Project 6

Tanner Jones
Version 1.0
10/15/15

Table of Contents

Class Index

Class List

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

File Index

File List

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

Class Documentation

BSTClass< NameType > Class Template Reference

Public Member Functions

- **BSTClass** ()
  Default constructor.
- **BSTClass** (const **BSTClass**< **NameType** > &copied)
  Copy constructor.
- **~BSTClass** ()
  Destructor.
- const **BSTClass** & **operator=** (const **BSTClass**< **NameType** > &rhData)
  operator=
- void **copyTree** (const **BSTClass**< **NameType** > &copied)
  copyTree
- void **clearTree** ()
  clearTree
- void **insert** (const **NameType** &newData)
  insert
- bool **findItem** (const **NameType** &searchDataItem) const
  findItem
- bool **removeItem** (const **NameType** &dataItem)
  removeItem
- bool **isEmpty** () const
  isEmpty
- void **preOrderTraversal** () const
  preOderTraversal
- void **inOrderTraversal** () const
  inOrderTraversal
- void **postOrderTraversal** () const
  postOrderTraversal
- int **getHeight** () const
- void **showStructure** () const
- **BSTClass** (const **BSTClass**< DataType > &copied)
- const **BSTClass** & **operator=** (const **BSTClass**< DataType > &rhData)
- void **copyTree** (const **BSTClass**< DataType > &copied)
- void **clearTree** ()
- void **insert** (const DataType &newData)
- bool **findItem** (const DataType &searchDataItem) const
- bool **removeItem** (const DataType &dataItem)
- bool **isEmpty** () const
- void **preOrderTraversal** () const
- void **inOrderTraversal** () const
- void **postOrderTraversal** () const
- int **getHeight** () const
- void **showStructure** () const

Static Public Attributes

- static const char **TAB** = '\t'

Constructor & Destructor Documentation

template<class NameType > BSTClass< NameType >::BSTClass ()

Default constructor.

Constructs **BSTClass**

Parameters:

None	

Note:

none

template<class NameType > BSTClass< NameType >::BSTClass (const BSTClass< NameType > & *copied*)

Copy constructor.

Constructs a copy of **BSTClass**

Parameters:

BSTClass	to copy

Note:

none

template<class NameType > BSTClass< NameType >::~BSTClass ()

Destructor.

destroys the **BSTClass**

Parameters:

None	

Note:

none

Member Function Documentation

template<class NameType > void BSTClass< NameType >::clearTree ()

clearTree

deletes the whole tree

Parameters:

none	

Note:

none

template<class NameType > void BSTClass< NameType >::copyTree (const BSTClass< NameType > & *copied*)

copyTree

copies a tree into another tree

Parameters:

binary	tree

Note:

none

template<class NameType > bool BSTClass< NameType >::findItem (const NameType & *searchDataItem*) const

findItem

finds a given **NameType**

Parameters:

NameType	

Note:

none

template<class NameType > void BSTClass< NameType >::inOrderTraversal () const

inOrderTraversal

loops through the tree and prints out in in order traversal

Parameters:

none	

Note:

none

template<class NameType > void BSTClass< NameType >::insert (const NameType & *newData*)

insert

inserts a new node

Parameters:

NameType	

Note:

none

template<class NameType > bool BSTClass< NameType >::isEmpty () const

isEmpty

tests to see if tree is empty

Parameters:

none	

Note:

none

template<class NameType > const BSTClass< NameType > & BSTClass< NameType >::operator= (const BSTClass< NameType > & *rhData*)

operator=

overloads the assignment operator

Parameters:

copied	tree

Note:

none

template<class NameType > void BSTClass< NameType >::postOrderTraversal () const

postOrderTraversal

loops through the tree and prints out in post order traversal

Parameters:

none	

Note:

none

template<class NameType > void BSTClass< NameType >::preOrderTraversal () const

preOderTraversal

loops through the tree and prints out in preOrder traversal

Parameters:

none	

Note:

none

template<class NameType > bool BSTClass< NameType >::removeItem (const NameType & *dataItem*)

removeItem

finds a given item and removes it

Parameters:

NameType	

Note:

none

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

- **BSTClass.h**
- BSTClass_PublicMethods.h
- **BSTClass.cpp**

BSTNode< NameType > Class Template Reference

Public Member Functions

- **BSTNode** (const **NameType** &nodeData, **BSTNode** *leftPtr, **BSTNode** *rightPtr)
  Default constructor.
- **BSTNode** (const DataType &nodeData, **BSTNode** *leftPtr, **BSTNode** *rightPtr)

Public Attributes

- **NameType dataItem**
- **BSTNode**< **NameType** > * **left**
- **BSTNode**< **NameType** > * **right**
- DataType **dataItem**
- **BSTNode**< DataType > * **left**
- **BSTNode**< DataType > * **right**

Constructor & Destructor Documentation

template<class NameType> BSTNode< NameType >::BSTNode (const NameType & *nodeData*, BSTNode< NameType > * *leftPtr*, BSTNode< NameType > * *rightPtr*)

Default constructor.

Constructs **BSTNode**

Parameters:

NameType	data, left and right pointers

Note:

none

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

- **BSTClass.h**
- BSTClass_PublicMethods.h
- **BSTClass.cpp**

NameType Class Reference

Public Member Functions

- **NameType** ()
  Default constructor.
- **NameType** (const char *newName)
  Initialization constructor.
- **NameType** (const **NameType** &newNameObject)
  Copy constructor.
- **~NameType** ()
  Destructor.
- const **NameType** & **operator=** (const **NameType** &rhName)
  Overloaded assignment operator.
- bool **setName** (const char *newName)
  Sets name in data type.
- void **getName** (char *retName) const
  Gets name from data type.
- int **compareTo** (const **NameType** &rhName) const throw (logic_error)
  Compares this name against another.

Static Public Attributes

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

Constructor & Destructor Documentation

NameType::NameType ()

Default constructor.

Constructs empty **NameType**

Parameters:

None	

Note:

None

NameType::NameType (const char * *newName*)

Initialization constructor.

Places name data into object

Parameters:

in	New string name

Note:

None

NameType::NameType (const NameType & *newNameObject*)

Copy constructor.

Places name data into object

Parameters:

in	New **NameType** object

Note:

None

NameType::~NameType ()

Destructor.

Non-acting destructor, no dynamic data

Parameters:

None	

Note:

None

Member Function Documentation

int NameType::compareTo (const NameType & *rhName*) const throw logic_error)

Compares this name against another.

Return < 0 if this item is less than right hand item Return > 0 if this item is greater than right hand item Return 0 if this item is equal to right hand item

Parameters:

out	returned name

Note:

None

void NameType::getName (char * *retName*) const

Gets name from data type.

Return data as c-string

Parameters:

out	returned name

Note:

None

const NameType & NameType::operator= (const NameType & *rhName*)

Overloaded assignment operator.

Assign data to other **NameType**

Parameters:

in	Assigned name

Note:

None

bool NameType::setName (const char * *newName*)

Sets name in data type.

Assign data to c-string

Parameters:

in	Assigned name

Note:

Attempts to standardize name (LastName, FirstName)

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

- **NameType.h**
- **NameType.cpp**

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** = '.'

Constructor & Destructor Documentation

SimpleTimer::SimpleTimer ()

Default constructor.

Constructs Timer class

Parameters:

None	

Note:

set running flag to false

SimpleTimer::~SimpleTimer ()

Default constructor.

Destructs Timer class

Parameters:

None	

Note:

No data to clear

Member Function Documentation

void SimpleTimer::start ()

Start control.

Takes initial time data

Parameters:

None	

Note:

None

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**

File Documentation

BSTClass.cpp File Reference

Definition file for Binary Search Tree class.

```
#include "BSTClass.h"
#include "NameType.h"
#include <iostream>
```

Detailed Description

Definition file for Binary Search Tree class.

Implements all given functions of the BST class

Author:

Tanner Jones

Version:

1.00 (15 October 2015)

None

BSTClass.h File Reference

Definition file for Binary Search Tree class.
`#include "NameType.h"`

Classes

- class **BSTNode< NameType >**
- class **BSTClass< NameType >**

Detailed Description

Definition file for Binary Search Tree class.

Defines all given functions of the BST class

Author:
Tanner Jones

Version:
1.00 (15 October 2015)

None

Specifies all data of the BST class

Author:
Michael Leverington

Version:
1.00 (03 October 2015)

None

NameType.cpp File Reference

Implementation file for **NameType** class.

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

Functions

- ostream & **operator<<** (ostream &outStream, const **NameType** &name)
  ostream output operator

Detailed Description

Implementation file for **NameType** class.

Implements the constructor method of the **NameType** class

Author:

Michael Leverington

Version:

1.00 (03 October 2015)

Requires **NameType.h**

Function Documentation

ostream& operator<< (ostream & *outStream*, const NameType & *name*)

ostream output operator

Free function outputs **NameType** to stream

Parameters:

in	ostream file object
in	**NameType** data item

Note:

None

NameType.h File Reference

Definition file for **NameType** class.
```
#include <ostream>
#include <stdexcept>
```

Classes

- class **NameType**

Functions

- ostream & **operator<<** (ostream &outStream, const **NameType** &name)
  ostream output operator

Detailed Description

Definition file for **NameType** class.

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

Author:
Michael Leverington

Version:
1.00 (03 October 2015)

None

Function Documentation

ostream& operator<< (ostream & *outStream*, const NameType & *name*)

ostream output operator

Free function outputs **NameType** to stream

Parameters:

in	ostream file object
in	**NameType** data item

Note:
None

PA06.cpp File Reference

Driver program to exercise the BST class.

```
#include <iostream>
#include <cstring>
#include "NameType.h"
#include "BSTClass.cpp"
```

Functions

- bool **getALine** (istream &consoleIn, char *str)
  Gets name in the form <Last name>="">, <First name>="">
- int **main** ()

Variables

- const char **ENDLINE_CHAR** = '\n'
- const char **NULL_CHAR** = '\0'
- const int **MAX_NAME_LEN** = 80

Detailed Description

Driver program to exercise the BST class.

Allows for testing the BST class, along with a timer class that will be used for evaluation

Version:

1.00 (3 October 2015)

Requires iostream, cstring, **NameType.h**, and **BSTClass.h**

Function Documentation

bool getALine (istream & *consoleIn*, char * *str*)

Gets name in the form <Last name>="">, <First name>="">

dates are input using cin, and then recombined for string accommodates testing (Submit) system

Parameters:

in	istream object
out	string with date

Note:

resolution for redirected input, getline did not work

SimpleTimer.cpp File Reference

Implementation file for **SimpleTimer** class.

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

Detailed Description

Implementation file for **SimpleTimer** class.

Author:

Michael Leverington

Implements member methods for timing

Version:

1.00 (11 September 2015)

Requires **SimpleTimer.h**.

SimpleTimer.h File Reference

Definition file for simple timer class.

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

Classes

- class **SimpleTimer**

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

INDEX