1 自定义控件封装
- 1.1 添加新文件 - Qt – 设计师界面类 (.h .cpp .ui)
- 1.2 .ui 中 设计 QSpinBox 和 QSlider 两个控件
- 1.3 Widget 中使用自定义控件，拖拽一个 Widget，点击提升为，点击添加，点击提升
- 1.4 实现功能，改变数字，滑动条跟着移动 ，信号槽监听。
- 1.5 提供 getNum 和 setNum 对外接口
- 1.6 测试接口

2 Qt 中的事件
- 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);

3 定时器 1
- 3.1 利用事件 void timerEvent (QTimerEvent * ev)
- 3.2 启动定时器 startTimer(1000) 毫秒单位
- 3.3 timerEvent 的返回值是定时器的唯一标示 可以和 ev->timerId 做比较

4 定时器 2
- 4.1 利用定时器类 QTimer
- 4.2 创建定时器对象 QTimer * timer = new QTimer(this)
- 4.3 启动定时器 timer->start(毫秒)
- 4.4 每隔一定毫秒，发送信号 timeout ,进行监听
- 4.5 暂停 timer->stop

5 event 事件
- 5.1 用途：用于事件的分发
- 5.2 也可以做拦截操作，不建议
- 5.3 bool event(QEvent * e);
- 5.4 返回值 如果是 true 代表用户处理这个事件，不向下分发了
- 5.5 e->type() == 鼠标按下 …

6 事件过滤器
- 6.1 在程序将时间分发到事件分发器前，可以利用过滤器做拦截
- 6.2 步骤
  - 6.2.1 1、给控件安装事件过滤器
  - 6.2.2 2、重写 eventFilter 函数 （obj ， ev）

7 QPainter 绘图
- 7.1 绘图事件 void paintEvent()
- 7.2 声明一个画家对象 QPainter painter(this) this 指定绘图设备

7.3 画线、画圆、画矩形、画文字
7.4 设置画笔 QPen 设置画笔宽度 、风格
7.5 设置画刷 QBrush 设置画刷 风格

8 QPainter 高级设置
8.1 抗锯齿 效率低
8.1.1 painter.setRenderHint(QPainter::Antialiasing);
8.2 对画家进行移动
8.2.1 painter.translate(100,0);
8.2.2 保存状态 save
8.2.3 还原状态 restore
8.3 如果想手动调用绘图事件 利用 update
8.4 利用画家画图片 painter.drawPixmap(x，y，QPixmap(路飞))

9 QPaintDevice 绘图设备
9.1 QPixmap QImage QBitmap(黑白色) QPicture QWidget
9.2 QPixmap 对不同平台做了显示的优化
9.2.1 QPixmap pix(300,300)
9.2.2 pix.fill(填充颜色)
9.2.3 利用画家 往 pix 上画画 QPainter painter(& pix)
9.2.4 保存 pix.save(“路径”)
9.3 Qimage 可以对像素进行访问
9.3.1 使用和 QPixmap 差不多 QImage img(300,300,**QImage::Format_RGB32**);
9.3.2 其他流程和 QPixmap 一样
9.3.3 可以对像素进行修改 img.setPixel(i,j,value);
9.4 QPicture 记录和重现 绘图指令
9.4.1 QPicture pic
9.4.2 painter.begin(&pic);
9.4.3 保存 pic.save(任意后缀名)
9.4.4 重现 利用画家可以重现 painter.drawPicture(0,0,pic);

10 QFile 对文件进行读写操作
10.1 QFile 进行读写操作
10.2 QFile file(path 文件路径)
10.3 读
10.3.1 file.open(打开方式) QIODevice::readOnly
10.3.2 全部读取 file.readAll() 按行读 file.readLine() atend()判断是否读到文件尾
10.3.3 默认支持编码格式 utf-8
10.3.4 利用编码格式类 指定格式 QTextCodeC
10.3.5 QTextCodec * codec = QTextCodec::codecForName("gbk");
10.3.6 //ui->textEdit->setText(codec->toUnicode(array));
10.3.7 文件对象关闭 close
10.4 写
10.4.1 file.open(QIODevice::writeOnly / Append)
10.4.2 file.write(内容)
10.4.3 file.close 关闭

11 QFileInfo 读取文件信息

11.1 QFileInfo info(路径)

11.2 qDebug() << "大小：" << info.size() << " 后缀名：" << **info.suffix()** << " 文件名称："<<info.fileName() << " 文件路径："<< info.filePath();

11.3 qDebug() << " 创 建 日 期 ： " << info.created().toString("yyyy/MM/dd hh:mm:ss");

11.4 qDebug() << "最后修改日期："<<info.lastModified().toString("yyyy-MM-dd hh:mm:ss");

12