QMapIterator Class

template <typename Key, typename T> class QMapIterator

QMapIterator 类提供了一个 Java 风格的 const 迭代器QMap。更多的...

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

- 所有成员的列表，包括继承的成员

公共职能

	QMapIterator(const QMap<Key, T> &*map*)
bool	**findNext**(const T &*value*)
bool	**findPrevious**(const T &*value*)
bool	**hasNext**() const
bool	**hasPrevious**() const
const Key &	**key**() const
QMapIterator::Item	**next**()
QMapIterator::Item	**peekNext**() const
QMapIterator::Item	**peekPrevious**() const
QMapIterator::Item	**previous**()
void	**toBack**()
void	**toFront**()
const T &	**value**() const
QMapIterator<Key, T> &	**operator=**(const QMap<Key, T> &*container*)

详细说明

QMap两者都有Java-style iterators和STL-style iterators。STL 风格的迭代器效率更高，应该是首选。

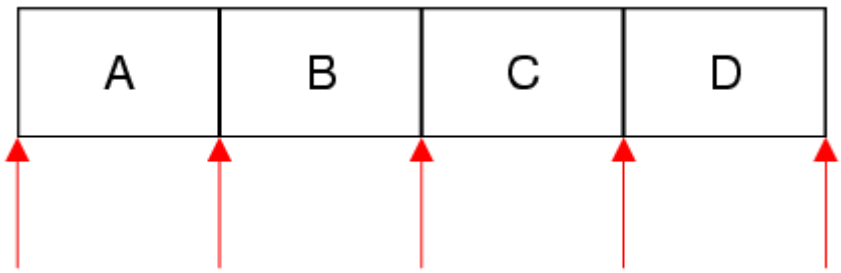
QMapIterator<Key, T> 允许您迭代QMap。如果您想在迭代地图时修改地图，请使用QMutableMapIterator反而。

QMapIterator 构造函数采用QMap作为论证。构造后，迭代器位于映射的最开始处（第一项之前）。以下是按顺序迭代所有元素的方法：

```
QMap<int, QWidget *> map;
...
QMapIterator<int, QWidget *> i(map);
while (i.hasNext()) {
    i.next();
    qDebug() << i.key() << ": " << i.value();
}
```

这next() 函数返回映射中的下一项并推进迭代器。这key （） 和value() 函数返回最后被跳过的项目的键和值。

与 STL 样式的迭代器不同，Java 样式的迭代器指向项*之间*而不是直接*指向*项。第一次致电next() 将迭代器前进到第一项和第二项之间的位置，并返回第一项；第二次致电next() 将迭代器前进到第二项和第三项之间的位置；等等。



以下是如何以相反的顺序迭代元素：

```
QMapIterator<int, QWidget *> i(map);
i.toBack();
while (i.hasPrevious()) {
    i.previous();
    qDebug() << i.key() << ": " << i.value();
}
```

如果您想查找特定值的所有出现位置，请使用findNext （） 或者findPrevious() 循环中。例如：

```
QMapIterator<int, QWidget *> i(map);
while (i.findNext(widget)) {
    qDebug() << "Found widget " << widget << " under key "
             << i.key();
}
```

可以在同一个映射上使用多个迭代器。如果在 QMapIterator 处于活动状态时修改了映射，则 QMapIterator 将继续迭代原始映射，忽略修改后的副本。

可以参考：QMutableMapIterator和QMap::const_iterator。

成员函数文档

QMapIterator::QMapIterator(const QMap<Key, T> &map)

构造一个迭代器用于遍历*map*。迭代器设置为位于映射的前面（第一项之前）。

可以参考： operator=()。

QMapIterator<Key, T> &QMapIterator::operator=(const QMap<Key, T> &container)

使迭代器运行*map*。迭代器设置为位于映射的前面（第一项之前）。

可以参考： toFront（） 和toBack()。

void QMapIterator::toFront()

将迭代器移动到容器的前面（第一项之前）。

可以参考： toBack（） 和next()。

void QMapIterator::toBack()

将迭代器移动到容器的后面（最后一项之后）。

可以参考： toFront（） 和previous()。

bool QMapIterator::hasNext() const

`true`如果迭代器前面至少有一项，即迭代器不在*容器*的后面，则返回；否则返回`false`.

可以参考： hasPrevious（） 和next()。

QMapIterator::Item QMapIterator::next()

返回下一项并将迭代器前进一个位置。

称呼key() 返回值来获取项目的键，以及value() 来获取值。

在位于容器后面的迭代器上调用此函数会导致未定义的结果。

可以参考： hasNext(),peekNext（），和previous()。

QMapIterator::Item QMapIterator::peekNext() const

返回下一项而不移动迭代器。

称呼key() 返回值来获取项目的键，以及value() 来获取值。

在位于容器后面的迭代器上调用此函数会导致未定义的结果。

可以参考： hasNext(),next（），和peekPrevious()。

bool QMapIterator::hasPrevious() const

`true`如果迭代器后面至少有一项，即迭代器不在*容器*的前面，则返回；否则返回`false`.

可以参考： hasNext（）和previous()。

QMapIterator::Item QMapIterator::previous()

返回前一项并将迭代器向后移动一个位置。

称呼key() 返回值来获取项目的键，以及value() 来获取值。

在位于容器前面的迭代器上调用此函数会导致未定义的结果。

可以参考： hasPrevious(),peekPrevious（），和next()。

QMapIterator::Item QMapIterator::peekPrevious() const

返回前一项而不移动迭代器。

称呼key() 返回值来获取项目的键，以及value() 来获取值。

在位于容器前面的迭代器上调用此函数会导致未定义的结果。

可以参考： hasPrevious(),previous（），和peekNext()。

const T &QMapIterator::value() const

返回使用遍历函数之一跳过的最后一项的值 (next(),previous(),findNext(),findPrevious())。

拨打电话后next（） 或者findNext(),value() 相当于peekPrevious()。value()。拨打电话后previous（） 或者findPrevious(),value() 相当于peekNext()。value()。

可以参考：key()。

const Key &QMapIterator::key() const

返回使用遍历函数之一跳过的最后一项的键（next(),previous(),findNext(),findPrevious())。

拨打电话后next（） 或者findNext(),key() 相当于peekPrevious()。key()。拨打电话后previous（） 或者findPrevious(),key() 相当于peekNext()。key()。

可以参考：value()。

bool QMapIterator::findNext(const T &value)

搜索次数*value*从当前迭代器位置开始向前。返回`true`（键，值）对是否具有值*value*被发现；否则返回`false`.

通话结束后，如果*value*找到后，迭代器就位于匹配项之后；否则，迭代器位于容器的后面。

可以参考：findPrevious()。

bool QMapIterator::findPrevious(const T &value)

搜索次数*value*从当前迭代器位置向后开始。返回`true`（键，值）对是否具有值*value*被发现；否则返回`false`.

通话结束后，如果*value*找到后，迭代器就位于匹配项之前；否则，迭代器位于容器的前面。

可以参考：findNext()。