Project 2.1

Revision 1

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:

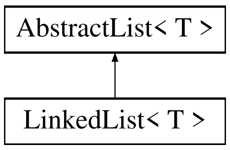
Chapter 3

Class Documentation

3.1 LinkedList< T > Class Template Reference

```
#include <LinkedList.hpp>
```

Inheritance diagram for LinkedList< T >:



Public Member Functions

- LinkedList (const LinkedList &rhs)
- LinkedList & operator= (LinkedList rhs)
- bool isEmpty () const
- int getLength () const
- bool insert (int newPosition, const T &newEntry)
- bool remove (int position)
- void clear ()
- T getEntry (int position) const
- T setEntry (int position, const T &newValue)
- void swap (LinkedList &lhs, LinkedList &rhs)

3.1.1 Detailed Description

template<typename T>
class LinkedList< T >

This is a LinkedList class. It uses Nodes to store data.

3.1.2 Constructor & Destructor Documentation

3.1.2.1 LinkedList()

```
template<typename T >
LinkedList< T >::LinkedList (
             const LinkedList< T > & rhs )
```

This is the copy constructor. It make a copy of the parameter. It is also used by the operator= in the copy-swap paradigm.

Parameters

rhs	- the LinkedList we are copying during construction

3.1.3 Member Function Documentation

3.1.3.1 clear()

```
template<typename T >
void LinkedList< T >::clear ( )
```

Removes all entries from this list.

Postcondition

The list contains no entries and the count of items is 0.

3.1.3.2 getEntry()

```
template<typename T >
T LinkedList< T >::getEntry (
             int position ) const
```

Gets the entry at the given position in this list.

Precondition

1 <= position <= getLength().

Postcondition

The desired entry has been returned.

Parameters

position	The list position of the desired entry.

Returns

The entry at the given position.

3.1.3.3 getLength()

```
template<typename T >
int LinkedList< T >::getLength ( ) const
```

Gets the current number of entries in this list.

Returns

The integer number of entries currently in the list.

3.1.3.4 insert()

```
template<typename T >
bool LinkedList< T >::insert (
             int newPosition,
             const T & newEntry )
```

Inserts an entry into this list at a given position.

Precondition

None.

Postcondition

If 1 <= position <= getLength() + 1 and the insertion is successful, newEntry is at the given position in the list, other entries are renumbered accordingly, and the returned value is true.

Parameters

newPosition	The list position at which to insert new↩Entry.
newEntry	The entry to insert into the list.

Returns

True if the insertion is successful, or false if not.

3.1.3.5 isEmpty()

```
template<typename T >
bool LinkedList< T >::isEmpty ( ) const
```

Sees whether this list is empty.

Returns

True if the list is empty; otherwise returns false.

3.1.3.6 operator=()

```
template<typename T >
LinkedList& LinkedList< T >::operator= (
              LinkedList< T > rhs )
```

This is the assignment operator. It uses the copy-swap paradigm to create a copy of the parameter

Parameters

rhs	- the LinkedList we are assigning to this

Returns

a reference to the list that was copied into, a.k.a. ∗this

3.1.3.7 remove()

```
template<typename T >
bool LinkedList< T >::remove (
              int position )
```

Removes the entry at a given position from this list.

Precondition

None.

Postcondition

If 1 <= position <= getLength() and the removal is successful, the entry at the given position in the list is removed, other items are renumbered accordingly, and the returned value is true.

Parameters

position	The list position of the entry to remove.

Returns

True if the removal is successful, or false if not.

3.1.3.8 setEntry()

```
template<typename T >
T LinkedList< T >::setEntry (
             int position,
             const T & newValue )
```

Sets the entry at the given position in this list with the new value.

Precondition

1 <= position <= getLength().

Postcondition

The value at the given position has new value

Parameters

position	The list position of the entry to set the new value
newVAlue	The new value to set at the givien position.

Returns

The replaced entry.

3.1.3.9 swap()

```
template<typename T >
void LinkedList< T >::swap (
             LinkedList< T > & lhs,
             LinkedList< T > & rhs )
```

This is the swap method. It will swap the internals of the two lists. Notably it is used in the operator= to implement the copy swap paradigm. It is also used by other C++ paradigms.

Parameters

lhs	- the LinkedList on the left...Left Hand Side (lhs)
rhs	- the LinkedList on the right...Right Hand Side (rhs)

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

- LinkedList.hpp

Index