UNIVERSITÀ
DI TRENTO

Dipartimento di Ingegneria e
Scienza dell'Informazione
DISI - Trento

Programmazione 1

11 - Esercitazione

Giovanni De Toni
giovanni.detoni@unitn.it

Attenzione

La presente esercitazione verrà trasmessa via Zoom. Essa verrà anche registrata e successivamente messa a disposizione degli studenti dell'Università degli Studi di Trento. Per gli utenti connessi attraverso Zoom, in caso non desideriate per qualunque motivo essere registrati, siete pregati di effettuare la disconnessione ora. La lezione sarà comunque visionabile in modo asincrono.

Anno Accademico 2021/2022

Nelle puntate precedenti

```
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”)

2. chiamate ricorsive

Nelle puntate precedenti

Funzioni Ricorsive

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

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

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

01 - Potenza ricorsiva

Scrivere una funzione che, dato in input due interi, base ed esponente, calcoli la potenza in modo ricorsivo.

02 - Massimo Comune Divisore

Scrivere un programma che prenda in input due interi e calcoli il Massimo Comune Divisore (MCD) tramite funzione ricorsiva usando l'algoritmo di Euclide.

```
MCD(m, n) = MCD(n, m % n), given that m > n
```

```
345
        =>    3
543
```

03 - Stampa caratteri

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

```
(a, g)=> "a,b,c,d,e,f,g"
```

04 - Fattorizzazione in Primi

Scrivere una procedura che fattorizzi in primi un dato numero intero tramite ricorsione. E' possibile definire delle funzioni ausiliarie (e.g., una funzione che computi se un numero è primo). Se possibile, le funzioni ausiliarie devono essere ricorsive loro stesse.

05 - Torre di Hanoi

Scrivere un programma che risolva la Torre di Hanoi tramite procedura ricorsiva.

05 - Torre di Hanoi

Scrivere un programma che risolva la Torre di Hanoi tramite procedura ricorsiva.

- Assegnate ad ogni disco X un numero, partendo dal disco più piccolo (e.g., il disco più piccolo sarà 1, il secondo 2, etc.)
- Dato un disco X, il pattern di movimento che farà sarà sempre costante e dipende se il numero è pari o dispari.
  - Dischi “dispari”: `A -> B -> C -> A`;
  - Dischi “pari”: `A -> C -> B -> A`;
- Per ogni iterazione, muovere sempre il disco più piccolo che non è stato mosso l’iterazione successiva.

05 - Torre di Hanoi

Scrivere un programma che risolva la Torre di Hanoi tramite procedura ricorsiva.

```
FUNZIONE MoveTower(disk, source, dest, spare):
IF disk == 0, THEN:
    move disk from source to dest
ELSE:
    MoveTower(disk - 1, source, spare, dest)   // Step 1
    move disk from source to dest              // Step 2
    MoveTower(disk - 1, spare, dest, source)   // Step 3
END IF
```

05 - Torre di Hanoi

Scrivere un programma che risolva la Torre di Hanoi tramite procedura ricorsiva.

```
FUNZIONE MoveTower(disk, source, dest, spare):
IF disk == 0, THEN:
    move disk from source to dest
ELSE:
    MoveTower(disk - 1, source, spare, dest)   // Step 1
    move disk from source to dest              // Step 2
    MoveTower(disk - 1, spare, dest, source)   // Step 3
END IF
```

$$\text{minimo numero di mosse} = 2^n - 1$$