

```
1 // A program to demonstrate the management of a small database of operational
2 // amplifiers.
3 //
4 // General description
5 //
6 // The database contains up to DATABASE_MAX operational amplifier elements.  ↗
7 // Each
8 // element contains the operation amplifier name, the number of pins in the  ↗
9 // package
10 // and stores the slew rate of the device.
11 //
12 // New elements can be added into the database by the user. The database can  ↗
13 // be saved
14 // to disk or loaded from disk. The database elements can be sorted either by  ↗
15 // name or
16 // by slew rate. There is also the facility to display the elements.
17 //
18 // Only a single database is required and the file name is fixed in the code  ↗
19 // (as
20 // DATABASE_FILENAME). This means that each time the database is saved to  ↗
21 // disk,
22 // any previous data in the file is overwritten. Also, when a database is  ↗
23 // loaded
24 // from a file it should overwrite any data already in memory.
25
26 #include <iostream>
27 #include <fstream>
28 #include <string.h>
29 using namespace std;
30
31 // the format of each of the elements in the database
32 struct OpAmps {
33     char Name[20]; // the name of the op-amp (e.g. "741")
34     unsigned int PinCount; // the number of pins in the package
35     double SlewRate; // the slew rate in volts per microsecond
36 };
37
38 // the length of the fixed array to be used for database - must be at least  ↗
39 // one
40 // and no greater the maximum value allowed in an unsigned long (see the file
41 // limits.h).
42 #define DATABASE_MAX 10
43
44 // file used for the database
45 #define DATABASE_FILENAME "database.txt"
46
47 // function prototypes
48 ///////////////////////////////////////////////////<enter code here>
49
50 void Enter(OpAmps &OpAmpVal, unsigned long &database_length);
51
52 void Save(OpAmps *Savetofile, unsigned long &database_length);
53
54 void Load(OpAmps *Loadfromfile, unsigned long &database_length);
55
56 void Sort(OpAmps *SortDB, unsigned long &database_length);
```

```
49
50 void Display(OpAmps *DisplayDB, unsigned long &database_length);
51
52 int SortByName(const void*a, const void* b);
53
54 int SortBySlewRate(const void*a, const void* b);
55
56 // Control the entering, saving, loading, sorting and displaying of elements  ↗
   in the
57 // database.
58 // Arguments: None
59 // Returns: 0 on completion
60 int main()
61 {
62     OpAmps OpAmp[DATABASE_MAX]; // the database
63     unsigned long database_length = 0; // the number of elements in the  ↗
       database
64     char UserInput;
65
66     // loop until the user wishes to exit
67     while (1) {
68
69         // show the menu of options
70         cout << endl;
71         cout << "Op-amp database menu" << endl;
72         cout << "-----" << endl;
73         cout << "1. Enter a new op-amp into the database" << endl;
74         cout << "2. Save the database to disk" << endl;
75         cout << "3. Load the database from disk" << endl;
76         cout << "4. Sort the database" << endl;
77         cout << "5. Display the database" << endl;
78         cout << "6. Exit from the program" << endl << endl;
79
80         // get the user's choice
81         cout << "Enter your option: ";
82         cin >> UserInput;
83         cout << endl;
84
85         // act on the user's input
86         switch (UserInput) {
87             case '1':
88                 Enter(OpAmp[database_length], database_length);
89                 break;
90
91             case '2':
92                 Save(OpAmp, database_length);
93                 break;
94
95             case '3':
96                 Load(OpAmp, database_length);
97                 break;
98
99             case '4':
100                 Sort(OpAmp, database_length);
101                 break;
102
```

```
103     case '5':
104         Display(OpAmp, database_length);
105         break;
106
107     case '6':
108         return 0;
109
110     default:
111         cout << "Invalid entry" << endl << endl;
112         break;
113     }
114 }
115 }
116
117
118 // Allow the user to enter a new element into the database. Note that the data ↗
119 // is simply added to the end the database (if not full) and no sorting is ↗
120 // carried
121 // out.
122 // Arguments:
123 // (1) the element in the database to be entered
124 // (2) the position of the element in the database
125 // Returns: void
126
127 void Enter(OpAmps &OpAmpVal, unsigned long &database_length)
128 {
129     // if the database is full, inform the user
130
131     if (database_length > DATABASE_MAX)
132     {
133         cout << "The database is full" << endl;
134     }
135     // if the database is not full, get the data from the user and alter the ↗
136     // database
137     // length
138     else
139     {
140         OpAmps* pOpAmp = &OpAmpVal;
141
142         cout << "Input the new OpAmp's name" << endl;
143         cin >> pOpAmp->Name;
144         cout << "Input the new OpAmp's pin number" << endl;
145         cin >> pOpAmp->PinCount;
146         cout << "Input the new OpAmp's Slew Rate (V/microseconds)" << endl;
147         cin >> pOpAmp->SlewRate;
148
149         database_length++;
150         return;
151     }
152 }
153 }
154
155
```

```
156 // Save the database to the file specified by DATABASE_FILENAME. If the file
157 // exists it is simply overwritten without asking the user.
158 // Arguments:
159 //   (1) the database
160 //   (2) the length of the database
161 // Returns: void
162
163 void Save(OpAmps *Savetofile, unsigned long &database_length)
164 {
165     fstream output_file; // file stream for output
166
167     output_file.open(DATABASE_FILENAME, ios::out); // open the file
168
169     if (!output_file.good())
170     {
171         // The file could not be opened
172         cerr << "FATAL ERROR: Could not create file database.";
173         exit(1);
174     }
175
176     // write length information to file
177     else
178     {
179         output_file << database_length << endl;
180     }
181
182     // write data to file
183     for (int i = 0; i < database_length; i++)
184     {
185         output_file << endl;
186         output_file << (Savetofile + i)->Name;
187         output_file << endl;
188         output_file << (Savetofile + i)->PinCount;
189         output_file << endl;
190         output_file << (Savetofile + i)->SlewRate;
191         output_file << endl;
192     }
193
194     // close the file
195     output_file.close();
196 }
197
198
199 // Load the database from the file specified by DATABASE_FILENAME. If the file
200 // exists it simply overwrites the data currently in memory without asking
201 // the user.
202 // Arguments:
203 //   (1) the database
204 //   (2) the length of the database
205 // Returns: void
206
207 void Load(OpAmps *Loadfromfile, unsigned long &database_length)
208 {
209     fstream input_file; // file stream for input
210
211     input_file.open(DATABASE_FILENAME, ios::in); // open the file
```

```
212
213     if (!input_file.good())
214     {
215         cerr << "FATAL ERROR: Could not read file database.";
216         exit(1);
217     }
218
219     // load database length information from file
220     else
221     {
222         input_file >> database_length;
223         input_file << endl;
224     }
225
226     // load data from file
227     for (int i = 0; i < database_length; i++)
228     {
229         input_file << endl;
230         input_file >> (Loadfromfile + i)->Name;
231         input_file << endl;
232         input_file >> (Loadfromfile + i)->PinCount;
233         input_file << endl;
234         input_file >> (Loadfromfile + i)->SlewRate;
235         input_file << endl;
236     }
237
238     // close the file
239     input_file.close();
240 }
241
242
243 // Sort the database either using the name of the op-amps or using the slew
244 // rate values.
245 // Arguments:
246 // (1) the database
247 // (2) the length of the database
248 // Returns: void
249
250 void Sort(OpAmps *SortDB, unsigned long &database_length)
251 {
252     char UserInput;
253
254     // show the menu of options
255     cout << endl;
256     cout << "Sorting options" << endl;
257     cout << "-----" << endl;
258     cout << "1. To sort by name" << endl;
259     cout << "2. To sort by slew rate" << endl;
260     cout << "3. No sorting" << endl << endl;
261
262     // get the user's choice of sorting operation required
263     cout << "Enter your option: ";
264     cin >> UserInput;
265     cout << endl;
266
267     // act on the user's input
```

```
268     switch (UserInput)
269     {
270     case '1':
271         qsort(SortDB, database_length, sizeof(OpAmps), SortByName);
272         break;
273
274     case '2':
275         qsort(SortDB, database_length, sizeof(OpAmps), SortBySlewRate);
276         break;
277
278     case '3':
279         break;
280     }
281 }
282
283 // Compare function for qsort, to help sort the elements by the Name member of
284 // OpAmps.
285 // Items should be sorted into alphabetical order.
286 // Arguments:
287 //     (1) a database item
288 //     (2) a database item
289 // Returns: result of the comparison
290 //
291
292 int SortByName(const void*a, const void* b)
293 {
294     return (*((OpAmps*)a)->Name - *((OpAmps*)b)->Name);
295 }
296
297 // Compare function for qsort, to help sort the elements by the SlewRate
298 // member of
299 // OpAmps.
300 // Items should be sorted in increasing value of slew rate.
301 // Arguments:
302 //     (1) a database item
303 //     (2) a database item
304 // Returns: result of the comparison
305
306 int SortBySlewRate(const void*a, const void* b)
307 {
308     return (((OpAmps*)a)->SlewRate - ((OpAmps*)b)->SlewRate);
309 }
310
311 // Display all of the messages in the database.
312 // Arguments:
313 //     (1) the database
314 //     (2) the length of the database
315 // Returns: void
316
317 void Display(OpAmps *DisplayMessages, unsigned long &database_length)
318 {
319     fstream input_file;
320
321     input_file.open(DATABASE_FILENAME, ios::in);
322     input_file >> database_length;
323     input_file << endl;
```

```
323     // if the database is empty, inform the user
324     if (database_length == 0)
325     {
326         cout << "The database is empty";
327         return;
328     }
329
330
331     // if the database is not empty, display all the elements in the database
332     else
333         for (int i = 0; i < database_length; i++)
334         {
335             cout << endl;
336             cout << (DisplayMessages + i)->Name;
337             cout << endl;
338             cout << (DisplayMessages + i)->PinCount;
339             cout << endl;
340             cout << (DisplayMessages + i)->SlewRate;
341             cout << endl;
342         }
343 }
```