Zip Code Project

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 Constructor & Destructor Documentation . . . 6

3.1.2.1 Buffer() . . . 6

3.1.3 Member Function Documentation . . . 7

3.1.3.1 get_zipcodes() . . . 7

3.1.3.2 populate_zipcodes() . . . 7

3.1.3.3 tokenize_line() . . . 8

3.1.4 Member Data Documentation . . . 8

3.1.4.1 reader . . . 8

3.1.4.2 zipcodes . . . 9

3.2 FileReader Class Reference . . . 9

3.2.1 Detailed Description . . . 10

3.2.2 Constructor & Destructor Documentation . . . 10

3.2.2.1 FileReader() . . . 10

3.2.3 Member Function Documentation . . . 11

3.2.3.1 get_lines() . . . 11

3.2.3.2 populate_lines() . . . 11

3.2.4 Friends And Related Symbol Documentation . . . 11

3.2.4.1 operator<< . . . 11

3.2.5 Member Data Documentation . . . 12

3.2.5.1 file . . . 12

3.2.5.2 lines . . . 12

3.3 ZipCodeData Struct Reference . . . 13

3.3.1 Detailed Description . . . 14

3.3.2 Constructor & Destructor Documentation . . . 14

3.3.2.1 ZipCodeData() . . . 14

3.3.3 Friends And Related Symbol Documentation . . . 14

3.3.3.1 operator<< . . . 14

3.3.4 Member Data Documentation . . . 15

3.3.4.1 county . . . 15

3.3.4.2 latitude . . . 15

3.3.4.3 longitude . . . 15

3.3.4.4 place_name . . . 15

3.3.4.5 state . . . 15

3.3.4.6 zip_code . . . 15

4 File Documentation **17**

4.1 buffer.cpp File Reference 17
4.2 buffer.cpp 17
4.3 buffer.h File Reference 18
4.4 buffer.h 19
4.5 filereader.cpp File Reference 20
4.5.1 Function Documentation 20
4.5.1.1 operator<<() 20
4.6 filereader.cpp 21
4.7 filereader.h File Reference 21
4.8 filereader.h 22
4.9 main.cpp File Reference 22
4.9.1 Function Documentation 23
4.9.1.1 main() 23
4.9.1.2 parse_data() 24
4.9.1.3 print_data() 25
4.10 main.cpp 25
4.11 zipcode.cpp File Reference 26
4.11.1 Function Documentation 27
4.11.1.1 operator<<() 27
4.12 zipcode.cpp 28
4.13 zipcode.h File Reference 28
4.14 zipcode.h 29

Chapter 1

Class Index

1.1 Class List

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

Buffer
: Reads zip code data from a file and processes it into a usable format 5

FileReader
: Handles reading lines from a file 9

ZipCodeData
: The ZipCodeData struct holds data for a single zip code, including its coordinates and place information 13

Chapter 2

File Index

2.1 File List

Here is a list of all files with brief descriptions:

buffer.cpp 17
buffer.h 18
filereader.cpp 20
filereader.h 21
main.cpp 22
zipcode.cpp 26
zipcode.h 28

Chapter 3

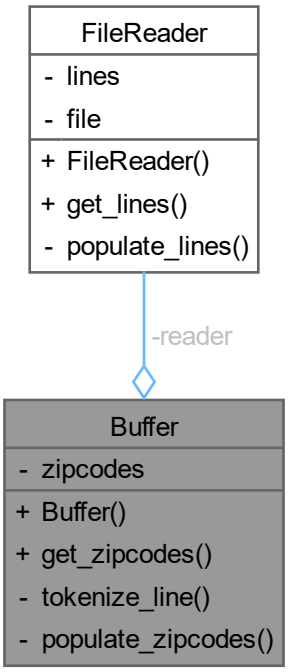
Class Documentation

3.1 Buffer Class Reference

The Buffer class reads zip code data from a file and processes it into a usable format.

`#include <buffer.h>`

Collaboration diagram for Buffer:



Public Member Functions

- Buffer (const std::string &)

  Constructs a Buffer object that reads data from the specified file.
- std::vector< ZipCodeData > get_zipcodes ()

  Returns the vector containing all zip code data.

Private Member Functions

- std::tuple< int, std::string, std::string, std::string, float, float > tokenize_line (const std::string &)

  Tokenizes a line of CSV data into individual zip code components.
- void populate_zipcodes ()

  Populates the zipcodes vector with data parsed from the file.

Private Attributes

- std::vector< ZipCodeData > zipcodes

  Vector to store all zip code data.
- FileReader reader

  FileReader object to handle file operations.

3.1.1 Detailed Description

The Buffer class reads zip code data from a file and processes it into a usable format.

Definition at line 18 of file buffer.h.

3.1.2 Constructor & Destructor Documentation

3.1.2.1 Buffer()

```
Buffer::Buffer (
            const std::string & file)
```

Constructs a Buffer object that reads data from the specified file.

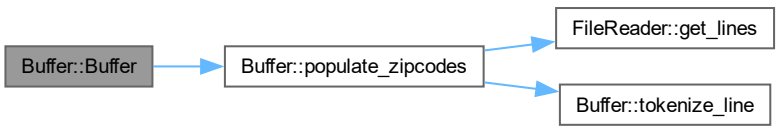
Parameters

file	The path to the input CSV file.

Definition at line 8 of file buffer.cpp.

References populate_zipcodes().

Here is the call graph for this function:



3.1.3 Member Function Documentation

3.1.3.1 get_zipcodes()

```
std::vector< ZipCodeData > Buffer::get_zipcodes ()
```

Returns the vector containing all zip code data.

Returns

A vector of ZipCodeData objects.

Definition at line 54 of file buffer.cpp.

References zipcodes.

Referenced by main().

Here is the caller graph for this function:

3.1.3.2 populate_zipcodes()

```
void Buffer::populate_zipcodes ()  [private]
```

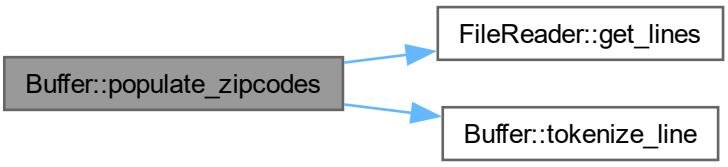
Populates the zipcodes vector with data parsed from the file.

Definition at line 44 of file buffer.cpp.

References FileReader::get_lines(), reader, tokenize_line(), and zipcodes.

Referenced by Buffer().

Here is the call graph for this function:



Here is the caller graph for this function:

3.1.3.3 tokenize_line()

```
std::tuple< int, std::string, std::string, std::string, float, float > Buffer::tokenize_line (
            const std::string & line)  [private]
```

Tokenizes a line of CSV data into individual zip code components.

Parameters

line	The input string representing a single line of CSV data.

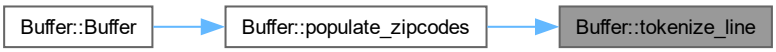
Returns

A tuple containing the zip code, place name, state, county, latitude, and longitude.

Definition at line 13 of file buffer.cpp.

Referenced by populate_zipcodes().

Here is the caller graph for this function:



3.1.4 Member Data Documentation

3.1.4.1 reader

```
FileReader Buffer::reader  [private]
```

FileReader object to handle file operations.

Definition at line 20 of file buffer.h.

Referenced by populate_zipcodes().

3.1.4.2 zipcodes

```
std::vector<ZipCodeData> Buffer::zipcodes  [private]
```

Vector to store all zip code data.

Definition at line 19 of file buffer.h.

Referenced by get_zipcodes(), and populate_zipcodes().

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

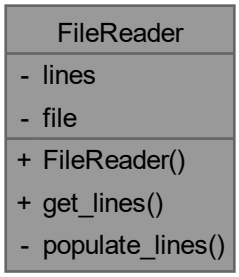
- buffer.h
- buffer.cpp

3.2 FileReader Class Reference

The FileReader class handles reading lines from a file.

```
#include <filereader.h>
```

Collaboration diagram for FileReader:



Public Member Functions

- FileReader (const std::string &)

  Constructs a FileReader object and opens the specified file.
- std::vector< std::string > get_lines ()

  Returns the lines read from the file.

Private Member Functions

- void populate_lines ()

  Populates the lines vector by reading each line from the file.

Private Attributes

- std::vector< std::string > lines

  Vector to store lines read from the file.
- std::ifstream file

  Input file stream.

Friends

- std::ostream & operator<< (std::ostream &outputstream, const FileReader &reader)

  Overloads the << operator to print all lines of the file to an output stream.

3.2.1 Detailed Description

The FileReader class handles reading lines from a file.

Definition at line 14 of file filereader.h.

3.2.2 Constructor & Destructor Documentation

3.2.2.1 FileReader()

```
FileReader::FileReader (
            const std::string & input_file)
```

Constructs a FileReader object and opens the specified file.

Parameters

input_file	The path to the input file.

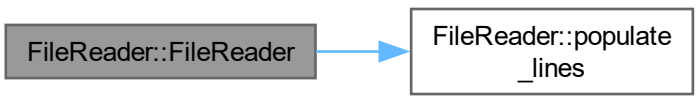
Exceptions

std::runtime_error	If the file cannot be opened.

Definition at line 6 of file filereader.cpp.

References file, and populate_lines().

Here is the call graph for this function:



3.2.3 Member Function Documentation

3.2.3.1 get_lines()

```
std::vector< std::string > FileReader::get_lines ()
```

Returns the lines read from the file.

Returns

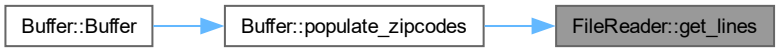
A vector of strings representing each line in the file.

Definition at line 25 of file filereader.cpp.

References lines.

Referenced by Buffer::populate_zipcodes().

Here is the caller graph for this function:



3.2.3.2 populate_lines()

```
void FileReader::populate_lines ()  [private]
```

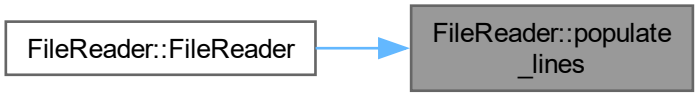
Populates the lines vector by reading each line from the file.

Definition at line 17 of file filereader.cpp.

References file, and lines.

Referenced by FileReader().

Here is the caller graph for this function:



3.2.4 Friends And Related Symbol Documentation

3.2.4.1 operator<<

```
std::ostream & operator<< (
            std::ostream & outputstream,
            const FileReader & reader)  [friend]
```

Overloads the << operator to print all lines of the file to an output stream.

Parameters

outputstream	The output stream.
reader	The FileReader object containing the lines to print.

Returns

The output stream after the lines are written.

Definition at line 29 of file filereader.cpp.

3.2.5 Member Data Documentation

3.2.5.1 file

```
std::ifstream FileReader::file  [private]
```

Input file stream.

Definition at line 17 of file filereader.h.

Referenced by FileReader(), and populate_lines().

3.2.5.2 lines

```
std::vector<std::string> FileReader::lines  [private]
```

Vector to store lines read from the file.

Definition at line 16 of file filereader.h.

Referenced by get_lines(), and populate_lines().

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

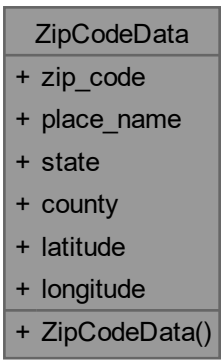
- filereader.h
- filereader.cpp

3.3 ZipCodeData Struct Reference

The ZipCodeData struct holds data for a single zip code, including its coordinates and place information.

`#include <zipcode.h>`

Collaboration diagram for ZipCodeData:



Public Member Functions

- ZipCodeData (std::tuple< int, std::string, std::string, std::string, float, float >)

  Constructs a ZipCodeData object from a tuple containing zip code details.

Public Attributes

- int zip_code

  The zip code.
- std::string place_name

  The name of the place corresponding to the zip code.
- std::string state

  The state (two-character abbreviation).
- std::string county

  The county of the place.
- float latitude

  The latitude coordinate.
- float longitude

  The longitude coordinate.

Friends

- std::ostream & operator<< (std::ostream &outputstream, const ZipCodeData &zipcode)

  Overloads the << operator to print the ZipCodeData to an output stream.

3.3.1 Detailed Description

The ZipCodeData struct holds data for a single zip code, including its coordinates and place information.

Definition at line 13 of file zipcode.h.

3.3.2 Constructor & Destructor Documentation

3.3.2.1 ZipCodeData()

```
ZipCodeData::ZipCodeData (
            std::tuple< int, std::string, std::string, std::string, float, float > tuple)
```

Constructs a ZipCodeData object from a tuple containing zip code details.

Parameters

tuple	A tuple containing zip code, place name, state, county, latitude, and longitude.

Definition at line 7 of file zipcode.cpp.

References county, latitude, longitude, place_name, state, and zip_code.

3.3.3 Friends And Related Symbol Documentation

3.3.3.1 operator<<

```
std::ostream & operator<< (
            std::ostream & outputstream,
            const ZipCodeData & zipcode)  [friend]
```

Overloads the << operator to print the ZipCodeData to an output stream.

Parameters

outputstream	The output stream.
zipcode	The ZipCodeData object to be printed.

Returns

The output stream after the zip code data is written.

Definition at line 12 of file zipcode.cpp.

3.3.4 Member Data Documentation

3.3.4.1 county

```
std::string ZipCodeData::county
```

The county of the place.

Definition at line 18 of file zipcode.h.

Referenced by ZipCodeData().

3.3.4.2 latitude

```
float ZipCodeData::latitude
```

The latitude coordinate.

Definition at line 19 of file zipcode.h.

Referenced by ZipCodeData().

3.3.4.3 longitude

```
float ZipCodeData::longitude
```

The longitude coordinate.

Definition at line 20 of file zipcode.h.

Referenced by ZipCodeData().

3.3.4.4 place_name

```
std::string ZipCodeData::place_name
```

The name of the place corresponding to the zip code.

Definition at line 16 of file zipcode.h.

Referenced by ZipCodeData().

3.3.4.5 state

```
std::string ZipCodeData::state
```

The state (two-character abbreviation).

Definition at line 17 of file zipcode.h.

Referenced by ZipCodeData().

3.3.4.6 zip_code

```
int ZipCodeData::zip_code
```

The zip code.

Definition at line 15 of file zipcode.h.

Referenced by ZipCodeData().

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

- zipcode.h
- zipcode.cpp

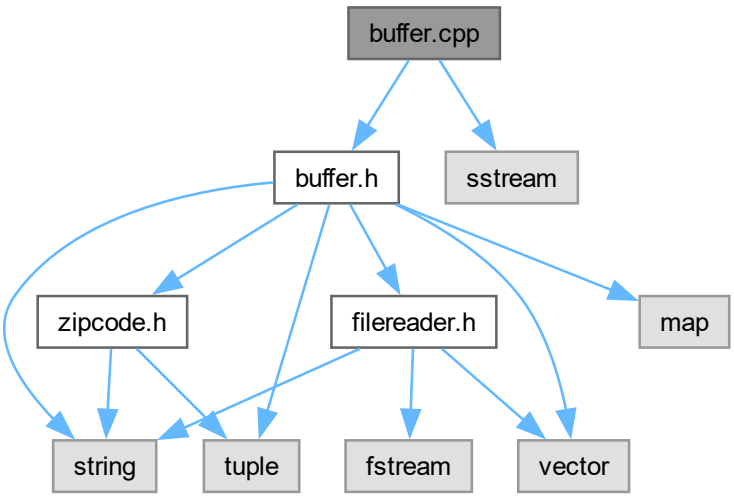
Chapter 4

File Documentation

4.1 buffer.cpp File Reference

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

Include dependency graph for buffer.cpp:



4.2 buffer.cpp

Go to the documentation of this file.

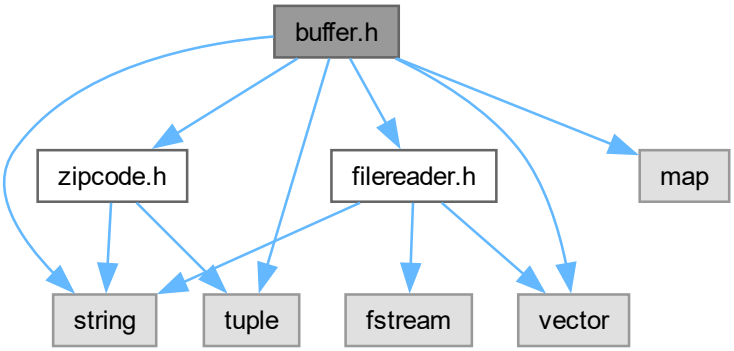
```
00001
00004 #include "buffer.h"
00005
00006 #include <sstream>
00007
00008 Buffer::Buffer(const std::string& file) : reader(FileReader(file))
```

```
00009 {
00010     this->populate_zipcodes();
00011 }
00012
00013 std::tuple<int, std::string, std::string, std::string, float, float> Buffer::tokenize_line(const
      std::string& line)
00014 {
00015     int zip_code;
00016     std::string place_name;
00017     std::string state;
00018     std::string county;
00019     float latitude;
00020     float longitude;
00021
00022     std::stringstream stream(line);
00023     std::string token;
00024
00025     std::getline(stream, token, ',');
00026     zip_code = std::stoi(token);
00027
00028     std::getline(stream, place_name, ',');
00029
00030     std::getline(stream, state, ',');
00031
00032     std::getline(stream, county, ',');
00033
00034     std::getline(stream, token, ',');
00035     latitude = std::stof(token);
00036
00037     std::getline(stream, token, ',');
00038     longitude = std::stof(token);
00039
00040
00041     return std::make_tuple(zip_code, place_name, state, county, latitude, longitude);
00042 }
00043
00044 void Buffer::populate_zipcodes()
00045 {
00046     std::vector<std::string> data = reader.get_lines();
00047
00048     for (int i = 1; i < data.size(); i++) {
00049         std::tuple<int, std::string, std::string, std::string, float, float> zipcode_data =
      tokenize_line(data[i]);
00050         this->zipcodes.push_back(ZipCodeData(zipcode_data));
00051     }
00052 }
00053
00054 std::vector<ZipCodeData> Buffer::get_zipcodes() {
00055     return this->zipcodes;
00056 }
```

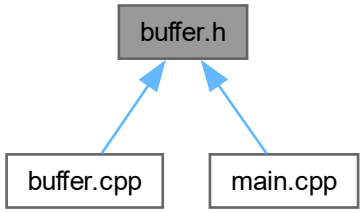
4.3 buffer.h File Reference

```
#include <string>
#include <map>
#include <vector>
#include <tuple>
#include "zipcode.h"
#include "filereader.h"
```

Include dependency graph for buffer.h:



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



Classes

- class Buffer

  The Buffer class reads zip code data from a file and processes it into a usable format.

4.4 buffer.h

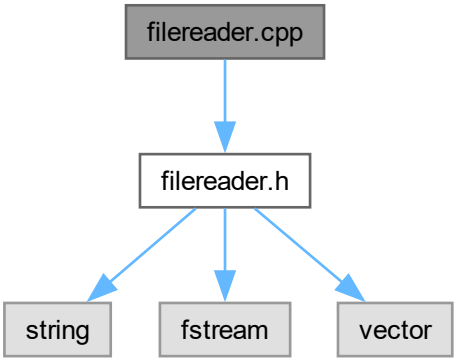
Go to the documentation of this file.

```
00001
00004 #ifndef BUFFER_H
00005 #define BUFFER_H
00006
00007 #include <string>
00008 #include <map>
00009 #include <vector>
00010 #include <tuple>
00011
00012 #include "zipcode.h"
00013 #include "filereader.h"
```

```
00014
00018 class Buffer {
00019     std::vector<ZipCodeData> zipcodes;
00020     FileReader reader;
00021
00028     std::tuple<int, std::string, std::string, std::string, float, float> tokenize_line(const
      std::string&);
00029
00033     void populate_zipcodes();
00034
00035 public:
00041     Buffer(const std::string&);
00042
00048     std::vector<ZipCodeData> get_zipcodes();
00049 };
00050
00051 #endif // BUFFER_H
```

4.5 filereader.cpp File Reference

`#include "filereader.h"`
Include dependency graph for filereader.cpp:



Functions

- std::ostream & operator<< (std::ostream &outputstream, const FileReader &reader)

4.5.1 Function Documentation

4.5.1.1 operator<<()

```
std::ostream & operator<< (
            std::ostream & outputstream,
            const FileReader & reader)
```

Parameters

outputstream	The output stream.
reader	The FileReader object containing the lines to print.

Returns

The output stream after the lines are written.

Definition at line 29 of file filereader.cpp.

4.6 filereader.cpp

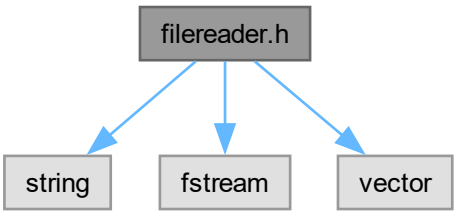
Go to the documentation of this file.

```
00001
00004 #include "filereader.h"
00005
00006 FileReader::FileReader(const std::string& input_file)
00007 {
00008     this->file.open(input_file.c_str());
00009
00010     if (!this->file.is_open()) {
00011         throw std::runtime_error("Failed to open the file");
00012     }
00013
00014     this->populate_lines();
00015 }
00016
00017 void FileReader::populate_lines()
00018 {
00019     std::string line;
00020     while (std::getline(this->file, line)) {
00021         this->lines.push_back(line);
00022     }
00023 }
00024
00025 std::vector<std::string> FileReader::get_lines() {
00026     return lines;
00027 }
00028
00029 std::ostream& operator«(std::ostream& outputstream, const FileReader& reader) {
00030     for (int i = 0; i < reader.lines.size(); i++) {
00031        outputstream « reader.lines[i] « std::endl;
00032     }
00033
00034     return outputstream;
00035 }
```

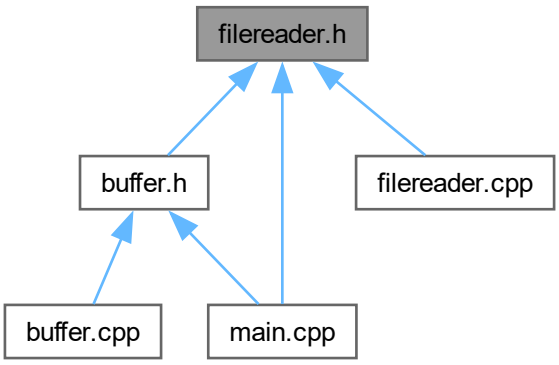
4.7 filereader.h File Reference

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

Include dependency graph for filereader.h:



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



Classes

- class FileReader

  The FileReader class handles reading lines from a file.

4.8 filereader.h

Go to the documentation of this file.

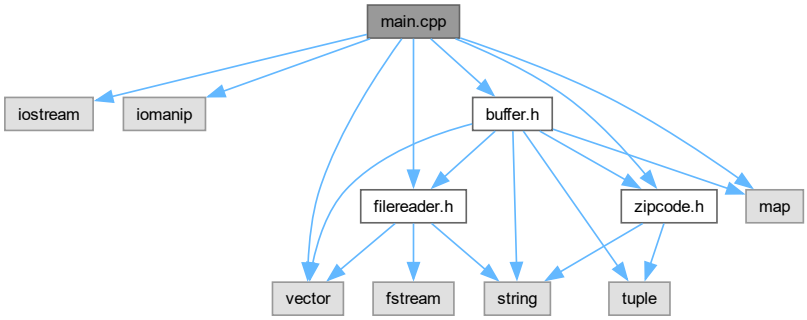
```
00001
00004 #ifndef FILE_READER_H
00005 #define FILE_READER_H
00006
00007 #include <string>
00008 #include <fstream>
00009 #include <vector>
00010
00014 class FileReader
00015 {
00016     std::vector<std::string> lines;
00017     std::ifstream file;
00018
00022     void populate_lines();
00023
00024 public:
00031     FileReader(const std::string&);
00032
00038     std::vector<std::string> get_lines();
00039
00047     friend std::ostream& operator<<(std::ostream&, const FileReader&);
00048 };
00049
00050 #endif // FILE_READER_H
```

4.9 main.cpp File Reference

```
#include <iostream>
#include <iomanip>
#include <vector>
```

```
#include <map>
#include "zipcode.h"
#include "filereader.h"
#include "buffer.h"
```

Include dependency graph for main.cpp:



Functions

- std::map< std::string, std::tuple< int, int, int, int > > parse_data (const std::vector< ZipCodeData > &records)

  Parses a vector of ZipCodeData records to find the Easternmost, Westernmost, Northernmost, and Southernmost zip codes for each state.
- void print_data (const std::map< std::string, std::tuple< int, int, int, int > > &data)

  Prints the parsed data of each state with the respective Easternmost, Westernmost, Northernmost, and Southernmost zip codes.
- int main ()

  Main function that reads zip code data from a CSV file, parses it, and prints the extreme zip codes (East, West, North, South) for each state.

4.9.1 Function Documentation

4.9.1.1 main()

```
int main ()
```

Main function that reads zip code data from a CSV file, parses it, and prints the extreme zip codes (East, West, North, South) for each state.

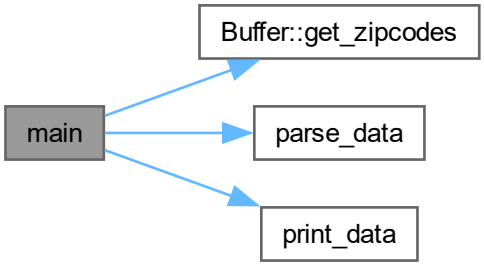
Returns

int Returns 0 if the program executes successfully.

Definition at line 104 of file main.cpp.

References Buffer::get_zipcodes(), parse_data(), and print_data().

Here is the call graph for this function:



4.9.1.2 parse_data()

```
std::map< std::string, std::tuple< int, int, int, int > > parse_data (
            const std::vector< ZipCodeData > & records)
```

Parses a vector of ZipCodeData records to find the Easternmost, Westernmost, Northernmost, and Southernmost zip codes for each state.

Parameters

records	A vector of ZipCodeData containing zip code information including latitude and longitude.

Returns

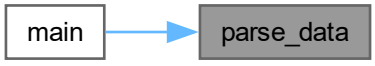
A map where the key is the state (two-letter string), and the value is a tuple of integers representing:

- Easternmost zip code (least longitude)
- Westernmost zip code (most longitude)
- Northernmost zip code (most latitude)
- Southernmost zip code (least latitude)

Definition at line 23 of file main.cpp.

Referenced by main().

Here is the caller graph for this function:



4.9.1.3 print_data()

```
void print_data (
            const std::map< std::string, std::tuple< int, int, int, int > > & data)
```

Prints the parsed data of each state with the respective Easternmost, Westernmost, Northernmost, and Southernmost zip codes.

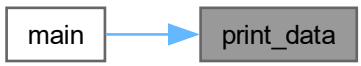
Parameters

data	A map where the key is the state (string) and the value is a tuple of zip codes (int) for East, West, North, and South extremes.

Definition at line 90 of file main.cpp.

Referenced by main().

Here is the caller graph for this function:



4.10 main.cpp

Go to the documentation of this file.

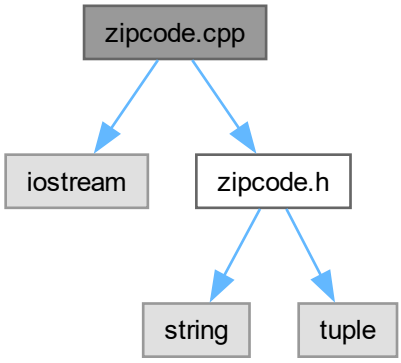
```
00001
00004 #include <iostream>
00005 #include <iomanip>
00006 #include <vector>
00007 #include <map>
00008
00009 #include "zipcode.h"
00010 #include "filereader.h"
00011 #include "buffer.h"
00012
00023 std::map<std::string, std::tuple<int, int, int, int» parse_data(const std::vector<ZipCodeData>
      &records)
00024 {
00025     // create map key is state, tuple is Easternmost (least longitude), Westernmost (most longitude),
      Northernmost (most latitude), Southernmost (least latitude)
00026     std::map<std::string, std::tuple<int, int, int, int» data;
00027     std::map<std::string, std::tuple<float, float, float, float» data_locations; // key is state,
      tuple for coordinates
00028
00029     for (const ZipCodeData &entry : records)
00030     {
00031         if (data.find(entry.state) == data.end())
00032         {
00033             // if state doesn't exist in the map, add it
00034             data[entry.state] = std::make_tuple(entry.zip_code, entry.zip_code, entry.zip_code,
      entry.zip_code);             // store the zip code of the first entry
00035             data_locations[entry.state] = std::make_tuple(entry.longitude, entry.longitude,
      entry.latitude, entry.latitude); // store the coordinates of the first entry
00036         }
00037         else
00038         {
00039             // grab the current tuple to compare with the new entry
00040             std::tuple<float, float, float, float> current = data_locations[entry.state];
```

```
00041             float east = std::get<0>(current);
00042             float west = std::get<1>(current);
00043             float north = std::get<2>(current);
00044             float south = std::get<3>(current);
00045 
00046             int east_zip_code = std::get<0>(data[entry.state]);
00047             int west_zip_code = std::get<1>(data[entry.state]);
00048             int north_zip_code = std::get<2>(data[entry.state]);
00049             int south_zip_code = std::get<3>(data[entry.state]);
00050 
00051             int zip_code = entry.zip_code;
00052 
00053             // check if the new entry is more eastern than the current easternmost
00054             if (entry.longitude < east)
00055             {
00056                 east = entry.longitude;
00057                 east_zip_code = zip_code;
00058             }
00059             else if (entry.longitude > west)
00060             {
00061                 west = entry.longitude;
00062                 west_zip_code = zip_code;
00063             }
00064 
00065             // check if the new entry is more northern or southern than the current northernmost or
      southernmost (not else if since it could be both)
00066             if (entry.latitude > north)
00067             {
00068                 north = entry.latitude;
00069                 north_zip_code = zip_code;
00070             }
00071             else if (entry.latitude < south)
00072             {
00073                 south = entry.latitude;
00074                 south_zip_code = zip_code;
00075             }
00076 
00077             data[entry.state] = std::make_tuple(east_zip_code, west_zip_code, north_zip_code,
      south_zip_code);
00078             data_locations[entry.state] = std::make_tuple(east, west, north, south);
00079         }
00080     }
00081 
00082     return data;
00083 }
00084 
00090 void print_data(const std::map<std::string, std::tuple<int, int, int, int» &data)
00091 {
00092     std::cout « std::left « std::setw(8) « "State" « std::setw(15) « "East" « std::setw(15) « "West" «
      std::setw(15) « "North" « std::setw(15) « "South" « std::endl;
00093     for (const auto &entry : data)
00094     {
00095         std::cout « std::left « std::setw(8) « entry.first « std::setw(15) « std::get<0>(entry.second)
      « std::setw(15) « std::get<1>(entry.second) « std::setw(15) « std::get<2>(entry.second) «
      std::setw(15) « std::get<3>(entry.second) « std::endl;
00096     }
00097 }
00098 
00104 int main()
00105 {
00106     std::string file_name = "us_postal_codes_csv.csv";
00107     Buffer buffer(file_name);
00108     std::map<std::string, std::tuple<int, int, int, int» data = parse_data(buffer.get_zipcodes());
00109     print_data(data);
00110     return 0;
00111 }
```

4.11 zipcode.cpp File Reference

```
#include <iostream>
#include "zipcode.h"
```

Include dependency graph for zipcode.cpp:



Functions

- std::ostream & operator<< (std::ostream &outputstream, const ZipCodeData &zipcode)

4.11.1 Function Documentation

4.11.1.1 operator<<()

```
std::ostream & operator<< (
            std::ostream & outputstream,
            const ZipCodeData & zipcode)
```

Parameters

outputstream	The output stream.
zipcode	The ZipCodeData object to be printed.

Returns

The output stream after the zip code data is written.

Definition at line 12 of file zipcode.cpp.

4.12 zipcode.cpp

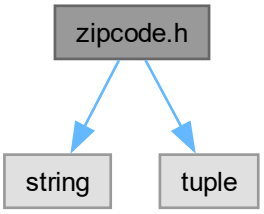
Go to the documentation of this file.

```
00001
00004 #include <iostream>
00005 #include "zipcode.h"
00006
00007 ZipCodeData::ZipCodeData(std::tuple<int, std::string, std::string, std::string, float, float> tuple)
00008 {
00009     std::tie(this->zip_code, this->place_name, this->state, this->county, this->latitude,
      this->longitude) = tuple;
00010 }
00011
00012 std::ostream& operator«(std::ostream& outputstream, const ZipCodeData& zipcode) {
00013     outputstream « "Zip Code: " « zipcode.zip_code « std::endl;
00014     outputstream « "Place Name: " « zipcode.place_name « std::endl;
00015     outputstream « "State: " « zipcode.state « std::endl;
00016     outputstream « "County: " « zipcode.county « std::endl;
00017     outputstream « "Latitude and Longitude: " « zipcode.latitude « ", " « zipcode.longitude «
      std::endl;
00018
00019     return outputstream;
00020 }
```

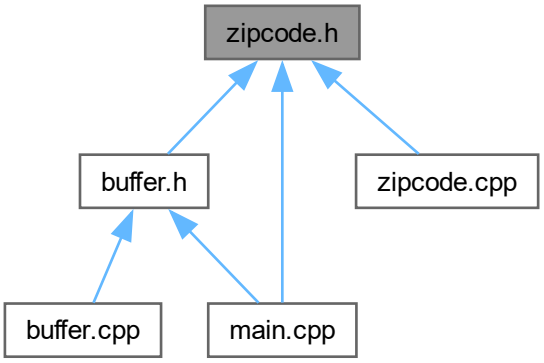
4.13 zipcode.h File Reference

```
#include <string>
#include <tuple>
```

Include dependency graph for zipcode.h:



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



Classes

- struct ZipCodeData

  The ZipCodeData struct holds data for a single zip code, including its coordinates and place information.

4.14 zipcode.h

Go to the documentation of this file.

```
00001
00004 #ifndef ZIP_CODE_H
00005 #define ZIP_CODE_H
00006
00007 #include <string>
00008 #include <tuple>
00009
00013 struct ZipCodeData
00014 {
00015     int zip_code;
00016     std::string place_name;
00017     std::string state;
00018     std::string county;
00019     float latitude;
00020     float longitude;
00021
00027     ZipCodeData(std::tuple<int, std::string, std::string, std::string, float, float>);
00028
00036     friend std::ostream& operator<<(std::ostream&, const ZipCodeData&);
00037 };
00038
00039 #endif // ZIP_CODE_H
```