Alberi (binari di ricerca) di elementi strutture

Obiettivi:

- Riprendere cosa cambia rispetto a alberi di scalari
- Vantaggi dall' adozione di progetto su più file
- Esempio da prove d'esame

Esame 26 Luglio 2018

In un file binario **alimenti.bin** sono scritte le quantità di prodotti alimentari raccolti dal Banco Alimentare in una settimana. Per ciascun prodotto, è memorizzato:
- il nome del prodotto (stringa di 50 char),
- e la quantità (intero, come numero di scatole).

Ad esempio, per il prodotto cracker si ha:

```
"cracker" 439
```

perché si sono raccolte 439 scatole di cracker.
Si realizzi un programma C, **organizzato in** almeno tre **funzioni**, rispettivamente dedicate a:

•a partire dal file **alimenti.bin**, creare **un albero binario di ricerca T,** ordinato sul nome del prodotto, che contiene i dati dei prodotti con più di 100 scatole; la **funzioneA** riceve come parametri:

- il puntatore al file,
- il puntatore a T (inizializzato a NULL nel main),

più eventuali parametri a scelta, e restituisce il puntatore alla radice dell'albero T;

PROVA 26 Luglio 2018

- stampare a video l'elenco ordinato di prodotti (e il numero di scatole di ognuno) dall'albero T; **la funzioneB** riceve come parametri:
  - il puntatore a T,

più eventuali parametri a scelta, e restituisce void ;

- stampare il contenuto dell'albero su un file testo di uscita, **out1.txt,** e il **totale** delle scatole**;** la **funzioneC** riceve come parametri
  - il puntatore al secondo file, aperto nel main
  - il puntatore a T,

più eventuali parametri a scelta, e restituisce void.

NOTA BENE: Si consegnino i sorgenti e il file di uscita generato. E' possibile utilizzare librerie C (ad es. per le stringhe). Nel caso si strutturi a moduli l'applicazione qualunque libreria utente va riportata nello svolgimento.

Domanda A+B (30 min in più)

Si scriva una funzione (**funzioneAB**) per riempire un array di 30 elementi, V, in cui ciascun elemento è costituito da una parola (stringa al massimo di 50 char). Sapendo che il file **alimenti.bin** ha più di 30 prodotti, la funzione inserisce nel vettore V i nomi dei primi 30 prodotti del file; la **funzioneAB** riceve come parametri il vettore V e il puntatore al file **alimenti.bin** più eventuali parametri a scelta, e restituisce void.

Nel main, si ordini il vettore V tramite una opportuna chiamata della funzione **qsort** e si stampi poi il contenuto dell'array V su un file di testo **outputAB.txt**

Domanda a)

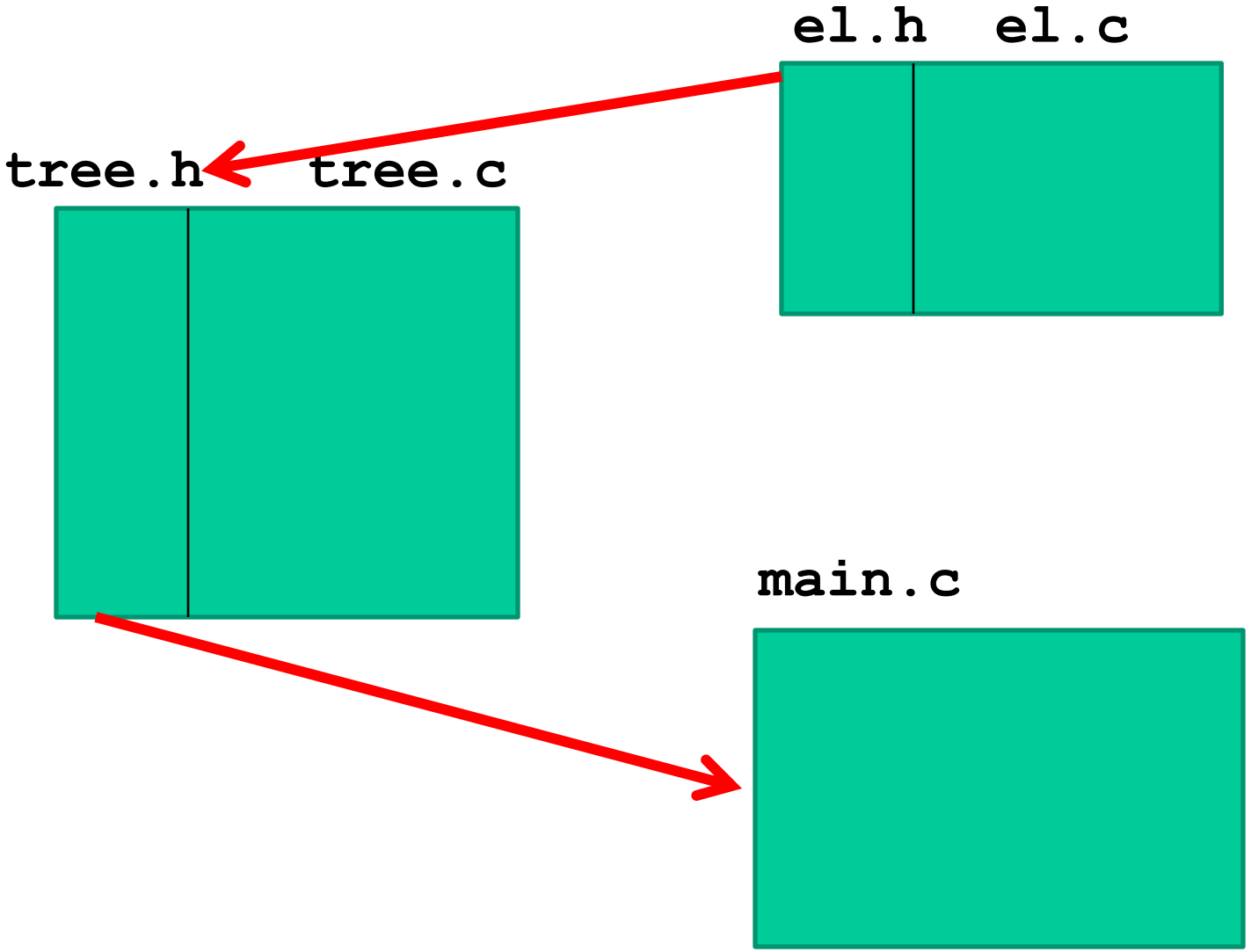
Il programma deve chiamare tre funzioni (da definire) dedicate rispettivamente a:

a) Caricare in un albero binario di ricerca T in memoria centrale i record del file, con quantità maggiore di 100

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

Usate la ***strcmp*** di string.h

COMPONENTI



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 **tree** (con elementi di tipo ***element***)

Definire albero binario ***T***

Domanda a), come fareste?

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

1 record per volta

Inserire l'elemento letto in T con ***insord***, che usa la funzione ***isless***

el.h

```
#define DIM 50

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

int isless(element a, element b);
void printel(element x);
```

el.c

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

int isless(element a, element b) {
    if (strcmp(a.nome, b.nome) > 0)
        return 0;
    else return 1;
}

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

tree.h

```
#include "el.h"
typedef struct nodo {
    element value ;
    struct nodo *left ;
    struct nodo *right ; } NODO;

typedef NODO *tree;

tree cons_tree(element el, tree l, tree r);
tree insord(element el, tree t);
void inorder(tree t);
```

tree.c

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

tree cons_tree(element el, tree l, tree r) {
    tree aux = (NODO *)malloc(sizeof(NODO));
    aux->value = el;
    aux->left = l;
    aux->right = r;
    return aux;       }

tree insord(element el, tree t) {
    if (t == NULL)  return cons_tree(el, NULL, NULL);
    else { if (isless(el, t->value))
                t->left = insord(el, t->left);
           else t->right = insord(el, t->right);
           return t;    }
}
```

tree.c

```
/* funzioneB */

void inorder(tree t) {
     if (t != NULL) {
          inorder(t->left);
          printel(t->value);
          inorder(t->right);
     }
}
```

main.c

```
#include <stdio.h>
#include <stdlib.h>
#include "tree.h"
#define NUM 100

tree funzioneA(FILE *f, tree T);
void funzioneC(FILE *output, tree T);
void funzioneAB(FILE *f, stringa *Vet)

main() { int tot=0;
    tree T = NULL;
    FILE *f, *output;
    f = fopen("alimenti.bin", "rb");
    if (f == NULL) {
        printf("Non aperto.\n");
        exit(-1);
    }
    T = funzioneA(f, T);
    fclose(f);
```

main.c (*continua*)

```
 inorder(T);    /* funzioneB */

  output = fopen("out1.txt", "wt");
  if (output == NULL) {
        printf("File non creato.\n");
        exit(-1);
    }

  funzioneC(output, T, &tot);

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

  fclose(output);

/* segue parte A+B */

}
```

main.c (*continua*)

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

PROVA 26 Luglio 2018 (*continua*)

- stampare a video l'elenco ordinato di prodotti (e il numero di scatole di ognuno) dall'albero T; **la funzioneB** riceve come parametri:
  - il puntatore a T,

più eventuali parametri a scelta, e restituisce void ;

- stampare il contenuto dell'albero su un file testo di uscita, **out1.txt,** e il **totale** delle scatole**;** la **funzioneC** riceve come parametri
  - il puntatore al secondo file, aperto nel main
  - il puntatore a T,

più eventuali parametri a scelta, e restituisce void.

main.c (*continua*)

```
inorder(T);    /* funzioneB */

 output = fopen("out1.txt", "wt");
 if (output == NULL) {
       printf("File non creato.\n");
       exit(-1);
   }

  funzioneC(output, T, &tot);

 fprintf(output,"N.ro scatole totali:%d\n", tot);

 fclose(output);

  /* segue parte A+B */
}
```

tree.c

```
/* funzioneB */

void inorder(tree t) {
     if (t != NULL) {
          inorder(t->left);
          printel(t->value);
          inorder(t->right);
     }
}
```

Domanda c)

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

Deve visitare l'albero in ordine l'albero, stampare su file ogni elemento, e tenere il conto ...

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

Aggiungiamo parametro intero passato per indirizzo

main.c (*continua*)

```
inorder(T);    /* funzioneB */

 output = fopen("out1.txt", "wt");
 if (output == NULL) {
       printf("File non creato.\n");
       exit(-1);
   }

  funzioneC(output, T, &tot);

 fprintf(output,"N.ro scatole totali:%d\n", tot);

 fclose(output);

  /* segue parte A+B */
}
```

main.c (*continua*)

```
void funzioneC(FILE *output,tree T,int *p)
{
    if (T != NULL)
    {   funzioneC(output, T->left, p);
        fprintf(output,"%s %d\n",
                        T->value.nome,
                        T->value.val);
        *p = *p+ T->value.val;
        funzioneC(output, T->right, p);
      }
}
```

PROVA 26 Luglio 2018 (*continua*) - esame A+B

Si scriva una funzione (**funzioneAB**) per riempire un array di 30 elementi, V, in cui ciascun elemento è costituito da una parola (stringa al massimo di 50 char). Sapendo che il file **alimenti.bin** ha più di 30 prodotti, la funzione inserisce nel vettore V i nomi dei primi 30 prodotti del file; la **funzioneAB** riceve come parametri il vettore V e il puntatore al file **alimenti.bin** più eventuali parametri a scelta, e restituisce void.

Nel main, si ordini il vettore V tramite una opportuna chiamata della funzione **qsort** e si stampi poi il contenuto dell'array V su un file di testo **outputAB.txt**

main.c (*continua*)

```
/* nel main ... Parte A+B*/
typedef char[50] stringa;
stringa V[30];  int i=0;
...
f = fopen("alimenti.bin", "rb");
output = fopen("outputAB.txt", "wt");

funzioneAB(f, V);
qsort(V,30, sizeof(stringa),strcmp);
for (i=0;i<30;i++)
        fprintf(output,"%s\n", V[i]);
fclose(f);
fclose(output);
/*fine main */
}
```

main.c (*continua*)

```
void funzioneAB(FILE *f, stringa *Vet) {
   int i=0;
   element el;
   while((fread(&el,sizeof(element),1,f)>0)&&
         (i<30))
      {    strcpy(Vet[i],el.nome);
           i++;   }
}
```

Considerazioni finali e consigli

- Svolgere man mano con la fruizione delle lezioni le esercitazioni proposte
- Presentarsi all'esame se e quando si è preparati (ovvero in grado di sviluppare in autonomia un programma, partendo dalle specifiche date)
- Leggere bene il testo della prova ...
- Almeno una domanda non standard su elaborazione di alberi (o liste) viene sempre richiesta