Programmi su più moduli - Esempio

Calcolo del logaritmo in base qualunque

- Incapsulare la logica di calcolo in una funzione
  - PASSO 1: definisco la dichiarazione della funzione (nome, parametri di input e di output)

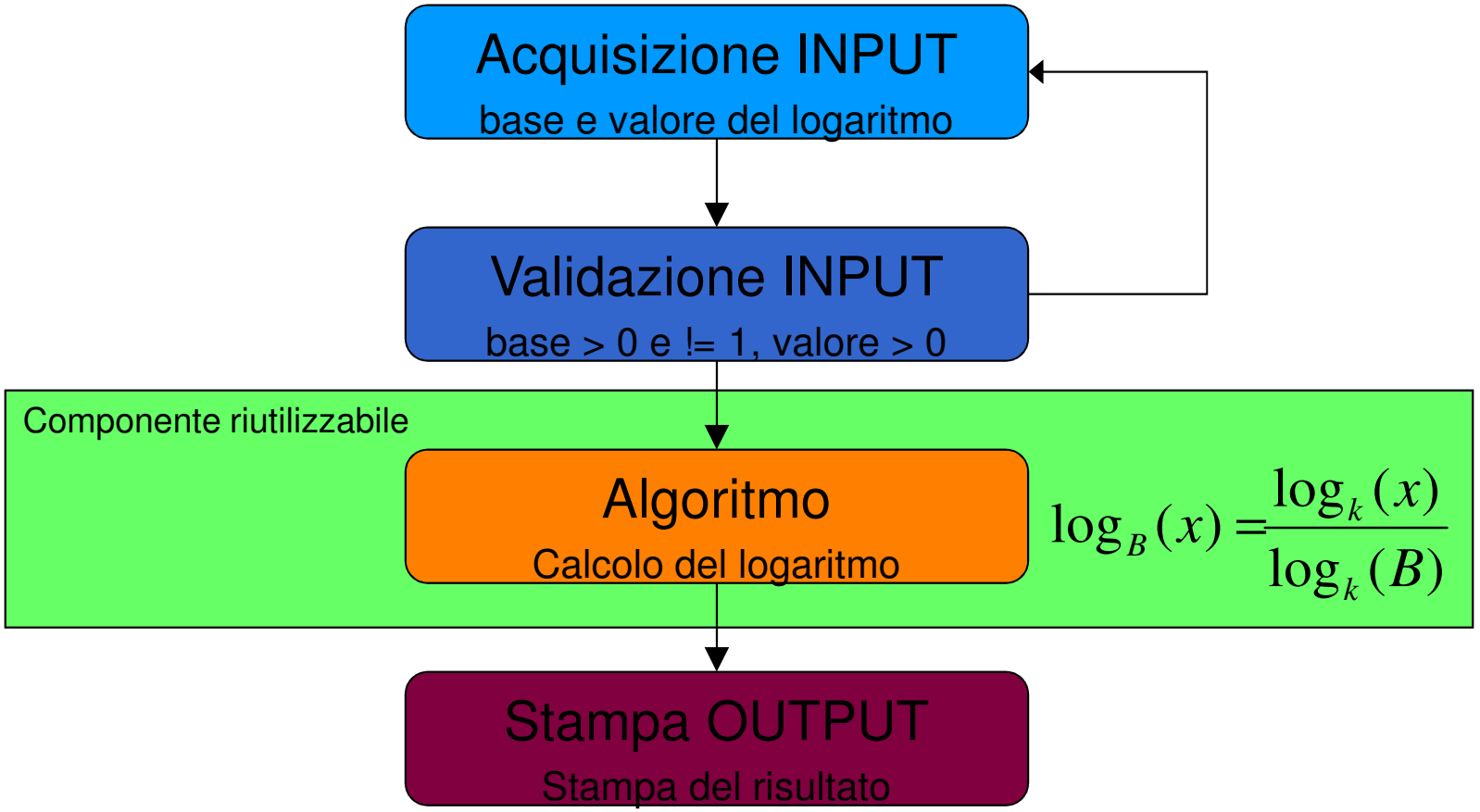
```
float mylog(float base, float value)
```

  - PASSO 2: realizzo la funzione

$$\log_B(x) = \frac{\log_k(x)}{\log_k(B)}$$

Programmi su più moduli - Esempio

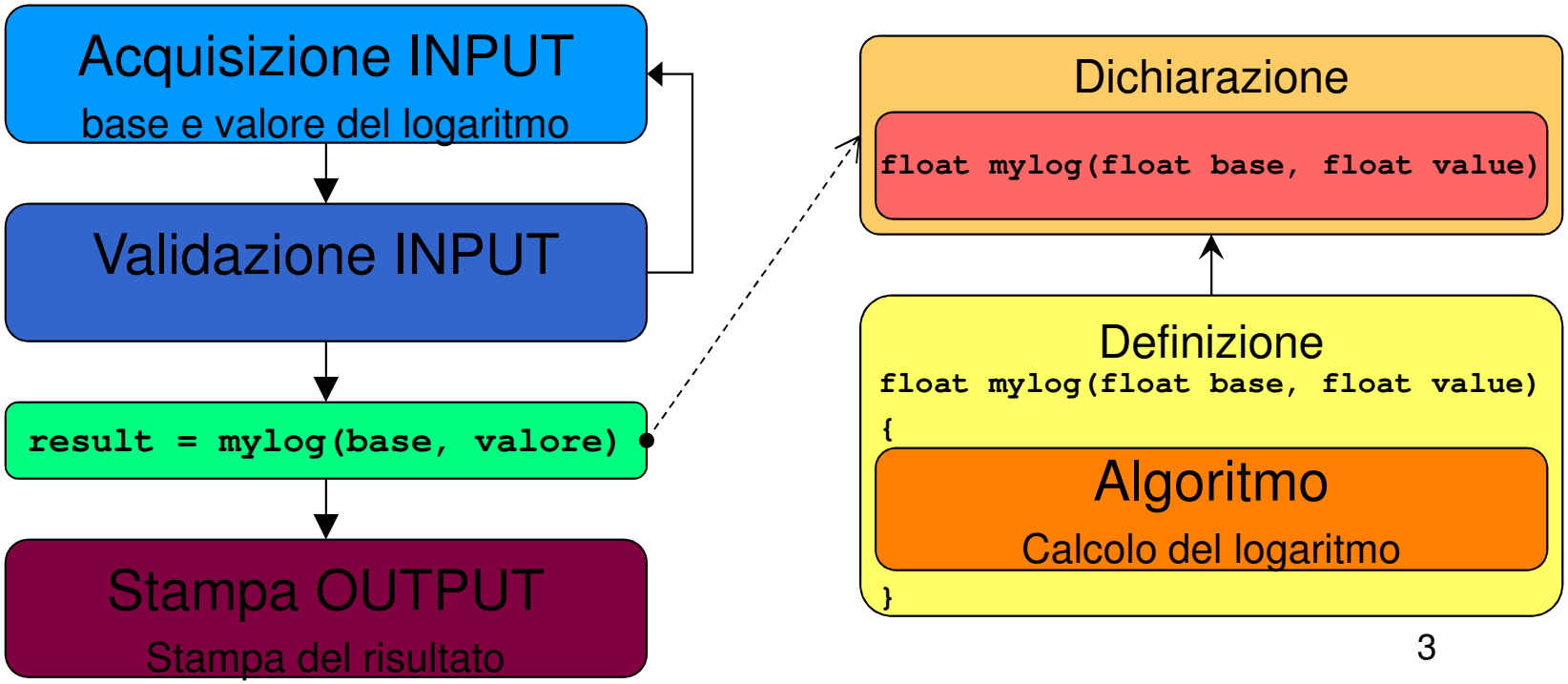
Calcolo del logaritmo in base qualunque - schema di soluzione



Programmi su più moduli - Esempio

- Logaritmo come componente: uso di una *funzione*!
  - PASSO 1: **dichiarazione della funzione (nome, parametri di input, parametri di output)**

    ```
    float mylog(float base, float value)
    ```
  - PASSO 2: **definizione della funzione** (ovvero implementazione)

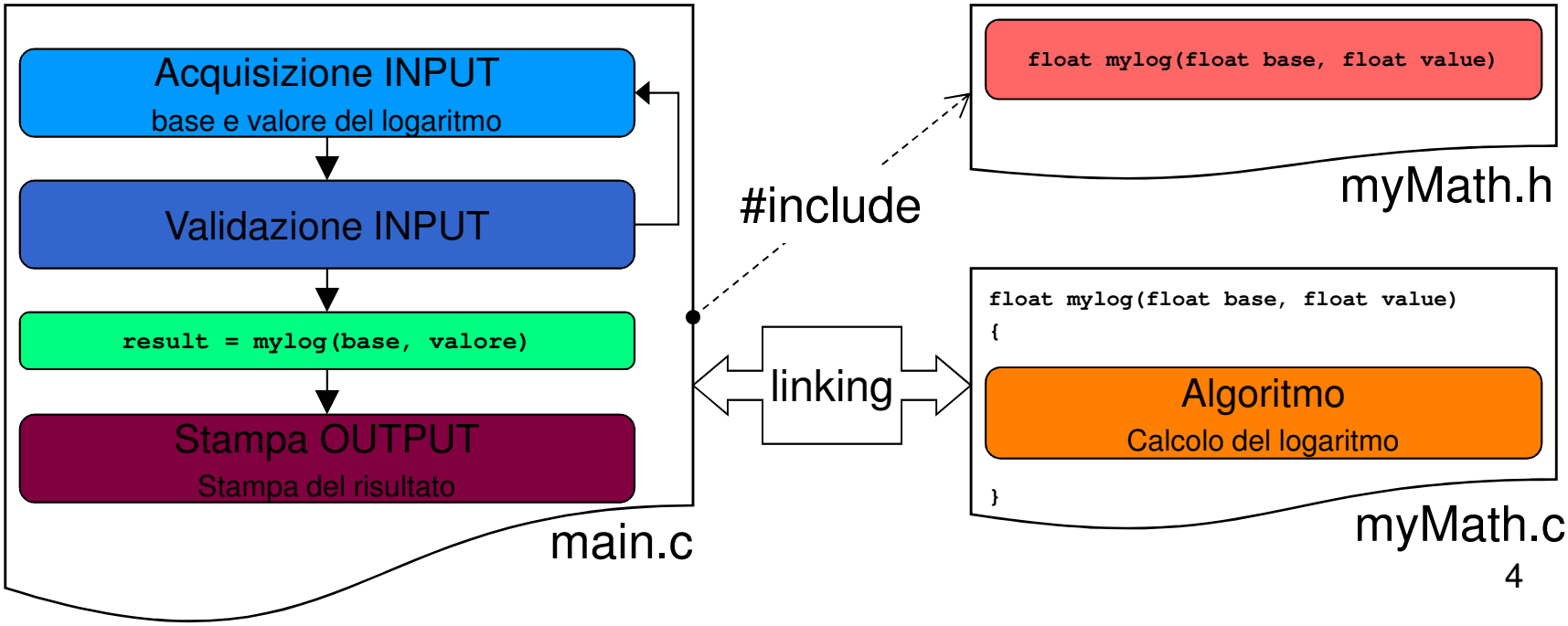


Programmi su più moduli - Esempio

Vogliamo rendere la funzione **mylog** davvero utilizzabile da più utenti in più programmi: creazione di un ***modulo apposito***

- header file contenente le dichiarazioni (ad es. “myMath.h”)
- file C contenente le definizioni (ad es. “myMath.c”)
- includiamo “myMath.h” nel modulo che contiene la funzione main
- compiliamo con l’istruzione opportuna:

```
cl myProg.c myMath.c /I myMath.h /o myProg.exe
```



Programmi su più moduli - Esempio

myMath.h:

```
float mylog(float base, float value);
```

myMath.c:

```
#include <math.h>

float mylog(float base, float value)
{
  return log(value) / log(base);
}
```

In realtà, dovrebbe contenere anche la validazione dei dati in input (*mai fidarsi del cliente!*) e restituire **errore** in caso di input non corretto

Programmi su più moduli - Esempio

main.c:

```
#include “myMath.h”
#include <stdio.h>


int main ()
{
  double b, x, result;
  …
  result = mylog(b, x);
  …
  return 0;
}
```

Esercizio 1
(Funzioni e programmi su più moduli)

Ciclo per il calcolo del massimo e del minimo

- Realizzare un programma che calcoli il minimo e il massimo di una serie di valori
- Il numero di valori deve essere costante e definito tramite una opportuna *costante simbolica*
- Se la differenza tra il massimo e il minimo supera 10, il programma termina, altrimenti aspetta una nuova serie di valori
- Incapsulare ***il calcolo del minimo e del massimo in funzioni apposite, definite in un apposito modulo***

Esercizio 1
(Funzioni e programmi su più moduli)

- Quanti cicli? 2
  - Uno esterno per capire se uscire dal programma o richiedere la serie di valori
  - Uno interno per acquisire i K valori
- Che tipo di cicli?
  - Ciclo esterno: verifica una condizione a posteriori

    ➔ do…while
  - Ciclo interno: conosce a priori il numero di iterazioni

    ➔ for
- Di quanti valori devo tener traccia?
  - Ricordarsi che il minimo ed il massimo si possono calcolare passo passo
- Quando devo re-inizializzare il massimo ed il minimo?

Esercizio 1 - Soluzione
(Funzioni e programmi su più moduli)

File "myMath.h":

```
int max(int v1, int v2);
int min(int v1, int v2);
```

File "myMath.c":

```
#include "myMath.h"

int max(int v1, int v2) {
    if(v1 > v2)
        return v1;
    else
        return v2;
}

int min(int v1, int v2) {
    return v1 < v2 ? v1 : v2;
}
```

Esercizio 1 - Soluzione
(Funzioni e programmi su più moduli)

File "main.c":

```
#include <stdio.h>
#include "myMath.h"
#define MAX_REQUEST 5

int main () {
   int curValue, maxValue, minValue, index;
   do {
        for(index = 0; index < MAX_REQUEST; index++) {
                printf("Inserire il valore %d: ", index+1);
                scanf("%d", &curValue);
                if(index == 0) //inizializzo max e min
                {
                        maxValue = curValue;
                        minValue = curValue;
                }
                else {
                        maxValue = max(maxValue, curValue);
                        minValue = min(minValue, curValue);
                }
        }
        printf("Calcolati: max = %d, min = %d\n", maxValue, minValue);
   } while(maxValue - minValue <= 10);
   return 0; }
```

Esercizio 2
(Funzioni e programmi su più moduli)

Calcolo del mcm tra numeri interi

- Realizzare un programma che prenda in input una serie di numeri interi, calcolando via via il minimo comune multiplo tra essi; il programma deve terminare quando mcm diventa più grande di 100
  - Ricordarsi che, come per il massimo e il minimo, anche mcm si può calcolare in modo parziale
  - Quindi basta utilizzare, per il calcolo, il valore di mcm calcolato al passo precedente e il numero inserito al passo corrente

    mcm(a, b, c) = mcm(mcm(a, b), c)

Esercizio 2
(Funzioni e programmi su più moduli)

- Utilizzare la relazione $mcm(a,b) = \frac{a \cdot b}{MCD(a,b)}$
- Utilizzare l'algoritmo di Euclide per il MCD tra due numeri
  - Finché M≠N:
    - se M>N, sostituisci a M il valore M' = M-N
    - altrimenti sostituisci a N il valore N' = N-M
    - MCD è il valore finale ottenuto quando M e N diventano uguali
- Incapsulare il calcolo di mcm e MCD in due funzioni
  - Ragionare per astrazione!
    - Individuare prima come le funzioni vengono viste dall'esterno (dichiarazione), poi realizzarle

Esercizio 2
(Funzioni e programmi su più moduli)

- Procedere per passi
  - Prima definiamo la funzione per MCD e proviamo a testarla su due valori
  - Poi definiamo la funzione per mcm e proviamo a testarla su due valori
  - Poi realizziamo il programma ciclico
- Per ultimo, utilizziamo l'approccio a moduli inserendo il calcolo di MCD e mcm in un modulo di libreria
  - *Header File contenente le dichiarazione delle funzioni*

Esercizio 2
(Funzioni e programmi su più moduli)

- Esempio di esecuzione

```
Inserisci il primo valore: 4
Inserisci un valore: 8
mcm corrente: 8
Inserisci un valore: 12
mcm corrente: 24
Inserisci un valore: 10
mcm corrente: 120
```

Esercizio 2 - Soluzione
(Funzioni e programmi su più moduli)

File "myMath.h":

```
int mcd(int a, int b);
int mcm(int a, int b);
```

File "myMath.c":

```
#include "myMath.h"
int mcd(int a, int b) {
    int m, n;
    m = a;
    n = b;
    while(m != n) {
        if(m > n)
            m = m - n;
        else
            n = n - m;
    }
    return m;
}
int mcm(int a, int b) {
    return (a * b) / mcd(a, b);
}
```

Esercizio 2 - Soluzione
(Funzioni e programmi su più moduli)

File "main.c":

```
#include "myMath.h"
#include <stdio.h>

int main() {
   int curMcm, curValue;
   printf("Inserisci il primo valore: ");
   scanf("%d", &curMcm);
   do {
       printf("Inserisci un valore: ");
       scanf("%d", &curValue);
       curMCD = mcm(curMcm, curValue);
       printf("mcm corrente: %d\n", curMcm);
   } while(curMcm <= 100);
   return 0;
}
```

Esercizio 3
(Funzioni e programmi su più moduli)

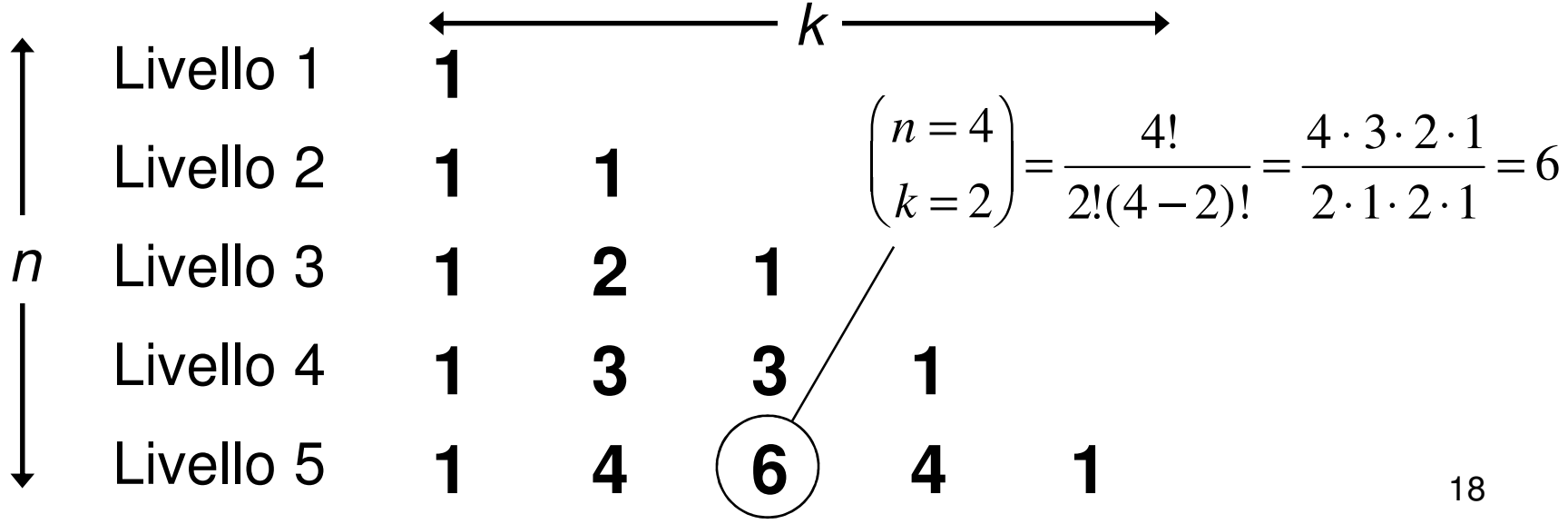
Triangolo di Tartaglia

- Realizzare un programma che, letto in input il massimo livello voluto, mostri a video il contenuto del triangolo di Tartaglia fino a quel livello
- Per la costruzione del triangolo di Tartaglia, si utilizzi la corrispondenza tra i suoi elementi e i coefficienti binomiali

Esercizio 3
(Funzioni e programmi su più moduli)

- Coefficiente binomiale $\binom{n}{k} = \frac{n!}{k!(n-k)!}$

- Triangolo di Tartaglia (5 livelli)
  - Allineato a sinistra per semplicità di stampa



Esercizio 3
(Funzioni e programmi su più moduli)

Organizzare il programma in due moduli separati

- I modulo (di libreria)
  - Funzione che calcola il fattoriale
    - Fattoriale di 0 = 1
    - Fattoriale di N = prodotto dei numeri da 1 a N
  - Funzione che calcola il coefficiente binomiale
    - A partire dalla funzione che calcola il fattoriale
  - Prima header file
- II modulo (main)
  - Acquisizione in input del numero dei livelli
  - Stampa del triangolo di Tartaglia
    - Come utilizzare i cicli? Quanti cicli sono? Che tipo di cicli?
    - Ricordarsi che il coefficiente binomiale è definito solo per $k \leq n$

Esercizio 3 - Soluzione
(Funzioni e programmi su più moduli)

File "myMath.h":

```
int fattoriale(int n);
int binomiale(int n, int k)
```

File "myMath.c":

```
#include "myMath.h"
int fattoriale(int n) {
    int fact = 1, index;
    for(index = n; index > 0; index--)
        fact = fact * index;
    return fact;
}

int binomiale(int n, int k) {
    return fattoriale(n)/ (fattoriale(k)*fattoriale(n-k));
}
```

Esercizio 3 - Soluzione
(Funzioni e programmi su più moduli)

File "myMath.c":

```
#include "myMath.h"
#include <stdio.h>

int main () {
    int N_livelli, n, k;
    scanf("%d",&N_livelli);
    for(n = 0; n < N_livelli; n++) {
        for(k = 0; k <= n; k++)
            printf("%d ", binomiale(n, k) );
        printf("\n");
    }
    return 0;
}
```