I/O avanzato e File

Esercizi risolti

1 Esercizio: "Minuti lavorati"

Un'azienda ha dotato i propri dipendenti di un sensore wireless che emette un codice numerico ogni volta che un dipendente attraversa la porta d'ingresso/uscita dell'azienda o ne transita nelle vicinanze. L'azienda ha meno di 1000 dipendenti. Ad ogni attraversamento, il sensore registra ora e minuti del passaggio, insieme al codice del dipendente (un codice alfanumerico di max 10 caratteri).

Si desidera sviluppare un programma in linguaggio C per il calcolo delle ore lavorative dei dipendenti dell'azienda. Il programma riceve sulla linea di comando un primo parametro, che rappresenta il nome del file contenente gli attraversamenti, ed un secondo parametro (opzionale), che rappresenta il codice numerico di un dipendente.

Il file è relativo ai passaggi di una sola giornata, ed è composto da una serie di righe, ciascuna delle quali corrisponde ad un passaggio, ed è composta da tre campi:

```
ora minuti codice_dipendente
```

Se il programma viene invocato con un due parametri sulla linea di comando (vi è il codice del dipendente), allora dovrà stampare, per il dipendente specificato, il numero totale di minuti lavorati. Per determinare il numero di minuti lavorati occorre confrontare l'orario del *primo* passaggio con l'orario dell'*ultimo* passaggio per quel dipendente.

Se invece il programma viene invocato con un solo parametro sulla linea di comando (il codice del dipendente è assente), allora il programma dovrà stampare il numero totale di dipendenti *diversi* che hanno lavorato in quel giorno (ossia che sono passati almeno una volta dalla porta).

Ad esempio, dato il seguente file di testo `passaggi.txt`:

```
8 30 abc222
8 30 abc123
8 31 azx112
9 10 abc123
12 10 abc123
```

il programma (chiamato `orario.c`) si dovrà comportare nel modo seguente:

```
c:> orario passaggi.txt
Ci sono 3 dipendenti diversi.

c:> orario passaggi.txt abc123
Il dipendente abc123 ha lavorato per 220 minuti.
```

Soluzione

```
1  /* PROGRAMMAZIONE IN C */
2
3  /* File: orario.c */
4  /* Soluzione proposta esercizio "Minuti lavorati" */
5
6  #include <stdio.h>
7  #include <stdlib.h>
8  #include <string.h>
9
10 int main( int argc, char *argv[] )
11 {
```

```
12      const int MAX = 100 ;
13      const int NUMDIP = 1000 ;
14      const int LUNMAT = 10 ;
15      FILE * f ;
16
17      int min, max, tempo, passaggi, r ;
18      int ore, minuti ;
19      char riga[MAX+1] ;
20      char matricola[LUNMAT+1];
21
22      char nomi[NUMDIP][LUNMAT+1] ;
23      int N, i, presente ;
24
25      /* controllo parametri:
26          argv[1] -> nome del file
27          argv[2] -> matricola dipendente (opzionale)
28      */
29      if ( argc != 2 && argc != 3 )
30      {
31          printf("ERRORE:␣numero␣di␣parametri␣errato\n") ;
32          exit(1) ;
33      }
34
35      /* apertura del file */
36      f = fopen(argv[1], "r") ;
37      if ( f==NULL )
38      {
39          printf("ERRORE:␣impossibile␣aprire␣il␣file␣%s\n", argv[1]) ;
40          exit(1) ;
41      }
42
43      if ( argc == 2 )
44      {
45          /* CALCOLO DEL NUMERO DI DIPENDENTI DIVERSI */
46          N = 0 ;
47
48          while ( fgets( riga,MAX, f) != NULL )
49          {
50              r = sscanf( riga, "%*d␣%*d␣%s", matricola ) ;
51              /* NOTA: gli asterischi nella stringa di formato della sscanf
52              (come %*d) servono per far leggere il dato corrispondente
53              ma non memorizzarlo in alcuna variabile.
54              In effetti qui i primi due campi numerici non ci servono */
55
56              if ( r != 1 )
57              {
58                  printf("Riga␣in␣formato␣errato␣-␣ignorata\n") ;
59              }
60              else
61              {
62                  /* Cerca se 'matricola' è già presente */
63                  presente = 0 ;
64                  for (i=0; i<N && presente==0; i++)
65                      if (strcmp(matricola, nomi[i])==0)
66                          presente=1;
67
```

```
68                  /* Se è nuovo, aggiungilo */
69                  if ( presente==0 )
70                  {
71                      strcpy( nomi[N], matricola ) ;
72                      N++ ;
73                  }
74              }
75          }
76          fclose(f) ;
77
78          printf("Ci␣sono␣%d␣dipendenti␣diversi\n", N) ;
79      }
80      else
81      {
82          /* CALCOLO DEL TEMPO LAVORATO DAL DIPENDENTE LA CUI
83          MATRICOLA È argv[2] */
84          max = 0 ;
85          min = 24*60 ;
86
87          passaggi = 0 ;
88
89          while ( fgets( riga, MAX, f ) != NULL )
90          {
91              r = sscanf( riga, "%d␣%d␣%s", &ore, &minuti, matricola ) ;
92              if ( r != 3 )
93              {
94                  printf("Riga␣in␣formato␣errato␣-␣ignorata\n") ;
95              }
96              else
97              {
98                  tempo = ore * 60 + minuti ;
99
100                 if ( strcmp( matricola, argv[2] ) == 0 )
101                 {
102                     if ( tempo<min )
103                         min = tempo ;
104                     if ( tempo>max )
105                         max = tempo ;
106                     passaggi ++ ;
107                 }
108             }
109         }
110         fclose(f) ;
111
112         if ( passaggi>=2 )
113             printf("Il␣dipendente␣di␣matricola␣%s␣ha␣lavorato␣per␣%d␣minuti\n",
114                    argv[2], max-min ) ;
115         else
116             printf("ERRORE:␣Il␣dipendente␣%s␣ha␣fatto␣solo␣%d␣passaggi\n",
117                    argv[2], passaggi) ;
118     }
119
120     exit(0);
121 }
```

2 Esercizio: "Cartoline"

Realizzare un programma in linguaggio C per registrare le cartoline scambiate tra un gruppo di amici (massimo 20 amici).

L'elenco delle cartoline è memorizzato in un file di testo, composto da un numero imprecisato di linee, ciascuna delle quali contiene tre elementi: il nome del mittente, il nome del destinatario ed il nome della località da cui la cartolina è stata inviata. I nomi degli amici e delle località sono da intendersi privi di spazi e lunghi al massimo 30 caratteri ciascuno.

Il programma riceve come primo parametro sulla linea di comando il nome del file di testo, mentre il secondo parametro può essere la stringa `new` oppure la stringa `find`.

Il comando `new` richiede ulteriori tre parametri sulla linea di comando, corrispondenti ai nomi degli amici e della località, e deve aggiungere tali informazioni in coda al file. Il programma deve segnalare con un messaggio errore l'eventuale tentativo di re-introdurre una cartolina identica ad una già esistente.

Il comando `find` è invece seguito da un solo ulteriore parametro sulla linea di comando, corrispondente al nome di un amico. In questo caso il programma deve stampare l'elenco degli amici che hanno spedito cartoline all'amico specificato e le località corrispondenti.

Esempio
Supponiamo che il programma si chiami `cartoline` e che il file `car.txt` contenga i seguenti dati:

```
Gino Toni Rimini
Gino Luigi Rimini
Toni Gino Maldive
Luigi Toni Moncalieri
```

In tal caso attivando il programma nel seguente modo:

```
cartoline car.txt find Toni
```

dovrà essere generato il seguente output:

```
Cartoline ricevute da Toni:
  Gino da Rimini
  Luigi da Moncalieri
```

Invece, attivando il programma col seguente comando:

```
cartoline car.txt new Toni Luigi Roma
```

dovrà essere aggiunta in coda al file `car.txt` la seguente riga:

```
Toni Luigi Roma
```

Soluzione

```
1  /* PROGRAMMAZIONE IN C */
2
3  /* File: cartoline.c */
4  /* Soluzione proposta esercizio "Cartoline" */
5
6  #include <stdio.h>
7  #include <stdlib.h>
8  #include <string.h>
9
10 int main( int argc, char *argv[] )
11 {
```

```
12      const int MAX = 100 ;
13      const int LUN = 30 ;
14
15      FILE *f ;
16      char riga[MAX+1] ;
17      char mitt[LUN+1], dest[LUN+1], luogo[LUN+1] ;
18      int r, esiste ;
19
20      /* Controlla i parametri sulla linea di comando */
21      if ( argc==4 && strcmp(argv[2], "find")==0 )
22      {
23          /* comando 'find' */
24          /* cerca all'interno del file se esiste un amico 'destinatario'
25          uguale ad argv[3] */
26          f = fopen( argv[1], "r" ) ;
27          if ( f==NULL )
28          {
29              printf("ERRORE:␣impossibile␣aprire␣file␣%s\n", argv[1]) ;
30              exit(1) ;
31          }
32
33          printf("Cartoline␣ricevute␣da␣%s:\n", argv[3]) ;
34
35          while ( fgets( riga, MAX, f ) != NULL )
36          {
37              r = sscanf( riga, "%s␣%s␣%s", mitt, dest, luogo ) ;
38
39              if ( r==3 )
40              {
41                  /* controlla se l'amico è quello giusto */
42                  if ( strcmp(dest, argv[3])==0 )
43                  {
44                      printf("␣␣␣%s␣da␣%s\n", mitt, luogo) ;
45                  }
46              }
47              else
48                  printf("Riga␣in␣formato␣errato␣-␣ignorata\n") ;
49          }
50
51          fclose(f) ;
52
53      }
54      else if ( argc==6 && strcmp(argv[2], "new")==0 )
55      {
56          /* comando 'new' */
57
58          /* controlla se esiste già una cartolina con
59              mittente == argv[3]
60              destinatario == argv[4]
61              luogo == argv[5]
62          */
63          esiste = 0 ;
64
65          f = fopen( argv[1], "r" ) ;
66          if ( f==NULL )
67          {
```

```
68              printf("ERRORE:␣impossibile␣aprire␣file␣%s\n", argv[1]) ;
69              exit(1) ;
70          }
71
72          while ( fgets( riga, MAX, f ) != NULL )
73          {
74              r = sscanf( riga, "%s␣%s␣%s", mitt, dest, luogo ) ;
75
76              if ( r==3 )
77              {
78                  /* controlla se la cartolina è duplicata */
79                  if ( strcmp(mitt, argv[3])==0 &&
80                          strcmp(dest, argv[4])==0 &&
81                          strcmp(luogo, argv[5])==0   )
82                  {
83                      esiste = 1;
84                      printf("Attenzione:␣cartolina␣già␣esistente\n") ;
85                  }
86              }
87              else
88                  printf("Riga␣in␣formato␣errato␣-␣ignorata\n") ;
89          }
90
91          fclose(f) ;
92
93          /* se non esiste ancora, aggiunge una nuova riga al file */
94          if ( esiste==0 )
95          {
96              /* aggiungi una riga */
97              f = fopen( argv[1], "a" ) ;
98              if ( f==NULL )
99              {
100                 printf("ERRORE:␣impossibile␣modificare␣il␣file␣%s\n", argv[1]) ;
101                 exit(1) ;
102             }
103
104             fprintf( f, "%s␣%s␣%s\n", argv[3], argv[4], argv[5] ) ;
105
106             fclose(f) ;
107         }
108     }
109     else
110     {
111         printf("ERRORE:␣Numero␣di␣parametri␣errato␣o␣comando␣sconosciuto\n") ;
112         printf("Utilizzo:␣%s␣nomefile␣find␣nomeamico\n", argv[0]) ;
113         printf("oppure␣␣:␣%s␣nomefile␣new␣amicomittente␣amicodestinatario␣luogo\n",\
114                 argv[0]) ;
115         exit(1) ;
116     }
117
118     exit(0) ;
119 }
```

3 Esercizio: “Registro d’esame”

Si desidera sviluppare un programma in linguaggio C per gestire in modo informatico un registro di esame.

Il registro è memorizzato in un file di testo con nome `registro.txt` e contiene i dati di N studenti, ove N è il numero intero scritto nella prima riga del file. Dopo questa prima riga, ogni riga successiva contiene il dato relativo ad un singolo studente, indicando il numero di matricola dello studente (numero intero compreso 1 e 999999) ed il voto conseguito (numero intero con valore tra 18 e 30, oppure zero per indicare che l'esame non è ancora stato sostenuto).

Il programma può essere attivato in due modi diversi.

Se viene attivato passando come primo parametro sulla linea di comando la parola `stat` allora deve fornire le seguenti statistiche: numero di studenti promossi (e relativa percentuale sul totale, espressa con una cifra dopo la virgola) e voto medio degli studenti promossi (indicato con una sola cifra dopo la virgola).

Il programma può anche essere attivato passando come primo parametro la parola `voto`, come secondo parametro il numero di matricola di uno studente e come ultimo parametro il voto conseguito dallo studente. In questo caso il programma deve inserire nel file il voto dello studente, segnalando però errore nel caso che lo studente non sia iscritto all'esame (ossia il suo numero di matricola non compaia nel file) oppure abbia già superato l'esame (ossia voto diverso da zero nella riga contenente la sua matricola).

Ad esempio se il file `registro.txt` contenesse i seguenti dati:

```
3
33467 30
24356 0
224678 18
```

ed il programma – supposto chiamarsi `esame` – venisse attivato con la seguente riga di comando:

```
esame stat
```

allora il programma dovrebbe produrre il seguente output:

```
promossi = 2 (66.7 %)
voto medio = 24.0
```

Se invece il programma venisse attivato nel seguente modo:

```
esame voto 24356 24
```

allora dopo l'esecuzione del programma il file `registro.txt` dovrebbe contenere i seguenti dati:

```
3
33467 30
24356 24
224678 18
```

Soluzione

```
1  /* PROGRAMMAZIONE IN C */
2
3  /* File: esame.c */
4  /* Soluzione proposta esercizio "Registro d'esame" */
5
6  #include <stdio.h>
7  #include <stdlib.h>
8  #include <string.h>
9
10 int main(int argc, char *argv[])
11 {
12     const int MAX = 1000 ;
13     const int LUN = 80 ;
```

```
14      const char nomefile[] = "registro.txt" ;
15
16      int matricola[MAX] ;
17      int voto[MAX] ;
18      int N ;
19
20      FILE * f  ;
21      char riga[LUN+1] ;
22      char comando[LUN+1] ;
23      int r, r1, r2, mat, vot, c, somma, i, trovato, pos ;
24
25      /* 1) Leggi il contenuto del file all'interno dei vettori */
26      f = fopen(nomefile, "r") ;
27      if ( f==NULL )
28      {
29          printf("ERRORE: impossibile aprire il file %s\n", nomefile) ;
30          exit(1) ;
31      }
32
33      fgets( riga, LUN, f ) ;
34      r = sscanf(riga, "%d", &N) ;
35      if ( r != 1 )
36      {
37          printf("ERRORE: formato errato nella prima riga\n") ;
38          exit(1) ;
39      }
40
41      c = 0 ;
42      while ( fgets( riga, LUN, f ) != NULL )
43      {
44          r = sscanf( riga, "%d %d", &mat,&vot ) ;
45
46          if ( r != 2 || mat<1 || mat>999999 ||
47                  !( vot==0 || ( vot>=18 && vot<=30) ) )
48          {
49              printf("ATTENZIONE: riga in formato errato - ignorata\n") ;
50          }
51          else
52          {
53              /* aggiungi ai vettori */
54              matricola[c] = mat ;
55              voto[c] = vot ;
56              c++ ;
57          }
58      }
59
60      fclose(f) ;
61
62      if ( c != N )
63      {
64          printf("ERRORE di coerenza nel file\n") ;
65          exit(1) ;
66      }
67
68      /* 2) Acquisisci il comando dell'utente */
69
```

```
70      if ( !(
71                  (argc==2 && strcmp(argv[1], "stat")==0) ||
72                  (argc==4 && strcmp(argv[1], "voto")==0)
73              ) )
74      {
75          printf("ERRORE: numero argomenti errato\n");
76          exit(1) ;
77      }
78
79      strcpy( comando, argv[1] ) ;
80      if ( strcmp( comando, "voto" )==0 )
81      {
82          r1 = sscanf(argv[2], "%d", &mat ) ;
83          r2 = sscanf(argv[3], "%d", &vot ) ;
84          if ( r1 != 1 || r2 != 1 )
85          {
86              printf("ERRORE: matricola e voto devono essere numerici\n") ;
87              exit(1) ;
88          }
89      }
90
91      /* 2a) "stat" */
92      if ( strcmp(comando, "stat")==0 )
93      {
94          /* 2a1) calcola e stampa le statistiche */
95          c = 0 ;
96          somma = 0 ;
97          for ( i = 0 ; i < N ; i++ )
98          {
99              if ( voto[i]!=0 )
100             {
101                 c++ ;
102                 somma = somma + voto[i] ;
103             }
104         }
105
106         printf("promossi = %d (%f %%)\n", c, (double)c/(double)N*100.0 ) ;
107         printf("voto medio = %f\n", (double)somma/(double)c ) ;
108
109     }
110     else if ( strcmp(comando, "voto")==0 )
111     {
112         /* 2b) "voto nmatricola valorevoto" */
113
114         /* ricerca 'mat' all'interno del vettore matricola[] */
115         /* output: trovato=1/0, pos */
116         trovato = 0 ;
117         for (i=0; i<N && trovato==0; i++)
118         {
119             if ( matricola[i] == mat )
120             {
121                 trovato = 1 ;
122                 pos = i ;
123             }
124         }
125
```

```
126         /* controlla se e' valido */
127         if ( trovato == 1 && voto[pos]==0 )
128         {
129             /* modifica il voto all'interno del vettore */
130             voto[pos] = vot ;
131
132             /* salva i vettori su file */
133             f = fopen( nomefile, "w" ) ;
134             if ( f==NULL )
135             {
136                 printf("ERRORE: impossibile scrivere il file modificato\n") ;
137                 exit(1) ;
138             }
139
140             /* la prima riga contiene il numero di studenti */
141             fprintf(f, "%d\n", N) ;
142
143             for (i=0; i<N; i++)
144                 fprintf(f, "%d %d\n", matricola[i], voto[i]) ;
145
146             fclose(f) ;
147         }
148         else
149         {
150             printf("Impossibile registrare il voto\n") ;
151             if (trovato==0)
152                 printf("Lo studente non esiste\n") ;
153             else
154                 printf("L'esame e' gia' stato superato\n" ) ;
155         }
156     }
157     else
158     {
159         printf("ERRORE: comando non valido\n") ;
160         exit(1) ;
161     }
162
163     exit(1) ;
164 }
```

4 Esercizio: "Sostituzione lettere"

Si desidera sviluppare un programma in linguaggio C per la modifica di un file di testo. La modifica consiste nel sostituire – scambiandoli tra loro – due caratteri alfabetici dati. In particolare, tutte le occorrenze del primo carattere dovranno essere sostituite dal secondo e viceversa. La sostituzione deve avvenire mantenendo la forma (maiuscola o minuscola) della lettera originaria.

Il programma riceve sulla linea di comando tre parametri: il nome del file di testo da elaborare, il nome di un secondo file di testo nel quale salvare il risultato ed una stringa di 2 caratteri che specifica i caratteri da scambiare.

Il file di testo è composto da un numero imprecisato di linee.

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

```
scambia TESTO.TXT MODIF.TXT ae
```

ed il file `TESTO.TXT` contenesse i seguenti dati:

```
QUEL RAMO del lago di Como, che volge a mezzogiorno,
tra due CATENE non interrotte di MONTI, tutto a seni e a golfi,
a seconda dello sporgere E DEL RIENTRARE di quelli, vien, quasi
```

allora il programma dovrebbe produrre il seguente file `MODIF.TXT` perché dovrebbe sostituire tutte le lettere A (a) con E (e) e tutte le lettere E (e) con A (a):

```
QUAL REMO dal lego di Como, cha volga e mazzogiorno,
tre dua CETANA non intarrotta di MONTI, tutto e sani a e golfi,
e saconde dallo sporgara A DAL RIANTRERE di qualli, vian, quesi
```

Soluzione

```c
/* PROGRAMMAZIONE IN C */

/* File: scambia.c */
/* Soluzione proposta esercizio "Sostituzione lettere" */

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

int main( int argc, char *argv[] )
{
    char let1, let2 ;
    FILE * f ;
    FILE * g ;

    int ch ;

    /* controlla la correttezza degli argomenti */

    if ( argc!=4 )
    {
        printf("ERRORE: numero di argomenti errato\n") ;
        printf("Utilizzo: %s file1 file2 ab\n", argv[0]) ;
        exit(1) ;
    }

    if ( strlen(argv[3])!=2 || !isalpha(argv[3][0]) || !isalpha(argv[3][1]) )
    {
        printf("ERRORE: parametro %s non valido\n", argv[3]) ;
        printf("Deve essere composto di 2 caratteri alfabetici\n") ;
        exit(1) ;
    }

    let1 = tolower(argv[3][0]) ;
    let2 = tolower(argv[3][1]) ;

    /* travasa il file argv[1] in argv[2] */
    f = fopen( argv[1], "r") ;
    g = fopen( argv[2], "w") ;

    if ( f==NULL || g==NULL )
    {
```

```
44          printf("ERRORE: impossibile aprire i file\n") ;
45          exit(1) ;
46      }
47
48      while ( ( ch = fgetc(f) ) != EOF )
49      {
50          /* controlla ch ed eventualmente modificalo */
51          if ( tolower(ch) == let1 )
52          {
53              if ( isupper(ch) )
54                  ch = toupper(let2) ;
55              else
56                  ch = let2 ;
57          }
58          else if ( tolower(ch) == let2 )
59          {
60              if ( isupper(ch) )
61                  ch = toupper(let1) ;
62              else
63                  ch = let1 ;
64          }
65
66          fputc( ch, g ) ;
67      }
68
69      fclose(f) ;
70      fclose(g) ;
71
72      exit(0) ;
73  }
```

5 Esercizio: “Superfici e Volumi”

Si desidera sviluppare un programma in linguaggio C per il calcolo delle superfici e dei volumi di un edificio.

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

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

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

- la superficie totale di tutte le stanze dell’edificio, espressa in metri quadri
- il volume totale di tutte le stanze dell’edificio, espresso in metri cubi.

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

```
dimef CASA.TXT 2
```

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

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

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

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

Soluzione

```c
/* PROGRAMMAZIONE IN C */

/* File: dimef.c */
/* Soluzione proposta esercizio "Superfici e Volumi" */

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

int main( int argc, char *argv[] )
{
    const int MAX = 100 ;

    int nPiani ;
    double areaTot ; /* superficie totale in m^2 */
    double volTot ;  /* volume totale in m^3 */
    double areaPiano ; /* superficie di 1 piano in m^2 */

    int p, s, x, y, r ;
    int nStanze, hPiano ;

    FILE * f ;
    char riga[MAX+1] ;

    if ( argc!=3 )
    {
        printf("ERRORE: numero argomenti errato\n") ;
        exit(1) ;
    }

    /* argv[1] -> nome del file */
    /* argv[2] -> numero di piani */

    r = sscanf( argv[2], "%d", &nPiani ) ;
    if ( r!=1 || nPiani<1 )
    {
        printf("ERRORE: numero piani errato\n") ;
        exit(1) ;
    }

    f = fopen( argv[1], "r" ) ;
    if ( f==NULL )
```

```
42      {
43          printf("ERRORE:␣impossibile␣aprire␣file␣%s\n", argv[1]) ;
44          exit(1) ;
45      }
46
47      areaTot = 0.0 ;
48      volTot = 0.0 ;
49
50      /* per ogni piano p=1...nPiani */
51      for ( p = 1 ; p <= nPiani ; p++ )
52      {
53          /* leggere nStanze e altezza hPiano */
54          if ( fgets( riga, MAX, f ) == NULL )
55          {
56              printf("ERRORE:␣il␣file␣e'␣finito␣troppo␣presto\n") ;
57              exit(1) ;
58          }
59          if ( sscanf( riga, "%d␣%d", &nStanze, &hPiano ) != 2 )
60          {
61              printf("ERRORE:␣riga␣in␣formato␣errato\n") ;
62              exit(1) ;
63          }
64
65          /* opzionale: controllare che nStanze>=1 e 1<=hPiano<=h_max */
66
67          areaPiano = 0.0 ;
68          /* per ogni stanza del piano, da 0 a nStanze-1 */
69          for ( s = 0 ; s < nStanze; s++ )
70          {
71              /* leggi le misure */
72              if ( fgets( riga, MAX, f ) == NULL )
73              {
74                  printf("ERRORE:␣il␣file␣e'␣finito␣troppo␣presto\n") ;
75                  exit(1) ;
76              }
77              if ( sscanf( riga, "%d␣%d", &x, &y ) != 2 )
78              {
79                  printf("ERRORE:␣riga␣in␣formato␣errato\n") ;
80                  exit(1) ;
81              }
82
83              /* aggiorna areaPiano */
84              areaPiano = areaPiano + (x * y)/10000.0 ;
85          }
86
87          areaTot = areaTot + areaPiano ;
88          volTot = volTot + areaPiano * (hPiano/100.0) ;
89      }
90
91      fclose(f) ;
92
93      printf("Superficie␣totale␣dell'edificio:␣%.2f␣metri␣quadri\n", areaTot) ;
94
95      printf("Volume␣totale␣dell'edificio:␣%.2f␣metri␣cubi\n", volTot ) ;
96
97      exit(0) ;
```

```
98 }
```

6 Esercizio: “Statistiche caratteri”

Si desidera sviluppare un programma in linguaggio C per il calcolo di statistiche sui caratteri presenti in un file di testo il cui nome è specificato come primo parametro sulla riga di comando.

Il programma deve considerare tutti i caratteri tranne quelli di spaziatura e fornire in output:

- il numero di righe di cui è composto il testo
- il numero totale di caratteri (esclusi quelli di spaziatura) presenti nel testo
- il numero massimo e medio di caratteri di una riga
- la riga più lunga incontrata nel file.

Ad esempio, se al programma fosse fornito un file col seguente testo:

```
La Vispa Teresa
tra l'erbetta
rincorrea
la farfalletta.
```

allora dovrebbe produrre il seguente output:

```
numero di righe: 4
numero di caratteri: 48
numero di caratteri per riga:
- medio 12.0
- massimo 14
riga più lunga:
  La Vispa Teresa
```

Soluzione

```
1  /* PROGRAMMAZIONE IN C */
2
3  /* File: statcar.c */
4  /* Soluzione proposta esercizio "Statistiche caratteri" */
5
6  #include <stdio.h>
7  #include <stdlib.h>
8  #include <string.h>
9
10 int main( int argc, char *argv[] )
11 {
12     const int MAX = 200 ;
13
14     char riga[MAX+1] ;
15     char lunga[MAX+1] ;
16
17     FILE *f ;
18
19     int nRighe ;
20     int nCarTot ;
21     int nCarMax ;
```

```
22
23      int nCarRiga ;
24      int i ;
25
26      if ( argc!=2 )
27      {
28          printf("ERRORE: numero di parametri errato\n") ;
29          exit(1) ;
30      }
31
32      f = fopen( argv[1], "r" ) ;
33      if ( f == NULL )
34      {
35          printf("ERRORE: impossibile aprire file %s\n", argv[1]) ;
36          exit(1) ;
37      }
38
39      nRighe = 0 ;
40      nCarTot = 0 ;
41      nCarMax = -1 ;
42
43      while ( fgets(riga, MAX, f) != NULL )
44      {
45          /* conta il numero di car. diversi da spazio nella riga corrente */
46          nCarRiga = 0 ;
47          for ( i=0; riga[i]!='\n'; i++ )
48          {
49              if ( riga[i] != ' ' )
50                  nCarRiga ++ ;
51          }
52
53          nRighe ++ ;
54          nCarTot = nCarTot + nCarRiga ;
55
56          if ( nCarRiga > nCarMax )
57          {
58              nCarMax = nCarRiga ;
59              strcpy( lunga, riga ) ;
60          }
61      }
62
63      printf("numero di righe: %d\n", nRighe) ;
64      printf("numero di caratteri: %d\n", nCarTot) ;
65      printf("numero di caratteri per riga:\n") ;
66      printf(" - medio %.1f\n", (double)nCarTot/(double)nRighe ) ;
67      printf(" - massimo %d\n", nCarMax ) ;
68      printf("riga piu' lunga:\n") ;
69      printf("%s", lunga ) ;
70
71      exit(0) ;
72  }
```

7 Esercizio: “Temperature”

Si desidera sviluppare un programma in linguaggio C per il calcolo di statistiche sulle temperature registrate in varie città italiane.

Il programma riceve in un file di testo (il cui nome è specificato come primo parametro sulla riga di comando) le informazioni sulle temperature. Ogni riga del file ha il seguente formato:

```
temperatura  luogo
```

dove:

- `temperatura` è un numero in formato floating-point che esprime la temperatura rilevata;
- `luogo` è il nome del luogo ove la temperatura è stata rilevata (stringa di caratteri priva di spazi composta al massimo da 31 caratteri).

Eventuali righe con formato errato devono essere scartate segnalando l'errore (es. *riga n. X errata - ignorata*).

Il programma riceve come secondo parametro sulla riga di comando il nome di una località per la quale deve calcolare il valore medio della temperatura.

Infine se è presente un terzo parametro sulla riga di comando (opzionale) allora esso indica una soglia di temperatura per la quale si chiede che il programma indichi il numero di giorni in cui tale soglia è stata superata.

Ad esempio, supponiamo che il file `tluoghi.txt` contenga i seguenti dati:

```
24.0 Torino
26.0 Milano
27.2 Milano
26.0 Torino
28.0 Torino
29.4 Milano
```

Se il programma – denominato `temperatura` – viene attivato con la seguente riga di comando:

```
temperatura tluoghi.txt Torino
```

allora deve produrre il seguente output:

```
Torino:
- temperatura media 26.0
```

Se invece il programma venisse attivato con la seguente riga di comando:

```
temperatura tluoghi.txt Torino 24.5
```

allora deve produrre il seguente output:

```
Torino:
- temperatura media 26.0
- 2 giorni con T > 24.5
```

Soluzione

```
1  /* PROGRAMMAZIONE IN C */
2
3  /* File: temperatura.c */
4  /* Soluzione proposta esercizio "Temperature" */
5
6  #include <stdio.h>
7  #include <stdlib.h>
8  #include <string.h>
9
```

```
10 int main(int argc, char * argv[])
11 {
12     FILE * f;
13     float somma, temperatura ;
14     int cont, r ;
15     char citta[31], riga[255] ;
16 
17     if (argc < 3)
18     {
19         printf ("ERRORE: numero di parametri non sufficiente") ;
20         exit (1) ;
21     }
22 
23     f = fopen (argv[1], "r");
24     if (f == NULL)
25     {
26         printf ("ERRORE: impossibile aprire il file");
27         exit (1) ;
28     }
29 
30     somma = 0;
31     cont = 0;
32     while (fgets (riga, 254, f) != NULL)
33     {
34         r = sscanf (riga, "%f %s", &temperatura, citta);
35         if (r==2)
36         {
37             if (strcmp(argv[2],citta) == 0)
38             {
39                 somma = somma+temperatura;
40                 cont ++;
41             }
42         }
43         else
44             printf("Riga in formato errato - ignorata\n") ;
45     }
46     printf ("La media delle temperature della citta' %s e' %f\n",
47             argv[2], somma/cont);
48     fclose (f);
49     exit(0) ;
50 }
```

8 Esercizio: “Presenze ai corsi”

Un professore vuole realizzare un programma che gli permetta di effettuare delle statistiche sulle presenze ai corsi universitari da lui tenuti.

Ogni corso universitario è caratterizzato da un codice (es. 06AZNDI). Ogni volta che il docente effettua una lezione, deve richiamare il programma per inserire le informazioni relative a tale lezione, ed in particolare: data e numero di studenti presenti alla lezione.

Le informazioni sono memorizzate in un *file di lavoro* denominato `lezioni.txt`. Tale file è composto da un numero variabile, non noto a priori, di righe, ciascuna delle quali contiene le informazioni relative ad una singola lezione. Il file può contenere le informazioni relative a molti corsi diversi, liberamente inframmezzati. Il formato di ciascuna riga del file è il seguente:

```
codice data numstudenti
```

dove:

- `codice` è il codice del corso (max 10 caratteri, senza spazi);
- `data` è la data della lezione, rappresentata come numero intero tra 1 e 365;
- `numstudenti` è il numero di studenti presenti, rappresentato come numero intero positivo.

Il programma viene richiamato con due argomenti sulla linea di comando: il primo argomento indica il codice del corso interessato, mentre il secondo indica l'operazione da eseguire. L'operazione può essere `I` per "inserimento" oppure `S` per "statistiche." In particolare:

- nel caso di inserimento di una nuova lezione (relativa al corso indicato sulla linea di comando), il programma chiederà all'utente le informazioni necessarie (data e numero di studenti) ed aggiornerà il file di lavoro aggiungendovi una riga. Compiuta tale elaborazione, il programma termina.
- stampa delle statistiche di un corso. In tal caso il programma calcola e stampa, per il corso indicato sulla linea di comando, le seguenti quantità: data della lezione con il maggior numero di studenti, data della lezione con il minor numero di studenti, numero medio di studenti presenti alle lezioni. In seguito il programma termina.

Ad esempio, supponendo che il programma sia denominato `registro`, e che il file `lezioni.txt` sia inizialmente vuoto, una possibile interazione con il programma è la seguente (si noti che `c:>` è il *prompt* del sistema operativo):

```
c:> registro 06AZNDI I
Data: 101
Studenti: 40
c:> registro 04KKZWE I
Data: 104
Studenti: 99
c:> registro 06AZNDI I
Data: 98
Studenti: 45
c:> registro 06AZNDI S
Il minimo di studenti si e' raggiunto in data 101
Il massimo di studenti si e' raggiunto in data 98
La media di studenti vale 42.5
```

Soluzione

```
1  /* PROGRAMMAZIONE IN C */
2
3  /* File: registro.c */
4  /* Soluzione proposta esercizio "Presenze ai corsi" */
5
6  #include <stdio.h>
7  #include <stdlib.h>
8  #include <string.h>
9
10 int main( int argc, char *argv[] )
11 {
12     const char nomefile[] = "studenti.txt" ;
13     const int MAX = 100 ;
14
15     char riga[MAX+1] ;
16     char codice[MAX+1] ;
17     int data, stud, r ;
```

```
18      FILE * f ;
19
20      int totStud ; /* somma tutte presenze */
21      int nLezioni ; /* numero di lezioni del corso */
22
23      int minStud, maxStud ;
24      int dataMinStud, dataMaxStud ;
25
26      /* Controlla i parametri ricevuti */
27      /* argv[1] -> codice del corso */
28      /* argv[2] -> comando "I" oppure "S" */
29
30      if ( argc!=3 )
31      {
32          printf("ERRORE: numero di parametri errato\n") ;
33          exit(1) ;
34      }
35
36      if ( strcmp(argv[2], "I")!=0 && strcmp(argv[2], "S")!=0 )
37      {
38          printf("ERRORE: comando %s non riconosciuto\n", argv[2]) ;
39          exit(1) ;
40      }
41
42      /* se il comando è 'I' */
43      if ( strcmp(argv[2], "I")==0 )
44      {
45          /* acquisisci i dati da tastiera */
46
47          printf("Data: ") ;
48          gets(riga) ;
49          r = sscanf( riga, "%d", &data ) ;
50
51          if ( r!=1 || data <1 || data > 366 )
52          {
53              printf("ERRORE: Data assente o non valida\n") ;
54              exit(1) ;
55          }
56
57          printf("Studenti: ") ;
58          gets(riga) ;
59          r = sscanf( riga, "%d", &stud ) ;
60
61          if ( r!=1 || stud <1 )
62          {
63              printf("ERRORE: Numero studenti assente o non valido\n") ;
64              exit(1) ;
65          }
66
67          /* aggiungi una riga al file */
68
69          f = fopen(nomefile, "a") ;
70          if ( f==NULL )
71          {
72              printf("ERRORE: non riesco a modificare il file %s\n", nomefile) ;
73              exit(1) ;
```

```
74          }
75
76          fprintf( f, "%s␣%d␣%d\n", argv[1], data, stud) ;
77
78          fclose(f) ;
79
80      }
81      else if ( strcmp(argv[2], "S")==0 )
82      {
83          /* se il comando è 'S' */
84
85          nLezioni = 0 ;
86          totStud = 0 ;
87
88          minStud = 5000 ;
89          maxStud = -1 ;
90
91          /* leggi tutte le righe il file */
92          f = fopen( nomefile, "r" ) ;
93          if ( f==NULL )
94          {
95              printf("ERRORE:␣impossibile␣leggere␣file␣%s\n", nomefile) ;
96              exit(1) ;
97          }
98
99          while ( fgets(riga, MAX, f) != NULL )
100         {
101             r = sscanf(riga, "%s␣%d␣%d", codice, &data,&stud) ;
102             if ( r!=3 )
103             {
104                 printf("Riga␣in␣formato␣errato␣-␣ignorata\n") ;
105             }
106             else
107             {
108                 /* se la riga è relativa al corso che mi interessa */
109                 if ( strcmp( codice, argv[1] ) == 0 )
110                 {
111                     /* aggiorna statistiche */
112                     nLezioni++ ;
113                     totStud = totStud + stud ;
114
115                     if ( stud > maxStud )
116                     {
117                         maxStud = stud ;
118                         dataMaxStud = data ;
119                     }
120
121                     if ( stud < minStud )
122                     {
123                         minStud = stud ;
124                         dataMinStud = data ;
125                     }
126                 }
127             }
128         }
129
```

```
130         fclose(f) ;
131
132         /* stampa statistiche */
133         if ( nLezioni>=1 )
134         {
135             printf("Il␣minimo␣di␣studenti␣si␣e'␣raggiunto␣in␣data␣%d\n",
136                    dataMinStud) ;
137             printf("Il␣massimo␣di␣studenti␣si␣e'␣raggiunto␣in␣data␣%d\n",
138                    dataMaxStud) ;
139             printf("La␣media␣del␣numero␣di␣studenti␣vale␣%.1f\n",
140                    (double)totStud / (double)nLezioni ) ;
141         }
142         else
143         {
144             printf("Non␣ci␣sono␣lezioni␣del␣corso␣%s\n", argv[1]) ;
145         }
146     }
147
148     exit(0) ;
149 }
```

9 Esercizio: "Media esami"

Si desidera calcolare e stampare il valor medio dei voti riportati dagli studenti in esami universitari. I voti sono riportati in un file di testo il cui nome è fornito come primo parametro sulla linea di comando.

Il file contiene una riga per ogni esame registrato. Ogni riga contiene in sequenza:

- il numero di matricola dello studente (al massimo 6 cifre)
- il codice dell'esame, composto da 4 cifre di cui quella più significativa indica l'anno di corso dell'esame (1 per il primo anno, 2 per il secondo anno, ...)
- la data dell'esame, composta da 8 cifre secondo il formato AAAAMMGG (es. il 23 gennaio 2007 sarebbe indicato come 20070123)
- il voto ottenuto (al massimo 2 cifre).

Non è noto a priori il numero di righe presenti nel file. La media deve essere stampata con una sola cifra dopo la virgola. Si noti che il file contiene la registrazione anche delle insufficienze (ossia voti < 18) ma tali voti non devono essere considerati nel calcolo della media.

Il programma riceve inoltre come ulteriori parametri sulla linea di comando delle indicazioni circa l'insieme di voti da considerare nel calcolo della media, secondo la seguente codifica:

- `-a`*N* media dei voti degli esami dell'anno N-esimo;
- `-s`*M* media dei voti dello studente con matricola M;
- `-e`*C* media dei voti dell'esame con codice C.

Si può assumere che sia presente sempre solo uno di questi tre parametri.

Ad esempio se il file `VOTI.TXT` contenesse i seguenti dati:

```
1234  1001 20050123 30
98765 1001 20050123 18
98765 1021 20050912 21
1234  2027 20051023 28
```

il programma (che si suppone chiamato `media`) dovrebbe generare i seguenti risultati quando attivato come indicato:

linea di comando	*output prodotto*
`media VOTI.TXT -s1234`	`29.0`
`media VOTI.TXT -a1`	`23.0`
`media VOTI.TXT -e1001`	`24.0`

Soluzione

```c
/* PROGRAMMAZIONE IN C */

/* File: media.c */
/* Soluzione proposta esercizio "Media esami" */

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

int main( int argc, char *argv[] )
{
    const int MAX= 100 ;

    char comando ;
    int valore ;

    int r, voto, matricola, codice ;
    int nVoti, totVoti ;

    FILE *f ;
    char riga[MAX+1] ;

    /* controlla gli argomenti */
    /* argv[1] -> nome del file */
    /* argv[2] -> comando
        argv[2][0] == '-'
        argv[2][1] == 'a' oppure 's' oppure 'e'
        argv[2][dal 2 in poi] -> numero intero */

    if ( argc!=3 )
    {
        printf("ERRORE: numero di argomenti errato\n") ;
        exit(1) ;
    }

    r = sscanf( argv[2], "-%c%d", &comando, &valore ) ;

    if ( r!=2 || ( comando!='a' && comando!='s' && comando!='e' ) )
    {
        printf("ERRORE: comando %s non riconosciuto\n", argv[2]) ;
        exit(1) ;
    }
```

```
43
44      /* leggi il file, per ogni riga controlla se deve essere
45      considerata (in funzione di comando) */
46      f = fopen(argv[1],"r") ;
47      if ( f==NULL )
48      {
49          printf("ERRORE:␣impossibile␣aprire␣file␣%s\n", argv[1]) ;
50          exit(1) ;
51      }
52
53      totVoti = 0;
54      nVoti = 0 ;
55
56      while ( fgets(riga, MAX, f) != NULL )
57      {
58          r = sscanf( riga, "%d␣%d␣%*s␣%d", &matricola, &codice, &voto ) ;
59          /* Nota: %*s fa sì che la stringa NON venga memorizzata */
60
61          if ( r == 3 )
62          {
63              if ( ( (comando == 's' && matricola == valore) ||
64                      (comando == 'e' && codice == valore ) ||
65                      (comando == 'a' && (codice/1000) == valore ))
66                      && voto>=18  )
67              {
68                  totVoti = totVoti + voto ;
69                  nVoti++ ;
70              }
71          }
72      }
73      fclose(f) ;
74
75      if ( nVoti>0 )
76      {
77          printf("Valore␣medio:␣%.1f\n", (double)totVoti / (double)nVoti ) ;
78      }
79      else
80      {
81          printf("Non␣ci␣sono␣esami␣che␣soddisfano␣i␣criteri␣di␣ricerca\n") ;
82      }
83
84      exit(0) ;
85  }
```

Soluzione più generale

Nel caso in cui volessimo permettere all'utente di specificare più di un filtro contemporameamente (ad esempio specificando simultaneamente i parametri `-s` e `-a` per indicare che si desidera la media dei voti che uno studente ha riportato in un certo anno di corso), si può ricorrere ad una soluzione più generale, riportata nel seguito.

In questo caso si è preferito definire alcune variabili di tipo logico (*flag*) per ricordare quali comandi sono stati specificati dall'utente: `com_a`, `com_e`, `com_s`. A ciascun flag è associata una variabile che contiene il valore specificato dall'utente come "filtro": `val_a`, `val_e`, `val_s`.

L'algoritmo funziona considerando, per ogni riga del file, se tale riga deve essere considerata o meno, in funzione dei comandi ricevuti. In particolare, se un comando X è assente (`com_X==0`), allora tale riga deve

essere considerata (non filtrata). In caso contrario (`com_X==1`), occorre controllare se il valore è quello corretto (`val_X==...`).

```
/* PROGRAMMAZIONE IN C */

/* File: media2.c */
/* Soluzione proposta esercizio "Media esami" - VERSIONE PIÙ GENERALE */

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

int main( int argc, char *argv[] )
{
    const int MAX= 100 ;

    char comando ;
    int valore ;

    int r, voto, matricola, codice,i ;
    int nVoti, totVoti ;

    FILE *f ;
    char riga[MAX+1] ;

    int com_a,com_e, com_s, val_a, val_e, val_s ;

    /* controlla gli argomenti */
    /* argv[1] -> nome del file */
    /* argv[2] -> comando
        argv[2][0] == '-'
        argv[2][1] == 'a' oppure 's' oppure 'e'
        argv[2][dal 2 in poi] -> numero intero */

    if ( argc<3 )
    {
        printf("ERRORE: numero di argomenti errato\n") ;
        exit(1) ;
    }

    com_s = 0 ;
    com_a = 0 ;
    com_e = 0 ;

    for ( i = 2 ; i<argc; i++ )
    {
        r = sscanf( argv[i], "-%c%d", &comando, &valore ) ;

        if ( r!=2 || ( comando!='a' && comando!='s' && comando!='e' ) )
        {
            printf("ERRORE: comando %s non riconosciuto\n", argv[i]) ;
            exit(1) ;
        }

        if ( comando=='a' )
        {
            com_a = 1 ;
```

```
55              val_a = valore ;
56          }
57          else if (comando=='e')
58          {
59              com_e = 1 ;
60              val_e = valore ;
61          }
62          else if (comando=='s')
63          {
64              com_s = 1 ;
65              val_s = valore ;
66          }
67      }
68
69      /* leggi il file, per ogni riga controlla se deve essere
70      considerata (in funzione di comando) */
71      f = fopen(argv[1],"r") ;
72      if ( f==NULL )
73      {
74          printf("ERRORE: impossibile aprire file %s\n", argv[1]) ;
75          exit(1) ;
76      }
77
78      totVoti = 0;
79      nVoti = 0 ;
80
81      while ( fgets(riga, MAX, f) != NULL )
82      {
83          r = sscanf( riga, "%d %d %*s %d", &matricola, &codice, &voto ) ;
84
85          if ( r == 3 )
86          {
87              if ( ( com_s == 0 || val_s==matricola ) &&
88                      ( com_a == 0 || val_a==codice/1000) &&
89                      ( com_e == 0 || val_e==codice) &&
90                      voto>=18  )
91              {
92                  totVoti = totVoti + voto ;
93                  nVoti++ ;
94              }
95
96          }
97      }
98      fclose(f) ;
99
100     if ( nVoti>0 )
101     {
102         printf("Valore medio: %.1f\n", (double)totVoti / (double)nVoti ) ;
103     }
104     else
105     {
106         printf("Non ci sono esami che soddisfano i criteri di ricerca\n") ;
107     }
108
109     exit(0) ;
110 }
```

10 Esercizio: “Consumi di toner”

Si desidera analizzare la statistica dei consumi di toner di un’azienda per ottimizzare gli acquisti futuri.

La quantità di cartucce di toner prelevate dal magazzino ogni giorno è riportata all’interno di un file di testo il cui nome è passato come primo parametro sulla riga di comando.

Il file contiene una riga per ogni giorno. Ogni riga contiene in sequenza:

- il nome del dipartimento che ha prelevato il toner (una stringa lunga al massimo 5 caratteri);
- un numero intero (valore minimo 1 e massimo 99) che indica la quantità di cartucce di toner prelevate in quel giorno da quel dipartimento.

Non è noto il numero di righe presenti nel file.

Il programma riceve inoltre come secondo argomento sulla linea di comando il nome di un dipartimento per il quale calcolare l’indicatore statistico dato come terzo argomento sulla linea di comando secondo la seguente codifica:

- `-min` indica che si desidera il valore minimo;
- `-max` indica che si desidera il valore massimo;
- `-med` indica che si desidera il valore medio (da stamparsi in output con un cifra dopo la virgola).

Ad esempio se il file `TONER.TXT` contenesse i seguenti dati:

```
CONT 10
MAGAZ 20
CONT 15
```

ed il programma (che si suppone chiamato `stat`) venisse attivato con la seguente linea di comando:

```
stat toner.txt CONT -med
```

allora dovrebbe generare in output la seguente riga;

```
12.5
```

Soluzione

```
1  /* PROGRAMMAZIONE IN C */
2
3  /* File: stat.c */
4  /* Soluzione proposta esercizio "Consumi di toner" */
5
6  #include <stdio.h>
7  #include <stdlib.h>
8  #include <string.h>
9
10 int main( int argc, char *argv[] )
11 {
12     const int MAX = 100 ;
13     const int LUNDIP = 5 ;
14
15     int cont, tot, min, max, r ;
16     FILE * f ;
17     char riga[MAX+1] ;
18     char nomedip[LUNDIP+1] ;
19     int consumo ;
```

```
20
21      if ( argc != 4 )
22      {
23          printf("ERRORE: numero di argomenti errato\n") ;
24          exit(1) ;
25      }
26
27      f = fopen( argv[1], "r" ) ;
28      if ( f==NULL )
29      {
30          printf("ERRORE: impossibile aprire file %s\n", argv[1]) ;
31          exit(1) ;
32      }
33
34      /* Si è scelto di far sì che il programma calcoli comunque tutte e
35      tre le statistiche, e poi stampi solamente quella richiesta.
36      Così facendo il codice è più semplice */
37
38      cont = 0 ;
39      tot = 0 ;
40      max = 0 ;
41      min = 1000 ;
42
43      while ( fgets( riga, MAX, f ) != NULL )
44      {
45          r = sscanf( riga, "%s %d", nomedip, &consumo ) ;
46
47          if ( strcmp( nomedip, argv[2] ) == 0 )
48          {
49              if ( consumo > max )
50                  max = consumo ;
51
52              if ( consumo < min )
53                  min = consumo ;
54
55              tot = tot + consumo ;
56              cont++ ;
57          }
58      }
59
60      fclose(f) ;
61
62      if ( cont>0 )
63      {
64          if ( strcmp( argv[3], "-min" ) == 0 )
65              printf("%d\n", min) ;
66          else if ( strcmp( argv[3], "-max" ) == 0 )
67              printf("%d\n", max) ;
68          else if ( strcmp( argv[3], "-med" ) == 0 )
69              printf("%.1f\n", (double)tot/cont ) ;
70          else
71              printf("Errore: comando %s non riconosciuto\n", argv[3]) ;
72      }
73      else
74          printf("Errore: dipartimento %s non trovato\n", argv[2]) ;
75
```

```
76      exit(0) ;
77  }
```

11 Esercizio: "Ricette di cucina"

Suor Germana vuole realizzare una versione elettronica delle sue famose ricette di cucina, sotto forma di un programma scritto in C. In particolare, si vuole che il programma identifichi quali sono le ricette cucinabili, dato il contenuto attuale del frigorifero di una massaia.

Il programma accede a due file:

1. un file di testo (denominato `Germana.txt`) contenente gli ingredienti necessari per tutte le ricette di Suor Germana secondo il seguente formato:
   - ogni riga è nella forma `ricetta ingrediente quantità`
   - `ricetta` è una stringa (max 20 caratteri, senza spazi) che indica il nome della ricetta
   - `ingrediente` è una stringa (max 20 caratteri, senza spazi) che indica il nome di un ingrediente
   - `quantità` è un numero reale che indica la quantità di tale ingrediente nella ricetta corrispondente
   - sia `ricetta`, sia `ingrediente` sono ripetuti più volte nel file, ma sempre in associazione a ingredienti o ricette diversi
   - non è noto a priori il numero di righe del file, né è specificato alcun ordinamento noto per il file.
2. un file di testo (il cui nome è passato come primo parametro sulla linea di comando) rappresentante il contenuto attuale del frigorifero secondo il seguente formato:
   - ogni riga è nella forma `ingrediente quantità`
   - `ingrediente` corrisponde ad uno degli ingredienti presenti nel file delle ricette
   - `quantità` è un numero reale che identifica la quantità presente di tale ingrediente nel frigorifero
   - ogni ingrediente è presente una sola volta in questo file
   - non è noto a priori il numero di righe del file, né è specificato alcun ordinamento noto per il file.

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

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

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

```
(frigo.txt)
   uovo 1
   olio 0.5
   parmigiano 0.1
```

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

```
  cerca frigo.txt frittata
```

allora dovrebbe produrre il seguente risultato:

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

Soluzione

```
/* PROGRAMMAZIONE IN C */

/* File: cerca.c */
/* Soluzione proposta esercizio "Ricette di cucina" */

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

int main( int argc, char *argv[] )
{
    const int MAXRIGA = 300 ;
    const int MAXINGR = 100 ;
    const int LUN = 20 ;

    const char filericette[] = "Germana.txt" ;

    /* COMPOSIZIONE DELLA RICETTA RICHIESTA */
    char ingredienti[MAXINGR][LUN+1] ;
    double quantita[MAXINGR] ;
    int Ningr ; /* numero ingredienti totale della ricetta */

    FILE * f ;

    int ok[MAXINGR] ;
    int i, r ;
    char riga[MAXRIGA+1] ;
    char ricetta[LUN+1] ;
    char ingr[LUN+1] ;
    double qta, qrichiesta ;
    int richiesto, pos, possibile ;

    /* Acquisisci argomenti sulla linea di comando */
    if ( argc != 3 )
    {
        printf("ERRORE: numero di argomenti errato\n") ;
        exit(1) ;
    }

    /* FASE 1: LEGGI IL FILE Germana.txt E RIEMPI I VETTORI
       ingredienti[], quantita[] SECONDO QUANTO RICHIESTO
       DALLA RICETTA argv[2] */
    f = fopen( filericette, "r" ) ;
    if (f==NULL)
    {
        printf("ERRORE: impossibile aprire il file %s\n", filericette ) ;
        exit(1) ;
    }
```

```
49
50      Ningr = 0 ;
51      while (fgets(riga, MAXRIGA, f) != NULL)
52      {
53          r = sscanf( riga, "%s␣%s␣%lf", ricetta, ingr, &qta ) ;
54
55          if ( r==3 )
56          {
57              if ( strcmp(ricetta, argv[2]) == 0 )
58              {
59                  strcpy( ingredienti[Ningr], ingr ) ;
60                  quantita[Ningr] = qta ;
61                  Ningr ++ ;
62              }
63          }
64          else
65              printf("Riga␣in␣formato␣errato:␣ignorata\n") ;
66      }
67      fclose(f) ;
68
69      if ( Ningr==0 )
70      {
71          printf("ERRORE:␣ricetta␣%s␣non␣trovata\n", argv[2]) ;
72          exit(1) ;
73      }
74
75      /* FASE 2: LEGGI IL FILE argv[1] E CONFRONTA CON GLI
76         INGREDIENTI RICHIESTI */
77
78      /* 2A: leggo argv[1] e per ogni ingrediente aggiorno il
79      vettore ok[] */
80
81      for (i=0; i<Ningr; i++)
82          ok[i] = 0 ;
83
84      f = fopen( argv[1], "r" ) ;
85
86      while ( fgets( riga, MAXRIGA, f ) != NULL )
87      {
88          r = sscanf( riga, "%s␣%lf", ingr, &qta ) ;
89
90          if ( r == 2 )
91          {
92              /* ingr è richiesto? */
93              richiesto = 0 ;
94              for (i=0; i<Ningr; i++)
95                  if ( strcmp(ingr, ingredienti[i]) == 0 )
96                  {
97                      richiesto = 1 ;
98                      qrichiesta = quantita[i] ;
99                      pos = i ;
100                 }
101
102             if ( richiesto==1 )
103             {
104                 if ( qrichiesta <= qta )
```

```
105                 {
106                     ok[pos] = 1 ;
107                     printf("%s: ok\n", ingr) ;
108                 }
109                 else
110                 {
111                     printf("%s: richiesti %f, disponibili %f\n",
112                            ingr, qrichiesta, qta ) ;
113                 }
114             }
115         }
116         else
117             printf("Riga in formato errato: ignorata\n") ;
118     }
119     fclose(f) ;
120 
121     /* 2A: sulla base del vettore ok[] decido se la ricetta
122     e' fattibile */
123     possibile = 1 ;
124     for ( i = 0 ; i<Ningr ; i++ )
125         if ( ok[i]==0 )
126             possibile = 0 ;
127 
128     if ( possibile==1 )
129         printf("Ricetta POSSIBILE!!!\n") ;
130     else
131         printf("Ricetta IMPOSSIBILE\n") ;
132 
133     exit(0) ;
134 }
```