

Fondamenti di Informatica e Laboratorio T-AB
Prova Pratica - 13 Gennaio 2009
Compito A

Prima di cominciare: si scarichi il file **StartKit2A.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 medico specializzato in dietologia tiene traccia delle informazioni relative ai pazienti tramite due file di testo. In particolare, nel file di nome **pazienti.txt**, il dottore memorizza, per ogni riga, il **cognome** di un paziente (al più 15 caratteri, senza spazi), il **nome** del paziente (al più 15 caratteri, senza spazi), ed un **codice** identificativo unico (un intero positivo). La prima riga del file però contiene solo un intero, rappresentante il numero di pazienti memorizzati nel file stesso.

In un secondo file, di nome **diete.txt**, sono memorizzati i cibi assegnati ad ogni paziente. In particolare, su ogni riga, è presente il **codice** identificativo del paziente (un intero), seguito dal **nome di un cibo** (una stringa di al più 31 caratteri), la **quantità** concessa al paziente (un float, indicante il valore in Kilogrammi), e le **calorie** per 100 grammi di alimento (un intero). Nel file gli alimenti consentiti allo stesso paziente non sono registrati in righe consecutive; si supponga inoltre, per semplicità, che ad ogni paziente siano stati assegnati al massimo 10 alimenti (ma eventualmente anche meno).

Esercizio 1 – Gestione dei pazienti (pazienti.h/pazienti.c)

Il candidato realizzi un modulo che definisca una struttura dati opportuna, di nome **paziente**, per rappresentare i dati relativi ad un paziente, e definisca le funzioni per (a) leggere da file l'elenco dei pazienti; (b) dati il nome ed il cognome di un paziente, restituisca il suo codice identificativo unico.

- a) La funzione di lettura riceve come parametri di ingresso il nome di un file (da cui andare a leggere) e deve restituire un array di pazienti allocato dinamicamente (si abbia cura di allocare solo ed unicamente lo spazio strettamente necessario); inoltre la dimensione del vettore deve essere restituita tramite un ulteriore apposito parametro in ingresso.

Paziente * leggiTutti(char * nomeFile, int * dim);

- b) La funzione di ricerca riceve come parametri di ingresso due stringhe (cognome e nome), e un array di strutture dati Paziente con la sua dimensione, e restituisce il codice identificativo del cliente. Qualora il nominativo cercato non sia presente, la funzione deve restituire il valore -1;

int cerca(char * cognome, char * nome, Paziente * p, int dim);

Contestualmente, il candidato scriva nel main opportune istruzioni per invocare le funzioni definite al fine di verificarne la corretta implementazione: usi a tale scopo il file **pazienti.txt** fornito nello starter kit. Una volta verificato il corretto funzionamento delle funzioni, il candidato non cancelli il codice nel main ma si limiti a commentarlo.

Esercizio 2 – Gestione dei cibi consentiti (pazienti.h/pazienti.c)

Il candidato definisca una opportuna struttura dati **cibo** per rappresentare un alimento e la quantità assegnata, e definisca opportune funzioni per (a) leggere da file tutti gli alimenti assegnati ad un certo paziente (identificato dal suo codice), e (b) stampare a video l'elenco dei cibi (con quantità e calorie) relativi ad un certo paziente.

Fondamenti di Informatica e Laboratorio T-AB
Prova Pratica - 13 Gennaio 2009
Compito A

- a) La funzione di lettura riceve in ingresso il nome del file ed il codice di un paziente, e restituisce un array di strutture di tipo **cibo** con tutti gli alimenti assegnati a quel paziente. Tramite un intero passato per riferimento, la funzione restituisce anche la dimensione di tale array.

Cibo * leggiCibo(char * nomeFile, int codice, int * dim);

- b) La procedura di stampa riceve in ingresso un array di strutture dati **cibo** e la visualizza a video.

void printCibi(Cibo * c, int dim);

Contestualmente, il candidato scriva nel main opportune istruzioni per invocare le funzioni definite al fine di verificarne la corretta implementazione. Una volta verificato il corretto funzionamento delle funzioni, il candidato non cancelli il codice nel main ma si limiti a commentarlo.

Esercizio 3 – Stampa dieta di un singolo paziente (pazienti.h/pazienti.c)

Il candidato definisca una procedura che, ricevuti in ingresso il cognome ed il nome di un paziente, il nome del file contenente la lista dei pazienti, ed il nome del file contenente i dati sugli alimenti, stampi a video tutti gli alimenti (con le relative informazioni) concessi a quel paziente.

void stampaDieta(char * cognome, char * nome, char * file Paz, char * fileCibi);

La procedura deve stampare a video anche l'ammontare totale di calorie: a tal scopo, si noti che le quantità di cibo sono espresse in Kg., mentre le calorie sono relative ad ogni 100 grammi di alimento.

Il candidato abbia cura di deallocare la memoria non più utilizzata. Contestualmente, il candidato scriva nel main opportune istruzioni per invocare le funzioni definite al fine di verificarne la corretta implementazione. Una volta verificato il corretto funzionamento delle funzioni, il candidato non cancelli il codice nel main ma si limiti a commentarlo.

Esercizio 4 – Ordinamento dei clienti (clienti.h/clienti.c)

Si realizzi una procedura che, ricevuto in ingresso un array di strutture dati Paziente, ordini tale array in ordine alfabetico considerando prima il cognome, e poi il nome del paziente.

void ordina(Paziente * p, 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. Si consiglia di realizzare una funzione compare(...) che, ricevuti in ingresso due pazienti, restituisce -1, 0 o 1 a seconda che il primo paziente preceda, eguagli o segua in ordine alfabetico il secondo paziente.

Esercizio 5 – Main (main.c)

Il candidato realizzi un programma **main.c** che chieda all'utente il cognome/nome di un paziente, e stampi a video la dieta di tale paziente (utilizzando le funzioni di cui al punto 1, 2 e 3). Il programma chieda poi all'utente se è interessato alla stampa di tutti i pazienti e, in caso di risposta positiva, stampi la dieta di tutti i pazienti, ordinati per ordine alfabetico (come specificato al punto 4). Il candidato abbia cura di deallocare (al termine del programma) tutte le strutture allocate dinamicamente.

Fondamenti di Informatica e Laboratorio T-AB
Prova Pratica - 13 Gennaio 2009
Compito A

"pazienti.h":

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

typedef struct {
    char cognome[16];
    char nome[16];
    int codice;
} Paziente;

typedef struct {
    char nomeCibo[32];
    float quant;
    int cal;
} Cibo;

Paziente * leggiTutti(char * nomeFile, int * dim);
int cerca(char * cognome, char * nome, Paziente * p, int dim);
Cibo * leggiCibo(char * nomeFile, int codice, int * dim);
void printCibi(Cibo * c, int dim);
void stampaDieta(char * cognome, char * nome, char * filePaz, char * fileCibi);

void ordina(Paziente * p, int dim);
```

Fondamenti di Informatica e Laboratorio T-AB
Prova Pratica - 13 Gennaio 2009
Compito A

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

Paziente * leggiTutti(char * nomeFile, int * dim) {
    Paziente * result = NULL;
    int tempDim;
    FILE * fp;

    *dim = 0;
    if ((fp=fopen(nomeFile, "r")) == NULL)
        return result;

    fscanf(fp, "%d", &tempDim);
    result = (Paziente*) malloc(sizeof(Paziente) * tempDim);
    while ((*dim < tempDim) &&
        fscanf(fp, "%s %s %d",
            result[*dim].cognome, result[*dim].nome, &(result[*dim].codice)) == 3)
        *dim = *dim + 1;
    fclose(fp);
    return result;
}

int cerca(char * cognome, char * nome, Paziente * p, int dim) {
    int i;
    int result = 0;

    for (i=0; i<dim && !result; i++) {
        if (strcmp(cognome, p[i].cognome) == 0 &&
            strcmp(nome, p[i].nome) == 0)
            result = 1;
    }
    if (result)
        return p[i-1].codice;
    else
        return -1;
}

Cibo * leggiCibo(char * nomeFile, int codice, int * dim) {
    Cibo * result = NULL;
    FILE * fp;
    Cibo temp;
    int tempCodice;

    *dim = 0;
    if ((fp=fopen(nomeFile, "r")) == NULL)
        return result;

    result = (Cibo *) malloc (sizeof(Cibo) * 10);

    while (fscanf(fp, "%d %s %f %d",
        &tempCodice, &(temp.nomeCibo), &(temp.quant), &(temp.cal)) == 4) {
        if (codice == tempCodice) {
            result[*dim] = temp;
            *dim = *dim + 1;
        }
    }
    fclose(fp);
    return result;
}

void printCibi(Cibo * c, int dim) {
    int i;
    for (i=0; i<dim; i++)
        printf("%s %f %d\n", c[i].nomeCibo, c[i].quant, c[i].cal);
}
```

Fondamenti di Informatica e Laboratorio T-AB
Prova Pratica - 13 Gennaio 2009
Compito A

```
void stampaDieta(char * cognome, char * nome, char * filePaz, char * fileCibi) {
    Paziente * p;
    int dimP;
    Cibo * c;
    int dimC;
    int codice;
    int i;
    float totale = 0;

    p = leggiTutti(filePaz, &dimP);
    codice = cerca(cognome, nome, p, dimP);
    c = leggiCibo(fileCibi, codice, &dimC);
    printf("Il paziente %s %s ha i seguenti cibi:\n", cognome, nome);
    printCibi(c, dimC);
    for (i=0; i< dimC; i++)
        totale = totale + c[i].quant*10*c[i].cal;
    printf("Totale Calorie: %f\n", totale);
    free(p);
    free(c);
}

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

int compare(Paziente p1, Paziente p2) {
    if (strcmp(p1.cognome, p2.cognome) >0)
        return 1;
    else
        if (strcmp(p1.cognome, p2.cognome) <0)
            return -1;
        else
            return strcmp(p1.nome, p2.nome);
}

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

    while (n>1 && ordinato==0) {
        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(Paziente * p, int dim) {
    bubbleSort(p, dim);
}
```

Fondamenti di Informatica e Laboratorio T-AB
Prova Pratica - 13 Gennaio 2009
Compito A

"main.c":

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

#include "pazienti.h"

int main(void) {
    char cognome[16];
    char nome[16];
    Paziente * p;
    int dimP;
    char answer;
    int i;

    printf("Inserire Cognome e nome: ");
    scanf("%s %s", cognome, nome);
    stampaDieta(cognome, nome, "pazienti.txt", "diete.txt");

    printf("Si vuole stampare la dieta di tutti i pazienti (s/n)?");
    scanf("%*c%c", &answer);
    if (answer == 's') {
        p = leggiTutti("pazienti.txt", &dimP);
        ordina(p, dimP);
        for (i=0; i<dimP; i++) {
            stampaDieta(p[i].cognome, p[i].nome, "pazienti.txt", "diete.txt");
        }
        free(p);
    }

    system("PAUSE");

    return (0);
}
```