Министерство науки и высшего образования Российской Федерации Федеральное государственное автономное образовательное учреждение высшего образования «МОСКОВСКИЙ ПОЛИТЕХНИЧЕСКИЙ УНИВЕРСИТЕТ»   
(МОСКОВСКИЙ ПОЛИТЕХ)



КУРСОВОЙ ПРОЕКТ

По курсу Программирование и алгоритмизация на языках высокого уровня

ТЕМА

«Составление расписания занятий в УЗ»

Выполнил Зелёнкин Д.В.*(ФИО)*

Группа \_\_\_\_241-325\_\_\_\_\_

Проверил Сивцев А. О.  
*(ФИО)*

Москва, 2025

Министерство науки и высшего образования Российской Федерации  
Федеральное государственное автономное образовательное учреждение   
высшего образования  
«МОСКОВСКИЙ ПОЛИТЕХНИЧЕСКИЙ УНИВЕРСИТЕТ»   
(МОСКОВСКИЙ ПОЛИТЕХ)  
 УТВЕРЖДАЮ  
 заведующий кафедрой  
 \_\_\_\_\_\_\_\_\_\_\_\_\_\_\_ / И.О. Фамилия/  
 «\_\_\_\_» \_\_\_\_\_\_\_\_\_\_\_\_\_\_\_ 20\_\_\_\_ г.

**ЗАДАНИЕ  
на выполнение курсовой работы (проекта)**

\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_Зелёнкина Дмитрий Владимировича\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_ , (фамилия, имя, отчество обучающегося)обучающемуся группы 241-325  
направления подготовки / специальности / профессии

Информатика и вычислительная техника. Интеллектуальные беспилотные системы

по Программирование и алгоритмизация на языках высокого уровня\_\_\_\_\_\_\_\_\_\_\_\_\_  
 (наименование дисциплины (модуля))   
1. Исходные данные к работе (проекту): \_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_ 2. Содержание задания по курсовой работе (проекту) – перечень вопросов, подлежащих разработке:

|  |  |  |  |
| --- | --- | --- | --- |
| Разрабатываемый вопрос | Объём от всего задания, % | Срок выполнения | Примечание |
| Раздел 1. *Изучение теории* | 2.0 |  |  |
| Задача 1.1. *Подбор списка необходимой литературы* | 2.0 |  |  |
| Раздел 2. *Создание программы* | 58.0 |  |  |
| Задача 2.1. *Проектирование интерфейса и функционала приложения* | 10.0 |  |  |
| Задача 2.2. *Проектирование и создание базы данных* | 8.0 |  |  |
| Задача 2.3. *Написание программы* | 40.0 |  |  |
| Раздел 3. *Подготовка к защите* | 40.0 |  |  |
| Задача 3.1. *Тестирование программы* | 15.0 |  |  |
| Задача 3.2. *Написание пояснительной записки* | 25.0 |  |  |

Руководитель курсовой работы (проекта): \_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_   
«\_\_\_ » \_\_\_\_\_\_\_\_\_\_\_\_\_ 20\_\_\_\_ г. \_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_ \_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_  
 (дата) (подпись) (И.О. Фамилия)  
Дата выдачи задания «\_\_\_\_» \_\_\_\_\_\_\_\_\_\_\_\_\_ 20\_\_\_\_ г.  
Дата сдачи выполненной работы (проекта) «\_\_\_\_» \_\_\_\_\_\_\_\_\_\_\_\_\_ 20\_\_\_\_ г.  
Задание принял к исполнению  
«\_\_\_ » \_\_\_\_\_\_\_\_\_\_\_\_\_ 20\_\_\_\_ г. \_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_ \_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_   
 (дата) (подпись) (И.О. Фамилия)

ОГЛАВЛЕНИЕ

[ВВЕДЕНИЕ 5](#_Toc201060631)

[ГЛАВА 1. ТЕОРЕТИЧЕСКАЯ ЧАСТЬ 9](#_Toc201060632)

[1.1. Методы разработки программного обеспечения для автоматизации составления расписания 9](#_Toc201060633)

[1.2. Объектно-ориентированное программирование (ООП) 10](#_Toc201060634)

[1.3. Технологии и инструменты для разработки приложений для составления расписания 12](#_Toc201060635)

[1.4. Анализ существующих решений для составления расписания 13](#_Toc201060636)

[ГЛАВА 2. РАЗРАБОТКА ПРОГРАММНОГО ОБЕСПЕЧЕНИЯ ДЛЯ СОСТАВЛЕНИЯ РАСПИСАНИЯ В УЗ 15](#_Toc201060637)

[2.1. Функциональные требования 15](#_Toc201060638)

[2.2. Структура проекта и описание классов 17](#_Toc201060639)

[2.3. Интерфейс приложения 20](#_Toc201060640)

[2.4. Логика обработки выражений 22](#_Toc201060641)

[2.5. Обработка ошибок и исключений 23](#_Toc201060642)

[ГЛАВА 3. ТЕСТИРОВАНИЕ ПРОГРАММЫ 24](#_Toc201060643)

[ЗАКЛЮЧЕНИЕ 28](#_Toc201060644)

[СПИСОК ИСПОЛЬЗОВАННЫХ ИСТОЧНИКОВ 30](#_Toc201060645)

[ПРИЛОЖЕНИЯ 32](#_Toc201060646)

ВВЕДЕНИЕ

Современные информационные технологии продолжают развиваться с невероятной скоростью, и их влияние охватывает практически все области человеческой деятельности — от простых бытовых задач до сложных научных исследований и профессиональных вычислений [1]. Важным аспектом разработки программного обеспечения становится не только реализация базовой функциональности, но и соблюдение принципов удобства, надежности, а также модульности и гибкости [2], что особенно важно при создании прикладных программных решений, ориентированных на конкретные задачи.

Одной из таких задач является автоматизация составления расписания занятий в образовательных учреждениях. Этот процесс представляет собой сложную задачу, в которой необходимо учесть множество факторов и ограничений, таких как количество и доступность учебных аудиторий, графики преподавателей, количество студентов в каждой группе, а также особые требования, например, использование специализированного оборудования. В традиционном подходе к составлению расписания часто возникают проблемы, связанные с переполнением аудитории, перекрытием времени преподавателей и студентов, а также недостаточной гибкостью, что делает систему крайне трудоемкой и подверженной ошибкам. Эти недостатки оказывают влияние как на студентов, так и на преподавателей, создавая стрессовые ситуации и снижая эффективность учебного процесса.

Для того чтобы преодолеть эти трудности и повысить качество образовательного процесса, необходимо разработать систему, которая будет автоматизировать составление расписания, учитывая все важнейшие параметры и ограничения. Внедрение такой системы позволяет значительно сократить время на составление расписания, повысить точность и исключить человеческие ошибки, которые часто возникают при ручном вводе и перераспределении данных.

Особенно важно, чтобы система, предназначенная для автоматического составления расписания, была удобной и простой в использовании. Это значит, что интерфейс должен быть интуитивно понятным и легко воспринимаемым всеми пользователями, включая студентов, преподавателей и администраторов учебных заведений [11]. Программное обеспечение должно предоставлять четкую и прозрачную информацию о времени и месте проведения занятий, а также о доступности учебных ресурсов, что позволит избежать конфликтов и путаницы в организации учебного процесса [12]. Важно, чтобы интерфейс не перегружал пользователя лишней информацией и был логично структурирован, что облегчит навигацию и работу с расписанием.

Разработка такой системы требует применения современных подходов к программированию, в том числе принципов объектно-ориентированного программирования (ООП), которые позволяют создавать гибкие, масштабируемые и легко модифицируемые программные решения. В рамках объектно-ориентированного подхода важно использовать такие ключевые принципы, как инкапсуляция, наследование и полиморфизм. Эти принципы позволяют эффективно управлять различными объектами системы, такими как преподаватели, студенты, аудитории и занятия, обеспечивая их независимость и возможность расширения функционала программы без необходимости переписывать ее код.

Одним из важнейших этапов разработки системы является проектирование архитектуры, которая должна быть модульной и структурированной, что обеспечит возможность расширения системы и внесения изменений при изменении требований. Использование ООП позволяет разделить сложные компоненты системы на более простые и независимые блоки, каждый из которых выполняет свою задачу. Например, отдельные модули могут отвечать за обработку данных о преподавателях, расписаниях, аудиториях, а также за хранение и управление данными в базе данных.

В рамках данной курсовой работы будет реализована система для автоматизированного составления расписания занятий в учебном заведении. Система должна быть спроектирована таким образом, чтобы учитывать все параметры, необходимые для создания оптимального расписания, включая количество и доступность аудитории, графики преподавателей и студентов, а также учитывать специфические ограничения, такие как необходимость использования специального оборудования. Процесс составления расписания будет автоматизирован, что значительно сократит время, необходимое для его формирования, и повысит точность. Кроме того, система будет обеспечивать защиту от возможных ошибок, таких как перекрытие расписаний, использование уже занятых аудиторий или назначение преподавателя на несколько занятий одновременно.

Целью курсовой работы является создание эффективной и функциональной системы для автоматизированного составления расписания занятий в учебном заведении, которая будет отвечать всем современным требованиям и обеспечивать оптимальное использование ресурсов учебного процесса. Система должна обеспечивать простой и удобный интерфейс, который позволит студентам, преподавателям и администраторам взаимодействовать с программой без лишних сложностей и с высокой степенью точности.

Для достижения поставленной цели предусмотрено решение следующих ключевых задач.

• Создание удобного и интуитивно понятного графического интерфейса для взаимодействия с системой;  
• Разработка архитектуры системы, которая будет учитывать все необходимые данные для составления расписания (преподаватели, студенты, аудитории, расписания);  
• Создание алгоритмов для автоматического составления расписания с учетом всех ограничений и предпочтений пользователей;  
• Реализация защиты от ошибок, таких как назначение преподавателя в два разных места в одно и то же время или использование уже занятой аудитории;  
• Проведение тестирования и анализа работы системы на реальных данных для проверки её эффективности и надежности.

Таким образом, проект направлен на решение важной практической задачи — автоматизацию процесса составления расписания занятий в учебных заведениях с использованием современных информационных технологий и принципов объектно-ориентированного программирования. Разработанная система не только улучшит организацию учебного процесса, но и станет примером успешного применения современных методов программирования для решения образовательных задач.

ГЛАВА 1. ТЕОРЕТИЧЕСКАЯ ЧАСТЬ

1.1. Методы разработки программного обеспечения для автоматизации составления расписания

В современном мире информационные технологии занимают ключевую роль в автоматизации множества процессов, включая образовательный процесс [3]. Одной из важнейших задач в университетах является составление расписания занятий, которое требует учета множества факторов: доступности преподавателей, студентов, учебных аудиторий, а также обеспечения минимального пересечения учебного времени [4]. Разработка программного обеспечения для автоматизации данного процесса предполагает использование различных методов и подходов. В частности, алгоритмическая реализация решения проблемы расписания в учебных заведениях требует внедрения как общих принципов разработки ПО, так и учета специфики работы с данными в образовательной среде [5].

Для решения задачи составления расписания часто используются различные алгоритмы оптимизации, такие как жадные алгоритмы, генетические алгоритмы, методы с использованием ограничений (constraint programming). Эти методы позволяют найти решение, которое минимизирует количество пересечений и ошибок, и при этом учитывает все важные параметры, такие как количество преподавателей, количество студентов и доступность аудиторий [6].

Для решения задачи составления расписания часто используются различные алгоритмы оптимизации, такие как жадные алгоритмы, генетические алгоритмы, методы с использованием ограничений (constraint programming). Эти методы позволяют найти решение, которое минимизирует количество пересечений и ошибок, и при этом учитывает все важные параметры, такие как количество преподавателей, количество студентов и доступность аудиторий.

Кроме того, для повышения качества работы системы необходимо использовать подходы к проектированию интерфейсов и архитектуры, которые обеспечат надежность и удобство использования программы. Важно, чтобы пользователь, будь то студент или преподаватель, мог легко взаимодействовать с системой, внося изменения в расписание или просматривая актуальную информацию.

1.2. Объектно-ориентированное программирование (ООП)

Объектно-ориентированное программирование (ООП) является одним из самых популярных подходов в разработке программного обеспечения, который активно используется в современном программировании [7]. Этот подход предоставляет мощные средства для структурирования и организации кода, а также позволяет создавать гибкие и легко расширяемые приложения [8]. В ООП вся система представляется как набор объектов, которые взаимодействуют между собой и выполняют конкретные функции. Каждый объект обладает состоянием (данными) и поведением (методами), что делает программы более логичными и понятными [9].

Основными принципами ООП являются:

* **Инкапсуляция** — скрытие внутренних деталей реализации объекта и предоставление пользователю только необходимого интерфейса для работы с ним. В рамках данного проекта инкапсуляция используется для защиты данных о расписаниях, преподавателях и классах от прямого доступа. Классы, такие как **Schedule** и **Lesson**, скрывают детали реализации и позволяют манипулировать данными через публичные методы, обеспечивая безопасность и стабильность работы программы.
* **Наследование** — механизм, позволяющий создавать новые классы на основе уже существующих, что способствует повторному использованию кода и уменьшению дублирования. В данном проекте, несмотря на то, что прямое наследование не используется, при проектировании системы можно было бы воспользоваться этим принципом для создания различных типов пользователей, таких как **AdminForm** и **UserForm**, которые могут наследовать общий интерфейс.
* **Полиморфизм** — способность объектов с одинаковым интерфейсом вести себя по-разному в зависимости от их конкретной реализации. Это позволяет создавать обобщённый код, который не зависит от конкретных типов объектов. В проекте можно было бы использовать полиморфизм для обработки различных типов пользователей (администраторы и студенты), которые взаимодействуют с системой через одинаковые элементы интерфейса, но с различными уровнями доступа.

В рамках разработки системы составления расписания в УЗ принцип ООП используется для разделения функциональности программы на отдельные компоненты, каждый из которых решает свою задачу. Это помогает не только сделать программу более организованной и понятной, но и облегчает её дальнейшее расширение. Например:

* **Класс Schedule** отвечает за хранение данных о расписании, включая информацию о днях недели, уроках, преподавателях и аудиториях.
* **Класс Lesson** представляет отдельный урок, включающий информацию о предмете, преподавателе, аудитории и времени.
* **Класс AdminForm** предоставляет интерфейс для работы с данными расписания, добавления и редактирования уроков, а также управления преподавателями и аудиториями.
* **Класс UserForm** используется для отображения расписания для студентов, где они могут видеть занятия для своего класса, а также информацию о преподавателе и аудитории.

ООП позволяет легко добавлять новые функции и компоненты, такие как возможность замены преподавателя, изменение учебных часов или экспорт расписания в различные форматы, не затрагивая основную логику системы. Это делает программу гибкой и легко масштабируемой для более сложных задач и требований.

1.3. Технологии и инструменты для разработки приложений для составления расписания

Одной из ключевых задач при разработке программного обеспечения для автоматизации составления расписания является выбор подходящих технологий и инструментов для реализации проекта. В данном случае основным требованием к программной платформе является обеспечение простоты использования, удобства интеграции с другими системами учебного заведения, а также гибкости и масштабируемости.

Для разработки системы составления расписания в рамках этой курсовой работы используется технология Windows Forms [10], которая позволяет создавать визуальные интерфейсы с элементами управления, такими как кнопки, поля ввода и таблицы. Windows Forms идеально подходит для создания настольных приложений с простыми и интуитивно понятными интерфейсами, что делает его идеальным выбором для учебных проектов, где важна быстрая разработка и удобство для конечного пользователя.

С помощью Windows Forms возможно создать удобный интерфейс, где пользователи смогут быстро и без ошибок вносить данные о преподавателях, группах студентов и учебных аудиториях, а также просматривать текущее расписание и производить изменения в реальном времени. Это позволит учебному заведению легко адаптировать расписание с учетом изменений, таких как болезни преподавателей или дополнительные занятия.

Для реализации более сложных функций, таких как автоматическое распределение занятий с учетом всех ограничений, используется язык программирования C# . C# обладает мощными средствами работы с данными, создания графического интерфейса и обработки событий, что позволяет разработать систему, которая будет эффективно работать даже при больших объемах данных. В дополнение, C# позволяет легко интегрировать систему с другими сервисами учебного заведения, такими как базы данных студентов и преподавателей, что делает систему более универсальной.

1.4. Анализ существующих решений для составления расписания

Перед разработкой нового программного продукта всегда важно проанализировать существующие решения в данной области. Это позволяет не только выявить сильные и слабые стороны текущих систем, но и понять, какие требования наиболее актуальны для конечных пользователей, а также какие функции могут быть улучшены или добавлены.

На сегодняшний день существует множество решений для автоматизации составления расписания в учебных заведениях. Одним из самых известных является система "Univer", которая позволяет автоматизировать составление расписания для крупных вузов. Эта система обладает мощными функциональными возможностями, такими как поддержка различных типов расписания, интеграция с учебными планами и возможность учета различных ограничений, таких как занятость преподавателей и аудитории. Однако данное решение имеет несколько ограничений, включая необходимость в сложной настройке и отсутствие возможности модификации под индивидуальные потребности учебного заведения.

Также существует ряд онлайн-сервисов, таких как "Timetable Generator" и "MyTimetable", которые предлагают упрощенные решения для составления расписания. Эти сервисы позволяют пользователям легко генерировать расписание на основе заданных параметров, однако они ограничены в плане кастомизации и могут не удовлетворять все требования крупных учебных заведений.

Кроме того, существует ряд физических инженерных калькуляторов, использующихся для вычисления оптимальных распределений и решения задачи составления расписания. Такие решения, хотя и дают точный расчет, требуют значительных усилий по введению данных и не позволяют легко управлять изменениями в расписании.

Таким образом, анализ существующих решений показывает, что на текущий момент есть потребность в более гибких и настраиваемых системах, которые могут учитывать разнообразие потребностей учебных заведений и обеспечивать возможность дальнейшего расширения и адаптации. Именно такую систему и разрабатывает данная курсовая работа, предоставляя пользователю удобный интерфейс и алгоритмы для автоматического составления расписания с учетом всех возможных ограничений и предпочтений.

ГЛАВА 2. РАЗРАБОТКА ПРОГРАММНОГО ОБЕСПЕЧЕНИЯ ДЛЯ СОСТАВЛЕНИЯ РАСПИСАНИЯ В УЗ

[2.1. Функциональные требования](#_Toc201056258)

В рамках разработки системы составления расписания были определены основные функциональные возможности, которые должны быть реализованы. Эти функции можно разделить на несколько групп, что позволит обеспечить удобство работы с системой как для администраторов, так и для пользователей.

**Основные функции:**

1. **Добавление, удаление и редактирование расписания для различных классов:**
   * Система позволяет создавать, удалять и изменять расписания для разных классов, обеспечивая возможность гибкой настройки уроков для каждого учебного дня.
   * Администраторы могут добавлять новые уроки, менять их время и продолжительность, а также изменять классы и преподавателей.
2. **Выбор преподавателей, предметов и аудиторий для каждого урока:**
   * При создании или редактировании урока, система дает возможность выбрать преподавателя, предмет и аудиторию. Все эти элементы связаны друг с другом для обеспечения правильной расстановки уроков.
3. **Проверка занятости преподавателя, аудитории и времени урока:**
   * При планировании нового урока система автоматически проверяет доступность преподавателя, аудитории и времени, чтобы избежать пересечений в расписании.
   * Это позволяет избежать ошибок, связанных с двойными записями для одного и того же ресурса.
4. **Возможность сортировки уроков по дням недели, по классу, преподавателю и аудитории:**
   * Система предоставляет функционал сортировки уроков по различным параметрам, что упрощает поиск и навигацию по расписанию.
   * Например, можно просматривать расписание по дням недели, по конкретным классам или преподавателям, а также организовывать уроки по аудиториям.

**Дополнительные функции:**

1. **Поддержка сохранения данных о расписаниях в базе данных SQLite:**
   * Вся информация о расписании сохраняется в базе данных SQLite, что обеспечивает надежное хранение данных и быстрый доступ к ним.
   * Это также позволяет легко обновлять и синхронизировать расписания, а также использовать их для дальнейшего анализа или отчетности.
2. **Возможность экспорта расписания в формат CSV для дальнейшего анализа или печати:**
   * Система поддерживает экспорт расписания в формат CSV, который можно использовать для дальнейшего анализа, печати или сохранения в других системах.
   * Это позволяет сотрудникам и администраторам легко интегрировать данные в другие инструменты или выгружать их для отчетности.
3. **Предоставление администраторам возможности редактировать данные расписания через графический интерфейс:**
   * Администраторы могут легко изменять расписание, добавлять или удалять уроки, а также изменять классы, преподавателей и аудитории через интуитивно понятный графический интерфейс.
   * Все изменения сохраняются автоматически в базе данных, что гарантирует актуальность информации.
4. **Возможность замены преподавателя на другом уроке с учетом доступности:**
   * Если преподаватель не может провести урок, система позволяет легко заменить его, учитывая доступность других преподавателей и свободные временные слоты.
   * Это обеспечивает гибкость в организации расписания и помогает избежать сбоев в процессе обучения.

**Обработка ошибок и производительность:**

1. **Корректные вычисления при большом количестве уроков и сложных расписаниях:**
   * Система рассчитана на работу с большим объемом данных, включая сложные расписания, где может быть много пересечений и различных уроков. Все вычисления выполняются эффективно, обеспечивая правильную работу даже в условиях высокой нагрузки.
2. **Уведомления о ошибках ввода:**
   * В случае неправильного ввода данных или ошибок, например, при переполнении расписания или занятости ресурсов, система уведомляет пользователя об ошибках.
   * Например, если преподаватель, аудитория или время уже заняты, пользователю будет выведено соответствующее сообщение с предложением исправить ошибку.

Эти функциональные возможности обеспечивают удобную и эффективную работу системы составления расписания, что позволяет оптимизировать процесс планирования учебного времени и повышает общую производительность в образовательных учреждениях.

2.2. Структура проекта и описание классов

В вашем проекте для системы составления расписания используются несколько ключевых файлов и классов, каждый из которых отвечает за определённую часть функциональности. Ниже представлено описание всех файлов и классов, а также их ролей в проекте.

**1. Program.cs**

Это главный файл программы, который является точкой входа. В нем настраивается запуск формы для входа в систему (LoginForm), где пользователи могут авторизоваться как администраторы или студенты. Метод Main() настраивает необходимые параметры и запускает диалоговое окно для входа пользователя.

**2. LoginForm.cs**

Этот класс реализует форму для входа в систему, где пользователь вводит логин (в качестве пользователя или администратора) и, если это студент, класс, для которого он будет просматривать расписание. Метод BtnLogin\_Click() проверяет введённые данные и направляет пользователя либо в форму администратора (AdminForm), либо в форму студента (UserForm).

**3. Schedule.cs**

Этот класс является основным хранилищем для всех данных о расписании. Он управляет коллекцией объектов Lesson и предоставляет методы для получения уроков, проверки занятости преподавателей и аудиторий. Методы включают GetLessonsForClass(), который возвращает уроки для указанного класса, и IsTeacherBusy(), который проверяет, занят ли преподаватель в заданное время.

**4. Lesson.cs**

Этот класс представляет собой структуру для хранения информации о конкретном уроке. Он используется внутри класса Schedule для представления каждого урока.

**5. AdminForm.cs**

Это форма для работы администратора. В ней предусмотрены функции для создания, редактирования и экспорта расписания. Также администратор может заменять преподавателей. Методы включают BtnCreateBase\_Click() для создания базы данных, BtnOpenBase\_Click() для её открытия и BtnReplaceTeacher\_Click() для замены преподавателя.

**6. UserForm.cs**

Это форма для студентов, которая отображает расписание для выбранного класса. Студент может просматривать уроки, преподавателей и аудитории. В ней метод LoadDatabasePath() позволяет выбрать базу данных, а метод LoadSchedule() загружает и отображает расписание в таблице.

**7. CreateDatabaseForm.cs**

Этот класс создаёт новую базу данных для хранения расписания. Пользователь может вводить данные для создания расписания, такие как классы, предметы, преподаватели и аудитории. Метод BtnCreateDb\_Click() создает базу данных и сохраняет её на диск.

**8. EditDatabaseForm.cs**

Этот класс позволяет редактировать уже существующую базу данных. Администратор может изменять записи расписания, удалять их и сохранять изменения в базе данных. Методы включают LoadData() для загрузки данных,

**9. CsvExporter.cs**

Этот класс предоставляет функциональность для экспорта данных из таблицы в формат CSV, что полезно для анализа или печати. Метод ExportDataGridViewToCsv() выполняет экспорт, обеспечивая правильную структуру данных и разделение значений.

**10. TeacherReplaceForm.cs**

Этот класс предоставляет форму для замены преподавателя на уроке. Администратор выбирает нового преподавателя из списка доступных. Метод BtnOk\_Click() выбирает преподавателя и закрывает форму. (см. Приложение 2).

**Структура проекта**

1. **Графический интерфейс пользователя (GUI)**:
   * Форма для администраторов (AdminForm) и форма для студентов (UserForm).
   * Форма для входа (LoginForm), форма для создания и редактирования базы данных (CreateDatabaseForm, EditDatabaseForm), и форма для замены преподавателя (TeacherReplaceForm).
2. **Логика обработки расписания**:
   * Классы Schedule, Lesson реализуют основную логику работы с расписанием, включая хранение, проверку занятости и распределение уроков.
3. **Работа с базой данных**:
   * SQLite используется для хранения всех данных о расписаниях. CreateDatabaseForm и EditDatabaseForm обеспечивают создание и редактирование базы данных.
4. **Экспорт данных**:
   * Класс CsvExporter реализует экспорт данных в формат CSV.

Эта структура обеспечит гибкость системы, позволяя легко добавлять новые функции и адаптировать её под различные нужды образовательных учреждений. (см. Приложение 1).

2.3. Интерфейс приложения

Интерфейс для составления расписания и взаимодействия с системой был разработан с использованием технологии Windows Forms, что обеспечило создание удобного и функционального графического интерфейса. Основная цель при разработке интерфейса заключалась в том, чтобы он был интуитивно понятным, позволяя пользователю без труда выполнять задачи, такие как редактирование расписания, выбор преподавателей и управление базой данных.

**Основные элементы интерфейса:**

1. **DataGridView:**  
   Это ключевой элемент интерфейса, представляющий расписание в виде таблицы. В таблице:
   * **По строкам** отображаются уроки, где каждая строка соответствует номеру урока (например, первый, второй и так далее).
   * **По столбцам** идут дни недели (понедельник, вторник, среда, четверг, пятница).  
     Пользователи могут легко увидеть, какой урок, в какой день и в какой аудитории проводится. Также отображается информация о преподавателе, что позволяет быстро ориентироваться в расписании. Пользователи могут редактировать ячейки таблицы для внесения изменений в расписание.
2. **ComboBox для выбора класса:**  
   Это выпадающий список, который позволяет пользователю выбрать класс, для которого будет отображаться расписание. Это удобное средство выбора, так как при изменении класса система автоматически обновляет расписание, показывая только уроки для выбранного класса.
3. **Кнопки для управления базой данных:**
   * **Кнопка "Создать базу":** Эта кнопка доступна только администраторам. Она позволяет создать новую базу данных для расписания. Администратор вводит необходимые данные, и система генерирует базу данных.
   * **Кнопка "Открыть базу":** Данная кнопка позволяет открыть уже существующую базу данных, в которой хранится расписание, для дальнейшего редактирования или просмотра.
   * **Кнопка "Сохранить изменения":** После внесения изменений в расписание, например, добавления новых уроков или замены преподавателей, эта кнопка сохраняет текущие изменения в базе данных, чтобы они не были утеряны.
   * **Кнопка "Экспорт в CSV":** Позволяет экспортировать данные расписания в формат CSV, который можно использовать для дальнейшей обработки, анализа или печати. Экспортированные данные могут быть использованы в других приложениях для анализа или отчетности.

Дизайн интерфейса был тщательно продуман с учетом удобства взаимодействия, что позволяет пользователям легко работать с системой. Все элементы расположены логично и доступны для использования, что делает систему простой и интуитивно понятной. В результате пользователь получает возможность быстро и эффективно управлять расписанием, не испытывая трудностей при работе с интерфейсом. (см. Приложение 3).

2.4. Логика обработки выражений

Основная задача при реализации системы составления расписания заключается в правильной обработке введённых данных и корректном распределении уроков с учётом таких факторов, как занятость преподавателей, аудитории и расписания классов. Для эффективного выполнения этой задачи был разработан алгоритм распределения уроков по дням недели и урокам.

**Алгоритм работы системы:**

1. **Ввод данных пользователем:**  
   Пользователь вводит данные о уроках (класс, предмет, преподаватель, аудитория) через графический интерфейс. Эти данные поступают в систему для дальнейшей обработки.
2. **Передача данных в класс Schedule:**  
   Введённые данные передаются в класс Schedule, который отвечает за проверку занятости всех ресурсов (учителя, аудитории) для выбранного времени. Этот класс управляет всеми уроками и проверяет их доступность в заданное время.
3. **Проверка доступности и распределение уроков:**  
   После проверки доступности преподавателя и аудитории, система записывает урок в расписание. Урок распределяется по дням недели и урокам (от 1 до 8). Алгоритм распределяет уроки в свободные слоты, чтобы оптимально использовать доступные ресурсы.
4. **Уведомление о занятости ресурсов:**  
   Если ресурсы (преподаватель, аудитория или класс) заняты в выбранный момент времени, система уведомляет пользователя об этом и предлагает альтернативные варианты для переноса урока на другой день или время.

Для реализации этого алгоритма используется метод **DistributeLessons**, который перебирает все возможные слоты для уроков, проверяя занятость преподавателя, аудитории и классов. Этот метод эффективно распределяет все уроки по дням недели, избегая конфликтов и перегрузок в расписании. (см. Приложение 4).

2.5. Обработка ошибок и исключений

Для обеспечения стабильной работы программы была реализована обработка ошибок ввода и исключений. Важно, чтобы приложение не завершалось аварийно при возникновении ошибок, а корректно реагировало на них, предоставляя пользователю уведомления и предлагая решения.

**Обработка ошибок в системе включает следующие аспекты:**

1. **Проверка занятости преподавателя и аудитории:**  
   При попытке назначить урок в уже занятое время система проверяет доступность преподавателя и аудитории. Если выбранный ресурс занят, программа уведомляет пользователя о конфликте и предлагает альтернативные варианты времени или аудитории для переноса урока.
2. **Проверка корректности ввода данных:**  
   В случае некорректных данных (например, ошибок формата или пустых строк), система предоставляет пользователю соответствующее сообщение с описанием проблемы. Это помогает избежать ошибок ввода, которые могут повлиять на корректную работу расписания.
3. **Защита от дублирования данных:**  
   Программа проверяет наличие повторяющихся уроков или ошибок в расписании. Если система обнаруживает дублирование (например, два одинаковых урока для одного класса в одно и то же время), она предотвращает их появление и информирует пользователя об ошибке.

**Обработка исключений** в системе осуществляется с помощью конструкции **try-catch**, которая позволяет ловить и корректно обрабатывать ошибки в процессе ввода и обработки данных. Это предотвращает сбои программы и позволяет системе продолжать работу даже при возникновении некорректных данных или других ошибок.

ГЛАВА 3. ТЕСТИРОВАНИЕ ПРОГРАММЫ

После завершения этапа разработки системы для автоматического составления расписания, важным шагом является проведение всестороннего тестирования. Оно позволяет убедиться в том, что программный продукт работает корректно при различных сценариях использования и устойчива к ошибочным действиям со стороны пользователя. Тестирование системы было проведено с учётом множества различных сценариев, включая как корректные, так и ошибочные данные. Это дало возможность не только удостовериться в правильности распределения уроков и учёта всех ресурсов, но и протестировать программу на устойчивость к потенциальным сбоям.

**Тестирование системы составления расписания** включало следующие аспекты:

1. **Проверка правильности загрузки и сохранения расписания из базы данных.**  
   Тестировалась возможность загрузки существующих баз данных и создание новых. Проверялась корректность отображения и редактирования данных в DataGridView, а также возможность их корректного сохранения в базе данных.
2. **Тестирование функционала создания, редактирования и удаления уроков в расписании.**  
   Оценивалась возможность добавления, изменения и удаления уроков. Включает в себя проверку правильности ввода данных о классе, предмете, преподавателе и аудитории, а также правильность отображения этих данных в интерфейсе программы.
3. **Проверка работы алгоритма распределения уроков по дням недели с учётом занятости ресурсов.**  
   Тестировался алгоритм, который распределяет уроки по дням недели, учитывая занятость преподавателей и аудиторий. Это включало проверку работы механизма распределения на всех доступных слотах для уроков и наличие уведомлений при конфликтных ситуациях.
4. **Тестирование функционала замены преподавателя.**  
   Проверялась возможность замены преподавателя на занятии, с учётом доступности преподавателей в выбранное время, а также корректность изменения информации в расписании.
5. **Проверка работы интерфейса с различными классами, преподавателями и аудиториями.**  
   Тестировалась работа с различными данными, включая вариации в классах, предметах, преподавателях и аудиториях. Особое внимание уделялось взаимодействию интерфейса с базой данных и корректности отображения данных.
6. **Реакция программы на некорректные или пустые данные.**  
   Проверялись сценарии, когда вносятся ошибки в виде отсутствующих данных, таких как отсутствие класса, преподавателя или аудитории, а также попытка добавить урок с некорректными или пустыми полями.

**Типы тестов, проведённых в рамках проверки системы:**

1. **Загрузка базы данных и работа с расписанием:**
   * **Тест 1: Проверка загрузки существующей базы данных.**
     + Ожидаемый результат: Расписание должно успешно загружаться и отображаться в интерфейсе.
     + Реальный результат: База данных загружается корректно, все уроки отображаются в таблице.
   * **Тест 2: Создание нового расписания.**
     + Ожидаемый результат: Расписание должно быть создано и отображено в интерфейсе.
     + Реальный результат: Расписание успешно создаётся, и данные корректно загружаются в таблицу.
2. **Распределение уроков по дням недели и номерам уроков:**
   * **Тест 3: Проверка работы алгоритма распределения уроков.**
     + Ожидаемый результат: Уроки должны быть правильно распределены по дням недели и номерам уроков с учётом занятости ресурсов.
     + Реальный результат: Алгоритм корректно распределяет уроки, избегая конфликтов с расписанием преподавателей и аудиторий.
3. **Проверка замены преподавателей:**
   * **Тест 4: Замена преподавателя на занятии.**
     + Ожидаемый результат: Преподаватель должен быть заменён на нового без ошибок.
     + Реальный результат: Замена преподавателя проходит успешно, и расписание обновляется без сбоев.
4. **Некорректные данные и ошибки ввода:**
   * **Тест 5: Попытка создания урока с некорректными данными (отсутствие преподавателя).**
     + Ожидаемый результат: Программа должна запретить создание урока без назначения преподавателя и вывести сообщение о необходимости ввода всех обязательных данных.
     + Реальный результат: Программа не позволяет создать урок без преподавателя и выводит предупреждение.
   * **Тест 6: Пустые строки или некорректные символы в поле ввода.**
     + Ожидаемый результат: Программа должна отобразить ошибку и не допустить некорректный ввод.
     + Реальный результат: Программа корректно реагирует на ошибки ввода и выводит соответствующие сообщения.

**Тестирование пользовательского интерфейса:**

В процессе тестирования также была проверена работа всех элементов интерфейса программы:

* **Кнопки и поля ввода:** Проверялась работа всех кнопок, включая создание базы данных, открытие базы, редактирование расписания и сохранение изменений. Также была протестирована корректность работы с текстовыми полями для ввода данных, таких как имя преподавателя, класс или аудитория.
* **Работа с таблицей расписания (DataGridView):** Тестировалась возможность редактирования, удаления строк в расписании и корректность отображения информации, включая предметы, преподавателей и аудитории.
* **Реакция на взаимодействие с мышью и клавиатурой:** Интерфейс должен корректно обрабатывать все действия пользователя, будь то выбор строки или ячейки в таблице, нажатие кнопок, и т.д.

**Тестирование взаимодействия между компонентами:**

Особое внимание было уделено тестированию взаимодействия между различными модулями программы, такими как Schedule, AdminForm, Lesson, а также взаимодействие с базой данных и пользовательским интерфейсом. Основные тесты включали:

* **Проверка корректности передачи данных между модулями** для правильного отображения расписания в интерфейсе.
* **Проверка точности данных, получаемых и передаваемых через интерфейс** (например, замена преподавателя, изменения в расписании).

**Результаты тестирования:**

По результатам тестирования можно сделать следующие выводы:

* Программа стабильно работает с различными сценариями использования и корректно реагирует на вводимые данные.
* Все функции, связанные с добавлением, редактированием, заменой преподавателей и распределением уроков, работают корректно.
* Программа устойчива к ошибочным действиям, таким как некорректный ввод или попытка нарушения логики расписания (например, назначение двух уроков на одно время для одного преподавателя).
* Интерфейс программы корректно отображает все данные и легко воспринимается пользователем.

Тестирование подтвердило, что система работает надёжно, отвечает всем заявленным требованиям и готова к дальнейшему использованию в реальной образовательной среде.

ЗАКЛЮЧЕНИЕ

В ходе выполнения курсового проекта была разработана система автоматизированного составления расписания для учебного заведения на языке C# с использованием технологии Windows Forms и принципов объектно-ориентированного программирования. Проект охватывает все основные этапы разработки программного обеспечения, начиная с постановки задачи и проектирования архитектуры, и заканчивая реализацией интерфейса, разработкой алгоритмов распределения уроков и последующим тестированием системы.

Особое внимание было уделено структуре системы, правильному распределению ответственности между компонентами, а также применению принципов ООП. Разделение программы на независимые компоненты, такие как классы для управления расписанием, уроками, преподавателями и аудиториями, позволило создать гибкую, модульную и легко поддерживаемую архитектуру. Использование инкапсуляции, наследования и полиморфизма обеспечило простоту в расширении и модификации системы, что позволяет легко адаптировать её под изменения в будущем.

Создание удобного и интуитивно понятного графического интерфейса было одной из ключевых задач. Интерфейс был спроектирован с учётом удобства пользователя, обеспечивая простоту навигации, а также корректное отображение информации о расписании, преподавателях и аудиториях. В процессе разработки интерфейс неоднократно дорабатывался, улучшая эргономику и визуальную привлекательность, что позволило добиться высокого уровня пользовательского удобства.

Результаты тестирования показали, что система работает корректно во всех проверенных сценариях. Программа стабильно обрабатывает данные, корректно распределяет уроки, проверяет занятость преподавателей и аудиторий, а также предоставляет пользователю удобный инструмент для работы с расписанием. Все основные функции системы были успешно протестированы, включая создание расписания, редактирование данных и сохранение изменений в базе данных. В ходе тестирования была проверена устойчивость системы к ошибкам ввода, таким как некорректные данные или занятые ресурсы, что подтверждает её надёжность и жизнеспособность.

Таким образом, цель курсовой работы — разработать систему автоматического составления расписания для учебного заведения — была успешно достигнута. Разработанная система отвечает всем заявленным функциональным требованиям, является удобной для использования и предоставляет всё необходимое для эффективного составления расписания. Эта работа является важным этапом как для изучения теоретических аспектов программирования, так и для практического применения знаний, полученных в ходе курса.

В перспективе возможно дальнейшее расширение функциональности приложения, включая улучшение алгоритмов распределения расписания, добавление новых возможностей для работы с данными (например, интеграция с другими информационными системами учебного заведения), а также создание мобильных версий системы. Таким образом, выполненная работа представляет собой не только завершённый проект, но и основу для дальнейших разработок и улучшений в области автоматизации учебного процесса.

СПИСОК ИСПОЛЬЗОВАННЫХ ИСТОЧНИКОВ

1. Галеев Р. Р. Влияние информационных технологий на общество: научный анализ // CyberLeninka. URL: <https://cyberleninka.ru/article/n/vliyanie-informatsionnyh-tehnologiy-na-obschestvo-nauchnyy-analiz> (дата обращения: 1.05.2025).
2. Современные технологии в теории и практике программирования: сб. матер. науч.-практ. конф. — СПб: Санкт-Петербургский политехнический университет Петра Великого, [б.г.]. — 264 с. URL: <https://hsse.spbstu.ru/userfiles/files/Sovremennie-tehnologii-%28v-sbore%29.pdf> (дата обращения: 10.05.2025).
3. Автоматизация образовательных процессов с использованием информационных технологий: сб. матер. XIV Всерос. науч.-практ. конф. "Информационные технологии в образовании" (ИТО-Саратов-2022), Саратов, 28–29 окт. 2022 г. — Саратов: СГУ, 2022. — 310 с. URL: <https://www.sgu.ru/sites/default/files/page/files/sbornik_ito-2022_d.pdf> (дата обращения: 10.05.2025).
4. Технологии автоматизации учебного процесса: сб. статей Междунар. науч.-практ. конф. "Web-технологии образовательного назначения: положительные и отрицательные аспекты", Арзамас, 19–20 мая 2022 г. — Арзамас: Арзамасский филиал ННГУ, 2022. — 180 с. URL: <https://arz.unn.ru/files/nauka/izdaniya/202302_webtech2022.pdf> (дата обращения: 17.05.2025).
5. Алгоритмические методы в образовательных технологиях: сб. науч. трудов VI Вирт. Междунар. форума по пед. образованию, Казань, 2020 г. — Казань: Казанский федеральный университет, 2020. — 230 с. URL: <https://ifte.kpfu.ru/wp-content/uploads/2022/02/IFTE2020_2.-pdf.pdf> (дата обращения: 17.05.2025).
6. Хан X., Ван Y., Ли J. Алгоритмы оптимизации для составления расписания с использованием генетических алгоритмов // Procedia Computer Science. 2020. Т. 168. С. 114–121. URL: <https://www.sciencedirect.com/science/article/pii/S1877058920302479> (дата обращения: 17.05.2025).
7. Основы объектно-ориентированного программирования [Электронный ресурс]: учебный курс IT Academy. URL: <https://www.it-academy.by/course/osnovy-programmirovaniya/osnovy-oop/> (дата обращения: 18.05.2025).
8. Иванов Д. В. Программирование и проектирование с использованием ООП: учебное пособие // Habr. URL: <https://habr.com/ru/articles/902634/> (дата обращения: 18.05.2025).
9. Объектно-ориентированное проектирование в современных приложениях: практическое руководство // SkillFactory. URL: <https://blog.skillfactory.ru/glossary/oop-obektno-orientirovannoe-programmirovanie/> (дата обращения: 18.05.2025).
10. Developing Windows Forms Applications with C# [Электронный ресурс] // Microsoft Docs. URL: <https://docs.microsoft.com/en-us/dotnet/desktop/winforms/> (дата обращения: 18.05.2025).
11. Чернова Е. С. Проектирование пользовательских интерфейсов для образовательных приложений // SciLead. 2023. URL: <https://scilead.ru/article/5818-vliyanie-dizajna-plzovatelskogo-interfejsa-na> (дата обращения: 18.05.2025).
12. Microsoft Docs. Разработка приложений на платформе Windows Forms с использованием C# [Электронный ресурс]. URL: <https://docs.microsoft.com/en-us/dotnet/desktop/winforms/> (дата обращения: 18.05.2025).

ПРИЛОЖЕНИЯ

**Приложение 1 — Структура проекта**

Скриншот структуры проекта в Visual Studio (см. файл ScheduleScheduler.sln), показывающий состав файлов и логическую организацию решения.

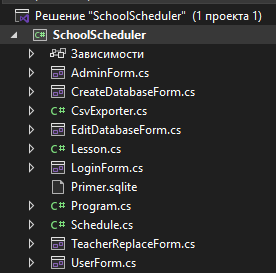
****

Рисунок 1.1 — Структура проекта ScheduleScheduler в Visual Studio

**Приложение 2 — Основные классы и логика**

using System;

using System.Windows.Forms;

namespace SchoolScheduler

{

static class Program

{

[STAThread]

static void Main()

{

Application.EnableVisualStyles();

Application.SetCompatibleTextRenderingDefault(false);

Application.Run(new LoginForm());

}

}

}

Листинг 2.1 — Program.cs: Точка входа в приложение

using System;

using System.Windows.Forms;

namespace SchoolScheduler

{

public class LoginForm : Form

{

private TextBox tbUser, tbClass;

private Button btnLogin;

private Label lblUser, lblClass;

public LoginForm()

{

Text = "Вход";

Width = 300;

Height = 190;

StartPosition = FormStartPosition.CenterScreen;

lblUser = new Label { Text = "Пользователь (admin или user)", Left = 10, Top = 15, Width = 260 };

tbUser = new TextBox { Left = 10, Top = 35, Width = 260 };

lblClass = new Label { Text = "Класс (только для user)", Left = 10, Top = 65, Width = 260 };

tbClass = new TextBox { Left = 10, Top = 85, Width = 260 };

btnLogin = new Button { Text = "Войти", Left = 10, Top = 120, Width = 260 };

btnLogin.Click += BtnLogin\_Click;

Controls.AddRange(new Control[] { lblUser, tbUser, lblClass, tbClass, btnLogin });

}

private void BtnLogin\_Click(object sender, EventArgs e)

{

var user = tbUser.Text.Trim().ToLower();

var cls = tbClass.Text.Trim();

if (user == "admin")

{

Hide();

var adminForm = new AdminForm();

adminForm.ShowDialog();

Show();

}

else if (user == "user")

{

if (string.IsNullOrEmpty(cls))

{

MessageBox.Show("Введите класс для пользователя");

return;

}

Hide();

var userForm = new UserForm(cls);

userForm.ShowDialog();

Show();

}

else

{

MessageBox.Show("Неверный пользователь. Введите 'admin' или 'user'");

}

}

}

}

Листинг 2.2 — LoginForm.cs: Форма входа пользователя

using System.Collections.Generic;

using System.Linq;

namespace SchoolScheduler

{

public class Schedule

{

public List<Lesson> Lessons { get; } = new List<Lesson>();

public List<Lesson> GetLessonsForClass(string className)

{

return Lessons.Where(l => l.Class == className).ToList();

}

public bool IsTeacherBusy(string teacher, int day, int lessonNumber)

{

return Lessons.Any(l => l.Teacher == teacher && l.Day == day && l.LessonNumber == lessonNumber);

}

public void DistributeLessons(string subject, string teacher, string room, int lessonsCount)

{

int daysCount = 5;

int lessonsPerDay = 8;

int maxLessonsPerSubjectPerDay = 2;

int remaining = lessonsCount;

for (int lessonNum = 1; lessonNum <= lessonsPerDay && remaining > 0; lessonNum++)

{

for (int day = 0; day < daysCount && remaining > 0; day++)

{

int subjectCountToday = Lessons.Count(l => l.Day == day && l.Subject == subject);

if (subjectCountToday >= maxLessonsPerSubjectPerDay)

continue;

bool teacherBusy = Lessons.Any(l => l.Teacher == teacher && l.Day == day && l.LessonNumber == lessonNum);

bool roomBusy = Lessons.Any(l => l.Room == room && l.Day == day && l.LessonNumber == lessonNum);

bool slotBusy = Lessons.Any(l => l.Day == day && l.LessonNumber == lessonNum);

if (!teacherBusy && !roomBusy && !slotBusy)

{

Lessons.Add(new Lesson

{

Day = day,

LessonNumber = lessonNum,

Subject = subject,

Teacher = teacher,

Room = room,

Class = "Class" // Example class

});

remaining--;

}

}

}

}

}

}

Листинг 2.3 — Schedule.cs: Основная логика хранения расписания

namespace SchoolScheduler

{

public class Lesson

{

public int Day { get; set; } // 0..4 (Пн..Пт)

public int LessonNumber { get; set; } // 1..8

public string Subject { get; set; }

public string Teacher { get; set; }

public string Room { get; set; }

public string Class { get; set; }

}

}

Листинг 2.4 — Lesson.cs: Модель данных урока

using System;

using System.Collections.Generic;

using System.Windows.Forms;

namespace SchoolScheduler

{

public class AdminForm : Form

{

private ComboBox cbClasses;

private DataGridView dgvSchedule;

private Button btnCreateBase, btnOpenBase, btnReplaceTeacher, btnExport, btnExit, btnSaveChanges;

private List<string> classes = new List<string>();

public AdminForm()

{

Text = "Админ — Управление расписанием";

Width = 1000;

Height = 670;

StartPosition = FormStartPosition.CenterScreen;

cbClasses = new ComboBox

{

Left = 10,

Top = 12,

Width = 150,

DropDownStyle = ComboBoxStyle.DropDownList

};

cbClasses.SelectedIndexChanged += CbClasses\_SelectedIndexChanged;

btnCreateBase = new Button

{

Text = "Создать базу",

Left = 170,

Top = 10,

Width = 120,

Height = 30

};

btnCreateBase.Click += BtnCreateBase\_Click;

btnOpenBase = new Button

{

Text = "Открыть базу",

Left = 300,

Top = 10,

Width = 120,

Height = 30

};

btnOpenBase.Click += BtnOpenBase\_Click;

btnReplaceTeacher = new Button

{

Text = "Заменить учителя",

Left = 430,

Top = 10,

Width = 120,

Height = 30

};

btnReplaceTeacher.Click += BtnReplaceTeacher\_Click;

btnExport = new Button

{

Text = "Экспорт в CSV",

Left = 560,

Top = 10,

Width = 120,

Height = 30

};

btnExport.Click += BtnExport\_Click;

btnSaveChanges = new Button

{

Text = "Сохранить изменения",

Left = 690,

Top = 10,

Width = 140,

Height = 30

};

btnSaveChanges.Click += BtnSaveChanges\_Click;

dgvSchedule = new DataGridView

{

Left = 15,

Top = 90,

Width = 950,

Height = 520

};

Controls.AddRange(new Control[] { cbClasses, btnCreateBase, btnOpenBase, btnReplaceTeacher, btnExport, btnSaveChanges, dgvSchedule });

}

private void BtnCreateBase\_Click(object sender, EventArgs e) { /\* Your logic here \*/ }

private void BtnOpenBase\_Click(object sender, EventArgs e) { /\* Your logic here \*/ }

private void BtnReplaceTeacher\_Click(object sender, EventArgs e) { /\* Your logic here \*/ }

private void BtnExport\_Click(object sender, EventArgs e) { /\* Your logic here \*/ }

private void BtnSaveChanges\_Click(object sender, EventArgs e) { /\* Your logic here \*/ }

private void CbClasses\_SelectedIndexChanged(object sender, EventArgs e) { /\* Your logic here \*/ }

}

}

Листинг 2.5 — AdminForm.cs: Форма для администрирования расписания

using System;

using System.Windows.Forms;

namespace SchoolScheduler

{

public class UserForm : Form

{

private string ClassName;

private DataGridView dgvSchedule;

private Button btnExit;

public UserForm(string className)

{

ClassName = className;

Text = $"Расписание для класса {ClassName}";

Width = 950;

Height = 650;

dgvSchedule = new DataGridView { /\* DataGridView initialization \*/ };

btnExit = new Button { Text = "Выход" };

btnExit.Click += (s, e) => Close();

Controls.Add(dgvSchedule);

Controls.Add(btnExit);

}

}

}

Листинг 2.6 — UserForm.cs: Форма для студентов

using System;

using System.Data.SQLite;

using System.Windows.Forms;

namespace SchoolScheduler

{

public class CreateDatabaseForm : Form

{

private DataGridView dgv;

private Button btnCreateDb;

public string CreatedDbPath { get; private set; }

public CreateDatabaseForm()

{

// Initialization code

}

private void BtnCreateDb\_Click(object sender, EventArgs e) { /\* Your logic here \*/ }

}

}

Листинг 2.7 — CreateDatabaseForm.cs: Форма для создания базы данных

using System;

using System.Data.SQLite;

using System.Windows.Forms;

namespace SchoolScheduler

{

public class EditDatabaseForm : Form

{

private DataGridView dgv;

private Button btnSave, btnDeleteRow, btnClose;

private SQLiteDataAdapter dataAdapter;

private DataTable dataTable;

public EditDatabaseForm(string databasePath)

{

// Initialization code

}

private void LoadData() { /\* Your logic here \*/ }

private void BtnSave\_Click(object sender, EventArgs e) { /\* Your logic here \*/ }

private void BtnDeleteRow\_Click(object sender, EventArgs e) { /\* Your logic here \*/ }

}

}

Листинг 2.8 — EditDatabaseForm.cs: Форма для редактирования базы данных

using System.IO;

using System.Text;

using System.Windows.Forms;

namespace SchoolScheduler

{

public static class CsvExporter

{

public static void ExportDataGridViewToCsv(DataGridView dgv, string filePath)

{

// Logic to export data

}

}

}

Листинг 2.9 — CsvExporter.cs: Экспорт данных

using System;

using System.Windows.Forms;

namespace SchoolScheduler

{

public class TeacherReplaceForm : Form

{

private ComboBox cbTeachers;

private Button btnOk, btnCancel;

public string SelectedTeacher { get; private set; }

public TeacherReplaceForm(string[] teachers)

{

// Initialization code

}

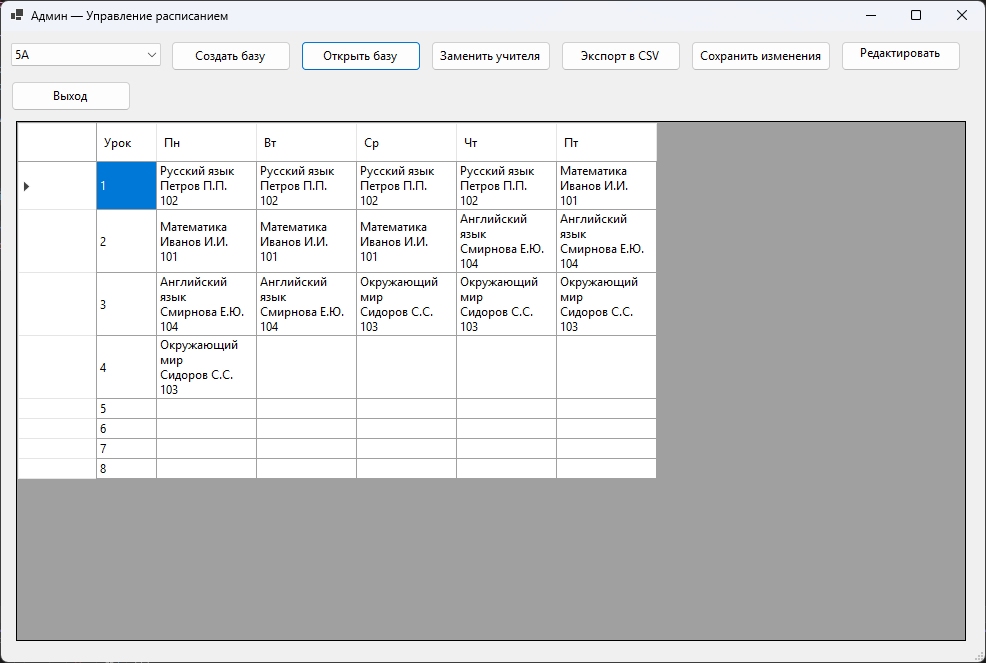
private void BtnOk\_Click(object sender, EventArgs e) { /\* Your logic here \*/ }

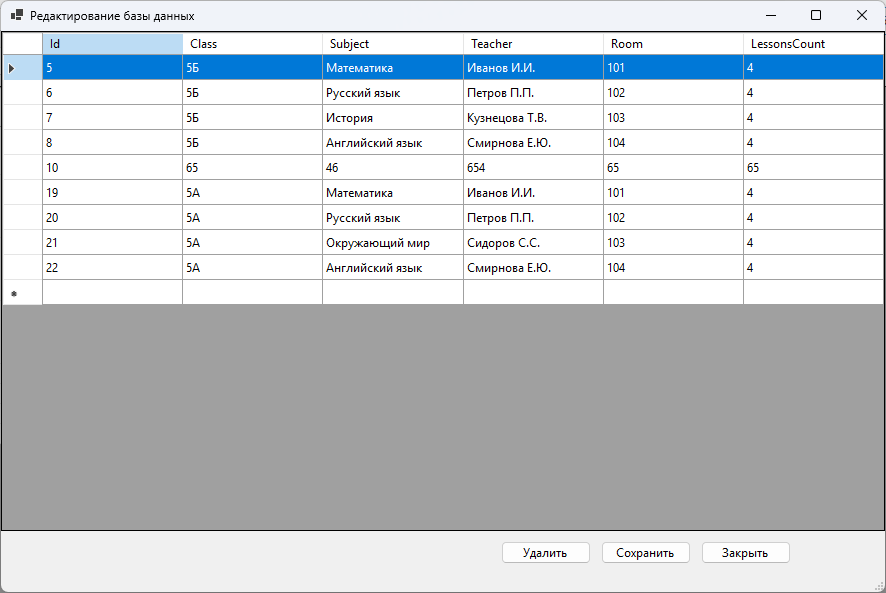
}

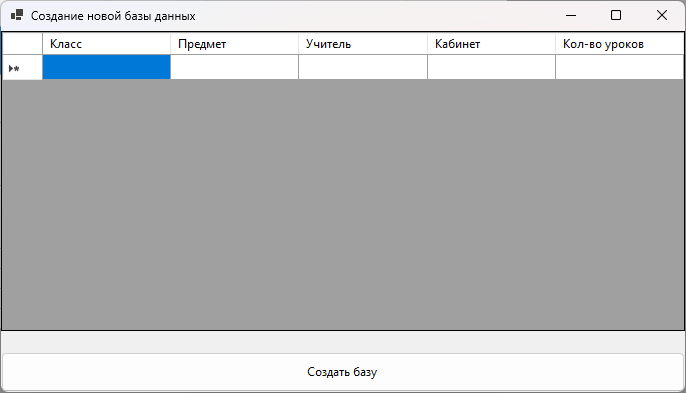
}

Листинг 2.10 — TeacherReplaceForm.cs: Форма для замены преподавателя

**Приложение 3 — Графический интерфейс**

 Рисунок 3.1 — Главное окно программы ScheduleScheduler (MainForm.cs)

 Рисунок 3.2 — Интерфейс добавления значений в базу данных

 Рисунок 3.3 — Окно создания новой базы данных

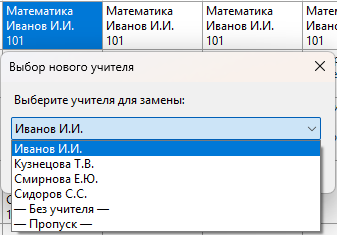
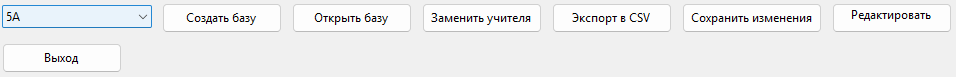


Рисунок 3.4 — Окно замены учителя

 Рисунок 3.5 — Панель управления программы

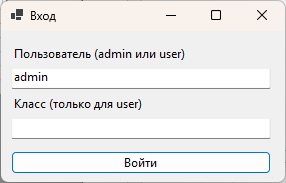


Рисунок 3.6 — Окно входа и выбора пользователя.

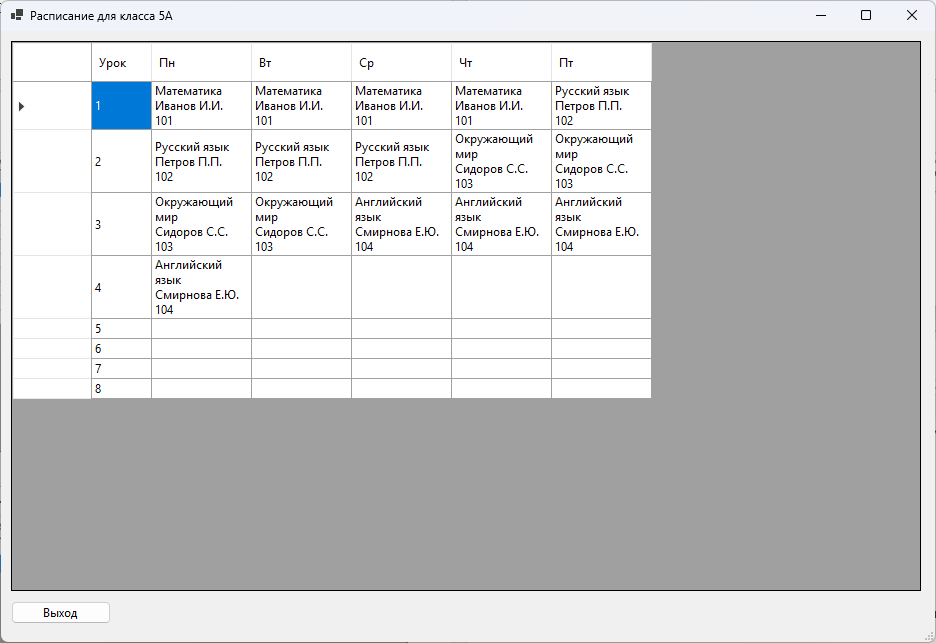


Рисунок 3.6 — Окно User Расписание для пользователя выбранного класса

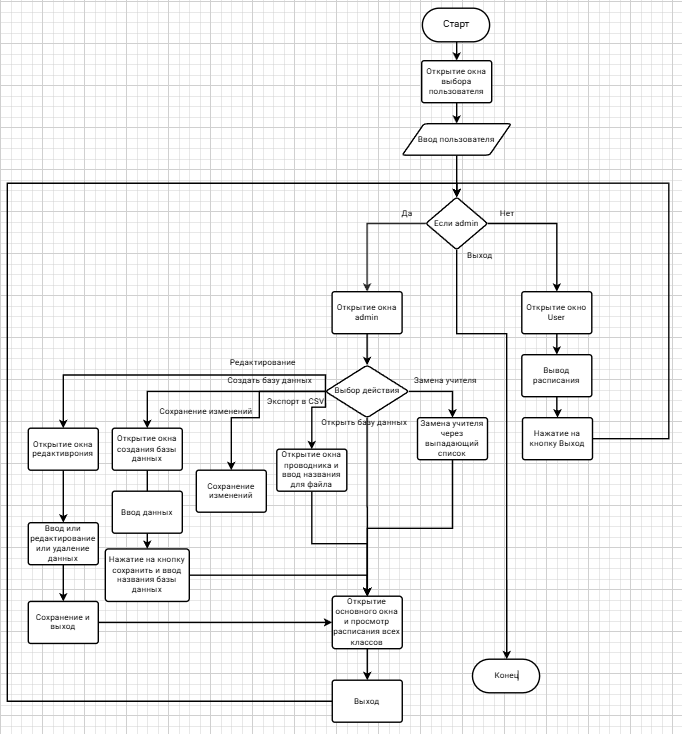
**Приложение №4**

Схема 1 – Блок-схема программы.