Project 2 - Group 4 - Tristan Adams, Andrew Clayton, Zachary Sunder

Generated by Doxygen 1.9.8

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

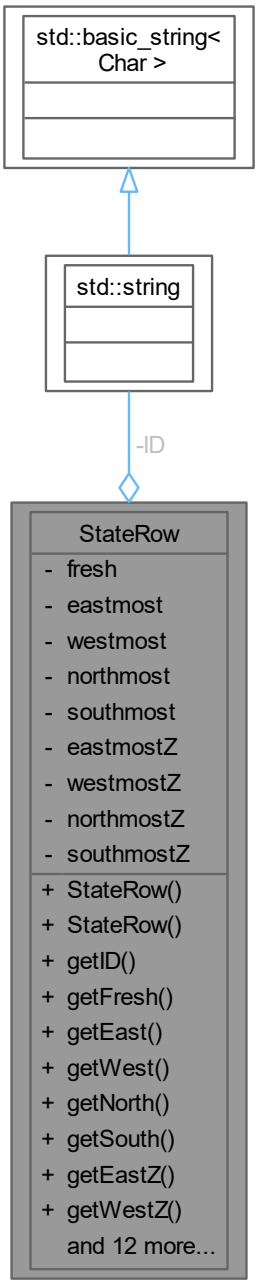
Class Documentation

3.1 StateRow Class Reference

A class representing a state with geographic data.

`#include <StateRow.h>`

Collaboration diagram for StateRow:



Public Member Functions

- StateRow ()

  Default constructor.
- StateRow (string id, double e, double w, double n, double s, int eZ, int wZ, int nZ, int sZ)

  Parameterized constructor.
- string getID ()

Get the state's ID.

- bool getFresh () const

  Get the fresh boolean.
- double getEast ()

  Get the easternmost longitude.
- double getWest ()

  Get the westernmost longitude.
- double getNorth ()

  Get the northernmost latitude.
- double getSouth ()

  Get the southernmost latitude.
- int getEastZ ()

  Get the easternmost longitude Zipcode.
- int getWestZ ()

  Get the westernmost longitude Zipcode.
- int getNorthZ ()

  Get the northernmost latitude Zipcode.
- int getSouthZ ()

  Get the southernmost latitude Zipcode.
- void setFresh (bool b)

  Set the fresh boolean.
- void setID (string id)

  Set the state's ID.
- void setEast (double east)

  Set the easternmost longitude.
- void setWest (double west)

  Set the westernmost longitude.
- void setNorth (double north)

  Set the northernmost latitude.
- void setSouth (double south)

  Set the southernmost latitude.
- void setEastZ (int eastZ)

  Set the easternmost longitude Zipcode.
- void setWestZ (int westZ)

  Set the westernmost longitude Zipcode.
- void setNorthZ (int northZ)

  Set the northernmost latitude Zipcode.
- void setSouthZ (int southZ)

  Set the southernmost latitude Zipcode.

Private Attributes

- bool fresh
- string ID

  State ID.
- double eastmost

  Easternmost longitude.
- double westmost

  Westernmost longitude.
- double northmost

Northernmost latitude.

- double southmost

  Southernmost latitude.

- int eastmostZ

  Easternmost longitude Zipcode.

- int westmostZ

  Westernmost longitude Zipcode.

- int northmostZ

  Northernmost latitude Zipcode.

- int southmostZ

  Southernmost latitude Zipcode.

Friends

- ostream & operator<< (ostream &out, const StateRow &row)

  Overloaded operator to print StateRow objects.

3.1.1 Detailed Description

A class representing a state with geographic data.

Definition at line 19 of file StateRow.h.

3.1.2 Constructor & Destructor Documentation

3.1.2.1 StateRow() [1/2]

```
StateRow::StateRow ( )
```

Default constructor.

Default constructor for StateRow.

Initializes a StateRow object with default values.

- State ID: "TEMP"
- Easternmost longitude: 00.00
- Westernmost longitude: 00.00
- Northernmost latitude: 00.00
- Southernmost latitude: 00.00

Definition at line 19 of file StateRow.cpp.

3.1.2.2 StateRow() [2/2]

```
StateRow::StateRow (
            string id,
            double e,
            double w,
            double n,
            double s,
            int eZ,
            int wZ,
            int nZ,
            int sZ )
```

Parameterized constructor.

Parameterized constructor for StateRow.

Initializes a StateRow object with the specified values.

Parameters

id	The state's ID.
e	Easternmost longitude.
w	Westernmost longitude.
n	Northernmost latitude.
s	Southernmost latitude.

Definition at line 41 of file StateRow.cpp.

3.1.3 Member Function Documentation

3.1.3.1 getEast()

```
double StateRow::getEast ( )
```

Get the easternmost longitude.

Get the easternmost longitude of the state.

Returns

The easternmost longitude.

Definition at line 66 of file StateRow.cpp.

Here is the caller graph for this function:

3.1.3.2 getEastZ()

```
int StateRow::getEastZ ( )
```

Get the easternmost longitude Zipcode.

Get the easternmost longitude Zipcode of the state.

Returns

The easternmost longitude Zipcode.

Definition at line 98 of file StateRow.cpp.

3.1.3.3 getFresh()

```
bool StateRow::getFresh ( ) const
```

Get the fresh boolean.

Returns

The fresh bool.

Definition at line 58 of file StateRow.cpp.

Here is the caller graph for this function:

3.1.3.4 getID()

```
string StateRow::getID ( )
```

Get the state's ID.

Returns

The state's ID.

Definition at line 130 of file StateRow.cpp.

3.1.3.5 getNorth()

```
double StateRow::getNorth ( )
```

Get the northernmost latitude.

Get the northernmost latitude of the state.

Returns

The northernmost latitude.

Definition at line 82 of file StateRow.cpp.

Here is the caller graph for this function:

3.1.3.6 getNorthZ()

```
int StateRow::getNorthZ ( )
```

Get the northernmost latitude Zipcode.

Get the northernmost latitude Zipcode of the state.

Returns

The northernmost latitude Zipcode.

Definition at line 114 of file StateRow.cpp.

3.1.3.7 getSouth()

```
double StateRow::getSouth ( )
```

Get the southernmost latitude.

Get the southernmost latitude of the state.

Returns

The southernmost latitude.

Definition at line 90 of file StateRow.cpp.

Here is the caller graph for this function:

3.1.3.8 getSouthZ()

```
int StateRow::getSouthZ ( )
```

Get the southernmost latitude Zipcode.

Get the southernmost latitude Zipcode of the state.

Returns

The southernmost latitude Zipcode.

Definition at line 122 of file StateRow.cpp.

3.1.3.9 getWest()

```
double StateRow::getWest ( )
```

Get the westernmost longitude.

Get the westernmost longitude of the state.

Returns

The westernmost longitude.

Definition at line 74 of file StateRow.cpp.

Here is the caller graph for this function:

3.1.3.10 getWestZ()

```
int StateRow::getWestZ ( )
```

Get the westernmost longitude Zipcode.

Get the westernmost longitude Zipcode of the state.

Returns

The westernmost longitude Zipcode.

Definition at line 106 of file StateRow.cpp.

3.1.3.11 setEast()

```
void StateRow::setEast (
            double e )
```

Set the easternmost longitude.

Set the easternmost longitude of the state.

Parameters

e	The easternmost longitude to set.

Definition at line 138 of file StateRow.cpp.

Here is the caller graph for this function:

3.1.3.12 setEastZ()

```
void StateRow::setEastZ (
            int eastZ )
```

Set the easternmost longitude Zipcode.

Set the easternmost longitude Zipcode of the state.

Parameters

e	The easternmost longitude Zipcode to set.

Definition at line 170 of file StateRow.cpp.

Here is the caller graph for this function:

3.1.3.13 setFresh()

```
void StateRow::setFresh (
            bool b )
```

Set the fresh boolean.

Parameters

b	bool to set fresh to.

Definition at line 210 of file StateRow.cpp.

Here is the caller graph for this function:

3.1.3.14 setID()

```
void StateRow::setID (
            string id )
```

Set the state's ID.

Parameters

id	The state's ID to set.

Definition at line 202 of file StateRow.cpp.

Here is the caller graph for this function:

3.1.3.15 setNorth()

```
void StateRow::setNorth (
            double n )
```

Set the northernmost latitude.

Set the northernmost latitude of the state.

Parameters

n	The northernmost latitude to set.

Definition at line 154 of file StateRow.cpp.

Here is the caller graph for this function:

3.1.3.16 setNorthZ()

```
void StateRow::setNorthZ (
            int northZ )
```

Set the northernmost latitude Zipcode.

Set the northernmost latitude Zipcode of the state.

Parameters

n	The northernmost latitude Zipcode to set.

Definition at line 186 of file StateRow.cpp.

Here is the caller graph for this function:

3.1.3.17 setSouth()

```
void StateRow::setSouth (
            double s )
```

Set the southernmost latitude.

Set the southernmost latitude of the state.

Parameters

s	The southernmost latitude to set.

Definition at line 162 of file StateRow.cpp.

Here is the caller graph for this function:

3.1.3.18 setSouthZ()

```
void StateRow::setSouthZ (
            int southZ )
```

Set the southernmost latitude Zipcode.

Set the southernmost latitude Zipcode of the state.

Parameters

s	The southernmost latitude Zipcode to set.

Definition at line 194 of file StateRow.cpp.

Here is the caller graph for this function:

3.1.3.19 setWest()

```
void StateRow::setWest (
            double w )
```

Set the westernmost longitude.

Set the westernmost longitude of the state.

Parameters

w	The westernmost longitude to set.

Definition at line 146 of file StateRow.cpp.

Here is the caller graph for this function:

3.1.3.20 setWestZ()

```
void StateRow::setWestZ (
            int westZ )
```

Set the westernmost longitude Zipcode.

Set the westernmost longitude Zipcode of the state.

Parameters

w	The westernmost longitude Zipcode to set.

Definition at line 178 of file StateRow.cpp.

Here is the caller graph for this function:

3.1.4 Friends And Related Symbol Documentation

3.1.4.1 operator<<

```
ostream & operator<< (
            ostream & out,
            const StateRow & row )  [friend]
```

Overloaded operator to print StateRow objects.

Parameters

out	Output stream.
row	StateRow object to print.

Returns

Reference to the output stream.

Definition at line 221 of file StateRow.cpp.

3.1.5 Member Data Documentation

3.1.5.1 eastmost

```
double StateRow::eastmost  [private]
```

Easternmost longitude.

Definition at line 23 of file StateRow.h.

3.1.5.2 eastmostZ

```
int StateRow::eastmostZ  [private]
```

Easternmost longitude Zipcode.

Definition at line 28 of file StateRow.h.

3.1.5.3 fresh

```
bool StateRow::fresh  [private]
```

Definition at line 21 of file StateRow.h.

3.1.5.4 ID

```
string StateRow::ID  [private]
```

State ID.

Definition at line 22 of file StateRow.h.

3.1.5.5 northmost

```
double StateRow::northmost  [private]
```

Northernmost latitude.

Definition at line 25 of file StateRow.h.

3.1.5.6 northmostZ

```
int StateRow::northmostZ  [private]
```

Northernmost latitude Zipcode.

Definition at line 30 of file StateRow.h.

3.1.5.7 southmost

```
double StateRow::southmost  [private]
```

Southernmost latitude.

Definition at line 26 of file StateRow.h.

3.1.5.8 southmostZ

```
int StateRow::southmostZ  [private]
```

Southernmost latitude Zipcode.

Definition at line 31 of file StateRow.h.

3.1.5.9 westmost

```
double StateRow::westmost  [private]
```

Westernmost longitude.

Definition at line 24 of file StateRow.h.

3.1.5.10 westmostZ

```
int StateRow::westmostZ  [private]
```

Westernmost longitude Zipcode.

Definition at line 29 of file StateRow.h.

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

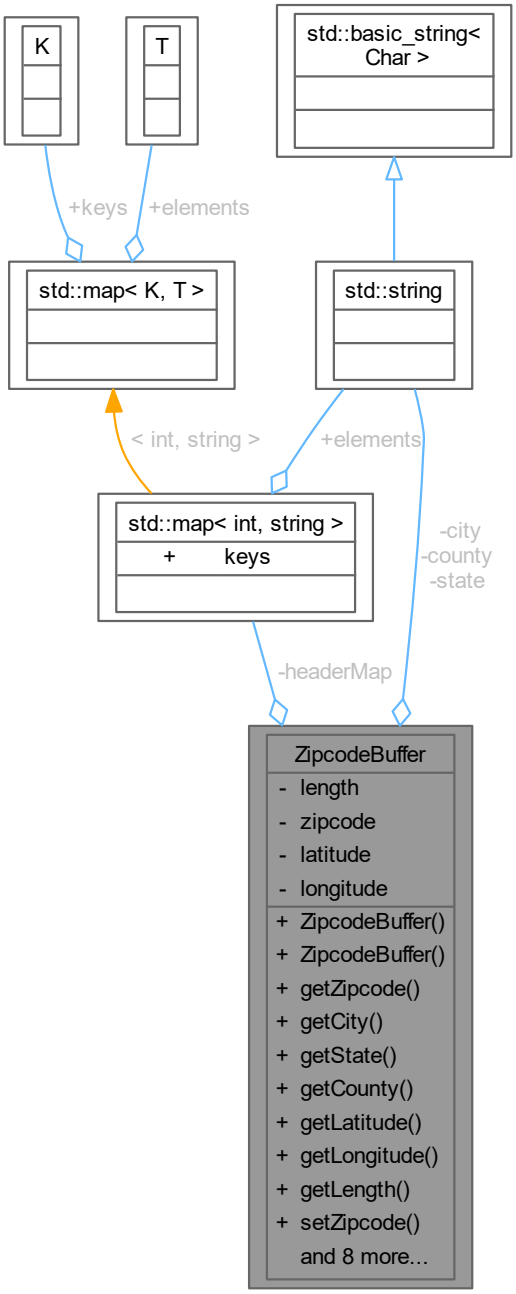
- StateRow.h
- StateRow.cpp

3.2 ZipcodeBuffer Class Reference

Class to represent a Zipcode and its related attributes.

```
#include <ZipcodeBuffer.h>
```

Collaboration diagram for ZipcodeBuffer:



Public Member Functions

- ZipcodeBuffer ()

  Default constructor.
- ZipcodeBuffer (int zipcode, string city, string state, string county, double latitude, double longitude)

  Parameterized constructor to initialize the object with given values.
- int getZipcode () const

Getter for zipcode.

- string getCity () const

  Getter for city.
- string getState () const

  Getter for state.
- string getCounty () const

  Getter for county.
- double getLatitude () const

  Getter for latitude.
- double getLongitude () const

  Getter for longitude.
- int getLength () const

  Getter for length.
- void setZipcode (int zipcode)

  Setter for zipcode.
- void setCity (string city)

  Setter for city.
- void setState (string state)

  Setter for state.
- void setCounty (string county)

  Setter for county.
- void setLatitude (double latitude)

  Setter for latitude.
- void setLongitude (double longitude)

  Setter for longitude.
- void setFromFile (string fileLine)

  Function to take in a line of the CSV file as a string and set the attributes of the object.
- void setLength (int length)

  Setter for length.
- void setHeaderMap (const string &headerLine)

  Function to set the header map of the CSV file.

Private Attributes

- int length

  The length of the line in the CSV file.
- int zipcode

  Zipcode as an integer.
- string city

  City name as a string.
- string state

  State abbreviation as a string.
- string county

  County name as a string.
- double latitude

  Latitude as a double.
- double longitude

  Longitude as a double.
- map< int, string > headerMap

  Map to store the header of the CSV file to keep column sorting flexible.

Friends

- istream & operator>> (istream &in, ZipcodeBuffer &buffer)

  Overloaded input operator for ZipcodeBuffer to read in a line of a CSV file as a string.
- ostream & operator<< (ostream &out, const ZipcodeBuffer &buffer)

  Overloaded output operator for ZipcodeBuffer.

3.2.1 Detailed Description

Class to represent a Zipcode and its related attributes.

Definition at line 25 of file ZipcodeBuffer.h.

3.2.2 Constructor & Destructor Documentation

3.2.2.1 ZipcodeBuffer() [1/2]

```
ZipcodeBuffer::ZipcodeBuffer ( )
```

Default constructor.

Precondition

None

Postcondition

ZipcodeBuffer object is created with default values

Definition at line 16 of file ZipcodeBuffer.cpp.

3.2.2.2 ZipcodeBuffer() [2/2]

```
ZipcodeBuffer::ZipcodeBuffer (
            int zipcode,
            string city,
            string state,
            string county,
            double latitude,
            double longitude )
```

Parameterized constructor to initialize the object with given values.

Parameters

zipcode	Integer value representing the Zipcode.
city	String representing the city.
state	String representing the state.
county	String representing the county.
latitude	Double representing the latitude.
longitude	Double representing the longitude.

Precondition

None

Postcondition

ZipcodeBuffer object is created with given values.

Definition at line 26 of file ZipcodeBuffer.cpp.

3.2.3 Member Function Documentation

3.2.3.1 getCity()

```
string ZipcodeBuffer::getCity ( ) const
```

Getter for city.

Precondition

ZipcodeBuffer object must exist

Postcondition

None

Returns

String value of city

Definition at line 41 of file ZipcodeBuffer.cpp.

3.2.3.2 getCounty()

```
string ZipcodeBuffer::getCounty ( ) const
```

Getter for county.

Precondition

ZipcodeBuffer object must exist

Postcondition

None

Returns

String value of county

Definition at line 49 of file ZipcodeBuffer.cpp.

3.2.3.3 getLatitude()

```
double ZipcodeBuffer::getLatitude ( ) const
```

Getter for latitude.

Precondition

ZipcodeBuffer object must exist

Postcondition

None

Returns

Double value of latitude

Definition at line 53 of file ZipcodeBuffer.cpp.

Here is the caller graph for this function:

3.2.3.4 getLength()

```
int ZipcodeBuffer::getLength ( ) const
```

Getter for length.

Precondition

ZipcodeBuffer object must exist

Postcondition

None

Returns

Integer value of length

Definition at line 143 of file ZipcodeBuffer.cpp.

3.2.3.5 getLongitude()

```
double ZipcodeBuffer::getLongitude ( ) const
```

Getter for longitude.

Precondition

ZipcodeBuffer object must exist

Postcondition

None

Returns

Double value of longitude

Definition at line 57 of file ZipcodeBuffer.cpp.

Here is the caller graph for this function:

3.2.3.6 getState()

```
string ZipcodeBuffer::getState ( ) const
```

Getter for state.

Precondition

ZipcodeBuffer object must exist

Postcondition

None

Returns

String value of state

Definition at line 45 of file ZipcodeBuffer.cpp.

Here is the caller graph for this function:

3.2.3.7 getZipcode()

```
int ZipcodeBuffer::getZipcode ( ) const
```

Getter for zipcode.

Precondition

ZipcodeBuffer object must exist

Postcondition

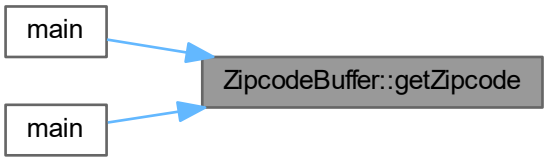
None

Returns

Integer value of zipcode

Definition at line 37 of file ZipcodeBuffer.cpp.

Here is the caller graph for this function:



3.2.3.8 setCity()

```
void ZipcodeBuffer::setCity (
            string city )
```

Setter for city.

Parameters

city	String value of city

Precondition

ZipcodeBuffer object must exist

Postcondition

ZipcodeBuffer object's city attribute is set to the given value

Definition at line 66 of file ZipcodeBuffer.cpp.

3.2.3.9 setCounty()

```
void ZipcodeBuffer::setCounty (
            string county )
```

Setter for county.

Parameters

county	String value of county

Precondition

ZipcodeBuffer object must exist

Postcondition

ZipcodeBuffer object's county attribute is set to the given value

Definition at line 74 of file ZipcodeBuffer.cpp.

3.2.3.10 setFromFile()

```
void ZipcodeBuffer::setFromFile (
            string fileLine )
```

Function to take in a line of the CSV file as a string and set the attributes of the object.

Parameters

fileLine	String representing a line of the CSV file

Precondition

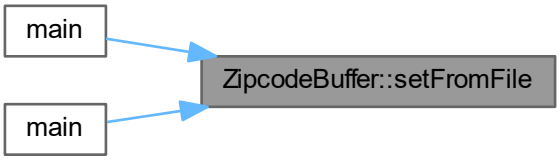
ZipcodeBuffer object must exist

Postcondition

ZipcodeBuffer object's attributes are set to the values in the given string

Definition at line 88 of file ZipcodeBuffer.cpp.

Here is the caller graph for this function:



3.2.3.11 setHeaderMap()

```
void ZipcodeBuffer::setHeaderMap (
            const string & headerLine )
```

Function to set the header map of the CSV file.

Parameters

headerLine	String representing the header line of the CSV file

Precondition

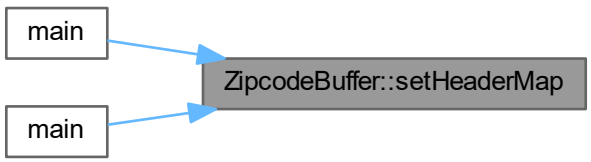
ZipcodeBuffer object must exist

Postcondition

ZipcodeBuffer object's headerMap is set to the values in the given string

Definition at line 126 of file ZipcodeBuffer.cpp.

Here is the caller graph for this function:



3.2.3.12 setLatitude()

```
void ZipcodeBuffer::setLatitude (
            double latitude )
```

Setter for latitude.

Parameters

latitude	Double value of latitude

Precondition

ZipcodeBuffer object must exist

Postcondition

ZipcodeBuffer object's latitude attribute is set to the given value

Definition at line 78 of file ZipcodeBuffer.cpp.

3.2.3.13 setLength()

```
void ZipcodeBuffer::setLength (
            int length )
```

Setter for length.

Parameters

length	Integer value of length

Precondition

ZipcodeBuffer object must exist

Postcondition

ZipcodeBuffer object's length attribute is set to the given value

Definition at line 147 of file ZipcodeBuffer.cpp.

3.2.3.14 setLongitude()

```
void ZipcodeBuffer::setLongitude (
            double longitude )
```

Setter for longitude.

Parameters

longitude	Double value of longitude

Precondition

ZipcodeBuffer object must exist

Postcondition

ZipcodeBuffer object's longitude attribute is set to the given value

Definition at line 82 of file ZipcodeBuffer.cpp.

3.2.3.15 setState()

```
void ZipcodeBuffer::setState (
            string state )
```

Setter for state.

Parameters

state	String value of state

Precondition

ZipcodeBuffer object must exist

Postcondition

ZipcodeBuffer object's state attribute is set to the given value

Definition at line 70 of file ZipcodeBuffer.cpp.

3.2.3.16 setZipcode()

```
void ZipcodeBuffer::setZipcode (
            int zipcode )
```

Setter for zipcode.

Parameters

zipcode	Integer value of zipcode

Precondition

ZipcodeBuffer object must exist

Postcondition

ZipcodeBuffer object's zipcode attribute is set to the given value

Definition at line 62 of file ZipcodeBuffer.cpp.

3.2.4 Friends And Related Symbol Documentation

3.2.4.1 operator<<

```
ostream & operator<< (
            ostream & out,
            const ZipcodeBuffer & buffer )  [friend]
```

Overloaded output operator for ZipcodeBuffer.

Parameters

out	Output stream
a	ZipcodeBuffer to display

Precondition

None

Postcondition

The first a.size elements of a.ptr are displayed to output

Returns

Updated output stream

Definition at line 162 of file ZipcodeBuffer.cpp.

3.2.4.2 operator>>

```
istream & operator>> (
            istream & in,
            ZipcodeBuffer & buffer )  [friend]
```

Overloaded input operator for ZipcodeBuffer to read in a line of a CSV file as a string.

Parameters

in	Input stream
a	ZipcodeBuffer to fill

Precondition

A ZipcodeBuffer object must exist

Postcondition

The first a.size elements of a.ptr are filled with integers read from input

Returns

Updated input stream

Definition at line 154 of file ZipcodeBuffer.cpp.

3.2.5 Member Data Documentation

3.2.5.1 city

```
string ZipcodeBuffer::city  [private]
```

City name as a string.

Definition at line 34 of file ZipcodeBuffer.h.

3.2.5.2 county

```
string ZipcodeBuffer::county  [private]
```

County name as a string.

Definition at line 40 of file ZipcodeBuffer.h.

3.2.5.3 headerMap

```
map<int, string> ZipcodeBuffer::headerMap  [private]
```

Map to store the header of the CSV file to keep column sorting flexible.

Definition at line 49 of file ZipcodeBuffer.h.

3.2.5.4 latitude

```
double ZipcodeBuffer::latitude  [private]
```

Latitude as a double.

Definition at line 43 of file ZipcodeBuffer.h.

3.2.5.5 length

```
int ZipcodeBuffer::length  [private]
```

The length of the line in the CSV file.

Definition at line 28 of file ZipcodeBuffer.h.

3.2.5.6 longitude

```
double ZipcodeBuffer::longitude  [private]
```

Longitude as a double.

Definition at line 46 of file ZipcodeBuffer.h.

3.2.5.7 state

```
string ZipcodeBuffer::state  [private]
```

State abbreviation as a string.

Definition at line 37 of file ZipcodeBuffer.h.

3.2.5.8 zipcode

```
int ZipcodeBuffer::zipcode  [private]
```

Zipcode as an integer.

Definition at line 31 of file ZipcodeBuffer.h.

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

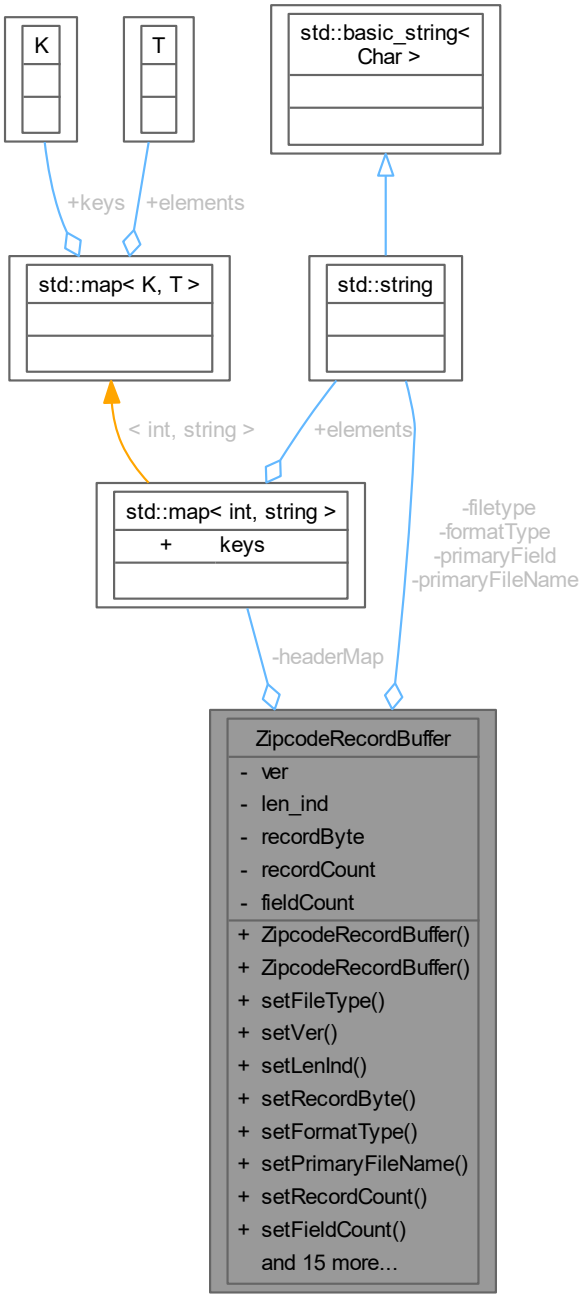
- ZipcodeBuffer.h
- ZipcodeBuffer.cpp

3.3 ZipcodeRecordBuffer Class Reference

Class to hold the information of a data file header record.

`#include <ZipcodeRecordBuffer.h>`

Collaboration diagram for ZipcodeRecordBuffer:



Public Member Functions

- ZipcodeRecordBuffer ()
  Default constructor.
- ZipcodeRecordBuffer (string fileName, string mainField)
  Parameterized constructor to initialize the object with given values.
- void setFileType (const string &FileType)
  Setter for fileType.
- void setVer (double Ver)
  Setter for ver.
- void setLenInd (int Len)
  Setter for len_ind.
- void setRecordByte (int RecordByte)
  Setter for recordByte.
- void setFormatType (const string &FormatType)
  Setter for formatType.
- void setPrimaryFileName (const string &FileName)
  Setter for primaryFileName.
- void setRecordCount (int RecordCount)
  Setter for recordCount.
- void setFieldCount (int FieldCount)
  Setter for fieldCount.
- void setPrimaryField (const string &PrimaryField)
  Setter for primaryField.
- string getFileType ()
  Getter for fileType.
- double getVer ()
  Getter for ver.
- int getLenInd ()
  Getter for len_ind.
- int getRecordByte ()
  Getter for recordByte.
- string getFormatType ()
  Getter for formatType.
- string getPrimaryFileName ()
  Getter for primaryFileName.
- int getRecordCount ()
  Getter for recordCount.
- int getFieldCount ()
  Getter for fieldCount.
- string getPrimaryField ()
  Getter for primaryField.
- string getFieldx (int position)
  Function to get the field specified by position.
- string getFieldType (int position)
  Function to get the field type of specified field.
- void getHeaderMap (const string &headerLine)
  Function to set the header map of the CSV file.
- void printHeaderMap ()
  Function to print the header of the CSV file.
- bool setHeaderMap (string fileName)
  Function to set the header of the CSV file.

Private Attributes

- string filetype
- double ver
- int len_ind
- int recordByte
- string formatType
- string primaryFileName
- int recordCount
- int fieldCount
- string primaryField
- map< int, string > headerMap

  Map to store the header of the CSV file to keep column sorting flexible.

Friends

- ostream & operator<< (ostream &out, const ZipcodeRecordBuffer &buffer)

  Overloaded output operator for ZipcodeRecordBuffer.

3.3.1 Detailed Description

Class to hold the information of a data file header record.

Definition at line 25 of file ZipcodeRecordBuffer.h.

3.3.2 Constructor & Destructor Documentation

3.3.2.1 ZipcodeRecordBuffer() [1/2]

```
ZipcodeRecordBuffer::ZipcodeRecordBuffer ( )
```

Default constructor.

Precondition

None

Postcondition

ZipcodeRecordBuffer object is created with default values

Definition at line 17 of file ZipcodeRecordBuffer.cpp.

3.3.2.2 ZipcodeRecordBuffer() [2/2]

```
ZipcodeRecordBuffer::ZipcodeRecordBuffer (
            string fileName,
            string mainField )
```

Parameterized constructor to initialize the object with given values.

Parameters

fileName	string representing the name of the CSV file
mainField	string representing the main field of the header record data file

Precondition

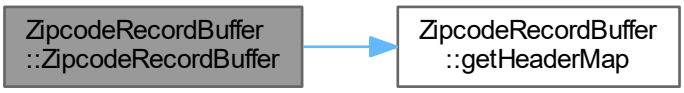
None

Postcondition

ZipcodeRecordBuffer object is created with given values and values pulled from the CSV file.

Definition at line 33 of file ZipcodeRecordBuffer.cpp.

Here is the call graph for this function:



3.3.3 Member Function Documentation

3.3.3.1 getFieldCount()

```
int ZipcodeRecordBuffer::getFieldCount ( )
```

Getter for fieldCount.

Precondition

ZipcodeRecordBuffer object must exist

Postcondition

None

Returns

Integer value of fieldCount

Definition at line 159 of file ZipcodeRecordBuffer.cpp.

3.3.3.2 getFieldType()

```
string ZipcodeRecordBuffer::getFieldType (
            int position )
```

Function to get the field type of specified field.

Parameters

position	integer representing the field number in the header

Precondition

ZipcodeRecordBuffer object must exist

Postcondition

None

Definition at line 174 of file ZipcodeRecordBuffer.cpp.

3.3.3.3 getFieldx()

```
string ZipcodeRecordBuffer::getFieldx (
            int position )
```

Function to get the field specified by position.

Parameters

position	integer representing the field number in the header

Precondition

ZipcodeRecordBuffer object must exist

Postcondition

None

Definition at line 169 of file ZipcodeRecordBuffer.cpp.

3.3.3.4 getFileType()

```
string ZipcodeRecordBuffer::getFileType ( )
```

Getter for fileType.

Precondition

ZipcodeRecordBuffer object must exist

Postcondition

None

Returns

String value of fileType

Definition at line 124 of file ZipcodeRecordBuffer.cpp.

3.3.3.5 getFormatType()

```
string ZipcodeRecordBuffer::getFormatType ( )
```

Getter for formatType.

Precondition

ZipcodeRecordBuffer object must exist

Postcondition

None

Returns

String value of formatType

Definition at line 144 of file ZipcodeRecordBuffer.cpp.

3.3.3.6 getHeaderMap()

```
void ZipcodeRecordBuffer::getHeaderMap (
            const string & headerLine )
```

Function to set the header map of the CSV file.

Parameters

headerLine	String representing the header line of the CSV file

Precondition

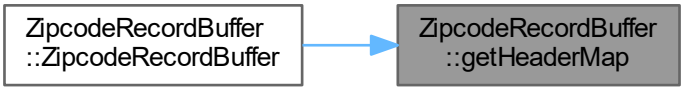
ZipcodeRecordBuffer object must exist

Postcondition

ZipcodeRecordBuffer object's headerMap is set to the values in the given string

Definition at line 191 of file ZipcodeRecordBuffer.cpp.

Here is the caller graph for this function:



3.3.3.7 getLenInd()

```
int ZipcodeRecordBuffer::getLenInd ( )
```

Getter for len_ind.

Precondition

ZipcodeRecordBuffer object must exist

Postcondition

None

Returns

Integer value of len_ind

Definition at line 134 of file ZipcodeRecordBuffer.cpp.

3.3.3.8 getPrimaryField()

```
string ZipcodeRecordBuffer::getPrimaryField ( )
```

Getter for primaryField.

Precondition

ZipcodeRecordBuffer object must exist

Postcondition

None

Returns

String value of primaryField

Definition at line 164 of file ZipcodeRecordBuffer.cpp.

3.3.3.9 getPrimaryFileName()

```
string ZipcodeRecordBuffer::getPrimaryFileName ( )
```

Getter for primaryFileName.

Precondition

ZipcodeRecordBuffer object must exist

Postcondition

None

Returns

String value of primaryFileName

Definition at line 149 of file ZipcodeRecordBuffer.cpp.

3.3.3.10 getRecordByte()

```
int ZipcodeRecordBuffer::getRecordByte ( )
```

Getter for recordByte.

Precondition

ZipcodeRecordBuffer object must exist

Postcondition

None

Returns

Integer value of recordByte

Definition at line 139 of file ZipcodeRecordBuffer.cpp.

3.3.3.11 getRecordCount()

```
int ZipcodeRecordBuffer::getRecordCount ( )
```

Getter for recordCount.

Precondition

ZipcodeRecordBuffer object must exist

Postcondition

None

Returns

Integer value of recordCount

Definition at line 154 of file ZipcodeRecordBuffer.cpp.

3.3.3.12 getVer()

```
double ZipcodeRecordBuffer::getVer ( )
```

Getter for ver.

Precondition

ZipcodeRecordBuffer object must exist

Postcondition

None

Returns

double value of ver

Definition at line 129 of file ZipcodeRecordBuffer.cpp.

3.3.3.13 printHeaderMap()

```
void ZipcodeRecordBuffer::printHeaderMap ( )
```

Function to print the header of the CSV file.

Precondition

ZipcodeRecordBuffer object must exist

Postcondition

None

Definition at line 314 of file ZipcodeRecordBuffer.cpp.

3.3.3.14 setFieldCount()

```
void ZipcodeRecordBuffer::setFieldCount (
            int FieldCount )
```

Setter for fieldCount.

Parameters

FieldCount	string value of fieldCount

Precondition

ZipcodeRecordBuffer object must exist

Postcondition

ZipcodeRecordBuffer object's fieldCount attribute is set to the given value

Definition at line 112 of file ZipcodeRecordBuffer.cpp.

3.3.3.15 setFileType()

```
void ZipcodeRecordBuffer::setFileType (
            const string & FileType )
```

Setter for fileType.

Parameters

FileType	string value of fileType

Precondition

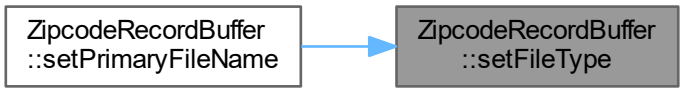
ZipcodeRecordBuffer object must exist

Postcondition

ZipcodeRecordBuffer object's fileType attribute is set to the given value

Definition at line 74 of file ZipcodeRecordBuffer.cpp.

Here is the caller graph for this function:



3.3.3.16 setFormatType()

```
void ZipcodeRecordBuffer::setFormatType (
            const string & FormatType )
```

Setter for formatType.

Parameters

FormatType	string value of formatType

Precondition

ZipcodeRecordBuffer object must exist

Postcondition

ZipcodeRecordBuffer object's formatType attribute is set to the given value

Definition at line 94 of file ZipcodeRecordBuffer.cpp.

3.3.3.17 setHeaderMap()

```
bool ZipcodeRecordBuffer::setHeaderMap (
            string fileName )
```

Function to set the header of the CSV file.

Parameters

fileName	String representing the name of the CSV file

Precondition

ZipcodeRecordBuffer object must exist

Postcondition

The CSV file with name fileName has its header rewritten to what what stored in the ZipcodeRecordBuffer

Definition at line 238 of file ZipcodeRecordBuffer.cpp.

3.3.3.18 setLenInd()

```
void ZipcodeRecordBuffer::setLenInd (
            int Len )
```

Setter for len_ind.

Parameters

Len	integer value of len_ind

Precondition

ZipcodeRecordBuffer object must exist

Postcondition

ZipcodeRecordBuffer object's len_ind attribute is set to the given value

Definition at line 84 of file ZipcodeRecordBuffer.cpp.

3.3.3.19 setPrimaryField()

```
void ZipcodeRecordBuffer::setPrimaryField (
            const string & PrimaryField )
```

Setter for primaryField.

Parameters

PrimaryField	string value of primaryField

Precondition

ZipcodeRecordBuffer object must exist

Postcondition

ZipcodeRecordBuffer object's primaryField attribute is set to the given value

Definition at line 117 of file ZipcodeRecordBuffer.cpp.

3.3.3.20 setPrimaryFileName()

```
void ZipcodeRecordBuffer::setPrimaryFileName (
            const string & FileName )
```

Setter for primaryFileName.

Parameters

FileName	string value of primaryFileName

Precondition

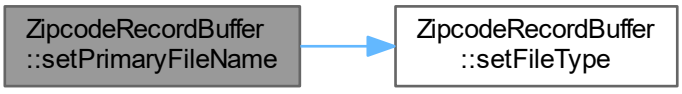
ZipcodeRecordBuffer object must exist

Postcondition

ZipcodeRecordBuffer object's primaryFileName attribute is set to the given value

Definition at line 99 of file ZipcodeRecordBuffer.cpp.

Here is the call graph for this function:



3.3.3.21 setRecordByte()

```
void ZipcodeRecordBuffer::setRecordByte (
            int RecordByte )
```

Setter for recordByte.

Parameters

RecordByte	integer value of recordByte

Precondition

ZipcodeRecordBuffer object must exist

Postcondition

ZipcodeRecordBuffer object's recordByte attribute is set to the given value

Definition at line 89 of file ZipcodeRecordBuffer.cpp.

3.3.3.22 setRecordCount()

```
void ZipcodeRecordBuffer::setRecordCount (
            int RecordCount )
```

Setter for recordCount.

Parameters

RecordCount	string value of recordCount

Precondition

ZipcodeRecordBuffer object must exist

Postcondition

ZipcodeRecordBuffer object's recordCount attribute is set to the given value

Definition at line 107 of file ZipcodeRecordBuffer.cpp.

3.3.3.23 setVer()

```
void ZipcodeRecordBuffer::setVer (
            double Ver )
```

Setter for ver.

Parameters

Ver	double value of ver

Precondition

ZipcodeRecordBuffer object must exist

Postcondition

ZipcodeRecordBuffer object's ver attribute is set to the given value

Definition at line 79 of file ZipcodeRecordBuffer.cpp.

3.3.4 Friends And Related Symbol Documentation

3.3.4.1 operator<<

```
ostream & operator<< (
            ostream & out,
            const ZipcodeRecordBuffer & buffer )  [friend]
```

Overloaded output operator for ZipcodeRecordBuffer.

Parameters

out	Output stream
a	ZipcodeRecordBuffer to display

Precondition

None

Postcondition

The the header record is displayed

Returns

Updated output stream

Definition at line 338 of file ZipcodeRecordBuffer.cpp.

3.3.5 Member Data Documentation

3.3.5.1 fieldCount

```
int ZipcodeRecordBuffer::fieldCount  [private]
```

Definition at line 50 of file ZipcodeRecordBuffer.h.

3.3.5.2 filetype

```
string ZipcodeRecordBuffer::filetype  [private]
```

Definition at line 29 of file ZipcodeRecordBuffer.h.

3.3.5.3 formatType

```
string ZipcodeRecordBuffer::formatType  [private]
```

Definition at line 41 of file ZipcodeRecordBuffer.h.

3.3.5.4 headerMap

```
map<int, string> ZipcodeRecordBuffer::headerMap  [private]
```

Map to store the header of the CSV file to keep column sorting flexible.

Definition at line 56 of file ZipcodeRecordBuffer.h.

3.3.5.5 len_ind

```
int ZipcodeRecordBuffer::len_ind  [private]
```

Definition at line 35 of file ZipcodeRecordBuffer.h.

3.3.5.6 primaryField

```
string ZipcodeRecordBuffer::primaryField  [private]
```

Definition at line 53 of file ZipcodeRecordBuffer.h.

3.3.5.7 primaryFileName

```
string ZipcodeRecordBuffer::primaryFileName  [private]
```

Definition at line 44 of file ZipcodeRecordBuffer.h.

3.3.5.8 recordByte

```
int ZipcodeRecordBuffer::recordByte  [private]
```

Definition at line 38 of file ZipcodeRecordBuffer.h.

3.3.5.9 recordCount

```
int ZipcodeRecordBuffer::recordCount  [private]
```

Definition at line 47 of file ZipcodeRecordBuffer.h.

3.3.5.10 ver

```
double ZipcodeRecordBuffer::ver  [private]
```

Definition at line 32 of file ZipcodeRecordBuffer.h.

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

- ZipcodeRecordBuffer.h
- ZipcodeRecordBuffer.cpp

Chapter 4

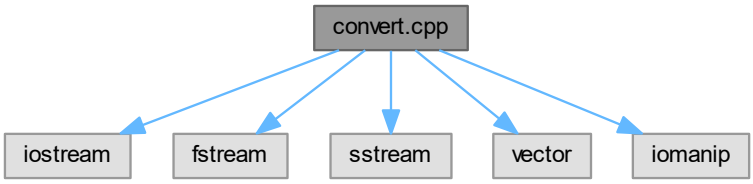
File Documentation

4.1 convert.cpp File Reference

Convert CSV data into length-indicated format.

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

Include dependency graph for convert.cpp:



Functions

- std::string convertToLengthIndicated (const std::string &line)

  Converts a given string to a length-indicated format.
- int main ()

  Main function to process the CSV data.

4.1.1 Detailed Description

Convert CSV data into length-indicated format.

Author

Andrew Clayton

Definition in file convert.cpp.

4.1.2 Function Documentation

4.1.2.1 convertToLengthIndicated()

```
std::string convertToLengthIndicated (
            const std::string & line )
```

Converts a given string to a length-indicated format.

If the length of the provided string is more than 99 characters, an error is output and the program exits. Otherwise, the string is prepended with its 2-digit length.

Parameters

line	String to be converted.

Returns

Length-indicated formatted string.

Definition at line 23 of file convert.cpp.

Here is the caller graph for this function:

4.1.2.2 main()

```
int main ( )
```

Main function to process the CSV data.

Reads from "us_postal_codes_ROWS_RANDOMIZED.csv", converts each line to length-indicated format and writes the result to "LI_R.csv".

Returns

Returns 0 on success, 1 on file open error.

Definition at line 48 of file convert.cpp.

Here is the call graph for this function:

4.2 convert.cpp

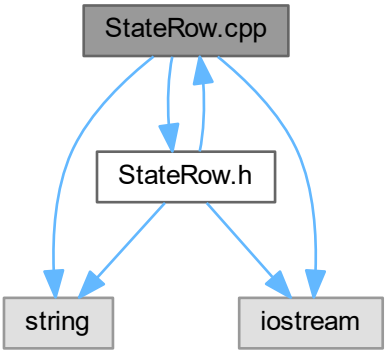
Go to the documentation of this file.

```
00001
00007 #include <iostream>
00008 #include <fstream>
00009 #include <sstream>
00010 #include <vector>
00011 #include <iomanip>
00012
00023 std::string convertToLengthIndicated(const std::string &line) {
00024     // Calculate the total length of the string excluding the length field
00025     int totalLength = line.size();
00026
00027     if (totalLength > 99) {
00028         std::cerr « "Error: Record too long to fit in 2-digit length field." « std::endl;
00029         exit(1);
00030     }
00031
00032     // Convert the length to a string with leading zeros (2-digit width)
00033     std::ostringstream lengthStream;
00034     lengthStream « std::setw(2) « std::setfill('0') « totalLength;
00035
00036     // Return the length-indicated string
00037     return lengthStream.str() + line;
00038 }
00039
00048 int main() {
00049     std::ifstream inFile("us_postal_codes_ROWS_RANDOMIZED.csv");
00050     std::ofstream outFile("LI_R.csv");
00051
00052     if (!inFile.is_open() || !outFile.is_open()) {
00053         std::cerr « "Error opening files." « std::endl;
00054         return 1;
00055     }
00056
00057     std::string line;
00058     // Skip header line
00059     std::getline(inFile, line);
00060
00061     while (std::getline(inFile, line)) {
00062         // Convert to length-indicated format
00063         std::string convertedLine = convertToLengthIndicated(line);
00064
00065         // Write to the output file
00066         outFile « convertedLine « std::endl;
00067     }
00068
00069     inFile.close();
00070     outFile.close();
00071     return 0;
00072 }
```

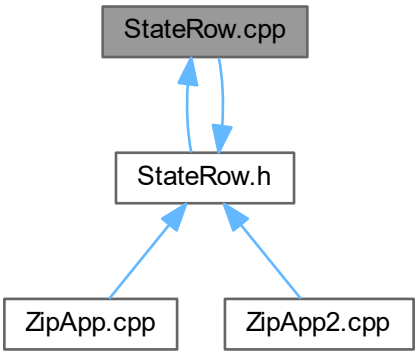
4.3 StateRow.cpp File Reference

```
#include "StateRow.h"
#include <string>
#include <iostream>
```

Include dependency graph for StateRow.cpp:



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



Functions

- ostream & operator<< (ostream &out, const StateRow &row)

  Overloaded operator to print StateRow objects.

4.3.1 Function Documentation

4.3.1.1 operator<<()

```
ostream & operator<< (
            ostream & out,
            const StateRow & row )
```

Overloaded operator to print StateRow objects.

Parameters

out	Output stream.
row	StateRow object to print.

Returns

Reference to the output stream.

Definition at line 221 of file StateRow.cpp.

4.4 StateRow.cpp

Go to the documentation of this file.

```
00001 // Authors: Tristan Adams and Preston Betz
00002
00003 #include "StateRow.h"
00004
00005 #include <string> // for the states ID
00006 #include <iostream>
00007
00008 using namespace std;
00009
00019 StateRow::StateRow(){
00020     fresh = true;
00021     ID = "TEMP";
00022     eastmost = 00.00;
00023     westmost = 00.00;
00024     northmost = 00.00;
00025     southmost = 00.00;
00026     eastmostZ = 0;
00027     westmostZ = 0;
00028     northmostZ = 0;
00029     southmostZ = 0;
00030 }
00031
00041 StateRow::StateRow(string id, double e, double w, double n, double s, int eZ, int wZ, int nZ, int sZ){
00042     this->fresh = true;
00043     this->ID = id;
00044     this->eastmost = e;
00045     this->westmost = w;
00046     this->northmost = n;
00047     this->southmost = s;
00048     this->eastmostZ = eZ;
00049     this->westmostZ = wZ;
00050     this->northmostZ = nZ;
00051     this->southmostZ = sZ;
00052 }
00053
00058 bool StateRow::getFresh() const {
00059     return fresh;
00060 }
00061
00066 double StateRow::getEast(){
00067     return eastmost;
00068 }
```

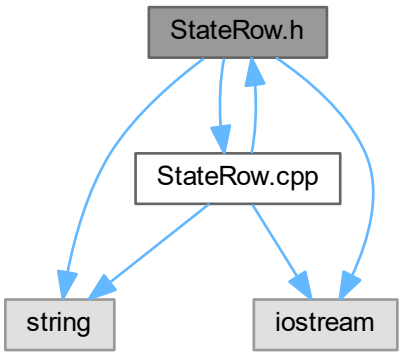
```
00069
00074 double StateRow::getWest(){
00075     return westmost;
00076 }
00077
00082 double StateRow::getNorth(){
00083     return northmost;
00084 }
00085
00090 double StateRow::getSouth(){
00091     return southmost;
00092 }
00093
00098 int StateRow::getEastZ(){
00099     return eastmostZ;
00100 }
00101
00106 int StateRow::getWestZ(){
00107     return westmostZ;
00108 }
00109
00114 int StateRow::getNorthZ(){
00115     return northmostZ;
00116 }
00117
00122 int StateRow::getSouthZ(){
00123     return southmostZ;
00124 }
00125
00130 string StateRow::getID(){
00131     return ID;
00132 }
00133
00138 void StateRow::setEast(double e){
00139     this->eastmost = e;
00140 }
00141
00146 void StateRow::setWest(double w){
00147     this->westmost = w;
00148 }
00149
00154 void StateRow::setNorth(double n){
00155     this->northmost = n;
00156 }
00157
00162 void StateRow::setSouth(double s){
00163     this->southmost = s;
00164 }
00165
00170 void StateRow::setEastZ(int eastZ){
00171     this->eastmostZ = eastZ;
00172 }
00173
00178 void StateRow::setWestZ(int westZ){
00179     this->westmostZ = westZ;
00180 }
00181
00186 void StateRow::setNorthZ(int northZ){
00187     this->northmostZ = northZ;
00188 }
00189
00194 void StateRow::setSouthZ(int southZ){
00195     this->southmostZ = southZ;
00196 }
00197
00202 void StateRow::setID(string id){
00203     this->ID = id;
00204 }
00205
00210 void StateRow::setFresh(bool b){
00211     this->fresh = b;
00212 }
00213
00214
00221 ostream& operator«(ostream& out, const StateRow& row){
00222     out « row.ID « " | "
00223         « row.eastmostZ « " | "
00224         « row.westmostZ « " | "
00225         « row.northmostZ « " | "
00226         « row.southmostZ;
00227     return out;
00228 }
```

4.5 StateRow.h File Reference

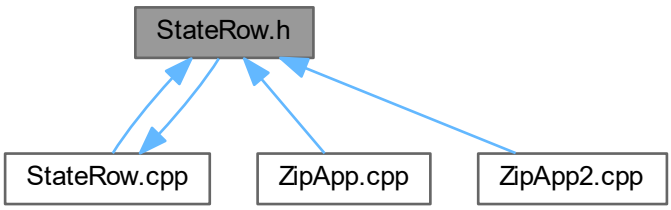
Declaration of the StateRow class.

```
#include <string>
#include <iostream>
#include "StateRow.cpp"
```

Include dependency graph for StateRow.h:



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



Classes

- class StateRow

  A class representing a state with geographic data.

4.5.1 Detailed Description

Declaration of the StateRow class.

Authors

Tristan Adams and Preston Betz

Definition in file StateRow.h.

4.6 StateRow.h

Go to the documentation of this file.

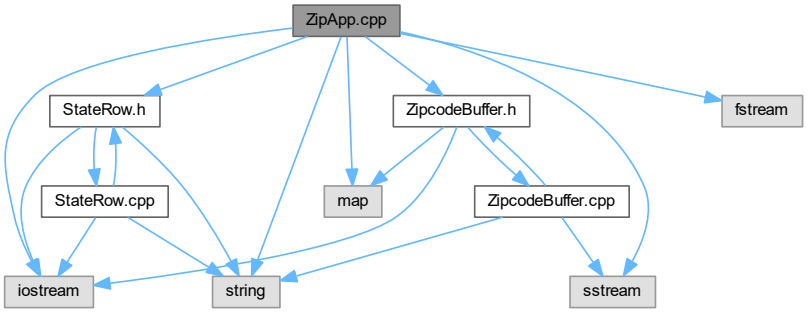
```
00001
00007 #ifndef STATEROW_H
00008 #define STATEROW_H
00009
00010 #include <string> // for the states ID
00011 #include <iostream>
00012
00013 using namespace std;
00014
00019 class StateRow {
00020 private:
00021     bool fresh;
00022     string ID;
00023     double eastmost;
00024     double westmost;
00025     double northmost;
00026     double southmost;
00027
00028     int eastmostZ;
00029     int westmostZ;
00030     int northmostZ;
00031     int southmostZ;
00032
00033 public:
00043     StateRow();
00044
00054     StateRow(string id, double e, double w, double n, double s, int eZ, int wZ, int nZ, int sZ);
00055
00060     string getID();
00061
00062     bool getFresh() const;
00063     // Getter methods
00064     double getEast();
00065     double getWest();
00066     double getNorth();
00067     double getSouth();
00068
00069
00070     int getEastZ();
00071     int getWestZ();
00072     int getNorthZ();
00073     int getSouthZ();
00074
00075     // Setter methods
00076     void setFresh(bool b);
00077     void setID(string id);
00078     void setEast(double east);
00079     void setWest(double west);
00080     void setNorth(double north);
00081     void setSouth(double south);
00082
00083     void setEastZ(int eastZ);
00084     void setWestZ(int westZ);
00085     void setNorthZ(int northZ);
00086     void setSouthZ(int southZ);
00087
00094     friend ostream& operator<<(ostream& out, const StateRow& row);
00095 };
00096 #include "StateRow.cpp"
00097 #endif
```

4.7 ZipApp.cpp File Reference

```
#include <iostream>
#include <map>
#include "ZipcodeBuffer.h"
#include "StateRow.h"
#include <sstream>
#include <string>
```

```
#include <fstream>
```

Include dependency graph for ZipApp.cpp:



Functions

- int main ()

  Project 2 Part 2: Main function to populate and display a table of StateRow objects using a map.

4.7.1 Function Documentation

4.7.1.1 main()

```
int main ( )
```

Project 2 Part 2: Main function to populate and display a table of StateRow objects using a map.

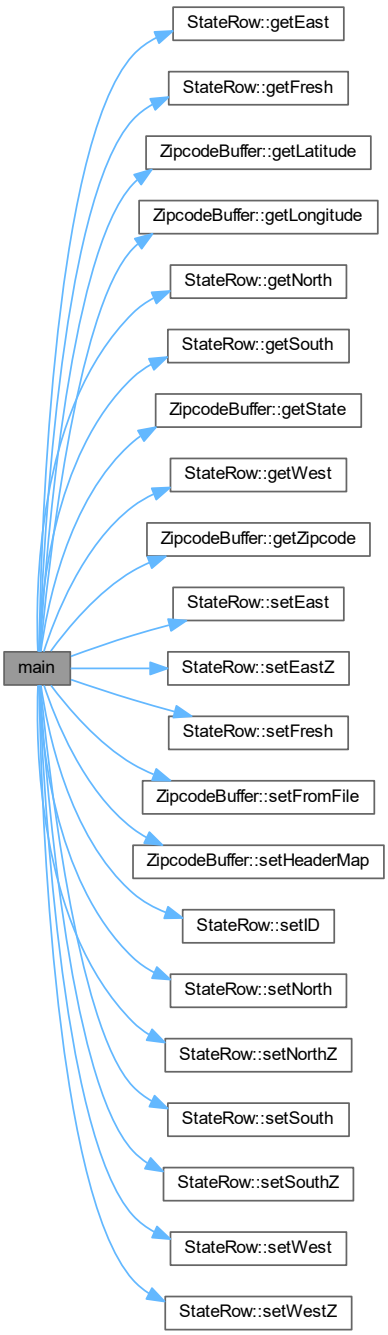
The main function reads from a CSV file, populates a map with StateRow objects indexed by state ID, and displays the final table of states with their easternmost, westernmost, northernmost, and southernmost zip codes.

Returns

0 on successful execution, -1 on error.

Definition at line 21 of file ZipApp.cpp.

Here is the call graph for this function:



4.8 ZipApp.cpp

Go to the documentation of this file.

```
00001 #include <iostream>
00002 #include <map>
00003 #include "ZipcodeBuffer.h"
00004 #include "StateRow.h"
```

```
00005 #include <sstream> // for string stream
00006 #include <string> // for getline
00007 #include <fstream> // for file reading
00008
00009 // Authors: Tristan Adams, Andrew Clayton, Preston Betz, and Zachary Sunder
00010 using namespace std;
00011
00021 int main() {
00022     // Makes a hashmap to hold state IDs
00023     map<string, StateRow> stateMap;
00024
00025     // Buffer to extract CSV data
00026     ZipcodeBuffer zipHolder;
00027     // ifstream inFile("us_postal_codes_ROWS_RANDOMIZED.csv");
00028     ifstream inFile("Length_R.csv");
00029     if (!inFile) {
00030         cout « "Error opening file" « endl;
00031         return -1;
00032     }
00033
00034     bool first = true;
00035     string line;
00036     while (getline(inFile, line)) {
00037         // Skip header line
00038         if(first) {
00039             zipHolder.setHeaderMap(line);
00040             first = false;
00041             continue;
00042         }
00043
00044         // Set values from the current CSV line
00045         zipHolder.setFromFile(line);
00046         string stateId = zipHolder.getState();
00047
00048         // Initialize or get reference to the StateRow object for the current state
00049         StateRow& currentRow = stateMap[stateId];
00050
00051         // Always apply comparisons, whether the state is encountered for the first time or not
00052         if (currentRow.getEast() > zipHolder.getLongitude() || currentRow.getFresh()) {
00053             currentRow.setID(stateId);
00054             currentRow.setEast(zipHolder.getLongitude());
00055             currentRow.setEastZ(zipHolder.getZipcode());
00056         }
00057         if (currentRow.getWest() < zipHolder.getLongitude() || currentRow.getFresh()) {
00058             currentRow.setWest(zipHolder.getLongitude());
00059             currentRow.setWestZ(zipHolder.getZipcode());
00060         }
00061         if (currentRow.getNorth() < zipHolder.getLatitude() || currentRow.getFresh()) {
00062             currentRow.setNorth(zipHolder.getLatitude());
00063             currentRow.setNorthZ(zipHolder.getZipcode());
00064         }
00065         if (currentRow.getSouth() > zipHolder.getLatitude() || currentRow.getFresh()) {
00066             currentRow.setSouth(zipHolder.getLatitude());
00067             currentRow.setSouthZ(zipHolder.getZipcode());
00068         }
00069
00070         // Update the fresh flag after processing the entry
00071         currentRow.setFresh(false);
00072     }
00073
00074     // Display the final table
00075     cout « "StateID, East Zip, West Zip, North Zip, South Zip" « endl;
00076     for (const auto& pair : stateMap) {
00077         if(!pair.second.getFresh()) {
00078             cout « pair.second « endl;
00079         }
00080     }
00081
00082     return 0;
00083 }
```

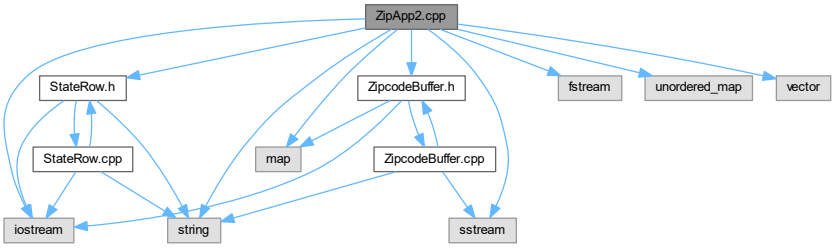
4.9 ZipApp2.cpp File Reference

This is the file for Project 2 Part 2, giving zipcode information given a zipcode in the command line arguments.

```
#include <iostream>
#include <map>
#include "ZipcodeBuffer.h"
#include "StateRow.h"
```

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

Include dependency graph for ZipApp2.cpp:



Functions

- int main (int argc, char ∗argv[ ])

4.9.1 Detailed Description

This is the file for Project 2 Part 2, giving zipcode information given a zipcode in the command line arguments.

Author

Andrew Clayton and Tristan Adams

Definition in file ZipApp2.cpp.

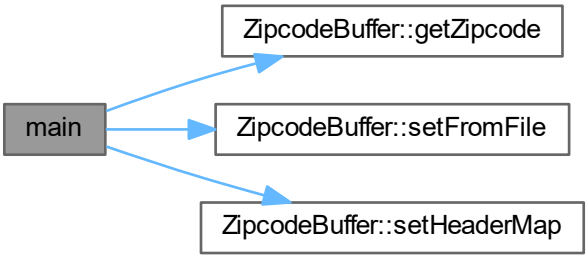
4.9.2 Function Documentation

4.9.2.1 main()

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

Definition at line 19 of file ZipApp2.cpp.

Here is the call graph for this function:



4.10 ZipApp2.cpp

Go to the documentation of this file.

```
00001 #include <iostream>
00002 #include <map>
00003 #include "ZipcodeBuffer.h"
00004 #include "StateRow.h"
00005 #include <sstream> // for string stream
00006 #include <string> // for getline
00007 #include <fstream> // for file reading
00008 #include <unordered_map>
00009 #include <vector>
00016 // Authors: Tristan Adams and Andrew Clayton
00017 using namespace std;
00018
00019 int main(int argc, char* argv[]) {
00020     // Makes a hashmap to hold zipcodes and the RRN (relative reference number) that go with them
00021     unordered_map<int, long> indexMap;
00022     unordered_map<int, ZipcodeBuffer> zipMap;
00023
00024     // Buffer to extract CSV data
00025     ZipcodeBuffer zipHolder;
00026
00027     // ifstream inFile("us_postal_codes_ROWS_RANDOMIZED.csv");
00028     ifstream inFile("Length_R.csv");
00029     if (!inFile) {
00030         cout « "Error opening file" « endl;
00031         return -1;
00032     }
00033
00034     bool first = true;
00035     string line;
00036
00037     // Creating new file for Index file (assuming there is not already one)
00038     // Open an output file stream to write the index
00039     ofstream indexFile("indexFile.txt");
00040     if (!indexFile) {
00041         cout « "Error: Unable to open index file for writing." « endl;
00042         return -1;  // This might need adjustment based on your main() function structure
00043     }
00044
00045     long rrn = inFile.tellg();  // Start by getting the current position of the file
00046
00047     while (getline(inFile, line)) {
00048         // If it's the first line, consider it as a header and skip
00049         if (first) {
00050             zipHolder.setHeaderMap(line);
00051             first = false;
00052             continue;
00053         }
00054
00055         // Populate the ZipcodeBuffer from the line
00056         zipHolder.setFromFile(line);
00057         zipMap[zipHolder.getZipcode()] = zipHolder;
```

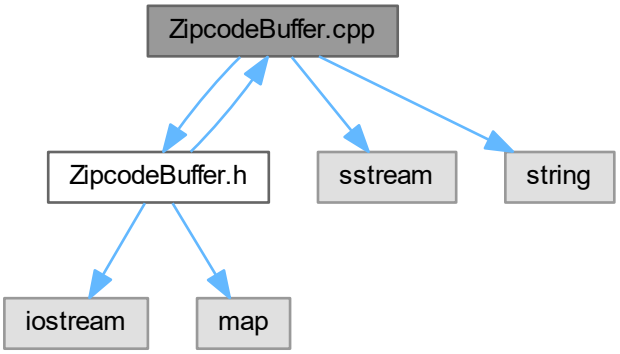
```
00058
00059         // Store the current rrn into the hashmap
00060         indexMap[zipHolder.getZipcode()] = rrn;
00061
00062         // Write the zipcode and its rrn to the index file
00063         indexFile « zipHolder.getZipcode() « " " « rrn « endl;
00064
00065         // Update rrn to point to the start of the next line/record
00066         rrn = inFile.tellg();
00067     }
00068
00069     // Display whatever zipcodes are indicated in the command line arguments
00070     // also make sure that command line arguments are accommodated
00071     if (argc <= 1) {
00072         cout « "No zipcodes given\n";
00073     } else {
00074        vector<int> zipCodeIntegers;
00075         int count = 0;
00076         for (int i = 1; i < argc; ++i) {
00077             try {
00078                 // Convert each command line argument (C-string) to an integer using atoi
00079              int intValue = atoi(argv[i]);
00080
00081              // Add the integer to the vector
00082                 zipCodeIntegers.push_back(intValue);
00083                 count++;
00084             } catch (const invalid_argument& e) {
00085                 cerr « "Invalid argument: " « e.what() « endl;
00086             }
00087         }
00088         //once the vector is made we can search through the whole vector and display the record that
     the zipcode grabs from the index
00089         for (int i = 0; i < count; ++i) {
00090
00091             if (zipMap.find(zipCodeIntegers[i]) != zipMap.end()) {
00092                 cout « zipMap[zipCodeIntegers[i]] « endl;
00093             } else {
00094                 cout « "Zipcode not found.\n";
00095             }
00096            }
00097     }
00098
00099     return 0;
00100 }
```

4.11 ZipcodeBuffer.cpp File Reference

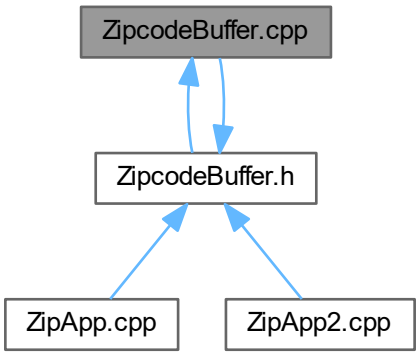
Implementation file for ZipcodeBuffer class.

```
#include "ZipcodeBuffer.h"
#include <sstream>
#include <string>
```

Include dependency graph for ZipcodeBuffer.cpp:



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



Functions

- istream & operator>> (istream &in, ZipcodeBuffer &buffer)
- ostream & operator<< (ostream &out, const ZipcodeBuffer &buffer)

4.11.1 Detailed Description

Implementation file for ZipcodeBuffer class.

Author

Andrew Clayton

Version

1.4

Definition in file ZipcodeBuffer.cpp.

4.11.2 Function Documentation

4.11.2.1 operator<<()

```
ostream & operator<< (
            ostream & out,
            const ZipcodeBuffer & buffer )
```

Parameters

out	Output stream
a	ZipcodeBuffer to display

Precondition

None

Postcondition

The first a.size elements of a.ptr are displayed to output

Returns

Updated output stream

Definition at line 162 of file ZipcodeBuffer.cpp.

4.11.2.2 operator>>()

```
istream & operator>> (
            istream & in,
            ZipcodeBuffer & buffer )
```

Parameters

in	Input stream
a	ZipcodeBuffer to fill

Precondition

A ZipcodeBuffer object must exist

Postcondition

The first a.size elements of a.ptr are filled with integers read from input

Returns

Updated input stream

Definition at line 154 of file ZipcodeBuffer.cpp.

4.12 ZipcodeBuffer.cpp

Go to the documentation of this file.

```
00001
00009 #include "ZipcodeBuffer.h"
00010 #include <sstream> //For stringstream
00011 #include <string> //For getline
00012
00013 using namespace std;
00014
00015 //Default constructor
00016 ZipcodeBuffer::ZipcodeBuffer() {
00017     zipcode = 0;
00018     city = "";
00019     state = "";
00020     county = "";
00021     latitude = 0.0;
00022     longitude = 0.0;
00023 }
00024
00025 //Constructor
00026 ZipcodeBuffer::ZipcodeBuffer(int zipcode, string city, string state, string county, double latitude,
      double longitude) {
00027     this->zipcode = zipcode;
00028     this->city = city;
00029     this->state = state;
00030     this->county = county;
00031     this->latitude = latitude;
00032     this->longitude = longitude;
00033 }
00034
00035
00036 //Getters
00037 int ZipcodeBuffer::getZipcode() const {
00038     return zipcode;
00039 }
00040
00041 string ZipcodeBuffer::getCity() const {
00042     return city;
00043 }
00044
00045 string ZipcodeBuffer::getState() const {
00046     return state;
00047 }
00048
00049 string ZipcodeBuffer::getCounty() const {
00050     return county;
00051 }
00052
00053 double ZipcodeBuffer::getLatitude() const {
00054     return latitude;
00055 }
00056
00057 double ZipcodeBuffer::getLongitude() const {
00058     return longitude;
00059 }
00060
00061 //Setters
00062 void ZipcodeBuffer::setZipcode(int zipcode) {
```

```
00063     this->zipcode = zipcode;
00064 }
00065
00066 void ZipcodeBuffer::setCity(string city) {
00067     this->city = city;
00068 }
00069
00070 void ZipcodeBuffer::setState(string state) {
00071     this->state = state;
00072 }
00073
00074 void ZipcodeBuffer::setCounty(string county) {
00075     this->county = county;
00076 }
00077
00078 void ZipcodeBuffer::setLatitude(double latitude) {
00079     this->latitude = latitude;
00080 }
00081
00082 void ZipcodeBuffer::setLongitude(double longitude) {
00083     this->longitude = longitude;
00084 }
00085
00086 //Other functions
00087
00088 void ZipcodeBuffer::setFromFile(string fileLine) {
00089     // Extract the first two characters and convert them to an integer
00090     if (fileLine.size() >= 2) {
00091         length = std::stoi(fileLine.substr(0, 2));
00092     } else {
00093         // Handle the case where the fileLine is shorter than 2 characters
00094         // For now, I'll set length to 0, but you may want to handle this differently
00095         length = 0;
00096     }
00097
00098     // Picks up after the first two characters (after the length indication)
00099     stringstream ss(fileLine.substr(2));
00100     string field;
00101     int pos = 0;
00102
00103     while (getline(ss, field, ',')) {
00104         string columnName = headerMap[pos];
00105
00106         if (columnName == "ZipCode") {
00107             zipcode = std::stoi(field);
00108         } else if (columnName == "PlaceName") {
00109             city = field;
00110         } else if (columnName == "State") {
00111             state = field;
00112         } else if (columnName == "County") {
00113             county = field;
00114         } else if (columnName == "Lat") {
00115             latitude = std::stod(field);
00116         } else if (columnName == "Long") {
00117             longitude = std::stod(field);
00118         }
00119         // You can add more else-if conditions if there are other columns in the CSV.
00120
00121         pos++;
00122     }
00123 }
00124
00125
00126 void ZipcodeBuffer::setHeaderMap(const string& headerLine) {
00127     // Picks up after the length indication
00128     stringstream ss(headerLine.substr(2));
00129     string field;
00130     int pos = 0;
00131
00132     while(getline(ss, field, ',')) {
00133         // If the CSV contains quoted fields, remove the quotes.
00134         if (field.front() == '"' && field.back() == '"') {
00135             field = field.substr(1, field.size() - 2);
00136         }
00137
00138         headerMap[pos] = field;
00139         pos++;
00140     }
00141 }
00142
00143 int ZipcodeBuffer::getLength() const {
00144     return length;
00145 }
00146
00147 void ZipcodeBuffer::setLength(int len) {
00148     length = len;
00149 }
```

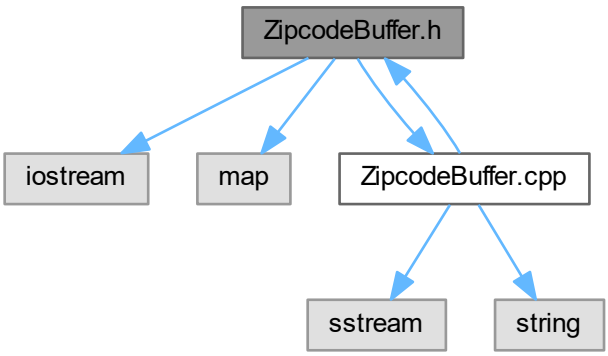
```
00150
00151
00152 //OVERLOADED OPERATORS
00153 //Overloaded input operator
00154 istream& operator»(istream& in, ZipcodeBuffer& buffer) {
00155     string fileLine;
00156     getline(in, fileLine);
00157     buffer.setFromFile(fileLine);
00158     return in;
00159 }
00160
00161 //Overloaded output operator
00162 ostream& operator«(ostream& out, const ZipcodeBuffer& buffer) {
00163     out « buffer.zipcode « ", " « buffer.city « ", " « buffer.state « ", " « buffer.county « ", " «
      buffer.latitude « ", " « buffer.longitude;
00164     return out;
00165 }
00166
```

4.13 ZipcodeBuffer.h File Reference

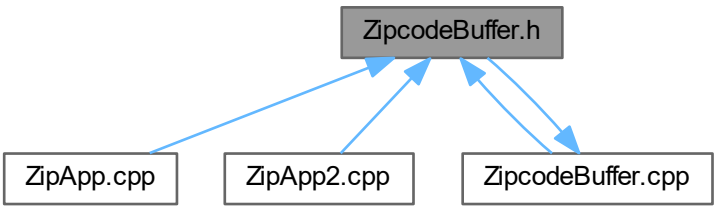
This header file defines the ZipcodeBuffer class, which is used to take in and store data from a Zipcode CSV file.

```
#include <iostream>
#include <map>
#include "ZipcodeBuffer.cpp"
```

Include dependency graph for ZipcodeBuffer.h:



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



Classes

- class ZipcodeBuffer

  Class to represent a Zipcode and its related attributes.

4.13.1 Detailed Description

This header file defines the ZipcodeBuffer class, which is used to take in and store data from a Zipcode CSV file.

Author

Andrew Clayton

Version

1.4

ZipcodeBuffer class: File to work with Zipcode CSV file

- Function to take in a line of the CSV file as a string and set the attributes of the object
- Overloaded input and output operators

Note

- A valid CSV line must be provided to set the attributes correctly

Definition in file ZipcodeBuffer.h.

4.14 ZipcodeBuffer.h

Go to the documentation of this file.

```
00001
00014 #ifndef ZIPCODEBUFFER_H
00015 #define ZIPCODEBUFFER_H
00016
00017 #include <iostream>
00018 #include <map>
00019 using namespace std;
00020
00025 class ZipcodeBuffer {
00026     private:
00028         int length;
00029
00031         int zipcode;
00032
00034         string city;
00035
00037         string state;
00038
00040         string county;
00041
00043         double latitude;
00044
00046         double longitude;
00047
00049         map<int, string> headerMap;
00050
00051
00052     public:
00058         ZipcodeBuffer();
00059
```

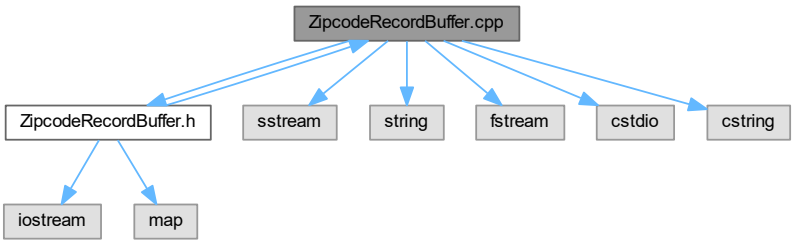
```
00071         ZipcodeBuffer(int zipcode, string city, string state, string county, double latitude, double
      longitude);
00072
00073
00074         //Getters
00075
00082         int getZipcode() const;
00083
00090         string getCity() const;
00091
00098         string getState() const;
00099
00106         string getCounty() const;
00107
00114         double getLatitude() const;
00115
00122         double getLongitude() const;
00123
00130         int getLength() const;
00131
00132
00133
00134         //Setters
00141         void setZipcode(int zipcode);
00142
00149         void setCity(string city);
00150
00157         void setState(string state);
00158
00165         void setCounty(string county);
00166
00173         void setLatitude(double latitude);
00174
00181         void setLongitude(double longitude);
00182
00189         void setFromFile(string fileLine);
00190
00197         void setLength(int length);
00198
00199
00200
00207         void setHeaderMap(const string& headerLine);
00208
00209
00210         //OVERLOADED OPERATORS
00219         friend istream& operator»(istream& in, ZipcodeBuffer& buffer);
00220
00229         friend ostream& operator«(ostream& out, const ZipcodeBuffer& buffer);
00230
00231
00232 };
00233
00234 #include "ZipcodeBuffer.cpp"
00235
00236 #endif
```

4.15 ZipcodeRecordBuffer.cpp File Reference

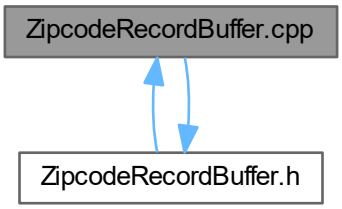
Implementation file for ZipcodeRecordBuffer class.

```
#include "ZipcodeRecordBuffer.h"
#include <sstream>
#include <string>
#include <fstream>
#include <cstdio>
#include <cstring>
```

Include dependency graph for ZipcodeRecordBuffer.cpp:



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



Functions

- ostream & operator<< (ostream &out, const ZipcodeRecordBuffer &buffer)

4.15.1 Detailed Description

Implementation file for ZipcodeRecordBuffer class.

Author

Zachary Sunder

Version

1.1

Definition in file ZipcodeRecordBuffer.cpp.

4.15.2 Function Documentation

4.15.2.1 operator<<()

```
ostream & operator<< (
            ostream & out,
            const ZipcodeRecordBuffer & buffer )
```

Parameters

out	Output stream
a	ZipcodeRecordBuffer to display

Precondition

None

Postcondition

The the header record is displayed

Returns

Updated output stream

Definition at line 338 of file ZipcodeRecordBuffer.cpp.

4.16 ZipcodeRecordBuffer.cpp

Go to the documentation of this file.

```
00001
00008 #include "ZipcodeRecordBuffer.h"
00009 #include <sstream> //For stringstream
00010 #include <string> //For getline
00011 #include <fstream> //for file processing
00012 #include <cstdio> //for updating the csv file
00013 #include <cstring> //convert string to char array
00014
00015
00016 //Default constructor
00017 ZipcodeRecordBuffer::ZipcodeRecordBuffer()
00018 {
00019     filetype = "";
00020     ver = 1.1;
00021     len_ind = 0;
00022     recordByte = 2;
00023     formatType = "ASCII";
00024     primaryFileName = "";
00025     recordCount = 0;
00026     fieldCount = 0;
00027     primaryField = "";
00028     headerMap.clear();
00029 }
00030
00031
00032 //Constructor
00033 ZipcodeRecordBuffer::ZipcodeRecordBuffer(string fileName, string mainField)
00034 {
00035     int pos = fileName.find(".");
00036     filetype = fileName.substr(pos);
00037     ver = 1.1;
00038     recordByte = 2;
00039     formatType = "ASCII";
00040     primaryFileName = fileName;
00041     primaryField = mainField;
00042
00043     ifstream inFile(fileName);
00044     string line;
00045
00046     if (!inFile)
00047     {
00048         cout « "Bad file or file name" « endl;
00049     }
00050
00051     bool first = true;
```

```
00052     int count = 0;
00053
00054     // Set the new header of the csv to newHeader and the rest stays the same
00055     while (getline(inFile, line))
00056     {
00057         if(first)
00058         {
00059             getHeaderMap(line);
00060             first = false;
00061             continue;
00062         }
00063         count++;
00064     }
00065     recordCount = count;
00066
00067
00068     len_ind = stoi(headerMap[0]);
00069     fieldCount = headerMap.size()-1;
00070 }
00071
00072
00073 //Setters
00074 void ZipcodeRecordBuffer::setFileType(const string& FileType)
00075 {
00076     filetype = FileType;
00077 }
00078
00079 void ZipcodeRecordBuffer::setVer(double Ver)
00080 {
00081     ver = Ver;
00082 }
00083
00084 void ZipcodeRecordBuffer::setLenInd(int Len)
00085 {
00086     len_ind = Len;
00087 }
00088
00089 void ZipcodeRecordBuffer::setRecordByte(int RecordByte)
00090 {
00091     recordByte = RecordByte;
00092 }
00093
00094 void ZipcodeRecordBuffer::setFormatType(const string& FormatType)
00095 {
00096     formatType = FormatType;
00097 }
00098
00099 void ZipcodeRecordBuffer::setPrimaryFileName(const string& FileName)
00100 {
00101     primaryFileName = FileName;
00102     int pos = FileName.find(".");
00103     string FileType = FileName.substr(pos);
00104     setFileType(FileType);
00105 }
00106
00107 void ZipcodeRecordBuffer::setRecordCount(int RecordCount)
00108 {
00109     recordCount = RecordCount;
00110 }
00111
00112 void ZipcodeRecordBuffer::setFieldCount(int FieldCount)
00113 {
00114     fieldCount = FieldCount;
00115 }
00116
00117 void ZipcodeRecordBuffer::setPrimaryField(const string& PrimaryField)
00118 {
00119     primaryField = PrimaryField;
00120 }
00121
00122
00123 //Getters
00124 string ZipcodeRecordBuffer::getFileType()
00125 {
00126     return filetype;
00127 }
00128
00129 double ZipcodeRecordBuffer::getVer()
00130 {
00131     return ver;
00132 }
00133
00134 int ZipcodeRecordBuffer::getLenInd()
00135 {
00136     return len_ind;
00137 }
00138
```

```
00139 int ZipcodeRecordBuffer::getRecordByte()
00140 {
00141     return recordByte;
00142 }
00143
00144 string ZipcodeRecordBuffer::getFormatType()
00145 {
00146     return formatType;
00147 }
00148
00149 string ZipcodeRecordBuffer::getPrimaryFileName()
00150 {
00151     return primaryFileName;
00152 }
00153
00154 int ZipcodeRecordBuffer::getRecordCount()
00155 {
00156     return recordCount;
00157 }
00158
00159 int ZipcodeRecordBuffer::getFieldCount()
00160 {
00161     return fieldCount;
00162 }
00163
00164 string ZipcodeRecordBuffer::getPrimaryField()
00165 {
00166     return primaryField;
00167 }
00168
00169 string ZipcodeRecordBuffer::getFieldx(int position)
00170 {
00171     return headerMap[position];
00172 }
00173
00174 string ZipcodeRecordBuffer::getFieldType(int position)
00175 {
00176     string Field = headerMap[position];
00177     if(!(isdigit(Field[0])))
00178     {
00179         return "String";
00180     }
00181     else if(Field.find("."))
00182     {
00183         return "Double";
00184     }
00185     else
00186     {
00187         return "Integer";
00188     }
00189 }
00190
00191 void ZipcodeRecordBuffer::getHeaderMap(const string& headerLine) {
00192     stringstream ss(headerLine);
00193     string field;
00194     int pos = 0;
00195
00196     headerMap.clear();
00197
00198     while(getline(ss, field, ',')) {
00199         if(isdigit(field[0]))
00200         {
00201             string length = field.substr(0,2);
00202             headerMap[0] = length;
00203             field = field.substr(2, field.size() - 2);
00204             pos++;
00205
00206             // Checks if the rest of the string has quotes
00207             if (field.front() == '"' && field.back() == '"')
00208             {
00209                 field = field.substr(1, field.size() - 2);
00210                 headerMap[pos] = field;
00211                 pos++;
00212
00213             }
00214             else
00215             {
00216                 headerMap[pos] = field;
00217                 pos++;
00218             }
00219         }
00220         // If the CSV contains quoted fields, remove the quotes.
00221         else if (field.front() == '"' && field.back() == '"') {
00222             field = field.substr(1, field.size() - 2);
00223             headerMap[pos] = field;
00224             pos++;
00225
```

```
00226         }
00227         else
00228         {
00229             headerMap[pos] = field;
00230             pos++;
00231         }
00232     }
00233 }
00234
00235
00236 // Other Functions
00237
00238 bool ZipcodeRecordBuffer::setHeaderMap(string fileName)
00239 {
00240     ifstream inFile(fileName);
00241     string line;
00242
00243     if (!inFile)
00244     {
00245         cout « "Bad file or file name" « endl;
00246         return false;
00247     }
00248
00249
00250     string newFile = "new" + fileName;
00251     ofstream file(newFile);
00252
00253
00254     // Get the header line string
00255     int count = headerMap.size();
00256     int iter = 0;
00257     string newHeader;
00258
00259     if(count == 0)
00260     {
00261         cout « "No set header line" « endl;
00262         return false;
00263     }
00264
00265     else
00266     {
00267         while(iter < count)
00268         {
00269             if(iter == (count-1) || iter == 0)
00270             {
00271                 newHeader = newHeader + headerMap[iter];
00272                 iter++;
00273             }
00274             else
00275             {
00276                 newHeader = newHeader + headerMap[iter] + ",";
00277                 iter++;
00278             }
00279         }
00280     }
00281
00282
00283     bool first = true;
00284     // Set the new header of the csv to newHeader and the rest stays the same
00285     while (getline(inFile, line))
00286     {
00287         if(first)
00288         {
00289             file « newHeader « "\n";
00290             first = false;
00291             continue;
00292         }
00293         file « line « "\n";
00294     }
00295
00296     //Copies the string fileName into the character array filename
00297     char filename[fileName.size() + 1];
00298     strcpy(filename, fileName.c_str());
00299
00300     //Closes and deletes the file with fileName name
00301     inFile.close();
00302     remove(filename);
00303
00304     //Copies the string newFile into the character array newfile
00305     char newfile[newFile.size() + 1];
00306     strcpy(newfile, newFile.c_str());
00307
00308     //Closes and renames the file with filename newFile to fileName
00309     file.close();
00310     rename(newfile, filename);
00311 }
00312
```

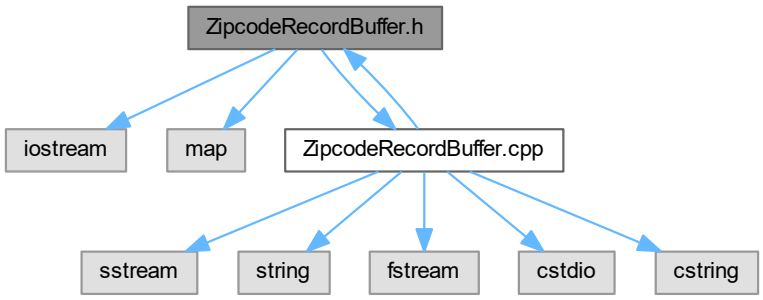
```
00313
00314 void ZipcodeRecordBuffer::printHeaderMap()
00315 {
00316     int count = headerMap.size();
00317     int iter = 0;
00318
00319     if(count == 0)
00320     {
00321         cout « "No set header line" « endl;
00322     }
00323
00324     else
00325     {
00326         while(iter < count)
00327         {
00328             cout « headerMap[iter] « " ";
00329             iter++;
00330         }
00331         cout « endl;
00332     }
00333 }
00334
00335
00336 // OVERLOADED OPERATORS
00337 // Overloaded output operator
00338 ostream& operator«(ostream& out, const ZipcodeRecordBuffer& buffer)
00339 {
00340     out « "File Type: "  « buffer.filetype « endl;
00341     out « "Version: " « buffer.ver « endl;
00342     out « "Length Indicated: " « buffer.len_ind « endl;
00343     out « "Record Bytes: " « buffer.recordByte « endl;
00344     out « "Format Type: " « buffer.formatType « endl;
00345     out « "Primary File Name: " « buffer.primaryFileName « endl;
00346     out « "Record Count: " « buffer.recordCount « endl;
00347     out « "Field Count: " « buffer.fieldCount « endl;
00348     out « "Primary Field: " « buffer.primaryField « endl;
00349 }
```

4.17 ZipcodeRecordBuffer.h File Reference

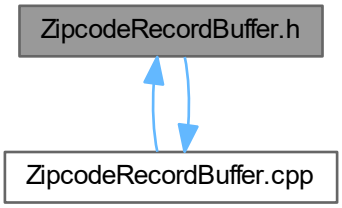
This header file defines the ZipcodeRecordBuffer class, which is used to read and write the data file header record.

```
#include <iostream>
#include <map>
#include "ZipcodeRecordBuffer.cpp"
```

Include dependency graph for ZipcodeRecordBuffer.h:



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



Classes

- class ZipcodeRecordBuffer

  Class to hold the information of a data file header record.

4.17.1 Detailed Description

This header file defines the ZipcodeRecordBuffer class, which is used to read and write the data file header record.

Author

Zachary Sunder

Version

1.1

ZipcodeRecordBuffer class: File to create and edit the data file header record

- Function to modify the data file header record created
- Overloaded output operator

  Note

- A valid CSV line must be provided to set the attributes correctly if not using the default constructor

Definition in file ZipcodeRecordBuffer.h.

4.18 ZipcodeRecordBuffer.h

Go to the documentation of this file.

```
00001
00014 #ifndef ZipcodeRecordBuffer_H
00015 #define ZipcodeRecordBuffer_H
00016
00017 #include <iostream>
00018 #include <map>
00019 using namespace std;
00020
00025 class ZipcodeRecordBuffer
00026 {
00027     private:
00028         // \brief filetype as a string
00029         string filetype;
00030
00031         // \brief ver as a double
00032         double ver;
00033
00034         // \brief len_ind as a int
00035         int len_ind;
00036
00037         // \brief recordByte as a int
00038         int recordByte;
00039
00040         // \brief formatType as a string
00041         string formatType;
00042
00043         // \brief primaryFileName as a string
00044         string primaryFileName;
00045
00046         // \brief recordCount as a int
00047         int recordCount;
00048
00049         // \brief fieldCount as a int
00050         int fieldCount;
00051
00052         // \brief primaryField as a string
00053         string primaryField;
00054
00056         map<int, string> headerMap;
00057
00058     public:
00064         ZipcodeRecordBuffer();
00065
00073         ZipcodeRecordBuffer(string fileName, string mainField);
00074
00075
00076         // Setters
00077
00084         void setFileType(const string& FileType);
00085
00092         void setVer(double Ver);
00093
00100         void setLenInd(int Len);
00101
00108         void setRecordByte(int RecordByte);
00109
00116         void setFormatType(const string& FormatType);
00117
00124         void setPrimaryFileName(const string& FileName);
00125
00132         void setRecordCount(int RecordCount);
00133
00140         void setFieldCount(int FieldCount);
00141
00148         void setPrimaryField(const string& PrimaryField);
00149
00150
00151         // Getters
00152
00159         string getFileType();
00160
00167         double getVer();
00168
00175         int getLenInd();
00176
00183         int getRecordByte();
00184
00191         string getFormatType();
00192
00199         string getPrimaryFileName();
00200
00207         int getRecordCount();
```

```
00208
00215         int getFieldCount();
00216
00223         string getPrimaryField();
00224
00231         string getFieldx(int position);
00232
00239         string getFieldType(int position);
00240
00247         void getHeaderMap(const string& headerLine);
00248
00249
00250
00251         //Other Functions
00252
00258         void printHeaderMap();
00259
00266         bool setHeaderMap(string fileName);
00267
00268
00269
00270
00271         //OVERLOADED OPERATORS
00280         friend ostream& operator«(ostream& out, const ZipcodeRecordBuffer& buffer);
00281 };
00282 #include "ZipcodeRecordBuffer.cpp"
00283
00284 #endif
```

Index