Università degli Studi di Ferrara

DE: Department of Engineering Ferrara

Esercitazioni di

FONDAMENTI DI INFORMATICA MODULO B

ESERCITAZIONE 5

Azzolini Damiano: damiano.azzolini@unife.it
Bertagnon Alessandro: alessandro.bertagnon@unife.it

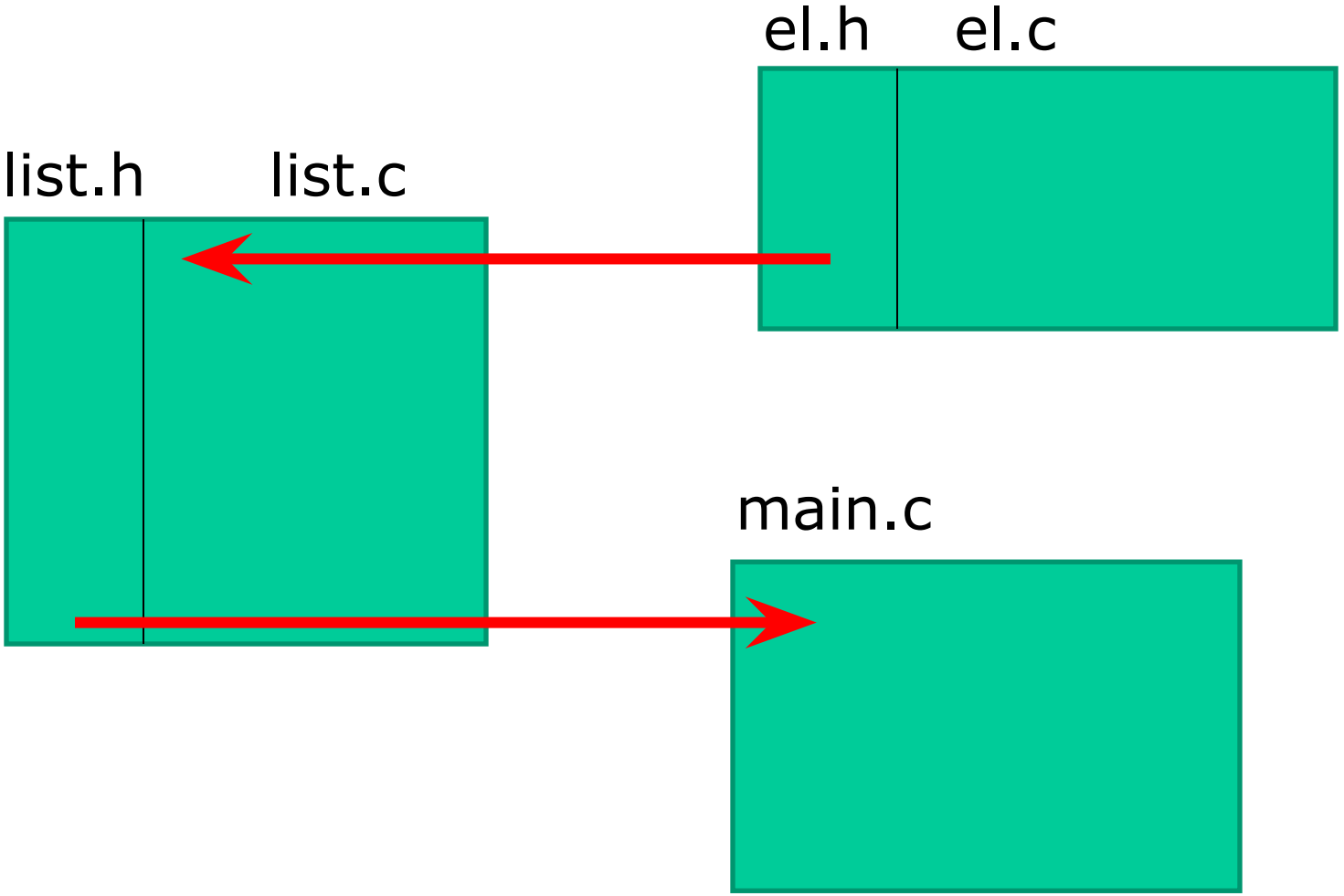
Esercizio 1. Liste di interi

Si legga una sequenza di numeri interi da tastiera (terminata da 0).

- Creare due liste L1 e L2 con **inserimento ordinato**. L'n-esimo elemento letto è inserito in L1 se n è dispari, in L2 se n è pari. In altre parole, se vengono letti gli elementi 2,4,5,3,9, L1 conterrà [2,5,9] ed L2 conterrà [4,3].
- Creare una funzione **append** che, date due liste, restituisca una terza lista contenente gli elementi della prima seguiti dagli elementi della seconda: **list append(list, list)**
- Creare una funzione **merge** che, date due liste ordinate, restituisca una lista **ordinata** contenente gli elementi delle due liste date (gli elementi possono essere ripetuti): **list merge(list, list)**

Si astragga il tipo di elemento delle liste in modo da rendere l'implementazione il più strutturata possibile.

Esercizio 1. Struttura



Esercizio 1. Interfaccia modulo elementi: el.h

```
/* ELEMENT TYPE - file el.h*/
#ifndef ELEMENT_H
#define ELEMENT_H

typedef int element;
typedef enum {false, true} boolean;

boolean isLess(element e1, element e2);
boolean isEqual(element e1, element e2);
element getElement(void);
void printElement(element e);

#endif
```

Esercizio 1. Liste di interi

Dove:

- **boolean isLess(element e1, element e2);**

Controlla se e1 < e2

- **boolean isEqual(element e1, element e2);**

Controlla se e1 == e2

- **element getElement(void);**

Legge in input da tastiera un elemento

- **void printElement(element e);**

Stampa a video un elemento

Esercizio 2. Liste di strutture

Si vuole tenere in memoria le informazioni riguardanti gli studenti iscritti ad un corso di informatica.

Per fare ciò si cominci a implementare una libreria (**studente.h**, **studente.c**) contenente la definizione della struttura **studente** e le funzioni principali per la sua manipolazione.

```
#define DIM_STR 20
typedef struct {
   char nome[DIM_STR];
   char cognome[DIM_STR];
   float media;
} element;
```

Esercizio 2. Liste di strutture

Creare le funzioni:

- **boolean isLess(element e1, element e2);**
  Controlla se un elemento è minore dell'altro
- **boolean isEqual(element e1, element e2);**
  Controlla uguaglianza fra due elementi
- **element getElement(void);**
  Legge in input da tastiera un elemento
- **void printElement(element e);**
  Stampa a video un elemento

Usare queste librerie per astrarre il tipo di elemento delle liste dall'implementazione.

Esercizio 2. Liste di strutture

Il programma deve leggere da un file **studenti1.txt** un numero N di studenti (N definito dall'utente in input) inserendo gli studenti presenti nel file nella lista **l1** e leggere da **studenti2.txt** un numero M di studenti inserendoli nella lista **l2**.
NB: in **l1** e **l2** effettuare l'inserimento **ordinato** in base al cognome (e al nome in caso di cognomi uguali).

Sviluppare un menu con le seguenti opzioni:

- Visualizzazione di tutti gli studenti.
- Ricerca di uno studente (per nome e cognome).
- Cancellazione di uno studente da una lista.
- Costruzione delle sottoliste di l1 e l2 a partire dall'indice i letto da tastiera.
- Append delle due liste.
- Copia di una delle due liste in una terza.
- Merge di due liste.
- Stampa del numero degli studenti nelle due liste.

Esercizio 2. Liste di strutture

Le funzioni richieste di manipolazione delle liste, da inserire nel file **list.c**, **list.h** secondo le solite modalità, sono state già introdotte a lezione. Di seguito un rapido ripasso:

```
typedef enum { false, true } boolean;
typedef struct node{
    element n;
    struct node *next;
} listNode;
typedef listNode* list;
```

Esercizio 2. Liste di strutture

```
list cons(element n, list l); // inserimento in testa (da usare nella insord)
list insord(element n, list l); // inserimento ordinato
void showlist(list l); // stampa della lista l
list copy(list l); // copia di l in un'altra lista  (diversa da l)
boolean member(element el, list l); // ricerca di el in l
int length(list l); // numero di elementi in l
list sublist(list l, int i); // estrae la sottolista a partire dall'elemento di indice i
list append (list l1, list l2); // append di l2 in coda a l1
list merge (list l1, list l2); // unione di due liste ordinate in una terza ordinata anch'essa
list deleteValue(element n, list l); // cancellazione di n da l
```

Tutte le funzioni dovranno essere implementate in maniera **RICORSIVA**.

Esercizio 2. Liste di strutture

Suggerimento: per verificare di aver copiato correttamente una lista, una volta eseguita la copia fare un inserimento o una cancellazione in una delle due liste (originale o copia) e controllare che la lunghezza delle due liste sia diversa.

Esercizio 2. Liste di strutture

Ricorsione, cosa gestire? Un esempio:

```
void recFunct(list l){
    if (l == NULL)
        \\ CASO BASE -> FINE DELLA RICORSIONE
    else {
        \\ GESTIONE DEGLI ALTRI CASI
        \\ CHIAMATA RICORSIVA ALLA FUNZIONE
        recFunct(l -> next);
    }
}
```

Esercizio 2. Liste di strutture

TIPS: valutare come eseguire la ricorsione è molto importante per evitare di complicare troppo le funzioni.
A volte conviene creare delle sotto funzioni che eseguono il calcolo ricorsivo (necessario soprattutto se le interfacce date non possono essere modificate). Per esempio:

```
float recFunct(list l){
    return recFunctInterna(l) * 5;
}

float recFunctInterna(list l) {
    if (l == NULL)
        \\ CASO BASE
    else {
        \\ GESTIONE DEGLI ALTRI CASI + CHIAMATA RICORSIVA
        recFunctInterna(l -> next);
    }
}
```

Testi d'esame su liste

Esame 22 Luglio 2010:
http://www.unife.it/ing/informazione/fond-info-modulo-b/esame/prove-desame/prove-c-e-teoria-parteb/22-luglio-2010
(file di input nella cartella della soluzione)

Esame 26 Luglio 2011:
http://www.unife.it/ing/informazione/fond-info-modulo-b/esame/prove-desame/prove-c-e-teoria-parteb/CorrezioneEsame2011-26Luglio.zip/view