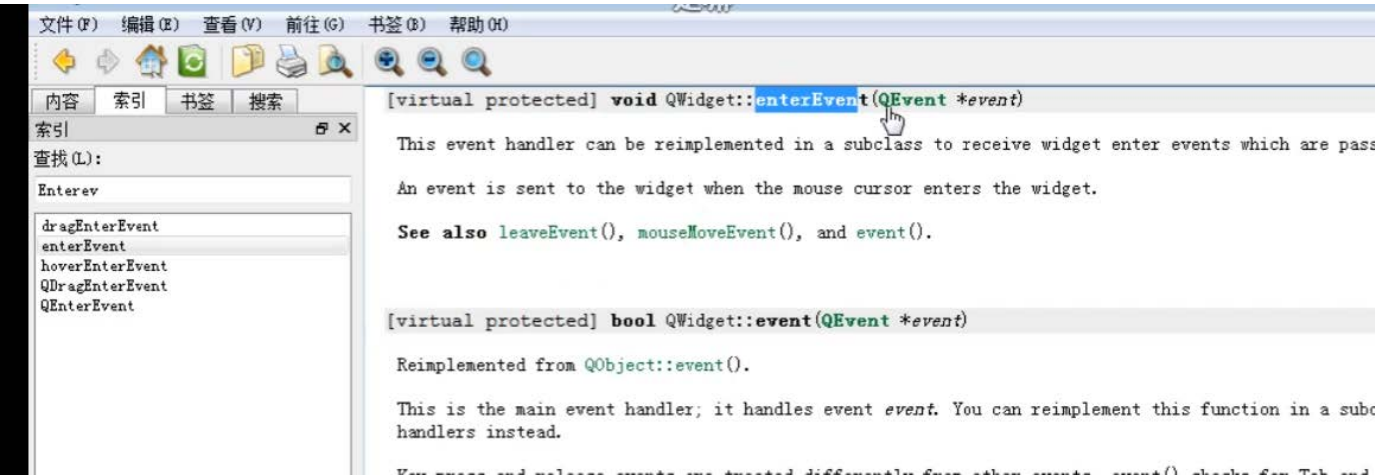
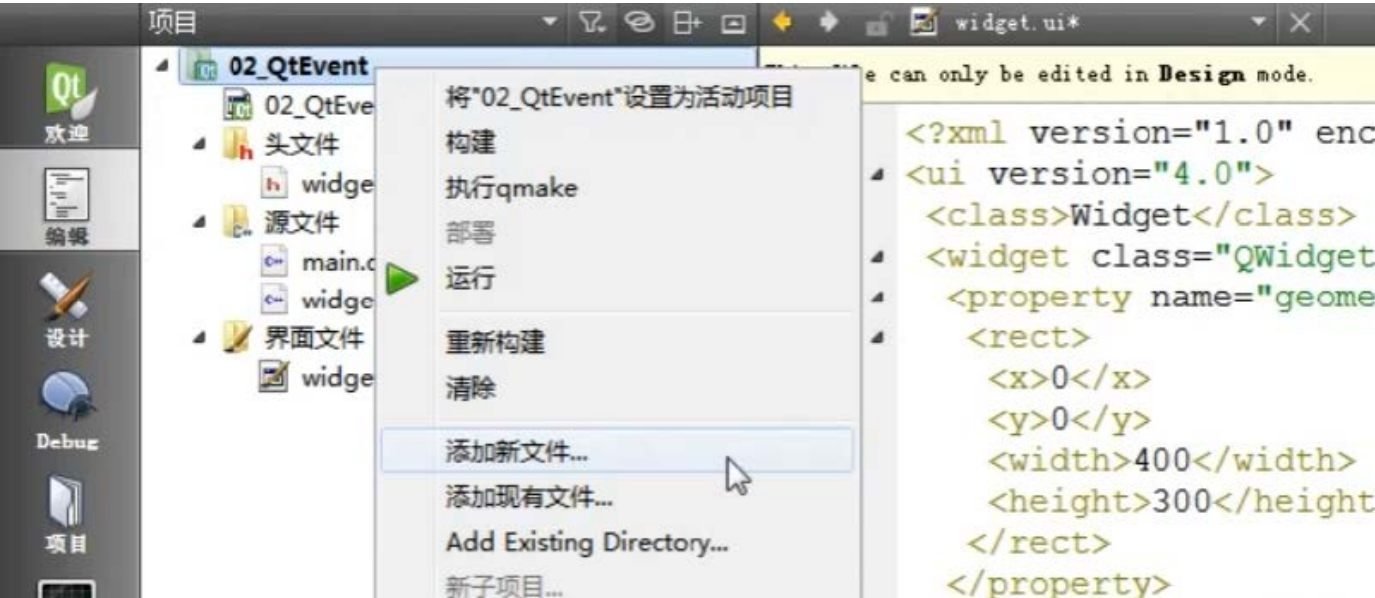


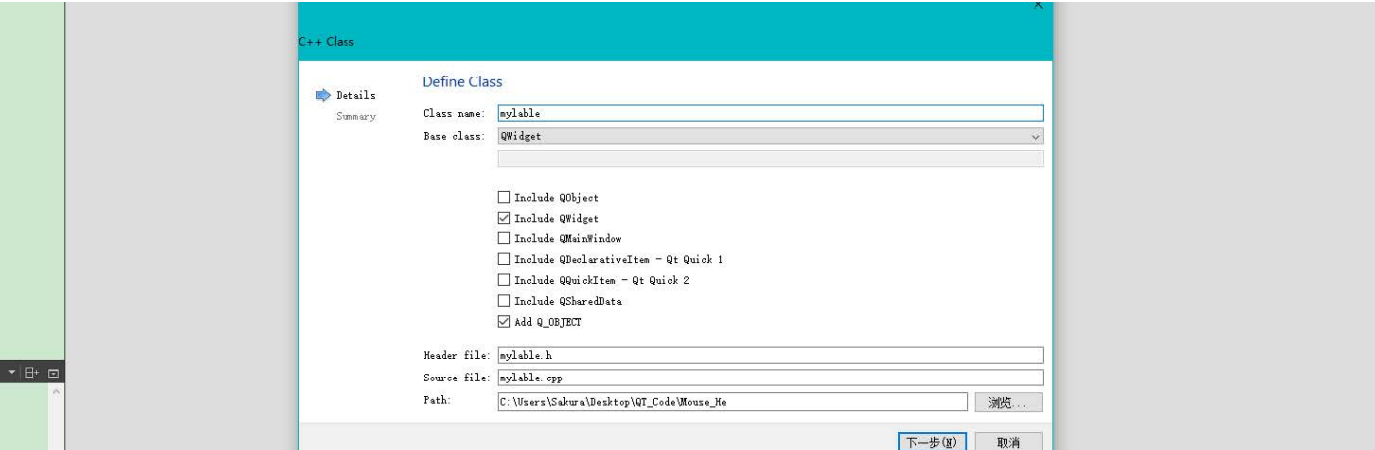
QEvent



事件案例



C++ class



函数声明

mylable.h

```
#ifndef MYLABEL_H
#define MYLABEL_H

#include <QLabel>

class mylable : public QLabel
{
    Q_OBJECT
public:
    explicit mylable(QWidget *parent = nullptr);

    // 鼠标进入事件
    void enterEvent(QEvent *);

    // 鼠标离开事件
    void leaveEvent(QEvent *);
signals:

};

#endif // MYLABEL_H
```

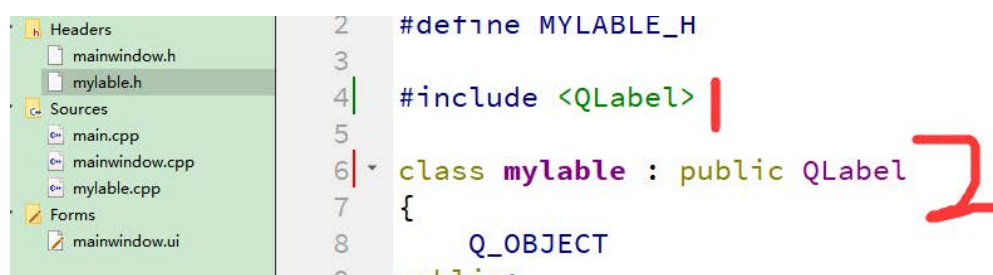
函数实现

mylable.cpp

```
void mylable::enterEvent(QEvent *)
{
    qDebug()<<"鼠标移入";
}
void mylable::leaveEvent(QEvent *)
{
    qDebug()<<"鼠标移出";
}
```

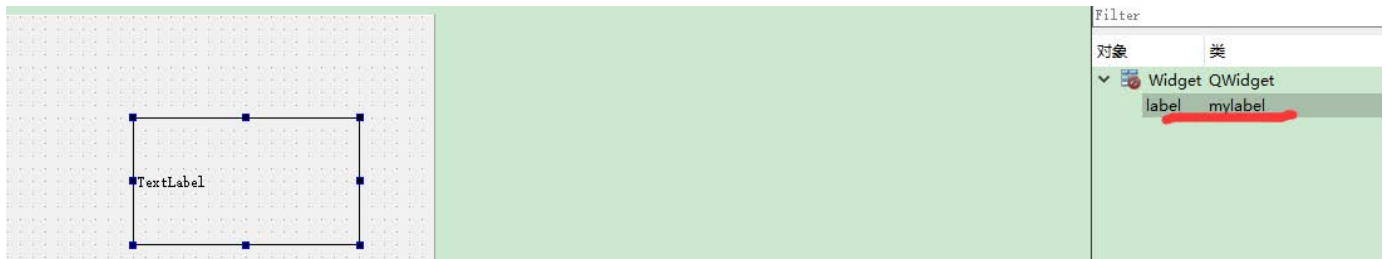
提升

由于我们要提升QLabel类型，因此，需要先改动以下三处的值为QLabel



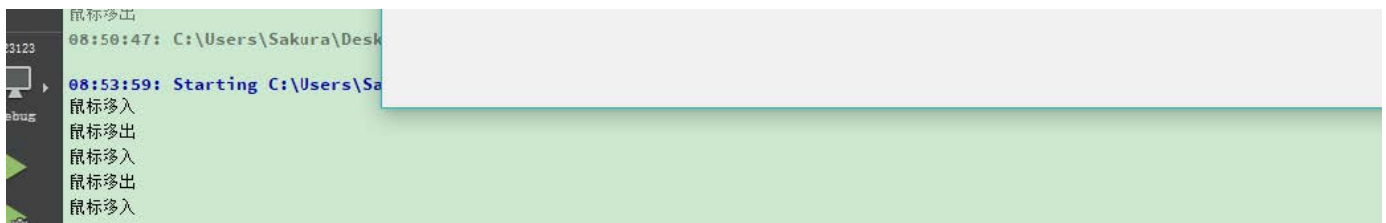


更改好之后对我们刚才的标签进行提升---提升为mylabel



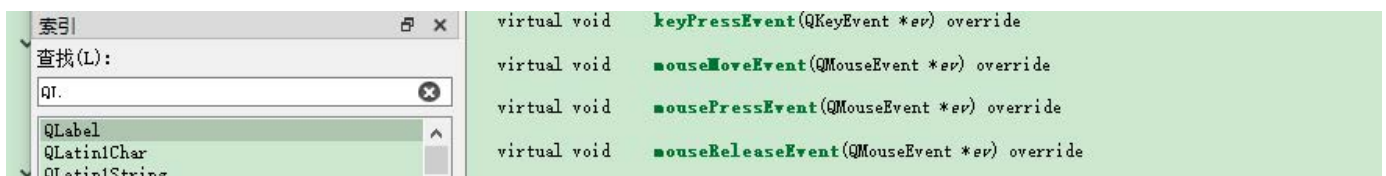
类型匹配才可以提升!!!!

现在, 我们已经能够捕获鼠标的进入和离开了



连接按钮（信号与槽）

在帮助文档中, 我们查到QLabel的三个虚函数



分别对应 移动, 按下, 离开

我们对这三个函数进行重写, 方便接下来使用

mylabel.h

```

virtual void mouseMoveEvent(QMouseEvent *ev); // 按下
virtual void mousePressEvent(QMouseEvent *ev); // 释放
virtual void mouseReleaseEvent(QMouseEvent *ev); // 移动

```

mylabel.cpp

```

void mylabel::mouseMoveEvent(QMouseEvent *ev)

```

```

{
    qDebug()<<"鼠标移动";
}
void mylabel::mousePressEvent(QMouseEvent *ev)
{
    qDebug()<<"鼠标按下";
}
void mylabel::mouseReleaseEvent(QMouseEvent *ev)
{
    qDebug()<<"鼠标释放";
}

```

```

鼠标按下
鼠标释放
鼠标移出
09:05:08: C:\Users\Sakura\Desktop\QT_Code\build-123123-Desktop_Qt_5_12_10_MinGW_64_bit-Debug\debug\123123.exe exited with code 0

```

*ev参数可以捕获所有信息

我们查询帮助文档

Public Functions

```
QMouseEvent(QEvent::Type type, const QPointF &localPos, Qt::MouseButton button, Qt::MouseButtons buttons, Qt::KeyboardModifiers modifiers)
```

```
QMouseEvent(QEvent::Type type, const QPointF &localPos, const QPointF &screenPos, Qt::MouseButton button, Qt::MouseButtons buttons, Qt::KeyboardModifiers modifiers)
```

```
QMouseEvent(QEvent::Type type, const QPointF &localPos, const QPointF &windowPos, const QPointF &screenPos, Qt::MouseButton button, Qt::MouseButtons buttons, Qt::KeyboardModifiers modifiers)
```

```
QMouseEvent(QEvent::Type type, const QPointF &localPos, const QPointF &windowPos, const QPointF &screenPos, Qt::MouseButton button, Qt::MouseButtons buttons, Qt::KeyboardModifiers modifiers, Qt::MouseEventSource source)
```

```
Qt::MouseButton button() const
```

```
Qt::MouseButtons buttons() const
```

```
Qt::MouseEventFlags flags() const
```

```
QPoint globalPos() const
```

```
int globalX() const
```

```
int globalY() const
```

```
const QPointF &localPos() const
```

```
QPoint pos() const
```

```
const QPointF &screenPos() const
```

```
Qt::MouseEventSource source() const
```

```
const QPointF &windowPos() const
```

```
int x() const
```

```
int y() const
```

我们想拿到它的X和Y（坐标）

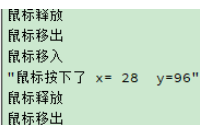
先包含头文件#include

然后，在按下的函数中输出（x，y）

mylabel.cpp

```
void mylabel::mousePressEvent(QMouseEvent *ev)
{
    QString str =QString("鼠标按下了 x= %1 y=%2").arg(ev->x()).arg(ev->y());
    qDebug()<<str;
}
```

这个是可以无限追加的。

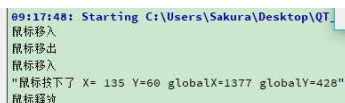


```
鼠标释放
鼠标移出
鼠标移入
"鼠标按下了 x= 28 y=96"
鼠标释放
鼠标移出
```

现在，我们再使用一下ev的global坐标值

mylabel.cpp

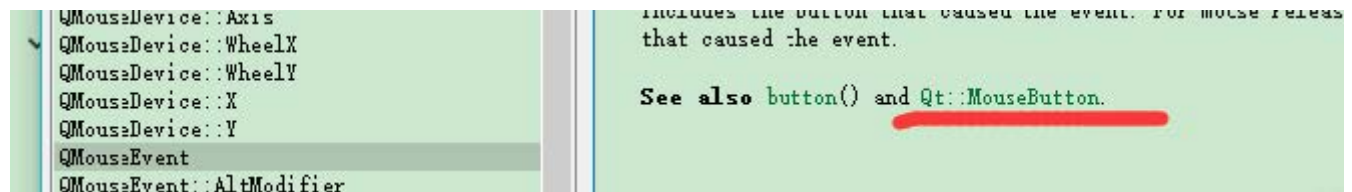
```
void mylabel::mousePressEvent(QMouseEvent *ev)
{
    QString str =QString("鼠标按下了 X= %1 Y=%2 globalX=%3 globalY=%4").arg(ev->x()).arg(ev->y()).arg(ev->globalX()).arg(ev->globalY());
    qDebug()<<str;
}
```



```
09:17:48: Starting C:\Users\Sakura\Desktop\QT
鼠标移入
鼠标移出
鼠标移入
"鼠标按下了 X= 135 Y=60 globalX=1377 globalY=428"
鼠标释放
```

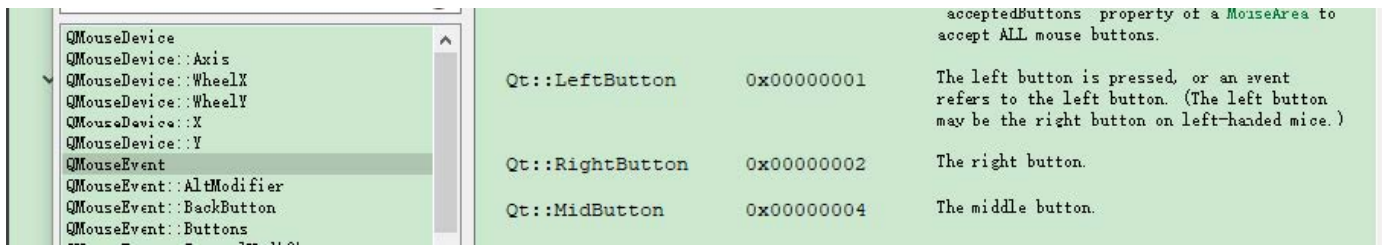
现在有了一个新的问题，就是——只有按下之后才会捕获移动信息，我们想让它停靠时也可以捕获位置

老规矩，查！



QMouseEvent::Axis	includes the button that caused the event. For mouse release that caused the event. See also <code>button()</code> and <code>Qt::MouseButton</code> .
QMouseEvent::WheelX	
QMouseEvent::WheelY	
QMouseEvent::X	
QMouseEvent::Y	
QMouseEvent	
QMouseEvent::AltModifier	

点进来之后可以看到很多的枚举值，我们挑选我们需要的



分别是，左击，右击，和滚轴按键

接下来是使用这三个函数

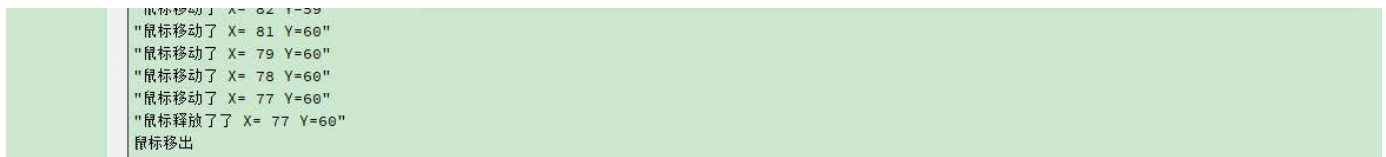
mylabel.cpp

```
void mylabel::mouseMoveEvent(QMouseEvent *ev)
{
    if(ev->button()==Qt::LeftButton)
    {
        QString str =QString("鼠标移动了 X= %1 Y=%2").arg(ev->x()).arg(ev->y());
        qDebug()<<str;
    }
}
```

此时便不会打印移动，因为按下和释放属于瞬间，而移动属于过程，因此，我们使用buttons联合按键，同真才为真。

mylabel.cpp

```
void mylabel::mouseMoveEvent(QMouseEvent *ev)
{
    if(ev->buttons() & Qt::LeftButton)//只有左键按下后才会打印
    {
        QString str =QString("鼠标移动了 X= %1 Y=%2").arg(ev->x()).arg(ev->y());
        qDebug()<<str;
    }
}
```



接下来，我们设置鼠标追踪

mylabel.cpp

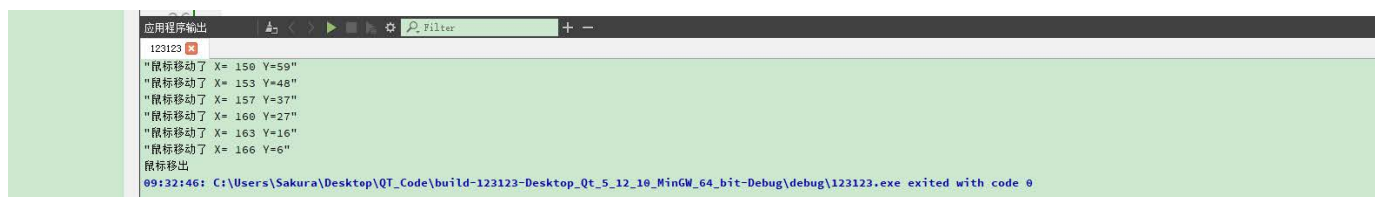
```
#include "mylabel.h"
#include<QDebug>
#include<QMouseEvent>
mylabel::mylabel(QWidget *parent) : QLabel(parent)
```

```

{
    setMouseTracking(true);
}
void mylabel::mouseMoveEvent(QMouseEvent *ev)
{
//    if(ev->buttons() & Qt::LeftButton)
//    {
        QString str = QString("鼠标移动了 X= %1 Y=%2").arg(ev->x()).arg(ev->y());
        qDebug() << str;
//    }
}

```

如此一来，我们不需要点击，也可以捕获鼠标坐标了



总结:

- 2.1 鼠标事件
- 2.2 鼠标进入事件 `enterEvent`
- 2.3 鼠标离开事件 `leaveEvent`
- 2.4 鼠标按下 `mousePressEvent (QMouseEvent ev)`
- 2.5 鼠标释放 `mouseReleaseEvent`
- 2.6 鼠标移动 `mouseMoveEvent`
- 2.7 `ev->x()` x坐标 `ev->y()` y坐标
- 2.8 `ev->button()` 可以判断所有按键 `Qt::LeftButton` `Qt::RightButton`
- 2.9 `ev->buttons()` 判断组合按键 判断move时候的左右键 结合 `&` 操作符
- 2.10 格式化字符串 `QString(" %1 %2 ").arg(111).arg(222)`
- 2.11 设置鼠标追踪 `setMouseTracking(true);`