Reading and Writing '.csv' files in C++

Johnjimy K. Som

June 9, 2021

Documentation of a C++ program 'read-write-csv.cpp' that reads and writes copies of '.csv' files using additional includes. The program also creates additional copies that checks that the .csv file has been read. More information can be found in: https://www.gormanalysis.com/blog/reading-and-writing-csv-files-with-cpp/writing-to-csv

1 read-write-csv.cpp

```cpp
#include <string>
#include <fstream>
#include <vector>
#include <utility> // std::pair
#include <stdexcept> // std::runtime_error
#include <sstream> // std::stringstream

std::vector<std::pair<std::string, std::vector<int>>> read_csv(std::string filename) {
  // Reads a CSV file into a vector of <string, vector<int>> pairs where
  // each pair represents <column name, column values>

  // Create a vector of <string, int vector> pairs to store the result
  std::vector<std::pair<std::string, std::vector<int>>> result;

  // Create an input filestream
  std::ifstream myFile(filename);

  // Make sure the file is open
  if (!myFile.is_open()) throw std::runtime_error("Could not open file");

  // Helper vars
  std::string line, colname;
  int val;

  // Read the column names
  if (myFile.good())
  {
    // Extract the first line in the file
    std::getline(myFile, line);

    // Create a stringstream from line
    std::stringstream ss(line);

    // Extract each column name
    while (std::getline(ss, colname, ',')) {

      // Initialize and add <colname, int vector> pairs to result
      result.push_back({ colname, std::vector<int> {} });
    }
  }

  // Read data, line by line
  while (std::getline(myFile, line))
  {
    // Create a stringstream of the current line
    std::stringstream ss(line);

    // Keep track of the current column index
    int colIdx = 0;

    // Extract each integer
    while (ss >> val) {

      // Add the current integer to the 'colIdx' column's values vector
      result.at(colIdx).second.push_back(val);

      // If the next token is a comma, ignore it and move on
      if (ss.peek() == ',') ss.ignore();

      // Increment the column index
      colIdx++;
    }
  }

  // Close file
  myFile.close();

  return result;
}


void write_csv(std::string filename, std::string colname, std::vector<int> vals) {
  // Make a CSV file with one column of integer values
  // filename - the name of the file
  // colname - the name of the one and only column
  // vals - an integer vector of values

  // Create an output filestream object
```

```cpp
    std::ofstream myFile(filename);

    // Send the column name to the stream
    myFile << colname << "\n";

    // Send data to the stream
    for (int i = 0; i < vals.size(); ++i)
    {
      myFile << vals.at(i) << "\n";
    }

    // Close the file
    myFile.close();
  }

  void write_csv(std::string filename, std::vector<std::pair<std::string, std::vector<int>>> dataset) {
    // Make a CSV file with one or more columns of integer values
    // Each column of data is represented by the pair <column name, column data>
    //   as std::pair<std::string, std::vector<int>>
    // The dataset is represented as a vector of these columns
    // Note that all columns should be the same size

    // Create an output filestream object
    std::ofstream myFile(filename);

    // Send column names to the stream
    for (int j = 0; j < dataset.size(); ++j)
    {
      myFile << dataset.at(j).first;
      if (j != dataset.size() - 1) myFile << ","; // No comma at end of line
    }
    myFile << "\n";

    // Send data to the stream
    for (int i = 0; i < dataset.at(0).second.size(); ++i)
    {
      for (int j = 0; j < dataset.size(); ++j)
      {
        myFile << dataset.at(j).second.at(i);
        if (j != dataset.size() - 1) myFile << ","; // No comma at end of line
      }
      myFile << "\n";
    }

    // Close the file
    myFile.close();
  }
```

```cpp

int main() {

  // Make a vector of length 100 filled with 1s
  std::vector<int> vec(100, 1);

  // Write the vector to CSV
  write_csv("ones.csv", "Col1", vec);

  // Make three vectors, each of length 100 filled with 1s, 2s, and 3s
  std::vector<int> vec1(100, 1);
  std::vector<int> vec2(100, 2);
  std::vector<int> vec3(100, 3);

  // Wrap into a vector
  std::vector<std::pair<std::string, std::vector<int>>> vals = { {"One", vec1}, {"Two", vec2}, {"Three", vec3} };

  // Write the vector to CSV
  write_csv("three_cols.csv", vals);
  // Read three_cols.csv and ones.csv
  std::vector<std::pair<std::string, std::vector<int>>> three_cols = read_csv("three_cols.csv");
  std::vector<std::pair<std::string, std::vector<int>>> ones = read_csv("ones.csv");

  // Write to another file to check that this was successful
  write_csv("three_cols_copy.csv", three_cols);
  write_csv("ones_copy.csv", ones);

  return 0;
}
```

Listing 1: read and write '.csv' in one '.cpp' file