# QT学习笔记

1. 带参数的自定义信号和槽连接

connect(发送者，发送的函数信号地址，接收者，接收的函数信号地址)

示例代码：

zt = new Teacher(this);

st = new Student(this);

void(Teacher:: \* teacherSignals)(QString) = &Teacher::hungry;

void(Student:: \* studentSlot)(QString) = &Student::treat;

connect(zt, teacherSignals, st, studentSlot); （这里采用函数指针）

1. 采用emit 自定义信号触发

emit->hungry();

1. QString转char \*

QString对象.toUtf8().data()

toUtf8()后得到QByteArray

1. 信号和槽可以断开连接

disconnect(发送者，发送的函数信号地址，接收者，接收的函数信号地址)

1. 信号和槽拓展
2. 信号可以连接信号
3. 一个信号可以连接多个槽函数
4. 多个信号可以连接同一个槽函数
5. 信号和槽函数的参数必须一一对应，参数个数可以不一致，信号的参数个数可以多于槽函数

6、Lambda表达式

[capture list] （parameter list） -> return type { function body }

1. 鼠标按钮事件分析

void EventLabel::***mousePressEvent***(QMouseEvent \*e)

{

if(e->button()==Qt::LeftButton){

// TODO

}

}

假设鼠标左键已经按下，

如果移动鼠标，会发生的move事件，button返回Qt::NoButton，buttons返回LeftButton。

再按下了右键，会发生press事件，button返回RightButton，buttons返回LeftButton | RightButton

再移动鼠标，会发生move事件，button返回Qt::NoButton，buttons返回LeftButton | RightButton

再松开左键，会发生Release事件，button返回LeftButton，buttons返回RightButton

也就是说，

button返回“哪个按钮发生了此事件”，buttons返回"发生事件时哪些按钮还处于按下状态"

if (event->buttons() & Qt::LeftButton) 是判断左键是否按下了,只要左键按下了就返回真,(也可能右键也同时按下了)

if (event->buttons() & Qt::LeftButton & Qt::LeftButton)是判断只有左键按下

if ((event->buttons() & Qt::LeftButton) == Qt::LeftButton)是判断只有左键按下

# C++学习笔记

1. 面向对象的程序设计的基本特征

继承（inheritancce）

封装（encapsulation）

多态（polymorphism）

抽象（abstraction）

1. 对引用的进一步说明 （引用一定要引用一个类型实体）
2. 不允许建立void类型的引用

void &r = 10; //错误

void只是在语法上相当于一个类型 本质上不是类型 void的含义是无类型或空类型 任何实际存在的变量都是属于非void类型的

1. 不允许建立引用的数组

int a[4] = ”abcd”;

int &ra[4] =a; //错误

数组名a只代表数组首元素的地址，本身并不是一个占有存储空间的变量

1. 不能建立引用的引用 不能建立指向引用的指针 引用本身不是一种数据类型，所以没有引用的引用，也没有引用的指针

int n = 10;

int &&r = n; //错误

int & \*p = n; //错误

1. 初始化列表

对于使用const修饰的数据成员或是使用引用的数据成员，是不允许使用赋值语句直接赋值的，只能使用初始化列表进行赋值

类成员是按照它们在类中被声明的顺序进行初始化的，与它们在初始化列表中出现的顺序无关

1. 浅拷贝和深拷贝

所谓浅拷贝，就是由默认的拷贝构造函数所实现的数据成员的逐一复制。通常的默认的拷贝构造函数是能够胜任此工作的，但若类中含有指针类型的数据，则这种按数据成员逐一复制的方法将会产生错误。