Fondamenti di Informatica T-1, 2016/2017 – Modulo 2

Prova d'Esame 2A di Giovedì 26 Gennaio 2017 – 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 famoso ristorante organizza pranzi e cene di lavoro per delle aziende: in seguito ad accordi tra queste aziende e il ristorante, i dipendenti consumano i pasti ma non pagano subito; il ristorante segna in un file di testo le informazioni relative al pasto, e a fine mese invia alle azienda un consuntivo di quanto deve essere pagato in totale.
In particolare in un primo file di testo di nome "**`pasti.txt`**" il ristorante scrive le informazioni relative ai pasti consumati, un pasto su ogni riga. Per ogni pasto il ristorante scrive il **cognome** del dipendente (una stringa di al più 31 caratteri utili, senza spazi); a seguire, il **numero** di persone che hanno preso parte a quel pasto (un intero); infine, **l'importo** di quel pasto (un float). Non è noto a priori quante righe siano presenti nel file.
In un secondo file di testo di nome "**`aziende.txt`**" il ristorante memorizza (su ogni riga) il cognome di un dipendente e il **nome** dell'azienda per cui lavora (una stringa di al più 63 caratteri utili senza spazi), ogni coppia <cognome azienda> su una singola riga. Non è noto a priori quante righe siano presenti nel file.
Si vedano, a titolo di esempio, i file di testo forniti nello StartKit.

Esercizio 1 – Strutture dati Pasto, e Dipendente, e funzioni di lett./scritt. (mod. element.h e rist.h/.c)

Si definiscano opportune strutture dati **`Pasto`**, (per memorizzare il cognome del dipendente, il numero di persone che vi hanno preso parte e l'importo relativo ad un pasto) e **`Dipendente`** (cognome del dipendente e azienda per cui lavora).

Si definisca la funzione:

```
Pasto leggiPasto(FILE * fp);
```

che, ricevuto in ingresso un puntatore ad una struttura dati di tipo **`FILE`**, contenente i dati relativi ai pasti, legga i dati contenuti su una sola riga e li restituisca tramite una struttura dati di tipo **`Pasto`**. Si noti che il file di testo "**`pasti.txt`**" può contenere delle righe in cui manca l'importo (alcune righe cioè sono incomplete dell'ultimo campo). In tali casi, la funzione deve ignorare tale riga e provare a leggere la riga successiva. Qualora si sia giunti al termine del file, o non sia possibile leggere, la funzione deve restituire una struttura dati con valore del campo cognome pari alla stringa "**`NULL`**".

Si definisca la funzione:

```
Pasto * leggiTutti(char * fileName, int * dim);
```

che, ricevuto in ingresso il nome di un file, legga da tale file tutte le informazioni relative ai pasti e le restituisca tramite un array di strutture dati di tipo **`Pasto`** allocato dinamicamente (della dimensione minima necessaria). Tramite il parametro **`dim`** passato per riferimento, la funzione deve restituire la dimensione del vettore. Poiché nel file alcuni pasti potrebbero avere un campo mancante (l'importo), tali pasti non devono ovviamente essere inseriti nell'array. Qualora la funzione incontri dei problemi nell'apertura del file, la funzione dovrà stampare un messaggio di errore a video, restituire un puntatore a **`NULL`**, e zero come valore della dimensione.
Si definisca la procedura:

```
void stampaPasti(Pasto * v, int dim);
```

che, ricevuto in ingresso un vettore di strutture dati di tipo **`Pasto`**, e la sua dimensione **`dim`**, stampi a video le informazioni riguardo i pasti.
Il candidato abbia cura di realizzare nel main opportuni test al fine di verificare il corretto funzionamento delle funzioni di cui sopra, avendo cura di deallocare la memoria, se necessario.

Fondamenti di Informatica T-1, 2016/2017 – Modulo 2

Prova d'Esame 2A di Giovedì 26 Gennaio 2017 – tempo a disposizione 2h

Esercizio 2 –Ordinamento e lettura aziende (moduli element.h/.c e rist.h/.c)

Il candidato definisca una procedura:

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

che, ricevuti in ingresso un vettore di strutture dati di tipo **`Pasto`** e la dimensione di tale vettore, ordini il vettore secondo il seguente criterio: in ordine alfabetico (lessicografico) rispetto al cognome del dipendente; a parità di cognome, in ordine decrescente in base al prezzo (i pranzi più costosi vengono *prima* nel vettore). A tal scopo, il candidato utilizzi l'algoritmo di ordinamento "insert sort" visto a lezione.

Si definisca poi una funzione:

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

che, ricevuto in ingresso il nome di un file contenente i cognomi dei dipendenti e le aziende per cui lavorano, restituisca una lista di strutture dati di tipo **`Dipendente`**, contenente le coppie cognome – azienda per cui lavora il dipendente. La lista dovrà essere ordinata in ordine alfabetico in base all'azienda; a parità di azienda, in base al cognome del dipendente. Qualora vi siano problemi di lettura o apertura del file, la funzione deve restituire una lista vuota.
Il candidato abbia cura di realizzare nel main opportuni test al fine di verificare il corretto funzionamento delle funzioni di cui sopra, avendo cura di deallocare la memoria, se necessario.

Esercizio 3 – Conteggio dei totali (modulo rist.h/rist.c)

Il ristorante, alla fine di ogni mese, deve inviare all'azienda la richiesta di pagamento per tutti i pranzi che i dipendenti di quella azienda hanno consumato presso il ristorante.
Si definisca una funzione:

```
float totaleAzienda(Pasto * v, int dim, list elenco, char * azienda);
```

che ricevuti come parametri un vettore di strutture dati di tipo **`Pasto`**, la sua dimensione, una lista di strutture dati di tipo **`Dipendente`**, e il nome di una azienda specifica, restituisca la somma di tutti gli importi che il ristorante deve chiedere all'azienda specificata. Tale somma dovrà comprendere tutti i pasti consumati dai dipendenti dell'azienda specificata.

Si definisca una procedura:

```
void totali(Pasto * v, int dim, list elenco);
```

che ricevuti come parametri un vettore di strutture dati di tipo **`Pasto`**, la sua dimensione, una lista di strutture dati di tipo **`Dipendente`**, stampi a video i totali dovuti da ogni azienda, senza ripetizioni.

Esercizio 4 Stampa dei risultati, e de-allocazione memoria (main.c)

Il candidato realizzi nella funzione **`main(…)`** un programma che legga dai file i dati relativi ai pasti e alle aziende, e stampi a video quanto ogni azienda deve pagare al ristorante.

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, 2016/2017 – Modulo 2

Prova d'Esame 2A di Giovedì 26 Gennaio 2017 – tempo a disposizione 2h

"element.h":

```
#include <string.h>

#ifndef _ELEMENT_H
#define _ELEMENT_H

#define DIM_COGNOME 32
#define DIM_AZIENDA 64

typedef struct {
	char cognome[DIM_COGNOME];
	int numero;
	float importo;
} Pasto;

typedef struct {
	char cognome[DIM_COGNOME];
	char azienda[DIM_AZIENDA];
} Dipendente;

typedef Dipendente element;

int compare(Pasto p1, Pasto p2);
int compareList(Dipendente d1, Dipendente d2);

#endif
```

"element.c":

```
#include "element.h"

int compare(Pasto p1, Pasto p2) {
	int result;
	result = strcmp(p1.cognome, p2.cognome);
	if (result == 0) {
		if (p2.importo<p1.importo)
			result = -1;
		else
			if (p2.importo>p1.importo)
			result = 1;
	}
	return result;
}

int compareList(Dipendente d1, Dipendente d2) {
	int result;
	result = strcmp(d1.azienda, d2.azienda);
	if (result == 0)
		result = strcmp(d1.cognome, d2.cognome);
	return result;
}
```

Fondamenti di Informatica T-1, 2016/2017 – Modulo 2

Prova d'Esame 2A di Giovedì 26 Gennaio 2017 – 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 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);
}
```

Fondamenti di Informatica T-1, 2016/2017 – Modulo 2

Prova d'Esame 2A di Giovedì 26 Gennaio 2017 – tempo a disposizione 2h

```
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("Cognome: %s, Azienda: %s\n",
                     temp.cognome, temp.azienda);
              showlist(tail(l));
              return;
       }
       else {
              printf("\n\n");
              return;
       }
}
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 (compareList(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, 2016/2017 – Modulo 2

Prova d'Esame 2A di Giovedì 26 Gennaio 2017 – tempo a disposizione 2h

"rist.h":

```
#ifndef _RIST_H
#define _RIST_H

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

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

// Es. 1
Pasto leggiPasto(FILE * fp);
Pasto * leggiTutti(char * fileName, int * dim);
void stampaPasti(Pasto * v, int dim);

// Es. 2
void ordina(Pasto * v, int dim);
list elenco(char * fileName);

// Es. 3
float totaleAzienda(Pasto * v, int dim, list elenco, char * azienda);
void totali(Pasto * v, int dim, list elenco);

#endif
```

"rist.c":

```
#include "rist.h"

Pasto leggiPasto(FILE * fp) {
	Pasto result;
	int ok;

	do {
		ok = fscanf(fp, "%s%d%f", result.cognome, &(result.numero),
&(result.importo));
	} while (ok!=3 && ok>0);

	if (ok!=3)
		strcpy(result.cognome, "NULL");
	return result;
}

Pasto * leggiTutti(char * fileName, int * dim) {
	FILE * fp;
	Pasto * result;
	Pasto temp;
	int i;

	result = NULL;
	*dim = 0;

	fp = fopen(fileName, "rt");
	if (fp!=NULL) {
		i = 0;
```

```
		temp = leggiPasto(fp);
		while (strcmp(temp.cognome, "NULL")) {
			i++;
			temp = leggiPasto(fp);
		}
		rewind(fp);
		result = (Pasto *) malloc(sizeof(Pasto) * i);
		i = 0;
		temp = leggiPasto(fp);
		while (strcmp(temp.cognome, "NULL")) {
			result[i] = temp;
			i++;
			temp = leggiPasto(fp);
		}
		*dim = i;
		fclose(fp);
	}
	else {
		printf("Errore nell'apertura del file %s\n", fileName);
	}
	return result;
}

void stampaPasti(Pasto * v, int dim) {
	int i;
	for (i=0; i<dim; i++) {
		printf("%s %d %f\n", v[i].cognome, v[i].numero, v[i].importo);
	}
	return;
}

void insOrd(Pasto v[], int pos){
	int i = pos-1;
	Pasto x = v[pos];
	while (i>=0 && compare(x,v[i])<0) {
		v[i+1]= v[i];
		i--;
	}
	v[i+1]=x;
}
void insertSort(Pasto v[], int n) {
	int k;
	for (k=1; k<n; k++)
		insOrd(v,k);
}
void ordina(Pasto * v, int dim) {
	insertSort(v, dim);
}

list elenco(char * fileName) {
	list result;
	FILE * fp;
	Dipendente temp;

	result = emptylist();
	fp = fopen(fileName, "rt");
	if (fp != NULL) {
```

```
		while (fscanf(fp, "%s%s", temp.cognome, temp.azienda) == 2)
			result = insord_p(temp, result);
		fclose(fp);
	}
	return result;
}

float totaleAzienda(Pasto * v, int dim, list elenco, char * azienda) {
	float result;
	int i;
	Dipendente temp;

	result = 0;
	while (!empty(elenco)) {
		temp = head(elenco);
		if (strcmp(temp.azienda, azienda)==0) {
			for (i=0; i<dim; i++) {
				if (strcmp(temp.cognome, v[i].cognome)==0)
					result = result + v[i].importo;
			}
		}
		elenco = tail(elenco);
	}
	return result;
}

void totali(Pasto * v, int dim, list elenco) {
	float tot;
	Dipendente current;
	list temp;

	temp = elenco;
	while (!empty(temp)) {
		current = head(temp);
		if (empty(tail(temp)) || strcmp(current.azienda, head(tail(temp)).azienda) )
{
			tot = totaleAzienda(v, dim, elenco, current.azienda);
			printf("Totale dovuto dall'azienda %s: %.2f\n", current.azienda, tot);
		}
		temp = tail(temp);
	}
	return;
}
```

Fondamenti di Informatica T-1, 2016/2017 – Modulo 2

Prova d'Esame 2A di Giovedì 26 Gennaio 2017 – tempo a disposizione 2h

"main.c":

```
#include "element.h"
#include "rist.h"

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

int main() {
	{	// Es. 1
		Pasto * v;
		int dim;
		v= leggiTutti("pasti.txt", &dim);
		stampaPasti(v, dim);
		free(v);
	}
	{	// Es. 2
		Pasto * v;
		int dim;
		list dip;
		v= leggiTutti("pasti.txt", &dim);
		ordina(v, dim);
		printf("\n\n");
		stampaPasti(v, dim);
		dip = elenco("aziende.txt");
		showlist(dip);
		freelist(dip);
		free(v);
	}
	{	// Es. 3 && 4
		Pasto * v;
		int dim;
		list dip;
		v= leggiTutti("pasti.txt", &dim);
		dip = elenco("aziende.txt");
		printf("\n\n");
		totali(v, dim, dip);
		freelist(dip);
		free(v);
	}

	return 0;
}
```

Fondamenti di Informatica T-1, 2016/2017 – Modulo 2

Prova d'Esame 2A di Giovedì 26 Gennaio 2017 – tempo a disposizione 2h

"pasti.txt":

```
chesani 3 67.30
mello 1 5.60
giannelli 3 45.80
chesani 5
chesani 4 110.00
chesani 4 98.00
mello 1 9.80
montali 3 56.60
montali 3 57.80
```

"aziende.txt"

```
chesani unibo
mello unibo
montali unibz
giannelli unife
```