|  |
| --- |
| **Electronic Bank Management System** |
| C++ Computer Science Project Report (2014-2014) |
| ***PREPARED BY: Rishabh Karajgi***  ***NIT Warangal*** |
|  |
|  |
|  |
|  |
|  |

**INDEX**

|  |  |
| --- | --- |
| **S.NO** | **Content** |
| 1 | Introduction |
| 2 | System Requirements |
| 3 | Data Flow diagram |
| 4 | Logical Description |
| 5 | Header Files used |
| 6 | Functions and files used |
| 7 | Source Code |

**INTRODUCTION**

We are all associated with banks in one way or the other. This project gives an ideal picture of the Electronic Bank Management System.

It provides functions to help us to create Account and close an existing Account. It also provides us functions to perform basic banking operations such as depositing and withdrawing cash, and the concept of ‘electronic money transfer’. Further it includes a function to view all the transactions made by the customer, making use of binary data files.

In this project, I’ve tried to use all features involved in Object Oriented Programming on the C++ platform. I will be thankful to receive any beneficial feedback.

**SYSTEM REQUIREMENTS**

* Pentium/Celeron(300mHz or above)
* 16mb RAM
* 5mb free space
* Floppy Disk Drive/Compact Disk Drive
* Windows 3.x or higher
* Turbo C++ 3.5 or higher

**DATA FLOW**

**LOGICAL DESCRIPTION**

The project starts with a welcome screen and goes to a menu which displays all the options. From this main page, a user can enter any of the 9 choices. Entering a wrong choice will redirect to the main menu again. All choices which require the account number to be entered check if the account number is less than 100. If not, the user is redirected to the main menu.

|  |
| --- |
| ***Option 1: CREATE NEW ACCOUNT->*** This invokes the function **void write\_account()**. Inside the function, a binary data file is opened. (ACCOUNT.DAT). The function calls a **class member function void account::create\_account()**, which asks the user to enter account number, account holder name, account type, and initial deposit. All these values are subject to certain conditions, which are shown to the user. If any condition is violated, the registration process *restarts*. Upon successful account registration, the user is redirected to the main menu. |
| ***Option 2 : ACCOUNT CREDIT->*** After account number verification, this invokes **void deposit\_withdraw(int1, int2)** . The function takes two arguments of int data type. int1 is the entered bank account number. int2 has a value 1 if the user selects option 2 for account credit in the main menu.  The function opens a binary data file “ACCOUNT.DAT” and displays the current account details/status. It asks the user for the amount to be credited. After checking for a valid amount, and displaying **the class member function float account::dep (float x)** is called. The function takes a float argument, which is the amount to be credited to the account. The member function adds the amount to the balance and the control returns to the main menu. |
| ***Option 3 : ACCOUNT DEBIT->*** After account number verification, this invokes **void deposit\_withdraw(int1, int2)** . The function takes two arguments of int data type. int1 is the entered bank account number. int2 has a value 2 if the user selects option 3 for account debit in the main menu.  The function opens a binary data file “ACCOUNT.DAT” and displays the current account details/status. It asks the user for the amount to be debited. After checking for a valid amount, and displaying the **class member function float account::draw(float x)** is called. The function takes a float argument, which is the amount to be debited from the account. The member function subtracts the amount from the balance and the control returns to the main menu. |
| ***Option 4 : ACCOUNT INFOMRATION ENQUIRY->*** After account number verification, this invokes void display\_sp(int). It searches the account number from the binary data file “ACCOUNT.DAT”. If the account number has no match, an error message is displayed and control returns to the main menu. Else, the class member function void account::show\_account() is invoked. The function displays the respective details of the account and returns control to the main menu. |
| ***Option 5 : ALL ACCOUNT HOLDERS LIST->*** The function **void display\_all()** is invoked. In the function, a binary data file ACCOUNT.DAT is opened. Data is read from the file and displayed as a tabulated output with four columns as Account No, Account Holder Name, Account Type and balance. With the press of a key, control is transferred to the main menu. |
| ***Option 6 : CLOSE AN ACCOUNT->***After account number verification, the function **void delete\_account(int**) is invoked. The function takes the entered account number as an argument. It opens two binary data files “ ACCOUNT.DAT” and “TEMP.DAT”. It searches the account number from the data file “ACCOUNT.DAT”. If the account number does not match, the record of that account is copied to “TEMP.DAT”. This happens for all account numbers that do not match using a while loop.  After the loop run, the account number matching with the entered account number remains in “ACCOUNT.DAT”. That file is now deleted. “TEMP.DAT” is *renamed* to “ACCOUNT.DAT”. An information message is displayed and the control returns to the main menu. |
| ***Option 7 : MODIFY ACCOUNT->*** After account number verification, the function **void modify\_account(int)** is invoked. The function takes the entered account number as an argument. It opens a binary data file “ACCOUNT.DAT” and searches for the entered account number. If no match is found , an error message is displayed and the control returns to the main menu.  Else, **the class member function void account::show\_account()** is invoked to display all details. Then, another **class member function, void account::modify()** is invoked to accept input from user for account number, account holder name, account type, and initial deposit. The record is modified according o the new user input and the control returns to the main menu. |
| ***OPTION 8 : MONEY TRANSFER BETWEEN TWO ACCOUNTS->***  The function void **moneyac\_transfer()** is invoked. In the function, the user enters the account number(int ac1) FROM which money is to be transferred and the account number (int ac2) TO which money is to be transferred. For the transfer to be successful, both the accounts should exist in the binary data file “ACCOUNT.DAT”, which is opened in the function. If either of the account number verifications fails, an error message is displayed, and the user is requested to re-enter the correct account numbers.  If the account numbers are correct, then the user enters the amount to be transferred from ac1 to ac2.  **The class member function float account::draw(float x)** is called for ac1 where x is the amount to be transferred. This would deduct the money from ac1.  Then, **the class member function float account::draw(float x)** is called for ac2, where x is the amount to be transferred. This would deposit the money to ac2. After successful money transfer, the control is transferred to the main menu. |
| ***OPTION 9 : EXIT->***  The program terminates. **void Box()** function is invoked which displays a box made of \*\*\* in which the names of the team members are written. |

**HEADER FILES USED**

|  |  |  |
| --- | --- | --- |
| S.No | Header file | Purpose |
| 1 | <fstream.h> | For file operations ( reading and writing) |
| 2 | <iostream.h> | For input/output operations |
| 3 | <iomanip.h> | For formatting output |
| 4 | <conio.h> | Controlling output (clrscr() etc. ) |
| 5 | <stdio.h> | For accepting/displaying string etc. |
| 6 | <process.h> | For exit() |

**FUNCTIONS USED**

|  |  |  |
| --- | --- | --- |
| **S.No** | **Function** | **Purpose** |
| 1 | void write\_account() | function to write record in binary file |
| 2 | void display\_sp(int) | function to display account details given by user |
| 3 | void modify\_account(int) | function to modify record of file |
| 4 | void delete\_account(int) | function to delete record of file |
| 5 | void display\_all() | function to display all account details |
| 6 | void deposit\_withdraw(int, int); | function to DEPOSIT AND WITHDRAW amount for given account |
| 7 | void moneyac\_transfer(); | function to transfer money between two specific accounts |
| 8 | void intro(); | introductory screen function |
| 9 | void box(); | function for making box made of \*\*\*\* |

**SOURCE CODE**

***//\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*HEADER FILES USED\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\****

#include<fstream.h>

#include<ctype.h>

#include<iomanip.h>

#include<conio.h>

#include<stdio.h>

#include<process.h>

***//\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*CLASS USED IN PROJECT\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\****

class account

{

int acno;

char name[50];

int deposit;

char type;

public:

void create\_account(); //function to get data from user

void show\_account(); //function to show data on screen

void modify(); //function to modify account data from user

void dep(float); //function to accept amount and add/DEPOSIT to balance amount

float draw(float); //function to accept amount and subtract/WITHDRAW from balance amount

void report(); //function to show data in tabular format

int retacno(); //function to return account number

int retdeposit(); //function to return balance amount

char rettype(); //function to return type of account

};

void account::create\_account()//Function to create account using specific details

{

accreate:

clrscr();

cout<<"\n\n MAIN MENU->1.NEW ACCOUNT-->REGISTRATION\n\n";

cout<<"\n WARNING: ANY ERROR WHILE REGISTERING, OR ENTERING ANY FORM OF DATA \n THAT IS NOT RECOGNISED BY THE RESPECTIVE FIELDS WILL RESTART THE REGISTRATION\n PROCESS. USERS ARE ADVISED TO BE CAREFUL WHILE REGISTERING. ";

cout<<"\n\n\nEnter The account No.(>100) :"; cin>>acno;

if (acno<100)

{

cout<<"\a\n\n Enter account number correctly.";getch();

goto accreate;

}

ifstream file;

file.open("account.dat",ios::binary);

account a1;

while(file)

{

file.read((char\*)&a1,sizeof(a1));

if(a1.retacno()==acno)

{

cout<<"\a\n\n Account number already exists.";

getch();

goto accreate;

}

}

cout<<"\n\nEnter The Name of The account Holder : "; gets(name);

cout<<"\nEnter Type of The account (C/S) : "; cin>>type;

type=toupper(type);//converting to uppercase character

cout<<"\n\n Your account type choice:"<<type;

if(type!='C'&&type!='S')

{

cout<<"\a\n\n Incorrect Account type. Choose from C or S. ";

goto accreate;

}

cout<<"\nEnter The Initial amount(>=500 for Saving and >=1000 for current ) : ";

cin>>deposit;

if((type=='S')&&(deposit<500))

{

cout<<"\n For Savings Account, enter an initial amount greater than or equal to 500.";

goto accreate;

}

else if ((type=='C')&&(deposit<1000))

{ cout<<"\n For Current Account, enter an initial amount greater than or equal to 1000.";

goto accreate;

}

else

cout<<"\n\n\nAccount Created Successfully. Press any key to return to MAIN MENU.";

}

void account::show\_account()//Function to show account details

{

cout<<"\nAccount No. : "<<acno;

cout<<"\nAccount Holder Name : "<<name;

cout<<"\nType of Account : "<<type;

cout<<"\nBalance amount : "<<deposit;

}

void account::modify()//Function to modify existing account

{

cout<<"\nAccount Number: "<<acno;

cout<<"\n\nEnter Name of Account Holder : "; gets(name);

cout<<"\nEnter Type of The account (C/S) : "; cin>>type;

type=toupper(type);

cout<<"\nEnter The amount : "; cin>>deposit; }

void account::dep(float x)//Function to ADD/DEPOSIT money in account

{

deposit+=x; }

float account::draw(float x)//Function to SUBTRACTD/WITHDRAW money in account

{

deposit-=x;

return(deposit); }

void account::report()//Function to PRINT tabulated report

{

cout<<acno<<setw(10)<<" "<<name<<setw(10)<<" "<<type<<setw(6)<<deposit<<endl; }

int account::retacno()//Function to return account number

{

return acno; }

int account::retdeposit()//Function to return deposit

{

return deposit; }

char account::rettype()//Function to return accoutn type

{

return type; }

//END OF CLASS ASSOCIATED FUCNTIONS

***//\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*FUNCTION DECLARATION\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\****

void write\_account(); //function to write record in binary file

void display\_sp(int); //function to display account details given by user

void modify\_account(int); //function to modify record of file

void delete\_account(int); //function to delete record of file

void display\_all(); //function to display all account details

void deposit\_withdraw(int, int); // function to DEPOSIT AND WITHDRAW amount for given account

void moneyac\_transfer();//function to transfer money between two specific accounts

void intro(); //introductory screen function

void box();

***//\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*THE MAIN FUNCTION OF THE PROGRAM\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\****

int main()

{

int ch;

int num;

clrscr();

intro();//Fucntion to display introductory screen

do

{ mainmenu:

clrscr();

cout<<"\n\n\n\tMAIN MENU";

cout<<"\n\n\n\t=========";

cout<<"\n\n\t01. NEW ACCOUNT";

cout<<"\n\n\t02. ACCOUNT CREDIT";

cout<<"\n\n\t03. ACCOUNT DEBIT";

cout<<"\n\n\t04. ACCOUNT INFORMATION ENQUIRY";

cout<<"\n\n\t05. ALL ACCOUNT HOLDER LIST";

cout<<"\n\n\t06. CLOSE AN ACCOUNT";

cout<<"\n\n\t07. MODIFY ACCOUNT ";

cout<<"\n\n\t08. MONEY TRANSFER BETWEEN TWO ACCOUNTS";

cout<<"\n\n\t09. EXIT";

cout<<"\n\n\tSelect Your Option (1-9): ";

cin>>ch;

clrscr();

if(ch>9)

{cout<<"\nEnter a choice less than 10. Redirecting......\n\n";

goto mainmenu;

}

switch(ch)

{

case 1://NEW ACCOUNT

write\_account(); break;

case 2://DEPOSIT AMOUNT: n=1->DEPOSIT

cout<<"\n\n MAIN MENU->2.ACCOUNT CREDIT";

cout<<"\n\n\n\n\tEnter The account No. : "; cin>>num;

if(num<100)

{

cout<<"\a\n WRONG ACCOUNT NUMBER. Going back to main menu...";

goto mainmenu;

long wait=-9999;

while(++wait<10000);

}

deposit\_withdraw(num, 1); break;

case 3://WITHDRAW AMOUNT: n=2->WITHDRAW

cout<<"\n\n MAIN MENU->3.ACCOUNT DEBIT";

cout<<"\n\n\n\n\tEnter The account No. : "; cin>>num;

if(num<100)

{

cout<<"\a\n WRONG ACCOUNT NUMBER. Going back to main menu...";

long wait=-9999;

while(++wait<10000);

goto mainmenu;

}

deposit\_withdraw(num, 2); break;

case 4://BALANCE ENQUIRY

cout<<"\n\n MAIN MENU->4.ACCOUNT INFORMATION ENQUIRY";

cout<<"\n\n\n\n\tEnter The account No. : "; cin>>num;if(num<100)

{

cout<<"\n WRONG ACCOUNT NUMBER. Going back to main menu...";

long wait=-9999;

while(++wait<10000);

goto mainmenu;

}

display\_sp(num); break;

case 5://DISPLAY REPORT

cout<<"\n\n MAIN MENU->5.ALL ACCOUNT HOLDER LIST\n\n\n";

display\_all(); break;

case 6://CLOSE ACCOUNT

cout<<"\n\n MAIN MENU->6.CLOSE ACCOUNT ";

cout<<"\n\n\n\n\tEnter The account No. : "; cin>>num;

if(num<100)

{

cout<<"\n WRONG ACCOUNT NUMBER. Going back to main menu...";

long wait=-9999;

while(++wait<10000);

goto mainmenu;

}

delete\_account(num); break;

case 7://MODIFY ACCOUNT

cout<<"\n\n MAIN MENU->7.MODIFY ACCOUNT ";

cout<<"\n\n\n\n\tEnter The account No. : "; cin>>num;

if(num<100)

{

cout<<"\n WRONG ACCOUNT NUMBER. Going back to main menu...";

long wait=-9999;

while(++wait<10000);

goto mainmenu;

}

modify\_account(num); break;

case 8://MONEY TRANSFER BETWEEN TWO ACCOUNTS

cout<<"\n\n MAIN MENU->8.MONEY TRANSFER BETWEEN TWO ACCOUNTS ";

moneyac\_transfer(); break;

case 9://EXIT

cout<<"\n\t Thank you for using the Electronic Bank Management System\n\n\n\n\n\n";

box();

exit(0);

}//end of switch case

getch();

}while(ch!=9);

return 0;

}

***//\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*FUNCTION TO WRITE IN FILE\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\****

void write\_account()

{

account ac;// Class:account , Class Object:ac

ofstream outFile; //ofstream mode:to write data to file

outFile.open("account.dat",ios::binary|ios::app); //binary file for data writing, append mode : to continue writing after the last record

ac.create\_account();//Member function of class to initialize values

outFile.write((char \*) &ac, sizeof(account));//Writing enterted details to record

outFile.close();//Closing file

}

***//\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*FUNCTION TO READ SPECIFIC ACCOUNT FROM FILE\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\****

void display\_sp(int n)//Function to display specific account details(given by user)

{

account ac;// Class:account Class Object:ac

int flag=0;

ifstream inFile;//ifstream mode:to read data from file

inFile.open("account.dat",ios::binary);//binary file for data reading

//Error Checking for error during file opening

if(!inFile)

{

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

return;

}

//Error Checking for error during file opening

cout<<"\nBALANCE DETAILS\n";

while(inFile.read((char \*) &ac, sizeof(account)))//Reading the file "InFile"

{

if(ac.retacno()==n)//Checking if saved account number = enyered account number

{

ac.show\_account();//Funciton to show account

flag=1;

}

}

inFile.close();//Closing File

if(flag==0)

cout<<"\a\n\nAccount number does not exist";

}

***//\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*FUNCTION TO MODIFY RECORD OF FILE\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\****

void modify\_account(int n)

{

int found=0;

account ac;

fstream File;

File.open("account.dat",ios::binary|ios::in|ios::out);//Opening File in both modes

//Error Checking for error during file opening

if(!File)

{

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

return;

}

//Error Checking for error during file opening

while(File.read((char \*) &ac, sizeof(account)) && found==0)//Reading the file "File"

{

if(ac.retacno()==n)//Checking if saved account number = entered account number

{

ac.show\_account(); //Function to show account

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

ac.modify();//Function to modify account

int pos=(-1)\*sizeof(account); //position for entry into binary data file

File.seekp(pos,ios::cur);//positioning the pointer for data entry

File.write((char \*) &ac, sizeof(account)); //writing data to file

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

found=1;

}

}

File.close();//closing file

if(found==0)

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

}

***//\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*FUNCTION TO DELETE ACCOUNT FROM FILE\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\****

void delete\_account(int n)

{

account ac;

ifstream inFile;//to read data from file

ofstream outFile;//to write dat to file

inFile.open("account.dat",ios::binary);

//Error Checking for error during file opening

if(!inFile)

{

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

return;

}

//Error Checking for error during file opening

outFile.open("Temp.dat",ios::binary);

inFile.seekg(0,ios::beg);//Placing pointer at the beginning of the file

while(inFile.read((char \*) &ac, sizeof(account)))//Reading records from file

{

if(ac.retacno()!=n)//chekcing: if saved account number = entered account number

{

outFile.write((char \*) &ac, sizeof(account));//writing records to file

}

}

inFile.close(); //Closing file

outFile.close();//Closing file

remove("account.dat"); //Deleting Binary file

rename("Temp.dat","account.dat"); //renaming Binary File

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

}

***//\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*FUNCTION TO DISPLAY ALL ACCOUNTS DEPOSIT LIST\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\****

void display\_all()

{

account ac;

ifstream inFile; //to read data from file

inFile.open("account.dat",ios::binary); //opening binary file to read data

//Error Checking for error during file opening

if(!inFile)

{

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

return;

}

//Error Checking for error during file opening

cout<<"\n\n\t\tACCOUNT HOLDER LIST\n\n";

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

cout<<"A/c no. NAME Type Balance\n";

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

while(inFile.read((char \*) &ac, sizeof(account)))//Reading file records

{

ac.report(); //Calling function to print in tabulated form

}

inFile.close();//Closing file

}

***//\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*FUNCTION TO DEPOSIT AND WITHDRAW AMOUNTS\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\****

void deposit\_withdraw(int n, int option)

{

int amt;

int found=0;

account ac;

fstream File;//TO read data from file and write data to file

File.open("account.dat", ios::binary|ios::in|ios::out);

//Error Checking for error during file opening

if(!File)

{

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

return;

}

//Error Checking for error during file opening

while(File.read((char \*) &ac, sizeof(account)) && found==0) //'account' is the class. 'ac' is the class object.

{

if(ac.retacno()==n)//Checking: if saved account number = entered account number

{

ac.show\_account(); //SHOWING ACCOUNT DETAILS AND CURRENT STATUS

if(option==1)//1->DEPOSIT

{

rp1:

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

cout<<"\n\nEnter The amount to be deposited";

cin>>amt;

//Checking for negative amount

if(amt<0)

{

cout<<"\a\n\n ERROR: Negative amount. Please re-enter.";

goto rp1;

}

else

ac.dep(amt);//'amt' is passed to member function: dep

}

if(option==2)//2->WITHDRAW

{

rp2:

cout<<"\n\n\tTO WITHDRAW AMOUNT ";

cout<<"\n\nEnter The amount to be withdraw";

cin>>amt;

if(amt<0)

{

cout<<"\a\n\n ERROR: Negative amount. Please re-enter.";

goto rp2;

}

else

{

int bal=ac.retdeposit()-amt;

if((bal<500 && ac.rettype()=='S') || (bal<1000 && ac.rettype()=='C'))

cout<<"Insufficient balance";

else

ac.draw(amt);//'amt' is passed to member function: draw

}

}

int pos=(-1)\* sizeof(ac);//position of record

File.seekp(pos,ios::cur);//Plaicing pointer at the current position

File.write((char \*) &ac, sizeof(account));//WRitign data to file

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

found=1;

}

}

File.close();//CLosign the file

if(found==0)

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

}

***// \*\*\*\*\*\*\*\*\*\*\*\*\*\*\*FUNCTION TO TRANSFER MONEY BETWEEN TWO ACCOUNTS\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\****

void moneyac\_transfer()

{ account ac;

int ac1,ac2,tf=0,pos1=0,pos2=0;

float acmt;

rp3:

cout<<"\n Enter account number FROM which money is to TRANSFERRED:";

cin>>ac1;

cout<<"\n Enter account number TO which money is to be DEPOSITED:";

cin>>ac2;

fstream File;//TO read data from file and write data to file

File.open("account.dat", ios::binary|ios::in);

//Error Checking for error during file opening

if(!File)

{

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

return;

}

while(File)

{

pos1=File.tellg();

File.read((char\*)&ac,sizeof(ac));

cout<<"\n Verifying account number...";

long wait=-99999;

while(++wait<10000);

if(ac.retacno()==ac1)//Checking: if saved account number1 = entered ac1

{

cout<<"\n Ac1 Matches.";

cout<<"\n Saved account number"<<ac.retacno();

cout<<"\n Entered account number1:"<<ac1;

tf=1;

break;

}

}

File.close();

if(tf==0)

{

cout<<"\n Account number 1 doesn't match. Retry.";

goto rp3;

}

else

{ tf=0;

File.open("account.dat", ios::binary|ios::in);

while(File)

{//start of while loop

pos2=File.tellg();

File.read((char\*)&ac,sizeof(ac));

if(ac.retacno()==ac2)//Checking: if saved account number1 = entered ac1

{ cout<<"\n Account number 2 checking...";

cout<<"\n Ac2 Matches.";

cout<<"\n Saved account number"<<ac.retacno();

cout<<"\n Entered account number2:"<<ac2;

tf=1;

break;

}//end of if

}//end of while

File.close();

if(tf==0)

{

cout<<"\n Account number 2 doesn't match. Retry.";

goto rp3;

}

else

{//start of else

cout<<"\nEnter the amount to be transferred:";

cin>>acmt;

//deducting money from account 1..

File.open("account.dat", ios::binary|ios::in|ios::out);

File.seekg(pos1);

File.read((char\*)&ac,sizeof(ac));

if(ac.draw(acmt)<0)

cout<<"\n\nTRANSFER NOT POSSIBLE DUE TO INSUFFICIENT BALANCE"<<endl;

else

{//start of else

File.seekp(pos1);

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

File.close();

cout<<"\n Rupees "<<acmt<<" is transferred from account: "<<ac1;

//depositing money to account 2..

File.open("account.dat", ios::binary|ios::in|ios::out);

File.seekg(pos2);

File.read((char\*)&ac,sizeof(ac));

ac.dep(acmt);

File.seekp(pos2);

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

File.close();

cout<<"\n Rupees "<<acmt<<" is transferred to account: "<<ac2;

}//end of else

}//end of else

}//end of else

}//end of moneyac\_transfer function

***//\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*INTRODUCTION FUNCTION\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\****

void intro()

{

cout<<"\n\n\n\t\t\t BANK MANAGEMENT SOFTWARE\n";

cout<<"\t\t ========================================";

cout<<"\n\t COMPUTER SCIENCE PROJECT FOR CLASS XII: 2013-2013";

cout<<"\n\n\n\n MADE BY : Rishabh Karajgi(36)\n ";

cout<<"\n\n\n Welcome to the Bank Management Software.\n Let's get started. Press any key to proceed. ";

getch();

}

void box()

{

int i,j;

for( j=0;j<29;++j)//loop controlling horizontal alignment

cout<<" ";

for (i=0; i<10; i++)

cout<<'\*'<<" "; //prints the first row

i=1;

while (i<6)

{

cout<<"\n";

for(int j=0;j<29;++j)

cout<<" ";

cout<<'\*';

if(i==1)

{cout<<" Done By:-";cout<<setw(8)<<'\*';}

else if(i==2)

{cout<<" Rishabh Karajgi";cout<<setw(2)<<'\*';}

i++;

}

cout<<"\n";

for(j=0;j<29;++j)//loop controlling horizontal alignment

cout<<" ";

for (i=0;i<10; i++)

cout<<'\*'<<" "; // prints the last row

cin.get();

}

***//\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*END OF PROJECT\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\****

**Conclusion**

This project gave me a good opportunity to exhibit me programming skills in a highly practical manner.

**BIBLIOGRAPHY**

1. Sumit Arora Class XII Computer Science Textbook