作业报告

项目名称：图像处理工具　　队伍：神人队

（一）程序功能介绍

一、项目概述

本项目为一款基于Qt框架，通过c++实现的简易图像处理工具，支持对绝大多数格式(如bmp,jpg,jpeg,png等)图像进行基础操作与高级特效处理。工具核心功能包括图像旋转裁切、调色处理、风格化、马赛克化等，同时扩展叠加水印和画笔等实用功能以满足多样化需求。主要思路为通过QPixmap和QImage类进行像素位操作，或依靠透明图层叠加实现图像的处理。

二、功能模块设计

1. 基础功能

功能名称	描述	实现方法
图像旋转	支持 90°、180°、270°及任意角度旋转	矩阵变换，双线性插值处理非直角旋转
灰度化	将彩色图像转换为灰度图	加权平均法
图像缩放	按比例或指定尺寸缩放图像	双线性插值或最近邻插值算法
反色处理	反转图像所有像素颜色	对 RGB 通道取反
马赛克	对指定区域进行像素块模糊	将区域划分为小方块，取块内像素平均值

2. 高级功能

功能名称	描述	实现方法
直方图均衡化	增强图像对比度	计算像素分布直方图，重新映射灰度值
二值化	将图像转换为黑白两色	设定阈值，高于阈值为白，低于为黑
对比度/亮度调整	调整图像整体对比度与亮度	线性或非线性变换
文字/水印叠加	在图像指定位置添加文本或 LOGO 水印	像素叠加与透明度混合
画笔	支持在图像上自由绘制手绘线条、图形	添加额外的透明图层

三、技术实现方案

1. **通过 QPixmap 实现对所有类型图片的转换和处理**
2. **像素级操作**
    - 通过二维数组或矩阵存储像素数据，支持快速读写与算法处理。
3. **算法优化**
    - 使用空间换时间策略缓存中间结果（如直方图数据）。
    - 多线程加速大规模像素计算（如模糊滤镜）。
4. **依靠透明图层叠加实现**

（二）项目各模块与类设计细节

AdjustDialog（调整对话框）

1. 功能概述：AdjustDialog 是一个基于 Qt 的对话框类，用于实现一个图像缩放因子的调整界面。用户可以通过滑动滑块来实时调整缩放因子，并通过信号机制将调整后的值传递给其他模块。
2. 类设计细节
基类：继承自 QDialog，提供标准的对话框功能。
信号：factorChanged(double factor)：当滑块值变化时发出，传递当前的缩放因子。
成员变量：
slider：水平滑块控件，用于用户交互。
factorLabel：标签控件，显示当前的缩放因子值。
成员函数：
AdjustDialog(QWidget *parent)：构造函数，初始化界面并连接信号槽。
setupUI()：私有函数，负责初始化界面布局和控件样式。
3. 核心逻辑
滑块值到缩放因子的转换：滑块范围：0 到 100，默认值为 50（中点）。
通过公式将滑块值转换为缩放因子：缩放因子范围：0.2（最小）到 5.0（最大），中点时为 1.0（原始大小）。
实时更新 factorLabel 显示当前缩放因子，并发射 factorChanged 信号。
4. 界面设计
布局：垂直布局（QVBoxLayout），包含一个标签和一个滑块。对话框背景：白色。
控件样式：factorLabel：字体颜色为黑色。 slider：自定义样式，包括黑色背景和黑色滑块手柄。
5. 信号与槽机制
当滑块值变化时（valueChanged），触发槽函数计算缩放因子并更新界面。
通过 factorChanged 信号将缩放因子传递给其他模块，实现实时交互。
6. 关键点
对数缩放：使用对数转换实现非线性缩放，使滑块调整更符合直观感受（小范围更敏感，大范围更平滑）。
实时反馈：滑块移动时立即更新标签和发射信号，提供流畅的用户体验。
样式定制：通过 Qt 样式表（setStyleSheet）自定义控件外观，增强视觉效果。

ClickableWidget（可点击控件）

1. 功能概述：ClickableWidget 是一个自定义的 Qt 控件，继承自 QWidget，用于实现一个可点击的区域。当用户点击该控件时，会触发 clicked 信号，并传递关联的文件路径（filePath）。该控件通常用于实现图片或其他资源的点击交互功能。
2. 类设计细节
基类：继承自 QWidget，提供基础的控件功能。
成员变量：filePath：QString 类型，存储与控件关联的文件路径。
信号：clicked(const QString& path)：当控件被点击时发出，传递 filePath。
成员函数：
ClickableWidget(const QString& path, QWidget* parent)：构造函数，初始化控件并绑定文件路径。
mousePressEvent(QMouseEvent* event)：重写父类方法，处理鼠标点击事件。
3. 核心逻辑
点击事件处理：当用户点击控件时，触发 mousePressEvent。在事件处理中，发射 clicked 信号，并将 filePath 作为参数传递。调用父类的 mousePressEvent 确保默认行为（如焦点切换）正常执行。
4. 界面设计
UI 文件：clickablewidget.ui 定义了控件的基本属性，如默认大小（320x240）和窗口标题（"Form"）。
动态功能：UI 文件仅提供静态布局，实际点击功能通过代码实现（mousePressEvent）。
5. 信号与槽机制

控件通过 clicked 信号与外部模块通信，例如：点击图片控件后，主窗口接收信号并加载对应路径的图片。与其他模块（如预览窗口、编辑器）联动，实现交互逻辑。
6. 关键点
轻量级设计：仅包含核心点击功能，通过信号传递数据，耦合度低。
灵活性：filePath 可在构造时绑定，支持动态更新路径。
可扩展性：可进一步重写其他鼠标事件（如双击、悬停）以增强交互。

ClickImageDialog（点击图片选择位置对话框）

1. 功能概述：ClickImageDialog 是一个基于 Qt 的对话框，用于显示一张缩放后的图片，并允许用户通过点击图片选择位置。点击后，对话框会将点击位置转换为原图坐标并发射 positionSelected 信号，随后自动关闭。主要用途是为水印或其他图像编辑功能提供交互式位置选择。
2. 类设计细节
基类：继承自 QDialog，提供标准对话框功能。
成员变量：
    label：QLabel 控件，用于显示缩放后的图片。
    imagePixmap：QPixmap 类型，存储原始图片。
    scaleFactor：double 类型，记录图片的缩放比例。
信号：positionSelected(QPoint pos)：当用户点击图片时发出，传递点击位置在原图中的坐标。
成员函数：
    ClickImageDialog(const QPixmap &pixmap, QWidget *parent)：构造函数，初始化对话框并加载图片。
    mousePressEvent(QMouseEvent *event)：重写父类方法，处理鼠标点击事件。
3. 核心逻辑
图片缩放与显示：构造函数中，图片按固定宽度（600 像素）缩放，高度按比例调整。
缩放比例 scaleFactor 计算为：目标宽度 / 原图宽度。
缩放后的图片通过 QLabel 显示，并居中布局。
坐标转换：用户点击位置（event->pos()）是相对于对话框窗口的坐标。
转换为 QLabel 内的坐标：clickedPoint - label->pos()。
通过缩放比例反向映射到原图坐标：使用 qBound 确保坐标不越界。
4. 关键点
坐标精确映射：通过缩放比例和控件位置计算，确保点击位置准确对应原图坐标。
资源管理：Qt::WA_DeleteOnClose 避免内存泄漏。
实时反馈：点击后立即发射信号，适合需要快速交互的场景。

ContrastDialog（对比度调整对话框）

1. 功能概述：ContrastDialog 是一个基于 Qt 的对话框，用于调整图像的对比度。用户通过滑动滑块来改变对比度值，界面会实时显示当前对比度系数，并通过信号 contrastChanged 将调整后的值传递给其他模块。
2. 类设计细节
基类：继承自 QDialog，提供标准对话框功能。
成员变量：slider：QSlider 控件，用于用户交互调整对比度。valueLabel：QLabel 控件，显示当前对比度值。
信号：contrastChanged(double factor)：当滑块值变化时发出，传递对比度系数（范围 0.0 ~ 3.0）。
成员函数：
    ContrastDialog(QWidget *parent)：构造函数，初始化界面并连接信号槽。
    setupUI()：私有函数，负责初始化界面布局和控件样式。
3. 核心逻辑
滑块值与对比度系数的映射：滑块范围：0 到 300，默认值为 100（对应对比度系数 1.0）。
对比度系数计算：factor = value / 100.0，范围为 0.0 ~ 3.0。
实时更新 valueLabel 显示当前对比度系数（保留两位小数）。

信号与槽机制：滑块值变化时（valueChanged），触发槽函数计算对比度系数并更新界面。

CropWindow（图像裁剪窗口）

1. 功能概述：CropWindow 是一个基于 Qt 的自定义窗口，用于交互式地裁剪图像。用户可以通过鼠标拖拽选择裁剪区域，确认后窗口会发射 croppedPixmap 信号，传递裁剪后的图像，并自动关闭。核心功能包括图像缩放显示、区域选择和坐标映射。
2. 类设计细节
基类：继承自 QWidget，提供基础的窗口功能。
成员变量：
    originalPixmap：原始图像。
    displayPixmap：缩放后的图像（用于显示）。
    rubberBand：QRubberBand 控件，用于可视化选择区域。
    origin：QPoint 类型，记录鼠标拖拽起始位置。
    scaleFactor：缩放比例（显示图像尺寸与原图尺寸的比值）。
信号：croppedPixmap(const QPixmap &croppedPixmap)：裁剪完成后发射，传递裁剪后的图像。
3. 核心逻辑
图像缩放：构造函数中，图像按最长边 600 像素等比例缩放，计算缩放比例 scaleFactor。
缩放后的图像通过 QPalette 设置为窗口背景。
坐标映射：
显示坐标到原图坐标的转换：原图坐标 = 显示坐标 / scaleFactor。
4. 关键点
缩放与精度：通过缩放显示大图，但最终裁剪基于原图坐标，保证精度。
资源管理：rubberBand 动态创建和隐藏，避免内存泄漏。
实时性：鼠标移动时实时更新选择框，操作流畅。

ImageLabel（图像显示标签）

1. 功能概述 ImageLabel 是一个自定义的 QLabel 控件，用于显示图像并支持两种模式：
    自适应模式：自动缩放图像以适配控件大小，保持宽高比。
    手动模式：允许用户通过鼠标拖拽查看图像的不同区域（适用于大图浏览）。
2. 类设计细节
基类：继承自 QLabel，提供基础的标签功能。
成员变量：
    image：QPixmap 类型，存储当前显示的图像。
    offset：QPoint 类型，记录手动模式下的图像偏移量。
    manualMode：bool 类型，标记当前是否为手动模式。
    dragging：bool 类型，标记是否正在拖拽图像。
    lastMousePos：QPoint 类型，记录拖拽时的上一鼠标位置。
成员函数：
    setImage：设置显示的图像。
    setOffset：设置手动模式下的图像偏移量（自动限制边界）。
    enableManualMode：切换手动/自适应模式。
重写事件：paintEvent、mousePressEvent、mouseMoveEvent、mouseReleaseEvent。
3. 核心逻辑
图像显示：自适应模式：图像缩放至控件大小，居中显示，保持宽高比。
手动模式：根据 offset 绘制图像的可见部分，用户可通过拖拽调整偏移量。
拖拽交互：
    鼠标按下时记录起始位置，进入拖拽状态（dragging = true）。

鼠标移动时计算位移差，更新 offset 并重绘图像。
鼠标释放时结束拖拽状态。
边界处理：setOffset 中通过 std::clamp 确保偏移量不超出图像边界：
4. 界面反馈
鼠标样式：拖拽时变为 Qt::ClosedHandCursor，释放后恢复默认箭头。
实时绘制：拖拽过程中动态更新可见区域，提供流畅体验。
5. 关键点
模式切换：通过 manualMode 灵活控制显示逻辑。
性能优化：仅绘制可见部分（手动模式），避免不必要的缩放计算。
用户体验：拖拽时的视觉反馈（光标变化、实时位移）增强交互直观性。

ImportWindow（图片导入窗口）

1. 功能概述：ImportWindow 是图片编辑器的启动窗口，提供三个核心功能：
新建图片：通过文件对话框选择图片文件并加载到主编辑器。
打开最近文件：从历史记录中选择最近编辑的图片。
退出程序：直接关闭应用。
2. 类设计细节
基类：继承自 QDialog，作为模态对话框使用。
UI 组件：
三个图标按钮：newpic（新建）、openpic（打开最近）、exit（退出）。
底部标签：显示版权信息。
成员函数：initButtons：初始化按钮样式（图标、透明背景）。paintEvent：绘制背景图片。
槽函数：处理按钮点击事件（on_xxx_clicked）。
3. 核心逻辑
新建图片（on_newpic_clicked）：调用 QFileDialog 选择图片文件（支持多种格式：BMP/PNG/JPG 等）。
创建 MainWindow 实例，加载图片并显示主编辑器，关闭当前窗口。
打开最近文件（on_openpic_clicked）。
加载历史记录文件列表（MainWindow::recentFiles）。
弹出 RecentFilesWindow 对话框供用户选择。
选择后更新历史记录，打开主编辑器并加载图片。
4. 界面设计
布局：水平布局（QHBoxLayout）排列三个功能按钮。固定窗口大小（320x240），背景图片铺满。
按钮样式：透明背景，50x50 像素，通过 setButtonStyle 统一设置图标。
资源路径：背景图：:/icons/mainlogo.jpg。按钮图标：:/icons/newpic.png、:/icons/openpic.png、:/icons/exit.png。
5. 关键点
多格式支持：文件对话框过滤常见图片格式（如 BMP/PNG/JPG）。
历史记录管理：通过 MainWindow::recentFiles 静态变量和 RecentFilesWindow 实现。
资源嵌入：使用 Qt 资源系统（:/icons/xxx）管理图标和背景图。

LightnessDialog（亮度调整对话框）

1. 功能概述：LightnessDialog 是一个基于 Qt 的对话框，用于调整图像的亮度。用户通过滑动滑块改变亮度值（范围 -100 到 100），界面实时显示当前值，并通过 lightnessChanged 信号将调整值传递给其他模块。
2. 类设计细节
基类：继承自 QDialog，提供标准对话框功能。
成员变量：
slider：水平滑块控件（QSlider），范围 -100 到 100。valueLabel：标签控件（QLabel），显示当前亮度值。
信号：lightnessChanged(int delta)：滑块值变化时发射，传递亮度调整值（-100 ~ 100）。

成员函数：LightnessDialog：构造函数，初始化界面并连接信号槽。setupUI：私有函数，设置布局和控件样式。
3. 核心逻辑
滑块交互：滑块范围设置为 -100 到 100，默认值为 0（无调整）。
值变化时，更新 valueLabel 显示当前值，并发射 lightnessChanged 信号。
信号传递：其他模块（如主编辑器）可连接 lightnessChanged 信号，实时调整图像亮度。
4. 关键点
实时反馈：滑块移动时立即更新标签和发射信号，提供流畅交互。
值范围：-100（最暗）到 100（最亮），0 为原始亮度，符合直观认知。
样式统一：黑色滑块与白色背景形成对比，风格简洁。

MosaicDialog（马赛克效果调整对话框）

1. 功能概述：MosaicDialog 是一个基于 Qt 的对话框，用于调整图像马赛克效果的像素块大小。用户通过滑动滑块选择块大小（范围 1~100），界面实时显示当前值，并通过 blockSizeChanged 信号将调整值传递给其他模块。
2. 类设计细节
基类：继承自 QDialog，提供标准对话框功能。
成员变量：
slider：水平滑块控件（QSlider），范围 1 到 100。
blockSizeLabel：标签控件（QLabel），显示当前块大小值。
信号：blockSizeChanged(int blockSize)：滑块值变化时发射，传递马赛克块大小（1 ~ 100）。
成员函数：MosaicDialog：构造函数，初始化界面并连接信号槽。setupUI：私有函数，设置布局和控件样式。
3. 核心逻辑
滑块交互：滑块范围设置为 1 到 100，默认值为 1（最小马赛克效果）。
值变化时，更新 blockSizeLabel 显示当前值，并发射 blockSizeChanged 信号。
信号传递：其他模块（如主编辑器）可连接 blockSizeChanged 信号，实时调整马赛克效果的块大小。
4. 关键点
实时反馈：滑块移动时立即更新标签和发射信号，提供流畅交互。
值范围：1（轻微马赛克）到 100（强烈马赛克），符合马赛克效果的特性。
样式统一：黑色滑块与白色背景形成对比，风格简洁。

PenSettingDialog（画笔设置对话框）

1. 功能概述：PenSettingDialog 是一个用于设置画笔参数的对话框，主要功能包括：
调整画笔宽度（1-30 像素），选择画笔颜色（支持颜色选择器），设置透明度（0-255）
实时预览颜色选择，确认/取消操作
2. 类设计细节
核心成员：
widthSpinBox QSpinBox 画笔宽度选择
opacitySlider QSlider 透明度调节
colorButton QPushButton 颜色选择按钮
selectedColor QColor 当前选择的颜色
信号：settingsConfirmed(int, QColor, int)：传递最终设置的宽度、颜色和透明度
3. 关键特性
样式统一：黑白主色调，一致的控件样式，悬停效果
交互设计：颜色按钮实时显示当前颜色，合理的参数范围限制，标准对话框按钮布局
数据封装：提供 getter 方法获取各参数，通过信号传递设置结果

PenWindow（绘图窗口）

1. 功能概述：PenWindow 是一个基于 Qt 的绘图对话框，允许用户在图像上自由绘制。主要功能包括：
支持手绘线条，可配置画笔参数（颜色、宽度、透明度）
实时预览绘制效果，确认/取消操作
2. 类设计细节
核心成员变量

originalPixmap	QPixmap	原始图像
currentPreviewPixmap	QPixmap	当前预览图像
baseImage	QImage	实际绘制的图像
scaleFactor	double	显示缩放比例
penColor	QColor	画笔颜色
penWidth	int	画笔宽度
penOpacity	int	画笔透明度

信号：
drawingConfirmed：确认绘图时发射，传递最终图像
drawingCancelled：取消绘图时发射
3. 关键特性
交互设计：流畅的手绘体验，坐标精确映射（处理缩放），抗锯齿绘制
性能优化：基于 QImage 的直接像素操作，仅更新变化区域（当前实现是全图更新）
UI 一致性：与主窗口相同的黑白风格，标准化按钮布局

ProcessImage（图像处理工具类）

1. 功能概述：ProcessImage 是一个静态工具类，提供了一系列图像处理功能，包括旋转、翻转、缩放、颜色调整（灰度化、反色、二值化）、特效处理（马赛克）以及亮度/对比度/饱和度调整。所有方法均为静态，可直接调用无需实例化。
2. 类设计细节
静态工具类：无成员变量，所有方法均为 static，通过 QPixmap 参数输入输出。
核心功能：
几何变换：旋转（90 度/任意角度）、翻转（水平/垂直）、缩放。
颜色处理：灰度化、反色、二值化。
特效处理：马赛克效果。
色彩调整：亮度、对比度、饱和度。
3. 核心逻辑与实现
几何变换：
旋转：通过 QTransform 实现，支持任意角度（rotate）和固定 90 度（rotate90）。
翻转：利用 QTransform.scale(-1, 1) 实现水平/垂直镜像。
缩放：根据缩放因子计算新尺寸，保持宽高比（Qt::KeepAspectRatio）。
颜色处理：
灰度化：使用公式 0.299*R + 0.587*G + 0.114*B 计算灰度值。
反色：对 RGB 分量取反（255 - value）。
二值化：根据阈值将灰度值转为黑白（0 或 255）。
特效处理：
马赛克：将图像分块，每块内像素替换为块内颜色的平均值。
色彩调整：
亮度：对 RGB 分量线性加减（delta 范围建议 [-100, 100]）。
对比度：基于偏离中值（128）的比例缩放。
饱和度：转换为 HSL 色彩空间后调整 S 分量。
4. 关键点
像素级操作：大部分方法通过 QImage 直接操作像素，保证效率。

透明通道保留：始终检查 qAlpha(px) 跳过透明像素。
边界保护：使用 qBound 或手动检查防止越界。
性能优化：避免重复计算（如马赛克的块内平均值）。

RecentFilesWindow（最近文件窗口）

1. 功能概述：RecentFilesWindow 是一个基于 Qt 的对话框，用于显示用户最近访问的图片文件列表。每个文件以缩略图+文件名形式展示，支持点击打开或清空历史记录。核心功能包括：
缩略图网格：两列布局展示文件缩略图和名称。
交互操作：点击文件触发打开，点击按钮清空记录。
自动过滤：跳过不存在或无法加载的图片文件。
2. 类设计细节
基类：继承自 QDialog，作为模态对话框使用。
成员变量：
无显式成员变量，通过构造函数参数 recentFiles 接收数据。
信号：fileSelected：用户选择文件时发射，传递文件路径。clearRequested：用户点击清空按钮时发射。
3. 核心逻辑
界面构建：
    滚动区域：QScrollArea 包裹网格布局，支持大量文件浏览。
    网格项：每个文件对应一个 ClickableWidget（自定义可点击控件），包含：
    缩略图（缩放至 100x100，保持宽高比）。
    文件名标签（居中显示，自动换行）。
    清空按钮：红色样式，触发 clearRequested 信号。
交互逻辑：
    点击文件项时，发射 fileSelected 并关闭窗口。
    点击清空按钮时，发射 clearRequested 并关闭窗口。
数据过滤：
    检查文件是否存在（QFile::exists）。
    检查图片是否有效（QPixmap.isNull）。
4. 关键点
性能优化：缩略图动态生成，避免直接加载大图。
用户体验：鼠标悬停时变为手型光标（Qt::PointingHandCursor）。
文件名自动换行，避免截断。
健壮性：自动跳过无效文件，避免崩溃。

RotateDialog（图像旋转对话框）

1. 功能概述：RotateDialog 是一个基于 Qt 的对话框，用于调整图像的旋转角度。用户通过滑动滑块选择旋转角度（范围 -180° 到 180°），界面实时显示当前角度值，并通过 angleChanged 信号将调整值传递给其他模块。
2. 类设计细节
基类：继承自 QDialog，提供标准对话框功能。
成员变量：
    slider：水平滑块控件（QSlider），范围 -180 到 180。
    angleLabel：标签控件（QLabel），显示当前角度值（带°符号）。
信号：angleChanged(int angle)：滑块值变化时发射，传递旋转角度（-180 ~ 180）。
成员函数：RotateDialog：构造函数，初始化界面并连接信号槽。setupUI：私有函数，设置布局和控件样式。
3. 关键点
实时反馈：滑块移动时立即更新标签和发射信号，提供流畅交互。
角度范围：-180°（逆时针）到 180°（顺时针），覆盖完整旋转需求。

样式统一：黑色滑块与白色背景形成对比，风格简洁。

SaturationDialog（饱和度调整对话框）

1. 功能概述：SaturationDialog 是一个基于 Qt 的对话框，用于调整图像的饱和度。用户通过滑动滑块选择饱和度系数（范围 0.0~3.0），界面实时显示当前值，并通过 saturationChanged 信号将调整值传递给其他模块。
2. 类设计细节
基类：继承自 QDialog
核心组件：QSlider：范围 0~300（对应 0.0~3.0）QLabel：显示当前饱和度系数（保留 2 位小数）
信号：saturationChanged(double)：传递调整后的饱和度值
3. 核心逻辑
值映射：滑块值除以 100 得到实际系数（100→1.0，300→3.0）默认值设为 100（1.0 倍原始饱和度）
实时交互：滑块移动时立即更新标签显示同步发射信号通知主窗口
4. 关键特性
非线性调整：支持从完全去色（0.0）到超饱和（3.0）
精确控制：保留两位小数显示
视觉反馈：实时数值更新

WatermarkDialog（水印添加对话框）

1. 功能概述：WatermarkDialog 是一个基于 Qt 的对话框，用于为图像添加自定义文字水印。
2. 类设计细节
基类：继承自 QDialog
核心组件：
QLineEdit：水印文字输入 QSpinBox：字体大小选择 QColorDialog：颜色选择器 QSlider：透明度调节
信号：confirmed()：传递水印参数（文字、字体、颜色、透明度）
3. 核心逻辑
参数收集：通过各控件获取用户输入，颜色选择使用系统颜色对话框
样式控制：统一黑色边框+白色背景风格，实时更新颜色按钮背景
数据传递：点击确认时打包所有参数发射信号
4. 关键特性
完整的水印控制：支持文字所有关键属性
实时预览：颜色按钮即时反馈
透明度调节：0（全透明）到 255（不透明）

三、小组成员分工情况

在整个 SimplePicEditor 修图软件的构建过程中，"神人队"小组组长孙海铭同学负责实现修图软件的各种功能，包括图像旋转（默认的特定角度、指定的任意角度）、图像缩放、图像的灰度化、反色、二值化、马赛克、对比度和饱和度的调节等；同时兼顾文件解析、算法优化等工作。小组成员刘津铭同学负责基本功能（图像旋转、灰度化、缩放、反色处理、马赛克）的函数编写，如：针对图像旋转的处理问题，利用图像变换和矩阵处理旋转逻辑以及双线性插值方法，通过 QPixmap 实现；反色处理，则是对像素点的 RGB 通道值取反，等等。马达同学则是负责直方图均衡化、图像二值化、图像对比度/亮度的调节、文本/水印叠加等。具体的实现方法如：通过计算像素分布直方图映射灰度值的方法，增强图像对比度；设定白、黑色阈值衡量像素点取值的方法，将所给图像二值化，等等。

四、项目总结与反思

总而言之，“神人队”小组的基于命令行或简易图形界面的图像编辑工具具有诸多优点，如：

1. 目标定位精准，聚焦 BMP 格式特性（后扩展至全格式兼容）。初期聚焦 BMP 的无压缩特性，解决 JPEG 编解码复杂度高的问题，降低开发难度；开发过程中发现 QPixmap 格式可兼容所有图片格式，同时支持像素级操作（如矩阵变换、双线性插值），因此将工具从 BMP Editor 升级为 PiceEditor，实现对 JPG、PNG 等全格式图片的处理，无需单独适配不同格式的编解码逻辑，大幅扩展工具适用性。

2. 功能模块设计系统化，覆盖场景全面。该工具具备旋转、灰度化、缩放等基础功能，反色、马赛克等满足基本修图需求，修改对比度、饱和度、二值化等操作符合专业图像处理场景。

3. 技术方案可行性高，兼顾工程实践。具备清晰的文件解析逻辑，采用“多线程加速”和“空间换时间”策略提升性能。
4. 架构设计具备扩展性。文件解析、像素处理、算法实现分离，新增滤镜（如高斯模糊）时无需修改核心架构；通过二维数组/矩阵存储像素数据，为对接第三方库（如 OpenCV）提供可能。

5. 格式兼容性升级，依托 QPixmap 实现全格式支持。开发中优化技术方案，利用 QPixmap 同时兼容多格式图片输入与像素级操作（如 RGB 通道修改、插值计算），彻底摆脱对 BMP 格式的依赖，使工具从单一格式编辑器升级为通用图像处理器，显著提升实用性。

当然，现在的修图软件还是存在一些不足之处的。如：未定义任意角度旋转后的“黑边填充策略”（如背景色填充、边缘拉伸），可能导致图像显示不完整；仅支持 24/32 位真彩色 BMP，未处理 8 位索引色图像（含调色板），可能导致解析时颜色错乱；未严格遵循 BMP“行对齐规则”（每行字节数需为 4 的倍数），大尺寸图像解析可能引发内存越界；采用二维数组存储像素，对 4K 及以上分辨率图像可能导致内存溢出；未利用 Qt 的 QImage 类内置优化（如像素缓存、格式转换接口）等。具体的改进，可以考虑补充旋转锚点设置、局部滤镜处理、增加实时预览、进度反馈、辅助构图工具，支持参数滑块调节，将会使这个工具更加健壮。

另外，我们在开发之初低估了stylesheet的复杂性与标准化不同参数滑块等“调节dialog”类的继承和派生的重要性。到最后，添加越来越多的功能后，要实现图形界面风格的简洁统一，便需要费很大功夫；而因为已开发的内容已经不少，我们最后也只能选择继续重复添加和设置，在事实上造成了一定的内容重复和资源浪费。我们意识到，在之后的开发中，项目的整体规划和接口在一开始就要处理好，这让我们受益匪浅。