Università degli Studi di Ferrara

DE: Department of Engineering Ferrara

Esercitazioni di

FONDAMENTI DI INFORMATICA MODULO B

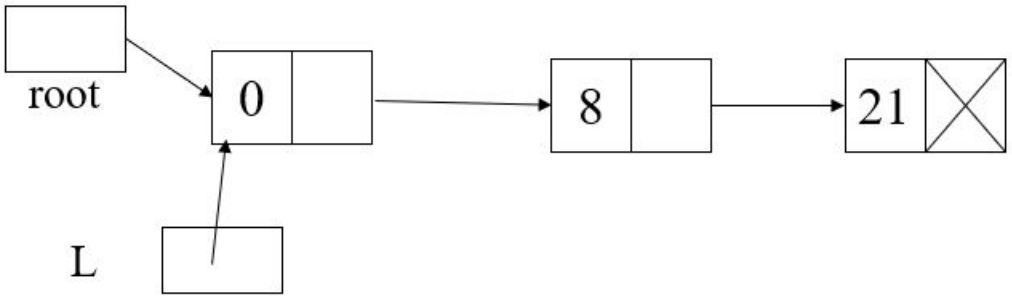
ESERCITAZIONE 4

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

Esercizio - Creazione di una lista

Si sviluppi un programma che legga da tastiera una sequenza di numeri interi terminata da 0. Inserire poi ogni elemento letto in una lista utilizzando l'**inserimento in testa**. La funzione deve avere il seguente prototipo: list cons(int n, list L)

Per ogni inserimento, **malloc** e aggiustamento dei puntatori.



root inizialmente **NULL**

Esercizio - Creazione di una lista

```
#include <stdlib.h>
#include <stdio.h>
typedef struct list_element { int value;
        struct list_element *next; } item;
typedef item *list;

int main() {
        list L; int i;
        do {
                printf("\n Introdurre valore (0 per terminare): ");
                scanf("%d", &i);

                L = cons(i, L); // da implementare

        } while (i!=0);
}
```

Esercizio - Stampa di una lista

Partendo dalla lista precedente, stamparne il contenuto utilizzando una funzione con il seguente prototipo:

void **showlist**(list l);

```
// se avessimo un vettore ...
i=0;
while (i<N) {
printf("\nValore : \t%d", V[i]);
      i++;
}
```

Progettarne una versione **ITERATIVA** e una **RICORSIVA**.

Esercizio - Ricerca elemento

Definire una funzione **member** che, data una lista **l** ed un elemento n, restituisca l'indice dell'elemento (-1 se non trovato).

```
printf("\n Introdurre un valore da cercare: \t");
scanf("%d", &i);
if (member(n, l) >= 0)
      printf("Trovato\n");
else
      printf("Non trovato\n");
}
```

Non abbiamo un "indice" da incrementare per scandire la lista, come succedeva per la rappresentazione sequenziale mediante vettori.

La lista va *scandita (sequenzialmente) con un* ***puntatore***

Progettarne una versione **ITERATIVA** e una **RICORSIVA**.

Esercizio - Inserimento in coda

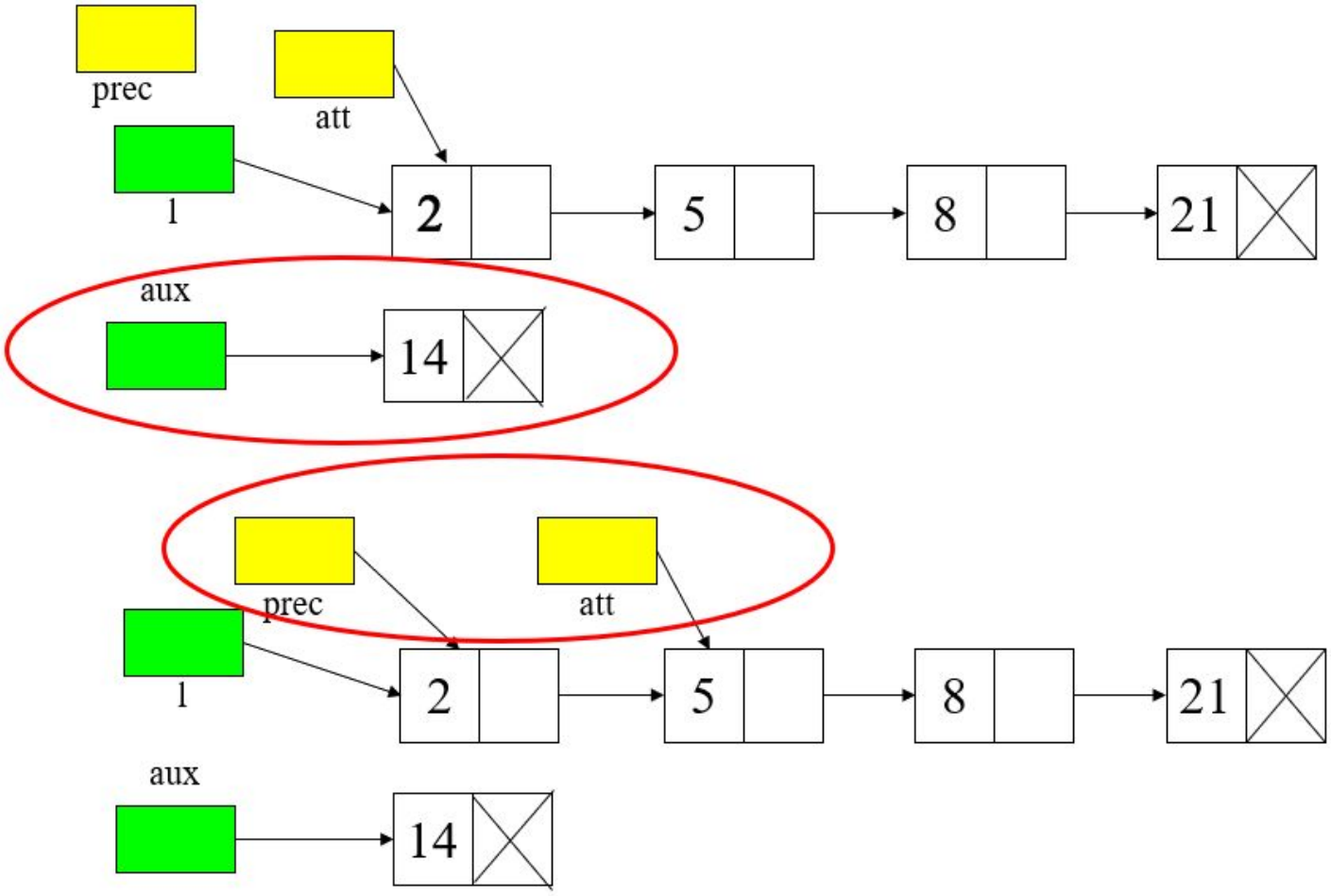
Sviluppare una funzione per l'inserimento di elementi in coda ad una lista con il seguente prototipo:

list **cons_tail**(int n, list l)

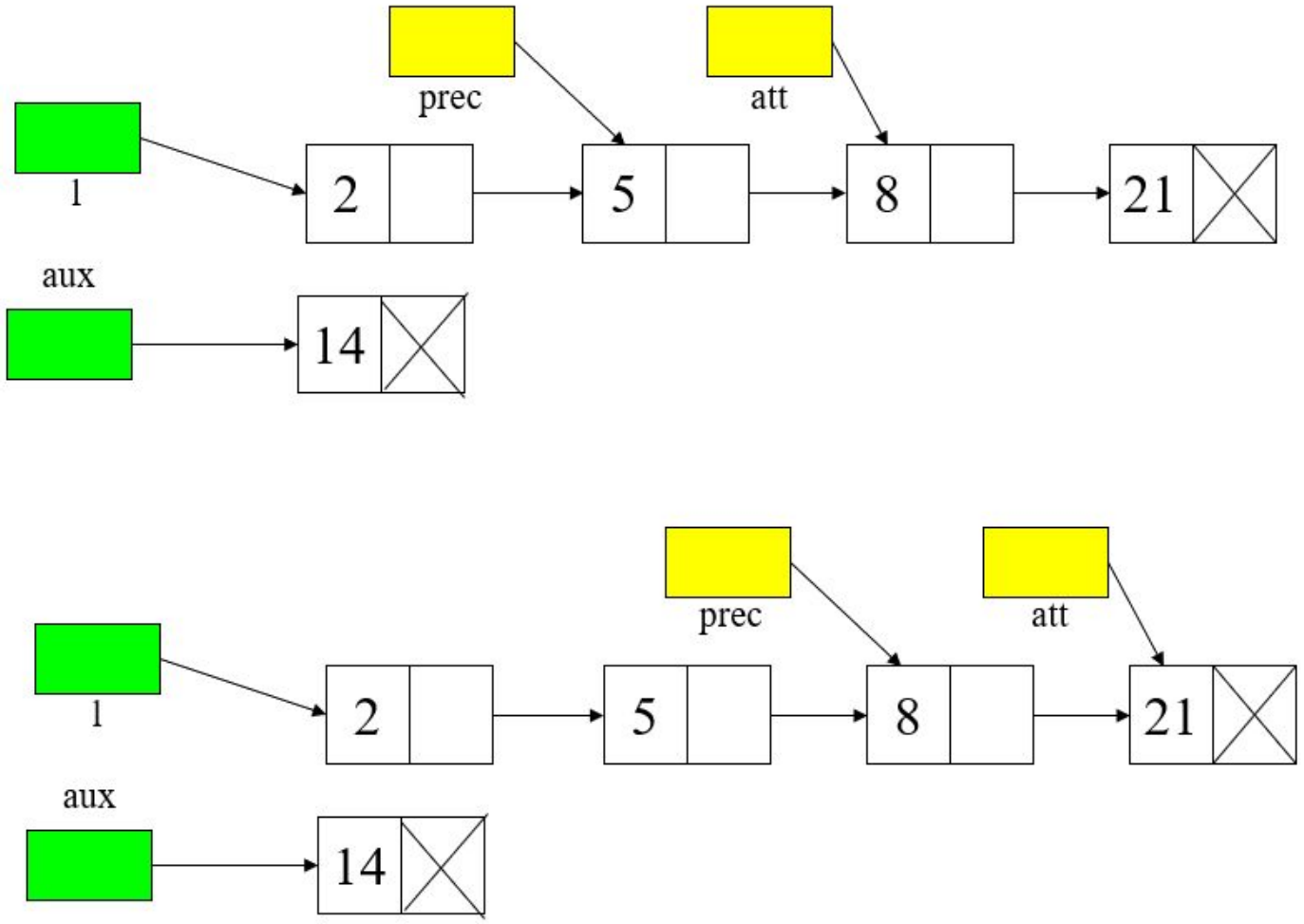
Aggiungiamo al main che utilizza già chiamate a **cons** e **showlist** anche **cons_tail**. Leggere una sequenza di interi da input e costruire due liste: L1 con inserimento in testa e L2 con inserimento in fondo. Infine stampare le due liste.

Come si presentano le due sequenze visualizzate?

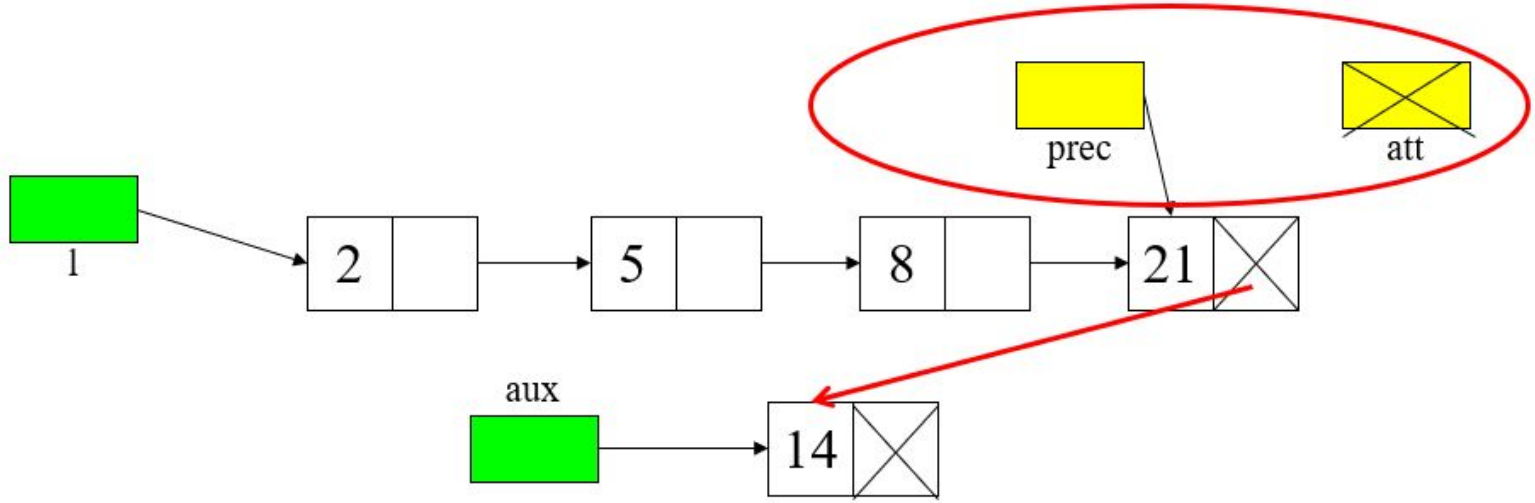
Esercizio - Inserimento in coda



Esercizio - Inserimento in coda



Esercizio - Inserimento in coda



Esercizio - Sublist

Definire una funzione **sublist** che, dato un intero positivo **n** e una lista **l**, restituisca una lista che rappresenti la sotto-lista di quella data a partire dall'elemento n-esimo.

ESEMPIO:

```
l = [1, 13, 7, 9, 10, 1]
// stampo dall'elemento di indice 2 in poi
sublist(2, l) = [7, 9, 10, 1]
```

Progettarne una versione **ITERATIVA** e una **RICORSIVA**.

Esercizio - Numero elementi

Definire una funzione **length** che, data una lista **l**, ne restituisca il numero di elementi contenuti.

ESEMPIO:

```
l = [1, 13, 7, 9, 10, 1]
length(l) = 6
```

Progettarne una versione **ITERATIVA** e una **RICORSIVA**.

Esercizio

Ricapitolando: implementare le seguenti funzioni:

- Funzione **cons** per l'inserimento di elementi in testa ad una lista.
- Funzione **cons_tail** per l'inserimento di elementi in coda ad una lista.
- Funzione **showlist** per la stampa di una lista. **R+IT**
- Funzione **member** per la ricerca di un elemento n nella lista. **R+IT**
- Funzione **length** per il calcolo della lunghezza della lista. **R+IT**
- Funzione **sumlist** per il calcolo della somma degli interi contenuti nella lista. **R+IT**
- Funzione **sublist** per l'estrapolazione di una sotto-lista, dalla posizione k alla fine. **R+IT**

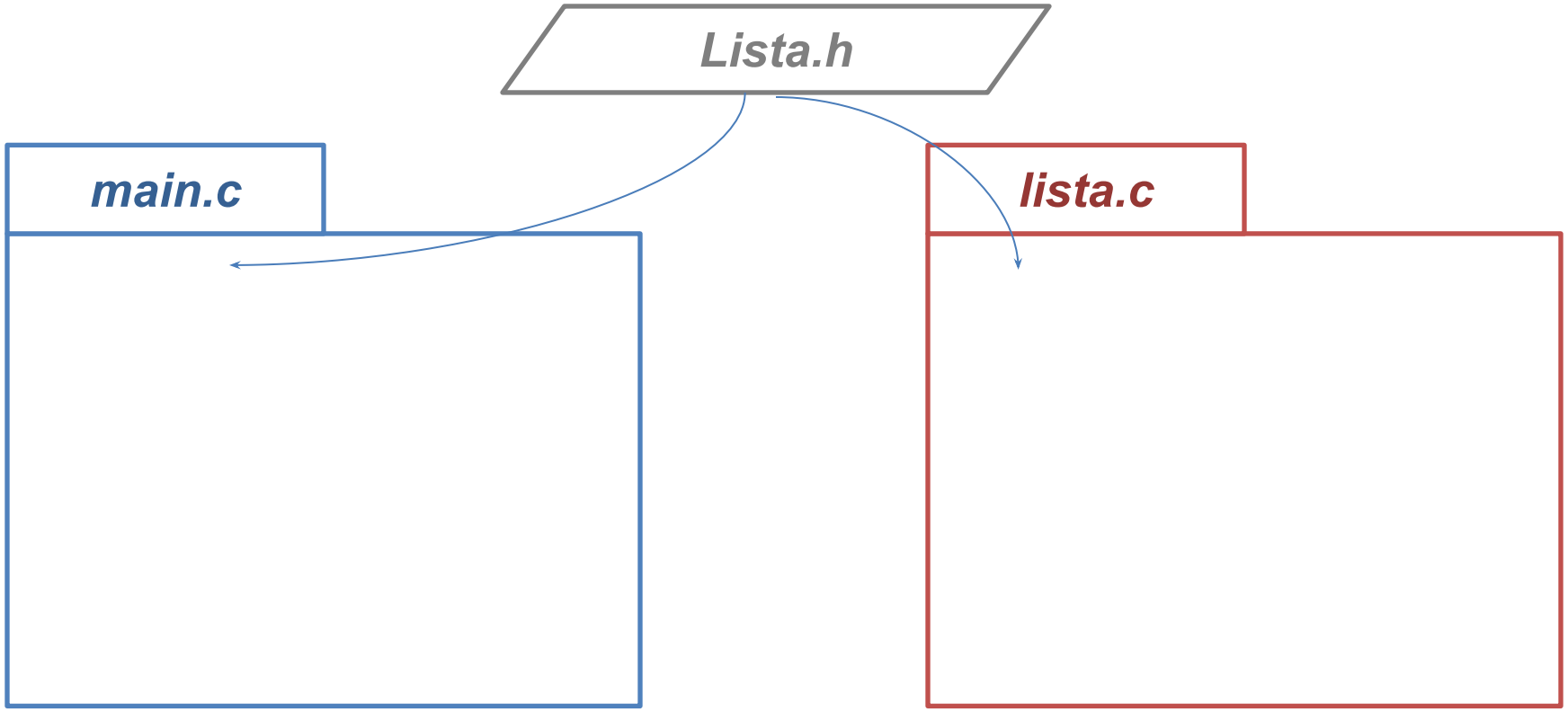
Strutturare il progetto su più file

Esercizio

Se la sequenza in ingresso è: 1,2,3,4,5,6,(0), il comportamento delle funzioni dovrà essere il seguente:

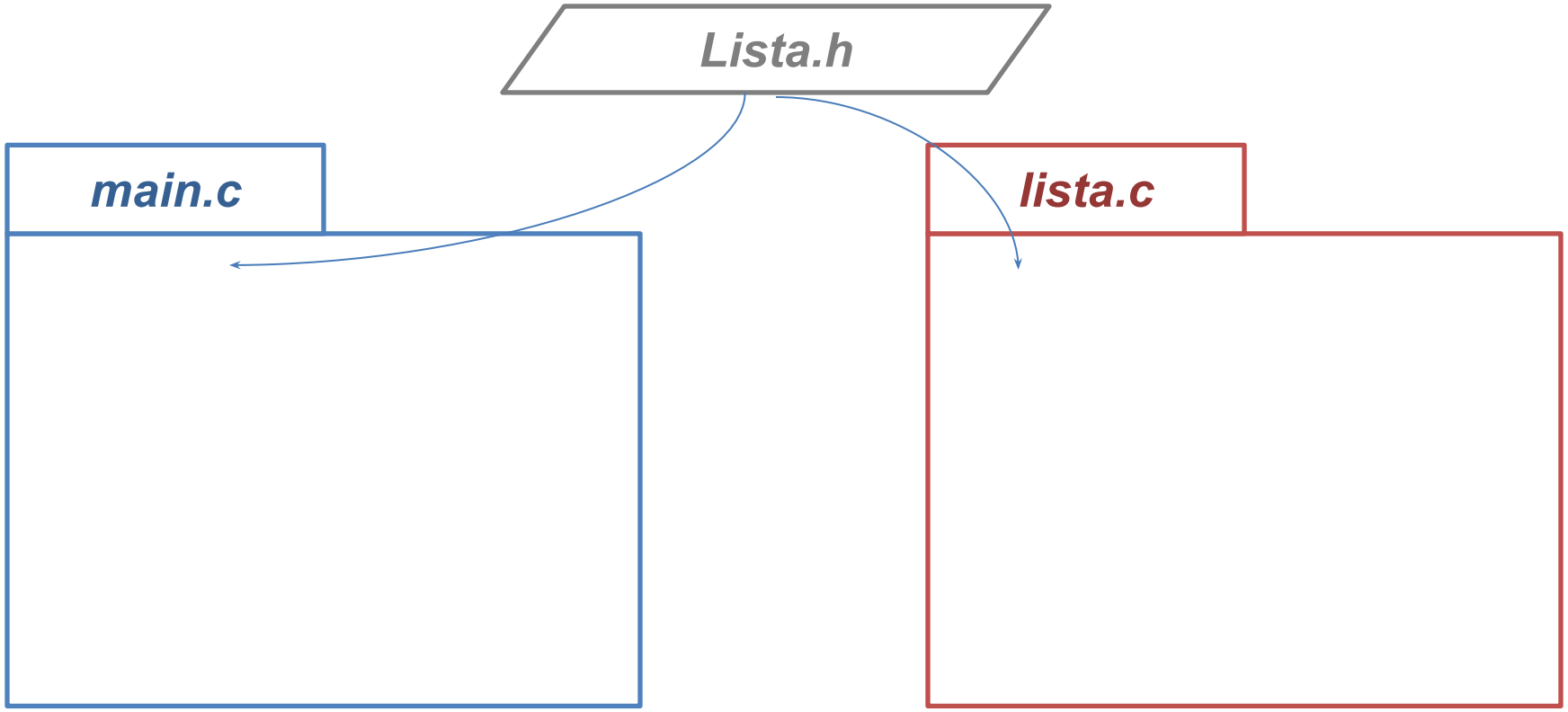
- **cons (+ showlist)**: 6 5 4 3 2 1
- **cons_tail (+ showlist)**: 1 2 3 4 5 6
- **member**:
  - inserisci un intero da cercare nella lista: 2
  - 2 e' membro della lista
- **length**: lunghezza lista 6
- **sumlist**: somma lista 21
- **sublist**:
  - estrapola la sottolista a partire dall'elemento in posizione: 2
  - 4 3 2 1
  - 3 4 5 6

Esercizio



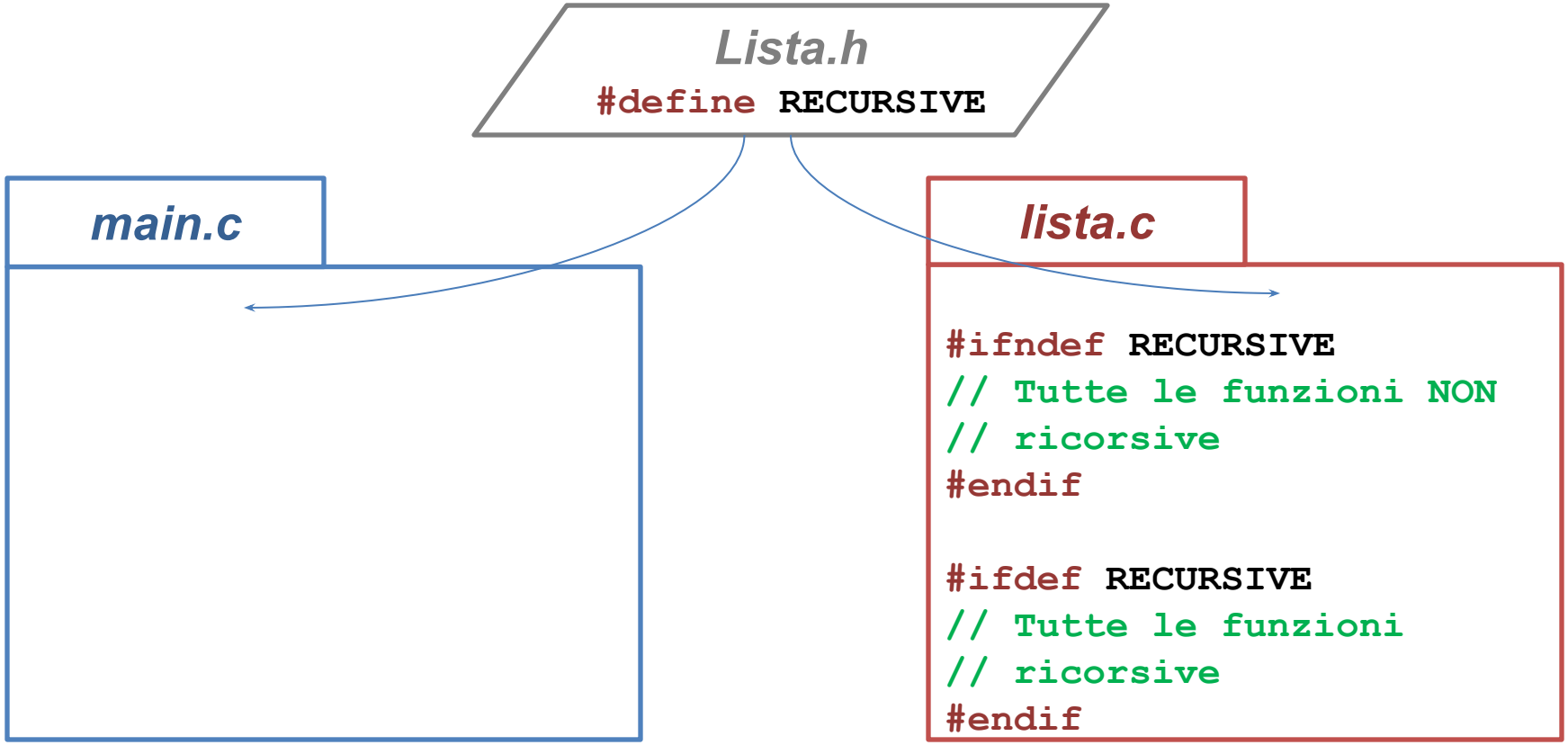
Come possiamo definire due funzioni che eseguono la stessa operazione ma in modi diversi?

Esercizio



Banalmente usando un nome di funzione diverso (**member**, **member_r**). Qual è un’altra soluzione possibile?

Esercizio



Banalmente usando un nome di funzione diverso (**member**, **member_r**)
Meno banalmente: usando istruzioni al precompilatore (**#ifdef**, **#ifndef**)

Esercizio

Lista.h

```
#define RECURSIVE

typedef struct node{
    int n;
    struct node *next;
} listNode;
typedef listNode* list;
typedef enum {FALSE, TRUE} boolean;

list cons(int n, list l);
list cons_tail(int n, list l);
int sumlist(list l); //R+IT
void showlist(list l); //R+IT
int length(list l); //R+IT
boolean member(int n, list l); //R+IT
list sublist(int k, list l); //R+IT
```