QLinkedList Class

template < typename T > class QLinkedList

QLinkedList类是一个提供链表的模板类。更多的...

Header:	**#include**
CMake:	find_package(Qt6 REQUIRED COMPONENTS Core5Compat) target_link_libraries(mytarget PRIVATE Qt6::Core5Compat)
qmake:	QT += core5compat

- 所有成员的列表，包括继承的成员
- QLinkedList 是隐式共享类的一部分。

注意：该类中的所有函数都是reentrant。

公共类型

class	**const_iterator**
class	**iterator**
	ConstIterator
	Iterator
	const_pointer
	const_reference
	const_reverse_iterator
	difference_type
	pointer
	reference
	reverse_iterator
	size_type
	value_type

公共职能

	QLinkedList()
	QLinkedList(const QLinkedList< T > &*other*)
	QLinkedList(std::initializer_list< T > *list*)
	QLinkedList(InputIterator *first*, InputIterator *last*)
	QLinkedList(QLinkedList< T > &&*other*)
	~QLinkedList()
void	**append**(const T &*value*)
T &	**back**()
const T &	**back**() const
QLinkedList::iterator	**begin**()
QLinkedList::const_iterator	**begin**() const
QLinkedList::const_iterator	**cbegin**() const
QLinkedList::const_iterator	**cend**() const
void	**clear**()
QLinkedList::const_iterator	**constBegin**() const
QLinkedList::const_iterator	**constEnd**() const
bool	**contains**(const T &*value*) const
int	**count**(const T &*value*) const
int	**count**() const
QLinkedList::const_reverse_iterator	**crbegin**() const
QLinkedList::const_reverse_iterator	**crend**() const
bool	**empty**() const
QLinkedList::iterator	**end**()
QLinkedList::const_iterator	**end**() const
bool	**endsWith**(const T &*value*) const
QLinkedList::iterator	**erase**(QLinkedList::iterator *pos*)
QLinkedList::iterator	**erase**(QLinkedList::iterator *begin*, QLinkedList::iterator *end*)
T &	**first**()

T &	**first**()
	QLinkedList()
const T &	**first**() const
T &	**front**()
const T &	**front**() const
QLinkedList::iterator	**insert**(QLinkedList::iterator *before*, const T &*value*)
bool	**isEmpty**() const
T &	**last**()
const T &	**last**() const
void	**pop_back**()
void	**pop_front**()
void	**prepend**(const T &*value*)
void	**push_back**(const T &*value*)
void	**push_front**(const T &*value*)
QLinkedList::reverse_iterator	**rbegin**()
QLinkedList::const_reverse_iterator	**rbegin**() const
int	**removeAll**(const T &*value*)
void	**removeFirst**()
void	**removeLast**()
bool	**removeOne**(const T &*value*)
QLinkedList::reverse_iterator	**rend**()
QLinkedList::const_reverse_iterator	**rend**() const
int	**size**() const
bool	**startsWith**(const T &*value*) const
void	**swap**(QLinkedList< T > &*other*)
T	**takeFirst**()
T	**takeLast**()
std::list< T >	**toStdList**() const
bool	**operator!=**(const QLinkedList< T > &*other*) const
QLinkedList< T >	**operator+**(const QLinkedList< T > &*other*) const
QLinkedList< T > &	**operator+=**(const QLinkedList< T > &*other*)

QLinkedList< T > &	operator+=(const QLinkedList< T > &*other*)
	QLinkedList()
QLinkedList< T > &	operator+=(const T &*value*)
QLinkedList< T > &	operator<<(const QLinkedList< T > &*other*)
QLinkedList< T > &	operator<<(const T &*value*)
QLinkedList< T > &	operator=(const QLinkedList< T > &*other*)
bool	operator==(const QLinkedList< T > &*other*) const

静态公共成员

QLinkedList< T >	**fromStdList(const std::list< T > &*list*)**

相关非成员

QDataStreamIfHasOStreamOperatorsContainer<QLinkedList< T > , T>	**operator<<(QDataStream &*out*, const QLinkedList< T > &*list*)**
QDataStreamIfHasIStreamOperatorsContainer<QLinkedList< T > , T>	operator>>(QDataStream &*in*, QLinkedList< T > &*list*)

详细说明

QLinkedList< T > 是 Qt 的泛型之一container classes。它存储值列表并提供基于迭代器的访问以及constant time插入和删除。

QList< T > 和 QLinkedList< T > 提供类似的功能。概述如下：

- 对于大多数目的来说，QList是正确使用的类。它基于索引的API比QLinkedList基于迭代器的API更方便。它的项目占据相邻的内存位置。它还可以扩展到可执行文件中更少的代码。
- 如果您需要一个真实的链表，并保证constant time在列表中间插入以及对项目而不是索引的迭代器，请使用QLinkedList。

下面是一个存储整数的 QLinkedList 和一个存储整数的 QLinkedList 的示例QTime价值观：

```
QLinkedList<int> integerList;
QLinkedList<QTime> timeList;
```

QLinkedList 存储项目列表。默认构造函数创建一个空列表。要将项目插入列表中，可以使用运算符<<()：

```
QLinkedList<QString> list;
list << "one" << "two" << "three";
// list: ["one", "two", "three"]
```

如果您想获取链接列表中的第一项或最后一项，请使用first（） 或者last()。如果要从列表的任一端删除项目，请使用removeFirst（） 或者removeLast()。如果要删除列表中给定值的所有出现，请使用removeAll()。

一个常见的要求是删除列表中的第一个或最后一个项目并对其执行某些操作。为此，QLinkedList提供了takeFirst（） 和takeLast()。这是一个循环，一次从列表中删除一个项目并调用delete它们：

```
QLinkedList<QWidget *> list;
...
while (!list.isEmpty())
    delete list.takeFirst();
```

QLinkedList 的值类型必须是assignable data type。这涵盖了大多数常用的数据类型，但是编译器不允许您将 QWidget 存储为值；相反，存储一个 QWidget *。一些功能有额外的要求；例如，contains（） 和removeAll() 期望值类型支持operator==()。这些要求是按功能记录的。

如果要在列表中间插入、修改或删除项目，则必须使用迭代器。QLinkedList 两者都提供Java-style iterators（ QLinkedListIterator 和 QMutableLinkedListIterator ） 和 STL-style iterators （ QLinkedList::const_iterator 和 QLinkedList::iterator）。有关详细信息，请参阅这些类的文档。

也可以看看QLinkedListIterator,QMutableLinkedListIterator， 和QList。

成员类型文档

QLinkedList::ConstIterator

Qt 风格的同义词QLinkedList::const_iterator。

QLinkedList::Iterator

Qt 风格的同义词QLinkedList::iterator。

QLinkedList::const_pointer

const T * 的类型定义。提供 STL 兼容性。

QLinkedList::const_reference

const T & 的 Typedef。提供 STL 兼容性。

QLinkedList::const_reverse_iterator

QLinkedList::const_reverse_iterator typedef 提供了一个 STL 风格的 const 反向迭代器QLinkedList。

它只是 的 typedef `std::reverse_iterator<QLinkedList::const_iterator>`。

警告：隐式共享容器上的迭代器的工作方式与 STL 迭代器不同。当迭代器在容器上处于活动状态时，您应该避免复制该容器。欲了解更多信息，请阅读Implicit sharing iterator problem。

也可以看看QLinkedList::rbegin(),QLinkedList::rend(),QLinkedList::reverse_iterator， 和QLinkedList::const_iterator。

QLinkedList::difference_type

ptrdiff_t 的类型定义。提供 STL 兼容性。

QLinkedList::pointer

T * 的类型定义。提供 STL 兼容性。

QLinkedList::reference

T & 的类型定义。提供 STL 兼容性。

QLinkedList::reverse_iterator

QLinkedList::reverse_iterator typedef 提供了一个 STL 风格的非常量反向迭代器QLinkedList。

它只是 的 typedef `std::reverse_iterator<QLinkedList::iterator>`。

警告：隐式共享容器上的迭代器的工作方式与 STL 迭代器不同。当迭代器在容器上处于活动状态时，您应该避免复制该容器。欲了解更多信息，请阅读Implicit sharing iterator problem。

也可以看看QLinkedList::rbegin(),QLinkedList::rend(),QLinkedList::const_reverse_iterator， 和QLinkedList::iterator。

QLinkedList::size_type

typedef 为 int。提供 STL 兼容性。

QLinkedList::value_type

T 的 Typedef。为 STL 兼容性而提供。

成员函数文档

QLinkedList::QLinkedList()

构造一个空列表。

QLinkedList::QLinkedList(const QLinkedList< T > &other)

构造一个副本*other*。

此操作发生在constant time，因为 QLinkedList 是implicitly shared。这使得从函数返回 QLinkedList 的速度非常快。如果共享实例被修改，它将被复制（写时复制），这需要linear time。

也可以看看operator=()。

QLinkedList::QLinkedList(std::initializer_list< T > list)

从 std::initializer_list 指定的构造列表*list*。

仅当编译器支持 C++11 初始值设定项列表时才启用此构造函数。

template < typename InputIterator> QLinkedList::QLinkedList(InputIterator first, InputIterator last)

使用迭代器范围 [中的内容构造一个列表*first*,*last*）。

的值类型`InputIterator`必须可转换为`T`。

QLinkedList::QLinkedList(QLinkedList< T > &&other)

移动构造一个 QLinkedList 实例，使其指向与*other*正在指着。

QLinkedList::~QLinkedList()

销毁列表。对该列表中的值以及该列表上的所有迭代器的引用都将变得无效。

void QLinkedList::append(const T &value)

刀片*value*在列表的末尾。

例子：

```
QLinkedList<QString> list;
list.append("one");
list.append("two");
list.append("three");
// list: ["one", "two", "three"]
```

这与 list.insert(相同end(),*value*) 。

也可以看看operator<<(),prepend （） ， 和insert()。

T &QLinkedList::back()

提供此函数是为了兼容 STL。它相当于last()。

const T &QLinkedList::back() const

这是一个过载功能。

QLinkedList::iterator QLinkedList::begin()

返回一个STL-style iterator指向列表中的第一项。

也可以看看constBegin （） 和end()。

QLinkedList::const_iterator QLinkedList::begin() const

这是一个过载功能。

QLinkedList::const_iterator QLinkedList::cbegin() const

返回一个常量STL-style iterator指向列表中的第一项。

也可以看看begin（） 和cend()。

QLinkedList::const_iterator QLinkedList::cend() const

返回一个常量STL-style iterator指向列表中最后一项之后的虚构项。

也可以看看cbegin（） 和end()。

void QLinkedList::clear()

删除列表中的所有项目。

也可以看看removeAll()。

QLinkedList::const_iterator QLinkedList::constBegin() const

返回一个常量STL-style iterator指向列表中的第一项。

也可以看看begin（） 和constEnd()。

QLinkedList::const_iterator QLinkedList::constEnd() const

返回一个常量STL-style iterator指向列表中最后一项之后的虚构项。

也可以看看constBegin（） 和end()。

bool QLinkedList::contains(const T &value) const

`true`如果列表包含出现则返回*value*; 否则返回`false`.

此函数要求值类型具有 的实现`operator==()`。

也可以看看QLinkedListIterator::findNext（） 和QLinkedListIterator::findPrevious()。

int QLinkedList::count(const T &value) const

返回出现的次数*value*在列表中。

此函数要求值类型具有 的实现`operator==()`。

也可以看看contains()。

int QLinkedList::count() const

与...一样size()。

QLinkedList::const_reverse_iterator QLinkedList::crbegin() const

返回一个常量STL-style反向迭代器以相反的顺序指向列表中的第一项。

也可以看看begin(),rbegin（），和rend()。

QLinkedList::const_reverse_iterator QLinkedList::crend() const

返回一个常量STL-style反向迭代器以相反的顺序指向列表中最后一项。

也可以看看end(),rend（），和rbegin()。

bool QLinkedList::empty() const

提供此函数是为了兼容 STL。它相当于isEmpty`true()`，如果列表为空则返回。

QLinkedList::iterator QLinkedList::end()

返回一个STL-style iterator指向列表中最后一项之后的虚构项。

也可以看看begin（） 和constEnd()。

QLinkedList::const_iterator QLinkedList::end() const

这是一个过载功能。

bool QLinkedList::endsWith(const T &value) const

如果列表`true`不为空且其最后一项等于*value*; 否则返回`false`.

也可以看看isEmpty（） 和last()。

QLinkedList::iterator QLinkedList::erase(QLinkedList::iterator pos)

删除迭代器指向的项*pos*从列表中，并返回一个迭代器到列表中的下一个项目（可能是end()）。

也可以看看insert()。

QLinkedList::iterator QLinkedList::erase(QLinkedList::iterator begin, QLinkedList::iterator end)

这是一个过载功能。

从中删除所有项目*begin*最多（但不包括）*end*。

T &QLinkedList::first()

返回对列表中第一项的引用。该函数假设列表不为空。

也可以看看last（） 和isEmpty()。

const T &QLinkedList::first() const

这是一个过载功能。

[static] QLinkedList< T > QLinkedList::fromStdList(const std::list< T > &list)

返回一个QLinkedList包含数据的对象*list*。中元素的顺序QLinkedList与中相同*list*。

例子：

```
std::list<double> stdlist;
list.push_back(1.2);
list.push_back(0.5);
list.push_back(3.14);

QLinkedList<double> list = QLinkedList<double>::fromStdList(stdlist);
```

也可以看看toStdList()。

T &QLinkedList::front()

提供此函数是为了兼容 STL。它相当于first()。

const T &QLinkedList::front() const

这是一个过载功能。

QLinkedList::iterator QLinkedList::insert(QLinkedList::iterator before, const T &value)

刀片*value*在迭代器指向的项前面*before*。返回一个指向插入项的迭代器。

也可以看看erase()。

bool QLinkedList::isEmpty() const

`true`如果列表不包含任何项目则返回；否则返回 false。

也可以看看size()。

T &QLinkedList::last()

返回对列表中最后一项的引用。该函数假设列表不为空。

也可以看看first（） 和isEmpty()。

const T &QLinkedList::last() const

这是一个过载功能。

void QLinkedList::pop_back()

提供此函数是为了兼容 STL。它相当于removeLast()。

void QLinkedList::pop_front()

提供此函数是为了兼容 STL。它相当于removeFirst()。

void QLinkedList::prepend(const T &value)

刀片*value*在列表的开头。

例子：

```
QLinkedList<QString> list;
list.prepend("one");
list.prepend("two");
list.prepend("three");
// list: ["three", "two", "one"]
```

这与 list.insert(相同begin(),*value*）。

也可以看看append（） 和insert()。

void QLinkedList::push_back(const T &value)

提供此函数是为了兼容 STL。它相当于附加（*value*）。

void QLinkedList::push_front(const T &value)

提供此函数是为了兼容 STL。它相当于前置(*value*）。

QLinkedList::reverse_iterator QLinkedList::rbegin()

返回一个STL-style反向迭代器以相反的顺序指向列表中的第一项。

也可以看看begin(),crbegin（），和rend()。

QLinkedList::const_reverse_iterator QLinkedList::rbegin() const

这是一个过载功能。

int QLinkedList::removeAll(const T &value)

删除所有出现的*value*在列表中。

例子：

```
QList<QString> list;
list << "sun" << "cloud" << "sun" << "rain";
list.removeAll("sun");
// list: ["cloud", "rain"]
```

此函数要求值类型具有 的实现`operator==()`。

也可以看看insert()。

void QLinkedList::removeFirst()

删除列表中的第一项。

这与擦除相同（begin()）。

也可以看看removeLast（） 和erase()。

void QLinkedList::removeLast()

删除列表中的最后一项。

也可以看看removeFirst（） 和erase()。

bool QLinkedList::removeOne(const T &value)

删除第一次出现的*value*在列表中。`true`成功回报；否则返回`false`.

例子：

```
QList<QString> list;
list << "sun" << "cloud" << "sun" << "rain";
list.removeOne("sun");
// list: ["cloud", "sun", "rain"]
```

此函数要求值类型具有 的实现`operator==()`。

也可以看看insert()。

QLinkedList::reverse_iterator QLinkedList::rend()

返回一个STL-style反向迭代器以相反的顺序指向列表中最后一项。

也可以看看end(),crend（），和rbegin()。

QLinkedList::const_reverse_iterator QLinkedList::rend() const

这是一个过载功能。

int QLinkedList::size() const

返回列表中的项目数。

也可以看看isEmpty（） 和count()。

bool QLinkedList::startsWith(const T &value) const

如果列表`true`不为空且其第一项等于*value*; 否则返回`false`.

也可以看看isEmpty（） 和first()。

void QLinkedList::swap(QLinkedList< T > &other)

掉期清单*other*有了这个清单。这个操作非常快并且永远不会失败。

T QLinkedList::takeFirst()

删除列表中的第一项并将其返回。

如果不使用返回值，removeFirst() 效率更高。

也可以看看takeLast（） 和removeFirst()。

T QLinkedList::takeLast()

删除列表中的最后一项并将其返回。

如果不使用返回值，removeLast() 效率更高。

也可以看看takeFirst（） 和removeLast()。

std::list< T > QLinkedList::toStdList() const

返回一个 std::list 对象，其中包含此对象中包含的数据QLinkedList。例子：

```
QLinkedList<double> list;
list << 1.2 << 0.5 << 3.14;

std::list<double> stdlist = list.toStdList();
```

也可以看看fromStdList()。

bool QLinkedList::operator!=(const QLinkedList< T > &other) const

返回`true`如果*other*不等于此列表；否则返回`false`.

如果两个列表包含相同顺序的相同值，则认为它们相等。

该函数需要值类型来实现`operator==()`。

也可以看看operator==()。

QLinkedList< T > QLinkedList::operator+(const QLinkedList< T > &other) const

返回一个列表，其中包含此列表中的所有项目，后跟*other*列表。

也可以看看operator+=()。

QLinkedList< T > &QLinkedList::operator+=(const QLinkedList< T > &other)

追加以下项目*other*list 到此列表并返回对此列表的引用。

也可以看看operator+ () 和append()。

QLinkedList< T > &QLinkedList::operator+=(const T &value)

这是一个过载功能。

追加*value*到列表中。

QLinkedList< T > &QLinkedList::operator<<(const QLinkedList< T > &other)

追加以下项目*other*list 到此列表并返回对此列表的引用。

也可以看看operator+= () 和append()。

QLinkedList< T > &QLinkedList::operator<<(const T &value)

这是一个过载功能。

追加*value*到列表中。

QLinkedList< T > &QLinkedList::operator=(const QLinkedList< T > &other)

分配*other*到此列表并返回对此列表的引用。

bool QLinkedList::operator==(const QLinkedList< T > &other) const

返回`true`如果*other*等于这个列表；否则返回 false。

如果两个列表包含相同顺序的相同值，则认为它们相等。

该函数需要值类型来实现`operator==()`。

也可以看看operator!=()。

相关非成员

template < typename T > QDataStreamIfHasOStreamOperatorsContainer<QLinkedList< T > , T> operator<<(QDataStream &out, const QLinkedList< T > &list)

写入链表*list*流式传输*out*。

该函数需要值类型来实现`operator<<()`。

也可以看看Format of the QDataStream operators。

template < typename T > QDataStreamIfHasIStreamOperatorsContainer<QLinkedList< T > , T> operator>>(QDataStream &in, QLinkedList< T > &list)

从流中读取链表*in*进入*list*。

该函数需要值类型来实现`operator>>()`。

也可以看看Format of the QDataStream operators。