Fondamenti di Informatica T-1, 2011/2012 – Modulo 2

Prova d'Esame 5A di Giovedì 12 Luglio 2012 – tempo a disposizione 2h

Prima di cominciare: si scarichi dal sito **`http://esamix.labx`** il file **`StartKit5A.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"*.

Il ministero dell'ambiente ha disposto sul territorio nazionale molte stazioni automatiche che provvedono a registrare le temperature ad ogni ora, ogni giorno. Tutti i dati rilevati sono spediti ad un computer che li salva su numerosi file di testo, un file per ognuna delle stazioni. In ognuno di questi file per ogni giorno viene scritta una riga con tutte le temperature registrate quel giorno, seguendo tale convenzione: all'inizio della riga viene memorizzato il **giorno** dell'anno a cui si riferisce la riga (un intero, compreso tra 1 e 365), e a seguire, separati da spazi, i **valori** rilevati di temperatura (un insieme di float); ogni riga è sempre terminata da un carattere **';'**. Per svariati motivi, qualche volta vengono persi tutti i dati relativi a un giorno (cioè manca tutta una riga relativa al tal giorno); oppure qualche volta non vengono registrate alcune temperature. Di conseguenza, in ogni file non è dato sapere a priori quante righe siano presenti, e quante misurazioni di temperature siano presenti in ogni riga (il numero varia da riga a riga). Si vedano a titolo di esempio i file di testo forniti nello StartKit.

Esercizio 1 – Struttura dati Misura e funzioni di lettura/stampa (mod. misure.h/misure.c)

Si definisca un'opportuna struttura dati **`Giornaliero`**, al fine di rappresentare i dati relativi alle misurazioni effettuate in un singolo giorno in una stazione. In particolare nella struttura dati si dovrà tener conto del giorno a cui si riferiscono le misure (un intero), e tramite una lista (sottocampo nella struttura dati) si dovranno memorizzare le rilevazioni effettuate (una lista di float).
Si definisca la funzione:

```
Giornaliero leggiSingoloGiorno(FILE* fp);
```

che, ricevuto in ingresso un puntatore a struttura dati FILE (rappresentante un file già aperto in lettura), provveda a leggere da tale file le misurazioni relative ad un solo giorno (cioè i dati presenti in una sola riga). La funzione deve restituire una struttura dati di tipo **`Giornaliero`**, dove devono essere memorizzati i dati letti.

Si definisca la procedura:

```
void stampaSingoloGiorno(Giornaliero m);
```

che, ricevuta in ingresso una struttura dati di tipo **`Giornaliero`**, stampi a video il giorno a cui si riferisce, e l'elenco delle misurazioni relative al tal giorno. Si abbia cura che i valori vengano stampati nello stesso ordine con cui erano memorizzati nel file di origine.
Il candidato abbia cura di realizzare nel main opportuni test al fine di verificare il corretto funzionamento delle funzioni di cui sopra. A tal fine, utilizzi come test uno dei file forniti nello StartKit.

Fondamenti di Informatica T-1, 2011/2012 – Modulo 2

Prova d'Esame 5A di Giovedì 12 Luglio 2012 – tempo a disposizione 2h

Esercizio 2 – Lettura e ordinamento di tutte le misurazioni relative ad una singola stazione automatica (moduli misure.h/misure.c)

Il candidato definisca una funzione:

```
Giornaliero * leggiTuttiGiorni(char * fileName, int * dim);
```

che, ricevuto in ingresso il nome di un file contenente tutte le misurazioni relative ad una singola stazione di rilevamento, ed un intero passato per riferimento, restituisca un vettore allocato dinamicamente (della dimensione minima necessaria) contenente tutte le rilevazioni memorizzate nel file indicato come parametro. Tali informazioni devono essere restituite tramite un array di strutture dati di tipo **`Giornaliero`**; la dimensione dell'array deve essere restituita tramite l'intero passato per riferimento. Al fine di semplificare la lettura il candidato utilizzi la funzione di cui al punto precedente.
Il candidato inoltre abbia cura che l'array sia ordinato secondo il seguente criterio: all'inizio dell'array vanno messe le strutture dati di tipo **`Giornaliero`** con più valori rilevati (cioè i giorni in cui sono state effettuate più rilevazioni di temperatura), e poi a seguire in ordine decrescente le strutture dati con meno valori rilevati. A tal scopo, si suggerisce di implementare una apposita funzione che effettui l'ordinamento richiesto.
Il candidato abbia cura di realizzare nel main opportuni test al fine di verificare il corretto funzionamento delle funzioni di cui sopra. A tal fine, utilizzi come test uno dei file forniti nello StartKit.

Esercizio 3 – Selezione dei giorni che appaiono più frequentemente (modulo misure.h/misure.c)

Al fine di poter confrontare i dati provenienti dalle diverse stazioni di rilevamento, è necessario considerare opportunamente i dati disponibili. Infatti poiché le stazioni di rilevamento certi giorni non registrano dati, può risultare impossibile confrontare i dati di stazioni diverse su un giorno scelto a caso. Sarà quindi necessario scegliere di fare i confronti riguardo ai giorni in cui tali dati siano disponibili.
A tal scopo, si definisca una funzione:

```
int * frequenze();
```

che provveda a determinare, per ogni giorno dell'anno, quante centraline hanno fornito dati per quel giorno. La funzione deve restituire un array di interi allocato dinamicamente, di dimensione pari a 365, che tiene traccia, per ogni giorno dell'anno, di quante centraline hanno registrato temperature in quel giorno (all'inizio dell'esecuzione tale array sarà tutto inizializzato a zero). La funzione deve chiedere all'utente di specificare il nome di un file contente le rilevazioni, e per ogni giorno in cui è stata memorizzata almeno una temperatura, deve incrementare il valore nella cella corrispondente dell'array risultato. Eseguito l'aggiornamento, la funzione deve chiedere all'utente di inserire un nuovo nome di file (ogni nome di file al massimo occupa 23 caratteri). L'utente può segnalare l'intenzione di fermarsi inserendo la parola "fine". In tal caso, la funzione deve stampare a video, per ogni giorno dell'anno, quante centraline hanno fornito dati, cosi ché si possano scegliere i giorni con più dati possibili.
Il candidato abbia una cura particolare nell'evitare fenomeni di memory leaking: a tal scopo quindi de-allochi sempre la memoria allocata dinamicamente.

Esercizio 4 – Stampa delle spese totali sostenute dai clienti, e de-allocazione memoria (main.c)

Il candidato realizzi nella funzione **`main(…)`** un programma che stampi a video per ogni giorno dell'anno (da 1 a 365), quante centraline hanno memorizzato almeno una rilevazione di temperatura. Al termine del programma, il candidato abbia cura di de-allocare tutta la memoria allocata dinamicamente, ivi compresa eventuale memoria allocata per le liste.

Fondamenti di Informatica T-1, 2011/2012 – Modulo 2

Prova d'Esame 5A di Giovedì 12 Luglio 2012 – tempo a disposizione 2h

"element.h":

```
#include <string.h>
#include "list.h"

#ifndef _ELEMENT_H
#define       _ELEMENT_H


typedef float element;



#endif /* _ELEMENT_H */
```

Fondamenti di Informatica T-1, 2011/2012 – Modulo 2

Prova d'Esame 5A di Giovedì 12 Luglio 2012 – 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);

#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);
}
```

Fondamenti di Informatica T-1, 2011/2012 – Modulo 2

Prova d'Esame 5A di Giovedì 12 Luglio 2012 – tempo a disposizione 2h

```
void showlist(list l) {
       element temp;
       if (!empty(l)) {
              temp = head(l);
              printf("%s %s %d\n", temp.cognome, temp.nome, temp.id);
              return showlist(tail(l));
       }
       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, 2011/2012 – Modulo 2

Prova d'Esame 5A di Giovedì 12 Luglio 2012 – tempo a disposizione 2h

"misure.h":

```
#ifndef MISURE_H_
#define MISURE_H_

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

typedef struct {
	int giorno;
	list valori;
} Giornaliero;

Giornaliero leggiSingoloGiorno(FILE* fp);
void stampaSingoloGiorno(Giornaliero m);

Giornaliero * leggiTuttiGiorni(char * fileName, int * dim);

int * frequenze();

#endif /* MISURE_H_ */
```

"misure.c":

```
#include "misure.h"

Giornaliero leggiSingoloGiorno(FILE* fp) {
	Giornaliero result;
	list listRes;
	float temp;
	listRes = emptylist();
	fscanf(fp, "%d", &(result.giorno));
	while (fgetc(fp)!=';') {
		fscanf(fp, "%f", &temp);
		listRes = cons(temp, listRes);
	}
	result.valori = listRes;
	return result;
}

void stampaInOrdine(list l) {
	if (!empty(l)) {
		stampaInOrdine(tail(l));
		printf(" %2.1f", head(l));
	}
}

void stampaSingoloGiorno(Giornaliero m) {
	printf("Giorno: %d, Valori: [", m.giorno);
	stampaInOrdine(m.valori);
	printf("]\n");
	return;
}
```

Fondamenti di Informatica T-1, 2011/2012 – Modulo 2

Prova d'Esame 5A di Giovedì 12 Luglio 2012 – tempo a disposizione 2h

```
int length(list l) {
	if (empty(l))
		return 0;
	else
		return 1 + length(tail(l));
}

int compareMisura(Giornaliero m1, Giornaliero m2) {
	int l1;
	int l2;
	l1 = length(m1.valori);
	l2 = length(m2.valori);
	return l2-l1;
}

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

Giornaliero * leggiTuttiGiorni(char * fileName, int * dim) {
	FILE * fp;
	int count;
	int i;
	char ch;
	Giornaliero * result;
	*dim = 0;
	result = NULL;
	fp = fopen(fileName, "rt");
	if (fp == NULL) {
		printf("Errore nell'apertura del file: %s\n", fileName);
		getc(stdin);
		exit(-1);
	}
	else {
		count = 0;
		do {
			ch = fgetc(fp);
			if (ch==';')
				count++;
		} while (ch!=EOF);
		rewind(fp);
```

```
            result = (Giornaliero*) malloc(sizeof(Giornaliero)*count);
            *dim = count;
            for (i=0; i<count; i++) {
                  result[i] = leggiSingoloGiorno(fp);
            }
            ordina(result, count);
            fclose(fp);
      }


      return result;
}


int * frequenze() {
      int * result;
      char nomeFile[24];
      Giornaliero * m;
      int dim;
      int i;

      result = (int*) malloc(sizeof(int)*365);
      for (i=0; i<365; i++)
            result[i] = 0;

      do {
            printf("Nome file? ");
            scanf("%s", nomeFile);
            if (strcmp("fine", nomeFile)!=0) {
                  m = leggiTuttiGiorni(nomeFile, &dim);
                  for (i=0; i<dim;i++)
                        if (!empty(m[i].valori))
                              result[m[i].giorno-1]++;
                  for(i=0; i<dim; i++)
                        freelist(m[i].valori);
                  free(m);
            }
      } while (strcmp("fine", nomeFile)!=0);
      return result;
}
```

Fondamenti di Informatica T-1, 2011/2012 – Modulo 2

Prova d'Esame 5A di Giovedì 12 Luglio 2012 – tempo a disposizione 2h

"main.c":

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

#include "misure.h"

int main(void) {
	{
		int dim;
		int i;
		Giornaliero * m1;

		m1 = leggiTuttiGiorni("staz15.txt", &dim);
		for (i=0; i<dim; i++) {
			stampaSingoloGiorno(m1[i]);
		}
		for (i=0; i<dim; i++)
			freelist(m1[i].valori);
		free(m1);
	}

	{
		int * freq;
		int i;
		int dim = 365;

		freq = frequenze();
		for (i=0; i<dim; i++)
			printf("Il giorno %d hanno rilevato dato num. %d centraline.\n", i+1,
freq[i]);
		free(freq);
	}
	return 0;
}
```

Fondamenti di Informatica T-1, 2011/2012 – Modulo 2

Prova d'Esame 5A di Giovedì 12 Luglio 2012 – tempo a disposizione 2h

"staz15.txt":

```
23 −5.0 −4.0 −1.0 2.0 2.2 2.4 3.0 2.5 2.0 −3.0;
128 20.0 23.0 28.0 31.0 31.1 25.0 19.0;
129 19.0 20.0 20.0 21.8 22.0 26.0;
364 −3.0 −3.0 −3.0 −2.8 −2.5 −2.1 −1.9 −1.9 −1.9 −3.0 −4.5 −6.7 −9.0 −12.0 −13.0;
```

"staz27.txt":

```
128 20.0 23.0 28.0 31.0 31.1 25.0 19.0;
129 19.0 20.0 20.0 21.8 22.0 26.0;
364 −3.0 −3.0 −3.0 −2.8 −2.5 −2.1 −1.9 −1.9 −1.9 −3.0 −4.5 −6.7 −9.0 −12.0 −13.0;
```