INFORMATICA A

A.A. 2017-18

Laboratorio n°5

Ing. Gian Enrico Conti

Dott. Michele Zanella

Info Logistiche (cont'd)

- Calendario laboratori

Data	Orario	Squadra	Aula	Resp.	Programma
23/10/2017	9:00 - 12:00	A	CS 0.2	Zanella	Lab 1
23/10/2017	9:00 - 12:00	B	CS 1.4	Conti	
26/10/2017	14:15-17:15	C	CS 0.11	Conti	
30/10/2017	9:00 - 12:00	A	CS 0.2	Zanella	Lab 2
30/10/2017	9:00 - 12:00	B	CS 1.4	Conti	
02/11/2017	14:15-17:15	C	CS 0.11	Conti	
06/11/2017	9:00 - 12:00	A	CS 0.2	Zanella	Lab 3
06/11/2017	9:00 - 12:00	B	CS 1.4	Conti	
09/11/2017	14:15-17:15	C	CS 0.11	Conti	
20/11/2017	9:00 - 12:00	A	CS 0.2	Zanella	Lab 4
20/11/2017	9:00 - 12:00	B	CS 1.4	Conti	
23/11/2017	14:15-17:15	C	CS 0.11	Conti	
27/11/2017	9:00 - 12:00	A	CS 0.2	Zanella	Lab 5
27/11/2017	9:00 - 12:00	B	CS 1.4	Conti	
30/11/2017	14:15-17:15	C	CS 0.11	Conti	
04/12/2017	9:00 - 12:00	A	CS 0.2	Zanella	Lab 6
04/12/2017	9:00 - 12:00	B	CS 1.4	Conti	
14/12/2017	14:15-17:15	C	CS 0.11	Conti	

Ricorsione

- La **ricorsione** è un approccio per la risoluzione di problemi che consiste nel riapplicare uno stesso algoritmo su un insieme di dati semplificato o suddiviso.
- Le soluzioni trovate per le chiamate ricorsive vengono poi combinate per formare la soluzione del problema iniziale.
- Una *funzione* è detta **ricorsiva** se richiama se stessa.
  Se due funzioni, invece, si chiamano l'un l'altra sono dette **mutuamente ricorsive**.
- Una funzione ricorsiva è formata da:
  - **Caso base**: è un caso particolare del problema che è possibile risolvere direttamente
  - **Passo ricorsivo:** la funzione viene richiamata passando un sottoinsieme dei dati originali.
- **L'obiettivo è quello di semplificare ad ogni passo ricorsivo i dati/il problema in modo da raggiungere un caso base.**
- La sequenza di chiamate ricorsive termina quando quella *annidata* (più interna) incontra uno dei casi basi, a questo punto si risale la sequenza ottenendo il risultato

Esempio Ricorsione: Fattoriale

- Funzione ricorsiva per calcolare il fattoriale di un numero:

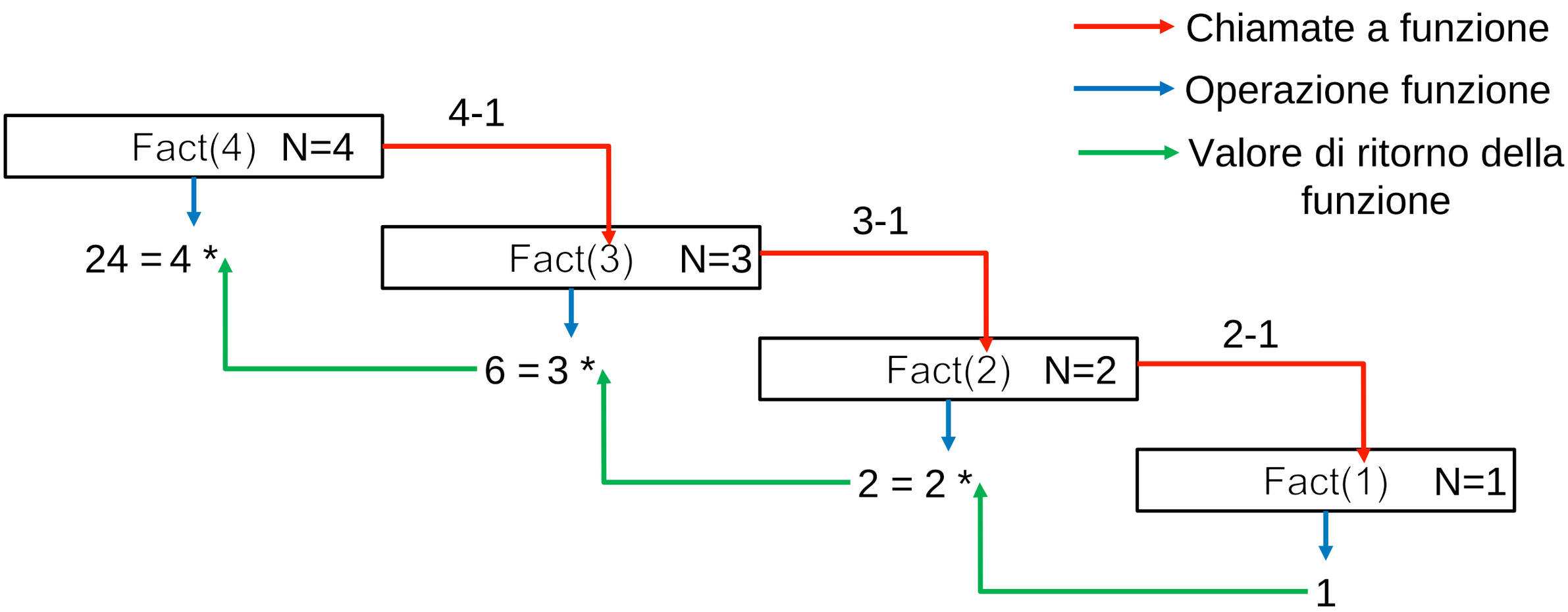
```
int fact(int a){
    if (n<=1)
        return 1;
    else
        return n*fact(n-1);
}
```

Caso base

Passo ricorsivo

- Cosa succede (chiamo `fact(4)`):
  1. In fact_1 n=4, non è il caso base e quindi per eseguire il calcolo `4*fact(3)`, la funzione chiama `fact(3)` e aspetta la sua terminazione.
  2. In fact_2 n=3, non è il caso base e quindi per eseguire il calcolo `3*fact(2)`, la funzione chiama `fact(2)` e aspetta la sua terminazione.
  3. In fact_3 n=2, non è il caso base e quindi per eseguire il calcolo `2*fact(1)`, la funzione chiama `fact(1)` e aspetta la sua terminazione.
  4. In fact_4 n=1, è il caso base, essa quindi ritorna subito il valore 1 a fact_3
  5. fact_3 riceve il valore 1, esegue il calcolo sospeso e ritorna 2 a fact_2
  6. fact_2 riceve il valore 2, esegue il calcolo sospeso e ritorna 6 a fact_3
  7. fact_1 riceve il valore 6, esegue il calcolo sospeso e ritorna 24.

Esempio Ricorsione: Fattoriale (cont'd)



Ricorsione

- Pro & Cons:
  - Problemi complessi -> si risolvono con poche righe di codice
  - Non è efficiente perché richiama molte volte una funzione e alloca sullo *stack* i parametri e le variabili ad ogni chiamata
  - Qualsiasi problema ricorsivo può essere svolto in modo *iterativo* (non ricorsivo), ma spesso quest'ultimo è molto più complesso.
    - Non bisognerebbe utilizzarla quando essa esegue a sua volta più di una chiamata ricorsiva

Esercizio 1: Somma per referenza

Si implementi una funzione che presi due numeri come parametri, li sommi e salvi il risultato in una terza variabile passata come parametro.

```
void my_sum(int n1, int n2, int* sum);
```

- **Hint:** definire la terza variabile come un *puntatore*

Esercizio 2: Array puntatore

Si implementi un programma che salvi n elementi in un array e li stampi a video scorrendo l'array come un puntatore.
Si utilizzi poi la funzione dell'esercizio 1 per calcolare la somma di tutti gli elementi dell'array.

Esercizio 3: Sorting by Pointer

Si implementi una funzione che ordini un array utilizzando i puntatori.

```
void sort_array(int* array);
```

Esercizio 4: File.txt

Si implementi un programma che scriva più linee inserite dall'utente nel file *test.txt* e ne successivamente ne legga il contenuto.

- Attenzione ad aprire e chiudere il file nelle due modalità!

Esercizio 5: Data.csv

Si implementi un programma che legga dal file *data.csv* le temperature massime e minime di alcune città italiane.
Poi trovi la più calda e la più fredda (considerando rispettivamente la massima e la minima).
Infine chieda all'utente di aggiungere gli stessi dati per la città nella quale si trova attualmente.

- Dichiarare le struct, typedef e define necessarie
- Prototipare le funzioni necessarie

Esercizio 6: Fibonacci

Si implementi una funzione ricorsiva che calcoli l'i-esimo numero di Fibonacci dato un numero i.

```
int fibonacci(int N);
```

- Il numero di Fibonacci F(n) di un numero n è la somma dei suoi numeridi Fibonacci precedenti.
- Più casi base
- Definiamo F(0) = 0
- Esempio:
  - Successione di Fibonacci: 1 1 2 3 5 8 13 ….
  - F(1) = 1
  - F(3) = 2
  - F(6) = 8

Esercizio 7: Strlen Ricorsiva

Si implementi una funzione ricorsiva che calcoli la lunghezza di una stringa.

```
int strle_ric(char* s);
```

Esercizio 8: Somma Ricorsiva

Si implementi una funzione ricorsiva che calcoli la somma degli elementi presenti in un array di interi

```
int somma_ric(int* pValori, int n);
```

- La funzione riceve il puntatore al primo elemento e il numero di elementi dell'array.

Esercizio 9: Stringa Inversa

Si implementi una funzione ricorsiva che stampi la stringa ricevuta come parametro, in ordine inverso.

```
void stampa_inversa(char* s);
```

Esercizio 10: Stringa Inversa

Si implementi una funzione ricorsiva `reverse` con prototipo:

```
void reverse(char* inversa, char* sorgente, int lung_sorgente);
```

che riempia la stringa puntata da "inversa" con i caratteri di "sorgente" nell'ordine opposto.

- Esempio: HELLO -> OLLEH