Project 2 Zip Code - Team 4

1.0

Generated by Doxygen 1.14.0

1 Bug List 1

2 Class Index 3

2.1 Class List . . . 3

3 File Index 5

3.1 File List . . . 5

4 Class Documentation 7

4.1 HeaderRecordPostalCodeItem Class Reference . . . 7

4.1.1 Detailed Description . . . 8

4.1.2 Constructor & Destructor Documentation . . . 8

4.1.2.1 HeaderRecordPostalCodeItem() [1/2] . . . 8

4.1.2.2 HeaderRecordPostalCodeItem() [2/2] . . . 9

4.1.3 Member Function Documentation . . . 9

4.1.3.1 getRecordLength() . . . 9

4.1.3.2 getZip() . . . 10

4.1.3.3 getPlace() . . . 10

4.1.3.4 getState() . . . 10

4.1.3.5 getCounty() . . . 10

4.1.3.6 getLatitude() . . . 11

4.1.3.7 getLongitude() . . . 11

4.1.3.8 setRecordLength() . . . 11

4.1.3.9 setZip() . . . 11

4.1.3.10 setPlace() . . . 12

4.1.3.11 setState() . . . 12

4.1.3.12 setCounty() . . . 12

4.1.3.13 setLatitude() . . . 13

4.1.3.14 setLongitude() . . . 13

4.1.3.15 printInfo() . . . 14

4.2 HeaderRecordPostalList Class Reference . . . 14

4.2.1 Detailed Description . . . 15

4.2.2 Constructor & Destructor Documentation . . . 15

4.2.2.1 HeaderRecordPostalList() . . . 15

4.2.3 Member Function Documentation . . . 15

4.2.3.1 addItem() . . . 15

4.2.3.2 getItem() . . . 15

4.2.3.3 findByZip() . . . 16

4.2.3.4 size() . . . 16

4.2.3.5 printAll() . . . 17

4.2.3.6 printSortedByZip() . . . 17

4.2.3.7 printSortedByState() . . . 17

4.3 PostalCodeItem Class Reference . . . 18

4.3.1 Detailed Description . . . 19
4.3.2 Constructor & Destructor Documentation . . . 19
4.3.2.1 PostalCodeItem() [1/2] . . . 19
4.3.2.2 PostalCodeItem() [2/2] . . . 20
4.3.3 Member Function Documentation . . . 20
4.3.3.1 getZip() . . . 20
4.3.3.2 getPlace() . . . 20
4.3.3.3 getState() . . . 21
4.3.3.4 getCounty() . . . 21
4.3.3.5 getLatitude() . . . 21
4.3.3.6 getLongitude() . . . 22
4.3.3.7 setZip() . . . 22
4.3.3.8 setPlace() . . . 22
4.3.3.9 setState() . . . 22
4.3.3.10 setCounty() . . . 23
4.3.3.11 setLatitude() . . . 23
4.3.3.12 setLongitude() . . . 23
4.3.3.13 printInfo() . . . 24
4.4 PostalList Class Reference . . . 24
4.4.1 Detailed Description . . . 25
4.4.2 Constructor & Destructor Documentation . . . 25
4.4.2.1 PostalList() . . . 25
4.4.3 Member Function Documentation . . . 25
4.4.3.1 addItem() . . . 25
4.4.3.2 getItem() . . . 25
4.4.3.3 findByZip() . . . 26
4.4.3.4 size() . . . 26
4.4.3.5 printAll() . . . 27
4.4.3.6 printSortedByZip() . . . 27
4.4.3.7 printSortedByState() . . . 27

5 File Documentation **29**

5.1 source/HeaderRecordPostalCodeItem.cpp File Reference . . . 29
5.1.1 Detailed Description . . . 30
5.2 HeaderRecordPostalCodeItem.cpp . . . 30
5.3 source/HeaderRecordPostalCodeItem.h File Reference . . . 32
5.3.1 Detailed Description . . . 33
5.4 HeaderRecordPostalCodeItem.h . . . 33
5.5 source/HeaderRecordPostalList.cpp File Reference . . . 34
5.5.1 Detailed Description . . . 35
5.6 HeaderRecordPostalList.cpp . . . 35
5.7 source/HeaderRecordPostalList.h File Reference . . . 36

5.7.1 Detailed Description . . . 37
5.8 HeaderRecordPostalList.h . . . 37
5.9 source/main1.cpp File Reference . . . 38
5.9.1 Detailed Description . . . 39
5.9.2 Function Documentation . . . 39
5.9.2.1 main() . . . 39
5.10 main1.cpp . . . 40
5.11 source/main2.cpp File Reference . . . 40
5.11.1 Detailed Description . . . 41
5.11.2 Function Documentation . . . 42
5.11.2.1 main() . . . 42
5.12 main2.cpp . . . 42
5.13 source/main3.cpp File Reference . . . 43
5.13.1 Detailed Description . . . 43
5.13.2 Function Documentation . . . 44
5.13.2.1 main() . . . 44
5.14 main3.cpp . . . 44
5.15 source/main4.cpp File Reference . . . 45
5.15.1 Detailed Description . . . 45
5.15.2 Function Documentation . . . 46
5.15.2.1 main() . . . 46
5.16 main4.cpp . . . 46
5.17 source/make_index.cpp File Reference . . . 47
5.17.1 Detailed Description . . . 47
5.17.2 Function Documentation . . . 47
5.17.2.1 split() . . . 47
5.17.2.2 printRecord() . . . 48
5.17.2.3 main() . . . 48
5.17.3 Variable Documentation . . . 49
5.17.3.1 DATA_FILE . . . 49
5.17.3.2 INDEX_FILE . . . 49
5.18 make_index.cpp . . . 49
5.19 source/PostalCodeItem.cpp File Reference . . . 51
5.19.1 Detailed Description . . . 52
5.20 PostalCodeItem.cpp . . . 52
5.21 source/PostalCodeItem.h File Reference . . . 53
5.21.1 Detailed Description . . . 54
5.22 PostalCodeItem.h . . . 55
5.23 source/PostalList.cpp File Reference . . . 55
5.23.1 Detailed Description . . . 56
5.24 PostalList.cpp . . . 57
5.25 source/PostalList.h File Reference . . . 58

5.25.1 Detailed Description . . . 59
5.26 PostalList.h . . . 59
5.27 source/readHeaderRecordPostalCodeBuffer.cpp File Reference . . . 60
5.27.1 Detailed Description . . . 61
5.27.2 Function Documentation . . . 61
5.27.2.1 inputCSVtoList() . . . 61
5.28 readHeaderRecordPostalCodeBuffer.cpp . . . 61
5.29 source/readPostalCodeBuffer.cpp File Reference . . . 62
5.29.1 Detailed Description . . . 63
5.29.2 Function Documentation . . . 64
5.29.2.1 inputCSVtoList() . . . 64
5.30 readPostalCodeBuffer.cpp . . . 64

Chapter 1

Bug List

File main1.cpp

None that we know of right now.

File main2.cpp

None that we know of right now.

File main3.cpp

None that we know of right now.

File main4.cpp

None that we know of right now.

File readPostalCodeBuffer.cpp

None that we know of right now.

Chapter 2

Class Index

2.1 Class List

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

HeaderRecordPostalCodeItem 7
HeaderRecordPostalList 14
PostalCodeItem 18
PostalList 24

Chapter 3

File Index

3.1 File List

Here is a list of all files with brief descriptions:

source/HeaderRecordPostalCodeItem.cpp
Defines and manages individual HeaderRecord postal code items 29
source/HeaderRecordPostalCodeItem.h
Defines and manages individual HeaderRecord postal code items 32
source/HeaderRecordPostalList.cpp
Defines and manages HeaderRecord postal code list structures 34
source/HeaderRecordPostalList.h
Defines and manages HeaderRecord postal code list structures 36
source/main1.cpp
Our first main file for "Zip Code Group 4 Project 2.0" that output the postal sorted by zip code . 38
source/main2.cpp
Our second main file for "Zip Code Group 4 Project 2.0" that output the postal sorted by state . 40
source/main3.cpp
Our third main file for "Zip Code Group 4 Project 2.0" that output the postal sorted by zip code . 43
source/main4.cpp
Our third main file for "Zip Code Group 4 Project 2.0" that output the postal sorted by zip code . 45
source/make_index.cpp
Builds or loads an index for fast lookup of postal code records from a CSV file 47
source/PostalCodeItem.cpp
Implementation of the PostalCodeItem class representing a postal code entry 51
source/PostalCodeItem.h
Defines the PostalCodeItem class for representing postal code records 53
source/PostalList.cpp
@ brief Implementation of the PostalList class for managing a collection of PostalCodeItem objects 55
source/PostalList.h
Defines the PostalList class for managing collections of postal codes 58
source/readHeaderRecordPostalCodeBuffer.cpp
Reads a CSV file containing postal code data and populates a HeaderRecordPostalList 60
source/readPostalCodeBuffer.cpp
Utility functions for reading postal code data from a CSV file 62

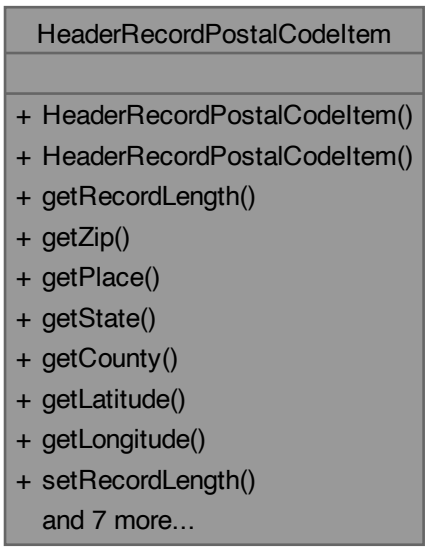
Chapter 4

Class Documentation

4.1 HeaderRecordPostalCodeItem Class Reference

`#include <HeaderRecordPostalCodeItem.h>`

Collaboration diagram for HeaderRecordPostalCodeItem:



Public Member Functions

- HeaderRecordPostalCodeItem ()

  Default constructor initializing member variables to default values.
- HeaderRecordPostalCodeItem (int r, int z, const string &p, const string &s, const string &c, double lat, double lon)

Parameterized constructor to initialize a HeaderRecordPostalCodeItem with specific values.

- int getRecordLength () const
- int getZip () const

  Get the ZIP code of the postal code item.
- string getPlace () const

  Get the place name of the postal code item.
- string getState () const

  Get the state name of the postal code item.
- string getCounty () const

  Get the county name of the postal code item.
- double getLatitude () const

  Get the latitude of the postal code item.
- double getLongitude () const

  Get the longitude of the postal code item.
- void setRecordLength (int newRecordLength)
- void setZip (int newZip)

  Set the ZIP code of the postal code item.
- void setPlace (const string &newPlace)

  Set the place name of the postal code item.
- void setState (const string &newState)

  Set the state name of the postal code item.
- void setCounty (const string &newCounty)

  Set the county name of the postal code item.
- void setLatitude (double newLat)

  Set the latitude of the postal code item.
- void setLongitude (double newLon)

  Set the longitude of the postal code item.
- void printInfo () const

  Print the postal code item's information in a formatted manner.

4.1.1 Detailed Description

Definition at line 13 of file HeaderRecordPostalCodeItem.h.

4.1.2 Constructor & Destructor Documentation

4.1.2.1 HeaderRecordPostalCodeItem() [1/2]

```
HeaderRecordPostalCodeItem::HeaderRecordPostalCodeItem ()
```

Default constructor initializing member variables to default values.

zip is set to 0, place, state, and county are set to empty strings, and latitude and longitude are set to 0.0. This ensures that a HeaderRecordPostalCodeItem object starts with a known state.

Note

This constructor can be used to create an empty HeaderRecordPostalCodeItem object, which can later be populated with actual data using the setter methods.

See also

HeaderRecordPostalCodeItem(int, const string&, const string&, const string&, double, double)

setZip(int)

setPlace(const string&)

setState(const string&)

setCounty(const string&)

setLatitude(double)

setLongitude(double)

printInfo() const

Definition at line 31 of file HeaderRecordPostalCodeItem.cpp.

4.1.2.2 HeaderRecordPostalCodeItem() [2/2]

```
HeaderRecordPostalCodeItem::HeaderRecordPostalCodeItem (
            int r,
            int z,
            const string & p,
            const string & s,
            const string & c,
            double lat,
            double lon)
```

Parameterized constructor to initialize a HeaderRecordPostalCodeItem with specific values.

Parameters

z	The ZIP code (integer).
p	The place name (string).
s	The state name (string).
c	The county name (string).
lat	The latitude (double).
lon	The longitude (double). This constructor allows for the creation of a fully initialized HeaderRecordPostalCodeItem object.

Note

Ensure that the provided values are valid and meaningful for the postal code entry.

Definition at line 53 of file HeaderRecordPostalCodeItem.cpp.

4.1.3 Member Function Documentation

4.1.3.1 getRecordLength()

```
int HeaderRecordPostalCodeItem::getRecordLength () const
```

Definition at line 64 of file HeaderRecordPostalCodeItem.cpp.

4.1.3.2 getZip()

```
int HeaderRecordPostalCodeItem::getZip () const
```

Get the ZIP code of the postal code item.

Returns

The ZIP code as an integer.

Definition at line 73 of file HeaderRecordPostalCodeItem.cpp.

4.1.3.3 getPlace()

```
string HeaderRecordPostalCodeItem::getPlace () const
```

Get the place name of the postal code item.

Returns

The place name as a string.

Definition at line 82 of file HeaderRecordPostalCodeItem.cpp.

4.1.3.4 getState()

```
string HeaderRecordPostalCodeItem::getState () const
```

Get the state name of the postal code item.

Returns

The state name as a string.

Definition at line 91 of file HeaderRecordPostalCodeItem.cpp.

4.1.3.5 getCounty()

```
string HeaderRecordPostalCodeItem::getCounty () const
```

Get the county name of the postal code item.

Returns

The county name as a string.

Note

County names may vary in format and length depending on the region. Ensure that the county name is correctly formatted for display or processing.

Definition at line 102 of file HeaderRecordPostalCodeItem.cpp.

4.1.3.6 getLatitude()

```
double HeaderRecordPostalCodeItem::getLatitude () const
```

Get the latitude of the postal code item.

Returns

The latitude as a double.

Note

Latitude values are typically in the range of -90 to 90 degrees.

Definition at line 112 of file HeaderRecordPostalCodeItem.cpp.

4.1.3.7 getLongitude()

```
double HeaderRecordPostalCodeItem::getLongitude () const
```

Get the longitude of the postal code item.

Returns

The longitude as a double.

Note

Longitude values are typically in the range of -180 to 180 degrees.

Definition at line 122 of file HeaderRecordPostalCodeItem.cpp.

4.1.3.8 setRecordLength()

```
void HeaderRecordPostalCodeItem::setRecordLength (
            int newRecordLength)
```

Definition at line 127 of file HeaderRecordPostalCodeItem.cpp.

4.1.3.9 setZip()

```
void HeaderRecordPostalCodeItem::setZip (
            int newZip)
```

Set the ZIP code of the postal code item.

Parameters

newZip	The new ZIP code to be set (integer).

Note

Ensure that the new ZIP code is a valid integer value.

Definition at line 137 of file HeaderRecordPostalCodeItem.cpp.

4.1.3.10 setPlace()

```
void HeaderRecordPostalCodeItem::setPlace (
            const string & newPlace)
```

Set the place name of the postal code item.

Parameters

newPlace	The new place name to be set (string).

Note

Ensure that the new place name is a valid string value.

Definition at line 147 of file HeaderRecordPostalCodeItem.cpp.

4.1.3.11 setState()

```
void HeaderRecordPostalCodeItem::setState (
            const string & newState)
```

Set the state name of the postal code item.

Parameters

newState	The new state name to be set (string).

Note

Ensure that the new state name is a valid string value.

Definition at line 157 of file HeaderRecordPostalCodeItem.cpp.

4.1.3.12 setCounty()

```
void HeaderRecordPostalCodeItem::setCounty (
            const string & newCounty)
```

Set the county name of the postal code item.

Parameters

newCounty	The new county name to be set (string).

Note

Ensure that the new county name is a valid string value.

Definition at line 167 of file HeaderRecordPostalCodeItem.cpp.

4.1.3.13 setLatitude()

```
void HeaderRecordPostalCodeItem::setLatitude (
            double newLat)
```

Set the latitude of the postal code item.

Parameters

newLat	The new latitude to be set (double).

Note

Ensure that the new latitude is within the valid range of -90 to 90 degrees. Invalid latitude values may lead to incorrect geographical representations.

Definition at line 178 of file HeaderRecordPostalCodeItem.cpp.

4.1.3.14 setLongitude()

```
void HeaderRecordPostalCodeItem::setLongitude (
            double newLon)
```

Set the longitude of the postal code item.

Parameters

newLon	The new longitude to be set (double).

Note

Ensure that the new longitude is within the valid range of -180 to 180 degrees. Invalid longitude values may lead to incorrect geographical representations.

Definition at line 189 of file HeaderRecordPostalCodeItem.cpp.

4.1.3.15 printInfo()

```
void HeaderRecordPostalCodeItem::printInfo () const
```

Print the postal code item's information in a formatted manner.

The information includes ZIP code, place name, state, county, latitude, and longitude. The output is aligned in columns for better readability.

Note

This method uses standard output (cout) to display the information.

Definition at line 200 of file HeaderRecordPostalCodeItem.cpp.

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

- source/HeaderRecordPostalCodeItem.h
- source/HeaderRecordPostalCodeItem.cpp

4.2 HeaderRecordPostalList Class Reference

```
#include <HeaderRecordPostalList.h>
```

Collaboration diagram for HeaderRecordPostalList:

Public Member Functions

- HeaderRecordPostalList ()=default
- void addItem (const HeaderRecordPostalCodeItem &item)

  Add a HeaderRecordPostalCodeItem to the list.
- HeaderRecordPostalCodeItem getItem (int index) const

  Get a HeaderRecordPostalCodeItem by index.
- const HeaderRecordPostalCodeItem * findByZip (int zip) const

  Find a HeaderRecordPostalCodeItem by its ZIP code.
- int size () const

  Get the number of items in the list.
- void printAll () const

  Print all HeaderRecordPostalCodeItems in the list.
- void printSortedByZip () const

  Print HeaderRecordPostalCodeItems sorted by ZIP code.
- void printSortedByState () const

  Print HeaderRecordPostalCodeItems sorted by state and then by ZIP code.

4.2.1 Detailed Description

Definition at line 16 of file HeaderRecordPostalList.h.

4.2.2 Constructor & Destructor Documentation

4.2.2.1 HeaderRecordPostalList()

```
HeaderRecordPostalList::HeaderRecordPostalList () [default]
```

4.2.3 Member Function Documentation

4.2.3.1 addItem()

```
void HeaderRecordPostalList::addItem (
            const HeaderRecordPostalCodeItem & item)
```

Add a HeaderRecordPostalCodeItem to the list.

Parameters

item	The HeaderRecordPostalCodeItem to be added.

Definition at line 20 of file HeaderRecordPostalList.cpp.

4.2.3.2 getItem()

```
HeaderRecordPostalCodeItem HeaderRecordPostalList::getItem (
            int index) const
```

Get a HeaderRecordPostalCodeItem by index.

Parameters

index	The index of the item to retrieve.

Returns

The HeaderRecordPostalCodeItem at the specified index.

Exceptions

out_of_range	if the index is invalid.

Definition at line 31 of file HeaderRecordPostalList.cpp.

4.2.3.3 findByZip()

```
const HeaderRecordPostalCodeItem * HeaderRecordPostalList::findByZip (
            int zip) const
```

Find a HeaderRecordPostalCodeItem by its ZIP code.

Parameters

zip	The ZIP code to search for.

Returns

A pointer to the HeaderRecordPostalCodeItem if found, nullptr otherwise.

Note

The returned pointer is valid as long as the HeaderRecordPostalList object exists and is not modified.

Definition at line 46 of file HeaderRecordPostalList.cpp.

4.2.3.4 size()

```
int HeaderRecordPostalList::size () const
```

Get the number of items in the list.

Returns

The number of HeaderRecordPostalCodeItem objects in the list.

Definition at line 62 of file HeaderRecordPostalList.cpp.

4.2.3.5 printAll()

```
void HeaderRecordPostalList::printAll () const
```

Print all HeaderRecordPostalCodeItems in the list.

Each item's information is printed followed by a separator line.

Note

The order of items is the same as the order they were added.

Definition at line 72 of file HeaderRecordPostalList.cpp.

4.2.3.6 printSortedByZip()

```
void HeaderRecordPostalList::printSortedByZip () const
```

Print HeaderRecordPostalCodeItems sorted by ZIP code.

Each item's information is printed followed by a separator line.

Note

Items are sorted in ascending order by ZIP code.

Definition at line 86 of file HeaderRecordPostalList.cpp.

4.2.3.7 printSortedByState()

```
void HeaderRecordPostalList::printSortedByState () const
```

Print HeaderRecordPostalCodeItems sorted by state and then by ZIP code.

Each item's information is printed followed by a separator line.

Note

Items are sorted first by state (alphabetically) and then by ZIP code (numerically) within each state.

Definition at line 109 of file HeaderRecordPostalList.cpp.

References HeaderRecordPostalCodeItem::getState().

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

- source/HeaderRecordPostalList.h
- source/HeaderRecordPostalList.cpp

4.3 PostalCodeItem Class Reference

`#include <PostalCodeItem.h>`

Collaboration diagram for PostalCodeItem:

Public Member Functions

- PostalCodeItem ()

  Default constructor initializing member variables to default values.
- PostalCodeItem (int z, const string &p, const string &s, const string &c, double lat, double lon)

  Parameterized constructor to initialize a PostalCodeItem with specific values.
- int getZip () const

  Get the ZIP code of the postal code item.
- string getPlace () const

  Get the place name of the postal code item.
- string getState () const

  Get the state name of the postal code item.
- string getCounty () const

  Get the county name of the postal code item.
- double getLatitude () const

  Get the latitude of the postal code item.
- double getLongitude () const

Get the longitude of the postal code item.

- void setZip (int newZip)

  Set the ZIP code of the postal code item.
- void setPlace (const string &newPlace)

  Set the place name of the postal code item.
- void setState (const string &newState)

  Set the state name of the postal code item.
- void setCounty (const string &newCounty)

  Set the county name of the postal code item.
- void setLatitude (double newLat)

  Set the latitude of the postal code item.
- void setLongitude (double newLon)

  Set the longitude of the postal code item.
- void printInfo () const

  Print the postal code item's information in a formatted manner.

4.3.1 Detailed Description

Definition at line 24 of file PostalCodeItem.h.

4.3.2 Constructor & Destructor Documentation

4.3.2.1 PostalCodeItem() [1/2]

```
PostalCodeItem::PostalCodeItem ()
```

Default constructor initializing member variables to default values.

zip is set to 0, place, state, and county are set to empty strings, and latitude and longitude are set to 0.0. This ensures that a PostalCodeItem object starts with a known state.

Note

This constructor can be used to create an empty PostalCodeItem object, which can later be populated with actual data using the setter methods.

See also

PostalCodeItem(int, const string&, const string&, const string&, double, double)

setZip(int)

setPlace(const string&)

setState(const string&)

setCounty(const string&)

setLatitude(double)

setLongitude(double)

printInfo() const

Definition at line 46 of file PostalCodeItem.cpp.

4.3.2.2 PostalCodeItem() [2/2]

```
PostalCodeItem::PostalCodeItem (
            int z,
            const string & p,
            const string & s,
            const string & c,
            double lat,
            double lon)
```

Parameterized constructor to initialize a PostalCodeItem with specific values.

Parameters

z	The ZIP code (integer).
p	The place name (string).
s	The state name (string).
c	The county name (string).
lat	The latitude (double).
lon	The longitude (double). This constructor allows for the creation of a fully initialized PostalCodeItem object.

Note

Ensure that the provided values are valid and meaningful for the postal code entry.

Definition at line 67 of file PostalCodeItem.cpp.

4.3.3 Member Function Documentation

4.3.3.1 getZip()

```
int PostalCodeItem::getZip () const
```

Get the ZIP code of the postal code item.

Returns

The ZIP code as an integer.

Definition at line 81 of file PostalCodeItem.cpp.

4.3.3.2 getPlace()

```
string PostalCodeItem::getPlace () const
```

Get the place name of the postal code item.

Returns

The place name as a string.

Definition at line 90 of file PostalCodeItem.cpp.

4.3.3.3 getState()

```
string PostalCodeItem::getState () const
```

Get the state name of the postal code item.

Returns

The state name as a string.

Definition at line 99 of file PostalCodeItem.cpp.

4.3.3.4 getCounty()

```
string PostalCodeItem::getCounty () const
```

Get the county name of the postal code item.

Returns

The county name as a string.

Note

County names may vary in format and length depending on the region. Ensure that the county name is correctly formatted for display or processing.

Definition at line 110 of file PostalCodeItem.cpp.

4.3.3.5 getLatitude()

```
double PostalCodeItem::getLatitude () const
```

Get the latitude of the postal code item.

Returns

The latitude as a double.

Note

Latitude values are typically in the range of -90 to 90 degrees.

Definition at line 120 of file PostalCodeItem.cpp.

4.3.3.6 getLongitude()

```
double PostalCodeItem::getLongitude () const
```

Get the longitude of the postal code item.

Returns

The longitude as a double.

Note

Longitude values are typically in the range of -180 to 180 degrees.

Definition at line 130 of file PostalCodeItem.cpp.

4.3.3.7 setZip()

```
void PostalCodeItem::setZip (
            int newZip)
```

Set the ZIP code of the postal code item.

Parameters

newZip	The new ZIP code to be set (integer).

Note

Ensure that the new ZIP code is a valid integer value.

Definition at line 140 of file PostalCodeItem.cpp.

4.3.3.8 setPlace()

```
void PostalCodeItem::setPlace (
            const string & newPlace)
```

Set the place name of the postal code item.

Parameters

newPlace	The new place name to be set (string).

Note

Ensure that the new place name is a valid string value.

Definition at line 150 of file PostalCodeItem.cpp.

4.3.3.9 setState()

```
void PostalCodeItem::setState (
            const string & newState)
```

Set the state name of the postal code item.

Parameters

newState	The new state name to be set (string).

Note

Ensure that the new state name is a valid string value.

Definition at line 160 of file PostalCodeItem.cpp.

4.3.3.10 setCounty()

```
void PostalCodeItem::setCounty (
            const string & newCounty)
```

Set the county name of the postal code item.

Parameters

newCounty	The new county name to be set (string).

Note

Ensure that the new county name is a valid string value.

Definition at line 170 of file PostalCodeItem.cpp.

4.3.3.11 setLatitude()

```
void PostalCodeItem::setLatitude (
            double newLat)
```

Set the latitude of the postal code item.

Parameters

newLat	The new latitude to be set (double).

Note

Ensure that the new latitude is within the valid range of -90 to 90 degrees. Invalid latitude values may lead to incorrect geographical representations.

Definition at line 181 of file PostalCodeItem.cpp.

4.3.3.12 setLongitude()

```
void PostalCodeItem::setLongitude (
            double newLon)
```

Set the longitude of the postal code item.

Parameters

newLon	The new longitude to be set (double).

Note

Ensure that the new longitude is within the valid range of -180 to 180 degrees. Invalid longitude values may lead to incorrect geographical representations.

Definition at line 192 of file PostalCodeItem.cpp.

4.3.3.13 printInfo()

```
void PostalCodeItem::printInfo () const
```

Print the postal code item's information in a formatted manner.

The information includes ZIP code, place name, state, county, latitude, and longitude. The output is aligned in columns for better readability.

Note

This method uses standard output (cout) to display the information.

Definition at line 203 of file PostalCodeItem.cpp.

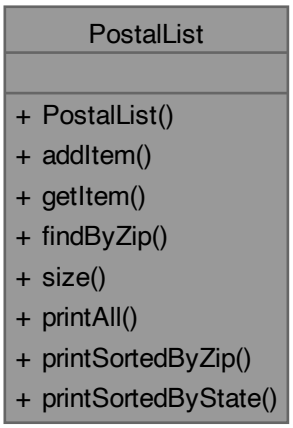
The documentation for this class was generated from the following files:

- source/PostalCodeItem.h
- source/PostalCodeItem.cpp

4.4 PostalList Class Reference

```
#include <PostalList.h>
```

Collaboration diagram for PostalList:



Public Member Functions

- PostalList ()=default
- void addItem (const PostalCodeItem &item)

  Add a PostalCodeItem to the list.
- PostalCodeItem getItem (int index) const

  Get a PostalCodeItem by index.
- const PostalCodeItem * findByZip (int zip) const

  Find a PostalCodeItem by its ZIP code.
- int size () const

  Get the number of items in the list.
- void printAll () const

  Print all PostalCodeItems in the list.
- void printSortedByZip () const

  Print PostalCodeItems sorted by ZIP code.
- void printSortedByState () const

  Print PostalCodeItems sorted by state and then by ZIP code.

4.4.1 Detailed Description

Definition at line 23 of file PostalList.h.

4.4.2 Constructor & Destructor Documentation

4.4.2.1 PostalList()

```
PostalList::PostalList ()  [default]
```

4.4.3 Member Function Documentation

4.4.3.1 addItem()

```
void PostalList::addItem (
            const PostalCodeItem & item)
```

Add a PostalCodeItem to the list.

Parameters

item	The PostalCodeItem to be added.

Definition at line 26 of file PostalList.cpp.

4.4.3.2 getItem()

```
PostalCodeItem PostalList::getItem (
            int index) const
```

Get a PostalCodeItem by index.

Parameters

index	The index of the item to retrieve.

Returns

The PostalCodeItem at the specified index.

Exceptions

out_of_range	if the index is invalid.

Definition at line 37 of file PostalList.cpp.

4.4.3.3 findByZip()

```
const PostalCodeItem * PostalList::findByZip (
            int zip) const
```

Find a PostalCodeItem by its ZIP code.

Parameters

zip	The ZIP code to search for.

Returns

A pointer to the PostalCodeItem if found, nullptr otherwise.

Note

The returned pointer is valid as long as the PostalList object exists and is not modified.

Definition at line 52 of file PostalList.cpp.

4.4.3.4 size()

```
int PostalList::size () const
```

Get the number of items in the list.

Returns

The number of PostalCodeItem objects in the list.

Definition at line 68 of file PostalList.cpp.

4.4.3.5 printAll()

```
void PostalList::printAll () const
```

Print all PostalCodeItems in the list.

Each item's information is printed followed by a separator line.

Note

The order of items is the same as the order they were added.

Definition at line 78 of file PostalList.cpp.

4.4.3.6 printSortedByZip()

```
void PostalList::printSortedByZip () const
```

Print PostalCodeItems sorted by ZIP code.

Each item's information is printed followed by a separator line.

Note

Items are sorted in ascending order by ZIP code.

Definition at line 92 of file PostalList.cpp.

4.4.3.7 printSortedByState()

```
void PostalList::printSortedByState () const
```

Print PostalCodeItems sorted by state and then by ZIP code.

Each item's information is printed followed by a separator line.

Note

Items are sorted first by state (alphabetically) and then by ZIP code (numerically) within each state.

Definition at line 115 of file PostalList.cpp.

References PostalCodeItem::getState().

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

- source/PostalList.h
- source/PostalList.cpp

Chapter 5

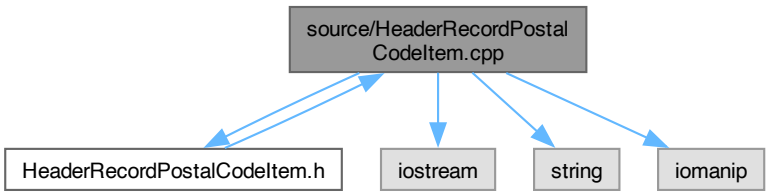
File Documentation

5.1 source/HeaderRecordPostalCodeItem.cpp File Reference

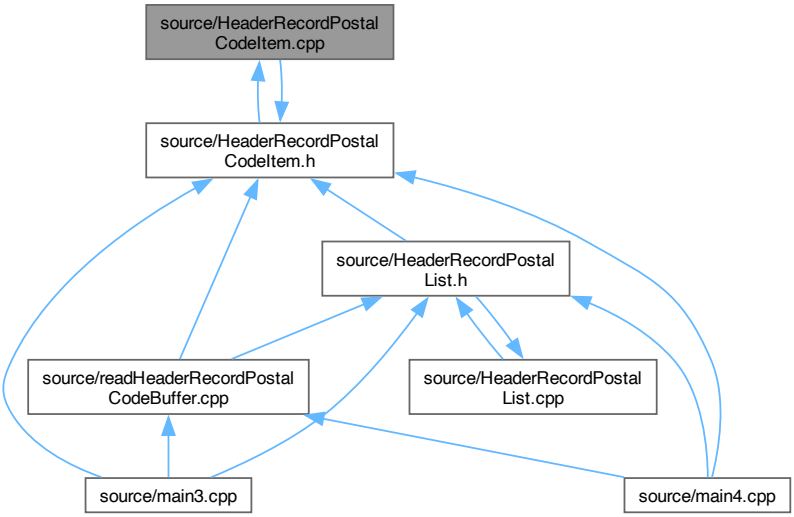
Defines and manages individual HeaderRecord postal code items.

```
#include "HeaderRecordPostalCodeItem.h"
#include <iostream>
#include <string>
#include <iomanip>
```

Include dependency graph for HeaderRecordPostalCodeItem.cpp:



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



5.1.1 Detailed Description

Defines and manages individual HeaderRecord postal code items.

Author

Mahad Farah, Kariniemi Carson, Tran Minh Quan, Rogers Mitchell, Asfaw Abel

Date

2025-10-17

Definition in file HeaderRecordPostalCodeItem.cpp.

5.2 HeaderRecordPostalCodeItem.cpp

Go to the documentation of this file.

```
00001
00007
00008 #include "HeaderRecordPostalCodeItem.h"
00009 #include <iostream>
00010 #include <string>
00011 #include <iomanip>
00012
00013 using namespace std;
00014
00031 HeaderRecordPostalCodeItem::HeaderRecordPostalCodeItem()
00032 {
00033     recordLength = 0;
00034     zip = 0;
00035     place = "";
00036     state = "";
00037     county = "";
```

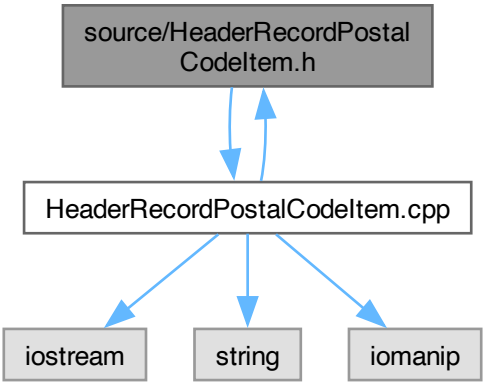
```
00038     latitude = 0;
00039     longitude = 0;
00040 }
00041
00053 HeaderRecordPostalCodeItem::HeaderRecordPostalCodeItem(int r, int z, const string &p, const string &s,
      const string &c, double lat, double lon)
00054 {
00055     recordLength = r;
00056     zip = z;
00057     place = p;
00058     state = s;
00059     county = c;
00060     latitude = lat;
00061     longitude = lon;
00062 }
00063
00064 int HeaderRecordPostalCodeItem::getRecordLength() const
00065 {
00066     return recordLength;
00067 }
00068
00073 int HeaderRecordPostalCodeItem::getZip() const
00074 {
00075     return zip;
00076 }
00077
00082 string HeaderRecordPostalCodeItem::getPlace() const
00083 {
00084     return place;
00085 }
00086
00091 string HeaderRecordPostalCodeItem::getState() const
00092 {
00093     return state;
00094 }
00095
00102 string HeaderRecordPostalCodeItem::getCounty() const
00103 {
00104     return county;
00105 }
00106
00112 double HeaderRecordPostalCodeItem::getLatitude() const
00113 {
00114     return latitude;
00115 }
00116
00122 double HeaderRecordPostalCodeItem::getLongitude() const
00123 {
00124     return longitude;
00125 }
00126
00127 void HeaderRecordPostalCodeItem::setRecordLength(int newRecordLength)
00128 {
00129     recordLength = newRecordLength;
00130 }
00131
00137 void HeaderRecordPostalCodeItem::setZip(int newZip)
00138 {
00139     zip = newZip;
00140 }
00141
00147 void HeaderRecordPostalCodeItem::setPlace(const string &newPlace)
00148 {
00149     place = newPlace;
00150 }
00151
00157 void HeaderRecordPostalCodeItem::setState(const string &newState)
00158 {
00159     state = newState;
00160 }
00161
00167 void HeaderRecordPostalCodeItem::setCounty(const string &newCounty)
00168 {
00169     county = newCounty;
00170 }
00171
00178 void HeaderRecordPostalCodeItem::setLatitude(double newLat)
00179 {
00180     latitude = newLat;
00181 }
00182
00189 void HeaderRecordPostalCodeItem::setLongitude(double newLon)
00190 {
00191     longitude = newLon;
00192 }
00193
00200 void HeaderRecordPostalCodeItem::printInfo() const
```

```
00201 {
00202     cout « left « setw(5) « recordLength
00203          « setw(10) « zip
00204          « setw(20) « place
00205          « setw(10) « state
00206          « setw(30) « county
00207          « setw(12) « latitude
00208          « setw(12) « longitude
00209          « endl;
00210 }
```

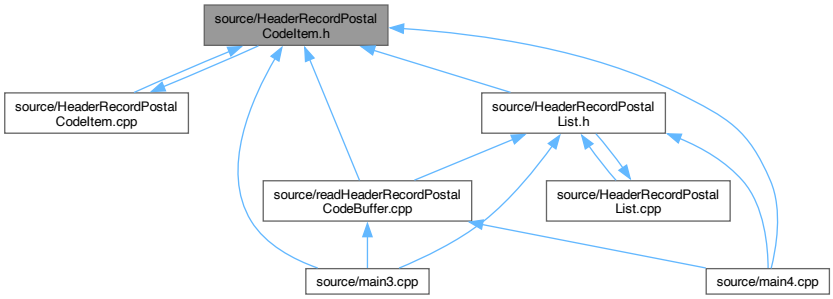
5.3 source/HeaderRecordPostalCodeItem.h File Reference

Defines and manages individual HeaderRecord postal code items.

`#include "HeaderRecordPostalCodeItem.cpp"`
Include dependency graph for HeaderRecordPostalCodeItem.h:



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



Classes

- class HeaderRecordPostalCodeItem

5.3.1 Detailed Description

Defines and manages individual HeaderRecord postal code items.

Author

Mahad Farah, Kariniemi Carson, Tran Minh Quan, Rogers Mitchell, Asfaw Abel

Date

2025-10-17

Definition in file HeaderRecordPostalCodeItem.h.

5.4 HeaderRecordPostalCodeItem.h

Go to the documentation of this file.

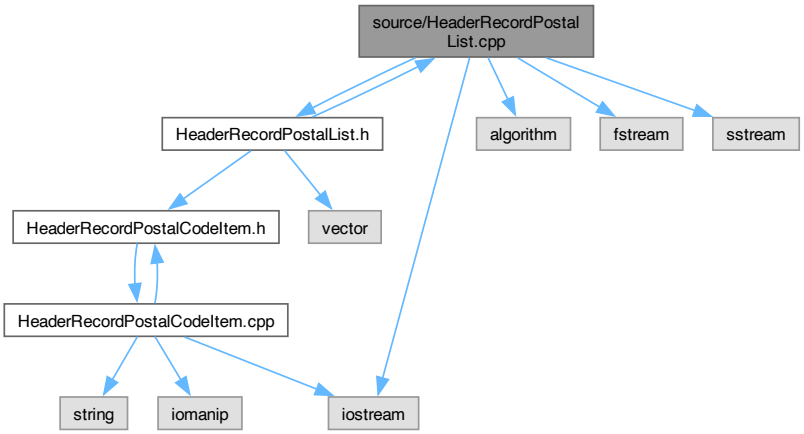
```
00001
00007
00008 #ifndef POSTAL_CODE_ITEM
00009 #define POSTAL_CODE_ITEM
00010
00011 using namespace std;
00012
00013 class HeaderRecordPostalCodeItem
00014 {
00015 private:
00016     int recordLength;
00017     int zip;
00018     string place;
00019     string state;
00020     string county;
00021     double latitude;
00022     double longitude;
00023
00024 public:
00041     HeaderRecordPostalCodeItem();
00042
00054     HeaderRecordPostalCodeItem(int r, int z, const string &p, const string &s, const string &c, double
      lat, double lon);
00055
00056     int getRecordLength() const;
00057
00062     int getZip() const;
00063
00068     string getPlace() const;
00069
00074     string getState() const;
00075
00082     string getCounty() const;
00083
00089     double getLatitude() const;
00090
00096     double getLongitude() const;
00097
00098     void setRecordLength(int newRecordLength);
00099
00105     void setZip(int newZip);
00106
00112     void setPlace(const string &newPlace);
00113
00119     void setState(const string &newState);
00120
00126     void setCounty(const string &newCounty);
00127
00134     void setLatitude(double newLat);
00135
00142     void setLongitude(double newLon);
00143
00150     void printInfo() const;
00151 };
00152
00153 #include "HeaderRecordPostalCodeItem.cpp"
00154 #endif
```

5.5 source/HeaderRecordPostalList.cpp File Reference

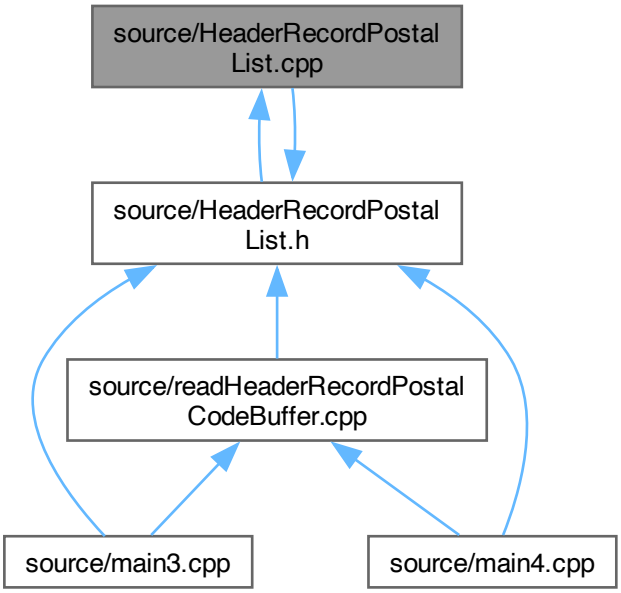
Defines and manages HeaderRecord postal code list structures.

```
#include "HeaderRecordPostalList.h"
#include <iostream>
#include <algorithm>
#include <fstream>
#include <sstream>
```

Include dependency graph for HeaderRecordPostalList.cpp:



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



5.5.1 Detailed Description

Defines and manages HeaderRecord postal code list structures.

Author

Mahad Farah, Kariniemi Carson, Tran Minh Quan, Rogers Mitchell, Asfaw Abel

Date

2025-10-17

Definition in file HeaderRecordPostalList.cpp.

5.6 HeaderRecordPostalList.cpp

Go to the documentation of this file.

```
00001
00007
00008 #include "HeaderRecordPostalList.h"
00009 #include <iostream>
00010 #include <algorithm>
00011 #include <fstream>
00012 #include <sstream>
00013
00014 using namespace std;
00015
00020 void HeaderRecordPostalList::addItem(const HeaderRecordPostalCodeItem &item)
00021 {
00022     items.push_back(item);
00023 }
00024
00031 HeaderRecordPostalCodeItem HeaderRecordPostalList::getItem(int index) const
00032 {
00033     if (index < items.size())
00034     {
00035         return items[index];
00036     }
00037     throw out_of_range("Index out of range in HeaderRecordPostalList::getItem");
00038 }
00039
00046 const HeaderRecordPostalCodeItem *HeaderRecordPostalList::findByZip(int zip) const
00047 {
00048     for (const auto &item : items)
00049     {
00050         if (item.getZip() == zip)
00051         {
00052             return &item;
00053         }
00054     }
00055     return nullptr;
00056 }
00057
00062 int HeaderRecordPostalList::size() const
00063 {
00064     return items.size();
00065 }
00066
00072 void HeaderRecordPostalList::printAll() const
00073 {
00074     for (int i = 0; i < items.size(); i++)
00075     {
00076         items[i].printInfo();
00077         cout «
      "-------------------------------------------------------------------------------------------" «
      endl;
00078     }
00079 }
00080
00086 void HeaderRecordPostalList::printSortedByZip() const
00087 {
00088     // Make a copy so original order is preserved
```

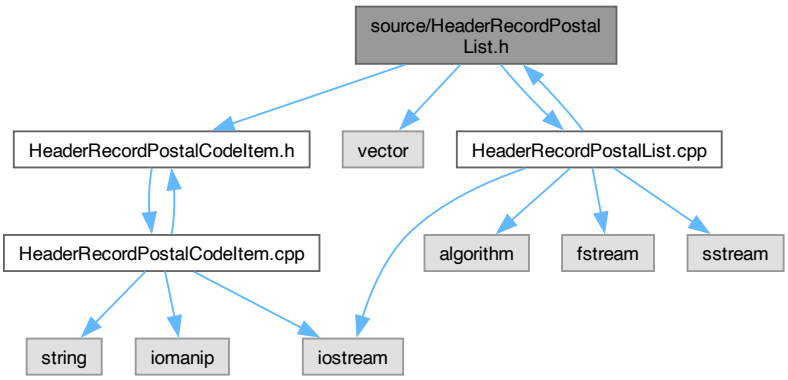
```
00089     vector<HeaderRecordPostalCodeItem> sortedItems = items;
00090
00091     sort(sortedItems.begin(), sortedItems.end(),
00092          [](const HeaderRecordPostalCodeItem &a, const HeaderRecordPostalCodeItem &b)
00093          {
00094              return a.getZip() < b.getZip();
00095          });
00096
00097     for (const auto &item : sortedItems)
00098     {
00099         item.printInfo();
00100         cout «
      "-------------------------------------------------------------------------------------------" «
      endl;
00101     }
00102 }
00103
00109 void HeaderRecordPostalList::printSortedByState() const
00110 {
00111     // Copy items so we don't change the internal order
00112     vector<HeaderRecordPostalCodeItem> sortedItems = items;
00113
00114     sort(sortedItems.begin(), sortedItems.end(),
00115          [](const HeaderRecordPostalCodeItem &a, const HeaderRecordPostalCodeItem &b)
00116          {
00117              if (a.getState() == b.getState())
00118              {
00119                  return a.getZip() < b.getZip(); // secondary sort by ZIP
00120              }
00121              return a.getState() < b.getState();
00122          });
00123
00124     for (const auto &item : sortedItems)
00125     {
00126         item.printInfo();
00127         cout «
      "-------------------------------------------------------------------------------------------" «
      endl;
00128     }
00129 }
```

5.7 source/HeaderRecordPostalList.h File Reference

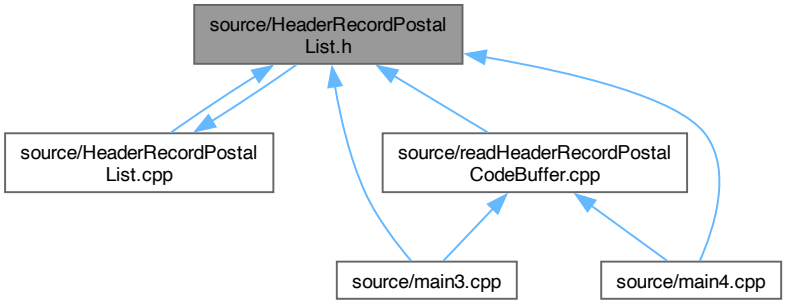
Defines and manages HeaderRecord postal code list structures.

```
#include "HeaderRecordPostalCodeItem.h"
#include <vector>
#include "HeaderRecordPostalList.cpp"
```

Include dependency graph for HeaderRecordPostalList.h:



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



Classes

- class HeaderRecordPostalList

5.7.1 Detailed Description

Defines and manages HeaderRecord postal code list structures.

Author

Mahad Farah

Date

2025-10-17

Definition in file HeaderRecordPostalList.h.

5.8 HeaderRecordPostalList.h

Go to the documentation of this file.

```
00001
00007
00008 #ifndef POSTAL_LIST_H
00009 #define POSTAL_LIST_H
00010
00011 #include "HeaderRecordPostalCodeItem.h"
00012 #include <vector>
00013
00014 using namespace std;
00015
00016 class HeaderRecordPostalList
00017 {
00018 private:
00019     vector<HeaderRecordPostalCodeItem> items;
00020
00021 public:
00022     // Constructors
00023     HeaderRecordPostalList() = default;
```

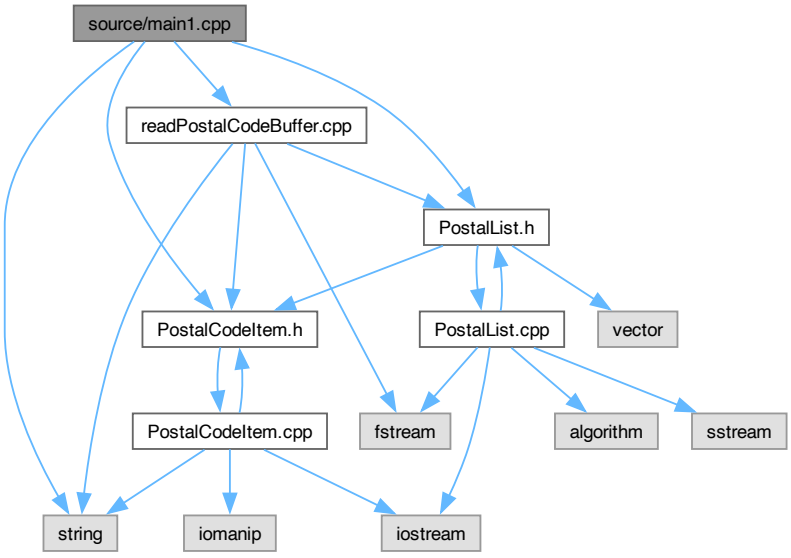
```
00024
00029     void addItem(const HeaderRecordPostalCodeItem &item);
00030
00037     HeaderRecordPostalCodeItem getItem(int index) const;
00038
00045     const HeaderRecordPostalCodeItem *findByZip(int zip) const;
00046
00051     int size() const;
00052
00058     void printAll() const;
00059
00065     void printSortedByZip() const;
00066
00072     void printSortedByState() const;
00073 };
00074
00075 #include "HeaderRecordPostalList.cpp"
00076 #endif
```

5.9 source/main1.cpp File Reference

Our first main file for "Zip Code Group 4 Project 2.0" that output the postal sorted by zip code.

```
#include <string>
#include "PostalCodeItem.h"
#include "PostalList.h"
#include "readPostalCodeBuffer.cpp"
```

Include dependency graph for main1.cpp:



Functions

- int main ()

  Starts the program, loads the CSV, and prints the table.

5.9.1 Detailed Description

Our first main file for "Zip Code Group 4 Project 2.0" that output the postal sorted by zip code.

@course CSCI 331 - Software Systems — Fall 2025 @project Zip Code Group Project 1.0

We read a CSV (made from the XLSX), load each row into our list, and then print the table the assignment asks for: one line per state (A–Z) showing, in this order, the ZIPs that are farthest East (smallest longitude), West (biggest longitude), North (biggest latitude), and South (smallest latitude). We also print a header first.

Authors

- Tran, Minh Quan
- Asfaw, Abel
- Kariniemi, Carson
- Rogers, Mitchell
- Farah, Mahad

Date

Sep 23rd 2025

Version

1.0

Bug None that we know of right now.

Definition in file main1.cpp.

5.9.2 Function Documentation

5.9.2.1 main()

```
int main ()
```

Starts the program, loads the CSV, and prints the table.

This main1.cpp will print the postal sorted by zip code.

Returns

0 if everything went fine.

Precondition

The file `us_postal_codes.csv` is in the same folder and has the expected columns.

Postcondition

We write the header and then one row per state to standard output.

Definition at line 44 of file main1.cpp.

References inputCSVtoList(), and PostalList::printSortedByZip().

5.10 main1.cpp

Go to the documentation of this file.

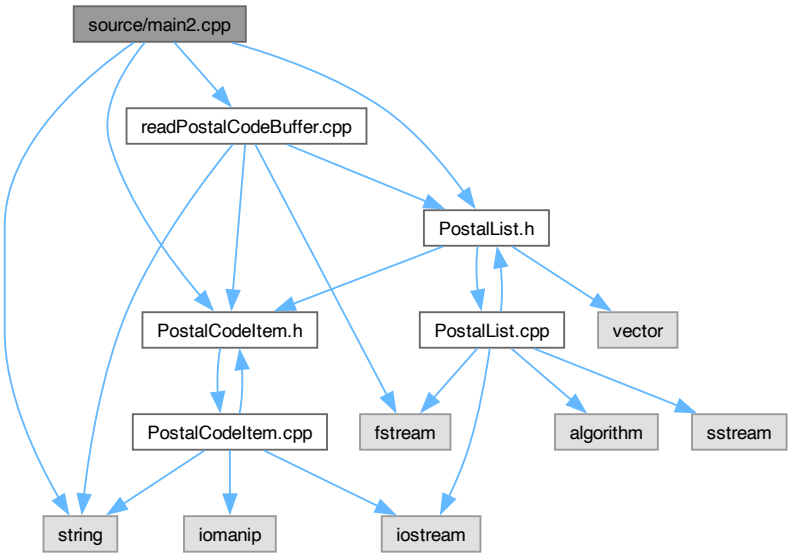
```
00001
00026
00027 #include <string>
00028 #include "PostalCodeItem.h"
00029 #include "PostalList.h"
00030 #include "readPostalCodeBuffer.cpp"
00031
00032 using namespace std;
00033
00044 int main()
00045 {
00046      // Create a variable for csv file name
00047      string fileName = "us_postal_codes.csv";
00048
00049      // Create an instance for PostalList
00050      PostalList myPostalList;
00051
00052      // Input the data from the CSV file to the postal list
00053      inputCSVtoList(myPostalList, fileName);
00054
00055      // Display the header with the appropriate table
00056      cout « "A table of all the postal sorted by zip:" « endl
00057           « endl;
00058      cout « left « setw(10) « "Zip Code"
00059           « setw(20) « "Place Name"
00060           « setw(10) « "State"
00061           « setw(30) « "County"
00062           « setw(12) « "Latitude"
00063           « setw(12) « "Longitude"
00064           « endl;
00065      cout «
      "-------------------------------------------------------------------------------------------" «
      endl;
00066
00067      // Display the table sorted by zip
00068      myPostalList.printSortedByZip();
00069
00070      return 0;
00071 }
```

5.11 source/main2.cpp File Reference

Our second main file for "Zip Code Group 4 Project 2.0" that output the postal sorted by state.

```
#include <string>
#include "PostalCodeItem.h"
#include "PostalList.h"
#include "readPostalCodeBuffer.cpp"
```

Include dependency graph for main2.cpp:



Functions

- int main ()

  Starts the program, loads the CSV, and prints the table.

5.11.1 Detailed Description

Our second main file for "Zip Code Group 4 Project 2.0" that output the postal sorted by state.

@course CSCI 331 - Software Systems — Fall 2025 @project Zip Code Group Project 1.0

We read a row randomized CSV (made from the XLSX), load each row into our list, and then print the table the assignment asks for: one line per state (A–Z) showing, in this order, the ZIPs that are farthest East (smallest longitude), West (biggest longitude), North (biggest latitude), and South (smallest latitude). We also print a header first.

Authors

- Tran, Minh Quan
- Asfaw, Abel
- Kariniemi, Carson
- Rogers, Mitchell
- Farah, Mahad

Date

Sep 23rd 2025

Version

1.0

Bug None that we know of right now.

Definition in file main2.cpp.

5.11.2 Function Documentation

5.11.2.1 main()

```
int main ()
```

Starts the program, loads the CSV, and prints the table.

This main2.cpp will print the postal sorted by zip code.

Returns

0 if everything went fine.

Precondition

The file `us_postal_codes.csv` is in the same folder and has the expected columns.

Postcondition

We write the header and then one row per state to standard output.

Definition at line 44 of file main2.cpp.

References inputCSVtoList(), and PostalList::printSortedByZip().

5.12 main2.cpp

Go to the documentation of this file.

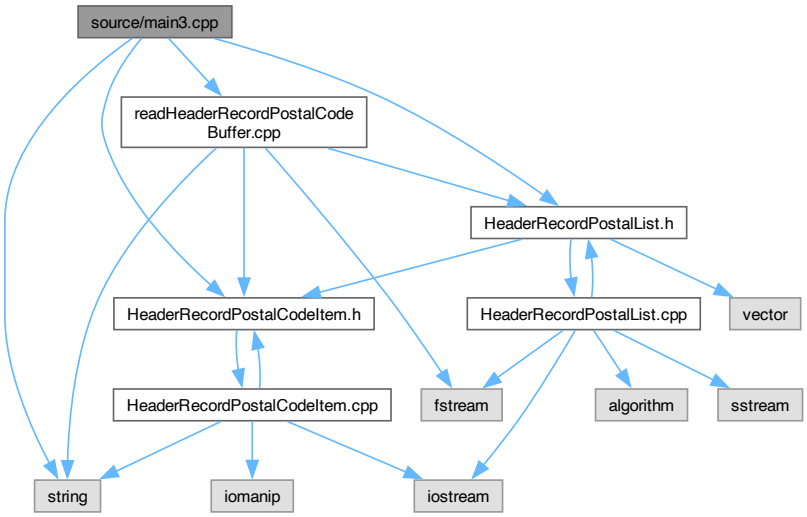
```
00001
00026
00027 #include <string>
00028 #include "PostalCodeItem.h"
00029 #include "PostalList.h"
00030 #include "readPostalCodeBuffer.cpp"
00031
00032 using namespace std;
00033
00044 int main()
00045 {
00046      // Create a variable for csv file name
00047      string fileName = "us_postal_codes_ROWS_RANDOMIZED.csv";
00048
00049      // Create an instance for PostalList
00050      PostalList myPostalList;
00051
00052      // Input the data from the CSV file to the postal list
00053      inputCSVtoList(myPostalList, fileName);
00054
00055      // Display the header with the appropriate table
00056      cout « "A table of all the postal sorted by zip from row randomized csv:" « endl
00057           « endl;
00058      cout « left « setw(10) « "Zip Code"
00059           « setw(20) « "Place Name"
00060           « setw(10) « "State"
00061           « setw(30) « "County"
00062           « setw(12) « "Latitude"
00063           « setw(12) « "Longitude"
00064           « endl;
00065      cout «
     "-------------------------------------------------------------------------------------------" «
     endl;
00066
00067      // Display the table sorted by zip
00068      myPostalList.printSortedByZip();
00069
00070      return 0;
00071 }
```

5.13 source/main3.cpp File Reference

Our third main file for "Zip Code Group 4 Project 2.0" that output the postal sorted by zip code.

```
#include <string>
#include "HeaderRecordPostalCodeItem.h"
#include "HeaderRecordPostalList.h"
#include "readHeaderRecordPostalCodeBuffer.cpp"
```

Include dependency graph for main3.cpp:



Functions

- int main ()

5.13.1 Detailed Description

Our third main file for "Zip Code Group 4 Project 2.0" that output the postal sorted by zip code.

@course CSCI 331 - Software Systems — Fall 2025 @project Zip Code Group Project 1.0

We read data from a file structure format with length indicated header record, load each row into our list, and then print the table the assignment asks for: one line per state (A–Z) showing, in this order, the header record with length indicated, the ZIPs that are farthest East (smallest longitude), West (biggest longitude), North (biggest latitude), and South (smallest latitude).

Authors

- Tran, Minh Quan
- Asfaw, Abel
- Kariniemi, Carson
- Rogers, Mitchell
- Farah, Mahad

Date

Oct 16th 2025

Version

1.0

Bug None that we know of right now.

Definition in file main3.cpp.

5.13.2 Function Documentation

5.13.2.1 main()

```
int main ()
```

Definition at line 35 of file main3.cpp.

References inputCSVtoList(), and HeaderRecordPostalList::printSortedByZip().

5.14 main3.cpp

Go to the documentation of this file.

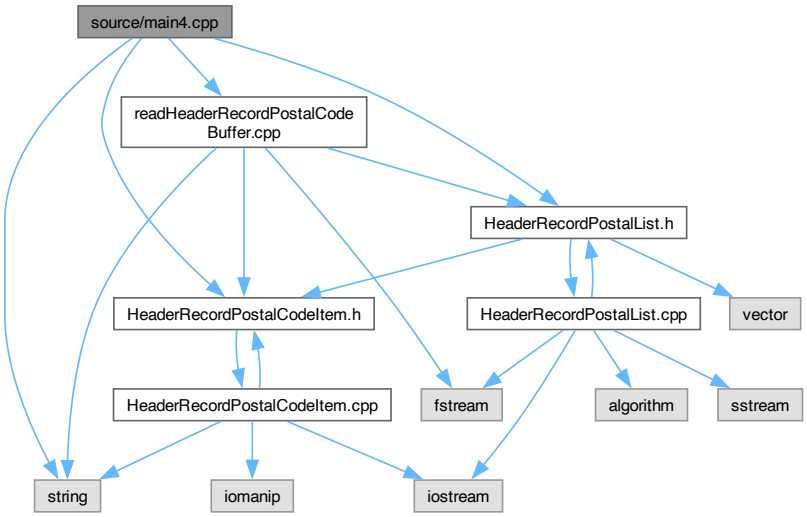
```
00001
00027
00028 #include <string>
00029 #include "HeaderRecordPostalCodeItem.h"
00030 #include "HeaderRecordPostalList.h"
00031 #include "readHeaderRecordPostalCodeBuffer.cpp"
00032
00033 using namespace std;
00034
00035 int main()
00036 {
00037      // Create a variable for csv file name
00038      string fileName = "us_postal_codes_length_indicated_header_record.txt";
00039
00040      // Create an instance for HeaderRecordPostalList
00041      HeaderRecordPostalList myHeaderRecordPostalList;
00042
00043      // Input the data from the CSV file to the postal list
00044      inputCSVtoList(myHeaderRecordPostalList, fileName);
00045
00046      // Display the header with the appropriate table
00047      cout « "A table of all the postal sorted by zip from length indicated records:" « endl
00048           « endl;
00049      cout « left « setw(5) « "HR"
00050           « setw(10) « "Zip Code"
00051           « setw(20) « "Place Name"
00052           « setw(10) « "State"
00053           « setw(30) « "County"
00054           « setw(12) « "Latitude"
00055           « setw(12) « "Longitude"
00056           « endl;
00057      cout «
     "-------------------------------------------------------------------------------------------" «
     endl;
00058
00059      // Display the table sorted by zip
00060      myHeaderRecordPostalList.printSortedByZip();
00061
00062      return 0;
00063 }
```

5.15 source/main4.cpp File Reference

Our third main file for "Zip Code Group 4 Project 2.0" that output the postal sorted by zip code.

```
#include <string>
#include "HeaderRecordPostalCodeItem.h"
#include "HeaderRecordPostalList.h"
#include "readHeaderRecordPostalCodeBuffer.cpp"
```

Include dependency graph for main4.cpp:



Functions

- int main ()

5.15.1 Detailed Description

Our third main file for "Zip Code Group 4 Project 2.0" that output the postal sorted by zip code.

@course CSCI 331 - Software Systems — Fall 2025 @project Zip Code Group Project 1.0

We read data from a row randomized file structure format with length indicated header record, load each row into our list, and then print the table the assignment asks for: one line per state (A–Z) showing, in this order, the header record with length indicated, the ZIPs that are farthest East (smallest longitude), West (biggest longitude), North (biggest latitude), and South (smallest latitude).

Authors

- Tran, Minh Quan
- Asfaw, Abel
- Kariniemi, Carson
- Rogers, Mitchell
- Farah, Mahad

Date

Oct 16th 2025

Version

1.0

Bug None that we know of right now.

Definition in file main4.cpp.

5.15.2 Function Documentation

5.15.2.1 main()

```
int main ()
```

Definition at line 35 of file main4.cpp.

References inputCSVtoList(), and HeaderRecordPostalList::printSortedByZip().

5.16 main4.cpp

Go to the documentation of this file.

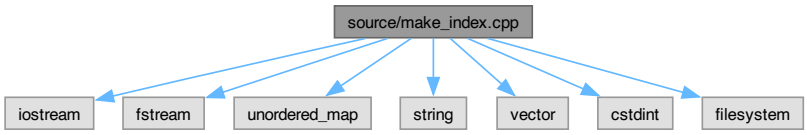
```
00001
00027
00028 #include <string>
00029 #include "HeaderRecordPostalCodeItem.h"
00030 #include "HeaderRecordPostalList.h"
00031 #include "readHeaderRecordPostalCodeBuffer.cpp"
00032
00033 using namespace std;
00034
00035 int main()
00036 {
00037      // Create a variable for csv file name
00038      string fileName = "us_postal_codes_ROWS_RANDOMIZED_length_indicated_header_record.txt";
00039
00040      // Create an instance for HeaderRecordPostalList
00041      HeaderRecordPostalList myHeaderRecordPostalList;
00042
00043      // Input the data from the CSV file to the postal list
00044      inputCSVtoList(myHeaderRecordPostalList, fileName);
00045
00046      // Display the header with the appropriate table
00047      cout « "A table of all the postal sorted by zip from row randomized length indicated records:" «
     endl
00048           « endl;
00049      cout « left « setw(5) « "HR"
00050           « setw(10) « "Zip Code"
00051           « setw(20) « "Place Name"
00052           « setw(10) « "State"
00053           « setw(30) « "County"
00054           « setw(12) « "Latitude"
00055           « setw(12) « "Longitude"
00056           « endl;
00057      cout «
     "-------------------------------------------------------------------------------------------" «
     endl;
00058
00059      // Display the table sorted by zip
00060      myHeaderRecordPostalList.printSortedByZip();
00061
00062      return 0;
00063 }
```

5.17 source/make_index.cpp File Reference

Builds or loads an index for fast lookup of postal code records from a CSV file.

```
#include <iostream>
#include <fstream>
#include <unordered_map>
#include <string>
#include <vector>
#include <cstdint>
#include <filesystem>
```

Include dependency graph for make_index.cpp:



Functions

- std::vector< std::string > split (const std::string &s, char delimiter)

  Splits a string by a given delimiter.
- void printRecord (const std::string &line)

  Prints a formatted postal code record from a CSV line.
- int main (int argc, char *argv[])

  Main function to build or load an index and retrieve postal code records.

Variables

- const std::string DATA_FILE = "us_postal_codes.csv"
- const std::string INDEX_FILE = "indexfile.bin"

5.17.1 Detailed Description

Builds or loads an index for fast lookup of postal code records from a CSV file.

Definition in file make_index.cpp.

5.17.2 Function Documentation

5.17.2.1 split()

```
std::vector< std::string > split (
            const std::string & s,
            char delimiter)
```

Splits a string by a given delimiter.

Parameters

s	The string to split.
delimiter	The character to split by.

Returns

A vector of substrings.

Definition at line 25 of file make_index.cpp.

5.17.2.2 printRecord()

```
void printRecord (
            const std::string & line)
```

Prints a formatted postal code record from a CSV line.

Parameters

line	A line from the CSV file representing a postal code record.

Definition at line 44 of file make_index.cpp.

References split().

5.17.2.3 main()

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

Main function to build or load an index and retrieve postal code records.

If the index file exists, it loads the index from disk. Otherwise, it builds a new index from the CSV file and writes it to disk. Then it retrieves and prints records for ZIP codes passed via command-line arguments using the `-Z` flag.

Parameters

argc	Argument count.
argv	Argument vector.

Returns

int Exit code.

Definition at line 68 of file make_index.cpp.

References DATA_FILE, INDEX_FILE, and printRecord().

5.17.3 Variable Documentation

5.17.3.1 DATA_FILE

```
const std::string DATA_FILE = "us_postal_codes.csv"
```

Definition at line 14 of file make_index.cpp.

5.17.3.2 INDEX_FILE

```
const std::string INDEX_FILE = "indexfile.bin"
```

Definition at line 15 of file make_index.cpp.

5.18 make_index.cpp

Go to the documentation of this file.

```
00001
00005
00006 #include <iostream>
00007 #include <fstream>
00008 #include <unordered_map>
00009 #include <string>
00010 #include <vector>
00011 #include <cstdint>
00012 #include <filesystem>
00013
00014 const std::string DATA_FILE = "us_postal_codes.csv";
00015 const std::string INDEX_FILE = "indexfile.bin";
00016
00024
00025 std::vector<std::string> split(const std::string &s, char delimiter)
00026 {
00027     std::vector<std::string> result;
00028     size_t start = 0, end;
00029     while ((end = s.find(delimiter, start)) != std::string::npos)
00030     {
00031         result.push_back(s.substr(start, end - start));
00032         start = end + 1;
00033     }
00034     result.push_back(s.substr(start));
00035     return result;
00036 }
00037
00043
00044 void printRecord(const std::string &line)
00045 {
00046     auto fields = split(line, ',');
00047     const char *labels[] = {"Zip Code", "Place Name", "State", "County", "Lat", "Long"};
00048
00049     for (size_t i = 0; i < fields.size() && i < 6; ++i)
00050     {
00051         std::cout << labels[i] << ": " << fields[i] << "\n";
00052     }
00053     std::cout << "\n";
00054 }
00055
00067
00068 int main(int argc, char *argv[])
00069 {
00070     std::unordered_map<std::string, std::streampos> index;
00071
00072     if (std::filesystem::exists(INDEX_FILE))
00073     {
00074         std::ifstream idx(INDEX_FILE, std::ios::binary);
00075         if (!idx.is_open())
00076         {
00077             std::cerr << "Error: unable to open " << INDEX_FILE << "\n";
00078             return 1;
00079         }
```

```
00080
00081         while (true)
00082         {
00083             uint8_t len;
00084             if (!idx.read(reinterpret_cast<char *>(&len), sizeof(len)))
00085                 break;
00086             std::string zip(len, '\0');
00087             idx.read(zip.data(), len);
00088             uint64_t offset;
00089             idx.read(reinterpret_cast<char *>(&offset), sizeof(offset));
00090             index[zip] = static_cast<std::streampos>(offset);
00091         }
00092
00093         idx.close();
00094         std::cout « "Loaded existing index file: " « INDEX_FILE
00095                   « " (" « index.size() « " entries)\n";
00096     }
00097     else
00098     {
00099         std::cout « "Index not found, building new one...\n";
00100
00101         std::ifstream data(DATA_FILE, std::ios::binary);
00102         if (!data.is_open())
00103         {
00104             std::cerr « "Error: unable to open " « DATA_FILE « "\n";
00105             return 1;
00106         }
00107
00108         std::string header;
00109         std::getline(data, header);
00110         std::string line;
00111         while (true)
00112         {
00113             std::streampos pos = data.tellg();
00114             if (!std::getline(data, line))
00115                 break;
00116             if (line.empty())
00117                 continue;
00118             if (!line.empty() && line.back() == '\r')
00119                 line.pop_back();
00120
00121             std::string zip = line.substr(0, line.find(','));
00122             index[zip] = pos;
00123         }
00124         data.close();
00125
00126         std::ofstream idx(INDEX_FILE, std::ios::binary);
00127         for (const auto &[zip, offset] : index)
00128         {
00129             uint8_t len = static_cast<uint8_t>(zip.size());
00130             uint64_t off = static_cast<uint64_t>(offset);
00131             idx.write(reinterpret_cast<const char *>(&len), sizeof(len));
00132             idx.write(zip.data(), len);
00133             idx.write(reinterpret_cast<const char *>(&off), sizeof(off));
00134         }
00135         idx.close();
00136
00137         std::cout « "Index built and written to " « INDEX_FILE
00138                   « " (" « index.size() « " entries)\n";
00139     }
00140
00141     std::vector<std::string> zips;
00142     for (int i = 1; i < argc; ++i)
00143     {
00144         std::string arg = argv[i];
00145         if (arg.rfind("-Z", 0) == 0)
00146             zips.push_back(arg.substr(2));
00147     }
00148
00149     std::ifstream data(DATA_FILE, std::ios::binary);
00150     if (!data.is_open())
00151     {
00152         std::cerr « "Error: unable to open " « DATA_FILE « "\n";
00153         return 1;
00154     }
00155
00156     for (const auto &zip : zips)
00157     {
00158         if (index.find(zip) == index.end())
00159         {
00160             std::cout « zip « " not found.\n\n";
00161             continue;
00162         }
00163
00164         data.clear();
00165         data.seekg(index[zip]);
00166         std::string line;
```

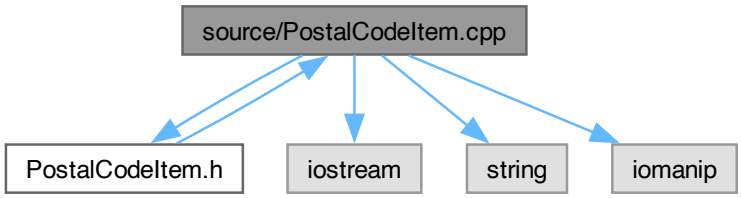
```
00167         std::getline(data, line);
00168         if (!line.empty() && line.back() == '\r')
00169             line.pop_back();
00170 
00171         printRecord(line);
00172     }
00173 
00174     return 0;
00175 }
```

5.19 source/PostalCodeItem.cpp File Reference

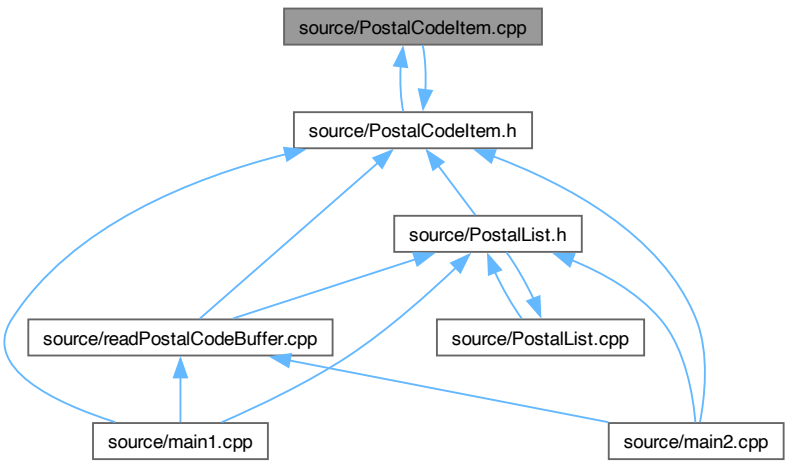
Implementation of the PostalCodeItem class representing a postal code entry.

```
#include "PostalCodeItem.h"
#include <iostream>
#include <string>
#include <iomanip>
```

Include dependency graph for PostalCodeItem.cpp:



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



5.19.1 Detailed Description

Implementation of the PostalCodeItem class representing a postal code entry.

Author

- Asfaw, Abel, Farah, Mahad, Kariniemi, Carson, Rogers, Mitchell Tran, Minh Quan

Version

1.0

Date

2025-9-23

Definition in file PostalCodeItem.cpp.

5.20 PostalCodeItem.cpp

Go to the documentation of this file.

```
00001
00013
00022
00023 #include "PostalCodeItem.h"
00024 #include <iostream>
00025 #include <string>
00026 #include <iomanip>
00027
00028 using namespace std;
00029
00046 PostalCodeItem::PostalCodeItem()
00047 {
00048     zip = 0;
00049     place = "";
00050     state = "";
00051     county = "";
00052     latitude = 0;
00053     longitude = 0;
00054 }
00055
00067 PostalCodeItem::PostalCodeItem(int z, const string &p, const string &s, const string &c, double lat,
      double lon)
00068 {
00069     zip = z;
00070     place = p;
00071     state = s;
00072     county = c;
00073     latitude = lat;
00074     longitude = lon;
00075 }
00076
00081 int PostalCodeItem::getZip() const
00082 {
00083     return zip;
00084 }
00085
00090 string PostalCodeItem::getPlace() const
00091 {
00092     return place;
00093 }
00094
00099 string PostalCodeItem::getState() const
00100 {
00101     return state;
00102 }
00103
00110 string PostalCodeItem::getCounty() const
00111 {
00112     return county;
00113 }
00114
00120 double PostalCodeItem::getLatitude() const
```

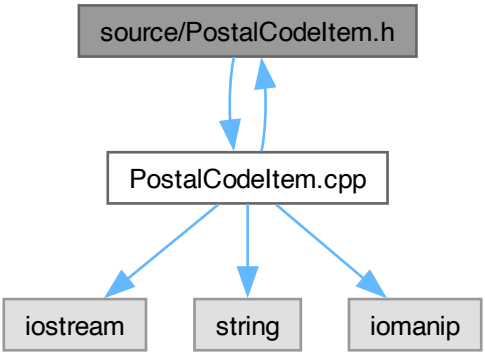
```
00121 {
00122     return latitude;
00123 }
00124
00130 double PostalCodeItem::getLongitude() const
00131 {
00132     return longitude;
00133 }
00134
00140 void PostalCodeItem::setZip(int newZip)
00141 {
00142     zip = newZip;
00143 }
00144
00150 void PostalCodeItem::setPlace(const string &newPlace)
00151 {
00152     place = newPlace;
00153 }
00154
00160 void PostalCodeItem::setState(const string &newState)
00161 {
00162     state = newState;
00163 }
00164
00170 void PostalCodeItem::setCounty(const string &newCounty)
00171 {
00172     county = newCounty;
00173 }
00174
00181 void PostalCodeItem::setLatitude(double newLat)
00182 {
00183     latitude = newLat;
00184 }
00185
00192 void PostalCodeItem::setLongitude(double newLon)
00193 {
00194     longitude = newLon;
00195 }
00196
00203 void PostalCodeItem::printInfo() const
00204 {
00205     cout « left « setw(10) « zip
00206          « setw(20) « place
00207          « setw(10) « state
00208          « setw(30) « county
00209          « setw(12) « latitude
00210          « setw(12) « longitude
00211          « endl;
00212 }
```

5.21 source/PostalCodeItem.h File Reference

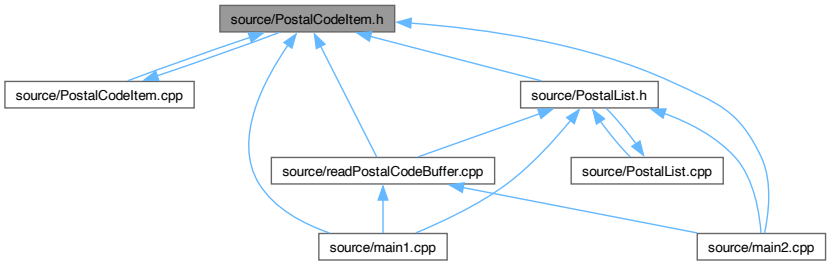
Defines the PostalCodeItem class for representing postal code records.

```
#include "PostalCodeItem.cpp"
```

Include dependency graph for PostalCodeItem.h:



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



Classes

- class PostalCodeItem

5.21.1 Detailed Description

Defines the PostalCodeItem class for representing postal code records.

Author

Asfaw, Abel, Farah, Mahad, Kariniemi, Carson, Rogers, Mitchell Tran, Minh Quan Each PostalCodeItem stores data about a single postal code including:

- ZIP code
- Place name
- State abbreviation
- County
- Latitude
- Longitude

Definition in file PostalCodeItem.h.

5.22 PostalCodeItem.h

Go to the documentation of this file.

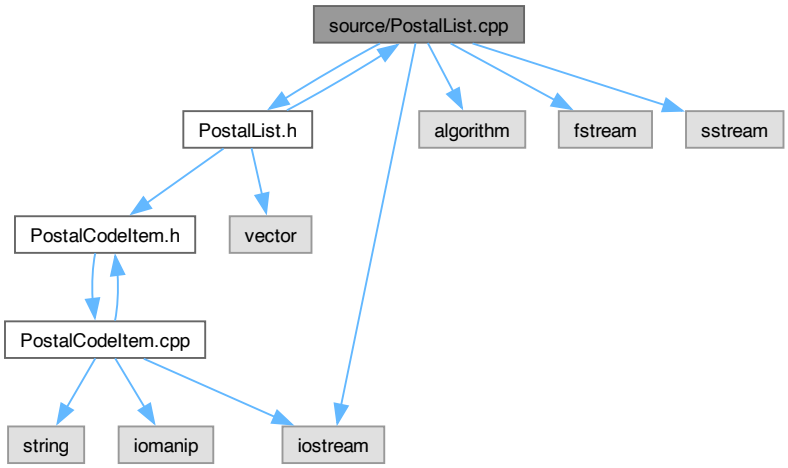
```
00001
00018
00019 #ifndef POSTAL_CODE_ITEM
00020 #define POSTAL_CODE_ITEM
00021
00022 using namespace std;
00023
00024 class PostalCodeItem
00025 {
00026 private:
00027     int zip;
00028     string place;
00029     string state;
00030     string county;
00031     double latitude;
00032     double longitude;
00033
00034 public:
00051     PostalCodeItem();
00052
00064     PostalCodeItem(int z, const string &p, const string &s, const string &c, double lat, double lon);
00065
00070     int getZip() const;
00071
00076     string getPlace() const;
00077
00082     string getState() const;
00083
00090     string getCounty() const;
00091
00097     double getLatitude() const;
00098
00104     double getLongitude() const;
00105
00111     void setZip(int newZip);
00112
00118     void setPlace(const string &newPlace);
00119
00125     void setState(const string &newState);
00126
00132     void setCounty(const string &newCounty);
00133
00140     void setLatitude(double newLat);
00141
00148     void setLongitude(double newLon);
00149
00156     void printInfo() const;
00157 };
00158
00159 #include "PostalCodeItem.cpp"
00160 #endif
```

5.23 source/PostalList.cpp File Reference

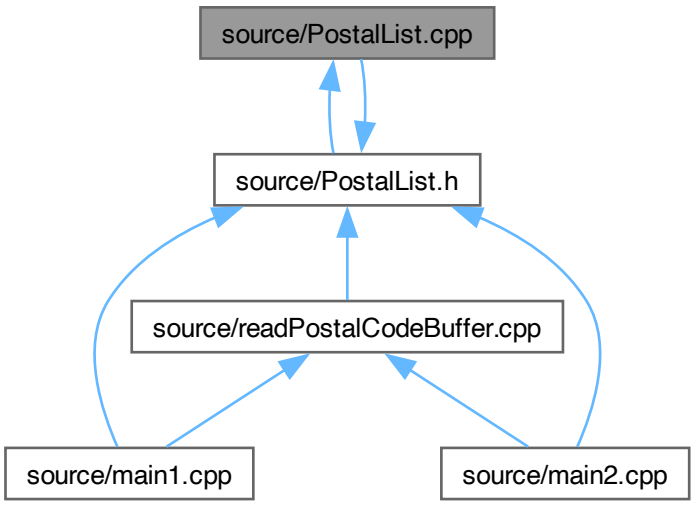
@ brief Implementation of the PostalList class for managing a collection of PostalCodeItem objects.

```
#include "PostalList.h"
#include <iostream>
#include <algorithm>
#include <fstream>
#include <sstream>
```

Include dependency graph for PostalList.cpp:



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



5.23.1 Detailed Description

@ brief Implementation of the PostalList class for managing a collection of PostalCodeItem objects.

@ author Asfaw, Abel, Farah, Mahad, Kariniemi, Carson, Rogers, Mitchell Tran, Minh Quan @ version 1.0 @ date 2025-9-23

Definition in file PostalList.cpp.

5.24 PostalList.cpp

Go to the documentation of this file.

```
00001
00013
00014 #include "PostalList.h"
00015 #include <iostream>
00016 #include <algorithm>
00017 #include <fstream>
00018 #include <sstream>
00019
00020 using namespace std;
00021
00026 void PostalList::addItem(const PostalCodeItem &item)
00027 {
00028     items.push_back(item);
00029 }
00030
00037 PostalCodeItem PostalList::getItem(int index) const
00038 {
00039     if (index < items.size())
00040     {
00041         return items[index];
00042     }
00043     throw out_of_range("Index out of range in PostalList::getItem");
00044 }
00045
00052 const PostalCodeItem *PostalList::findByZip(int zip) const
00053 {
00054     for (const auto &item : items)
00055     {
00056         if (item.getZip() == zip)
00057         {
00058             return &item;
00059         }
00060     }
00061     return nullptr;
00062 }
00063
00068 int PostalList::size() const
00069 {
00070     return items.size();
00071 }
00072
00078 void PostalList::printAll() const
00079 {
00080     for (int i = 0; i < items.size(); i++)
00081     {
00082         items[i].printInfo();
00083         cout «
      "-------------------------------------------------------------------------------------------------" «
      endl;
00084     }
00085 }
00086
00092 void PostalList::printSortedByZip() const
00093 {
00094     // Make a copy so original order is preserved
00095     vector<PostalCodeItem> sortedItems = items;
00096
00097     sort(sortedItems.begin(), sortedItems.end(),
00098          [](const PostalCodeItem &a, const PostalCodeItem &b)
00099          {
00100              return a.getZip() < b.getZip();
00101          });
00102
00103     for (const auto &item : sortedItems)
00104     {
00105         item.printInfo();
00106         cout «
      "-------------------------------------------------------------------------------------------------" «
      endl;
00107     }
00108 }
00109
00115 void PostalList::printSortedByState() const
00116 {
00117     // Copy items so we don't change the internal order
00118     vector<PostalCodeItem> sortedItems = items;
00119
00120     sort(sortedItems.begin(), sortedItems.end(),
00121          [](const PostalCodeItem &a, const PostalCodeItem &b)
00122          {
00123              if (a.getState() == b.getState())
00124              {
```

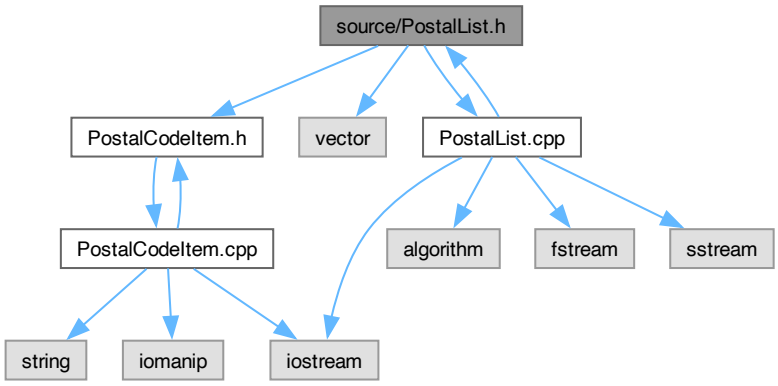
```
00125                 return a.getZip() < b.getZip(); // secondary sort by ZIP
00126             }
00127             return a.getState() < b.getState();
00128         });
00129
00130     for (const auto &item : sortedItems)
00131     {
00132         item.printInfo();
00133         cout «
      "-------------------------------------------------------------------------------------------" «
      endl;
00134     }
00135 }
```

5.25 source/PostalList.h File Reference

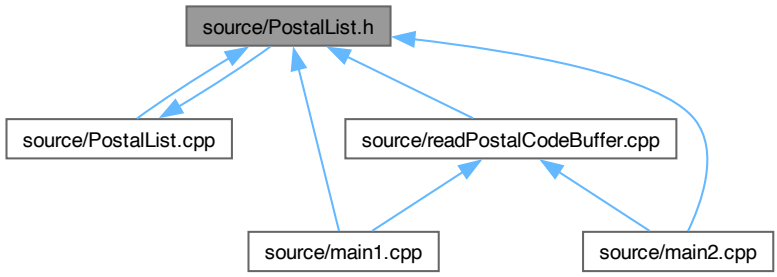
Defines the PostalList class for managing collections of postal codes.

```
#include "PostalCodeItem.h"
#include <vector>
#include "PostalList.cpp"
```

Include dependency graph for PostalList.h:



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



Classes

- class PostalList

5.25.1 Detailed Description

Defines the PostalList class for managing collections of postal codes.

Author

Asfaw, Abel, Farah, Mahad, Kariniemi, Carson, Rogers, Mitchell Tran, Minh Quan The PostalList class provides storage and utility functions for handling multiple PostalCodeItem objects, including adding, searching, and printing data in sorted order.

Definition in file PostalList.h.

5.26 PostalList.h

Go to the documentation of this file.

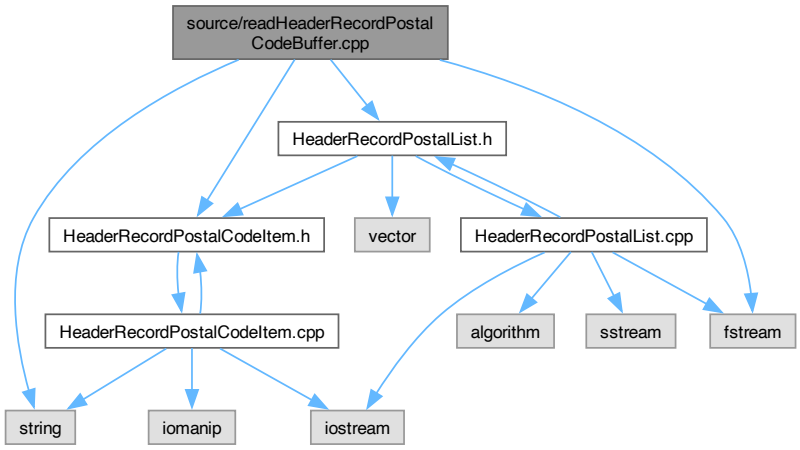
```
00001
00014
00015 #ifndef POSTAL_LIST_H
00016 #define POSTAL_LIST_H
00017
00018 #include "PostalCodeItem.h"
00019 #include <vector>
00020
00021 using namespace std;
00022
00023 class PostalList
00024 {
00025 private:
00026     vector<PostalCodeItem> items;
00027
00028 public:
00029     // Constructors
00030     PostalList() = default;
00031
00036     void addItem(const PostalCodeItem &item);
00037
00044     PostalCodeItem getItem(int index) const;
00045
00052     const PostalCodeItem *findByZip(int zip) const;
00053
00058     int size() const;
00059
00065     void printAll() const;
00066
00072     void printSortedByZip() const;
00073
00079     void printSortedByState() const;
00080 };
00081
00082 #include "PostalList.cpp"
00083 #endif
```

5.27 source/readHeaderRecordPostalCodeBuffer.cpp File Reference

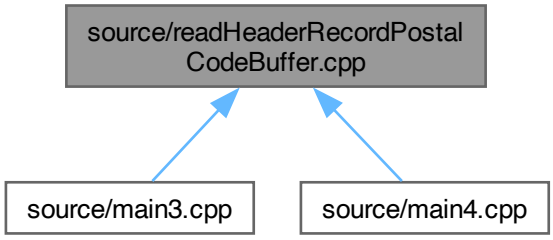
Reads a CSV file containing postal code data and populates a HeaderRecordPostalList.

```
#include <string>
#include "HeaderRecordPostalCodeItem.h"
#include "HeaderRecordPostalList.h"
#include <fstream>
```

Include dependency graph for readHeaderRecordPostalCodeBuffer.cpp:



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



Functions

- void inputCSVtoList (HeaderRecordPostalList &inputList, string fileName)

  Reads a CSV file and populates a HeaderRecordPostalList with parsed postal code records.

5.27.1 Detailed Description

Reads a CSV file containing postal code data and populates a HeaderRecordPostalList.

Definition in file readHeaderRecordPostalCodeBuffer.cpp.

5.27.2 Function Documentation

5.27.2.1 inputCSVtoList()

```
void inputCSVtoList (
            HeaderRecordPostalList & inputList,
            string fileName)
```

Reads a CSV file and populates a HeaderRecordPostalList with parsed postal code records.

The CSV file is expected to have the following header: "zip,place,state,county,latitude,longitude"

Each line is parsed into a HeaderRecordPostalCodeItem and added to the input list.

Parameters

inputList	Reference to the HeaderRecordPostalList to populate.
fileName	Name of the CSV file to read.

Definition at line 26 of file readHeaderRecordPostalCodeBuffer.cpp.

References HeaderRecordPostalList::addItem(), HeaderRecordPostalCodeItem::setCounty(), HeaderRecordPostalCodeItem::setLatit
HeaderRecordPostalCodeItem::setLongitude(), HeaderRecordPostalCodeItem::setPlace(), HeaderRecordPostalCodeItem::setRecord
HeaderRecordPostalCodeItem::setState(), and HeaderRecordPostalCodeItem::setZip().

5.28 readHeaderRecordPostalCodeBuffer.cpp

Go to the documentation of this file.

```
00001
00006
00007 #include <string>
00008 #include "HeaderRecordPostalCodeItem.h"
00009 #include "HeaderRecordPostalList.h"
00010 #include <fstream>
00011
00012 using namespace std;
00013
00025
00026 void inputCSVtoList(HeaderRecordPostalList &inputList, string fileName)
00027 {
00028     HeaderRecordPostalCodeItem item;
00029     string line = "";
00030     int location = 0;
00031
00032     ifstream myFile;
00033     myFile.open(fileName);
00034
00035     // Skip the header: "zip,place,state,county,latitude,longitude"
00036     getline(myFile, line);
00037
00038     while (getline(myFile, line))
00039     {
00040         // Record Length
00041         item.setRecordLength(stoi(line.substr(0, 2)));
```

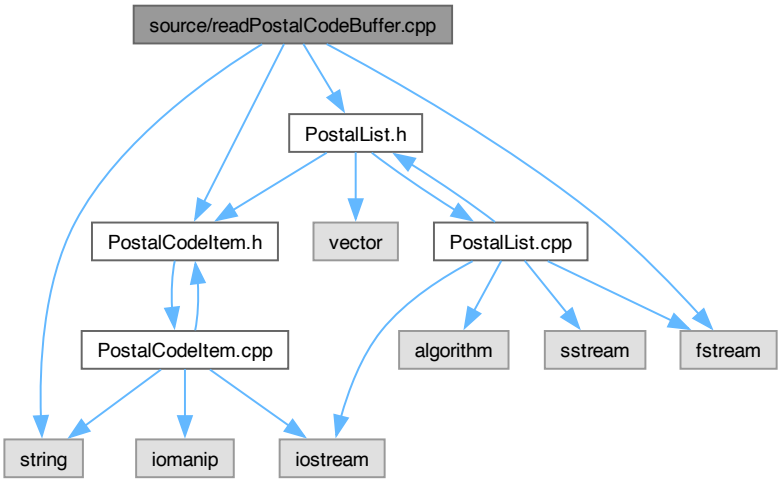
```
00042         line = line.substr(3, line.length());
00043         // ZIP
00044         location = line.find(",");
00045         item.setZip(stoi(line.substr(0, location)));
00046         line = line.substr(location + 1, line.length());
00047
00048         // Place
00049         location = line.find(",");
00050         item.setPlace(line.substr(0, location));
00051         line = line.substr(location + 1, line.length());
00052
00053         // State
00054         location = line.find(",");
00055         item.setState(line.substr(0, location));
00056         line = line.substr(location + 1, line.length());
00057
00058         // County
00059         location = line.find(",");
00060         item.setCounty(line.substr(0, location));
00061         line = line.substr(location + 1, line.length());
00062
00063         // Latitude
00064         location = line.find(",");
00065         item.setLatitude(stod(line.substr(0, location)));
00066         line = line.substr(location + 1, line.length());
00067
00068         // Longitude (last part of the line)
00069         item.setLongitude(stod(line));
00070
00071         // Add it to our list
00072         inputList.addItem(item);
00073     }
00074
00075     myFile.close();
00076 }
```

5.29 source/readPostalCodeBuffer.cpp File Reference

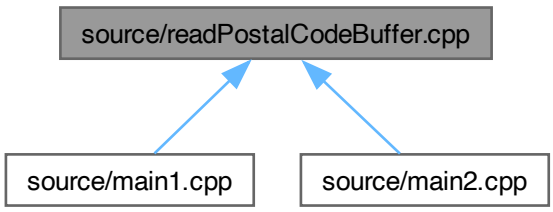
Utility functions for reading postal code data from a CSV file.

```
#include <string>
#include "PostalCodeItem.h"
#include "PostalList.h"
#include <fstream>
```

Include dependency graph for readPostalCodeBuffer.cpp:



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



Functions

- void inputCSVtoList (PostalList &inputList, string fileName)

  Reads the CSV and adds each row to the list.

5.29.1 Detailed Description

Utility functions for reading postal code data from a CSV file.

@course CSCI 331 - Software Systems — Fall 2025 @project Zip Code Group Project 1.0

This file defines functions that read U.S. postal code data stored in a CSV file and populate a PostalList object. Each line of the CSV contains one postal record.

Authors

- Tran, Minh Quan
- Asfaw, Abel
- Kariniemi, Carson
- Rogers, Mitchell
- Farah, Mahad

Date

Sep 23rd 2025

Version

1.0

Bug None that we know of right now.

Definition in file readPostalCodeBuffer.cpp.

5.29.2 Function Documentation

5.29.2.1 inputCSVtoList()

```
void inputCSVtoList (
            PostalList & inputList,
            string fileName)
```

Reads the CSV and adds each row to the list.

The file has a header, then each line has 6 pieces: ZIP, Place, State, County, Latitude, Longitude

Parameters

inputList	Where we store all the items.
fileName	The CSV file we open.

Precondition

- The CSV exists and we can open it.
- Lines are simple comma-separated (no quotes/commas inside fields).

Postcondition

- Every good line becomes a PostalCodeItem in `inputList`.
- The file is closed before we leave.

Note

We keep this simple on purpose. If the CSV has quotes or weird commas, this version won't handle it.

Definition at line 52 of file readPostalCodeBuffer.cpp.

References PostalList::addItem(), PostalCodeItem::setCounty(), PostalCodeItem::setLatitude(), PostalCodeItem::setLongitude(), PostalCodeItem::setPlace(), PostalCodeItem::setState(), and PostalCodeItem::setZip().

5.30 readPostalCodeBuffer.cpp

Go to the documentation of this file.

```
00001
00024
00025 #include <string>
00026 #include "PostalCodeItem.h"
00027 #include "PostalList.h"
00028 #include <fstream>
00029
00030 using namespace std;
00031
00051
00052 void inputCSVtoList(PostalList &inputList, string fileName)
00053 {
00054     PostalCodeItem item;
00055     string line = "";
00056     int location = 0;
00057
00058     ifstream myFile;
```

```
00059     myFile.open(fileName);
00060
00061     // Skip the header: "zip,place,state,county,latitude,longitude"
00062     getline(myFile, line);
00063
00064     while (getline(myFile, line))
00065     {
00066         // ZIP
00067         location = line.find(",");
00068         item.setZip(stoi(line.substr(0, location)));
00069         line = line.substr(location + 1, line.length());
00070
00071         // Place
00072         location = line.find(",");
00073         item.setPlace(line.substr(0, location));
00074         line = line.substr(location + 1, line.length());
00075
00076         // State
00077         location = line.find(",");
00078         item.setState(line.substr(0, location));
00079         line = line.substr(location + 1, line.length());
00080
00081         // County
00082         location = line.find(",");
00083         item.setCounty(line.substr(0, location));
00084         line = line.substr(location + 1, line.length());
00085
00086         // Latitude
00087         location = line.find(",");
00088         item.setLatitude(stod(line.substr(0, location)));
00089         line = line.substr(location + 1, line.length());
00090
00091         // Longitude (last part of the line)
00092         item.setLongitude(stod(line));
00093
00094         // Add it to our list
00095         inputList.addItem(item);
00096     }
00097
00098     myFile.close();
00099 }
```