

Algorithms Level 3



26+ Years
of Experience

PROGRAMMING ADVICES

LEARN THE
RIGHT WAY

Mohammed Abu-Hadhoud

MBA, PMOC, PgMP®, PMP®, PMI-RMP®, CM, ITILF, MCPD, MCSD



حقوق النشر محفوظة، أسعار الكورسات في المنصة هي أسعار
رمزية 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



Project 1 – Bank 1

```
#include <iostream>
#include <fstream>
#include <string>
#include <vector>
#include <iomanip>

using namespace std;
const string ClientsFileName = "Clients.txt";

void ShowMainMenu();

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
        position 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? ";

    // Usage of std::ws will extract all the whitespace character
    getline(cin >> ws, Client.AccountNumber);

    while (ClientExistsByAccountNumber(Client.AccountNumber,
    ClientsFileName))
    {
        cout << "\nClient with [" << Client.AccountNumber << "]
already exists, Enter another Account Number? ";
        getline(cin >> ws, Client.AccountNumber);
    }

    cout << "Enter PinCode? ";
    getline(cin, Client.PinCode);

    cout << "Enter Name? ";
    getline(cin, Client.Name);

    cout << "Enter Phone? ";
    getline(cin, Client.Phone);

    cout << "Enter AccountBalance? ";
    cin >> Client.AccountBalance;

    return Client;
}
```



Project 1 – Bank 1

```
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;
    cout << "|" << setw(10) << left << Client.PinCode;
    cout << "|" << setw(40) << left << Client.Name;
    cout << "|" << setw(12) << left << Client.Phone;
    cout << "|" << setw(12) << left << Client.AccountBalance;
}
```



Project 1 – Bank 1

```
void ShowAllClientsScreen()
{
    vector <sClient> vClients =
LoadCleintsDataFromFile(ClientsFileName);

    cout << "\n\t\t\t\t\tClient List (" << vClients.size() << ")
Client(s).";
    cout <<
"\n_";
    cout << " _\n" << endl;

    cout << "| " << left << setw(15) << "Accout Number";
    cout << "| " << left << setw(10) << "Pin Code";
    cout << "| " << left << setw(40) << "Client Name";
    cout << "| " << left << setw(12) << "Phone";
    cout << "| " << left << setw(12) << "Balance";
    cout <<
"\n_";
    cout << " _\n" << endl;

    if (vClients.size() == 0)
        cout << "\t\t\t\t\tNo Clients Available In the System!";
    else

        for (sClient Client : vClients)
        {

            PrintClientRecordLine(Client);
            cout << endl;

        }

    cout <<
"\n_";
    cout << " _\n" << endl;
}
```



```
void PrintClientCard(sClient Client)
{
    cout << "\nThe following are the client details:\n";
    cout << "-----\n";
    cout << "\nAccount Number: " << Client.AccountNumber;
    cout << "\nPin Code      : " << Client.PinCode;
    cout << "\nName          : " << Client.Name;
    cout << "\nPhone         : " << Client.Phone;
    cout << "\nAccount Balance: " << Client.AccountBalance;
    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 << "\nEnter PinCode? ";
    getline(cin >> ws, Client.PinCode);

    cout << "Enter Name? ";
    getline(cin, Client.Name);

    cout << "Enter Phone? ";
    getline(cin, Client.Phone);

    cout << "Enter AccountBalance? ";
    cin >> Client.AccountBalance;
    return Client;
}
```




Project 1 – Bank 1

```
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;
}
```



Project 1 – Bank 1

```
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";

        AddNewClient();
        cout << "\nClient Added Successfully, do you want to add
more clients? Y/N? ";
        cin >> AddMore;

    } while (toupper(AddMore) == 'Y');
}
```



Project 1 – Bank 1

```
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
? ";
        cin >> Answer;
        if (Answer == 'y' || Answer == 'Y')
        {
            MarkClientForDeleteByAccountNumber(AccountNumber, vClients);
            SaveCleintsDataToFile(ClientsFileName, vClients);

            //Refresh Clients
            vClients = LoadCleintsDataFromFile(ClientsFileName);

            cout << "\n\nClient Deleted Successfully.";
            return true;
        }
    }
    else
    {
        cout << "\nClient with Account Number (" << AccountNumber
<< ") is Not Found!";
        return false;
    }
}

bool UpdateClientByAccountNumber(string AccountNumber, vector<sClient>& vClients)
{
    sClient Client;
    char Answer = 'n';

    if (FindClientByAccountNumber(AccountNumber, vClients, Client))
    {
```



Project 1 – Bank 1

```
PrintClientCard(Client);
cout << "\n\nAre you sure you want update this client? y/n
? ";
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.";
    return true;
}

}
else
{
    cout << "\nClient with Account Number (" << AccountNumber
<< ") is Not Found!";
    return false;
}
}

string ReadClientAccountNumber()
{
    string AccountNumber = "";

    cout << "\nPlease enter AccountNumber? ";
    cin >> AccountNumber;
    return AccountNumber;
}
```



```
void ShowDeleteClientScreen()
{
    cout << "\n-----\n";
    cout << "\tDelete Client Screen";
    cout << "\n-----\n";

    vector <sClient> vClients =
LoadCleintsDataFromFile(ClientsFileName);
    string AccountNumber = ReadClientAccountNumber();
    DeleteClientByAccountNumber(AccountNumber, vClients);
}

void ShowUpdateClientScreen()
{
    cout << "\n-----\n";
    cout << "\tUpdate Client Info Screen";
    cout << "\n-----\n";

    vector <sClient> vClients =
LoadCleintsDataFromFile(ClientsFileName);
    string AccountNumber = ReadClientAccountNumber();
    UpdateClientByAccountNumber(AccountNumber, vClients);
}

void ShowAddNewClientsScreen()
{
    cout << "\n-----\n";
    cout << "\tAdd New Clients Screen";
    cout << "\n-----\n";

    AddNewClients();
}
```



```
void ShowFindClientScreen()
{
    cout << "\n-----\n";
    cout << "\tFind Client Screen";
    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
<< "] is not found!";
}

void ShowEndScreen()
{
    cout << "\n-----\n";
    cout << "\tProgram Ends :-)\n";
    cout << "\n-----\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...";
    system("pause>0");
    ShowMainMenue();
}

short ReadMainMenueOption()
{
    cout << "Choose what do you want to do? [1 to 6]? ";
    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 ShowMainMenu()
{
    system("cls");
    cout << "=====\n";
    cout << "\t\tMain Menu Screen\n";
    cout << "=====\n";
    cout << "\t[1] Show Client List.\n";
    cout << "\t[2] Add New Client.\n";
    cout << "\t[3] Delete Client.\n";
    cout << "\t[4] Update Client Info.\n";
    cout << "\t[5] Find Client.\n";
    cout << "\t[6] Exit.\n";
    cout << "=====\n";

    PerfromMainMenuOption((enMainMenuOptions)ReadMainMenuOption());
}

int main()
{
    ShowMainMenu();
    system("pause>0");
    return 0;
}
```