PA10

Generated by Doxygen 1.8.9.1

Wed Apr 20 2016 23:50:10

Contents

1 Class Index

1.1 Class List

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

DataNode< DataType >	**2**
HashClass< DataType >	**3**

2 File Index

2.1 File List

Here is a list of all documented files with brief descriptions:

3 Class Documentation

3.1 DataNode< DataType > Struct Template Reference

Public Member Functions

- DataNode ()

  Default/Initialization DataNode constructor.

Public Attributes

- NodeState **usedState**
- DataType **nodeData**

3.1.1 Constructor & Destructor Documentation

3.1.1.1 template<typename DataType > **DataNode< DataType >::DataNode ( )**

Default/Initialization DataNode constructor.

sets usedState

Precondition

assumes Uninitialized DataNode object

Postcondition

DataNode initizilied

N/A

Exceptions

None	

Returns

None

None

Note

None

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

- HashClass.h
- HashClass.cpp
- HashClass_AssignmentBase.cpp

3.2 HashClass< DataType > Class Template Reference

Public Member Functions

- HashClass ()

  Default/Initialization HashClass constructor.
- HashClass (const HashClass< DataType > &copied)

  Default/Initialization HashClass copy constructor.
- ∼HashClass ()

  destructer HashClass
- const HashClass & operator= (const HashClass< DataType > &rhData)

  assignment operator
- bool setTableLength (int newTableLength, bool clearListFlag, int &maxProbes, int &totalProbes)

  setTableLength
- void setHashLetterCount (int newHashLetterCount)

  setHashLetterCount
- void setProbeAttempts (int newNumProbeAttempts)

  setProbeAttempts
- bool addItem (const DataType &newData, int &probeAttempts)

  addItem
- int findItem (const DataType &dataItem, int &probeAttempts) const

  findItem
- bool removeItem (const DataType &dataItem, int &probeAttempts)

  removeItem
- void clearList ()

  clearList
- bool isEmpty () const

  isEmpty
- void showStructure () const

  showStructure

Static Public Attributes

- static const char **TAB** = '\t'
- static const int **DEFAULT_HASH_TABLE_LENGTH** = 10
- static const int **DEFAULT_HASH_LETTER_COUNT** = 3
- static const int **DEFAULT_PROBE_ATTEMPT_LIMIT** = 10
- static const int **FAILED_PROBE_PROCESS** = -1
- static const int **STD_STR_LEN** = 50
- static const int **LARGE_STR_LEN** = 100
- static const bool **CLEAR_LIST** = true
- static const bool **NO_LIST_CLEAR** = false

Private Member Functions

- int hash (const DataType &dataItem, int workingTableLength) const

    hash
- int addItemHelper (DataNode< DataType > ∗destHashTable, const DataType &newData)

    addItemHelper
- bool resizeList (int newSize, bool clearFlag, int &maxProbes, int &totaProbes)

    resizeList
- void copyList (const DataNode< DataType > ∗copiedList)

    copyList
- int toPower (int base, int exponent) const

    power function

Private Attributes

- int **tableLength**
- int **maxProbeAttempts**
- int **hashLetterCount**
- DataNode< DataType > ∗ **hashList**

3.2.1 Constructor & Destructor Documentation

3.2.1.1 template<typename DataType > HashClass< DataType >::HashClass ( )

Default/Initialization HashClass constructor.

sets tableLength, MaxProbeAttemps,hashLetterCount,hashList

Precondition

assumes Uninitialized hashClass object

Postcondition

hashClass initizilied

N/A

Exceptions

None	

Returns

None

None

Note

None

3.2.1.2 template<typename DataType > HashClass< DataType >::HashClass (const HashClass< DataType > & *copied*)

Default/Initialization HashClass copy constructor.

sets tableLength, MaxProbeAttemps,hashLetterCount,hashList to copied class

Precondition

assumes Uninitialized hashClass object

Postcondition

hashClass initizilied

N/A

Exceptions

None	

Returns

None

copied - a hashclass that has its values copied

Note

None

3.2.1.3 template<typename DataType > HashClass< DataType >::∼HashClass ( )

destructer HashClass

deletes dynamic memory

Precondition

assumes initialized hashClass object

Postcondition

hashClass uninitizilied

N/A

Exceptions

None	

Returns

None

None

Note

None

3.2.2 Member Function Documentation

3.2.2.1 template<typename DataType > bool HashClass< DataType >::addItem (const DataType & *newData,* int & *probeAttempts*)

addItem

hashes item and adds it to the hashList

Precondition

assumes initialized hashClass object

Postcondition

hashes item and adds it to the hashList

N/A

Exceptions

None	

Returns

bool of success

newData - data to add probeAttempts - attempts to add data

Note

None

3.2.2.2 template<typename DataType > int HashClass< DataType >::addItemHelper (DataNode< DataType > * *destHashTable,* const DataType & *newData*) `[private]`

addItemHelper

does the probing of the addItem function

Precondition

N/a

Postcondition

adds data to hashlist and returns probeAttempts

None

Exceptions

None	

Parameters

destHash↩ Table,newData	

Returns

returns probeAttempts

Note

None

3.2.2.3 template<typename DataType > void HashClass< DataType >::clearList ( )

clearList

clears list by setting all to unused

Precondition

N/a

Postcondition

clears list by setting all to unused

None

Exceptions

None	

Parameters

None	

Returns

None

Note

None

3.2.2.4 template<typename DataType > void HashClass< DataType >::copyList (const DataNode< DataType > * *copiedList*) [private]

copyList

copies copiedList to this list

Precondition

N/a

Postcondition

copies copiedList to this list

None

Exceptions

None	

Parameters

copiedList	- to copy data from

Returns

None

Note

None

3.2.2.5 template<typename DataType > int HashClass< DataType >::findItem (const DataType & *dataItem,* int & *probeAttempts*) const

findItem

returns int of items index

Precondition

assumes initialized hashClass object

Postcondition

returns int of items index

N/A

Exceptions

None	

Returns

int of items index

dataItem - data to find probeAttempts - attempts to add data

Note

None

3.2.2.6 template< typename DataType > int HashClass< DataType >::hash (const DataType & *dataItem,* int *workingTableLength*) const [private]

hash

calls hashlist of DataType

Precondition

N/a

Postcondition

returns hash of DataType

None

Exceptions

None	

Parameters

newSize,clear↩ Flag,max↩ Probes,total↩ Probes	

Returns

int of hash value

Note

None

3.2.2.7 template< typename DataType > bool HashClass< DataType >::isEmpty ( ) const

isEmpty

returns if there is no data

Precondition

assumes initialized hashClass object

Postcondition

returns a bool

algorithm

calls findItem to get index to remove

Exceptions

None	

Returns

false

None

Note

None

3.2.2.8 template<typename DataType > const HashClass< DataType > & HashClass< DataType >::operator= (const HashClass< DataType > & *rhData*)

assignment operator

sets tableLength, MaxProbeAttemps,hashLetterCount,hashList to rhHashTable

Precondition

assumes initialized hashClass object

Postcondition

hashClass values set to rhHashTable values

N/A

Exceptions

None	

Returns

None

rhHashTable - copied values from

Note

None

3.2.2.9 template<typename DataType > bool HashClass< DataType >::removeItem (const DataType & *dataItem,* int & *probeAttempts*)

removeItem

returns int of items index

Precondition

assumes initialized hashClass object

Postcondition

removes dataItem

algorithm

calls findItem to get index to remove

Exceptions

None	

Returns

success of operation

dataItem - data to remove probeAttempts - attempts to add data

Note

None

3.2.2.10 template< typename DataType > bool HashClass< DataType >::resizeList (int *newSize*, bool *clearFlag*, int & *maxProbes*, int & *totalProbes*) [private]

resizeList

clears list if param allows, or makes size bigger

Precondition

N/a

Postcondition

resizes list to the newSize

None

Exceptions

None	

Parameters

newSize,clear↩ Flag,max↩ Probes,total↩ Probes	

Returns

bool if successfull or not

Note

None

3.2.2.11 template< typename DataType > void HashClass< DataType >::setHashLetterCount (int *newHashLetterCount*)

setHashLetterCount

sets hashLetterCount to newHashLetterCount

Precondition

assumes initialized hashClass object

Postcondition

sets hashLetterCount to newHashLetterCount

N/A

Exceptions

None	

Returns

None

newHashLetterCount - to change hashLetter to

Note

None

3.2.2.12 template<typename DataType > void HashClass< DataType >::setProbeAttempts (int *newNumProbeAttempts*)

setProbeAttempts

MaxProbeAttemps to newNumProbeAttempts

Precondition

assumes initialized hashClass object

Postcondition

MaxProbeAttemps to newNumProbeAttempts

N/A

Exceptions

None	

Returns

None

newNumProbeAttempts

Note

None

3.2.2.13 template<typename DataType > bool HashClass< DataType >::setTableLength (int *newTableLength,* bool *clearListFlag,* int & *maxProbes,* int & *totalProbes*)

setTableLength

sets tableLength

Precondition

assumes initialized hashClass object

Postcondition

tableLength to newTableLength and resizes

Algorithm

calls resizeList

Exceptions

None	

Returns

bool of success

newTableLength, clearListFlag, maxProbes, TotalProbes

Note

None

3.2.2.14 template<typename DataType > void HashClass< DataType >::showStructure ( ) const

showStructure

prints out data of hashList in order

Precondition

N/a

Postcondition

prints out data of hashList in order

None

Exceptions

None	

Parameters

None	

Returns

None

Note

None

3.2.2.15 template<typename DataType > int HashClass< DataType >::toPower (int *base,* int *exponent*) const [private]

power function

does power operations

Precondition

N/a

Postcondition

returns power operation

Algorithm

Exceptions

None	

Parameters

base,exponent	

Returns

None

Note

None

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

- HashClass.h
- HashClass.cpp
- HashClass_AssignmentBase.cpp

3.3 MedType Class Reference

Public Member Functions

- **MedType** (const char ∗patientName, const char ∗medCodeNum, char patientGender)
- **MedType** (const MedType &newMedObject)
- const MedType & **operator=** (const MedType &rhMed)
- void **setAccount** (const char ∗patientName, const char ∗medicalCodeNum, char patientGender)
- void **getAccount** (char ∗patientName, char ∗medicalCodeNum, char &patientGender) const
- int **compareTo** (const MedType &rhMed) const throw (logic_error)
- void **toString** (char ∗medStr)
- int **hash** (int numLetters, int hashTableLength)

Static Public Attributes

- static const char **NULL_CHAR** = '\0'
- static const char **COMMA** = ','
- static const char **SPACE** = ' '
- static const char **BASE_STR_LEN** = 20
- static const int **STD_NAME_LEN** = 100

Private Member Functions

- void **copyString** (char ∗destination, const char ∗source) const
- void **concatString** (char ∗destination, const char ∗source) const
- void **concatChar** (char ∗destination, const char source) const
- int **getStrLen** (const char ∗str) const
- char **toUpper** (char letter) const

Private Attributes

- char ∗ **name**
- char ∗ **medCodeNum**
- char **gender**

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

- MedType.h
- MedType.cpp

3.4 SimpleTimer Class Reference

Public Member Functions

- SimpleTimer ()

  Default constructor.
- ∼SimpleTimer ()

  Default constructor.
- void start ()

  Start control.
- void stop ()

  Stop control.
- void **getElapsedTime** (char ∗timeStr)

Static Public Attributes

- static const char **NULL_CHAR** = '\0'
- static const char **RADIX_POINT** = '.'

Private Attributes

- struct timeval startData **endData**
- long int **beginTime**
- long int **endTime**
- long int **secTime**
- long int **microSecTime**
- bool **running**
- bool **dataGood**

3.4.1 Constructor & Destructor Documentation

3.4.1.1 SimpleTimer::SimpleTimer ( )

Default constructor.

Constructs Timer class

Parameters

None	

Note

set running flag to false

3.4.1.2 SimpleTimer::∼SimpleTimer ( )

Default constructor.

Destructs Timer class

Parameters

None	

Note

No data to clear

3.4.2 Member Function Documentation

3.4.2.1 void SimpleTimer::start ( )

Start control.

Takes initial time data

Parameters

None	

Note

None

3.4.2.2 void SimpleTimer::stop ( )

Stop control.

Takes final time data, calculates duration

Parameters

None	

Note

None

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

- SimpleTimer.h
- SimpleTimer.cpp

4 File Documentation

4.1 HashClass.cpp File Reference

Implementation file for HashClass class.

```
#include "HashClass.h"
```

4.1.1 Detailed Description

Implementation file for HashClass class.

Implements the constructor method of the HashClass class

Version

1.10 Michael Leverington (06 April 2016) Updated with probing

1.00 Michael Leverington (06 November 2015) Original code

Requires HashClass.h

4.2 HashClass.h File Reference

Definition file for HashClass.

```
#include <iostream>
```

Classes

- struct DataNode< DataType >
- class HashClass< DataType >

Enumerations

- enum **NodeState** { **USED**, **UNUSED** }

4.2.1 Detailed Description

Definition file for HashClass.

Specifies all data and other members of the HashClass

Version

1.10 Michael Leverington (06 April 2016) Updated with probing

1.00 Michael Leverington (06 November 2015) Original code

None

4.3 MedType.cpp File Reference

Implementation file for MedType class.

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

4.3.1 Detailed Description

Implementation file for MedType class.

Implements member actions of the MedType class

Author

Michael Leverington

Version

1.00 (30 October 2015)

Requires MedType.h

4.4 MedType.h File Reference

Definition file for MedType class.

```
#include <ostream>
#include <stdexcept>
```

Classes

- class MedType

Variables

- const char **NAME_DEFAULT** [ ] = "Name Default"
- const char **CODE_NUM_DEFAULT** [ ] = "Code Num Default"

4.4.1 Detailed Description

Definition file for MedType class.

Specifies all data of the MedType class, along with the constructor, MedType class is entered and stored as a string

Author

Michael Leverington

Version

1.00 (30 October 2015)

None

4.5 SimpleTimer.cpp File Reference

Implementation file for SimpleTimer class.

```
#include "SimpleTimer.h"
```

4.5.1 Detailed Description

Implementation file for SimpleTimer class.

Author

Michael Leverington

Implements member methods for timing

Version

1.00 (11 September 2015)

Requires SimpleTimer.h.

4.6 SimpleTimer.h File Reference

Definition file for simple timer class.

```
#include <sys/time.h>
#include <cstring>
```

Classes

- class SimpleTimer

4.6.1 Detailed Description

Definition file for simple timer class.

Author

Michael Leverington

Specifies all member methods of the SimpleTimer

Version

1.00 (11 September 2015)

None

Index