49-作业报告——“文件分类助手”

一、程序功能介绍

“文件分类助手”是一款桌面文件整理工具，旨在帮助您告别混乱的文件夹，轻松实现文件的自动化归档。

核心功能亮点

1. **多样化的分类策略：**
    - **按文件类型：**自动识别文件的后缀名（如 .jpg，.pdf，.docx），将相同类型的文件归集到一起。支持将数量稀少的类型智能合并为“其他”类别。
    - **按文件体积：**允许您自定义文件大小区间（如“小于 1MB”、“1MB-100MB”、“大于 100MB”），对不同体积的文件进行分类。
    - **按修改时间：**根据文件的最后修改日期进行整理，您可以灵活设置“最近 7 天内”、“最近 3 个月内”或“1 年内”等规则。
2. **交互式预览，安全第一：**
    - 在执行任何实际的文件移动前，程序会生成一个可视化的分类预览界面。
    - 您可以清晰地看到哪些文件将被移动到哪个新建文件夹中。
    - 在预览阶段，您可以自由修改目标文件夹的名称，或者取消勾选任何不想移动的单个文件。
3. **内置文件查看器：**
    - 在预览界面，无需离开程序，即可直接点击“预览”按钮查看图片和文本文档的内容，方便您快速决策。
4. **可靠的执行与撤销机制：**
    - 文件移动过程由一个清晰的进度条展示。
    - 最核心的保障是一键撤销功能。无论是在处理过程中还是在全部分类完成之后，只要您点击“撤销”，所有被移动的文件都会被安全地恢复到原始位置，确保您的文件万无一失。

使用流程

1. **选择目录：**启动程序，选择您需要整理的文件夹。
2. **分析与设置：**程序会快速分析文件夹内容，并以图表形式展示文件类型分布。您在右侧选择一种分类模式并设定好具体规则。
3. **预览与微调：**进入预览界面，检查自动生成的分类方案。您可以按需修改文件夹名或排除特定文件。
4. **执行或撤销：**确认无误后点击“执行”。如果中途或事后反悔，随时可以点击“撤销”来还原一切。

二、项目技术实现与类设计详解

1. 整体架构设计

本项目采用 C++17 标准，并基于 Qt 6 Widgets 框架进行开发，严格遵循了模块化和关注点分离 的设计原则。整体架构可以概括为一个多窗口、事件驱动的应用程序。

- **UI 与逻辑分离**：每个窗口的界面布局由.ui 文件（XML 格式）定义，窗口的行为和业务逻辑则在对应的.h（头文件）和.cpp（源文件）中实现。这种方式使得界面设计与后端逻辑解耦，便于维护和迭代。
- **事件驱动模型**：程序的核心交互依赖于 Qt 强大的**信号与槽 (Signals & Slots)**机制。用户的操作（如点击按钮）会发射一个信号，该信号会触发与之连接的槽函数，从而执行相应的业务逻辑。
- **窗口流转**：程序的执行流程被设计为一系列模态对话框 (Modal Dialog) 的线性流转，确保了用户操作的聚焦和流程的清晰性：
    1. MainWindow (路径选择)
    2. classificationWindow (分析与方案选择)
    3. PreviewWindow / SizePreviewWindow / TimePreviewWindow (分类预览)
    4. ExecuteWindow (执行与撤销)

2. 核心模块与类详细设计

2.1. 启动与路径选择模块 (MainWindow)

- **相关文件**: mainwindow.h, mainwindow.cpp, mainwindow.ui
- **核心类**: MainWindow
- **职责**: 入口窗口，提供一个界面，让用户选择要文件整理的根目录。
- **设计细节**:
    - 界面极其简洁，只有一个核心的 choseFileButton 按钮。
    - 槽函数 on_choseFileButton_clicked() 是其核心逻辑：
        1. 弹出一个临时的 QDialog，内置路径输入框和“浏览”按钮。
        2. 使用 QFileDialog::getExistingDirectory 来调起系统标准的文件夹选择器。
        3. 用户确认选择后，MainWindow 会保存所选路径，然后隐藏自身。
        4. 实例化核心的 classificationWindow，并将选择的路径作为构造函数参数传入，然后以模态方式 (exec()) 显示它。

2.2. 分析与分类中心模块 (ClassificationWindow)

- **相关文件**: classificationwindow.h, classificationwindow.cpp, classificationwindow.ui
- **核心类**: classificationWindow
- **职责**: 这是程序的功能中枢。它接收 MainWindow 传来的路径，对该路径下的文件进行统计分析，通过图表进行可视化展示，并提供三种分类方案的入口。
- **设计细节**:
    - **数据成员**:
        - selectedPath: 保存当前操作的根目录路径。
        - fileTypeChart, fileTypeChartView: Qt Charts 模块的实例，用于绘制和显示文件类型占比的饼图。
        - 一系列布尔型标志位（如 is_type1_activated, is_smallKB_used）用于记录用户在 UI 上选择的分类子选项，直接控制后续的分类逻辑。
    - **核心方法**:
        - updateFileStatistics(): 窗口初始化和用户更改分类选项时被调用。它会遍历所有文件，统计数量、体积和类型，然后调用 updateChart()来刷新饼图。
        - initChart() / updateChart(): 负责饼图的创建、样式设置和数据填充。设计上将标签外置，避免了拥挤，提升了可读性。
        - on_pushButton_..._clicked() (三个分类按钮的槽函数):
            1. 调用 collectAllFiles()收集所有文件信息。
            2. 根据当前分类模式和 UI 上的选项（如复选框、输入框的值），对文件列表进行逻辑分组，生成一个 QMap 数据结构。这个 Map 的键是分类名（如 "pdf", "< 1MB"），值是该分类下的文件列表。
            3. 实例化对应的预览窗口 (PreviewWindow, SizePreviewWindow, 或 TimePreviewWindow)。
            4. 将分组后的 QMap 数据和根路径传入预览窗口。
            5. 以模态方式 (exec()) 显示预览窗口，等待用户下一步操作。

2.3. 交互式预览模块（系列窗口）

这是本项目中设计模式应用得最好的模块，体现了组合优于继承的思想。三个预览窗口 (PreviewWindow, SizePreviewWindow, TimePreviewWindow) 共享一套几乎完全相同的架构，只是处理的数据类型不同。

- **通用架构**:
    - **外层容器**: QDialog -> QScrollArea -> QWidget (内容面板) -> QHBoxLayout。这种结构实现了一个可以水平滚动的、用于容纳多个分类卡片的区域。
    - **分类卡片 (Category Widget)**: FileTypeWidget, FileSizeTypeWidget, FileTimeTypeWidget。它们都继承自

QFrame，负责展示一个分类（如 "pdf" 类）的所有信息，包括自定义文件夹名、文件总数和一个文件列表。
    - **文件项 (Item Widget)**: FileItemWidget, FileSizeItemWidget, FileTimeItemWidget。它们是列表中的最小单元，负责显示单个文件的信息和一个可交互的“选中/未选中”按钮及“预览”按钮。
- PreviewWindow (按类型)
    - **核心类**: PreviewWindow, FileTypeWidget, FileItemWidget
    - **数据流**: 接收 QMap<QString, QStringList> 数据。PreviewWindow 为 Map 中的每个键（文件类型）创建一个 FileTypeWidget。FileTypeWidget 则为值（文件名列表）中的每个文件名创建一个 FileItemWidget。
- SizePreviewWindow (按体积)
    - **核心类**: SizePreviewWindow, FileSizeTypeWidget, FileSizeItemWidget
    - **数据流**: 接收 QMap<QString, QList<FileInfo>> 数据。这里使用了自定义结构体 FileInfo {QString fileName; qint64 fileSize; ...}，这是一个优秀的设计，将文件名和文件大小绑定在一起，避免了数据的不一致。流程与上面类似。
- TimePreviewWindow (按时间)
    - **核心类**: TimePreviewWindow, FileTimeTypeWidget, FileTimeItemWidget
    - **数据流**: 接收 QMap<QString, QList<FileTimeInfo>> 数据。同样使用了自定义结构体 FileTimeInfo，封装了文件名和修改时间。
- **信号槽交互**:
    - FileItemWidget 中的按钮被点击时，会发射 selectionChanged 信号。
    - FileTypeWidget 接收此信号，并更新其内部维护的一个 QMap<QString, bool>来追踪每个文件的选中状态。
    - 当最终执行时，PreviewWindow 会向每个 FileTypeWidget 查询其 getSelectedFiles()，从而收集所有被选中的文件。

2.4. 执行与撤销模块 (ExecuteWindow)

- **相关文件**: executewindow.h, executewindow.cpp, executewindow.ui
- **核心类**: ExecuteWindow
- **职责**: 负责执行实际的文件移动操作，提供可视化的进度反馈，并实现健壮的撤销功能。
- **设计细节**:
    - **异步处理**: 为了防止在处理大量文件时 UI 卡死，文件移动操作并没有放在一个简单的循环里，而是通过 QTimer 来实现。startFileClassification()启动定时器，定时器每隔一个很短的时间（如 20 毫秒）触发一次 updateProgress()槽函数。

- **单步执行**：updateProgress()每次只处理一个文件。它从待处理文件列表中取出一个文件，根据传入的 folderNameMap 确定目标文件夹，执行移动，然后将这次操作记录到 history 中，并更新进度条。
- **撤销机制的基石**：关键数据成员 QList<QPair<QString, QString>> history。它记录了每一次成功的文件移动，存储了文件的“新路径”和“旧路径”。
- **撤销逻辑 undoFileClassification()**：这是安全保障的核心。
  1. 反向遍历：它从 history 列表的末尾向前遍历。
  2. 恢复文件：将文件从“新路径”移回“旧路径”。
  3. 清理空目录：在移动文件后，它会检查原来的分类文件夹是否变空，如果是，则安全地删除该文件夹。这个过程会递归向上，直到遇到非空目录或根目录为止。
- **状态管理**：通过 isProcessing, isFinished 等布尔标志位来管理窗口状态，决定“完成”和“撤销”按钮的可用性和行为。

2.5. 辅助工具模块 (FilePreviewDialog)

- **相关文件**：filepreviewdialog.h, filepreviewdialog.cpp
- **核心类**：FilePreviewDialog
- **职责**：一个独立的、可重用的对话框，用于预览指定文件的内容。
- **设计细节**：
  - **类型判断**：在构造函数中，通过 isImageFile()和 isTextFile()辅助函数判断文件类型。
  - **动态创建 UI**：根据文件类型，动态创建 QLabel（用于显示图片）或 QPlainTextEdit（用于显示文本），而不是在.ui 文件中写死。
  - **内容加载**：loadFileContent()负责从磁盘读取文件。对于图片，它使用 QPixmap 加载并进行了智能缩放以适应屏幕。对于文本，它使用 QTextStream 读取，并对超大文件做了截断处理，防止程序因加载巨大文本而崩溃。
  - **高内聚**：这个类封装了文件预览的所有逻辑，与主程序其他部分完全解耦，仅通过构造函数接收一个 QFileInfo 对象即可工作。

三、小组成员分工

1、高润宇（组长）

负责前端界面设计。

2、唐天睿

主要完成程序前后端连接（将前端按钮逻辑匹配后端分类逻辑），部分 bug 修复，以及文件预览功能（辅助工具模块）。

3、赵天润
主要完成程序后端实现，如读取文件、移动操作文件等功能。

四、项目总结与反思

项目总结

本项目成功交付了一款基于 C++/Qt 的“文件分类助手”。程序通过类型、体积、时间三种维度实现文件自动化整理，其核心设计在于“先预览、后执行”的交互模式与一键撤销的安全保障机制，确保了操作的高效与安全。

亮点与展望

项目最大的亮点是其用户为中心的设计，将操作的灵活性与安全性放在首位。未来可在性能（引入多线程处理大目录）、功能（增加递归扫描与自定义规则）、预览文件（增加支持预览的文件类型）等方面进行深化与扩展。