**PROGETTAZIONE ADT** **list.c/.h**

* First (puntatore a nodo)
* Size (intero)

Struttura Node:

* C (puntatore a corso)
* Next (puntatore a nodo)

**PROGETTAZIONE FUNZIONI**

newList:

1. Creo e alloco una variabile lista.
2. Se la creazione è andata a buon fine, imposto il primo elemento della lista a NULL e la size della lista a 0
3. Ritorno la lista appena creata.

isEmpty:

1. Controllo se la lista esiste (caso negativo ritorno 1).
2. Se il primo elemento della lista non esiste, ritorno 1, altrimenti 0.

* Ritorno 1 Lista vuota o inesistente
* Ritorno 0 Lista piena

insertList:

1. Assegno ad una variabile nodo il ritorno della funzione insertNode, che prende come parametri, il puntatore alla testa della lista, la posizione e la variabile corso da inserire nella lista.
2. Controllo che l’operazione sia andata a buon fine (caso negativo ritorno 0).
3. Aggiorno il puntatore al primo nodo della lista con il nuovo nodo.
4. Incremento di uno la dimensione della lista.
5. Restituisco 1 per indicare il corretto inserimento nella lista.

insertNode:

1. Dichiaro due variabili nodo, e faccio puntare uno dei due nodi all nodo in testa alla lista, passato dai paramentri.
2. Alloco lo spazio per il nodo che conterrà il corso e controllo che la memoria sia stata allocata correttamente (caso negativo ritorno NULL).
3. Copio la variabile corso all’interno del nodo.
4. Controllo se l’inserimento deve essere in testa, o all’interno della lista.

* Se in testa (pos=0), assegno al puntatore next del nodo creato, l’indirizzo del nodo in testa alla lista. Ritorno il nodo creato (diventato la testa della lista).
* Se la posizione è diversa da 0, imposto un ciclo che scorre la lista fino alla posizione inserita o fino alla fine della lista.

Se il nodo è uguale a NULL, vuol dire che la lista è più corta della posizione indicata, dealloco lo spazio per il nodo che contiene il corso e ritorno NULL.

Se invece non è NULL, il nuovo nodo punterrà al nodo successivo di della posizione indicata, e aggiorno il nodo precedente al nuovo facendolo puntare al nodo stesso.

Ritorno la lista.

removeList:

1. Controlla se la lista è vuota o non inizializzata e ritorno 0.
2. Altrimenti richiamo la funzione removeNode, che prende la lista e la posizione e restituisce la lista aggiornata.
3. Decremento di uno la dimensione della lista.
4. Ritorno 1 per indicare la corretta rimozione dell’elemento.

removeNode:

1. Creo una variabile nodo, che punterà al nodo da eliminare.
2. Se l’eliminazione è in testa (pos=0):
   1. Assegno l’indirizzo della lista al nuovo nodo creato.
   2. Aggiorno la lista, assegnando al nodo in testa alla lista l’indirizzo del nodo successivo.
   3. Libero il nodo creato che conteneva il corso da cancellare.
3. Se l’eliminazione non è in testa:
   1. Scorro la lista fino alla posizione indicata.
   2. Controllo che il nodo alla posizione esista.
   3. Se esiste assegno l’indirizzo del nodo da cancellare al nodo creato, collego il nodo precedente al nodo successivo al nodo da cancellare e libero il nodo da cancellare.
4. Ritorno la lista aggiornata.

reverseList:

1. Creo la lista che conterrà gli elementi invertiti.
2. Scorro la lista passata come parametro.
3. Man mano che scorro inserisco nella nuova lista gli elementi, tramite la funzione insertList, con posizione uguale a 0, per l’inserimento in testa.
4. Se l’inserimento fallisce il programma si chiude.
5. Ritorno la lista con gli elementi invertiti.

ricercaGenericaLista:

1. Controllo che la lista esista (caso negativo ritorno 0).
2. Creo una lista che conterrà i riscontri della ricerca (questo per gestire corsi che hanno lo stesso nome).
3. Scorro ogni elemento della lista e tramite un selettore imposto il parametro di ricerca:

* Selettore=0 Ricerca per ID
* Selettore=1 Ricerca per Nome

1. Quando trovo l’elemento lo inserisco nella nuova lista tramite il comando insertList, con posizione uguale a 0.
2. Al termine del ciclo ritorno la lista.

ricercaData:

1. Controllo che la lista esista (caso negativo ritorno 0).
2. Creo una lista che conterrà i riscontri della ricerca (questo per gestire corsi che hanno la stessa data).
3. Scorro ogni elemento della lista.
4. Per confrontare le date uso la funzione confrontaData, che ritorna 0 quando sono uguali.
5. Quando trovo l’elemento lo inserisco nella nuova lista tramite il comando insertList, con posizione uguale a 0.
6. Al termine del ciclo ritorno la lista.

ricercaMese:

1. Controllo che la lista esista (caso negativo ritorno 0).
2. Creo una lista che conterrà i riscontri della ricerca (questo per gestire corsi che hanno lo stesso mese).
3. Scorro ogni elemento della lista.
4. Quando trovo l’elemento lo inserisco nella nuova lista tramite il comando insertList, con posizione uguale a 0.
5. Al termine del ciclo ritorno la lista.

ricercaOrario:

1. Controllo che la lista esista (caso negativo ritorno 0).
2. Creo una lista che conterrà i riscontri della ricerca (questo per gestire corsi che hanno lo stesso orario).
3. Scorro ogni elemento della lista.
4. Per confrontare gli orari uso la funzione confrontaOrario, che ritorna 0 quando sono uguali.
5. Quando trovo l’elemento lo inserisco nella nuova lista tramite il comando insertList, con posizione uguale a 0.
6. Al termine del ciclo ritorno la lista.

lezioniInEvidenza:

1. Controllo che la lista esista.
2. Inizializzo tre variabili intere e tre variabili corso, che serviranno per tenere traccia delle lezioni più frequentate.
3. Scorro la lista per analizzare uno ad uno il numero di partecipanti di ogni corso.

* Se il numero di partecipanti è maggiore del valore della variabile primo, i valori vengono scalati, sia le variabili che contengono il numero di partecipanti che i corsi stessi.
* Stesso discorso viene applicato per il secondo e il terzo posto.

1. Se le variabili corso esistono, stampo in ordine i tre corsi più frequentati, tramite la funzione stampaCorso.

stampaLista:

1. Controllo che la lista esista.
2. Scorro la lista elemento per elemento, per poi stampare le informazioni del corso tramite la funzione stampaCorso.

scriviFileCorso:

1. Apro il file corsi.txt in modalità scrittura, e controllo la corretta apertura del file.
2. Scorro la lista elemento per elemento.
3. Tramite la funzione scriviCorso scrivo su corsi.txt i campi di corso.
4. Chiudo il file corso.txt.

getFirstCorso:

1. Controllo che la lista esista (caso negativo ritorno NULL).
2. Ritorno il corso del primo nodo della lista.

cancellaCorso:

1. Controllo che la lista esista (caso negativo ritorno 0).
2. Scorro la lista elemento per elemento, affinche non trovo il corso con lo stesso ID passato come parametro.
3. Per cancellarlo richiamo la fuzione removeList e controllo l’uscita del valore:

* remoreList==0 ritorno 0 errore nella cancellazione.
* removeList==1 ritorno 1 cancellazione avvenuta.

1. Se al termine del ciclo non viene trovato l’elemento da cancellare ritorno 0.