

base.hpp

```
1 using namespace std;
2
3 class Base
4 {
5 public:
6     virtual int Add() const = 0;
7     virtual void Save() const = 0;
8     virtual int Delete() const = 0;
9     virtual int Edit() const = 0;
10 };
```

book.hpp

```
1 #define FILESYSTEM_BOOK "./data/book/"
2 #include <fstream>
3 #include <iconv.h>
4
5 string utf8_to_gbk(const string &utf8_str)
6 {
7     iconv_t cd = iconv_open("GBK", "UTF-8");
8     if (cd == (iconv_t)-1)
9         return "";
10    size_t in_bytes_left = utf8_str.size();
11    size_t out_bytes_left = in_bytes_left * 2;
12    char *in_buf = const_cast<char *>(utf8_str.c_str());
13    char out_buf[out_bytes_left];
14    char *out_buf_start = out_buf;
15    size_t ret = iconv(cd, &in_buf, &in_bytes_left, &out_buf_start,
16    &out_bytes_left);
17    if (ret == (size_t)-1)
18    {
19        iconv_close(cd);
20        return "";
21    }
22    *out_buf_start = '\0';
23    iconv_close(cd);
24    return string(out_buf);
25 }
26
27 string gbk_to_utf8(const string &gbk_str)
28 {
29     iconv_t cd = iconv_open("UTF-8", "GBK");
30     if (cd == (iconv_t)-1)
31         return "";
32    size_t in_bytes_left = gbk_str.size();
33    size_t out_bytes_left = in_bytes_left * 2;
34    char *in_buf = const_cast<char *>(gbk_str.c_str());
35    char out_buf[out_bytes_left];
36    char *out_buf_start = out_buf;
37    size_t ret = iconv(cd, &in_buf, &in_bytes_left, &out_buf_start,
38    &out_bytes_left);
39    if (ret == (size_t)-1)
40    {
```

```

39         iconv_close(cd);
40         return "";
41     }
42     *out_buf_start = '\0';
43     iconv_close(cd);
44     return string(out_buf);
45 }
46
47 class Book : public Base
48 {
49 public:
50     string title;
51     string author;
52     string category;
53     string keywords;
54     string summary;
55     int borrowTimes = 0;
56     bool isBorrowed = false;
57
58     Book(string Title = "", string Author = "", string Category = "",
59 string Keywords = "", string Summary = "") : title(Title), author(Author),
60 category(Category), keywords(Keywords), summary(Summary) {}
61
62     Book(const Book &book) : title(book.title), author(book.author),
63 category(book.category), keywords(book.keywords), summary(book.summary),
64 borrowTimes(book.borrowTimes), isBorrowed(book.isBorrowed) {}
65
66     ~Book() {}
67
68     int Add() const override
69     {
70         string filePath = FILESYSTEM_BOOK + utf8_to_gbk(this->title) +
71 ".txt";
72         if (ifstream(filePath))
73             return 0;
74         else
75         {
76             ofstream file(filePath);
77             if (!file)
78                 return -1;
79             else
80             {
81                 file << this->title << endl;
82                 file << this->author << endl;
83                 file << this->category << endl;
84                 file << this->keywords << endl;
85                 file << this->summary << endl;
86                 file << this->isBorrowed << endl;
87                 file << this->borrowTimes << endl;
88                 file.close();
89                 return 1;
90             }
91         }
92     }
93
94     void Save() const override
95     {

```

```

89     string filePath = FILESYSTEM_BOOK + utf8_to_gbk(this->title) +
".txt";
90     ofstream file(filePath);
91     file << this->title << endl;
92     file << this->author << endl;
93     file << this->category << endl;
94     file << this->keywords << endl;
95     file << this->summary << endl;
96     file << this->isBorrowed << endl;
97     file << this->borrowTimes << endl;
98     file.close();
99 }
100
101 int Delete() const override
102 {
103     string filePath = FILESYSTEM_BOOK + utf8_to_gbk(this->title) +
".txt";
104     if (remove(filePath.c_str()) == 0)
105         return 1;
106     else
107         return -1;
108 }
109
110 int Edit() const override
111 {
112     ofstream file(FILESYSTEM_BOOK + utf8_to_gbk(this->title) + ".txt");
113     if (!file)
114         return -1;
115     else
116     {
117         this->Save();
118         file.close();
119         return 1;
120     }
121 }
122
123 friend ostream &operator<<(ostream &, const Book &);
124 };
125
126 ostream &operator<<(ostream &os, const Book &book)
127 {
128     os << "书名: " << book.title << endl;
129     os << "作者: " << book.author << endl;
130     os << "分类: " << book.category << endl;
131     os << "关键词: " << book.keywords << endl;
132     os << "简介: " << book.summary << endl;
133     if (book.isBorrowed)
134         os << "借出状态: 已借出" << endl;
135     else
136         os << "借出状态: 未借出" << endl;
137     os << "借出次数: " << book.borrowTimes << endl;
138     return os;
139 }

```

```

1  #define FILESYSTEM_USER "./data/user/"
2  #include <vector>
3
4  struct Record
5  {
6      string bookName = "";
7      string borrowTime = "";
8      string returnTime = "";
9      bool isReturned = false;
10 };
11
12 class User : public Base
13 {
14 public:
15     string name;
16     vector<Record> borrowRecords;
17     int borrowTimes = 0;
18
19     User(string Name = "") : name(Name) {}
20     User(const User &user) : name(user.name),
        borrowRecords(user.borrowRecords), borrowTimes(user.borrowTimes) {}
21     ~User() {}
22
23     int Add() const override
24     {
25         string filePath = FILESYSTEM_USER + utf8_to_gbk(this->name) +
        ".txt";
26         if (ifstream(filePath))
27             return 0;
28         else
29         {
30             ofstream file(filePath);
31             if (!file)
32                 return -1;
33             else
34             {
35                 file.close();
36                 return 1;
37             }
38         }
39     }
40
41     void Save() const override
42     {
43         string filePath = FILESYSTEM_USER + utf8_to_gbk(this->name) +
        ".txt";
44         ofstream file(filePath);
45         for (auto record : this->borrowRecords)
46         {
47             file << record.bookName << endl;
48             file << record.borrowTime << endl;
49             file << record.returnTime << endl;
50             file << record.isReturned << endl;
51         }
52         file.close();

```

```

53     }
54
55     int Delete() const override
56     {
57         string filePath = FILESYSTEM_USER + utf8_to_gbk(this->name) +
".txt";
58         if (remove(filePath.c_str()) == 0)
59             return 1;
60         else
61             return -1;
62     }
63
64     int Edit() const override
65     {
66         ofstream file(FILESYSTEM_USER + utf8_to_gbk(this->name) + ".txt");
67         if (!file)
68             return -1;
69         else
70         {
71             this->Save();
72             file.close();
73             return 1;
74         }
75     }
76
77     friend ostream &operator<<(ostream &, const User &);
78 };
79
80 ostream &operator<<(ostream &os, const User &user)
81 {
82     os << "借阅次数: " << user.borrowTimes << endl;
83     os << endl;
84     os << "借阅记录: " << endl;
85     os << endl;
86     for (auto record : user.borrowRecords)
87     {
88         os << "书名: " << record.bookName << endl;
89         os << "借书时间: " << record.borrowTime << endl;
90         if (record.isReturned)
91             os << "还书时间: " << record.returnTime << endl;
92         else
93             os << "还书时间: 未还" << endl;
94         os << endl;
95     }
96     return os;
97 }

```

manager.hpp

```

1  #include "base.hpp"
2  #include "book.hpp"
3  #include "user.hpp"
4  #include <filesystem>
5  #include <algorithm>
6

```

```

7  class BookManager
8  {
9  public:
10     Book getBook(const string &title) const
11     {
12         string filePath = FILESYSTEM_BOOK + utf8_to_gbk(title) + ".txt";
13         if (!ifstream(filePath))
14         {
15             return Book();
16         }
17         else
18         {
19             ifstream file(filePath);
20             if (!file)
21             {
22                 return Book();
23             }
24             else
25             {
26                 Book book;
27                 getline(file, book.title);
28                 getline(file, book.author);
29                 getline(file, book.category);
30                 getline(file, book.keywords);
31                 getline(file, book.summary);
32                 string line;
33                 getline(file, line);
34                 book.isBorrowed = (line == "1");
35                 getline(file, line);
36                 book.borrowTimes = stoi(line);
37                 file.close();
38                 return book;
39             }
40         }
41     }
42
43     vector<Book> searchBook(const string &keyword) const
44     {
45         vector<Book> results;
46         for (const auto &entry :
filesystem::directory_iterator(FILESYSTEM_BOOK))
47         {
48             string filePath = entry.path().string();
49             filePath = utf8_to_gbk(filePath);
50             ifstream file(filePath);
51             if (file)
52             {
53                 Book book;
54                 getline(file, book.title);
55                 getline(file, book.author);
56                 getline(file, book.category);
57                 getline(file, book.keywords);
58                 getline(file, book.summary);
59                 string line;
60                 getline(file, line);
61                 book.isBorrowed = (line == "1");

```

```

62         getline(file, line);
63         book.borrowTimes = stoi(line);
64         file.close();
65         if (book.title.find(keyword) != string::npos ||
66             book.author.find(keyword) != string::npos ||
67             book.category.find(keyword) != string::npos ||
68             book.keywords.find(keyword) != string::npos ||
69             book.summary.find(keyword) != string::npos)
70         {
71             results.push_back(book);
72         }
73     }
74 }
75 return results;
76 }
77
78 vector<Book> tenHotBooks() const
79 {
80     vector<Book> results;
81     for (const auto &entry :
filesystem::directory_iterator(FILESYSTEM_BOOK))
82     {
83         string filePath = entry.path().string();
84         filePath = utf8_to_gbk(filePath);
85         ifstream file(filePath);
86         if (file)
87         {
88             Book book;
89             getline(file, book.title);
90             getline(file, book.author);
91             getline(file, book.category);
92             getline(file, book.keywords);
93             getline(file, book.summary);
94             string line;
95             getline(file, line);
96             book.isBorrowed = (line == "1");
97             getline(file, line);
98             book.borrowTimes = stoi(line);
99             file.close();
100             if (book.borrowTimes > 0)
101             {
102                 results.push_back(book);
103             }
104         }
105     }
106     sort(results.begin(), results.end(), [](Book a, Book b)
107         { return a.borrowTimes > b.borrowTimes; });
108     if (results.size() > 10)
109     {
110         results.resize(10);
111     }
112     return results;
113 }
114
115 int deleteAllBooks() const
116 {

```

```

117     for (const auto &entry :
filesystem::directory_iterator(FILESYSTEM_BOOK))
118     {
119         string filePath = entry.path().string();
120         filePath = utf8_to_gbk(filePath);
121         remove(filePath.c_str());
122     }
123     return 1;
124 }
125 };
126
127 class UserManager
128 {
129 public:
130     User getUser(const string &name) const
131     {
132         string filePath = FILESYSTEM_USER + name + ".txt";
133         filePath = utf8_to_gbk(filePath);
134         if (!ifstream(filePath))
135         {
136             return User();
137         }
138         else
139         {
140             ifstream file(filePath);
141             if (!file)
142             {
143                 return User();
144             }
145             else
146             {
147                 User user(name);
148                 string line;
149                 while (getline(file, line))
150                 {
151                     Record record;
152                     record.bookName = line;
153                     getline(file, line);
154                     record.borrowTime = line;
155                     getline(file, line);
156                     record.returnTime = line;
157                     getline(file, line);
158                     record.isReturned = (line == "1");
159                     user.borrowRecords.push_back(record);
160                 }
161                 user.borrowTimes = user.borrowRecords.size();
162                 file.close();
163                 return user;
164             }
165         }
166     }
167
168     vector<User> searchUser(const string &keyword) const
169     {
170         vector<User> results;

```



```

171         for (const auto &entry :
filesystem::directory_iterator(FILESYSTEM_USER))
172         {
173             string filePath = entry.path().string();
174             filePath = utf8_to_gbk(filePath);
175             ifstream file(filePath);
176             if (file)
177             {
178                 User user;
179                 user.name = gbk_to_utf8(filePath.substr(12, filePath.size()
- 16));
180
181                 string line;
182                 while (getline(file, line))
183                 {
184                     Record record;
185                     record.bookName = line;
186                     getline(file, line);
187                     record.borrowTime = line;
188                     getline(file, line);
189                     record.returnTime = line;
190                     getline(file, line);
191                     record.isReturned = (line == "1");
192                     user.borrowRecords.push_back(record);
193                 }
194                 user.borrowTimes = user.borrowRecords.size();
195                 file.close();
196                 if (user.name.find(keyword) != string::npos)
197                 {
198                     results.push_back(user);
199                 }
200             }
201             return results;
202         }
203
204     vector<User> tenActiveUsers() const
205     {
206         vector<User> results;
207         for (const auto &entry :
filesystem::directory_iterator(FILESYSTEM_USER))
208         {
209             string filePath = entry.path().string();
210             filePath = utf8_to_gbk(filePath);
211             ifstream file(filePath);
212             if (file)
213             {
214                 User user;
215                 user.name = gbk_to_utf8(filePath.substr(12, filePath.size()
- 16));
216
217                 string line;
218                 while (getline(file, line))
219                 {
220                     Record record;
221                     record.bookName = line;
222                     getline(file, line);

```

```

223         getline(file, line);
224         record.returnTime = line;
225         getline(file, line);
226         record.isReturned = (line == "1");
227         user.borrowRecords.push_back(record);
228     }
229     user.borrowTimes = user.borrowRecords.size();
230     file.close();
231     if (user.borrowTimes > 0)
232     {
233         results.push_back(user);
234     }
235 }
236 }
237 sort(results.begin(), results.end(), [](User a, User b)
238     { return a.borrowTimes > b.borrowTimes; });
239 if (results.size() > 10)
240 {
241     results.resize(10);
242 }
243 return results;
244 }
245
246 int deleteAllUsers() const
247 {
248     for (const auto &entry :
filesystem::directory_iterator(FILESYSTEM_USER))
249     {
250         string filePath = entry.path().string();
251         filePath = utf8_to_gbk(filePath);
252         remove(filePath.c_str());
253     }
254     return 1;
255 }
256 };
257
258 class Manager : public BookManager, public UserManager
259 {
260 public:
261     string getCurrentDateTime() const
262     {
263         time_t now = time(0);
264         tm *ltm = localtime(&now);
265         char buffer[80];
266         strftime(buffer, sizeof(buffer), "%Y-%m-%d %H:%M:%S", ltm);
267         return buffer;
268     }
269
270     int borrowBook(const string &userName, const string &bookName) const
271     {
272         Book book = getBook(bookName);
273         if (book.title == "")
274             return 0;
275         User user = getUser(userName);
276         if (user.name == "")
277             return -2;

```

```

278         if (book.isBorrowed)
279             return -1;
280         Record record;
281         record.bookName = bookName;
282         record.borrowTime = getCurrentDateTime();
283         record.isReturned = false;
284         user.borrowRecords.push_back(record);
285         user.borrowTimes = user.borrowRecords.size();
286         user.Save();
287         book.isBorrowed = true;
288         book.borrowTimes++;
289         book.Save();
290         return 1;
291     }
292
293     int returnBook(const string &userName, const string &bookName) const
294     {
295         Book book = getBook(bookName);
296         if (book.title == "")
297             return 0;
298         User user = getUser(userName);
299         if (user.name == "")
300             return -2;
301         bool found = false;
302         for (auto &record : user.borrowRecords)
303         {
304             if (record.bookName == bookName && !record.isReturned)
305             {
306                 record.returnTime = getCurrentDateTime();
307                 record.isReturned = true;
308                 user.Save();
309                 book.isBorrowed = false;
310                 book.Save();
311                 return 1;
312             }
313         }
314         return -1;
315     }
316 };

```

gui.hpp

```

1  #include "manager.hpp"
2  #include <conio.h>
3  #include <iostream>
4
5  class GUI : public Manager
6  {
7  public:
8      void ShowMenu() const
9      {
10         system("cls");
11         cout << endl;
12         cout << endl;
13         cout << "        图书管理系统" << endl;

```

```

14     cout << "-----" << endl;
15     cout << "1. 添加书籍          9. 图书借阅" << endl;
16     cout << "2. 删除书籍          10. 图书归还" << endl;
17     cout << "3. 查找书籍          11. 借阅记录" << endl;
18     cout << "4. 编辑书籍          12. 十大热门书籍" << endl;
19     cout << "5. 添加用户          13. 十大活跃用户" << endl;
20     cout << "6. 删除用户          14. 删除所有书籍" << endl;
21     cout << "7. 查找用户          15. 删除所有用户" << endl;
22     cout << "8. 编辑用户          16. 退出" << endl;
23     cout << "-----" << endl;
24     cout << endl;
25     cout << "请选择操作: ";
26 }
27
28 string RemoveBlank(const string &str) const
29 {
30     auto start = str.find_first_not_of(" \t\n\r\f\v");
31     if (start == string::npos)
32         return "";
33     auto end = str.find_last_not_of(" \t\n\r\f\v");
34     return str.substr(start, end - start + 1);
35 }
36
37 void DisplayBook(const Book &book) const
38 {
39     cout << book;
40 }
41
42 void AddBook() const
43 {
44     system("cls");
45     Book book;
46     cout << endl;
47     cout << endl;
48     cout << "添加书籍" << endl;
49     cout << endl;
50     cout << endl;
51     cout << "请输入书名: ";
52     getline(cin, book.title);
53     book.title = RemoveBlank(book.title);
54     if (book.title.empty())
55     {
56         cout << endl;
57         cout << "书名不能为空" << endl;
58         cout << endl;
59         cout << "按任意键返回" << endl;
60         getch();
61         return;
62     }
63     else if (getBook(book.title).title == book.title)
64     {
65         cout << "书籍已存在" << endl;
66         cout << endl;
67         cout << "按任意键返回" << endl;
68         getch();
69         return;

```

```

70     }
71     cout << "请输入作者: ";
72     getline(cin, book.author);
73     book.author = RemoveBlank(book.author);
74     cout << "请输入分类: ";
75     getline(cin, book.category);
76     book.category = RemoveBlank(book.category);
77     cout << "请输入关键词: ";
78     getline(cin, book.keywords);
79     book.keywords = RemoveBlank(book.keywords);
80     cout << "请输入简介: ";
81     getline(cin, book.summary);
82     book.summary = RemoveBlank(book.summary);
83     cout << endl;
84     int result = book.Add();
85     switch (result)
86     {
87     case 0:
88         cout << "书籍已存在" << endl;
89         break;
90     case -1:
91         cout << "保存失败" << endl;
92         break;
93     case 1:
94         cout << "保存成功" << endl;
95         break;
96     }
97     cout << endl;
98     cout << "按任意键返回" << endl;
99     getch();
100 }
101
102 void DeleteBook() const
103 {
104     system("cls");
105     cout << endl;
106     cout << endl;
107     cout << "删除书籍" << endl;
108     cout << endl;
109     cout << endl;
110     cout << "请输入书名: ";
111     string title;
112     getline(cin, title);
113     title = RemoveBlank(title);
114     cout << endl;
115     if (title.empty())
116     {
117         cout << "书名不能为空" << endl;
118         cout << endl;
119         cout << "按任意键返回" << endl;
120         getch();
121         return;
122     }
123     if (getBook(title).title.empty())
124     {
125         cout << "书籍不存在" << endl;

```

```

126         cout << endl;
127         cout << "按任意键返回" << endl;
128         getch();
129         return;
130     }
131     Book book(title);
132     cout << "确认删除? (y/n)";
133     string c;
134     getline(cin, c);
135     cout << endl;
136     if (c != "y")
137     {
138         cout << "取消删除" << endl;
139         cout << endl;
140         cout << "按任意键返回" << endl;
141         getch();
142         return;
143     }
144     int result = book.Delete();
145     switch (result)
146     {
147     case -1:
148         cout << "删除失败" << endl;
149         break;
150     case 1:
151         cout << "删除成功" << endl;
152         break;
153     }
154     cout << endl;
155     cout << "按任意键返回" << endl;
156     getch();
157 }
158
159 void SearchBook() const
160 {
161     system("cls");
162     cout << endl;
163     cout << endl;
164     cout << "查找书籍" << endl;
165     cout << endl;
166     cout << endl;
167     cout << "请输入搜索词（回车显示所有书籍）： ";
168     string title;
169     getline(cin, title);
170     title = RemoveBlank(title);
171     cout << endl;
172     cout << "查询结果" << endl;
173     cout << endl;
174     vector<Book> books = searchBook(title);
175     int result = books.size() == 0 ? 0 : 1;
176     switch (result)
177     {
178     case 0:
179         cout << "书籍不存在" << endl;
180         break;
181     case 1:

```

```

182         for (int i = 0; i < books.size(); i++)
183         {
184             cout << "书籍" << i + 1 << endl;
185             DisplayBook(books[i]);
186             cout << endl;
187         }
188     }
189     cout << endl;
190     cout << "按任意键返回" << endl;
191     getch();
192 }
193
194 void EditBook() const
195 {
196     system("cls");
197     cout << endl;
198     cout << endl;
199     cout << "编辑书籍" << endl;
200     cout << endl;
201     cout << endl;
202     cout << "请输入书名: ";
203     string title;
204     getline(cin, title);
205     title = RemoveBlank(title);
206     cout << endl;
207     if (title.empty())
208     {
209         cout << "书名不能为空" << endl;
210         cout << endl;
211         cout << "按任意键返回" << endl;
212         getch();
213         return;
214     }
215     Book oldBook = getBook(title);
216     int result = oldBook.title.empty() ? 0 : 1;
217     switch (result)
218     {
219     case 0:
220         cout << "书籍不存在" << endl;
221         break;
222     case 1:
223         DisplayBook(oldBook);
224         Book book;
225         cout << endl;
226         cout << "请输入新书名: ";
227         getline(cin, book.title);
228         book.title = RemoveBlank(book.title);
229         if (book.title.empty())
230         {
231             cout << endl;
232             cout << "书名不能为空" << endl;
233             cout << endl;
234             cout << "按任意键返回" << endl;
235             getch();
236             return;
237         }

```

```

238         cout << "请输入新作者: ";
239         getline(cin, book.author);
240         book.author = RemoveBlank(book.author);
241         cout << "请输入新分类: ";
242         getline(cin, book.category);
243         book.category = RemoveBlank(book.category);
244         cout << "请输入新关键词: ";
245         getline(cin, book.keywords);
246         book.keywords = RemoveBlank(book.keywords);
247         cout << "请输入新简介: ";
248         getline(cin, book.summary);
249         book.summary = RemoveBlank(book.summary);
250         cout << endl;
251         book.isBorrowed = oldBook.isBorrowed;
252         book.borrowTimes = oldBook.borrowTimes;
253         oldBook.Delete();
254         int result = book.Edit();
255         switch (result)
256         {
257             case -1:
258                 cout << "保存失败" << endl;
259                 break;
260             case 1:
261                 cout << "保存成功" << endl;
262                 break;
263         }
264     }
265     cout << endl;
266     cout << "按任意键返回" << endl;
267     getch();
268 }
269
270 void DisplayUser(const User &user) const
271 {
272     cout << user;
273 }
274
275 void AddUser() const
276 {
277     system("cls");
278     User user;
279     cout << endl;
280     cout << endl;
281     cout << "添加用户" << endl;
282     cout << endl;
283     cout << endl;
284     cout << "请输入用户名: ";
285     getline(cin, user.name);
286     user.name = RemoveBlank(user.name);
287     cout << endl;
288     if (user.name.empty())
289     {
290         cout << "用户名不能为空" << endl;
291         cout << endl;
292         cout << "按任意键返回" << endl;
293         getch();

```



```

294         return;
295     }
296     int result = user.Add();
297     switch (result)
298     {
299     case 0:
300         cout << "用户已存在" << endl;
301         break;
302     case -1:
303         cout << "保存失败" << endl;
304         break;
305     case 1:
306         cout << "保存成功" << endl;
307         break;
308     }
309     cout << endl;
310     cout << "按任意键返回" << endl;
311     getch();
312 }
313
314 void DeleteUser() const
315 {
316     system("cls");
317     cout << endl;
318     cout << endl;
319     cout << "删除用户" << endl;
320     cout << endl;
321     cout << endl;
322     cout << "请输入用户名: ";
323     string name;
324     getline(cin, name);
325     name = RemoveBlank(name);
326     cout << endl;
327     if (name.empty())
328     {
329         cout << "用户名不能为空" << endl;
330         cout << endl;
331         cout << "按任意键返回" << endl;
332         getch();
333         return;
334     }
335     if (getUser(name).name.empty())
336     {
337         cout << "用户不存在" << endl;
338         cout << endl;
339         cout << "按任意键返回" << endl;
340         getch();
341         return;
342     }
343     User user(name);
344     cout << "确认删除? (y/n)";
345     string c;
346     getline(cin, c);
347     cout << endl;
348     if (c != "y")
349     {

```

```

350         cout << "取消删除" << endl;
351         cout << endl;
352         cout << "按任意键返回" << endl;
353         getch();
354         return;
355     }
356     int result = user.Delete();
357     switch (result)
358     {
359     case 1:
360         cout << "删除成功" << endl;
361         break;
362     case -1:
363         cout << "删除失败" << endl;
364         break;
365     }
366     cout << endl;
367     cout << "按任意键返回" << endl;
368     getch();
369 }
370
371 void SearchUser() const
372 {
373     system("cls");
374     cout << endl;
375     cout << endl;
376     cout << "查找用户" << endl;
377     cout << endl;
378     cout << endl;
379     cout << "请输入用户名（回车显示所有用户）： ";
380     string name;
381     getline(cin, name);
382     cout << endl;
383     cout << "查询结果" << endl;
384     cout << endl;
385     vector<User> users = searchUser(name);
386     int result = users.size() == 0 ? 0 : 1;
387     switch (result)
388     {
389     case 0:
390         cout << "用户不存在" << endl;
391         break;
392     case 1:
393         for (int i = 0; i < users.size(); i++)
394         {
395             cout << "用户" << i + 1 << ": " << users[i].name << endl;
396             cout << endl;
397         }
398     }
399     cout << endl;
400     cout << "按任意键返回" << endl;
401     getch();
402 }
403
404 void EditUser() const
405 {

```

```

406     system("cls");
407     cout << endl;
408     cout << endl;
409     cout << "编辑用户" << endl;
410     cout << endl;
411     cout << endl;
412     cout << "请输入用户名: ";
413     string oldname;
414     getline(cin, oldname);
415     oldname = RemoveBlank(oldname);
416     cout << endl;
417     if (oldname.empty())
418     {
419         cout << "用户名不能为空" << endl;
420         cout << endl;
421         cout << "按任意键返回" << endl;
422         getch();
423         return;
424     }
425     User oldUser = getUser(oldname);
426     int result = oldUser.name.empty() ? 0 : 1;
427     switch (result)
428     {
429     case 0:
430         cout << "用户不存在" << endl;
431         break;
432     case 1:
433         cout << "请输入新用户名: ";
434         User user;
435         getline(cin, user.name);
436         user.name = RemoveBlank(user.name);
437         cout << endl;
438         if (user.name.empty())
439         {
440             cout << "用户名不能为空" << endl;
441             cout << endl;
442             cout << "按任意键返回" << endl;
443             getch();
444             return;
445         }
446         oldUser.Delete();
447         int result = user.Edit();
448         switch (result)
449         {
450         case -1:
451             cout << "保存失败" << endl;
452             break;
453         case 1:
454             cout << "保存成功" << endl;
455             break;
456         }
457     }
458     cout << endl;
459     cout << "按任意键返回" << endl;
460     getch();
461 }

```

```
462
463 void BorrowBook() const
464 {
465     system("cls");
466     cout << endl;
467     cout << endl;
468     cout << "图书借阅" << endl;
469     cout << endl;
470     cout << endl;
471     cout << "请输入书名: ";
472     string title;
473     getline(cin, title);
474     title = RemoveBlank(title);
475     cout << endl;
476     if (title.empty())
477     {
478         cout << "书名不能为空" << endl;
479         cout << endl;
480         cout << "按任意键返回" << endl;
481         getch();
482         return;
483     }
484     cout << "请输入用户名: ";
485     string name;
486     getline(cin, name);
487     name = RemoveBlank(name);
488     cout << endl;
489     if (name.empty())
490     {
491         cout << "用户名不能为空" << endl;
492         cout << endl;
493         cout << "按任意键返回" << endl;
494         getch();
495         return;
496     }
497     int result = borrowBook(name, title);
498     switch (result)
499     {
500     case 0:
501         cout << "书籍不存在" << endl;
502         break;
503     case -1:
504         cout << "书籍已借出" << endl;
505         break;
506     case -2:
507         cout << "用户不存在" << endl;
508         break;
509     case 1:
510         cout << "图书借阅成功" << endl;
511         break;
512     }
513     cout << endl;
514     cout << "按任意键返回" << endl;
515     getch();
516 }
517
```

```

518 void ReturnBook() const
519 {
520     system("cls");
521     cout << endl;
522     cout << endl;
523     cout << "图书归还" << endl;
524     cout << endl;
525     cout << endl;
526     cout << "请输入书名: ";
527     string title;
528     getline(cin, title);
529     title = RemoveBlank(title);
530     cout << endl;
531     if (title.empty())
532     {
533         cout << "书名不能为空" << endl;
534         cout << endl;
535         cout << "按任意键返回" << endl;
536         getch();
537         return;
538     }
539     cout << "请输入用户名: ";
540     string name;
541     getline(cin, name);
542     name = RemoveBlank(name);
543     cout << endl;
544     if (name.empty())
545     {
546         cout << "用户名不能为空" << endl;
547         cout << endl;
548         cout << "按任意键返回" << endl;
549         getch();
550         return;
551     }
552     int result = returnBook(name, title);
553     switch (result)
554     {
555     case 0:
556         cout << "书籍不存在" << endl;
557         break;
558     case -1:
559         cout << "未借此书籍" << endl;
560         break;
561     case -2:
562         cout << "用户不存在" << endl;
563         break;
564     case 1:
565         cout << "图书归还成功" << endl;
566         break;
567     }
568     cout << endl;
569     cout << "按任意键返回" << endl;
570     getch();
571 }
572
573 void BorrowRecord() const

```

```

574     {
575         system("cls");
576         cout << endl;
577         cout << endl;
578         cout << "借阅记录" << endl;
579         cout << endl;
580         cout << endl;
581         cout << "请输入用户名: ";
582         string name;
583         getline(cin, name);
584         name = RemoveBlank(name);
585         cout << endl;
586         if (name.empty())
587         {
588             cout << "用户名不能为空" << endl;
589             cout << endl;
590             cout << "按任意键返回" << endl;
591             getch();
592             return;
593         }
594         User user = getUser(name);
595         int result = user.name.empty() ? 0 : 1;
596         switch (result)
597         {
598             case 0:
599                 cout << "用户不存在" << endl;
600                 break;
601             case 1:
602                 DisplayUser(user);
603         }
604         cout << endl;
605         cout << "按任意键返回" << endl;
606         getch();
607     }
608
609     void TenHotBooks() const
610     {
611         system("cls");
612         cout << endl;
613         cout << endl;
614         cout << "十大热门书籍" << endl;
615         cout << endl;
616         cout << endl;
617         vector<Book> books = tenHotBooks();
618         if (books.size() == 0)
619         {
620             cout << "无记录" << endl;
621             cout << endl;
622             cout << "按任意键返回" << endl;
623             getch();
624             return;
625         }
626         for (int i = 0; i < books.size(); i++)
627         {
628             cout << "书籍" << i + 1 << endl;
629             DisplayBook(books[i]);

```

```

630         cout << endl;
631     }
632     cout << endl;
633     cout << "按任意键返回" << endl;
634     getch();
635 }
636
637 void TenActiveUsers() const
638 {
639     system("cls");
640     cout << endl;
641     cout << endl;
642     cout << "十大活跃用户" << endl;
643     cout << endl;
644     cout << endl;
645     vector<User> users = tenActiveUsers();
646     if (users.size() == 0)
647     {
648         cout << "无记录" << endl;
649         cout << endl;
650         cout << "按任意键返回" << endl;
651         getch();
652         return;
653     }
654     for (int i = 0; i < users.size(); i++)
655     {
656         cout << "用户" << i + 1 << ": " << users[i].name << endl;
657         cout << "借阅次数: " << users[i].borrowTimes << endl;
658         cout << endl;
659     }
660     cout << endl;
661     cout << "按任意键返回" << endl;
662     getch();
663 }
664
665 void DeleteAllBooks() const
666 {
667     system("cls");
668     cout << endl;
669     cout << endl;
670     cout << "确认删除所有书籍? (y/n)";
671     string c;
672     getline(cin, c);
673     cout << endl;
674     if (c == "y")
675     {
676         int result = deleteAllBooks();
677         switch (result)
678         {
679             case 1:
680                 cout << "删除成功" << endl;
681                 break;
682             default:
683                 cout << "删除失败" << endl;
684                 break;
685         }

```

```

686     }
687     else
688     {
689         cout << "取消删除" << endl;
690     }
691     cout << endl;
692     cout << "按任意键返回" << endl;
693     getch();
694 }
695
696 void DeleteAllUsers() const
697 {
698     system("cls");
699     cout << endl;
700     cout << endl;
701     cout << "确认删除所有用户? (y/n)";
702     string c;
703     getline(cin, c);
704     cout << endl;
705     if (c == "y")
706     {
707         int result = deleteAllUsers();
708         switch (result)
709         {
710             case 1:
711                 cout << "删除成功" << endl;
712                 break;
713             default:
714                 cout << "删除失败" << endl;
715                 break;
716         }
717     }
718     else
719     {
720         cout << "取消删除" << endl;
721     }
722     cout << endl;
723     cout << "按任意键返回" << endl;
724     getch();
725 }
726
727 void Exit() const
728 {
729     exit(0);
730 }
731
732 void Error() const
733 {
734     cout << endl;
735     cout << "无效输入, 请重新输入" << endl;
736 }
737 };

```



```

1  #include "gui.hpp"
2
3  enum Choice
4  {
5      AddBook = 1,
6      DeleteBook,
7      SearchBook,
8      EditBook,
9      AddUser,
10     DeleteUser,
11     SearchUser,
12     EditUser,
13     BorrowBook,
14     ReturnBook,
15     BorrowRecord,
16     TenHotBooks,
17     TenActiveUsers,
18     DeleteAllBooks,
19     DeleteAllUsers,
20     Exit
21 };
22
23 class Library : public GUI
24 {
25 public:
26     void CheckDirectory()
27     {
28         if (!filesystem::exists(FILESYSTEM_BOOK))
29             filesystem::create_directories(FILESYSTEM_BOOK);
30         if (!filesystem::exists(FILESYSTEM_USER))
31             filesystem::create_directories(FILESYSTEM_USER);
32     }
33 };

```

main.cpp

```

1  #include "library.hpp"
2
3  bool IsPureNumber(const string &input)
4  {
5      return all_of(input.begin(), input.end(), ::isdigit);
6  }
7
8  int main()
9  {
10     Library library;
11     bool error = false;
12     system("chcp 65001");
13
14     while (true)
15     {
16         library.CheckDirectory();
17         library.ShowMenu();
18         if (error)
19             library.Error();

```

```
20
21     string input;
22     int choice;
23     getline(cin, input);
24     if (!IsPureNumber(input) ||
25         input.empty() ||
26         (choice = stoi(input)) > Exit || choice < AddBook)
27     {
28         error = true;
29         continue;
30     }
31
32     switch (choice)
33     {
34     case AddBook:
35         library.AddBook();
36         break;
37     case DeleteBook:
38         library.DeleteBook();
39         break;
40     case SearchBook:
41         library.SearchBook();
42         break;
43     case EditBook:
44         library.EditBook();
45         break;
46     case AddUser:
47         library.AddUser();
48         break;
49     case DeleteUser:
50         library.DeleteUser();
51         break;
52     case SearchUser:
53         library.SearchUser();
54         break;
55     case EditUser:
56         library.EditUser();
57         break;
58     case BorrowBook:
59         library.BorrowBook();
60         break;
61     case ReturnBook:
62         library.ReturnBook();
63         break;
64     case BorrowRecord:
65         library.BorrowRecord();
66         break;
67     case TenHotBooks:
68         library.TenHotBooks();
69         break;
70     case TenActiveUsers:
71         library.TenActiveUsers();
72         break;
73     case DeleteAllBooks:
74         library.DeleteAllBooks();
75         break;
```

```
76         case DeleteAllUsers:
77             library.DeleteAllUsers();
78             break;
79         case Exit:
80             library.Exit();
81     }
82     error = false;
83 }
84 return 0;
85 }
```