UNIVERSITÀ DEGLI STUDI DI BERGAMO

Informatica
Modulo di Programmazione

Informatica
Modulo di Programmazione
Recursive Functions

mauro.pelucchi@gmail.com
Mauro Pelucchi

2023/2024

Agenda

- Recursive Functions

Recursion

- Recursive functions
  - Functions that call themselves
  - Can only solve a base case
- If not base case
  - Break problem into smaller problem(s)
  - Launch new copy of function to work on the smaller problem (recursive call/recursive step)
    - Slowly converges towards base case
    - Function makes call to itself inside the return statement
  - Eventually base case gets solved
    - Answer works way back up, solves entire problem

Recursion

- Example: factorial

$$n! = n * (n - 1) * (n - 2) * \ldots * 1$$

  - Recursive relationship (n! = n * (n – 1)!)

$$5! = 5 * 4!$$

$$4! = 4 * 3! \ldots$$

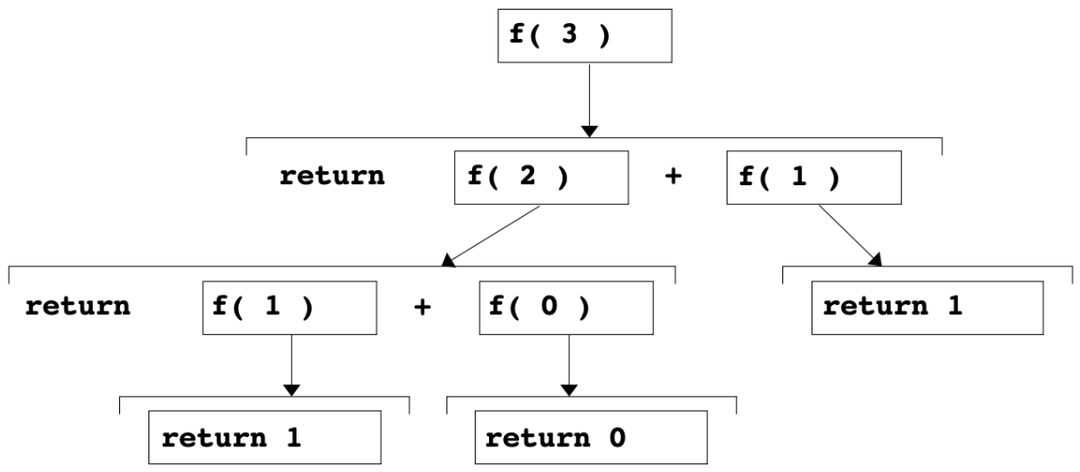
  - Base case (1! = 0! = 1)

Recursion

- Fibonacci series: 0, 1, 1, 2, 3, 5, 8...
  - Each number sum of two previous ones
  - Example of a recursive formula:
    - *fib(n) = fib(n-1) + fib(n-2)*
- C++ code for Fibonacci function

```
long fibonacci( long n )
{
   if ( n == 0 || n == 1 )  // base case
       return n;
   else
    return fibonacci( n - 1 ) +
           fibonacci( n - 2 );
}
```

Recursion



- Order of operations
  - `return fibonacci( n - 1 ) + fibonacci( n - 2 );`
- Recursive function calls
  - Each level of recursion doubles the number of function calls
    - 30th number = 2^30 ~ 4 billion function calls
  - Exponential complexity

Recursion vs. Iteration

- Repetition
  - Iteration: explicit loop
  - Recursion: repeated function calls
- Termination
  - Iteration: loop condition fails
  - Recursion: base case recognized
- Both can have infinite loops
- Balance between performance (iteration) and good software engineering (recursion)

Esercizio 1

Scrivere un programma C che, dato un numero calcola la somma dei primi N numeri pari positivi in maniera ricorsiva.

Specifica Liv 1:

La somma dei primi N numeri pari è data dalla seguente,

$$S_N = 2*1 + 2*2 + 2*3 + \ldots + 2*i + \ldots + 2*(N-1) + 2*N.$$

Specifica Liv 2:

se N =1, S_N = 2, **(CASO BASE)**

se N >1, S_N = 2 * N + S_{N-1} **(PASSO INDUTTIVO)**

(somma dell'N-esimo numero pari +
la somma dei primi N-1 numeri pari.)

```
int somma_pari(int N){
   if (N <= 1)
     return 2;
   else
     return 2*N + somma_pari(N-1);
}
```

Esercizio 2

Scrivere una funzione che calcoli il valore M^N, dove M è un numero in virgola mobile e N un numero intero.

1mo caso:
Hp: M intero, N intero con N>=0

Specifica Liv 1:

M^N = M*M*M*…*M; N volte.

Specifica Liv 2:

Se N = 0, allora $M^N = 1$ (caso base)
Se N è pari, allora $M^N = (M^{N/2})^2$ (passo induttivo)
Se N è dispari, allora $M^N = (M^{N-1})*M$ (passo induttivo)

per considerare anche il caso di potenze negative cosa occorrerebbe fare?

Esercizio 3

Scrivere una funzione ricorsiva

void BoomBang(int k)

che stampa k volte la stringa “Boom” seguita dalla stampa della stringa “Bang” anch’essa k volte.

Esempio:
BoomBang(3)
output: Boom Boom Boom Bang Bang Bang

Esercizio 4

Scrivere un sottoprogramma che restituisca il valore massimo di un vettore di interi con procedimento ricorsivo.

Specifica Liv 1:
Si pensi di assegnare il primo elemento dell'*array* come massimo e confrontarlo con tutti gli altri cambiando il valore del massimo se questo è minore della cella corrente del vettore.

Specifica Liv 2:
Detta `N` la lunghezza del vettore `array`

```
Se N = 1, allora max = array[0]              (caso base)
```

Se `N > 1`, allora il massimo del vettore di `N` elementi (max) sarà uguale al risultato del confronto tra `array[0]` e il massimo degli elementi del sotto-vettore che che va da `array[1]` a `array[N]`. (N.B.: quindi è lungo `N-1`) (passo induttivo)

Esercizio 5

Scrivere un sottoprogramma che inverta un vettore di interi con procedimento ricorsivo.

Esercizio 6

Scrivere un sottoprogramma ricorsivo che stabilisca se una stringa passata come parametro contiene una parola palindroma. In caso affermativo il sottoprogramma restituisce 1, altrimenti restituisce 0.

Esercizio 7

Calcolo del coefficiente binomiale
Scrivere un programma C che stampi sullo standard output tutti i valori del triangolo di Tartaglia per un certo ordine N, utilizzando una funzione ricorsiva per il calcolo dei coefficienti binomiali, avente il seguente prototipo:

```
int cobin(int n, int k);
```

La costruzione del triangolo di Tartaglia è mostrata di seguito.
Si ricorda inoltre che ognuno dei coefficienti del triangolo prende il nome di coefficiente binomiale `cobin(n, k)`

```
k= 0  1  2   3   4   5
   1                          n = 0
   1  1                       n = 1
   1  2  1                    n = 2
   1  3   3   1               n = 3
  1  4   6   4   1            n = 4
  1  5   10  10  5   1        n = 5
.....................................
```

Esercizio 8
Ricerca Dicotomica

- Si scriva un sottoprogramma C che effettui la ricerca di un intero in un vettore ordinato (in senso crescente) in maniera ricorsiva, restituendo l'indice della prima occorrenza nel vettore del valore cercato.