**北京科技大学实验报告**

学院：计算机与通信工程 专业：通信工程 班级：

姓名： 学号： 实验日期：2024 年 6 月 5 日

**实验名称：哈利波特书籍检索-界面要求**

**实验目的：**

实现书籍检索功能,并使用QT制作图形界面

1. 罗列查询结果。支持输入人名/地名，显示查询结果。
2. 查询结果多条，对多条记录编号，采用类似列表形式显示；
3. 选择序号，或者单击查询记录行，能够显示到指定位置人名/地名前后的一段文字。如选择序号1，应显示所在段落、上一个段落和下一个段落的文字内容

**实验仪器：**

计算机：台式电脑

CPU：12th Gen Intel(R) Core(TM) i5-12400F 2.50 GHz

内存：16GB DDR4 2667MHz

硬盘：SSD GM7 M.2 2TB

显卡：NVIDIA GeForce RTX 4060 8GB

操作系统：Windows 11 专业版

编译器：MinGW 11.2.0 64-bit for C++

集成开发环境：Clion（2024.1.3） Qt（6.6.2）

**实验原理：**

总流程：

图示

描述已自动生成

分析文本流程：

图示

描述已自动生成

↑写完核心代码之后，用Qt做图形页面

1. 绘制页面：

图形用户界面, 文本

描述已自动生成

1. 从多个文本文件中读取内容并存储到一个全局向量中
2. 点击按钮获取左侧输入框的内容，在所有书籍内容中搜索指定的人名或地名，并记录出现的位置。
3. 在tableWidget表格中显示搜索结果，包括页码、章节、书名等信息。
4. 双击tableWidget表格中的某一行，在最下面的框中显示该行对应的文本内容

**实验内容与步骤：（字体：中文小四，英文Times New Roman小四，均不加粗）**

1. main.cpp 文件，用于创建窗口对象，加载窗口

|  |
| --- |
| 1. #include *"mainwindow.h"* 2. #include *<QApplication>* 3. int main(int argc, char \*argv[]) 4. { 5. QApplication a(argc, argv); 6. MainWindow w; 7. w.show(); 8. **return** a.exec(); 9. } |

1. mainwindow.ui 文件，用于描述图形页面样式

图形用户界面, 应用程序

描述已自动生成

1. mainwindow.h 文件，用于定义主窗口类。

|  |
| --- |
| 1. #ifndef MAINWINDOW\_H 2. #define MAINWINDOW\_H 3. #include *<QMainWindow>* 4. QT\_BEGIN\_NAMESPACE 5. namespace Ui { 6. class MainWindow; 7. } 8. QT\_END\_NAMESPACE 9. class MainWindow : public QMainWindow 10. { 11. Q\_OBJECT 12. public: 13. MainWindow(QWidget \*parent = nullptr); 14. ~MainWindow(); 15. private slots: 16. void on\_pushButton\_clicked(); 17. void on\_tableWidget\_cellDoubleClicked(int row, int column); 18. private: 19. Ui::MainWindow \*ui; 20. **static** MainWindow \*my; 21. }; 22. #endif *// MAINWINDOW\_H* |

主窗口类中的slots函数响应按钮按下，和tableWight中点击链接的操作。

1. mainwindow.cpp 文件，用于实现mainwindow.h中的相关函数，创建Text类

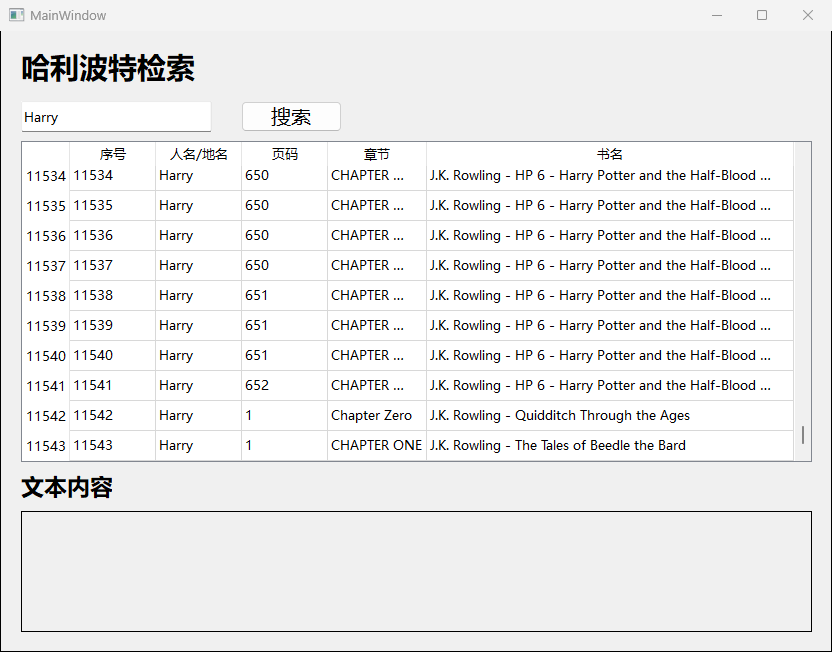
|  |
| --- |
| 1. #include "mainwindow.h" 2. #include "./ui\_mainwindow.h" 3. #include <fstream> 4. #include <vector> 5. #include <sstream> 6. #include <iostream> 7. #include <QMessageBox> 8. #include <QDebug> 9. #include <string> 10. **using** namespace std; 11. // **Text** 类，用于存储每个文本条目信息 12. **class** **Text** 13. { 14. **public**: 15. **int** page; // 页码 16. string chapter; // 章节 17. string book; // 书名 18. string content; // 内容 19. }; 20. // 函数声明 21. void readBooks(const vector<string>& filenames); 22. void searchNameInBooks(string info, vector<**int**> & s); 23. void displayResults(vector<**int**> s, string info); 24. // 全局变量 25. vector<**Text**> **text**; // 用于存储所有文本条目 26. **Text** TempContent; // 临时 **Text** 对象，用于读取文件内容 27. vector<**int**> search\_Info; // 用于存储搜索结果 28. MainWindow \*MainWindow::my = nullptr; // 指向 MainWindow 实例的指针 29. // 哈利波特系列书籍文件名 30. vector<string> filenames = { 31. "HP2--Harry\_Potter\_and\_the\_Chamber\_of\_Secrets\_Book\_2\_", 32. "HP7--Harry\_Potter\_and\_the\_Deathly\_Hallows\_Book\_7\_", 33. "J.K. Rowling - HP 0 - Harry Potter Prequel", 34. "J.K. Rowling - HP 3 - Harry Potter and the Prisoner of Azkaban", 35. "J.K. Rowling - HP 4 - Harry Potter and the Goblet of Fire", 36. "J.K. Rowling - HP 6 - Harry Potter and the Half-Blood Prince", 37. "J.K. Rowling - Quidditch Through the Ages", 38. "J.K. Rowling - The Tales of Beedle the Bard" 39. }; 40. // 构造函数 41. MainWindow::MainWindow(QWidget \*parent) 42. : QMainWindow(parent) 43. , ui(**new** Ui::MainWindow) 44. { 45. ui->setupUi(this); 46. my = this; // 将当前 MainWindow 实例指针赋值给 my 47. // 读取文本文件内容 48. readBooks(filenames); 49. // ui->tableWidget->setColumnWidth(0,20); 50. // ui->tableWidget->setColumnWidth(1,20); 51. // ui->tableWidget->setColumnWidth(2,20); 52. // ui->tableWidget->setColumnWidth(3,20); 53. // ui->tableWidget->setColumnWidth(4,20); 54. ui->tableWidget->setColumnCount(5); // 设置表格列数为5 55. ui->tableWidget->setHorizontalHeaderLabels(QStringList() << "序号"<<"人名/地名"<<"页码"<<"章节"<<"书名"); // 设置表格头 56. // 设置列内容自适应宽度 57. ui->tableWidget->horizontalHeader()->setSectionResizeMode(QHeaderView::Stretch); 58. } 59. // 析构函数 60. MainWindow::~MainWindow() 61. { 62. **delete** ui; 63. } 64. // 读取文件，将内容存储到 **text** 向量中 65. void readBooks(const vector<string>& filenames) { 66. **for** (string filename : filenames) { 67. TempContent.book = filename; // 假设文件名即为书名 68. TempContent.page = 1; 69. TempContent.chapter = "Chapter Zero"; 70. ifstream **file**; // 创建文件输入流对象 71. **file**.**open**(filename+".txt"); // 打开文件 72. **if** (!**file**.is\_open()) { // 如果文件无法打开 73. qDebug()<<"无法打开文件"+filename+"/n"; // 输出调试信息 74. **continue**; // 跳过该文件 75. } 76. **int** page\_num; 77. string line; 78. bool isChapter\_begin = **true**; 79. **while** (getline(**file**, line)) { // 按行读取文件内容 80. **if** (line.substr(0,7) == "Chapter" || line.substr(0,7) == "CHAPTER") { // 如果是章节行 81. TempContent.chapter = line; // 更新章节信息 82. } 83. **else** **if** ('0' < line[0] && line[0] <= '9' && line.**size**() < 4){ // 如果是页码行 84. istringstream ss(line); // 创建字符串流对象 85. ss >> page\_num; // 读取页码 86. TempContent.page = page\_num + 1; // 更新页码信息 87. } 88. **else** { // 如果是内容行 89. TempContent.content = line; // 更新内容信息 90. **text**.push\_back(TempContent); // 将内容存入 **text** 向量中 91. } 92. } 93. **file**.**close**(); // 关闭文件 94. } 95. } 96. // 搜索人名/地名在文本中的出现位置并记录相关信息 97. void searchNameInBooks(string info, vector<**int**> & s) //查找信息 98. { 99. **int** all\_len = **text**.**size**(); // 获取 **text** 向量的大小 100. string::size\_type p; 101. **for** (**int** i = 0; i < all\_len; i++) 102. { 103. p = **text**[i].content.find(info); // 在内容中查找信息 104. **while** (!(p == string::npos)){ // 如果找到信息 105. s.push\_back(i); // 记录位置 106. p = **text**[i].content.find(info,p+1); // 继续查找下一个出现位置 107. } 108. } 109. } 110. // 搜索按钮点击事件处理函数 111. void MainWindow::on\_pushButton\_clicked() 112. { 113. string info = ui->lineEdit\_info->**text**().toStdString(); // 获取输入框中的搜索信息 114. searchNameInBooks(info, search\_Info); // 调用搜索函数 115. **if** (!search\_Info.empty()){ // 如果找到信息 116. ui->tableWidget->setRowCount(search\_Info.**size**()); // 设置表格行数 117. ui->tableWidget->horizontalHeader()->setSectionResizeMode(3, QHeaderView::ResizeToContents); // 设置列宽自适应内容 118. ui->tableWidget->horizontalHeader()->setSectionResizeMode(4, QHeaderView::ResizeToContents); // 设置列宽自适应内容 119. **for** (**int** i = 0; i < search\_Info.**size**(); i++){ 120. ui->tableWidget->setItem(i,0,**new** QTableWidgetItem(QString::number(i+1))); // 设置序号 121. ui->tableWidget->setItem(i,1,**new** QTableWidgetItem(QString::fromStdString(info))); // 设置人名/地名 122. ui->tableWidget->setItem(i,2,**new** QTableWidgetItem(QString::number(**text**[search\_Info[i]].page))); // 设置页码 123. ui->tableWidget->setItem(i,3,**new** QTableWidgetItem(QString::fromStdString(**text**[search\_Info[i]].chapter))); // 设置章节 124. ui->tableWidget->setItem(i,4,**new** QTableWidgetItem(QString::fromStdString(**text**[search\_Info[i]].book))); // 设置书名 125. } 126. } 127. **else** { // 如果找不到信息 128. QMessageBox::warning(my,"警告","找不到呢>\_<"); // 显示警告对话框 129. } 130. } 131. // 表格单元格双击事件处理函数 132. void MainWindow::on\_tableWidget\_cellDoubleClicked(**int** **row**, **int** **column**) 133. { 134. ui->label\_content->setText(QString::fromStdString(**text**[search\_Info[row]-1].content + **text**[search\_Info[row]].content)); // 显示前后两行内容 135. } |

* 1. readBooks(const vector<string>& filenames) 用于读取文件
  2. searchNameInBooks(string info, vector<int> & s) 用于搜索人名/地名在文本中的出现位置并记录相关信息
  3. MainWindow::on\_pushButton\_clicked() 用来处理点击查找按钮。
  4. MainWindow::on\_tableWidget\_cellDoubleClicked(int row, int column) 用于处理tableWidget的双击事件。

**实验结果与分析：每个步骤有运行截图和分析**

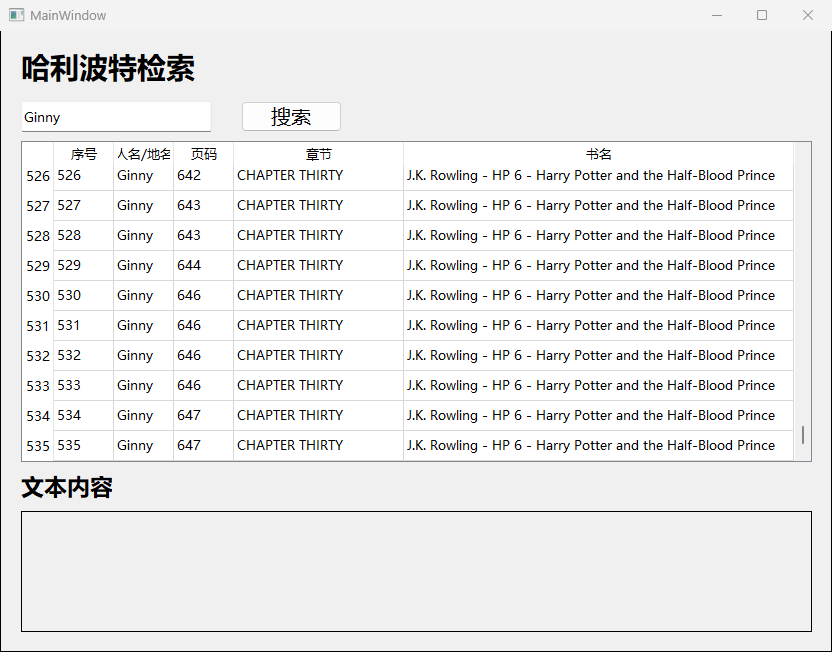
测试用例1：在输入人名地名处输入“Harry”，点击查询。

运行截图1：显示查询到的11543个结果。



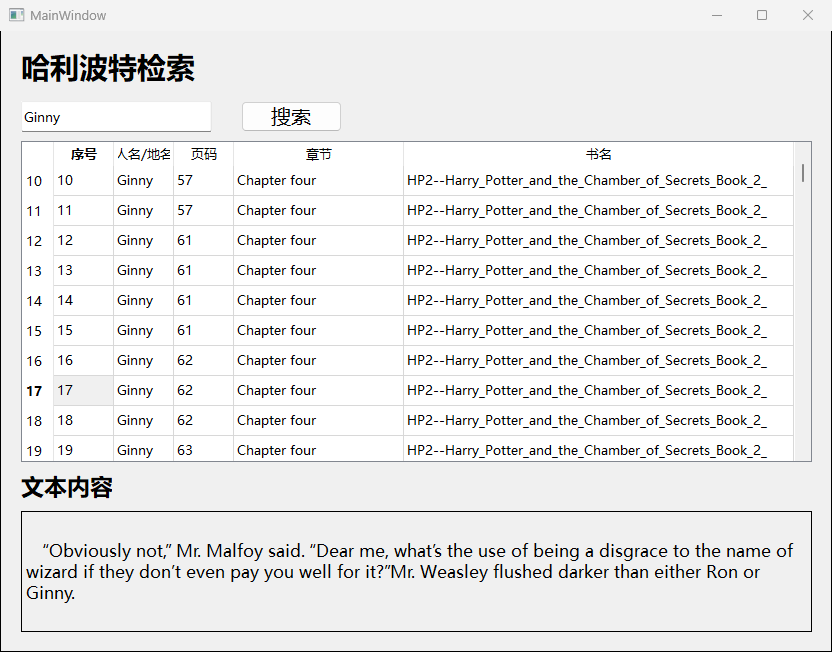
测试用例2：在输入人名地名处输入“Ginny”，点击查询。

运行截图2：显示查询到的535个结果。



测试用例3：在输入人名地名处输入“Ginny”，点击查询，点击序号17。

运行截图3：显示那一句话。



测试用例4：在输入人名地名处输入“Andy”，点击查询。

运行截图4：显示查找不到。

图形用户界面, 应用程序

描述已自动生成

**实验结论与感想**

学到了很多有关QT的知识