Keurig

Julian van Doorn (s2518074)

October 11, 2019

1 Explanation

This program aims to provide the functionality described in the assignment 'Keurig'. It will ask the user for an input file, output file and a tab width. It will then parse the input file and export it to the output file. It will remove comments created with '//' and properly indent the code. Furthermore it will look for Lychrel numbers.

2 Time

Writing the basis took around 3 hours I think (I had some issues with reading/writing files). After that an additional hour debugging and improving code quality. An additional half an hour was spent writing documentation and the report.

Code

```
1  /*
2   * doorn2.cpp
3   * Author: Julian van Doorn (2518074)
4   * October 10th 2019
5   *
6   * This program aims to provide the functionality described in assignment 2 'Keurig
        '. It will ask the user to input a
7   * tab-width and the file they want to check and where the result should go to. The
        program will remove all comments
8   * made using // and indent it properly. Additionally it will look for Lychrel
        numbers.
9   */
10
11 #include <iostream>
12 #include <fstream>
13
14 bool is_number(char c) {
15     /*
16      * Checks if a character is a number.
17      */
18     return ('0' <= c && c <= '9');
19 }
20
21 int reverse_int(int n) {
22     /*
23      * Reverses an integer.
```

```
24          */
25          int reversed_n = 0, remainder;
26          while (n != 0) {
27              // The remainder is the last digit in the number.
28              remainder = n % 10;
29
30              // Shift all digits in the reversed number to the right.
31              reversed_n *= 10;
32              // Add the remainder to the reversed number.
33              reversed_n += remainder;
34              // Divide the original number by 10 (floating point will be ignored).
35              n /= 10;
36          }
37          // Return the reversed number.
38          return reversed_n;
39      }
40
41      int lychrel(int n) {
42          /*
43           * Tries to compute a palindrome for n. If it fails to do so before reaching
                 INT_MAX we will assume that it is a
44           * Lychrel number. If the return is < 0 we reached INT_MAX, if the return is > 0
                 then that is the amount of
45           * iterations it took to find a palindrome.
46           */
47          int rn, j = 1;
48          while (true) {
49              // Get the reverse of N.
50              rn = reverse_int(n);
51              if (n == rn) {
52                  // N is equal to RN thus a palindrome.
53                  return j;
54              } else {
55                  // See if we can add N and RN together without exceeding INT_MAX.
56                  if (n < INT_MAX - rn) {
57                      // If we can add them we do so.
58                      n += rn;
59                  } else {
60                      // If not we return j negated (we haven't found a palindrome).
61                      return -j;
62                  }
63              }
64              // Increase j with each iteration.
65              j++;
66          }
67      }
68
69      int main() {
70          std::cout << "=============================================" << std::endl
71                    << "|This program is written by:               |" << std::endl
72                    << "|Julian van Doorn (s2518074 - 2019)         |" << std::endl
73                    << "|                                           |" << std::endl
74                    << "|The assignment is called: 'Keurig'         |" << std::endl
75                    << "|                                           |" << std::endl
76                    << "|It will ask you to enter a input filename, |" << std::endl
77                    << "|an output filename and a tab-width. It will|" << std::endl
```

```
78              << "|then format the input file and export it to|" << std::endl
79              << "|the output file.                           |" << std::endl
80              << "=============================================" << std::endl;
81
82      int d = 0; // The current depth
83      int tab; // The width of a tab
84
85      char previous; // The previous character
86      char current; // The current character
87
88      int total_in = 1; // Total amount of characters that have been read.
89      int total_out = 0; // Total amount of characters that have been written.
90
91      bool comment = false; // Checks if we are in a comments currently.
92
93      std::string number; // A number if we find one (Lychrel).
94      int iterations = 0; // Biggest amount of iterations for the Lychrel numbers.
95      int integer = 0; // The integer corresponding to those iterations.
96
97      std::string input_file; // The input file.
98      std::cout << "Which file would you like to parse: ";
99      std::cin >> input_file;
100     std::ifstream input(input_file); // Input stream
101     // Check if we can read the file (a.k.a. does it exist?).
102     if (input.fail()) {
103         std::cout << "The file does not exist." << std::endl;
104         return 1;
105     }
106
107     std::string output_file; // the output file.
108     std::cout << "Where would you like the output to go: ";
109     std::cin >> output_file;
110     std::ofstream output(output_file); // Output stream
111
112     std::cout << "What tab with would you like to use: ";
113     std::cin >> tab;
114
115     // Start parsing the file
116     while (input.get(current)) {
117         total_in += 1; // We've read an additional character.
118
119         // Check if we are currently in a comment.
120         if (comment) {
121             // Linebreaks end comments.
122             if (current == '\n') {
123                 comment = false;
124                 output.put('\n');
125                 total_out += 1;
126                 previous = '\n';
127                 continue;
128             } else {
129                 // Skip the character
130                 continue;
131             }
132         }
133
```

```
134         // Check if we are at the start of a comment
135         if (previous == '/') {
136             if (current == '/') {
137                 comment = true;
138                 continue;
139             } else {
140                 output.put(previous);
141                 total_out += 1;
142             }
143         }
144 
145         // Check if we just had a newline
146         if (previous == '\n') {
147             if (current == ' ' || current == '\t') {
148                 // Skip all spaces/tabs after a newline.
149                 continue;
150             } else {
151                 // After removing all spaces/tabs insert our own.
152                 int indent;
153                 if (current == '}') {
154                     indent = tab * (d - 1);
155                 } else {
156                     indent = tab * d;
157                 }
158                 for (int i = 0; i < indent; i++) {
159                     output.put(' ');
160                     total_out += 1;
161                 }
162             }
163         }
164 
165         if (current == '{') {
166             // { increase indent level
167             d += 1;
168         } else if (current == '}') {
169             // { decrease indent level
170             d -= 1;
171         } else if (is_number(current)) {
172             // numbers are added to the current number string
173             number += current;
174         } else if (is_number(previous)) {
175             // we just had a number, now we need to parse it.
176             int i = std::stoi(number);
177             int l = lychrel(i);
178 
179             if (l < 0) {
180                 std::cout << "Found the number: " << i
181                           << " it does not become a palindrome before reaching
                                INT_MAX after " << -l << " iterations."
182                           << std::endl;
183                 if (-l > iterations) {
184                     iterations = -l;
185                     integer = i;
186                 }
187             } else {
```

```
188                 std::cout << "Found the number: " << i << " it becomes a palindrome
                        after " << l << " iterations."
189                           << std::endl;
190                 if (l > iterations) {
191                     iterations = l;
192                     integer = i;
193                 }
194             }
195 
196             number = "";
197         }
198 
199         if (current != '/') {
200             output.put(current);
201             total_out += 1;
202         }
203         previous = current;
204     }
205 
206     // Close in and output.
207     input.close();
208     output.close();
209 
210     // give a summary
211     std::cout << "We have read " << total_in << " and written " << total_out << "
            characters." << std::endl;
212     if (iterations != 0) {
213         std::cout << "The biggest amount of iterations for Lychrel was " <<
                iterations << " for the integer " << integer
214                   << std::endl;
215     } else {
216         std::cout << "No integers were tested if they are Lychrel numbers.";
217     }
218 
219     return 0;
220 }
```