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;
    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);
}

class Book : public Base
{
public:
    string title;
    string author;
    string category;
    string keywords;
    string summary;
    int borrowTimes = 0;
    bool isBorrowed = false;

    Book(string Title = "", string Author = "", string Category = "", string Keywords = "", string Summary = "") : title(Title), author(Author), category(Category), keywords(Keywords), summary(Summary) {}
    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) {}
    ~Book() {}

    int Add() const override
    {
        string filePath = FILESYSTEM_BOOK + utf8_to_gbk(this->title) + ".txt";
        if (ifstream(filePath))
            return 0;
        else
        {
            ofstream file(filePath);
            if (!file)
                return -1;
            else
            {
                file << this->title << endl;
                file << this->author << endl;
                file << this->category << endl;
                file << this->keywords << endl;
                file << this->summary << endl;
                file << this->isBorrowed << endl;
                file << this->borrowTimes << endl;
                file.close();
                return 1;
            }
        }
    }

    void Save() const override
    {
```

```
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

```cpp
#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

```
#include "base.hpp"
#include "book.hpp"
#include "user.hpp"
#include <filesystem>
#include <algorithm>

```

```
class BookManager
{
public:
    Book getBook(const string &title) const
    {
        string filePath = FILESYSTEM_BOOK + utf8_to_gbk(title) + ".txt";
        if (!ifstream(filePath))
        {
            return Book();
        }
        else
        {
            ifstream file(filePath);
            if (!file)
            {
                return Book();
            }
            else
            {
                Book book;
                getline(file, book.title);
                getline(file, book.author);
                getline(file, book.category);
                getline(file, book.keywords);
                getline(file, book.summary);
                string line;
                getline(file, line);
                book.isBorrowed = (line == "1");
                getline(file, line);
                book.borrowTimes = stoi(line);
                file.close();
                return book;
            }
        }
    }

    vector<Book> searchBook(const string &keyword) const
    {
        vector<Book> results;
        for (const auto &entry :
filesystem::directory_iterator(FILESYSTEM_BOOK))
        {
            string filePath = entry.path().string();
            filePath = utf8_to_gbk(filePath);
            ifstream file(filePath);
            if (file)
            {
                Book book;
                getline(file, book.title);
                getline(file, book.author);
                getline(file, book.category);
                getline(file, book.keywords);
                getline(file, book.summary);
                string line;
                getline(file, line);
                book.isBorrowed = (line == "1");
```

```
                getline(file, line);
                book.borrowTimes = stoi(line);
                file.close();
                if (book.title.find(keyword) != string::npos ||
                    book.author.find(keyword) != string::npos ||
                    book.category.find(keyword) != string::npos ||
                    book.keywords.find(keyword) != string::npos ||
                    book.summary.find(keyword) != string::npos)
                {
                    results.push_back(book);
                }
            }
        }
        return results;
    }

    vector<Book> tenHotBooks() const
    {
        vector<Book> results;
        for (const auto &entry :
filesystem::directory_iterator(FILESYSTEM_BOOK))
        {
            string filePath = entry.path().string();
            filePath = utf8_to_gbk(filePath);
            ifstream file(filePath);
            if (file)
            {
                Book book;
                getline(file, book.title);
                getline(file, book.author);
                getline(file, book.category);
                getline(file, book.keywords);
                getline(file, book.summary);
                string line;
                getline(file, line);
                book.isBorrowed = (line == "1");
                getline(file, line);
                book.borrowTimes = stoi(line);
                file.close();
                if (book.borrowTimes > 0)
                {
                    results.push_back(book);
                }
            }
        }
        sort(results.begin(), results.end(), [](Book a, Book b)
             { return a.borrowTimes > b.borrowTimes; });
        if (results.size() > 10)
        {
            results.resize(10);
        }
        return results;
    }

    int deleteAllBooks() const
    {
```

```
        for (const auto &entry : filesystem::directory_iterator(FILESYSTEM_BOOK))
        {
            string filePath = entry.path().string();
            filePath = utf8_to_gbk(filePath);
            remove(filePath.c_str());
        }
        return 1;
    }
};

class UserManager
{
public:
    User getUser(const string &name) const
    {
        string filePath = FILESYSTEM_USER + name + ".txt";
        filePath = utf8_to_gbk(filePath);
        if (!ifstream(filePath))
        {
            return User();
        }
        else
        {
            ifstream file(filePath);
            if (!file)
            {
                return User();
            }
            else
            {
                User user(name);
                string line;
                while (getline(file, line))
                {
                    Record record;
                    record.bookName = line;
                    getline(file, line);
                    record.borrowTime = line;
                    getline(file, line);
                    record.returnTime = line;
                    getline(file, line);
                    record.isReturned = (line == "1");
                    user.borrowRecords.push_back(record);
                }
                user.borrowTimes = user.borrowRecords.size();
                file.close();
                return user;
            }
        }
    }

    vector<User> searchUser(const string &keyword) const
    {
        vector<User> results;
```

```
        for (const auto &entry : filesystem::directory_iterator(FILESYSTEM_USER))
        {
            string filePath = entry.path().string();
            filePath = utf8_to_gbk(filePath);
            ifstream file(filePath);
            if (file)
            {
                User user;
                user.name = gbk_to_utf8(filePath.substr(12, filePath.size() - 16));
                string line;
                while (getline(file, line))
                {
                    Record record;
                    record.bookName = line;
                    getline(file, line);
                    record.borrowTime = line;
                    getline(file, line);
                    record.returnTime = line;
                    getline(file, line);
                    record.isReturned = (line == "1");
                    user.borrowRecords.push_back(record);
                }
                user.borrowTimes = user.borrowRecords.size();
                file.close();
                if (user.name.find(keyword) != string::npos)
                {
                    results.push_back(user);
                }
            }
        }
        return results;
    }

    vector<User> tenActiveUsers() const
    {
        vector<User> results;
        for (const auto &entry : filesystem::directory_iterator(FILESYSTEM_USER))
        {
            string filePath = entry.path().string();
            filePath = utf8_to_gbk(filePath);
            ifstream file(filePath);
            if (file)
            {
                User user;
                user.name = gbk_to_utf8(filePath.substr(12, filePath.size() - 16));
                string line;
                while (getline(file, line))
                {
                    Record record;
                    record.bookName = line;
                    getline(file, line);
                    record.borrowTime = line;
```

```
                    getline(file, line);
                    record.returnTime = line;
                    getline(file, line);
                    record.isReturned = (line == "1");
                    user.borrowRecords.push_back(record);
                }
                user.borrowTimes = user.borrowRecords.size();
                file.close();
                if (user.borrowTimes > 0)
                {
                    results.push_back(user);
                }
            }
        }
        sort(results.begin(), results.end(), [](User a, User b)
             { return a.borrowTimes > b.borrowTimes; });
        if (results.size() > 10)
        {
            results.resize(10);
        }
        return results;
    }

    int deleteAllUsers() const
    {
        for (const auto &entry :
filesystem::directory_iterator(FILESYSTEM_USER))
        {
            string filePath = entry.path().string();
            filePath = utf8_to_gbk(filePath);
            remove(filePath.c_str());
        }
        return 1;
    }
};

class Manager : public BookManager, public UserManager
{
public:
    string getCurrentDateTime() const
    {
        time_t now = time(0);
        tm *ltm = localtime(&now);
        char buffer[80];
        strftime(buffer, sizeof(buffer), "%Y-%m-%d %H:%M:%S", ltm);
        return buffer;
    }

    int borrowBook(const string &userName, const string &bookName) const
    {
        Book book = getBook(bookName);
        if (book.title == "")
            return 0;
        User user = getUser(userName);
        if (user.name == "")
            return -2;
```

```
        if (book.isBorrowed)
            return -1;
        Record record;
        record.bookName = bookName;
        record.borrowTime = getCurrentDateTime();
        record.isReturned = false;
        user.borrowRecords.push_back(record);
        user.borrowTimes = user.borrowRecords.size();
        user.Save();
        book.isBorrowed = true;
        book.borrowTimes++;
        book.Save();
        return 1;
    }

    int returnBook(const string &userName, const string &bookName) const
    {
        Book book = getBook(bookName);
        if (book.title == "")
            return 0;
        User user = getUser(userName);
        if (user.name == "")
            return -2;
        bool found = false;
        for (auto &record : user.borrowRecords)
        {
            if (record.bookName == bookName && !record.isReturned)
            {
                record.returnTime = getCurrentDateTime();
                record.isReturned = true;
                user.Save();
                book.isBorrowed = false;
                book.Save();
                return 1;
            }
        }
        return -1;
    }
};
```

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;
```

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

```
            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);
        cout << "确认删除？(y/n)";
        string c;
        getline(cin, c);
        cout << endl;
        if (c != "y")
        {
```

```
            cout << "取消删除" << endl;
            cout << endl;
            cout << "按任意键返回" << endl;
            getch();
            return;
        }
        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();
    }

    void SearchUser() const
    {
        system("cls");
        cout << endl;
        cout << endl;
        cout << "查找用户" << endl;
        cout << endl;
        cout << endl;
        cout << "请输入用户名（回车显示所有用户）: ";
        string name;
        getline(cin, name);
        cout << endl;
        cout << "查询结果" << endl;
        cout << endl;
        vector<User> users = searchUser(name);
        int result = users.size() == 0 ? 0 : 1;
        switch (result)
        {
        case 0:
            cout << "用户不存在" << endl;
            break;
        case 1:
            for (int i = 0; i < users.size(); i++)
            {
                cout << "用户" << i + 1 << ": " << users[i].name << endl;
                cout << endl;
            }
        }
        cout << endl;
        cout << "按任意键返回" << endl;
        getch();
    }

    void EditUser() const
    {
```

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

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

```
    void ReturnBook() const
    {
        system("cls");
        cout << endl;
        cout << endl;
        cout << "图书归还" << 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;
        }
        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;
        }
        int result = returnBook(name, title);
        switch (result)
        {
        case 0:
            cout << "书籍不存在" << endl;
            break;
        case -1:
            cout << "未借此书籍" << endl;
            break;
        case -2:
            cout << "用户不存在" << endl;
            break;
        case 1:
            cout << "图书归还成功" << endl;
            break;
        }
        cout << endl;
        cout << "按任意键返回" << endl;
        getch();
    }

    void BorrowRecord() const
```

```
    {
        system("cls");
        cout << endl;
        cout << endl;
        cout << "借阅记录" << 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;
        }
        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

```cpp
#include "gui.hpp"

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

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

```cpp
#include "library.hpp"

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();
```

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

```
        case DeleteAllUsers:
            library.DeleteAllUsers();
            break;
        case Exit:
            library.Exit();
        }
        error = false;
    }
    return 0;
}
```