CSCI_331_GP2_T2

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

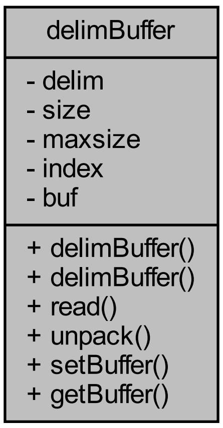
Class Documentation

3.1 delimBuffer Class Reference

class to store each record and parse each field

`#include <delimBuffer.h>`

Collaboration diagram for delimBuffer:



Public Member Functions

- delimBuffer ()

  Constructor for the delimBuffer class.
- delimBuffer (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 delimBuffer.
- void setBuffer (string x)

  Gives the delimBuffer string

- string getBuffer ()

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 delimBuffer() `[1/2]`

```
delimBuffer::delimBuffer ( )
```

Constructor for the delimBuffer class.

Precondition

Takes in the address to the us_postal_codes.csv file

Postcondition

inFile, index and buf are all initialized

delimBuffer.CPP Member function definitions for the delimBuffer class.

3.1.2.2 delimBuffer() `[2/2]`

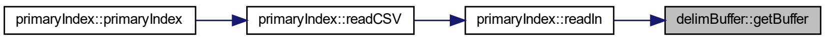
```
delimBuffer::delimBuffer (
             char delim = ',',
             int maxsize = 1000 )
```

3.1.3 Member Function Documentation

3.1.3.1 getBuffer()

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

Here is the caller graph for this function:



3.1.3.2 read()

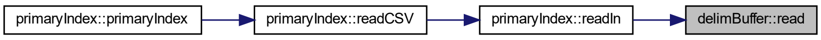
```
bool delimBuffer::read (
             ifstream & inFile )
```

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 setBuffer()

```
void delimBuffer::setBuffer (
             string x )  [inline]
```

Gives the delimBuffer string

Postcondition

Returns the delimBuffer string

3.1.3.4 unpack()

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

Seperates each field from the line on the delimBuffer.

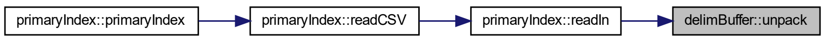
Precondition

delimBuffer 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 delimBuffer::buf  [private]
```

3.1.4.2 delim

```
char delimBuffer::delim  [private]
```

3.1.4.3 index

```
int delimBuffer::index  [private]
```

3.1.4.4 maxsize

```
int delimBuffer::maxsize [private]
```

3.1.4.5 size

```
int delimBuffer::size [private]
```

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

- delimBuffer.h
- delimBuffer.cpp

3.2 indexElement Struct Reference

```
#include <primaryindex.h>
```

Collaboration diagram for indexElement:

indexElement
+ zip
+ offset

Public Attributes

- int zip
- unsigned long int offset

3.2.1 Detailed Description

primaryindex.h Class containing the primary index and the byte offset of the data file for the corresponding primary key.

3.2.2 Member Data Documentation

3.2.2.1 offset

```
unsigned long int indexElement::offset
```

3.2.2.2 zip

```
int indexElement::zip
```

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

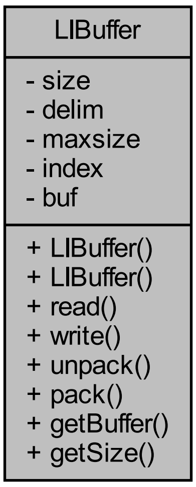
- primaryindex.h

3.3 LIBuffer Class Reference

class to store each record and parse each field

```
#include <LIBuffer.h>
```

Collaboration diagram for LIBuffer:



Public Member Functions

- LIBuffer ()

  Constructor for the LIBuffer class.
- LIBuffer (char, int)
- bool read (fstream &inFile, unsigned long offset)

  reads from csv file and places on string
- void write (fstream &outFile)
- bool unpack (string &field)

  Seperates each field from the line on the LIBuffer.
- void pack (string &field)
- string getBuffer ()

  Gives the LIBuffer string.
- int getSize ()

Private Attributes

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

3.3.1 Detailed Description

class to store each record and parse each field

3.3.2 Constructor & Destructor Documentation

3.3.2.1 LIBuffer() `[1/2]`

```
LIBuffer::LIBuffer ( )
```

Constructor for the LIBuffer class.

Precondition

Takes in the address to the us_postal_codes.csv file

Postcondition

inFile, index and buf are all initialized

3.3.2.2 LIBuffer() `[2/2]`

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

3.3.3 Member Function Documentation

3.3.3.1 getBuffer()

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

Gives the LIBuffer string.

Postcondition

Returns the LIBuffer string

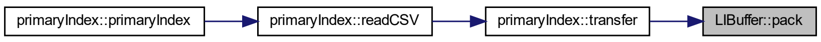
3.3.3.2 getSize()

```
int LIBuffer::getSize ( )  [inline]
```

3.3.3.3 pack()

```
void LIBuffer::pack (
             string & field )
```

Here is the caller graph for this function:



3.3.3.4 read()

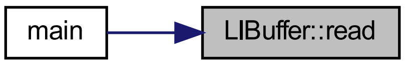
```
bool LIBuffer::read (
             fstream & inFile,
             unsigned long offset )
```

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:



3.3.3.5 unpack()

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

Seperates each field from the line on the LIBuffer.

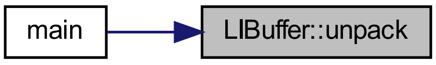
Precondition

LIBuffer must not be empty

Postcondition

Makes parameter string equal to correct field in record

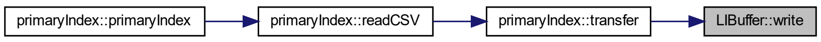
Here is the caller graph for this function:



3.3.3.6 write()

```
void LIBuffer::write (
             fstream & outFile )
```

Here is the caller graph for this function:



3.3.4 Member Data Documentation

3.3.4.1 buf

```
string LIBuffer::buf  [private]
```

3.3.4.2 delim

```
char LIBuffer::delim  [private]
```

3.3.4.3 index

```
int LIBuffer::index  [private]
```

3.3.4.4 maxsize

```
int LIBuffer::maxsize  [private]
```

3.3.4.5 size

```
int LIBuffer::size  [private]
```

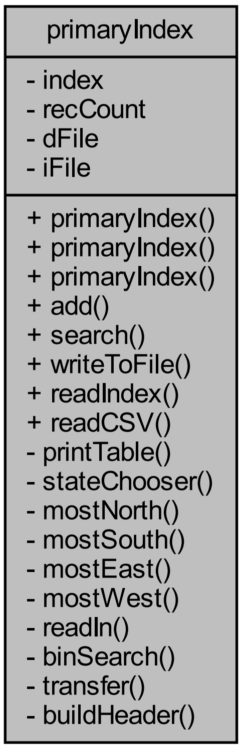
The documentation for this class was generated from the following files:

- LIBuffer.h
- LIBuffer.cpp

3.4 primaryIndex Class Reference

```
#include <primaryindex.h>
```

Collaboration diagram for primaryIndex:



Public Member Functions

- primaryIndex ()
- primaryIndex (string iFileName, string dFileName)
- primaryIndex (ifstream &infile)
- void add (int z, unsigned long o)
- unsigned long search (int target)
- void writeToFile ()
- void readIndex ()
- void readCSV (ifstream &)

Private Member Functions

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

  Prints the state arrays zip code state code.
- short stateChooser (string x)

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

- short mostNorth (vector< zip >)

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

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

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

  Finds the most west zipcode of a given state.
- string readIn (ifstream &inFile, vector< vector< zip > > &states)

  Read in data from the csv, place on buffer, and parse onto zip class data members;.
- unsigned long binSearch (int target, int l, int r)
- void transfer (vector< vector< zip > > &, string)
- string buildHeader (string)

Private Attributes

- vector< indexElement > index
- int recCount
- fstream dFile
- fstream iFile

3.4.1 Constructor & Destructor Documentation

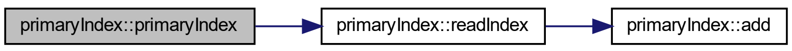
3.4.1.1 primaryIndex() [1/3]

```
primaryIndex::primaryIndex ( )
```

3.4.1.2 primaryIndex() [2/3]

```
primaryIndex::primaryIndex (
              string iFileName,
              string dFileName )   [inline]
```

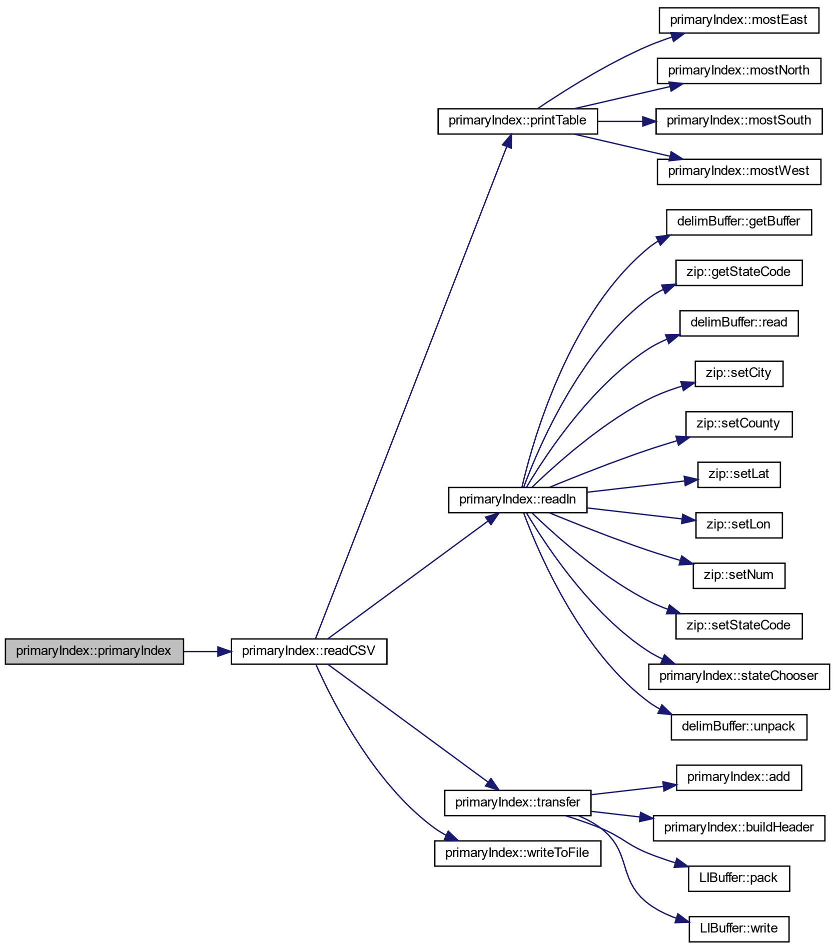
Here is the call graph for this function:



3.4.1.3 primaryIndex() [3/3]

```
primaryIndex::primaryIndex (
             ifstream & infile )  [inline]
```

Here is the call graph for this function:

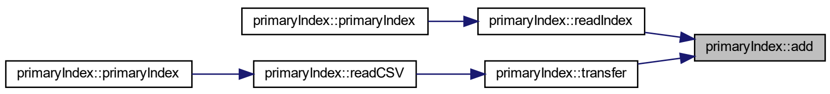


3.4.2 Member Function Documentation

3.4.2.1 add()

```
void primaryIndex::add (
             int z,
             unsigned long o )
```

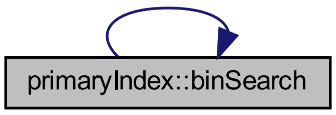
Here is the caller graph for this function:



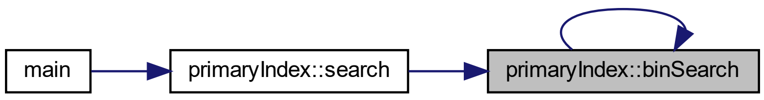
3.4.2.2 binSearch()

```
unsigned long primaryIndex::binSearch (
            int target,
            int l,
            int r )  [private]
```

Here is the call graph for this function:



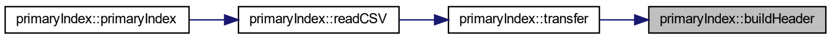
Here is the caller graph for this function:



3.4.2.3 buildHeader()

```
string primaryIndex::buildHeader (
            string headerData )  [private]
```

Here is the caller graph for this function:



3.4.2.4 mostEast()

```
short primaryIndex::mostEast (
            vector< zip > state )  [private]
```

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

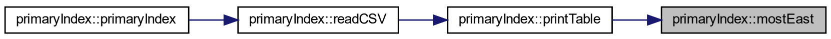
Precondition

Takes an element of the state array.

Postcondition

returns the index of the most east zipcode.

Here is the caller graph for this function:



3.4.2.5 mostNorth()

```
short primaryIndex::mostNorth (
             vector< zip > state )  [private]
```

Finds the most north zipcode of a given state.

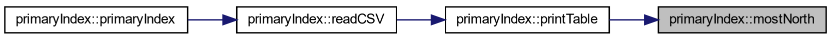
Precondition

Takes an element of the state array.

Postcondition

returns the index of the most north zipcode.

Here is the caller graph for this function:



3.4.2.6 mostSouth()

```
short primaryIndex::mostSouth (
             vector< zip > state )  [private]
```

Finds the most south zipcode of a given state.

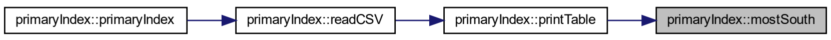
Precondition

Takes an element of the state array.

Postcondition

returns the index of the most south zipcode.

Here is the caller graph for this function:



3.4.2.7 mostWest()

```
short primaryIndex::mostWest (
            vector< zip > state )  [private]
```

Finds the most west zipcode of a given state.

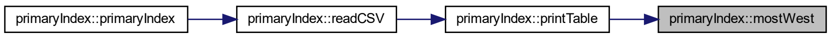
Precondition

Takes an element of the state array.

Postcondition

returns the index of the most west zipcode.

Here is the caller graph for this function:



3.4.2.8 printTable()

```
string primaryIndex::printTable (
            vector< vector< zip > > & states )  [private]
```

Prints the state arrays zip code state code.

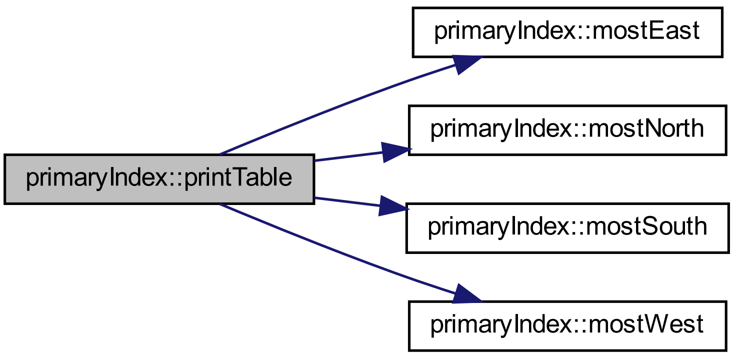
Precondition

Receives the array of state objects

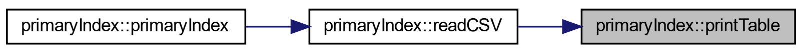
Postcondition

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

Here is the call graph for this function:



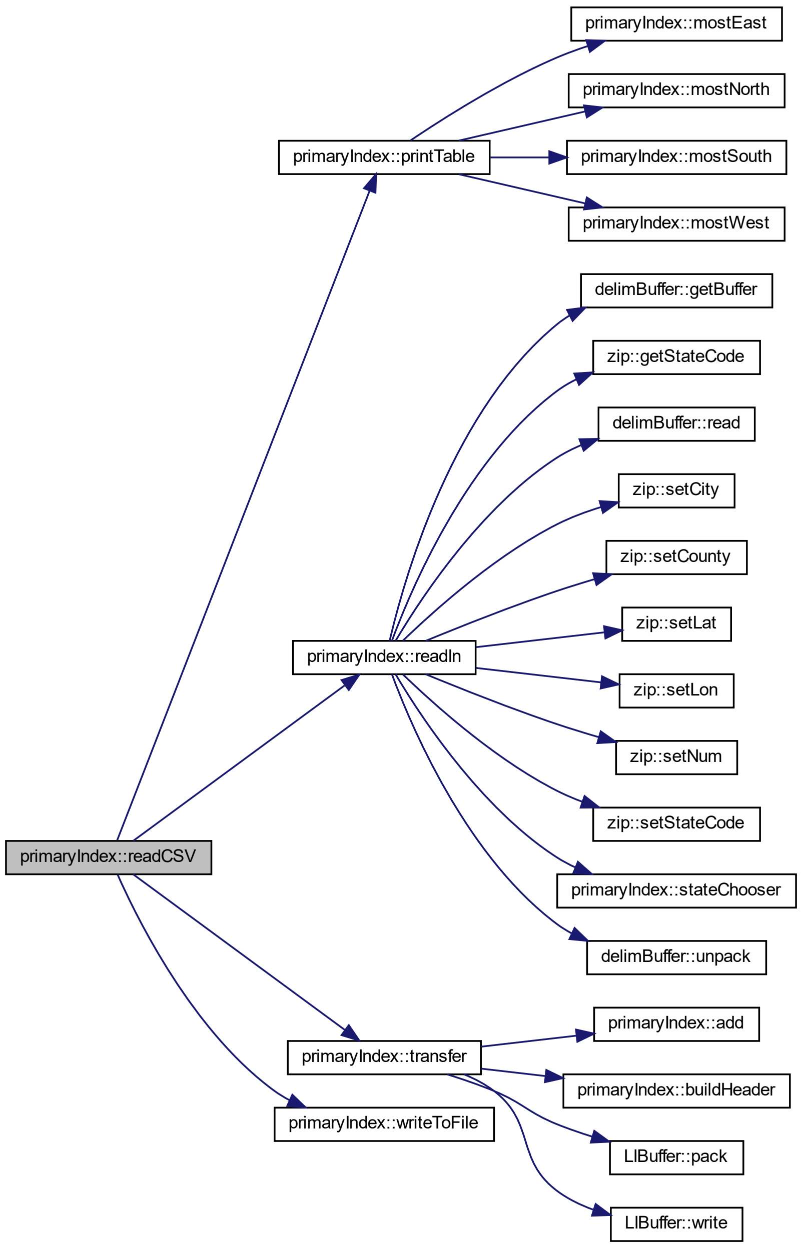
Here is the caller graph for this function:



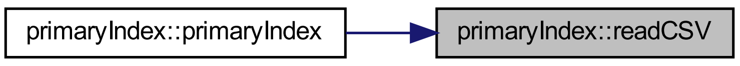
3.4.2.9 readCSV()

```
void primaryIndex::readCSV (
             ifstream & infile )
```

Here is the call graph for this function:



Here is the caller graph for this function:



3.4.2.10 readIn()

```
string primaryIndex::readIn (
            ifstream & inFile,
            vector< vector< zip > > & states )  [private]
```

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

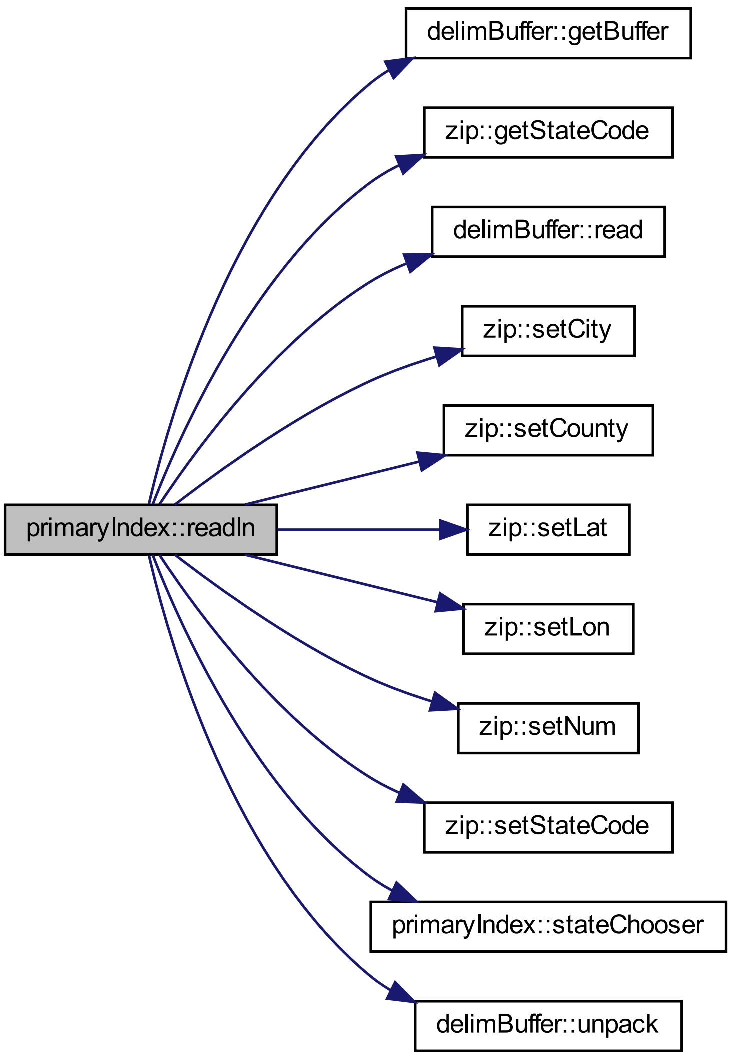
Precondition

Receives address of the file stream, receives a pointer to an array of state vectors.

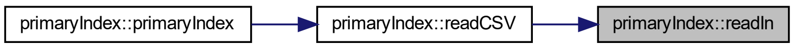
Postcondition

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:



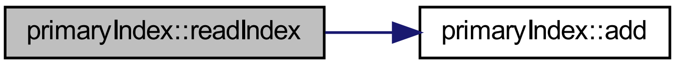
Here is the caller graph for this function:



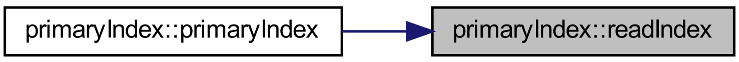
3.4.2.11 readIndex()

```
void primaryIndex::readIndex ( )
```

Here is the call graph for this function:



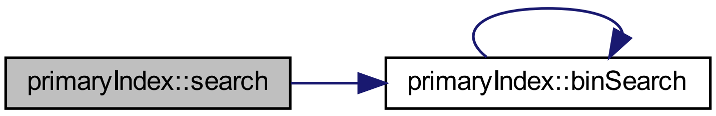
Here is the caller graph for this function:



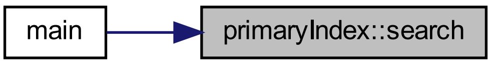
3.4.2.12 search()

```
unsigned long primaryIndex::search (
              int target )
```

Here is the call graph for this function:



Here is the caller graph for this function:



3.4.2.13 stateChooser()

```
short primaryIndex::stateChooser (
            string x )  [private]
```

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

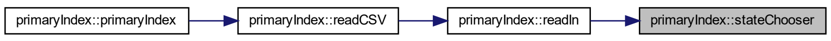
Precondition

two character state code in a string is used as parameter

Postcondition

Returns the correct array index as an int

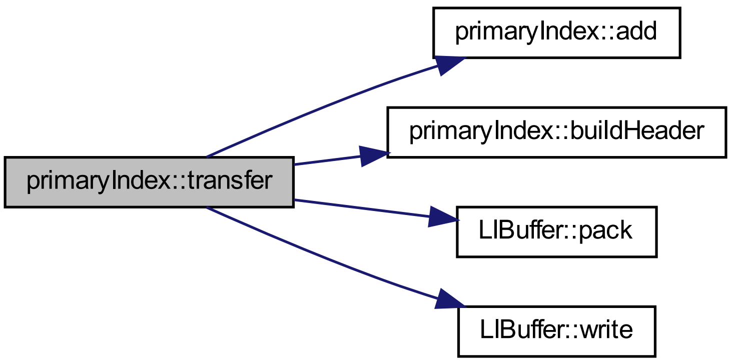
Here is the caller graph for this function:



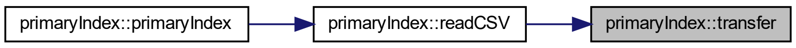
3.4.2.14 transfer()

```
void primaryIndex::transfer (
             vector< vector< zip > > & states,
             string headerData )  [private]
```

Here is the call graph for this function:



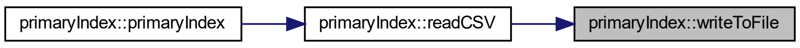
Here is the caller graph for this function:



3.4.2.15 writeToFile()

```
void primaryIndex::writeToFile ( )
```

Here is the caller graph for this function:



3.4.3 Member Data Documentation

3.4.3.1 dFile

`fstream primaryIndex::dFile [private]`

3.4.3.2 iFile

`fstream primaryIndex::iFile [private]`

3.4.3.3 index

`vector<indexElement> primaryIndex::index [private]`

3.4.3.4 recCount

`int primaryIndex::recCount [private]`

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

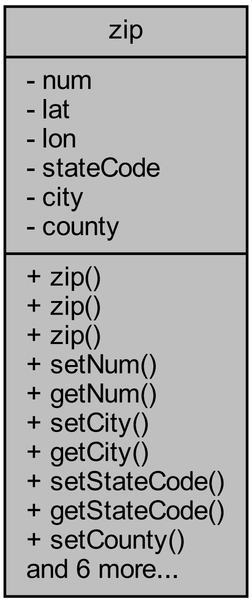
- primaryindex.h
- primaryindex.cpp

3.5 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 (const 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 ()
- void print ()

Private Attributes

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

3.5.1 Detailed Description

class to store each zip code as an object

3.5.2 Constructor & Destructor Documentation

3.5.2.1 zip() [1/3]

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

default constructor

Postcondition

initializes zip object to be empty

3.5.2.2 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.

3.5.2.3 zip() [3/3]

```
zip::zip (
             const zip & oldZip )
```

copy constructor

3.5.3 Member Function Documentation

3.5.3.1 getCity()

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

3.5.3.2 getCounty()

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

3.5.3.3 getLat()

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

3.5.3.4 getLon()

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

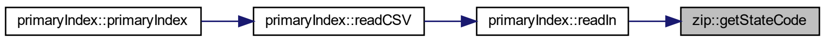
3.5.3.5 getNum()

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

3.5.3.6 getStateCode()

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

Here is the caller graph for this function:



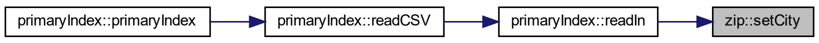
3.5.3.7 print()

```
void zip::print ( )  [inline]
```

3.5.3.8 setCity()

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

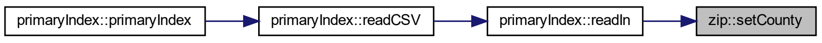
Here is the caller graph for this function:



3.5.3.9 setCounty()

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

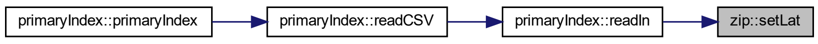
Here is the caller graph for this function:



3.5.3.10 setLat()

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

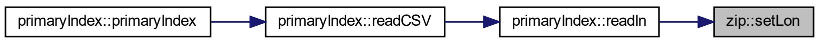
Here is the caller graph for this function:



3.5.3.11 setLon()

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

Here is the caller graph for this function:

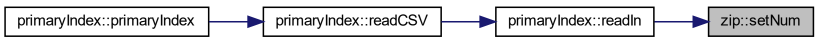


3.5.3.12 setNum()

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

Inline setters and getters.

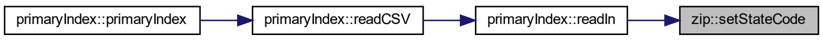
Here is the caller graph for this function:



3.5.3.13 setStateCode()

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

Here is the caller graph for this function:



3.5.4 Member Data Documentation

3.5.4.1 city

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

3.5.4.2 county

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

3.5.4.3 lat

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

3.5.4.4 lon

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

3.5.4.5 num

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

3.5.4.6 stateCode

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

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

- zip.h
- zip.cpp

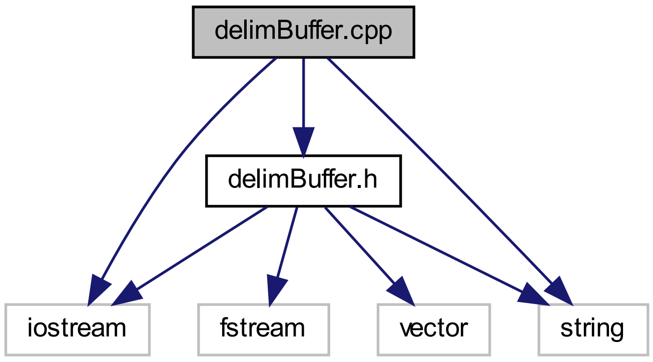
Chapter 4

File Documentation

4.1 delimBuffer.cpp File Reference

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

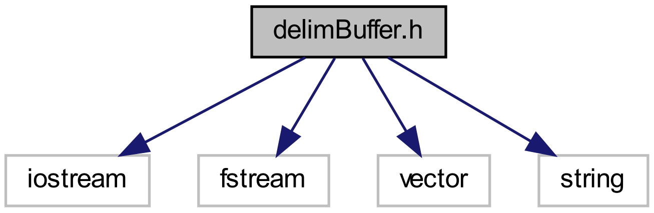
Include dependency graph for delimBuffer.cpp:



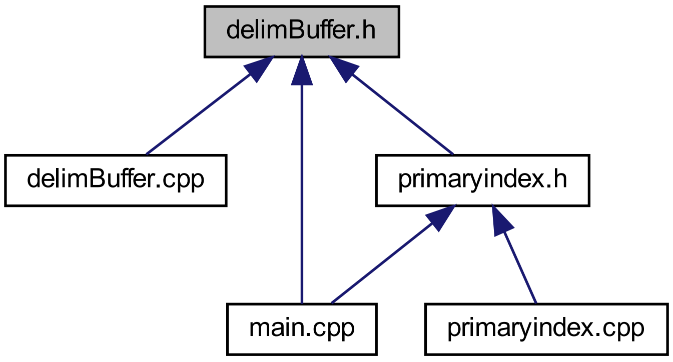
4.2 delimBuffer.h File Reference

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

```
#include <string>
```
Include dependency graph for delimBuffer.h:



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



Classes

- class delimBuffer

    class to store each record and parse each field

4.3 delimBuffer.h

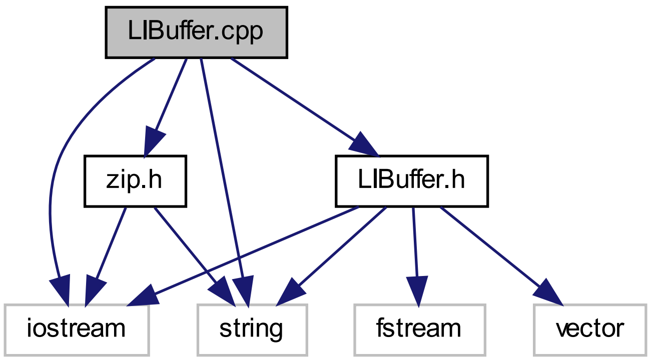
Go to the documentation of this file.
```
#ifndef DELIMBUFFER_h
#define DELIMBUFFER_h

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

```
using namespace std;

class delimBuffer {
public:

    delimBuffer();
    delimBuffer(char, int);

    bool read(ifstream& inFile);

    bool unpack(string & field);

    void setBuffer(string x) { buf = x; };
    string getBuffer() { return buf; };




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

};
#endif
```

4.4 LIBuffer.cpp File Reference

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

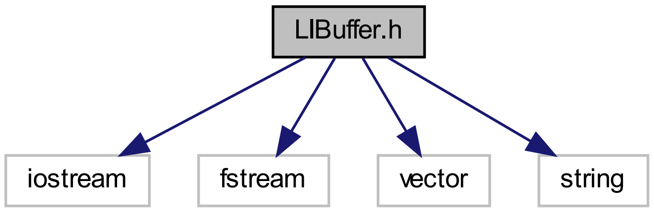
Include dependency graph for LIBuffer.cpp:



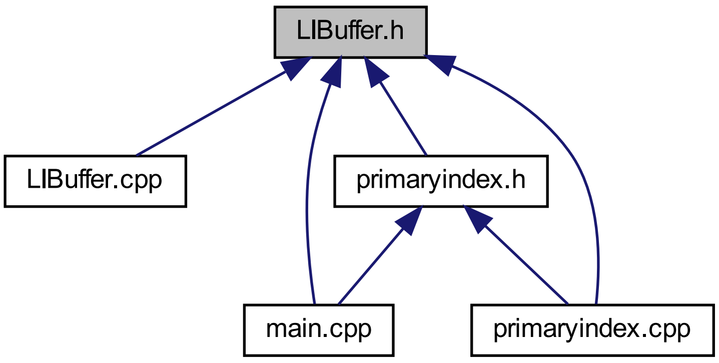
4.5 LIBuffer.h File Reference

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

`#include <string>`
Include dependency graph for LIBuffer.h:



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



Classes

- class LIBuffer

    class to store each record and parse each field

4.6 LIBuffer.h

Go to the documentation of this file.

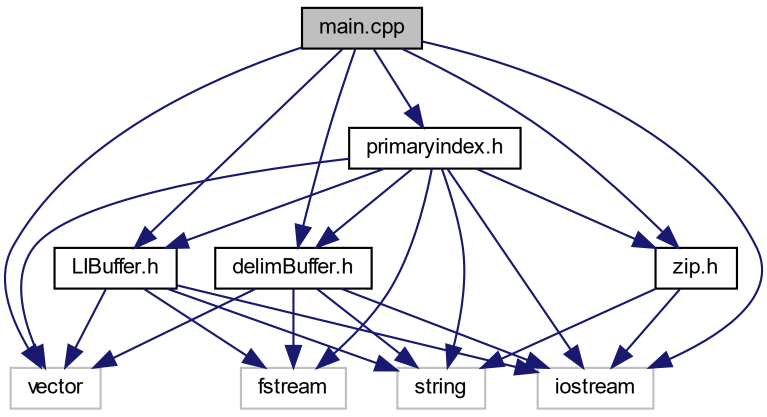
```
#ifndef LIBUFFER_h
#define LIBUFFER_h

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

```
using namespace std;

class LIBuffer {
public:

    LIBuffer();
    LIBuffer(char, int);

    bool read(fstream& inFile, unsigned long offset);

    void write(fstream& outFile);

    bool unpack(string& field);

    void pack(string& field);

    string getBuffer() { return buf; }


    int getSize() { return buf.size(); }


private:

    int size;
    char delim;
    int maxsize;
    int index;
    string buf;
};
#endif
```

4.7 main.cpp File Reference

```
#include "primaryindex.h"
#include "delimBuffer.h"
#include "LIBuffer.h"
#include "zip.h"
#include <vector>
#include <iostream>
```

Include dependency graph for main.cpp:



Functions

- int main (int argc, char ∗argv[ ])

Variables

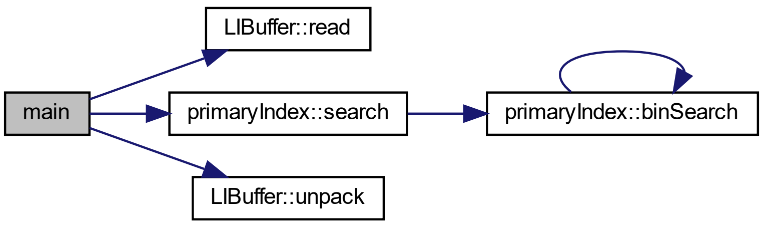
- const string manual

4.7.1 Function Documentation

4.7.1.1 main()

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

Here is the call graph for this function:



4.7.2 Variable Documentation

4.7.2.1 manual

```
const string manual
```

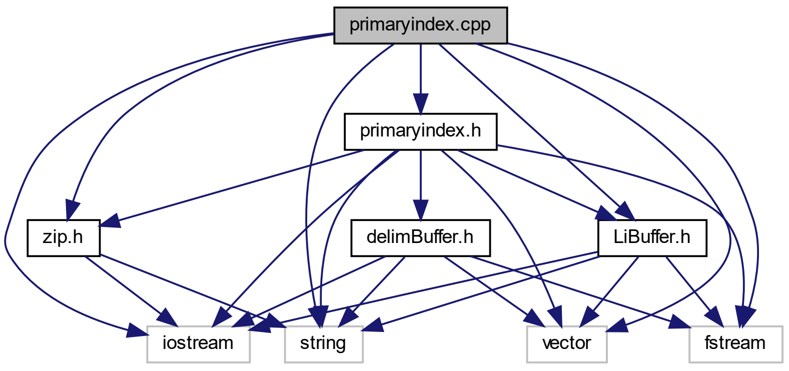
Initial value:

```
=
"sample input: programname -r filename.csv\noptions: \n-r <filename.csv>\n-z <zip code> \nprogram must be
      run once with a csv file to generate the datafile and index"
```

4.8 primaryindex.cpp File Reference

```
#include "LiBuffer.h"
#include "primaryindex.h"
#include "zip.h"
#include <iostream>
#include <string>
#include <fstream>
#include <vector>
```

Include dependency graph for primaryindex.cpp:



Variables

- static const short numStates = 57

4.8.1 Variable Documentation

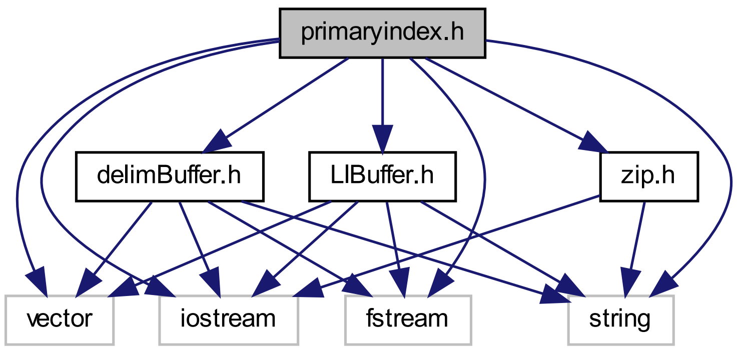
4.8.1.1 numStates

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

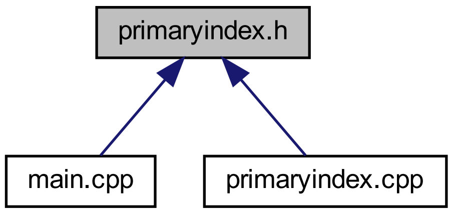
4.9 primaryindex.h File Reference

```
#include <vector>
#include <iostream>
#include <fstream>
#include <string>
#include "LIBuffer.h"
#include "zip.h"
```

`#include "delimBuffer.h"`
Include dependency graph for primaryindex.h:



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



Classes

- struct indexElement
- class primaryIndex

4.10 primaryindex.h

Go to the documentation of this file.

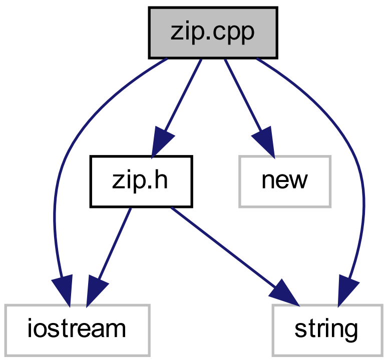
```

#include <vector>
#include <iostream>
#include <fstream>
#include <string>
#include "LIBuffer.h"
#include "zip.h"
#include "delimBuffer.h"

```

```

struct indexElement {

    int zip;
    unsigned long int offset;
};

class primaryIndex {
public:
    primaryIndex();

    primaryIndex(string iFileName, string dFileName) { iFile.open(iFileName); dFile.open(dFileName);
    readIndex(); iFile.close(); dFile.close(); }

    primaryIndex(ifstream& infile) { readCSV(infile); }

    void add(int z, unsigned long o);

    unsigned long search(int target);

    void writeToFile();

    void readIndex();

    void readCSV(ifstream&);

private:

    string printTable(vector<vector<zip»&); // output data table

    short stateChooser(string x);   // return index of state with given 2 letter code

    short mostNorth(vector<zip>); // searches a given state to find the most northern zipcode

    short mostSouth(vector<zip>); // searches a given state to find the most southern zipcode

    short mostEast(vector<zip>); // searches a given state to find the most eastern zipcode //moost
    steeast

    short mostWest(vector<zip>); // searches a given state to find the most western zipcode

    string readIn(ifstream& inFile, vector<vector<zip»& states);

    unsigned long binSearch(int target, int l, int r);

    void transfer(vector<vector<zip»&, string);

    string buildHeader(string);

    vector<indexElement> index;
    int recCount;
    fstream dFile, iFile;

};
```

4.11 zip.cpp File Reference

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

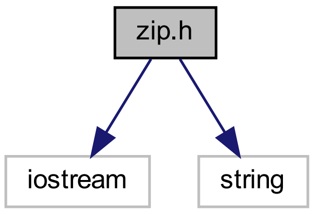
Include dependency graph for zip.cpp:



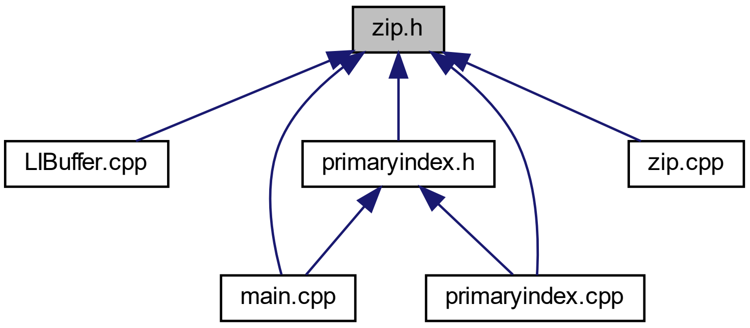
4.12 zip.h File Reference

```
#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

4.13 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(const 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; };

    void print() {
        cout « "\nZip Code:" « to_string(num) « ", City: " « city « ", County: " « county
            « ", stateCode: " « stateCode « ", Lat: " « to_string(lat) « ", Lon: " « to_string(lon) «
    "\n";
    }

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

Index