Experiment No.9 (CBS)

1. Title: Demonstrate file handling operations to create Employee Payroll.
2. Problem Statement:

Write a menu driven program that will create a data file containing the list of Employees & their Salary details in the following form

1522 Prakash 9500 4940 1900 1634 14706 1651

Kuldeep Hule 35000 18200 7000 6020 54180

…… …………………. …… ……….

Use a class object to store each set of data, access the file created and implement the following tasks

* 1. Create New Record.
  2. Determine the Employee Record of the specified Employee name
  3. Determine the Employee Record of the specified Employee id
  4. Update the Employee Salary details, whenever there is a change.

1. Objectives:
   1. To learn and understand file handling functions.
   2. To demonstrate random accessing file.
2. Outcomes:
   1. Students will be able to learn and understand file handling operations.
   2. Students will be able to demonstrate random accessing file.
3. Software/Hardware/Other Requirements:

* Any CPU with Pentium Processor / similar, 256 MB RAM/ more, 1GB HDD / more.
* Operating System – ubuntu/Fedora 64bit OS  Software: G++ compiler/ GCC compiler, Code Editor

1. Theory: A. File

* A file is a stream of bytes/ set of characters stored on some secondary storage devices.
* Types of Files:
  + Text file: A text file stores information in readable and printable form. Each line of text is terminated with an EOL (End of Line) character.
  + Binary file: A binary file contains information in the non-readable form i.e. in the same format in which it is held in memory.
* We can read as well as write data in the file whenever required.
* File handling possible with stream classes. I/O system handles file more like input output with console.
* Stream which feeds data to program through file is known as file input stream.
* Stream which stores program data to file is known as output file stream.

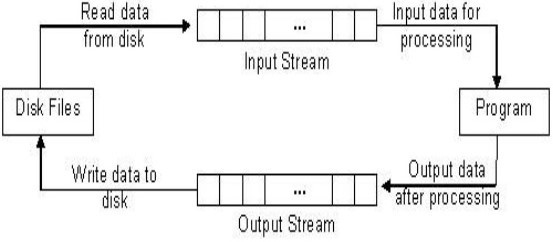


Fig: Working of Disk File IO 

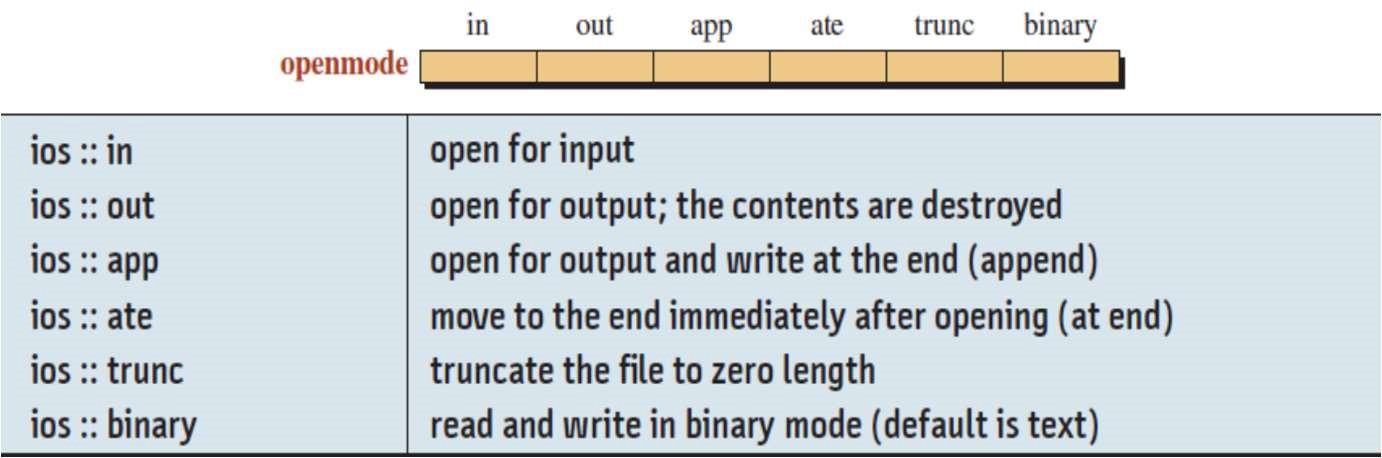
File handling provides three new datatypes:

|  |  |
| --- | --- |
| Data Type | Description |
| ofstream | This data type represents the output file stream and is used to create files and to write information to files. |
| ifstream | This data type represents the input file stream and is used to read information from files. |
| fstream | This data type represents the file stream generally, and has the capabilities of both ofstream and ifstream which means it can create files, write information to files, and read information from files. |

1. Opening a File:

* A file must be opened before you can read from it or write to it.
* Either the ofstream or fstream object may be used to open a file for writing and ifstream object is used to open a file for reading purpose only.
* Following is the standard syntax for open () function, which is a member of fstream, ifstream, and ofstream objects.

void open (const char \*filename, ios::openmode mode);

* Here, the first argument specifies the name and location of the file to be opened and the second argument of the open () member function defines the mode in which the file should be opened.

1. Closing a File:

* When a C++ program terminates it automatically closes flushes all the streams, release all the allocated memory and close all the opened files. But it is always a good practice that a programmer should close all the opened files before program termination.
* Following is the standard syntax for close () function, which is a member of fstream, ifstream, and ofstream objects.

void close ();

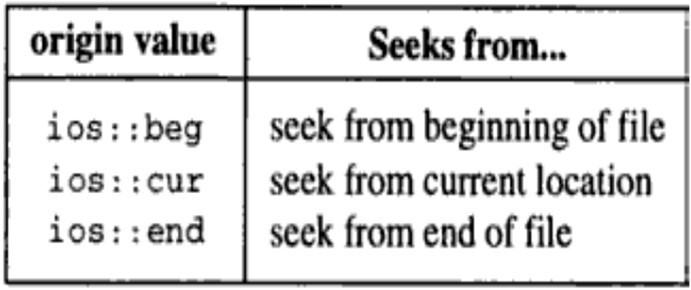
1. Working on Binary Files

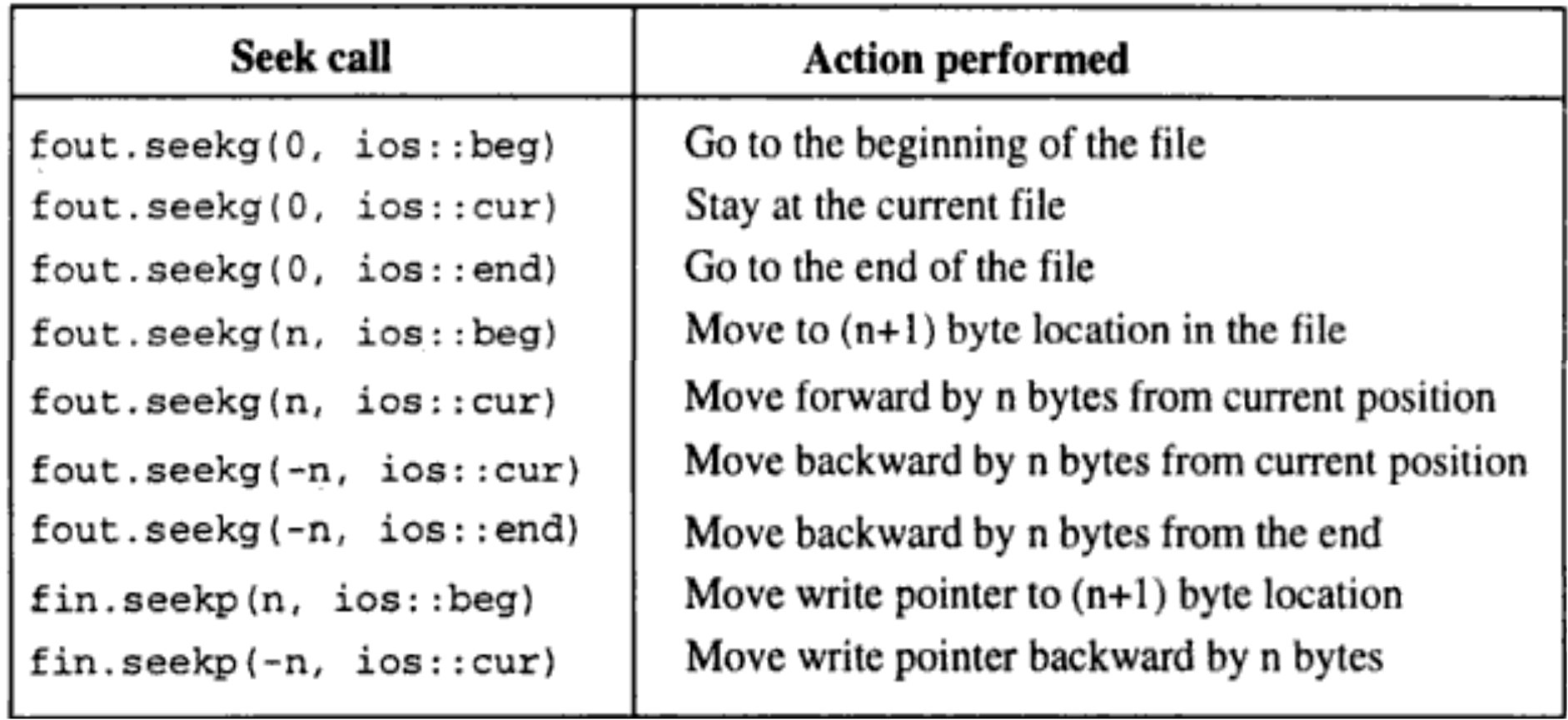
* A binary file contains information in the non-readable form i.e. in the same format in which it is held in memory.
* To achieve operations on binary files the file must be opened in ios::binary mode.
* Binary file functions:
  1. read()
  + read a block of binary data or reads a fixed number of bytes from the specified stream and store in a buffer.
  + Syntax: Stream\_object.read((char \*)& Object, sizeof(Object));
  + Example: filestream.read((char \*)&s, sizeof(s));
  1. write()
     + write a block of binary data or writes fixed number of bytes from a specific memory location to the specified stream.
     + Syntax: Stream\_object.write((char \*)& Object, sizeof(Object));
     + Example: filestream.write((char \*)&s, sizeof(s));
* Note: Both functions take two arguments: Address of the variable- (must be cast to type char \*(i.e. pointer to character type)). & Length of that variable in bytes. E. File Pointers and Manipulators:
* Each file has two pointers known as file pointers, one is called the input pointer and the other is called output pointer.
* The input pointer is used for reading the contents of of a given file location and the output pointer is used for writing to a given file location.
* Each time an input or output operation takes place, the appropriate pointer is automatically advanced.

F. Functions for Manipulations of File pointer:

* All the actions on the file pointers takes place by default.
* For controlling the movement of file pointers file stream classes support the following functions

1. seekg() Moves get pointer (input)to a specified location.
2. seekp() Moves put pointer (output) to a specified location.
3. tellg() Give the current position of the get pointer.
4. tellp() Give the current position of the put pointer.

* For example, the statement seekg(10); moves the pointer to the byte number 10.The bytes in a file are numbered beginning from zero. Therefore, the pointer to the 11 bytes in the file.
* The two seek function have the following prototype: istream &seekg(long offset, seek\_dir origin=ios::beg) ostream &seekp(long offset, seek\_dir origin=ios::beg);
* Both functions set a file pointer to a certain offset relative to the specified origin.
* The second parameter origin represents the reference point from where the offset is measured. In can be specified by using enumeration declarations (seek\_dir) given in the ios class.
* Following table shows the file pointer manipulation at various stages:



1. Algorithm for the problem Statement:

Write appropriate algorithm/ pseudocode & get it corrected from faculty.

1. Flowchart

Note: student have to draw the flowchart for this assignment and attach to the assignment.

9.

Implementation:

C++ Program for Telephone Directory

#include<iostream> #include<fstream> #include<iomanip> using namespace std;

class Employee

{ char ename[30]; int eid; double basic,da,hra,it,netsal; public:

void getdata()

{ cout<<"\n\t Enter Employee Detail:"; cout<<"\n Employee ID:"; cin>>eid; cout<<"\n Employee Name:"; fflush(stdin);

gets(ename);

cout<<"\n Employee Basic Pay:"; cin>>basic; net\_sal();

}

void putdata()

{

cout<<setw(5)<<eid<<setw(30)<<left<<ename<<setw(10)<<setprecision(7)<<l eft<<basic<<setw(10)<<setprecision(7)<<netsal<<endl;

}

char \*geteName()

{ return ename;

}

int getid()

{ return eid;

}

void net\_sal()

{ da=(0.52)\*basic; hra=(0.2)\*basic; double grosssal=da+basic+hra; it=0.1\*grosssal; netsal=grosssal-it;

}

void update(char \*nam,double sal)

{ strcpy(ename,nam); basic=sal; net\_sal();

}

};

int main()

{ char name[20],ch; int cho,flag=0,cnt=0,eid; double sal; fstream f; Employee t;

* 1. pen("EData.dat",ios::out|ios::in|ios::binary); do

{ cout<<"\n Employee Database";

cout<<"\n "; cout<<"\n 1. Create Record"; cout<<"\n 2. Display All Records"; cout<<"\n 3. Search by Name"; cout<<"\n 4. Search by ID"; cout<<"\n 5. Update Employee Name & Salary by ID"; cout<<"\n 6. Exit";

cout<<"\n "; cout<<"\n Enter your choice:";

cin>>cho; switch(cho)

{ case 1:

t.getdata();

f.write((char\*)&t,sizeof(t)); break;

case 2: cout<<"\n All Employee Records are:"<<endl; cout<<setw(5)<<"Eid "<<setw(30)<<left<<"Name "<<setw(7)<<"Basic"<<setw(7)<<"Net Salary"<<endl; f.seekg(0,ios::beg);

for(f.read((char\*)&t,sizeof(t));!f.eof();f.read((char\*)&t,sizeof(t))) t.putdata();

f.clear(); break;

case 3: cout<<"\n Enter Name for Searching:";

fflush(stdin);

gets(name);

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

for(f.read((char\*)&t,sizeof(t));!f.eof();f.read((char\*)&t,sizeof(t)))

{

if(strcmp(name,t.geteName())==0)

{

cout<<setw(5)<<"Eid "<<setw(30)<<left<<"Name "<<setw(10)<<"Basic"<<setw(10)<<"Net Salary"<<endl;

t.putdata(); flag=1;

}

}

f.clear(); if(flag==0)

cout<<"\n Record not Found.";

break;

case 4: cout<<"\n Enter Employee ID for Searching:";

cin>>eid; f.seekg(0,ios::beg); flag=0;

for(f.read((char\*)&t,sizeof(t));!f.eof();f.read((char\*)&t,sizeof(t)))

{

if(eid==t.getid())

{

cout<<setw(5)<<"Eid "<<setw(30)<<left<<"Name "<<setw(7)<<"Basic"<<setw(7)<<"Net Salary"<<endl;

t.putdata(); flag=1;

}

}

f.clear(); if(flag==0)

cout<<"\n Record not Found.";

break;

case 5: cout<<"\n Enter Employee ID for Updation:";

cin>>eid; flag=0;

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

for(f.read((char\*)&t,sizeof(t));!f.eof();f.read((char\*)&t,sizeof(t)))

{

cnt++; if(eid==t.getid())

{

flag=1;

int loc=(cnt-1)\*sizeof(t);

|  |
| --- |
| if(f.eof())  f.clear();  cout<<"\n Enter New Employee Name: "; fflush(stdin); gets(name); cout<<"\n Enter New Employee Basic:"; cin>>sal; f.seekp(loc);  t.update(name,sal); f.flush();  f.write((char \*) &t, sizeof(t)); |
| }  }  f.clear(); if(flag==0)  cout<<"\n Record not Found.";  break; case 6: return 0;  default: cout<<"\n Enter valid Choice."; break; |
| }  cout<<"\n Do you want to continue(y/n):"; cin>>ch;  } while(ch=='Y'||ch=='y'); return 0; |
| } |

Output:

Employee Database 1.

Create Record

1. Display All Records
2. Search by Name
3. Search by ID
4. Update Employee Name & Salary by ID
5. Exit

Enter your choice:1

Enter Employee Detail: Employee ID:1651

Employee Name:Kuldeep Hule Employee Basic Pay:35000

Do you want to continue(y/n):y Employee Database

1. Create Record
2. Display All Records
3. Search by Name
4. Search by ID
5. Update Employee Name & Salary by ID
6. Exit

Enter your choice:1

Enter Employee Detail: Employee ID:1652

Employee Name:Priyanka Holkar Employee Basic Pay:9500

Do you want to continue(y/n):y Employee Database

* + 1. Create Record
    2. Display All Records
    3. Search by Name
    4. Search by ID
    5. Update Employee Name & Salary by ID
    6. Exit

Enter your choice:1

Enter Employee Detail: Employee ID:1123

Employee Name:Prakash K Employee Basic Pay:12000

Do you want to continue(y/n):y Employee Database

1. Create Record
2. Display All Records
3. Search by Name
4. Search by ID
5. Update Employee Name & Salary by ID
6. Exit

Enter your choice:2

All Employee Records are:

Eid Name Basic Net Salary 1651 Kuldeep Hule 35000 54180

1652 Priyanka Holkar 9500 14706

1123 Prakash K 12000 18576

Do you want to continue(y/n):y Employee Database

1. Create Record
2. Display All Records
3. Search by Name
4. Search by ID
5. Update Employee Name & Salary by ID
6. Exit

Enter your choice:3

Enter Name for Searching:Prkash K Record not Found.

Do you want to continue(y/n):y Employee Database

1. Create Record
2. Display All Records
3. Search by Name
4. Search by ID
5. Update Employee Name & Salary by ID
6. Exit

Enter your choice:3

Enter Name for Searching:Prakash K

Eid Name Basic Net Salary

1123 Prakash K 12000 18576

|  |
| --- |
| Do you want to continue(y/n):y Employee Database   1. Create Record 2. Display All Records 3. Search by Name 4. Search by ID 5. Update Employee Name & Salary by ID 6. Exit |
| Enter your choice:4  Enter Employee ID for Searching:1651  Eid Name Basic Net Salary 1651 Kuldeep Hule 35000 54180 Do you want to continue(y/n):y  Employee Database   1. Create Record 2. Display All Records 3. Search by Name 4. Search by ID 5. Update Employee Name & Salary by ID 6. Exit |
| Enter your choice:5  Enter Employee ID for Updation:1651 Enter New Employee Name: Kuldeep Hule Enter New Employee Basic:40000  Do you want to continue(y/n):y Employee Database   1. Create Record 2. Display All Records 3. Search by Name 4. Search by ID 5. Update Employee Name & Salary by ID 6. Exit |
| Enter your choice:2 All Employee Records are: Eid Name Basic Net Salary 1651 Kuldeep Hule 40000 61920  1652 Priyanka Holkar 9500 14706  1123 Prakash K 12000 18576  Do you want to continue(y/n):n |

1. Conclusion:

From above experiment I have concluded that:

\*(Please Write Conclusion what you have learn from this experiment)

1. Review Questions & Exercises:
   1. Fill in the Blanks
      1. The **write()** member function writes “raw” binary data to a file.
      2. In **sequential** file access, the contents of the file are read in the order they appear in the file, from the file’s start to its end.
      3. The **tellg** member function returns a file’s current read position.
      4. A negative offset causes the file’s read or write position to be moved **backwards**

in the file from the position specified by the mode.

* 1. Answer the following:
     1. What is the difference between the seekg and seekp member functions?

# seekg moves the file input pointer(position of reading frm file) while seekp moves file output pointer( position f writing to file).

* + 1. If a program has read to the end of a file, what must you do before using either the seekg or seekp member functions?

# Using ios\_base::end statement inside seekg or seekp

* + 1. How do you rewind a sequential-access file?

# Using the rewind() method that sets the file position indicator to the beginning of the file

* 1. Algorithm Workbench
     1. Write code that determines the number of bytes contained in the file associated with the file stream object dataFile.

# dataFile.seekg(0L, ios::end); cout << dataFile.tellg() << endl;

* + 1. Write code that sorts all Employee records of the files from employee id field. struct Employee

# {

**std::string firstName; std::string lastName; std::string id;**

**};**

**std::istream& operator>>(std::istream& is, Employee& s)**

**{**

**return is >> s.firstName >> s.lastName >> s.id;**

**}**

**//reading int main()**

**{**

**std::ifstream input("Text\_File.txt"); std::vector<Emlpoyee> employee; Employee s;**

**while (input >> s)**

**{**

**employee.push\_back(s);**

**}**

**//sorting std::sort(employee.begin(),**

**employee.end(),**

**[](const Employee& s1, const Employee& s2) { return s1.id < s2.id; });**

**}**

* 1. Programming Assignments:
     1. File encryption is the science of writing the contents of a file in a secret code. Your encryption program should work like a filter, reading the contents of one file, modifying the data into a code, and then writing the coded contents out to a second file. The second file will be a version of the first file, but written in a secret code.

# // File Encryption Filter #include <iostream> #include <fstream> using namespace std;

**// Constant for array size const int SIZE = 81;**

**int main()**

**{**

**ifstream inFile; // Input file object ofstream outFile; // Output file object char inName[SIZE]; // Input file name char outName[SIZE]; // Output file name**

**char input; // To hold a character of input**

**// Get the input file name.**

**cout << "Enter the input file name: "; cin.getline(inName, SIZE);**

**// Open the input file. inFile.open(inName);**

**// Test for errors. if (!inFile)**

**{**

**// There was an error so display an error**

**// message and end the program.**

**cout << "Error opening " << inFile << endl; exit(EXIT\_FAILURE);**

**}**

**// Get the output file name.**

**cout << "Enter the output file name: ";**

**cin.getline(outName, SIZE);**

**// Open the output file. outFile.open(outName);**

**// Test for errors. if (!outFile)**

**{**

**// There was an error so display an error**

**// message and end the program.**

**cout << "Error opening " << outFile << endl; exit(EXIT\_FAILURE);**

**}**

**// Read the contents of the input file, one**

**// character at a time, encrypt it, and store**

**// it in the output file. while (!inFile.eof())**

**{**

**// Get a character from inFile. inFile.get(input);**

**// Encrypt the character by adding 10 to it. input += 10;**

**// Write the character to outFile. outFile.put(input);**

**}**

**cout << "The file has been encrypted.\n";**

**// Close both files. inFile.close(); outFile.close();**

**return 0;**

**}**

* + 1. Write a program that decrypts the file produced by the program in Programming Assignment i. The decryption program should read the contents of the coded file, restore the data to its original state, and write it to another file.

**// File Dencryption Filter**

**#include <iostream> #include <fstream> using namespace std; const int SIZE = 81;**

**int main()**

**{**

**ifstream inFile; // Input file object ofstream outFile; // Output file object char inName[SIZE]; // Input file name char outName[SIZE]; // Output file name**

**char input; // To hold a character of input**

**// Get the input file name.**

**cout << "Enter the input file name: "; cin.getline(inName, SIZE);**

**// Open the input file. inFile.open(inName);**

**// Test for errors. if (!inFile)**

**{**

**// There was an error so display an error**

**// message and end the program.**

**cout << "Error opening " << inFile << endl; exit(EXIT\_FAILURE);**

**}**

**// Get the output file name.**

**cout << "Enter the output file name: "; cin.getline(outName, SIZE);**

**// Open the output file. outFile.open(outName);**

**// Test for errors. if (!outFile)**

**{**

**// There was an error so display an error**

**// message and end the program.**

**cout << "Error opening " << outFile << endl; exit(EXIT\_FAILURE);**

**}**

**// Read the contents of the input file, one**

**// character at a time, decrypt it, and store**

**// it in the output file. while (!inFile.eof())**

**{**

**// Get a character from inFile. inFile.get(input);**

**// Dencrypt the character by subtracting**

**// 10 from it. input -= 10;**

**// Write the character to outFile. outFile.put(input);**

**}**

**cout << "The file has been decrypted.\n";**

**// Close both files. inFile.close(); outFile.close();**

**return 0;**

**}**

* + 1. Write a program that updates a binary file for customers in a bank. The data consist of a customer identity (an integer) and customer balance (a double value). Write two application files. In the first application program, create the binary file with identity (starting with 1000) and balances for five customers. In the second application program, let the balance be changed (using deposit or withdrawal) for some customers (update the file) and then print the contents of the file after updating.

# #include<iostream> #include<fstream> #include<cctype> #include<iomanip> using namespace std;

**class account**

**{**

**int acno;**

**char name[50]; double deposit; char type;**

**public:**

**void create\_account(); //function to get data from user void show\_account() const; //function to show data on screen**

**void dep(int); //function to accept amount and add to balance amount**

**void draw(int); //function to accept amount and subtract from balance amount**

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

**int retdeposit() const; //function to return balance amount char rettype() const; //function to return type of account void report() const;**

**}; //class ends here**

**void account::create\_account()**

**{**

**cout << "\nEnter The account No. : "; cin >> acno;**

**cout << "\n\nEnter The Name of The account Holder : "; cin.ignore();**

**cin.getline(name, 50);**

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

**type = toupper(type);**

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

**cin >> deposit;**

**cout << "\n\n\nAccount Created..";**

**}**

**void account::show\_account() const**

**{**

**cout << "\nAccount No. : " << acno; cout << "\nAccount Holder Name : "; cout << name;**

**cout << "\nType of Account : " << type; cout << "\nBalance amount : " << deposit;**

**}**

**void account::dep(int x)**

**{**

**deposit += x;**

**}**

**void account::draw(int x)**

**{**

**deposit -= x;**

**}**

**void account::report() const**

**{**

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

**}**

**int account::retacno() const**

**{**

**return acno;**

**}**

**int account::retdeposit() const**

**{**

**return deposit;**

**}**

**char account::rettype() const**

**{**

**return type;**

**}**

**void write\_account(); //function to write record in binary file void display\_all(); //function to display all account details**

**void deposit\_withdraw(int, int); // function to desposit/withdraw amount for given account**

**int main()**

**{**

**char ch; int num; do**

**{**

**system("cls");**

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

**cout << "\n\n\t01. NEW ACCOUNT"; cout << "\n\n\t02. DEPOSIT AMOUNT";**

**cout << "\n\n\t03. WITHDRAW AMOUNT";**

**cout << "\n\n\t04. ALL ACCOUNT HOLDER LIST";**

**cout << "\n\n\t05. EXIT";**

**cout << "\n\n\tSelect Your Option "; cin >> ch;**

**system("cls"); switch (ch)**

**{**

**case '1':**

**write\_account(); break;**

**case '2':**

**cout << "\n\n\tEnter The account No. : "; cin >> num; deposit\_withdraw(num, 1);**

**break; case '3':**

**cout << "\n\n\tEnter The account No. : "; cin >> num; deposit\_withdraw(num, 2);**

**break; case '4':**

**display\_all(); break;**

**case '5':**

**cout << "\n\n\tThanks for using bank managemnt system"; break;**

**default:cout << "\a";**

**}**

**cin.ignore();**

**cin.get();**

**} while (ch != '8'); return 0;**

**}**

**void write\_account()**

**{**

**account ac; ofstream outFile;**

**outFile.open("account.dat", ios::binary | ios::app); ac.create\_account();**

**outFile.write(reinterpret\_cast<char \*> (&ac), sizeof(account)); outFile.close();**

**}**

**void display\_all()**

**{**

**account ac; ifstream inFile;**

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

**{**

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

**}**

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

**cout <<**

**"====================================================\n";**

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

**cout <<**

**"====================================================\n";**

**while (inFile.read(reinterpret\_cast<char \*> (&ac), sizeof(account)))**

**{**

**ac.report();**

**}**

**inFile.close();**

**}**

**void deposit\_withdraw(int n, int option)**

**{**

**int amt;**

**bool found = false; account ac; fstream File;**

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

**{**

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

**}**

**while (!File.eof() && found == false)**

**{**

**File.read(reinterpret\_cast<char \*> (&ac), sizeof(account)); if (ac.retacno() == n)**

**{**

**ac.show\_account();**

**if (option == 1)**

**{**

**cout << "\n\n\tTO DEPOSITE AMOUNT ";**

**cout << "\n\nEnter The amount to be deposited"; cin >> amt;**

**ac.dep(amt);**

**}**

**if (option == 2)**

**{**

**cout << "\n\n\tTO WITHDRAW AMOUNT "; cout << "\n\nEnter The amount to be withdraw"; cin >> amt;**

**int bal = ac.retdeposit() - amt;**

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

**else**

**}**

**cout << "Insufficience balance";**

**ac.draw(amt);**

**int pos = (-1)\*static\_cast<int>(sizeof(ac)); File.seekp(pos, ios::cur);**

**File.write(reinterpret\_cast<char \*> (&ac), sizeof(account)); cout << "\n\n\t Record Updated";**

**found = true;**

**}**

**}**

**}**

1. References:

# File.close();

**if (found == false)**

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

* 1. E Balagurusamy Object-Oriented Programming with C++.7th edition. McGraw-

Hill Publication, ISBN 10: 9352607996 ISBN 13: 9789352607990

* 1. Tony Gaddis- “STARTING OUT WITH C++ From Control Structures through Objects”, Pearson Education, ISBN 13: 978-0-13-376939-5