Facoltà di Ingegneria
Università di Udine
Dispense del corso di

FONDAMENTI DI INFORMATICA
(Ing. Meccanica)
A. A. 2013-14

Compiti d'esame svolti

a cura di
Andrea Schaerf

26 settembre 2013

Introduzione

Obiettivi formativi

Introduzione all'informatica e alla programmazione in linguaggio C, illustrando e applicando i princìpi della programmazione strutturata e modulare.

Programma del corso

L'elaborazione automatica: La nozione di algoritmo. Linguaggi per la descrizione di algoritmi. I diagrammi di flusso. Linguaggi di programmazione. Cenni sul sistema di elaborazione.

Elementi del linguaggio C: La sintassi del C. Assegnazioni, espressioni e istruzioni. Input ed output. Tipi di dato semplici. Strutture di controllo condizionali e di ciclo. Tipi di dato strutturati: vettori e record. Funzioni e procedure. Passaggio dei parametri e valore restituito. I puntatori. La gestione dei file: lettura e scrittura da/su file.

Principi e tecniche di programmazione: Il ciclo di vita del software. Le qualità dei programmi. Astrazione e decomposizione. La programmazione modulare.

Libro di testo

H.M. Deitel e P.J. Deitel, Il linguaggio C (Fondamenti e tecniche di programmazione). Pearson, 2013.
(Parti del testo da saltare: Cap 7: 7.5-7.7, 7.11-7.13; Cap 8: 8.8-8.11; Cap 10: 10.7-10.11; Cap 11: 11.6-11.10; Cap 12: tutto. Cap 13: 13.4-13.11; Cap 14: 14.1-14.3, 14.5-14.10. Cap 15: 15.1-15.7, 15.9, 15.10, 15.12, 15.13.

Modalità d'esame

L'esame si compone di 3 parti: compito scritto, prova a casa e discussione orale:

Compito scritto: Il compito scritto si compone di un insieme di esercizi di programmazione in C. Durante il compito, la cui durata è normalmente di 2 ore, non è consentito consultare libri, appunti, dispense o qualsiasi altro materiale e non è consentito uscire dall'aula.

La valutazione dello scritto verrà resa nota solo *dopo* la discussione dello scritto e dell'elaborato svolto a casa, quindi tutti gli studenti che hanno sostenuto lo scritto sono implicitamente ammessi a sostenere l'orale.

Prova a casa: Nel tempo che intercorre tra il compito scritto e la prova orale (tipicamente 3-4 giorni) lo studente è tenuto a compilare ed eseguire la soluzione del compito.

La soluzione proposta a casa deve essere tendenzialmente la stessa di quella svolta nel compito, e la sua correttezza deve essere verificata scrivendo tutte le parti di codice mancanti e usando dei dati di test scelti opportunamente. Nel caso la soluzione svolta in classe non sia corretta, questa deve essere modificata opportunamente fino ad ottenere il corretto funzionamente su tutti i dati di test.

Nel caso lo studente non abbia svolto parte dell'esercizio nel compito, dovrà comunque portare la soluzione svolta a casa; tale soluzione può ovviamente essere qualsiasi.

Scopo principale della prova a casa è quello di fare una autovalutazione precedente all'orale, che porta ad una discussione (e valutazione) dello scritto più proficua. In particolare, tale soluzione serve allo studente per dimostrare che, a parte eventuali modifiche, la soluzione proposta nel compito è corretta.

Discussione orale: Lo studente deve presentarsi all'orale obbligatoriamente con una memoria USB contenente i file che compongono la nuova soluzione e possibilmente con la stampa dei file stessi e dei test effettuati.

La discussione orale è costituita essenzialmente da una discussione del compito scritto e della soluzione svolta a casa dallo studente (più eventuali domande aggiuntive a discrezione del docente).

Materiale disponibile via rete

All'indirizzo `http://www.diegm.uniud.it/schaerf/Fondamenti/` sono disponibili:

Compiti d'esame: La versione elettronica di questa dispensa e i testi d'esame che si sono aggiunti successivamente.

Esercitazioni: Il testo e le soluzioni di tutte le esercitazioni del corso.

Altro materiale: Programmi svolti in aula, esercizi aggiuntivi ed informazioni generali sul corso.

Utilizzo del compilatore `gcc` in ambiente `cygwin`

Prerequisiti

- Sistema operativo Windows XP o successivo (per Linux il compilatore è presente nella distribuzione iniziale, per Mac OS occorre scaricare `XCode`).

Installazione di `cygwin`

- Dal sito `http://www.cygwin.com` scaricare il file `setup-x86.exe` (per una installazione a 32-bit) o `setup-x86_64.exe` (per una installazione a 64-bit). Si proceda selezionando *Install from Internet* nella prima finestra di dialogo, e accettando tutti i default proposti dal programma di installazione. Quando appare la finestra in cui sono listati i pacchetti da installare, selezionare dalla categoria *Devel* i pacchetti `gcc` e `gdb`. Per tutti gli altri pacchetti accettare la scelta di default. L'installazione richiede pochi minuti.

- Verifica dell'installazione: Al termine dell'installazione il programma avrà creato un'icona (presente sia sul desktop che sul menù Avvio) con nome Cygwin. Cliccando su tale icona si apre una finestra con interfaccia a linea di comando il cui *prompt* è il carattere `$`. A questo punto, per verificare se il compilatore è stato installato correttamente si può digitare `gcc -v` e si otterrà un messaggio del tipo:

```
Reading specs from ...
gcc version ...
```

Installazione di `Notepad++`

Dal sito `http://notepad-plus-plus.org` scaricare il file `npp.X.Y.Z.Installer.exe` (la versione corrente è la 6.4.5). Eseguire il file selezionando la cartella di installazione nella prima finestra di dialogo, e accettando tutti i default proposti dal programma di installazione. Anche questa installazione richiede pochi minuti.

Nota: L'uso dell'editor di testi `Notepad++` non è obbligatorio, ma semplicemente consigliato. Qualsiasi altro editor che permetta di salvare i programmi in formato testo va altrettanto bene.

Testi

Compito del 19 marzo 2001 (soluzione a pagina 56)

Esercizio 1 (punti 16) Un file contiene una sequenza (di lunghezza ignota) di numeri interi positivi, uno per riga. I numeri sono scritti in lettere cifra per cifra, e sono terminati dalla parola `stop`. Come esempio si consideri il seguente file:

```
otto cinque nove stop
due due stop
sette zero sette stop
```

Il file contiene i numeri `859`, `22` e `707`.

Si scriva un programma C, opportunamente organizzato in funzioni, che legga da tastiera il nome del file e stampi il valore della somma dei numeri ivi contenuti. Nell'esempio il programma deve stampare `1588`.

Si assuma che i numeri siano di dimensione tale da essere rappresentabili tramite il tipo `int`.

Esercizio 2 (punti 12) Si consideri il seguente gioco a due giocatori. In una partita vengono lanciati un dado per ogni giocatore ed una sola moneta. Se il lancio della moneta è testa, la partita è vinta dal giocatore che ha ottenuto il punteggio più alto nel lancio del proprio dado. Se invece esce croce, la partita è vinta dal giocatore che ha ottenuto il punteggio più basso. Se i dadi hanno lo stesso valore la partita è un pareggio.

Si consideri la seguente definizione di tipo

```
struct Partita
{
  char moneta;
  int ris_dado1;
  int ris_dado2;
};
```

che rappresenta il risultato dei lanci associati ad una partita. I campi `ris_dado1` e `ris_dado2` hanno un valore compreso tra 1 e 6, e il campo `moneta` ha valore `'t'` (testa) oppure `'c'` (croce).

Si scriva una funzione C che dati come parametri un vettore di partite e la sua lunghezza restituisca:

`1`	se il giocatore 1 ha vinto più partite del giocatore 2
`-1`	se il giocatore 2 ha vinto più partite del giocatore 1
`0`	in caso di pari numero di vittorie.

Esercizio 3 (punti 4) Si scriva la funzione `main` necessaria per verificare la correttezza della funzione dell'esercizio 2 (si noti che questo esercizio può essere svolto indipendentemente dall'esercizio 2).

Compito del 2 aprile 2001 (soluzione a pagina 58)

Esercizio 1 (punti 14) Un file contiene una sequenza (di lunghezza ignota) di studenti iscritti per una prova d'esame. Ogni riga del file contiene la matricola, il nome, il cognome e l'anno di corso di uno studente, più un eventuale commento. Come esempio si consideri il seguente file.

```
29333 Mario,       Rossi,      2 RG
34211 Irene,       De Neri,    3 FC
23432 Pier Maria,  Bianchi,    3 RG  ciao mamma
35211 Giuseppe,    Verdi,      4 FC  odio questo esame!
22222 Chiara,      Blu,        2 RG
```

Si noti che il nome e il cognome possono contenere spazi bianchi e sono seguiti da una virgola. Si noti inoltre che l'anno di corso è composto da un numero, uno spazio bianco e i due caratteri `FC` o `RG`, a seconda che lo studente sia fuori corso o regolare, rispettivamente. Si assuma infine che il numero di spazi bianchi dopo la virgola sia arbitrario.

Si scriva una funzione C che prenda come parametri il nome del file e un anno di corso (un intero tra 1 e 5), e restituisca il numero di studenti iscritti fuori corso per quell'anno.

Nel caso che il file non esista, oppure l'anno di corso non sia compreso tra 1 e 5 la funzione deve restituire il valore `-1`.

Esercizio 2 (punti 4) Si scriva una funzione `main` necessaria per verificare la correttezza della funzione dell'esercizio 1. La funzione deve ricevere il nome del file e l'anno di corso sulla linea di comando.

Esercizio 3 (punti 14) Si considerino le seguenti definizioni di tipo che rappresentano rispettivamente un numero complesso ed un segmento nel piano complesso (identificato dai suoi estremi).

```
struct Complesso
{
   float re; /* parte reale */
   float im; /* parte immaginaria */
};

struct Segmento
{
    struct Complesso p1;
    struct Complesso p2;
};
```

Diciamo che un segmento *appartiene* ad un quadrante (tra 1 e 4) se entrambi i suoi estremi sono contenuti nello stesso quadrante. Se gli estremi sono in quadranti diversi, un segmento non appartiene ad alcun quadrante. Si assuma che i punti non possano giacere sugli assi del piano.

Si scriva una funzione C che prenda come parametri un vettore di segmenti e la sua dimensione e restituisca un valore tra 1 e 4 che corrisponde al quadrante del piano complesso a cui appartiene il maggior numero di segmenti.

Compito del 9 luglio 2001 **(soluzione a pagina 61)**

Esercizio 1 (punti 16) Un file contiene una sequenza (di lunghezza ignota) di uguaglianze tra somme di interi positivi, una per riga, ciascuna terminata da un punto e virgola, e senza spazio bianchi. Come esempio, si consideri il seguente file

```
2+3+12=9+8;
2+3+4=9;
22=3+4+5+10;
3+5+1=4+44;
```

Si noti che le uguaglianze possono essere sia corrette (le prime tre) che sbagliate (l'ultima).

Si scriva una funzione C che prenda come parametro il nome del file e restituisca la frazione di uguaglianze corrette del file. Nell'esempio la funzione deve restituire 0.75.

Esercizio 2 (punti 14) Le immagini *bitmap* sono rappresentate mediante una matrice di punti (pixel) ciascuno dei quali è descritto da una tripletta di colori RGB, che contiene i valori di intensità, da 0 a 255, dei tre colori fondamentali rosso, verde e blu. Si considerino dei file bitmap organizzati come

segue: la prima riga contiene, nell'ordine, i due numeri interi N e M che rappresentano il numero di righe e il numero di colonne della matrice di pixel; poi il file contiene le triplette che rappresentano i pixel (un pixel per riga) a partire dalla prima riga, prima colonna (pixel in alto a sinistra nell'immagine), e procedendo con tutte le colonne della riga. Ogni pixel è rappresentato dai tre numeri nella sequenza: rosso, verde, blu.

Ad esempio, il file a sinistra rappresenta un'immagine 3×4 (4 righe e 3 colonne) con un punto in alto a sinistra bianco (i tre colori sono al massimo dell'intensità) e il resto della prima riga nero (tutti zeri); poi una riga grigia, e due righe con punti rossi $(255, 0, 0)$, verdi $(0, 255, 0)$, blu $(0, 0, 255)$, azzurri $(0, 255, 255)$, gialli $(255, 255, 0)$ e magenta $(255, 0, 255)$.

```
4 3
255 255 255
0   0   0
0   0   0
127 127 127
127 127 127
127 127 127
255 0   0
0   255 0
0   255 255
0   0   255
255 255 0
255 0   255
```

```
6 5
255 255 255
255 255 255
255 255 255
255 255 255
255 255 255
255 255 255
255 255 255
0 0 0
0 0 0
255 255 255
255 255 255
127 127 127
127 127 127
127 127 127
255 255 255
255 255 255
85 85 85
85 85 85
170 170 170
255 255 255
255 255 255
85 85 85
170 170 170
170 170 170
255 255 255
255 255 255
255 255 255
255 255 255
255 255 255
255 255 255
```

Si scriva un programma che trasformi un'immagine bitmap a colori in un'altra in bianco e nero, cioè in cui i tre colori di ogni tripletta hanno uguale intensità, pari alla media delle intensità dei colori del pixel originale. Inoltre, il programma deve aggiungere alla figura un contorno costituito da una linea bianca (spessa un pixel).

Il file sopra a destra è il risultato del programma applicato al file a sinistra.

I nomi dei file bitmap di ingresso e di uscita devono essere passati come parametri sulla linea di comando. Si supponga che le immagini abbiano dimensioni massime 600×480 (colonne $\times$ righe).

Esercizio 3 (punti 2) Si descriva almeno una delle principali qualità del software i si forniscano semplice esempi di programmi che godono (o non godono) di tale proprietà.

Compito del 23 luglio 2001 **(soluzione a pagina 64)**

Esercizio 1 (punti 24) Lo schema di un cruciverba è memorizzato in un file nel seguente formato: la prima riga contiene due numeri interi che indicano, rispettivamente, il numero di righe e il

numero di colonne del cruciverba; le righe successive riportano le righe del cruciverba, con uno spazio in corrispondenza delle caselle nere. Come esempio si consideri il file seguente.

```
8 6
SUGLI
ODIATI
TIRI N
TRI CD
IE POI
L MELA
EDISON
 CAINO
```

Come è noto, è necessaria una definizione per ogni parola, verticale o orizzontale, lunga almeno due lettere. Si scriva un programma C che riceva come argomento sulla linea di comando il nome del file contenente il cruciverba e scriva nel file `parole.txt` l'elenco delle parole di cui serve una definizione, suddivise in orizzontali e verticali (separate da una linea vuota). Non si richiede che l'ordine in cui compaiono le parole nel file sia lo stesso con cui vengono numerate le definizioni dei normali cruciverba. Si assuma che la dimensione massima di un cruciverba sia 50×50. Nell'esempio, il file `parole.txt` sarà il seguente.

```
SUGLI
ODIATI
TIRI
TRI
CD
IE
POI
MELA
EDISON
CAINO

SOTTILE
UDIRE
DC
GIRI
MIA
LAI
PESI
IT
COLON
INDIANO
```

Il programma deve essere realizzato in modo modulare, scomponendolo nelle seguenti funzioni:

- la funzione `main` con passaggio dell'argomento sulla linea di comando (5 punti);
- una funzione di lettura del cruciverba dal file (5 punti);
- una funzione di ricerca delle parole e scrittura sul file (14 punti).

Esercizio 2 (punti 8) Dato il seguente programma:

```
void f(int a, int b, int* c, int* d)
{ *c = 0; *d = a;
  while(*d >= b)
   { *d -= b;
     (*c)++;
   }
  a = 0; b = 1;
}
int main()
```

```
{ int m, n, p, q;
  scanf("%d%d%d%d", &m, &n, &p, &q);
  f(m,n,&p,&q);
  printf("%d %d %d %d", m, n, p, q);
}
```

Spiegare che cosa calcola la funzione `f`. Scrivere che cosa stampa il programma se l'input è: `20 3 25 4`.

Compito del 6 settembre 2001 **(soluzione a pagina 67)**

Un robot è in grado di muoversi nel piano nelle quattro direzioni nord, sud, est ed ovest (denotate rispettivamente dai caratteri '`N`', '`S`', '`E`' ed '`O`'). In particolare, il robot accetta comandi che consistono in una direzione e un intero positivo, che rappresenta la distanza (in metri) da percorrere in quella direzione. I comandi sono immagazzinati in un file, in cui ogni riga contiene un comando, e vengono eseguiti dal robot in sequenza.

Si riportano, come esempio, tre file di comandi:

```
N 10           | N 10           | N 10           |
E 5            | E 5            | E 5            |
S 4            | S 4            | S 4            |
S 9            | S 6            | S 6            |
O 3            | O 3            | O 5            |
---------------+----------------+----------------+
ESEMPIO 1      | ESEMPIO 2      | ESEMPIO 3     |
```

Esercizio 1 (punti 16) Dopo avere eseguito i comandi contenuti nel file di input, il robot si è spostato, rispetto alla posizione iniziale, in due (cfr. esempio 1), una (cfr. esempio 2) o nessuna (cfr. esempio 3) delle quattro direzioni '`N`', '`S`', '`E`' ed '`O`'.

Scrivere un funzione C che, ricevuto come parametro il nome del file di input, stampi su schermo in quali direzioni e di quanti metri si è spostato il robot. Ad esempio, facendo riferimento ai file sopra riportati, si devono effettuare le stampe:

```
3 METRI VERSO S | 2 METRI VERSO E |                |
2 METRI VERSO E |                  |                |
----------------+------------------+----------------+
ESEMPIO 1       |  ESEMPIO 2       | ESEMPIO 3      |
```

Esercizio 2 (punti 16) Si vuole determinare la sequenza di comandi, della stessa lunghezza di quella di input, da dare al robot per tornare al punto di partenza *ripercorrendo a ritroso esattamente la stessa strada*.

Si scriva una funzione C che prenda come parametro il nome del file di input ed il nome di un file di output e scriva sul file di output la sequenza determinata. Si assuma che la lunghezza massima della sequenza di comandi sia pari a 100.

Ad esempio, facendo riferimento ai file sopra riportati, le sequenze di comandi che fanno ritornare il robot al punto di partenza sono le seguenti:

```
E 3            | E 3            | E 5            |
N 9            | N 6            | N 6            |
N 4            | N 4            | N 4            |
O 5            | O 5            | O 5            |
S 10           | S 10           | S 10           |
---------------+----------------+----------------+
ESEMPIO 1      | ESEMPIO 2      | ESEMPIO 3      |
```

Compito del 20 settembre 2001 (soluzione a pagina 69)

Esercizio 1 (punti 16) Un dato linguaggio di programmazione prevede l'inserimento di commenti nel codice sorgente mediante il carattere `'#'` che deve precedere il testo del commento. Il commento termina alla fine della riga. Come esempio si consideri il seguente file.

```
# questo e' un commento
start: xor  r0 r0     # questo e' un altro commento
       mv   r1 r2     # ecc.
# inizia il ciclo
loop:  ldbr r3 r2     # copia in R3 l'i-esimo carattere della stringa
       jmpz end       # se vale zero ('\0') la stringa e' finita
       inc  r0        # incrementa il contatore
       inc  r2        # incrementa il puntatore
       jmp  loop      # ripete
end:   ret
# (commento) fine del programma
```

Si scriva una funzione C che riceva come parametro il nome del file sorgente contenente i commenti, che ha estensione `.tic`, e scriva un file con il medesimo nome ma estensione `.toc` contenente soltanto tutte le istruzioni e non i commenti. Il file risultante dall'esempio sarà:

```
start: xor  r0 r0
       mv   r1 r2

loop:  ldbr r3 r2
       jmpz end
       inc  r0
       inc  r2
       jmp  loop
end:   ret
```

Esercizio 2 (punti 16) Un file contiene l'elenco degli ingredienti di una ricetta con i relativi pesi, nel seguente formato: $\langle$ *ingrediente unità di misura quantità* $\rangle$. Le unità di misura possibili sono `l` (litri), `g` (grammi) e `u` (unità). Come esempio si consideri il seguente file.

```
latte  l  0.25
farina g  300
olio   l  0.05
uova   u  2
burro  g  50
yogurt g  50
```

Si consideri inoltre la seguente dichiarazione, con un esempio di vettore di record del tipo, che contiene i pesi specifici (in grammi) degli ingredienti liquidi e i pesi medi degli ingredienti misurati in unità (ad esempio le uova). Le unità di misura di riferimento sono le stesse utilizzate nel file della ricetta. Non è previsto l'uso né di multipli (es. kg) né di sottomultipli (es. ml).

```
struct PesiSpecifici
{
   char ingrediente[32];
   float peso;
};

struct PesiSpecifici V[4] = {{"latte", 1000}, {"olio", 950}, {"acqua", 1000}, {"uova", 75.0} };
```

che contiene i pesi specifici (in grammi) degli ingredienti liquidi e i pesi medi degli ingredienti misurati in unità (ad esempio le uova). Le unità di misura di riferimento sono le stesse utilizzate nel file della ricetta. Non è previsto l'uso né di multipli (es. kg) né di sottomultipli (es. ml).

Si scriva una funzione C che riceva come parametri:

1. il nome del file che contiene l'elenco degli ingredienti della ricetta
2. il vettore di pesi specifici
3. la lunghezza del vettore dei pesi specifici

e restituisce il peso totale del composto.

Nell'esempio, supponendo che il vettore dei pesi specifici passato alla funzione sia il vettore `V` sopra dichiarato la funzione deve restituire il valore `847.5`.

Compito del 10 dicembre 2001 (soluzione a pagina 71)

Esercizio 1 (punti 13) Un file contiene una sequenza (di lunghezza ignota, massimo 1000 elementi) di valori reali. Come esempio si consideri il seguente file.

```
2.3 4.56 2 1.23 8.65 10 -12.3 4.34 16.22 2.3
```

Si scriva una funzione C che prenda come parametri il nome del file e due valori interi s_1 ed s_2 (con $s_1 < s_2$), e modifichi il file stesso nel modo seguente: i valori v tali che $v < s_1$ vengono a trovarsi tutti all'inizio del file. I valori v tali che $s_1 \leq v < s_2$ vengono a trovarsi tutti al centro del file, e i valori v tali che $v \geq s_2$ vengono a trovarsi alla fine del file. L'ordine all'interno di ciascuna parte del file deve rimanere lo stesso del file originario. Infine, le tre parti del file vengono separate dal carattere `#`.

Ad esempio, se il file è quello precedente e le soglie sono rispettivamente 2 e 5, il file diventa come segue.

```
1.23 -12.3 # 2.3 4.56 2 4.34 2.3 # 8.65 10 16.22
```

Esercizio 2 (punti 15) Un listino prezzi in lire deve essere convertito in euro. Purtroppo, il listino è stato gestito in modo un po' disordinato, e la descrizione (massimo 127 caratteri) e il prezzo dei vari articoli non è strutturata in modo uniforme. Inoltre, alcuni articoli introdotti recentemente hanno il prezzo già espresso in euro. Ogni articolo è descritto da una stringa che può assumere una qualsiasi delle seguenti forme (`XXXX` rappresenta le cifre numeriche del prezzo in lire, `YYY.YY` rappresenta il prezzo in euro):

```
descrizione articolo: L. XXXXX
descrizione articolo: XXXXX lire
descrizione articolo: euro YYY.YY
descrizione articolo: YYY.YY euro
```

Scopo di questo esercizio è scrivere una funzione di servizio al programma che convertirà il file del listino prezzi. Si scriva pertanto una funzione in linguaggio C che riceva come parametro una stringa in uno qualsiasi dei formati descritti e la modifichi, se necessario, trasformandola nel formato:

```
descrizione articolo: YYY.YY euro
```

Come esempi si considerino le seguenti quattro stringhe.

```
scarpe da montagna:  L. 274000   →  scarpe da montagna:  141.51 euro
giacca a vento:  lire 492000     →  giacca a vento:  254.10 euro
berretto:  13.02 euro            →  berretto:  13.02 euro
occhiali da sole:  euro 122.17   →  occhiali da sole:  122.17 euro
```

Suggerimento: Può risultare utile la funzione `sprintf()` che ha un comportamento simile alla funzione `fprintf()`, ma memorizza su una stringa invece che scrivere su un file. Ad esempio, se `s1` è la stringa `"mario"`, ed `n` è uguale a `31`, l'invocazione `sprintf(s,"%s := %d;", s1, n)`, memorizza `"mario := 31;"` nella stringa `s`.

Esercizio 3 (punti 4) Si scriva un programma main per provare la funzione dell'esercizio 2.

Compito del 20 marzo 2002 (soluzione a pagina 74)

Esercizio 1 (punti 16) Un file contiene le informazioni sui movimenti di un conto corrente bancario (in euro) secondo il seguente formato:

- la prima riga contiene il numero di conto corrente ed il saldo iniziale (sempre positivo);
- le righe seguenti formano una sequenza (di lunghezza ignota) di *movimenti*, cioè di coppie data_movimento, espressa nel formato `gg/mm`, e importo_movimento, espresso come numero reale positivo seguito da uno spazio e dal segno `+` (entrata) o `-` (uscita). Ciascun movimento è posto su di una singola riga del file.

Un esempio di file secondo tale formato è riportato di seguito:

```
12345 2000.00
   01/01 1200.00 +
   20/01 100.50 -
   05/02 523.10 +
   10/02 3000.00 -
   14/02 1430.23 -
   20/03 1555.55 +
   01/04 56.00 +
```

Si scriva una funzione C che prenda come parametri il nome di un file siffatto e un limite di credito c (con $c < 0$), e restituisca la data in cui il saldo è sceso sotto il valore limite c. Nel caso che il saldo non scenda mai sotto tale valore, la funzione deve restituire la data dell'ultimo movimento.

Ad esempio, nel file precedente se $c = -500.00$, la funzione restituisce la data 14/2. Se invece $c = -1000.00$, la funzione restituisce la data 1/4 dell'ultimo movimento.

Esercizio 2 (punti 16) Sia dato un file che contiene una matrice rettangolare di interi (massimo 100×200) preceduta dalle sue dimensioni (separate dal carattere `X`). Come esempio si consideri il seguente file.

```
5X6
1  3  4 -8  9 -5
0  0  0  0  0  0
3 -2  5 -2  0  0
2  3  3  3  0 -1
0 -1  1 -1  1  1
```

Scrivere una funzione C che prenda il nome di un file siffatto e modifichi il file eliminando le righe che hanno come primo valore 0 ed invertendo l'ordine delle righe. Il numero di righe deve essere riscritto in modo che corrisponda alle righe della nuova matrice.

Nell'esempio il contenuto del file al termine dell'esecuzione deve essere il seguente.

```
3X6
2  3  3  3  0 -1
3 -2  5 -2  0  0
1  3  4 -8  9 -5
```

Suggerimento: per modificare il file è necessario aprirlo in lettura, leggerne il contenuto, chiuderlo e successivamente riaprirlo in scrittura.

Compito dell'8 aprile 2002 (soluzione a pagina 76)

Esercizio 1 (punti 16) Un file contiene un insieme di operazioni aritmetiche binarie (cioè con due operandi) fra numeri interi, ciascuna posta su di una singola linea del file, e rappresentate con la usuale notazione $\mathsf{operando}_1$ op $\mathsf{operando}_2$ (con op $\in \{+, -, *\}$). Tra gli operandi e l'operatore è presente un numero arbitrario di spazi.

Un esempio di file corrispondente al formato descritto è il seguente:

```
3 + 5
2  -    4
-5  *  12
3 +2
```

Si scriva una funzione C che prenda come parametri il nome di un file siffatto da utilizzarsi come input e restituisca la somma dei risultati di ciascuna operazione. Nell'esempio la funzione dovrà restituire il valore `-49` (pari a $8 + -2 + -60 + 5$).

Si scriva inoltre una funzione `main` che verifichi la funzione suddetta ricevendo il nome del file sulla riga di comando.

Esercizio 2 (punti 16) Due file contengono gli esiti di due parti di un esame universitario e sono formati da una sequenza di righe ciascuna delle quali contiene il dato relativo ad uno studente. Il formato di ciascuna riga è: Cognome Nome, Matricola, Voto, dove Matricola e Voto sono dei valori interi (il voto espresso in trentesimi), mentre Cognome e Nome delle stringhe (senza spazi).

I file contengono *esattamente* la stessa sequenza di studenti. Un esempio di due file secondo il formato descritto è il seguente:

```
Bianchi Giulia, 23891, 24
Grigi Filiberto, 52342, 27
De_Rossi Orazio, 34601, 18
Verdi Maria_Concetta, 12345, 25
```

```
Bianchi Giulia, 23891, 27
Grigi Filiberto, 52342, 15
De_Rossi Orazio, 34601, 16
Verdi Maria_Concetta, 12345, 29
```

Si scriva una funzione C che prenda come parametri il nome di due file aventi tale formato e il nome di un file di output, e scriva nel file di output gli studenti che abbiamo ottenuto un voto medio uguale o superiore a 18 (arrotondato per eccesso).

Gli studenti devono comparire nel file di output nel formato: Nome Cognome: Voto. Nell'esempio, il file di ouput sarà il seguente.

```
Giulia Bianchi: 26
Filiberto Grigi: 21
Maria_Concetta Verdi: 27
```

Compito del 20 giugno 2002 (soluzione a pagina 78)

Esercizio 1 (punti 16) Un file contiene la descrizione di un insieme di alberghi, uno per riga. Ogni descrizione contiene

- il nome dell'albergo (massimo 20 caratteri),
- le *stelle* (carattere `*`) dell'albergo racchiuse tra parentesi e seguite da una virgola,
- il numero di servizi presenti nell'albergo,
- ciascun servizio dell'albergo (massimo 30 caratteri).

Ciascuna delle suddette informazioni è priva di spazi bianchi all'interno ed è separata dalle altre informazione tramite uno spazio bianco.

Un esempio di file del formato descritto è il seguente:

```
Miramonti (****), 3 televisione garage aria_condizionata
Posta (****), 2 garage giardino
Ambasciatori (***), 5 televisione bagno_in_camera giardino ristorante frigo_bar
Olimpia (**), 0
Vecchio_Mulino (**), 3 ristorante garage giardino
```

Si scriva una funzione C che prenda come parametri (*i*) il nome di un file siffatto, (*ii*) il nome di un servizio s ed (*iii*) un intero n, e restituisca il numero di alberghi nel file che abbiano il servizio s ed un numero di stelle pari o superiore ad n.

Nel file dell'esempio, se il servizio è `garage` ed $n = 3$, la funzione deve restituire 2, pari al numero di alberghi con almeno 3 stelle che abbiano il garage (`Miramonti` e `Posta`).

Esercizio 2 (punti 16) Un quadrato magico di ordine n è una matrice quadrata $n \times n$ composta da tutti a soli i numeri interi da 1 a n^2 e tale che la somma degli elementi su ogni riga, ogni colonna ed ognuna delle diagonali principali sia la stessa.

```
8 3 4        3 8 4        1 5 9        5 5 5
1 5 9        9 5 1        8 3 4        5 5 5
6 7 2        6 7 2        6 7 2        5 5 5
```

La prima matrice è un quadrato magico di ordine 3. La seconda non lo è poiché la somma degli elementi sulls prima colonna (18) è diversa dalla somma degli elementi su ognuna delle righe (15). La terza matrice non lo è poiché la somma degli elementi su una delle diagonali principali (6) è diversa dalla somma degli elementi su ognuna delle righe e colonne (15). La quarta non è un quadrato magico poiché, nonostante la somma degli elementi su ognuna delle righe, delle colonne e delle diagonali principali sia la stessa, non contiene tutti i numeri da 1 a 9.

Scrivere la funzione C `QuadratoMagico()` avente la seguente dichiarazione

```
int QuadratoMagico(int mat[][N]);
```

che prenda come parametro la matrice `mat` di dimensione `N`, con `N` costante definita a priori, e restituisca `1` se la matrice è un quadrato magico e `0` altrimenti.

Si consideri disponibile (già scritta) la funzione

```
int Presente(int m[][N], int val);
```

che restituisce `1` se `val` è presente nella matrice `m`, restituisce `0` altrimenti.

Suggerimento: La somma di ciascuna riga, colonna e diagonale di un quadrato magico di ordine n è pari a $n(n^2 + 1)/2$.

Compito del 15 luglio 2002 **(soluzione a pagina 81)**

Esercizio 1 (punti 16) Si scriva una funzione che prenda come parametri due stringhe e copi la prima stringa nella seconda, mettendo prima tutte le vocali nello stesso ordine in cui compaiono, poi tutte le consonanti, anch'esse in ordine, poi tutti gli altri caratteri (sempre in ordine).

Come esempi si considerino le stringhe qui sotto a sinistra, che devono essere copiate nella seconda stringa trasformandole nelle stringhe a destra:

```
  Una Stringa      →      UaianStrg
a + B + E = 4      →      aEB + + = 4
    2 + 2 = 5      →      2 + 2 = 5
```

Possono risultare utili le funzioni di libreria standard `isalpha`, che verifica se un carattere è alfabetico, e `tolower`, che, dato un carattere alfabetico maiuscolo, restituisce il corrispondente carattere minuscolo (dato qualsiasi altro carattere restituisce il carattere stesso).

Esercizio 2 (punti 16) Si scriva una funzione che prenda come parametro il nome di un file e restituisca, nel modo che si ritiene più opportuno, i seguenti quattro valori:

1. il numero totale di caratteri presenti nel file (inclusi spazi bianchi e ritorno a capo)
2. il numero di parole totali presenti nel file, dove per parola si intende qualsiasi sequenza non nulla di caratteri saparata da spazi bianchi o ritorno a capo;
3. il numero di righe totali del file.
4. la lunghezza della parola più lunga presente nel file

Si assuma che non ci siano mai spazi bianchi prima del carattere di ritorno a capo, e che anche l'ultima riga sia sempre terminata dal carattere di ritorno a capo. È invece possibile la presenza di righe vuote (che devono essere contate) e di sequenze di lunghezza maggiore di 1 di spazi bianchi.

Ad esempio se in file è il seguente

```
Ognuno sta solo sul cuor della terra,
trafitto da un raggio di sole:

ed  è  subito  sera.
```

la funzione deve restituire i valori $\langle 91, 17, 4, 8\rangle$

Compito del 9 settembre 2002 (soluzione a pagina 83)

Esercizio 1 (punti 16) Un file contiene un testo in cui possono comparire delle sequenze speciali composte da una lettera maiuscola racchiusa tra parentesi quadre. Come esempio si consideri il seguente file.

```
Oggi ho comprato [E] mele, [C] pere e [S] banane.
Ho pagato in tutto [P].[G] euro.
   Udine, [C]/[A]/2002
```

Un vettore di record a due campi contiene delle corrispondenze tra lettere maiuscole e numeri interi. Come esempio si consideri il seguente vettore.

A	9
C	17
G	78
B	12
P	6

Si scriva una funzione C che riceva come parametri

- il nome di un file di ingresso come descritto,
- il nome di un file di uscita,
- un vettore di corrispondenze come descritto e la sua dimensione,
- un intero k,

e che copi nel file di uscita il testo del file di ingresso sostituendo alle sequenze speciali i valori numerici corrispondenti alle lettere. Se una lettera non è presente nel vettore la sequenza di caratteri deve essere sostituita dal valore del parametro k.

Ad esempio, se il file di input ed il vettore sono quelli mostrati sopra e $k = 25$, il file di uscita è il seguente.

```
Oggi ho comprato 25 mele, 17 pere e 25 banane.
Ho pagato in tutto 6.78 euro.
   Udine, 17/9/2002
```

Si assuma che il file di ingresso non contenga i caratteri `[` e `]`, se non nelle sequenze speciali suddette.

Esercizio 2 (punti 16) Si scriva una funzione C che stampi sul video le date di tutte le domeniche di un anno. La funzione riceve come parametri l'anno di interesse ed il giorno della settimana con cui tale anno comincia (codificato da un intero nel modo seguente: 1 = Lunedì, 2 = Martedì, ..., 7 = Domenica).

Le date devono apparire nel formato `GG/MM/AAAA` ed essere separate da uno spazio bianco. Ad esempio se i parametri sono `2000` e `6` (Sabato), la funzione dovrà fornire il seguente output

```
2/1/2000 9/1/2000 16/1/2000 23/1/2000 ... 17/12/2000 24/12/2000 31/12/2000
```

Si consideri già disponobile la funzione `DataSuccessiva()` che abbia la seguente dichiarazione

```
struct Data DataSuccessiva(struct Data d);
```

e che restituisca la data successiva a quella del parametro `d`. Il tipo `struct Data` è definito come segue.

```
struct Data
{ int giorno;
  int mese;
  int anno;
};
```

Compito del 25 settembre 2002 (soluzione a pagina 86)

Esercizio 1 (punti 12) Si consideri un vettore di record aventi il seguente formato:

```
struct Autore
{ char nome[20];
  int dato;
};
```

Il contenuto del vettore è ordinato in senso decrescente in base al nome (l'ordinamento è lessicografico, cioè quello indotto dalla funzione C `strcmp`).

Un esempio di contenuto del vettore è il seguente:

Silone	3
Pirandello	49
Eco	25
D'Annunzio	32
Calvino	18

Si scriva una funzione C che accetti come parametri un vettore di record `v`, la lunghezza di `v` ed un singolo record `r` e sommi il dato di `r` al dato della locazione di `v` che abbia lo stesso nome di `r`.

La funzione deve inoltre restituire un valore intero, che varrà `1` nel caso in cui il nome di `r` sia presente in `v`, e `0` nel caso che questo sia assente (in tal caso `v` rimane inalterato).

Nel vettore dell'esempio precedente, se `r` è il record `Eco 14`, la funzione restituisce `1` e modifica il vettore come segue:

Silone	3
Pirandello	49
Eco	39
D'Annunzio	32
Calvino	18

Esercizio 2 (punti 20) È convenzione comune, nell'uso del sistema di messaggistica SMS dei telefoni cellulari, comprimere le parole effettuando delle sostituzioni che permettono una più veloce digitazione del testo. Per esempio, la sequenza `per` viene sostituita da `x`, `ch` da `k`, e così via.

Si scriva in linguaggio C la funzione

```
void ComprimiSMS(char sms_in[], char sms_out[], char nome_file[])
```

che riceva il testo del messaggio SMS da comprimere nella stringa `sms_in` e ne scriva la versione compressa nella stringa `sms_out`. Le sostituzioni da effettuare per la compressione dei messaggi sono descritte nel file il cui nome è passato come ultimo parametro alla funzione. Tale file contiene un massimo di 64 righe e ciascuna di esse rappresenta una sostituzione, nel formato `<stringa originale> <stringa compressa>`

Si assuma che né le stringhe originali né quelle compresse possano contenere spazi o altri separatori. Le sostituzioni devono essere effettuate anche se la stringa originale non si presenta da sola nel testo ma all'interno di una parola. Nella stessa parola possono avvenire più sostituzioni.

Come esempio si consideri il seguente file di sostituzioni

```
per x
ch  k
piu' +
```

Se il testo originale è `perche' non mi ami piu'?`, il testo compresso sarà `xke' non mi ami +?`.

Suggerimento 1: può essere utile sfruttare la funzione di libreria

```
int strncmp(const char *s1, const char *s2, int maxlen);
```

che funziona come la `strcmp`, ma invece di comparare le stringhe intere (cioè fino al terminatore) compara soltanto i primi `maxlen` caratteri.

Suggerimento 2: si noti che le funzioni sulle stringhe (`strcpy`, `strcmp`, `strncmp`, ...) possono agire anche su una parte di una stringa passando alla funzione non il nome della variabile stringa, ma il nome stesso sommato ad un intero. Ad esempio, se la stringa `s` ha valore `"Mario Rossi"`, l'istruzione `strcpy(t,s+4)` assegna a `t` il valore `"o Rossi"`.

Compito del 16 dicembre 2002 (soluzione a pagina 89)

Esercizio 1 (punti 16) Un file contiene una sequenza di lunghezza ignota di giocatori di basket (uno per riga). Di ogni giocatore sono memorizzati il cognome (massimo 20 caratteri, senza spazi), l'altezza in centimetri, la percentuale di realizzazioni e il numero di falli, quest'ultimo compreso tra 0 e 5, scritto a lettere e racchiuso tra parentesi quadre. Come esempio si consideri il seguente file.

```
Rossi   192   70.5    [tre falli]
Bianchi 202   65.2    [tre falli]
Neri    178   66.76   [due falli]
Viola   197   80.41   [cinque falli]
Gialli  205   85.25   [un fallo]
Neri    198   80.02   [zero falli]
```

Si scriva una funzione C che prenda come parametri il nome del file di ingresso nel formato suddetto ed una percentuale di realizzazione, e restituisca il numero medio di falli dei giocatori che abbiano una percentuale superiore a quella data e che siano alti almeno 190 cm.

Se nell'esempio la percentuale richiesta è 70.1%, la funzione deve restituire 2.25, ottenuto come media tra il numero di falli di `Rossi`, `Viola`, `Gialli` e `Neri`.

Esercizio 2 (punti 16) Si considerino delle stringhe composte dai soli caratteri `'0'` e `'1'` (oltre ovviamente al terminatore di stringa). Si scriva la funzione C che riceva come parametri due stringhe siffatte di uguale lunghezza e ne calcoli la somma in binario. La somma dovrà essere scritta nello stesso modo in un'altra stringa (anch'essa della stessa lunghezza) passata come terzo parametro alla funzione.

La funzione deve inoltre restituire un valore booleano che registra l'eventuale condizione di *overflow* (`0`: nessun *overflow*, `1`: c'è stato *overflow*).

Ad esempio, se le stringhe di ingresso sono `0111001` e `0011100`, la stringa in uscita sarà `1010101` ed il valore restituito sarà `0`. Se invece le stringhe in ingresso sono `10110011` e `11111001`, la stringa in uscita sarà `10101100` ed il valore restituito sarà `1`.

Compito del 19 marzo 2003 (soluzione a pagina 91)

Esercizio 1 (punti 15) Un file contiene i dati di un insieme di appuntamenti, organizzati uno per riga. Ciascuna riga contiene la data, l'ora, la descrizione (massimo 20 caratteri senza spazi) e il luogo (massimo 10 caratteri senza spazi) dell'appuntamento, separati da un numero arbitrario di spazi bianchi. Come esempio si consideri il seguente file.

```
20-3-2003 ore 12    Dentista                    Udine
22-3-2003 ore 23    Discoteca                   Portogruaro
```

```
19-3-2003 ore 15    Compito_Fondamenti          Rizzi
20-3-2003 ore 18    Partita_Pallavolo           Feletto
```

Si consideri inoltre il tipo di record `Orario`, per la memorizzazione di un istante di tempo (data e ora):

```
struct Orario
{
  int giorno;
  int mese;
  int anno;
  int ora;
};
```

Si scriva una funzione C che accetti come parametri il nome di un file siffatto, il nome di un file di output e un record `d` di tipo `Orario`. La funzione deve copiare dal file di input a quello di output ciascun appuntamento che sia successivo all'istante `d`, sostituendo le sequenze di spazi bianchi con un singolo spazio bianco.

Ad esempio, nel caso in cui i valori contenuti in `d` siano `20`, `3`, `2003` e `13` rispettivamente, il file di output avrebbe il seguente contenuto:

```
22-3-2003 ore 23 Discoteca Portogruaro
20-3-2003 ore 18 Partita_Pallavolo Feletto
```

Suggerimento: si definisca una funzione `ComparaOrari` che prenda come parametri due record di tipo `Orario` e restituisca `1` se il primo istante è successivo al secondo, `0` altrimenti.

Esercizio 2 (punti 15) Una matrice si dice *sparsa* se la maggior parte dei suoi elementi ha uno stesso unico valore, detto valore *dominante*. Ad esempio la seguente matrice di dimensione 6×4 è sparsa con valore dominante 3.

3	3	3	2
3	0	3	3
5	3	3	4
1	0	3	3
3	3	3	3
3	3	3	3

Si consideri un file che memorizza una matrice sparsa (di dimensione massima 100×100) nel seguente modo: la prima riga contiene il numero di righe, il numero di colonne e il valore dominante della matrice. Le righe successive contengono la riga, la colonna ed il valore di ciascun elemento diverso dal valore dominante. Ad esempio, il file corrispondente alla matrice precedente è quello riportato in figura 1.

Si scriva una funzione C che prenda come parametro il nome di un file contenente una matrice memorizzata come spiegato e scriva la stessa matrice nel file `matrice.dat` in forma estesa. Nell'esempio precedente, al termine dell'esecuzione il file `matrice.dat` dovrà avere il contenuto di figura 2.

```
6 4 3
0 3 2
1 1 0
2 0 5
2 3 4
3 0 1
3 1 0
```

FIGURA 1

```
3  3  3  2
3  0  3  3
5  3  3  4
1  0  3  3
3  3  3  3
3  3  3  3
```

FIGURA 2

Compito del 31 marzo 2003 (soluzione a pagina 93)

Esercizio 1 (punti 15) Un file *indice* contiene una sequenza di nome di file di dati (uno per riga). I nomi hanno tutti il formato `datiXX.txt`, dove `XX` sono due caratteri numerici. Come esempio si consideri il seguente file indice.

```
dati12.txt
dati05.txt
dati89.txt
dati44.txt
dati32.txt
```

I file di dati contengono ciascuno una sequenza di valori interi separati da spazi e/o ritorno a capo. Come esempio si consideri il seguente file

```
3 47 23   12
45  8
   74
```

Si scriva una funzione C che prenda come parametro il nome di un file indice ed un valore intero k e restituisca la somma dei valori presenti nei soli file di dati tali che k è *maggiore* dell'intero corrispondente ai caratteri `XX` nel nome.

Ad esempio, se il file indice è quello qui sopra e $k = 34$, la funzione deve restituire la somma dei valori presenti nei soli file `dati89.txt` e `dati44.txt`.

Esercizio 2 (punti 15) Si consideri un file contenente le informazioni sui voli in partenza da un determinato aeroporto avente il seguente formato:

```
FCO AZ 1252 12345
FCO AZ 1253 67
CDG AF 94 1234567
AMS KL 2348 12357
ZRH LX 239 1357
```

I primi tre caratteri del file costituiscono il codice dell'aeroporto di destinazione, i due caratteri successivi identificano la compagnia aerea ed il valore numerico costituisce il numero del volo. I dati suddetti sono separati da un singolo spazio. Segue (separata dai dati precedenti da un singolo spazio) una sequenza di caratteri numerici compresi fra 1 e 7, **non** separati da spazi, che indica in quali giorni quel volo è previsto (il numero 1 corrisponde a lunedì, il 2 a martedì, e così via fino al numero 7 che corrisponde a domenica).

Ad esempio, nel file precedente la riga `CDG AF 94 1234567` ha come destinazione l'aeroporto `CDG`, con il volo `AF` numero `94`, che viene effettuato tutti i giorni della settimana. Il volo `ZRH LX 239 1357`, invece, viene effettuato solo nei giorni di lunedì, mercoledì, venerdì e domenica.

Si scriva una funzione C che prenda come parametri

1. *una stringa* contenente il nome di un file siffatto,
2. *una stringa* di tre caratteri contenente l'aeroporto di destinazione `dest`,
3. *un valore intero* `giorno`, corrispondente ad un giorno della settimana secondo la codifica sopra descritta.

La funzione deve restituire un record che contenga i seguenti dati relativi al volo verso la destinazione richiesta nel giorno desiderato:

1. il codice della compagnia,
2. il numero del volo.

Qualora nel file non sia presente un volo verso la destinazione scelta per il giorno desiderato, la funzione deve restituire un record in cui il codice della compagnia è vuoto (cioè la locazione `0` contiene il carattere `\0`) e il numero del volo è `-1`. Si assuma, inoltre, che nel file non possano essere presenti più voli verso la destinazione scelta per il giorno desiderato.

Ad esempio, nel caso del file precedente, se `dest` è `FCO` e `giorno` è `6`, la funzione deve restituire nell'apposita `struct` i valori `AZ` e `1253`.

Compito del 30 giugno 2003 **(soluzione a pagina 95)**

Esercizio 1 (punti 15) Un file contiene una sequenza di lunghezza ignota (massimo 20) di risultati di partite di calcio (uno per riga). I risultati sono scritti nel formato illustrato dal seguente file di esempio.

```
Genoa - Ascoli          1-2
Ancona - Bari           1-2
Cosenza - Catania       3-1
Siena - Livorno         2-0
Palermo - Napoli        2-1
Triestina - Ternana     4-3
Messina - Venezia       1-1
Lecce - Hellas_Verona   1-1
```

I nomi delle squadre non contengono spazi e sono separati da uno spazio, un trattino (`-`) ed un altro spazio; i nomi e i gol segnati sono separati da uno o più spazi; i gol segnati sono separati tra loro dal trattino (senza spazi).

Si scriva una funzione C che prenda come parametro il nome di un file nel formato suddetto e *modifichi il file stesso* aggiungendo alla fine di ciascuna riga il simbolo del totocalcio corrispondente al risultato (`1` ha vinto la prima squadra, `2` ha vinto la seconda squadra, `X` pareggio, cioè le squadre hanno segnato lo stesso numero di gol).

Ad esempio, se il file è quello qui sopra la funzione deve modificarlo facendogli assumere il seguente contenuto.

```
Genoa - Ascoli  1-2 2
Ancona - Bari  1-2 2
Cosenza - Catania  3-1 1
Siena - Livorno  2-0 1
Palermo - Napoli  2-1 1
Triestina - Ternana  4-3 1
Messina - Venezia  1-1 X
Lecce - Hellas_Verona  1-1 X
```

Suggerimento: per poter scrivere sullo stesso file è consigliato copiare tutto il contenuto del file in una opportuna struttura dati in memoria centrale e successivamente ritrasferire i dati dalla memoria al file (a questo scopo è necessario aprire e chiudere il file due volta con modalità diverse).

Esercizio 2 (punti 15) Si consideri una matrice di caratteri 8×8 che contiene la posizione di una partita di scacchi. I pezzi bianchi sono rappresentati dalla loro iniziale minuscola: `r`, `d`, `a`, `c`, `t`, e `p` rispettivamente per re, donna, alfiere, cavallo, torre e pedone; i pezzi neri sono rappresentati dalla stessa iniziale maiuscola `R`, `D`, `A`, `C`, `T`, e `P`. Le caselle vuote sono rappresentate dallo spazio. Ad esempio la seguente matrice rappresenta la posizione iniziale.

t	c	a	r	d	a	c	t
p	p	p	p	p	p	p	p
P	P	P	P	P	P	P	P
T	C	A	R	D	A	C	T

Si vuole scrivere una funzione che valuti chi è in vantaggio nella posizione corrente dal punto di vista del materiale presente sulla scacchiera. A questo scopo si usi la seguente valutazione dei pezzi: pedone = 1, cavallo = 3, alfiere = 3, torre = 5, donna = 9, re = non ha valore.

Si scriva una funzione C (opportunamente modularizzata tramite funzioni ausiliarie) che prenda come parametro una matrice siffatta e restituisca:

- 1, se il bianco ha più materiale del nero
- 2, se il nero ha più materiale del bianco
- 0, se il materiale è pari
- -1, se la posizione sulla scacchiera non è legale.

Ad esempio, se viene passata la posizione iniziale, la funzione deve restituire 0, perché il materiale è pari (39 ciascuno: 5+3+3+9+3+3+5+1+1+1+1+1+1+1+1).

Una posizione non è legale solo nei seguenti tre casi:

- Un pedone si trova nella riga 0 oppure nella riga 7
- Uno dei due re non è presente sulla scacchiera
- Il numero di cavalli di un giocatore è maggiore di due.

Suggerimento: possono risultare utili le funzioni `islower(char)` e `isupper(char)` che restituiscono 1 se il parametro è un carattere alfabetico minuscolo o maiuscolo rispettivamente; e la funzione `tolower(char)` che restituisce il carattere alfabetico minuscolo corrispondente al parametro maiuscolo (e restituisce il parametro stesso negli altri casi).

Compito del 14 luglio 2003 **(soluzione a pagina 98)**

Esercizio 1 (punti 18) Si consideri un file di testo che descrive il tabellone di un gioco. Da ogni casella del tabellone (massimo 100 caselle) è possibile muoversi su altre tre caselle spostandosi rispettivamente a sinistra, dritto o a destra. Ogni riga del file rappresenta una caselle del tabellone ed è costituita da (a) un intero che identifica la casella, (b) una stringa (senza spazi) che costituisce la descrizione, e (c) le tre *scelte* (sinistra/dritto/destra) costituite da un intero, che indica la casella di arrivo, tra parentesi e separate da virgole.

Come esempio si consideri il seguente file.

```
1 Atrio (2,3,4)
2 Bancone (4,5,1)
3 Scala (4,6,2)
4 Porta (2,3,5)
5 Camera_da_letto (0,3,3)
6 Bagno (2,0,0)
```

Il valore `0` di una scelta indica che con quella scelta si termina il gioco.

Una mossa è identificata da una delle lettere `S` (sinistra), `C` (centrale o dritto) oppure `D` (destra), e, applicata ad una casella, porta nella casella corrispondente alla scelta fatta. Ad esempio, sequenza di mosse `SCCD` a partire dall'atrio, porta al bancone, poi alla camera da letto, poi alla scala ed infine nuovamente al bancone.

Si scriva una funzione C che prenda come parametri (*i*) il nome del file contenente il tabellone ed (*ii*) una stringa contenente una sequenza di mosse e, supponendo che si inizi dalla casella 1, restituisca il numero della casella finale di arrivo.

Nel caso la stringa (o anche una sua parte iniziale) porti alla terminazione del gioco, la funzione deve restituire il valore `-1`. Ad esempio, se il file è quello mostrato sopra e la stringa è `DDSCDS`, la funzione deve restituire `-1`, in quanto le prime tre mosse `DDS` portano alla terminazione del gioco.

Esercizio 2 (punti 12) Un file contiene del testo in cui, in vari punti, compare il prezzo (in euro, senza centesimi) di un qualche oggetto, e ciascun prezzo è racchiuso tra parentesi quadre. Come esempio si consideri il seguente file.

```
Le mele [3], le pere [7], e le banane [11], ma
anche il pane [2] e il latte [1]. Poi la carne [13].
```

Si scriva una funzione C che prenda come parametri il nome di un file siffatto, il nome di un file di uscita, ed un intero k, e scriva nel file di uscita lo stesso testo (mantenendo quindi anche spazi e ritorni a capo) in cui tutti i prezzi sono stati aumentati di k. Ad esempio se $k = 3$, il file di uscita dovrà essere il seguente:

```
Le mele [6], le pere [10], e le banane [14], ma
anche il pane [5] e il latte [4]. Poi la carne [16].
```

Si assuma che i caratteri '[' e ']' non appiano in alcuna altra parte del testo.

Suggerimento: Si utilizzi una lettura carattere per carattere del testo (utilizzando, ad esempio, `getc()`) e non una lettura di stringhe.

Compito del 2 settembre 2003 (soluzione a pagina 100)

Esercizio 1 (punti 15) Un file contiene una sequenza di date, una per riga, ordinate cronologicamente. Ciascuna data è scritta nel formato `GG/MM/AAAA`. Come esempio si consideri il seguente file.

```
23/6/1999
12/4/2001
16/4/2001
18/12/2003
20/12/2003
4/4/2004
```

Si scriva una funzione C che prenda come parametri il nome di un file siffatto ed una data di riferimento e restituisca la prima data presente nel file che sia posteriore alla data passata come parametro.

Nel caso che tutte le date siano anteriori al parametro, la funzione deve restituire il valore del parametro stesso.

Ad esempio, nel caso del file precedente e della data di riferimento `23/2/2003`, la funzione deve restituire la data `18/12/2003`. Se invece la data di riferimento è `4/11/2006`, la funzione deve restituire la data `4/11/2006`.

Esercizio 2 (punti 15) Si consideri un file contenente una sequenza di quotazioni di borsa con il seguente formato:

```
nome_società#val1#val2#val3...#valn$
```

Su ciascuna riga del file è presente il nome della società (massimo 20 caratteri senza spazi) a cui si riferiscono le quotazioni e i valori delle azioni (preceduto dal carattere `#`, senza spazi) in diversi istanti della giornata. Il numero di valori rilevati può variare da società a società, ma è compreso tra 1 e 100. In tutti i casi la lista è terminata dal carattere `$`.

Un esempio di file siffatto è il seguente:

```
Axis#12.5#11.8#13.7#11.9#10.7$
BiRom#5.2#4.8#4.9#5.0$
Comeco#24.3#24.4#24.6#24.3#24.2#24.5#24.2#24.3#24.3$
```

Si scriva una funzione C che prenda come parametri il nome di un file di input nel formato precedente ed il nome di un file di output, e scriva in quest'ultimo i dati relativi alle quotazioni minima, mediana e massima registrate per ciascuna società.

La mediana di una lista di valori è quel valore che ha un numero uguale di valori più grandi e più piccoli. Nel caso di numero di valori pari, la mediana è quel valore che ha un numero di valori più grandi maggiore di uno rispetto a quelli più piccoli.

Nel caso del file precedente, la funzione deve produrre il seguente file:

```
Axis 10.7 11.8 13.7
BiRom 4.8 4.9 5.2
Comeco 24.2 24.3 24.6
```

Si consideri già disponibile la funzione

```
void Ordina(float v[], int n);
```

che ordina un vettore di `n` elementi.

Compito del 16 settembre 2003 (soluzione a pagina 102)

Esercizio 1 (punti 17) Una sequenza di caratteri può essere rappresentata tramite un vettore di coppie, in cui il primo elemento è il codice ASCII del carattere ed il secondo è il numero di ripetizioni consecutive dello stesso.

Ad esempio, la sequenza `aaabaaaaccccc==;aaa` è rappresentata dal vettore ⟨(`'a'`,3), (`'b'`,1), (`'a'`,4), (`'c'`,5),(`'='`,2),(`';'`,1),(`'a'`,3)⟩.

Si supponga di voler espandere una sequenza siffatta (in cui non compaia mai il carattere `\n`) in un file di testo con righe tutte di lunghezza r, eventualmente ad eccezione dell'ultima.

Si scriva una funzione C che prenda come parametri:

- il vettore che descrive la sequenza e la lunghezza di tale vettore
- il nome di un file di ouptut
- la lunghezza r delle righe sul file di output

e scriva sul file di output la rappresentazione estesa della sequenza. Ad esempio, nel caso precedente con $r = 3$ il file di output deve essere il seguente:

```
aaa
baa
aac
ccc
c==
;aa
a
```

Si noti come l'ultima riga può anche essere più corta di r.

Esercizio 2 (punti 13) Una stringa contiene nome (massimo 15 caratteri) e cognome (massimo 20 caratteri) di una persona, senza spazi e con le iniziali maiuscole. La stringa può contenere indifferentemente prima il nome oppure prima il cognome. Come esempi si considerino le stringhe `"MarcoRossi"` e `"BianchiGiovanni"`. Non sono ammessi nomi o cognomi doppi.

Si scriva una funzione C che prenda come parametri una stringa siffatta ed una stringa di uscita, e scriva nella stringa di uscita prima il nome e poi il cognome, separati da una virgola ed uno spazio. Nei casi precendenti la stringa di uscita dovrà essere `"Marco, Rossi"` e `"Giovanni, Bianchi"`, rispettivamente.

Per poter risolvere l'esercizio è necessario distinguere il nome dal cognome. A questo scopo, si utilizzi una funzione (supposta già scritta e disponibile) chiamata `ProbNome` che prende come parametro una stringa e restituisce un valore reale tra 0 e 1, che indica la probabilità che quella stringa sia un nome di persona (e non un cognome). Si consideri come nome la stringa che ottiene dalla funzione `ProbNome` il valore di probabilità più alto (e come cognome l'altra stringa).

Ad esempio, se la stringa passata alla funzione principale è `"MarcoTiberio"`, e la funzione `ProbNome` restituisce rispettivamente 0.9 e 0.7 quando invocata con i valori del parametro `"Marco"` e `"Tiberio"`, allora si deduce che il nome è `"Marco"` (e il cognome `"Tiberio"`) e la stringa di uscita deve essere `"Marco, Tiberio"`.

Compito del 9 dicembre 2003 (soluzione a pagina 105)

Esercizio 1 (punti 15) Un file contiene un numero reale positivo scritto nel seguente modo. Ciascuna riga contiene il carattere `%` seguito da una cifra in notazione unaria, cioè con un numero di caratteri `1` pari al suo valore. Le cifre sono scritte dalla più significativa alla meno significativa. La parte intera e quella frazionaria sono separate da una riga vuota. Si può assumere che nel file non siano presenti spazi bianchi o altri caratteri.

Ad esempio, il seguente file contiene il numero 5049,23.

```
%11111
%
```

```
%1111
%111111111

%11
%111
```

Si scriva una funzione C che prenda come parametro il nome di un file siffatto e restituisca il numero letto.

Esercizio 2 (punti 10) Si consideri una matrice quadrata di caratteri A di dimensione $n \times n$ che contiene i risultati di un campionato di calcio a n squadre, con partite di andata e ritorno. Ciascuna squadra è identificata da un valore numerico compreso fra 0 e $n-1$.

Ciascun elemento della matrice A contiene il simbolo risultante della partita tra la squadra i (in casa) e la squadra j:

- A(i,j) = `'1'` se la partita si è conclusa con la vittoria di i;
- A(i,j) = `'2'` se la partita si è conclusa con la vittoria di j;
- A(i,j) = `'X'` se la partita si è conclusa con un pareggio.

Ovviamente gli elementi $A(i,i)$ sulla diagonale principale della matrice vanno ignorati, e possiamo assumere che essi valgano `'0'`.

Il numero di punti di ciascuna squadra è calcolato nel modo seguente: 3 punti per la vittora, 1 per il pareggio, 0 per la sconfitta.

Si scriva una funzione C che prende come parametri *(i)* una matrice quadrata secondo il formato descritto di dimensione fisica 100×100, *(ii)* il numero di squadre n (con $n \leq 100$) e *(iii)* un vettore di interi `v` di n elementi. La funzione deve modificare il vettore `v` inserendo in ciascun elemento `v[i]` il numero di punti della squadra `i`.

Ad esempio, nel caso della seguente matrice 4×4

```
0  1  1  2
2  0  1  2
X  1  0  1
1  2  2  0
```

il valore di `v` in uscita sarà {`10, 6, 10, 9`}.

Esercizio 3 (punti 5) Si scriva la funzione `main()` per l'esercizio 2 in modo che legga la matrice da tastiera e stampi i valori del vettore.

Compito del 18 marzo 2004 (soluzione a pagina 108)

Esercizio 1 (punti 15) Un file contiene un insieme di appuntamenti per la settimana, uno per riga. Ciascun appuntamento è costituito da una descrizione (30 caratteri al massimo, senza spazi), un giorno della settimana ed un orario, scritti come nell'esempio seguente.

```
Dentista  il Lunedi'  ore 10:05
Partita   il Martedi' ore 10:35
Teatro    la Domenica ore 16:20
Cinema    il Giovedi' ore 20:20
```

Si scriva una funzione C che prenda come parametri il nome di un file siffatto, il nome di un file di uscita ed una data `dl` e scriva nel file di uscita gli stessi appuntamenti del file di ingresso sostituendo il giorno della settimana con la data corrispondente, dove il parametro `dl` contiene la data del Lunedì della settimana. Le date sono rappresentate per mezzo del record `struct Data` a tre campi interi (`giorno`, `mese` e `anno`), e devono essere scritte sul file separando i valori con il carattere `'/'`.

Ad esempio, se `dl` ha il valore `29/3/2004` e il file di ingresso è quello dell'esempio, il file di uscita dovrà essere il seguente.

```
Dentista 29/3/2004 ore 10:05
Partita 30/3/2004 ore 10:35
Teatro 4/4/2004 ore 16:20
Cinema 1/4/2004 ore 20:20
```

Si considerino disponibili (già scritte) le funzioni

```
struct Data DataSuccessiva(struct Data d);
struct Data DataPrecedente(struct Data d);
```

che restituiscono rispettivamente la data successiva e le data precedente alla data `d`.

Esercizio 2 (punti 10) Si scriva una funzione C che prenda come parametri due stringhe `d` e `s`, e restituisca un valore intero. La funzione deve copiare il contenuto della stringa `s` nella stringa `d` sostituendo ciascun carattere numerico nella stringa `s` con il corrispondente numero scritto in lettere (cioè `1` diventa `uno`, `2` diventa `due` e così via). Inoltre, la funzione deve restituire il numero di operazioni di sostituzione che sono state effettuate.

Ad esempio, se la stringa `s` è la seguente:

```
2 comodini, 5 vasi cinesi ming, 4 tappeti persiani.
```

la funzione deve inserire nella stringa `d` il testo

```
due comodini, cinque vasi cinesi ming, quattro tappeti persiani.
```

e restituire il valore 3.

Si consideri disponibile (già scritta) la funzione

```
void TraduciNumero(char num[], int n);
```

che scrive nella stringa `num` il valore in lettere del numero `n` (con `n` compreso tra 0 e 9). Ad esempio, se `n = 5`, all'uscita della funzione `TraduciNumero` la stringa `num` avrà il valore `"cinque"`.

Si ricordi l'esistenza della funzione `int isdigit(char ch);` che restituisce `1` se `ch` è un carattere numerico, `0` altrimenti.

Esercizio 3 (punti 5) Si scriva la funzione `main()` per l'esercizio 2 che legga la stringa di ingresso da tastiera e stampi la stringa di uscita e il valore restituito dalla funzione su video. La stringa di ingresso può ovviamente contenere degli spazi ed è terminata da un punto (che fa parte della stringa stessa).

Compito del 1 aprile 2004 **(soluzione a pagina 110)**

Esercizio 1 (punti 15) Un file contiene le informazioni sui recapiti di una serie di persone, ciascuno su di una riga, secondo il formato dell'esempio seguente

```
Pietro Savorgnan di Brazza', Via Brazzaville, 33030 Brazzacco; UD
Giacomo Leopardi, Via Ginestra 123/A, 62019 Recanati; AN
Galileo Galilei, Campo dei miracoli 1, 56100 Pisa; PI
Carlo Goldoni, Calle Marco Polo 2A, 30100 Venezia; VE
```

in cui la sigla della provincia è preceduta da un punto e virgola ed uno spazio.

Si scriva una funzione C che prenda come parametri il nome di un file siffatto, una stringa `p` contenente una serie di sigle di provincia separate da spazi ed il nome di un file di output. La funzione dovrà scrivere nel file di output solo i recapiti di quelle persone che abitano in una delle province contenute nella stringa `p`.

Ad esempio, nel caso del file precedente e della stringa `p` con valore `"VE UD MI"`, la funzione deve produrre il file seguente:

```
Pietro Savorgnan di Brazza', Via Brazzaville, 33030 Brazzacco; UD
Carlo Goldoni, Calle Marco Polo 2A, 30100 Venezia; VE
```

Esercizio 2 (punti 15) Si consideri il seguente tipo per la memorizzazione dei risultati di una partita di tennis (al meglio dei tre set).

```
struct Partita
{
  char gioc1[21];
```

```
  char gioc2[21];
  int ris1[3];
  int ris2[3];
};
```

Tale struttura è composta da due stringhe contenenti i nominativi dei due giocatori (massino 20 caratteri) e da due vettori di interi, di tre elementi ciascuno contenenti i punti di ciascun set per il giocatore 1 e il giocatore 2, rispettivamente. Se una partita è finita in 2 set, la terza locazione di `ris1` e `ris2` conterrà il valore `0`.

Si scriva una funzione C che prenda come parametri un vettore di record di tipo `struct Partita`, la sua dimensione, ed una stringa contenente il nominativo di un giocatore e restituisca il numero di partite vinte dal giocatore specificato.

Ad esempio, nel caso del vettore seguente (in cui ciascuna riga corrisponde ad un elemento):

gioc1	gioc2	ris1			ris2		
Bianchi	Rossi	4	7	4	6	5	6
Gialli	Bianchi	6	6	0	3	2	0
Rossi	Gialli	6	2	7	2	6	6

e del nominativo `Bianchi`, la funzione deve restituire il valore 0 in quanto `Bianchi` ha perso sia la prima partita (4-6, 7-5, 4-6) sia la seconda (3-6, 2-6); mentre nel caso dello stesso vettore e del nominativo `Rossi` deve restituire il valore 2 avendo `Rossi` vinto due partite.

Suggerimento: Si scriva una funzione che prenda un parametro di tipo `struct Partita` e restituisca `1` oppure `2` a seconda che abbia vinto la partita il giocatore 1 oppure il giocatore 2.

Compito del 28 giugno 2004 **(soluzione a pagina 113)**

Esercizio 1 (punti 15) Un file contiene le informazioni su temperatura e situazione meteorologica di un insieme di città, una per riga, secondo il formato dell'esempio seguente

```
Roma, 30.2c nuvoloso
Udine, 24.6c variabile
Boston, 67.2f sereno
Rio de Janeiro, 23.6c neve abbondante
New York, 78.1f nebbia
```

in cui il nome della città (massimo 30 caratteri, anche con spazi) è seguito da una virgola, la temperatura è seguita da un carattere (`'c'` oppure `'f'`) che indica la scala (Celsius o Fahrenheit) ed è separata dalla situazione meteorologica da uno spazio.

Si scriva una funzione C che prenda come parametri il nome di un file siffatto, un carattere `t` (`'c'` oppure `'f'`) ed il nome di un file di output. La funzione dovrà scrivere nel file di output le stesse informazioni del file di input convertendo tutte le temperature nella scala specificata da `t`.

Ad esempio, nel caso del file precedente, se `t` è uguale a `'c'`, la funzione deve produrre il file qui sotto a sinistra, mentre se `t` è uguale a `'f'` la funzione deve produrre quello a destra.

```
Roma, 30.2c nuvoloso
Udine, 24.6c variabile
Boston, 19.5556c sereno
Rio de Janeiro, 23.6c neve abbondante
New York, 25.6111c nebbia
```

```
Roma, 86.36f nuvoloso
Udine, 76.28f variabile
Boston, 67.2f sereno
Rio de Janeiro, 74.48f neve abbondante
New York, 78.1f nebbia
```

Si considerino disponibili (già scritte) le funzioni `float C2F(float)` e `float F2C(float)` che convertono i valori da una scala all'altra.

Esercizio 2 (punti 15) Un file contiene un insieme di date nel formato `(aaaa/mm/gg)`, separate tra loro da un numero arbitrario di spazi e ritorni a capo. Come esempio si consideri il seguente file.

```
(2004/8/23)    (2004/9/21)
                  (2003/12/21)
   (2002/12/26)    (2004/4/2)
```

Si scriva una funzione C che prenda come parametro il nome di un file siffatto e restituisca la data massima (cioè più recente) presente nel file. Nel caso il file contenga una o più date non valide (ad es. `(2004/2/30)`), la funzione deve restituire la data convenzionale `(2000/1/1)`.

Dato il tipo record `struct Data` a tre campi interi, si considerino disponibili le funzioni

```
int ComparaDate(struct Data d1, struct Data d2);
int DataValida(struct Data d);
```

La prima restituisce `-1` se `d1` è precedente a `d2`, `0` se sono uguali e `1` se `d1` è successiva a `d2`. La seconda restituisce `1` se `d` è una data valida, `0` altrimenti.

Nel caso dell'esempio (tutte le date sono valide) la funzione deve restituire la data `(2004/9/21)`.

Compito del 8 luglio 2004 **(soluzione a pagina 116)**

Esercizio 1 (punti 15) Si consideri un file di testo, contenente delle letture dei sensori di un autovettura, con il seguente formato:
`orario_lettura distanza_percorsa velocità_istantanea`

Un esempio di file secondo tale formato è il seguente:

```
11:52 102.3Km 82.5Km/h
11:54 105.4Km 125.3Km/h
11:57 112.1Km 73.1Km/h
12:06 120.7Km 52.0Km/h
12:16 128.7Km 141.8Km/h
```

Si scriva una funzione C che prenda come parametri il nome di un file siffatto, una distanza `d` espressa in Km ed una velocità di soglia `v` espressa in Km/h e restituisca (nel modo che si ritiene più opportuno) l'orario della prima lettura del file che soddisfa entrambe le seguenti proprietà:

- la velocità istantanea è maggiore o uguale alla velocità di soglia `v`;
- la distanza percorsa è maggiore o uguale della soglia `d`.

Ad esempio, nel caso del file precedente e di distanza minina `d = 120` e della velocità di soglia `v = 90.0`, la funzione dovrebbe restituire il valore `12:16`.

Nel caso che nessuna lettura soddisfi entrambe le proprietà, la funzione deve restituire l'orario `00:00`.

Esercizio 2 (punti 15) Una stringa contiene i seguenti dati personali di una persona:

- nome e cognome, al massimo 30 caratteri anche con spazi;
- sesso, un carattere `F` o `M`;
- anno di nascita;
- luogo di nascita, al massimo 30 caratteri anche con spazi.

I dati sono separati tra loro da una virgola ed uno spazio bianco. Come esempi si considerino le seguenti stringhe:

- `"Marco Rossi Mori, M, 1978, Udine"`
- `"Maria Rosa Verdi, F, 1967, S. Giorgio"`

Si scriva una funzione che prenda come parametro una stringa siffatta e restituisca un record di un tipo opportunamente definito contenete tutte le informazioni della stringa una per ogni campo del record.

Compito del 3 settembre 2004 (soluzione a pagina 118)

Esercizio 1 (punti 15) Un dato linguaggio di programmazione prevede l'inserimento di commenti nel codice sorgente racchiudendoli tra due occorrenze consecutive del carattere `<` ed il carattere `|`. Come esempio si consideri il seguente file.

```
<< questo programma calcola ...
   e poi esegue .... |
start: xor  r0 r0     << questa istruzione ... |
       mv   r1 < r2   << il contenuto di r1 viene ... |

loop:  ldbr r3 r2     << copia in R3 l'i-esimo carattere della stringa |
       jmpz end       << se vale zero ('\0') la stringa e' finita |
       inc  r0        << incrementa il contatore |
       jmp  loop      << ripete |
end:   ret
```

Si scriva una funzione C che riceva come parametri il nome del file sorgente contenente i commenti ed il nome di un file di uscita, e scriva sul file di uscita soltanto tutte le istruzioni e non i commenti. Il file risultante dall'esempio sarà:

```
start: xor  r0 r0
       mv   r1 < r2

loop:  ldbr r3 r2
       jmpz end
       inc  r0
       jmp  loop
end:   ret
```

Ovviamente è necessario considerare il caso in cui il carattere `<` compaia *singolarmente* all'interno del programma (o di un commento), e questo non va interpretato come inizio commento.

Esercizio 2 (punti 15) Si considerino dei vettore di record aventi il seguente formato:

```
struct Giocatore
{ char nome[20];
  int punti;
};
```

Due esempi di un vettore di questo tipo sono i seguenti:

Marco	3
Paolo	49
Giulia	25
Marta	32
Irene	18

Marta	11
Paolo	43
Luca	25
Giovanni	32

Si scriva una funzione C che accetti come parametri due vettori, con le relative lunghezze, ed un terzo vettore, e scriva nel terzo vettore (in ordine qualsiasi) la *somma* dei punti di ciascun giocatore. La funzione deve inoltre restituire la lunghezza del vettore somma.

Ad esempio, se i primi due vettori sono quelli dell'esempio precedente, il vettore somma sarà il seguente

Marco	3
Paolo	92
Giulia	25
Marta	43
Irene	18
Luca	25
Giovanni	32

e il valore restituito deve essere 7.

Suggerimento: Si modularizzi la funzione scrivendo una funzione ausiliaria che prenda come parametri un giocatore `g`, un vettore di giocatori `v` e la sua lunghezza `n` e restituisca la locazione di `v` in cui compare `g`, e restituisca `-1` se `g` non è presente in `v`.

Compito del 14 settembre 2004 **(soluzione a pagina 120)**

Esercizio 1 (punti 10) Un programma per gestire una piccola biblioteca rappresenta i volumi con un vettore di strutture i cui elementi base sono definiti come segue:

```
struct Volume
{
  char titolo[65];
  char autori[65];
  char collocazione[20];
  int in_prestito;
  struct Data fine_prestito;
};
```

dove `struct Data` è l'usuale tipo record visto a lezione per la memorizzazione delle date e il campo `in_prestito` vale `1` se il libro è attualmente in prestito e `0` altrimenti.

Si scriva una funzione in linguaggio C che riceva come parametri il vettore dei volumi `v`, la sua lunghezza `n`, la data odierna `d` ed un vettore di volumi `r` e restituisca un valore intero. La funzione deve copiare nel vettore di volumi `r` i record contenuti nel vettore `v` la cui data di `fine_prestito` è precedente alla data odierna `d` e restituire la dimensione finale del vettore `r`.

Si assuma già disponibile la funzione `int ComparaDate(struct Data d1, structData d2)`, vista a lezione, che restituisce `-1` se la data `d1` è precedente a `d2`, `0` se le due date sono uguali e `1` se `d1` è successiva a `d2`.

Esercizio 2 (punti 10) Si scriva una funzione C che prenda come parametri il nome di un file di input ed un vettore di volumi e legga il contenuto del vettore dei volumi dell'esercizio precedente da un file contenente i dati dei libri. La funzione deve inoltre restituire il numero di volumi letti dal file.

Il formato del file è il seguente:

$$\langle\ \textit{titolo}\ \rangle\,|\,\langle\ \textit{autori}\ \rangle\,|\,\langle\ \textit{collocazione}\ \rangle\,|\,\langle\ \textit{in prestito}\ \rangle\,|\,\langle\ \textit{data fine prestito}\ \rangle$$

In cui i campi $\langle$ *titolo* $\rangle$ e $\langle$ *autori* $\rangle$ sono delle stringhe che possono contenere degli spazi al loro interno, $\langle$ *collocazione* $\rangle$ è una stringa priva di spazi, $\langle$ *in prestito* $\rangle$ è un valore intero e $\langle$ *data fine prestito* $\rangle$ è una data espressa come stringa nell'usuale formato *GG-MM-AAAA*.

Un esempio di file siffatto è il seguente:

```
Linguaggio C|B. Kernighan, D. Ritchie|A.123|1|13-09-2004
Database Systems|Elmasri, Navathe|B.231|0|20-01-2003
```

Esercizio 3 (punti 10) Si scriva una funzione in linguaggio C che prenda come parametri il nome di un file di testo contenente alcune frasi (senza punteggiatura) e il nome di un file di output e scriva nel file di output solo le parole del file che contengono lettere doppie.

Ad esempio, nel caso del seguente file di testo dato in input:

```
In Frioli Paese Quantunque Freddo Lieto Di Belle Montagne Di Piu' Fiumi E Di Chiare Fontane E' Una
Terra Chiamata Udine Nella Quale Fu Gia' Una Bella E Nobile Donna Chiamata Madonna Dianora E Moglie
D'un Gran Ricco Uomo Nominato Gilberto Assai Piacevole E Di Buona Aria.
```

la funzione deve scrivere nel file di output il contenuto seguente:

```
freddo belle terra nella bella donna madonna ricco assai
```

Compito del 9 dicembre 2004 (soluzione a pagina 123)

Esercizio 1 (punti 16) Un file contiene un insieme di operazioni effettuate su un'apparecchiatura. Ogni riga del file (minimo 2 righe) corrisponde ad una operazione, ed è composta dal nome (massimo 20 caratteri, senza spazi) e dalla data in cui l'operazione è stata effettuata. Le operazioni sul file sono in ordine cronologico crescente.

Come esempio si consideri il seguente file

```
Controllo_Testata 23/7/2004
Pulitura 15/8/2004
Lucidatura 25/8/2004
Lavaggio 12/9/2004
Controllo_Cilindro 17/9/2004
```

Si scriva una funzione C che riceva come parametri il nome di un file siffatto ed un intero positivo n, e verifichi se vi sono due operazioni sul file effettuate a meno di n giorni di distanza tra di loro.

Nel caso tali operazioni esistano, la funzione deve restituire `1` e deve scrivere il nome della prima operazione in una stringa passata come ulteriore parametro. Nel caso contrario, la funzione restituisce `0` ed il valore del parametro ulteriore non è significativo.

Ad esempio, nel file precedente se $n = 4$ la funzione deve restituire 0, se invece $n = 7$ la funzione deve restituire 1 e scrivere `Lavaggio` nel parametro stringa.

Si considerino disponibili il tipo `struct Data` a tre campi interi `giorno`, `mese` e `anno` e la funzione

```
int DistanzaTraDate(struct Data d1, struct Data d2);
```

che restituisce la distanza in giorni tra `d1` e `d2` (con `d1` minore di `d2`).

Esercizio 2 (punti 14) Un file contiene i voti ottenuti da un insieme di studenti, uno per riga. Ciascuna riga è composta dal nome, il cognome e il voto dello studente separati tra loro da una virgola ed uno spazio. Nome e cognome possono contenere anche degli spazi all'interno.

I voti possibili sono gli interi compresi tra 18 e 30, oltre a "30 e lode" e "insufficiente". Come esempio si consideri il seguente file.

```
Mario, Rossi, 28
Filippo Maria, Gialli, 23
Irene, De Verdi, 26
Paolo, Neri, insufficiente
Marta, Viola, 30 e lode
Maria Grazia, Bianchi, 27
Elena Sofia, De Neri, insufficiente
```

Si scriva una funzione C che riceva come parametro il nome di un file siffatto e restituisca la media dei voti degli studenti presenti nel file. Nel calcolo della media, gli studenti insufficienti vanno ignorati, mentre il voto "30 e lode" va considerato come 30.

Nel caso del file precedente la funzione restituisce 26.8, ottenuto come media dei valori 28, 23, 26, 30 e 27.

Compito del 17 marzo 2005 (soluzione a pagina 125)

Esercizio 1 (punti 15) Un file contiene i risultati di un insieme di prove sperimentali. A ciascuna prova corrisponde una riga del file, ed è caratterizzata da un codice di 3 caratteri alfanumerici, un risultato intero positivo ed il tempo in secondi necessario per eseguire la prova. Il formato del file è quello che si evince dal seguente esempio.

```
X01   123   34.5s
X03   121   33.5s
X06   119   33.4s
Y18   112   32.6s
ZK9   132   36.0s
```

Si scriva una funzione C che riceva come parametri il nome di un file siffatto contenente al massimo 100 righe e restituisca in una opportuna struttura dati il codice, il risultato e il tempo della prova con il risultato più alto tra quelle *veloci*. Una prova si dice veloce se il suo tempo di esecuzione è di al più un secondo superiore al tempo minimo di esecuzione di una prova.

Nel file dell'esempio la funzione deve restituire i valori `"X03"`, `121`, e `33.5`, in quanto `121` è il valore massimo tra le prove che hanno un tempo minore o uguale a `33.6s` (ottenuto sommando uno al tempo minimo `32.6`).

Esercizio 2 (punti 15) Un file contiene un insieme di righe che riportano delle operazioni (somma o sottrazione) su due variabili, nel formato mostrato dal seguente file (si noti che tra le variabili e l'operatore c'è un numero arbitrario di spazi bianchi).

```
alfa  +  beta
beta  -   gamma
gamma    +  delta
```

Si consideri inoltre la seguente definizione di tipo

```
struct Variabile
{
  char nome[21];
  int valore;
};
```

per memorizzare il valore di una variabile.

Si scriva una funzione C che prenda come parametri il nome di un file siffatto, il nome di un file di output ed un vettore di tipo `Variabile` (e la sua lunghezza).

La funzione deve scrivere nel file di output le stesse operazioni del file di ingresso aggiungendo il risultato preceduto dal carattere `'='` (con uno spazio bianco tra le variabili e l'operatore). Se una variabile non è contenuta nel vettore dei valori allora il suo valore è 0.

Ad esempio, per il file di ingresso precedente, se il vettore è il seguente

alfa	3
beta	4
gamma	21

il file di uscita dopo l'esecuzione della funzione avrà il seguente contenuto

```
alfa + beta = 7
beta - gamma = -17
gamma + delta = 21
```

Compito del 7 aprile 2005 **(soluzione a pagina 127)**

Esercizio 1 (punti 15) Un file contiene le informazioni sui recapiti di una serie di persone, ciascuno su di una riga, secondo il seguente formato:

nominativo; indirizzo; capecitta; siglaprovincia

in cui i dati relativi a ciascuna voce possono contenere degli spazi e sono separati da punti e virgola come nell'esempio seguente:

```
Pietro Savorgnan di Brazza'; Via del castello; 33030 Brazzacco; UD
Giacomo Leopardi; Via Ginestra 123/A; 62019 Recanati; AN
Galileo Galilei; Campo dei miracoli 1; 56100 Pisa; PI
Carlo Goldoni; Fondamenta Polonese 2A; 30100 Venezia; VE
Marco Polo; Calle Goldoni 3; 30100 Venezia; VE
```

Si scriva una funzione C che prende come parametri il nome di un file siffatto, una stringa `s` ed il nome di un file di output. La funzione dovrà scrivere nel file di output solo i recapiti di quelle persone per i quali la stringa `s` è contenuta come sottostringa nel campo indirizzo.

Ad esempio, nel caso del file precedente e della stringa `s = "lo"`, la funzione dovrebbe produrre il file seguente:

```
Pietro Savorgnan di Brazza'; Via del castello; 33030 Brazzacco; UD
Carlo Goldoni; Fondamenta Polonese 2A; 30100 Venezia; VE
```

Nota: risulta utile la funzione `strstr(char s[], char t[])` che restituisce il valore `NULL` se la stringa `t` *non* è presente nella stringa `s` e un valore diverso da `NULL` in caso contrario.

Esercizio 2 (punti 15) Data una matrice A di dimensione $m \times n$ (con n e n minori o uguali a 100) contenente degli interi è possibile definire un operatore di *smussatura* che trasforma la matrice in una nuova matrice A' assegnando a ciascun elemento a'_{ij} della matrice il valore medio dei suoi *vicini* nella matrice originale (arrotondato per mezzo della funzione `round` che riceve un `float` e restituisce l'intero che rappresenta il suo arrotondamento). Un *vicino* di un elemento a_{ij} è un elemento a_{kl} in cui l'indice k differisce al più di 1 dal valore i e/o l'indice l differisce al più di 1 dal valore j. Ad esempio nel caso della matrice A nella figura di sinistra riportata di seguito, i vicini dell'elemento $d_{2,2}$ sono quelli contrassegnati, e fra i vicini è compreso anche l'elemento stesso.

Gli elementi posti sui bordi della matrice (cioè sulla prima o ultima riga o colonna) hanno un numero inferiore di vicini che include solamente gli elementi i cui indici soddisfano la proprietà e si trovano all'interno della matrice. A titolo esemplificativo, sulla parte destra della figura sottostante sono riportati i vicini dell'elemento $a_{0,1}$.

$$A = \begin{pmatrix} 0 & 1 & -4 & 2 & 2 \\ 5 & 3 & 0 & 7 & -1 \\ 2 & -1 & 4 & 3 & 0 \\ 6 & 12 & -9 & 1 & 2 \end{pmatrix} \qquad A = \begin{pmatrix} 0 & 1 & -4 & 2 & 2 \\ 5 & 3 & 0 & 7 & -1 \\ 2 & -1 & 4 & 3 & 0 \\ 6 & 12 & -9 & 1 & 2 \end{pmatrix}$$

Si scriva una funzione C che prende come parametri una matrice `A`, le sue dimensioni `m` e `n` e una seconda matrice `B` delle stesse dimensioni. La funzione deve scrivere nella matrice `B` il valore ottenuto dalla smussatura di tutti gli elementi della matrice `A`. Ad esempio, nel caso della matrice A precedente il risultato sarà la matrice B con i seguenti elementi:

$$B = \begin{pmatrix} 2 & 1 & 2 & 1 & 3 \\ 2 & 1 & 2 & 1 & 2 \\ 5 & 2 & 2 & 1 & 2 \\ 5 & 2 & 2 & 0 & 2 \end{pmatrix}$$

Compito del 15 luglio 2005 **(soluzione a pagina 129)**

Esercizio 1 (punti 17) Il gioco del Sudoku si gioca su di una matrice $n^2 \times n^2$ divisa in n^2 riquadri di $n \times n$ elementi. Ciascun elemento della matrice è un numero intero compreso fra 1 e n^2 e *una soluzione* al gioco del Sudoku è una matrice nelle cui righe, colonne e riquadri, *non* compaiono ripetizioni dello stesso numero. Un esempio di una matrice valida per $n = 3$ è riportato di seguito sulla sinistra; i riquadri sono evidenziati dalle doppie linee. La matrice riportata sulla destra, invece, non è una soluzione corretta perché contiene alcune ripetizioni di riga, di colonna e di riquadro (il numero ripetuto è evidenziato in grassetto, le violazioni in grigio).

3	6	4	9	2	7	8	1	5
8	9	7	4	1	5	6	3	2
1	2	5	3	8	6	4	7	9
4	1	8	6	9	3	5	2	7
9	7	6	1	5	2	3	8	4
2	5	3	7	4	8	9	6	1
6	4	1	2	3	9	7	5	8
7	8	2	5	6	4	1	9	3
5	3	9	8	7	1	2	4	6

Soluzione valida al Sudoku

3	**6**	4	9	2	7	8	1	**6**
8	9	7	4	1	5	**6**	3	2
1	2	5	3	8	6	4	7	9
4	1	8	6	9	3	5	2	7
9	7	6	1	5	2	3	8	4
2	5	3	7	4	8	9	6	1
6	4	1	2	3	9	7	5	8
7	8	2	5	6	4	1	9	3
5	3	9	8	7	1	2	4	**6**

Soluzione **non** valida al Sudoku

Si scriva una funzione C che prenda come parametri il valore n (assumendo $n \leq 5$) e il nome di un file di input contenente una matrice "Sudoku" memorizzata per righe (in cui, cioè, ciascuna delle righe del file contiene gli elementi di una riga della matrice, separati da spazi) e restituisca il valore 1 se la matrice è una matrice Sudoku corretta e, in caso contrario, restituisca il valore 0.

Esercizio 2 (punti 13) Un record contiene le seguenti informazioni:

- una stringa, di lunghezza massima 20 caratteri, che rappresenta il nome di una sala convegni;
- un valore intero che rappresenta il numero di persone che la sala può contenere;
- un valore reale che rappresenta il prezzo per l'affitto della sala in euro.

Si definisca il tipo necessario per rappresentare tale record e si scriva una funzione C che prende come parametri un vettore di record siffatti, la sua lunghezza n ed un intero p. La funzione deve cercare la sala che contenga almeno p persone e il cui costo di affitto sia minimo. Se tale sala viene trovata la funzione restituisce il suo prezzo, altrimenti se nessuna sala ha almeno p posti, la funzione restituisce il valore -1.

Compito del 30 agosto 2005 **(soluzione a pagina 132)**

Esercizio 1 (punti 10) Si consideri una matrice quadrata di dimensione $n \times n$ (con $n \leq 10$) tale che tutti gli elementi di *bordo* contengono il valore 0, mentre gli elementi *interni* contengono valori interi positivi tutti diversi tra loro. Come esempio si consideri la seguente matrice 7×7.

0	0	0	0	0	0	0
0	10	24	32	18	33	0
0	21	40	27	28	47	0
0	22	61	82	45	14	0
0	17	23	58	53	23	0
0	26	10	13	29	90	0
0	0	0	0	0	0	0

Due elementi della matrice si definiscono *vicini* se i loro indici di riga e colonna differiscono per al più 1. Ogni elemento interno ha quindi esattamente 9 vicini (incluso se stesso).

Data una matrice ed un suo elemento interno, si definisce *miglior vicino* l'elemento di valore massimo tra tutti i suoi vicini.

Si scriva una funzione C che prenda come parametri una matrice siffatta, la sua dimensione n ed un elemento interno e restituisca il suo miglior vicino.

Per rappresentare gli elementi della matrice, sia il parametro che il valore restituito, si usi un record del seguente tipo.

```
struct Casella
{
  int riga;
  int colonna;
};
```

Esercizio 2 (punti 10) Si scriva una funzione C che prenda come parametri una matrice come definita nell'esercizio 1, la sua dimensione ed un elemento interno della matrice (di tipo `struct Casella`). La funzione deve restituire il massimo valore che si raggiunge dall'elemento dato con la sequenza di spostamenti verso il miglior vicino, fino ad arrivare ad un punto che è miglior vicino di se stesso.

Per la matrice precedente se il punto di partenza è (1,2) il valore restituito è 82, ottenuto dalla sequenza di spostamenti: (1,2):24 $\rightarrow$ (2,2):40 $\rightarrow$ (3,3):82. Se il punto di partenza è (3,5) il valore restituito è 90, ottenuto dalla sequenza (3,5):14 $\rightarrow$ (4,4):53 $\rightarrow$ (5,5):90. Infine se il punto di partenza è (2,5) il valore restituito è 47, in quanto questo valore è il miglior vicino di se stesso.

Suggerimento: Si utilizzi la funzione sviluppata per l'esercizio 1.

Esercizio 3 (punti 10) Si scriva la funzione `main()` per verificare la correttezza della funzione dell'esercizio 2. La funzione `main()` deve leggere dalla linea di comando il nome del file in cui è scritta

la matrice. Nel file è presente solo la parte interna della matrice, preceduta dalla sua dimensione. Ad esempio, la matrice 7×7 dell'esempio, sarà scritta su file nel seguente modo.

```
5
10 24 32 18 33
21 40 27 28 47
22 61 82 45 14
17 23 58 53 23
26 10 13 29 90
```

Compito del 20 settembre 2005 (soluzione a pagina 133)

Esercizio 1. (15 punti) Un file contiene i movimenti di un conto corrente bancario, ordinati per data, ciascuno su una riga secondo il seguente formato:

⟨data_movimento⟩ ⟨importo_movimento⟩ ⟨valuta⟩

dove la valuta può essere `Euro` oppure `Dollari`. Nel caso la valuta sia `Dollari` sulla riga è presente un ulteriore dato, corrispondente al cambio `Dollari/Euro` di quella giornata, riportato fra parentesi.

Un esempio di file secondo tale formato è riportato di seguito:

```
01/01/2005 +12000.00 Euro
23/02/2005 -100.50 Dollari (0.81)
05/03/2005 +523.10 Euro
12/04/2005 -43.22 Dollari (0.78)
```

Si scriva una funzione C che prenda come parametri il nome di un file siffatto e una data `d` (dell'usuale tipo `struct Data`) e restituisca il saldo del conto corrente espresso in `Euro` a quella data.

Ad esempio, nel caso del file precedente e della data `10/03/2005`, la funzione dovrebbe restituire il valore 12441.694 (ottenuto come somma dei primi tre movimenti, di cui quello espresso in Dollari è stato convertito in Euro).

Si assuma di avere a disposizione la funzione `int ComparaDate(struct Data d1, struct Data d2)` che restituisce il valore `-1` se la data `d1` viene prima di `d2`, `0` se le date sono uguali e `+1` nel caso rimanente.

Esercizio 2. (15 punti) Si scriva una funzione C che, data una matrice `A` di interi di dimensione $n \times m$ (con n ed m al massimo 20), ed un vettore `d`, riempia il vettore `d` con tutti e soli gli elementi *distinti* contenuti nella matrice e restituisca la dimensione di tale vettore.

Ad esempio nel caso della seguente matrice di interi:

$$A = \begin{pmatrix} 2 & 0 & 1 & 1 & 2 \\ 1 & 1 & 1 & 1 & 2 \\ 6 & 2 & 2 & 0 & 2 \\ 6 & 2 & 1 & 0 & 1 \end{pmatrix}$$

la funzione dovrebbe riempire il vettore `d` come segue:

$$d = (2, 0, 1, 6)$$

e restituire il valore `4`.

Nel riempimento del vettore `d` non è richiesto alcun particolare ordine, è sufficiente che, al termine della funzione, il vettore `d` contenga tutti e soli gli elementi distinti della matrice.

Compito del 12 dicembre 2005 (soluzione a pagina 135)

Esercizio 1 (punti 15) Un file contiene in ciascuna riga la descrizione di un appartamento formata dall'indirizzo (massimo 30 caratteri) e dalla sequenza dei locali che lo compongono. L'indirizzo è terminato

dai due punti, mentre i locali (in numero qualsiasi, massimo 20 caratteri ciascuno, senza spazi) sono separati tra loro da virgole e terminati da un punto.

Come esempio si consideri il seguente file (si noti che sia i due punti che la virgola sono seguiti da uno spazio).

```
Via Roma 12: salotto, ripostiglio, cucina, bagno.
Via Milano 7c: cucina, salone, camera, cameretta.
Via Po 34/1: bagno, camera, cameretta, bagno, cucina, terrazza.
Via del campo 11: salone, camera, camera, bagno.
Piazza A. Rossi 3: salone, cucina, bagno, cucina.
Piazza Po 34/2: salotto.
```

Si scriva una funzione C che prenda come parametro il nome di un file siffatto e restituisca il numero di appartamenti che sono *abitabili*. Un appartamento è abitabile se ha almeno un bagno ed esattamente una cucina.

Nell'esempio, la funzione deve restituire 2 in quanto sono abitabili solo il primo e il terzo appartamento. Ad esempio, il quinto appartamento non è abitalile perché ha due cucine.

Esercizio 2 (punti 15) Un file contiene in ogni riga un valore numerico seguito da un fattore moltiplicativo espresso in lettere. Per esempio:

```
123.456 micro
0.12 mega
1000.1 kilo
7 pico
45.6 unita'
999 peta
```

Si scriva una funzione C che riceva come parametri il nome di un file siffatto e il nome di un file di uscita. Il programma dovrà scrivere nel file di uscita gli stessi valori del file di ingresso in notazione scientifica (quindi usando `%g` nel formato di output), senza più la parte testuale. Relativamente all'esempio sopra riportato, il file di uscita dovrà contenere:

```
0.000123456
120000
1.0001e+06
7e-12
45.6
9.99e+17
```

Ai fini della conversione dei fattori, il programma dovrà utilizzare un vettore di record, che verrà passato alla funzione come ulteriore parametro (insieme alla sua lunghezza), utilizzando la seguente definizione.

```
struct ValorePrefisso
{
   char nome[10];
   int valore;
};
```

Ciascun elemento del vettore conterrà una stringa con il nome del fattore moltiplicativo (`pico`, `mega`, `unita'`, ...) e il valore dell'esponente corrispondente (`-12`, `6`, `1`, ...).

Compito del 16 marzo 2006 (soluzione a pagina 137)

Esercizio 1 (punti 15) Un file contiene in ciascuna riga una data seguita da una stringa contenente gli appuntamenti del giorno (massimo 40 caratteri, racchiusi tra parentesi quadre). Quando la stringa è composta dal solo carattere `*` significa che in quella data non ci sono appuntamenti. Come esempio, si consideri il seguente file.

```
03/03/2006 *
08/03/2006 [Dentista, Palestra]
06/03/2006 *
12/03/2006 [Partita di calcetto, Discoteca]
18/03/2006 [Esame di Informatica]
15/03/2006 *
```

Si scriva una funzione C che prenda come parametri il nome di un file siffatto, una data (nel classico formato a tre campi interi), ed una stringa. La funzione deve scrivere nella stringa passata come terzo parametro gli appuntamenti della giornata; nel caso che non ci siamo appuntamenti, la funzione deve scrivere nella stringa la parola `Niente`.

Inoltre la funzione deve restituire `1` se la data è presente nel file e `0` altrimenti. Se la funzione restituisce `0`, il contenuto della stringa non è significativo.

Nell'esempio, nel caso in cui la data sia `18/3/2006`, la funzione deve restituire `1` e scrivere nella stringa `Esame di Informatica`, nel caso in cui sia`6/3/2006` deve restituire `1` e scrivere nella stringa `Niente`, infine nel caso in cui sia `14/3/2006` la funzione deve restituire `0` (e non scrivere nella stringa).

Si assuma già disponibile la funzione `int ComparaDate(struct Data d1, structData d2)`, che restituisce `-1` se la data `d1` è precedente a `d2`, `0` se le due date sono uguali e `1` se `d1` è successiva a `d2`.

Esercizio 2 (punti 15) Data una matrice di interi $n \times m$ chiamiamo *fascia di* k una sequenza di valori tutti uguali a k consecutivi sulla riga, che si può estendere anche su più righe. Ad esempio, nella seguente matrice, i valori in grassetto costituiscono una fascia di 4 di lunghezza 7.

3	5	2	**4**
4	**4**	**4**	**4**
4	**4**	3	0
0	4	4	6
7	3	1	2

Si scriva una funzione C che riceva come parametri una matrice di interi (di dimensione fisica 100×100) e le sue dimensioni logiche n ed m, ed un intero k. La funzione deve restituire la lunghezza della prima fascia di k nella matrice.

Compito del 30 marzo 2006

Esercizio 1 (punti 15) Un file contiene un elenco di libri, uno per riga, con l'indicazione dell'autore, del titolo, del genere, del prezzo di copertina e del numero di copie presenti a magazzino. Come esempio si consideri il seguente file.

```
E. Salgari, "Il Corsaro Nero", avventura, 17.50, 5
C. Harness, "Astronave senza tempo", fantascienza, 9.95, 1
E. Salgari, "Jolanda, la figlia del Corsaro Nero", avventura, 18.50, 4
J. R. Tolkien, "Il signore degli anelli", fantasy, 80.00, 3
I. Asimov, "Il crollo della Galassia Centrale", fantascienza, 14.90, 2
S. Singh, "L'ultimo teorema di Fermat", saggio, 16.53, 1
```

Si noti che all'interno del titolo di un libro possono comparire delle virgole, e che l'autore e il titolo possono contenere spazi. Si assuma invece che il genere non contenga spazi.

Si scriva una funzione C che riceva come parametri il nome di un file siffatto e il nome di un genere. La funzione deve creare un file di uscita di nome uguale al genere passato per parametro ed estensione .txt. Tale file deve contenere l'elenco dei libri di quel genere nel formato riportato nel seguente file `fantascienza.txt`, generato nel caso del file di input precedente e genere `fantascienza`.

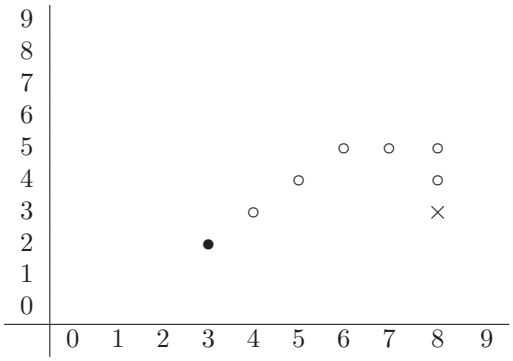
```
Autore: C. Harness
Titolo: Astronave senza tempo

Autore: I. Asimov
Titolo: Il crollo della Galassia Centrale
```

Esercizio 2 (punti 15) Si consideri un piano discreto di dimensione $N \times N$, contenente N^2 punti (con N costante fissata). Si consideri inoltre un file che rappresenti, per punti adiacenti, una traiettoria in tale piano specificando le coordinate del punto di partenza e la sequenza di movimenti che rappresenta la traiettoria. I movimenti sono limitati a un incremento o decremento di uno, o al mantenimento del valore, di ciascuna delle due coordinate. Tali movimenti sono indicati nel file da una coppia di simboli in cui '`+`' rappresenta un incremento di 1, '`-`' un decremento di 1 e '`0`' il mantenimento dell'ultimo valore per le coordinate. Per ciascuna coppia di simboli, il primo simbolo si riferisce alla riga ed il secondo alla colonna. Per esempio, se $N = 10$, il seguente file

```
2 3       ++ ++ ++ 0+ 0+ -0 -0
```

rappresenta la traiettoria mostrata nella figura qui sotto (con • punto iniziale e × punto finale).



Si scriva una funzione C che riceva come parametro il nome di un file contenente una traiettoria nel formato sopra descritto, restituisca nel modo che si ritiene più opportuno, le coordinate del punto di arrivo. Nel caso che la traiettoria porti fuori dal piano di N^2 punti, la funzione deve restituire i valori `-1, -1`.

Compito del 10 luglio 2006 **(soluzione a pagina 139)**

Esercizio 1 (punti 15) Un file contiene un elenco di risultati di esperimenti, uno per riga. Ogni esperimento è rappresentato da un codice di 3 caratteri seguito dai due punti e da una sequenza (di lunghezza positiva ignota) di misure a valore reale. Le misure sono separate tra loro da virgola e spazio e terminate dal punto e virgola. Come esempio si consideri il seguente file:

```
aa1: 3.4, 56.4, 23.78;
ab2: 5.6, 7.89, 9.45, 12.89;
ac2: -90.2, -92.3, -31, 1000;
xx3: 130, 800;
```

Si scriva una funzione C che riceva come parametro il nome di un file siffatto. La funzione deve restituire, in un'opportuna struttura dati, le seguenti informazioni:

- Il codice dell'esperimento che ha la somma delle misure massima, considerando però solo gli esperimenti con almeno tre misure
- Il valore della suddetta somma massima
- Il numero totale di esperimenti presenti nel file

Nel caso dell'esempio, la funzione deve restituire `ac2`, 786.5 e 4. Infatti, l'esperimento `xx3`, che ha somma maggiore, è composto da due sole misure. Gli altri hanno tutti somma minore di quella di `ac2`.

Nel caso di cui non esista alcun esperimento con almeno tre valori, la funzione deve restituire il codice convenzionale "`---`" e la somma non è significativa.

Esercizio 2 (punti 15) Un vettore di interi contiene il valore delle diverse monete di una valuta in ordine crescente di valore. Ad esempio, il seguente vettore {`1, 2, 5, 10, 20, 50, 100, 200`}, di 8 locazioni, rappresenta le monete dell'euro, espresse in centesimi.

Data una somma (rappresentata da un intero) vogliamo ottenere il suo valore in monete utilizzando sempre le monete più grandi possibili. Ad esempio, la somma 344 si ottiene con una moneta da 200, una da 100, due da 20 e due da 2 centesimi.

Si scriva una funzione C che riceva come parametri la somma, il vettore dei valori delle monete e la sua lunghezza, e scriva su un vettore di pari lunghezza (passato come ulteriore parametro) il numero di monete di ciascun valore necessarie per comporre la somma. Ad esempio, se il vettore dei valori è quello dato e la somma è 344, il vettore in uscita sarà {`0, 2, 0, 0, 2, 0, 1, 1`}.

Si assuma la presenza nel vettore della moneta di valore 1, in modo che la somma è sicuramente componibile con le monete del vettore.

Compito del 6 settembre 2006 **(soluzione a pagina 141)**

Esercizio 1 (punti 12) Si consideri un file contenente informazioni sulle date di nascita di un gruppo di persone espresse secondo il seguente formato:

```
Cognome persona, Nome persona, GG/MM/AAAA
```

Le diverse parti dell'informazione sono separate da virgole e su ciascuna riga del file è presente un singolo dato.

Un esempio di file siffatto è il seguente:

```
Neri, Maria Francesca, 24/03/1973
Bianchi, Giuseppe Maria, 12/03/1982
De Rossi, Andrea, 30/06/1965
Cremisi, Massimo, 12/12/1992
```

Si scriva una funzione C che prenda come parametro il nome di un file siffatto e un vettore di interi di 13 elementi . La funzione deve modificare il vettore di interi inserendo in ciascun elemento i del vettore il numero di persone nate nel mese i-esimo (la locazione 0 del vettore non è usata).

Ad esempio, nel caso del file dell'esempio precedente, al termine dell'esecuzione della funzione il vettore deve contenere i seguenti valori:

```
0 0 0 2 0 0 1 0 0 0 0 0 1
```

Esercizio 2 (punti 18) Si consideri una matrice quadrata di interi Q di dimensione $n \times n$ i cui elementi possono assumere i valori 0 e 1. Essa rappresenta le posizioni delle *regine* sulla scacchiera; in particolare se l'elemento Q_{ij} è uguale a uno significa che nella posizione (i, j) della scacchiera è posizionata una regina, mentre nel caso sia zero significa che la casella è vuota.

Ad esempio, la seguente matrice 5×5 rappresenta la configurazione della scacchiera riportata al centro.

1	0	0	0	0
0	0	0	1	0
0	1	0	0	0
0	0	0	0	1
0	0	1	0	0

	0	1	2	3	4
0	Q				
1				Q	
2		Q			
3					Q
4			Q		

	0	1	2	3	4
0	Q	↑		↗	
1	↖	↑	↗	Q	
2	←	**Q**	→	→	→
3	↙	↓	↘		Q
4		↓	Q	↘	

Le regine poste sulla scacchiera *attaccano* secondo la regola degli scacchi: ad esempio la regina in posizione (i, j) può attaccare tutti i pezzi presenti sulla colonna j, sulla riga i e sulle due diagonali che passano per (i, j). Nell'esempio in figura, sulla destra, sono indicate con delle frecce tutte le posizioni attaccate dalla regina in posizione $(2, 1)$ (indicata in grassetto). Come è possibile osservare, in questa configurazione non ci sono attacchi di una regina nei confronti delle altre.

Si scriva una funzione C che prende come parametri una matrice quadrata secondo il formato descritto in precedenza e la sua dimensione. La funzione deve verificare:

1. se il numero di regine posizionate sulla scacchiera è esattamente n;
2. quale sia il numero di regine che si attaccano.

La funzione deve restituire un valore intero, pari a -1, qualora il numero di regine posizionate sulla scacchiera non sia pari a n, e il numero di regine che si attaccano in caso contrario.

Compito del 22 settembre 2006 (soluzione a pagina 143)

Esercizio 1 (punti 15) Si consideri un file contenente informazioni su un insieme di attività di una settimana (una per riga, in numero qualsiasi), con il formato evinto dal seguente esempio.

```
Studio da Lunedi' ore 10 a Giovedi' ore 8
Partita da Martedi' ore 18 a Martedi' ore 21
Allenamenti da Sabato ore 16 a Domenica ore 23
Cinema da Giovedi' ore 20 a Giovedi' ore 22
```

Il nome dell'attività è una stringa (massimo 20 caratteri) senza spazi, e tutte le informazioni nel file sono separate tra loro da uno o più spazi. L'orario è sempre descritto solo da un intero rappresentante le ore (non ci sono i minuti).

Inoltre, tutte le attività iniziano e terminano nella stessa settimana, quindi non è possibile ad esempio che un'attività inizi il Venerdì e termini il Martedì, oppure che inizi Sabato alle 20 e termini Sabato alle 17.

Si scriva una funzione C che prenda come parametro il nome di un file siffatto e restituisca la lunghezza (in ore) dell'attività più lunga presente nel file.

Ad esempio, nel caso del file dell'esempio precedente, la funzione deve restituire 70 che è la durata della attività `Studio`, ottenuta come $24 \times 3 + (8 - 10)$.

Suggerimento: Si utilizzi una funzione ausiliaria che calcola l'intero (da 1 a 7) corrispondente al giorno della settimana passatogli come parametro di tipo stringa (Lunedì $\rightarrow$ 1, Martedì $\rightarrow$ 2, ..., Domenica $\rightarrow$ 7).

Esercizio 2 (punti 15) Si consideri la definizione del seguenti tipo record

```
#define LUNG_NOME 20
struct Partita
{
  char nome_squadra_casa[LUNG_NOME];
  char nome_squadra_ospite[LUNG_NOME];
  int gol_casa;
  int gol_ospiti;
};
```

che viene utilizzato per rappresentare il risultato di una partita di calcio.

Si scriva una funzione C che prende come parametri un vettore di partite utilizzando il record descritto (e la sua dimensione) e restituisca, in un'opportuna struttura dati, le seguenti informazioni:

1. il numero di partite vinte dalla squadra di casa
2. il numero di pareggi
3. il numero di partite vinte dalla squadra ospite
4. il nome della squadra che ha segnato più gol in una partita

Compito del 10 gennaio 2007

Esercizio 1 (punti 16) Si consideri un file contenente informazioni su un insieme di riferimenti bibliografici (uno per riga, in numero qualsiasi).

Ciascun riferimento è racchiuso tra parentesi quadre e contiene l'autore, il titolo, il nome della rivista su cui è pubblicato, il numero del fascicolo della rivista, le pagine del fascicolo e l'anno di pubblicazione.

Le informazione sono separate da virgole e organizzate nel formato evinto dal seguente esempio.

```
[Mario Rossi, Dialoghi sul mondo, Rivista di qualcosa, 4, 20-30, 2006]
[Franco Verdi, Critica della ragion pura, Rivista di qualcos'altro, 5, 12-45, 2005]
[Paola Gialli, La terra e altri pianeti, Rivista su tutto e niente, 2, 35-64, 2007]
```

Si consideri il seguente tipo di record per memorizzare alcune delle informazioni di un riferimento bibliografico.

```
struct Articolo
{
  char autore[30];
  int numero_pagine;
  int anno;
};
```

Si scriva una funzione C che prenda come parametri il nome di un file di riferimenti ed un vettore di record di tipo `struct Articolo` (di dimensione sufficiente).

La funzione deve riempire il vettore con i dati di tutti i riferimenti bibliografici presenti nel file. Inoltre la funzione deve restituire il numeri di riferimenti che sono stati inseriti nel vettore.

Ad esempio, se il file è quello dell'esempio il vettore in uscita deve contenere i seguenti valori, e la funzione deve restituire 3.

Mario Rossi	11	2006
Franco Verdi	34	2005
Giovanna Gialli	30	2007

Esercizio 2 (punti 14) Si scriva una funzione C che prenda come parametri due stringhe `d` e `s`. La funzione deve copiare il contenuto della stringa `s` nella stringa `d` sostituendo ciascuna sequenza di caratteri che indicano dei numeri (racchiusi tra parentesi quadrate e compresi tra `[uno]` e `[nove]`) presenti nella stringa `s` con il corrispondente valore numerico (cioè `[uno]` diventa `1`, `[due]` diventa `2` e così via).

Ad esempio, se la stringa `s` è la seguente:

```
[due] comodini, [cinque] vasi cinesi ming, [quattro] tappeti persiani.
```

la funzione deve inserire nella stringa `d` il testo

```
2 comodini, 5 vasi cinesi ming, 4 tappeti persiani.
```

Suggerimento: si consiglia di definire una funzione `int TraduciNumero(char num[]);` che traduce il valore in lettere corrispondente alla stringa `num` nel valore intero corrispondente.

Compito del 18 marzo 2008 **(soluzione a pagina 146)**

Esercizio 1 (punti 16) Si consideri un file contenente informazioni su un insieme di articoli presenti in magazzino (uno per riga, al massimo 50).

Per ciascun articolo sono memorizzati il numero di pezzi disponibili e il nome, separati da uno spazio. Il nome è lungo al massimo 20 caratteri e può contenere anche spazi. Come esempio si consideri il seguente file:

```
40 Viti M4
37 Viti da legno
20 Martelli
15 Chiavi inglesi
150 Chiodi
3 Pinze
```

Si scriva una funzione C che prenda come parametri il nome di un file siffatto, il nome di un file di output ed un valore reale che rappresenta una soglia percentuale.

La funzione deve scrivere nel file di output gli articoli del file di input nel formato: nome e percentuale di pezzi di quell'articolo rispetto al numero totale di pezzi in magazzino (separati dal carattere `':'`). La funzione deve però scrivere solo gli articoli che superano in percentuale il valore di soglia e deve anche restituire il numero di articoli nel file di uscita.

Ad esempio, se il file di ingresso è quello dell'esempio e la soglia è 10.2, il file di uscita sarà il seguente (in quanto `Martelli`, `Chiavi inglesi` e `Pinze` sono in quantità inferiore al 10.2% del totale) e il valore restituito sarà 3:

```
Viti M4: 15.09
Viti da legno: 13.96
Chiodi: 56.60
```

Esercizio 2 (punti 16) Si considerino le seguenti dichiarazioni di tipo di strutture dati:

```
struct Punto
{
  double X;
  double Y;
  double Z;
};
```

```
struct Triangolo
{
  struct Punto p1;
  struct Punto p2;
  struct Punto p3;
};
```

Si abbia un vettore di `struct Triangolo` che contiene la descrizione di n triangoli. Si scriva una funzione C che prenda come parametri tale vettore (e la sua dimensione n) e restituisca l'indice del vettore che contiene il triangolo di area massima.

Si ricorda che l'area A di un triangolo, note le lunghezze dei sui lati a, b e c, è calcolata dalla formula $A = s \cdot (s-a) \cdot (s-b) \cdot (s-c)$, dove s è il semiperimetro $(a+b+c)/2$.

Si ricorda inoltre che la lunghezza di un segmento nello spazio di estremi (x_1, y_1, z_1) e (x_2, y_2, z_2) è pari a $\sqrt{(x_1-x_2)^2 + (y_1-y_2)^2 + (z_1-z_2)^2}$.

Compito del 7 aprile 2008 **(soluzione a pagina 148)**

Esercizio 1 (punti 16) Si consideri un file che contiene una sequenza di cognomi (uno per riga, senza spazi) di candidati, tale che ogni elemento rappresenta un voto espresso a favore di quel candidato. Come esempio si consideri il seguente file

```
Verdi
Rossi
Bianchi
Rossi
Bianchi
Rossi
Rossi
De_Rossi
Verdi
Rossi
Verdi
```

Si assuma che il numero di voti sia qualsiasi, ma il numero di candidati votati sia al massimo 100.

Si scriva una funzione C che riceva come parametro un file siffatto e restituisca *attraverso un ulteriore parametro* il cognome del candidato che ha ricevuto più voti. La funzione deve inoltre restituire il numero di voti presi dal candidato suddetto.

Si assuma che tutti i candidati ricevano un numero diverso di voti, e quindi non si verifichino situazioni di parità.

Nell'esempio, la funzione deve restituire come candidato `Rossi` e come valore 5.

Esercizio 2 (punti 16) Si consideri disponibile la funzione `int FX(int v[], int n)` che verifica se la sequenza di valori interi contenuti nel vettore `v` di lunghezza `n` soddisfa una data proprietà matematica X.

Si scriva una funzione C che prenda come parametri una matrice quadrata (di dimensione massima 100), la sua dimensione effettiva e un intero positivo k. La funzione deve restituire 1 se esiste una sequenza di lunghezza k contenuta in una colonna della matrice che soddisfa la proprietà X, e restituire 0 se la sequenza non esiste.

Ad esempio, siano date le seguenti matrici di dimensione 8, sia $k = 5$ e si assuma $X =$ *"ordinata in senso crescente"*. La funzione deve restituire 1 per la matrice di sinistra, in quanto è presente una sequenza di 5 elementi nella colonna 2 (mostrati in grassetto) che soddisfa X e nessuna di lunghezza 6 o più; deve invece restituire 0 per la matrice di destra in quanto non sono presenti sequenze verticali di 5 elementi che soddisfino X.

$$\begin{pmatrix} 4 & 5 & 11 & 13 & 2 & 13 & 8 & 10 \\ 6 & 11 & 12 & 11 & 10 & 4 & 11 & 2 \\ 4 & 15 & 4 & 10 & 12 & 12 & 1 & 1 \\ 7 & 4 & \mathbf{2} & 13 & 7 & 12 & 9 & 11 \\ 2 & 8 & \mathbf{4} & 4 & 1 & 0 & 14 & 8 \\ 11 & 3 & \mathbf{11} & 5 & 15 & 14 & 7 & 3 \\ 14 & 13 & \mathbf{12} & 10 & 8 & 15 & 11 & 15 \\ 4 & 12 & \mathbf{15} & 11 & 12 & 5 & 6 & 14 \end{pmatrix} \qquad \begin{pmatrix} 0 & 7 & 8 & 12 & 1 & 3 & 4 & 5 \\ 10 & 11 & 12 & 4 & 4 & 3 & 0 & 8 \\ 9 & 13 & 15 & 1 & 13 & 14 & 8 & 9 \\ 2 & 10 & 11 & 0 & 6 & 3 & 13 & 6 \\ 10 & 5 & 3 & 11 & 9 & 7 & 0 & 3 \\ 2 & 13 & 7 & 6 & 0 & 7 & 15 & 9 \\ 5 & 14 & 11 & 2 & 12 & 3 & 11 & 14 \\ 13 & 6 & 14 & 4 & 10 & 11 & 10 & 4 \end{pmatrix}$$

Compito del 26 giugno 2008 **(soluzione a pagina 150)**

Esercizio 1 (punti 16) Si consideri un file che contiene una sequenza di riferimenti bibliografici (uno per riga). Ciascun riferimento è scritto con la seguente sintassi:

```
[numero] Autori; Titolo. MesePubblicazione, AnnoPubblicazione.
```

Il mese di pubblicazione è scritto a lettere e in inglese. Come esempio si consideri il seguente file. La numerazione nel file è crescente ma può essere discontinua.

```
[1] Stallman and Anderson; Highly-available information for robots. May, 2005.
[2] Sutherland and Morrison; Deploying IPv6 using collaborative methodologies. July, 1995.
[4] Suzuki; The effect of wireless archetypes on cryptoanalysis. September, 2001.
[6] Takahashi and Culler; Investigating scatter/gather I/O using flexible symmetries. October, 2002.
[10] Thompson; Studying multicast heuristics using atomic epistemologies. February, 2001.
[17] Thompson; Studying SCSI disks and Voice-over-IP using SibSaturn. March, 1994.
[21] Turing; A study of congestion control using sew. May, 2004.
```

Si scriva una funzione C che prenda come parametri il nome di un file siffatto ed un intero k e restituisca il mese di pubblicazione del riferimento corrispondente al numero k. Il mese deve essere restituito come intero (gennaio $\rightarrow$ 1, ..., dicembre $\rightarrow$ 12). Nel caso in cui il numero k non esista nel file, la funzione deve restituire 0.

Ad esempio, se passiamo alla funzione il nome del file dell'esempio e il valore 17, deve restituire 3 in quanto il riferimento 17 è pubblicato in marzo (*march*). Se invece gli passiamo il valore 19, deve restituire 0 in quanto non esiste il valore 19 nel file.

Esercizio 2 (punti 16) Si considerino i seguenti tipi per la memorizzazione dei risultati di una partita di pallavolo (al meglio dei cinque set).

```
struct Set
{
  int punti_casa;
  int punti_ospiti;
};
```

```
struct Partita
{
  char squadra_casa[21];
  char squadra_ospite[21];
  struct Set punti_set[5];
};
```

La struttura `Partita` è composta da due stringhe contenenti i nominativi delle squadre (massimo 20 caratteri) e da un vettore di 5 elementi contenente i punti di ciascun set (struttura `Set`). Se una partita è finita in meno di 5 set, le locazioni rimanenti conterranno tutti valori `0`.

Si scriva una funzione C che prenda come parametri un vettore di record di tipo `struct Partita`, la sua dimensione, ed una stringa contenente il nominativo di una squadra e restituisca il numero di partite vinte dalla squadra specificata.

Ad esempio, nel caso del vettore seguente (in cui ciascuna riga corrisponde ad un elemento):

squadra_casa	sduadra_ospite	1	2	3	4	5
Milano	Modena	23–25	25–20	20–25	25–19	9–15
Treviso	Milano	20–25	25–20	25–20	25–19	0–0
Modena	Treviso	25–12	25 –23	25–22	0–0	0–0

e del nominativo `Milano`, la funzione deve restituire il valore 0 in quanto la squadra `Milano` ha perso sia la prima partita (3–2) sia la seconda (3–1); mentre nel caso dello stesso vettore e del nominativo `Modena` deve restituire il valore 2 avendo la squadra `Modena` vinto due partite.

Compito del 11 settembre 2008 **(soluzione a pagina 153)**

Esercizio 1 (punti 16) Si considerino i dati relativi ad un insieme di alberghi. Un albergo è rappresentato dalla sua categoria (un intero), dal numero di stanze da cui è composto (massimo 100) e da un vettore di valori che descrivono ciascuna stanza. Una stanza è rappresentata dal numero di letti, dal il piano in cui è ubicata nell'albergo e da un valore booleano che indica se è libera o occupata.

Si scrivano le due definizioni di tipo necessarie per rappresentate una stanza e un albergo, rispettivamente.

Si scriva una funzione C che prenda come parametri:

1. un vettore di alberghi
2. la lunghezza del vettore
3. un valore intero che rappresenta una categoria desiderata
4. un valore intero che rappresenta un piano desiderato
5. un valore intero che rappresenta il numero di letti disiderato

La funzione deve restituire il numero totale di stanze libere tra tutti gli alberghi di categoria *uguale o superiore* a quella desiderata che siano ubicate *esattamente* al piano desiderato e che abbiamo un numero di letti *almeno pari* a quello desiderato.

Esercizio 2 (punti 16) Si consideri un file che contiene una matrice (massimo 50×50) di valori reali. La prima riga del file contiene il numero di righe e il numero di colonne della matrice. Le righe successive del file contengono le righe della matrice con i valori separati da uno o più spazi.

Come esempio si consideri il seguente file che contiene una matrice di 5 righe e 6 colonne.

```
5 6
1.2  2.33  1.4  0.34  0.032   0.02
1.2  4.55  1.1     1      0      0
  0     0  0.3   0.3    0.4  -10.3
  0     1    0   0.3   -0.4    5.5
  0     0  1.4  0.32   0.41  -1.31
```

Si scriva una funzione C che prenda come parametro il nome di un file contenente una matrice e restituisca, in un'opportuna struttura di dati, i quattro valori corrispondenti agli indici delle colonne di somma minima e massima e ai rispettivi valori delle somme (gli indici partono da 0).

Nell'esempio, la funzione deve restituire la quadrupla $\langle 5, -6.09, 1, 7.88 \rangle$ essendo la somma minima (pari a -6.09) quella della colonna 5 e la massima (7.88) quella della colonna 1.

Compito del 22 gennaio 2009 (soluzione a pagina 155)

Esercizio 1 (punti 15) Un file contiene le informazioni riguardanti degli studenti, uno per riga . In dettaglio, ciascuna riga è composta da un intero progressivo, la matricola (7 caratteri), il nome, il cognome, la data di nascita ed un eventuale commento.

Le informazioni sono separate tra loro da una virgola ed uno spazio. Si assuma che il nome e il cognome siano formati al massimo da 20 caratteri e possano contenere spazi bianchi (la matricola invece non contiene spazi). Come esempio, si consideri il seguente file.

```
1, I-29333, Mario, Rossi, 10/12/1982, trasferito da Padova
2, D-34211, Irene, De Neri, 1/1/1982, iscritta al II anno
3, S-23432, Pier Maria, Bianchi, 12/12/1985,
4, I-35211, Giuseppe, Verdi Chiari, 23/1/1989, in attesa di conferma
5, D-22222, Chiara, Blu, 24/6/1988,
```

Si scriva una funzione C che prenda come parametro il nome di un file siffatto e restituisca attraverso un ulteriore parametro il cognome dello studente più giovane. Nell'esempio, la funzione deve scrivere la stringa `Verdi Chiari`.

Si assumano già disponibili la definizione del tipo `struct Data` (nel classico formato a tre campi) e la funzione `int ComparaDate(struct Data d1, struct Data d2)`, che restituisce `-1` se la data `d1` è precedente a `d2`, `0` se le due date sono uguali e `1` se `d1` è successiva a `d2`.

Esercizio 2 (punti 15) Si scriva una funzione C che prenda come parametri una matrice quadrata `m` di interi e la sua dimensione `n` (con `n` al massimo 50). La funzione deve scrivere in due vettori `v` e `f` (passati come ulteriori parametri) gli elementi distinti della matrice (in `v`) e la loro rispettiva frequenza (in `f`). La funzione deve inoltre restituire la dimensione dei vettori `v` e `f`. Ad esempio nel caso della seguente matrice di interi:

$$m_{4\times4} = \begin{pmatrix} 11 & 11 & 2 & 11 \\ 11 & 11 & 11 & 7 \\ 11 & 11 & 11 & 2 \\ 7 & 7 & 2 & 7 \end{pmatrix}$$

la funzione dovrebbe riempire i vettori `v` ed `f` come segue: `v` $= (11, 2, 7)$ e `f` $= (9, 3, 4)$ e restituire il valore `3`. Infatti, gli elementi distinti presenti nella matrice sono 3 e precisamente 11, 2 e 7, ed hanno frequenza 9, 3 e 4, rispettivamente.

Suggerimento: Si scriva e si utilizzi la funzione ausiliaria

`int PosizioneElemento(int vet[], int dim, int el)`

che verifica se l'elemento `el` è già stato inserito nel vettore `vet` di dimensione `dim`. La funzione dovrà restituire la locazione di `el` nel vettore se questo è presente, altrimenti il valore convenzionale `-1`.

Compito del 23 febbraio 2009 (soluzione a pagina 157)

Esercizio 1 (punti 15) Si consideri un file che contiene uno per riga degli spostamenti di veicoli da una locazione ad un'altra. Ciascuna riga è composta da: il nome della locazione di partenza, la sequenza di caratteri `-->` e il nome della locazione di arrivo (separati da uno o più spazi bianchi). I nomi delle locazioni sono lunghi al massimo 10 caratteri e non contengono spazi bianchi.

Come esempio si consideri il seguente file:

```
Roma      -->   Milano
Milano    -->   Torino
Roma      -->   Genova
Roma      -->   Genova
```

Si consideri inoltre il seguente tipo di dato che memorizza per ogni locazione il nome e il numero di veicoli presenti.

```
struct Deposito
{
  char locazione[11];
  int num_veicoli;
};
```

Si scriva una funzione C che prenda come parametri il nome di un file il cui contenuto ha il formato descritto e un vettore di elementi di tipo `struct Deposito` (e la sua lunghezza). Il vettore descrive la situazione iniziale dei veicoli nelle varie locazioni. La funzione deve modificare il vettore in modo che corrisponda alla situazione dopo aver eseguito gli spostamenti presenti nel file.

Ad esempio, se il vettore contiene i valori qui sotto a sinistra e il file è quello dell'esempio, allora all'uscita della funzione il vettore dovrà avere i valori descritti a destra.

Roma	3
Milano	8
Torino	5
Genova	10

Roma	0
Milano	8
Torino	6
Genova	12

La funzione deve inoltre restituire il valore `1` nel caso in cui almeno una delle locazioni del file non sia presente nel vettore; deve restituire `2` nel caso in cui si verifichi una situazione (anche temporanea) in cui il numero di veicoli in una locazione è minore di 0. Deve restituire `0` in caso di funzionamento normale.

Esercizio 2 (punti 15) Si scriva una funzione C che prenda come parametri due stringhe, di cui la prima contiene una data nel formato `gg mese_in_lettere aaaa`. La funzione deve scrivere nella seconda stringa la data tradotta in inglese nel formato illustrato dai seguenti esempi (si noti che nel formato inglese non è presente lo `0` davanti ai giorni composti da una sola cifra).

Stringa in ingresso	Stringa in uscita
`12 gennaio 1982`	`January 12th, 1982`
`03 febbraio 1977`	`February 3rd, 1977`
`05 marzo 1977`	`March 5th, 1977`
`21 maggio 1977`	`May 21st, 1977`

Si consideri disponibile la funzione:

`void CalcolaSuffisso(char giorno[], char suffisso[]);`

che scrive nel parametro `suffisso` la stringa corretta (`"th"`, `"st"`, `"nd"` o `"rd"`) corrispondente al giorno scritto nella stringa `giorno`.

Per il compito in classe non è richiesto di scrivere esplicitamente tutti i 12 casi dei nomi dei mesi, ma è sufficienti scrivere 2-3 casi e lasciare in bianco gli altri.

Compito del 26 giugno 2009

Esercizio 1 (punti 15) Un file contiene delle informazioni riguardanti degli studenti, uno per riga. In dettaglio, ciascuna riga è composta da: la matricola, il nome, il cognome, la data e la città di nascita dello studente. Le informazioni sono separate tra loro da una virgola ed uno spazio. Si assuma che il nome, il cognome e la città di nascita siano formati al massimo da 30 caratteri e possano contenere spazi, la matricola invece è formata da esattamente 7 caratteri e non contiene spazi. Come esempio, si consideri il seguente file.

```
I-29333, Mario, Rossi, 10/12/1982, Udine
D-34211, Irene, De Neri, 1/1/1982, San Vito al Tagliamento
S-23432, Pier Maria, Bianchi, 12/12/1985, Tolmezzo
I-35211, Giuseppe, Verdi Chiari, 23/1/1989, San Daniele
D-22222, Chiara, Blu, 24/6/1988, Udine
```

Un ulteriore file contiene, una per riga, le verbalizzazioni di alcuni esami. Ogni riga contiene la materia (massimo 50 caratteri), la data, il voto e la matricola dello studente. I dati sono separati tra loro da uno o più spazi e nessuno contiene spazi al suo interno. Come esempio, si consideri il seguente file.

```
Fondamenti_Di_Informatica  26/06/2009  28    I-29333
Fondamenti_Di_Informatica  26/06/2009  24    D-34211
Fondamenti_Di_Informatica  26/06/2009  22    S-23432
Fondamenti_Di_Informatica  26/06/2009  19    I-35211
Basi_Di_Dati               22/06/2009  28    I-29333
Basi_Di_Dati               22/06/2009  24    D-34211
Basi_Di_Dati               22/06/2009  22    S-23432
Reti_Di_Calcolatori        16/06/2009  20    I-29333
```

Si scriva una funzione C che riceva 4 parametri: il nome di un file di studenti, il nome di un file di esami, il nome e il cognome di uno studente. La funzione deve restituire la media dei voti dello studente (si assuma che sia unico) avente il nome e il cognome passati come parametri.

Nel caso che lo studente non esista oppure che non abbia fatto esami, la funzione deve restituire il valore 0.

Ad esempio, se i due file sono quelli qui sopra, il nome è `Mario` e il cognome è `Rossi` la funzione deve restituire 25.33, ottenuto come media dei suoi voti (28, 28 e 20).

Esercizio 2 (punti 15) Si consideri una matrice (di dimensione massima 100×100 di valori 0 e 1 in cui gli 1 sono disposti in modo da formare dei rettangoli che rappresentano dei componenti su una scheda. I rettangoli non possono "toccarsi" tra loro neanche per un vertice. Come esempio si consideri la seguente matrice 10×14.

```
0 0 0 0 0 0 0 0 0 1 1 1 1 0
0 0 0 1 1 1 0 0 0 1 1 1 1 0
0 0 0 1 1 1 0 0 0 1 1 1 1 0
0 0 0 0 0 0 0 0 0 0 0 0 0 0
1 1 1 0 0 0 0 0 0 0 0 0 0 0
0 0 0 0 0 0 0 0 0 1 1 1 1 1
0 0 0 1 1 0 0 0 0 1 1 1 1 1
0 0 0 1 1 0 0 0 0 1 1 1 1 1
0 0 0 0 0 0 0 0 0 0 0 0 0 0
0 1 1 1 1 0 0 0 0 0 0 0 0 0
```

Si scriva una funzione C che riceva come parametri una matrice e le sue dimensioni e restituisca, nel modo che si ritiene opportuno, le coordinate del vertice in alto a sinistra del rettangolo di area massima (che si assume unico).

Nell'esempio, la funzione deve restituire la coppia $\langle 5, 9 \rangle$ che sono le coordinate (che partono da 0) del vertice in alto a sinistra del rettangolo di area 15.

Compito del 13 luglio 2009

Esercizio 1 (punti 16) Un file contiene delle informazioni riguardanti le date di alcuni eventi, uno per riga. In dettaglio, ciascuna riga è composta dal nome dell'evento e dalla sua collocazione temporale, separati da una virgola ed uno spazio. Il nome è lungo al massimo 50 caratteri e può contenere spazi al suo interno. La collocazione temporale può essere sia una data singola che un intervallo di date; in quest'ultimo caso i giorni di inizio e fine sono separati da un trattino. Il formato completo è quello che si evince dal seguente esempio. Si noti che il mese della data è scritto a lettere.

```
Fiera del libro, 3-5 giugno 2009
Partita di calcio, 28 settembre 2009
Sagra del carciofo, 7-12 luglio 2009
Concerto di Bartolomeo Pestalozzi, 28 settembre 2008
```

Si escluda la possibilità che un evento possa iniziare in un mese e finire in un altro mese.

Si scriva una funzione C che riceva come parametri due stringhe: il nome di un file di eventi e il nome di un evento. La funzione deve restituire la data iniziale (o la data unica) dell'evento la cui descrizione è passata per parametro. La data deve essere restituita nel classico formato a tre valori interi riportato qui sotto.

```
struct Data
{
  int giorno;
  int mese;
  int anno;
};
```

Ad esempio, se il file è quello mostrato sopra e l'evento è `"Sagra del carciofo"`, la funzione deve restituire una struttura con `giorno` uguale a 7, `mese` uguale a 7 e `anno` uguale a 2009. Nel caso in cui l'evento non sia presente nel file, la funzione deve restituire una data convenzionale in cui `giorno` uguale è a 1, `mese` uguale è a 1 e `anno` è uguale a 2000.

Esercizio 2 (punti 14) Una matrice si dice *sparsa* se oltre il 70% dei suoi elementi ha uno stesso unico valore, detto valore *dominante*. Ad esempio la seguente matrice di interi di dimensione 6×4 è sparsa con valore dominante 13.

13	13	13	2
13	0	13	13
5	13	13	4
1	0	13	13
13	13	13	13
13	13	13	13

Si scriva una funzione C che riceva come parametri una matrice di interi non negativi e le sue dimensioni (al massimo 100) e restituisca il valore dominante della matrice nel caso questa sia sparsa. Se invece la matrice non è sparsa la funzione deve restituire il valore `-1`.

Compito del 7 settembre 2009

Esercizio 1 (punti 16) Un file contiene delle informazioni riguardanti la media dei voti degli esami sostenuti da alcuni studenti, uno per riga. In dettaglio, ciascuna riga è composta da: matricola, numero totale di esami sostenuti e media dei voti. Le informazioni sono separate tra loro da uno spazio. Si assuma che la matricola è formata da esattamente 7 caratteri e non contiene spazi. Come esempio, si consideri il seguente file.

```
I-29333 7 23.28
D-34211 5 23.80
I-35211 11 27.27
D-22222 4 24.50
```

Un ulteriore file contiene, una per riga, le verbalizzazioni di alcuni esami. Ogni riga contiene la materia, la data, il voto e la matricola dello studente. I dati sono separati tra loro da uno o più spazi e nessuno contiene spazi al suo interno. Come esempio, si consideri il seguente file.

```
Fondamenti_Di_Informatica  26/06/2009  28    I-29333
Fondamenti_Di_Informatica  26/06/2009  30    D-34211
Fondamenti_Di_Informatica  26/06/2009  21    S-23432
Fondamenti_Di_Informatica  26/06/2009  19    I-35211
Basi_Di_Dati               22/06/2009  28    I-29333
Basi_Di_Dati               22/06/2009  24    D-34211
Basi_Di_Dati               22/06/2009  22    S-23432
Reti_Di_Calcolatori        16/06/2009  27    I-29333
```

Si scriva una funzione C che riceva 3 parametri: (*i*) il nome di un file di medie in ingresso, (*ii*) il nome di un file di verbalizzazioni e (*iii*) il nome di un file di medie in uscita.

La funzione deve scrivere nel file di uscita i dati del file in ingresso *aggiornati* tenendo conto dei voti degli esami nel file delle verbalizzazioni. Gli studenti presenti nel file delle verbalizzazioni che non compaiono nel file delle medie in ingresso devono essere aggiunti in fondo al file in uscita.

Ad esempio, nel caso in cui i due file siano i precedenti, il file di uscita sarà in seguente.

```
I-29333 10 24.60
D-34211 7 24.71
I-35211 12 26.58
D-22222 4 24.50
S-23432 2 21.50
```

Si assuma che i file delle medie possano contenere al massimo 100 studenti, non c'è invece limite al numero di righe del file delle verbalizzazioni. Si assuma infine che per il calcolo della media tutti gli esami abbiano uguale valore.

Esercizio 2 (punti 14) Un file contiene le rilevazioni della posizione di un veicolo nel piano in istanti di tempo successivi, a distanza di *un secondo* una dall'altra. Ciascuna riga contiene le due coordinate *in metri* rispetto ad una origine prefissata. Come esempio si consideri il seguente file.

```
10 10
11.3 11.6
12.6 13.4
13.5 15.3
15.4 17.6
14.4 20.7
16.5 23.8
20 30.6
```

Si scriva una funzione C che riceva come parametri il nome di un file contenente le rilevazioni ed un intero che rappresenta una velocità di soglia espressa in chilometri l'ora.

La funzione deve restituire le coordinate del punto in cui il veicolo supera la velocità di soglia per la prima volta. La velocità in un punto è calcolata sulla base della distanza nel piano rispetto al punto precedente nel file. Se il veicolo non supera mai tale velocità, allora la funzione deve restituire il valore convenzionale $(0.0, 0.0)$.

Ad esempio nel file precedente, se il valore di soglia è 10, la funzione deve restituire il punto $(15.4, 17.6)$, in quanto la velocità di soglia si raggiunge per la prima volta in quel punto. Infatti, la lunghezza del segmento dal punto $(13.5, 15.3)$ al punto $(15.4, 17.6)$ è pari a 2.98 metri. La velocità di 2.98 m/s corrisponde a $2.98 \times 3.6 = 10.74$ Km/h che supera il valore di soglia 10. Il valore 10 invece non viene superato nelle rilevazioni precedenti.

Compito del 26 gennaio 2010

Esercizio 1 (punti 16) Un file contiene informazioni su comune e data di nascita su un insieme di persone; le persone sono scritte una per riga e sono identificate da un numero intero. Le informazioni sono scritte nel formato che si evince dal seguente esempio.

```
1. Mario, Rossi; 13/2/1980 Aviano
2. Giuseppe, De Rossi; 12/4/1987 Chiopris-Viscone
3. Giulio Maria, De Verdi; 12/12/1986 Codroipo
4. Francesca, Rossi Gialli; 1/11/1991 Fogliano_Redipuglia
5. Giovanna, Neri; 21/3/1988 Camino_al_Tagliamento
```

Si noti che nome e cognome (massimo 20 caratteri ciascuno) possono contenere spazi bianchi, mentre il nome del comune (massimo 30 caratteri) non li può contenere.

Si consideri inoltre un vettore, del tipo `struct Locazione` definito come segue, che determina la provincia di appartenenza di ciascun comune. Come esempio si consideri il vettore a destra.

```
struct Locazione
{
  char comune[31];
  char provincia[3];
};
```

aula	capienza
A3	22
B1	30
DIS	50
C22	48

Si scriva una funzione C che riceva come parametri il nome di un file di persone nel formato illustrato, un vettore di `struct Locazione` e la sua dimensione, e una stringa contenente una provincia. La funzione deve stampare sul video nome, cognome e, tra parentesi, numero identificativo delle persone che sono nate in un comune della provincia passata come parametro.

Ad esempio, se la provincia passata come parametro è `"UD"`, il vettore e il file sono quelli precedenti, allora la funzione deve stampare i seguenti nomi

```
Giuseppe De Rossi (2)
Giulio Maria De Verdi (3)
Giovanna Neri (5)
```

Esercizio 2 (punti 14) Si considerino le seguenti definizioni di tipo che rappresentano rispettivamente un punto e un cerchio nel piano cartesiano.

```
struct PuntoNelPiano
{
  float x;
  float y;
};
```

```
struct Cerchio
{
  struct PuntoNelPiano posizione_centro;
  float raggio;
};
```

Si scriva una funzione C che prenda come parametri un punto p nel piano e un vettore di cerchi (e la sua dimensione) e restituisca il punto che rappresenta il centro del più piccolo cerchio del vettore che contiene il punto p. Nel caso che nessun cerchio appartenente al vettore contenga p la funzione deve restituire il punto $(0.0, 0.0)$, cioè l'origine degli assi.

Compito del 13 luglio 2010

Esercizio 1 (punti 15) Un file contiene una sequenza di interi positivi scritti in varie basi, uno per riga. Ciascuna riga contiene il numero n tra parentesi seguito da uno spazio e dalla sua base b, con $2 \leq b \leq 36$. I numeri in base $b > 10$ sono scritti utilizzando le prime $b - 10$ lettere maiuscole.

Come esempio si consideri il seguente file.

```
(23) 10
(EF0) 18
(2NG) 33
(1204) 5
```

Si scriva una funzione C che riceva come parametro il nome di un file siffatto e restituisca la somma dei numeri scritti nel file.

Nell'esempio, la funzione deve restituire il valore 7961, che è la somma di 23, 4806, 2953 e 179, corrispondenti rispettivamente a $(23)_{10}$, $(EF0)_{18}$, $(2NG)_{33}$, $(1204)_5$.

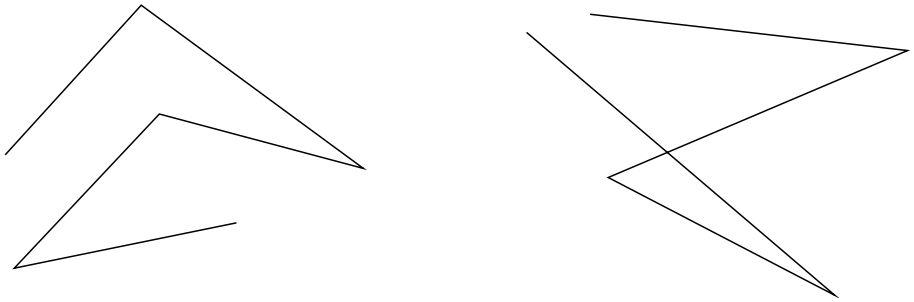
Esercizio 2 (punti 15) Si considerino le seguenti dichiarazioni di tipo, utilizzate per rappresentare rispettivamente un punto ed un segmento nel piano.

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

```
struct Segmento
{
  struct Punto p1;
  struct Punto p2;
};
```

Una linea spezzata (aperta) è rappresentata da una sequenza di punti $p_1, \ldots, p_n$, tali che ogni coppia di punti consecutivi p_i, p_{i+1} corrisponde ad un suo segmento.

Una linea spezzata è *non intrecciata* se ogni coppia di lati non consecutivi è priva di punti in comune. Ad esempio, la linea qui sotto a sinistra è non intrecciata, mentre quella a destra è intrecciata.



Si scriva una funzione C che riceva come parametri un vettore di punti (e la sua lunghezza) e restituisca 1 se la linea spezzata che il vettore rappresenta è non intrecciata, 0 se è intrecciata.

Si consideri disponibile (già realizzata) la funzione

```
int PuntiInComune(struct Segmento s1, struct Segmento s2);
```

che restituisce 1 se i segmenti `s1` e `s2` hanno punti in comune, 0 altrimenti.

Compito del 10 settembre 2010

Esercizio 1 (punti 15) Un file contiene le informazioni sui recapiti di una serie di persone, ciascuno su una riga, secondo il formato dell'esempio seguente

```
Pietro Savorgnan di Brazza', Via Brazzaville, 33040 Povoletto (UD)
Giacomo Leopardi, Via Ginestra 123/A, 62019 Recanati (AN)
Galileo Galilei, Campo dei miracoli 1, 56100 Pisa (PI)
Carlo Goldoni, Calle Marco Polo 2A, 30100 Venezia (VE)
Filippo Rossi, Via della quercia 72, 33040 Savorgnano del Torre (UD)
```

in cui il nome e l'indirizzo sono seguiti da una virgola e la sigla della provincia è racchiusa tra parentesi.

Inoltre, si consideri un vettore di tipo `struct Provincia` (definita qui sotto) che contiene tutte le corrispondenze tra il nome di una provincia e la sua sigla.

```
struct Provincia
{
  char Nome[31];
  char Sigla[3];
};
```

Si scriva una funzione C che prende come parametri il nome di un file nel formato specificato, il vettore delle province (e la sua lunghezza), il nome di una provincia e una stringa contenente un CAP. La funzione deve restituire il numero di persone presenti nel file la cui residenza corrisponde alla provincia e al CAP passati come argomenti alla funzione.

Per il file di esempio, se alla funzione vengono passati i valori `Udine` e `33040` il risultato restituito deve essere 2.

Esercizio 2 (punti 10) Si consideri un campionato di calcio a n squadre a girone unico (solo andata), in cui ciascuna squadra è identificata da un valore numerico compreso fra 0 e $n-1$.

Una matrice quadrata di interi A di dimensione $n \times n$ contiene i gol segnati dalle squadre. In dettaglio, l'elemento A_{ij} rappresenta il numero di gol segnati dalla squadra i alla squadra j, mentre A_{ji} è in numero di gol segnati dalla squadra j alla squadra i nella stessa partita.

Ovviamente gli elementi A_{ii} sulla diagonale principale della matrice vanno ignorati, e possiamo assumere che essi valgano `0`.

Il numero di punti di ciascuna squadra è calcolato nel modo seguente: 3 punti per la vittora, 1 per il pareggio, 0 per la sconfitta. Vince la partita la squadra che segna più gol.

Si scriva una funzione C che prende come parametri *(i)* una matrice quadrata secondo il formato descritto e di dimensione fisica 100×100, *(ii)* il numero di squadre n (con $n \leq 100$) e *(iii)* un vettore di interi `v` di n elementi. La funzione deve modificare il vettore `v` inserendo in ciascun elemento `v[i]` il numero totale di punti della squadra `i`.

Ad esempio, nel caso della seguente matrice 4×4

```
0  2  1  1
0  0  1  2
3  0  0  1
1  4  2  0
```

il valore di `v` in uscita sarà {`4, 3, 3, 7`}.

Esercizio 3 (punti 5) Si scriva la funzione `main()` per l'esercizio 2 in modo che legga la matrice da tastiera e stampi i valori del vettore.

Compito del 22 settembre 2010

Esercizio 1 (punti 15) Un file contiene una sequenza (di lunghezza massima 1000) di interi separati da una virgola e uno spazio. Come esempio, si consideri il seguente file

```
14, -3, 14, 7, 91, 12, 14, -3, 2, 14, 1, 4
```

Si scriva una funzione C che prenda come parametri i nomi di due file siffatti e restituisca `1` se essi contengono esattamente gli stessi valori, `0` altrimenti. Nel caso in cui un valore compaia più volte, il numero di occorrenze deve essere lo stesso nei due file. L'ordine con cui compaiono non è significativo.

Ad esempio, se i due file passati come parametri sono quello mostrato sopra e il seguente

```
-3, 14, 1, 7, 4, 14, 12, -2, 14, 91, 3, 1
```

la funzione deve restituire `0` in quanto, ad esempio, il valore `14` compare 4 volte nel primo file e 3 nel secondo.

Esercizio 2 (punti 15) Le immagini *bitmap* sono rappresentate mediante una matrice di punti (pixel) ciascuno dei quali è descritto da una tripletta di colori RGB, che contiene i valori di intensità, da 0 a 255, dei tre colori fondamentali: rosso, verde e blu. Si considerino dei file bitmap organizzati come segue: la prima riga contiene, nell'ordine, i due numeri interi N e M che rappresentano il numero di righe e il numero di colonne della matrice di pixel; poi il file contiene le triplette che rappresentano i pixel (un pixel per riga) a partire dalla prima riga, prima colonna (pixel in alto a sinistra nell'immagine), e procedendo con tutte le colonne della riga. Ogni pixel è rappresentato dai tre numeri nella sequenza: rosso, verde, blu.

Ad esempio, il file seguente rappresenta un'immagine 4×3 (4 righe e 3 colonne) con un punto in alto a sinistra bianco (i tre colori sono al massimo dell'intensità) e il resto della prima riga nero (tutti zeri); poi una riga grigia, e due righe con punti rossi $(255, 0, 0)$, verdi $(0, 255, 0)$, blu $(0, 0, 255)$, azzurri $(0, 255, 255)$, gialli $(255, 255, 0)$ e magenta $(255, 0, 255)$.

```
4 3
255 255 255
0   0   0
0   0   0
127 127 127
127 127 127
127 127 127
255 0   0
0   255 0
0   255 255
0   0   255
255 255 0
255 0   255
```

Si considerino inoltre le seguenti definizioni di tipo che rappresentano rispettivamente un pixel e una associazione tra un valore intero (che rappresenta l'altitudine di una zona di terreno) ed un pixel.

```
struct Pixel
{
  int r;
```

```
struct Colore
{
  struct Pixel pixel;
```

```
  int g;                    int altitudine;
  int b;                  };
};
```

Si scriva una funzione C che prenda come parametri:

- una matrice quadrata di interi `m` (che rappresenta l'altitudine di una zona di terreno) e la sua dimensione `n` (al massimo 100);
- il nome di un file di uscita;
- un vettore `v` di tipo `struct Colore` e la sua dimensione `k`.

La funzione deve scrivere sul file l'immagine bitmap corrispondente alla matrice `m` sostituendo a ciascun valore di altitudine la tripletta corrispondente nel vettore `v`. Nel caso un valore di altitudine non sia presente nel vettore, bisogna inserire nel file in pixel bianco.

Compito del 24 gennaio 2011

Esercizio 1 (punti 16) Un file contiene informazioni sui risultati di un corcorso da parte dei candidati. La prima riga contiene il numero di prove (al massimo 5), la seconda i punteggi massimi di ciascuna prova (separati da spazi) e quelle successive i punteggi dei candidati alle prove. I punteggi sono tutti valori interi.

Le prove *essenziali* sono contrassegnate con un asterisco dopo il punteggio massimo. Il formato esatto è quello che si evince dal seguente file di esempio.

```
Esercizi: 3
Punteggio: 40* 30* 30
1. Mario, Rossi, 30 30 15
2. Giuseppe, Bianchi, 22 30 15
3. Francesca Maria, Verdi, 32 17 20
4. Anna Paola, De Neri, 28 21 5
```

Si noti che nome e cognome (massimo 30 caratteri ciascuno) possono contenere spazi bianchi.

Si scriva una funzione C che riceva come parametri il nome di un file nel formato illustrato e il nome di un file di output.

La funzione deve scrivere nel file di output i candidati che abbiano preso almeno 6/10 del punteggio su ciascuna prova essenziale. Per i candidati selezionati si deve scrivere cognome, nome e somma dei punti.

Ad esempio, se il file è quello precedente, allora la funzione deve stampare i seguenti candidati

```
Rossi, Mario, 75
De Neri, Anna Paola, 54
```

Si noti che i candidati `Bianchi` e `Verdi` sono stati esclusi per aver preso preso un punteggio inferiore al minimo nelle prove essenziali 1 e 2 rispettivamente.

Esercizio 2 (punti 14) Si considerino le seguenti definizioni di tipo che rappresentano rispettivamente un punto e un segmento nel piano cartesiano.

```
struct Punto                struct Segmento
{                           {
  float x;                     struct Punto inizio;
  float y;                     struct Punto fine;
};                          };
```

Si scriva una funzione C che prenda come parametri un vettore di punti (e la sua dimensione n, con $n > 0$) che rappresenta una linea spezzata nel piano, memorizzata tramite i suoi vertici. La funzione deve restituire tramite una variabile del tipo `struct Segmento` il segmento più lungo della spezzata.

Compito del 14 giugno 2012 (soluzione a pagina 160)

Esercizio 1 (punti 15) Un file contiene un insieme di clienti (uno per riga) di una ditta di trasporti. Ciascuna riga contiene, separati da spazi, le coordinate piane del cliente (in Km) relative al deposito della ditta (che è quindi posizionato nel punto $(0, 0)$) e il volume (intero, in litri) del carico che la ditta deve raccogliere dal cliente. Come esempio si consideri il seguente file.

```
12.4 14.2 1000
-23.4 22.9 500
12.3 -15.23 700
6.78 -5.43 1000
4.3 -5.4 2000
```

La ditta possiede un mezzo che partendo dal deposito visita i clienti nell'ordine con cui compaiono nel file, per poi tornare al deposito. Se la somma del carichi dei clienti supera la capacità del mezzo, allora il mezzo non visita gli ultimi cliente e torna al deposito "saltando" tutti i clienti che gli farebbero eccedere la capacità (il carico è indivisibile).

Si scriva una funzione C che riceva come parametri un file siffatto ed un intero che rappresenta la capacità del mezzo e restituisca in un'opportuna struttura di dati il numero di clienti visitati e la quantità di km percorsi. A questo proposito si considerino le distanze euclidee ("in linea d'aria") tra due punti.

Ad esempio, se il file è quello dell'esempio e la capacità del mezzo è 3000 litri allora la funzione deve restituire la coppia $(3, 127.50)$ in quanto il mezzo può visitare solo i primi 3 clienti e la distanza percorsa è: 116.61 Km. Infatti la distanza dal deposito $(0, 0)$ al primo cliente $(12.414.2)$ è 8.85, dal primo al secondo $(-23.4, 22.9)$ è 36.84, dal secondo al terzo $(12.3, -15.23)$ è 52.23 e la distanza dal terzo al deposito è 8.69. Quindi la distanza totale è $18.85 + 36.84 + 52.23 + 19.58 = 116.61$.

Esercizio 2 (punti 9) Si consideri un campionato di pallavolo a n squadre a girone unico (solo andata), in cui ciascuna squadra è identificata da un valore numerico compreso fra 0 e $n-1$. Una matrice quadrata di interi A di dimensione $n \times n$ contiene i set vinti dalle squadre. In dettaglio, l'elemento A_{ij} rappresenta il numero di set vinti dalla squadra i contro la squadra j, mentre A_{ji} è in numero di set vinti dalla squadra j contro la squadra i nella stessa partita. Gli elementi A_{ii} sulla diagonale principale valgono tutti `0`.

Il numero di punti di ciascuna squadra è calcolato nel modo seguente: 3 punti per la vittoria 3-1 o 3-0, 2 punti per la vittoria 3-2, 1 per la sconfitta 3-2, 0 per la sconfitta 3-1o 3-0.

Si scriva una funzione C che prende come parametri *(i)* una matrice quadrata secondo il formato descritto e di dimensione fisica 50×50, *(ii)* il numero di squadre n (con $n \leq 50$) e *(iii)* un vettore di interi `v` di n elementi. La funzione deve modificare il vettore `v` inserendo in ciascun elemento `v[i]` il numero totale di punti della squadra `i`.

Ad esempio, nel caso della seguente matrice 4×4

```
0  3  1  2
0  0  1  2
3  3  0  0
3  3  3  0
```

il valore di `v` in uscita sarà {`4, 1, 6, 7`}.

Esercizio 3 (punti 6) Si scriva la funzione `main()` per l'esercizio 2 in modo che legga la matrice da un file ricevuto sulla riga di comando come parametro della funzione `main` (il numero di squadre è contenuto nella prima riga del file) e stampi a video i valori del vettore.

Compito del 2 luglio 2012 (soluzione a pagina 163)

Esercizio 1 (punti 15) Un file contiene la descrizione di una sequenza (di lunghezza ignota) di autocisterne che devono scaricare il loro contenuto in un deposito. Ciascuna riga contiene la targa del mezzo (senza spazi) e la quantità di prodotto che trasporta (in m^3). Come esempio, si consideri il seguente file

```
XX333PP 2.45
ZZ000RE 5.78
AA000AA 12.82
AA001AA 9.89
KK111RR 13.9
KK112ZZ 5.24
KK113ZZ 3.24
```

Il deposito contiene un insieme di n silos tutti di capacità c, inizialmente vuoti. Ciascun autocisterna può scaricare il carico in un solo silo e soltanto se tutto il carico entra completamente nel silo (altrimenti non scarica).

I carichi vengono scaricati nell'ordine presente nel file, sempre nel primo silo con capacità disponibile. Se un carico non può essere scaricato in alcun silo l'autocisterna rinuncia e si passa alla successiva.

Si scriva una funzione in linguaggio C che riceve come parametri il nome di un file nel formato suddetto, il numero di silos n e la loro capacità c. La funzione deve restituire il numero di autocisterne che è possibile svuotare con il procedimento illustrato.

Se ad esempio il file è quello mostrato e $n = 3$ e $c = 14.0$, allora la funzione deve restituire 6, in quando è possibile scaricare 6 autocisterne. Infatti, la prima autocisterna si scarica nel primo silo, la seconda ancora nel primo silo, la terza nel secondo silo (nel primo non entra), la quarta nel terzo silo (non entra nel primo e nel secondo), la quinta rinuncia, e la sesta va ancora nel primo silo e infine la settima nel terzo.

Esercizio 2 (punti 15) Una matrice quadrata $n \times n$ (con $n \leq 1000$) descrive l'orografia di un territorio. Ogni elemento della matrice rappresenta un quadrato di terreno e il suo valore (di tipo `float`) rappresenta la sua altitudine media.

Chiamiamo *punto di massimo pericolo di frane* il punto in cui c'è il massimo dislivello rispetto ad uno dei punti ad esso adiacenti. Si considerano adiacenti due punti che differiscono di 1 in uno o entrambi gli indici. Si considera quindi l'adiacenza anche in diagonale, per cui i punti adiacenti ad un dato punto sono al massimo 8.

Ad esempio, data la seguente matrice 10×10

	0	1	2	3	4	5	6	7	8	9
0	90.63	37.81	31.08	66.97	72.15	14.50	89.89	84.26	84.60	21.83
1	90.41	65.41	77.28	42.58	36.68	24.04	67.79	89.28	90.34	69.86
2	24.98	10.92	83.42	5.52	28.58	75.20	83.78	88.33	32.92	80.16
3	39.19	23.55	17.98	70.28	90.52	90.13	84.78	80.41	74.39	69.38
4	2.24	64.80	34.79	79.52	7.38	71.48	3.56	75.17	60.76	93.90
5	45.02	85.73	4.82	28.45	91.25	33.40	3.65	75.03	21.72	36.57
6	55.20	60.92	60.12	73.17	31.20	50.64	63.30	15.98	31.05	37.69
7	85.36	33.29	2.49	20.15	12.81	9.87	91.63	16.37	85.03	52.39
8	10.27	30.06	38.12	15.09	58.50	29.37	48.48	62.15	4.40	70.21
9	98.72	59.60	31.12	58.84	32.77	62.32	9.48	**96.07**	78.30	40.53

il punto di massimo pericolo è il punto $(9, 7)$, evidenziato in grassetto, essendo presente il punto $(8, 8)$ tale che la differenza tra i due, pari a $96.07 - 4.40 = 91.67$, è la massima presente nella matrice.

Si scriva una funzione in linguaggio C che riceve come parametri una matrice siffatta e la sua dimensione n e restituisce (tramite opportuna struttura dati) il punto di massimo pericolo e il relativo dislivello. Nel caso in esame, la funzione deve restituire la tripla $\langle 9, 7, 91.67 \rangle$.

Compito del 7 settembre 2012

Esercizio 1 (punti 15) Un file contiene l'elenco degli esami superati da un insieme di studenti, uno per riga. Per ogni esame, il file contiene la matricola, il cognome e il nome dello studente, la materia, il voto e la data. Ciascun dato è separato dal successivo da una virgola e da uno o più spazi. Si noti che il nome, il cognome e la materia possono contenere degli spazi (la matricola invece non ha spazi).

Come esempio si consideri il seguente file

```
55555,      De Rossi,     Mario,    Fondamenti di informatica, 25,      24-7-2012
12345,    Verdi,     Maria Luisa,    Fondamenti di informatica, 27,     27-9-2011
34343,     Gialli, Pierfilippa,      Reti di calcolatori,  30,  30-10-2010
7888,   Del Neri, Giovanni Maria,   Reti di calcolatori,     30,   12-12-2011
12345, Verdi,  Maria Luisa,   Reti di calcolatori, 24,     4-4-2012
12345,    Verdi,     Maria Luisa,      Basi di dati, 26,     22-10-2011
```

Si scriva una funzione in C che riceva come parametri il nome di un file siffatto, il nome e il cognome di uno studente e una data. La funzione deve restituire la media dei voti dello studenti considerando solo gli esame superati entro tale data.

Per il file dell'esempio, se nome e cognome sono `Maria Luisa` e `Verdi` e la data è il `2-2-2012`, la funzione deve restituire il valore 26.5, in quando l'esame di `Reti di calcolatori` è stao superato dalla studentessa dopo il `2-2-2012`.

Si considerino disponibili il tipo `struct Data` a tre campi interi `giorno`, `mese` e `anno` e la funzione

```
int ComparaDate(struct Data d1, struct Data d2);
```

che restituisce il valore `-1` se la data `d1` è anteriore a `d2`, `0` se le date sono uguali e `+1` nel caso rimanente.

Esercizio 2 (punti 10) Si scriva una funzione in linguaggio C che riceva come parametri una matrice di caratteri A, le sue dimensioni n e m (con $n, m < 1000$), ed una matrice di interi B con le medesime dimensioni di A (che conterrà il risultato dell'operazione) e restituisce un valore intero. La funzione deve riempire la matrice B nel modo seguente: l'elemento $B_{i,j}$ deve corrispondere alla somma delle differenze nei codici ASCII tra l'elemento $A_{i,j}$ e i suoi (al più) 8 vicini $A_{i-1,j-1}$, $A_{i-1,j}$, $A_{i-1,j+1}$, $A_{i,j-1}$, $A_{i,j+1}$, $A_{i+1,j-1}$, $A_{i+1,j}$, $A_{i+1,j+1}$. Inoltre, la funzione deve restituire il massimo in valore assoluto dei valori scritti nella matrice B.

Ad esempio, nel caso della matrice A riportata di seguito, la funzione deve riempire la matrice B come indicato e restituire il valore 273.

$$A = \begin{pmatrix} a & x & d & 1 & f \\ \% & g & m & d & : \\ f & \$ & a & , & C \end{pmatrix} \qquad B = \begin{pmatrix} 31 & 154 & 19 & -224 & 99 \\ -273 & 126 & 223 & 174 & -72 \\ 130 & -268 & 93 & -211 & -1 \end{pmatrix} \tag{1}$$

Esercizio 3 (punti 5) Si scriva la funzione `main()` per l'esercizio 2 in modo che legga n, m ed A da un file il cui nome è passato sulla riga di comando (cioè come parametro della funzione `main`).

Compito del 29 gennaio 2013

Esercizio 1 (punti 14) Un file contiene le previsioni del tempo di una giornata, una previsione per riga. Ciascuna riga è formata dall'ora (ore e minuti), la previsione atmosferica e la temperatura prevista. La previsione è una stringa terminata da una virgola che può contenere anche spazi bianchi. La temperatura è un valore reale, che può essere scritto sia in gradi Celsius che Fahrenheit (contraddistinti dalla lettera `C` oppure `F`). Come esempio, si consideri il seguente file.

```
1.30 coperto con qualche pioggia, 0.8C
4.00 pioggia debole, 0.7C
7.30 pioggia debole, 28.8F
10.00 pioggia debole, 2.7C
13.00 pioggia debole, 31.9F
16.20 pioggia debole, 4.1C
19.00 pioggia e schiarite, 2.5C
21.15 sereno, -1.3C
22.10 sereno, 0.2C
```

Si scriva una funzione che riceva come parametro il nome di un file siffatto e restituisca, in una opportuna struttura dati (`struct`), la temperatura minima della giornata (scritta in gradi Celsius) e l'orario in cui è prevista.

Si ricordi che la temperatura in gradi Celsius (°C) a partire da quella in gradi Fahrenheit (°F) si ottiene con la seguente formula: °C = (°F - 32)/1.8.

Nell'esempio, la funzione dovrà restituire l'orario 7:30 e la temperatura -1.778 (corrispondente a 28.8°F).

Esercizio 2 (punti 9) Si consideri la seguente definizione per la rappresentazione dei numeri complessi.

```
struct Complesso
{
  double re;
  double im;
};
```

Si scriva una funzione che riceva come parametri due vettori di numeri complessi e la loro dimensione n (la stessa per i due vettori). La funzione deve restituire il numero complesso risultante dal prodotto scalare dei due vettori.

Suggerimento: Si scriva una funzione ausiliaria che calcola il prodotto tra due numeri complessi.

Esercizio 3 (punti 7) Si scriva la funzione `main()` per verificare la funzione dell'esercizio precedente (assumendo $n \leq 100$). La funzione `main()` riceve *sulla riga di comando* la lunghezza n e i nomi di due file in cui sono scritti i vettori con parte reale e parte immaginaria separate da uno spazio. Come esempio di file di input, si consideri il seguente (con $n = 5$).

```
7.6 -34.1
23.44 12.2
12.3 2.32
4.33 10
0.2 -11.29
```

Compito del 26 febbraio 2013 (soluzione a pagina 166)

Esercizio 1 (punti 15) Un file contiene i risultati di un insieme di esperimenti, un esperimento per riga. Ciascuna riga è formata dalla data di esecuzione, il valore dei due parametri di controllo A e B con cui è stato eseguito l'esperimento (interi positivi) e il risultato ottenuto (valore reale). I dati sono scritti il formato CSV (*comma separated values*), cioè racchiusi tra doppi apici e separati tra loro da virgole. Non ci sono spazi bianchi nel file.

Come esempio, si consideri il seguente file.

```
"23/12/2012","10","8","22.44"
"22/12/2012","9","12","24.76"
"27/12/2012","7","14","24.47"
"23/12/2012","4","16","22.55"
"29/12/2012","10","17","21.98"
"23/12/2012","8","18","22.15"
"3/1/2013","9","10","22.35"
"23/12/2012","11","8","22.32"
"24/12/2012","9","8","21.75"
```

Si scriva una funzione che riceve come parametri il nome di un file siffatto e una data `d` e restituisce la coppia di valori dei parametri A e B che hanno ottenuto il risultato massimo tra gli esperimenti eseguiti dopo la data `d`. Nel caso in cui nessun esperimento è stato eseguito dopo la data `d`, allora la funzione deve restituire la coppia $\langle 0, 0 \rangle$.

Si assumano già disponibili la definizione del tipo `struct Data` (nel classico formato a tre campi interi) e la funzione `int ComparaDate(struct Data d1, struct Data d2)`, che restituisce `-1` se la data `d1` è precedente a `d2`, `0` se le due date sono uguali e `1` se `d1` è successiva a `d2`.

Nell'esempio, se la data è `28/12/2012` allora la funzione deve restituire la coppia $\langle 9, 10 \rangle$ (il valore massimo è 22.35), mentre se la data è `22/12/2012` allora deve restiuire la coppia $\langle 7, 14 \rangle$ (valore massimo 24.47).

Esercizio 2 (punti 15) Una matrice intera di dimensione $n \times n$ (con $n \leq 500$) rappresenta l'altitudine in metri di una zona di terreno.

Si scriva una funzione *booleana* che riceve come parametri una matrice siffatta (e la sua dimensione n) ed un intero positivo k e verifica se esiste o no un *altopiano* di estensione k interamente compreso nella matrice. Si definisce altopiano di estensione k una regione di dimensione $k \times k$ tale che la differenza tra l'altitudine minima e massima della regione è al più 10 metri.

Come esempio si consideri la seguente matrice (con $n = 14$).

120	116	119	120	101	121	107	117	120	116	107	115	121	101
120	117	111	116	100	106	103	125	109	101	110	115	118	124
117	105	107	111	121	102	107	125	124	114	118	118	107	125
109	102	100	104	121	113	120	123	122	125	122	107	103	107
122	121	107	115	101	114	100	124	119	109	123	117	124	115
111	107	117	120	111	119	100	106	107	120	105	105	122	102
112	125	111	111	122	118	100	123	109	103	122	102	112	121
121	112	113	106	119	104	102	104	123	103	112	106	123	118
113	121	122	100	120	107	111	117	125	113	116	110	116	114
114	105	110	109	119	123	117	113	103	120	119	100	123	108
109	122	102	122	118	124	122	114	105	109	107	106	125	124
117	117	112	107	122	124	117	116	121	110	105	124	104	100
101	101	108	110	100	110	108	118	108	107	106	115	118	116
122	117	114	115	109	102	122	107	101	115	125	124	102	104

Nella matrice dell'esempio esiste un altopiano di dimensione $k = 3$ (evidenziato in figura), ma non esiste alcun altopiano di dimensione $k \geq 4$.

Soluzioni

Soluzione compito del 19 marzo 2001

Soluzione esercizio 1

La funzione che risolve l'esercizio si compone di due cicli annidati, di cui quello esterno legge le righe, e quello interno le singole parole.

Per una corretta modularizzazione della funzione è indispensabile che il compito di tradurre i numeri scritti a lettere in interi sia affidato ad una funzione specifica. Tale funzione, chiamata `Valore()`, prende come parametro una stringa e restituisce un numero intero.

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

int SommaInteri(char nome_file[]);
int Valore(char cifra_lettere[]);

int main()
{
  char nome_file[21];

  printf("Nome del file : ");
  scanf("%s", nome_file);
  printf("La somma e' %d\n", SommaInteri(nome_file));
  return 0;
}

int SommaInteri(char nome_file[])
{
  int somma = 0, num;
  char cifra_lettere[8];
  FILE* fp;

  fp = fopen(nome_file,"r");
  while (fscanf(fp,"%s", cifra_lettere) != EOF)
    {
      num = 0;
      while (strcmp(cifra_lettere,"stop") != 0)
        {
          num = num * 10 + Valore(cifra_lettere);
          fscanf(fp,"%s", cifra_lettere);
        }
      somma += num;
    }
  fclose(fp);
  return somma;
}

int Valore(char cifra_lettere[])
```

```
{
  if (strcmp(cifra_lettere,"uno") == 0)
    return 1;
  else if (strcmp(cifra_lettere,"due") == 0)
    return 2;
  else if (strcmp(cifra_lettere,"tre") == 0)
    return 3;
  else if (strcmp(cifra_lettere,"quattro") == 0)
    return 4;
  else if (strcmp(cifra_lettere,"cinque") == 0)
    return 5;
  else if (strcmp(cifra_lettere,"sei") == 0)
    return 6;
  else if (strcmp(cifra_lettere,"sette") == 0)
    return 7;
  else if (strcmp(cifra_lettere,"otto") == 0)
    return 8;
  else if (strcmp(cifra_lettere,"nove") == 0)
    return 9;
  else
    return 0;
}
```

Soluzione esercizi 2 e 3

Per l'esercizio 3 viene proposta una soluzione basata sull'inserimento di valori casuali, utilizzando le funzioni `srand()` e `rand()`. Questa soluzione va al di là degli argomenti nel programma del corso, e viene presentata come approfondimento. La soluzione basata sull'inserimento dei valori da tastiera viene lasciata per esercizio allo studente.

```
#include <stdio.h>
#include <stdlib.h>
#include <time.h>
#define PARTITE 5

struct Partita
{
  int dado1;
  int dado2;
  char moneta;
};

int VincitoreSfida(struct Partita partite[], int n);

int main()
{
  int i, vincitore;
  struct Partita partite[PARTITE];

  srand(time(NULL)); /* inizializza la sorgente random
                        con l'orologio di sistema */
  for (i = 0; i < PARTITE; i++)
    {
      partite[i].dado1 = rand() % 6 + 1;
      partite[i].dado2 = rand() % 6 + 1;
      if (rand() % 2 == 0)
        partite[i].moneta = 't';
      else
        partite[i].moneta = 'c';
      printf("%d-%d-%c\n", partite[i].dado1, partite[i].dado2, partite[i].moneta);
```

```
    }

  vincitore = VincitoreSfida(partite,PARTITE);
  if (vincitore == 1)
    printf("Ha vinto il giocatore 1\n");
  else if (vincitore == -1)
    printf("Ha vinto il giocatore 2\n");
  else
    printf("Parita'\n");
  return 0;
}


int VincitoreSfida(struct Partita sfida[], int n)
{
  int vittorie1 = 0, vittorie2 = 0, i;

  for (i = 0; i < n; i++)
    {
      if (sfida[i].moneta == 't')
        {
          if (sfida[i].dado1 > sfida[i].dado2)
            vittorie1++;
          else if (sfida[i].dado1 < sfida[i].dado2)
            vittorie2++;
        }
      else
        {
          if (sfida[i].dado1 < sfida[i].dado2)
            vittorie1++;
          else if (sfida[i].dado1 > sfida[i].dado2)
            vittorie2++;
        }
    }

  if (vittorie1 > vittorie2)
    return 1;
  else if (vittorie2 > vittorie1)
    return -1;
  else
    return 0;
}
```

Soluzione compito del 2 aprile 2001

Soluzione esercizi 1 e 2

Siccome nome e cognome possono contenere degli spazi è preferibile usare per la loro scansione una lettura carattere per carattere (funzione `getc()`). Si noti inoltre che non interessa memorizzare il contenuto del nome e del cognome, e quindi non serve assegnare il valore restituito dalla `getc()` ad una variabile.

La lettura dell'eventuale commento è ancora fatta con un ciclo governato dalla lettura di un carattere. In questo caso, il carattere deve essere memorizzato perché è necessario confrontarlo non solo con `\n`, ma anche con `EOF`, per gestire il caso che l'ultima linea del file non contenga il carattere `\n`.

```
#include <stdio.h>

int StudentiFuoriCorso(char nome_file[], int anno_cercato);
```

```
int main(int argc, char *argv[])
{
  int anno, studenti;
  if (argc != 3)
    {
      printf("Errore nell'inserimento dei parametri\n");
      exit(1);
    }

  anno = atoi(argv[2]);
  studenti = StudentiFuoriCorso(argv[1],anno);
  if (studenti != -1)
    printf("Il numero di studenti fuori corso nell'anno %d e' %d\n",
           anno, studenti);
  else
    printf("Errore nei dati\n");
}

int StudentiFuoriCorso(char nome_file[], int anno_cercato)
{
  FILE *fp;
  int conta = 0, anno_corso;
  char ch, posizione[3];

  if (anno_cercato < 1 || anno_cercato > 5)
    return -1;

  if ((fp = fopen(nome_file, "r")) == NULL)
    return -1;

  while ((ch = getc(fp)) != EOF)
    {
      while(getc(fp) != ',')
        ;
      while(getc(fp) != ',')
        ;
      fscanf(fp,"%d%s",&anno_corso,posizione);
      if (anno_corso == anno_cercato && strcmp(posizione,"FC") == 0)
        conta++;
      while ((ch = getc(fp)) != '\n' && ch != EOF)
        ;
    }
  fclose(fp);
  return conta;
}
```

Soluzione esercizio 3

Una buona modularizzazione consiste nel definire una funzione, chiamata `QuadranteSegmento()` che restituisce il quadrante del segmento, se esiste, e 0 se il segmento non appartiene ad alcun quadrante. Questa funzione, a sua volta, utilizza una funzione, chiamata `QuadrantePunto()` che restituisce il quadrante di un singolo punto.

Il risultato di `QuadranteSegmento()` può essere utilizzato direttamente come indice di un vettore che conta il numero di segmenti per ciascun settore. I segmenti che non appartengono ad alcun settore vengono contati nella locazione `0`.

```
#include <stdio.h>

#define DIM 3
```

```
struct Complesso
{
  float re; /* parte reale */
  float im; /* parte immaginaria */
};

struct Segmento
{
  struct Complesso p1;
  struct Complesso p2;
};

int MaxSegmenti(struct Segmento v[], int n);
int QuadrantePunto(struct Complesso p);
int QuadranteSegmento(struct Segmento s);


/* non richiesto per il compito in classe */
int main()
{
  struct Segmento v[DIM];
  int i;
  for (i = 0; i < DIM; i++)
    {
      printf("Segmento %d\n", i+1);
      printf("Coordinate primo punto : ");
      scanf("%g%g",&v[i].p1.re,&v[i].p1.im);
      printf("Coordinate secondo punto : ");
      scanf("%g%g",&v[i].p2.re,&v[i].p2.im);
    }
  printf("Il massimo numero di punti e' nel quadrante %d\n",
         MaxSegmenti(v,DIM));
  return 0;
}

int MaxSegmenti(struct Segmento v[], int n)
{
  int i, a[5] = {0}, max = 1;
  for (i = 0; i < n; i++)
    a[QuadranteSegmento(v[i])]++;

  /* la locazione 0 del vettore a contiene il numero di
     segmenti che non appartengono ad alcun quadrante.
     Questa locazione non viene considerata nella ricerca
     del massimo del vettore */

  for (i = 2; i < 5; i++)
    if (a[i] > a[max])
      max = i;
  return max;
}


/* restituisce il quadrante del segmento, 0 se il segmento
   non appartiene ad alcun quadrante */
int QuadranteSegmento(struct Segmento s)
{
  int q1, q2;
```

```
  q1 = QuadrantePunto(s.p1);
  q2 = QuadrantePunto(s.p2);

  if (q1 == q2)
    return q1;
  else
    return 0;
}

int QuadrantePunto(struct Complesso p)
{
  if (p.re > 0)
    if (p.im > 0)
      return 1;
    else
      return 4;
  else
    if (p.im > 0)
      return 2;
    else
      return 3;
}
```

Soluzione compito del 9 luglio 2001

Soluzione esercizio 1

La soluzione prescelta ignora la divisione in righe e si basa soltanto sui caratteri che seguono i numeri (cioè `+`, `=` e `;`). In base a tali caratteri, la funzione intraprende azioni diverse: somma il valore, passa ad analizzare la sommatoria a destra, oppure passa ad una nuova equazione, rispettivamente. Inoltre, essa utilizza una variabile booleana, chiamata `parte_sin`, che tiene traccia del fatto che si sta analizzando la parte a sinistra o a destra del segno di uguaglianza.

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

float ContaEquazioni(char nome_file[]);

/* non richiesto per il compito in classe */
int main(int argc, char *argv[])
{
  if (argc != 2)
    {
      printf("Errore nell'inserimento dei parametri\n");
      exit(1);
    }
  printf("La frazione di equazioni corrette e' : %f\n",ContaEquazioni(argv[1]));
  return 0;
}

float ContaEquazioni(char nome_file[])
{
  FILE *fp;
  int num, sin_val = 0, des_val = 0,
    equazioni = 0, equazioni_corrette = 0;
  int parte_sin = 1;
  char ch;
```

```
  fp = fopen(nome_file,"r");
  while (fscanf(fp,"%d%c",&num,&ch) != EOF)
    {
      if (ch == '+')
        if (parte_sin) /* siamo nella parte sinistra */
          sin_val += num;
        else
          des_val += num;
      else if (ch == '=')
        {
          sin_val += num;
          parte_sin = 0;
        }
      else /* ch == ';' */
        {
          des_val += num;
          equazioni++;
          if (sin_val == des_val)
            equazioni_corrette++;
          sin_val = 0;
          des_val = 0;
          parte_sin = 1;
        }
    }
  fclose(fp);
  return (float) equazioni_corrette/equazioni;
}
```

Soluzione esercizio 2

Definiamo due funzioni ausiliarie che rispettivamente leggono e scrivono i dati di un singolo pixel. Definiamo inoltre una funzione, chiamata `BiancoENero()` che, dato un pixel, restituisce il corrispondente valore in bianco e nero.

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

struct Pixel
{
  int r;
  int g;
  int b;
};

struct Pixel LeggiPixel(FILE *fin);
void ScriviPixel(FILE *fout, struct Pixel p);
struct Pixel BiancoENero(struct Pixel p);

int main (int argc, char *argv[])
{
   struct Pixel p;
   struct Pixel bianco = {255, 255, 255};
   int r, c, i, j;
   FILE *fin, *fout;

   if (argc != 3)
   {
      printf ("Errato numero di parametri\n");
      exit (EXIT_FAILURE);
```

```
   }

   fin = fopen (argv[1], "r");
   fout = fopen (argv[2], "w");

   fscanf (fin, "%d %d", &r, &c);
   /* incrementa le dimensioni per tenere conto del bordo da aggiungere */
   r += 2;
   c += 2;
   fprintf (fout, "%d %d\n", r, c);

   /* bordo superiore */
   for (j = 0; j < c; j++)
      ScriviPixel(fout, bianco);

   for (i = 1; i < r-1; i++)
   {
      /* bordo sinistro */
      ScriviPixel (fout, bianco);

      /* riga dell'immagine originale */
      for (j = 1; j < c-1; j++)
      {
         p = LeggiPixel(fin);
         ScriviPixel(fout, BiancoENero(p));
      }
      /* bordo destro */
      ScriviPixel (fout, bianco);
   }

   /* bordo inferiore*/
   for (j = 0; j < c; j++)
      ScriviPixel (fout, bianco);

   fclose(fin);
   fclose(fout);
   return 0;
}

struct Pixel LeggiPixel(FILE *fin)
{
   struct Pixel p;
   fscanf(fin, "%d %d %d", &p.r, &p.g, &p.b);
   return p;
}


void ScriviPixel(FILE *fout, struct Pixel p)
{
   fprintf (fout, "%d %d %d\n", p.r, p.g, p.b);
}


struct Pixel BiancoENero(struct Pixel p)
{
   int media;
   struct Pixel grigio;
   media = (p.r + p.g + p.b) / 3;
   grigio.r = media;
```

```
   grigio.g = media;
   grigio.b = media;
   return grigio;
}
```

Soluzione compito del 23 luglio 2001

Soluzione esercizio 1

Per poter trovare le parole verticali è indispensabile memorizzare il cruciverba in una matrice in memoria centrale. Per questo compito si utilizza una funzione ausiliaria, che memorizza non solo la matrice, ma anche le sue dimensioni (attraverso parametri per riferimento).

La funzione `TrovaParoleOrizzontali()` memorizza ciascuna parola in una stringa, variabile `parola`, e quando la parola è stata letta integralmente la scrive sul file di uscita. Per gestire la memorizzazione delle parole dalla matrice alla stringa si utilizza la variabile booleana `in_parola` che dice se si è all'interno di una parola oppure no.

La funzione `TrovaParolaVerticali()` è analoga a `TrovaParoleOrizzontali()`, ma ha semplicemente gli indici della matrice invertiti.

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

#define MAXR 50
#define MAXC 50

void LeggiCruciverba(char nome_file[], char cruci[MAXR][MAXC], int *rp, int *cp);
void TrovaParoleOrizzontali(char cruci[MAXR][MAXC], int r, int c, FILE *fpout);
void TrovaParoleVerticali(char cruci[MAXR][MAXC], int r, int c, FILE *fpout);

int main (int argc, char *argv[])
{
   /* il massimo numero di colonne e' aumentato di uno per inserire
      \0 alla fine di ogni riga */
   char cruci[MAXR][MAXC];
   int r, c;

   FILE *fp;

   if (argc != 2)
   {
      printf ("E' richiesto il nome del file\n");
      exit(1);
   }

   LeggiCruciverba (argv[1], cruci, &r, &c);
   fp = fopen("parole.txt", "w");
   TrovaParoleOrizzontali (cruci, r, c, fp);
   fprintf (fp, "\n");
   TrovaParoleVerticali (cruci, r, c, fp);
   fclose (fp);
   return 0;
}

void LeggiCruciverba(char nome_file[], char cruci[MAXR][MAXC], int *rp, int *cp)
{
   int i, j, r, c;
   FILE *fp;
```

```
   fp = fopen (nome_file, "r");
   fscanf (fp, "%d %d", &r, &c);
   while (getc (fp) != '\n')
     ;

   for (i = 0; i < r; i++)
   {
     for (j = 0; j < c; j++)
       cruci[i][j] = getc(fp);
     getc(fp); /* legge il \n */
     cruci[i][c] = '\0';
   }
   *rp = r;
   *cp = c;
   fclose(fp);
}

void TrovaParoleOrizzontali(char cruci[MAXR][MAXC], int r, int c, FILE *fpout)
{
   int i, j, k = 0;
   int in_parola = 0; /* booleano: vale true se stiamo leggendo una parola */
   char parola[MAXC+1];

   /* trova le parole e scrive nel file quelle di lunghezza
      maggiore o uguale a 2 */

   for (i = 0; i < r; i++)
   {
      for (j = 0; j < c; j++)
      {
         if (cruci[i][j] != ' ')
         {
            if (!in_parola)
            {
               in_parola = 1;
               parola[k] = cruci[i][j];
               k++;
            }
            else
            {
               parola[k] = cruci[i][j];
               k++;
            }
         }
         else  /* cruci[i][j] == ' ' */
         {
            if (in_parola)
              { /* la parola e' finita */
               in_parola = 0;
               parola[k] = '\0';

               if (k > 1)
                 fprintf (fpout, "%s\n", parola);

               k = 0;
            }
         }
      }
```

```
      /* fine della riga, chiudi la parola */
      parola [k] = '\0';

      if (k > 1)
         fprintf (fpout, "%s\n", parola);

      k = 0;
   }
}


/* La seguente funzione e' uguale alla precedente, ma l'ordine dei
   cicli e' invertito. E' anche possibile scrivere un'unica funzione
   parametrica, in cui la direzione (orizzontale o verticale) e' passata
   come argomento. */

void TrovaParoleVerticali(char cruci[MAXR][MAXC], int r, int c, FILE *fpout)
{
   int i, j, k = 0;
   int in_parola = 0;
   char parola[MAXR+1];

   /* trova le parole e scrive nel file quelle di lunghezza
      maggiore o uguale a 2 */

   for (j = 0; j < c; j++)
 {
      for (i = 0; i < r; i++)
      {
         if (cruci[i][j] != ' ')
         {
            if (!in_parola)
            {
               in_parola = 1;
               parola[k] = cruci[i][j];
               k++;
            }
            else
            {
              parola[k] = cruci[i][j];
              k++;
            }
         }
         else  /* cruci[i][j] == ' ' */
         {
            if (in_parola)
            {
               in_parola = 0;
               parola [k] = '\0';

               if (k > 1)
                  fprintf (fpout, "%s\n", parola);

               k = 0;
            }
         }
      }

      /* fine della riga, chiudi la parola */
```

```
      parola [k] = '\0';

      if (k > 1)
         fprintf (fpout, "%s\n", parola);

      k = 0;
   }
}
```

Soluzione esercizio 2

La funzione `f()` assegna alle variabili il cui indirizzo è passato come terzo e quarto parametro (`c` e `d`) il quoziente ed il resto della divisione del primo per il secondo parametro (`a` e `b`).

Nel caso specifico, il programma stampa `20 3 6 2` in quanto `6` e `2` sono il modulo e il resto della divisione di `20` per `3`, i quali a loro volta sono i valori di `m` e `n` che rimangono inalterati (essendo passati per valore ad `f()`).

Soluzione compito del 6 settembre 2001

Soluzione esercizi 1 e 2

Per l'esercizio 1, si utilizzano due variabili intere che memorizzano lo spostamento totale nelle due direzioni (nord-sud ed est-ovest). Ad esempio, la variabile `spostamento_nord` memorizza lo spostamento nord-sud, ed assume valore positivo se lo spostamento è verso nord e negativo se è verso sud.

Per l'esercizio 2 è sufficiente memorizzare tutti gli spostamenti in un vettore, e riscrivere il contenuto del vettore sul file di uscita al contrario, invertendo le direzioni.

```
#include <stdio.h>

#define MAX_DIM 100

void StampaSpostamento(char nome_file[]);
void PercorsoInverso(char nome_file_input[], char nome_file_output[]);
char DirezioneInversa(char d);

struct Spostamento
{
  char direzione;
  int distanza;
};

/* non richiesto per il compito in classe */
int main()
{
  char file_input[20];
  char file_output[20];

  printf("Esercizio 1\n");
  printf("Nome file di input : ");
  scanf("%s",file_input);
  StampaSpostamento(file_input);

  printf("Esercizio 2\n");
  printf("Nome file di input : ");
  scanf("%s",file_input);
  printf("Nome file di output : ");
  scanf("%s",file_output);
```

```
  PercorsoInverso(file_input,file_output);

  return 0;
}

void StampaSpostamento(char nome_file[])
{
  FILE* fp;
  int spostamento_nord = 0, spostamento_est = 0;
  int distanza;
  char direzione, ch;

  fp = fopen(nome_file, "r");
  while (fscanf(fp,"%c%d", &direzione, &distanza) != EOF)
    {
      switch(direzione)
        {
        case 'N': spostamento_nord += distanza; break;
        case 'S': spostamento_nord -= distanza; break;
        case 'E': spostamento_est += distanza; break;
        case 'O': spostamento_est -= distanza; break;
        }
      while ((ch = getc(fp)) != '\n' && ch != EOF)
        ; /* vai all'inizio della riga successiva (se esiste) */
    }
  fclose(fp);

  if (spostamento_nord > 0)
    printf("%d METRI VERSO N\n", spostamento_nord);
  else if (spostamento_nord < 0)
    printf("%d METRI VERSO S\n", -spostamento_nord);
  if (spostamento_est > 0)
    printf("%d METRI VERSO E\n", spostamento_est);
  else if (spostamento_est < 0)
    printf("%d METRI VERSO O\n", -spostamento_est);
}

void PercorsoInverso(char nome_file_input[], char nome_file_output[])
{
  FILE *ifp, *ofp;
  int num_spostamenti, i = 0;
  char ch;
  struct Spostamento vet[MAX_DIM];

  ifp = fopen(nome_file_input, "r");
  ofp = fopen(nome_file_output, "w");

  while (fscanf(ifp,"%c%d", &vet[i].direzione, &vet[i].distanza) != EOF)
  {
    i++;
    while ((ch = getc(ifp)) != '\n' && ch != EOF)
      ; /* vai all'inizio della riga successiva (se esiste)*/
  }
  fclose(ifp);
  num_spostamenti = i;

  for (i = num_spostamenti - 1; i >= 0; i--)
    fprintf(ofp,"%c %d\n", DirezioneInversa(vet[i].direzione), vet[i].distanza);
  fclose(ofp);
```

```
}

char DirezioneInversa(char d)
{
  switch(d)
    {
    case 'N': return 'S';
    case 'S': return 'N';
    case 'E': return 'O';
    case 'O': return 'E';
    }
  return 'X'; /* input non corretto */
}
```

Soluzione compito del 20 settembre 2001

Soluzione esercizio 1

Il nome del file di uscita viene ottenuto semplicemente copiando il nome del file di ingresso e cambiando la `i` di `tic` in `o`.

Per ottenere l'eliminazione dei commenti è indispensabile leggere il file per carattere per carattere (funzione `getc()`). Quando si legge un carattere normale lo si scrive, quando invece si legge il carattere `#`, si salta il carattere stesso e tutto il resto della riga.

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

void EliminaCommenti(char nome_file_input[]);

/* non richiesto per il compito in classe */
int main (int argc, char *argv[])
{
  if (argc != 2)
    {
      printf("Passare il nome del file di input sulla riga di comando!");
      exit(1);
    }
  EliminaCommenti(argv[1]);
  return 0;
}

void EliminaCommenti(char nome_file_input[])
{

  FILE *fin, *fout;
  char nome_file_output[21], ch;

  strcpy(nome_file_output, nome_file_input);
  /* cambia in o la i di tic (in penultima posizione) */
  nome_file_output[strlen(nome_file_output)-2] = 'o';

  fin = fopen (nome_file_input, "r");
  fout = fopen (nome_file_output, "w");
  while ((ch = getc (fin)) != EOF)
    {
      if (ch != '#')
        fprintf(fout,"%c",ch);
      else
```

```
        {
          while ((ch = getc (fin)) != EOF && ch != '\n')
            ; /* istruzione vuota */
          fprintf(fout,"\n");
        }
    }
   fclose (fin);
   fclose (fout);
}
```

Soluzione esercizio 2

La funzione che risolve l'esercizio, chiamata `PesoTotale()`, si compone di un semplice ciclo di lettura dal file, in cui ad ogni iterazione si legge una riga e si calcola il peso dell'ingrediente.

Si noti che, per una buona modularità, la ricerca di un ingrediente nel vettore è delegata ad una funzione ausiliaria.

Si noti inoltre che, per maggiore generalità, si è supposto che l'unità di misura possa essere anche più lunga di un carattere (fino a tre caratteri).

Si noti infine che non c'è differenza tra ingredienti espressi in unità oppure in litri: per entrambi si moltiplica il peso per il loro peso specifico.

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

#define DIM 32

struct PesiSpecifici
{
   char ingrediente[DIM];
   float peso;
};

float PesoTotale(char nomefile[], struct PesiSpecifici pesi[], int n_pesi);
int CercaIngrediente(char nome_ingrediente[], struct PesiSpecifici pesi[], int n_pesi);

/* non richiesto per il compito in classe */
int main (int argc, char *argv[])
{
 struct PesiSpecifici pesi[4] = {
   {"latte", 1000},
   {"olio", 950},
   {"acqua", 1000},
   {"uova", 75.0}
 };

  if (argc != 2)
   {
      printf("E' richiesto il nome del file\n");
      exit(1);
   }

   printf("Peso totale: %f\n", PesoTotale(argv[1], pesi, 4));

   return 0;
}

float PesoTotale (char nomefile[], struct PesiSpecifici pesi[], int n_pesi)
{
```

```
  FILE *fp;
  float quantita, peso_totale = 0.0;
  char ingrediente[DIM], unita[4];
  int indice_ingrediente;

  fp = fopen (nomefile, "r");
  while (fscanf (fp, "%s %s %f", ingrediente, unita, &quantita) != EOF)
    {
      if (strcmp(unita,"g") == 0)
        peso_totale += quantita;
      else
        {
          /* cerca il peso specifico */
         indice_ingrediente = CercaIngrediente(ingrediente, pesi, n_pesi);
         peso_totale += pesi[indice_ingrediente].peso * quantita;
        }
    }
   fclose (fp);
   return peso_totale;
}

int CercaIngrediente(char nome_ingrediente[], struct PesiSpecifici pesi[], int n_pesi)
{
   int i;
   for (i = 0; i < n_pesi; i++)
     if (strcmp(nome_ingrediente, pesi[i].ingrediente) == 0)
       return i;
   return -1;
}
```

Soluzione compito del 10 dicembre 2001

Soluzione esercizio 1

L'esercizio si risolve memorizzando i valori letti dal file in tre vettori distinti in base al confronto con le due soglie. Alla fine della lettura, i tra vettori vengono riversati sul file uno dopo l'altro.

```
#include <stdio.h>
#include <stdlib.h>
#define MAX_DIM 1000

void DividiVettore(char nome_file[], int s1, int s2);


int main(int argc, char* argv[])
{/* non richiesto per il compito in classe */
  int a, b;
  if (argc != 4

  printf("Soglia 1 : ");
  scanf("%d",&a);
  printf("Soglia 2 : ");
  scanf("%d",&b);
  DividiVettore("Dati.txt",a,b);
  return 0;
}

void DividiVettore(char nome_file[], int s1, int s2)
{
```

```
  FILE* fp;
  float v1[MAX_DIM], v2[MAX_DIM], v3[MAX_DIM], num;
  int i, i1 = 0, i2 = 0, i3 = 0;

  fp = fopen(nome_file,"r");
  if (fp == NULL)
    {
      printf("Errore nell'apertura del file %s\n", nome_file);
      exit(1);
    }

  while (fscanf(fp,"%g", &num) != EOF)
    {
      if (num < s1)
        {
          v1[i1] = num;
          i1++;
        }
      else if (num < s2)
        {
          v2[i2] = num;
          i2++;
        }
      else
        {
          v3[i3] = num;
          i3++;
        }
    }
  fclose(fp);
  fp = fopen(nome_file,"w");

  for (i = 0; i < i1; i++)
    fprintf(fp,"%g ", v1[i]);
  fprintf(fp,"# ");
  for (i = 0; i < i2; i++)
    fprintf(fp,"%g ", v2[i]);
  fprintf(fp,"# ");
  for (i = 0; i < i3; i++)
    fprintf(fp,"%g ", v3[i]);

  fclose(fp);
}
```

Soluzione esercizio 2

La funzione inizialmente copia la descrizione dell'articolo nella variabile `descrizione`. La fine della descrizione è sancita dalla presenza del carattere `':'`.

Successivamente vengono letti l'importo e la valuta (lire oppure euro). Per capire se sia presente prima l'importo oppure prima la valuta, si verifica se il primo carattere dopo lo spazio che segue la descrizione è numerico oppure alfabetico.

Finita la lettura, se necessario si converte la valuta, e infine si ricostruisce la stringa di uscita sfruttando la funzione `sprintf()`.

```
#include <stdio.h>
#include <string.h>
#include <stdlib.h>
#define MAX_DESCRL 128
```

```
void ConvertiArticolo (char s[]);

/* non richiesto per il compito in classe */
int main (int argc, char *argv[])
{
  char s[256];
  int i = 0;

  printf("Inserisci la stringa (terminata da ;) : ");
  while ((s[i] = getchar()) != ';')
    i++;
  /* elimina il carattere ';' */
  s[i] = '\0';

  printf("La stringa in ingresso e' : %s\n", s);
  ConvertiArticolo (s);
  printf("La stringa risultante e' : %s\n", s);
  return 0;
}

void ConvertiArticolo (char s[])
{
   char descrizione[MAX_DESCRL];
   char prezzo[16], valuta[16];
   int i, j;

   for (i = 0; s[i] != ':'; i++)
      descrizione[i] = s[i];
   descrizione[i] = '\0';
   i++; /* si posiziona oltre i ':' */
   while (s[i] == ' ')
      i++;
   /* ora legge i restanti due elementi */
   if (s[i] >= '0' && s[i] <= '9')
   {
      j = 0;
      while (s[i] != ' ')
        {
          prezzo[j] = s[i];
          j++;
          i++;
        }
      prezzo[j] = '\0';
      while (s[i] == ' ')
         i++;
      j = 0;
      while (s[i] != '\0')
        {
          valuta[j] = s[i];
          j++;
          i++;
        }
      valuta[j] = '\0';
   }
   else
     {
       j = 0;
       while (s[i] != ' ')
```

```
        {
          valuta[j] = s[i];
          j++;
          i++;
        }
       valuta[j] = '\0';
      while (s[i] == ' ')
        i++;
      j = 0;
      while (s[i] != '\0')
        {
          prezzo[j] = s[i];
          j++;
          i++;
        }
      prezzo[j] = '\0';
     }
   if (valuta[0] == 'l' || valuta[0] == 'L')
     {
       /* conversione da lire in euro */
       int lire;
       lire = atoi(prezzo);
       sprintf (prezzo, "%.2f", (float) lire/1936.27);
   }
   sprintf (s, "%s: %s euro", descrizione, prezzo);
}
```

Soluzione compito del 20 marzo 2002

Soluzione esercizio 1

La funzione si compone di un semplice ciclo di lettura del file riga per riga. Per ogni riga letta si aggiorna il saldo e si verifica se quest'ultimo è sceso sotto il valore di soglia. In caso positivo, la funzione termina restituendo la data del movimento corrente.

Se il controllo giunge alla fine del ciclo, si restituisce la data dell'ultimo movimento.

```
#include <stdio.h>

struct Data
{
  int giorno;
  int mese;
};

struct Data DataLimite(char nome_file[], float limite);

/* non richiesto per il compito in classe */
int main()
{
  char nome[21];
  struct Data data;
  float limite;

  printf("Nome del file dei movimenti : ");
  scanf("%s",nome);
  printf("Valore limite (negativo) : ");
  scanf("%g",&limite);
```

```
  data = DataLimite(nome,limite);
  printf("Data limite %d-%d\n",data.giorno,data.mese);
  return 0;
}

struct Data DataLimite(char nome_file[], float limite)
{
  FILE* fp;
  float saldo, movimento;
  struct Data ris;
  char segno;

  fp = fopen(nome_file,"r");
  fscanf(fp,"%*s%g",&saldo);

  while (fscanf(fp,"%d%*c%d%f%*c%c",&ris.giorno,&ris.mese,&movimento,&segno) != EOF)
    {
      if (segno == '+')
        saldo += movimento;
      else
        {
          saldo -= movimento;
          if (saldo < limite)
            break;
        }
    }
  fclose(fp);
  return ris;
}
```

Soluzione esercizio 2

La soluzione si compone dei seguenti passi:

1. Si memorizza la matrice integralmente in memoria centrale (variabile `mat`)
2. Si calcola il numero di righe della nuova matrice (variabile `n1`)
3. Si scandisce la matrice per righe all'inverso per riscriverla sul file.

Si noti che i passi 2 e 3 non possono essere fatti insieme perché il nuovo numero di righe deve essere scritto in cima al file (quindi prima del passo 3).

```
#include <stdio.h>

#define MAX_RIGHE 100
#define MAX_COLONNE 200

void InvertiRigheMatrice(char nome_file[]);

/* non richiesto per il compito in classe */
int main()
{
  char nome[21];

  printf("Nome del file contenente la matrice : ");
  scanf("%s",nome);
  InvertiRigheMatrice(nome);
  return 0;
}
```

```
void InvertiRigheMatrice(char nome_file[])
{
  FILE* fp;
  int mat[MAX_RIGHE][MAX_COLONNE], n, n1, m, i, j;

  fp = fopen(nome_file,"r");
  fscanf(fp,"%d%*c%d",&n,&m);
  for (i = 0; i < n; i++)
    for (j = 0; j < m; j++)
      fscanf(fp, "%d", &mat[i][j]);
  n1 = n;
  for (i = 0; i < n; i++)
    {
      if (mat[i][0] == 0)
        n1--;
    }
  fclose(fp);

  fp = fopen(nome_file,"w");
  fprintf(fp, "%dX%d\n", n1, m);
  for (i = n-1; i >= 0; i--)
    {
      if (mat[i][0] != 0)
        {
          for (j = 0; j < m; j++)
            fprintf(fp, "%3d ",mat[i][j]);
          fprintf(fp, "\n");
        }
    }
  fclose(fp);
}
```

Soluzione compito del 8 aprile 2002

Soluzione esercizio 1

Visto che la presenza di almeno uno spazio tra l'operatore e il secondo operando non è garantita, la lettura utilizzando il formato `%d%s%d` (scelta da alcuni studenti in aula) non funziona. Infatti, se tale spazio è assente il secondo operando viene letto insieme all'operatore.

Una possibile soluzione è di utilizzare una lettura per caratteri per trovare l'operatore, leggendo invece gli operandi tranquillamente con la `fscanf()`.

Un'alternativa pù semplice, che utilizziamo nella nostra soluzione, sfrutta il fatto che gli operatori sono tutti formati da un solo carattere. In base a questo, possiamo usare per la lettura dell'operatore il formato `%1s` che legge una stringa di lunghezza massima 1.

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

int SommaOperazioni(char nome_file[]);

/* non richiesto per il compito in classe */
int main(int argc, char *argv[])
{
  if (argc < 2)
    {
      printf("Errore, numero di argomenti insufficiente\n");
      exit(1);
```

```
    }
  else
    printf("La somma delle operazioni contenute nel file e' %d\n", SommaOperazioni(argv[1]));
  return 0;
}

int SommaOperazioni(char nome_file[])
{
  int operando1, operando2, somma = 0;
  char op[2];
  FILE* fp;

  fp = fopen(nome_file, "r");
  while (fscanf(fp, "%d%1s%d", &operando1, op, &operando2) != EOF)
    {
      switch (op[0])
        {
        case '+':
          somma += operando1 + operando2;
          break;
        case '-':
          somma += operando1 - operando2;
          break;
        case '*':
          somma += operando1 * operando2;
          break;
        default:
          printf("Errore, operatore %s non corretto", op);
          exit(1);
          break;
        }
    }
  return somma;
}
```

Soluzione esercizio 2

L'esercizio non necessita di particolari commenti. Si noti solo che la media arrotondata per eccesso si ottiene aggiungendo uno alla somma dei voti e sfruttando la divisione tra interi.

```
#include <stdio.h>
#include <stdlib.h> /* serve per la funzione exit */
#include <string.h>

void CalcolaMedie(char nome_file1[], char nome_file2[], char nome_file_output[]);

/* non richiesto per il compito in classe */
int main(int argc, char *argv[])
{
  if (argc < 4)
    {
      printf("Errore, numero di argomenti insufficiente\n");
      exit(-1);
    }
  else
    CalcolaMedie(argv[1],argv[2],argv[3]);
  return 0;
}

void CalcolaMedie(char nome_file1[], char nome_file2[], char nome_file_output[])
```

```
{
  FILE *fp1, *fp2, *fp_out;
  char cognome[20], nome[20];
  int voto1, voto2, media;

  fp1 = fopen(nome_file1, "r");
  fp2 = fopen(nome_file2, "r");
  fp_out = fopen(nome_file_output, "w");

  if (fp1 == NULL || fp2 == NULL || nome_file_output == NULL)
    {
      printf("Errore, non posso aprire qualche file\n");
      exit(1);
    }
  while (fscanf(fp1,"%s%s%*s%d",cognome,nome,&voto1) != EOF)
    {
      nome[strlen(nome)-1] = '\0'; /* toglie la virgola al nome */
      fscanf(fp2,"%*s%*s%*s%d",&voto2);
      media = (voto1 + voto2 + 1)/ 2; /* arrotondamento per eccesso */
      if (media >= 18)
        fprintf(fp_out, "%s %s: %d\n", nome, cognome, media);
    }
}
```

Soluzione compito del 20 giugno 2002

Soluzione esercizio 1

La funzione si compone di una semplice lettura del file riga per riga. Per ciascuna riga vengono scanditi i servizi alla ricerca di quello passato per parametro. Si noti che per contare le stelle dell'albergo è sufficiente sottrarre 3 alla lunghezza della stringa che le contiene (le due parentesi + la virgola), senza bisogno di contare effettivamente il numero di ripetizioni del carattere `'*'`.

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

int NumeroAlberghi(char nome_file[], char servizio[], int stelle);

/* non richiesto per il compito in classe */
int main(int argc, char *argv[])
{
  if (argc != 4)
    {
      printf("Errore, numero di argomenti sbagliato\n");
      exit(1);
    }
  else
      printf("Il numero di alberghi e' %d\n", NumeroAlberghi(argv[1],argv[2],atoi(argv[3])));
  return 0;
}

int NumeroAlberghi(char nome_file[], char servizio[], int stelle)
{
  FILE *fp;
  char nome_albergo[21], stringa_stelle[8], servizio_albergo[31];
  int stelle_albergo, numero_servizi, i, conta_alberghi = 0;
```

```
  fp = fopen(nome_file,"r");
  while (fscanf(fp,"%s",nome_albergo) != EOF)
  {
    fscanf(fp,"%s%d", stringa_stelle, &numero_servizi);
    stelle_albergo = strlen(stringa_stelle) - 3; /* -3: non conta le
                                                    parentesi e la virgola */
    if (stelle_albergo < stelle)
      while (getc(fp) != '\n') /* ignora il resto della riga */
        ;
    else
      for (i = 0; i < numero_servizi; i++)
        {
          fscanf(fp,"%s",servizio_albergo);
          if (strcmp(servizio_albergo, servizio) == 0)
            conta_alberghi++;
        }
  }
  return conta_alberghi;
}
```

Soluzione esercizio 2

Questo esercizio si risolve facilmente con una buona modularizzazione. In particolare, si suddivide il problema di verificare se una matrice è un quadrato magico in quattro verifiche distinti e separate: somma orizzontale, somma verticale, somma diagonale, presenza di tutti i numeri. Ciascuna verifica è gestita da una funzione specifica, che riceve come parametro la matrice e restituisce 0 o 1. La funzione principale si riduce ad un *and* tra i risultati delle funzioni ausiliarie.

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

#define N 3
#define MAGIC (N * (N * N + 1)/2)



int QuadratoMagico(int mat[][N]);
int VerificaOrizzontale(int mat[][N]);
int VerificaVerticale(int mat[][N]);
int VerificaDiagonale(int mat[][N]);
int TuttiPresenti(int mat[][N]);
int Presente(int m[][N], int e);

/* non richiesto per il compito in classe */
int main()
{
  int m1[N][N] = {{8,3,4},{1,5,9},{6,7,2}};
  int m2[N][N] = {{3,8,4},{9,5,1},{6,7,2}};
  int m3[N][N] = {{1,5,9},{8,3,4},{6,7,2}};
  int m4[N][N] = {{5,5,5},{5,5,5},{5,5,5}};

  if (QuadratoMagico(m1))
    printf("M1 e' magico\n");
  if (QuadratoMagico(m2))
    printf("M2 e' magico\n");
  if (QuadratoMagico(m3))
    printf("M3 e' magico\n");
  if (QuadratoMagico(m4))
    printf("M4 e' magico\n");
  return 0;
```

```
}


int QuadratoMagico(int mat[][N])
{
  return
    VerificaOrizzontale(mat) &&
    VerificaVerticale(mat) &&
    VerificaDiagonale(mat) &&
    TuttiPresenti(mat);
}

int VerificaOrizzontale(int mat[][N])
{
  int i, j, somma;

  for (i = 0; i < N; i++)
    {
      somma = 0;
      for (j = 0; j < N; j++)
        somma += mat[i][j];
      if (somma != MAGIC)
        return 0;
    }
  return 1;
}

int VerificaVerticale(int mat[][N])
{
  int i, j, somma;

  for (j = 0; j < N; j++)
    {
      somma = 0;
      for (i = 0; i < N; i++)
        somma += mat[i][j];
      if (somma != MAGIC)
        return 0;
    }
  return 1;
}

int VerificaDiagonale(int mat[][N])
{
  int i, somma1 = 0, somma2 = 0;

  for (i = 0; i < N; i++)
    {
      somma1 += mat[i][i];
      somma2 += mat[i][N-i-1];
    }
  return (somma1 == MAGIC && somma2 == MAGIC);
}

int TuttiPresenti(int mat[][N])
{
  int i;
  for (i = 1; i <= N*N; i++)
    if (!Presente(mat,i))
```

```
      return 0;
  return 1;
}

int Presente(int m[][N], int e)
{
  int i, j;
  for (i = 0; i < N; i++)
    for (j = 0; j < N; j++)
      if (m[i][j] == e)
        return 1;
  return 0;
}
```

Soluzione compito del 15 luglio 2002

Soluzione esercizio 1

La funzione si basa su tre cicli consecutivi di scansione della stringa di ingresso in cui vengono scritte nella nuova stringa i tre tipi di caratteri (vocali, consonanti, altro) della stringa.

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

#define MAX_LEN 30

void CopiaOrdinato(char s1[], char s2[]);
int Vocale(char ch);

/* non richiesto per il compito in classe */
int main()
{
  char s[MAX_LEN+1], t[MAX_LEN+1];
  int i = 0;

  printf("Inserisci una stringa (massimo %d caratteri) terminata da \"invio\" (\\n) :", MAX_LEN);
  while ((s[i] = getchar()) != '\n')
    i++;
  s[i] = '\0';
  CopiaOrdinato(s,t);
  printf("La stringa ordinata e' : %s\n",t);
  return 0;
}

void CopiaOrdinato(char s1[], char s2[])
{
  int i, j = 0, n = strlen(s1);
  printf("%d\n",n);

  for (i = 0; i < n; i++)
    if (Vocale(s1[i]))
      {
        s2[j] = s1[i];
        putchar(s2[j]);
        j++;
      }

  for (i = 0; i < n; i++)
```

```
    if (isalpha(s1[i]) && !Vocale(s1[i]))
      {
        s2[j] = s1[i];
        j++;
      }

  for (i = 0; i < n; i++)
    if (!isalpha(s1[i]))
      {
        s2[j] = s1[i];
        j++;
      }
  s2[j] = '\0';
}


int Vocale(char ch)
{
  return (ch == 'a' || ch == 'e' || ch == 'i' || ch == 'o' || ch == 'u'
          || ch == 'A' || ch == 'E' || ch == 'I' || ch == 'O' || ch == 'U');
}
```

Soluzione esercizio 2

Per restituire il risultato della funzione è necessario definire un record specifico, chiamato `struct Stat`, a quattro campi che contenga tutti i valori richiesti.

Abbiamo scelto di attuare una soluzione che leggesse il file una sola volta, e che calcolasse quindi contemporaneamente tutte le quantità richieste. Essendo una di queste il numero totale di caratteri, è indispensabile eseguire una lettura per carattere.

Per contare le parole e cercare la parola più lunga è necessario mantenere una variabile booleana, chiamata `in_parola`, che ci dice se il carattere letto si trova all'interno di una parola oppure no. Il valore di `in_parola` cambia in funzione del carattere letto e del suo valore attuale. Ad esempio, se viene letto un carattere di spaziatura e `in_parola` vale `1` significa che la parola corrente è finita. E quindi necessario contarla (`ris.parole++`) e confrontare la sua lunghezza con la massima lunghezza trovata fino a quel punto.

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

struct Stat
{
  int caratteri;
  int parole;
  int righe;
  int max_lunghezza;
};

struct Stat StatisticheFile(char nome_file[]);

/* non richiesto per il compito in classe */
int main()
{
  char nome_file[21];
  struct Stat dati;

  printf("Nome del file di testo : ");
  scanf("%s",nome_file);
  dati = StatisticheFile(nome_file);
  printf("Caratteri: %d\nParole : %d\nRighe : %d\nParola piu' lunga : %d\n",
         dati.caratteri, dati.parole, dati.righe, dati.max_lunghezza);
```

```
  return 0;
}

struct Stat StatisticheFile(char nome_file[])
{
  FILE *fp;
  char ch;
  struct Stat ris = {0,0,0,0};
  int in_parola = 0; /* booleano: vale 1 se si e' all'interno di una parola */
  int lunghezza_parola = 0;


  fp = fopen(nome_file,"r");
  while ((ch = getc(fp)) != EOF)
    {
      ris.caratteri++;
      if (ch == ' ' || ch == '\n')
        {
          if (in_parola)
            { /* parola finita */
              ris.parole++;
              if (lunghezza_parola > ris.max_lunghezza)
                ris.max_lunghezza = lunghezza_parola;
              lunghezza_parola = 0;
            }
          in_parola = 0;
          if (ch == '\n')
            ris.righe++;
        }
      else
        { /* ch non e' un carattere di spaziatura */
          lunghezza_parola++;
          /* in_parola viene posto ad 1 indipendentemente dal fatto che
             fosse precedentemente a 0 oppure ad 1 */
          in_parola = 1;
        }
    }
  return ris;
}
```

Soluzione compito del 9 settembre 2002

Soluzione esercizio 1

Il file deve necessariamente essere letto per caratteri (e non per stringhe), al fine di evitare di perdere i caratteri di spaziatura. Quando si incontra il carattere `'['`, si legge un carattere (la lettera) e si cerca il valore corrispondente nel vettore.

```
#include <stdio.h>

struct Corrispondenza
{
  char lettera;
  int numero;
};

void InserisciNumeri(char nome_file_input[], char nome_file_output[],
```

```
                    struct Corrispondenza vet[], int n, int k);
int ValoreLettera(char ch, struct Corrispondenza vet[], int n, int k);

/* non richiesto per il compito in classe */
int main()
{
  struct Corrispondenza V[5] = {{'A',9}, {'C',17}, {'G',78}, {'B',12}, {'P',6}};
  InserisciNumeri("input.txt","output.txt",V,5,25);
  return 0;
}


void InserisciNumeri(char nome_file_input[], char nome_file_output[],
                    struct Corrispondenza vet[], int n, int k)
{
  FILE* ifp;
  FILE* ofp;
  char ch, lettera;

  ifp = fopen(nome_file_input,"r");
  ofp = fopen(nome_file_output,"w");
  while ((ch = getc(ifp)) != EOF)
  {
    if (ch != '[')
      fprintf(ofp,"%c",ch);
    else
      {
        lettera = getc(ifp);
        fprintf(ofp, "%d", ValoreLettera(lettera,vet,n,k));
        getc(ifp);  /* consuma il carattere ] */
      }
  }
  fclose(ifp);
  fclose(ofp);
}

int ValoreLettera(char ch, struct Corrispondenza vet[], int n, int k)
{
  int i;
  for (i = 0; i < n; i++)
    if (vet[i].lettera == ch)
      return vet[i].numero;
  return k;
}
```

Soluzione esercizio 2

La funzione utilizza una variabile `d` di tipo `Data` a cui viene inizialmente dato il valore del primo gennaio dell'anno passato come parametro.

Un primo ciclo modifica `d` fino a fargli assumere il valore della prima domenica dell'anno, sfruttando la funzione `DataSuccessiva()` e utilizzando il parametro `giorno_iniziale`.

Successivamente, si esegue un doppio ciclo in cui ad ogni iterazione del ciclo esterno si trova una nuova domenica, mentre il ciclo interno esegue semplicemente `DataSuccessiva()` per 7 volte. Il ciclo esterno termina quando si raggiunge una domenica dell'anno successivo.

```
#include <stdio.h>

struct Data
{
  int giorno;
```

```
  int mese;
  int anno;
};

int Bisestile(int);
int GiorniDelMese(int, int);
struct Data DataSuccessiva(struct Data);
void StampaDomeniche(int anno, int giorno_iniziale);

/* non richiesto per il compito in classe */
int main()
{
  int a, g;
  printf("Anno : ");
  scanf("%d", &a);
  printf("Giorno del 1 gennaio del %d : ", a);
  scanf("%d", &g);
  StampaDomeniche(a,g);
  return 0;
}

void StampaDomeniche(int anno, int giorno_iniziale)
{
  int i, j;
  struct Data d;

  d.anno = anno;
  d.mese = 1;
  d.giorno = 1;
  for (i = 0; i < 7-giorno_iniziale; i++)
    d = DataSuccessiva(d);
  while (d.anno == anno)
    {
      printf("%d/%d/%d ", d.giorno, d.mese, d.anno);
      for (j = 0; j < 7; j++)
        d = DataSuccessiva(d);
    }
}


struct Data DataSuccessiva(struct Data d)
{
  struct Data ris;
  if (d.giorno != GiorniDelMese(d.mese,d.anno))
    {
      ris.anno = d.anno;
      ris.mese = d.mese;
      ris.giorno = d.giorno + 1;
    }
  else
    if (d.mese != 12)
      {
        ris.anno = d.anno;
        ris.mese = d.mese + 1;
        ris.giorno = 1;
      }
    else
      {
        ris.anno = d.anno + 1;
```

```
        ris.mese = 1;
        ris.giorno = 1;
      }
  return ris;
}

int GiorniDelMese(int mese, int anno)
{
  if (mese == 4 || mese == 6 || mese == 9 || mese == 11)
    return 30;
  else if (mese == 2)
    if (Bisestile(anno))
      return 29;
    else
      return 28;
  else
    return 31;
}

int Bisestile(int a)
{
  if (a % 4 != 0)
    return 0;
  else if (a % 100 != 0)
    return 1;
  else if (a % 400 != 0)
    return 0;
  else
    return 1;
}
```

Soluzione compito del 25 settembre 2002

Soluzione esercizio 1

```
#include <stdio.h>
#include <string.h>
#define DIM 5

struct Autore
{
  char nome[20];
  int dato;
};

int AggiungiDato(struct Autore v[], int n, struct Autore r);

/* non richiesto per il compito in classe */
int main()
{
  struct Autore a;
  struct Autore vet[DIM] = {{"Silone", 3}, {"Pirandello",49}, {"Eco", 25},
                            {"D'annunzio",32}, {"Calvino", 18}};
  int i, ris;

  printf("Autore : ");
  scanf("%s",a.nome);
  printf("Dato : ");
  scanf("%d",&a.dato);
```

```
  ris = AggiungiDato(vet,DIM,a);
  if (ris)
    {
      printf("Dato inserito. Nuovo vettore:\n");
      for (i = 0; i < DIM; i++)
        printf("%-20s %2d\n", vet[i].nome, vet[i].dato);
      /* si notino i caratteri di specifica di formato della printf
         utilizzati (non in programma d'esame): i valori 20 e 2 nei
         formati %s e %d rispettivamente determinano la stampa del dato
         con 20 e 2 caratteri, aggiungento dei black; il segno meno nel
         formato %s fa si' che la stringa sia giustificata a sinistra
      */
    }
  else
    printf("Autore non presente\n");
  return 0;
}

int AggiungiDato(struct Autore v[], int n, struct Autore r)
{
  int i;
  for (i = 0; i < n; i++)
    if (strcmp(v[i].nome, r.nome) == 0)
      {
        v[i].dato += r.dato;
        return 1;
      }
  return 0;
}
```

Soluzione esercizio 2

Il primo passo della funzione è quello di memorizzare in una struttura dati in memoria centrale (un vettore di record) il contenuto del file delle abbreviazioni. Questo lavoro è fatto dalla funzione ausiliaria `LeggiFileSostituzioni()`.

Successivamente si esegue in un ciclo che itera su tutti i caratteri della stringa di partenza. Per ogni carattere si scandisce il vettore delle abbreviazioni e si verifica se una di esse è applicabile. Se se ne trova una applicabile si pone la variabile `trovato` al valore `1`. Non ci poniamo il problema di cosa fare quando più di una sostituzione è applicabile; nel nostro caso viene applicata semplicemente la prima che viene trovata.

Alla fine della scansione, se `trovato` vale `1`, allora si scrive nella stringa di uscito la stringa abbreviata e si va avanti di un numero di caratteri corrispondente alla lunghezza delle stringhe. Se invece nessuna abbreviazione è applicabile da quel punto si scrive il singolo carattere e si va avanti di uno.

Si noti che l'indice della stringa di ingresso e quello della stringa di uscita sono distinti (variabili `j` e `k`). Questo perché a seguito di una sostituzione la lunghezza delle stringhe si differenzia.

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

#define MAXSTRLEN     32
#define MAX_COPPIE  64

struct CoppiaStringhe
{
   char iniziale[MAXSTRLEN];
   char finale[MAXSTRLEN];
};
```

```
int LeggiFileSostituzioni(char nome_file[], struct CoppiaStringhe v[]);
void ComprimiSMS(char sms_in[], char sms_out[], char nome_file[]);

int main(int argc, char* argv[])
{/* non richiesto per il compito in classe */
   char s[64];

   if (argc != 2)
   {
      printf ("numero di parametri errato\n");
      exit (EXIT_SUCCESS);
   }
   ComprimiSMS(argv[1], s, "Compressioni.dat");
   printf ("SMS compresso: %s\n", s);
   return 0;
}



void ComprimiSMS(char sms_in[], char sms_out[], char nome_file[])
{
   struct CoppiaStringhe vet[MAX_COPPIE];
   int n, i, j = 0, k = 0, trovato;

   n = LeggiFileSostituzioni(nome_file, vet);

   sms_out[0] = '\0';
   while (sms_in[j] != '\0')
   {
      i = 0;
      trovato = 0;
      while (!trovato && i < n)
        {
          if (strncmp(sms_in + j, vet[i].iniziale,strlen(vet[i].iniziale)) == 0)
            trovato = 1;
          else
            i++;
        }

      if (trovato)
        {
          strcat(sms_out, vet[i].finale);
          j += strlen(vet[i].iniziale);
          k += strlen(vet[i].finale);
        }
      else
        {
          sms_out[k] = sms_in[j];
          k++;
          j++;
          sms_out[k] = '\0';
        }
   }
}


int LeggiFileSostituzioni(char nome_file[], struct CoppiaStringhe v[])
{
   FILE *fin;
```

```
   int i = 0;

   fin = fopen(nome_file, "r");
   while (fscanf(fin, "%s%s", v[i].iniziale, v[i].finale) != EOF)
      i++;
   return i;
}
```

Soluzione compito del 16 dicembre 2002

Soluzione esercizio 1

La soluzione si basa su un semplice ciclo di lettura da file riga per riga e dell'analisi di ciascuna riga durante la lettura.

L'unico aspetto interessante è la lettura del numero di falli. Non essendo fisso il numero di spazi prima del carattere `'['` sembrerebbe necessario utilizzare un ciclo di lettura per carattere. Utilizziamo qui però una caratteristica del formato `%s` della funzione `fscanf()` che ci semplifica il compito: inserendo un intero positivo tra `%` ed `s` si specifica il numero massimo di caratteri che devono essere letti[1] . Quindi la specifica `%1s` legge il singolo carattere (`[` nel nostro caso) e permette alla successiva specifica `%s` di "catturare" il numero di falli (scritto a lettere).

Per una buona modularità, il numero di falli viene convertito in intero da una funzione ausiliaria.

```
#include <stdio.h>
#include <string.h>
#define ALTEZZA_MINIMA 190

float MediaFalli(char nome_file[], float percentuale_richiesta);
int ConvertiLettereNumero(char numero[]);

/* non richiesto per il compito in classe */
int main()
{
  char nome[] = "giocatori.txt";
  float p;

  printf("Percentuale minima : ");
  scanf("%g",&p);
  printf("Il numero medio di falli nel file %s e' %g\n", nome, MediaFalli(nome,p));
  return 0;
}

float MediaFalli(char nome_file[], float percentuale_richiesta)
{
  FILE* fp;
  char falli[8]; /* massima lunghezza: "quattro" (7+1 caratteri) */
  int altezza, num_giocatori = 0, somma_falli = 0;
  float percentuale;

  fp = fopen(nome_file,"r");
  while (fscanf(fp,"%*s%d%g%*1s%s%*s", &altezza, &percentuale, falli) != EOF)
         /* il formato %*1s legge e ignora una stringa di 1 carattere
            significativo: quindi porta il cursore oltre il carattere [ */
  {
    if (altezza >= ALTEZZA_MINIMA && percentuale > percentuale_richiesta)
      {
        num_giocatori++;
        somma_falli += ConvertiLettereNumero(falli);
      }
```

[1] Vedere anche la soluzione all'esercizio 1 del compito del 4 aprile 2002 (pagina 76)

```
  }
  fclose(fp);
  return (float)somma_falli/ num_giocatori;
}

int ConvertiLettereNumero(char numero[])
{
  if (strcmp(numero,"zero") == 0)
    return 0;
  else if (strcmp(numero,"un") == 0)
    return 1;
  else if (strcmp(numero,"due") == 0)
    return 2;
  else if (strcmp(numero,"tre") == 0)
    return 3;
  else if (strcmp(numero,"quattro") == 0)
    return 4;
  else /* strcmp(numero,"cinque") == 0) */
    return 5;

}
```

Soluzione esercizio 2

Partendo dalla cifra meno significativa (quella in posizione `n-1`) si esegue la somma in binario cifra per cifra, sommando anche il riporto dalla cifra precedente. Inizialmente il riporto è 0, mentre l'ultimo riporto corrisponde alla condizione di *overflow*.

Si noti che il passaggio dai caratteri numerici ai numeri corrispondenti (o viceversa) si ottiene sottraendo (o sommando) il codice ASCII di `0`, cioè il valore di `'0'`.

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

#define MAX_LEN 30

int SommaStringheBinarie(char s1[], char s2[], char ris[]);

/* non richiesto per il compito in classe */
int main()
{
  char r[MAX_LEN+1], s[MAX_LEN+1], t[MAX_LEN+1];
  int overflow;

  printf("Inserisci la prima stringa (massimo %d cifre) : ", MAX_LEN);
  scanf("%s",r);
  printf("Inserisci la seconda stringa (%d cifre) : ", strlen(r));
  scanf("%s",s);
  overflow = SommaStringheBinarie(r,s,t);
  printf("Il risultato e' %s", t);
  if (overflow)
    printf(" (con overflow)\n");
  else
    printf(" (senza overflow)\n");
  return 0;
}

int SommaStringheBinarie(char s1[], char s2[], char ris[])
{
  int riporto = 0, i, somma;
  int n = strlen(s1);
```

```
  for (i = n-1; i >=0; i--)
    {
      somma = s1[i] - '0' + s2[i] - '0' + riporto;
      riporto = somma/2;
      ris[i] = (somma % 2) + '0';
    }
  return riporto;
}
```

Soluzione compito del 19 marzo 2003

Soluzione esercizio 1

Questo esercizio, come molti altri, si risolve con una semplice lettura e processamento del file di ingresso riga per riga. Come suggerito, si utilizza una funzione chiamata `ComparaOrari()` che verifica se un orario è maggiore di un altro.

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

struct Orario
{
  int giorno;
  int mese;
  int anno;
  int ora;
};

void SelezionaAppuntamenti(char nome_file_input[], char nome_file_output[],
                           struct Orario d);

int ComparaOrari(struct Orario o1, struct Orario o2);

/* non richiesto per il compito in classe */
int main(int argc, char* argv[])
{
  struct Orario ora_soglia;

  if (argc != 3)
    {
      printf("Inserire i nomi dei due file sulla riga di comando!\n");
      exit(1);
    }
  printf("Inserisci giorno, mese, anno e ora (separati da spazi): ");
  scanf("%d%d%d%d",&ora_soglia.giorno, &ora_soglia.mese,
        &ora_soglia.anno,&ora_soglia.ora);
  SelezionaAppuntamenti(argv[1],argv[2],ora_soglia);
  return 0;
}


void SelezionaAppuntamenti(char nome_file_input[], char nome_file_output[],
                           struct Orario d)
{
  struct Orario ore;
  char descrizione[21], luogo[11];

  FILE *fp1, *fp2;
```

```
  fp1 = fopen(nome_file_input,"r");
  fp2 = fopen(nome_file_output,"w");

  while (fscanf(fp1,"%d%*c%d%*c%d%*s%d%s%s", &ore.giorno, & ore.mese,
                &ore.anno, &ore.ora, descrizione, luogo) != EOF)
    if (ComparaOrari(ore,d))
      fprintf(fp2,"%d-%d-%d ore %d %s %s\n", ore.giorno, ore.mese, ore.anno,
              ore.ora, descrizione, luogo);

  fclose(fp1);
  fclose(fp2);
}

int ComparaOrari(struct Orario o1, struct Orario o2)
{
  return (o1.anno > o1.anno)
    ||   (o1.anno == o2.anno && o1.mese > o2.mese)
    ||   (o1.anno == o2.anno && o1.mese == o2.mese && o1.giorno > o2.giorno)
    ||   (o1.anno == o2.anno && o1.mese == o2.mese
          && o1.giorno == o2.giorno && o1.ora > o2.ora);
}
```

Soluzione esercizio 2

La soluzione concettualmente più semplice e più generale dell'esercizio fa uso di una matrice in memoria centrale. Di contro, il passaggio dal file di ingresso al file di uscita senza utilizzare la matrice è possibile solo nell'ipotesi che il file di ingresso sia ordinato (per righe e per colonne), e comunque è più complessa da implementare.

La matrice viene inizialmente riempita con tutti valori uguali al valore dominante, e successivamente aggiornata con i valori provenienti dal file di ingresso. Infine, la matrice viene scritta sul file di uscita.

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

#define MAX_DIM 100

void TrasformaMatrice(char nome_file[]);

/* non richiesto per il compito in classe */
int main(int argc, char* argv[])
{
  if (argc != 2)
    {
      printf("Inserire il nome del file di input sulla riga di comando!\n");
      exit(1);
    }
  TrasformaMatrice(argv[1]);
  return 0;
}

void TrasformaMatrice(char nome_file[])
{
  FILE *fp1, *fp2;
  int m[MAX_DIM][MAX_DIM];
  int i, j, dom, righe, colonne, val;

  fp1 = fopen(nome_file,"r");
  fp2 = fopen("matrice.dat","w");
```

```
  fscanf(fp1,"%d%d%d",&righe,&colonne,&dom);
  for (i = 0; i < righe; i++)
    for (j = 0; j < colonne; j++)
      m[i][j] = dom;

  while (fscanf(fp1,"%d%d%d", &i, &j, &val) != EOF)
    m[i][j] = val;

  for (i = 0; i < righe; i++)
    {
      for (j = 0; j < colonne; j++)
        fprintf(fp2,"%d ", m[i][j]);
      fprintf(fp2,"\n");
    }

  fclose(fp1);
  fclose(fp2);
}
```

Soluzione compito del 31 marzo 2003

Soluzione esercizio 1

Per risolvere questo esercizio è necessario tenere aperti due file contemporaneamente: il file indice da cui si leggono i nomi ed il file di dati corrente. Serve utilizzare quindi due variabili di tipo `FILE*`. Una delle due variabili viene utilizzata per aprire tutti i file di dati, ovviamente in sequenza e non contemporaneamente.

Il problema di estrarre dal nome il numero in esso inserito, si risolve trasformando i caratteri numerici in numeri e sommando le due cifre, avendo moltiplicato la prima per 10.

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

int SommaFileSelezionati(char nome_file_indice[], int k);

int main(int argc, char* argv[])
{ /* non richiesto per il compito in classe */
  int soglia, somma;
  if (argc != 3)
    {
      printf("Inserire file indice e valore soglia sulla riga di comando\n");
      exit(1);
    }
  soglia = atoi(argv[2]);
  somma = SommaFileSelezionati(argv[1],soglia);
  printf("La somma e' %d\n", somma);
  return 0;
}

int SommaFileSelezionati(char nome_file_indice[], int k)
{
  char nome_file[11];
  FILE* fp1, *fp2;
  int numero_file, val, somma = 0;

  fp1 = fopen(nome_file_indice,"r");
```

```
  while(fscanf(fp1,"%s",nome_file) != EOF)
    {
      numero_file = (nome_file[4] - '0') * 10 + (nome_file[5] - '0');
      if (k > numero_file)
        {
          fp2 = fopen(nome_file,"r");
          while (fscanf(fp2,"%d",&val) != EOF)
            somma += val;
          fclose(fp2);
        }
    }
  fclose(fp1);
  return somma;
}
```

Soluzione esercizio 2

La funzione legge il file riga per riga immagazzinando i giorni di servizio di ciascun volo in una singola stringa. All'interno del ciclo, un altro ciclo verifica se il giorno cercato è presente o meno all'interno della stringa.

Per restituire i dati richiesti si utilizza un tipo record definito specificamente per l'esercizio.

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

struct Volo
{
  char codice_compagnia[3];
  int numero_volo;
};

struct Volo CercaVolo(char nome_file[], char destinazione[], int giorno);

/* non richiesto per il compito in classe */
int main()
{
  char nome_file[21], destinazione[4];
  int giorno;
  struct Volo risultato;
  printf("Immetti il nome del file voli: ");
  scanf("%s", nome_file);
  printf("Immetti la destinazione cercata (3 caratteri): ");
  scanf("%s", destinazione);
  printf("Immetti il giorno di partenza (1 = lunedi', 2 = martedi', ..., 7 = domenica): ");
  scanf("%d", &giorno);
  risultato = CercaVolo(nome_file, destinazione, giorno);
  if (risultato.numero_volo == -1)
    printf("Non ho trovato nessun volo\n");
  else
    printf("Ho trovato il volo %s %d\n", risultato.codice_compagnia, risultato.numero_volo);
  return 0;
}

struct Volo CercaVolo(char nome_file[], char destinazione[], int giorno)
{
  FILE* fp;
  char dest[4], codice[3], profilo[8];
  int numero, i;
  struct Volo v;
```

```
  fp = fopen(nome_file, "r");
  if (fp == NULL)
    {
      v.codice_compagnia[0] = '\0';
      v.numero_volo = -1;
      return v;
    }
  while (fscanf(fp, "%s %s %d %s", dest, codice, &numero, profilo) != EOF)
    {
      if (strcmp(dest, destinazione) == 0)
        for (i = 0; profilo[i] != '\0'; i++)
          if (profilo[i] - '0' == giorno)
            {
              strcpy(v.codice_compagnia, codice);
              v.numero_volo = numero;
              return v;
            }
    }
  fclose(fp);
  v.codice_compagnia[0] = '\0';
  v.numero_volo = -1;
  return v;
}
```

Soluzione compito del 30 giugno 2003

Soluzione esercizio 1

Come suggerito dal testo, tutto il contenuto del file viene memorizzato in un'opportuna struttura di dati in memoria centrale.

Successivamente, dopo la chiusura e riapertura del file in scrittura, il contenuto della struttura di dati viene scritto sul file di uscita. Per ogni riga si aggiunge il segno, che si ottiene semplicemente confrontando i due campi corrispondenti ai gol segnati.

```
#include <stdio.h>
#include <stdlib.h>
#define MAX_PARTITE 20

struct Partita
{
  char squadra_di_casa[20];
  char squadra_ospite[20];
  int goal_casa;
  int goal_ospiti;
};

void AggiungiSegno(char nome_file[]);

int main(int argc, char* argv[])
{ /* non richiesto per il compito in classe */
  if (argc != 2)
    {
      printf("Inserire il nome del file\n");
      exit(1);
    }
  AggiungiSegno(argv[1]);
  return 0;
}
```

```
void AggiungiSegno(char nome_file[])
{
  FILE* fp;
  struct Partita v[MAX_PARTITE];
  int n, i = 0;

  fp = fopen(nome_file,"r");

  while(fscanf(fp,"%s%*s%s%d%*c%d",v[i].squadra_di_casa,v[i].squadra_ospite,
               &v[i].goal_casa, &v[i].goal_ospiti) != EOF)
    i++;
  n = i;

  fclose(fp);
  fp = fopen(nome_file,"w");

  for (i = 0; i < n; i++)
    {
      fprintf(fp, "%s - %s  %d-%d  ", v[i].squadra_di_casa,v[i].squadra_ospite,
               v[i].goal_casa, v[i].goal_ospiti);
      if (v[i].goal_casa > v[i].goal_ospiti)
        fprintf(fp, "1");
      else if (v[i].goal_casa == v[i].goal_ospiti)
        fprintf(fp, "X");
      else
        fprintf(fp, "2");
      fprintf(fp, "\n");
    }
}
```

Soluzione esercizio 2

Definiamo una funzione ausiliaria per contare i pezzi di un dato tipo sulla scacchiera. Utilizzando questa funzione si verificano il secondo e il terzo caso di illegalità di una posizione specificati nel testo. Il primo caso viene invece gestito da una funzione specifica.

Si utilizza inoltre una funzione, chiamata `Punti()`, che dato un carattere restituisce il valore del pezzo rappresentato. Per semplicità questa funzione accetta solo i caratteri minuscoli (i pezzi bianchi). Per i pezzi neri, si utilizza preventivamente la funzione `tolower()` che restituisce il corrispettivo minuscolo del carattere passatogli.

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

void LeggiScacchiera(char s[8][8]);
void StampaScacchiera(char s[8][8]);
int ValutaPosizione(char s[8][8]);
int ContaPezzo(char s[8][8], char pezzo);
int PedoniFuoriPosto(char s[8][8]);
int Punti(char c);

/* non richiesto per il compito in classe */
int main()
{
  char s[8][8];

  LeggiScacchiera(s);
  StampaScacchiera(s);
  switch (ValutaPosizione(s))
    {
    case -1:
```

```
      printf("Posizione non valida\n");
      break;
    case 0:
      printf("Posizione di parita'\n");
      break;
    case 1:
      printf("Posizione di vantaggio bianco\n");
      break;
    case 2:
      printf("Posizione di vantaggio nero\n");
      break;
    }
  return 0;
}

void LeggiScacchiera(char s[8][8])
{
  FILE* fp;
  int i, j;

  fp = fopen("scacchiera.txt","r");
  for (i = 0; i < 8; i++)
    {
      for (j = 0; j < 8; j++)
        s[i][j] = getc(fp);
      getc(fp);
    }
}

void StampaScacchiera(char s[8][8])
{
  int i, j;
  for (i = 0; i < 8; i++)
    {
      for (j = 0; j < 8; j++)
        putchar(s[i][j]);
      putchar('\n');
    }
}

int ValutaPosizione(char s[8][8])
{
  int punti_bianco = 0, punti_nero = 0, i, j;

  if (ContaPezzo(s,'r') != 1 || ContaPezzo(s,'R') != 1
      || ContaPezzo(s,'c') > 2 || ContaPezzo(s,'C') > 2
      || PedoniFuoriPosto(s))
    return -1;

  for (i = 0; i < 8; i++)
    for (j = 0; j < 8; j++)
      if (islower(s[i][j]))
        punti_bianco += Punti(s[i][j]);
      else if (isupper(s[i][j]))
        punti_nero += Punti(tolower(s[i][j]));

  if (punti_bianco > punti_nero)
    return 1;
  else if (punti_nero > punti_bianco)
```

```
    return 2;
  else
    return 0;
}

int Punti(char c)
{
  switch(c)
    {
    case 'p': return 1;
    case 'c': case 'a': return 3;
    case 't': return 5;
    case 'd': return 9;
    default: return 0;
    }
}

int ContaPezzo(char s[8][8], char pezzo)
{
  int i, j, conta = 0;
  for (i = 0; i < 8; i++)
    for (j = 0; j < 8; j++)
      if (s[i][j] == pezzo)
        conta++;
  return conta;
}

int PedoniFuoriPosto(char s[8][8])
{
  int i = 0;
  for (i = 0; i < 8; i++)
    if (tolower(s[0][i]) == 'p' || tolower(s[7][i]) == 'p')
      return 1;
  return 0;
}
```

Soluzione compito del 14 luglio 2003

Soluzione esercizio 1

Per risolvere questo esercizio correttamente è necessario memorizzare la descrizione del tabellone in una opportuna struttura di dati in memoria centrale. In particolare, utilizziamo un vettore di record, in cui ciascun record contiene la descrizione della casella e le caselle di arrivo corrispondenti alle tre scelte. Il record non comprende il numero della casella stessa perché questo viene fatto corrispondere alla locazione del record nel vettore.

Poiché il gioco parte dalla casella 1, si è deciso di non utilizzare la locazione 0 del vettore e conseguentemente di sovradimensionarlo di una cella. In questo modo la corrispondenza tra numero della casella e locazione è esatta (e non spostata di uno).

La prima operazione della funzione sarà quindi la lettura del file e la memorizzazione nel vettore delle caselle. A questo scopo, si noti che nell'esempio le caselle appaiono nel file in ordine crescente di numero. Però siccome questo non è esplicitamente detto nel testo, nella soluzione si e' deciso di considerare anche il caso di caselle in ordine qualsiasi.

Di conseguenza, nel ciclo di lettura i dati sono inizialmente memorizzati in un record ausiliario, in quando non è ancora nota la loro locazione; una volta letta la locazione (variabile `id`) questi vengono copiati nella locazione corretta del vettore.

La fase successiva è un ciclo che esegue le mosse sul tabellone. Questo ciclo ha due possibile cause di terminazione: la fine della sequenza e la terminazione del gioco (cioè l'arrivo nella casella fittizia 0). Di conseguenza ci sono due condizioni (in *and*) nell'istruzione `while`.

Infine, la casella, se diversa da 0, di arrivo viene restituita. Altrimenti viene restituito il valore -1.

```
#include <stdio.h>
#define MAX_DIM 100

struct Casella
{
  char descrizione[31];
  int scelta[3];
};

int TrovaCasellaFinale(char nome_file[], char sequenza[]);

/* non richiesto per il compito in classe */
int main()
{
  char nome_file[] = "tabellone.txt";
  char sequenza[20];
  int finale;

  printf("Inserisci la sequenza : ");
  scanf("%s",sequenza);
  finale = TrovaCasellaFinale(nome_file, sequenza);
  if (finale == -1)
    printf("La sequenza termina il gioco\n");
  else
    printf("La casella finale e' %d\n", finale);
  return 0;
}

int TrovaCasellaFinale(char nome_file[], char sequenza[])
{
  FILE* fp;
  int id, i = 0;
  /* la locazione 0 del vettore tabellone non e' utilizzata */
  struct Casella tabellone[MAX_DIM+1];
  struct Casella s;

  fp = fopen(nome_file, "r");
  while (fscanf(fp, "%d%s %*c%d%*c%d%*c%d%*c", &id, s.descrizione,
                &s.scelta[0], &s.scelta[1], &s.scelta[2]) != EOF)
    tabellone[id] = s;
  fclose(fp);
  id = 1;
  while (sequenza[i] != '\0' && id != 0)
    {
      if (sequenza[i] == 'S')
        id = tabellone[id].scelta[0];
      else if (sequenza[i] == 'C')
        id = tabellone[id].scelta[1];
      else /* if (sequenza[i] == 'D') */
        id = tabellone[id].scelta[2];
      i++;
    }
  if (id == 0)
    return -1;
  else
    return id;
}
```

Soluzione esercizio 2

L'esercizio si risolve con una lettura carattere per carattere del file. Quando si incontra il carattere `'['`, si legge il valore intero successivo e si inserisci il valore stesso aumentato di `k`.

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

void CambiaPrezzo(char nome_file_input[], char nome_file_output[], int k);

/* non richiesto per il compito in classe */
int main(int argc, char *argv[])
{
  if (argc != 4)
    printf("Numero di parametri sbagliato!!");
  else
    CambiaPrezzo(argv[1],argv[2],atoi(argv[3]));
  return 0;
}

void CambiaPrezzo(char nome_file_input[], char nome_file_output[], int k)
{
  FILE *fp1, *fp2;
  char ch;
  int p;

  fp1 = fopen(nome_file_input,"r");
  fp2 = fopen(nome_file_output,"w");
  while ((ch = getc(fp1)) != EOF)
    {
      if (ch != '[')
        fprintf(fp2,"%c",ch);
      else
        {
          fscanf(fp1,"%d%*c",&p);
          fprintf(fp2,"[%d]",p+k);
        }
    }
  fclose(fp1);
  fclose(fp2);
}
```

Soluzione compito del 2 settembre 2003

Soluzione esercizio 1

Per la soluzione di questa esercizio è indispensabile dichiarare un record per la gestione delle date. A questo scopo dichiariamo il tipo `struct Data` che verrà usato sia per il passaggio dei parametri che per il valore restituito.

Per una corretta modularizzazione è bene definire una funzione ausiliaria che compari due date. Per generalità definiamo questa funzione con tre possibili valori di uscita, in base ai tre casi: minore, uguale, maggiore. Per l'esercizio sarebbe bastata anche una funzione a due valori di uscita (minore-uguale e maggiore).

```
#include <stdio.h>
#include <stdlib.h>
#include "../Utils/util_date.h"

struct Data UltimaDataPrecedente(char nome_file[], struct Data d);
```

```
int main(int argc, char *argv[])
{ /* non richiesto per il compito in classe */
  struct Data d1,d2;

  if (argc != 3)
    {
      printf("Numero di parametri sbagliato\n");
      exit(1);
    }
  sscanf(argv[2],"%d/%d/%d",&d1.giorno,&d1.mese,&d1.anno);
  d2 = UltimaDataPrecedente(argv[1],d1);
  printf("La prima data successiva nel file al %d/%d/%d e' %d/%d/%d\n",
         d1.giorno,d1.mese,d1.anno, d2.giorno,d2.mese,d2.anno);
  return 0;
}

struct Data UltimaDataPrecedente(char nome_file[], struct Data soglia)
{
  struct Data d;
  FILE *fp;

  fp = fopen(nome_file,"r");
  while (fscanf(fp,"%d%*c%d%*c%d",&d.giorno,&d.mese,&d.anno) != EOF)
    if (ComparaDate(d,soglia) == 1)
      return d;
  return soglia;
}
```

Soluzione esercizio 2

La difficoltà dell'esercizio risiede nel fatto che le informazioni sono scritte senza spazi di separazione e quindi non possiamo affidarci completamente alle caratteristiche della funzione `fscanf()` che sfrutta gli spazi tra i valori.

Utilizziamo quindi una lettura per carattere (`getc()`) per leggere il nome della società; e invece la `fscanf()` per leggere i valori.

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

void VariazioneQuotazioni(char nome_file_in[], char nome_file_out[]);
void Ordina(float v[], int n);

int main(int argc, char *argv[])
{ /* non richiesto per il compito in classe */
  if (argc != 3)
    {
      printf("Numero di argomenti del programma sbagliato\n");
      exit(1);
    }
  VariazioneQuotazioni(argv[1], argv[2]);
  return 0;
}

void VariazioneQuotazioni(char nome_file_in[], char nome_file_out[])
{
  FILE *fpr, *fpw;
  char nome_societa[21], ch;
  int i;
  float valori[100];
```

```
  fpr = fopen(nome_file_in, "r");
  fpw = fopen(nome_file_out, "w");

  while ((ch = getc(fpr)) != EOF)
    {
      i = 0;
      do
        {
          nome_societa[i] = ch;
          ch = getc(fpr);
          i++;
        }
      while (ch != '#');
      nome_societa[i] = '\0';
      printf("%s\n",nome_societa);
      i = 0;
      do
        {
          fscanf(fpr,"%g%c",&valori[i],&ch);
          i++;
        }
      while (ch != '$');
      while (getc(fpr) != '\n')
        ;


      Ordina(valori,i);

      fprintf(fpw, "%s %g %g %g\n", nome_societa, valori[0], valori[(i-1)/2], valori[i-1]);
    }
}


void Ordina(float v[], int n)
{ /* Ordinamento con l'algoritmo per selezione */
  int min, i, j;
  float tmp;

  for (i = 0; i < n - 1; i++)
    {
      min = i;
      for (j = i + 1; j < n; j++)
        if (v[j] < v[min])
          min = j;
      /* scambia v[i] e v[min] */
      tmp = v[i];
      v[i] = v[min];
      v[min] = tmp;
    }
}
```

Soluzione compito del 16 settembre 2003

Soluzione esercizio 1

L'esercizio viene risolto dalla funzione `ScriviFile()`. La parte centrale della funzione è un doppio ciclo, di cui il ciclo esterno scandisce il vettore delle frequenze, mentre quello interno fa un numero di iterazioni pari al singolo valore di frequenza, stampando il relativo carattere.

Ad ogni iterazione del ciclo interno viene incrementato un contatore di colonna, e si verifica se il sua valore è pari ad `r`. In caso positivo, si scrive sul file il carattere `\n` e si azzera il contatore di colonna.

Per una maggiore facilità di test, nella soluzione viene anche fornita una funzione, chiamata `CodificaFile()`, che legge un file di testo e crea il vettore delle frequenze.

```
#include <stdio.h>
#include <stdlib.h>
#define MAX_CODIFICA 1000

struct run
{
  char carattere;
  int quantita;
};

int CodificaFile(char nome_file[], struct run codifica[]);
void ScriviFile(struct run codifica[], int l, char nome_file[], int r);

int main()
{ /* non richiesto per il compito in classe */
  char nome_file[21];
  struct run codifica[MAX_CODIFICA];
  int r, l, i;
  printf("Immetti la dimensione delle righe: ");
  scanf("%d", &r);
  printf("Immetti il nome del file da codificare: ");
  scanf("%s", nome_file);
  l = CodificaFile(nome_file, codifica);
  for (i = 0; i < l; i++)
    printf("('%c', %d) ", codifica[i].carattere, codifica[i].quantita);
  printf("\n");
  printf("Immetti il nome del file di output: ");
  scanf("%s", nome_file);
  ScriviFile(codifica,l,nome_file,r);
  return 0;
}

void ScriviFile(struct run codifica[], int l, char nome_file[], int r)
{
  FILE* fp;
  int c, i, j;
  c = 0;
  fp = fopen(nome_file, "w");
  if (fp == NULL)
    {
      printf("Errore: non e' possibile aprire il file %s in scrittura", nome_file);
      exit(-1);
    }
  for (j = 0; j < l; j++)
    for (i = 0; i < codifica[j].quantita; i++)
      if (c < r)
        {
          fprintf(fp, "%c", codifica[j].carattere);
          c++;
        }
      else
        {
          fprintf(fp, "%c\n", codifica[j].carattere);
          c = 0;
        }
```

```
  if (c < r)
    fprintf(fp, "\n");
  fclose(fp);
}


int CodificaFile(char nome_file[], struct run codifica[])
{ /* funzione non richiesta per il compito. Costruisce il vettore a
     partire da un file e restituisce la dimensione del vettore */
  FILE* fp;
  int c, cp, contatore = 1, n = 0;
  fp = fopen(nome_file, "r");
  cp = getc(fp);
  while ((c = getc(fp)) != EOF)
  {
    if (c == cp)
      contatore++;
    else if (c != '\n')
      {
        codifica[n].carattere = cp;
        codifica[n].quantita = contatore;
        contatore = 1;
        cp = c;
        n++;
      }
  }
  codifica[n].carattere = cp;
  codifica[n].quantita = contatore;
  fclose(fp);
  return n + 1;
}
```

Soluzione esercizio 2

Il problema consiste nel separare le due parti della stringa in ingresso. Questo viene fatto utilizzando la funzione di libreria `islower()`, che verifica se un carattere è una lettera minuscola.

Le due parti vengono scritte nelle stringhe di appoggio `nome1` e `nome2`. Si noti che la lunghezza di entrambe queste stringhe è pari al massimo tra la lunghezza del nome e quella del cognome, cioè 20 caratteri. Ciò è necessario per disporre della lunghezza necessaria in entrambe le situazioni.

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

void ModificaStringa(char input[], char output[]);
float ProbNome(char nome[]);

int main(int argc, char *argv[])
{ /* non richiesto per il compito in classe */
  char nome_cognome[51];

  if (argc != 2)
    {
      printf("Manca il parametro!\n");
      exit(1);
    }
  ModificaStringa(argv[1], nome_cognome);
  printf("%s\n",nome_cognome);
  return 0;
```

```
}

void ModificaStringa(char input[], char output[])
{
  char nome1[31], nome2[31];
  int i = 0, j = 0;

  do
    {
      nome1[i] = input[i];
      i++;
    }
  while (islower(input[i]));
  nome1[i] = '\0';
  do
    {
      nome2[j] = input[i];
      i++;
      j++;
    }
  while (input[i] != '\0');
  nome2[j] = '\0';

  if (ProbNome(nome1) > ProbNome(nome2))
    {
      strcpy(output,nome1);
      strcat(output,", ");
      strcat(output,nome2);
    }
  else
    {
      strcpy(output,nome2);
      strcat(output,", ");
      strcat(output,nome1);
    }
}

float ProbNome(char nome[])
{ /* funzione non richiesta per il compito */
  if (strcmp(nome,"Marco") == 0 ||
      strcmp(nome,"Giovanni") == 0  ||
      strcmp(nome,"Irene") == 0)
    return 1;
  else if (strcmp(nome,"Salvatore")  == 0 ||
           strcmp(nome,"Dario") == 0  ||
           strcmp(nome,"Franca") == 0 )
    return 0.8;
  else
    return 0.2;
}
```

Soluzione compito del 9 dicembre 2003

Soluzione esercizio 1

La funzione legge separatamente la parte intera e quella frazionaria del numero. La parte intera termina quando il primo carattere di una riga è un `\n` oppure il file è finito (nel caso che la parte frazionaria non esista).

Per la ricostruzione del numero a partire dalle sue cifre si è deciso di usare due metodi diversi per la due parti. Per quella intera si procede con le moltiplicazioni nel valore corrente per 10, mentre per quella frazionaria si divide il coefficiente moltiplicativo per 10.

```
#include <stdio.h>
#include <stdlib.h>  /* serve per exit() */
#include <string.h>  /* serve per strlen() */

double NumeroUnarioSuFile(char nome_file[]);

int main(int argc, char* argv[])
{/* non richiesto per il compito in classe */
  if (argc != 2)
    {
      printf("Numero di parametri sbagliato\n");
      exit(1);
    }
  printf("Il numero presente sul file e' : %10.3f\n", NumeroUnarioSuFile(argv[1]));
  return 0;
}

double NumeroUnarioSuFile(char nome_file[])
{
  FILE* fp;
  int parte_intera = 0, num_uni;
  double parte_frazionaria = 0.0, coefficiente = 0.1;
  char ch;

  fp = fopen(nome_file,"r");

  while(((ch = getc(fp)) != '\n')  && (ch != EOF))
    {
      num_uni = 0;
      do
        {
          ch = getc(fp);
          if (ch == '1')
            num_uni++;
        }
      while (ch != '\n');
      parte_intera = parte_intera * 10 + num_uni;
    }
  while((ch = getc(fp)) != EOF)
    {
      num_uni = 0;
      do
        {
          ch = getc(fp);
          if (ch == '1')
            num_uni++;
        }
      while (ch != '\n');
      parte_frazionaria += num_uni * coefficiente;
      coefficiente /= 10;
    }
  fclose(fp);
  return parte_intera + parte_frazionaria;
}
```

Soluzione esercizio 2

La funzione dell'esercizio 2 esegue un doppio ciclo in cui analizza ciascun elemento della matrice, esclusi quelli della diagonale principale. Per ciascun elemento attribuisce i punti alla squadra (o le squadre, in caso di simbolo `X`) che ne hanno conseguiti.

```
#include <stdio.h>
#define DIM 100
void CalcolaPunti(char ris[][DIM], int n, int v[]);

int main()
{
  char ris[DIM][DIM];
  int v[DIM];
  int i, j, n;

  printf("Inserisci il numero di squadre: ");
  scanf("%d%*c",&n);
  printf("Inserisci i risultati\n ");

  for (i = 0; i < n; i++)
    for (j = 0; j < n; j++)
      if (i != j)
        {
          printf("Risultato di %d-%d : ",i,j);
          scanf("%c%*c", &ris[i][j]);
        }
      else
        ris[i][j] = '0';

  CalcolaPunti(ris,n,v);

  for (i = 0; i < n; i++)
    printf("%d ", v[i]);
  printf("\n");

  for (i = 0; i < n; i++)
    {
      for (j = 0; j < n; j++)
        printf("%c ", ris[i][j]);
      printf("\n");
    }
  return 0;
}

void CalcolaPunti(char ris[][DIM], int n, int v[])
{
  int i, j;
  for (i = 0; i < n; i++)
    v[i] = 0;
  for (i = 0; i < n; i++)
    for (j = 0; j < n; j++)
      {
        if (i != j)
          {
            if (ris[i][j] == '1')
              v[i] += 3;
            else if (ris[i][j] == 'X')
                {
                  v[i] += 1;
```

```
                v[j] += 1;
              }
            else
              v[j] += 3;
            }
      }
}
```

Soluzione compito del 18 marzo 2004

Soluzione esercizio 1

La soluzione utilizza una funzione ausiliaria, chiamata `NumGiornoSettimana`, che dato il giorno della settimana restituisce il suo numero d'ordine (`Lunedi'`:1, `Martedi'`:2, ..., `Domenica`: 7).

Nella funzione principale, per ciascun appuntamento, si esegue un ciclo interno che esegue `DataSuccessiva` per un numero di volte pari al risultato di `NumGiornoSettimana` invocata con il giorno dell'appuntamento come parametro attuale.

```
#include <stdio.h>
#include <stdlib.h>
#include <string.h>
#include "../Utils/util_date.h"
void CambiaFormato(char nome_file_input[], char nome_file_output[], struct Data d);
int NumGiornoSettimana(char giorno[]);

int main(int argc, char* argv[])
{ /* non richiesto per il compito in classe */
  struct Data inizio;
  if (argc != 3)
    {
      printf("Errore nei parametri\n");
      exit(1);
    }
  printf("Inserisci la data (gg mm aaaa): ");
  scanf("%d%d%d",&inizio.giorno, &inizio.mese, &inizio.anno);
  CambiaFormato(argv[1],argv[2],inizio);
  return 0;
}

void CambiaFormato(char nome_file_input[], char nome_file_output[], struct Data d)
{
  FILE* fp1;
  FILE* fp2;
  char nome[31];
  char giorno[10];
  char ora[6];
  struct Data d1;
  int i, num_giorno;

  fp1 = fopen(nome_file_input,"r");
  fp2 = fopen(nome_file_output,"w");

  while(fscanf(fp1,"%s%*s%s%*s%s",nome,giorno,ora) != EOF)
    {
      d1 = d;
      num_giorno = NumGiornoSettimana(giorno);
      for (i = 1; i < num_giorno; i++)
        d1 = DataSuccessiva(d1);
      fprintf(fp2,"%s %d/%d/%d ore %s\n", nome, d1.giorno, d1.mese, d1.anno,ora);
```

```
    }
  fclose(fp1);
  fclose(fp2);
}

int NumGiornoSettimana(char giorno[])
{
  if (strcmp(giorno,"Lunedi'") == 0)
    return 1;
  else if (strcmp(giorno,"Martedi'") == 0)
    return 2;
  else if (strcmp(giorno,"Mercoledi'") == 0)
    return 3;
  else if (strcmp(giorno,"Giovedi'") == 0)
    return 4;
  else if (strcmp(giorno,"Venerdi'") == 0)
    return 5;
  else if (strcmp(giorno,"Sabato") == 0)
    return 6;
  else
    return 7;
}
```

Soluzione esercizi 2 e 3

La funzione dell'esercizio 2 costruisce la stringa `d` di uscita copiando i caratteri non numerici e utilizzando `strcat` per quelli numerici. Per il corretto funzionamento di `strcat` è indispensabile che la stringa `d` contenga il terminatore, quindi questo viene inserito ogni qual volta si entra nel ramo `else`.

```
#include <stdio.h>
#include <string.h>
#include <ctype.h>
#define MAX_LEN 101

int RiscriviNumero(char d[], char s[]);
void TraduciNumero(char num[], int n);

int main()
{
  char testo_originale[MAX_LEN], testo_cambiato[MAX_LEN];
  int i = -1;

  printf("Inserisci la stringa originale (terminata dal punto) : ");
  do
    {
      i++;
      testo_originale[i] = getchar();
    }
  while (testo_originale[i] != '.');
  testo_originale[i+1] = '\0';

  RiscriviNumero(testo_cambiato,testo_originale);
  printf("La stringa originale e': %s\nLa stringa modificata e': %s\n", testo_originale, testo_cambiato
  return 0;
}

int RiscriviNumero(char d[], char s[])
{
  int i, j = 0, n, k = 0;
  char numero[21];
```

```
  n = strlen(s);
  for (i = 0; i < n; i++)
    {
      if (!isdigit(s[i]))
        {
          d[j] = s[i];
          j++;
        }
      else
        {
          TraduciNumero(numero,s[i] - '0');
          d[j] = '\0';
          strcat(d, numero);
          j += strlen(numero);
          k++;
        }
    }
  d[j] = '\0';
  return k;
}

void TraduciNumero(char num[], int n)
{
  if (n == 0)
    strcpy(num, "zero");
  else if (n == 1)
    strcpy(num, "un");
  else if (n == 2)
    strcpy(num, "due");
  else if (n == 3)
    strcpy(num, "tre");
  else if (n == 4)
    strcpy(num, "quattro");
  else if (n == 5)
    strcpy(num, "cinque");
  else if (n == 6)
    strcpy(num, "sei");
  else if (n == 7)
    strcpy(num, "sette");
  else if (n == 8)
    strcpy(num, "otto");
  else if (n == 9)
    strcpy(num, "nove");
  else
    strcpy(num, "errore");
}
```

Soluzione compito del 1 aprile 2004

Soluzione esercizio 1

La funzione legge (e memorizza in una stringa) ciascuna riga per caratteri (`getc`) fino al carattere punto e virgola. A questo punto legge la provincia e poi compie un ciclo che elimina il resto della riga (si è assunto che possano anche esserci degli spazi dopo la provincia).

La verifica se la provincia letta è nella lista è delegata ad una funzione ausiliaria che restituisce `1` se la provincia è nella lista, `0` altrimenti.

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

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

void FiltraPerProvince(char file_in[], char file_out[], char prov[]);
int CercaProvincia(char p[], char lista_p[]);

int main()
{/* non richiesto per il compito in classe */
  char nome_file_in[21], nome_file_out[21], tmp[3], province[201] = "";
  int fine_lettura = 0;
  printf("Immetti il nome del file di input: ");
  scanf("%s", nome_file_in);
  printf("Immetti il nome del file di outpput: ");
  scanf("%s", nome_file_out);
  printf("Immetti le sigle delle province (separate da spazi, spazio e punto per terminare): ");
  while (!fine_lettura)
    {
      scanf("%3s", tmp);
      if (strcmp(tmp, ".") == 0)
        {
          fine_lettura = 1;
          continue;
        }
      strcat(province, tmp);
      strcat(province, " ");
    }
  province[strlen(province) - 1] = '\0'; /* ci sarà uno spazio in più */
  FiltraPerProvince(nome_file_in, nome_file_out, province);
  return 0;
}

void FiltraPerProvince(char file_in[], char file_out[], char lista_prov[])
{
  FILE *fp_in, *fp_out;
  char buf[256], prov_cfr[3], ch;
  int i;
  fp_in = fopen(file_in, "r");
  fp_out = fopen(file_out, "w");

  while ((ch = getc(fp_in)) != EOF)
    {
      i = 0;
      do
        {
          buf[i] = ch;
          i++;
          ch = getc(fp_in);
        }
      while(ch != ';');
      buf[i] = '\0';
      fscanf(fp_in,"%s",prov_cfr);
      while ((ch = getc(fp_in) != '\n' && ch != EOF))
        ; /* legge il resto della riga compreso il \n finale */
      if (CercaProvincia(prov_cfr, lista_prov))
        fprintf(fp_out, "%s; %s\n", buf, prov_cfr);
    }
  fclose(fp_in);
  fclose(fp_out);
}
```

```
int CercaProvincia(char p[], char lista_p[])
{
  int i;
  char tmp[3];
  i = 0;
  while (lista_p[i] != '\0')
    {
      if (lista_p[i] == p[0] && lista_p[i+1] == p[1])
        return 1;
      i += 3; /* passa alla provincia successiva */
    }
  return 0;
}
```

Soluzione esercizio 2

Come suggerito, si definisce una funzione, chiamata `Vincitore`, che calcola chi ha vinto una partita, restituendo 1 o 2.

La funzione principale scandisce il vettore verificando in ogni partita se il giocatore è presente come primo o come secondo e se questi ha vinto la partita.

```
#include <stdio.h>
#include <string.h>
#define PARTITE 4

struct Partita
{
  char gioc1[21];
  char gioc2[21];
  int ris1[3], ris2[3];
};

int PartiteVinte(struct Partita v[], int n, char nome[]);
int Vincitore(struct Partita p);

int main()
{
  struct Partita v[PARTITE];
  int i, j;
  char nome[21];

  for (i = 0; i < PARTITE; i++)
    {
      printf("Immetti i dati della partita %d\n", i);
      printf("Nome giocatore 1: ");
      scanf("%s", v[i].gioc1);
      printf("Nome giocatore 2: ");
      scanf("%s", v[i].gioc2);
      printf("Risultati giocatore 1: ");
      for (j = 0; j < 3; j++)
        scanf("%d", &v[i].ris1[j]);
      printf("Risultati gioc 2: ");
      for (j = 0; j < 3; j++)
        scanf("%d", &v[i].ris2[j]);
    }
  do
    {
      printf("Immetti il nome del giocatore di interesse (inserisci un punto per terminare): ");
      scanf("%s", nome);
```

```
      if (strcmp(nome, ".") == 0)
        break;
      printf("Il numero di partite vinte dal giocatore %s e' %d\n",
             nome, PartiteVinte(v, PARTITE, nome));
    }
  while (1);
  return 0;
}

int PartiteVinte(struct Partita v[], int n, char nome[])
{
  int i, vinte = 0;
  for (i = 0; i < n; i++)
    if ((strcmp(v[i].gioc1, nome) == 0 && Vincitore(v[i]) == 1)
     || (strcmp(v[i].gioc2, nome) == 0 && Vincitore(v[i]) == 2) )
        vinte++;
  return vinte;
}

int Vincitore(struct Partita p)
{
  int i, vinti1 = 0, vinti2 = 0;
  for (i = 0; i < 3; i++)
    {
      if (p.ris1[i] > p.ris2[i])
        vinti1++;
      else if (p.ris1[i] < p.ris2[i])
        vinti2++;
    }
  if (vinti1 > vinti2)
    return 1;
  else
    return 2;
}
```

Soluzione compito del 28 giugno 2004

Soluzione esercizio 1

Il nome della città e la situazione meteorologica vengono lette per caratteri (`getc`) in quanto possono contenere degli spazi. La temperatura e la scala vengono letti invece con la `fscanf`.

Nella lettura della città è necessario fare attenzione a non perdere il primo carattere che viene letto dal controllo del ciclo più esterno e viene inserito nella stringa prima di una nuova lettura.

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

void TrasformaMeteo(char file_in[], char t, char file_out[]);
float F2C(float temp);
float C2F(float temp);

int main()
{/* non richiesto per il compito in classe */
  char nome_file_in[21], nome_file_out[21], conversione;
  printf("Immetti il nome del file di input: ");
  scanf("%s", nome_file_in);
  do
    {
```

```
      printf("Immetti il tipo di conversione\n");
      printf("Farenheit -> Celsius = 'c', Celsius -> Farenheit = 'f': ");
      scanf("%*c%c", &conversione);
    }
  while (conversione != 'c' && conversione != 'f');
  printf("Immetti il nome del file di output: ");
  scanf("%s", nome_file_out);
  TrasformaMeteo(nome_file_in, conversione, nome_file_out);
  return 0;
}

void TrasformaMeteo(char file_in[], char t, char file_out[])
{
  FILE *fp_in, *fp_out;
  int i;
  char ch, citta[31], scala, condizione[31];
  float temp;
  fp_in = fopen(file_in, "r");
  fp_out = fopen(file_out, "w");

  while ((ch = getc(fp_in)) != EOF)
    {
      i = 0;
      while (ch != ',')
        {
          citta[i] = ch;
          i++;
          ch = getc(fp_in);
        }
      citta[i] = '\0';
      fscanf(fp_in, "%f%c", &temp, &scala);
      i = 0;
      while ((ch = getc(fp_in)) != '\n')
        {
          condizione[i] = ch;
          i++;
        }
      condizione[i] = '\0';
      if (t != scala)
        {
          if (t =='c')
            temp = F2C(temp);
          else
            temp = C2F(temp);
        }
      fprintf(fp_out, "%s, %f%c %s\n", citta, temp, t, condizione);
    }
  fclose(fp_in);
  fclose(fp_out);
}

float F2C(float temp)
{
  return (temp - 32.0) * 5.0 / 9.0 ;
}

float C2F(float temp)
{
  return temp * 9.0 / 5.0 + 32.0;
```

```
}
```

Soluzione esercizio 2

La funzione legge (e controlla) la prima data, prima di entrare in un ciclo in cui si confronta ciascuna data letta con la data massima.

Per modularità, la data convenzionale da restituire in caso di errore viene generata da una funzione ausiliaria.

```
#include <stdio.h>
#include <stdlib.h>
#include "../Utils/util_date.h"

struct Data UltimaData(char nome_file[]);
struct Data DataConvenzionale();

int main(int argc, char *argv[])
{ /* non richiesto per il compito in classe */
  struct Data d1, d2;

  if (argc != 2)
    {
      printf("Numero di parametri sbagliato\n");
      exit(1);
    }

  d1 = UltimaData(argv[1]);
  d2 = DataConvenzionale();

  if (ComparaDate(d1,d2) == 0)
    printf("Nel file e' presente una data non valida\n");
  else
    printf("L'ultima data nel file e' %d/%d/%d\n", d1.giorno,d1.mese,d1.anno);
  return 0;
}

struct Data UltimaData(char nome_file[])
{
  struct Data d, d_max;
  FILE *fp;

  fp = fopen(nome_file,"r");
  fscanf(fp,"%*1s%d%*c%d%*c%d%*c",&d_max.anno,&d_max.mese,&d_max.giorno);
  if (!DataValida(d_max))
    return DataConvenzionale();
  while (fscanf(fp,"%*1s%d%*c%d%*c%d%*c",&d.anno,&d.mese,&d.giorno) != EOF)
    if (!DataValida(d))
      return DataConvenzionale();
    else if (ComparaDate(d,d_max) == 1)
      d_max = d;
  return d_max;
}

struct Data DataConvenzionale()
{
  struct Data d;
  d.giorno = 1;
  d.mese = 1;
  d.anno = 2000;
  return d;
```

```
}
```

Soluzione compito del 8 luglio 2004

Soluzione esercizio 1

Per rappresentare l'orario si utilizza un semplice record a due campi (`ore` e `minuti`). L'esercizio si risolve con un semplice ciclo in cui si analizza una lettura alla volta e si verificano le due condizioni. Alla prima lettura che verifica entrambe le condizioni si interrompe il ciclo e si restituisce il valore. Se nessuna lettura verifica le condizioni, allora si prosegue con l'inserimento dei valori di default (`0` per entrambe).

```
#include <stdio.h>

struct Orario
{
  int ora;
  int minuti;
};

struct Orario PrimoCampione(char nome_file[], float d, float v);

int main()
{/* non richiesto per il compito in classe */
  struct Orario orario_soglia;
  char nome_file[21];
  float vel, dist;

  printf("Nome file di input: ");
  scanf("%s", nome_file);
  printf("Inserisci velocita' e distanza di soglia: ");
  scanf("%g%g", &vel, &dist);
  orario_soglia = PrimoCampione(nome_file, vel, dist);
  printf("Orario di soglia: %d:%d\n",
         orario_soglia.ora, orario_soglia.minuti);
  return 0;
}

struct Orario PrimoCampione(char nome_file[], float d, float v)
{
  struct Orario ris;
  FILE* fp;
  float distanza, velocita;

  fp = fopen(nome_file,"r");
  while (fscanf(fp,"%d:%d %g%*s%g%*s", &ris.ora, &ris.minuti,
                &distanza, &velocita) != EOF)
    if (distanza >= d && velocita >= v)
      {
        fclose(fp);
        return ris;
      }
  ris.ora = 0;
  ris.minuti = 0;
  fclose(fp);
  return ris;
}
```

Soluzione esercizio 2

La soluzione si compone di una sequenza di cicli in ciascuno dei quali viene letta una parte della stringa e copiata nel corrispondente campo del record.

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

struct DatiPersona
{
  char nome[31];
  char sesso;
  int anno_nascita;
  char luogo_nascita[31];
};

struct DatiPersona ConvertiDatiDaStringa(char s[]);

int main()
{/* non richiesto per il compito in classe */
  char str[101];
  struct DatiPersona ris;
  int i = 0;

  printf("Inserisci i dati della persona (Invio per terminare): ");
  while ((str[i] = getchar()) != '\n')
    i++;
  str[i] = '\0';
  printf("stringa inserita: %s\n", str);
  ris = ConvertiDatiDaStringa(str);

  printf("Nome e cognome: %s\nSesso: %c\n", ris.nome, ris.sesso);
  printf("Anno di Nascita: %d\nLuogo di nascita: %s\n", ris.anno_nascita,
         ris.luogo_nascita);
  return 0;
}

struct DatiPersona ConvertiDatiDaStringa(char s[])
{
  struct DatiPersona dati;
  int i = 0, j = 0;
  char stringa_anno[5];

  while (s[i] != ',')
    {
      dati.nome[i] = s[i];
      i++;
    }
  dati.nome[i] = '\0';
  i += 2; /* salta la virgola e il blank dopo nome e cognome */
  dati.sesso = s[i];
  i += 3; /* salta sesso, virgola e blank */

  j = 0;
  while (s[i] != ',')
    {
      stringa_anno[j] = s[i];
      i++;
      j++;
    }
```

```
  stringa_anno[j] = '\0';
  dati.anno_nascita = atoi(stringa_anno);

  i += 2;
  j = 0;
  while (s[i] != '\0')
    {
      dati.luogo_nascita[j] = s[i];
      i++;
      j++;
    }
  dati.luogo_nascita[j] = '\0';
  return dati;
}
```

Soluzione compito del 3 settembre 2004

Soluzione esercizio 1

L'esercizio si risolve con un ciclo di lettura del file di ingresso per caratteri (`getc`). Quando si incontra il carattere `<`, si analizza il carattere successivo, e se anche questo è `<` si procede all'eliminazione del commento, altrimenti entrambi i caratteri vengono scritti nel file di uscita.

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

void EliminaCommenti(char nome_file_input[], char nome_file_output[]);

/* non richiesto per il compito in classe */
int main (int argc, char *argv[])
{
  if (argc != 3)
    {
      printf("Passare i nomi dei file sulla riga di comando!\n");
      exit(1);
    }
  EliminaCommenti(argv[1], argv[2]);
  return 0;
}

void EliminaCommenti(char nome_file_input[], char nome_file_output[])
{

  FILE *fin, *fout;
  char ch1, ch2, ch;

  fin = fopen (nome_file_input, "r");
  fout = fopen (nome_file_output, "w");
  while ((ch1 = getc (fin)) != EOF)
    {
      if (ch1 != '<')
        fprintf(fout,"%c",ch1);
      else
        {
          ch2 = getc(fin);
          if (ch2 != '<')
            fprintf(fout,"%c%c",ch1,ch2);
          else
```

```
            while ((ch = getc (fin)) != EOF && ch != '|')
              ; /* elimina tutto fino alla fine del commento (o del file) */
        }
    }
   fclose (fin);
   fclose (fout);
}
```

Soluzione esercizio 2

La soluzione si compone di due cicli in sequenza. Nel primo ciclo vengono copiati i giocatori che compaiono nel primo vettore, sommandogli anche l'eventuale il punteggio che questi hanno nel secondo vettore. Nel secondo ciclo, vengono accodati i giocatori che compaiono nel secondo vettore ma non nel primo.

La funzione soluzione si avvale di una funzione ausiliaria, chiamata `Posizione`, che restituisce la posizione di un nome in un vettore, oppure `-1` nel caso in cui il nome non sia presente.

```
#include <stdio.h>
#include <string.h>
#define DIM 10

struct Giocatore
{
  char nome[20];
  int punti;
};

int LeggiVettore(struct Giocatore v[]);
int Posizione(struct Giocatore g, struct Giocatore v[], int n);
int SommaPunti(struct Giocatore v1[], int n1,
               struct Giocatore v2[], int n2,
               struct Giocatore r[]);

/* non richiesto per il compito in classe */
int main()
{
  struct Giocatore vet1[DIM], vet2[DIM], vet3[DIM];
  int dim1, dim2, dim3;
  int i;

  printf("Inserisci in primo vettore\n");
  dim1 = LeggiVettore(vet1);
  printf("Inserisci in secondo vettore\n");
  dim2 = LeggiVettore(vet2);
  dim3 = SommaPunti(vet1,dim1,vet2,dim2,vet3);
  printf("Vettore somma:\n");
  for (i = 0; i < dim3; i++)
    printf("%-20s %2d\n", vet3[i].nome, vet3[i].punti);
  return 0;
}

int LeggiVettore(struct Giocatore v[])
{
  int i,d;
  printf("Numero di elementi : ");
  scanf("%d",&d);
  for (i = 0; i < d; i++)
    {
      printf("Nome giocatore (senza spazi): ");
      scanf("%s",v[i].nome);
```

```
      printf("Punti giocatore : ");
      scanf("%d",&v[i].punti);
    }
  return d;
}

int SommaPunti(struct Giocatore v1[], int n1,
               struct Giocatore v2[], int n2,
               struct Giocatore r[])
{
  int i, j, k = n1;
  for (i = 0; i < n1; i++)
    {
      r[i] = v1[i];
      j = Posizione(v1[i],v2,n2);
      if (j != -1)
        r[i].punti += v2[j].punti;
    }

  for (i = 0; i < n2; i++)
    {
      j = Posizione(v2[i],v1,n1);
      if (j == -1)
        {
          r[k] = v2[i];
          k++;
        }
    }
  return k;
}

int Posizione(struct Giocatore g, struct Giocatore v[], int n)
{
  int i;
  for (i = 0; i < n; i++)
    if (strcmp(g.nome,v[i].nome) == 0)
      return i;
  return -1;
}
```

Soluzione compito del 14 settembre 2004

Soluzione esercizi 1 e 2

L'esercizio 1 non richiede particolari commenti. Per l'esercizio 2 il problema principale è quello di verificare e registrare la fine del file. Si è deciso di segnalare questa condizione con il fatto che il titolo del libro che si sta leggendo si trova ad essere vuoto (primo carattere uguale al terminatore).

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

#include "../Utils/util_date.h"

#define MAX_DIM 10

struct Volume
{
   char titolo[65];
   char autori[64];
```

```
   char collocazione[16];
   int in_prestito;
   struct Data fine_prestito;
};

int InPrestito(struct Volume v[], int n, struct Data d, struct Volume r[]);
int LeggiLibri(char nome_file[], struct Volume v[]);
struct Volume LeggiRecord(FILE* fp);
void LeggiCampo(FILE* fp, char campo[]);

int main()
{/* non richiesto per il compito in classe */
  struct Volume biblioteca[MAX_DIM], prestiti[MAX_DIM];
  struct Data oggi = {16, 9, 2004};
  int n1, n2, i;

  n1 = LeggiLibri("libri.txt",biblioteca);
  n2 = InPrestito(biblioteca, n1, oggi, prestiti);
  for (i = 0; i < n2; i++)
    {
      printf ("%s, di %s, collocazione: %s\n",
              prestiti[i].titolo, prestiti[i].autori,
              prestiti[i].collocazione);
     }
  return 0;
}

int InPrestito(struct Volume v[], int n, struct Data d, struct Volume r[])
{
  int i, j = 0;
   for (i = 0; i < n; i++)
     if (ComparaDate(v[i].fine_prestito,d) == -1)
       {
         r[j] = v[i];
         j++;
       }
   return j;
}

int LeggiLibri(char nome_file[], struct Volume v[])
{
  FILE* fp;
  int i = 0;
  int fine;

  fp = fopen(nome_file,"r");

  do
    {
      v[i] = LeggiRecord(fp);
      fine = (v[i].titolo[0] == '\0');
      i++;
    }
  while (!fine);
  return i-1;
}

struct Volume LeggiRecord(FILE* fp)
{
```

```
  struct Volume vol;
  LeggiCampo(fp,vol.titolo);
  if (vol.titolo[0] != '\0')
    {
      LeggiCampo(fp,vol.autori);
      LeggiCampo(fp,vol.collocazione);
      fscanf(fp,"%d%*c%d%*c%d%*c%d", &vol.in_prestito,
             &vol.fine_prestito.giorno, &vol.fine_prestito.mese,
             &vol.fine_prestito.anno);
      while (getc(fp) != '\n')
        ;
    }
  return vol;
}

void LeggiCampo(FILE* fp, char campo[])
{
  int j = 0;
  char ch;
  while ((ch = getc(fp)) != '|'  && ch != EOF)
    {
      campo[j] = ch;
      j++;
    }
  campo[j] = '\0';
}
```

Soluzione esercizio 3

La funzione si compone di un semplice ciclo di lettura di ciascuna parola e di verifica della presenza delle doppie (tramite funzione ausiliaria `ContieneDoppie`).

Si noti nella funzione `ContieneDoppie` il primo `if` che serve unicamente per gestire in modo corretto il caso in cui gli venga passata una stringa vuota.

```
#include <stdio.h>
#include <stdlib.h>
#define MAXSTRLEN 256

void TrascriviDoppie(char nome_file_input[], char nome_file_output[]);
int ContieneDoppie(char s[]);

int main(int argc, char *argv[])
{/* non richiesto per il compito in classe */
   if (argc != 3)
   {
      printf ("Numero di argomenti non valido\n");
      exit (-1);
   }
   TrascriviDoppie(argv[1], argv[2]);
   return 0;
}

void TrascriviDoppie(char nome_file_input[], char nome_file_output[])
{
  FILE* fp_in, *fp_out;
  char parola[21];

  fp_in = fopen(nome_file_input, "r");
  fp_out = fopen(nome_file_output, "w");
```

```
  while (fscanf(fp_in,"%s", parola) != EOF)
    if (ContieneDoppie(parola))
      fprintf(fp_out,"%s ", parola);

  fclose(fp_in);
  fclose(fp_out);
}

int ContieneDoppie(char s[])
{
   int i = 0;

   if (s[i] == '\0')
      return 0;
   while (s[i+1] != '\0')
   {
      if (s[i] == s[i+1])
         return 1;
      i++;
   }
   return 0;
}
```

Soluzione compito del 9 dicembre 2004

Soluzione esercizio 1

Dopo aver letto una prima data, la funzione entra in un ciclo in cui valuta la distanza tra le date di due operazioni consecutive nel file. Se la distanza è minore del parametro, la funzione termina, altrimenti la seconda data diventa la prima ed una nuova seconda data viene letta.

```
#include <stdio.h>
#include <string.h>
#include "../Utils/util_date.h"

int ControlloDate(char nome_file[], int giorni, char attivita[]);

int main ()
{ /* non richiesto per il compito in classe */
  char nome_file[] = "Operazioni.txt";
  char operazione[21];
  int giorni;

  printf("Numero giorni massimo : ");
  scanf("%d",&giorni);
  if (ControlloDate(nome_file,giorni,operazione))
    printf("Attivita' : %s\n",operazione);
  else
    printf("Non esistono operazioni vicine\n");
  return 0;
}

int ControlloDate(char nome_file[], int n, char oper[])
{
  FILE* fp;
  struct Data d1, d2;
  char op1[21], op2[21];

  fp = fopen(nome_file,"r");
```

```
  fscanf(fp,"%s%d%*c%d%*c%d", op1, &d1.giorno,&d1.mese,&d1.anno);

  while(fscanf(fp,"%s%d%*c%d%*c%d", op2, &d2.giorno,&d2.mese,&d2.anno) != EOF)
    {
      if (DistanzaTraDate(d1,d2) < n)
        {
          strcpy(oper,op1);
          fclose(fp);
          return 1;
        }
      else
        {
          d1 = d2;
          strcpy(op1,op2);
        }
    }
  fclose(fp);
  return 0;
}
```

Soluzione esercizio 2

L'esercizio viene risolto leggendo tutte le informazioni sul file per caratteri (usando `getc`). Nome e cognome vengono letti da dei cicli che vanno avanti fino alla virgola. Riguardo al voto, se il primo carattere è `i` viene riconosciuto il caso `insufficiente`, altrimenti vengono lette le due cifre e ricostruito il numero corrispondente. In tutti i casi viene successivamente letto (e ignorato) il resto della riga, che a seconda dei casi può anche contenere caratteri alfabetici (`nsufficiente` oppure `e lode`), che però non sono significativi per la funzione.

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

float MediaStudenti(char nome_file[]);

int main ()
{ /* non richiesto per il compito in classe */
  char nome_file[] = "Studenti.txt";
  printf("Media file : %g\n", MediaStudenti(nome_file));
  return 0;
}

float MediaStudenti(char nome_file[])
{
  FILE* fp;
  int somma = 0, conta = 0, voto;
  char ch, ch2;

  fp = fopen(nome_file,"r");

  while (getc(fp) != EOF)
    {
      while (getc(fp) != ',')
        ; /* legge il nome */
      while (getc(fp) != ',')
        ; /* legge il cognome */
      getc(fp); /* legge lo spazio dopo la virgola */
      ch = getc(fp);
      if (ch != 'i')
        {
```

```
          ch2 = getc(fp); /* legge la seconda cifra */
          voto = (ch - '0') * 10 + (ch2 - '0');
          somma += voto;
          conta++;
        }
      while (getc(fp) != '\n')
        ; /* legge il resto della riga */
    }
  return (float)somma/conta;
}
```

Soluzione compito del 17 marzo 2005

Soluzione esercizio 1

Per la soluzione di questo esercizio è necessario memorizzare tutto il contenuto del file in un'opportuna struttura di dati in memoria centrale. A questo scopo viene definito il tipo record chiamato `DatiProva` e viene utilizzato un vettore (sovradimensionato) di elementi di tale tipo in cui viene copiato il contenuto del file nel ciclo `while` iniziale.

In un secondo ciclo viene calcolato il tempo minimo di esecuzione, e in un terzo viene calcolato l'indice della prova che soddisfa le condizioni richieste.

```
#include <stdio.h>
#define MAX_DIM 100

struct DatiProva
{
  char codice[4];
  int risultato;
  float tempo;
};

struct DatiProva MaxProveVeloci(char nome_file[]);

int main(int argc, char* argv[])
{ /* non richiesto per il compito in classe */
  struct DatiProva max;
  max = MaxProveVeloci(argv[1]);
  printf("Il massimo delle prove veloci e' (%s,%d,%gs)\n",
         max.codice, max.risultato, max.tempo);
  return 0;
}

struct DatiProva MaxProveVeloci(char nome_file[])
{
  struct DatiProva prove[MAX_DIM];
  int i = 0, n, max, indice_max;
  float min_tempo;
  FILE* fp;

  fp = fopen(nome_file,"r");
  while(fscanf(fp,"%s%d%g%*c", prove[i].codice,
               &prove[i].risultato, &prove[i].tempo) != EOF)
      i++;
  n = i;
  min_tempo = prove[0].tempo;
```

```
  for (i = 1; i < n; i++)
    if (prove[i].tempo < min_tempo)
      min_tempo = prove[i].tempo;
  max = 0;
  indice_max = -1;
  for (i = 1; i < n; i++)
    if (prove[i].risultato > max && prove[i].tempo < min_tempo + 1)
      {
        max = prove[i].risultato;
        indice_max = i;
      }
  return prove[indice_max];
}
```

Soluzione esercizio 2

Per una buona modularità della soluzione è indispensabile scrivere una funzione ausiliaria, chiamata `Valore`, che dato il nome di una variabile ed il vettore restituisca il valore della variabile (eventualmente 0).

Utilizzando la funzione `Valore`, la funzione principale dell'esercizio si risolve in un semplice ciclo di lettura da un file tramite `fscanf` e di scrittura sull'altro file.

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

struct Variabile
{
  char nome[21];
  int valore;
};

void AggiungiRisultato(char nome_file_inut[], char nome_file_output[],
                       struct Variabile valori[], int n);
int Valore(char nome[], struct Variabile valori[], int n);

int main(int argc, char* argv[])
{/* non richiesto per il compito in classe */
  struct Variabile w[3] = {{"alfa",3}, {"beta",4}, {"gamma", 21}};
  AggiungiRisultato(argv[1],argv[2], w,3);
  return 0;
}

void AggiungiRisultato(char nome_file_input[], char nome_file_output[],
                       struct Variabile valori[], int n)
{
  FILE* fp_in;
  FILE* fp_out;
  char var1[21], var2[21], op[2];
  int ris;

  fp_in = fopen(nome_file_input,"r");
  fp_out = fopen(nome_file_output,"w");
  while (fscanf(fp_in,"%s%s%s", var1, op, var2) != EOF)
    {
      if (strcmp(op,"+") == 0)
        ris = Valore(var1,valori,n) + Valore(var2,valori,n);
      else
        ris = Valore(var1,valori,n) - Valore(var2,valori,n);
      fprintf(fp_out,"%s %s %s = %d\n", var1, op, var2, ris);
    }
```

```
}

int Valore(char nome[], struct Variabile valori[], int n)
{
  int i;
  for (i = 0; i < n; i++)
    if (strcmp(nome,valori[i].nome) == 0)
      return valori[i].valore;
  return 0;
}
```

Soluzione compito del 7 aprile 2005

Soluzione esercizio 1

Per modularità abbiamo definito una funzione ausiliaria, chiamata `LeggiCampo`, che legge un singolo campo dal file e lo scrive in una stringa passatagli come parametro. Inoltre la funzione `LeggiCampo` restituisce un valore booleano che registra la condizione di terminazione del file. Questo valore viene utilizzato per la condizione di permanenza nel ciclo della funzione principale.

Si noti che nella funzione principale la verifica di fine file viene fatta sia nella lettura del primo campo che nell'ultimo. Questo consente di gestire correttamente sia il caso che l'ultimo riga contenga il carattere di ritorno a capo sia che non lo contenga.

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

void CercaIndirizzi(char file_in[], char file_out[], char s[]);
int LeggiCampo(FILE* fp, char campo[]);

int main()
{ /* non richiesto per il compito in classe */
  char nome_file_in[21], nome_file_out[21], match[21];

  printf("Immetti il nome del file di input: ");
  scanf("%s", nome_file_in);
  printf("Immetti il nome del file di output: ");
  scanf("%s", nome_file_out);
  printf("Immetti la stringa da cercare: ");
  scanf("%s", match);
  CercaIndirizzi(nome_file_in, nome_file_out, match);
  return 0;
}

void CercaIndirizzi(char file_in[], char file_out[], char s[])
{
  FILE *fp_in, *fp_out;
  char dato[255], campo[65];
  int da_scrivere, finito = 0;

  fp_in = fopen(file_in, "r");
  fp_out = fopen(file_out, "w");
  while (!finito)
    {
      da_scrivere = 0;
      dato[0] = '\0';
        /* Legge i dati relativi al nome */
      finito = LeggiCampo(fp_in, campo);
      if (!finito)
```

```
        {
          strcat(dato, campo);
            /* Legge i dati relativi all'indirizzo */
          LeggiCampo(fp_in, campo);
          strcat(dato, campo);
          da_scrivere = (strstr(campo, s) != NULL);
            /* Legge i dati relativi a cap e citta' */
          LeggiCampo(fp_in, campo);
          if (da_scrivere)
            strcat(dato, campo);
            /* Legge i dati relativi alla provincia */
          finito = LeggiCampo(fp_in, campo);
          if (da_scrivere)
            {
              strcat(dato, campo);
              fprintf(fp_out, "%s", dato);
            }
        }
    }
  fclose(fp_in);
  fclose(fp_out);
}

int LeggiCampo(FILE* fp, char campo[])
{
  char ch;
  int i = 0;

  while ((ch = fgetc(fp)) != EOF && ch != ';' && ch != '\n')
    {
      campo[i] = ch;
      i++;
    }
  if (ch == EOF)
    {
      campo[i] = '\0';
      return 1;
    }
  else
    {
      campo[i] = ch;
      campo[i+1] = '\0';
      return 0;
    }
}
```

Soluzione esercizio 2

Per modularità il calcolo del valore medio è delegato ad una funzione ausiliaria. Questa funzione, chiamata `Medio`, verifica anche le condizioni di bordo e conta il numero di vicini dell'elemento.

```
#include <stdio.h>
#include <math.h>
#define MAX_DIM 100

void Smussatura(int A[][MAX_DIM], int m, int n, int B[][MAX_DIM]);
int Medio(int A[][MAX_DIM], int i, int j, int m, int n);

int main()
{ /* non richiesto per il compito in classe */
```

```
  int M1[MAX_DIM][MAX_DIM], M2[MAX_DIM][MAX_DIM];
  int i, j, n, m;

  printf("Immetti il numero di righe della matrice: ");
  scanf("%d", &m);
  printf("Immetti il numero di colonne della matrice: ");
  scanf("%d", &n);
  for (i = 0; i < m; i++)
    {
      printf("Immetti gli elementi della %da riga (separati da spazi): ", i+1);
      for (j = 0; j < n; j++)
        scanf("%d",&M1[i][j]);
    }
  Smussatura(M1, m, n, M2);
  for (i = 0; i < m; i++)
    {
      for (j = 0; j < n; j++)
        printf("%d ", M2[i][j]);
      printf("\n");
    }
  return 0;
}

void Smussatura(int A[][MAX_DIM], int m, int n, int B[][MAX_DIM])
{
  int i, j;
  for (i = 0; i < m; i++)
    for (j = 0; j < n; j++)
      B[i][j] = Medio(A,i,j,m,n);
}

int Medio(int A[][MAX_DIM], int i, int j, int m, int n)
{
  int h, k, v = 0, medio = 0;
  for (k = -1; k <= 1; k++)
    for (h = -1; h <= 1; h++)
      if (i + k >= 0 && i + k < m && j + h >= 0 && j + h < n)
        {
          medio += A[i+k][j+h];
          v++;
        }
  return (int)round(medio/(float)v);
}
```

Soluzione compito del 15 luglio 2005

Soluzione esercizio 1

Nella soluzione proposta viene definita una costante `N_MAX` (pari a 5) e viene utilizzata una matrice (sovradimensionata) di dimensione `N_MAX * N_MAX`.

La matrice $n^2 \times n^2$ viene letta dal file e sottoposta alle tre verifiche: righe, colonne e riquadri. In dettaglio, ciascun numero tra 1 e $n \times n$ viene cercato dell'area richiesta, e si verifica che compaia una ed una sola volta.

```
#include <stdio.h>
#include <stdlib.h>
#define N_MAX 5

int Sudoku(char nomefile[], int n);
```

```
int main(int argc, char* argv[])
{/* non richiesto per il compito in classe */
  if (argc < 2)
    {
      printf("Errore: uso %s <nomefile> <base>\n", argv[0]);
      exit(-1);
    }
  printf("Errori Sudoku: %d\n", Sudoku(argv[1], atoi(argv[2])));
  return 0;
}

int Sudoku(char nome_file[], int n)
{
  FILE* fp;
  int s[N_MAX*N_MAX][N_MAX*N_MAX], i, j, r, cercato, errori = 0, max_val;

  fp = fopen(nome_file, "r");
  max_val = n * n;
  for (i = 0; i < max_val; i++)
    for (j = 0; j < max_val; j++)
      fscanf(fp, "%d", &s[i][j]);
  fclose(fp);
  for (cercato = 1; cercato <= max_val; cercato++)
    {/* Ricerca per righe */
    for (i = 0; i < max_val; i++)
      {
        int trovato = 0;
        for (j = 0; j < max_val; j++)
          if (s[i][j] == cercato)
            trovato++;
        if (trovato > 1)
          errori++;
      }
    /* Ricerca per colonne */
    for (j = 0; j < max_val; j++)
      {
        int trovato = 0;
        for (i = 0; i < max_val; i++)
          if (s[i][j] == cercato)
            trovato++;
        if (trovato > 1)
          errori++;
      }
    /* Ricerca per riquadri */
    for (r = 0; r < max_val; r++)
      {
        /* r enumera i riquadri in questo modo:
           0 1 2 ... n-1
           n n+1 ... 2n-1
           ... */
        int trovato = 0;
        for (i = r / n * n; i < r / n * n + n; i++)
          for (j = (r % n) * n; j < (r % n + 1) * n; j++) {
            if (s[i][j] == cercato)
              trovato++;
          }
        if (trovato > 1)
          errori++;
```

```
      }
    }
  return errori > 0;
}
```

Soluzione esercizio 2

La soluzione si compone di un semplice ciclo di visita del vettore. Tale ciclo fa uno di una variabile booleana, chiamata `nessuna_sala`, che vale `1` se non è stata ancora trovata una sala della capienza richiesta e `0` altrimenti.

```
#include <stdio.h>

struct Sala
{
  char nome[21];
  int capienza;
  float prezzo;
};

float PrezzoSala(struct Sala v[], int n, int p);

int main()
{ /* non richiesto per il compito in classe */
  int cap;
  float prezzo;
  struct Sala vet[] = { {"Sala Rossini", 30, 200 }, {"Sala Verde", 40, 222.22},
                        {"Saletta X", 10, 50.5}, {"Aula Magna", 100, 300} };

  printf("Capienza minima : ");
  scanf("%d",&cap);
  prezzo = PrezzoSala(vet,4,cap);
  if (prezzo > 0)
    printf("Il prezzo minimo e' %f\n", prezzo);
  else
    printf("Non ci sono sale di capienza %d\n", cap);
  return 0;
}

float PrezzoSala(struct Sala v[], int n, int p)
{
  int i, min;
  int nessuna_sala = 1;

  for (i = 0; i < n; i++)
    {
      if (v[i].capienza >= p)
        {
          if (nessuna_sala)
            {
              nessuna_sala = 0;
              min = i;
            }
          else if (v[i].prezzo < v[min].prezzo)
            min = i;
        }
    }
  if (nessuna_sala)
    return -1.0;
  else
```

```
    return v[min].prezzo;
}
```

Soluzione compito del 30 agosto 2005

L'esercizio 1 si risolve con un doppio ciclo (ciascuno di 3 iterazioni) che esplora il vicinato dell'elemento passato come parametro. Si noti che essendo esplicitamente specificato che l'elemento è interno non è necessario preoccuparsi della possibilità di incorrere in un vicino che sia fuori della matrice.

L'esercizio 2 si risolve con un ciclo che passa da un elemento al suo miglior vicino fino a quando questi sono uguali. A questo scopo si utilizzano due variabili, chiamate `casella_vecchia` e `casella_nuova`, che diventano uguali quando la condizione è soddisfatta.

Per l'esercizio 3 è richiesto esplicitamente di passare il nome del file sulla riga di comando. Si è scelto di passare anche gli altri dati di ingresso sulla riga di comando stessa. La riga e la colonna della matrice, essendo dei valori interi, necessitano di essere convertite da stringhe ad interi (usando `atoi`).

```
#include <stdio.h>
#include <stdlib.h>  /* per atoi() */
#include <assert.h>  /* per assert() */
#define MAX_DIM 10

struct Casella
{
  int riga;
  int colonna;
};

struct Casella MigliorVicino(int m[][MAX_DIM], struct Casella c, int n);
int SalitaRipida(int m[][MAX_DIM], struct Casella s, int n);

int main(int argc, char* argv[])
{
  FILE* fp;
  int i, j, n, max;
  int mat[MAX_DIM][MAX_DIM];
  struct Casella c;

  if (argc != 4)
    {
      printf("Utilizzo: %s <nome_file> <indice_riga> <indice_colonna>\n", argv[0]);
      exit(1);
    }

  c.riga = atoi(argv[2]);
  c.colonna = atoi(argv[3]);
  fp = fopen(argv[1],"r");
  fscanf(fp,"%d",&n);
  n += 2; /* si aggiunge il bordo */
  for (i = 0; i < n; i++)
    for (j = 0; j < n; j++)
      if (i == 0 || i == n-1 || j == 0 || j == n-1)
        mat[i][j] = 0;
      else
        fscanf(fp,"%d",&mat[i][j]);
  fclose(fp);
  max = SalitaRipida(mat,c,n);
  printf("Il massimo raggiunto e' %d\n", max);
  return 0;
}
```

```
struct Casella MigliorVicino(int m[][MAX_DIM], struct Casella c, int n)
{
  int i, j, max;
  struct Casella casella_max = c;
  max = m[c.riga][c.colonna];
  for (i = -1; i <= 1; i++)
    for (j = -1; j <= 1; j++)
      {
        if (m[c.riga+i][c.colonna+j] > max)
          {
            max = m[c.riga+i][c.colonna+j];
            casella_max.riga = c.riga + i;
            casella_max.colonna = c.colonna + j;
          }
      }
  return casella_max;
}

int SalitaRipida(int m[][MAX_DIM], struct Casella sorgente, int n)
{
  struct Casella casella_vecchia, casella_nuova;
  int finito = 0;
  casella_vecchia = sorgente;
  while (!finito)
    {
      casella_nuova = MigliorVicino(m,casella_vecchia,n);
      if (casella_nuova.riga == casella_vecchia.riga
          && casella_nuova.colonna == casella_vecchia.colonna)
        finito = 1;
      else
        casella_vecchia = casella_nuova;
    }
  return m[casella_nuova.riga][casella_nuova.colonna];
}
```

Soluzione compito del 20 settembre 2005

Soluzione esercizio 1

La soluzione consiste in un singolo ciclo di lettura, in cui si scandisce il file riga per riga. Solo in caso di valuta `Dollari` viene letto anche il tasso di cambio.

Si noti l'utilizzo del formato `%*1s`e non `%*c` per poter ignorare gli spazi che precedono la parentesi. Si noti inoltre che appena si incontra una data successiva a quella cercata, si può terminare il ciclo (tramite `break`), perché il testo specifica che le date sono ordinate.

```
#include <stdio.h>
#include <string.h>
#include "../Utils/util_date.h"

float Saldo(char nome_file[], struct Data d);

int main()
{ /* non richiesto per il compito in classe */
  struct Data oggi;
  char nome[21];

  printf("Nome del file : ");
  scanf("%s", nome);
  printf("Data odierna (gg/mm/aaaa): ");
```

```
  scanf("%d%*c%d%*c%d", &oggi.giorno, &oggi.mese, &oggi.anno);
  printf("Il saldo e' %f\n",Saldo(nome,oggi));
  return 0;
}

float Saldo(char nome_file[], struct Data d)
{
  FILE* fp;
  struct Data d1;
  char valuta[8];
  float valore, cambio, saldo = 0.0;

  fp = fopen(nome_file,"r");
  while (fscanf(fp,"%d%*c%d%*c%d%f%s", &d1.giorno, &d1.mese, &d1.anno,
                &valore, valuta) != EOF)
    {
      if (strcmp(valuta,"Dollari") == 0)
        {
          fscanf(fp,"%*1s%f%*c", &cambio);
          valore *= cambio;
        }
      if (ComparaDate(d1,d) == -1)
        saldo += valore;
      else
        break;
    }
  return saldo;
}
```

Soluzione esercizio 2

La via più semplice (e più modulare) per risolvere il problema si ottiene definendo una funzione ausiliaria che verifica la presenza di un elemento in un vettore. Ogni elemento della matrice viene inserito nel vettore, previa verifica della sua assenza nella parte già riempita del vettore stesso (dandogli quindi come dimensione la sua dimensione corrente).

```
#include <stdio.h>
#define MAX_DIM 20

int RiempiVettoreDistinti(int a[][MAX_DIM], int n, int m, int v[]);
int Presente(int e, int v[], int n);

int main()
{ /* non richiesto per il compito in classe */
  int mat[MAX_DIM][MAX_DIM];
  int vet[MAX_DIM * MAX_DIM];
  int i, j, n, m, nv;

  printf("Dimensioni della matrice : ");
  scanf("%d%d",&n,&m);
  printf("Inserisci la matrice (per righe) : ");
  for (i = 0; i < n; i++)
    for (j = 0; j < m; j++)
      scanf("%d",&mat[i][j]);
  nv = RiempiVettoreDistinti(mat,n,m,vet);
  printf("Vettore degli elementi distinti (%d elementi) : ", nv);
  for (i = 0; i < nv; i++)
    printf("%d ", vet[i]);
  printf("\n");
  return 0;
```

```
}

int RiempiVettoreDistinti(int a[][MAX_DIM], int n, int m, int v[])
{
  int i, j, dim = 0;
  for (i = 0; i < n; i++)
    for (j = 0; j < m; j++)
      if (!Presente(a[i][j], v, dim))
        {
          v[dim] = a[i][j];
          dim++;
        }
  return dim;
}

int Presente(int e, int v[], int n)
{
  int i;
  for (i = 0; i < n; i++)
    if (v[i] == e)
      return 1;
  return 0;
}
```

Soluzione compito del 12 dicembre 2005

Soluzione esercizio 1

La soluzione consiste in un ciclo esterno che esamina gli appartamenti uno alla volta, ed un ciclo interno che esamina i locali uno alla volta. L'indirizzo dell'appartamento viene saltato leggendo semplicemente fine al carattere due punti.

Per il locali, essendo il carattere punteggiatura attaccata al nome del locale, questa viene letta nella stessa stringa. Successivamente il carattere viene copiato in una variabile e rimosso della stringa.

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

int NumeroAbitabili(char nome_file[]);

int main(int argc, char* argv[])
{ /* non richiesto per il compito in classe */
  if (argc != 2)
    printf("Utilizzo: %s nome_file\n",argv[0]);
  else
    printf("Il numero di appartamenti abitabili e' %d\n", NumeroAbitabili(argv[1]));
  return 0;
}

int NumeroAbitabili(char nome_file[])
{
  FILE* fp;
  char ch;
  char locale[21];
  int bagni, cucine, abitabili = 0;

  fp = fopen(nome_file,"r");
  do
    {
      bagni = 0;
```

```
      cucine = 0;
      while ((ch = getc(fp)) != ':' && ch != EOF)
        ; /* salta l'indirizzo */
      if (ch != EOF)
        {
          do
            {
              fscanf(fp,"%s", locale);
              ch = locale[strlen(locale) - 1]; /* ch contiene la punteggiatura */
              locale[strlen(locale) - 1] = '\0';
              if (strcmp(locale,"bagno") == 0)
                bagni++;
              else if (strcmp(locale,"cucina") == 0)
                cucine++;
            }
          while (ch == ',');
          if (bagni >= 1 && cucine == 1)
            abitabili++;
        }
    }
  while (ch != EOF);
  return abitabili;
}
```

Soluzione esercizio 2

La soluzione consiste in un semplice ciclo di lettura in cui per ogni misura si cerca l'esponente attraverso una funzione ausiliaria che riceve come parametro il vettore delle potenze.

```
#include <stdio.h>
#include <stdlib.h>
#include <string.h>
#include <math.h>
#define MAXSLEN 128

struct ValorePrefisso
{
   char nome[10];
   int valore;
};

void ConvertiFile(char nome_file_input[], char nome_file_output[],
                  struct ValorePrefisso prefissi[], int n );
int CercaMisura (char s[], struct ValorePrefisso prefissi[], int n );

int main(int argc, char *argv[])
{
   struct ValorePrefisso prefissi[] =
     {
       { "pico", -12 }, { "nano", -9 }, { "micro", -6 }, { "milli", -3 },
       { "unita'", 0 }, { "kilo", 3 }, { "mega", 6 }, { "giga", 9 },
       { "tera", 12 }, { "peta", 15 }
     };
   if (argc != 3)
     {
      printf ("servono due parametri\n");
      exit (-1);
   }
   ConvertiFile(argv[1], argv[2], prefissi, 10);
   return 0;
```

```
}

void ConvertiFile(char nome_file_input[], char nome_file_output[],
                  struct ValorePrefisso prefissi[], int n )
{
  FILE *fp_in, *fp_out;
  float valore;
  char misura[21];
  int esp;

  fp_in = fopen (nome_file_input, "r");
  fp_out = fopen (nome_file_output, "w");
  while (fscanf(fp_in, "%f%s", &valore, misura) != EOF)
    {
      esp = CercaMisura(misura, prefissi, n);
      valore *= pow(10, esp);
      fprintf(fp_out,"%g\n", valore);
    }
}

int CercaMisura (char s[], struct ValorePrefisso prefissi[], int n )
{
   int i = 0;
   for (i = 0; i < n; i++)
     {
       if (strcmp (prefissi[i].nome, s) == 0)
         return prefissi[i].valore;
     }
   return -1; /* prefisso non trovato */
}
```

Soluzione compito del 16 marzo 2006

Soluzione esercizio 1

```
#include <stdio.h>
#include <stdlib.h>
#include <string.h>
#include "../Utils/util_date.h"
int Appuntamento(char nome_file[], struct Data d, char appuntamento[]);

int main(int argc, char* argv[])
{ /* non richiesto per il compito in classe */
  struct Data d;
  char s[31];
  if (argc != 3)
    {
      printf("Errore nel numero di parametri\n");
      exit(1);
    }
  sscanf(argv[2], "%d%*c%d%*c%d", &d.giorno, &d.mese, &d.anno);
  if (Appuntamento(argv[1], d, s))
    printf("Gli appuntamenti sono: %s\n", s);
  else
    printf("La data non esiste\n");
  return 0;
}

int Appuntamento(char nome_file[], struct Data d, char appuntamento[])
```

```
{
  struct Data d1;
  FILE* fp;
  int trovato = 0, i;
  char ch;

  fp = fopen(nome_file, "r");
  while(fscanf(fp,"%d%*c%d%*c%d", &d1.giorno, &d1.mese, &d1.anno) != EOF)
    {
      if (ComparaDate(d,d1) == 0)
        {
          trovato = 1;
          while((ch = getc(fp)) == ' ')
            ;
          if (ch == '*')
            strcpy(appuntamento,"Niente");
          else
            {
              i = 0;
              while((appuntamento[i] = getc(fp)) != ']')
                i++;
              appuntamento[i] = '\0';
            }
          break;
        }
      else
        while(getc(fp) != '\n')
          ;
    }
  return trovato;
}
```

Soluzione esercizio 2

```
#include <stdio.h>

#define MAX_DIM 100

int Fascia(int mat[][MAX_DIM], int n, int m, int k);

int main()
{ /* non richiesto per il compito in classe */
  int mat[MAX_DIM][MAX_DIM], i, j, r, c, k;

  printf("Righe e colonne (max %d) : ", MAX_DIM);
  scanf("%d%d", &r, &c);
  printf("Inserisci la matrice %d X %d (per righe)\n", r, c);
  for (i = 0; i < r; i++)
    for (j = 0; j < c; j++)
      scanf("%d", &mat[i][j]);
  printf("inserisci K :");
  scanf("%d", &k);
  printf("La lunghezza della fascia e' %d\n", Fascia(mat,r,c,k));
  return 0;
}

int Fascia(int mat[][MAX_DIM], int n, int m, int k)
{
  int i, j;
```

```
  int lunghezza = 0;

  for (i = 0; i < n; i++)
    for (j = 0; j < m; j++)
      {
        if (lunghezza > 0)
          if(mat[i][j] == k)
            lunghezza++;
          else
            return lunghezza;
        else
          if (mat[i][j] == k)
            lunghezza = 1;
      }
  return lunghezza;
}
```

Soluzione compito del 10 luglio 2006

Soluzione esercizio 1

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

struct Risultato
{
  char stringa[3];
  double somma_max;
  int num_righe;
};

struct Risultato SommaMassima(char nome_file[]);

int main(int argc, char* argv[])
{ /* non richiesto per il compito in classe */
  struct Risultato ris;
  ris = SommaMassima(argv[1]);
  if (strcmp(ris.stringa,"---") == 0)
    printf("Nessun esperimento significativo\n");
  else
    printf("Il risultato e' %s %g %d\n", ris.stringa, ris.somma_max, ris.num_righe);
  return 0;
}

struct Risultato SommaMassima(char nome_file[])
{
  struct Risultato ris;
  FILE* fp;
  char s[5], ch;
  int i;
  double val, somma;

  ris.num_righe = 0;
  strcpy(ris.stringa,"---");
  fp = fopen(nome_file, "r");
  while(fscanf(fp,"%s", s) != EOF)
    {
      ris.num_righe++;
```

```
      somma = 0.0;
      i = 0;
      do
        {
          fscanf(fp, "%lg%c", &val, &ch);
          somma += val;
          i++;
        }
      while (ch == ',');
      if (i >= 3 &&
  /* e' il primo esperimento valido oppure ha somma maggiore */
  (strcmp(ris.stringa,"---") == 0 || somma > ris.somma_max))
        {
          s[3] = '\0'; /* rimuove la virgola finale */
          strcpy(ris.stringa,s);
          ris.somma_max = somma;
        }
    }
  return ris;
}
```

Soluzione esercizio 2

```
#include <stdio.h>
#define MONETE 8

void CalcolaMonete(int somma, int valore[], int monete[], int n);

int main(int argc, char* argv[])
{ /* non richiesto per il compito in classe */
  int s, i;
  int val[MONETE] = {1, 2, 5, 10, 20, 50, 100, 200};
  int mon[MONETE];

  printf("Somma : ");
  scanf("%d", &s);
  CalcolaMonete(s, val, mon, MONETE);
  printf("Monete: ");
  for (i = 0; i < MONETE; i++)
    printf("%d da %d, ", mon[i], val[i]);
  printf("\n");
  return 0;
}

void CalcolaMonete(int somma, int valore[], int monete[], int n)
{
  int i, k = n-1;
  int somma_residua = somma;

  for (i = 0; i < n; i++)
    monete[i] = 0;
  while (somma_residua > 0)
    {
      if (somma_residua >= valore[k])
        {
          somma_residua -= valore[k];
          monete[k]++;
        }
      else
```

```
        k--;
    }
}
```

Soluzione compito del 6 settembre 2006

Soluzione esercizio 1

```
#include <stdio.h>
#define MAX_COL 100

void StampaScacchiera(int M[][MAX_COL], int n);
int NRegine(int M[][MAX_COL], int n);

int main()
{ /* non richiesto per il compito in classe */
  int M[MAX_COL][MAX_COL], dim, i, j, risultato;

  printf("Immetti la dimensione della scacchiera: ");
  scanf("%d", &dim);
  for (i = 0; i < dim; i++)
    {
      printf("Immetti la riga %d-esima: ", i);
      for (j = 0; j < dim; j++)
        scanf("%d", &M[i][j]);
    }
  StampaScacchiera(M, dim);
  risultato = NRegine(M, dim);
  if (risultato == -1)
    printf("Non tutte le regine sono presenti\n");
  else if (risultato > 0)
    printf("Ci sono %d regine che si attaccano\n", risultato);
  else
    printf("La scacchiera e' in equilibrio\n");
  return 0;
}

void StampaScacchiera(int M[][MAX_COL], int n)
{
  int i, j;
  printf("\n");
  for (i = 0; i < n; i++)
    {
      for (j = 0; j < n; j++)
        if (M[i][j])
          printf("Q");
        else
          printf("-");
      printf("\n");
    }
}

int NRegine(int M[][MAX_COL], int n)
{
  int i, j, k, l, numero_regine = 0, attacchi = 0;

  for (i = 0; i < n; i++)
    {
      /* cerca la regina sulla riga i */
```

```
      for (j = 0; j < n; j++)
        if (M[i][j])
          break;
      if (j < n)
        {
          numero_regine++;
          /* attacchi sulla riga */
          for (k = 0; k < n; k++)
            if (k != j && M[i][k])
              attacchi++;
          /* attacchi sulla colonna */
          for (k = 0; k < n; k++)
            if (k != i && M[k][j])
              attacchi++;
          /* attacchi sulla diagonale principale */
          k = i - 1; l = j - 1;
          /* in alto a sinistra */
          while (k >= 0 && l >= 0)
            {
              if (M[k][l])
                attacchi++;
              k--; l--;
            }
          k = i + 1; l = j + 1;
          /* in basso a destra */
          while (k < n && l < n)
            {
              if (M[k][l])
                attacchi++;
              k++; l++;
            }
          /* attacchi sulla diagonale secondaria */
          k = i - 1; l = j + 1;
          /* in alto a destra */
          while (k >= 0 && l < n)
            {
              if (M[k][l])
                attacchi++;
              k--; l++;
            }
          k = i + 1; l = j - 1;
          /* in basso a sinistra */
          while (k < n && l >= 0)
            {
              if (M[k][l])
                attacchi++;
              k++; l--;
            }
        }
    }
  if (numero_regine != n)
    return -1;
  else
    return attacchi;
}
```

Soluzione esercizio 2

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

```
void StatisticheNascite(char* nome_file, int nascite[]);

int main()
{ /* non richiesto per il compito in classe */
  char nome[21];
  int n[12], i;

  printf("Immetti il nome del file da elaborare: ");
  scanf("%s", nome);
  StatisticheNascite(nome, n);
  for (i = 0; i < 12; i++)
    printf("%d ", n[i]);
  printf("\n");
  return 0;
}

void StatisticheNascite(char* nome_file, int nascite[])
{
  FILE* fp;
  int mese, i;
  char ch;

  for (i = 0; i < 12; i++)
    nascite[i] = 0;
  fp = fopen(nome_file, "r");
  while ((ch = fgetc(fp)) != EOF)
    {
      /* ignora il primo campo */
      while ((ch = fgetc(fp)) != ',')
        ;
      /* ignora il secondo campo  */
      while ((ch = fgetc(fp)) != ',')
        ;
      fscanf(fp, "%*d%*c%d", &mese);
      nascite[mese]++;
      /* ignora tutto fino alla fine della riga  */
      while ((ch = fgetc(fp)) != '\n')
        ;
    }
  fclose(fp);
}
```

Soluzione compito del 22 settembre 2006

Soluzione esercizio 1

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

int AppuntamentoLungo(char nome_file_input[]);
int NumGiornoSettimana(char giorno[]);

int main(int argc, char* argv[])
{ /* non richiesto per il compito in classe */
  if (argc != 2)
    {
      printf("Errore nei parametri\n");
```

```
      exit(1);
    }
  printf("L'appuntamento piu' lungo e' di %d ore\n", AppuntamentoLungo(argv[1]));
  return 0;
}

int AppuntamentoLungo(char nome_file_input[])
{
  FILE* fp;
  int ora_inizio, ora_fine;
  char giorno_inizio[11], giorno_fine[11];
  int num_giorni, num_ore, max_num_ore = 0;

  fp = fopen(nome_file_input,"r");
  while(fscanf(fp,"%*s%*s%s%*s%d%*s%s%*s%d",giorno_inizio,
        &ora_inizio, giorno_fine, &ora_fine) != EOF)
    {
      num_giorni = NumGiornoSettimana(giorno_fine) - NumGiornoSettimana(giorno_inizio);
      num_ore = 24 * num_giorni + (ora_fine - ora_inizio);
      if (num_ore > max_num_ore)
        max_num_ore = num_ore;
    }
  fclose(fp);
  return max_num_ore;
}

int NumGiornoSettimana(char giorno[])
{
  if (strcmp(giorno,"Lunedi'") == 0)
    return 1;
  else if (strcmp(giorno,"Martedi'") == 0)
    return 2;
  else if (strcmp(giorno,"Mercoledi'") == 0)
    return 3;
  else if (strcmp(giorno,"Giovedi'") == 0)
    return 4;
  else if (strcmp(giorno,"Venerdi'") == 0)
    return 5;
  else if (strcmp(giorno,"Sabato") == 0)
    return 6;
  else /* giorno = Domenica */
    return 7;
}
```

Soluzione esercizio 2

```
#include <stdio.h>
#include <stdlib.h>
#include <string.h>
#define MAX_DIM 100
#define LUNG_NOME 20

struct Partita
{
  char sq_casa[LUNG_NOME];
  char sq_ospite[LUNG_NOME];
  int gol_casa;
  int gol_ospite;
};
```

```
struct Risultato
{
  int vittorie_casa;
  int vittorie_ospite;
  int pareggi;
  char squadra[LUNG_NOME];
};

struct Risultato CalcolaRisultati(struct Partita v[], int n);
int LeggiDati(char nome_file[], struct Partita v[]);

int main(int argc, char* argv[])
{ /* non richiesto per il compito */
  int dim;
  struct Partita vet[MAX_DIM];
  struct Risultato dati;
  dim = LeggiDati(argv[1], vet);
  dati = CalcolaRisultati(vet,dim);
  printf("Vittorie casa : %d\nVittorie ospite : %d\nPareggi : %d\nSquadra : %s\n",
         dati.vittorie_casa, dati.vittorie_ospite, dati.pareggi, dati.squadra);
  return 0;
}

int LeggiDati(char nome_file[], struct Partita v[])
{ /* non richiesto per il compito */
  int i = 0;
  FILE *fp;

  fp = fopen(nome_file,"r");
  while (fscanf(fp, "%s%s%d%d",
         v[i].sq_casa, v[i].sq_ospite, &v[i].gol_casa, &v[i].gol_ospite) != EOF)
    i++;
  return i;
}

struct Risultato CalcolaRisultati(struct Partita v[], int n)
{
  struct Risultato ris;

  ris.vittorie_casa = 0;
  ris.vittorie_ospite = 0;
  ris.pareggi = 0;
  int max_gol = 0, i;
  for (i = 0; i < n; i++)
    {
      if (v[i].gol_casa > v[i].gol_ospite)
        ris.vittorie_casa++;
      else if (v[i].gol_casa < v[i].gol_ospite)
        ris.vittorie_ospite++;
      else
        ris.pareggi++;
      if (v[i].gol_casa > max_gol)
        {
          max_gol = v[i].gol_casa;
          strcpy(ris.squadra, v[i].sq_casa);
        }
      if (v[i].gol_ospite > max_gol)
        {
```

```
          max_gol = v[i].gol_ospite;
          strcpy(ris.squadra, v[i].sq_ospite);
        }
    }
  return ris;
}
```

Soluzione compito del 18 marzo 2008

Soluzione esercizio 1

```
#include <stdio.h>
#include <stdlib.h>
#define MAX_DIM 100

struct Articolo
{
  char nome[21];
  int numero;
};

int TrasformaFile(char nome_file_input[], char nome_file_output[], float soglia);
int LeggiVettore(char nome_file[], struct Articolo v[]);

int main(int argc, char* argv[])
{ /* non richiesto per il compito in classe */
  int k;
  if (argc != 4)
    printf("Utilizzo: %s <nome_file_input> <nome_file_output> <soglia>\n", argv[0]);
  else
    {
      k = TrasformaFile(argv[1],argv[2],atof(argv[3]));
      printf("Il nuovo file contiene %d elementi\n",k);
    }
  return 0;
}

int TrasformaFile(char nome_file_input[], char nome_file_output[], float soglia)
{
  int i = 0, n, m = 0, totale = 0;
  float percentuale;
  struct Articolo articoli[MAX_DIM];
  FILE* fp;

  n = LeggiVettore(nome_file_input, articoli);
  for (i = 0; i < n; i++)
    totale += articoli[i].numero;
  fp = fopen(nome_file_output,"w");
  for (i = 0; i < n; i++)
    {
      percentuale = 100.0 * articoli[i].numero / totale;
      if (percentuale > soglia)
        {
          fprintf(fp,"%s: %.2f\n",articoli[i].nome,percentuale);
          m++;
        }
    }
  fclose(fp);
  return m;
```

```
}

int LeggiVettore(char nome_file[], struct Articolo v[])
{
  int i = 0, j;
  FILE* fp;

  fp = fopen(nome_file,"r");
  while(fscanf(fp,"%d",&v[i].numero) != EOF)
    {
      j = 0;
      getc(fp);
      while ((v[i].nome[j] = getc(fp)) != '\n')
        j++;
      v[i].nome[j] = '\0';
      i++;
    }
  return i;
}
```

Soluzione esercizio 2

```
#include <stdio.h>
#include <math.h>
#define MAX_DIM 10

struct Punto
{
  double X;
  double Y;
  double Z;
};

struct Triangolo
{
  struct Punto p1;
  struct Punto p2;
  struct Punto p3;
};

int AreaMassima(struct Triangolo v[], int n);
double LunghezzaSegmento(struct Punto s1, struct Punto s2);
double Area(struct Triangolo t);

int main()
{ /* non richiesto per il compito in classe */
  struct Triangolo v[MAX_DIM];
  int i,n;
  printf("Numero di triangoli: ");
  scanf("%d", &n);

  for (i = 0; i < n; i++)
    {
      printf("Triangolo di indice %d\n", i);
      printf("Coordinate primo vertice : ");
      scanf("%lg%lg%lg",&v[i].p1.X,&v[i].p1.Y,&v[i].p1.Z);
      printf("Coordinate secondo vertice : ");
      scanf("%lg%lg%lg",&v[i].p2.X,&v[i].p2.Y,&v[i].p2.Z);
      printf("Coordinate terzo vertice : ");
```

```
      scanf("%lg%lg%lg",&v[i].p3.X,&v[i].p3.Y,&v[i].p3.Z);
    }
  printf("Il triangolo di area massima e' quello di indice %d\n", AreaMassima(v,n));
  return 0;
}

int AreaMassima(struct Triangolo v[], int n)
{
  int i, max = 0;
  double area, area_max = Area(v[0]);

  for (i = 1; i < n; i++)
    {
      area = Area(v[i]);
      if (area > area_max)
        {
          max = i;
          area_max = area;
        }
    }
  return max;
}

double LunghezzaSegmento(struct Punto s1, struct Punto s2)
{
  return sqrt((s1.X - s2.X) * (s1.X - s2.X)
              + (s1.Y - s2.Y) * (s1.Y - s2.Y)
              + (s1.Z - s2.Z) * (s1.Z - s2.Z));
}

double Area(struct Triangolo t)
{
  double a,b,c,s;

  a = LunghezzaSegmento(t.p1,t.p2);
  b = LunghezzaSegmento(t.p1,t.p3);
  c = LunghezzaSegmento(t.p2,t.p3);
  s = (a+b+c)/2;
  return sqrt(s * (s - a) * (s - b) * (s - c));
}
```

Soluzione compito del 7 aprile 2008

Soluzione esercizio 1

```
#include <stdio.h>
#include <string.h>
#define MAX_DIM 30

struct Voti
{
  char candidato[MAX_DIM+1];
  int num_voti;
};

int Vincitore(char nome_file[], char nome_vincitore[]);
int IndicePosizione(char nome[], struct Voti v[], int n);

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

```
{ /* non richiesto per il compito in classe */
  char nome[MAX_DIM+1];
  int v = Vincitore(argv[1], nome);
  printf("Il piu' votato e' %s (con voti %d)\n", nome, v);
  return 0;
}

int Vincitore(char nome_file[], char nome_vincitore[])
{
  int n = 0, i, k, max;
  char nome[MAX_DIM+1];
  struct Voti voti[100];
  FILE* fp;

  fp = fopen(nome_file,"r");
  while(fscanf(fp,"%s",nome) != EOF)
    {
      k = IndicePosizione(nome,voti,n);
      if (k == -1)
        {
          strcpy(voti[n].candidato,nome);
          voti[n].num_voti = 1;
          n++;
        }
      else
        voti[k].num_voti++;
    }

  fclose(fp);
  max = 0;
  for (i = 1; i < n; i++)
    if (voti[i].num_voti > voti[max].num_voti)
      max = i;
  strcpy(nome_vincitore,voti[max].candidato);
  return voti[max].num_voti;
}

int IndicePosizione(char nome[], struct Voti v[], int n)
{
  int i;

  for (i = 0; i < n; i++)
    if (strcmp(nome,v[i].candidato) == 0)
      return i;
  return -1;
}
```

Soluzione esercizio 2

```
#include <stdio.h>
#include <time.h>
#include <stdlib.h>
#define DIM 100

int SequenzaVerticale(int m[][DIM], int n, int k);
int Verifica(int i, int j, int k, int m[][DIM]);
int FX(int v[], int n);

int main()
```

```
{ /* non richiesto per il compito in classe */
  int i,j,k,n,mat[DIM][DIM];

  printf("Dimensione della matrice: ");
  scanf("%d",&n);
  printf("Lunghezza cercata : ");
  scanf("%d",&k);
  srand(time(NULL));
  for (i = 0; i < n; i++)
    {
      for (j = 0; j < n; j++)
        {
          mat[i][j] = rand() % (2*n);
          printf("%3d ", mat[i][j]);
        }
      printf("\n");
    }
  printf("\n");
  if (SequenzaVerticale(mat,n,k))
    printf("La sequenza esiste\n");
  else
    printf("La sequenza non esiste\n");
  return 0;
}

int SequenzaVerticale(int m[][DIM], int n, int k)
{
  int i, j;

  for (i = 0; i < n-k+1; i++)
    for (j = 0; j < n; j++)
      if (Verifica(i,j,k,m))
        return 1;
  return 0;
}

int Verifica(int i, int j, int k, int m[][DIM])
{
  int h, v[DIM];
  for (h = 0; h < k; h++)
    v[h] = m[i+h][j];
  return FX(v,k);
}

int FX(int v[], int n)
{/* funzione di prova: tutti uguali */
  int i;
  for (i = 0; i < n - 1; i++)
    if (v[i] > v[i+1])
      return 0;
  return 1;
}
```

Soluzione compito del 26 giugno 2008

Soluzione esercizio 1

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

```
#include <string.h>

int ConvertiMese(char stringa_mese[]);

int TrovaMesePubblicazione(char nome_file[], int num_citazione);

int main(int argc, char* argv[])
{ /* non richiesto per il compito in classe */
  int mese_pubblicazione, numero;

  if (argc != 3)
    {
      printf("Utilizzo: %s <nome_file> <num_citazione>", argv[0]);
      exit(1);
    }
  numero = atoi(argv[2]);
  mese_pubblicazione = TrovaMesePubblicazione(argv[1], numero);
  if (mese_pubblicazione != 0)
    printf("Il mese di pubblicazione e' %d\n", mese_pubblicazione);
  else
    printf("Pubblicazione inesistente\n");
  return 0;
}

int ConvertiMese(char stringa_mese[])
{
  int i;
  char * mesi_in_inglese[12] = {"January","February","March","April","May","June",
                                "July","August","September","October","November","December"};

  for (i = 0; i < 12; i++)
    if (strcmp(stringa_mese,mesi_in_inglese[i]) == 0)
      return i+1;
  return -1; /* Errore */
}

int TrovaMesePubblicazione(char nome_file[], int num_citazione)
{
  FILE* fp;
  char mese[21];
  int numero_letto;

  fp = fopen(nome_file,"r");
  while (fscanf(fp,"%*c%d%*c",&numero_letto) != EOF)
    {
      if (numero_letto < num_citazione)
        while (getc(fp) != '\n')
          ;
      else if (numero_letto == num_citazione)
        { /* numero trovato */
          while (getc(fp) != '.')
            ;
          fscanf(fp,"%s", mese);
          mese[strlen(mese) - 1] = '\0'; /* rimuove la virgola */
          break;
        }
      else /* numero superato */
        break;
    }
```

```
  fclose(fp);
  if (numero_letto == num_citazione)
    return ConvertiMese(mese);
  else
    return 0;
}
```

Soluzione esercizio 2

```
#include <stdio.h>
#include <string.h>
#define PARTITE 4

struct Set
{
  int punti_casa;
  int punti_ospiti;
};

struct Partita
{
  char squadra_casa[21];
  char squadra_ospite[21];
  struct Set punti_set[5];
};

int PartiteVinte(struct Partita v[], int n, char nome[]);
int Vincitore(struct Set ps[]);

int main()
{ /* non richiesto per il compito in classe */
  struct Partita v[PARTITE];
  int i, j;
  char nome[21];

  for (i = 0; i < PARTITE; i++)
    {
      printf("Immetti i dati della partita %d (su %d)\n", i+1, PARTITE);
      printf("Nome squadra di casa: ");
      scanf("%s", v[i].squadra_casa);
      printf("Nome squadra ospite: ");
      scanf("%s", v[i].squadra_ospite);
      for (j = 0; j < 5; j++)
        {
          printf("Risultato set %d :", j+1);
          scanf("%d%d", &v[i].punti_set[j].punti_casa, &v[i].punti_set[j].punti_ospiti);
        }
    }
  do
    {
      printf("Immetti il nome della squadra di interesse (inserisci un punto per terminare): ");
      scanf("%s", nome);
      if (strcmp(nome, ".") == 0)
        break;
      printf("Il numero di partite vinte dalla suqadra %s e' %d\n",
             nome, PartiteVinte(v, PARTITE, nome));
    }
  while (1);
  return 0;
```

```
}

int PartiteVinte(struct Partita v[], int n, char nome[])
{
  int i, vinte = 0;
  for (i = 0; i < n; i++)
    if ((strcmp(v[i].squadra_casa, nome) == 0 && Vincitore(v[i].punti_set) == 1)
     || (strcmp(v[i].squadra_ospite, nome) == 0 && Vincitore(v[i].punti_set) == 2) )
        vinte++;
  return vinte;
}

int Vincitore(struct Set ps[])
{
  int i, vinti1 = 0, vinti2 = 0;
  for (i = 0; i < 5; i++)
    {
      if (ps[i].punti_casa > ps[i].punti_ospiti)
        vinti1++;
      else if (ps[i].punti_casa < ps[i].punti_ospiti)
        vinti2++;
    }
  if (vinti1 > vinti2)
    return 1;
  else
    return 2;
}
```

Soluzione compito del 11 settembre 2008

Soluzione esercizio 1

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

#define DIM_MAX 100

struct Camera
{
  int num_letti;
  int piano;
  int libera;
};

struct Albergo
{
  int categoria;
  int num_camere;
  struct Camera vet_camere[DIM_MAX];
};

int TrovaCamere(struct Albergo alberghi[], int n, int cat, int piano, int letti);

int main(int argc, char *argv[])
{ /* non richiesto per il compito in classe */
  struct Albergo alberghi[10];
  int i, j, num_camere, num_alberghi;
  FILE* fp;
  fp = fopen(argv[1],"r");
```

```
  fscanf(fp,"%d",&num_alberghi);
  for(i = 0; i < num_alberghi; i++)
    {
      fscanf(fp,"%d%d",&alberghi[i].categoria,&alberghi[i].num_camere);
      for(j = 0; j < alberghi[i].num_camere; j++)
        fscanf(fp,"%d%d%d",
               &alberghi[i].vet_camere[j].num_letti,
               &alberghi[i].vet_camere[j].piano,
               &alberghi[i].vet_camere[j].libera);
    }

  num_camere = TrovaCamere(alberghi,num_alberghi,atoi(argv[2]),atoi(argv[3]),atoi(argv[4]));
  printf("Numero camere disponibili : %d\n", num_camere);
  return 0;
}

int TrovaCamere(struct Albergo alberghi[], int n, int cat, int piano, int letti)
{
  int i, j, num_camere_disponibili = 0;

  for(i = 0; i < n; i++)
    {
      if (alberghi[i].categoria >= cat)
        for(j = 0; j < alberghi[i].num_camere; j++)
          if (alberghi[i].vet_camere[j].libera
              && alberghi[i].vet_camere[j].num_letti >= letti
              &&alberghi[i].vet_camere[j].piano == piano)
            num_camere_disponibili++;
    }
  return num_camere_disponibili;
}
```

Soluzione esercizio 2

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

#define DIM_MAX 100

struct DatiColonne
{
  int i_min, i_max;
  double min, max;
};
struct DatiColonne CalcolaDatiColonne(char nome_file[]);
double SommaColonna(double m[][DIM_MAX], int righe, int colonne, int col);

int main(int argc, char *argv[])
{ /* non richiesto per il compito in classe */
  struct DatiColonne ris;
  ris = CalcolaDatiColonne(argv[1]);
  printf("Minimo: %g (colonna %d)\n",  ris.min, ris.i_min);
  printf("Massimo: %g (colonna %d)\n",  ris.max, ris.i_max);
  return 0;
}

struct DatiColonne CalcolaDatiColonne(char nome_file[])
{
```

```
  int i,j,r,c;
  FILE* fp;
  double mat[DIM_MAX][DIM_MAX], val;
  struct DatiColonne ris;

  fp = fopen(nome_file,"r");
  fscanf(fp,"%d%d",&r,&c);
  for(i = 0; i < r; i++)
    for(j = 0; j < c; j++)
      fscanf(fp,"%lg",&mat[i][j]);
  fclose(fp);
  ris.i_min = 0;
  ris.i_max = 0;
  ris.min = SommaColonna(mat,r,c,0);
  ris.max = ris.min;
  for(j = 1; j < c; j++)
    {
      val = SommaColonna(mat,r,c,j);
      if (val > ris.max)
        {
          ris.max = val;
          ris.i_max = j;
        }
      else if (val < ris.min)
        {
          ris.min = val;
          ris.i_min = j;
        }
    }
  return ris;
}

double SommaColonna(double m[][DIM_MAX], int righe, int colonne, int col)
{
  double somma = 0;
  int i;

  for (i = 0; i < righe; i++)
    somma += m[i][col];
  return somma;
}
```

Soluzione compito del 22 gennaio 2009

Soluzione esercizio 1

```
#include <stdio.h>
#include <stdlib.h>
#include <string.h>
#include "../Utils/util_date.h"

void TrovaGiovane(char nome_file[], char cognome_restituito[]);

int main(int argc, char* argv[])
{/* non richiesto per il compito in classe */
  char cognome[31];
  if (argc != 2)
    {
      printf("Inserire il nome del file\n");
```

```
      exit(1);
    }
  else
    {
      TrovaGiovane(argv[1], cognome);
      printf("Il cognome del piu' giovane e' %s\n", cognome);
    }
  return 0;
}

void TrovaGiovane(char nome_file[], char cognome_restituito[])
{
  char cognome[31], ch;
  FILE* fp;
  int primo = 1, i;
  struct Data data, data_min;

  fp = fopen(nome_file,"r");
  while (fscanf(fp,"%*s%*s") != EOF)
    {
      /* leggo il nome (senza memorizzarlo) */
      while (getc(fp) != ',')
        ;
      getc(fp);
      i = 0;
      /* leggo il cognome */
      while ((cognome[i] = getc(fp)) != ',')
        i++;
      cognome[i] = '\0';
      /* leggo la data */
      fscanf(fp,"%d%*c%d%*c%d", &data.giorno, &data.mese, &data.anno);
      if (primo || ComparaDate(data, data_min) == 1)
        {
          strcpy(cognome_restituito, cognome);
          data_min = data;
          primo = 0;
        }
      while ((ch = getc(fp)) != '\n' && ch != EOF)
        ;
    }
}
```

Soluzione esercizio 2

```
#include <stdio.h>
#define MAX_DIM 50

int RiempiVettoriFreqeunze(int a[][MAX_DIM], int n, int elementi[], int frequenze[]);
int PosizioneElemento(int vet[], int dim, int el);

int main()
{ /* non richiesto per il compito in classe */
  int mat[MAX_DIM][MAX_DIM];
  int vet[MAX_DIM * MAX_DIM], freq[MAX_DIM * MAX_DIM];
  int i, j, n, nv;

  printf("Dimensione della matrice : ");
  scanf("%d",&n);
  printf("Inserisci la matrice (per righe) : ");
```

```
  for (i = 0; i < n; i++)
    for (j = 0; j < n; j++)
      scanf("%d",&mat[i][j]);
  nv = RiempiVettoriFreqeunze(mat,n,vet,freq);
  printf("Vettore degli elementi distinti (%d elementi):\n", nv);
  for (i = 0; i < nv; i++)
    printf("%d (%d volte)\n", vet[i], freq[i]);
  printf("\n");
  return 0;
}

int RiempiVettoriFreqeunze(int a[][MAX_DIM], int n, int elementi[], int frequenze[])
{
  int i, j, k, dim = 0;
  for (i = 0; i < n; i++)
    for (j = 0; j < n; j++)
      {
        k = PosizioneElemento(elementi, dim, a[i][j]);
        if (k == -1)
          {
            elementi[dim] = a[i][j];
            frequenze[dim] = 1;
            dim++;
          }
        else
          frequenze[k]++;
      }
  return dim;
}

int PosizioneElemento(int vet[], int dim, int el)
{
  int i;
  for (i = 0; i < dim; i++)
    if (vet[i] == el)
      return i;
  return -1;
}
```

Soluzione compito del 23 febbraio 2009

Soluzione esercizio 1

```
#include <stdio.h>
#include <string.h>
#define DEPOSITI 5

struct Deposito
{
  char locazione[21];
  int num_veicoli;
};

int EseguiSpostamenti(char nome_file[], struct Deposito depositi[], int n);
int CercaLocazione(struct Deposito deposito[], int n, char locazione[]);

int main(int argc, char* argv[])
{ /* non richiesto per il compito in classe */
  int ris, i;
```

```
  struct Deposito depositi[DEPOSITI] = {{"Roma",3}, {"Milano",8},
                  {"Torino",5}, {"Genova",10}, {"Napoli", 11}};

  ris = EseguiSpostamenti(argv[1], depositi, DEPOSITI);
  if (ris == 1)
    printf("Locazione sconosciuta nel file\n");
  else if (ris == 2)
    printf("Spostamento impossibile\n");
  else
    for (i = 0; i < DEPOSITI; i++)
      printf("%s : %d\n", depositi[i].locazione, depositi[i].num_veicoli);
  return 0;
}

int EseguiSpostamenti(char nome_file[], struct Deposito depositi[], int n)
{
  char sorgente[21], destinazione[21];
  FILE* fp;
  int s, d;

  fp = fopen(nome_file,"r");
  while (fscanf(fp,"%s%*s%s", sorgente, destinazione) != EOF)
    {
      s = CercaLocazione(depositi,n,sorgente);
      d = CercaLocazione(depositi,n,destinazione);
      if (s == -1 || d == -1)
        return 1;
      depositi[s].num_veicoli--;
      depositi[d].num_veicoli++;
      if (depositi[s].num_veicoli < 0)
        return 2;
    }
  fclose(fp);
  return 0;
}

int CercaLocazione(struct Deposito deposito[], int n, char locazione[])
{
  int i;
  for (i = 0; i < n; i++)
    if (strcmp(deposito[i].locazione, locazione) == 0)
      return i;
  return -1;
}
```

Soluzione esercizio 2

La funzione inizialmente inserisce il giorno nella nuova stringa `giorno` distinguendo i due casi: con lo `0` davanti e senza. Successivamente vengono copiato il mese e l'anno nelle stringhe `mese` e `anno`. La traduzione del mese in inglese e il calcolo del suffisso vengono delegati a due funzioni ausiliarie. Infine, tutte le parti della stringa di uscita (mese in inglese, giorno, suffisso e anno) vengono concatenate insieme utilizzando la funzione `strcat`.

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

void TraduciData(char s_in[], char s_out[]);
```

```
void TraduciMese(char me[], char mo[]);
void CalcolaSuffisso(char giorno[], char suffisso[]);

int main(int argc, char* argv[])
{
  char risultato[31];
  TraduciData(argv[1], risultato);
  printf("Stringa tradotta: %s\n",risultato);
  return 0;
}

void TraduciData(char s_in[], char s_out[])
{
  int i = 3, j = 0;
  char giorno[3], mese[20], month[20], anno[5], suffisso[3];

  /* estrae il giorno */
  if (s_in[0] == '0')
    {
      giorno[0] = s_in[1];
      giorno[1] = '\0';
    }
  else
    {
      giorno[0] = s_in[0];
      giorno[1] = s_in[1];
      giorno[2] = '\0';
    }
/* estrae il mese */
  while (s_in[i] != ' ')
    {
      mese[j] = s_in[i];
      i++;
      j++;
    }
  mese[j] = '\0';
  j = 0;
  i++;
  while(s_in[i] != '\0')
    {
      anno[j] = s_in[i];
      i++;
      j++;
    }
  anno[j] = '\0';

  TraduciMese(mese,month);
  CalcolaSuffisso(giorno,suffisso);

  strcpy(s_out,month);
  strcat(s_out," ");
  strcat(s_out,giorno);
  strcat(s_out,suffisso);
  strcat(s_out,", ");
  strcat(s_out,anno);
}

void TraduciMese(char me[], char mo[])
{
```

```
  char* mesi[12] = {"gennaio", "febbraio", "marzo", "aprile", "maggio",
                    "giugno", "luglio", "agosto", "settembre", "ottobre",
                    "novembre", "dicembre"};
  char* months[12] = {"January", "February", "March", "April", "May",
                      "June", "July", "August", "September", "October",
                      "November", "December"};
  int i;

  for (i = 0; i < 12; i++)
    if (strcmp(me,mesi[i]) == 0)
      {
        strcpy(mo,months[i]);
        return;
      }
  strcpy(mo,"Unknown");
}

void CalcolaSuffisso(char giorno[], char suffisso[])
{
  switch(atoi(giorno))
    {
    case 1: case 21: case 31:
      strcpy(suffisso,"st");
      break;
    case 2: case 22:
      strcpy(suffisso,"nd");
      break;
    case 3: case 23:
      strcpy(suffisso,"rd");
      break;
    default:
      strcpy(suffisso,"th");
      break;
    }
}
```

Soluzione compito del 14 giugno 2012

Soluzione esercizio 1

Per memorizzare la posizione geografica del mezzo si utilizza un tipo record chiamato `Punto`. Un secondo tipo record è costruito per restituire il risultato della funzione. Il tipo `Punto` è utilizzato anche per i parametri della funzione `Distanza`, la cui definizione consente una migliore modularizzazione.

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

struct Dati
{
  int num_clienti;
  float km_percorsi;
};

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

```
struct Dati CalcolaPercorso(char nome_file[], int capienza);
float Distanza(struct Punto p1, struct Punto p2);

int main(int argc, char* argv[])
{/* non richiesto per il compito in classe */
  struct Dati ris;
  if (argc != 3)
    {
      printf("Utilizzo: %s <file> <capienza>\n", argv[0]);
      exit(1);
    }
  ris = CalcolaPercorso(argv[1], atoi(argv[2]));
  printf("Clienti serviti: %d, Km percorsi: %.2f\n", ris.num_clienti, ris.km_percorsi);
  return 0;
}

struct Dati CalcolaPercorso(char nome_file[], int capienza)
{
  FILE* fp;
  struct Punto deposito = {0.0, 0.0}, pos_attuale, pos_precedente = deposito;
  int carico, carico_totale = 0;
  struct Dati ris = {0,0.0};

  fp = fopen(nome_file,"r");

  while(fscanf(fp,"%f%f%d", &pos_attuale.x, &pos_attuale.y, &carico) != EOF)
    {
      if (carico_totale + carico > capienza)
        break;
      else
        {
          ris.km_percorsi += Distanza(pos_attuale, pos_precedente);
          carico_totale += carico;
          ris.num_clienti++;
          pos_precedente = pos_attuale;
        }
    }
  fclose(fp);
  ris.km_percorsi += Distanza(pos_precedente, deposito);
  return ris;
}

float Distanza(struct Punto p1, struct Punto p2)
{
  return sqrt((p2.x - p1.x) * (p2.x - p1.x) + (p2.y - p1.y) * (p2.y - p1.y));
}
```

Soluzione esercizi 2 e 3

La funzione si avvale di una funzione ausiliaria che calcola i punti ottenuti dalla squadra in base ai set vinti dalle due squadre. Si noti che viene visitata solo la parte triangolare superiore della matrice, in quanto entrambi gli indici sono considerati per ogni cella.

Si noti inoltre che è necessario azzerare il vettore `v` all'interno della funzione `CalcolaPunti`. Azzerare il vettore nella funzione `main`, creerebbe un inutile accoppiamento tra le funzioni.

```
#include <stdio.h>
#include <stdlib.h>
#define DIM 100

void CalcolaPunti(int ris[][DIM], int n, int v[]);
```

```
int Punti(int set1, int set2);

int main(int argc, char* argv[])
{
  int set[DIM][DIM];
  int v[DIM];
  int i, j, n;
  FILE* fp;

  if (argc != 2)
    {
      printf("Inserire il nome del file!");
      exit(1);
    }

  fp = fopen(argv[1], "r");
  fscanf(fp, "%d",&n);
  for (i = 0; i < n; i++)
    for (j = 0; j < n; j++)
      fscanf(fp, "%d", &set[i][j]);

  CalcolaPunti(set,n,v);

  printf("Punti fatti: ");
  for (i = 0; i < n; i++)
    printf("%d ", v[i]);
  printf("\n");
  return 0;
}

void CalcolaPunti(int m[][DIM], int n, int v[])
{
  int i, j;
  for (i = 0; i < n; i++)
    v[i] = 0;

  for (i = 0; i < n; i++)
    for (j = i+1; j < n; j++)
      {
        v[i] += Punti(m[i][j], m[j][i]);
        v[j] += Punti(m[j][i], m[i][j]);
      }
}

int Punti(int set1, int set2)
{
  if (set1 == 3 && set2 <= 1)
    return 3;
  else if (set1 == 3 && set2 == 2)
    return 2;
  else if (set1 == 2 && set2 == 3)
    return 1;
  else /* caso (set1 <= 1 && set2 == 3) */
    return 0;
}
```

Soluzione compito del 2 luglio 2012

Soluzione esercizio 1

Per la soluzione si utilizza un vettore che memorizza il livello di riempimento corrente di ciascun silo. Per poter dimensionare il vettore, si assume che il numero di silos sia al massimo `MAX_SILOS` (pari a 50).

```
#include <stdio.h>
#include <stdlib.h>
#define MAX_SILOS 50

int RiempimentoSilos(char nome_file[], int num_silos, float dim_silos);
int SilosDisponibile(float c, float v[], int n, float d);

int main(int argc, char* argv[])
{/* non richiesto per il compito in classe */
  int num, cisterne;
  float dim;

  if (argc == 2)
    {
      printf("Numero di silos: ");
      scanf("%d", &num);
      printf("Dimensione dei silos: ");
      scanf("%f", &dim);
    }
  else if (argc == 4)
    {
      num = atoi(argv[2]);
      dim = atof(argv[3]);
    }
  else
    {
      printf("Numero di parametri sbagliato per %s\n", argv[0]);
      exit(1);
    }

  cisterne = RiempimentoSilos(argv[1], num, dim);
  printf("Si riescono a scaricare %d autocisterne\n", cisterne);
  return 0;
}

int RiempimentoSilos(char nome_file[], int num_silos, float dim_silos)
{
  float carico, riempimento[MAX_SILOS] = {0.0};
  int i, num_cisterne = 0;
  FILE* fp;

  fp = fopen(nome_file, "r");

  while (fscanf(fp, "%*s%f", &carico) != EOF)
    {
      i = SilosDisponibile(carico, riempimento, num_silos, dim_silos);
      if (i != -1)
        {
          num_cisterne++;
          riempimento[i] += carico;
        }
   }
  return num_cisterne;
```

```
}

int SilosDisponibile(float c, float v[], int n, float d)
{/* restituisce l'indice del primo silos disponibile, -1 se non ce ne sono */
  int i;
  for (i = 0; i < n; i++)
    if (c + v[i] <= d)
      return i;
  return -1;
}
```

Soluzione esercizio 2

Utilizziamo una funzione ausiliaria che esplora i punti adiacenti di ciascun punto e restituisce la massima pendenza.

Per il *driver* si è scelto di popolare la matrice di numeri casuali tra 0 e 100. La dimensione della matrice viene passata da riga di comando.

Si noti che utilizzando `cygwin`, lo spazio assegnato dal sistema operativo al programma non consente di allocare una matrice di `float` dimensione 1000×1000. È quindi necessario ridurre la dimensione massima, rispetto a quella considerata nel testo (ad esempio a 500).

```
#include <stdio.h>
#include <stdlib.h>
#include <time.h>
#define MAX_DIM 100

struct Punto
{
  int riga;
  int colonna;
  float dislivello;
};

float Random(float min, float max);
struct Punto MassimaPendenza(float m[][MAX_DIM], int n);
float PendenzaMassima(float m[][MAX_DIM], int n, int r, int c);

int main(int argc, char* argv[])
{/* non richiesto per il compito in classe */
  int i,j,n;
  float mat[MAX_DIM][MAX_DIM];
  struct Punto p;

  if (argc != 2)
    {
      printf("Numero di parametri sbagliato per %s\n", argv[0]);
      exit(1);
    }

  n = atoi(argv[1]);

  /* la matrice viene riempita con valori casuali tra 0 e 100 */
  srand(time(NULL));
  for (i = 0; i < n; i++)
    for (j = 0; j < n; j++)
      mat[i][j] = Random(0.0, 100.0);
```

```
  p = MassimaPendenza(mat,n);

  for (i = 0; i < n; i++)
    {
      for (j = 0; j < n; j++)
        printf("%.2f ", mat[i][j]);
      printf("\n");
    }
  printf("Il punto di massimo dislivello e' (%d,%d), con dislivello %.2f\n",
         p.riga, p.colonna, p.dislivello);
  return 0;
}

struct Punto MassimaPendenza(float m[][MAX_DIM], int n)
{
  struct Punto ris = {0, 0, 0.0};
  int i, j;
  float pendenza;

  for (i = 0; i < n; i++)
    for (j = 0; j < n; j++)
      {
        pendenza = PendenzaMassima(m,n,i,j);
        if (pendenza > ris.dislivello)
          {
            ris.dislivello = pendenza;
            ris.riga = i;
            ris.colonna = j;
          }
      }
  return ris;
}

float PendenzaMassima(float m[][MAX_DIM], int n, int r, int c)
{
  float max = 0.0, dislivello;
  int i, j;

  for (i = r-1; i <= r+1; i++)
    for (j = c-1; j <= c+1; j++)
      if ((i != r || j != c) && i >= 0 && j >= 0 && i < n && j < n)
        {
          dislivello = m[r][c] - m[i][j];
          if (dislivello > max)
            max = dislivello;
        }
  return max;
}

float Random(float min, float max)
{ /* utilizzata dal driver */
  return min + (float)rand()/((float)RAND_MAX/(max-min));
}
```

Soluzione compito del 26 febbraio 2013

Soluzione esercizio 1

L'esercizio non presenta particolari difficoltà. Il file può essere letto tramite un'unica `fscanf` in un ciclo `while`, anche se con una stringa di formato molto lunga.

Si noti che nel *driver*, la data viene passata per comodità sulla riga di comando come stringa, e successivamente suddivisa nelle tre parti tramite la funzione `sscanf` (non in programma d'esame).

```
#include <stdio.h>
#include <stdlib.h>
#include "../Utils/util_date.h"

struct Parametri
{
  int A;
  int B;
};

struct Parametri TrovaMassimo(char nome_file[], struct Data d)
{
  FILE* fp;
  fp = fopen(nome_file, "r");
  struct Data d1;
  struct Parametri p, p_max = {0,0};
  float ris, max_ris;

  while(fscanf(fp,"%*1s%d%*c%d%*c%d%*c%*c%*c%d%*c%*c%*c%d%*c%*c%*c%g%*c",
               &d1.giorno,&d1.mese,&d1.anno,&p.A,&p.B,&ris) != EOF)
    {
      if (ComparaDate(d1,d) == 1)
        {
          if ((p.A == 0 && p.B == 0) || ris > max_ris)
            {
              max_ris = ris;
              p_max = p;
            }
        }
    }
  fclose(fp);
  return  p_max;
}

int main(int argc, char* argv[])
{/* non richiesto per il compito in classe */
  struct Data d;
  struct Parametri p;

  if (argc != 3)
    {
      printf("Utilizzo: %s <nome_file> <gg/mm/aaa>\n", argv[0]);
      exit(1);
    }
  sscanf(argv[2], "%d%*c%d%*c%d", &d.giorno,&d.mese,&d.anno);
  p = TrovaMassimo(argv[1], d);
  printf("La configurazione che restituisce il massimo e' (%d,%d)\n", p.A, p.B);
  return 0;
}
```

Soluzione esercizio 2

La funzione principale dell'esercizio `CercaAltopiano` si avvale di una funzione ausiliaria `Altopiano` che verifica se una zona di terreno rappresenta un altopiano.

Il *driver* invece di limitarsi ad invocare la funzione dell'esercizio, è congegnato per generare una matrice casuale e cercare in essa la dimensione dell'altopiano più grande.

```
#include <stdio.h>
#include <stdlib.h>
#include <time.h>
#define MAX_DIM 500
#define DIFF 10

int CercaAltopiano(int m[][MAX_DIM], int n, int k);
int Altopiano(int m[][MAX_DIM], int i, int j, int k);
int Random(int min, int max);

int main(int argc, char* argv[])
{/* non richiesto per il compito in classe */
 /* crea una matrice con valori casuali e cerca l'altopiano
    piu' grande nel terreno */
  int mat[MAX_DIM][MAX_DIM];
  int i,j,n,k;

  srand(time(NULL));
  n = Random(10,20);
  for (i = 0; i < n; i++)
    for (j = 0; j < n; j++)
      mat[i][j] = Random(200, 230);

  for (i = 0; i < n; i++)
    {
      for (j = 0; j < n; j++)
        printf("%d ", mat[i][j]);
      printf("\n");
    }

  for (k = n; k >= 1; k--)
    if (CercaAltopiano(mat,n,k))
      {
        printf("Esiste un altopiano di dimensione %d\n", k);
        break;
      }
  return 0;
}

int CercaAltopiano(int m[][MAX_DIM], int n, int k)
{
  int i, j;
  for (i = 0; i <= n - k; i++)
    for (j = 0; j <= n - k; j++)
      if (Altopiano(m,i,j,k))
        return 1;
  return 0;
}

int Altopiano(int m[][MAX_DIM], int i, int j, int k)
{
  int r, c;
  int min = m[i][j], max = m[i][j];
```

```
  for (r = i; r < i + k; r++)
    for (c = j; c < j + k; c++)
      if (m[r][c] < min)
        min = m[r][c];
      else if (m[r][c] > max)
        max = m[r][c];
  return max - min <= DIFF;
}

int Random(int min, int max)
{ /* utilizzata dal driver (main) */
  return min + rand() % (max-min+1);
}
```