Iterazione - esercizi

Liceo G.B. Brocchi – Bassano del Grappa (VI)
Liceo Scientifico – opzione scienze applicate
Giovanni Mazzocchin

Iterazione - esercizi

- **NB**: le soluzioni di alcuni degli esercizi sono date in uno pseudocodice simile a Python. Lo studente si occuperà di tradurli in C/C++

- **NB**: risolvere tutti gli esercizi utilizzando sia il costrutto `while` sia il costrutto `for`

- **NB**: quando possibile, risolvere l'esercizio anche per mezzo di un foglio di calcolo

Programmazione su GNU/Linux

- Utilizzare la riga di comando (shell Bash)
- Text editor da utilizzare: `nano`, `vim`, `emacs`, `gedit`
- Compilazione di un file sorgente C:

```
gcc -o program program.c
```

- Compilazione di un file sorgente C++:

```
g++ -o program program.cpp
```

- Compilazione per il debugging:

```
gcc -g -o program program.c
```

- Lancio di un eseguibile ELF:

```
./program
```

Differenze C – C++

- C e C++ sono linguaggi molto diversi
- Per i nostri scopi, durante i primi 2 anni, C e C++ saranno linguaggi sostanzialmente uguali
- Le differenze che dobbiamo conoscere sono così poche da essere sintetizzabili nel programma *hello world*

C – *hello world*

```c
#include <stdio.h>

int main() {
  printf("hello world\n");
}
```

C++ – *hello world*

```
#include <iostream>
using namespace std;

int main() {
  cout << "hello world" << endl;
}
```

Generazione di triple di somma *n*

- **Problema**: dato $n \geq 3$ naturale, generare tutte le triple di naturali (x, y, z) con $x \geq 1,\ y \geq 1,\ z \geq 1$ t.c. $x + y + z = n$

- Prima di pensare all'algoritmo, capiamo bene quale deve essere l'output della procedura

Generazione di triple di somma *n*

- **Problema**: dato $n \geq 3$ naturale, generare tutte le triple di naturali (x, y, z) con $x \geq 1,\ y \geq 1,\ z \geq 1$ t.c. $x + y + z = n$

				n	5
x (1 up to n - 2)	**y (1 up to n - x - 1)**	**z (n - x - y)**	**sum**		
1	1	3	5		
1	2	2	5		
1	3	1	5		
2	1	2	5		
2	2	1	5		
3	1	1	5		

Generazione di triple di somma *n*

- **Problema**: dato $n \geq 3$ naturale, generare tutte le triple di naturali (x, y, z) con $x \geq 1,\ y \geq 1,\ z \geq 1$ t.c. $x + y + z = n$

```
n = ...
x = 1

while x <= n - 2:
  y = 1
  while y <= n - x - 1:
    z = n - x - y
    y = y + 1
  x = x + 1
```

Ricerca di terne pitagoriche

- **Problema**: utilizzare l'algoritmo visto sopra per ricercare terne pitagoriche di somma data

- Recuperare dalla Matematica la definizione di terna pitagorica

Un'equazione diofantea

- **Problema**: utilizzare l'algoritmo visto sopra per ricercare le soluzioni dell'equazione diofantea $x^n + y^n = z^n$, con $n > 2$

- Trovato qualcosa?

Un'equazione diofantea

- **Problema**: utilizzare l'algoritmo visto sopra per ricercare le soluzioni dell'equazione diofantea $x^n + y^n = z^n$, con $n > 2$

- L'Ultimo Teorema di Fermat

Fattoriale

- **Problema**: implementare il calcolo del fattoriale di un numero naturale. Ricordiamo la definizione di fattoriale:

$$n! = n \cdot (n - 1) \cdot (n - 2) \cdot \ \ldots \cdot \ (n - k) \ \ldots \cdot 1$$

Fattoriale

- **Problema**: implementare il calcolo del fattoriale di un numero naturale. Ricordiamo la definizione di fattoriale:

$$n! = n \cdot (n - 1) \cdot (n - 2) \cdot \ldots \cdot (n - k) \ldots \cdot 1$$

```
n = ...
acc = 1
i = n
while i >= 1:
  acc = acc * i
  i = i - 1
```

Fattoriale

- **Problema**: implementare il calcolo del fattoriale di un numero naturale. Ricordiamo la definizione di fattoriale:

$$n! = n \cdot (n - 1) \cdot (n - 2) \cdot \ \dots \cdot \ (n - k) \ \dots \cdot 1$$

	acc	i	n
end of iteration 0 (before loop)	1	6	6
end of iteration 1	6	5	6
end of iteration 2	30	4	6
end of iteration 3	120	3	6
end of iteration 4	360	2	6
end of iteration 5	720	1	6
end of iteration 6	720	0	6

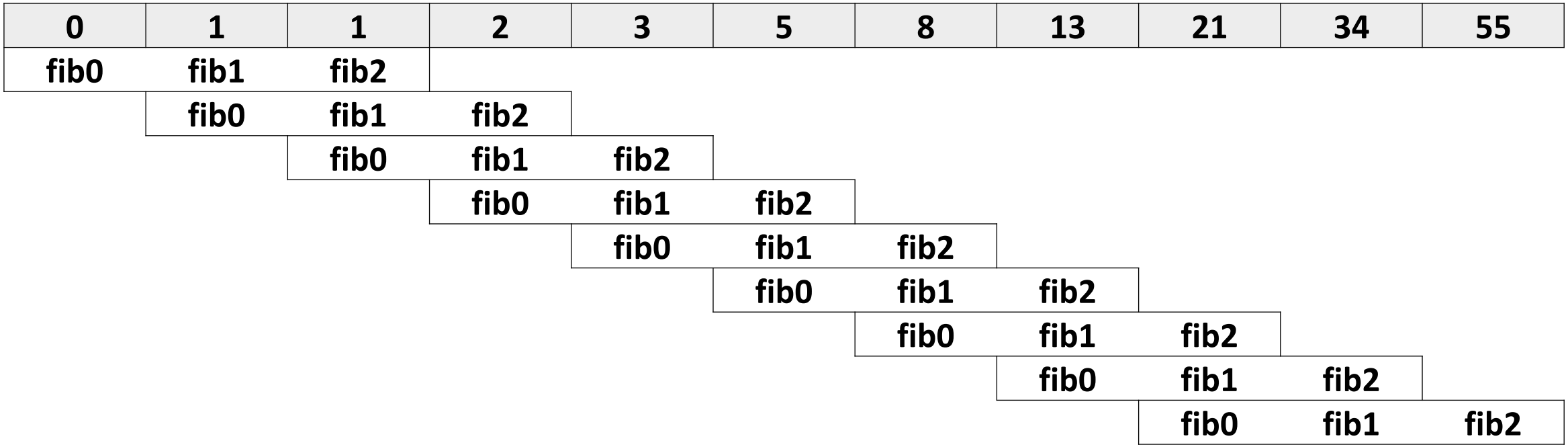
La successione di Fibonacci

- **Definizione** (*relazione di ricorrenza*):

$$f_0 = 0$$
$$f_1 = 1$$
$$f_n = f_{n-1} + f_{n-2}$$

La successione di Fibonacci

- Calcolo dei numeri di Fibonacci con 3 variabili



La successione di Fibonacci

- Calcolo dei numeri di Fibonacci con 3 variabili, pseudocodice

```
//compute the first 10 Fibonacci numbers
fib0 = 0
fib1 = 1
i = 1
while i <= 8:
  fib2 = fib0 + fib1
  fib0 = fib1
  fib1 = fib2
  i = i + 1
```

L'algoritmo di Euclide per l'MCD

- **Definizione**:

$$\gcd(a, 0) = a$$
$$\gcd(a, b) = \gcd(b, a \bmod b) \qquad if\ b > 0$$

- Scrivere un programma che verifica se due numeri sono *primi tra loro*, sfruttando l'algoritmo di Euclide
- *gcd*: **g**reatest **c**ommon **d**ivisor

L'algoritmo di Euclide per l'MCD

a	b
7553625	231342
231342	150681
150681	80661
80661	70020
70020	10641
10641	6174
6174	4467
4467	1707
1707	1053
1053	654
654	399
399	255
255	144
144	111
111	33
33	12
12	9
9	3
3	0

verificare anche con Python, con i seguenti comandi:

```
import sympy
sympy.gcd(7553625, 231342)
```

notare quanto è efficiente questo algoritmo: a e b sono piuttosto grandi, ma i passaggi sono pochi

Approssimazione della radice quadrata

- Approssimiamo la radice quadrata di un numero per tentativi
- Ci sono metodi più efficienti per estrarre la radice quadrata, ma per ora implementiamo questo per esercizio
- L'approssimazione procede in questo modo:
  - approssimazione all'unità
  - approssimazione ad 1 cifra decimale
  - approssimazione a 2 cifre decimali
  - …
  - approssimazione a n cifre decimali
- Eseguiamo il procedimento con uno spreadsheet prima di implementarlo in C

Test di primalità *trial-division*

- Scrivere un programma che verifica se un numero è primo

- Recuperare dalla Matematica la definizione di numero primo

Test di primalità *trial-division*

			n
trial divisor	**remainder**		119
3	2		
5	4		
7	0	divisor	
9	2		
11	9		
13	2		
15	14		
17	0	divisor	
19	5		

si conclude che 119 non è primo, ma composto

si può fermare l'iterazione dopo aver trovato il primo divisore

Test di primalità *trial-division*

- Proviamo a limitare l'insieme di numeri in cui cercare divisori
- Notiamo subito che dato n, non ha senso cercare divisori oltre $\frac{n}{2}$
- Ma possiamo fare di meglio, notando che:
  - se n è composto, allora $n = a \cdot b$, per qualche a, b naturale
  - dato che $n = \sqrt{n} \cdot \sqrt{n}$, necessariamente $a \leq \sqrt{n}$ o $b \leq \sqrt{n}$
  - se fosse vero che $a > \sqrt{n}$ e $b > \sqrt{n}$, allora sarebbe vero che

    $$n = a \cdot b > \sqrt{n} \cdot \sqrt{n} = n$$

    che è assurdo

- Quindi i divisori vanno cercati nell'insieme $\{2 \,..\sqrt{n}\}$

Fattorizzazione

- *Fattorizzazione* significa *scomposizione in fattori primi*
- Scrivere un programma che fattorizza un numero naturale
- I fattori vanno semplicemente stampati separati da un carattere di tabulazione orizzontale

Stampa di figure su `stdout`

- Scrivere un programma che stampa questa figura:

Stampa di figure su `stdout`

- Scrivere un programma che stampa questa figura:

```
*********
 *******
  *****
   ***
    *
```

  - successivamente, scrivere un programma che la stampa al contrario

Stampa di figure su `stdout`

- Scrivere un programma che stampa questa figura:

```
########
  ########
#######
  ########
########
```

La congettura di Collatz

- Partendo da un n naturale qualsiasi, generare una successione utilizzando questa formula:

$$c_n = \frac{c_{n-1}}{2} \quad if\ c_{n-1}\ is\ even$$

$$c_n = 3c_{n-1} + 1 \quad otherwise$$

- Visitare questa pagina: https://en.wikipedia.org/wiki/Collatz_conjecture

π con la formula di Leibniz

- Scrivere un programma che implementa la formula di Leibniz per il calcolo del *pi greco*. La formula è la seguente:

$$4 - \frac{4}{3} + \frac{4}{5} - \frac{4}{7} + \frac{4}{9} - \frac{4}{11} + \frac{4}{13} \ldots$$

- La costruzione della sommatoria è piuttosto evidente

- Si dimostra che la somma, all'infinito, converge a π

Potenza

- Scrivere un programma che implementa l'elevamento a potenza con base reale ed esponente intero (potenzialmente negativo)

Divisione come sottrazione ripetuta

- Scrivere un programma che implementa la divisione euclidea (*divisione con quoziente e resto*) come sottrazione ripetuta

Logaritmo intero in base 2

- Scrivere un programma che calcola la potenza intera di 2 che si avvicina di più (per difetto) ad un numero naturale letto da standard input