

Project 1 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 PostalCodeItem Class Reference	7
4.1.1 Detailed Description	8
4.1.2 Constructor & Destructor Documentation	8
4.1.2.1 PostalCodeItem() [1/2]	8
4.1.2.2 PostalCodeItem() [2/2]	9
4.1.3 Member Function Documentation	10
4.1.3.1 getZip()	10
4.1.3.2 getPlace()	10
4.1.3.3 getState()	10
4.1.3.4 getCounty()	10
4.1.3.5 getLatitude()	11
4.1.3.6 getLongitude()	11
4.1.3.7 setZip()	11
4.1.3.8 setPlace()	11
4.1.3.9 setState()	12
4.1.3.10 setCounty()	12
4.1.3.11 setLatitude()	12
4.1.3.12 setLongitude()	13
4.1.3.13 printInfo()	13
4.2 PostalList Class Reference	14
4.2.1 Detailed Description	14
4.2.2 Constructor & Destructor Documentation	15
4.2.2.1 PostalList()	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()	15
4.2.3.4 size()	16
4.2.3.5 printAll()	16
4.2.3.6 printSortedByZip()	16
4.2.3.7 printSortedByState()	17
5 File Documentation	19
5.1 source/main1.cpp File Reference	19
5.1.1 Detailed Description	20

5.1.2 Function Documentation	20
5.1.2.1 main()	20
5.2 main1.cpp	21
5.3 source/main2.cpp File Reference	21
5.3.1 Detailed Description	22
5.3.2 Function Documentation	23
5.3.2.1 main()	23
5.4 main2.cpp	23
5.5 source/PostalCodeltem.cpp File Reference	24
5.5.1 Detailed Description	24
5.6 PostalCodeltem.cpp	25
5.7 source/PostalCodeltem.h File Reference	26
5.7.1 Detailed Description	27
5.8 PostalCodeltem.h	27
5.9 source/PostalList.cpp File Reference	28
5.9.1 Detailed Description	29
5.10 PostalList.cpp	29
5.11 source/PostalList.h File Reference	30
5.11.1 Detailed Description	31
5.12 PostalList.h	32
5.13 source/readCSV.cpp File Reference	32
5.13.1 Detailed Description	33
5.13.2 Function Documentation	34
5.13.2.1 inputCSVtoList()	34
5.14 readCSV.cpp	35

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 [readCSV.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:

PostalCodeItem	7
PostalList	14

Chapter 3

File Index

3.1 File List

Here is a list of all files with brief descriptions:

source/ main1.cpp	
Our first main file for "Zip Code Group 4 Project 1.0" that output the postal sorted by zip code	19
source/ main2.cpp	
Our second main file for "Zip Code Group 4 Project 1.0" that output the postal sorted by state	21
source/ PostalCodeItem.cpp	
Implementation of the PostalCodeItem class representing a postal code entry	24
source/ PostalCodeItem.h	
Defines the PostalCodeItem class for representing postal code records	26
source/ PostalList.cpp	
@ brief Implementation of the PostalList class for managing a collection of PostalCodeItem objects	28
source/ PostalList.h	
Defines the PostalList class for managing collections of postal codes	30
source/ readCSV.cpp	
Utility functions for reading postal code data from a CSV file	32

Chapter 4

Class Documentation

4.1 PostalCodelItem Class Reference

```
#include <PostalCodelItem.h>
```

Collaboration diagram for PostalCodelItem:

PostalCodelItem
+ PostalCodelItem() + PostalCodelItem() + getZip() + getPlace() + getState() + getCounty() + getLatitude() + getLongitude() + setZip() + setPlace() + setState() + setCounty() + setLatitude() + setLongitude() + printInfo()

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.

4.1.1 Detailed Description

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

4.1.2 Constructor & Destructor Documentation

4.1.2.1 [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](#).

4.1.2.2 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

<i>z</i>	The ZIP code (integer).
<i>p</i>	The place name (string).
<i>s</i>	The state name (string).
<i>c</i>	The county name (string).
<i>lat</i>	The latitude (double).
<i>lon</i>	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](#).

4.1.3 Member Function Documentation

4.1.3.1 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](#).

4.1.3.2 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](#).

4.1.3.3 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](#).

4.1.3.4 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](#).

4.1.3.5 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](#).

4.1.3.6 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](#).

4.1.3.7 setZip()

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

Set the ZIP code of the postal code item.

Parameters

<i>newZip</i>	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](#).

4.1.3.8 setPlace()

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

Set the place name of the postal code item.

Parameters

<i>newPlace</i>	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](#).

4.1.3.9 setState()

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

Set the state name of the postal code item.

Parameters

<i>newState</i>	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](#).

4.1.3.10 setCounty()

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

Set the county name of the postal code item.

Parameters

<i>newCounty</i>	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](#).

4.1.3.11 setLatitude()

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

Set the latitude of the postal code item.

Parameters

<i>newLat</i>	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](#).

4.1.3.12 setLongitude()

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

Set the longitude of the postal code item.

Parameters

<i>newLon</i>	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](#).

4.1.3.13 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](#)

4.2 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.

4.2.1 Detailed Description

Definition at line 23 of file `PostalList.h`.

4.2.2 Constructor & Destructor Documentation

4.2.2.1 PostalList()

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

4.2.3 Member Function Documentation

4.2.3.1 addItem()

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

Add a [PostalCodeItem](#) to the list.

Parameters

<i>item</i>	The PostalCodeItem to be added.
-------------	---

Definition at line 26 of file [PostalList.cpp](#).

4.2.3.2 getItem()

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

Get a [PostalCodeItem](#) by index.

Parameters

<i>index</i>	The index of the item to retrieve.
--------------	------------------------------------

Returns

The [PostalCodeItem](#) at the specified index.

Exceptions

<i>out_of_range</i>	if the index is invalid.
---------------------	--------------------------

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

4.2.3.3 findByZip()

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

Find a [PostalCodeItem](#) by its ZIP code.

Parameters

<i>zip</i>	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](#).

4.2.3.4 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](#).

4.2.3.5 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](#).

4.2.3.6 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](#).

4.2.3.7 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

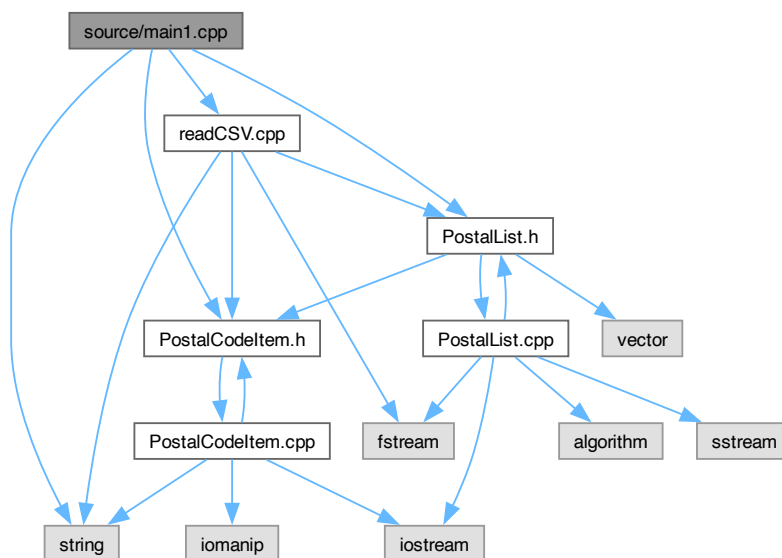
File Documentation

5.1 source/main1.cpp File Reference

Our first main file for "Zip Code Group 4 Project 1.0" that output the postal sorted by zip code.

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

Include dependency graph for main1.cpp:



Functions

- `int main ()`

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

5.1.1 Detailed Description

Our first main file for "Zip Code Group 4 Project 1.0" 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](#).

5.1.2 Function Documentation

5.1.2.1 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\(\)](#).

5.2 main1.cpp

[Go to the documentation of this file.](#)

```

00001
00026
00027 #include <string>
00028 #include "PostalCodeItem.h"
00029 #include "PostalList.h"
00030 #include "readCSV.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 <<
00066           "-----" <<
00067           endl;
00068     // Display the table sorted by zip
00069     myPostalList.printSortedByZip();
00070
00071     return 0;
00072 }
```

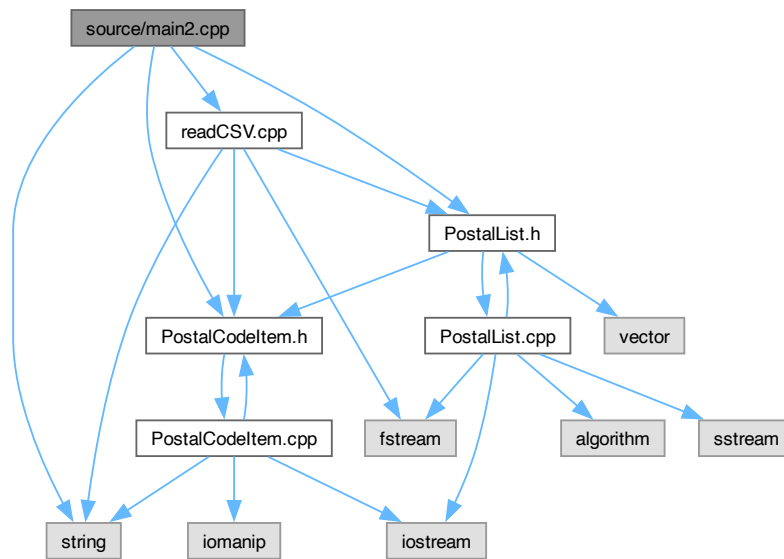
5.3 source/main2.cpp File Reference

Our second main file for "Zip Code Group 4 Project 1.0" that output the postal sorted by state.

```

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

Include dependency graph for main2.cpp:



Functions

- `int main ()`

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

5.3.1 Detailed Description

Our second main file for "Zip Code Group 4 Project 1.0" that output the postal sorted by state.

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

We read a 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](#).

5.3.2 Function Documentation

5.3.2.1 main()

```
int main ()
```

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

This [main2.cpp](#) will print the postal sorted by state.

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::printSortedByState\(\)](#).

5.4 main2.cpp

[Go to the documentation of this file.](#)

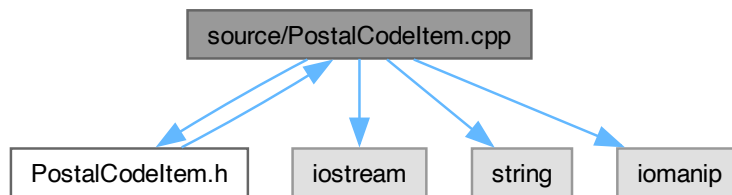
```
00001
00026
00027 #include <string>
00028 #include "PostalCodeItem.h"
00029 #include "PostalList.h"
00030 #include "readCSV.cpp"
00031
00032 using namespace std;
00033
00044 int main()
00045 {
00046     // Create a variable for csv file name
00047     string fileName = "us_postal_codes.csv";
00048
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 state:" << 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 <<
00066           "-----" <<
00067           endl;
00068     // Display the table sorted by state
00069     myPostalList.printSortedByState();
00070     return 0;
00071 }
```

5.5 source/PostalCodeItem.cpp File Reference

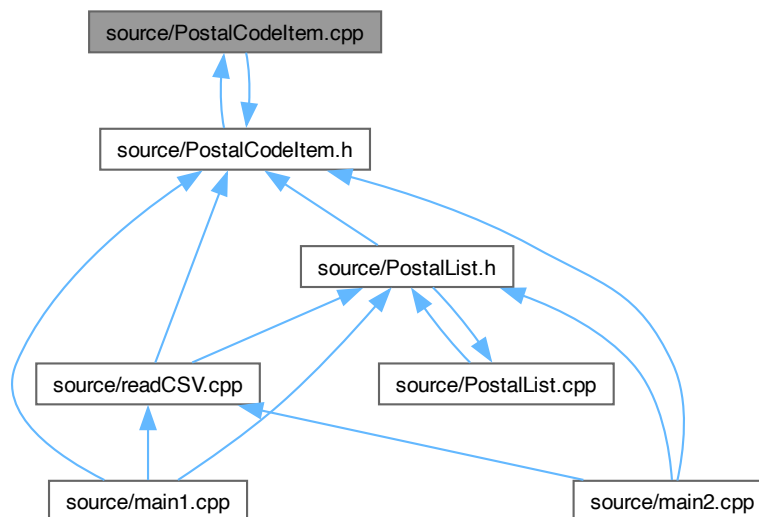
Implementation of the [PostalCodeItem](#) class representing a postal code entry.

```
#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:



5.5.1 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](#).

5.6 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 }

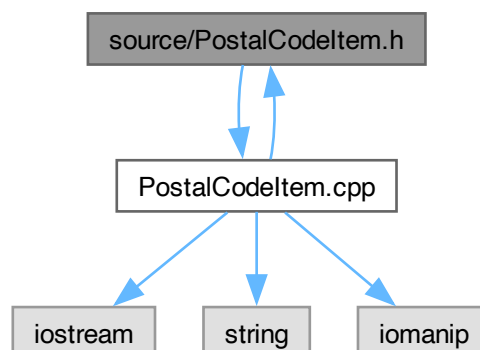
```

5.7 source/PostalCodeItem.h File Reference

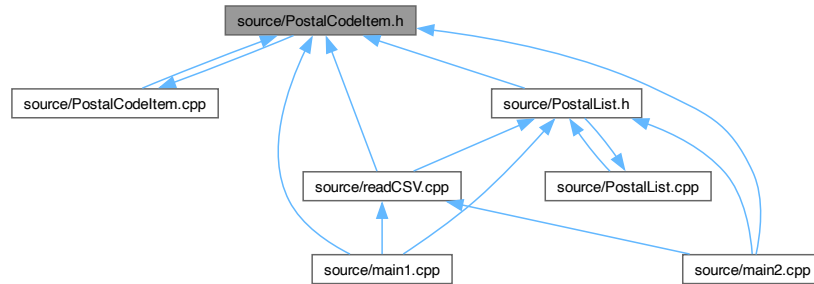
Defines the [PostalCodeItem](#) class for representing postal code records.

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

Include dependency graph for PostalCodeItem.h:



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



Classes

- class [PostalCodeItem](#)

5.7.1 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](#).

5.8 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

```

5.9 source/PostalList.cpp File Reference

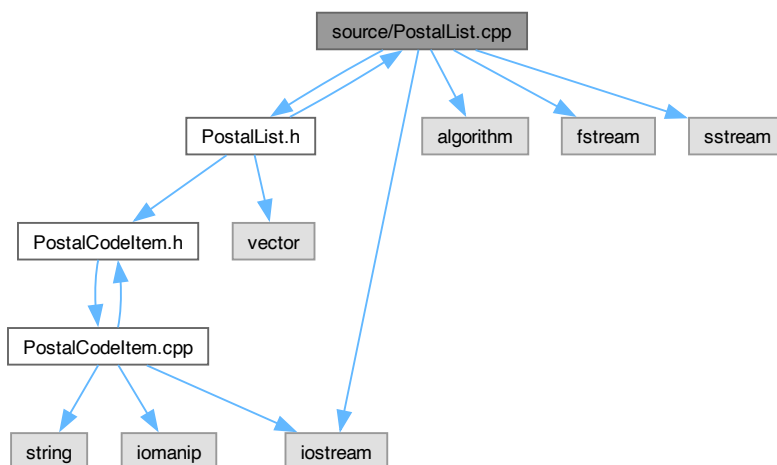
@ brief Implementation of the [PostalList](#) class for managing a collection of [PostalCodeItem](#) objects.

```

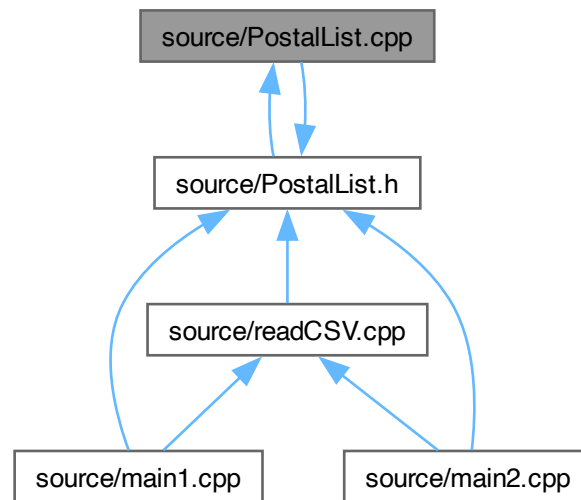
#include "PostalList.h"
#include <iostream>
#include <algorithm>
#include <fstream>
#include <sstream>

```

Include dependency graph for PostalList.cpp:



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



5.9.1 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](#).

5.10 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 *PostallList::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 PostallList::size() const
00069 {
00070     return items.size();
00071 }
00072
00078 void PostallList::printAll() const
00079 {
00080     for (int i = 0; i < items.size(); i++)
00081     {
00082         items[i].printInfo();
00083         cout <<
00084         endl;
00085     }
00086
00092 void PostallList::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 <<
00107         endl;
00108     }
00109
00115 void PostallList::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 <<
00134         endl;
00135     }

```

5.11 source/PostalList.h File Reference

Defines the [PostalList](#) class for managing collections of postal codes.

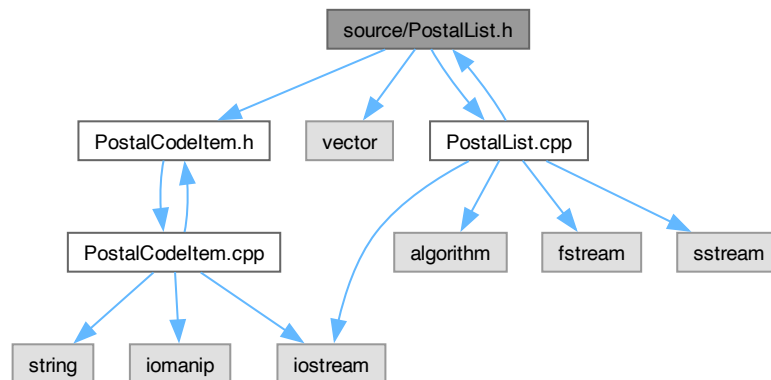
```

#include "PostalCodeItem.h"
#include <vector>

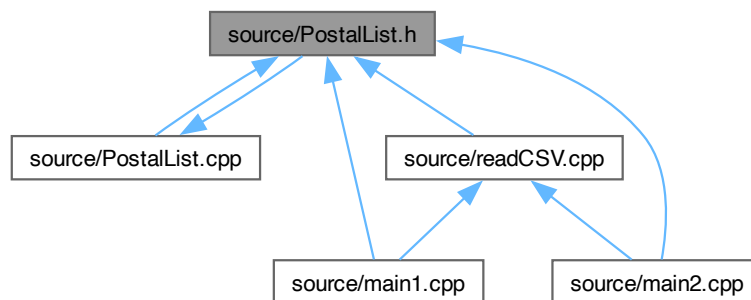
```

```
#include "PostalList.cpp"
```

Include dependency graph for PostalList.h:



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



Classes

- class [PostalList](#)

5.11.1 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](#).

5.12 PostalList.h

[Go to the documentation of this file.](#)

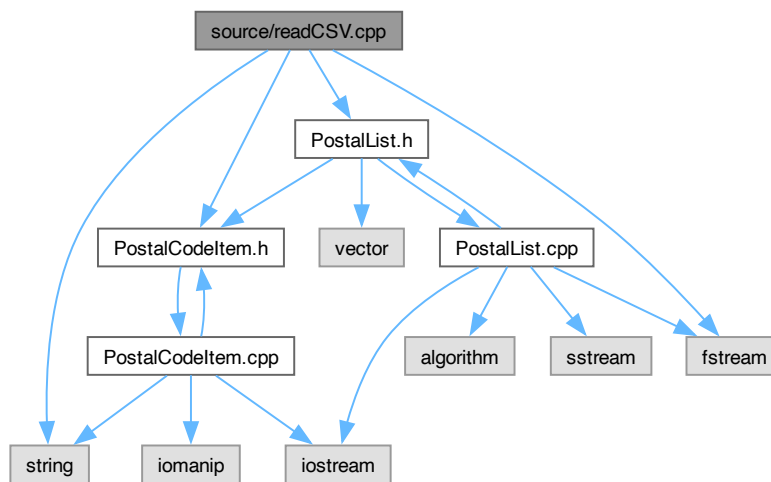
```
00001
00014
00015 #ifndef POSTAL_LIST_H
00016 #define POSTAL_LIST_H
00017
00018 #include "PostalCodeItem.h"
00019 #include <vector>
00020
00021 using namespace std;
00022
00023 class PostalList
00024 {
00025 private:
00026     vector<PostalCodeItem> items;
00027
00028 public:
00029     // Constructors
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
```

5.13 source/readCSV.cpp File Reference

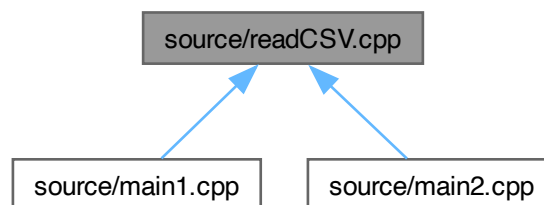
Utility functions for reading postal code data from a CSV file.

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

Include dependency graph for readCSV.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.

5.13.1 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 [readCSV.cpp](#).

5.13.2 Function Documentation

5.13.2.1 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

<i>inputList</i>	Where we store all the items.
<i>fileName</i>	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 [readCSV.cpp](#).

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

5.14 readCSV.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 }
```

