Project 1

Generated by Doxygen 1.12.0

1 Class Index 1

1.1 Class List . . . 1

2 File Index 3

2.1 File List . . . 3

3 Class Documentation 5

3.1 Buffer Class Reference . . . 5

3.1.1 Detailed Description . . . 6

3.1.2 Member Function Documentation . . . 6

3.1.2.1 get_state_zip_codes() . . . 6

3.1.2.2 parse_csv_line() . . . 6

3.1.2.3 read_csv() . . . 7

3.1.3 Member Data Documentation . . . 8

3.1.3.1 records . . . 8

3.2 CSVProcessing Class Reference . . . 9

3.2.1 Detailed Description . . . 9

3.2.2 Member Function Documentation . . . 9

3.2.2.1 addHeader() . . . 9

3.2.2.2 csvOutput() . . . 10

3.2.2.3 sortBuffer() . . . 11

3.3 ZipCodeRecord Struct Reference . . . 12

3.3.1 Detailed Description . . . 12

3.3.2 Member Data Documentation . . . 12

3.3.2.1 latitude . . . 12

3.3.2.2 longitude . . . 13

3.3.2.3 state_id . . . 13

3.3.2.4 zip_code . . . 13

4 File Documentation 15

4.1 C:/Users/mujah/OneDrive/Desktop/project/doxy/buffer.cpp File Reference . . . 15

4.1.1 Detailed Description . . . 15

4.2 buffer.cpp . . . 16

4.3 C:/Users/mujah/OneDrive/Desktop/project/doxy/buffer.h File Reference . . . 17

4.3.1 Detailed Description . . . 18

4.4 buffer.h . . . 19

4.5 C:/Users/mujah/OneDrive/Desktop/project/doxy/CSVProcessing.cpp File Reference . . . 19

4.6 CSVProcessing.cpp . . . 20

4.7 C:/Users/mujah/OneDrive/Desktop/project/doxy/CSVProcessing.h File Reference . . . 21

4.8 CSVProcessing.h . . . 22

4.9 C:/Users/mujah/OneDrive/Desktop/project/doxy/main.cpp File Reference . . . 23

4.9.1 Function Documentation . . . 24

4.9.1.1 csvConvert_sort() . . . 24

4.9.1.2 main() 24
4.10 main.cpp 25

Index **27**

Chapter 1

Class Index

1.1 Class List

Here are the classes, structs, unions and interfaces with brief descriptions:

Buffer
: Buffer class for reading and storing Zip Code records from a CSV file 5

CSVProcessing 9

ZipCodeRecord
: Structure to hold a single Zip Code record 12

Chapter 2

File Index

2.1 File List

Here is a list of all files with brief descriptions:

C:/Users/mujah/OneDrive/Desktop/project/doxy/buffer.cpp
Implementation of the Buffer class and ZipCodeRecord struct 15
C:/Users/mujah/OneDrive/Desktop/project/doxy/buffer.h
Header file for the Buffer class and ZipCodeRecord struct 17
C:/Users/mujah/OneDrive/Desktop/project/doxy/CSVProcessing.cpp 19
C:/Users/mujah/OneDrive/Desktop/project/doxy/CSVProcessing.h 21
C:/Users/mujah/OneDrive/Desktop/project/doxy/main.cpp 23

Chapter 3

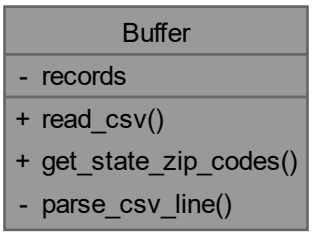
Class Documentation

3.1 Buffer Class Reference

Buffer class for reading and storing Zip Code records from a CSV file.

`#include <buffer.h>`

Collaboration diagram for Buffer:



Public Member Functions

- bool read_csv ()

  Reads the CSV file and populates the zip code records.
- std::map< std::string, std::vector< ZipCodeRecord > > get_state_zip_codes () const

  Retrieves the records grouped by state.

Private Member Functions

- ZipCodeRecord parse_csv_line (const std::string &line) const

  Parses a single CSV line into a ZipCodeRecord.

Private Attributes

- std::vector< ZipCodeRecord > records

3.1.1 Detailed Description

Buffer class for reading and storing Zip Code records from a CSV file.

This class handles reading Zip Code data from a CSV file, storing the records, and providing a method to retrieve the records grouped by state. The records include information such as the Zip Code, state, and geographical coordinates.

Definition at line 43 of file buffer.h.

3.1.2 Member Function Documentation

3.1.2.1 get_state_zip_codes()

```
std::map< std::string, std::vector< ZipCodeRecord > > Buffer::get_state_zip_codes () const
```

Retrieves the records grouped by state.

Groups the Zip Code records by state.

This function organizes the Zip Code records by state and returns a map where each state ID maps to a vector of its corresponding ZipCodeRecord structures.

Returns

A map with state IDs as keys and vectors of ZipCodeRecord as values.

This function organizes the Zip Code records into a map where each state ID is a key, and the value is a vector of ZipCodeRecord structures associated with that state.

Returns

A map with state IDs as keys and vectors of ZipCodeRecord structures as values.

Definition at line 59 of file buffer.cpp.

Here is the caller graph for this function:

3.1.2.2 parse_csv_line()

```
ZipCodeRecord Buffer::parse_csv_line (
            const std::string & line) const  [private]
```

Parses a single CSV line into a ZipCodeRecord.

Parses a line from the CSV into a ZipCodeRecord.

This function takes a line of CSV data as input and converts it into a ZipCodeRecord structure, extracting fields like the Zip Code, state ID, latitude, and longitude.

Parameters

line	A string representing a single line from the CSV file.

Returns

A ZipCodeRecord structure containing the parsed data.

This function takes a single line of CSV data and extracts the Zip Code, state ID, latitude, and longitude to populate a ZipCodeRecord structure.

Parameters

line	A string representing a single line from the CSV file.

Returns

A ZipCodeRecord structure containing the parsed data.

Definition at line 80 of file buffer.cpp.

Here is the caller graph for this function:

3.1.2.3 read_csv()

```
bool Buffer::read_csv ()
```

Reads the CSV file and populates the zip code records.

Reads the CSV file and stores the zip code records.

This function reads a CSV file containing Zip Code data and stores the data in a vector of ZipCodeRecord structures.

Parameters

file_name	The path to the CSV file containing Zip Code data.

Returns

True if the file is successfully read and parsed, false otherwise.

This function opens the CSV file, reads its contents, and parses each line into a ZipCodeRecord, which is stored in a vector.

Parameters

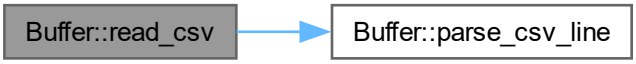
file_name	The path to the CSV file (us_postal_codes.csv).

Returns

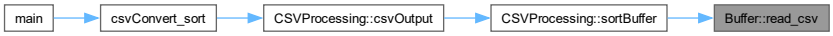
True if the file is read successfully, false otherwise.

Definition at line 28 of file buffer.cpp.

Here is the call graph for this function:



Here is the caller graph for this function:



3.1.3 Member Data Documentation

3.1.3.1 records

```
std::vector<ZipCodeRecord> Buffer::records  [private]
```

Definition at line 67 of file buffer.h.

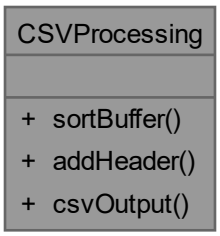
The documentation for this class was generated from the following files:

- C:/Users/mujah/OneDrive/Desktop/project/doxy/buffer.h
- C:/Users/mujah/OneDrive/Desktop/project/doxy/buffer.cpp

3.2 CSVProcessing Class Reference

`#include <CSVProcessing.h>`

Collaboration diagram for CSVProcessing:



Public Member Functions

- map< string, vector< ZipCodeRecord > > sortBuffer ()

  Sorts the CSV buffer and finds the zip codes (eastmost, westmost, northmost, southmost) for each state.
- void addHeader (string &file_name)

  Creates and adds a header to the CSV file.
- bool csvOutput (string &file_name)

  Outputs the processed zip code data to a CSV file.

3.2.1 Detailed Description

Definition at line 11 of file CSVProcessing.h.

3.2.2 Member Function Documentation

3.2.2.1 addHeader()

```
void CSVProcessing::addHeader (
            string & file_name)
```

Creates and adds a header to the CSV file.

This function adds a header row to the specified CSV file. The header includes columns for State, Easternmost, Westernmost, Northernmost, and Southernmost zip codes.

Parameters

file_name	The name of the CSV file to which the header will be added.

Definition at line 92 of file CSVProcessing.cpp.

Here is the caller graph for this function:

3.2.2.2 csvOutput()

```
bool CSVProcessing::csvOutput (
            string & file_name)
```

Outputs the processed zip code data to a CSV file.

This function takes the sorted buffer of zip code records and writes them to a CSV file. Each row contains the state ID and the zip codes for the easternmost, westernmost, northernmost, and southernmost points in that state.

Parameters

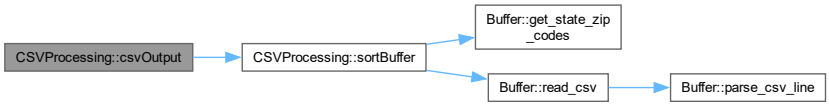
file_name	The name of the CSV file to which the data will be written.

Returns

true if the data was successfully written to the file, false otherwise.

Definition at line 112 of file CSVProcessing.cpp.

Here is the call graph for this function:



Here is the caller graph for this function:

3.2.2.3 sortBuffer()

```
std::map< string, std::vector< ZipCodeRecord > > CSVProcessing::sortBuffer ()
```

Sorts the CSV buffer and finds the zip codes (eastmost, westmost, northmost, southmost) for each state.

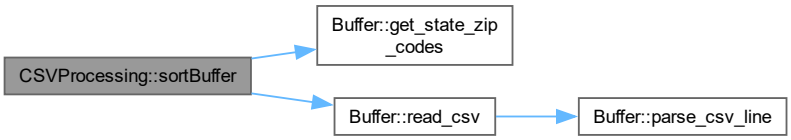
This method reads the CSV data, processes it to identify the easternmost, westernmost, northernmost, and southernmost zip codes for each state, and then stores these in a map (automatically sorts alphebetically).

Returns

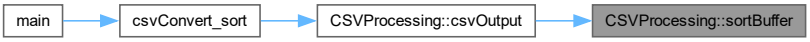
A map where the key is the state ID and the value is a vector containing the four ZipCodeRecord. The output looks as follows: [stateID] : { { east most zip, stateID, Cords }, { west most zip, stateID, Cords }, { northern most zip, stateID, Cords }, { southern most zip, stateID, Cords } }

Definition at line 32 of file CSVProcessing.cpp.

Here is the call graph for this function:



Here is the caller graph for this function:



The documentation for this class was generated from the following files:

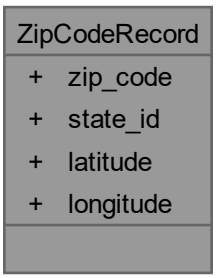
- C:/Users/mujah/OneDrive/Desktop/project/doxy/CSVProcessing.h
- C:/Users/mujah/OneDrive/Desktop/project/doxy/CSVProcessing.cpp

3.3 ZipCodeRecord Struct Reference

Structure to hold a single Zip Code record.

`#include <buffer.h>`

Collaboration diagram for ZipCodeRecord:



Public Attributes

- std::string zip_code
  The Zip Code as a string.
- std::string state_id
  The two-character state ID.
- double latitude
  Latitude of the Zip Code location.
- double longitude
  Longitude of the Zip Code location.

3.3.1 Detailed Description

Structure to hold a single Zip Code record.

This struct stores information about a Zip Code, including the Zip Code itself, the state it belongs to, and its geographical coordinates (latitude and longitude).

Definition at line 29 of file buffer.h.

3.3.2 Member Data Documentation

3.3.2.1 latitude

`double ZipCodeRecord::latitude`

Latitude of the Zip Code location.

Definition at line 32 of file buffer.h.

3.3.2.2 longitude

```
double ZipCodeRecord::longitude
```

Longitude of the Zip Code location.

Definition at line 33 of file buffer.h.

3.3.2.3 state_id

```
std::string ZipCodeRecord::state_id
```

The two-character state ID.

Definition at line 31 of file buffer.h.

3.3.2.4 zip_code

```
std::string ZipCodeRecord::zip_code
```

The Zip Code as a string.

Definition at line 30 of file buffer.h.

The documentation for this struct was generated from the following file:

- C:/Users/mujah/OneDrive/Desktop/project/doxy/buffer.h

Chapter 4

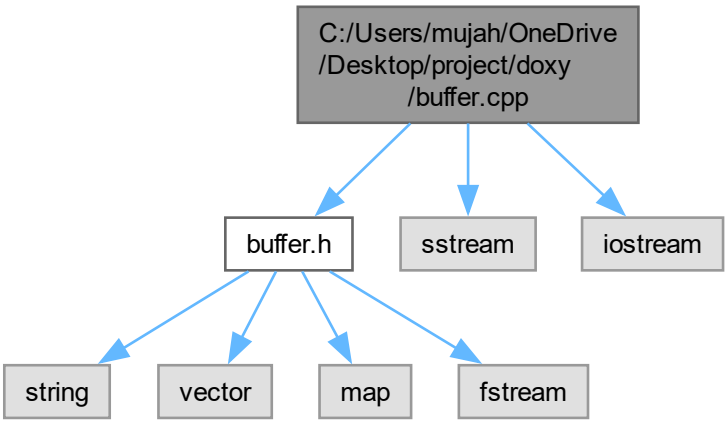
File Documentation

4.1 C:/Users/mujah/OneDrive/Desktop/project/doxy/buffer.cpp File Reference

Implementation of the Buffer class and ZipCodeRecord struct.

```
#include "buffer.h"
#include <sstream>
#include <iostream>
```

Include dependency graph for buffer.cpp:



4.1.1 Detailed Description

Implementation of the Buffer class and ZipCodeRecord struct.

Implemention of the Buffer class for handling Zip Code data read from the CSV file us_postal_codes.csv.

Author

Daniel Eze

Date

9/29/2024

Definition in file buffer.cpp.

4.2 buffer.cpp

Go to the documentation of this file.

```
00001 // Buffer.cpp
00002 #include "buffer.h"
00003 #include <sstream>
00004 #include <iostream>
00005
00028 bool Buffer::read_csv() {
00029     std::ifstream file( "us_postal_codes.csv" ); // Open the file
00030     if (!file.is_open()) {
00031         std::cerr « "Error opening file: us_postal_codes.csv" « std::endl;
00032         return false;
00033     }
00034
00035     std::string line;
00036     std::getline(file, line); // Skip the header line
00037
00038     // Read each line of the file
00039     while (std::getline(file, line)) {
00040         records.push_back(parse_csv_line(line)); // Parse and store the line
00041     }
00042
00043     file.close(); // Close the file
00044     std::cout « "CSV is now in the buffer" « std::endl;
00045     return true; // Return true if reading was successful
00046 }
00047
00048
00059 std::map<std::string, std::vector<ZipCodeRecord» Buffer::get_state_zip_codes() const {
00060     std::map<std::string, std::vector<ZipCodeRecord» state_zip_map; // Create a map to hold state
    records
00061
00062     // Loop through all records
00063     for (const auto& record : records) {
00064         state_zip_map[record.state_id].push_back(record); // Add record to the correct state
00065     }
00066
00067     return state_zip_map; // Return the grouped records
00068 }
00069
00080 ZipCodeRecord Buffer::parse_csv_line(const std::string& line) const {
00081     std::stringstream ss(line); // Use stringstream to parse the line
00082     ZipCodeRecord record; // Create a ZipCodeRecord to hold the data
00083     std::string skip;
00084     // Extract and store each field
00085     std::getline( ss, record.zip_code, ',' ); // Get Zip Code
00086     std::getline( ss, skip, ',' ); // Get Zip Code
00087     std::getline( ss, record.state_id, ',' ); // Get State ID
00088     std::getline( ss, skip, ',' ); // Get Zip Code
00089     std::string latitude_str, longitude_str;
00090     std::getline(ss, latitude_str, ','); // Get Latitude as string
00091     std::getline(ss, longitude_str, ','); // Get Longitude as string
00092     try {
00093         if ( !latitude_str.empty() ) {
00094             record.latitude = std::stod( latitude_str ); // Convert to double
00095         }
00096         else {
00097             std::cerr « "Invalid latitude value for Zip Code: " « record.zip_code « std::endl;
00098             record.latitude = 0.0; // Default value or handle appropriately
00099         }
00100
00101         if ( !longitude_str.empty() ) {
00102             record.longitude = std::stod( longitude_str ); // Convert to double
00103         }
00104         else {
```

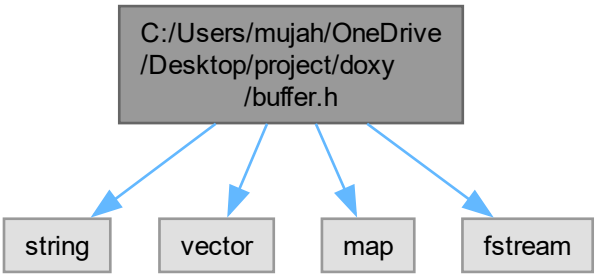
```
00105             std::cerr « "Invalid longitude value for Zip Code: " « record.zip_code « std::endl;
00106             record.longitude = 0.0; // Default value or handle appropriately
00107         }
00108     }
00109     catch ( const std::invalid_argument& e ) {
00110         std::cerr « "Error: Invalid numeric value in CSV for Zip Code: " « record.zip_code «
      record.state_id « std::endl;
00111         record.latitude = 0.0;  // Default value or handle appropriately
00112         record.longitude = 0.0; // Default value or handle appropriately
00113     }
00114     catch ( const std::out_of_range& e ) {
00115         std::cerr « "Error: Out of range numeric value in CSV for Zip Code: " « record.zip_code «
      std::endl;
00116         record.latitude = 0.0;  // Default value or handle appropriately
00117         record.longitude = 0.0; // Default value or handle appropriately
00118     }
00119
00120     return record; // Return the populated record
00121 }
```

4.3 C:/Users/mujah/OneDrive/Desktop/project/doxy/buffer.h File Reference

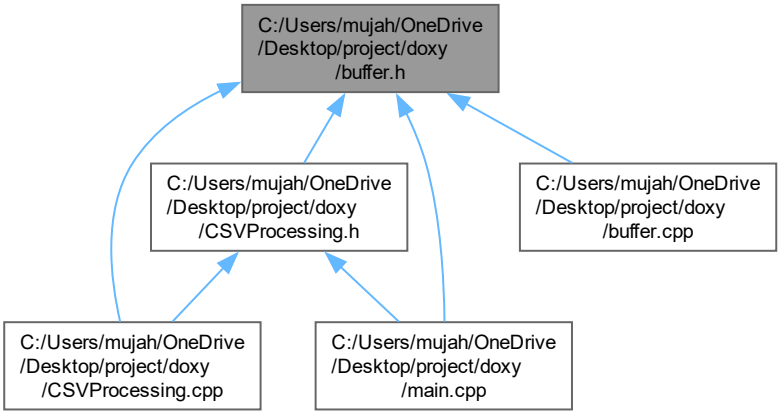
Header file for the Buffer class and ZipCodeRecord struct.

```
#include <string>
#include <vector>
#include <map>
#include <fstream>
```

Include dependency graph for buffer.h:



This graph shows which files directly or indirectly include this file:



Classes

- struct ZipCodeRecord

  Structure to hold a single Zip Code record.
- class Buffer

  Buffer class for reading and storing Zip Code records from a CSV file.

4.3.1 Detailed Description

Header file for the Buffer class and ZipCodeRecord struct.

This file contains the declarations for the Buffer class, which handles reading and storing Zip Code records from a CSV file, and the ZipCodeRecord struct.

Author

Thomas Hoerger

Date

9/27/2024

Definition in file buffer.h.

4.4 buffer.h

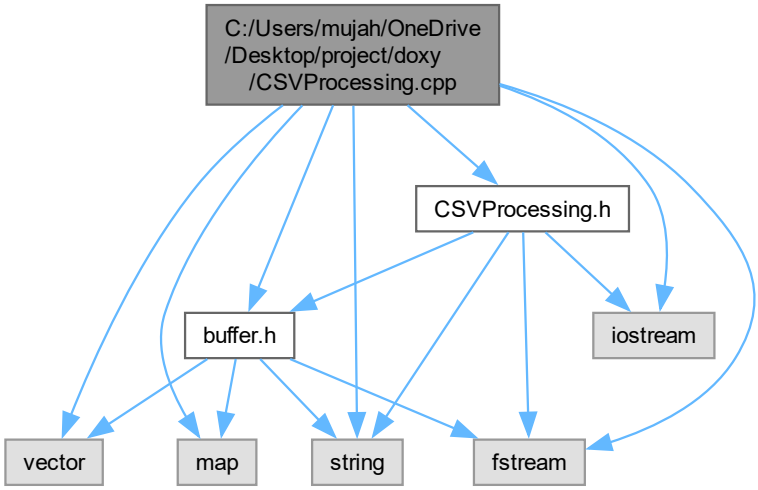
Go to the documentation of this file.

```
00001 // Buffer.h
00002 #ifndef BUFFER_H
00003 #define BUFFER_H
00004
00005 #include <string>
00006 #include <vector>
00007 #include <map>
00008 #include <fstream>
00009
00029 struct ZipCodeRecord {
00030     std::string zip_code;
00031     std::string state_id;
00032     double latitude;
00033     double longitude;
00034 };
00035
00043 class Buffer {
00044 public:
00054     bool read_csv();
00055
00064     std::map<std::string, std::vector<ZipCodeRecord» get_state_zip_codes() const;
00065
00066 private:
00067     std::vector<ZipCodeRecord> records;
00068
00079     ZipCodeRecord parse_csv_line(const std::string& line) const;
00080 };
00081
00082 #endif // BUFFER_H
```

4.5 C:/Users/mujah/OneDrive/Desktop/project/doxy/CSVProcessing.cpp File Reference

```
#include "buffer.h"
#include "CSVProcessing.h"
#include <iostream>
#include <fstream>
#include <string>
#include <map>
#include <vector>
```

Include dependency graph for CSVProcessing.cpp:



4.6 CSVProcessing.cpp

Go to the documentation of this file.

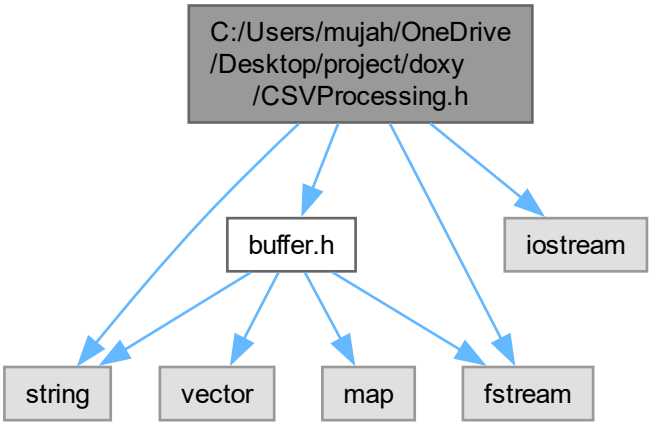
```
00001 #include "buffer.h"
00002 #include "CSVProcessing.h"
00003 #include <iostream>
00004 #include <fstream>
00005 #include <string>
00006 #include <map>
00007 #include <vector>
00008 //using namespace std;
00009
00010
00011 // void CSVProcessing::printZipCodeRecord( const ZipCodeRecord& record ) {
00012 //     std::cout « "Zip Code: " « record.zip_code
00013 //         « ", State ID: " « record.state_id
00014 //         « ", Latitude: " « record.latitude
00015 //         « ", Longitude: " « record.longitude « std::endl;
00016 // }
00032 std::map<string, std::vector<ZipCodeRecord» CSVProcessing::sortBuffer() {
00033     float eastMost, westMost, northMost, southMost;
00034     Buffer CSVBuffer;
00035     CSVBuffer.read_csv( );
00036     std::map<string, std::vector<ZipCodeRecord» state_zip_map = CSVBuffer.get_state_zip_codes();
00037     std::map<string, std::vector<ZipCodeRecord» sorted_directions;
00038     for ( auto& state : state_zip_map ) {
00039         const std::string& stateID = state.first;
00040         const std::vector<ZipCodeRecord>& stateInfo = state.second;
00041         // intial loading of directions
00042         ZipCodeRecord easternmost = stateInfo[ 0 ];
00043         ZipCodeRecord westernmost = stateInfo[ 0 ];
00044         ZipCodeRecord northernmost = stateInfo[ 0 ];
00045         ZipCodeRecord southernmost = stateInfo[ 0 ];
00046         // checks if the current records zip is one of the maxed directions
00047         for ( const auto& record : stateInfo ) {
00048             if ( record.longitude < easternmost.longitude ) {
00049                 easternmost = record;
00050             }
00051             if ( record.longitude > westernmost.longitude ) {
00052                 westernmost = record;
00053             }
00054             if ( record.latitude > northernmost.latitude ) {
00055                 northernmost = record;
00056             }
00057             if ( record.latitude < southernmost.latitude ) {
```

```
00058                 southernmost = record;
00059             }
00060         }
00061         sorted_directions[ stateID ] = { easternmost, westernmost, northernmost, southernmost };
00062         // std::cout « "State: " « stateID « std::endl;
00063         // std::cout « "  Easternmost: ";
00064         // printZipCodeRecord( easternmost );
00065         // std::cout « "  Westernmost: ";
00066         // printZipCodeRecord( westernmost );
00067         // std::cout « "  Northernmost: ";
00068         // printZipCodeRecord( northernmost );
00069         // std::cout « "  Southernmost: ";
00070         // printZipCodeRecord( southernmost );
00071         // std::cout « std::endl;  // Add an extra line for readability
00072     }
00073     // sorted_directions looks like this
00074     // [stateID] : {
00075     //     { east most zip, stateID, directions },
00076     //     { west most zip, stateID, directions },
00077     //     { northern most zip, stateID, directions },
00078     //     { southern most zip, stateID, directions }
00079     // }
00080
00081     return sorted_directions;
00082 }
00083
00092 void CSVProcessing::addHeader(std::string& file_name) {
00093     std::ofstream file(file_name);
00094     if (file.is_open()) {
00095         file « "State,Easternmost,Westernmost,Northernmost,Southernmost\n";
00096         file.close();
00097         std::cout « "Header added successfully to " « file_name « std::endl;
00098     } else {
00099         std::cerr « "Unable to open file: " « file_name « std::endl;
00100     }
00101 }
00112 bool CSVProcessing::csvOutput(std::string& file_name) {
00113     std::map<std::string, std::vector<ZipCodeRecord» sorted_data = sortBuffer();
00114     std::ofstream file(file_name, std::ios::app);  // Open in append mode
00115
00116     if (!file.is_open()) {
00117         std::cerr « "Unable to open file: " « file_name « std::endl;
00118         return false;
00119     }
00120     for (const auto& [state, records] : sorted_data) {
00121         if (records.size() == 4) {  // Ensure we have all 4 directional records
00122             file « state « ","
00123                  « records[0].zip_code « ","  // Easternmost
00124                  « records[1].zip_code « ","  // Westernmost
00125                  « records[2].zip_code « ","  // Northernmost
00126                  « records[3].zip_code « "\n";  // Southernmost
00127         }
00128     }
00129     file.close();
00130     std::cout « "Data successfully written to " « file_name « std::endl;
00131     return true;
00132 }
```

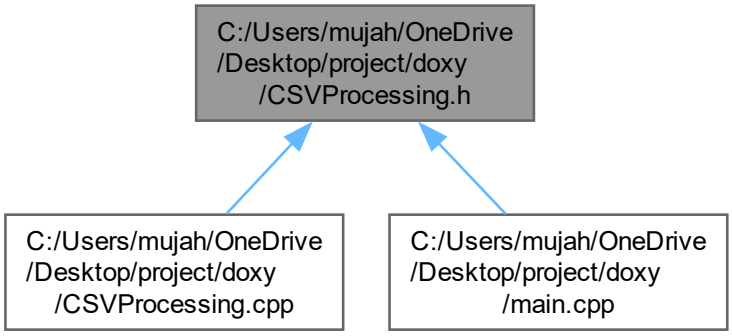
4.7 C:/Users/mujah/OneDrive/Desktop/project/doxy/CSVProcessing.h File Reference

```
#include "buffer.h"
#include <iostream>
#include <fstream>
#include <string>
```

Include dependency graph for CSVProcessing.h:



This graph shows which files directly or indirectly include this file:



Classes

- class CSVProcessing

4.8 CSVProcessing.h

Go to the documentation of this file.

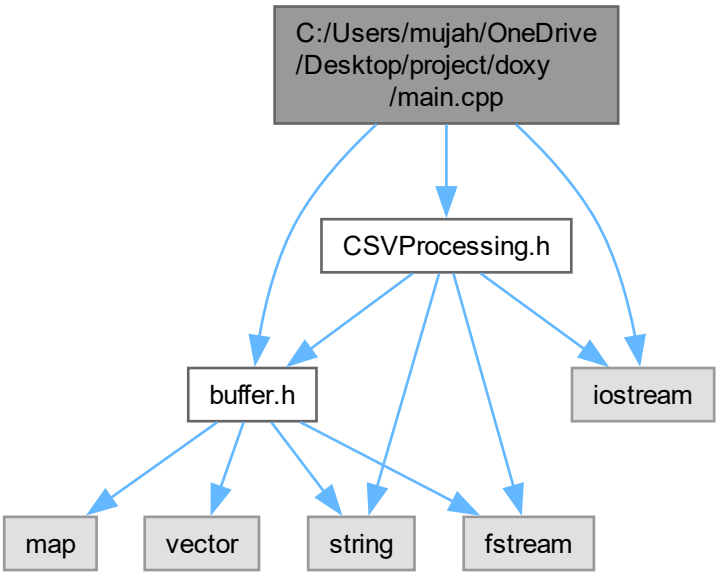
```
00001 #ifndef CSVProcessing_H
00002 #define CSVProcessing_H
00003
```

```
00004 #include "buffer.h"
00005 #include <iostream>
00006 #include <fstream>
00007 #include <string>
00008
00009 using namespace std;
00010
00011 class CSVProcessing {
00012 public:
00028     map<string, vector<ZipCodeRecord» sortBuffer(); // sort by state with the hashmap but how once it
      is sorted we can do the direction farthest zip
00029     // we could also set up a const variable that will have the state ids based on their index/hasmap
      key and with that we can instantlly find where the zip should go
00030     //void printZipCodeRecord( const ZipCodeRecord& record ); for testing purposes
00039     void addHeader( string& file_name ); // state id, Easternmost (least longitude), Westernmost,
      Northernmost (greatest latitude), and Southernmost Zip Code
00050     bool csvOutput( string& file_name ); // fill from the sortered buffer? either output as we go from
      the buffer or create an array or vector to put all the sorting and then output to the csv
00051 };
00052
00053 #endif
```

4.9 C:/Users/mujah/OneDrive/Desktop/project/doxy/main.cpp File Reference

```
#include "CSVProcessing.h"
#include "buffer.h"
#include <iostream>
```

Include dependency graph for main.cpp:



Functions

- void csvConvert_sort (CSVProcessing origin, string file)
- int main ()

4.9.1 Function Documentation

4.9.1.1 csvConvert_sort()

```
void csvConvert_sort (
            CSVProcessing origin,
            string file)
```

Definition at line 6 of file main.cpp.

Here is the call graph for this function:

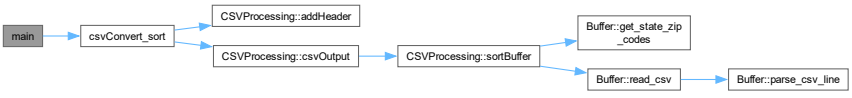
Here is the caller graph for this function:

4.9.1.2 main()

```
int main ()
```

Definition at line 24 of file main.cpp.

Here is the call graph for this function:



4.10 main.cpp

Go to the documentation of this file.

```
00001 #include "CSVProcessing.h"
00002 #include "buffer.h"
00003 #include <iostream>
00004 using namespace std;
00005
00006 void csvConvert_sort(CSVProcessing origin, string file)
00007 {
00008     cout « "Generating header row." « endl;
00009     origin.addHeader(file);
00010     cout « "Checking for errors" « endl « "Errors: ";
00011
00012     if (origin.csvOutput(file)) {
00013         cout « "No" « endl « "File made!";
00014     } else {
00015         cout « "Yes" « endl « "File not made.";
00016     }
00017
00018 };
00019
00020
00021
00022
00023
00024 int main() {
00025     CSVProcessing csvProcessor;
00026
00027     std::string file_name = "output.csv";
00028     csvConvert_sort(csvProcessor, file_name);
00029     //csvProcessor.sortBuffer();
00030
00031
00032
00033
00034
00035     return 0;
00036 }
```

Index

addHeader
CSVProcessing, 9

Buffer, 5
get_state_zip_codes, 6
parse_csv_line, 6
read_csv, 7
records, 8

C:/Users/mujah/OneDrive/Desktop/project/doxy/buffer.cpp, 15, 16
C:/Users/mujah/OneDrive/Desktop/project/doxy/buffer.h, 17, 19
C:/Users/mujah/OneDrive/Desktop/project/doxy/CSVProcessing.cpp, 19, 20
C:/Users/mujah/OneDrive/Desktop/project/doxy/CSVProcessing.h, 21, 22
C:/Users/mujah/OneDrive/Desktop/project/doxy/main.cpp, 23, 25
csvConvert_sort
main.cpp, 24
csvOutput
CSVProcessing, 10
CSVProcessing, 9
addHeader, 9
csvOutput, 10
sortBuffer, 10

get_state_zip_codes
Buffer, 6

latitude
ZipCodeRecord, 12
longitude
ZipCodeRecord, 12

main
main.cpp, 24
main.cpp
csvConvert_sort, 24
main, 24

parse_csv_line
Buffer, 6

read_csv
Buffer, 7
records
Buffer, 8

sortBuffer
CSVProcessing, 10
state_id
ZipCodeRecord, 13

zip_code
ZipCodeRecord, 13
ZipCodeRecord, 12
latitude, 12
longitude, 12
state_id, 13
zip_code, 13