This is a program in C++ that defines a class called Vehicle and three classes (Car, Bike, and Bus) that inherit from Vehicle.

The Vehicle class has the following public member functions:

addVehicle: adds one or more records of vehicles to a file.

displayVehicle: displays the data for all vehicles stored in a file.

search: searches for a record of a vehicle in a file based on the vehicle number and displays the data for the vehicle if it is found.

deleteData: deletes a record of a vehicle from a file based on the vehicle number.

update: updates the record of a vehicle in a file based on the vehicle number.

The Vehicle class also has the following public member variables:

vNum: an integer representing the vehicle number.

name: a character array for storing the name of the vehicle.

oName: a character array for storing the name of the owner of the vehicle.

The Car, Bike, and Bus classes inherit from the Vehicle class and have the following additional member variables:

carCounter, bikeCounter, busCounter: an integer representing the number of vehicles of each type (cars, bikes, or buses).

carCharge, bikeCharge, busCharge: an integer representing the charge for parking a vehicle of each type.

The Car, Bike, and Bus classes also have the following public member functions:

getCarCounter, getCarCounter, getCarCounter: returns the number of vehicles of each type.

getCarCost, getCarCost, getCarCost: returns the charge for parking a vehicle of each typeThe class also has five member functions:

1. This is a function in the Vehicle class called addVehicle. It takes two arguments:

out: an ofstream object (output file stream) that is used to write data to a file.

vehicle: an object of the Vehicle class.

The purpose of this function is to allow the user to add one or more records of vehicles to a file called "car.txt". Here's how it works:

The function starts by checking if the "car.txt" file is open for writing. If it is not, it displays an error message.

The function then enters a loop that prompts the user to enter data for the vehicle number, name, and owner name. It then writes the data to the "car.txt" file as a binary record (out.write((char \*)&vehicle, sizeof(vehicle))). The loop continues until the user indicates that they do not want to add any more vehicles (by entering "n" or "N" when prompted).

Finally, the function closes the "car.txt" file.

2. This is a function in the Vehicle class called displayVehicle. It takes two arguments:

in: an ifstream object (input file stream) that is used to read data from a file.

vehicle: an object of the Vehicle class.

The purpose of this function is to display the data for all vehicles stored in a file called "car.txt". Here's how it works:

The function starts by opening the "car.txt" file in binary mode for reading (ios::binary | ios::app). If the file fails to open, it displays an error message.

The function then displays a header line with the column names ("Vnum", "Vname", and "Oname").

The function reads the contents of "car.txt" into the vehicle object one record at a time until the end of the file is reached. For each record, it displays the data for the vehicle (vehicle number, name, and owner name).

Finally, the function closes the "car.txt" file.3. This is a function in the Vehicle class called search. It takes three arguments:

in: an ifstream object (input file stream) that is used to read data from a file.

vehicle: an object of the Vehicle class.

num: an integer representing the vehicle number.

The purpose of this function is to search for a record of a vehicle in a file called "car.txt" based on the vehicle number and display the data for the vehicle if it is found. Here's how it works:

The function starts by opening the "car.txt" file in binary mode for reading (ios::binary | ios::app). If the file fails to open, it displays an error message.

The function then reads the contents of "car.txt" into the vehicle object one record at a time until the end of the file is reached. For each record, it checks if the vehicle number stored in the object (vehicle.vNum) matches the num argument. If it does match, the function displays the data for the vehicle and breaks out of the loop. If the vehicle number does not match, the function sets a flag (exist) to indicate that the vehicle was not found.

After the loop has completed, the function checks the value of the exist flag. If it is false, the function displays a message indicating that the vehicle was not found.

Finally, the function closes the "car.txt" file

4. This is a function in the Vehicle class called deleteData. It takes four arguments:

in: an ifstream object (input file stream) that is used to read data from a file.

out: an ofstream object (output file stream) that is used to write data to a file.

vehicle: an object of the Vehicle class.

num: an integer representing the vehicle number.

The purpose of this function is to delete a record of a vehicle from a file called "car.txt" based on the vehicle number. Here's how it works:

The function starts by opening the "car.txt" file in binary mode for reading (ios::binary | ios::app | ios::in) and opens a file called "temp.txt" in binary mode for writing (ios::binary | ios::out). If either file fails to open, it displays an error message.

The function then reads the contents of "car.txt" into the vehicle object one record at a time until the end of the file is reached. For each record, it checks if the vehicle number stored in the object (vehicle.vNum) matches the num argument. If it does not match, it writes the record to "temp.txt". If it does match, the record is skipped and not written to "temp.txt".

After all records have been processed, the function closes both "car.txt" and "temp.txt" and then removes "car.txt". It then renames "temp.txt" to "car.txt", effectively replacing the original file with the modified version.

Finally, the function displays a message indicating that the vehicle with the specified number has been removed successfully

5. This is a function in the Vehicle class called update. It takes three arguments:

iof: an fstream object (file stream) that is used for both reading and writing data to a file.

vehicle: an object of the Vehicle class.

num: an integer representing the vehicle number.

The purpose of this function is to update the record of a vehicle in a file called "car.txt" based on the vehicle number. Here's how it works:

The function starts by opening the "car.txt" file in binary mode for both reading and writing (ios::binary | ios::app | ios::out | ios::in) and moves the file pointer to the beginning of the file (iof.seekg(0)). If the file fails to open, it displays an error message.

The function then reads the contents of "car.txt" into the vehicle object one record at a time until the end of the file is reached. For each record, it checks if the vehicle number stored in the object (vehicle.vNum) matches the num argument. If it does match, the function displays the existing data for the vehicle, prompts the user to enter updated data for the vehicle number, name, and owner name, and then writes the updated record back to the file at the current position of the file pointer (iof.seekp(-sizeof(vehicle), ios::cur)).

If the vehicle number does not match any records in the file, the function displays a message indicating that the vehicle does not exist.

Finally, the function closes the "car.txt" file

This is a class called Car that is derived from the base class Vehicle. It has two private member variables, carCounter and carCharge, and three public member functions. The first is a constructor, which is a special type of function that is called when an object of this class is created. The constructor initializes the carCounter variable to 0 and the carCharge variable to 500. The other two member functions are getters, which are functions that allow other parts of the code to retrieve the values of the private member variables. The getCarCounter function returns the value of the carCounter variable, and the getCarCost function returns the value of the carCharge variable.

The class Bike is a subclass of Vehicle which means it inherits the properties and methods of the Vehicle class. The Bike class has two private member variables: bikeCounter and bikeCharge. It also has a constructor which initializes these member variables to 0 and 500 respectively. The class also has two public member functions: getCarCounter and getCarCost which return the values of the bikeCounter and bikeCharge member variables respectively. These member functions can be called on an object of the Bike class to get the values of these member variables.

This code defines a class called Bus that represents a bus in some system. The Bus class is a subclass of the Vehicle class, meaning that it inherits all of the properties and methods of the Vehicle class.

The Bus class has three private variables: busCounter, busCharge, and a default constructor that initializes these variables to 0 and 500, respectively. It also has two public methods: getCarCounter and getCarCost, which return the values of busCounter and busCharge, respectively.

The Bus class is similar to the Car and Bike classes, which are also subclasses of Vehicle and have similar structure and functionality.

This is a function defined in the Vehicle class that adds a vehicle to a file. The function takes an ofstream object and a Vehicle object as arguments. The ofstream object is used to write data to a file and the Vehicle object represents the vehicle that is to be added to the file.

The function first checks if the ofstream object is valid. If it is not, it prints an error message indicating that the file could not be opened. If the ofstream object is valid, the function enters a loop that prompts the user to input the vehicle number, name, and owner name. These values are then stored in the fields of the Vehicle object. The Vehicle object is then written to the file using the write function of the ofstream object. The loop continues until the user enters "n" or "N" to indicate that they do not want to add another vehicle. Finally, the ofstream object is closed.

This function reads the contents of a file called "car.txt" and displays the contents of the file on the screen. The file is opened in binary mode and in append mode, which means that new data can be added to the file but existing data cannot be overwritten. If the file cannot be opened, an error message is printed to the screen. Otherwise, the function displays a table header and then reads the contents of the file one record at a time using the read() function. For each record, the function extracts the values of the vNum, name, and oName fields of the vehicle object and displays them on the screen. Once all records have been read and displayed, the file is closed.

This function is a member function of the Vehicle class. It searches for a vehicle record in a file with a given number. The file is specified by the input file stream in. The vehicle object is used to store the data read from the file. The num parameter specifies the number of the vehicle to search for.

The function starts by opening the file "car.txt" in binary mode for appending. If the file fails to open, it prints an error message. Otherwise, it reads the records in the file one by one and compares the vehicle number with the given number num. If a matching record is found, it prints the vehicle number, name, and owner name. It sets the exist flag to true and breaks out of the loop. If no matching record is found, it sets the exist flag to false and continues reading until the end of the file. After the loop, it checks the exist flag. If it is false, it prints a message saying that the vehicle does not exist. Finally, it closes the file.

This function is designed to delete a record from a file that contains records of vehicles. The function takes in three parameters: two file streams (in and out), and an integer (num). The in file stream is opened in binary mode with ios::binary, and the ios::app and ios::in flags, which allow reading from and appending to the file. The out file stream is opened in binary mode with ios::binary and the ios::out flag, which allows writing to the file.

The function begins by opening the file "car.txt" using the in file stream, and opening a new file "temp.txt" using the out file stream. If either of these file streams cannot be opened, an error message is displayed.

Next, the function enters a loop that reads the contents of the file "car.txt" one record at a time, using the in.read function and casting the vehicle object to a character array. If the vehicle number of the current record does not match the value of num, the record is written to the file "temp.txt" using the out.write function. If the vehicle number does match num, the record is skipped and not written to the file.

After the loop has completed, the function closes both file streams and removes the file "car.txt". It then renames the file "temp.txt" to "car.txt", effectively replacing the original file with a version that does not contain the record with the specified vehicle number. A message is displayed to confirm that the record has been deleted successfully

The class Vehicle is a base class that represents a vehicle with a vehicle number, name, and owner name. It has several member functions which allow you to add a vehicle to a file, display all the vehicles in a file, search for a specific vehicle in the file, delete a vehicle from the file, and update the information for a vehicle in the file.

The class Car is a derived class from Vehicle and represents a specific type of vehicle. It has private member variables carCounter and carCharge, and public member functions getCarCounter and getCarCost. The Car class has a default constructor which initializes carCounter to 0 and carCharge to 500.

The class Bike is a derived class from Vehicle and represents a specific type of vehicle. It has private member variables bikeCounter and bikeCharge, and public member functions getBikeCounter and getBikeCost. The Bike class has a default constructor which initializes bikeCounter to 0 and bikeCharge to 500.

The class Bus is a derived class from Vehicle and represents a specific type of vehicle. It has private member variables busCounter and busCharge, and public member functions getBusCounter and getBusCost. The Bus class has a default constructor which initializes busCounter to 0 and busCharge to 500.

The addVehicle function allows you to add a new vehicle to a file. It takes in an output file stream out and an object of the Vehicle class vehicle. It prompts the user to enter the vehicle number, name, and owner name, and writes this information to the file. It also allows the user to add multiple vehicles by prompting the user to enter another vehicle.

The displayVehicle function allows you to display all the vehicles in a file. It takes in an input file stream in and an object of the Vehicle class vehicle. It reads the file and displays the vehicle number, name, and owner name for each vehicle.

The search function allows you to search for a specific vehicle in a file. It takes in an input file stream in, an object of the Vehicle class vehicle, and an integer num. It searches the file for a vehicle with a matching vehicle number and displays the vehicle number, name, and owner name if it is found. If the vehicle is not found, it displays a message saying so.

The deleteData function allows you to delete a specific vehicle from a file. It takes in an input file stream in, an output file stream out, an object of the Vehicle class vehicle, and an integer num. It reads the file and writes all the vehicles to a temporary file except for the one with a matching vehicle number. It then removes the original file and renames the temporary file to the original file name, effectively deleting the specified vehicle.

This is a program that defines a class called Vehicle and its derived classes Car, Bike, and Bus. The Vehicle class has data members vNum, name, and oName, which represent the vehicle number, name, and owner name, respectively. It also has member functions addVehicle, displayVehicle, search, deleteData, and update which perform various operations on vehicle data.

The Car, Bike, and Bus classes are derived from the Vehicle class and each has data members carCounter, bikeCounter, and busCounter, and carCharge, bikeCharge, and busCharge, respectively. These classes also have member functions getCarCounter, getCarCost, getBikeCounter, getBikeCost, getBusCounter, and getBusCost, which return the respective data members.

The main function provides a menu for the user to perform various operations on vehicle data, such as adding a new vehicle, searching for a vehicle, deleting a vehicle, displaying all vehicles, updating a vehicle, and exiting the program. It uses objects of the derived classes Car, Bike, and Bus to call the member functions of the Vehicle class to perform these operations. The program also uses file streams ofstream, ifstream, and fstream to read from and write to a file called car.txt.