



UNIVERSITÀ
DI TRENTO

Dipartimento di Ingegneria e
Scienza dell'Informazione
DISI - Trento

Programmazione 1

10 - Esercitazione

Andrea Mazzullo
andrea.mazzullo@unitn.it

Anno Accademico 2023/2024

Nelle puntate precedenti

```
1. char lettera = 'g'; ← variabile globale

2. void f() {
3.     char lettera = 'f'; ← variabile locale
4.     cout << lettera;    // f
5. }

6. int main() {
7.     f();
8.     cout << lettera;    // g
9.     char lettera = 'm'; ← variabile locale
10.    cout << lettera;    // m
11. }
```

Nelle puntate precedenti

- **Per valore**

copia il valore del parametro attuale

eventuali modifiche non si
riflettono sul parametro attuale

- **Per riferimento**

il parametro è un riferimento (&) al parametro attuale

- **Per puntatore**

il parametro è l'indirizzo del parametro attuale

passaggio per valore del
puntatore, ma ovviamente
si può modificare la
variabile puntata

Nelle puntate precedenti

Overloading è dare lo stesso nome a funzioni con diverso numero, ordine o tipo di parametri formali

```
int max(int numero1, int numero2);  
int max(int numero1, int numero2, int numero3);  
int max(char carattere1, char carattere2);  
...
```

Nelle puntate precedenti

Usati per fornire parametri opzionali con valori di default

```
int max(int n1, int n2, int n3=0, int n4=0, int n5=0);  
  
int main() {  
    cout << max(1,2,3);           // 3  
}  
  
int max(int n1, int n2, int n3, int n4, int n5) {  
    ...  
}
```

00 - Funzioni che Ritornano un Riferimento

```
int& max(int& x, int& y) {  
    return (x > y ? x : y);  
}
```

```
int main() {  
    int n1=44, n2=22;  
    max(n1, n2) = 55;  
}
```

00 - Ricorsione

```
int fib(int n) {  
    int returnValue;  
    if (n == 1 || n == 0) {  
        returnValue = 1;  
    }  
    else {  
        returnValue = fib(n-1) + fib(n-2);  
    }  
    return returnValue;  
}
```

00 - Ricorsione

```
int fib(int n) {  
    int returnValue;  
    if (n == 1 || n == 0) {  
        returnValue = 1;  
    }  
    else {  
        returnValue = fib(n-1) + fib(n-2);  
    }  
    return returnValue;  
}
```

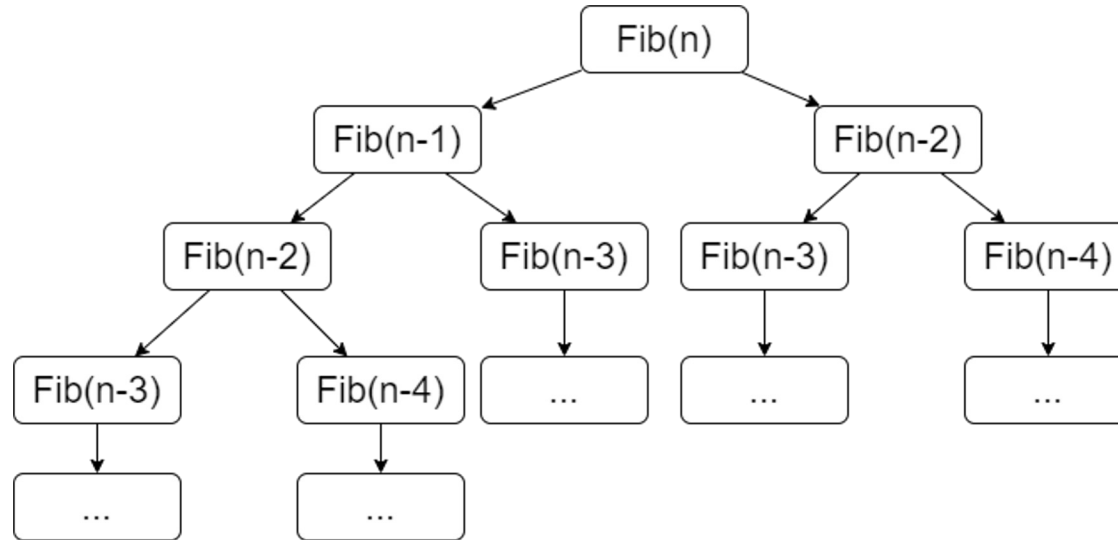
← 1. condizione di terminazione
(spesso chiamato “caso base”)

00 - Ricorsione

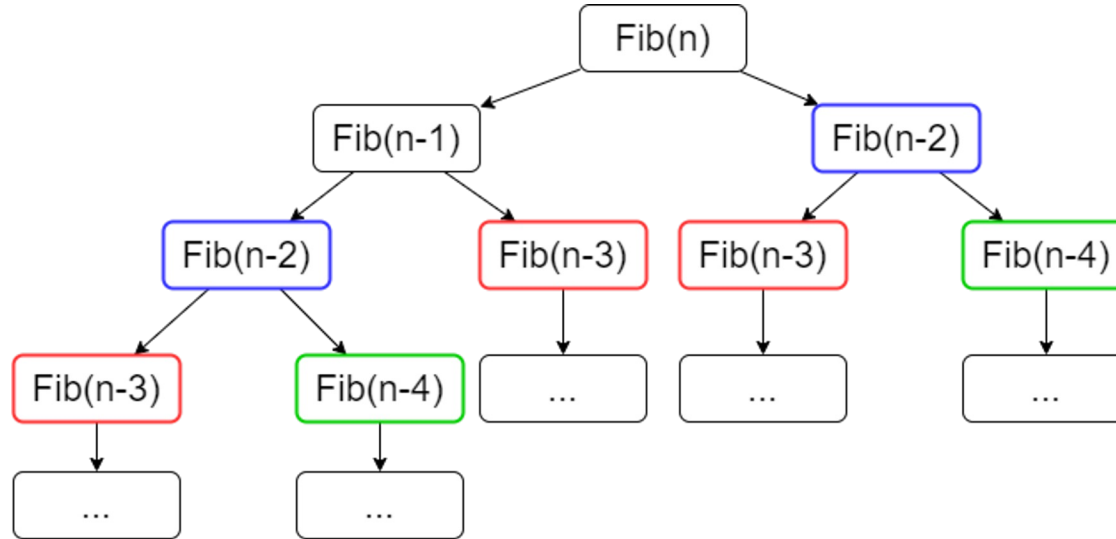
```
int fib(int n) {  
    int returnValue;  
    if (n == 1 || n == 0) {  
        returnValue = 1;  
    }  
    else {  
        returnValue = fib(n-1) + fib(n-2);  
    }  
    return returnValue;  
}
```

← 2. chiamate
ricorsive

00 - Ricorsione



00 - Ricorsione

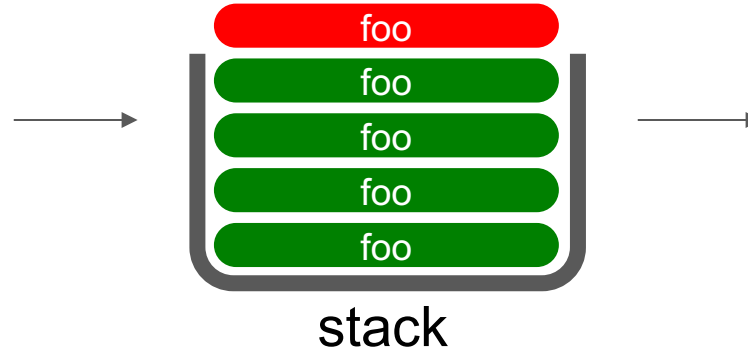


“Insidie” Computazionali

00 - Stack Overflow

```
int foo() {  
    return  
    foo();  
}
```

```
int main() {  
    return  
    foo();  
}
```



Segmentation fault
(core dumped)

00 - Algoritmi Ricorsivi

Alcuni algoritmi ricorsivi:

- [Ricerca binaria](#)
- [Massimo comune divisore](#)
- [MergeSort](#)
- [Algoritmo di Strassen](#) per la moltiplicazione
- ...

00 - Funzione tail-recursive

Una funzione è **tail-recursive** se la chiamata ricorsiva è ultima istruzione.

Funzioni tail-recursive possono *facilmente* essere trasformate in iterative

```
int mul(int n, int cont, int value = 0) {  
    if (cont == 0) {  
        return value;  
    }  
    return mul(n, cont - 1, value + n);  
}
```

00 - Funzioni Mutuamente Ricorsive

```
bool even(int n) {  
    bool isEven;  
    if (n == 0) {  
        isEven = true;  
    }  
    else {  
        isEven = odd(n-1);  
    }  
    return isEven;  
}
```

```
bool odd(int n) {  
    bool isOdd;  
    if (n == 0) {  
        isOdd = false;  
    }  
    else {  
        isOdd = even(n-1);  
    }  
    return isOdd;  
}
```

00 - Funzioni Mutuamente Ricorsive

```
bool even(int n) {  
    bool isEven;  
    if (n == 0) {  
        isEven = true;  
    }  
    else {  
        isEven = odd(n-1);  
    }  
    return isEven;  
}
```

← condizione di
terminazione →

```
bool odd(int n) {  
    bool isOdd;  
    if (n == 0) {  
        isOdd = false;  
    }  
    else {  
        isOdd = even(n-1);  
    }  
    return isOdd;  
}
```


00 - Funzioni Mutuamente Ricorsive

```
bool even(int n) {  
    bool isEven;  
    if (n == 0) {  
        isEven = true;  
    }  
    else {  
        isEven = odd(n-1);  
    }  
    return isEven;  
}
```

← chiamate
ricorsive →

```
bool odd(int n) {  
    bool isOdd;  
    if (n == 0) {  
        isOdd = false;  
    }  
    else {  
        isOdd = even(n-1);  
    }  
    return isOdd;  
}
```

01 - Fattoriale

Scrivere un programma che calcoli il fattoriale di un intero positivo tramite funzione ricorsiva.

$$n! := \prod_{k=1}^n k = 1 \cdot 2 \cdot 3 \cdots (n - 1) \cdot n$$

02 - Divisione

**Scrivere un programma che calcoli la divisione di un intero
tramite funzione ricorsiva.**

03 - Stampa Binario

Scrivere un programma che prenda in input un numero e ne stampi a video la rappresentazione binaria tramite procedura ricorsiva.

345 => 101011001

04 - Palindromo

**Scrivere un programma che, preso in input un numero intero, controlli se il numero è un palindromo.
Scrivere una funzione ricorsiva che faccia il controllo.**

Esercizi Aggiuntivi

05 - Somma delle cifre

Scrivere una funzione ricorsiva che, dato in input un numero intero, ritorni la somma delle cifre che compongono il numero intero.

$$25 \Rightarrow 7$$

05 - Stampa caratteri

Scrivere un programma che prenda in input due caratteri e stampi a video tutti i caratteri compresi tramite procedura ricorsiva.

$(a, g) \Rightarrow \text{"a, b, c, d, e, f, g"}$