Zip Code

Generated by Doxygen 1.16.1

Chapter 1

Class Index

1.1 Class List

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

Chapter 2

File Index

2.1 File List

Here is a list of all documented files with brief descriptions:

Chapter 3

Class Documentation

3.1 StateExtremes Struct Reference

Holds the extreme ZIP codes for a single state.

Public Member Functions

- StateExtremes ()
  Default constructor.
- void update (const ZipCodeRecord &record)
  Update extreme values with a new record.

Public Attributes

- ZipCodeRecord easternmost
- ZipCodeRecord westernmost
- ZipCodeRecord northernmost
- ZipCodeRecord southernmost
- bool initialized

3.1.1 Detailed Description

Holds the extreme ZIP codes for a single state.

Precondition

All fields must be properly initialized before use

Postcondition

Instances of StateExtremes can be used to track and update extreme ZIP codes for a state

Note

This structure maintains the four geographic extreme ZIP codes for a state: easternmost, westernmost, northernmost, and southernmost. It updates these values as records are processed.

3.1.2 Constructor & Destructor Documentation

3.1.2.1 StateExtremes()

```
StateExtremes::StateExtremes ()  [inline]
```

Default constructor.

Parameters

None	

Precondition

None

Postcondition

All fields are initialized to default values

Returns

An instance of StateExtremes with default values

Initializes the structure with sentinel values that will be replaced by the first actual record encountered.

3.1.3 Member Function Documentation

3.1.3.1 update()

```
void StateExtremes::update (
            const ZipCodeRecord & record)  [inline]
```

Update extreme values with a new record.

Parameters

record	The ZIP code record to consider

Precondition

record must contain valid ZIP code data

Postcondition

Extreme values are updated if the record represents a new extreme in any direction

Returns

None

Compares the provided record's coordinates with current extremes and updates if the record represents a new extreme in any direction.

Note

For longitude in the US (negative values):

- Easternmost has the LEAST (most negative) value
- Westernmost has the GREATEST (least negative) value

3.1.4 Member Data Documentation

3.1.4.1 easternmost

```
ZipCodeRecord StateExtremes::easternmost
```

ZIP code with least (most negative) longitude

3.1.4.2 initialized

```
bool StateExtremes::initialized
```

Flag indicating if any data has been set

3.1.4.3 northernmost

```
ZipCodeRecord StateExtremes::northernmost
```

ZIP code with greatest latitude

3.1.4.4 southernmost

```
ZipCodeRecord StateExtremes::southernmost
```

ZIP code with least latitude

3.1.4.5 westernmost

```
ZipCodeRecord StateExtremes::westernmost
```

ZIP code with greatest (least negative) longitude

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

- main.cpp

3.2 ZipCodeBuffer Class Reference

A class to manage a buffer of zip codes Assumptions:

```
#include <ZipCodeBuffer.h>
```

Public Member Functions

- ZipCodeBuffer (const std::string &csvFilename)

  Constructor.
- ~ZipCodeBuffer ()

  Destructor.
- bool open ()

  Open the CSV file for reading.
- void close ()

  Close the CSV file.
- bool getNextRecord (ZipCodeRecord &record)

  Read the next record from the file.
- bool isOpen () const

  Check if the file is currently open.
- std::string getFilename () const

  Get the Filename object.
- long getRecordCount () const

  Get the number of records read so far.
- bool reset ()

  Reset the buffer to the beginning of the file.

3.2.1 Detailed Description

A class to manage a buffer of zip codes Assumptions:

Buffered reader for ZIP code CSV files.

Class to handle buffered reading of ZIP code records from a CSV file Assumptions: – The CSV file has a header row that should be skipped – Each subsequent row contains valid ZIP code data – The buffer reads one record at a time – The file is well-formed and accessible – Zip codes are represented as integers – Latitude and longitude are represented as doubles – String fields do not contain commas – The class manages its own file stream – The user is responsible for checking if the file opened successfully – The user is responsible for handling end-of-file conditions – The class provides methods to open, close, and read records from the file – The class tracks the number of records read – The class provides a method to reset the read position to the beginning of the file – The class provides methods to check if the file is open and to get the filename – The class provides a method to get the number of records read so far – The class provides a method to trim whitespace from strings – The class provides a method to extract fields from a CSV line – The class provides a method to parse a CSV line into a ZipCodeRecord.

- The CSV file is well-formed with the expected columns in the correct order.
- The file uses UTF-8 encoding and standard CSV formatting (commas as delimiters, quotes for fields containing commas).
- The class is responsible for managing the file stream and parsing the CSV data into ZipCodeRecord structures.
- The class does not handle concurrent access or multi-threading scenarios; it is intended for single-threaded use.
- The class assumes that the CSV file is not modified externally while it is being read, and does not implement file change detection.

3.2.2 Constructor & Destructor Documentation

3.2.2.1 ZipCodeBuffer()

```
ZipCodeBuffer::ZipCodeBuffer (
            const std::string & csvFilename)  [explicit]
```

Constructor.

Construct a new Zip Code Buffer:: Zip Code Buffer object.

Parameters

csvFilename	Path to the CSV file

Precondition

csvFilename is a valid file path

Postcondition

ZipCodeBuffer is initialized but file is not opened

Returns

An instance of ZipCodeBuffer

Note

File will be opened using open() method

Precondition

csvFilename is a valid file path

Parameters

csvFilename	

Postcondition

ZipCodeBuffer is initialized but file is not opened

Returns

An instance of ZipCodeBuffer

Note

File will be opened using open() method

3.2.2.2 ~ZipCodeBuffer()

```
ZipCodeBuffer::~ZipCodeBuffer ()
```

Destructor.

Destroy the Zip Code Buffer:: Zip Code Buffer object.

Precondition

File is opened

Postcondition

File is closed if it was open

Returns

None

Note

Calls close() if file is still open

3.2.3 Member Function Documentation

3.2.3.1 close()

```
void ZipCodeBuffer::close ()
```

Close the CSV file.

Precondition

File is open

Postcondition

File is closed

Returns

None

Note

Safe to call even if file is already closed

3.2.3.2 getFilename()

```
std::string ZipCodeBuffer::getFilename () const
```

Get the Filename object.

Get the filename of the CSV file being read.

Precondition

None

Postcondition

Returns the name of the CSV file being read

Note

Useful for logging or error messages

Returns

* std::string

Filename as a string

Precondition

None

Postcondition

None

Returns

Filename as a string

3.2.3.3 getNextRecord()

```
bool ZipCodeBuffer::getNextRecord (
            ZipCodeRecord & record)
```

Read the next record from the file.

Parameters

record	Reference to ZipCodeRecord to populate

Returns

true if record read successfully, false if EOF or error

Precondition

File is open and ready for reading

Postcondition

record is populated with the next record's data

Note

Skips header row if not already skipped

Parameters

record	Reference to ZipCodeRecord to populate

Returns

true if record read successfully, false if EOF or error

Precondition

File is open and ready for reading

Postcondition

record is populated with the next record's data elements true is return otherwise

Returns

true if arrays are not equal, false otherwise

Remarks

Uses operator== for implementation

3.2.3.4 getRecordCount()

```
long ZipCodeBuffer::getRecordCount () const
```

Get the number of records read so far.

Precondition

None

Postcondition

Returns the count of records read

Returns

Number of records read

Note

Useful for tracking progress

Returns

Number of records read

Precondition

None

Postcondition

None

Returns

Number of records read

3.2.3.5 isOpen()

```
bool ZipCodeBuffer::isOpen () const
```

Check if the file is currently open.

Precondition

None

Postcondition

Returns the open status of the file

Returns

true if file is open, false otherwise

Note

Useful for verifying file state before reading

Returns

true if file is open, false otherwise

Precondition

None

Postcondition

None

Returns

true if file is open, false otherwise

3.2.3.6 open()

```
bool ZipCodeBuffer::open ()
```

Open the CSV file for reading.

Returns

true if file opened successfully

Precondition

File is not already open

Postcondition

File is opened and ready for reading

Note

Handles file not found or access errors

3.2.3.7 reset()

```
bool ZipCodeBuffer::reset ()
```

Reset the buffer to the beginning of the file.

Precondition

File is open

Postcondition

File read position is reset to the beginning, record count set to zero

Returns

true if reset successful

Note

Skips header row after reset

Returns

true if reset successful, false otherwise

Precondition

File is open

Postcondition

File is reset to the beginning and ready for reading

Note

Closes and reopens the file to reset state

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

- ZipCodeBuffer.h
- ZipCodeBuffer.cpp

3.3 ZipCodeRecord Struct Reference

Represents a single ZIP code record with associated data This program reads a CSV file containing US postal code data and generates a report showing the easternmost, westernmost, northernmost, and southernmost ZIP codes for each state. The results are presented in alphabetical order by state abbreviation.

```
#include <ZipCodeRecord.h>
```

Public Member Functions

- ZipCodeRecord ()

  Default constructor.
- ZipCodeRecord (int zip, const std::string &place, const std::string &st, const std::string &cnty, double lat, double lon)

  Parameterized constructor.

Public Attributes

- int zipCode
- std::string placeName
- std::string state
- std::string county
- double latitude
- double longitude

3.3.1 Detailed Description

Represents a single ZIP code record with associated data This program reads a CSV file containing US postal code data and generates a report showing the easternmost, westernmost, northernmost, and southernmost ZIP codes for each state. The results are presented in alphabetical order by state abbreviation.

Structure to hold information for a single ZIP code.

Structure to hold information for a single ZIP code This header file defines the ZipCodeRecord structure, which is used to store information about US postal codes. Each record includes the ZIP code, place name, state, county, and geographic coordinates (latitude and longitude). This structure serves as the fundamental data unit for managing and processing ZIP code information in our application.

Geographic extremes are determined by:

- Easternmost: Least (most negative) longitude value
- Westernmost: Greatest (least negative) longitude value
- Northernmost: Greatest latitude value
- Southernmost: Least latitude value

Note

In the US, longitude values are negative (west of Prime Meridian) so "least longitude" means "furthest east"

Precondition

All fields must be properly initialized before use

Postcondition

Instances of ZipCodeRecord can be used to store and retrieve ZIP code information

Note

This structure contains all fields from the US Postal Code database:
- ZIP code (5-digit postal code)
- Place name (city/town name)
- State (2-character state abbreviation)
- County name
- Geographic coordinates (latitude and longitude)

Longitude values are negative for locations west of the Prime Meridian

Latitude values are positive for locations north of the Equator

3.3.2 Constructor & Destructor Documentation

3.3.2.1 ZipCodeRecord() [1/2]

```
ZipCodeRecord::ZipCodeRecord ()  [inline]
```

Default constructor.

Precondition

None

Postcondition

All fields are initialized to default values

Returns

An instance of ZipCodeRecord with default values

Note

Initializes all numeric fields to zero and string fields to empty strings.

3.3.2.2 ZipCodeRecord() [2/2]

```
ZipCodeRecord::ZipCodeRecord (
            int zip,
            const std::string & place,
            const std::string & st,
            const std::string & cnty,
            double lat,
            double lon)  [inline]
```

Parameterized constructor.

Parameters

zip	ZIP code value
place	Place name
st	State abbreviation
cnty	County name
lat	Latitude coordinate
lon	Longitude coordinate

Precondition

All parameters must be provided with valid values

Postcondition

An instance of ZipCodeRecord is created with the provided values

Returns

An instance of ZipCodeRecord initialized with specified values

Note

This constructor allows for convenient initialization of all fields in one step.

The caller is responsible for ensuring that the provided values are valid (e.g., zip code is 5 digits, state is 2 characters, etc.).

3.3.3 Member Data Documentation

3.3.3.1 county

```
std::string ZipCodeRecord::county
```

County name

3.3.3.2 latitude

```
double ZipCodeRecord::latitude
```

Latitude in decimal degrees (positive for north of equator)

3.3.3.3 longitude

```
double ZipCodeRecord::longitude
```

Longitude in decimal degrees (negative for west of prime meridian)

3.3.3.4 placeName

```
std::string ZipCodeRecord::placeName
```

City or town name

3.3.3.5 state

```
std::string ZipCodeRecord::state
```

2-character state abbreviation

3.3.3.6 zipCode

```
int ZipCodeRecord::zipCode
```

5-digit ZIP code

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

- ZipCodeRecord.h

Chapter 4

File Documentation

4.1 main.cpp File Reference

Application to find extreme ZIP codes by state.

```
#include <iostream>
#include <iomanip>
#include <map>
#include <string>
#include <limits>
#include "ZipCodeBuffer.h"
#include "ZipCodeRecord.h"
```

Classes

- struct StateExtremes

  Holds the extreme ZIP codes for a single state.

Functions

- void printTableHeader ()

  Print a formatted table header.
- void printStateRow (const std::string &state, const StateExtremes &extremes)

  Print extreme ZIP codes for a single state.
- int processZipCodeFile (const std::string &filename)

  Process CSV file and generate extreme ZIP codes report.
- int main (int argc, char ∗argv[ ])

  Main entry point of the application.

4.1.1 Detailed Description

Application to find extreme ZIP codes by state.

Author

CSCI 331 Team 2

Date

02/12/2026

4.1.2 Function Documentation

4.1.2.1 main()

```
int main (
            int argc,
            char * argv[])
```

Main entry point of the application.

Parameters

argc	Number of command-line arguments
argv	Array of command-line argument strings

Precondition

None

Postcondition

The program processes the specified CSV file and generates a report

Returns

0 on success, 1 on error

The program expects one command-line argument: the path to the CSV file. If no argument is provided, it uses a default filename.

Usage: ./zipcode_extremes <csv_filename>

The program reads the CSV file and generates a report showing the easternmost, westernmost, northernmost, and southernmost ZIP codes for each state, listed in alphabetical order by state.

4.1.2.2 printStateRow()

```
void printStateRow (
            const std::string & state,
            const StateExtremes & extremes)
```

Print extreme ZIP codes for a single state.

Parameters

state	Two-character state abbreviation
extremes	StateExtremes structure containing the extreme values

Precondition

state is a valid state abbreviation and extremes contains valid data

Postcondition

A formatted row of extreme ZIP codes is printed to standard output

Returns

None

Prints one row of the report showing the four extreme ZIP codes for the specified state. All ZIP codes are right-aligned.

4.1.2.3 printTableHeader()

```
void printTableHeader ()
```

Print a formatted table header.

Precondition

None

Postcondition

Column headers are printed to standard output

Returns

None Prints column headers for the extreme ZIP codes report. Headers are formatted to align with data columns.

4.1.2.4 processZipCodeFile()

```
int processZipCodeFile (
            const std::string & filename)
```

Process CSV file and generate extreme ZIP codes report.

Parameters

filename	Path to the CSV file containing ZIP code data

Precondition

filename is a valid path to a CSV file with the expected format

Postcondition

The CSV file is read and a report of extreme ZIP codes by state is printed

Returns

0 on success, non-zero on error

This function:

1. Opens the CSV file using ZipCodeBuffer
2. Reads all records and tracks extremes for each state
3. Prints a formatted report of results

The report lists states alphabetically and shows the four extreme ZIP codes (east, west, north, south) for each state.

4.2 test_buffer.cpp File Reference

Test program for ZipCodeBuffer class.

```
#include <iostream>
#include <iomanip>
#include "ZipCodeBuffer.h"
```

Functions

- int **main** ()

4.2.1 Detailed Description

Test program for ZipCodeBuffer class.

Author

[Your Name]

Date

[Date]

4.3 ZipCodeBuffer.cpp File Reference

Implementation of the ZipCodeBuffer class.

```
#include "ZipCodeBuffer.h"
#include <sstream>
#include <iostream>
#include <algorithm>
```

4.3.1 Detailed Description

Implementation of the ZipCodeBuffer class.

Author

CSCI 331 Team 2

Date

02/12/2026

4.4 ZipCodeBuffer.h File Reference

Buffer class for reading ZIP code records from CSV files.

```
#include <fstream>
#include <string>
#include "ZipCodeRecord.h"
```

Classes

- class ZipCodeBuffer

  A class to manage a buffer of zip codes Assumptions:

4.4.1 Detailed Description

Buffer class for reading ZIP code records from CSV files.

Author

CSCI 331 Team 2

Date

2/12/2026

Version

1.0

4.5 ZipCodeBuffer.h

Go to the documentation of this file.

```
00001
00031
00032 #ifndef ZIPCODEBUFFER_H
00033 #define ZIPCODEBUFFER_H
00034
00035 #include <fstream>
00036 #include <string>
00037 #include "ZipCodeRecord.h"
00038
00044 class ZipCodeBuffer {
00045 private:
00046     std::ifstream inputFile;
00047     std::string filename;
00048     bool fileIsOpen;
00049     bool headerSkipped;
00050     int recordCount;
00051
00061     bool parseCsvLine(const std::string& line, ZipCodeRecord& record);
00062
00063
00073     std::string extractField(const std::string& line, size_t& startPos);
00074
00083     std::string trim(const std::string& str);
00084
00085 public:
00094     explicit ZipCodeBuffer(const std::string& csvFilename);
00095
00103     ~ZipCodeBuffer();
00104
00113     bool open();
00114
00123     void close();
00124
00133     bool getNextRecord(ZipCodeRecord& record);
00134
00142     bool isOpen() const;
00143
00151     std::string getFilename() const;
00152
00160     long getRecordCount() const;
00161
00169     bool reset();
00170 };
00171 #endif // ZIPCODEBUFFER_H
```

4.6 ZipCodeRecord.h File Reference

Data structure for US Postal Code records.

```
#include <string>
```

Classes

- struct ZipCodeRecord

  Represents a single ZIP code record with associated data This program reads a CSV file containing US postal code data and generates a report showing the easternmost, westernmost, northernmost, and southernmost ZIP codes for each state. The results are presented in alphabetical order by state abbreviation.

4.6.1 Detailed Description

Data structure for US Postal Code records.

Author

CSCI 331 Team 2

Date

02/12/2026

Version

1.0

4.7 ZipCodeRecord.h

Go to the documentation of this file.

```
00001
00012
00013 #ifndef ZIPCODERECORD_H
00014 #define ZIPCODERECORD_H
00015
00016 #include <string>
00017
00032 struct ZipCodeRecord {
00033     int zipCode;
00034     std::string placeName;
00035     std::string state;
00036     std::string county;
00037     double latitude;
00038     double longitude;
00039
00047     ZipCodeRecord() : zipCode(0), latitude(0.0), longitude(0.0) {}
00048
00063     ZipCodeRecord(int zip, const std::string& place, const std::string& st,
00064                   const std::string& cnty, double lat, double lon)
00065         : zipCode(zip), placeName(place), state(st), county(cnty), latitude(lat), longitude(lon) {}
00066 };
00067
00068 #endif // ZIPCODERECORD_H
```