Doubly Linked List

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 DLinkedList< T > Class Template Reference

A generic Doubly Linked List class.

`#include <DLinkedList.hpp>`

Public Member Functions

- DLinkedList ()
  Default Constructor.
- DLinkedList (const DLinkedList< T > &copyList)
  Copy Constructor.
- DLinkedList (DLinkedList< T > &&moveList)
  Move Constructor.
- ~DLinkedList ()
  Class Destructor.
- void addFirst (const T data)
  Adds data to the front of the list.
- void addLast (const T data)
  Adds data to the end of the list.
- void insert (const T data, const int index)
  Inserts data into the list at the specified index.
- void remove (const int index)
  Removes data from the list at the specified index.
- void clear ()
  Clears the entire list recursively and resets all field elements to default.
- T pop ()
  Removes and returns the head of the list.
- T pop (const int index)
  Removes and returns data from the list at the specified index.
- T peek ()
  Returns, but does not remove, the head of the list.
- T peek (const int index)
  Returns, but does not remove, data from the list at the specified index.

- int size ()

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

  Returns true if the list is empty and false otherwise.
- T & operator[ ] (const int index)

  Subscript operator.
- bool operator== (const DLinkedList &compareList)

  Equality comparison operator.
- DLinkedList< T > & operator= (const DLinkedList &copyList)

  Copy assignment operator.
- DLinkedList< T > & operator= (DLinkedList &&moveList)

  Move assignment operator.

Friends

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

3.1.1 Detailed Description

template<typename T>
class DLinkedList< T >

A generic Doubly Linked List class.

This Doubly Linked List is templated to use any data type or class.

Template Parameters

T	Any data type or class.

3.1.2 Constructor & Destructor Documentation

3.1.2.1 DLinkedList() [1/3]

```
template<typename T >
DLinkedList< T >::DLinkedList
```

Default Constructor.

Template Parameters

T	Any data type or class.

Initializes this linked list object with a nullptr head and tail and size of 0;

3.1.2.2 DLinkedList() [2/3]

```
template<typename T >
DLinkedList< T >::DLinkedList (
            const DLinkedList< T > & copyList )
```

Copy Constructor.

Template Parameters

T	Any data type or class.

Parameters

copyList	The list whose contents will be copied into this linked list object.

3.1.2.3 DLinkedList() [3/3]

```
template<typename T >
DLinkedList< T >::DLinkedList (
            DLinkedList< T > && moveList )
```

Move Constructor.

Template Parameters

T	Any data type or class.

Parameters

moveList	The list whose contents will be moved into this linked list object.

Note

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

3.1.2.4 ~DLinkedList()

```
template<typename T >
DLinkedList< T >::~DLinkedList
```

Class Destructor.

Template Parameters

T	Any data type or class.

Recursively clears the list using clear() function.

3.1.3 Member Function Documentation

3.1.3.1 addFirst()

```
template<typename T >
void DLinkedList< T >::addFirst (
            const T data )
```

Adds data to the front of the list.

Template Parameters

T	Any data type or class.

Parameters

data	The data you want to add to the list.

3.1.3.2 addLast()

```
template<typename T >
void DLinkedList< T >::addLast (
            const T data )
```

Adds data to the end of the list.

Template Parameters

T	Any data type or class.

Parameters

data	The data you want to add to the list.

3.1.3.3 clear()

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

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

Template Parameters

T	Any data type or class.

3.1.3.4 empty()

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

Returns true if the list is empty and false otherwise.

Template Parameters

T	Any data type or class.

Returns

A boolean flag.

3.1.3.5 insert()

```
template<typename T >
void DLinkedList< T >::insert (
            const T data,
            const int index )
```

Inserts data into the list at the specified index.

Template Parameters

T	Any data type or class.

Parameters

data	The data you want to add to the list.
index	Index at which to add the data into the list.

3.1.3.6 operator=() [1/2]

```
template<typename T >
DLinkedList< T > & DLinkedList< T >::operator= (
            const DLinkedList< T > & copyList )
```

Copy assignment operator.

Template Parameters

T	Any data type or class.

Parameters

copyList	The linked list object from which to copy elements.

Returns

A reference to a copied linked list object.

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

3.1.3.7 operator=() [2/2]

```
template<typename T >
DLinkedList< T > & DLinkedList< T >::operator= (
            DLinkedList< T > && moveList )
```

Move assignment operator.

Template Parameters

T	Any data type or class.

Parameters

moveList	The linked list object from which to move elements.

Returns

A reference to a moved linked list object.

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

Note

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

3.1.3.8 operator==()

```
template<typename T >
bool DLinkedList< T >::operator== (
            const DLinkedList< T > & compareList )
```

Equality comparison operator.

Template Parameters

T	Any data type or class.

Parameters

compareList	The linked list object with which to compare this linked list object.

Returns

A boolean flag.

Returns true only if the objects are either the same object or both objects have the exact same elements in the exact same index locations. If the elements are the same but in different index locations, the objects are not considered similar. A false flag is returned for all other outcomes.

3.1.3.9 operator[]()

```
template<typename T >
T & DLinkedList< T >::operator[] (
            const int index )
```

Subscript operator.

Template Parameters

T	Any data type or class.

Parameters

index	Index at which to retrieve data reference from the list.

Returns

The reference to the data located at the specified index in the list.

Exceptions

std::out_of_range	

Warning

Throws an Out Of Range exception if the list is empty or index is out of range when function is called.

Works the same way that the [] subscript operator does for arrays.

3.1.3.10 peek() [1/2]

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

Returns, but does not remove, the head of the list.

Template Parameters

T	Any data type or class.

Returns

The data located at the head of the list.

Exceptions

std::out_of_range	

Warning

Throws an Out Of Range exception if the list is empty when function is called.

3.1.3.11 peek() [2/2]

```
template<typename T >
T DLinkedList< T >::peek (
            const int index )
```

Returns, but does not remove, data from the list at the specified index.

Template Parameters

T	Any data type or class.

Parameters

index	Index at which to retrieve data from the list.

Returns

The data located at the specified index of the list.

Exceptions

std::out_of_range	

Warning

Throws an Out Of Range exception if the list is empty or index is out of range when function is called.

3.1.3.12 pop() [1/2]

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

Removes and returns the head of the list.

Template Parameters

T	Any data type or class.

Returns

The removed data.

Exceptions

std::out_of_range	

Warning

Throws an Out Of Range exception if the list is empty when function is called.

3.1.3.13 pop() [2/2]

```
template<typename T >
T DLinkedList< T >::pop (
            const int index )
```

Removes and returns data from the list at the specified index.

Template Parameters

T	Any data type or class.

Parameters

index	Index at which to retrieve and remove data from the list.

Returns

The removed data.

Exceptions

std::out_of_range	

Warning

Throws an Out Of Range exception if the list is empty or index is out of range when function is called.

3.1.3.14 remove()

```
template<typename T >
void DLinkedList< T >::remove (
            const int index )
```

Removes data from the list at the specified index.

Template Parameters

T	Any data type or class.

Parameters

index	Index at which to remove data from the list.

3.1.3.15 size()

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

Returns the size of the list.

Template Parameters

T	Any data type or class.

Returns

The size of the list.

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

- DLinkedList.hpp

3.2 Node< T > Struct Template Reference

The Node struct is meant to hold the data and pointers to the previous and next list element.

```
#include <DLinkedList.hpp>
```

Public Member Functions

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

Public Attributes

- T data
- Node< T > * next
- Node< T > * previous

Friends

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

3.2.1 Detailed Description

template<typename T>
struct Node< T >

The Node struct is meant to hold the data and pointers to the previous and next list element.

Template Parameters

T	Any data type or class.

3.2.2 Constructor & Destructor Documentation

3.2.2.1 Node() [1/2]

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

Copy Constructor.

3.2.2.2 Node() [2/2]

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

Move Constructor.

3.2.2.3 ~Node()

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

Struct Destructor.

3.2.3 Member Data Documentation

3.2.3.1 data

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

The data.

3.2.3.2 next

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

Pointer to the next node in the list.

3.2.3.3 previous

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

Pointer to the previous node in the list.

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

- DLinkedList.hpp

Chapter 4

File Documentation

4.1 DLinkedList.hpp File Reference

A generic Doubly Linked List data structure.

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

Classes

- struct Node< T >

  The Node struct is meant to hold the data and pointers to the previous and next list element.
- class DLinkedList< T >

  A generic Doubly Linked List class.

Functions

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

  Output stream operator.

4.1.1 Detailed Description

A generic Doubly Linked List data structure.

Author

Al Timofeyev

Date

January 3, 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 DLinkedList< T > & list )
```

Output stream operator.

Template Parameters

T	Any data type or class.

Parameters

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

Prints the list elements to the specified output stream.

Note

Any class or data type used with this linked list 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 linked list class NEEDS to implement its own operator<<.

Index