Министерство образования Республики Беларусь

Учреждение образования

«БЕЛОРУССКИЙ ГОСУДАРСТВЕННЫЙ УНИВЕРСИТЕТ ИНФОРМАТИКИ И РАДИОЭЛЕКТРОНИКИ»

Кафедра электронных вычислительных машин

ОТЧЕТ

О ЛАБОРАТОРНОЙ РАБОТЕ № 3

Наследование

по дисциплине «Программирование на языках высокого уровня»

Выполнил ст. гр. 450503 Д.С. Абрамчук

Проверил асс. каф. ЭВМ И.Г. Скиба

Минск 2025

# **1 ПОСТАНОВКА ЗАДАЧИ**

Реализовать класс «Статья» с полями «название статьи» и «имя автора» и методами, позволяющими посмотреть эти поля. Реализовать класс «Библиотечная карточка» хранящая автора, название, авторский знак, инвентарный номер, код по тематическому каталогу. Класс должен иметь методы, возвращающие эти поля и иметь метод для алфавитного поиска, возвращающий объединенное значение полей «автор» и «название».

Реализовать производные от него классы:

1) «Карточка для самостоятельного издания» с методами, возвращающими издательство, год издания, тираж, количество страниц и производные от него классы:

а) «карточка для сборника статей» с методом, возвращающим карточки, описывающие статьи (см. ниже);

б) «карточка для книги» (книга состоит из одной статьи), дополнительный метод должен возвращать статью.

2) «Карточка для статьи» с методами, возвращающими кроме имени автора, статью и ссылку на карточку для издания, в котором статья опубликована. Реализовать класс «Каталог», с производными «Тематический каталог», «Алфавитный каталог» с методами, реализующими поиск по шифру или автору (названию).

# **2 ЛИСТИНГ КОДА**

Файл main.cpp

#include "function.h"  
  
int main() {  
 ShowMenu();  
 return 0;  
}

Файл function.cpp

#include "../header/ABCCatalog.h"  
#include "../header/article.h"  
#include "../header/BookCard.h"  
#include "../header/function.h"  
#include "../header/ThematicCatalog.h"  
#include <iostream>  
#include <array>  
  
using namespace std;  
  
void show\_menu() {  
 auto \*main\_catalog = new Catalog();  
 auto \*thematic\_catalog = new ThematicCatalog();  
 auto \*abc\_catalog = new ABCCatalog();  
  
 array<CollectionCard \*, 100> collections{};  
 int collection\_count = 0;  
  
 int choice;  
  
 do {  
 cout << "\n=== Библиотечная система ===" << endl;  
 cout << "1. Добавить сборник статей" << endl;  
 cout << "2. Добавить книгу" << endl;  
 cout << "3. Добавить статью в сборник" << endl;  
 cout << "4. Поиск по шифру" << endl;  
 cout << "5. Поиск по автору или названию" << endl;  
 cout << "6. Показать все в алфавитном порядке" << endl;  
 cout << "0. Выход" << endl;  
 cout << "Выберите действие: ";  
 cin >> choice;  
 cin.ignore();  
  
 switch (choice) {  
 case 1: {  
 string author;  
 string title;  
 string code;  
 string publisher;  
 int year;  
 int circulation;  
 int pages;  
  
 cout << "Автор сборника: ";  
 getline(cin, author);  
 cout << "Название сборника: ";  
 getline(cin, title);  
 cout << "Код каталога: ";  
 getline(cin, code);  
 cout << "Издательство: ";  
 getline(cin, publisher);  
 cout << "Год издания: ";  
 cin >> year;  
 cout << "Тираж: ";  
 cin >> circulation;  
 cout << "Количество страниц: ";  
 cin >> pages;  
 cin.ignore();  
  
 auto \*collection = new CollectionCard(  
 author, title, code, publisher, year, circulation, pages);  
  
 if (collection\_count < 100) {  
 collections[collection\_count] = collection;  
 collection\_count++;  
 }  
  
 int article\_count;  
 cout << "Сколько статей добавить в сборник? ";  
 cin >> article\_count;  
 cin.ignore();  
  
 for (int i = 0; i < article\_count; i++) {  
 string article\_author;  
 string article\_title;  
 cout << "Статья " << (i + 1) << " автор: ";  
 getline(cin, article\_author);  
 cout << "Статья " << (i + 1) << " название: ";  
 getline(cin, article\_title);  
  
 auto \*article = new Article(article\_title, article\_author);  
 collection->add\_article(article);  
 }  
  
 main\_catalog->add\_card(collection);  
 thematic\_catalog->add\_card(collection);  
 abc\_catalog->add\_card(collection);  
  
 cout << "Сборник статей добавлен!" << endl;  
 break;  
 }  
  
 case 2: {  
 string author;  
 string title;  
 string code;  
 string publisher;  
 int year;  
 int circulation;  
 int pages;  
  
 cout << "Автор: ";  
 getline(cin, author);  
 cout << "Название: ";  
 getline(cin, title);  
 cout << "Код каталога: ";  
 getline(cin, code);  
 cout << "Издательство: ";  
 getline(cin, publisher);  
 cout << "Год издания: ";  
 cin >> year;  
 cout << "Тираж: ";  
 cin >> circulation;  
 cout << "Количество страниц: ";  
 cin >> pages;  
 cin.ignore();  
  
 auto \*book\_article = new Article(title, author);  
 auto book = new BookCard(BookData{author, title, code, publisher, year, circulation, pages},  
 book\_article);  
  
 main\_catalog->add\_card(book);  
 thematic\_catalog->add\_card(book);  
 abc\_catalog->add\_card(book);  
  
 cout << "Книга добавлена!" << endl;  
 break;  
 }  
  
 case 3: {  
 if (collection\_count == 0) {  
 cout << "Сначала добавьте сборник статей!" << endl;  
 break;  
 }  
  
 cout << "Доступные сборники:" << endl;  
 for (int i = 0; i < collection\_count; i++) {  
 cout << (i + 1) << ". " << collections[i]->get\_title() << endl;  
 }  
  
 int collection\_choice;  
 cout << "Выберите сборник: ";  
 cin >> collection\_choice;  
 cin.ignore();  
  
 if (collection\_choice < 1 || collection\_choice > collection\_count) {  
 cout << "Неверный выбор!" << endl;  
 break;  
 }  
  
 string article\_author;  
 string article\_title;  
 cout << "Автор статьи: ";  
 getline(cin, article\_author);  
 cout << "Название статьи: ";  
 getline(cin, article\_title);  
  
 auto \*new\_article = new Article(article\_title, article\_author);  
 collections[collection\_choice - 1]->add\_article(new\_article);  
  
 cout << "Статья добавлена!" << endl;  
 break;  
 }  
  
 case 4: {  
 string cipher;  
 cout << "Введите шифр для поиска: ";  
 getline(cin, cipher);  
 main\_catalog->search\_by\_cipher(cipher);  
 break;  
 }  
  
 case 5: {  
 string search\_term;  
 cout << "Введите автора или название: ";  
 getline(cin, search\_term);  
 main\_catalog->search\_by\_author\_or\_title(search\_term);  
 break;  
 }  
  
 case 6: {  
 abc\_catalog->show\_all\_sorted();  
 break;  
 }  
  
 case 0:  
 cout << "Выход из программы..." << endl;  
 break;  
  
 default:  
 cout << "Неверный выбор!" << endl;  
 }  
  
 } while (choice != 0);  
  
 for (int i = 0; i < collection\_count; i++) {  
 delete collections[i];  
 }  
  
 delete main\_catalog;  
 delete thematic\_catalog;  
 delete abc\_catalog;  
}

Файл function.h

#ifndef LAB\_\_\_3\_\_FUNCTION\_H  
#define LAB\_\_\_3\_\_FUNCTION\_H  
  
void show\_menu();  
  
#endif

Файл ABCCatalog.cpp

#include "../header/ABCCatalog.h"  
#include <iostream>  
  
void ABCCatalog::show\_all\_sorted() const {  
 std::cout << "Все карточки в алфавитном порядке:" << std::endl;  
 int count = get\_card\_count();  
  
 if (count == 0) {  
 std::cout << "Каталог пуст." << std::endl;  
 return;  
 }  
  
 auto titles = new std::string[count];  
  
 for (int i = 0; i < count; i++) {  
 titles[i] = get\_card(i)->get\_alphabetical\_search();  
 }  
  
 for (int i = 0; i < count - 1; i++) {  
 for (int j = i + 1; j < count; j++) {  
 if (titles[i] > titles[j]) {  
 std::swap(titles[i], titles[j]);  
 }  
 }  
 }  
  
 for (int i = 0; i < count; i++) {  
 std::cout << titles[i] << std::endl;  
 }  
  
 delete[] titles;  
}

Файл ABCCatalog.h

#ifndef LAB\_\_\_3\_\_ABCCATALOG\_H  
#define LAB\_\_\_3\_\_ABCCATALOG\_H  
  
#include "../header/Catalog.h"  
  
class ABCCatalog : public Catalog {  
public:  
 ~ABCCatalog() = default;  
  
 void show\_all\_sorted() const;  
};  
  
#endif

Файл article.cpp

#include "../header/article.h"  
  
Article::Article(const std::string &article\_title, const std::string &author)  
 : title(article\_title), author\_name(author) {  
}  
  
Article::~Article() = default;  
  
std::string Article::get\_title() const {  
 return title;  
}  
  
std::string Article::get\_author\_name() const {  
 return author\_name;  
}

Файл article.h

#ifndef LAB\_\_\_3\_\_ARTICLE\_H  
#define LAB\_\_\_3\_\_ARTICLE\_H  
  
#include <string>  
  
  
class Article {  
private:  
 std::string title;  
 std::string author\_name;  
  
public:  
 Article(const std::string &article\_title, const std::string &author);  
  
 ~Article();  
  
 std::string get\_title() const;  
  
 std::string get\_author\_name() const;  
};  
  
#endif

Файл BookCard.cpp

#include "../header/BookCard.h"  
#include <string>  
  
BookCard::~BookCard() {  
 delete book\_article;  
}  
  
BookCard::BookCard(const BookCard &other)  
 : SelfPublishCard(other),  
 book\_article(other.book\_article ? new Article(\*other.book\_article) : nullptr) {  
}  
BookCard &BookCard::operator=(const BookCard &other) {  
 if (this != &other) {  
 SelfPublishCard::operator=(other);  
 delete book\_article;  
 book\_article = other.book\_article ? new Article(\*other.book\_article) : nullptr;  
 }  
 return \*this;  
}  
  
BookCard::BookCard(BookCard &&other) noexcept  
 : SelfPublishCard(other), book\_article(other.book\_article) {  
 other.book\_article = nullptr;  
}  
BookCard &BookCard::operator=(BookCard &&other) noexcept {  
 if (this != &other) {  
 SelfPublishCard::operator=(other);  
 delete book\_article;  
 book\_article = other.book\_article;  
 other.book\_article = nullptr;  
 }  
 return \*this;  
}

Файл BookCard.h

#ifndef LAB\_\_\_3\_\_BOOKCARD\_H  
#define LAB\_\_\_3\_\_BOOKCARD\_H  
  
#include "../header/CollectionCard.h"  
#include "../header/article.h"  
#include <string>  
  
struct BookData {  
 std::string author;  
 std::string title;  
 std::string code;  
 std::string publisher;  
 int year;  
 int circulation;  
 int pages;  
};  
  
class BookCard : public SelfPublishCard {  
private:  
 Article \*book\_article;  
  
public:  
 BookCard(const BookData &data, Article \*article)  
 : SelfPublishCard(data.author, data.title, data.code,  
 data.publisher, data.year, data.circulation, data.pages),  
 book\_article(article) {}  
  
 ~BookCard();  
  
 BookCard(const BookCard &other);  
  
 BookCard &operator=(const BookCard &other);  
  
 BookCard(BookCard &&other) noexcept;  
  
 BookCard &operator=(BookCard &&other) noexcept;  
};  
  
#endif

Файл Catalog.cpp

#include "../header/Catalog.h"  
#include <iostream>  
  
Catalog::~Catalog() {  
 for (int i = 0; i < card\_count; i++) {  
 delete cards[i];  
 }  
 delete[] cards;  
}  
  
Catalog::Catalog(Catalog&& other) noexcept  
 : capacity(other.capacity),  
 cards(other.cards),  
 card\_count(other.card\_count) {  
 other.cards = nullptr;  
 other.card\_count = 0;  
 other.capacity = 0;  
}  
  
Catalog& Catalog::operator=(Catalog&& other) noexcept {  
 if (this != &other) {  
 for (int i = 0; i < card\_count; i++) {  
 delete cards[i];  
 }  
 delete[] cards;  
  
 cards = other.cards;  
 card\_count = other.card\_count;  
 capacity = other.capacity;  
  
 other.cards = nullptr;  
 other.card\_count = 0;  
 other.capacity = 0;  
 }  
 return \*this;  
}  
  
void Catalog::add\_card(LibCard\* card) {  
 if (card\_count >= capacity) {  
 int new\_capacity = (capacity == 0) ? 2 : capacity \* 2;  
  
 if (new\_capacity <= card\_count) {  
 new\_capacity = card\_count + 1;  
 }  
  
 auto new\_cards = new LibCard \*[new\_capacity];  
 for (int i = 0; i < card\_count; i++) {  
 new\_cards[i] = cards[i];  
 }  
 delete[] cards;  
 cards = new\_cards;  
 capacity = new\_capacity;  
 }  
  
 cards[card\_count] = card;  
 card\_count++;  
}  
  
void Catalog::search\_by\_cipher(const std::string& cipher) const {  
 std::cout << "Поиск по шифру '" << cipher << "':" << std::endl;  
 bool found = false;  
  
 for (int i = 0; i < card\_count; i++) {  
 if (cards[i]->get\_thematic\_code() == cipher) {  
 std::cout << "Найдено: " << cards[i]->get\_alphabetical\_search() << std::endl;  
 found = true;  
 }  
 }  
  
 if (!found) {  
 std::cout << "Ничего не найдено." << std::endl;  
 }  
}  
  
void Catalog::search\_by\_author\_or\_title(const std::string& search\_term) const {  
 std::cout << "Поиск '" << search\_term << "':" << std::endl;  
 bool found = false;  
  
 for (int i = 0; i < card\_count; i++) {  
 if (cards[i]->get\_author().contains(search\_term) ||  
 cards[i]->get\_title().contains(search\_term)) {  
 std::cout << "Найдено: " << cards[i]->get\_alphabetical\_search() << std::endl;  
 found = true;  
 }  
 }  
  
 if (!found) {  
 std::cout << "Ничего не найдено." << std::endl;  
 }  
}

Файл Catalog.h

#ifndef LAB\_\_\_3\_\_CATALOG\_H  
#define LAB\_\_\_3\_\_CATALOG\_H  
  
#include "../header/libcard.h"  
#include <string>  
  
class Catalog {  
private:  
 int capacity = 0;  
 LibCard\*\* cards = nullptr;  
 int card\_count = 0;  
  
protected:  
 int get\_card\_count() const { return card\_count; }  
 int get\_capacity() const { return capacity; }  
 LibCard\* get\_card(int index) const {  
 return (index >= 0 && index < card\_count) ? cards[index] : nullptr;  
 }  
  
 void set\_card(int index, LibCard\* card) {  
 if (index >= 0 && index < card\_count) {  
 cards[index] = card;  
 }  
 }  
 void increment\_card\_count() { card\_count++; }  
 void set\_capacity(int cap) { capacity = cap; }  
 void set\_cards(LibCard\*\* new\_cards) { cards = new\_cards; }  
  
public:  
 Catalog() = default;  
 ~Catalog();  
  
 Catalog(const Catalog &) = delete;  
 Catalog &operator=(const Catalog &) = delete;  
 Catalog(Catalog &&other) noexcept;  
 Catalog &operator=(Catalog &&other) noexcept;  
  
 void add\_card(LibCard \*card);  
 void search\_by\_cipher(const std::string &cipher) const;  
 void search\_by\_author\_or\_title(const std::string &search\_term) const;  
};  
  
#endif

Файл CollectionCard.cpp

#include "../header/CollectionCard.h"  
#include <string>  
CollectionCard::CollectionCard(const std::string &auth, const std::string &titl,  
 const std::string &code, const std::string &publ,  
 int yr, int circ, int pages)  
 : SelfPublishCard(auth, titl, code, publ, yr, circ, pages) {  
}  
  
CollectionCard::~CollectionCard() {  
 for (int i = 0; i < article\_count; i++) {  
 delete articles[i];  
 }  
 delete[] articles;  
}  
  
CollectionCard::CollectionCard(CollectionCard&& other) noexcept  
 : SelfPublishCard(other),  
 articles(other.articles),  
 article\_count(other.article\_count),  
 capacity(other.capacity) {  
 other.articles = nullptr;  
 other.article\_count = 0;  
 other.capacity = 0;  
}  
  
CollectionCard& CollectionCard::operator=(CollectionCard&& other) noexcept {  
 if (this != &other) {  
 for (int i = 0; i < article\_count; i++) {  
 delete articles[i];  
 }  
 delete[] articles;  
  
 SelfPublishCard::operator=(other);  
  
 articles = other.articles;  
 article\_count = other.article\_count;  
 capacity = other.capacity;  
  
 other.articles = nullptr;  
 other.article\_count = 0;  
 other.capacity = 0;  
 }  
 return \*this;  
}  
  
void CollectionCard::add\_article(Article \*article) {  
 if (article\_count >= capacity) {  
 int new\_capacity = (capacity == 0) ? 2 : capacity \* 2;  
  
 if (new\_capacity <= article\_count) {  
 new\_capacity = article\_count + 1;  
 }  
  
 auto new\_articles = new Article \*[new\_capacity];  
 for (int i = 0; i < article\_count; i++) {  
 new\_articles[i] = articles[i];  
 }  
 delete[] articles;  
 articles = new\_articles;  
 capacity = new\_capacity;  
 }  
  
 articles[article\_count] = article;  
 article\_count++;  
}

Файл CollectionCard.h

#ifndef LAB\_\_\_3\_\_COLLECTIONCARD\_H  
#define LAB\_\_\_3\_\_COLLECTIONCARD\_H  
  
#include "../header/SelfPublishCard.h"  
#include "../header/article.h"  
#include <string>  
  
class CollectionCard : public SelfPublishCard {  
private:  
 Article \*\*articles = nullptr;  
 int article\_count = 0;  
 int capacity = 0;  
  
public:  
 CollectionCard(const std::string &auth, const std::string &titl,  
 const std::string &code, const std::string &publ,  
 int yr, int circ, int pages);  
  
 ~CollectionCard();  
  
 CollectionCard(const CollectionCard &) = delete;  
  
 CollectionCard &operator=(const CollectionCard &) = delete;  
  
 CollectionCard(CollectionCard &&other) noexcept;  
  
 CollectionCard &operator=(CollectionCard &&other) noexcept;  
  
 void add\_article(Article \*article);  
  
};  
  
#endif

Файл libcard.cpp

#include "../header/libcard.h"  
  
  
LibCard::LibCard(const std::string &auth, const std::string &titl, const std::string &code)  
 : author(auth), title(titl), thematic\_code(code) {  
}  
  
LibCard::~LibCard() = default;  
  
std::string LibCard::get\_author() const {  
 return author;  
}  
  
std::string LibCard::get\_title() const {  
 return title;  
}  
  
std::string LibCard::get\_thematic\_code() const {  
 return thematic\_code;  
}  
  
std::string LibCard::get\_alphabetical\_search() const {  
 return author + " - " + title;  
}

Файл libcard.h

#ifndef LAB\_\_\_3\_\_LIBCARD\_H  
#define LAB\_\_\_3\_\_LIBCARD\_H  
  
#include <string>  
  
  
class LibCard {  
private:  
 std::string author;  
 std::string title;  
 std::string thematic\_code;  
  
public:  
 LibCard(const std::string &auth, const std::string &titl, const std::string &code);  
  
 ~LibCard();  
  
 std::string get\_author() const;  
  
 std::string get\_title() const;  
  
 std::string get\_thematic\_code() const;  
  
 std::string get\_alphabetical\_search() const;  
};  
#endif

Файл SelfPublishedCard.cpp

#include "../header/SelfPublishCard.h"  
  
  
SelfPublishCard::SelfPublishCard(const std::string &auth, const std::string &titl,  
 const std::string &code, const std::string &publ,  
 int yr, int circ, int pages)  
 : LibCard(auth, titl, code), publisher(publ), year(yr),  
 circulation(circ), page\_count(pages) {  
}

Файл SelfPublishedCard.h

#ifndef LAB\_\_\_3\_\_SELFPUBLISHCARD\_H  
#define LAB\_\_\_3\_\_SELFPUBLISHCARD\_H  
  
#include "../header/libcard.h"  
#include <string>  
  
class SelfPublishCard : public LibCard {  
private:  
 std::string publisher;  
 int year;  
 int circulation;  
 int page\_count;  
  
public:  
 SelfPublishCard(const std::string &auth, const std::string &titl,  
 const std::string &code, const std::string &publ,  
 int yr, int circ, int pages);  
  
 ~SelfPublishCard() = default;  
};  
  
#endif

**3 РЕЗУЛЬТАТ РАБОТЫ ПРОГРАММЫ**

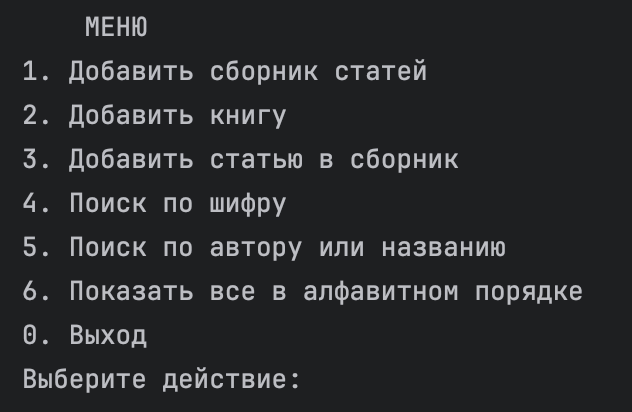


Рисунок 3.1 – меню, с которым взаимодействует пользователь

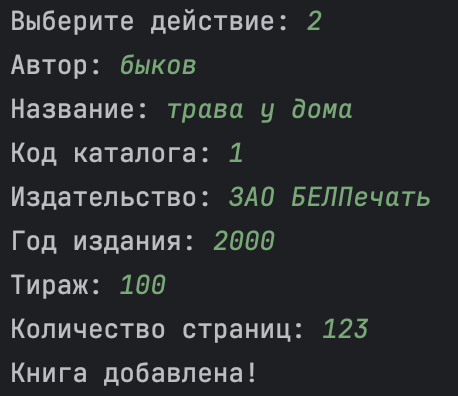


Рисунок 3.2 – добавление книги

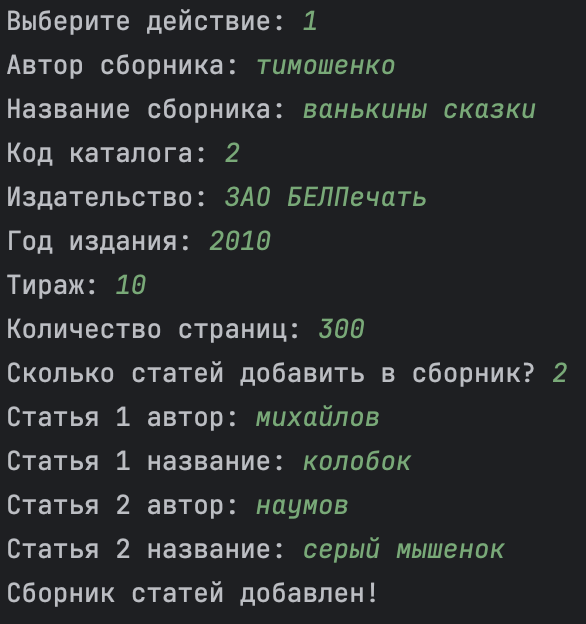


Рисунок 3.3 – добавление сборника статей

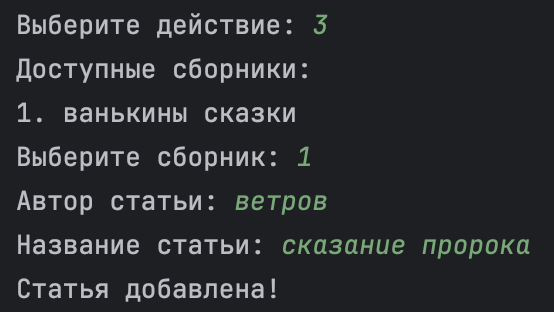


Рисунок 3.4 – добавление статьи в сборник

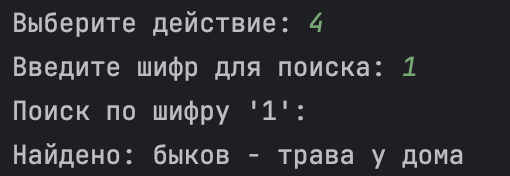
****

Рисунок 3.5 – поиск по коду каталога

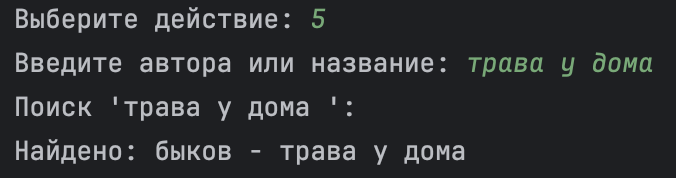


Рисунок 3.6 – поиск по автору

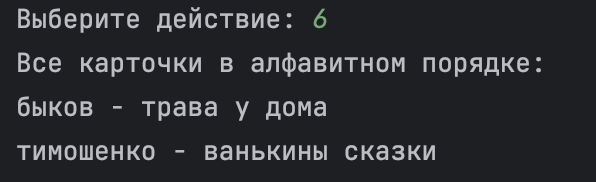
****

Рисунок 3.7 – вывод всех сборников на экран