LAB 00 - Primi passi nella programmazione in C

1 Obiettivi della sessione di laboratorio

Gli **obiettivi** di tale esercitazione sono l'installazione e un primo utilizzo del compilatore e di un editor. Si consiglia di provare autonomamente (prima della sessione di laboratorio) a installare il software e a svolgere gli esercizi. Nella prima parte del laboratorio in ogni caso verranno svolti semplici esempi di programmazione assistita per prendere confidenza con l'ambiente di programmazione, quindi non preoccupatevi se dovessero sorgere problemi in questa fase. Nel caso vi siano problemi di installazione o perplessità sugli esercizi il tutor è a disposizione non solo durante l'esercitazione, ma anche nel forum sulla piattaforma moodle (di cui si consiglia l'utilizzo) e via e-mail. Ecco quindi che i consigli per prendere confidenza con la programmazione sono i seguenti:

1. Lavorare e *"sbatterci la testa"* fin da subito. I laboratori sono caratterizzati da una difficoltà crescente e se non sono assodati i concetti dei laboratori precedenti diventerà sempre più difficile poi recuperarli.

2. Non avere problemi a fare domande, anche le più banali. Le attività di laboratorio sono affidate al servizio di **tutorato**, che per sua natura non deve *valutare* ma *supportare* lo studente.

3. Utilizzare tutte le risorse a disposizione, in particolar modo il forum su moodle, strumento fondamentale per il confronto *guidato*.

2 Primi passi e struttura del file

Dopo aver impostato[2] nel proprio PC il compilatore e un editor, il primo passo consiste nel creare un nuovo file sorgente (.c). In questi primi semplici esercizi tale file avrà una struttura di questo tipo:

1. Direttive a preprocessore (si valuti la necessità, ad esempio, di includere qualche libreria standard mediante la direttiva #*include*);

2. La funzione *main*.

Nel seguito un esempio della struttura del file da utilizzare.

```c
#include </* fill in */>

int main (void)
{

  /* fill in */

  return 0;
} /* main */
```

structure.c

Creare quindi una cartella nella quale inserire tutti i file necessari per questa esercitazione (si consiglia di inserirla all'interno di una cartella dedicata al laboratorio) e inserirci il file sorgente completo. Aprire quindi il **terminale** e muoversi nella cartella nella quale il file sorgente è stato salvato[1]. A questo punto è possibile procedere alla compilazione e al linking, mediante il comando da terminale:

gcc -Wall -Werror -o pun pun.c

Mediante tale comando è stato creato, a partire dal sorgente contenuto nel file *pun.c* (è un esempio), un eseguibile dal nome *pun* ed è stato salvato nella cartella stessa. Mentre quindi l'ultimo campo (nell'esempio *pun.c*) deve essere il nome del file sorgente, il penultimo (in questo caso *pun*) è arbitrario ed è il nome che il programmatore attribuisce all'eseguibile, che non per forza deve essere lo stesso del sorgente. Inoltre:

gcc richiama il compilatore installato;

[1] In linux basta fare tasto destro su una sezione vuota della cartella desiderata e selezionare *"apri terminale qui"*. In Windows fare "esegui → cmd" e poi muoversi nella cartella desiderata. Il video *"Tutorial su File System e Terminali"*, presente su moodle, spiega bene la natura del file system e il funzionamento del terminale. Se ne consiglia la visione data l'importanza degli argomenti trattati. Durante la sessione di laboratorio una breve introduzione al *file system* e all'utilizzo del *terminale* verrà presentata.

-Wall è un flag opzionale che permette di visualizzare tutti i *warnings*;

-Werror è un flag opzionale che permette di trattare tutti i *warnings* come *errori*[2].

A questo punto procedere con l'esecuzione del programma mediante la seguente istruzione:

./pun *Se si lavora in ambiente linux* **pun.exe** *Se si lavora in ambiente windows*

3 Consegne

Vengono proposti nel seguito degli esercizi di difficoltà crescente, che quindi si consiglia di svolgere nell'ordine dato.

3.1 Perimetro di un triangolo rettangolo

Si completi, utilizzando direttamente il file sorgente fornito, il seguente codice in C che, ottenuti mediante funzione *scanf()* i valori c_1 e c_2 dei cateti di un triangolo rettangolo, ne calcoli il perimetro.

[2] Attenzione alla differenza tra i due concetti. Un *warning* è solo un consiglio che il compilatore da` al programmatore ma l'eseguibile viene creato, un *errore* è invece associato a un errore sintattico che non permette la creazione dell'eseguibile. L'utilizzo dei due flag opzionali è consigliato perchè molto spesso i warnings sono associati ad errori logici che, pur non costituendo errore sintattico, possono portare a risultati errati nell'esecuzione del codice

```c
#include </* to fill in */>
#include </* to fill in */>

/* main function definition */
int main (void)
{
  float c1, c2, /* to fill in */;

  /* Print the system menu */
  printf("\n/************** Calculation of the perimeter of a
    right triangle *************/\n\n");

  /* Reading of the first cathetus from the user */
  printf("-> Insert the values of the cathetus <-\n");
  printf("c1 = ");
  scanf("%f", &c1);

  /* to fill in */

  /* Print the perimeter of the triangle*/
  printf("\n-> Perimeter of the triangle <-\n");
  printf("2p = %f\n\n", perimeter);

  return 0;
} /* end of main */
```

1 triangle.c

Consigli

1. Utilizzare la funzione *sqrt()*, contenuta nella libreria *math.h*;
2. Per calcolare un valore al quadrato è lecito chiedersi se esista una funzione specifica in *math.h*, provare a googlare e verificarne la presenza. Attenzione che non è strettamente necessaria tale funzione!
3. Al fine di linkare correttamente alcune librerie a volte è necessario utilizzare flag aggiuntivi nel comando di compilazione.

La libreria *math.h* necessita del flag *-lm*. D'ora in poi non verranno più forniti tali suggerimenti: nel manuale[3] infatti queste informazioni sono presenti (ci si riferisca a Figura 1). Il manuale consigliato permette quindi di cercare le funzioni desiderate e avere informazioni circa le librerie da includere, i flag necessari, i parametri e i valori di ritorno.

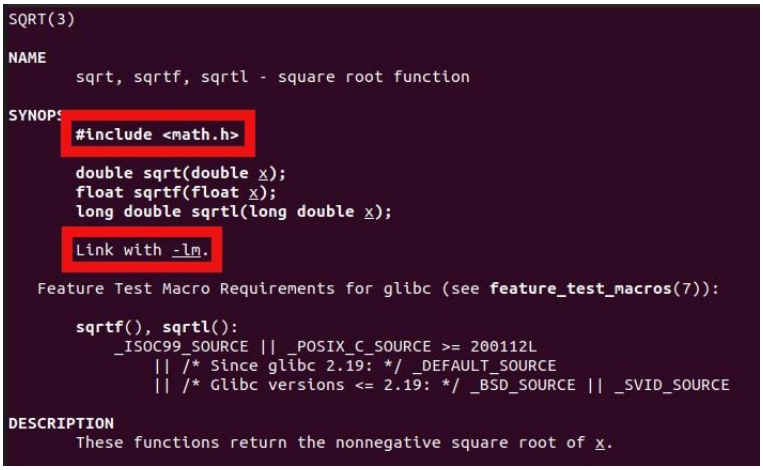

Figura 1: Esempio di uso del manuale *"Linux Programmer's Manual"*, dove è stata effettuata una ricerca sulla funzione *sqrt()*.

.

3.2 Il resto della divisione

Si scriva un programma C che calcoli il resto della divisione tra due interi positivi senza usare l'operatore %.

Consigli

1. Inizialmente si consideri il caso in cui il divisore sia non nullo. In generale, un obiettivo del programmatore è quello di rendere il codice *"robusto"*, ossia capace di rispondere correttamente a tutti i possibili set di valori di input. Tale obiettivo però esula dagli scopi della prima

[3] Per accedere al manuale su linux da terminale usare il comando *"man funzione"* (nel nostro caso ad esempio *"man sqrt"*). Altrimenti googlare *"Linux Programmer's manual, e cercare la funzione di interesse."*.

esercitazione, sarà una capacità da acquisire nel tempo. Potete eventualmente pensare di gestire il caso particolare (**è facoltativo**) solo al termine di tutti gli esercizi.

3.3 La chiocciola (FACOLTATIVO)

Una chiocciola debba risalire un pozzo profondo *P* metri. Partendo dal fondo, la chiocciola risale *D* metri durante le ore diurne, ma slitta verso il basso di *N* metri durante le ore notturne

Si scriva un programma in C che, ottenuti i valori di *P*, *D* e *N*, calcoli il numero *G* di giorni necessari per uscire dal pozzo.