```
 1  /*  Program: A5P1-Text Analyzer
 2      Author: Tom Stutler
 3      Last Date Modified: 5/7/2015
 4
 5      The intent of this program is to take a passage from a text file,
 6      analyze it, then output the analysis to a new file.
 7  */
 8
 9  #include <iostream>
10  #include <fstream>
11
12  using namespace std;
13
14  string get_inFile(ifstream& fin);
15  int count_Words(ifstream& fin);
16  void count_searchWord(ofstream& fout, string* arrayParam, string wordParam, int
sizeParam);
17  string* analyze_Text(ifstream& fin, ofstream& fout, int wordCount, float& small,
float& large);
18  void list_shortWords(ofstream& fout, string *arrayParam, int sizeParam, float
wordParam);
19  void list_longWords(ofstream& fout, string *arrayParam, int sizeParam, float
wordParam);
20
21  int main()
22  {
23      string userInFile, userOutFile, searchWord, *wordsArray;
24      int sel, totalCount, searchCount;
25      float smallest, largest;
26      ofstream outFile;
27      ifstream inFile;
28
29      cout << "Enter output file name: ";
30      getline(cin, userOutFile);
31      outFile.open(userOutFile.c_str());
32      if (outFile.fail()) {
33          cout << "Opening " << userOutFile << " failed.\n"
34               << "Please restart the program and try again.\n";
35      }
36
37      do
38      {
39          cout << "\nSelect an option:\n"
40               << "\t1 - Enter input file name\n"
41               << "\t2 - Determine word statistics\n"
42               << "\t3 - List shortest words\n"
43               << "\t4 - List longest words\n"
44               << "\t5 - Search for a word\n"
45               << "\t6 - Exit\n";
46
47          cin >> sel;
48          cin.ignore();
49
50          switch (sel)
51          {
52          case 1:
53              userInFile = get_inFile(inFile);
54              totalCount = count_Words(inFile);
55              break;
56
57          case 2:
58              cout << "\nFilename: " << userInFile << endl;
59              outFile << "\nFilename: " << userInFile << endl;
60              wordsArray = analyze_Text(inFile, outFile, totalCount, smallest,
largest);
61              break;
```

```
62
63          case 3:
64              cout << "\nFilename: " << userInFile << endl;
65              outFile << "\nFilename: " << userInFile << endl;
66              list_shortWords(outFile, wordsArray, totalCount, smallest);
67              break;
68
69          case 4:
70              cout << "\nFilename: " << userInFile << endl;
71              outFile << "\nFilename: " << userInFile << endl;
72              list_longWords(outFile, wordsArray, totalCount, largest);
73              break;
74
75          case 5:
76              cout << "\nFilename: " << userInFile << endl;
77              cout << "\nEnter word to find in file: ";
78              cin >> searchWord;
79              count_searchWord(outFile, wordsArray, searchWord, totalCount);
80              break;
81
82          case 6:
83              cout << "Thank you, come again!\n";
84              break;
85
86          default:
87              cout << "Invalid entry, try again.\n";
88              break;
89          }
90      } while (sel!=6);
91  }
92
93  string get_inFile(ifstream& fin)
94  {
95      string fileName;
96
97      if (fin.is_open()) {
98          fin.close();
99      }
100
101     cout << "Enter input file name: ";
102     getline(cin, fileName);
103     cout << "Filename entered: " << fileName << endl;
104     fin.open(fileName.c_str());
105
106     if (fin.fail()) {
107         cout << "Input file openeing failed in get_inFile\n"
108              << "File name: " << fileName << endl
109              << "Please exit the program and try again.\n";
110     } else {
111         return fileName;
112     }
113 }
114
115 int count_Words(ifstream& fin)
116 {
117     int count=0;
118     string word;
119
120     fin >> word;
121
122     do
123     {
124         if (word!="\n" && word!=" ") {
125             count++;
126         }
127         fin >> word;
```

```
128
129      } while (!fin.eof());
130
131      return count;
132  }
133
134  void count_searchWord(ofstream& fout, string* arrayParam, string wordParam, int
sizeParam)
135  {
136      int count=0;
137
138      for (int i=0; i<sizeParam; i++) {
139          if (*(arrayParam+i) == wordParam) {
140              count++;
141          }
142      }
143
144      cout << "Search word: " << wordParam << endl
145           << "This word appears " << count << " time(s) in the file.\n";
146
147      fout << "Search word: " << wordParam << endl
148           << "This word appears " << count << " time(s) in the file.\n";
149  }
150
151  string* analyze_Text(ifstream& fin, ofstream& fout, int wordCount, float& small,
float& large)
152  {
153      int index=0, punctCount=0, charCount=0;
154      float avg=0;
155      string word, *temp = new string[wordCount];
156
157      if (fin.is_open()) {
158
159          fin.clear();
160          fin.seekg(0);
161
162          fin >> word;
163          small = word.length();
164          large = word.length();
165
166          do
167          {
168              if (word!="\n") {
169
170                  //Checks each word for leading/trailing punctuation,
171                  //then counts and removes it.
172                  while (ispunct(word[0]) || ispunct(word[word.length()-1]))
173                  {
174                      if (ispunct(word[0])) {
175                          word.erase(word.begin());
176                          punctCount++;
177                      }
178                      if (ispunct(word[word.length()-1])) {
179                          word.erase(word.end()-1);
180                          punctCount++;
181                      }
182                  }
183
184                  charCount += word.length();
185
186                  //Checks each word if it's the smallest.
187                  if (small>word.length()) {
188                      small = word.length();
189                  }
190                  //Checks each word if it's the largest.
191                  if (large<word.length()) {
```

```
                    large = word.length();
                }

                //Adds the word to the array.
                *(temp+index) = word;
                index++;
            }
            fin >> word;

        } while(!fin.eof());

        avg = static_cast<float>(charCount)/static_cast<float>(wordCount);

        cout << "\nTotal number of words = " << wordCount
             << "\nAverage word length = " << avg << " characters."
             << "\nTotal number of word characters = " << charCount
             << "\nTotal number of punctuation characters = " << punctCount
             << "\nShortest word length = " << small
             << "\nLongest word length = " << large << endl;

        fout << "\nTotal number of words = " << wordCount
             << "\nAverage word length = " << avg << " characters."
             << "\nTotal number of word characters = " << charCount
             << "\nTotal number of punctuation characters = " << punctCount
             << "\nShortest word length = " << small
             << "\nLongest word length = " << large << endl;

        return temp;

    } else {
        cout << "No input file currently open.\n";
        delete [] temp;
    }
}

void list_shortWords(ofstream& fout, string *arrayParam, int sizeParam, float
wordParam)
{
    string currWord;

    cout << "Shortest words in file:\n";
    fout << "Shortest words in file:\n";

    for (int i=0; i<sizeParam; i++) {
        currWord = *(arrayParam+i);
        if (currWord.length()==wordParam) {
            cout << currWord << endl;
            fout << currWord << endl;
        }
    }
}

void list_longWords(ofstream& fout, string *arrayParam, int sizeParam, float
wordParam)
{
    string currWord;

    cout << "Longest words in file:\n";
    fout << "Longest words in file:\n";

    for (int i=0; i<sizeParam; i++) {
        currWord = *(arrayParam+i);
        if (currWord.length()==wordParam) {
            cout << currWord << endl;
            fout << currWord << endl;
        }
```

```
256       }
257   }
```