Reference Manual

Generated by Doxygen 1.5.1

Wed Sep 12 10:34:23 2007

Contents

Chapter 1

Class Index

1.1 Class List

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

Chapter 2

Class Documentation

2.1 Node< T > Class Template Reference

`#include <node.h>`

Public Member Functions

- **Node** (const T &=T())
- void **setObject** (const T &)
- T & **getObject** ()
- void **setNextPtr** (**Node**< T > ∗)
- **Node**< T > ∗ **getNextPtr** () const

2.1.1 Detailed Description

template<typename T> class Node< T >

Uses a template class to implement a node for a singly-linked list.

Author:

Mark Maloof

Version:

1.0, 28 August 2007

2.1.2 Constructor & Destructor Documentation

2.1.2.1 template<typename T> Node< T >::Node (const T & *object* = `T()`)

Constructor. Stores the object, which may be the default object, in this node. Sets this node's next pointer to null.

Parameters:

object the object to be stored in this node

2.1.3 Member Function Documentation

2.1.3.1 template<typename T> void Node< T >::setObject (const T & *object*)

Sets the object of this node.

Parameters:

object the object to be stored in this node

2.1.3.2 template<typename T> T & Node< T >::getObject ()

Returns a reference to the object stored in this node

Returns:

a reference to the object

2.1.3.3 template<typename T> void Node< T >::setNextPtr (Node< T > ∗ *nextPtr*)

Sets the next pointer of this node.

Parameters:

nextPtr the address to be stored in this node

2.1.3.4 template<typename T> Node< T > ∗ Node< T >::getNextPtr () const

Returns the pointer stored in this node.

Returns:

a pointer to the next node

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

- node.h

2.2 Stack< T > Class Template Reference

`#include <stack.h>`

Public Member Functions

- **Stack** ()
- **Stack** (const **Stack** &) throw (bad_alloc)
- ~**Stack** ()
- bool **empty** () const
- unsigned **size** () const
- void **clear** ()
- void **push** (const T &) throw (bad_alloc)
- T **pop** () throw (StackEmpty)
- T & **top** () const throw (StackEmpty)
- const **Stack**< T > & **operator=** (const **Stack**< T > &) throw (bad_alloc)

2.2.1 Detailed Description

template<typename T> class Stack< T >

Implements a stack. Uses a dynamically-allocated, singly-linked list of Node<T> objects.

Author:

Mark Maloof

Version:

1.0, 28 August 2007

2.2.2 Constructor & Destructor Documentation

2.2.2.1 template<typename T> Stack< T >::Stack ()

Default constructor.

2.2.2.2 template<typename T> Stack< T >::Stack (const Stack< T > & *s*) throw (bad_alloc)

Copy constructor.

Parameters:

s the stack to be copied

Exceptions:

bad_alloc if memory for the new stack cannot be allocated

2.2.2.3 template<typename T> Stack< T >::~Stack ()

Destructor.

2.2.3 Member Function Documentation

2.2.3.1 template<typename T> bool Stack< T >::empty () const

Returns true if this stack is empty; otherwise, returns false.

Returns:

a bool indicating whether the stack is empty

2.2.3.2 template<typename T> unsigned Stack< T >::size () const

Returns the number of items in the stack.

Returns:

an unsigned int indicating the number of items

2.2.3.3 template<typename T> void Stack< T >::clear ()

Clears the stack by deleting each item.

2.2.3.4 template<typename T> void Stack< T >::push (const T & *item*) throw (bad_alloc)

Pushes the item onto this stack.

Parameters:

item the item to be added to the top of this stack

Exceptions:

bad_alloc if memory for the new item cannot be allocated

2.2.3.5 template<typename T> T Stack< T >::pop () throw (StackEmpty)

Pops (i.e., removes) the item on the top of the stack.

Returns:

the item on top of the stack

Exceptions:

StackEmpty *(*p. *8)* if the stack is empty

2.2.3.6 template<typename T> T & Stack< T >::top () const throw (StackEmpty)

Returns a reference to the item on the top of the stack.

Returns:

a reference to the item on top of the stack

Exceptions:

StackEmpty (p. ***8)*** if the stack is empty

2.2.3.7 template<typename T> const Stack< T > & Stack< T >::operator= (const Stack< T > & *s*) throw (bad_alloc)

Overloads the memberwise copy operator. Returns a reference to the copied stack for cascaded assignments (i.e., s1 = s2 = s3).

Parameters:

s the stack to be copied

Returns:

a reference to the copied stack

Exceptions:

bad_alloc if memory for the new stack cannot be allocated

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

- stack.h

2.3 StackEmpty Class Reference

`#include <stack.h>`

Public Member Functions

StackEmpty (const string &what)

2.3.1 Detailed Description

Implements a runtime exception class for empty stacks.

Author:

Mark Maloof

Version:

1.0, 28 August 2007

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

stack.h

Index