ZipCodeProject

Generated by Doxygen 1.14.0

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 files with brief descriptions:

Chapter 3

Class Documentation

3.1 StateRecord Struct Reference

Holds the four extreme ZIP codes AND their coordinates.

Public Attributes

- string easternmost_zip
- double easternmost_lon = -numeric_limits<double>::max()
- string westernmost_zip
- double westernmost_lon = numeric_limits<double>::max()
- string northernmost_zip
- double northernmost_lat = -numeric_limits<double>::max()
- string southernmost_zip
- double southernmost_lat = numeric_limits<double>::max()

3.1.1 Detailed Description

Holds the four extreme ZIP codes AND their coordinates.

- Tracks easternmost, westernmost, northernmost, and southernmost ZIP codes.
- Initialized with extreme numeric values to ensure first record is correctly stored.

3.1.2 Member Data Documentation

3.1.2.1 easternmost_lon

```
double StateRecord::easternmost_lon = -numeric_limits<double>::max()
```

Longitude of easternmost ZIP

3.1.2.2 easternmost_zip

```
string StateRecord::easternmost_zip
```

ZIP code with largest longitude.

3.1.2.3 northernmost_lat

```
double StateRecord::northernmost_lat = -numeric_limits<double>::max()
```

Latitude of northernmost ZIP.

3.1.2.4 northernmost_zip

```
string StateRecord::northernmost_zip
```

ZIP code with largest latitude.

3.1.2.5 southernmost_lat

```
double StateRecord::southernmost_lat = numeric_limits<double>::max()
```

Latitude of southernmost ZIP.

3.1.2.6 southernmost_zip

```
string StateRecord::southernmost_zip
```

ZIP code with smallest latitude.

3.1.2.7 westernmost_lon

```
double StateRecord::westernmost_lon = numeric_limits<double>::max()
```

Longitude of westernmost ZIP

3.1.2.8 westernmost_zip

```
string StateRecord::westernmost_zip
```

ZIP code with smallest longitude.

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

- main.cpp

3.2 ZipCodeRecordBuffer Class Reference

Buffer class for reading and storing ZIP code records from a CSV file.

```
#include <ZipCodeRecordBuffer.h>
```

Public Member Functions

- ZipCodeRecordBuffer ()

  Default constructor. Initializes all fields to empty strings.
- bool ReadRecord (ifstream &file)

  Reads a single record from a CSV file.
- string getZipCode () const

  Get the ZIP code field.
- string getPlaceName () const

  Get the place name field.
- string getState () const

  Get the state abbreviation field.
- string getCounty () const

  Get the county field.
- double getLatitude () const

  Get the latitude field.
- double getLongitude () const

  Get the longitude field.

3.2.1 Detailed Description

Buffer class for reading and storing ZIP code records from a CSV file.

Precondition

A properly formatted CSV file must be opened before calling ReadRecord().

Postcondition

After a successful call to ReadRecord(), internal fields contain the parsed data.

Remarks

Truncates fields longer than their maximum allowed size.

See also

getLatitude(), getLongitude()

3.2.2 Constructor & Destructor Documentation

3.2.2.1 ZipCodeRecordBuffer()

```
ZipCodeRecordBuffer::ZipCodeRecordBuffer ()  [inline]
```

Default constructor. Initializes all fields to empty strings.

3.2.3 Member Function Documentation

3.2.3.1 getCounty()

```
string ZipCodeRecordBuffer::getCounty () const  [inline]
```

Get the county field.

Returns

The county name as a string.

3.2.3.2 getLatitude()

```
double ZipCodeRecordBuffer::getLatitude () const  [inline]
```

Get the latitude field.

Returns

The latitude as a double.

3.2.3.3 getLongitude()

```
double ZipCodeRecordBuffer::getLongitude () const  [inline]
```

Get the longitude field.

Returns

The longitude as a double.

3.2.3.4 getPlaceName()

```
string ZipCodeRecordBuffer::getPlaceName () const  [inline]
```

Get the place name field.

Returns

The place name as a string.

3.2.3.5 getState()

```
string ZipCodeRecordBuffer::getState () const  [inline]
```

Get the state abbreviation field.

Returns

The 2-character state abbreviation.

3.2.3.6 getZipCode()

```
string ZipCodeRecordBuffer::getZipCode () const  [inline]
```

Get the ZIP code field.

Returns

The ZIP code as a string.

3.2.3.7 ReadRecord()

```
bool ZipCodeRecordBuffer::ReadRecord (
             ifstream & file)  [inline]
```

Reads a single record from a CSV file.

Parameters

file	Input file stream containing CSV data.

Returns

True if a record was successfully read, false if EOF is reached.

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

- include/ZipCodeRecordBuffer.h

Chapter 4

File Documentation

4.1 include/ZipCodeRecordBuffer.h File Reference

Declaration of the ZipCodeRecordBuffer class for reading ZIP code CSV records.

```
#include <string>
#include <fstream>
#include <sstream>
#include <iomanip>
#include <cstdlib>
#include <cmath>
```

Classes

- class ZipCodeRecordBuffer

    Buffer class for reading and storing ZIP code records from a CSV file.

Variables

- const int ZIP_CODE_LENGTH = 5

    Maximum length for each field.
- const int PLACE_NAME_LENGTH = 50
- const int STATE_LENGTH = 2
- const int COUNTY_LENGTH = 50
- const int LAT_LONG_LENGTH = 10

4.1.1 Detailed Description

Declaration of the ZipCodeRecordBuffer class for reading ZIP code CSV records.

Authors

Evan Brisbin, Jason Donkor, Ethan Fischer, Tim Stevens, Markose Mesay

Date

2025-09-22

Version

1.0

Provides functionality to read, store, and access U.S. ZIP code records from a CSV file. Each record includes:

- ZIP code
- City
- State abbreviation
- County
- Latitude and Longitude Fields are stored in fixed-length strings with truncation applied if values exceed their maximum allowed length.

4.1.2 Variable Documentation

4.1.2.1 COUNTY_LENGTH

```
const int COUNTY_LENGTH = 50
```

Maximum county name length.

4.1.2.2 LAT_LONG_LENGTH

```
const int LAT_LONG_LENGTH = 10
```

Maximum latitude/longitude length.

4.1.2.3 PLACE_NAME_LENGTH

```
const int PLACE_NAME_LENGTH = 50
```

Maximum city/place name length.

4.1.2.4 STATE_LENGTH

```
const int STATE_LENGTH = 2
```

Maximum state abbreviation length.

4.1.2.5 ZIP_CODE_LENGTH

```
const int ZIP_CODE_LENGTH = 5
```

Maximum length for each field.

Maximum ZIP code length.

4.2 ZipCodeRecordBuffer.h

Go to the documentation of this file.

```


#ifndef ZipCodeRecordBuffer_H
#define ZipCodeRecordBuffer_H

#include <string>
#include <fstream>
#include <sstream>
#include <iomanip> // For setprecision
#include <cstdlib> // For atof
#include <cmath>   // For fabs

using namespace std;

const int ZIP_CODE_LENGTH = 5;
const int PLACE_NAME_LENGTH = 50;
const int STATE_LENGTH = 2;
const int COUNTY_LENGTH = 50;
const int LAT_LONG_LENGTH = 10;

class ZipCodeRecordBuffer {
public:
    ZipCodeRecordBuffer() {
        // Initialize all fields to empty strings
        for (int i = 0; i < 5; ++i) {
            m_fields[i] = "";
        }
    }
    bool ReadRecord(ifstream& file) {
        string line;
        if (!getline(file, line)) {
            return false;
        }

        stringstream ss(line);
        string field;
        int field_count = 0;

        // Read and store each field, truncating if necessary
        // Order: Zip Code, Place Name, State, County, Lat, Long
        // You'll need to know the exact column order of your CSV
        while (getline(ss, field, ',') && field_count < 6) {
            // Truncate fields if they exceed the fixed length
            if (field_count == 0 && field.length() > ZIP_CODE_LENGTH) {
                m_fields[0] = field.substr(0, ZIP_CODE_LENGTH);
            } else if (field_count == 1 && field.length() > PLACE_NAME_LENGTH) {
                m_fields[1] = field.substr(0, PLACE_NAME_LENGTH);
            } else if (field_count == 2 && field.length() > STATE_LENGTH) {
                m_fields[2] = field.substr(0, STATE_LENGTH);
            } else if (field_count == 3 && field.length() > COUNTY_LENGTH) {
                m_fields[3] = field.substr(0, COUNTY_LENGTH);
            } else if (field_count >= 4 && field.length() > LAT_LONG_LENGTH) {
                m_fields[field_count] = field.substr(0, LAT_LONG_LENGTH);
            } else {
                m_fields[field_count] = field;
            }
            field_count++;
        }
        return true;
    }

    // Accessor methods to retrieve data, converting from string to the correct type
    string getZipCode() const { return m_fields[0]; }

    string getPlaceName() const { return m_fields[1]; }

    string getState() const { return m_fields[2]; }

    string getCounty() const { return m_fields[3]; }

    double getLatitude() const {
        return atof(m_fields[4].c_str());
    }

    double getLongitude() const {
        return atof(m_fields[5].c_str());
    }

private:
    string m_fields[6];
};

#endif // FIXED_ZIP_CODE_RECORD_BUFFER_H
```

4.3 main.cpp File Reference

Main program to read ZIP code CSV and calculate geographic extremes per state.

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

Classes

- struct StateRecord

  Holds the four extreme ZIP codes AND their coordinates.

Functions

- int main ()

  Main program entry point.

4.3.1 Detailed Description

Main program to read ZIP code CSV and calculate geographic extremes per state.

Authors

Evan Brisbin, Jason Donkor, Ethan Fischer, Tim Stevens, Markose Mesay

Date

2025-09-22

Version

1.0

- Opens a ZIP code CSV file and reads each record using ZipCodeRecordBuffer.
- Groups records by state and updates easternmost, westernmost, northernmost, and southernmost ZIP codes.
- Prints a formatted table of results

4.3.2 Function Documentation

4.3.2.1 main()

```
int main ()
```

Main program entry point.

- Reads the ZIP code CSV.
- Updates StateRecord map with geographic extremes.
- Prints results table.

  Precondition

  "us_postal_codes.csv" must exist and be accessible.

  Postcondition

  Map `all_states` contains geographic extremes for all states found in the CSV.

  Returns

  0 if program succeeds, 1 if file cannot be opened.

< [OUT] Map storing extreme ZIP codes for each state.

< Skip header line.

< [IN, OUT] Reads record and updates buffer fields.

Print table header for state extremes.

< Separator line.

Print each state's geographic extremes.

Loops through `all_states` and prints ZIP codes in aligned columns.

Note

States are printed in alphabetical order.

< Reference to current state record.

Index