

```
1 //Ben Scherer
2 // 8/3/2017
3 // Final Project
4 // Uses delimited text file as movie "Database". Allows search,update,delete,create functions.
5 // *** Leverages struct instead of paralelle arrays
6
7
8 //Headers to include
9 #include <iostream> //cout
10 #include <iomanip> // used to manipulate cout
11 #include <string> //needed for string variable
12 #include <math.h> //used for basic arithmetic
13 #include <limits> //user for numeric_limits
14 #include <vector> //needed to use vectors
15 #include <fstream> //file handling
16 #include <sstream> //used for string bufffer
17 #include <cstdlib>
18
19 using namespace std;
20
21 //data structure for each individual movie
22 struct Movie {
23     string title; //name of movie
24     string studio; //name of studio
25     string contentRating;
26     int year; //year of release
27     float rating; //10 star rating system
28     string genre; //genre of movie
29     string releaseDate; //date of release
30     string writers; //writers of movie
31     string runtime; //runtime
32     string directors; //directors
33     string actors; //actors
34
35
36
37 };
38
39 //Functions
40 void displayWelcome();
41 void movieDetails(vector<Movie> &movies, int movieIndex); //display details about movie
42 int getIntInput(string questionToAsk, string errorMsg); //validates int input and int range
43 int getIntInput(string questionToAsk, string errorMsg, int lowRange, int highRange); //validates int input and int range
44 string getInput(string questionToAsk, string errorMsg); //validates string input
45 Movie parseData(string movieString); //parses data from comma delimited string and returns Movie object
46 vector<Movie> loadDB(string strFileName); //loads data from flat file to vector
47 Movie addMovie(); //add movie to database
```

```

48 vector<int> searchTitle(vector<Movie> movies); //search for movies by title
49 vector<int> searchGenre(vector<Movie> movies);
50 vector<int> searchActor(vector<Movie> movies);
51 vector<int> searchYear(vector<Movie> movies);
52 vector<int> searchDirector(vector<Movie> movies);
53 void searchMovies(vector<Movie> &searchVector);
54 char getSentinel(string message,string errMsg); //Input validation for
    sentinel. simple y/n
55 Movie updateMovie(Movie mvObj); //update movie
56 void displayMovie(Movie mvObj); //display movie
57 float getFloatInput(string questionToAsk, string errorMsg, int lowRange, int
    highRange);
58 void updateDB(vector<Movie> movies,string file); //write changes to file
59
60
61
62 int main() {
63     const string dbFile = "movies.csv";
64     vector<Movie> movies; //contents of DB will be loaded into vector for runtime
65     //cout << "Welcome to the CIS111 offline movie database\n";
66     movies = loadDB(dbFile);
67
68     int menuChoice;
69     displayWelcome();
70     do {
71
72         menuChoice = getIntInput("", "Error: Enter valid number");
73         switch (menuChoice) {
74             case 1: searchMovies(movies); break;
75             case 2: movies.push_back(addMovie());
76
77         }
78         displayWelcome();
79     } while (menuChoice == 1 || menuChoice == 2);
80
81     cout << "Writing changes to database file\n";
82     updateDB(movies, dbFile);
83
84     system("pause");
85 }
86
87 //write changes to db file
88 void updateDB(vector<Movie> movies,string file) {
89     ofstream outFile(file);
90     for (Movie mvObj : movies) {
91         //
92         title,studio,content,year,rating,genre,releasedate,writers,runtime,directors,actors
93         outFile << mvObj.title << "|" << mvObj.studio << "|" <<
94             mvObj.contentRating << "|" << mvObj.year << "|" << mvObj.rating << "|" <<
95             mvObj.genre << "|" << mvObj.releaseDate << "|" << mvObj.writers <<
96             "|" << mvObj.runtime << "|" << mvObj.directors << "|" << mvObj.actors

```

```
<< endl;
93     }
94     outFile.close();
95 }
96
97
98
99 //Add movie to database
100 Movie addMovie() {
101     Movie mvObj;
102     mvObj.title = getInput("Enter new title: ", "Error: enter valid string");
103     mvObj.year = getIntInput("Enter new Year: ", "Error: Enter valid year");
104     mvObj.releaseDate = getInput("Enter new release date: ", "Error: Enter valid
date");
105     mvObj.contentRating = getInput("Enter new content rating: ", "Error: Enter
valid rating");
106     mvObj.genre = getInput("Enter new genre: ", "Error: Enter valid string");
107     mvObj.directors = getInput("Enter new directors: ", "Error: Enter valid
string");
108     mvObj.writers = getInput("Enter new writers: ", "Error: Enter valid string");
109     mvObj.actors = getInput("Enter new actors: ", "Error: Enter valid string");
110     mvObj.rating = getFloatInput("Enter new rating(0-10): ", "Error: Enter valid
number", 0.0, 10.0);
111     return mvObj;
112
113 }
114
115 void searchMovies(vector<Movie> &searchVector) {
116
117     int menuChoice;
118     vector<int> foundMovies;
119     cout << "Please select search type.\n";
120     cout << "1. By Title\n"
121           << "2. By Actor\n"
122           << "3. By Year\n"
123           << "4. By Genre\n"
124           << "Enter a number between 1 -5. Any other number will  \n";
125
126     menuChoice = getIntInput("", "Error: Enter valid number");
127
128     if (menuChoice < 1 || menuChoice > 5) return;
129
130     switch (menuChoice) {
131     case 1: foundMovies = searchTitle(searchVector); break;
132     case 2: foundMovies = searchActor(searchVector); break;
133     case 3: foundMovies = searchYear(searchVector); break;
134     case 4: foundMovies = searchGenre(searchVector); break;
135
136     }
137     int count = 0;
138     if (!foundMovies.empty()) {
139         cout << "Matching Movies found:\n";
```

```
140     for (int i : foundMovies) {
141
142         cout << count << ": " << searchVector[foundMovies[count]].title <<
143             endl;
144         count++;
145     }
146     menuChoice = getIntInput("Enter number of movie to view details", "Error:
147         Enter valid number", 0, count);
148     movieDetails(searchVector, foundMovies[menuChoice]);
149 }
150 else {
151     cout << "No movies matched the search string\n";
152 }
153 }
154 vector<int> searchTitle(vector<Movie> movies) {
155     string searchString = getInput("Enter title of movie to search for: ",
156         "Error: Enter valid string\n");
157     vector<int> movieIndex;
158     int count = 0;
159     for (Movie mvObj : movies) {
160         // cout << mvObj.title.find(searchString) << mvObj.title << endl;
161         if (mvObj.title.find(searchString) != string::npos) {
162             movieIndex.push_back(count);
163         }
164         count++;
165     }
166     return movieIndex;
167 }
168
169 //search by actor
170 vector<int> searchActor(vector<Movie> movies) {
171     string searchString = getInput("Enter title of movie to search for: ",
172         "Error: Enter valid string\n");
173     vector<int> movieIndex;
174     int count = 0;
175     for (Movie mvObj : movies) {
176         // cout << mvObj.title.find(searchString) << mvObj.title << endl;
177         if (mvObj.actors.find(searchString) != string::npos) {
178             movieIndex.push_back(count);
179         }
180         count++;
181     }
182     return movieIndex;
183 }
184
185 //search by year
186 vector<int> searchYear(vector<Movie> movies) {
187     int searchString = getIntInput("Enter new Year", "Error: Enter valid year");
```

```
188     vector<int> movieIndex;
189     int count = 0;
190     for (Movie mvObj : movies) {
191         // cout << mvObj.title.find(searchString) << mvObj.title << endl;
192         if (mvObj.year == searchString) {
193             movieIndex.push_back(count);
194         }
195         count++;
196     }
197
198     return movieIndex;
199 }
200
201 //search by genre
202 vector<int> searchGenre(vector<Movie>movies) {
203     string searchString = getInput("Enter title of movie to search for: ",
204                                     "Error: Enter valid string\n");
205     vector<int> movieIndex;
206     int count = 0;
207     for (Movie mvObj : movies) {
208         // cout << mvObj.title.find(searchString) << mvObj.title << endl;
209         if (mvObj.genre.find(searchString) != string::npos) {
210             movieIndex.push_back(count);
211         }
212         count++;
213     }
214
215     return movieIndex;
216 }
217
218 vector<Movie> loadDB(string strFileName) {
219     /* Reads db file in and loads into vector
220     http://www.fluentcpp.com/2017/04/21/how-to-split-a-string-in-c/
221     */
222     vector<Movie> movies; //holds movie data
223     ifstream dbFile(strFileName); //Open DB File
224
225     if (dbFile.fail()) {
226         cout << "ERROR: Unable to open database file: " << strFileName << endl;
227         exit(1);
228     }
229     cout << "Loading database.....\n";
230     string strHold; //placeholder for getline
231     while (getline(dbFile, strHold)) {
232         //cout << strHold << endl;
233         movies.push_back(parseData(strHold));
234     }
235     dbFile.close();
236     return movies;
237 }
238
```

```
239 }
240
241 //Parsed movie data from comma seperated string
242 ///http://www.fluentcpp.com/2017/04/21/how-to-split-a-string-in-c/
243 Movie parseData(string movieString) {
244     istringstream strStream(movieString); //string stream var, needed to split
245     string tempStr; //temporary holder for splitting string
246     vector<string> parsedItems; //vector to hold the parsed data
247     while (getline(strStream, tempStr, '|')) {
248         parsedItems.push_back(tempStr);
249     }
250
251     //populate Movie struct with data
252     Movie movieObj;
253     //cout << parsedItems[0] << endl;
254     movieObj.title = parsedItems[0];
255     movieObj.studio = parsedItems[1];
256     movieObj.contentRating = parsedItems[2];
257     try {
258         movieObj.year = stoi(parsedItems[3]);
259     }
260     catch (exception e) {
261         movieObj.year = 0000;
262     }
263     try {
264         movieObj.rating = stof(parsedItems[4]);
265     }
266     catch (exception e) {
267         movieObj.rating = 0.0;
268     }
269
270     movieObj.genre = parsedItems[5];
271     movieObj.releaseDate = parsedItems[6];
272     movieObj.writers = parsedItems[7];
273     movieObj.runtime = parsedItems[8];
274     movieObj.directors = parsedItems[9];
275     movieObj.actors = parsedItems[10];
276
277     return movieObj;
278
279
280
281 }
282
283 //Displays initial welcome screen
284 void displayWelcome() {
285     cout <<
286         "-----\n";
287     cout << "Welcome to the Offline Movie Database. Please choose from an
288         option below\n";
289     cout <<
```

```

    "-----\n";
    -\n";
288     cout << "1. Search for a movie\n"
289         << "2. Add new movie\n"
290         << "Any other number to quit\n";
291 }
292
293 void movieDetails(vector<Movie> &movies,int movieIndex) {
294     displayMovie(movies[movieIndex]);
295
296     int menuChoice = getIntInput("Enter 1 to update movie. Enter 2 to delete
    movie. Enter any other number to return to menu\n", "Error: Enter valid
    number");
297     switch (menuChoice) {
298     case 1: {
299         movies[movieIndex] = updateMovie(movies[movieIndex]);
300         break;
301     }
302     case 2: {
303         cout << "Removing " << movies[movieIndex].title << " from data base\n";
304         movies.erase(movies.begin() + movieIndex);
305     }
306     }
307 }
308
309
310 }
311
312 //display movie details
313 void displayMovie(Movie mvObj) {
314     cout << "-----\n";
315     cout << "\tMovie Details\n";
316     cout << "-----\n";
317     cout << left << setw(20) << "0 - Title: " << mvObj.title << endl;
318     cout << left << setw(20) << "1 - Year: " << mvObj.year << endl;
319     cout << left << setw(20) << "2 - Release: " << mvObj.releaseDate << endl;
320     cout << left << setw(20) << "3 - Content Rating: " << mvObj.contentRating <<
    endl;
321     cout << left << setw(20) << "4 - Genre: " << mvObj.genre << endl;
322     cout << left << setw(20) << "5 - Director: " << mvObj.directors << endl;
323     cout << left << setw(20) << "6 - Writers: " << mvObj.writers << endl;
324     cout << left << setw(20) << "7 - Actors: " << mvObj.actors << endl;
325     cout << left << setw(20) << "8 - Rating: " << mvObj.rating << endl;
326 }
327
328 //Update fields of movie
329 Movie updateMovie(Movie mvObj) {
330     int menuChoice;
331     do {
332
333
334         menuChoice = getIntInput("Enter number of field to modify: \n", "Error:

```

```
        Enter valid number", 0, 8);
335     switch (menuChoice) {
336         case 0: mvObj.title = getInput("Enter new title: ", "Error: enter
        valid string"); break;
337         case 1: mvObj.year = getIntInput("Enter new Year: ", "Error: Enter
        valid year"); break;
338         case 2: mvObj.releaseDate = getInput("Enter new release date: ",
        "Error: Enter valid date"); break;
339         case 3: mvObj.contentRating = getInput("Enter new content rating: ",
        "Error: Enter valid rating"); break;
340         case 4: mvObj.genre = getInput("Enter new genre: ", "Error: Enter
        valid string"); break;
341         case 5: mvObj.directors = getInput("Enter new directors: ", "Error:
        Enter valid string"); break;
342         case 6: mvObj.writers = getInput("Enter new writers: ", "Error: Enter
        valid string"); break;
343         case 7: mvObj.actors = getInput("Enter new actors: ", "Error: Enter
        valid string"); break;
344         case 8: mvObj.rating = getFloatInput("Enter new rating(0-10): ",
        "Error: Enter valid number",0,10); break;
345     }
346     displayMovie(mvObj);
347
348 } while (getSentinel("Enter 'y' to make further updates. Enter 'n' to exit:
349     ", "Error: Enter 'y'") == 'y');
350 return mvObj;
351 }
352
353 //Validates input based on a range. returns int
354 int getIntInput(string questionToAsk, string errorMsg) {
355     int usrInput;
356     cout << questionToAsk;
357     while (!(cin >> usrInput) ) { //Loop until integer in the specified range is
        entered
358         cout << errorMsg << endl;
359         cin.clear();
360         cin.ignore(numeric_limits<streamsize>::max(), '\n');
361     }
362     cin.ignore(std::numeric_limits<std::streamsize>::max(), '\n');
363     return int(usrInput);
364 }
365
366 //Validates input based on a range. returns float
367 float getFloatInput(string questionToAsk, string errorMsg, int lowRange, int
        highRange) {
368     float usrInput;
369     cout << questionToAsk;
370     while (!(cin >> usrInput) || usrInput < lowRange || usrInput > highRange)
371     { //Loop until integer in the specified range is entered
372         cout << errorMsg << endl;
373         cin.clear();
```



```
373     cin.ignore(numeric_limits<streamsize>::max(), '\n');
374 }
375 cin.ignore(std::numeric_limits<std::streamsize>::max(), '\n');
376 return usrInput;
377 }
378
379
380 //Validates input based on a range. returns int
381 int getIntInput(string questionToAsk, string errorMsg,int lowRange,int highRange) ↗
{
382     int usrInput;
383     cout << questionToAsk;
384     while (!(cin >> usrInput) || usrInput < lowRange || usrInput > highRange) ↗
    { //Loop until integer in the specified range is entered
385         cout << errorMsg << endl;
386         cin.clear();
387         cin.ignore(numeric_limits<streamsize>::max(), '\n');
388     }
389     cin.ignore(std::numeric_limits<std::streamsize>::max(), '\n');
390     return int(usrInput);
391 }
392
393 //Validates input, returns string
394 string getInput(string questionToAsk, string errorMsg) {
395     string usrInput;
396     cout << questionToAsk;
397     cin.sync();
398
399     getline(cin, usrInput);
400
401     return usrInput;
402 }
403
404
405
406
407 //validates sentinel input, then returns char value
408 char getSentinel(string question,string errMessage) {
409     char varToReturn;
410     bool isValidInput = false;
411
412     // loop until a valid y or n char is entered
413     do {
414         cout << question;
415         if (!(cin >> varToReturn) || (tolower(varToReturn) != 'y' && tolower ↗
            (varToReturn) != 'n')) {
416             cout << errMessage;
417
418             cin.clear();
419             cin.ignore(numeric_limits<streamsize>::max(), '\n');
420         }
421         else {
```

```
422         isValidInput = true;
423     }
424 } while (!isValidInput);
425 cin.ignore(std::numeric_limits<std::streamsize>::max(), '\n');
426 return tolower(varToReturn);
427 }
```