Il problema della torre di Hanoi

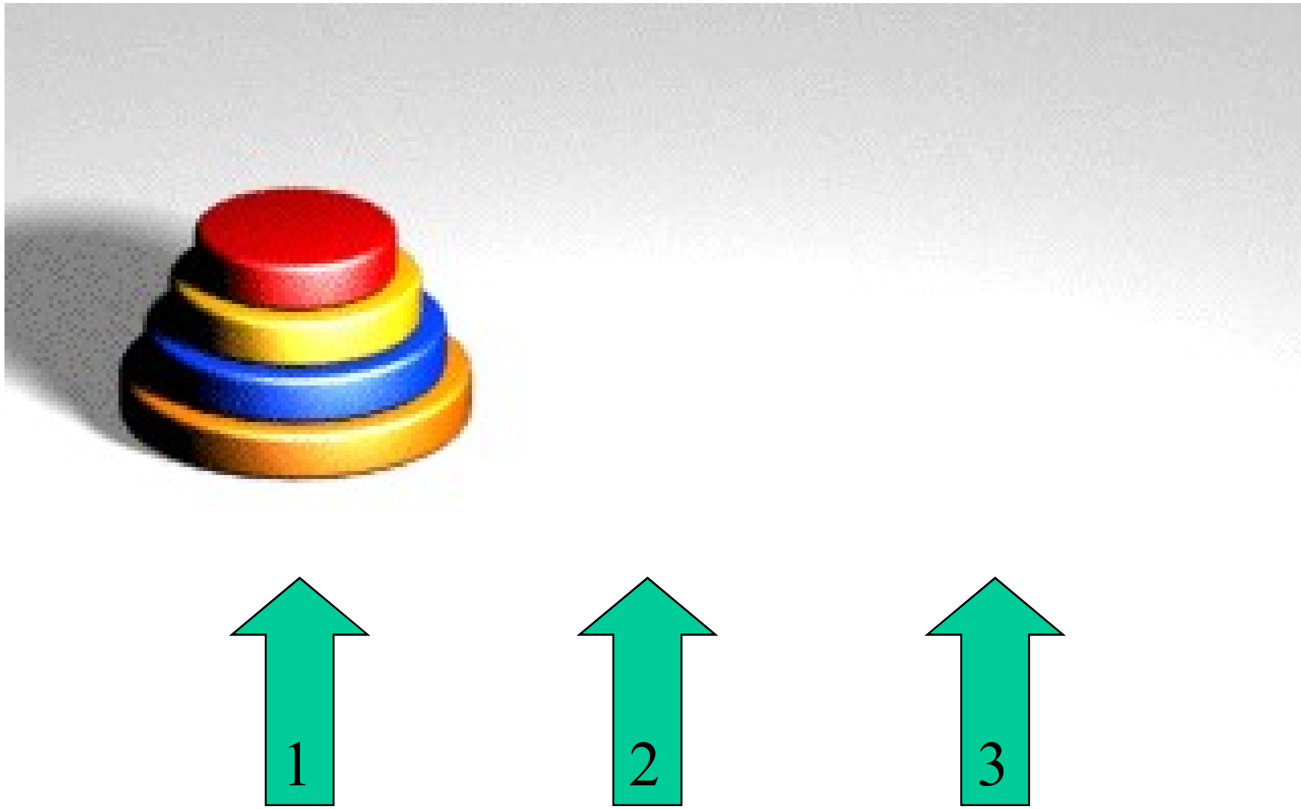
Nel tempio Indù di Brahma, ad Hanoi, in Vietnam, alcuni monaci sono costantemente impegnati a spostare 64 dischi d'oro su tre colonne di diamante. La leggenda narra che quando i monaci completeranno il lavoro, il mondo finirà.

Poiché il numero di spostamenti per i 64 dischi è pari a 2^{64}-1 (ovvero 18.446.744.073.709.551.615), supponendo che i monaci impieghino un secondo per ogni mossa, il mondo finirà tra 585 miliardi di anni !

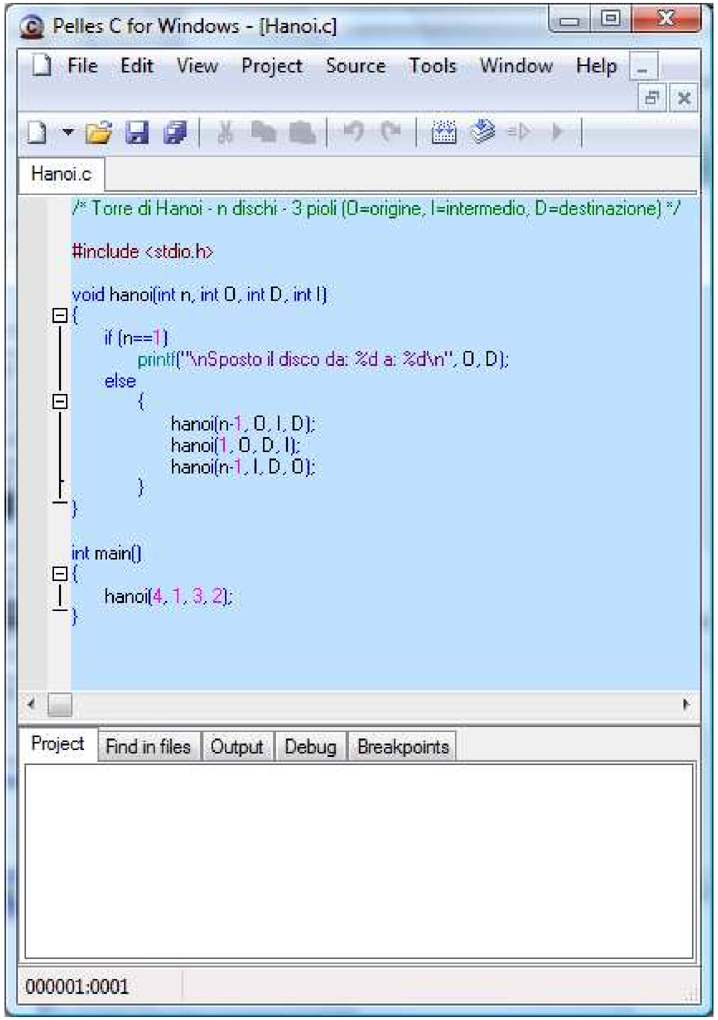
(Edouard Lucas, matematico francese, 1883)

Il problema della torre di Hanoi

La soluzione con 4 dischi su 3 pioli



Torre di Hanoi



HANOI.c -> IL LISTATO IN C CHE IMPLEMENTA L’ALGORITMO RICORSIVO PER RISOLVERE IL PROBLEMA DELLA TORRE DI HANOI CON 4 DISCHI E 3 PIOLI

Torre di Hanoi: risultato esecuzione codice C

```
C:\Users\andrea sergiacomi>"C:
0\listati\03\Hanoi.exe"

Sposto il disco da: 1 a: 2

Sposto il disco da: 1 a: 3

Sposto il disco da: 2 a: 3

Sposto il disco da: 1 a: 2

Sposto il disco da: 3 a: 1

Sposto il disco da: 3 a: 2

Sposto il disco da: 1 a: 2

Sposto il disco da: 1 a: 3

Sposto il disco da: 2 a: 3

Sposto il disco da: 2 a: 1

Sposto il disco da: 3 a: 1

Sposto il disco da: 2 a: 3

Sposto il disco da: 1 a: 2

Sposto il disco da: 1 a: 3

Sposto il disco da: 2 a: 3

C:\Users\andrea sergiacomi>
```

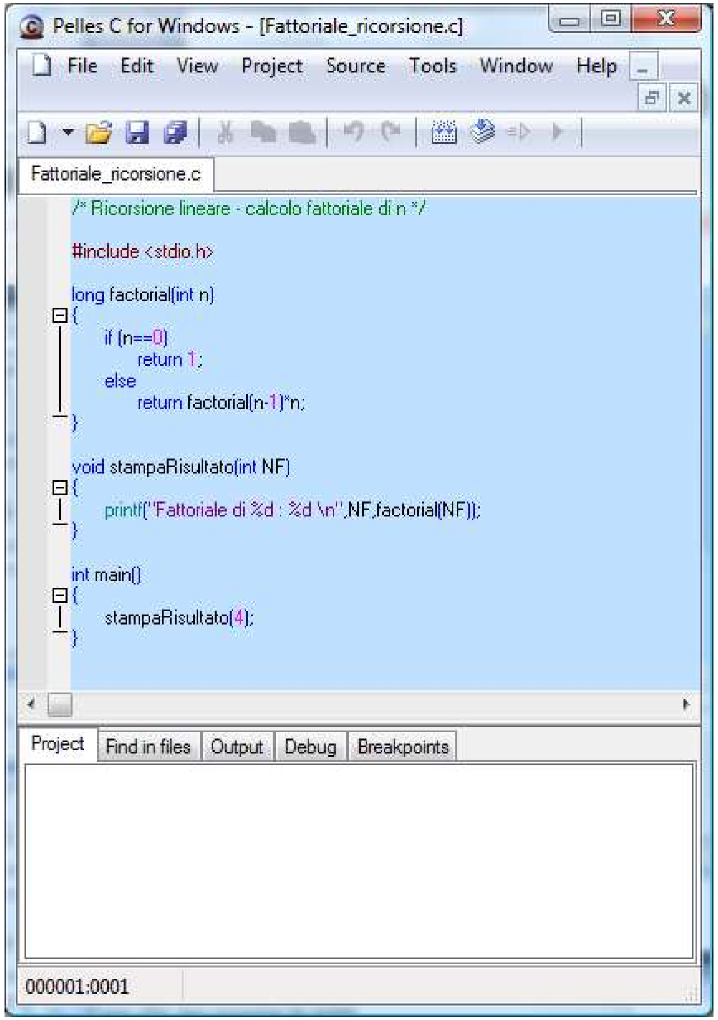
La funzione fattoriale

Per n numero naturale positivo **n!** è il prodotto dei numeri naturali positivi compresi tra 1 ad n

(per convenzione la funzione si estende anche allo 0, assumendo 0! = 1)

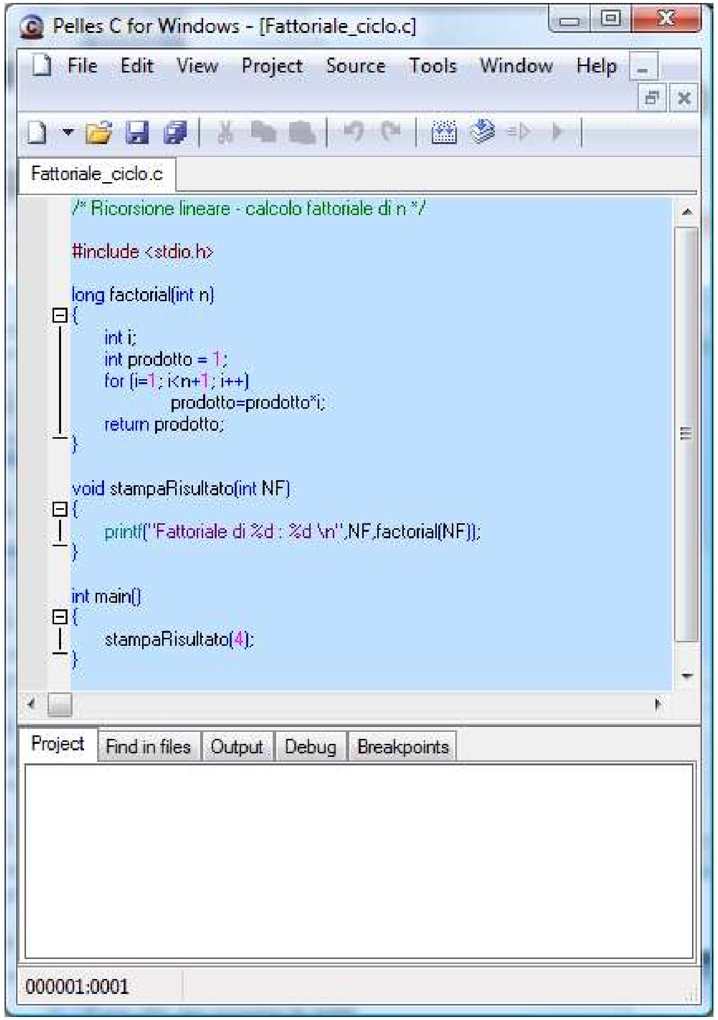
per n >= 1 è verificato che n! = n * (n – 1)!

Calcolo Fattoriale



FATTORIALE_RICORSIONE.c -> IL LISTATO IN C CHE IMPLEMENTA L'ALGORITMO **RICORSIVO LINEARE** (una sola chiamata a se stesso) **NON IN CODA** (la chiamata ricorsiva non è l'ultima azione eseguita dalla funzione stessa) PER CALCOLARE IL FATTORIALE DEL NUMERO 4

Calcolo Fattoriale



FATTORIALE_CICLO.c -> IL LISTATO IN C CHE IMPLEMENTA L'ALGORITMO **ITERATIVO** PER CALCOLARE IL FATTORIALE DEL NUMERO 4

Si noti che entrambe le versioni con cui abbiamo implementato l'algoritmo di calcolo del fattoriale (iterativa e ricorsiva) hanno **complessità computazionale analoga** (funzione della grandezza di NF, il numero passato per argomento).

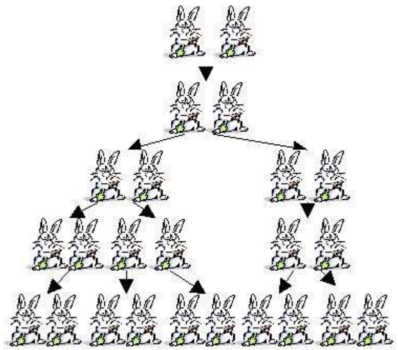
La ricorsione ha un vantaggio fondamentale: permette di scrivere **poche linee di codice** per risolvere un problema anche molto complesso.

Tuttavia, essa ha anche un enorme svantaggio: le **prestazioni**. Infatti, la ricorsione genera una quantità enorme di overhead, occupando lo stack per un numero di istanze pari alle chiamate della funzione che è necessario effettuare per risolvere il problema. Funzioni che occupano una grossa quantità di **spazio in memoria**, pur potendo essere implementate ricorsivamente, potrebbero dare problemi con riferimento al **tempo di esecuzione**. Inoltre, oltre alla RAM, la ricorsione impegna comunque il **processore** in maniera maggiore, per riuscire a popolare e a distruggere gli stack.

La successione di Fibonacci

è una successione F(n) di numeri interi naturali tale che:
F(0)= 0 , F(1)= 1 ,
per ogni successivo numero >1 F(n)= F(n-1) + F(n-2)

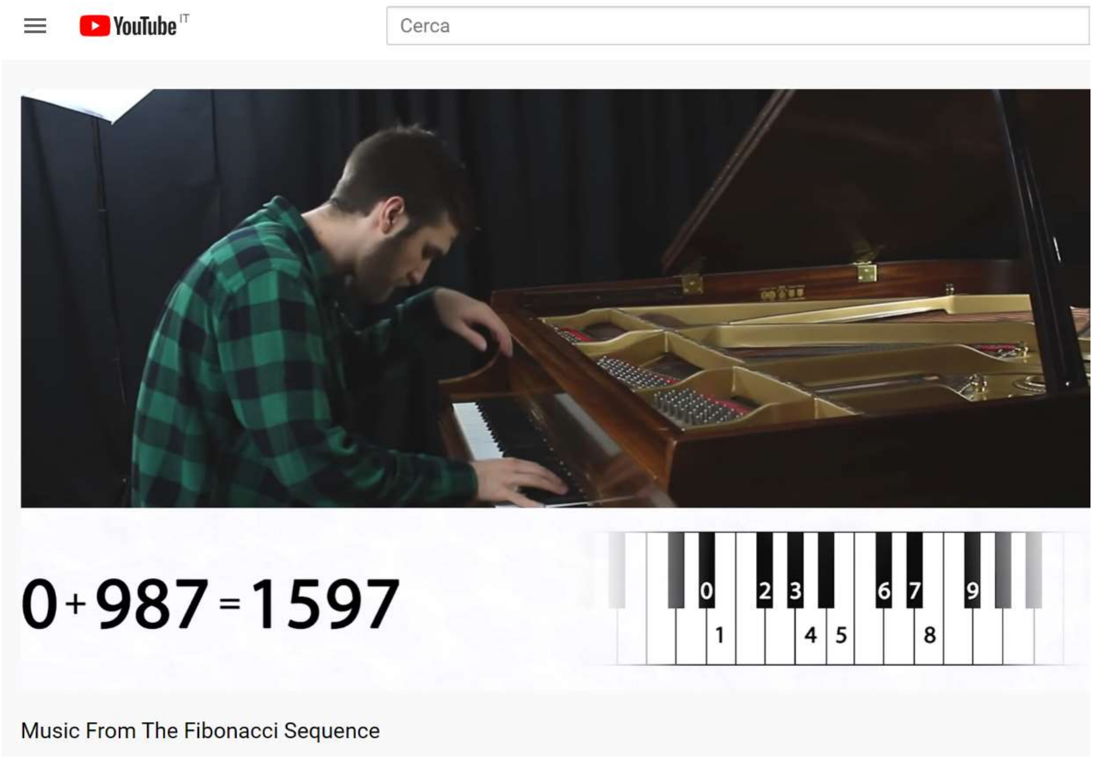
La sequenza prende il nome da Leonardo Fibonacci, matematico pisano del 13^ secolo, il cui intento fu di trovare una legge che descrivesse la crescita di una popolazione di conigli.
La successione descrive il numero di coppie di conigli in ogni mese assumendo: che la prima coppia diventi fertile al compimento del primo mese e dia alla luce una nuova coppia al compimento del secondo mese e che le nuove coppie nate si comportino in modo analogo.



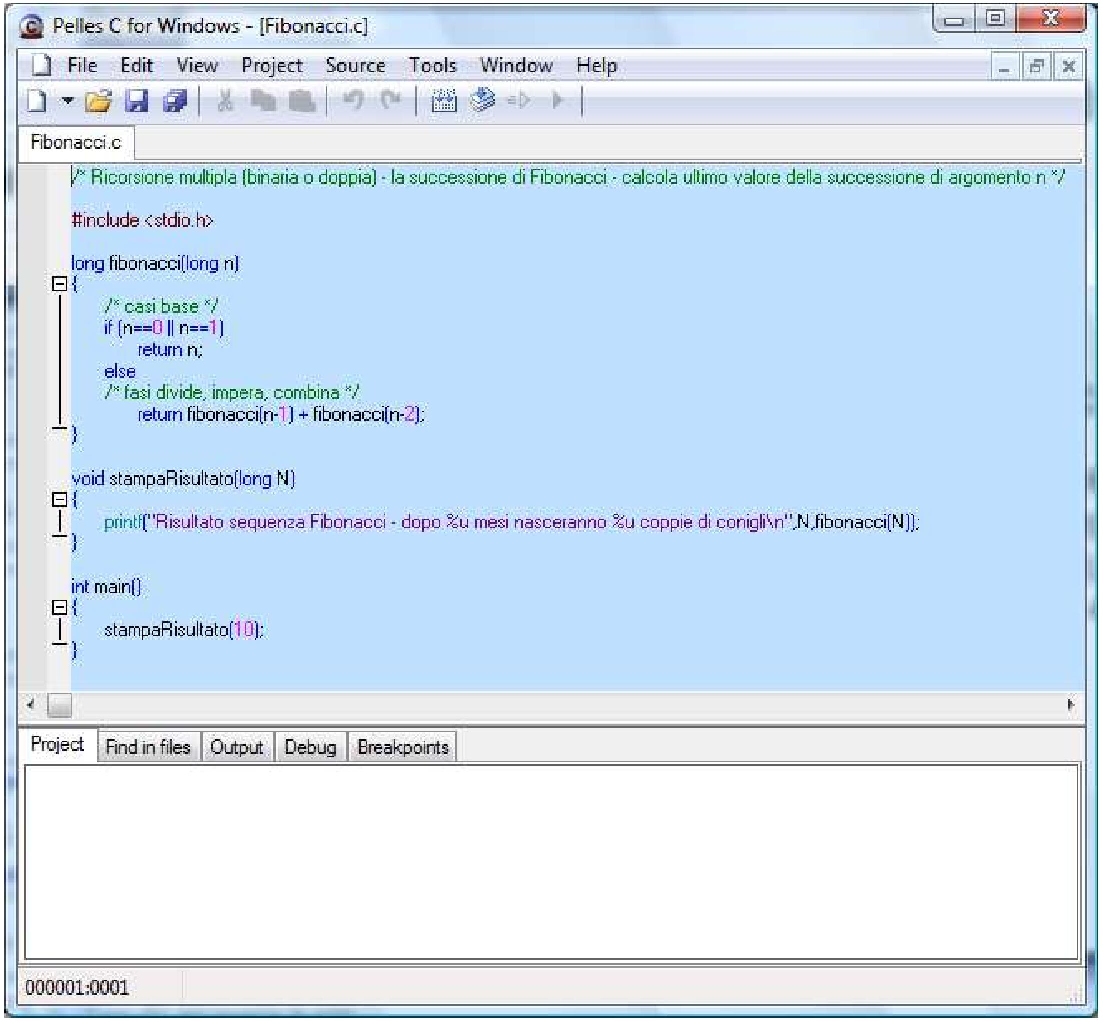
I primi 11 numeri di Fibonacci sono:
0, 1, 1, 2, 3, 5, 8, 13, 21, 34, 55 (=F10)

La successione di Fibonacci

Music from the Fibonacci Sequence
https://www.youtube.com/watch?v=IGJeGOw8TzQ



Funzione di Fibonacci



FIBONACCI.c -> IL LISTATO IN C CHE IMPLEMENTA L'ALGORITMO **RICORSIVO MULTIPLO** (più chiamate a se stesso) DI FIBONACCI F(N) PER N = 10

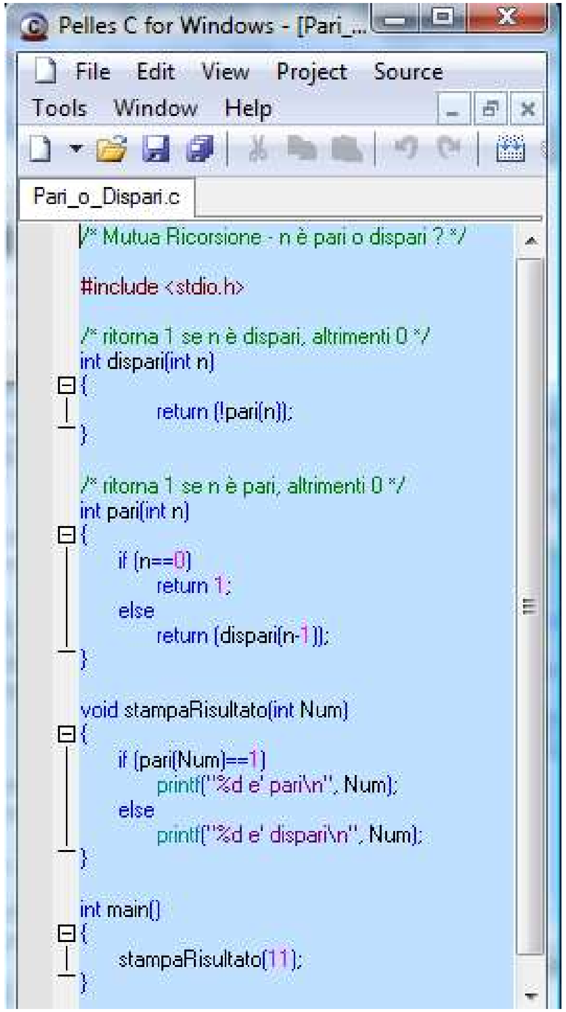
Pari o Dispari ?

Per determinare in modo ricorsivo se un numero è pari o dispari introduciamo due funzioni omonime, definite l’una in termini dell’altra.

Pari (n) è vera per n=0 o se Dispari (n) è falsa

Dispari (n) è vera se pari (n) è falsa

Pari o Dispari ?



PARI_O_DISPARI.c -> IL LISTATO IN C CHE IMPLEMENTA L'ALGORITMO A **MUTUA RICORSIONE** (la prima funzione ne chiama una seconda che a sua volta richiama la prima) PER DETERMINARE SE IL NUMERO IN INPUT E' PARI O DISPARI.

C standard ci permette di usare valori int come fossero booleani (intendendo FALSE == 0 e TRUE ogni numero diverso da 0, nel ns. caso == 1).

Curiosità: un metodo alternativo (non ricorsivo), più efficiente in termini di complessità computazionale [**Θ**(1)], consisterebbe nel dividere il numero intero, avuto in input, per 2 utilizzando l'operatore MOD (n%2):

- se il risultato è 0 il numero è pari
- se il risultato è >0 (ovvero c'è un resto decimale) il numero è dispari

La funzione di Ackermann #1

La **funzione di Ackermann** è una funzione *f(x,y,z)* che ha come dominio l'insieme delle terne di numeri naturali e come codominio i numeri naturali. Essa è definita per ricorrenza nel seguente modo:

$$A(m,n) = \begin{cases} n+1 & \text{se } m = 0 \\ A(m-1,1) & \text{se } m > 0 \text{ e } n = 0 \\ A(m-1, A(m,n-1)) & \text{se } m > 0 \text{ e } n > 0. \end{cases}$$

oppure:

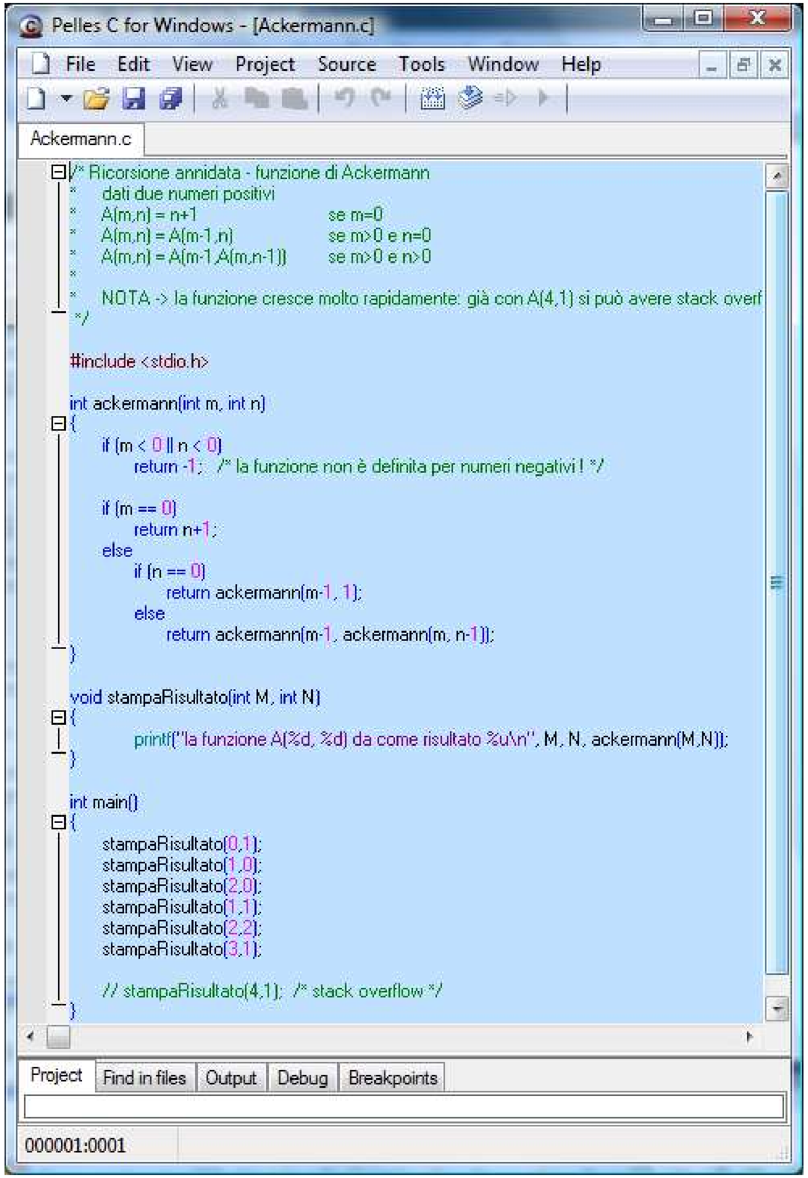
$$f(0,0,z) = z$$
$$f(0,y+1,z) = f(0,y,z) + 1$$
$$f(1,0,z) = 0$$
$$f(x+2,0,z) = 1$$
$$f(x+1,y+1,z) = f(x,[f(x+1,y,z)],z)$$

La funzione di Ackermann #2

La funzione di Ackermann si può rappresentare come una famiglia di funzioni definite al variare di un parametro individuato dalla prima variabile. Per ogni valore del parametro si ha una funzione che è ottenuta iterando la funzione precedente per un numero di volte individuato dalla seconda variabile.

In quest'ottica le prime funzioni della famiglia sono funzioni familiari come l'addizione, la moltiplicazione e la potenza, e successivamente si hanno funzioni sempre più complesse.
E' una funzione che cresce più velocemente di qualsiasi funzione ricorsiva primitiva.
Il meccanismo di calcolo della funzione è estremamente semplice quanto pesante dal punto di vista computazionale.
Risulta quindi una funzione con una complessità estremamente elevata anche per valori di input semplici.

Funzione di Ackermann



ACKERMANN.c -> IL LISTATO IN C CHE IMPLEMENTA L'ALGORITMO DI ACKERMANN A **RICORSIONE INNESTATA** (la funzione ha come argomento una chiamata alla funzione stessa). VENGONO RAPPRESENTATI I DIVERSI OUTPUT PER DIVERSE COPPIE DI INPUT M ed N