HEADER FILES USED

iostream.h: input output stream header file   
Provides functionality to use an abstraction called streams specially designed to perform input and output operations on sequences of character, like files or strings.

fstream.h:  for defining several iostreams template classes that manipulate exteral files   
fstream is a standard C++ library that handles reading from and writing to files either in text or in binary formats. It is an object oriented alternative to C's FILE from the C standard library. fstream is the result of a multiple inheritance with ifstream and ofstream, which both inherit from ios.

conio.h: It is a C header file used in old MS-DOS compilers to create text user interfaces .

stdio.h: contains the function definitions of your basic input/output (I/O) functions. Without these declarations your code wouldn't compile and you wouldn't be able to use functions such as printf etc.

process.h: is a C header file which contains function declarations and macros used in working with threads and processes.

string.h: Contains the definition of class string from the C++ Standard Library.

iomanip.h: Provides facilities to manipulate output formatting, such as the base used when formatting integers and the precision of floating point values.

ctype.h: The ctype.h header file of the C Standard Library declares several functions that are useful for testing and mapping characters.

stdlib.h: The stdlib.h header defines four variable types, several macros, and various functions for performing general functions.

dos.h: dos.h header file of C language contains functions for handling interrupts, producing sound, date and time functions etc. It is Borland specific and works in Turbo C compiler.

BINARY FILES USED

NEWRECOR.DAT

TRANSACT.DAT

**SOURCE CODE**

//header files

#include<iostream.h>

#include<fstream.h>

#include<process.h>

#include<string.h>

#include<stdlib.h>

#include<stdio.h>

#include<ctype.h>

#include<conio.h>

#include<dos.h>

#include<iomanip.h>

//menu class

class menus

{

public:

void showmenu(void);

private:

void closemenu(void);

};

//class to display records

class disprecords

{

public:

void adddetails(int,char name[30],char address[60],float);

void displaycustomers(void);

void deleteaccount(int);

void updatebalance(int,float);

void updatecustomer(void);

int lastaccount(void);

int accountexists(int);

char \*getname(int);

char \*getaddress(int);

float getbalance(int);

int getrecord(int);

void display(int);

// void displaylist(void);

int accountnumber;

char name[50],address[50];

float intbalance;

};

//transaction class

class accounttransactions

{

public:

void new\_account(void);

void closeaccount(void);

void display\_account(void);

void transaction(void);

void adddetails(int,int,int,int,char,char typetransaction[15],float,float,float);

void deleteaccount(int);

int datediffer(int,int,int,int,int,int);

float getinterest(int,float);

void display(int);

void showaccount(int);

int accountnumber;

char trantype[10];

int dday,mmonth,yyear;

char transactions;

//withdrawl of account

float intinterest,intamount,intbalance;

static float calcinterest;

void showinterest(void);

};

void menus::showmenu(void)

{

char choice;

while(1)

{

system("cls");//clrscr()

cout<<"\n ----welcome to banking system application- \n";

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

cout<<" choose from options\n";

cout<<" 1:open an account\n";

cout<<" 2:view an account\n";

cout<<" 3:show all accounts\n";

cout<<" 4:make a transaction\n";

cout<<" 5:calculate interest\n";

cout<<" 6:close an account\n";

cout<<" 7:exit\n\n";

cout<<" please select a choice:";

choice=getche();

if(choice=='1')

{

accounttransactions objat;

objat.new\_account();

}

else

if(choice=='2')

{

accounttransactions objat;

objat.display\_account();

}

else

if(choice=='3')

{

disprecords newrec;

newrec.displaycustomers();

}

else

if(choice=='4')

{

accounttransactions objat;

objat.transaction();

}

else

if(choice=='5')

{

accounttransactions objat;

objat.showinterest();

}

else

if(choice=='6')

{

closemenu();

}

else

if(choice=='7')

{

cout<<"\n Thanks for using this application.please press any key to exit\n";

getch();

break;

}

}

}

void menus::closemenu(void)

{

char choice;

while(1)

{

system("cls");//clrscr()

cout<<" -close menu \n";

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

cout<<" 1:close/delete an account\n";

cout<<" 0:exit from this menu\n\n";

cout<<"select a choice : ";

choice=getche();

if(choice=='1')

{

accounttransactions at;

at.closeaccount();

break;

}

else

if(choice=='0')

{

cout<<"\n you have entered 0 to go back to the previous menu.\n";

getch();

break;

}

}

}

int disprecords::lastaccount(void)

{

fstream filename;

filename.open("newrecords.dat",ios::in);

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

int count=0;

while(filename.read((char\*)this,sizeof(disprecords)))

count=accountnumber;

filename.close();

return count;

}

int disprecords::getrecord(int retrieve\_accno)

{

fstream filename;

filename.open("newrecords.dat",ios::in);

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

int count=0;

while(filename.read((char\*)this,sizeof(disprecords)))

{

count++;

if(retrieve\_accno==accountnumber)

break;

}

filename.close();

return count;

}

//newrecords.dat file

void disprecords::display(int retrieve\_accno)

{

int record;

record = getrecord(retrieve\_accno);

fstream filename;

filename.open("newrecords.dat", ios::in);

filename.seekg(0,ios::end);

int location;

location = (record) \* sizeof(disprecords);

filename.seekp(location);

while (filename.read((char\*)this,sizeof(disprecords)))

{

if(retrieve\_accno == accountnumber)

{

cout<<endl<<"account no. "<<accountnumber;

cout<<endl<<"name : "<<name;

cout<<endl<<"address :"<<address;

cout<<endl<<"balance :"<<intbalance;

break;

}

}

filename.close();

}

char \*disprecords :: getname(int retrieve\_accno)

{

fstream filename;

filename.open("newrecords.dat", ios::in);

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

char retrieve\_custname[30];

while (filename.read((char\*)this,sizeof(disprecords)))

{

if(accountnumber == retrieve\_accno)

{

strcpy(retrieve\_custname,name);

}

}

filename.close();

return retrieve\_custname;

}

char \*disprecords :: getaddress(int retrieve\_accno)

{

fstream filename;

filename.open("newrecords.dat", ios::in);

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

char retrieve\_address[60];

while (filename.read((char\*)this,sizeof(disprecords)))

{

if(accountnumber == retrieve\_accno)

{

strcpy(retrieve\_address,address);

}

}

filename.close();

return retrieve\_address;

}

float disprecords :: getbalance(int retrieve\_accno)

{

fstream filename;

filename.open("newrecords.dat", ios::in);

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

float ibalance;

while (filename.read((char\*)this,sizeof(disprecords)))

{

if(accountnumber == retrieve\_accno)

{

ibalance = intbalance;

}

}

filename.close();

return ibalance;

}

//------------------------------------------

int disprecords :: accountexists(int retrieve\_accno)

{

fstream filename;

filename.open("newrecords.dat", ios::in);

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

int count = 0;

while (filename.read((char\*)this,sizeof(disprecords)))

{

if(accountnumber == retrieve\_accno)

{

count = 1;

break ;

}

}

filename.close();

return count;

}

//-------------------------------------

void disprecords :: displaylist()

{

cout<<" ";

cout<<" "<<endl;

int day1,month1,year1;

struct date dateval;

getdate(&dateval);

day1 = dateval.da\_day;

month1 = dateval.da\_mon;

year1 = dateval.da\_year;

cout<<endl<<"Date: "<<day1 <<"/" <<month1 <<"/" <<year1<<endl;

cout<<setw(80)<<"\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_"<<endl;

cout<<setw(23)<<"ACCOUNT NO.";

cout<<setw(23)<<"NAME OF THE PERSON";

cout<<setw(23)<<"BALANCE"<<endl;

cout<<setw(80)<<"\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_"<<endl;

}

//----------------------------------------

void disprecords :: displaycustomers(void)

{

system("cls");//clrscr()

int len1;

int row=8, check;

fstream filename;

FILE \* pfile;

pfile = fopen("newrecords.dat","r");

if(pfile == NULL)

{

cout<<endl<<"no account exists. please go back to the previous menu."<<endl;

getch();

return;

}

else

{

displaylist();

filename.open("newrecords.dat", ios::in);

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

while (filename.read((char\*)this,sizeof(disprecords)))

{

check = 0;

cout.fill(' ');

cout<<setw(20);

cout.setf(ios::right,ios::adjustfield);

cout<<accountnumber;

cout.fill(' ');

cout<<setw(25);

cout.setf(ios::internal,ios::adjustfield);

cout<<name;

cout<<setw(23);

cout.setf(ios::right,ios::adjustfield);

cout<<intbalance<<endl;

row++;

if(row == 23)

{

check = 1;

row = 8;

cout<<"\n\n continue the application..."<<endl;

getch();

system("cls");//clrscr()

displaylist();

}

}

}

filename.close();

if(!check)

{

cout<<"\n\n continue the application..."<<endl;

getch();

}

}

//--------------------------------------

void disprecords :: adddetails(int retrieve\_accno, char retrieve\_custname[30],char retrieve\_address[60], float ibalance)

{

accountnumber = retrieve\_accno ;

strcpy(name,retrieve\_custname) ;

strcpy(address,retrieve\_address) ;

intbalance = ibalance;

fstream filename;

filename.open("newrecords.dat", ios::out | ios::app);

filename.write((char\*)this,sizeof(disprecords));

filename.close();

}

//------------------------------

void disprecords :: deleteaccount(int retrieve\_accno)

{

fstream filename;

filename.open("newrecords.dat", ios::in);

fstream temp;

temp.open("calculations.txt", ios::out);

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

while (!filename.eof() )

{

filename.read((char\*)this,sizeof(disprecords));

if(filename.eof())

break;

if( accountnumber != retrieve\_accno)

temp.write((char\*)this,sizeof(disprecords));

}

filename.close();

temp.close();

filename.open("newrecords.dat", ios::out) ;

temp.open("calculations.txt", ios::in) ;

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

while (!temp.eof() )

{

temp.read((char\*)this,sizeof(disprecords));

if(temp.eof())

break;

filename.write((char\*)this,sizeof(disprecords));

}

filename.close();

temp.close();

}

//---------------------------------

void disprecords :: updatebalance(int retrieve\_accno, float ibalance)

{

int record ;

record = getrecord(retrieve\_accno);

fstream filename;

filename.open("newrecords.dat", ios::out | ios::ate);

intbalance = ibalance;

int location;

location = (record-1) \* sizeof(disprecords);

filename.seekp(location);

filename.write((char\*)this,sizeof(disprecords));

filename.close();

}

//-----------------------------

void accounttransactions :: adddetails(int retrieve\_accno, int day1, int month1, int year1, char t\_tran, char typetransaction[10], float interest\_accured, float t\_amount, float ibalance)

{

fstream filename;

filename.open("transactions.dat", ios::app);

accountnumber = retrieve\_accno ;

dday = day1;

mmonth = month1;

yyear = year1;

transactions = t\_tran;

strcpy(trantype,typetransaction);

intinterest = interest\_accured;

intamount = t\_amount;

intbalance = ibalance;

filename.write((char\*)this,sizeof(accounttransactions));

filename.close();

}

//--------------------------------------------

void accounttransactions :: deleteaccount(int retrieve\_accno)

{

fstream filename;

filename.open("transactions.dat", ios::in);

fstream temp;

temp.open("calculations.txt", ios::in) ;

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

while (!filename.eof() )

{

filename.read((char\*)this,sizeof(accounttransactions));

if(filename.eof())

break;

if( accountnumber != retrieve\_accno)

temp.write((char\*)this,sizeof(accounttransactions));

}

filename.close();

temp.close();

filename.open("transactions.dat", ios::out);

temp.open("calculations.txt", ios::in) ;

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

while (!temp.eof() )

{

temp.read((char\*)this,sizeof(accounttransactions));

if(temp.eof())

break;

filename.write((char\*)this,sizeof(accounttransactions));

}

filename.close();

temp.close();

}

//-----------------------------

void accounttransactions :: new\_account(void)

{

char choice;

int i, check;

system("cls");//clrscr()

disprecords newrec ;

cout<<" Please press 0 to go back to previous menu,"<<endl;

cout<<" "<<endl;

cout<<" -open a new bank account- "<<endl;

cout<<" \*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*"<<endl;

int day1, month1, year1;

struct date dateval;

getdate(&dateval);

day1 = dateval.da\_day;

month1 = dateval.da\_mon;

year1 = dateval.da\_year;

int retrieve\_accno;

retrieve\_accno = newrec.lastaccount();

retrieve\_accno++;

if(retrieve\_accno == 1)

{

newrec.adddetails(retrieve\_accno,"Ravi","Delhi",1.1);

newrec.deleteaccount(retrieve\_accno);

adddetails(retrieve\_accno,1,1,1997,'D',"Default value",1.1,1.1,1.1);

deleteaccount(retrieve\_accno);

}

char retrieve\_custname[30], tran\_acc[10], retrieve\_address[60];

float t\_bal, ibalance;

cout<<" Date : "<<day1<<"/" <<month1 <<"/"<<year1<<endl;

cout<<" Account no. # "<<retrieve\_accno;

do

{

cout<<"\n\n please enter the name of the account holder :";

check = 1;

gets(retrieve\_custname);

if (retrieve\_custname[0] =='0')

{

cout<<"\n\t Invalid customer name.";

getch();

return;

}

strupr(retrieve\_custname);

if (strlen(retrieve\_custname) == 0 || strlen(retrieve\_custname) > 30)

{

check = 0;

cout<<"\n\t Customer name is either blank or greater than 30 characters"<<endl;

getch();

}

}

while(!check);

do

{

cout << endl<< " Please enter the account holder's address :";

check = 1;

gets(retrieve\_address);

if (retrieve\_address[0] == '0')

{

cout<<"\n\t Invalid customer address.";

getch();

return;

}

strupr(retrieve\_address);

if (strlen(retrieve\_address) == 0 || strlen(retrieve\_address) > 60)

{

check = 0;

cout<<"\n\t Customer address is either blank or its length is greater than 60 characters."<<endl;

getch();

}

}while(!check);

do

{

char chr\_verifyingperson[30];

cout <<endl<<" Please enter the Name of the Verifying Person of the Account Holder : ";

check = 1;

gets(chr\_verifyingperson);

if(chr\_verifyingperson[0]=='0')

{

cout<<"\n\t Invalid Verifying Person Name.";

getch();

return;

}

strupr(chr\_verifyingperson);

if(strlen(chr\_verifyingperson)<1||strlen(chr\_verifyingperson)>30)

{

check=0;

cout<<"\t\n The Verifying Person's Name is either blank or greater than 30 characters. Please try again.\n";

getch();

}

}while(!check);

do

{

cout<<"\n Please enter the Deposit Amount while opening a New Account :";

check=1;

gets(tran\_acc);

t\_bal=atof(tran\_acc);

ibalance=t\_bal;

if(strlen(tran\_acc)<1)

{

cout<<"\n Invalid Transaction value. Exiting form the current Menu.\n";

getch();

return;

}

if(ibalance < 1000)

{

check=0;

cout<<"\t\n The Minimum Deposit Amount should be' Rs.1000'. Please try again. \n";

getch();

}

}while(!check);

do

{

cout<<"\n Do you want to save the record?(y/n):";

choice=getche();

choice=toupper(choice);

}

while(choice!='N' && choice!='Y');

if(choice=='N' || choice=='n')

{

cout<<"\n The Customer Account is not created\n. Please continue with the application.\n";

getch();

return;

}

float t\_amount,interest\_accured;

t\_amount=ibalance;

interest\_accured=0.0;

char t\_tran,typetransaction[10];

t\_tran='D';

strcpy(typetransaction," ");

newrec.adddetails(retrieve\_accno,retrieve\_custname,retrieve\_address,ibalance);

adddetails(retrieve\_accno,day1,month1,year1,t\_tran,typetransaction,interest\_accured,t\_amount,ibalance);

cout<<"\n\n The New Account is successfully created.\n Please continue with the application.\n";

getch();

}

//----------------------------

void accounttransactions::showaccount(int retrieve\_accno)

{

cout<<" \n";

cout<<" \n";

int day1,month1,year1;

struct date dateval;

getdate(&dateval);

day1 =dateval.da\_day;

month1=dateval.da\_mon;

year1 =dateval.da\_year;

cout<<"Date:"<<day1<<"/"<<month1<<"/"<<year1<<"\n";

cout<<"Account no."<<retrieve\_accno;

disprecords newrec;

char retrieve\_custname[30];

strcpy(retrieve\_custname,newrec.getname(retrieve\_accno));

char retrieve\_address[60];

strcpy(retrieve\_address,newrec.getaddress(retrieve\_accno));

cout<<setw(25)<<"\n Account Holder's Name:"<<retrieve\_custname;

cout<<"\nAddress : "<<retrieve\_address<<"\n";

cout<<setw(80)<<"\n----------------------------------\n";

cout<<setw(10)<<"Dated";

cout<<setw(12)<<"Details";

cout<<setw(15)<<"Deposited";

cout<<setw(12)<<"Withdrawn";

cout<<setw(10)<<" ";

cout<<setw(80)<<"Balance";

cout<<setw(80)<<"\n------------------------------\n";

}

//-----------------------------

void accounttransactions::display\_account(void)

{

system("cls");//clrscr()

char t\_acc[10];

int tran\_acc,retrieve\_accno;

disprecords obj2;

cout<<" Press 0 to go back to previous menu\n";

cout<<" Please enter Account No. you want to view:";

gets(t\_acc);

tran\_acc=atoi(t\_acc);

retrieve\_accno=tran\_acc;

if(retrieve\_accno==0)

{

cout<<"\n You have pressed 0 to exit\n";

getch();

return;

}

system("cls");//clrscr()

disprecords newrec;

accounttransactions aa;

int row=8,check;

fstream filename;

FILE \* pfile;

pfile=fopen("newrecords.dat","r");

if(pfile==NULL)

{

cout<<"\n No such Account Exists. Please create a New Account. \n";

getch();

return;

}

else if(!newrec.accountexists(retrieve\_accno))

{

cout<<"\t\n Account does not exist.\n";

getch();

return;

}

else

{

showaccount(retrieve\_accno);

filename.open("transactions.dat",ios::in);

while(filename.read((char\*)this,sizeof(accounttransactions)))

{

if(accountnumber==retrieve\_accno)

{

check=0;

cout<<setw(3)<<dday<<"/"<<mmonth<<"/"<<yyear;

cout<<setw(10)<<trantype;

if (transactions=='d')

{

cout.setf(ios::right,ios::adjustfield);

cout<<setw(15);

cout<<intamount;

cout<<setw(20);

cout<<" ";

}

else

{

cout.setf(ios::right,ios::adjustfield);

cout<<setw(25);

cout<<intamount;

cout<<setw(10);

cout<<" ";

}

cout<<setw(15);

cout.setf(ios::right,ios::adjustfield);

cout<<intbalance<<"\n";

row++;

if(row==23)

{

check=1;

row=8;

cout<<"\n\n Please continue with the application."<<"\n";

getch();

system("cls");//clrscr()

showaccount(retrieve\_accno);

}

}

}

}

filename.close();

if(!check)

{

cout<<"\n\n Press any key to continue with the application\n";

getch();

}

}

//------------------------------------

int accounttransactions::datediffer(int day1,int month1,int year1,int day2,int month2,int year2)

{

static int montharr[12]={31,28,31,30,31,30,31,31,30,31,30,31};

int days=0;

while(day1!=day2 || month1!=month2 || year1!=year2)

{

days++;

day1++;

if(day1>montharr[month1-1])

{

day1=1;

month1++;

}

if(month1>12)

{

month1=1;

year1++;

}

}

return days;

}

//---------------------------------

float accounttransactions::getinterest(int retrieve\_accno,float ibalance)

{

fstream filename;

filename.open("transactions.dat",ios::in);

disprecords newrec;

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

int day1,month1,year1,month\_day;

while(filename.read((char\*) this,sizeof(accounttransactions)))

{

if(accountnumber==retrieve\_accno)

{

day1=dday;

month1=mmonth;

year1=yyear;

ibalance=newrec.getbalance(retrieve\_accno);

break;

}

}

int day2,month2,year2;

struct date dateval;

getdate(&dateval);

day2=dateval.da\_day;

month2=dateval.da\_mon;

year2=dateval.da\_year;

float interest\_accured=0.0;

int yeardiff=year2-year1;

if((year2<year1) || (year2==year1 && month2<month1) || (year2==year1 && month2==month1 && day2<day1))

{

return interest\_accured;

}

month\_day=datediffer(day1,month1,year1,day2,month2,year2);

int months;

if(month\_day>=30)

{

months=month\_day/30;

}

else

{

months=month\_day/30;

}

if(interest\_accured==0 && yeardiff==1)

{

interest\_accured=((ibalance\*0.5)/100)\*(months);

}

else if(yeardiff>1 && yeardiff<25 && interest\_accured==0)

{

interest\_accured=((ibalance\*0.5)/100)\*(months);

}

else

{

interest\_accured=0;

}

filename.close();

return interest\_accured;

}

//---------------------------

void accounttransactions::showinterest(void)

{

system("cls");//clrscr()

char t\_acc[10];

int tran\_acc,retrieve\_accno,check;

cout<<strupr("\n important information:interest should be generated only \n once in a year.\n\n\t if you have already generated interest for an account,\n\t please ignore that account\n\t.thank you.\n");

cout<<"\n Press 0 to go back to previous menu\n";

cout<<"\n to view the transactions of the account,please enter it:";

gets(t\_acc);

tran\_acc=atoi(t\_acc);

retrieve\_accno=tran\_acc;

if(retrieve\_accno==0)

return;

system("cls");//clrscr()

disprecords newrec;

if(!newrec.accountexists(retrieve\_accno))

{

cout<<"\t\n Account does not exist.\n";

getch();

return;

}

cout<<" Press 0 to go back to previous menu\n";

cout<<" \n";

cout<<endl<<" -Please enter the Account no. to generate interest- "<<endl;

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

int day1,month1,year1;

struct date dateval;

getdate(&dateval);

day1=dateval.da\_day;

month1=dateval.da\_mon;

year1=dateval.da\_year;

cout<<" Date:"<<day1<<"/"<<month1<<"/"<<year1<<"\n";

cout<<" Account no."<<retrieve\_accno<<"\n";

char retrieve\_custname[30];

char retrieve\_address[60];

float ibalance;

strcpy(retrieve\_custname,newrec.getname(retrieve\_accno));

strcpy(retrieve\_address,newrec.getaddress(retrieve\_accno));

ibalance=newrec.getbalance(retrieve\_accno);

cout<<" Customer name: "<<retrieve\_custname;

cout<<"\n Customer Address: "<<retrieve\_address;

cout<<"\n Bank Balance: "<<ibalance;

float interest\_accured;

interest\_accured=getinterest(retrieve\_accno,ibalance);

cout<<"\n\tInterest generated: "<<interest\_accured;

getch();

ibalance=ibalance+interest\_accured;

disprecords obj2;

obj2.updatebalance(retrieve\_accno,ibalance);

adddetails(retrieve\_accno,day1,month1,year1,'d',"interest",interest\_accured,interest\_accured,ibalance);

}

//--------------------------------------------------

void accounttransactions::transaction(void)

{

system("cls");//clrscr()

char t\_acc[10];

int tran\_acc,retrieve\_accno,check;

cout<<" Press 0 to go back to previous menu.\n";

cout<<" To view the transaction of the acccount,please enter it:";

gets(t\_acc);

tran\_acc=atoi(t\_acc);

retrieve\_accno=tran\_acc;

if(retrieve\_accno==0)

return;

system("cls");//clrscr()

disprecords newrec;

if(!newrec.accountexists(retrieve\_accno))

{

cout<<"\t\n account does not exist.\n";

getch();

return;

}

cout<<" press 0 to go back to previous menu.\n";

cout<<" \n";

cout<<"\n -make correct entry for the transaction below-\n";

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

int day1,month1,year1;

struct date dateval;

getdate(&dateval);

day1=dateval.da\_day;

month1=dateval.da\_mon;

year1=dateval.da\_year;

cout<<" date:"<<day1<<"/"<<month1<<"/"<<year1<<"\n";

cout<<" account no."<<retrieve\_accno<<"\n";

char retrieve\_custname[30];

char retrieve\_address[60];

float ibalance;

float interest\_accured=0.0;

strcpy(retrieve\_custname,newrec.getname(retrieve\_accno));

strcpy(retrieve\_address,newrec.getaddress(retrieve\_accno));

ibalance=newrec.getbalance(retrieve\_accno);

cout<<" customer name:"<<retrieve\_custname;

cout<<"\ncustomer address:"<<retrieve\_address;

cout<<"\nbank balance:"<<ibalance;

char trandetails,typetransaction[10],tm[10];

float t\_amount,t\_amt;

do

{

cout<<"\n please enter d for deposit or w for withdrawal of amount:";

trandetails=getche();

if(trandetails=='0')

{

cout<<"\n\n you have pressed 0 to exit";

getch();

return;

}

trandetails=toupper(trandetails);

}

while(trandetails!='W' && trandetails!='D');

do

{

cout<<"\n the transaction type is either cash or cheque. .\n";

check=1;

cout<<" (cash/cheque):";

gets(typetransaction);

strupr(typetransaction);

if(typetransaction[0]=='0')

{

cout<<"\n\n you have pressed 0 to exit";

getch();

return;

}

if (strlen(typetransaction)<1|| (strcmp(typetransaction,"CASH") && strcmp(typetransaction,"CHEQUE")))

{

check=0;

cout<<"\n the transaction is invalid.please enter in cheque or cash.\n";

getch();

}

}

while(!check);

do

{

cout<<"\n please enter the transaction amount: \n";

check=1;

cout<<"\n";

cout<<" amount: Rs.";

gets(tm);

t\_amt=atof(tm);

t\_amount=t\_amt;

if(t\_amount<1 || (trandetails=='w' && t\_amount>ibalance))

{

check=0;

cout<<"\n either amount is not a numeric value or\n it is blank or\n you are trying to withdraw more amount than in account. . . \n";

getch();

}

}while(!check);

char choice;

do

{

cout<<"\n save changes made in transaction details?(y/n)";

choice=getche();

choice=toupper(choice);

}

while(choice!='N'&& choice!='Y');

if(choice=='N'&& choice=='n')

{

cout<<"\n transaction is not saved";

getch();

return;

}

if(trandetails=='d')

{

cout<<"\n the amount is deposited in the bank";

getch();

ibalance=ibalance+t\_amount;

}

else

{

cout<<"\n the amount is withdrawn from the bank\n";

getch();

ibalance=ibalance-t\_amount;

}

newrec.updatebalance(retrieve\_accno,ibalance);

adddetails(retrieve\_accno,day1,month1,year1,trandetails,typetransaction,interest\_accured,t\_amount,ibalance);

}

//------------------------------------------------------

void accounttransactions::closeaccount(void)

{

system("cls");//clrscr()

char t\_acc[10];

int tran\_acc,retrieve\_accno;

cout<<"\n press 0 to go back to previous menu\n";

cout<<"\n please enter the account you want to close : ";

gets(t\_acc);

tran\_acc=atoi(t\_acc);

retrieve\_accno=tran\_acc;

system("cls");//clrscr()

disprecords newrec;

if(!newrec.accountexists(retrieve\_accno))

{

cout<<"\n you have entered invalid account or it doesn't exists\n";

cout<<"\n please try again\n";

getch();

return;

}

cout<<"\n press 0 to go to previous menu \n";

cout<<"\n closing this account\n";

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

int day1,month1,year1;

struct date dateval;

getdate(&dateval);

day1=dateval.da\_day;

month1=dateval.da\_mon;

year1=dateval.da\_year;

cout<<"\n date "<<day1<<"/"<<month1<<"/"<<year1<<"\n";

char choice;

newrec.display(retrieve\_accno);

do

{

cout<<"\n are you sure you want to close this account ?(y/n)";

choice=getche();

choice=toupper(choice);

}

while(choice!='N'&& choice!='Y');

if(choice=='N'|| choice=='n')

{

cout<<"\n this account is not closed\n";

getch();

return;

}

newrec.deleteaccount(retrieve\_accno);

deleteaccount(retrieve\_accno);

cout<<"\t\n\n rtecord deleted succesfully\n";

cout<<"\n please continue with the application . . . \n";

getch();

}

int login(void)

{

char username[9],ch;

char username1[]="banking";

int i=0;

char a,b[9],pass[]="banking";

cout<<"\n\n";

cout<<"\n \t login to the banking application.\n";

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

cout<<"\n\n enter username:";

cin>>username;

cout<<"\n \n\t please enter password to authenticate yourself:";

fflush(stdin);

do

{

ch=getch();

if(isalnum(ch))

{

b[i]=ch;

cout<<"\*";

i++;

}

else

if(ch=='\r')

b[i]='\0';

else if(ch=='\b')

{

i--;

cout<<"\b\b";

}

}

while(ch!='\r');

b[i]='\0';

fflush(stdin);

if((strcmp(b,pass)==0)&&(strcmp(username1,username)==0))

{

cout<<"\n \n\t you have entered succesfully";

return(1);

}

else

{

cout<<"\n incorrect username or password";

cout<<"\n ";

return(0);

}

}

int main(void)

{

system("cls");//clrscr()

int val,ch;

a:val=login();

if(val==0)

{

cout<<"\n \t want to try again ";

cout<<"\t 1.try again";

cout<<"\t 2.exit";

cout<<"\n enter your choice and press enter";

cin>>ch;

if(ch==1)

{

system("cls");//clrscr()

goto a;

}

else

{

exit(0);

}

}

menus obj1;

obj1.showmenu();

return 0;

}