WORKED EXAMPLE 8.1 Looking for for Duplicates

Your task is to write a program that reads a file and prints all lines that contain a repeated word (such as an accidental "the the"), together with their line numbers.

Step 1 Understand the processing task.

Whenever we find a line containing a repeated word, we are to print it like this:

```
 360:bat?' when suddenly, thump! thump! down she came upon a heap of
2103:'Twinkle, twinkle, twinkle, twinkle--' and went on so long that
```

A word is only counted as repeated when it is the same as its predecessor. For example, a line that contains two "the" that are not adjacent would not be reported. The words must be exactly the same. For example, "Twinkle" and "twinkle" don't match.

Step 2 Determine which files you need to read and write.

We only need to read one file, the one with the words. The result is displayed in the console window; no output file is required.

Step 3 Choose a method for obtaining the file names.

This is a student program with console output; we'll ask the user through the console.

Step 4 Choose between line, word, and character-based input.

We definitely want to use line-based input because we need to count line numbers and print the entire line if it contains repeating words.

Step 5 With line-oriented input, extract the required data.

When we have an input line, we still need to extract the words. The easiest approach is to use a string stream, and read words off that stream. We will keep a variable that holds the previous word.

```
For each word in the line
    If word equals previous word
        Found a duplicate.
    Else
        previous word = word
```

Step 6 Place repeatedly occurring tasks into functions.

In this program, there are no repeated tasks. But let's take the bigger view. Scanning lines and printing out the ones that match a particular criterion is a fairly common task. Therefore, let's put the checking for repeated words into a separate function,

```
bool has_repeated_words(string line)
```

Then the basic processing loop becomes very simple:

```
string line;
int line_number = 0;
while (getline(in_file, line))
{
   line_number++;
   if (has_repeated_words(line))
   {
      cout << setw(7) << line_number << ":" << line << endl;
   }
}
```

Step 7 If required, use manipulators to format the output.

There is only one formatting job: to print the line numbers so that the lines line up. Since an integer has no more than 7 digits, we use

```
cout << setw(7) << line_number << ":" << line << endl;
```

Here's the complete program, `ch08/repeated.cpp`:

```
#include <fstream>
#include <iostream>
#include <iomanip>
#include <sstream>
#include <string>

using namespace std;

/**
   Checks whether a given line has repeated words (such as “the the”).
   @param line a line of text
   @return true if the line contains repeated words
*/
bool has_repeated_words(string line)
{
   istringstream strm;
   strm.str(line); // This string stream reads the contents of the line
   string previous_word = "";
   string word;
   while (strm >> word) // For each word in the line
   {
      if (word == previous_word) // Found a duplicate
      {
         return true;
      }
      else // Remember this word for the next iteration
      {
         previous_word = word;
      }
   }
   return false;
}

int main()
{
   string filename;
   cout << "Enter filename: ";
   cin >> filename;
   ifstream in_file;
   in_file.open(filename.c_str());

   int line_number = 0;
   string line;
   while (getline(in_file, line)) // For each line in the file
   {
      line_number++;
      // Print line if it has repeated words
      if (has_repeated_words(line))
      {
         cout << setw(7) << line_number << ":" << line << endl;
      }
   }
   return 0;
}
```