Linguaggi I2

Serie operatori, funzioni e biblioteche

1. Consideriamo tutti i numeri interi positivi di 3 cifre. Fra tutti questi numeri ce ne sono alcuni, il cui valore corrisponde esattamente alla somma dei cubi delle loro cifre.

Es: $153 = 1^3+5^3+3^3$

Scrivere un programma che, letto un numero intero:

a) Verifichi **con una funzione** che sia un numero positivo di 3 cifre (nel caso contrario ripete la domanda)

b) Verifichi la proprietà descritta sopra (rispondendo "Sì" oppure "No").

Es:
```
Numero positivo di 3 cifre:  26
Numero positivo di 3 cifre:  -145
Numero positivo di 3 cifre:  356
Risposta: No
```

c) Modificare il funzionamento in modo che sia il programma, senza ricevere input, a generare i numeri per cui vale la proprietà vista.

2. Definire **una funzione**, che ricevuto un intero in input restituisce la metà del suo argomento se questi è pari o il triplo più uno se invece è dispari.

Usarla per stabilire per quale valore di n tra 1 e 100 la sequenza di applicazioni ripetute della funzione fino a 1 è la più lunga (per applicazione ripetuta si intende che il valore restituito da una chiamata viene passato come argomento ad una chiamata successiva).

Visualizzare inoltre la lunghezza della sequenza trovata.

Es:
```
n = 10
sequenza: 10 5 16 8 4 2 1
lungezza: 6
```

3. Scrivere un programma in grado di contare caratteri, parole e linee contenuti in un file (attraverso la chiamata con ridirezione).

Chiamata: `prog < testo.txt`

Output:

```
Caratteri:     568
Parole:         74
Linee           12
```

Varianti:

a) Contare le parole con il numero di spazi e il numero di eol.

b) Considerare che per una sequenza di spazi venga calcolata una sola parola.

c) Considerare che per una sequenza di eol venga calcolata una sola parola.

4. Realizzare la funzione *fattoriale* (sia versione iterativa che versione ricorsiva):

a) Inserirla in un programma in cui venga richiamate da una funzione principale che si occupa di input e output.

b) Terminare l'esercizio inserendo i controlli di overflow. Utilizzare le costanti definite nel file header <limits.h> (INT_MAX, DBL_MAX, SHRT_MAX, ecc.)

5. Approssimare una soluzione di un'equazione data (es: `2x-cos(x)`) utilizzando il metodo della bisettrice.

La routine principale dev'essere di questo tipo:

```
double bisettrice(double a, double b, double eps)
{...}
```

I due parametri *a* e *b* sono i punti di partenza dell'iterazione, *eps* rappresenta invece la precisione desiderata (potrebbe anche essere definito come costante).

a) Richiedere *a* e *b* da input (controllando che il prodotto dei risultati della funzione sia negativo)

b) Scrivere una funzione che sia in grado di proporre i due valori di partenza.

6. Utilizzando le funzioni di biblioteca `int rand()` e `int abs(int n)` contenute in <stdlib.h>, determinare qual è la distanza massima e quale quella minima tra due eventi consecutivi in una generazione di 1000 numeri con valori oscillanti tra 0 e 10'000.

Visualizzare in output la distanza minima e quella massima.

7. Scrivere un programma che, ricevuti in input due valori *a* e *b* di tipo double esegua i seguenti calcoli, utilizzando la libreria <math.h>

- a^b
- l'ipotenusa di un triangolo rettangolo con cateti *a* e *b*
- la tangente di *a*, usando unicamente le funzioni *sin()* e *cos()*.
- il valore arrotondato (per difetto) di tre posizioni dopo la virgola di a/b

8. Realizzare la funzione per la generazione dei numeri Fibonacci, basata sulla seguente definizione ricorsiva:

```
fibo(0)=fibo(1) = 1
fibo(n) = fibo(n-1) + fibo(n-2)
```

a) Discutere i problemi della soluzione ricorsiva.

b) Modificare la funzione per la generazione dei numeri Fibonacci, utilizzando variabili per la memorizzazione dei dati intermedi.