1. Qmap

----------。更详细的说明，请查看《Qt 帮助手册》或其他资源。

Qt中的**QMap**介绍与使用，在坛子里逛了一圈，发现在使用**QMap**中，出现过很多的问题，Map是一个很有用的数据结构。它以“键-值”的形式保存数据。在使用的时候，通过提供字符标示（键）即可得到想要的数据。这个“数据”即可以是一个字符串，也可以是任意对象，当然也包括自己定义的类对象。说明：map是以值传递的形式保存数据的。

1. 基本应用

下面以“键-值”都是QString的例子说明**QMap**的基本使用方法。更详细的说明，请查看《Qt帮助手册》或其他资源。

```
#include <qmap.h>
#include <iostream>
using namespace std;
class MapTest
{
public:
    void showMap()
    {
if(!m_map.isEmpty()) return; //判断map是否为空
m_map.insert("111", "aaa"); //向map里添加一对“键-值”
if(!m_map.contains("222")) //判断map里是否已经包含某“键-值”
     m_map.insert("222", "bbb");
m_map["333"] = "ccc"; //另一种添加的方式
qDebug("map[333] , value is : " + m_map["333"]); //这种方式既可以用于添加，也可以用

if(m_map.contains("111")){
     QMap<QString,QString>::iterator it = m_map.find("111"); //找到特定的“键-值”
     qDebug("find 111 , value is : " + it.data()); //获取map里对应的值
}
cout<< endl;
qDebug("size of this map is : %d", m_map.count()); //获取map包含的总数
cout<< endl;
QMap<QString,QString>::iterator it; //遍历map
for ( it = m_map.begin(); it != m_map.end(); ++it ) {
            qDebug( "%s: %s", it.key().ascii(), it.data().ascii()); //用key()和
        }

m_map.clear(); //清空map
    }
private:
    QMap<QString,QString> m_map; //定义一个QMap对象
};
```

调用类函数showMap(),显示结果：

```
map[333] , value is : ccc
find 111 , value is : aaa
size of this map is : 3
111: aaa
222: bbb
333: ccc
```

2. 对象的使用

map当中还可以保存类对象、自己定义类对象，例子如下（摘自QT帮助文档《Qt Assistant》，更详细的说明参考之）：

以注释形式说明

```
#include <qstring.h>
#include <qmap.h>
#include <qstring.h>

//自定义一个Employee类，包含fn、sn、sal属性
class Employee
{
public:
    Employee(): sn(0) {}
    Employee( const QString& forename, const QString& surname, int salary )
        : fn(forename), sn(surname), sal(salary)
    { }

    QString forename() const { return fn; }
    QString surname() const { return sn; }
    int salary() const { return sal; }
    void setSalary( int salary ) { sal = salary; }

private:
    QString fn;
    QString sn;
    int sal;
};

int main(int argc, char **argv)
{
    QApplication app( argc, argv );

    typedef QMap<QString, Employee> EmployeeMap; //自定义一个map类型，值为Employe
    EmployeeMap map;

    map["JD001"] = Employee("John", "Doe", 50000); //向map里插入键-值
    map["JW002"] = Employee("Jane", "Williams", 80000);
    map["TJ001"] = Employee("Tom", "Jones", 60000);

    Employee sasha( "Sasha", "Hind", 50000 );
    map["SH001"] = sasha;
    sasha.setSalary( 40000 ); //修改map值的内容，因为map采用值传递，所以无效

    //批量打印
    EmployeeMap::Iterator it;
    for ( it = map.begin(); it != map.end(); ++it ) {
        printf( "%s: %s, %s earns %d\n",
                it.key().latin1(),
                it.data().surname().latin1(),
                it.data().forename().latin1(),
                it.data().salary() );
    }
    return 0;
}
```

```
Program output:
JD001: Doe, John earns 50000
JW002: Williams, Jane earns 80000
SH001: Hind, Sasha earns 50000
TJ001: Jones, Tom earns 60000
```

2. qlistwidget 选中某项

```
oid ListWidget::mousePressEvent(QMouseEvent *event)
{
    __super::mousePressEvent(event);
    QListWidgetItem *pItem = itemAt(event->pos());
    if (pItem)
    {
        int iCount = count();
        for (size_t i = 0; i < iCount; i++)
        {
            QListWidgetItem *p = item(i);
            if (!p)
                break;

            if (p != pItem)
                p->setSelected(false);
            else
                p->setSelected(true);
        }
    }
}
```

//

举个例子：

比如我的 ListWidget 中已经有几个 Item 了，我现在想把第二个默认选中，那代码就是：

```
QlistWidgetItem *item = ListWidget->item(1);
item->setSelected(true);
```