

图书管理系统

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;
```

```

36     size_t ret = iconv(cd, &in_buf, &in_bytes_left, &out_buf_start,
&out_bytes_left);
37     if (ret == (size_t)-1)
38     {
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 = "",
string Keywords = "", string Summary = "") : title(Title), author(Author),
category(Category), keywords(Keywords), summary(Summary) {}
59     Book(const Book &book) : title(book.title), author(book.author),
category(book.category), keywords(book.keywords), summary(book.summary),
borrowTimes(book.borrowTimes), isBorrowed(book.isBorrowed) {}
60     ~Book() {}
61
62     int Add() const override
63     {
64         string filePath = FILESYSTEM_BOOK + utf8_to_gbk(this->title) +
".txt";
65         if (ifstream(filePath))
66             return 0;
67         else
68         {
69             ofstream file(filePath);
70             if (!file)
71                 return -1;
72             else
73             {
74                 file << this->title << endl;
75                 file << this->author << endl;
76                 file << this->category << endl;
77                 file << this->keywords << endl;
78                 file << this->summary << endl;
79                 file << this->isBorrowed << endl;
80                 file << this->borrowTimes << endl;
81                 file.close();
82                 return 1;
83             }
84         }
85     }

```

```
86
87     void Save() const override
88     {
89         string filePath = FILESYSTEM_BOOK + utf8_to_gbk(this->title) +
90         ".txt";
91         ofstream file(filePath);
92         file << this->title << endl;
93         file << this->author << endl;
94         file << this->category << endl;
95         file << this->keywords << endl;
96         file << this->summary << endl;
97         file << this->isBorrowed << endl;
98         file << this->borrowTimes << endl;
99         file.close();
100     }
101
102     int Delete() const override
103     {
104         string filePath = FILESYSTEM_BOOK + utf8_to_gbk(this->title) +
105         ".txt";
106         if (remove(filePath.c_str()) == 0)
107             return 1;
108         else
109             return -1;
110     }
111
112     int Edit() const override
113     {
114         ofstream file(FILESYSTEM_BOOK + utf8_to_gbk(this->title) + ".txt");
115         if (!file)
116             return -1;
117         else
118         {
119             this->Save();
120             file.close();
121             return 1;
122         }
123     }
124
125     friend ostream &operator<<(ostream &, const Book &);
126 };
127
128 ostream &operator<<(ostream &os, const Book &book)
129 {
130     os << "书名: " << book.title << endl;
131     os << "作者: " << book.author << endl;
132     os << "分类: " << book.category << endl;
133     os << "关键词: " << book.keywords << endl;
134     os << "简介: " << book.summary << endl;
135     if (book.isBorrowed)
136         os << "借出状态: 已借出" << endl;
137     else
138         os << "借出状态: 未借出" << endl;
139     os << "借出次数: " << book.borrowTimes << endl;
140     return os;
141 }
```

```
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),
21 borrowRecords(user.borrowRecords), borrowTimes(user.borrowTimes) {}
22     ~User() {}
23
24     int Add() const override
25     {
26         string filePath = FILESYSTEM_USER + utf8_to_gbk(this->name) +
27 ".txt";
28         if (ifstream(filePath))
29             return 0;
30         else
31         {
32             ofstream file(filePath);
33             if (!file)
34                 return -1;
35             else
36             {
37                 file.close();
38                 return 1;
39             }
40         }
41
42     void Save() const override
43     {
44         string filePath = FILESYSTEM_USER + utf8_to_gbk(this->name) +
45 ".txt";
46         ofstream file(filePath);
47         for (auto record : this->borrowRecords)
48         {
49             file << record.bookName << endl;
50             file << record.borrowTime << endl;
51             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 #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 }

```

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 };

```

library.hpp

```
1  #include "gui.hpp"
2
3  class Library : public GUI
4  {
5  public:
6      void CheckDirectory()
7      {
8          if (!filesystem::exists(FILESYSTEM_BOOK))
9              filesystem::create_directories(FILESYSTEM_BOOK);
10         if (!filesystem::exists(FILESYSTEM_USER))
11             filesystem::create_directories(FILESYSTEM_USER);
12     }
13 };
```

main.cpp

```
1  #include "library.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 bool IsPureNumber(const string &input)
24 {
25     return all_of(input.begin(), input.end(), ::isdigit);
26 }
27
28 int main()
29 {
30     Library library;
31     bool error = false;
32     system("chcp 65001");
33
34     while (true)
35     {
36         library.CheckDirectory();
37         library.ShowMenu();
```

```
38         if (error)
39             library.Error();
40
41         string input;
42         int choice;
43         getline(cin, input);
44         if (!IsPureNumber(input) ||
45             input.empty() ||
46             (choice = stoi(input)) > Exit || choice < AddBook)
47         {
48             error = true;
49             continue;
50         }
51
52         switch (choice)
53         {
54             case AddBook:
55                 library.AddBook();
56                 break;
57             case DeleteBook:
58                 library.DeleteBook();
59                 break;
60             case SearchBook:
61                 library.SearchBook();
62                 break;
63             case EditBook:
64                 library.EditBook();
65                 break;
66             case AddUser:
67                 library.AddUser();
68                 break;
69             case DeleteUser:
70                 library.DeleteUser();
71                 break;
72             case SearchUser:
73                 library.SearchUser();
74                 break;
75             case EditUser:
76                 library.EditUser();
77                 break;
78             case BorrowBook:
79                 library.BorrowBook();
80                 break;
81             case ReturnBook:
82                 library.ReturnBook();
83                 break;
84             case BorrowRecord:
85                 library.BorrowRecord();
86                 break;
87             case TenHotBooks:
88                 library.TenHotBooks();
89                 break;
90             case TenActiveUsers:
91                 library.TenActiveUsers();
92                 break;
93             case DeleteAllBooks:
```

```
94         library.DeleteAllBooks();
95         break;
96     case DeleteAllUsers:
97         library.DeleteAllUsers();
98         break;
99     case Exit:
100         library.Exit();
101     }
102     error = false;
103 }
104 return 0;
105 }
```