INDEX Page No

[About C++ 2](#_Toc26019585)

[Header File Used and Outline 5](#_Toc26019586)

[Source Code 8](#_Toc26019587)

[Sample Outputs 16](#_Toc26019588)

[Conclusion 1](#_Toc26019589)8

[Bibliography](#_Toc26019590) 20



# About C++

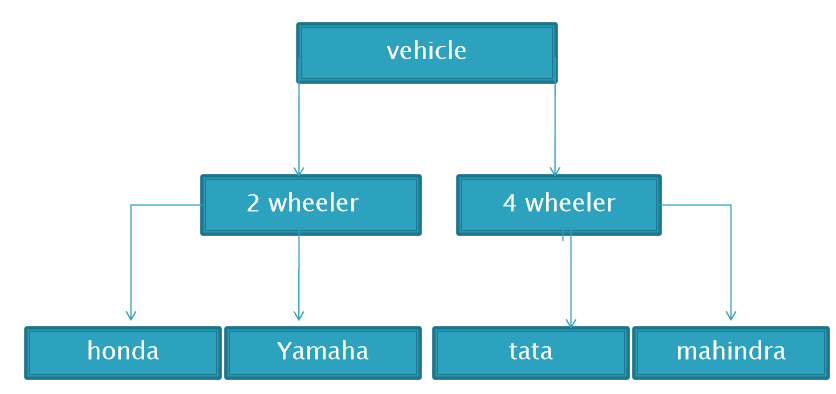
C++ is an extension to C Programming language. It was developed at AT&T Bell Laboratories in the early 1980s by Bjarne Stroustrup. It is a deviation from traditional procedural languages in the sense that it follows object oriented programming (OOP) approach which is quite suitable for managing large and complex programs.

**Oop Concepts**

* Inheritance
* Polymorphism
* Encapsulation

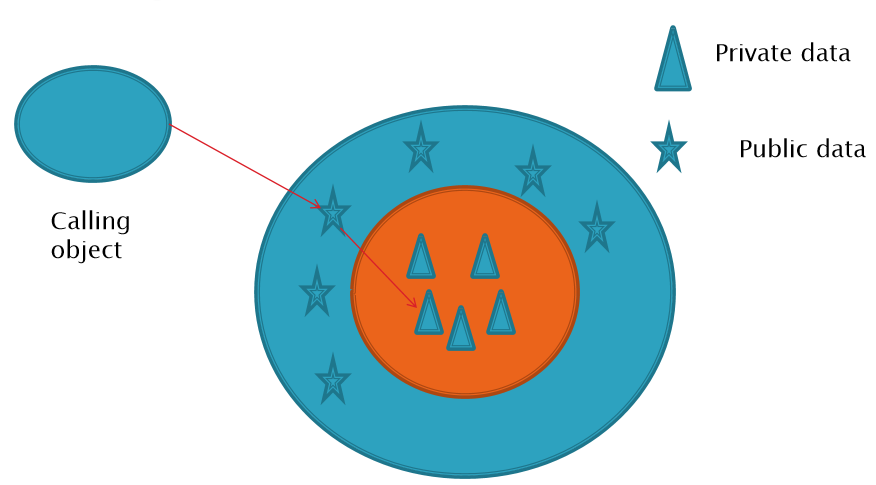
**Inheritance**

The capability of one class to inherit properties from another class as a child inherits some properties from his/her parents. The most important advantage of inheritance is code reusability. Once a base class is written and debugged, it can be used in various situations without having to redefine it or rewrite it. Reusing existing code saves time, money and efforts of writing the code again. Without redefining the old class, you can add new properties to desired class and redefine an inherited class member function.



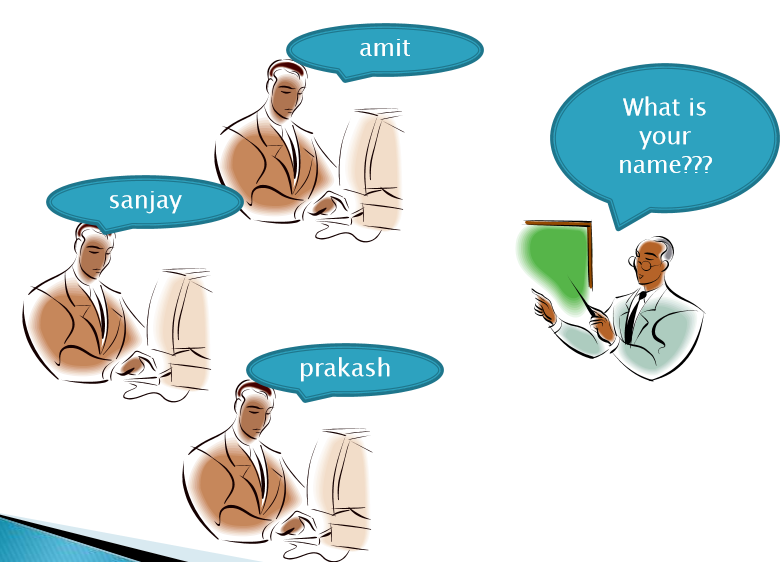
**Encapsulation**

Encapsulation is the most basic concept of OOP. It is the way of combining both data and the functions that operate on that data under a single unit. The only way to access the data is provided by the functions (that are combined along with the data). These functions are considered as member functions in C++. It is not possible to access the data directly. If you want to reach the data item in an object, you call a member function in the object. It will read the data item and return the value to you. The data is hidden, so it is considered as safe and far away from accidental alternation. Data and its functions are said to be encapsulated into a single entity.



Polymorphism

Polymorphism is a key to the power of OOP. It is the concept that supports the capability of data to be processed in more than one form.



**About this Project VMS:**

Vehicle Management System is a DOS application written for 64-bit Windows operating systems which focused in the area of adding, editing and deleting the customers, staff etc . The software can accommodate multiple vehicle types. Using this software a person can be register as a user and he can manage the car/lorry/bus routes and the staff or passengers‟ details.

# HEADER FILES

**1. #include<stdio.h>  (Standard input-output header)**

Used to perform input and output operations in C++ like scanf() and printf().

**2. #include<string.h> (String header)**

Perform string manipulation operations like strlen and strcpy.

**3. #include<conio.h> (Console input-output header)**

Perform console input and console output operations like clrscr() to clear the screen and getch() to get the character from the keyboard.

**4. #include<stdlib.h> (Standard library header)**

Perform standard utility functions like dynamic memory allocation, using functions such as malloc() and calloc().

**5. #inlcude<iostream> (Input Output Stream) –**Used as a stream of Input and Output

**6. #include<fstream.h> (File stream) –**Used to control the data to read from a file as an input and data to write into the file as an output.

# OUTLINE OF PROGRAM

class Vehicle{ // This is the base class!

public:

char Make[20];

char Model[20];

charRegNo[20];

Vehicle()

{

strcpy (Make, "");

strcpy (Model, "");

strcpy (RegNo, "");

}

};

class Record :public Vehicle

{

public:

charCust\_Name[20];

Record ()

{

strcpy (Cust\_Name, "");

}

void show ()

{

cout<<Cust\_Name<<" " << Make <<" "<< Model <<" " <<RegNo;

}

};

classData\_File

{

public:

fstreamrdwrite\_myfile;

Data\_File ()

{

}

~Data\_File ()

{

}

void Add ();

void Update ();

void Delete ();

void Display ();

Record ReadFromConsole();

voidwriteIntofile(Record \*N);

};

# Source Code

#include "HR\_VEHI.H"

void

Data\_File::Add ()

{

Record N;

char line[20];

// Open the file in read + write + File append mode

rdwrite\_myfile.open ("data.txt",ios::in|ios::out|ios::app);

N=ReadFromConsole();

// ensure you write at the end of file

// rdwrite\_myfile.seekp (0, ios::end);

writeIntofile(&N);

rdwrite\_myfile.close ();

}

void

Data\_File::Update ()

{

Record N;

char find[20];

unsigned char line[30];

char c='n';

longpos;

cout<< "Customer Name u want to update: ";

cin>> find;

cout<<endl;

//open the file with read + write + file pointer movement option options

rdwrite\_myfile.open ("data.txt",ios::in|ios::out|ios::ate);

// Intentionally bring the pointer to beggining of file

rdwrite\_myfile.seekg (0, ios::beg);

//strcpy(line,"");

while (!(rdwrite\_myfile.eof ()) )

{

//Search for the string which ends with eol character

rdwrite\_myfile.getline (line,20,'\n');

// Check if the searched string matches with "find" string

if (strcmp(line,find)==0)

{

//sucessfullyserched the string in the file

pos=rdwrite\_myfile.tellp();// Find the existing pointer position of the string find

pos=pos-2;//remove extra characters

pos=pos-strlen(find);//remove the size of the string

//You have the right pointe location to update the file now

rdwrite\_myfile.seekp(pos);

// If write empty file flag is set you need to delete record

//if(write\_empty\_record==0)

N=ReadFromConsole();

//rdwrite\_myfile.seekp (0, ios::end);

writeIntofile(&N);

//Your task of searching string is done exit the while after setting the yes flag

c='y';

break;

}//end of if

}//end of while

if(c!='y') cout<<"String not found\n";

rdwrite\_myfile.close ();

}//end of update

void

Data\_File::Delete ()

{

chardeleteline[20];

unsigned char line[30];

char c='n';

cout<< "Customer Name u want to Delete: ";

cin>>deleteline ;

cout<<endl;

ifstream sup;

sup.open("data.txt");

ofstream temp;

temp.open("temp.txt");

while (!(sup.eof()))

{

sup.getline (line,20,'\n');

if (strcmp(line,deleteline)!=0)

{ // copy all contents from existing file to the temp file

temp<< line <<endl;

}

else{

// You have been successfull in searching the record that needs to be deleted

//So Do not copy next 3+2 lines

for (int i=0;i<5;i++)

sup.getline (line,20,'\n');

c='y'; // set the delete success flag

}

}

temp.close();

sup.close();

if(c!='y')

{

cout<<"Delete string not found\n";

remove("temp.txt");

}

else

{ //copy the contents you have in the temp.txt file to data.txt

remove("data.txt");

rename("temp.txt","data.txt");

}

}//end of delete

void

Data\_File::Display ()

{

Record N;

int i=0;

char line[20];

rdwrite\_myfile.open ("data.txt",ios::in);

system("cls");

while (!(rdwrite\_myfile.eof ()) )

{

//Read a record

for (i=0;i<6;i++)

{

rdwrite\_myfile.getline (line,20,'\n');

//Print only if string length is greater tha 1

if( (strlen(line)>1) || (i>3) )

{

switch (i)

{

case 0:

{

cout<< "Customer Name:"<<line;

break;

}

case 1:

{

cout<< "Vehicle Make: " << line;

break;

}

case 2:

{

cout<< "Vehicle Model: "<< line;

break;

}

case 3:

{

cout<< "Vehicle Registration No:" << line;

break;

}

case 4:

case 5:

case 6:

{

cout<< line;

break;

}

//cout<<endl;

}; // end of switch

cout<<endl;

} //end of if

}//end of for

}//end of while

rdwrite\_myfile.close ();

}// end of display

Record Data\_File::ReadFromConsole()

{

Record NN;

cout<< "Customer Name: ";

cin>>NN.Cust\_Name;

cout<<endl;

cout<< "Vehicle Make: " ;

cin>>NN.Make;

cout<<endl;

cout<< "Vehicle Model: ";

cin>>NN.Model;

cout<<endl;

cout<< "Registration No: ";

cin>>NN.RegNo;

cout<<endl;

return NN;

}

voidData\_File::writeIntofile(Record \*N)

{

rdwrite\_myfile<< N->Cust\_Name<<endl;

rdwrite\_myfile<< N->Make <<endl;

rdwrite\_myfile<< N->Model <<endl;

rdwrite\_myfile<< N->RegNo<<endl<<endl<<endl;

}

Data\_FileFileObject;

int

main ()

{

char

c = '0';

system("cls");

while (1)

{

if (c != '\n')

{

do{

cout<< "Press a key to continue ...";

}while (cin.get()!='\n');

clrscr();

cout<< " \t VEHICALE DATA MANAGEMENT\n";

cout<< "Press 1: To Add an vehicle entry " <<endl;

cout<< "Press 2: To Edit an existing vehicle entry " <<endl;

cout<< "Press 3: To Delete a existing vehicle entry " <<endl;

cout<< "Press 4: To Display all vehicle entries" <<endl;

cout<< "Press 5: To Exit " <<endl;

}

scanf ("%c", &c);

switch (c)

{

case '1':

{

FileObject.Add ();

break;

}

case '2':

{

FileObject.Update ();

break;

}

case '3':

{

FileObject.Delete ();

break;

}

case '4':

{

FileObject.Display ();

break;

}

case '5':

{

exit (0);

break;

}

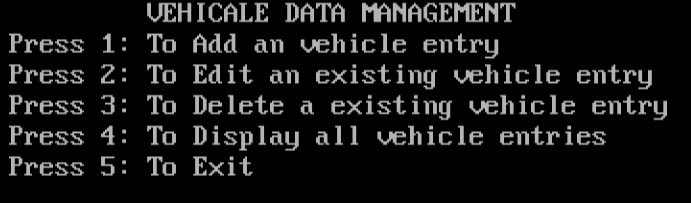
}; // end of switch

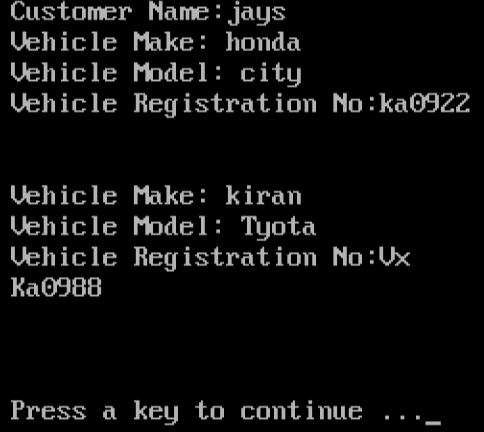
} // end of while

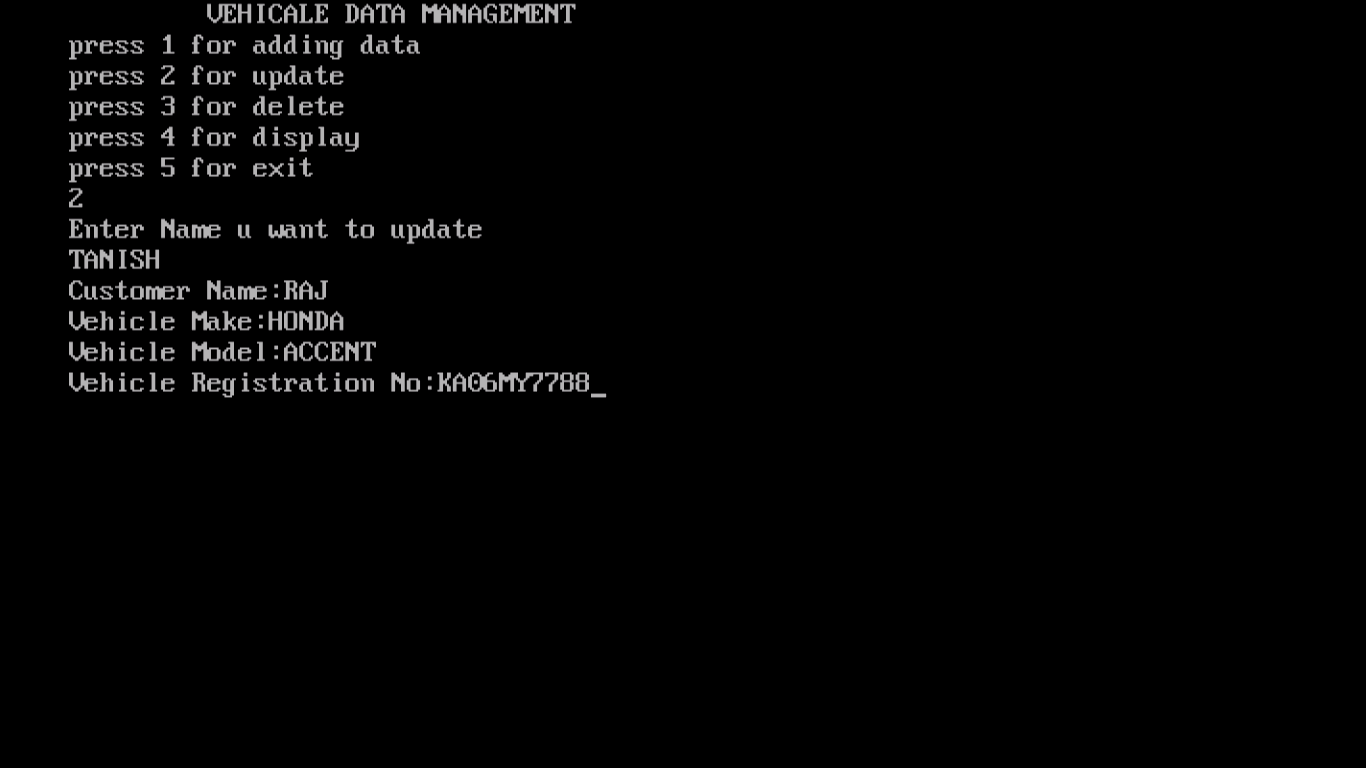
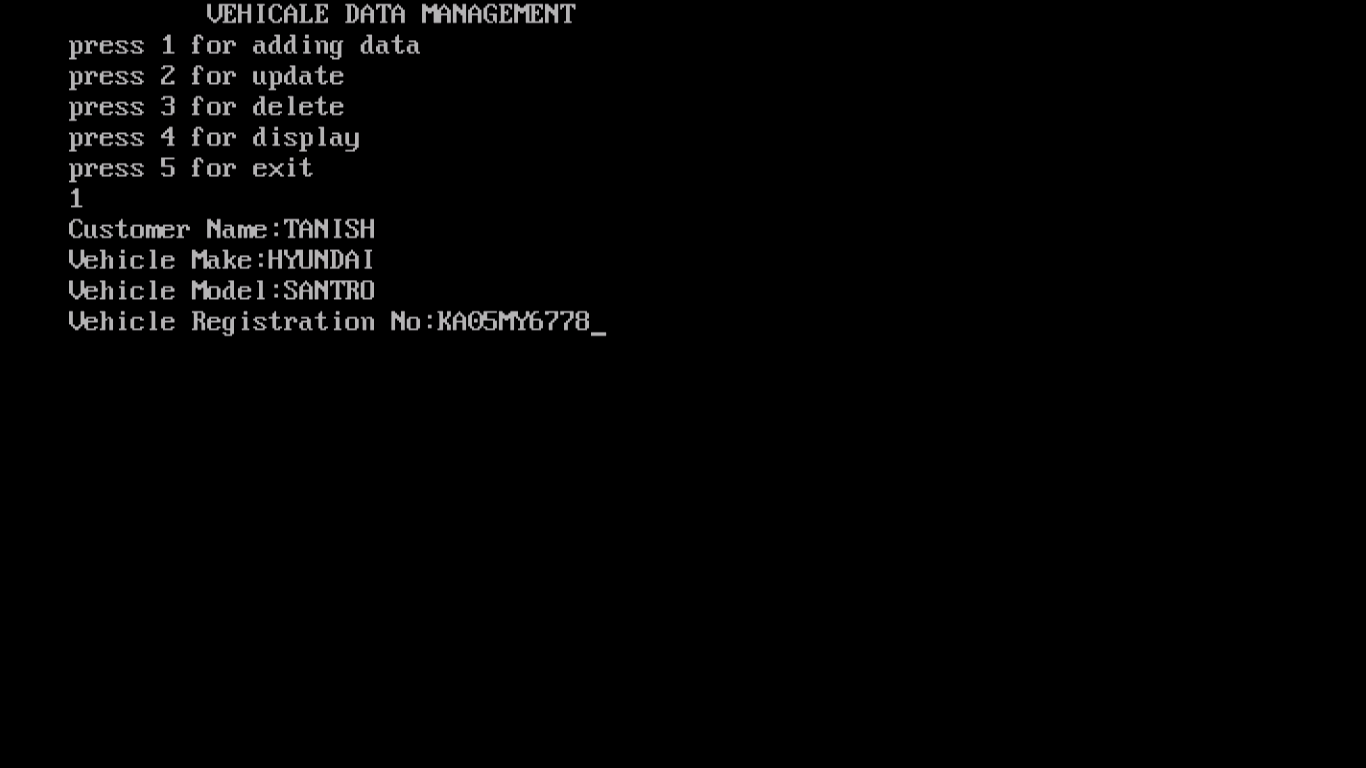
return 0;

} // end of main

# Sample Outputs







# Conclusion

The Vehicle Management System (VMS ) is the Proactive management science of using and maintaining a fleet of vehicles in such a fashion that whilst spending the minimum amount of money , a Vehicle manager is able to offer his/her users maximum flexibility , availability and reliability.

VMS has been developed in conjunction with the Transport Industry in an effort to provide a total Dos-based solution for Maintenance, Operational and Administrative Controls in any vehicle situation.

The competitive nature of our transport world today requires that we manage all aspects of the transport business. The VMS includes customer’s resources, the vehicles that play an integral part in delivering your products and services to the customer.

VMS’S - incredibly focused philosophy: To provide clients with strategic solutions that offer competitive

Advantages - resulting in quick paybacks, improvements in long-term costs and excellent customer service.

Specifically:

1. Reductions in total local & global vehicle operating costs

2. Improvements in local & international data control and flexibility

3. Improvements in overall transportation savings

The Benefits

It is widely accepted by VMS that available feature changes the playing field by providing numerous benefits. The science of Vehicle Management has been forever changed by these benefits:

• Improved Customer Service: more accurate prediction of arrival times, faster pickups and deliveries (dispatch closest vehicle by having real time information), reduce operator tardiness (information makes managing this a reality), Resolve Billing Disputes (recorded arrival and departure times)

• Efficient Logistics Management: assist lost drivers, eliminate employee downtime, Dispatch at a glance (view part or the entire Vehicle), knowledge will create confidence and reduce stress and assures better management decisions (route evaluation, attract and re-train the best employees, efficient scheduling)

• Profitability: serve more customers in less time, save on gas and slow Vehicle wear and tear, reduce unnecessary overtime, minimize unauthorized use of company vehicles, fewer resources to do equal or more work.

# Bibliography

[**http://cbseacademic.nic.in/**](http://cbseacademic.nic.in/web_material/CurriculumMain20/SrSecondary/Computer_Science_OLD-XII.pdf)

[**https://www.w3schools.in/**](https://www.w3schools.in/operating-system-tutorial/file-system-interface/)

[**http://web.engr.oregonstate.edu/**](http://web.engr.oregonstate.edu/~budd/Books/oopintro3e/info/chap01.pdf)

[**https://www.freecodecamp.org/**](https://www.freecodecamp.org/news/object-oriented-programming-concepts-21bb035f7260/)