BASIC BANKING APPLICATION

<u>Group Project Documentation Report</u>

By BATCH-11 CS 101

Members:

Team 1:	
Amol Mandhane*	100020006
Aditya Joshi	100020002
Parul Verma	100020043
Swati Kharole	100020047
Team 2 :	
Najmuddin Ahmad Saqib *	100020036
Akash Bansal	100020050
Jeet Juneja	100020011
Savi Rakhecha	100020040
Team 3 :	
Aditya Patil *	100020017
Mahesh Avasare	100020008
Kanishka Kayathwal	100020020
Saurabh Kumar Suman	100020064
Sapavat Ani Babu	100020060

Contents :

1. Problem Statement

The Problem Statement is to design a basic banking application which allows user to create a new account, delete or modify existing account and transactions . The Application should calculate interest by 5% per annum rate which is to be deposited in account after a certain time interval decided by the bank. In Transactions part the application must provide a lower balance limit and upper transaction limit. The application must have a user friendly interface.

2. Introduction

The definition of a bank varies from country to country.A bank is a financial intermediary that accepts deposits and channels those deposits into lending activities, either directly or through capital market. A bank connects customers with capital deficits to customers with capital surpluses.

3. Design Principle

The application has to have a strong file management system along with a set of functions so as to make a user friendly interface in which various tasks can be performed such as modification,transaction,deletion,addition etc. of an account. Since it is a basic application, it does not venture into the fields of giving out loans or transfers. We tried to create a basic interface which could be used as a small level basic bank.

4. Implementation

4.1 Entering the records

For this purpose, we have made a getdata function in a class named bank in the various details of the customer are entered.

4.2 File handling

For this purpose,we have used fstream class and some use of pointers is also done in order to maintain the records entered and modified later. Almost the entire file handling has been carried out using the fstream objects.

4.3 Implementing the various operations

For this,we have defined many functions in the class involving some private as well as public functions .also we have defined some universal functions.there are some large functions such as modification, transaction, addition, deletion as well as some of the smaller functions that are being called within some other functions.
Update() is a function which was designed to update a particular record say after transaction.
The account numbers have been allocated by using a text file “a1.txt” which initially has 1000 as its entry. Whenever a new accoutn is created, assignactno() is called which returns a value from the file incremented by 1 and also writes the same onto the file. By this way, we were able to implement primary keys in our program as well as access the records directly instead of searching through the entire contents to get to a particualr record.
The del() function which performs deletion task just converts the name into a null string. This is helpful as we can still access the records directly or else deleting a record from middle would result in shifting the records and hence our search algorithm would fail.

5. Illustrative code

5.1 Getdata()

we have made this function in order to enter the details of the individual while creating a new account.

```
  void bank::getdata()//This function is used to add a new customer in the existing database .It takes whole information
from the user and stores it .
{
            system("clear");
                    char ch;
                    char tmp[100];
            cout<<" [ADDITION] ";
                    obj.actno=assignactno();
                    cout<<" \n ENTER NAME : ";
                    cin.getline(obj.name,30,'\n');
                    cout<<"\n ENTER YOUR DATE OF BIRTH ( YYYY - DD - MM) :";
            cin>>dob[0];
    testdob:   //checks validity of the entered date since it can not be more than 31.
            cin>>dob[2];
            if (dob[2] > 31)
            {
      cout<<"PLEASE RE-ENTER DATE";
                      goto testdob;
                    }
    testmob:
                    cin>>dob[1];
                    cin.get(ch);
                    if (dob[1] >12)//since month can not be more than 12.
                    {
                      cout<<"PLEASE RE-ENTER MONTH-";
                              goto testmob;
                    }
                    cout<<" \n ENTER FATHER's NAME : ";
                    cin.getline(obj.fname,30,'\n');
                    cout<<" \n ENTER PHONE NUMBER : ";
    testphone:
                    cin>>obj.phone;
                    sprintf(tmp,"%l",obj.phone);
                    if(strlen(tmp) >11)
                    {
                              cout<<"PLEASE RE-ENTER PHONE NUMBER";
                              goto testphone;
                    }
                    cin.get(ch);
                    cout<<" \n ENTER HOUSE ADDRESS : ";
                    cin.getline(obj.address,50,'\n');
                    cout<<" \n ENTER EMAIL ADDRESS : ";
                    cin.getline(obj.emailadd,25,'\n');
            obj.balance=0;
    {
```

```
                                        char temp[10];
                                        FILE *fp;
                                        system("date --rfc-3339 date>a.txt");
                                        fp = fopen("a.txt","r");
                                        fgets(temp,10,fp);//in 'date' you get the system date
                                        fclose(fp);
                                        system("rm a.txt");

                                        int temp1[10];
                                        for(int i = 0 ; i<10 ; i++)
                                                temp1[i]=atoi(&temp[i]);
                                        obj.base_date[0]= (temp1[0])*1000 + (temp1[1])*100 + (temp1[2]) *10 +
(temp1[3]);
                                        obj.base_date[1]= (temp1[5])*10 + (temp1[6]);
                                        obj.base_date[2]= (temp1[8])*10 + (temp1[9]);
          }
}
```

This code is written which also validates the data being entered by the user.
The function is called by the addition function which writes the inputred data onto the file.

5.2 transaction()

We have made a transaction function in which the system asks the user what does he wish to do including both deposition and withdrawal and it also it shows an error message if some invalid transaction is tried to be done.

A part of the code is illustrated below.

```
void bank:: transaction(int n)  // Makes the transaction and updates the record
{

        if (obj.balance==0)
                ch=2;
        else
        {

                cout<<"Enter what you wish to do:"<<endl<<"Withdrawl-1    Deposit-2"<<endl;
                cin>>ch;
        }
        switch(ch)
                {

                case 1:   cout<<"Enter the amount to be withdrawn:";
                          cin>>amount;
             flag= verify_transaction(obj.balance,amount,ch);
             if(flag == 1)
                                        {
                                        obj.balance-=amount;
                                        cout<<endl<<"collect your cash";
                                        cout<<"\n The balance now available is:"<<obj.balance<<endl;
                                        }
             else if(flag == 2)
                                        {cout<<endl<<"Insufficient funds. Sorry your transaction has been cancelled";
                                        return;
                                        }
```

```
                        break;
               case 2: cout<<"Enter the amount to be deposited:";
            flag = verify_transaction(obj.balance,amount,ch);
            if (flag==1)
               {
               cin>>amount;
                                 obj.balance+=amount;
                                 cout<<endl<<"The balance available now is:"<<obj.balance<<endl;
                       }
            else if(flag ==2)
            {
                cout<<"Invalid Transaction"<<endl;
                return;
            }
                        break;
               default:cout<<"\nEnter a valid choice!";
       }
update(obj,n);

}
```

The function uses a basic switch case and also calls a function verify_transaction() which would validate the transaction.

5.3 modification()

Here the program asks the user to change whichever he/she wants to change.

A part of the function is illustrated below.

```
void bank::modification() //TO MODIFY ANY DATA IN  THE RECORD IF NECESSARY
  {

                  cout<<"\n ENTER ACCOUNT NO. :";
                  cin>>pn;
                  ch=cin.get();
                        int pos;
                        pos=pn-1000;
                        if(pn>returnaccno())
                                 {
                                 char ch;
                                 cout<<"\n Such a record does not exist.";
                                 cin.get(ch);
                                 return;
                                 }
                        fin.seekg((pos-1)*sizeof(obj),ios::beg);
                        fin.read((char*)&obj,sizeof(obj));
                        fin.close();
                        if(obj.name[0]=='\0')
                                 {
                                 char ch;
                                 cout<<"\n This account has been deleted.";
                                 ch=cin.get();
                                 return;
                                 }
                        if(check("NAME")=='y')
                          {
```

```
                                    cout<<"\n ENTER NEW NAME : ";
                                    cin.getline(obj.name,30,'\n');
                                  }

                                if(check("PHONE NUMBER")=='y')
                                  {
                                    cout<<"\n ENTER NEW PHONE NUMBER :";
                                    cin>>obj.phone;
                                    ch=cin.get();
                                  }



update(obj,pn);
cout<<"The record has been updated. Press any key to view the record";
ch=cin.get();
view(pn);
}
```

The function makes use of a function check() which gets a character pointer as a parameter. Check() function is illustrated below.

```
char bank::check(char *s)
{
        cout<<"\n MODIFY \t "<<s<<"\t"<<"Y/N";
             char ch;
             cin>>ch;
        char ch1;
        ch1=cin.get();
             if((ch=='y')||(ch=='Y'))
                    return 'y';
             else
                    return 'n';
}
```

5.4 interest_rate() function

This work is being done using another function with a desired interest rate already set in the program by the management system.

```
void bank::interest()
{
   char temp[10];
        FILE *fp;
        system("date --rfc-3339 date>a.txt");
        fp = fopen("a.txt","r");
        fgets(temp,11,fp);//in 'date' you get the system date
        fclose(fp);
        system("rm a.txt");

   int current_date[3];
   current_date[0]= atoi(temp[0])*1000 + atoi(temp[1])*100 + atoi(temp[2]) *10 + atoi(temp[3]);
   current_date[1]= atoi(temp[5])*10 + atoi(temp[6]);
   current_date[2]= atoi(temp[8])*10 + atoi(temp[9]);


   fstream f;
```

```
    f.open("bank11.dat",ios::in|ios::binary);
    while(f.read((char*)&obj,sizeof(obj)))
    {
        int flag = compare_dates(current_date , obj.base_date);

        {
                    obj.balance+=((obj.balance*interest_rate)*flag);
                    obj.base_date[0]=current_date[0];
                    obj.base_date[1]=current_date[1];
                        obj.base_date[2]=current_date[2];
        }

        if(current_date[1]==dob[1] && current_date[2]==dob[2]) //age increment done in interest decrease the number
of processes.
                        obj.age=current_date[0]-obj.dob[0];
    }
}
void bank :: int compare_dates(int current_date[], int base_date[]){
    char temp[10];
            FILE *fp;
            system("date --rfc-3339 date>a.txt");
            fp = fopen("a.txt","r");
            fgets(temp,11,fp);//in 'date' you get the system date
            fclose(fp);
            system("rm a.txt");

    int current_date[3];
    current_date[0]= atoi(temp[0])*1000 + atoi(temp[1])*100 + atoi(temp[2]) *10 + atoi(temp[3]);
    current_date[1]= atoi(temp[5])*10 + atoi(temp[6]);
    current_date[2]= atoi(temp[8])*10 + atoi(temp[9]);




            int d;
                    if(current_date[1]==base_date[1]){
                            d=current_date[2]-base_date[2];
                            return d;
                    }
                    else
                            if(base_date[1]==1 || base_date[1]==3 || base_date[1]==5 || base_date[1]==7 ||
base_date[1]==8 || base_date[1]==10 || base_date[1]==12){
                                    d=current_date[2]+31-base_date[2];
                                    return d;
                            }
                            else if(base_date[1]==2 && base_date[0]%4 != 0){
                                    d=current_date[2]+28-base_date[2];
                                    return d;
                            }

                            else if(base_date[1]==4 || base_date[1]==6 || base_date[1]==9 || base_date[1]==11){
                                    d=current_date[2]+30-base_date[2];
                                    return d;
                            }
                            else if(base_date[1]==2 && base_date[0]%4 == 0){
                                    d=current_date[2]+29-base_date[2];
                                    return d;
                            }
```

```
}
```

5.5 Assigning account number

For the purpose of uniqueness of each account as well as accessing the accounts directly, we made a .txt file with initially stored value 1000. Whenever a new record is created, what happens is that the value is taken from the file, incremented by 1, and then returned by the function as well as overwritten onto the file.

An illustration is as follows:

```
int assignactno()
{
        ifstream fin;
        ofstream fout;
        fin.open("a1.txt");
        int accno;
        fin>>accno;
        int newac=accno+1;
        fin.close();
        fout.open("a1.txt");
        fout<<newac;
        fout.close();
        return newac;
}
```

5.6 Searching for a particular record

Assigning of the account number has been done in such a manner that we can directly access the records by placing the pointers, opening the file using ifstream objects.

```
void bank::search()//The function searches the desired account in the database file and shows its position.
{
        system("clear");
        int t=test();
             if(t==0)
             {    char ch;
                       cout<<"\n FILE IS EMPTY ! ";
                       ch=cin.get();
                       return;
             }
        cout<<"ENTER ACCOUNT NUMBER TO BE SEARCHED:";
        int n;
        cin>>n;
        if(n>returnaccno())
        {
              char ch,ch1;
              cout<<"\n SUCH A RECORD DOES NOT EXIST";
              cin.get(ch);
              cin.get(ch1);
              return;
        }
        int pos=n-1000;
        view(pos);
        return;
}
```

```
void bank::view(int pos)//The function displays any record of given position.
{		ifstream f;
		f.open("bank11.dat", ios::in| ios::binary);
		f.seekg((pos-1)*sizeof(obj),ios::beg);
		f.read((char*)&obj,sizeof(obj));
		if(obj.name[0]=='\0')
		{
			char ch,ch1;
			cout<<"\n THIS ACCOUNT HAS BEEN DELETED.";
			ch=cin.get();
			ch1=cin.get();
			return;
		}
		display();
		return;
}
```

The search function passes the position at which the record is present in the file. View() then opens the file, places the pointer, reads the record into obj and then calls display() function.

6. Testing and results

We tested the code several time and removed the errors. The most important testing was the test of data storage. Entire program is based on correct data storage. Further testing was done to verify whether each function is working perfectly or not. Some modifications were done based on the results.
Some grave mistakes cropped up while testing. First one was that at first we were assigning the account numbers using a global variable accno. But the program wouldn't re compile and would result in tons of errors. Its a totally unacceptable logical error as an application for banking has to be re compiled day in and day out.
So after that, what we did was to store the primary key in a text file and hence it was unaffected by the number of times the program gets executed.

7. Challenging section

The most challenging part of the project was file handling . We worked upon various file types and reliased that .txt file will create many troubles. So,we decided to use .dat file for data storage.
Next challenging part was designing a GUI. We worked upon many GUI languages. We used GTK+ , GTKMM , QT Creator ,QT designer , Jambi , Win32 , MFC in our studies . But finally, We realised that QT designer will be most easy to use. We studied many tutorials of QT designer availiable over internet. But , in the end , we were stuck at data verification and association of interface with the main code. So, we skipped the idea of GUI and started working on a user friendly UI in terminal. And finally , we ended up with one.
One of the other road blocks was the idea of updating the records in the file. We wanted to over write just the updated record and not touch the rest of the file. But the problem we faced was that we didn't have seek() function in the ostream class and hence a more elaborate and a less efficient way of updating records was designed which used a dynamically created array, updates in the array and then rewrites the entire thing onto the truncated file.

8. Conclusion

We have made a succesfully working basic banking application project that is very easy to understand and user friendly. We have also covered almost all the operations that have to be included in a basic banking system.we have provided the user many options including that of transactions and modification in an existing record .We have also defined a constraint for minimum balance required to be kept in a running account and a contraint for a maximum transaction amount that is considered valid by the system. The application uses a .dat file to store data. The application requires Linux base and can be run in terminal only.

References

1. www.cplusplus.com
2. www.google.co.in
3. www.cppreference.com
4. Resources available on moodle.iitb.ac.in
5. C++textbook by Cohoon
6. Wikipedia
7. Textbook of GUI design with C++ in QT.
8. Various other references of GUI