Fondamenti di Informatica T-1, 2012/2013 – Modulo 2

Prova d'Esame 6A di Giovedì 12 Settembre 2013 – tempo a disposizione 2h

Prima di cominciare: si scarichi dal sito **`http://esamix.labx`** il file **`StartKit6A.zip`** contenente i file necessari (*progetto Visual Studio* ed eventuali altri file di esempio).
Avvertenze per la consegna: apporre all'inizio di ogni file sorgente un commento contenente i propri dati (**cognome, nome, numero di matricola**) e il **numero** della prova d'esame. Al termine, **consegnare tutti i file sorgenti** e i file contenuti nello StartKit.
Nota: **il main non è opzionale**; i *test* richiesti vanno implementati.
Consiglio: per verificare l'assenza di *warning*, eseguire di tanto in tanto *"Rebuild All"*.

Un noto sito web è specializzato nel vendere biglietti per voli aerei a prezzi vantaggiosi. Per ogni biglietto acquistato l'applicazione web scrive su un file di testo le informazioni relative alla prenotazione del volo aereo, una prenotazione su ogni riga. In particolare vengono salvate nell'ordine le seguenti informazioni, separate da uno spazio: codice unico del **cliente** (una stringa di esattamente 6 caratteri alfanumerici, senza spazi); sigla aeroporto di **partenza** (una stringa di esattamente 3 caratteri alfanumerici, senza spazi); sigla aeroporto di **arrivo** (una stringa di esattamente 3 caratteri alfanumerici, senza spazi); **giorno** della partenza (un intero, compreso tra 1 e 365); e infine **codice del volo** (una stringa di esattamente 5 caratteri alfanumerici). A titolo di esempio si veda il file "**`voli.txt`**" fornito nello StartKit.

Esercizio 1 – Struttura dati Volo e funzioni di lettura e stampa (mod. element.h/.c e voli.h/.c)

Si definisca un'opportuna struttura dati **`Volo`**, al fine di rappresentare i dati relativi alla prenotazione di un volo: in particolare si dovrà tenere traccia del codice del cliente, degli aeroporti di partenza e di arrivo, del giorno del volo e del codice del volo.

Si definisca la funzione:

```
Volo * leggiVoli(char * nomeFile, int * dim);
```

che ricevuta come parametro il nome di un file contenente i voli prenotati tramite il sito web, legga i dati ivi contenuti e restituisca come risultato un vettore di strutture dati di tipo **`Volo`**, allocato dinamicamente. Non è noto a priori quanti voli siano registrati nel file: la funzione quindi tramite il parametro **`dim`** passato per riferimento deve restituire la dimensione del vettore, ovvero il numero di prenotazioni di voli lette dal file. La dimensione del vettore restituito deve essere la minima necessaria per contenere tutti i dati presenti nel file. In caso di errore nell'apertura del file, la funzione deve stampare a video un messaggio di errore, e deve restituire un puntatore a **`NULL`**, e il parametro **`dim`** pari a zero (nota: il programma non deve terminare).

Si definisca poi la procedura:

```
void stampaVoli(Volo * lista, int dim);
```

che, ricevuta in ingresso un vettore di strutture dati di tipo **`Volo`**, e la dimensione **`dim`** di tale vettore, stampi a video l'elenco delle prenotazioni.

Si definisca la funzione:

```
int equals(Volo v1, Volo v2);
```

che, ricevute in ingresso due strutture dati di tipo **`Volo`**, restituisca un valore interpretabile come vero se tali strutture sono uguali. Due prenotazioni sono da considerarsi uguali se tutti i sotto-campi sono identici.

Il candidato abbia cura di realizzare nel main opportuni test al fine di verificare il corretto funzionamento delle funzioni di cui sopra.

Fondamenti di Informatica T-1, 2012/2013 – Modulo 2

Prova d'Esame 6A di Giovedì 12 Settembre 2013 – tempo a disposizione 2h

Esercizio 2 – Estrazione dei voli in partenza e rimozione dei duplicati (moduli element.h/.c e voli.h/.c)

Si definisca una funzione:

```
Volo * filtra(Volo * elenco, int dimV, char * partenza, int * dimR);
```

che ricevuto in ingresso un vettore di strutture dati di tipo **`Volo`** e la dimensione **`dimV`** di tale vettore, restituisca un nuovo vettore allocato dinamicamente, e che contenga i soli voli relativi all'aeroporto di partenza specificato tramite il parametro **`partenza`**. Il vettore restituito come risultato deve essere di dimensione minima possibile; la dimensione di tale vettore deve essere restituita tramite il parametro dimR. Inoltre il vettore restituito deve essere ordinato secondo il seguente criterio: in ordine crescente in base al giorno di partenza, e poi in ordine lessicografico in base al codice del volo. Per ordinare il vettore, il candidato deve utilizzare l'algoritmo bubbleSort visto a lezione.

Si definisca quindi una funzione:

```
list selCliente(Volo * v, int dim, char * cliente);
```

che ricevuta in ingresso un vettore di strutture dati di tipo **`Volo`**, e il numero di elementi ivi contenuti tramite il parametro **`dim`**, restituisca una lista di strutture dati di tipo **`Volo`**, contenente tutte le prenotazioni effettuate dal cliente specificato tramite il parametro **`cliente`**. Si faccia attenzione al fatto che a causa di un piccolo bug l'applicazione web ogni tanto registra due volte la stessa identica prenotazione. La funzione quindi deve evitare di inserire nella lista da restituire gli eventuali duplicati.

Il candidato abbia cura di realizzare nel main opportuni test al fine di verificare il corretto funzionamento delle funzioni di cui sopra.

Esercizio 3 – Identificazione dei viaggiatori frequenti (modulo voli.h/voli.c)

Alcuni viaggiatori sono detti frequenti perché viaggiano spesso, tipicamente partendo sempre dallo stesso aeroporto di partenza. Si deve realizzare un algoritmo che permetta di identificare tali viaggiatori frequenti, a partire dall'aeroporto di partenza. A tal scopo il candidato definisca una procedura:

```
void freq(Volo * v, int dim, char * partenza);
```

che riceva in ingresso un vettore di strutture dati di tipo **`Volo`** e la sua dimensione **`dim`**, e un aeroporto di partenza di cui si intende fare l'analisi. La procedura deve stampare a video, per ogni cliente che parte da quell'aeroporto, tutti i voli prenotati da tale cliente. Il candidato utilizzi le funzioni di cui all'Esercizio 2: in particolare deve procedere a determinare tutte le prenotazioni che interessano l'aeroporto specificato nel parametro d'ingresso; quindi a partire da tale elenco, per ogni cliente, deve determinare la lista di tutte le prenotazioni, e deve stamparle a video. I voli relativi ad un singolo cliente devono essere stampati una e una sola volta, onde evitare possibili confusioni.

Attenzione: il candidato abbia cura di non generare memory leaking, tramite deallocazione opportuna delle liste e degli array di appoggio eventualmente creati.

Il candidato abbia cura di realizzare nel main opportuni test al fine di verificare il corretto funzionamento delle procedure di cui sopra.

Esercizio 4 – Stampa dei viaggiatori frequenti di un certo aeroporto, e de-allocazione memoria (main.c)

Il candidato realizzi nella funzione **`main(…)`** un programma che dopo aver letto dal file fornito nello StartKit l'elenco delle prenotazioni, chieda all'utente di specificare il nome di un aeroporto. Di seguito poi stampi tutti i voli dei clienti che partono da quell'aeroporto, come spiegato nell'Es. 3.

Il candidato abbia cura di de-allocare tutta la memoria allocata dinamicamente, ivi compresa la memoria allocata per le liste. Si noti che potrebbe essere necessario deallocare della memoria anche all'interno delle singole funzioni definite negli esercizi precedenti.

Fondamenti di Informatica T-1, 2012/2013 – Modulo 2

Prova d'Esame 6A di Giovedì 12 Settembre 2013 – tempo a disposizione 2h

"element.h":

```
#ifndef _ELEMENT_H
#define _ELEMENT_H

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

#define DIM 1024

typedef struct {
	char cliente[7];
	char partenza[4];
	char arrivo[4];
	int giorno;
	char codice_volo[6];
} Volo;

typedef Volo element;

int equals(Volo v1, Volo v2);
int compare(Volo v1, Volo v2);

#endif
```

"element.c":

```
#include "element.h"


int equals(Volo v1, Volo v2) {
	return
		(strcmp(v1.cliente, v2.cliente) == 0) &&
		(strcmp(v1.codice_volo, v2.codice_volo) == 0) &&
		(strcmp(v1.partenza, v2.partenza) == 0) &&
		(strcmp(v1.arrivo, v2.arrivo) == 0) &&
		v1.giorno == v2.giorno;
}



int compare(Volo v1, Volo v2) {
	int res;

	res = v1.giorno - v2.giorno;
	if (res == 0) {
		return strcmp(v1.codice_volo, v2.codice_volo);
	}
	else
		return res;
}
```

Fondamenti di Informatica T-1, 2012/2013 – Modulo 2

Prova d'Esame 6A di Giovedì 12 Settembre 2013 – tempo a disposizione 2h

"list.h"

```
#ifndef LIST_H
#define LIST_H

#include "element.h"

typedef struct list_element
{
    element value;
    struct list_element  *next;
} item;

typedef  item* list;

typedef  int boolean;

/* PRIMITIVE  */
list emptylist(void);
boolean empty(list);
list cons(element, list);
element head(list);
list tail(list);

void showlist(list l);
void freelist(list l);
//int member(element el, list l);

//list insord_p(element el, list l);

#endif
```

"list.c":

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

#include "list.h"

/* OPERAZIONI PRIMITIVE */
list  emptylist(void)            /* costruttore lista vuota */
{
    return NULL;
}

boolean  empty(list l)     /* verifica se lista vuota */
{
    return (l==NULL);
}

list  cons(element e, list l)
```

```
{
  list t;       /* costruttore che aggiunge in testa alla lista */
  t=(list)malloc(sizeof(item));
  t->value=e;
  t->next=l;
  return(t);
}

element head(list l) /* selettore testa lista */
{
  if (empty(l)) exit(-2);
  else return (l->value);
}

list  tail(list l)         /* selettore coda lista */
{
  if (empty(l)) exit(-1);
  else return (l->next);
}


void showlist(list l) {
       element temp;
       if (!empty(l)) {
              temp = head(l);
              printf("%s %s %s %d %s\n", temp.cliente, temp.partenza, temp.arrivo,
temp.giorno, temp.codice_volo);
              showlist(tail(l));
              return;
       }
       else {
              printf("\n\n");
              return;
       }
}



void freelist(list l) {
       if (empty(l))
              return;
       else {
              freelist(tail(l));
              free(l);
       }
       return;
}
```

Fondamenti di Informatica T-1, 2012/2013 – Modulo 2

Prova d'Esame 6A di Giovedì 12 Settembre 2013 – tempo a disposizione 2h

"voli.h":

```
#ifndef _VOLI
#define _VOLI

#include "element.h"
#include "list.h"

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

Volo * leggiVoli(char * nomeFile, int * dim);
void stampaVoli(Volo * lista, int dim);

Volo * filtra(Volo * elenco, int dimV, char * aeroporto, int * dimR);
list selCliente(Volo * v, int dim, char * cliente);

void freq(Volo * v, int dim, char * partenza);

#endif
```

"voli.c":

```
#include "voli.h"

Volo * leggiVoli(char * nomeFile, int * dim) {
	Volo temp;
	Volo * result;
	FILE * fp;
	int i;

	*dim = 0;
	fp = fopen(nomeFile, "rt");
	if (fp == NULL) {
		printf("Errore nell'apertura del file %s.", nomeFile);
		return NULL;
	}
	else {
		i=0;
		while (fscanf(fp, "%s%s%s%d%s", temp.cliente, temp.arrivo, temp.partenza,
&temp.giorno, temp.codice_volo) == 5)
			i++;
		rewind(fp);
		*dim = i;
		result = (Volo*) malloc(sizeof(Volo) * i);
		i=0;
		while (fscanf(fp, "%s%s%s%d%s", temp.cliente, temp.partenza, temp.arrivo,
&temp.giorno, temp.codice_volo) == 5) {
			result[i] = temp;
			i++;
		}
		fclose(fp);
		return result;
	}
}
```

Fondamenti di Informatica T-1, 2012/2013 – Modulo 2

Prova d'Esame 6A di Giovedì 12 Settembre 2013 – tempo a disposizione 2h

```
void stampaVoli(Volo * lista, int dim) {
	int i;
	for (i=0; i<dim; i++)
		printf("%s %s %s %d %s\n", lista[i].cliente, lista[i].partenza,
lista[i].arrivo, lista[i].giorno, lista[i].codice_volo);
}

void scambia(Volo *a, Volo *b) {
	Volo tmp = *a;
	*a = *b;
	*b = tmp;
}
// bubble sort
void bubbleSort(Volo * v, int n) {
	int i, ordinato = 0;
	while (n>1 && !ordinato) {
		ordinato = 1;
		for (i=0; i<n-1; i++)
			if (compare(v[i],v[i+1])>0) {
				scambia(&v[i],&v[i+1]);
				ordinato = 0;
			}
		n--;
	}
}

Volo * filtra(Volo * elenco, int dimV, char * partenza, int * dimR) {
	Volo * result = NULL;
	int i, j;
	*dimR = 0;
	for (i=0; i<dimV; i++)
		if (strcmp(partenza, elenco[i].partenza) == 0)
			(*dimR)++;
	result = (Volo * ) malloc (sizeof(Volo) * *dimR);
	j=0;
	for (i=0; i<dimV; i++)
		if (strcmp(partenza, elenco[i].partenza) == 0) {
			result[j] = elenco[i];
			j++;
		}
	bubbleSort(result, *dimR);
	return result;
}

int member(element el, list t) {
	while (!empty(t)) {
		if (equals(el, head(t)))
			return 1;
		t = tail(t);
	}
	return 0;
}

list selCliente(Volo * v, int dim, char * cliente) {
	list result = emptylist();
	int i;
```

```
	for (i=0; i<dim; i++) {
		if (strcmp(cliente, v[i].cliente) == 0 &&
				!member(v[i], result) )
			result = cons(v[i], result);
	}
	return result;
}

void freq(Volo * v, int dim, char * partenza) {
	Volo * filtered;
	int dimF;
	int i, j;
	int trovato = 0;
	list temp;

	filtered = filtra(v, dim, partenza, &dimF);
	for (i=0; i<dimF; i++) {
		trovato = 0;
		for (j=0; j<i && !trovato; j++)
			if (strcmp(v[i].cliente, v[j].cliente)==0)
				trovato = 1;
		if (!trovato) {
			temp = selCliente(filtered, dimF, v[i].cliente);
			showlist(temp);
			freelist(temp);
		}
	}
	free(filtered);
	return;
}
```

Fondamenti di Informatica T-1, 2012/2013 – Modulo 2

Prova d'Esame 6A di Giovedì 12 Settembre 2013 – tempo a disposizione 2h

"main.c":

```
#include <stdio.h>
#include <string.h>
#include <stdlib.h>
#include "element.h"
#include "list.h"
#include "voli.h"

int main() {
	{ // Es. 1
		Volo * v;
		int dim;
		v = leggiVoli("voli.txt", &dim);
		stampaVoli(v, dim);
		free(v);
		printf("\n\n");
	}
	{ // Es. 2
		Volo * v, * f;
		int dimV, dimF;
		list elenco;
		v = leggiVoli("voli.txt", &dimV);
		f = filtra(v, dimV, "BLQ", &dimF);
		stampaVoli(f, dimF);
		printf("\n\n");
		elenco = selCliente(f, dimF, "ITE456");
		showlist(elenco);
		free(v);
		free(f);
		freelist(elenco);
		printf("\n\n");
	}
	{ // Es. 3 & 4
		Volo * v;
		int dimV;
		char partenza[4];
		v = leggiVoli("voli.txt", &dimV);
		printf("nome aeroporto? ");
		scanf("%s", partenza);
		freq(v, dimV, partenza);
		free(v);
		printf("\n\n");
	}
}
```

"voli.txt":

```
ITE456 BLQ JFK 289 AZ348
ITE456 JFK BLQ 291 AZ349
VF698H BLQ CIA 301 BA544
VF700J CIA BLQ 301 BA545
ITE456 BLQ JFK 303 AZ348
ITE456 JFK BLQ 306 AZ349
ITE456 BLQ JFK 307 AZ348
ITE456 BLQ JFK 307 AZ348
ITE456 JFK BLQ 312 AZ349
```