```cpp
//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); }
```

```
133
134
135  /* function print_accts
136  Input: acct_num, balance, num_accts, ofstream
137  Process:
138      Prints title of table
139      Prints headings of table
140      Prints account and then balance #
141  Output: print account and balance data in outfile
142  */
143
144  void print_accts(int acctNum_array[], double balance_array[],
145                   int num_accts, ofstream &out1) {
146
147      //Table header and titles
148           out1 << "Accounts\tBalances" << endl;
149
150      //Table data
151      for (int x=0; x <= num_accts; x++)
152      {
153         out1 << acctNum_array[x] << "\t\t$" << balance_array[x]
154              << endl;
155      }
156
157      out1 << endl;
158  return; }
159
160  /* function menu
161  Input: None
162  Process: displays the menu
163  Output: the menu to console
164  */
165
166  void menu() {
167      //not sending to outfile since it's a prompt
168      cout << "W - Withdrawal" << endl <<
169           "D - Deposit" << endl <<
170           "N - New account" << endl <<
171           "B - Balance" << endl <<
172           "Q - Quit" << endl <<
173           "X - Delete Account" << endl << endl;
174  return; }
175
176  /* function findAcct / find account number
177   Input:
178      acctNum, num_accts, account
179  Process:
180      Initiate integer p (for index)
181      Evaluates whether each value in acctNum array is equal to account
182      If value is equal, p = the index of account and loop breaks
183      If no value is equal, p = -1
184  Output:
185      if acct exists: returns p = account number
186      if acct does not: returns p = -1
187  */
188
189  int findAcct(int acctNum_array[], int num_accts, int account) {
190      int p;
191      for (int x=0; x <= num_accts; x++) {
192          if (account == acctNum_array[x]) {
193              p=x;
194              break;
195          }
196          else
197              p=-1;
198      }
```

```
199
200 return (p);
201 }
202
203 /* function balance
204 Input: acctNum array, balance array, num_accts, ifstream and ofstream
205 Process:
206     Initiates integer acct and exist (index value of acct in array)
207     prompts user for account number, and puts value in int acct
208     calls function findAcct to evaluate if account exists
209     prints error if it doesn't, otherwise
210     Prints requested account balance
211 Output: sends acct number and acct balance to file
212 */
213
214 void balanceFunct(int acctNum_array[], double balance_array[],
215                   int num_accts, ofstream&out2, ifstream&in2) {
216     int acct, exist;
217
218     cout << endl << "Enter the account number: ";
219     in2 >> acct;
220
221     exist = findAcct(acctNum_array, num_accts, acct);
222
223     if (exist==-1) {
224         out2 <<"Action Balance could not be completed. "<<
225              "Account does not exist. " << endl <<endl;
226     }
227     else {
228         out2 << "Transaction Balance" << endl <<
229             "Account Number entered: " << acct << endl <<
230             "Balance: $" << balance_array[exist] << endl << endl;
231     }
232 return; }
233
234 /*function deposit
235 Input: acctnum_array, balance_array, num_accts, ifstream, ofstream
236 Process:
237     Initiates int acct,deposit, exist (store index value of acc)
238     Asks for acct number,evaluates answer with call to findAcct function
239     if exists, Asks user for amount to deposit, stores value in deposit
240     adds deposit to balance_array[exist]
241 Output:
242     Prints account number, amount deposited, old balance and new balance
243 */
244
245 void deposit(int acctNum_array[], double balance_array[], int num_accts,
246              ofstream&out3, ifstream &in3) {
247
248     int acct, deposit, exist;
249
250     cout << endl << "Enter account number: ";
251     in3 >> acct;
252
253     exist = findAcct(acctNum_array, num_accts, acct);
254
255     if (exist==-1) {
256         out3 << "Account does not exist. " <<
257              "Action Deposit could not be completed." << endl << endl;
258     }
259     else {
260         cout << "Enter amount to deposit: $";
261         in3 >> deposit;
262
263         out3 << "Transaction Deposit" << endl <<
264             "Account Number: " << acct << endl <<
```

```
265             "Amount to deposit: $" << deposit << endl <<
266             "Old Balance: $" << balance_array[exist] << endl;
267
268         balance_array[exist] += deposit;
269
270         out3 << "New Balance: $" << balance_array[exist] << endl<<endl;
271     }
272
273 return;}
274
275 /* Function withdrawal
276 Input: acctnum_array, balance_array, num_accts, ofstream, ifstream
277 Process:
278     Initiates int acct,withdraw, exist (store index value of acc)
279     Asks for acct number,evaluates answer with call to findAcct function
280     if exists, asks for amount to withdraw
281     if sufficient funds, withdraws money. else, error message
282 Output:
283     Prints account number, amount withdrawn, old balance and new balance
284 */
285
286 void withdrawal(int acctNum_array[], double balance_array[],
287                 int num_accts, ofstream&out4, ifstream &in4) {
288
289     int acct, exist, withdraw;
290
291     cout << endl << "Enter the account number: ";
292     in4 >> acct;
293
294     exist = findAcct(acctNum_array, num_accts, acct);
295
296     if (exist==-1) {
297             out4 << "Action withdrawal cannot be completed." <<
298                  "Account does not exist. " << endl << endl;
299     }
300     else {
301         cout << "Enter amount to withdraw: $";
302         in4 >> withdraw;
303
304         if ((balance_array[exist]-withdraw) > 0 &&
305             (balance_array[exist]-withdraw) < balance_array[exist]) {
306                 out4 << "Transaction Withdrawal" << endl <<
307                 "Account Number: " << acct << endl <<
308                 "Amount to withdraw: $" << withdraw << endl <<
309                 "Old Balance: $" << balance_array[exist] << endl;
310
311                 balance_array[exist]-=withdraw;
312
313                 out4 << "New Balance: $" << balance_array[exist] << endl
314                  << endl;
315         }
316         else
317             out4 << "Insufficient funds, " <<
318                  "could not perform Withdrawal transaction." << endl
319                  << endl;
320     }
321 return; }
322
323 /*Function newAcct
324 input: acctnum_array, balance_array, num_accts, ofstream, ifstream
325 Process:
326     initiates int newacc, exist (holds index value)
327     prompts user for new account number, stores in newacc.
328     calls findAcct to see if it already exits
329     If exists, prints error message. Otherwise,
330     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];
        }
```

```
397
398          out6 << "Transaction Delete Account Completed" << endl <<
399               "Account to be deleted: " << deleteacc << endl <<
400               "Old number of accounts: " << num_accts+1 << endl;
401
402          num_accts-=1;
403
404          out6 << "New number of accounts: " << num_accts+1 << endl
405               << endl;
406      }
407  return num_accts; }
408
```