# Lines in Data.h

* **addNode** - line 73
* **deleteNode** - line 78
* **returnNode** - line 144
* **displayList** - line 215
  + **movieSearch** - 228
  + **showPoster** - 253
* **showList** - line 277
* **checkNode** - line 297
* **newCustomer** - line 326
* **displayCustomer** - line 346
* **listRented** - line 363
* **rentSearch** – line 378
* **movieRent** – line 413
* **loadMemory** – line 426
* **updateSave** – line 574

# Main

**Variables**

* **functionlist execute** - object of class functionlist
* **int ch** - for main choice [1-8]
* **int ch2** - for customer menu choice [1-3]
* **char ch3** - for customer menu: if you would like to try again [Y/N]

**execute.loadMemory()** – Will load all the information from the text file to the linked list to be able to input all the default information from the text file to the nodes of the list.

**Inside the do loop:**

* **ch = mainMenu()** – Assign the returned value of tempch to ch.

**Inside if:**

* **Int movieSize = movieList.size()** – Assign the size of the movieList to movieSize.
* **movieSize += 101** – Used so that the movie ID would start at 126. Since there are 25 movies inside the list, the size is 25 (25 nodes). We added 101 because the auto generated ID should start at 126.
* **Choice 1:** //add movie
  + - **cout << movieSize** – Output the auto generated id
    - **mDetails.mid = movieSize** – assign the value of auto generated code from movieSize to mDetails.code (linked list variable).
    - **Input:** movie title, genre, production, copy, file name (**mDetails**: Object of the struct **movieList** )
    - **movieSize++** – Increment the auto generated code.
    - **execute.addNode()** – Create a node and add the new movie information to the linked list.
* **Choice 2:** //delete or rent
* **Input:** cDetails.cid (inside the struct of customerInfo), mDetails.code (inside the struct of movieInfo)
* **execute.deleteNode()** – Delete the node where the customer ID and movie ID matched.
* **Choice 3:** //append or return
* **Input:** cDetails.cid (inside the struct of customerInfo), mDetails.code (inside the struct of movieInfo)
* **execute.returnNode()** – Will delete the specific node in the linked list where the customer ID and movie ID matched since it is already returned. Will output the information about the movie returned and will show that the copies incremented.
* **Choice 4:** //traverse or display specific movie info based on the movie code
* **Input:** mDetails.code
* **execute.displayList()** – Will traverse the linked list and print every element or movie info inside the specific node where the movie ID matched. Will also show the movie poster of that particular movie.
* **Choice 5:** //traverse all movies and display all movie info
* **execute.showList ()** – Will traverse inside the linked list and print every element in each node to display all the movie information. It will output all the movie inside the list.
* **Choice 6:** //check availability based on movie ID
* **Input:** mDetails.code
* **execute.checkNode** – Check if a specific movie is available or in the list. Will also check if the movie still has copies. If the movie ID matched, it will output the movie information and show the movie poster.
* **Choice 7:** //customer maintenance
* Inside do:

**ch2 = customerMenu()** – Assign the returned value of tempch to ch2.

Inside if:

* **Choice 1:** **execute.newCustomer()** – Add customer info into the linked list with auto generated ID starting from 11, **Input:** name & address.
* **Choice 2:** **execute.displayCustomer()** – Display customer details (name & address) based on the customer ID.
* **Choice 3:** **execute.listRented()** – List all rented movies based on the customer ID.
* **Choice 8:** //update text file
* **execute.updateSave()** – Will output or transfer all the information inside the linked list into the text file.It doesn’t use append. It rewrites everything inside the text file.Since the linked lists are being updated as the program runs, the text files will also be updated when you choose exit.

**mainMenu()**

int tempch then return and pass the value to ch

**customerMenu()**

int tempch then return and pass the value to ch2

# Data.h

**Structs:**

struct movieInfo //all info for movie

struct customerInfo //all info for customer

struct customerRent //all info for rent

**Linked List Declaration:**

declaration of stl linked list (for all struct)

declaration for iterators (for all struct)

**class functionlist:**

General Functions - all load and update

Movie Functions - all movie functions (append, traverse, delete, file handling)

Customer Functions - all customer functions (append, traverse, delete, file handling)

## Movie Functions:

**addNode()**

uses linked list stl

movieList.push\_back(mDetails) – Adds a new element/node mDetails at the end of the list.

-------------------------------------------------

**deleteNode()**

bool msearch = false;

bool csearch = false;

**Inside for loop:**

1. Traverse first to last using for loop with .begin(), returns an iterator pointing to the first element of the list, and .end(), returns an iterator pointing to the theoretical last element which follows the last element.
2. if nagmatch, yung cit--> name mapupunta sa cDetails.name then cit --> address mapupunta sa cDetails.address. Lastly, yung bool na csearch magiging true

else - if di namatch mapupunta sa else continue ng for then lalabas ng for and mapupunta sa if-else the else then output "ERROR. Please enter valid Customer ID."

1. if nagmatch yung customer ID, diba magiging true ung csearch. If nagtrue naman yung csearch, papasok siya sa second for loop

**Inside second for loop:**

1. Traverse first to last using for loop with .begin(), returns an iterator pointing to the first element of the list, and .end(), returns an iterator pointing to the theoretical last element which follows the last element.
2. if nagmatch yung movie ID and may copies pa papasok sita sa if ulit
3. Inside if:

Movie successfully rented then output the rented movie information

updating number of copies then subtract 1 to show that one copy is rented

3.1 pass mDetails.title to iterator mit-->title

3.2 execute the function **movieRent(cDetails.cid, cDetails.name, cDetails.address, mDetails.mid, mDetails.title)**

3.3 , yung bool na msearch magiging true

3.4 break

if no copies are available, Error.

If it didn't match, error //edit

-------------------------------------------------

**returnNode()**

bool rseacrh = false;

bool msearch = false;

rDetails.rcid = cDetails.cid

rDetails.rmid = cDetails.mid

\* Meaning nito ipapasa yung user input ng main sa struct kung saan nandun yung pang rent

**Inside for loop:**

1. Traverse first to last using for loop with .begin(), returns an iterator pointing to the first element of the list, and .end(), returns an iterator pointing to the theoretical last element which follows the last element.
2. Inside if: if rit--->rcid == rDetails.rcid, go inside second if
3. Inside second if: if rit-->rmid == rDetails.rmid
   1. if mathced, erase the rit

\*erase() – Removes a single element or a range of elements from the list.

3.2 rsearch = true

3.3 else: continue - pupunta siya sa labas na else and output “"Error. Invalid Customer/Movie ID!"

5. If rsearch == true, iterate inside the movieList

**Inside if:**

if (mit->mid == mDetails.mid) – if movie ID is found

**Inside inner if:**

if(mit->copy > 0) – check if copy is greater than 0, if yes then output the returned movie and increment the copy to show that the movie has been returned. Finally, set msearch to true.

**Else:** continue 0 – will output "Error. No copies available!"

-------------------------------------------------

**displayList(int movieID)**

**Inside if:**

1. execute **movieSearch(int movieID),** output the movie details based on the movie ID, if movie wasn't found, output error
2. else: continue - output outside if above, "Error. The movie is not on the list!"

**Inside another if:**

1. execute **showPoster(int movieID),** show poster based on movie ID
2. if the poster wasn't found, output error
3. else: continue - output outside if above, "Error. File not found!"

-------------------------------------------------

**showList()**

bool msearch = false;

**Inside for loop:**

1. Traverse first to last using for loop with .begin(), returns an iterator pointing to the first element of the list, and .end(), returns an iterator pointing to the theoretical last element which follows the last element.
2. Display all the movies with their details in table form.

-------------------------------------------------

**checknode(int movieID)**

**Inside if:**

1. execute **movieSearch(int movieID),** output the movie details based on the movie ID
2. if movie wasn't found, output error
3. else: continue - output outside if above, "Error. The movie is not on the list!"

**Inside another if that checks the number of copies:**

1. **Inside if:**

1.1 if mit-->copy is greater than 0, output movie is available

1.2 else: output movie is not available

**Inside another if:**

1. execute **showPoster(int movieID),** show poster based on movie ID
2. if poster wasn't found, output error
3. else: continue - output outside if above, "Error. File not found!"

## Customer Functions:

**newCustomer()**

1. get the size of the customerList then assign to customerSize
2. customerSize++
3. cDetails.cid = customerSize
4. Input:

cDetails.name

cDetails.address

4.1 customerList.push\_back(cDetails) ) – Adds a new element/node cDetails at the end of the list.

-------------------------------------------------

**displayCustomer()**

1. Input: cDetails.cid

Inside for loop:

Traverse first to last using for loop with .begin(), returns an iterator pointing to the first element of the list, and .end(), returns an iterator pointing to the theoretical last element which follows the last element.

Inside if:

1. if cit-->cid == cDetails.cid
2. output cit-->name and cit-->address

-------------------------------------------------

**listRented()**

Input: cDetails.cid

Inside if:

-------------------------------------------------

## Rent Functions:

**bool functionList::execute rentSearch(int rentID)**

bool rsearch = false

bool msearch = false;

Inside for loop: Iterate the rentList

1. Inside If: if rit-->rcid == rentID
2. search = true
3. output the customer name, address, rcid

Inside For loop:

1. traverse rented list
2. Output:

rit --> rmid (movie ID)

rit --> rtitle (movie title)

1. break;
2. else: continue - if movie is not rented, output "Error. Customer not found!"

-------------------------------------------------

**movieRent(int customerID, string customerName, string customerAddress, int movieID, string movieTitle)**

assign:

rDetails.rcid = customerID

rDetails.rmid = movieID

rDetails.rname = customerName

rDetails.raddress = customerAddress

rDetails.title = movieTitle

rentList.push\_back(rDetails); – Adds a new element/node rDetails at the end of the list.

## General Functions:

**loadMemory()**

**Variables:**

string movieLine, customerLine, rentLine – declared these strings to be later used when each line in a file was read.

Int perLine – for switch case.

Ifstream movieFile, customerFile, rentFile - we used if stream since we want to access the info inside the text file and put it inside a linked list.

**Inside if:**

\* .is\_open() - Returns whether the stream is currently associated to a file.

1. **Assign perLine to 0**
2. **Inside While:**

While loop with getline (movieFile, movieLine) means while reading each line from movieFile text file and assigning it to movieLine.

\*getline - a part of the <string> header. The getline() function extracts characters from the input stream and appends it to the string object until the delimiting character is encountered.

1. **Inside switch**

switch perLine for assigning each line to perLine

3.1 case 0 means first line of the text file and it will be assigned as movie ID in the struct movieInfo with the object mDetails and so on.

\*atoi() - Convert string to integer

1. **Inside If:**
   1. If perLine reaches the 6th line, it will increment so the next movie details will be read.
   2. Else perline will be 0 again and it will be added to the movieList Linked List

\*Same logic with the other two (customerList and rentList)

-------------------------------------------------

**updateSave()**

**Variable:**

Ofstream movieFile, customerFile, textFile - we want to output or transfer what is inside the linked list into the text file

.open() - Opens the file identified by argument filename

**Inside for loop:**

1. Traverse first to last using for loop with .begin(), returns an iterator pointing to the first element of the list, and .end(), returns an iterator pointing to the theoretical last element which follows the last element.
2. We used a movieInfo iterator, the movieInfo is the struct that holds the variables in each node. And we named the iterator as mit.
3. To get inside each node, we used a for loop and the pointer named mit will traverse until the last node. Before it gets into the next node, it prints all the information inside each node.The lines inside the for loop basically means that the mit pointer will point to a specific node and it will print all the information inside each node.
4. To be able to access the specific element inside, we used the mit -->mid etc.
5. Then we close the file. - A file needs to be closed after a read or write operation to release the memory allocated by the program.
6. In the update save we didn't use append, everything will be printed again and the information before inside the text file will be overwritten

\*Same logic with the other two (customerList and rentList)

**loadMemory()**

Since the assigned value is 0, it will go inside the case 0, where the first line will be read and it will be assigned to movie ID inside the struct of movieInfo (mDetails.mid).

Then it will get outside the loop and will go to the if statement, the code inside the statement will increment the perLine to 1.

Since it is already 1, it will go inside the case 1. This process will repeat until it reaches the case 6.

When it reaches the case 6, it will go to the else statement. The statement will assign the perLine to 0 again and will push all the information inside the movieList.

After that, since it is still inside the while loop and there are still lines inside the text file, it will repeat and go inside the case 0 again until all the information are pushed inside the movieList.