Sequence< Key, Info > Class Template Reference

Basic singly-lined list implementation. More...

Classes

class **iterator**

Public Member Functions

	Sequence ()
	Sequence (const **Sequence**< Key, Info > &x)
	~Sequence ()
Sequence< Key, Info > &	**operator=** (const **Sequence**< Key, Info > &x)
iterator	**begin** () const
iterator	**end** () const
iterator	**find** (const Key &vKey, int occurrence=1) const
void	**clear** ()
bool	**empty** () const
void	**push_front** (const Key &key, const Info &info)
void	**push_front** (const **iterator** &it)
bool	**push_after** (const Key &key, const Info &info, const Key &where, int occurrence=1)
bool	**pop_front** ()
bool	**remove** (const Key &key, int occurrence=1)

Friends

template<typename A , typename B >

std::ostream & **operator<<** (std::ostream &os, const **Sequence**< A, B > &x)

Detailed Description

template<typename Key, typename Info>
class Sequence< Key, Info >

Basic singly-lined list implementation.

Constructor & Destructor Documentation

◆ Sequence() [1/2]

template<typename Key , typename Info >

Sequence< Key, Info >::**Sequence**

Constructs the list.

◆ Sequence() [2/2]

template<typename Key , typename Info >

Sequence< Key, Info >::**Sequence** (const **Sequence**< Key, Info > & x)

Copy constuctor.

◆ ~Sequence()

template<typename Key , typename Info >

Sequence< Key, Info >::**~Sequence**

Destructs the list

Member Function Documentation

◆ begin()

template<typename Key , typename Info >

iterator Sequence< Key, Info >::begin ( ) const inline

Iterator to first element of the list.

◆ clear()

template<typename Key , typename Info >

void **Sequence**< Key, Info >::clear

Clears all of the elements of the list.

◆ empty()

template<typename Key , typename Info >

bool **Sequence**< Key, Info >::empty

Outputs information about the status of the elements.

Returns

: False - if there are elements in the list, True - otherwise.

◆ end()

template<typename Key , typename Info >

iterator Sequence< Key, Info >::end ( ) const inline

Iterator to last element of the list + 1.

◆ find()

template<typename Key , typename Info >

Sequence< Key, Info >::**iterator Sequence**< Key, Info >::find (const Key & vKey,
int occurrence = 1
) const

Method finding an element.

◆ operator=()

template<typename Key , typename Info >

Sequence< Key, Info > & **Sequence**< Key, Info >::operator= (const **Sequence**< Key, Info > & x)

Assign values to the container

◆ pop_front()

template<typename Key , typename Info >

bool **Sequence**< Key, Info >::pop_front

Deletes the first element of the list.

Returns

: True - first element was deleted. False - otherwise.

◆ push_after()

template<typename Key , typename Info >

```
bool Sequence< Key, Info >::push_after ( const Key &  key,
                                         const Info &  info,
                                         const Key &  where,
                                         int  occurrence = 1
                                        )
```

Pushes value and key after the given occurence number of Key.

Parameters

[in] **key** : Key to be inserted.

[in] **info** : Info to be inserted

[in] **where** : Element after which we want to insert.

[in] **occurrence** : Number of occurences of a where. 1 by default.

Returns

: True - if element has been added, False - otherwise.

◆ push_front() [1/2]

template<typename Key , typename Info >

void **Sequence**< Key, Info >::push_front (const **iterator** & it)

Pushes value and key to the front of the list.

Parameters

[in] **key** : Key

[in] **info** : Info

◆ push_front() [2/2]

template<typename Key , typename Info >

void **Sequence**< Key, Info >::push_front (const Key & key,
const Info & info
)

Pushes value and key to the front of the list.

Parameters

[in] **key** : Key

[in] **info** : Info

◆ remove()

template<typename Key , typename Info >

bool **Sequence**< Key, Info >::remove (const Key & key,
int occurrence = 1
)

Removes a given occurence of a Key

Parameters

[in] **key** : Key

[in] **occurrence** : Occurence of a key.

Returns

: True - element was deleted. False - otherwise.

Friends And Related Function Documentation

◆ operator<<

template<typename Key , typename Info >

template<typename A , typename B >

std::ostream& operator<< (std::ostream & os,
const **Sequence**< A, B > & x
)

friend

Overloading the << operator.