Functions:

➢ Log In Control Functions::

- **int main();**
  - **Purpose**: It is the entire point of the program. Without this program cannot run.
  - **Description**: This function controls the menu on the screen and gets choice. Here I called the logIn page function which takes a choice in parameter to control the all functionality of log-in.

- **void logInPage(char choice);**
  - **Purpose**: Manage the logIn functionality of the program.
  - **Parameters**:`choice` (character data type): Based on choice is decided on which side log-in admin/customer
  - **Description**: dynamically allocate memory. and called the logInControl function to log in the desired side , this function takes three arguments.

- **void logInControl(LogIn *info, int &size, int &infoIdx);**

  - **Purpose**: Manages user login authentication and user index retrieval.
  - **Parameters**:
    - `info` (pointer to LogIn): Array containing user credentials (user ID and password).
    - `size` (a reference to an integer): Size of the array containing user credentials.
    - `infoIdx` (reference to an integer): Identifier of the authenticated user.
  - **Description**: This function handles the login authentication process by validating user-provided credentials against the stored user data. It iterates through the array of user credentials to match the entered user ID and password.

    - If the credentials match, it assigns the `infoIdx` variable to the identifier of the authenticated user, granting access to specific functionalities based on the user's role (admin or customer).
    - The `size` parameter helps control the iteration through the array of user credentials, ensuring all stored credentials are checked for validation.

➢ Admin Functions::

- **`void adminPage(int &infoIdx);`**

    - **Purpose**: Manages the functionality accessible to an admin after successful login.
    - **Parameters**:
        - `infoIdx` (a reference to an integer): Index or identifier for the admin user where the log-in admin data is stored.
    - **Description**: This function controls and displays various administrative functionalities such as changing passwords, managing customers, cars, generating reports, etc.

- **void changePassword(int &infoIdx);`**

    - **Purpose**: Allows the logged-in user, both admin and customer, to change their password.
    - **Parameters**:
        - `infoIdx` (a reference to an integer): Index or identifier for the user whose password is being changed.
    - **Description**: This function facilitates the change of passwords for logged-in users, providing a secure method to update their login credentials.

- **void addRemoveCus();`**

    - **Purpose**: Handles the addition or removal of customer records.
    - **Description**: This function provides functionality for administrators to add or remove customer data from the system.

- **void addRemoveCar();`**

    - **Purpose**: Manages the addition or removal of car records.
    - **Description**: This function allows administrators to add new cars to the rental system or remove existing ones.

- **void viewCusCar();`**

    - **Purpose**: Displays the available cars to customers for rental.
    - **Description**: This function presents a list of cars that are available for rent to customers using the system.

- **void customerDataUpdate();`**

    - **Purpose**: Handles updates to customer data.
    - **Description**: Provides functionality for updating customer information or records within the system.

- **void reportGenerate();`**

    - **Purpose**: Generates reports related to the car rental system.
    - **Description**: This function generates various reports related to customer activity, car rentals, revenue, etc., providing valuable insights into system usage.

➢ Customer Functions::

- **void customerPage(int &infoIdx);`**

    - **Purpose**: functionalities accessible to customers after successful login.
    - **Parameters**:
        - `infoIdx` (a reference to an integer): index of customer array where the data is stored of a customer who log-in.
    - **Description**: Manages various functionalities available to customers such as viewing available cars, renting cars, returning cars, generating reports, etc.

- **void cusViewAvailCar();`**
    - **Purpose**: Displays available cars to customers for rental.
    - **Description**: This function specifically shows available cars to customers for rental purposes.

- **void rentCar(int &infoIdx, CD *customerInfo, int &NUM_OF_CUSTOMER);`**
    - **Purpose**: Facilitates the rental process for customers.
    - **Parameters**:
        - `infoIdx` (a reference to an integer): index of customer array where the data is stored of a customer who log-in.
        - `customerInfo` (pointer to CD): Array of customer information.
        - `NUM_OF_CUSTOMER` (reference to an integer): Total number of customers in the system.
    - **Description**: Allows a customer to rent a car by managing the rental process and updating related data accordingly.

- **void returnCar(int &infoIdx, CD *customerInfo, int &NUM_OF_CUSTOMER);`**
    - **Purpose**: Handles the process of returning a rented car by a customer.
    - **Parameters**:
        - `infoIdx` (reference to an integer): Index of customer array where the data is stored of a customer who log-in.
        - `customerInfo` (pointer to CD): Array of customer information.
        - `NUM_OF_CUSTOMER` (reference to an integer): Total number of customers in the system.
    - **Description**: Manages the return process of a car rented by a customer, updating the relevant data in the system.

- **void cusReport(string userId);`**
    - **Purpose**: Generates reports related to a specific customer.
    - **Parameters**:
        - `userId` (string): Identifier for the customer whose report is being generated.
    - **Description**: Creates reports specific to a particular customer, providing details about their rental history or other relevant information.

➢ Input Validation Functions::

- **void charValidate(char &ch, string str, char mn, char mx);**
    - **Purpose**: Validates and restricts user input to a specific character range.
    - **Parameters**:

        - `ch` (reference to a character): User input character to be validated.
        - `str` (string): Message or prompt displayed to guide user input.
        - `mn` (character): Minimum allowed character in the range.
        - `mx` (character): Maximum allowed character in the range.
    - **Description**: This function ensures the validation of user input within a specified character range. It prompts the user with the provided message (`str`) and restricts input to be within the defined character boundaries (`mn` to `mx` inclusive)

- **void stringInput(string str, string &value);**
    - **Purpose**: Accepts and stores user input as a string.
    - **Parameters**:
        - str` (string): Message or prompt displayed to guide user input.
        - `value` (a reference to a string): Variable to store the user input.
    - **Description**: This function prompts the user with the provided message (`str`) to enter a string value. It accepts the user input as a string and stores it in the referenced variable `value`.
    - It validates that anywhere no data is missed as a string.
- **void fileChecking(fstream &);**
    - - **Purpose**: Checks the status or validity of a file stream.
    - - **Parameters**:
    - - `fstream &` (a reference to a file stream): The file stream to be checked.
    - - **Description**: Verifies the status of a file stream, ensuring its readiness for file operations.

Other Function::

This function is made because line of code is repeat multiple times in. So, reduce code repetition. It follows:

- **void displayHeader(string str);**
- **void cusDataFetch(CD *customerInfo, int &NUM_OF_CUSTOMER, fstream &Fetch);**
- **void cusDataWrite(CD *customerInfo, int &NUM_OF_CUSTOMER, fstream &Write);**
- **void carDataFetch (CarDetail *carInfo, int &NUM_OF_CAR, fstream &Fetch);**
- **void carDataWrite(CarDetail *carInfo, int &NUM_OF_CAR, fstream &Write);**

STRUCTURES:

Three structures: `LogIn`, `CustomerDetail` (alias `CD`), and `CarDetail` are defined:

➢ LogIn` Structure:

****Members**:**

- `userId` (string): Stores the user ID for login authentication.
- `password` (string): Stores the password for login authentication.

➢ CustomerDetail` Structure (Alias `CD`):

****Members**:**

- `cusId` (LogIn): Stores the login credentials for the customer.
- `status` (string): Represents the status of the customer.
- `name` (string): Stores the name of the customer.
- `city` (string): Stores the city information for the customer.
- `phoneNo` (string): Stores the phone number of the customer.

➢ CarDetail` Structure:

****Members**:**

- `noPlate` (string): Represents the number plate of the car.
- `name` (string): Stores the name of the car.
- `model` (string): Represents the model of the car.
- `status` (string): Represents the availability status of the car (defaulted to "Available").
- `rentCusId` (string): Stores the customer ID to whom the car is rented.
- `date` (string): Represents the date of car rental.
- `rentedDay` (integer): Stores the number of days the car has been rented.
- `numRented` (integer): Stores the number of times the car has been rented.
- `price` (float): Represents the rental price of the car.