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

Chi siamo

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

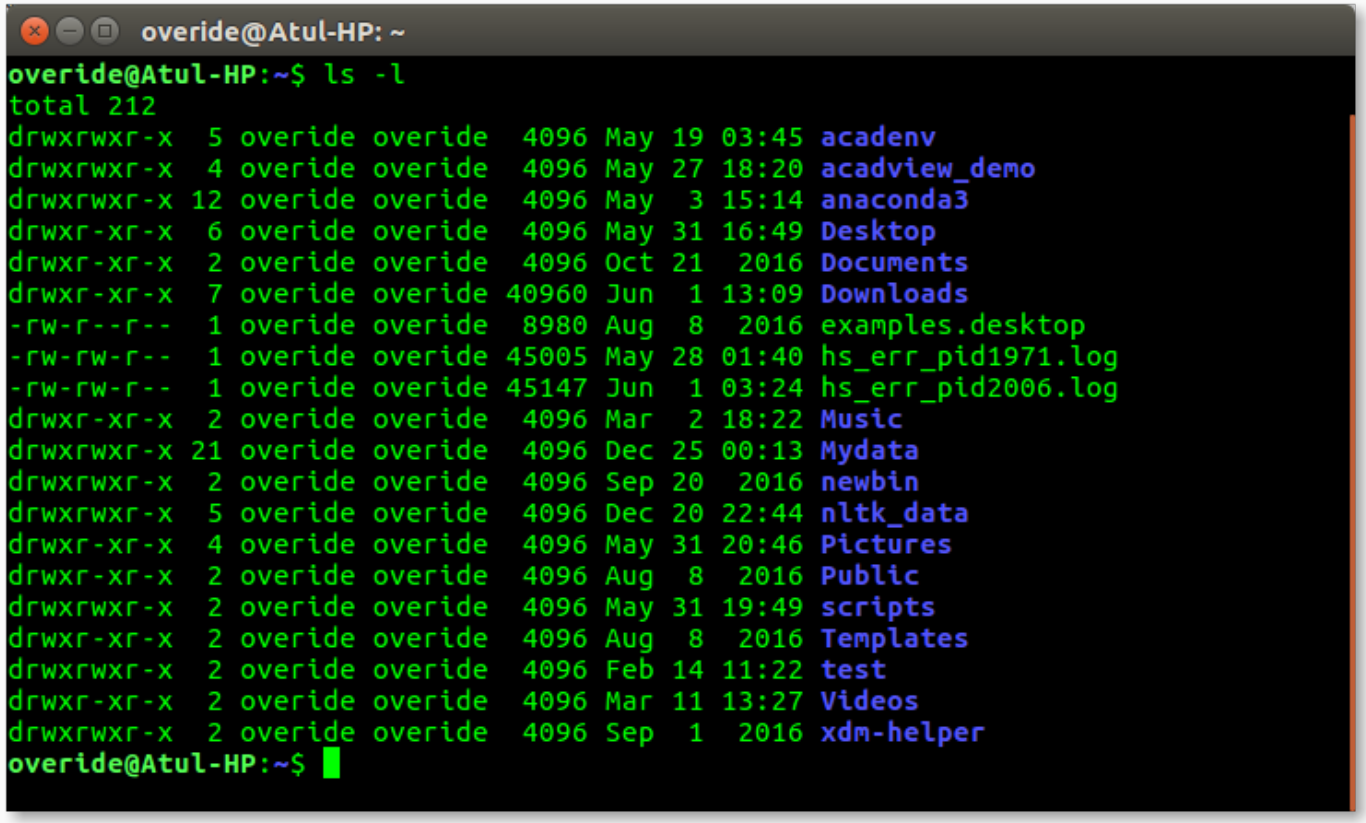
Semplice come blocco note (salviamo file di testo con estensione ".c")

Terminale

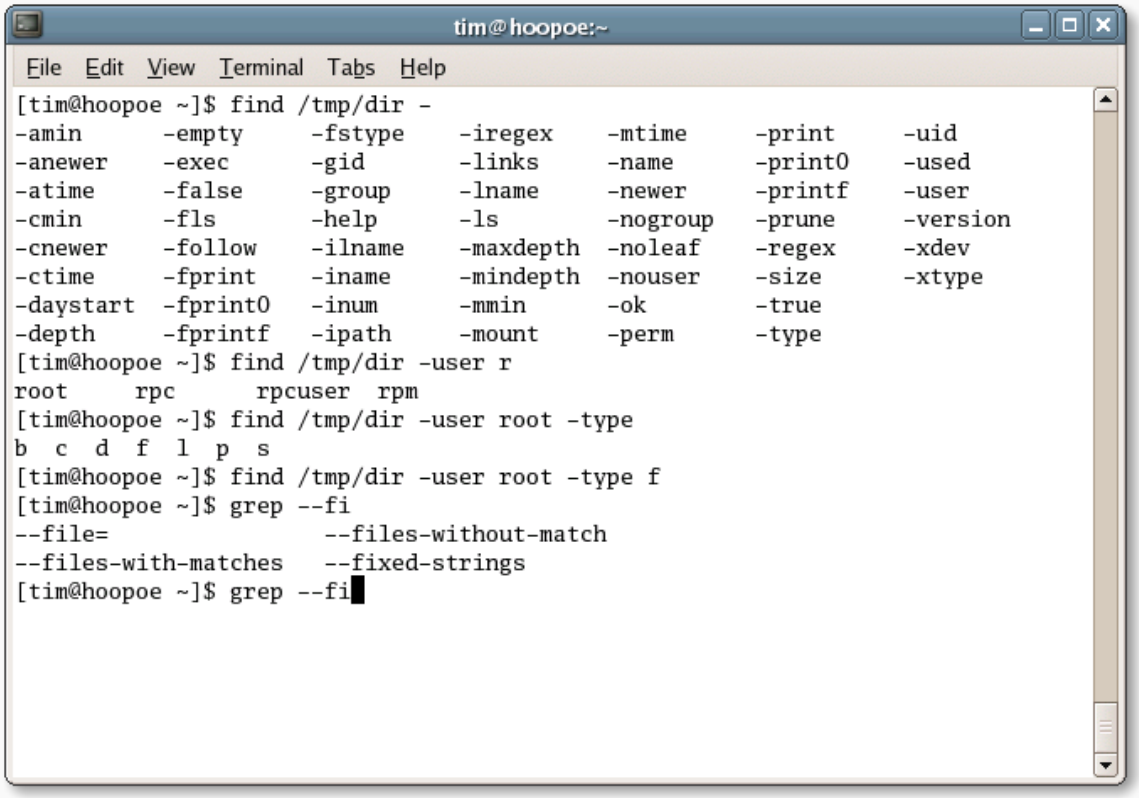
Editor

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.



Cosa è il terminale?

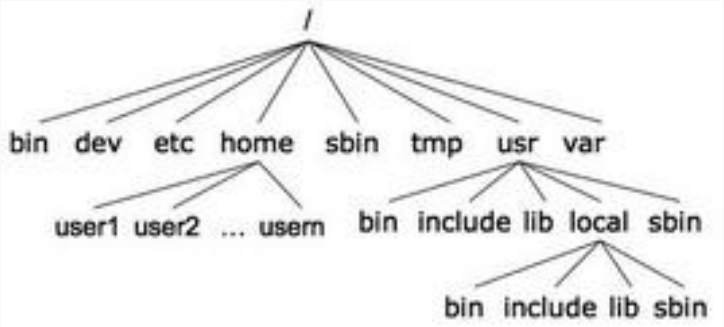


- Programma che fornisce un'interfaccia testuale alle funzionalità del sistema;
- Legge i comandi digitati dall'utente e li esegue (ad es. navigare sul file system, creare file e directory, eseguire programmi).

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*:

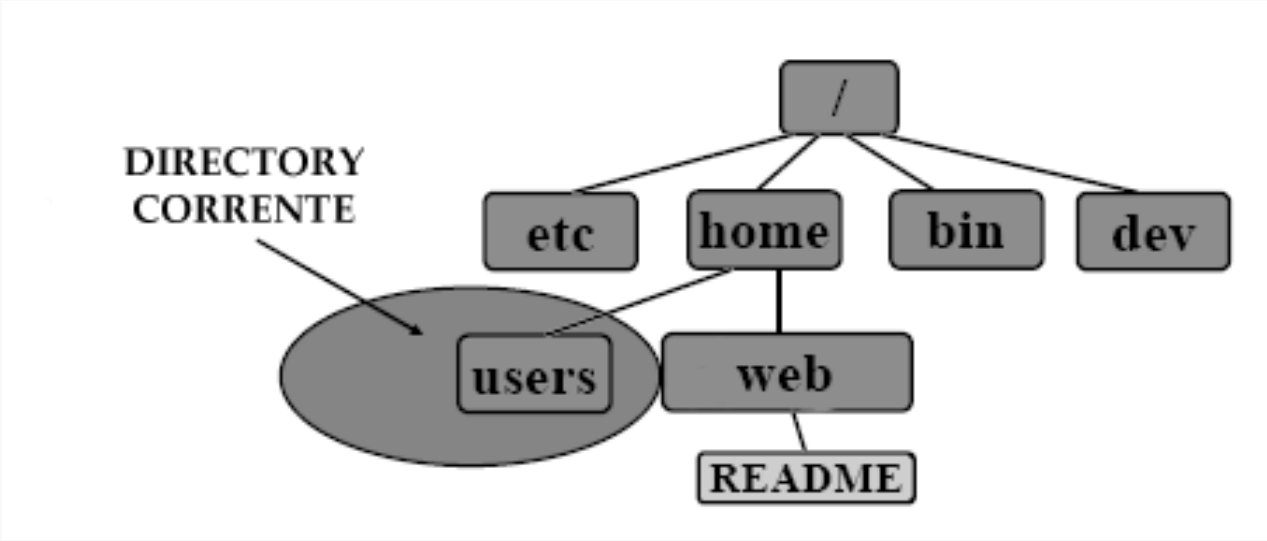


Ogni nodo dell'albero è o un file o una directory di file.

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

I Processi

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

Pre-processing

- 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-processore 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`

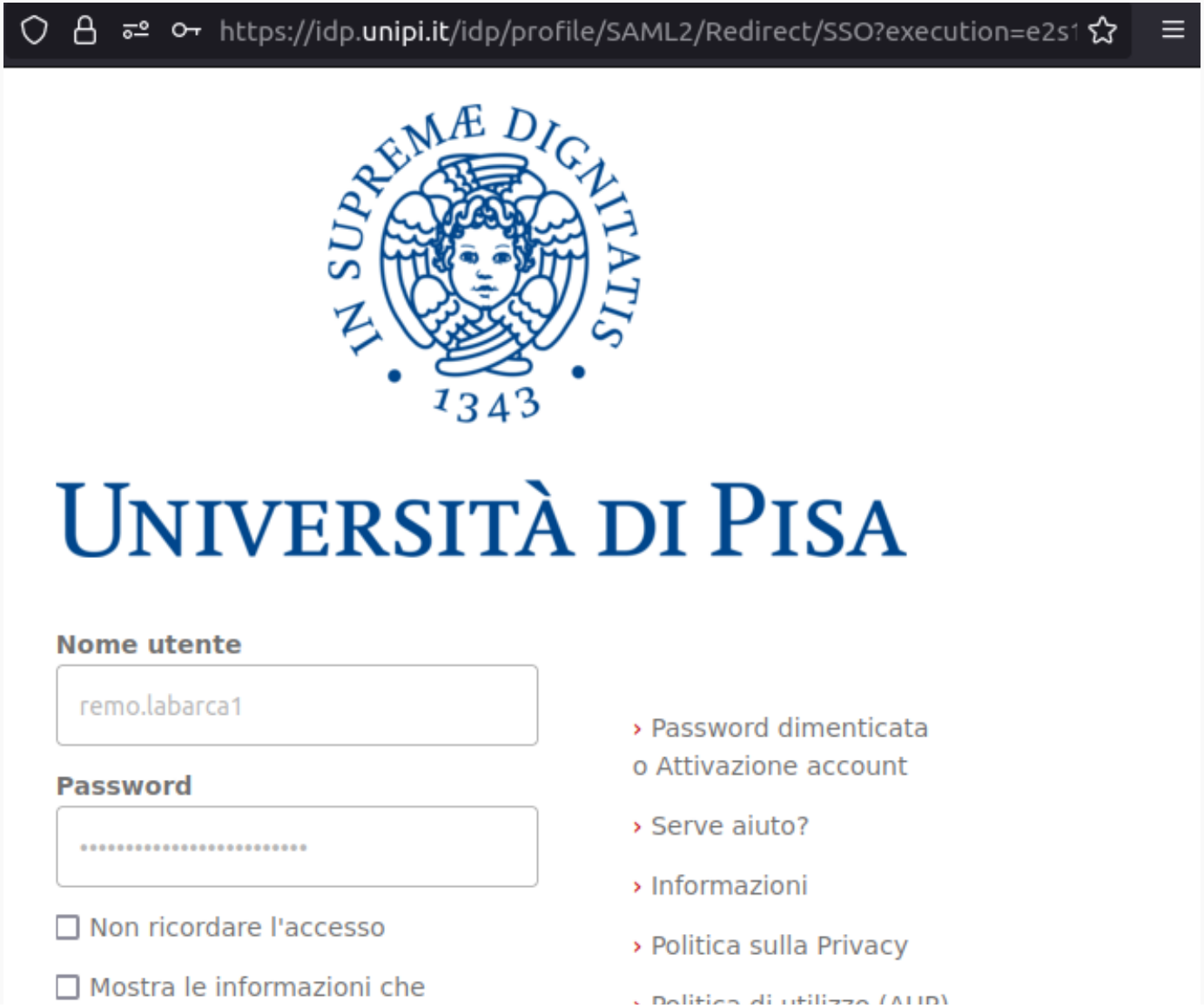
Opzioni importanti di GCC

- `-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

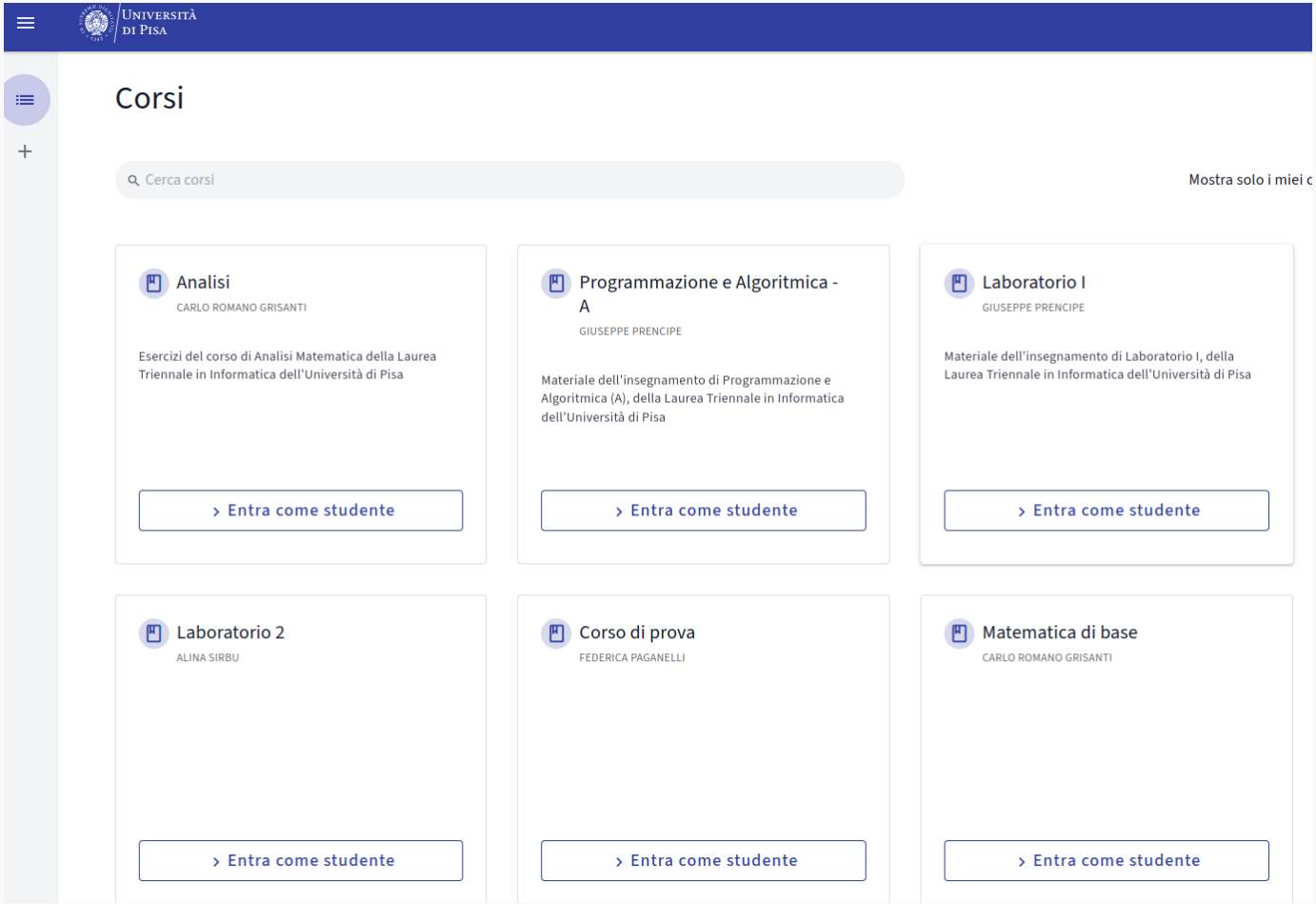
Piattaforma EVO

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



Piattaforma EVO

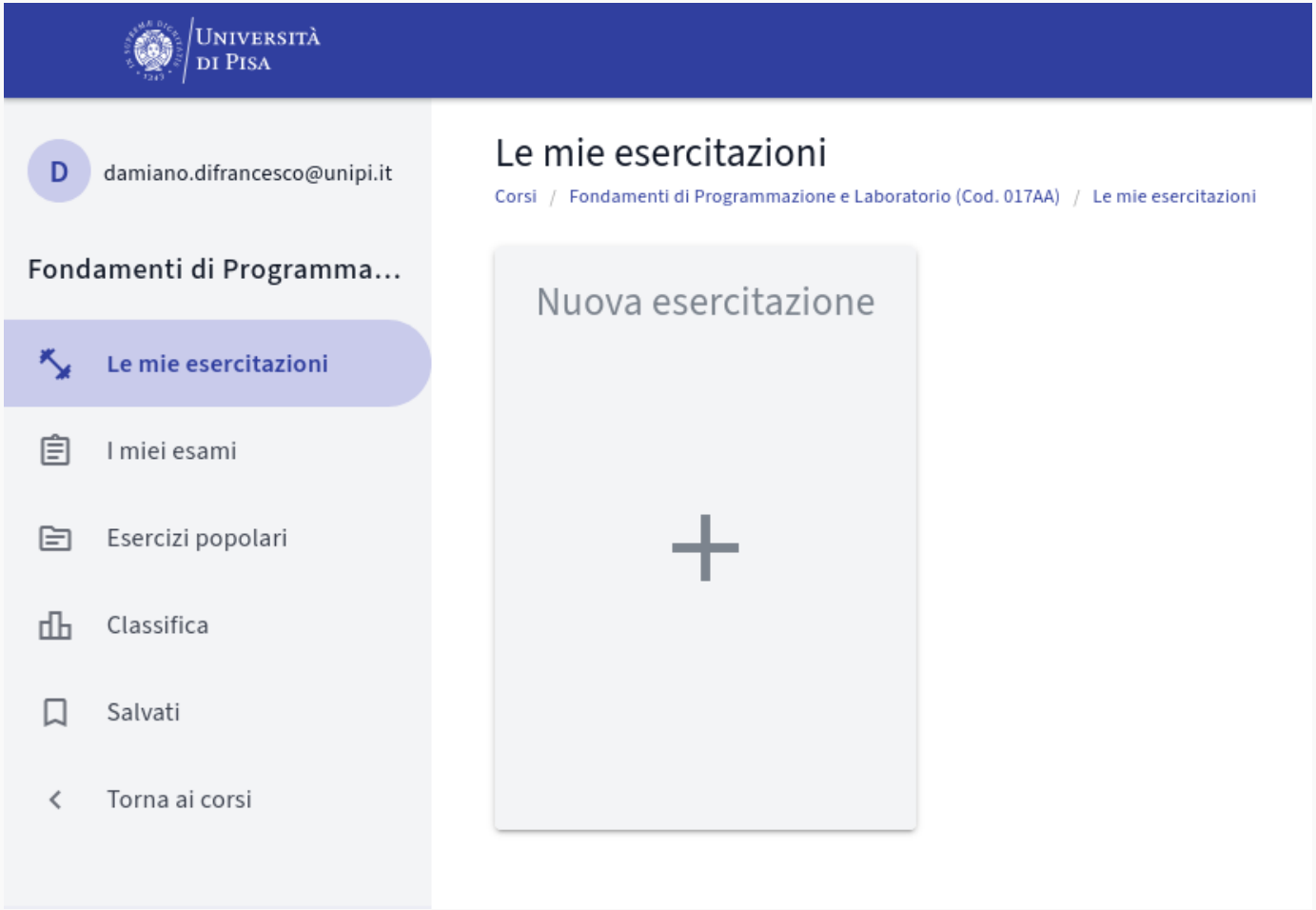
Dopo aver effettuato il login con le credenziali di ateneo dovreste poter vedere tutti i corsi a voi aperti.



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

Piattaforma EVO

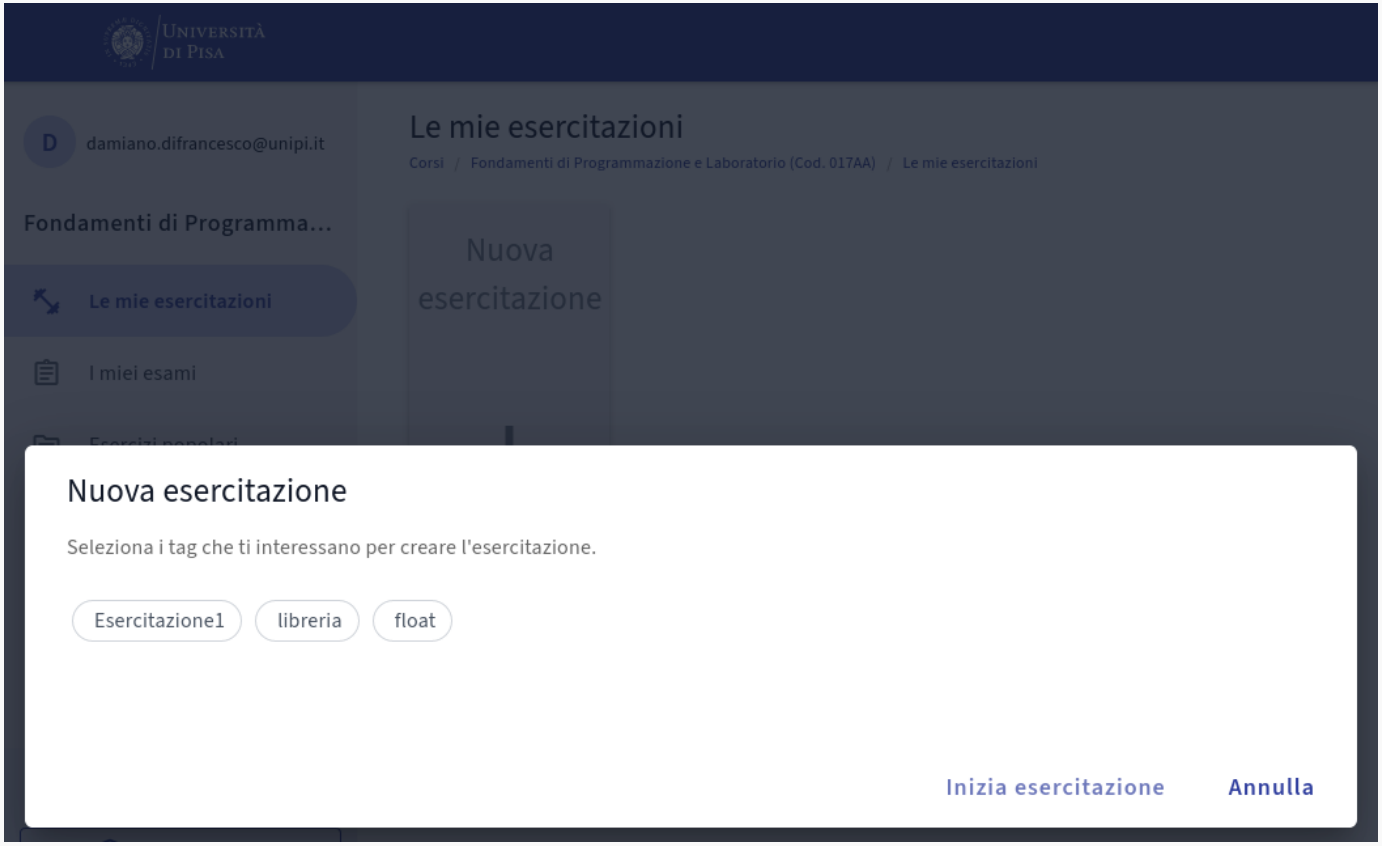
All'interno del corso potete trovare gli esercizi per ogni lezione.



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

Piattaforma EVO

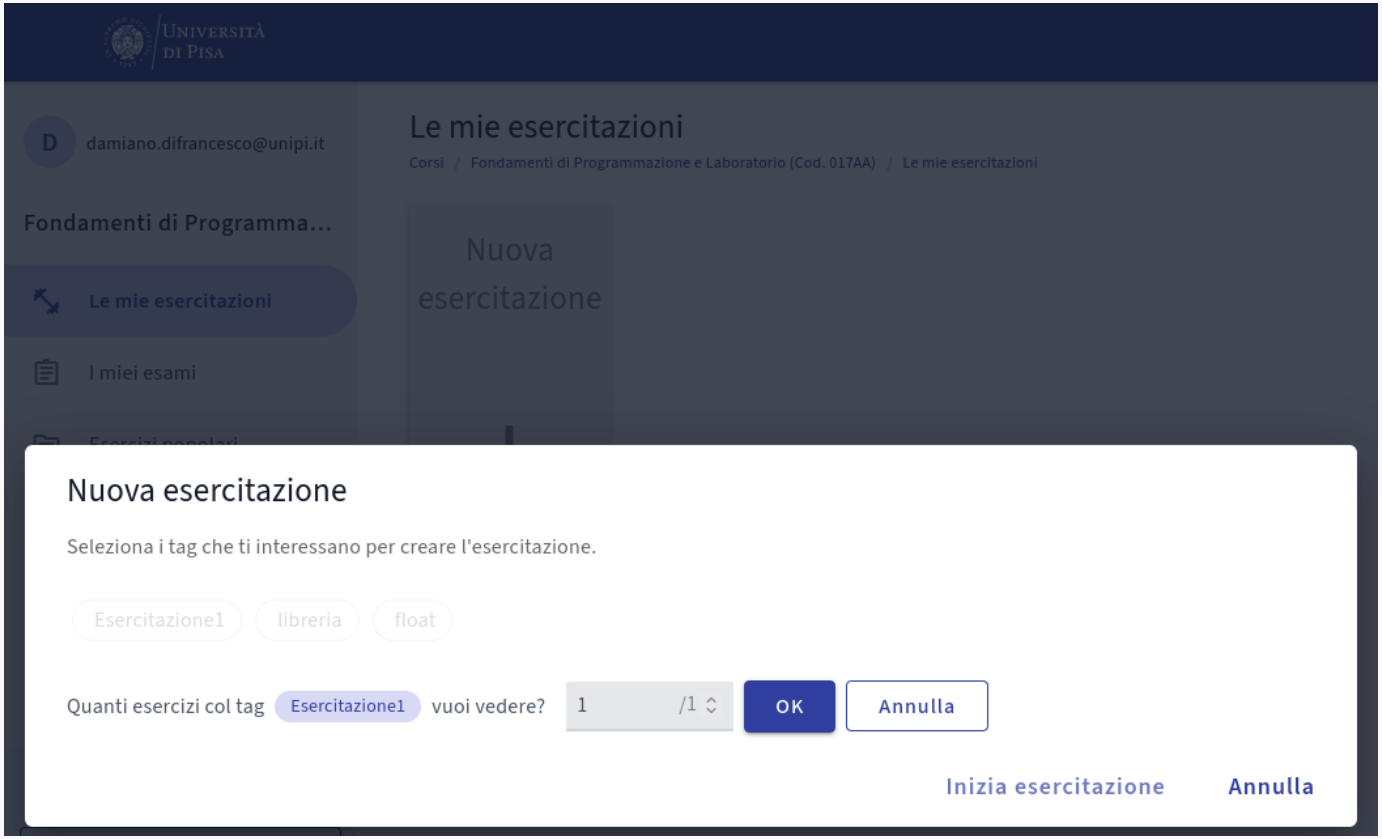
All'interno del corso potete trovare gli esercizi per ogni lezione.



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

Piattaforma EVO

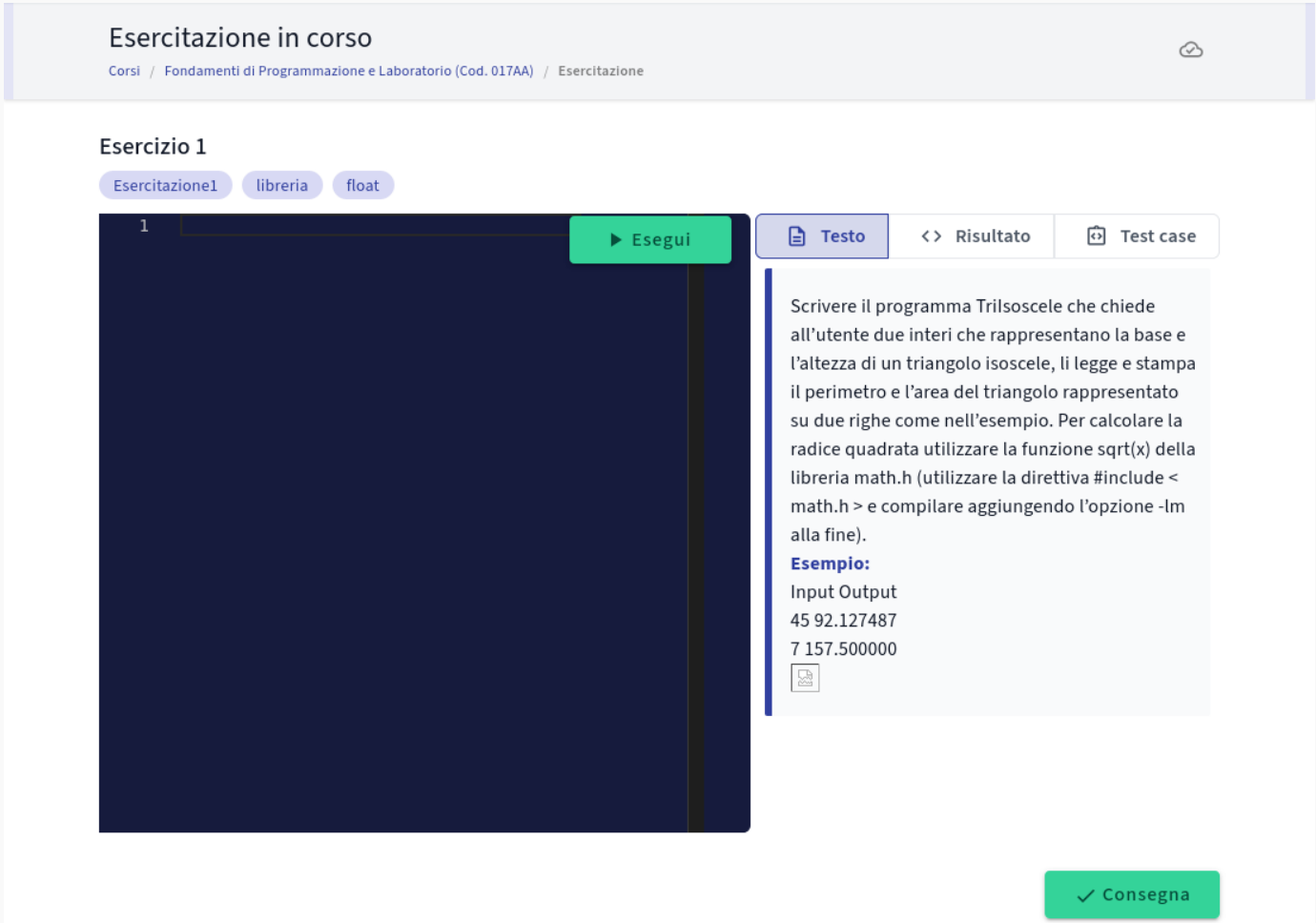
All'interno del corso potete trovare gli esercizi per ogni lezione.



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

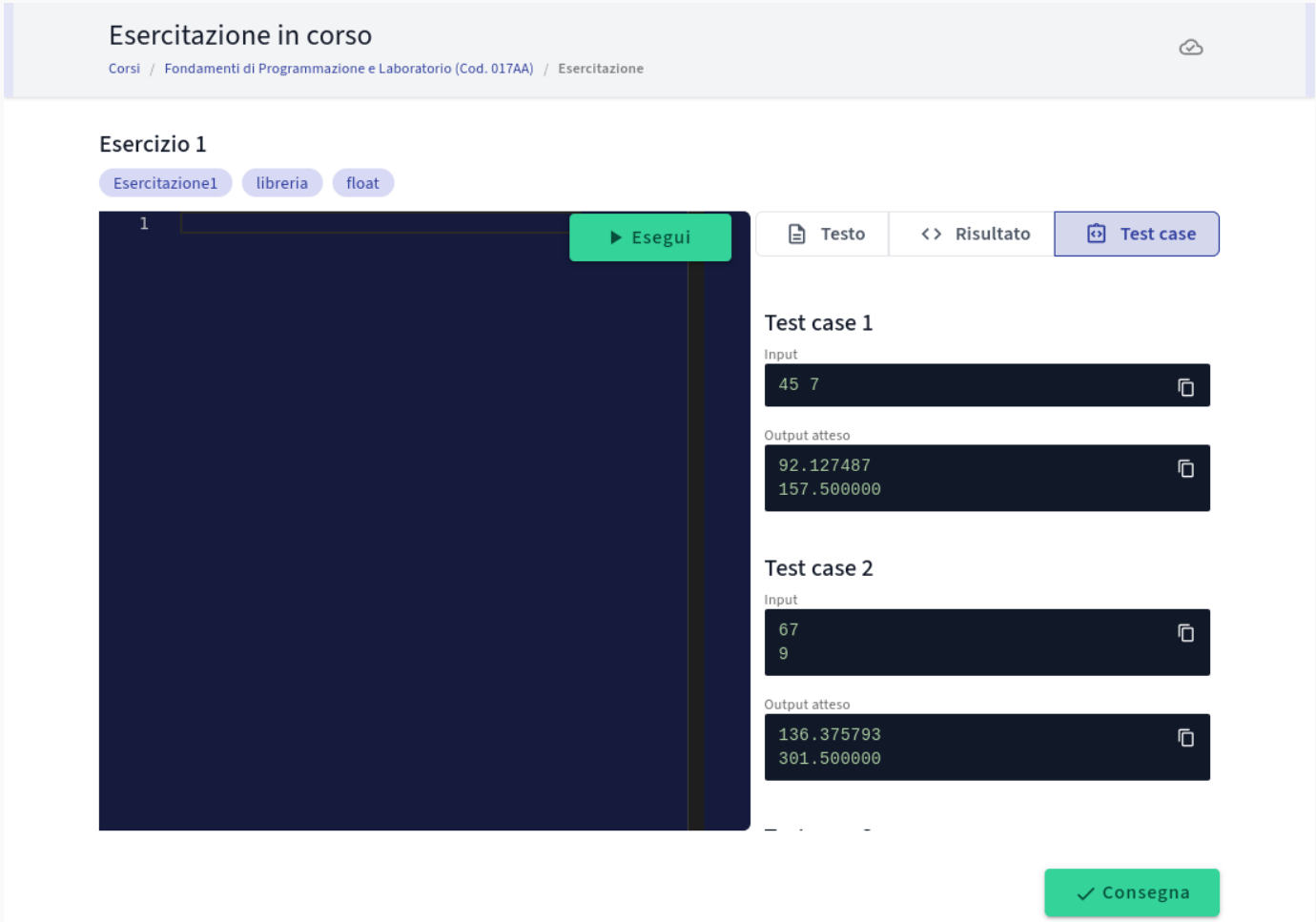
Piattaforma EVO

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.



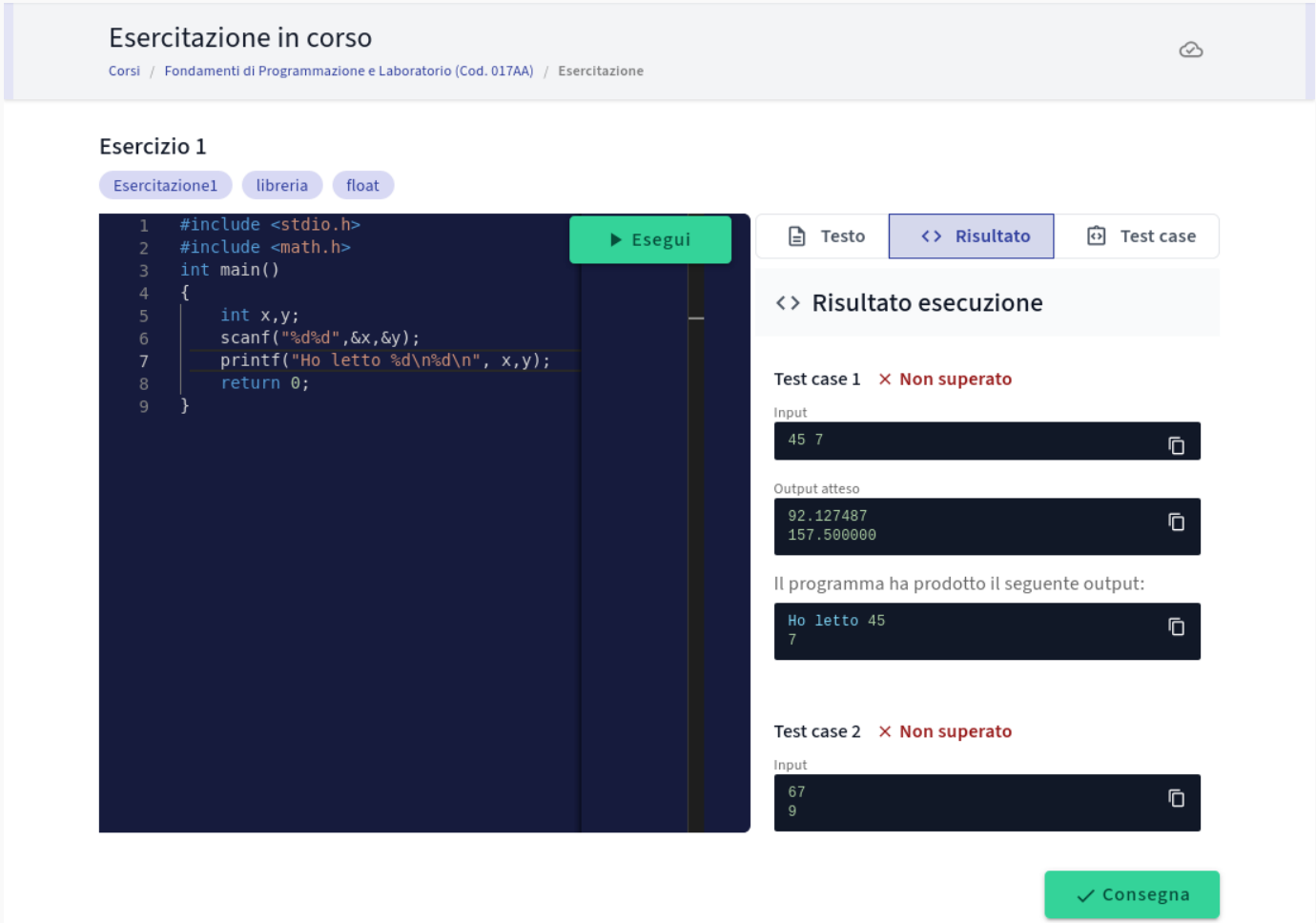
Piattaforma EVO

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.



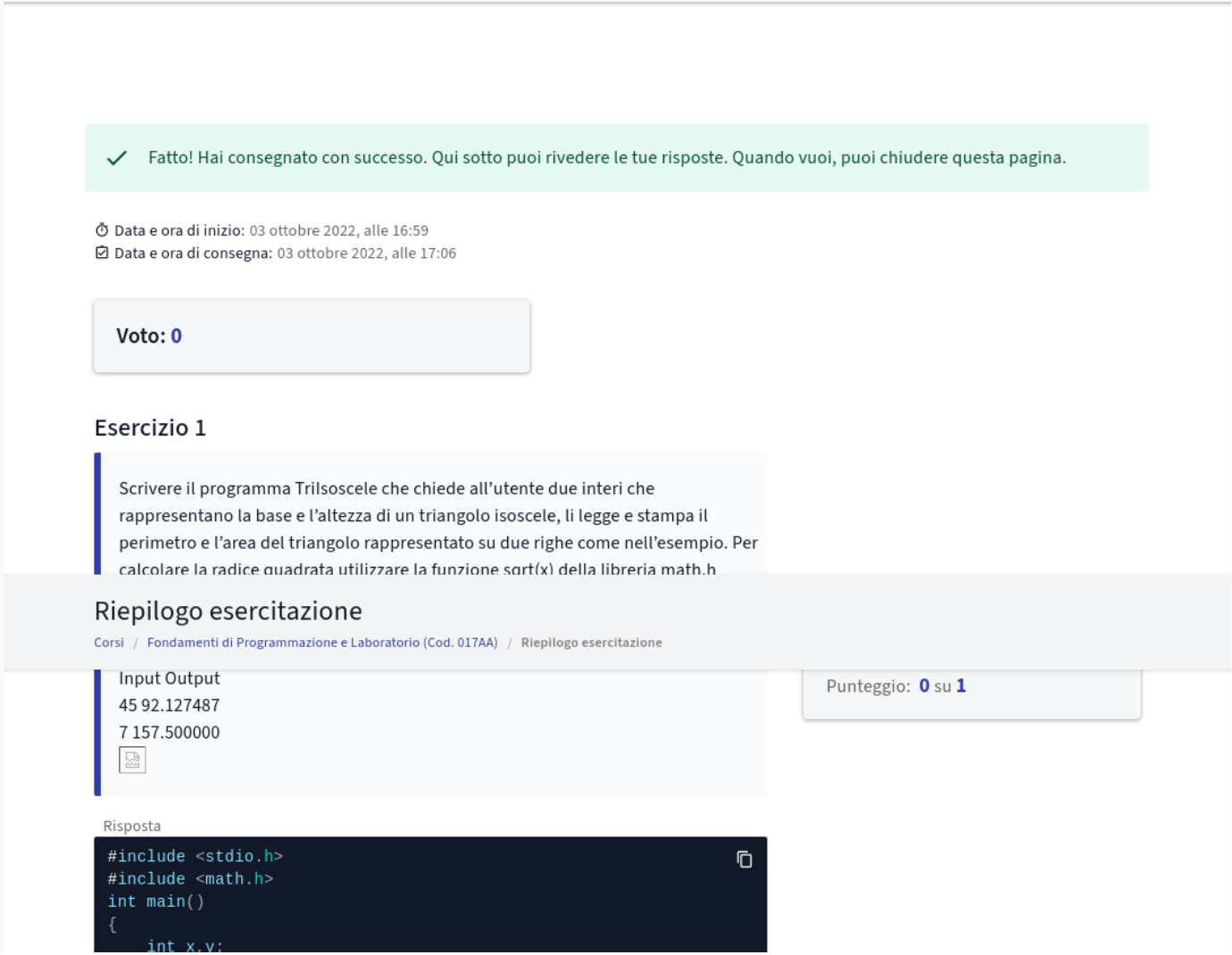
Piattaforma EVO

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.



Piattaforma EVO

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



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

```c
#ifdef _WIN32
#include <Windows.h>
#else
#include <unistd.h>
#endif
#include<stdio.h>

int main(){
    while(1<2){
        printf("Stop me if you can...\n");
        sleep(1);
    }
    return 0;
}
```

main.c 1

Se il programma precedente non compila provate il seguente

```c
#include<stdio.h>

int main(){
    printf("Stop me if you can!\n");
    while(1<2){}
    return 0;
}
```

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

```c
#include <stdio.h>  /* Libreria standard per IO */

/* Il main definisce il punto di partenza del nostro programma.
 * void -> non prende parametri in ingresso (in questo caso)
 * int -> restituisce un intero
 */
int main(void) {

    /* Stampa a schermo la stringa passata come argomento */
    printf("Hello World!\n"); /* \n carattere di escape speciale */

    /* Valore restituito dal main al S.O. (0 -> OK) */
    return 0;
}
```