

```

1  /*****
2  * AUTHOR      : JOSHUA SALZEDO, Chris Burrell
3  * LAB #8      : Array Train - Set # 1
4  * CLASS       : CS1A
5  * SECTION     : MW: 8AM
6  * DUE DATE    : 04/23/2018
7  *****/
8  #include <iostream>
9  #include <iomanip>
10 #include "MyHeader.h"
11 #include "functions.h"
12
13 /*****
14 *   Array Train - Set # 1
15 *   -----
16 *   This program outputs min, max, avg, and sum of two parallel arrays
17 *   and outputs selected elements from user input
18 *   -----
19 *   INPUT:
20 *   Age to saerch by
21 *   name so earch by
22 *
23 *   OUTPUT:
24 *   Average age , minimum age, maximum age, sum of ages,
25 *   namees and ages from user input
26 *****/
27
28 int main()
29 {
30 /*****
31 *   CONSTANTS
32 *   -----
33 *   OUTPUT - USED FOR CLASS HEADING
34 *   -----
35 *   PROGRAMMER      :   Programmer's Name
36 *   CLASS           :   students course
37 *   SECTION         :   Class Days and Times
38 *   LAB_NUM         :   Lab Number
39 *   LAB_NAME        :   Title of the lab
40 *   -----
41 *   PROCESSING - USED FOR PROGRAM EXECUTION
42 *   -----
43 *   NAMES           :   hardcoded input names (parallel array)
44 *   AGES            :   hardcoded input ages (parallel array)
45 *   PROMPT_FIND_INT :   Prompt for an age input
46 *   PROMPT_NAME     :   Prompt for a name input
47 *****/
48     const char PROGRAMMER[] = "Joshua Salzedo, Chris Burrell";
49     const char CLASS[] = "CS1A";
50     const char SECTION[] = "MW: 8:00a - 12:00p";
51     const int LAB_NUM = 25;
52     const char LAB_NAME[] = "Array Train # 1";
53
54     const int AR_LEN = 11;
55
56     const string NAMES[] = {
57         "zac",
58         "Kasra",
59         "Bas",
60         "Sara",
61         "Nick",

```

```

62         "Delvin",
63         "Justin",
64         "Abe",
65         "Jeremy",
66         "Farah",
67         "Maryan",
68     };
69     const int AGES[] = {
70         22,
71         75,
72         19,
73         21,
74         18,
75         12,
76         19,
77         5,
78         62,
79         21,
80         21
81     };
82     const string PROMPT_FIND_INT = "Please enter an age you want to look "
83                                     "for: ";
84     const string PROMPT_NAME = "enter a name to find: ";
85
86
87     double averageAges;           // CALC & OUT - average value of AGES
88     //                           array
89     int firstInstanceOfAgeIndex;   // CALC & OUT - index for First position
90     //                           for input int
91     int totalOccurrences;         // CALC & OUT - total occurrences of
92     //                           an input int
93     int nameIndex;               // CALC & OUT - index for found name
94     int sumAges;                 // CALC & OUT - sum of AGES array
95     int youngestPersonIndex;     // CALC & OUT -index of the smallest age
96     int oldestPersonIndex;      // CALC & OUT - index of the biggest age
97     int searchInt;              // IN & OUT - user provided input
98     string searchString;        // IN & OUT - user provided input
99     string formattedString;      // CALC & OUT - Formatted output strings
100    string oldestPerson;         // CALC & OUT - name of oldest person
101    string youngestPerson;       // CALC & OUT - name of youngest person
102
103    // OUTPUT - class header
104    DisplayHeader(PROGRAMMER, CLASS, SECTION, LAB_NUM, LAB_NAME);
105
106    // compute the average of the AGES array
107    averageAges = AverageIntArray(AGES, AR_LEN);
108
109    // set up formatting
110    cout << fixed << setprecision(2);
111    cout << "Average value of ages array is : " << averageAges;
112    // clean up after ourselves
113    cout << scientific << setprecision(6) << endl << endl;
114
115    // find an int to search for
116    searchInt = SaneInputCinInt(0, 100, PROMPT_FIND_INT);
117
118    // find the first instance of the searchInt
119    firstInstanceOfAgeIndex = FindFirstInstance(AGES, AR_LEN, searchInt);
120
121    // check if an instance was actually found
122    if (firstInstanceOfAgeIndex != AR_LEN)

```

```

123 {
124     formattedString = "you found ";
125     formattedString += NAMES[firstInstanceOfAgeIndex];
126     formattedString += " who is ";
127     formattedString += to_string(AGES[firstInstanceOfAgeIndex]);
128     formattedString += " years old at index # ";
129     formattedString += to_string(firstInstanceOfAgeIndex) += '\n';
130
131 }
132 else
133 {
134     formattedString = "Name not found on the list.\n";
135 }
136 cout << formattedString << endl;
137
138 // find total occurrences of the age
139 totalOccurrences = FindOccurrencesInt(AGES, AR_LEN, searchInt);
140 // reinit
141 formattedString = "";
142
143 // switch over # occurrences, for word choice and verb tense.
144 switch (totalOccurrences)
145 {
146     case 0:
147         formattedString = "There is noone with that age.\n";
148         break;
149
150     case 1:
151         formattedString = "there is a grand total of one person with "
152                             "that specific age.";
153         break;
154
155     default:
156         formattedString = "There are a total of ";
157         formattedString += to_string(totalOccurrences);
158         formattedString += " people with that age.";
159         // do something
160         break;
161 }
162
163 cout << formattedString << endl;
164
165
166 cout << PROMPT_NAME;
167 getline(cin, searchString);
168
169 nameIndex = FindString(NAMES, AR_LEN, searchString);
170 formattedString = "";
171
172 if (nameIndex == AR_LEN)
173 {
174     formattedString = "Im sorry, \n";
175     formattedString += searchString += "\n" was not found in my records.";
176 }
177 else
178 {
179     formattedString += NAMES[nameIndex];
180     formattedString += "is";
181     formattedString += to_string(AGES[nameIndex]) += " years old and "
182                                     "exists at index #";
183     formattedString += to_string(nameIndex);

```

```

184     }
185
186     cout << formattedString << endl << endl;
187
188     // fetch the index of the oldest person
189     oldestPersonIndex = FindLargestInt(AGES, AR_LEN);
190     // and assign their name to a value we can more easily use
191     oldestPerson = NAMES[oldestPersonIndex];
192
193     // fetch the index of the youngest person
194     youngestPersonIndex = FindSmallestInt(AGES, AR_LEN);
195     youngestPerson = NAMES[youngestPersonIndex];
196
197     cout << "The oldest person is " << oldestPerson << " who is "
198           << AGES[oldestPersonIndex] << " years old which exists at index #"
199           << oldestPersonIndex << endl;
200
201     cout << "The youngest person is " << youngestPerson << " who is "
202           << AGES[youngestPersonIndex] << " years old which exists at index #"
203           << oldestPersonIndex << endl;
204
205     sumAges = SumIntArray(AGES, AR_LEN);
206     cout << "Overall, the total combined age is " << sumAges << endl;
207
208     cout << left << left;
209     cout << "======" << endl;
210     cout << "===-- Begin Part 3&4 ---==" << endl;
211     cout << "======" << endl;
212
213     cout << " -----= Test #1=-----" << endl;
214
215     cout << fixed << setprecision(2);
216     // compute average
217     averageAges = AverageIntArray(AGES, AR_LEN);
218
219     cout << "The average of the array is " << averageAges << endl;
220     // restore defaults
221     cout << scientific << setprecision(6);
222
223     cout << " -----= Test #2=-----" << endl;
224     // loop over calculating first instance of the array
225     for (int i = 0; i < 4; ++i)
226     {
227         int myInt;
228         int result;
229
230         myInt = SaneInputCinInt(0, 100, PROMPT_FIND_INT);
231         result = FindFirstInstance(AGES, AR_LEN, myInt);
232
233         if (result == AR_LEN)
234         {
235             cout << myInt << " does not exist within the array." << endl;
236         }
237         else
238         {
239             cout << NAMES[result] << " is " << AGES[result] << " years old"
240                   << "(index # " << result << ')' << endl;
241         }
242     }
243 }
244

```

```

245     cout << endl;
246
247     cout << " -----= Test #3=-----" << endl;
248     for (int i = 0; i < 4; ++i)
249     {
250         int instances;
251         int input;
252
253         input = SaneInputCinInt(0, 100, PROMPT_FIND_INT);
254         instances = FindOccurrencesInt(AGES, AR_LEN, input);
255
256         cout << input << " Occured " << instances << " time(s)." << endl;
257     }
258
259     cout << endl;
260     cout << " -----= Test #4=-----" << endl;
261     // test for name search
262     for (int i = 0; i < 4; ++i)
263     {
264         int resultIndex;
265         string searchName;
266
267         cout << PROMPT_NAME;
268         // fetch a name
269         getline(cin, searchName);
270
271         // get the result
272         resultIndex = FindString(NAMES, AR_LEN, searchName);
273
274         if (resultIndex != AR_LEN)
275         {
276             // array hit
277             cout << searchName << " is " << AGES[resultIndex] << " years old"
278                 << "( index # " << resultIndex << ' )';
279         }
280         else
281         {
282             // array miss
283             cout << searchName << " does not exist within the array.";
284         }
285         cout << endl;
286     }
287
288     cout << " -----= Test #5=-----" << endl << endl;
289     {
290         int youngestIndex;
291
292         youngestIndex = FindSmallestInt(AGES, AR_LEN);
293
294         cout << "The youngest person is " << NAMES[youngestIndex]
295             << " who is ";
296         cout << AGES[youngestIndex] << " years old (Index # "
297             << youngestIndex << ' )';
298     }
299     cout << endl;
300     cout << " -----= Test #6=-----" << endl << endl;
301     {
302         int oldestIndex;
303         oldestIndex = FindLargestInt(AGES, AR_LEN);
304
305         cout << "The oldest person is " << NAMES[oldestIndex]

```

```
306         << " who is ";
307     cout << AGES[oldestIndex] << " years old (Index # "
308         << oldestIndex << ')';
309 }
310 cout << endl << " -----= Test #7=-----" << endl << endl;
311 {
312     int mSumAges;
313
314     mSumAges = SumIntArray(AGES, AR_LEN);
315     cout << "Sum of all elements of the array is: " << mSumAges;
316 }
317 cout << endl;
318 cout << "Done.";
319
320 return 0;
321 }
322
323
```