo punto: ");
    scanf("%f", &s.a.x);

    printf("\nInserire coordinata y del primo punto: ");
    scanf("%f", &s.a.y);

    printf("Inserire coordinata x del secondo punto: ");
    scanf("%f", &s.b.x);

    printf("\nInserire coordinata y del secondo punto: ");
    scanf("%f", &s.b.y);

    //2
    num_coincidenti = 0;
    for (i = 0; i < 100; i++)
        if (dati[i].a.x == s.a.x &&
            dati[i].a.y == s.a.y &&
            dati[i].b.x == s.b.x &&
            dati[i].b.y == s.b.y ||

            dati[i].b.x == s.a.x &&
```

```
            dati[i].b.y == s.a.y &&
            dati[i].a.x == s.b.x &&
            dati[i].a.y == s.b.y) {

            ris[num_coincidenti].a.x = s.a.x;
            ris[num_coincidenti].a.y = s.a.y;
            ris[num_coincidenti].b.x = s.b.x;
            ris[num_coincidenti].b.y = s.b.y;

            num_coincidenti++; //3
        }

    //4
    for (i = 0; i < 100-1; i++)
        if (dati[i].b.x == dati[i+1].a.x &&
            dati[i].b.y == dati[i+1].a.y)
            printf("(%f, %f)--(%f, %f)--(%f, %f)\n",
                   dati[i].a.x, dati[i].a.y,
                   dati[i].b.x, dati[i].b.y,
                   dati[i+1].b.x, dati[i+1].b.y);

    return 0;
}
```

Soluzione dell'esercizio 7.12

```
#define DIM 10

#include <stdio.h>

int main(void)
{
    //definisco il tipo matrix_t, matrice di interi
    typedef int matrix_t[DIM][DIM];

    //definisco due matrici
    matrix_t m,
             n;

    //variabili ausiliarie
    int i,                      //indice delle righe
        j,                      //indice delle colonne
        x,                      //indice delle righe della sottomatrice
        y,                      //indice delle colonne della sottomatrice
        Imin, Imax,             //limiti delle righe della sottomatrice
        Jmin, Jmax;             //limiti della colonne della sottomatrice

    //inizializzo la matrice con valori crescenti
    for (i = 0; i < DIM; i++) {
        for (j = 0; j < DIM; j++) {
            m[i][j] = i * j;

            printf("%2d ", m[i][j]);
        }

        printf("\n");
    }

    for (i = 0; i < DIM; i++) {
```

```
        for (j = 0; j < DIM; j++) {
            /* limiti della sottomatrice:
             *
             * Imin = min(i-1,0),
             * Jmin = min(0,j-1)
             *
             * Imax = min(i+1,DIM)
             * Jmax = min(j+1,DIM)
             */

            //calcolo Imin
            if (i-1 < 0)
                Imin = 0;
            else
                Imin = i-1;

            //calcolo Imax
            if (i+1 > DIM)
                Imax = DIM;
            else
                Imax = i+1;

            //calcolo Jmin
            if (j-1 < 0)
                Jmin = 0;
            else
                Jmin = j-1;

            //calcolo Jmax
            if (j+1 > DIM)
                Jmax = DIM;
            else
                Jmax = j+1;

            //inizializzo a zero
            n[i][j] = 0;

            //scansione della sottomatrice
            for (x = Imin; x < Imax; x++)
                for (y = Jmin; y < Jmax; y++)
                    n[i][j] = n[i][j] + m[x][y];

            printf("%3d ", n[i][j]);
        }

        printf("\n");
    }

    return 0;
}
```

8 Introduzione MATLAB

8.1 Basi

Per pulire il workspace (eliminare tutte le variabili esistenti):

```
clear
```

Per pulire la finestra dei comandi (command window):

```
clc
```

In MATLAB non è necessario dichiarare le variabili ed esse non sono tipizzate

```
a = 15;
b = 5;
```

La stampa di ogni contenuto nella finestra dei comandi è implicita. La presenza del ";" nasconde il risultato dell'istruzione dalla finestra dei comandi

```
c = a + b;
c = a + b
```

Il risultato di una qualunque operazione, se non è assegnato ad altre variabili, viene associato alla variabile di default "ans". Il comando "whos" mostra il contenuto del workspace (i.e., le variabili attualmente dichiarate). Compare anche la loro dimensione, ad esempio 1×1 o 3×4: il primo numero è il numero di righe, il secondo il numero di colonne. Per MATLAB tutto è una matrice (il singolo valore è una matrice 1×1).

```
whos
```

ATTENZIONE Esistono dei nomi riservati, come `i`, che è l'unità immaginaria, quindi evitate di dichiarare una variabile con quello stesso nome. Così facendo, sovrascrivereste la variabile predefinita.

```
ii = 2;
i
pi
```

8.2 Vettori

La dichiarazione di un vettore riga mediante operatore CAT orizzontale [... , ...] (le virgole sono opzionali).

```
riga = [10, 11, 12, 13, 14];
```

Accesso ad un elemento del vettore alla posizione corrispondente ad un indice (intero) `ii` avviene attraverso le parentesi tonde

```
riga(ii)
```

ATTENZIONE Gli indici in MATLAB iniziano da 1, quindi il primo elemento è `riga(1)`. I vettori non hanno dimensioni fissate Per accodare mediante l'operazione CAT orizzontale, un elemento al vettore riga:

```
riga = [riga, 8]
```

In questo caso stiamo sovrascrivendo al vettore `riga` un vettore di dimensione maggiore. In MATLAB possiamo fare assegnamenti di vettori (diversamente dal C).

Se si provasse ad accedere alla posizione 10 di riga

```
riga(10)
```

MATLAB da errore `index out of matrix dimensions` Se si assegna un elemento alla posizione 10 del vettore riga (non allocato precedentemente), il vettore viene allungato e ai valori intermedi viene associato di default 0

```
riga(10) = 8;
```

Per avere un vettore colonna possiamo trasporre il vettore riga oppure nella dichiarazione usare il CAT verticale: [... ; ...]

```
col = riga';
col = [4; 5; 6];
col = [0; col]
whos
```

In questo caso riga sarà di dimensioni $1 \times n$ mentre col $n \times 1$.

La dichiarazione di matrici avviene usando congiuntamente CAT orizzontale e verticale

```
A = [1, 2; 3, 4]
A = [1 2; 3 4]
```

La matrice viene sviluppata in un vettore leggendo le lungo le colonne (ossia viene memorizzata per colonne)

```
aa = A(:)
```

E' possibile accedere agli elementi della matrice specificando dei valori ad entrambi gli indici

```
A(1, 2)
```

Se si fornisce un solo indice si intende la posizione all'interno di `A(:)`

```
A(3)
```

Allo stesso modo se cerchiamo di accedere ad un elemento al di fuori della matrice MATLAB ci comunicherà un errore

```
A(4, 3)
```

in particolare "index exceeds matrix dimensions".

Nel caso non volessimo dichiarare il vettore elemento per elemento possiamo anche inizializzarne uno scegliendo elemento iniziale, elemento finale e passo (step) tra gli elementi

```
inizio = 9;
step = 15;
fine = 223;

v = [inizio : step : fine]
%oppure
v = inizio : step : fine
```

Se non si specifica il passo di default abbiamo passo 1

Possiamo estrarre dei sottovettori specifici andando a selezionare solo alcuni indici:

```
indici = [1, 8, 3];
c = vettore(indici)
```

oppure andando a selezionare una slice (fetta) del vettore iniziale:

```
c = v(1 : 3)
```

La keyword "end", se utilizzata all'interno degli indici di un vettore, assume il valore corrispondente alla lunghezza del vettore

```
d = v(end)
```

E' anche possibile riordinare il vettore v, specificando un opportuno vettore di indici

```
vettoreAlContrario = v([end : -1 : 1]);
```

Posso dichiarare matrici di zeri e uni (tasselli base per costruire le altre matrici):

```
A = zeros(5);
A = ones(5);
```

Posso sostituire valori all'interno di una matrice mediante definizione di sottoindici

```
a = A([1 : end] , 3)
%oppure
a = A(: , 3)
```

per esempio la precedente espressione ssocia il vettore estratto, ossia tutti gli elementi di A con indice della riga da 1 a end e indice di colonna 3 (terza colonna), al vettore `a` Se assegno ad una sottomatrice un valore scalare, esso viene ripetuto per tutta la sottomatrice:

```
A([1 : end], 3) = 2
A(3 : end , 3 : end ) = 1
```

E' possibile sovrascrivere alla terza colonna di A un vettore colonna di 5 elementi (le dimensioni sono consistenti)

```
A(: , 3) = [1 : 5]'
A(: , 3) = [1 : 5]
```

Il secondo domando funziona allo stesso modo perchè MATLAB si occupa automaticamente della trasposizione. Allo stesso modo possiamo copiare parte della matrice in un'altra posizione, sempre controllando che le dimensioni siano consistenti:

```
A([ 1 :  2 ], 1) = A([4 , 5] , 3); %SI
A([ 1 :  2 ], 1) = A([5 , 5] , 3); %SI
A([ 1 :  2 ], 1) = A([5,4] , 3); %SI
A([ 1 :  2 ], 1) = A([3 , 3, 3 ] , 3); %NO
```

Per visualizzare la matrice possiamo usare le seguenti istruzioni:

```
figure()
imagesc(A)
```

La somma tra matrici avviene elemento per elemento. Si possono sommare due matrici solo nel caso in cui siano della stessa dimensione o che una delle due sia uno scalare (che verrà replicato in maniera opportuna da MATLAB)

```
a = [1 2 3]
b = [1 2 4]
c = a + b
d = c + 3
```

Il prodotto elemento per elemento viene eseguito dall'operatore .∗:

```
a .* b
```

mentre l'elevamento a quadrato elemento per elemento è dato dall'operatore .ˆ

```
a .^ 2
```

8.3 Costrutti

Ogni costrutti inizia con una parola chiave e finisce con la parola chiave `end` Il costrutto condizionale `if` ha la seguenta sintassi

```
if %condizione
        %istruzione 1
        %istruzione 2
        %...
elseif
        %istruzione 3
        %istruzione 4
        %...
else
        %istruzione 5
        %istruzione 6
        %...
end
```

dove l'`elseif` ha il significato di un if annidato.

Il while è analogo a quello visto in C:

```
while %condizione
        %istruzione 1
        %istruzione 2
        %...
end
```

In `MATLAB` non esiste il `do ... while`, ma possiamo aggirare il problema.

Il `for` itera su di un vettore:

```
for ii = %vettore
        %istruzione 1
        %istruzione 2
end
```

La sintassi dello `switch` è:

```
switch %variabile
        case %valore1
                %istruzione 1
                %istruzione 2
        case %valore2
```

```
6                   %istruzione 3
7                   %istruzione 4
8           otherwise
9                   %istruzione 5
10                  %istruzione 6
11  end
```

Possiamo fare confronti con valori interi e con stringhe. Le istruzioni all'interno dei vari case sono esclusive.

8.4 Operazioni logiche

Anche in MATLAB abbiamo la possibilità di valutare condizioni logiche. Allo stesso modo di C, lo zero sarà considerato come falso e qualunque altro valore come vero. Le operazioni logiche sono:

```
a = 0;
b = 1;

a && b %and
a & b %and
a || b %or
a | b %or
~a %negazione

a == b %uguale
a ~= b %diverso
a > b %maggiore
a < b %minore
```

8.5 Esercizi

Esercizio 8.1

Scrivere uno script che chieda un anno all'utente. Stampare a video se l'anno è bisestile. Il programma deve continuare a chiedere all'utente anni, finchè gli anni inseriti sono bisestili. Stampare a video il numero totale di anni bisestili inseriti.

Esercizio 8.2

Scrivere uno script che analizzi i voti del primo compitino degli anni precedenti, stampando a schermo:

- media dei voti
- la media dei voti sufficienti
- la varianza dei voti sufficienti
- il numero di promossi al primo compitino

```
% A.A. 2012--2013
voti = [2 8 4 8.1 9.25 11.25 4.75 17 6.25 13 10 2 3.25 3.75...
8.5 16 8 1 2.5 12 10.75 6 12 10 11.75 3.5 10.5 8.5 14.25...
16.5 10.75 8 12 1 10 13 6.75 5.75 9.5 12.75 11 8.5 10.25...
 14.5 4.25 5.5 9.75 16.5 13 15 13 13.75 13.5];

% A.A. 2013--2014
voti = [2 6 11.5 9 11.5 8.5 16 12.75 8.5 8.25 11.5 10 6...
11.25 6 11.25 16.5 9.5 8.5 14.75 10 10.25 16 8.5 14.5 6 8...
8.75 10 16 0 10.25 13 10.25 13.5 13.5 7.5 10 0.5 12.5 9.5...
6.5 15 2 11 10 14.5 10 7.25 12 7.5 4.5 0 4 16.25 14.25 7...
10.5 14 14 9 15.5 2 6 11.5 9 11.5 8.5 16 12.75 8.5 8.25...
11.5 10 6 11.25 6 11.25 16.5 9.5 8.5 14.75 10 10.25 16...
8.5 14.5 6 8 8.75 10];
```

Esercizio 8.3

Scrivere uno script che legga una frase in ingresso e la converta in alfabeto farfallino.

Soluzioni

Soluzione dell'esercizio 8.1

```
clear
clc

bisestile = 1;
counter = 0;

while(bisestile)
    n = input(['inserire anno ']);

    div_4 = (mod(n , 4) == 0);
    div_100 = (mod(n , 100) == 0);
    div_400 = (mod(n , 400) == 0);

    bisestile = ((div_4) && ~(div_100)) || (div_400);

    stringa_output = num2str(n);

    if(bisestile == 0)
        stringa_output = [stringa_output , ' NON e'' '];
    else
        stringa_output = [stringa_output ,' e'' '];
        counter = counter + 1;
    end

    stringa_output = [stringa_output , 'bisestile'];
    disp(stringa_output);
end

disp(['game over hai inserito esattamente ' , num2str(counter)
   , ' bisestili'])
```

Soluzione dell'esercizio 8.2

```
clear
clc

% A.A. 2012--2013
voti = [2 8 4 8.1 9.25 11.25 4.75 17 6.25 13 10 2 3.25 3.75 8.5
    16 8 1 2.5 12 10.75 6 12 10 11.75 3.5 10.5 8.5 14.25 16.5
```

```
   10.75 8 12 1 10 13 6.75 5.75 9.5 12.75 11 8.5 10.25 14.5
   4.25 5.5 9.75 16.5 13 15 13 13.75 13.5];

% A.A. 2013--2014
voti = [2 6 11.5 9 11.5 8.5 16 12.75 8.5 8.25 11.5 10 6 11.25 6
    11.25 16.5 9.5 8.5 14.75 10 10.25 16 8.5 14.5 6 8 8.75 10
   16 0 10.25 13 10.25 13.5 13.5 7.5 10 0.5 12.5 9.5 6.5 15 2
   11 10 14.5 10 7.25 12 7.5 4.5 0 4 16.25 14.25 7 10.5 14 14 9
    15.5 2 6 11.5 9 11.5 8.5 16 12.75 8.5 8.25 11.5 10 6 11.25
   6 11.25 16.5 9.5 8.5 14.75 10 10.25 16 8.5 14.5 6 8 8.75
   10];


% calcolo media: soluzione "alla C"
count = 0;
tot = 0;
for ii = voti
    tot = tot + ii;
    count = count + 1;
end
media = tot/count;

% oppure
% soluzione alla MATLAB
media = mean(voti);

disp(['media: ' , num2str(media)]);

% calcolo media sufficienti: soluzione "alla C"
count = 0;
tot = 0;
for ii = voti(voti >= 8)
    tot = tot + ii;
    count = count + 1;
end
media_suff = tot/count;

% oppure
% soluzione alla MATLAB
media_suff = mean(voti(voti >= 8))

disp(['media dei sufficienti: ' , num2str(media_suff)]);

% calcolo della varianza dei voti sufficienti "alla C"
```

```
count = 0;
tot = 0;
for ii = voti(voti >= 8)
    tot = tot + (ii - media_suff).^2;
    count = count + 1;
end
var_suff = tot/count;

% oppure
% soluzione alla MATLAB
var_suff = var(voti(voti >= 8))

disp(['varianza dei sufficienti: ' , num2str(var_suff)]);

% numero di voti sufficienti: soluzione "alla C"
n_suff = 0;
for ii = (voti >= 18)
    n_suff = n_suff + ii ;
end

% oppure
% numero di voti sufficienti: soluzione "alla MATLAB"
n_suff = sum(voti >= 18);

disp(['numero di voti sufficienti: ' , num2str(n_suff)]);
```

Soluzione dell'esercizio 8.3

```
clear
clc

frase = input(['inserire una stringa '] , 's');

far = [];

for c = frase
    far = [far , c ];
    if (c == 'a' || c == 'e' || c == 'i' || c == 'o' || c == 'u
        ')
        far = [far ,'f' ,c];
    end
end

```

```
15 disp([frase , ' in alfabeto far diventa ' , far])
```

9 MATLAB

9.1 Esercizi

Esercizio 9.1

Scrivere uno script che calcoli la sequenza di Fibonacci di lunghezza 20, e la stampi a schermo. Successivamente si richieda di inserire un numero $2 \leq n \leq 4180$ e valuti se il numero è di Fibonacci. Altrimenti restituisce il numero di Fibonacci più vicino. La successione di Fibonacci è definita così:

$$F(0) = 0$$
$$F(1) = 1$$
$$F(n) = F(n-1) + F(n-2), n > 1$$

Esercizio 9.2

Utilizzando il fatto che il quadrato di n è uguale alla somma dei primi n numeri dispari, calcolare il quadrato di un numero ($n < 100$) inserito dall'utente.

Esercizio 9.3

Chiedere all'utente due parole e stampare a video se una è anagramma dell'altra.

Esercizio 9.4

Creare una matrice `M` di dimensioni 7×5 contenente 0, 1, 2, matrice che rappresenta una situazione in una partita di forza 4 in corso.

Chiedere ai due giocatori, finché uno di questi non inserisce la lettera 'q' (quit), di inserire la colonna (tra 1 e 7) dove intende inserire la propria pedina.

Inserire la pedina nella colonna corretta e visualizzare la matrice `M` così ottenuta.

Bonus: scrivere una porzione dello script che controlli se un giocatore ha vinto, ovvero se ci sono 4 pedine adiacenti dello stesso giocatore in orizzontale, in verticale o **in diagonale**.

Esercizio 9.5

Verificare se una matrice quadrata di dimensione arbitraria è un quadrato magico. Una matrice è un quadrato magico se la somma degli elementi sulle righe, sulle colonne e sulla diagonale principale è la stessa.

Esercizio 9.6

Data una matrice 20×20 che rappresenta le partite di un campionato di calcio (con 0 per vittoria in casa, 1 per pareggio 2 per vittoria in trasferta come risultati possibili). Calcolare la classifica finale ordinata.

Esercizio 9.7

Data una matrice quadrata, leggerla a spirale e metterne il contenuto in un vettore. La lettura a spirale avviene andando a leggere la prima riga, poi l'ultima colonna, quindi l'ultima riga ed infine la prima colonna.

Esercizio 9.8

(TdE 2010 - modificato) Dopo una gara automobilistica si ha come risultato tre tabelle le cui colonne rappresentano gli n partecipanti (numerati da 1 a n) e le righe gli m giri di pista effettuati. Il valore di ogni generica cella (i,j) delle tabelle rappresenta il tempo impiegato (in minuti, secondi e millesimi) dal partecipante j per percorrere il giro i.

Si scrivano le istruzioni per:

- calcolare il tempo totale medio che è stato impiegato dai partecipanti per completare la gara;
- determinare il vincitore della gara (cioè il numero del partecipante il cui tempo di percorrenza totale è minore di quello degli altri partecipanti);

Supponiamo di avere un vettore che ci dica per ogni pilota quanti giri ha effettivamente percorso. Come cambia lo script?

Esercizio 9.9

Scrivere uno script che, ricevendo una matrice M di numeri interi, stampa a video una matrice MR, ottenuta da M nel seguente modo:

- calcola la media aritmetica dei valori di M e ne arrotonda il valore all'intero piú vicino;
- per i valori che in M sono minori della media, in MR si scriva nella posizione corrispondente il valore -1;

- per quelli superiori alla media si pone il valore 1;
- per gli altri (quelli uguali alla media) si pone lo stesso valore in M.

Esercizio 9.10

Si sviluppi uno script che riceve una matrice 8x8, che rappresenta la scacchiera su cui sono disposte le 8 regine, e restituisce se la configurazione delle 8 regine è corretta (nessuna regina mette in scacco un'altra regina), o meno. Si supponga che la matrice contenga il valore 0 in tutte le posizioni libere e il valore 1 nelle posizioni occupate dalle regine.

Soluzioni

Soluzione dell'esercizio 9.1

```
clear
clc
close all

% Inizializza sequenza
fibo = zeros(1,20);

% Calcolo i primi 20 numeri di fibonacci
fibo(1) = 0;
fibo(2) = 1;
for i = 3:20
    fibo(i) = fibo(i-1) + fibo(i-2);
end

fibo

a = input('Inserire un numero (tra 2 e 4180): ');

if sum(a == fibo) > 0
    disp([num2str(a) 'e'' un numero di Fibonacci']);
else
    % Cerco i numeri di fibonacci più piccoli di a e scelgo l'
       ultimo
    inferiori = fibo(fibo < a);
    inferiori = inferiori(end);

    % Cerco i numeri di Fibonacci più grandi di a e scelgo il
       primo
    superiori = fibo(fibo > a);
    superiori = superiori(1);

    % Cerco il più vicino tra il più grande numero di fibonacci
        più piccolo
    % di a e il più piccolo numero di Fibonacci più grandi di a
    if superiori - a < a - inferiori
        vicino = superiori;
    else
        vicino = inferiori;
    end
```

```
37
38      disp(['Il numero di Fibonacci piu'' vicino a ' num2str(a) '
            e'' ' num2str(vicino)]);
39
40  end
```

Soluzione dell'esercizio 9.2

```
1   clc
2   clear
3
4   N = input('Inserire il numero da elevare al quadrato (n < 100):
        ');
5
6   numeri = 2 * [0 : N - 1] + 1;
7
8   % Soluzione alla C 1
9   c = 1;
10  somma_while = 0;
11  while c <= N
12      somma_while = somma_while + numeri(c);
13      c = c + 1;
14  end
15
16  % Soluzione ibrida
17  somma_ibrida = 0;
18  for c = numeri
19      somma_ibrida = somma_ibrida + c;
20  end
21
22  % Soluzione alla Matlab
23  somma_matlab = sum(numeri);
24
25  fprintf('%d^2 = %d\n', N, somma_while);
26  fprintf('%d^2 = %d\n', N, somma_ibrida);
27  fprintf('%d^2 = %d\n', N, somma_matlab);
```

Soluzione dell'esercizio 9.3

```
1   clear
2   clc
3
4   parola1 = input('Inserire la prima parola: ','s');
```

```
parola2 = input('Inserire la seconda parola: ','s');

isto1 = zeros(1,255);
isto2 = zeros(1,255);

for ii = parola1
    isto1(ii) = isto1(ii) + 1;
end

for ii = parola2
    isto2(ii) = isto2(ii) + 1;
end

fprintf('Le due parole ');
if any(isto1 ~= isto2)
    fprintf('non ');
end
fprintf('sono una l''anagramma dell''altra\n');
```

Soluzione dell'esercizio 9.4

```
clear
clc
close all;

M = [ 0 0 0 0 0 0 0 ;...
      0 0 0 0 0 0 0 ;...
      0 0 0 0 0 0 0 ;...
      0 0 0 0 0 0 0 ;...
      0 0 0 0 0 0 0 ];

turno_giocatore = 1;
a = 6;
while (a ~= 'q')
    disp(['E'' il turno del giocatore ' num2str(turno_giocatore
       )]);
    a = input('Inserire una giocata (numero di colonna 1-7) o
       uscire (q): ');

    if a~='q'
        if (M(1,a) ~= 0)
            disp('Giocata illegale');
        else
```

```
            indici = M(:,a) == 0;
            pos_libera = sum(indici);
            M(pos_libera,a) = turno_giocatore;
            imagesc(M);
            if turno_giocatore == 1
                turno_giocatore = 2;
            else
                turno_giocatore = 1;
            end
        end
    end
end
```

Soluzione dell'esercizio 9.5

```
clear
clc
close all

M = magic(4);
%M = randi(4,3);
[r, c] = size(M);

% Controllo matrice quadrata
assert(r == c);

% Calcolo somme su righe
somme = zeros(1,2 * r + 1);
for ii = 1:r
    somme(ii) = sum(M(ii,:));
end

% Calcolo somme su colonne
for ii = (r+1):2*r
    somme(ii) = sum(M(:,ii-r));
end

% Calcolo somma su diagonale
somme(2*r+1) = sum(diag(M));

somme

if sum(somme == somme(1)) == 2*r+1
```

```
29      disp('La matrice e'' un quadrato magico');
30  else
31      disp('La matrice non e'' un quadrato magico');
32  end
```

Soluzione dell'esercizio 9.6

```
1   clear
2   clc
3
4   squadre = { 'Atalanta' 'Bologna' 'Carpi' 'Chievo' 'Empoli' '
        Fiorentina' 'Frosinone' ...
5       'Genoa' 'Inter' 'Juventus' 'Lazio'  'Milan' 'Napoli' '
            Palermo' 'Roma'...
6       'Sampdoria' 'Sassuolo' 'Torino' 'Udinese' 'Verona'};
7   squadre_alt = squadre;
8
9   risultati = randi(3,20)-1;
10  for ii = 1:20
11      risultati(ii,ii) = -1;
12  end
13
14  % Versione alla C
15  punti = zeros(20,1);
16  for ii = 1:20
17      punti(ii) = sum(risultati(ii,:) == 0) * 3 + sum(risultati(
            ii,:) == 1) + ...
18          sum(risultati(:,ii) == 2) * 3 + sum(risultati(:,ii) ==
                1);
19  end
20
21  %Versione alla Matlab
22  punti_alt = sum(risultati == 0,2) * 3 + sum(risultati == 1,2) +
        ...
23          sum(risultati' == 2,2) * 3 + sum(risultati' == 1,2);
24
25  assert(sum(punti == punti_alt) == 20)
26
27  %Ordiniamo le squadre
28  while (~isempty(punti))
29      maxi = max(punti);
30      trovato = 0;
31      contatore = 1;
```

```
32      while trovato == 0
33          if punti(contatore) == maxi
34              disp(['La squadra ' squadre{contatore} ' ha
                    totalizzato ' num2str(punti(contatore)) ' punti.
                    ']);
35              punti(contatore) = [];
36              squadre(contatore) = [];
37              trovato = 1;
38          else
39              contatore = contatore + 1;
40          end
41      end
42  end
43  disp('-----------------------------------');
44
45  %Oppure chiediamo a MATLAB
46  [punti_alt, indici] = sort(punti_alt,'descend');
47  squadre_alt = squadre_alt(indici);
48  for ii = 1:20
49      disp(['La squadra ' squadre_alt{ii} ' ha totalizzato '
           num2str(punti_alt(ii)) ' punti.']);
50  end
```

Soluzione dell'esercizio 9.7

```
1   clear
2   clc
3
4   M = randi(10,7);
5   M_old = M;
6
7   vec = [];
8
9   while(~isempty(M))
10
11      vec = [vec M(1,:)];
12      M(1,:) = [];
13      if (~isempty(M))
14          vec = [vec M(:,end)'];
15          M(:,end) = [];
16      end
17      if (~isempty(M))
18          vec = [vec M(end,end:-1:1)];
```

```
19          M(end,:) = [];
20      end
21      if (~isempty(M))
22          vec = [vec M(end:-1:1,1)'];
23          M(:,1) = [];
24      end
25  end
26
27  assert(sum(vec) == sum(M_old(:)));
28
29  M_old
30  vec
```

Soluzione dell'esercizio 9.8

```
1   clear
2   clc
3
4   n_piloti = 10;
5   n_giri = 30;
6
7   minuti = randi(2,n_piloti,n_giri);
8   secondi = 60 * rand(n_piloti,n_giri);
9   millesimi = 1000 * rand(n_piloti,n_giri);
10
11  tempo_medio = mean(minuti * 60 + secondi + millesimi / 1000, 2)
        ;
12  tempo_vinc = min(tempo_medio);
13
14  vinc = find(tempo_medio == tempo_vinc);
15
16  disp(['Il vincitore e '' ' num2str(vinc)]);
```

Soluzione dell'esercizio 9.9

```
1   clear
2   clc
3
4   M = randi(20,5);
5
6
7   MR = zeros(size(M));
8
```

```
media = round(mean(M(:)));

MR(M < media) = -1;
MR(M > media) = 1;
MR(M == media) = media;
```

Soluzione dell'esercizio 9.10

```
clear
clc
close all

scacchiera = randi(2,8) - 1;

%Controllo righe
righe_ok = all(sum(scacchiera,2) <= 1);

%Controllo colonne
colonne_ok = all(sum(scacchiera) <= 1);

diag_ok = 1;
anti_diag_ok = 1;

%Controllo diagonali principali (alla C)
if righe_ok && colonne_ok && diag_ok
    for ii = 1:7
        somma = 0;
        count_col = 1;
        count_row = ii;
        while (count_row <= 8)
            somma = somma + scacchiera(count_row,count_col);
            count_col = count_col + 1;
            count_row = count_row + 1;
        end
        if somma > 1
            diag_ok = 0;
        end
    end
end

if righe_ok && colonne_ok && diag_ok
    for ii = 2:7
        somma = 0;
```

```
36          count_col = ii;
37          count_row = 1;
38          while (count_col <= 8)
39              somma = somma + scacchiera(count_row,count_col);
40              count_col = count_col + 1;
41              count_row = count_row + 1;
42          end
43          if somma > 1
44              diag_ok = 0;
45          end
46      end
47  end
48
49  %Controllo diagonali principali (alla Matlab)
50  if righe_ok && colonne_ok
51      sum_diag = zeros(13,1);
52      for ii = -6:6
53          sum_diag(ii+7) = sum(diag(scacchiera,ii));
54      end
55      diag_ok = all(sum_diag <= 1);
56  end
57
58  %Controllo antidiagonali
59  if righe_ok && colonne_ok && diag_ok
60      antiscacchiera = flip(scacchiera);
61      sum_anti_diag = zeros(13,1);
62      for ii = -6:6
63          sum_anti_diag(ii+7) = sum(diag(antiscacchiera,ii));
64      end
65      anti_diag_ok = all(sum_anti_diag <= 1);
66  end
67
68  if righe_ok && colonne_ok && diag_ok && anti_diag_ok
69      disp('Le regine sono ben disposte');
70  else
71      disp('Almeno una coppia di regine e'' mal disposta');
72  end
```

10 MATLAB

10.1 Plot

Il comando `plot` viene utilizzato per la visualizzazione di grafici. Con

```
plot(x,y)
```

viene disegnato un grafico che passa per i punti le cui ascisse e ordinate sono contenute rispettivamente nei vettori `x` e `y`.

Per disegnare il grafico della parabola $y = x^2$ nell'intervallo $[-1, 1]$:

```
x = linspace(-1,1,100);
y = x.^2;
plot(x,y)
```

Si possono specificare diversi stili per il plot

```
plot(x,y) % linea continua blu
plot(x,y,'r') % linea continua rossa
plot(x,y,'b--') % linea tratteggiata blu
plot(x,y,'g-.') % linea-punto verde
plot(x,y,'c.') % punti azzurri
```

Per aprire una nuova figura:

```
figure
```

Ogni nuovo plot sovrascrive il precedente. Per disegnare più di un plot sulla stessa figura:

```
x = linspace(-1,1,100);
figure
hold on
plot(x,x)
plot(x,x.^2,'r')
hold off
```

Per aggiungere un titolo alla figura

```
title('Due plot')
```

È possibile utilizzare dei marker oltre alle linee nei plot

```
plot(x,x.^2,'-o')
plot(x,x,'rx')
```

Si possono disegnare anche punti singoli

```
x = 1;
y = 2;
plot(x,y,'o')
```

Per una lista completa degli stili di linea, dei colori e dei marker che si possono usare

```
help plot
doc plot
```

Si possono disegnare anche semplici forme geometriche, come ad esempio dei rettangoli

```
x = 0;
y = 1   ;
base = 3;
altezza = 5;
rectangle('Position',[x, y, base, altezza],'FaceColor','b')
```

Per ridefinire i limiti del plot

```
xlim([-1 5])
ylim([0 10])
```

10.2 Esercizi

Esercizio 10.1

(TdE 2010 - modificato) Dopo una gara automobilistica si ha come risultato tre matrici le cui colonne rappresentano gli n partecipanti (numerati da 1 a n) e le righe gli m giri di pista effettuati. Il valore di ogni generica cella (i,j) delle tabelle rappresenta il tempo impiegato (in minuti, secondi e millesimi) dal partecipante j per percorrere il giro i.

Si scrivano le istruzioni per:

- calcolare il tempo totale medio che è stato impiegato dai partecipanti per completare la gara;
- determinare il vincitore della gara (cioè il numero del partecipante il cui tempo di percorrenza totale è minore di quello degli altri partecipanti);
- disegnare un grafico in cui l'asse delle x rappresenta i partecipanti. Tracciare quindi
  - una linea continua nera indicante i tempi medi (in secondi) sul giro per ciascun pertecipante;
  - una linea tratteggiata rossa indicante i tempi minimi (in secondi) sul giro per ciascun partecipante;
  - una linea tratteggiata verde indicante i tempi massimi (in secondi) sul giro per ciascun partecipante;
  - un asterisco blu in corrispondenza del tempo medio sul giro ottenuto dal vincitore;

Esercizio 10.2

Si costruisca una matrice in MATLAB per la memorizzazione delle precipitazioni atmosferiche registrate da una stazione meteorologica. La matrice deve avere 4 righe e n colonne, dove n indica il numero dei giorni monitorati (tutti i giorni degli anni dal 2012 al 2014). Ogni colonna fa riferimento ad un determinato giorno: i primi tre elementi di ciascuna colonna indicano rispettivamente il giorno, il mese e l'anno, mentre l'ultimo elemento rappresenta il valore in mm della quantità di pioggia caduta.

Si tracci un grafico in cui l'asse delle ascisse rappresenta i mesi del 2014. In corrispondenza di ogni mese tracciare un segmento verticale le cui coordinate sono espresse dai valori massimo e minimo delle precipitazioni in quel mese. Si mostri inoltre con un asterisco il valore medio delle precipitazioni di ciascun mese.

Esercizio 10.3

Si prepari un programma MATLAB in grado di disegnare la configurazione iniziale di una partita di battaglia navale. Si assuma che il piano di battaglia sia una griglia 10×10 e che la disposizione delle navi venga fornita attraverso 4 vettori contententi rispettivamente: l'ascissa e l'ordinata della prua della nave, e l'ascissa e l'ordinata della poppa.

Il piano di battaglia deve essere disegnato mediante la costruzione di una griglia, mentre le navi verranno rappresentate come rettangoli rossi. Le ascisse delle coordinate verrano indicate con le lettere maiuscole dalla A alla J, mentre le ordinate con numeri da 1 a 10.

Soluzioni

Soluzione dell'esercizio 10.1

```
clear
clc

n_piloti = 10;
n_giri = 30;

minuti = randi(2,n_piloti,n_giri);
secondi = randi(60,n_piloti,n_giri) - 1;
millesimi = randi(1000,n_piloti,n_giri) - 1;

tempo_totale = minuti * 60 + secondi + millesimi / 1000;
tempo_medio = mean(tempo_totale, 2);

[tempo_vinc, vinc] = min(tempo_medio);
disp(['Il vincitore e '' ' num2str(vinc)]);

tempo_minimo = min(tempo_totale,[],2);
tempo_massimo = max(tempo_totale,[],2);

figure
hold on
title('Tempi ottenuti da ciascun partecipante')
piloti = 1:n_piloti;
piloti = piloti';
plot(piloti,tempo_medio,'k');
plot(piloti,tempo_minimo,'r--');
plot(piloti,tempo_massimo,'g--');
plot(vinc,tempo_medio(vinc),'b*');
```

Soluzione dell'esercizio 10.2

```
clear
clc

n_giorni = 366+365*2;
prec = zeros(4,n_giorni);

c = 0;
```

```
for anno=2012:2014
    for mese=1:12
        switch mese
            case {1,3,5,7,8,10,12}
                giorni_mese = 31;
            case 2
                if(mod(anno,4) == 0)
                    giorni_mese = 29;
            else
                giorni_mese = 28;
                end
            otherwise
                giorni_mese = 30;
        end
        for giorno=1:giorni_mese
            c = c + 1;
            prec(1,c) = giorno;
            prec(2,c) = mese;
            prec(3,c) = anno;
            prec(4,c) = randi(101)-1;
        end
    end
end

anno = 2014;

prec_min = zeros(1,12);
prec_max = zeros(1,12);
prec_medie = zeros(1,12);
for mese=1:12
    colonne = (prec(3,:) == 2014) & (prec(2,:) == mese);
    prec_min(mese) = min(prec(4,colonne));
    prec_max(mese) = max(prec(4,colonne));
    prec_medie(mese) = mean(prec(4,colonne));
end

figure
hold on
for mese=1:12
    plot([mese mese],[prec_min(mese) prec_max(mese)],'b');
    plot(mese,prec_medie(mese),'r*');
end
```

Soluzione dell'esercizio 10.3

```
clear
clc

figure
hold on

for x=0:10
    plot([x x],[0 10],'k');
end
for y=0:10
    plot([0 10], [y y],'k');
end

ascisse_prua   = ['A', 'G', 'E'];
ordinate_prua  = [ 3 ,  8 ,  2 ];
ascisse_poppa  = ['A', 'I', 'E'];
ordinate_poppa = [ 4 ,  8 ,  6 ];


n_navi = length(ascisse_prua);

for nave=1:n_navi
    x_prua = ascisse_prua(nave) - 'A';
    y_prua = ordinate_prua(nave) - 1;
    x_poppa = ascisse_poppa(nave) - 'A';
    y_poppa = ordinate_poppa(nave) - 1;

    base = abs(x_prua-x_poppa) + 0.8;
    altezza = abs(y_prua-y_poppa) +0.8;
    x_start = min(x_prua,x_poppa) + 0.1;
    y_start = min(y_prua,y_poppa) + 0.1;
    rectangle('Position',[x_start y_start base altezza],'
       FaceColor','r');
end

title('Battaglia navale')

set(gca,'XTick',[0.5:1:9.5])
set(gca,'XTickLabel',['A' 'B' 'C' 'D' 'E' 'F' 'G' 'H' 'I' 'J'
   ]')

set(gca,'YTick',[0.5:1:9.5])
```

```
41 set(gca,'YTickLabel',[1:1:10]')
```

11 Funzioni MATLAB

Le funzioni in un programma sono utilizzate per strutturare il codice in sottoparti e per evitare di replicare inutilmente il codice. In MATLAB le funzioni vengono identificate con la parola chiave `function` ed è buona regola che abbiano lo stesso nome dello script che le contiene. In generale sono strutturate nel seguente modo:

```
function [output1, output2, ..] = nome(input1, input2, ..)
%istruzioni
output1 = ..
output2 = ..
...
```

dove

- `output1, output2, ..` sono gli output (opzionali), che, se dichiarati, devono essere inizializzati dalla funzione
- `input1, input2, ..` sono gli input (opzionali) che servono per il calcolo degli output

Prima di iniziare a scrivere il corpo di una funzione, la prima cosa da stabilire è quali siano input (argomenti) e output (valori restituiti) necessari.

MATLAB vede solo le funzioni delle proprie librerie (come `mean(),min()`) o le funzioni dichiarate nella stessa cartella di dove viene eseguito lo script in cui si richiamano le funzioni.

Un elenco (non esaustivo) delle funzioni di MATLAB è il seguente:

```
zeros(m,n) %crea una matrice di zeri di dimensioni m*n
ones(m,n) %crea una matrice di uni di dimensioni m*n
eye(n) %crea una matrice identita' di ordine n
rand(m,n) %crea una matrice di numeri casuali in [0,1] di
   dimensioni m*n

length(v) %restituisce la lunghezza del vettore v
size(M) %restituisce le dimensioni della matrice M

ceil(x) % arrotonda x all'intero superiore
```

```
10 floor(x) %arrotonda x all'intero inferiore
11 fix(x) %arrotonda x all'intero piu' vicino a 0
12
13 max(v) %restituisce il massimo del vettore v
14 min(v) %restituisce il minimo del vettore v
15 mean(v) %restituisce la media del vettore v
16 mod(m,n) % restituisce il modulo n di m
17
18 find(p) %restituisce gli indici degli elementi che soddisfano p
```

11.1 Esercizi

Esercizio 11.1

Scrivere uno script che dato un numero intero positivo n minore di 6765, stabilisca se è di Fibonacci e nel caso non lo sia restituisca il più grande numero di Fibonacci minore di n e il più piccolo numero di Fibonacci maggiore di n.

Strutturare lo script in modo da utilizzare una funzione che dato un numero $N = 20$ restituisca i primi N numero di Fibonacci.

Esercizio 11.2

Scrivere una funzione `closestVal` che prende in ingresso:

- un vettore `v`
- un valore `n`

e restituisce un valore di `v` pivìcino a `n`, ossia a distanza minima.

Ad esempio:

```
1 v = [1 4 40];
2 n = 20;
3 closestVal(v, n)
```

restituisce 4, mentre

```
1 v = [1 4 40];
2 n = 32;
3 closestVal(v, n)
```

restituisce 40.

Commentare il caso in cui si debbano restituire tutti i valori a distanz minim (nel caso ce ne sia più di uno).

Esercizio 11.3

Scrivere uno script che chiede all'utente di inserire un numero positivo a (nel caso in cui il numero non è positivo ripetere l'inserimento) e verifichi se il numero è perfetto, in caso contrario dice se è abbondante o difettivo.

Un numero è perfetto se corrisponde alla somma dei suoi divisori, escluso se stesso (6 è perfetto $1 + 2 + 3 = 6$), è abbondante se è minore della somma dei suoi divisori (20 visto che $1 + 2 + 4 + 5 + 10 > 20$) e altrimenti è difettivo (19 e tutti i numeri primi sono difettivi).

Richiede un altro numero b e controlla se i due numeri sono amici due numeri a,b

Due numeri a e b sono amici se la somma dei divisori di a è uguale a b e viceversa (es 284 e 220 sono amici).

Strutturare lo script con delle funzioni:

- `inserisciPositivo` che legge un numero intero positivo;
- `sommaDivisori` che dato un numero restituisce la somma dei suoi divisori;
- `controllaSePerfetto` che restituisce se un numero è perfetto, se è abbondante o difettivo:
- `controllaSeAmici` che restituisce se due numeri sono amici.

Esercizio 11.4

Quest'anno Babbo Natale ha deciso di farsi aiutare per la consegna dei regali da KwanzaBot e Superman.[1] Babbo natale ha un elenco di bambini (`elenco` matrice di caratteri) e deve consegnare un regalo particolare ad ognuno di essi tra piccolo (1), medio (2) e grande (3) a seconda di quanto sono stati buoni (`buono` vettore di interi). Per assegnare il regalo ci sono delle soglie di punteggi: fino a 700 punti il bambino ha un regalo piccolo, fino a 900 medio, altrimenti grande. Ad ogni consegna si deve stampare a video il nome del bambino, il suo regalo e chi l'ha portato.

Per dividersi il lavoro i tre hanno deciso che Babbo Natale avrebbe consegnato i regali partendo dal primo dell'elenco, KwanzaBot dal fondo e Superman scegliendo uno dei bambini a caso (`randi()`). I bambini che hanno il regalo grande lasciano anche dei biscotti a chi consegna il regalo, quindi ringrazia riportando il suo apprezzamento per i biscotti (stampandolo a video).

Strutturare lo script che consegni i regali a tutti i bambini in funzioni che dividano in maniera logica l'azione di selezione del regalo (`selezionaRegalo`), la consegna del regalo (`consegnaRegalo`) e l'eliminazione dall'elenco di chi ha già ricevuto il regalo (`cancellaBambino`).

[1] `http://futurama.wikia.com/wiki/Kwanzaabot`

Soluzioni

Soluzione dell'esercizio 11.1

```
clear
clc
close all

N = 20;
F = fibonacci(N);

n = input(' Inserire un numero ');

if(any(F == n))
    disp([num2str(n), ' e'' di Fibonacci'])
else
    indx_M = find(F > n);
    M = F(indx_M(1));

    indx_m = find(F < n);
    m = F(indx_m(end));

    str = [num2str(n), ' e'' compreso tra '];
    str = [str, num2str(m), ' e ', num2str(M)];
    disp(str);
end
```

```
function F = fibonacci(n)
% FIBONACCI calcola i primi n numeri di fibonacci

F = zeros(n,1);

if(n >= 1)
    F(1) = 1;
end
if (n >= 2)
    F(2) = 1;
end

for ii = 3 : n
    F(ii) = F(ii-1) + F(ii-2);
end
```

Soluzione dell'esercizio 11.2

```
clc
clear
close all

%% Sezione 1
v = [1 4 40];
n = 20;
closestVal(v, n)

n = 32;
closestVal(v, n)

%% Sezione 2
vett = [10 3 -5 7 5];
valore = 4;

res = closestVal(vett, valore);
```

```
function [val] = closestVal(v, n)
% CLOSESTVAL prende in ingresso un vettore  v ed un valore n e
% restituisce il valore di v più vicino a n

dist = abs(v - n);
[~, pos] = min(dist);
val = v(pos);
```

```
function [val] = closestVal2(v, n)
% CLOSESTVAL prende in ingresso un vettore  v ed un valore n e
% restituisce i valori di v più vicini a n

dist = abs(v - n);
pos = dist == min(dist);
val = v(pos);
```

Soluzione dell'esercizio 11.3

```
clc
clear

% richiedere numero
n = inserisciPositivo();
```

```

[perf, abb, dif] = controllaSePerfetto(n);

if(perf == 1)
    disp([num2str(n), ' e'' perfetto']);
else
    disp([num2str(n), ' NON e'' perfetto']);
    if(abb == 1)
        disp([num2str(n), ' e'' abbondante']);
    else
        disp([num2str(n), ' e'' difettivo']);
    end
end

m = inserisciPositivo();

amici = controllaSeAmici(n,m);

if(amici)
    disp([num2str(n), ' e ', num2str(m), ' sono amici'])
else
    disp([num2str(n), ' e ', num2str(m), ' NON sono amici'])
end
```

```
function n = inserisciPositivo()

n = -1;
while(n < 0)
    n = input('Inserire un numero positivo ');
end
```

```
function s = sommaDivisori(n)

D = 1 : 1 : n / 2;
A = mod(n, D);
s = sum(D(A == 0));
```

```
function s = sommaDivisoriCLike(n)

v = [];
for ii = 1 : n /2
    if(mod(n, ii) == 0)
        v = [v, ii];
```

```
    end
end
s = sum(v);
```

```
function [perf, abb, dif] = controllaSePerfetto(n)

if(n == sommaDivisori(n))
    perf = 1;
    abb = 0;
    dif = 0;
elseif(n < sommaDivisori(n))
    perf = 0;
    abb = 1;
    dif = 0;
else
    perf = 0;
    abb = 0;
    dif = 1;
end
```

```
function res = controllaSeAmici(n, m)

if (sommaDivisori(n) == m && sommaDivisori(m) == n)
    res = 1;
else
    res = 0;
end
```

Soluzione dell'esercizio 11.4

```
clear
clc
close all

%%
load('dati_babbo_natale');
n_rimasti = size(elenco,1);

while n_rimasti > 0

    %BN
    regaloBN = selezionaRegalo(buono,1);
```

```
13      elenco = consegnaRegalo(elenco, 1, regaloBN, 'Babbo Natale'
            );
14      [elenco, buono] = cancellaBambino(elenco,buono,1);
15      n_rimasti = n_rimasti - 1;
16
17      %QB
18      if n_rimasti > 0
19          regaloQB = selezionaRegalo(buono,n_rimasti);
20          consegnaRegalo(elenco, n_rimasti, regaloQB, 'QwanzaBot'
                );
21          [elenco, buono] = cancellaBambino(elenco, buono,
                n_rimasti);
22          n_rimasti = n_rimasti - 1;
23      end
24
25      %SM
26      if n_rimasti > 0
27          ind = randi(n_rimasti);
28          regaloSM = selezionaRegalo(buono,ind);
29          consegnaRegalo(elenco, ind, regaloSM, 'Superman');
30          [elenco, buono] = cancellaBambino(elenco, buono, ind);
31          n_rimasti = n_rimasti - 1;
32      end
33      pause();
34  end
```

```
1   function regalo = selezionaRegalo(buono, ind)
2
3
4   if buono(ind) < 700
5       regalo = 1; %regalo piccolo
6   elseif buono(ind) < 900
7       regalo = 2; %regalo medio
8   else
9       regalo = 3; %regalo grande
10  end
```

```
1   function consegnaRegalo(elenco, idx, regalo, identita)
2
3   fprintf(['Quest''anno ' elenco(idx,:) ' ha ricevuto un regalo
        di tipo ']);
4   if regalo == 1
5       fprintf('piccolo');
6   elseif regalo == 2
```

```
7       fprintf('medio');
8   else
9       fprintf('grande\n');
10      fprintf([identita ' ha gradito i biscotti']);
11  end
12  fprintf('\n');
```

```
1   function [elenco, buono] = cancellaBambino(elenco, buono, ind)
2
3   buono(ind) = [];
4   elenco(ind,:) = [];
```

12 Strutture con MATLAB

Per inizializzare le strutture si può utillizare il costrutto:

```
S = struct('campo1',val1,'campo2',val2, ...);
```

oppure inizializzarne direttamente i campi con una serie di istruzioni:

```
S.campo1 = val1;
S.campo2 = val2;
...
```

12.1 Esercizi

Esercizio 12.1

Scrivere un programma per la gestione di un magazzino dove ogni prodotto nel magazzino è univocamente identificato da un codice a barre (un numero intero).

Il software di gestione associa ad ogni prodotto un carattere che indica la tipologia del prodotto e due numeri, il primo che indica il numero di pezzi in stock il secondo che indica il numero di pezzi ordinati.

Si ipotizzi che codice a barre, tipo, stock, ed ordine siano 4 vettori, già popolati, contenenti tutte le informazioni necessarie per la gestione del magazzino. (l'i-esimo elemento di stock e di ordine rappresentano le quantità relative al prodotto a cui è associato l'i-esimo elemento del vettore dei codici a barre).

Ad esempio un magazzino popolato sarà:

```
barcodes = [123 ; 1312 ; 12312 ; 1231 ; 99123];
tipo = ['A' ; 'A' ; 'X' ; 'W' ; 'W' ];
stock = [0 ; 300 ; 5 ; 6 ; 0 ];
ordine = [23 ; 100 ; 2 ; 100 ; 0 ];
```

Si strutturi la struttura magazzino e si scriva:

- la funzione `ricerca` che prende in ingresso un codice a barre ed il magazzino e restituisce un messaggio contenente il tipo di prodotto, il numero di pezzi in stock ed in ordine;
- un esempio di chiamata alla funzione `ricerca`;
- la funzione `ricercaMancanti` che, dato un parametro P ed il magazzino, restituisce al programma chiamante un vettore contenente i codici a barre dei prodotti:
  - se $P = 0$, non presenti in stock ma in ordine;
  - se $P = 1$, non presenti in stock che non sono nemmeno in ordine;
  - se $P = 2$, per cui ci sono più pezzi in ordine che attualmente in stock;
- Scrivere un esempio di chiamata alla funzione `ricercaMancanti`;
- Si scriva la funzione `aggiungiProdotto`, che permette di aggiungere al magazzino un nuovo prodotto (barcode, stock ed ordine);
- Scrivere un esempio di chiamata alla funzione `aggiungiProdotto`.

Esercizio 12.2

Scrivere un programma per simulare il gioco della roulette.

La roulette possiede 38 numeri (da 1 a 36, lo zero e il doppiozero). 0 (zero) e 00 (doppiozero) non sono ne pari ne dispari (vince il banco). Inizialmente, banco e giocatori possiedono 5000 euro ciascuno.

Implementare la simulazione di una serie di giocate di due giocatori Pippo e Pluto, che giocano seguendo le seguenti strategie:

- ad ogni giocata il giocatore Pippo punta 5 euro su pari o dispari con stessa probabilità. Se vince ottiene 2 volte la posta, se perde il banco incassa il valore giocato;
- ad ogni giocata il giocatore Pluto punta 1 euro sul 15 (se esce 15 vince 36 volte la posta).

Il gioco termina quando o il banco viene sbancato (arriva a 0 euro) o entrambi i giocatori non hanno più soldi per fare la propria puntata.

Si tenga traccia delle somme a disposizione di ogni giocatore e del banco ad ogni giocata dall'inizio del gioco fino alla sua fine. Grazie a queste informazioni, disegnare l'evoluzione della disponibilità monetaria dei due giocatori e del banco. Si disegnino i valori con delle linee di spessore 2, in rosso per Pippo, in blu per Pluto e in nero per il banco. Si disegni la legenda, il titolo e si forniscano le etichette per gli assi x e y.

Esercizio 12.3

Scrivere in `MATLAB` una funzione per analizzare i codici IBAN dei conti correnti. Un codice IBAN è una sequenza di 27 caratteri alfanumerici così composta:

- 2 caratteri maiuscoli (sigla della nazione)
- 2 cifre (CIN Europeo)
- 1 carattere maiuscolo (CIN italiano)
- 5 cifre (ABI)
- 5 cifre (CAB)
- 12 cifre (numero di conto corrente)

Si scrivano prima le seguenti tre funzioni:

- `remove_spaces`, che prende in ingresso `str_in` e restituisce `str_out` conte-

nente tutti i caratteri di `str_in` tranne gli spazi.

- `all_upper`, che prende in ingresso una stringa e restituisce 1 solo se la stringa contiene soltanto caratteri maiuscoli, 0 altrimenti.
- `all_digit`, che prende in ingresso una stringa e restituisce 1 solo se la stringa contiene solo caratteri corrispondenti a cifre, 0 altrimenti.

Si usino poi tali funzioni per scrivere la funzione `check_iban` che richiede all'utente l'inserimento di un codice IBAN e restituisce 1 solo se, una volta tolti gli spazi dalla stringa IBAN, essa rispetta lo schema previsto.

Esercizio 12.4

Il sistema di messaggistica di Facebook permette di ricevere messaggi da qualsiasi mittente. Un messaggio è caratterizzato da un mittente e da un testo.

Vogliamo implementare un sistema di filtraggio per rilevare automaticamente messaggi indesiderati, basandoci sulle seguenti ipotesi semplificative:

- se il messaggio è ricevuto da una persona conosciuta, ossia da una persona nella lista degli amici, allora il messaggio non è da scartare
- se il messaggio è ricevuto da una persona sconosciuta, ossia non presente nella lista degli amici, allora è necessario esaminare la storia dei messaggi ricevuti in passato, per determinare un "valore atteso" che ci permetta di decidere se il messaggio appena ricevuto è "nella media".

Quindi servirà una funzione:

```
1 [buono, motivo] = filtra_messaggio(messaggio, messaggi, amici);
```

che riceve in ingresso:

- `messaggio`: una struttura dati a due campi: mittente (nome e cognome) e testo. Ad esempio:

```
1 messaggio.testo = 'Ciao come stai?';
2 messaggio.mittente.nome = 'Federico';
3 messaggio.mittente.cognome = 'Maggi';
```

- `messaggi`: un vettore di messaggi (definiti come `messaggio`) contenente i messaggi ricevuti in passato;
- `amici`: un vettore contenente gli amici. Utilizzeremo una struttura dati contenente i campi nome e cognome.

e restituisce:

- `buono`: di tipo logical ed è vero solo se il messaggio è buono;
- `motivo`: di tipo char, e vale:
  - `a` ad indicare che il messaggio è buono perchè inviato da un amico;
  - `m` ad indicare che il messaggio è buono perchè "nella media" dei messaggi passati;
  - `x` ad indicare che il messaggio è cattivo perchè non ha passato nessuno dei due criteri precedenti;

Per capire se un messaggio è "nella media" controlleremo se la sua lunghezza, senza spazi, il numero di vocali e il numero di consonanti sono simili a quelli medi dei messaggi passati. Implementare:

```
1 [l v c] = estrai_caratteristiche(testo)
```

- `testo` è il testo del messaggio da analizzare;
- `l` è la lunghezza del messaggio, esclusi gli spazi;
- `v` è il numero di vocali;
- `c` è il numero di consonanti.

```
1 [Mm, Dm] = valore_atteso(messaggi)
```

- Mm e' un vettore riga di 3 colonne, con il valor medio dei tre valori [l v c] calcolati su tutti i messaggi
- Dm e' un vettore riga di 3 colonne, con la deviazione standard dei tre valori [l v c] calcolati su tutti i messaggi

```
1 buono = controlla_contenuto(messaggio, messaggi)
```

la quale ritornerà un valore logical vero solo se il messaggio ha le caratteristiche `[l v c]` che soddisfano tutte le tre seguenti condizioni:

- media(l) - sqrt(2) * std(l) <= l <= media(l) + sqrt(2) * std(l)
- media(v) - sqrt(2) * std(v) <= v <= media(v) + sqrt(2) * std(v)
- media(c) - sqrt(2) * std(c) <= c <= media(c) + sqrt(2) * std(c)

Supporre che le strutture dati `amici` e `messaggi` siano già disponibili in un file `facebook.mat` e siano caricate all'inizio dello script.

Esercizio 12.5

Modellizzare il gioco del Giacomonero. Esso si svolge nel seguente modo: il dealer assegna ad un giocatore due carte ed una a sè stesso. A questo punto il giocatore ha due possibilità: chiedere una carta o stare. Se la somma delle carte del giocatore supera il 21, egli sballa e risulta perdente. Se non ha sballato può continuare a chidere carte finché non decide di stare. Se alla fine di questo processo il giocatore non ha sballato, il dealer deve estrarre delle carte finché il suo punteggio è inferiore o uguale a 16. Se oltrepassa il 21 il banco sballa e vince il giocatore. In caso contrario, se il punteggio del banco è strettamente inferiore a quello del giocatore (e il giocatore non ha sballato), la vittoria va al giocatore, altrimenti al dealer.

Il gioco del Giacomonero si gioca con un 6 mazzi da 52 carte (dal due al re, quattro semi). Il punteggio delle figure (Fante, Donna e Re) equivale a 10, l'Asso vale a discrezione del giocatore 11 oppure 1. Le altre carte valgono quanto il loro numero.

Scrivere uno script e delle funzioni in `MATLAB` che:

- crei un mazzo di carte completo (`crea_mazzo`);
- mischi il mazzo ordinato (`mescola_mazzo`);
- estragga una carta (`estrai_carta`);
- conti i punti di una mano (`somma_carte`);
- implementi la logica del gioco (`main_giacomonero`);

Opzionale: implementare le il gioco del Blackjack nella sua versione originale, in modo da considerare le opzioni di split, assicurazioni e la regola sul 21 a due carte.[1]

Esercizio 12.6

Scrivere un programma che chieda all'utente di inserire una serie di dati contenenti ognuno i seguenti attributi:

- città (stringa)
- giorno (intero positivo)
- mese (intero positivo)
- anno (intero positivo)

[1] `https://en.wikipedia.org/wiki/Blackjack`

- tipo di misurazione (char)
- valore (reale)

Ad esempio, l'utente potrà inserire:

```
Milano
04
12
2012
10.5
N
```

Dopo aver acquisito una certa quantità di dati, il programma dovrà chiedere all'utente il nome di una città e un tipo di misurazione. A questo punto il programma cercherà nell'archivio tutti i record riguardanti la città e il tipo di misurazione richiesti. Stamperà poi a video i dati selezionati ed il relativo valore minimo, massimo e medio dei valori.

Soluzioni

Soluzione dell'esercizio 12.1

```
clear
clc
close all

%% Inizializzazione magazzino
magazzino.barcodes = [123 ; 1312 ; 12312 ; 1231 ; 99123];
magazzino.tipo = ['A' ; 'A' ; 'X' ; 'W' ; 'W' ];
magazzino.stock = [0 ; 300 ; 0 ; 6 ; 0 ];
magazzino.ordine = [23 ; 100 ; 2 ; 100 ; 0 ];

%% Chiamata a ricerca
messaggio = ricerca(magazzino,123);
disp(messaggio);

%% Chiamata a ricercaMancanti
prodotti_non_in_stock = ricercaMancanti(magazzino, 0);
disp(['Prodotti esauriti, ma in ordine: ' mat2str(
   prodotti_non_in_stock)]);
prodotti_non_ordinati = ricercaMancanti(magazzino, 1);
disp(['Prodotti esauriti e non in ordine: ' mat2str(
   prodotti_non_ordinati)]);
prodotti_esauriti = ricercaMancanti(magazzino, 2);
disp(['Prodotti con piu'' ordine che stock: ' mat2str(
   prodotti_esauriti)]);

%% Chiamata a aggiungiProdotti
barcode = 111;
tipo = 'X';
stock = 12;
ordine = 0;
magazzino = aggiungiProdotto(magazzino, barcode, tipo ,stock,
   ordine)
```

```
function msg = ricerca (magazzino, barcode)

bc_indici = find(magazzino.barcodes == barcode);
if isempty(bc_indici)
    msg = ['Il prodotto corrispondente al codice a barre ',
       num2str(barcode), ...
```

```
    ' non e'' in magazzino'];
else
    t = magazzino.tipo(bc_indici);
    s = magazzino.stock(bc_indici);
    o = magazzino.ordine(bc_indici);

    msg = ['Il prodotto corrispondente al codice a barre ',
       num2str(barcode), ...
    ' e'' di tipo ', num2str(t), '. Elementi in stock: ', ...
    num2str(s), ', in ordine: ', num2str(o) '.'];
end
```

```
function prodotti = ricercaMancanti(magazzino, P)

switch P
    case 0 % esauriti ma in ordine
        bc_indici = find(magazzino.stock == 0 & magazzino.
           ordine > 0);
    case 1 % esauriti e non in ordine
        bc_indici = find(magazzino.stock == 0 & magazzino.
           ordine == 0);
    case 2 % prodotti con piu' ordine che stock
        bc_indici = find(magazzino.ordine > magazzino.stock);
end

prodotti = magazzino.barcodes(bc_indici);
```

```
function magazzino = aggiungiProdotto(magazzino, barcode, tipo
   ,stock, ordine)

magazzino.barcodes = [magazzino.barcodes; barcode];
magazzino.tipo = [magazzino.tipo; tipo];
magazzino.stock = [magazzino.stock; stock];
magazzino.ordine = [magazzino.ordine; ordine];
```

Soluzione dell'esercizio 12.2

```
clear
close all
clc

cifra_iniziale = 50;

```

```
7  banco = cifra_iniziale;
8  storicoBanco = cifra_iniziale;
9
10 giocatore.nome = 'Pippo';
11 giocatore.budget = cifra_iniziale;
12 giocatore.posta = 5;
13 giocatore.fattoreVittoria = 1;
14 giocatore.storicoBudget = cifra_iniziale;
15
16 giocatore(2).nome = 'Pluto';
17 giocatore(2).budget = cifra_iniziale;
18 giocatore(2).posta = 1;
19 giocatore(2).fattoreVittoria = 36;
20 giocatore(2).storicoBudget = cifra_iniziale;
21
22 % iterazioni del gioco
23 while (siContinuaAGiocare(giocatore, banco))
24
25     % scegliere giocata del giocatore1
26     % se dispari == 0 giocatore 1 sceglie pari
27     % se dispari == 1 giocatore 1 sceglie dispari
28     dispari = round(rand(1));
29
30     % giro la roulette, numero random tra 0 - 37
31     % 37 equivale a 00
32     numero = giraLaRoulette();
33
34     %Calcolo del vettore della vittoria dei giocatori
35     if(numero == 37 || numero == 0)
36         vince([1, 2]) = 0;
37     else
38         if(mod(numero,2) == dispari)
39             vince(1) = 0;
40         else
41             vince(1) = 1;
42         end
43
44         if numero == 15
45             vince(2) = 1;
46         else
47             vince(2) = 0;
48         end
49     end
50
```

```
51      %Calcolo ricompense giocatori e banco
52      for ii = 1 : numel(giocatore)
53          if giocatore(ii).budget >= giocatore(ii).posta
54              if vince(ii) == 0
55                  %Sconfitta giocatore
56                  giocatore(ii).budget =  giocatore(ii).budget -
                        giocatore(ii).posta;
57                  banco =  banco + giocatore(ii).posta;
58              elseif vince(ii) == 1
59                  %Vittoria giocatore
60                  giocatore(ii).budget =  giocatore(ii).budget +
                        giocatore(ii).fattoreVittoria * giocatore(ii
                        ).posta;
61                  banco =  banco - giocatore(ii).fattoreVittoria
                        * giocatore(ii).posta;
62              end
63          end
64      end
65
66      %Aggiorno storico giocatori e banco
67      for ii = 1 : numel(giocatore)
68          giocatore(ii).storicoBudget(end + 1) = giocatore(ii).
                budget;
69      end
70      storicoBanco(end + 1) = banco;
71
72  end
73
74  plotRoulette(giocatore, storicoBanco);
```

```
1   function numero = giraLaRoulette()
2
3   numero = randi(38)-1;
```

```
1   function res = siContinuaAGiocare(giocatore, banco)
2
3   budgetCorrenti = [giocatore.budget];
4   posta = [giocatore.posta];
5   res = (any(budgetCorrenti >= posta) && banco > 0);
```

```
1   function plotRoulette(giocatore, storicoBanco)
2
3   spessore = 2;
```

```
4
5  figure();
6  plot(giocatore(1).storicoBudget , 'r' , 'LineWidth' , spessore)
7  hold on;
8  plot(giocatore(2).storicoBudget, 'b' , 'LineWidth' , spessore)
9  plot(storicoBanco , 'k' , 'LineWidth' , spessore)
10 title('Evoluzione della Roulette nel tempo');
11 xlabel('Numero giocata');
12 ylabel('Euro');
13 legend(giocatore(1).nome , giocatore(2).nome, 'Banco','Location
      ','northwest');
```

Soluzione dell'esercizio 12.3

```
1 clear
2 clc
3 close all
4 %IT 02 L 12345 12345 123456789012
5 check_iban()
```

```
1 function str_out = remove_spaces(str_in)
2     str_out = str_in(str_in ~= ' ');
3 end
```

```
1 function str_out = all_alpha(str_in)
2     str_out = all(str_in >= 'A' & str_in <= 'Z');
3 end
```

```
1 function r = all_digit(str_in)
2     r = all(str_in >= '0' & str_in <= '9');
3 end
```

```
1 function is_valid = check_iban()
2     % Inserimento IBAN
3     iban = input('Inserire IBAN: ', 's');
4
5     % Rimuovo spazi
6     iban = remove_spaces(iban);
7
8     % Controllo validita'
9     is_valid = all_alpha(iban(1:2)) & all_digit(iban(3:4)) &
         all_alpha(iban(5)) & ...
```

```
        all_digit(iban(6:end)) & length(iban) == 27;

    % Oppure:
    %is_valid = all_alpha(iban([1, 2, 5])) & all_digit(iban([3, 4, 6:end])) & ...
    %    length(iban) == 27;
end
```

Soluzione dell'esercizio 12.4

```
clear
clc
close all

load('facebook.mat', 'amici', 'messaggi');

% Nuovo messaggio
% messaggio.testo = 'Ciao come stai?';
% messaggio.mittente.nome = 'Federico';
% messaggio.mittente.cognome = 'Maggi';
%
% % Controllo del contenuto di un messaggio, dati i messaggi precedenti
% [buono, motivo] = filtra_messaggio(messaggio, messaggi, amici);

% Nuovo messaggio
messaggio.testo = 'Cras aliquam massa ullamcorper sapien';
messaggio.mittente.nome = 'Federico';
messaggio.mittente.cognome = 'Maggi';

% Controllo del contenuto di un messaggio, dati i messaggi precedenti
[buono, motivo] = filtra_messaggio(messaggio, messaggi, amici);
```

```
function [buono, motivo] = filtra_messaggio(messaggio, messaggi, amici)

buono = 0;
for ii = 1:numel(amici)
    if strcmp(messaggio.mittente.nome,amici(ii).nome) && ...
            strcmp(messaggio.mittente.cognome,amici(ii).cognome)
```

```
        buono = 1;
        motivo = 'a';
        return;
    end
end


if controlla_contenuto(messaggio, messaggi)
    buono = 1;
    motivo = 'm';
else
    motivo = 'x';
end
```

```
function [l, v, c] = estrai_caratteristiche(testo)

spazi = testo == ' ';
vocali = testo == 'a' | testo == 'e' | testo == 'i' | testo ==
   'o' | testo == 'u';
consonanti = ~spazi & ~vocali;

l = length(testo(~spazi));
v = length(testo(vocali));
c = length(testo(consonanti));
```

```
function [Mm, Dm] = valore_atteso(messaggi)

N = length(messaggi);
M = zeros([N 3]);

% per ogni messaggio
for ii = 1:N
    msg = messaggi(ii).testo;
    [l, v, c] = estrai_caratteristiche(msg);
    M(ii, :) = [l v c];
end

Mm = mean(M);
Dm = std(M);
```

```
function buono = controlla_contenuto(messaggio, messaggi)

% calcolo valore atteso su tutti i messaggi
```

```
[Mm, Dm] = valore_atteso(messaggi);

% caratteristiche del testo da esaminare
msg = messaggio.testo;
[l, v, c] = estrai_caratteristiche(msg);
F = [l v c];

% estremi inferiori
int_inf = Mm - sqrt(2) * Dm;

% estremi superiori
int_sup = Mm + sqrt(2) * Dm;

% confronto tutte le caratteristiche con l'intervallo cosi`
   costruito
buono = all(F >= int_inf) && all(F <= int_sup);
```

Soluzione dell'esercizio 12.5

```
clear
clc
close all

mazzo = crea_mazzo();
mazzo = mescola_mazzo(mazzo);

[mazzo, carte_banco] = estrai_carta(mazzo);
disp(['Il banco ha ' carte_banco.numero ' di ' carte_banco.seme
   ]);
[mazzo, carte_giocatore] = estrai_carta(mazzo);
disp(['Il giocatore ha ' carte_giocatore.numero ' di '
   carte_giocatore.seme]);
[mazzo, carte_giocatore(2)] = estrai_carta(mazzo);
disp(['Il giocatore ha ' carte_giocatore(2).numero ' di '
   carte_giocatore(2).seme]);

play = 'S';
while (play == 'S')
    %Giocata giocatore
    play = input('Vuoi una carta? (S,N) ','s');
    if play == 'S'
        [mazzo, carte_giocatore(end+1)] = estrai_carta(mazzo);
```

```
21          disp(['Il giocatore ha pescato ' carte_giocatore(end).
               numero ' di ' carte_giocatore(end).seme]);
22      end
23
24      if somma_carte(carte_giocatore) > 21
25          disp('Hai sballato!!!');
26          play = 'N';
27      end
28
29  end
30
31  %% Giocata banco
32  flag_banco = 0;
33  while somma_carte(carte_giocatore) <= 21 && flag_banco == 0
34      [mazzo, carte_banco(end+1)] = estrai_carta(mazzo);
35      disp(['Il banco ha pescato ' carte_banco(end).numero ' di '
            carte_banco(end).seme]);
36
37      if somma_carte(carte_banco) > 16
38          flag_banco = 1;
39      end
40  end
41
42  %% Controllo vittoria
43  if somma_carte(carte_giocatore) > somma_carte(carte_banco) &&
       somma_carte(carte_giocatore) < 22
44      disp('Hai vinto');
45  else
46      disp('Vince il banco');
47  end
```

```
1   function mazzo = crea_mazzo()
2
3   numeri = '1234567689JQK';
4   semi = 'CQFP';
5   count = 1;
6   singolo_mazzo = struct('seme',[],'numero',[]);
7   for ii = 1:length(semi)
8       for jj = 1:length(numeri)
9           singolo_mazzo(count).seme = semi(ii);
10          singolo_mazzo(count).numero = numeri(jj);
11          count = count + 1;
12      end
13  end
```

```
14
15 mazzo = [];
16 for ii = 1:6
17     mazzo = [mazzo singolo_mazzo];
18 end
```

```
1  function mazzo = mescola_mazzo(mazzo)
2
3  n_carte = length(mazzo);
4  ind = randperm(n_carte);
5  mazzo = mazzo(ind);
```

```
1  function [mazzo, carta] = estrai_carta(mazzo)
2
3  carta = mazzo(1);
4  mazzo(1) = [];
```

```
1  function somma = somma_carte(carte)
2
3  somma = 0;
4  flag_asso = 0;
5  for ii = 1:length(carte)
6      if carte(ii).numero == 'J' || carte(ii).numero == 'Q' ||
          carte(ii).numero == 'K'
7          somma = somma + 10;
8      elseif str2double(carte(ii).numero) > 1
9          somma = somma + str2double(carte(ii).numero);
10     elseif flag_asso == 0
11         somma = somma + 11;
12         flag_asso = 1;
13     else
14         somma = somma + 1;
15     end
16 end
17
18 if somma > 21 && flag_asso == 1
19     somma = somma - 10;
20 end
```

Soluzione dell'esercizio 12.6

```
1  clear
2  clc
```

```
close all

% acquisizione dati
dati = acquisizione_dati_meteo();

% richiesta dato da visualizzare
[city, tipo] = interrogazione_archivio_meteo();

% ricerca dati e restituzione min, media, max
[dati_selezionati, minimo, medio, massimo] = ...
    calcolo_statistiche_meteo(dati, city, tipo);

% stampa a video delle statistiche
stampa_statistiche(dati_selezionati, city, tipo, minimo, medio,
    massimo);
```

```
function dati = acquisizione_dati_meteo()
    next = 1;
    dati = [];
    ii = 0;

    while next == 1
        ii = ii + 1;

        dati(ii).city = input('Citta'': ','s');
        dati(ii).giorno = input('Giorno: ');
        dati(ii).mese = input('Mese: ');
        dati(ii).anno = input('Anno: ');
        dati(ii).tipo = input('Tipo: ','s');
        dati(ii).valore = input('Valore: ');

        next = input('Per inserire un nuovo record premere 1,
           altrimenti 0: ');
    end

    fprintf('%d dati inseriti.\n', ii);
end
```

```
function [city, tipo] = interrogazione_archivio_meteo()
    city = input('Citta` di interesse: ', 's');
    tipo = input('Tipo misura da selezionare: ', 's');
end
```

```
function [dati_selezionati, minimo, medio, massimo] = ...
    calcolo_statistiche_meteo(dati, city, tipo)

    for ii = 1:numel(dati)
        res(ii) = strcmp(dati(ii).city,city);
    end

    indici = res & [dati.tipo] == tipo;

    dati_selezionati = dati(indici);

    minimo = min([dati_selezionati.valore]);
    massimo = max([dati_selezionati.valore]);
    medio = mean([dati_selezionati.valore]);
end
```

```
function stampa_statistiche(dati_selezionati, city, tipo,
   minimo, medio, massimo)
    fprintf('Statistiche della misura %c in citta'' %s\n', tipo
       , city);

    for r = dati_selezionati
        fprintf('%d/%d/%d %f\n', r.giorno, r.mese, r.anno, r.
           valore);
    end

    fprintf('\nMin: %3.2f, med: %3.2f, max: %3.2f\n', minimo,
       medio, massimo);
end
```

13 Ricorsione con MATLAB

13.1 Esercizi

Esercizio 13.1

Scrivere una funzione che verifichi iterativamente se una stringa è palindroma. Scrivere poi una funzione che implementi la stessa funzionalità in modo ricorsivo.

Si stampi a schermo il passo iterativo e ricorsivo che viene seguito dalla soluzione proposta.

Esercizio 13.2

Implementare una funzione iterativa (e poi una sua versione ricorsiva) per tradurre i caratteri di una stringa da minuscoli a maiuscoli. Assumere che la funzione riceva in ingresso una stringa di caratteri minuscoli.

Suggerimento: la traduzione viene effettuata semplicemente sottraendo 32 al carattere da tradurre, e applicando `char()`. Ad esempio:

```
1 trad = char('a' - 32)
```

stampa a video il carattere `A`.

Esercizio 13.3

Si implementi in `MATLAB` una funzione che, preso un intero positivo `n` ne restituisca i coefficienti binomiali di ordine `n`.

Suggerimento: la costruzione del triangolo di tartaglia restituisce come risultato i coefficienti binomiali. Esso viene definito in maniera ricorsiva come:

- Il primo elemento è il numero uno;
- La riga successiva viene costruita mettendo degli 1 agli estremi e sommando a coppie i numeri vicini della riga precedente.

Esercizio 13.4

Si implementi in MATLAB uno script che, preso un array A, calcola il massimo comun divisore (MCD) fra tutti gli elementi di A. Ad esempio: sull'array `[9 18 6 27 30 42]` restituisce 3.

Si ricorda che, dati tre numeri a, b, c, l'MCD tra a, b e c è uguale all'MCD tra c e l'MCD tra a e b.

Esercizio 13.5

Si scriva una funzione `calcolaZeri(f, a, b, tolX, tolY)` che prende in ingresso:

- un function handle f
- un estremo inferiore di un intervallo a
- un estremo superiore di un intervallo b
- una tolleranza per la variabile x chiamata $tolX$
- una tolleranza per la variabile y chiamata $tolY$

che calcoli uno zero della funzione f, se esso esiste, in maniera ricorsiva tramite il metodo di bisezione. La funzione si deve fermare se l'intervallo è diventato più corto di $tolX$ oppure se la funzione ha valore assoluto in un punto minore di $tolY$.

Si chiami la funzione sull'intervallo $[-1; 3]$ per la funzione $y = x^2 - 4$ con tolleranze 0.0001 per entrambe le variabili.

Esercizio 13.6

Si implementi in MATLAB uno script che, preso un array v, lo ordini ricorsivamente utilizzando l'algoritmo Mergesor, definito come segue:

- Se la sequenza da ordinare ha lunghezza 1, è già ordinata
- Se la sequenze è lunga 2 o più, la sequenza viene divisa in due metà;
- Ognuna di queste sottosequenze viene ordinata, applicando ricorsivamente l'algoritmo;
- Le due sottosequenze ordinate vengono fuse. Per fare questo, si estrae ripetutamente il minimo delle due sottosequenze e lo si pone nella sequenza in uscita, che risulterà ordinata.

Soluzioni

Soluzione dell'esercizio 13.1

```
1
2
3  % parola palindroma
4  R = 'abbAbba';
5
6  % parola non palindroma
7  P = 'abbiibaia';
8
9  % invocazione funzione iterativa
10 fprintf('
      ----------------------------------------------------\n');
11 palindroma_iterativa(R);
12 fprintf('
      ----------------------------------------------------\n');
13 palindroma_iterativa(P);
14
15 % invocazione funzione ricorsiva
16 fprintf('
      ----------------------------------------------------\n')
      ;
17 palindroma_ricorsiva(R, 0);
18 fprintf('
      ----------------------------------------------------\n')
      ;
19 palindroma_ricorsiva(P, 0);
```

```
1  function [res] = palindroma_iterativa(parola)
2  res = 1;
3
4  for ii = 1:floor(length(parola)/2)
5      %Stampa a schermo del passo iterativo
6      fprintf('passo iterativo = %d: %c =?= %c\n', ii, parola(ii)
          , parola(end - ii + 1));
7
8      %Controllo di caratteri nella parola
9      if parola(ii) ~= parola(end - ii + 1)
10         res = 0;
11         return;
12     end
```

```
end
```

```
function [res] = palindroma_ricorsiva(parola, passo)

    %Stampa inizio passo ricorsivo
    for ii = 1:passo
        fprintf('\t');
    end
    fprintf('|--> inizio palindroma_ricorsiva(%s, %d)\n',
       parola, passo);

    if length(parola) < 2
        res = 1;
    else
        % controllo se gli estremi sono uguali
        %
        if parola(1) == parola(end)
            % Passo ricorsivo
            res = palindroma_ricorsiva(parola(2:end-1), passo
               +1);
            % da qui, l'esecuzione e` bloccata fino a che la
               funzione
            % precedente non ritorna
        else
            res = 0;
        end
    end

    %Stampa fine passo ricorsivo
    for ii = 1:passo
        fprintf('\t');
    end
    fprintf('|--> fine   palindroma_ricorsiva(%s, %d)\n',
       parola, passo);
```

Soluzione dell'esercizio 13.2

```
clc
clear
close all

s = 'ciaocomestai';

```

```
fprintf('
   ------------------------------------------------------\n');
tic
S = maiuscola_iterativa(s);
disp(S)
toc

fprintf('
   ------------------------------------------------------\n');
tic
S = maiuscola_ricorsiva1(s);
disp(S);
toc

fprintf('
   ------------------------------------------------------\n');
tic
S = maiuscola_ricorsiva2(s);
disp(S);
toc
```

```
function S = maiuscola_iterativa(s)

S = s;
for ii = 1:length(s)
    S(ii) = char(s(ii) - 32);
end
```

```
function S = maiuscola_ricorsiva1(s)

% Caso base: stringa di un carattere
if length(s) == 1
    S = [char(s(1) - 32)];
else
    S = [char(s(1) - 32) maiuscola_ricorsiva1(s(2:end))];
end
```

```
function S = maiuscola_ricorsiva2(s)

% Caso base: stringa di un carattere
if length(s) == 1
    S = char(s(1) - 32);
else
```

```
    m = round(length(s) / 2);
    S = [maiuscola_ricorsiva2(s(1:m)) maiuscola_ricorsiva2(s((m
       +1):end))];
end
```

Soluzione dell'esercizio 13.3

```
clear
clc
close all;

triangoloTartaglia(10)
```

```
function T = triangoloTartaglia(n)

% Caso base
if n == 0
    T = 1;
% Chiamata ricorsiva
else
    T = triangoloTartaglia(n-1);

    % Soluzione alla C :creo un vettore I con la somma delle
       coppie degli elementi di T al passo n-1
    %     I = [];
    %     for ii = 2 : numel(T)
    %         I = [I, T(ii- 1)+ T(ii)];
    %     end
    %     T=[1, I, 1];

    % Soluzione alla matlab, shifto T
    T = [1,T(1:end-1)+T(2:end),1];
end
disp(T)
```

Soluzione dell'esercizio 13.4

```
clear
clc
close all

res = mcd(15, 18, 0)
res = mcd_array([12 15 18])
```

```
% Calcolo dell'MCD con algoritmo di Euclide
%
%     * se m = n, MCD(m,n) = m              (caso base)
%     * se m > n, MCD(m,n) = MCD(m-n, n)    (ricorsione)
%     * se m < n, MCD(m,n) = MCD(m, n-m)    (ricorsione)

function [M] = mcd(m, n, passo)
% la variabile "passo" non e` parte della soluzione.

% stampo "passi volte" il carattere TAB per "visualizzare" a
   che punto
% della ricorsione mi trovo
for ii = 1:passo
    fprintf('\t');
end
% stampo l'inizio dell'invocazione corrente
fprintf('|--> inizio esecuzione mcd: m = %d, n = %d, passo = %d
   \n', m, n, passo);

if m == n
    M = m;
else
    if m > n
        M = mcd(m-n, n, passo+1);
    else
        M = mcd(m, n-m, passo+1);
    end
end

% stampo "passi volte" il carattere TAB per "visualizzare" a
   che punto
% della ricorsione mi trovo
for ii = 1:passo
    fprintf('\t');
end
% stampo la fine dell'invocazione corrente
fprintf('|--> fine esecuzione mcd: m = %d, n = %d, passo = %d\n
   ', m, n, passo);
```

```
function res = mcd_array(v)

len_v = length(v);

```

```
5  if len_v == 1
6      res = v;
7  else
8      res = mcd(v(1),v(2),0);
9      for ii = 3:len_v
10         res = mcd(res,v(ii),0);
11     end
12 end
```

Soluzione dell'esercizio 13.5

```
1  clear
2  clc
3  close all
4
5  f = @(x)(x.^2 - 4);
6  a = -1;
7  b = 3;
8  tolX = 0.0001;
9  tolY = 0.0001;
10
11 zero = calcolaZeri(f,a,b,tolX,tolY);
12
13 x = a:0.01:b;
14 y = f(x);
15
16 plot(x,y);
17 hold on;
18 plot(zero,f(zero),'ro');
```

```
1  function x = calcolaZeri(f, a, b, tolX, tolY)
2
3  if (f(a) * f(b) > 0)
4      x = NaN;
5      return;
6  end
7
8  if (abs(f(a)) < tolY)
9      x = a;
10     return;
11 elseif (abs(f(b)) < tolY)
12     x = b;
13     return;
```

```
14 end
15
16 if ((b-a) < tolX)
17     x = (b-a) / 2;
18     return;
19 end
20
21 c = (b - a) / 2;
22 if (f(a) * f(c) < 0)
23     x = calcolaZeri(f, a, c, tolX, tolY);
24 else
25     x = calcolaZeri(f, c, b, tolX, tolY);
26 end
```

Soluzione dell'esercizio 13.6

```
1 clear
2 clc
3 close all
4
5 v = rand(100,1);
6 plot(v);
7 hold on;
8 sorted_v = merge_sort(v);
9 plot(sorted_v,'r');
```

```
1  function v = merge_sort(v)
2
3  len_v = length(v);
4  if  len_v > 1
5      %Divide
6      middle_point = round(len_v / 2);
7      v1 = merge_sort(v(1:middle_point));
8      v2 = merge_sort(v(middle_point+1:end));
9
10     %Impera
11     len_v1 = length(v1);
12     len_v2 = length(v2);
13     ii = 1;
14     jj = 1;
15     v = [];
16     while ii <= len_v1 && jj <= len_v2
17         if v1(ii) < v2(jj)
```

```
18              v = [v v1(ii)];
19              ii = ii + 1;
20          else
21              v = [v v2(jj)];
22              jj = jj + 1;
23          end
24      end
25      if ii <= len_v1
26          v = [v v1(ii:end)];
27      else
28          v = [v v2(jj:end)];
29      end
30      assert(length(v) == len_v1 + len_v2)
31  end
```