Project 3

Generated by Doxygen 1.10.0

Chapter 1

Hierarchical Index

1.1 Class Hierarchy

This inheritance list is sorted roughly, but not completely, alphabetically:

Chapter 2

Class Index

2.1 Class List

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

Chapter 3

File Index

3.1 File List

Here is a list of all files with brief descriptions:

Chapter 4

Class Documentation

4.1 BufferFile Class Reference

`#include <buffile.h>`

Public Member Functions

- BufferFile (IOBuffer &)
- int Open (const char *filename, std::ios_base::openmode MODE)
- int Create (const char *filename, std::ios_base::openmode MODE)
- int Close ()
- int Rewind ()
- int Read (int recaddr=-1)
- int Write (int recaddr=-1)
- int Append ()
- IOBuffer & GetBuffer ()

Protected Member Functions

- int ReadHeader ()
- int WriteHeader ()

Protected Attributes

- IOBuffer & Buffer
- fstream File
- int HeaderSize

4.1.1 Constructor & Destructor Documentation

4.1.1.1 BufferFile()

```
BufferFile::BufferFile (
            IOBuffer & from )
```

4.1.2 Member Function Documentation

4.1.2.1 Append()

```
int BufferFile::Append ( )
```

4.1.2.2 Close()

```
int BufferFile::Close ( )
```

4.1.2.3 Create()

```
int BufferFile::Create (
            const char * filename,
            std::ios_base::openmode MODE )
```

4.1.2.4 GetBuffer()

```
IOBuffer & BufferFile::GetBuffer ( )
```

4.1.2.5 Open()

```
int BufferFile::Open (
            const char * filename,
            std::ios_base::openmode MODE )
```

4.1.2.6 Read()

```
int BufferFile::Read (
            int recaddr = -1 )
```

4.1.2.7 ReadHeader()

```
int BufferFile::ReadHeader ( )  [protected]
```

4.1.2.8 Rewind()

```
int BufferFile::Rewind ( )
```

4.1.2.9 Write()

```
int BufferFile::Write (
            int recaddr = -1 )
```

4.1.2.10 WriteHeader()

```
int BufferFile::WriteHeader ( )  [protected]
```

4.1.3 Member Data Documentation

4.1.3.1 Buffer

```
IOBuffer& BufferFile::Buffer  [protected]
```

4.1.3.2 File

```
fstream BufferFile::File  [protected]
```

4.1.3.3 HeaderSize

```
int BufferFile::HeaderSize  [protected]
```

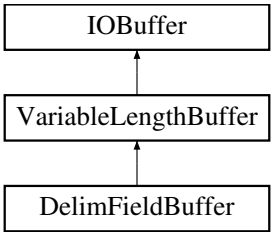
The documentation for this class was generated from the following files:

- C:/Users/silas/CLionProjects/project3_khalil/buffile.h
- C:/Users/silas/CLionProjects/project3_khalil/buffile.cpp

4.2 DelimFieldBuffer Class Reference

```
#include <delim.h>
```

Inheritance diagram for DelimFieldBuffer:



Public Member Functions

- DelimFieldBuffer (char Delim=-1, int maxBytes=1000)
- DelimFieldBuffer (const DelimFieldBuffer &buffer)
- void Clear ()
- int Pack (const void *, int size=-1)
- int Unpack (void *field, int maxBytes=-1)
- int ReadHeader (istream &stream)
- int WriteHeader (ostream &stream) const
- void Print (ostream &) const
- int Init (char delim=0)

Public Member Functions inherited from VariableLengthBuffer

- VariableLengthBuffer (int MaxBytes=1000)
- VariableLengthBuffer (const VariableLengthBuffer &buffer)
- void Clear ()
- int Read (istream &)
- int Write (ostream &) const
- int ReadHeader (istream &)
- int WriteHeader (ostream &) const
- int PackFixLen (void *, int)
- int PackDelimeted (void *, int)
- int PackLength (void *, int)
- void Print (ostream &) const
- int SizeOfBuffer () const
- int Init ()

Public Member Functions inherited from IOBuffer

- IOBuffer (int maxBytes=1000)
- IOBuffer & operator= (const IOBuffer &)
- int Init (int maxBytes)
- virtual int DRead (istream &, int recref)
- virtual int DWrite (ostream &, int recref) const

Static Public Member Functions

- static void SetDefaultDelim (char delim)

Protected Attributes

- char Delim

Protected Attributes inherited from IOBuffer

- int Initialized
- char * Buffer
- int BufferSize
- int MaxBytes
- int NextByte
- int Packing

Static Protected Attributes

- static char DefaultDelim = 0

4.2.1 Constructor & Destructor Documentation

4.2.1.1 DelimFieldBuffer() [1/2]

```
DelimFieldBuffer::DelimFieldBuffer (
            char Delim = -1,
            int maxBytes = 1000 )
```

4.2.1.2 DelimFieldBuffer() [2/2]

```
DelimFieldBuffer::DelimFieldBuffer (
            const DelimFieldBuffer & buffer )  [inline]
```

4.2.2 Member Function Documentation

4.2.2.1 Clear()

```
void DelimFieldBuffer::Clear ( )  [virtual]
```

Reimplemented from IOBuffer.

4.2.2.2 Init()

```
int DelimFieldBuffer::Init (
            char delim = 0 )
```

4.2.2.3 Pack()

```
int DelimFieldBuffer::Pack (
            const void * field,
            int size = -1 )  [virtual]
```

Implements IOBuffer.

4.2.2.4 Print()

```
void DelimFieldBuffer::Print (
            ostream & stream ) const  [virtual]
```

Reimplemented from IOBuffer.

4.2.2.5 ReadHeader()

```
int DelimFieldBuffer::ReadHeader (
            istream & stream )  [virtual]
```

Reimplemented from IOBuffer.

4.2.2.6 SetDefaultDelim()

```
void DelimFieldBuffer::SetDefaultDelim (
            char delim )  [static]
```

4.2.2.7 Unpack()

```
int DelimFieldBuffer::Unpack (
            void * field,
            int maxBytes = -1 )  [virtual]
```

Implements IOBuffer.

4.2.2.8 WriteHeader()

```
int DelimFieldBuffer::WriteHeader (
            ostream & stream ) const  [virtual]
```

Reimplemented from IOBuffer.

4.2.3 Member Data Documentation

4.2.3.1 DefaultDelim

```
char DelimFieldBuffer::DefaultDelim = 0  [static], [protected]
```

4.2.3.2 Delim

```
char DelimFieldBuffer::Delim  [protected]
```

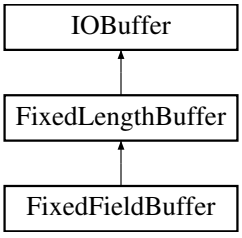
The documentation for this class was generated from the following files:

- C:/Users/silas/CLionProjects/project3_khalil/delim.h
- C:/Users/silas/CLionProjects/project3_khalil/delim.cpp

4.3 FixedFieldBuffer Class Reference

```
#include <fixfld.h>
```

Inheritance diagram for FixedFieldBuffer:



Public Member Functions

- FixedFieldBuffer (int maxFields, int RecordSize=1000)
- FixedFieldBuffer (int maxFields, int *fieldSize)
- FixedFieldBuffer (const FixedFieldBuffer &)
- FixedFieldBuffer & operator= (const FixedFieldBuffer &)
- void Clear ()
- int AddField (int fieldSize)
- int ReadHeader (istream &)
- int WriteHeader (ostream &) const
- int Pack (const void *field, int size=-1)
- int Unpack (void *field, int maxBytes=-1)
- void Print (ostream &) const
- int NumberOfFields () const
- int Init (int maxFields)
- int Init (int numFields, int *fieldSize)

Public Member Functions inherited from FixedLengthBuffer

- FixedLengthBuffer (int recordSize=1000)
- FixedLengthBuffer (const FixedLengthBuffer &buffer)
- void Clear ()
- int Read (istream &)
- int Write (ostream &) const
- int ReadHeader (istream &)
- int WriteHeader (ostream &) const
- void Print (ostream &) const
- int SizeOfBuffer () const

Public Member Functions inherited from IOBuffer

- IOBuffer (int maxBytes=1000)
- IOBuffer & operator= (const IOBuffer &)
- int Init (int maxBytes)
- virtual int DRead (istream &, int recref)
- virtual int DWrite (ostream &, int recref) const

Protected Attributes

- int * FieldSize
- int MaxFields
- int NumFields
- int NextField

Protected Attributes inherited from IOBuffer

- int Initialized
- char * Buffer
- int BufferSize
- int MaxBytes
- int NextByte
- int Packing

Additional Inherited Members

Protected Member Functions inherited from FixedLengthBuffer

- int Init (int recordSize)
- int ChangeRecordSize (int recordSize)

4.3.1 Constructor & Destructor Documentation

4.3.1.1 FixedFieldBuffer() [1/3]

```
FixedFieldBuffer::FixedFieldBuffer (
            int maxFields,
            int RecordSize = 1000 )
```

4.3.1.2 FixedFieldBuffer() [2/3]

```
FixedFieldBuffer::FixedFieldBuffer (
            int maxFields,
            int * fieldSize )
```

4.3.1.3 FixedFieldBuffer() [3/3]

```
FixedFieldBuffer::FixedFieldBuffer (
            const FixedFieldBuffer & buffer )  [inline]
```

4.3.2 Member Function Documentation

4.3.2.1 AddField()

```
int FixedFieldBuffer::AddField (
            int fieldSize )
```

4.3.2.2 Clear()

```
void FixedFieldBuffer::Clear ( )  [virtual]
```

Reimplemented from IOBuffer.

4.3.2.3 Init() [1/2]

```
int FixedFieldBuffer::Init (
            int maxFields )
```

4.3.2.4 Init() [2/2]

```
int FixedFieldBuffer::Init (
            int numFields,
            int * fieldSize )
```

4.3.2.5 NumberOfFields()

```
int FixedFieldBuffer::NumberOfFields ( ) const
```

4.3.2.6 operator=()

```
FixedFieldBuffer & FixedFieldBuffer::operator= (
            const FixedFieldBuffer & buffer )
```

4.3.2.7 Pack()

```
int FixedFieldBuffer::Pack (
            const void * field,
            int size = -1 )  [virtual]
```

Implements IOBuffer.

4.3.2.8 Print()

```
void FixedFieldBuffer::Print (
            ostream & stream ) const  [virtual]
```

Reimplemented from IOBuffer.

4.3.2.9 ReadHeader()

```
int FixedFieldBuffer::ReadHeader (
            istream & stream )  [virtual]
```

Reimplemented from IOBuffer.

4.3.2.10 Unpack()

```
int FixedFieldBuffer::Unpack (
            void * field,
            int maxBytes = -1 )  [virtual]
```

Implements IOBuffer.

4.3.2.11 WriteHeader()

```
int FixedFieldBuffer::WriteHeader (
            ostream & stream ) const  [virtual]
```

Reimplemented from IOBuffer.

4.3.3 Member Data Documentation

4.3.3.1 FieldSize

```
int* FixedFieldBuffer::FieldSize  [protected]
```

4.3.3.2 MaxFields

```
int FixedFieldBuffer::MaxFields  [protected]
```

4.3.3.3 NextField

```
int FixedFieldBuffer::NextField  [protected]
```

4.3.3.4 NumFields

```
int FixedFieldBuffer::NumFields  [protected]
```

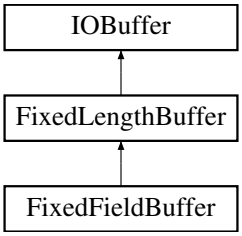
The documentation for this class was generated from the following files:

- C:/Users/silas/CLionProjects/project3_khalil/fixfld.h
- C:/Users/silas/CLionProjects/project3_khalil/fixfld.cpp

4.4 FixedLengthBuffer Class Reference

```
#include <fixlen.h>
```

Inheritance diagram for FixedLengthBuffer:



Public Member Functions

- FixedLengthBuffer (int recordSize=1000)
- FixedLengthBuffer (const FixedLengthBuffer &buffer)
- void Clear ()
- int Read (istream &)
- int Write (ostream &) const
- int ReadHeader (istream &)
- int WriteHeader (ostream &) const
- void Print (ostream &) const
- int SizeOfBuffer () const

Public Member Functions inherited from IOBuffer

- IOBuffer (int maxBytes=1000)
- IOBuffer & operator= (const IOBuffer &)
- virtual int Pack (const void *field, int size=-1)=0
- virtual int Unpack (void *field, int maxbytes=-1)=0
- int Init (int maxBytes)
- virtual int DRead (istream &, int recref)
- virtual int DWrite (ostream &, int recref) const

Protected Member Functions

- int Init (int recordSize)
- int ChangeRecordSize (int recordSize)

Additional Inherited Members

Protected Attributes inherited from IOBuffer

- int Initialized
- char * Buffer
- int BufferSize
- int MaxBytes
- int NextByte
- int Packing

4.4.1 Constructor & Destructor Documentation

4.4.1.1 FixedLengthBuffer() [1/2]

```
FixedLengthBuffer::FixedLengthBuffer (
            int recordSize = 1000 )
```

4.4.1.2 FixedLengthBuffer() [2/2]

```
FixedLengthBuffer::FixedLengthBuffer (
            const FixedLengthBuffer & buffer )  [inline]
```

4.4.2 Member Function Documentation

4.4.2.1 ChangeRecordSize()

```
int FixedLengthBuffer::ChangeRecordSize (
            int recordSize )  [protected]
```

4.4.2.2 Clear()

```
void FixedLengthBuffer::Clear ( )  [virtual]
```

Reimplemented from IOBuffer.

4.4.2.3 Init()

```
int FixedLengthBuffer::Init (
            int recordSize )  [protected]
```

4.4.2.4 Print()

```
void FixedLengthBuffer::Print (
            ostream & stream ) const  [virtual]
```

Reimplemented from IOBuffer.

4.4.2.5 Read()

```
int FixedLengthBuffer::Read (
            istream & stream )  [virtual]
```

Implements IOBuffer.

4.4.2.6 ReadHeader()

```
int FixedLengthBuffer::ReadHeader (
            istream & stream )  [virtual]
```

Reimplemented from IOBuffer.

4.4.2.7 SizeOfBuffer()

```
int FixedLengthBuffer::SizeOfBuffer ( ) const
```

4.4.2.8 Write()

```
int FixedLengthBuffer::Write (
            ostream & stream ) const  [virtual]
```

Implements IOBuffer.

4.4.2.9 WriteHeader()

```
int FixedLengthBuffer::WriteHeader (
            ostream & stream ) const  [virtual]
```

Reimplemented from IOBuffer.

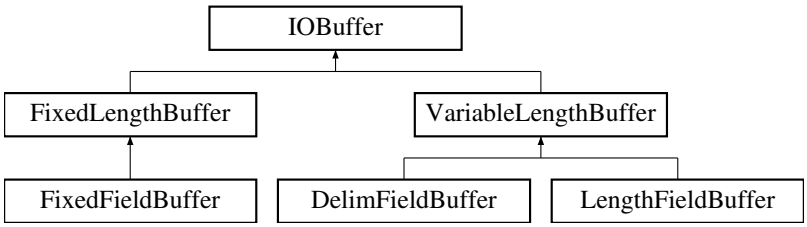
The documentation for this class was generated from the following files:

- C:/Users/silas/CLionProjects/project3_khalil/fixlen.h
- C:/Users/silas/CLionProjects/project3_khalil/fixlen.cpp

4.5 IOBuffer Class Reference

```
#include <iobuffer.h>
```

Inheritance diagram for IOBuffer:



Public Member Functions

- IOBuffer (int maxBytes=1000)
- IOBuffer & operator= (const IOBuffer &)
- virtual void Clear ()
- virtual int Pack (const void *field, int size=-1)=0
- virtual int Unpack (void *field, int maxbytes=-1)=0
- virtual void Print (ostream &) const
- int Init (int maxBytes)
- virtual int Read (istream &)=0
- virtual int Write (ostream &) const =0
- virtual int DRead (istream &, int recref)
- virtual int DWrite (ostream &, int recref) const
- virtual int ReadHeader (istream &)
- virtual int WriteHeader (ostream &) const

Protected Attributes

- int Initialized
- char * Buffer
- int BufferSize
- int MaxBytes
- int NextByte
- int Packing

4.5.1 Constructor & Destructor Documentation

4.5.1.1 IOBuffer()

```
IOBuffer::IOBuffer (
            int maxBytes = 1000 )
```

4.5.2 Member Function Documentation

4.5.2.1 Clear()

```
void IOBuffer::Clear ( )  [virtual]
```

Reimplemented in DelimFieldBuffer, FixedFieldBuffer, FixedLengthBuffer, LengthFieldBuffer, and VariableLengthBuffer.

4.5.2.2 DRead()

```
int IOBuffer::DRead (
            istream & stream,
            int recref )  [virtual]
```

4.5.2.3 DWrite()

```
int IOBuffer::DWrite (
            ostream & stream,
            int recref ) const  [virtual]
```

4.5.2.4 Init()

```
int IOBuffer::Init (
            int maxBytes )
```

4.5.2.5 operator=()

```
IOBuffer & IOBuffer::operator= (
            const IOBuffer & buffer )
```

4.5.2.6 Pack()

```
virtual int IOBuffer::Pack (
            const void * field,
            int size = -1 )  [pure virtual]
```

Implemented in DelimFieldBuffer, FixedFieldBuffer, and LengthFieldBuffer.

4.5.2.7 Print()

```
void IOBuffer::Print (
            ostream & stream ) const  [virtual]
```

Reimplemented in DelimFieldBuffer, FixedFieldBuffer, FixedLengthBuffer, LengthFieldBuffer, and VariableLengthBuffer.

4.5.2.8 Read()

```
virtual int IOBuffer::Read (
            istream &  )  [pure virtual]
```

Implemented in FixedLengthBuffer, and VariableLengthBuffer.

4.5.2.9 ReadHeader()

```
int IOBuffer::ReadHeader (
            istream & stream )  [virtual]
```

Reimplemented in FixedFieldBuffer, FixedLengthBuffer, VariableLengthBuffer, and DelimFieldBuffer.

4.5.2.10 Unpack()

```
virtual int IOBuffer::Unpack (
            void * field,
            int maxbytes = -1 )  [pure virtual]
```

Implemented in DelimFieldBuffer, FixedFieldBuffer, and LengthFieldBuffer.

4.5.2.11 Write()

```
virtual int IOBuffer::Write (
            ostream &  ) const  [pure virtual]
```

Implemented in FixedLengthBuffer, and VariableLengthBuffer.

4.5.2.12 WriteHeader()

```
int IOBuffer::WriteHeader (
            ostream & stream ) const  [virtual]
```

Reimplemented in FixedFieldBuffer, FixedLengthBuffer, VariableLengthBuffer, and DelimFieldBuffer.

4.5.3 Member Data Documentation

4.5.3.1 Buffer

```
char* IOBuffer::Buffer  [protected]
```

4.5.3.2 BufferSize

```
int IOBuffer::BufferSize  [protected]
```

4.5.3.3 Initialized

```
int IOBuffer::Initialized  [protected]
```

4.5.3.4 MaxBytes

```
int IOBuffer::MaxBytes  [protected]
```

4.5.3.5 NextByte

```
int IOBuffer::NextByte  [protected]
```

4.5.3.6 Packing

```
int IOBuffer::Packing  [protected]
```

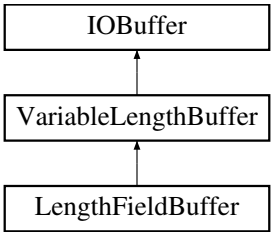
The documentation for this class was generated from the following files:

- C:/Users/silas/CLionProjects/project3_khalil/iobuffer.h
- C:/Users/silas/CLionProjects/project3_khalil/iobuffer.cpp

4.6 LengthFieldBuffer Class Reference

```
#include <length.h>
```

Inheritance diagram for LengthFieldBuffer:



Public Member Functions

- LengthFieldBuffer (int maxBytes=1000)
- LengthFieldBuffer (const LengthFieldBuffer &buffer)
- void Clear ()
- int Pack (const void *field, int size=-1)
- int Unpack (void *field, int maxBytes=-1)
- void Print (ostream &) const
- int Init ()

Public Member Functions inherited from VariableLengthBuffer

- VariableLengthBuffer (int MaxBytes=1000)
- VariableLengthBuffer (const VariableLengthBuffer &buffer)
- void Clear ()
- int Read (istream &)
- int Write (ostream &) const
- int ReadHeader (istream &)
- int WriteHeader (ostream &) const
- int PackFixLen (void *, int)
- int PackDelimeted (void *, int)
- int PackLength (void *, int)
- void Print (ostream &) const
- int SizeOfBuffer () const
- int Init ()

Public Member Functions inherited from IOBuffer

- IOBuffer (int maxBytes=1000)
- IOBuffer & operator= (const IOBuffer &)
- int Init (int maxBytes)
- virtual int DRead (istream &, int recref)
- virtual int DWrite (ostream &, int recref) const

Additional Inherited Members

Protected Attributes inherited from IOBuffer

- int Initialized
- char * Buffer
- int BufferSize
- int MaxBytes
- int NextByte
- int Packing

4.6.1 Constructor & Destructor Documentation

4.6.1.1 LengthFieldBuffer() [1/2]

```
LengthFieldBuffer::LengthFieldBuffer (
            int maxBytes = 1000 )
```

4.6.1.2 LengthFieldBuffer() [2/2]

```
LengthFieldBuffer::LengthFieldBuffer (
            const LengthFieldBuffer & buffer )  [inline]
```

4.6.2 Member Function Documentation

4.6.2.1 Clear()

```
void LengthFieldBuffer::Clear ( )  [virtual]
```

Reimplemented from IOBuffer.

4.6.2.2 Init()

```
int LengthFieldBuffer::Init ( )
```

4.6.2.3 Pack()

```
int LengthFieldBuffer::Pack (
            const void * field,
            int size = -1 )  [virtual]
```

Implements IOBuffer.

4.6.2.4 Print()

```
void LengthFieldBuffer::Print (
            ostream & stream ) const  [virtual]
```

Reimplemented from IOBuffer.

4.6.2.5 Unpack()

```
int LengthFieldBuffer::Unpack (
            void * field,
            int maxBytes = -1 )  [virtual]
```

Implements IOBuffer.

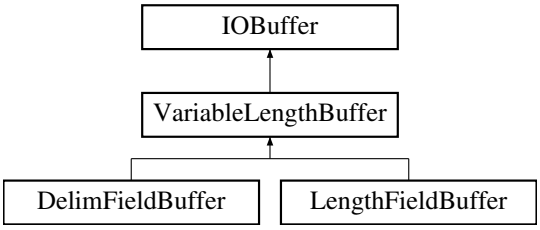
The documentation for this class was generated from the following files:

- C:/Users/silas/CLionProjects/project3_khalil/length.h
- C:/Users/silas/CLionProjects/project3_khalil/length.cpp

4.7 VariableLengthBuffer Class Reference

`#include <varlen.h>`

Inheritance diagram for VariableLengthBuffer:



Public Member Functions

- VariableLengthBuffer (int MaxBytes=1000)
- VariableLengthBuffer (const VariableLengthBuffer &buffer)
- void Clear ()
- int Read (istream &)
- int Write (ostream &) const
- int ReadHeader (istream &)
- int WriteHeader (ostream &) const
- int PackFixLen (void *, int)
- int PackDelimeted (void *, int)
- int PackLength (void *, int)
- void Print (ostream &) const
- int SizeOfBuffer () const
- int Init ()

Public Member Functions inherited from IOBuffer

- IOBuffer (int maxBytes=1000)
- IOBuffer & operator= (const IOBuffer &)
- virtual int Pack (const void *field, int size=-1)=0
- virtual int Unpack (void *field, int maxbytes=-1)=0
- int Init (int maxBytes)
- virtual int DRead (istream &, int recref)
- virtual int DWrite (ostream &, int recref) const

Additional Inherited Members

Protected Attributes inherited from IOBuffer

- int Initialized
- char * Buffer
- int BufferSize
- int MaxBytes
- int NextByte
- int Packing

4.7.1 Constructor & Destructor Documentation

4.7.1.1 VariableLengthBuffer() [1/2]

```
VariableLengthBuffer::VariableLengthBuffer (
            int MaxBytes = 1000 )
```

4.7.1.2 VariableLengthBuffer() [2/2]

```
VariableLengthBuffer::VariableLengthBuffer (
            const VariableLengthBuffer & buffer )  [inline]
```

4.7.2 Member Function Documentation

4.7.2.1 Clear()

```
void VariableLengthBuffer::Clear ( )  [virtual]
```

Reimplemented from IOBuffer.

4.7.2.2 Init()

```
int VariableLengthBuffer::Init ( )
```

4.7.2.3 PackDelimeted()

```
int VariableLengthBuffer::PackDelimeted (
            void * ,
            int  )
```

4.7.2.4 PackFixLen()

```
int VariableLengthBuffer::PackFixLen (
            void * ,
            int  )
```

4.7.2.5 PackLength()

```
int VariableLengthBuffer::PackLength (
            void * ,
            int  )
```

4.7.2.6 Print()

```
void VariableLengthBuffer::Print (
            ostream & stream ) const  [virtual]
```

Reimplemented from IOBuffer.

4.7.2.7 Read()

```
int VariableLengthBuffer::Read (
            istream & stream )  [virtual]
```

Implements IOBuffer.

4.7.2.8 ReadHeader()

```
int VariableLengthBuffer::ReadHeader (
            istream & stream )  [virtual]
```

Reimplemented from IOBuffer.

4.7.2.9 SizeOfBuffer()

```
int VariableLengthBuffer::SizeOfBuffer ( ) const
```

4.7.2.10 Write()

```
int VariableLengthBuffer::Write (
            ostream & stream ) const  [virtual]
```

Implements IOBuffer.

4.7.2.11 WriteHeader()

```
int VariableLengthBuffer::WriteHeader (
            ostream & stream ) const  [virtual]
```

Reimplemented from IOBuffer.

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

- C:/Users/silas/CLionProjects/project3_khalil/varlen.h
- C:/Users/silas/CLionProjects/project3_khalil/varlen.cpp

4.8 Zipcode Class Reference

Zipcode Information.

`#include <Zipcode.h>`

Public Member Functions

- Zipcode ()
  Default constructor.
- const char * getZip () const
  Gets the Zipcode identifier.
- const char * getPlace () const
  Getter methods of the Zipcode.
- const char * getState () const
- const char * getCounty () const
- const char * getLatitude () const
- const char * getLongitude () const
- void setZip (const char *z)
  Setter methods of the Zipcode.
- void setPlace (const char *p)
- void setState (const char *s)
- void setCounty (const char *c)
- void setLatitude (const char *lat)
- void setLongitude (const char *lon)
- void Clear ()
  Clears all data members of the Zipcode object.
- int Unpack (IOBuffer &)
  Unpacks the Zipcode object.
- int Pack (DelimFieldBuffer) const
  Packs the Zipcode object.
- void Print (ostream &, char *label=0) const
  Prints the Zipcode information to the output stream.

Static Public Member Functions

- static int InitBuffer (DelimFieldBuffer &)
  DelimFieldBuffer.
- static int InitBuffer (LengthFieldBuffer &)
  LengthFieldBuffer.
- static int InitBuffer (FixedFieldBuffer &)
  FixedFieldBuffer.

Public Attributes

- char zip [6]
- char place [24]
- char state [3]
- char county [16]
- char latitude [10]
- char longitude [10]

4.8.1 Detailed Description

Zipcode Information.

4.8.2 Constructor & Destructor Documentation

4.8.2.1 Zipcode()

```
Zipcode::Zipcode ( )
```

Default constructor.

Default constructor for the Zipcode class. Initializes the Zipcode object by calling the Clear() method.

4.8.3 Member Function Documentation

4.8.3.1 Clear()

```
void Zipcode::Clear ( )
```

Clears all data members of the Zipcode object.

Clears all fields of the Zipcode object.

4.8.3.2 getCounty()

```
const char * Zipcode::getCounty ( ) const  [inline]
```

4.8.3.3 getLatitude()

```
const char * Zipcode::getLatitude ( ) const  [inline]
```

4.8.3.4 getLongitude()

```
const char * Zipcode::getLongitude ( ) const  [inline]
```

4.8.3.5 getPlace()

```
const char * Zipcode::getPlace ( ) const  [inline]
```

Getter methods of the Zipcode.

Returns

Info about the Zipcode.

4.8.3.6 getState()

```
const char * Zipcode::getState ( ) const  [inline]
```

4.8.3.7 getZip()

```
const char * Zipcode::getZip ( ) const  [inline]
```

Gets the Zipcode identifier.

Returns

The Zipcode identifier.

4.8.3.8 InitBuffer() [1/3]

```
int Zipcode::InitBuffer (
            DelimFieldBuffer & Buffer )  [static]
```

DelimFieldBuffer.

Initializes a DelimFieldBuffer to be used for packing Zipcode objects.

Parameters

Buffer	The DelimFieldBuffer to initialize.

Returns

1 if successful, 0 otherwise.

Parameters

Buffer	The DelimFieldBuffer to initialize.

Returns

TRUE if initialization succeeds, FALSE otherwise.

4.8.3.9 InitBuffer() [2/3]

```
int Zipcode::InitBuffer (
            FixedFieldBuffer & Buffer )  [static]
```

FixedFieldBuffer.

Initializes a FixedFieldBuffer to be used for packing Zipcode objects.

Parameters

Buffer	The FixedFieldBuffer to initialize.

Returns

1 if successful, 0 otherwise.

Parameters

Buffer	The FixedFieldBuffer to initialize.

Returns

TRUE if initialization succeeds, FALSE otherwise.

4.8.3.10 InitBuffer() [3/3]

```
int Zipcode::InitBuffer (
            LengthFieldBuffer & Buffer )  [static]
```

LengthFieldBuffer.

Initializes a LengthFieldBuffer to be used for packing Zipcode objects.

Parameters

Buffer	The LengthFieldBuffer to initialize.

Returns

1 if successful, 0 otherwise.

Parameters

Buffer	The LengthFieldBuffer to initialize.

Returns

TRUE if initialization succeeds, FALSE otherwise.

4.8.3.11 Pack()

```
int Zipcode::Pack (
            DelimFieldBuffer Buffer ) const
```

Packs the Zipcode object.

Packs the fields of the Zipcode object into the provided buffer.

Parameters

Buffer	IOBuffer containing the packed Zipcode object.

Returns

1 if successful, 0 otherwise.

Parameters

Buffer	The IOBuffer to pack the fields into.

Returns

TRUE if packing succeeds, FALSE otherwise.

4.8.3.12 Print()

```
void Zipcode::Print (
            ostream & stream,
            char * label = 0 ) const
```

Prints the Zipcode information to the output stream.

Prints the zipcode information to the specified output stream.

Parameters

stream	Output stream to which the Zipcode information will be printed.
label.	
stream	The output stream to which the zipcode information will be printed.
label	Optional label to prepend to the output. Defaults to nullptr.

4.8.3.13 setCounty()

```
void Zipcode::setCounty (
            const char * c )  [inline]
```

4.8.3.14 setLatitude()

```
void Zipcode::setLatitude (
            const char * lat )  [inline]
```

4.8.3.15 setLongitude()

```
void Zipcode::setLongitude (
            const char * lon )  [inline]
```

4.8.3.16 setPlace()

```
void Zipcode::setPlace (
            const char * p )  [inline]
```

4.8.3.17 setState()

```
void Zipcode::setState (
            const char * s )  [inline]
```

4.8.3.18 setZip()

```
void Zipcode::setZip (
            const char * z )  [inline]
```

Setter methods of the Zipcode.

Parameters

item	it is setting.

4.8.3.19 Unpack()

```
int Zipcode::Unpack (
            IOBuffer & Buffer )
```

Unpacks the Zipcode object.

Unpacks the fields of the Zipcode object from the provided buffer.

Parameters

Buffer	IOBuffer containing the packed Zipcode object.

Returns

1 if successful, 0 otherwise.

Parameters

Buffer	The IOBuffer to unpack the fields from.

Returns

TRUE if unpacking succeeds, FALSE otherwise.

4.8.4 Member Data Documentation

4.8.4.1 county

```
char Zipcode::county[16]
```

Zipcode address

4.8.4.2 latitude

```
char Zipcode::latitude[10]
```

Zipcode enrollment date

4.8.4.3 longitude

```
char Zipcode::longitude[10]
```

Zipcode enrollment date

4.8.4.4 place

```
char Zipcode::place[24]
```

Zipcode first name

4.8.4.5 state

```
char Zipcode::state[3]
```

Zipcode last name

4.8.4.6 zip

```
char Zipcode::zip[6]
```

Zipcode identifier

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

- C:/Users/silas/CLionProjects/project3_khalil/Zipcode.h
- C:/Users/silas/CLionProjects/project3_khalil/Zipcode.cpp

Chapter 5

File Documentation

5.1 C:/Users/silas/CLionProjects/project3_khalil/buffile.cpp File Reference

```
#include "buffile.h"
```

5.2 C:/Users/silas/CLionProjects/project3_khalil/buffile.h File Reference

```
#include <stdlib.h>
#include <fstream>
#include "iobuffer.h"
```

Classes

- class BufferFile

5.3 buffile.h

Go to the documentation of this file.

```
00001 // buffile.h
00002
00003 #ifndef BUFFILE_H
00004 #define BUFFILE_H
00005
00006 #include <stdlib.h>
00007 #include <fstream>
00008 #include "iobuffer.h"
00009
00010
00011 #ifndef FALSE
00012 #define FALSE (0)
00013 #define TRUE (1)
00014 #endif
00015
00016 class BufferFile
00017 // Class to represent buffered file operations
00018 //  Used in conjunction with the IOBuffer classes
00019 // Each buffered file is associated with a disk file of a specific
```

```
00020 //  record type.
00021 // Each buffered file object has a buffer object which can be used
00022 //  for file I/O
00023 // Sequential and random access read and write are supported
00024 //  each write returns the record address of the record
00025 //  this record address can be used to read that record
00026 //  the values of the record address depend on the type of file and buffer
00027 {
00028    public:
00029     BufferFile (IOBuffer &); // create with a buffer
00030
00031     int Open (const char * filename, std::ios_base::openmode MODE); // open an existing file
00032     int Create (const char * filename, std::ios_base::openmode MODE); // create a new file
00033     int Close ();
00034     int Rewind (); // reset to the first data record
00035     // Input and Output operations
00036     int Read (int recaddr = -1);
00037         // read a record into the buffer
00038         // return the record address
00039         // return <0 if read failed
00040         // if recaddr == -1, read the next record in the file
00041         // if recaddr != -1, read the record at that address
00042     int Write (int recaddr = -1); // write the current buffer contents
00043     int Append (); // write the current buffer at the end of file
00044
00045     // Access to IOBuffer
00046     IOBuffer & GetBuffer ();
00047
00048    protected:
00049     IOBuffer & Buffer;
00050     fstream File;
00051     int HeaderSize; // size of header
00052     int ReadHeader ();
00053     int WriteHeader ();
00054 };
00055
00056 #endif
00057
```

5.4 C:/Users/silas/CLionProjects/project3_khalil/delim.cpp File Reference

```
#include "delim.h"
#include <string.h>
```

5.5 C:/Users/silas/CLionProjects/project3_khalil/delim.h File Reference

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

Classes

- class DelimFieldBuffer

5.6 delim.h

Go to the documentation of this file.

```
00001 // delim.h
00002 #ifndef DELIM_H
00003 #define DELIM_H
```

```
00004
00005 #include <iostream>
00006 #include "varlen.h"
00007
00008 using namespace std;
00009
00010 class DelimFieldBuffer: public VariableLengthBuffer
00011 // a buffer which holds delimited text fields.
00012 // Record variables can be packed into and extracted from a buffer.
00013 // Input and Output of packed buffers
00014 //
00015 // To use this class, create a DelimFieldBuffer variable and associate definitions with the fields.
00016 // operations are provided to allow values to be associated with the fields (Pack)
00017 //  and to fetch the values of fields (Unpack)
00018 {  public:
00019     DelimFieldBuffer (char Delim = -1, int maxBytes = 1000); // construct with a maximum of maxBytes
00020         // construct with fields with delimeters
00021     DelimFieldBuffer (const DelimFieldBuffer & buffer); // copy constructor
00022
00023     void Clear (); // clear fields from buffer
00024     int Pack (const void*, int size = -1); // set the value of the next field of the buffer;
00025     int Unpack (void * field, int maxBytes = -1); // extract the value of the next field of the buffer
00026     int ReadHeader (istream & stream);
00027     int WriteHeader (ostream & stream) const;
00028     void Print (ostream &) const;
00029     int Init (char delim = 0);
00030     static void SetDefaultDelim (char delim);
00031  protected:
00032     char Delim;
00033     static char DefaultDelim;
00034 };
00035
00036
00037 inline DelimFieldBuffer :: DelimFieldBuffer
00038     (const DelimFieldBuffer & buffer) // copy constructor
00039         : VariableLengthBuffer (buffer)
00040 {
00041     Init (buffer . Delim);
00042 }
00043
00044 //#include "delim.cpp"
00045 #endif
00046
```

5.7 C:/Users/silas/CLionProjects/project3_khalil/fixfld.cpp File Reference

```
#include "fixfld.h"
#include "length.h"
#include <string.h>
```

5.8 C:/Users/silas/CLionProjects/project3_khalil/fixfld.h File Reference

```
#include <stdlib.h>
#include <iostream>
#include "fixlen.h"
```

Classes

- class FixedFieldBuffer

5.9 fixfld.h

Go to the documentation of this file.

```
00001 // fixfld.h
00002 #ifndef FIXFLD_H
00003 #define FIXFLD_H
00004
00005 #include <stdlib.h>
00006 #include <iostream>
00007 #include "fixlen.h"
00008
00009 using namespace std;
00010
00011 class FixedFieldBuffer: public FixedLengthBuffer
00012 // Abstract class designed to support fixed length records
00013 // Use of this class requires that all fields be defined before
00014 //    reading and writing can take place
00015 {
00016    public:
00017     FixedFieldBuffer (int maxFields, int RecordSize = 1000);
00018     FixedFieldBuffer (int maxFields, int * fieldSize);
00019     // initialize all fields at once
00020     FixedFieldBuffer (const FixedFieldBuffer &); //copy constructor
00021     FixedFieldBuffer & operator = (const FixedFieldBuffer &);
00022     void Clear (); // clear values from buffer
00023     int AddField (int fieldSize); // define the next field
00024     int ReadHeader (istream &); // write a buffer to the stream
00025     int WriteHeader (ostream &) const; // write a buffer to the stream
00026     int Pack (const void * field, int size = -1); // set the value of the next field of the buffer;
00027     int Unpack (void * field, int maxBytes = -1); // extract the value of the next field of the buffer
00028     void Print (ostream &) const;
00029     int NumberOfFields () const; // return number of defined fields
00030     int Init (int maxFields);
00031     int Init (int numFields, int * fieldSize);
00032  protected:
00033     int * FieldSize; // array to hold field sizes
00034     int MaxFields; // maximum number of fields
00035     int NumFields; // actual number of defined fields
00036     int NextField; // index of next field to be packed/unpacked
00037 };
00038
00039 inline FixedFieldBuffer :: FixedFieldBuffer (const FixedFieldBuffer & buffer)
00040     : FixedLengthBuffer (buffer)
00041 {
00042     Init (buffer . NumFields, buffer . FieldSize);
00043 }
00044 //#include "fixfld.cpp"
00045 #endif
00046
```

5.10 C:/Users/silas/CLionProjects/project3_khalil/fixlen.cpp File Reference

```
#include "fixlen.h"
#include "length.h"
#include <string.h>
```

5.11 C:/Users/silas/CLionProjects/project3_khalil/fixlen.h File Reference

```
#include <stdlib.h>
#include <iostream>
#include "iobuffer.h"
```

Classes

- class FixedLengthBuffer

5.12 fixlen.h

Go to the documentation of this file.

```
00001 // fixlen.h
00002 #ifndef FIXLEN_H
00003 #define FIXLEN_H
00004
00005 #include <stdlib.h>
00006 #include <iostream>
00007 #include "iobuffer.h"
00008 using namespace std;
00009
00010 class FixedLengthBuffer: public IOBuffer
00011 // Abstract class designed to support fixed length records
00012 {
00013    public:
00014     FixedLengthBuffer (int recordSize = 1000);
00015     FixedLengthBuffer (const FixedLengthBuffer & buffer); // copy constructor
00016
00017     void Clear (); // clear values from buffer
00018     int Read (istream &);
00019     int Write (ostream &) const;
00020     int ReadHeader (istream &); // read header from stream
00021     int WriteHeader (ostream &) const; // write a header to the stream
00022     void Print (ostream &) const;
00023     int SizeOfBuffer () const; // return size of buffer
00024  protected:
00025     int Init (int recordSize);
00026     int ChangeRecordSize (int recordSize);
00027 };
00028
00029 inline FixedLengthBuffer :: FixedLengthBuffer (const FixedLengthBuffer & buffer)
00030 : IOBuffer (buffer)
00031 {
00032     Init (buffer . BufferSize);
00033 }
00034
00035 //#include "fixlen.cpp"
00036 #endif
```

5.13 C:/Users/silas/CLionProjects/project3_khalil/iobuffer.cpp File Reference

```
#include "iobuffer.h"
#include <string.h>
```

5.14 C:/Users/silas/CLionProjects/project3_khalil/iobuffer.h File Reference

```
#include <cstdlib>
#include <iostream>
```

Classes

- class IOBuffer

Macros

- #define FALSE (0)
- #define TRUE (1)

Functions

- int PackFixed (char *buffer, void *field, int size=-1)
- int PackDelimeted (char *buffer, void *field, int size=-1)
- int PackLength (char *buffer, void *field, int size=-1)

5.14.1 Macro Definition Documentation

5.14.1.1 FALSE

```
#define FALSE (0)
```

5.14.1.2 TRUE

```
#define TRUE (1)
```

5.14.2 Function Documentation

5.14.2.1 PackDelimeted()

```
int PackDelimeted (
            char * buffer,
            void * field,
            int size = -1 )
```

5.14.2.2 PackFixed()

```
int PackFixed (
            char * buffer,
            void * field,
            int size = -1 )
```

5.14.2.3 PackLength()

```
int PackLength (
            char * buffer,
            void * field,
            int size = -1 )
```

5.15 iobuffer.h

Go to the documentation of this file.

```
00001 // iobuffer.h
00002 #ifndef IOBUFFER_H
00003 #define IOBUFFER_H
00004 //#include <stdlib.h>
00005 #include <cstdlib>
00006 #include <iostream>
00007
00008 using namespace std;
00009
00010 #ifndef FALSE
00011 #define FALSE (0)
00012 #define TRUE (1)
00013 #endif
00014
00015 class IOBuffer
00016 // An abstract base class for file buffers
00017 // Record variables can be packed into and extracted from a buffer.
00018 // Input and Output of packed buffers
00019 // When each field has a value, the buffer can be written into an ostream.
00020 //
00021 // operations are provided to allow values to be associated with the fields (Pack)
00022 //  and to fetch the values of fields (Unpack)
00023 //
00024 {  public:
00025     IOBuffer (int maxBytes = 1000); // a maximum of maxBytes
00026     IOBuffer & operator = (const IOBuffer &);
00027     virtual void Clear (); // clear fields from buffer
00028     virtual int Pack (const void * field, int size = -1) = 0; // set the value of the next field of
      the buffer;
00029     virtual int Unpack (void * field, int maxbytes = -1) = 0; // extract the value of the next field
      of the buffer
00030     virtual void Print (ostream &) const;
00031     int Init (int maxBytes);
00032     // the read and write operations return the address of the record
00033     // sequential read and write operations
00034     virtual int Read (istream &) = 0; // read a buffer from the stream
00035     virtual int Write (ostream &) const = 0; // write a buffer to the stream
00036
00037     // these are the direct access read and write operations
00038     virtual int DRead (istream &, int recref); // read specified record
00039     virtual int DWrite (ostream &, int recref) const; // write specified record
00040
00041     // these header operations return the number of bytes in the header
00042     virtual int ReadHeader (istream &);
00043             // read from the stream and write to a buffer
00044     virtual int WriteHeader (ostream &) const; // write a buffer to the stream
00045
00046  protected:
00047     int Initialized; // TRUE if buffer is initialized
00048     char * Buffer; // character array to hold field values
00049     int BufferSize; // sum of the sizes of packed fields
00050     int MaxBytes; // maximum number of characters in the buffer
00051     int NextByte; // index of next byte to be packed/unpacked
00052     int Packing; // TRUE if in packing mode, FALSE, if unpacking
00053 };
00054
00055 // field packing operations
00056 // pack a field into a buffer
00057 int PackFixed (char * buffer, void * field, int size = -1);
00058 int PackDelimeted (char *buffer, void * field, int size = -1);
00059 int PackLength (char * buffer, void * field, int size = -1);
00060
00061 //#include "iobuffer.cpp"
00062 #endif
```

5.16 C:/Users/silas/CLionProjects/project3_khalil/length.cpp File Reference

```
#include "length.h"
#include <string.h>
```

5.17 C:/Users/silas/CLionProjects/project3_khalil/length.h File Reference

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

Classes

- class LengthFieldBuffer

5.18 length.h

Go to the documentation of this file.

```
00001 // length.h
00002
00003 #ifndef LENGTH_H
00004 #define LENGTH_H
00005
00006 #include <iostream>
00007 #include "varlen.h"
00008
00009 using namespace std;
00010
00011 class LengthFieldBuffer: public VariableLengthBuffer
00012 // a buffer which holds a length plus value fields.
00013 // Record variables can be packed into and extracted from a buffer.
00014 //
00015 // To use this class, create a LengthFieldBuffer variable and associate
00016 // definitions with the fields.
00017 // operations are provided to allow values to be associated with the
00018 // fields (Pack) and to fetch the values of fields (Unpack)
00019 {  public:
00020     LengthFieldBuffer (int maxBytes = 1000); // construct with a maximum of maxFields
00021         // construct with fields of specific size
00022     LengthFieldBuffer (const LengthFieldBuffer & buffer) // copy constructor
00023     : VariableLengthBuffer (buffer) {}
00024     void Clear (); // clear fields from buffer
00025     int Pack (const void * field, int size = -1); // set the value of the next field of the buffer;
00026     int Unpack (void *field, int maxBytes = -1); // extract the value of the next field of the buffer
00027     void Print (ostream &) const;
00028     int Init ();
00029   protected:
00030 };
00031
00032 #endif
```

5.19 C:/Users/silas/CLionProjects/project3_khalil/testPlace.cpp File Reference

```
#include <iostream>
#include <fstream>
#include <sstream>
#include <iomanip>
#include <algorithm>
#include <map>
#include <limits>
#include <vector>
#include "Zipcode.h"
```

Functions

- vector< Zipcode > readDataFromFile (const string &filename)

  Reads data from a CSV file containing zip code information.
- bool compareByState (const Zipcode &a, const Zipcode &b)

  Comparator function to sort Zipcode objects by state.
- map< string, pair< pair< string, string >, pair< string, string > > > findExtremeCoordinates (const vector< Zipcode > &zipcodes)

  Finds the extreme coordinates (longitude and latitude) for each state.
- int main ()

  Main function to read data from file, find extreme coordinates, and output in table format.

5.19.1 Function Documentation

5.19.1.1 compareByState()

```
bool compareByState (
            const Zipcode & a,
            const Zipcode & b )
```

Comparator function to sort Zipcode objects by state.

Parameters

a	First Zipcode object to compare.
b	Second Zipcode object to compare.

Returns

bool True if the state of 'a' is less than the state of 'b', false otherwise.

5.19.1.2 findExtremeCoordinates()

```
map< string, pair< pair< string, string >, pair< string, string > > > findExtremeCoordinates
(
            const vector< Zipcode > & zipcodes )
```

Finds the extreme coordinates (longitude and latitude) for each state.

Parameters

zipcodes	Vector of Zipcode objects.

Returns

map<string, pair<pair<string, string>, pair<string, string> > > Map containing extreme coordinates for each state.

5.19.1.3 main()

```
int main ( )
```

Main function to read data from file, find extreme coordinates, and output in table format.

Returns

int Exit status.

5.19.1.4 readDataFromFile()

```
vector< Zipcode > readDataFromFile (
            const string & filename )
```

Reads data from a CSV file containing zip code information.

Parameters

filename	The name of the file to read.

Returns

vector<Zipcode> A vector containing Zipcode objects read from the file.

5.20 C:/Users/silas/CLionProjects/project3_khalil/testZip.cpp File Reference

```
#include <iostream>
#include <fstream>
#include <sstream>
#include <iomanip>
#include <algorithm>
#include <map>
#include <limits>
#include <vector>
#include "Zipcode.h"
```

Functions

- vector< Zipcode > readDataFromFile (const string &filename)

  Reads data from a CSV file containing zip code information.
- bool compareByState (const Zipcode &a, const Zipcode &b)

  Comparator function to sort Zipcode objects by state.
- map< string, pair< pair< string, string >, pair< string, string > > > findExtremeCoordinates (const vector< Zipcode > &zipcodes)

  Finds the extreme coordinates (longitude and latitude) for each state.
- int main ()

  Main function to read data from file, find extreme coordinates, and output in table format.

5.20.1 Function Documentation

5.20.1.1 compareByState()

```
bool compareByState (
            const Zipcode & a,
            const Zipcode & b )
```

Comparator function to sort Zipcode objects by state.

Parameters

a	First Zipcode object to compare.
b	Second Zipcode object to compare.

Returns

bool True if the state of 'a' is less than the state of 'b', false otherwise.

5.20.1.2 findExtremeCoordinates()

```
map< string, pair< pair< string, string >, pair< string, string > > > findExtremeCoordinates
(
            const vector< Zipcode > & zipcodes )
```

Finds the extreme coordinates (longitude and latitude) for each state.

Parameters

zipcodes	Vector of Zipcode objects.

Returns

map<string, pair<pair<string, string>, pair<string, string> > > Map containing extreme coordinates for each state.

5.20.1.3 main()

```
int main ( )
```

Main function to read data from file, find extreme coordinates, and output in table format.

Returns

int Exit status.

5.20.1.4 readDataFromFile()

```
vector< Zipcode > readDataFromFile (
            const string & filename )
```

Reads data from a CSV file containing zip code information.

Parameters

filename	The name of the file to read.

Returns

vector<Zipcode> A vector containing Zipcode objects read from the file.

5.21 C:/Users/silas/CLionProjects/project3_khalil/varlen.cpp File Reference

```
#include "varlen.h"
#include <string.h>
```

Variables

- const char * headerStrV = "Variable"
- const int headerSizeV = strlen (headerStrV)

5.21.1 Variable Documentation

5.21.1.1 headerSizeV

```
const int headerSizeV = strlen (headerStrV)
```

5.21.1.2 headerStrV

```
const char* headerStrV = "Variable"
```

5.22 C:/Users/silas/CLionProjects/project3_khalil/varlen.h File Reference

```
#include <cstdlib>
#include <iostream>
#include "iobuffer.h"
```

Classes

- class VariableLengthBuffer

5.23 varlen.h

Go to the documentation of this file.

```
00001 // fvarlen.h
00002 #ifndef VARLEN_H
00003 #define VARLEN_H
00004
00005 #include <cstdlib>
00006 #include <iostream>
00007 #include "iobuffer.h"
00008
00009 using namespace std;
00010
00011 class VariableLengthBuffer: public IOBuffer
00012 // Abstract class designed to support variablelength records
00013 // Fields may be of a variety of types
00014 //
00015 {  public:
00016     VariableLengthBuffer (int MaxBytes = 1000);
00017     VariableLengthBuffer (const VariableLengthBuffer & buffer) // copy constructor
00018         : IOBuffer(buffer){}
00019
00020     void Clear (); // clear fields from buffer
00021     int Read (istream &);
00022     int Write (ostream &) const;
00023     int ReadHeader (istream &); // write a buffer to the stream
00024     int WriteHeader (ostream &) const; // write a buffer to the stream
00025     int PackFixLen (void *, int);
00026     int PackDelimeted (void *, int);
00027     int PackLength (void *, int);
00028     void Print (ostream &) const;
00029     int SizeOfBuffer () const; // return current size of buffer
00030     int Init ();
00031  protected:
00032 };
00033 //#include "varlen.cpp"
00034 #endif
```

5.24 C:/Users/silas/CLionProjects/project3_khalil/Zipcode.cpp File Reference

Implementation file for the Zipcode class.

```
#include "Zipcode.h"
```

5.24.1 Detailed Description

Implementation file for the Zipcode class.

5.25 C:/Users/silas/CLionProjects/project3_khalil/Zipcode.h File Reference

```
#include <iostream>
#include <cstring>
#include "fixfld.h"
#include "length.h"
#include "delim.h"
```

Classes

- class Zipcode

  Zipcode Information.

5.26 Zipcode.h

Go to the documentation of this file.

```
00001 // Zipcode.h
00002 #ifndef ZIPCODE_H
00003 #define ZIPCODE_H
00004
00005 #include <iostream>
00006 #include <cstring>
00007 #include "fixfld.h"
00008 #include "length.h"
00009 #include "delim.h"
00010 using namespace std;
00011
00015 class Zipcode
00016 {
00017   public:
00018
00019     // fields
00020     char zip [6];
00021     char place [24];
00022     char state [3];
00023     char county [16];
00024     char latitude[10];
00025     char longitude [10];
00031     Zipcode();
00032
00033     //operations
00038     const char* getZip() const { return zip; }
00039
00044     const char* getPlace() const { return place; }
00045     const char* getState() const { return state; }
00046     const char* getCounty() const { return county; }
00047     const char* getLatitude() const { return latitude; }
00048     const char* getLongitude() const { return longitude; }
00049
00054     void setZip(const char* z) { strcpy(zip, z); }
00055     void setPlace(const char* p) { strcpy(place, p); }
00056     void setState(const char* s) { strcpy(state, s); }
00057     void setCounty(const char* c) { strcpy(county, c); }
00058     void setLatitude(const char* lat) { strcpy(latitude, lat); }
00059     void setLongitude(const char* lon) { strcpy(longitude, lon); }
00060
00066     static int InitBuffer (DelimFieldBuffer &);
00067
00073     static int InitBuffer (LengthFieldBuffer &);
00074
00080     static int InitBuffer (FixedFieldBuffer &);
00081
00085     void Clear ();
00086
00092     int Unpack (IOBuffer &);
00093
00099     int Pack (DelimFieldBuffer) const;
00100
00106     void Print (ostream &, char * label=0) const;
00107 };
00108 //#include "Zipcode.cpp"
00109 #endif
```