## 问题描述

**设计一个图书管理系统设计，具体说明如下：**

**一、系统概述**

图书管理系统旨在为图书馆或相关机构提供高效便捷的图书信息管理服务。系统借助二进制文件存储图书的核心信息，涵盖图书编号、图书名称和图书价格，运用字符串流技术实现内存数据的灵活格式化，能够满足用户添加新记录、随机读取指定记录、修改图书价格以及查看所有记录、搜索图书等多样化需求，确保图书信息管理的准确性和高效性。

**二、功能设计**

（一）添加新记录

用户选择添加记录功能后，系统提示用户依次输入图书编号、图书名称和图书价格。程序创建 Book 类对象，通过对象的 input 方法接收用户输入的数据。随后，系统以追加模式打开二进制文件，将新的图书信息以二进制形式写入文件末尾，完成记录添加操作，并在成功写入后提示用户 “记录添加成功”。

（二）随机读取指定位置的记录

当用户选择读取记录功能时，系统要求用户输入想要读取的记录位置。程序根据用户输入的位置信息，计算出该记录在二进制文件中的偏移量，使用文件流的 seekg 函数将文件指针定位到相应位置。若成功读取到数据，则将读取到的图书信息转换为 Book 类对象，并通过 format 方法将其格式化为易读的字符串形式输出；若读取失败，则提示用户 “记录不存在”。

（三）修改任意记录的价格

用户选择修改价格功能后，首先需输入要修改价格的记录位置，接着输入新的价格。程序定位到指定位置读取图书记录，将新价格更新到 Book 对象中，再通过文件流的 seekp 函数将文件指针重新定位到该记录位置，将修改后的图书信息写回文件，最后提示用户 “修改成功”。

（四）显示所有记录

用户选择显示全部功能时，系统从二进制文件开头开始读取数据，每次读取一个图书记录的字节数，将其转换为 Book 类对象。在读取过程中，使用循环不断读取直到文件末尾，每读取一条记录，就通过 format 方法将其格式化并输出，同时记录已读取的记录数量。若文件中无记录，则提示用户 “没有记录”。

（五）搜索图书

用户选择搜索图书功能后，系统提示用户输入搜索关键词。程序从二进制文件开头开始读取数据，每次读取一个图书记录的字节数，将其转换为 Book 类对象。将图书的编号和名称组合成一个字符串，使用 find 方法查找关键词是否存在于该字符串中。若找到匹配的记录，则通过 format 方法将其格式化并输出，同时记录匹配的记录数量。若未找到匹配的记录，则提示用户 “未找到匹配的图书记录”。

**三、类设计**

（一）Book 类

数据成员：包含 id（图书编号）、title（图书名称）和 price（图书价格），分别用于存储图书的编号、名称和价格信息。

成员函数：

input 函数：通过标准输入流接收用户输入的图书编号、图书名称和图书价格数据，并进行相应的输入处理，如清除输入缓冲区残留数据，确保输入准确性。

format 函数：利用 stringstream 字符串流技术，将图书的编号、名称和价格信息格式化为规范的字符串形式，便于在界面展示和输出。

（二）BookFileManager 类

数据成员：filename 用于存储存储图书信息的二进制文件名，在类的构造函数中进行初始化。

成员函数：

addRecord 函数：实现添加新记录功能，创建 Book 对象接收用户输入，将数据写入二进制文件。

readRecord 函数：根据用户指定位置读取图书记录，成功读取后格式化输出，失败则提示错误。

modifyPrice 函数：完成指定位置记录的价格修改操作，读取记录、更新价格并写回文件。

displayAll 函数：遍历二进制文件，读取所有图书记录并格式化输出，若文件为空则给出提示。

searchBooks 函数：根据用户输入的关键词搜索图书记录，将匹配的记录格式化输出，若未找到匹配记录则给出提示。

（三）MenuSystem 类

数据成员：包含 BookFileManager 类对象，用于调用各类文件操作功能。

成员函数：

showMenu 函数：在控制台输出系统功能菜单，向用户展示可选择的操作选项。

run 函数：通过循环接收用户输入的操作选择，使用 switch 语句根据不同选择调用 BookFileManager 类的相应功能函数，实现系统的交互操作，直到用户选择退出系统。

**四、数据存储设计**

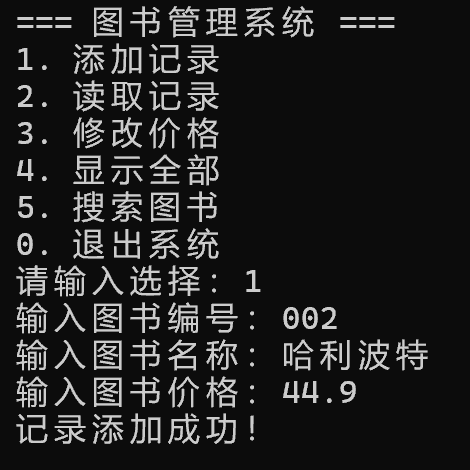
系统采用二进制文件存储图书信息，每个图书记录占用固定的字节数，由图书编号（整数类型占用字节数）、图书名称（字符数组固定长度 50 字节）和图书价格（浮点数类型占用字节数）组成。这种存储方式能够有效节省存储空间，同时方便进行随机访问和数据修改操作，确保数据的存储和读取高效准确。

## 实现代码

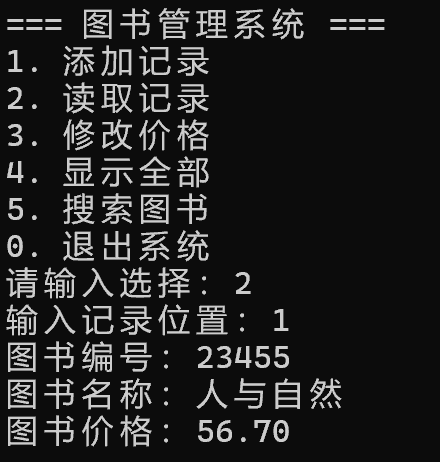
|  |  |
| --- | --- |
| 001  002  003  004  005  006  007  008  009  010  011  012  013  014  015  016  017  018  019  020  021  022  023  024  025  026  027  028  029  030  031  032  033  034  035  036  037  038  039  040  041  042  043  044  045  046  047  048  049  050  051  052  053  054  055  056  057  058  059  060  061  062  063  064  065  066  067  068  069  070  071  072  073  074  075  076  077  078  079  080  081  082  083  084  085  086  087  088  089  090  091  092  093  094  095  096  097  098  099  100  101  102  103  104  105  106  107  108  109  110  111  112  113  114  115  116  117  118  119  120  121  122  123  124  125  126  127  128  129  130  131  132  133  134  135  136  137  138  139  140  141  142  143  144  145  146  147  148  149  150  151  152  153  154  155  156  157  158  159  160  161  162  163  164  165  166  167  168  169  170  171  172  173  174  175  176  177  178  179  180  181  182  183  184  185  186  187  188  189  190  191  192  193  194  195  196  197  198 | #include <iostream>  #include <fstream>  #include <sstream>  #include <iomanip>  #include <string>  **using** **namespace** std;    **class** Book {  **public**:  **int** id;  **char** title[50];  **float** price;    **void** input() {          cout << "输入图书编号：";          cin >> id;          cout << "输入图书名称：";          cin.ignore();          cin.getline(title, 50);          cout << "输入图书价格：";          cin >> price;      }        string format() **const** {          stringstream ss;          ss << "图书编号：" << id << "\n图书名称：" << title             << "\n图书价格：" << fixed << setprecision(2) << price;  **return** ss.str();      }  };    **class** BookFileManager {      string filename;    **public**:      BookFileManager(**const** string& fname) : filename(fname) {}    **void** addRecord() {          ofstream outFile(filename, ios::binary | ios::app);          Book b;          b.input();    **if** (outFile.write(**reinterpret\_cast**<**char**\*>(&b), **sizeof**(Book))) {              cout << "记录添加成功！" << endl;          } **else** {              cerr << "写入失败！" << endl;          }      }    **void** readRecord(**int** pos) {          ifstream inFile(filename, ios::binary);  **if** (!inFile) {              cerr << "文件不存在！" << endl;  **return**;          }            inFile.seekg((pos-1)\***sizeof**(Book));          Book b;  **if** (inFile.read(**reinterpret\_cast**<**char**\*>(&b), **sizeof**(Book))) {              cout << b.format() << endl;          } **else** {              cerr << "记录不存在！" << endl;          }      }    **void** modifyPrice(**int** pos, **float** newPrice) {          fstream file(filename, ios::binary | ios::in | ios::out);  **if** (!file) {              cerr << "文件不存在！" << endl;  **return**;          }            file.seekg((pos-1)\***sizeof**(Book));          Book b;  **if** (file.read(**reinterpret\_cast**<**char**\*>(&b), **sizeof**(Book))) {              b.price = newPrice;              file.seekp((pos-1)\***sizeof**(Book));              file.write(**reinterpret\_cast**<**char**\*>(&b), **sizeof**(Book));              cout << "修改成功！" << endl;          } **else** {              cerr << "记录不存在！" << endl;          }      }    **void** displayAll() {          ifstream inFile(filename, ios::binary);  **if** (!inFile) {              cerr << "文件不存在！" << endl;  **return**;          }            Book b;  **int** counter = 0;  **while** (inFile.read(**reinterpret\_cast**<**char**\*>(&b), **sizeof**(Book))) {              cout << "记录#" << ++counter << "\n"                   << b.format() << "\n\n";          }    **if** (counter == 0) {              cout << "没有记录！" << endl;          }      }    **void** searchBooks(**const** string& keyword) {          ifstream inFile(filename, ios::binary);  **if** (!inFile) {              cerr << "文件不存在！" << endl;  **return**;          }            Book b;  **int** counter = 0;  **while** (inFile.read(**reinterpret\_cast**<**char**\*>(&b), **sizeof**(Book))) {              stringstream ss;              ss << b.id << b.title;              string bookInfo = ss.str();  **if** (bookInfo.find(keyword) != string::npos) {                  cout << "记录#" << ++counter << "\n"                       << b.format() << "\n\n";              }          }    **if** (counter == 0) {              cout << "未找到匹配的图书记录！" << endl;          }      }  };    **class** MenuSystem {      BookFileManager manager;    **void** showMenu() {          cout << "\n=== 图书管理系统 ==="               << "\n1. 添加记录"               << "\n2. 读取记录"               << "\n3. 修改价格"               << "\n4. 显示全部"               << "\n5. 搜索图书"               << "\n0. 退出系统"               << "\n请输入选择：";      }    **public**:      MenuSystem() : manager("books.dat") {}    **void** run() {  **int** choice;  **do** {              showMenu();              cin >> choice;              cin.ignore();    **switch**(choice) {  **case** 1:                      manager.addRecord();  **break**;  **case** 2: {  **int** pos;                      cout << "输入记录位置：";                      cin >> pos;                      manager.readRecord(pos);  **break**;                  }  **case** 3: {  **int** pos;  **float** price;                      cout << "输入记录位置：";                      cin >> pos;                      cout << "输入新价格：";                      cin >> price;                      manager.modifyPrice(pos, price);  **break**;                  }  **case** 4:                      manager.displayAll();  **break**;  **case** 5: {                      string keyword;                      cout << "输入搜索关键词：";                      getline(cin, keyword);                      manager.searchBooks(keyword);  **break**;                  }  **case** 0:                      cout << "系统已退出！" << endl;  **break**;  **default**:                      cerr << "无效选项！" << endl;              }          } **while** (choice != 0);      }  };    **int** main() {      MenuSystem **system**;  **system**.run();  **return** 0;  } |

## 运行结果

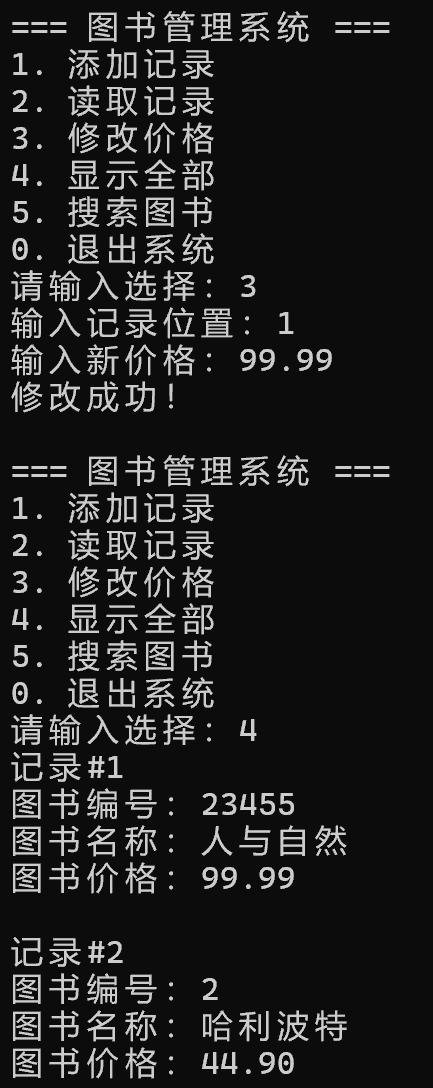
添加记录： 显示全部：

读取指定位置记录：



修改价格：



搜索图书： 退出系统：

