

PROGETTAZIONE CASI DEI TEST

Sommario

PROGETTAZIONE CASI DEI TEST	1
Test Suite	2
ADT test	2
Come avviene il test? Qual è il flusso di esecuzione? Chi crea i file?	3
mainTest	3
runTest	3
Test Case per le Prenotazioni.....	4
TC1_Prenotazioni.....	5
TC2_Prenotazioni.....	5
TC3_Prenotazioni.....	6
TC4_Prenotazioni.....	6
Test Case per i Clienti	7
TC1_Cliente	7
TC2_Cliente	7
TC3_Cliente	8
Test Case per il Report.....	8
TC1_Report	8

Test Suite

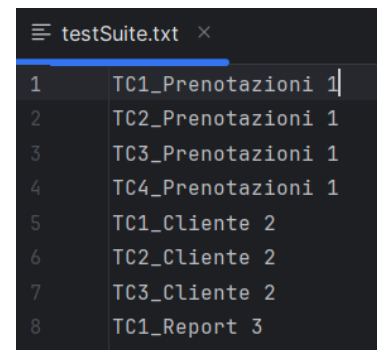
Per la gestione della palestra sono stati sviluppati casi di test per verificare il corretto funzionamento delle principali funzionalità del programma, in particolare siamo andati a testare:

- La corretta registrazione delle prenotazioni e dell'aggiornamento delle disponibilità.
- La gestione degli abbonamenti e della verifica della validità.
- Che il report generato contenga informazioni corrette e complete sulle prenotazioni.

Per i casi di test abbiamo optato per una Test Suite composta da otto casi di test (immagine a destra):

- 4 Test Case per le Prenotazioni.
- 3 Test Case per i Clienti.
- 1 Test Case per il report.

I numeri che seguono i nomi dei test verranno poi spiegati in seguito.



ADT test

Per lo sviluppo delle funzioni che testeranno il programma, abbiamo ideato un ADT a parte, l'ADT test; questo presenta essenzialmente sei funzioni:

- `mainTest (hashtable hClienti, list ICorsi, listP IPrenotazioni)`
- `runTest (string nomeTest, int sel, hashtable hClienti, list ICorsi, listP IPrenotazioni)`
- `testValiditàAbbonamento (string input, string output, string oracle, hashtable hClienti)`
- `testCorrettaRegistrazionePrenotazione (string input, string output, string oracle, hashtable hClienti, list ICorsi)`
- `testReport (string input, string output, string oracle, hashtable hClienti)`
- `confrontaFile (string file1, string file2)`

La progettazione e le specifiche delle seguenti funzioni sono presenti nel seguente percorso:

Progetto → Documentazione

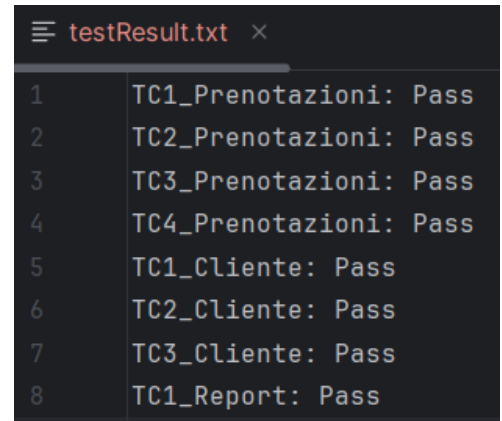
Queste funzioni leggono dalla test suite, i nomi dei test da effettuare, per ognuno di essi vengono creati tre file:

- `"TC1_..._Input.txt"`
- `"TC1_..._Oracle.txt"`
- `"TC1_..._Output.txt"`

I primi due vengono compilati dal programmatore, il terzo invece viene scritto dalla funzione di calcolo di appartenenza, in base al tipo di test da eseguire.

L'obiettivo del testing è assicurarsi che l'output proveniente dalla macchina, sia lo stesso previsto dal programmatore.

Il risultato del test viene salvato poi in un altro file di testo (immagine a destra), contenente tutti i risultati dei test. La struttura del file prevede il nome del test, seguito dall'esito, che può essere Pass o Fail, a seconda se il test sia fallito o meno.



```

1 TC1_Prenotazioni: Pass
2 TC2_Prenotazioni: Pass
3 TC3_Prenotazioni: Pass
4 TC4_Prenotazioni: Pass
5 TC1_Cliente: Pass
6 TC2_Cliente: Pass
7 TC3_Cliente: Pass
8 TC1_Report: Pass
  
```

Come avviene il test? Qual è il flusso di esecuzione? Chi crea i file?

Come abbiamo detto prima, le funzioni per il testing sono presenti nell'ADT Test, ma il "cuore pulsante" della fase di test sono principalmente due funzioni, il `mainTest` e il `runTest`.

mainTest

Questa funzione si occupa della lettura e della scrittura dei risultati dei test, nel programma sono presenti due file di testo:

- *testSuite.txt* (Immagine Pag. 2)
- *testResult.txt* (Immagine Pag. 3)

quando viene richiamata la funzione *mainTest*, dal main principale (i test vengono eseguiti prima del programma stesso, ad ogni avvio), questa apre i due file precedentemente nominati, il primo in lettura e il secondo in scrittura.

Per analizzare uno ad uno gli elementi del *testSuite*, abbiamo implementato un ciclo che termina alla fine del file, ad ogni riga letta, tramite la *fscanf*, salviamo il nome e il numero presenti nel file all'interno di due variabili.

All'interno del corpo del ciclo troviamo essenzialmente due istruzioni, la prima è la chiamata alla funzione *runTest*, il cui ritorno viene assegnato ad una variabile intera *result*; la seconda istruzione consiste nel controllare il valore di *result*, se 1, scriviamo nel file *testResult*, il nome del test seguito da *Pass*, se 0, scriviamo nel medesimo file, il nome seguito da *Fail*.

runTest

Questa funzione si occupa della vera e propria esecuzione del testing; infatti, è proprio questa funzione che tramite uno switch e il numero dopo il nome del test, che richiama la funzione specifica per quel test.

Innanzitutto, la funzione genera tre stringhe, ognuna di essi rappresenta il nome del file da cui la funziona dovrà leggere, scrivere o confrontare. Dopodiché passa alla selezione del caso di test:

1. Test Case per le Prenotazioni

Richiama la funzione *testCorrettaRegistrazionePrenotazione* e ritorna il valore di ritorno di essa.

2. Test Case per i Clienti

Richiama la funzione *testValiditàAbbonamento* e ritorna il valore di ritorno di essa.

3. Test Case per il Report

Richiama la funzione *testReport* e ritorna il valore di ritorno di essa.

NB: tutti i dati su cui lavorano i test, non alterano il normale utilizzo del programma, questi sono isolati e se utilizzano i dati già caricati dal programma, questi non vengono modificati o alterati. Ad esempio, per i test delle prenotazioni viene passata l'*hashtable* dei clienti, ma il suo utilizzo è limitato al solo controllo che l'ID presente nel file sia presente nella *hashtable*.

Per facilitare il test visivo all'utente, riporto qui i due file contenenti i dati degli iscritti e dei corsi.

iscritti.txt							
1	CLT003	Emanuele	Palumbo	3	6	2025	5
2	CLT002	Giovanni	Scarico	15	4	2025	1
3	CLT005	Salvatore	Esposito	18	5	2025	9
4	CLT004	Gennaro	Esposito	15	5	2025	1
5	CLT001	Giuseppe	Scarpa	15	4	2025	5

corsi.txt							
1	CRS010	KickBox	13	6	2025	11:00	2
2	CRS009	Yoga	30	6	2025	18:00	3
3	CRS008	Pilates	29	8	2025	10:00	0
4	CRS007	Pilates	28	6	2025	16:00	0
5	CRS006	Spinning	12	6	2025	15:30	0
6	CRS005	KickBox	12	6	2025	16:30	0
7	CRS004	Yoga	18	7	2025	15:00	1
8	CRS003	Zumba	18	6	2025	16:00	0
9	CRS002	Pilates	15	7	2025	16:00	1
10	CRS001	Yoga	18	9	2025	12:00	0
11	CRS011	KickBox	12	6	2025	11:00	0

Test Case per le Prenotazioni

Questi test case hanno l'obiettivo di testare la corretta registrazione delle prenotazioni e dell'aggiornamento delle disponibilità.

Abbiamo sperimentato quattro tipi di test per questa richiesta:

- **TC1:** Sia il Cliente che il Corso esistono.
- **TC2:** Esiste il Cliente, ma non il Corso.
- **TC3:** Esiste il Corso, ma non il Cliente.
- **TC4:** Non esiste né il Cliente né il Corso.

Riportiamo la struttura della prenotazione (immagine a destra).

```
struct prenotazione{
    string IDPrenotazione;
    string IDCorso;
    string IDCliente;
    Data dataPrenotazione;
};
```

TC1_Prenotazioni

Caso: Sia il Cliente che il Corso esistono

```
TC1_Prenotazioni_Input.txt x
1 CRS001 CLT001
```

- *CLT001*: Giuseppe Scarpa, abbonamento valido fino al 15/09/25.
- *CRS001*: Yoga, lezione successiva ad oggi (18/09/2025), posti disponibili: 20.

I dati sono validi, l'abbonamento non è scaduto e il corso non è antecedente alla data di oggi; quindi, il risultato atteso sarà una prenotazione di *CLT001* per *CRS001* e un incremento di partecipanti a *CRS001* (+1).

```
TC1_Prenotazioni_Oracle.txt x
1 CRS001 CLT001 1
```

Per l'oracolo abbiamo scelto di stampare soltanto i tre dati necessari alla verifica del test, quindi *l'IDCorso*, *l'IDCliente*, e il nuovo numero di partecipanti al Corso.

```
TC1_Prenotazioni_Output.txt x
1 CRS001 CLT001 1
```

Da come puoi osservare l'output generato dalla funzione è lo stesso previsto dallo sviluppatore. Test Passato.

TC2_Prenotazioni

Caso: Esiste il Cliente, ma non il Corso

```
TC2_Prenotazioni_Input.txt x
1 CRS012 CLI001
```

- *CLT001*: Giuseppe Scarpa, abbonamento valido fino al 15/09/25.
- *CRS012*: Lezione inesistente.

I dati non sono validi, il cliente esiste, ma non il corso; quindi, il risultato atteso sarà un messaggio che dice, "*Cliente o Corso inesistente*" (messaggio impostato dal programmatore).

```
TC2_Prenotazioni_Oracle.txt x
1 Cliente o Corso inesistente
```

```
TC2_Prenotazioni_Output.txt x
1 Cliente o Corso inesistente
```

Da come puoi osservare l'output generato dalla funzione è lo stesso previsto dallo sviluppatore. Test Passato.

TC3_Prenotazioni

Caso: Esiste il Corso, ma non il Cliente.

```
TC3_Prenotazioni_Input.txt x
1 CRS003 CLT012
```

- *CLT012*: Cliente inesistente
- *CRS003*: Zumba, lezione successiva ad oggi (18/06/2025), posti disponibili: 20

I dati non sono validi, il Corso esiste, ma non il Cliente; quindi, il risultato atteso sarà un messaggio che dice, “*Cliente o Corso inesistente*” (messaggio impostato dal programmatore).

```
TC3_Prenotazioni_Oracle.txt x
1 Cliente o Corso inesistente
```

```
TC3_Prenotazioni_Output.txt x
1 Cliente o Corso inesistente
```

Da come puoi osservare l’output generato dalla funzione è lo stesso previsto dallo sviluppatore. Test Passato.

TC4_Prenotazioni

Caso: Non esiste né il Cliente né il Corso.

```
TC4_Prenotazioni_Input.txt x
1 CRS012 CLT012
```

- *CLT012*: Cliente inesistente
- *CRS012*: Corso inesistente

I dati non sono validi, non esiste né il Corso né il Cliente; quindi, il risultato atteso sarà un messaggio che dice, “*Cliente o Corso inesistente*” (messaggio impostato dal programmatore).

```
TC4_Prenotazioni_Oracle.txt x
1 Cliente o Corso inesistente
```

```
TC4_Prenotazioni_Output.txt x
1 Cliente o Corso inesistente
```

Da come puoi osservare l’output generato dalla funzione è lo stesso previsto dallo sviluppatore. Test Passato.

Test Case per i Clienti

Questi test case hanno l'obiettivo di testare la corretta gestione e validità degli abbonamenti.

Abbiamo sperimentato tre tipi di test per questa richiesta:

- **TC1:** Abbonamento Valido
- **TC2:** Abbonamento scaduto
- **TC3:** Cliente inesistente

TC1_Cliente

Caso: Abbonamento valido

```
TC1_Cliente_Input.txt x
1 CLT001
```

- *CLT001*: Giuseppe Scarpa, abbonamento valido fino al 15/09/25.

Il cliente esiste e il suo abbonamento non è scaduto; quindi, il risultato atteso sarà *"Abbonamento Valido"*.

```
TC1_Cliente_Oracle.txt x
1 Abbonamento Valido

TC1_Cliente_Output.txt x
1 Abbonamento Valido
```

Da come puoi osservare l'output generato dalla funzione è lo stesso previsto dallo sviluppatore. Test Passato.

TC2_Cliente

Caso: Abbonamento scaduto

```
TC2_Cliente_Input.txt x
1 CLT002
```

- *CLT002*: Giovanni Scarico, abbonamento scaduto il 15/05/25.

Il cliente esiste ma il suo abbonamento è scaduto; quindi, il risultato atteso sarà *"Abbonamento Scaduto"*.

```
TC2_Cliente_Oracle.txt x
1 Abbonamento Scaduto

TC2_Cliente_Output.txt x
1 Abbonamento Scaduto
```

Da come puoi osservare l'output generato dalla funzione è lo stesso previsto dallo sviluppatore. Test Passato.

TC3_Cliente

Caso: Cliente inesistente

```
TC3_Cliente_Input.txt x
1 CLT100
```

- *CLT100*: Cliente inesistente.

Il cliente non esiste, quindi il risultato atteso sarà "*Cliente inesistente*".

```
TC3_Cliente_Oracle.txt x
1 Cliente inesistente
```

```
TC3_Cliente_Output.txt x
1 Cliente inesistente
```

Da come puoi osservare l'output generato dalla funzione è lo stesso previsto dallo sviluppatore. Test Passato.

Test Case per il Report

TC1_Report

L'obiettivo del test è verificare che il report generato contenga informazioni corrette e complete sulle prenotazioni.

Per il testing del report abbiamo dovuto inserire più una prenotazione in quanto il nostro report riporta:

- Numero di prenotazioni del mese corrente.
- I tre corsi più frequentati del mese corrente.

```
TC1_Report_Input.txt x
1 CRS010 CLT005
2 CRS010 CLT001
3 CRS007 CLT001
4 CRS007 CLT003
5 CRS007 CLT004
6 CRS006 CLT004
7 CRS004 CLT001
8 CRS001 CLT001
9 CRS007 CLT004
10 CRS010 CLT003
11 CRS001 CLT005
```

```
prenotazioni.txt x
1 PRT008 CRS009 CLT001 6 6 2025
2 PRT007 CRS009 CLT005 6 6 2025
3 PRT006 CRS010 CLT005 6 6 2025
4 PRT005 CRS009 CLT003 6 6 2025
5 PRT004 CRS010 CLT001 6 6 2025
6 PRT003 CRS004 CLT003 6 6 2025
7 PRT002 CRS002 CLT003 6 6 2025
```


Ai dati forniti da *TC1_Report_Input* aggiungiamo anche le prenotazioni già esistenti nel programma, in totale troviamo un numero di prenotazione pari a 18; inoltre sono presenti anche prenotazioni a lezioni che non si terranno il mese corrente, per testare il completo funzionamento dell'algoritmo. Per individuare invece le prime tre lezioni del mese, dobbiamo controllare i corsi che hanno lezione nel mese corrente (giugno).

1. *CRS010*, KickBox, data lezione: 13/6/2025, numero partecipanti: 5
2. *CRS007*, Pilates, data lezione: 28/6/2025, numero partecipanti: 4
3. *CRS009*, Yoga, data lezione: 30/6/2025, numero partecipanti: 3

Il risultato atteso sarà allora:

TC1_Report_Oracle.txt		TC1_Report_Output.txt	
1	Report di Giugno	1	Report di Giugno
2		2	
3	Numero Prenotazioni del Mese: 18	3	Numero Prenotazioni del Mese: 18
4		4	
5	Corsi del Mese:	5	Corsi del Mese:
6	CRS010 KickBox 13/6/2025 5	6	CRS010 KickBox 13/6/2025 5
7	CRS007 Pilates 28/6/2025 4	7	CRS007 Pilates 28/6/2025 4
8	CRS009 Yoga 30/6/2025 3	8	CRS009 Yoga 30/6/2025 3

Confrontando l'oracolo con l'output della funzione, i due file risultano uguali. Test Passato.