ProjectAlpha

Generated by Doxygen 1.9.1

Chapter 1

Hierarchical Index

1.1 Class Hierarchy

This inheritance list is sorted roughly, but not completely, alphabetically:

Chapter 2

Class Index

2.1 Class List

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

Chapter 3

Class Documentation

3.1 Bondinator_dList< NodeType > Class Template Reference

The Bondinator class for DoublyLinkedList.

`#include <doubly_linked_list.hpp>`

Public Member Functions

- **Bondinator_dList** (DListNodeptr< NodeType > element=nullptr)
- Bondinator_dList & **operator++** ()
- Bondinator_dList **operator++** (int)
- bool **operator==** (const Bondinator_dList &other) const
- bool **operator!=** (const Bondinator_dList &other) const
- reference **operator*** () const

3.1.1 Detailed Description

template<typename NodeType>
class Bondinator_dList< NodeType >

The Bondinator class for DoublyLinkedList.

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

- include/datatypes/doubly_linked_list/doubly_linked_list.hpp
- include/datatypes/doubly_linked_list/doubly_linked_list.ipp

3.2 Bondinator_List< NodeType > Class Template Reference

The Bondinator class for List.

`#include <list.hpp>`

Public Member Functions

- **Bondinator_List** (ListNodeptr< NodeType > element=nullptr)
- Bondinator_List & **operator++** ()
- Bondinator_List **operator++** (int)
- bool **operator==** (const Bondinator_List &other) const
- bool **operator!=** (const Bondinator_List &other) const
- reference **operator*** () const
- ListNodeptr< NodeType > **getNodePtr** () const

3.2.1 Detailed Description

template<typename NodeType>
class Bondinator_List< NodeType >

The Bondinator class for List.

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

- include/datatypes/list/list.hpp
- include/datatypes/list/list.ipp

3.3 Bondinator_pQueue< NodeType > Class Template Reference

The Bondinator class for PriorityQueue.

`#include <priority_queue.hpp>`

Public Member Functions

- **Bondinator_pQueue** (NodeSharedPtr< NodeType > element=nullptr)
- Bondinator_pQueue & **operator++** ()
- Bondinator_pQueue **operator++** (int)
- bool **operator==** (const Bondinator_pQueue &other) const
- bool **operator!=** (const Bondinator_pQueue &other) const
- reference **operator*** () const
- NodeType & **getData** () const

3.3.1 Detailed Description

template<typename NodeType>
class Bondinator_pQueue< NodeType >

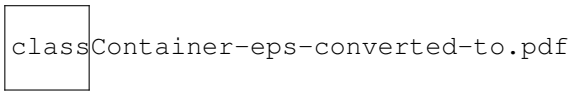
The Bondinator class for PriorityQueue.

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

- include/datatypes/priority_queue/priority_queue.hpp
- include/datatypes/priority_queue/priority_queue.ipp

3.4 Container Class Reference

Inheritance diagram for Container:



Public Member Functions

- size_t size () const
  Get the size of the container.

Protected Attributes

- size_t **container_size** = 0

3.4.1 Member Function Documentation

3.4.1.1 size()

```
size_t Container::size ( ) const  [inline]
```

Get the size of the container.

Returns

An amount of the elements

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

- include/container/container.hpp

3.5 DoublyLinkedList< NodeType > Class Template Reference

The DoublyLinkedList class.

```
#include <doubly_linked_list.hpp>
```

Inheritance diagram for DoublyLinkedList< NodeType >:

Public Member Functions

- DListNodeptr< NodeType > get_first () const override

  Get the first element.
- DListNodeptr< NodeType > get_last () const override

  Get the last element.
- DListNodeptr< NodeType > insert_front (const NodeType &insert_element) override

  Insert an element at the front.
- DListNodeptr< NodeType > insert_after (const DListNodeptr< NodeType > &pred, const NodeType &insert_element) override

  Insert an element after given one.
- DListNodeptr< NodeType > remove (const DListNodeptr< NodeType > &element) override

  Remove an element.
- DListNodeptr< NodeType > next (const DListNodeptr< NodeType > &insert_element) const override

  Get the Data element after given one.
- DListNodeptr< NodeType > prev (const DListNodeptr< NodeType > &element) const override

  Get the previos element after given one.
- DListNodeptr< NodeType > find (const NodeType &element) override

  Find an element in a DoublyLinkedList.
- Bondinator_dList< NodeType > **begin** ()
- Bondinator_dList< NodeType > **end** ()

Additional Inherited Members

3.5.1 Detailed Description

template<typename NodeType>
class DoublyLinkedList< NodeType >

The DoublyLinkedList class.

3.5.2 Member Function Documentation

3.5.2.1 find()

```
template<typename NodeType >
DListNodeptr< NodeType > DoublyLinkedList< NodeType >::find (
            const NodeType & element ) [override], [virtual]
```

Find an element in a DoublyLinkedList.

Parameters

element	The element to find

Returns

The element if it's contained

Implements DoublyLinkedListInterface< NodeType >.

3.5.2.2 get_first()

```
template<typename NodeType >
DListNodeptr< NodeType > DoublyLinkedList< NodeType >::get_first ( ) const  [override], [virtual]
```

Get the first element.

Returns

The first element

Implements DoublyLinkedListInterface< NodeType >.

3.5.2.3 get_last()

```
template<typename NodeType >
DListNodeptr< NodeType > DoublyLinkedList< NodeType >::get_last ( ) const  [override], [virtual]
```

Get the last element.

Returns

The last element

Implements DoublyLinkedListInterface< NodeType >.

3.5.2.4 insert_after()

```
template<typename NodeType >
DListNodeptr< NodeType > DoublyLinkedList< NodeType >::insert_after (
            const DListNodeptr< NodeType > & pred,
            const NodeType & insert_element )  [override], [virtual]
```

Insert an element after given one.

Parameters

pred	The element after which to insert the another one
insert_element	The element to insert

Returns

The element that was inserted

Implements DoublyLinkedListInterface< NodeType >.

3.5.2.5 insert_front()

```
template<typename NodeType >
DListNodeptr< NodeType > DoublyLinkedList< NodeType >::insert_front (
            const NodeType & insert_element )  [override], [virtual]
```

Insert an element at the front.

Parameters

insert_element	The element to insert

Returns

The front element

Implements DoublyLinkedListInterface< NodeType >.

3.5.2.6 next()

```
template<typename NodeType >
DListNodeptr< NodeType > DoublyLinkedList< NodeType >::next (
            const DListNodeptr< NodeType > & insert_element ) const  [override], [virtual]
```

Get the Data element after given one.

Parameters

insert_element	The element after Get next one

Returns

The next element after given

Implements DoublyLinkedListInterface< NodeType >.

3.5.2.7 prev()

```
template<typename NodeType >
DListNodeptr< NodeType > DoublyLinkedList< NodeType >::prev (
            const DListNodeptr< NodeType > & element ) const  [override], [virtual]
```

Get the previos element after given one.

Parameters

element	The element from which the previos element to Get

Returns

The previos element from given one

Implements DoublyLinkedListInterface< NodeType >.

3.5.2.8 remove()

```
template<typename NodeType >
DListNodeptr< NodeType > DoublyLinkedList< NodeType >::remove (
            const DListNodeptr< NodeType > & element )  [override], [virtual]
```

Remove an element.

Parameters

element	The element to remove

Returns

The element that was removed

Implements DoublyLinkedListInterface< NodeType >.

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

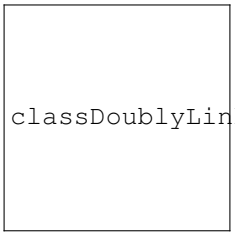
- include/datatypes/doubly_linked_list/doubly_linked_list.hpp
- include/datatypes/doubly_linked_list/doubly_linked_list.ipp

3.6 DoublyLinkedListInterface< NodeType > Class Template Reference

The DoublyLinkedList Interface.

```
#include <doubly_linked_list.hpp>
```

Inheritance diagram for DoublyLinkedListInterface< NodeType >:



Public Member Functions

- virtual DListNodeptr< NodeType > get_first () const =0
  Get the first element.
- virtual DListNodeptr< NodeType > get_last () const =0
  Get the last element.
- virtual DListNodeptr< NodeType > insert_front (const NodeType &insert_element)=0
  Insert an element at the front.
- virtual DListNodeptr< NodeType > insert_after (const DListNodeptr< NodeType > &pred, const NodeType &insert_element)=0
  Insert an element after given one.
- virtual DListNodeptr< NodeType > remove (const DListNodeptr< NodeType > &element)=0
  Remove an element.
- virtual DListNodeptr< NodeType > next (const DListNodeptr< NodeType > &insert_element) const =0
  Get the Data element after given one.
- virtual DListNodeptr< NodeType > prev (const DListNodeptr< NodeType > &element) const =0
  Get the previos element after given one.
- virtual DListNodeptr< NodeType > find (const NodeType &element)=0
  Find an element in a DoublyLinkedList.

Additional Inherited Members

3.6.1 Detailed Description

template<typename NodeType>
class DoublyLinkedListInterface< NodeType >

The DoublyLinkedList Interface.

3.6.2 Member Function Documentation

3.6.2.1 find()

```
template<typename NodeType >
virtual DListNodeptr<NodeType> DoublyLinkedListInterface< NodeType >::find (
            const NodeType & element )  [pure virtual]
```

Find an element in a DoublyLinkedList.

Parameters

element	The element to find

Returns

The element if it's contained

Implemented in DoublyLinkedList< NodeType >.

3.6.2.2 get_first()

```
template<typename NodeType >
virtual DListNodeptr<NodeType> DoublyLinkedListInterface< NodeType >::get_first ( ) const
[pure virtual]
```

Get the first element.

Returns

The first element

Implemented in DoublyLinkedList< NodeType >.

3.6.2.3 get_last()

```
template<typename NodeType >
virtual DListNodeptr<NodeType> DoublyLinkedListInterface< NodeType >::get_last ( ) const
[pure virtual]
```

Get the last element.

Returns

The last element

Implemented in DoublyLinkedList< NodeType >.

3.6.2.4 insert_after()

```
template<typename NodeType >
virtual DListNodeptr<NodeType> DoublyLinkedListInterface< NodeType >::insert_after (
            const DListNodeptr< NodeType > & pred,
            const NodeType & insert_element ) [pure virtual]
```

Insert an element after given one.

Parameters

pred	The element after which to insert the another one
insert_element	The element to insert

Returns

The element that was inserted

Implemented in DoublyLinkedList< NodeType >.

3.6.2.5 insert_front()

```
template<typename NodeType >
virtual DListNodeptr<NodeType> DoublyLinkedListInterface< NodeType >::insert_front (
            const NodeType & insert_element )  [pure virtual]
```

Insert an element at the front.

Parameters

insert_element	The element to insert

Returns

The front element

Implemented in DoublyLinkedList< NodeType >.

3.6.2.6 next()

```
template<typename NodeType >
virtual DListNodeptr<NodeType> DoublyLinkedListInterface< NodeType >::next (
            const DListNodeptr< NodeType > & insert_element ) const  [pure virtual]
```

Get the Data element after given one.

Parameters

insert_element	The element after Get next one

Returns

The next element after given

Implemented in DoublyLinkedList< NodeType >.

3.6.2.7 prev()

```
template<typename NodeType >
virtual DListNodeptr<NodeType> DoublyLinkedListInterface< NodeType >::prev (
            const DListNodeptr< NodeType > & element ) const  [pure virtual]
```

Get the previos element after given one.

Parameters

element	The element from which the previos element to Get

Returns

The previos element from given one

Implemented in DoublyLinkedList< NodeType >.

3.6.2.8 remove()

```
template<typename NodeType >
virtual DListNodeptr<NodeType> DoublyLinkedListInterface< NodeType >::remove (
            const DListNodeptr< NodeType > & element )  [pure virtual]
```

Remove an element.

Parameters

element	The element to remove

Returns

The element that was removed

Implemented in DoublyLinkedList< NodeType >.

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

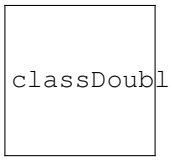
- include/interfaces/doubly_linked_list/doubly_linked_list.hpp

3.7 DoublyLinkedListNode< NodeType > Class Template Reference

The DoublyLinkedListNode class for DListNodeptr.

`#include <doubly_linked_listnode.hpp>`

Inheritance diagram for DoublyLinkedListNode< NodeType >:



Public Member Functions

- **DoublyLinkedListNode** (NodeType data)
- NodeType & getData () override

  Get the Data of the element.
- void setData (NodeType &data) override

  Set the Data.
- std::shared_ptr< DoublyLinkedListNodeInterface< NodeType > > getNext () const override

  Get the next element of the current one.
- std::weak_ptr< DoublyLinkedListNodeInterface< NodeType > > getPred () const override

  Get the previous element of the current one.
- void setNext (std::shared_ptr< DoublyLinkedListNodeInterface< NodeType >> nextNode) override

  Set the next element to current.
- void setPred (std::weak_ptr< DoublyLinkedListNodeInterface< NodeType >> predNode) override

  Set the previous element ftom current.
- bool isHead () override

  Check if the node is the head.
- bool isTail () override

  Check if the node is the tail.

3.7.1 Detailed Description

template<typename NodeType>
class DoublyLinkedListNode< NodeType >

The DoublyLinkedListNode class for DListNodeptr.

3.7.2 Member Function Documentation

3.7.2.1 getData()

```
template<typename NodeType >
NodeType & DoublyLinkedListNode< NodeType >::getData ( )  [override], [virtual]
```

Get the Data of the element.

Returns

Data of the element

Implements DoublyLinkedListNodeInterface< NodeType >.

3.7.2.2 getNext()

```
template<typename NodeType >
std::shared_ptr< DoublyLinkedListNodeInterface< NodeType > > DoublyLinkedListNode< NodeType
>::getNext ( ) const  [override], [virtual]
```

Get the next element of the current one.

Returns

The next element

Implements DoublyLinkedListNodeInterface< NodeType >.

3.7.2.3 getPred()

```
template<typename NodeType >
std::weak_ptr< DoublyLinkedListNodeInterface< NodeType > > DoublyLinkedListNode< NodeType
>::getPred ( ) const  [override], [virtual]
```

Get the previous element of the current one.

Returns

The next element

Implements DoublyLinkedListNodeInterface< NodeType >.

3.7.2.4 isHead()

```
template<typename NodeType >
bool DoublyLinkedListNode< NodeType >::isHead ( )  [override], [virtual]
```

Check if the node is the head.

Returns

whether the node is head or not

Implements DoublyLinkedListNodeInterface< NodeType >.

3.7.2.5 isTail()

```
template<typename NodeType >
bool DoublyLinkedListNode< NodeType >::isTail ( )  [override], [virtual]
```

Check if the node is the tail.

Returns

whether the node is tail or not

Implements DoublyLinkedListNodeInterface< NodeType >.

3.7.2.6 setData()

```
template<typename NodeType >
void DoublyLinkedListNode< NodeType >::setData (
            NodeType & data )  [override], [virtual]
```

Set the Data.

Parameters

data	The Data to insert

Implements DoublyLinkedListNodeInterface< NodeType >.

3.7.2.7 setNext()

```
template<typename NodeType >
void DoublyLinkedListNode< NodeType >::setNext (
```

```
            std::shared_ptr< DoublyLinkedListNodeInterface< NodeType >> nextNode )  [override],
[virtual]
```

Set the next element to current.

Parameters

nextNode	The element to set

Implements DoublyLinkedListNodeInterface< NodeType >.

3.7.2.8 setPred()

```
template<typename NodeType >
void DoublyLinkedListNode< NodeType >::setPred (
            std::weak_ptr< DoublyLinkedListNodeInterface< NodeType >> predNode )  [override],
[virtual]
```

Set the previous element ftom current.

Parameters

predNode	The element to set

Implements DoublyLinkedListNodeInterface< NodeType >.

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

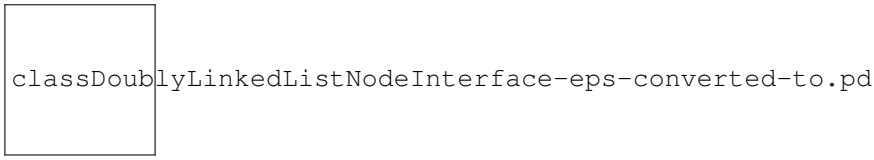
- include/datatypes/doubly_linked_list/doubly_linked_listnode.hpp
- include/datatypes/doubly_linked_list/doubly_linked_listnode.ipp

3.8 DoublyLinkedListNodeInterface< NodeType > Class Template Reference

The DoublyLinkedListNode Interface.

```
#include <doubly_linked_listnode.hpp>
```

Inheritance diagram for DoublyLinkedListNodeInterface< NodeType >:



Public Member Functions

- virtual NodeType & getData ()=0

  Get the Data of the element.
- virtual void setData (NodeType &data)=0

  Set the Data.
- virtual std::shared_ptr< DoublyLinkedListNodeInterface< NodeType > > getNext () const =0

  Get the next element of the current one.
- virtual std::weak_ptr< DoublyLinkedListNodeInterface< NodeType > > getPred () const =0

  Get the previous element of the current one.
- virtual void setNext (std::shared_ptr< DoublyLinkedListNodeInterface< NodeType >> nextNode)=0

  Set the next element to current.
- virtual void setPred (std::weak_ptr< DoublyLinkedListNodeInterface< NodeType >> predNode)=0

  Set the previous element ftom current.
- virtual bool isHead ()=0

  Check if the node is the head.
- virtual bool isTail ()=0

  Check if the node is the tail.

3.8.1 Detailed Description

template<typename NodeType>
class DoublyLinkedListNodeInterface< NodeType >

The DoublyLinkedListNode Interface.

3.8.2 Member Function Documentation

3.8.2.1 getData()

```
template<typename NodeType >
virtual NodeType& DoublyLinkedListNodeInterface< NodeType >::getData ( )  [pure virtual]
```

Get the Data of the element.

Returns

Data of the element

Implemented in DoublyLinkedListNode< NodeType >.

3.8.2.2 getNext()

```
template<typename NodeType >
virtual std::shared_ptr<DoublyLinkedListNodeInterface<NodeType> > DoublyLinkedListNodeInterface<
NodeType >::getNext ( ) const  [pure virtual]
```

Get the next element of the current one.

Returns

The next element

Implemented in DoublyLinkedListNode< NodeType >.

3.8.2.3 getPred()

```
template<typename NodeType >
virtual std::weak_ptr<DoublyLinkedListNodeInterface<NodeType> > DoublyLinkedListNodeInterface<
NodeType >::getPred ( ) const  [pure virtual]
```

Get the previous element of the current one.

Returns

The next element

Implemented in DoublyLinkedListNode< NodeType >.

3.8.2.4 isHead()

```
template<typename NodeType >
virtual bool DoublyLinkedListNodeInterface< NodeType >::isHead ( )  [pure virtual]
```

Check if the node is the head.

Returns

whether the node is head or not

Implemented in DoublyLinkedListNode< NodeType >.

3.8.2.5 isTail()

```
template<typename NodeType >
virtual bool DoublyLinkedListNodeInterface< NodeType >::isTail ( )  [pure virtual]
```

Check if the node is the tail.

Returns

whether the node is tail or not

Implemented in DoublyLinkedListNode< NodeType >.

3.8.2.6 setData()

```
template<typename NodeType >
virtual void DoublyLinkedListNodeInterface< NodeType >::setData (
            NodeType & data )  [pure virtual]
```

Set the Data.

Parameters

data	The Data to insert

Implemented in DoublyLinkedListNode< NodeType >.

3.8.2.7 setNext()

```
template<typename NodeType >
virtual void DoublyLinkedListNodeInterface< NodeType >::setNext (
            std::shared_ptr< DoublyLinkedListNodeInterface< NodeType >> nextNode )  [pure
virtual]
```

Set the next element to current.

Parameters

nextNode	The element to set

Implemented in DoublyLinkedListNode< NodeType >.

3.8.2.8 setPred()

```
template<typename NodeType >
```

```
virtual void DoublyLinkedListNodeInterface< NodeType >::setPred (
            std::weak_ptr< DoublyLinkedListNodeInterface< NodeType >> predNode ) [pure
virtual]
```

Set the previous element ftom current.

Parameters

predNode	The element to set

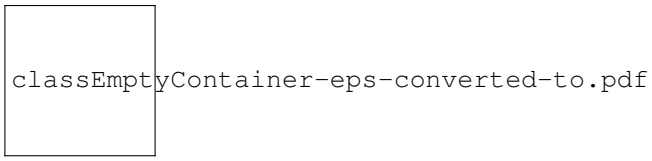
Implemented in DoublyLinkedListNode< NodeType >.

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

- include/interfaces/doubly_linked_list/doubly_linked_listnode.hpp

3.9 EmptyContainer Class Reference

Inheritance diagram for EmptyContainer:

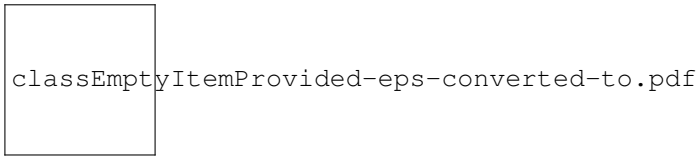


The documentation for this class was generated from the following file:

- include/exceptions/datatypes.hpp

3.10 EmptyItemProvided Class Reference

Inheritance diagram for EmptyItemProvided:



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

- include/exceptions/datatypes.hpp

3.11 Hashtable< NodeType > Class Template Reference

The Hashtable Class.

`#include <hashtable.hpp>`

Inheritance diagram for Hashtable< NodeType >:

Public Member Functions

- **Hashtable** (std::function< size_t(const NodeType &)> custom_hash)
- void insert (const NodeType &insert_element)

  Insert an element.
- bool find (const NodeType &element)

  Find an element in a set.
- void remove (const NodeType &element)

  Remove an element.
- const std::vector< DoublyLinkedList< NodeType > > & **getHashtabelle** () const
- const size_t & **getNum_buckets** () const

Additional Inherited Members

3.11.1 Detailed Description

template<typename NodeType>
class Hashtable< NodeType >

The Hashtable Class.

3.11.2 Member Function Documentation

3.11.2.1 find()

```
template<typename NodeType >
bool Hashtable< NodeType >::find (
            const NodeType & element )  [virtual]
```

Find an element in a set.

Parameters

element	The element to find

Returns

The element if it's contained

Implements HashtableInterface< NodeType >.

3.11.2.2 insert()

```
template<typename NodeType >
void Hashtable< NodeType >::insert (
            const NodeType & insert_element )  [virtual]
```

Insert an element.

Parameters

insert_element	The element to insert

Implements HashtableInterface< NodeType >.

3.11.2.3 remove()

```
template<typename NodeType >
void Hashtable< NodeType >::remove (
            const NodeType & element )  [virtual]
```

Remove an element.

Parameters

element	The element to remove

Implements HashtableInterface< NodeType >.

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

- include/datatypes/hashtable/hashtable.hpp
- include/datatypes/hashtable/hashtable.ipp

3.12 HashtableInterface< NodeType > Class Template Reference

The Hashtable Interface.

`#include <hashtable.hpp>`

Inheritance diagram for HashtableInterface< NodeType >:

Public Member Functions

- virtual void insert (const NodeType &insert_element)=0
  Insert an element.
- virtual bool find (const NodeType &element)=0
  Find an element in a set.
- virtual void remove (const NodeType &element)=0
  Remove an element.

Additional Inherited Members

3.12.1 Detailed Description

template<typename NodeType>
class HashtableInterface< NodeType >

The Hashtable Interface.

3.12.2 Member Function Documentation

3.12.2.1 find()

```
template<typename NodeType >
virtual bool HashtableInterface< NodeType >::find (
            const NodeType & element )  [pure virtual]
```

Find an element in a set.

Parameters

element	The element to find

Returns

The element if it's contained

Implemented in Hashtable< NodeType >.

3.12.2.2 insert()

```
template<typename NodeType >
virtual void HashtableInterface< NodeType >::insert (
            const NodeType & insert_element )  [pure virtual]
```

Insert an element.

Parameters

insert_element	The element to insert

Implemented in Hashtable< NodeType >.

3.12.2.3 remove()

```
template<typename NodeType >
virtual void HashtableInterface< NodeType >::remove (
            const NodeType & element )  [pure virtual]
```

Remove an element.

Parameters

element	The element to remove

Implemented in Hashtable< NodeType >.

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

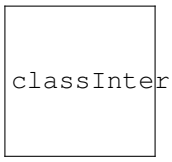
- include/interfaces/hashtable/hashtable.hpp

3.13 Interface_Stack< NodeType > Class Template Reference

The Stack Interface.

`#include <stack.hpp>`

Inheritance diagram for Interface_Stack< NodeType >:



Public Member Functions

- virtual void push (NodeType data)=0
  Insert an element at the front.
- virtual NodeType pop ()=0
  Remove the first element.

3.13.1 Detailed Description

template<typename NodeType>
class Interface_Stack< NodeType >

The Stack Interface.

3.13.2 Member Function Documentation

3.13.2.1 pop()

```
template<typename NodeType >
virtual NodeType Interface_Stack< NodeType >::pop ( )  [pure virtual]
```

Remove the first element.

Returns

The first element

Implemented in Stack< NodeType >.

3.13.2.2 push()

```
template<typename NodeType >
virtual void Interface_Stack< NodeType >::push (
            NodeType data )  [pure virtual]
```

Insert an element at the front.

Parameters

data	The element to insert

Implemented in Stack< NodeType >.

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

- include/interfaces/stack/stack.hpp

3.14 ItemAlreadyExisting Class Reference

Inheritance diagram for ItemAlreadyExisting:

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

- include/exceptions/datatypes.hpp

3.15 ItemNotFound Class Reference

Inheritance diagram for ItemNotFound:

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

- include/exceptions/datatypes.hpp

3.16 List< NodeType > Class Template Reference

The List Class.

`#include <list.hpp>`

Inheritance diagram for List< NodeType >:

Public Member Functions

- ListNodeptr< NodeType > insert_front (const NodeType &insert_element) override
  Insert an element at the front.
- ListNodeptr< NodeType > insert_after (const ListNodeptr< NodeType > &pred, const NodeType &insert_element) override
  Insert an element after given one.
- ListNodeptr< NodeType > remove_front () override
  Remove the first element.
- ListNodeptr< NodeType > remove_after (const ListNodeptr< NodeType > &pred) override
  Remove an element after given one.
- ListNodeptr< NodeType > next (const ListNodeptr< NodeType > &insert_element) const override
  Get the next element after given one.
- Bondinator_List< NodeType > **begin** ()
- Bondinator_List< NodeType > **end** ()

Additional Inherited Members

3.16.1 Detailed Description

template<typename NodeType>
class List< NodeType >

The List Class.

3.16.2 Member Function Documentation

3.16.2.1 insert_after()

```
template<typename NodeType >
ListNodeptr< NodeType > List< NodeType >::insert_after (
            const ListNodeptr< NodeType > & pred,
            const NodeType & insert_element )  [override], [virtual]
```

Insert an element after given one.

Parameters

pred	The element after which to insert the another one
insert_element	The element to insert

Returns

The element that was inserted

Implements ListInterface< NodeType >.

3.16.2.2 insert_front()

```
template<class NodeType >
ListNodeptr< NodeType > List< NodeType >::insert_front (
            const NodeType & insert_element )  [override], [virtual]
```

Insert an element at the front.

Parameters

insert_element	The element to insert

Returns

The front element

Implements ListInterface< NodeType >.

3.16.2.3 next()

```
template<typename NodeType >
ListNodeptr< NodeType > List< NodeType >::next (
            const ListNodeptr< NodeType > & insert_element ) const  [override], [virtual]
```

Get the next element after given one.

Parameters

insert_element	The element after which to get next one

Returns

The next element after given

Implements ListInterface< NodeType >.

3.16.2.4 remove_after()

```
template<typename NodeType >
ListNodeptr< NodeType > List< NodeType >::remove_after (
            const ListNodeptr< NodeType > & pred )  [override], [virtual]
```

Remove an element after given one.

Parameters

pred	The element after which to remove the another one

Returns

The element that was removed

Implements ListInterface< NodeType >.

3.16.2.5 remove_front()

```
template<typename NodeType >
ListNodeptr< NodeType > List< NodeType >::remove_front  [override], [virtual]
```

Remove the first element.

Returns

The old first element

Implements ListInterface< NodeType >.

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

- include/datatypes/list/list.hpp
- include/datatypes/list/list.ipp

3.17 ListInterface< NodeType > Class Template Reference

The List Interface.

`#include <list.hpp>`

Inheritance diagram for ListInterface< NodeType >:

Public Member Functions

- virtual ListNodeptr< NodeType > insert_front (const NodeType &insert_element)=0

  Insert an element at the front.
- virtual ListNodeptr< NodeType > insert_after (const ListNodeptr< NodeType > &pred, const NodeType &insert_element)=0

  Insert an element after given one.
- virtual ListNodeptr< NodeType > remove_front ()=0

  Remove the first element.
- virtual ListNodeptr< NodeType > remove_after (const ListNodeptr< NodeType > &pred)=0

  Remove an element after given one.
- virtual ListNodeptr< NodeType > next (const ListNodeptr< NodeType > &insert_element) const =0

  Get the next element after given one.

Additional Inherited Members

3.17.1 Detailed Description

template<typename NodeType>
class ListInterface< NodeType >

The List Interface.

3.17.2 Member Function Documentation

3.17.2.1 insert_after()

```
template<typename NodeType >
virtual ListNodeptr<NodeType> ListInterface< NodeType >::insert_after (
            const ListNodeptr< NodeType > & pred,
            const NodeType & insert_element ) [pure virtual]
```

Insert an element after given one.

Parameters

pred	The element after which to insert the another one
insert_element	The element to insert

Returns

The element that was inserted

Implemented in List< NodeType >.

3.17.2.2 insert_front()

```
template<typename NodeType >
virtual ListNodeptr<NodeType> ListInterface< NodeType >::insert_front (
            const NodeType & insert_element )  [pure virtual]
```

Insert an element at the front.

Parameters

insert_element	The element to insert

Returns

The front element

Implemented in List< NodeType >.

3.17.2.3 next()

```
template<typename NodeType >
virtual ListNodeptr<NodeType> ListInterface< NodeType >::next (
            const ListNodeptr< NodeType > & insert_element ) const  [pure virtual]
```

Get the next element after given one.

Parameters

insert_element	The element after which to get next one

Returns

The next element after given

Implemented in List< NodeType >.

3.17.2.4 remove_after()

```
template<typename NodeType >
virtual ListNodeptr<NodeType> ListInterface< NodeType >::remove_after (
            const ListNodeptr< NodeType > & pred )  [pure virtual]
```

Remove an element after given one.

Parameters

pred	The element after which to remove the another one

Returns

The element that was removed

Implemented in List< NodeType >.

3.17.2.5 remove_front()

```
template<typename NodeType >
virtual ListNodeptr<NodeType> ListInterface< NodeType >::remove_front ( )  [pure virtual]
```

Remove the first element.

Returns

The old first element

Implemented in List< NodeType >.

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

- include/interfaces/list/list.hpp

3.18 ListNode< NodeType > Class Template Reference

The ListNode Class.

```
#include <listnode.hpp>
```

Inheritance diagram for ListNode< NodeType >:

Public Member Functions

- **ListNode** (NodeType data)
- NodeType & getData () override

  Get the Data of the element.
- void setData (NodeType &data) override

  Set the Data.
- std::shared_ptr< ListNodeInterface< NodeType > > getNext () const override

  Get the next element of the current one.
- void setNext (const std::shared_ptr< ListNodeInterface< NodeType >> &nextNode) override

  Set the next element to current.

3.18.1 Detailed Description

template<typename NodeType>
class ListNode< NodeType >

The ListNode Class.

3.18.2 Member Function Documentation

3.18.2.1 getData()

```
template<typename NodeType >
NodeType & ListNode< NodeType >::getData ( )  [override], [virtual]
```

Get the Data of the element.

Returns

Data of the element

Implements ListNodeInterface< NodeType >.

3.18.2.2 getNext()

```
template<typename NodeType >
std::shared_ptr< ListNodeInterface< NodeType > > ListNode< NodeType >::getNext ( ) const
[override], [virtual]
```

Get the next element of the current one.

Returns

The next element

Implements ListNodeInterface< NodeType >.

3.18.2.3 setData()

```
template<typename NodeType >
void ListNode< NodeType >::setData (
            NodeType & data )  [override], [virtual]
```

Set the Data.

Parameters

data	The Data to insert

Implements ListNodeInterface< NodeType >.

3.18.2.4 setNext()

```
template<typename NodeType >
void ListNode< NodeType >::setNext (
            const std::shared_ptr< ListNodeInterface< NodeType >> & nextNode )  [override],
[virtual]
```

Set the next element to current.

Parameters

nextNode	The element to set

Implements ListNodeInterface< NodeType >.

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

- include/datatypes/list/listnode.hpp
- include/datatypes/list/listnode.ipp

3.19 ListNodeInterface< NodeType > Class Template Reference

The ListNode Interface.

```
#include <listnode.hpp>
```

Inheritance diagram for ListNodeInterface< NodeType >:

Public Member Functions

- virtual NodeType & getData ()=0
  Get the Data of the element.
- virtual void setData (NodeType &data)=0
  Set the Data.
- virtual std::shared_ptr< ListNodeInterface< NodeType > > getNext () const =0
  Get the next element of the current one.
- virtual void setNext (const std::shared_ptr< ListNodeInterface< NodeType >> &nextNode)=0
  Set the next element to current.

3.19.1 Detailed Description

template<typename NodeType>
class ListNodeInterface< NodeType >

The ListNode Interface.

3.19.2 Member Function Documentation

3.19.2.1 getData()

```
template<typename NodeType >
virtual NodeType& ListNodeInterface< NodeType >::getData ( )  [pure virtual]
```

Get the Data of the element.

Returns

Data of the element

Implemented in ListNode< NodeType >.

3.19.2.2 getNext()

```
template<typename NodeType >
virtual std::shared_ptr<ListNodeInterface<NodeType> > ListNodeInterface< NodeType >::getNext
( ) const  [pure virtual]
```

Get the next element of the current one.

Returns

The next element

Implemented in ListNode< NodeType >.

3.19.2.3 setData()

```
template<typename NodeType >
virtual void ListNodeInterface< NodeType >::setData (
            NodeType & data )  [pure virtual]
```

Set the Data.

Parameters

data	The Data to insert

Implemented in ListNode< NodeType >.

3.19.2.4 setNext()

```
template<typename NodeType >
virtual void ListNodeInterface< NodeType >::setNext (
            const std::shared_ptr< ListNodeInterface< NodeType >> & nextNode ) [pure virtual]
```

Set the next element to current.

Parameters

nextNode	The element to set

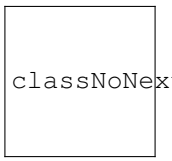
Implemented in ListNode< NodeType >.

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

- include/interfaces/list/listnode.hpp

3.20 NoNextItem Class Reference

Inheritance diagram for NoNextItem:

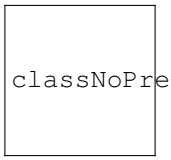


The documentation for this class was generated from the following file:

- include/exceptions/datatypes.hpp

3.21 NoPrevItem Class Reference

Inheritance diagram for NoPrevItem:



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

- include/exceptions/datatypes.hpp

3.22 PriorityQueue< NodeType > Class Template Reference

The PriorityQueue Class.

`#include <priority_queue.hpp>`

Inheritance diagram for PriorityQueue< NodeType >:

Public Member Functions

- const NodeSharedPtr< NodeType > get_root () const override
  Get the root.
- void insert (size_t prio, NodeType data) override
  Insert an element in tree.
- PriorityQueueNodeSharedPtr< NodeType > remove () override
  Remove an element.
- Bondinator_pQueue< NodeType > **begin** ()
- Bondinator_pQueue< NodeType > **end** ()

Additional Inherited Members

3.22.1 Detailed Description

template<typename NodeType>
class PriorityQueue< NodeType >

The PriorityQueue Class.

3.22.2 Member Function Documentation

3.22.2.1 get_root()

```
template<typename NodeType >
const NodeSharedPtr< NodeType > PriorityQueue< NodeType >::get_root ( ) const  [override],
[virtual]
```

Get the root.

Returns

Root

Implements PriorityQueueInterface< NodeType >.

3.22.2.2 insert()

```
template<typename NodeType >
void PriorityQueue< NodeType >::insert (
            size_t prio,
            NodeType data )  [override], [virtual]
```

Insert an element in tree.

Parameters

prio	Priority of the element
data	Data of the element

Implements PriorityQueueInterface< NodeType >.

3.22.2.3 remove()

```
template<typename NodeType >
PriorityQueueNodeSharedPtr< NodeType > PriorityQueue< NodeType >::remove ( )  [override],
[virtual]
```

Remove an element.

Parameters

pred	The element to remove

Implements PriorityQueueInterface< NodeType >.

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

- include/datatypes/priority_queue/priority_queue.hpp
- include/datatypes/priority_queue/priority_queue.ipp

3.23 PriorityQueueInterface< NodeType > Class Template Reference

The PriorityQueue Interface.

`#include <priority_queue.hpp>`

Inheritance diagram for PriorityQueueInterface< NodeType >:

Public Member Functions

- virtual const NodeSharedPtr< NodeType > get_root () const =0
  Get the root.
- virtual void insert (size_t prio, NodeType data)=0
  Insert an element in tree.
- virtual PriorityQueueNodeSharedPtr< NodeType > remove ()=0
  Remove an element.

Additional Inherited Members

3.23.1 Detailed Description

template<typename NodeType>
class PriorityQueueInterface< NodeType >

The PriorityQueue Interface.

3.23.2 Member Function Documentation

3.23.2.1 get_root()

```
template<typename NodeType >
virtual const NodeSharedPtr<NodeType> PriorityQueueInterface< NodeType >::get_root ( ) const
[pure virtual]
```

Get the root.

Returns

Root

Implemented in PriorityQueue< NodeType >.

3.23.2.2 insert()

```
template<typename NodeType >
virtual void PriorityQueueInterface< NodeType >::insert (
            size_t prio,
            NodeType data )  [pure virtual]
```

Insert an element in tree.

Parameters

prio	Priority of the element
data	Data of the element

Implemented in PriorityQueue< NodeType >.

3.23.2.3 remove()

```
template<typename NodeType >
virtual PriorityQueueNodeSharedPtr<NodeType> PriorityQueueInterface< NodeType >::remove ( )
[pure virtual]
```

Remove an element.

Parameters

pred	The element to remove

Implemented in PriorityQueue< NodeType >.

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

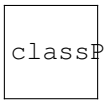
- include/interfaces/priority_queue/priority_queue.hpp

3.24 PriorityQueueNode< NodeType > Class Template Reference

The PriorityQueueNode Class.

`#include <priority_queuenode.hpp>`

Inheritance diagram for PriorityQueueNode< NodeType >:



Public Member Functions

- **PriorityQueueNode** (NodeType prio, NodeType data)
- **PriorityQueueNode** (NodeType prio, NodeType data, std::weak_ptr< PriorityQueueNodeInterface< Node↩ Type >> parent)
- NodeType & getPrio () override

  Get a priority of a node.
- NodeType & getData () override

  Get a data of a node.
- const NodeType get_left_child_data () const override

  Get a data of a left child of the node.
- const NodeType get_right_child_data () const override

  Get a data of a right child of the node.
- std::shared_ptr< PriorityQueueNodeInterface< NodeType > > get_left_child () override

  Get the left child of the node.
- std::shared_ptr< PriorityQueueNodeInterface< NodeType > > get_right_child () override

  Get the right child of the node.
- std::weak_ptr< PriorityQueueNodeInterface< NodeType > > get_parent () const override

  Get parent of a node.
- void set_parent (std::weak_ptr< PriorityQueueNodeInterface< NodeType >> predNode) override

  Get a parent of a node.
- void set_left_child (std::shared_ptr< PriorityQueueNodeInterface< NodeType >> nextNode) override

  Set the left child of the node.
- void set_right_child (std::shared_ptr< PriorityQueueNodeInterface< NodeType >> nextNode) override

  Set the right child of the node.
- void del_left_child () override

  Delete the left child of the node.
- void del_right_child () override

  Delete the right child of the node.

3.24.1 Detailed Description

template<typename NodeType>
class PriorityQueueNode< NodeType >

The PriorityQueueNode Class.

3.24.2 Member Function Documentation

3.24.2.1 get_left_child()

```
template<typename NodeType >
std::shared_ptr< PriorityQueueNodeInterface< NodeType > > PriorityQueueNode< NodeType >↩
::get_left_child ( )  [override], [virtual]
```

Get the left child of the node.

Returns

The left child of the Node

Implements PriorityQueueNodeInterface< NodeType >.

3.24.2.2 get_left_child_data()

```
template<typename NodeType >
const NodeType PriorityQueueNode< NodeType >::get_left_child_data ( ) const  [override], [virtual]
```

Get a data of a left child of the node.

Returns

The left child's Data

Implements PriorityQueueNodeInterface< NodeType >.

3.24.2.3 get_parent()

```
template<typename NodeType >
std::weak_ptr< PriorityQueueNodeInterface< NodeType > > PriorityQueueNode< NodeType >::get↩
_parent ( ) const  [override], [virtual]
```

Get parent of a node.

Returns

Parent of a node

Implements PriorityQueueNodeInterface< NodeType >.

3.24.2.4 get_right_child()

```
template<typename NodeType >
std::shared_ptr< PriorityQueueNodeInterface< NodeType > > PriorityQueueNode< NodeType >↩
::get_right_child ( )  [override], [virtual]
```

Get the right child of the node.

Returns

The right child of the Node

Implements PriorityQueueNodeInterface< NodeType >.

3.24.2.5 get_right_child_data()

```
template<typename NodeType >
const NodeType PriorityQueueNode< NodeType >::get_right_child_data ( ) const  [override],
[virtual]
```

Get a data of a right child of the node.

Returns

The right child's Data

Implements PriorityQueueNodeInterface< NodeType >.

3.24.2.6 getData()

```
template<typename NodeType >
NodeType & PriorityQueueNode< NodeType >::getData ( )  [override], [virtual]
```

Get a data of a node.

Returns

Data of a node

Implements PriorityQueueNodeInterface< NodeType >.

3.24.2.7 getPrio()

```
template<typename NodeType >
NodeType & PriorityQueueNode< NodeType >::getPrio ( )  [override], [virtual]
```

Get a priority of a node.

Returns

Priority of a node

Implements PriorityQueueNodeInterface< NodeType >.

3.24.2.8 set_left_child()

```
template<typename NodeType >
void PriorityQueueNode< NodeType >::set_left_child (
            std::shared_ptr< PriorityQueueNodeInterface< NodeType >> nextNode )  [override],
[virtual]
```

Set the left child of the node.

Parameters

nextNode	Node to set a left child

Implements PriorityQueueNodeInterface< NodeType >.

3.24.2.9 set_parent()

```
template<typename NodeType >
void PriorityQueueNode< NodeType >::set_parent (
            std::weak_ptr< PriorityQueueNodeInterface< NodeType >> predNode )  [override],
[virtual]
```

Get a parent of a node.

Parameters

predNode	Node to set a parent

Implements PriorityQueueNodeInterface< NodeType >.

3.24.2.10 set_right_child()

```
template<typename NodeType >
void PriorityQueueNode< NodeType >::set_right_child (
            std::shared_ptr< PriorityQueueNodeInterface< NodeType >> nextNode )  [override],
[virtual]
```

Set the right child of the node.

Parameters

nextNode	Node to set a right child

Implements PriorityQueueNodeInterface< NodeType >.

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

- include/datatypes/priority_queue/priority_queuenode.hpp
- include/datatypes/priority_queue/priority_queuenode.ipp

3.25 PriorityQueueNodeInterface< NodeType > Class Template Reference

The PriorityQueueNode Interface.

```
#include <priority_queuenode.hpp>
```

Inheritance diagram for PriorityQueueNodeInterface< NodeType >:

Public Member Functions

- virtual NodeType & getPrio ()=0

  Get a priority of a node.
- virtual NodeType & getData ()=0

  Get a data of a node.
- virtual const NodeType get_left_child_data () const =0

  Get a data of a left child of the node.
- virtual const NodeType get_right_child_data () const =0

  Get a data of a right child of the node.
- virtual std::weak_ptr< PriorityQueueNodeInterface< NodeType > > get_parent () const =0

  Get parent of a node.
- virtual void set_parent (std::weak_ptr< PriorityQueueNodeInterface< NodeType >> predNode)=0

  Get a parent of a node.

- virtual std::shared_ptr< PriorityQueueNodeInterface< NodeType > > get_left_child ()=0
  Get the left child of the node.
- virtual std::shared_ptr< PriorityQueueNodeInterface< NodeType > > get_right_child ()=0
  Get the right child of the node.
- virtual void set_left_child (std::shared_ptr< PriorityQueueNodeInterface< NodeType >> nextNode)=0
  Set the left child of the node.
- virtual void set_right_child (std::shared_ptr< PriorityQueueNodeInterface< NodeType >> nextNode)=0
  Set the right child of the node.
- virtual void del_left_child ()=0
  Delete the left child of the node.
- virtual void del_right_child ()=0
  Delete the right child of the node.

3.25.1 Detailed Description

template<typename NodeType>
class PriorityQueueNodeInterface< NodeType >

The PriorityQueueNode Interface.

3.25.2 Member Function Documentation

3.25.2.1 get_left_child()

```
template<typename NodeType >
virtual std::shared_ptr<PriorityQueueNodeInterface<NodeType> > PriorityQueueNodeInterface<
NodeType >::get_left_child ( )  [pure virtual]
```

Get the left child of the node.

Returns

The left child of the Node

Implemented in PriorityQueueNode< NodeType >.

3.25.2.2 get_left_child_data()

```
template<typename NodeType >
virtual const NodeType PriorityQueueNodeInterface< NodeType >::get_left_child_data ( ) const
[pure virtual]
```

Get a data of a left child of the node.

Returns

The left child's Data

Implemented in PriorityQueueNode< NodeType >.

3.25.2.3 get_parent()

```
template<typename NodeType >
virtual std::weak_ptr<PriorityQueueNodeInterface<NodeType> > PriorityQueueNodeInterface<
NodeType >::get_parent ( ) const  [pure virtual]
```

Get parent of a node.

Returns

Parent of a node

Implemented in PriorityQueueNode< NodeType >.

3.25.2.4 get_right_child()

```
template<typename NodeType >
virtual std::shared_ptr<PriorityQueueNodeInterface<NodeType> > PriorityQueueNodeInterface<
NodeType >::get_right_child ( )  [pure virtual]
```

Get the right child of the node.

Returns

The right child of the Node

Implemented in PriorityQueueNode< NodeType >.

3.25.2.5 get_right_child_data()

```
template<typename NodeType >
virtual const NodeType PriorityQueueNodeInterface< NodeType >::get_right_child_data ( ) const
[pure virtual]
```

Get a data of a right child of the node.

Returns

The right child's Data

Implemented in PriorityQueueNode< NodeType >.

3.25.2.6 getData()

```
template<typename NodeType >
virtual NodeType& PriorityQueueNodeInterface< NodeType >::getData ( )  [pure virtual]
```

Get a data of a node.

Returns

Data of a node

Implemented in PriorityQueueNode< NodeType >.

3.25.2.7 getPrio()

```
template<typename NodeType >
virtual NodeType& PriorityQueueNodeInterface< NodeType >::getPrio ( )  [pure virtual]
```

Get a priority of a node.

Returns

Priority of a node

Implemented in PriorityQueueNode< NodeType >.

3.25.2.8 set_left_child()

```
template<typename NodeType >
virtual void PriorityQueueNodeInterface< NodeType >::set_left_child (
            std::shared_ptr< PriorityQueueNodeInterface< NodeType >> nextNode )  [pure virtual]
```

Set the left child of the node.

Parameters

nextNode	Node to set a left child

Implemented in PriorityQueueNode< NodeType >.

3.25.2.9 set_parent()

```
template<typename NodeType >
virtual void PriorityQueueNodeInterface< NodeType >::set_parent (
            std::weak_ptr< PriorityQueueNodeInterface< NodeType >> predNode )  [pure virtual]
```

Get a parent of a node.

Parameters

predNode	Node to set a parent

Implemented in PriorityQueueNode< NodeType >.

3.25.2.10 set_right_child()

```
template<typename NodeType >
virtual void PriorityQueueNodeInterface< NodeType >::set_right_child (
            std::shared_ptr< PriorityQueueNodeInterface< NodeType >> nextNode )  [pure virtual]
```

Set the right child of the node.

Parameters

nextNode	Node to set a right child

Implemented in PriorityQueueNode< NodeType >.

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

- include/interfaces/priority_queue/priority_queuenode.hpp

3.26 Queue< NodeType > Class Template Reference

The Queue Class.

```
#include <queue.hpp>
```

Inheritance diagram for Queue< NodeType >:

Public Member Functions

- void push (const NodeType &insert_element) override
  Insert an element at the end.
- NodeType pop () override
  Remove the first element.
- NodeType front () const override
  Get the first element.
- NodeType back () const override
  Get the last element.

Additional Inherited Members

3.26.1 Detailed Description

template<typename NodeType>
class Queue< NodeType >

The Queue Class.

3.26.2 Member Function Documentation

3.26.2.1 back()

```
template<typename NodeType >
NodeType Queue< NodeType >::back ( ) const  [override], [virtual]
```

Get the last element.

Returns

The last element

Implements QueueInterface< NodeType >.

3.26.2.2 front()

```
template<typename NodeType >
NodeType Queue< NodeType >::front ( ) const  [override], [virtual]
```

Get the first element.

Returns

The first element

Implements QueueInterface< NodeType >.

3.26.2.3 pop()

```
template<typename NodeType >
NodeType Queue< NodeType >::pop ( )  [override], [virtual]
```

Remove the first element.

Returns

The first element

Implements QueueInterface< NodeType >.

3.26.2.4 push()

```
template<typename NodeType >
void Queue< NodeType >::push (
            const NodeType & insert_element )  [override], [virtual]
```

Insert an element at the end.

Parameters

insert_element	The element to insert

Implements QueueInterface< NodeType >.

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

- include/datatypes/queue/queue.hpp
- include/datatypes/queue/queue.ipp

3.27 QueueInterface< NodeType > Class Template Reference

The Queue Interface.

```
#include <queue.hpp>
```

Inheritance diagram for QueueInterface< NodeType >:

Public Member Functions

- virtual void push (const NodeType &insert_element)=0
  Insert an element at the end.
- virtual NodeType pop ()=0
  Remove the first element.
- virtual NodeType front () const =0
  Get the first element.
- virtual NodeType back () const =0
  Get the last element.

Additional Inherited Members

3.27.1 Detailed Description

template<typename NodeType>
class QueueInterface< NodeType >

The Queue Interface.

3.27.2 Member Function Documentation

3.27.2.1 back()

```
template<typename NodeType >
virtual NodeType QueueInterface< NodeType >::back ( ) const  [pure virtual]
```

Get the last element.

Returns

The last element

Implemented in Queue< NodeType >.

3.27.2.2 front()

```
template<typename NodeType >
virtual NodeType QueueInterface< NodeType >::front ( ) const  [pure virtual]
```

Get the first element.

Returns

The first element

Implemented in Queue< NodeType >.

3.27.2.3 pop()

```
template<typename NodeType >
virtual NodeType QueueInterface< NodeType >::pop ( )  [pure virtual]
```

Remove the first element.

Returns

The first element

Implemented in Queue< NodeType >.

3.27.2.4 push()

```
template<typename NodeType >
virtual void QueueInterface< NodeType >::push (
            const NodeType & insert_element )  [pure virtual]
```

Insert an element at the end.

Parameters

insert_element	The element to insert

Implemented in Queue< NodeType >.

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

- include/interfaces/queue/queue.hpp

3.28 SetInterface< NodeType > Class Template Reference

The Set Interface.

```
#include <set.hpp>
```

Public Member Functions

- virtual void insert (const NodeType &insert_element)=0
  Insert an element.
- virtual void remove (const NodeType &element)=0
  Remove an element.
- virtual bool is_contained (const NodeType &element) const =0
  Check if an element is contained.
- virtual SetNodeptr< NodeType > find (const NodeType &element) const =0
  Find an element in a set.

3.28.1 Detailed Description

template<typename NodeType>
class SetInterface< NodeType >

The Set Interface.

3.28.2 Member Function Documentation

3.28.2.1 find()

```
template<typename NodeType >
virtual SetNodeptr<NodeType> SetInterface< NodeType >::find (
            const NodeType & element ) const  [pure virtual]
```

Find an element in a set.

Parameters

element	The element to find

Returns

The element if it's contained

3.28.2.2 insert()

```
template<typename NodeType >
virtual void SetInterface< NodeType >::insert (
            const NodeType & insert_element )  [pure virtual]
```

Insert an element.

Parameters

insert_element	The element to insert

3.28.2.3 is_contained()

```
template<typename NodeType >
virtual bool SetInterface< NodeType >::is_contained (
            const NodeType & element ) const  [pure virtual]
```

Check if an element is contained.

Parameters

pred	The element to check

Returns

True, if an element is contained; else - False

3.28.2.4 remove()

```
template<typename NodeType >
virtual void SetInterface< NodeType >::remove (
            const NodeType & element )  [pure virtual]
```

Remove an element.

Parameters

pred	The element to remove

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

- include/interfaces/set/set.hpp

3.29 SetNodeInterface< NodeType > Class Template Reference

The SetNode Interface.

```
#include <setnode.hpp>
```

Public Member Functions

- virtual NodeType & getData ()=0
  Get the Data of the element.
- virtual void setData (NodeType &data)=0
  Set the Data.
- virtual std::shared_ptr< SetNode< NodeType > > getNext () const =0
  Get the next element of the current one.
- virtual void setNext (const std::shared_ptr< SetNode< NodeType >> &nextNode)=0
  Set the next element to current.

3.29.1 Detailed Description

template<typename NodeType>
class SetNodeInterface< NodeType >

The SetNode Interface.

3.29.2 Member Function Documentation

3.29.2.1 getData()

```
template<typename NodeType >
virtual NodeType& SetNodeInterface< NodeType >::getData ( )  [pure virtual]
```

Get the Data of the element.

Returns

Data of the element

3.29.2.2 getNext()

```
template<typename NodeType >
virtual std::shared_ptr<SetNode<NodeType> > SetNodeInterface< NodeType >::getNext ( ) const
[pure virtual]
```

Get the next element of the current one.

Returns

The next element

3.29.2.3 setData()

```
template<typename NodeType >
virtual void SetNodeInterface< NodeType >::setData (
            NodeType & data )  [pure virtual]
```

Set the Data.

Parameters

data	The Data to insert

3.29.2.4 setNext()

```
template<typename NodeType >
virtual void SetNodeInterface< NodeType >::setNext (
            const std::shared_ptr< SetNode< NodeType >> & nextNode )  [pure virtual]
```

Set the next element to current.

Parameters

nextNode	The element to set

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

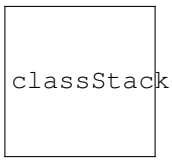
- include/interfaces/set/setnode.hpp

3.30 Stack< NodeType > Class Template Reference

The Stack Class.

```
#include <stack.hpp>
```

Inheritance diagram for Stack< NodeType >:



Public Member Functions

- void push (NodeType data) override
  Insert an element at the front.
- NodeType pop () override
  Remove the first element.

Additional Inherited Members

3.30.1 Detailed Description

template<typename NodeType>
class Stack< NodeType >

The Stack Class.

3.30.2 Member Function Documentation

3.30.2.1 pop()

```
template<typename NodeType >
NodeType Stack< NodeType >::pop  [override], [virtual]
```

Remove the first element.

Returns

The first element

Implements Interface_Stack< NodeType >.

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

- include/datatypes/stack/stack.hpp
- include/datatypes/stack/stack.ipp