

حقوق النشر محفوظة، أسعار الكورسات في المنصة هي أسعار رمزيه جدا، ارجو عدم نشر هذه الوثيقة لان نشرها سيمنعنا من الاستمرار في تقديم العلم للآخرين

ارجو عدم استخدام هذه الوثيقة من غير وجه حق لأنك ستحرم الاف الناس من التعلم

ProgrammingAdvices.com



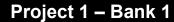


```
#include <iostream>
#include <fstream>
#include <string>
#include <vector>
#include <iomanip>
using namespace std;
const string ClientsFileName = "Clients.txt";
void ShowMainMenue();
struct sClient
    string AccountNumber;
    string PinCode;
    string Name;
    string Phone;
    double AccountBalance;
    bool MarkForDelete = false;
};
vector<string> SplitString(string S1, string Delim)
    vector<string> vString;
    short pos = 0;
    string sWord; // define a string variable
    // use find() function to get the position of the delimiters
    while ((pos = S1.find(Delim)) != std::string::npos)
        sWord = S1.substr(0, pos); // store the word
        if (sWord != "")
            vString.push_back(sWord);
        }
        S1.erase(0, pos + Delim.length()); /* erase() until
positon and move to next word. */
    }
    if (S1 != "")
        vString.push_back(S1); // it adds last word of the string.
    }
    return vString;
}
```

Project 1 - Bank 1



```
sClient ConvertLinetoRecord(string Line, string Seperator =
"#//#")
{
    sClient Client;
    vector<string> vClientData;
    vClientData = SplitString(Line, Seperator);
    Client.AccountNumber = vClientData[0];
    Client.PinCode = vClientData[1];
    Client.Name = vClientData[2];
    Client.Phone = vClientData[3];
    Client.AccountBalance = stod(vClientData[4]);//cast string to
double
    return Client;
}
string ConvertRecordToLine(sClient Client, string Seperator =
"#//#")
{
    string stClientRecord = "";
    stClientRecord += Client.AccountNumber + Seperator;
    stClientRecord += Client.PinCode + Seperator;
    stClientRecord += Client.Name + Seperator;
    stClientRecord += Client.Phone + Seperator;
    stClientRecord += to_string(Client.AccountBalance);
    return stClientRecord;
}
```

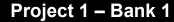




```
bool ClientExistsByAccountNumber(string AccountNumber, string
FileName)
{
    vector <sClient> vClients;
    fstream MyFile;
    MyFile.open(FileName, ios::in);//read Mode
    if (MyFile.is_open())
    {
        string Line;
        sClient Client;
        while (getline(MyFile, Line))
            Client = ConvertLinetoRecord(Line);
            if (Client.AccountNumber == AccountNumber)
                MyFile.close();
                return true;
            vClients.push_back(Client);
        }
        MyFile.close();
    }
    return false;
}
```



```
sClient ReadNewClient()
    sClient Client;
    cout << "Enter Account Number? ";</pre>
    // Usage of std::ws will extract allthe whitespace character
    getline(cin >> ws, Client.AccountNumber);
    while (ClientExistsByAccountNumber(Client.AccountNumber,
ClientsFileName))
    {
        cout << "\nClient with [" << Client.AccountNumber << "]</pre>
already exists, Enter another Account Number? ";
        getline(cin >> ws, Client.AccountNumber);
    }
    cout << "Enter PinCode? ";</pre>
    getline(cin, Client.PinCode);
    cout << "Enter Name? ";</pre>
    getline(cin, Client.Name);
    cout << "Enter Phone? ";</pre>
    getline(cin, Client.Phone);
    cout << "Enter AccountBalance? ";</pre>
    cin >> Client.AccountBalance;
    return Client;
}
```





```
vector <sClient> LoadCleintsDataFromFile(string FileName)
   vector <sClient> vClients;
   fstream MyFile;
   MyFile.open(FileName, ios::in);//read Mode
   if (MyFile.is_open())
       string Line;
       sClient Client;
       while (getline(MyFile, Line))
       {
          Client = ConvertLinetoRecord(Line);
          vClients.push_back(Client);
       MyFile.close();
   }
   return vClients;
}
void PrintClientRecordLine(sClient Client)
   cout << " " << setw(15) << left << Client.AccountNumber;</pre>
   cout << " " << setw(10) << left << Client.PinCode;</pre>
   cout << "      " << setw(12) << left << Client.Phone;</pre>
   }
```



```
void ShowAllClientsScreen()
   vector <sClient> vClients =
LoadCleintsDataFromFile(ClientsFileName);
   cout << "\n\t\t\t\tClient List (" << vClients.size() << ")</pre>
Client(s).";
   cout <<
"\n_____
           cout << "_____\n" << endl;
   cout << "| " << left << setw(15) << "Accout Number";</pre>
   cout << "| " << left << setw(10) << "Pin Code";</pre>
   cout << "| " << left << setw(40) << "Client Name";</pre>
   cout << "| " << left << setw(12) << "Phone";</pre>
   cout << "| " << left << setw(12) << "Balance";</pre>
   cout <<
"\n_____
           cout << "_____\n" << endl;
   if (vClients.size() == 0)
      cout << "\t\t\t\tNo Clients Available In the System!";</pre>
   else
      for (sClient Client : vClients)
         PrintClientRecordLine(Client);
         cout << endl;</pre>
      }
   cout <<
           _____
   cout << "_____\n" << endl;
}
```



```
void PrintClientCard(sClient Client)
    cout << "\nThe following are the client details:\n";</pre>
    cout << "----":
    cout << "\nAccout Number: " << Client.AccountNumber;</pre>
    cout << "\nPin Code : " << Client.PinCode;
cout << "\nName : " << Client.Name;
cout << "\nPhone : " << Client.Phone;</pre>
    cout << "\nAccount Balance: " << Client.AccountBalance;</pre>
    cout << "\n----\n";
}
bool FindClientByAccountNumber(string AccountNumber, vector
<sClient> vClients, sClient& Client)
    for (sClient C : vClients)
        if (C.AccountNumber == AccountNumber)
             Client = C;
             return true;
        }
    }
    return false:
}
sClient ChangeClientRecord(string AccountNumber)
    sClient Client;
    Client.AccountNumber = AccountNumber;
    cout << "\n\nEnter PinCode? ";</pre>
    getline(cin >> ws, Client.PinCode);
    cout << "Enter Name? ";</pre>
    getline(cin, Client.Name);
    cout << "Enter Phone? ";</pre>
    getline(cin, Client.Phone);
    cout << "Enter AccountBalance? ";</pre>
    cin >> Client.AccountBalance;
    return Client;
}
```



```
bool MarkClientForDeleteByAccountNumber(string AccountNumber,
vector <sClient>& vClients)
{
    for (sClient& C : vClients)
        if (C.AccountNumber == AccountNumber)
            C.MarkForDelete = true;
            return true;
        }
    }
    return false;
}
vector <sClient> SaveCleintsDataToFile(string FileName, vector
<sClient> vClients)
{
    fstream MyFile;
    MyFile.open(FileName, ios::out);//overwrite
    string DataLine;
    if (MyFile.is_open())
        for (sClient C : vClients)
            if (C.MarkForDelete == false)
                //we only write records that are not marked for
delete.
                DataLine = ConvertRecordToLine(C);
                MyFile << DataLine << endl;
            }
        }
        MyFile.close();
    }
    return vClients;
}
```



```
void AddDataLineToFile(string FileName, string stDataLine)
    fstream MyFile;
    MyFile.open(FileName, ios::out | ios::app);
    if (MyFile.is_open())
        MyFile << stDataLine << endl;
        MyFile.close();
    }
}
void AddNewClient()
    sClient Client;
    Client = ReadNewClient();
    AddDataLineToFile(ClientsFileName,
ConvertRecordToLine(Client));
void AddNewClients()
    char AddMore = 'Y';
    do
        //system("cls");
        cout << "Adding New Client:\n\n";</pre>
        AddNewClient();
        cout << "\nClient Added Successfully, do you want to add</pre>
more clients? Y/N? ";
        cin >> AddMore;
    } while (toupper(AddMore) == 'Y');
}
```



```
bool DeleteClientByAccountNumber(string AccountNumber, vector
<sClient>& vClients)
{
    sClient Client;
    char Answer = 'n';
    if (FindClientByAccountNumber(AccountNumber, vClients,
Client))
    {
        PrintClientCard(Client);
        cout << "\n\nAre you sure you want delete this client? y/n</pre>
? ";
        cin >> Answer;
        if (Answer == 'y' || Answer == 'Y')
            MarkClientForDeleteByAccountNumber(AccountNumber,
vClients);
            SaveCleintsDataToFile(ClientsFileName, vClients);
            //Refresh Clients
            vClients = LoadCleintsDataFromFile(ClientsFileName);
            cout << "\n\nClient Deleted Successfully.";</pre>
            return true:
        }
    }
    else
        cout << "\nClient with Account Number (" << AccountNumber</pre>
<< ") is Not Found!";</pre>
        return false:
    }
}
bool UpdateClientByAccountNumber(string AccountNumber, vector
<sClient>& vClients)
{
    sClient Client;
    char Answer = 'n';
    if (FindClientByAccountNumber(AccountNumber, vClients,
Client))
    {
```



```
PrintClientCard(Client);
        cout << "\n\nAre you sure you want update this client? y/n</pre>
? ";
        cin >> Answer;
        if (Answer == 'y' || Answer == 'Y')
            for (sClient& C : vClients)
                 if (C.AccountNumber == AccountNumber)
                     C = ChangeClientRecord(AccountNumber);
                     break;
                 }
             }
             SaveCleintsDataToFile(ClientsFileName, vClients);
            cout << "\n\nClient Updated Successfully.";</pre>
            return true;
        }
    }
    else
        cout << "\nClient with Account Number (" << AccountNumber</pre>
<< ") is Not Found!";</pre>
        return false;
    }
}
string ReadClientAccountNumber()
    string AccountNumber = "";
    cout << "\nPlease enter AccountNumber? ";</pre>
    cin >> AccountNumber;
    return AccountNumber;
}
```



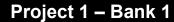
```
void ShowDeleteClientScreen()
   cout << "\n----\n";
   cout << "\tDelete Client Screen";
cout << "\n-----\n";</pre>
   vector <sClient> vClients =
LoadCleintsDataFromFile(ClientsFileName);
   string AccountNumber = ReadClientAccountNumber();
   DeleteClientByAccountNumber(AccountNumber, vClients);
}
void ShowUpdateClientScreen()
   cout << "\n----\n";
   cout << "\tUpdate Client Info Screen";
cout << "\n----\n";</pre>
   vector <sClient> vClients =
LoadCleintsDataFromFile(ClientsFileName);
   string AccountNumber = ReadClientAccountNumber();
   UpdateClientByAccountNumber(AccountNumber, vClients);
}
void ShowAddNewClientsScreen()
{
   cout << "\n----\n";
   cout << "\tAdd New Clients Screen";</pre>
   cout << "\n----\n":
   AddNewClients();
}
```



```
void ShowFindClientScreen()
   cout << "\n----\n";
   cout << "\tFind Client Screen";</pre>
   cout << "\n----\n":
   vector <sClient> vClients =
LoadCleintsDataFromFile(ClientsFileName);
   sClient Client;
   string AccountNumber = ReadClientAccountNumber();
   if (FindClientByAccountNumber(AccountNumber, vClients,
Client))
       PrintClientCard(Client);
   else
       cout << "\nClient with Account Number[" << AccountNumber</pre>
<< "] is not found!";</pre>
}
void ShowEndScreen()
   cout << "\n----\n";
   cout << "\tProgram Ends :-)";</pre>
                              ,
-----\n":
   cout << "\n-----
}
enum enMainMenueOptions
   eListClients = 1, eAddNewClient = 2,
   eDeleteClient = 3, eUpdateClient = 4,
   eFindClient = 5, eExit = 6
};
void GoBackToMainMenue()
{
   cout << "\n\nPress any key to go back to Main Menue...";</pre>
   system("pause>0");
   ShowMainMenue();
}
short ReadMainMenueOption()
   cout << "Choose what do you want to do? [1 to 6]? ";</pre>
   short Choice = 0;
   cin >> Choice;
   return Choice;
}
```



```
void PerfromMainMenueOption(enMainMenueOptions MainMenueOption)
    switch (MainMenueOption)
    case enMainMenueOptions::eListClients:
        system("cls");
        ShowAllClientsScreen();
        GoBackToMainMenue();
        break;
    }
    case enMainMenueOptions::eAddNewClient:
        system("cls");
        ShowAddNewClientsScreen();
        GoBackToMainMenue();
        break;
    case enMainMenueOptions::eDeleteClient:
        system("cls");
        ShowDeleteClientScreen();
        GoBackToMainMenue();
        break;
    case enMainMenueOptions::eUpdateClient:
        system("cls");
        ShowUpdateClientScreen();
        GoBackToMainMenue();
        break:
    case enMainMenueOptions::eFindClient:
        system("cls");
        ShowFindClientScreen();
        GoBackToMainMenue();
        break;
    case enMainMenueOptions::eExit:
        system("cls");
        ShowEndScreen();
        break;
    }
}
```





```
void ShowMainMenue()
   system("cls");
   cout << "========\n":
   cout << "\t\tMain Menue Screen\n";</pre>
   cout << "=======\n":
   cout << "\t[1] Show Client List.\n";</pre>
   cout << "\t[2] Add New Client.\n";</pre>
   cout << "\t[3] Delete Client.\n";</pre>
   cout << "\t[4] Update Client Info.\n";</pre>
   cout << "\t[5] Find Client.\n";</pre>
   cout << "\t[6] Exit.\n";</pre>
   cout << "=======\n";
PerfromMainMenueOption((enMainMenueOptions)ReadMainMenueOption());
}
int main()
{
   ShowMainMenue();
   system("pause>0");
   return 0;
}
```