

# Introduzione agli strumenti

Laboratorio di Fondamenti di Programmazione

---

Chiara Bodei, *Damiano Di Francesco Maesa*, *Roberta Gori*

CdL in Matematica, Università di Pisa

a.a. 2023/2024

## Docenti

- [teoria] Chiara Bodei
- [pratica] Damiano Di Francesco Maesa
- [pratica] Roberta Gori

## Assistenti

- Enrico Calandrini

# Strumenti

Le esercitazioni di laboratorio saranno svolte dagli studenti usando i propri portatili con i propri strumenti di sviluppo. Sono richiesti:

- **Editor di testo:** per scrivere il codice degli esercizi.
- ● **Terminale:** per eseguire il compilatore ed ottenere un eseguibile.
- ● **Compilatore:** GCC per Linguaggio C.
- **Piattaforma EVO:** per sottomettere l'eseguibile per testing e attestare il superamento dell'esercizio.

Editor di testo e terminale dipendono dal vostro sistema operativo e preferenza personale.

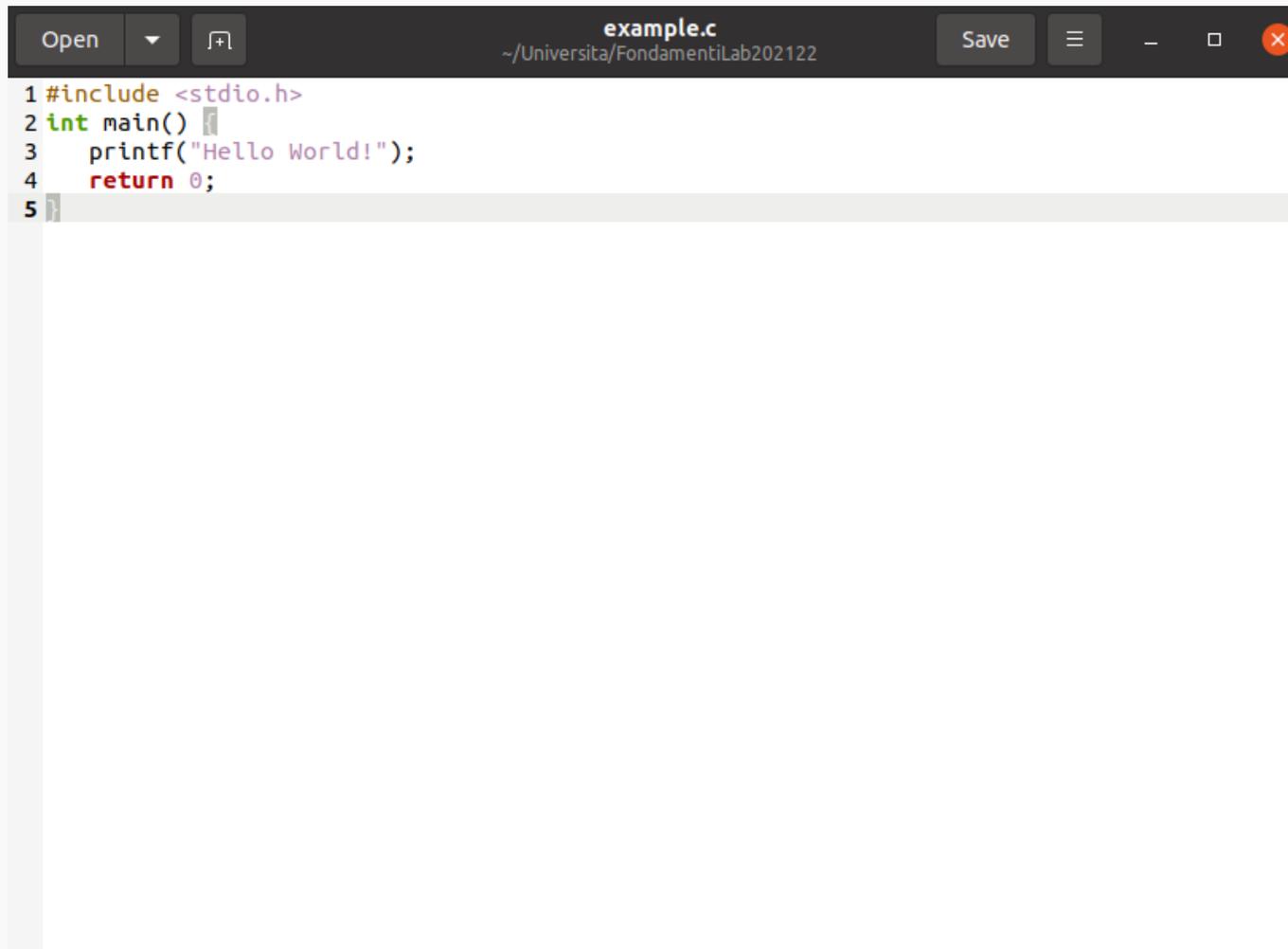
Vi chiediamo di seguire le istruzioni nel file GCCinstructions nella cartella su teams per scaricare un compilatore GCC (e il corrispondente terminale dove necessario).

# Editor testuale

---

# Editor

Semplice come blocco note (salviamo file di testo con estensione “.c”)



The image shows a screenshot of a code editor window. The window title is "example.c" and the path is "~/Universita/FondamentiLab202122". The editor contains the following C code:

```
1 #include <stdio.h>
2 int main() {
3     printf("Hello World!");
4     return 0;
5 }
```

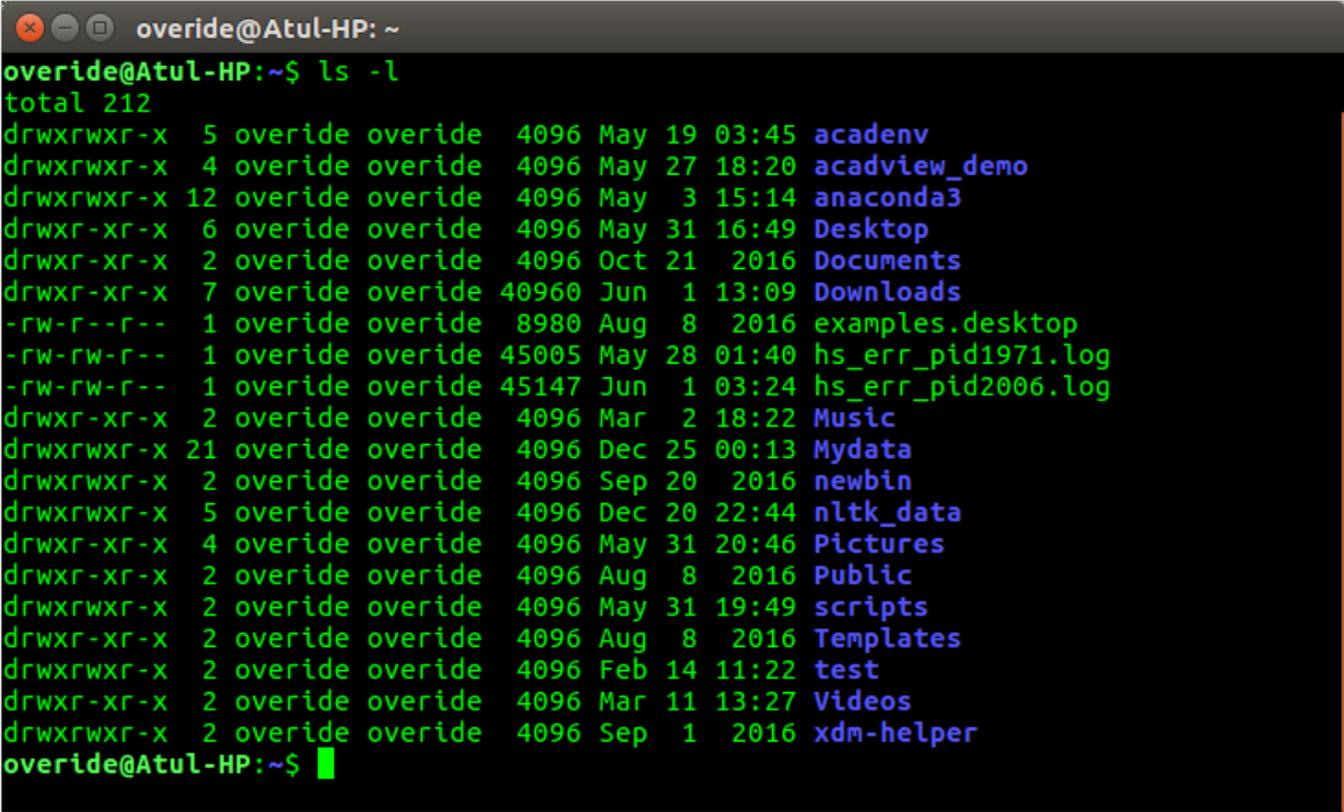
# Terminale

---

# Editor

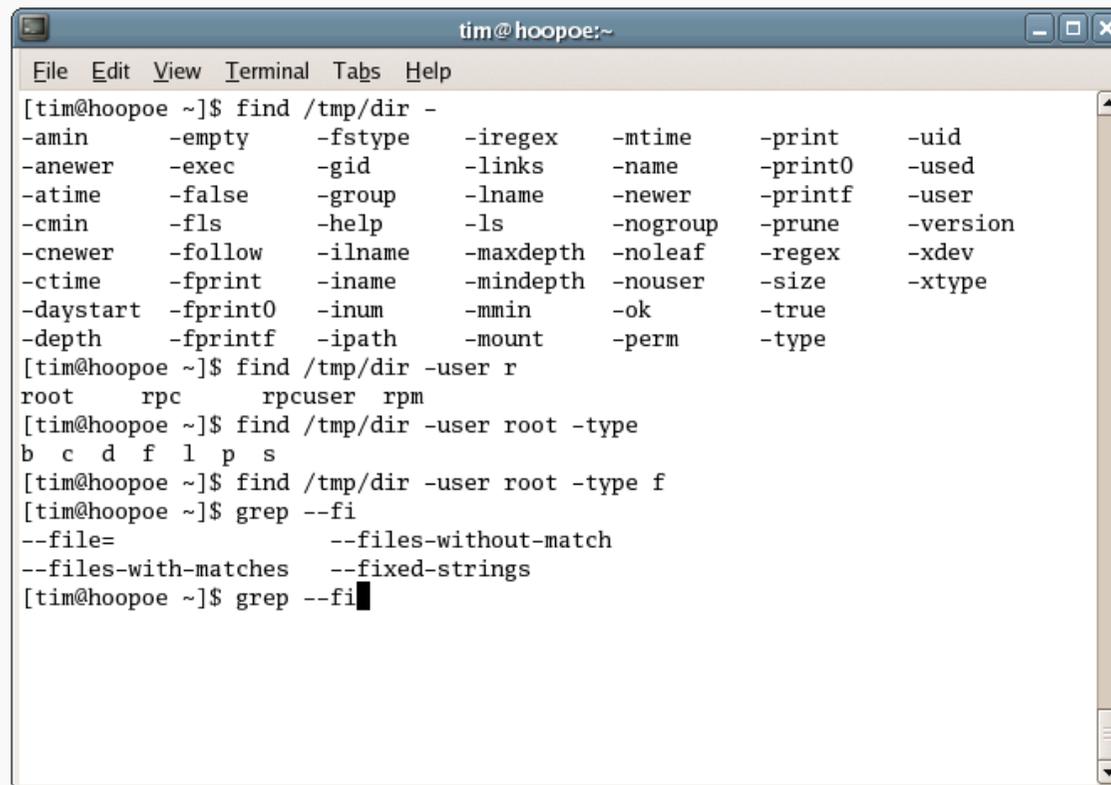
Possiamo vedere un terminale come un programma (“shell”) che permette di indicare istruzioni direttamente al vostro sistema operativo.

Verrà usato durante il corso per indicare alla macchina di eseguire un determinato programma C da noi creato.



```
override@Atul-HP: ~  
override@Atul-HP:~$ ls -l  
total 212  
drwxrwxr-x  5 override override 4096 May 19 03:45 acadenv  
drwxrwxr-x  4 override override 4096 May 27 18:20 acadview_demo  
drwxrwxr-x 12 override override 4096 May  3 15:14 anaconda3  
drwxr-xr-x  6 override override 4096 May 31 16:49 Desktop  
drwxr-xr-x  2 override override 4096 Oct 21  2016 Documents  
drwxr-xr-x  7 override override 40960 Jun  1 13:09 Downloads  
-rw-r--r--  1 override override 8980 Aug  8  2016 examples.desktop  
-rw-rw-r--  1 override override 45005 May 28 01:40 hs_err_pid1971.log  
-rw-rw-r--  1 override override 45147 Jun  1 03:24 hs_err_pid2006.log  
drwxr-xr-x  2 override override 4096 Mar  2 18:22 Music  
drwxrwxr-x 21 override override 4096 Dec 25 00:13 Mydata  
drwxrwxr-x  2 override override 4096 Sep 20  2016 newbin  
drwxrwxr-x  5 override override 4096 Dec 20 22:44 nltk_data  
drwxr-xr-x  4 override override 4096 May 31 20:46 Pictures  
drwxr-xr-x  2 override override 4096 Aug  8  2016 Public  
drwxrwxr-x  2 override override 4096 May 31 19:49 scripts  
drwxr-xr-x  2 override override 4096 Aug  8  2016 Templates  
drwxrwxr-x  2 override override 4096 Feb 14 11:22 test  
drwxr-xr-x  2 override override 4096 Mar 11 13:27 Videos  
drwxrwxr-x  2 override override 4096 Sep  1  2016 xdm-helper  
override@Atul-HP:~$ █
```

# Cosa è il terminale?



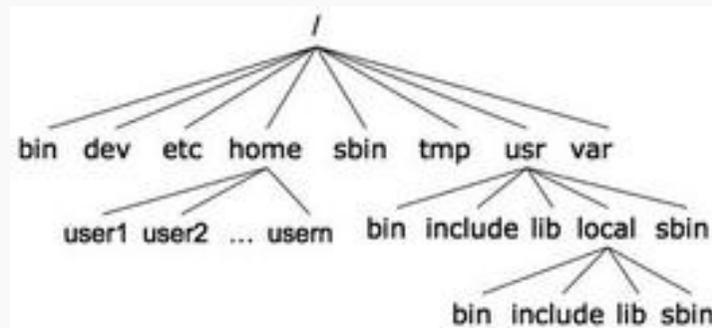
```
tim@hoopoe:~  
File Edit View Terminal Tabs Help  
[tim@hoopoe ~]$ find /tmp/dir -  
-amin      -empty     -fstype    -iregex    -mtime     -print     -uid  
-anewer    -exec      -gid       -links     -name      -print0    -used  
-atime     -false     -group     -lname     -newer     -printf    -user  
-cmin      -fls       -help      -ls        -nogroup   -prune     -version  
-cnewer    -follow    -ilname    -maxdepth  -noleaf    -regex     -xdev  
-ctime     -fprint    -iname     -mindepth  -nouser    -size      -xtype  
-daystart  -fprint0   -inum      -mmin      -ok        -true  
-depth     -fprintf   -ipath     -mount     -perm      -type  
[tim@hoopoe ~]$ find /tmp/dir -user r  
root      rpc      rpcuser  rpm  
[tim@hoopoe ~]$ find /tmp/dir -user root -type  
b c d f l p s  
[tim@hoopoe ~]$ find /tmp/dir -user root -type f  
[tim@hoopoe ~]$ grep --fi  
--file=          --files-without-match  
--files-with-matches --fixed-strings  
[tim@hoopoe ~]$ grep --fi
```

- Programma che fornisce un'interfaccia testuale alle funzionalità del sistema;
- Legge i comandi digitati dall'utente e li esegue (ad es. navigare sul file system, creare file e directory, eseguire programmi).

# Il file system

Un **file system** è il meccanismo fornito dal sistema operativo che regola l'organizzazione fisica e logica delle informazioni sui dispositivi (disco, cd-rom, dvd, ecc.).

Il file system può essere visto come un *albero*:

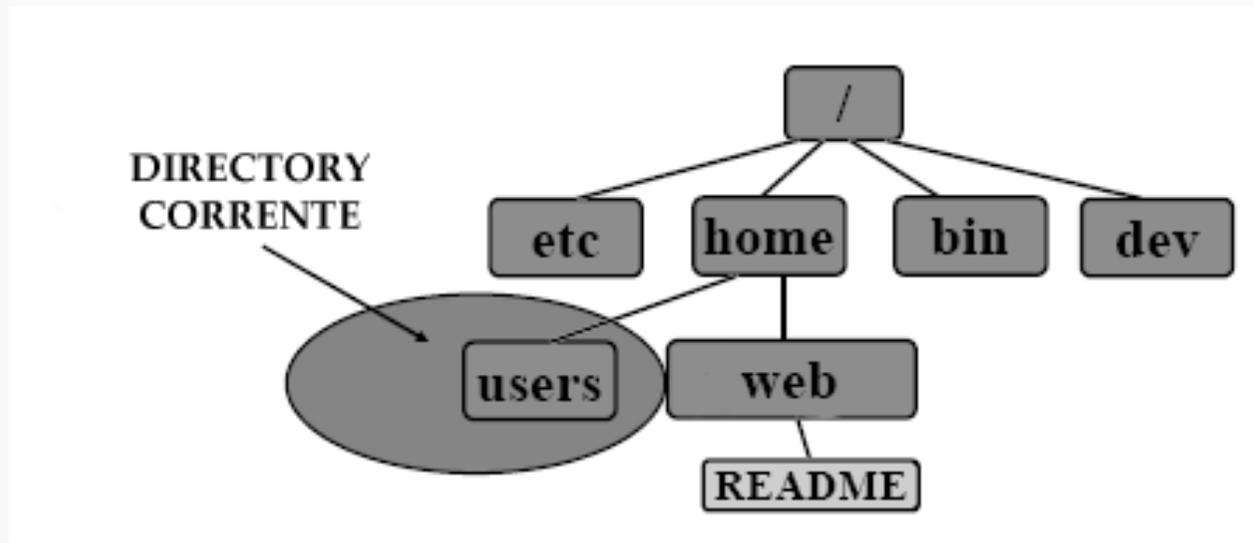


Ogni nodo dell'albero è o un file o una directory di file.

Un file, identificato da un **path name**.

Il path name di un file può essere **assoluto**, riferito alla radice della gerarchia ( / ), oppure **relativo**, riferito alla posizione dell'utente nel file system.

# Path assoluti/relativi



**PATH ASSOLUTO:** /home/web/README

**PATH RELATIVO:** ../web/README

# Navigare nel filesystem

Supponiamo di essere nella directory `/home/user` e di voler compilare il file `hello.c` che è nella directory `/home/user/documenti`.

Avete le due seguenti opzioni

- vi spostate nella directory `documenti` e poi chiamate il compilatore su `hello.c`,  
`cd <nomedirectory>` serve per muoversi attraverso le directory.

## Esempio

```
cd documenti  
gcc hello.c
```

- dalla directory in cui siete chiamate `gcc` con il path (relativo o assoluto) del file.

## Esempio

```
gcc documenti/hello.c
```

# Redirezione

Di default i comandi del terminale prendono l'input da tastiera (**standard input - stdin**) e mandano l'output ed eventuali messaggi di errore su video (**standard output - stdout, standard error - stderr**). L'input/output può essere rediretto da/verso file, utilizzando opportuni metacaratteri:

Metacarattere	Significato
>	ridirezione dell'output
>>	ridirezione dell'output (append)
<	ridirezione dell'input
<<	ridirezione dell'input dalla linea di comando

Un processo è un programma in esecuzione.

**Il terminale esegue ripetutamente i seguenti passi:**

- stampa il prompt e attende l'input dell'utente;
- legge la linea di comando (e espande le eventuali alias e wildcard);
- lancia un processo per eseguire il comando mettendosi in attesa;
- quando l'esecuzione del comando termina, riprende il controllo.

# Comandi per la gestione dei processi

- CTRL-z: combinazione di tasti che sospende il comando in esecuzione
- CTRL-c: **combinazione di tasti che termina il comando in esecuzione**

# Compilazione

---

# Dal sorgente all'eseguibile

Prima di poter essere eseguito, un programma deve essere

1. pre-processato (pre-processing)
2. compilato (compiling)
3. collegato (linking)
4. caricato in memoria (loading)

I file testuali .c sono parte del **codice sorgente** di un programma

- Il **compilatore** trasforma il **sorgente .c** in **codice oggetto .o** (binario)
- Il **linking** collega i file oggetto in un **eseguibile**

In questa fase ci concentriamo su **pre-processore** e **compilatore**

- Fase preliminare alla compilazione
- Comporta la **sostituzione di informazioni simboliche** (testo) nel codice sorgente con un contenuto che viene specificato dal programmatore mediante **direttive per il pre-processore**
- Le direttive per il pre-processore vanno scritte **in testa ai sorgenti C** e sono precedute dal **simbolo #**
  - Inclusione file: `#include`
  - Macro: `#define`

# Il compilatore

Trasforma il file preprocessato (senza più `#include` o `#define`) in un file eseguibile che contiene il codice assembler **eseguibile dal processore target**

Il compilatore C esegue una serie di **controlli di correttezza**

- Sintassi dei comandi: ad es. terminazione con `;`, parentesi bilanciate, ...
- Coerenza dei tipi di dato: ad es. operatori algebrici usati solo con variabili numeriche, parametri delle funzioni,...
- Ogni variabile o funzione utilizzata deve essere stata dichiarata in precedenza

# GNU Compiler Collection (GCC)

- Sviluppato per sistemi Linux permette generare **eseguibili per diverse piattaforme target**
- Non è solo un **compilatore** ma è anche **pre-processor** e **linker**

Sintassi:

```
gcc [OPT] file.c
```

- `file.c` - Nome del file sorgente
- `[OPT]` - Insieme di opzioni che controllano le funzionalità di GCC (le vediamo nella prossima slide)
- il file eseguibile sarà memorizzato nel file `a.out`
- per vedere il risultato dell'esecuzione dobbiamo eseguire `a.out` con il comando `./a.out`

# Opzioni importanti di GCC

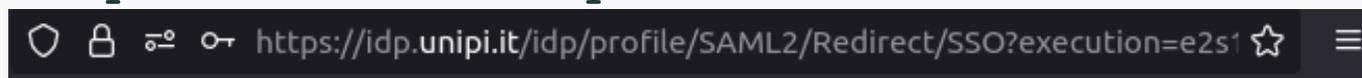
- `-o fileExec` - Imposta il nome del file eseguibile a `fileExec` (`a.out` di default)
- `-Wall -pedantic` opzioni che aumentano il numero di controlli e di messaggi di avvertimento (**warnings**) visualizzati

# Piattaforma di autovalutazione

---

# Piattaforma EVO

Per usare la piattaforma EVO learning accedere al link <https://evo.di.unipi.it/> con le credenziali di ateneo.



## UNIVERSITÀ DI PISA

**Nome utente**

**Password**

Non ricordare l'accesso

Mostra le informazioni che

[› Password dimenticata  
o Attivazione account](#)

[› Serve aiuto?](#)

[› Informazioni](#)

[› Politica sulla Privacy](#)

[› Politica di utilizzo \(AUP\)](#)

# Piattaforma EVO

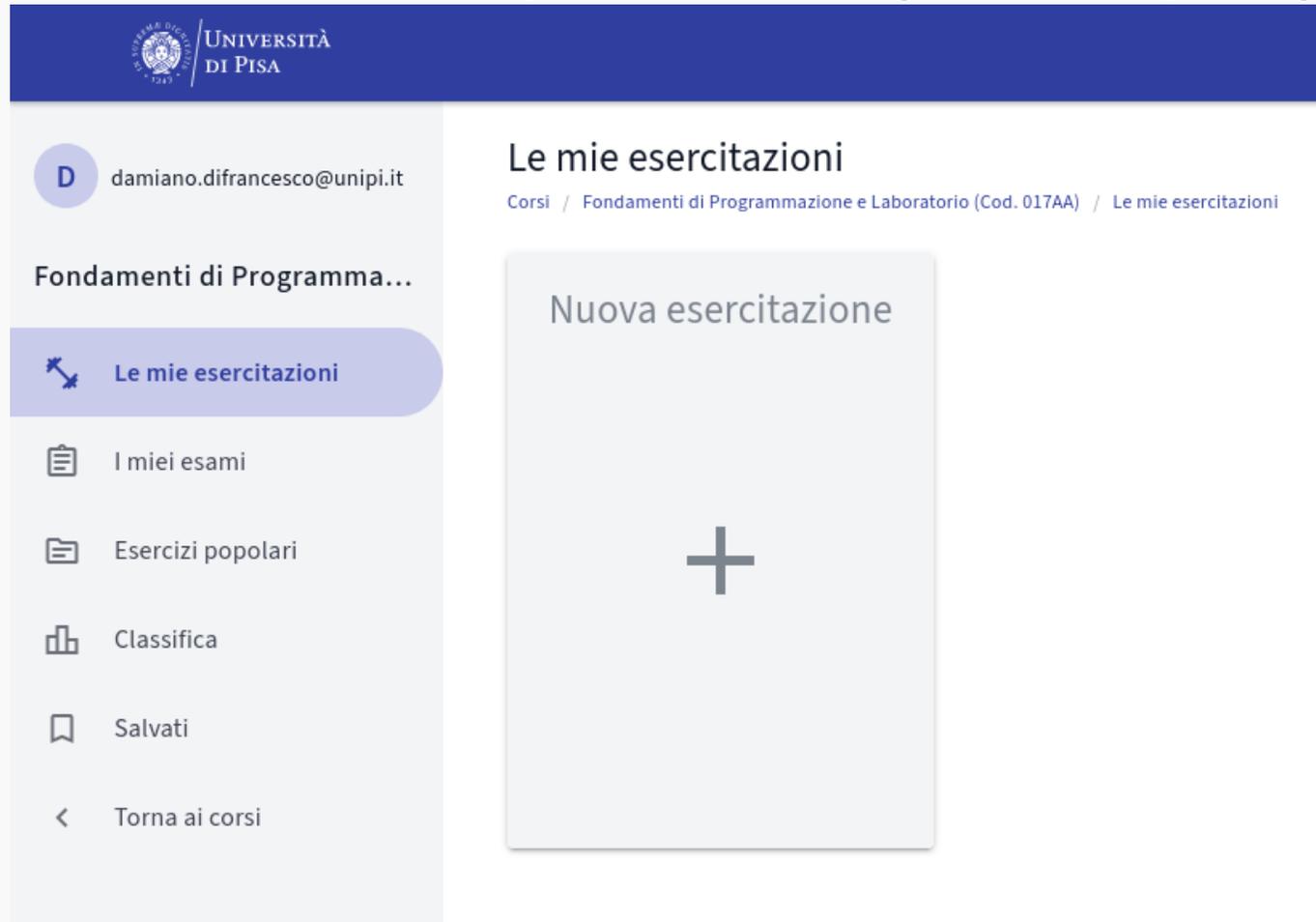
Dopo aver effettuato il login con le credenziali di ateneo dovrete poter vedere tutti i corsi a voi aperti.

The screenshot shows the 'Corsi' (Courses) page on the EVO platform. At the top, there is a search bar with the text 'Cerca corsi' and a filter option 'Mostra solo i miei c'. Below the search bar, there are six course cards arranged in a 2x3 grid. Each card contains the course title, the instructor's name, a brief description, and a button labeled '> Entra come studente'.

Course Title	Instructor	Description
Analisi	CARLO ROMANO GRISANTI	Esercizi del corso di Analisi Matematica della Laurea Triennale in Informatica dell'Università di Pisa
Programmazione e Algoritmica - A	GIUSEPPE PRENCIPE	Materiale dell'insegnamento di Programmazione e Algoritmica (A), della Laurea Triennale in Informatica dell'Università di Pisa
Laboratorio I	GIUSEPPE PRENCIPE	Materiale dell'insegnamento di Laboratorio I, della Laurea Triennale in Informatica dell'Università di Pisa
Laboratorio 2	ALINA SIRBU	
Corso di prova	FEDERICA PAGANELLI	
Matematica di base	CARLO ROMANO GRISANTI	

Cercare il corso **“Fondamenti di Programmazione e Laboratorio (Cod. 017AA)”**

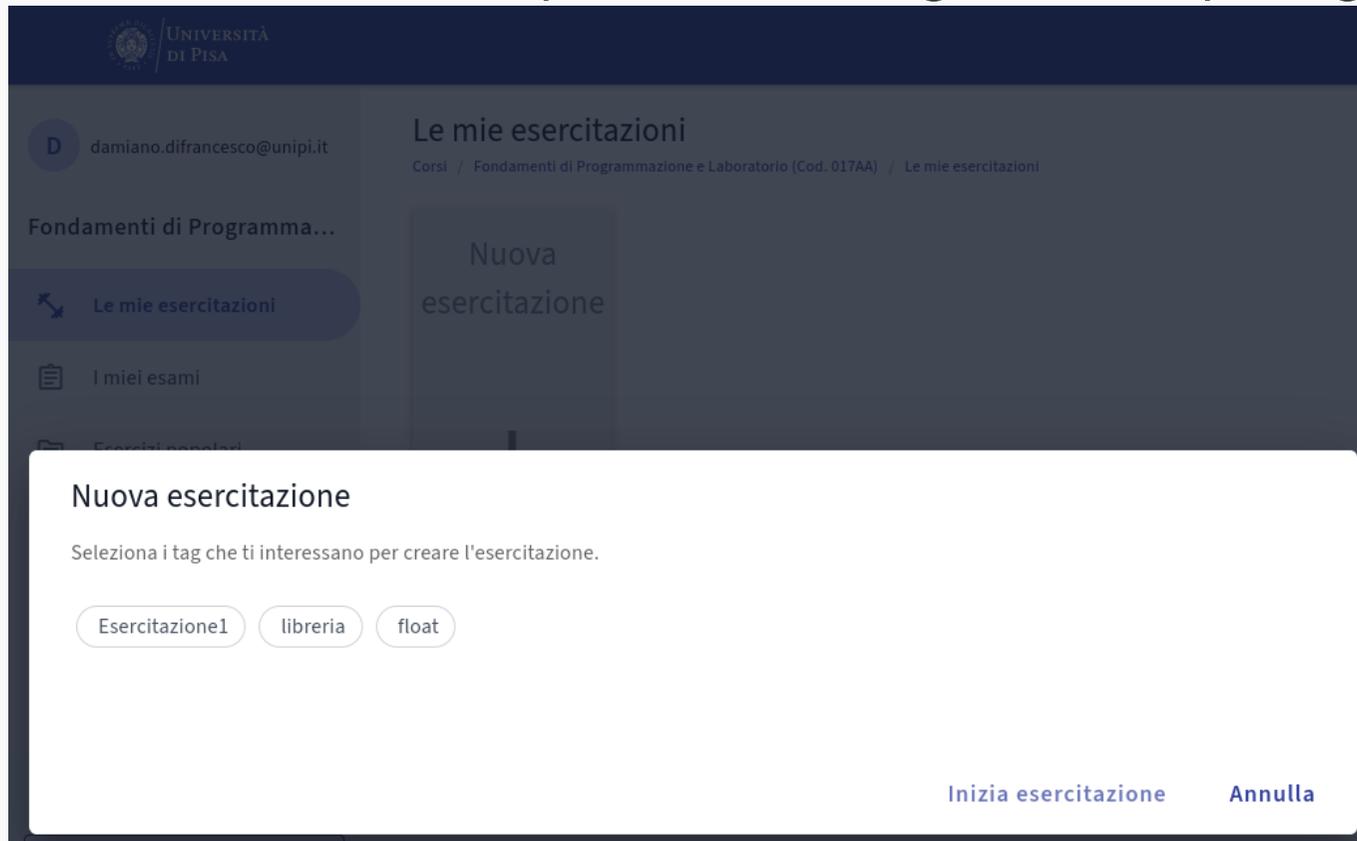
All'interno del corso potete trovare gli esercizi per ogni lezione.



The screenshot displays the user interface of the EVO platform. At the top, there is a dark blue header with the University of Pisa logo and name. Below this, a light blue sidebar contains the user's profile (damiano.difrancesco@unipi.it) and a list of navigation options: 'Fondamenti di Programma...', 'Le mie esercitazioni' (highlighted), 'I miei esami', 'Esercizi popolari', 'Classifica', 'Salvati', and 'Torna ai corsi'. The main content area is titled 'Le mie esercitazioni' and shows the breadcrumb path 'Corsi / Fondamenti di Programmazione e Laboratorio (Cod. 017AA) / Le mie esercitazioni'. A large, light gray button with a plus sign and the text 'Nuova esercitazione' is centered in the main area.

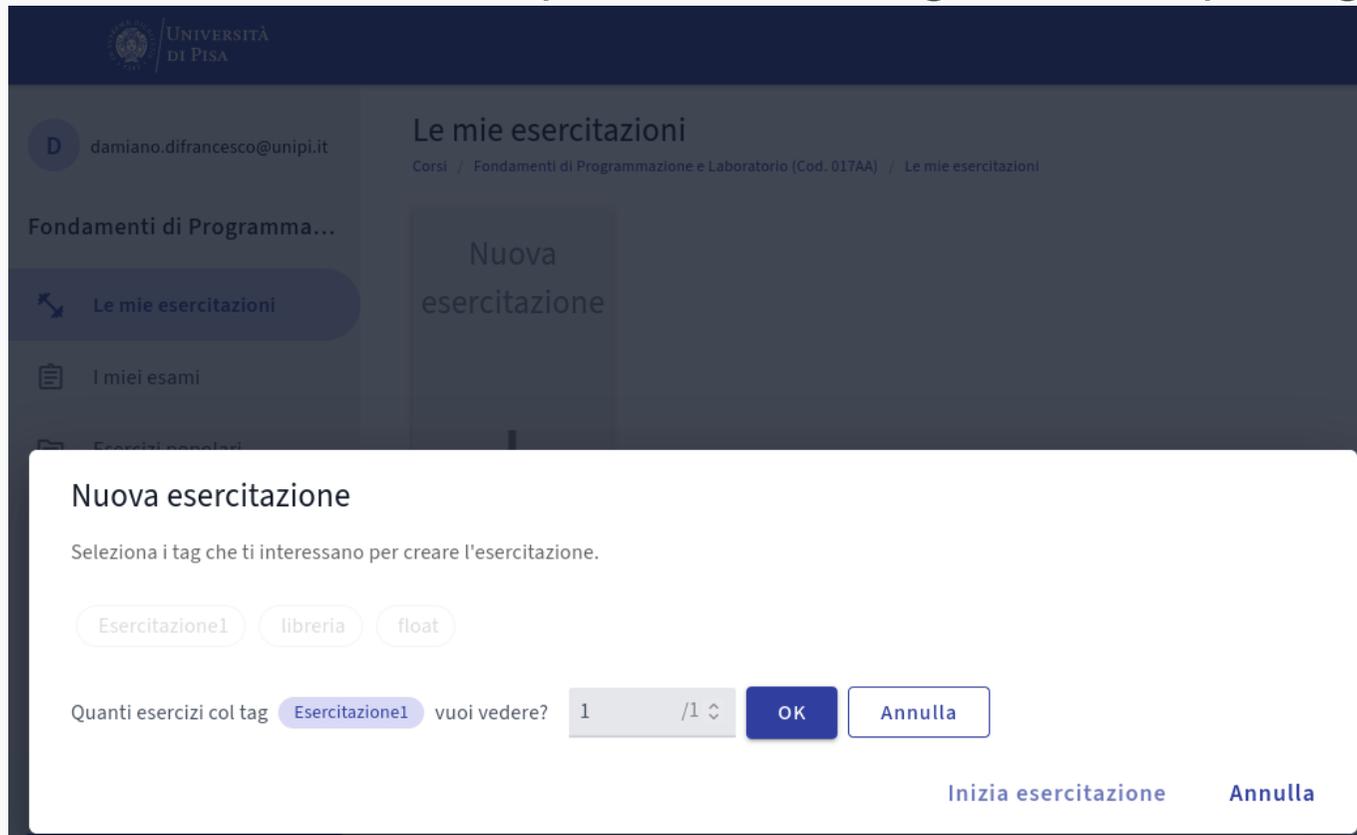
Cercare gli esercizi di una determinata esercitazione attraverso il tag **“esercitazioneX”**

All'interno del corso potete trovare gli esercizi per ogni lezione.



Cercare gli esercizi di una determinata esercitazione attraverso il tag **“esercitazioneX”**

All'interno del corso potete trovare gli esercizi per ogni lezione.



Cercare gli esercizi di una determinata esercitazione attraverso il tag **“esercitazioneX”**

# Piattaforma EVO

Per ogni esercitazione potete vedere il testo dell'esercizio e dei test proposti. Potete poi sottomettere e valutare la vostra soluzione scrivendo/copiando il vostro codice ed eseguendo.

Esercitazione in corso 

Corsi / Fondamenti di Programmazione e Laboratorio (Cod. 017AA) / Esercitazione

### Esercizio 1

Esercitazione1 libreria float

1 

Scrivere il programma Trilsoscele che chiede all'utente due interi che rappresentano la base e l'altezza di un triangolo isoscele, li legge e stampa il perimetro e l'area del triangolo rappresentato su due righe come nell'esempio. Per calcolare la radice quadrata utilizzare la funzione `sqrt(x)` della libreria `math.h` (utilizzare la direttiva `#include <math.h>` e compilare aggiungendo l'opzione `-lm` alla fine).

**Esempio:**

Input	Output
45 92	127487
7 157	500000





# Piattaforma EVO

Per ogni esercitazione potete vedere il testo dell'esercizio e dei test proposti. Potete poi sottomettere e valutare la vostra soluzione scrivendo/copiando il vostro codice ed eseguendo.

The screenshot shows the EVO platform interface for an exercise. At the top, there is a header "Esercitazione in corso" with a breadcrumb trail "Corsi / Fondamenti di Programmazione e Laboratorio (Cod. 017AA) / Esercitazione" and a cloud icon. Below this, the exercise is titled "Esercizio 1" with sub-labels "Esercitazione1", "libreria", and "float". A dark blue code editor on the left contains a single line of code with the number "1" and a green "Esegui" button. To the right, there are three tabs: "Testo", "Risultato", and "Test case". The "Test case" tab is active, showing two test cases. Each test case has an "Input" field and an "Output atteso" field. Test case 1 has input "45 7" and expected output "92.127487" and "157.500000". Test case 2 has input "67" and "9" and expected output "136.375793" and "301.500000". A green "Consegna" button is located at the bottom right of the interface.

Esercitazione in corso

Corsi / Fondamenti di Programmazione e Laboratorio (Cod. 017AA) / Esercitazione

Esercizio 1

Esercitazione1 libreria float

1 ▶ Esegui

Testo <> Risultato Test case

Test case 1

Input

45 7

Output atteso

92.127487  
157.500000

Test case 2

Input

67  
9

Output atteso

136.375793  
301.500000

✓ Consegna

# Piattaforma EVO

Per ogni esercitazione potete vedere il testo dell'esercizio e dei test proposti. Potete poi sottomettere e valutare la vostra soluzione scrivendo/copiando il vostro codice ed eseguendo.

Esercitazione in corso 🔗

Corsi / Fondamenti di Programmazione e Laboratorio (Cod. 017AA) / Esercitazione

### Esercizio 1

Esercitazione1 libreria float

```
1 #include <stdio.h>
2 #include <math.h>
3 int main()
4 {
5     int x,y;
6     scanf("%d%d",&x,&y);
7     printf("Ho letto %d\n%d\n", x,y);
8     return 0;
9 }
```

Esegui

Testo <> Risultato Test case

#### <> Risultato esecuzione

Test case 1 **✗ Non superato**

Input

```
45 7
```

Output atteso

```
92.127487
157.500000
```

Il programma ha prodotto il seguente output:

```
Ho letto 45
7
```

Test case 2 **✗ Non superato**

Input

```
67
9
```

✓ Consegna

Una volta consegnato potrete vedere un riepilogo con il vostro punteggio. Ovviamente non conviene consegnare esercizi che già non passano i test...

✓ Fatto! Hai consegnato con successo. Qui sotto puoi rivedere le tue risposte. Quando vuoi, puoi chiudere questa pagina.

🕒 Data e ora di inizio: 03 ottobre 2022, alle 16:59

📅 Data e ora di consegna: 03 ottobre 2022, alle 17:06

Voto: 0

## Esercizio 1

Scrivere il programma Trisoscele che chiede all'utente due interi che rappresentano la base e l'altezza di un triangolo isoscele, li legge e stampa il perimetro e l'area del triangolo rappresentato su due righe come nell'esempio. Per calcolare la radice quadrata utilizzare la funzione `sqrt(x)` della libreria `math.h`

## Riepilogo esercitazione

Corsi / Fondamenti di Programmazione e Laboratorio (Cod. 017AA) / Riepilogo esercitazione

Input Output

45 92.127487

7 157.500000



Punteggio: 0 su 1

Risposta

```
#include <stdio.h>
#include <math.h>
int main()
{
    int x,y;
```

# Esercitazione

---

# Esercizio 1

- Scrivere in un editor il codice nella slide seguente su un file `main.c`.
- Aprite una `shell` e posizionatevi sulla directory dove avete salvato `main.c`
- Compilate `main.c` usando `GCC` ed eseguitelo
- Interrompere l'esecuzione del comando in modo da riottenere il prompt (CTRL-C).

# main.c

```
1 #ifdef _WIN32
2 #include <Windows.h>
3 #else
4 #include <unistd.h>
5 #endif
6 #include <stdio.h>
7
8 int main() {
9     while(1<2){
10         printf("Stop me if you can...\n");
11         sleep(1);
12     }
13     return 0;
14 }
```

Se il programma precedente non compila provate il seguente

```
1 #include <stdio .h>
2
3 int main() {
4     printf("Stop me if you can!\n");
5     while(1<2){}
6     return 0;
7 }
```

## Esercizio 2

1. Create un file `hello.c`
2. Scrivete in `hello.c` il vostro personale `Hello World!` come riportato nella prossima slide (senza commenti)
3. Aprite una `shell` e posizionatevi sulla directory dove avete salvato `hello.c`
4. Compilate `hello.c` usando `GCC` ed eseguitelo

# Hello World

```
1  #include <stdio.h>  /* Libreria standard per IO */
2
3  /* Il main definisce il punto di partenza del nostro
   programma.
4  * void -> non prende parametri in ingresso (in
   questo caso)
5  * int -> restituisce un intero
6  */
7  int main(void) {
8
9      /* Stampa a schermo la stringa passata come
   argomento */
10     printf("Hello World!\n"); /* \n carattere di escape
   speciale */
11
12     /* Valore restituito dal main al S.O. (0 -> OK) */
13     return 0;
14 }
```