INFORMATICA A

A.A. 2019-20

Laboratorio n°5

Dott. Michele Zanella

Ing. Gian Enrico Conti

HEAP LAB

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:

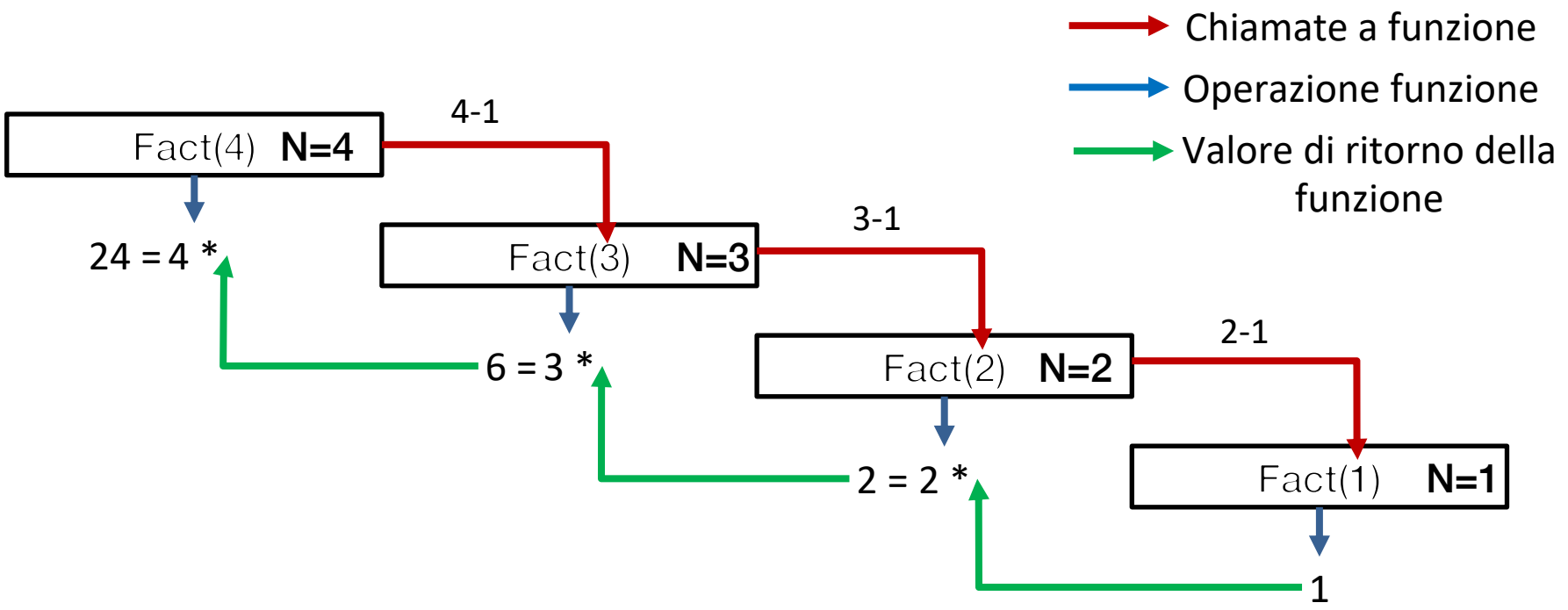
```
int fact(int a){
    if (n<=1)
        return 1;
    else
        return n*fact(n-1);
}
```

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

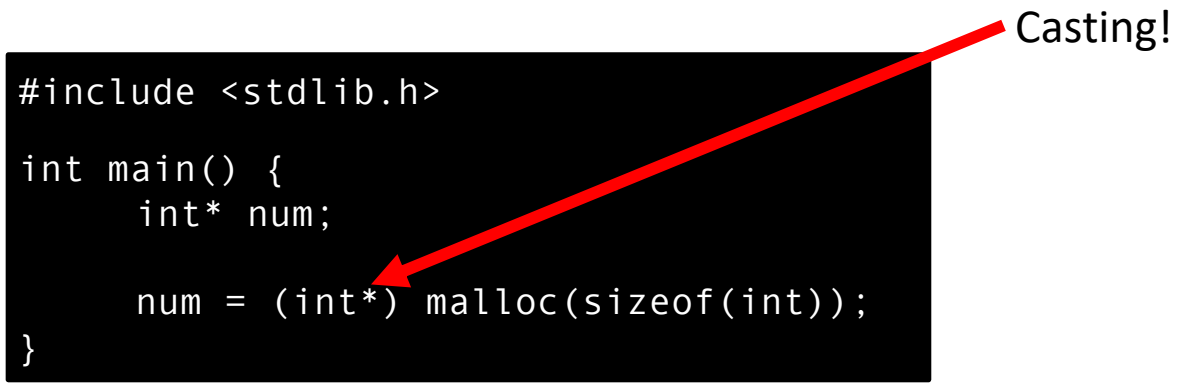
- Pro & Cons:
  - Problemi complessi -> si risolvono con poche righe di codice
  - Non è efficiente perché richiama molte volte una funzione e alloca sullo *stack* i parametri e le variabili ad ogni chiamata
  - Qualsiasi problema ricorsivo può essere svolto in modo *iterativo* (non ricorsivo), ma spesso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

Allocazione dinamica

- La memoria è divisa sostanzialmente in due parti:
  - STACK: contiene le variabili locali delle funzioni, gli argomenti passati e viene allocata quando viene eseguita una chiamata a funzione
  - HEAP: contiene le variabili allocate durante l'esecuzione del programma ("runtime"), la cui dimensione non è nota a priori.

Allocazione dinamica

- Funzioni utili, bisogna includere *stdlib.h*:
  - `malloc()`: permette di allocare sullo heap blocchi di Byte in numero e di dimensione a priori (i.e., in fase di compilazione) non noto.
  - `free()`: libera la memoria allocata
  - `realloc()`: modifica uno spazio di memoria precedentemente allocato



Esercizio 5.1: Stringa inversa (con RICORSIONE)

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

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

Esercizio 5.2: Strlen Ricorsiva

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

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

Esercizio 5.3: 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 5.4: Array dinamico

Si scriva un programma che permetta di inserire dei numeri interi in un array di dimensione decisa dall'utente

Hint: Allocare dinamicamente la memoria dell'array una volta saputa la dimensione dall'utente

Esercizio 5.5: Matrice dinamica

Si scriva un programma che permetta di creare una matrice dinamica con dimensione decisa dall'utente. Poi scali i valori della matrice per un fattore deciso dall'utente e stampi la matrice inserita e quella scalata.

Il programma deve accettare tramite il main dei parametri che corrispondono a: numero di righe, numero di colonne e fattore di scala.

Hint:

- Allocare dinamicamente la memoria della matrice una volta saputa la dimensione dall'utente.
- Verificare i parametri passati al momento dell'esecuzione
- Creare la funzione per stampare la matrice
- Creare la funzione per scalare la matrice
- Passare la matrice per referenza alle due funzioni

Esercizio a casa 5.1: Sudoku ricorsivo

Scrivere un programma per la soluzione RICORSIVA di un **Sudoku**.

https://1sudoku.com/play/sudoku-kids-free/sudoku-4x4/

Regole di base:

- 4 Regioni 2x2
- La regione va riempita con i numeri da 1 a 4
- Sulla stessa colonna e sulla stessa riga un numero deve comparire una sola volta
- All'interno della regione un numero deve comparire una sola volta
- Riempire le celle vuote
- La soluzione deve essere RICORSIVA

Hints:

- Utilizzare una matrice 4x4
- Identificare il caso base
- Identificare il passo iterativo

Esercizio a casa 5.1 : Sudoku ricorsivo (cont'd)

Organizzazione delle funzioni:

- `risolvi`: funzione ricorsiva che risolve il Sudoku
- `valido`: funzione che controlla se la configurazione è valida (e.g., nessuna ripetizione nelle righe, nelle colonne e nelle regioni)
- `completo`: funzione che controlla se la configurazione non contiene zeri (e.g., celle vuote)
- `controlla_riga`: funzione per controllare se la riga i-esima non contiene numeri >0 ripetuti
- `controlla_colonna`: funzione per controllare se la colonna i-esima non contiene numeri >0 ripetuti
- `controlla_regione`: funzione per controllare se la regione con riquadro in alto a sinistra `(i,j)` non ha ripetizioni

Esercizio a casa 5.2: Runtime Array

Si scriva un programma che permetta di inserire dei numeri interi in un array di dimensione calcolata runtime.

In questo caso l'utente può continuare ad aggiungere numeri ad ogni iterazione.

Hint: Reallocare la memoria dinamicamente ad ogni iterazione.