图书管理系统

base.hpp

```
using namespace std;

class Base
{
public:
    virtual int Add() const = 0;
    virtual void Save() const = 0;
    virtual int Delete() const = 0;
    virtual int Edit() const = 0;
};
```

book.hpp

```
#define FILESYSTEM_BOOK "./data/book/"
#include <fstream>
#include <iconv.h>

string utf8_to_gbk(const string &utf8_str)
{
    iconv_t cd = iconv_open("GBK", "UTF-8");
    if (cd == (iconv_t)-1)
        return "";
    size_t in_bytes_left = utf8_str.size();
    size_t out_bytes_left = in_bytes_left * 2;
    char *in_buf = const_cast<char *>(utf8_str.c_str());
    char out_buf[out_bytes_left];
    char *out_buf_start = out_buf;
    size_t ret = iconv(cd, &in_buf, &in_bytes_left, &out_buf_start,
&out_bytes_left);
    if (ret == (size_t)-1)
    {
        iconv_close(cd);
        return "";
    }
    *out_buf_start = '\0';
    iconv_close(cd);
    return string(out_buf);
}

string gbk_to_utf8(const string &gbk_str)
{
    iconv_t cd = iconv_open("UTF-8", "GBK");
    if (cd == (iconv_t)-1)
        return "";
    size_t in_bytes_left = gbk_str.size();
    size_t out_bytes_left = in_bytes_left * 2;
    char *in_buf = const_cast<char *>(gbk_str.c_str());
    char out_buf[out_bytes_left];
    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) +
    ".txt";
90          ofstream file(filePath);
91          file << this->title << endl;
92          file << this->author << endl;
93          file << this->category << endl;
94          file << this->keywords << endl;
95          file << this->summary << endl;
96          file << this->isBorrowed << endl;
97          file << this->borrowTimes << endl;
98          file.close();
99      }
100
101     int Delete() const override
102     {
103         string filePath = FILESYSTEM_BOOK + utf8_to_gbk(this->title) +
    ".txt";
104         if (remove(filePath.c_str()) == 0)
105             return 1;
106         else
107             return -1;
108     }
109
110     int Edit() const override
111     {
112         ofstream file(FILESYSTEM_BOOK + utf8_to_gbk(this->title) + ".txt");
113         if (!file)
114             return -1;
115         else
116         {
117             this->Save();
118             file.close();
119             return 1;
120         }
121     }
122
123     friend ostream &operator<<(ostream &, const Book &);
124 };
125
126 ostream &operator<<(ostream &os, const Book &book)
127 {
128     os << "书名：" << book.title << endl;
129     os << "作者：" << book.author << endl;
130     os << "分类：" << book.category << endl;
131     os << "关键词：" << book.keywords << endl;
132     os << "简介：" << book.summary << endl;
133     if (book.isBorrowed)
134         os << "借出状态：已借出" << endl;
135     else
136         os << "借出状态：未借出" << endl;
137     os << "借出次数：" << book.borrowTimes << endl;
138     return os;
139 }
```

user.hpp

```
#define FILESYSTEM_USER "./data/user/"
#include <vector>

struct Record
{
    string bookName = "";
    string borrowTime = "";
    string returnTime = "";
    bool isReturned = false;
};

class User : public Base
{
public:
    string name;
    vector<Record> borrowRecords;
    int borrowTimes = 0;

    User(string Name = "") : name(Name) {}
    User(const User &user) : name(user.name), borrowRecords(user.borrowRecords), borrowTimes(user.borrowTimes) {}
    ~User() {}

    int Add() const override
    {
        string filePath = FILESYSTEM_USER + utf8_to_gbk(this->name) + ".txt";
        if (ifstream(filePath))
            return 0;
        else
        {
            ofstream file(filePath);
            if (!file)
                return -1;
            else
            {
                file.close();
                return 1;
            }
        }
    }

    void Save() const override
    {
        string filePath = FILESYSTEM_USER + utf8_to_gbk(this->name) + ".txt";
        ofstream file(filePath);
        for (auto record : this->borrowRecords)
        {
            file << record.bookName << endl;
            file << record.borrowTime << endl;
            file << record.returnTime << endl;
```

```
            file << record.isReturned << endl;
        }
        file.close();
    }

    int Delete() const override
    {
        string filePath = FILESYSTEM_USER + utf8_to_gbk(this->name) +
".txt";
        if (remove(filePath.c_str()) == 0)
            return 1;
        else
            return -1;
    }

    int Edit() const override
    {
        ofstream file(FILESYSTEM_USER + utf8_to_gbk(this->name) + ".txt");
        if (!file)
            return -1;
        else
        {
            this->Save();
            file.close();
            return 1;
        }
    }

    friend ostream &operator<<(ostream &, const User &);
};

ostream &operator<<(ostream &os, const User &user)
{
    os << "借阅次数：" << user.borrowTimes << endl;
    os << endl;
    os << "借阅记录：" << endl;
    os << endl;
    for (auto record : user.borrowRecords)
    {
        os << "书名：" << record.bookName << endl;
        os << "借书时间：" << record.borrowTime << endl;
        if (record.isReturned)
            os << "还书时间：" << record.returnTime << endl;
        else
            os << "还书时间：未还" << endl;
        os << endl;
    }
    return os;
}
```

manager.hpp

```
#define FILESYSTEM_USER "./data/user/"
#include <vector>

```

```
struct Record
{
    string bookName = "";
    string borrowTime = "";
    string returnTime = "";
    bool isReturned = false;
};

class User : public Base
{
public:
    string name;
    vector<Record> borrowRecords;
    int borrowTimes = 0;

    User(string Name = "") : name(Name) {}
    User(const User &user) : name(user.name), borrowRecords(user.borrowRecords), borrowTimes(user.borrowTimes) {}
    ~User() {}

    int Add() const override
    {
        string filePath = FILESYSTEM_USER + utf8_to_gbk(this->name) + ".txt";
        if (ifstream(filePath))
            return 0;
        else
        {
            ofstream file(filePath);
            if (!file)
                return -1;
            else
            {
                file.close();
                return 1;
            }
        }
    }

    void Save() const override
    {
        string filePath = FILESYSTEM_USER + utf8_to_gbk(this->name) + ".txt";
        ofstream file(filePath);
        for (auto record : this->borrowRecords)
        {
            file << record.bookName << endl;
            file << record.borrowTime << endl;
            file << record.returnTime << endl;
            file << record.isReturned << endl;
        }
        file.close();
    }

    int Delete() const override
    {
```

```
        string filePath = FILESYSTEM_USER + utf8_to_gbk(this->name) + ".txt";
        if (remove(filePath.c_str()) == 0)
            return 1;
        else
            return -1;
    }

    int Edit() const override
    {
        ofstream file(FILESYSTEM_USER + utf8_to_gbk(this->name) + ".txt");
        if (!file)
            return -1;
        else
        {
            this->Save();
            file.close();
            return 1;
        }
    }

    friend ostream &operator<<(ostream &, const User &);
};

ostream &operator<<(ostream &os, const User &user)
{
    os << "借阅次数：" << user.borrowTimes << endl;
    os << endl;
    os << "借阅记录：" << endl;
    os << endl;
    for (auto record : user.borrowRecords)
    {
        os << "书名：" << record.bookName << endl;
        os << "借书时间：" << record.borrowTime << endl;
        if (record.isReturned)
            os << "还书时间：" << record.returnTime << endl;
        else
            os << "还书时间：未还" << endl;
        os << endl;
    }
    return os;
}
```

gui.hpp

```
#include "manager.hpp"
#include <conio.h>
#include <iostream>

class GUI : public Manager
{
public:
    void ShowMenu() const
    {
        system("cls");
```

```
        cout << endl;
        cout << endl;
        cout << "                图书管理系统" << endl;
        cout << "-------------------------------------" << endl;
        cout << "1. 添加书籍             9. 图书借阅" << endl;
        cout << "2. 删除书籍             10. 图书归还" << endl;
        cout << "3. 查找书籍             11. 借阅记录" << endl;
        cout << "4. 编辑书籍             12. 十大热门书籍" << endl;
        cout << "5. 添加用户             13. 十大活跃用户" << endl;
        cout << "6. 删除用户             14. 删除所有书籍" << endl;
        cout << "7. 查找用户             15. 删除所有用户" << endl;
        cout << "8. 编辑用户             16. 退出" << endl;
        cout << "-------------------------------------" << endl;
        cout << endl;
        cout << "请选择操作：";
    }

    string RemoveBlank(const string &str) const
    {
        auto start = str.find_first_not_of(" \t\n\r\f\v");
        if (start == string::npos)
            return "";
        auto end = str.find_last_not_of(" \t\n\r\f\v");
        return str.substr(start, end - start + 1);
    }

    void DisplayBook(const Book &book) const
    {
        cout << book;
    }

    void AddBook() const
    {
        system("cls");
        Book book;
        cout << endl;
        cout << endl;
        cout << "添加书籍" << endl;
        cout << endl;
        cout << endl;
        cout << "请输入书名：";
        getline(cin, book.title);
        book.title = RemoveBlank(book.title);
        if (book.title.empty())
        {
            cout << endl;
            cout << "书名不能为空" << endl;
            cout << endl;
            cout << "按任意键返回" << endl;
            getch();
            return;
        }
        else if (getBook(book.title).title == book.title)
        {
            cout << "书籍已存在" << endl;
            cout << endl;
```

```
            cout << "按任意键返回" << endl;
            getch();
            return;
        }
        cout << "请输入作者：";
        getline(cin, book.author);
        book.author = RemoveBlank(book.author);
        cout << "请输入分类：";
        getline(cin, book.category);
        book.category = RemoveBlank(book.category);
        cout << "请输入关键词：";
        getline(cin, book.keywords);
        book.keywords = RemoveBlank(book.keywords);
        cout << "请输入简介：";
        getline(cin, book.summary);
        book.summary = RemoveBlank(book.summary);
        cout << endl;
        int result = book.Add();
        switch (result)
        {
        case 0:
            cout << "书籍已存在" << endl;
            break;
        case -1:
            cout << "保存失败" << endl;
            break;
        case 1:
            cout << "保存成功" << endl;
            break;
        }
        cout << endl;
        cout << "按任意键返回" << endl;
        getch();
    }

    void DeleteBook() const
    {
        system("cls");
        cout << endl;
        cout << endl;
        cout << "删除书籍" << endl;
        cout << endl;
        cout << endl;
        cout << "请输入书名：";
        string title;
        getline(cin, title);
        title = RemoveBlank(title);
        cout << endl;
        if (title.empty())
        {
            cout << "书名不能为空" << endl;
            cout << endl;
            cout << "按任意键返回" << endl;
            getch();
            return;
        }
```

```
        if (getBook(title).title.empty())
        {
            cout << "书籍不存在" << endl;
            cout << endl;
            cout << "按任意键返回" << endl;
            getch();
            return;
        }
        Book book(title);
        int result = book.Delete();
        switch (result)
        {
        case -1:
            cout << "删除失败" << endl;
            break;
        case 1:
            cout << "删除成功" << endl;
            break;
        }
        cout << endl;
        cout << "按任意键返回" << endl;
        getch();
    }

    void SearchBook() const
    {
        system("cls");
        cout << endl;
        cout << endl;
        cout << "查找书籍" << endl;
        cout << endl;
        cout << endl;
        cout << "请输入搜索词（回车显示所有书籍）: ";
        string title;
        getline(cin, title);
        title = RemoveBlank(title);
        cout << endl;
        cout << "查询结果" << endl;
        cout << endl;
        vector<Book> books = searchBook(title);
        int result = books.size() == 0 ? 0 : 1;
        switch (result)
        {
        case 0:
            cout << "书籍不存在" << endl;
            break;
        case 1:
            for (int i = 0; i < books.size(); i++)
            {
                cout << "书籍" << i + 1 << endl;
                DisplayBook(books[i]);
                cout << endl;
            }
        }
        cout << endl;
        cout << "按任意键返回" << endl;
```

```
        getch();
    }

    void EditBook() const
    {
        system("cls");
        cout << endl;
        cout << endl;
        cout << "编辑书籍" << endl;
        cout << endl;
        cout << endl;
        cout << "请输入书名：";
        string title;
        getline(cin, title);
        title = RemoveBlank(title);
        cout << endl;
        if (title.empty())
        {
            cout << "书名不能为空" << endl;
            cout << endl;
            cout << "按任意键返回" << endl;
            getch();
            return;
        }
        Book oldBook = getBook(title);
        int result = oldBook.title.empty() ? 0 : 1;
        switch (result)
        {
        case 0:
            cout << "书籍不存在" << endl;
            break;
        case 1:
            DisplayBook(oldBook);
            Book book;
            cout << endl;
            cout << "请输入新书名：";
            getline(cin, book.title);
            book.title = RemoveBlank(book.title);
            if (book.title.empty())
            {
                cout << endl;
                cout << "书名不能为空" << endl;
                cout << endl;
                cout << "按任意键返回" << endl;
                getch();
                return;
            }
            cout << "请输入新作者：";
            getline(cin, book.author);
            book.author = RemoveBlank(book.author);
            cout << "请输入新分类：";
            getline(cin, book.category);
            book.category = RemoveBlank(book.category);
            cout << "请输入新关键词：";
            getline(cin, book.keywords);
            book.keywords = RemoveBlank(book.keywords);
```

```
            cout << "请输入新简介：";
            getline(cin, book.summary);
            book.summary = RemoveBlank(book.summary);
            cout << endl;
            book.isBorrowed = oldBook.isBorrowed;
            book.borrowTimes = oldBook.borrowTimes;
            oldBook.Delete();
            int result = book.Edit();
            switch (result)
            {
            case -1:
                cout << "保存失败" << endl;
                break;
            case 1:
                cout << "保存成功" << endl;
                break;
            }
        }
        cout << endl;
        cout << "按任意键返回" << endl;
        getch();
    }

    void DisplayUser(const User &user) const
    {
        cout << user;
    }

    void AddUser() const
    {
        system("cls");
        User user;
        cout << endl;
        cout << endl;
        cout << "添加用户" << endl;
        cout << endl;
        cout << endl;
        cout << "请输入用户名：";
        getline(cin, user.name);
        user.name = RemoveBlank(user.name);
        cout << endl;
        if (user.name.empty())
        {
            cout << "用户名不能为空" << endl;
            cout << endl;
            cout << "按任意键返回" << endl;
            getch();
            return;
        }
        int result = user.Add();
        switch (result)
        {
        case 0:
            cout << "用户已存在" << endl;
            break;
        case -1:
```

```
            cout << "保存失败" << endl;
            break;
        case 1:
            cout << "保存成功" << endl;
            break;
        }
        cout << endl;
        cout << "按任意键返回" << endl;
        getch();
    }

    void DeleteUser() const
    {
        system("cls");
        cout << endl;
        cout << endl;
        cout << "删除用户" << endl;
        cout << endl;
        cout << endl;
        cout << "请输入用户名：";
        string name;
        getline(cin, name);
        name = RemoveBlank(name);
        cout << endl;
        if (name.empty())
        {
            cout << "用户名不能为空" << endl;
            cout << endl;
            cout << "按任意键返回" << endl;
            getch();
            return;
        }
        if (getUser(name).name.empty())
        {
            cout << "用户不存在" << endl;
            cout << endl;
            cout << "按任意键返回" << endl;
            getch();
            return;
        }
        User user(name);
        int result = user.Delete();
        switch (result)
        {
        case 1:
            cout << "删除成功" << endl;
            break;
        case -1:
            cout << "删除失败" << endl;
            break;
        }
        cout << endl;
        cout << "按任意键返回" << endl;
        getch();
    }

```

```
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);
```

```
        int result = user.name.empty() ? 0 : 1;
        switch (result)
        {
        case 0:
            cout << "用户不存在" << endl;
            break;
        case 1:
            DisplayUser(user);
        }
        cout << endl;
        cout << "按任意键返回" << endl;
        getch();
    }

    void TenHotBooks() const
    {
        system("cls");
        cout << endl;
        cout << endl;
        cout << "十大热门书籍" << endl;
        cout << endl;
        cout << endl;
        vector<Book> books = tenHotBooks();
        if (books.size() == 0)
        {
            cout << "无记录" << endl;
            cout << endl;
            cout << "按任意键返回" << endl;
            getch();
            return;
        }
        for (int i = 0; i < books.size(); i++)
        {
            cout << "书籍" << i + 1 << endl;
            DisplayBook(books[i]);
            cout << endl;
        }
        cout << endl;
        cout << "按任意键返回" << endl;
        getch();
    }

    void TenActiveUsers() const
    {
        system("cls");
        cout << endl;
        cout << endl;
        cout << "十大活跃用户" << endl;
        cout << endl;
        cout << endl;
        vector<User> users = tenActiveUsers();
        if (users.size() == 0)
        {
            cout << "无记录" << endl;
            cout << endl;
            cout << "按任意键返回" << endl;
```

```
            getch();
            return;
        }
        for (int i = 0; i < users.size(); i++)
        {
            cout << "用户" << i + 1 << ": " << users[i].name << endl;
            cout << "借阅次数: " << users[i].borrowTimes << endl;
            cout << endl;
        }
        cout << endl;
        cout << "按任意键返回" << endl;
        getch();
    }

    void DeleteAllBooks() const
    {
        system("cls");
        cout << endl;
        cout << endl;
        cout << "确认删除所有书籍? (y/n)";
        string c;
        getline(cin, c);
        cout << endl;
        if (c == "y")
        {
            int result = deleteAllBooks();
            switch (result)
            {
            case 1:
                cout << "删除成功" << endl;
                break;
            default:
                cout << "删除失败" << endl;
                break;
            }
        }
        else
        {
            cout << "取消删除" << endl;
        }
        cout << endl;
        cout << "按任意键返回" << endl;
        getch();
    }

    void DeleteAllUsers() const
    {
        system("cls");
        cout << endl;
        cout << endl;
        cout << "确认删除所有用户? (y/n)";
        string c;
        getline(cin, c);
        cout << endl;
        if (c == "y")
        {
```

```
            int result = deleteAllUsers();
            switch (result)
            {
            case 1:
                cout << "删除成功" << endl;
                break;
            default:
                cout << "删除失败" << endl;
                break;
            }
        }
        else
        {
            cout << "取消删除" << endl;
        }
        cout << endl;
        cout << "按任意键返回" << endl;
        getch();
    }

    void Exit() const
    {
        exit(0);
    }

    void Error() const
    {
        cout << endl;
        cout << "无效输入，请重新输入" << endl;
    }
};
```

library.hpp

```
#include "gui.hpp"

class Library : public GUI
{
public:
    void CheckDirectory()
    {
        if (!filesystem::exists(FILESYSTEM_BOOK))
            filesystem::create_directories(FILESYSTEM_BOOK);
        if (!filesystem::exists(FILESYSTEM_USER))
            filesystem::create_directories(FILESYSTEM_USER);
    }
};
```

main.cpp

```
#include "library.hpp"

enum Choice
{
```

```
    AddBook = 1,
    DeleteBook,
    SearchBook,
    EditBook,
    AddUser,
    DeleteUser,
    SearchUser,
    EditUser,
    BorrowBook,
    ReturnBook,
    BorrowRecord,
    TenHotBooks,
    TenActiveUsers,
    DeleteAllBooks,
    DeleteAllUsers,
    Exit
};

bool IsPureNumber(const string &input)
{
    return all_of(input.begin(), input.end(), ::isdigit);
}

int main()
{
    Library library;
    bool error = false;
    system("chcp 65001");

    while (true)
    {
        library.CheckDirectory();
        library.ShowMenu();
        if (error)
            library.Error();

        string input;
        int choice;
        getline(cin, input);
        if (!IsPureNumber(input) ||
            input.empty() ||
            (choice = stoi(input)) > Exit || choice < AddBook)
        {
            error = true;
            continue;
        }

        switch (choice)
        {
        case AddBook:
            library.AddBook();
            break;
        case DeleteBook:
            library.DeleteBook();
            break;
        case SearchBook:
```

```
            library.SearchBook();
            break;
        case EditBook:
            library.EditBook();
            break;
        case AddUser:
            library.AddUser();
            break;
        case DeleteUser:
            library.DeleteUser();
            break;
        case SearchUser:
            library.SearchUser();
            break;
        case EditUser:
            library.EditUser();
            break;
        case BorrowBook:
            library.BorrowBook();
            break;
        case ReturnBook:
            library.ReturnBook();
            break;
        case BorrowRecord:
            library.BorrowRecord();
            break;
        case TenHotBooks:
            library.TenHotBooks();
            break;
        case TenActiveUsers:
            library.TenActiveUsers();
            break;
        case DeleteAllBooks:
            library.DeleteAllBooks();
            break;
        case DeleteAllUsers:
            library.DeleteAllUsers();
            break;
        case Exit:
            library.Exit();
        }
        error = false;
    }
    return 0;
}
```