2007--QList 类&QLinkedList 类--零声教育 vico 老师

一、QList 类

对于不同的数据类型，QList<T>采取不同的存储策略，存储策略如下：

- 如果 T 是一个指针类型或指针大小的基本类型（该基本类型占有的字节数和指针类型占有的字节数相同），QList<T>将数值直接存储在它的数组当中。
- 如果 QList<T>存储对象的指针，则该指针指向实际存储的对象。

【案例分析】

```
#include <QCoreApplication>

#include <QDebug>

int main(int argc, char *argv[])
{
    QCoreApplication a(argc, argv);

    // QList 类
    QList<int> qlist; // 初始化一个空的 QList<int>列表
    for(int i=0;i<10;i++)
        qlist.insert(qlist.end(),i+10);
    qDebug()<<qlist;

    // 通过 QList<int>::iterator 读写迭代器
    QList<int>::iterator x;
    qDebug()<<endl;
    qDebug()<<"Result1:";
    for(x=qlist.begin();x!=qlist.end();x++)
    {
        qDebug()<<(*x);
        *x=(*x)*10+6;
    }

    // 初始化一个 QList<int>const_iterator 只读迭代器
    qDebug()<<endl;
    qDebug()<<"Result1:";
```

```
    QList<int>::const_iterator qciter;
    // 输出列表所有的值
    for(qciter=qlist.constBegin();qciter!=qlist.constEnd();qciter++)
        qDebug()<<*qciter;

    // 向 qlist 添加元素
    qlist.append(666);
    QList<int>::iterator itr1;
    qDebug()<<endl;
    qDebug()<<"Result2:";
    for(itr1=qlist.begin();itr1!=qlist.end();itr1++)
        qDebug()<<*itr1;

    // 查询 qlist 当中元素
    qDebug()<<endl;
    qDebug()<<"Result3:";
    qDebug()<<qlist.at(3);
    qDebug()<<qlist.contains(77);
    qDebug()<<qlist.contains(166);

    // 修改 qlist 列表里面的元素值
    qDebug()<<endl;
    qDebug()<<"Result4:";
    qlist.replace(5,888);
    qDebug()<<qlist;

    // 删除元素
    qDebug()<<endl;
    qDebug()<<"Result5:";
    qlist.removeAt(0);
    qlist.removeFirst();
    qlist.removeAt(6);
    qDebug()<<qlist;

    return a.exec();
}
```

二、QLinkedList 类

QLinkedList<T>是一个链式列表，它以非连续的内存块保存数据。QLinkedList<T>不能使用下标，只能使用迭代器访问它的数据项。与 QList 相比，当对一个很大的列表进行插入操作时，QLinkedList 具有更高的效率。

【案例分析】

```
#include <QCoreApplication>

#include <QDebug>
#include <qlinkedlist.h>

int main(int argc, char *argv[])
{
    QCoreApplication a(argc, argv);

    // QLinkedList 类
    QLinkedList<QString> qAllMonth;

    for(int i=1;i<=12;i++)
        qAllMonth<<QString("%1%2").arg("Month:").arg(i);

    // 读写迭代器
    qDebug()<<"Result1:";
    QLinkedList<QString>::iterator itrw=qAllMonth.begin();
    for(;itrw!=qAllMonth.end();itrw++)
        qDebug()<<*itrw;

    // 只读迭代器
    qDebug()<<endl<<"Result2:";
    QLinkedList<QString>::const_iterator itr=qAllMonth.constBegin();
    for(;itr!=qAllMonth.constEnd();itr++)
        qDebug()<<*itr;

    return a.exec();
}
```

QLinkedList 类不能通过索引方式访问元素（链表），保存大规模数量数据信息建议使用 QLinkedList（插入元素和删除元素速度快、效率高）。

三、STL 风格迭代器遍历容器

容 器 类	只读迭代器类	读写迭代器类
QList<T>,QQueue<T>	QList<T>::const_iterator	QList<T>::iterator
QLinkedList<T>	QLinkedList<T>::const_iterator	QLinkedList<T>::iterator