## 中国矿业大学计算机学院

**2023 级本科生课程设计报告**

课程名称 程序设计综合实践

报告时间 2023年12月2日

学生姓名 \*\*\*\*\*\*\*\*

学 号 \*\*\*\*\*\*\*\*

专 业 计算机科学与技术

任课教师 孙中彬

《程序设计综合实践》课程评分表

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
| 序号 | 课程教学目标 | 考查方式与考查点 | 占比 | 得分 |
| 1 | **目标1：**掌握一门计算机高级语言，并能使用特定的软件开发工具，设计、开发、调试及运行应用程序。 | **软件代码**  使用程序设计集成开发工具设计开发、调试应用程序，考察计算机工程基础知识 | 10% |  |
| 2 | **目标2：**针对具体的应用问题，进行功能需求分析，确定设计目标，并能绘制算法流程图。 | **设计报告**  系统需求、系统流程图、软件界面设计、关键类图及软件扩展描述，考察问题分析能力 | 40% |  |
| 3 | **目标3：**在进行需求分析的基础上，设计软件运行界面、关键类、编写代码，调试并正确运行满足需求的应用程序。 | **软件代码**  软件代码编写、调试、运行演示、系统功能扩展，考察计算机工程实践能力 | 50% |  |
| 总分 | | | 100% |  |

评阅人：

《程序设计综合实践》课程指导教师评语

|  |  |
| --- | --- |
|  | |
| 指导教师签名 |  |

## 目 录

[实验一 简单计算器 1](#_Toc28055)

[1.1 系统概述 1](#_Toc3432)

[1.2 系统设计 1](#_Toc8566)

[1.3 系统实现（运行调试） 4](#_Toc64)

[1.4 系统扩展（贷款计算器） 5](#_Toc29365)

[1.5 总结 6](#_Toc4044)

[实验二 拼图游戏 7](#_Toc14421)

[2.1 系统概述 7](#_Toc26557)

[2.2 系统设计 7](#_Toc9519)

[2.3 系统实现（运行调试） 10](#_Toc30154)

[2.4 系统扩展 12](#_Toc25941)

[2.5 总结 14](#_Toc15888)

[实验三 多文档文本编辑器 15](#_Toc7036)

[3.1 系统概述 15](#_Toc29520)

[3.2 系统设计 15](#_Toc26296)

[3.3 系统实现（运行调试） 18](#_Toc31298)

[3.4 系统扩展 23](#_Toc6928)

[3.5 总结 25](#_Toc30804)

[实验四 CSP模拟认证考试训练 26](#_Toc27569)

[4.1 第一题：密码 26](#_Toc20877)

[4.2 第二题：字符串变换 27](#_Toc31696)

[4.3 第三题：补丁应用 29](#_Toc22130)

[4.4 第四题：通讯延迟 31](#_Toc13174)

[4.5 第五题：木板切割 32](#_Toc10779)

[4.6 总结 34](#_Toc24522)

**实验一 简单计算器**

* 1. 系统概述

本项目旨在开发一个名为My Calculator的四则运算计算器，支持连续计算功能。用户可以通过点击按钮输入类似4+5+6或5\*8+16的连续计算表达式，系统将计算结果并在输出文本框中显示。除基本的四则运算外，计算器还具备清空、后退等辅助功能。系统采用C++语言开发，使用Qt作为图形用户界面框架。IDE为Clion。计算器不仅支持基本的加减乘除运算，还能处理带括号的复杂表达式，并具有实时错误检测功能，为用户提供直观、易用的计算体验。

1.2 系统设计

1.2.1 设计目标

本计算器的主要设计目标是实现一个功能完整、操作简便的四则运算工具。用户界面采用按钮布局，支持通过鼠标点击输入数字和运算符。计算器能够处理连续的算术表达式，包括带括号的复杂计算，并在文本框中实时显示输入和计算结果。系统还应具备清除输入、删除单个字符等辅助功能，以提高使用便利性。在运算过程中，计算器需要进行实时的错误检测，如除零错误或表达式语法错误，并给予用户适当的反馈。总体而言，设计旨在为用户提供一个直观、高效且可靠的计算工具。

1.2.2 设计分析与算法流程

本计算器的核心功能是解析和计算用户输入的数学表达式。设计采用中缀表达式转后缀表达式的算法来处理复杂计算。首先，程序会对输入的表达式进行合法性检查，包括括号匹配和运算符使用是否正确。然后，将中缀表达式转换为后缀表达式，这一过程使用栈来处理运算符的优先级。转换完成后，程序再次使用栈来计算后缀表达式的结果。在计算过程中，系统会进行实时的错误检测，如除零错误。用户界面设计采用Qt框架，通过信号槽机制将按钮点击事件与相应的处理函数连接，实现用户交互。整个计算过程实时反映在显示框中，为用户提供直观的操作体验。

以下是运算算法的流程图：

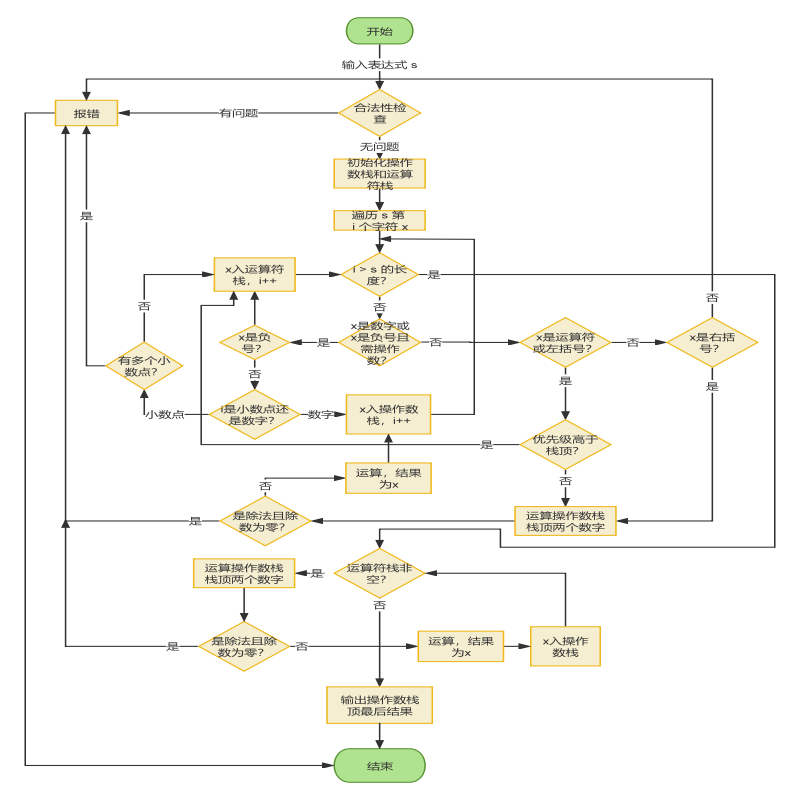


图 1.1 计算器运算算法流程

1.2.3 界面设计

程序页面设计如图所示：

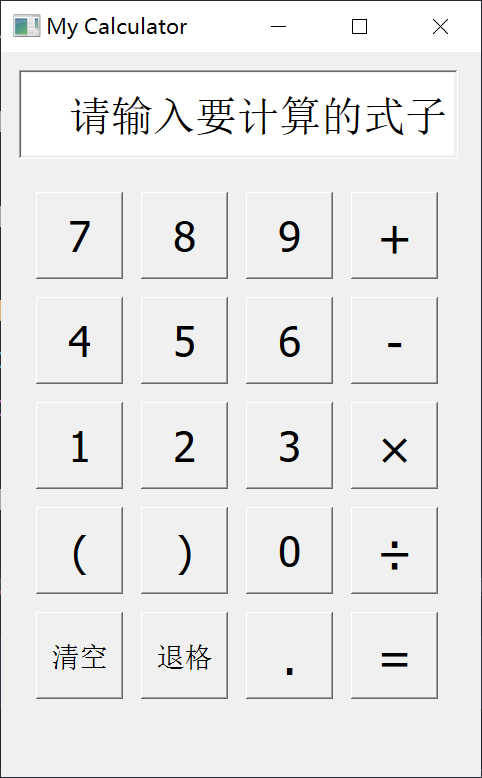


图1.2 计算器界面设计

1.2.4 关键类图

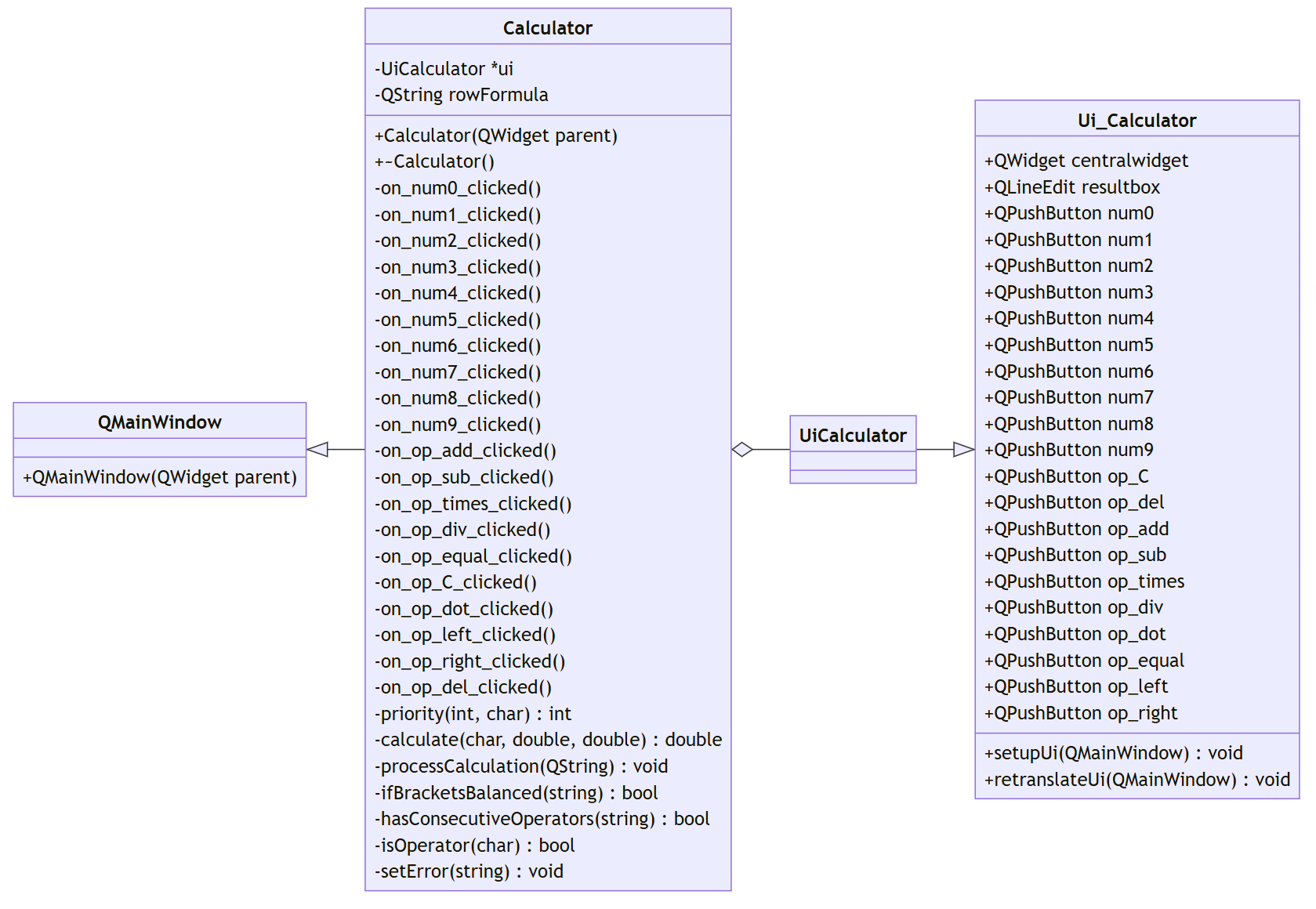


图1.3 计算器关键类图

1.3 系统实现（运行调试）

1. 基本加减乘除、小数点、括号、等于测试。

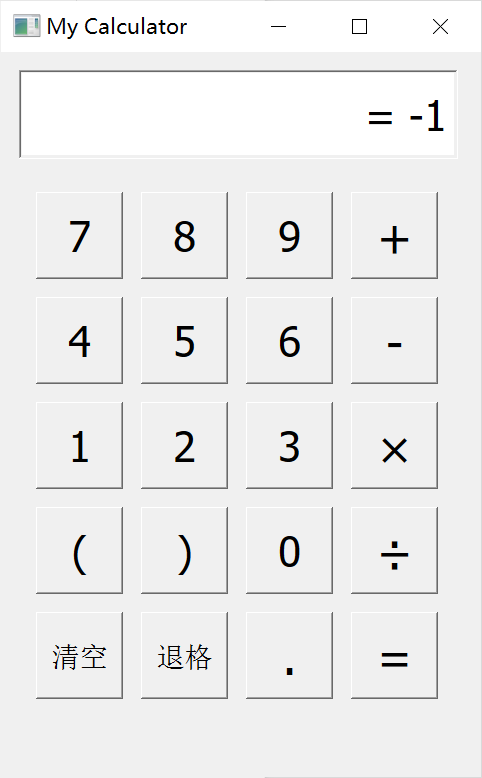
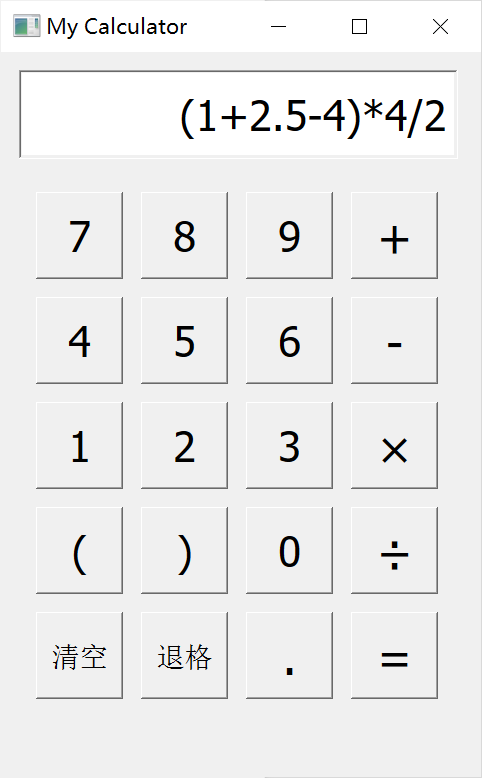


图1.4 计算器计算测试

1. 退格键、清零键测试。

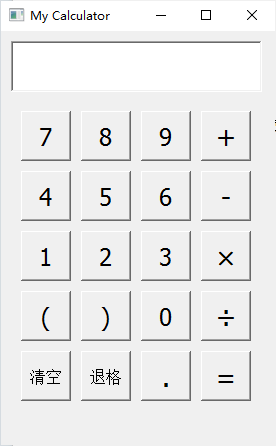
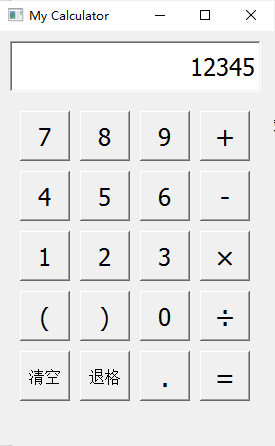


图1.5 计算器退格两次，随后清空

1. 除以零报错测试。

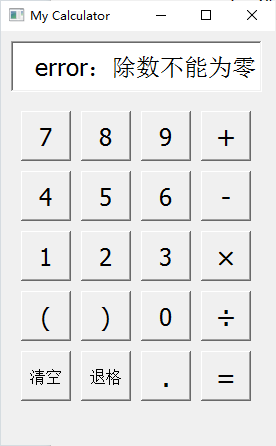
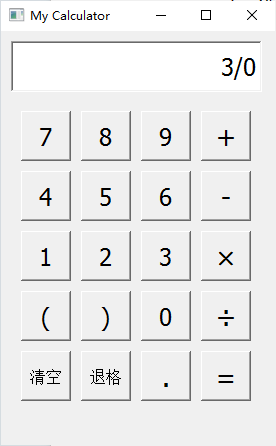


图1.6 除以零报错测试

1. 括号不匹配报错测试。

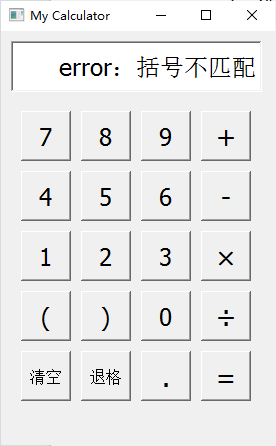
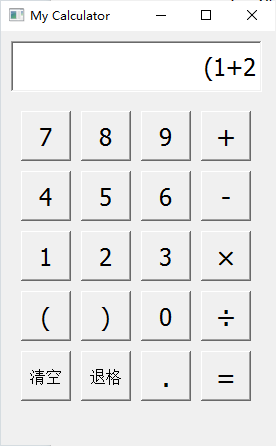


图1.7 括号不匹配报错测试

1. 运算符不匹配报错测试。

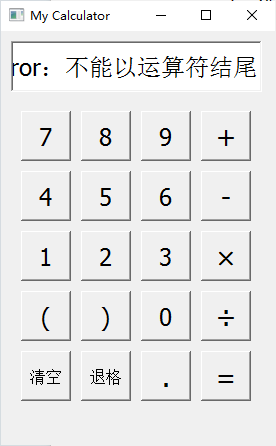
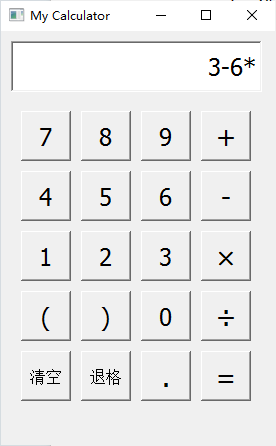


图1.8 运算符不匹配报错测试

1.4 系统扩展（贷款计算器）

贷款是一种重要的金融工具，帮助人们在需要资金时实现各种目标，如购房、购车、创业等。了解贷款的详细还款情况对于借款人来说至关重要，一个好的贷款计算器能够帮助用户即时计算出每个月的还款数额以及总利息支出。这对于个人财务规划非常重要。本程序设置了六个QLineEdit，当用户输入贷款年限、贷款金额和贷款利率的时候，可以根据用户选择的还款方式（等额本息或等额本金），计算出用户的每月平均还款额、利息总额以及还款总额。



图1.9 扩展：贷款计算器

1.5 总结

本次实验成功开发了一个功能完备的四则运算计算器，具备基本的加减乘除和括号运算，同时实现了清空、后退等辅助功能。系统采用C++语言和Qt框架，具有良好的用户界面和操作体验。在此基础上，开发了贷款计算器扩展模块，帮助用户计算贷款的月供及总额。通过实时错误检测和直观的输出展示，项目为用户提供了一个高效可靠的计算工具。系统整体运行稳定，满足设计目标，达成预期效果。通过本次实验，我深入理解了四则运算的算法实现和Qt图形界面的设计。通过逐步完善功能和调试错误，我提升了编程技能和问题解决能力。今后要注重美化程序的UI设计。

**实验二 拼图游戏**

2.1 系统概述

本项目目标是设计开发一个支持点击与空白块交换图片块进行拼图的游戏软件myPuzzle。软件能够自动对加载的图片进行分割，并打乱顺序后放置在不同的拼图块中，用户通过点击与空白块相邻的图片块来进行交换位置，从而完成拼图。拼图成功后，系统会自动进行提示。系统采用Python语言开发，使用PyQt6和QFluentWidgets作为图形用户界面框架，IDE为PyCharm。游戏不仅支持基本的拼图操作，还提供查看原图、添加新图片、重新打乱图片等辅助功能，并可选择不同的拼图难度，为用户提供了丰富的互动体验。

2.2 系统设计

2.2.1 设计目标

本项目旨在开发一款名为myPuzzle的拼图游戏。游戏的主要目标是提供一个简单而有趣的拼图体验。游戏应该有以下基本模块和功能：

基本拼图操作：1、将图片分割成拼图块；2、通过点击移动拼图块；3、检测拼图是否完成

图片管理：1、查看原图功能；2、切换图片功能；3、随机选择图片功能

拼图操作：1、图片重排（重新打乱拼图）；2、自动生成可解的拼图

用户界面：1、使用PyQt6构建图形界面；2、使用QFluentWidgets提供现代UI组件；3、响应式布局

2.2.2 设计分析与算法流程

游戏开始时，系统初始化并打乱拼图，确保生成可解的状态。然后，游戏进入主循环：玩家点击拼图块，系统验证并执行有效移动，同时检查拼图是否完成。这个过程不断重复，直到玩家成功还原拼图，游戏结束。以下是具体流程图：

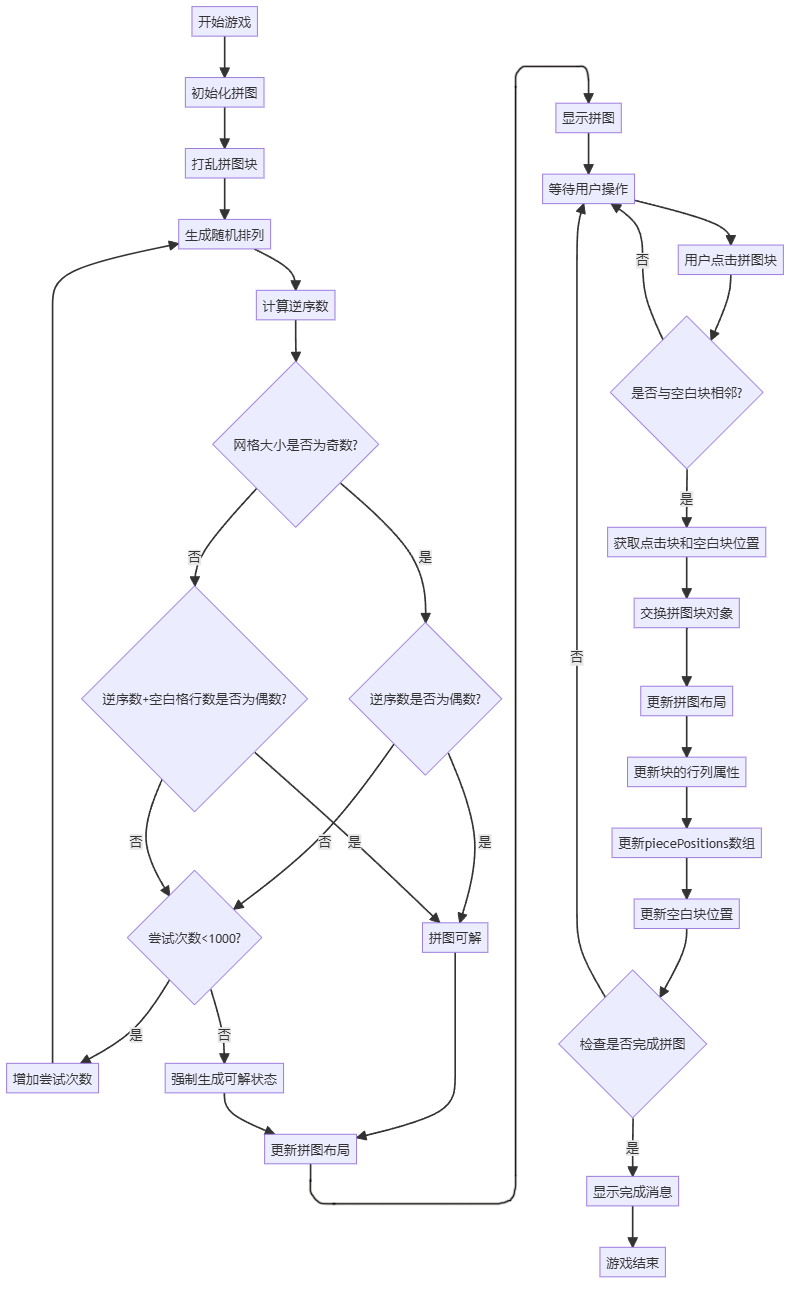


图2.1 基本游戏设计分析与算法流程

2.2.3 界面设计



图2.2游戏界面

2.2.4 关键类图



图2.3拼图游戏关键类图

2.3 系统实现（运行调试）

1. 启动游戏，系统自动加载默认图片，可以直接开始游戏。

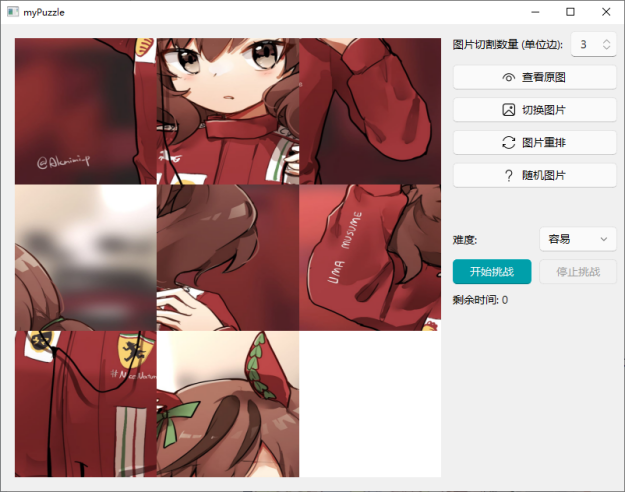


图2.4 默认自由模式

2. 点击与空白块相邻的拼图块，可以将之移动到空白块的位置。

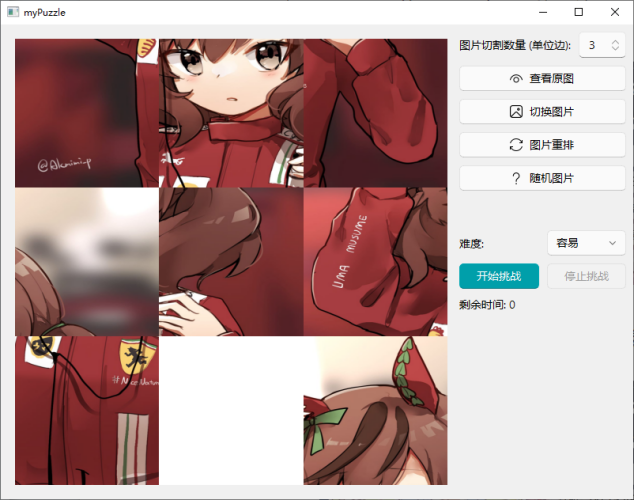


图2.5 交换空白图块 (与图2.4相比)

3. 点击“查看原图”，将新开一个窗口显示原图。显示原图的同时也可以参照原图进行游戏。

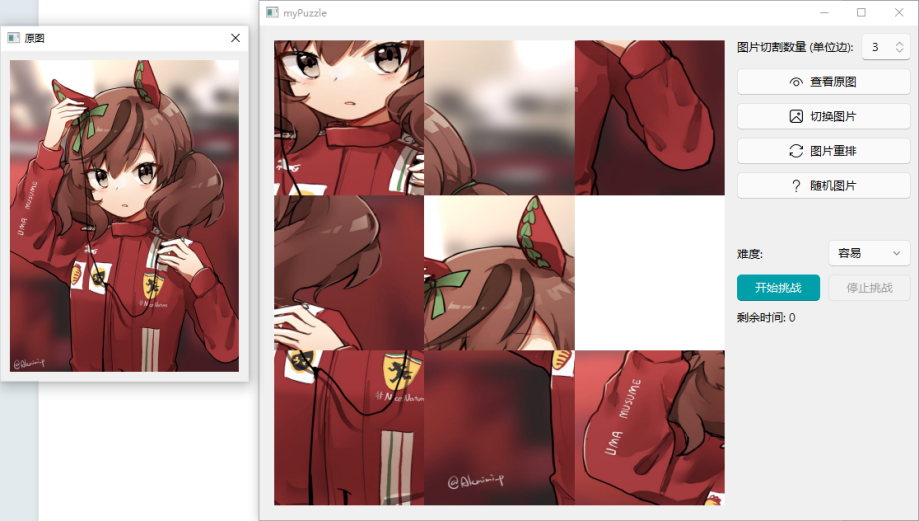


图2.6 查看原图功能

4. 自由模式拼图完成后，自动提示成功界面。



图2.7 提示拼图成功

5. 点击“切换图片”，可以切换其他图片。文件名可以为中文。

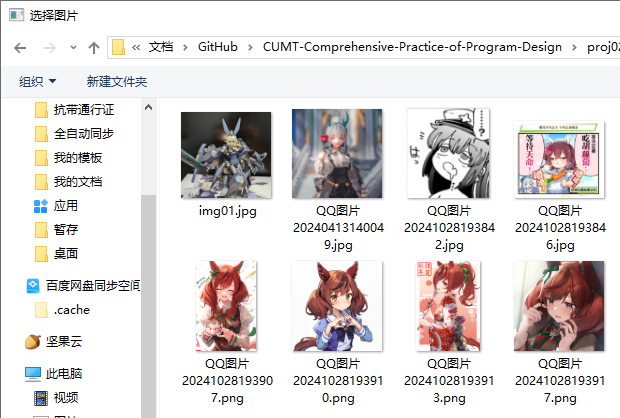


图2.8 切换图片

6. 点击“图片重排”，自动重新生成可解的游戏布局。



图2.9 重新排列图片

7. 点击“随机图片”，会从预存储和先前打开的图片中随机选取图片进行游戏。



图2.10 随机选择图片进行游戏

2.4 系统扩展

默认情况下，游戏为三阶拼图。如果只有三阶拼图则游戏模式过于单调。在拓展中，加入了可自行调整图片切割数量的功能。同时也添加了分为容易、中等、困难三个等级的挑战模式。

手动输入数字或通过按钮自行调整图片切割数量：

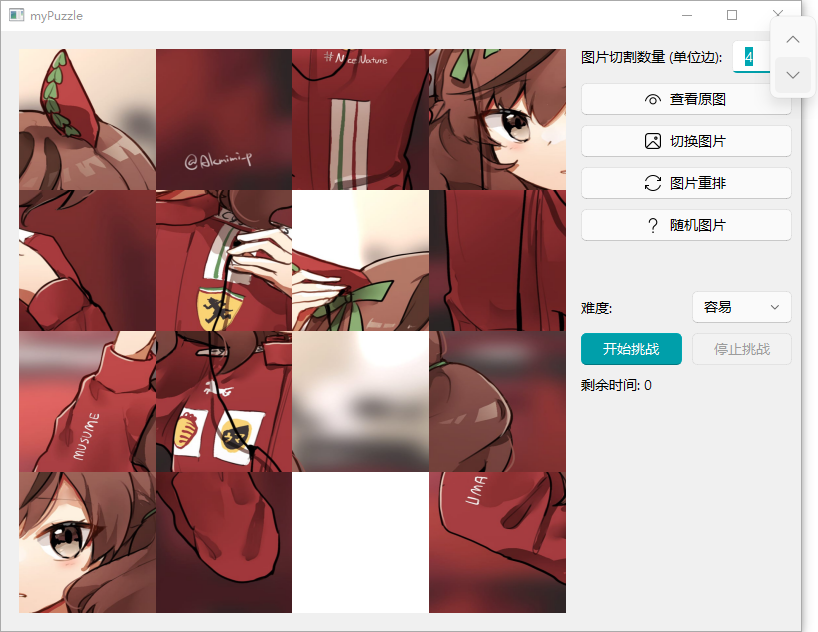


图2.11 自行设置单位边数

开始挑战模式，规定时间内完成拼图则成功，否则失败：



图2.12 调整挑战难度



图2.13 开始挑战模式，自动倒计时

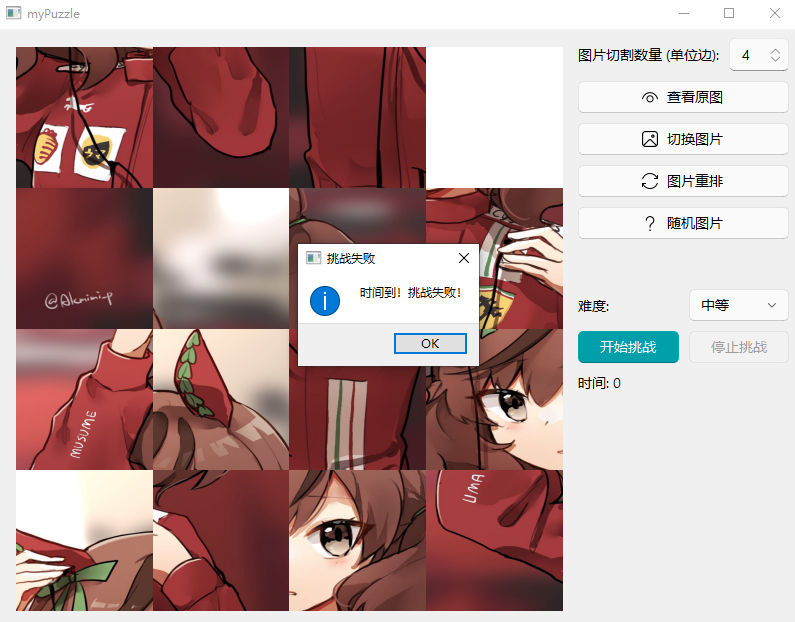


图2.14 挑战失败提示



图2.15 挑战成功提示

2.5 总结

完成这个拼图游戏项目让我收获颇丰。在设计和开发过程中，我深刻体会到了清晰设计思路的重要性。从最初的构想到最终实现，每一步都需要周密考虑。特别是在提升用户体验方面，添加查看原图和调整难度等功能，让我更加关注如何让游戏更有趣、更吸引人。

在技术层面，实现拼图打乱和判断可解性的算法让我对编程有了更深的理解。使用PyQt6和QFluentWidgets进行界面开发，也让我掌握了很多GUI设计的技巧。整个开发过程像是一次小型的软件工程实践，从需求分析到最终测试，让我对软件开发的完整流程有了切身体会。这个项目不仅巩固了我的编程技能，还激发了我继续探索和创新的热情。总的来说，这次经历让我更加确信，软件开发是一个充满挑战但也极其有趣的领域。

**实验三 多文档文本编辑器**

3.1 系统概述

本项目旨在开发一款多文档文本编辑器。该软件采用多文档界面（MDI）设计，允许用户在单一应用程序窗口内同时操作多个独立文档。编辑器将具备文件的基本操作功能，包括新建、打开及保存文本文件。此外，用户可自定义字体及字型，以满足多样化的文本编辑需求。开发环境选用Python语言，配合PyCharm集成开发环境进行编码。项目将使用Qt作为主要的图形用户界面框架，并结合QFluentWidgets库来实现一个功能完备、界面美观且性能优异的多文档文本编辑器。

3.2 系统设计

3.2.1 设计目标

本项目的设计目标是创建一个功能全面、用户友好的多文档界面文本编辑器。它旨在提供一个现代化的工作环境，使用户能够高效地处理多个文本文档。程序的核心目标是实现基本的文本编辑功能，包括创建新文档、打开现有文件、编辑文本内容、保存文件、撤销重做操作、文本对齐设置等功能。通过采用标签式界面和自定义标题栏，程序提供了直观且易于管理的多文档工作环境。同时，程序的界面设计遵循现代UI设计原则，注重美观性和易用性，以满足不同用户群体的需求。

3.2.2 设计分析与算法流程

各个功能模块具体功能如下：

（1）新建文档：创建一个新的标签页，其中包含一个空白的文本编辑框。同时在标签栏中添加相应的标签。新标签页默认命名为"新建标签 X"，其中X为递增的数字。

（2）打开文档：允许用户选择并打开文本文件（.txt）或HTML文件（.html）。读取文件内容并在新建的标签页中显示。如果打开的是HTML文件，保留其格式。打开成功后，将文件名设置为标签名。

（3）保存文档：兼容中文。对于新建的未保存文档，弹出"保存"对话框让用户选择保存路径和文件名。对于已存在的文档，直接保存到原路径。同时，如果文档包含富文本格式，额外保存一个HTML版本。保存成功后更新标签名为文件名。

（4）文本编辑：实现基本的文本输入、删除、选择功能。支持复制、粘贴、剪切等基本操作。编辑操作会更新文档的修改状态。

（5）文本格式化：允许用户选择文本并设置字体、字号、颜色。支持应用粗体、斜体、下划线、删除线等样式。提供文本对齐（左对齐、居中、右对齐）功能。

（6）查找和替换：实现文本查找功能，支持向前和向后查找。替换功能支持单个替换和全部替换。

（7）多标签页管理：实现标签页的添加、切换、关闭功能。每个标签页对应一个独立的文档编辑区域。

（8）自定义标题栏：实现包含菜单、标签栏、窗口控制按钮的自定义标题栏。

（9）响应式布局：使用QVBoxLayout和QHBoxLayout组织界面元素，确保界面能够根据窗口大小的变化进行适当调整。

（10）文件状态管理：为每个文档维护一个修改状态标志。当文档内容变化时更新此标志。在关闭未保存的文档或退出程序时，根据此标志决定是否提示用户保存。

3.2.3 界面设计

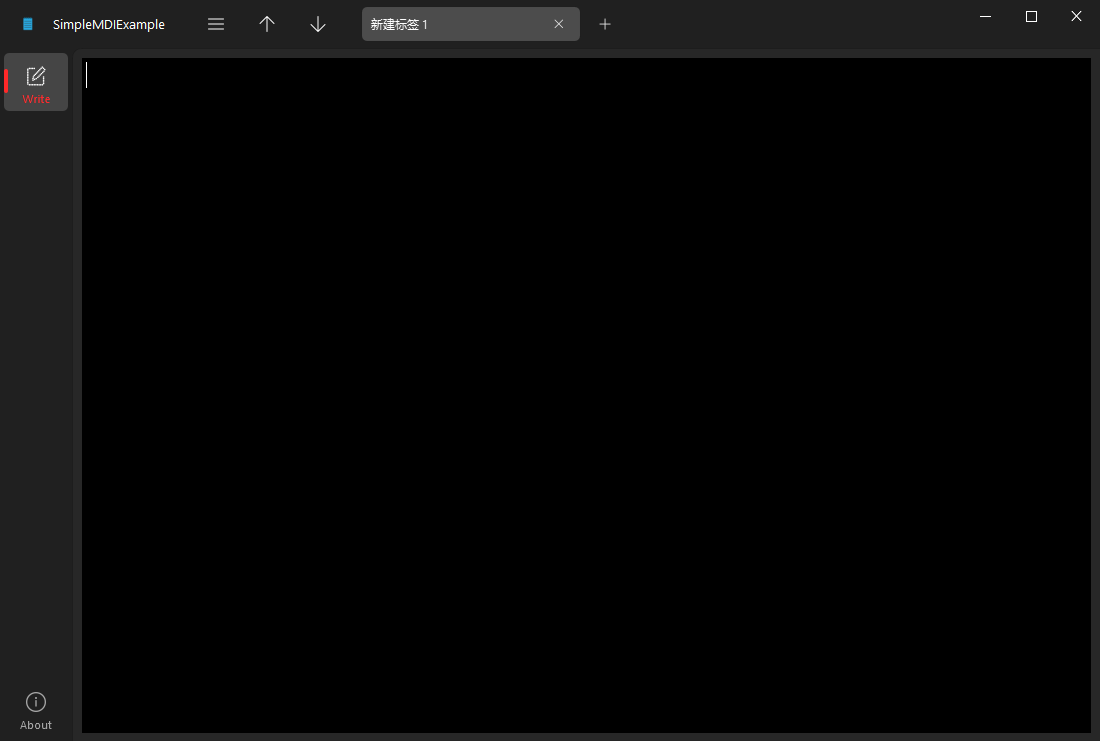


图3.1 多文本编辑器界面

3.2.4 关键类图

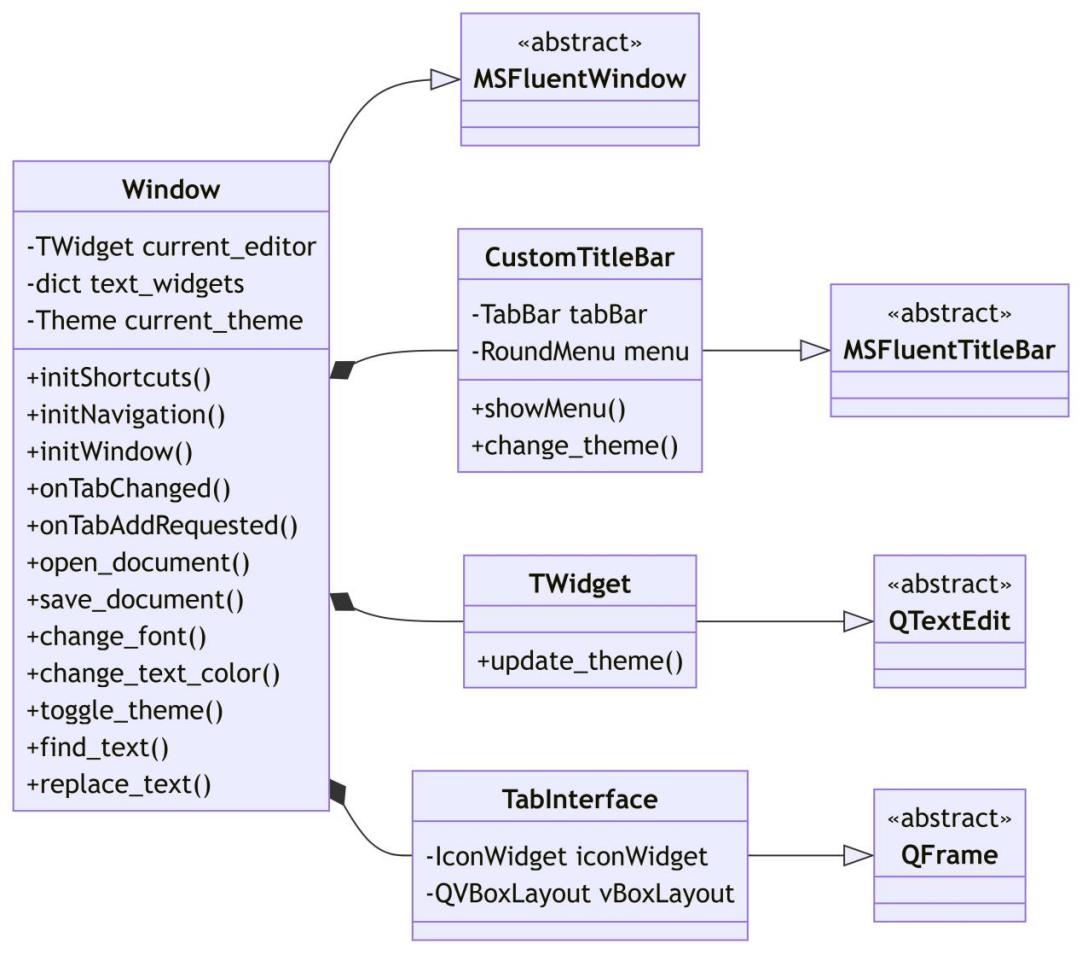


图3.2 多文本编辑器关键类图

3.3 系统实现（运行调试）

1. 打开程序，各功能均在菜单中可以选择。此处出于极简主义及纯键盘开发主义设计，具体说明见系统拓展部分。

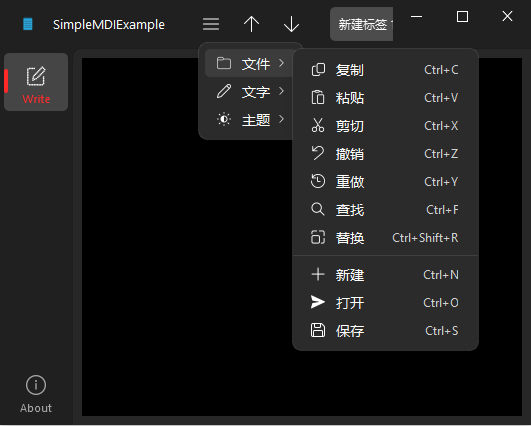


图3.3 菜单文件界面



图3.4 菜单文字界面

1. 点击加号，新建标签页。

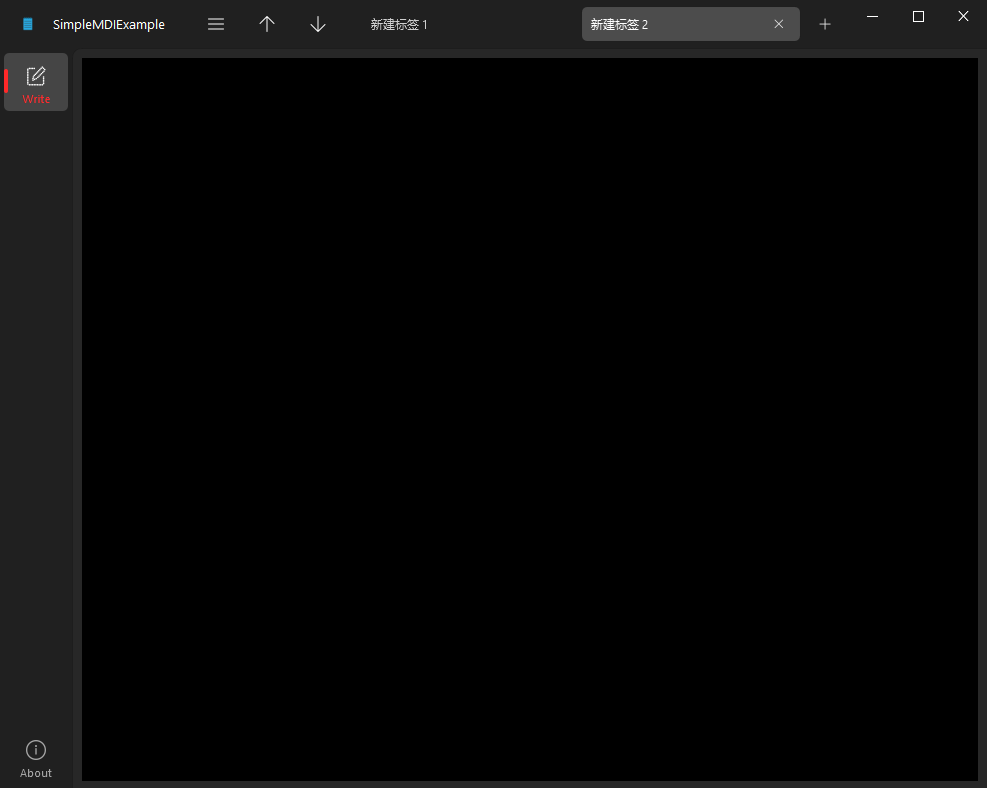


图3.5 新建标签页

1. 点击“打开”，打开已有文件。

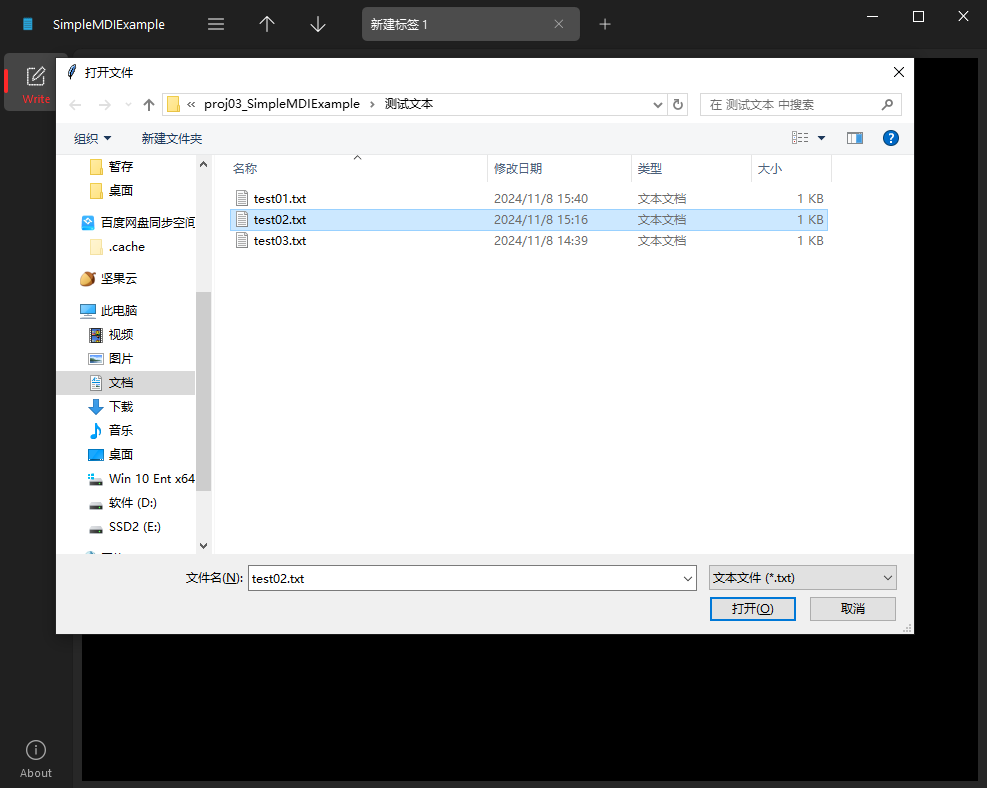


图3.6 打开文件

1. 点击“保存”，保存文件。若是新文件，则弹出保存对话框进行保存。



图3.7 保存文件

1. 复制、粘贴、剪切功能

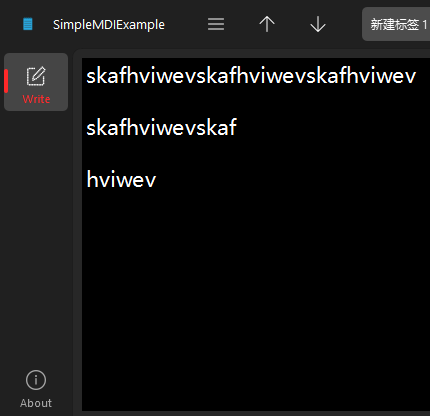
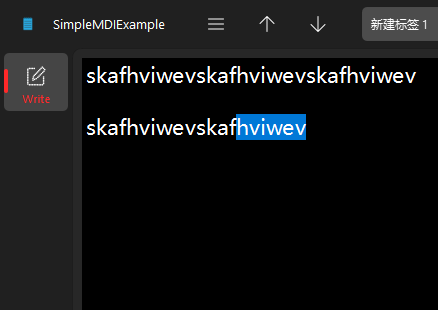


图3.8 复制、粘贴、剪切

1. 撤销、重做功能

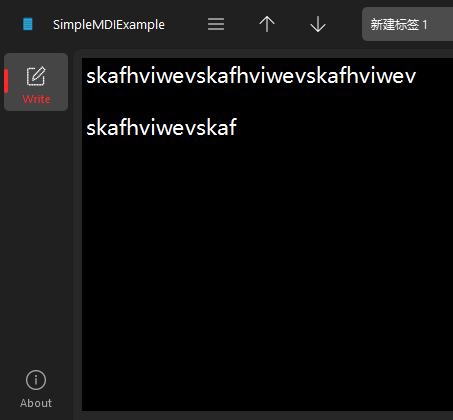


图3.9 撤销、重做

1. 粗体、斜体、下划线、删除线、调整文字字体、大小、颜色功能。

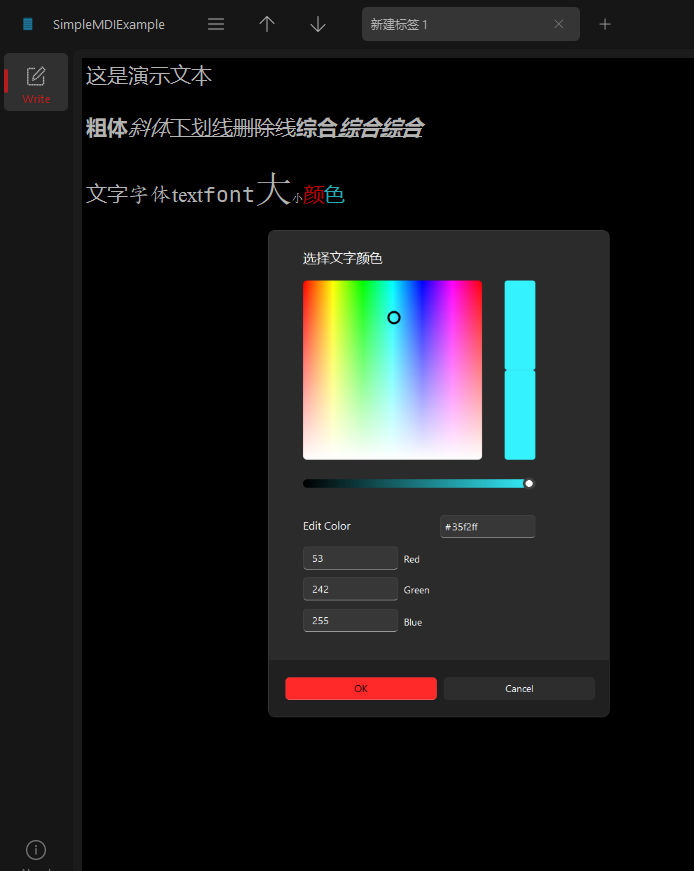


图3.10 粗体、斜体、下划线、删除线、调整文字字体、大小、颜色

1. 文本对齐

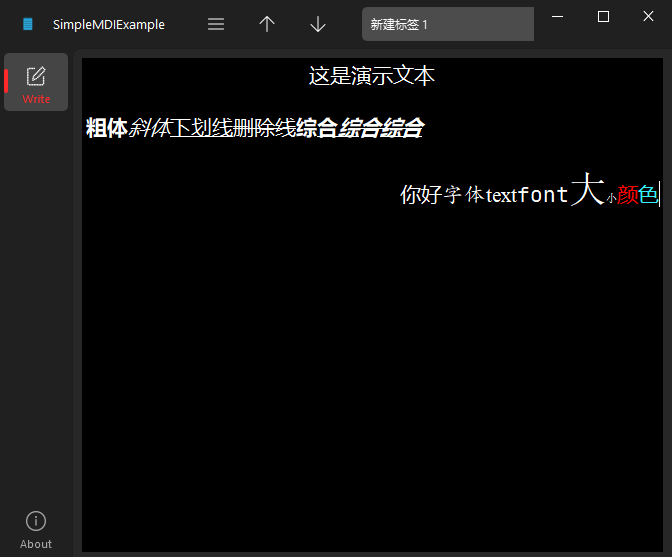


图3.11 居中与左右对齐

1. 查找

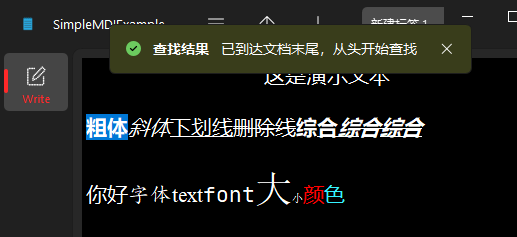
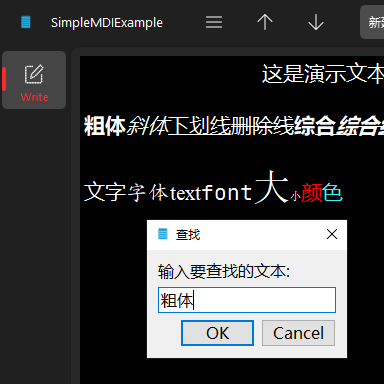


图3.12 查找文本

1. 替换

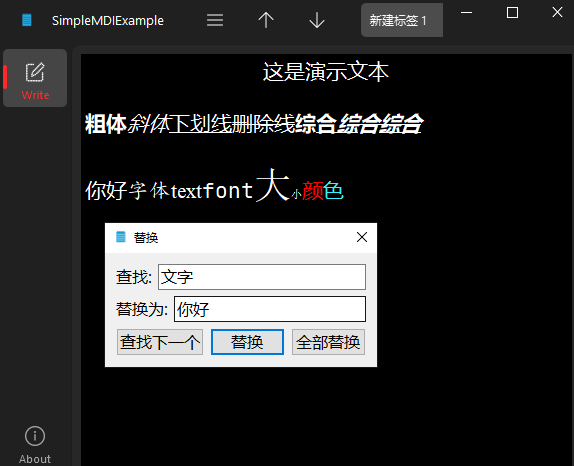
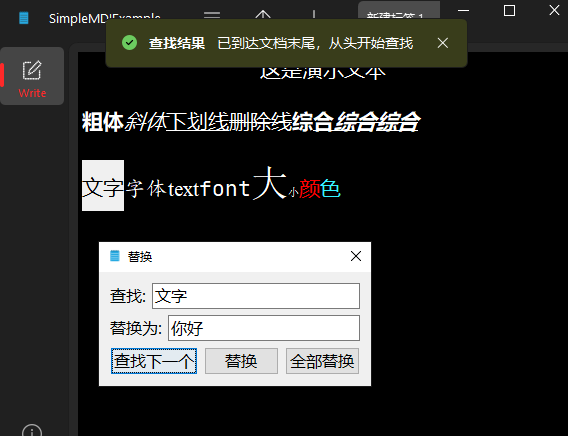


图3.13 替换文本

3.4 系统扩展

相对于基础程序，系统添加了快捷键、主题切换、连续查找替换文本三个扩展内容。

快捷键：本程序的界面设计并没有如参考文档中一样，单独拉出一栏防止复制粘贴等按钮。出于极简主义及键盘开发主义，一个优秀的文本编辑器应该可以仅使用键盘就能完成全部操作。因此程序将所有按钮归入统一的一个菜单按钮中，并为所有功能添加了快捷键。快捷键设置参考JetBrains系列软件设置，快捷键表如下：

表3.1 快捷键表

|  |  |  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- | --- | --- |
| 新建 | 打开 | 保存 | 复制 | 粘贴 | 剪切 | 撤销 | 重做 | 查找 |
| Ctrl+N | Ctrl+O | Ctrl+S | Ctrl+C | Ctrl+V | Ctrl+X | Ctrl+Z | Ctrl+Y | Ctrl+F |

|  |  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- | --- |
| 替换 | 字体 | 颜色 | 左对齐 | 居中对齐 | 右对齐 | 粗体 | 斜体 |
| Ctrl+Shift+R | Alt+A | Alt+C | Alt+← | Alt+↑或↓ | Alt+→ | Ctrl+B | Ctrl+I |

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
| 下划线 | 删除线 | 查找上/下一个 | 切换主题 | (上)下一个标签页 |
| Ctrl+U | Alt+s | Ctrl+↑/↓ | Alt+T | Ctrl(+Shift)+Tab |

主题切换：按Alt+T可以切换暗黑与明亮主题。

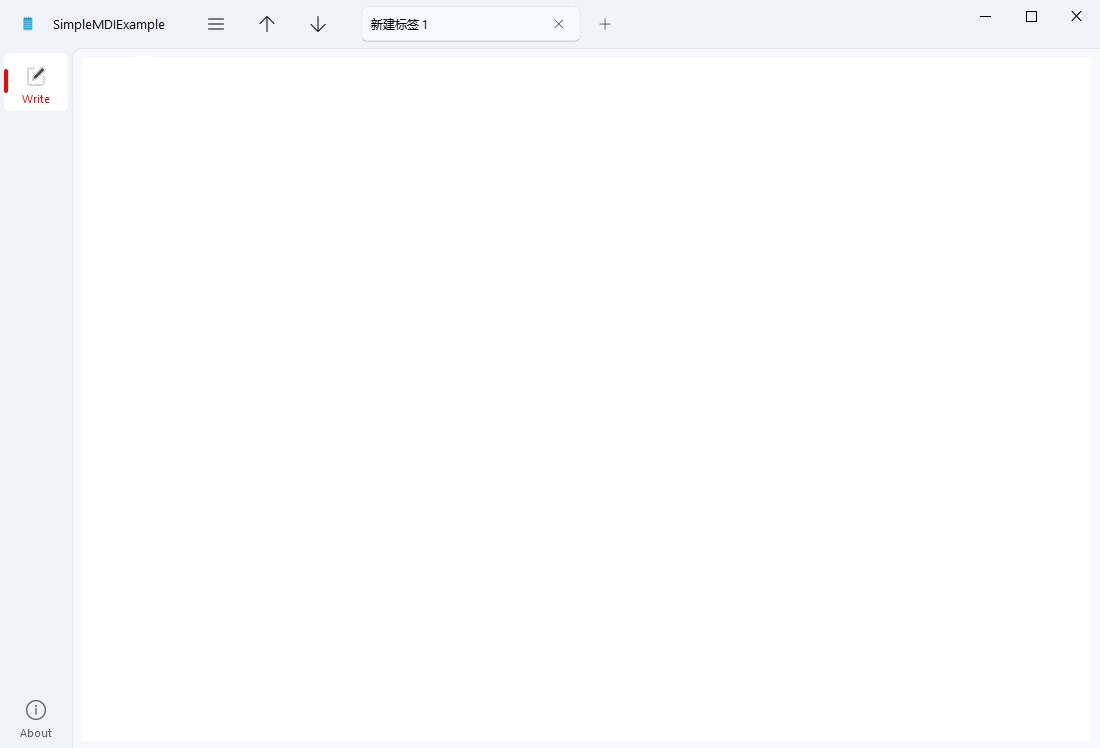


图3.14 明亮主题

连续查找替换文本：按Ctrl+↑/↓可以向上/下连续查找上一次查找的文本。如果是替换，随后按Enter可以替换文本。当查找到文本最后一个后会自动从头查找并弹出提示框。

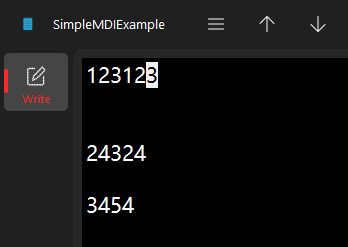
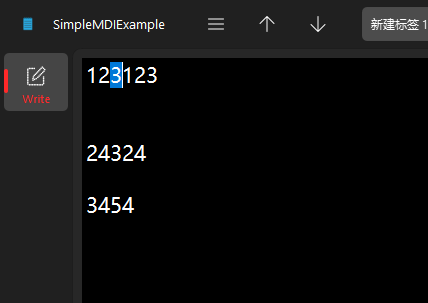


图3.15 向下连续查找

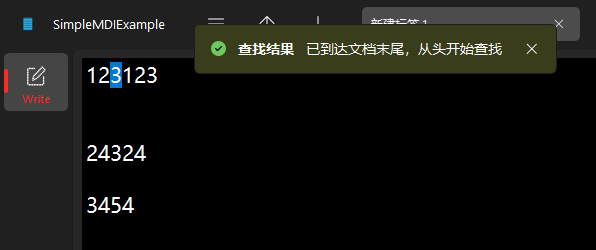


图3.16 自动从头重新查找

3.5 总结

开发SimpleMDIExample多文档界面文本编辑器的过程让我深刻体会到了软件设计和开发的复杂性与挑战性。通过将编辑器功能模块化，我学会了更有效地组织代码，提高了开发效率和可维护性。使用PyQt6和QFluentWidgets进行界面开发，让我深入理解了图形用户界面设计的原则，学会创建既美观又实用的界面。在实现文本编辑、格式化和扩展功能时遇到的技术难题，如富文本处理和主题切换等，不仅提高了我的问题解决能力，也让我更深入地理解了PyQt6的各种机制。本次实验最大收获就是掌握了Qt中各种窗口调用的方法，包括文件打开与保存、字体、颜色、设置以及自定义对话框。

**实验四 CSP模拟认证考试训练**

4.1 第一题：密码

4.1.1 题干大意

根据用户输入的密码字符串，根据密码的复杂程度评定安全级别，分为高（2）、中（1）、低（0）三个级别，其规则如下：

**高安全级别**：密码长度不少于6个字符，同时包含大写字母、小写字母、数字和特殊字符，并且每个字符的出现次数不超过2次。

**中安全级别**：密码长度不少于6个字符，同时包含大写字母、小写字母、数字和特殊字符，但未能满足高安全级别的条件。

**低安全级别**：密码长度不少于6个字符，但未能满足中安全级别的条件。

程序依据以上条件对输入的密码进行分级，并输出安全等级。

4.1.2 解题思路

**输入处理**：读取用户输入的密码数量n和各个密码字符串。

**初始化参数**：为每个密码初始化字符计数数组cha、布尔变量Z、num、spe及安全等级level。

**统计字符出现次数**：遍历每个密码字符串，更新字符计数数组cha。

**检查字符类型**：检查字符**是否**为数字、大写字母、小写字母或特殊字符（如\*或#），分别更新布尔变量num、Z及spe。

**检查字符重复性**：检查字符是否重复超过2次，若是则更新布尔变量diff为false。

**评定安全等级**：根据以下条件评定密码安全等级：

level = 2：满足高安全级别条件

level = 1：满足中安全级别条件

level = 0：默认等级，低安全级别

**输出结果**：输出每个密码的安全等级。

4.1.3 代码流程图

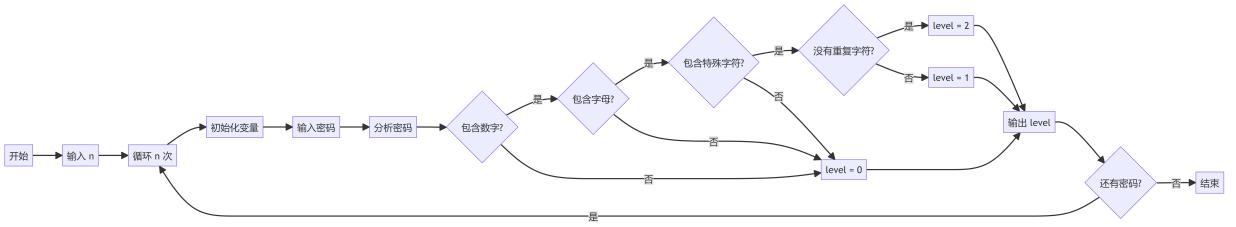


图4.1 第一题代码流程图

4.2 第二题：字符串变换

4.2.1 题干大意

给定一个长度为n的字符串f和一个长度为m的字符串F。定义将字符串f转换为字符串F的操作，每次操作会去除或合并部分字符，需要计算将字符串f变换成字符串F的最小总操作成本。

**输入格式：**共n+4行。第一行：一个整数n表示字符串f的长度。第二行：一个整数m表示字符串F的长度。第三行：一个长度为n的字符串f。第四行：一个长度为m的字符串 F。第五行：n个整数，第i个整数表示将f中第i个字符转移或删除的成本。

**输出格式：**输出：一个整数，表示将f变换为F的最小总操作成本。

4.2.2 解题思路

**读取输入：**首先读取需要处理的字符串str，并去掉首尾的引号。然后读取字符替代规则的数量n，以及之后每条替代规则，将每个字符及其替代映射保存在哈希映射c中。

**构建字符连续序列：**构建每个字符的替代序列charSeq，记录连续替代序列的字符列表。使用布尔标记记录该字符是否形成完整循环isSeq。

**读取操作次数：**读取操作次数m，以及每次操作的参数knum数组。

**执行转换操作：**对字符串中的每个字符逐一进行转换：判断字符是否在替代序列charSeq中。如果在替代序列中，并且形成完整循环，用k对列表长度取模找到替代字符。如果不形成完整循环，直接用参数k选择替代字符，超出序列长度则取最后一个字符。最终生成转换后的新字符串，并按照要求输出。

4.2.3 代码流程图

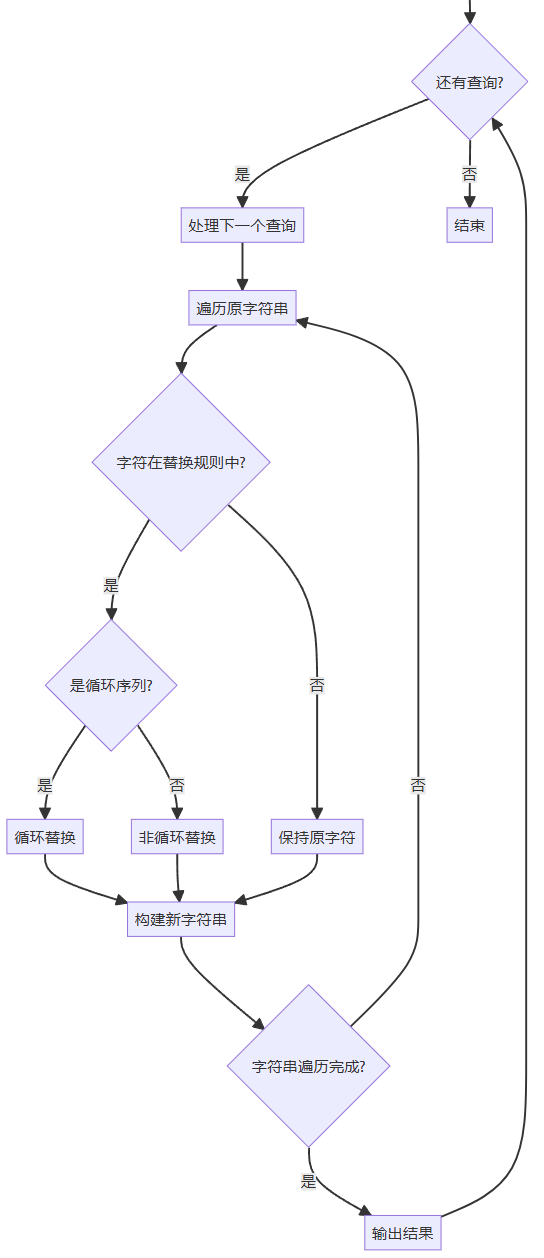
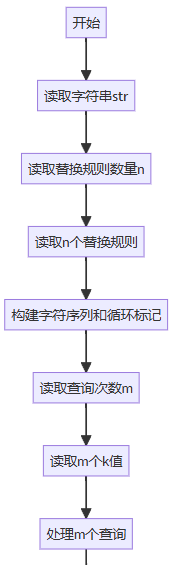


图4.2 第二题代码流程图

4.3 第三题：补丁应用

4.3.1 题干大意

**任务描述：**实现一个patch程序，处理diff工具生成的补丁文件，将修改应用到原代码中。

**输入格式：**首先读取原文件内容。以@开头的行表示补丁块的开始。每个补丁块包含以下内容：第一行采用@@ -NN,MM +nn,mm @@格式，表示修改的位置和范围。其中：

NN表示修改在原文件的起始行。

MM表示修改涉及的原文件行数。

nn表示修改后的起始行。

mm表示修改后的行数。

后续若干行文本，表示具体的变更内容：以-开头的行表示在原文件中删除的文本。以+开头的行表示在新文件中添加的文本。以空格开头的行表示未变化的文本。

在处理补丁文件时需要注意块内行号的匹配，并且按照diff的输出规则，删除的行数和未变化的行数总和应等于MM，而添加的行数和未变化的行数总和应等于mm。

**输出格式：**最终输出应用补丁后的新文件内容。

4.3.2 解题思路

**读取输入：**读取文件的行数 a，并存储原文件的内容到向量 c 中；读取补丁文件的内容到向量 d 中。

**处理每个补丁块：**首先遍历补丁文件内容，当遇到以@@开头的行时，提取补丁块的信息，包括起始行号和行数；然后验证补丁块的行号和行数是否在有效范围内，如果不在范围内，输出“Patch is damaged”并退出。

**解析补丁块的具体修改内容：**创建两个临时向量 k 和 l 分别存储被删除的行和添加的行；继续读取补丁块中定义的修改内容，以-开头的行表示删除行，以+开头的行表示添加行，空格开头的行表示未改变的行。

**验证删除的行内容：**比较原文件中对应位置的行内容与补丁块中标记的删除行内容是否一致，如果不一致，输出“Patch is damaged”并退出。

**应用补丁：**将原文件中被删除的行删除，并在相应位置插入补丁块中添加的新行内容。根据补丁块更新原文件的行数。

**输出结果：**最终输出应用补丁后的新文件内容。

4.3.3 代码流程图

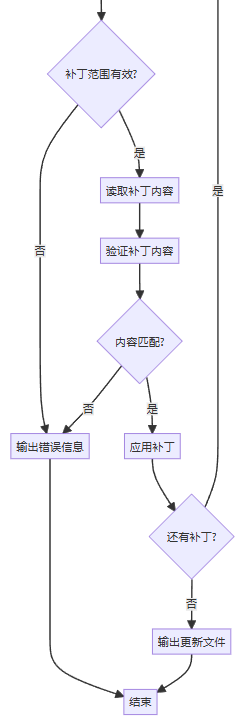
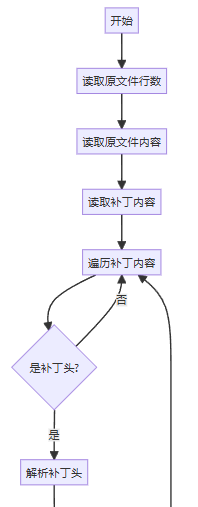


图4.3 第三题代码流程图

4.4 第四题：通讯延迟

4.4.1 题干大意

给定一个二维平面上的n个节点，以及m个通讯基站。每个基站可以覆盖以其坐标为中心的正方形区域，所有节点以单位时间的延迟进行相互通讯。要求从节点1到节点n的最短通讯延迟。

**输入格式：**从标准输入读取输入。第一行包含两个正整数n和m；接下来n行，每行包含一个节点的坐标；接下来m行，每行包含一个通讯基站的坐标和覆盖范围。通讯基站按覆盖范围来保证节点间通讯的延迟。

**输出格式：**输出从节点1到节点n的最短通讯延迟。如果无法通信，则输出Nan。

4.4.2 解题思路

**读取输入：**读取节点数n和基站数m；读取各节点的坐标；读取每个基站的坐标和覆盖范围，并构建图结构，其中每条边的权重表示基站间的延迟时间。

**构建图结构：**通过遍历所有基站和节点，确定哪些节点在基站的覆盖范围内，为这些节点及基站建立双向边，并将对应的延迟时间记录下来。

**初始化距离数组：**将所有节点和基站的初始距离设置为无穷大。

**使用Dijkstra算法求最短路径：**从起点节点1开始，初始化其距离为0，并使用优先级队列维护当前发现的最短路径；不断从优先级队列中取出距离最短的节点，标记其为已处理，然后更新其邻接节点的距离；如果找到更短的路径，则更新优先级队列。

**输出结果：**在处理过程中，如果最终目标节点n的距离仍然是无穷大，则表示无法通信，输出“Nan”；否则，输出从节点1到节点n的最短通讯延迟，由于每条边的延迟时间是加倍的，因此最终结果需要除以2。

4.4.3 代码流程图

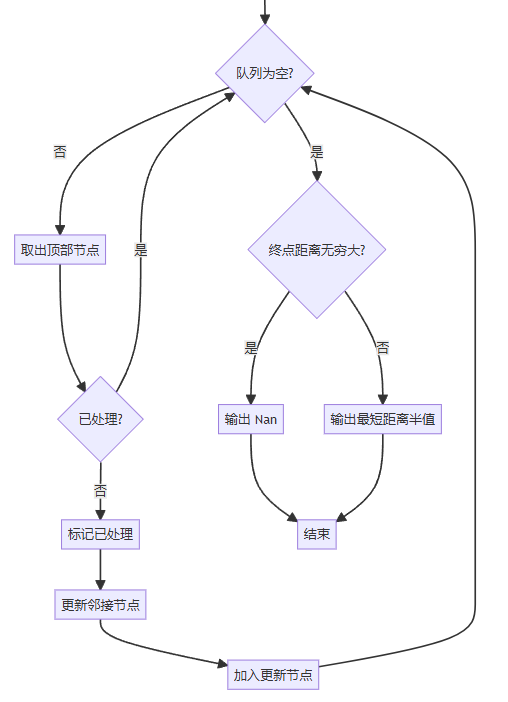
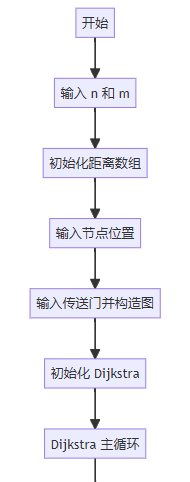


图4.4 第四题代码流程图

4.5 第五题：木板切割

4.5.1 题干大意

给定一块长度为n的木板和m种颜色，木板被平均分成n段，分别编号为1, 2, ..., n。对于每次切割操作，定义如下规则：表示将在木板中第i段至第j段之间所有的段从下标为k，作为第i至第j段，为第1号木板。在此范围外的同色区域应当被认为是一块整体，不评分，即为原木板的公差ci，每块木板被切割次数ai。需要计算每次切割后的下列木板：1.合成多块木板。2.每个木板有不同的颜色。3.每个木板有不同面积。

**输入格式：**从标准输入读取输入。输入从第1+2行。第一行包含三个正整数n，m，k。每段木板有m段。接下来的k每行三个整数ai，bi，ci表示一次切割操作。

**输出格式：**共两行，第一行两个正整数分别表示第k次切割不同颜色段的最大长度c和数量y。

4.5.2 解题思路

**读取输入：**首先读取木板段数n、操作数m、查询数q，然后读取每段木板的颜色并存储于数组a中，将这些木板初始化到Treap数据结构中。

**构建Treap：**初始化Treap，将每段木板推入Treap中，根据其索引及颜色进行维护，每段木板的颜色信息保存在哈希映射中以便快速查找和更新。

**处理查询操作：**对于每个查询，从输入中读取x、l、r三个参数，并调用treap::solve函数执行切割操作。在solve函数中，将Treap中第x段木板从第l到第r段切分出来作为新的木板，并更新原木板和新木板的颜色段数量及种类。根据木板段数和操作数的大小对两个木板进行深度优先搜索dfs，以更新章节的颜色信息。

**最终输出结果：**对于每个查询，输出切割成的新木板的颜色数量和不同颜色段的数量。每次查询都输出两个整数，分别表示新木板的颜色种类数和不同颜色段的数量。

4.5.3 代码流程图

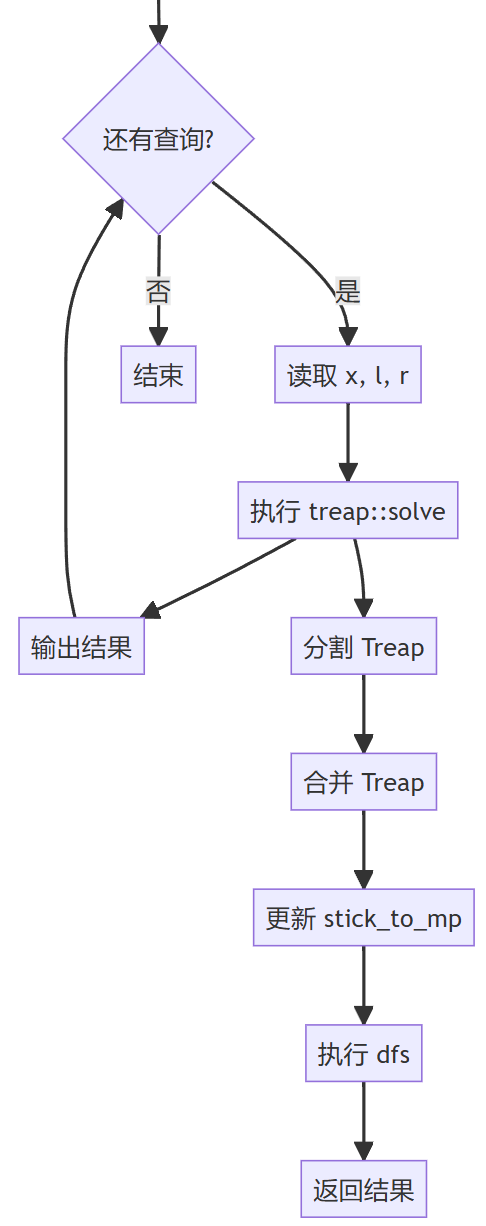


图4.5 第五题代码流程图

4.6 总结

这次CSP模拟认证考试训练让我收获颇丰。通过解答五道不同类型的编程题目，我深入理解了多个重要的算法和数据结构概念，包括字符串处理、动态规划、文件操作、图论算法和高级数据结构的应用。这些题目不仅提高了我的编程技能，还锻炼了我解决复杂问题的思维能力。每道题目都要求仔细分析、设计合适的算法并高效实现，这个过程让我更加熟练地运用各种编程技巧和算法策略。

总的来说，这次训练是对我编程能力的全面检验，为未来参加实际的CSP认证考试打下了坚实的基础。通过这次经历，我更加确信持续练习和学习对提升编程水平的重要性。