PROJECT TITLE: METRO RESERVATION SYSTEM

NAME : P.KAUSHIK

USN : 1NH17CS409

CHAPTER

INTRODUCTION

Introduction

The “Metro Reservation System” has been developed to override the problems prevailing in the practicing manual system. This application is supported to eliminate and, in some case, reduce the hardships faced by the existing system. More over this system is designed for the particular need of the company to carry out operations in a smooth and effective manner.

This system is reduced as much as much as possible to avoid errors while entering the data. It also provides error message while entering invalid data. No formal knowledge is needed for the user to use this system. Thus, by this all it proves it is user-friendly.

Metro reservation system as described can lead error free, secure, reliable and fast reservation system. It can assist the user to concentrate on their activities rather to concentrate on the record keeping. Thus, it will help organisation in better utilization or resources. Every organisation, whether big or small, has challenges to overcome and managing the data. Thus, this project is developed to help the people to save and reduce manpower.

CHAPTER

ANALYSIS AND DESIGN

Objective of the project

The objective of the project is to reduce man power, to automate the existing system. This project mainly aims at atomizing and making work easy. Through this project people can know the arrival of the train and information of the train. People can select more than one seat. People can save maximum time by just booking the tickets.

The Functionality of this system is:

- Provides the searching facilities based on various factors.
- It also manages the station details.
- Shows the information and description of the tickets booked.
- To increase efficiency of managing the metro.
- It deals with monitoring the ticket price.
- It manages the information of fare.

Requirement Specification

Hardware & Software Requirements:

Ram – 256MB

Compiler – Dev C++

Hard-disk – 1 GB

Operating System – Windows Xp, 8, 9,10.

CHAPTER

IMPLEMENTATION

Oops concept description

The prime purpose of C++ programming was to add object orientation to the C Programming language, which is in itself one of the most powerful programming languages. The core of the pure object-oriented programming is to create an object, in code, that has certain properties and methods. While designing C++ modules, we try to see whole world in the form of objects. For example, a car is an object which has certain properties such as colour, number of doors, etc., It also has certain methods such as accelerate, brake, and so on.

There are a few principle concepts that form the foundation of object-oriented Programming−

Classes and objects

Abstraction

Encapsulation

Polymorphism

Inheritance

Friend Function

Virtual Function

Dynamic Memory Allocation

Dynamic Binding

Classes and Objects

Object is the basic unit of object-oriented programming. That is both data and function that Operate on data are bundled as a unit called as object. When you define a class, you define a blueprint for an object. This doesn't actually define any data, but it does define what the class name means, that is, what an object of the class will consist of and what operations can be performed on such an object.

Abstraction

Data abstraction refers to, providing only essential information to the outside world and hiding their background details, i.e., to represent the needed information in program without presenting the details.

For example, a database system hides certain details of how data is stored and created and maintained. Similar way, C++ classes provides different methods to the outside world without giving internal detail about those methods and data.

Encapsulation

Encapsulation is placing the data and the functions that work on that data in the same place. While working with procedural languages, it is not always clear which functions work on which variables but object-oriented programming provides you framework to place the data and the relevant functions together in the same object.

Polymorphism

The ability to use an operator or function in different ways in other words giving different meaning or functions to the operators or functions is called polymorphism. Poly refers to many. That is a single function or an operator functioning in many ways different upon the usage is called polymorphism.

Overloading – The concept of overloading is also a branch of polymorphism. When the existing operator or function is made to operate on new data type, it is said to be overloaded.

There are two types in polymorphism:

Function Overloading Operator

Overloading

Function Overloading – Function Overloading means function having the same name but passing different arguments.

Operator Overloading – Operator Overloading means one operator performing different functions.

Inheritance

One of the most important concepts in object-oriented programming is that of inheritance. Inheritance allows us to define a class in terms of another class, which makes it easier to create and maintain an application. This also provides an opportunity to reuse the code functionality and fast implementation time. When creating a class, instead of writing completely new data members and member functions, the programmer can designate that the new class should inherit the members of an existing class. This existing class is called the base class, and the new class is referred to as the derived class.

Base class & derived class – A class can be derived from more than one classes, which means it can inherit data and functions from multiple base classes. To define a derived class, we use a class derivation list to specify the base class(es). A class derivation list names one or more base classes and has the form –

Where access-specifier is one of public, protected, or private, and base-class is the name of a previously defined class. If the access-specifier is not used, then it is private by default.

Access Control – A derived class can access all the non-private members of its base base class. Thus base-class members that should not be accessible to the member functions of derived classes should be declared private in the base class.

Types of Inheritance

Single Inheritance: In Single Inheritance, a class is allowed to inherit from only one class, i.e., one sub class is inherited by one base class only.

Multiple Inheritance: Multiple Inheritance is a feature of C++ where a class can inherit from more than one classes i.e., one sub class is inherited from more than one base classes.

Multilevel Inheritance: In this type of inheritance, a derived class is created from another derived class.

Hierarchical Inheritance: In this type of inheritance, more than one sub class are inherited from a single base class i.e., more than one derived class is created from a single base class.

Hybrid(Virtual) Inheritance: Hybrid inheritance is implemented by combining more than one type of inheritance. For example: Hierarchical inheritance and Multiple Inheritance.

Friend Class & Function

A friend class can access private and protected members of other class in which it is declared as friend. It is sometimes useful to allow a particular class to access private members of other class. Friend function: Like friend class, a friend function can be given special grant to access private and protected members. A friend function can be:

A method of another class

A global function

Following are some important points about friend functions and classes:

Friends should be used only for limited purpose. Too many functions or external classes are declared as friend of a class with protected or private data, it lessens the value of encapsulation of separate classes in object-oriented programming.

Friendship is not mutual. If a class A is friend of B, then B doesn’t become friend of a automatically.

Friendship is not inherited.

Virtual Function

A virtual function is a member function which is declared within base class and is redefined(overridden) by derived class. When you refer to a derived class object using a pointer or a reference to the base class, you can call a virtual function for that object and execute the derived class’s version of the function.

Virtual functions ensure that the correct function is called for an object,

Regardless of the type of reference used for function call.

They are mainly used to achieve Runtime Polymorphism.

Functions are declared with a virtual keyword in base class.

The resolving of function call is done at Run-time.

Rules for virtual function

They must be declared in public section of class.

Virtual functions cannot be static and also cannot be a friend function of another class. Virtual functions should be accessed using pointer or reference of base class type to achieve run time polymorphism.

The prototype of virtual functions should be same in base as well as derived class.

A class may have virtual destructor but it cannot have a virtual constructor.

Late binding (Runtime) is done in accordance with the content of pointer and an early binding is done accordance to the type of pointer.

Dynamic Memory Allocation

When the memory needed depends on user input, on this case, programs need to dynamically allocate the memory, for which the C++ language integrates the operators New and Delete.

Operator new and new[ ]

Dynamically memory is allocated suing operator new, new is followed by a data type specifier and, if a sequence of more than one element is required, the number of these within brackets [ ]. It returns a pointer to the beginning of the new block of memory allocated. Its syntax is:

pointer = new type pointer = new type

[number of elements]

The first expression is used to allocate memory to contain one single element of type ‘type’. The second one is used to allocate a block (an array) of elements of type ‘type’ where number of elements is an integer value representing the amount.

Operator delete and delete [ ]

In most cases, memory allocated dynamically is only needed during specific periods of time within a program, once it is no longer needed, it can be freed so that the memory becomes available again for other requests of dynamic memory. This is the purpose of operator delete, syntax is:

delete pointer; delete

[ ] pointer;

The first statement releases the memory of a single element allocated using new, and the second one releases the memory allocated for arrays of elements using new size in bracket ([ ]).

Dynamic Binding

Dynamic binding also called dynamic dispatch is the process of linking procedure call to a specific sequence of code at run-time. It means that the code to be execute for a specific procedure call is not known until run-time. Dynamic binding is also known as late-binding or run-time binding. Dynamic binding is an object-oriented programming concept and it is related with polymorphism and inheritance. Dynamic binding means that a block of code executed with reference to a procedure call is determined at run time.

CHAPTER

CONCLUSION

This project helps the travellers to book their seat, reserve the seats, check for the train timing etc which results in time saving for the people. In this busy schedule people don’t find time, rush to metro station miss train, people have to travel standing if they don’t get seats so it would be helpful if they reserve the seats. So, this project is just a sample project to implement Metro reservation. This project aims at helping people, saving time and reduce manual power. The further implementation of this project is to assign driver to train, online booking, displaying train details continuously with updated information, user login etc