Esercizi sulla ricorsione

Docente: Violetta Lonati

PAS classe A042
Corso introduttivo pluridisciplinare in Informatica

1 Numeri di Fibonacci

I numeri di Fibonacci furono introdotti da Leonardo di Pisa, detto Fibonacci, per descrivere la crescita di una popolazione di conigli (sebbene in modo idealizzato e non plausibile dal punto di vista biologico). Per calcolare il numero di coppie di conigli nella popolazione all'n-esimo mese, egli fece le seguenti assunzioni: al primo mese, c'è una sola coppia di conigli; le coppie neonate diventano fertili a partire dal secondo mese di vita; ogni mese le coppie fertili generano una nuova coppia ciascuna; i conigli non muoiono mai.

Se traduciamo queste ipotesi in formule, chiamato F_n il numero di copppie di conigli presenti all'n-esimo mese, avremo $F_1 = F_2 = 1$ e, per i mesi successivi, $F_n = F_{n-1} + F_{n-2}$, infatti ad ogni mese avremo le coppie del mese precedente, più le nuove coppie generate dalle coppie nate due mesi prima. Il numero di coppie di conigli cresce come $1, 1, 2, 3, 5, 8, 13, 21, \ldots$.

Scrivete un programma (chiamato `fibonacci`) che, basandosi su una funzione ricorsiva, calcoli il valore di F_n, dove n è letto come primo passo del programma.

Esempio di funzionamento

```
4 3
```

Esempio di funzionamento

```
7 13
```

Modificate il programma perchè, oltre a stampare F_n, stampi anche il numero di volte che la funzione ricorsiva viene chiamata per calcolare tale valore.

Scrivete una versione del programma che non usi la ricorsione. Prima di impostarlo, ponetevi questa domanda: di quante variabili ho effettivamente bisogno?

2 Torri di Hanoi

Il gioco delle torri di Hanoi (anche detto delle torri di Brahma) è stato inventato nel 1883 dal matematico francese Edouard Lucas. L'obiettivo è spostare da un paletto ad un altro un certo numero di dischi forati di dimensione crescente infilati sul paletto e appoggiati l'uno sull'altro. Le regole del gioco sono che si può spostare solo il disco che è in cima ad una pila e non si deve mai appoggiare un disco di dimensione più grande sopra uno più piccolo; per aiutarsi, è possibile usare un terzo paletto come appoggio ausiliario. Secondo una leggenda (forse inventata dal matematico stesso) alcuni monaci di un tempio Indù sono costantemente impegnati a spostare sessantaquattro dischi secondo le regole del gioco; la leggenda dice che quando i monaci completeranno il lavoro, il mondo finirà!

Obiettivo dell'esercizio è scrivere un programma in grado di giocare al gioco delle torri di Hanoi, ossia di specificare la sequenza di mosse da effettuare per risolvere il rompicapo data l'altezza $n > 0$ della pila. Potete

assumere che i tre paletti siano numerati da 0 a 2 e che gli n dischi siano inizialmente impilati dal più piccolo (in cima alla pila) al più grande (sotto tutta la pila) sul paletto 0 e vadano spostati al paletto 2. Per semplicità, le mosse sono date semplicemente dall'indicazione del paletto da e verso cui si deve muovere il disco.

Ad esempio, una soluzione per una pila di altezza 3 è data dalla seguente sequenza di mosse:

```
0 -> 2
0 -> 1
2 -> 1
0 -> 2
1 -> 0
1 -> 2
0 -> 2
```

Questo vuol dire che il disco più piccolo va spostato dal paletto 0 al paletto 2, quindi il disco mediano, ora in cima al paletto 0, va spostato al paletto 1; a questo punto il disco più piccolo (rimasto sul paletto 2) va rimesso sopra il mediano (ora sul paletto 1) e, finalmente, il disco più grande va spostato dal paletto 0 al paletto 2, nella sua posizione finale. Le restanti tre mosse, spostando i due dischi rimanenti dal paletto 1 al paletto 2.

Scrivete una funzione `hanoi( int n, int from, int temp, int to );` che stampi le mosse per spostare `n` dischi dal paletto `from` al paletto `to` aiutandosi, se necessario, con il paletto ausiliario `temp`. Osservate che tale funzione, posto che siano state stampate le mosse per spostare $n-1$ dischi da `from` a `temp` (usando eventualmente `to` come paletto ausiliario) può stampare la mossa `from ->to` e quindi stampare le rimanenti mosse necessarie a spostare gli $n-1$ dischi da `temp` a `to` (usando eventualmente `from` come paletto ausiliario). Questa osservazione dovrebbe suggerirvi immediatamente una implementazione ricorsiva della funzione `hanoi`.

La fine del mondo. Modificate la funzione precedente perché restituisca soltanto il numero di mosse effettuate (invece che stamparle). Come cresce tale numero al crescere del numero di dischi? Per rendervi meglio conto del tasso di crescita, provate a considerare il logaritmo del numero di mosse, come cresce?

Vi sembra realistico che per spostare 64 dischi ci voglia un tempo pari alla durata del mondo?

Le torri in dettaglio Supponendo ora di chiamare i dischi dal più grande al più piccolo con le lettere `A`, `B`, `C`, ..., scrivete ora una versione più dettagliata della funzione appena sviluppata che stampi ad ogni passo del gioco il contenuto di ogni paletto (usando una linea per ogni passo e separando il contenuto dei tre paletti con una virgola). Ad esempio, se la pila iniziale ha altezza 3, ed è quindi data da `ABC`, la nuova funzione deve scrivere le seguenti mosse

```
ABC, ,
AB, , C
A, B, C
A, BC,
, BC, A
C, B, A
C, , AB
, , ABC
```

3 Fiocco di Koch

Libreria `libpsgraph.c`

Questo esercizio fa uso di una libreria chiamata `libpsgraph.c` che consente di produrre dei semplici grafici in PostScript. Per usare questa libreria dovete procedere come segue:

- create una directory;
- scaricate in essa dalla homepage del corso i file `libpsgraph.c` e `libpsgraph.h`;
- scrivete la vostra applicazione, aggiungendo in testa la direttiva

```
#include "libpsgraph.h"
```

- compilate con il comando

  ```
  gcc -lm libpsgraph.c pippo.c -o pippo
  ```

  dove al posto di `pippo.c` metterete il nome del vostro file sorgente, mentre al posto di `pippo` metterete il nome che volete che abbia il vostro eseguibile.

Funzioni fornite dalla libreria

La libreria vi permette di disegnare usando le funzioni della *turtlegraphics*; il disegno prodotto viene scritto in formato PostScript su un file (il cui nome dovete specificare quando iniziate a disegnare) e dopo aver eseguito il programma potete vedere il risultato aprendo il file con un opportuno viewer (`ggv` oppure `gv`, o un altro viewer che vi verrà segnalato dai tutor).

La libreria va usata così:

- per prima cosa, invocate la funzione `start(nomefile)` passandole il nome del file in cui volete che il grafico venga salvato, ad esempio `start("prova.ps");`
- a questo punto, per disegnare potete usare le seguenti funzioni (che prendono come argomento un double):
  - `draw(x)`: disegna un segmento lungo `x` millimetri;
  - `move(x)`: si sposta (senza disegnare) di un segmento lungo `x` millimetri;
  - `turn(x)`: si gira a destra di `x` gradi;
- alla fine, dovete invocare la funzione `end()`.

Ecco, ad esempio, un programma che disegna un quadrato:

```
#include "libpsgraph.h"

int main() {
        start("quadrato.ps");
        draw(50);
        turn(90);
        draw(50);
        turn(90);
        draw(50);
        turn(90);
        draw(50);
        end();
        return 0;
}
```

Curva di Koch

Realizzate una funzione che, data una lunghezza in millimetri x e un intero i, produce la curva di Koch di ordine i e di lunghezza x. Essa è definita come segue:

- se $i = 0$, è un segmento di lunghezza x;
- se $i > 0$, è ottenuta giustapponendo quattro curve di Koch di ordine $i-1$ e di lunghezza $x/3$, come in Figura 1.

Dopo averla realizzata, verificate che funzioni (scrivendo un main che la invoca e guardando il file risultante).

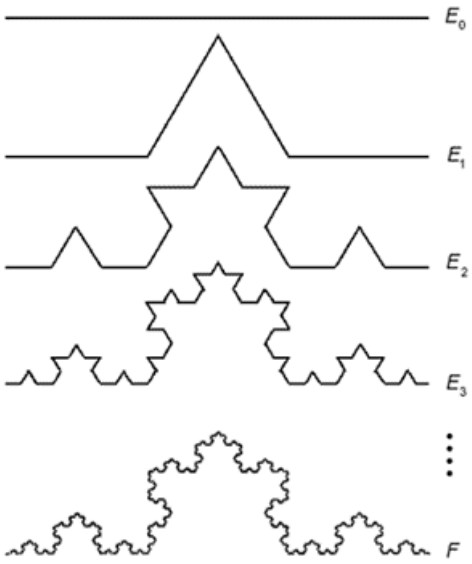

Figura 1: Costruzione di una curva di Koch (E_i è una curva di Koch di ordine i).

Fiocco di neve di Koch

Realizzate ora una funzione che, data una lunghezza in millimetri x e un intero i, produce il fiocco di neve di Koch di ordine i e di lunghezza x: esso si ottiene come un triangolo equilatero di lunghezza x i cui lati siano stati sostituiti con curve di Koch di ordine i e lunghezza x (vedi Figura 2).

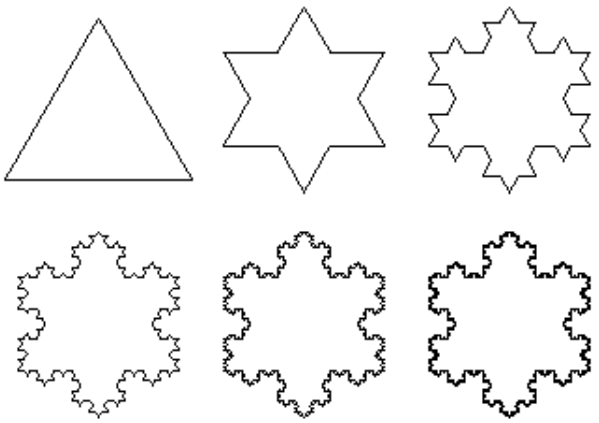

Figura 2: Fiocco di neve di Koch.