Corso Luigi Einaudi, 55 - Torino

Appunti universitari
Tesi di laurea
Cartoleria e cancelleria
Stampa file e fotocopie
Print on demand
Rilegature

NUMERO : 279 DATA : 16/04/2012

A P P U N T I

STUDENTE : Rombi

MATERIA : Informatica, temi d'esame

Prof. Acquaviva

Il presente lavoro nasce dall'impegno dell'autore ed è distribuito in accordo con il Centro Appunti.
Tutti i diritti sono riservati. È vietata qualsiasi riproduzione, copia totale o parziale, dei contenuti inseriti nel presente volume, ivi inclusa la memorizzazione, rielaborazione, diffusione o distribuzione dei contenuti stessi mediante qualunque supporto magnetico o cartaceo, piattaforma tecnologica o rete telematica, senza previa autorizzazione scritta dell'autore.

ATTENZIONE: QUESTI APPUNTI SONO FATTI DA STUDENTI E NON SONO STATI VISIONATI DAL DOCENTE.
IL NOME DEL PROFESSORE, SERVE SOLO PER IDENTIFICARE IL CORSO.

© Proprietà riservata dell'autore - Digitalizzazione e distribuzione a cura del CENTRO APPUNTI - Corso Luigi Einaudi, 55 - Torino / Pagina 3 di 48

Informatica – I Facoltà di Ingegneria

Prova scritta del 20/6/2005

Sono ammessi libri e appunti (tempo a disposizione: 2h)

Sono dati tre file di testo contenenti le informazioni sulle operazioni di una ditta di import/export, che ha relazioni commerciali con società appartenenti a vari stati.
Un primo file (il cui nome è `STATI.DAT`) contiene l'elenco degli stati (**al massimo 100**) con cui esistono relazioni commerciali, nel seguente formato:

<codice_stato> <nome_stato>

dove *<codice_stato>* è un codice numerico progressivo (un numero intero: 0 per il primo stato, 1 per il secondo, ...), mentre *<nome_stato>* è il nome dello stato, privo di spazi.
Un secondo file (il cui nome `DITTE.DAT`) contiene l'elenco delle ditte (**al massimo 1000**) , con cui esistono relazioni commerciali, nel seguente formato:

<codice_stato> <nome_societa>

dove *<codice_stato>* è il codice dello stato cui appartiene la società, *<nome_societa>* è il nome della società.
Un terzo file (il cui nome è `TRANSAZIONI.DAT`) contiene le informazioni su un certo numero di transazioni commerciali, ognuna delle quali viene rappresentata, su una riga del file, secondo il formato:

<nome_societa> <importo> <data>

dove *<nome_società>* è il nome della società, *<importo>* è un intero, che rappresenta l'importo della transazione (valore negativo per importazione, positivo per esportazione), *<data>* è la data della transazione (formato gg/mm/aaaa).

Si scriva un programma in linguaggio C che legga ed elabori i dati secondo le richieste seguenti.

1. Calcoli il numero totale delle transazioni e i bilanci complessivi delle transazioni di importazione e esportazione (somma degli importi relativi, rispettivamente, a importazioni ed esportazioni), che vanno stampati su video.
2. Calcoli, per ogni stato, il bilancio complessivo commerciale di import/export (cioé la sommatoria, per ogni stato, degli importi relativi a tutte le società appartenenti allo stato). Per ogni stato vanno stampati su video il nome e il bilancio totale.

Esempio:

STATI.DAT	TRANSAZIONI.DAT
0 Stati_Uniti 1 Corea 2 Giappone 3 Francia 4 Germania	Mitsubishi 101 21/01/2004 Chrysler -30 01/02/2004 General_Motors -2000 10/02/2004 Chevrolet 50 04/03/2004 Kia 500 12/03/2004 BMW -1200 15/04/2004 Mercedes -400 23/04/2004 BMW -1000 22/06/2004 Mercedes 300 23/06/2004 Peugeot 403 02/07/2004 General_Motors 1124 05/07/2004 Chevrolet 350 10/08/2004 Citroen -325 11/08/2004 Honda -670 30/08/2004 Peugeot -768 21/09/2004 Kia -800 23/09/2004
DITTE.DAT	
1 Kia 4 BMW 4 Mercedes 3 Peugeot 3 Citroen 2 Honda 2 Mitsubishi 0 Chrysler 0 General_Motors 0 Chevrolet	

Sulla base di questi dati, l'output del programma sarà il seguente:

```
Numero transazioni: 16
Totale importazioni: -7193
Totale esportazioni: 2828

STATO:  Stati_Uniti   BILANCIO:  -506    (ottenuto come: -30-2000+50+1124+350)
STATO:  Corea         BILANCIO:  -300    (ottenuto come: 500-800)
STATO:  Giappone      BILANCIO:  -569    (ottenuto come: 101-670)
STATO:  Francia       BILANCIO:  -690    (ottenuto come: 403-325-768)
STATO:  Germania      BILANCIO:  -2300   (ottenuto come: -1200-400-1000+300)
```

© Proprietà riservata dell'autore - Digitalizzazione e distribuzione a cura del CENTRO APPUNTI - Corso Luigi Einaudi, 55 - Torino / Pagina 3 di 48

© Proprietà riservata dell'autore - Digitalizzazione e distribuzione a cura del CENTRO APPUNTI - Corso Luigi Einaudi, 55 - Torino / Pagina 5 di 48

Esame 4

es2006051[1].c

```
/*********************************************************************/
/* soluzione compito del 20/6/2004                                   */
/*                                                                   */
/* Si evitano per semplicità controlli di errore sul formato dei file */
/* sugli argomenti al main, sull'apertura corretta dei file          */
/* La soluzione proposta prevede due tipi struct (t_stato e t_societa)*/
/* per le informazioni relative, rispettivamente, a stati e societa'. */
/* I primi due file vengono infatti letti caricando i dati un due    */
/* vettori di struct (tabella_stati e tabella_societa). Il codice di  */
/* uno stato non viene inserito nella struttura t_stato in quanto    */
/* coincide con l'indice dello stato stesso nel vettore (0 per il    */
/* primo stato, 1 per il secondo, ...).                              */
/* Per i dati nel terzo file non sono previste tabelle, in quanto non */
/* e' necessario disporre di tutte le informazioni (su tutte le      */
/* transazioni) contemporaneamente: il file viene letto riga per riga */
/* aggiornando le statistiche sulle transazioni (globali e per stato) */
/* mediante i dati appena letti: le date vangono lette ma ignorate,  */
/* in quanto non necessarie alla soluzione del problema.             */
/*********************************************************************/

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

/* tipi struct per stato e societa' */

typedef struct {
  char nome[30];
  int bilancio;
} t_stato;

typedef struct {
  char nome[30];
  int codice_stato;
} t_societa;

/* prototipo di funzione */
int cercaCodiceStato(t_societa tabella_societa[], int n, char nome[]);

int main (void)
{
  FILE *fp;
  int i, nst, nsoc;
  int codice_stato, importo;
  int nt, si, se;
  char nome[30];
  t_stato tabella_stati[100];
  t_societa tabella_societa[1000];

  /* lettura file stati.txt: tabella stati */
  fp = fopen("a:/stati.txt","r");
  nst=0;
  while (fscanf(fp,"%d%s", &codice_stato, nome)!=EOF) {
    nst++;
    strcpy(tabella_stati[codice_stato].nome,nome);
    tabella_stati[codice_stato].bilancio = 0;
  }
  fclose(fp);

  /* lettura file ditte.txt: tabella societa' */
  fp = fopen("a:/ditte.txt","r");
  nsoc=0;
  while (fscanf(fp,"%d%s", &codice_stato, nome)!=EOF) {
```


© Proprietà riservata dell'autore - Digitalizzazione e distribuzione a cura del CENTRO APPUNTI - Corso Luigi Einaudi, 55 - Torino / Pagina 5 di 48

© Proprietà riservata dell'autore - Digitalizzazione e distribuzione a cura del CENTRO APPUNTI - Corso Luigi Einaudi, 55 - Torino / Pagina 7 di 48

Informatica – I Facoltà di Ingegneria

Prova scritta del 14/7/2005

Sono ammessi libri e appunti (tempo a disposizione: 2h)

Si desidera realizzare un programma in linguaggio C che consenta di gestire i conti dei clienti di una bar. Il programma permette di inserire da tastiera il codice numerico di ognuna delle consumazioni di un cliente.
Il codice **0** indica che l'inserimento è finito e che il programma deve stampare il conto; il conto stampato dal programma contiene la descrizione e il costo di ognuna delle consumazioni, nonché l'importo totale dovuto dal cliente.

Si noti che il codice 0 non implica la fine del programma, il quale deve invece ritornare all'inizio dell'inserimento di un nuovo conto. **Il programma termina quando viene introdotto il codice –1.**

La corrispondenza tra il codice di una consumazione e un articolo si trova in un primo file il cui nome viene specificato come primo **parametro sulla linea di comando**, mentre la corrispondenza tra un articolo ed il suo prezzo si trova in un primo file il cui nome viene specificato come secondo **parametro sulla linea di comando**.

Si facciano inoltre le seguenti assunzioni:

- Ciascun conto consiste **al massimo di 10 consumazioni**
- Sono previsti **al massimo 100 articoli diversi.**

Esempio di esecuzione:

Supponendo che il file CODICI.DAT contenga i seguenti dati:

```
1 Birra_Media
2 Birra_Piccola
5 Panino,
12 Coca_Cola
15 Acqua_Minerale
```

e che il file PREZZI.DAT contenga i seguenti dati:

```
Birra_Media     4.00
Birra_Piccola   2.50
Panino          3.50
Coca_Cola       2.00
Acqua_Minerale  1.00
```

Invocando il programma di nome CONTO con il seguente comando: `CONTO CODICI.DAT PREZZI.DAT`
Inserendo i seguenti dati, il programma produrrà il seguente output:

```
Consumazione?  1
Consumazione?  5
Consumazione?  1
Consumazione?  0

Birra_Media    4.00
Panino         3.50
Birra_Media    4.00

Totale 11.50 Euro.

Consumazione?
```

© Proprietà riservata dell'autore - Digitalizzazione e distribuzione a cura del CENTRO APPUNTI - Corso Luigi Einaudi, 55 - Torino / Pagina 7 di 48

© Proprietà riservata dell'autore - Digitalizzazione e distribuzione a cura del CENTRO APPUNTI - Corso Luigi Einaudi, 55 - Torino / Pagina 9 di 48

7


```
         printf("Consumazione?  ");
         scanf("%d", &consumazione);
         if (consumazione <= 0)   /* esce con 0 o -1 */
            break;
         listaConsumazioni[i] = consumazione;
      }
      quanteConsumazioni = i;
      somma = 0;
      for (i=0; i<quanteConsumazioni; i++)
      {
         lc = listaConsumazioni[i]; /* codice della i-esima consumazione */
         printf("%-15s %7.2f\n", prezzi[lc].articolo, prezzi[lc].prezzo);
         somma += prezzi[lc].prezzo;
      }
   }while (consumazione >= 0);     /* esce con -1 */

   return EXIT_SUCCESS;
}

/* Nota: la struttura dati che contiene le informazioni e' un vettore di struct,
l'indice
   del vettore viene usato come numero di codice dell'oggetto; ogni struct contiene
   il nome e il prezzo. Nel vettore l'elemento 0 non viene utilizzato.
*/
```


© Proprietà riservata dell'autore - Digitalizzazione e distribuzione a cura del CENTRO APPUNTI - Corso Luigi Einaudi, 55 - Torino / Pagina 9 di 48

© Proprietà riservata dell'autore - Digitalizzazione e distribuzione a cura del CENTRO APPUNTI - Corso Luigi Einaudi, 55 - Torino / Pagina 11 di 48

050714

```
/*es140705*/
#include <stdio.h>
#include <string.h>
struct articolo {
   char nome[30];
   int cod;
   float pr;
};

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

struct articolo art[100];
FILE *fp1, *fp2;
int i=0, j, cons=0,n=0, k, vett[10];
char prodotto [30];
float tot=0.0, prezzo;

if(argc != 3)  {
   printf ("errore parametri\n");
   return 1;
}
if ((fp1 = fopen (argv[1],"r")) == NULL) {
   printf("Errore nell'apertura del file %s\n", argv[1]);
}
if ((fp2 = fopen (argv[2],"r")) == NULL) {
   printf("Errore nell'apertura del file %s\n", argv[2]);
}

while (fscanf(fp1,"%d%s",&art[i].cod,art[i].nome) != EOF)
   i++;
while (fscanf(fp2,"%s%f",prodotto,&prezzo) != EOF)
   for (j=0; j<i;j++)
      if (strcmp(prodotto,art[j].nome) == 0)
         art[j].pr=prezzo;

while (cons!=-1&&n<10){
   printf("Consumazione? ");
   scanf("%d", &cons);
   if (cons >0){
      vett[n]=cons;
      n++;
         }

   if (cons==0){
         for (j=0; j<n; j++)
             for (k=0; k<i; k++)
                      if (vett[j]==art[k].cod){
                         tot += art[k].pr;
                         printf ("%-30s%5.2f\n",art[k].nome,art[k].pr);
                   }
         printf ("Totale %5.2f Euro.\n",tot);
         tot=0;
         n=0;
         }
   }
getchar();
getchar();
return 0;
}
```


© Proprietà riservata dell'autore - Digitalizzazione e distribuzione a cura del CENTRO APPUNTI - Corso Luigi Einaudi, 55 - Torino / Pagina 11 di 48

© Proprietà riservata dell'autore - Digitalizzazione e distribuzione a cura del CENTRO APPUNTI - Corso Luigi Einaudi, 55 - Torino / Pagina 13 di 48


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

double interpola(double x, FILE *fp);

int main(int argc, char *argv[])
{
   FILE *fp;
   double min, max, val;

   if (argc < 2)
   {
      fprintf(stderr, "Specificare nome del file con funzione\n");
      return EXIT_FAILURE;
   }

   if ((fp = fopen(argv[1], "r")) == NULL)
   {
      fprintf(stderr, "Errore nell'apertura del file\n");
      return EXIT_FAILURE;
   }

   /* scandisce il file per determinare il min e il max, cosi' non serve che la
      funzione restituisca anche questa informazione */
   fscanf(fp, "%lf%*lf", &min);   /* primo valore */
   while (fscanf(fp, "%lf%*lf", &max) != EOF) /* l'ultimo valore resta in max */
      ;

   /* input utente */
   printf("Introdurre un valore: ");
   while (scanf("%lf", &val) != EOF)
   {
      if (val < min || val > max)
         printf("Errore: il valore specificato e' fuori dall'intervallo.\n");
      else
         printf("Il valore della funzione nel punto %f e': %f.\n", val,
interpola(val,fp));
      printf("Introdurre un valore: ");
   }

   fclose(fp);

   return EXIT_SUCCESS;
}

/* funzione che interpola i valori, scandendo ogni volta il file come richiesto
*/
double interpola(double x, FILE *fp)
{
   double tempx, tempy, x1=0, y1=0, x2, y2;
   rewind(fp);
   /* cerca il valore precedente x */
   while (fscanf(fp, "%lf%lf", &tempx, &tempy) != EOF && tempx <= x)
   {
      x1 = tempx;
      y1 = tempy;
   }
   /* x > tempx allora la fscanf ha gia' letto i valori successivi */
   x2 = tempx;
   y2 = tempy;
   /* se x == tempx ed e' l'ultimo valore, x1==x2 e y1==y2, quindi il calcolo e'
ok */

   if (x == x1)
      return y1;
```


© Proprietà riservata dell'autore - Digitalizzazione e distribuzione a cura del CENTRO APPUNTI - Corso Luigi Einaudi, 55 - Torino / Pagina 13 di 48

© Proprietà riservata dell'autore - Digitalizzazione e distribuzione a cura del CENTRO APPUNTI - Corso Luigi Einaudi, 55 - Torino / Pagina 15 di 48

Informatica – I Facoltà di Ingegneria

Prova scritta del 31/1/2006

Sono ammessi libri e appunti (tempo a disposizione: 2h)

Si desidera realizzare un programma in linguaggio C a cui vengono forniti **come parametri sulla riga di comando** due nomi di file e un numero intero **N** maggiore di zero.

Il primo file contiene un testo in formato ASCII, e consiste di un numero indefinito di righe ciascuna formata al massimo da 80 caratteri.

Il programma deve crittografare il contenuto del primo file e scriverlo nel secondo file, anch'esso in formato ASCII.

Il metodo crittografico considerato funziona nel seguente modo: **ogni lettera maiuscola e minuscola dell'alfabeto deve essere sostituita con la lettera dell'alfabeto stesso che si ottiene spostandosi di N posizioni in ordine alfabetico crescente**, partendo dalla lettera che si vuole crittografare. Ad esempio se N=5 e il carattere da crittografare è il carattere "m", il carattere crittografato sarà "r" (5 posizioni nell'ordine alfabetico dopo la "m").

Lo spostamento deve essere effettuato in modo "circolare": Se nello spostamento, dalla lettera considerata, si supera la fine dell'alfabeto, si deve ripartire a contare dall'inizio. Ad esempio, se N=1, e il carattere il carattere da crittografare è "z" il carattere da scrivere nel secondo file sarà il carattere "a".

A sostituzione avvenuta, la riga crittografata deve essere scritta invertita nel secondo file (cioè l'ultimo carattere diventa il primo, il penultimo diventa il secondo e così via).

Tutti gli altri caratteri (punteggiatura, spaziatura) rimangono inalterati.

NOTE:

- Si consiglia di processare un carattere per volta usando le opportune funzioni di interrogazione dell'insieme dei caratteri ASCII

Esempio di esecuzione:

Supponendo di avere N=1 e che il file consista di un'unica riga:

Primo file
`Questo compito e' facilissimo!`

Nel secondo file verrà scritto:
`!pnjttjmdbg 'f pujpnpd putfvr`

Ottenuto invertendo la stringa `Rvftup dpnqjup f' gbdjmjttjnp!`

© Proprietà riservata dell'autore - Digitalizzazione e distribuzione a cura del CENTRO APPUNTI - Corso Luigi Einaudi, 55 - Torino / Pagina 15 di 48

© Proprietà riservata dell'autore - Digitalizzazione e distribuzione a cura del CENTRO APPUNTI - Corso Luigi Einaudi, 55 - Torino / Pagina 17 di 48

Informatica – I Facoltà di Ingegneria

Prova scritta del 28/6/2006

Sono ammessi libri e appunti (tempo a disposizione: 2h)

INDICARE SUL COMPITO IL NOME DEL DOCENTE CON IL QUALE SI E' SEGUITO IL CORSO

Una ditta di tessuti ha in magazzino pezze di varia dimensione e colore, e quando riceve un ordine da un cliente deve soddisfarlo dandogli **la più piccola pezza di dimensione superiore o uguale a quella richiesta**.

Il contenuto del magazzino è memorizzato in un file il cui nome è specificato come primo parametro sulla riga di comando, e che contiene su ogni riga le domensioni di una pezza, nel seguente formato:

<lunghezza> <larghezza > <colore>

dove "lunghezza" e "larghezza" sono due numeri interi, e colore è una stringa di al più 31 caratteri.
Il numero di pezze si può ipotizzare minore di 1000.

Il programma, una volta letto il file, deve, per ogni richiesta dell'utente introdotta da tastiera, individuare la pezza in grado di soddisfarla. Una richiesta può essere soddisfatta **se esiste una pezza di dimensione superiore o uguale a quella richiesta dello stesso colore.**
Si noti che, per soddisfare la richiesta, **le pezze possono essere ruotate di 90 gradi** (per es., una pezza di dimensione 100x50 può soddisfare una richiesta di una pezza 50x100).

Se è possibile soddisfare la richiesta, il programma deve stampare la pezza selezionata ed **eliminarla dal magazzino**; in caso contrario il programma stamperà un messaggio di errore.

Il programma termina quando l'utente specifica una pezza di dimensione 0 x 0.

Esempio di esecuzione:

Contenuto del file specificato sulla linea di comando

```
100 50  verde
100 200 blu
100 200 verde
200 200 rossa
100 200 rossa
100 50  giallo
100 200 indaco

Inserire la richiesta dell'utente
100 50 verde
Pezza migliore: 100x50 verde

Inserire la richiesta dell'utente
100 50 verde
Pezza migliore: 100x200 verde

Inserire la richiesta dell'utente
100 50 verde
Non è possibile soddisfare la richiesta

Inserire la richiesta dell'utente
0 0
Programma terminato
```

NOTA: le parti scritte in grassetto nell'esempio corrispondono all'input da parte dell'utente.

© Proprietà riservata dell'autore - Digitalizzazione e distribuzione a cura del CENTRO APPUNTI - Corso Luigi Einaudi, 55 - Torino / Pagina 17 di 48

© Proprietà riservata dell'autore - Digitalizzazione e distribuzione a cura del CENTRO APPUNTI - Corso Luigi Einaudi, 55 - Torino / Pagina 19 di 48


```
      pezze[iareamin].colore[0] = '\0';  /* elimina la pezza */
    }
    else
      printf("Non e' possibile soddisfare la richiesta\n");

    printf("\nInserire la richiesta dell'utente\n");
    gets(risposta);
    sscanf(risposta, "%d%d%s", &lung, &larg, col);
  }

  printf("Programma terminato\n");

  return EXIT_SUCCESS;
}

/* Nota: lo schema gets() + sscanf() e' necessario per poter ricevere in input
   la coppia di valori 0 0 che non e' seguita dal colore.
   Per semplificare la condizione, si poteva mettere in lunghezza il lato
   maggiore. Per non usare la costante LONG_MAX (massimo valore di un long,
   definita in <limits.h>), si poteve mettere un valore sensato grande a piacere
   es. 2 miliardi (un long e' almeno su 32 bit).
   Il testo non dice niente se una sola delle dimensioni richieste e' 0, da' la
   piu' piccola delle pezze di quel colore con uno dei lati come richiesto.
   Per eliminare una pezza dal vettore di struct, si annulla la stringa del
colore
   (e' la prima delle caratteristiche considerate nel controllo).
   Si poteva alternativamente sostituire la struct da eliminare con l'ultima
   del vettore, accorciandolo di 1:    pezze[iareamin] = pezze[--numpezze];
*/
```


© Proprietà riservata dell'autore - Digitalizzazione e distribuzione a cura del CENTRO APPUNTI - Corso Luigi Einaudi, 55 - Torino / Pagina 19 di 48

© Proprietà riservata dell'autore - Digitalizzazione e distribuzione a cura del CENTRO APPUNTI - Corso Luigi Einaudi, 55 - Torino / Pagina 21 di 48


```
#include <stdio.h>
#include <stdlib.h>
#include <string.h>
#include <math.h>
#define MAXFILENAME 128

main(int argc, char *argv[])
{
   double livello, plivello;
   int ore, pore, minuti, pminuti; /* p sta per precedente */
   int numerrori = 0;
   const double errore = 1e-3;

   char filename[MAXFILENAME];
   FILE *fpIN, *fpOUT;

   if (argc < 2)
   {
      fprintf(stderr, "Specificare il nome del file\n");
      return EXIT_FAILURE;
   }

   strcpy(filename, argv[1]);
   strcat(filename, ".WD");
   if ((fpIN = fopen(filename, "r")) == NULL)
   {
      fprintf(stderr, "Errore nell'apertura del file di input\n");
      return EXIT_FAILURE;
   }
   strcpy(filename, argv[1]);
   strcat(filename, ".WOK.txt");
   if ((fpOUT = fopen(filename, "w")) == NULL)
   {
      fprintf(stderr, "Errore nell'apertura del file di output\n");
      return EXIT_FAILURE;
   }

   if (fscanf(fpIN, "%lf%d%d", &plivello, &pore, &pminuti) == EOF)  /* prima
misurazione */
   {
      fprintf(stderr, "Il file e' vuoto\n");
      return EXIT_FAILURE;
   }
   fprintf(fpOUT, "%10.3f   %-2d   %02d\n", plivello, pore, pminuti);

   while (fscanf(fpIN, "%lf%d%d", &livello, &ore, &minuti) != EOF)
   {
      if (fabs(livello-plivello)/plivello >
((ore*60+minuti)-(pore*60+pminuti))*errore  )
      {
         livello = plivello;
         numerrori++;
      }
      fprintf(fpOUT, "%10.3f   %-2d   %02d\n", livello, ore, minuti);
      plivello = livello;
      pore = ore;
      pminuti = minuti;
   }

   fclose(fpIN);
   fclose(fpOUT);

   printf("Soppresse %d misure non attendbili\n", numerrori);

   return EXIT_SUCCESS;
}

/* Nota:
```


© Proprietà riservata dell'autore - Digitalizzazione e distribuzione a cura del CENTRO APPUNTI - Corso Luigi Einaudi, 55 - Torino / Pagina 21 di 48

© Proprietà riservata dell'autore - Digitalizzazione e distribuzione a cura del CENTRO APPUNTI - Corso Luigi Einaudi, 55 - Torino / Pagina 23 di 48

Informatica – I Facoltà di Ingegneria

Prova scritta del 13/9/2006

Sono ammessi libri e appunti (tempo a disposizione: 2h)

Si scriva un programma in linguaggio C che realizzi il gioco "DINOSAURI". DINOSAURI è un semplice gioco di carte per due giocatori. Il mazzo è composto da 30 carte: 10 rosse, 10 verdi e 10 gialle. Sulle rosse è raffigurato un dinosauro enorme, sulle verdi un dinosauro medio e sulle gialle un dinosauro piccolo.
Unica regola: il dinosauro più grosso mangia il dinosauro più piccolo.

Le carte rosse valgono 5 punti, le carte verdi 3 punti, le carte gialle 1 punto. Inizialmente ogni giocatore ha in mano 15 carte.
In ogni mano ogni giocatore gira una carta e la posa sul tavolo. Se le carte giocate hanno colori diversi chi ha buttato il dinosauro più grande vince la mano e prende tutte le carte sul tavolo. Se invece le carte giocate hanno lo stesso colore si lasciano sul tavolo.

Vince chi, al termine delle 15 mani, totalizza il punteggio più alto. Le carte NON possono essere scelte. I mazzi sono tenuti coperti e le carte vanno giocate (girate) nell'ordine in cui compaiono.
Il programma deve:

- Leggere da un primo file (acquisito come primo parametro sulla riga di comando) le carte del primo giocatore (15 carte).
- Leggere da un secondo file (acquisito come secondo parametro sulla riga di comando) le carte del secondo giocatore (15 carte).
- Giocare la carta girata dal primo giocatore in ogni mano e stamparla a video.
- Giocare la carta girata dal secondo giocatore in ogni mano e stamparla a video.
- Visualizzare il risultato di ogni mano e il punteggio corrente.

Al termine delle 15 mani, stampare il nome del vincitore e il punteggio totale ottenuto dal vincitore.

Ad esempio, supponendo di avere un mazzo di 6 carte, se il file "`carte_primo_giocatore`" contiene le seguenti carte:

```
Gialla
Verde
Rossa
```

e il file "`carte_secondo_giocatore`" contiene le seguenti carte:

```
Gialla
Rossa
Verde
```

Il programma lanciato con "*DINOSAURI carte_primo_giocatore carte_secondo_giocatore*" dovrà generare il seguente output:

```
Punteggio giocatore 1: 0
Punteggio giocatore 2: 0

Mano n. 1
Carta giocatore 1: Gialla
Carta giocatore 2: Gialla
Risultato: Pareggio
Punteggio giocatore 1: 0
Punteggio giocatore 2: 0

Mano n. 2
Carta giocatore 1: Verde
Carta giocatore 2: Rossa
Risultato:  Vince la mano il giocatore 2
Punteggio giocatore 1: 0
Punteggio giocatore 2: 10

Mano n. 3
Carta giocatore 1: Rossa
Carta giocatore 2: Verde
Risultato:  Vince la mano il giocatore 1
Punteggio giocatore 1: 8
Punteggio giocatore 2: 10

Vince il giocatore 2 con 10 punti.
```

© Proprietà riservata dell'autore - Digitalizzazione e distribuzione a cura del CENTRO APPUNTI - Corso Luigi Einaudi, 55 - Torino / Pagina 23 di 48

© Proprietà riservata dell'autore - Digitalizzazione e distribuzione a cura del CENTRO APPUNTI - Corso Luigi Einaudi, 55 - Torino / Pagina 25 di 48


```
      printf("\nVince il giocatore 2 con %d punti.\n", punteggioPluto);
   else
      printf("\nLa partita e' patta con %d punti ciascuno.\n", punteggioPippo);


   return EXIT_SUCCESS;
}

int Colore2Valore(char c[])
{
   switch (c[0])  /* basta il primo carattere */
   {
   case 'R':
   case 'r':
      return 5;   /* non serve il break */
   case 'V':
   case 'v':
      return 3;
   case 'G':
   case 'g':
      return 1;
   }
   return 0;      /* elimina il warning che dice che potrebbe uscire senza
restit un val */
}
```


© Proprietà riservata dell'autore - Digitalizzazione e distribuzione a cura del CENTRO APPUNTI - Corso Luigi Einaudi, 55 - Torino / Pagina 25 di 48

© Proprietà riservata dell'autore - Digitalizzazione e distribuzione a cura del CENTRO APPUNTI - Corso Luigi Einaudi, 55 - Torino / Pagina 27 di 48

Esame 8


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

#define MORSEFILE "morse.txt"
#define NUMLETTERE 26
#define MAXLUNGMORSE 10

int main(int argc, char *argv[])
{
   FILE *fp, *morse;
   enum {CIFRATURA, DECIFRATURA} op;
   int i, c;
   char stringa[MAXLUNGMORSE];
   char codice[NUMLETTERE][MAXLUNGMORSE]; /* matrice di char, ossia vettore di
stringhe.
         Ogni stringa contiene la codifica in codice Morse delle lettere
dell'alfabeto,
         in posizione 0 la codifica della A, in posizione 1 quella della B, etc.
         quindi e' sovrabbondante una struct che contenga la lettera e la sua
codifica */

   if (argc < 3)
   {
      fprintf(stderr, "Specificare operazione e nome del file\n");
      return EXIT_FAILURE;
   }

   /* gestisce l'opzione -c o -d */
   if (strcmp(argv[1], "-c") == 0)
      op = CIFRATURA;
   else if (strcmp(argv[1], "-d") == 0)
      op = DECIFRATURA;
   else
   {
      fprintf(stderr, "Specificare opzione -c o -d\n");
      return EXIT_FAILURE;
   }

   /* apre il file con il codice Morse */
   if ((morse = fopen(MORSEFILE, "r")) == NULL)
   {
      fprintf(stderr, "Errore nell'apertura del file" MORSEFILE "\n");
      return EXIT_FAILURE;
   }

   /* legge il file con il codice Morse e mette la codifica in codice[] */
   /* la lettera dell'alfabeto non serve sia memorizzata: 0 = A, 1 = B etc. */
   for (i=0; i<NUMLETTERE && fscanf(morse, "%*c%s%*c", codice[i])!=EOF; i++)
      ;
   if (i<NUMLETTERE)
   {
      fprintf(stderr, "Lettere dell'alfabeto inferiori a 26...\n");
      return EXIT_FAILURE;
   }
   fclose(morse);

   /* apre il file da [de]codificare */
   if ((fp = fopen(argv[2], "r")) == NULL)
   {
      fprintf(stderr, "Errore nell'apertura del file\n");
      return EXIT_FAILURE;
```


© Proprietà riservata dell'autore - Digitalizzazione e distribuzione a cura del CENTRO APPUNTI - Corso Luigi Einaudi, 55 - Torino / Pagina 27 di 48

© Proprietà riservata dell'autore - Digitalizzazione e distribuzione a cura del CENTRO APPUNTI - Corso Luigi Einaudi, 55 - Torino / Pagina 29 di 48

070131

```
#include <stdio.h>

int main(int argc, char *argv[])
{
  FILE * fp, * fp2;
  char d;
  fp = fopen("w","c:/morse.txt");
  fprintf (fp,"x\n");
  fp2 = fopen ("r","c:/morse.txt");
  fscanf (fp2,"%c",&d);
  printf ("%c",d);
  return 0;
}
```


© Proprietà riservata dell'autore - Digitalizzazione e distribuzione a cura del CENTRO APPUNTI - Corso Luigi Einaudi, 55 - Torino / Pagina 29 di 48

© Proprietà riservata dell'autore - Digitalizzazione e distribuzione a cura del CENTRO APPUNTI - Corso Luigi Einaudi, 55 - Torino / Pagina 31 di 48

Esame 3


```
#include <stdio.h>
#include <stdlib.h>
#include <string.h>
#define MAXNOMI 20
#define MAX_LEN_NOME 15
#define TROVATO 1
#define NONTROVATO 0

int cerca(char nome[], char nomi[][MAX_LEN_NOME], int numnomi);
int incompatibili(char nome1[], char nome2[], FILE *fp);

int main(int argc, char *argv[])
{
   char danzatori[MAXNOMI][MAX_LEN_NOME]={""}, danzatore[MAX_LEN_NOME];
   char danzatrici[MAXNOMI][MAX_LEN_NOME]={""}, danzatrice[MAX_LEN_NOME];
   FILE *fp;
   int numdanzatori, numdanzatrici;
   int esci; /* variabile per uscire dal ciclo: quando vale 1 esce */

   if (argc < 4)
   {
      fprintf(stderr, "Specificare: danzatrici danzatori incompatibili\n");
      return EXIT_FAILURE;
   }

   /* carica il file con con i nomi delle danzatrici */
   if ((fp = fopen(argv[1], "r")) == NULL)
   {
      fprintf(stderr, "Errore nell'apertura del file delle danzatrici\n");
      return EXIT_FAILURE;
   }
   numdanzatrici = 0;
   while (numdanzatrici<MAXNOMI && fscanf(fp, "%s", danzatrici[numdanzatrici]) !=
EOF)
      numdanzatrici++;
   fclose(fp);

   /* carica il file con con i nomi dei danzatori */
   if ((fp = fopen(argv[2], "r")) == NULL)
   {
      fprintf(stderr, "Errore nell'apertura del file dei danzatori\n");
      return EXIT_FAILURE;
   }
   numdanzatori = 0;
   while (numdanzatori<MAXNOMI && fscanf(fp, "%s", danzatori[numdanzatori]) != EOF)
      numdanzatori++;
   fclose(fp);

   /* apre il file degli incompatibili */
   if ((fp = fopen(argv[3], "r")) == NULL)
   {
      fprintf(stderr, "Errore nell'apertura del file degli incompatibili\n");
      return EXIT_FAILURE;
   }

   esci = 0;     /* quando vale 1 esce dal ciclo */
   do
   {
      printf("Inserire una coppia: ");
      scanf("%s%s", danzatrice, danzatore);
      if (strcmp(danzatore, "x") == 0 && strcmp(danzatrice, "x") == 0)
         esci = 1;
      else if (cerca(danzatore,danzatori,numdanzatori) == NONTROVATO)
```


© Proprietà riservata dell'autore - Digitalizzazione e distribuzione a cura del CENTRO APPUNTI - Corso Luigi Einaudi, 55 - Torino / Pagina 31 di 48

© Proprietà riservata dell'autore - Digitalizzazione e distribuzione a cura del CENTRO APPUNTI - Corso Luigi Einaudi, 55 - Torino / Pagina 33 di 48

Informatica – I Facoltà di Ingegneria

Prova scritta del 13 febbraio 2007

Sono ammessi libri e appunti (tempo a disposizione: 2h)

Si scriva un programma in linguaggio C per gestire una sala di ballo. Il programma deve leggere da tre file rispettivamente: l'elenco delle danzatrici; l'elenco dei danzatori; l'elenco delle incompatibilità (caratteriali, di altezza, ...). I nomi dei tre file sono specificati nell'ordine sulla linea di comando. I primi due contengono elenchi di nomi, un nome per riga. I nomi non contengono spazi. Sono presenti al massimo 20 nomi; la lunghezza massima di un nome è 15 caratteri. Il terzo elenca le coppie non valide nel formato nel formato *<danzatrice>* *<danzatore>*, una informazione per riga. Ad esempio:

danzatrici.txt

```
Fiorenza
Maria
Diana
```

danzatori.txt

```
Paolo
Giovanni
Giorgio
Riccardo
```

incompatibili.txt

```
Maria Paolo
Diana Riccardo
Maria Riccardo
Fiorenza Paolo
```

Successivamente il programma legge da tastiera una sequenza di coppie nel formato *<danzatrice>* *<danzatore>* di cui controllare la validità, ovvero: la danzatrice deve comparire nell'elenco; il danzatore deve comparire nell'elenco; la coppia non deve essere incompatibile. Il programma termina quando l'utente inserisce la coppia "x x".

Ad esempio con i file definiti sopra (l'input dell'utente è **evidenziato**):

```
C:\> DANZA danzatrici.txt danzatori.txt incompatibili.txt
Inserire una coppia: Fiorenza Giorgio
Coppia valida!
Inserire una coppia: Fiorenza Luca
Errore: Luca non e` un danzatore
Inserire una coppia: Fiorenza Paolo
Errore: coppia incompatibile
Inserire una coppia: x x
Programma terminato
```

© Proprietà riservata dell'autore - Digitalizzazione e distribuzione a cura del CENTRO APPUNTI - Corso Luigi Einaudi, 55 - Torino / Pagina 33 di 48

© Proprietà riservata dell'autore - Digitalizzazione e distribuzione a cura del CENTRO APPUNTI - Corso Luigi Einaudi, 55 - Torino / Pagina 35 di 48

Informatica – I Facoltà di Ingegneria
Prova scritta del 20 giugno 2007
Sono ammessi libri e appunti (tempo a disposizione: 2h)

Si scriva un programma C che permetta di gestire la classifica di un campionato di calcio. La classifica è memorizzata in un file di testo (per ordine di classifica delle squadre) che ha il seguente formato:

```
<squadra> <punti> <partite giocate> <vinte> <pareggiate> <perse> <gol fatti>
<gol subiti>
```

Ad esempio:

```
INTER        97  38 30  7  1 80 34
ROMA         75  38 22  9  7 74 34
LAZIO        62  38 18 11  9 59 33
MILAN        61  38 19 12  7 57 36
FIORENTINA   58  38 21 10  7 62 31
PALERMO      58  38 16 10 12 58 51
EMPOLI       54  38 14 12 12 42 43
ATALANTA     50  38 12 14 12 56 54
SAMPDORIA    49  38 13 10 15 64 64
UDINESE      46  38 12 10 16 49 55
```

Il numero massimo di squadre che possono partecipare ad un campionato è 20.

In un secondo file compare l'elenco delle partite per la giornata in corso nel formato:

```
<squadra 1> <squadra 2> <goal squadra 1> <goal squadra 2>
```

Ad esempio:

```
INTER ROMA  0 1
LAZIO MILAN 3 1
FIORENTINA PALERMO 1 0
EMPOLI ATALANTA 2 0
SAMPDORIA UDINESE 2 2
```

Si scriva un programma che legga la classifica, le partite, aggiorni la classifica generale e stampi la classifica aggiornata tenendo conto che:

- a parità di punti viene prima la squadra che ha la differenza reti migliore
- a parità anche di differenza reti, viene prima la squadra che ha segnato più gol

Nell'esempio utilizzato, la classifica aggiornata da visualizzare è:

```
INTER        97  39 30  7  2 80 35
ROMA         78  39 23  9  7 75 34
LAZIO        65  39 19 11  9 62 34
FIORENTINA   61  39 22 10  7 63 31
MILAN        61  39 19 12  8 58 39
PALERMO      58  39 16 10 13 58 52
EMPOLI       57  39 15 12 12 44 43
SAMPDORIA    50  39 13 11 15 66 66
ATALANTA     50  39 12 14 13 56 56
UDINESE      47  39 12 11 16 51 57
```

I nomi dei due file sono passati al programma tramite linea di comando (il primo argomento è il file con la classifica, il secondo quello con l'elenco delle partite)

© Proprietà riservata dell'autore - Digitalizzazione e distribuzione a cura del CENTRO APPUNTI - Corso Luigi Einaudi, 55 - Torino / Pagina 35 di 48

© Proprietà riservata dell'autore - Digitalizzazione e distribuzione a cura del CENTRO APPUNTI - Corso Luigi Einaudi, 55 - Torino / Pagina 37 di 48


```
                                        c[i].pu+=3;
                                        c[i].v++;
                                }
                                if (g1==g2)
                                {
                                        c[i].pu+=1;
                                        c[i].pr++;
                                }
                                if (g2<g1)
                                        c[i].ps++;
                        }
                }
        }
        for (i=0; i<nsq-1; i++)
                for (j=0; j<nsq-1; j++)
                        if (comp (c[j],c[j+1]) <= 0)
                        {
                                temp = c[j];
                                c[j] = c[j+1];
                                c[j+1]=temp;
                        }
        for (i=0; i<nsq; i++)
                printf ("%-20s%4d%4d%4d%4d%4d%4d%4d\n",c[i].sq,c[i].pu,
                        c[i].pg,c[i].v,c[i].pr,c[i].ps,c[i].gf,c[i].gs);
        return 0;

}
```


© Proprietà riservata dell'autore - Digitalizzazione e distribuzione a cura del CENTRO APPUNTI - Corso Luigi Einaudi, 55 - Torino / Pagina 37 di 48

© Proprietà riservata dell'autore - Digitalizzazione e distribuzione a cura del CENTRO APPUNTI - Corso Luigi Einaudi, 55 - Torino / Pagina 39 di 48

070812

```
/*pgm per leggere i file della ciurma attuale
degli aspiranti pirati e dei pirati morti
dalla linea di comando e creare la nuova ciurma*/

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

#define MAXCIURMA 100
#define NUM_MORTI 100
#define NUM_RECLUTE 100

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

int i,j,l,npir,naspir,nmor,etapir,trovato_nei_morti;

typedef char stringa [20];
typedef struct  { stringa nome;
                                  int eta;
                                } tpirata;

typedef tpirata tciurma[MAXCIURMA];
typedef tpirata treclute[NUM_RECLUTE];
typedef stringa tmorti[NUM_MORTI];

tciurma myCiurma,nuovaCiurma;
treclute Reclute;
stringa nomepir;
tmorti Morti;

FILE *pfciurma, *pfaspiranti, *pfmorti;

//leggo file e creo lista pirati
pfciurma=fopen(argv[1],"r");
npir=0;
while(fscanf(pfciurma,"%s %d",nomepir,&etapir)!=EOF)
        {
        strcpy(myCiurma[npir].nome,nomepir);
        myCiurma[npir].eta=etapir;
        npir++;
        }
fclose(pfciurma);

//leggo file e creo lista pirati aspiranti;
pfaspiranti=fopen(argv[2],"r");
naspir=0;
while(fscanf(pfaspiranti,"%s %d",nomepir,&etapir)!=EOF)
        {
        strcpy(Reclute[naspir].nome,nomepir);
        Reclute[naspir].eta=etapir;
        naspir++;
        }
fclose(pfaspiranti);

//leggo file e creo lista pirati morti;
pfmorti=fopen(argv[3],"r");
nmor=0;
while(fscanf(pfmorti,"%s",nomepir)!=EOF)
        {
        strcpy(Morti[nmor],nomepir);
        nmor++;
        }
fclose(pfmorti);
```


© Proprietà riservata dell'autore - Digitalizzazione e distribuzione a cura del CENTRO APPUNTI - Corso Luigi Einaudi, 55 - Torino / Pagina 39 di 48

© Proprietà riservata dell'autore - Digitalizzazione e distribuzione a cura del CENTRO APPUNTI - Corso Luigi Einaudi, 55 - Torino / Pagina 41 di 48

Informatica – I Facoltà di Ingegneria

Prova scritta del 23 gennaio 2008

Sono ammessi libri e appunti (tempo a disposizione: 2h)

La compagnia aerea Fly's Flights opera collegamenti aerei tra diversi aeroporti. I codici degli aeroporti serviti sono al massimo 100 e sono elencati in ordine alfabetico in un file che in ogni riga contiene, separati da uno spazio (uno solo):

- il codice dell'aeroporto (es. TRN, FCO, LAX, sempre su 3 caratteri),
- il nome dell'aeroporto (può avere spazi intermedi, ad es. Torino Caselle, Roma Fiumicino, Los Angeles International),

Esempio:

```
AMS Amsterdam Schiphol
FCO Roma Fiumicino
JFK New York John Fitzgerald Kennedy
LAX Los Angeles International
LHR London Heatrow
TRN Torino Caselle
MXP Milano Malpensa
```

Un secondo file contiene le tratte aeree servite dalla compagnia aerea (sono al massimo 100), su ogni riga sono indicati il codice dell'aeroporto di partenza seguito da quello di arrivo, separati da spazi (si noti che se una tratta viene servita in entrambe le direzioni sono presenti due righe con i codici invertiti di posto). Non hanno alcun ordine.

Esempio:

```
TRN FCO
FCO TRN
MXP JFK
FCO JFK
MXP AMS
AMS LHR
LHR MXP
FCO LAX
JFK LAX
MXP FCO
MXP LHR
```

Si scriva un programma di nome FLY in linguaggio C che richieda sulla riga di comando il nome del file con gli aeroporti seguito da quello con le tratte, chieda all'utente di inserire i codici di due aeroporti, verifichi che i due codici esistano, determini se esiste un **volo diretto** tra le due località e lo visualizzi sullo schermo utilizzando i nomi degli aeroporti. Il programma quindi torna a chiedere i codici di altri due aeroporti, mentre l'esecuzione termini quando si inserisce la coppia END END. Se un collegamento diretto non esiste, il programma deve elencare TUTTI i collegamenti con **un solo** scalo intermedio tra i due aeroporti, indicando i nomi dei tre aeroporti. Nel caso non ci fosse alcun collegamento con un solo scalo il programma deve indicarne l'assenza.

Esempio

```
C:\>FLY aeroporti.txt tratte.txt
Inserire partenza e destinazione: TRN CDG
 Codice CDG sconosciuto
Inserire partenza e destinazione: LAX MXP
 Non ci sono collegamenti
Inserire partenza e destinazione: TRN FCO
 Torino Caselle -> Roma Fiumicino
Inserire partenza e destinazione: TRN LAX
 Torino Caselle -> Roma Fiumicino -> Los Angeles International
Inserire partenza e destinazione: MXP LAX
 Milano Malpensa -> New York John Fitzgerald Kennedy -> Los Angeles International
 Milano Malpensa -> Roma Fiumicino -> Los Angeles International
Inserire partenza e destinazione: END END
```

© Proprietà riservata dell'autore - Digitalizzazione e distribuzione a cura del CENTRO APPUNTI - Corso Luigi Einaudi, 55 - Torino / Pagina 41 di 48

© Proprietà riservata dell'autore - Digitalizzazione e distribuzione a cura del CENTRO APPUNTI - Corso Luigi Einaudi, 55 - Torino / Pagina 43 di 48


```
if ((fpAERO = fopen(argv[1], "r")) == NULL)
{
   fprintf(stderr, "Errore nell'apertura del file degli aeroporti\n");
   return EXIT_FAILURE;
}

if ((fpTRATTE = fopen(argv[2], "r")) == NULL)
{
   fprintf(stderr, "Errore nell'apertura del file delle tratte\n");
   return EXIT_FAILURE;
}

/* legge il file con gli aeroporti */
for (i=0; fscanf(fpAERO, "%s ", aeroporti[i].codice) != EOF; i++)
   /* lo spazio dopo %s serve per leggere ed eliminare lo spazio dopo il codic
{
   fgets(aeroporti[i].nome, LEN_NOME, fpAERO);
   /* fgets legge il \n finale e lo include nella stringa,
      bisogna rimuoverlo, ma l'ultima riga potrebbe non averlo */
   if (aeroporti[i].nome[strlen(aeroporti[i].nome)-1] == '\n')
      aeroporti[i].nome[strlen(aeroporti[i].nome)-1] = '\0';
}
fclose(fpAERO);
numAeroporti = i;

/* legge le tratte */
for (i=0; fscanf(fpTRATTE, "%s%s", tratte[i].partenza, tratte[i].arrivo) != EO
   ;
numTratte = i;

/* main loop */
printf("Inserire partenza e destinazione: ");
while (scanf("%s%s", partenza, arrivo ) != EOF &&
       !(strcmp(partenza,"END") == 0 && strcmp(arrivo,"END") == 0))
{
   /* cerca codice partenza */
   if (cercaCodice(partenza, aeroporti, numAeroporti) == CODICE_NON_ESISTENTE)
   {
      printf(" Codice %s sconosciuto\n", partenza);
      continue;
   }
   /* cerca cdice arrivo */
   if (cercaCodice(arrivo, aeroporti, numAeroporti) == CODICE_NON_ESISTENTE)
   {
      printf(" Codice %s sconosciuto\n", arrivo);
      continue;
   }
   /* cerca la tratta diretta */
   trovata = FALSE;
   for (i=0; i<numTratte; i++)
      if (strcmp(partenza,tratte[i].partenza) == 0 && strcmp(tratte[i].arrivo,
      {
         /* trovata */
         /* cerca il nome corrispondente al codice partenza */
         /* cerca il nome corrispondente al codice arrivo */
         printf(" %s -> %s\n", aeroporti[cercaCodice(partenza,aeroporti,numAer
                               aeroporti[cercaCodice(arrivo,aeroporti,numAerop
         trovata = TRUE;
         break;
      }
   if (!trovata) /* cerca lo scalo */
      for (i=0; i<numTratte; i++)
         for (j=0; j<numTratte; j++)
```


© Proprietà riservata dell'autore - Digitalizzazione e distribuzione a cura del CENTRO APPUNTI - Corso Luigi Einaudi, 55 - Torino / Pagina 43 di 48

© Proprietà riservata dell'autore - Digitalizzazione e distribuzione a cura del CENTRO APPUNTI - Corso Luigi Einaudi, 55 - Torino / Pagina 45 di 48

Informatica - 01BHD
Prova scritta esame 19 Gennaio 2010
Sono ammessi libri e appunti (tempo a disposizione : 1 h 30)

All'aeroporto Charles De Gaulle di Parigi sono attesi il volo AZ222 da Milano ed il volo AE333 da Pantelleria. I due relativi gruppi di passeggeri dovranno ripartire per NewYork sul volo AF444.
Quando l'ultimo dei due aerei sarà atterrato, verranno elaborati dal sistema informativo dell'aeroporto i due file **`AZ222.txt`** e **`AE333.txt`**, contenenti **l'elenco dei passeggeri, il peso del bagaglio al seguito e l'indicazione della classe di volo** (**b** per "business" , **t** per "turistic" – vedi esempio) per ciascuno dei corrispondenti voli. **Non sono previsti casi di omonimia.**

Scrivere un programma in linguaggio C che gestisca questi voli intercontinentali con scalo intermedio, a cui vengono **passati i nomi dei file relativi ai due voli in arrivo ed il nome del file relativo al volo in partenza come argomenti sulla linea di comando**.

Il programma deve pertanto preparare il file **`AF444.txt`** contenente la **lista complessiva dei passeggeri in partenza, col relativo peso del bagaglio al seguito e con l'indicazione della relativa classe di volo,** nonché coll'**eventuale extra-costo da pagare**. Quest'ultimo va calcolato come segue: ogni passeggero della classe "turistic" che abbia un bagaglio di peso superiore ai 10 kg dovrà pagare un extra-costo di 10.25 euro per ogni kg eccedente i 10 (tale situazione va segnalata sul file `AF444.txt`). Nessun sovrapprezzo è previsto per i passeggeri in business.

Il programma deve **inoltre stampare una sorta di riassunto a video che includa i seguenti dati: lista dei costi extra per passeggero, numero totale di passeggeri sul volo, peso complessivo dei bagagli, e numero dei passeggeri in business** (vedi esempio).

Esempio di esecuzione (i file nel caso tipico avranno piu' righe!)**:**

file AE333.txt : (formato <nome_cognome> <pesobagaglio> <classedivolo>)

```
Paolo_Rossi      25      b
Mario_Neri       35      t
Dario_Bianchi     8      t
```

file AZ222.txt : (formato <nome_cognome> <pesobagaglio> <classedivolo>)

```
Ugo_Verdi        18      t
Marco_Gialli     23      b
```

file AF444.txt: (formato <nome_cognome> <pesobagaglio> <classedivolo> <extracosto>)

```
Paolo_Rossi      25      b       0.00
Mario_Neri       35      t     256.25
Dario_Bianchi     8      t       0.00
Ugo_Verdi        18      t      82.00
Marco_Gialli     23      b       0.00
```

Riassunto a video:

```
extra costo di euro 256.25 per il passeggero Mario_Neri
extra costo di euro  82.00 per il passeggero Ugo_Verdi
passeggeri sul volo in partenza     :     5
peso bagagli dei passeggeri         :     109 kg
passeggeri in classe business       :     2
```

© Proprietà riservata dell'autore - Digitalizzazione e distribuzione a cura del CENTRO APPUNTI - Corso Luigi Einaudi, 55 - Torino / Pagina 45 di 48

© Proprietà riservata dell'autore - Digitalizzazione e distribuzione a cura del CENTRO APPUNTI - Corso Luigi Einaudi, 55 - Torino / Pagina 47 di 48


```
        volo3[k].extra_costo=0.00;
      }
   fprintf (pf3,"%s %d %c %6.2f ",volo3[k].nomepas,volo3[k].pbag,volo3[k].classep
   pbag2+=volo3[k].pbag;
   k++;
   }
fclose(pf2);
fclose (pf3);

/*calcolo per definire il volo da attrezzare*/
npass=k;
pbagag=pbag1+pbag2;
npasbusiness=npasbus1+npasbus2;
printf ("passeggeri sul volo per New Jork = %d\n", npass );
printf ("peso bagagli dei passeggeri      = %d kg\n", pbagag );
printf ("passeggeri in classe business    = %d\n", npasbusiness );
/*parte esecutiva*/
if ((npass>100)||(pbagag>2000)||(npasbusiness>12))
        {
         printf("\nATTREZZARE IL VOLO AF444 CON L'AEREO BOEJNG 747 \"NewYork City
        }
else
        {
         printf("\nATTREZZARE IL VOLO AF444 CON L'AEREO CONCORDE \"Ile de France\
        }
system ("pause");
}
```


© Proprietà riservata dell'autore - Digitalizzazione e distribuzione a cura del CENTRO APPUNTI - Corso Luigi Einaudi, 55 - Torino / Pagina 47 di 48