**Project Report on**

HOTEL MANAGEMENT

**In Computer Science (C++)**

**Session 2013-14**

**Name of Student**

Shrikant Dubey

**Under the Guidance of**

N. Chandrashekhar

**For the Partial Fulfillment of**

AISSCE (2013-14), CBSE, New Delhi

**Submitted to**

Jawaharlal Nehru School

Habibganj, BHEL, Bhopal

CERTIFICATE

This is to certify that **Shrikant Dubey** of class 12th ‘A’ has successfully completed his project report in C++ on ‘**Hotel Management**’ under my guidance for the AISSCE as prescribed by CBSE in the year

2013-14.

|  |  |  |  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- | --- | --- | --- |
| Date | : | |  | | --- | |  | | / | |  | | --- | |  | | / | |  | | --- | |  | |

|  |  |
| --- | --- |
|  |  |
|  |  |
|  |  |
|  |  |
| \_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_ | \_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_ |
|  |  |
| Signature of Internal Examiner | Signature of External Examiner |

ACKNOWLEDGEMENT

I would like to thank my C++ teachers for all that they have taught me and their guidance for making this program. I would also like to thank my friends for all the extra help.

PREFACE

This file provides interaction about general **OOPs (Object Oriented Programming) concept** and **Database** **concept** to help understand the structure and mechanism of the program **Hotel Management**.

Each **function** of the program has been provided with **algorithm** and with **screenshots** so that readers can understand better, what is happening during the course of program usage.

The program does not hold well in all place of situation; thus **limitations** of the program have been listed at the end of the file.

TABLE OF CONTENT

|  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- |
|  |  |  |  |  | Page no. |  |
|  | 01. |  | Certificate |  | 2 |  |
|  | 02. |  | Acknowledgement |  | 3 |  |
|  | 03. |  | Preface |  | 4 |  |
|  | 04. |  | Object Oriented Programming (OOP) |  | 6 |  |
|  | 05. |  | Database concept |  | 8 |  |
|  | 06. |  | Documentation |  | 10 |  |
|  | 07. |  | Introduction |  | 13 |  |
|  | 08. |  | Algorithm |  | 16 |  |
|  | 09. |  | Coding |  | 22 |  |
|  | 10. |  | Screenshots |  | 46 |  |
|  | 11. |  | Limitations |  | 49 |  |
|  | 12. |  | Bibliography |  | 50 |  |

OBJECT ORIENTED PROGRAMMING

With the defeat of **procedural** oriented programming due to complexities and errors, object oriented programming was introduced with set of concepts to greatly decrease the complexities and cope with errors.

Object Oriented Programming lays emphasis on **objects** and provides a full set of operations for each **class**. Indeed it serves two most important purposes of a programming language :

|  |  |
| --- | --- |
| 1. | It provides a vehicle for the programmer to specify actions to be executed. |
|  |  |
| 2. | It provides a set of concepts for the programmer to use when thinking about what can be done. |

|  |  |  |
| --- | --- | --- |
| **procedural** | **:** | The programming approach that focuses on the procedures for the solution of a problem is known as procedural programming **paradigm**. This approach emphasizes on ‘doing’ rather than the ‘data’. |
|  |  |  |
| **object** | **:** | Object is an identifiable entity with some characteristics and behavior. |
|  |  |  |
| **class** | **:** | A class is a template, representing a group of objects that share common properties and relationships. |
|  |  |  |
| **paradigm** | **:** | Paradigm means organizing principle of a program. It is an approach to programming. |

|  |  |  |
| --- | --- | --- |
| **Modular programming** | **:** | Breaking down of a large program into smaller units called **functions** (sub-programs). The idea of breaking a program into functions can further be extended by grouping a number of functions together into a larger entity called a **module**. |
|  |  |  |
| **Module** | **:** | A set of related procedure with data they manipulate is called a module, |

Basic concepts of Object Oriented Programming (OOP) :

|  |  |  |
| --- | --- | --- |
| **Data Abstraction** | **:** | Abstraction refers to the act of representing essential features without including the background details or explanations. |
|  |  |  |
| **Data Encapsulation** | **:** | The wrapping up of data and operations (**functions** : that operates on the data) into a single unit (**class**) is known as data encapsulation. |
|  |  |  |
| **Modularity** | **:** | It is the property of a system that has been decomposed into a set of cohesive and loosely coupled modules. |
|  |  |  |
| **Inheritance** | **:** | It is the capability of one class of things to inherit capabilities or properties from another class. |
|  |  |  |
| **Polymorphism** | **:** | It is the ability of a data to be processed in more than one form. |

DATABASE CONCEPT

The collection of data, arranged in a systematic order in the system through some software, is known as a database.

Whenever a user wants to share some data with more than one place by the multiuser, we have to use database.

Storage of data in a \*.dat file is only a method to store data for a long time. When dealing with huge load of data kept stored in a ‘data file’ the most difficult task is ‘how to display the data’ so as to meet the user demands.. It is done by general form of representation called table.

A table consists of rows and columns. In database concept rows are called **tuples**, and columns are called **attributes**.

Consider a list of products in the form of a bill :-

|  |  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- | --- |
|  | NAME | ADDRESS | PH.NO | ADULTS | CHILDRENS | R.NO. | HOURS/DAYS |
| 1. | SD | BHEL | 123 | 2 | 0 | 1 | 9 DAY |
| 2. | KD | OM NAGAR | 456 | 1 | 0 | 2 | 2 HOUR |
| 3. | MR.C | WOW APARTMENT | 345 | 4 | 0 | 3 | 2 DAY |
| 4. | DJ | SHAHPURA | 101 | 2 | 0 | 4 | 5 HOUR |
| 5. | KP | 7 NO. | 786 | 2 | 2 | 5 | 4 DAY |

Here, NAME, ADDRESS, PH. NO., ADULTS, CHILDRENS, R. NO., HOURS, DAYS are called **attributes**.

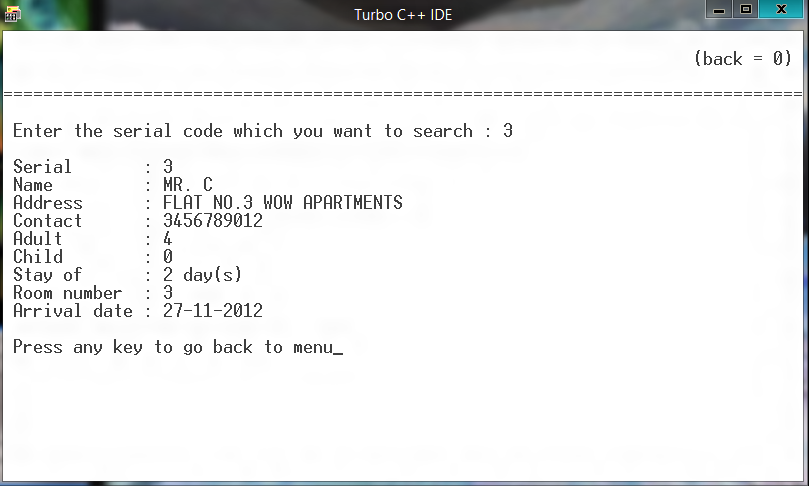
The row representing details of one customer is called **tuple**.

Collection of attributes is called **degree of table**.

Collection of tuples is called **cardinality of table**.

In the program available in this file, ‘Hotel Management’ the Data model of the Database is taken to be **Relational data model**.

|  |  |  |
| --- | --- | --- |
| **Relational data model** | **:** | In the given table, the relationship between the attributes is known as relational data model. |
|  |  |  |
| Example | **:** | When we enter the serial no., the screen displays the data connected with that serial no.. |



DOCUMENTATION

|  |  |  |
| --- | --- | --- |
| **Name of the project** | **:** | Hotel Management |
|  |  |  |
| **Purpose of the project** | **:** | Store records of customers staying in a hotel and prepare a bill according to the number of days/hours, no. of people, |
|  |  |  |
| **Name of the source file** | **:** | Hotel Management |
|  |  |  |
| **Name of data files** | **:** | HOTEL.dat |
|  |  |  |
| **Size of the source file** | **:** | 43.8 KB |
|  |  |  |
| **Operating system used** | **:** | Windows 8 Pro |
|  |  |  |
| **C++ version used** | **:** | Turbo C++ 3.0 |

List of **member functions** and their uses :

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
|  | **Name** | **Parameter** | **Return Type** | **Usage** |
|  |  |  |  |  |
| 01. | Last\_serial | Void | Integer | Returns the greatest serial number in HOTEL.dat |
|  |  |  |  |  |
| 02. | Find\_serial | Integer | Integer | Returns truth value if the given serial number is found in the HOTEL.dat |
|  |  |  |  |  |
| 03. | Find\_room | Integer | Integer | Returns truth value if the given room number is found in HOTEL.dat |
|  |  |  |  |  |
| 04. | Display\_entry | Integer | Void | Puts the given serial number and its details on the screen. |
|  |  |  |  |  |
| 05. | Sort\_list | Void | Void | Sorts the entire HOTEL.dat |
|  |  |  |  |  |
| 06. | Date\_1 | Void | Void | Stores arrival date of customer in the data member |
|  |  |  |  |  |
| 07. | Date\_2 | Void | Void | Store leaving date of customer in the data member |
|  |  |  |  |  |
| 08. | Deleter | Integer | Void | Deletes the given serial code from HOTEL.dat |
|  |  |  |  |  |
| 09. | Write\_function | Void | Void | Accesses to write or append in the HOTEL.dat |

List of **member functions** and their uses :

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
|  | **Name** | **Parameter** | **Return Type** | **Usage** |
|  |  |  |  |  |
| 10. | Read\_function | Void | Void | Accesses to read from HOTEL.dat |
|  |  |  |  |  |
| 11. | Search\_function | Void | Void | Accesses to search from HOTEL.dat |
|  |  |  |  |  |
| 12. | Insert\_function | Void | Void | Accesses to insert in HOTEL.dat |
|  |  |  |  |  |
| 13. | Modify\_function | Void | Void | Accesses to modify an entry in the HOTEL.dat |
|  |  |  |  |  |
| 14. | Delete\_function | Void | Void | Accesses to delete an entry from HOTEL.dat |
|  |  |  |  |  |
| 15. | Billing | Void | void | Accesses to prepare bill of customer and save the changes in HOTEL.dat |

INTRODUCTION

Library functions used in the program :

|  |  |  |
| --- | --- | --- |
| **conio.h** | **:** | Stands for ‘console input & output’. Most of its functions process without buffer and without echo. |
|  |  |  |
| **ctype.h** | **:** | Its functions process on a single character at a time and check according to the given condition |
|  |  |  |
| **fstream.h** | **:** | Declares standard file input and output stream. |
|  |  |  |
| **process.h** | **:** | It contains function declarations and macros used in working with threads and processes. |
|  |  |  |
| **stdio.h** | **:** | It defines types and macros needed for the standard input-output package. |
|  |  |  |
| **stdlib.h** | **:** | It declares several commonly used routines, like conversion routines, search routines, sort routines, etc. |
|  |  |  |
| **string.h** | **:** | It declares several string manipulation and memory manipulation routines. |
|  |  |  |
| **time.h** | **:** | This header file contains definitions of functions to get and manipulate date and time information. |

The program operates on one \*.dat files :

|  |  |  |
| --- | --- | --- |
| **HOTEL.dat** | : | Used to store and handle the details of the customer staying in the hotel. |

|  |  |  |
| --- | --- | --- |
| Name of the class | : | **Hotel** |
|  |  |  |
| Number of private data members | : | 9 |
|  |  |  |
| Number of public data members | : | 0 |
|  |  |  |
| Number of private member functions | : | 5 |
|  |  |  |
| Number of public member functions | : | 10 |

List of **data members** and their uses :

|  |  |  |  |
| --- | --- | --- | --- |
|  | **Variable** | **Data type** | **Usage** |
|  |  |  |  |
| 1. | Serial | Integer | Used to store customer’s serial no. for the identification of individual customer. |
|  |  |  |  |
| 2. | Name | Character | Used to store the name of individual customer |
|  |  |  |  |
| 3. | Address | Character | Used to store residential address of individual customer |
|  |  |  |  |
| 4. | Contact | Integer | Used to store the contact no. of the individual customer |
|  |  |  |  |
| 5. | Adults | Integer | Used to store the no. of adults in the group or individual. |

List of **data members** and their uses :

|  |  |  |  |
| --- | --- | --- | --- |
|  | **Variable** | **Data type** | **Usage** |
|  |  |  |  |
| 6. | Child | Integer | Used to store the no. of children in the group. |
|  |  |  |  |
| 7. | Room | Integer | Used to store the room no. issued to the customer. |
|  |  |  |  |
| 8. | Days | Integer | Used to store the no. of days customer have to stay in the hotel. |
|  |  |  |  |
| 9. | Hours | Integer | Used to store the no. of hours the customer have to stay in the hotel. |

ALGORITHM

|  |  |
| --- | --- |
| 1. | Algorithm for **writing** data in the data file : |
|  |  |
|  | |  |  | | --- | --- | | 1. | Define input-output stream to open data file. | |  |  | | 2. | Open data file while including ios::out.  stream.open(“\*.dat”,ios::out); | |  |  | | 3. | Input values to data members with respect to an object. | |  |  | | 4. | Use write() function with the stream, object and sizeof(class).  stream.write((char\*)&object,sizeof(class)); | |  |  | | 5. | Close the stream. stream.close(); | |

|  |  |
| --- | --- |
| 2. | Algorithm for **reading** data from the data file : |
|  |  |
|  | |  |  | | --- | --- | | 1. | Define input-output stream to open data file. | |  |  | | 2. | Open data file while including ios::in.  stream.open(“\*.dat”,ios::in); | |  |  | | 3. | Set pointer to the beginning of the file.  stream.seekg(0,ios::beg); | |  |  | | 4. | Until end of file eof(), repeat step 3. | |  |  | | 5. | Use read() function with the stream, object and sizeof(class).  stream.read((char\*)&object,sizeof(class)); | |  |  | | 6. | Close the stream. stream.close(); | |

|  |  |
| --- | --- |
| 3. | Algorithm to **append** data in the data file : |
|  |  |
|  | |  |  | | --- | --- | | 1. | Define file stream to open data file. | |  |  | | 2. | Open data file while including ios::out & ios::app.  stream.open(“\*.dat”,ios::out|ios::app); | |  |  | | 3. | Input values to data members with respect to an object. | |  |  | | 4. | Use write() function with the stream, object and sizeof(class).  stream.write((char\*)&object,sizeof(class)); | |  |  | | 5. | Close the stream. stream.close(); | |

|  |  |
| --- | --- |
| 4. | Algorithm to **search** data in the data file : |
|  |  |
|  | |  |  | | --- | --- | | 1. | Define file stream to open data file. | |  |  | | 2. | Open data file while including ios::in.  stream.open(“\*.dat”,ios::in); | |  |  | | 3. | Set pointer to the beginning of the file.  stream.seekg(0,ios::beg); | |  |  | | 4. | Input the serial number of the record to be searched.  cin>>s; | |  |  | | 5. | Until end of file eof(), repeat step 6. | |

|  |  |  |  |  |  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- |
|  | |  |  | | --- | --- | | 6. | while(!stream.eof())  stream.read((char\*)&object,sizeof(class)); | |  |  | | 7. | if(s==(serial code))  object.class::display(); | |  |  | | 8. | Close the stream.  stream.close(); | |

|  |  |
| --- | --- |
| 5. | Algorithm to **insert** data in the data file : |
|  |  |
|  | |  |  | | --- | --- | | 1. | Define file stream to open data file. | |  |  | | 2. | Open data file using ios::out & ios::app.  stream.open(“\*.dat”,ios::out|ios::app); | |  |  | | 3. | Input the details of the record to be inserted. | |  |  | | 4. | Use write() function with the stream, object and sizeof(class).  stream.write((char\*)&object,sizeof(class)); | |  |  | | 5. | Close the stream.  stream.close(); | |  |  | | 6. | Sort the data file using Selection sort. | |  |  | |  | |  |  | | --- | --- | | 1. | for(int i=1 to i=n-1) | |  |  | | 2. | small=list\_of\_i;  position=I; | | |

|  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- |
|  | |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  | | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | |  | |  |  | | --- | --- | | 3. | for(int j=1 to j=n)  repeat step 4. | |  |  | | 4. | if(list\_of\_j > small)  then small=list\_of\_j; | |  |  | | 5. | temp=list\_of\_i | |  |  | | 6. | list\_of\_i=list\_of\_j; | |  |  | | 7. | list\_position=temp; | | |

|  |  |
| --- | --- |
| 6. | Algorithm to **delete** data from the data file : |
|  |  |
|  | |  |  | | --- | --- | | 1. | Define file stream to open data file. | |  |  | | 2. | Open data file using ios::in.  stream.open(“\*.dat”,ios::in); | |  |  | | 3. | Input the serial number of the record to be searched.  cin>>s; | |  |  | | 4. | Until end of file eof(), repeat step 5. | |  |  | | 5. | while(!stream.eof())  stream.read((char\*)&object,sizeof(class)); | |  |  | | 6. | Define another file stream temp to open TEMPORARY data file. | |

|  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- |
|  | |  |  | | --- | --- | | 7. | Open TEMPORARY data file using ios::out.  stream.open(“\*.dat”,ios::out); | |  |  | | 8. | Until serial code not equal to s, repeat step 9. | |  |  | | 9. | if(serial!=s)  temp.write((char\*)&object,sizeof(class)); | |  |  | | 10. | Close the file streams.  stream.close();  temp.close(); | |  |  | | 11. | Remove original data file.  Rename temporary data file to original data file.  remove(“\*.dat”);  rename(“TEMPORARY.dat”,”\*.dat”); | |

|  |  |
| --- | --- |
| 7. | Algorithm to **modify** data in data file : |
|  |  |
|  | |  |  | | --- | --- | | 1. | Define file stream to open data file. | |  |  | | 2. | Open data file using ios::in.  stream.open(“\*.dat”,ios::in); | |  |  | | 3. | Input the serial number of the record to be searched.  cin>>s; | |  |  | | 4. | Until end of file eof(), repeat step 5. | |  |  | | 5. | while(!stream.eof())  stream.read((char\*)&object,sizeof(class)); | |

|  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- |
|  | |  |  | | --- | --- | | 6. | if(serial==s)  Input new details. | |  |  | | 7. | Close stream(); | |  |  | | 8. | Open data file using ios::out & ios::ate.  stream.open(“\*.dat”,ios::out|ios::ate); | |  |  | | 9. | Move pointer to the specific serial  int pointer=(s-1)\*sieof(class); | |  |  | | 10. | seekp(pointer); | |  |  | | 11. | stream.write((char\*)&object,sizeof(class)); | |  |  | | 12. | Close the stream. stream.close(); | |

SCREENSHOTS

|  |
| --- |
| C:\Users\Kenny\Desktop\Desktop\Hotel Management\1.jpg |
|  |
| C:\Users\Kenny\Desktop\Desktop\Hotel Management\2.jpg |

|  |
| --- |
| C:\Users\Kenny\Desktop\Desktop\Hotel Management\3.jpg |
|  |
| C:\Users\Kenny\Desktop\Desktop\Hotel Management\4.jpg |

|  |
| --- |
| C:\Users\Kenny\Desktop\Desktop\Hotel Management\5.jpg |
|  |
| C:\Users\Kenny\Desktop\Desktop\Hotel Management\6.jpg |

|  |
| --- |
| C:\Users\Kenny\Desktop\Desktop\Hotel Management\7.jpg |
|  |
| C:\Users\Kenny\Desktop\Desktop\Hotel Management\8.jpg |

|  |
| --- |
| C:\Users\Kenny\Desktop\Desktop\Hotel Management\9.jpg |
|  |
|  |

LIMITATIONS

1. Large quantity of data cannot be stored in the data file through the means of this program.
2. The program is only but a satisfactory and is imperfect & has less of a user interface.
3. The program has no password protection on its database and anyone can access it.
4. The program may sometimes malfunction.

BIBLIOGRAPHY

|  |
| --- |
| **System details** : |
|  |
| |  |  |  | | --- | --- | --- | | Operating system used | : | Microsoft Windows 8 Pro | | TurboC version | : | 3.0 | | File prepared on | : | MS Office Word Processor 2013 | | Printer | : | HP | |
|  |
| **Resources** : |
|  |
| Sumita Arora, class XI, 6th edition; Sumita Arora, class XII, 9th edition |