Fondamenti di Informatica T-1, 2014/2015 – Modulo 2

Prova d'Esame 1A di Martedì 13 Gennaio 2015 – 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"*.

Una famosa catena di negozi di dispositivi elettronici vuole realizzare un registro informatico degli articoli in vendita, con le loro caratteristiche e altre informazioni quali il prezzo. In particolare, la catena memorizza in un file di testo le informazioni relative ad ogni articolo, un articolo per ogni riga (non è noto a priori quanti articoli vi siano memorizzati). All'inizio di ogni riga c'è il **nome** dell'articolo (una stringa di al più 255 caratteri utili, contenenti spazi): per evitare confusioni, il nome dell'articolo è sempre posto tra una coppia di doppi apici ("). A seguire, separato da uno spazio, c'è il **prezzo** (un float). Quindi, dopo un altro spazio, vi è una sequenza di **informazioni tecniche** sull'articolo, organizzate nella forma di "stringa_feature stringa_valore", ogni elemento di tale coppia separato da uno spazio, e ogni coppia separata dalla successiva ancora da uno spazio. Entrambe le stringhe di ogni coppia sono al massimo di 127 caratteri utili, non contengono spazi, e per ogni articolo sono memorizzate al massimo 16 coppie feature/valore (ma spesso ve ne sono meno). Ogni riga è terminata da un carattere di "a-capo", compresa anche l'ultima riga. Si veda a titolo di esempio il file di testo fornito nello StartKit.

Esercizio 1 – Strutture dati Feature, Articolo, e funzioni di lett./scritt. (mod. element.h e articoli.h/.c)

Si definisca un'opportuna struttura dati **`Feature`**, al fine di rappresentare il nome della feature tecnica, e il valore assunto da tale feature (entrambe stringhe di al più 127 caratteri utili). Si definisca poi la struttura dati **`Articolo`**, al fine di rappresentare i dati relativi al singolo articolo (nome, prezzo, ed elenco delle feature).
Si definisca la funzione:

```
int leggiSingoloArticolo(FILE * fp, Articolo * dest);
```

che, ricevuto in ingresso un puntatore ad un file, legga da tale file le informazioni relative ad un solo articolo, e restituisca tali informazioni tramite la struttura dati **`dest`** di tipo **`Articolo`** passata per riferimento. La funzione restituisce un valore intero interpretabile come "vero" se la lettura è andata a buon fine, falso in caso contrario. Si assuma per semplicità che il file di input sia sempre ben formato secondo le specifiche date sopra: quindi l'unico caso in cui la funzione restituirà un valore "falso", è quando si sarà giunti al termine del file. Si noti che tale condizione è facilmente rilevabile dal fatto che, da specifiche, tutte le righe cominciano sempre col carattere "doppi apici", cioè ' " '.

Si definisca la funzione:

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

che, ricevuto in ingresso il nome di un file di testo contenente gli articoli, restituisca una lista di strutture dati di tipo **`Articolo`**, contenente tutte le informazioni presenti nel file il cui nome è passato come parametro. Qualora si verifichino dei problemi nell'apertura del file, la funzione deve restituire una lista vuota.

Si definisca la procedura:

```
void stampaArticoli(list elenco);
```

che, ricevuto in ingresso una lista di strutture dati di tipo **`Articolo`**, stampi a video il contenuto di tali strutture dati.

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, 2014/2015 – Modulo 2

Prova d'Esame 1A di Martedì 13 Gennaio 2015 – tempo a disposizione 2h

Esercizio 2 – Filtro e ordinamento (moduli element.h/.c e articoli.h/.c)

Il candidato definisca una funzione:

```
Articolo * filtra(list elenco, float prezzoMassimo, int * dim);
```

che, ricevuto in ingresso una lista di strutture dati di tipo **`Articolo`**, e un float indicante il prezzo massimo che si è disposti a pagare per l'articolo, restituisca un array allocato della dimensione minima, contenente tutti gli articoli che hanno prezzo pari o inferiore a quello specificato in ingresso. Tramite l'intero **`dim`** passato per riferimento la funzione dovrà restituire la dimensione dell'array. Qualora non vi siano articoli con prezzo inferiore a quello specificato, il puntatore restituito dovrà valere NULL, mentre la dimensione dovrà essere pari a zero.

Il candidato definisca poi una procedura:

```
void stampaArray(Articolo * elenco, int dim);
```

che, ricevuto in ingresso un array di strutture dati di tipo **`Articolo`**, e la sua dimensione **`dim`**, ne stampi a video il contenuto.

Il candidato definisca poi una procedura:

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

che, ricevuti in ingresso un vettore di strutture dati di tipo **`Articolo`** e la dimensione di tale vettore, ordini il vettore secondo il seguente criterio: in maniera crescente in base al prezzo; a parità di prezzo, in ordine lessicografico in base al nome dell'articolo. A tal fine, il candidato utilizzi l'algoritmo "bubble 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 –Ricerca degli articoli desiderati (modulo articoli.h/articoli.c)

Si vuole realizzare una funzionalità che permetta di ricercare tutti gli articoli che abbiano una feature tra un insieme di caratteristiche specificate dall'utente. A tal scopo, il candidato definisca una funzione:

```
list ricerca(list elenco);
```

che, ricevuto come parametro una lista di strutture dati di tipo **`Articolo`**, provveda a chiedere all'utente di inserire tramite tastiera una prima feature, e poi una seconda, etc. etc. . L'utente segnala il desiderio di non specificare più feature inserendo la stringa specifica "end". La funzione deve restituire una nuova lista di strutture dati di tipo **`Articolo`**, contenente tutti gli articoli che contengono almeno una tra le feature specificate dall'utente, senza ripetizioni. Due articoli sono uguali se hanno lo stesso nome.

Esercizio 4 – Stampa dei articoli interessanti, e de-allocazione memoria (main.c)

Il candidato realizzi nella funzione **`main(…)`** un programma che, usando le informazioni fornite tramite il file di esempio fornito nello StartKit e le funzioni definite agli esercizi precedenti, chieda all'utente di specificare le feature a cui è interessato (tramite la funzione di cui all'es. 3), e poi chieda di specificare un prezzo massimo e stampi a video l'elenco degli articoli che soddisfano entrambi i criteri di feature e di prezzo (tramite le funzioni di cui all'es. 2).

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, 2014/2015 – Modulo 2

Prova d'Esame 1A di Martedì 13 Gennaio 2015 – tempo a disposizione 2h

"element.h":

```
#include <stdio.h>

#ifndef _ELEMENT_H
#define _ELEMENT_H

#define DIM_FEATURE 128
#define DIM_NOME 256
#define NUM_FEATURE 16

typedef struct {
	char key[DIM_FEATURE];
	char value[DIM_FEATURE];
} Feature;

typedef struct{
	char nome[DIM_NOME];
	float prezzo;
	Feature elenco[NUM_FEATURE];
	int numFeature;
} Articolo;

typedef Articolo element;

int compare(Articolo a1, Articolo a2);

#endif
```

"element.c":

```
#include <string.h>

#include "element.h"



int compare(Articolo a1, Articolo a2) {
	if (a1.prezzo < a2.prezzo)
		return -1;
	else if (a1.prezzo > a2.prezzo)
		return 1;
	else
		return strcmp(a1.nome, a2.nome);
}
```

Fondamenti di Informatica T-1, 2014/2015 – Modulo 2

Prova d'Esame 1A di Martedì 13 Gennaio 2015 – 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;
}
```

Fondamenti di Informatica T-1, 2014/2015 – Modulo 2

Prova d'Esame 1A di Martedì 13 Gennaio 2015 – tempo a disposizione 2h

"articoli.h":

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

#ifndef _ARTICOLI_H
#define _ARTICOLI_H

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

// Es. 1
int leggiSingoloArticolo(FILE * fp, Articolo * dest);
list leggiArticoli(char* fileName);
void stampaArticoli(list elenco);


// Es. 2
Articolo * filtra(list elenco, float prezzoMassimo, int * dim);
void stampaArray(Articolo * elenco, int dim);
void ordina(Articolo * v, int dim);


// Es. 3
list ricerca(list elenco);



#endif
```

"articoli.c":

```
#include "articoli.h"

int leggiSingoloArticolo(FILE * fp, Articolo * dest) {
	char temp;
	int dimNome = 0;
	int numFeature = 0;

	// lettura del nome dell'articolo
	temp = fgetc(fp); // doppi apici...
	if (temp == '"') {
		do {
			temp = fgetc(fp);
			if (temp != '"' && dimNome<DIM_NOME-1) {
				(*dest).nome[dimNome] =  temp;
				dimNome++;
			}
		} while (temp!='"' && dimNome<DIM_NOME-1);
		(*dest).nome[dimNome] = '\0';

		// lettura prezzo
		fscanf(fp, "%f", &((*dest).prezzo) );

		// lettura feature
```

```
		(*dest).numFeature = 0;
		numFeature = 0;
		do {
			temp = fgetc(fp);
			if (temp==' ' && numFeature<NUM_FEATURE) {
				fscanf(fp, "%s%s", (*dest).elenco[numFeature].key,
(*dest).elenco[numFeature].value);
				numFeature++;
			}
		} while (temp==' ' && numFeature<NUM_FEATURE);
		(*dest).numFeature = numFeature;
		return 1;
	}

	return 0;
}

list leggiArticoli(char* fileName) {
	list result = emptylist();
	FILE * fp;
	Articolo temp;
	int ok;

	fp = fopen(fileName, "rt");
	if (fp != NULL) {
		do {
			ok = leggiSingoloArticolo(fp, &temp);
			if (ok)
				result = cons(temp, result);
		} while (ok);
		fclose(fp);
	}

	return result;
}

void stampaArticoli(list elenco) {
	Articolo temp;
	int i;

	while (!empty(elenco)) {
		temp = head(elenco);
		printf("Nome: %s\n", temp.nome);
		printf("Prezzo: %.2f\n", temp.prezzo);
		for (i=0; i<temp.numFeature; i++) {
			printf("\t%s : %s\n", temp.elenco[i].key, temp.elenco[i].value);
		}

		elenco = tail(elenco);
	}
}

Articolo * filtra(list elenco, float prezzoMassimo, int * dim) {
	int size;
	Articolo * result = NULL;
	list temp;
	Articolo tempArt;
```

```
	int trovato;

	*dim = 0;
	size = 0;
	temp = elenco;

	while (!empty(temp)) {
		tempArt = head(temp);
		if (tempArt.prezzo<=prezzoMassimo)
			size++;
		temp = tail(temp);
	}
	if (size > 0) {
		temp = elenco;
		result = (Articolo *) malloc(sizeof(Articolo) * size);
		size = 0;
		while (!empty(temp)) {
			tempArt = head(temp);
			if (tempArt.prezzo<=prezzoMassimo) {
				result[size] = tempArt;
				size++;
			}
			temp = tail(temp);
		}
		*dim = size;
	}
	return result;
}

void stampaArray(Articolo * elenco, int dim) {
	int i,j;
	for (i=0; i<dim; i++) {
		printf("Nome: %s\n", elenco[i].nome);
		printf("Prezzo: %.2f\n", elenco[i].prezzo);
		for (j=0; j<elenco[i].numFeature; j++) {
			printf("\t%s : %s\n", elenco[i].elenco[j].key,
elenco[i].elenco[j].value);
		}
	}
	return;
}

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

void bubbleSort(Articolo 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--;
      }
}

void ordina(Articolo * v, int dim) {
      bubbleSort(v, dim);
}

int hasFeature(Articolo art, char * unaFeature) {
      int i;
      int trovato = 0;
      for (i=0; i<art.numFeature && !trovato; i++) {
            if (strcmp(unaFeature, art.elenco[i].key)==0)
                  trovato = 1;
      }
      return trovato;
}

int member(Articolo art, list elenco) {
      int result = 0;
      while (!empty(elenco) && !result) {
            if (strcmp(art.nome, head(elenco).nome)==0)
                  result = 1;
            elenco = tail(elenco);
      }
      return result;
}

list ricerca(list elenco) {
      char feature[DIM_FEATURE];
      list temp;
      list result;
      Articolo tempArt;

      result = emptylist();

      do {
            printf("Feature da cercare? ");
            scanf("%s", feature);
            if (strcmp("end", feature)) {
                  temp = elenco;
                  while (!empty(temp)) {
                        tempArt = head(temp);
                        if (hasFeature(tempArt, feature) && !member(tempArt, result) )
                              result = cons(tempArt, result);
                        temp = tail(temp);
                  }
            }
      } while (strcmp("end", feature));
      return result;
}
```

Fondamenti di Informatica T-1, 2014/2015 – Modulo 2

Prova d'Esame 1A di Martedì 13 Gennaio 2015 – tempo a disposizione 2h

"main.c":

```
#include "element.h"
#include "articoli.h"

int main() {
	// es. 1
	{
		char nomeFile[] = "articoli.txt";
		list elenco;
		printf("ESERCIZIO 1\n");
		elenco = leggiArticoli(nomeFile);
		stampaArticoli(elenco);
		printf("\n\n");
		freelist(elenco);
	}
	// es. 2
	{
		char nomeFile[] = "articoli.txt";
		list elenco;
		Articolo * vett;
		int dim;
		printf("ESERCIZIO 2\n");
		elenco = leggiArticoli(nomeFile);
		vett = filtra(elenco, 10000.0, &dim);
		ordina(vett,dim);
		stampaArray(vett, dim);
		printf("\n\n");
		free(vett);
		freelist(elenco);
	}
	// es. 3
	{
		char nomeFile[] = "articoli.txt";
		list elenco;
		list filtrato;
		int dim;
		printf("ESERCIZIO 3\n");
		elenco = leggiArticoli(nomeFile);
		filtrato = ricerca(elenco);
		stampaArticoli(filtrato);
		printf("\n\n");
		freelist(elenco);
		freelist(filtrato);
	}
	// es. 4
	{
		char nomeFile[] = "articoli.txt";
		list elenco;
		list filtrato;
		Articolo * vett;
		int dim;
		float prezzo;
		printf("ESERCIZIO 4\n");
		elenco = leggiArticoli(nomeFile);
		filtrato = ricerca(elenco);
		printf("Prezzo massimo: ");
```

```
            scanf("%f", &prezzo);
            vett = filtra(filtrato, prezzo, &dim);
            ordina(vett,dim);
            stampaArray(vett, dim);
            printf("\n\n");
            free(vett);
            freelist(elenco);
            freelist(filtrato);
        }


        getchar();
        getchar();
        return 0;
}
```

Fondamenti di Informatica T-1, 2014/2015 – Modulo 2

Prova d'Esame 1A di Martedì 13 Gennaio 2015 – tempo a disposizione 2h

"articoli.txt":

```
"MAPPLE BOOK PRO 47 pollici" 9599.0 HDD 1TB RAM 128GB schermo 47pollici
"MAPPLE BOOK PRO 46 pollici" 9598.0 HDD 1TB RAM 127GB schermo 46pollici
"MIPHONE v12 123 pollici" 899.0 ssd 128GB RAM 4GB LTE 4g
"MOSCH LAVATRICE PRO" 14599.0 carico 15Kg classeEnergetica A
```