ELEMENTI DI INFORMATICA LB
ESERCIZI di ALLENAMENTO

Files, Strutture e Liste

ESERCIZIO n.2
Una agenzia di viaggi memorizza su di un file **di testo** le temperature di località di interesse. Ogni linea del file contiene le seguenti informazioni riguardo ad una località:
· il *nome* della località, rappresentato da una stringa di 20 caratteri (inclusi eventuali spazi bianchi finali e priva di spazi bianchi interni);
· la *temperatura* della località, rappresentata da un intero.

Ad esempio un file di temperature potrebbe contenere:
londra 15
vienna 18
roma 32
parigi 22
amsterdam 12
madrid 38

Si chiede di risolvere i seguenti punti:
1. Progettare le strutture dati C da utilizzarsi per risolvere i punti seguenti.

2. Progettare una funzione C che, preso come parametro il nome di un file di temperature, costruisca e restituisca all'unità chiamante una lista di record contenente un elemento per ogni località nel file. Ogni elemento della lista deve contenere il nome della località ed un codice, stabilito in base alla temperatura della località al seguente modo:
o `f' se la temperatura è minore di 20;
o `t' se la temperatura è compresa tra 20 e 30;
o `c' se la temperatura è maggiore di 30.
L'ordine degli elementi nella lista è irrilevante.

3. Progettare una funzione C che, presi come parametri una lista come quella costruita al punto (2) ed il nome di un file, scriva sul file, una per riga, le località con il loro codice per la temperatura, ordinate secondo tale codice. Devono cioè essere scritte sul file prima tutte le località calde (`c'), poi tutte quelle temperate (`t'), ed infine tutte quelle fredde (`f'). L'ordine delle località aventi lo stesso codice di temperatura è irrilevante. Inoltre, la funzione deve **cancellare la lista** passata come parametro, liberando la memoria occupata dai suoi elementi.

Ad esempio, per il file di temperature di sopra, il file prodotto potrebbe contenere:
roma c
madrid c
parigi t
londra f
vienna f
amsterdam f

SOLUZIONE

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


/* Dichiarazioni di costanti e tipi */

#define LUNGLOC 20

struct localita {
  char nome[LUNGLOC+1];
  char codice;
  struct localita *next;
};
typedef struct localita Localita;
typedef Localita *ListaLocalita;



/* Funzioni che risolvono il punto 2 */

char temp2codice(int temp)
  /* Restituisce il codice corrispondente alla tempertura temp. */
{
  char cod;

  if (temp < 20)
    cod = 'f';
  else if (temp <= 30)
    cod = 't';
  else
    cod = 'c';

  return cod;
}


ListaLocalita leggiLocalita(char *nomefile)
{
  FILE *fp;
  ListaLocalita aux, lista = NULL;
  int temp;
  char nome[LUNGLOC+1];

  if ((fp = fopen(nomefile, "r")) == NULL) {
    printf("Errore in apertura in lettura del file %s\n", nomefile);
    exit(1);
  }

  while (fscanf(fp, "%20s%d", nome, &temp) == 2) {
    aux = malloc(sizeof(Localita));
    strcpy(aux->nome, nome);
    aux->codice = temp2codice(temp);
    aux->next = lista;
    lista = aux;
  }
```

```
  fclose(fp);
  return lista;
}



/* Funzioni che risolvono il punto 3 */

void stampaLocalitaConCodice(FILE *fp, ListaLocalita lista, char codice)
  /* Stampa sul file *fp le localita in lista il cui codice di temperatura e`
     pari a codice.
  */
{
  while (lista != NULL) {
    if (lista->codice == codice)
      fprintf(fp, "%-20s %c\n", lista->nome, codice);
    lista = lista->next;
  }
}


void cancellaLista(ListaLocalita *plis)
{
  ListaLocalita aux;

  while (*plis != NULL) {
    aux = *plis;
    *plis = (*plis)->next;
    free(aux);
  }
}


void stampaLocalita(char *nomefile, ListaLocalita lista)
{
  FILE *fp;

  if ((fp = fopen(nomefile, "w")) == NULL) {
    printf("Errore in apertura in scrittura del file %s\n", nomefile);
    exit(1);
  }

  stampaLocalitaConCodice(fp, lista, 'c');
  stampaLocalitaConCodice(fp, lista, 't');
  stampaLocalitaConCodice(fp, lista, 'f');

  fclose(fp);

  cancellaLista(&lista);
}



/* funzioni ausiliarie per la verifica del programma */

int main (void)
{
```

```
  char nomefile[256];
  ListaLocalita lista;

  printf("Immetti il nome del file di input con le localita: ");
  scanf("%s", nomefile);
  lista = leggiLocalita(nomefile);

  printf("Immetti il nome del file di output: ");
  scanf("%s", nomefile);
  stampaLocalita(nomefile, lista);

  return 0;
}
```

ESERCIZIO n.3

Una compagnia assicurativa registra i principali dati relativi alle polizze RC-auto e i dati relativi agli incidenti automobilistici segnalati all'assicurazione su due file.

Il file relativo agli incidenti contiene per ogni incidente le seguenti informazioni:

- targa dell'autoveicolo a cui si riferisce la segnalazione (stringa di 7 caratteri);
- data dell'incidente (formato a scelta del candidato);
- un carattere che può essere ``S'' oppure ``N'', e che denota se il veicolo è stato responsabile (S) oppure no (N) dell'incidente.

Il file relativo alle polizze contiene per ogni polizza le seguenti informazioni:

- targa dell'autoveicolo (stringa 7 caratteri);
- classe di merito (intero compreso tra 0 e 20);
- nome dell'intestatario della polizza (stringa di al più 20 caratteri **non contenente spazi**).

In base ai dati contenuti nei due file la compagnia deve aggiornare le classi di merito associate alle polizze secondo il seguente criterio: se l'autoveicolo non è responsabile di alcun incidente, la classe di merito viene decrementata di uno (fino ad un minimo di 0), mentre per ogni incidente di cui l'autoveicolo è responsabile, la classe di merito viene incrementata di 2 (fino ad un massimo di 20).

Si richiede di risolvere i seguenti punti:

1. Scrivere le definizioni dei tipi di dato C da utilizzarsi per risolvere i punti successivi.
2. Scrivere una funzione C che riceva come parametro **il nome di un file di incidenti** e, in base alle informazioni contenute in tale file, costruisca e restituisca opportunamente una lista, rappresentata mediante strutture e puntatori, in cui ogni elemento è una coppia <*t*,*n*>, dove *t* è una targa di autoveicolo ed *n* è un intero positivo che denota il numero di incidenti di cui l'autoveicolo è stato responsabile.

3. Scrivere una funzione C che riceva come parametri **il nome di un file di polizze** ed **una lista** di coppie <*t,n*> (come definita al punto 2), ed aggiorni le classi di merito nel file di polizze in base alle informazioni contenute nella lista.

Soluzione

```
/* File: assicura.c */
/* Time-stamp: "2001-05-25 15:44:41 calvanes" */
/* Scopo: soluzione esercizio 1 compito esame di Fondamenti di Informatica del
          14/10/2000 */

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


/* ******* SOLUZIONE PUNTO 1 ******* */

typedef int bool;
#define TRUE 1
#define FALSE 0

#define lung_targa 7
#define merito_max 20
#define lung_nome  20

typedef char TipoTarga[lung_targa + 1];
typedef char TipoNome[lung_nome + 1];

typedef char TipoData[9];          /* formato: GGMMAAAA */
typedef char TipoResponsabilita;   /* valori ammessi: 'S' o 'N' */


struct nodoLista {
  TipoTarga targa;
  int num_incidenti;
  struct nodoLista *next;
};
typedef struct nodoLista NodoLista;
typedef NodoLista *TipoLista;


/* ******* SOLUZIONE PUNTO 2 ******* */


TipoLista CercaTarga(char *t, TipoLista lis)
  /* Cerca il veicolo con targa t in lis e restituisce il puntatore
     all'elemento trovato, se questo esiste, NULL altrimenti.
     Usata al punto 2 per verificare se una targa e` gia` presente nella lista,
        ed al punto 3 per trovare il numero di incidenti. */
{
  TipoLista ris = NULL;
  bool trovato = FALSE;

  while (lis != NULL && !trovato)
    if (strcmp(lis->targa, t) == 0) {
      trovato = TRUE;
```

```
      ris = lis;
    }
    else
      lis = lis->next;

  return ris;
}  /* CercaTarga */


TipoLista ContaIncidenti(char *nomefile)
  /* In base alle informazioni contenute nel file di nome nomefile,
     costruisce una lista di coppie <t,n>, dove t e` la targa di un autoveicolo
     nel file, ed n e` il numero di incidenti di cui l'autoveicolo e` stato
     responsabile. */
{
  FILE *finc;
  TipoTarga targa;
  TipoData data;
  TipoResponsabilita resp;
  TipoLista lis_incidenti = NULL, punt;

  if ((finc = fopen(nomefile, "r")) == NULL) {
    printf("Errore in apertura in lettura di %s\n", nomefile);
    return NULL;
  }

  /* lettura degli incidenti e aggiornamento della lista */

  fscanf(finc, "%7s %8s %c", targa, data, &resp);
  while (!feof(finc)) {
    if (resp == 'S') {
      punt = CercaTarga(targa, lis_incidenti);
      if (punt == NULL) {   /* inserisci un nuovo record in testa alla lista */
        punt = malloc(sizeof(NodoLista));
        strcpy(punt->targa, targa);
        punt->num_incidenti = 1;
        punt->next = lis_incidenti;
        lis_incidenti = punt;
      } else         /* aggiorna il numero di incidenti associato alla targa */
        punt->num_incidenti++;
    }
    fscanf(finc, "%7s %8s %c", targa, data, &resp);
  }

  fclose(finc);
  return lis_incidenti;
}  /* ContaIncidenti */


/* ******* SOLUZIONE PUNTO 3 ******* */

void AggiornaClasse(int *classe, int ninc)
  /* Aggiorna la classe di merito in base al numero di incidenti. */
{
  if (ninc == 0) {
    if (*classe > 0)
      (*classe)--;
  } else
```

```
    if (*classe + ninc * 2 > merito_max)
      *classe = merito_max;
    else
      *classe += ninc * 2;
}  /* AggiornaClasse */


void AggiornaPolizze(char *nomefile, TipoLista lis_incidenti)
{
  FILE *fpolizze, *fappoggio;
  TipoTarga targa;
  int classe;
  TipoNome nome;
  TipoLista punt;
  int ninc;

  /* apertura del file di polizze e di un file di appoggio su cui scrivere le
     polizze aggiornate */

  if ((fpolizze = fopen(nomefile, "r")) == NULL) {
    printf("Errore in apertura in lettura di %s\n", nomefile);
    return;
  }

  if ((fappoggio = fopen("appoggio.txt", "w")) == NULL) {
    printf("Errore in apertura in scritture di %s\n", nomefile);
    return;
  }

  /* lettura delle polizze, aggiornamento e scrittura sul file di appoggio */
  fscanf(fpolizze, "%7s %d %20s", targa, &classe, nome);
  fgetc(fpolizze);
  while (!feof(fpolizze)) {
    punt = CercaTarga(targa, lis_incidenti);
    if (punt == NULL)
      ninc = 0;
    else
      ninc = punt->num_incidenti;
    AggiornaClasse(&classe, ninc);
    fprintf(fappoggio, "%s %2d %s\n", targa, classe, nome);
    fscanf(fpolizze, "%7s %d %20s", targa, &classe, nome);
  }

  fclose(fpolizze);
  fclose(fappoggio);

  /* rinomina il file di polizze in modo da non perderlo;
     non richiesto nella specifica */
  rename(nomefile, "polizze-orig.txt");

  /* rinomina il file di appoggio */
  rename("appoggio.txt", nomefile);
}  /* AggiornaPolizze */


/* ******* PARTE NON RICHIESTA DAL TESTO ******* */

void CancellaLista(TipoLista *lis)
```

```
  /* Pone lis uguale alla lista vuota e rende disponibile la memoria occupata.
     Versione iterativa. */
{
  TipoLista paux;

  while (*lis != NULL) {
    paux = *lis;
    *lis = (*lis)->next;
    free(paux);
  }
}  /* CancellaLista */


int main(void)
{
  char nome_file_incidenti[256], nome_file_polizze[256];
  TipoLista lis;

  printf("Immetti il nome di un file di incidenti: ");
  scanf("%256s", nome_file_incidenti);
  printf("Immetti il nome di un file di polizze: ");
  scanf("%256s", nome_file_polizze);
  lis = ContaIncidenti(nome_file_incidenti);
  AggiornaPolizze(nome_file_polizze, lis);
  CancellaLista(&lis);

  return 0;
}  /* Assicurazione */
```

ESERCIZIO n.4

Una compagnia di traghetti offre 12 viaggi giornalieri per un'isola (ad es., uno ogni ora dalle 8 alle 19), ciascuno identificato da un codice intero compreso tra 0 e 11.Ogni traghetto della compagnia può trasportare autoveicoli per una lunghezza complessiva massima fissata. Il numero di autoveicoli già assegnati e la disponibilità residua in metri per ciascuno dei 12 viaggi sono memorizzati in una opportuna struttura di dati in memoria centrale.

Si richiede di risolvere i seguenti punti:
1. Progettare le strutture di dati da utilizzarsi per risolvere i punti 2 e 3.

2. Progettare una funzione C che, presi in ingresso attraverso opportuni parametri la struttura di dati che memorizza, per ciascuno dei 12 viaggi, il numero di auto veicoli già assegnati e la disponibilità residua in metri, la lunghezza in metri di un autoveicolo (reale), e il codice di un viaggio per cui è richiesto l'imbarco (intero compreso tra 0 e 11),

restituisca alla funzione chiamante il codice del primo viaggio, da quello richiesto in poi, su cui l'autoveicolo può effettivamente essere imbarcato, considerando le disponibilità residue e la lunghezza dell'autoveicolo. La funzione deve inoltre aggiornare in modo opportuno la struttura di dati passata come parametro. Se non vi è più disponibilità residua sufficiente su alcuno dei viaggi da quello richiesto in poi, la funzione deve restituire -1 e lasciare inalterata la struttura di dati.

Ad esempio, se al viaggio 5 sono già assegnati 30 autoveicoli con disponibilità

residua di 4.20 metri, al viaggio 6 sono già assegnati 28 autoveicoli con disponibilità residua di 3.40 metri, e al viaggio 7 sono già assegnati 32 autoveicoli con disponibilità residua di 6.80 metri, allora, ad un autoveicolo di 4.50 metri che richiede l'imbarco sul viaggio 5, può essere assegnato il viaggio 7, e per tale viaggio 7 il numero di autoveicoli assegnati diventa 33 con disponibilità residua di 2.30 metri.

3. Le richieste di imbarco sono memorizzate in ordine di arrivo su un file, una per riga. Per ogni richiesta sono specificati (separati da spazi bianchi): la targa dell'autoveicolo (stringa di 7 caratteri, non contenente spazi bianchi), la lunghezza in metri dell'autoveicolo (reale), il codice del viaggio per cui è richiesto l'imbarco.

Progettare una funzione C che, presi in ingresso attraverso opportuni parametri il nome di un file di richieste di imbarco e la struttura di dati che memorizza, per ciascuno dei 12 viaggi, la disponibilità residua in metri e il numero di autoveicoli già assegnati, costruisca e restituisca alla funzione chiamante una lista, rappresentata attraverso strutture e puntatori, in cui ciascun elemento contiene la targa di un autoveicolo ed il codice del viaggio che gli viene assegnato. Gli autoveicoli devono comparire nella lista nello stesso ordine che avevano nel file. Gli autoveicoli a cui non può essere assegnato alcun viaggio non devono comparire nella lista. La funzione deve inoltre aggiornare la struttura di dati passata come parametro in base alle assegnazioni effettuate. Ovviamente, per determinare il codice del viaggio a cui un autoveicolo può essere assegnato, la funzione dovrà attivare in modo opportuno la funzione sviluppata al punto 2.