Laboratorio di Calcolo per Fisici, Seconda esercitazione

Canale Lp-P, Docente: Cristiano De Michele

Lo scopo della seconda esercitazione di laboratorio è di scrivere da zero dei semplici programmi in C usando le funzioni della libreria matematica `math.h` e di familiarizzare con la shell di linux. Inoltre utilizzerete ancora python per realizzare dei semplici grafici.

► **Prima parte:**

1. **Se siete in laboratorio in via Tiburtina:**

   (a) effettuare il *login* sulla propria macchina *Unix* utilizzando lo *userid* `lccdmXXX`, dove `XXX` è il numero del gruppo a cui siete stati assegnati (la password è identica allo *userid*),

   (b) aprite una finestra del *terminale*.

   Se siete in laboratorio al nuovo edificio "Fermi" di Fisica:

   (a) entrate usando come *userid* `studente` e come password `informatica`

   (b) aprite il terminale

   (c) create con il comando `mkdir` la cartella `LCCDMXXX` (lettere maiuscole), dove `XXX` è il numero che vi è stato assegnato come gruppo

   (d) entrate nella cartella appena creata con il comando `cd`.

2. Creare una cartella `EX2TMP` che conterrà il materiale di questa seconda esercitazione.

3. Entrare nella cartella appena creata con il comando `cd EX2TMP`

4. Nella cartella `EX2TMP` aprire con l'editor di testo il file `matematica.c`, e digitare il listato sottostante. Salvare il contenuto del file. *Suggerimento:* per aprire il file usare il comando `emacs matematica.c`.

```c
#include <stdio.h>
#include <math.h>

int main(void)
{
  printf("Benvenuto! Questo è un programma dimostrativo delle funzioni matematiche in
    C.\n");
  printf("8 + 7 = %d\n", 8+7); // %d indica che si vuole stampare un intero
  printf("3.2 + 8.4 = %f\n", 3.2+8.4); // %f indica un numero floating point in doppia
     precisione
  printf("2^2+4^2 = %f\n", pow(2, 2) + pow(4, 2));
  printf("sqrt(36) = %f\n", sqrt(36));
```

```
  printf("cos(pi/6) = %f\n", cos(M_PI/6.0));
  printf("sin(pi/6)=%f\n", sin(M_PI/6));
  printf("sinh(0) = %f  tanh(1)=%f\n", sinh(0.0),tanh(1.0));
  printf("cos(pi/6) + 10 = %f  cos(pi/6) + 10 = %f\n", cos(M_PI/6) + 10., cos(M_PI/6)
    + 10.);
  printf("cos(pi/6)*10^4 = %f  cos(pi/6)/10^2 = %f\n", cos(M_PI/6)*10000, cos(M_PI
    /6.0)/100);
  printf("%f %f %f\n", 0.5, 6.2*cos(0.1*0.5), 6.2*sin(0.1*0.5));//R*cos(w*t) con R
    =6.2,w=0.1,t=0.5
}
```

Listato 1: Programma `matematica.c`

Notare che il comando `printf`, richiede degli *argomenti* che vanno racchiusi tra parentesi tonde e separati da virgole, ovvero vanno forniti nel seguente modo:

```
(argomento1, argomento2, ...)
```

Il primo *argomento* (`argomento1`) deve essere una stringa ovvero una serie di caratteri compresi tra due doppi apici, ad es, `"ciao"` è una stringa in C. Nella stringa fornita come primo argomento possono essere inseriti delle sequenze speciali, come ad es. `%d` (intero) e `%f` (numero in virgola mobile). Per ognuna di tali sequenze speciali va fornito un argomento corrispondente nella `printf` dopo il primo (la stringa), che consisterà in un'espressione numerica. Ad esempio per stampare `1.2+1.3` utilizzeremo il comando:

```
printf("%f\n", 1.2+1.3);
```

dove `\n` rappresenta il carattere (speciale) di ritorno a capo.

5. Compilare il programma ed eseguirlo con i seguenti comandi:

```
> gcc matematica.c -lm -o matematica.exe
> ./matematica.exe
```

Notare l'opzione `-lm` che è necessaria per poter utilizzare le funzioni matematiche.

6. Creare una copia di *backup* dei sorgenti del programma appena eseguito con il comando `cp` (*copy*)

```
> cp matematica.c matematica_bak.c
```

7. Cancellare la copia di backup con il comando `rm` (*remove*)

```
> rm matematica_bak.c
```

8. Spostarsi nella cartella superiore a quella attuale con il comando `cd` (*change directory*) e rinominare la cartella `EX2TMP` in `EX2` con il comando `mv` (*move*), impartendo i seguenti comandi:

```
> cd ..
> mv EX2TMP EX2
```

9. Mostrare il contenuto della cartella corrente (`HOME`) e poi spostarsi nella cartella `EX2` con i comandi:

```
> ls -l
> cd EX2
```

► **Seconda parte:**
Un moto circolare uniforme sul piano (x, y) è descritto dalle equazioni:

$$\begin{cases} x(t) = R\cos(\omega t) \\ y(t) = R\sin(\omega t)\,, \end{cases}$$

dove R è il raggio della traiettoria e ω la velocità angolare. L'obiettivo della seconda parte dell'esercitazione è quello di scrivere un programma `circle.c` che utilizzi la libreria `math.h` per stampare alcuni punti della traiettoria di un moto circolare uniforme con $R = 6.2$ m, $\omega = 0.1$ rad/s. In particolare, tale programma dovrà calcolare $x(t)$ e $y(t)$ per i valori: $t = 0$, $t = 0.5$, $t = 10$, $t = 20$ s.

1. Si faccia anzitutto una copia del sorgente `matematica.c` chiamata `circle.c`, con il comando

```
> cp matematica.c circle.c
```

2. Si apra il file `circle.c` con il comando:

```
> emacs circle.c
```

   E si cancellino tutte le righe contenenti l'istruzione `printf` eccettuata la riga 16 (vedi Listato 1).

3. Inserire righe analoghe all'unica rimasta cambiando opportunamente il tempo secondo i valori indicati in precedenza. Quello che dovrete avere è una riga per ogni tempo indicato, ovvero un codice del tipo:

```
printf("%f %f %f\n", 0.0, 6.2*cos(0.1*0.0), 6.2*sin(0.1*0.0)); // t=0.0
printf("%f %f %f\n", 0.5, 6.2*cos(0.1*0.5), 6.2*sin(0.1*0.5)); // t=0.5
...
```

4. Compilare il programma con il comando:

```
> gcc circle.c -lm -o circle.exe
```

   ed eseguitelo con il comando

```
> ./circle.exe
```

In modo da ottenere in output i tre valori t, x, y su tre colonne come nell'esempio sottostante:

```
  0.000000 6.200000 0.000000
  0.500000 6.192252 0.309871
  ...
```

5. Create un file, chiamato `circle.dat`, che contenga l'output del programma, utilizzando il seguente comando:

```
> ./circle.exe > circle.dat
```

con cui l'output del comando, in questo caso il programma `circle.exe` viene "ridirezionato" sul file `circle.dat`

6. *Suggerimento:* Per non far compilare parti di programma senza cancellarle o per chiarire cosa faccia il codice, può essere utile inserire un *commento*. In C un commento è racchiuso dai delimitatori `/*` e `*/`, come nell'esempio sottostante:

```
  /* Questo è un commento */
```

oppure è sufficiente iniziare la linea con `//`, ovvero:

```
  // Anche questo è un commento
```

► **Terza parte (obbligatoria)**

In questa terza parte dell'esercitazione useremo il programma `circle.c` per studiare l'andamento del moto circolare uniforme, utilizzando `python`.

1. Inserite nel programma `circle.c` un numero sufficiente di `printf` con diversi tempi per avere una distribuzione di punti su tutta la traiettoria circolare e "ridirezionate" l'output del programma (come avete già fatto nella seconda parte dell'esercitazione) su un file chiamato `traiettoria.dat` che contenga tre colonne t, x, y. *Ricordatevi che se volete inserire un'intestazione in un file di dati letto tramite* `loadtxt` *in* `python`*, dovete aggiungere il parametro* `comments='#'`*, assumendo che i commenti inizino con il simbolo* `#` (si veda il punto seguente).

2. Utilizzare `python` per graficare la traiettoria, graficando y in funzione di x. *Per graficare solo due colonne di un file che ne contiene molte, la funzione* `loadtxt` *va chiamata con il parametro* `usecols=(i,j)`*, dove* `i` *e* `j` *sono le colonne da graficare (partendo da* 0*, che indica la prima colonna). Nel nostro caso, se volessimo graficare solo la seconda e la terza colonna il comando completo da usare nello script* `python` *diventerebbe:*

```
  np.loadtxt('traiettoria.dat', comments=['#'], usecols=(1,2), unpack=True)
```

In questo modo uno script python utile allo scopo è il seguente:

```
import matplotlib.pyplot as plt # abilita l'utilizzo delle funzioni per fare grafici
import numpy as np # abilita utili funzioni matematiche e di input/output (ad es. np.
    loadtxt)

plt.title('Moto Circolare Uniforme')  # imposta il titolo del plot
plt.xlabel('x') # imposta la label dell'asse x
plt.ylabel('y') # imposta la label dell'asse y
# il comando seguente legge la seconda e la terza colonna del file traiettoria.dat
x, y = np.loadtxt('traiettoria.dat', comments=['#'], usecols=(1,2), unpack=True)
plt.plot(x, y, 'x',label='traiettoria') # grafica la traiettoria
plt.xlim(-8.0, 8.0) # imposta l'intervallo per l'asse delle x
plt.legend() # aggiunge la legenda
plt.show() # mostra il grafico
```

3. Creare altri due grafici che mostrino l'andamento della coordinata x e della coordinata y in funzione del tempo, rispettivamente. *Suggerimento: potete copiare e poi modificare lo script Python precedentemente creato.*

4. Salvare i tre grafici in tre file separati, chiamati: `traiettoria.png`, `x.png` e `y.png`. *Per salvare i grafici con `matplotlib`, bisogna usare il seguento comando python che dovete aggiungere prima di `plt.show()`*

```
    plt.savefig('traiettoria.png')
```

Suggerimento: per salvare $x(t)$ e $y(t)$ sostituite `traiettoria.png` con `x.png` e `y.png` rispettivamente.

► Quarta parte (facoltativa)

L'andamento delle *componenti* della velocità in un moto circolare uniforme è dato da:

$$\begin{cases} v_x(t) = -\omega R \sin(\omega t) \\ v_y(t) = \omega R \cos(\omega t)\,, \end{cases}$$

1. Copiare il file `circle.c` in un nuovo file chiamato `circle-vel.c` per calcolare anche la velocità del punto lungo la traiettoria e inserire i risultati ottenuti nel file *traiettoria.dat* (ridirezionando di nuovo l'output del programma in tale file come avete fatto prima), creando altre due colonne oltre quelle esistenti con le componenti dalle velocità v_x e v_y. Le righe con le `printf` in tale nuovo programma dovranno essere quindi del tipo:

```
printf("%f %f %f %f %f\n",0.0,6.2*cos(0.1*0.0),6.2*sin(0.1*0.0),-0.1*6.2*sin(0.1*0.0)
    ,0.1*6.2*cos(0.1*0.0));
printf("%f %f %f %f %f\n",0.5,6.2*cos(0.1*0.5),6.2*sin(0.1*0.5),-0.1*6.2*sin(0.1*0.5)
    ,0.1*6.2*cos(0.1*0.5));
...
```

dove come potete notare è stata aggiunta la stampa di due ulteriori colonne che corrispondono a $v_x(t)$ e $v_y(t)$.

2. Graficare l'andamento della componente x della velocità in funzione del tempo lungo la traiettoria (graficando le colonne 0 e 3) e salvare il grafico nel file `vx.png`.

3. Graficare l'andamento della componente y della velocità in funzione del tempo lungo la traiettoria (graficando le colonne 0 e 4) e salvare il grafico nel file `vy.png`.

4. Disegnare il vettore velocità lungo la traiettoria, come una freccia che punta nella direzione corretta sfruttando il seguente codice *python*:

```
x, y, vx, vy = np.loadtxt('traiettoria.dat', usecols=(1,2,3,4), unpack=True)
for cc in range(0, len(x)):
  xi = x[cc]
  yi = y[cc]
  vxi = vx[cc]
  vyi = vy[cc]
  plt.arrow(xi, yi, vxi, vyi, width=0.2, head_width=0.5, head_length=0.3, fc='r', ec
  ='r')
```

che disegna `len(x)` frecce di lunghezza `(vx,vy)` nei punti `(x,y)`. Questo codice va aggiunto prima di `plt.savefig()`.

Funzioni piú comuni della libreria `math.h`:

acos(x) arcocoseno
asin(x) arcoseno
atan(x) arcotangente
atan2(x) arcotangente di due parametri
ceil(x) il più piccolo intero non minore del parametro
cos(x) coseno
cosh(x) coseno iperbolico
exp(x) funzione esponenziale, calcola e^x
fabs(x) valore assoluto
floor(x) il più grande intero non maggiore del parametro
fmod(x) resto del numero in virgola mobile
frexp(x,y) frazione e potenza di due.
ldexp(x,y) operazione in virgola mobile
log(x) logaritmo naturale
log10(x) logaritmo in base 10
pow(x, y) eleva un valore dato ad esponente, x^y
sin(x) seno
sinh(x) seno iperbolico
sqrt(x) radice quadrata
tan(x) tangente
tanh(x) tangente iperbolica