

# Linguaggio C++

marco.falda@unipd.it

## Ordinamento per inserzione

```
int a[5] = { 5, 2, 4, 1, 3 };

for (i = 1; i < 5; i++) {
    // memorizzo da parte il pivot
    tmp = a[i];
    m = i - 1;
    // scorro gli elementi precedenti al pivot
    while (m >= 0 && a[m] > tmp) {
        // se sono qui lo devo spostare
        a[m + 1] = a[m];
        m--;
    }
    // alla fine inserisco il pivot
    a[m + 1] = tmp;
}
```

## Le funzioni

- Permettono di organizzare e riutilizzare il codice
- Definizione:

```
tipo nome(param. formali)
```

  - param. formali:

```
tipo nome, ..., tipo nome
```
- Chiamata:

```
nome(param. attuali)
```

## Esempio - definizione

```
int Raddoppia(int num)
{
    int ris;
    ris = num * 2;
    return ris;
}
```

## Esempio - chiamata

```
int main( )
{
    int r, r1 = 3;

    r = Raddoppia(2);
    r = Raddoppia(r1);

    return 0;
}
```

## Ricorsione

- Una funzione può chiamare se stessa (e si dice ricorsiva)
  - ci deve essere una condizione di terminazione
- È il principio del “*Divide et impera*”
  - si decompone un problema complesso in sottoproblemi
  - si risolvono i casi di base

## Esempio – fattoriale

- Definizione:

$$\begin{cases} \text{Fatt}(0) = 1 \\ \text{Fatt}(n) = n * \text{Fatt}(n - 1) \end{cases}$$

caso base

caso ricorsivo

## Esempio – fattoriale

```
int Fatt(int n)
{
    // caso base
    if (n == 0)
        return 1;

    // caso ricorsivo
    return n * Fatt(n - 1);
}
```

## Esempio – serie di Fibonacci

■ Definizione:

$$\begin{cases} \text{Fib}_0 = 0 \\ \text{Fib}_1 = 1 \\ \text{Fib}_{n>1} = \text{Fib}_{n-1} + \text{Fib}_{n-2} \end{cases}$$

casi di base

caso ricorsivo

## Esempio – serie di Fibonacci

```
int Fib(int n)
{
    // caso base
    if (n < 2)
        return n;

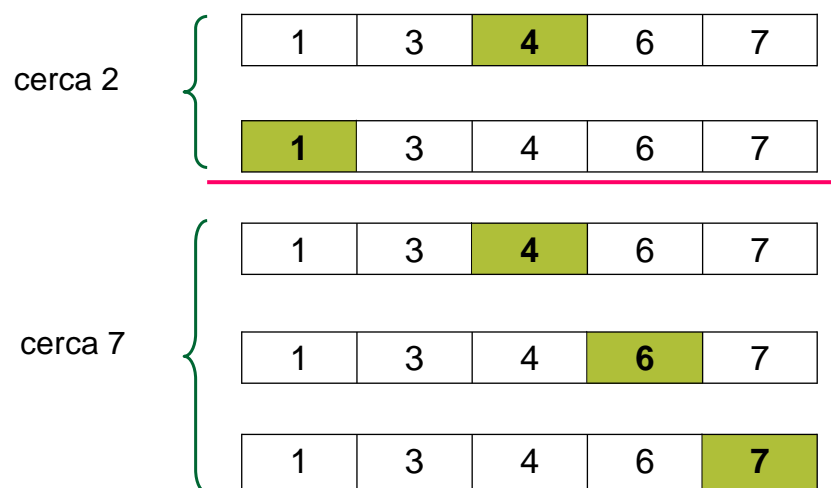
    // caso ricorsivo
    return Fib(n - 1) + Fib(n - 2);
}
```

## Esercizio

### ricerca binaria

- Scrivere un programma per cercare un numero in un *array* ordinato in questo modo:
  1. considera l'elemento al centro dell'*array*
  2. se è uguale all'elemento cercato bene
  3. se è minore cercalo ricorsivamente nella metà sinistra
  4. se è maggiore cercalo ricorsivamente nella metà destra

## Illustrazione



## Suggerimenti

- Per riferirsi all'*array*
  - usare una variabile globale
  - oppure passarlo specificando un parametro formale tipo nome[]
  
- Formula per l'elemento medio:
$$\text{mezzo} = \text{da} + (\text{a} - \text{da}) / 2$$
  
- Assicurarsi che  $\text{da} \leq \text{a}$ !
  - questo è un ulteriore caso base