

**程 序 设 计 专 题**

**大 程 序 报 告**



大程名称： Blockchain Editor 笔记管理系统

**指导老师： 张引**

**2022~2023春夏学期 2024 年 6 月 7 日**

**报告撰写注意事项**

1. 图文并茂。文字通顺，语言流畅，无错别字。
2. 书写格式规范，排版良好，内容完整。
3. 存在拼凑、剽窃等现象一律认定为抄袭；0分
4. 蓝色文字为说明，在最后提交的终稿版本，请删除这些文字。

**目 录**

[1 大程序简介 4](#_Toc65352773)

[1.1 选题背景及意义 4](#_Toc65352774)

[1.2 目标要求 4](#_Toc65352775)

[1.3 术语说明 4](#_Toc65352776)

[2 需求分析 4](#_Toc65352777)

[2.1 业务需求 4](#_Toc65352778)

[2.2 功能需求 4](#_Toc65352779)

[2.3 数据需求 5](#_Toc65352780)

[2.4 性能需求 5](#_Toc65352781)

[3 程序开发设计 5](#_Toc65352782)

[3.1 总体架构设计 5](#_Toc65352783)

[3.2 功能模块设计 5](#_Toc65352784)

[3.3 数据结构设计 5](#_Toc65352785)

[3.4 源代码文件组织设计 5](#_Toc65352786)

[3.5 函数设计描述 6](#_Toc65352787)

[4 部署运行和使用说明 6](#_Toc65352788)

[4.1 编译安装 6](#_Toc65352789)

[4.2 运行测试 6](#_Toc65352790)

[4.3 使用操作 7](#_Toc65352791)

[5 团队合作 7](#_Toc65352792)

[5.1 任务分工 7](#_Toc65352793)

[5.2 开发计划 7](#_Toc65352794)

[5.3 编码规范 7](#_Toc65352795)

[5.4 合作总结 7](#_Toc65352796)

[5.5 收获感言 8](#_Toc65352797)

[6 参考文献资料 8](#_Toc65352798)

Blockchain Editor 笔记管理系统设计项目

# 大程序简介

## 选题背景及意义

**项目背景：**

在现代社会，信息技术的飞速发展正在深刻影响和改变人们的工作和学习方式。特别是在数字化时代，我们日常生活中生成和处理的信息量急剧增加，如何有效地记录、管理和检索这些信息成为了一项重要的任务。

**项目意义：**

**1. 提高个人效率和管理能力**

一个功能完善的笔记管理系统，可以显著提高用户记录、整理、搜索和回顾笔记的效率。其便捷的分类和检索功能，能帮助用户快速找到所需的信息，优化了时间管理和信息处理的流程。

**2. 促进知识积累和共享**

通过数字化笔记系统，用户可以轻松记录各类学习和工作笔记，并进行系统整理和归档，有助于知识的积累和系统化管理。

**3. 支持多媒体信息记录**

与传统纸质笔记不同，数字化笔记管理系统支持文本、图片等多种形式的信息记录。这样可以更全面地捕捉和保存各种类型的信息，丰富笔记内容，满足多样化的记录需求。

**4. 提供智能化和个性化服务**

先进的笔记管理系统可以集成和数据分析技术，提供智能化的笔记整理与分析服务。比如，自动分类、关键字提取、笔记摘要生成等功能，可以极大地提升信息管理的智能化水平。

**5. 环保与可持续发展**

电子笔记管理系统减少了对纸张的依赖，符合环保和可持续发展的理念。无纸化系统节约了能源、水资源等。

## 目标要求

**基本功能要求：**

实现图形用户界面，支持用户使用各项功能

实现笔记内容结构的合理设计

支持笔记的打开、关闭和保存

支持笔记的创建、内容的文本编辑、删除

支持笔记的分级、分类、排序、搜索

提供系统使用说明或帮助功能

**（2）较高功能要求**

支持笔记图文输入编辑和管理

支持数据分析，提供用户笔记使用情况、内容知识分布情况等可视化分析

## 术语说明

**块（Block）：**一个块是内容显示的最小单元。块有不同的类型，例如文本、图片等。在许多情况下，每个块独立保存其数据和位置，通过通用接口进行操作。

**即时增量搜索：**每键入或删除一个字符即进行一次搜索，并实时显示所有搜索结果。

# 需求分析

## 业务需求

需要设计一个支持对笔记页面进行排版的笔记管理程序，具体需求如下：

1. 支持不同块的定义，不仅能支持纯文本块，还需要支持图片数据的正常显示，同时可以保留相应接口制作更多扩展块，如列表、错题集等，以简便的增强程序功能。
2. 支持块排版功能，可以分列显示、分列对齐各块。
3. 在文本块的表现类似于传统的文本编辑器，即支持光标跨块批量选择的功能。

## 功能需求

1. 具有图形用户界面。
2. 支持对块样式的设计。
3. 支持在文本块内部对某些字符样式的修改。
4. 支持传统文本编辑器的光标操作，也支持对块的位置进行一定程度的自定义。
5. 支持插入图片。
6. 提供帮助和使用说明。
7. 支持文件操作，可以打开和关闭文件。
8. 支持创建读取存档文件，以显示如编辑时间等统计内容，并在下一次打开程9. 序时快捷调出已有笔记文件。

10.支持对笔记文件添加标签，以作为分类和统计的依据。

11.支持基于存档文件进行浏览，快速打开笔记文件，或者对笔记文件进行分级、分类、排序、搜索。

12.支持基于存档文件进行的数据统计。

## 数据需求

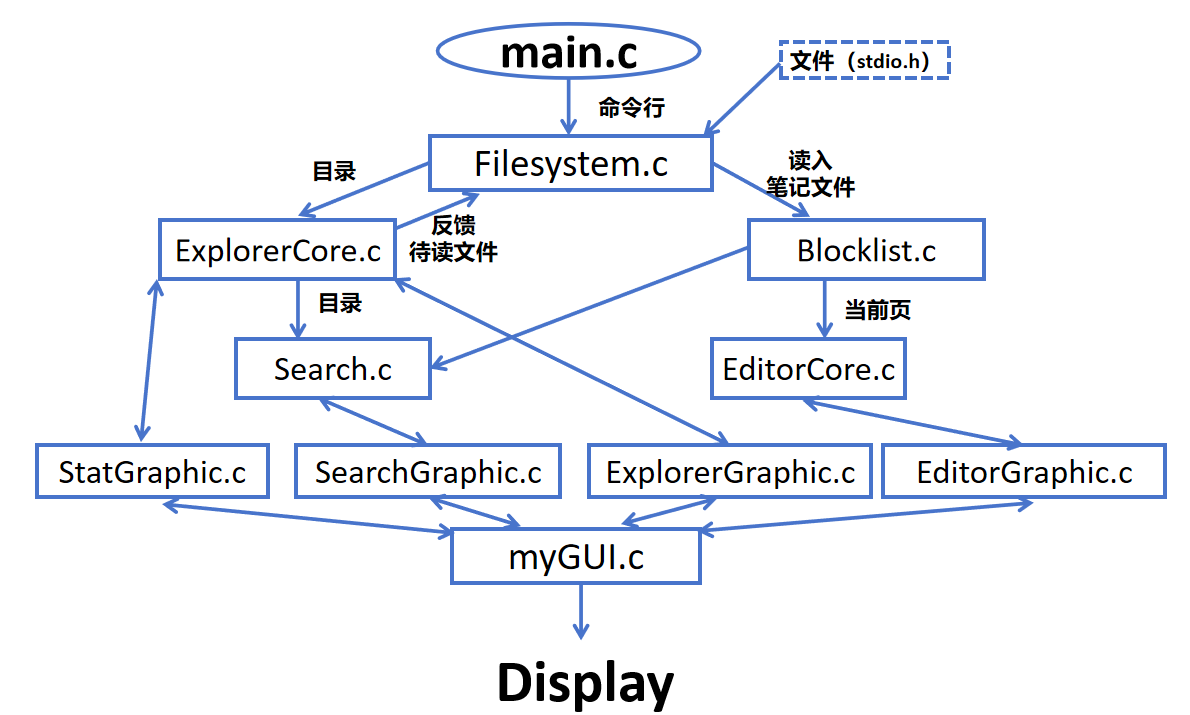
1. **输入**
2. **字符内容：**键盘的输入字符，用于编辑文件。字符内容应该只包含键盘上的可视字符或制表符。
3. **图片：**未经压缩的BMP格式图片。
4. **鼠标操作：**进行光标操作、浏览、按钮操作和使用浏览窗口。鼠标操作应该全程位于GUI窗口内。
5. **输出**
6. **屏幕内容：**使用libgraphics输出显示笔记文件中的某一部分，包含文本块、图片块以及潜在的其他类型块结构。对于文本块，自定义的字符颜色上限为255，字符样式包含正常、粗体、斜体、粗体+斜体四种。对于图片块，应该从未经压缩的BMP格式图片读取。
7. **统计数据：**使用DrawArc功能绘制饼图，统计每个标签或文件夹的数据量、编辑时长等等。
8. **浏览信息：**使用libgraphics输出浏览信息，在页面左侧以树状显示，并且高亮显示正在编辑的文件名称。
9. **存档文件：**输出的存档文件应该包含笔记文件和目录文件两者，其中笔记文件应该完整记录笔记内容，目录文件应该记录统计信息、文件地址、文件名和文件分级分类信息等非笔记信息。

## 性能需求

在对某一个字符串进行搜索时，如果搜索范围是所有已存在的笔记文件，文本串长度可能会是100000级别。此时如果使用逐个匹配的搜索方法，则无法做到即时增量搜索。

# 程序开发设计

## 总体架构设计



main.c接受可能的命令行参数，将文件地址输入FileSystem.c，并且使其读取目录文件。FileSystem.c读取目录文件输入ExplorerCore.c，ExplorerCore.c解析目录文件之后反馈给FileSystem.c使其继续读入笔记文件，转发给Blocklist.c进行读入和分块处理，这些数据一部分用于Search.c初始化SAM结构，另一部分用于EditorCore.c进一步建立BlockChain结构用于文本编辑。这些模块最终通过相应的标准化Graphic方法与myGUI.c进行交互，完成最终的显示与交互。

## 功能模块设计

**myGUI.c：**

该部分代码实现了一个图形用户界面（GUI）系统的基本结构搭建与信息转发。

**一、局部数据设计**

**1.常量定义：**通过宏定义各种界面元素的尺寸和边距，这些常量定义控制了界面布局和样式。

// 宏定义

#define TITLE\_HEIGHT 0 // 标题区高度

#define MENU\_HEIGHT 0.3 // 菜单区高度

#define TOOL\_HEIGHT 0.5 // 工具区高度

#define BUTTON\_WIDTH 1.0 // 按钮宽度

#define BUTTON\_GAP 0.05 // 按钮之间的间隙

#define SCROLLBAR\_WIDTH 0.1 // 滚动条宽度

#define WINDOW\_MARGIN 0 // 窗口边缘的间隙

#define INPUT\_BAR\_HEIGHT 0.3 // 输入栏高度

1. **函数指针**：为多种界面操作和事件处理设置了函数指针，以便使得GUI逻辑与其余部分相互分离，提高模块的封装程度。

// 区域绘制函数指针

static AreaPaintMethod explorerDraw = NULL;

static AreaPaintMethod editorGraphicDraw = NULL;

static GetInfoMethod rollerUpperBoundDraw = NULL;

static GetInfoMethod rollerLowerBoundDraw = NULL;

static SetValueMethod setRollerHeight = NULL;

static SetValueMethod setEditorWidth = NULL;

// 按钮、键盘事件处理函数指针

static ButtonEvent editorMouseMiddleRollup = NULL;

static ButtonEvent editorMouseMiddleRolldown = NULL;

static KeyboardEventMethod editorKeyboard = NULL;

static KeyboardEventMethod editorKeyboardSpecial = NULL;

static ButtonEvent imageButton = NULL;

static ButtonEvent italicButton = NULL;

static ButtonEvent boldButton = NULL;

1. **菜单项相关数据：**定义了菜单项名称及其对应的回调函数，以便处理用户在菜单栏中的选择。

// 菜单项对应的回调函数指针

static ButtonEvent FileMenuMethod[5] = {};

static ButtonEvent ToolsMenuMethod[3] = {};

static ButtonEvent AboutMenuMethod[2] = {};

// 菜单项的名称

static char \*fileMenu[] = {"File", "Save", "Open", "New", "Save As", "Close"};

static char \*toolsMenu[] = {"Tools", "Stats", "Search", "Browse"};

static char \*aboutMenu[] = {"About", "Settings", "Help"};

1. **用户输入相关数据：**用于处理用户输入、确定页面处于何种状态及输入缓冲区的相关数据。

//用户输入相关的变量

static int inputMode = 0; // 输入模式标志位

static ButtonEventWithInput inputConfirmedCallback = NULL;

static char inputBuffer[256]; // 输入栏缓冲区

static int inputBufferLength = 0;

static char majorInputMessage[256]; // 弹窗输入提示消息

static int keyboardConfirmed = 0;

static int keyboardCancelled = 0;

1. **字母显示定义相关数据、指针：**存储最后选择的颜色，目前的字体字号等及其在UI中的定义方法。

// 颜色、字体、字号相关变量及其回调函数指针

static char lastSelectedColor[32] = "Red"; // 存储快捷颜色

static ButtonEventWithInput colorButton = NULL;

static ButtonEventWithInput sizeButton = NULL;

static ButtonEventWithInput fontButton = NULL;

1. **全局数据：**用于保存窗口大小、窗口运行时间的相关数据。

//窗口控制，计时开始时间

static clock\_t startTime;

double screenHeight; // 屏幕高度

double screenWidth; // 屏幕宽度

1. **鼠标处理相关数据**：储存鼠标的处理函数和相应状态

// 鼠标事件相关变量及其处理函数指针

static int mouseFocusID = 0;

static MouseEventMethod mouseDownFuncs[3][5]; // NULL, left, right ; NULL, editor, explorer, roller, search

static ButtonEvent mouseUpFuncs[3][5];

static int mouseActivated[3];

static double mouseActivatedDx;

static double mouseActivatedDy;

1. **键盘处理相关数据**：定义了键盘事件的处理相关状态和方法，其中还包含了判断特殊键状态的宏定义和辅助函数。

// 键盘状态相关常量及变量

#define KEYSTATE\_CONTROL 1

#define KEYSTATE\_SHIFT 2

#define KEYSTATE\_ALT 4

static int UIKeyState;

// 判断Ctrl键状态

static int GetCtrlKeyState() { return UIKeyState | KEYSTATE\_CONTROL; }

// 判断Shift键状态

static int GetShiftKeyState() { return UIKeyState | KEYSTATE\_SHIFT; }

// 判断Alt键状态

static int GetAltKeyState() { return UIKeyState | KEYSTATE\_ALT; }

// 键盘事件相关常量定义

#define KEYBOARD\_BACKSPACE 0x08

#define KEYBOARD\_ENTER 0x0D

#define KEYBOARD\_SHIFT 0x10

#define KEYBOARD\_CTRL 0x11

#define KEYBOARD\_ALT 0x12

#define KEYBOARD\_ESC 0x1B

#define KEYBOARD\_SPACE 0x20

#define KEYBOARD\_LEFT 0x25

#define KEYBOARD\_UP 0x26

#define KEYBOARD\_RIGHT 0x27

#define KEYBOARD\_DOWN 0x28

#define KEYBOARD\_DELETE 0x2E

#define NUMBER\_BEGIN 0x30

#define ALPHA\_BEGIN 0x41

**二、功能实现算法**

**1. 界面绘制**

界面绘制是实现用户界面交互的基础，分为菜单栏、工具栏、主区域和输入区域的绘制。

**1.1 菜单栏绘制：**主要使用 DrawMenuBar\_ 函数。

**算法核心：**

调用 SetPenColor 和 drawRectangle 绘制菜单栏背景。

使用 menuList 函数分别绘制文件、工具和关于菜单项，记录用户选择的菜单项索引。

根据用户选择，调用 FileMenuMethod, ToolsMenuMethod, AboutMenuMethod 中对应的回调函数。

**数据流向：**

用户点击菜单项，选择结果存储到 choice1, choice2, choice3。

对应的回调函数从 FileMenuMethod 等数组中被调用，实现相应的操作。

**1.2 工具栏绘制：**主要使用 DrawToolBar 函数。

**算法核心：**

调用 SetPenColor 和 drawRectangle 绘制工具栏背景。

根据 BUTTON\_GAP 和 BUTTON\_WIDTH 计算每个按钮的位置和尺寸。

调用 button 函数绘制颜色、字体、粗体等按钮，捕获用户的点击事件。

**数据流向：**

用户点击按钮，button 函数返回非零值。

根据返回值，调用如 colorButton, boldButton 等相应的回调函数。

**2. 事件处理**

事件处理模块主要包括鼠标事件和键盘事件处理，确保用户交互的顺畅。

**2.1 鼠标事件处理：**主要使用 myMouseEventProcess 函数。

**算法核心：**

使用 uiGetMouse 将鼠标事件的基本信息传递给simpleGUI库。

调用 mouseDownProcess, mouseUpProcess, mouseMoveProcess 等函数处理鼠标按下、抬起和移动事件。

**数据流向：**

根据鼠标位置和事件类型分发给 mouseDownFuncs, mouseUpFuncs 中的回调函数。使用回调函数数组使程序简洁紧凑，同时保证模块的独立性。

**2.2 键盘事件处理：**主要使用 myKeyboardEventProcess 和 myCharEventProcess 函数。

**算法核心：**

使用 uiGetKeyboard 和 uiGetChar 捕获键盘和字符事件。

根据当前输入模式，分别调用 editorKeyboard, inputConfirmedCallback 等处理用户输入。

**数据流向：**

根据键盘事件与输入模式状态，调用不同的回调函数或更新 inputBuffer 与 inputBufferLength。

更新如 UIKeyState 这样的状态变量管理键盘状态。

**3. 界面切换**

界面切换模块允许程序在不同界面模式之间切换。

**算法核心：**

ChangeDisplayMethodToMain,

ChangeDisplayMethodToInput,

ChangeDisplayMethodToMajorInput,

ChangeDisplayMethodToSearch,

ChangeDisplayMethodToStat,

ChangeDisplayMethodToSetting

上述函数可以根据需要切换界面模式。

**数据流向：**

使用 inputMode 标志变量存储当前界面模式。

根据切换逻辑，调用如 setRollerHeight 更新滚动区域高度，确保界面的正确绘制和交互。

**4. 初始化与启动**

启动函数 GUI\_Start 负责初始化并注册必要的事件处理函数。

**算法核心：**

调用 InitGraphics 初始化图形系统。

设置初始界面宽度和高度，调用 setEditorWidth 确定编辑区域宽度。

使用 registerMouseEvent, registerKeyboardEvent, registerCharEvent 注册事件处理函数。

**数据流向：**

初始化过程中，编辑区域宽度被交给editorCore.c的全局变量，并且向graphics库提供了事件处理函数以进入消息循环。

## 数据结构设计

1. **BlockList数据结构**

保存在BlockList.c中，由BlockList.h声明并共享。保存块的基本信息，同时可以使用type字段提供统一的访问接口，以减少编码工作量。

1. **DictionaryList 数据结构**

由DictionaryList.c实现并提供了通用操作函数，主要由explorerCore.c使用，用于表示复杂的目录层级关系，同时支持筛选与显示操作，用于搜索或显示。

1. **BlockChain数据结构**

由EditorCore.c私有实现，基于BlockList进一步处理的结构，能够在保有排版灵活性的基础上保证纯文本内容的连续性，使得跨块的光标框选操作得以可能。

1. **SAM结构**

预计由Search.c私有实现，建立所有文件内容的SAM（后缀自动机），以此实现快速的增量搜索功能。

1. **ColorTable与FontTable结构**

由FileSystemCore.c实现，保存颜色名、颜色定义和字体名以及其对应的ID，一方面能够在文件读取的过程中提供简便的颜色表示方法，另一方面能够提供稳定的字符串地址以在杜绝可能的野指针的同时不需要为每一个块定义独立的内存空间。

## 函数设计描述

**myGUI.c:（更新于6.5 16:58）**

·函数原型：**void GUI\_Start();**

·功能描述：初始化图形界面，设置初始参数和事件处理函数，并进入主界面。

·参数描述：无。

·返回值描述：无。

·重要局部变量定义：startTime: 记录程序启动的时间。

·重要局部变量用途描述：用于计算程序运行的时间。

·函数算法描述：

1.初始化图形界面和时间。

2.设置初始界面参数。

3.设置菜单和按钮的颜色。

4.注册鼠标、键盘和字符事件的处理函数。

5.切换到主界面。

·与3.2中模块的对应关系：myGUI.c

·原型和定义保存在哪个.c/.h文件中：myGUI.h 和 myGUI.c

·函数原型：**static void changeDisplayMethodProcess();**

·功能描述：在界面更改时处理各种逻辑，包括取消输入和取消鼠标拖动。

·参数描述：无。

·返回值描述：无。

·重要局部变量定义：无。

·重要局部变量用途描述：无。

·函数算法描述：

1.如果有正在输入的内容，调用输入确认回调函数并将其置空。

2.调用 mouseUpProcess 函数取消鼠标左键和右键的拖动状态。

·与3.2中模块的对应关系：myGUI.c

·原型和定义保存在哪个.c/.h文件中：myGUI.h 和 myGUI.c

·函数原型：**static void clearInputBuffers();**

·功能描述：清空输入缓冲区。

·参数描述：无。

·返回值描述：无。

·重要局部变量定义：无。

·重要局部变量用途描述：无。

·函数算法描述：

1.将 inputBufferLength 置为0。

2.将 inputBuffer 的第一个字符置为 \0。

·与3.2中模块的对应关系myGUI.c

·原型和定义保存在哪个.c/.h文件中：myGUI.h 和 myGUI.c

·函数原型：**void ChangeDisplayMethodToMain();**

·功能描述：切换到主界面。

·参数描述：无。

·返回值描述：无。

·重要局部变量定义：无。

·重要局部变量用途描述：无。

·函数算法描述：调用 changeDisplayMethodProcess 函数。

1.设置滚动条高度。

2.设置 inputMode 为 0。

·与3.2中模块的对应关系：myGUI.c

·原型和定义保存在哪个.c/.h文件中：myGUI.h 和 myGUI.

·函数原型：**void ChangeDisplayMethodToInput(ButtonEventWithInput callback);**

·功能描述：将显示模式切换为输入模式，并注册带输入参数的按钮事件处理函数。

·参数描述：callback：带输入参数的按钮事件处理函数。

·返回值描述：无。

·重要局部变量定义：无。

·重要局部变量用途描述：无。

·函数算法描述:

1.调用显示方式处理函数。

2.计算主显示区域的高度，减去标题栏、菜单栏、工具栏、输入栏和窗口边距的高度。

3.设置滚动条高度为计算后的主显示区域高度。

4.设置输入模式标志位为 1。

5.将传入的回调函数赋值给全局变量。

6.清空输入缓冲区。

·与3.2中模块的对应关系：myGUI.c

·原型和定义保存在哪个.c/.h文件中：myGUI.h 和 myGUI.c.

·函数原型：**void ChangeDisplayMethodToMajorInput(char\* inputMessage, ButtonEventWithInput callback);**

·功能描述：切换到弹窗输入界面。

·参数描述：

inputMessage：弹窗输入提示消息。

callback：输入确认后的回调函数。

·返回值描述：无。

·重要局部变量定义：无。

·重要局部变量用途描述：无。

·函数算法描述：

1.调用 changeDisplayMethodProcess 函数。

2.设置滚动条高度。

3.设置 inputMode 为 2。

4.设置 inputConfirmedCallback 为传入的回调函数。

5.清空输入缓冲区。

6.复制 inputMessage 到 majorInputMessage。

·与3.2中模块的对应关系：myGUI.c

·原型和定义保存在哪个.c/.h文件中：myGUI.h 和 myGUI.c

·函数原型：

**void ChangeDisplayMethodToSearch(ButtonEventWithInput callback);**

·功能描述：切换到搜索界面。

·参数描述：callback：输入确认后的回调函数。

·返回值描述：无。

·重要局部变量定义：无。

·重要局部变量用途描述：无。

·函数算法描述：

1.调用 changeDisplayMethodProcess 函数。

2.设置 inputMode 为 3。

3.设置 inputConfirmedCallback 为传入的回调函数。

4.清空输入缓冲区。

·与3.2中模块的对应关系：myGUI.c

·原型和定义保存在哪个.c/.h文件中：myGUI.h 和 myGUI.c

·函数原型：**void ChangeDisplayMethodToStat();**

·功能描述：切换到统计界面。

·参数描述：无。

·返回值描述：无。

·重要局部变量定义：无。

·重要局部变量用途描述：无。

·函数算法描述：

1.调用 changeDisplayMethodProcess 函数。

2.设置 inputMode 为 4。

·与3.2中模块的对应关系：myGUI.c

·原型和定义保存在哪个.c/.h文件中：myGUI.h 和 myGUI.c

·函数原型：**void ChangeDisplayMethodToSetting();**

·功能描述：切换到设置界面。

·参数描述：无。

·返回值描述：无。

·重要局部变量定义：无。

·重要局部变量用途描述：无。

·函数算法描述：

1.调用 changeDisplayMethodProcess 函数。

2.设置 inputMode 为 5。

·与3.2中模块的对应关系：界面切换模块。

·原型和定义保存在哪个.c/.h文件中：myGUI.h 和 myGUI.c

·函数原型：**static void DrawMenuBar\_(double x, double y, double width, double height);**

·功能描述：绘制菜单栏，包括背景和菜单选项。

·参数描述：

x：菜单栏的起始横坐标。

y：菜单栏的起始纵坐标。

width：菜单栏的宽度。

height：菜单栏的高度。

·返回值描述：无。

·重要局部变量定义：无。

·重要局部变量用途描述：无。

·函数算法描述：

1.绘制菜单栏背景（仅边框，不填充）。

2.绘制三个菜单选项，并根据用户选择调用相应的处理函数。

3.如果用户选择了某个菜单选项，切换显示模式并执行对应的菜单操作。

·与3.2中模块的对应关系：myGUI.c

·原型和定义保存在哪个.c/.h文件中:myGUI.h 和 myGUI.c

·函数原型：**void colorInputCallback(const char\* color);**

·功能描述：处理颜色输入回调，更新最后选择的颜色并调用相应的按钮函数。

·参数描述：color：用户输入的颜色字符串。

·返回值描述：无。

·重要局部变量定义：无。

·重要局部变量用途描述：无。

·函数算法描述：

1.检查输入的颜色是否有效。

2.如果颜色有效，更新最后选择的颜色。

3.调用颜色按钮函数。

·与3.2中模块的对应关系：myGUI.c

·原型和定义保存在哪个.c/.h文件中:myGUI.h 和 myGUI.c

·函数原型：**void DrawToolBar(double x, double y, double width, double height);**

·功能描述：绘制工具栏，包括背景和各个工具按钮。

·参数描述：

x：工具栏的起始横坐标。

y：工具栏的起始纵坐标。

width：工具栏的宽度。

height：工具栏的高度。

·返回值描述：无。

·重要局部变量定义：无。

·重要局部变量用途描述：无。

·函数算法描述：

1.绘制工具栏背景（仅边框，不填充）。

2.定义按钮的位置和属性。

3.绘制每个按钮，并根据用户点击调用相应的处理函数。

·与3.2中模块的对应关系：myGUI.c

·原型和定义保存在哪个.c/.h文件中:myGUI.h 和 myGUI.c

·函数原型：**void DrawMainArea(double x, double y, double width, double height);**

·功能描述：绘制主显示区域，包括浏览区、编辑区和滚动条。

·参数描述：

x：主显示区域的起始横坐标。

y：主显示区域的起始纵坐标。

width：主显示区域的宽度。

height：主显示区域的高度。

·返回值描述：无。

·重要局部变量定义：无。

·重要局部变量用途描述：无。

·函数算法描述：

1.计算各部分的位置和尺寸。

2.绘制浏览区、编辑区和滚动条（仅边框，不填充）。

3.根据需要调用相应的绘制函数。

·与3.2中模块的对应关系：myGUI.c

·原型和定义保存在哪个.c/.h文件中：myGUI.h 和 myGUI.c

·函数原型：**void DrawInputBar(double x, double y, double width, double height);**

·功能描述：绘制输入栏，包括背景、标签和按钮。

·参数描述：

x：输入栏的起始横坐标。

y：输入栏的起始纵坐标。

width：输入栏的宽度。

height：输入栏的高度。

·返回值描述：无。

·重要局部变量定义：无。

·重要局部变量用途描述：无。

·函数算法描述：

1.绘制输入栏背景（仅边框，不填充）。

2.绘制输入标签和当前输入内容。

3.绘制确认和退出按钮，并根据用户点击调用相应的处理函数。

·与3.2中模块的对应关系：myGUI.c

·原型和定义保存在哪个.c/.h文件中：myGUI.h 和 myGUI.c

·函数原型：**void DrawSettingPage(double x, double y, double width, double height);**

·功能描述：绘制设置页面，包括颜色和字体的配置输入区域以及确认和退出按钮。

·参数描述：

x：设置页面的起始横坐标。

y：设置页面的起始纵坐标。

width：设置页面的宽度。

height：设置页面的高度。

·返回值描述：无。

·重要局部变量定义：

typedef struct {

char colorName[32];

char red[32];

char green[32];

char blue[32];

char fontName[32];

} Settings;

static Settings settings = {"Black", "0", "0", "0", "Arial"};

·重要局部变量用途描述：

settings：保存颜色和字体的配置，包括颜色名称、RGB值和字体名称。

·函数算法描述：

1.设置画笔颜色为黑色。

2.绘制颜色配置输入区域，包括颜色名称、红色、绿色和蓝色的输入框及其标签。

3.绘制字体配置输入区域，包括字体名称的输入框及其标签。

4.绘制确认和退出按钮，并根据用户点击调用相应的处理函数。

·与3.2中模块的对应关系myGUI.c

·原型和定义保存在哪个.c/.h文件中:myGUI.h 和 myGUI.c

·函数原型：**static void displayBegin();**

·功能描述：设置初始显示界面背景为白色，并设置绘制颜色为黑色。

·参数描述：无。

·返回值描述：无。

·重要局部变量定义：无。

·重要局部变量用途描述：无。

·函数算法描述：

1.设置背景颜色为白色，填充整个屏幕区域。

2.设置绘制颜色为黑色。

·与3.2中模块的对应关系：myGUI.c

·原型和定义保存在哪个.c/.h文件中：myGUI.h 和 myGUI.c

·函数原型：**static void displayMod0();**

·功能描述：绘制主界面，包括菜单栏、工具栏和主显示区域。

·参数描述：无。

·返回值描述：无。

·重要局部变量定义：无。

·重要局部变量用途描述：无。

·函数算法描述：

1.绘制菜单栏。

2.绘制工具栏。

3.计算并绘制主显示区域。

4.如果输入模式不是主界面模式，调用 display 函数。

·与3.2中模块的对应关系：myGUI.c

·原型和定义保存在哪个.c/.h文件中:myGUI.h 和 myGUI.c

·函数原型：static void displayMod1();

·功能描述：绘制输入框界面，包括菜单栏、工具栏、主显示区域和输入栏。

·参数描述：无。

·返回值描述：无。

·重要局部变量定义：无。

·重要局部变量用途描述：无。

·函数算法描述：

1.绘制菜单栏。

2.绘制工具栏。

3.计算并绘制主显示区域。

4.绘制输入栏。

5.如果输入模式不是输入框界面模式，调用 display 函数。

·与3.2中模块的对应关系：输入框界面处理模块。

·原型和定义保存在哪个.c/.h文件中：myGUI.h 和 myGUI.c

·函数原型：**static void displayMod2();**

·功能描述：绘制弹窗界面，包括菜单栏、工具栏、主显示区域和弹窗。

·参数描述：无。

·返回值描述：无。

·重要局部变量定义：无。

·重要局部变量用途描述：无。

·函数算法描述：

1.绘制菜单栏。

2.绘制工具栏。

3.计算并绘制主显示区域。

4.计算并绘制弹窗。

5.绘制弹窗中的输入内容和按钮。

6.如果输入模式不是弹窗界面模式，调用 display 函数。

·与3.2中模块的对应关系：myGUI.c

·原型和定义保存在哪个.c/.h文件中：myGUI.h 和 myGUI.c

·函数原型：**static void displayMod3();**

·功能描述：绘制搜索界面，包括菜单栏、搜索输入栏和搜索结果显示区。

·参数描述：无。

·返回值描述：无。

·重要局部变量定义：无。

·重要局部变量用途描述：无。

·函数算法描述：

1.绘制菜单栏。

2.计算并绘制搜索输入栏。

3.绘制搜索结果显示区。

4.如果输入模式不是搜索界面模式，调用 display 函数。

·与3.2中模块的对应关系:搜索界面处理模块。

·原型和定义保存在哪个.c/.h文件中:myGUI.h 和 myGUI.c

·函数原型：**static void displayMod4();**

·功能描述：绘制统计界面，包括菜单栏和统计面板。

·参数描述：无。

·返回值描述：无。

·重要局部变量定义：无。

·重要局部变量用途描述：无。

·函数算法描述：

1.绘制菜单栏。

2.绘制统计面板。

3.绘制退出按钮。

4.如果输入模式不是统计界面模式，调用 display 函数。

·与3.2中模块的对应关系：myGUI.c

·原型和定义保存在哪个.c/.h文件中：myGUI.h 和 myGUI.c

·函数原型：**static void displayMod5();**

·功能描述：绘制设置界面，包括菜单栏、工具栏和设置页面。

·参数描述：无。

·返回值描述：无。

·重要局部变量定义：无。

·重要局部变量用途描述：无。

·函数算法描述：

1.绘制菜单栏。

2.绘制工具栏。

3.计算并绘制设置页面。

4.如果输入模式不是设置界面模式，调用 display 函数。

·与3.2中模块的对应关系：myGUI.c

·原型和定义保存在哪个.c/.h文件中:myGUI.h 和 myGUI.c

·函数原型：**static void display();**

·功能描述：根据当前输入模式调用相应的显示函数。

·参数描述;无。

·返回值描述；无。

·重要局部变量定义;无。

·重要局部变量用途描述；无。

·函数算法描述;

1.获取当前窗口的宽度和高度。

2.调用 displayBegin 函数设置初始显示背景。

3.根据当前输入模式调用相应的显示函数。

·与3.2中模块的对应关系：myGUI.c

·原型和定义保存在哪个.c/.h文件中:myGUI.h 和 myGUI.c

·函数原型：**static void mouseDownProcess(double mx, double my, int button, double dx, double dy, double cx, double cy, int id);**

·功能描述：处理鼠标按下事件，判断鼠标点击区域并调用相应的事件处理函数。

·参数描述：

mx：鼠标点击的横坐标。

my：鼠标点击的纵坐标。

button：鼠标按键编号（1为左键，2为右键）。

dx：区域左上角的横坐标。

dy：区域左上角的纵坐标。

cx：区域右下角的横坐标。

cy：区域右下角的纵坐标。

id：区域的ID。

·返回值描述：无。

·重要局部变量定义：无。

·重要局部变量用途描述：无。

·函数算法描述：

1.判断鼠标点击是否在指定区域内。

2.如果有焦点ID且有相应的鼠标抬起处理函数，调用该函数。

3.设置当前按键激活状态并调用相应的鼠标按下处理函数。

4.更新焦点ID和激活区域的偏移。

·与3.2中模块的对应关系：myGUI.c

·原型和定义保存在哪个.c/.h文件中：myGUI.h 和 myGUI.c

·函数原型：**static void mouseUpProcess(int button);**

·功能描述：处理鼠标抬起事件，调用相应的事件处理函数。

·参数描述：button：鼠标按键编号（1为左键，2为右键）。

·返回值描述：无。

·重要局部变量定义：无。

·重要局部变量用途描述：无。

·函数算法描述

1.如果当前按键激活状态为真且有相应的鼠标抬起处理函数，调用该函数。

2.重置按键激活状态和焦点ID。

·与3.2中模块的对应关系：myGUI.c

·原型和定义保存在哪个.c/.h文件中：myGUI.h 和 myGUI.c

·函数原型：**static void mouseMoveProcess(double mx, double my);**

·功能描述：处理鼠标移动事件，调用相应的事件处理函数。

·参数描述：

mx：鼠标移动到的横坐标。

my：鼠标移动到的纵坐标。

·返回值描述：无。

·重要局部变量定义：无。

·重要局部变量用途描述：无。

·函数算法描述：

1.如果没有按键激活状态，返回。

2.根据当前激活按键调用相应的鼠标移动处理函数。

·与3.2中模块的对应关系myGUI.c

·原型和定义保存在哪个.c/.h文件中:myGUI.h 和 myGUI.c

·函数原型：**void MouseEventProcess(int x, int y, int button, int event);**

·功能描述：处理鼠标事件，包括按下、抬起、移动和滚动。

·参数描述：

x：鼠标事件发生时的横坐标。

y：鼠标事件发生时的纵坐标。

button：鼠标按键编号。

event：鼠标事件类型（按下、抬起、移动或滚动）。

·返回值描述：无。

·重要局部变量定义：无。

·重要局部变量用途描述：无。

·函数算法描述：

1.清除显示并获取鼠标事件。

2.将屏幕坐标转换为libgraphics坐标。

3.根据事件类型调用相应的处理函数。

4.调用 display 函数更新显示。

·与3.2中模块的对应关系：myGUI.c

·原型和定义保存在哪个.c/.h文件中：myGUI.h 和 myGUI.c

·函数原型：**void CharEventProcess(char ch);**

·功能描述：处理字符输入事件，更新输入缓冲区并调用相应的处理函数。

·参数描述：ch：输入的字符。

·返回值描述：无。

·重要局部变量定义：无。

·重要局部变量用途描述：无。

·函数算法描述：

1.清除显示并获取字符事件。

2.如果没有按下控制键或Alt键，处理字符输入。

3.根据当前输入模式更新输入缓冲区或调用相应的回调函数。

4.调用 display 函数更新显示。

·与3.2中模块的对应关系：myGUI.c

·原型和定义保存在哪个.c/.h文件中:myGUI.h 和 myGUI.c

·函数原型：**void KeyboardEventProcess(int key, int event);**

·功能描述：处理键盘事件，包括按键按下和抬起。

·参数描述：

key：按下或抬起的键盘键值。

event：键盘事件类型（按下或抬起）。

·返回值描述：无。

·重要局部变量定义：无。

·重要局部变量用途描述：无。

·函数算法描述：

1.清除显示并获取键盘事件。

2.更新UI按键状态。

3.如果按下的键是控制键、Alt键或Shift键，直接返回。

4.判断是否有快捷键被触发并调用相应的回调函数。

5.处理特殊字符输入并调用相应的处理函数。

6.用 display 函数更新显示。

·与3.2中模块的对应关系：myGUI.c

·原型和定义保存在哪个.c/.h文件中：myGUI.h 和 myGUI.c

**textStructure.c：（更新于6.5 16:58）**

·函数原型：**StyleString\* ReadStyleString(FILE\* f);**

·功能描述：从文件中读取一个 StyleString 结构体。

·参数描述：f：文件指针，指向要读取的文件。

·返回值描述：返回一个指向 StyleString 结构体的指针。

·重要局部变量定义：bufferInt：用于临时存储从文件中读取的整数数据。

·重要局部变量用途描述：bufferInt：用于存储 StyleString 结构体的元数据（如字体大小、缩进、颜色等）。

·函数算法描述：

1.分配一个 StyleString 结构体的内存。

2.从文件中读取5个整数到 bufferInt 数组中。

3.将 bufferInt 中的数据赋值给 StyleString 结构体的相应字段。

4.根据读取的颜色和字体ID查找对应的颜色和字体名称。

5.分配 StyleString 结构体的 context 字段的内存。

6.从文件中读取 context 数据到 StyleString 结构体中。

·与3.2中模块的对应关系：textStructure.c

·原型和定义保存在哪个.c/.h文件中：textStructure.c

·函数原型:**void WriteStyleString(StyleString\* str, FILE\* f);**

·功能描述:将一个 StyleString 结构体写入文件。

·参数描述:

str：指向要写入的 StyleString 结构体。

f：文件指针，指向要写入的文件。

·返回值描述:无。

·重要局部变量定义:bufferInt：用于临时存储要写入文件的整数数据。

·重要局部变量用途描述:bufferInt：用于存储 StyleString 结构体的元数据（字体大小、缩进、颜色）。

·函数算法描述:

1.将 StyleString 结构体的字段值存储到 bufferInt 数组中。

2.根据颜色和字体名称查找对应的颜色和字体ID。

3.将 bufferInt 数组中的数据写入文件。

4.将 StyleString 结构体的 context 数据写入文件。

·与3.2中模块的对应关系:文件写操作模块。

·原型和定义保存在哪个.c/.h文件中:textStructure.c

**imageStructure.c：（更新于6.5 16:58）**

·函数原型：**int ReadBMP(int count, FILE\* f);**

·功能描述：从文件中读取指定字节数的BMP数据，并将其转换为整数。

·参数描述：

count：要读取的字节数。

f：文件指针，指向要读取的文件。

·返回值描述：返回读取的整数值。

·重要局部变量定义：

temp：用于存储从文件中读取的字节数据。

target：用于存储最终的整数值。

·重要局部变量用途描述：

temp：存储从文件中读取的字节数据。

target：存储最终的整数值，通过将字节数据转换为整数得到。

·函数算法描述：

1.从文件中读取 count 个字节到 temp 数组中。

2.将每个字节的数据按位移转换为整数，并累加到 target 中。

·与3.2中模块的对应关系：imageStructure.c

·原型和定义保存在哪个.c/.h文件中:imageStructure.h 和 imageStructure.c

·函数原型：**ImageInfo\* OpenImage(FILE\* f);**

·功能描述：从文件中读取BMP图像信息，并返回一个 ImageInfo 结构体指针。

·参数描述：f：文件指针，指向要读取的文件。

·返回值描述：返回一个指向 ImageInfo 结构体的指针。

·重要局部变量定义：

tempchar：用于存储文件类型标识符。

biCompression：用于存储BMP压缩方式。

bfOffBits：用于存储从文件开头到bitmap信息的字节偏移量。

ptr：指向 ImageInfo 结构体的指针。

·重要局部变量用途描述：

tempchar：存储文件类型标识符以判断文件是否为BMP格式。

biCompression：存储BMP压缩方式以判断是否支持。

bfOffBits：存储从文件开头到bitmap信息的字节偏移量，用于定位像素数据。

ptr：指向 ImageInfo 结构体的指针，用于存储图像信息。

·函数算法描述：

1.读取文件类型标识符并判断文件是否为BMP格式。

2.获取BMP压缩方式并判断是否支持。

3.获取从文件开头到bitmap信息的字节偏移量。

4.分配 ImageInfo 结构体的内存。

5.获取图像的宽度、高度和每个像素所需的比特位数。

6.根据比特位数判断是否有调色板，并读取调色板信息。

7.读取像素数据并存储到 ImageInfo 结构体中。

·与3.2中模块的对应关系：imageStructure.c

·原型和定义保存在哪个.c/.h文件中：imageStructure.h 和 imageStructure.c

·函数原型：**ImageInfo\* ReadImage(FILE\* f)**;

·功能描述：从文件中读取图像信息，并返回一个 ImageInfo 结构体指针。

·参数描述：f：文件指针，指向要读取的文件。

·返回值描述：返回一个指向 ImageInfo 结构体的指针。

·重要局部变量定义：

temp：用于存储从文件中读取的图像元数据。

ptr：指向 ImageInfo 结构体的指针。

·重要局部变量用途描述：

temp：存储从文件中读取的图像元数据。

ptr：指向 ImageInfo 结构体的指针，用于存储图像信息。

·函数算法描述：

1.分配 ImageInfo 结构体的内存。

2.读取图像的元数据并存储到 temp 数组中。

3.将 temp 数组中的数据赋值给 ImageInfo 结构体的相应字段。

4.读取像素数据并存储到 ImageInfo 结构体中。

·与3.2中模块的对应关系：imageStructure.c

·原型和定义保存在哪个.c/.h文件中：imageStructure.h 和 imageStructure.c

·函数原型：**void WriteImage(ImageInfo\* ptr, FILE\* f);**

·功能描述：将一个 ImageInfo 结构体写入文件。

·参数描述：

ptr：指向要写入的 ImageInfo 结构体。

f：文件指针，指向要写入的文件。

·返回值描述：无。

·重要局部变量定义：temp：用于存储要写入文件的图像元数据。

·重要局部变量用途描述：temp：存储要写入文件的图像元数据。

·函数算法描述：

1.将 ImageInfo 结构体的字段值存储到 temp 数组中。

2.将 temp 数组中的数据写入文件。

3.将 ImageInfo 结构体的像素数据写入文件。

·与3.2中模块的对应关系：imageStructure.c

·原型和定义保存在哪个.c/.h文件中：imageStructure.h 和 imageStructure.c

**blocklist.c：（更新于6.5 16:58）**

函数原型：void RegisterReaderMethod(int type, FileReader func);

* **功能描述**：为特定的块类型注册读取方法。
* **参数描述**：type：要注册读取方法的块类型。
  + func：指向读取方法的函数指针。
* **返回值描述**：无。
* **重要局部变量定义**：无。
* **重要局部变量用途描述**：无。
* **函数算法描述**：将提供的函数指针 func 分配给 readers 数组中对应 type 的索引。
* **与 3.2 中模块的对应关系**：blockList.c
* **原型和定义保存在哪个 .c/.h 文件中**：blockList.h 和 blockList.c

函数原型：void RegisterWriterMethod(int type, FileWriter func);

* **功能描述**：为特定的块类型注册写入方法。
* **参数描述**：type：要注册写入方法的块类型。
  + func：指向写入方法的函数指针。
* **返回值描述**：无。
* **重要局部变量定义**：无。
* **重要局部变量用途描述**：无。
* **函数算法描述**：将提供的函数指针 func 分配给 writers 数组中对应 type 的索引。
* **与 3.2 中模块的对应关系**：blockList.c

函数原型：void SetColumnInfo(int columnNum, double\* columns);

* **功能描述**：设置列信息。
* **参数描述**：columnNum：列的数量。
  + columns：指向列宽数组的指针。
* **返回值描述**：无。
* **重要局部变量定义**：无。
* **重要局部变量用途描述**：无。
* **函数算法描述**：将 columnNum 赋值给 colNum。
  + 使用 memcpy 将 columns 数组中的数据复制到 columnsW 数组。
* **与 3.2 中模块的对应关系**：blockList.c

函数原型：int GetColumnNum(void);

* **功能描述**：获取列的数量。
* **参数描述**：无。
* **返回值描述**：返回列的数量。
* **重要局部变量定义**：无。
* **重要局部变量用途描述**：无。
* **函数算法描述**：返回 colNum 的值。
* **与 3.2 中模块的对应关系**：blockList.c
* **原型和定义保存在哪个 .c/.h 文件中**：blockList.h 和 blockList.c

函数原型：double GetColumnWidth(int cID);

* **功能描述**：获取指定列的宽度。
* **参数描述**：cID：列的编号。
* **返回值描述**：返回指定列的宽度。
* **重要局部变量定义**：无。
* **重要局部变量用途描述**：无。
* **函数算法描述**：返回 columnsW 数组中对应 cID 索引的值。
* **与 3.2 中模块的对应关系**：blockList.c
* **原型和定义保存在哪个 .c/.h 文件中**：blockList.h 和 blockList.c

函数原型：void LoadBlockList(FILE\* f);

* **功能描述**：从文件加载块列表。
* **参数描述**：f：指向要读取的文件的文件指针。
* **返回值描述**：无。
* **重要局部变量定义**：blocknum：用于存储块的数量。
* **重要局部变量用途描述**：blocknum：存储从文件读取的块数量。
* **函数算法描述**：从文件中读取列的数量 colNum。
  + 从文件中读取列宽数组 columnsW。
  + 从文件中读取块的数量 blocknum。
  + 从文件中读取块的基本信息并存储到 blocksBuffer 中。
  + 如果当前页面有效，遍历读取的块，将其信息存储到 blocklist 中并更新 blocklistLength。
  + 遍历读取的块，通过对应类型的读取方法读取块的具体数据，并将其存储到 blocklist 中。
* **与 3.2 中模块的对应关系**：blockList.c
* **原型和定义保存在哪个 .c/.h 文件中**：blockList.h 和 blockList.c

函数原型：void SaveBlockList(FILE\* f);

* **功能描述**：将块列表保存到文件。
* **参数描述**：f：指向要写入的文件的文件指针。
* **返回值描述**：无。
* **重要局部变量定义**：blocknum：用于存储当前页面的块数量。
  + realBlkNum：用于存储实际需要写入的块数量。
* **重要局部变量用途描述**：blocknum：存储当前页面的块数量。
  + realBlkNum：存储实际需要写入的块数量。
* **函数算法描述**：将 colNum 和 columnsW 写入文件。
  + 计算并写入实际需要写入的块数量 realBlkNum。
  + 将 blocksBuffer 中的实际块信息写入文件。
  + 遍历当前页面的块，通过对应类型的写入方法将块的具体数据写入文件。
* **与 3.2 中模块的对应关系**：blockList.c
* **原型和定义保存在哪个 .c/.h 文件中**：blockList.h 和 blockList.c

函数原型：void ChangePageOfBlockList(int p);

* **功能描述**：更改块列表的当前页面。
* **参数描述**：p：新页面的编号。
* **返回值描述**：无。
* **重要局部变量定义**：无。
* **重要局部变量用途描述**：无。
* **函数算法描述**：将 curPage 设置为 p。
* **与 3.2 中模块的对应关系**：blockList.c
* **原型和定义保存在哪个 .c/.h 文件中**：blockList.h 和 blockList.c

函数原型：int GetPageOfBlockList(void);

* **功能描述**：获取块列表的当前页面。
* **参数描述**：无。
* **返回值描述**：返回当前页面的编号。
* **重要局部变量定义**：无。
* **重要局部变量用途描述**：无。
* **函数算法描述**：返回 curPage 的值。
* **与 3.2 中模块的对应关系**：blockList.c
* **原型和定义保存在哪个 .c/.h 文件中**：blockList.h 和 blockList.c

函数原型：Block BlockCreate(int type, void dataptr);

* **功能描述**：创建一个新的块。
* **参数描述**：type：块的类型。
  + dataptr：指向块数据的指针。
* **返回值描述**：返回指向新创建的块的指针。如果当前页面无效，返回 NULL。
* **重要局部变量定义**：curid：用于存储当前块的 ID。
* **重要局部变量用途描述**：curid：存储新创建块的 ID。
* **函数算法描述**：确保当前页面有效。
  + 确保块列表有足够的空间。
  + 创建新的块，并设置其 ID、类型和数据指针。
  + 更新 blocklistLength 并返回新创建的块的指针。
* **与 3.2 中模块的对应关系**：blockList.c
* **原型和定义保存在哪个 .c/.h 文件中**：blockList.h 和 blockList.c

函数原型：void BlockDelete(Block\* blk);

* **功能描述**：删除指定的块。
* **参数描述**：blk：指向要删除的块的指针。
* **返回值描述**：无。
* **重要局部变量定义**：blockNum：当前页面的块数量。
* **重要局部变量用途描述**：blockNum：获取当前页面的块数量。
* **函数算法描述**：将要删除的块标记为已删除。
  + 更新当前页面中的块编号。
* **与 3.2 中模块的对应关系**：blockList.c
* **原型和定义保存在哪个 .c/.h 文件中**：blockList.h 和 blockList.c

函数原型：void ClearBlockList(void);

* **功能描述**：清空块列表。
* **参数描述**：无。
* **返回值描述**：无。
* **重要局部变量定义**：无。
* **重要局部变量用途描述**：无。
* **函数算法描述**：释放当前页面的块列表内存。
  + 释放当前页面的块删除标记数组内存。
  + 释放当前页面的块删除数量数组内存。
  + 将当前页面的块列表、块数量和空间大小重置为 0。
* **与 3.2 中模块的对应关系**：blockList.c
* **原型和定义保存在哪个 .c/.h 文件中**：blockList.h 和 blockList.c

函数原型：void BlockMove(int ID, AlignmentInfo align);

* **功能描述**：移动指定块的对齐方式。
* **参数描述**：ID：要移动的块的 ID。
  + align：新的对齐方式。
* **返回值描述**：无。
* **重要局部变量定义**：无。
* **重要局部变量用途描述**：无。
* **函数算法描述**：确保当前页面有效。
  + 断言要对齐的块的编号小于目标块的编号。
  + 更新指定块的对齐方式。
* **与 3.2 中模块的对应关系**：blockList.c
* **原型和定义保存在哪个 .c/.h 文件中**：blockList.h 和 blockList.c

函数原型：void TraverseBlockList(BlockListTraverseFunc func);

* **功能描述**：遍历块列表并对每个块执行指定操作。
* **参数描述**：func：要执行的操作的函数指针。
* **返回值描述**：无。
* **重要局部变量定义**：blocknum：当前页面的块数量。
* **重要局部变量用途描述**：blocknum：获取当前页面的块数量。
* **函数算法描述**：确保当前页面有效。
  + 遍历当前页面的所有块，对每个块执行指定操作。
* **与 3.2 中模块的对应关系**：blockList.c
* **原型和定义保存在哪个 .c/.h 文件中**：blockList.h 和 blockList.c

函数原型：void AccumlateBlockList(BlockListAccumlateFunc func, void beginValue);

* **功能描述**：遍历块列表并根据指定操作累加结果。
* **参数描述**：func：用于累加操作的函数指针。
  + beginValue：累加的起始值。
* **返回值描述**：返回累加结果。
* **重要局部变量定义**：blocknum：当前页面的块数量。
* **重要局部变量用途描述**：blocknum：获取当前页面的块数量。
* **函数算法描述**：确保当前页面有效。
  + 遍历当前页面的所有块，对每个块执行指定操作并累加结果。
  + 返回累加结果。
* **与 3.2 中模块的对应关系**：blockList.c
* **原型和定义保存在哪个 .c/.h 文件中**：blockList.h 和 blockList.c

函数原型：Block\* GetBlock(int blockID);

* **功能描述**：获取指定 ID 的块。
* **参数描述**：blockID：要获取的块的 ID。
* **返回值描述**：返回指定 ID 的块的指针。
* **重要局部变量定义**：无。
* **重要局部变量用途描述**：无。
* **函数算法描述**：返回指定 ID 的块的指针。
* **与 3.2 中模块的对应关系**：blockList.c
* **原型和定义保存在哪个 .c/.h 文件中**：blockList.h 和 blockList.c

函数原型：void RegisterGetHeightFunc(int type, GetHeightFunc func);

* **功能描述**：为指定块类型注册获取高度的函数。
* **参数描述**：type：要注册高度函数的块类型。
  + func：指向高度函数的函数指针。
* **返回值描述**：无。
* **重要局部变量定义**：无。
* **重要局部变量用途描述**：无。
* **函数算法描述**：无。
* **与 3.2 中模块的对应关系**：blockList.c
* **原型和定义保存在哪个 .c/.h 文件中**：blockList.h 和 blockList.c

**dictionarylist.c：（更新于6.5 16:58）**

static const char\* generateLimitedOutput(drawCommand dc, double width);

* **功能描述**: 生成在给定宽度限制内的字符串输出。
* **参数描述**:dc: 绘制命令，包含绘制所需的信息。
  + width: 字符串的最大宽度。
* **返回值描述**: 返回生成的受限输出字符串。
* **重要局部变量定义**:offset: 前缀后的字符串起始位置。
  + i: 用于遍历字符串的索引。
* **重要局部变量用途描述**:offset: 确定实际内容开始的位置。
  + i: 遍历并检查字符串的宽度是否超出限制。
* **函数算法描述**:根据缩进和前缀生成初始字符串。
  + 如果名称为空，检查字符串宽度是否超出限制，返回结果。
  + 否则，逐个字符添加名称到字符串中，确保总宽度不超过限制。
  + 根据最终宽度调整字符串结尾并返回结果。
* **与 3.2 中模块的对应关系**: graphics.c
* **原型和定义保存在哪个 .c/.h 文件中**: graphics.h 和 graphics.c

static void fillDrawCommandsItem(DictionaryGraphicDatas\* dl, DictionaryItem\* x, int curindent);

* **功能描述**: 填充绘制命令列表中的项目。
* **参数描述**:dl: 字典图形数据。
  + x: 当前字典项目。
  + curindent: 当前缩进级别。
* **返回值描述**: 无。
* **重要局部变量定义**: 无。
* **重要局部变量用途描述**: 无。
* **函数算法描述**:如果项目为空，返回。
  + 增加当前行数，设置绘制命令的相关属性。
  + 递归调用处理下一个项目。
* **与 3.2 中模块的对应关系**: dictionaryList.c
* **原型和定义保存在哪个 .c/.h 文件中**: dictionaryList.h 和 dictionaryList.c

static void fillDrawCommandsFolder(DictionaryGraphicDatas\* dl, DictionaryFolder\* x, int curindent);

* **功能描述**: 填充绘制命令列表中的文件夹。
* **参数描述**:dl: 字典图形数据。
  + x: 当前字典文件夹。
  + curindent: 当前缩进级别。
* **返回值描述**: 无。
* **重要局部变量定义**: 无。
* **重要局部变量用途描述**: 无。
* **函数算法描述**:如果文件夹为空，返回。
  + 增加当前行数，设置绘制命令的相关属性。
  + 如果文件夹展开，递归处理文件夹中的项目和子文件夹。
  + 递归调用处理下一个文件夹。
* **与 3.2 中模块的对应关系**: dictionaryList.c
* **原型和定义保存在哪个 .c/.h 文件中**: dictionaryList.h 和 dictionaryList.c

static void fillDrawCommands(DictionaryGraphicDatas\* dl, DictionaryFolder\* x);

* **功能描述**: 填充绘制命令列表中的文件夹和项目。
* **参数描述**:dl: 字典图形数据。
  + x: 根字典文件夹。
* **返回值描述**: 无。
* **重要局部变量定义**: 无。
* **重要局部变量用途描述**: 无。
* **函数算法描述**:确保字典图形数据有效。
  + 初始化当前行数为0。
  + 调用递归函数填充绘制命令。
* **与 3.2 中模块的对应关系**: dictionaryList.c
* **原型和定义保存在哪个 .c/.h 文件中**: dictionaryList.h 和 dictionaryList.c

static void checkInitialized();

* **功能描述**: 检查并初始化默认字典图形数据。
* **参数描述**: 无。
* **返回值描述**: 无。
* **重要局部变量定义**: 无。
* **重要局部变量用途描述**: 无。
* **函数算法描述**:检查默认字典图形数据是否已初始化。
  + 如果未初始化，设置默认值。
* **与 3.2 中模块的对应关系**: dictionaryList.c
* **原型和定义保存在哪个 .c/.h 文件中**: dictionaryList.h 和 dictionaryList.c

static void bufferChangeGraphicState(DictionaryGraphicDatas dl);

* **功能描述**: 缓存当前图形状态并应用新的图形设置。
* **参数描述**:dl: 字典图形数据。
* **返回值描述**: 无。
* **重要局部变量定义**:penColor: 当前笔颜色。
  + pointSizeBuffer: 当前字体大小。
  + fontBuffer: 当前字体。
* **重要局部变量用途描述**:penColor: 存储当前笔颜色。
  + pointSizeBuffer: 存储当前字体大小。
  + fontBuffer: 存储当前字体。
* **函数算法描述**:缓存当前笔颜色。
  + 根据字典图形数据设置新的字体大小和字体。
* **与 3.2 中模块的对应关系**: graphics.c
* **原型和定义保存在哪个 .c/.h 文件中**: graphics.h 和 graphics.c

static void retGraphicState(DictionaryGraphicDatas dl);

* **功能描述**: 恢复之前缓存的图形状态。
* **参数描述**:dl: 字典图形数据。
* **返回值描述**: 无。
* **重要局部变量定义**: 无。
* **重要局部变量用途描述**: 无。
* **函数算法描述**:恢复之前缓存的笔颜色、字体大小和字体。
* **与 3.2 中模块的对应关系**: graphics.c
* **原型和定义保存在哪个 .c/.h 文件中**: graphics.h 和 graphics.c

double GetDictionaryListHeight(DictionaryGraphicDatas\* dl, DictionaryFolder\* fl, double width);

* **功能描述**: 获取字典列表的高度。
* **参数描述**:dl: 字典图形数据。
  + fl: 根字典文件夹。
  + width: 可用宽度。
* **返回值描述**: 返回字典列表的总高度。
* **重要局部变量定义**: 无。
* **重要局部变量用途描述**: 无。
* **函数算法描述**:检查并初始化默认字典图形数据。
  + 缓存当前图形状态并应用新的图形设置。
  + 填充绘制命令。
  + 计算并返回字典列表的高度。
  + 恢复图形状态。
* **与 3.2 中模块的对应关系**: dictionaryList.c
* **原型和定义保存在哪个 .c/.h 文件中**: dictionaryList.h 和 dictionaryList.c

void DrawDictionaryList(DictionaryGraphicDatas\* dl, DictionaryFolder\* fl, double cx, double cy, double width, double begH, double endH);

* **功能描述**: 绘制字典列表。
* **参数描述**:dl: 字典图形数据。
  + fl: 根字典文件夹。
  + cx: 绘制起始位置的 x 坐标。
  + cy: 绘制起始位置的 y 坐标。
  + width: 可用宽度。
  + begH: 起始高度。
  + endH: 结束高度。
* **返回值描述**: 无。
* **重要局部变量定义**:cyLimL: 当前 y 坐标的限制。
  + fHeight: 字体高度。
* **重要局部变量用途描述**:cyLimL: 计算 y 坐标限制。
  + fHeight: 获取字体高度。
* **函数算法描述**:检查并初始化默认字典图形数据。
  + 缓存当前图形状态并应用新的图形设置。
  + 计算当前 y 坐标的限制。
  + 填充绘制命令。
  + 遍历绘制命令列表，逐行绘制字典项。
  + 恢复图形状态。
* **与 3.2 中模块的对应关系**: dictionaryList.c
* **原型和定义保存在哪个 .c/.h 文件中**: dictionaryList.h 和 dictionaryList.c

DictionaryCursor PositionizeDictionaryList(DictionaryGraphicDatas\* dl, DictionaryFolder\* fl, double width, double mX, double mY);

* **功能描述**: 确定字典列表中鼠标点击的位置。
* **参数描述**:dl: 字典图形数据。
  + fl: 根字典文件夹。
  + width: 可用宽度。
  + mX: 鼠标 x 坐标。
  + mY: 鼠标 y 坐标。
* **返回值描述**: 返回表示鼠标点击位置的光标。
* **重要局部变量定义**:resLine: 结果行号。
* **重要局部变量用途描述**:resLine: 根据 y 坐标计算的行号。
* **函数算法描述**:检查并初始化默认字典图形数据。
  + 缓存当前图形状态并应用新的图形设置。
  + 填充绘制命令。
  + 根据鼠标 y 坐标计算结果行号。
  + 根据结果行号确定光标位置并返回。
* **与 3.2 中模块的对应关系**: dictionaryList.c
* **原型和定义保存在哪个 .c/.h 文件中**: dictionaryList.h 和 dictionaryList.c

DictionaryGraphicDatas\* CopyDictionaryGraphicDatas(DictionaryGraphicDatas\* source);

* **功能描述**: 拷贝字典图形数据。
* **参数描述**:source: 源字典图形数据。
* **返回值描述**: 返回新的字典图形数据副本。
* **重要局部变量定义**: 无。
* **重要局部变量用途描述**: 无。
* **函数算法描述**:分配新的字典图形数据结构。
  + 拷贝源字典图形数据到新结构。
  + 返回新的字典图形数据。
* **与 3.2 中模块的对应关系**: dictionaryList.c
* **原型和定义保存在哪个 .c/.h 文件中**: dictionaryList.h 和 dictionaryList.c

DictionaryGraphicDatas\* CreateDictionaryGraphicDatas();

* **功能描述**: 创建新的字典图形数据。
* **参数描述**: 无。
* **返回值描述**: 返回新的字典图形数据。
* **重要局部变量定义**: 无。
* **重要局部变量用途描述**: 无。
* **函数算法描述**:检查并初始化默认字典图形数据。
  + 拷贝默认字典图形数据并返回。
* **与 3.2 中模块的对应关系**: dictionaryList.c
* **原型和定义保存在哪个 .c/.h 文件中**: dictionaryList.h 和 dictionaryList.c

void FreeDictionaryGraphicDatas(DictionaryGraphicDatas\* dg);

* **功能描述**: 释放字典图形数据。
* **参数描述**:dg: 要释放的字典图形数据。
* **返回值描述**: 无。
* **重要局部变量定义**: 无。
* **重要局部变量用途描述**: 无。
* **函数算法描述**:释放字典图形数据。
* **与 3.2 中模块的对应关系**: dictionaryList.c
* **原型和定义保存在哪个 .c/.h 文件中**: dictionaryList.h 和 dictionaryList.c

DictionaryItem\* CreateDictionaryItem();

* **功能描述**: 创建新的字典项。
* **参数描述**: 无。
* **返回值描述**: 返回新的字典项。
* **重要局部变量定义**:IDs: 静态变量，用于生成唯一的项ID。
* **重要局部变量用途描述**:IDs: 生成唯一的项ID。
* **函数算法描述**:生成新的唯一项ID。
  + 分配并初始化字典项结构。
  + 设置默认项名称为 "Noname"。
  + 返回新的字典项。
* **与 3.2 中模块的对应关系**: dictionaryList.c
* **原型和定义保存在哪个 .c/.h 文件中**: dictionaryList.h 和 dictionaryList.c

void FreeDictionaryItem(DictionaryItem\* di);

* **功能描述**: 释放字典项。
* **参数描述**:di: 要释放的字典项。
* **返回值描述**: 无。
* **重要局部变量定义**: 无。
* **重要局部变量用途描述**: 无。
* **函数算法描述**:如果字典项在其所属文件夹中，更新文件夹的项列表。
  + 释放字典项的名称和结构。
* **与 3.2 中模块的对应关系**: dictionaryList.c
* **原型和定义保存在哪个 .c/.h 文件中**: dictionaryList.h 和 dictionaryList.c

DictionaryItem\* CopyDictionaryItem(DictionaryItem\* source);

* **功能描述**: 拷贝字典项。
* **参数描述**:source: 源字典项。
* **返回值描述**: 返回新的字典项副本。
* **重要局部变量定义**: 无。
* **重要局部变量用途描述**: 无。
* **函数算法描述**:创建新的字典项。
  + 拷贝源字典项的属性。
  + 返回新的字典项。
* **与 3.2 中模块的对应关系**: dictionaryList.c
* **原型和定义保存在哪个 .c/.h 文件中**: dictionaryList.h 和 dictionaryList.c

DictionaryFolder\* CreateDictionaryFolder();

* **功能描述**: 创建新的字典文件夹。
* **参数描述**: 无。
* **返回值描述**: 返回新的字典文件夹。
* **重要局部变量定义**:IDs: 静态变量，用于生成唯一的文件夹ID。
* **重要局部变量用途描述**:IDs: 生成唯一的文件夹ID。
* **函数算法描述**:生成新的唯一文件夹ID。
  + 分配并初始化字典文件夹结构。
  + 设置默认文件夹名称为 "Noname"。
  + 返回新的字典文件夹。
* **与 3.2 中模块的对应关系**: dictionaryList.c
* **原型和定义保存在哪个 .c/.h 文件中**: dictionaryList.h 和 dictionaryList.c

void FreeDictionaryFolder(DictionaryFolder\* df);

* **功能描述**: 释放字典文件夹。
* **参数描述**:df: 要释放的字典文件夹。
* **返回值描述**: 无。
* **重要局部变量定义**: 无。
* **重要局部变量用途描述**: 无。
* **函数算法描述**:如果字典文件夹在其父文件夹中，更新父文件夹的子文件夹列表。
  + 递归释放子文件夹和项目。
  + 释放字典文件夹的名称和结构。
* **与 3.2 中模块的对应关系**: dictionaryList.c
* **原型和定义保存在哪个 .c/.h 文件中**: dictionaryList.h 和 dictionaryList.c

DictionaryFolder\* CopyDictionaryFolder(DictionaryFolder\* source);

* **功能描述**: 拷贝字典文件夹。
* **参数描述**:source: 源字典文件夹。
* **返回值描述**: 返回新的字典文件夹副本。
* **重要局部变量定义**: 无。
* **重要局部变量用途描述**: 无。
* **函数算法描述**:创建新的字典文件夹。
  + 拷贝源文件夹的属性。
  + 递归拷贝子文件夹和项目。
  + 返回新的字典文件夹。
* **与 3.2 中模块的对应关系**: dictionaryList.c
* **原型和定义保存在哪个 .c/.h 文件中**: dictionaryList.h 和 dictionaryList.c

**filesystem.c：（更新于6.5 16:58）**

void ChangePageAtFileSystem(int page);

* **功能描述**: 改变当前文件系统所在的页面。
* **参数描述**:page: 要切换到的页面。
* **返回值描述**: 无。
* **重要局部变量定义**: curPage: 当前页面。
* **重要局部变量用途描述**: 用于记录当前页面的编号。
* **函数算法描述**:设置全局变量 curPage 为指定的页面编号。
* **与 3.2 中模块的对应关系**: fileSystem.c
* **原型和定义保存在哪个 .c/.h 文件中**: fileSystem.h 和 fileSystem.c

void ClearAllItemsOnPage(int page);

* **功能描述**: 清除指定页面上的所有项目。
* **参数描述**:page: 要清除项目的页面编号。
* **返回值描述**: 无。
* **重要局部变量定义**: 无。
* **重要局部变量用途描述**: 无。
* **函数算法描述**:更改颜色表、字体表和块列表到指定页面并清除。
* **与 3.2 中模块的对应关系**: fileSystem.c
* **原型和定义保存在哪个 .c/.h 文件中**: fileSystem.h 和 fileSystem.c

int LoadFileAtPage(int page, const char\* fileName);

* **功能描述**: 在指定页面加载文件。
* **参数描述**:page: 要加载文件的页面编号。
  + fileName: 要加载的文件名称。
* **返回值描述**: 成功返回0，失败返回1。
* **重要局部变量定义**: FILE\* f: 文件指针。
* **重要局部变量用途描述**: 用于文件操作。
* **函数算法描述**:清除指定页面上的所有项目。
  + 尝试打开指定文件，如果打开失败返回1。
  + 读取颜色表、字体表和块列表。
  + 关闭文件，返回0表示成功。
* **与 3.2 中模块的对应关系**: fileSystem.c
* **原型和定义保存在哪个 .c/.h 文件中**: fileSystem.h 和 fileSystem.c

void SaveFileAtPage(int page, const char\* fileName);

* **功能描述**: 在指定页面保存文件。
* **参数描述**:page: 要保存文件的页面编号。
  + fileName: 要保存的文件名称。
* **返回值描述**: 无。
* **重要局部变量定义**: FILE\* f: 文件指针。
* **重要局部变量用途描述**: 用于文件操作。
* **函数算法描述**:打开指定文件进行写操作。
  + 切换到指定页面并写入颜色表、字体表和块列表。
  + 关闭文件。
* **与 3.2 中模块的对应关系**: fileSystem.c
* **原型和定义保存在哪个 .c/.h 文件中**: fileSystem.h 和 fileSystem.c

void LoadFile(const char\* fileName);

* **功能描述**: 在当前页面加载文件。
* **参数描述**:fileName: 要加载的文件名称。
* **返回值描述**: 无。
* **重要局部变量定义**: 无。
* **重要局部变量用途描述**: 无。
* **函数算法描述**:调用 LoadFileAtPage 函数在当前页面加载文件。
* **与 3.2 中模块的对应关系**: fileSystem.c
* **原型和定义保存在哪个 .c/.h 文件中**: fileSystem.h 和 fileSystem.c

void SaveFile(const char\* fileName);

* **功能描述**: 在当前页面保存文件。
* **参数描述**:fileName: 要保存的文件名称。
* **返回值描述**: 无。
* **重要局部变量定义**: 无。
* **重要局部变量用途描述**: 无。
* **函数算法描述**:调用 SaveFileAtPage 函数在当前页面保存文件。
* **与 3.2 中模块的对应关系**: fileSystem.c
* **原型和定义保存在哪个 .c/.h 文件中**: fileSystem.h 和 fileSystem.c

void RegisterSavFilePath(const char\* savFilePath);

* **功能描述**: 注册保存文件的路径。
* **参数描述**:savFilePath: 保存文件的路径。
* **返回值描述**: 无。
* **重要局部变量定义**: savFilePathstore[255]: 保存文件路径的存储。
* **重要局部变量用途描述**: 存储保存文件的路径。
* **函数算法描述**:使用 memcpy 函数将提供的路径存储到 savFilePathstore。
* **与 3.2 中模块的对应关系**: fileSystem.c
* **原型和定义保存在哪个 .c/.h 文件中**: fileSystem.h 和 fileSystem.c

void LoadSavFile();

* **功能描述**: 加载保存文件。
* **参数描述**: 无。
* **返回值描述**: 无。
* **重要局部变量定义**: FILE\* f: 文件指针。
* **重要局部变量用途描述**: 用于文件操作。
* **函数算法描述**:打开保存文件路径中指定的文件进行读取。
  + 调用 ReadSavFile 函数读取文件内容。
  + 关闭文件。
* **与 3.2 中模块的对应关系**: fileSystem.c
* **原型和定义保存在哪个 .c/.h 文件中**: fileSystem.h 和 fileSystem.c

void StoreSavFile();

* **功能描述**: 保存当前状态到保存文件。
* **参数描述**: 无。
* **返回值描述**: 无。
* **重要局部变量定义**: FILE\* f: 文件指针。
* **重要局部变量用途描述**: 用于文件操作。
* **函数算法描述**:打开保存文件路径中指定的文件进行写入。
  + 调用 WriteSavFile 函数写入当前状态到文件。
  + 关闭文件。
* **与 3.2 中模块的对应关系**: fileSystem.c
* **原型和定义保存在哪个 .c/.h 文件中**: fileSystem.h 和 fileSystem.c

**filesystemCore.c：（更新于6.5 16:58）**

void ChangePageOfColorTable(int p);

* **功能描述**: 更改当前的颜色表页面，并在必要时创建新的页面。
* **参数描述**:p: 要切换到的页面编号。
* **返回值描述**: 无。
* **重要局部变量定义**: curColorPage, colorPage, colorPageExsists。
* **重要局部变量用途描述**: curColorPage 记录当前的颜色表页面。colorPage 存储颜色表的页面。colorPageExsists 标记页面是否存在。
* **函数算法描述**:检查页面是否存在，如果存在则切换到该页面。
  + 如果页面不存在，则创建新页面，并初始化颜色表。
* **与 3.2 中模块的对应关系**: colorTable.c
* **原型和定义保存在哪个 .c/.h 文件中**: colorTable.h 和 colorTable.c

void ClearColorTable();

* **功能描述**: 清除当前页面的颜色表。
* **参数描述**: 无。
* **返回值描述**: 无。
* **重要局部变量定义**: colorPage, curColorPage。
* **重要局部变量用途描述**: colorPage 存储颜色表的页面。curColorPage 记录当前的颜色表页面。
* **函数算法描述**:释放当前页面的颜色表，并重新分配内存进行初始化。
* **与 3.2 中模块的对应关系**: colorTable.c
* **原型和定义保存在哪个 .c/.h 文件中**: colorTable.h 和 colorTable.c

void ReadColorTable(FILE\* f);

* **功能描述**: 从文件中读取颜色表数据。
* **参数描述**:f: 文件指针。
* **返回值描述**: 无。
* **重要局部变量定义**: tempname, tempcode。
* **重要局部变量用途描述**: tempname 存储临时颜色名称。tempcode 存储临时颜色代码。
* **函数算法描述**:从文件中读取颜色名称和代码，并注册到颜色表中，直到遇到结束标志。
* **与 3.2 中模块的对应关系**: colorTable.c
* **原型和定义保存在哪个 .c/.h 文件中**: colorTable.h 和 colorTable.c

void WriteColorTable(FILE\* f);

* **功能描述**: 将当前页面的颜色表写入文件。
* **参数描述**:f: 文件指针。
* **返回值描述**: 无。
* **重要局部变量定义**: p, firstl, secondl。
* **重要局部变量用途描述**: p 指向当前页面的颜色表。firstl 和 secondl 分别记录第一部分和第二部分的长度。
* **函数算法描述**:将颜色表的第一部分和第二部分分别写入文件，并写入结束标志。
* **与 3.2 中模块的对应关系**: colorTable.c
* **原型和定义保存在哪个 .c/.h 文件中**: colorTable.h 和 colorTable.c

int RegisterColorTable(const char\* colorName);

* **功能描述**: 注册颜色表中的颜色名称。
* **参数描述**:colorName: 要注册的颜色名称。
* **返回值描述**: 返回颜色名称的ID。
* **重要局部变量定义**: p, l。
* **重要局部变量用途描述**: p 指向当前页面的颜色表。l 记录颜色表的长度。
* **函数算法描述**:检查颜色名称是否已经存在，如果存在则返回ID。
  + 如果不存在，则注册新的颜色名称并返回ID。
* **与 3.2 中模块的对应关系**: colorTable.c
* **原型和定义保存在哪个 .c/.h 文件中**: colorTable.h 和 colorTable.c

int RegisterColorName(const char\* colorName, double r, double g, double b);

* **功能描述**: 注册颜色名称和对应的颜色值。
* **参数描述**:colorName: 要注册的颜色名称。
  + r, g, b: 颜色的RGB值。
* **返回值描述**: 成功返回0，失败返回1。
* **重要局部变量定义**: p, l。
* **重要局部变量用途描述**: p 指向当前页面的颜色表。l 记录颜色表的长度。
* **函数算法描述**:检查颜色名称是否已经存在，如果存在则返回1。
  + 如果不存在，则注册新的颜色名称和颜色值。
* **与 3.2 中模块的对应关系**: colorTable.c
* **原型和定义保存在哪个 .c/.h 文件中**: colorTable.h 和 colorTable.c

const char\* LookupIDInColorTable(int id);

* **功能描述**: 根据ID查找颜色名称。
* **参数描述**:id: 颜色名称的ID。
* **返回值描述**: 返回颜色名称。
* **重要局部变量定义**: p。
* **重要局部变量用途描述**: p 指向当前页面的颜色表。
* **函数算法描述**:根据ID查找并返回对应的颜色名称。
* **与 3.2 中模块的对应关系**: colorTable.c
* **原型和定义保存在哪个 .c/.h 文件中**: colorTable.h 和 colorTable.c

int LookupColorNameInColorTable(const char\* colorName);

* **功能描述**: 根据颜色名称查找ID。
* **参数描述**:colorName: 颜色名称。
* **返回值描述**: 返回颜色名称的ID。
* **重要局部变量定义**: p, l。
* **重要局部变量用途描述**: p 指向当前页面的颜色表。l 记录颜色表的长度。
* **函数算法描述**:查找颜色名称并返回其ID，如果不存在则返回0。
* **与 3.2 中模块的对应关系**: colorTable.c
* **原型和定义保存在哪个 .c/.h 文件中**: colorTable.h 和 colorTable.c

void TraverseColorDefinitions(ColorDefinitionTraverseFunction func);

* **功能描述**: 遍历当前页面的颜色定义并执行指定的操作。
* **参数描述**:func: 用于处理颜色定义的函数指针。
* **返回值描述**: 无。
* **重要局部变量定义**: p。
* **重要局部变量用途描述**: p 指向当前页面的颜色表。
* **函数算法描述**:遍历当前页面的颜色定义并对每个定义执行指定的操作。
* **与 3.2 中模块的对应关系**: colorTable.c
* **原型和定义保存在哪个 .c/.h 文件中**: colorTable.h 和 colorTable.c

void ChangePageOfFontTable(int p);

* **功能描述**: 更改当前的字体表页面，并在必要时创建新的页面。
* **参数描述**:p: 要切换到的页面编号。
* **返回值描述**: 无。
* **重要局部变量定义**: curFontPage, fontPage, fontPageExsists。
* **重要局部变量用途描述**: curFontPage 记录当前的字体表页面。fontPage 存储字体表的页面。fontPageExsists 标记页面是否存在。
* **函数算法描述**:检查页面是否存在，如果存在则切换到该页面。
  + 如果页面不存在，则创建新页面，并初始化字体表。
* **与 3.2 中模块的对应关系**: fontTable.c
* **原型和定义保存在哪个 .c/.h 文件中**: fontTable.h 和 fontTable.c

**editorCore.c：（更新于6.5 16:58）**

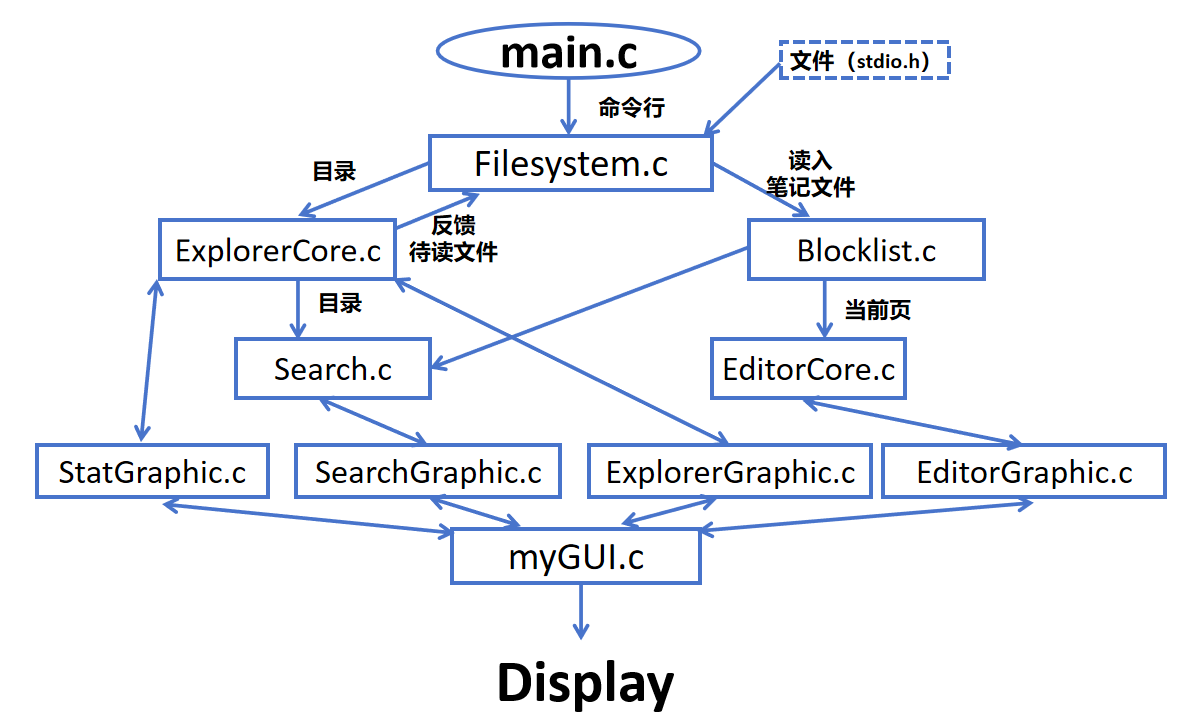
**explorerCore.c：（更新于6.5 16:58）**

## 源代码文件组织设计

<文件目录结构>

1）文件函数结构

因文件与函数过多不可能一一给出，此处仅画出概略图



2）多文件构成机制

**文件包含（File Inclusion）**：

* 在C/C++中，使用#include指令可以将一个文件的内容包含到另一个文件中。
* 这种机制允许将公共的函数、变量或宏定义放在单独的头文件中，并在需要使用它们的地方包含这些头文件。
* 例如，如果有一个名为utils.h的头文件，其中包含一些工具函数的声明，则可以在需要使用这些函数的文件中使用#include "utils.h"来包含这个头文件。
* 本文件广泛使用头文件以实现模块化的多文件程序设计。

**#define 保护**：

* 在头文件中使用宏定义保护可以防止头文件的多次包含，避免因多重包含而引起的重定义错误。
* 做法是在头文件的开头和结尾使用预处理指令，例如：

#ifndef UTILS\_H

#define UTILS\_H

// 头文件内容

#endif // UTILS\_H

* 这样做可以确保头文件只被包含一次，避免了重复定义的问题。
* 本程序实现了对于每一个头文件都做到头文件保护。

**外部变量（External Variables）**：

* 如果一个变量在一个文件中定义，但需要在另一个文件中使用，可以使用extern关键字声明该变量。
* 在变量的定义文件中，使用关键字extern来声明该变量为外部变量；在使用该变量的文件中，则直接使用它。
* 例如，如果在一个文件file1.c中定义了一个外部变量extern int global\_var;，则可以在另一个文件file2.c中直接使用这个变量而不需要重新定义它。
* 外部变量可能会破坏模块的封装性，因此本程序更多地使用Getter与Setter方法或者Traverse方法代替直接暴露全局变量的方法进行数据交互。

**外部函数（External Functions）**：

* 类似于外部变量，如果一个函数在一个文件中定义，但需要在另一个文件中使用，可以使用函数声明和定义的组合来实现。
* 在函数的定义文件中，声明函数为外部函数；在使用该函数的文件中，只需包含函数的声明即可调用该函数。
* 例如，如果在一个文件functions.c中定义了一个函数void myFunction();，则可以在另一个文件中通过包含函数的声明void myFunction();来调用这个函数。
* 本程序将所有允许使用的外部函数包含在头文件中，通过#include对应的头文件来实现外部函数声明，以此增加可读性和模块的封装程度。

# 部署运行和使用说明

## 编译安装

使用DevC++打开.dev文件并编译即可。

## 运行测试

通过运行程序并交互能够发现引起崩溃的操作，从而借助gdb定位引起崩溃的原因进行修改。

## 用户使用手册

见根目录help.txt。

# 团队合作

## 开发计划

### 需求分析阶段：

* **确定系统功能**：明确系统应具备的功能，例如创建、编辑、删除笔记、标记重要笔记、搜索笔记等。
* **用户画像**：了解目标用户，他们的需求、偏好和使用场景，以便设计出更符合用户期望的系统。

### 2. 设计阶段：

* **文件设计**：设计文件结构，包括笔记、用户、标签等相关的表结构，并确定它们之间的关系。
* **界面设计**：设计用户界面，包括页面布局、交互设计、视觉设计等，确保用户友好性和易用性。
* **系统架构设计**：设计系统整体架构，包括前后端分离等。

### 3. 开发阶段：

* **前端开发**：根据设计稿实现前端界面，使用simpleGUI，libgraphics等进行开发。
* **后端开发**：实现系统的业务逻辑和数据处理部分，设计接口供前端调用，并实现文件管理等功能。

### 4. 测试阶段：

* **单元测试**：对系统中的各个模块进行单元测试，确保其功能的正确性。
* **集成测试**：测试系统各个组件之间的集成情况，确保系统各部分能够协同工作。

## 编码规范

### 命名规范：

* **变量命名**：使用有意义的变量名，采用大小写字母的组合
* **常量命名**：使用大写字母和下划线的组合，例如：#define MAX\_SIZE 100;
* **函数命名**：采用驼峰命名法，首字母小写，例如：void calculateAverage();
* **结构体和枚举命名**：采用驼峰命名法，首字母大写，例如：typedef struct PersonInfo;

### 2. 缩进和空格：

* 使用一致的缩进风格。
* 在关键字和标识符之间添加空格，以增加代码的可读性。

### 3. 注释规范：

* 使用注释解释代码的功能、意图、约束等信息，增加代码的可读性。
* 对于复杂的代码逻辑或关键算法，添加适当的注释来说明。

### 4. 函数和模块规范：

* 函数应该短小、功能单一，方便测试和复用。
* 对于复杂的函数，考虑将其拆分成多个小函数，以提高可读性和可维护性。

### 5. 错误处理：

* 对于可能出现错误的地方，进行适当的错误处理，例如检查返回值、使用错误码等。

### 6. 其他规范：

* 避免使用全局变量，尽可能使用局部变量和函数参数。
* 避免使用宏定义，尽可能使用常量或枚举类型。
* 对于内存管理，确保正确地分配和释放内存，避免内存泄漏。

### 7. 格式化和排版：

* 保持代码的一致性，统一的格式化和排版有助于提高代码的可读性。
* 使用适当的空行来组织代码块，提高代码结构的清晰度。

## 任务分工

贡献百分比依照提供的代码行数粗略计算。ImageStructure.c因涉及图像信息的处理，较为复杂，按照提供代码行数的三倍计算。

组员1（组长）：负责总体的架构和代码整合，以及大部分涉及架构的编码工作。（60%）

组员2：负责myGUI.c的初步开发（20%）

组员3：负责fileSystemCore.c、ImageStructure.c的开发（20%）

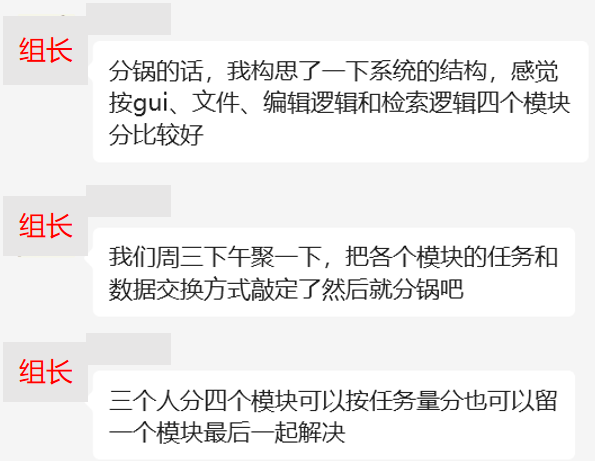
## 个人遇到的难点与解决方案

## 工作总结

**开发过程总结：**

**2024年5月13日：**

由组长带头，初步构思了代码内容和分工。

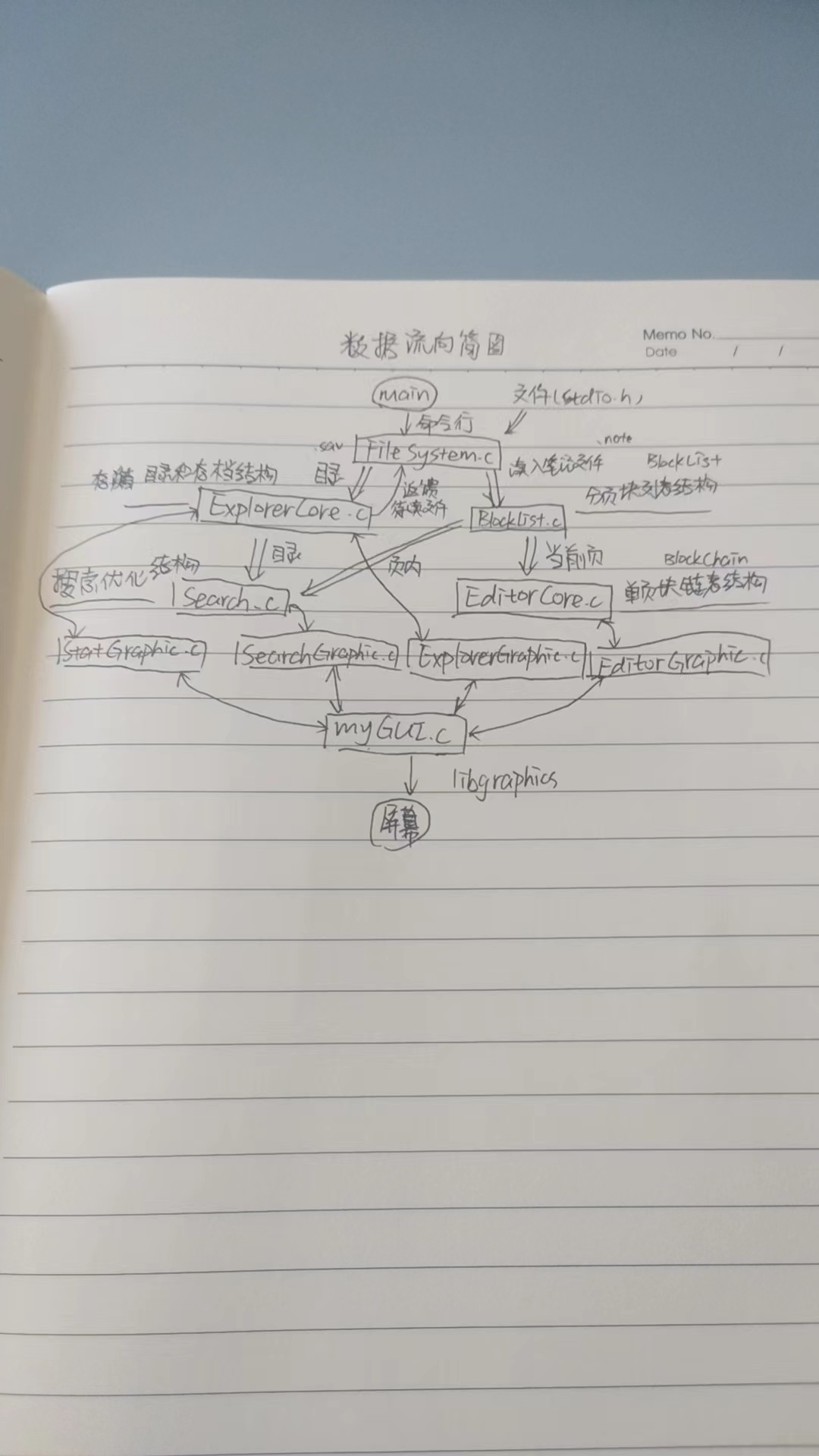
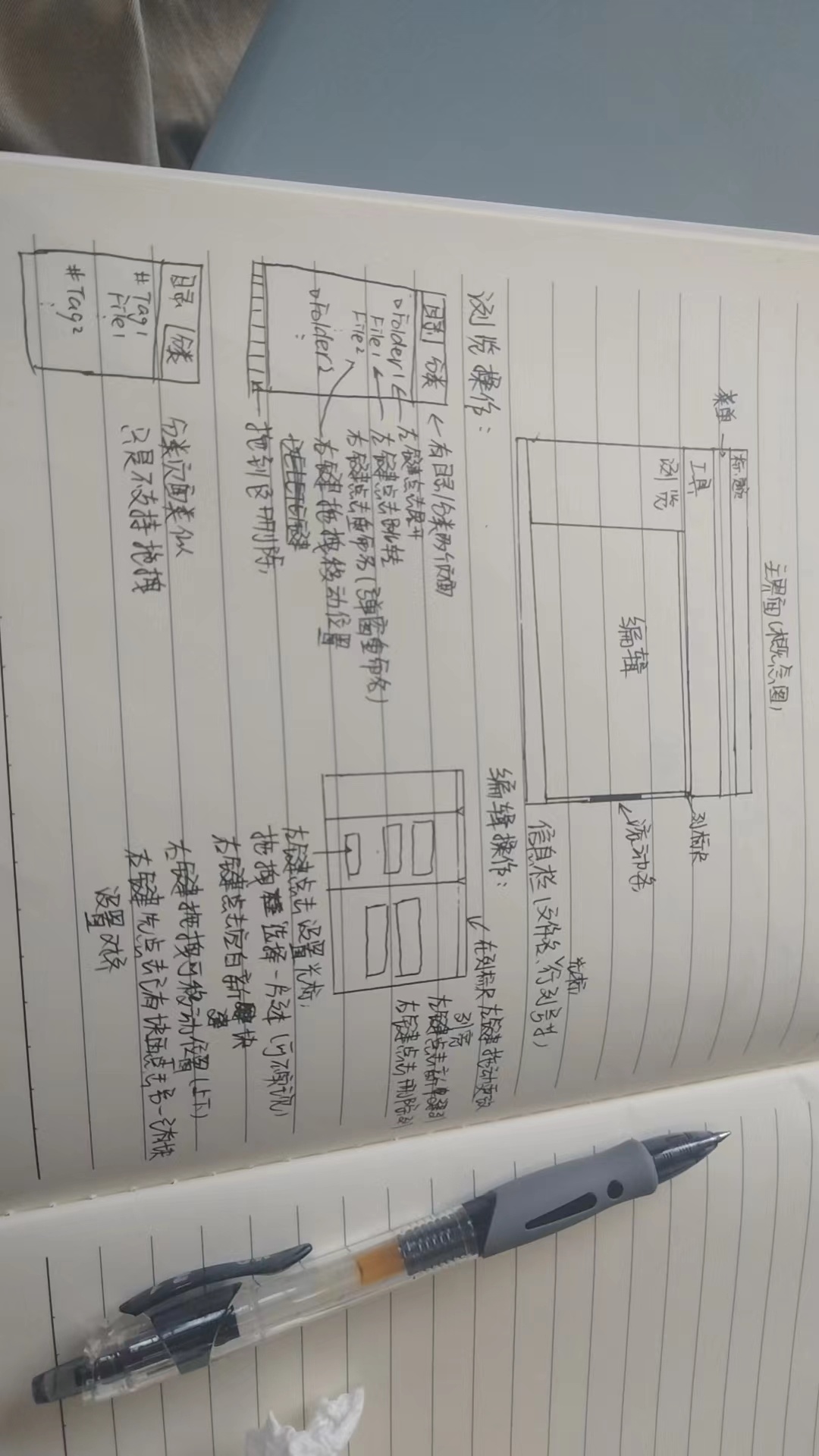
****

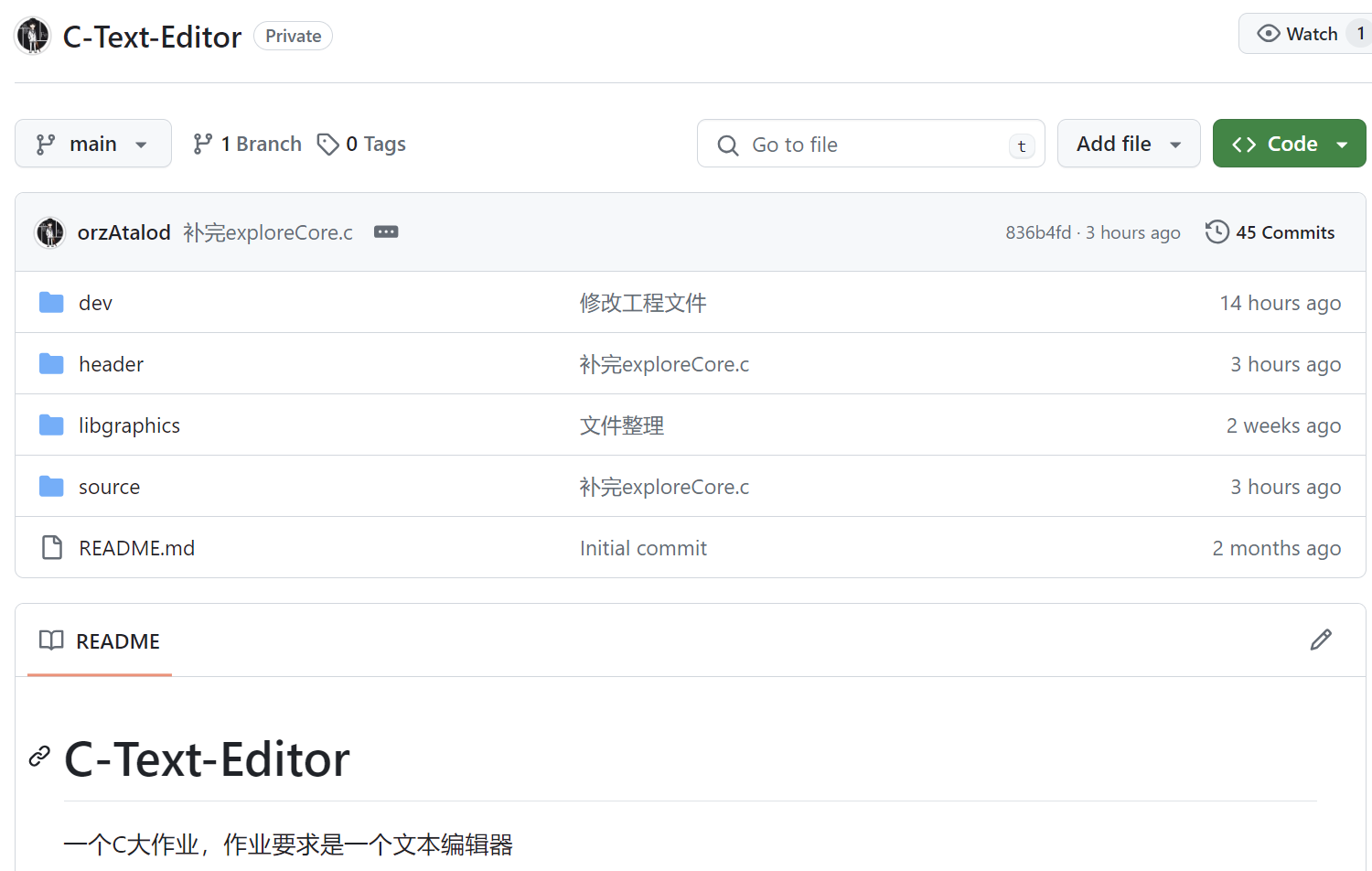
**2024年5月15日：**

进行线下讨论，初步确定了程序的实现目标、具体框架、数据结构等。

并在结束讨论后开放了Github项目，开始合作编程。

确定了由组长编写.h文件与核心结构并上传在Github上，组员实现组长分配的.h文件对应.c内容的形式。



**2024年5月24日：**



针对具体的数据内容和表达有在.h文件中存在的疑惑进行了及时的交流与沟通。

2024年6月1日：

组员相应代码写好后，将调试文件发予组长，以便组长进行相关修改。



## 收获感言

组员1（组长）：本次小组协作使我理解了协作开发软件的主要难点和痛点，为大型软件开发协作的技术打下基础，同时极大的锻炼了我的代码能力。尽管时间非常紧迫，许多预期想要实现的功能都没能实现，我也从失败经历中吸取了宝贵的教训。

组员2：

组员3：

# 参考文献资料

1. SAM（后缀自动机）的论文
2. PNG文件结构的CSDN博客