

Laboratorio di programmazione

30 novembre 2007

Sequenze di Collatz

Considerate la seguente regola: dato un numero intero positivo n , se n è pari lo si divide per 2, se è dispari lo si moltiplica per 3 e si aggiunge 1 al risultato. Quando n è 1 ci si ferma.

Questa semplice regola permette di costruire delle sequenze: la sequenza che si costruisce a partire dal numero n è detta *sequenza di Collatz di n* . Ad esempio, la sequenza di Collatz di 7 è:

7 22 11 34 17 52 26 13 40 20 10 5 16 8 4 2 1

È un noto problema aperto stabilire se ogni sequenza di Collatz termina (cioè, se arriva a 1).

Scrivete innanzitutto una funzione che, dato un numero, dia il successivo in una sequenza di Collatz. Quindi, inseritela in un programma che chiede all'utente un numero e mostra la sequenza di Collatz del numero (con tanto di lunghezza).

Esempio di funzionamento

```
Numero: 7
7 22 11 34 17 52 26 13 40 20 10 5 16 8 4 2 1
Lunghezza: 17
```

```
Numero: 9
9 28 14 7 22 11 34 17 52 26 13 40 20 10 5 16 8 4 2 1
Lunghezza: 20
```

Garibaldi fu ferito

Scrivere una funzione che dati due caratteri restituisca il primo carattere se il secondo è una vocale minuscola, altrimenti restituisca il secondo carattere. Utilizzando tale funzione, scrivete un programma che legga un carattere, uno spazio e quindi una sequenza di caratteri minuscoli terminati da . e che stampi quanto ha letto dopo il primo spazio, ma sostituendo tutte le vocali con il primo carattere letto.

Esempio di funzionamento

```
Stringa: u garibaldi fu ferito, fu ferito in una gamba.
gurubuldu fu furutu, fu furutu un unu gumbu
```

Ore

Scrivete un programma che, data un'ora sotto forma di tre numeri interi (ore, minuti, secondi), stampi la rappresentazione testuale dell'ora indicata.

Esempio di input

```
12 30 22
```

Esempio di output

```
Ore dodici, trenta minuti e ventidue secondi.
```

Suggerimenti

È importante strutturare bene il programma in funzioni. Un punto di partenza è creare una funzione in grado di stampare la parte delle decine, cioè che su input 2 scriva “venti”, su input 3 scriva “trenta” e così via (usate l'istruzione `switch`). Sulla base di questa è poi possibile creare una funzione che stampi in maniera testuale un numero tra 0 e 99. Si noti però che la prima funzione deve avere un secondo parametro che specifica (a seconda del suo valore) se bisogna stampare l'ultima vocale oppure no (altrimenti numeri come “trentuno” non sarebbero stampabili).

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 coppie 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, ...

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
7 13
```

Modificate il programma perché, 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.

Suggerimento. Una soluzione iterativa elementare si può ottenere utilizzando un vettore nell' n -posizione del quale sia immagazzinato F_n . Per riempire tale vettore basta un semplice ciclo `for`. Ma, a ben guardare, ogni volta userete solo due posizioni del vettore, questo suggerisce una soluzione che utilizzi solo tre variabili (invece di un vettore con n posti), memorizzando solo F_n , F_{n-1} e F_{n-2} .

Indovina il numero

Il gioco *Indovina il numero* funziona come segue: un giocatore *A* pensa a un numero intero x (con $0 \leq x \leq 1000$), e l'altro giocatore, *B*, lo deve indovinare. Per farlo, *B* pone domande del tipo "Il numero è y ?", cui *A* può rispondere = (per indicare che il numero è stato indovinato), oppure < (per indicare che x è minore del numero y), oppure > (per indicare che x è maggiore del numero y).

Scrivete un programma che giochi come giocatore *B* al gioco.

Esempio di funzionamento

```
Il numero è 500? <
Il numero è 249? <
Il numero è 124? <
Il numero è 61? <
Il numero è 30? >
Il numero è 45? <
Il numero è 37? =
```

Suggerimento. Il programma deve tenere due variabili, diciamo *left* e *right* che contengono gli estremi entro cui può stare il valore da indovinare. All'inizio, *left* sarà 0 e *right* sarà 1000. Ogni volta, il programma chiede se il valore è quello in mezzo all'intervallo (la media fra *left* e *right*), e a seconda della risposta dell'utente il programma o esce oppure modifica i valori di *left* e *right* opportunamente per restringere l'intervallo.

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 *.c -o pippo
```

dove 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 (p.es. `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).

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).

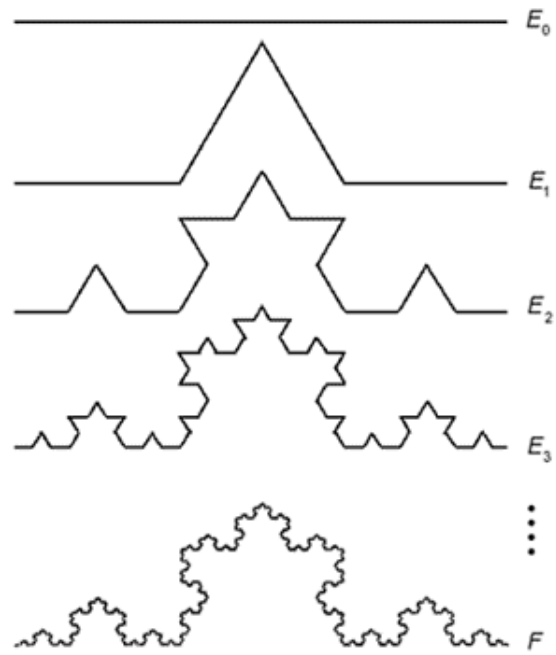


Figura 1: Costruzione di una curva di Koch (E_i è una curva di Koch di ordine i).

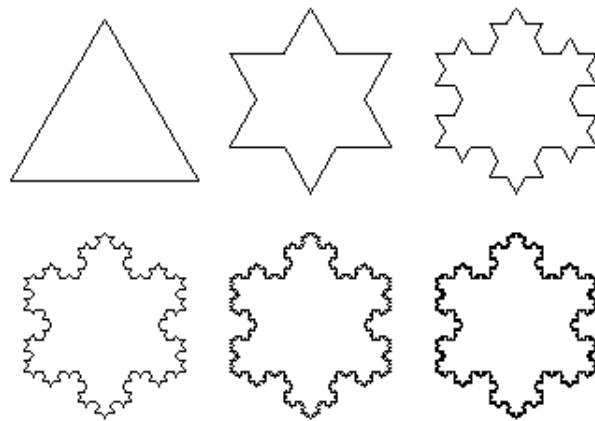


Figura 2: Fiocco di neve di Koch.