COMPUTER SCIENCE

PROJECT



CAR RENTAL SYSTEM

Midhun Sushil

XII-J ( 2017-18 )

THE INDIAN SCHOOL BAHRAIN

TABLE OF CONTENTS

1. Certificate
2. Acknowledgement
3. Introduction
4. About C++
5. System Analysis
6. System Design
7. Sample Outputs
8. Source Code
9. Bibliography

Certificate

This is to certify that **MIDHUN .P. SUSHIL,** student of **XII-J, INDIAN SCHOOL BAHRAIN** has successfully completed the project based on the topic **“SOFTWARE FOR CAR RENTAL MANAGEMENT SYSTEM”**

in fulfilment of Computer Science Practical Examination conducted by AISSCE during the academic year **2017-2018**.

\_\_\_\_\_\_\_\_\_\_\_\_\_ \_\_\_\_\_\_\_\_\_\_\_\_\_

Signature of Signature of

Subject Teacher Examiner

Acknowledgement

In accomplishment of this project successfully, many people have best owned me their blessings and heart pledged support, this time I am utilizing to thank all the people who have been concerned with the project.

I would like to express my gratitude towards our

Computer Science Teacher Mrs. Smitha John and the

HOD of Computer Science Mr. Pius Mathew, whose valuable guidance has been the ones that helped me patch this project and make it full proof success.

Last but not the least I wish to thank my teammates Milan Mani Mathew, Joel George, Mervin D’Costa and Adarsh Chandran who have supported us in various phases of the completion of the project.

Introduction

The mini project **‘Car Rental Management System’** is the sample project for managing car rental. The project aims at developing car rental system using the C++ language that enables an organization to maintain its cars and customers.

The project demonstrates the creation of a user interface of a system. The application uses basic C++ function to generate login screen, menus, show boxes and print text on the screen. It also effectively applies the various C++ concepts such as file operations, looping and branching constructs and string manipulation functions.

It uses DOS file support as back end to save the records.

The application also implements the concept of classes to define the car and customer records and the concept of encapsulation to make all data private or protected, and to make public only those functions that are part of minimal interface for users of the class. This can hide the details of data implementation, allowing the developer to later change the implementation without changing the interface in any way.

About C ++

Loops in c++

Loops are used to repeat a block of code. Being able to have your program repeatedly execute a block of code is one of the most basic but useful tasks in programming.

**FOR** - *FOR* loops are the most useful type. The syntax is:

for ( variable initialization; condition; variable update ) {

Code to execute while the condition is true

}

**WHILE** - *WHILE* loops are very simple. The basic structure is:

while ( condition )

{ Code to execute while the condition is true }

The true represents a boolean expression which could be x == 1 or while ( x != 7 ) (x does not equal 7). It can be any combination of Boolean statements that are legal. Even, (while x ==5 || v == 7) which says execute the code while x equals five or while v equals 7.

DO.WHILE - *DO.WHILE*  loops are useful for things that want to loop at least once. The structure is:

do {

} while ( condition );

Loop control statements

Jump statements allow altering the flow of a program by performing jumps to specific locations.

**The break statement**

break leaves a loop, even if the condition for its end is not fulfilled. It can be used to end an infinite loop, or to force it to end before its natural end.

**The continue statement**

The continue statement causes the program to skip the rest of the loop in the current iteration, as if the end of the statement block had been reached, causing it to jump to the start of the following iteration.

**The goto statement**

goto allows to make an absolute jump to another point in the program. This unconditional jump ignores nesting levels, and does not cause any automatic stack unwinding. Therefore, it is a feature to use with care, and preferably within the same block of statements, especially in the presence of local variables.

## **Infinite Loop**

There may exist some loops which can iterate or occur infinitely. These are called Infinite Loop. These loops occur infinitely because their condition is always true.

CLASSES

A **class** in **C++** is a user defined type or data structure declared with keyword *class* that has data and functions (also called methods) as its members whose access is governed by the three access specifiers *private*, *protected* or *public* (by default access to members of a class is *private*). The private members are not accessible outside the class; they can be accessed only through methods of the class. The public members form an interface to the class and are accessible outside the class.

Instances of a class data type are known as [objects](https://en.wikipedia.org/wiki/Object_(computer_science)) and can contain [member variables](https://en.wikipedia.org/wiki/Variable_(programming)), [constants](https://en.wikipedia.org/wiki/Variable_(programming)), [member functions](https://en.wikipedia.org/wiki/Method_(computer_science)), and [overloaded operators](https://en.wikipedia.org/wiki/Operator_overloading) defined by the programmer.

An important feature of the C++ class and structure are **member functions**. Each datatype can have its own built-in functions (referred to as methods) that have access to all (public and private) members of the datatype. In the body of these non-static member functions, the keyword [this](https://en.wikipedia.org/wiki/This_(computer_science)) can be used to refer to the object for which the function is called. This is commonly implemented by passing the address of the object as an implicit first argument to the function.

#### Encapsulation

[Encapsulation](https://en.wikipedia.org/wiki/Information_hiding) is the hiding of information to ensure that data structures and operators are used as intended and to make the usage model more obvious to the developer. C++ provides the ability to define classes and functions as its primary encapsulation mechanisms. Within a class, members can be declared as either public, protected, or private to explicitly enforce encapsulation. A public member of the class is accessible to any function. A private member is accessible only to functions that are members of that class and to functions and classes explicitly granted access permission by the class ("friends"). A protected member is accessible to members of classes that inherit from the class in addition to the class itself and any friends.

#### Inheritance

[Inheritance](https://en.wikipedia.org/wiki/Inheritance_(computer_science)) allows one data type to acquire properties of other data types. Inheritance from a [base class](https://en.wikipedia.org/wiki/Base_class) may be declared as public, protected, or private. This access specifier determines whether unrelated and derived classes can access the inherited public and protected members of the base class. Only public inheritance corresponds to what is usually meant by "inheritance". The other two forms are much less frequently used. If the access specifier is omitted, a "class" inherits privately, while a "struct" inherits publicly. Base classes may be declared as virtual; this is called [virtual inheritance](https://en.wikipedia.org/wiki/Virtual_inheritance). Virtual inheritance ensures that only one instance of a base class exists in the inheritance graph, avoiding some of the ambiguity problems of multiple inheritance.

[Multiple inheritance](https://en.wikipedia.org/wiki/Multiple_inheritance) is a C++ feature not found in most other languages, allowing a class to be derived from more than one base class; this allows for more elaborate inheritance relationships. For example, a "Flying Cat" class can inherit from both "Cat" and "Flying Mammal". Some other languages, such as [C#](https://en.wikipedia.org/wiki/C_Sharp_(programming_language)) or [Java](https://en.wikipedia.org/wiki/Java_(programming_language)), accomplish something similar (although more limited) by allowing inheritance of multiple [interfaces](https://en.wikipedia.org/wiki/Interface_(computer_science)) while restricting the number of base classes to one (interfaces, unlike classes, provide only declarations of member functions, no implementation or member data). An interface as in C# and Java can be defined in C++ as a class containing only pure virtual functions, often known as an [abstract base class](https://en.wikipedia.org/wiki/Abstract_base_class) or "ABC". The member functions of such an abstract base class are normally explicitly defined in the derived class, not inherited implicitly. C++ virtual inheritance exhibits an ambiguity resolution feature called [dominance](https://en.wikipedia.org/wiki/Dominance_(C%2B%2B)).

Polymorphism

[Polymorphism](https://en.wikipedia.org/wiki/Type_polymorphism) enables one common interface for many implementations, and for objects to act differently under different circumstances: A **polymorphic type** is one whose operations can also be applied to values of some other type, or types. There are several fundamentally different kinds of polymorphism:

* [*Ad hoc polymorphism*](https://en.wikipedia.org/wiki/Ad_hoc_polymorphism): when a function denotes different and potentially heterogeneous implementations depending on a limited range of individually specified types and combinations. Ad hoc polymorphism is supported in many languages using [function overloading](https://en.wikipedia.org/wiki/Function_overloading).
* [*Parametric polymorphism*](https://en.wikipedia.org/wiki/Parametric_polymorphism): when code is written without mention of any specific type and thus can be used transparently with any number of new types. In the [object-oriented programming](https://en.wikipedia.org/wiki/Object-oriented_programming) community, this is often known as *generics* or [*generic programming*](https://en.wikipedia.org/wiki/Generic_programming). In the [functional programming](https://en.wikipedia.org/wiki/Functional_programming) community, this is often shortened to *polymorphism*.
* [*Subtyping*](https://en.wikipedia.org/wiki/Subtyping) (also called *subtype polymorphism* or *inclusion polymorphism*): when a name denotes instances of many different classes related by some common superclass. In the object-oriented programming community, this is often referred to as simply *Inheritance*

Pointers

In [computer science](https://en.wikipedia.org/wiki/Computer_science), a **pointer** is a [programming language](https://en.wikipedia.org/wiki/Programming_language) object, whose value refers to (or "**points** to") another value stored elsewhere in the [computer memory](https://en.wikipedia.org/wiki/Computer_memory) using its [memory address](https://en.wikipedia.org/wiki/Memory_address). A pointer *references* a location in memory, and obtaining the value stored at that location is known as [*dereferencing*](https://en.wikipedia.org/wiki/Dereference_operator) the pointer. As an analogy, a page number in a book's index could be considered a pointer to the corresponding page; dereferencing such a pointer would be done by flipping to the page with the given page number and reading the text found on the indexed page.

Pointers to data significantly improve performance for repetitive operations such as traversing [strings](https://en.wikipedia.org/wiki/String_(computer_science)#String_processing_algorithms), [lookup tables](https://en.wikipedia.org/wiki/Lookup_table), [control tables](https://en.wikipedia.org/wiki/Control_table) and [tree](https://en.wikipedia.org/wiki/Tree_(data_structure))structures. In particular, it is often much cheaper in time and space to copy and dereference pointers than it is to copy and access the data to which the pointers point.

File Handling

Files are a means to store data in a storage device. C++ file handling provides a mechanism to store output of a program in a file and read from a file on the disk. So far, we have been using <iostream> header file which provide functions cin and cout to take input from console and write output to a console respectively. Now, we introduce one more header file <fstream> which provides data types or classes ( ifstream , ofstream , fstream ) to read from a file and write to a file.

**File Opening Modes**  
A file can be opened in different modes to perform read and write operations. Function to open a file i.e **open( )**takes two arguments : **char \*filename** and **ios :: mode**. C++ supports the following file open modes :

|  |  |
| --- | --- |
| **Mode** | **Explanation** |
| ios :: in | Open a file for reading |
| ios :: out | Open a file for writing |
| ios :: app | Appends data to the end of the file |
| ios :: ate | File pointer moves to the end of the file but allows to writes data in any location in the file |
| ios :: binary | Binary File |
| ios :: trunc | Deletes the contents of the file before opening |

SYSTEM ANALYSIS

The program is user friendly and coding has been done with proper efficiency. The software is made as an administrator operated programs. So, the features of this program can be used by the customers only with the supervision of the administrator.

The Structure of the program:

*Login Screen*

The admin must enter the username and password to login.

*Main Menu*

The Main Menu contains 8 options where the option number has to be entered to select the option.

The options are:

*1. Rent A Car:* The customer has to enter the required details along with from and to dates of rent period. After entering the details, the inventory is displayed, and the customer has to choose an available car by entering the serial number. Then the contract is displayed with the options Save and Cancel.

*2. Add Car:* The administrator is asked about the number of cars to be added and necessary details of the cars to be added will asked as many times.

*3. Search and Display:* On opening this option 5 more options are displayed.

They are:

1. *Display All Cars:* In this option all the details of the cars available in the system are displayed one by one.
2. *Display All Customers:* Here the contract of each customer is displayed one by one.
3. *Search Car:* In this option any one field among the asked options must be entered to search for a car.
4. *Search Customer:* Here any one field among the asked fields must be entered and the contract of that customer will be displayed.
5. *Go Back:* To return to the main menu.

*4. Reports:* On entering this function

The user is asked to enter the one among the following options:

1. *View All Car reports:* Where the inventory is displayed.
2. *View Customer Report:* As the name suggests.
3. *View Customer-Car Report:* Here the details of the customers are displayed in a tabular form with only relevant details.
4. *Go Back:* To return to the main menu.

*5. Edit Record:* This includes editing of all the record that the system handles with. They are:

1. *Update Car:* To Update car details
2. *Update Customer:* To update contract.
3. *Delete Car:* To delete a car by entering any one among the asked details.
4. *Delete Customer:* To delete the contract pf any customer.
5. *Edit Status:* This has further options like.
6. Change all status to Available.
7. Change all status to Booked
8. Switch Status.
9. Go Back: To return to Edit Record menu .
10. *Go Back:* To return to the main menu.

*6. Return and Billing:* This option is to be used when the customer returns the car.

Any one among the required field must be entered to view the contract.

Then on proceeding to the bill the current date is asked. The billing works in this manner- If it matches with the earlier mentioned to date in the contract the customer would not be fined. Whereas if it’s a late return the 150% of the rent per day will be fined for each day.

But as the dates in the contract are fixed early return will not be refunde.

*7. About:* About our company

*8. Exit*.

*Header Files Used:*

<fstream.h> <iomanip.h> <stdio.h> <string.h> <stdlib.h> <conio.h> <ctype.h>

*Special User Defined Functions Used:*

* 1. ventry():

For validated input of a string of specific size (N) which can be passed as argument. This prevents users from inputting invalid characters such as inputting alphabets in a string that stores phone number.

Usage: ventry(string,code,size); //by default size=1

code=0 >> for inputting alphabets(a-Z), space, comma(,) & dot(.)

code=1 >> for inputting numbers(0-9), dot(.) & '/'

code=2 >> for inputting username / ID

code=3 >> for normal input without space, ':' & ';'

code=4 >> for menu input (only numbers 0-9)

* 1. clear():

For clearing specific character, characters or lines

This function basically uses gotoxy() and outputs space over to clear the specified no. of characters or lines. It's more convenient than the library function clrscr() (clears entire console output) as it can clear specified locations.

Usage: clear(x, y, noc, nol ). //by default nol=1

(x,y) >> Margin/position of first character;

noc >> number of characters to clear;

nol >> number of lines to clear;

* 1. border():

For creating border of length l(horizontal) & width w(vertical).

usage: border(x, y, l, w, type)

Where (p,q)-margin (coordinate of top-left vertices of box border);

type >> 0(+ \* !);

type >> 1(single border), 4(slow-motion)(\* ! +);

type >> 2(double character border)(/);

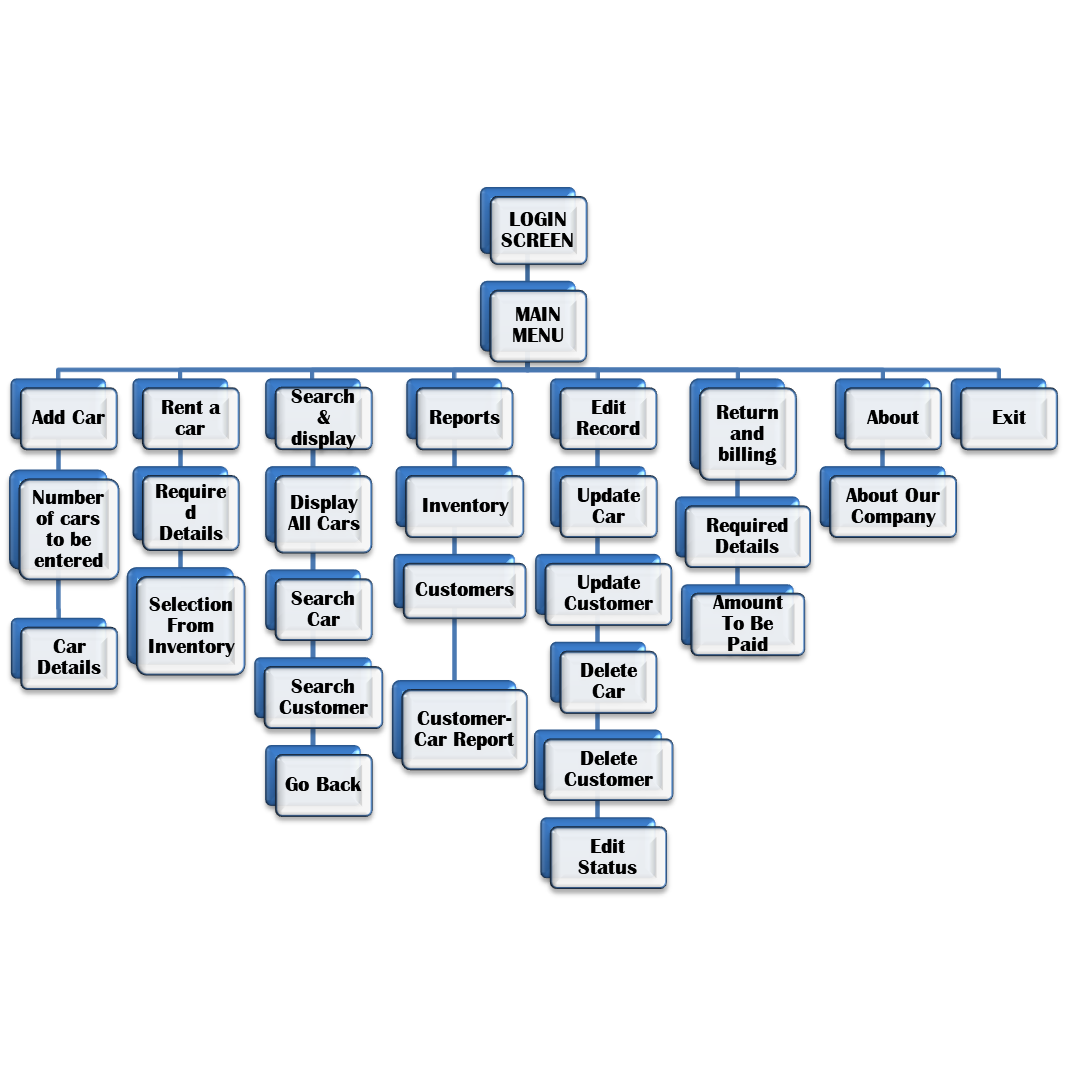
* 1. namecmp():

For checking for a word in a string; Used in the program to find name inside fullname; If string size less than word size function returns (-1);

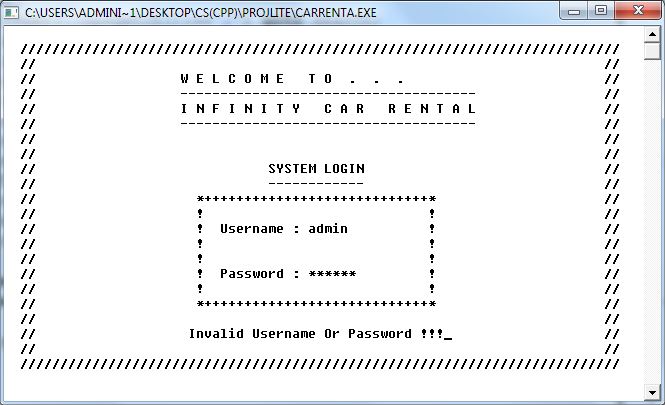
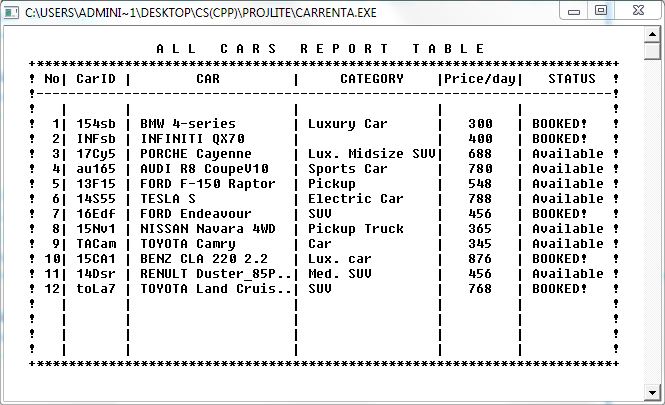
If the entire word is in the string, function returns (1) else returns (0).

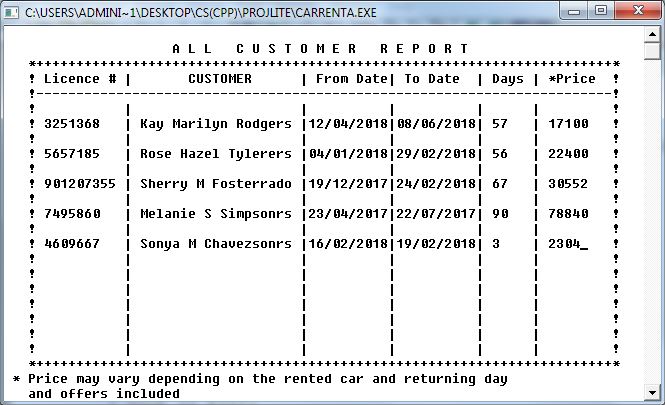
Usage: namecmp(string, word).

SYSTEM DESIGN

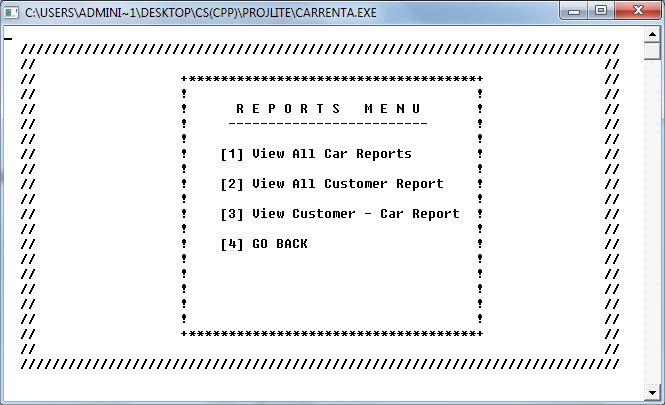


SCREENSHOTS

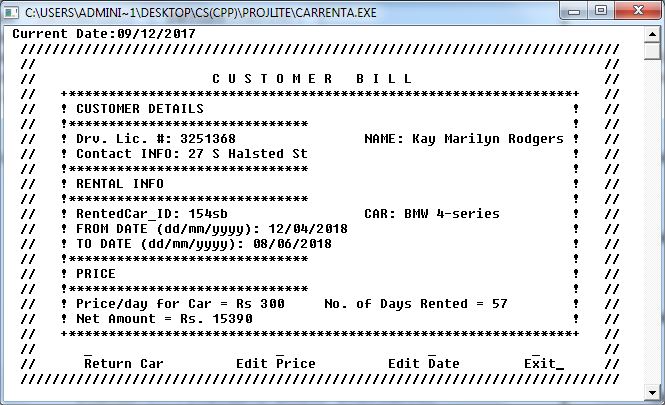
1. ***Login Screen***
2. ***Main Menu***
3. ***Car Report Table***
4. ***Customer Report***



1. ***Report Menu***

******

1. ***View Bill***

******

|  |  |
| --- | --- |
| 1  2  3  4  5  6  7  8  9  10  11  12  13  14  15  16  17  18  19  20  21  22  23  24  25  26  27  28  29  30  31  32  33  34  35  36  37  38  39  40  41  42  43  44  45  46  47  48  49  50  51  52  53  54  55  56  57  58  59  60  61  62  63  64  65  66  67  68  69  70  71  72  73  74  75  76  77  78  79  80  81  82  83  84  85  86  87  88  89  90  91  92  93  94  95  96  97  98  99  100  101  102  103  104  105  106  107  108  109  110  111  112  113  114  115  116  117  118  119  120  121  122  123  124  125  126  127  128  129  130  131  132  133  134  135  136  137  138  139  140  141  142  143  144  145  146  147  148  149  150  151  152  153  154  155  156  157  158  159  160  161  162  163  164  165  166  167  168  169  170  171  172  173  174  175  176  177  178  179  180  181  182  183  184  185  186  187  188  189  190  191  192  193  194  195  196  197  198  199  200  201  202  203  204  205  206  207  208  209  210  211  212  213  214  215  216  217  218  219  220  221  222  223  224  225  226  227  228  229  230  231  232  233  234  235  236  237  238  239  240  241  242  243  244  245  246  247  248  249  250  251  252  253  254  255  256  257  258  259  260  261  262  263  264  265  266  267  268  269  270  271  272  273  274  275  276  277  278  279  280  281  282  283  284  285  286  287  288  289  290  291  292  293  294  295  296  297  298  299  300  301  302  303  304  305  306  307  308  309  310  311  312  313  314  315  316  317  318  319  320  321  322  323  324  325  326  327  328  329  330  331  332  333  334  335  336  337  338  339  340  341  342  343  344  345  346  347  348  349  350  351  352  353  354  355  356  357  358  359  360  361  362  363  364  365  366  367  368  369  370  371  372  373  374  375  376  377  378  379  380  381  382  383  384  385  386  387  388  389  390  391  392  393  394  395  396  397  398  399  400  401  402  403  404  405  406  407  408  409  410  411  412  413  414  415  416  417  418  419  420  421  422  423  424  425  426  427  428  429  430  431  432  433  434  435  436  437  438  439  440  441  442  443  444  445  446  447  448  449  450  451  452  453  454  455  456  457  458  459  460  461  462  463  464  465  466  467  468  469  470  471  472  473  474  475  476  477  478  479  480  481  482  483  484  485  486  487  488  489  490  491  492  493  494  495  496  497  498  499  500  501  502  503  504  505  506  507  508  509  510  511  512  513  514  515  516  517  518  519  520  521  522  523  524  525  526  527  528  529  530  531  532  533  534  535  536  537  538  539  540  541  542  543  544  545  546  547  548  549  550  551  552  553  554  555  556  557  558  559  560  561  562  563  564  565  566  567  568  569  570  571  572  573  574  575  576  577  578  579  580  581  582  583  584  585  586  587  588  589  590  591  592  593  594  595  596  597  598  599  600  601  602  603  604  605  606  607  608  609  610  611  612  613  614  615  616  617  618  619  620  621  622  623  624  625  626  627  628  629  630  631  632  633  634  635  636  637  638  639  640  641  642  643  644  645  646  647  648  649  650  651  652  653  654  655  656  657  658  659  660  661  662  663  664  665  666  667  668  669  670  671  672  673  674  675  676  677  678  679  680  681  682  683  684  685  686  687  688  689  690  691  692  693  694  695  696  697  698  699  700  701  702  703  704  705  706  707  708  709  710  711  712  713  714  715  716  717  718  719  720  721  722  723  724  725  726  727  728  729  730  731  732  733  734  735  736  737  738  739  740  741  742  743  744  745  746  747  748  749  750  751  752  753  754  755  756  757  758  759  760  761  762  763  764  765  766  767  768  769  770  771  772  773  774  775  776  777  778  779  780  781  782  783  784  785  786  787  788  789  790  791  792  793  794  795  796  797  798  799  800  801  802  803  804  805  806  807  808  809  810  811  812  813  814  815  816  817  818  819  820  821  822  823  824  825  826  827  828  829  830  831  832  833  834  835  836  837  838  839  840  841  842  843  844  845  846  847  848  849  850  851  852  853  854  855  856  857  858  859  860  861  862  863  864  865  866  867  868  869  870  871  872  873  874  875  876  877  878  879  880  881  882  883  884  885  886  887  888  889  890  891  892  893  894  895  896  897  898  899  900  901  902  903  904  905  906  907  908  909  910  911  912  913  914  915  916  917  918  919  920  921  922  923  924  925  926  927  928  929  930  931  932  933  934  935  936  937  938  939  940  941  942  943  944  945  946  947  948  949  950  951  952  953  954  955  956  957  958  959  960  961  962  963  964  965  966  967  968  969  970  971  972  973  974  975  976  977  978  979  980  981  982  983  984  985  986  987  988  989  990  991  992  993  994  995  996  997  998  999  1000  1001  1002  1003  1004  1005  1006  1007  1008  1009  1010  1011  1012  1013  1014  1015  1016  1017  1018  1019  1020  1021  1022  1023  1024  1025  1026  1027  1028  1029  1030  1031  1032  1033  1034  1035  1036  1037  1038  1039  1040  1041  1042  1043  1044  1045  1046  1047  1048  1049  1050  1051  1052  1053  1054  1055  1056  1057  1058  1059  1060  1061  1062  1063  1064  1065  1066  1067  1068  1069  1070  1071  1072  1073  1074  1075  1076  1077  1078  1079  1080  1081  1082  1083  1084  1085  1086  1087  1088  1089  1090  1091  1092  1093  1094  1095  1096  1097  1098  1099  1100  1101  1102  1103  1104  1105  1106  1107  1108  1109  1110  1111  1112  1113  1114  1115  1116  1117  1118  1119  1120  1121  1122  1123  1124  1125  1126  1127  1128  1129  1130  1131  1132  1133  1134  1135  1136  1137  1138  1139  1140  1141  1142  1143  1144  1145  1146  1147  1148  1149  1150  1151  1152  1153  1154  1155  1156  1157  1158  1159  1160  1161  1162  1163  1164  1165  1166  1167  1168  1169  1170  1171  1172  1173  1174  1175  1176  1177  1178  1179  1180  1181  1182  1183  1184  1185  1186  1187  1188  1189  1190  1191  1192  1193  1194  1195  1196  1197  1198  1199  1200  1201  1202  1203  1204  1205  1206  1207  1208  1209  1210  1211  1212  1213  1214  1215  1216  1217  1218  1219  1220  1221  1222  1223  1224  1225  1226  1227  1228  1229  1230  1231  1232  1233  1234  1235  1236  1237  1238  1239  1240  1241  1242  1243  1244  1245  1246  1247  1248  1249  1250  1251  1252  1253  1254  1255  1256  1257  1258  1259  1260  1261  1262  1263  1264  1265  1266  1267  1268  1269  1270  1271  1272  1273  1274  1275  1276  1277  1278  1279  1280  1281  1282  1283  1284  1285  1286  1287  1288  1289  1290  1291  1292  1293  1294  1295  1296  1297  1298  1299  1300  1301  1302  1303  1304  1305  1306  1307  1308  1309  1310  1311  1312  1313  1314  1315  1316  1317  1318  1319  1320  1321  1322  1323  1324  1325  1326  1327  1328  1329  1330  1331  1332  1333  1334  1335  1336  1337  1338  1339  1340  1341  1342  1343  1344  1345  1346  1347  1348  1349  1350  1351  1352  1353  1354  1355  1356  1357  1358  1359  1360  1361  1362  1363  1364  1365  1366  1367  1368  1369  1370  1371  1372  1373  1374  1375  1376  1377  1378  1379  1380  1381  1382  1383  1384  1385  1386  1387  1388  1389  1390  1391  1392  1393  1394  1395  1396  1397  1398  1399  1400  1401  1402  1403  1404  1405  1406  1407  1408  1409  1410  1411  1412  1413  1414  1415  1416  1417  1418  1419  1420  1421  1422  1423  1424  1425  1426  1427  1428  1429  1430  1431  1432  1433  1434  1435  1436  1437  1438  1439  1440  1441  1442  1443  1444  1445  1446  1447  1448  1449  1450  1451  1452  1453  1454  1455  1456  1457  1458  1459  1460  1461  1462  1463  1464  1465  1466  1467  1468  1469  1470  1471  1472  1473  1474  1475  1476  1477  1478  1479  1480  1481  1482  1483  1484  1485  1486  1487  1488  1489  1490  1491  1492  1493  1494  1495  1496  1497  1498  1499  1500  1501  1502  1503  1504  1505  1506  1507  1508  1509  1510  1511  1512  1513  1514  1515  1516  1517  1518  1519  1520  1521  1522  1523  1524  1525  1526  1527  1528  1529  1530  1531  1532  1533  1534  1535  1536  1537  1538  1539  1540  1541  1542  1543  1544  1545  1546  1547  1548  1549  1550  1551  1552  1553  1554  1555  1556  1557  1558  1559  1560  1561  1562  1563  1564  1565  1566  1567  1568  1569  1570  1571  1572  1573  1574  1575  1576  1577  1578  1579  1580  1581  1582  1583  1584  1585  1586  1587  1588  1589  1590  1591  1592  1593  1594  1595  1596  1597  1598  1599  1600  1601  1602  1603  1604  1605  1606  1607  1608  1609  1610  1611  1612  1613  1614  1615  1616  1617  1618  1619  1620  1621  1622  1623  1624  1625  1626  1627  1628  1629  1630  1631  1632  1633  1634  1635  1636  1637  1638  1639  1640  1641  1642  1643  1644  1645  1646  1647  1648  1649  1650  1651  1652  1653  1654  1655  1656  1657  1658  1659  1660  1661  1662  1663  1664  1665  1666  1667  1668  1669  1670  1671  1672  1673  1674  1675  1676  1677  1678  1679  1680  1681  1682  1683  1684  1685  1686  1687  1688  1689  1690  1691  1692  1693  1694  1695  1696  1697  1698  1699  1700  1701  1702  1703  1704  1705  1706  1707  1708  1709  1710  1711  1712  1713  1714  1715  1716  1717  1718  1719  1720  1721  1722  1723  1724  1725  1726  1727  1728  1729  1730  1731  1732  1733  1734  1735  1736  1737  1738  1739  1740  1741  1742  1743  1744  1745  1746  1747  1748  1749  1750  1751  1752  1753  1754  1755  1756  1757  1758  1759  1760  1761  1762  1763  1764  1765  1766  1767  1768  1769  1770  1771  1772  1773  1774  1775  1776  1777  1778  1779  1780  1781  1782  1783  1784  1785  1786  1787  1788  1789  1790  1791  1792  1793  1794  1795  1796  1797  1798  1799  1800  1801  1802  1803  1804  1805  1806  1807  1808  1809  1810  1811  1812  1813  1814  1815  1816  1817  1818  1819  1820  1821  1822  1823  1824  1825  1826  1827  1828  1829  1830  1831  1832  1833  1834  1835  1836  1837  1838  1839  1840  1841  1842  1843  1844  1845  1846  1847  1848  1849  1850  1851  1852  1853  1854  1855  1856  1857  1858  1859  1860  1861  1862  1863  1864  1865  1866  1867  1868  1869  1870  1871  1872  1873  1874  1875  1876  1877  1878  1879  1880  1881  1882  1883  1884  1885  1886  1887  1888  1889  1890  1891  1892  1893  1894  1895  1896  1897  1898  1899  1900  1901  1902  1903  1904  1905  1906  1907  1908  1909  1910  1911  1912  1913  1914  1915  1916  1917  1918  1919  1920  1921  1922  1923  1924  1925  1926  1927  1928  1929  1930  1931  1932  1933  1934  1935  1936  1937  1938  1939  1940  1941  1942  1943  1944  1945  1946  1947  1948  1949  1950  1951  1952  1953  1954  1955  1956  1957  1958  1959  1960  1961  1962  1963  1964  1965  1966  1967  1968  1969  1970  1971  1972  1973  1974  1975  1976  1977  1978  1979  1980  1981  1982  1983  1984  1985  1986  1987  1988  1989  1990  1991  1992  1993  1994  1995  1996  1997  1998  1999  2000  2001 | #include <fstream.h>  #include <iomanip.h>  #include <stdio.h>  #include <stdlib.h>  #include <conio.h>  #include <string.h>  #include <ctype.h>  char op;  void menu();  void delay() {  **for** (unsigned int i = 0; i < 5000; i++)  **for** (unsigned int j = 0; j < 9000; j++) { */\*To delay output\*/*  }  }  void loading(short x, short y, short N = 7)  */\* (x,y)-Position of Load Animation*  *Loading Running delay(N=1 for 0.4786 s delay approx.)*  *\*/*  {  **const** char slash[5] = {92, 45, 47, 124};  **for** (unsigned short i = 0; i < N; i++)  **for** (short j = 0; j < 4; j++) {  gotoxy(x, y);  cout << slash[j];  delay();  delay();  }  }  void clear(short x, short y, short n, short l = 1)  */\* (x,y)-margin; n - no. of characters to clear*  *l - no. of lines to clear*  *\*/*  {  **for** (short i = 0; i < l; i++)  **for** (short j = 0; j < n; j++) {  gotoxy(x + j, y + i);  cout << " ";  }  }  void line(int p, int q, int l, char type = 'v', int type2 = 0) {  int i;  char hl, vl;  */\* Where (p,q)-margin (coordinate of top-left vertice of box border);*  *type=v - For Vertical Line (default)*  *type=h - For Horizontal Line*  *l-length(horizontal distance) of border;*  *\*/*  **if** (type2 == 1) {  hl = '\*';  vl = '!';  } **else** {  hl = 45;  vl = 124;  } *// hyphen(-)and or (|)*  *//(WHITE);*  **if** (type == 'v')  **for** (i = 0; i < l; i++) *// Vertical line of length l*  {  gotoxy(p, q + i);  printf(  "%c",  vl); *//'7C'(hyphen); 'DE'(accented i); '7C' (or line); 'B3' (border);*  }  **else**  **for** (i = 0; i < l; i++) *// Horizontal Line of length l*  {  gotoxy(p + i, q);  printf("%c", hl);  }  }  void border(short p, short q, short l, short w,  short type = 0) *// Use to make borders as the name suggests*  { */\* Where (p,q)-margin (coordinate of top-left vertice of box border);*  *w-width(vertical distance) of border;*  *l-length(horizontal distance) of border;*  *type => 2(double border); type => 0,1(single border); type =>*  *3(slowmotion)*  *\*/*  char cc, hl, vl;  **if** (type == 2) {  cc = '/';  hl = '/';  vl = '/';  } **else** **if** (type == 1) {  cc = '\*';  hl = '+';  vl = '!';  } **else** **if** (type == 4 || type == 0) {  cc = '+';  hl = '\*';  vl = '!';  } **else** {  cc = '\*';  hl = '\*';  vl = '\*';  }  gotoxy(p, q);  printf("%c", cc); *// top-left*  gotoxy(p + l, q);  printf("%c", cc); *// top-right*  gotoxy(p, q + w);  printf("%c", cc); *// bottom-left*  gotoxy(p + l, q + w);  printf("%c", cc); *// bottom-right*  **for** (int i = 1, j = 1, m = (w - 1), n = (l - 1);; i++, j++, m--, n--) {  **if** (i < w) {  gotoxy(p, q + i);  **if** (type == 2) cout << vl;  printf("%c", vl);  }  **if** (j < l) {  gotoxy(p + i, q);  printf("%c", hl);  **if** (type == 2) cout << hl;  }  **if** (type == 4) {  delay();  } *// for delay*  **if** (m > 0) {  gotoxy(p + l, q + m);  **if** (type == 2) cout << '\b' << vl;  printf("%c", vl);  }  **if** (n > 0) {  gotoxy(p + n, q + w);  printf("%c", hl);  **if** (type == 2) cout << hl;  }  **if** (type == 4) {  delay();  } *// for delay*  **if** (j >= l && i >= w) **break**;  }  }  void cursor() *// To get position of pointer in console window(for output*  *// purpose)*  {  int x = 1, y = 1;  char key;  **do** {  gotoxy(x, y);  cout << " ";  gotoxy(x, y);  key = getch();  **if** (key == ' ')  x++;  **else** **if** (key == '\b')  x--;  **else** **if** (key == '\r')  y++;  **else** if (key == 'p') {  gotoxy(1, 1);  cout << x << " " << y;  } **else** if (key == 'r') {  x = 1;  y = 1;  }  } **while** (key != 'N');  }  void ventry(char t[], short code = 4, short N = 1)  */\* code=0 >> for inputing alphabets(a-Z), space, comma(,) & dot(.)*  *code=1 >> for inputing numbers(0-9), dot(.) & '/'*  *code=2 >> for inputing username / ID*  *code=3 >> for normal input without space, ':' & ';'*  *code=4 >> for menu input (only numbers 0-9)*  *\*/*  {  unsigned short i = 0;  **if** (code == 0) {  **while** (i < N && (t[i] = getch()) != '\r')  **if** ((t[i] >= 97 && t[i] <= 122) || (t[i] >= 65 && t[i] <= 90) ||  t[i] == 32 || t[i] == '.' || t[i] == ',') {  **if** (t[0] != ' ' &&  t[0] != '\_') *// To prevent space or '\_' at first input*  {  cout << t[i];  i++;  }  } **else** **if** (t[i] == 8 && i > 0) {  cout << "**\b** **\b**";  i--;  }  } **else** **if** (code == 1) {  **while** (i < N && (t[i] = getch()) != '\r')  **if** ((t[i] >= 48 && t[i] <= 57) || t[i] == 46 || t[i] == '/') {  cout << t[i];  i++;  } **else** **if** (t[i] == 8 && i > 0) {  cout << "**\b** **\b**";  i--;  }  }  **if** (code == 2) {  **while** (i < N && (t[i] = getch()) != '\r')  **if** ((t[i] >= 95 && t[i] <= 122) || (t[i] >= 46 && t[i] <= 57) ||  (t[i] >= 65 && t[i] <= 90) || t[i] == '@') {  cout << t[i];  i++;  } **else** **if** (t[i] == 8 && i > 0) {  cout << "**\b** **\b**";  i--;  }  }  **if** (code == 3) {  **while** (i < N && (t[i] = getch()) != '\r')  **if** ((t[i] >= 33 && t[i] <= 126) && t[i] != ':' && t[i] != ';') {  cout << t[i];  i++;  } **else** **if** (t[i] == 8 && i > 0) {  cout << "**\b** **\b**";  i--;  }  } **else** **if** (code == 4) {  **while** (i < N && (t[i] = getch()) != '\r')  **if** (t[i] >= 48 &&  t[i] <= 57) { *// cout<<t[i]; //Uncomment to display input no.*  i++;  } **else** **if** (t[i] == 8 && i > 0) {  cout << "**\b** **\b**";  i--;  }  }  t[i] = '\0';  }  void voutput(short N, char t[], char o[] = NULL)  */\* N >> for outputing N characters of the concatinated string*  *\*/*  {  unsigned short i = 0;  strcat(t, " ");  strcat(t, o);  **if** (N < strlen(t)) {  **for** (i = N; i < N + 3; i++) t[i] = '.';  strcat(t, "..");  N += 2;  }  **for** (i = 0; i < N && i < strlen(t); i++) cout << t[i];  }  void About() {  border(3, 2, 74, 22, 2);  gotoxy(17, 4);  cout << "A B O U T I N F I N I T Y C A R R E N T A L";  gotoxy(17, 5);  cout << "-------------------------------------------------";  gotoxy(11, 6);  cout << "Our company started in 2001 in a small town in Joerdhailun";  gotoxy(14, 8);  cout << "which has now expanded globally having locations in";  gotoxy(12, 10);  cout << "Asia, Europe, North America. We have received numerous";  gotoxy(12, 12);  cout << "accolades for our service and the quality of our cars.";  gotoxy(15, 14);  cout << "We strive to provide our clients with quality our";  gotoxy(17, 16);  cout << "clients with quality cars at affordable prices.";  gotoxy(13, 18);  cout << "We have a wide selection of cars including high range";  gotoxy(18, 20);  cout << "of luxury cars like Rolls Royce, Bentley";  gotoxy(22, 22);  cout << "and even a few exotic sports cars. ";  }  **class** **Login** {  char usr[13];  char pss[13];  public:  int check();  } L;  int Login::check() {  xx:  clrscr();  border(3, 2, 74, 21, 2);  int i = 0;  gotoxy(23, 4);  printf("W E L C O M E T O . . .");  gotoxy(23, 5);  cout << "-------------------------------------";  gotoxy(23, 6);  cout << "I N F I N I T Y C A R R E N T A L";  gotoxy(23, 7);  printf("-------------------------------------");  gotoxy(34, 10);  printf("SYSTEM LOGIN");  gotoxy(34, 11);  printf("------------");  border(25, 12, 29, 7, 1);  gotoxy(28, 14);  cout << "Username : ";  gotoxy(28, 17);  cout << "Password : ";  gotoxy(39, 14);  ventry(usr, 2, 13);  gotoxy(39, 17);  **while** (i < 12 && (pss[i] = getch()) != '\r') {  **if** (pss[i] == '\b' && i > 0) {  cout << '\b' << ' ' << '\b';  i--;  } **else** **if** (i != 0 || pss[i] != '\b') {  cout << "\*";  i++;  }  }  pss[i] = '\0';  **if** (strcmp(usr, "admin") == 0 && strcmp(pss, "pass123") == 0) {  cout << '\a';  clrscr();  **return** (1);  } **else** {  gotoxy(24, 21);  printf("Invalid Username Or Password !!!");  getch();  **goto** xx;  }  }  short namecmp(char \*name, char \*word)  *// To search for a 'word' in a 'name'*  {  **if** (strlen(name) < strlen(word)) {  clrscr();  cout << "Error exit function! String size less than word size.... ";  **return** (-1);  }  short sc, count = 0;  **for** (short i = 0; i <= (strlen(name) - strlen(word)); i++) {  sc = 0;  **for** (short j = 0; j < strlen(word); j++)  **if** (tolower(name[i + j]) != tolower(word[j])) {  sc = 1;  **break**;  }  **if** (sc == 0) count++;  }  **return** count;  }  *//\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*################## class Cars*  *//\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*##################*  *//## stored in cardata*  **class** **Cars** {  char carid[6];  char make[20];  char model[20];  char year[5];  char category[20];  char passengers;  float price;  short status; *// 1=avail. , 0=booked*  public:  void getcar();  void putcar();  void makecarid();  void CarTable();  void AvailTable(char \*);  void switch\_status(int op = 2);  char \*retcarid() { **return** (carid); }  char \*retmodel() { **return** (model); }  char \*retmake() { **return** (make); }  float retprice() { **return** (price); }  ~Cars() { */\*Destructor to distroy this class when the scope is over\*/*  }  } C;  void Cars::getcar() {  cout << "**\n\t**MAKE => ";  gets(make);  cout << "**\n\t**Model => ";  gets(model);  cout << "**\n\t**YEAR => ";  ventry(year, 1, 5);  cout << "**\n\t**CATEGORY => ";  ventry(category, 0, 16);  cout << "**\n\t**Passengers => ";  cin >> passengers;  cout << "**\n\t**Price per day of car => ";  cin >> price;  status = 1; *// status=1 means available;*  makecarid();  cout << "**\n\t**CarID => " << carid;  getch();  }  void Cars::putcar() {  cout << "**\n\t**CarID : " << carid; *// Displayed on Table*  cout << "**\n\n\t**MAKE : " << make; *// MAKE+model Displayed on table*  cout << "**\t\t**Model : " << model;  cout << "**\n\n\t**YEAR : " << year;  cout << "**\t\t**CATEGORY : " << category;  cout << "**\n\n\t**Passengers : " << passengers;  }  void Cars::makecarid() {  short lm = strlen(make);  short ly = strlen(year);  int pos = 0, i, j;  **for** (j = 0; j < 20; j++) i = random(4);  **if** (i == 0) {  carid[pos++] = tolower(make[0]);  carid[pos++] = tolower(make[1]);  }  **if** (i == 1) {  carid[pos++] = make[0];  carid[pos++] = make[lm - 1];  }  **if** (i == 2) {  carid[pos++] = make[0];  carid[pos++] = make[1];  carid[pos++] = make[2];  }  **if** (i == 3) {  carid[pos++] = year[ly - 2];  carid[pos++] = year[ly - 1];  carid[pos++] = model[0];  carid[pos++] = model[2];  }  randomize();  **for** (j = 0; j < 20; j++) i = random(4);  **if** (i == 0) {  carid[pos++] = year[ly - 2];  carid[pos++] = year[ly - 1];  }  **if** (i == 1) {  carid[pos++] = model[0];  carid[pos++] = model[1];  }  **if** (i == 2 && pos < 3) {  carid[pos++] = model[0];  carid[pos++] = model[1];  carid[pos++] = model[2];  }  **if** (i == 3) {  carid[pos++] = tolower(make[0]);  carid[pos++] = '#';  }  **for** (i = 0; i < 5; i++)  **if** (carid[i] == NULL || carid[i] == ' ') carid[i] = passengers;  carid[5] = '\0';  }  void Cars::CarTable() {  border(4, 3, 73, 20);  line(8, 4, 19); *// Vertical line After No.*  line(16, 4, 19); *// Vertical line After Carid*  line(37, 4, 19); *// Vertical line After Car*  line(55, 4, 19); *// After Category*  line(65, 4, 19); *// After Price*  line(5, 5, 72, 'h'); *// Horizontal line*  gotoxy(5, 4);  cout << " No";  gotoxy(9, 4);  cout << " CarID";  gotoxy(25, 4);  cout << "CAR";  gotoxy(43, 4);  cout << "CATEGORY";  gotoxy(56, 4);  cout << "Price/day";  gotoxy(69, 4);  cout << "STATUS";  gotoxy(1, 1);  int i = 0, r = 7;  ifstream carfile("carsdata", ios::in | ios::binary);  gotoxy(20, 2);  cout << "A L L C A R S R E P O R T T A B L E";  *// Read From File*  **while** (carfile.read((char \*)&C, **sizeof**(Cars))) {  gotoxy(6, r);  (i > 8) ? cout << (i + 1) : cout << " " << (i + 1);  gotoxy(10, r);  cout << carid;  gotoxy(18, r);  voutput(17, make, model);  gotoxy(39, r);  cout << category;  gotoxy(59, r);  cout << C.price;  gotoxy(67, r);  (status == 1) ? cout << "Available" : cout << "BOOKED!";  i++;  r++;  }  }  void Cars::AvailTable(char \*rentcarid) {  clrscr();  border(4, 4, 73, 17);  line(9, 5, 16, 'v'); *// Vertical line After No.*  line(16, 5, 16, 'v'); *// Vertical line After Carid*  line(39, 5, 16, 'v'); *// Vertical line After Car*  line(58, 5, 16, 'v'); *// After Category*  line(70, 5, 16, 'v'); *// After Price*  line(5, 6, 72, 'h'); *// Horizontal line*  gotoxy(5, 5);  cout << " No";  gotoxy(10, 5);  cout << " Year";  gotoxy(26, 5);  cout << "CAR";  gotoxy(45, 5);  cout << "CATEGORY";  gotoxy(60, 5);  cout << "Price/day";  gotoxy(71, 5);  cout << "Seats";  gotoxy(1, 2);  ifstream carfile("carsdata", ios::in | ios::binary);  *// Read From File*  int i, r;  cout << "**\t\t\t**DISPLAY AVAILABLE CARS";  **for** (i = 1, r = 8; carfile.read((char \*)&C, **sizeof**(C));)  **if** (status == 1) {  gotoxy(5, r);  cout << "[" << i << "]";  gotoxy(11, r);  cout << year;  gotoxy(18, r);  voutput(19, make, model);  gotoxy(41, r);  cout << category;  gotoxy(63, r);  cout << C.price;  gotoxy(73, r);  cout << passengers;  i++, r++;  }  carfile.clear();  carfile.seekg(0);  gotoxy(5, 24);  cout << "Select Car No. >> ";  cin >> r;  i = 1;  **while** (carfile.read((char \*)&C, **sizeof**(C))) {  **if** (i == r && status == 1) {  strcpy(rentcarid, carid);  **break**;  }  **if** (status == 1) {  i++;  }  }  }  void Cars::switch\_status(int op) {  **if** (op == 1)  status = 1;  **else** **if** (op == 0)  status = 0;  **else**  (status == 0) ? status = 1 : status = 0;  }  void status\_edit(char \*op = "#Nothing") {  fstream carfile("carsdata", ios::in | ios::out | ios::binary);  **while** (carfile.read((char \*)&C, **sizeof**(C))) {  **if** (\*op == '1' && \*(op + 1) == '\0')  C.switch\_status(1);  **else** if (\*op == '0' && \*(op + 1) == '\0')  C.switch\_status(0);  **else** if (strcmp(C.retcarid(), op) == 0) {  C.switch\_status();  carfile.seekg(carfile.tellg() - **sizeof**(C));  carfile.write((char \*)&C, **sizeof**(C));  **break**;  }  carfile.seekg(carfile.tellg() - **sizeof**(C));  carfile.write((char \*)&C, **sizeof**(C));  }  }  void AddCar() {  Cars C;  int i, N;  cout << "**\n**Enter no. of cars to be added => ";  cin >> N;  clrscr();  ofstream carfile("carsdata", ios::out | ios::binary | ios::app);  *// Write TO File*  cout << "**\n\t\t\t**ADD NEW CAR";  **for** (i = 0; i < N; i++) {  clrscr();  cout << "**\n\n\t**Add Car No. " << (i + 1) << endl;  C.getcar();  carfile.write((char \*)&C, **sizeof**(Cars));  }  carfile.close();  }  short SearchCar(Cars &D, char \*objective = NULL)  */\* Can only search for make, model, carid and price*  *\*/*  {  short count = 0, searchstatus = 0, i = 0;  Cars M[10];  ifstream carfile("carsdata", ios::in | ios::out | ios::binary);  **if** (\*objective == NULL) **return** 0;  *// Read from File and Write to File*  **while** (carfile.read((char \*)&C, **sizeof**(Cars))) {  **if** (strcmp(objective, C.retcarid()) == 0 ||  strcmpi(objective, C.retmodel()) == 0) {  D = C;  searchstatus++;  **break**;  } **else** **if** (strcmpi(objective, C.retmake()) == 0)  count++;  }  carfile.clear();  carfile.seekg(0);  **if** (count == 1)  **while** (carfile.read((char \*)&C, **sizeof**(Cars)))  **if** (strcmpi(objective, C.retmake()) == 0) {  D = C;  searchstatus++;  **break**;  }  **if** (count > 1) { *// clrscr();*  gotoxy(13, 13);  cout << "More than one record found !**\a**";  **while** (carfile.read((char \*)&C, **sizeof**(Cars)))  **if** (strcmpi(objective, C.retmake()) == 0) {  M[i] = C;  i++;  }  **for** (i = 0; i < count; i++) {  gotoxy(9, 15 + i);  cout << "[" << (i + 1) << "]"  << "**\t**CarID: " << M[i].retcarid();  gotoxy(36, 15 + i);  cout << "CAR: " << M[i].retmake() << " " << M[i].retmodel();  }  int op;  gotoxy(6, 18);  cout << " => ";  cin >> op;  carfile.clear();  D = M[op - 1];  searchstatus++;  }  carfile.close();  **if** (searchstatus)  **return** 1;  **else**  **return** 0;  }  void UpdateCar() {  short updatestatus = 0;  Cars D;  char update[20];  fstream carfile("carsdata", ios::in | ios::out | ios::binary);  *// Read from File and Write to File*  clrscr();  cout << "**\n**Enter Carid/Model/Make of Car to Update => ";  gets(update);  SearchCar(D, update);  D.putcar();  getch();  **while** (carfile.read((char \*)&C, **sizeof**(Cars)))  **if** (strcmp(update, C.retcarid()) == 0 ||  strcmpi(update, C.retmodel()) == 0 ||  strcmp(D.retcarid(), C.retcarid()) == 0) {  cout << "**\n**Car Found!";  updatestatus = 1;  carfile.seekg(carfile.tellg() - **sizeof**(Cars));  C.putcar();  C.getcar();  carfile.write((char \*)&C, **sizeof**(Cars));  **break**;  }  **if** (updatestatus)  cout << "**\n\t\t**Car Details Updated ! ";  **else**  cout << "**\n\t\t**Car Not Found !";  carfile.close();  getch();  }  void SearchCar\_UI() {  char info[30];  search:  clrscr();  border(3, 2, 74, 21, 2);  gotoxy(30, 4);  cout << "SEARCH AND DISPLAY CAR";  gotoxy(30, 5);  cout << "----------------------";  gotoxy(8, 7);  cout << "Carid: ";  gotoxy(8, 9);  cout << "Model: ";  gotoxy(8, 11);  cout << "MAKE: ";  gotoxy(36, 7);  cout << "# Press ENTER to skip";  gotoxy(36, 8);  cout << "# Atleast one field should be entered";  gotoxy(36, 9);  cout << "# Model=Car Model(eg. Corolla,Innova)";  gotoxy(36, 10);  cout << "# MAKE=Company(eg. BMW,TOYOTA)";  line(6, 12, 45, 'h');  gotoxy(15, 7);  gets(info);  **if** (info[0] == NULL || SearchCar(C, info) == 0) {  gotoxy(15, 9);  gets(info);  **if** (info[0] == NULL || SearchCar(C, info) == 0) {  gotoxy(14, 11);  gets(info);  **if** (SearchCar(C, info) == 0) {  gotoxy(8, 13);  cout << "RECORD NOT FOUND !";  gotoxy(8, 15);  cout << "Do you wish to continue (y/n)....";  **if** (tolower(::op = getch()) == 'n') {  flushall();  **return**;  }  **goto** search;  }  }  }  flushall();  clear(7, 13, 65, 7);  gotoxy(13, 12);  C.putcar();  cout << "**\n\n\t\t**Do you wish to search again ? (y/n)......";  border(3, 2, 74, 21, 2);  choose:  **if** (tolower(::op = getch()) == 'y')  **goto** search;  **else** if (::op == 'n') {  flushall();  **return**;  } **else**  **goto** choose;  }  void DeleteCar() {  char info[30];  search:  clrscr();  border(3, 2, 74, 21, 2);  gotoxy(35, 4);  cout << "DELETE CAR";  gotoxy(35, 5);  cout << "----------";  gotoxy(8, 7);  cout << "Carid: ";  gotoxy(8, 9);  cout << "Model: ";  gotoxy(8, 11);  cout << "MAKE: ";  gotoxy(36, 7);  cout << "# Press ENTER to skip";  gotoxy(36, 8);  cout << "# Atleast one field should be entered";  gotoxy(36, 9);  cout << "# Model=Car Model(eg. Corolla,Innova)";  gotoxy(36, 10);  cout << "# MAKE=Company(eg. BMW,TOYOTA)";  line(6, 12, 45, 'h');  gotoxy(15, 7);  gets(info);  **if** (info[0] == NULL || SearchCar(C, info) == 0) {  gotoxy(15, 9);  gets(info);  **if** (info[0] == NULL || SearchCar(C, info) == 0) {  gotoxy(14, 11);  gets(info);  **if** (SearchCar(C, info) == 0) {  gotoxy(8, 13);  cout << "RECORD NOT FOUND !";  gotoxy(8, 15);  cout << "Do you wish to continue (y/n)....";  **if** (tolower(::op = getch()) == 'n') {  flushall();  **return**;  }  **goto** search;  }  }  }  flushall();  clear(7, 13, 65, 7);  gotoxy(13, 12);  C.putcar();  border(3, 2, 74, 21, 2);  gotoxy(17, 21);  cout << '\_';  gotoxy(17, 22);  cout << "Delete";  gotoxy(38, 21);  cout << '\_';  gotoxy(38, 22);  cout << "Retry";  gotoxy(58, 21);  cout << '\_';  gotoxy(58, 22);  cout << "Cancel";  choose:  **if** (tolower(::op = getch()) == 'r')  **goto** search;  **else** if (::op == 'c') {  flushall();  **return**;  } **else** if (::op == 'd') {  del:  clrscr();  border(3, 2, 74, 21, 2);  gotoxy(8, 4);  cout << "**\a**Are you sure you want to DELETE RECORD !! (Y/N): ";  ventry(&::op, 3, 2);  **if** (::op == 'Y') {  Cars D;  ifstream carfile;  ofstream temp;  carfile.open("carsdata", ios::binary);  temp.open("temp", ios::binary);  **while** (carfile.read((char \*)&D, **sizeof**(Cars)))  **if** (strcmp(C.retcarid(), D.retcarid()) != 0)  temp.write((char \*)&D, **sizeof**(D));  temp.close();  carfile.close();  flushall();  remove("carsdata");  rename("temp", "carsdata");  gotoxy(8, 6);  cout << "RECORD Successfully DELETED....";  getch();  } **else** **if** (::op == 'N')  **return**;  **else**  **goto** del;  } **else**  **goto** choose;  }  void DisplayCar() {  Cars D;  int i = 0;  ifstream carfile("carsdata", ios::in | ios::binary);  *// Read From File*  clrscr();  cout << "**\n\t\t\t**DISPLAY AVAILABLE CARS";  **while** (carfile.read((char \*)&D, **sizeof**(Cars))) {  cout << "**\n\n\t**Car No. " << (i + 1) << endl;  D.putcar();  getch();  i++;  }  carfile.close();  }  void rentcar\_design() {  border(3, 2, 74, 21, 2);  gotoxy(9, 7);  cout << "Drv. Lic. #: ";  gotoxy(46, 7);  cout << "Name: ";  line(8, 8, 60, 'h');  gotoxy(9, 10);  cout << "PhoneNo. & or Address: ";  gotoxy(9, 12);  cout << "City: ";  gotoxy(46, 12);  cout << "ZIPCode: ";  gotoxy(9, 14);  cout << "Email: ";  gotoxy(46, 14);  cout << "Rent Car ID: ";  line(8, 16, 60, 'h');  gotoxy(9, 17);  cout << "FROM DATE (dd/mm/yyyy): ";  gotoxy(45, 17);  cout << "TO DATE (dd/mm/yyyy): ";  }  *//#############\*\*\*\*\*\*\*\*\*\*\*\*\*\* CLASS rentDate \*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*###############*  *// TO Calculate difference in dates*  **struct** date {  unsigned short day, month, year;  };  void getdate(date &, short, short);  void putdate(date &, short, short);  **class** **rentDate** {  public:  date from, to;  unsigned int convert\_datetodays(date D, unsigned short y = 0);  unsigned int calc\_diff() {  **return** (convert\_datetodays(to, from.year) - convert\_datetodays(from));  }  short leap\_year(unsigned short year) {  **if** ((year % 4 == 0) && !(year % 100 == 0) || year % 400 == 0)  **return** 1;  **else**  **return** 0;  }  short verifydate(short, short code = 1);  void Get\_fromdate(short p, short q) { getdate(from, p, q); }  void Get\_todate(short p, short q) { getdate(to, p, q); }  void Put\_fromdate(short p, short q) { putdate(from, p, q); }  void Put\_todate(short p, short q) { putdate(to, p, q); }  ~rentDate() { */\*Destructor, the destroyer\*/*  }  } A;  void getdate(date &d, short p, short q) {  in:  clear(p, q, 12);  gotoxy(p, q);  cout << " / / ";  gotoxy(p, q);  cin >> d.day;  **if** (!A.verifydate(d.day, 1)) {  cin.clear();  cin.ignore(1000, '\n');  **goto** in;  }  **if** (d.day < 10) {  gotoxy(p, q);  cout << 0 << d.day;  }  gotoxy(p + 3, q);  cin >> d.month;  **if** (!A.verifydate(d.month, 2)) {  cin.clear();  cin.ignore(100, '\n');  **goto** in;  }  **if** (d.month < 10) {  gotoxy(p + 3, q);  cout << 0 << d.month;  }  gotoxy(p + 6, q);  cin >> d.year;  **if** (!A.verifydate(d.year, 3)) {  cin.clear();  cin.ignore(100, '\n');  **goto** in;  }  }  void putdate(date &d, short p, short q) {  gotoxy(p, q);  cout << setw(2) << setfill('0') << d.day << '/';  cout << setw(2) << setfill('0') << d.month << '/' << d.year;  }  short rentDate::verifydate(short d, short code) {  **if** (code == 2) {  **if** (d >= 1 && d <= 12)  **return** 1;  **else**  **return** 0;  } **else** **if** (code == 3) {  **if** (d >= 2016 && d <= 2070)  **return** 1;  **else**  **return** 0;  } **else** {  **if** (d >= 1 && d <= 31)  **return** 1;  **else**  **return** 0;  }  }  unsigned int rentDate::convert\_datetodays(date D, unsigned short y) {  unsigned int days = 0;  short i;  **for** (unsigned short k = y; k < D.year; k++) {  **if** (y == 0) **break**;  **if** (leap\_year(k))  days += 366;  **else**  days += 365;  }  **for** (i = 1; i < D.month; i++) {  **if** (i == 4 || i == 6 || i == 9 || i == 11)  days += 30;  **else** if (i == 2) {  **if** (leap\_year(D.year))  days += 29;  **else**  days += 28;  } **else**  days += 31;  }  days += D.day;  **return** days;  }  *//\*\*\*\*\*\*\*\*\*\*\*\*/\*\*\*\*\*\*\*\*\*\*\*\*\*\*/\*\*\*\*\*\*\*\*\*\*\*\*\* class Customer*  *//\*\*\*\*\*\*\*\*\*\*\*\*\*/\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*/\*\*\*\*\*\*\*\*\*\*\*\*\*/\*\*\*\*\**  */\* stored in CUST\_record \*/*  **class** **Customer** {  char lic\_no[12];  char name[30], email[30];  char address[100], city[30];  char rentcarid[10];  char zipcode[10];  public:  rentDate T;  void getcustomer();  void putcustomer();  void CustomerCar\_table();  void get\_rentcarid() { C.AvailTable(rentcarid); }  void customer\_report();  char \*ret\_rentcarid() { **return** rentcarid; }  char \*ret\_licno() { **return** lic\_no; }  char \*ret\_name() { **return** name; }  char \*ret\_email() { **return** email; }  char \*ret\_address() { **return** address; }  ~Customer() { */\*Destructor the destroyer\*/*  }  } N;  void Customer::getcustomer() {  lic:  gotoxy(22, 7);  ventry(lic\_no, 1, 10);  **if** (lic\_no[0] == NULL) **goto** lic;  name:  gotoxy(52, 7);  ventry(name, 0, 23);  **if** (name[0] == NULL) **goto** name;  gotoxy(33, 10);  gets(address);  gotoxy(15, 12);  gets(city);  gotoxy(55, 12);  ventry(zipcode, 1, 6);  gotoxy(16, 14);  ventry(email, 3, 26);  date:  T.Get\_fromdate(13, 18);  T.Get\_todate(49, 18);  **if** (T.calc\_diff() > 365) {  gotoxy(13, 20);  cout << "Sorry, maximum rent period is 365 days !..........";  getch();  clear(13, 18, 60, 4);  **goto** date;  }  flushall();  }  void Customer::putcustomer() {  gotoxy(22, 7);  cout << lic\_no;  gotoxy(52, 7);  cout << name;  gotoxy(33, 10);  cout << address;  gotoxy(15, 12);  cout << city;  gotoxy(55, 12);  puts(zipcode);  gotoxy(16, 14);  puts(email);  gotoxy(59, 14);  puts(rentcarid);  T.Put\_fromdate(13, 18);  T.Put\_todate(50, 18);  }  void Customer::CustomerCar\_table() {  clrscr();  gotoxy(20, 2);  cout << "C U S T O M E R - C A R R E P O R T T A B L E";  ifstream customer("CUST\_record", ios::in | ios::binary);  **if** (!customer) {  cout << "**\n\n\t\a**File Does Not Exist / Unable to Open File ";  getch();  } **else** {  Cars D;  border(4, 3, 73, 20, 1);  line(16, 4, 19); *// Vertical line After Lic. #*  line(38, 4, 19); *// Vertical line After Customer*  line(46, 4, 19); *// After CarID*  line(67, 4, 19); *// After CAR*  line(5, 5, 72, 'h'); *// Horizontal line*  gotoxy(5, 4);  cout << " Licence #";  gotoxy(23, 4);  cout << " CUSTOMER ";  gotoxy(39, 4);  cout << " CarID";  gotoxy(54, 4);  cout << " CAR";  gotoxy(68, 4);  cout << "Price/day";  short i = 0, r = 7;  *// Read From File*  **while** (customer.read((char \*)&N, **sizeof**(N))) {  SearchCar(D, rentcarid);  gotoxy(40, r);  cout << rentcarid;  gotoxy(6, r);  cout << lic\_no;  gotoxy(48, r);  voutput(17, D.retmake(), D.retmodel());  gotoxy(18, r);  cout << name;  gotoxy(71, r);  cout << D.retprice();  *// getch();*  i++;  r += 2;  }  }  }  void Customer::customer\_report() {  clrscr();  gotoxy(22, 2);  cout << "A L L C U S T O M E R R E P O R T";  ifstream customer("CUST\_record", ios::in | ios::binary);  **if** (!customer) {  gotoxy(8, 4);  cout << "**\a**File Does Not Exist / Unable to Open File ";  } **else** {  Cars D;  border(4, 3, 73, 20, 1);  gotoxy(2, 24);  cout << "\* Price may vary depending on the rented car and returning day";  gotoxy(4, 25);  cout << "and offers included";  line(16, 4, 19); *// Vertical line After Lic. #*  line(38, 4, 19); *// Vertical line After Customer*  line(49, 4, 19); *// After fromdate*  line(60, 4, 19); *// After todate*  line(67, 4, 19); *// After days*  line(5, 5, 72, 'h'); *// Horizontal line*  gotoxy(5, 4);  cout << " Licence #";  gotoxy(23, 4);  cout << " CUSTOMER ";  gotoxy(40, 4);  cout << "From Date";  gotoxy(51, 4);  cout << "To Date";  gotoxy(61, 4);  cout << " Days";  gotoxy(69, 4);  cout << "\*Price";  short i = 0, r = 7;  *// Read From File*  **while** (customer.read((char \*)**this**, **sizeof**(Customer))) {  SearchCar(D, rentcarid);  gotoxy(6, r);  cout << lic\_no;  gotoxy(18, r);  cout << name;  T.Put\_fromdate(39, r);  T.Put\_todate(50, r);  gotoxy(62, r);  cout << T.calc\_diff();  gotoxy(69, r);  cout << (T.calc\_diff() \* D.retprice());  i++;  r += 2; *// getch();*  }  }  }  void AddCustomer() {  clrscr();  gotoxy(32, 4);  cout << "RENT A CAR SYSTEM";  gotoxy(32, 5);  cout << "-----------------";  rentcar\_design();  N.getcustomer();  N.get\_rentcarid();  clrscr();  rentcar\_design();  gotoxy(32, 4);  cout << "RENT A CAR SYSTEM";  gotoxy(32, 5);  cout << "-----------------";  N.putcustomer();  gotoxy(23, 21);  cout << '\_';  gotoxy(23, 22);  cout << "Save";  gotoxy(51, 21);  cout << '\_';  gotoxy(51, 22);  cout << "Cancel";  save:  ::op = tolower(getch());  **if** (::op == 's') { *// Write TO File*  ofstream cust;  cust.open("CUST\_record", ios::out | ios::binary | ios::app);  cust.write((char \*)&N, **sizeof**(**class** **Customer**));  cust.close();  flushall();  status\_edit(N.ret\_rentcarid());  clrscr();  gotoxy(30, 12);  cout << " PLEASE WAIT ! ";  loading(52, 12);  gotoxy(30, 12);  cout << " CUSTOMER SAVED ! ";  **for** (short i = 0; i < 5; i++) delay();  } **else** **if** (::op == 'c') {  flushall();  **return**;  } **else**  **goto** save;  }  void DisplayCust() {  clrscr();  gotoxy(32, 4);  cout << "RENT A CAR SYSTEM";  gotoxy(32, 5);  cout << "-----------------";  ifstream cust;  cust.open("CUST\_record", ios::in | ios::binary);  cust.clear();  cust.seekg(0);  *// Read From File*  **while** (cust.read((char \*)&N, **sizeof**(**class** **Customer**))) {  clear(4, 6, 73, 18);  rentcar\_design();  N.putcustomer();  flushall();  getch();  }  cust.close();  }  short SearchCust(Customer &D, char \*objective = "#Nothing", short name = 0)  */\* Can only search for Drv\_Lic.#, \*Name, email\_id and rentcarid*  *\* more than one record;*  *\*/* {  short count = 0, searchstatus = 0, i = 0;  Customer M[10];  ifstream customer("CUST\_record", ios::in | ios::out | ios::binary);  **if** (\*objective == NULL) **return** 0;  *// Read from File and Write to File*  **while** (customer.read((char \*)&N, **sizeof**(Customer))) {  **if** (strcmp(objective, N.ret\_rentcarid()) == 0 ||  strcmp(objective, N.ret\_email()) == 0 ||  strcmp(objective, N.ret\_licno()) == 0) {  D = N;  searchstatus++;  **break**;  } **else** **if** (name == 1 && namecmp(N.ret\_name(), objective))  count++;  }  *// gotoxy(1,1);cout<<"{ "<<count<<" }"; //For Testing*  customer.clear();  customer.seekg(0);  **if** (count == 1)  **while** (customer.read((char \*)&N, **sizeof**(Customer)))  **if** (namecmp(N.ret\_name(), objective)) {  D = N;  searchstatus++;  **break**;  }  **if** (count > 1) {  gotoxy(8, 16);  cout << "More than one record found !**\a**";  **while** (customer.read((char \*)&N, **sizeof**(Customer)))  **if** (namecmp(N.ret\_name(), objective)) {  M[i] = N;  i++;  }  gotoxy(9, 17);  cout << "Option**\t\t** Drv\_Lic.#"  << "**\t\t\t**Full Name";  gotoxy(7, 18);  cout << "------------------------------------------------------------";  **for** (i = 0; i < count; i++) {  gotoxy(10, 19 + i);  cout << "[" << (i + 1) << "]"  << "**\t\t**" << setw(10) << setfill(' ') << M[i].ret\_licno();  cout << "**\t\t**" << setw(17) << M[i].ret\_name();  }  int op;  opt:  gotoxy(8, 22);  cout << "Enter Option / 0 to exit => ";  cin >> op;  **if** (op < i + 2) {  **if** (op == 0) {  clear(6, 16, 65, 7);  **return** 0;  }  D = M[op - 1];  searchstatus++;  } **else**  **goto** opt;  }  customer.close();  **if** (searchstatus)  **return** 1;  **else**  **return** 0;  }  void SearchCust\_UI() {  char info[30];  search:  clrscr();  border(3, 2, 74, 21, 2);  gotoxy(26, 4);  cout << "SEARCH AND DISPLAY CUSTOMER";  gotoxy(26, 5);  cout << "---------------------------";  gotoxy(8, 8);  cout << "Drv. Lic. #: ";  gotoxy(8, 10);  cout << "Rent Car ID: ";  gotoxy(8, 12);  cout << "Name: ";  gotoxy(8, 14);  cout << "Email: ";  gotoxy(37, 8);  cout << "# Press ENTER to skip";  gotoxy(37, 10);  cout << "# Atleast one field should be entered";  gotoxy(37, 12);  cout << "# Drv. Lic. # = Licence No.of Customer";  gotoxy(37, 14);  cout << "# Rent Car ID = CarID of rented Car";  line(6, 15, 45, 'h');  gotoxy(21, 8);  gets(info); *// input lic\_no.*  **if** (info[0] == NULL || SearchCust(N, info) == 0) *// search lic\_no.*  {  gotoxy(21, 10);  gets(info); *// input rentcarid*  **if** (info[0] == NULL || SearchCust(N, info) == 0) *// search rentcarid*  {  gotoxy(14, 12);  gets(info); *// input name*  **if** (info[0] == NULL || SearchCust(N, info, 1) == 0) *// search name*  {  gotoxy(15, 14);  gets(info); *// input email*  **if** (SearchCust(N, info) == 0) *// search email*  {  gotoxy(8, 16);  cout << "RECORD NOT FOUND !";  gotoxy(8, 18);  cout << "Do you wish to continue (y/n)....";  **if** (tolower(::op = getch()) == 'n') {  flushall();  **return**;  }  **goto** search;  }  }  }  }  flushall();  clrscr();  rentcar\_design();  N.putcustomer();  gotoxy(29, 4);  cout << "RENTAL INFORMATION";  gotoxy(29, 5);  cout << "------------------";  gotoxy(20, 21);  cout << "Do you wish to search again ? (y/n)......";  choose:  **if** (tolower(::op = getch()) == 'y')  **goto** search;  **else** if (::op == 'n') {  flushall();  **return**;  } **else**  **goto** choose;  }  void deletecust(Customer D) {  ifstream customer;  ofstream temp;  customer.open("CUST\_record", ios::binary);  temp.open("temp", ios::binary);  **while** (customer.read((char \*)&N, **sizeof**(Customer)))  **if** (strcmp(D.ret\_licno(), N.ret\_licno()) != 0)  temp.write((char \*)&N, **sizeof**(Customer));  temp.close();  customer.close();  flushall();  remove("Cust\_record");  rename("temp", "Cust\_record");  }  void DeleteCust() {  char info[30];  search:  clrscr();  border(3, 2, 74, 21, 2);  gotoxy(33, 4);  cout << "DELETE CUSTOMER";  gotoxy(33, 5);  cout << "---------------";  gotoxy(8, 8);  cout << "Drv. Lic. #: ";  gotoxy(8, 12);  cout << "Name: ";  gotoxy(8, 14);  cout << "Email: ";  gotoxy(8, 10);  cout << "Rent Car ID: ";  gotoxy(37, 8);  cout << "# Press ENTER to skip";  gotoxy(37, 10);  cout << "# Atleast one field should be entered";  gotoxy(37, 12);  cout << "# Drv. Lic. # = Licence No.of Customer";  gotoxy(37, 14);  cout << "# Rent Car ID = CarID of rented Car";  line(6, 15, 45, 'h');  gotoxy(21, 8);  gets(info); *// input lic\_no.*  **if** (info[0] == NULL || SearchCust(N, info) == 0) *// search lic\_no.*  {  gotoxy(21, 10);  gets(info); *// input rentcarid*  **if** (info[0] == NULL || SearchCust(N, info) == 0) *// search rentcarid*  {  gotoxy(14, 12);  gets(info); *// input name*  **if** (info[0] == NULL || SearchCust(N, info, 1) == 0) *// search name*  {  gotoxy(15, 14);  gets(info); *// input email*  **if** (SearchCust(N, info) == 0) *// search email*  {  gotoxy(8, 16);  cout << "RECORD NOT FOUND !";  gotoxy(8, 18);  cout << "Do you wish to continue (y/n)....";  **if** (tolower(::op = getch()) == 'n') {  flushall();  **return**;  }  **goto** search;  }  }  }  }  flushall();  clrscr();  rentcar\_design();  N.putcustomer();  gotoxy(29, 4);  cout << "RENTAL INFORMATION";  gotoxy(29, 5);  cout << "------------------";  gotoxy(20, 21);  gotoxy(17, 21);  cout << '\_';  gotoxy(17, 22);  cout << "Delete";  gotoxy(38, 21);  cout << '\_';  gotoxy(38, 22);  cout << "Retry";  gotoxy(58, 21);  cout << '\_';  gotoxy(58, 22);  cout << "Cancel";  choose:  **if** (tolower(::op = getch()) == 'r')  **goto** search;  **else** if (::op == 'c') {  flushall();  **return**;  } **else** if (::op == 'd') {  del:  clrscr();  border(3, 2, 74, 21, 2);  gotoxy(8, 4);  cout << "**\a**Are you sure you want to DELETE RECORD !! (Y/N): ";  ventry(&::op, 3, 2);  **if** (::op == 'Y') {  status\_edit(N.ret\_rentcarid());  deletecust(N);  gotoxy(8, 6);  cout << "RECORD Successfully DELETED....";  getch();  } **else** **if** (::op == 'N')  **return**;  **else**  **goto** del;  } **else**  **goto** choose;  }  **class** **Bill** { *// Customer details*  char lic\_no[12];  char name[30];  char contact[100];  *// Car details*  char carid[10];  char car[40];  float carprice;  rentDate D;  date currentdate;  float totalprice;  public:  void makeBill(Customer);  void getcar(char \*);  *// void getdate(date&,short,short);*  void calc\_totalprice();  unsigned int calc\_returndaydiff();  void editdate();  void test() {  gotoxy(8, 1);  cout << totalprice;  }  void editprice();  void printbill();  void savebill(); *// Saves Bill in binaryfile"bills.dat"*  };  void Bill::makeBill(Customer N) {  border(3, 2, 74, 21, 2);  gotoxy(10, 4);  cout << "Enter Current Date: ";  getdate(currentdate, 30, 4);  strcpy(lic\_no, N.ret\_licno());  strcpy(name, N.ret\_name());  strcpy(contact, N.ret\_address());  getcar(N.ret\_rentcarid());  D = N.T;  calc\_totalprice();  }  unsigned int Bill::calc\_returndaydiff() {  signed int days;  unsigned int a = A.convert\_datetodays(currentdate, 2016);  unsigned int b = A.convert\_datetodays(D.to, 2016);  days = a - b;  **if** (days > 2)  **return** days;  **else**  **return** 0;  }  void Bill::editdate() {  clear(32, 15, 12);  getdate(D.to, 32, 15);  calc\_totalprice();  }  void Bill::editprice() {  clear(10, 20, 60);  gotoxy(10, 20);  cout << "New Amount ( Rs " << totalprice << " )= Rs ";  cin >> totalprice;  }  void Bill::getcar(char \*rentcarid) {  Cars T;  SearchCar(T, rentcarid);  strcpy(carid, T.retcarid());  strcpy(car, T.retmake());  strcat(car, " ");  strcat(car, T.retmodel());  carprice = T.retprice();  }  void Bill::calc\_totalprice() {  totalprice = D.calc\_diff() \* carprice;  **if** (calc\_returndaydiff())  totalprice += (carprice \* 1.5 \* calc\_returndaydiff());  **if** (totalprice > 100000) {  *// gotoxy(38,20);cout<<"You have recieved 25% discount!!"<<endl;*  totalprice \*= 0.75;  } **else** **if** (totalprice > 50000) {  *// gotoxy(38,20);cout<<"You have recieved 20% discount!!"<<endl;*  totalprice \*= 0.8;  } **else** **if** (totalprice > 25000) {  *// gotoxy(38,20);cout<<"You have recieved 15% discount!!";*  totalprice \*= 0.85;  } **else** **if** (totalprice > 10000) {  *// gotoxy(38,20);cout<<"You have recieved 10% discount!!";*  totalprice \*= 0.9;  }  }  void Bill::printbill() {  gotoxy(2, 1);  cout << "Current Date: ";  putdate(currentdate, 15, 1);  border(3, 2, 74, 22, 2);  border(8, 5, 64, 16, 0);  gotoxy(27, 4);  cout << "C U S T O M E R B I L L ";  gotoxy(10, 6);  cout << "CUSTOMER DETAILS";  line(9, 7, 30, 'h', 1);  gotoxy(10, 8);  cout << "Drv. Lic. #: " << lic\_no;  gotoxy(46, 8);  cout << "NAME: " << name;  gotoxy(10, 9);  cout << "Contact INFO: " << contact;  line(9, 10, 30, 'h', 1);  gotoxy(10, 11);  cout << "RENTAL INFO";  line(9, 12, 30, 'h', 1);  gotoxy(10, 13);  cout << "RentedCar\_ID: " << carid;  gotoxy(46, 13);  cout << "CAR: " << car;  gotoxy(10, 14);  cout << "FROM DATE (dd/mm/yyyy): ";  D.Put\_fromdate(34, 14);  gotoxy(10, 15);  cout << "TO DATE (dd/mm/yyyy): ";  D.Put\_todate(32, 15);  line(9, 16, 30, 'h', 1);  gotoxy(10, 17);  cout << "PRICE";  line(9, 18, 30, 'h', 1);  gotoxy(10, 19);  cout << "Price/day for Car = Rs " << carprice;  gotoxy(41, 19);  cout << "No. of Days Rented = " << D.calc\_diff();  **if** (calc\_returndaydiff()) cout << "+" << calc\_returndaydiff();  **if** (calc\_returndaydiff()) {  gotoxy(41, 20);  cout << "Fine(Late Return) = Rs "  << (1.5 \* calc\_returndaydiff() \* carprice);  }  gotoxy(10, 20);  cout << "Net Amount = Rs. " << totalprice;  }  void Bill::savebill() {  ofstream bill;  bill.open("bills.dat", ios::out | ios::app);  bill.write((char \*)**this**, **sizeof**(Bill));  bill.close();  }  void ReturnCar() {  char info[30];  search:  clrscr();  border(3, 2, 74, 21, 2);  gotoxy(24, 4);  cout << "R E T U R N A N D B I L L I N G";  gotoxy(24, 5);  cout << "-----------------------------------";  gotoxy(8, 8);  cout << "Drv. Lic. #: ";  gotoxy(8, 10);  cout << "Rent Car ID: ";  gotoxy(8, 12);  cout << "Name: ";  gotoxy(8, 14);  cout << "Email: ";  gotoxy(37, 8);  cout << "# Press ENTER to skip";  gotoxy(37, 10);  cout << "# Atleast one field should be entered";  gotoxy(37, 12);  cout << "# Drv. Lic. # = Licence No.of Customer";  gotoxy(37, 14);  cout << "# Rent Car ID = CarID of rented Car";  line(6, 15, 45, 'h');  gotoxy(21, 8);  gets(info); *// input lic\_no.*  **if** (info[0] == NULL || SearchCust(N, info) == 0) *// search lic\_no.*  {  gotoxy(21, 10);  gets(info); *// input rentcarid*  **if** (info[0] == NULL || SearchCust(N, info) == 0) *// search rentcarid*  {  gotoxy(14, 12);  gets(info); *// input name*  **if** (info[0] == NULL || SearchCust(N, info, 1) == 0) *// search name*  {  gotoxy(15, 14);  gets(info); *// input email*  **if** (SearchCust(N, info) == 0) *// search email*  {  gotoxy(8, 16);  cout << "RECORD NOT FOUND !";  gotoxy(8, 18);  cout << "Do you wish to continue (y/n)....";  **if** (tolower(::op = getch()) == 'n') {  flushall();  **return**;  }  **goto** search;  }  }  }  }  flushall();  clrscr();  rentcar\_design();  N.putcustomer();  gotoxy(29, 4);  cout << "RENTAL INFORMATION";  gotoxy(29, 5);  cout << "------------------";  gotoxy(20, 21);  gotoxy(17, 21);  cout << " \_";  gotoxy(17, 22);  cout << "View Bill";  gotoxy(38, 21);  cout << '\_';  gotoxy(38, 22);  cout << "Retry";  gotoxy(58, 21);  cout << '\_';  gotoxy(58, 22);  cout << "Cancel";  choose:  **if** (tolower(::op = getch()) == 'r')  **goto** search;  **else** if (::op == 'c') {  flushall();  **return**;  } **else** if (::op == 'b') {  clrscr();  Bill B;  B.makeBill(N);  billing:  clrscr();  B.printbill();  gotoxy(11, 22);  cout << '\_';  gotoxy(11, 23);  cout << "Return Car";  gotoxy(30, 22);  cout << " \_";  gotoxy(30, 23);  cout << "Edit Price";  gotoxy(49, 22);  cout << " \_";  gotoxy(49, 23);  cout << "Edit Date";  gotoxy(66, 22);  cout << " \_";  gotoxy(66, 23);  cout << "Exit";  **if** (tolower(::op = getch()) == 'r') {  clrscr();  border(3, 2, 74, 21, 2);  gotoxy(8, 4);  cout << "Confirm Return Car [y/n] > ";  cin >> ::op;  **if** (tolower(op) == 'y') {  status\_edit(N.ret\_rentcarid());  deletecust(N);  B.savebill();  gotoxy(8, 6);  cout << "Car Returned And Bill Saved......";  getch();  } **else**  **goto** billing;  } **else** **if** (::op == 'p') {  B.editprice();  B.test();  getch();  **goto** billing;  } **else** **if** (::op == 'd') {  B.editdate();  **goto** billing;  } **else** **if** (::op == 'x') {  flushall();  clrscr();  **return**;  } **else** {  cout << '\a';  **goto** billing;  }  } **else** {  cout << '\a';  **goto** choose;  }  }  void report\_menu() {  menu:  clrscr();  border(3, 2, 74, 21, 2);  border(23, 4, 37, 17, 0);  gotoxy(30, 6);  cout << "R E P O R T S M E N U";  gotoxy(29, 7);  cout << "-------------------------";  gotoxy(27, 9);  cout << " [1] View All Car Reports";  gotoxy(27, 11);  cout << " [2] View All Customer Report";  gotoxy(27, 13);  cout << " [3] View Customer - Car Report";  gotoxy(27, 15);  cout << " [4] GO BACK**\n**";  gotoxy(1, 1);  ventry(&::op);  **switch** (::op) {  **case** '1':  clrscr();  C.CarTable();  getch();  **break**;  **case** '2':  clrscr();  N.customer\_report();  getch();  **break**;  **case** '3':  clrscr();  N.CustomerCar\_table();  getch();  **break**;  **case** '4':  clrscr();  **return**;  default:  cout << '\a';  }  **goto** menu;  }  void status\_editmenu() {  menu:  clrscr();  cout << "**\n\t\t\t**EDIT CAR STATUS**\n\n**";  cout << "**\n\t**[1] EDIT All Status to [ AVAILABLE ] **\n** ";  cout << "**\n\t**[2] EDIT All Status to [ BOOKED! ]**\n**";  cout << "**\n\t**[3] Car Status Switch**\n**";  cout << "**\n\t**[4] GO BACK**\n**";  cout << "**\n**Enter your option => ";  ventry(&::op);  char carid[6];  **switch** (::op) {  **case** '1':  clrscr();  cout << '\a';  status\_edit("1");  getch();  **break**;  **case** '2':  clrscr();  cout << '\a';  status\_edit("0");  getch();  **break**;  **case** '3':  clrscr();  cout << '\a';  cout << "**\n\t**Enter CarID to Update: ";  gets(carid);  status\_edit(carid);  getch();  **break**;  **case** '4':  clrscr();  **return**;  default:  cout << '\a';  }  **goto** menu;  }  void recordedit\_menu() {  menu:  clrscr();  border(3, 2, 74, 21, 2);  border(23, 4, 37, 17, 0);  gotoxy(27, 5);  cout << "**\t**EDIT RECORD | MENU |";  gotoxy(27, 8);  cout << " [1] Update Car";  gotoxy(27, 10);  cout << " [2] Update Customer";  gotoxy(27, 12);  cout << " [3] DELETE Car";  gotoxy(27, 14);  cout << " [4] DELETE Customer";  gotoxy(27, 16);  cout << " [5] EDIT Status";  gotoxy(27, 18);  cout << " [6] GO BACK**\n**";  gotoxy(1, 1);  ventry(&::op);  **switch** (::op) {  **case** '1':  clrscr();  UpdateCar();  **break**;  **case** '2':  clrscr();  *// UpdateCust();*  **break**;  **case** '3':  clrscr();  DeleteCar();  **break**;  **case** '4':  clrscr();  DeleteCust();  **break**;  **case** '5':  clrscr();  status\_editmenu();  **break**;  **case** '6':  clrscr();  **return**;  default:  cout << '\a';  }  **goto** menu;  }  void SearchDisplay\_menu() {  menu:  clrscr();  border(3, 2, 74, 22, 2);  border(23, 4, 37, 18, 0);  gotoxy(27, 6);  cout << "**\t** SEARCH AND DISPLAY";  gotoxy(27, 10);  cout << " [1] Display ALL Cars";  gotoxy(27, 12);  cout << " [2] Display ALL Customers";  gotoxy(27, 14);  cout << " [3] SEARCH Car";  gotoxy(27, 16);  cout << " [4] SEARCH Customer";  gotoxy(27, 18);  cout << " [5] GO BACK**\n**";  gotoxy(1, 1);  ventry(&::op);  **switch** (::op) {  **case** '1':  clrscr();  DisplayCar();  **break**;  **case** '2':  clrscr();  DisplayCust();  **break**;  **case** '3':  clrscr();  SearchCar\_UI();  **break**;  **case** '4':  clrscr();  SearchCust\_UI();  **break**;  **case** '5':  clrscr();  **return**;  default:  cout << '\a';  }  **goto** menu;  }  void menu() {  clrscr();  border(3, 2, 74, 22, 2);  *// border(23,3,37,20,4);*  menu:  clrscr();  *// border(1,1,78,24,2);*  border(3, 2, 74, 22, 2);  border(23, 3, 37, 20, 0);  gotoxy(32, 4);  cout << "/ M A I N M E N U /";  gotoxy(32, 5);  cout << "--------------------";  gotoxy(27, 7);  cout << "**\t**[1] RENT A CAR";  gotoxy(27, 9);  cout << "**\t**[2] ADD CAR";  gotoxy(27, 11);  cout << "**\t**[3] SEARCH & DISPLAY";  gotoxy(27, 13);  cout << "**\t**[4] RETURN & BILLING";  gotoxy(27, 15);  cout << "**\t**[5] REPORTS";  gotoxy(27, 17);  cout << "**\t**[6] EDIT RECORD";  gotoxy(27, 19);  cout << "**\t**[7] ABOUT";  gotoxy(27, 21);  cout << "**\t**[8] EXIT**\n**";  gotoxy(1, 1);  ventry(&::op, 4, 1);  **switch** (::op) {  **case** '2':  clrscr();  AddCar();  **break**;  **case** '1':  clrscr();  AddCustomer();  **break**;  **case** '3':  clrscr();  SearchDisplay\_menu();  **break**;  **case** '5':  clrscr();  report\_menu();  **break**;  **case** '6':  clrscr();  recordedit\_menu();  **break**;  **case** '4':  clrscr();  ReturnCar();  **break**;  **case** '7':  clrscr();  About();  getch();  **break**;  **case** '8':  clrscr();  cout << "PROJECT HALTED";  exit(-1);  default:  cout << '\a';  }  **goto** menu;  }  void main() {  *// loading(4,4,15);*  L.check();  menu();  } |

Bibliography

* C++ Text Book
* Previous year’s projects
* <http://cppqa.blogspot.com>
* https://en.wikipedia.org
* <https://stackoverflow.com>
* <https://github.com>
* http://hilite.me