明白指针都指到哪里去了

对象用. 指针用->

指针函数 ，返回值是指针

指针 &函数名 和函数名 数组名a和&a[0]

函数名和函数指针 函数名和函数指针 通过指针调用函数 fn()

类对象和类指针 a\*= new A（）

字符指针 cha \*p=“” 不可以修改 print(p)

指向结构的指针

调用dll 用win32api loadliberary getprocaddress 取函数指针

Qliberary .load

C++ 构造函数后的冒号实际上是赋值，对对象成员初始化

QT是高度面向对象的，封装完美的

头文件

Qt中 .会被自动识别

对象和对象指针的区别 . ->

如何用代码控制ui 利用ui->成员

调用win32api 需要改 pro

win{

LIBS += -lUser32

}

WinID() 获取部件句柄

winEvent() 接受原始win消息

Qt quick

增加开发效率的手段 使用ui拖 使用css 样式表 使用qml

一、 项目名 类名 类名基类是Qwindow

Ui对象 安装ui 加入按钮->设置文字

设置字符编码

设计界面 对话框

子窗口的打开?

设计模式中设置信号和槽 发送信号， 接收槽 发送者 接受者 都是函数

设计模式

菜单 菜单图标 资源 代码实现

动作Action

二、 widgets 项目创建 .pro ui h cpp

Main.cpp windowmain.cpp

用ui 设计区 部件 对象查看器 属性编辑 动作和信号槽

2、不用ide直接运行程序缺少dll库，引入debug会非常大，应该发布成 release版，需要带上release依赖

设置应用图标 .icon 修改 .pro

使用原生win32api，需要引入windows .h和修改pro

3、纯代码 ui 自定义类 qmake生成makefile mingw32-make 编译

入口是main函数 引用一个 窗口对象 w.show() 在对象的构造函数中写控件

原生类 Dialog 等 继承类

编码 QOBject::tr()

Qapplication qlabel qdialog label(&w) label.setText w.resize() label.move()

setWindowTitle

UI界面 属性栏修改属性 .ui- 生成.h头文件 和纯代码生成是一样的

引入ui的头文件，然后Ui.setupUI(&w)

三、部件 主窗口，带菜单和工具栏Qmainwindow qdialog qwidget基类，继承qobject,qpaintdevice

窗口，子部件

Qwidget \*widget=new Qwidget() c++必须用指针指向new ，对象实际上不用new

设置标题 文本 Qlabel \*label2=new Qlabel(widget) //指定父窗口，有父窗口可以不调用show方法

QT消除父对象时会消除子对象，所以 没有父对象需要 delete

窗口部件（顶层），子部件

New 构造函数的参数 \*parent 父 默认是0 窗口标志 划分窗口类型 flags

类型 QT::Dialog 对话框 QT::SplashScreen 欢迎, 无边框 Framelesswindowhint 停留在其他窗口上windowstaysontophint

用|管道符连接

设置窗口状态 setwindowstate qt::windowstate枚举 最大化windowmaximazed 最小化minimized 全屏fullsceen 活动窗口 active nostate

几何布局 窗口的大小 ，位置 包含 边框 x() y() frameGeoetry() pos() move()

不包含边框（客户区,子部件?） width() height rect size

2、调试 设置断点 单步调试 汇编 堆栈 线程 3、Qdebug() 输出信息到控制台

直接输出字符串 输出流qdebug()<<

对话框 模态 非模态

关于this ,一般看情况，指当前的窗口对象

2、 Qdialog dialog(this) 和 \*dialog=new Qdialog(this)的区别 定义在构造函数中，属于堆栈，构造函数执行完会自动释放 写在main中就不会释放，要不用 new开辟堆空间，这样就必须手动回收

指向类对象的指针

Dialog.show 非 dialog.exec 模态可以不用new 直接用Qdialog dialog(this)定义

也可以new后用 setModal 然后 show

3.2.2多窗口切换 信号和槽 一个部件发射信号，另一个部件使用槽（自定义的回调）

可能是对象方法或者普通函数

对象名(id)

槽定义 用slots 或者connect语法 ,写槽函数的功能，如弹出，显示，修改界面

Connect(发射对象，信号，接受对象，槽函数)

静态方法和静态变量的使用 &类名::方法 这里的参数是回调，所以无（）

接收对象不一定重要，设成全局this就好，然后在槽中修改其他对象属性

2、自动关联 重命名， on\_theButton\_cliked() on\_发射对象\_消息（）

3、用设计器做 关联信号和槽

4、常用对话框类 获取颜色 获取文件 获取字体 输入 消息 进度 错误消息 向导

点击按钮弹出 clicked()

类静态方法::get获取颜色文件输入等， 生成对象法，更加灵活

消息: 问题 提示 警告 错误 关于

&是取指针，或者引用，在参数中的使用

5．常用部件 Qframe 边框部件 标签 工具箱 滑动区 文本编辑

边框效果:边框阴影 边框形状 组合出不同的边框 框 水平线 凸起 立体

setFrameShape setFrameShadow 或者setFrameStyle

设置线宽 setLineWidth setMidLineWidth

1. 标签，显示文字或者图片 设置字体.setFont(font) fontMetrics setText

Qstring str=tr() 图片 <qpixmap> setpixmap(图片地址)

Gif Qmovie \*movie=new Qmovie(“gif”) label.setMovie()

1. 字体类 Qfont font .设置字体家族 设置像素大小 设置加粗 设置斜体
2. 液晶数字
3. 部件栈 有多个页面，每次显示一个界面 需要加一个列表 或者ComboBox 作为选择器 信号
4. 工具栏
5. 下压按钮 可以改成切换按钮 on\_button\_toggled(bool checked) 有两种状态 ，设置文本 设置图标

菜单 Qmenu \*menu=new Qmenu(this) menu->addAction()

指定加速键 & 加速键就是快捷键 触发菜单或者用alt

1. 单选 复选 需要用到Group box 里面拖入checkbox 属性:tristate 不改变 选中 未选中

信号 stateChanged() ischecked()

1. 单行文本编辑 1显示模式echoMode normal noecho不显示 password 2、输入掩码 限制 输入模式 3、验证器 Qvalidator \* =new setValidator() QdoubleValidator QREgexpvalidator 4、自动补全 Qcompleter
2. 设定框 （选择框）
3. 滑块

四、布局 子部件位置 默认大小，最小大小 变化时处理 内容改变时更新（绑定）

隐藏显示 移除

Qlayout 父类 子项 基本布局 栅格布局 窗体布局 栈布局

属性 函数 addWidget()增加部件 设置间隔setSpacing 设置到边界的距离setContentMargins() 窗口使用布局setLayout()

网格 addWidget(部件，行，列)

水平布局 综合使用，拖动

2、部件大小 大小提示 最大大小提示 sizeHint() set

大小策略属性:默认布局行为 伸缩因子

显示隐藏setVisible(bool) hide() show()

4、分裂器

5、伙伴 加速键

6、利用tab切换焦点setTabOrder(前一个，后一个)

五、主窗口

包含的部件（可选） 菜单栏 工具栏 中心部件 dock部件停靠 状态栏

Ui模式 添加菜单 在这里输入 Action编辑器

菜单项是动作类

使用资源 添加资源 qrc其实是xml

2、代码模式加入菜单 菜单栏增加菜单 \* edit= ui->menuBar->addMenu()

\* Acton=Edit->AddAction() 增加菜单项

工具栏增加菜单项 ui->mainToolBar->addAction

设置快捷键 action->setShortCut() QkeySequence(“Ctrl+0”)

可以加入间隔器/水平线 设计器或者 addSeparator

动作（菜单项）包括图标 文本 快捷键 状态栏 提示文本 checkable 已选中

setCheckable

动作组 可以设置互斥 group=new Q group->addAction()

动作信号 triggered() 和槽 on\_action\_triggered()

创建菜单 Qmenu \*menu=new Menu(this) 给按钮增加菜单toolBtn->setMenu()

->setPopupMode() 弹出模式 按钮加在菜单栏

中心部件 text edit 富文本 纯文本 富文本浏览器（只读）

拖入

\*edit=new QtextEdit(this)

创建子窗口 QmidSubWindow \*child=ui->mdiArea->addSubWindow(edit) 位于midArea

显示子窗口 child->show()

多文档部件MDI(管理多个窗口) QmdiArea

Dock可以停在window中 也可以成为顶级窗口 属性标题 features可否关闭移动悬浮

allowedArea允许停靠的区域

状态栏 正常消息 addWidget 加部件到左边 临时消息showMessage 永久消息 增加一个Label addPermanentWidget 右边

自定义菜单，菜单内加入任意部件 继承QwidgetAction 重写createWidget()

富文本 对象 光标 框架 文本快 表格 列表 图片 所有的属性和方法 查找功能：

find()

拖放 打开文件 事件处理函数，鼠标

Html

六、事件处理 键盘事件 鼠标事件 定时器事件 信号和事件不同，但是可以同时发生

1、事件处理 重写部件的paintEvent() mousePressEvent()

重写notify 一次只能处理一个事件

Qapplication 安装事件过滤器，全局

重写event()

不同对象上安装事件过滤器

2、事件传递 exec() 事件循环 event对象参数

执行事件的默认处理（避免出问题） 使用类名：：keyPressEvent

忽略事件 event->ignore()

3、事件过滤器 声明过滤器函数 部件->installEventFilter(this) eventFilter(obj,event)

4、鼠标和滚轮事件 按下 释放 双击 Widget::mouseReleaseEvent(event)

光标设置 Qcursor cursor cursor.setShape()

获取指针位置 event->globalpos 窗口位置pos event->button()按下哪个键

双击全屏

Event->delta()>0 滚轮上滚 <0下滚

->zoomin()放大 zoomout()缩小

4、键盘 event->key() 按键 Qt::Key\_Return 修饰键 event->modifies() 是否按下

按键释放

部件移动，->move

5、定时器随机数 Ti qrand qsrand

startTimer(1000) timerEvent()

startTimer 返回定时器句柄(id) event->timerID()来获取区分

对象 new Qtimer(this) 然后关联信号 connect(timer,信号，接受，回调)

时间 Qtime time=Qtime::currentTime() time.tostring(格式) time.second

6、事件过滤器 安装过滤器 事件过滤函数 每个部件都可以安装

eventFilter(obj,event) 判断是哪个部件的事件 判断事件类型 event->type(),转型

if else

发送事件 sendEvent 立即处理 postEvent等待事件循环

七、对象模型和容器类 信号和槽 属性系统 对象树和拥有权 元对象系统

字符串 字节数组 正则

字符编码 翻译机制 动态对象转换 守护指针

1. 信号和槽 可以一对多 多对一 多槽 多信号

自定义信号 发送信号 emit 函数（） 然后连接

关联类型

自动关联 断开关联 disconnect

1. 基于元对象的属性系统 Qbbject Q\_PROPERTY()宏
2. 对象树Qobject的子类 父对象 parent 子对象 children 父销毁时子也会销毁

方便管理 主窗口应该在栈上而不是在堆上 子对象在堆上时需要定义父new 不用delete 重定义父部件

4、元对象 信号和槽 返回类的元对象 类似反射 ::metaObject() className inherits() Qobject::tr trUtf8 ::property ::setproperty newInstance()

5、容器 向量 线性表 链表 栈 队列 Map Set Hash 迭代器 遍历容器

hasNext() next() hasPrevious() previous()

foreach 通用算法

6、Qstring 隐式共享（写时复制） 字符串操作 大小 替换 查找 增删查改 拆分 转换 isNull()

字节数组 共用体 正则

八、界面外观

1、风格 选择 Qapplication.setStyle()

2调色板 激活 失效 非激活 每个部件一个调色板 颜色组 setPalette pallette()

1. 样式表 部件->setStyleSheet(“属性:值”); 类似css

设计模式中 使用样式表 部件{属性:值 } 类似于css 选择器 部件 属性 id 后代

类选择（选择一种类型）

伪状态 :hover :pressed 层叠继承 盒模型 边框 圆角

规则 前景色 背景色 字体 边框圆角 外边距内边距 宽高 定位 渐变

最大最小宽

读入qss文件 读取文件 tr()转字符串 setStyleSheet()

1. 遮罩，不规则窗口 ->setMask()
2. 透明 背景色 或者是 setWindowOpacity() 或者 setWindowFlags setAttribute
3. 毛玻璃

九、国际化 帮助系统 QT插件

1、翻译 tr() 加速键 QkeySequence 动态文本

2、工具提示 状态提示 帮助提示 QwhatsThis::createAction(this)

十、2D绘图 paintEvent Qpaint paint(this) 绘图区 painter.drawLine() .begin .end drawArc drawRect 画椭圆 圆弧 画路径 。save .restore

2、画笔 Qpen pen(画刷，宽，风格，端点风格，连接风格)

可以用.setBrush setWidth setStyle setCAPstyle setjoinstyle

painter.setPen 给painter设置画笔

3、画刷 用于对图形填充 Qbrush bursh(颜色，填充模式) paint.setBrush 填充

4、渐变填充 5、坐标系统 抗锯齿渲染 抗锯齿绘图 坐标系变换，坐标系缩放平移旋转，窗口视口 painter.rotate .scale .translate() 6、绘制文字 painter.drawText() drawStaticText() 7、绘制路径 path对象 .moveTo lineto() 贝赛尔 .cubicTo() 路径加入椭圆.add painter.drawPath 路径填充规则setBrush 8、图像 Qimage是独立对象 painter可以在图像上画 image.load painter.drawImage(,image) 转换器对象 复制图片 image.mirrored() image2.transformed(转换对象 ) 9、像素对象 fill 从图像加载像素。 fromIamge 像素转换为图像.toImage .load painter.drawPixelmap 10、截取屏幕 Screen 11 复合模式 12、双缓冲绘吐 13、剪切图像 setClipRect 14、重绘 repaint() update() 15、读取和写入图像 save 16、播放gif svg

十一、图形视图，动画，状态机

类似于游戏开发了

1、图形的控制 拖动 位置检测 碰撞检测 场景 视图 图形项

图形交互 ，点击移动 bsp树

2、场景 items() addItem() itemAt() 根据深度 创建图形项 new

3、视图，给场景和图形项增加显示 view(场景或者图形项) 前景色 背景色

View.resize view.show

4、图形项 支持事件 分组 碰撞检测 存储自定义数据（属性）setDate

方形 圆形

坐标系统，相对于父图形项.setPos()

事件处理和传播

5、图形效果 模糊 染色 阴影 透明

6、动画 碰撞 定时器或者advance推进场景 （2） 定义图形项形状 重写shape（不重写默认是使用绑定的rect） colidesWithItem()

7。、打印和opengl渲染 8、图形项的几何和布局

9、动画框架 属性框架（缓动） 属于状态机框架的一部分

10、状态机

十二、3D绘图

十三、音视频播放

十四、相机 录音

十五、文件目录 io

十六、模型/视图 mvc 数据源

十七、数据库 xml

十八、网络编程 请求

十九、进程线程 线程同步 可重入 线程安全 进程间通信

二十、网络引擎，嵌入网页 依赖谷歌内核