Fondamenti di Informatica T-1, 2013/2014 – Modulo 2

Prova d'Esame 2A di Giovedì 30 Gennaio 2014 – tempo a disposizione 2h

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 offre la possibilità di vendere oggetti usati e supporta a tal scopo un meccanismo di asta online al rialzo (asta "all'inglese"). Le informazioni relative agli oggetti in asta e alle offerte sono memorizzate in due file di testo.
In un primo file di testo, denominato "**`oggetti.txt`**", sono memorizzate le informazioni relative agli oggetti in vendita: per ogni riga sono memorizzate in ordine **l'identificatore** unico dell'oggetto (un intero), a seguire separata da spazio **la data** di chiusura dell'asta (tre interi, nel formato "yyyy mm gg"), e infine un **testo** di descrizione dell'oggetto in vendita (una stringa di al più 1023 caratteri utili, contenente spazi). Ogni riga è sempre terminata da un 'new line', compresa l'ultima riga.
Nel secondo file di testo, di nome "**`offerte.txt`**", sono memorizzate in ordine casuale le offerte fatte per gli oggetti in vendita, una offerta in ogni riga. Su ogni riga sono memorizzate in ordine le seguenti informazioni, tutte separate da spazi: l'identificatore unico del **cliente** che ha fatto l'offerta (un intero), l'identificatore unico dell'**oggetto** al quale si riferisce l'offerta (un intero, come sopra), la **data** in cui è stata fatta l'offerta (tre interi, nel formato "yyyy mm gg") e infine **l'importo** dell'offerta (un float).
Si vedano, a titolo di esempio, i file forniti nello StartKit.

Esercizio 1 – Strutture dati Data, Oggetto, Offerta, e funzioni di lett./scritt. (mod. element.h/c e aste.h/.c)

Si definisca un'opportuna struttura dati **`Data`**, al fine di rappresentare una data (potrà essere la scadenza di una asta, così come la data in cui una offerta è stata effettuata). Si realizzino poi le strutture dati **`Oggetto`** e **`Offerta`**, destinate a contenere rispettivamente le informazioni relative a un oggetto in vendita e ad una offerta fatta per l'acquisto di un oggetto.
Si definisca la funzione:

```
list leggiOggetti(char* fileName);
```

che, ricevuto in ingresso il nome di un file contenente gli oggetti in vendita, legga tale elenco di oggetti e restituisca le informazioni tramite una lista di strutture dati di tipo **`Oggetto`**. In caso di errore nell'apertura del file la funzione deve stampare un messaggio di errore a video e poi far terminare immediatamente il programma.
Si definisca poi una funzione:

```
list ordinaOggetti(list elenco);
```

che, ricevuto in ingresso una lista di strutture dati di tipo **`Oggetto`**, restituisca una nuova lista contenente le stesse strutture dati fornite in ingresso, ma ordinate in base alla data di scadenza, in senso crescente. Cioè in cima alla lista si dovranno trovare gli oggetti la cui asta scade prima in senso temporale.
Si definisca infine la procedura:

```
void stampaOggetti(list elenco);
```

che, ricevuta in ingresso una lista di strutture dati di tipo **`Oggetto`**, stampi a video tale elenco.
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, 2013/2014 – Modulo 2

Prova d'Esame 2A di Giovedì 30 Gennaio 2014 – tempo a disposizione 2h

Esercizio 2 – Lettura offerte, filtraggio e ordinamento (moduli element.h/.c e aste.h/.c)

Il candidato definisca una funzione:

```
Offerta * leggiOfferte(char * fileName, Oggetto target, int * dim);
```

che, ricevuto in ingresso il nome di un file contenente le offerte effettuate e una struttura dati di tipo **`Oggetto`**, restituisca un vettore di strutture dati di tipo **`Offerta`** (vettore allocato dinamicamente e della dimensione minima necessaria), contenente tutte le offerte relative all'oggetto specificato. La funzione deve restituire la dimensione del vettore risultato tramite il parametro **`dim`** passato per riferimento. In caso di errore nell'apertura del file, il programma deve terminare. Si presti attenzione al fatto che per un dato oggetto potrebbero non esserci offerte. In tal caso la funzione deve restituire un puntatore a **`NULL`**, e **`dim`** pari a zero.

Il candidato definisca poi una procedura:

```
void stampaOfferte(Offerta * leOfferte, int dim);
```

che, ricevuta in ingresso un array di strutture dati di tipo **`Offertta`**, e la sua dimensione **`dim`**, stampi a video il contenuto dell'array.

Il candidato definisca poi una procedura:

```
void ordina(Offerta * v, int dim);
```

che, ricevuti in ingresso un vettore di strutture dati di tipo **`Offerta`** e la dimensione di tale vettore, ordini il vettore secondo il seguente criterio: in maniera crescente in base alla data in cui è stata effettuata l'offerta; a parità di data, in base all'importo dell'offerta (in senso crescente); a parità di importo, in base all'identificatore del cliente (ancora in senso crescente). A tal fine, il candidato utilizzi l'algoritmo "quick sort" visto a lezione.

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 delle aste problematiche (modulo aste.h/aste.c)

Purtroppo le aste online possono andare incontro a diversi problemi. Un classico problema accade quando il sistema registra offerte per aste che sono già terminate, cioè registra offerte con data posteriore alla data di chiusura dell'asta. Altre volte può accadere che il sistema registri una offerta il cui importo è inferiore all'offerta precedente: tale offerta doveva essere rifiutata, ma a causa di un qualche bug l'offerta è stata registrata ugualmente.

Il candidato implementi una funzione:

```
Offerta * filtra(Offerta * leOfferte, int dim, Oggetto target, int * dimV);
```

che, ricevuti come parametri un vettore di strutture dati di tipo **`Offerta`** e la sua dimensione **`dim`**, e l'oggetto **`target`** a cui si riferiscono le offerte, restituisca un nuovo vettore di strutture dati di tipo **`Offerta`** e la sua dimensione **`dimV`** (non necessariamente della dimensione minima), dove siano state inserite tutte le offerte corrette. Una offerta è corretta se la data dell'offerta precede o è pari alla data di chiusura dell'asta per l'oggetto e se l'importo è sempre maggiore stretto dell'importo della precedente offerta. Si presti attenzione al fatto che per un dato oggetto potrebbero non esserci offerte (cioè **`dim`** pari a zero). In tal caso la funzione si deve limitare a restituire un puntatore a **`NULL`** e parametro **`dimV`** pari a zero.

Esercizio 4 – Stampa delle offerte vincitrici, e de-allocazione memoria (main.c)

Il candidato realizzi nella funzione **`main(…)`** un programma che, usando le informazioni fornite tramite i file di esempio forniti nello StartKit e le funzioni definite agli esercizi precedenti, stampi a video per ogni oggetto in asta i dati dell'offerta vincitrice dopo aver ovviamente filtrato le offerte non corrette. Qualora non vi siano offerte per un oggetto specifico, il programma deve stampare una apposita stringa per segnalare la problematica.

Al termine del programma, il candidato abbia cura di de-allocare tutta la memoria allocata dinamicamente, ivi compresa la memoria allocata per le liste.

Fondamenti di Informatica T-1, 2013/2014 – Modulo 2

Prova d'Esame 2A di Giovedì 30 Gennaio 2014 – tempo a disposizione 2h

"element.h":

```
#include <stdio.h>
#ifndef _ELEMENT_H
#define _ELEMENT_H
#define DIM_DESCRIZIONE 1024
typedef struct {
	int anno;
	int mese;
	int giorno;
} Data;
typedef struct {
	int id;
	Data scadenza;
	char testo[DIM_DESCRIZIONE];
} Oggetto;
typedef struct {
	int cliente;
	int oggetto;
	Data data_off;
	float importo;
} Offerta;
typedef Oggetto element;
int compareData(Data d1, Data d2);
int compareOggetto(Oggetto o1, Oggetto o2);
int compareOfferta(Offerta o1, Offerta o2);
#endif
```

"element.c":

```
#include "element.h"

int compareOggetto(Oggetto o1, Oggetto o2) {
	return compareData(o1.scadenza, o2.scadenza);
}
int compareData(Data d1, Data d2) {
	if (d1.anno != d2.anno)
		return d1.anno - d2.anno;
	if (d1.mese != d2.mese)
		return d1.mese - d2.mese;
	if (d1.giorno != d2.giorno)
		return d1.giorno - d2.giorno;
	return 0;
}
int compareOfferta(Offerta o1, Offerta o2) {
	int temp;
	float val;
	temp = compareData(o1.data_off, o2.data_off);
	if (temp == 0) {
		val = o1.importo - o2.importo;
		if (val < 0) temp = -1;
		if (val==0) temp = 0;
		if (val > 0) temp = 1;
		if (temp == 0) { temp = o1.cliente - o2.cliente; }
	}
	return temp;
}
```

Fondamenti di Informatica T-1, 2013/2014 – Modulo 2

Prova d'Esame 2A di Giovedì 30 Gennaio 2014 – 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 freelist(list l) {
       if (empty(l))
              return;
       else {
              freelist(tail(l));
              free(l);
       }
       return;
}

list insord_p(element el, list l) {
       list pprec, patt = l, paux;
       int trovato = 0;

       while (patt!=NULL && !trovato) {
              if (compareOggetto(el, patt->value)<0)
                     trovato = 1;
              else {
                     pprec = patt;
                     patt = patt->next;
              }
       }
       paux = (list) malloc(sizeof(item));
       paux->value = el;
       paux->next = patt;
       if (patt==l)
              return paux;
       else {
              pprec->next = paux;
              return l;
       }
}
```

Fondamenti di Informatica T-1, 2013/2014 – Modulo 2

Prova d'Esame 2A di Giovedì 30 Gennaio 2014 – tempo a disposizione 2h

"aste.h":

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

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

#ifndef _ASTE_H
#define _ASTE_H

// es 1
list leggiOggetti(char* fileName);
list ordinaOggetti(list elenco);
void stampaOggetti(list elenco);

// es 2
Offerta * leggiOfferte(char * fileName, Oggetto target, int * dim);
void stampaOfferte(Offerta * leOfferte, int dim);
void ordina(Offerta * v, int dim);

// es 3
Offerta * filtra(Offerta * leOfferte, int dim, Oggetto target, int * dimV);
#endif
```

"aste.c":

```
#include "aste.h"

list leggiOggetti(char* fileName) {
	FILE * fp;
	list result;
	Oggetto temp;
	int i;
	char ch;

	fp = fopen(fileName, "rt");
	if (fp == NULL) {
		printf("Errore nellì'apertura del file: %s\n", fileName);
		system("pause");
		exit(-1);
	}
	else{
		result = emptylist();
		while (fscanf(fp, "%d%d%d%d", &(temp.id), &(temp.scadenza.anno),
			&(temp.scadenza.mese), &(temp.scadenza.giorno)) == 4) {
				i=0;
				ch = fgetc(fp);
				while ( ch != '\n') {
					temp.testo[i] = ch;
					i++;
					ch = fgetc(fp);
				}
				temp.testo[i] = '\0';
				result = cons(temp, result);
		}
```

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

list ordinaOggetti(list elenco) {
	list result;

	result = emptylist();
	while (!empty(elenco)) {
		result = insord_p(head(elenco), result);
		elenco = tail(elenco);
	}
	return result;
}

void stampaOggetti(list elenco) {
	Oggetto temp;
	if (!empty(elenco)) {
		temp = head(elenco);
		printf("%d %d-%d-%d %s\n", temp.id, temp.scadenza.anno,
			temp.scadenza.mese, temp.scadenza.giorno, temp.testo);
		stampaOggetti(tail(elenco));
	}
	return;
}

Offerta * leggiOfferte(char * fileName, Oggetto target, int * dim) {
	FILE * fp;
	int count;
	int i;
	Offerta temp;
	Offerta * result = NULL;

	fp = fopen(fileName, "rt");
	if (fp == NULL) {
		printf("Errore nell'apertura del file: %s\n", fileName);
		system("pause");
		exit(-2);
	}
	else {
		*dim = 0;
		count = 0;
		while (fscanf(fp, "%d%d%d%d%d%f",
				&(temp.cliente), &(temp.oggetto),
				&(temp.data_off.anno), &(temp.data_off.mese),
&(temp.data_off.giorno),
				&(temp.importo)) == 6) {
					if (temp.oggetto == target.id)
						count++;
		};
		rewind(fp);
		if (count>0) {
			result = (Offerta *) malloc(sizeof(Offerta) * count);
			*dim = count;
			i=0;
			while (fscanf(fp, "%d%d%d%d%d%f",
```

```
                                &(temp.cliente), &(temp.oggetto),
                                &(temp.data_off.anno), &(temp.data_off.mese),
&(temp.data_off.giorno),
                                &(temp.importo)) == 6) {
                                        if (temp.oggetto == target.id) {
                                                result[i] = temp;
                                                i++;
                                        }
                        };
                }
                fclose(fp);
        }
        return result;
}

void stampaOfferte(Offerta * leOfferte, int dim) {
        int i;
        for (i=0; i<dim; i++) {
                printf("%d %d %d-%d-%d Euro: %6.2f\n",
                        leOfferte[i].cliente, leOfferte[i].oggetto,
                        leOfferte[i].data_off.anno, leOfferte[i].data_off.mese,
                        leOfferte[i].data_off.giorno, leOfferte[i].importo);
        }
        printf("\n\n");
        return;
}

void scambia(Offerta *a, Offerta *b) {
        Offerta tmp = *a;
        *a = *b;
        *b = tmp;
}

void quickSortR(Offerta a[], int iniz, int fine) {
        int i, j, iPivot;
        Offerta pivot;
        if (iniz < fine) {
                i = iniz;
                j = fine;
                iPivot = fine;
                pivot = a[iPivot];
                do { /* trova la posizione del pivot */
                        while (i < j && compareOfferta(a[i], pivot) <= 0) i++;
                        while (j > i && compareOfferta(a[j], pivot) >= 0) j--;
                        if (i < j) scambia(&a[i], &a[j]);
                } while (i < j);

                /* determinati i due sottoinsiemi */
                /* posiziona il pivot */
                if (i != iPivot && compareOfferta(a[i], a[iPivot]) != 0) {
                        scambia(&a[i], &a[iPivot]);
                        iPivot = i;
                }

                /* ricorsione sulle sottoparti, se necessario */
                if (iniz < iPivot - 1)
                        quickSortR(a, iniz, iPivot - 1);
```

```
            if (iPivot + 1 < fine)
                  quickSortR(a, iPivot + 1, fine);
      } /* (iniz < fine) */
} /* quickSortR */

void ordina(Offerta * v, int dim) {
      quickSortR(v, 0, dim-1);
}

Offerta * filtra(Offerta * leOfferte, int dim, Oggetto target, int * dimV) {
      Offerta * result = NULL;
      int i;

      *dimV = 0;
      if (dim>0) {
            result = (Offerta * ) malloc(sizeof(Offerta) * dim);
            for (i=0; i<dim; i++) {
                  if (compareData(leOfferte[i].data_off, target.scadenza)<=0) {
                        if ((i==0) || (leOfferte[i].importo>leOfferte[i-1].importo)) {
                              result[*dimV] = leOfferte[i];
                              *dimV = *dimV + 1;
                        }
                  }
            }
      }
      return result;
}
```

Fondamenti di Informatica T-1, 2013/2014 – Modulo 2

Prova d'Esame 2A di Giovedì 30 Gennaio 2014 – tempo a disposizione 2h

"main.c":

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

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

int main() {
	{	// es. 1
		list oggetti;
		list oggettiOrd;

		oggetti = leggiOggetti("oggetti.txt");
		stampaOggetti(oggetti);
		printf("\n\n");
		oggettiOrd = ordinaOggetti(oggetti);
		stampaOggetti(oggettiOrd);
		printf("\n\n");
		freelist(oggetti);
		freelist(oggettiOrd);
	}
	{ // es. 2
		list oggetti;
		Offerta * leOfferte;
		int dim;

		oggetti = leggiOggetti("oggetti.txt");
		leOfferte = leggiOfferte("offerte.txt", head(oggetti), &dim);
		stampaOfferte(leOfferte, dim);
		printf("\n\n");
		ordina(leOfferte, dim);
		stampaOfferte(leOfferte, dim);
		printf("\n\n");
		freelist(oggetti);
		free(leOfferte);
	}
	{	// es. 3 e 4
		list temp;
		list oggetti;
		Offerta * leOfferte;
		Offerta * filtrate;
		Oggetto target;
		int dim;
		int dimF;

		oggetti = leggiOggetti("oggetti.txt");
		temp = oggetti;
		while (!empty(temp)) {
			target = head(temp);
			leOfferte = leggiOfferte("offerte.txt", target, &dim);
			ordina(leOfferte, dim);
			filtrate = filtra(leOfferte, dim, target, &dimF);
			if (dimF>0) {
				printf("Oggetto %d - %s\n", target.id, target.testo);
```

```
                        printf("Offerta vincitrice: cliente %d, euro: %f\n\n",
filtrate[dimF-1].cliente, filtrate[dimF-1].importo);
                    }
                    else {
                        printf("Nessuna offerta per l'oggetto %d - %s\n", target.id,
target.testo);
                    }
                    temp = tail(temp);
                    if (dim>0)
                        free(leOfferte);
                    if (dimF>0)
                        free(filtrate);
                }
                freelist(oggetti);
        }

        system("pause");
        return 0;
}
```

Fondamenti di Informatica T-1, 2013/2014 – Modulo 2

Prova d'Esame 2A di Giovedì 30 Gennaio 2014 – tempo a disposizione 2h

"oggetti.txt":

```
57 2014 02 15 Pentola usata con calcare e manico rotto
24 2014 02 28 Ceppo di legno con accetta
34 2014 01 20 Video terminale dec compatibile VT100
61 2014 01 30 Macchina fotografica bella fuori ma rotta dentro
```

"offerte.txt":

```
192 34 2014 01 10 56.34
168 34 2014 01 09 35.00
168 34 2014 01 12 70.00
192 34 2014 01 13 76.01
168 34 2014 01 14 78.00
169 34 2014 01 20 99.99
231 24 2014 02 27 67.00
268 24 2014 02 28 65.00
356 57 2014 02 16 3.00
389 57 2014 02 18 5.00
```