**Report**

# **File**

## main.c

## prenotazioni.c

## prenotazioni.h

## abbonamenti.c

**Nodo\* crea\_nodo(Cliente c)**

Specifica sintattica:

Nodo\* crea\_nodo(Cliente c);

Specifica semantica:

* Input
  + Cliente c – la struttura con i dati del nuovo cliente che dovremmo aggiungere nel nodo che creeremo
* Output
  + Puntatore al nodo appena creato con all’ interno i dati del cliente c ed i puntatori ai nodi sx e dx
* Pre Condizione
  + Nessuna
* Post Condizione
  + Viene restituito un puntatore ad un nuovo nodo che viene allocato dinamicamente con i puntatori a dx e sx vuoti ed il cliente nuovo salvato
* Side Effect
  + Viene occupato uno spazio di memoria con la chiamata del malloc quindi dovremmo ricordarci di effettuare una free successivamente

**Nodo\* inserisci\_cliente(Nodo\* radice, Cliente c)**

Specifica sintattica:

Nodo\* inserisci\_cliente(Nodo\* radice, Cliente c)

Specifica semantica:

* Input
  + Nodo\* radice: il puntatore alla radice dell’ albero che avviamo creato(può essere anche NULL se si tratta del primo inserimento)
  + Cliente nuovo: i dati del cliente che andremo ad inserire nell’ albero
* Output
  + Puntatore alla radice aggiornata dell’ albero che include anche i dati del nuovo cliente
* Pre Condizione
  + L’albero deve essere binario e si deve basare sull’ id per la ricerca
* Post Condizione
  + Se l’ id non esiste già viene inserito correttamente rispettando l’ ordine dell’ albero
  + Se è gia presente, non ci sono modifiche è viene stampato un errore
* Side Effect
  + Potrebbero essere salvati dei duplicati

**Int abbonamento\_valido(Cliente c);**

Specifica sintattica:

Int abbonamento\_valido(Cliente c)

Specifica semantica:

* Input
  + Cliente c – la struttura con i dati del cliente
* Output
  + Restituisce 1 se l’abbonamento è ancora valido, 0 altrimenti
* Pre Condizione
  + Il tempo di attivazione deve essere stato inizializzato correttamente
* Post Condizione
  + La funzione calcola il tempo trascorso dalla mdata d’ attivazione alla data odierna e lo confronta con la durata dell’abbonamento (confrontando i giorni)
* Side Effect
  + Nessuno la funzione legge solo i dati non li modifica

**Void stampa\_clienti\_ordinati(Nodo\* radice)**

Specifica sintattica:

void stampa\_clienti\_ordinati(Nodo\* radice)

Specifica semantica:

* Input
  + Puntatore alla radice dell’ albero binario dove sono salvati i clienti
* Output
  + Nessuno la funzione è void
* Pre Condizione
  + L’albero deve essere già ordinato
* Post Condizione
  + Tutti i clienti ssalvati nell’ albero vengono stampati in ordine crescente di codice fiscale, con i dati riguardanti i loro abbonamenti tramite una funzione ricorsiva
* Side Effect
  + Viene stampato un output a schermo

**Void libera\_clienti(Nodo\* radice)**

Specifica sintattica:

void libera\_clienti(Nodo\* radice)

Specifica semantica:

* Input
  + Puntatore alla radice dell’ albero binario dove sono salvati i clienti
* Output
  + Nessuno la funzione è void
* Pre Condizione
  + L’albero deve essere stato gia allocato correttamente
* Post Condizione
  + Tutta la memoria allocata per l’ albero dei clienti viene liberata
* Side Effect
  + Dopo l’ esecuzione il puntatore a radice e tutti i nodi non potreanno più essere usati

## abbonamenti.h

## lezioni.c

## lezioni.h