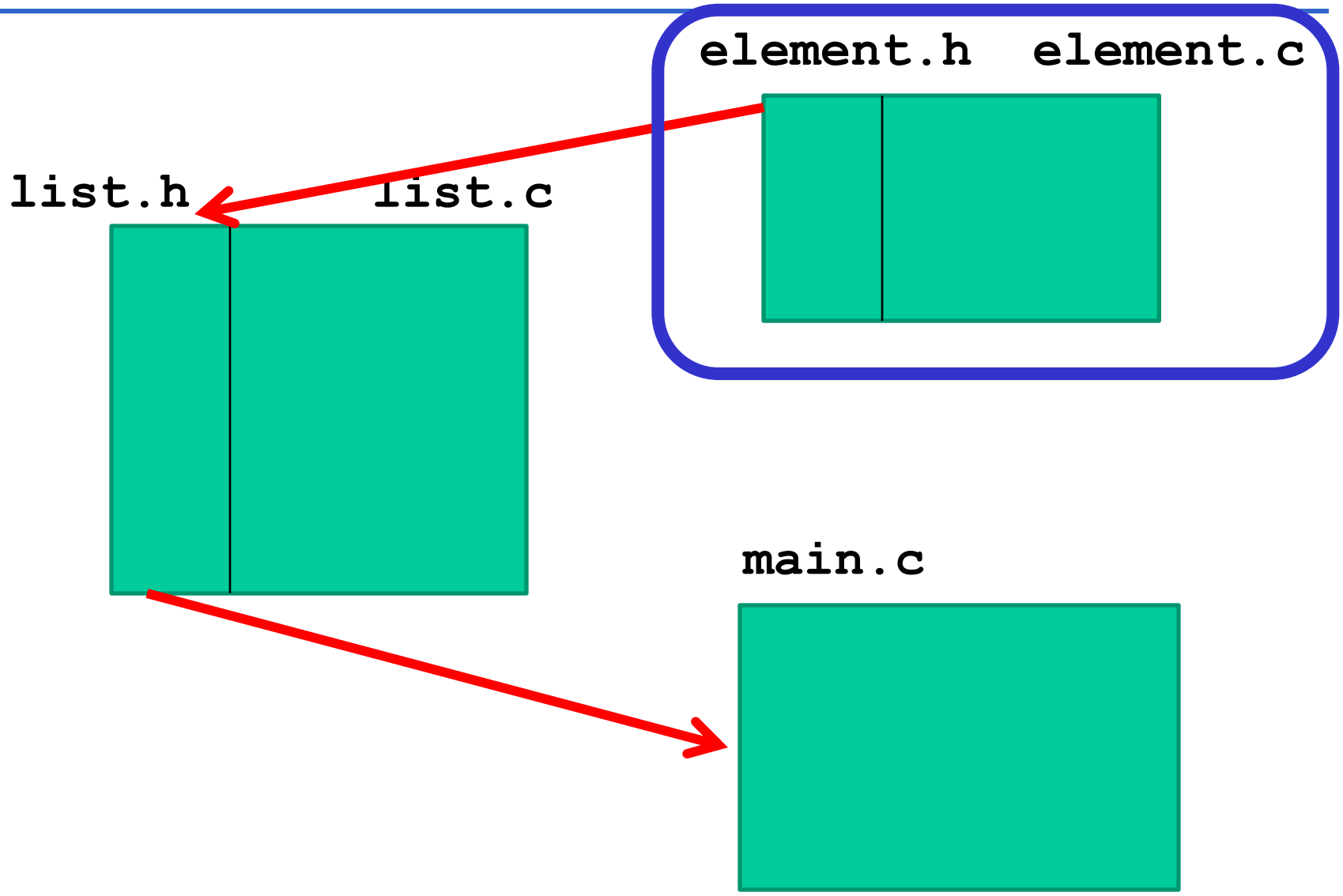
Liste di elementi strutture

Obiettivi:

- Vedere cosa cambia rispetto a lista di scalari
- Vantaggi dall' adozione di progetto su più file
- Esempi da prove d'esame

SVILUPPO PER COMPONENTI



PROVA INTERMEDIA 20 Aprile 2017

In un file binario **alimenti.bin** sono scritti i valori calorici (per 100 grammi di prodotto) dei prodotti alimentari. Per ciascun prodotto, il file **alimenti.bin** contiene

- il nome del prodotto (stringa di 50 char),
- e il valore dell'energia (intero, in Kilo-calorie).

Ad esempio, per il prodotto cracker si ha:

```
"cracker" 439
```

perché i cracker hanno, per 100 grammi di prodotto, 439 kcal.

PROVA INTERMEDIA 20 Aprile 2017

Si realizzi un programma C, **organizzato in** almeno **tre funzioni**, rispettivamente dedicate a:

- a partire dal file **alimenti.bin**, creare **una lista L** in memoria centrale che contiene i dati dei prodotti, **ordinata in base al nome del prodotto**; la **funzioneA** riceve come parametri:
  - il puntatore al file,
  - il puntatore a L (inizializzata a NULL nel main),

più eventuali parametri a scelta, e restituisce il puntatore alla radice della lista L;

PROVA INTERMEDIA 20 Aprile 2017

- stampare la lista L a video; la **funzioneB** riceve come parametri:
  - il puntatore a L,

più eventuali parametri a scelta, e restituisce void ;

PROVA INTERMEDIA 20 Aprile 2017

- a partire dalla lista L creata, determinare **quanti e quali alimenti sono iper-calorici** (ovvero hanno più di 300 calorie per 100 grammi di prodotto), stampando il nome di ciascun alimento e le sue calorie per 100 grammi di prodotto e il numero totale di alimenti iper-calorici su un **file di uscita di tipo testo output.txt** da consegnare con i codici sorgente; la **funzioneC** riceve come parametri
  - il puntatore al secondo file, aperto nel main
  - il puntatore a L,

  più eventuali parametri a scelta, e restituisce void.

CONSIDERAZIONI

Leggete bene il testo, e comprendete cosa vi viene chiesto

Pensate se qualche **funzionalità** può essere **riutilizzata** …

Domanda a)

Il programma deve chiamare tre funzioni (da definire) dedicate rispettivamente a:

a) Caricare in una lista L in memoria centrale i record del file

Inserimento ordinato in lista con ordinamento in base al nome del prodotto (stringa)

Possiamo usare la ***strcmp*** di string.h

Domanda a), come fareste?

Definire il tipo del record (struct, ***element***)

Definire una funzione ***isLess*** che dati ***due element*** restituisce:

- vero (1), se il primo (il suo nome) precede o è uguale al nome del secondo
- falso (0), viceversa

Definire il tipo **list** (con elementi di tipo ***element***)

Definire la lista ***L***

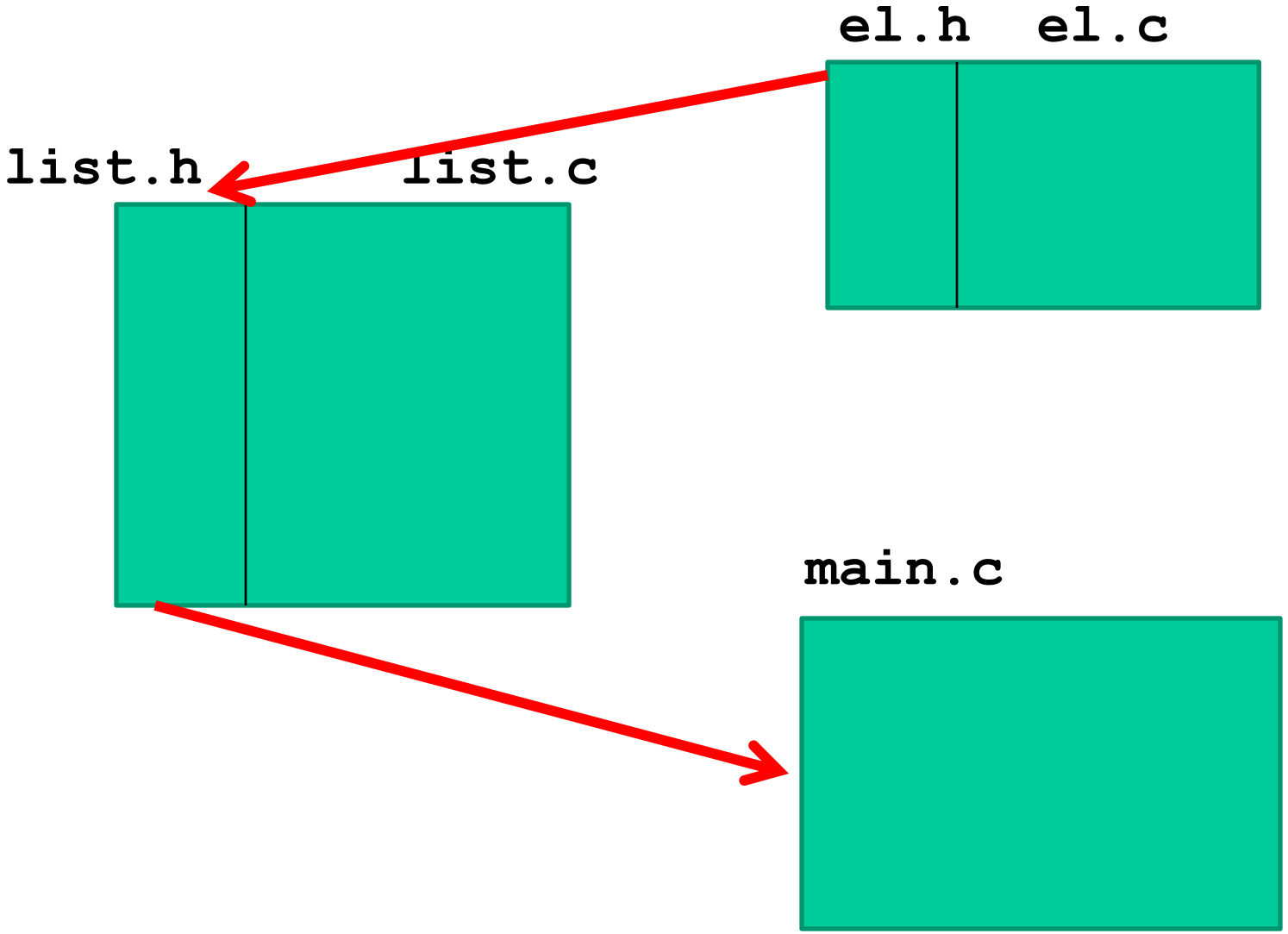
Domanda a), come fareste?

Ciclo di lettura da file (fino alla file del file), leggendo

1 record per volta

Inserire l'elemento letto in lista L con ***insord***, che usa la funzione ***isLess***

Prova 20 Aprile 2017 - COMPONENTI



el.h

```
#define DIM 50

typedef struct {
     char nome[DIM];
     int val; } element;

int isLess(element a, element b);
void printElement(element x);
```

el.c

```
#include <stdio.h>
#include <string.h>
#include "el.h"

int isLess(element a, element b) {
// return 1 se minore o uguale
    if (strcmp(a.nome,b.nome)>0)
            return 0;
    else    return 1;
}

void printElement(element x) {
    printf("%s, %d\n", x.nome, x.val);
}
```

list.h

```
#include "el.h"

typedef struct list_element {
    element value;
    struct list_element *next;
  } item;

typedef item *list;

list cons(element el, list l);
list insord(element el, list l);
void showList(list l);
```

list.c

```
#include <stdlib.h>
#include "list.h"

list cons(element el, list l) {
    list aux = (item*)malloc(sizeof(item));
    aux->value = el;
    aux->next = l;
    return aux;       }

list insord(element el, list l) {
    if (l == NULL)  return cons(el, l);
    else { if (isLess(el, l->value))
                    return cons(el, l);
           else {l->next =insord(el,l->next);
                  return l;  }   }
}
```

list.c

```
#include <stdlib.h>
#include "list.h"

list cons(element el, list l) {
    list aux = (item*)malloc(sizeof(item));
    aux->value = el;
    aux->next = l;
    return aux;       }

list insord(element el, list l) {
 if (l == NULL)  return cons(el, l);
 else {if (strcmp(el.nome,l->value.nome)<=0)
                        return cons(el, l);
       else { l->next = insord(el, l->next);
              return l;  }   }
}
```

main.c

```
#include <stdio.h>
#include <stdlib.h>
#include "list.h"
#define IPER 300

list funzioneA(FILE *f,list L);
void funzioneC(FILE *output,list L);

main() {
     list L = NULL;
     FILE *f, *output;
     f = fopen("alimenti.bin", "rb");
     if (f == NULL) {
          printf("Non aperto.\n");
          exit(-1);    }
     L = funzioneA(f, L);
     fclose(f);
     showlist(L);        /* funzioneB */
```

list.c

```
void showList(list l) {
// versione iterativa
    while (l != NULL) {
        printElement(l->value);
        l=l->next;
    }
}
```

list.c

```
void showListR(list l) {
// versione ricorsiva
     if (l != NULL) {
          printElement(l->value);
          showlistR(l->next);
     }
}
```

main.c (*continua*)

```
  showlist(L);      /* funzioneB */

  output = fopen("output.txt", "wt");
  if (output == NULL) {
        printf("File non creato.\n");
        exit(-1);
     }

  funzioneC(output, L);

  fclose(output);

}
```

funzioneA

```
list funzioneA(FILE *f, list L) {
  element el;
  while (fread(&el,sizeof(element),1,f)>0) {
        L = insord(el, L);
    }
    return L;
}
```

Domanda c)

• Scandire in contenuto della lista ***L***, scrivendo in uscita sul file di uscita

La funzione del punto c) riceve il puntatore al file di uscita (preventivamente aperto in scrittura), e la lista (ed eventuali altri parametri a vostra scelta).

Deve scorrere la lista, stampare su file un elemento se è iper-claorico, e tenere il conto ...

E' una variante della solita **showList** ... con un conteggio in più

funzioneC

```
void funzioneC(FILE *f, list L) {
    int tot=0;
    while (L != NULL) {
        if (L->value.val > IPER) {
fprintf(f,"%s %d\n",L->value.nome,L->value.val);
            tot++;   }
         L=L->next;
     }

fprintf(f,"N. alimenti ipercalorici: %d\n",tot);

}
```

COMPONENTI