Certified to be the bonafide work done by

Mr. **ANKIT MISHRA** of class **XII – A1**

during the year **2014-2015**

Date in the subject of **COMPUTER SCIENCE**.

Date:

Registration No. :

Signature of Internal Examiner:

I would like to express my sincere gratitude to my Computer

science mentor **Mr. Shwetank Kakkar ,** for his vital

support, guidance and encouragement - without which this

project would not have come forth.

Student Signature:

Ankit Mishra

**TOPIC PAGE NO.**

* **Introduction To C++ 6-8**
* **Introduction to Telephone 10-11**

**Directory Management**

* **Hardware and Software 13**

**Requirements**

* **Limitations and Future 15**

**Scope**

* **Header Files and Functions 17-19**
* **Flow Chart 21-25**
* **Coding 27-65**
* **Input and Output 67-71**
* **Bibliography 72**

**Resource Requirement** :

* **Software Used** : 1. Windows XP

2. Turbo C++

3. Notepad

4. MS Word

* **Hardware Used**: 1. Pentium III onwards

2. 128 MB RAM

3. 40GB HDD

**TELEPHONE DIRECTORY MANAGEMENT**

The telephone directory maintenance system is designed to insure a smooth allotment of telephone to the employee of a particular department and query the telephone directory whenever required.

**CURRENT SYSTEM**

Currently telephones are been allotted to employees at random on a first come first served basis. There is no proper tracking of who is allotted which telephone number. Any number of telephones gets allotted to the employees.

LIMITATION OF THE CURRENT SYSTEM

Because of the current system of allotment of telephone numbers to the employees at random, the same telephone number at a time gets allotted to more than one employee. There is no proper record of how many phones are allotted to each department which results in tracing of employees being difficult. There is no mechanism to find out which department a particular employee belongs to.

**PROPOSED SYSTEM**

The proposed system is being designed to maintain the details of employees address,pincode,telephone numbers and all their details through an application in a file. Unique record id are generated for any new employee/new company and new telephone number.

**OBJECTIVES:**

The system design will meet the below objectives

* Unique number generation for the new employee/new telephone allotment.
* Easy retrieval of telephone details based on employee names.
* Easy retrieval of telephone details based on the record id number.
* Deletion of the record id deletes all the information belonging to that employee.
* Employee location details can be modified whenever the employee gets transferred.

**SCOPE AND BOUNDARY:**

The project is designed for

* Record maintenance
  + Add records
  + Delete records
* Employee Maintenance
  + Employee addition
  + Employee modification
* Telephone Directory maintenance
  + Add telephone number
  + Enquiry by record id
  + Enquiry by Employee name
  + Enquiry by telephone no.

**LIMITATIONS OF THE PROJECT**

There are several limitations of the project. The project is subdivided into three sub-modules.

1. Record maintenance system.
2. Employee maintenance system.
3. Telephone directory maintenance system.

There are several limitations in the module. Some of they are:

* User can store maximum of 2500 employee records.
* All the constraints are not checked properly.

**Future Scope:**

The future scope of this project is immense. If the small difficulties and problems can be eliminated, the software will be of great help in the real life scenario.

* [**www.icbse.com**](http://www.icbse.com)
* [**www.google.com**](http://www.google.com)
* **Class 12th C.S. book-Sumita Arora**

**INTRODUCTION TO C++**

**Data File Handling:**

**File.** The information / data stored under a specific name on a storage device, is called a file.

**Stream.** It refers to a sequence of bytes.

**Text file.** It is a file that stores information in ASCII characters. In text files, each line of text is terminated with a special character known as EOL (End of Line) character or delimiter character. When this EOL character is read or written, certain internal translations take place.

**Binary file.** It is a file that contains information in the same format as it is held in memory. In binary files, no delimiters are used for a line and no translations occur here.

## Classes For File Stream Operation:

**ofstream**: Stream class to write on files  
**ifstream**: Stream class to read from files  
**fstream:** Stream class to both read and write from/to files.

### Input And Output Operation:

**put() and get() function**  
the function put() writes a single character to the associated stream. Similarly, the function get() reads a single character form the associated stream.  
example :  
file.get(ch);  
file.put(ch);

**write() and read() function**  
write() and read() functions write and read blocks of binary data.

internal stream pointers that point to the reading or writing locations within a stream can be manipulated using the following member functions:

|  |  |
| --- | --- |
| seekg() | moves get pointer(input) to a specified location |
| seekp() | moves put pointer (output) to a specified location |
| tellg() | gives the current position of the get pointer |
| tellp() | gives the current position of the put pointer. |

|  |  |
| --- | --- |
| **File mode parameter** | **Meaning** |
| ios::app | Append to end of file |
| ios::ate | go to end of file on opening |
| ios::binary | file open in binary mode |
| ios::in | open file for reading only |
| ios::out | open file for writing only |
| ios::nocreate | open fails if the file does not exist |
| ios::noreplace | open fails if the file already exist |
| ios::trunc | delete the contents of the file if it exist. |
|  |  |

**Linked Lists:**

A linked list is a technique of creating a list with the ability to add, delete, or retrieve items. Additional operations can also be provided to a more elaborate list such as finding an item, deleting an item, etc.

**The Beginning of a Linked List:**

If you create an [array-based list](http://www.functionx.com/cpp/articles/arraybasedlist.htm), you can start by declaring an array member variable that would hold the items and each item can be located by an index that is assigned to it when the item is added to the list. When  you do this, you must provide an estimate of the maximum number of items that will be allowed in the list. Without good planning, the dimension you specify could be too high or too low but C++ doesn't allow you to declare an array without a dimension if you are not initializing the array. This means that, when you create an array-based list, you must also specify the maximum number of items that the list can hold.

A linked list is a list that can grow or shrink as the [user wishes](http://www.functionx.com/cpp/articles/linkedlist.htm). This means that, when creating the list, you don't need to predict the maximum number of items that will be added to the list. To use this flexibility, the items must be managed through pointers. Because the list would use a member that defines its item, you can declare a member variable that is conform to the intended items.

When a list starts, it is empty or at least it must be considered like that, before any item is added to it. To specify this, you should declare a primary member variable. Although you can call it anything, it is usually called Head. This member can be made private if you don't intend to access it outside of the class. If you want clients of the class to access it, you can make it public. Although this member is declared as a pointer and it marks the beginning of the list, you should not allocate memory for it in the constructor. Its memory would be managed when it is accessed. Therefore, you can simply initialize is as NULL.

Once a list exists, the user can explorer it. One of the operations performed on items is to locate and retrieve one. To do this, you can declare a method that takes as argument as index. The method would examine the argument with regards to the number of items in the list to make sure the argument's value is in the range of [current items](http://www.functionx.com/cpp/articles/linkedlist.htm) of the list. If the number is too low or too high, the method can return null or 0. If the number is in the range, the method can return the item at that position.

One of the operations hardly performed on a list is to find one. This is because if you ask a list to locate a particular item, you must provide as much information as possible. Probably the most expedient way you can do this is to completely define an item and pass it to the list. Only if the item is found in the list would it be recognized.

**TELEPHONE**

**DIRECTORY**

**MANAGEMENT**

**CODING**

**INTRODUCTION**

**TO C++**

**SCREEN SHOTS**