Liste di elementi strutture

Obiettivi:

- Esempio da prova d'esame
- Vantaggi da sviluppo per componenti
- *Considerazioni finali su liste e ADT list*

PROVA INTERMEDIA 26 Aprile 2018

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.

I prodotti con molte calorie, detti iper-calorici, hanno più di 300 calorie per 100 grammi di prodotto. Ad esempio, il prodotto cracker è iper-calorico.

PROVA INTERMEDIA 26 Aprile 2018

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 iper-calorici, **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 26 Aprile 2018

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

più eventuali parametri a scelta, e restituisce void ;

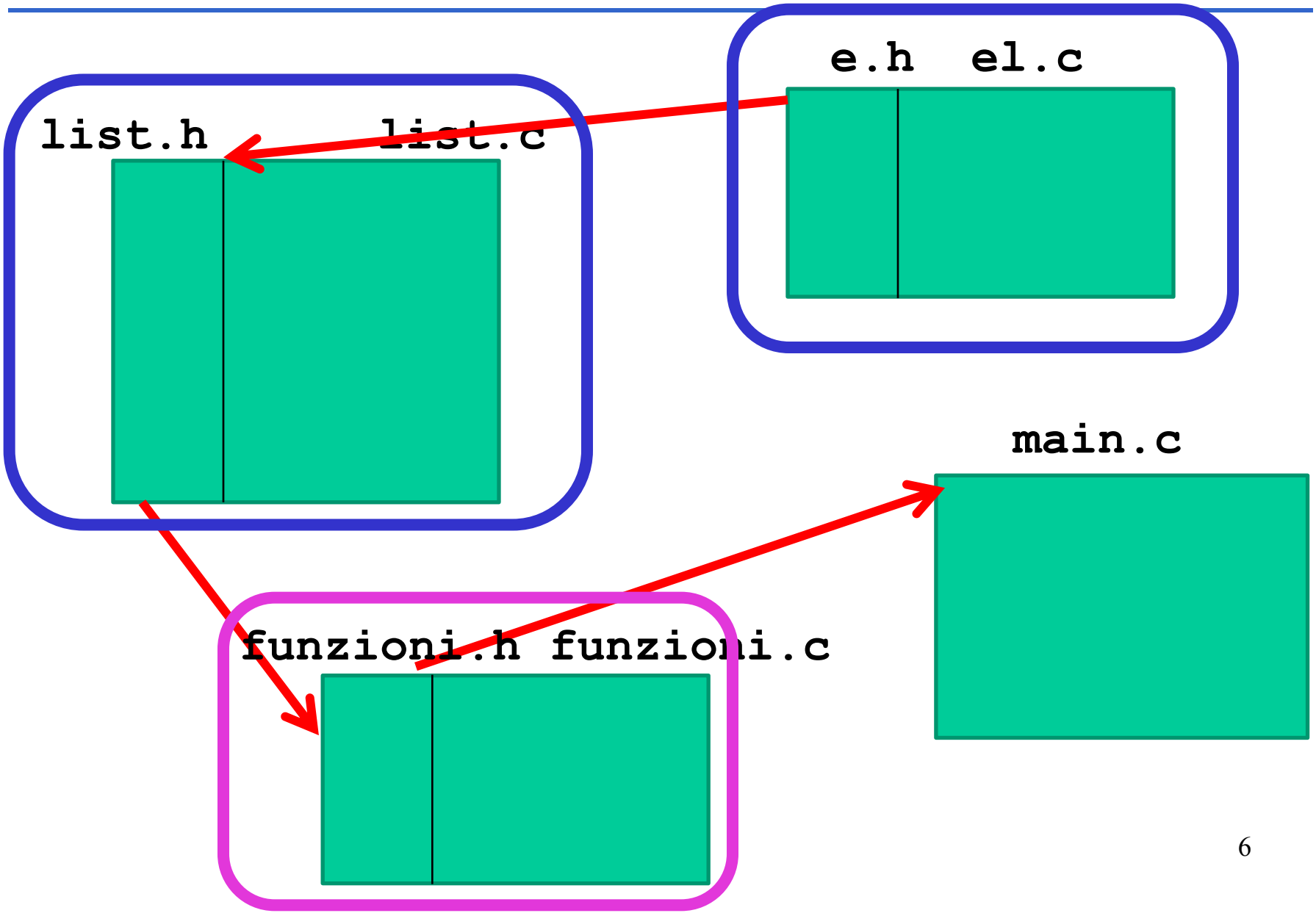
PROVA INTERMEDIA 26 Aprile 2018

- a partire dalla lista L creata, determinare **il valore medio dell'energia** (tipo float), calcolato come media di quelli degli alimenti nella lista L, stampando tale valore (float) su un file di uscita di tipo testo **iper.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.

Cosa cambia?

COMPONENTI (*cerchiati in blu non cambiano*)



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 <string.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

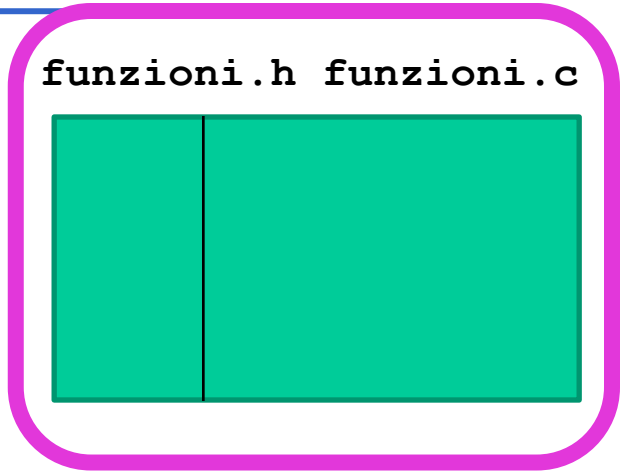
```
#include <stdio.h>
#include <stdlib.h>
#include "funzioni.h"

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 */
```

main.c (*continua*)

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

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

     funzioneC(output, L);

     fclose(output);

}
```

funzioni.h e funzioni.c

```
/*funzioni.h */
#include "list.h"
list funzioneA(FILE *f,list L);
void funzioneC(FILE *output,list L);


/*funzioni.c */
#include <stdio.h>
#include "funzioni.h"
#define IPER 300

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



Domanda c)

• Scandire in contenuto della lista ***L***, scrivendo in uscita sul file di uscita la media delle calorie

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, sommare tutti valori delle calorie, e tenere il conto ...

E' una variante della solita **showList** ... con due conteggi in più (una somma e un conteggio)

funzioni.c (*continua*)

```
void funzioneC(FILE *output, list L) {
    float media=0.0;
    int tot=0;
    while (L != NULL) {
            media = media + L->value.val);
            tot++;
            L=L->next;
     }
    fprintf(output,"Media: %f\n",media/tot);
}
```

CONSIDERAZIONI (*finali*)

Organizzare il codice per componenti ci è, in genere, utile!

Qualche **funzionalità** può essere **riutilizzata** …

CONSIDERAZIONI (*finali*)

In sede di esame, leggete bene il testo, e comprendete cosa vi viene chiesto

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

Almeno una domanda non standard c'è ... (conteggio, media, selezione di elementi da file o da lista che rispettano una certa proprietà, etc)

Soluzione su un solo file

```
#include <stdio.h>
#include <string.h>
#include <stdlib.h>
#define DIM 50
#define IPER 300

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

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

typedef item *list;
```

Soluzione su un solo file

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

Soluzione su un solo file

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

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;  }   }
}
```

Soluzione su un solo file

```
void funzioneC(FILE *output, list L) {
    float media=0.0;
    int tot=0;
    while (L != NULL) {
            media = media + L->value.val);
            tot++;
            L=L->next;
     }
    fprintf(output,"Media: %f\n",media/tot);
}
```

Soluzione su un solo file

```
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 */
```

Soluzione su un solo file

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

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

     funzioneC(output, L);

     fclose(output);

}
```