

**Programming Lab**

**Electrical Engineering Department**

**EE-20A**

**ATM Project**

**Submitted to:**

Sir Haroon Ibrahim

**Submitted by:**

Muhammad Qasim Bukhari(210401068)

Hira Nabeel (210401064)

Abdullah Nawaz (210401037)

**Abstract /Description:**

This program intends to simulate the working of an atm machine in which there are two ways to login. First, as admin and second, as user. The admin can view all accounts, find them by name, by type, including the option to add and/or delete accounts. The user on the other hand, can login, check their balance, view their account’s info, apply for a loan, and edit their pin all while being able to withdraw, deposit and transfer money. Aside from this, when the user is low on money, the machine will prompt the user to get a loan. The machine also can distinguish two accounts with the same name but different pins. And finally, the machine can also tell if the account exists or not when logging in.

**Limitation: if two users were to have the same pin, then the all the handling of cash will be done for the account that is higher in position in the database file. We couldn’t bring about a solution for this.**

**List of Functionalities**

1. **Login as admin**

* Add account in database
* Print database
* Searchacc by names
* Search account by type (savings / current )
* Delete an account from database

1. **Login as standard user**

* Withdraw
* Deposit
* Transfer
* Check remaining balance
* Edit pin (the one which was used to login)
* Apply for loan
* Print account info (holder, type etc.)

**Note:** some of the functionalities listed may seem bizarre (loan for example), but through research we know for a fact that atm can have/ does have all of the functionalities mentioned in the **“login as user”** section

**Topic used for projects Implementation:**

* Classes
* Inheritance
* Keyword access specifiers
* member functions
* string streams
* generating random numbers
* Char array concatenation and comparison
* Carriage return
* color of console output
* char and ascii codes
* Functions and their prototypes, declarations, and definitions
* Use of scope resolution operator
* Loops : do while, while & for loop
* Switch case statements
* Nested if else
* Input and output manipulation
* ~~Check Total transactions made/left for savings account~~
* File handling
* Character pointer

Note : the option to “Check Total transactions made/left” was mentioned in the original proposal but we were not able to integrate it into our code because:

1. if we were to store the left transactions into and int then it would have to be initialized before being used and due to this, whenever the project would have been compiled again the value would be re-initialized therefore giving user the ability to make as many transactions as they want
2. Usually, savings accounts are given a certain amount of monthly transaction such that the transaction they can make reset every month, but we couldn’t think of simulating a condition in which the number of transactions left could be reset.

**Algorithm / Pseudocode**

* Start
* declare char opt, declare &define char gap array { '-', '\0' };
* declare string cpin, cpin1 ,actype,holder
* declare fstream dbfile, pinfile,tfile
* char inf[60], character pointer \*charp \*charp1
* define beeps func with 3 Beep(2200,45)
* define progress function
* { char a=219, char b =176
* For I from 0-25 with i++
* Print b
* Carriage return
* For i from 0-25 with i++
* Print a & add sleep(30)
* }
* Make int main prototype
* Make class ATM
* {

**Protected member:**

* double amount, Tot, bamnt
* Initialize all to zero
* Char name[20],type[10],findp[10],findt[40],findn[20]

**Public member:**

* Void database,credential,showBal,display, withdraw,deposit, transfer,receipt,loan,edpin
* Function prototype declared
* }; end of class atm
* Make class admin which inherits public members of atm
* {

**Public members**

* Void adm, delAcc
* }; //end of class admin
* Obj of admin a
* Obj of ATM A
* Database() definition
* {
* Open “database.txt”
* Ask user for name, account type,
* Ask user for auto generated pin through srand or user entered pin
* Store all info with gap array in between and add endlin
* Create file from the pin entere and ask user to enter initial cash
* Enter cash into this file opened using pinfile variable
* Ask admin if they want to make another entry
* Put all this in do while loop
* }
* Credential definition
* {
* Do
* Ask user to enter pin
* If pin is 8123813
* Print progress bar and call adm() member function through object a
* A.adm();
* Else if pin is 0
* Exit code with error(0)
* End of else if
* do
* When pin is entered, ask for confirmation
* If option n
* Ask user to re enter pin
* While opt!=y
* }
* Ask for account type and store in findt array
* Store type into findp char array
* Open database file
* While dbfile is putting data into info array
* {
* Define charp= str str (inf,findt)
* Define charp1=strstsr(inf,finp)
* If (both found)
* {
* Store value of findp into cpin to open cpin.txt
* Break loop
* }
* }
* Pinfile.close
* Database file close
* } end of while loop
* If (either not found)
* {
* Print “error”
* And call main function so program restarts
* }
* showBal() definition
* {
* Open pinfile in reading mode
* Extract value into double amount
* Close file
* Print bamnt
* }
* display() definition
* {
* Open database file
* While loop( dbfile>>inf)
* {
* Define charp= str str (inf,findt)
* Define charp1=strstsr(inf,finp)
* If (both found)
* {
* Store value of fint in actype and findp in cpin
* Break loop
* }
* Getline from dbfile into holder till hypher
* Holder.erase null character for 1 length
* Print holder, type , cpin
* } end of displ funciton
* withdraw() definition
* {
* Open pinfile
* Ask user to choose from options
* And set amount equal to that option
* Extract money from pinfile into bamnt
* Subtract bamnt and amount and store their value into Tot
* Close pinfile
* Reopen in trunc mode
* And truncate Tot into it
* } end of withdraw func
* deposit () definition
* {
* Ask user for amount to deposit and store it in amount
* Open pinfile in read mode
* Extract amount into bamnt
* Store sum of bamnt and amount int tot
* Close pinfile
* Reopen in trunc mode
* Truncate Tot into it
* } end of deposit function
* transfer () definition
* {
* Open pinfile of person who has logged in
* Ask user for all details of other person
* User char pointer to find name, type ,and pin
* If all three found on 1 line
* Open pinfile of other user
* Ask user for amount to be transferred and store it in amount
* Extract money from user and store in bamnt
* Subtract amount from bamnt
* And then extract amount from other user(to whom the money is being transferred ) and add store it in bamnt
* Now add bamnt and amount and truncate it into pinfile of the user to whom money is being transferred
* } end of transfer func
* receipt () definition
* {
* Open database file
* Use file loop to read file line by line
* If the logged in user’s pin and type are found on same line
* Break loop
* And store users name in string using getline which read till hyphen
* Print holder,type, and remaining balance by extracting it from pinfile of the logged in user
* } end of receipt
* loan () definition
* {
* Char arr[4] (‘y’,’n’,’y’,’n’)
* Srand time null
* Int x= rand % 4
* If arr[x] = ‘y’
* Print loan approved
* Ask user for amount he need
* Store it in amount
* Open pinfil in read/ trunc mode
* Extract amount from users pinfile
* Store sum in Tot
* And add tot back to pinfile
* Print receipt
* }
* edpin() definition
* {
* Store pin into cpin using string streatm and equate it to oldpin string
* **DO**
* {
* Ask user to enter their own pin
* Or generate using srand
* If srand chose : srand time null
* Pin= rand% 9000 + 1000
* Ask for pin confirmation
* Cin opt
* }while(opt!=y)
* Use string stream to convert new pin tos string and store it into cpin1
* Store cpin1 in nPin
* Make two char arrays cp and cp1
* Use string stream to store cpin in cp
* And cpin1 in cp
* Add .txt at end of each usin strcat
* Rename (cp,cp1)
* Open database file
* Initialize dash to zero
* Use while loop to read file line by line
* Use for loop to find hyphens
* When one hypher found add +1 to dash int
* When two hyphens found
* If line with pin to be updated is found, set it as targline
* Store line uptill that point into char
* Ask user for pin
* Add this new pin at end of the and store it into updated line
* Open temp.txt in write mode
* Use another while loop to compare line
* If line is equal to target line
* Store updated line into temp file or else store standard line
* Remove database.txt
* And rename temp.txt to database.txt
* }end of edpin
* adm() definition
* {
* Give admin option to
* 1) print database 2) search by name 3) seatch by type 4) add account 5) delete acc 6) exit
* Cin opt
* Case 1
* Use while loop to read file line by line
* Print all lines
* And for every line add increment into count int
* Print count integer
* Break;
* Case 2
* Ask user to enter name in char array
* Use str str to find all itereations through while loop
* Print all accounts and count
* Break;
* Case 3
* Same as case 2 but search for type instead of name
* Case 4:
* Call database()
* Break;
* Case 5
* call delAcc()
* break;
* case 6
* call main();
* }//end of func definition
* delAcc() definition
* {
* Open database gile
* Ask admin to enter name
* Then ask them to enter type and finally ask for pin
* Store all of this in different strings
* And add hyphen between all lines
* Equate this new line with hyphens equal to target line
* Open temporary txt file
* Use while lloop to read file line by line and find this line
* If line is found, skip it
* If the current line is not equal to target line (in while loop)
* Store that line into temp file
* Remove database.txt
* Rename temp.txt to database.txt
* }
* main() definition
* {
* **Do**
* Color 0E to get yellow text
* System clear screen
* Print “welcome”
* Call cred function
* Ask user to choose option
* 1)with 2) dep 3)trans 4)balance 5) edit pin 6) loan 7)info 8) exit
* Ask user for input of char opt
* Switch (opt)
* Case 1:
* A1.withdraw();
* Print “collect card”
* Beeps()
* A1.receipt
* Break;
* Case 2:
* A1.deposit();
* Print “collect card”
* Beeps()
* A1.receipt
* Break;
* Case 3:
* A1.transfer ();
* Print “collect card”
* Beeps()
* Break:
* Case 4:
* A1.showbal ();
* Print “collect card”
* Beeps()
* Break:
* Case 5:
* A1.edpin();
* Print “collect card”
* Beeps()
* Break;
* Case 6:
* A1.loan();
* Print “collect card”
* Beeps()
* Break;
* Case 7:
* A1.display();
* Print “collect card”
* Beeps()
* Break;
* Case 8:
* Print “collect card”
* Beeps()
* Exit(0)
* Break:
* System(“pause”)
* }while(1)
* end

**Important instruction: The admin’s pin is 8123813**

**Note : the code may note run due to it being copied in chunks to get syntax highlights, so please refer to the .cpp or .txt file attached**

**The code may also not work on repl due to use of windows.h library**

**Code:**

//the admin's pin is 8123813

// comments which have to be written/repeaeted too many times

// 1) string streams are cleared as their value has to be flushed before use

// 2) all files are closed before opening to either execute pending functions or to enable opening file in another mode

// 3) ss streams are used as option for accc type to enter option instead of typing out "savings" or "current"

// 4) char pointers are used to find one array in another one

#include <iostream> //standard c++ pre processor directive for input and output

#include <fstream> //for use of functions such as set w

#include <stdlib.h> //used for rand and srand function

#include <string>

#include <stdio.h> //used for remove/rename function

#include <string.h> //enables use of strings and other functions like strstr

#include <time.h> //to be used with srand to seed random digits

#include <sstream> //for string / char array convertion into int and vice versa

#include <iomanip> //for tinkering with output/input eg using setw for spaces

#include <windows.h> // used for beep function

using namespace std;

char opt; //char used so that it has mutliple use-cases e.g options and/or confirmations

char gap[2] = { '-','\0' };

//this gap will be put into database file

stringstream ss;

string cpin, cpin1;

//strings for opening pin files of users and string stream for conversion of data

string actype, holder;

//actype to print account type and holder to print holders name

fstream dbfile, pinfile, tfile;

//dbfile for database, pinfile for pins and tfile for temporary file;

char inf[60]; // array to read and print files line by line

char\* charp, \* charp1;

//character pointers to be used for str str function

// function for atm beeps

void Beeps()

{

Beep(2200, 45);

Beep(2200, 45);

Beep(2200, 45);

}

//function to print progress bar

void progress()

{

char a = 219; //character for gray block

char b = 176; //charcater for dotted transparent square

cout << "\t\t\t\t\tProcessing......\n";

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

for (int i = 0; i <= 25; i++)

cout << b;

cout << "\r"; //carriage return

//moves active positoin of to the initial pos, which in this case would be after 5 \t 's

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

for (int i = 0; i <= 25; i++)

{

cout << a;

Sleep(30); //stops for loop for 30 miliseconds before second execution

}

}

int main(); //prototype of main function defined so it can be called afterward

class ATM

{

protected:

double amount = 0, Tot = 0, bamnt = 0;

char name[20], type[10],findp[10], findt[40], findn[20];

int pin;

//protected members are used since they can be accessed by derived classes too

public:

void database();

void credentials();

void showBal();

void display();

void withdraw();

void deposit();

void transfer();

void receipt();

void loan();

void edpin();

//all member functin declared

//yet to be defined

}; //termination of atm's blueprint

class admin :public ATM

{

public:

void adm();

void delAcc();

//functions declared

}; //termination of admin's blueprint

admin a;

ATM A1;

//objects of both classes made as they are to be used in function definitions

//function to add account into database file

void ATM::database()

{

do {

dbfile.open("Database.txt", ios::out | ios::app);

if (dbfile.is\_open())

cout << "\nDatabase opened\n";

cout << "\nenter Your name: ";

cin >> name;

cout << "Account type ? \n1. savings\n2. current\n";

cout << "option: ";

cin >> opt;

ss.clear();

if (opt == '1')

{

actype = "savings";

ss << actype;

ss >> type;

}

else if (opt == '2')

{

actype = "current";

ss << actype;

ss >> type;

}

//stream used to prevent typing out savings/current

cout << "\n1.create you own pin or \n2. generate it automatically ?";

cout << "\noption : ";

cin >> opt;

if (opt == '1')

{

cout << "enter pin: ";

cin >> pin;

}

else

{

cout << "generating randomized pin";

srand(time(NULL));

pin = rand() % 9000 + 1000;

//generates random 4 digit pin between 1000-9999 using unix clock as reference

cout << "\nyour pin is " << pin << endl;

}

dbfile << name << gap << type << gap << pin << endl;

//all data taken from user and added to file with gap array added in between

dbfile.close(); //file closed to for writing operation to be executed

cout << "\n\naccount created\n";

ss.clear();

ss << pin;

ss >> cpin;

//integer pin converted into string to create its file

pinfile.open(cpin + ".txt", ios::out | ios::app);

cout << "enter the amount that will be present in the account by default: ";

cin >> amount;

pinfile << amount;

pinfile.close(); //file closed to execute functions

cout << "Want to make another data entry? (y/n) \nOption: ";

cin >> opt;

} while (opt == 'y' && opt != 'n');

//loop reruns if user wants to make another entry

} //end of database function

void ATM::credentials()

{

cout << setw(25) << "\nNote: \n" << "1) To login as admin, enter admin's password instead of pin \n";

cout << setw(25) << "2) to exit code enter 0 \n\n";

cout << "\nEnter your pin: ";

cin >> pin;

do {

if (pin == 8123813)

{

progress();

a.adm(); //admin's member function called

}

else if (pin == 0)

{

cout << "exiting...\n\n";

exit(0);

//exit's code without error val

}

cout << "\nconfirm pin ? (y/n): ";

cin >> opt;

if (opt == 'n' && opt != 'y')

{

cout << "re-enter you pin :";

cin >> pin;

}

} while (opt != 'y');

cout << "Account type ? \n1. savings\n2. current\n";

cout << "option: ";

cin >> opt;

ss.clear();

if (opt == '1')

{

actype = "savings";

ss << actype;

ss >> findt;

}

else if (opt == '2')

{

actype = "current";

ss << actype;

ss >> findt;

}

//string streams used to prevent typing out the whole text of type

ss.clear(); //string stream cleared to stop previous value from interfering

ss << pin;

ss >> findp;

//conversion used since str str allows char arrays as argument only

dbfile.close();

dbfile.open("Database.txt", ios::in);

while (dbfile >> inf)

{

charp = strstr(inf, findp);

charp1 = strstr(inf, findt);

//charactet pointer compares addresses

if (charp && charp1)

{

cpin = findp;

pinfile.open(cpin + ".txt", ios::in);

break;

}

}

pinfile.close();

dbfile.close();

if (!charp || !charp1)

{

Beep(1400, 300);

cout << "Erros! No account found try again\n\n";

system("pause");

main(); //main function called if entered acc is not foud

}

}

void ATM::showBal()

{

progress(); //progress bar

system("cls");

cout << setw(65) << "Balance";

pinfile.open(cpin + ".txt", ios::in);

pinfile >> bamnt;

pinfile.close();

cout << "\nYour Account's Balance is ";

cout << bamnt;

pinfile.close();

//extracts value from pinfile and prints it

}

void ATM::display() //function to print user's data

{

progress();

system("cls");//clear previous screen

cout << setw(65) << "Account Information\n ";

dbfile.close(); //to close if file was opened in anyother mode

dbfile.open("Database.txt", ios::in);

dbfile.seekp(0); //moves cursor to starting position

getline(dbfile, holder, '-');

while (dbfile >> inf)

{

ss.clear(); //string stream cleared to stop previous value from interfering

ss << pin;

ss >> findp;

charp = strstr(inf, findp);

charp1 = strstr(inf, findt);

if (charp && charp1)

{

char actype[10];

ss << findt;

ss >> actype;

cpin = findp;

cout << endl;

break;

}

getline(dbfile, holder, '-');

holder.erase('\0', 1);

}

cout << "Account Holder: " << holder;

cout << "\nAccount type: " << actype;

cout << "\nPin: " << cpin;

cout << endl;

}

void ATM::withdraw()

{

progress();

system("cls");

cout << setw(65) << "WITHDRAWAL\n";

cout << "Choose amount to be withdrawn: \n";

cout << setw(40) << "1. 1000 Rs" << setw(20) << "2. 2000 Rs\n\n";

cout << setw(40) << "3. 5000 Rs" << setw(21) << "4. 10000 Rs\n\n";

cout << setw(56) << "5.Choose Your Own Amount \n";

cout << "option chosen: ";

cin >> opt;

switch (opt)

{

case '1':

amount = 1000;

break;

case '2':

amount = 2000;

break;

case '3':

amount = 5000;

break;

case '4':

amount = 10000;

break;

case '5':

cout << "enter amount: ";

cin >> amount;

break;

}

pinfile.open(cpin + ".txt", ios::in);

pinfile >> bamnt;

pinfile.close();

//extcracting amount from pinfgile

progress();

Tot = bamnt - (amount + (amount \* 0.03)); //3% tax deducted

if (bamnt > Tot)

{

pinfile.open(cpin + ".txt", ios::out | ios::trunc);

system("cls");

cout << "\nTransaction of " << amount << " rs successful\n";

pinfile << Tot;

cout << endl;

if (Tot <= 2000)

{

//if less than 2k balance is remaining bank prompts user for load

cout << "You balance is running low !! you only have less than 2000 left, would you like to apply for a loan: (y/n) ";

cin >> opt;

if (opt == 'y');

loan(); //loan function called

}

}

else if (bamnt <= Tot)

{

cout << "\nInsufficient balance\n";

cout << "would you like to apply for a loan ? (y/n) ";

cin >> opt;

if (opt == 'y')

loan();

else if (opt == 'n')

{

Beeps();

system("pause");

cout << "Collect your card";

main();

}

}

}

void ATM::deposit()

{

progress();

system("cls");

cout << setw(65) << "DEPOSIT\n";

cout << "Choose amount to be Deposited: \n";

cout << setw(40) << "1. 1000 Rs" << setw(20) << "2. 2000 Rs\n\n";

cout << setw(40) << "3. 5000 Rs" << setw(21) << "4. 10000 Rs\n\n";

cout << setw(56) << "5.Choose Your Own Amount \n";

cout << "option chosen: ";

cin >> opt;

switch (opt)

{

case '1':

amount = 1000;

break;

case '2':

amount = 2000;

break;

case '3':

amount = 5000;

break;

case '4':

amount = 10000;

break;

case '5':

cout << "enter amount: ";

cin >> amount;

break;

}

pinfile.open(cpin + ".txt", ios::in);

pinfile >> bamnt;

pinfile.close();

//extracting cash from pinfile

Tot = bamnt + amount;

pinfile.open(cpin + ".txt", ios::out | ios::trunc);

//truncate mode used to erase previous amount and store Tot into file

progress();

cout << "\nDepost of " << amount << " successful\n";

pinfile << Tot;

cout << endl;

dbfile.close();

pinfile.close();

}

void ATM::transfer() //to tranfer from one user to another

{

progress();

system("cls");

cout << setw(65) << "Fund Transfer\n\n";

cout << "Note: To Transfer, you must know the pin of other user for cash transfer\n";

cout << "\nEnter other user's pin: ";

cin >> findp;

cout << "Other person's Account type ? \n1. savings\n2. current\n";

cout << "option: ";

cin >> opt;

ss.clear();

if (opt == '1')

{

actype = "savings";

ss << actype;

ss >> findt;

}

else if (opt == '2')

{

actype = "current";

ss << actype;

ss >> findt;

}

cout << "Enter other user's name: ";

cin >> findn;

dbfile.close(); //to close if file was opened in anyother mode

dbfile.open("Database.txt", ios::in);

while (dbfile >> inf)

{

charp = strstr(inf, findp);

charp1 = strstr(inf, findt);

if (charp && charp1)

{

ss.clear();

ss << findn;

ss >> holder;

cout << "Account Holder: " << holder; //check added

cout << "\nAccout Type: " << findt;

break;

}

};

//loop to find other user who's details are mentioned

cout << "\nDo you want to transfer to " << holder << " (y / n) : ";

cin >> opt;

if (opt == 'y' && opt != 'n')

{

pinfile.close();

pinfile.open(cpin + ".txt", ios::in);

pinfile >> bamnt;

cout << "Enter The amount you want to transfer ";

cin >> amount;

if (amount >= bamnt)

{

cout << "\nInsufficient balance\n";

Beep(1800, 100);

cout << "apply for loan ? (y/n) ?";

cout << "\nOption: ";

cin >> opt;

do {

if (opt == 'y')

loan();

else if (opt == 'n')

exit(0);

} while (opt != 'y' || opt != 'n');

}

//to check for user's balance

Tot = bamnt - amount;

pinfile.close();

pinfile.open(cpin + ".txt", ios::out | ios::trunc);

pinfile << Tot;

pinfile.close();

ss.clear();

ss << findp;

ss >> cpin1;

pinfile.close();

pinfile.open(cpin1 + ".txt", ios::in);

pinfile >> bamnt;

pinfile.close();

pinfile.open(cpin1 + ".txt", ios::out | ios::trunc);

Tot = bamnt + amount;

pinfile << Tot;

cout << "\nTransfer Successful\n";

pinfile.close();

}

dbfile.close();

}

void ATM::receipt()

{

system("pause");

cout << "\n collect cash/receipt \n";

Beeps();

system("pause");

system("cls");

cout << setw(65) << "RECEIPT\n";

dbfile.close(); //to close if file was opened in another mode

dbfile.open("Database.txt", ios::in);

dbfile.seekp(0);

getline(dbfile, holder, '-'); //reads file till hyphen

while (dbfile >> inf)

{

ss.clear(); //string stream cleared to stop previous value from interfering

ss << pin;

ss >> findp;

charp = strstr(inf, findp);

charp1 = strstr(inf, findt);

if (charp && charp1)

{

ss.clear();

char actype[10];

ss << findt;

ss >> actype;

cpin = findp;

cout << endl;

break;

}

getline(dbfile, holder, '-');

holder.erase('\0', 1);

}

cout << "Account Holder: " << holder;

cout << "\nAccout Type: " << actype;

cout << "\nRemaining Balance " << Tot;

cout << endl;

cout << setw(75) << "Thank You For Banking With Us\n\n";

dbfile.close();

}

void ATM::loan()

{

progress();

system("cls");

cout << setw(65) << "Loan\n\n";

char App[4] = { 'y','n','n','y' };

cout << "How much amount do you need for loan ";

cin >> amount;

cout << "Confirm Application for loan ? (y/n) ";

cin >> opt;

srand(time(NULL));

int x = rand() % 4; //to seed random numbers from 0-4 exclusive of 4

if (App[x] == 'y')

{

cout << "loan approved\n";

pinfile.close(); //closed any prev iteration of file opened

pinfile.open(cpin + ".txt", ios::in);

pinfile >> bamnt;

pinfile.close();

pinfile.open(cpin + ".txt", ios::out | ios::trunc);

Tot = bamnt + amount;

cout << "\nyour final amount would be " << Tot << endl;

pinfile << Tot;

receipt();

}

else

{

cout << "Sorry you loan wasn't approved";

Beep(1400, 300);

}

}

void ATM::edpin()

{

ss.clear();

ss << pin;

ss >> cpin;

string oldpin = cpin;

//new string used for clarity

do {

progress();

system("cls");

cout << setw(65) << "Pin Editting\n\n";

cout << "\n1.create new pin on your own or \n2.Get auto generated pin ?";

cout << "\noption: ";

cin >> opt;

if (opt == '1')

{

cout << "enter new pin: ";

cin >> pin;

}

else

{

cout << "generating randomized pin";

srand(time(NULL)); //use of unix clock to generate random numbers

pin = rand() % 9000 + 1000; //genertes random 4 digit pin

cout << "\nyour pin is " << pin;

}

cout << "\nFinalise pin ? (y/n) ";

cin >> opt;

} while (opt != 'y');

char conv[5] = { '.','t','x','t','\0' };

ss.clear();

ss << pin;

ss >> cpin1;

string nPin = cpin1;

//newly made pin is stoed in nPin

char cp[10], cp1[10];

//character arrays to be used for renaming files as rename only takes char arrays as argument

ss.clear();

ss << cpin;

ss >> cp;

ss.clear();

ss << cpin1;

ss >> cp1;

strcat\_s(cp, conv);

strcat\_s(cp1, conv);

//concatenates converter at the end of each array

rename(cp, cp1);

//file renamed so the new pin's file has the same amount as prev pinfile

//for changing pin

string data;

string currPin;

string targline;

string line;

dbfile.close();

dbfile.open("Database.txt", ios::in);

while (dbfile >> line)

{

int dash = 0;

int startp = 0;

//loop 1 for setting position of start pin

for (int i = 0; i < line.length(); i++)

{

if (line[i] == '-')

dash++;

//if dash is found, increment the value of dashes found

if (line[i] == '-' && dash == 2)

startp = i;

}

currPin = line.substr(startp + 1);

if (currPin == oldpin)

{

//loop 2 for writing new pin

for (int i = 0; i < line.length(); i++)

{

if (line[i] == '-')

dash++;

//if dash is found, increment the value of dashes found

if (line[i] == '-' && dash == 2)

startp = i;

}

dash = 0;

data = line.substr(0, startp + 1);

data += nPin;

targline = line;

}//end of if statement

//the line with current pin becomes the tareted line to change

}//end of while loop

cout << "you updatd account is " << data;

string cline;

dbfile.close();

dbfile.open("Database.txt", ios::in);

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

//temp file opened in writing mode and dbfile in reading mode

if (dbfile.is\_open() && tfile.is\_open())

{

while (dbfile >> cline)

{

if (cline == targline)

tfile << data << endl;

if (cline != targline)

tfile << cline << endl;

}

}

dbfile.close();

tfile.close();

//closing both files to execute all the commands on files

remove("Database.txt");

rename("temp.txt", "Database.txt");

//deleting database file and renaming temporary file and to database file so that update file is ready to use the next time code runs

}

//all function of atm class defined

void admin::adm()

{

do {

int count = 0;

system("cls");

cout << setw(65) << "Welcome Admin\n\n";

cout << "What would you like to do: \n\n";

cout << setw(40) << "1. Print database" << setw(30) << "2. Search by name\n\n";

cout << setw(40) << "3. Search by type" << setw(35) << "4. Add acc in database\n\n";

cout << setw(43) << "5. Delete an account" << setw(17) << "6. Exit\n\n";

cout << "Option: ";

cin >> opt;

switch (opt)

{

case '1':

dbfile.close();

//closes any previously opened iteration

progress();

system("cls");

dbfile.open("database.txt", ios::in);

while (dbfile >> inf)

{

cout << inf << endl;

count++;

}

cout << "\nTotal accounts found: " << count << endl;

if (count == 0)

{

cout << "do you want to add account in database ? (y/n) ";

cout << "option: ";

cin >> opt;

if (opt == 'y')

database();

}

dbfile.close();

//reads total line of file to print accounts found

break;

case '2':

progress();

system("cls");

cout << "Enter the name by which you wish to find the account: ";

cin >> name;

cout << "\nfinding account with the name " << name << "....\n\n";

dbfile.close();

dbfile.open("Database.txt", ios::in);

//to extract accounts and find acc by comparing names

while (dbfile.getline(inf, 60))

{

charp = strstr(inf, name);

if (charp)

{

cout << inf << endl;

count++;

}

}

cout << "\naccounts found: " << count << endl;

break;

case '3':

{

progress();

system("cls");

cout << "Account type to be found ? \n1. savings\n2. current\n";

cout << "option: ";

cin >> opt;

ss.clear();

if (opt == '1')

{

actype = "savings";

ss << actype;

ss >> findt;

}

else if (opt == '2')

{

actype = "current";

ss << actype;

ss >> findt;

}

cout << "finding account with " << findt << " type " << endl;

dbfile.open("Database.txt", ios::in);

//to extract accounts

while (dbfile.getline(inf,60) )

{

charp = strstr(inf, findt);

if (charp)

{

cout << inf << endl;

count++;

}

}

cout << "\naccounts found: " << count << endl;

dbfile.close();

}

break;

case '4':

{

progress();

system("cls");

//to add acc in database

A1.database();

}

break;

case '5':

delAcc(); //functin call to delete account

break;

case '6':

{

cout << "\nExiting......\n";

Sleep(40);

main(); //returns to main function

}

break;

default:

continue;

}

system("pause");

} while (1); //causes loop to infinitely run till exited through function call or case 6

}

void admin::delAcc()

{

string P;

string line, tline;

cout << "Enter name: ";

cin >> holder;

cout << "Enter type:";

cin >> actype;

cout << "Enter pin: ";

cin >> P; //char array

ss.clear();

tline = holder + "-" + actype + "-" + P;

dbfile.close();

dbfile.open("Database.txt", ios::in);

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

//loop compares entered line with target line

while (getline(dbfile, line))

{

if (line == tline)

{

//if target line is found, line is skipped and all other lines are added to tempfile

cout << "\nAcc found , confirm deletetion ? (y/n) \n";

cout << "option: ";

cin >> opt;

do {

if (opt == 'y')

{

cout << "Account deleted successfully\n\n";

continue;

}

else if (opt == 'n')

tfile << line << endl;

} while (opt != 'y' && opt != 'n');

}

else

{

tfile << line << endl;

}

}

tfile.close();

dbfile.close();

remove("Database.txt"); //old database file removed

rename("temp.txt", "Database.txt"); //temporary file with renamed to database

}

//function of admin class defined

int main()

{

do {

system("color 0E"); //changes color of text to yello

system("cls"); //clear screen

cout << setw(63) << "Welcome To our ATM\n" << setw(66) << "Please insert your card\n";

A1.credentials(); //calls credential function to ask for login

cout << "\nPlease choose an option: \n";

cout << setw(40) << "1. Withdraw" << setw(24) << "2. Deposit\n\n";

cout << setw(40) << "3. Transfer" << setw(30) << "4. Check Balance\n\n";

cout << setw(40) << "5. Edit Pin" << setw(25) << "6. Get Loan\n\n";

cout << setw(46) << "7. Print acc info" << setw(15) << "8. Exit\n\n";

cout << "\nOption: ";

cin >> opt;

switch (opt)

{

case '1':

A1.withdraw();

cout << "\ncollect your card\n";

Beeps();

A1.receipt();

break;

case '2':

A1.deposit();

cout << "\ncollect your card\n";

Beeps();

A1.receipt();

break;

case '3':

A1.transfer();

cout << "\ncollect your card\n";

Beeps();

break;

case'4':

A1.showBal();

cout << "\ncollect your card\n";

Beeps();

break;

case'5':

A1.edpin();

cout << "\ncollect your card\n";

Beeps();

break;

case'6':

A1.loan();

cout << "\ncollect your card\n";

Beeps();

break;

case'7':

A1.display();

cout << "\nCollect your card\n";

Beeps();

break;

case'8':

cout << "\nCollect your card\n";

Beeps();

exit(0);

break;

}

system("pause");

} while (1); //loop runs infinitely till exited through case 8 or moved out of using function call

}

**Output(s):**

**Login screen**

**Text

Description automatically generated**

**Code exits when 0 is entered as pin**

**Text

Description automatically generated**

**Pin entered : 8123813**

**Logging in as admin and adding accounts in database**

Text

Description automatically generated

Adding two accounts, one with user defined pin and the other with auto generated pin:

Text

Description automatically generated

files created:

**Graphical user interface, text, application, table, Excel

Description automatically generated**

After this, 3 more acc were added into file

**Printing whole database**

**Graphical user interface, text, application

Description automatically generatedText

Description automatically generated**

**Searching account by name:**

Text

Description automatically generated

**Finding acc by type**

**Text

Description automatically generated** **Text

Description automatically generated**

**Deleting acc**

**Text

Description automatically generated**

**when option n is chosen :**

**Text

Description automatically generated**

Nothing changes, but

**when option y is chosen:**

**Text

Description automatically generated**Text

Description automatically generated

Account deleted

**When acc doesn’t exist**

**Text

Description automatically generated**

**Logging in:**

**Text

Description automatically generated**

**Withdrawal options:**

**Text

Description automatically generated**

**Withdrawing 1k,**

**Receipt:**

**A screenshot of a computer screen

Description automatically generated**

**Logged in as babar and checking balance**

Graphical user interface, text, application

Description automatically generated

Proof:

Graphical user interface, text, application, chat or text message

Description automatically generated

**Note: after this a total of 9000 were withdrawn from babar’s acc leaving him with 1000 only.. at last transaction of 500, this prompt is shown**

**Text

Description automatically generated**

**After approval of loan of 1000**

**Graphical user interface, text, application, website

Description automatically generated**

**Depositing 10k**

**Text

Description automatically generated**

:

Text

Description automatically generated

**Account info printed for Abdullah**

**Graphical user interface, text

Description automatically generated**

**Editing pin of Fazeel with proof:**

**Text

Description automatically generatedText

Description automatically generated**

**Transferring from haris to babar:**

**Money in both acc:**

Graphical user interface, application

Description automatically generated

**Text

Description automatically generated**

**Text

Description automatically generated**

**Money after transfer:**

**Graphical user interface, text

Description automatically generated**

**Graphical user interface, text, application, chat or text message

Description automatically generated**

**Repl link:** [**https://replit.com/@Around-TheThe/ATM-project#main2.cpp**](https://replit.com/@Around-TheThe/ATM-project#main2.cpp)

**Contribution of each member:**

All three members did 5 functions each and helped each other in devising a solution to get each of the functionality to work.

**Learning outcome/ New Topics learned**

* Renaming and removing text files
* Use of truncate option in file handling
* Properly generating fully random numbers
* Using random numbers to get choices through arrays
* Proper use of keyword access specifiers
* Changing color of console output
* Stopping a loop for a certain time through sleep functions
* Making beeps through beep function
* Concatenation of character arrays
* Making progress bar through loops, ascii codes, and carriage return