Table of contents

[CHAPTER 1: INTRODUCTION 2](#_Toc394144249)

[CHAPTER 2: OBJECTIVES 3](#_Toc394144250)

[CHAPTER 3: FEATURES 4](#_Toc394144251)

CHAPTER 4: CLASS DIAGRAMS……………………….....……………………5

CHAPTER 5: [DESCRIPTIONS OF THE FUNCTIONS: 6](#_Toc394144252)

CHAPTER 6:[SOURCE CODE OF THE PROGRAM 8](#_Toc394144253)

CHAPTER 7:[SAMPLE OUTPUT 23](#_Toc394144254)

CHAPTER 8:[LIMITATIONS AND ENHANCEMENTS 27](#_Toc394144257)

CHAPTER 9:[CONCLUSION 28](#_Toc394144258)

CHAPTER 1: INTRODUCTION

Object oriented programming offers new and powerful way to cope with complexity instead of viewing a program as a series of steps to be carried out, it views a group of objects that have certain properties and can take certain action. An object-oriented program typically consists of a number of objects, which communicate with each other by calling one another’s functions. This is called *sending a message* to the object. This kind of relation is provided with the help of communication between two objects and this communication is done through information called *message*. In addition, object-oriented programming supports *encapsulation*, *abstraction*, *inheritance*, and *polymorphism* to write programs efficiently. Examples of object-oriented languages include *Simula*, *Smalltalk*, *C++*, *Python*, *C#*, *Visual Basic .NET* and *Java* etc.

C++ is an object oriented programming language. It was developed by Bjarne Stroustrup at AT&T Bell Lab USA. It was developed on the base of C programming language and the first added part to C is concept of ***Class***. Hence originally this language was named “C with class”. Later name was changed to C++ on the basis of idea of increment operator available in C. since it is the augmented or incremented version of C, it is named C++.

Banking Management System keeps the day by day tally record as a complete  
banking. It can keep the information of Account type, account opening form, Deposit,  
Withdrawal, and Searching the transaction, Transaction report, Individual account  
opening form, Group Account. The exciting part of this project is; it displays Transaction  
reports, Statistical Summary of Account type and Interest Information.  
  
A computer based management system is designed to handle all the primary information required to calculate monthly statements of customer account which include monthly statement of any month. Separate database is maintained to handle all the details required for the correct statement calculation and generation. This project intends to introduce more friendliness in the various activities such as record updation, maintenance and searching. The searching of record has been made quite simple as all the details of the customer can be obtained by simple keying in the identification or account number of that customer can be obtained by simply keying in the identification or account number of that customer. Similarly, record maintenance and updation can also be accomplish by using the account number with all the details being automatically generated. These details are also being promptly automatically updated in the master file thus keeping the record absolutely up-to-date.

The entire information has maintained in the database or files and whoever wants to retrieve can’t retrieve, only authorization user can retrieve the necessary information which can be easily accessible from file.

# CHAPTER 2: OBJECTIVES

A bank management system is designed to handle all the primary information required to calculate monthly statements of customer account which include monthly statements of any months. This project intends to introduce more user friendly in various activities such as record updation, maintenance and searching. The searching of record in simple way as the details of the customer can be obtained by simply typing the account number of that customer. Similarly, the record maintenance and updation can be accomplished by using the account number with all the details being automatically generated.

The main objectives of our project is providing the different types of customer facility, and to find out the actual customer service etc.

* It should fulfil almost all the process requirements of any bank.
* It should increase the productivity of bank by utilizing the working hours more and more, within the minimum manpower.
* It should allow only authorized user to access various function and processed available in the system.
* It should locate any A/C wanted by the user.
* It should reduce clerical work as most of the work is done by computer.
* It must provide greater speed & reduce time consumption.

This project includes the entire upgrade feature required for the computerization banking system. This system is very easy to use, so that any user can use without getting pre-knowledge about this. It is very user friendly and meets some of the daily working process requirements.

# CHAPTER 3: FEATURES

The following are the features of the program that has been created:

1. The program is divided into two user modes: **USER** and **ADMIN**. The **USER** mode is for the customers and the **ADMIN** mode is for the bank operators.

The **USER** mode provides some basic functions such as creating a bank account, depositing an amount, withdrawing some amount and viewing the customer’s account information.

The **ADMIN** mode provides the operator with some basic functions. With this mode the user can view all the records of all the customers that have deposited the money in the bank and the total volume of cash collected

.

1. Allows the customers to create their accounts, deposit money into their account and withdraw some money from the same account keeping a specific minimum balance. The users can create an account, deposit money in that account, modify the account and withdraw money from that account via an interactive character line input. The users can also delete the accounts that are no longer needed by the use of the same interactive character line input.
2. This program provides the basic form of security to the banks records via the use of passwords. Before entering the ADMIN mode, the operator must provide the program with a password. If the password matches, then the access is granted else the access is denied. This basic security system protects the sensitive data stored in the program from unauthorized access.
3. This program uses file handling rather than storing the data temporarily in RAM. The input information is stored in files that makes it possible to access them every time the program is run.

Due to the simplicity of the program, this program uses only two files to store the data. One file “**try1.txt**” stores all the accounting information- the user’s name, his account number, account type, his initial balance, etc. Another file “**deposit.txt**” stores the amount of money that the customers deposit.

**CHAPTER 4: CLASS DIAGRAM**

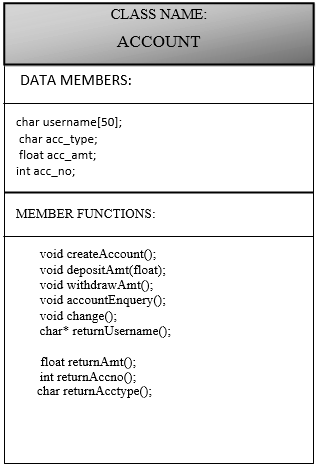


Figure 1: CLASS DIAGRAM

The above is a class diagram of the class **account** used in the program. This class is a simple class that uses no inheritance. The private section is the data members and the member functions are the functions that can access this private data.

# CHAPTER 5: DESCRIPTIONS OF THE FUNCTIONS:

A brief description of the functions used in this program

**MEMBER FUNCTIONS**

**void createAccount();**

This function creates a new account.

**void depositAmt(float);**

Function to deposit an amont to money in the user's account

**void withdrawAmt();**

Function to withdraw an amount

**void accountEnquery();**

Function to display complete information of the user's account

**void change();**

Function to modify the contents of a user's account

**char\* returnUsername()**

Function that returns the name of the user;

**float returnAmt()**

Function to return the amount present in the account

**int returnAccno()**

Function to return the account number

**char returnAcctype()**

Function to return the type of account of the user either savings or fixed

**GLOBAL FUNCTIONS**

**void write\_account();**

A function to write user account information to a file

**void display\_records();**

A function to display the records of a specfic user

**void delete\_account();**

A function to delete the records of the user

**void withdraw\_amount();**

A function to withdraw an amount from the user's account

**void deposit\_amount();**

A function to deposit an amount into the user's account

**void modify\_account();**

A function to modify the records of a specific user

**void display\_all\_records();**

An admin level function that can be used to display all the records of all the users

**void total\_cash();**

An admin level function that shows the total amount of cash that has been collected

**void startMenu();**

A function that calls the start menu of the program

**void user();**

A function that calls the user account

**void admin();**

A function that calls the administrator account

**void bank\_box()**

Displays the initial box with the bank’s name

**void decorate();**

A function that prints a series of asterieks(\*) as required

# SOURCE CODE OF THE PROGRAM

//\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*

//HEADER FILES USED IN THE PROGRAM

//\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*

#include<iostream>

#include<fstream>

#include<string.h>

#include<iomanip>

#include<conio.h>

#include<stdlib.h>

#include<Windows.h>

//\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*

//\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*

using namespace std;

//\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\* //CLASSES USED IN THE PROGRAM

//\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*

class account

{

char username[50];

char acc\_type;

float acc\_amt;

int acc\_no;

public:

//\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*

//MEMBER FUNCTION DECLERATIONS

//\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*

void createAccount(); // FUNCTION TO CREATE A NEW ACCOUNT

void depositAmt(float); // FUNCTION TO DEPOSIT AN AMONT TO MONEY IN THE USER’S ACCOUNT

void withdrawAmt(float); //FUNCTION TO WITHDRAW AN AMOUNT

void accountEnquery(); //FUNCTION TO DISPLAY COMPLETE INFORMATION OF the user account

void change(); //FUNCTION TO MODIFY THE CONTENTS OF A USER'S ACCOUNT

char\* returnUsername() {return username;} //A FUNCTION TO RETURN THE NAME OF THE CUSTOMER

float returnAmt() {return acc\_amt;} //FUNCTION TO RETURN THE AMOUNT PRESENT IN THE ACCOUNT

int returnAccno() {return acc\_no;} //FUNCTION TO RETURN THE ACCOUNT NUMBER

char returnAcctype(){return acc\_type;} //FUNCTION TO RETURN THE TYPE OF ACCOUNT OF THE USER EITHER SAVINGS OR FIXED

};

//\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*

//MEMBER FUNCTION DEFINATIONS

//\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*

void account::createAccount() //BEGIN FUNCTION createAccout()

{

cout<<"enter your name "; cin>>username;

cout<<"enter the account number ";cin>>acc\_no;

cout<<"\n\nFixed account minimun Rs.5000 "<<endl

<<"Savings account minimum Rs. 1000 "<<endl;

cout<<"enter the account type(S for Saving and F for fixed) "; cin>>acc\_type;

aa: //LINE LABEL

cout<<"enter the amount to be deposited "; cin>>acc\_amt;

acc\_type=toupper(acc\_type);

if(acc\_type=='S'&&acc\_amt<1000)

{

cout<<"invalid entry "<<endl;

goto aa;

}

if(acc\_type=='F'&&acc\_amt<5000)

{

cout<<"invalid entry "<<endl;

goto aa;

}

} // END FUNCTION createAccount()

void account::withdrawAmt(float num) //BEGIN FUNCTION withdrawAmt()

{

acc\_amt=acc\_amt-num;

} //END FUNCTION withdrawAmt()

void account::accountEnquery() //BEGIN FUNCTION acountEnquery()

{

cout<<"Name: "<<username<<endl

<<"Account no.: "<<acc\_no<<endl

<<"Account type: "<<acc\_type<<endl

<<"Account amount: "<<acc\_amt<<endl;

} //END FUNCION accountEnquery()

void account::depositAmt(float num)

{

acc\_amt=acc\_amt+num;

}

void account::change()

{

cout<<"enter the account number ";cin>>acc\_no;

cout<<"enter new name "; cin>>username;

cout<<"\n\nFixed account minimun Rs.5000 "<<endl

<<"Savings account minimum Rs. 1000 "<<endl;

cout<<"enter the new account type(S for Saving and F for fixed) "; cin>>acc\_type;

aa: //LINE LABEL

cout<<"enter the amount to be deposited "; cin>>acc\_amt;

acc\_type=toupper(acc\_type);

if(acc\_type=='S'&&acc\_amt<1000)

{

cout<<"invalid entry "<<endl;

goto aa;

}

if(acc\_type=='F'&&acc\_amt<5000)

{

cout<<"invalid entry "<<endl;

goto aa;

}

}

//\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*

//END MEMBER FUNCTION DEFINATIONS

//\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*

//\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*

//FUCTIONS RELATED TO FILE(FUNCTION DECLERATIONS)

// \*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*

void write\_account(); //A FUNCTION TO WRITE USER ACCOUNT INFORMATION TO A FILE

void display\_records(); //A FUNCTION TO DISPLAY THE RECORDS OF A SPECFIC USER

void modify\_account(); // A FUNCTION TO MODIFY THE RECORDS OF A SPECIFIC USER

void delete\_account(); // A FUNCTION TO DELETE THE RECORDS OF THE USER

void withdraw\_amount(); //A FUNCTION TO WITHDRAW AN AMOUNT FROM THE USER'S ACCOUNT

void deposit\_amount(); //A FUNCTION TO DEPOSIT AN AMOUNT INTO THE USER'S ACCOUNT

void display\_all\_records(); //AN ADMIN LEVEL FUNCTION THAT CAN BE USED

//TO DISPLAY ALL THE RECORDS OF ALL THE USERS

void total\_cash(); //AN ADMIN LEVEL FUNCTION THAT SHOWS THE TOTAL

// AMOUNT OF CASH THAT HAS BEEN COLLECTED

// \*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\* //END FUNCTIONS RELATED TO FILES

// \*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*

// \*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\* //FUCTIONS TO CALL USER AND ADMIN ACCOUNTS

// \*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*

void startMenu(); // A FUNCTION THAT CALLS THE START MENU

void user(); // A FUNCTION THAT CALLS THE USER ACCOUNT

void admin(); // A FUNCTION THAT CALLS THE ADMINSTRATOR ACCOUNT

//\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\* //END FUMCTION DECLERATIONS

// \*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*

// \*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*

//FUNCTIONS RELATED TO DESIGN

//\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*void decorate(int); // A FUNCTION THAT PRINTS A SERIES OF ASTERIEKS(\*)

void bank\_box(); // A FUNCTION THAT DISPLAYS THE NAME OF THE BANK

// \*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*

//FUNCTIONS RELATED TO DESIGN

// \*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*

void write\_account()

{

fstream file;

account ac;

file.open("try1.txt",ios::out|ios::app|ios::binary); //OPENS A FILE NAMED "try1.txt" IN WRITE AND AND APPEND MODE

ac.createAccount(); //CALLS THE createAccount OF THE CLASS account

file.write((char\*)&ac,sizeof(ac));

file.close();

}

void deposit\_amount()

{

account ac;

fstream file;

ofstream outf;

float tempamt;

int tempaccno,found=0;

cout<<"enter the account number "; cin>>tempaccno;

file.open("try1.txt",ios::binary|ios::in|ios::out);

outf.open("deposit.txt",ios::binary);

while(file.read((char\*)&ac, sizeof(ac))&&found==0)

{

if(ac.returnAccno()==tempaccno)

{

ac.accountEnquery();

cout<<"\n\n\n";

cout<<"TO DEPOSIT AMOUNT\n\n";

cout<<"enter the amount to be deposited "; cin>>tempamt;

outf<<tempamt;

outf<<ac.returnUsername();

ac.depositAmt(tempamt);

cout<<"\n\n\n AFTER DEPOSIT\n\n";

ac.accountEnquery();

found=1;

}

}

file.close();

}

void withdraw\_amount()

{

account ac;

fstream file;

float tempamt,balance;

int found=0;

file.open("try1.txt",ios::out|ios::in|ios::app|ios::binary);

if(!file)

cout<<"the file cannot be opened... "<<endl;

while(file.read((char\*)&ac,sizeof(ac)))

{

cout<<"enter the amount to be withdrawn "; cin>>tempamt;

balance=ac.returnAmt()-tempamt;

if((balance<500&&ac.returnAccno()=='S')||balance<1000&&ac.returnAccno()=='F')

{

cout<<"Insufficient balance...\n";

}

else

ac.withdrawAmt(tempamt);

int pos=(-1)\* sizeof(ac);

file.seekp(pos,ios::cur);

file.write((char \*) &ac, sizeof(account));

cout<<"\n\n\t Record Updated";

found=1;

}

file.close();

if(found==0)

cout<<"\n\n Record Not Found ";

}

void modify\_account()

{

int found=0,tempaccno;

account ac;

fstream file;

file.open("try1.txt",ios::binary|ios::in|ios::out);

if(!file)

{

cout<<"file could not be open !! Press any Key...";

}

cout<<"enter the account number "; cin>>tempaccno;

while(file.read((char \*)&ac, sizeof(ac)) && found==0)

{

if(ac.returnAccno()==tempaccno)

{

ac.accountEnquery();

cout<<"\n\nEnter The New Details of account"<<endl;

ac.change();

int pos=(-1)\*sizeof(ac);

file.seekp(pos,ios::cur);

file.write((char \*) &ac, sizeof(ac));

cout<<"\n\n\t Record Updated";

found=1;

}

}

file.close();

if(found==0)

cout<<"\n\n Record Not Found ";

}

void display\_records()

{

int tempaccno;

account ac;

int flag=0;

fstream file;

file.open("try1.txt",ios::in|ios::binary);

if(!file)

{

cout<<"the file could not be opened "<<endl;

}

cout<<"enter the account number "; cin>>tempaccno;

cout<<"balance details "<<endl<<endl;

while(file.read((char\*)&ac,sizeof(ac)))

{

if(tempaccno==ac.returnAccno())

{

ac.accountEnquery();

flag=1;

}

}

file.close();

if(flag==0)

cout<<"Record not found....."<<endl;

}

void display\_all\_records()

{

fstream file;

account ac;

file.open("try1.txt",ios::in|ios::binary);

system("cls");

while(file.read((char\*)&ac,sizeof(ac)))

{

ac.accountEnquery();

cout<<"\*\*\*\*\*\*\*\*\n";

}

file.close();

}

void delete\_account()

{

account ac;

ifstream inf;

ofstream outf;

int tempaccno;

inf.open("try1.txt",ios::binary);

if(!inf)

{

cout<<"File could not be open !! Press any Key...";

}

cout<<"enter the account number to be deleted "; cin>>tempaccno;

outf.open("Temp.txt",ios::binary);

inf.seekg(0,ios::beg);

while(inf.read((char\*)&ac,sizeof(ac)))

{

if(ac.returnAccno()!=tempaccno)

{

outf.write((char \*) &ac, sizeof(ac));

}

}

inf.close();

outf.close();

remove("try1.txt");

rename("Temp.txt","try1.txt");

cout<<"\n\n\tRecord Deleted ..";

}

void total\_cash()

{

ifstream inf;

account ac;

float sum=0,amount;

inf.open("deposit.txt",ios::binary);

inf.seekg(0,ios::beg);

while(inf)

{

inf>>amount;

sum=sum+amount;

inf>>ac.returnUsername();

}

char name[50];

strcpy(name,ac.returnUsername());

cout<<"the total cash deposited is "<<sum<<endl;

cout<<"the name is "<<name<<endl;

}

void startMenu()

{

char ch;

do

{

system("cls");

cout<<"\n\t\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*MENU\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\n\n";

cout<<"\t1. USER \n\n"

<<"\t2. ADMIN \n\n";

cout<<"enter your choice "; cin>>ch;

system("cls");

switch(ch)

{

case '1':

user();

break;

case '2':

admin();

break;

case '3':

exit(0);

break;

default:

cout<<"invalid choice \n";

cout<<"\a";

}

}while(ch!='4');

}

void user()

{

char ch;

do

{

system("cls");

cout<<"\n\n\n\t\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*MENU\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\n\n\n";

cout<<"\t1. Create Account \n\n"

<<"\t2. Display Records \n\n"

<<"\t3. Deposit Amount \n\n"

<<"\t4. Withdraw Amount \n\n"

<<"\t5. Modify Account \n\n"

<<"\t6. Delete Account \n\n"

<<"\t7. Exit \n\n"

<<"\t8. Go back to main menu\n\n";

cout<<"enter your choice "; cin>>ch;

system("cls");

switch(ch)

{

case '1':

system("cls");

write\_account();

break;

case '2':

system("cls");

display\_records();

break;

case '3':

system("cls");

deposit\_amount();

break;

case '4':

system("cls");

withdraw\_amount();

break;

case '5':

system("cls");

modify\_account();

break;

case '6':

system("cls");

delete\_account();

break;

case '7':

exit(0); break;

case '8':

system("cls");

startMenu();

break;

default:

cout<<"invalid choice \n"<<'\a';

}

system("pause");

}while(ch!='8');

}

void admin()

{

char ch,choice;

char password[8];

int inc=0, counter=0;

cout<<"you have three chances to enter the password...\n"

<<"the password must be 8 characters long \n";

do

{

while(inc!=8)

{

ch=getch();

if(ch!=13)

{

cout<<"\*";

password[inc]=ch;

inc++;

}

else

break;

}

password[inc]='\0';

if(strcmpi("password",password)==0)

{

cout<<"\nTHE PASSWORD IS CORRECT.... YOU HAVE ACCESS TO THE ADMIN FUNCTIONS....\n\n";

system("pause");

system("cls");

do

{

system("cls");

cout<<"\n\n\t\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*MENU\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\n\n\n";

cout<<"\t1. Display all accounts \n\n"

<<"\t2. Total Cash Deposited \n\n"

<<"\t3. Go back to main menu\n\n"

<<"\t4. Exit\n\n";

cout<<"enter your choice "; cin>>choice;

switch(choice)

{

case '1':

display\_all\_records();

system("pause");

break;

case '2':

total\_cash();

system("pause");

break;

case '3':

system("cls");

startMenu();

break;

case '4':

exit(0);

break;

default:

cout<<"invalid choice \n";

cout<<"\a";

}

}while(ch!='4');

break;

}

else

{

counter=counter+1;

cout<<"\npassword incorrect... \n ";

cout<<"you have "<<3-counter<<" chances\n";

inc=0;

strcpy(password," ");

}

}while(counter!=3);

}

void decorate(int n)

{

for(int i=1;i<=n;i++)

cout<<"\*";

cout<<endl;

}

void bank\_box()

{

cout<<"\t\t";

decorate(33);

cout<<"\n\t\t\*\t\t\t\t\*"<<endl;

cout<<"\n\t\t\*\tPRABMK BANK SERVICE\t\*"<<endl;

cout<<"\n\t\t\*\t\t\t\t\*\n\n";

cout<<"\t\t";

decorate(33);

}

int main()

{

bank\_box();

cout<<setw(46)<<"\n\nWELCOME TO PRABKM BANK\n\n "<<endl;

system("pause");

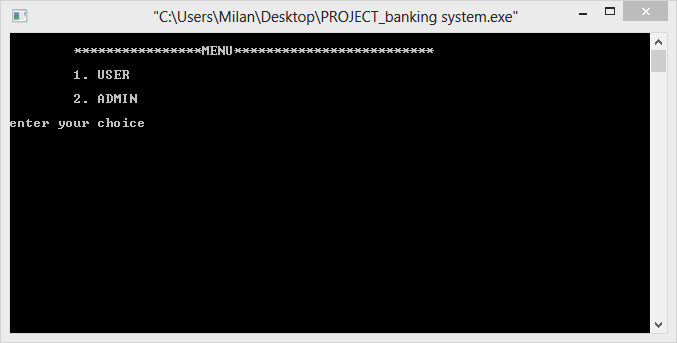
startMenu();

}//END OF PROJECT

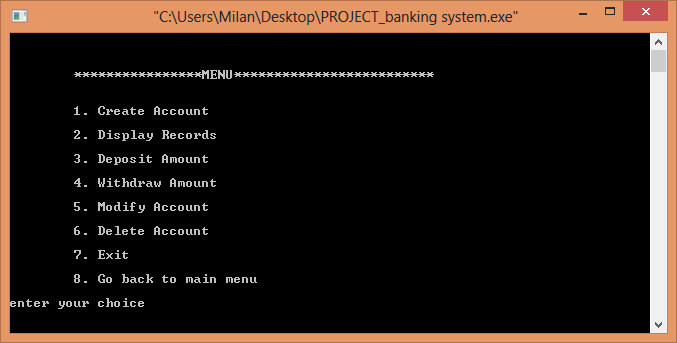
# Sample output

# 

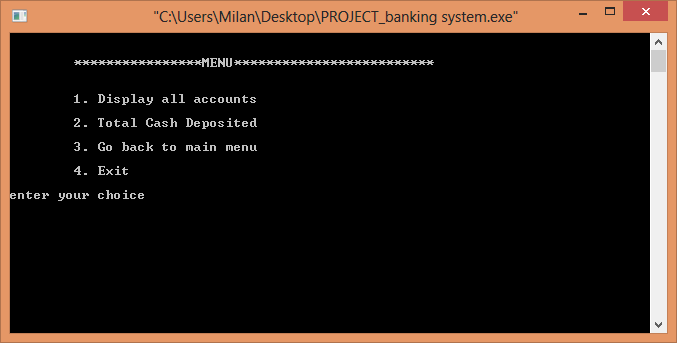
Welcome screen



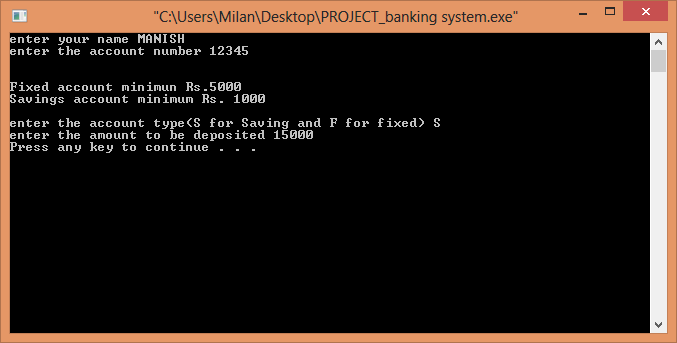
USER AND ADMIN WINDOW



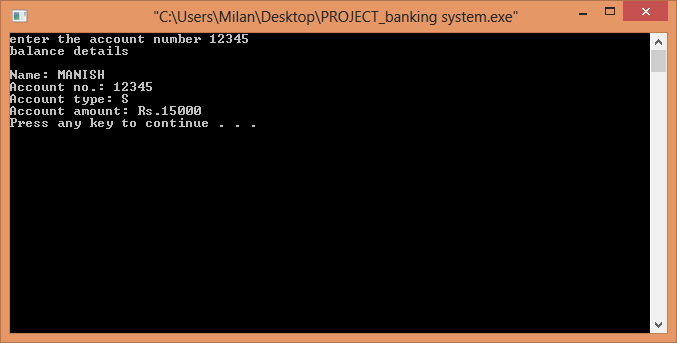
USER MENU FUNCTIONS



ADMIN MENU FUNCTIONS



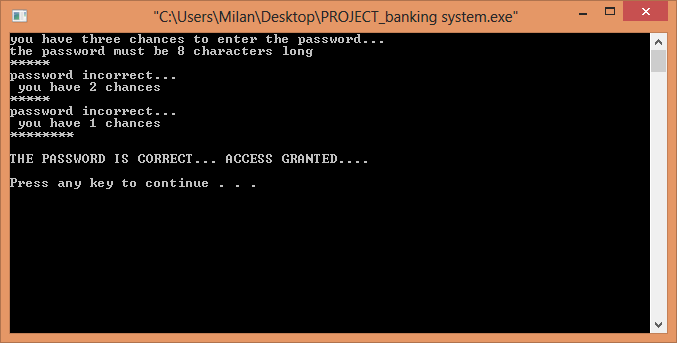
ACCOUNT BEING CREATED



CREATED ACCOUNT DISPLAYED

# 

AMOUNT BEING DEPOSITED



PASSWORD ENTRY OF ADMIN ACCOUNT

# CHAPTER 8: LIMITATIONS AND ENHANCEMENTS

This mini-project titled “BANKING SYSTEM”, being a small endeavor of our team, does not fully model a real banking system software, so it has its limitations and drawbacks as well as several aspects that can be added and some that can be further enhanced.

Some of the limitations of this program are listed below:

1. Perhaps the greatest limitation of this software is that it does not model banking system software precisely. Several key features of banking system software such as bank statement, provision of loans, ATM banking have been left out due to complexity.
2. This program does not have the features of interest calculations.
3. The transactions done in this program are not sorted by date hence the program cannot offer the date in which the transaction occurred, for instance, the program cannot offer the date in which an account was created, some amount of money was deposited or a certain amount of money was withdrawn.
4. This program uses a command-line method instead of a graphical interface, so it is not as user-friendly.

Some enhancements that can be made in program are:

1. An interest calculating module can be added which calculates the interest on the basis of the type of account(either savings or fixed);
2. Several modules for handling various kinds of loans such as education loan, mortgage overdraft, hire-and-purchase loan can be added.
3. All the transactions have to be made on a time basis which is not done in the current program.
4. A graphical user interface can be added to make the program user friendly and even simpler to use.
5. More features to the admin menu can be added such as modules showing cash currently available at the bank, the amount of cash that has been given as loans, list of customers in the bank and their daily transaction ,etc.

# CHAPTER 9: CONCLUSION

In conclusion we would again like to extend our heartfelt gratitude to our teachers who gave us the required knowledge to complete this project. We are ever grateful to them for patiently listening to our queries and helping us, correcting our mistakes as well as guiding us in our project.

Through this project we have had a chance to broaden our minds about the concepts of programming and as well as the its practical application in our lives. Using the concepts that we have been taught to us and some that we have learned on our own, we have tried to create a software that is modeled on real-world application. We learnt many things in and from this project and it is with our hard work and team effort that we could complete this project in the designated time.

**References**

1. Lafore, Robert “Object Oriented Programming in C++”, *4th edition, Pearson Edition*
2. Dietel & Dietel, “C++ How to program”, *5th edition, Pearson Education*
3. E. Balaguruswamy, “Programming with C++”, *4th edition, Tata McGraw Hill Companies*

Websites:

* [*www.about.com/cplusplus*](http://www.about.com/cplusplus)
* [*www.google.com*](http://www.google.com)
* [*www.cplusplus.com*](http://www.cplusplus.com)