Doxygen Competency

Generated by Doxygen 1.9.3

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

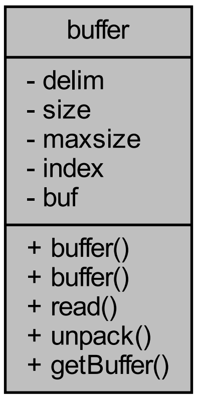
Class Documentation

3.1 buffer Class Reference

class to store each record and parse each field

`#include <buffer.h>`

Collaboration diagram for buffer:



Public Member Functions

- buffer ()

  Constructor for the buffer class.
- buffer (char, int)
- bool read (ifstream &inFile)

  reads from csv file and places on string
- bool unpack (string &field)

  Seperates each field from the line on the buffer.
- string getBuffer ()

  Gives the buffer string

Private Attributes

- char delim
- int size
- int maxsize
- int index
- string buf

3.1.1 Detailed Description

class to store each record and parse each field

3.1.2 Constructor & Destructor Documentation

3.1.2.1 buffer() `[1/2]`

```
buffer::buffer ( )
```

Constructor for the buffer class.

Precondition

Takes in the address to the us_postal_codes.csv file

Postcondition

inFile, index and buf are all initialized

BUFFER.CPP Member function definitions for the buffer class.

3.1.2.2 buffer() `[2/2]`

```
buffer::buffer (
              char delim = ',',
              int maxsize = 1000 )
```

3.1.3 Member Function Documentation

3.1.3.1 getBuffer()

```
string buffer::getBuffer ( )  [inline]
```

Gives the buffer string

Postcondition

Returns the buffer string

3.1.3.2 read()

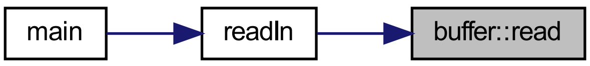
```
bool buffer::read (
             ifstream & inFile )
```

reads from csv file and places on string

Postcondition

returns the string of one line of us_postal_codes.csv

Here is the caller graph for this function:



3.1.3.3 unpack()

```
bool buffer::unpack (
             string & field )
```

Seperates each field from the line on the buffer.

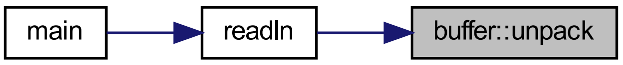
Precondition

Buffer must not be empty

Postcondition

Makes parameter string equal to correct field in record

Here is the caller graph for this function:



3.1.4 Member Data Documentation

3.1.4.1 buf

```
string buffer::buf  [private]
```

3.1.4.2 delim

```
char buffer::delim  [private]
```

3.1.4.3 index

```
int buffer::index  [private]
```

3.1.4.4 maxsize

```
int buffer::maxsize  [private]
```

3.1.4.5 size

```
int buffer::size  [private]
```

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

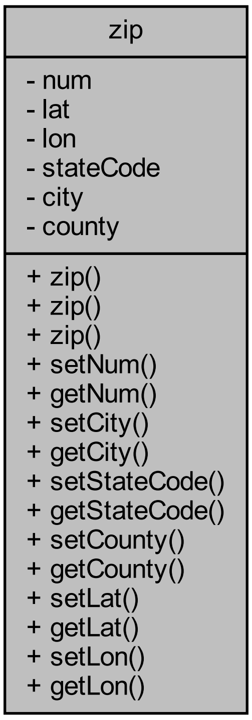
- buffer.h
- buffer.cpp

3.2 zip Class Reference

class to store each zip code as an object

```
#include <zip.h>
```

Collaboration diagram for zip:



Public Member Functions

- zip ()

  default constructor
- zip (int newNum, string newCity, string newStateCode, string newCounty, float newLat, float newLon)

  specified constructor
- zip (zip ∗oldZip)

  copy constructor
- void setNum (int newNum)

  Inline setters and getters.
- int getNum ()
- void setCity (string newCity)
- string getCity ()
- void setStateCode (string newStateCode)
- string getStateCode ()
- void setCounty (string newCounty)
- string getCounty ()
- void setLat (float newLat)
- float getLat ()
- void setLon (float newLon)
- float getLon ()

Private Attributes

- int num
- float lat
- float lon
- string stateCode
- string city
- string county

3.2.1 Detailed Description

class to store each zip code as an object

3.2.2 Constructor & Destructor Documentation

3.2.2.1 zip() [1/3]

```
zip::zip ( )
```

default constructor

Postcondition

initializes zip object to be empty

3.2.2.2 zip() [2/3]

```
zip::zip (
            int newNum,
            string newCity,
            string newStateCode,
            string newCounty,
            float newLat,
            float newLon )
```

specified constructor

Precondition

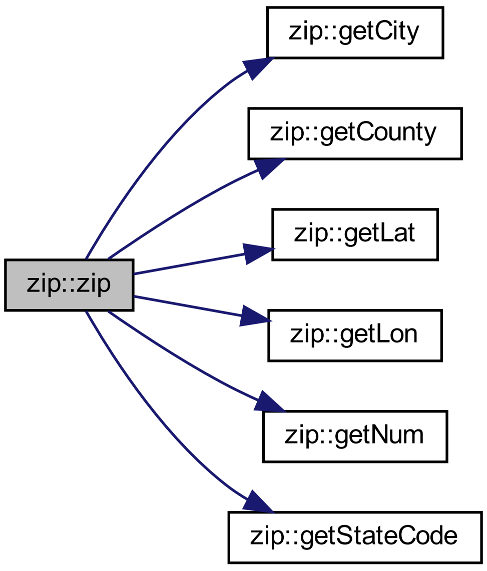
Takes in the zipcode, city of zipcode, 2 character string statecode, string for the county, floating point of the latitude, and floating point of the longitude.

3.2.2.3 zip() [3/3]

```
zip::zip (
            zip * oldZip )
```

copy constructor

Here is the call graph for this function:

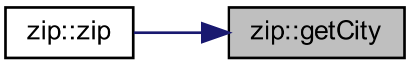


3.2.3 Member Function Documentation

3.2.3.1 getCity()

`string zip::getCity ( )  [inline]`

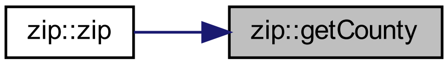
Here is the caller graph for this function:



3.2.3.2 getCounty()

`string zip::getCounty ( )  [inline]`

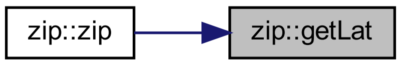
Here is the caller graph for this function:



3.2.3.3 getLat()

`float zip::getLat ( )  [inline]`

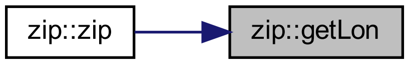
Here is the caller graph for this function:



3.2.3.4 getLon()

```
float zip::getLon ( )  [inline]
```

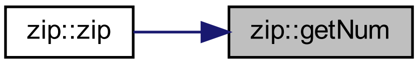
Here is the caller graph for this function:



3.2.3.5 getNum()

```
int zip::getNum ( )  [inline]
```

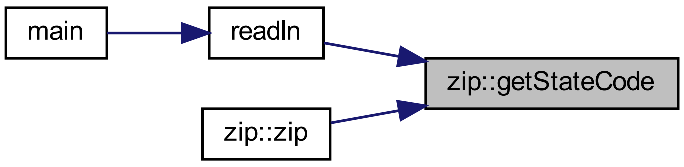
Here is the caller graph for this function:



3.2.3.6 getStateCode()

```
string zip::getStateCode ( )  [inline]
```

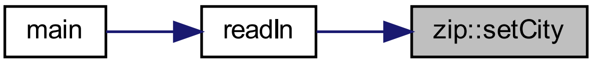
Here is the caller graph for this function:



3.2.3.7 setCity()

```
void zip::setCity (
            string newCity )  [inline]
```

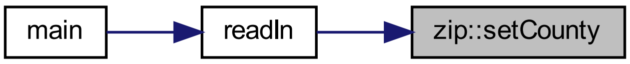
Here is the caller graph for this function:



3.2.3.8 setCounty()

```
void zip::setCounty (
            string newCounty )  [inline]
```

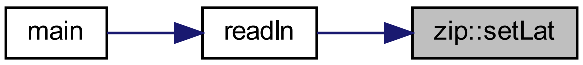
Here is the caller graph for this function:



3.2.3.9 setLat()

```
void zip::setLat (
            float newLat )  [inline]
```

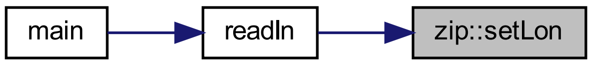
Here is the caller graph for this function:



3.2.3.10 setLon()

```
void zip::setLon (
             float newLon )  [inline]
```

Here is the caller graph for this function:

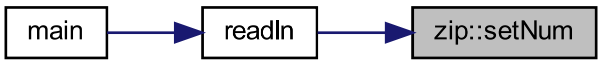


3.2.3.11 setNum()

```
void zip::setNum (
             int newNum )  [inline]
```

Inline setters and getters.

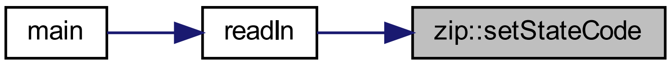
Here is the caller graph for this function:



3.2.3.12 setStateCode()

```
void zip::setStateCode (
             string newStateCode )  [inline]
```

Here is the caller graph for this function:



3.2.4 Member Data Documentation

3.2.4.1 city

```
string zip::city  [private]
```

3.2.4.2 county

```
string zip::county  [private]
```

3.2.4.3 lat

```
float zip::lat  [private]
```

3.2.4.4 lon

```
float zip::lon  [private]
```

3.2.4.5 num

```
int zip::num  [private]
```

3.2.4.6 stateCode

```
string zip::stateCode  [private]
```

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

- zip.h
- zip.cpp

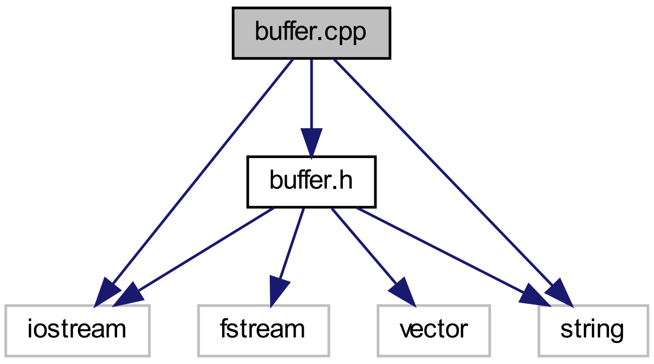
Chapter 4

File Documentation

4.1 buffer.cpp File Reference

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

Include dependency graph for buffer.cpp:

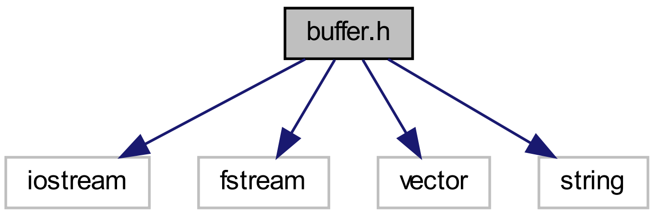


4.2 buffer.h File Reference

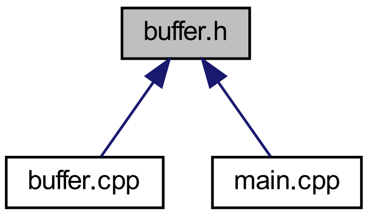
```
#include <iostream>
#include <fstream>
#include <vector>
```

`#include <string>`

Include dependency graph for buffer.h:



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



Classes

- class buffer

  class to store each record and parse each field

4.3 buffer.h

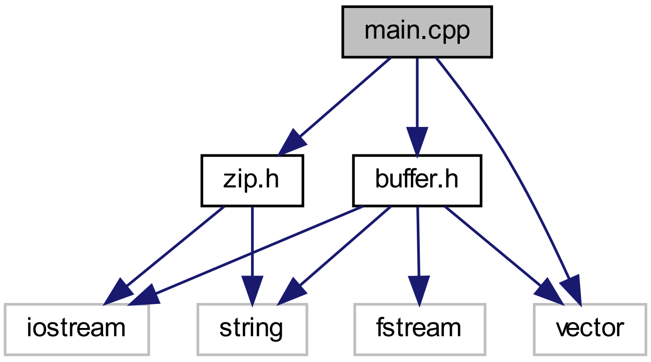
Go to the documentation of this file.

```

#ifndef BUFFER_h
#define BUFFER_h

#include <iostream>
#include <fstream>
#include <vector>
#include <string>
using namespace std;

class buffer {
public:

    buffer();
    buffer(char, int);

    bool read(ifstream& inFile);

    bool unpack(string & field);


    string getBuffer() { return buf; };




private:
    char delim;
    int size;
    int maxsize;
    int index;
    string buf;

};
#endif
```

4.4 main.cpp File Reference

```
#include "buffer.h"
#include "zip.h"
#include <vector>
```

Include dependency graph for main.cpp:



Functions

- string printTable (vector< vector< zip > > states)

    Prints the state arrays zip code state code

- void readIn (ifstream &inFile, vector< vector< zip > > &states)

    Read in data from the csv, place on buffer, and parse onto zip class data members;.
- short stateChooser (string x)

    Chooses which state array index is correct with the use of a switch statement.
- short mostNorth (vector< zip > state)

    Finds the most north zipcode of a given state.
- short mostSouth (vector< zip > state)

Finds the most south zipcode of a given state.
- short mostEast (vector< zip > state)

  Finds the most "esta" zipcode of a given state.
- short mostWest (vector< zip > state)

  Finds the most west zipcode of a given state.
- void cleanup (vector< vector< zip > > &)
- int main ()

Variables

- static const short numStates = 57

4.4.1 Function Documentation

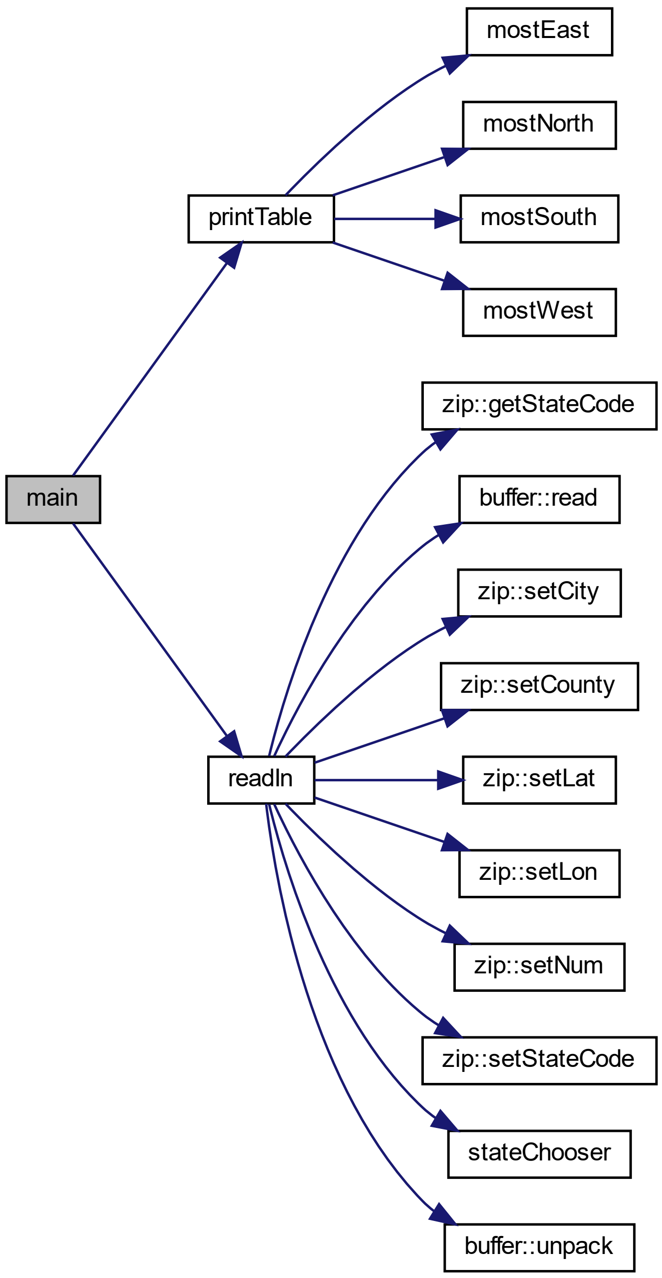
4.4.1.1 cleanup()

```
void cleanup (
            vector< vector< zip > > &  )
```

4.4.1.2 main()

```
int main ( )
```

Here is the call graph for this function:



4.4.1.3 mostEast()

```
short mostEast (
            vector< zip > state )
```

Finds the most "esta" zipcode of a given state.

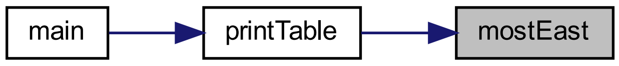
Precondition

Takes an element of the state array.

Postcondition

returns the index of the most east zipcode.

Here is the caller graph for this function:



4.4.1.4 mostNorth()

```
short mostNorth (
            vector< zip > state )
```

Finds the most north zipcode of a given state.

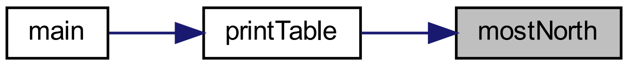
Precondition

Takes an element of the state array.

Postcondition

returns the index of the most north zipcode.

Here is the caller graph for this function:



4.4.1.5 mostSouth()

```
short mostSouth (
            vector< zip > state )
```

Finds the most south zipcode of a given state.

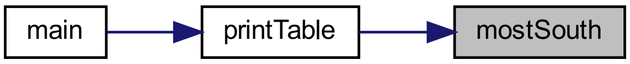
Precondition

Takes an element of the state array.

Postcondition

returns the index of the most south zipcode.

Here is the caller graph for this function:



4.4.1.6 mostWest()

```
short mostWest (
            vector< zip > state )
```

Finds the most west zipcode of a given state.

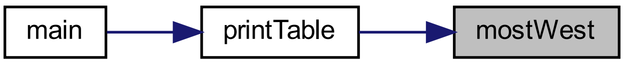
Precondition

Takes an element of the state array.

Postcondition

returns the index of the most west zipcode.

Here is the caller graph for this function:



4.4.1.7 printTable()

```
string printTable (
            vector< vector< zip > > states )
```

Prints the state arrays zip code state code

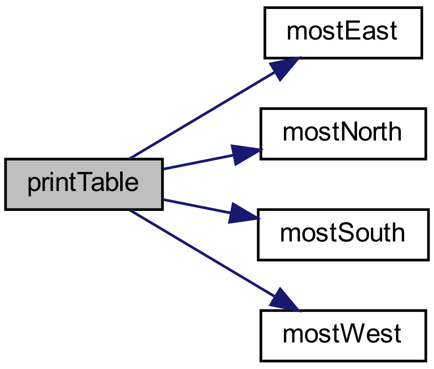
Precondition

Receives the array of state objects

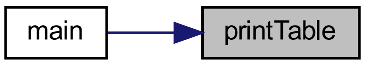
Postcondition

prints a table of the most north, south, east, and west zip codes of each state

Here is the call graph for this function:



Here is the caller graph for this function:



4.4.1.8 readIn()

```
void readIn (
            ifstream & inFile,
            vector< vector< zip > > & states )
```

Read in data from the csv, place on buffer, and parse onto zip class data members;.

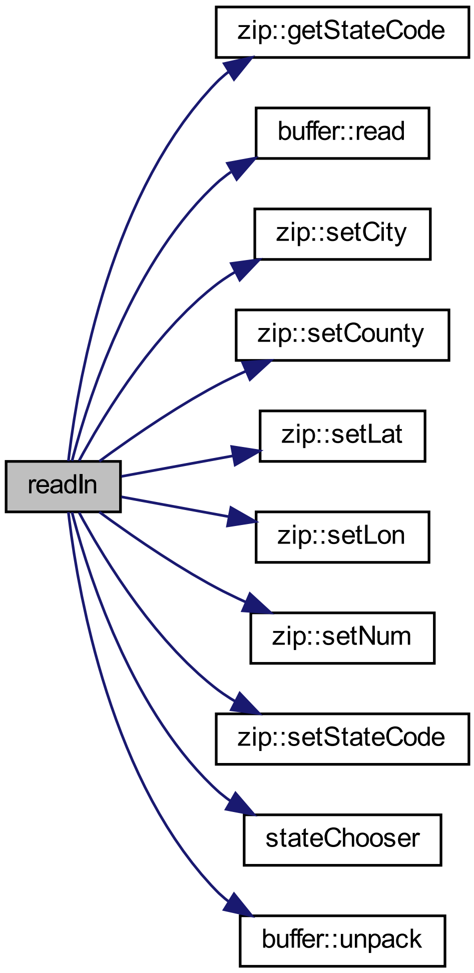
Precondition

Recieves address of the file stream, recieves a pointer to an array of state vectors.

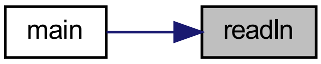
Postcondition

zip code records have been read into zip objects, zip objects have been sorted to their respective state vectors.

Here is the call graph for this function:



Here is the caller graph for this function:



4.4.1.9 stateChooser()

```
short stateChooser (
            string x )
```

Chooses which state array index is correct with the use of a switch statement.

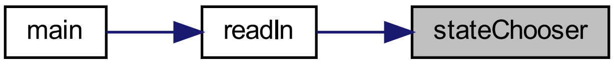
Precondition

two character state code in a string is used as parameter

Postcondition

Returns the correct array index as an int

Here is the caller graph for this function:



4.4.2 Variable Documentation

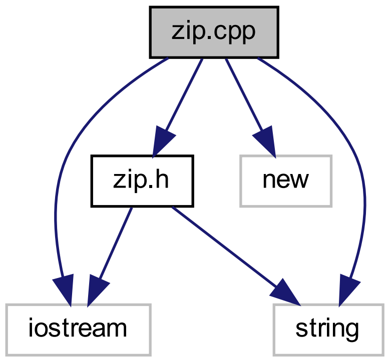
4.4.2.1 numStates

```
const short numStates = 57  [static]
```

4.5 zip.cpp File Reference

```
#include <iostream>
#include <string>
#include <new>
#include "zip.h"
```

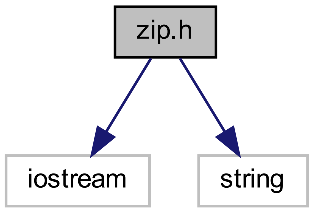
Include dependency graph for zip.cpp:



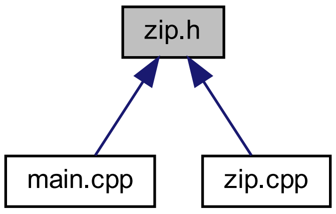
4.6 zip.h File Reference

```
#include <iostream>
#include <string>
```

Include dependency graph for zip.h:



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



Classes

- class zip

  class to store each zip code as an object

4.7 zip.h

Go to the documentation of this file.

```
#ifndef ZIP
#define ZIP

#include <iostream>
#include <string>
using namespace std;

class zip {
public:

    zip();

    zip(int newNum, string newCity, string newStateCode, string newCounty, float newLat, float newLon);

    zip(zip* oldZip);

    void setNum(int newNum) { num = newNum; };

    int getNum() { return num; };

    void setCity(string newCity) { city = newCity; };

    string getCity() { return city; };

    void setStateCode(string newStateCode) { stateCode = newStateCode; };

    string getStateCode() { return stateCode; };

    void setCounty(string newCounty) { county = newCounty; };

    string getCounty() { return county; };

    void setLat(float newLat) { lat = newLat; };

    float getLat() { return lat; };

    void setLon(float newLon) { lon = newLon; };

    float getLon() { return lon; };

private:
    int num;
    float lat;
    float lon;
    string stateCode;
    string city;
    string county;
};
#endif
```

Index