Contents

[INTRODUCTION 2](#_Toc17088214)

[OBJECTIVE 7](#_Toc17088215)

[PROGRAM CODE 8](#_Toc17088216)

[OUTPUT 16](#_Toc17088217)

[CONCLUSION 18](#_Toc17088218)

[REFERENCE 19](#_Toc17088219)

# INTRODUCTION

**i.** Object Oriented Programming

Object-Oriented programming is a programming methodology that associates data structures with a set of operators which act upon it. In OOP, an instance of such an entity is known as object. OOP is a method of implementation in which programs are organized as co-operative collections of objects, each of which represents an instance of some class and whose classes are all members of a hierarchy of classes united through the property called inheritance.

The features of Object-Oriented Programming are listed below :-

* Object
* Class
* Abstraction
* Encapsulation
* Inheritance
* Reusability
* Polymorphism
* Dynamic binding
* Message passing

1.Object :- Objects are the entities in an object oriented system through which we perceive the world around us. We naturally see our environment as being composed of things which have recognizable identities & behavior. The entities are then represented as objects in the program. They may represent a person, a place, a bank account, or any item that the program must handle.For example Automobiles are objects as they have size, weight, color etc as attributes (that is data) and starting, pressing the brake, turning the wheel, pressing accelerator pedal etc as operation (that is functions).

2.Class :- Object consists of data and function tied together in a single unit. Functions are used to manipulate on the data. The entire construct of objects can be represented by a user defined data type in programming. The class is the user defined data type used to declare the objects. Actually objects are the variable of the user defined data type implemented as class. Once a class is defined, we can create any number of objects of its type. Each object that is created from the user defined type implemented as class is associated with the data type of that class. For example, manager, peon, secretary clerk are the objects of the class employee. Similarly, car, bus, jeep, truck are the objects of the class vehicle. Classes are user defined data type (like a struct in C programming language) and behave much like built in data type (like int, char, float) of programming language. It specifies what data and functions will be included in objects of that class. Defining class doesn’t create an object; however defining process specifies the data and function to be in the objects of its type.

3.Abstraction :- Abstraction is representing essential features of an object without including the background details or explanation. It focuses the outside view of an object, separating its essential behavior from its implementation. We can manage complexity through abstraction. Let’s take an example of vehicle. It is constructed from thousands of parts. The abstraction allows the driver of the vehicle to drive without having detail knowledge of the complexity of the parts. The driver can drive the whole vehicle treating like a single object. Similarly Operating System like Windows, UNIX provides abstraction to the user. The user can view his files and folders without knowing internal detail of Hard disk like the sector number, track number, cylinder number or head number. Operating System hides the truth about the disk hardware and presents a simple file-oriented interface. The class is a construct in object oriented programming for creating user-defined data for abstraction. When data and it operation are presented together, the construct is call ADT (Abstract Data Type). In OOP classes are used in creating ADT. For example, a student class can be made and can be available to be used in programs. The programmer can implement the class in creating objects and its manipulation without knowing its implementation. The program can use the function Sort\_name() to sort the names in alphabetical order without knowing whether the implementation uses bubble sort, merge sort, quick sort algorithms.

4.Encapsulation :- The mechanism of wrapping up of data and function into a single unit is called encapsulation. Because of encapsulation data and its manipulating function can be kept together. We can assume encapsulation as a protective wrapper that prevents the data being accessed by other code defined outside the wrapper. By making use of encapsulation we can easily achieve abstraction. The purpose of a class is to encapsulate complexity. Each data or function in a class can be marked as private or public. The public interface of a class represents everything that external users of the class may know about the data and function. The private function and data can only be accessed by code that is a member of a class. The code other than member of a class cannot access a private function or data. This insulation of data from direct access by the program is called data hiding. After hiding data by making them private, it then safe from accidental alteration.

5.Inheritance :- Inheritance is the process by which objects of one class acquire the characteristics of object of another class. We can use additional features to an existing class without modifying it. This is possible by deriving a new class (derived class) from the existing one (base class).This process of deriving a new class from the existing base class is called inheritance. It provides the concept of hierarchical classification. It allows the extension and reuse of existing code without having to rewrite the existing code. We naturally view the whole world is made up of objects. Many objects are related to each other in a hierarchical way, such as vehicle, four wheeler, and car. If we describe vehicle in an abstract way, the attributes may be such as color, number of seats etc. All vehicles have common behavioral aspect like; they move, accelerate, turn and stop. The more specific class of vehicle is four wheeler that acquires all features of class vehicle and has more specific attributes like engine number, chases number etc. The class vehicle is called base class (or super class) and class four wheeler is called derives class (or subclass).

6.Reusability :- Like library functions in procedural programming a class in Object Oriented Programming can be distributed for further use. In OOP, the concept of inheritance provides the idea of reusability. Once a class is completed and tested, it can be distributed for the development other programs too. The programmer can add new features or make some changes or can derive new classes from the existing class. This idea saves time and effort of a programmer. The testing of software will become easier as the already tested class should not be tested again. Suppose we have got a tested class Employee and we have to design a new class for Manager. The class Manager has all common features to class Employee. We can add some more features to class Manager using all features of class Employee. If a software company creates generic classes for one project then the company can use the same class and its extensions in the new project with less time, effort and investment.

7.Polymorphism :- Polymorphism means ‘having many forms’. The polymorphism allows different objects to respond to the same operation in different ways, the response being specific to the type of object. The different ways of using same function or operator depending on what they are operating on is called polymorphism. Example of polymorphism in OOP is operator overloading, function overloading. Still another type of polymorphism exist which is achieved at run time also called dynamic binding. For example operator symbol ‘+’ is used for arithmetic operation between two numbers, however by overloading (means given additional job) it can be used over Complex Object like currency that has Rs and Paisa as its attributes, complex number that has real part and imaginary part as attributes. By overloading same operator ‘+’ can be used for different purpose like concatenation of strings. When same function name is used in defining different function to operate on different data (type or number of data) then this feature of polymorphism is function overloading.

8.Dynamic binding :- The linking of a function call to the code to be executed in response to the call is called binding. There are two types of binding one is static binding( also called early binding) and another is dynamic binding( also called late binding). Function overloading and operator overloading construct in OOP are the examples of early binding. The early binding occurs at the compile time. This type of polymorphism occurring at compile time is called compile time polymorphism. Dynamic binding means that the code associated with a given function call is not known until the time of the call at run time. It is achieved at run time so called as run time polymorphism. Dynamic binding is possible only when we use inheritance and access the objects through pointers. If classes Circle, Box, and Triangle are derived from same function draw(). During the function call draw() through the pointer variable an appropriate function belonging to that class is involved.

9.Message Passing :- Procedural programming languages have function driven communication. That is a function is invoked for a piece of data. Object oriented language have message driven communication. A message is sent to an object. Communications among the objects are analogous to exchanging messages among people. An Object-Oriented program consists of set of objects that communicate with each other. Object communicates with each other by sending and receiving message (information). A message for an object is a request for execution of a procedure and therefore will invoke a function or procedure in receiving object that generate the desired result. Message passing involves specifying the name of the object name of the function (message) and the information (arguments to function) to be sent. In word, the message for an object is a request for the execution of a function belonging to an object which generates the desired result for the given argument.

**ii.** Integrated Development Environment

Dev-C++ is a full-featured Integrated Development Environment (IDE) for the C/C++ programming language. As similar IDEs, it offers to the programmer a simple and unified tool to edit, compile, link, and debug programs. It also provides support for the management of the files of a program in “projects” containing all the elements required to produce a final executable program.

Dev-C++ uses Mingw port of GCC (GNU Compiler Collection) as a compiler. It can creates native Win32 executables, either console or GUI, as well as DLLs and static libraries. Dev-C++ can also be used in combination with Cygwin or any other GCC based compiler. In this session, we will use Mingw --included in the default Dev-C++ distribution-- to create console C programs. Dev-C++ is a Free Software distributed under the terms of the GNU General Public License (GPL).

Dev-C++ features are:

- Support GCC based compilers (Mingw included)

- Integrated debugging (with GDB)

- Support for multiple languages (localization)

- Class Browser - Debug variable Browser

- Code Completion

- Function Listing

- Project Manager

- Customizable syntax highlighting editor

- Quickly create Windows, console, static libraries and DLL

- Support of templates for creating your own project types

- Makefile creation

- Edit and compile Resource files

- Tool Manager

- Print support

- Find and replace facilities

- Package manager, for easy installation of add-on libraries

**iii.** Project Introduction

Parking Management System is developed to support the customers and also the management of parking with a flexible and Hassle-free experience. The system enables the customer to make registrations and reservations for parking of their vehicles. To avoid the problem of road jamm, Parking Management System is useful for both the customers and the management. In this system all the events are managed by the Admin only. Admin adds, delete and update the information in the program. In the availability of parking slot of the vehicles, the customer can simply go to that parking place and park the vehicle thereby creating a hassle free experience.

# OBJECTIVE

The objective of PARKING MANAGEMENT STSTEM are listed below:

* To reduce heavy traffic congestion,
* To create more saleable space,
* To provide more security,
* To reduce the unwanted spaces for other purpose,
* To enhance the development of the country,
* To provide a systematic and flexible system,
* To increase revenue,

# PROGRAM CODE

#include<iostream>

#include<stdio.h>

#include<fstream>

#include<iomanip>

using namespace std;

class basicdetails{

private:

int hrs;

int min;

char name[20];

public:

int vehicle\_no;

void getdata()

{

cout<<"Enter the owner name : ";

cin>>name;

cout<<"Enter the vehicle number : ";

cin>>vehicle\_no;

cout<<"Arrival time (hrs:min) : ";//it is the exact time when the data was entered

cin>>hrs;

cout<<":";

cin>>min;

}

void showdata()

{

cout<<setw(33)<<"!!!Car Parking Details!!!\n";

cout<<"-----------------------------------------\n";

cout<<"Name"<<setw(20)<<" \t\t: "<<name<<endl;

cout<<"Vehicle number"<<setw(20)<<" \t: "<<vehicle\_no<<endl;

}

};

class twowheeler:public basicdetails{

int duration;

float total;

public:

void getdata()

{

cout<<"Estimated duratiom of stay (in hrs) : ";

cin>>duration;

}

void display()

{

float rate=0.25;

total=duration\*15;

cout<<"Total duration of stay"<<setw(12)<<" : "<<duration<<" hrs"<<endl;

cout<<"Parking charge"<<setw(20)<<" \t: Rs."<<total;

cout<<"\n-----------------------------------------\n";

}

}t1;

class fourwheeler:public basicdetails{

int duration;

float total;

public:

void getdata()

{

cout<<"total duration of parking:";

cin>>duration;

}

void display()

{

float rate=0.25;

total=duration\*15;

cout<<"Total duration of stay : "<<duration<<" hrs"<<endl;

cout<<"Parking charge : Rs."<<total;

cout<<"\n-----------------------------------------\n";

}

}f1;

void delete\_record(void);

void options(void);

inline re\_turn(void);

int main()

{

options();

}

void options()

{

int choice;

cout<<"\n\t\t\t\t\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\t\t\t\n"<<endl;

cout<<"\t\t\t\t\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\t\t\t\n"<<endl;

cout<<"\t\t\t\t1.Arrival of vehicle\t\t\t"<<endl;

cout<<"\t\t\t\t2.Total parking charges of all vehicles with details\t\t\t"<<endl;

cout<<"\t\t\t\t3.Departure of vehicles\t\t\t"<<endl;

cout<<"\t\t\t\t4.Exit program"<<endl;

cout<<"\n\t\t\t\t\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\n"<<endl;

cout<<"\t\t\t\t\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\t\t\t\n"<<endl;

cout<<"\t\t\t\tSelect your choice : ";

cin>>choice;

switch(choice)

{

case 1:

system("CLS");

int ch;

cout<<"Is it two wheeler or four wheeler (2/4) : ";

cin>>ch;

if(ch==2)//if it is two wheeler then the data is stored in twowheels.txt

{

ofstream fout;

fout.open("twowheels.txt",ios::binary|ios::out|ios::app);

t1.basicdetails::getdata();

t1.getdata();

fout.write((char\*)&t1,sizeof(twowheeler));

fout.close();

re\_turn();

}

else

{

ofstream fout;

fout.open("fourwheels.txt",ios::binary|ios::out|ios::app);

f1.basicdetails::getdata();

f1.getdata();

fout.write((char\*)&f1,sizeof(fourwheeler));

fout.close();

re\_turn();

}

break;

case 2:

system("CLS");

int d;

cout<<"press 2 for two wheller and 4 for four wheller:"<<endl;

cin>>d;

if(d==2)

{

ifstream fin;

fin.open("twowheels.txt",ios::in|ios::binary);

while(fin.read((char\*)&t1,sizeof(twowheeler)))

{

system("cls");

t1.showdata();

t1.display();

system("pause");

}

re\_turn();

}

else

{

ifstream fin;

fin.open("fourwheels.txt",ios::in|ios::binary);

while(fin.read((char\*)&t1,sizeof(fourwheeler)))

{

system("cls");

f1.showdata();

f1.display();

system("pause");

}

re\_turn();

}

break;

case 3:

delete\_record();

re\_turn();

break;

case 4:

exit(0);

default:

cout<<"\n!!!Invalid Choice!!!";

}

}

inline re\_turn()

{

char ch;

cout<<"do you want to return?(y/n) : ";

cin>>ch;

if(ch=='y'||ch=='Y')

{

system("cls");

options();

}

else

{

exit (0);

}

}

void delete\_record()

{

int n,d;

system("CLS");

cout<<"Enter the type of vehicle 2 wheeler or 4 wheeler (2/4) : ";

cin>>d;

cout<<"\n\n\t\tEnter the car number that has departed : ";

cin>>n;

if(d==2)

{

ifstream Fin\_two("twowheels.txt",ios::binary);

ofstream saved\_file("temp.txt", ios::out | ios::binary);

while(Fin\_two.read((char\*)&t1, sizeof(t1)))

{

if(t1.vehicle\_no!= n)

{

saved\_file.write((char\*)&t1, sizeof(t1));

}

}

cout<<"\n\n\t\t Parking Record Deleted";

Fin\_two.close();

saved\_file.close();

remove("twowheels.txt");

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

}

else

{

ifstream Fin\_two("fourwheels.txt",ios::binary);

ofstream saved\_file("temp.txt", ios::out | ios::binary);

while(Fin\_two.read((char\*)&t1, sizeof(t1)))

{

if(t1.vehicle\_no!= n)

{

saved\_file.write((char\*)&t1, sizeof(t1));

}

}

cout<<"\n\n\t\t Parking Record Deleted";

Fin\_two.close();

saved\_file.close();

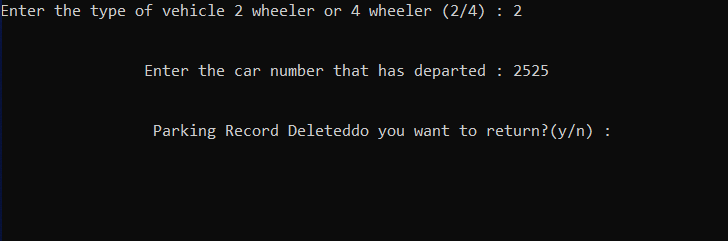
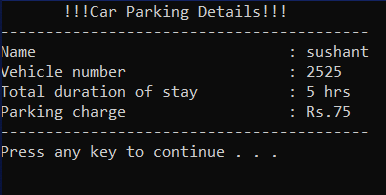
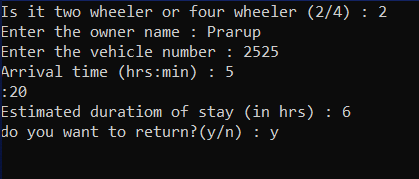
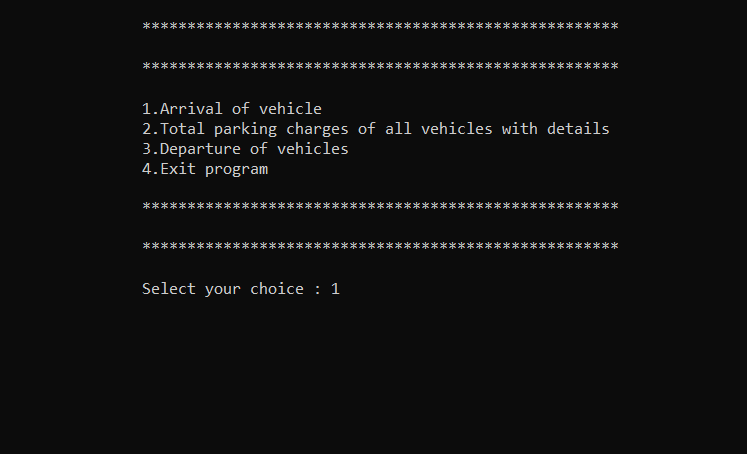
remove("fourwheels.txt");

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

}

}

## OUTPUT



# CONCLUSION

In this paper the development of the parking system with vehicle number plate recognition, parking lots status and guidance parking system and electronic billing system is successfully implemented. For the shake of the pollution free and green environment without any trafic congestion , the program is run successfully. The Parking Management System provides the better experience than before.

# REFERENCE

* <https://code-projects.org/> - Code Project
* Text Book - Object Oriented Programming