

Моя программа:

```
7labс++.cpp 7labс++ Book
1 #include <iostream>
2 #include <string>
3 #include <vector>
4 #include <algorithm>
5 #include <fstream>
6
7 class Book
8 {
9 public:
10     std::string getTitle()
11     {
12         return _title;
13     }
14
15     std::string getAuthor()
16     {
17         return _author;
18     }
19
20     std::string getGenre()
21     {
22         return _genre;
23     }
24
25     int getAvailability()
26     {
27         return _availability;
28     }
29
30     void setAvailability(int av)
31     {
32         _availability = av;
33     }
34
35     bool operator==(const Book other)
36     {
37         return ((this->_title == other._title) && (this->_author == other._author) && (this->_genre == other._genre));
38     }
39
40     std::string serialize() const {
41         return _title + ";" + _author + ";" + _genre + ";" + std::to_string(_availability);
42     }
43
44     void deserialize(const std::string& data) {
45         size_t pos = 0;
46         size_t next_pos = data.find(';', pos);
47         _title = data.substr(pos, next_pos - pos);
```

```

47     _title = data.substr(pos, next_pos - pos);
48     pos = next_pos + 1;
49     next_pos = data.find(';', pos);
50     _author = data.substr(pos, next_pos - pos);
51     pos = next_pos + 1;
52     next_pos = data.find(';', pos);
53     _genre = data.substr(pos, next_pos - pos);
54     pos = next_pos + 1;
55     _availability = std::stoi(data.substr(pos));
56 }
57
58 private:
59     std::string _title, _author, _genre;
60     int _availability;
61 };
62
63 class Library
64 {
65 public:
66     void add(Book book)
67     {
68         _lib.push_back(book);
69     }
70
71     void take(Book book)
72     {
73         auto p = std::find(_lib.begin(), _lib.end(), book);
74         int ind = std::distance(_lib.begin(), p);
75         if (p != _lib.end())
76         {
77             if (_lib[ind].getAvailability() == 1)
78             {
79                 std::cout << book.getTitle() << " is taken successfully!" << std::endl;
80                 _lib[ind].setAvailability(0);
81             }
82             else
83                 std::cout << book.getTitle() << " can't be taken, isn't available!" << std::endl;
84         }
85         else
86         {
87             std::cout << book.getTitle() << " doesn't exist in Library!" << std::endl;
88         }
89     }
90
91     void give(Book book)
92     {

```

```

92     {
93         auto p = std::find(_lib.begin(), _lib.end(), book);
94         int ind = std::distance(_lib.begin(), p);
95         if (p != _lib.end())
96         {
97             if (_lib[ind].getAvailability() == 0)
98             {
99                 std::cout << book.getTitle() << " is given back successfully!" << std::endl;
100                 _lib[ind].setAvailability(1);
101             }
102             else
103                 std::cout << book.getTitle() << " can't be given back, it's there already!" << std::endl;
104         }
105         else
106         {
107             std::cout << book.getTitle() << " doesn't exist in Library!" << std::endl;
108         }
109     }
110
111     void out()
112     {
113         for (auto& it : _lib)
114         {
115             std::cout << it.getTitle() << '|' << it.getAuthor() << '|' << it.getGenre() << '|' << it.getAvailability() << std::endl;
116         }
117     }
118
119     private:
120         std::vector<Book> _lib;
121     };
122
123     void saveLibrary(const std::vector<Book>& library, const std::string& filename)
124     {
125         std::ofstream fileOut(filename);
126         for (const auto& book : library) {
127             fileOut << book.serialize() << std::endl;
128         }
129         fileOut.close();
130     }
131
132     void loadLibrary(std::vector<Book>& library, const std::string& filename)
133     {
134         std::ifstream fileIn(filename);
135         std::string line;
136         while (std::getline(fileIn, line)) {
137             Book book;
138             book.deserialize(line);

```

```

138             book.deserialize(line);
139             library.push_back(book);
140         }
141         fileIn.close();
142     }
143

```

Примеры:

```
Test (Глобальная область)
1  #include <iostream>
2  #include <vector>
3  #include "locale.h"
4  int main() {
5      setlocale(LC_ALL, "Russian");
6      // Создание вектора для хранения целых чисел
7      std::vector<int> numbers;
8      // Добавление элементов в вектор
9      numbers.push_back(10);
10     numbers.push_back(20);
11     numbers.push_back(30);
12     // Доступ к элементам с использованием оператора []
13     std::cout << "Элемент с индексом 1: " << numbers[1] << std::endl;
14     // Итерация по вектору с использованием цикла range-based for
15     std::cout << "Содержимое вектора: ";
16     for (int num : numbers) {
17         std::cout << num << " ";
18     }
19     std::cout << std::endl;
20     return 0;
21 }
```

```
Test.cpp* X Test (Глобальная область)
1  #include <iostream>
2  #include <list>
3  #include "locale.h"
4  int main() {
5      setlocale(LC_ALL, "Russian");
6      // Создание списка для хранения целых чисел
7      std::list<int> numbers;
8      // Добавление элементов в список
9      numbers.push_back(10); // Добавление в конец списка
10     numbers.push_back(20);
11     numbers.push_front(5); // Добавление в начало списка
12     // Итерация по списку с использованием итератора для доступа к элементам
13     std::cout << "Содержимое списка: ";
14     for (auto it = numbers.begin(); it != numbers.end(); ++it) {
15         std::cout << *it << " ";
16     }
17     std::cout << std::endl;
18     // Удаление элемента из списка
19     numbers.remove(20); // Удаление всех вхождений элемента со значением 20
20     // Повторная итерация по списку для вывода его содержимого после удаления элемента
21     std::cout << "Содержимое списка после удаления элемента: ";
22     for (int num : numbers) {
23         std::cout << num << " ";
24     }
25     std::cout << std::endl;
26     return 0;
27 }
28
```

```

1  #include <iostream>
2  #include <vector>
3  #include <list>
4  #include <map>
5  #include <string>
6  #include "locale.h"
7  class Employee {
8  public:
9      int id;
10     std::string name;
11     std::string department;
12     Employee(int id, std::string name, std::string department) : id(id), name(name), department(department) {}
13     virtual void display() {
14         std::cout << "ID: " << id << ", Name: " << name << ", Department: " << department << std::endl;
15     }
16 };
17 class FullTimeEmployee : public Employee {
18 public:
19     double salary;
20     FullTimeEmployee(int id, std::string name, std::string department, double salary)
21         : Employee(id, name, department), salary(salary) {}
22     void display() override {
23         Employee::display();
24         std::cout << "Salary: " << salary << std::endl;
25     }
26 };
27 class PartTimeEmployee : public Employee {
28 public:
29     double hourlyRate;
30     PartTimeEmployee(int id, std::string name, std::string department, double hourlyRate)
31         : Employee(id, name, department), hourlyRate(hourlyRate) {}
32     void display() override {
33         Employee::display();
34         std::cout << "Hourly Rate: " << hourlyRate << std::endl;
35     }
36 };
37
38 int main() {
39     setlocale(LC_ALL, "Russian");
40     std::vector<Employee*> employees;
41     std::list<std::string> departments;
42     std::map<std::string, std::vector<Employee*>> departmentEmployees;
43     // Добавление сотрудников и отделов в контейнеры
44     employees.push_back(new FullTimeEmployee(1, "Иван Иванов", "Разработка", 50000));
45     employees.push_back(new PartTimeEmployee(2, "Петр Петров", "Маркетинг", 300));
46     departments.push_back("Разработка");
47     departments.push_back("Маркетинг");
48     for (auto& emp : employees) {
49         departmentEmployees[emp->department].push_back(emp);
50     }
51     // Вывод информации о сотрудниках
52     for (auto& emp : employees) {
53         emp->display();
54     }
55     // Очистка динамически выделенной памяти
56     for (auto& emp : employees) {
57         delete emp;
58     }
59     return 0;
60 }
61

```

```

1  #include <iostream>
2  #include <map>
3  #include <string>
4  #include "locale.h"
5  class Product {
6  public:
7      int productID;
8      std::string productName;
9      double price;
10     Product(int productID, std::string productName, double price)
11         : productID(productID), productName(productName)
12     {
13         virtual void display() {
14             std::cout << "Product ID: " << productID << ", Price: " << price <<
15             std::endl;
16         }
17     };
18
19     class Electronics : public Product {
20     public:
21         Electronics(int productID, std::string productName, double price)
22             : Product(productID, productName, price) {}
23         void display() override {
24             Product::display();
25             std::cout << "Category: Electronics" << std::endl;
26         }
27     };
28
29     class Clothing : public Product {
30     public:
31         Clothing(int productID, std::string productName, double price)
32             : Product(productID, productName, price) {}
33         void display() override {
34             Product::display();
35             std::cout << "Category: Clothing" << std::endl;
36         }
37     };
38
39     int main() {
40         setlocale(LC_ALL, "Russian");
41         std::map<int, std::pair<Product*, int>> inventory;
42         // Добавление продуктов в инвентарь
43         inventory[1] = std::make_pair(new Electronics(1, "Smartphone", 500.0), 10);
44         inventory[2] = std::make_pair(new Clothing(2, "T-Shirt", 20.0), 50);
45         // Вывод инвентаря
46         for (const auto& item : inventory) {
47             std::cout << "Stock: " << item.second.second << " ";
48             item.second.first->display();
49         }
50         // Очистка памяти

```

```

45     std::cout << "Stock: " << item.second.second << " ";
46     item.second.first->display();
47 }
48 // Очистка памяти
49 for (auto& item : inventory) {
50     delete item.second.first;
51 }
52 return 0;
53 }

```