МИНОБРНАУКИ РОССИИ

федеральное государственное бюджетное образовательное учреждение

высшего образования

«Московский государственный технологический университет «СТАНКИН»

(ФГБОУ ВО «МГТУ «СТАНКИН»)

Институт
информационных технологий

Кафедра
информационных систем

Основная образовательная программа 09.03.02
«Информационные системы и технологии»

КУРСОВАЯ РАБОТА

по дисциплине «Объектно-ориентированное программирование»

Тема: «База данных книжного магазина»

Руководитель
к.т.н., доцент _____ **Разумовский А.И.**
подпись

Выполнил
студент группы ИДБ-22-06 _____ **Мустафаева П.М.**
подпись

Москва, 2024

ОГЛАВЛЕНИЕ

ВВЕДЕНИЕ

Целью данной курсовой работы будет являться написание базы данных книжного магазина на языке C++. Создание базы данных для книжного магазина является важным решением для управления ассортиментом, заказами и клиентской информацией.

В данной курсовой работе будет разработана база данных книжного магазина на языке C++ с применением ООП (объектно-ориентированного программирования), которое позволяет создавать более эффективные и надежные программы. Основными пунктами в данной работе будут: реализация структуры базы данных, включая таблицы для хранения информации о книгах, авторах, жанрах, клиентах и заказах.

Целью данной работы является изучение и применение принципов ООП при разработке базы данных книжного магазина. Результатом проекта будет полноценная база данных, которая может быть использована для управления операциями магазина, включая инвентаризацию и заказы.

Проект будет состоять из двух частей. В первой части будет рассмотрена теория ООП, во второй – реализация структуры базы данных и её взаимодействия с пользовательским интерфейсом.

ГЛАВА 1. СРЕДСТВА РАЗРАБОТКИ

1.1. ИСТОРИЯ ООП

Термины «объектно-» и «ориентированный» в современном смысле этих слов появились в MIT в конце 1950 начале 1960 годов. В среде специалистов по искусственному интеллекту термин «объект» мог относиться к идентифицированным элементам (атомы Lisp) со свойствами (атрибутами). Алан Кэй позже писал, что понимание внутреннего устройства Лиспа оказало серьезное влияние на его мышление в 1966 г. Другим ранним примером ООП в MIT был Sketchpad созданный Иваном Сазерлендом в 1960-61. В глоссарии подготовленного в 1963 г. технического отчета, основанного на его диссертации о Sketchpad, Сазерленд определяет понятия «объект» и «экземпляр» с концепцией классов на основе «мастера» или «определения», хотя все эти термины относились к графическому представлению объектов, то есть в Sketchpad было основное изображение, на основе которого строились копии. При изменении основного – копии тоже менялись.
В ранней MIT-версии ALGOL AED-0 структуры данных («плексы» на диалекте Алгола) напрямую были связаны с процедурами, которые впоследствии были названы сообщениями, методами или функциями-членами.

В результате первым языком, в котором был введён новой тип – объект, стал Simula-1 в конце 60-х годов. В описании этого типа одновременно указывались данные (поля) и процедуры, их обрабатывающие – методы. Родственные объекты объединялись в классы, описания которых оформлялись в виде блоков программы. При этом класс можно использовать в качестве префикса к другим классам, которые становятся в этом случае подклассами первого. Впоследствии Simula-1 был обобщен, и появился первый универсальный ООП-ориентированный язык программирования – Simula-67 (67 – по году создания). Авторы Simula — Оле-Йохан Даль и Кристен Нюгорд из Норвежского компьютерного центра в Осло. Также в языке использовался

автоматический сборщик мусора, который был изобретен ранее для функционального языка Lisp. Simula использовалась тогда преимущественно для физического моделирования. Идеи Simula оказали серьезное влияние на более поздние языки, такие как Smalltalk, варианты Lisp (CLOS), Object Pascal, и C++.

Язык Smalltalk, который был изобретен в компании Xerox PARC Аланом Кэем (Alan Kay) и некоторыми другими учеными, фактически навязывал использование «объектов» и «сообщений» как базиса для вычислений. Создателей Smalltalk вдохновляли некоторые идеи Simula, но Smalltalk разрабатывался как полностью динамичная система, в которой классы могут создаваться и изменяться динамически, а не только статически как в Simula. Smalltalk и ООП с его помощью были представлены широкой аудитории в журнале Byte magazine в августе 1981.

Объектно-ориентированное программирование развилось в доминирующую методологию программирования в начале и середине 1990 годов, когда стали широко доступны поддерживающие ее языки программирования, такие как Visual FoxPro 3.0, C++, и Delphi. Доминирование этой системы поддерживалось ростом популярности графических интерфейсов пользователя, которые основывались на техниках ООП. Пример тесной связи между динамической библиотекой GUI и объектно-ориентированного языка программирования можно найти, посмотрев на фреймворк Cocoa на Mac OS X, который был написан на Objective-C, объектно-ориентированном расширении к C, основанном на Smalltalk с поддержкой динамических сообщений. Инструментарии ООП повлияли на популярность событийно-ориентированного программирования (хотя, эта концепция не ограничивается одним ООП). Некоторые даже думали, что кажущаяся или реальная связь с графическими интерфейсами – это то, что вынесло ООП на передний план технологий.

В настоящее время количество прикладных языков программирования, реализующих объектно-ориентированную парадигму, является наибольшим

по отношению к другим парадигмам. Наиболее распространённые в промышленности языки (C++, Delphi, C#, Java и др.) воплощают объектную модель Simula. Примерами языков, опирающихся на модель Smalltalk, являются Objective-C, Python, Ruby.

1.2. ООП – КАК ИНСТРУМЕНТ БОРЬБЫ СО СЛОЖНОСТЬЮ РАЗРАБОТКИ ПО

Процесс разработки программного обеспечения с использованием объектно-ориентированного подхода включает четыре этапа:

1) анализ;
2) проектирование;
3) эволюция;
4) модификация.

Анализ. Цель анализа - максимально полное описание задачи. На этом этапе выполняется анализ предметной области задачи, объектная декомпозиция разрабатываемой системы и определяются важнейшие особенности поведения объектов (описание абстракций). По результатам анализа разрабатывается структурная схема программного продукта, на которой показываются основные объекты и сообщения, передаваемые между ними, а также выполняется описание абстракций.

Проектирование.

Различают:

а) логическое проектирование, при котором принимаемые решения практически не зависят от условий эксплуатации (операционной системы и используемого оборудования);

б) физическое проектирование, при котором приходится принимать во внимание указанные факторы.

Логическое проектирование заключается в разработке структуры классов: определяются поля для хранения составляющих состояния объектов и алгоритмы методов, реализующих аспекты поведения объектов. При этом

используются рассмотренные выше приемы разработки классов (наследование, композиция, наполнение, полиморфизм и т.д.). Результатом является иерархия или диаграмма классов, отражающие взаимосвязь классов, и описание классов.

Физическое проектирование включает объединение описаний классов в модули, выбор схемы их подключения (статическое или динамическая компоновка), определение способов взаимодействия с оборудованием, с операционной системой и/или другим программным обеспечением (например, базами данных, сетевыми программами), обеспечение синхронизации процессов для систем параллельной обработки и т.д.

Эволюция системы – это процесс поэтапной реализации и подключения классов к проекту.

Процесс начинается с создания основной программы или проекта будущего программного продукта. Затем реализуются и подключаются классы, так чтобы создать грубый, но, по возможности, работающий прототип будущей системы. Он тестируется и отлаживается. Например, таким прототипом может служить система, включающая реализацию основного интерфейса программного продукта (передача сообщений в отсутствующую пока часть системы не выполняется). В результате мы получаем работоспособный прототип продукта, который может быть, например, показан заказчику для уточнения требований.

Затем к системе подключается следующая группа классов, например, связанная с реализацией некоторого пункта меню. Полученный вариант также тестируется и отлаживается, и так далее, до реализации всех возможностей системы.

Использование поэтапной реализации существенно упрощает тестирование и отладку программного продукта.

Модификация – это процесс добавления новых функциональных возможностей или изменение существующих свойств системы. Как правило, изменения затрагивают реализацию класса, оставляя без изменения его

интерфейс, что при использовании ООП обычно обходится без особых неприятностей, так как процесс изменений затрагивает локальную область. Изменение интерфейса - так же не очень сложная задача, но ее решение может повлечь за собой необходимость согласования процессов взаимодействия объектов, что потребует изменений в других классах программы. Однако сокращение количества параметров в интерфейсной части по сравнению с модульным программированием существенно облегчает и этот процесс.

Простота модификации позволяет сравнительно легко адаптировать программные системы к изменяющимся условиям эксплуатации, что увеличивает время жизни систем, на разработку которых затрачиваются огромные временные и материальные ресурсы.

Особенностью ООП является то, что объект или группа объектов могут разрабатываться отдельно, и, следовательно, их проектирование может находиться на различных этапах. Например, интерфейсные классы уже реализованы, а структура классов предметной области еще только уточняются. Обычно, проектирование начинается, когда какой-либо фрагмент предметной области достаточно полно описан в процессе анализа.

1.3. ОСНОВНЫЕ ПРИНЦИПЫ ОБЪЕКТНО-ОРИЕНТИРОВАННОГО ПРОГРАММИРОВАНИЯ

В настоящее время объектно-ориентированный подход при разработке систем различной степени сложности общепризнан. Более того, он применяется не только при разработке, но и при использовании широко распространенных объектно-ориентированных систем.

Далее мы перейдем к принципам ООП. Основные принципы объектно-ориентированного программирования (ООП) – это базовые концептуальные структуры, которые используется при проектировании и написании программного обеспечения. Вот более подробное описание каждого из основных принципов ООП:

1. Инкапсуляция – это процесс скрытия внутренней реализации объекта от внешнего мира. Каждый объект в системе должен быть скрыт от других объектов и должен предоставлять только те методы и свойства, которые необходимы для общения с другими объектами. Это обеспечивает защиту объектов от неправильного использования и изменения другими компонентами системы.

2. Полиморфизм – это возможность объекта при определенных условиях обладать разным поведением. При использовании полиморфизма, объект может представлять собой более общий тип, но иметь разные функции в зависимости от конкретного контекста. Полиморфизм делает код более гибким и уменьшает количество необходимого кода.

3. Абстракция – это выделение существенных характеристик объекта и отброс лишних подробностей. Абстракция позволяет создавать упрощенную модель объекта, которая может быть использована для создания еще более сложных систем. При использовании абстракции, мы сосредотачиваемся на сильны сторонах объекта, игнорируя его слабые стороны, что делает моделирование системы более удобным и понятным.

Эти три основных принципа ООП образуют основу для проектирования систем, которые могут быть расширены, поддерживаемы и возможно, более эффективными и надежными по сравнению с системами, созданными без использования ООП, документация об основных понятиях ООП представлена в списке используемой литературы [1].

1.4. ИСПОЛЬЗОВАНИЕ WINDOWS FORMS

Для создания оконного приложения была использована Windows forms. Windows Forms - новый стиль построения приложения на базе классов .NET Framework class library. Они имеют собственную модель программирования, которая более совершеннее, чем модели, основанные на Win32 API или MFC, и они выполняются в управляемой среде .NET Common Language Runtime (CLR). Эта статья дает представление о том, что такое Windows Forms,

рассматривая ее от модели программирования до Microsoft Intermediate Language и JIT-транслятора. Главная выгода от написания Windows-приложений с использованием Windows Forms — это то, что Windows Forms создают более однородную структуру программную модель и устраняют многие ошибки и противоречия от использования Windows API. Например, каждый опытный программист под Windows знает, что некоторые стили окна могут применяться только к окну, когда оно уже создано. Windows Forms в значительной степени устраняют такое противоречие. Если вы хотите существующему окну задать стиль, который может быть присвоен только в момент создания окна, то Windows Forms спокойно уничтожит окно и вновь создаст его с указанным стилем. Кроме того, .NET Framework class library намного более богаче, чем Windows API, и когда вы будете писать приложения, используя Windows Forms, вы получите в распоряжение больше возможностей. Написание приложения с использованием Windows Forms потребует меньшего количества кода, чем приложения, которые используют Windows API или MFC [2].

Также архитектура Windows Forms включает в себя множество встроенных элементов, таких как кнопки, текстовые поля, метки, списки и другие, которые можно легко использовать и комбинировать для создания пользовательских интерфейсов. Основой Windows Forms является модель программирования, где действия пользователя (например, нажатие кнопок) генерируют события, на которые приложение может реагировать. Windows Forms также поддерживает различные методы размещения элементов на форме, включая фиксированные размеры, автоматическое размещение и использование контейнеров для динамического изменения размеров и расположения элементов. Windows Forms поддерживает специальные возможности для средств чтения с экрана и средств голосового ввода с голосовыми командами. Однако пользовательский интерфейс необходимо разрабатывать с учетом специальных возможностей. Элементы управления

Windows Forms предоставляют различные свойства для обработки специальных возможностей.

1.5. ИСПОЛЬЗОВАНИЕ DLL, ЕГО ПРЕИМУЩЕСТВА И НЕДОСТАТКИ

Библиотека DLL - это библиотека, содержащая код и данные, которые могут использоваться несколькими программами одновременно. Например, в Windows операционных системах библиотека DLL Comdlg32 выполняет общие функции, связанные с диалогом. Каждая программа может использовать функции, содержащиеся в этой библиотеке DLL, для реализации диалогового окна "Открыть". Это помогает повысить эффективность повторного использования кода и эффективного использования памяти.

С помощью библиотеки DLL программу можно разделить на отдельные компоненты. Например, программа учета может быть продана модулем. Каждый модуль можно загрузить в основную программу во время выполнения, если он установлен. Так как модули являются отдельными, время загрузки программы ускоряется. Модуль загружается только при запросе этой функции.

Кроме того, обновления проще применять к каждому модулю, не затрагивая другие части программы. Например, у вас может быть программа заработной платы, и налоговые ставки меняются каждый год. Если эти изменения изолированы в библиотеке DLL, можно применить обновление, не требуя сборки или установки всей программы еще раз.

В следующем списке описаны некоторые преимущества, которые предоставляются, когда программа использует библиотеку DLL:

- Использует меньше ресурсов. Если несколько программ используют ту же библиотеку функций, библиотека DLL может уменьшить дублирование кода, загруженного на диск и в физической памяти. Это может значительно повлиять на производительность не только программы, выполняемой на

переднем плане, но и других программ, работающих в Windows операционной системе.

- Повышение уровня модульной архитектуры. Библиотека DLL помогает повысить уровень разработки модульных программ. Она помогает разрабатывать крупные программы, для которых требуется несколько языковых версий, или программы, для которых требуется модульная архитектура. Примером модульной программы является программа учета с множеством модулей, которые можно динамически загрузить во время выполнения.
- Упрощает развертывание и установку. Если функции в библиотеке DLL требуется обновление или исправление, для развертывания и установки библиотеки DLL не требуется повторное связывание программы с библиотекой DLL. Кроме того, если несколько программ используют ту же библиотеку DLL, все эти программы будут пользоваться преимуществами обновления или исправления. Эта проблема может возникать чаще при использовании библиотеки DLL сторонних разработчиков, которая регулярно обновляется или исправлена.

Единственным крупным недостатком можно считать тот факт, что программа, использующая dll-библиотеку, не может иметь полный функционал в ее отсутствии. Если приложение использует функцию в библиотеке, или одна библиотека использует функцию из другой библиотеки, то получается зависимость, из-за которой приложение или библиотека становятся зависимыми, теряют свою самодостаточность. Соответственно при следующих событиях происходят ошибки:

– Соответствующая библиотека обновилась до новой версии.

– Внесены изменения в зависимую dll-библиотеку.

– Соответствующий dll-файл перезаписывался с более ранней версией.

– Соответствующий dll-файл не найден системой или удален с компьютера.

Обычно эти действия называются конфликтами dll-библиотек. Если не обеспечивается обратная совместимость, программа не может быть успешно запущена. Такие действия называют конфликтом dll-библиотек. Программа может быть успешно запушена и нормально использоваться только в случае обеспечения обратной совместимости. Для предотвращения таких ситуация разработчики должны строго следить за управлением версиями библиотек DLL и их зависимостями, обеспечивая совместимость между различными версиями приложений и используемыми ими библиотеками.

1.6. ОБЫЧНЫЕ И АССОЦИАТИВНЫЕ КОНТЕЙНЕРЫ

C++ предоставляет программисту две основные категории структур данных - это обычные и ассоциативные контейнеры.

Обычные контейнеры и ассоциативные контейнеры в C++ отличаются своей основной функцией. Обычные контейнеры используются для хранения элементов данных в последовательности или коллекции, и имеют доступ к этим элементам через итераторы или индексы. Ассоциативные контейнеры используют пары ключ-значение для хранения элементов данных и имеют доступ к ним через ключ.

Вот некоторые основные отличия между этими двумя типами контейнеров:

1. Доступ к элементам: Обычные контейнеры имеют доступ к элементам по индексу или итератору, тогда как ассоциативные контейнеры имеют доступ к ним только через ключ.

2. Сортировка: Ассоциативные контейнеры сортируют ключи автоматически для обеспечения более быстрого поиска, тогда как обычные контейнеры не сортируют элементы автоматически.

3. Уникальность: В ассоциативных контейнерах ключи должны быть уникальными, тогда как в обычных контейнерах могут быть дубликаты элементов.

4. Скорость доступа: Ассоциативные контейнеры обеспечивают поиск элементов по ключу за время O(log n), тогда как обычные контейнеры обеспечивают доступ к элементам за время O(1).

5. Размер: Обычные контейнеры лучше подходят для хранения большого количества элементов, тогда как ассоциативные контейнеры лучше подходят для небольшого количества элементов.

6. Использование: Обычные контейнеры могут использоваться для различных целей, тогда как ассоциативные контейнеры часто используются для организации данных в структуры типа словарей и таблиц.

Оба типа контейнеров предоставляют разработчикам гибкость и эффективность при работе с элементами данных, но выбор конкретного типа контейнера зависит от задачи [3].

К обычным контейнерам относятся:

– Vector – контейнер, который хранит элементы в виде динамического массива. Контейнер эффективно обрабатывает произвольную выборку элементов с помощью индексации [] или метода at. Однако вставка элемента в любую позицию, кроме конца вектора, неэффективна. Для этого потребуется сдвинуть все последующие элементы путем копирования их значений. По этой же причине неэффективным является удаление любого элемента, кроме последнего.

– Deque – контейнер, который представляет собой двустороннюю очередь (двунаправленный список) элементов. Контейнер во много аналогичен вектору: элементы хранятся в непрерывной области памяти. Но в отличие от вектора, deque эффективно поддерживает вставку и удаление первого элемента (так же, как и последнего).

– List – контейнер, который хранит элементы в виде двунаправленного списка. Каждый элемент списка содержит три поля: значение элемента, указатель на предшествующий и последующий элементы списка. Однако список не поддерживает произвольного доступа к своим элементам: например,

для выборки n-го элемента нужно последовательно выбрать предыдущие n–1 элементов.

– Array – контейнер, который хранит фиксированное количество элементов в массиве заданного размера.

– Forward_list – контейнер, который хранит элементы в виде однонаправленного списка.

Для вставки и удаления последнего элемента предназначены методы push_back и pop_back. Кроме того, list и deque поддерживают операции вставки и удаления первого элемента контейнера push_front и pop_front. Данные методы не возвращают удаленное значение. Чтобы считать первый элемент, используется метод front, а для считывания последнего элемента – метод back. Кроме этого, все типы контейнеров имеют более общие операции вставки и удаления.

К ассоциативным контейнерам относятся:

– Map – контейнер, который хранит элементы в отсортированном порядке по ключу. Каждому значению сопоставлен один ключ.

– unordered_map – контейнер, который хранит элементы в неупорядоченном списке, где доступ к элементам осуществляется по ключу с помощью хэш-функции.

– set – контейнер, в котором хранятся объекты, упорядоченные по некоторому ключу, являющемуся атрибутом самого объекта. Если в множестве хранятся значения одного из встроенных типов, например int, то ключом является сам элемент.

– unordered_set – контейнер, который хранит уникальные ключи в неупорядоченном списке с помощью хэш-функции.

Также, из базовых контейнеров можно создавать специализированные, с помощью конструкции, называющейся адаптером контейнера. Они обладают более простым интерфейсом, чем обычные контейнеры. Специализированные контейнеры, реализованные в STL:

– stack – характерен доступ к данным только с одного конца, реализуется как вектор, список или двусвязная очередь. Шаблонный класс stack определен как tamplate<class T, class Container = deque<T>> class stack {/*…*/}; где параметр Container задает класс-прототип. По умолчанию для стека прототипом является класс deque. Смысл такой реализации заключается в том, что специализированный класс переопределяет интерфейс класса-прототипа, ограничивая его только методами, необходимыми новому классу. В соответствии со своим значение стек не только не позволяет выполнить произвольный доступ к своим элементам, но даже не дает возможности пошагового перемещения, в связи с чем концепция итераторов в стеке не поддерживается.

– queue – шаблонный класс, является адаптером, который может быть реализован на основе двусвязной очереди (по умолчанию) или списка. Класс vector в качестве прототипа не подходит, так как в нем нет выборки из начала контейнера. Очередь использует для проталкивания данных один конец, а для выталкивания – другой.

– priority_queue – шаблонный класс, поддерживающий такие же операции, что и в классе queue, но реализация класса возможна либо на основе вектора (реализация по умолчанию), либо на основе списка. Очередь с приоритетами отличается от обычной тем, что для извлечения выбирается максимальный элемент из хранимых в контейнере. Поэтому после каждого изменения состояния очереди максимальный элемент из оставшихся сдвигается в начало контейнера. Если очередь с приоритетами организуется для объектов класса, в этом классе следует определить операцию <.

1.7. АЛГОРИТМ EQUAL_RANGE

Алгоритм std::equal_range() возвращает диапазон, содержащий все элементы, эквивалентные value в диапазоне [first, last).

Диапазон [first, last) должен быть хотя бы частично упорядочен относительно value, то есть он должен удовлетворять всем следующим требованиям:

- секционируется относительно element < value или comp(element, value) (то есть все элементы, для которых выражение является true предшествуют всем элементам, для которых выражение является false);
- разбит на части по отношению к !(value < element) или !comp(value, element);
- для всех элементов, если element < value или comp(element, value) - true , тогда !(value < element) или !comp(value, element) - также true.

Полностью отсортированный диапазон отвечает этим критериям. Возвращаемый диапазон определяется двумя итераторами, один из которых указывает на первый элемент, которым является не менее чем value а другой указывает на первый элемент greater чем value.

Этот алгоритм обеспечивает эффективный способ нахождения диапазона, содержащего все элементы, эквивалентные value, в упорядоченной последовательности. Он часто используется в ситуациях, когда требуется найти все вхождения заданного значения в отсортированном контейнере или массиве.

Применение std::equal_range() может быть полезным, например, при реализации поиска по индексу в отсортированных базах данных или при работе с отсортированными списками. Он обеспечивает быстрый доступ к интервалу, содержащему все эквивалентные элементы, что делает его ценным инструментом для эффективной обработки данных [4].

1.8. АЛГОРИТМ SET_INTERSECTION

Алгоритм std::set_intersection из стандартной библиотеки C++ предназначен для нахождения пересечения двух отсортированных диапазонов. Пересечение множеств — это набор элементов, которые присутствуют одновременно в обоих множествах. Этот алгоритм, являясь

частью заголовочного файла <algorithm>, широко используется для выполнения операций над множествами [5].

Основная функция std::set_intersection принимает два диапазона, представленных итераторами, и возвращает диапазон с пересекающимися элементами. Важно, чтобы исходные последовательности были отсортированы, иначе результаты будут некорректными. Существует два варианта функции: первый использует оператор сравнения по умолчанию (оператор меньше), а второй позволяет задать пользовательскую функцию сравнения.

Алгоритм последовательно сравнивает элементы обоих диапазонов. Если текущие элементы равны, они добавляются в выходной диапазон. Если не равны, алгоритм продвигается вперед в том диапазоне, элемент которого меньше. Такой подход обеспечивает линейную сложность O(N), где N — суммарное количество элементов в обоих диапазонах.

std::set_intersection находит применение в различных задачах, связанных с обработкой множеств, таких как поиск общих элементов в базах данных, анализ текстов и больших данных. Например, для нахождения общих слов или фраз между документами.

При использовании std::set_intersection следует учитывать, что исходные диапазоны должны быть отсортированы, а выходной контейнер — достаточно велик для всех элементов пересечения. Также важно правильно реализовать пользовательскую функцию сравнения, если она используется. Это обеспечит корректное выполнение алгоритма и предотвратит непредвиденные ошибки при работе с данными.

Важно отметить, что std::set_intersection возвращает только уникальные элементы, присутствующие в обоих исходных диапазонах. Если один из исходных диапазонов содержит несколько эквивалентных элементов, они будут обработаны только один раз в результирующем диапазоне.

Этот алгоритм также может быть использован для определения различий между двумя множествами. Например, если применить

std::set_intersection к двум множествам и затем удалить из первого множества элементы результирующего диапазона, мы получим только те элементы первого множества, которые отсутствуют во втором множестве.

Важно отметить, что алгоритм std::set_intersection завершает свою работу, когда один из исходных диапазонов достигает своего конца. Это означает, что, если один из диапазонов короче другого, алгоритм прекратит выполнение, когда короткий диапазон будет полностью обработан.

Помимо стандартной библиотеки C++, алгоритм поиска пересечения множеств может быть реализован и вручную. Например, это можно сделать с помощью цикла сравнения элементов двух отсортированных массивов, что может быть полезно в случае использования других языков программирования или сред разработки.

Алгоритм std::set_intersection может быть использован не только для операций с множествами, но и в различных алгоритмических задачах, таких как поиск общих элементов в отсортированных массивах или списке чисел. Это позволяет решать широкий спектр задач, включая задачи из области анализа данных, оптимизации и машинного обучения.

Благодаря эффективной линейной сложности, алгоритм std::set_intersection хорошо масштабируется даже для больших наборов данных. Это делает его предпочтительным выбором в задачах, где требуется обработка больших объемов информации, таких как анализ логов, сортировка и фильтрация данных из баз данных и других источников.

Для удобства использования и предотвращения ошибок следует убедиться, что исходные диапазоны не только отсортированы, но и лишены дубликатов. Это гарантирует корректное выполнение алгоритма и избежание непредвиденных результатов.

ГЛАВА 2. ПРОГРАММНАЯ РЕАЛИЗАЦИЯ

2.1. ОПИСАНИЕ РАБОТЫ ПРОГРАММЫ И ВИЗУАЛИЗАЦИЯ

Рассматриваемый в данной курсовой работе программный продукт, является базой данных книжного магазина. Программа предназначена для удобного управления информацией о книгах в магазине. Она представляет графический интерфейс для добавления новых книг и просмотра уже имеющихся с возможностью сортировки и фильтрации по жанрам книг.

Взаимодействие между пользователем и базой данных книжного магазина осуществляется через графический интерфейс и элементы управления, которые позволяют вводить, отображать и манипулировать данными, созданные на основе Windows Forms.

При запуске приложения пользователь видит основную форму, содержащую элементы управления, такие как кнопки, позволяющие выбрать одно из действий. Основная форма представляет собой окно, которое предоставляет пользователю два варианта: «Посмотреть данные» и «Добавить запись» (рис. 2.1). Функции обработки нажатия кнопок прописаны в Приложении 1.

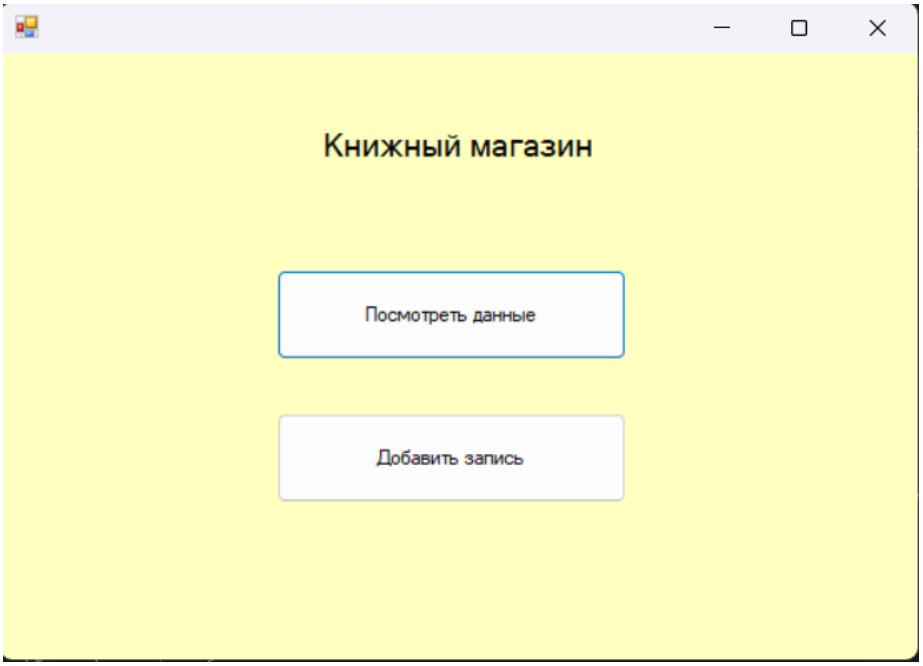

Рис. 2.1 Главный экран

При нажатии кнопки «Посмотреть данные» пользователю доступно окно просмотра всех имеющихся книг (рис. 2.2), которые можно отсортировать по цене (рис. 2.3), категории (рис. 2.4) и автору (рис. 2.5). Функция представления данных определенной категории (рис. 2.6) позволяет фильтровать и просматривать только те товары, которые принадлежат к выбранной категории, обеспечивая более целенаправленный и быстрый доступ к информации. Также доступен просмотр количества книг любой из категорий и удаления выбранной книги (рис. 2.1.7). Для данной формы были использованы следующие элементы управления: кнопки «По стоимости», «По категории», «По автору», «Показать» и «Удалить», выпадающие списки для быстрого выбора фильтрации по категориям и удаления книг, и текстовые поля, содержащие информацию о количестве книг (Приложение 2).

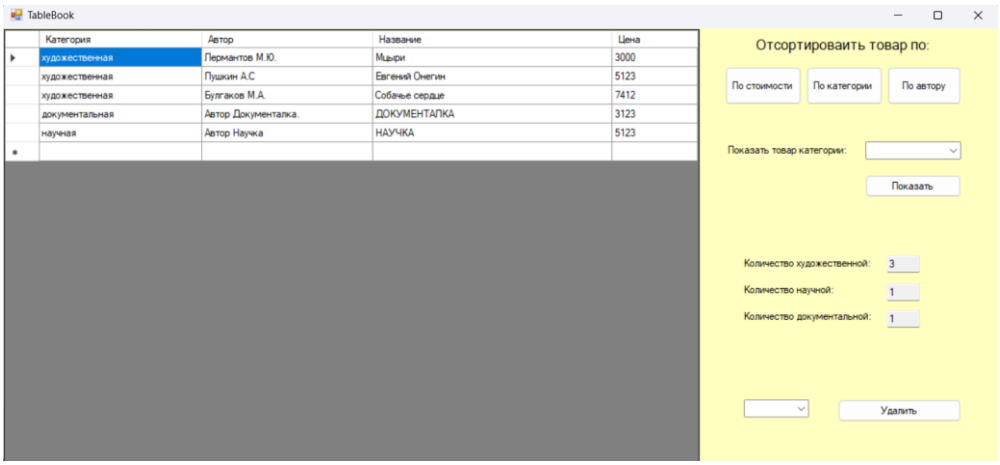

Рис. 2.2 Просмотр всех данных в магазине

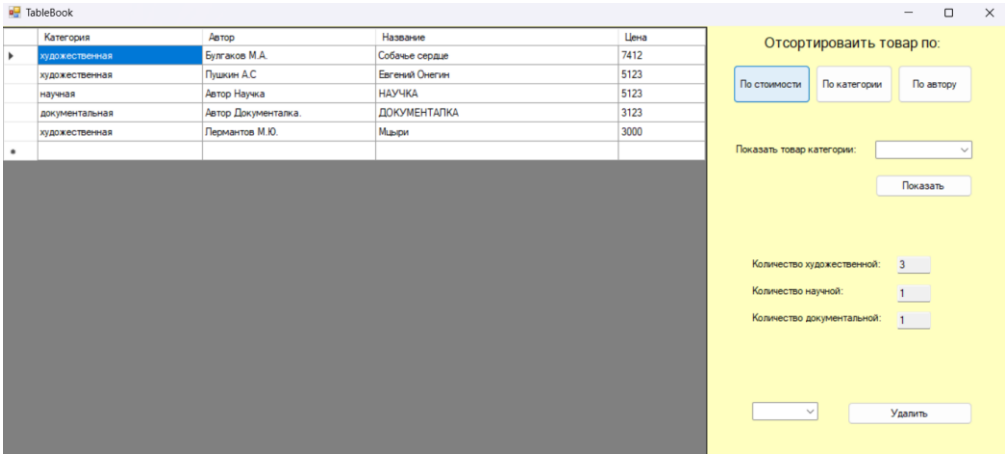

Рис. 2.3 Сортировка книг по стоимости

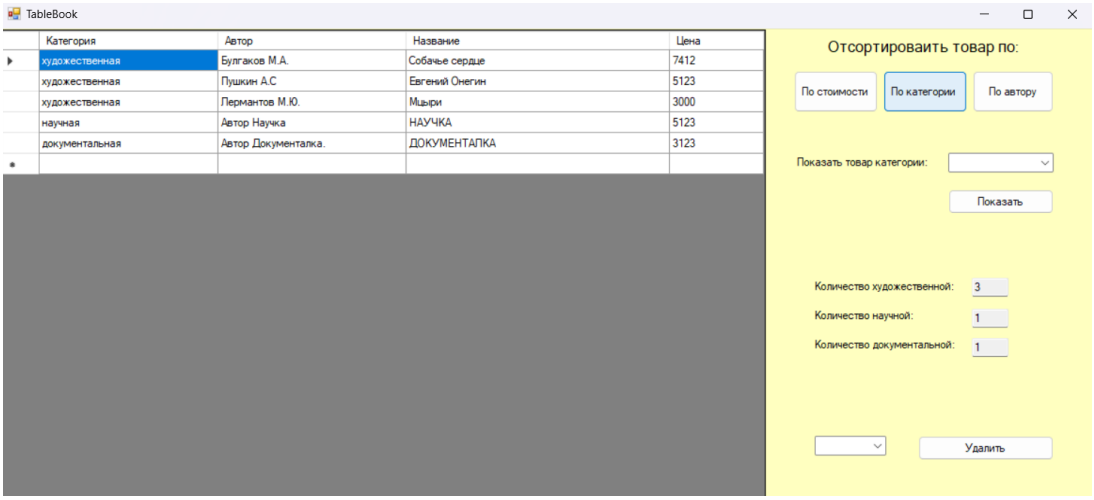

Рис. 2.4 Сортировка книг по категории

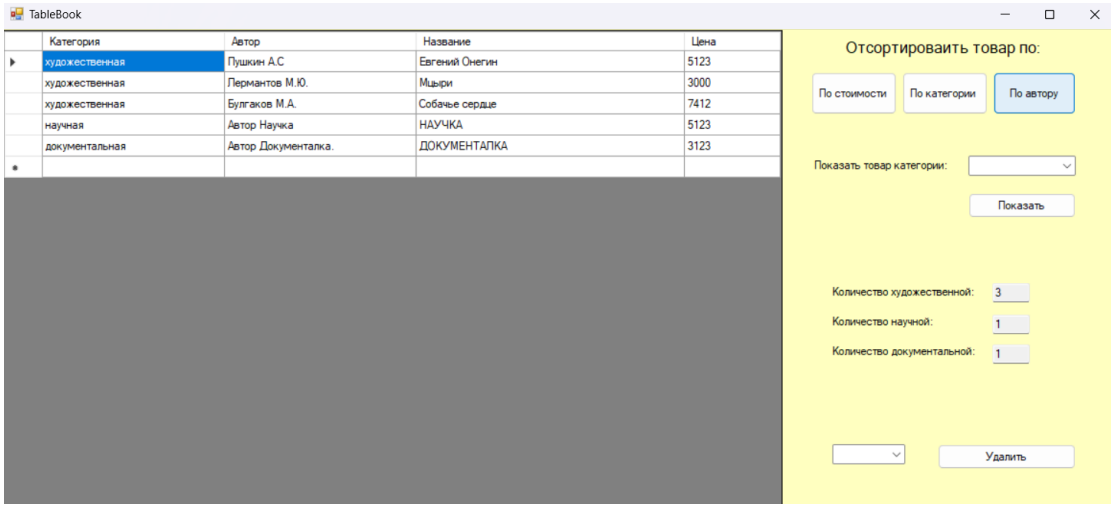

Рис. 2.5 Сортировка книг по автору

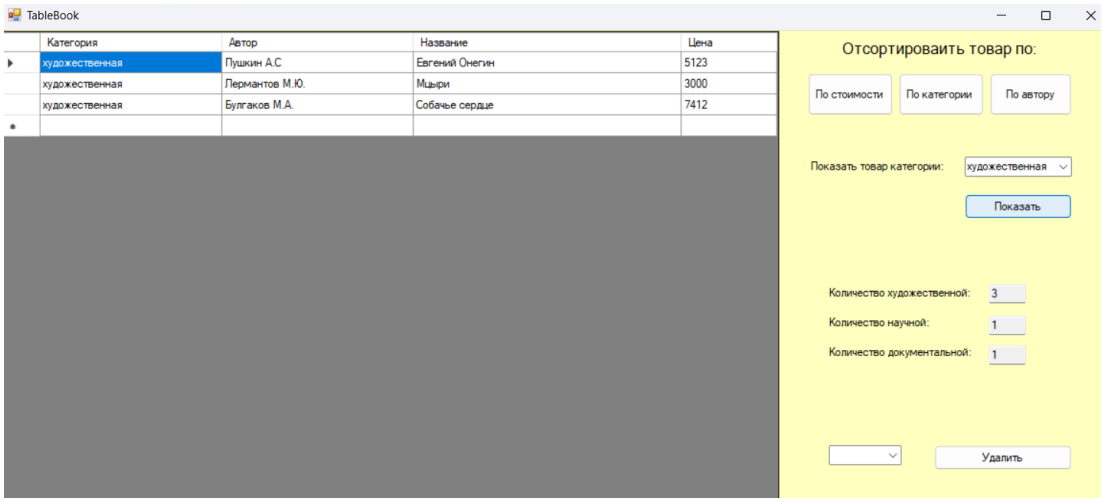

Рис. 2.6 Показать товары определенной категории

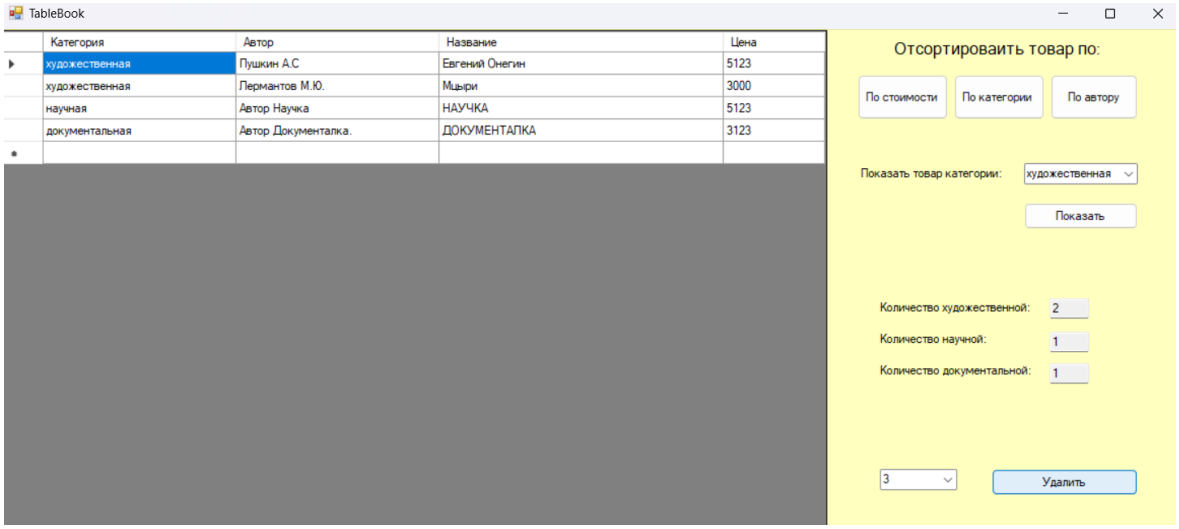

Рис. 2.7 Удаление данных из базы

При нажатии кнопки «Добавить запись» пользователю доступно окно, предназначенное для ввода информации о новой книге (рис. 2.8). Это окно содержит поля, в которые пользователь может вводить следующие данные: категория, автор, название и цена. В поле «Категория» реализован выпадающий список, который содержит все доступные категории книг и предоставляет быстрый доступ к нужной категории без необходимости ввода текста вручную. После заполнения всех необходимых полей пользователь нажимает кнопку «Добавить», в результате чего информация о новой книге добавляется в базу данных. Программная реализация данной формы прописана в Приложении 3.

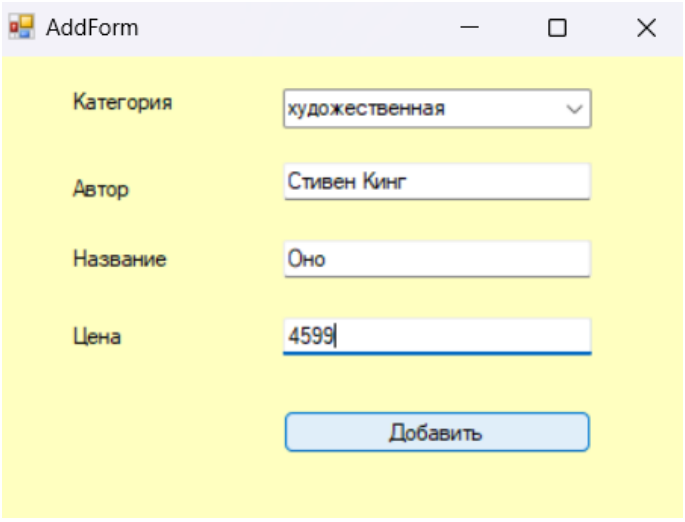

Рис. 2.8 Добавление книг в базу данных

ЗАКЛЮЧЕНИЕ

В результате проведенной работы была разработана база данных книжного магазина, основанная на принципах объектно-ориентированного программирования. При разработки данного программного продукта были использованы принципы ООП и технологии C++/CLI .NET framework. Несмотря на свои скромные размеры, благодаря применению объектно-ориентированного подхода к проектированию, функционал данной программы всегда можно расширить, прибегая к минимальным изменениям существующего кода. Использование DLL библиотеки позволяет более гибко разрабатывать и расширять интерфейс программы.

Таким образом мы на практике научились проектировать и реализовывать программные продукты с использованием объектно-ориентированного подхода, изучили особенности работы с DLL библиотеками и разработали современный интерфейс на платформе .NET.

Программа работает с использованием интерфейса, написанного на windows forms, что позволяет пользователям быстро освоить интерфейс и начать пользоваться базой данных.

Итогом курсовой работы можно считать закрепление знаний в области создания классов, шаблонов, навыков программирования на языке C++, полученных в ходе курса «объектно-ориентированное программирование».

СПИСОК ЛИТЕРАТУРЫ

1. Романов С.С. Ключевые понятия и особенности объектно-ориентированного программирования // Таврическиӣ научныӣ обозреватель. 2016. No12-2 (17). [Электронный ресурс] – https://cyberleninka.ru/article/n/klyuchevye-ponyatiya-i-osobennosti-obektno-orientirovannogo-programmirovaniya – Режим доступа: свободный. Дата обращения: 20.05.2024 г.

2. STL: стандартная библиотека шаблонов C++ [Электронныӣ ресурс/статья] – https://tproger.ru/articles/stl-cpp – Режим доступа: свободный. Дата обращения: 18.05.2024 г.

3. Официальныӣ саӣт Microsoft [Электронныӣ ресурс] – https://learn.microsoft.com – Режим доступа: свободный. Дата обращения: 18.05.2024 г.

4. Официальныӣ саӣт Cppreference [Электронныӣ ресурс] – https://ru.cppreference.com/w/ – Режим доступа: свободный. Дата обращения: 21.05.2024 г.

5. Официальный саӣт Habr [Электронныӣ ресурс] – https://habr.com/ru/companies/piter/articles/418469/ – Режим доступа: свободный. Дата обращения: 21.05.2024 г.

ПРИЛОЖЕНИЕ 1.

ЗАГОЛОВОЧНЫЙ ФАЙЛ FirstForm.h

```
#pragma once

#include <msclr\marshal_cppstd.h>
#include "dll/dll.h"
#pragma comment(lib, "bin\\dll.lib")

namespace Cursovaya {

	using namespace System;
	using namespace System::ComponentModel;
	using namespace System::Collections;
	using namespace System::Windows::Forms;
	using namespace System::Data;
	using namespace System::Drawing;

	public ref class FirstForm : public
System::Windows::Forms::Form
	{
	public:
		FirstForm(void)
		{
			InitializeComponent();
		}

	protected:
		~FirstForm()
		{
			if (components)
			{
				delete components;
			}
		}
	private: System::Windows::Forms::Button^ LookBut;
	private: System::Windows::Forms::Button^ AddBut;

	protected:

	protected:


	private: System::Windows::Forms::Label^ label1;

	private: System::ComponentModel::IContainer^ components;
	protected:

	private:
```

```
     #pragma region Windows Form Designer generated code
                 void InitializeComponent(void)
                 {
                          this->LookBut            =            (gcnew
System::Windows::Forms::Button());
                          this->AddBut             =            (gcnew
System::Windows::Forms::Button());
                          this->label1             =            (gcnew
System::Windows::Forms::Label());
                          this->SuspendLayout();
                          //
                          // LookBut
                          //
                          this->LookBut->Location                    =
System::Drawing::Point(166, 130);
                          this->LookBut->Name = L"LookBut";
                          this->LookBut->Size                        =
System::Drawing::Size(210, 54);
                          this->LookBut->TabIndex = 1;
                          this->LookBut->Text = L"Посмотреть данные";
                          this->LookBut->UseVisualStyleBackColor     =
true;
                          this->LookBut->Click        +=        gcnew
System::EventHandler(this, &FirstForm::LookBut_Click);
                          //
                          // AddBut
                          //
                          this->AddBut->Location                     =
System::Drawing::Point(166, 216);
                          this->AddBut->Name = L"AddBut";
                          this->AddBut->Size                         =
System::Drawing::Size(210, 54);
                          this->AddBut->TabIndex = 2;
                          this->AddBut->Text = L"Добавить запись";
                          this->AddBut->UseVisualStyleBackColor = true;
                          this->AddBut->Click         +=         gcnew
System::EventHandler(this, &FirstForm::AddBut_Click);
                          //
                          // label1
                          //
                          this->label1->AutoSize = true;
                          this->label1->Font          =         (gcnew
System::Drawing::Font(L"Microsoft      Sans      Serif",      14.25F,
System::Drawing::FontStyle::Regular,
System::Drawing::GraphicsUnit::Point,
                                   static_cast<System::Byte>(204)));
                          this->label1->Location                     =
System::Drawing::Point(190, 43);
                          this->label1->Name = L"label1";
```

```
                        this->label1->Size                              =
System::Drawing::Size(172, 24);
                        this->label1->TabIndex = 4;
                        this->label1->Text = L"Книжный магазин";
                        //
                        // FirstForm
                        //
                        this->AutoScaleDimensions                       =
System::Drawing::SizeF(6, 13);
                        this->AutoScaleMode                             =
System::Windows::Forms::AutoScaleMode::Font;
                        this->AutoSizeMode                              =
System::Windows::Forms::AutoSizeMode::GrowAndShrink;
                        this->BackColor                                 =
System::Drawing::Color::FromArgb(static_cast<System::Int32>(static_cas
t<System::Byte>(255)),
static_cast<System::Int32>(static_cast<System::Byte>(255)),

        static_cast<System::Int32>(static_cast<System::Byte>(192)));
                        this->ClientSize = System::Drawing::Size(551,
364);
                        this->Controls->Add(this->label1);
                        this->Controls->Add(this->AddBut);
                        this->Controls->Add(this->LookBut);
                        this->Font              =              (gcnew
System::Drawing::Font(L"Microsoft       Sans       Serif",       7.875F,
System::Drawing::FontStyle::Regular,
System::Drawing::GraphicsUnit::Point,
                                    static_cast<System::Byte>(0)));
                        this->Location  =  System::Drawing::Point(10,
10);
                        this->Margin                                    =
System::Windows::Forms::Padding(6);
                        this->Name = L"FirstForm";
                        this->Text = L"  ";
                        this->Load              +=              gcnew
System::EventHandler(this, &FirstForm::FirstForm_Load);
                        this->ResumeLayout(false);
                        this->PerformLayout();
                }
     #pragma endregion
        private:  System::Void  FirstForm_Load(System::Object^  sender,
System::EventArgs^ e);
        private:  System::Void  LookBut_Click(System::Object^  sender,
System::EventArgs^ e);
        private:  System::Void  AddBut_Click(System::Object^  sender,
System::EventArgs^ e);
     };

     }
```

ПРИЛОЖЕНИЕ 2.

ЗАГОЛОВОЧНЫЙ ФАЙЛ TableBook.h

```
#pragma once
namespace Cursovaya {
   using namespace System;
   using namespace System::ComponentModel;
   using namespace System::Collections;
   using namespace System::Windows::Forms;
   using namespace System::Data;
   using namespace System::Drawing;

   public ref class TableBook : public
System::Windows::Forms::Form
   {
   public:
           TableBook(void)
           {
                   InitializeComponent();
           }

   protected:
           ~TableBook()
           {
                   if (components)
                   {
                           delete components;
                   }
           }
   private: System::Windows::Forms::TextBox^ SciBox;
   private: System::Windows::Forms::TextBox^ NonBox;
   private: System::Windows::Forms::TextBox^ ArtBox;
   protected:
```

```
        protected:
        private: System::Windows::Forms::Label^ label5;
        private: System::Windows::Forms::Label^ label4;
        private: System::Windows::Forms::Label^ label3;
        private: System::Windows::Forms::Button^ show_cat_button;

        private: System::Windows::Forms::Label^ label2;
        private: System::Windows::Forms::Label^ label1;
        private: System::Windows::Forms::ComboBox^ comboBox1;
        private: System::Windows::Forms::Button^ cat_button;

        private: System::Windows::Forms::Button^ price_button;
        private: System::Windows::Forms::Button^ title_button;

        private: System::Windows::Forms::DataGridView^ dataGridView1;
        private:    System::Windows::Forms::DataGridViewTextBoxColumn^
Column1;
        private:    System::Windows::Forms::DataGridViewTextBoxColumn^
Column2;
        private:    System::Windows::Forms::DataGridViewTextBoxColumn^
Column3;
        private:    System::Windows::Forms::DataGridViewTextBoxColumn^
Цена;
        private: System::Windows::Forms::Button^ del_button;

        private: System::Windows::Forms::ComboBox^ comboBox2;

        private:
                System::ComponentModel::Container ^components;

     #pragma region Windows Form Designer generated code
                void InitializeComponent(void)
                {
```

```
                        this->SciBox            =           (gcnew
System::Windows::Forms::TextBox());
                        this->NonBox            =           (gcnew
System::Windows::Forms::TextBox());
                        this->ArtBox            =           (gcnew
System::Windows::Forms::TextBox());
                        this->label5            =           (gcnew
System::Windows::Forms::Label());
                        this->label4            =           (gcnew
System::Windows::Forms::Label());
                        this->label3            =           (gcnew
System::Windows::Forms::Label());
                        this->show_cat_button       =       (gcnew
System::Windows::Forms::Button());
                        this->label2            =           (gcnew
System::Windows::Forms::Label());
                        this->label1            =           (gcnew
System::Windows::Forms::Label());
                        this->comboBox1          =          (gcnew
System::Windows::Forms::ComboBox());
                        this->cat_button         =          (gcnew
System::Windows::Forms::Button());
                        this->price_button        =         (gcnew
System::Windows::Forms::Button());
                        this->title_button        =         (gcnew
System::Windows::Forms::Button());
                        this->dataGridView1       =         (gcnew
System::Windows::Forms::DataGridView());
                        this->Column1           =           (gcnew
System::Windows::Forms::DataGridViewTextBoxColumn());
                        this->Column2           =           (gcnew
System::Windows::Forms::DataGridViewTextBoxColumn());
                        this->Column3           =           (gcnew
System::Windows::Forms::DataGridViewTextBoxColumn());
```

```
                        this->Цена              =              (gcnew
System::Windows::Forms::DataGridViewTextBoxColumn());
                        this->del_button          =          (gcnew
System::Windows::Forms::Button());
                        this->comboBox2           =          (gcnew
System::Windows::Forms::ComboBox());
        (cli::safe_cast<System::ComponentModel::ISupportInitialize^>(t
his->dataGridView1))->BeginInit();
                        this->SuspendLayout();
                        //
                        // SciBox
                        //
                        this->SciBox->Location                    =
System::Drawing::Point(1034, 298);
                        this->SciBox->Name = L"SciBox";
                        this->SciBox->ReadOnly = true;
                        this->SciBox->Size                        =
System::Drawing::Size(39, 20);
                        this->SciBox->TabIndex = 36;
                        //
                        // NonBox
                        //
                        this->NonBox->Location                    =
System::Drawing::Point(1034, 329);
                        this->NonBox->Name = L"NonBox";
                        this->NonBox->ReadOnly = true;
                        this->NonBox->Size                        =
System::Drawing::Size(39, 20);
                        this->NonBox->TabIndex = 35;
                        //
                        // ArtBox
                        //
                        this->ArtBox->Location                    =
System::Drawing::Point(1034, 265);
```

```
                        this->ArtBox->Name = L"ArtBox";
                        this->ArtBox->ReadOnly = true;
                        this->ArtBox->Size                          =
System::Drawing::Size(39, 20);
                        this->ArtBox->TabIndex = 34;
                        //
                        // label5
                        //
                        this->label5->AutoSize = true;
                        this->label5->Location                      =
System::Drawing::Point(864, 298);
                        this->label5->Name = L"label5";
                        this->label5->Size                          =
System::Drawing::Size(112, 13);
                        this->label5->TabIndex = 33;
                        this->label5->Text = L"Количество научной:";
                        //
                        // label4
                        //
                        this->label4->AutoSize = true;
                        this->label4->Location                      =
System::Drawing::Point(864, 329);
                        this->label4->Name = L"label4";
                        this->label4->Size                          =
System::Drawing::Size(156, 13);
                        this->label4->TabIndex = 32;
                        this->label4->Text       =       L"Количество
документальной:";
                        //
                        // label3
                        //
                        this->label3->AutoSize = true;
                        this->label3->Location                      =
System::Drawing::Point(864, 267);
```

```
                        this->label3->Name = L"label3";
                        this->label3->Size                          =
System::Drawing::Size(155, 13);
                        this->label3->TabIndex = 31;
                        this->label3->Text      =      L"Количество
художественной:";
                        //
                        // show_cat_button
                        //
                        this->show_cat_button->Location            =
System::Drawing::Point(1009, 171);
                        this->show_cat_button->Name                =
L"show_cat_button";
                        this->show_cat_button->Size                =
System::Drawing::Size(112, 26);
                        this->show_cat_button->TabIndex = 30;
                        this->show_cat_button->Text = L"Показать";
                        this->show_cat_button-
>UseVisualStyleBackColor = true;
                        this->show_cat_button->Click     +=     gcnew
System::EventHandler(this, &TableBook::show_cat_button_Click);
                        //
                        // label2
                        //
                        this->label2->AutoSize = true;
                        this->label2->Font          =          (gcnew
System::Drawing::Font(L"Microsoft        Sans        Serif",        12,
System::Drawing::FontStyle::Regular,
System::Drawing::GraphicsUnit::Point,
                                    static_cast<System::Byte>(204)));
                        this->label2->Location                     =
System::Drawing::Point(877, 9);
                        this->label2->Name = L"label2";
```

```
                        this->label2->Size                              =
System::Drawing::Size(212, 20);
                        this->label2->TabIndex = 29;
                        this->label2->Text  =  L"Отсортироваить  товар
по:";
                        //
                        // label1
                        //
                        this->label1->AutoSize = true;
                        this->label1->Location                          =
System::Drawing::Point(845, 135);
                        this->label1->Name = L"label1";
                        this->label1->Size                              =
System::Drawing::Size(146, 13);
                        this->label1->TabIndex = 28;
                        this->label1->Text    =    L"Показать    товар
категории:";
                        //
                        // comboBox1
                        //
                        this->comboBox1->FormattingEnabled = true;
                        this->comboBox1->Items->AddRange(gcnew
cli::array<  System::Object^    >(3)  {  L"художественная",  L"научная",
L"документальная" });
                        this->comboBox1->Location                       =
System::Drawing::Point(1009, 132);
                        this->comboBox1->Name = L"comboBox1";
                        this->comboBox1->Size                           =
System::Drawing::Size(112, 21);
                        this->comboBox1->TabIndex = 27;
                        //
                        // cat_button
                        //
```

```
                        this->cat_button->Location                  =
System::Drawing::Point(940, 45);
                        this->cat_button->Name = L"cat_button";
                        this->cat_button->Size                      =
System::Drawing::Size(89, 44);
                        this->cat_button->TabIndex = 25;
                        this->cat_button->Text = L"По категории ";
                        this->cat_button->UseVisualStyleBackColor   =
true;
                        this->cat_button->Click       +=       gcnew
System::EventHandler(this, &TableBook::cat_button_Click);
                        //
                        // price_button
                        //
                        this->price_button->Location                =
System::Drawing::Point(845, 45);
                        this->price_button->Name = L"price_button";
                        this->price_button->Size                    =
System::Drawing::Size(89, 44);
                        this->price_button->TabIndex = 24;
                        this->price_button->Text = L"По стоимости";
                        this->price_button->UseVisualStyleBackColor =
true;
                        this->price_button->Click      +=      gcnew
System::EventHandler(this, &TableBook::price_button_Click);
                        //
                        // title_button
                        //
                        this->title_button->Location                =
System::Drawing::Point(1035, 45);
                        this->title_button->Name = L"title_button";
                        this->title_button->Size                    =
System::Drawing::Size(86, 44);
                        this->title_button->TabIndex = 26;
```

```
                        this->title_button->Text = L"По автору";
                        this->title_button->UseVisualStyleBackColor =
true;
                        this->title_button->Click      +=      gcnew
System::EventHandler(this, &TableBook::title_button_Click);
                        //
                        // dataGridView1
                        //
                        this->dataGridView1->BackgroundColor        =
System::Drawing::Color::Gray;
                        this->dataGridView1-
>ColumnHeadersHeightSizeMode                                        =
System::Windows::Forms::DataGridViewColumnHeadersHeightSizeMode::AutoS
ize;
                        this->dataGridView1->Columns->AddRange(gcnew
cli::array< System::Windows::Forms::DataGridViewColumn^  >(4) {
                                    this->Column1,
                                            this->Column2,         this-
>Column3, this->Цена
                        });
                        this->dataGridView1->Location               =
System::Drawing::Point(1, 1);
                        this->dataGridView1->Name = L"dataGridView1";
                        this->dataGridView1->Size                   =
System::Drawing::Size(814, 508);
                        this->dataGridView1->TabIndex = 23;
                        //
                        // Column1
                        //
                        this->Column1->HeaderText = L"Категория";
                        this->Column1->Name = L"Column1";
                        this->Column1->Width = 190;
                        //
                        // Column2
```

```
                        //
                        this->Column2->HeaderText = L"Автор";
                        this->Column2->Name = L"Column2";
                        this->Column2->Width = 200;
                        //
                        // Column3
                        //
                        this->Column3->HeaderText = L"Название";
                        this->Column3->Name = L"Column3";
                        this->Column3->Width = 280;
                        //
                        // Цена
                        //
                        this->Цена->HeaderText = L"Цена";
                        this->Цена->Name = L"Цена";
                        //
                        // del_button
                        //
                        this->del_button->Location                  =
System::Drawing::Point(977, 433);
                        this->del_button->Name = L"del_button";
                        this->del_button->Size                      =
System::Drawing::Size(144, 26);
                        this->del_button->TabIndex = 37;
                        this->del_button->Text = L"Удалить";
                        this->del_button->UseVisualStyleBackColor   =
true;
                        this->del_button->Click       +=       gcnew
System::EventHandler(this, &TableBook::del_button_Click);
                        //
                        // comboBox2
                        //
                        this->comboBox2->FormattingEnabled = true;
```

```
                        this->comboBox2->Location                   =
System::Drawing::Point(867, 433);
                        this->comboBox2->Name = L"comboBox2";
                        this->comboBox2->Size                       =
System::Drawing::Size(76, 21);
                        this->comboBox2->TabIndex = 38;
                        //
                        // TableBook
                        //
                        this->AutoScaleDimensions                   =
System::Drawing::SizeF(6, 13);
                        this->AutoScaleMode                         =
System::Windows::Forms::AutoScaleMode::Font;
                        this->BackColor                             =
System::Drawing::Color::FromArgb(static_cast<System::Int32>(static_cas
t<System::Byte>(255)),
static_cast<System::Int32>(static_cast<System::Byte>(255)),

        static_cast<System::Int32>(static_cast<System::Byte>(192)));
                        this->ClientSize                            =
System::Drawing::Size(1165, 508);
                        this->Controls->Add(this->comboBox2);
                        this->Controls->Add(this->del_button);
                        this->Controls->Add(this->SciBox);
                        this->Controls->Add(this->NonBox);
                        this->Controls->Add(this->ArtBox);
                        this->Controls->Add(this->label5);
                        this->Controls->Add(this->label4);
                        this->Controls->Add(this->label3);
                        this->Controls->Add(this->show_cat_button);
                        this->Controls->Add(this->label2);
                        this->Controls->Add(this->label1);
                        this->Controls->Add(this->comboBox1);
                        this->Controls->Add(this->cat_button);
```

```
                        this->Controls->Add(this->price_button);
                        this->Controls->Add(this->title_button);
                        this->Controls->Add(this->dataGridView1);
                        this->Name = L"TableBook";
                        this->Text = L"TableBook";
                        this->Activated           +=           gcnew
System::EventHandler(this, &TableBook::TableBook_Activated);
                        this->Load              +=              gcnew
System::EventHandler(this, &TableBook::TableBook_Load);

        (cli::safe_cast<System::ComponentModel::ISupportInitialize^>(t
his->dataGridView1))->EndInit();
                        this->ResumeLayout(false);
                        this->PerformLayout();

                }
     #pragma endregion

     private:  System::Void  price_button_Click(System::Object^  sender,
System::EventArgs^ e);
     private:  System::Void  cat_button_Click(System::Object^  sender,
System::EventArgs^ e);
     private:  System::Void  title_button_Click(System::Object^  sender,
System::EventArgs^ e);
     private:   System::Void   TableBook_Load(System::Object^   sender,
System::EventArgs^ e);
     private:    System::Void    show_cat_button_Click(System::Object^
sender, System::EventArgs^ e);
     private: System::Void TableBook_Activated(System::Object^ sender,
System::EventArgs^ e);
     private:  System::Void  del_button_Click(System::Object^  sender,
System::EventArgs^ e);
     };
     }
```

ПРИЛОЖЕНИЕ 3.

ЗАГОЛОВОЧНЫЙ ФАЙЛ AddForm.h

```
#pragma once

namespace Cursovaya {

	using namespace System;
	using namespace System::ComponentModel;
	using namespace System::Collections;
	using namespace System::Windows::Forms;
	using namespace System::Data;
	using namespace System::Drawing;

	public ref class AddForm : public System::Windows::Forms::Form
	{
	public:
		AddForm(void)
		{
			InitializeComponent();
		}

	protected:
		~AddForm()
		{
			if (components)
			{
				delete components;
			}
		}
	private: System::Windows::Forms::ComboBox^ categoryBox;
	private: System::Windows::Forms::Button^ add_button;
	protected:

	protected:

	protected:

	private: System::Windows::Forms::Label^ label5;
	private: System::Windows::Forms::Label^ label4;
	private: System::Windows::Forms::Label^ label2;
	private: System::Windows::Forms::TextBox^ titleBox;

	private: System::Windows::Forms::Label^ label1;
	private: System::Windows::Forms::TextBox^ authorBox;

	private: System::Windows::Forms::Label^ Price;
	private: System::Windows::Forms::TextBox^ priceBox;
```

```
        private:
                System::ComponentModel::Container ^components;

#pragma region Windows Form Designer generated code
                void InitializeComponent(void)
                {
                        this->categoryBox = (gcnew
System::Windows::Forms::ComboBox());
                        this->add_button = (gcnew
System::Windows::Forms::Button());
                        this->label5 = (gcnew
System::Windows::Forms::Label());
                        this->label4 = (gcnew
System::Windows::Forms::Label());
                        this->label2 = (gcnew
System::Windows::Forms::Label());
                        this->titleBox = (gcnew
System::Windows::Forms::TextBox());
                        this->label1 = (gcnew
System::Windows::Forms::Label());
                        this->authorBox = (gcnew
System::Windows::Forms::TextBox());
                        this->Price = (gcnew
System::Windows::Forms::Label());
                        this->priceBox = (gcnew
System::Windows::Forms::TextBox());
                        this->SuspendLayout();
                        //
                        // categoryBox
                        //
                        this->categoryBox->FormattingEnabled = true;
                        this->categoryBox->Items->AddRange(gcnew
cli::array< System::Object^  >(3) { L"художественная", L"научная",
L"документальная" });
                        this->categoryBox->Location =
System::Drawing::Point(205, 21);
                        this->categoryBox->Margin =
System::Windows::Forms::Padding(4, 4, 4, 4);
                        this->categoryBox->Name = L"categoryBox";
                        this->categoryBox->Size =
System::Drawing::Size(216, 24);
                        this->categoryBox->TabIndex = 22;
                        //
                        // add_button
                        //
                        this->add_button->Location =
System::Drawing::Point(205, 230);
                        this->add_button->Margin =
System::Windows::Forms::Padding(4, 4, 4, 4);
                        this->add_button->Name = L"add_button";
```

```
                        this->add_button->Size =
System::Drawing::Size(217, 28);
                        this->add_button->TabIndex = 21;
                        this->add_button->Text = L"Добавить";
                        this->add_button->UseVisualStyleBackColor =
true;
                        this->add_button->Click += gcnew
System::EventHandler(this, &AddForm::add_button_Click);
                        //
                        // label5
                        //
                        this->label5->AutoSize = true;
                        this->label5->Location =
System::Drawing::Point(53, 21);
                        this->label5->Margin =
System::Windows::Forms::Padding(4, 0, 4, 0);
                        this->label5->Name = L"label5";
                        this->label5->Size =
System::Drawing::Size(78, 16);
                        this->label5->TabIndex = 20;
                        this->label5->Text = L"Категория ";
                        //
                        // label4
                        //
                        this->label4->AutoSize = true;
                        this->label4->Location =
System::Drawing::Point(53, 123);
                        this->label4->Margin =
System::Windows::Forms::Padding(4, 0, 4, 0);
                        this->label4->Name = L"label4";
                        this->label4->Size =
System::Drawing::Size(73, 16);
                        this->label4->TabIndex = 19;
                        this->label4->Text = L"Название";
                        //
                        // label2
                        //
                        this->label2->AutoSize = true;
                        this->label2->Location =
System::Drawing::Point(231, 180);
                        this->label2->Margin =
System::Windows::Forms::Padding(4, 0, 4, 0);
                        this->label2->Name = L"label2";
                        this->label2->Size = System::Drawing::Size(0,
16);
                        this->label2->TabIndex = 18;
                        //
                        // titleBox
                        //
                        this->titleBox->Location =
System::Drawing::Point(205, 119);
```

```
                        this->titleBox->Margin =
System::Windows::Forms::Padding(4, 4, 4, 4);
                        this->titleBox->Name = L"titleBox";
                        this->titleBox->Size =
System::Drawing::Size(216, 22);
                        this->titleBox->TabIndex = 17;
                        //
                        // label1
                        //
                        this->label1->AutoSize = true;
                        this->label1->Location =
System::Drawing::Point(53, 78);
                        this->label1->Margin =
System::Windows::Forms::Padding(4, 0, 4, 0);
                        this->label1->Name = L"label1";
                        this->label1->Size =
System::Drawing::Size(47, 16);
                        this->label1->TabIndex = 16;
                        this->label1->Text = L"Автор";
                        //
                        // authorBox
                        //
                        this->authorBox->Location =
System::Drawing::Point(205, 69);
                        this->authorBox->Margin =
System::Windows::Forms::Padding(4, 4, 4, 4);
                        this->authorBox->Name = L"authorBox";
                        this->authorBox->Size =
System::Drawing::Size(216, 22);
                        this->authorBox->TabIndex = 15;
                        //
                        // Price
                        //
                        this->Price->AutoSize = true;
                        this->Price->Location =
System::Drawing::Point(53, 174);
                        this->Price->Margin =
System::Windows::Forms::Padding(4, 0, 4, 0);
                        this->Price->Name = L"Price";
                        this->Price->Size = System::Drawing::Size(40,
16);
                        this->Price->TabIndex = 14;
                        this->Price->Text = L"Цена";
                        //
                        // priceBox
                        //
                        this->priceBox->Location =
System::Drawing::Point(205, 170);
                        this->priceBox->Margin =
System::Windows::Forms::Padding(4, 4, 4, 4);
                        this->priceBox->Name = L"priceBox";
```

```
                        this->priceBox->Size =
System::Drawing::Size(216, 22);
                        this->priceBox->TabIndex = 13;
                        //
                        // AddForm
                        //
                        this->AutoScaleDimensions =
System::Drawing::SizeF(8, 16);
                        this->AutoScaleMode =
System::Windows::Forms::AutoScaleMode::Font;
                        this->BackColor =
System::Drawing::Color::FromArgb(static_cast<System::Int32>(static_cas
t<System::Byte>(255)),
static_cast<System::Int32>(static_cast<System::Byte>(255)),

        static_cast<System::Int32>(static_cast<System::Byte>(192)));
                        this->ClientSize = System::Drawing::Size(493,
305);
                        this->Controls->Add(this->categoryBox);
                        this->Controls->Add(this->add_button);
                        this->Controls->Add(this->label5);
                        this->Controls->Add(this->label4);
                        this->Controls->Add(this->label2);
                        this->Controls->Add(this->titleBox);
                        this->Controls->Add(this->label1);
                        this->Controls->Add(this->authorBox);
                        this->Controls->Add(this->Price);
                        this->Controls->Add(this->priceBox);
                        this->Margin =
System::Windows::Forms::Padding(4, 4, 4, 4);
                        this->Name = L"AddForm";
                        this->Text = L"AddForm";
                        this->Load += gcnew
System::EventHandler(this, &AddForm::AddForm_Load);
                        this->ResumeLayout(false);
                        this->PerformLayout();

                }
#pragma endregion
private: System::Void add_button_Click(System::Object^ sender,
System::EventArgs^ e);
private: System::Void AddForm_Load(System::Object^ sender,
System::EventArgs^ e) {
}
};
}
```

ПРИЛОЖЕНИЕ 4.

ЗАГОЛОВОЧНЫЙ ФАЙЛ dll.h

```
#ifdef DLL_EXPORTS
#define DLL_API __declspec(dllexport)
#else
#define DLL_API __declspec(dllimport)
#endif

#include "framework.h"
#include<iostream>
#include<fstream>
#include<vector>
#include<unordered_map>
#include<string>
#include<algorithm>
#include<string>

using namespace std;




class DLL_API Book {
public:
   string author, cat, title;
   int price;
   Book();
   Book(string a, string c, string t, int p);
   Book(Book& b);
   virtual Book* copy();
   virtual ~Book();
};

class DLL_API Art : public Book
{
public:
   Art();
   Art(Art& a);
   Art(string a, string c, string t, int p);
   Art* copy();
   ~Art();
};

class DLL_API Sci : public Book
{
public:
   Sci();
   Sci(string a, string c, string t, int p);
```

```
    Sci(Sci& p);
    Sci* copy();
    ~Sci();
};

class DLL_API Non : public Book
{
public:
    Non();
    Non(Non& p);
    Non(string a, string c, string t, int p);
    Non* copy();
    ~Non();
};

class DLL_API Library
{
public:
    vector<Book*> v;
    vector<Book*> b;
    int del_book;
    void del();
    Library();
    Library(const Library& p);
    Library& operator=(const Library& db);
    void price_sort();
    void author_sort();
    void cat_sort();
    void add(string a, string c, string t, int p);
    void choose_cat(string cat);
    int count_art();
    int count_sci();
    int count_non();
    ~Library();
};

extern DLL_API Library lib;
```

ПРИЛОЖЕНИЕ 5.

ЗАГОЛОВОЧНЫЙ ФАЙЛ framework.h

```
#pragma once

#define WIN32_LEAN_AND_MEAN
#include <windows.h>
```

ПРИЛОЖЕНИЕ 6.

ИСХОДНЫЙ ФАЙЛ AddForm.cpp

```
     #include "FirstForm.h"
     #include "AddForm.h"


     using namespace System;
     using namespace System::Windows::Forms;

     System::Void  Cursovaya::AddForm::add_button_Click(System::Object^
sender, System::EventArgs^ e)
     {
        setlocale(LC_ALL, "");
        string   c   =   msclr::interop::marshal_as<string>(categoryBox-
>Text),
                 a    =    msclr::interop::marshal_as<string>(authorBox-
>Text),
                 t    =    msclr::interop::marshal_as<string>(titleBox-
>Text);

        int p;
        try {
                         p = Int32::Parse(priceBox->Text);
        }
        catch (FormatException^) {}

        lib.add(a, c, t, p);
     }
```

ПРИЛОЖЕНИЕ 7.

ИСХОДНЫЙ ФАЙЛ FirstForm.cpp

```
     #include "FirstForm.h"
     #include "TableBook.h"
     #include "AddForm.h"
     #include <msclr\marshal_cppstd.h>

     using namespace System;
     using namespace System::Windows::Forms;
     using namespace Cursovaya;

     [STAThreadAttribute]

     int main()
     {
        ContextMenuStrip;
        Application::SetCompatibleTextRenderingDefault(false);
        Application::EnableVisualStyles();
        FirstForm form;
        Application::Run(gcnew FirstForm);
     }

     Void FirstForm::FirstForm_Load(Object^ sender, EventArgs^ e)
     {
        return Void();
     }

     Void  Cursovaya::FirstForm::LookBut_Click(System::Object^  sender,
System::EventArgs^ e)
     {
        TableBook^ tab = gcnew TableBook();
        tab->Show();
        return Void();
     }

     Void  Cursovaya::FirstForm::AddBut_Click(System::Object^  sender,
System::EventArgs^ e)
     {
        AddForm^ add = gcnew AddForm();
        add->Show();
        return Void();
     }
```

ПРИЛОЖЕНИЕ 8.

ИСХОДНЫЙ ФАЙЛ TableBook.cpp

```
     #include "TableBook.h"
     #include "FirstForm.h"


     System::Void
Cursovaya::TableBook::price_button_Click(System::Object^        sender,
System::EventArgs^ e)
     {
        dataGridView1->Rows->Clear();

        lib.price_sort();

        for (int i = 0; i < lib.v.size(); i++) {
                String^ pr = (lib.v[i]->price).ToString();
                String^                      ct                      =
msclr::interop::marshal_as<String^>((lib.v[i]->cat));
                String^                      aut                     =
msclr::interop::marshal_as<String^>((lib.v[i]->author));
                String^                      tit                     =
msclr::interop::marshal_as<String^>((lib.v[i]->title));
                dataGridView1->Rows->Add(ct, aut, tit, pr);
        }
     }

     System::Void
Cursovaya::TableBook::cat_button_Click(System::Object^          sender,
System::EventArgs^ e)
     {
        dataGridView1->Rows->Clear();

        lib.cat_sort();

        for (int i = 0; i < lib.v.size(); i++) {
                String^ pr = (lib.v[i]->price).ToString();
                String^                      ct                      =
msclr::interop::marshal_as<String^>((lib.v[i]->cat));
                String^                      aut                     =
msclr::interop::marshal_as<String^>((lib.v[i]->author));
                String^                      tit                     =
msclr::interop::marshal_as<String^>((lib.v[i]->title));
                dataGridView1->Rows->Add(ct, aut, tit, pr);
        }
     }
```

```
     System::Void
Cursovaya::TableBook::title_button_Click(System::Object^        sender,
System::EventArgs^ e)
     {
        dataGridView1->Rows->Clear();

        lib.author_sort();

        for (int i = 0; i < lib.v.size(); i++) {
                String^ pr = (lib.v[i]->price).ToString();
                String^                      ct                      =
msclr::interop::marshal_as<String^>((lib.v[i]->cat));
                String^                      aut                     =
msclr::interop::marshal_as<String^>((lib.v[i]->author));
                String^                      tit                     =
msclr::interop::marshal_as<String^>((lib.v[i]->title));
                dataGridView1->Rows->Add(ct, aut, tit, pr);
        }
     }

     System::Void  Cursovaya::TableBook::TableBook_Load(System::Object^
sender, System::EventArgs^ e)
     {
        dataGridView1->Rows->Clear();

        ArtBox->Text = (lib.count_art()).ToString();
        NonBox->Text = (lib.count_non()).ToString();
        SciBox->Text = (lib.count_sci()).ToString();

        for (int i = 0; i < lib.v.size(); i++) {
                String^ pr = (lib.v[i]->price).ToString();
                String^                      ct                      =
msclr::interop::marshal_as<String^>((lib.v[i]->cat));
                String^                      aut                     =
msclr::interop::marshal_as<String^>((lib.v[i]->author));
                String^                      tit                     =
msclr::interop::marshal_as<String^>((lib.v[i]->title));
                dataGridView1->Rows->Add(ct, aut, tit, pr);
        }

     }

     System::Void
Cursovaya::TableBook::show_cat_button_Click(System::Object^     sender,
System::EventArgs^ e)
     {
        dataGridView1->Rows->Clear();

        lib.choose_cat(msclr::interop::marshal_as<string>(comboBox1-
>Text));
```

```
        for (int i = 0; i < lib.b.size(); i++) {
                String^ pr = (lib.b[i]->price).ToString();
                String^                      ct                      =
msclr::interop::marshal_as<String^>((lib.b[i]->cat));
                String^                     aut                      =
msclr::interop::marshal_as<String^>((lib.b[i]->author));
                String^                     tit                      =
msclr::interop::marshal_as<String^>((lib.b[i]->title));
                dataGridView1->Rows->Add(ct, aut, tit, pr);
        }
     }

     System::Void
Cursovaya::TableBook::TableBook_Activated(System::Object^       sender,
System::EventArgs^ e)
     {
        dataGridView1->Rows->Clear();
        comboBox2->Items->Clear();

        ArtBox->Text = (lib.count_art()).ToString();
        NonBox->Text = (lib.count_non()).ToString();
        SciBox->Text = (lib.count_sci()).ToString();

        for (int i = 0; i < lib.v.size(); i++) {
                comboBox2->Items->Add(i + 1);
                String^ pr = (lib.v[i]->price).ToString();
                String^                      ct                      =
msclr::interop::marshal_as<String^>((lib.v[i]->cat));
                String^                     aut                      =
msclr::interop::marshal_as<String^>((lib.v[i]->author));
                String^                     tit                      =
msclr::interop::marshal_as<String^>((lib.v[i]->title));
                dataGridView1->Rows->Add(ct, aut, tit, pr);
        }
     }

     System::Void
Cursovaya::TableBook::del_button_Click(System::Object^          sender,
System::EventArgs^ e)
     {
        lib.del_book                                                    =
atoi((msclr::interop::marshal_as<string>((comboBox2->Text))).c_str()) -
1;
        lib.del();

        dataGridView1->Rows->Clear();
        comboBox2->Items->Clear();

        ArtBox->Text = (lib.count_art()).ToString();
        NonBox->Text = (lib.count_non()).ToString();
        SciBox->Text = (lib.count_sci()).ToString();
```

```
        for (int i = 0; i < lib.v.size(); i++) {
                comboBox2->Items->Add(i + 1);
                String^ pr = (lib.v[i]->price).ToString();
                String^                      ct                      =
msclr::interop::marshal_as<String^>((lib.v[i]->cat));
                String^                     aut                      =
msclr::interop::marshal_as<String^>((lib.v[i]->author));
                String^                     tit                      =
msclr::interop::marshal_as<String^>((lib.v[i]->title));
                dataGridView1->Rows->Add(ct, aut, tit, pr);
        }
     }
```

ПРИЛОЖЕНИЕ 9.

ИСХОДНЫЙ ФАЙЛ dll.cpp

```
#include "dll.h"

struct Leaks { ~Leaks() { _CrtDumpMemoryLeaks(); } };
Leaks leaks;

DLL_API Library lib;

typedef unordered_map<string, int> Mymap;

Book::Book()
{
    author = "";
    cat = "";
    price = 0;
    title = "";
}

Book::Book(string a, string c, string t, int p)
{
    author = a;
    cat = c;
    price = p;
    title = t;
}


Book::Book(Book& b)
{
    author = b.author;
    cat = b.cat;
    price = b.price;
    title = b.title;
}

Book* Book::copy()
{
    return new Book(*this);
}

Book::~Book() {}

Art::Art() {
    cat = "художественная";
}
Art::Art(Art& p) : Book(p) { }
```

```
Art::Art(string a, string c, string t, int p) :Book(a, c, t, p) {}

Art* Art::copy()
{
    return new Art(*this);
}

Art::~Art() {}

Sci::Sci()
{
    cat = "научная";
}

Sci::Sci(string a, string c, string t, int p) :Book(a, c, t, p) {}

Sci::Sci(Sci& p) : Book(p) { }

Sci* Sci::copy() { return new Sci(*this); }

Sci::~Sci() {}

Non::Non() {}

Non::Non(Non& p) : Book(p) { }

Non::Non(string a, string c, string t, int p) :Book(a, c, t, p) {}
Non* Non::copy()
{
    return new Non(*this);
}
Non::~Non()
{
    cat = "документальная";
}

void Library::del()
{
    vector<Book*> vec;
    for (int i = 0; i < v.size(); i++)
    {
        if (i != del_book)
            vec.push_back(v[i]->copy());
    }

    for (int i = 0; i < v.size(); i++)
        delete v[i];
    v.clear();

    for (int i = 0; i < vec.size(); i++)
    {
```

```
            v.push_back(vec[i]->copy());
        }

        for (int i = 0; i < vec.size(); i++)
            delete vec[i];
        vec.clear();
    }

    Library::Library()
    {
        del_book = -1;
        v.push_back(new   Art("Лермантов   М.Ю.",   "художественная",
"Мцыри", 3000));
        v.push_back(new Art("Пушкин А.С", "художественная", "Евгений
Онегин", 5123));
        v.push_back(new   Art("Булгаков   М.А.",   "художественная",
"Собачье сердце", 7412));

        v.push_back(new Non("Автор Документалка.", "документальная",
"ДОКУМЕНТАЛКА", 3123));
        v.push_back(new  Sci("Автор  Научка",  "научная",  "НАУЧКА",
5123));
    }

    Library::Library(const Library& db)
    {
        for (vector<Book*>::iterator it = v.begin(); it != v.end();
it++)
            delete* it;
        v.clear();

        for (vector<Book*>::const_iterator it = db.v.begin(); it !=
db.v.end(); it++)
        {
            Book* new_obj = (*it)->copy();
            v.push_back(new_obj);
        }
    }

    Library& Library::operator=(const Library& db)
    {
        if (this == &db)
            return *this;

        for (vector<Book*>::iterator it = v.begin(); it != v.end();
it++)
            delete* it;
        v.clear();

        for (vector<Book*>::const_iterator it = db.v.begin(); it !=
db.v.end(); it++)
```

```
        {
            Book* new_obj = (*it)->copy();
            v.push_back(new_obj);
        }

        return *this;
    }

    void Library::price_sort()
    {
        sort(v.begin(), v.end(), [](Book* a, Book* b) {return a->price
> b->price; });
    }
    void Library::author_sort()
    {
        sort(v.begin(), v.end(), [](Book* a, Book* b) {return a->author
> b->author; });
    }
    void Library::cat_sort()
    {
        sort(v.begin(), v.end(), [](Book* a, Book* b) {return a->cat >
b->cat; });
    }
    void Library::add(string a, string c, string t, int p)
    {
        setlocale(LC_ALL, "");

        if (c == "художественная")
            v.push_back(new Art(a, c, t, p));

        else if (c == "научная")
            v.push_back(new Sci(a, c, t, p));

        else if (c == "документальная")
            v.push_back(new Non(a, c, t, p));
    }
    void Library::choose_cat(string s)
    {
        setlocale(LC_ALL, "");
        for (int i = 0; i < b.size(); i++)
            delete b[i];
        b.clear();

        cat_sort();

        vector<int> cats, al, need;

        for (vector<Book*>::iterator it = v.begin(); it != v.end();
it++)
        {
            if ((*it)->cat == s)
```

```
                cats.push_back(distance(v.begin(), it));
            al.push_back(distance(v.begin(), it));
        }

        set_intersection(
            cats.begin(), cats.end(),
            al.begin(), al.end(),
            back_inserter(need)
        );

        Mymap m;

        for (vector<Book*>::iterator it = v.begin(); it != v.end();
it++)
        {
            m.insert(Mymap::value_type((*it)->cat,
distance(v.begin(), it)));
        }

        pair<Mymap::iterator,      Mymap::iterator>      pair1      =
m.equal_range(s);


        for (Book* i : v)
        {
            if (i->cat == s)
                b.push_back(i->copy());
        }
    }

    int Library::count_art()
    {
        return count_if(v.begin(), v.end(), [](Book* a) {return a->cat
== "художественная"; });
    }

    int Library::count_sci()
    {
        return count_if(v.begin(), v.end(), [](Book* a) {return a->cat
== "научная"; });
    }

    int Library::count_non()
    {
        return count_if(v.begin(), v.end(), [](Book* a) {return a->cat
== "документальная"; });
    }

    Library::~Library()
    {
        for (int i = 0; i < v.size(); i++)
```

```
            delete v[i];
        v.clear();

        for (int i = 0; i < b.size(); i++)
            delete b[i];
        b.clear();
    }
```

ПРИЛОЖЕНИЕ 10.

ИСХОДНЫЙ ФАЙЛ dllmain.cpp

```
#include "framework.h"

BOOL APIENTRY DllMain( HMODULE hModule,
                       DWORD  ul_reason_for_call,
                       LPVOID lpReserved
                     )
{
    switch (ul_reason_for_call)
    {
    case DLL_PROCESS_ATTACH:
    case DLL_THREAD_ATTACH:
    case DLL_THREAD_DETACH:
    case DLL_PROCESS_DETACH:
        break;
    }
    return TRUE;
}
```