```
//Allina Khan
//Chapter 7 Bank Accounts
#include <iostream>
#include <fstream>
using namespace std;

//function prototypes
int read_accts(int [], double [], int);
void print_accts(int [], double [], int, ofstream &);
//prints read in acct info
void menu();
void balanceFunct(int [], double [], int, ofstream &, ifstream & );
int findAcct(int [], int, int );
void deposit(int[], double[], int, ofstream &, ifstream &);
void withdrawal(int[], double[], int, ofstream &, ifstream &);
int newAcct(int[], double[], int, ofstream &, ifstream &);
int deleteAcc(int[], double[], int, ofstream &, ifstream &);

int main() {

    //redirecting cin and cout to a file
    ofstream outfile("allinachap7out.txt");
    ifstream infile("allinachap7in2.txt");
    //ofstream outfile("con");
    //ifstream infile("con");

    const int max_num=500;        //number of accounts bank can handle
    int acctNum[max_num];
    double balance[max_num];
    int num_accts;          // actual number of accounts read in
    bool quit=true;
    char select;

    outfile << "Allina Khan " << endl << "Project 7 Bank Account"
            << endl << endl;

    //reading in acctNum and balance arrays and putting total accts
    //in integer num_accts
    num_accts = read_accts(acctNum,balance,max_num );

    //Printing all read in account information
    outfile << "Initial account data: " << endl;
    print_accts(acctNum, balance, num_accts, outfile);

    //repeats input until user wants to quit.
    do {
        //Prints menu to console
        menu();

        //Prompts user for menu selection
        cout << "Make a selection from above options: " << endl;
        infile >> select;

        //Checks if user input is valid and calls correct operation
        switch (select) {
            case 'W':
            case 'w':
                withdrawal(acctNum, balance, num_accts,outfile,infile);
                break;
            case 'D':
            case 'd':
                deposit(acctNum, balance, num_accts, outfile, infile);
                break;
            case 'N':
            case 'n':
                num_accts = newAcct(acctNum, balance, num_accts,
```

```
                                    outfile, infile);
                    break;
                case 'B':
                case 'b':
                    balanceFunct(acctNum, balance, num_accts, outfile,
                                 infile);
                    break;
                case 'Q':
                case 'q':
                    quit=false;
                    outfile << "You have quit the program." << endl << endl;
                    break;
                case 'X':
                case 'x':
                    num_accts = deleteAcc(acctNum, balance, num_accts,
                                          outfile, infile);
                    break;
                default:
                    outfile << "Incorrect selection: " << select
                            <<  ", Choose from functions provided."
                            << endl << endl;
            }
        } while (quit);

        outfile << "New Account Data: " << endl;
        print_accts(acctNum, balance, num_accts, outfile);

    return (0); }


    /* function read_accts()
    Input:
        acctNum[], balance[], max_num
    Process:
        Calls in-file allinachap7in1.txt
        Declares variable p and adds 1 as each account is read in
        while (infile.eof() ) remains true, reads in acct num & balance
        checks if number of total accounts recorded are valid
    Output:
        The filled balance and acctNum arrays
        returns value of p to calling function
    */

    int read_accts(int acctNum[], double balance[], int max_accts) {
        //preparing for separate infiles
        ifstream infile("allchap7in1.txt");

        int p = 0;        //count numbers of accounts

        //evaluates infile true false to read in a variable number of data.
        while(!infile.eof() )
        {
            if (p >= 0 && p <= max_accts)
            {
                cout << "Enter account number";
                infile >> acctNum[p];
                //cout<<"ACCOUNT NUMBER AT: "<<p<<" is "<<acctNum[p]<<endl;
                cout << "Enter balance of account: ";
                infile >> balance[p];
                //cout<<"BLALNCE NUMBER AT: "<<p<<" is "<<balance[p]<<endl;
                p++;
                //cout << "INDEX AT: " << p << endl;
            }
        }

    return (p-1); }
```

```
/* function print_accts
Input: acct_num, balance, num_accts, ofstream
Process:
    Prints title of table
    Prints headings of table
    Prints account and then balance #
Output: print account and balance data in outfile
*/

void print_accts(int acctNum_array[], double balance_array[],
                 int num_accts, ofstream &out1) {

    //Table header and titles
         out1 << "Accounts\tBalances" << endl;

    //Table data
    for (int x=0; x <= num_accts; x++)
    {
       out1 << acctNum_array[x] << "\t\t$" << balance_array[x]
            << endl;
    }

    out1 << endl;
return; }

/* function menu
Input: None
Process: displays the menu
Output: the menu to console
*/

void menu() {
    //not sending to outfile since it's a prompt
    cout << "W - Withdrawal" << endl <<
         "D - Deposit" << endl <<
         "N - New account" << endl <<
         "B - Balance" << endl <<
         "Q - Quit" << endl <<
         "X - Delete Account" << endl << endl;
return; }

/* function findAcct / find account number
 Input:
    acctNum, num_accts, account
Process:
    Initiate integer p (for index)
    Evaluates whether each value in acctNum array is equal to account
    If value is equal, p = the index of account and loop breaks
    If no value is equal, p = -1
Output:
    if acct exists: returns p = account number
    if acct does not: returns p = -1
*/

int findAcct(int acctNum_array[], int num_accts, int account) {
    int p;
    for (int x=0; x <= num_accts; x++) {
        if (account == acctNum_array[x]) {
            p=x;
            break;
        }
        else
            p=-1;
    }

return (p);
```

```
}

/* function balance
Input: acctNum array, balance array, num_accts, ifstream and ofstream
Process:
    Initiates integer acct and exist (index value of acct in array)
    prompts user for account number, and puts value in int acct
    calls function findAcct to evaluate if account exists
    prints error if it doesn't, otherwise
    Prints requested account balance
Output: sends acct number and acct balance to file
*/

void balanceFunct(int acctNum_array[], double balance_array[],
                  int num_accts, ofstream&out2, ifstream&in2) {
    int acct, exist;

    cout << endl << "Enter the account number: ";
    in2 >> acct;

    exist = findAcct(acctNum_array, num_accts, acct);

    if (exist==-1) {
        out2 <<"Action Balance could not be completed. "<<
             "Account does not exist. " << endl <<endl;
    }
    else {
        out2 << "Transaction Balance" << endl <<
            "Account Number entered: " << acct << endl <<
            "Balance: $" << balance_array[exist] << endl << endl;
    }
return; }

/*function deposit
Input: acctnum_array, balance_array, num_accts, ifstream, ofstream
Process:
    Initiates int acct,deposit, exist (store index value of acc)
    Asks for acct number,evaluates answer with call to findAcct function
    if exists, Asks user for amount to deposit, stores value in deposit
    adds deposit to balance_array[exist]
Output:
    Prints account number, amount deposited, old balance and new balance
*/

void deposit(int acctNum_array[], double balance_array[], int num_accts,
             ofstream&out3, ifstream &in3) {

    int acct, exist;
    double deposit;
    //deposit must be a double if you want decimals in balance array.
    cout << endl << "Enter account number: ";
    in3 >> acct;

    exist = findAcct(acctNum_array, num_accts, acct);

    if (exist==-1) {
        out3 << "Account does not exist. " <<
             "Action Deposit could not be completed." << endl << endl;
    }
    else {
        cout << "Enter amount to deposit: $";
        in3 >> deposit;

        out3 << "Transaction Deposit" << endl <<
            "Account Number: " << acct << endl <<
            "Amount to deposit: $" << deposit << endl <<
```

```
            "Old Balance: $" << balance_array[exist] << endl;
        //cout precision last for rest of program not the rest of funct.
        // but only for balance?
        out3.setf(ios::fixed,ios::floatfield);
        out3.precision(2);

        balance_array[exist] += deposit;

        out3 << "New Balance: $" << balance_array[exist] << endl<<endl;
    }

return;}

/* Function withdrawal
Input: acctnum_array, balance_array, num_accts, ofstream, ifstream
Process:
    Initiates int acct,withdraw, exist (store index value of acc)
    Asks for acct number,evaluates answer with call to findAcct function
    if exists, asks for amount to withdraw
    if sufficient funds, withdraws money. else, error message
Output:
    Prints account number, amount withdrawn, old balance and new balance
*/

void withdrawal(int acctNum_array[], double balance_array[],
                int num_accts, ofstream&out4, ifstream &in4) {

    int acct, exist;
    double withdraw;

    cout << endl << "Enter the account number: ";
    in4 >> acct;

    exist = findAcct(acctNum_array, num_accts, acct);

    if (exist==-1) {
            out4 << "Action withdrawal cannot be completed." <<
                 "Account does not exist. " << endl << endl;
    }
    else {
        cout << "Enter amount to withdraw: $";
        in4 >> withdraw;

        if ((balance_array[exist]-withdraw) > 0 &&
            (balance_array[exist]-withdraw) < balance_array[exist]) {
                out4 << "Transaction Withdrawal" << endl <<
                "Account Number: " << acct << endl <<
                "Amount to withdraw: $" << withdraw << endl <<
                "Old Balance: $" << balance_array[exist] << endl;

                balance_array[exist]-=withdraw;

                out4 << "New Balance: $" << balance_array[exist] << endl
                 << endl;
        }
        else
            out4 << "Insufficient funds, " <<
                 "could not perform Withdrawal transaction." << endl
                 << endl;
    }
return; }

/*Function newAcct
input: acctnum_array, balance_array, num_accts, ofstream, ifstream
Process:
    initiates int newacc, exist (holds index value)
```

```
    prompts user for new account number, stores in newacc.
    calls findAcct to see if it already exits
    If exists, prints error message. Otherwise,
    adds the account to acctNum array with an initial balance of 0.
Output:
    returns new number of accounts
*/

int newAcct(int acctNum_array[],double balance_array[],int num_accts,
            ofstream&out5, ifstream &in5) {

    int newacc, exist;

    cout << "Enter new account number: ";
    in5 >> newacc;

    exist=findAcct(acctNum_array, num_accts, newacc);

    if (exist==-1) {
        num_accts++;
        acctNum_array[num_accts]=newacc;
        balance_array[num_accts]=0;

        out5 << "Transaction New Account Completed" << endl
             << "New Account Number: " << newacc << endl
             << "New Account Balance: $" << balance_array[num_accts]
             << endl << endl;
    }
    else {
        out5 << "Action New Account cannot be completed. " <<
             "Account already exists."
             << endl << endl;
    }
return num_accts;}

/*Function deleteAcc
Input: acctnum_array, balance_array, num_accts, ofstream, ifstream
Process:
    initiates int deleteacc, exist, q (index value),
    p(# of accts to re-assign)
    Prompt to enter account for deletion then call function findAcct
    if exists, re-assign array actNum and balance values to delete
    requested acct number
Output: new number of accounts
*/

int deleteAcc(int acctNum_array[],double balance_array[],int num_accts,
            ofstream&out6, ifstream &in6) {

    int deleteacc, exist, q, p=num_accts;

    cout << "Enter account for deletion: ";
    in6 >> deleteacc;

    exist = findAcct(acctNum_array, num_accts, deleteacc);

    if (exist == -1) {
       out6 << "Action Delete Account cannot be completed, " <<
            "Account does not exist." << endl << endl;
    }
    else {
         p-=exist;
        for (int x=p; x > -1; x--)
        {
            //Must assign from bottom to top! Top to bottom assigns the
            //same value.
```

```
            q=num_accts-x;
            acctNum_array[q]=acctNum_array[q+1];
            balance_array[q]=balance_array[q+1];
        }

        out6 << "Transaction Delete Account Completed" << endl <<
             "Account to be deleted: " << deleteacc << endl <<
             "Old number of accounts: " << num_accts+1 << endl;

        num_accts-=1;

        out6 << "New number of accounts: " << num_accts+1 << endl
             << endl;
    }
return num_accts; }
```