Project 3.0

Generated by Doxygen 1.9.8

1 Class Index	1
1.1 Class List	1
2 File Index	3
2.1 File List	3
3 Class Documentation	5
3.1 Block Class Reference	5
3.1.1 Constructor & Destructor Documentation	5
3.1.1.1 Block()	5
3.1.2 Member Function Documentation	6
3.1.2.1 deserialize()	6
3.1.2.2 dump()	6
3.1.2.3 serialize()	6
3.1.3 Member Data Documentation	6
3.1.3.1 blockNumber	6
3.1.3.2 nextBlock	7
3.1.3.3 records	7
3.2 BlockBuffer Class Reference	7
3.2.1 Member Function Documentation	7
3.2.1.1 readBlocks()	7
3.2.1.2 writeBlocks()	8
3.3 Buffer Class Reference	8
3.3.1 Member Function Documentation	9
3.3.1.1 getBuffer()	9
3.3.1.2 pack()	9
3.3.1.3 readHeader()	9
3.3.1.4 unpack()	9
3.3.1.5 writeHeader()	10
3.4 Record Class Reference	10
3.4.1 Constructor & Destructor Documentation	10
3.4.1.1 Record()	10
3.4.2 Member Function Documentation	10
3.4.2.1 deserialize()	10
3.4.2.2 serialize()	11
3.4.3 Member Data Documentation	11
3.4.3.1 field1	11
3.4.3.2 field2	11
3.4.3.3 field3	11
3.4.3.4 index	11
4 File Documentation	13
4.1 Block.h File Reference	13

	4.2 Block.h	14
	4.3 BlockBuffer.cpp File Reference	15
	4.4 BlockBuffer.h File Reference	16
	4.5 BlockBuffer.h	16
	4.6 Buffer.cpp File Reference	17
	4.7 Buffer.h File Reference	17
	4.8 Buffer.h	18
	4.9 main.cpp File Reference	18
	4.9.1 Function Documentation	19
	4.9.1.1 createBlocks()	19
	4.9.1.2 dumpLogical()	19
	4.9.1.3 dumpPhysical()	20
	4.9.1.4 main()	20
	4.9.1.5 readCSV()	20
	4.10 Record.h File Reference	20
	4.11 Record.h	21
Inc	dex	23

# **Chapter 1**

# **Class Index**

## 1.1 Class List

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

Block	5
BlockBuffer	7
Buffer	
Record	10

2 Class Index

# **Chapter 2**

# File Index

## 2.1 File List

Here is a list of all files with brief descriptions:

ock.h	1
ockBuffer.cpp	1
ockBuffer.h	1
uffer.cpp	1
uffer.h	1
ain.cpp	1
ecord.h	2

File Index

## **Chapter 3**

## **Class Documentation**

## 3.1 Block Class Reference

```
#include <Block.h>
```

#### **Public Member Functions**

- Block ()
- std::string serialize () const

Serializes the block to a string.

• void dump () const

Dumps the block content to standard output.

### **Static Public Member Functions**

• static Block deserialize (const std::string &data)

Deserializes a block from a string.

### **Public Attributes**

• int blockNumber

Sequential number of the block.

• int nextBlock

Logical pointer to the next block (-1 if none).

std::vector< Record > records

List of records in this block.

### 3.1.1 Constructor & Destructor Documentation

## 3.1.1.1 Block()

```
Block::Block ( ) [inline]
```

6 Class Documentation

## 3.1.2 Member Function Documentation

#### 3.1.2.1 deserialize()

Deserializes a block from a string.

Expects the first line to be the block header. Each subsequent line is a packed record.

#### **Parameters**

data	The serialized block string.
------	------------------------------

#### Returns

A Block object.

### 3.1.2.2 dump()

```
void Block::dump ( ) const [inline]
```

Dumps the block content to standard output.

## 3.1.2.3 serialize()

```
std::string Block::serialize ( ) const [inline]
```

Serializes the block to a string.

First writes a header line: blockNumber,recordCount,nextBlock Then, for each record, packs the record using Buffer and writes the result.

### Returns

The serialized block string.

### 3.1.3 Member Data Documentation

## 3.1.3.1 blockNumber

int Block::blockNumber

Sequential number of the block.

### 3.1.3.2 nextBlock

```
int Block::nextBlock
```

Logical pointer to the next block (-1 if none).

#### 3.1.3.3 records

```
std::vector<Record> Block::records
```

List of records in this block.

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

· Block.h

## 3.2 BlockBuffer Class Reference

```
#include <BlockBuffer.h>
```

### **Public Member Functions**

- bool writeBlocks (const std::string &filename, const std::vector < Block > &blocks)
   Writes a blocked sequence set file.
- bool readBlocks (const std::string &filename, std::vector < Block > &blocks)
   Reads a blocked sequence set file.

## 3.2.1 Member Function Documentation

### 3.2.1.1 readBlocks()

Reads a blocked sequence set file.

## **Parameters**

filename	The input file name.	
blocks	A vector to receive the blocks.	

#### Returns

true on success.

8 Class Documentation

### 3.2.1.2 writeBlocks()

Writes a blocked sequence set file.

The file consists of:

- A file header (packed using Buffer)
- · A line with the number of blocks
- For each block: a length indicator (the size of the packed block) and the packed block data.

#### **Parameters**

filename	The output file name.
blocks	A vector of blocks to write.

#### Returns

true on success.

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

- · BlockBuffer.h
- BlockBuffer.cpp

## 3.3 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 the string (ignores the length indicator).

• 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.

• std::string getBuffer () const

Returns the internal packed string.

3.3 Buffer Class Reference 9

## 3.3.1 Member Function Documentation

### 3.3.1.1 getBuffer()

```
std::string Buffer::getBuffer ( ) const [inline]
```

Returns the internal packed string.

Returns

The packed string.

#### 3.3.1.2 pack()

Packs a string into a length-indicated format.

Example: "Hello" becomes "5,Hello"

**Parameters** 

data The string to pack.

## 3.3.1.3 readHeader()

Reads the header record from the input file stream.

#### **Parameters**

file The input stream.

## 3.3.1.4 unpack()

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

Unpacks the string (ignores the length indicator).

Unpacks a length-indicated string from the buffer.

Returns

The original string.

10 Class Documentation

#### 3.3.1.5 writeHeader()

Writes the header record to the output file stream.

#### **Parameters**

```
file The output stream.
```

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

- Buffer.h
- · Buffer.cpp

## 3.4 Record Class Reference

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

## **Public Member Functions**

- Record ()
- std::string serialize () const

Serializes the record as a CSV string.

#### **Static Public Member Functions**

• static Record deserialize (const std::string &data)

Deserializes a CSV string into a Record.

## **Public Attributes**

- int index
- std::string field1
- std::string field2
- std::string field3

## 3.4.1 Constructor & Destructor Documentation

## 3.4.1.1 Record()

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

### 3.4.2 Member Function Documentation

#### 3.4.2.1 deserialize()

Deserializes a CSV string into a Record.

#### **Parameters**

data	The CSV string.
------	-----------------

### Returns

A Record object.

## 3.4.2.2 serialize()

```
std::string Record::serialize ( ) const [inline]
```

Serializes the record as a CSV string.

Format: index,field1,field2,field3

### 3.4.3 Member Data Documentation

## 3.4.3.1 field1

std::string Record::field1

## 3.4.3.2 field2

std::string Record::field2

## 3.4.3.3 field3

std::string Record::field3

#### 3.4.3.4 index

int Record::index

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

· Record.h

12 Class Documentation

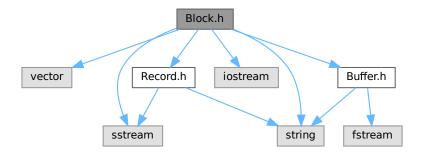
## **Chapter 4**

## **File Documentation**

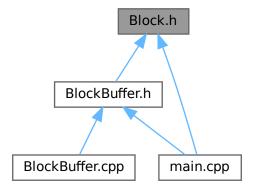
#### 4.1 **Block.h File Reference**

```
#include <vector>
#include <string>
#include <sstream>
#include <iostream>
#include "Record.h"
#include "Buffer.h"
```

Include dependency graph for Block.h:



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



#### **Classes**

· class Block

## 4.2 Block.h

#### Go to the documentation of this file.

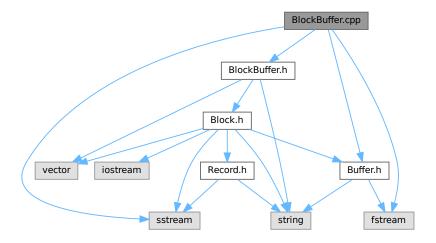
```
00001 #ifndef BLOCK_H
00002 #define BLOCK_H
00003
00004 #include <vector>
00005 #include <string>
00006 #include <sstream>
00007 #include <iostream>
00008 #include "Record.h"
00009 #include "Buffer.h"
00010
00011 class Block {
00012 public:
00013
          int blockNumber;
00014
           int nextBlock;
00015
          std::vector<Record> records;
00016
          Block() : blockNumber(0), nextBlock(-1) {}
00017
00018
00027
           std::string serialize() const {
00028
              std::stringstream ss;
               // Write block header.
ss « blockNumber « "," « records.size() « "," « nextBlock « "\n";
00029
00030
               \ensuremath{//} Write each record (packed with Buffer).
00031
00032
               for (const auto &rec : records) {
00033
                   Buffer buf;
00034
                    std::string recStr = rec.serialize();
00035
                   buf.pack(recStr);
00036
                   ss « buf.getBuffer() « "\n";
00037
00038
               return ss.str();
00039
          }
00040
00050
          static Block deserialize(const std::string &data) {
00051
              Block blk;
00052
               std::stringstream ss(data);
00053
               std::string line;
00054
               // Get header line.
00055
               if (getline(ss, line)) {
00056
                    std::stringstream headerStream(line);
```

```
std::string token;
00058
                   getline(headerStream, token, ',');
00059
                   blk.blockNumber = std::stoi(token);
                   getline(headerStream, token, ','); // record count (not used here)
00060
00061
                   int recordCount = std::stoi(token);
                   getline(headerStream, token, ',');
00062
00063
                   blk.nextBlock = std::stoi(token);
00064
00065
               // Read each packed record.
               while (getline(ss, line)) {
   if (line.empty())
00066
00067
00068
                        continue:
00069
                   // Unpack the record manually.
00070
                   size_t commaPos = line.find(',');
00071
                   if (commaPos == std::string::npos)
00072
                        continue;
00073
                   int len = std::stoi(line.substr(0, commaPos));
                   record r = Record::deserialize(recData);
00074
00075
00076
                   blk.records.push_back(r);
00077
00078
               return blk;
00079
          }
00080
00084
          void dump() const {
00085
            std::cout « "Block Number: " « blockNumber « ", Next Block: " « nextBlock « std::endl; std::cout « "Records:" « std::endl;
00086
               for (const auto &r: records) {
    std::cout « r.index « " | " « r.field1 « " | " « r.field2 « " | " « r.field3 « std::end1;
00087
00088
00089
00090
00091 };
00092
00093 #endif
```

## 4.3 BlockBuffer.cpp File Reference

```
#include "BlockBuffer.h"
#include "Buffer.h"
#include <fstream>
#include <sstream>
```

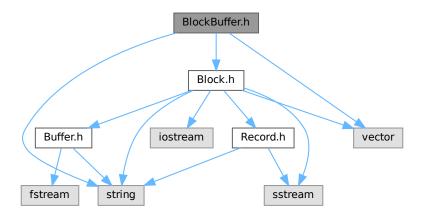
Include dependency graph for BlockBuffer.cpp:



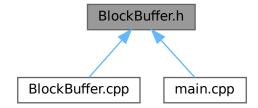
#### BlockBuffer.h File Reference 4.4

```
#include <string>
#include <vector>
#include "Block.h"
```

Include dependency graph for BlockBuffer.h:



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



#### Classes

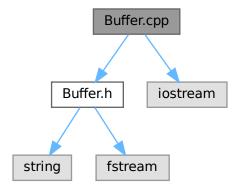
class BlockBuffer

#### BlockBuffer.h 4.5

```
Go to the documentation of this file.
00001 #ifndef BLOCKBUFFER_H
00002 #define BLOCKBUFFER_H
00004 #include <string>
```

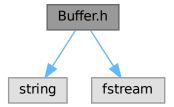
## 4.6 Buffer.cpp File Reference

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

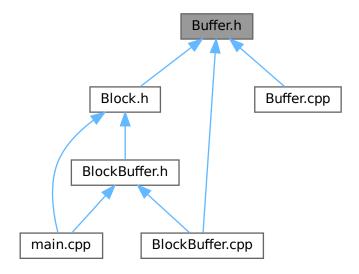


## 4.7 Buffer.h File Reference

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



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



#### Classes

class Buffer

## 4.8 Buffer.h

## Go to the documentation of this file.

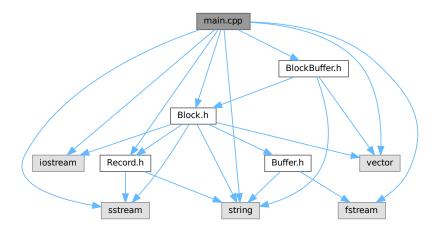
```
00001 #ifndef BUFFER_H
00002 #define BUFFER_H
00003
00004 #include <string>
00005 #include <fstream>
00006
00007 class Buffer {
00008 public:
00016
          void pack(const std::string& data);
00017
00023
          std::string unpack();
00024
00030
          void readHeader(std::ifstream& file);
00031
00037
          void writeHeader(std::ofstream& file);
00038
00044
          std::string getBuffer() const { return buffer; }
00045
00046 private:
00047
          std::string buffer;
00048 };
00049
00050 #endif
```

## 4.9 main.cpp File Reference

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

```
#include <sstream>
#include <vector>
#include <string>
#include "BlockBuffer.h"
#include "Block.h"
#include "Record.h"
```

Include dependency graph for main.cpp:



#### **Functions**

- vector< string > readCSV (const string &filename)
  - Reads a CSV file (with a header) and returns a vector of CSV record strings.
- vector < Block > createBlocks (const vector < string > &records, int recordsPerBlock)
   Creates blocks from CSV record strings.
- void dumpPhysical (const vector < Block > &blocks)
  - Dump blocks in physical order (as stored in file).
- void dumpLogical (const vector< Block > &blocks)
  - Dump blocks in logical order (following nextBlock pointer).
- int main (int argc, char \*argv[])

#### 4.9.1 Function Documentation

## 4.9.1.1 createBlocks()

Creates blocks from CSV record strings.

## 4.9.1.2 dumpLogical()

```
void dumpLogical ( {\tt const\ vector} < {\tt Block} \ > \ \& \ blocks \ )
```

Dump blocks in logical order (following nextBlock pointer).

## 4.9.1.3 dumpPhysical()

```
void dumpPhysical ( {\tt const\ vector} < {\tt Block} \ > \ \& \ blocks \ )
```

Dump blocks in physical order (as stored in file).

## 4.9.1.4 main()

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

## 4.9.1.5 readCSV()

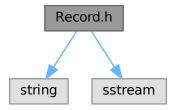
```
vector< string > readCSV ( {\tt const\ string\ \&\ \it filename\ )}
```

Reads a CSV file (with a header) and returns a vector of CSV record strings.

## 4.10 Record.h File Reference

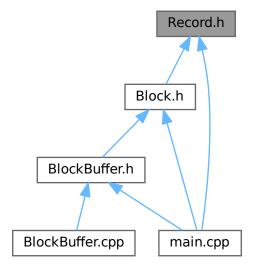
```
#include <string>
#include <sstream>
```

Include dependency graph for Record.h:



4.11 Record.h 21

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



#### **Classes**

class Record

## 4.11 Record.h

## Go to the documentation of this file.

```
00001 #ifndef RECORD_H
00002 #define RECORD_H
00003
00004 #include <string>
00005 #include <sstream>
00006
00007 class Record {
00008 public:
         int index;
00009
00010
           std::string field1;
00011
           std::string field2;
00012
           std::string field3;
00013
00014
           Record() : index(0) {}
00015
00021
           std::string serialize() const {
              std::stringstream ss;
ss « index « "," « field1 « "," « field2 « "," « field3;
00022
00023
                return ss.str();
00024
00025
           }
00026
00033
           static Record deserialize(const std::string &data) {
            Record r;
00034
00035
                std::stringstream ss(data);
               std::string token;
getline(ss, token, ',');
r.index = std::stoi(token);
00036
00037
00038
               getline(ss, r.field1, ',');
getline(ss, r.field2, ',');
00039
00040
                getline(ss, r.field3, ',');
00041
00042
                return r;
00043
00044 };
00045
00046 #endif
```

# Index

Block, 5	main
Block, 5	main.cpp, 20
blockNumber, 6	main.cpp, 18
deserialize, 6	createBlocks, 19
dump, 6	dumpLogical, 19
nextBlock, 6	dumpPhysical, 19
records, 7	main, 20
serialize, 6	readCSV, 20
Block.h, 13	n a vitDla alc
BlockBuffer, 7	nextBlock
readBlocks, 7	Block, 6
writeBlocks, 7	pack
BlockBuffer.cpp, 15	Buffer, 9
BlockBuffer.h, 16	Danci, V
blockNumber	readBlocks
Block, 6	BlockBuffer, 7
Buffer, 8	readCSV
getBuffer, 9	main.cpp, 20
pack, 9	readHeader
readHeader, 9	Buffer, 9
unpack, 9	Record, 10
writeHeader, 9	deserialize, 10
Buffer.cpp, 17	field1, 11
Buffer.h, 17	field2, 11
. 81	field3, 11
createBlocks	index, 11
main.cpp, 19	Record, 10
donorializa	serialize, 11
deserialize	Record.h, 20
Block, 6	records
Record, 10	Block, 7
dump	Blook, 7
Block, 6	serialize
dumpLogical	Block, 6
main.cpp, 19	Record, 11
dumpPhysical	,
main.cpp, 19	unpack
field1	Buffer, 9
Record, 11	
field2	writeBlocks
Record, 11	BlockBuffer, 7
field3	writeHeader
Record, 11	Buffer, 9
Hecord, TT	
getBuffer	
Buffer, 9	
indov	
INCOV	

Record, 11