# Laboratorio di Calcolo per Fisici, quarta esercitazione

Canale Lp-P, Docente: Cristiano De Michele

Lo scopo della quarta esercitazione di laboratorio è di iniziare ad utilizzare delle strutture di controllo del flusso, quali i costrutti di selezione (cioè `if...then...else`) e i cicli.

---

► **Prima parte:**
Vediamo anzitutto con degli esempi come si possono implementare le strutture di controllo in C. Il costrutto di selezione `if...then...else` si può realizzare nel seguente modo:

```
if ( /* inserire qui una condizione ad es. x==0 */ )
  {
    /* inserire qui le istruzioni da eseguire
       se la condizione è vera */
  }
else
  {
     /* inserire qui le istruzioni da eseguire
        se la condizione è falsa */
  }
```

Un modo invece di realizzare i cicli è quello di usare il costrutto `while(...)` che in C s'implementa così:

```
while (/* inserire al posto di questo commento la condizione per cui il ciclo
          termina se essa è falsa (ad es. i < 65) */ )
  {
    /* inserire qui le istruzioni che verranno eseguite ad ogni ciclo */
  }
```

Il ciclo termina quando la condizione, che va inserita tra le parentesi tonde dopo il while, è falsa. Ad esempio, per realizzare in C un ciclo che stampi i numeri interi da 0 a 64 su righe separate, potete usare il seguente codice:

```
i=0;
while (i < 65)
  {
     printf("i=%d\n", i);
     i = i + 1;
  }
```

dove la variabile intera `i` andrà opportunamente dichiarata dopo la parentesi graffa che si trova dopo l'istruzione `int main(void)`. Analogamente se volessimo stampare 65 punti equispaziati nell'intervallo $[-5, 5]$ potremmo usare il seguente codice C:

```
i = 0;
x = 0.0;
Np = 65;
xmin = -5.0;
xmax = 5.0;
dx = (xmax - xmin)/(Np-1);
while (i < Np)
  {
     x = xmin + dx*i;
     printf("x=%.7f\n", x);
     i = i + 1;
  }
```

dove si ricorda ancora una volta che le variabili in virgola mobile `x`, `xmin`, `xmax` e `dx` e la variabile intera `Np` vanno anch'esse dichiarate.
Sfruttando quanto abbiamo appena visto, scrivere un programma `sinxx.c` che, utilizzando un ciclo `while` calcoli il valore della funzione $f(x) = \frac{\sin(x)}{x}$ in una serie di punti equispaziati lungo l'asse $x$, nell'intervallo $x \in [-15, 15]$. *Suggerimento: si ricordi che per usare la funzione matematica* `sin` *è necessario aggiungere la direttiva al preprocessore* `#include<math.h>` *all'inizio del file e che per compilare è necessario utilizzare il flag -lm.* Si noti che:

1. Tutto il materiale di questa esercitazione si dovrà trovare in una cartella chiamata `EX4` in cui dovreste "spostarvi" con il comando da terminal `cd` dopo averla creata con il comando `mkdir`.

2. Il numero di punti $N_p$ deve essere sufficiente per ottenere un grafico ragionevolmente continuo della funzione (ad es. $N_p = 129$);

3. Se la funzione viene calcolata nel punto $x = 0$ il programma restituisce il valore NAN (*not a number*), dal momento che si tratta di valutare un'espressione del tipo $\frac{0}{0}$. Il limite per $x \to 0$ di $\frac{\sin(x)}{x}$ è noto e vale: $\lim_{x\to 0} \frac{\sin(x)}{x} = 1$. Inserire nel programma un'opportuna istruzione **`if...else`** che corregga il risultato numerico con il valore analitico corretto in $x = 0$.

4. Scrivere i risultati su un file `sinxx.dat` di $N_p$ righe e 2 colonne, nella forma: `x y`. Vi ricordo che per compiere questa operazione potete o incollare manualmente l'output dello schermo su un file vuoto, o *ridirezionare* l'output del vostro programma su un file con il comando:

   ```
   ./sinxx.exe > sinxx.dat
   ```

   dove si assume che il nome dell'eseguibile che avete creato con la compilazione sia `sinxx.exe`.

5. Creare con `python` un grafico della funzione $\frac{\sin(x)}{x}$ utilizzando i punti calcolati e salvarlo sul file `sinxx.png`. Quanti zeri ci sono nell'intervallo $[-15, 15]$? Dove sono?

---

► **Seconda parte:**
Scrivere un programma `zero.c` che calcoli automaticamente il numero degli zeri della funzione $f(x) = \frac{\sin(x)}{x}$ nell'intervallo $[a, b] = [-15, 15]$ mediante il seguente algoritmo, basato sul teorema degli zeri di

una funzione continua:

1. L'intervallo di partenza viene diviso in $N$ intervalli contigui (attenzione: $N = N_p - 1$), tutti uguali, di estremi $x_L^i, x_R^i$, con $i = 0, \cdots, (N-1)$.
2. Per ciascun intervallo l'algoritmo controlla la presenza di eventuali zeri calcolando il valore di $s = f(x_L^i) \cdot f(x_R^i)$. Se $s > 0$, non ci sono zeri; se $s \leq 0$, c'è uno zero.
3. Alla fine dell'esecuzione, il programma restituisce il numero totale di zeri trovati dall'algoritmo, $\mathrm{N}_{zeri}$.

Questo algoritmo andrà dunque implementato con un ciclo sugli $N$ intervalli (e non come nella prima parte sul numero di punti $N_p$), dove la funzione $\frac{\sin(x)}{x}$ andrà valutata per i due estremi di ogni intervallo. Nel codice che scriverete si richiede che, se un estremo dell'intervallo è nullo, vada calcolato il valore numerico corretto della funzione come già fatto nella prima parte, ovvero utilizzando opportunamente il costrutto `if...else`.
*Suggerimento: copiate il programma creato nella prima parte con il comando* `cp` *nel file chiamato* `zero.c` *e poi modificatelo opportunamente.*

► **Attenzione!** A volte, per via di un errore di programmazione, può succedere che un programma entri in un *loop infinito*. Se questo succede, ci sono due modi di interrompere l'esecuzione:

- Se il programma è in esecuzione nella *shell*: premere la combinazione di tasti `CTRL+C`.
- Se il programma è in esecuzione in modalità *background*: aprire una nuova shell. Lanciare il programma `top` (non in modalità *background*). Individuare il PID (*Process Identifier*) del programma bloccato. Premere `q` per uscire da top. Digitare della shell il comando: `kill -9 PID` (Enter).