# 总体方案设计报告

第二组 任泽高 安轶东 董佳鑫

[总体方案设计报告](#总体方案设计报告)  
 [一，软件开发环境](#一软件开发环境)  
 [二，总体结构](#二总体结构)  
 [项目文件类型分析](#项目文件类型分析)  
 [data文件夹分析](#data文件夹分析)  
 [源文件分析](#源文件分析)  
 [data\_map.json 文件分析](#datamapjson-文件分析)  
 [顶层结构](#顶层结构)  
 [每个景点的结构](#每个景点的结构)  
 [建筑物（builds）](#建筑物builds）)  
 [食物（foods）](#食物foods）)  
 [路线（routes）](#路线routes）)  
 [示例分析](#示例分析)  
 [user\_accounts.json 文件分析](#useraccountsjson-文件分析)  
 [数据结构](#数据结构-1)  
 [分析](#分析-1)  
 [示例](#示例)  
 [diaries.json 文件分析](#diariesjson-文件分析)  
 [数据结构](#数据结构-2)  
 [分析](#分析-2)  
 [三，模块划分](#三模块划分)  
 [3.1 登录模块](#Xea27a26b958fca6a1cce3ac1a50ffa6ca70de60)  
 [login.cpp 的功能分析](#logincpp-的功能分析)  
 [用户登录界面初始化](#用户登录界面初始化)  
 [用户登录逻辑处理](#用户登录逻辑处理)  
 [错误处理与提示](#错误处理与提示)  
 [界面跳转](#界面跳转)  
 [login\_page.cpp 的功能分析](#loginpagecpp-的功能分析)  
 [账户验证](#账户验证)  
 [账户创建](#账户创建)  
 [密码验证](#密码验证)  
 [数据存储与管理](#数据存储与管理)  
 [总结](#总结-1)  
 [3.2 功能选择模块](#Xe94f2b53b68ef74bd9cf3fe64f94ad41f35a8da)  
 [functionselectpage.cpp 的功能分析](#functionselectpagecpp-的功能分析)  
 [初始化和设置](#初始化和设置-1)  
 [功能选择确认](#功能选择确认)  
 [功能页面的切换](#功能页面的切换)  
 [退出功能](#退出功能)  
 [总结](#总结-2)  
 [3.3 日记搜索和浏览模块](#X3d6b5a5a15bcc77a362fbd50896b6c95b13b50b)  
 [diarymanagement.cpp 的功能分析](#diarymanagementcpp-的功能分析)  
 [初始化和设置](#初始化和设置-2)  
 [日记展示](#日记展示)  
 [搜索功能](#搜索功能)  
 [排序功能](#排序功能-1)  
 [日记管理](#日记管理)  
 [界面切换](#界面切换-1)  
 [总结](#总结-3)  
 [3.4 日记创建模块](#Xdfdf6450690054ba8ddba95f16adbdeb426ffbe)  
 [diarycreate.cpp 的功能分析](#diarycreatecpp-的功能分析)  
 [初始化和设置](#初始化和设置-3)  
 [取消创建](#取消创建)  
 [确认创建](#确认创建)  
 [解析日记内容](#解析日记内容)  
 [保存新日记](#保存新日记)  
 [总结](#总结-4)  
 [3.5 日记展示模块](#X034125d946d7c9d2eaec2f47f7b86a80e1f6162)  
 [diaryview.cpp 的功能分析](#diaryviewcpp-的功能分析)  
 [初始化和设置](#初始化和设置-4)  
 [确认操作](#确认操作-1)  
 [读取和解析 JSON 文件](#读取和解析-json-文件)  
 [更新 JSON 文件](#更新-json-文件)  
 [界面切换](#界面切换-2)  
 [总结](#总结-5)  
 [3.6 游学推荐模块](#Xdb2474be06ab5b109b326203974ee451fa90069)  
 [destinationrecommend.cpp 的功能分析](#destinationrecommendcpp-的功能分析)  
 [初始化和设置](#初始化和设置-5)  
 [功能选择](#功能选择)  
 [推荐模式](#推荐模式)  
 [搜索模式](#搜索模式)  
 [排序功能](#排序功能-2)  
 [界面切换](#界面切换-3)  
 [总结](#总结-6)  
 [3.7 美食推荐/设施查询/路线规划模块](#X103f1476d96c07a971ae76ec6db035c91ef0d66)  
 [mymap.cpp 的功能分析](#mymapcpp-的功能分析)  
 [初始化和设置](#初始化和设置-6)  
 [返回操作](#返回操作)  
 [路线规划](#路线规划)  
 [路线编辑](#路线编辑)  
 [设施搜索](#设施搜索)  
 [快速排序](#快速排序)  
 [精确匹配](#精确匹配)  
 [人气和距离筛选](#人气和距离筛选)  
 [总结](#总结-7)  
 [3.8 路线规划逻辑模块](#X03f0252d3deb47bd57310db004f4068fd0ca7f1)  
 [route\_plan.cpp 的功能分析](#routeplancpp-的功能分析)  
 [多点路线规划](#多点路线规划)  
 [两点路线规划](#两点路线规划)  
 [Floyd-Warshall算法](#floyd-warshall算法)  
 [Dijkstra算法](#dijkstra算法)  
 [设施列表](#设施列表)  
 [路径打印](#路径打印)  
 [总结](#总结-8)  
 [3.9 路线绘制模块](#X997dd85fcff28041942052b0fde0188d6f49c7f)  
 [mylabel.cpp 的功能分析](#mylabelcpp-的功能分析)  
 [初始化和设置](#初始化和设置-7)  
 [路线绘制事件](#路线绘制事件)  
 [单点路线绘制](#单点路线绘制)  
 [多点路线绘制](#多点路线绘制)  
 [路线更新](#路线更新)  
 [总结](#总结-9)  
 [3.10 多点路线规划的编辑模块](#X4ab52ccf22725502a93020b0677b164bb1338bf)  
 [editmultipoint.cpp 的功能分析](#editmultipointcpp-的功能分析)  
 [初始化和设置](#初始化和设置-8)  
 [确认操作](#确认操作-2)  
 [总结](#总结-10)  
 [3.11 读取地图文件和地图构建模块](#X3422e4c1ff04c854b2ce27f6f822293e069ee0f)  
 [myGraph.cpp 的功能分析](#mygraphcpp-的功能分析)  
 [景点数据加载](#景点数据加载)  
 [景点数据保存](#景点数据保存)  
 [餐饮信息加载](#餐饮信息加载)  
 [总结](#总结-11)  
 [3.12 哈夫曼编码模块](#Xd2fd5dbbff328660590cc36b32d3e90302c07b9)  
 [huffman.cpp 的功能分析](#huffmancpp-的功能分析)  
 [哈夫曼树的生成](#哈夫曼树的生成)  
 [哈夫曼编码生成](#哈夫曼编码生成)  
 [字符串编码和解码](#字符串编码和解码)  
 [编码表的保存和加载](#编码表的保存和加载)  
 [字符串压缩和解压缩](#字符串压缩和解压缩)  
 [二进制文件读写](#二进制文件读写)  
 [总结](#总结-12)  
 [3.13 kmp字符匹配模块](#Xbc22a8d876616c9ea69bf6d978b4fe06fc41ca1)  
 [kmp.cpp 的功能分析](#kmpcpp-的功能分析)  
 [构建部分匹配表](#构建部分匹配表)  
 [KMP字符串匹配](#kmp字符串匹配)  
 [模式匹配](#模式匹配)  
 [总结](#总结-13)  
 [3.14 程序进入模块](#X7042e97bdd35c3eeb55219086c4ae24339956db)  
 [main.cpp 的功能分析](#maincpp-的功能分析)  
 [初始化和加载数据](#初始化和加载数据)  
 [Qt 应用程序启动](#qt-应用程序启动)  
 [程序主循环](#程序主循环)  
 [总结](#总结-14)

## 一，软件开发环境

Windows 11 操作系统

Visual Studio Code

MSVC 2019 64-Bit

Qt 6.5.3

## 二，总体结构

项目结构示意图如下所示：

project\_root/   
│   
├── build-StudyTourSystem-Desktop\_Qt\_6\_5\_3\_MSVC2019\_64bit-Debug/   
│ ├── bupt\_map.jpg   
│ ├── bupt\_map.bmp   
│ ├── data\_map.json   
│ ├── diaries.json   
│ ├── user\_accounts.json   
│ └── data/   
│   
└── StudyTourSystem/   
 ├── cJSON.c   
 ├── destinationrecommend.cpp   
 ├── diarycreate.cpp   
 ├── diarymanagement.cpp   
 ├── diaryview.cpp   
 ├── editmultipoint.cpp   
 ├── functionselectpage.cpp   
 ├── huffman.cpp   
 ├── kmp.cpp   
 ├── login.cpp   
 ├── login\_page.cpp   
 ├── main.cpp   
 ├── myGraph.cpp   
 ├── mylabel.cpp   
 ├── mymap.cpp   
 ├── route\_plan.cpp   
 ├── StudyTourSystem.pro   
 ├── destinationrecommend.ui   
 ├── diarycreate.ui   
 ├── diarymanagement.ui   
 ├── diaryview.ui   
 ├── editmultipoint.ui   
 ├── functionselectpage.ui   
 ├── login.ui   
 ├── mymap.ui   
 ├── cJSON.h   
 ├── destinationrecommend.h   
 ├── diarycreate.h   
 ├── diarymanagement.h   
 ├── diaryview.h   
 ├── editmultipoint.h   
 ├── functionselectpage.h   
 ├── huffman.h   
 ├── kmp.h   
 ├── login.h   
 ├── login\_page.h   
 ├── myGraph.h   
 ├── mylabel.h   
 ├── mymap.h   
 ├── route\_plan.h   
 ├── route\_plan\_page.h   
 ├── ui\_destinationrecommend.h   
 ├── ui\_diarycreate.h   
 ├── ui\_diarymanagement.h   
 ├── ui\_diaryview.h   
 ├── ui\_editmultipoint.h   
 ├── ui\_functionselectpage.h   
 ├── ui\_login.h   
 └── ui\_mymap.h

### 项目文件类型分析

项目文件可分为“头文件”，“源文件”，“UI文件”，“数据文件”，“Qt项目文件”五大类：

1. 头文件：各种.h文件，主要包含函数的声明，各种全局变量的设定，数据结构的定义，便于其他模块引用相关内容
2. 源文件：实现各种逻辑和算法的文件，主要是cpp文件和c文件，如 destinationrecommend.cpp、diarycreate.cpp 等。实现了.h文件中的函数，与程序所有算法逻辑。是主要功能模块源代码
3. UI文件：Qt中用于设计用户界面的文件。例如，diarycreate.ui 定义了日记创建界面的布局，包含输入框、按钮等 UI 元素。
4. 数据文件：包括json文件和图片，用于后台数据的存储，包括用户信息，地图信息，日记信息等。
5. Qt项目文件：StudyTourSystem.pro，Qt 项目文件，定义了项目的配置信息，如源文件、头文件、库依赖等。该文件用于指导 qmake 工具生成适合不同平台的 Makefile。

### data文件夹分析

data文件夹中，存放着压缩后的日记文件的内容。

1. **哈夫曼编码表的 JSON 文件**：每个压缩字符串的哈夫曼编码表都会被保存为一个 JSON 文件，文件名格式为 huffmanCodes\_ID.json，其中 ID 是对应压缩字符串的唯一标识符。
2. **压缩后的二进制文件**：每个日记在压缩后会被存储为一个二进制文件，文件名格式为 compressedString\_ID.bin，其中 ID 同样是对应压缩日记内容的唯一标识符。

### 源文件分析

1. 登录模块: Login.cpp 与 Login\_page.cpp
2. 功能选择模块: functionseLectpage.cpp
3. 日记搜索和浏览模块: diarymanagement.cpp
4. 日记创建模块: diarycreate.cpp
5. 日记展示模块: diaryview.cpp
6. 游学推荐模块: destinationrecommend.cpp
7. 美食推荐/设施查询/路线规划模块: mymap.cpp
8. 路线规划逻辑模块:route\_plan.cpp
9. 路线绘制模块: mylable.cpp
10. 多点路线规划的编辑模块: editmultipoint.cpp
11. 读取地图文件和地图构建模块:myGraph.cpp

### data\_map.json 文件分析

该 JSON 文件包含多个景点（Attractions）的信息，每个景点都包括详细的属性和关联的数据。这些景点的数据结构可以分为以下几个部分：

#### 顶层结构

JSON 文件是一个数组，其中每个元素代表一个景点。

#### 每个景点的结构

每个景点对象包含以下属性：

1. **name**: 景点的名称，例如 "Sally 河"。
2. **ID**: 景点的唯一标识符，例如 0。
3. **isSchool**: 布尔值，表示该景点是否是学校，例如 false。
4. **popular**: 景点的受欢迎程度，例如 90。
5. **rating**: 景点的评分，例如 0.29。
6. **builds**: 一个数组，包含景点内的建筑物信息。
7. **foods**: 一个数组，包含景点内的食物信息。
8. **routes**: 一个数组，包含景点与其他景点之间的路线信息。

#### 建筑物（builds）

builds 数组中的每个对象代表一个建筑物，包含以下属性：

1. **name**: 建筑物的名称，例如 "Isabella 食堂"。
2. **ID**: 建筑物的唯一标识符，例如 0。
3. **isFacility**: 布尔值，表示该建筑物是否是设施，例如 false。
4. **coordinates**: 数组，包含建筑物的坐标，例如 [29, 159]。

#### 食物（foods）

foods 数组中的每个对象代表一种食物，包含以下属性：

1. **name**: 食物的名称，例如 "披萨"。
2. **restaurantID**: 餐馆的唯一标识符，例如 0。
3. **cuisine**: 食物的菜系，例如 "西餐"。
4. **popularity**: 食物的受欢迎程度，例如 71。

#### 路线（routes）

routes 数组中的每个对象代表一条路线，包含以下属性：

1. **start**: 起点的建筑物 ID，例如 1。
2. **end**: 终点的建筑物 ID，例如 3。
3. **start\_coordinates**: 起点的坐标，例如 [206, 166]。
4. **end\_coordinates**: 终点的坐标，例如 [281, 190]。
5. **midpoint**: 路线的中点坐标，例如 [206, 190]。
6. **distance**: 路线的距离，例如 99。
7. **walkCongestion**: 步行拥堵程度，例如 0.8。
8. **bikeCongestion**: 自行车拥堵程度，例如 0.67。
9. **electricCongestion**: 电动车拥堵程度，例如 0.78。

#### 示例分析

以下是景点 "Sally 河" 的示例数据（为方便在文档展示，这里的数据量不符合要求）：

{  
 "name": "Sally 河",  
 "ID": 0,  
 "isSchool": false,  
 "popular": 90,  
 "rating": 0.29,  
 "builds": [  
 {  
 "name": "Isabella 食堂",  
 "ID": 0,  
 "isFacility": false,  
 "coordinates": [29, 159]  
 },  
 {  
 "name": "Lucy 教学楼",  
 "ID": 1,  
 "isFacility": false,  
 "coordinates": [206, 166]  
 },  
 {  
 "name": "Emma 咖啡厅",  
 "ID": 2,  
 "isFacility": true,  
 "coordinates": [48, 113]  
 },  
 {  
 "name": "Hu Supermarket",  
 "ID": 3,  
 "isFacility": true,  
 "coordinates": [281, 190]  
 }  
 ],  
 "foods": [  
 {  
 "name": "披萨",  
 "restaurantID": 0,  
 "cuisine": "西餐",  
 "popularity": 71  
 },  
 {  
 "name": "酸辣汤",  
 "restaurantID": 0,  
 "cuisine": "湖南菜",  
 "popularity": 2  
 }  
 ],  
 "routes": [  
 {  
 "start": 1,  
 "end": 3,  
 "start\_coordinates": [206, 166],  
 "end\_coordinates": [281, 190],  
 "midpoint": [206, 190],  
 "distance": 99,  
 "walkCongestion": 0.8,  
 "bikeCongestion": 0.67,  
 "electricCongestion": 0.78  
 },  
 {  
 "start": 2,  
 "end": 1,  
 "start\_coordinates": [48, 113],  
 "end\_coordinates": [206, 166],  
 "midpoint": [206, 113],  
 "distance": 211,  
 "walkCongestion": 0.83,  
 "bikeCongestion": 0.83,  
 "electricCongestion": 0.96  
 }  
 ]  
}

### user\_accounts.json 文件分析

该 JSON 文件包含一个名为 "accounts" 的对象，表示用户账号和密码的键值对。这种结构用于简单的账号管理系统，其中每个账号（用户名）都对应一个密码。

#### 数据结构

{   
 "accounts": {   
 "1": "1",   
 "11": "11",   
 "123": "123",   
 "22": "22",   
 "222": "2222",   
 "3443": "3434",   
 "username1": "password1",   
 }   
}

#### 分析

1. **顶层对象**: JSON 文件的根对象包含一个名为 "accounts" 的对象。
2. **"accounts" 对象**: 这个对象存储了多个键值对，每个键值对代表一个账号和密码。
   * 键: 表示账号（用户名），例如 "1"、"11"、"ren"、"username1" 等。
   * 值: 表示对应账号的密码，例如 "1"、"11"、"ren"、"password1" 等。

#### 示例

以下是几个示例账号和密码的键值对：

1. "1": "1"
   * 用户名: "1"
   * 密码: "1"
2. "username1": "password1"
   * 用户名: "username1"
   * 密码: "password1"

### diaries.json 文件分析

该 JSON 文件包含一个名为 "diaries" 的数组对象，每个数组元素表示一篇日记。每篇日记包含多个属性，如标题、ID、目的地、作者、评分、受欢迎度和内容。

#### 数据结构

{  
 "diaries": [{  
 "title": "探索林 海滩的美",  
 "ID": 1,  
 "destination": "林 海滩",  
 "author": "gp",  
 "rating": 3.1817,  
 "popularity": 72,  
 "content": ""  
 },  
 {  
 "title": "Lost in the Beauty of Nature",  
 "ID": 2,  
 "destination": "David Forest",  
 "author": "qnhk",  
 "rating": 2.0711,  
 "popularity": 12,  
 "content": ""  
 }  
 ]  
}

#### 分析

1. **顶层对象**: JSON 文件的根对象包含一个名为 "diaries" 的数组。
2. **"diaries" 数组**: 这个数组存储了多个日记对象，每个日记对象代表一篇日记，包含以下属性：
   * **title**: 日记的标题，如 "探索林 海滩的美" 和 "Lost in the Beauty of Nature"。
   * **ID**: 日记的唯一标识符，如 1 和 2。
   * **destination**: 日记的目的地，如 "林 海滩" 和 "David Forest"。
   * **author**: 日记的作者，如 "gp" 和 "qnhk"。
   * **rating**: 日记的评分，表示这篇日记的评价程度，如 3.1817 和 2.0711。
   * **popularity**: 日记的受欢迎度，表示这篇日记的流行程度，如 72 和 12。
   * **content**: 日记的内容，目前为空字符串。

## 三，模块划分

***该部分仅涉及每个模块的功能介绍，仅为总体介绍，不包含模块的具体实现，算法思想，主要函数分析，模块调用等内容，具体的模块内容介绍详见模块设计报告。***

### 3.1 登录模块

登录模块：login.cpp与login\_page.cpp

#### login.cpp 的功能分析

##### 用户登录界面初始化

* 负责初始化和设置用户登录界面，包括用户名和密码输入框、登录按钮等。

##### 用户登录逻辑处理

* 当用户点击登录按钮时，从界面获取用户输入的用户名和密码，并进行登录验证。
* 检查账户是否存在，如果存在则进一步验证密码是否正确。如果账户不存在，则创建一个新账户。

##### 错误处理与提示

* 如果用户名或密码错误，显示错误提示并清空输入框。
* 如果登录成功，显示成功提示并进入功能选择页面。

##### 界面跳转

* 登录成功后，隐藏当前登录界面，打开功能选择界面。

#### login\_page.cpp 的功能分析

##### 账户验证

* 通过检查 JSON 文件中是否存在输入的账户来验证账户是否存在。

##### 账户创建

* 如果账户不存在，通过将新账户和密码写入 JSON 文件来创建新账户。

##### 密码验证

* 验证输入的密码是否与 JSON 文件中存储的密码匹配。

##### 数据存储与管理

* 读取和写入 JSON 文件，管理用户账户信息。
* 使用 cJSON 库进行 JSON 数据的解析和生成。

#### 总结

* login.cpp 主要处理用户界面的交互和逻辑，包括获取用户输入、显示提示信息、处理错误、以及界面跳转。
* login\_page.cpp 主要处理与用户数据的交互，包括验证账户和密码、创建新账户、以及读写 JSON 文件。
* 这两个文件协同工作，共同实现了用户登录功能，包括账户验证、密码验证、账户创建和界面跳转。

### 3.2 功能选择模块

功能选择模块：functionselectpage.cpp

#### functionselectpage.cpp 的功能分析

##### 初始化和设置

* 初始化功能选择页面，并设置界面组件。
* 将用户的账户信息存储在成员变量中。

##### 功能选择确认

* 当用户点击确认按钮时，根据组合框中的选项索引来决定进入哪一个功能页面。
* 选项索引为0时，进入目的地推荐功能页面。
* 选项索引为1时，进入日记管理功能页面。
* 隐藏当前功能选择页面并显示相应的功能页面。

##### 功能页面的切换

* 在进入新功能页面时，隐藏当前功能选择页面。
* 为日记管理功能页面添加信号和槽连接，以便在退出日记管理功能时重新显示功能选择页面。

##### 退出功能

* 当用户点击退出按钮时，关闭当前页面并发出 quit\_function 信号。

#### 总结

* functionselectpage.cpp 负责初始化和设置功能选择页面，并根据用户的选择进入相应的功能页面。
* 处理用户在功能选择页面上的交互，包括确认选择和退出功能。
* 实现了页面之间的切换和信号槽机制，确保用户体验的连贯性。

### 3.3 日记搜索和浏览模块

日记搜索和浏览模块：diarymanagement.cpp

#### diarymanagement.cpp 的功能分析

##### 初始化和设置

* 初始化日记管理页面，设置界面组件。
* 加载和解析 diaries.json 文件，将日记数据存储到模型中，并在界面上显示。

##### 日记展示

* 将所有日记信息提取并格式化为字符串，存入数据模型，并在界面上显示。
* 禁止用户编辑界面上的日记信息。

##### 搜索功能

* 提供多种搜索方式，包括按地点、标题和内容搜索。
* 根据用户输入的搜索条件，从日记数据中筛选匹配的日记。

##### 排序功能

* 提供多种排序方式，包括按评分、人气和兴趣排序。
* 对匹配的日记集合进行排序，并显示前10条日记。

##### 日记管理

* 提供创建新日记和查看日记详情的功能。
* 新建日记时，跳转到日记创建页面；双击日记项时，跳转到日记详情页面。

##### 界面切换

* 在不同功能页面之间进行切换，隐藏当前页面并显示相应的功能页面。
* 处理界面返回操作，重新加载和显示日记列表。

#### 总结

* diarymanagement.cpp 主要处理日记管理的逻辑，包括日记的加载、展示、搜索和排序。
* 通过信号槽机制，实现了用户在界面上的交互操作和功能切换。
* 提供了创建和查看日记的功能，增强了用户对日记管理的操作体验。

### 3.4 日记创建模块

日记创建模块：diarycreate.cpp

#### diarycreate.cpp 的功能分析

##### 初始化和设置

* 初始化日记创建页面，设置界面组件。
* 将用户的账户信息存储在成员变量中。

##### 取消创建

* 用户点击取消按钮时，关闭当前页面并发出 quit\_function 信号，返回到上一页面。

##### 确认创建

* 用户点击确认按钮时，从界面获取新日记的标题和地点等信息。
* 读取现有的 diaries.json 文件，解析并获取已有的日记列表。
* 创建新的日记对象，设置新日记的属性（如ID、作者、评分、人气等），并将其添加到日记列表中。
* 将更新后的日记列表保存回 diaries.json 文件。

##### 解析日记内容

* 从 diaries.json 文件中读取并解析日记内容，将其转换为 Diary 对象列表。
* 检查 JSON 文件的有效性，并提取每条日记的相关信息。

##### 保存新日记

* 将新的日记对象添加到现有的 JSON 结构中。
* 将更新后的 JSON 结构写回到 diaries.json 文件中。

#### 总结

* diarycreate.cpp 主要处理日记创建的逻辑，包括获取用户输入、解析现有日记数据、创建新日记对象以及保存新日记。
* 提供取消和确认创建日记的功能，确保用户可以灵活操作。
* 通过解析和保存 JSON 数据，实现了日记数据的持久化存储。

### 3.5 日记展示模块

日记展示模块：diaryview.cpp

#### diaryview.cpp 的功能分析

##### 初始化和设置

* 初始化日记查看页面，设置界面组件。
* 根据传入的日记列表和日记ID，显示特定日记的标题、地点和内容。

##### 确认操作

* 用户点击确认按钮时，更新当前日记的评分和人气。
* 根据用户选择的评分更新日记的总评分，使用加权平均法计算新的评分。
* 增加日记的人气值。

##### 读取和解析 JSON 文件

* 打开 diaries.json 文件，以只读方式读取文件内容。
* 使用 cJSON 库解析 JSON 数据，获取包含所有日记的数组。

##### 更新 JSON 文件

* 在解析的 JSON 数据中找到对应 ID 的日记项，更新其评分和人气值。
* 将更新后的 JSON 数据写回 diaries.json 文件中。

##### 界面切换

* 在确认操作后，关闭当前日记查看页面并发出 quit\_function 信号，返回到上一页面。

#### 总结

* diaryview.cpp 主要处理日记查看和更新的逻辑，包括显示日记详情、更新评分和人气值。
* 通过读取和解析 JSON 数据，实现对日记数据的更新和持久化存储。
* 提供了用户交互界面，使用户能够查看和评分日记，并保持数据的一致性。

### 3.6 游学推荐模块

游学推荐模块：destinationrecommend.cpp

#### destinationrecommend.cpp 的功能分析

##### 初始化和设置

* 初始化推荐目的地页面，设置界面组件。
* 根据用户账户信息初始化页面。

##### 功能选择

* 根据用户在组合框中的选择，确定推荐模式或搜索模式。
* 点击确认按钮时，调用相应的模式处理函数。

##### 推荐模式

* 根据用户选择的类别，从预定义的类别列表中选择匹配的目的地。
* 依据选择的排序方式（按热度或评分），对匹配的目的地进行排序并显示前10个结果。
* 使用 kmpSearch 函数进行字符串匹配，确定符合类别的目的地。

##### 搜索模式

* 根据用户输入的关键字，搜索匹配的目的地。
* 若找到匹配的目的地，则根据用户选择的排序方式进行排序并显示结果。
* 若未找到匹配项，则显示错误提示信息。

##### 排序功能

* 提供快速排序算法，对目的地按照热度或评分进行排序。
* 实现了部分排序功能，只显示前10个匹配结果。
* 使用分区函数实现快速排序的核心逻辑。

##### 界面切换

* 在双击列表项时，打开地图页面显示选定的目的地详情，并隐藏当前页面。
* 实现信号槽机制，支持页面之间的切换和返回操作。

#### 总结

* destinationrecommend.cpp 主要处理目的地推荐和搜索的逻辑，包括用户输入处理、目的地匹配、排序和显示。
* 通过用户界面提供交互功能，使用户能够根据不同的类别和关键字搜索和推荐目的地。
* 实现了目的地信息的显示和页面切换，提高了用户体验的流畅性和便捷性。

### 3.7 美食推荐/设施查询/路线规划模块

美食推荐/设施查询/路线规划模块：mymap.cpp

#### mymap.cpp 的功能分析

##### 初始化和设置

* **初始化地图页面**：在地图页面初始化时，设置界面组件，并加载传入的景点信息和索引。
* **填充位置和目的地选项**：根据景点的图数据，将景点的所有位置和目的地加载到当前页面的下拉列表中。
* **加载交通工具选项**：根据景点的属性（是否为学校），加载不同的交通工具选项，如自行车或电瓶车。
* **加载食物种类**：根据景点的餐饮信息，动态加载不同的食物种类选项到界面的下拉列表中。
* **设置餐馆选项**：将景点中的所有餐馆信息加载到页面的餐馆下拉列表中。

##### 返回操作

* **返回按钮**：提供返回按钮功能，允许用户关闭当前地图页面并返回到上一页面。点击返回按钮时，发出 quit\_function 信号。

##### 路线规划

* **单点到单点路线规划**：提供单点到单点的路线规划功能，根据用户选择的当前点和目的地，计算并显示最优路线。
  + 根据用户选择的交通工具（自行车或电瓶车）和路线类别（距离最短或时间最短），选择合适的算法（如Dijkstra算法）进行路线规划。
  + 将规划好的路线显示在地图上。
* **多点路线规划**：提供多点路线规划功能，允许用户选择多个途经点，并根据选择的途经点进行最优路线规划。
  + 用户可以通过勾选多点路线选项来启用多点路线规划功能。
  + 根据选择的途经点，使用Floyd算法计算最优路线，并在地图上显示。

##### 路线编辑

* **编辑途经点**：提供编辑路线功能，允许用户选择多个途经点并编辑其顺序。
  + 用户可以在编辑途经点对话框中选择和调整途经点的顺序。
  + 编辑完成后，将选择的途经点显示在页面的途经点列表中。

##### 设施搜索

* **设施搜索功能**：用户可以输入设施名称进行搜索，并根据搜索结果显示符合条件的设施列表。
  + 根据用户输入的关键字，使用KMP算法在景点的图数据中查找匹配的设施名称。
  + 将符合条件的设施按距离排序，并显示在搜索结果列表中。
* **不同类别的设施搜索**：提供多种设施类别选项，用户可以选择特定类别的设施进行搜索。
  + 根据选择的设施类别（如体育设施、餐饮设施、厕所等），进一步筛选和排序搜索结果。

##### 快速排序

* **快速排序算法**：实现了快速排序算法，用于对搜索结果进行排序。
  + 对搜索结果按照不同维度（如距离、评分和人气）进行排序，确保搜索结果按用户需求显示。

##### 精确匹配

* **精确匹配功能**：提供精确匹配功能，用于查找与用户输入完全匹配的字符串。
  + 逐字符比较用户输入和目标字符串，确保精确匹配。

##### 人气和距离筛选

* **人气筛选**：筛选出前10个人气最高的餐饮或设施选项，并按人气排序显示。
  + 使用快速排序和部分排序算法，确保只筛选出前10个结果，提高筛选效率。
* **距离筛选**：筛选出前10个距离最近的餐饮或设施选项，并按距离排序显示。
  + 通过Dijkstra算法计算各个设施与当前点的距离，并按距离排序显示结果。

#### 总结

* **地图和路线规划**：提供丰富的路线规划功能，用户可以选择单点到单点或多点路线，并根据交通工具和路线类别进行规划和显示。
* **设施和餐饮搜索**：提供多种搜索和筛选功能，用户可以根据关键字、设施类别和食物种类查找目标设施和餐饮，并按人气和距离排序。
* **用户交互**：通过动态加载和显示用户选项，提高用户体验的流畅性和便捷性。用户可以根据自己的需求灵活选择和规划路线，查找所需的设施和餐饮。

### 3.8 路线规划逻辑模块

路线规划逻辑模块：route\_plan.cpp

#### route\_plan.cpp 的功能分析

##### 多点路线规划

* **输入用户位置和目的地**：用户输入当前位置和希望访问的多个目的地的ID。
* **选择路径类型**：用户选择计算最短距离或最短时间的路径。
* **路径计算**：根据用户选择的路径类型，使用Floyd-Warshall算法计算最短路径，并考虑是否使用交通工具（如自行车或电瓶车）。
* **路径展示**：展示计算出的最短路径及其距离或时间。

##### 两点路线规划

* **输入起点和终点**：用户输入起点和终点的ID。
* **选择路径类型**：用户选择计算最短距离或最短时间的路径。
* **路径计算**：根据用户选择的路径类型，使用Dijkstra算法计算最短路径，并考虑是否使用交通工具（如自行车或电瓶车）。
* **路径展示**：展示计算出的最短路径及其距离或时间。
* **路径详情**：展示所有路径点和中间点的坐标信息。

##### Floyd-Warshall算法

* **初始化权重矩阵**：初始化距离和前驱节点矩阵。
* **计算最短路径**：通过Floyd-Warshall算法计算所有节点对之间的最短路径。
* **路径展示**：在计算多点路径时，展示最短路径及其总距离或时间。

##### Dijkstra算法

* **初始化**：初始化距离、时间和前驱节点数组。
* **计算最短路径**：通过Dijkstra算法计算从起点到所有其他节点的最短路径。
* **更新邻居节点**：更新当前节点的邻居节点的距离或时间。
* **路径展示**：在计算两点路径时，展示最短路径及其总距离或时间。

##### 设施列表

* **展示所有设施**：展示图中所有设施的ID和名称，供用户选择和参考。

##### 路径打印

* **打印路径**：根据计算出的前驱节点，递归打印从起点到终点的完整路径。
* **打印中间点**：展示路径中所有点和中间点的坐标信息。

#### 总结

* **多点路线规划**：用户输入多个目的地，通过Floyd-Warshall算法计算最短路径，并展示路径详情。
* **两点路线规划**：用户输入起点和终点，通过Dijkstra算法计算最短路径，并展示路径详情。
* **选择交通工具**：用户可以选择不同的交通工具，如自行车或电瓶车，影响路径计算的权重。
* **路径和设施展示**：详细展示计算出的路径和图中的所有设施，帮助用户进行路线规划和选择。
* **用户交互**：通过输入起点、终点和目的地，选择路径类型和交通工具，实现交互式的路线规划和路径展示。

### 3.9 路线绘制模块

路线绘制模块：mylable.cpp

#### mylabel.cpp 的功能分析

##### 初始化和设置

* **初始化标签**：在标签初始化时，加载并显示背景地图图片。
* **设置标签属性**：设置标签的几何属性，使其匹配地图图片的大小。

##### 路线绘制事件

* **绘制事件处理**：在标签需要重绘时，根据当前状态决定绘制单点路线或多点路线。
* **更新路线绘制**：当需要更新路线绘制时，调用 update 方法触发重新绘制。

##### 单点路线绘制

* **加载路线数据**：根据传入的起点和终点，从景点图数据中获取路线信息。
* **递归绘制路线**：递归获取从起点到终点的路径，并绘制每段路径。
* **绘制路线**：根据路径坐标，使用绘图工具绘制路线，并在每段路线的终点绘制箭头。
* **处理中间点**：如果路径中存在中间点，绘制从起点到中间点，再绘制从中间点到终点的两段路线。

##### 多点路线绘制

* **加载多点路径数据**：从景点图数据中获取多点路径信息。
* **迭代绘制多点路径**：迭代获取每两个相邻路径点，并绘制每段路径。
* **绘制多点路径**：根据路径坐标，使用绘图工具绘制多段路径，并在每段路径的终点绘制箭头。
* **绘制序号**：在中间点和终点上绘制序号，标记路径点的顺序。

##### 路线更新

* **更新路线信息**：接收新的景点数据、绘制标志、起点和终点信息，触发更新绘制事件。
* **绘制触发**：根据新的路线数据和绘制标志，调用 update 方法触发重绘。

#### 总结

* **地图初始化**：加载并显示背景地图图片，设置标签的大小和位置。
* **绘制单点路线**：递归获取路径节点，绘制起点到终点的路线，并在终点绘制箭头。
* **绘制多点路线**：迭代获取多点路径节点，绘制多个路径段，并在每段路径的终点绘制箭头和序号。
* **动态更新**：根据新的路线数据和绘制标志，动态更新并重绘路线图，提高用户交互体验。

### 3.10 多点路线规划的编辑模块

多点路线规划的编辑模块：editmultipoint.cpp

#### editmultipoint.cpp 的功能分析

##### 初始化和设置

* **初始化多点编辑对话框**：在对话框初始化时，设置界面组件，并将传入的路径点加载到界面的下拉列表中。
* **设置路径点选项**：将所有路径点加载到界面上的多个下拉列表中，用户可以从这些列表中选择途经点。

##### 确认操作

* **确认途经点**：用户点击确认按钮后，收集用户选择的途经点，并存储到路径数组中。
* **触发确认信号**：确认按钮被点击后，发出 accept 信号，表示用户已确认选择的途经点。

#### 总结

* **多点编辑初始化**：加载传入的路径点，并将其填充到界面的下拉列表中，供用户选择。
* **用户选择途经点**：用户可以从多个下拉列表中选择途经点，形成一个多点路径。
* **确认途经点选择**：用户点击确认按钮后，收集用户选择的途经点，存储并触发确认信号，返回上一页面。

### 3.11 读取地图文件和地图构建模块

读取地图文件和地图构建模块：myGraph.cpp

#### myGraph.cpp 的功能分析

##### 景点数据加载

* **读取JSON文件**：从指定文件路径读取JSON数据。
* **解析景点数据**：使用cJSON库解析JSON数据，提取景点的基本信息（如名称、是否为学校、受欢迎程度和评分）。
* **加载景点图数据**：解析每个景点的建筑信息，包括建筑名称、设施类型和坐标，初始化图的顶点。
* **加载路径数据**：解析景点之间的路径信息，包括起点、终点、距离、拥堵程度和中点坐标，构建图的边。

##### 景点数据保存

* **读取旧JSON文件**：从指定文件路径读取旧的JSON数据，用于更新景点信息。
* **创建新的JSON对象**：创建新的JSON对象数组，用于保存更新后的景点信息。
* **保存景点数据**：遍历景点数组，将每个景点的基本信息、建筑信息、路径信息和餐饮信息添加到新的JSON对象中。
* **写入新JSON文件**：将新的JSON对象数据写入指定文件路径，覆盖旧的景点数据。

##### 餐饮信息加载

* **解析餐饮信息**：从JSON数据中提取餐饮信息，包括餐饮名称、餐馆ID、菜系和受欢迎程度。
* **加载餐饮信息**：将解析出的餐饮信息加载到景点的餐饮数组中。

#### 总结

* **景点数据加载和解析**：从JSON文件中读取和解析景点的基本信息、图结构和餐饮信息，构建景点数组和图的结构。
* **路径数据构建**：解析景点之间的路径信息，构建图的边，并设置路径的相关属性（如距离和拥堵程度）。
* **数据持久化**：将更新后的景点信息保存到新的JSON文件中，实现景点数据的持久化存储。
* **用户交互**：通过加载和保存景点数据，实现用户对景点信息的查询、更新和持久化操作，提高数据管理的便捷性和可靠性。

### 3.12 哈夫曼编码模块

哈夫曼编码模块：huffman.cpp

#### huffman.cpp 的功能分析

##### 哈夫曼树的生成

* **创建叶子节点**：根据字符频率表创建叶子节点，并将它们存入节点数组中。
* **构建哈夫曼树**：通过合并频率最小的两个节点，逐步构建哈夫曼树，直到只剩下一个根节点。

##### 哈夫曼编码生成

* **生成编码表**：遍历哈夫曼树，生成每个字符对应的哈夫曼编码，并存入编码表中。

##### 字符串编码和解码

* **字符串编码**：使用哈夫曼编码表将输入字符串转换为哈夫曼编码的二进制字符串。
* **字符串解码**：根据哈夫曼编码表将二进制字符串解码为原始字符串。

##### 编码表的保存和加载

* **保存编码表到JSON文件**：将哈夫曼编码表保存为JSON格式，并写入指定文件。
* **从JSON文件加载编码表**：从JSON文件中读取哈夫曼编码表，并将其加载到内存中。

##### 字符串压缩和解压缩

* **字符串压缩**：对输入字符串进行哈夫曼编码，并将编码表和编码后的二进制数据分别保存到文件中。
* **字符串解压缩**：从文件中加载哈夫曼编码表和编码后的二进制数据，解码后返回原始字符串。

##### 二进制文件读写

* **写入二进制文件**：将编码后的二进制数据写入文件，包括数据的实际有效位数和填充的位数。
* **读取二进制文件**：从文件中读取二进制数据，并转换为二进制字符串。

#### 总结

* **哈夫曼树构建**：根据字符频率构建哈夫曼树，生成哈夫曼编码表。
* **字符串编码和解码**：使用哈夫曼编码表进行字符串的编码和解码。
* **编码表的持久化**：将哈夫曼编码表保存到JSON文件中，并能从JSON文件中加载编码表。
* **字符串压缩和解压缩**：将字符串压缩为二进制文件，并能从二进制文件中解压缩还原原始字符串。
* **二进制文件处理**：处理哈夫曼编码后的二进制数据的读写操作，实现高效的数据存储和读取。

### 3.13 kmp字符匹配模块

kmp字符匹配模块：kmp.cpp

#### kmp.cpp 的功能分析

##### 构建部分匹配表

* **生成LPS数组**：构建部分匹配表（LPS数组），用于存储每个子串的最长前缀后缀长度，以优化模式串的匹配过程。
* **初始化LPS数组**：初始化LPS数组的值为0。
* **计算LPS值**：通过遍历模式串，计算每个字符对应的LPS值，根据前一个字符的LPS值进行递推。

##### KMP字符串匹配

* **初始化指针**：初始化文本和模式串的指针，用于遍历和比较字符。
* **匹配字符**：逐字符比较文本和模式串，如果字符匹配，移动指针；如果字符不匹配，根据LPS值调整模式串指针，继续匹配。
* **匹配成功**：当模式串指针达到模式串末尾时，表示匹配成功，返回true。
* **匹配失败**：如果遍历完文本仍未匹配成功，返回false。

##### 模式匹配

* **多模式匹配**：接受一个模式串数组和一个文本，依次对每个模式串使用KMP算法进行匹配。
* **返回匹配结果**：如果任意一个模式串匹配成功，返回true；如果所有模式串都未匹配成功，返回false。

#### 总结

* **构建部分匹配表**：通过计算模式串的最长前缀后缀长度，生成LPS数组，以优化匹配过程。
* **KMP字符串匹配**：使用KMP算法进行字符串匹配，通过部分匹配表优化字符比较过程，实现高效匹配。
* **多模式匹配**：对多个模式串进行匹配，如果任意一个模式串匹配成功，则返回成功结果。

### 3.14 程序进入模块

程序进入模块：main.cpp

#### main.cpp 的功能分析

##### 初始化和加载数据

* **加载景点数据**：在程序启动时，从 data\_map.json 文件中加载景点数据。调用 loadAttractionArray 函数，解析 JSON 文件，并将景点信息存储到程序内存中。

##### Qt 应用程序启动

* **创建应用实例**：通过 QApplication 类创建 Qt 应用程序实例 a，这将初始化应用程序的环境和资源。
* **创建登录窗口**：实例化 login 类对象 w，用于显示登录界面。
* **显示登录窗口**：调用 w.show() 方法，将登录窗口显示在屏幕上，等待用户输入和操作。

##### 程序主循环

* **运行主事件循环**：调用 a.exec() 方法，启动 Qt 应用程序的主事件循环。该循环将处理用户输入事件、窗口更新事件等，直到程序退出。

#### 总结

* **数据加载**：从 JSON 文件中加载景点数据，并在内存中初始化相关数据结构。
* **应用程序初始化**：创建并初始化 Qt 应用程序实例，设置应用程序的环境和资源。
* **界面展示**：创建并显示登录窗口，用户可以通过该窗口进行登录操作。
* **事件循环**：启动应用程序的主事件循环，处理用户输入和窗口更新等事件，直到程序退出。