

```
1  #pragma once
2  #include <iconv.h>
3  #include <filesystem>
4  #include <algorithm>
5  #include <vector>
6  #include <fstream>
7  #include <iostream>
8  #include <conio.h>
9
10 using namespace std;
11
12 string utf8_to_gbk(const string &utf8_str)
13 {
14     iconv_t cd = iconv_open("GBK", "UTF-8");
15     if (cd == (iconv_t)-1)
16         return "";
17     size_t in_bytes_left = utf8_str.size();
18     size_t out_bytes_left = in_bytes_left * 2;
19     char *in_buf = const_cast<char *>(utf8_str.c_str());
20     char out_buf[out_bytes_left];
21     char *out_buf_start = out_buf;
22     size_t ret = iconv(cd, &in_buf, &in_bytes_left, &out_buf_start,
23 &out_bytes_left);
24     if (ret == (size_t)-1)
25     {
26         iconv_close(cd);
27         return "";
28     }
29     *out_buf_start = '\0';
30     iconv_close(cd);
31     return string(out_buf);
32 }
33
34 string gbk_to_utf8(const string &gbk_str)
35 {
36     iconv_t cd = iconv_open("UTF-8", "GBK");
37     if (cd == (iconv_t)-1)
38         return "";
39     size_t in_bytes_left = gbk_str.size();
40     size_t out_bytes_left = in_bytes_left * 2;
41     char *in_buf = const_cast<char *>(gbk_str.c_str());
42     char out_buf[out_bytes_left];
43     char *out_buf_start = out_buf;
44     size_t ret = iconv(cd, &in_buf, &in_bytes_left, &out_buf_start,
45 &out_bytes_left);
46     if (ret == (size_t)-1)
47     {
48         iconv_close(cd);
49         return "";
50     }
51     *out_buf_start = '\0';
52     iconv_close(cd);
53     return string(out_buf);
54 }
```

```

52 }
53
54 bool IsPureNumber(const string &input)
55 {
56     return all_of(input.begin(), input.end(), ::isdigit);
57 }
58
59 class Base
60 {
61 public:
62     virtual int Add() const = 0;
63     virtual void Save() const = 0;
64     virtual int Delete() const = 0;
65     virtual int Edit() const = 0;
66 };

```

book.hpp

```

1  #define FILESYSTEM_BOOK "./data/book/"
2  #include "base.hpp"
3
4  class Book : public Base
5  {
6  public:
7      string title;
8      string author;
9      string category;
10     string keywords;
11     string summary;
12     int borrowTimes = 0;
13     bool isBorrowed = false;
14
15     Book(string Title = "", string Author = "", string Category = "", string
Keywords = "", string Summary = "") : title(Title), author(Author),
category(Category), keywords(Keywords), summary(Summary) {}
16     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) {}
17     ~Book() {}
18
19     int Add() const override
20     {
21         string filePath = FILESYSTEM_BOOK + utf8_to_gbk(this->title) +
".txt";
22         if (ifstream(filePath))
23             return 0;
24         else
25         {
26             ofstream file(filePath);
27             if (!file)
28                 return -1;
29             else
30             {
31                 file << this->title << endl;
32                 file << this->author << endl;

```

```

33         file << this->category << endl;
34         file << this->keywords << endl;
35         file << this->summary << endl;
36         file << this->isBorrowed << endl;
37         file << this->borrowTimes << endl;
38         file.close();
39         return 1;
40     }
41 }
42 }
43
44 void Save() const override
45 {
46     string filePath = FILESYSTEM_BOOK + utf8_to_gbk(this->title) +
".txt";
47     ofstream file(filePath);
48     file << this->title << endl;
49     file << this->author << endl;
50     file << this->category << endl;
51     file << this->keywords << endl;
52     file << this->summary << endl;
53     file << this->isBorrowed << endl;
54     file << this->borrowTimes << endl;
55     file.close();
56 }
57
58 int Delete() const override
59 {
60     string filePath = FILESYSTEM_BOOK + utf8_to_gbk(this->title) +
".txt";
61     if (remove(filePath.c_str()) == 0)
62         return 1;
63     else
64         return -1;
65 }
66
67 int Edit() const override
68 {
69     ofstream file(FILESYSTEM_BOOK + utf8_to_gbk(this->title) + ".txt");
70     if (!file)
71         return -1;
72     else
73     {
74         this->Save();
75         file.close();
76         return 1;
77     }
78 }
79
80 friend ostream &operator<<(ostream &, const Book &);
81 };
82
83 ostream &operator<<(ostream &os, const Book &book)
84 {
85     os << "书名: " << book.title << endl;
86     os << "作者: " << book.author << endl;

```

```

87     os << "分类: " << book.category << endl;
88     os << "关键词: " << book.keywords << endl;
89     os << "简介: " << book.summary << endl;
90     if (book.isBorrowed)
91         os << "借出状态: 已借出" << endl;
92     else
93         os << "借出状态: 未借出" << endl;
94     os << "借出次数: " << book.borrowTimes << endl;
95     return os;
96 }

```

user.hpp

```

1  #define FILESYSTEM_USER "./data/user/"
2  #include "base.hpp"
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 }

```

```

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 }
```

bookmanager.hpp

```
1  #include "book.hpp"
2
3  class BookManager
4  {
5  public:
6      Book getBook(const string &title) const
7      {
8          string filePath = FILESYSTEM_BOOK + utf8_to_gbk(title) + ".txt";
9          if (!ifstream(filePath))
10         {
11             return Book();
12         }
13         else
14         {
15             ifstream file(filePath);
16             if (!file)
17             {
18                 return Book();
19             }
20             else
21             {
22                 Book book;
23                 getline(file, book.title);
24                 getline(file, book.author);
25                 getline(file, book.category);
26                 getline(file, book.keywords);
27                 getline(file, book.summary);
28                 string line;
29                 getline(file, line);
30                 book.isBorrowed = (line == "1");
31                 getline(file, line);
32                 book.borrowTimes = stoi(line);
33                 file.close();
34                 return book;
35             }
36         }
37     }
38
39     vector<Book> searchBook(const string &keyword) const
40     {
41         vector<Book> results;
42         for (const auto &entry :
filesystem::directory_iterator(FILESYSTEM_BOOK))
43         {
44             string filePath = entry.path().string();
45             filePath = utf8_to_gbk(filePath);
46             ifstream file(filePath);
47             if (file)
48             {
```

```

49         Book book;
50         getline(file, book.title);
51         getline(file, book.author);
52         getline(file, book.category);
53         getline(file, book.keywords);
54         getline(file, book.summary);
55         string line;
56         getline(file, line);
57         book.isBorrowed = (line == "1");
58         getline(file, line);
59         book.borrowTimes = stoi(line);
60         file.close();
61         if (book.title.find(keyword) != string::npos ||
62             book.author.find(keyword) != string::npos ||
63             book.category.find(keyword) != string::npos ||
64             book.keywords.find(keyword) != string::npos ||
65             book.summary.find(keyword) != string::npos)
66         {
67             results.push_back(book);
68         }
69     }
70 }
71 return results;
72 }
73
74 vector<Book> tenHotBooks() const
75 {
76     vector<Book> results;
77     for (const auto &entry :
filesystem::directory_iterator(FILESYSTEM_BOOK))
78     {
79         string filePath = entry.path().string();
80         filePath = utf8_to_gbk(filePath);
81         ifstream file(filePath);
82         if (file)
83         {
84             Book book;
85             getline(file, book.title);
86             getline(file, book.author);
87             getline(file, book.category);
88             getline(file, book.keywords);
89             getline(file, book.summary);
90             string line;
91             getline(file, line);
92             book.isBorrowed = (line == "1");
93             getline(file, line);
94             book.borrowTimes = stoi(line);
95             file.close();
96             if (book.borrowTimes > 0)
97             {
98                 results.push_back(book);
99             }
100         }
101     }
102     sort(results.begin(), results.end(), [](Book a, Book b)
103         { return a.borrowTimes > b.borrowTimes; });

```

```

104         if (results.size() > 10)
105         {
106             results.resize(10);
107         }
108         return results;
109     }
110
111     int deleteAllBooks() const
112     {
113         for (const auto &entry :
filesystem::directory_iterator(FILESYSTEM_BOOK))
114         {
115             string filePath = entry.path().string();
116             filePath = utf8_to_gbk(filePath);
117             remove(filePath.c_str());
118         }
119         return 1;
120     }
121 };

```

usermanager.hpp

```

1  #include "user.hpp"
2
3  class UserManager
4  {
5  public:
6      User getUser(const string &name) const
7      {
8          string filePath = FILESYSTEM_USER + name + ".txt";
9          filePath = utf8_to_gbk(filePath);
10         if (!ifstream(filePath))
11         {
12             return User();
13         }
14         else
15         {
16             ifstream file(filePath);
17             if (!file)
18             {
19                 return User();
20             }
21             else
22             {
23                 User user(name);
24                 string line;
25                 while (getline(file, line))
26                 {
27                     Record record;
28                     record.bookName = line;
29                     getline(file, line);
30                     record.borrowTime = line;
31                     getline(file, line);
32                     record.returnTime = line;
33                     getline(file, line);

```



```

34         record.isReturned = (line == "1");
35         user.borrowRecords.push_back(record);
36     }
37     user.borrowTimes = user.borrowRecords.size();
38     file.close();
39     return user;
40 }
41 }
42 }
43
44 vector<User> searchUser(const string &keyword) const
45 {
46     vector<User> results;
47     for (const auto &entry :
filesystem::directory_iterator(FILESYSTEM_USER))
48     {
49         string filePath = entry.path().string();
50         filePath = utf8_to_gbk(filePath);
51         ifstream file(filePath);
52         if (file)
53         {
54             User user;
55             user.name = gbk_to_utf8(filePath.substr(12, filePath.size()
- 16));
56             string line;
57             while (getline(file, line))
58             {
59                 Record record;
60                 record.bookName = line;
61                 getline(file, line);
62                 record.borrowTime = line;
63                 getline(file, line);
64                 record.returnTime = line;
65                 getline(file, line);
66                 record.isReturned = (line == "1");
67                 user.borrowRecords.push_back(record);
68             }
69             user.borrowTimes = user.borrowRecords.size();
70             file.close();
71             if (user.name.find(keyword) != string::npos)
72             {
73                 results.push_back(user);
74             }
75         }
76     }
77     return results;
78 }
79
80 vector<User> tenActiveUsers() const
81 {
82     vector<User> results;
83     for (const auto &entry :
filesystem::directory_iterator(FILESYSTEM_USER))
84     {
85         string filePath = entry.path().string();
86         filePath = utf8_to_gbk(filePath);

```

```

87         ifstream file(filePath);
88         if (file)
89         {
90             User user;
91             user.name = gbk_to_utf8(filePath.substr(12, filePath.size()
- 16));
92             string line;
93             while (getline(file, line))
94             {
95                 Record record;
96                 record.bookName = line;
97                 getline(file, line);
98                 record.borrowTime = line;
99                 getline(file, line);
100                 record.returnTime = line;
101                 getline(file, line);
102                 record.isReturned = (line == "1");
103                 user.borrowRecords.push_back(record);
104             }
105             user.borrowTimes = user.borrowRecords.size();
106             file.close();
107             if (user.borrowTimes > 0)
108             {
109                 results.push_back(user);
110             }
111         }
112     }
113     sort(results.begin(), results.end(), [](User a, User b)
114         { return a.borrowTimes > b.borrowTimes; });
115     if (results.size() > 10)
116     {
117         results.resize(10);
118     }
119     return results;
120 }
121
122 int deleteAllUsers() const
123 {
124     for (const auto &entry :
filesystem::directory_iterator(FILESYSTEM_USER))
125     {
126         string filePath = entry.path().string();
127         filePath = utf8_to_gbk(filePath);
128         remove(filePath.c_str());
129     }
130     return 1;
131 }
132 };

```

manager.hpp

```

1 #include "bookmanager.hpp"
2 #include "userManager.hpp"
3
4 class Manager : public BookManager, public UserManager

```

```

5 {
6 public:
7     string getCurrentDateTime() const
8     {
9         time_t now = time(0);
10        tm *ltm = localtime(&now);
11        char buffer[80];
12        strftime(buffer, sizeof(buffer), "%Y-%m-%d %H:%M:%S", ltm);
13        return buffer;
14    }
15
16    int borrowBook(const string &userName, const string &bookName) const
17    {
18        Book book = getBook(bookName);
19        if (book.title == "")
20            return 0;
21        User user = getUser(userName);
22        if (user.name == "")
23            return -2;
24        if (book.isBorrowed)
25            return -1;
26        Record record;
27        record.bookName = bookName;
28        record.borrowTime = getCurrentDateTime();
29        record.isReturned = false;
30        user.borrowRecords.push_back(record);
31        user.borrowTimes = user.borrowRecords.size();
32        user.Save();
33        book.isBorrowed = true;
34        book.borrowTimes++;
35        book.Save();
36        return 1;
37    }
38
39    int returnBook(const string &userName, const string &bookName) const
40    {
41        Book book = getBook(bookName);
42        if (book.title == "")
43            return 0;
44        User user = getUser(userName);
45        if (user.name == "")
46            return -2;
47        bool found = false;
48        for (auto &record : user.borrowRecords)
49        {
50            if (record.bookName == bookName && !record.isReturned)
51            {
52                record.returnTime = getCurrentDateTime();
53                record.isReturned = true;
54                user.Save();
55                book.isBorrowed = false;
56                book.Save();
57                return 1;
58            }
59        }
60        return -1;

```

```

61     }
62 };

```

gui.hpp

```

1  #include "manager.hpp"
2
3  class GUI : public Manager
4  {
5  public:
6      void ShowMenu() const
7      {
8          system("cls");
9          cout << endl;
10         cout << endl;
11         cout << "                图书管理系统" << endl;
12         cout << "-----" << endl;
13         cout << "1. 添加书籍          9. 图书借阅" << endl;
14         cout << "2. 删除书籍          10. 图书归还" << endl;
15         cout << "3. 查找书籍          11. 借阅记录" << endl;
16         cout << "4. 编辑书籍          12. 十大热门书籍" << endl;
17         cout << "5. 添加用户          13. 十大活跃用户" << endl;
18         cout << "6. 删除用户          14. 删除所有书籍" << endl;
19         cout << "7. 查找用户          15. 删除所有用户" << endl;
20         cout << "8. 编辑用户          16. 退出" << endl;
21         cout << "-----" << endl;
22         cout << endl;
23         cout << "请选择操作: ";
24     }
25
26     string RemoveBlank(const string &str) const
27     {
28         auto start = str.find_first_not_of(" \t\n\r\f\v");
29         if (start == string::npos)
30             return "";
31         auto end = str.find_last_not_of(" \t\n\r\f\v");
32         return str.substr(start, end - start + 1);
33     }
34
35     void DisplayBook(const Book &book) const
36     {
37         cout << book;
38     }
39
40     void AddBook() const
41     {
42         system("cls");
43         Book book;
44         cout << endl;
45         cout << endl;
46         cout << "添加书籍" << endl;
47         cout << endl;
48         cout << endl;
49         cout << "请输入书名: ";
50         getline(cin, book.title);

```

```

51     book.title = RemoveBlank(book.title);
52     if (book.title.empty())
53     {
54         cout << endl;
55         cout << "书名不能为空" << endl;
56         cout << endl;
57         cout << "按任意键返回" << endl;
58         getch();
59         return;
60     }
61     else if (getBook(book.title).title == book.title)
62     {
63         cout << "书籍已存在" << endl;
64         cout << endl;
65         cout << "按任意键返回" << endl;
66         getch();
67         return;
68     }
69     cout << "请输入作者: ";
70     getline(cin, book.author);
71     book.author = RemoveBlank(book.author);
72     cout << "请输入分类: ";
73     getline(cin, book.category);
74     book.category = RemoveBlank(book.category);
75     cout << "请输入关键词: ";
76     getline(cin, book.keywords);
77     book.keywords = RemoveBlank(book.keywords);
78     cout << "请输入简介: ";
79     getline(cin, book.summary);
80     book.summary = RemoveBlank(book.summary);
81     cout << endl;
82     int result = book.Add();
83     switch (result)
84     {
85     case 0:
86         cout << "书籍已存在" << endl;
87         break;
88     case -1:
89         cout << "保存失败" << endl;
90         break;
91     case 1:
92         cout << "保存成功" << endl;
93         break;
94     }
95     cout << endl;
96     cout << "按任意键返回" << endl;
97     getch();
98 }
99
100 void DeleteBook() const
101 {
102     system("cls");
103     cout << endl;
104     cout << endl;
105     cout << "删除书籍" << endl;
106     cout << endl;

```

```
107     cout << endl;
108     cout << "请输入书名: ";
109     string title;
110     getline(cin, title);
111     title = RemoveBlank(title);
112     cout << endl;
113     if (title.empty())
114     {
115         cout << "书名不能为空" << endl;
116         cout << endl;
117         cout << "按任意键返回" << endl;
118         getch();
119         return;
120     }
121     if (getBook(title).title.empty())
122     {
123         cout << "书籍不存在" << endl;
124         cout << endl;
125         cout << "按任意键返回" << endl;
126         getch();
127         return;
128     }
129     Book book(title);
130     cout << "确认删除? (y/n)";
131     string c;
132     getline(cin, c);
133     cout << endl;
134     if (c != "y")
135     {
136         cout << "取消删除" << endl;
137         cout << endl;
138         cout << "按任意键返回" << endl;
139         getch();
140         return;
141     }
142     int result = book.Delete();
143     switch (result)
144     {
145     case -1:
146         cout << "删除失败" << endl;
147         break;
148     case 1:
149         cout << "删除成功" << endl;
150         break;
151     }
152     cout << endl;
153     cout << "按任意键返回" << endl;
154     getch();
155 }
156
157 void SearchBook() const
158 {
159     system("cls");
160     cout << endl;
161     cout << endl;
162     cout << "查找书籍" << endl;
```

```

163     cout << endl;
164     cout << endl;
165     cout << "请输入搜索词（回车显示所有书籍）： ";
166     string title;
167     getline(cin, title);
168     title = RemoveBlank(title);
169     cout << endl;
170     cout << "查询结果" << endl;
171     cout << endl;
172     vector<Book> books = searchBook(title);
173     int result = books.size() == 0 ? 0 : 1;
174     switch (result)
175     {
176     case 0:
177         cout << "书籍不存在" << endl;
178         break;
179     case 1:
180         for (int i = 0; i < books.size(); i++)
181         {
182             cout << "书籍" << i + 1 << endl;
183             DisplayBook(books[i]);
184             cout << endl;
185         }
186     }
187     cout << endl;
188     cout << "按任意键返回" << endl;
189     getch();
190 }
191
192 void EditBook() const
193 {
194     system("cls");
195     cout << endl;
196     cout << endl;
197     cout << "编辑书籍" << endl;
198     cout << endl;
199     cout << endl;
200     cout << "请输入书名： ";
201     string title;
202     getline(cin, title);
203     title = RemoveBlank(title);
204     cout << endl;
205     if (title.empty())
206     {
207         cout << "书名不能为空" << endl;
208         cout << endl;
209         cout << "按任意键返回" << endl;
210         getch();
211         return;
212     }
213     Book oldBook = getBook(title);
214     int result = oldBook.title.empty() ? 0 : 1;
215     switch (result)
216     {
217     case 0:
218         cout << "书籍不存在" << endl;

```

```

219         break;
220     case 1:
221         DisplayBook(oldBook);
222         Book book;
223         cout << endl;
224         cout << "请输入新书名: ";
225         getline(cin, book.title);
226         book.title = RemoveBlank(book.title);
227         if (book.title.empty())
228         {
229             cout << endl;
230             cout << "书名不能为空" << endl;
231             cout << endl;
232             cout << "按任意键返回" << endl;
233             getch();
234             return;
235         }
236         cout << "请输入新作者: ";
237         getline(cin, book.author);
238         book.author = RemoveBlank(book.author);
239         cout << "请输入新分类: ";
240         getline(cin, book.category);
241         book.category = RemoveBlank(book.category);
242         cout << "请输入新关键词: ";
243         getline(cin, book.keywords);
244         book.keywords = RemoveBlank(book.keywords);
245         cout << "请输入新简介: ";
246         getline(cin, book.summary);
247         book.summary = RemoveBlank(book.summary);
248         cout << endl;
249         book.isBorrowed = oldBook.isBorrowed;
250         book.borrowTimes = oldBook.borrowTimes;
251         oldBook.Delete();
252         int result = book.Edit();
253         switch (result)
254         {
255             case -1:
256                 cout << "保存失败" << endl;
257                 break;
258             case 1:
259                 cout << "保存成功" << endl;
260                 break;
261         }
262     }
263     cout << endl;
264     cout << "按任意键返回" << endl;
265     getch();
266 }
267
268 void DisplayUser(const User &user) const
269 {
270     cout << user;
271 }
272
273 void AddUser() const
274 {

```



```

275     system("cls");
276     User user;
277     cout << endl;
278     cout << endl;
279     cout << "添加用户" << endl;
280     cout << endl;
281     cout << endl;
282     cout << "请输入用户名: ";
283     getline(cin, user.name);
284     user.name = RemoveBlank(user.name);
285     cout << endl;
286     if (user.name.empty())
287     {
288         cout << "用户名不能为空" << endl;
289         cout << endl;
290         cout << "按任意键返回" << endl;
291         getch();
292         return;
293     }
294     int result = user.Add();
295     switch (result)
296     {
297     case 0:
298         cout << "用户已存在" << endl;
299         break;
300     case -1:
301         cout << "保存失败" << endl;
302         break;
303     case 1:
304         cout << "保存成功" << endl;
305         break;
306     }
307     cout << endl;
308     cout << "按任意键返回" << endl;
309     getch();
310 }
311
312 void DeleteUser() const
313 {
314     system("cls");
315     cout << endl;
316     cout << endl;
317     cout << "删除用户" << endl;
318     cout << endl;
319     cout << endl;
320     cout << "请输入用户名: ";
321     string name;
322     getline(cin, name);
323     name = RemoveBlank(name);
324     cout << endl;
325     if (name.empty())
326     {
327         cout << "用户名不能为空" << endl;
328         cout << endl;
329         cout << "按任意键返回" << endl;
330         getch();

```

```

331         return;
332     }
333     if (getUser(name).name.empty())
334     {
335         cout << "用户不存在" << endl;
336         cout << endl;
337         cout << "按任意键返回" << endl;
338         getch();
339         return;
340     }
341     User user(name);
342     cout << "确认删除? (y/n)";
343     string c;
344     getline(cin, c);
345     cout << endl;
346     if (c != "y")
347     {
348         cout << "取消删除" << endl;
349         cout << endl;
350         cout << "按任意键返回" << endl;
351         getch();
352         return;
353     }
354     int result = user.Delete();
355     switch (result)
356     {
357     case 1:
358         cout << "删除成功" << endl;
359         break;
360     case -1:
361         cout << "删除失败" << endl;
362         break;
363     }
364     cout << endl;
365     cout << "按任意键返回" << endl;
366     getch();
367 }
368
369 void SearchUser() const
370 {
371     system("cls");
372     cout << endl;
373     cout << endl;
374     cout << "查找用户" << endl;
375     cout << endl;
376     cout << endl;
377     cout << "请输入用户名（回车显示所有用户）： ";
378     string name;
379     getline(cin, name);
380     cout << endl;
381     cout << "查询结果" << endl;
382     cout << endl;
383     vector<User> users = searchUser(name);
384     int result = users.size() == 0 ? 0 : 1;
385     switch (result)
386     {

```

```

387         case 0:
388             cout << "用户不存在" << endl;
389             break;
390         case 1:
391             for (int i = 0; i < users.size(); i++)
392             {
393                 cout << "用户" << i + 1 << ": " << users[i].name << endl;
394                 cout << endl;
395             }
396         }
397         cout << endl;
398         cout << "按任意键返回" << endl;
399         getch();
400     }
401
402     void EditUser() const
403     {
404         system("cls");
405         cout << endl;
406         cout << endl;
407         cout << "编辑用户" << endl;
408         cout << endl;
409         cout << endl;
410         cout << "请输入用户名: ";
411         string oldname;
412         getline(cin, oldname);
413         oldname = RemoveBlank(oldname);
414         cout << endl;
415         if (oldname.empty())
416         {
417             cout << "用户名不能为空" << endl;
418             cout << endl;
419             cout << "按任意键返回" << endl;
420             getch();
421             return;
422         }
423         User oldUser = getUser(oldname);
424         int result = oldUser.name.empty() ? 0 : 1;
425         switch (result)
426         {
427             case 0:
428                 cout << "用户不存在" << endl;
429                 break;
430             case 1:
431                 cout << "请输入新用户名: ";
432                 User user;
433                 getline(cin, user.name);
434                 user.name = RemoveBlank(user.name);
435                 cout << endl;
436                 if (user.name.empty())
437                 {
438                     cout << "用户名不能为空" << endl;
439                     cout << endl;
440                     cout << "按任意键返回" << endl;
441                     getch();
442                     return;

```

```

443     }
444     oldUser.Delete();
445     int result = user.Edit();
446     switch (result)
447     {
448     case -1:
449         cout << "保存失败" << endl;
450         break;
451     case 1:
452         cout << "保存成功" << endl;
453         break;
454     }
455 }
456 cout << endl;
457 cout << "按任意键返回" << endl;
458 getch();
459 }
460
461 void BorrowBook() const
462 {
463     system("cls");
464     cout << endl;
465     cout << endl;
466     cout << "图书借阅" << endl;
467     cout << endl;
468     cout << endl;
469     cout << "请输入书名: ";
470     string title;
471     getline(cin, title);
472     title = RemoveBlank(title);
473     cout << endl;
474     if (title.empty())
475     {
476         cout << "书名不能为空" << endl;
477         cout << endl;
478         cout << "按任意键返回" << endl;
479         getch();
480         return;
481     }
482     cout << "请输入用户名: ";
483     string name;
484     getline(cin, name);
485     name = RemoveBlank(name);
486     cout << endl;
487     if (name.empty())
488     {
489         cout << "用户名不能为空" << endl;
490         cout << endl;
491         cout << "按任意键返回" << endl;
492         getch();
493         return;
494     }
495     int result = borrowBook(name, title);
496     switch (result)
497     {
498     case 0:

```

```

499         cout << "书籍不存在" << endl;
500         break;
501     case -1:
502         cout << "书籍已借出" << endl;
503         break;
504     case -2:
505         cout << "用户不存在" << endl;
506         break;
507     case 1:
508         cout << "图书借阅成功" << endl;
509         break;
510 }
511 cout << endl;
512 cout << "按任意键返回" << endl;
513 getch();
514 }
515
516 void ReturnBook() const
517 {
518     system("cls");
519     cout << endl;
520     cout << endl;
521     cout << "图书归还" << endl;
522     cout << endl;
523     cout << endl;
524     cout << "请输入书名: ";
525     string title;
526     getline(cin, title);
527     title = RemoveBlank(title);
528     cout << endl;
529     if (title.empty())
530     {
531         cout << "书名不能为空" << endl;
532         cout << endl;
533         cout << "按任意键返回" << endl;
534         getch();
535         return;
536     }
537     cout << "请输入用户名: ";
538     string name;
539     getline(cin, name);
540     name = RemoveBlank(name);
541     cout << endl;
542     if (name.empty())
543     {
544         cout << "用户名不能为空" << endl;
545         cout << endl;
546         cout << "按任意键返回" << endl;
547         getch();
548         return;
549     }
550     int result = returnBook(name, title);
551     switch (result)
552     {
553     case 0:
554         cout << "书籍不存在" << endl;

```

```

555         break;
556     case -1:
557         cout << "未借此书籍" << endl;
558         break;
559     case -2:
560         cout << "用户不存在" << endl;
561         break;
562     case 1:
563         cout << "图书归还成功" << endl;
564         break;
565     }
566     cout << endl;
567     cout << "按任意键返回" << endl;
568     getch();
569 }
570
571 void BorrowRecord() const
572 {
573     system("cls");
574     cout << endl;
575     cout << endl;
576     cout << "借阅记录" << endl;
577     cout << endl;
578     cout << endl;
579     cout << "请输入用户名: ";
580     string name;
581     getline(cin, name);
582     name = RemoveBlank(name);
583     cout << endl;
584     if (name.empty())
585     {
586         cout << "用户名不能为空" << endl;
587         cout << endl;
588         cout << "按任意键返回" << endl;
589         getch();
590         return;
591     }
592     User user = getUser(name);
593     int result = user.name.empty() ? 0 : 1;
594     switch (result)
595     {
596     case 0:
597         cout << "用户不存在" << endl;
598         break;
599     case 1:
600         DisplayUser(user);
601     }
602     cout << endl;
603     cout << "按任意键返回" << endl;
604     getch();
605 }
606
607 void TenHotBooks() const
608 {
609     system("cls");
610     cout << endl;

```

```

611     cout << endl;
612     cout << "十大热门书籍" << endl;
613     cout << endl;
614     cout << endl;
615     vector<Book> books = tenHotBooks();
616     if (books.size() == 0)
617     {
618         cout << "无记录" << endl;
619         cout << endl;
620         cout << "按任意键返回" << endl;
621         getch();
622         return;
623     }
624     for (int i = 0; i < books.size(); i++)
625     {
626         cout << "书籍" << i + 1 << endl;
627         DisplayBook(books[i]);
628         cout << endl;
629     }
630     cout << endl;
631     cout << "按任意键返回" << endl;
632     getch();
633 }
634
635 void TenActiveUsers() const
636 {
637     system("cls");
638     cout << endl;
639     cout << endl;
640     cout << "十大活跃用户" << endl;
641     cout << endl;
642     cout << endl;
643     vector<User> users = tenActiveUsers();
644     if (users.size() == 0)
645     {
646         cout << "无记录" << endl;
647         cout << endl;
648         cout << "按任意键返回" << endl;
649         getch();
650         return;
651     }
652     for (int i = 0; i < users.size(); i++)
653     {
654         cout << "用户" << i + 1 << ": " << users[i].name << endl;
655         cout << "借阅次数: " << users[i].borrowTimes << endl;
656         cout << endl;
657     }
658     cout << endl;
659     cout << "按任意键返回" << endl;
660     getch();
661 }
662
663 void DeleteAllBooks() const
664 {
665     system("cls");
666     cout << endl;

```

```

667     cout << endl;
668     cout << "确认删除所有书籍? (y/n)";
669     string c;
670     getline(cin, c);
671     cout << endl;
672     if (c == "y")
673     {
674         int result = deleteAllBooks();
675         switch (result)
676         {
677             case 1:
678                 cout << "删除成功" << endl;
679                 break;
680             default:
681                 cout << "删除失败" << endl;
682                 break;
683         }
684     }
685     else
686     {
687         cout << "取消删除" << endl;
688     }
689     cout << endl;
690     cout << "按任意键返回" << endl;
691     getch();
692 }
693
694 void DeleteAllUsers() const
695 {
696     system("cls");
697     cout << endl;
698     cout << endl;
699     cout << "确认删除所有用户? (y/n)";
700     string c;
701     getline(cin, c);
702     cout << endl;
703     if (c == "y")
704     {
705         int result = deleteAllUsers();
706         switch (result)
707         {
708             case 1:
709                 cout << "删除成功" << endl;
710                 break;
711             default:
712                 cout << "删除失败" << endl;
713                 break;
714         }
715     }
716     else
717     {
718         cout << "取消删除" << endl;
719     }
720     cout << endl;
721     cout << "按任意键返回" << endl;
722     getch();

```



```

723     }
724
725     void Exit() const
726     {
727         exit(0);
728     }
729
730     void Error() const
731     {
732         cout << endl;
733         cout << "无效输入, 请重新输入" << endl;
734     }
735 };

```

library.hpp

```

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  int main()
4  {
5      Library library;
6      bool error = false;
7      system("chcp 65001");
8
9      while (true)
10     {
11         library.CheckDirectory();
12         library.ShowMenu();
13         if (error)
14             library.Error();
15
16         string input;
17         int choice;
18         getline(cin, input);
19         if (!IsPureNumber(input) ||
20             input.empty() ||
21             (choice = stoi(input)) > Exit || choice < AddBook)
22         {
23             error = true;
24             continue;
25         }
26
27         switch (choice)
28         {
29             case AddBook:
30                 library.AddBook();
31                 break;
32             case DeleteBook:
33                 library.DeleteBook();
34                 break;
35             case SearchBook:
36                 library.SearchBook();
37                 break;
38             case EditBook:
39                 library.EditBook();
40                 break;
41             case AddUser:
42                 library.AddUser();
43                 break;
44             case DeleteUser:
45                 library.DeleteUser();
46                 break;
47             case SearchUser:
48                 library.SearchUser();
49                 break;
50             case EditUser:
51                 library.EditUser();
52                 break;
53             case BorrowBook:
54                 library.BorrowBook();
55                 break;
56             case ReturnBook:
57                 library.ReturnBook();
58                 break;
```

```
59     case BorrowRecord:
60         library.BorrowRecord();
61         break;
62     case TenHotBooks:
63         library.TenHotBooks();
64         break;
65     case TenActiveUsers:
66         library.TenActiveUsers();
67         break;
68     case DeleteAllBooks:
69         library.DeleteAllBooks();
70         break;
71     case DeleteAllUsers:
72         library.DeleteAllUsers();
73         break;
74     case Exit:
75         library.Exit();
76     }
77     error = false;
78 }
79 return 0;
80 }
```

