Stack

1.0

Generated by Doxygen 1.9.0

Chapter 1

Class Index

1.1 Class List

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

Chapter 2

File Index

2.1 File List

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

Chapter 3

Class Documentation

3.1 Node< T > Struct Template Reference

The Node struct is meant to hold the data and pointer to the previous stack element.

```
#include <Stack.hpp>
```

Public Member Functions

- Node (const T &data)
- Node (T &&data)
- ~Node ()

Public Attributes

- T data
- Node< T > * previous

Friends

- std::ostream & **operator<<** (std::ostream &output, const Node< T > &node)

3.1.1 Detailed Description

template<typename T>
struct Node< T >

The Node struct is meant to hold the data and pointer to the previous stack element.

Template Parameters

T	Any data type or class.

3.1.2 Constructor & Destructor Documentation

3.1.2.1 Node() [1/2]

```
template<typename T >
Node< T >::Node (
             const T & data )  [inline]
```

Copy Constructor.

3.1.2.2 Node() [2/2]

```
template<typename T >
Node< T >::Node (
             T && data )  [inline]
```

Move Constructor.

3.1.2.3 ∼Node()

```
template<typename T >
Node< T >::∼Node ( )  [inline]
```

Struct Destructor.

3.1.3 Member Data Documentation

3.1.3.1 data

```
template<typename T >
T Node< T >::data
```

The data.

3.1.3.2 previous

```
template<typename T >
Node<T>* Node< T >::previous
```

Pointer to the previous node in the stack.

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

- Stack.hpp

3.2 Stack< T > Class Template Reference

A generic Stack class.

```
#include <Stack.hpp>
```

Public Member Functions

- Stack ()

  Default Constructor.
- Stack (const Stack< T > &copyStack)

  Copy Constructor.
- Stack (Stack< T > &&moveStack)

  Move Constructor.
- ∼Stack ()

  Class Destructor.
- void push (const T data)

  Adds element to the top of the stack.
- T pop ()

  Removes and returns the top of the stack.
- T peek ()

  Returns, but does not remove, the top of the stack.
- int size ()

  Returns the size of the stack.
- bool empty ()

  Returns true if the stack is empty and false otherwise.
- void clear ()

  Clears the entire stack recursively and resets all field elements to default.
- Stack< T > & operator= (const Stack &copyStack)

  Copy assignment operator.
- Stack< T > & operator= (Stack &&moveStack)

  Move assignment operator.

Friends

- template<typename Type >
  std::ostream & **operator<<** (std::ostream &output, const Stack< Type > &stack)

3.2.1 Detailed Description

template<typename T>
class Stack< T >

A generic Stack class.

This Stack class is templated to use any data type or class.

Template Parameters

T	Any data type or class.

3.2.2 Constructor & Destructor Documentation

3.2.2.1 Stack() [1/3]

```
template<typename T >
Stack< T >::Stack
```

Default Constructor.

Template Parameters

T	Any data type or class.

Initializes this stack object with a nullptr top and size of 0;

3.2.2.2 Stack() [2/3]

```
template<typename T >
Stack< T >::Stack (
             const Stack< T > & copyStack )
```

Copy Constructor.

Template Parameters

T	Any data type or class.

Parameters

copyStack	The stack whose elements will be copied into this stack object.

3.2.2.3 Stack() [3/3]

```
template<typename T >
Stack< T >::Stack (
             Stack< T > && moveStack )
```

Move Constructor.

Template Parameters

T	Any data type or class.

Parameters

moveStack	The stack whose elements will be moved into this stack object.

Note

std::move() needs to be used to call this constructor.

3.2.2.4 ∼Stack()

```
template<typename T >
Stack< T >::∼Stack
```

Class Destructor.

Template Parameters

T	Any data type or class.

Recursively clears the stack using clear() function.

3.2.3 Member Function Documentation

3.2.3.1 clear()

```
template<typename T >
void Stack< T >::clear
```

Clears the entire stack recursively and resets all field elements to default.

Template Parameters

T	Any data type or class.

3.2.3.2 empty()

```
template<typename T >
bool Stack< T >::empty
```

Returns true if the stack is empty and false otherwise.

Template Parameters

T	Any data type or class.

Returns

A boolean flag.

3.2.3.3 operator=() [1/2]

```
template<typename T >
Stack< T > & Stack< T >::operator= (
              const Stack< T > & copyStack )
```

Copy assignment operator.

Template Parameters

T	Any data type or class.

Parameters

copyStack	The stack object from which to copy elements.

Returns

A reference to a copied stack object.

Copies a stack with the help of the copy constructor and a custom swap function.

3.2.3.4 operator=() [2/2]

```
template<typename T >
Stack< T > & Stack< T >::operator= (
              Stack< T > && moveStack )
```

Move assignment operator.

Template Parameters

T	Any data type or class.

Parameters

moveStack	The stack object from which to move elements.

Returns

A reference to a moved stack object.

Moves stack elements from the provided stack into this stack object. The provided stack object is empty after the move is complete.

Note

std::move() needs to be used to call this operator.

3.2.3.5 peek()

```
template<typename T >
T Stack< T >::peek
```

Returns, but does not remove, the top of the stack.

Template Parameters

T	Any data type or class.

Returns

The element located at the top of the stack.

Exceptions

std::underflow_error	

Warning

Throws an Underflow Error exception if the stack is empty when function is called.

3.2.3.6 pop()

```
template<typename T >
T Stack< T >::pop
```

Removes and returns the top of the stack.

Template Parameters

T	Any data type or class.

Returns

The removed element.

Exceptions

std::underflow_error	

Warning

Throws an Underflow Error exception if the stack is empty when function is called.

3.2.3.7 push()

```
template<typename T >
void Stack< T >::push (
             const T data )
```

Adds element to the top of the stack.

Template Parameters

T	Any data type or class.

Parameters

data	The element you want to add to the stack.

3.2.3.8 size()

```
template<typename T >
int Stack< T >::size
```

Returns the size of the stack.

Template Parameters

T	Any data type or class.

Returns

The size of the stack.

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

- Stack.hpp

Chapter 4

File Documentation

4.1 Stack.hpp File Reference

A generic Stack data structure.

```
#include <iostream>
#include <stdexcept>
```

Classes

- struct Node< T >

  The Node struct is meant to hold the data and pointer to the previous stack element.
- class Stack< T >

  A generic Stack class.

Functions

- template<typename T >
  std::ostream & operator<< (std::ostream &output, const Stack< T > &stack)

  Output stream operator.

4.1.1 Detailed Description

A generic Stack data structure.

Copyright © 2021 Al Timofeyev. All rights reserved.

Author

Al Timofeyev

Date

January 18, 2021

Version: 1.0 Modified By: Modified Date:

4.1.2 Function Documentation

4.1.2.1 operator<<()

```
template<typename T >
std::ostream& operator<< (
            std::ostream & output,
            const Stack< T > & stack )
```

Output stream operator.

Template Parameters

T	Any data type or class.

Parameters

output	The output stream (usually std::cout).
stack	The stack object that will be printed.

Prints the stack elements to the specified output stream. Prints starting from the bottom of the stack and finishes printing at the top of the stack (BOTTOM, ... , TOP).

Note

Any class or data type used with this stack class MUST implement its own operator<< in order for this operator<< to work correctly. All primitive data types (int, float, double, char, string, bool) already have this operator functionality so no implementation for them is needed. But any custom class that is used with this stack class NEEDS to implement its own operator<<.

Index