

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

Prova d'Esame 1A di Mercoledì 19 Gennaio 2011 – tempo a disposizione 2h

Prima di cominciare: si scarichi dal sito <http://esamix.labx> il file **StartKit1A.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*".

Uno studente vuole realizzare un sito per la compravendita di materiale didattico usato (libri, fotocopie di appunti, testi d'esame, etc.). A tal scopo deve realizzare un programma che permetta la memorizzazione del materiale in suo possesso, e la ricerca di tale materiale. Su un file di testo (denominato "materiale.txt") vengono salvate le informazioni relative ad ogni oggetto in vendita, un oggetto su ogni riga. Per ogni oggetto sono memorizzate le seguenti informazioni: una **descrizione** dell'oggetto (una stringa di al più 60 caratteri, senza spazi), il **costo** di tale oggetto (un float), ed il numero di **copie** presenti (un intero). Si veda a tal scopo il file di esempio fornito nello StartKit.

Esercizio 1 – Struttura dati Oggetto e funzioni di lettura/scrittura (moduli `element.h` e `oggetto.h/oggetto.c`)

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

Si definisca poi la funzione:

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

che, ricevuto in ingresso il nome di un file di testo contenente strutture dati di tipo **Oggetto** (come specificato in precedenza), restituisca una lista delle strutture dati contenute nel file.

Si definisca anche una procedura:

```
void stampaOggetti(list l);
```

che, ricevuto in ingresso una lista di strutture dati di tipo **Oggetto**, stampi a video il contenuto di tale lista.

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.

Esercizio 2 – Ricerca degli oggetti disponibili in base alla loro descrizione (modulo `oggetto.h/oggetto.c`)

Si realizzi una funzione:

```
list cercaOggetti(list l, char desc[]);
```

che, ricevuto in ingresso una lista di strutture dati di tipo **Oggetto** (come definita all'esercizio 1), ed una stringa **desc** contenente una stringa ben formata, restituisca una nuova lista di strutture dati **Oggetto**, che contenga solo quegli oggetti che rispondono contemporaneamente ad entrambi i seguenti criteri: (1) la stringa **desc** deve essere contenuta (in parte o totalmente) nella descrizione dell'oggetto; (2) l'oggetto deve essere disponibile (cioè il numero di copie deve essere maggiore di zero). Al fine di verificare se una stringa è contenuta in un'altra, si suggerisce l'uso della funzione `char * strstr(char * cs, char * ct)` definita in "`string.h`". Tale funzione restituisce un puntatore alla prima occorrenza di **ct** in **cs**, o **NULL** se **ct** non compare in **cs**. Il candidato verifichi la corretta implementazione della funzione con apposite istruzioni inserite nel **main(...)**.

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

Prova d'Esame 1A di Mercoledì 19 Gennaio 2011 – tempo a disposizione 2h

Esercizio 3 – Ordinamento degli oggetti in base al prezzo (modulo oggetto.h/oggetto.c)

Si realizzi una funzione:

```
Oggetto * piuEconomici(list l, int * dim);
```

che, ricevuto in ingresso una lista di strutture dati di tipo `Oggetto`, allochi dinamicamente un vettore di strutture dati di tipo `Oggetto`, e copi tali strutture dalla lista al vettore. La dimensione del vettore deve essere restituita tramite il parametro `dim` passato per riferimento, e la funzione deve restituire il vettore allocato dinamicamente. Tale vettore deve essere ordinato in base al prezzo degli oggetti, in ordine crescente (gli oggetti più economici all'inizio del vettore); a parità di prezzo, gli oggetti devono essere ordinati in senso alfabetico in base alla `descrizione`. A tal scopo, il candidato usi una qualunque funzione di ordinamento vista a lezione, o anche direttamente una funzione di inserimento ordinato nel vettore. Si abbia cura di verificare il corretto funzionamento della funzione implementata nel `main`.

Esercizio 4 – Selezione e scrittura su file binario dei risultati di una ricerca (main.c)

Il candidato realizzi nella funzione `main (...)` un programma che chieda all'utente di specificare parte di una stringa da cercare, legga opportunamente dal file contenente l'elenco degli oggetti i dati contenuti, selezioni gli oggetti disponibili che sono relativi a quanto specificato dall'utente, li ordini (in maniera crescente) in base al prezzo/descrizione, e stampi a video tali oggetti. Si usino a tal fine tutte le funzioni sviluppate negli esercizi precedenti.

Inoltre, il programma salvi in un file binario il vettore così ottenuto: il nome del file deve essere richiesto esplicitamente all'utente, ed il programma deve avere cura di verificare l'esistenza o meno di un file con tale nome. Qualora il file esista già, il programma deve continuare a chiedere all'utente un nuovo nome di file dove salvare il vettore di strutture dati `Oggetto`. Si suggerisce per verificare l'esistenza o meno di un file di provare ad aprirlo in lettura: se l'apertura ha successo, tale file esiste già (ricordarsi quindi di chiuderlo). Se l'apertura fallisce, tale file non esiste ancora.

Al termine del programma, il candidato abbia cura di de-allocare tutta la memoria allocata dinamicamente, ivi compresa la memoria allocata per le liste.

Al fine di verificare la corretta implementazione delle funzioni, si suggerisce di usare il file di esempio contenuto nello StartKit, e di eseguire una ricerca con la stringa "Fondamenti". L'output risultante dovrebbe essere qualcosa del tipo:

```
Inserire la stringa da cercare: Fondamenti
FondamentiDiInformaticaT2 7.000000 7
FondamentiDiInformaticaT1 12.000000 5
FondamentiDiTermodinamicaApplicata 38.000000 2
```

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

Prova d'Esame 1A di Mercoledì 19 Gennaio 2011 – tempo a disposizione 2h

"element.h":

```
#ifndef _ELEMENT_H
#define _ELEMENT_H

#define DIM_DESC 35

typedef struct {
    char descrizione[DIM_DESC];
    float costo;
    int copie;
} Oggetto;

typedef Oggetto element;

int compare(element e1, element e2);

#endif /* _ELEMENT_H */
```

"element.c":

```
#include "element.h"

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

int compare(element e1, element e2) {
    if (e1.costo > e2.costo)
        return 1;
    else
        if (e1.costo < e2.costo)
            return -1;
        else
            return strcmp(e1.descrizione, e2.descrizione);
}
```

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

Prova d'Esame 1A di Mercoledì 19 Gennaio 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 */
```

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

Prova d'Esame 1A di Mercoledì 19 Gennaio 2011 – tempo a disposizione 2h

```
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 1A di Mercoledì 19 Gennaio 2011 – tempo a disposizione 2h

"oggetto.h":

```
#ifndef OGGETTO
#define OGGETTO

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

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

list leggiOggetti(char* fileName);
void stampaOggetti(list l);
list cercaOggetti(list l, char desc[]);
Oggetto * piuEconomici(list l, int * dim);

#endif
```

"oggetto.c":

```
#include "oggetto.h"
list leggiOggetti(char* fileName) {
    FILE * fp;
    list result;
    Oggetto temp;

    if ((fp=fopen(fileName, "rt")) == NULL) {
        printf("Errore nell'aprire il file %s\n", fileName);
        system("PAUSE");
        exit(-1);
    }
    result = emptylist();
    while (fscanf(fp, "%s %f %d", temp.descrizione, &temp.costo, &temp.copie)
== 3)
        result = cons(temp, result);

    fclose(fp);
    return result;
}

void stampaOggetti(list l) {
    Oggetto temp;
    while (!empty(l)) {
        temp = head(l);
        printf("%s %6.2f %d\n", temp.descrizione, temp.costo, temp.copie);
    }
}
```

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

Prova d'Esame 1A di Mercoledì 19 Gennaio 2011 – tempo a disposizione 2h

```
        l = tail(l);
    }
}

list cercaOggetti(list l, char desc[]) {
    list result;
    Oggetto temp;

    result = emptylist();
    while (!empty(l)) {
        temp = head(l);
        if (strstr(temp.descrizione, desc)!=NULL && temp.copie>0)
            result = cons(temp, result);
        l = tail(l);
    }
    return result; }

void myInsOrd(Oggetto v[], int dimLogica, int dimFisica, Oggetto x){
    int i = dimLogica-1;
    if (dimLogica<dimFisica) {
        if (dimLogica == 0)
            v[0] = x;
        else {
            while (i>=0 && compare(x,v[i])<0) {
                v[i+1] = v[i];
                i--;
            }
            v[i+1] = x;
        }
    }
}

Oggetto * piuEconomici(list l, int * dim) {
    Oggetto * result;
    list temp;
    int dimLogica = 0;
    temp = l;
    *dim = 0;
    while (!empty(temp)) {
        (*dim)++;
        temp = tail(temp);
    }
    result = (Oggetto * ) malloc(sizeof(Oggetto) * *dim);
    while (!empty(l)) {
        myInsOrd(result, dimLogica, *dim, head(l));
        dimLogica++;
        l = tail(l);
    }
    return result; }
```

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

Prova d'Esame 1A di Mercoledì 19 Gennaio 2011 – tempo a disposizione 2h

"main.c":

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

int main() {
    /* TEST Es. 1*/
    {
        list l;
        l = leggiOggetti("materiale.txt");
        stampaOggetti(l);
        printf("\n\n");
        freelist(l);
    }
    /* TEST Es. 2*/
    {
        list l;
        list l2;
        l = leggiOggetti("materiale.txt");
        l2 = cercaOggetti(l, "Fondamenti");
        stampaOggetti(l2);
        printf("\n\n");
        freelist(l);
        freelist(l2);
    }
    /* TEST Es. 3 */
    {
        list l;
        list l2;
        Oggetto * v;
        int dim_v;
        int i;
        l = leggiOggetti("materiale.txt");
        l2 = cercaOggetti(l, "Fondamenti");
        v = piuEconomici(l2, &dim_v);
        for (i=0; i<dim_v; i++)
            printf("%s %f %d\n", v[i].descrizione, v[i].costo, v[i].copie);
        printf("\n\n");
        freelist(l);
        freelist(l2);
        free(v);
    }
    /* TEST Es. 4 */
    {
        list l;
    }
```

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

Prova d'Esame 1A di Mercoledì 19 Gennaio 2011 – tempo a disposizione 2h

```
list l2;
Oggetto * v;
int dim_v;
int i;
int success;
char desc[DIM_DESC];
char fileName[64];
FILE * fp;

printf("Inserire la stringa da cercare: ");
scanf("%s", desc);
l = leggiOggetti("materiale.txt");
l2 = cercaOggetti(l, desc);
v = piuEconomici(l2, &dim_v);
for (i=0; i<dim_v; i++)
    printf("%s %f %d\n", v[i].descrizione, v[i].costo, v[i].copie);
printf("\n\n");
success = 0;
do {
    printf("Inserire il nome del file binario: ");
    scanf("%s", fileName);
    if ((fp = fopen(fileName, "rb")) == NULL) {
        if ((fp = fopen(fileName, "wb")) == NULL) {
            printf("Errore nell'apertura del file %s\n",
fileName);

            system("pause");
            exit(-2);
        }
        fwrite(v, sizeof(Oggetto), dim_v, fp);
        fclose(fp);
        success = 1;
    }
    else printf("Un file di nome %s esiste gia'.\n", fileName);
} while (!success);
freelist(l);
freelist(l2);
free(v);
}
system("pause");
return 0; }
```

“materiale.txt”:

```
FondamentiDiInformaticaT1 12.00 5
FondamentiDiInformaticaT2 7.00 7
AnalisiMatematica1 20.00 12
AnalisiMatematica2 25.00 13
AnalisiMatematica3 28.00 3
FondamentiDiTermodinamicaApplicata 38.00 2
FondamentiDiTermodinamica 35.00 0
```