1. 自定义控件封装
   1. 右键项目- 添加新文件
   2. Qt - Qt设计师界面类
   3. 在ui文件中设计自定义控件
   4. 在主窗口中 拖拽 Widget做提升的操作
   5. 在自定义控件中 封装对外接口
      1. SpinBox和 slider控件交互
      2. getData获取显示数字
      3. setData设置数字
2. Qt鼠标常用事件
   1. 鼠标进入
      1. EnterEvent
   2. 鼠标离开
      1. LeaveEvent
   3. 鼠标按下
      1. MousePressEvent
   4. 鼠标释放
      1. MouseReleaseEvent
   5. 鼠标移动
      1. MouseMoveEvent
   6. 获取 x y 坐标 ev->x() ev->y()
   7. 判断如果是左键 按下
      1. ev->button() 移动 ev->buttons() &
   8. 设置鼠标追踪状态
      1. setMouseTracking(true);
3. 定时器
   1. 定时器事件 void timerEvent( QTimerEvent \* e)
   2. 启动定时器 id1 = startTimer(毫秒)
   3. 判断具体定时器标准 e->timerId() == id1
4. 定时器2
   1. 通过定时器类实现 QTimer类
   2. 创建定时器对象
      1. QTimer \* timer1 = new QTimer(this);
   3. 开启定时器
      1. timer1->start(x毫秒)
   4. 每隔x毫秒 会抛出一个信号timeout出来
   5. 监听信号处理逻辑
   6. 暂停定时器 timer1->stop();
5. 事件分发器
   1. 用途：用于分发事件
   2. 原型 bool event(QEvent \* e)；
   3. 返回值如果是 true代表用户自己处理事件，不在向下分发
6. 事件过滤器
   1. 步骤1 给控件安装过滤器
      1. ui->label->installEventFilter(this);
   2. 步骤2 重写过滤器事件
      1. bool Widget::*eventFilter*(QObject \* obj , QEvent \* e)
7. QPainter绘图
   1. 绘图事件 void paintEvent(QPaintEvent \*);
   2. 创建画家 QPainter painter( this ) this代表绘图设备
   3. 画线、圆、矩形、文字
   4. 设置画笔 QPen painter.setPen() 设置笔宽度、风格
   5. 设置画刷 QBrush painter.setBrush() 设置画刷风格
8. 绘图高级设置
   1. 设置抗锯齿
      1. painter.setRenderHint(QPainter::Antialiasing);
   2. 移动画家
      1. painter.translate(100,0);
   3. 保存画家状态
      1. painter.save();
   4. 取出画家状态
      1. painter.restore();
   5. 利用画家画资源图片
      1. painter.drawPixmap(posX,0,QPixmap(":/Image/Luffy.png"));
   6. 手动调用绘图事件
      1. update()
9. 绘图设备
   1. QPixmap
      1. 对不同的平台显示做了优化
   2. QImage
      1. 可以对像素级进行访问
   3. QPicture
      1. 记录和重新绘图指令
10. 文件读写
    1. 读文件
       1. QFile file(文件路径)
       2. file.open 指定打开方式 QIODevice::ReadOnly
       3. 利用 file.readAll一次性全部读取
       4. 利用 file.readLine 按行读取
       5. file.close 关闭文件对象
       6. 读取gbk格式
          1. QTextCodec \* codec = QTextCodec::codecForName("gbk");
          2. codec->toUnicode(array)
    2. 写文件
       1. QFile file(路径)
       2. file.open(QIODevice::Append); //QIODevice::Append追加的方式写文件
       3. file.write("啊啊啊啊!");
       4. file.close();
11. 文件信息类
    1. QFileInfo info(路径)
    2. info获取文件信息
    3. 后缀名： suffix 文件大小 size 文件名 filename 文件路径 filePath
    4. 创建日期 created 修改日期 lastModify
    5. QDateTime 需要用 toString转为自己制定的输出格式