CS 446 Project 1

Generated by Doxygen 1.8.6

Wed Sep 20 2017 22:43:08

Contents

Chapter 1

README

Author: Eugene Nelson Date: 9/17/17 Name: Queue Data Structure

```
This folder contains the source code for a custom linked queue data stucture.
```

Error Codes: 21 - Error in parse

31 - Error in meta data code 32 - Meta Data Code is is wrong case or non alphabetic 33 - Meta Data Code is missing

41 - Error in meta data descriptor 42 - Meta data descriptor missing

51 - Error in cycle values 52 - Cycle value is negative 53 - Cycle value is missing 54 - Cycle value is not numeric

61 -

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 documented files with brief descriptions:

Chapter 4

Class Documentation

4.1 ConfigFileInput Class Reference

Public Member Functions

- **ConfigFileInput** (char *fileName)
- **ConfigFileInput** (const ConfigFileInput &copyInput)
- int **GetNumberOfProcesses** ()
- int GetProcessValue (const char processName[])
- char * **GetProcessName** (const int position)
- char **GetLogOutputSpecification** ()
- char * **GetFilePath** ()
- char * **GetLogFilePath** ()

Protected Member Functions

- bool ParseLine (char lineToParse[])
- void RemoveSpaces (char lineToRemoveSpaces[])
- void AdjustLineElements (char lineToAdjust[], int positionToAdjust)

4.1.1 Member Function Documentation

4.1.1.1 void ConfigFileInput::AdjustLineElements (char *lineToAdjust[],* int *positionToAdjust*) [protected]

< Helper function to make removing spaces easier

Precondition

None.

Postcondition

The specified string will have all leading characters from the position specified moved back one space.

Parameters

lineToAdjust	The line that will be adjusted.
positionToAdjust	The position in the string to which we are adjusting.

Returns

void

4.1.1.2 int ConfigFileInput::GetProcessValue (const char *processName[]*)

< Searches for the specified process name in the list of processes to get it's value Simply searches through the list to find the position of the process we are looking for and uses that position to get the value.

Precondition

None.

Postcondition

The value of the process with the given name is returned.

Parameters

processName	The name of the process to search for.

Returns

The value of the specified process.

4.1.1.3 bool ConfigFileInput::ParseLine (char *lineToParse[]*) [protected]

< Helper function that parses the lines in the config file and stores the relevent information.

Precondition

None.

Postcondition

The line will be parsed through and the tokens will be extracted and saved in the nodes in the list.

Parameters

lineToParse	The line to parse through.

Returns

The status of the parse line function.

4.1.1.4 void ConfigFileInput::RemoveSpaces (char *lineToRemoveSpaces[]*) [protected]

< Helper function to make parsing easier

Precondition

None.

Postcondition

The specified string will have all space characters removed.

Parameters

lineToRemove-Spaces	The string who's spaces will be removed.

Returns

void

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

- ConfigFileInput.hh
- ConfigFileInput.cc

4.2 ConfigFileInputNode Class Reference

Public Member Functions

- **ConfigFileInputNode** (const char newProcessName[], const int newProcessValue)
- **ConfigFileInputNode** (const ConfigFileInputNode &copyNode)
- char * **GetProcessName** ()
- int **GetProcessValue** ()
- void **SetProcessName** (char newProcessName[])
- void **SetProcessValue** (int newProcessValue)
- bool **operator==** (const char processName[]) const
- ConfigFileInputNode **operator=** (const ConfigFileInputNode &copyNode)

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

- ConfigFileInputNode.hh
- ConfigFileInputNode.cc

4.3 LinkedList< ItemType > Class Template Reference

Public Member Functions

- LinkedList ()
- LinkedList (LinkedList< ItemType > &Source)
- ~LinkedList ()
- bool IsEmpty ()
- bool IsFull ()
- int GetLength ()
- bool RemoveEntry (int Position)
- void Clear ()
- bool InsertEntry (int Position, const ItemType &Data)
- bool push (const ItemType &Data)
- bool pop ()
- ItemType * GetEntry (int Position)
- void print ()

4.3.1 Constructor & Destructor Documentation

4.3.1.1 template<class ItemType > LinkedList< ItemType >::LinkedList ( )

< Default constructor for LinkedList

4.3.1.2 template<class ItemType> LinkedList< ItemType >::LinkedList (LinkedList< ItemType > & *Source*)

< Copy constructor for list

4.3.1.3 template<class ItemType > LinkedList< ItemType >::~LinkedList ( )

< Deconstructor for LinkedList

4.3.2 Member Function Documentation

4.3.2.1 template<class ItemType > void LinkedList< ItemType >::Clear ( )

< Clears the list

Returns

void

Precondition

RemoveEntry and IsEmpty functions are working properly

4.3.2.2 template<class ItemType > ItemType * LinkedList< ItemType >::GetEntry (int *Position*)

< Peeks at the data at a specified position

Returns

Returns a pointer to the data at specified position

Parameters

Position	The position of where to GetEntry

4.3.2.3 template<class ItemType > int LinkedList< ItemType >::GetLength ( )

< Returns the size of the LinkedList

Returns

Number of elements in the list

4.3.2.4 template<class ItemType> bool LinkedList< ItemType >::InsertEntry (int *Position*, const ItemType & *Data*)

< Inserts a new element at specified position

Returns

The success value of the function

Parameters

Position	The position where the element will be InsertEntryed
Data	The data that will be InsertEntryed

4.3.2.5 template<class ItemType > bool LinkedList< ItemType >::IsEmpty ( )

< Checks if the list is empty

Returns

Success value of function

4.3.2.6 template<class ItemType > bool LinkedList< ItemType >::IsFull ( )

< Checks if the list is full

Returns

Always returns false as linked lists do not have max size

4.3.2.7 template<class ItemType > bool LinkedList< ItemType >::pop ( )

< Pops data off of last element of list

Returns

Success value of function

4.3.2.8 template<class ItemType > void LinkedList< ItemType >::print ( )

< Prints all value in the linked list

Returns

void

4.3.2.9 template<class ItemType> bool LinkedList< ItemType >::push (const ItemType & *Data*)

< Pushes value onto the end of the list

Returns

The success value of the function

Parameters

Data	The data that will be pushed

4.3.2.10 template<class ItemType > bool LinkedList< ItemType >::RemoveEntry (int *Position*)

< Removes an element at specified position

Returns

Returns the Success value for the function

Parameters

Position	The position at which it is desired that an element be removed.

Postcondition

Element will be deleted from specified position

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

- LinkedList.hh

4.4 LinkedQueue< ItemType > Class Template Reference

`#include <LinkedQueue.hh>`

Public Member Functions

- LinkedQueue ()
- LinkedQueue (const LinkedQueue< ItemType > &source)
- ~LinkedQueue ()
- bool IsEmpty () const
- bool Enqueue (const ItemType &newEntry)
- bool Dequeue ()
- ItemType * PeekFront ()
- int **GetSize** ()

4.4.1 Detailed Description

template<class ItemType>class LinkedQueue< ItemType >

Default constructor for linked queue

4.4.2 Constructor & Destructor Documentation

4.4.2.1 template<class ItemType > LinkedQueue< ItemType >::LinkedQueue ( )

Copy constructor for Linked queue class

4.4.2.2 template<class ItemType> LinkedQueue< ItemType >::LinkedQueue (const LinkedQueue< ItemType > & *source*)

Deconstructor for linked queue class

4.4.2.3 template<class ItemType > LinkedQueue< ItemType >::~LinkedQueue ( )

Checks if linked queue is empty

Returns

succes status of function

4.4.3 Member Function Documentation

4.4.3.1 template<class ItemType > bool LinkedQueue< ItemType >::Dequeue ( )

Peeks at the front element of the queue

Returns

poiner to element at front of queue

4.4.3.2 template<class ItemType> bool LinkedQueue< ItemType >::Enqueue (const ItemType & *newEntry*)

Removes an element from the front of the queue

Returns

success status of the function

4.4.3.3 template<class ItemType > bool LinkedQueue< ItemType >::IsEmpty ( ) const

Adds new element into the queue

Returns

success status of function

4.4.3.4 template<class ItemType > ItemType * LinkedQueue< ItemType >::PeekFront ( )

Returns the number of elements in the queue

Returns

the size of the queue

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

- LinkedQueue.hh

4.5 MetaDataInfo Class Reference

Public Member Functions

- **MetaDataInfo** (char *fileName)
- void ProcessData (ConfigFileInput &configFile)

Protected Member Functions

- bool ParseLine (char lineToParse[])
- void RemoveSpaces (char lineToRemoveSpaces[])
- void AdjustLineElements (char lineToAdjust[], int positionToAdjust)
- void ProcessErrorCode (char logSpecification, char errorCode, ofstream &logFile)
- void **LogOutput** (char logSpecification, char *logMessage, ofstream &logFile)
- void itoa (int inputValue, char *outputString, int base)
- void ReverseString (char *string, int size)

4.5.1 Member Function Documentation

4.5.1.1 void MetaDataInfo::AdjustLineElements (char *lineToAdjust[],* int *positionToAdjust*) [protected]

< Helper function to make removing spaces easier

Precondition

None.

Postcondition

The specified string will have all leading characters from the position specified moved back one space.

Parameters

lineToAdjust	The line that will be adjusted.
positionToAdjust	The position in the string to which we are adjusting.

Returns

void

4.5.1.2 void MetaDataInfo::itoa (int *inputValue,* char * *outputString,* int *base*) [protected]

< Helper function to convert from an integer to a string

Precondition

None.

Postcondition

The integer value passed into the function will be converted into a string representation to make logging easier.

Parameters

inputValue	The integer value that will be converted to a string.
ouputString	The string where the converted value will be stored.
base	The base to convert to. i.e. 10 = decimal, 16 = hexadecimal.

Returns

void

4.5.1.3 bool MetaDataInfo::ParseLine (char *lineToParse[]*) [protected]

< Parses the line from Meta Data and stores the tokens in nodes in the queue

Precondition

None.

Postcondition

Meta Data line will be parsed and all important information will be stored in nodes within the queue of processes to be simulated later.

Parameters

lineToParse	The line from the MetaData file that will be parsed

Returns

The status of the parse, indicating errors in the parsing process.

Note

Will also set error codes for any potentially troublesome processes so that they may be delt with later, without crashing the program.

4.5.1.4 void MetaDataInfo::ProcessData (ConfigFileInput & *configFile*)

< Processes the data from the Meta Data file and logs it to appropriate output

Precondition

ConfigFile has all necessary information to process.

Postcondition

Simulation will have been run and output logged to appropriate outlets.

Parameters

configFile	The configuration file with necessary info to preform the simulation.

Returns

void

4.5.1.5 void MetaDataInfo::ProcessErrorCode (char *logSpecification,* char *errorCode,* ofstream & *logFile*) [protected]

< Helper function make dealing with errors easier

Precondition

None.

Postcondition

The specified error code will be handled and logged appropriately.

Parameters

logSpecification	The specification code indicating where to log error.
errorCode	The code that needs to be handled.
logFile	The file where the error will be logged if specified to log to file.

Returns

void Helper function to make logging messages easier

Precondition

None.

Postcondition

The message to be logged will be logged to the appropriate destinations.

Parameters

logSpecification	The specification code indicating where to log error.
logMessage	The message that needs to be logged.
logFile	The file where the error will be logged if specified to log to file.

Returns

void

4.5.1.6 void MetaDataInfo::RemoveSpaces (char *lineToRemoveSpaces[]*) [protected]

< Helper function to make parsing easier

Precondition

None.

Postcondition

The specified string will have all space characters removed.

Parameters

lineToRemove-Spaces	The string who's spaces will be removed.

Returns

void

4.5.1.7 void MetaDataInfo::ReverseString (char * *string,* int *size*) [protected]

< Helper function to make converting from int to string easier.

Precondition

None.

Postcondition

The string passed in will be reversed.

Parameters

string	The string to be reversed.
size	The size of the string passed in.

Returns

void

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

- MetaDataInfo.hh
- MetaDataInfo.cc

4.6 MetaDataInfoNode Class Reference

Public Member Functions

- **MetaDataInfoNode** (const MetaDataInfoNode &copyNode)
- **MetaDataInfoNode** (const char newMetaDataCode, const char newMetaDataDescriptor[], const int newNumberOfCycles, const int newErrorCode)
- char **SetMetaDataCode** (const char newMetaDataCode)
- char * **SetMetaDataDescriptor** (const char newMetaDataDescriptor[])
- int **SetNumberOfCycles** (const int newNumberOfCycles)
- int **SetErrorCode** (const int newErrorCode)
- char **GetMetaDataCode** () const
- char * **GetMetaDataDescriptor** () const
- int **GetNumberOfCycles** () const
- int **GetErrorCode** () const
- MetaDataInfoNode & **operator=** (const MetaDataInfoNode &copyNode)

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

- MetaDataInfoNode.hh
- MetaDataInfoNode.cc

4.7 Node< ItemType > Class Template Reference

Public Member Functions

- Node (const ItemType &SourceData, Node *NextNode)
- ~Node ()
- ItemType * getData ()
- Node< ItemType > * getNext ()
- void setData (const ItemType &SourceData)
- void setNext (Node< ItemType > *NextNode)

4.7.1 Constructor & Destructor Documentation

4.7.1.1 template<class ItemType> Node< ItemType >::Node (const ItemType & *SourceData,* Node< ItemType > * *NextNode*)

< Default paramaterized constructor for Node class

4.7.1.2 template<class ItemType > Node< ItemType >::~Node ( )

< Deconstructor for Node class

4.7.2 Member Function Documentation

4.7.2.1 template<class ItemType > ItemType * Node< ItemType >::getData ( )

< Gets the data stored in the node

Returns

Returns a pointer to the data stored in Node

4.7.2.2 template<class ItemType > Node< ItemType > * Node< ItemType >::getNext ( )

< Gets the value of the next pointer in Node

Returns

Pointer to next Node

4.7.2.3 template<class ItemType> void Node< ItemType >::setData (const ItemType & *SourceData*)

< Sets the data for the Node object

Returns

void

Parameters

SourceData	The data that will be used to set the data stored in Node

4.7.2.4 template<class ItemType> void Node< ItemType >::setNext (Node< ItemType > * *NextNode*)

< Sets the next Node

Returns

void

Parameters

NextNode	The node that will be set as the next Node

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

- Node.hh

Chapter 5

File Documentation

5.1 ConfigFileInput.hh File Reference

```
#include <iostream>
#include <fstream>
#include <cstring>
#include <stdlib.h>
#include "LinkedList.hh"
#include "ConfigFileInputNode.hh"
```

Classes

- class ConfigFileInput

Macros

- #define **FILE_NAME_MAX_LENGTH** 30
- #define **STR_MAX_LENGTH** 300
- #define **LOG_FILE_NAME_MAX_LENGTH** 30

5.1.1 Detailed Description

Author

Eugene Nelson The header file for the ConfigFileInput class.

Version

1.0 Eugene Nelson Originally developed (9 - 19 - 17)

5.2 ConfigFileInputNode.hh File Reference

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

Classes

- class ConfigFileInputNode

Macros

- #define **STR_MAX_LENGTH** 300

5.2.1 Detailed Description

<

Author

Eugene Nelson The header file for the ConfigFileInputNode class

Version

1.0 Eugene Nelson Originally Developed (9 - 18 - 17)

< Configuration file input nodes.

Nodes will store information from the configuration file and parse the config file to retrieve necessary information.

Author

Eugene Nelson 9/18/17

Version

1.0 (Eugene Nelson 9/18/17)

5.3 MetaDataInfo.cc File Reference

```
#include "MetaDataInfo.hh"
```

5.3.1 Detailed Description

Author

Eugene Nelson The implamentation file for the MetaDataInfo class.

Version

1.0 Eugene Nelson Originally developed (9 - 19 - 17)

5.4 MetaDataInfo.hh File Reference

```
#include <iostream>
#include <fstream>
#include <cstring>
#include <stdlib.h>
#include <cstdlib>
#include "MetaDataInfoNode.hh"
#include "LinkedQueue.hh"
#include "ConfigFileInput.hh"
```

Classes

- class MetaDataInfo

Macros

- #define **LINE_MAX_LEGNTH** 300
- #define **MESSAGE_MAX_LENGTH** 500

5.4.1 Detailed Description

Author

Eugene Nelson The header file for the MetaDataInfo class.

Version

1.0 Eugene Nelson Originally developed (9 - 19 - 17)

5.5 MetaDataInfoNode.cc File Reference

```
#include "MetaDataInfoNode.hh"
```

5.5.1 Detailed Description

Author

Eugene Nelson The implamentation for the MetaDataInfoNode class

Version

1.0 Eugene Nelon Originally developed (9 - 20 - 17)

5.6 MetaDataInfoNode.hh File Reference

```
#include <cstring>
```

Classes

- class MetaDataInfoNode

5.6.1 Detailed Description

Author

Eugene Nelson The header for the MetaDataInfoNode class

Version

1.0 Eugene Nelon Originally developed (9 - 20 - 17)

5.7 Node.hh File Reference

Classes

- class Node< ItemType >

5.7.1 Detailed Description

Author

Eugene Nelson The header/implamentation for the templated Node class

Version

1.0 Eugene Nelon Originally developed (10 - 15 - 16)

Index