

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
43     void Save() const override
44     {
45         string filePath = FILESYSTEM_USER + utf8_to_gbk(this->name) +
46 ".txt";
47         ofstream file(filePath);
48         for (auto record : this->borrowRecords)
49         {
50             file << record.bookName << endl;
51             file << record.borrowTime << endl;
52             file << record.returnTime << endl;
53         }
54     }
55 }
```

```

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

```

```

179     getch();
180 }
181
182 void EditBook() const
183 {
184     system("cls");
185     cout << endl;
186     cout << endl;
187     cout << "编辑书籍" << endl;
188     cout << endl;
189     cout << endl;
190     cout << "请输入书名: ";
191     string title;
192     getline(cin, title);
193     title = RemoveBlank(title);
194     cout << endl;
195     if (title.empty())
196     {
197         cout << "书名不能为空" << endl;
198         cout << endl;
199         cout << "按任意键返回" << endl;
200         getch();
201         return;
202     }
203     Book oldBook = getBook(title);
204     int result = oldBook.title.empty() ? 0 : 1;
205     switch (result)
206     {
207     case 0:
208         cout << "书籍不存在" << endl;
209         break;
210     case 1:
211         DisplayBook(oldBook);
212         Book book;
213         cout << endl;
214         cout << "请输入新书名: ";
215         getline(cin, book.title);
216         book.title = RemoveBlank(book.title);
217         if (book.title.empty())
218         {
219             cout << endl;
220             cout << "书名不能为空" << endl;
221             cout << endl;
222             cout << "按任意键返回" << endl;
223             getch();
224             return;
225         }
226         cout << "请输入新作者: ";
227         getline(cin, book.author);
228         book.author = RemoveBlank(book.author);
229         cout << "请输入新分类: ";
230         getline(cin, book.category);
231         book.category = RemoveBlank(book.category);
232         cout << "请输入新关键词: ";
233         getline(cin, book.keywords);
234         book.keywords = RemoveBlank(book.keywords);

```

```

235         cout << "请输入新简介: ";
236         getline(cin, book.summary);
237         book.summary = RemoveBlank(book.summary);
238         cout << endl;
239         book.isBorrowed = oldBook.isBorrowed;
240         book.borrowTimes = oldBook.borrowTimes;
241         oldBook.Delete();
242         int result = book.Edit();
243         switch (result)
244         {
245             case -1:
246                 cout << "保存失败" << endl;
247                 break;
248             case 1:
249                 cout << "保存成功" << endl;
250                 break;
251         }
252     }
253     cout << endl;
254     cout << "按任意键返回" << endl;
255     getch();
256 }
257
258 void DisplayUser(const User &user) const
259 {
260     cout << user;
261 }
262
263 void AddUser() const
264 {
265     system("cls");
266     User user;
267     cout << endl;
268     cout << endl;
269     cout << "添加用户" << endl;
270     cout << endl;
271     cout << endl;
272     cout << "请输入用户名: ";
273     getline(cin, user.name);
274     user.name = RemoveBlank(user.name);
275     cout << endl;
276     if (user.name.empty())
277     {
278         cout << "用户名不能为空" << endl;
279         cout << endl;
280         cout << "按任意键返回" << endl;
281         getch();
282         return;
283     }
284     int result = user.Add();
285     switch (result)
286     {
287         case 0:
288             cout << "用户已存在" << endl;
289             break;
290         case -1:

```

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

```

347 void SearchUser() const
348 {
349     system("cls");
350     cout << endl;
351     cout << endl;
352     cout << "查找用户" << endl;
353     cout << endl;
354     cout << endl;
355     cout << "请输入用户名（回车显示所有用户）： ";
356     string name;
357     getline(cin, name);
358     cout << endl;
359     cout << "查询结果" << endl;
360     cout << endl;
361     vector<User> users = searchUser(name);
362     int result = users.size() == 0 ? 0 : 1;
363     switch (result)
364     {
365     case 0:
366         cout << "用户不存在" << endl;
367         break;
368     case 1:
369         for (int i = 0; i < users.size(); i++)
370         {
371             cout << "用户" << i + 1 << ": " << users[i].name << endl;
372             cout << endl;
373         }
374     }
375     cout << endl;
376     cout << "按任意键返回" << endl;
377     getch();
378 }
379
380 void EditUser() const
381 {
382     system("cls");
383     cout << endl;
384     cout << endl;
385     cout << "编辑用户" << endl;
386     cout << endl;
387     cout << endl;
388     cout << "请输入用户名： ";
389     string oldname;
390     getline(cin, oldname);
391     oldname = RemoveBlank(oldname);
392     cout << endl;
393     if (oldname.empty())
394     {
395         cout << "用户名不能为空" << endl;
396         cout << endl;
397         cout << "按任意键返回" << endl;
398         getch();
399         return;
400     }
401     User oldUser = getUser(oldname);
402     int result = oldUser.name.empty() ? 0 : 1;

```

```

403         switch (result)
404         {
405             case 0:
406                 cout << "用户不存在" << endl;
407                 break;
408             case 1:
409                 cout << "请输入新用户名: ";
410                 User user;
411                 getline(cin, user.name);
412                 user.name = RemoveBlank(user.name);
413                 cout << endl;
414                 if (user.name.empty())
415                 {
416                     cout << "用户名不能为空" << endl;
417                     cout << endl;
418                     cout << "按任意键返回" << endl;
419                     getch();
420                     return;
421                 }
422                 oldUser.Delete();
423                 int result = user.Edit();
424                 switch (result)
425                 {
426                     case -1:
427                         cout << "保存失败" << endl;
428                         break;
429                     case 1:
430                         cout << "保存成功" << endl;
431                         break;
432                 }
433             }
434             cout << endl;
435             cout << "按任意键返回" << endl;
436             getch();
437     }
438
439     void BorrowBook() const
440     {
441         system("cls");
442         cout << endl;
443         cout << endl;
444         cout << "图书借阅" << endl;
445         cout << endl;
446         cout << endl;
447         cout << "请输入书名: ";
448         string title;
449         getline(cin, title);
450         title = RemoveBlank(title);
451         cout << endl;
452         if (title.empty())
453         {
454             cout << "书名不能为空" << endl;
455             cout << endl;
456             cout << "按任意键返回" << endl;
457             getch();
458             return;

```

```

459     }
460     cout << "请输入用户名: ";
461     string name;
462     getline(cin, name);
463     name = RemoveBlank(name);
464     cout << endl;
465     if (name.empty())
466     {
467         cout << "用户名不能为空" << endl;
468         cout << endl;
469         cout << "按任意键返回" << endl;
470         getch();
471         return;
472     }
473     int result = borrowBook(name, title);
474     switch (result)
475     {
476     case 0:
477         cout << "书籍不存在" << endl;
478         break;
479     case -1:
480         cout << "书籍已借出" << endl;
481         break;
482     case -2:
483         cout << "用户不存在" << endl;
484         break;
485     case 1:
486         cout << "图书借阅成功" << endl;
487         break;
488     }
489     cout << endl;
490     cout << "按任意键返回" << endl;
491     getch();
492 }
493
494 void ReturnBook() const
495 {
496     system("cls");
497     cout << endl;
498     cout << endl;
499     cout << "图书归还" << endl;
500     cout << endl;
501     cout << endl;
502     cout << "请输入书名: ";
503     string title;
504     getline(cin, title);
505     title = RemoveBlank(title);
506     cout << endl;
507     if (title.empty())
508     {
509         cout << "书名不能为空" << endl;
510         cout << endl;
511         cout << "按任意键返回" << endl;
512         getch();
513         return;
514     }

```



```

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

```

```
571     int result = user.name.empty() ? 0 : 1;
572     switch (result)
573     {
574     case 0:
575         cout << "用户不存在" << endl;
576         break;
577     case 1:
578         DisplayUser(user);
579     }
580     cout << endl;
581     cout << "按任意键返回" << endl;
582     getch();
583 }
584
585 void TenHotBooks() const
586 {
587     system("cls");
588     cout << endl;
589     cout << endl;
590     cout << "十大热门书籍" << endl;
591     cout << endl;
592     cout << endl;
593     vector<Book> books = tenHotBooks();
594     if (books.size() == 0)
595     {
596         cout << "无记录" << endl;
597         cout << endl;
598         cout << "按任意键返回" << endl;
599         getch();
600         return;
601     }
602     for (int i = 0; i < books.size(); i++)
603     {
604         cout << "书籍" << i + 1 << endl;
605         DisplayBook(books[i]);
606         cout << endl;
607     }
608     cout << endl;
609     cout << "按任意键返回" << endl;
610     getch();
611 }
612
613 void TenActiveUsers() const
614 {
615     system("cls");
616     cout << endl;
617     cout << endl;
618     cout << "十大活跃用户" << endl;
619     cout << endl;
620     cout << endl;
621     vector<User> users = tenActiveUsers();
622     if (users.size() == 0)
623     {
624         cout << "无记录" << endl;
625         cout << endl;
626         cout << "按任意键返回" << endl;
```

```

627         getch();
628         return;
629     }
630     for (int i = 0; i < users.size(); i++)
631     {
632         cout << "用户" << i + 1 << ": " << users[i].name << endl;
633         cout << "借阅次数: " << users[i].borrowTimes << endl;
634         cout << endl;
635     }
636     cout << endl;
637     cout << "按任意键返回" << endl;
638     getch();
639 }
640
641 void DeleteAllBooks() const
642 {
643     system("cls");
644     cout << endl;
645     cout << endl;
646     cout << "确认删除所有书籍? (y/n)";
647     string c;
648     getline(cin, c);
649     cout << endl;
650     if (c == "y")
651     {
652         int result = deleteAllBooks();
653         switch (result)
654         {
655             case 1:
656                 cout << "删除成功" << endl;
657                 break;
658             default:
659                 cout << "删除失败" << endl;
660                 break;
661         }
662     }
663     else
664     {
665         cout << "取消删除" << endl;
666     }
667     cout << endl;
668     cout << "按任意键返回" << endl;
669     getch();
670 }
671
672 void DeleteAllUsers() const
673 {
674     system("cls");
675     cout << endl;
676     cout << endl;
677     cout << "确认删除所有用户? (y/n)";
678     string c;
679     getline(cin, c);
680     cout << endl;
681     if (c == "y")
682     {

```

```

683         int result = deleteAllUsers();
684         switch (result)
685         {
686             case 1:
687                 cout << "删除成功" << endl;
688                 break;
689             default:
690                 cout << "删除失败" << endl;
691                 break;
692         }
693     }
694     else
695     {
696         cout << "取消删除" << endl;
697     }
698     cout << endl;
699     cout << "按任意键返回" << endl;
700     getch();
701 }
702
703 void Exit() const
704 {
705     exit(0);
706 }
707
708 void Error() const
709 {
710     cout << endl;
711     cout << "无效输入, 请重新输入" << endl;
712 }
713 };

```

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 }
```

