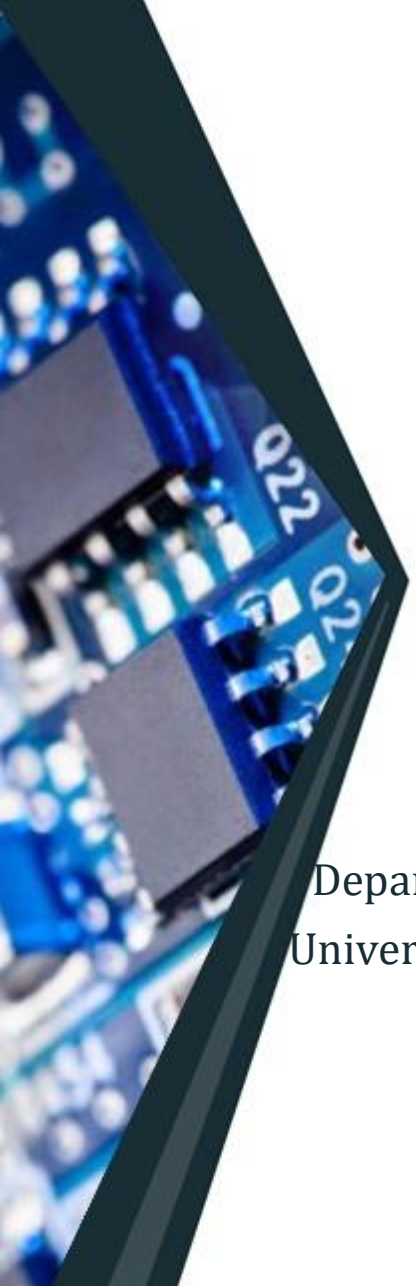
FINAL REPORT
[DSA]
PHONE-BOOK



NAME:

JAMSHID ALI

REGISTRATION NO:

18PWCSE1654

LAB INSTRUCTOR:

Dr. KHURAM SHEHZAD

Department of Computer Systems Engineering

University of Engineering & Technology, Peshawar

Contents

INTRODUCTION

Phonebook System is a simple console application without graphics which is developed in C++ platform. The main aim of this project is to provide an application that will be helpful to record any person's details just like in mobile phones. Besides, it helps to add, list, modify, and delete phonebook related records.

Basically, this is an easy and fundamental level of tiny projects for learning purposes. Moreover, the user can also modify this scheme according to their requirements. They can also create a project that is a perfect advance level project.

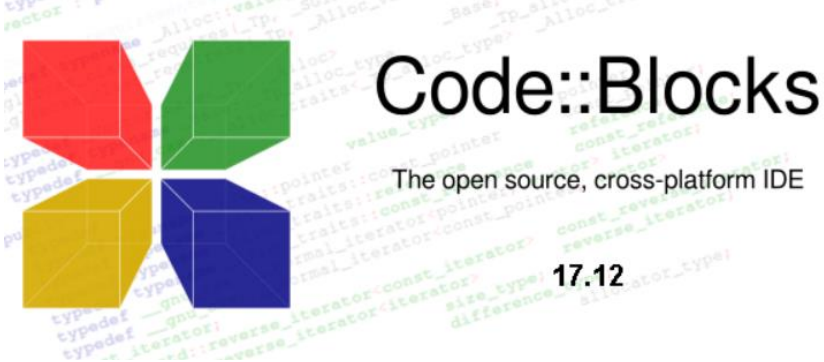
Adding new records, listing them, modifying them and updating, search for contacts saved, and deleting the phonebook records are the basic functions which make up the main menu of this Phonebook application (as shown in the main menu screenshot below).

Personal information such as name, phone number and email address are asked while adding a record into the Phonebook. These records can then be modified, listed, searched for and removed.

PROJECT ARCHITECTURE

a) IDE

I am using Code Blocks 17.12 as an IDE in this Project.



b) Language

I am using C++ as programming language in this project

a. **Libraries**
   1. #include <iostream>
   2. #include<cstring>

b. **Classes**
   1. class dnode
   2. class dlist

c. **Functions**
   1. void insert();
   2. void sort();
   3. void deletecontact(char n[20]);
   4. void searchbyname(char p[20]);
   5. void searchbynumber(char no[30]);
   6. void searchbygmail(char g[20]);
   7. void accept();
   8. void display();
   9. void update(char ch[10]);

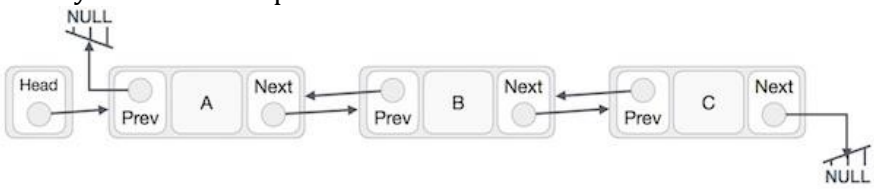
c) ALGORITHMS

I am using Double linked list Algorithm because while searching first element but the current status of pointer is in between middle and first element so it should traverse backward because then it will take less time.

Doubly Linked List is a variation of Linked list in which navigation is possible in both ways, either forward and backward easily as compared to Single Linked List. Following are the important terms to understand the concept of doubly linked list.

- **Link** – Each link of a linked list can store a data called an element.
- **Next** – Each link of a linked list contains a link to the next link called Next.
- **Prev** – Each link of a linked list contains a link to the previous link called Prev.
- **LinkedList** – A Linked List contains the connection link to the first link called First and to the last link called Last.

  Doubly Linked List Representation



  As per the above illustration, following are the important points to be considered.

- Doubly Linked List contains a link element called first and last.
- Each link carries a data field(s) and two link fields called next and prev.
- Each link is linked with its next link using its next link.
- Each link is linked with its previous link using its previous link.
- The last link carries a link as null to mark the end of the list.

1. **BUBBLE SORT**

   In this project I am using bubble sort for sorting the contact list by name.

2. **LINEAR SEARCH**

   In this project I am using linear search for searching contact in the contact list. There are three type of search
   By Name
   By Number
   By Gmail

3. **DELETE CONTACT**

   To delete a contact I am using linear delete in this project.

d) MAIN OPERATIONS

1) DISPLAY YOUR PHONE BOOK
2) INSERT NEW CONTACT
3) UPDATE DETAILS ON EXISTING CONTACT
4) DELETE CONTACT
5) SEARCH CONTACT LIST

TIME COMPLEXITIES

Time complexity of an algorithm signifies the total time required by the program to run till its completion.
The time complexity of algorithms is most commonly expressed using the **big O notation**. It's an asymptotic notation to represent the time complexity.
Time Complexity is most commonly estimated by counting the number of elementary steps performed by any algorithm to finish execution.
And since the algorithm's performance may vary with different types of input data, hence for an algorithm we usually use the **worst-case Time complexity** of an algorithm because that is the maximum time taken for any input size.
The time complexities for the operations are given bewlow:

a) SEARCH

a. Lower Bound

Lower Bound complexity or Best case complexity for searching in this project is:
$\Omega(1)$

b. Upper bound

Upper Bound complexity or Worst case complexity for searching in this project is:
O(n)

c. Tight bound

Tight Bound complexity or average case complexity for searching in this project is:
$\Theta(n)$

b) SORT

a. Lower Bound

Lower Bound complexity or Best case complexity for sorting in this project is:
$\Omega(n)$

b. Upper bound

Upper Bound complexity or Worst case complexity for sorting in this project is:
O(n*log n)

c. **Tight bound**

Tight Bound complexity or average case complexity for searching in this project is:
$\Theta(n*\log n)$

c) DELETE

a. **Lower Bound**

Lower Bound complexity or Best case complexity to delete contact in this project is:
$\Omega(1)$

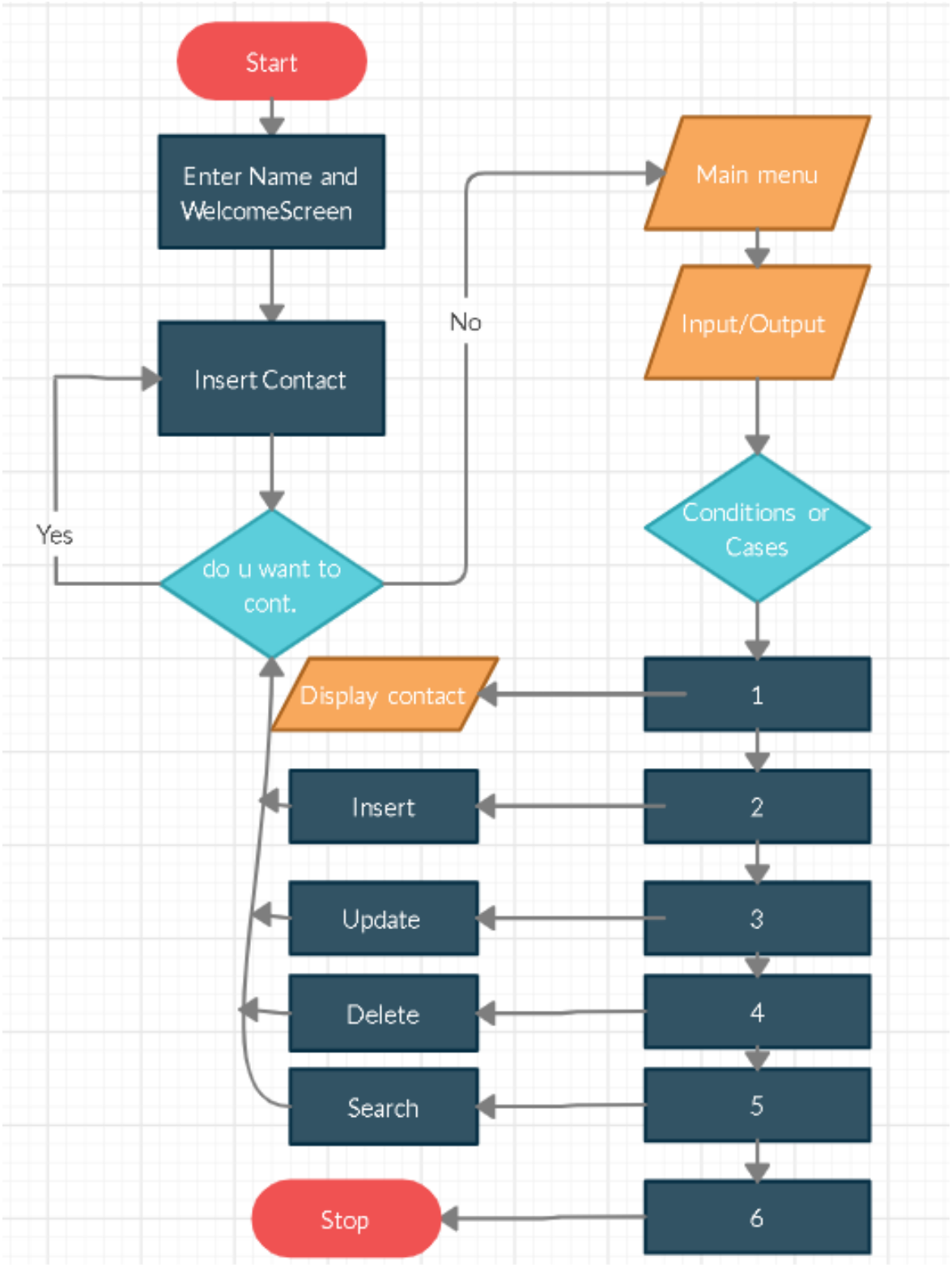
b. **Upper bound**

Upper Bound complexity or Worst case complexity to delete contact in this project is:
O(n)

c. **Tight bound**

Tight Bound complexity or average case complexity to delete contact in this project is:
$\Theta(n)$

FLOWCHART


Start
Enter Name and WelcomeScreen
Insert Contact
do u want to cont.
Yes
No
Main menu
Input/Output
Conditions or Cases
1
2
3
4
5
6
Display contact
Insert
Update
Delete
Search
Stop

CONSOLE OUTPUT