Project 2.0

Generated by Doxygen 1.13.2

Chapter 1

Class Index

1.1 Class List

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

Chapter 2

File Index

2.1 File List

Here is a list of all files with brief descriptions:

Chapter 3

Class Documentation

3.1 Buffer Class Reference

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

Public Member Functions

- void pack (const std::string &data)

  Packs a string into a length-indicated format.
- std::string unpack ()

  Unpacks a length-indicated string from the buffer.
- void readHeader (std::ifstream &file)

  Reads the header record from the input file stream.
- void writeHeader (std::ofstream &file)

  Writes the header record to the output file stream.

Private Attributes

- std::string buffer

3.1.1 Member Function Documentation

3.1.1.1 pack()

```
void Buffer::pack (
            const std::string & data)
```

Packs a string into a length-indicated format.

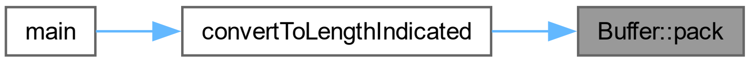
Converts the input string into a format where the first part of the packed data indicates the string's length, followed by the actual string.

Example: Input: "Hello" Output: "5,Hello"

Parameters

data	The string to be packed.

Here is the caller graph for this function:



3.1.1.2 readHeader()

```
void Buffer::readHeader (
             std::ifstream & file)
```

Reads the header record from the input file stream.

Reads a single line from the provided input file stream and stores it in the buffer. This function is used to process the header information of a file.

Parameters

file	An input file stream object from which the header is read.

3.1.1.3 unpack()

```
std::string Buffer::unpack ()
```

Unpacks a length-indicated string from the buffer.

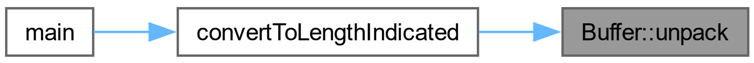
Extracts the string from a buffer containing a length-indicated format. The length is used to validate the extracted substring.

Example: Input Buffer: "5,Hello" Output: "Hello"

Returns

The unpacked string.

Here is the caller graph for this function:



3.1.1.4 writeHeader()

```
void Buffer::writeHeader (
             std::ofstream & file)
```

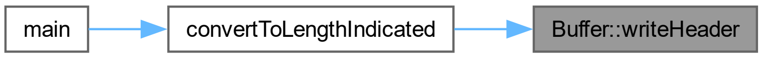
Writes the header record to the output file stream.

Writes the content of the buffer, typically containing a header, to the output file stream.

Parameters

file	An output file stream object to which the header is written.

Here is the caller graph for this function:



3.1.2 Member Data Documentation

3.1.2.1 buffer

```
std::string Buffer::buffer  [private]
```

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

- Buffer.h
- Buffer.cpp

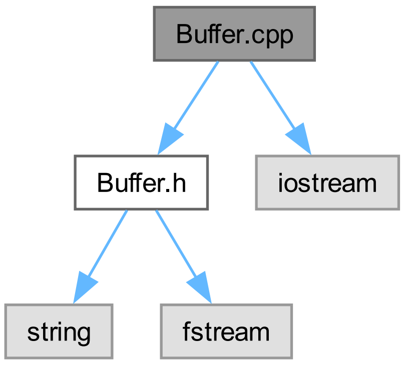
Chapter 4

File Documentation

4.1 Buffer.cpp File Reference

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

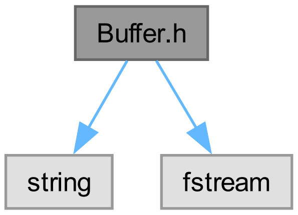
Include dependency graph for Buffer.cpp:



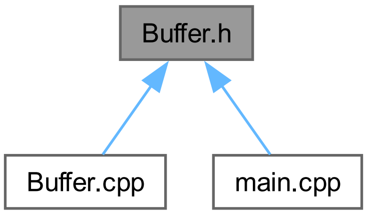
4.2 Buffer.h File Reference

```
#include <string>
#include <fstream>
```

Include dependency graph for Buffer.h:



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



Classes

- class Buffer

4.3 Buffer.h

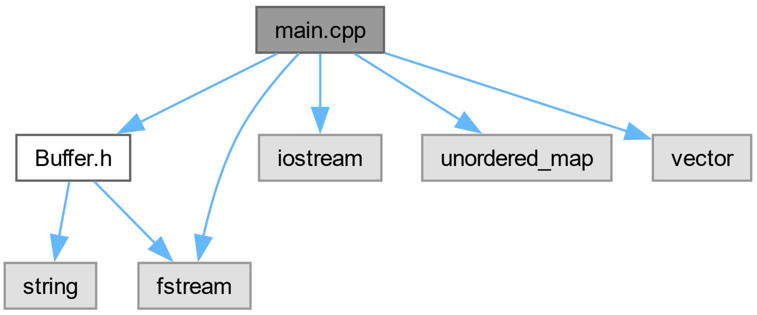
Go to the documentation of this file.

```
#ifndef BUFFER_H
#define BUFFER_H

#include <string>
#include <fstream>

class Buffer {
public:
    void pack(const std::string& data);
    std::string unpack();
    void readHeader(std::ifstream& file);
    void writeHeader(std::ofstream& file);

private:
    std::string buffer;
};

#endif
```

4.4 main.cpp File Reference

Converts CSV files to length-indicated format and creates primary key indices.

```
#include "Buffer.h"
#include <fstream>
#include <iostream>
#include <unordered_map>
#include <vector>
```
Include dependency graph for main.cpp:



Functions

- void convertToLengthIndicated (const string &inputFile, const string &outputFile)

  Converts a CSV file to a length-indicated format.
- unordered_map< string, size_t > createIndex (const string &filename)

  Creates an index mapping primary keys (Zip Codes) to file offsets.
- void writeIndexToFile (const unordered_map< string, size_t > &index, const string &filename)

  Writes the created index to a file.
- int main (int argc, char *argv[ ])

  Main function to execute CSV conversion, indexing, and zip code lookup.

4.4.1 Detailed Description

Converts CSV files to length-indicated format and creates primary key indices.

This program processes CSV files to transform them into a length-indicated format. It then creates an index based on the primary key (Zip Code) and writes this index to a file. The program also supports searching for zip codes using the generated index.

Usage: Run the program with optional command-line arguments to search for specific zip codes. Example↩ : ./program -Z12345 -Z67890

4.4.2 Function Documentation

4.4.2.1 convertToLengthIndicated()

```
void convertToLengthIndicated (
             const string & inputFile,
             const string & outputFile)
```

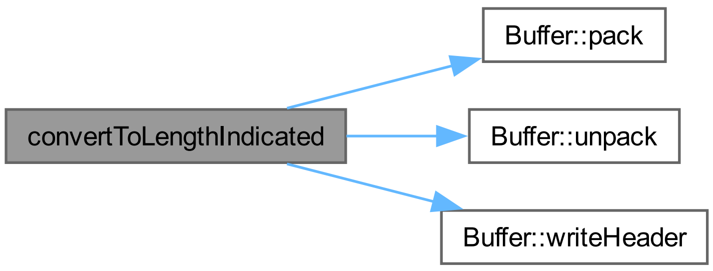
Converts a CSV file to a length-indicated format.

Reads data from the input CSV file, packs each line into a length-indicated format, and writes it to the output file.

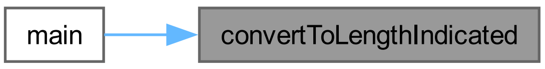
Parameters

inputFile	The name of the input CSV file.
outputFile	The name of the output file to store length-indicated records.

Here is the call graph for this function:



Here is the caller graph for this function:



4.4.2.2 createIndex()

```
unordered_map< string, size_t > createIndex (
             const string & filename)
```

Creates an index mapping primary keys (Zip Codes) to file offsets.

Reads the CSV file and extracts the Zip Code from each record. Stores the file offset of each record in an unordered_map for fast lookup.

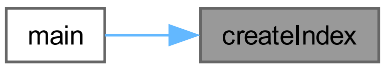
Parameters

filename	The name of the file containing length-indicated records.

Returns

An unordered_map containing Zip Code keys and their respective file offsets.

Here is the caller graph for this function:



4.4.2.3 main()

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

Main function to execute CSV conversion, indexing, and zip code lookup.

Converts CSV files to length-indicated format, generates primary key indices, and supports searching for zip codes using the index.

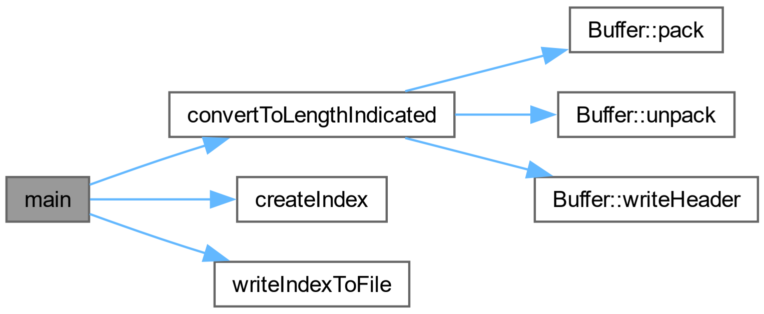
Parameters

argc	Number of command-line arguments.
argv	Command-line arguments.

Returns

Exit status (0 on success, 1 on failure).

Here is the call graph for this function:



4.4.2.4 writeIndexToFile()

```
void writeIndexToFile (
            const unordered_map< string, size_t > & index,
            const string & filename)
```

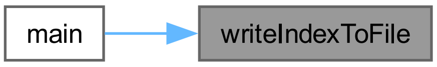
Writes the created index to a file.

Outputs the Zip Code and corresponding file offset to a specified file.

Parameters

index	The unordered_map containing Zip Code keys and file offsets.
filename	The name of the output file where the index will be stored.

Here is the caller graph for this function:



Index