**北京科技大学实验报告**

学院： 计算机与通信工程学院 专业： 物联网工程 班级：

姓名： 学号： 实验日期： 2023 年 5 月 5 日

**实验名称：哈利波特检索**

**实验目的：**

1. 将哈利波特的7本书（txt格式）读入。

2. 在指定了人名/地名后，显示查询结果。  
3. 选择指定查询结果序号（选择查询内容），能够显示指定查询结果所在位置前后的一段文字。

**实验仪器：**

计算机：MacBook Pro 2018

CPU：2.2GHz IntelCore i7 8550H

内存：16GB 2133MHz LPDDR4

硬盘：256GB

显卡：Intel Iris Plus Graphics 650 1536 MB

Radeon Pro 555X 4GB

操作系统：macOS 13.3.1 (a)

编译器：Apple clang version 14.0.3 (clang-1403.0.22.14.1)

集成开发环境：Clion 2023.1.1, Qt 5.15.2

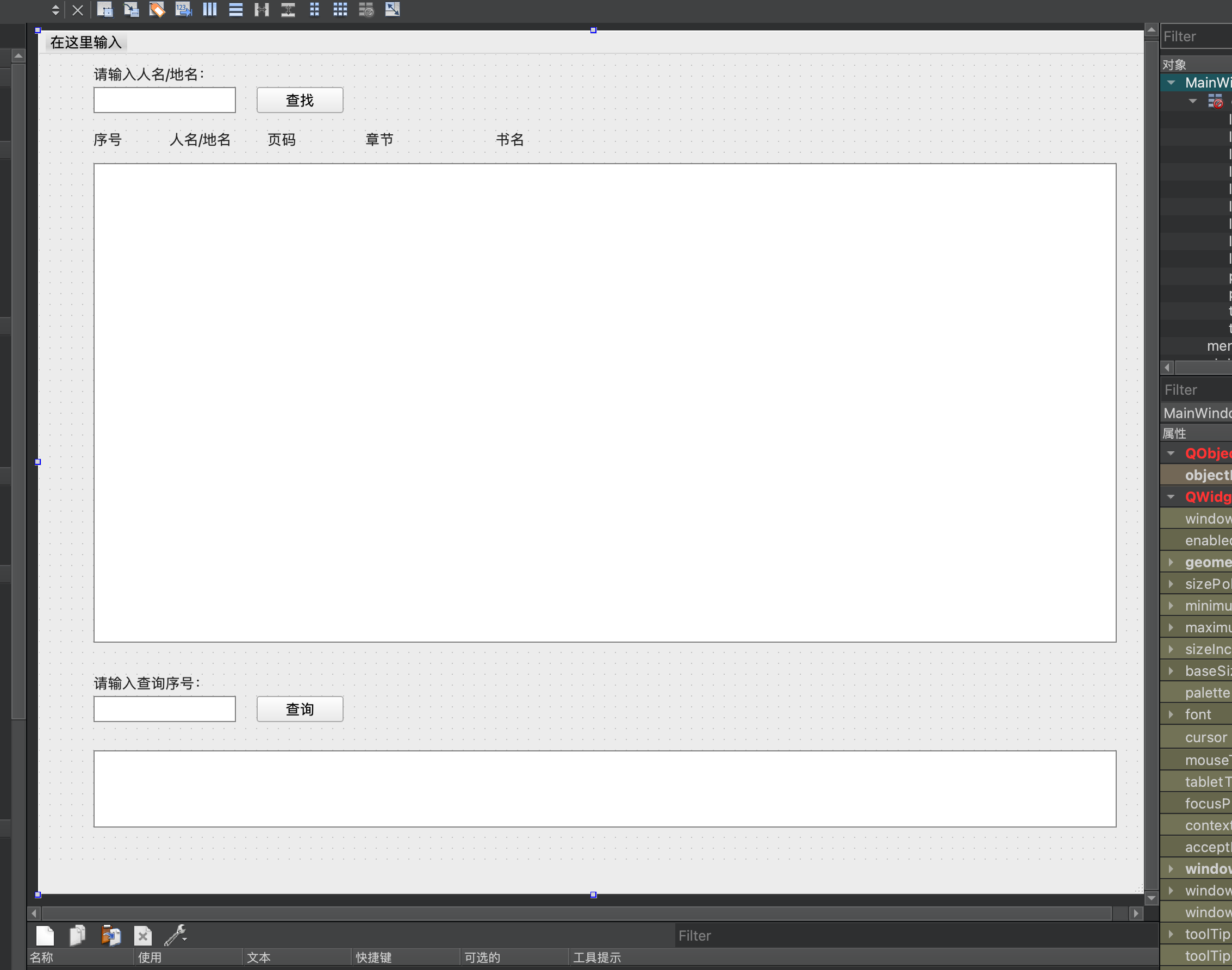
**实验原理：**

先尝试读取一本书，然后读入要查询的name变量。

1. 通过while循环，不断从文件对象中读取每一行的内容到一个字符串数组中。
2. 判断这一行的内容，如果开头是CHAPTER（注意避免开头的utf-8标识），则记录章节的变量+1，如果这一整行都是数字，则记录页码的变量+1，并且continue。
3. 再在该字符串数组中循环，每一位与name做匹配，如果匹配成功，则输出该行内容。
4. 关闭当前文件。

↑写完核心代码之后，用Qt做图形页面。

1. 绘制页面：



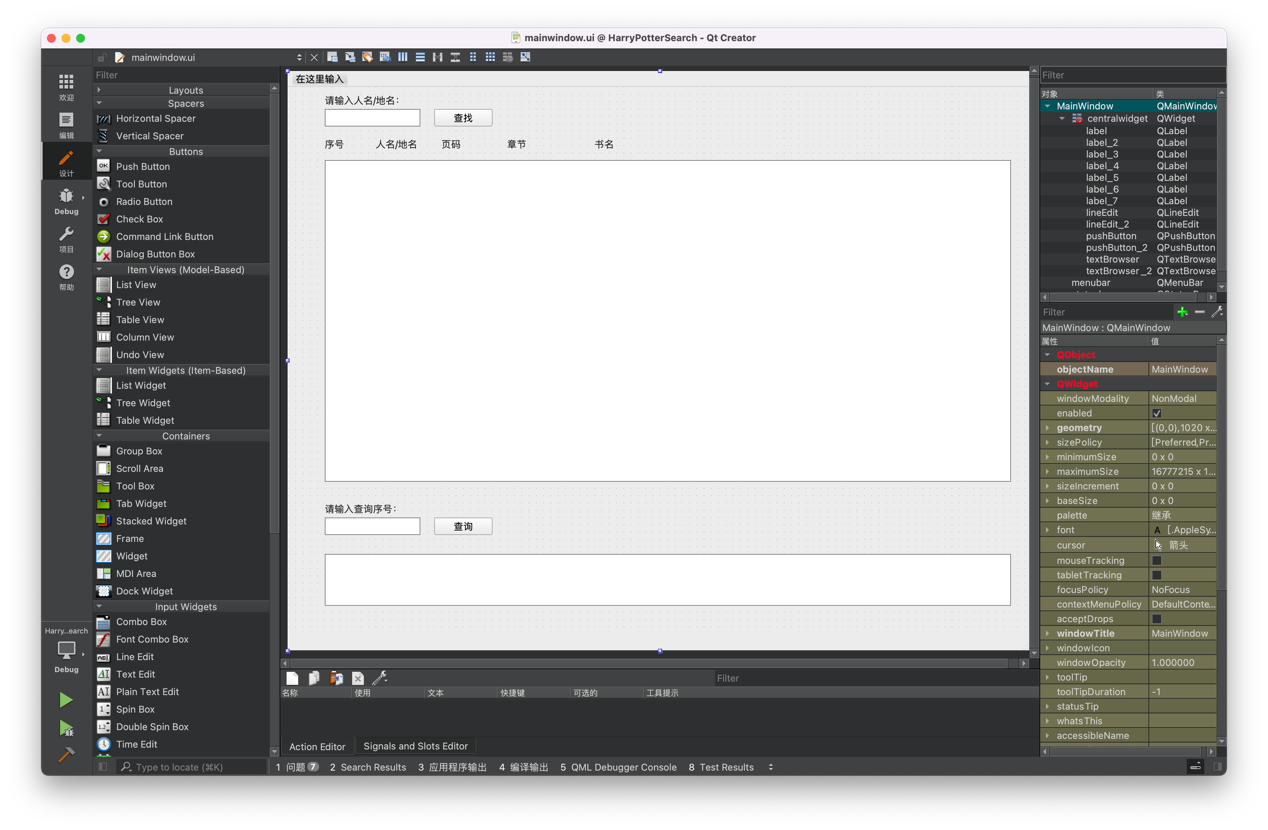
1. 把查询一本书，改为循环依次查询所有的书。
2. 点击按钮获取左侧输入框的内容，执行上一步写的核心代码。
3. 创建了一个类用于存储找到的内容，把核心代码中的输出，改为存储到这个类的vector对象中。
4. 在中间的textBrowser中输出内容。
5. 点击textBrowser中的每行的序号的链接，后台获取到链接的内容，可以直接进行查找，并将查询的结果的那一句话，显示在最下面的框中。
6. 查询按钮同上。

**实验内容与步骤：**

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



1. mainwindow.h 文件，用于定义主窗口类和存储查询信息的类。

|  |
| --- |
| 1. #ifndef MAINWINDOW\_H 2. #define MAINWINDOW\_H 3. #include <QMainWindow> 4. #include "string" 5. *using* std::string; 6. QT\_BEGIN\_NAMESPACE 7. *namespace* **Ui** { *class* **MainWindow**; } 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\_pushButton\_2\_clicked**(); 18. void **on\_textBrowser\_anchorClicked**(*const* QUrl &arg1); 19. *private*: 20. Ui::MainWindow \*ui; 21. }; 22. *class* **SearchInfo** { 23. *private*: 24. int bookNum; *//书名的序号* 25. int page; *//页数* 26. int chapter; *//章节* 27. string line; *//当前行的内容* 28. *public*: 29. **SearchInfo**(int bookNum, int page, int chapter, string line); 30. ~**SearchInfo**(); 31. string **getLine**(); 32. }; 33. #endif *//* *MAINWINDOW\_H* |

* 1. 主窗口类中的slots函数分别相应两个按钮按下，和textBrowser中点击链接的操作。
  2. SearchInfo类，用于存储查询到的数据。

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

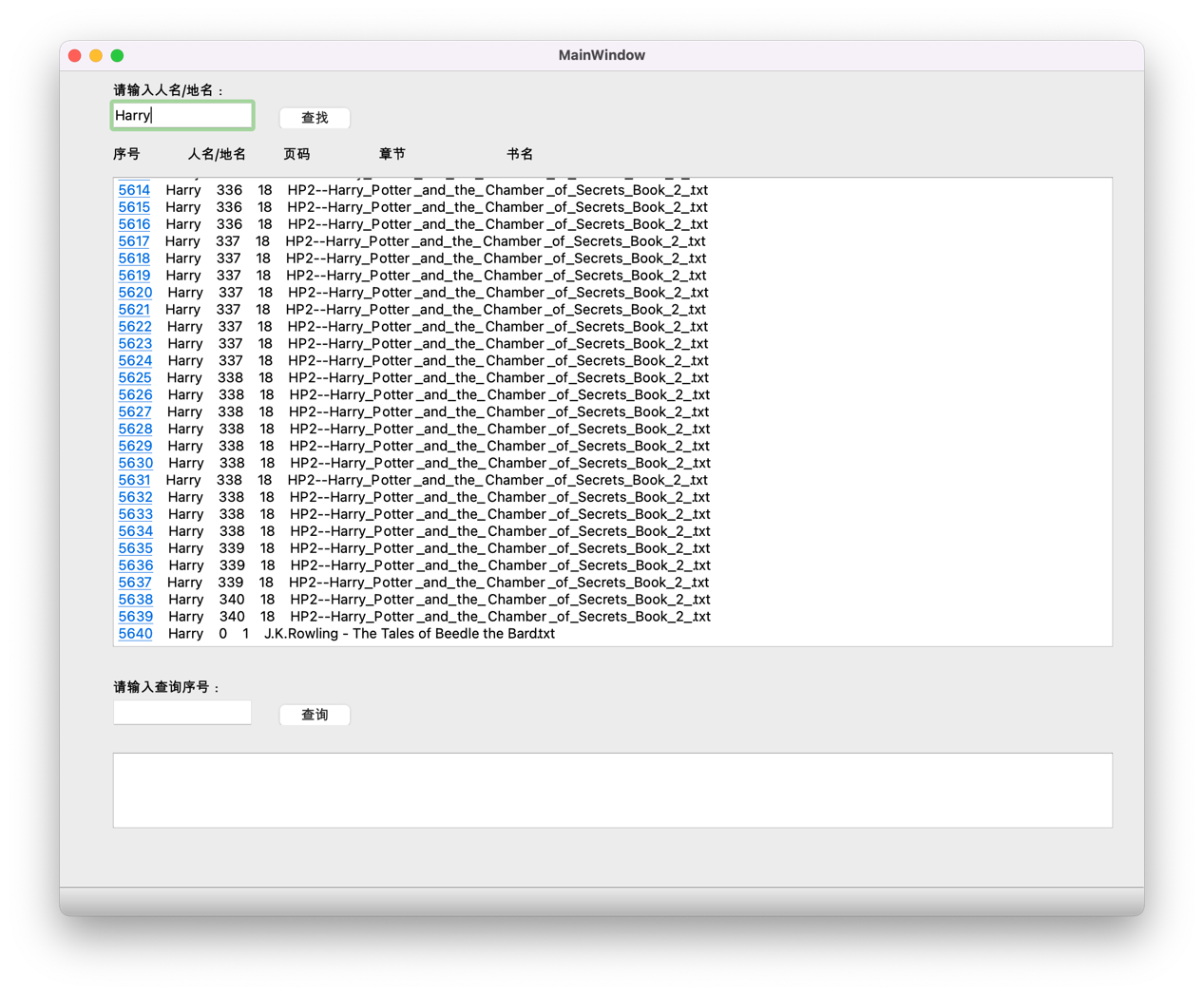
|  |
| --- |
| 1. #include "mainwindow.h" 2. #include "ui\_mainwindow.h" 3. #include "iostream" 4. #include "fstream" 5. #include "cstdio" 6. #include "QString" 7. #include "string" 8. #include "vector" 9. #include "QMessageBox" 10. *using* *namespace* std; 11. MainWindow::**MainWindow**(QWidget \*parent) 12. : QMainWindow(*parent*) 13. , ui(*new* Ui::MainWindow) 14. { 15. ui->setupUi(*this*); 16. } 17. MainWindow::~***MainWindow***() 18. { 19. *delete* ui; 20. } 21. vector<SearchInfo> searchInfo; 22. *const* char \*books[8] = { 23. "J.K. Rowling - Quidditch Through the Ages.txt", 24. "J.K. Rowling - HP 6 - Harry Potter and the Half-Blood Prince.txt", 25. "J.K. Rowling - HP 4 - Harry Potter and the Goblet of Fire.txt", 26. "J.K. Rowling - HP 3 - Harry Potter and the Prisoner of Azkaban.txt", 27. "J.K. Rowling - HP 0 - Harry Potter Prequel.txt", 28. "HP7--Harry\_Potter\_and\_the\_Deathly\_Hallows\_Book\_7\_.txt", 29. "HP2--Harry\_Potter\_and\_the\_Chamber\_of\_Secrets\_Book\_2\_.txt", 30. "J.K. Rowling - The Tales of Beedle the Bard.txt" 31. }; 32. *//* *点击查找按钮* 33. void MainWindow::**on\_pushButton\_clicked**() 34. { 35. ui->textBrowser->clear(); *//清屏* 36. searchInfo.clear(); *//清空上一次查找的缓存* 37. QString text = ui->lineEdit->text(); *//获取输入文本* 38. int num = 0; 39. *for*(int i = 0; i < 8; i++) { 40. fstream file; 41. file.open(books[i], ios::in); 42. *if* (!file.is\_open()) { 43. cerr << "文件打开失败" << endl; 44. } 45. char name[100]; 46. char buf[1024] = {0}; 47. string chapter = "CHAPTER"; 48. string chapter1 = "Chapter"; 49. int cur\_chapter = 0; 50. int cur\_page = 0; 51. *//scanf("%s",* *name);* *//输入查找的姓名* 52. strcpy(*name*, text.toStdString().c\_str()); *//转换用来查找的姓名* 53. *while* (file.getline(*buf*, *sizeof*(buf))) { 54. *if*(buf[0] == 0 && (buf[1] == '\r' || buf[1] == '\n')) *continue*; 55. *try* { 56. *if* (buf[0] == chapter[0] && buf[1] == chapter[1] && buf[2] == chapter[2] && 57. buf[3] == chapter[3] && buf[4] == chapter[4] && buf[5] == chapter[5] && buf[6] == chapter[6]) { 58. cur\_chapter++; 59. *continue*; 60. } *else* *if* (buf[3] == chapter[0] && buf[4] == chapter[1] && buf[5] == chapter[2] && 61. buf[6] == chapter[3] && buf[7] == chapter[4] && buf[8] == chapter[5] && buf[9] == chapter[6]){ 62. cur\_chapter++; *//用于跳过开头utf-8标识* 63. *continue*; 64. } *else* *if*(buf[0] == chapter1[0] && buf[1] == chapter1[1] && buf[2] == chapter1[2] && 65. buf[3] == chapter1[3] && buf[4] == chapter1[4] && buf[5] == chapter1[5] && buf[6] == chapter1[6]) { 66. cur\_chapter++; 67. *continue*; 68. } *else* *if* (buf[3] == chapter1[0] && buf[4] == chapter1[1] && buf[5] == chapter1[2] && 69. buf[6] == chapter1[3] && buf[7] == chapter1[4] && buf[8] == chapter1[5] && buf[9] == chapter1[6]){ 70. cur\_chapter++; *//用于跳过开头utf-8标识* 71. *continue*; 72. }*else* *if* (stoi(string(buf)) && stoi(string(buf)) == cur\_page + 1) { *//判断是否一行都是数字，一行都是数字说明是页码* 73. cur\_page++; 74. *continue*; 75. } 76. } *catch* (...) { 77. *//do* *nothing* 78. } 79. char \*buf\_point = buf; 80. *while* (\*buf\_point != 0) { 81. *if* (\*buf\_point == \*name) { 82. int p = 0; 83. int sign = 1; 84. *while*(\*(name+p) != 0){ 85. *if*(\*(name+p) != \*(buf\_point+p)){ 86. sign = 0; 87. *break*; 88. } 89. p++; 90. } 91. *if* (sign == 0) { 92. buf\_point++; 93. *continue*; 94. } *else* { 95. *//cout* *<<* *cur\_page* *<<* *"* *"* *<<* *cur\_chapter* *<<* *"* *"* *<<* *endl* *<<* *buf* *<<* *endl;* 96. searchInfo.push\_back(SearchInfo(i, cur\_page, cur\_chapter, string(buf))); 97. ui->textBrowser->append(QString::fromStdString("<a href='" + to\_string(++num) + "'>" + to\_string(num) + "</a>"+ "&nbsp;&nbsp;&nbsp;&nbsp;" + name + "&nbsp;&nbsp;&nbsp;&nbsp;" + to\_string(cur\_page) + "&nbsp;&nbsp;&nbsp;&nbsp;" + to\_string(cur\_chapter) + "&nbsp;&nbsp;&nbsp;&nbsp;"+ string(books[i]))); 98. } 99. } 100. buf\_point++; 101. } 102. } 103. file.close(); 104. } 105. } 106. SearchInfo::**SearchInfo**(int bookNum, int page, int chapter, std::string line):bookNum(bookNum), page(page), chapter(chapter), line(line) 107. { 108. } 109. SearchInfo::~**SearchInfo**(){} 110. string SearchInfo::**getLine**() 111. { 112. *return* *this*->line; 113. } 114. void MainWindow::**on\_pushButton\_2\_clicked**() 115. { 116. QString text = ui->lineEdit\_2->text();*//获取查询的text* 117. int num = 0; 118. *try* { 119. num = stoi(text.toStdString()); 120. } *catch* (...) { 121. QMessageBox::warning(*this*,tr("警告信息"),tr("请输入正确的编号")); 122. } 123. num--; 124. *if*(num >= int(searchInfo.size())){ 125. QMessageBox::warning(*this*,tr("警告信息"),tr("输入的编号过大")); 126. *return*; 127. } 128. ui->textBrowser\_2->clear(); 129. ui->textBrowser\_2->append(QString::fromStdString(searchInfo[num].getLine())); 130. } 131. void MainWindow::**on\_textBrowser\_anchorClicked**(*const* QUrl &arg1) 132. { 133. ui->textBrowser->setOpenExternalLinks(*false*); 134. ui->textBrowser->setOpenLinks(*false*); 135. QString clickedText = arg1.toString(); 136. *//cout* *<<* *clickedText.toStdString()* *<<* *endl;* 137. int num = stoi(clickedText.toStdString()) - 1; 138. ui->textBrowser\_2->clear(); 139. ui->textBrowser\_2->append(QString::fromStdString(searchInfo[num].getLine())); 140. } |

* 1. MainWindow::on\_pushButton\_clicked() 用来处理点击查找按钮，由最开始的核心代码修改。
  2. SearchInfo::SearchInfo() 用于构造对象。
  3. SearchInfo::getLine() 返回该对象存储的一行的内容。
  4. MainWindow::on\_pushButton\_2\_clicked() 用于处理点击第二个查询按钮，在最下面的textBrowser显示查询的那一句内容。
  5. MainWindow::on\_textBrowser\_anchorClicked(const QUrl &arg1) 用于处理中间位置的textBrowser的链接点击内容。

**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5640个结果。



测试用例2：在输入人名地名处输入“Ginny”，点击查询。

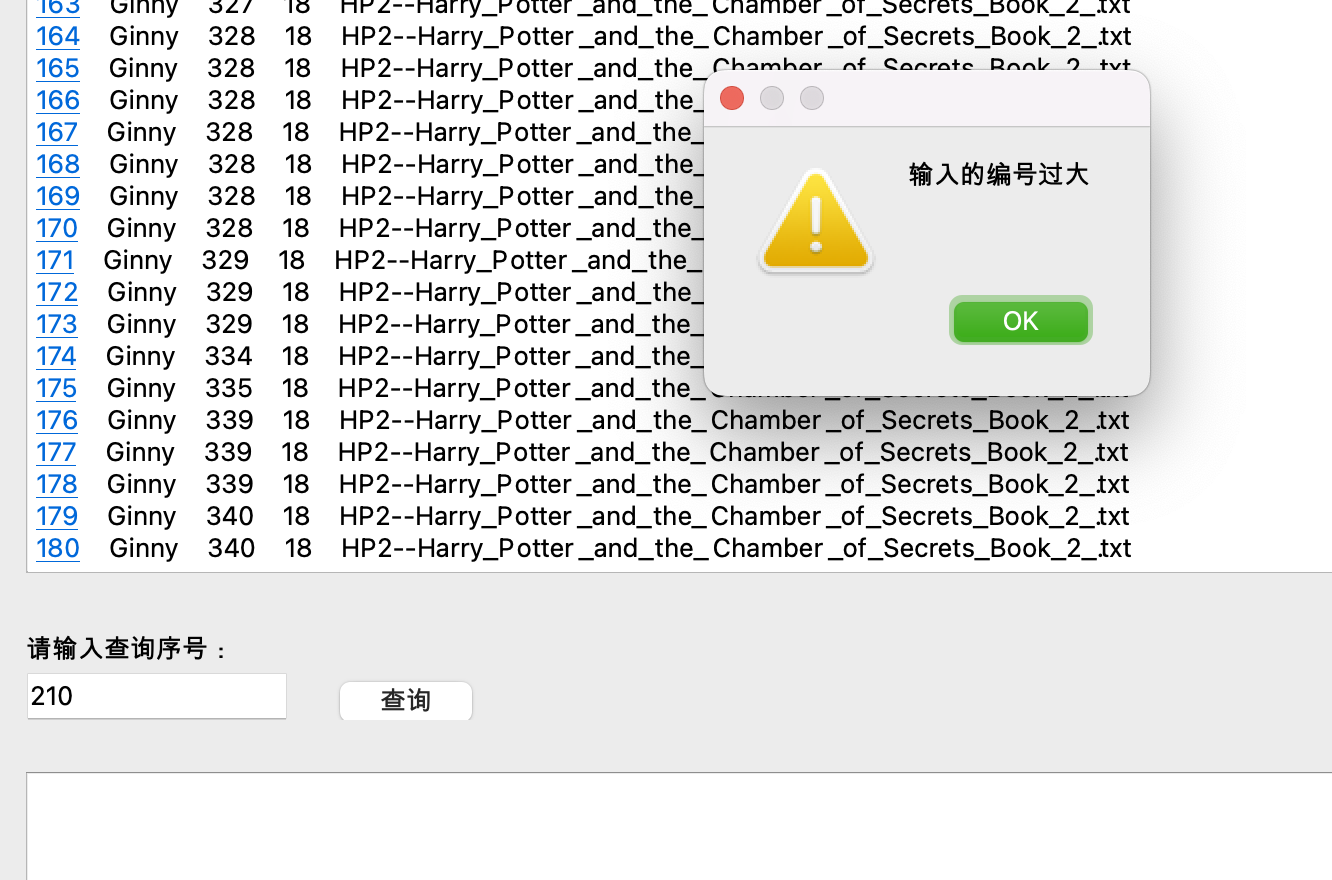
运行截图2：显示查询到的180个结果。

图形用户界面, 文本

描述已自动生成

测试用例3：输入查询序号210，点击查询

运行截图3：显示溢出。



测试用例4：输入查询序号170，点击查询

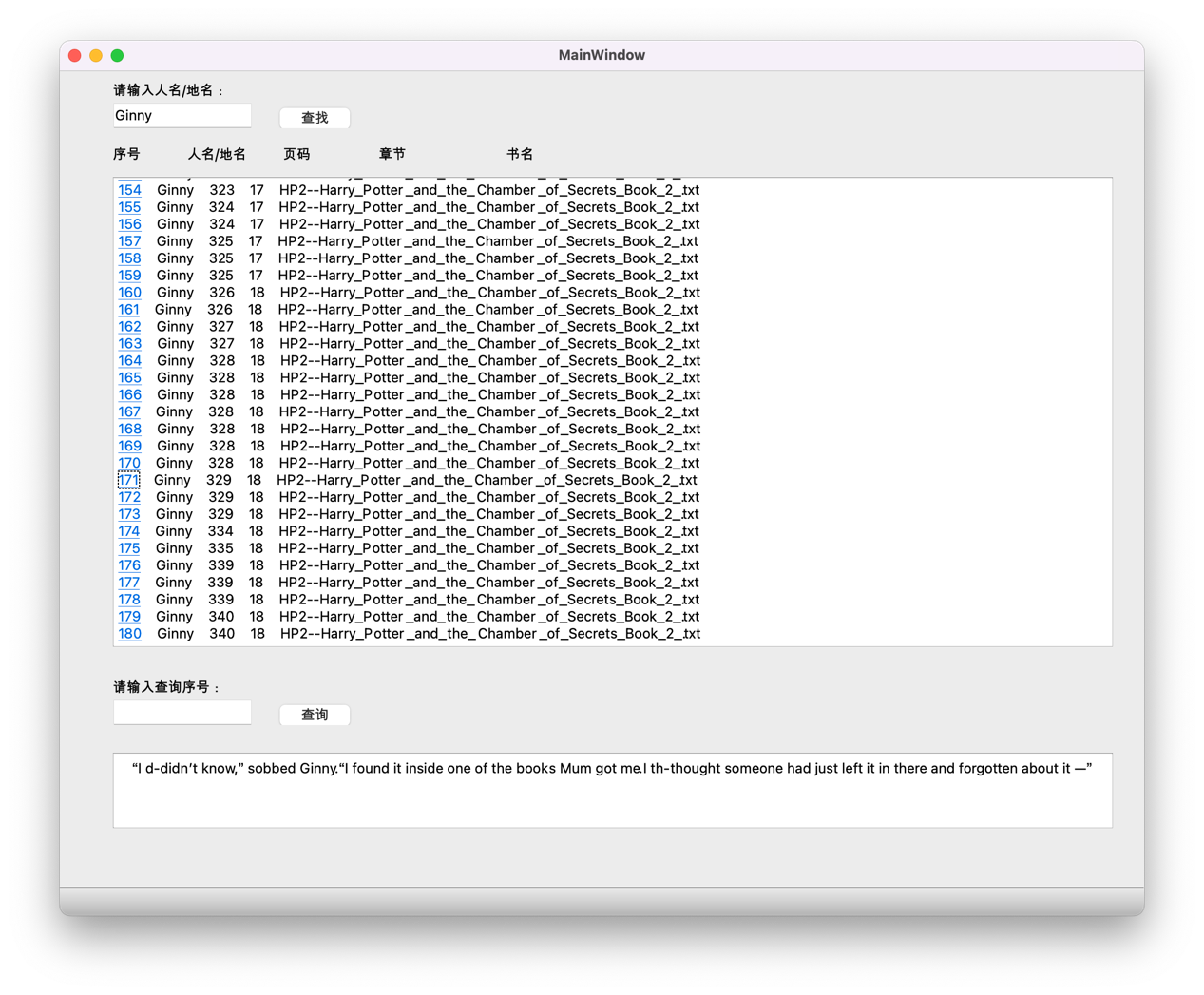
运行截图4：显示那一句话。

图形用户界面, 文本

描述已自动生成

测试用例5：点击序号171。

运行截图5：显示那一句话。



**实验结论与感想**

我学到了非常多。