

Fondamenti di Informatica e Laboratorio T-AB
Prova Pratica - 15 Settembre 2009
Compito A

Prima di cominciare: si scarichi il file **StartKit6A.zip** contenente i file di esempio.

Avvertenze per la consegna: nominare i file sorgenti come richiesto nel testo del compito, 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 sorgente** ed i file contenuti nello StartKit.

Rispettare le specifiche, in particolare inserire le funzioni nei file specificati fra parentesi dopo il nome della funzione. Chi non rispetta le specifiche sarà opportunamente penalizzato. **NON SARANNO CORRETTI** gli elaborati che presenteranno un numero "non ragionevole" di errori di compilazione.

Consiglio: per verificare l'assenza di *warnings*, effettuare di tanto in tanto un *Rebuild All*.

Un laboratorio di analisi dei materiali è specializzato nella misura di precisione della resistenza elettrica dei campioni sottoposti ad indagine. L'addetto collega due conduttori elettrici ad ogni campione di materiale, vi invia degli impulsi di corrente e legge su un opportuno display i valori di resistenza elettrica minima e massima offerta dal campione. Quindi annota in un file di testo tali valori. Il file di testo, denominato "**misure.txt**", è così organizzato: su una riga l'impiegato annota il **codice** del campione sotto analisi (al più 8 caratteri alfanumerici, senza spazi); nella riga successiva l'impiegato annota il valore di resistenza minimo **min** (un float); nella riga successiva ancora l'impiegato annota il valore di resistenza massimo **max** (un float). Tale schema di annotazione si ripete per tutti i campioni sottoposti ad analisi. Purtroppo a volte l'impiegato compie degli errori di annotazione, e annota dei valori palesemente sbagliati.

Al fine di elaborare i valori in maniera automatica, è necessario convertire il file di testo contenente le annotazioni in un file binario; inoltre, è necessario scartare le misure palesemente errate e comunicare i codici sospetti all'operatore, così che l'addetto possa eventualmente ripetere le misurazioni.

Esercizio 1 – Lettura di un "campione" e valutazione ([misure.h/misure.c](#))

Si definisca un'opportuna struttura dati **misura**, al fine di rappresentare i dati relativi ad un singolo campione (codice del campione, valore min e valore max). Quindi si realizzi una funzione:

```
misura leggiUnaMisura(FILE * fp);
```

che, ricevuto in ingresso un puntatore ad un file già opportunamente aperto, legga i dati relativi ad un campione e li restituisca come struttura dati **misura**.

Il candidato realizzi poi una funzione:

```
int valuta(misura m);
```

che, ricevuto come parametro di ingresso una struttura dati **m** di tipo **misura**, valuti se la misura è corretta o meno. Una misura è considerata corretta se la differenza tra valore massimo e minimo è minore o uguale al 10% del valore minimo. Ad esempio, dati i valori 30.0 e 27.0, la loro differenza è pari a 3.0; il 10% del valore minimo è pari a 2.7; siccome $3.0 > 2.7$, la misura non è corretta. La funzione deve restituire come risultato un valore intero che possa essere interpretato come "vero" se la misura è corretta, come "falso" altrimenti.

Nello **StartKit6A.zip** il candidato troverà a disposizione un file di testo, "**misure.txt**", contenente una lista di misurazioni di esempio.

Contestualmente, il candidato scriva nel main opportune istruzioni per invocare le funzioni qui definite, al fine di verificarne la corretta implementazione: il candidato abbia cura, dopo aver letto i dati relativi agli studenti, di stampare a video i dati letti al fine di verificarne l'effettiva e corretta corrispondenza con quanto memorizzato in "**misure.txt**". Una volta verificato il corretto funzionamento delle funzioni, il candidato non cancelli il codice nel main ma si limiti a commentarlo.

Fondamenti di Informatica e Laboratorio T-AB
Prova Pratica - 15 Settembre 2009
Compito A

Esercizio 2 – Lettura delle misurazioni corrette/scorrette (measure.h/measure.c)

Il candidato realizzi due funzioni che leggano dal file le misurazioni corrette/scorrette, e copino le strutture dati **misura** in opportune zone di memoria allocate dinamicamente. In particolare, non è noto a priori quante misurazioni siano presenti nel file; si assuma comunque che i codici dei campioni siano tutti diversi tra di loro, e che quindi ogni misurazione sia relativa ad un campione distinto.

A tal fine, il candidato realizzi una funzione:

```
misura * leggiTutteCorrette(FILE * fp, int * dim);
```

che, ricevuto in ingresso un puntatore **fp** a file, determini quante misure corrette vi siano nel file (si usino a tal scopo le funzioni di cui al punto 1), allochi memoria a sufficienza e copi in tale area di memoria le strutture dati ritenute corrette. La funzione restituisca un puntatore all'area di memoria opportunamente allocata, e tramite il parametro **dim** passato per riferimento, restituisca il numero di elementi memorizzati.

A tal fine, il candidato realizzi una funzione:

```
misura * leggiTutteSbagliate(FILE * fp, int * dim);
```

che, ricevuto in ingresso un puntatore **fp** a file, determini quante misure sbagliate vi siano nel file (si usino a tal scopo le funzioni di cui al punto 1), allochi memoria a sufficienza e copi in tale area di memoria le strutture dati ritenute sbagliate. La funzione restituisca un puntatore all'area di memoria opportunamente allocata, e tramite il parametro **dim** passato per riferimento, restituisca il numero di elementi memorizzati.

Esercizio 3 – Main (main.c)

Il candidato realizzi un programma **main.c** che provveda a leggere le misurazioni dal file "**measure.txt**" e a memorizzarle in due distinti array (corrette e sbagliate) tramite le funzioni di cui al punto 2. Quindi il programma stampi a video tutte le misurazioni "sbagliate", e scriva invece su un file binario "**output.dat**" tutte le misurazioni ritenute corrette; all'inizio di tale file si abbia cura di scrivere, come prima cosa, il numero di misurazioni considerate corrette.

Il candidato abbia infine cura di deallocare (al termine del programma) tutte le strutture allocate dinamicamente.

Esercizio 4 – Ordinamento delle misurazioni (measure.h/measure.c)

Si realizzi una procedura che, ricevuto in ingresso un array di strutture dati **misura**, ordini tale array in base al valore **max** di resistenza massima, in ordine crescente.

```
void ordina(misura * m, int dim);
```

A tal scopo, si utilizzi uno qualunque degli algoritmi di ordinamento visti a lezione, avendo cura di modificarlo opportunamente per il caso in questione (se necessario). Quindi si estenda opportunamente la funzione "**main.c**" nel modo seguente: prima di scrivere sul file binario le misurazioni corrette, si abbia cura di ordinare tale vettore tramite la funzione appena definita.

Fondamenti di Informatica e Laboratorio T-AB
Prova Pratica - 15 Settembre 2009
Compito A

"misure.h":

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

```
#ifndef MISURA
#define MISURA
```

```
typedef struct {
    char codice[9];
    float max;
    float min;
} misura;
```

```
#endif
```

```
misura leggiUnaMisura(FILE * fp);
int valuta(misura m);
misura * leggiTutteCorrette(FILE * fp, int * dim);
misura * leggiTutteSbagliate(FILE * fp, int * dim);
void ordina(misura * m, int dim);
```

Fondamenti di Informatica e Laboratorio T-AB
Prova Pratica - 15 Settembre 2009
Compito A

```
"misura.c":
#include "misura.h"

misura leggiUnaMisura(FILE * fp) {
    misura result;
    int ok;
    ok = fscanf(fp, "%s%f%f", result.codice, &(result.min), &(result.max));
    if (ok!=3)
        printf("Errore nella lettura di una misurazione!\n");
    return result;
}

int valuta(misura m) {
    float diff;
    diff = m.max - m.min;
    if (diff<=(m.min/10))
        return 1;
    else return 0;
}

misura * leggiTutteCorrette(FILE * fp, int * dim) {
    int count = 0;
    misura * result;
    misura temp;

    rewind(fp);
    *dim=0;
    count=0;
    while (fscanf(fp, "%s%f%f", temp.codice, &(temp.min), &(temp.max)) == 3) {
        if (valuta(temp))
            count++;
    }
    rewind(fp);
    result = (misura *) malloc(sizeof(misura)*count);
    while ((*dim)<count) {
        result[*dim] = leggiUnaMisura(fp);
        if (valuta(result[*dim]))
            (*dim)++;
    }
    return result;
}

misura * leggiTutteSbagliate(FILE * fp, int * dim) {
    int count = 0;
    misura * result;
    misura temp;

    rewind(fp);
    *dim=0;
    count=0;
    while (fscanf(fp, "%s%f%f", temp.codice, &(temp.min), &(temp.max)) == 3) {
        if (!valuta(temp))
            count++;
    }
    rewind(fp);
    result = (misura *) malloc(sizeof(misura)*count);
    while ((*dim)<count) {
        result[*dim] = leggiUnaMisura(fp);
    }
}
```

Fondamenti di Informatica e Laboratorio T-AB
Prova Pratica - 15 Settembre 2009
Compito A

```
        if (!valuta(result[*dim]))
            (*dim)++;
    }
    return result;
}

void scambia(misura * a, misura * b) {
    misura temp;
    temp = *a;
    *a = *b;
    *b = temp;
}

void bubbleSort(misura v[], int n) {
    int i;
    int ordinato = 0;

    while (n>1 && ordinato==0) {
        ordinato = 1;
        for (i=0; i<n-1; i++) {
            if ( (v[i].max) > (v[i+1].max) ) {
                scambia( &v[i], &v[i+1]);
                ordinato = 0;
            }
        }
        n--;
    }
}

void ordina(misura * m, int dim) {
    bubbleSort(m, dim);
}
```

Fondamenti di Informatica e Laboratorio T-AB
Prova Pratica - 15 Settembre 2009
Compito A

"main.c":

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

#include "misure.h"

void main() {
    FILE * fp, *fp2;
    misura * ok;
    int dim_ok;
    misura * nok;
    int dim_nok;
    int i;

    fp = fopen("misure.txt", "r");
    if (fp==NULL) {
        printf("Errore nell'apertura del file in lettura...\n");
        exit(-1);
    }
    fp2 = fopen("output.dat", "wb");
    if (fp2==NULL) {
        printf("Errore nell'apertura del file in scrittura...\n");
        exit(-1);
    }

    ok = leggiTutteCorrette(fp, &dim_ok);
    nok = leggiTutteSbagliate(fp, &dim_nok);
    fwrite(&dim_ok, sizeof(int), 1, fp2);
    ordina(ok, dim_ok);
    for (i=0; i<dim_ok; i++)
        fwrite(&(ok[i]), sizeof(misura), 1, fp2);
    for (i=0; i<dim_nok; i++)
        printf("Errore: %s %f %f\n", nok[i].codice, nok[i].min, nok[i].max);

    fclose(fp);
    fclose(fp2);
    free(ok);
    free(nok);
    system("PAUSE");
}
```