

Fondamenti di Informatica T-1, 2010/2011 – Modulo 2

Prova d'Esame 2B di Mercoledì 2 Febbraio 2011 – tempo a disposizione 2h

Prima di cominciare: si scarichi dal sito <http://esamix.labx> il file **StartKit2B.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 1: NON SARANNO CORRETTI gli elaborati che presenteranno un numero "non affrontabile" di errori di compilazione.

Nota 2: 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'azienda che vende online libri pubblicizza i suoi prodotti tramite email inviate direttamente ai clienti. Al fine di limitare i messaggi, invia ai clienti solo le informazioni che potrebbero essere di loro interesse, in base ai loro gusti. A tal scopo, l'azienda mantiene registrato in un file di testo, su ogni riga, una **keyword** che denota un'area di preferenza dell'utente (una stringa di al più 23 caratteri, senza spazi), e, separato da uno spazio, il codice **id** di un cliente (un intero). Ovviamente, un utente può essere interessato a più aree d'interesse: in tal caso il suo **id** compare più volte nel file, in corrispondenza di **keyword** differenti.

In un secondo file di testo l'azienda mantiene un'anagrafe semplificata dei clienti. Sulla prima riga, per comodità, è registrato tramite un intero il numero di righe, cioè di clienti, memorizzati nel file. Nelle righe successive poi, su ogni riga, senza ripetizioni, sono memorizzati l'**id** del cliente; separato da un carattere ','; poi, vi si trova l'**email** assieme ad uno username associato alla email (una stringa di al più 255 caratteri, con spazi); infine, separato ancora da un carattere ';', vi si trova l'età (un intero). Nota: nella struttura dati è necessario memorizzare tutto l'indirizzo, cioè username e indirizzo effettivo.

Nello StartKit sono presenti due file di testo, "pref.txt" e "anag.txt", contenenti rispettivamente le preferenze degli utenti e l'anagrafe, come descritto sopra.

Esercizio 1 – Struttura dati Cliente e funzioni di lettura/scrittura (moduli *element.h* e *cliente.h/cliente.c*)

Si definisca un'opportuna struttura dati **Cliente**, al fine di rappresentare i dati anagrafici relativi a un cliente, come descritto in precedenza.

Si definisca poi la funzione:

```
Cliente * leggiClienti(char* fileName, int * dim);
```

che, ricevuto in ingresso il nome di un file di testo contenente l'anagrafe, restituisca un vettore di strutture dati di tipo **Cliente**, allocato dinamicamente, e "riempito" con i dati presenti sul file specificato come parametro. A tal fine, nello StartKit sono forniti anche i file "readField.h/readField.c", con la funzione readField(...). Si assuma inoltre, per semplicità, che il file di testo sia sempre "ben formato", cioè che tutti i campi previsti siano sempre presenti.

Si definisca anche una procedura:

```
void stampaClienti(Cliente * v, int dim);
```

che, ricevuto in ingresso un vettore di strutture dati di tipo **Cliente**, stampi a video il contenuto di tale array.

Il candidato infine abbia cura di inserire nel **main(...)** alcune istruzioni per "testare" la corretta implementazione delle funzioni, usando come prova il file di esempio fornito nello StartKit, e de-allocando eventuale memoria allocata dinamicamente. Tali istruzioni per il test devono rimanere, commentate, nella funzione main.

Fondamenti di Informatica T-1, 2010/2011 – Modulo 2

Prova d'Esame 2B di Mercoledì 2 Febbraio 2011 – tempo a disposizione 2h

Esercizio 2 – Lettura e scrittura delle preferenze dei clienti (modulo `element.h` e `cliente.h/cliente.c`)

Si definisca un'opportuna struttura dati `Pref`, al fine di rappresentare le preferenze di un cliente, come descritto in precedenza (si rappresenti quindi nella struttura l'`id` del cliente e la `keyword`).

Si realizzi una funzione:

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

che, ricevuto in ingresso il nome di un file contenente le preferenze degli utenti, come descritto in precedenza, legga tali preferenze e le restituisca come una lista di strutture dati di tipo `Pref`.

Si definisca poi una procedura:

```
void stampaPref(list l);
```

che stampa a video le strutture dati di tipo `Pref` contenute nella lista.

Si inseriscano poi nel main apposite istruzioni per verificare la corretta implementazione delle funzioni. Al termine di questo test, si abbia cura di de-allocare la memoria allocata dinamicamente. Si lascino tali istruzioni, eventualmente commentate, nel main.

Esercizio 3 – Ricerca di una preferenza (modulo `element.h` e `cliente.h/cliente.c`)

Si realizzi una funzione:

```
int cerca(list l, int id, char * keyword);
```

che valuti se nella lista `l` è presente una struttura dati di tipo `Pref`, con cliente uguale a quello specificato da `id`, e `keyword` pari a quella passata come parametro. Il risultato restituito deve poter essere interpretabile come vero/falso secondo la convenzione del linguaggio C. A tal scopo, si realizzi una funzione:

```
int isEqual(Pref p1, Pref p2);
```

che restituisca un valore interpretabile come "vero" se le due preferenze `p1` e `p2` sono uguali. Due preferenze sono uguali se riguardano lo stesso cliente (stesso `id`) e la stessa `keyword`.

Esercizio 4 – Stampa dei clienti interessati ad un certo argomento (main.c)

Il candidato realizzi nella funzione `main(...)` un programma che chieda all'utente di specificare una `keyword` riguardo una preferenza, e stampi a video i dati (anagrafici) relativi ai clienti che hanno interesse per l'argomento rappresentato da tale `keyword`, in ordine crescente in base all'età. A tal scopo, dopo aver letto i dati dai file, si allochi dinamicamente un vettore di strutture dati di tipo `Cliente` (la dimensione del vettore non deve essere necessariamente quella minima), e si copino in tale vettore i dati relativi ai clienti che sono interessati all'argomento specificato dall'utente. Si provveda ad ordinare tale vettore in ordine crescente in base all'età, e poi se ne stampi il contenuto a video. Si suggerisce di usare le funzioni sviluppate negli esercizi precedenti.

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, 2010/2011 – Modulo 2

Prova d'Esame 2B di Mercoledì 2 Febbraio 2011 – tempo a disposizione 2h

"element.h":

```
#ifndef _ELEMENT_H
#define    _ELEMENT_H

#define DIM_INDIRIZZO 256
#define DIM_KEYWORD 24

typedef struct {
    int id;
    char email[DIM_INDIRIZZO];
    int eta;
} Cliente;

typedef struct {
    int id;
    char keyword[DIM_KEYWORD];
} Pref;

typedef Pref element;

int isEqual(Pref p1, Pref p2);
int compare(Cliente c1, Cliente c2);

#endif    /* _ELEMENT_H */
```

"element.c":

```
#include "element.h"

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

int compare(Cliente c1, Cliente c2) {
    if (c1.eta>c2.eta)
        return 1;
    else
        if (c1.eta<c2.eta)
            return -1;
        else
            return 0;
}

int isEqual(Pref p1, Pref p2) {
    if (p1.id == p2.id &&
        strcmp(p1.keyword, p2.keyword) == 0)
        return 1;
    else
        return 0;
}
```

Fondamenti di Informatica T-1, 2010/2011 – Modulo 2

Prova d'Esame 2B di Mercoledì 2 Febbraio 2011 – 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);
#endif

"list.c":
#include <stdio.h>
#include <stdlib.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, 2010/2011 – Modulo 2

Prova d'Esame 2B di Mercoledì 2 Febbraio 2011 – tempo a disposizione 2h

"readField.h":

```
#include <stdio.h>

int readField(char buffer[], char sep, FILE *f);
```

"readField.c":

```
#include "readField.h"

int readField(char buffer[], char sep, FILE *f){
    int i = 0;
    char ch = fgetc(f);
    while (ch != sep && ch != 10 && ch != EOF){
        buffer[i] = ch;
        i++;
        ch = fgetc(f);
    }
    buffer[i] = '\0';
    return i;
}
```

Fondamenti di Informatica T-1, 2010/2011 – Modulo 2

Prova d'Esame 2B di Mercoledì 2 Febbraio 2011 – tempo a disposizione 2h

"cliente.h":

```
#ifndef CLIENTE
#define CLIENTE

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

Cliente * leggiClienti(char* fileName, int * dim);
void stampaClienti(Cliente * v, int dim);
list leggiPref(char * fileName);
void stampaPref(list l);
int cerca(list l, int id, char * keyword);

#endif
```

"cliente.c":

```
#include "cliente.h"
Cliente * leggiClienti(char* fileName, int * dim) {
    FILE * fp;
    Cliente * result = NULL;
    int okRead;
    int i = 0;
    *dim = 0;
    fp = fopen(fileName, "rt");
    if (fp == NULL) {
        printf("Errore nell'apertura del file %s\n", fileName);
        exit(-1);
    }
    if (fscanf(fp, "%d", dim)==1) {
        result = (Cliente*) malloc(sizeof(Cliente) * *dim);
        okRead = 1;
        while (okRead && (i < *dim)) {
            okRead = fscanf(fp, "%d", &(result[i].id));
            if (okRead)
                okRead = ( ';' == fgetc(fp));
            if (okRead)
                okRead = (readField(result[i].email, ';', fp) > 0);
            if (okRead)
                okRead = fscanf(fp, "%d", &(result[i].eta));
            i++;
        }
    }
    fclose(fp);
    return result;
}
```

Fondamenti di Informatica T-1, 2010/2011 – Modulo 2

Prova d'Esame 2B di Mercoledì 2 Febbraio 2011 – tempo a disposizione 2h

```
void stampaClienti(Cliente * v, int dim) {
    int i;
    for (i=0; i<dim; i++) {
        printf("%d %s --- %d\n", v[i].id, v[i].email, v[i].eta);
    }
}

list leggiPref(char * fileName) {
    FILE * fp;
    Pref temp;
    list result;

    result = emptylist();
    fp = fopen(fileName, "rt");
    if (fp == NULL) {
        printf("Errore nell'apertura del file %s\n", fileName);
        exit(-1);
    }

    while ( fscanf(fp, "%s %d", temp.keyword, &(temp.id)) == 2)
        result = cons(temp, result);

    fclose(fp);
    return result;
}

void stampaPref(list l) {
    while (!empty(l)) {
        printf("%d %s\n", head(l).id, head(l).keyword);
        l = tail(l);
    }
    return;
}

int cerca(list l, int id, char * keyword) {
    int result = 0;
    Pref temp;

    temp.id = id;
    strcpy(temp.keyword, keyword);
    while (!result && !empty(l)) {
        result = isEqual(temp, head(l));
        l = tail(l);
    }
    return result;
}
```

Fondamenti di Informatica T-1, 2010/2011 – Modulo 2

Prova d'Esame 2B di Mercoledì 2 Febbraio 2011 – tempo a disposizione 2h

"main.c":

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

void bubbleSort(Cliente v[], int n);

int main() {
    /* Test es. 1 */ {
        Cliente * v;
        int dim;
        v = leggiClienti("anag.txt", &dim);
        stampaClienti(v, dim);
        free(v);
    }

    /* Test es. 2 */ {
        list l;
        l = leggiPref("pref.txt");
        stampaPref(l);
        freelist(l);
    }

    /* Test es. 3 */ {
        list l;
        l = leggiPref("pref.txt");
        //stampaPref(l);
        if (cerca(l, 434, "meta-mathematic"))
            printf("Il cliente %s con preferenza %s e' presente in
lista\n", "id0434", "meta-mathematic");
        freelist(l);
    }

    /* Test es. 4 */ {
        Cliente * v;
        Cliente * v_ord;
        int dim, dim_v_ord;
        int i;
        char key[DIM_KEYWORD];
        list l;
        printf("Inserire una keyword: ");
        scanf("%s", key);
        v = leggiClienti("anag.txt", &dim);
        l = leggiPref("pref.txt");
        v_ord = (Cliente *) malloc(sizeof(Cliente) * dim);
        dim_v_ord = 0;
    }
}
```


Fondamenti di Informatica T-1, 2010/2011 – Modulo 2

Prova d'Esame 2B di Mercoledì 2 Febbraio 2011 – tempo a disposizione 2h

```
        for (i=0; i<dim; i++)
            if (cerca(l, v[i].id, key)) {
                v_ord[dim_v_ord] = v[i];
                dim_v_ord++;
            }
        bubbleSort(v_ord, dim_v_ord);
        for (i=0; i<dim_v_ord; i++)
            printf("%d %s %d\n", v_ord[i].id, v_ord[i].email,
v_ord[i].eta);
        free(v);
        free(v_ord);
        freelist(l);
    }

    system("pause");
    return 0;
}
void scambia(Cliente *a, Cliente *b) {
    Cliente tmp = *a;
    *a = *b;
    *b = tmp;
}
void bubbleSort(Cliente 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--;
    }
}
"anag.txt":
3
434;"Mario Bianchi" <mario.bianchi@mail.it>;25
654;"Paolo Rossi" <paolo.rossi@mailgratis.it>;27
745;"Gianni Verdi" <gianni.verdi@mailfree.it>;31
"pref.txt":
thriller 434
romance 654
history 654
history 745
physics 654
romance 434
meta-mathematic 434
informatics 654
science 654
informatics 745
meta-mathematic 745
```