

Assignment 2

Generated by Doxygen 1.9.1

| | |
|--|----------|
| 1 Class Index | 1 |
| 1.1 Class List | 1 |
| 2 File Index | 3 |
| 2.1 File List | 3 |
| 3 Class Documentation | 5 |
| 3.1 DelimBuffer Class Reference | 5 |
| 3.1.1 Detailed Description | 6 |
| 3.1.2 Constructor & Destructor Documentation | 6 |
| 3.1.2.1 DelimBuffer() | 6 |
| 3.1.3 Member Function Documentation | 6 |
| 3.1.3.1 clear() | 6 |
| 3.1.3.2 read() | 7 |
| 3.1.3.3 unpack() | 7 |
| 3.1.4 Member Data Documentation | 8 |
| 3.1.4.1 buffer | 8 |
| 3.1.4.2 bufferSize | 8 |
| 3.1.4.3 delim | 8 |
| 3.1.4.4 nextCharIndex | 8 |
| 3.2 DelimTextBuffer Class Reference | 9 |
| 3.2.1 Detailed Description | 9 |
| 3.2.2 Constructor & Destructor Documentation | 9 |
| 3.2.2.1 DelimTextBuffer() | 10 |
| 3.2.3 Member Function Documentation | 10 |
| 3.2.3.1 Read() | 10 |
| 3.2.4 Member Data Documentation | 10 |
| 3.2.4.1 Buffer | 10 |
| 3.2.4.2 Buffersize | 10 |
| 3.2.4.3 Delim | 10 |
| 3.2.4.4 MaxBytes | 11 |
| 3.2.4.5 NextByte | 11 |
| 3.3 Location Class Reference | 11 |
| 3.3.1 Detailed Description | 13 |
| 3.3.2 Constructor & Destructor Documentation | 13 |
| 3.3.2.1 Location() [1/3] | 13 |
| 3.3.2.2 Location() [2/3] | 14 |
| 3.3.2.3 Location() [3/3] | 14 |
| 3.3.3 Member Function Documentation | 15 |
| 3.3.3.1 getCounty() | 15 |
| 3.3.3.2 getLat() | 16 |
| 3.3.3.3 getLong() | 16 |
| 3.3.3.4 getName() | 17 |

| | |
|--|-----------|
| 3.3.3.5 getState() | 17 |
| 3.3.3.6 getZipCode() | 18 |
| 3.3.3.7 operator<() | 18 |
| 3.3.3.8 operator=() | 19 |
| 3.3.3.9 operator>() | 20 |
| 3.3.3.10 setCounty() | 21 |
| 3.3.3.11 setLat() | 22 |
| 3.3.3.12 setLong() | 23 |
| 3.3.3.13 setName() | 23 |
| 3.3.3.14 setState() | 24 |
| 3.3.3.15 setZipCode() | 25 |
| 3.3.3.16 unpack() | 26 |
| 3.3.4 Friends And Related Function Documentation | 27 |
| 3.3.4.1 operator<< | 27 |
| 3.3.5 Member Data Documentation | 27 |
| 3.3.5.1 county | 27 |
| 3.3.5.2 latitude | 27 |
| 3.3.5.3 longitude | 27 |
| 3.3.5.4 name | 28 |
| 3.3.5.5 state | 28 |
| 3.3.5.6 zipcode | 28 |
| 3.4 Record Class Reference | 28 |
| 3.4.1 Detailed Description | 29 |
| 3.4.2 Constructor & Destructor Documentation | 29 |
| 3.4.2.1 Record() [1/2] | 29 |
| 3.4.2.2 Record() [2/2] | 29 |
| 3.4.2.3 ~Record() | 30 |
| 3.4.3 Member Data Documentation | 30 |
| 3.4.3.1 County | 30 |
| 3.4.3.2 Lat | 30 |
| 3.4.3.3 Long | 30 |
| 3.4.3.4 placeName | 30 |
| 3.4.3.5 State | 30 |
| 3.4.3.6 zipCode | 31 |
| 3.5 State Class Reference | 31 |
| 3.5.1 Detailed Description | 31 |
| 4 File Documentation | 33 |
| 4.1 Buffer.cpp File Reference | 33 |
| 4.2 Buffer.cpp | 33 |
| 4.3 Buffer.h File Reference | 34 |
| 4.4 Buffer.h | 34 |

| | |
|--|-----------|
| 4.5 DelimBuffer.cpp File Reference | 35 |
| 4.6 DelimBuffer.cpp | 35 |
| 4.7 DelimBuffer.h File Reference | 36 |
| 4.8 DelimBuffer.h | 37 |
| 4.9 KeyNode.cpp File Reference | 37 |
| 4.10 KeyNode.cpp | 38 |
| 4.11 Location.cpp File Reference | 38 |
| 4.11.1 Function Documentation | 38 |
| 4.11.1.1 operator<<() | 38 |
| 4.12 Location.cpp | 39 |
| 4.13 Location.h File Reference | 40 |
| 4.14 Location.h | 41 |
| 4.15 main.cpp File Reference | 42 |
| 4.15.1 Function Documentation | 42 |
| 4.15.1.1 main() | 42 |
| 4.16 main.cpp | 43 |
| 4.17 QuickSort.cpp File Reference | 44 |
| 4.17.1 Function Documentation | 45 |
| 4.17.1.1 insertSort() | 45 |
| 4.17.1.2 partition() | 45 |
| 4.17.1.3 qSort() | 46 |
| 4.17.1.4 sortFirstMidLast() | 47 |
| 4.17.1.5 swap_loc() | 48 |
| 4.18 QuickSort.cpp | 48 |
| 4.19 Record.h File Reference | 50 |
| 4.20 Record.h | 50 |
| Index | 51 |

Chapter 1

Class Index

1.1 Class List

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

| | |
|---------------------------------|----|
| DelimBuffer | 5 |
| DelimTextBuffer | 9 |
| Location | 11 |
| Record | 28 |
| State | 31 |

Chapter 2

File Index

2.1 File List

Here is a list of all files with brief descriptions:

| | |
|---------------------------------|----|
| Buffer.cpp | 33 |
| Buffer.h | 34 |
| DelimBuffer.cpp | 35 |
| DelimBuffer.h | 36 |
| KeyNode.cpp | 37 |
| Location.cpp | 38 |
| Location.h | 40 |
| main.cpp | 42 |
| QuickSort.cpp | 44 |
| Record.h | 50 |

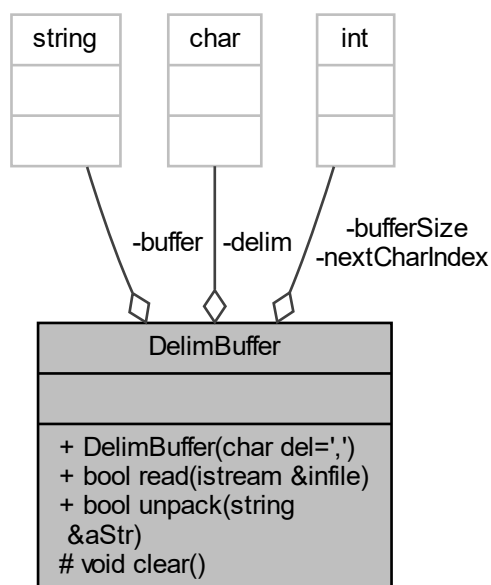
Chapter 3

Class Documentation

3.1 DelimBuffer Class Reference

```
#include <DelimBuffer.h>
```

Collaboration diagram for DelimBuffer:



Public Member Functions

- `DelimBuffer` (char del=',')
Default Constructor.
- `bool read` (istream &infile)
method reads from the file stream and tests to see if stream open
- `bool unpack` (string &aStr)
method "unpacks" the passed string pointer

Protected Member Functions

- void `clear` ()
clear the buffer

Private Attributes

- char `delim` = ''
Delimiter.
- string `buffer`
The buffer.
- int `bufferSize` = 0
Size of the buffer string.
- int `nextCharIndex` = 0
Size of the buffer string.

3.1.1 Detailed Description

Definition at line 21 of file [DelimBuffer.h](#).

3.1.2 Constructor & Destructor Documentation

3.1.2.1 DelimBuffer()

```
DelimBuffer::DelimBuffer (  
    char del = ',' ) [inline]
```

Definition at line 26 of file [DelimBuffer.h](#).

3.1.3 Member Function Documentation

3.1.3.1 clear()

```
void DelimBuffer::clear ( ) [protected]
```

Parameters

| | |
|-----------|-------------------|
| <i>no</i> | parameters passed |
|-----------|-------------------|

Returns

no return values

Definition at line 16 of file [DelimBuffer.cpp](#).

3.1.3.2 read()

```
bool DelimBuffer::read (
    istream & infile )
```

Parameters

| | |
|---------------|--------------------------|
| <i>infile</i> | file stream is passed in |
|---------------|--------------------------|

Returns

returns a boolean value about whether the stream is open or not

Definition at line 25 of file [DelimBuffer.cpp](#).

Here is the caller graph for this function:

**3.1.3.3 unpack()**

```
bool DelimBuffer::unpack (
    string & aStr )
```

Parameters

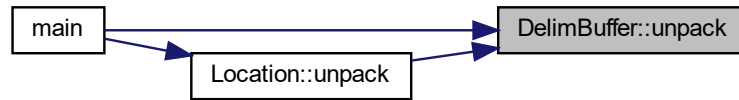
| | |
|---------------|--------------|
| <i>string</i> | pointer aStr |
|---------------|--------------|

Returns

Returns a boolean value if the delimiter is in the string or not

Definition at line 36 of file [DelimBuffer.cpp](#).

Here is the caller graph for this function:



3.1.4 Member Data Documentation

3.1.4.1 buffer

```
string DelimBuffer::buffer [private]
```

Definition at line 41 of file [DelimBuffer.h](#).

3.1.4.2 bufferSize

```
int DelimBuffer::bufferSize = 0 [private]
```

Definition at line 42 of file [DelimBuffer.h](#).

3.1.4.3 delim

```
char DelimBuffer::delim = ' ' [private]
```

Definition at line 40 of file [DelimBuffer.h](#).

3.1.4.4 nextCharIndex

```
int DelimBuffer::nextCharIndex = 0 [private]
```

Definition at line 43 of file [DelimBuffer.h](#).

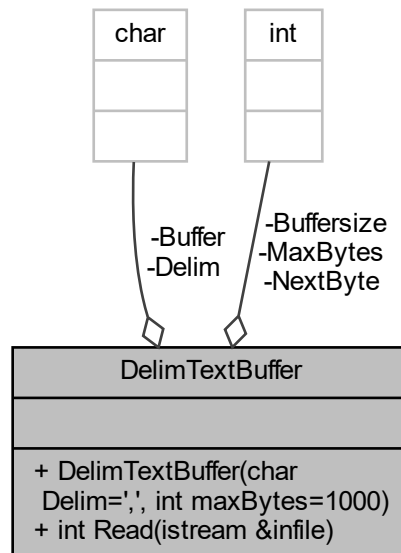
The documentation for this class was generated from the following files:

- [DelimBuffer.h](#)
- [DelimBuffer.cpp](#)

3.2 DelimTextBuffer Class Reference

```
#include <Buffer.h>
```

Collaboration diagram for DelimTextBuffer:



Public Member Functions

- [DelimTextBuffer](#) (char [Delim](#)=';', int maxBytes=1000)
- int [Read](#) (istream &infile)

Private Attributes

- char [Delim](#)
- char * [Buffer](#)
- int [Buffersize](#)
- int [MaxBytes](#)
- int [NextByte](#)

3.2.1 Detailed Description

Definition at line 8 of file [Buffer.h](#).

3.2.2 Constructor & Destructor Documentation

3.2.2.1 DelimTextBuffer()

```
DelimTextBuffer::DelimTextBuffer (
    char Delim = ' ',
    int maxBytes = 1000 )
```

3.2.3 Member Function Documentation

3.2.3.1 Read()

```
int DelimTextBuffer::Read (
    istream & infile )
```

Definition at line 9 of file [Buffer.cpp](#).

3.2.4 Member Data Documentation

3.2.4.1 Buffer

```
char* DelimTextBuffer::Buffer [private]
```

Definition at line 20 of file [Buffer.h](#).

3.2.4.2 Buffersize

```
int DelimTextBuffer::Buffersize [private]
```

Definition at line 21 of file [Buffer.h](#).

3.2.4.3 Delim

```
char DelimTextBuffer::Delim [private]
```

Definition at line 19 of file [Buffer.h](#).

3.2.4.4 MaxBytes

```
int DelimTextBuffer::MaxBytes [private]
```

Definition at line 22 of file [Buffer.h](#).

3.2.4.5 NextByte

```
int DelimTextBuffer::NextByte [private]
```

Definition at line 23 of file [Buffer.h](#).

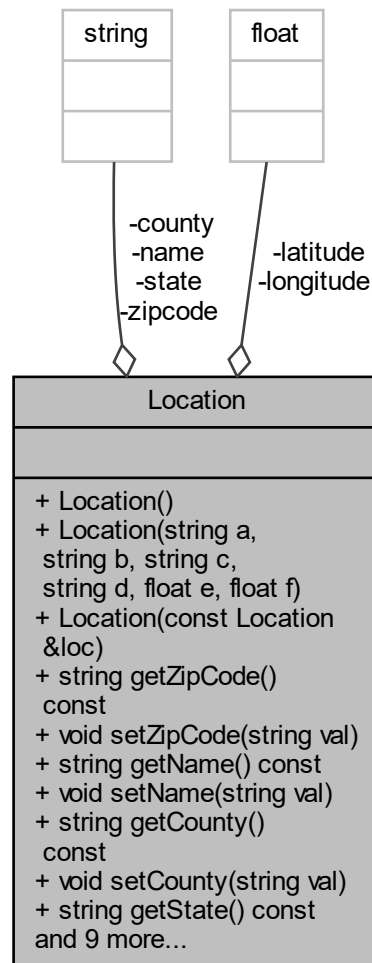
The documentation for this class was generated from the following files:

- [Buffer.h](#)
- [Buffer.cpp](#)

3.3 Location Class Reference

```
#include <Location.h>
```

Collaboration diagram for Location:



Public Member Functions

- [Location](#) ()
- [Location](#) (string a, string b, string c, string d, float e, float f)
overloaded constructor
- [Location](#) (const [Location](#) &loc)
copy constructor
- string [getZipCode](#) () const
- void [setZipCode](#) (string val)
- string [getName](#) () const
- void [setName](#) (string val)
- string [getCounty](#) () const
- void [setCounty](#) (string val)
- string [getState](#) () const
- void [setState](#) (string val)

- float [getLat](#) () const
- void [setLat](#) (float val)
- float [getLong](#) () const
- void [setLong](#) (float val)
- bool [unpack](#) ([DelimBuffer](#) &buffer)
This method "unpacks" the delimiter object that is passed and reads from its provided file.
- void [operator=](#) (const [Location](#) &loc)
overloaded assignment operator for a location object
- bool [operator<](#) (const [Location](#) &loc) const
less than comparison operator overloaded
- bool [operator>](#) (const [Location](#) &loc) const
less than comparison operator overloaded

Private Attributes

- string [zipcode](#)
Member variable "zipcode".
- string [name](#)
Member variable "name".
- string [county](#)
Member variable "county".
- string [state](#)
Member variable "state".
- float [latitude](#)
Member variable "lat".
- float [longitude](#)
Member variable "long".

Friends

- ostream & [operator<<](#) (ostream &out, const [Location](#) &loc)
ostream operator overloaded

3.3.1 Detailed Description

Definition at line 21 of file [Location.h](#).

3.3.2 Constructor & Destructor Documentation

3.3.2.1 [Location\(\)](#) [1/3]

```
Location::Location ( ) [inline]
```

Default constructor

Definition at line 25 of file [Location.h](#).

3.3.2.2 Location() [2/3]

```
Location::Location (
    string a,
    string b,
    string c,
    string d,
    float e,
    float f ) [inline]
```

Parameters

| | |
|---------------|------------------------------|
| <i>string</i> | a sets zipcode |
| <i>string</i> | b sets name of place |
| <i>string</i> | c sets name of county |
| <i>string</i> | d sets two letter state code |
| <i>float</i> | e sets the latitude value |
| <i>float</i> | f sets the longitude value |

Definition at line 42 of file [Location.h](#).

3.3.2.3 Location() [3/3]

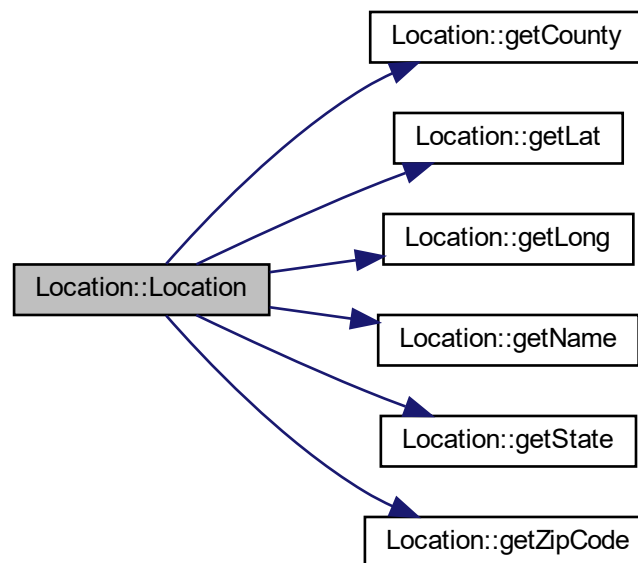
```
Location::Location (
    const Location & loc ) [inline]
```

Parameters

| | |
|---------------|--|
| <i>passes</i> | a Location object to create another object |
|---------------|--|

Definition at line 54 of file [Location.h](#).

Here is the call graph for this function:



3.3.3 Member Function Documentation

3.3.3.1 `getCounty()`

```
string Location::getCounty ( ) const
```

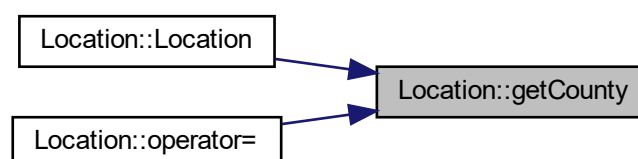
Access county

Returns

The current value of county

Definition at line 35 of file [Location.cpp](#).

Here is the caller graph for this function:



3.3.3.2 getLat()

```
float Location::getLat ( ) const
```

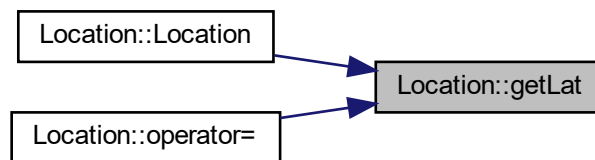
Access lat

Returns

The current Latitude value

Definition at line 37 of file [Location.cpp](#).

Here is the caller graph for this function:



3.3.3.3 getLong()

```
float Location::getLong ( ) const
```

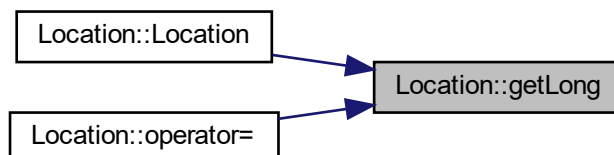
Access long

Returns

The current Longitude value

Definition at line 38 of file [Location.cpp](#).

Here is the caller graph for this function:



3.3.3.4 getName()

```
string Location::getName ( ) const
```

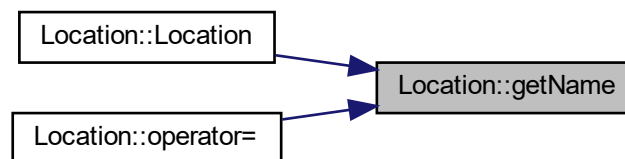
Access name

Returns

The current value of name

Definition at line 34 of file [Location.cpp](#).

Here is the caller graph for this function:



3.3.3.5 getState()

```
string Location::getState ( ) const
```

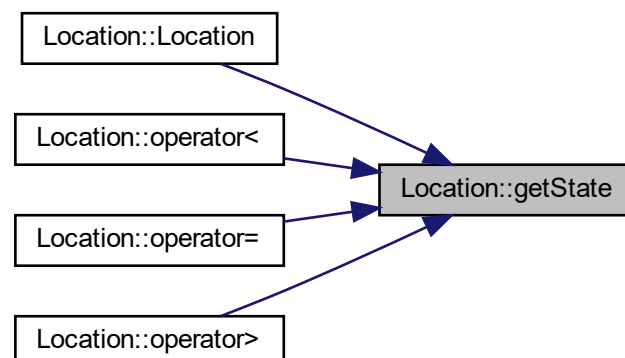
Access state

Returns

The current value of state

Definition at line 36 of file [Location.cpp](#).

Here is the caller graph for this function:



3.3.3.6 getZipCode()

```
string Location::getZipCode ( ) const
```

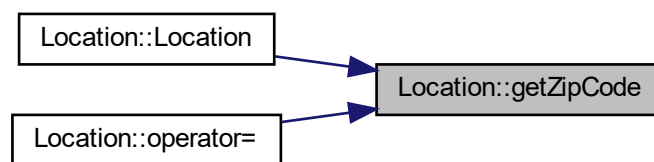
Access zipcode

Returns

The current value of zipcode

Definition at line 33 of file [Location.cpp](#).

Here is the caller graph for this function:



3.3.3.7 operator<()

```
bool Location::operator< (
    const Location & loc ) const
```

Parameters

| | |
|--------------------------|--------------|
| Location | class object |
|--------------------------|--------------|

Returns

returns the overloaded operator's greater operand

Definition at line 95 of file [Location.cpp](#).

Here is the call graph for this function:



3.3.3.8 operator=()

```
void Location::operator= (
    const Location & loc )
```

Parameters

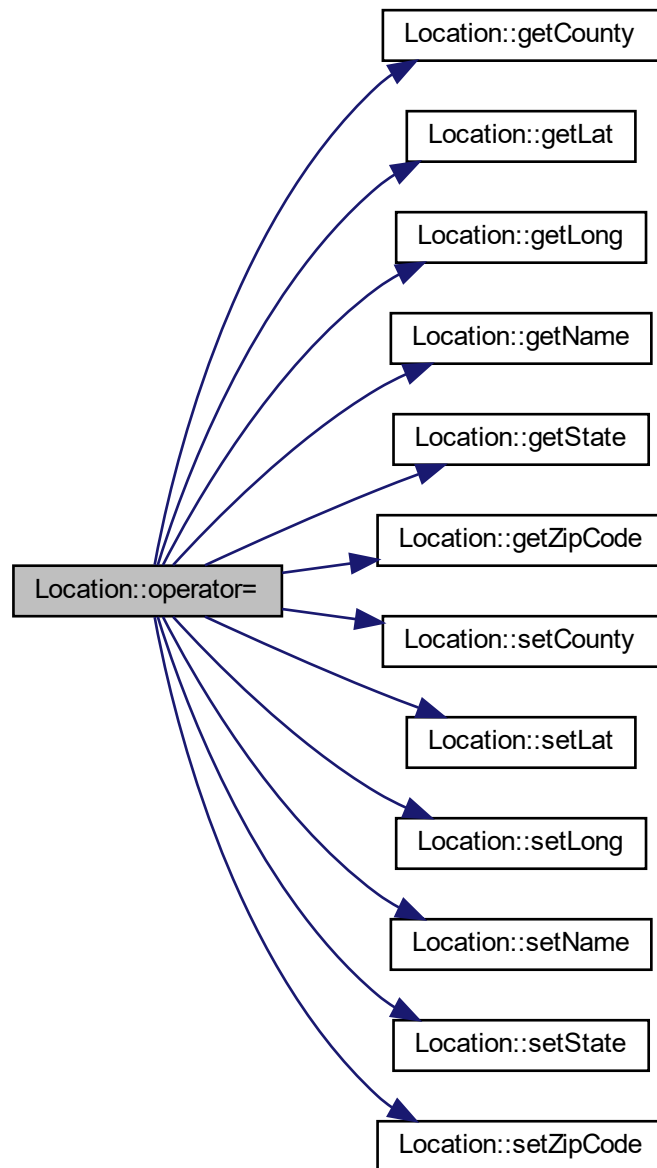
| | |
|---------------|---|
| <i>passed</i> | a Location class object |
|---------------|---|

Returns

no return value

Definition at line 67 of file [Location.cpp](#).

Here is the call graph for this function:



3.3.3.9 operator>()

```
bool Location::operator> (  
    const Location & loc ) const
```

Parameters

| | |
|--------------------------|--------------|
| Location | class object |
|--------------------------|--------------|

Returns

returns the overloaded operator's least operand

Definition at line 103 of file [Location.cpp](#).

Here is the call graph for this function:



3.3.3.10 setCounty()

```
void Location::setCounty (
    string val )
```

Set county

Parameters

| | |
|------------|------------------|
| <i>val</i> | New value to set |
|------------|------------------|

Postcondition

sets val to County

Parameters

| | |
|---------------|-----------------------------------|
| <i>passed</i> | a string value to set county name |
|---------------|-----------------------------------|

Returns

no return value

Definition at line 50 of file [Location.cpp](#).

Here is the caller graph for this function:



3.3.3.11 `setLat()`

```
void Location::setLat (
    float val )
```

Set latitude

Parameters

| | |
|------------|------------------|
| <i>val</i> | New value to set |
|------------|------------------|

Postcondition

sets val to Latitude

Parameters

| | |
|---------------|--------------------------------|
| <i>passed</i> | a string value to set latitude |
|---------------|--------------------------------|

Returns

no return value

Definition at line 58 of file [Location.cpp](#).

Here is the caller graph for this function:



3.3.3.12 setLong()

```
void Location::setLong (
    float val )
```

Set long

Parameters

| | |
|------------|------------------|
| <i>val</i> | New value to set |
|------------|------------------|

Postcondition

sets val to Longitude

Parameters

| | |
|---------------|---------------------------------------|
| <i>passed</i> | a string value to set place longitude |
|---------------|---------------------------------------|

Returns

no return value

Definition at line 62 of file [Location.cpp](#).

Here is the caller graph for this function:



3.3.3.13 setName()

```
void Location::setName (
    string val )
```

Set name

Parameters

| | |
|------------|------------------|
| <i>val</i> | New value to set |
|------------|------------------|

Postcondition

sets val to Name

Parameters

| | |
|---------------|----------------------------------|
| <i>passed</i> | a string value to set place name |
|---------------|----------------------------------|

Returns

no return value

Definition at line 46 of file [Location.cpp](#).

Here is the caller graph for this function:

**3.3.3.14 setState()**

```
void Location::setState (
    string val )
```

Set state

Parameters

| | |
|------------|------------------|
| <i>val</i> | New value to set |
|------------|------------------|

Postcondition

sets val to [State](#)

Parameters

| | |
|---------------|----------------------------------|
| <i>passed</i> | a string value to set state name |
|---------------|----------------------------------|

Returns

no return value

Definition at line 54 of file [Location.cpp](#).

Here is the caller graph for this function:

**3.3.3.15 setZipCode()**

```
void Location::setZipCode (  
    string val )
```

Set zipcode

Parameters

| | |
|------------|------------------|
| <i>val</i> | New value to set |
|------------|------------------|

Postcondition

sets val to the zipCode

Parameters

| | |
|---------------|-------------------------------|
| <i>passed</i> | a string value to set zipcode |
|---------------|-------------------------------|

Returns

no return value

Definition at line 42 of file [Location.cpp](#).

Here is the caller graph for this function:



3.3.3.16 unpack()

```
bool Location::unpack (
    DelimBuffer & buffer )
```

Parameters

| | |
|--------------------|---------------|
| <i>DelimBuffer</i> | object buffer |
|--------------------|---------------|

Returns

Returns a boolean result about whether the object piece is actually there

Definition at line 18 of file [Location.cpp](#).

Here is the call graph for this function:



Here is the caller graph for this function:



3.3.4 Friends And Related Function Documentation

3.3.4.1 operator<<

```
ostream& operator<< (  
    ostream & out,  
    const Location & loc ) [friend]
```

Parameters

| | |
|-----------------|--------------|
| <i>ostream</i> | |
| <i>Location</i> | class object |

Returns

returns the overloaded operator right hand operand

Definition at line 81 of file [Location.cpp](#).

3.3.5 Member Data Documentation

3.3.5.1 county

```
string Location::county [private]
```

Definition at line 135 of file [Location.h](#).

3.3.5.2 latitude

```
float Location::latitude [private]
```

Definition at line 137 of file [Location.h](#).

3.3.5.3 longitude

```
float Location::longitude [private]
```

Definition at line 138 of file [Location.h](#).

3.3.5.4 name

```
string Location::name [private]
```

Definition at line 134 of file [Location.h](#).

3.3.5.5 state

```
string Location::state [private]
```

Definition at line 136 of file [Location.h](#).

3.3.5.6 zipcode

```
string Location::zipcode [private]
```

Definition at line 133 of file [Location.h](#).

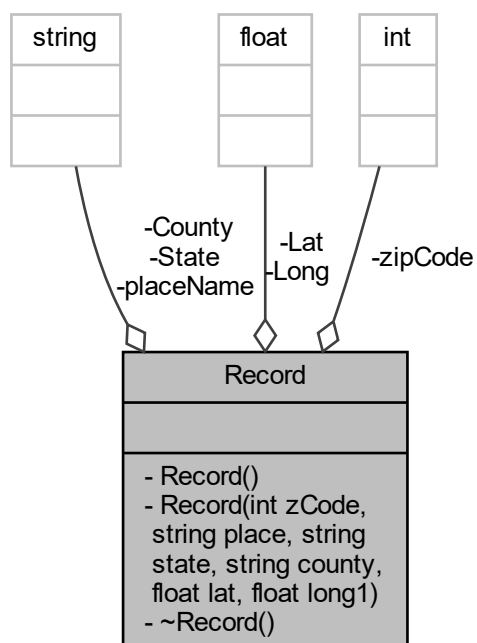
The documentation for this class was generated from the following files:

- [Location.h](#)
- [Location.cpp](#)

3.4 Record Class Reference

```
#include <Record.h>
```

Collaboration diagram for Record:



Private Member Functions

- [Record](#) ()
- [Record](#) (int zCode, string place, string state, string county, float lat, float long1)
- [~Record](#) ()

Private Attributes

- int [zipCode](#) = 0
- string [placeName](#) = ""
- string [State](#) = ""
- string [County](#) = ""
- float [Lat](#) = 0
- float [Long](#) = 0

3.4.1 Detailed Description

Definition at line 17 of file [Record.h](#).

3.4.2 Constructor & Destructor Documentation

3.4.2.1 [Record\(\)](#) [1/2]

```
Record::Record ( ) [inline], [private]
```

Definition at line 26 of file [Record.h](#).

3.4.2.2 [Record\(\)](#) [2/2]

```
Record::Record (
    int zCode,
    string place,
    string state,
    string county,
    float lat,
    float long1 ) [inline], [private]
```

Definition at line 27 of file [Record.h](#).

3.4.2.3 ~Record()

```
Record::~~Record ( ) [inline], [private]
```

Definition at line 36 of file [Record.h](#).

3.4.3 Member Data Documentation

3.4.3.1 County

```
string Record::County = " " [private]
```

Definition at line 22 of file [Record.h](#).

3.4.3.2 Lat

```
float Record::Lat = 0 [private]
```

Definition at line 23 of file [Record.h](#).

3.4.3.3 Long

```
float Record::Long = 0 [private]
```

Definition at line 24 of file [Record.h](#).

3.4.3.4 placeName

```
string Record::placeName = " " [private]
```

Definition at line 20 of file [Record.h](#).

3.4.3.5 State

```
string Record::State = " " [private]
```

Definition at line 21 of file [Record.h](#).

3.4.3.6 zipCode

```
int Record::zipCode = 0 [private]
```

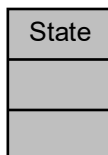
Definition at line 19 of file [Record.h](#).

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

- [Record.h](#)

3.5 State Class Reference

Collaboration diagram for State:



3.5.1 Detailed Description

Definition at line 5 of file [KeyNode.cpp](#).

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

- [KeyNode.cpp](#)

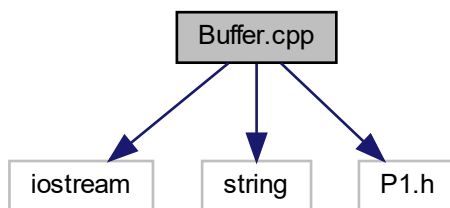
Chapter 4

File Documentation

4.1 Buffer.cpp File Reference

```
#include <iostream>
#include <string>
#include "P1.h"
```

Include dependency graph for Buffer.cpp:



4.2 Buffer.cpp

```
00001 #include <iostream>
00002 #include <string>
00003 // #include "libxl.h"
00004
00005 #include "P1.h"
00006
00007 using namespace std;
00008
00009 int DelimTextBuffer::Read(ifstream & infile)
00010 {
00011     string info;
00012
00013     int counter = 0;
00014
00015     infile.open("us_postal_codes.csv");
00016
00017
00018
00019     if (infile.is_open())
00020     {
```

```

00021         cout << "File has been opened" << endl;
00022     }
00023     else
00024     {
00025         cout << "NO FILE HAS BEEN OPENED" << endl;
00026     }
00027
00028     while (!infile.eof())
00029     {
00030         counter++;
00031         //infile >> info;
00032         getline(infile, info);
00033         cout << info << endl;
00034     }
00035     infile.close();
00036     return 0;
00037 }
00038

```

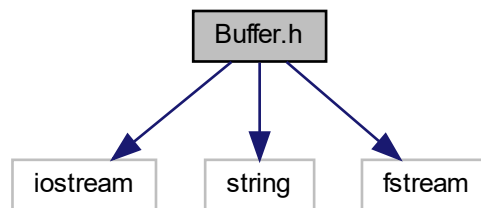
4.3 Buffer.h File Reference

```

#include <iostream>
#include <string>
#include <fstream>

```

Include dependency graph for Buffer.h:



Classes

- class [DelimTextBuffer](#)

4.4 Buffer.h

```

00001 #include <iostream>
00002 #include <string>
00003 #include <fstream>
00004
00005
00006 using namespace std;
00007
00008 class DelimTextBuffer
00009 {
00010 public:
00011     DelimTextBuffer(char Delim = ',', int maxBytes = 1000);
00012     int Read (istream & infile);
00013     //int Write(ostream & file);
00014     //int Pack (const char * str, int size = -1);
00015     //int Unpack (char * str);
00016
00017 private:

```



```

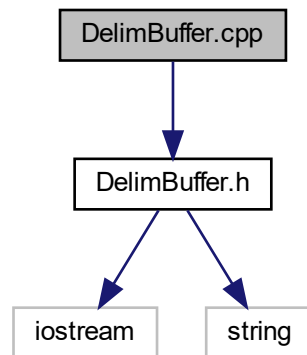
00018
00019     char Delim; // delimiter
00020     char* Buffer; // Array to hold information piece in
00021     int BufferSize; // size of field
00022     int MaxBytes; // Max number of chars in the buffer
00023     int NextByte; // packing/unpacking position in buffer
00024 };
00025

```

4.5 DelimBuffer.cpp File Reference

```
#include "DelimBuffer.h"
```

Include dependency graph for DelimBuffer.cpp:



4.6 DelimBuffer.cpp

```

00001
00010
00011 #include "DelimBuffer.h"
00012
00016 void DelimBuffer :: clear()
00017 {
00018     bufferSize = 0;
00019     nextCharIndex = 0;
00020 }
00021
00025 bool DelimBuffer :: read (istream& infile)
00026 {
00027     if (infile.fail()) return false;
00028     getline(infile, buffer, (char)infile.eof());
00029     bufferSize = buffer.length();
00030     return true;
00031 }
00032
00036 bool DelimBuffer :: unpack(string& aStr)
00037 {
00038     if (nextCharIndex > bufferSize) return false;
00039
00040     int len = -1; // length of unpacking string
00041     int start = nextCharIndex; // first character to be unpacked
00042
00043     for (int i = start; i < bufferSize; i++)
00044         if (buffer[i] == delim || buffer[i] == '\n') {len = i - start; break;}
00045
00046     if (len == -1) return false; // delimiter not found
00047     nextCharIndex += len + 1;
00048

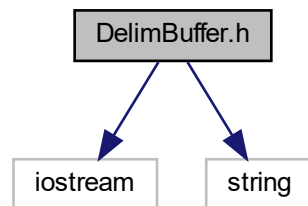
```

```
00049     aStr = buffer.substr(start, len);
00050     aStr[len] = '\\0';
00051     return true;
00052 }
```

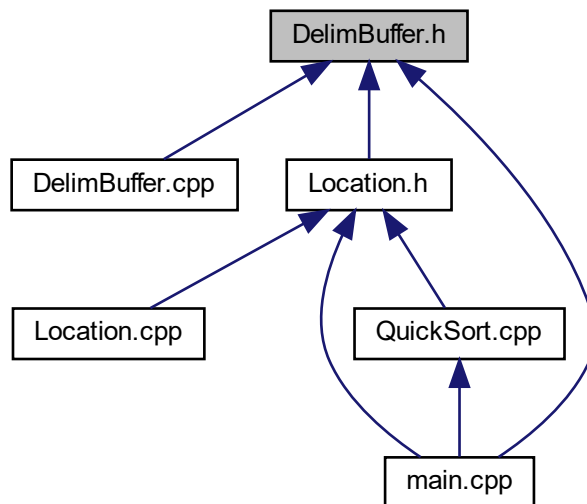
4.7 DelimBuffer.h File Reference

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

Include dependency graph for DelimBuffer.h:



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



Classes

- class [DelimBuffer](#)

4.8 DelimBuffer.h

```

00001
00012 #ifndef DELIMBUFFER_HEADER
00013 #define DELIMBUFFER_HEADER
00014
00015 #include <iostream>
00016 #include <string>
00017
00018
00019 using namespace std;
00020
00021 class DelimBuffer
00022 {
00023     public:
00024
00026         DelimBuffer(char del = ',')
00027         {
00028             delim = del;
00029             bufferSize = 0;
00030             nextCharIndex = 0;
00031         }
00032
00033         bool read(istream& infile); // read stream method
00034         bool unpack(string& aStr); // unpack the string method
00035
00036     protected:
00037         void clear();
00038
00039     private:
00040         char delim = ' ';
00041         string buffer;
00042         int bufferSize = 0;
00043         int nextCharIndex = 0; //index of next char
00044 };
00045 #endif

```

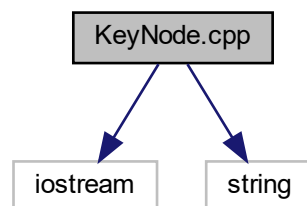
4.9 KeyNode.cpp File Reference

```

#include <iostream>
#include <string>

```

Include dependency graph for KeyNode.cpp:



Classes

- class [State](#)

4.10 KeyNode.cpp

```

00001 #include <iostream>
00002 #include <string>
00003 using namespace std;
00004
00005 class State
00006 {
00007
00008 };

```

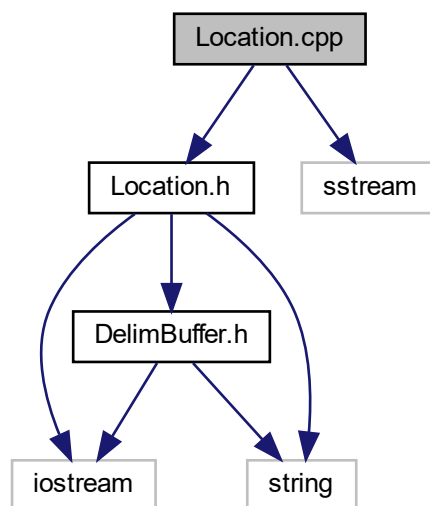
4.11 Location.cpp File Reference

```

#include "Location.h"
#include <sstream>

```

Include dependency graph for Location.cpp:



Functions

- ostream & [operator<<](#) (ostream &out, const [Location](#) &loc)
ostream operator overloaded

4.11.1 Function Documentation

4.11.1.1 operator<<()

```

ostream& operator<< (
    ostream & out,
    const Location & loc )

```

Parameters

| | |
|-----------------|--------------|
| <i>ostream</i> | |
| <i>Location</i> | class object |

Returns

returns the overloaded operator right hand operand

Definition at line 81 of file [Location.cpp](#).

4.12 Location.cpp

```

00001
00010
00011 #include "Location.h"
00012 #include <sstream>
00013
00018 bool Location :: unpack(DelimBuffer &buffer)
00019 {
00020     bool result;
00021     result = buffer.unpack(zipcode);
00022     result = result && buffer.unpack(name);
00023     result = result && buffer.unpack(state);
00024     result = result && buffer.unpack(county);
00025     string lat_str, long_str;
00026     result = result && buffer.unpack(lat_str);
00027     result = result && buffer.unpack(long_str);
00028     stringstream(lat_str) >> latitude;
00029     stringstream(long_str) >> longitude;
00030     return result;
00031 }
00032
00033 string Location :: getZipCode() const { return zipcode; } // @return Fetches zipcode value
00034 string Location :: getName() const { return name; } // @return Fetches Name value
00035 string Location :: getCounty() const { return county; } // @return Fetches County value
00036 string Location :: getState() const { return state; } // @return Fetches State value
00037 float Location :: getLat() const { return latitude; } // @return Fetches Latitude value
00038 float Location :: getLong() const { return longitude; } // @return Fetches longitude value
00039
00042 void Location :: setZipCode(string val) { zipcode = val; }
00043
00046 void Location :: setName(string val) { name = val; }
00047
00050 void Location :: setCounty(string val) { county = val; }
00051
00054 void Location :: setState(string val) { state = val; }
00055
00058 void Location :: setLat(float val) { latitude = val; }
00059
00062 void Location :: setLong(float val) { longitude = val; }
00063
00067 void Location :: operator= (const Location &loc)
00068 {
00069     setZipCode(loc.getZipCode());
00070     setName(loc.getName());
00071     setState(loc.getState());
00072     setCounty(loc.getCounty());
00073     setLat(loc.getLat());
00074     setLong(loc.getLong());
00075 }
00076
00081 ostream& operator<< (ostream& out, const Location &loc)
00082 {
00083     out << loc.getZipCode() << ' '
00084         << loc.getName() << ' '
00085         << loc.getState() << ' '
00086         << loc.getCounty() << ' '
00087         << loc.getLat() << ' '
00088         << loc.getLong() << ' ';
00089     return out;
00090 }
00091
00095 bool Location :: operator< (const Location &loc) const
00096 {
00097     return getState() < loc.getState();

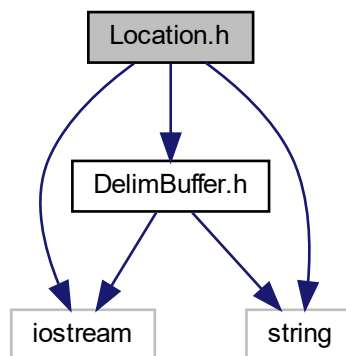
```

```
00098 }  
00099  
00103 bool Location::operator> (const Location &loc) const  
00104 {  
00105     return getState() > loc.getState();  
00106 }
```

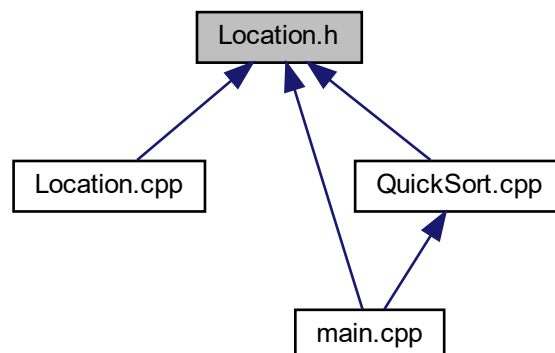
4.13 Location.h File Reference

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

Include dependency graph for Location.h:



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



Classes

- class [Location](#)

4.14 Location.h

```

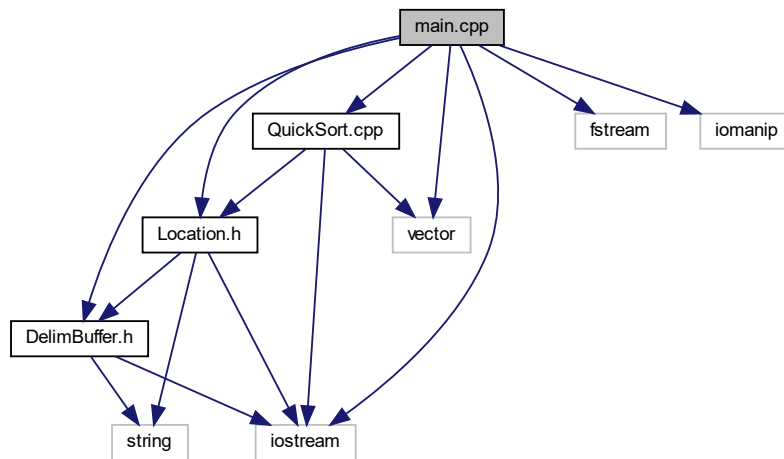
00001
00012 #ifndef LOCATION_HEADER
00013 #define LOCATION_HEADER
00014
00015 #include "DelimBuffer.h"
00016 #include <string>
00017 #include <iostream>
00018
00019 using namespace std;
00020
00021 class Location
00022 {
00023     public:
00024         Location()
00025         {
00026             zipcode = " ";
00027             name = " ";
00028             county = " ";
00029             state = " ";
00030             latitude = 0;
00031             longitude = 0;
00032         };
00033
00034         Location(string a, string b, string c, string d, float e, float f)
00035         {
00036             zipcode = a;
00037             name = b;
00038             county = c;
00039             state = d;
00040             latitude = e;
00041             longitude = f;
00042         };
00043
00044         Location(const Location& loc)
00045         {
00046             zipcode = loc.getZipCode();
00047             name = loc.getName();
00048             county = loc.getCounty();
00049             state = loc.getState();
00050             latitude = loc.getLat();
00051             longitude = loc.getLong();
00052         };
00053
00054         string getZipCode() const;
00055         void setZipCode(string val);
00056         string getName() const;
00057         void setName(string val);
00058         string getCounty() const;
00059         void setCounty(string val);
00060         string getState() const;
00061         void setState(string val);
00062         float getLat() const;
00063         void setLat(float val);
00064         float getLong() const;
00065         void setLong(float val);
00066         bool unpack(DelimBuffer &buffer);
00067         void operator= (const Location &loc);
00068         friend ostream& operator<< (ostream& out, const Location &loc);
00069
00070         bool operator< (const Location &loc) const;
00071
00072         bool operator> (const Location &loc) const;
00073
00074     private:
00075         string zipcode;
00076         string name;
00077         string county;
00078         string state;
00079         float latitude;
00080         float longitude;
00081     };
00082 #endif

```

4.15 main.cpp File Reference

```
#include "Location.h"  
#include "DelimBuffer.h"  
#include <vector>  
#include <iostream>  
#include <fstream>  
#include "QuickSort.cpp"  
#include <iomanip>
```

Include dependency graph for main.cpp:



Functions

- int [main](#) ()

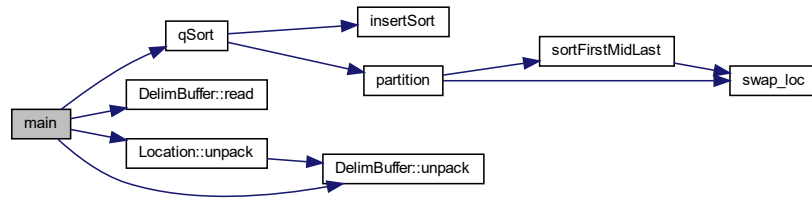
4.15.1 Function Documentation

4.15.1.1 main()

```
int main ( )
```

Definition at line 24 of file [main.cpp](#).

Here is the call graph for this function:



4.16 main.cpp

```

00001
00012
00013 #include "Location.h"
00014 #include "DelimBuffer.h"
00015 #include <vector>
00016 #include <iostream>
00017 #include <fstream>
00018 #include "QuickSort.cpp"
00019 #include <iomanip>
00020
00021
00022 using namespace std;
00023
00024 int main()
00025 {
00026     vector<Location> location;
00027     fstream infile;
00028     infile.open("us_postal_codes.csv");
00029     DelimBuffer buffer;
00030     if (!buffer.read(infile)) return 0;
00031     string field[6];
00032     for (int i = 0; i < 6; i++) { buffer.unpack(field[i]); }
00033
00034     while (1)
00035     {
00036         Location temp;
00037         if (!temp.unpack(buffer)) {break;}
00038         location.push_back(temp);
00039     }
00040
00041     int size = location.size();
00042     qSort(location, 0, size - 1);
00043
00044     cout << "+-----+" << endl;
00045     cout << '| ' << setw(5) << "State"
00046           << '| ' << setw(15) << "Westernmost"
00047           << '| ' << setw(15) << "Easternmost"
00048           << '| ' << setw(15) << "Northernmost"
00049           << '| ' << setw(15) << "Southernmost" << '| ' << endl;
00050     cout << "+-----+" << endl;
00051     int currentIndex = 0;
00052     while (currentIndex < size)
00053     {
00054         string currentState = location[currentIndex].getState();
00055         int w_most = currentIndex,
00056            e_most = currentIndex,
00057            s_most = currentIndex,
00058            n_most = currentIndex;
00059
00060         currentIndex++;
00061         while (currentState == location[currentIndex].getState())
00062         {
00063             if (location[currentIndex].getLong() > location[w_most].getLong()) w_most = currentIndex;
00064             if (location[currentIndex].getLong() < location[e_most].getLong()) e_most = currentIndex;
00065             if (location[currentIndex].getLat() > location[n_most].getLat()) n_most = currentIndex;
00066             if (location[currentIndex].getLat() < location[s_most].getLat()) s_most = currentIndex;
00067             currentIndex++;
00068             if (currentIndex == size) break;
00069         }
00070         cout << '| ' << setw(5) << location[currentIndex - 1].getState()
00071              << '| ' << setw(15) << location[w_most].getZipCode()

```

```

00072         << ' | ' << setw(15) << location[e_most].getZipCode()
00073         << ' | ' << setw(15) << location[n_most].getZipCode()
00074         << ' | ' << setw(15) << location[s_most].getZipCode() << ' | ' << endl;
00075     }
00076     cout << "+-----+" << endl;
00077 }
00078

```

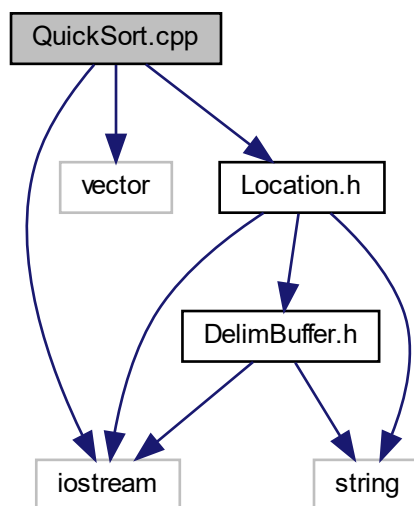
4.17 QuickSort.cpp File Reference

```

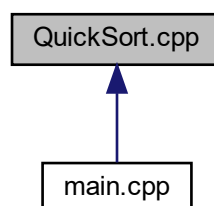
#include <iostream>
#include <vector>
#include "Location.h"

```

Include dependency graph for QuickSort.cpp:



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



Functions

- `template<class T >`
`void swap_loc (vector< T > &aVector, const int &i, const int &j)`
- `template<class T >`
`void insertSort (vector< T > &anArray, const int &first, const int &last)`
- `template<class T >`
`void sortFirstMidLast (vector< T > &anArray, const int &first, const int &mid, const int &last)`
- `template<class T >`
`int partition (vector< T > &anArray, const int &first, const int &last)`
- `void qSort (vector< Location > &anArray, const int &first, const int &last)`

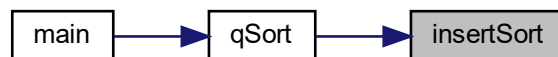
4.17.1 Function Documentation

4.17.1.1 insertSort()

```
template<class T >
void insertSort (
    vector< T > & anArray,
    const int & first,
    const int & last )
```

Definition at line 28 of file [QuickSort.cpp](#).

Here is the caller graph for this function:

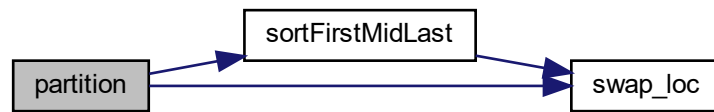


4.17.1.2 partition()

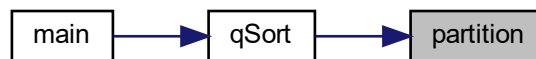
```
template<class T >
int partition (
    vector< T > & anArray,
    const int & first,
    const int & last )
```

Definition at line 75 of file [QuickSort.cpp](#).

Here is the call graph for this function:



Here is the caller graph for this function:

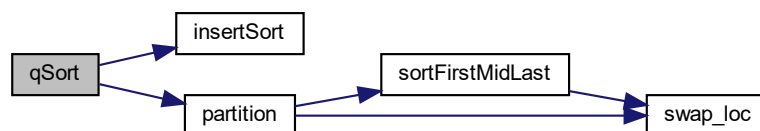


4.17.1.3 qSort()

```
void qSort (
    vector< Location > & anArray,
    const int & first,
    const int & last ) [inline]
```

Definition at line 128 of file [QuickSort.cpp](#).

Here is the call graph for this function:



Here is the caller graph for this function:

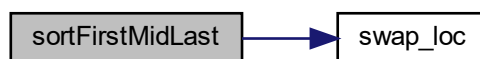


4.17.1.4 sortFirstMidLast()

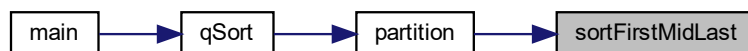
```
template<class T >
void sortFirstMidLast (
    vector< T > & anArray,
    const int & first,
    const int & mid,
    const int & last )
```

Definition at line 62 of file [QuickSort.cpp](#).

Here is the call graph for this function:



Here is the caller graph for this function:

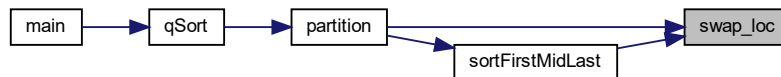


4.17.1.5 swap_loc()

```
template<class T >
void swap_loc (
    vector< T > & aVector,
    const int & i,
    const int & j )
```

Definition at line 20 of file [QuickSort.cpp](#).

Here is the caller graph for this function:



4.18 QuickSort.cpp

```
00001
00011
00012
00013 #include <iostream>
00014 #include <vector>
00015 #include "Location.h"
00016
00017 using namespace std;
00018
00019 template <class T>
00020 void swap_loc(vector<T> &aVector, const int &i, const int &j)
00021 {
00022     T temp = aVector[i];
00023     aVector[i] = aVector[j];
00024     aVector[j] = temp;
00025 }
00026
00027 template <class T>
00028 void insertSort (vector<T> &anArray, const int &first, const int &last)
00029 {
00030     // unsorted = first index of the unsorted region
00031     // loc = last index the sorted region + 1
00032     // nextItem = next item in the sorted region (the current item that is being placed to the sorted
    region)
00033     for (int unsorted = first + 1; unsorted <= last; unsorted++)
00034     {
00035         // sorted region is anArray[0..unsorted-1]
00036
00037         // get the next item
00038         T nextItem = anArray[unsorted];
00039         // get the right most index of sorted region + 1 (increase size by 1)
00040         int loc = unsorted;
00041
00042         // shift right to make room for the nextItem
00043         while ((loc > 0) && anArray[loc - 1] > nextItem)
00044         {
00045             anArray[loc] = anArray[loc - 1];
00046             loc--;
00047         }
00048
00049         // place the next item to the correct position
00050         anArray[loc] = nextItem;
00051     }
00052 }
00053
00054 /* This function sort 3 first, mid, last entries in increasing order
00055    @pre: first <= mid <= last.
00056    @post: first, mid, last entries are sorted in increasing order
00057    @param: anArray - a given array
00058           first: first index of the first half
```

```

00059         mid: last index of the first half
00060         last: last index of the second half */
00061 template <class T>
00062 void sortFirstMidLast (vector<T> &anArray, const int &first, const int &mid, const int &last)
00063 {
00064     if (anArray[first] > anArray[mid])
00065     { swap_loc(anArray, first, mid); }
00066
00067     if (anArray[mid] > anArray[last])
00068     { swap_loc(anArray, mid, last); }
00069
00070     if (anArray[first] > anArray[mid])
00071     { swap_loc(anArray, first, mid); }
00072 }
00073
00074 template <class T>
00075 int partition (vector<T> &anArray, const int &first, const int &last)
00076 {
00077     // get the middle index
00078     int mid = first + (last - first)/2;
00079     // sort first, mid, last
00080     sortFirstMidLast(anArray, first, mid, last);
00081
00082     // swap the middle index with the last - 1
00083     swap_loc(anArray, mid, last - 1);
00084
00085     // make that pivot
00086     int pivotIndex = last - 1;
00087     T pivot = anArray[pivotIndex];
00088
00089     // start checking from left and right
00090     int indLeft = first + 1;
00091     int indRight = last - 2;
00092     bool done = false;
00093     while (!done)
00094     {
00095         // look for the larger than pivot in the left
00096         while (anArray[indLeft] < pivot)
00097         {
00098             indLeft++;
00099         }
00100
00101         // look for smaller than pivot in the right
00102         while (anArray[indRight] > pivot)
00103         {
00104             indRight--;
00105         }
00106
00107         // swap them if they are in the wrong side
00108         if (indLeft < indRight)
00109         {
00110             swap_loc(anArray, indLeft, indRight);
00111             indLeft++;
00112             indRight--;
00113         }
00114
00115         // done with the current pivot
00116         else
00117         {
00118             done = true;
00119         }
00120     }
00121
00122     // swap again to place the pivot to the correct position
00123     swap_loc(anArray, pivotIndex, indLeft);
00124
00125     return indLeft;
00126 }
00127
00128 inline void qSort(vector<Location> &anArray, const int &first, const int &last)
00129 {
00130     if (last - first > 0)
00131     {
00132         // if the array size is less than 4, use insert sort
00133         if ((last - first + 1) < 4)
00134         {
00135             insertSort(anArray, first, last);
00136         }
00137         // quick sort here
00138         else
00139         {
00140             // find the pivot index
00141             int pivotIndex = partition(anArray, first, last);
00142
00143             // divide the array and use quick sort for the 2 subarrays
00144             qSort(anArray, first, pivotIndex - 1);
00145             qSort(anArray, pivotIndex + 1, last);

```

```

00146     }
00147     }
00148 }

```

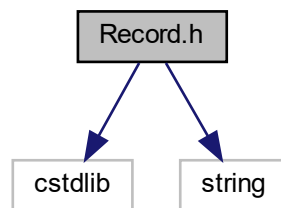
4.19 Record.h File Reference

```

#include <cstdlib>
#include <string>

```

Include dependency graph for Record.h:



Classes

- class [Record](#)

4.20 Record.h

```

00001
00010
00011 // #include <cstring>
00012 #include <cstdlib>
00013 #include <string>
00014
00015 using namespace std;
00016
00017 class Record
00018 {
00019     int zipCode = 0;
00020     string placeName = " ";
00021     string State = " ";
00022     string County = " ";
00023     float Lat = 0;
00024     float Long = 0;
00025
00026     Record(){};
00027     Record(int zCode, string place, string state, string county, float lat, float long1)
00028     {
00029         zipCode = zCode;
00030         placeName = place;
00031         State = state;
00032         County = county;
00033         Lat = lat;
00034         Long = long1;
00035     }
00036     ~Record(){};
00037
00038
00039 };

```


Index

- ~Record
 - Record, [29](#)
- Buffer
 - DelimTextBuffer, [10](#)
- buffer
 - DelimBuffer, [8](#)
- Buffer.cpp, [33](#)
- Buffer.h, [34](#)
- Buffersize
 - DelimTextBuffer, [10](#)
- bufferSize
 - DelimBuffer, [8](#)
- clear
 - DelimBuffer, [6](#)
- County
 - Record, [30](#)
- county
 - Location, [27](#)
- Delim
 - DelimTextBuffer, [10](#)
- delim
 - DelimBuffer, [8](#)
- DelimBuffer, [5](#)
 - buffer, [8](#)
 - bufferSize, [8](#)
 - clear, [6](#)
 - delim, [8](#)
 - DelimBuffer, [6](#)
 - nextCharIndex, [8](#)
 - read, [7](#)
 - unpack, [7](#)
- DelimBuffer.cpp, [35](#)
- DelimBuffer.h, [36](#), [37](#)
- DelimTextBuffer, [9](#)
 - Buffer, [10](#)
 - Buffersize, [10](#)
 - Delim, [10](#)
 - DelimTextBuffer, [9](#)
 - MaxBytes, [10](#)
 - NextByte, [11](#)
 - Read, [10](#)
- getCounty
 - Location, [15](#)
- getLat
 - Location, [16](#)
- getLong
 - Location, [16](#)
- getName
 - Location, [16](#)
- getState
 - Location, [17](#)
- getZipCode
 - Location, [18](#)
- insertSort
 - QuickSort.cpp, [45](#)
- KeyNode.cpp, [37](#), [38](#)
- Lat
 - Record, [30](#)
- latitude
 - Location, [27](#)
- Location, [11](#)
 - county, [27](#)
 - getCounty, [15](#)
 - getLat, [16](#)
 - getLong, [16](#)
 - getName, [16](#)
 - getState, [17](#)
 - getZipCode, [18](#)
 - latitude, [27](#)
 - Location, [13](#), [14](#)
 - longitude, [27](#)
 - name, [27](#)
 - operator<, [18](#)
 - operator<<, [27](#)
 - operator>, [20](#)
 - operator=, [19](#)
 - setCounty, [21](#)
 - setLat, [22](#)
 - setLong, [22](#)
 - setName, [23](#)
 - setState, [24](#)
 - setZipCode, [25](#)
 - state, [28](#)
 - unpack, [26](#)
 - zipcode, [28](#)
- Location.cpp, [38](#), [39](#)
 - operator<<, [38](#)
- Location.h, [40](#), [41](#)
- Long
 - Record, [30](#)
- longitude
 - Location, [27](#)
- main

- main.cpp, [42](#)
- main.cpp, [42](#), [43](#)
 - main, [42](#)
- MaxBytes
 - DelimTextBuffer, [10](#)
- name
 - Location, [27](#)
- NextByte
 - DelimTextBuffer, [11](#)
- nextCharIndex
 - DelimBuffer, [8](#)
- operator<
 - Location, [18](#)
- operator<<
 - Location, [27](#)
 - Location.cpp, [38](#)
- operator>
 - Location, [20](#)
- operator=
 - Location, [19](#)
- partition
 - QuickSort.cpp, [45](#)
- placeName
 - Record, [30](#)
- qSort
 - QuickSort.cpp, [46](#)
- QuickSort.cpp, [44](#), [48](#)
 - insertSort, [45](#)
 - partition, [45](#)
 - qSort, [46](#)
 - sortFirstMidLast, [47](#)
 - swap_loc, [47](#)
- Read
 - DelimTextBuffer, [10](#)
- read
 - DelimBuffer, [7](#)
- Record, [28](#)
 - ~Record, [29](#)
 - County, [30](#)
 - Lat, [30](#)
 - Long, [30](#)
 - placeName, [30](#)
 - Record, [29](#)
 - State, [30](#)
 - zipCode, [30](#)
- Record.h, [50](#)
- setCounty
 - Location, [21](#)
- setLat
 - Location, [22](#)
- setLong
 - Location, [22](#)
- setName
 - Location, [23](#)
- setState
 - Location, [24](#)
- setZipCode
 - Location, [25](#)
- sortFirstMidLast
 - QuickSort.cpp, [47](#)
- State, [31](#)
 - Record, [30](#)
- state
 - Location, [28](#)
- swap_loc
 - QuickSort.cpp, [47](#)
- unpack
 - DelimBuffer, [7](#)
 - Location, [26](#)
- zipCode
 - Record, [30](#)
- zipcode
 - Location, [28](#)