



Green University of Bangladesh

Department of Computer Science and Engineering (CSE)

Faculty of Sciences and Engineering

Semester: (Fall, Year:2021), B.Sc. in CSE (Day)

Course Title: Algorithm Lab

Course Code: CSE 206

Section: DD201

Lab Project Name: PhoneBook Application.

Student Details

Name		ID
1.	MD Sakhawat Hossain Rabbi	201002311

Submission Date : 29-12-2021

Course Teacher's Name : Ms. Sultana Umme Habiba

[For Teachers use only: Don't Write Anything inside this box]

Lab Project Status

Marks:

Signature:

Comments:

Date:

Table of Contents:

<u>1. Introduction</u>	<u>03</u>
<u>2. Aim of the project:.....</u>	<u>03</u>
<u>3. Design/Development/Implementation of the Project</u>	<u>04</u>
<u>4.These functions, under the class phonebook, perform the following operations:</u>	<u>05</u>
<u>5. Simulation Procedure/ Code:</u>	<u>07</u>
<u>6. Output.....</u>	<u>13</u>
<u>6. Advantages & Disadvantages of phonebook Management System:... </u>	<u>15</u>
<u>7. Future scope:</u>	<u>15</u>

Introduction:

The phonebook is a very simple C++ project that can help you understand the basic concepts of functions, file handling, Linear Search Algorithm and selection sort . This program will teach you how to add, list, change or edit, search and remove data from/to a file. Adding new records, listing them, editing and updating them, looking for saved contacts, and removing phonebook records are simple Functions that make up the main menu of this Phonebook program.



Personal information, such as name, phone number, is requested when you add a record to your phonebook. These records can then be updated, entered, searched, and deleted. I've used a lot of functions in this mini-project. These functions are easy to understand since their name means just their respective operations.

Aim of the project:

Create a "Contact Phonebook" framework using C programming. This software is very useful nowadays to store full information under a single contact number. The software also has options for removing and changing the contact number entered

Design/Development/Implementation of the Project:

In this project, I have used Quick sort and Linear Search.

For Name and Number in display section I have used Quick sort for sort the Name and Number list.

Quick sort Algorithm:

Like Merge Sort, QuickSort is a Divide and Conquer algorithm. It picks an element as pivot and partitions the given array around the picked pivot. There are many different versions of quickSort that pick pivot in different ways.

QuickSort is one of the most efficient sorting algorithms and is based on the splitting of an array into smaller ones. The name comes from the fact that quicksort is capable of sorting a list of data elements significantly faster than any of the common sorting algorithms. And like Merge sort, Quick sort also falls into the category of divide and conquer approach of problem-solving methodology.

Quicksort algorithm works in the following way

- Taking the analogical view in perspective, consider a situation where one had to sort the papers bearing the names of the students, by name. One might use the approach as follows –
 - Select a splitting value, say L. The splitting value is also known as **Pivot**.
 - Divide the stack of papers into two. A-L and M-Z. It is not necessary that the piles should be equal.
 - Repeat the above two steps with the A-L pile, splitting it into its significant two halves. And M-Z pile, split into its halves. The process is repeated until the piles are small enough to be sorted easily.
 - Ultimately, the smaller piles can be placed one on top of the other to produce a fully sorted and ordered set of papers.
- The approach used here is recursion at each split to get to the single-element array.
- At every split, the pile was divided and then the same approach was used for the smaller piles.
- Due to these features, quicksort is also called a partition exchange sort

So we can say,

1. Quick sort Always pick first element as pivot.
2. Always pick last element as pivot
3. Pick a random element as pivot.
4. Pick median as pivot.

The key process in quickSort is partition(). Target of partitions is, given an array and an element x of array as pivot, put x at its correct position in sorted array and put all smaller elements (smaller than x) before x, and put all greater elements (greater than x) after x. All this should be done in linear time.

Pseudo Code for recursive QuickSort function :

```
/* low --> Starting index, high --> Ending index */
quickSort(arr[], low, high)
{
    if (low < high)
    {
        /* pi is partitioning index, arr[p] is now
           at right place */
        pi = partition(arr, low, high);

        quickSort(arr, low, pi - 1); // Before pi
        quickSort(arr, pi + 1, high); // After pi
    }
}
```

Header files:

```
#include<iostream>
#include<windows.h>
```

These functions, under the class phonebook, perform the following operations:

1. Add Contacts()- It allow user to enter new Contacts.
2. Display All Contacts()- It displays all the entries .
3. Search by Number(char *nm)- It searches for added record by number.
4. Search by Name()- It searches for added record by phone name.
5. Update()- This function is used to modify added records.
6. Delete ()- It deletes a contact from file.
7. Delete All()- It deletes all the contact list.
8. Number of Contacts() – it shows the total Number of contact list.
9. Exit()- It close the menu

1. Add Contacts	2. Display All Contacts	3. Search by Number	4. Search by Name	5. Update	6. Delete	7. Delete All	8. Number of contacts	9. Exit
-----------------------	----------------------------------	------------------------------	----------------------------	--------------	--------------	---------------------	--------------------------------	------------

Simulation Procedure/ Code:

```
#include<iostream>
#include<windows.h>
using namespace std;
void start();
int manu();
int k=0;
string name[100];
string no[100];
void swap(string* namea, string* nameb,string* numc,string* numd)
{
    string t = *namea;
    *namea = *nameb;
    *nameb = t;
    t = *numc;
    *numc = *numd;
    *numd = t;
}

int partition (string arr[], int low, int high)
{
    string pivot = arr[high];    // pivot
    int i = (low - 1);

    for (int j = low; j <= high- 1; j++)
    {
        //if current element is smaller than pivot, increment the low element
        //swap elements at i and j
        if (arr[j] <= pivot)
        {
            i++;    // increment index of smaller element
            swap(&arr[i], &arr[j],&no[i], &no[j]);

        }
    }
    swap(&arr[i + 1], &arr[high],&no[i + 1], &no[high]);
    return (i + 1);
}
```

```

}

//quicksort algorithm
void quickSort(string arr[], int low, int high)
{
    if (low < high)
    {
        //partition the array
        int pivot = partition(arr, low, high);

        //sort the sub arrays independently
        quickSort(arr, low, pivot - 1);
        quickSort(arr, pivot + 1, high);
    }
}

int main()
{
    start();

    int check=0;
    int Total_contacts=0;

    do
    {

        // Add check=manu();contacts
        if(check==1)
        {
            cout<<"\t\t\t\t\t Name :";
            cin>>name[k];
            cout<<"\t\t\t\t\t Phone Number :";
            cin>>no[k];
            k++;
            Total_contacts++;
        }
    }
}

```



```

//Diplay contacts
else if (check==2)
{

    cout<<"===="<<Total_contacts;
    quickSort( name, 0, Total_contacts-1);
    //cout << k << " \t " << Total_contacts << endl;
    int check2=0;
    for(int i=0; i<100;i++)
    {
        if(name[i]!='\0")
            cout<<"\t\t\t\t Name : "<<name[i]<< "    Phone
: "<<no[i]<<endl;

            check2++;
    }
    if(check2==0)
    {
        cout<<"\t\t\t\t";
    }
}
//Search by Number
else if(check==3)
{
    string temp;
    cout<<"\t\t\t\t\tNumber : ";
    cin>>temp;
    int check2=0;
    for(int i=0;i<100;i++)
    {
        if(temp==no[i])
        {
            cout<<"\t\t\t\t\tNumber is Found\n";
            cout<<"\t\t\t\t\tName : "<<name[i]<<"    Phone
: "<<no[i]<<endl;

            check2++;
        }
    }
    if(check2==0)
    {

```

```

        cout<<"\t\t\t\t This Number is Not found in your
contact list\n";
    }
}
//Search By Name
else if(check==4)
{
    string temp;
    cout<<"\t\t\t\tName : ";
    cin>>temp;
    int check2=0;
    for(int i=0;i<100;i++)
    {
        if(temp==name[i])
        {
            cout<<"\t\t\t\tName is Found\n";
            cout<<"\t\t\t\tName : "<<name[i]<<"    Phone
: "<<no[i]<<endl;
            check2++;
        }
    }
    if(check2==0)
    {
        cout<<"\t\t\t\t This name is Not found in your contact
list\n";
    }
}
// Update
else if(check==5)
{
    string temp,temp2,temp3;
    cout<<"\t\t\t\tName : ";
    cin>>temp;
    int check2=0;
    for(int i=0;i<100;i++)
    {
        if(temp==name[i])
        {
            cout<<"\t\t\t\tNew Name : ";

```

```

        cin>>temp2;
        cout<<"\t\t\t\tNew Number : ";
        cin>>temp3;
        name[i]=temp2;
        no[i]=temp3;
        check2++;
        cout<<"\t\t\t\tUpdated Successfully ";
    }

}

if(check2==0)
{
    cout<<"\t\t\t\t This name is Not found in your contact
list\n";
}
}
// delete
else if(check==6)
{
    string temp;
    cout<<"\t\t\t\tFor Delete Enter Name : ";
    cin>>temp;
    int check2=0;
    for(int i=0;i<100;i++)
    {
        if(temp==name[i])
        {
            cout<<"\t\t\t\tDeleted Successfully\n";
            cout<<"\t\t\t\tName : "<<name[i]<<"    Phone
: "<<no[i]<<endl;

            name[i]="\0";
            no[i]="\0";
            check2++;
            Total_contacts--;
        }
    }
    if(check2==0)
    {
        cout<<"\t\t\t\t This name is Not found in your contact
list\n";

```

```

    }
}
// delete All
else if(check==7)
{
    cout<<"\t\t\t\t All Deleted Successfully\n";
    for(int i=0;i<k;i++)
    {
        name[i]="\0";
        no[i]="\0";
    }
    k=0;
    Total_contacts=0;
}
// Diplay numbers of contacts
else if(check==8)
{
    cout<<"\t\t\t\tTotal Number of contact list are :
"<<Total_contacts<<endl;

}

check=manu();

}while(check!=9);

}
int manu()
{
    cout<<"\n\n\n\n\n";
    cout<<"\t\t\t\t-----\n";
    cout<<"\t\t\t\t-----\n";
    cout<<"\t\t\t\t      PHONE BOOK APPLICATION          \n";
    cout<<"\t\t\t\t-----\n";
    cout<<"\t\t\t\t                                \n";
    cout<<"\t\t\t\t [1] Add Contacts                \n";
    cout<<"\t\t\t\t [2] Display All Contacts         \n";
    cout<<"\t\t\t\t [3] Search by Number             \n";
    cout<<"\t\t\t\t [4] Search by Name               \n";

```

```
cout<<"\t\t\t\t\t [5] Update \n";  
cout<<"\t\t\t\t\t [6] Delete \n";  
cout<<"\t\t\t\t\t [7] Delete All \n";  
cout<<"\t\t\t\t\t [8] Number of contacts \n";  
cout<<"\t\t\t\t\t \n";  
cout<<"\t\t\t\t\t-----\n";  
cout<<"\t\t\t\t\t [9] Exit \n";  
cout<<"\t\t\t\t\t-----\n";  
  
int a;  
cin>>a;  
system("cls");  
return a;  
}  
  
void start()  
{  
    system("Color 02");  
    cout<<"\n\n\n\n\n\n\n\n\n";  
    cout<<"\t\t\t\t\t-----\n";  
    cout<<"\t\t\t\t\t-----\n";  
    cout<<"\t\t\t\t\tPHONE BOOK APPLICATION\n";  
    cout<<"\t\t\t\t\t-----\n\n";  
    cout<<"\t\t\t\t\tLoading ";  
    char x = 272;  
    for(int i=0; i<35; i++)  
    {  
        cout<<x;  
        if(i<10)  
            Sleep(300);  
        if(i>=10 && i<20)  
            Sleep(150);  
        if(i>=10)  
            Sleep(25);  
    }  
    system("cls");  
}
```

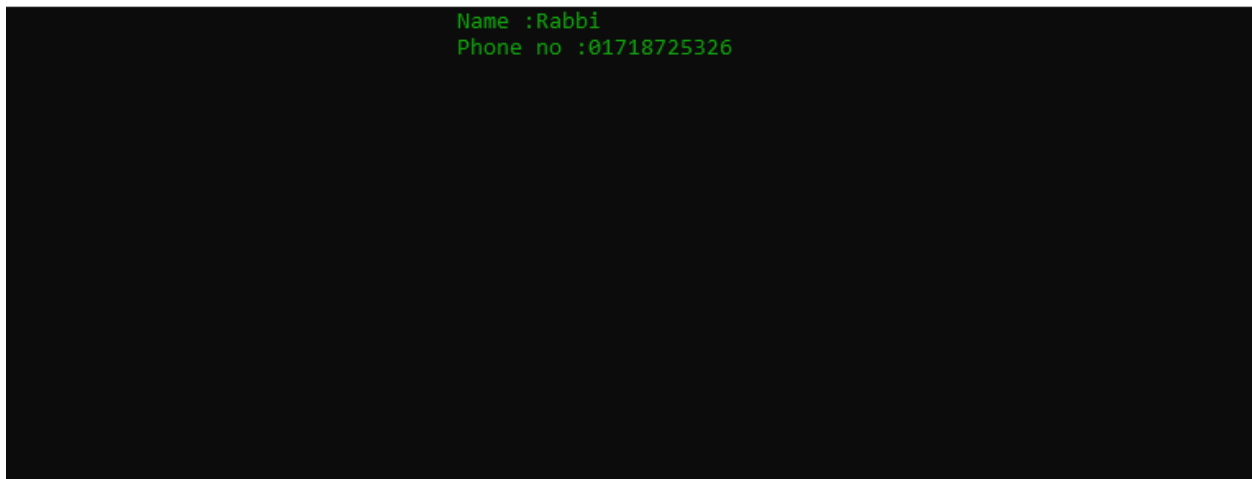
Output:



```
-----  
-----  
PHONE BOOK APPLICATION  
-----  
-----  
[1] Add Contacts  
[2] Display All Contacts  
[3] Search by Number  
[4] Search by Name  
[5] Update  
[6] delete  
[7] Delete All  
[8] Number of contacts  
-----  
[9] Exit  
-----
```

Fig: Menu

downloads\Phone Book.exe"



```
Name :Rabbi  
Phone no :01718725326
```

Fig; Add Contact

Phone Book.exe"

```
Name :Rabbi      Phone :01718725326
Name :pontu     Phone :02

-----
PHONE BOOK APPLICATION
-----

[1] Add Contacts
[2] Display All Contacts
[3] Search by Number
[4] Search by Name
[5] Update
[6] delete
[7] Delete All
[8] Number of contacts

-----
[9] Exit
-----
```

Fig; Display Contact

```
For Delete Enter Name : s
Deleted Successfully
Name : s      Phone : 34

-----
PHONE BOOK APPLICATION
-----

[1] Add Contacts
[2] Display All Contacts
[3] Search by Number
[4] Search by Name
[5] Update
[6] Delete
[7] Delete All
[8] Number of contacts

-----
[9] Exit
-----
```

Fig; Deleted Contact

```
Number : 4
This Number is Not found in your contact list

-----
PHONE BOOK APPLICATION
-----

[1] Add Contacts
[2] Display All Contacts
[3] Search by Number
[4] Search by Name
[5] Update
[6] Delete
[7] Delete All
[8] Number of contacts

-----
[9] Exit
-----
```

Fig: Search contact by Number

Advantages & Disadvantages of phonebook Management System:

It is simple for the user to store complete about his or her contact. It's simple for the user to simply check his appropriate contact number by entering the contact name. Often it's hard to store more contacts. It is also difficult to store contacts with two or more contact numbers.

Future scope:

The scope of android app development is not limited to domains such as e-commerce, education, social media, gaming, banking & finance etc. Freelancing is another aspect where an individual can utilize his android app development skills and make personal profits. Android OS vs iOS.