Specifiche Sintattiche e Semantiche Progetto PSD

**Progetto: Gestione Prenotazioni Di Una Palestra “Traccia 2”  
Autore: Giovanni Pierno, Tedesco Simone  
Data: Aprile 2025**

# Indice

*1. Specifiche Sintattiche dei File Sorgente*

*2. Specifiche Semantiche dei File Sorgente*

## File: util.c

### 1. Specifiche Sintattiche

* **Include di: stdio.h, stdlib.h, util.h, data.h**
* **Definizioni di macro per colori ANSI**
* **Funzioni: pulisciSchermo(), messaggio\_errore(), schermata\_arrivederci()**

### 2. Specifiche Semantiche

## PulisciSchermo()

* **Sintassi: void pulisciSchermo()**
* **Semantica: Pulisce il terminale e stampa la data odierna usando oggi() e stampaData().**
* **Precondizioni: Nessuna**
* **Postcondizioni: La schermata viene ripulita e aggiornata con la data attuale.**

## messaggio\_errore()

* **Sintassi: void messaggio\_errore()**
* **Semantica: Stampa un messaggio d’errore rosso in caso di input non valido.**
* **Precondizioni: Nessuna**
* **Postcondizioni: Il messaggio viene visualizzato nel terminale.**

## Schermata\_arrivederci()

* **Sintassi: void schermata\_arrivederci()**
* **Semantica: Mostra un messaggio finale di ringraziamento con cornice decorativa.**
* **Precondizioni: Nessuna**
* **Postcondizioni: Il messaggio di chiusura viene stampato.**

**Esempio di codice pulisciSchermo():**

Immagine che contiene testo, Carattere, schermata

Il contenuto generato dall'IA potrebbe non essere corretto.

## File: main.c

### 1. Specifiche Sintattiche

* **Include di: stdio.h, stdlib.h, menu.h, util.h**
* **Funzione main() con ciclo while(1) per la gestione continua del menu.**
* **Uso di scanf() e controllo input con pulizia del buffer.**
* **Switch su scelta utente con gestione di 3 casi (0=uscita, 1=gestore, 2=cliente).**

### 2. Specifiche Semantiche

## Main()

* **Sintassi: int main()**
* **Semantica: Gestisce l'avvio del programma. Mostra il menu iniziale, riceve input utente e smista il controllo ai menu cliente o gestore.**
* **Precondizioni: Nessuna**
* **Postcondizioni: Il programma rimane in esecuzione fino alla scelta di uscita da parte dell'utente.**

## File: data.c

### 1. Specifiche Sintattiche

* **Include di: data.h, stdio.h, time.h**
* **Funzioni statiche e pubbliche: giorniNelMese(), dataValida(), leggiData(), stampaData(), confrontaDate(), aggiungiMesi(), oggi(), eScaduto()**

### 2. Specifiche Semantiche

## GiorniNelMese()

* **Sintassi: int giorniNelMese(int mese, int anno)**
* **Semantica: Restituisce il numero di giorni in un mese considerando anni bisestili.**
* **Precondizioni: Mese compreso tra 1 e 12; anno >= 0**
* **Postcondizioni: Restituisce un intero rappresentante il numero di giorni.**

## DataValida()

* **Sintassi: int dataValida(Data d)**
* **Semantica: Verifica che una data sia valida.**
* **Precondizioni: Data con valori coerenti (es. giorno 1-31)**
* **Postcondizioni: Restituisce 1 se valida, altrimenti 0.**

## LeggiData()

* **Sintassi: Data leggiData()**
* **Semantica: Chiede in input una data finché non è valida.**
* **Precondizioni: Input utente interattivo**
* **Postcondizioni: Restituisce una data valida.**

## StampaData()

* **Sintassi: void stampaData(Data d)**
* **Semantica: Stampa una data nel formato gg/mm/aaaa.**
* **Precondizioni: Data valida come parametro**
* **Postcondizioni: La data viene stampata a schermo.**

## ConfrontaDate()

* **Sintassi: int confrontaDate(Data d1, Data d2)**
* **Semantica: Confronta due date restituendo -1, 0 o 1.**
* **Precondizioni: Due date valide**
* **Postcondizioni: Restituisce -1 se d1 < d2, 0 se uguali, 1 se d1 > d2.**

## AggiungiMesi()

* **Sintassi: Data aggiungiMesi(Data d, int mesi)**
* **Semantica: Aggiunge mesi a una data gestendo overflow.**
* **Precondizioni: Data valida, mesi >= 0**
* **Postcondizioni: Restituisce la nuova data aggiornata.**

## Oggi()

* **Sintassi: Data oggi()**
* **Semantica: Restituisce la data odierna usando la libreria time.**
* **Precondizioni: Nessuna**
* **Postcondizioni: Ritorna la data corrente del sistema.**

## Escaduto()

* **Sintassi: int eScaduto(Data d)**
* **Semantica: Verifica se una data è scaduta rispetto a oggi.**
* **Precondizioni: Data valida**
* **Postcondizioni: Restituisce 1 se scaduta, 0 altrimenti.**

## File: menu.c

### 1. Specifiche Sintattiche

* **Include di: stdio.h, stdlib.h, abbonamento.h, data.h, util.h, menu.h,**
* **Definizioni di macro per colori ANSI**
* **Funzione: gestione\_abbonamenti() con ciclo switch basato su input utente.**
* **Funzioni: menu\_iniziale(), menu\_gestore(), menu\_cliente()**

### 2. Specifiche Semantiche

## Menu\_Iniziale()

* **Sintassi: void menu\_iniziale()**
* **Semantica: Stampa il menu principale con opzioni per utente e gestore.**
* **Precondizioni: Nessuna**
* **Postcondizioni: Il menu viene visualizzato a schermo.**

## Menu\_Gestore()

* **Sintassi: void menu\_gestore()**
* **Semantica: Mostra le opzioni disponibili per l’amministratore (gestione abbonamenti, gestione lezioni, ecc.) e ne gestisce l'esecuzione.**
* **Precondizioni: L’utente ha scelto l’opzione "gestore".**
* **Postcondizioni: Attiva funzioni relative alla gestione.**

## Menu\_Cliente()

* **Sintassi: void menu\_cliente()**
* **Semantica: Mostra il menu per il cliente.**
* **Precondizioni: L’utente ha selezionato "cliente".**
* **Postcondizioni: Viene mostrato il menu, ma nessuna operazione effettiva è eseguita.**

## File: abbonamento.c

### Specifiche Sintattiche

### Include di: abbonamento.h, stdio.h, stdlib.h, string.h, data.h, file.h

### Funzione definita: gestione\_abbonamenti().

### 2. Specifiche Semantiche

## Gestione\_Abbonamenti()

* **Sintassi: void gestione\_abbonamenti()**
* **Semantica: Gestisce il menu interattivo per aggiungere, modificare, eliminare o visualizzare abbonamenti. Mostra il menu, riceve input numerico e attiva l'opzione corrispondente.**
* **Precondizioni: Deve esistere una struttura dati compatibile (es. ListaAbbonamenti) e un terminale per l'interazione.**
* **Postcondizioni: n base all'input dell'utente, viene modificata la lista abbonamenti o mostrato un messaggio.**

## File: file.c

### Specifiche Sintattiche • Include di: stdio.h, stdlib.h, string.h, abbonamento.h

### Funzioni definite:

* **caricaAbbonamenti(ListaAbbonamenti \*)**
* **salvaAbbonamenti(ListaAbbonamenti)**

### 2. Specifiche Semantiche

**caricaAbbonamenti()**

* **Sintassi: void caricaAbbonamenti(ListaAbbonamenti \*lista);**
* **Semantica: Carica gli abbonamenti dal file "abbonamenti.txt" e li inserisce nella lista passata come parametro.**
* **Precondizioni: Il file deve esistere e la lista dev'essere inizializzata.**
* **Postcondizioni: La lista lista contiene tutti gli abbonamenti letti dal file.**

**salvaAbbonamenti()**

* **Sintassi**: void salvaAbbonamenti(ListaAbbonamenti lista);
* **Semantica**: Scrive i dati della lista di abbonamenti nel file abbonamenti.txt, sovrascrivendolo.
* **Precondizioni**: La lista lista deve essere valida e popolata.
* **Postcondizioni**: Il file "abbonamenti.txt" contiene tutti gli abbonamenti presenti nella lista.

## File: gestione\_abbonamenti.c

### Specifiche Sintattiche

* **Include di: stdio.h, stdlib.h, string.h, abbonamento.h**

**Funzioni definite:**

* **aggiungiAbbonamento(ListaAbbonamenti \*)**
* **modificaAbbonamento(ListaAbbonamenti \*)**
* **eliminaAbbonamento(ListaAbbonamenti \*)**
* **visualizzaAbbonamenti(ListaAbbonamenti)**

### 2. Specifiche Semantiche

**aggiungiAbbonamento()**

* **Sintassi**: void aggiungiAbbonamento(ListaAbbonamenti \*lista);
* **Semantica**: Aggiunge un nuovo abbonamento alla lista, richiedendo i dati all'utente.
* **Precondizioni**: La lista lista deve essere inizializzata correttamente.
* Postcondizioni: Un nuovo abbonamento è stato allocato e inserito in lista.

**modificaAbbonamento()**

* **Sintassi**: void modificaAbbonamento(ListaAbbonamenti \*lista);
* **Semantica**: Modifica un abbonamento esistente nella lista, identificato da un codice fornito dall'utente.
* **Precondizioni**: La lista lista deve contenere almeno un abbonamento.
* **Postcondizioni**: L'abbonamento selezionato è stato modificato con i nuovi dati forniti.

**eliminaAbbonamento()**

* **Sintassi: void eliminaAbbonamento(ListaAbbonamenti \*lista)**
* **Semantica: Rimuove un abbonamento identificato da codice dalla lista.**
* **Precondizioni: La lista deve contenere almeno un elemento.**
* **Postcondizioni: L’abbonamento è stato rimosso e la memoria deallocata.**

**visualizzaAbbonamenti()**

* **Sintassi: void visualizzaAbbonamenti(ListaAbbonamenti lista)**
* **Semantica: Stampa a schermo tutti gli abbonamenti presenti nella lista.**
* **Precondizioni: La lista deve essere inizializzata.**
* **Postcondizioni: I dati sono stati mostrati a terminale.**

**gestione\_lezioni.c**

1. **Specifiche Sintattiche**

* **Include di: stdio.h, stdlib.h, string.h, lezione.h**

**Funzioni definite:**

* **aggiungiLezione(ListaLezioni \*)**
* **modificaLezione(ListaLezioni \*)**
* **eliminaLezione(ListaLezioni \*)**
* **visualizzaLezioni(ListaLezioni)**

### 2. Specifiche Semantiche

**aggiungiLezione()**

* **Sintassi: void aggiungiLezione(ListaLezioni \*lista);**
* **Semantica: Aggiunge una nuova lezione alla lista, richiedendo i dati all'utente.**
* **Precondizioni: La lista lista deve essere inizializzata correttamente.**
* **Postcondizioni: Una nuova lezione è aggiunta alla lista.**

**modificaLezione()**

* **Sintassi: void modificaLezione(ListaLezioni \*lista);**
* **Semantica: Modifica una lezione esistente nella lista, identificata da un codice fornito dall'utente.**
* **Precondizioni: La lista lista deve contenere almeno una lezione.**
* **Postcondizioni: La lezione selezionata è stata modificata con i nuovi dati forniti.**

**eliminaLezione()**

* **Sintassi: void eliminaLezione(ListaLezioni \*lista)**
* **Semantica: Elimina una lezione dalla lista, identificata da codice.**
* **Precondizioni: La lista deve essere non vuota.**
* **Postcondizioni: La lezione è stata rimossa.**

**visualizzaLezioni()**

* **Sintassi: void visualizzaLezioni(ListaLezioni lista)**
* **Semantica: Mostra tutte le lezioni presenti nella lista in ordine.**
* **Precondizioni: Lista inizializzata.**
* **Postcondizioni: Le lezioni sono stampate nel terminale.**

**lezione.c**

1. **Specifiche Sintattiche**

**• Include di: stdio.h, stdlib.h, string.h, lezione.h, orario.h**

**Funzioni definite:**

* **creaLezione()**
* **stampaLezione(Lezione)**

### Specifiche Semantiche

**creaLezione()**

* **Sintassi: Lezione creaLezione();**
* **Semantica: Richiede all’utente i dati relativi a una nuova lezione (orario, istruttore, disciplina) e li restituisce in una struttura Lezione.**
* **Precondizioni: Nessuna.**
* **Postcondizioni: Restituisce una struttura Lezione inizializzata con i dati forniti.**

**stampaLezione()**

* **Sintassi: void stampaLezione(Lezione l);**
* **Semantica: Visualizza a terminale tutte le informazioni relative alla lezione fornita.**
* **Precondizioni: La struttura Lezione deve essere inizializzata correttamente.**
* **Postcondizioni: I dettagli della lezione sono visualizzati a schermo.**

**list.c**

**Specifiche Sintattiche**  
**• Include di: stdio.h, stdlib.h, list.h**

**Funzioni definite:**

* **inizializzaLista(Lista \*)**
* **inserisciElemento(Lista \*, Elemento)**

**inizializzaLista()**

* **Sintassi: void inizializzaLista(Lista \*lista);**
* **Semantica: Inizializza una lista vuota.**
* **Precondizioni: La struttura Lista deve essere dichiarata.**
* **Postcondizioni: La lista è inizializzata e pronta per l'uso.**

**inserisciElemento()**

* **Sintassi: void inserisciElemento(Lista \*lista, Elemento e);**
* **Semantica: Inserisce un nuovo elemento alla fine della lista.**
* **Precondizioni: La lista lista deve essere inizializzata; l'elemento e deve essere valido.**
* **Postcondizioni: L'elemento e è aggiunto alla lista.**

**list\_Lezioni.c**

1. **Specifiche Sintattiche  
   • Include di: stdio.h, stdlib.h, list\_Lezioni.h**

**Funzioni definite:**

* **inizializzaListaLezioni(ListaLezioni \*)**
* **inserisciLezione(ListaLezioni \*, Lezione)**

### 2. Specifiche Semantiche

**inizializzaListaLezioni()**

* **Sintassi: void inizializzaListaLezioni(ListaLezioni \*lista);**
* **Semantica: Inizializza una lista di lezioni vuota.**
* **Precondizioni: La struttura ListaLezioni deve essere dichiarata.**
* **Postcondizioni: La lista è inizializzata e pronta per l'uso.**

**inserisciLezione()**

* **Sintassi: void inserisciLezione(ListaLezioni \*lista, Lezione l);**
* **Semantica: Inserisce una nuova lezione alla fine della lista.**
* **Precondizioni: La lista lista deve essere inizializzata; la lezione l deve essere valida.**
* **Postcondizioni: La lezione è aggiunta alla lista.**

**list\_prenotazioni.c**

1. **Specifiche Sintattiche**

* **Include di: stdio.h, stdlib.h, list\_prenotazioni.h**

**Funzioni definite:**

* **inizializzaListaPrenotazioni(ListaPrenotazioni \*)**
* **inserisciPrenotazione(ListaPrenotazioni \*, Prenotazione)**

### Specifiche Semantiche

**inizializzaListaPrenotazioni()**

* **Sintassi: void inizializzaListaPrenotazioni(ListaPrenotazioni \*lista)**
* **Semantica: Inizializza la lista delle prenotazioni a vuota.**
* **Precondizioni: Struttura dichiarata.**
* **Postcondizioni: Lista prenotazioni pronta per l’uso**

**inserisciPrenotazione()**

* **Sintassi: void inserisciPrenotazione(ListaPrenotazioni \*lista, Prenotazione p)**
* **Semantica: Inserisce la prenotazione p in coda alla lista.**
* **Precondizioni: Lista inizializzata e prenotazione valida.**
* **Postcondizioni: La prenotazione è stata inserita.**

**orario.c**

1. **Specifiche Sintattiche  
   • Include di: stdio.h, stdlib.h, orario.h**

**Funzioni definite:**

* **verificaDisponibilitaOrario(ListaLezioni, Orario)**
* **stampaOrario(Orario)**

### 2. Specifiche Semantiche

**verificaDisponibilitaOrario()**

* **Sintassi: int verificaDisponibilitaOrario(ListaLezioni lista, Orario o);**
* **Semantica: Verifica se l'orario o è disponibile nella lista delle lezioni.**
* **Precondizioni: La lista lista deve essere inizializzata; l'orario o deve essere valido.**
* **Postcondizioni: Restituisce 1 se l'orario è disponibile, 0 altrimenti.**

**stampaOrario()**

* **Sintassi: void stampaOrario(Orario o);**
* **Semantica: Stampa a schermo l'orario fornito.**
* **Precondizioni: L'orario o deve essere valido.**
* **Postcondizioni: L'orario è visualizzato a schermo.**

**prenotazioni.c**

**Specifiche Sintattiche**

* **Include di: stdio.h, stdlib.h, string.h, prenotazioni.h**

**Funzioni definite:**

* **creaPrenotazione()**
* **stampaPrenotazione(Prenotazione)**

### 2. Specifiche Semantiche

**creaPrenotazione**

* **Sintassi: Prenotazione creaPrenotazione()**
* **Semantica: Richiede all’utente i dati per creare una prenotazione e restituisce la struttura popolata.**
* **Precondizioni: Input da terminale.**
* **Postcondizioni: Viene restituita una struttura Prenotazione inizializzata.**

**stampaPrenotazione**

* **Sintassi: void stampaPrenotazione(Prenotazione p)**
* **Semantica: Stampa sul terminale i dati contenuti nella prenotazione.**
* **Precondizioni: Prenotazione valida.**
* **Postcondizioni: I dettagli sono mostrati a video.**