**Введение**

Не всегда дошкольные образовательные учреждения используют автоматизированные информационные системы, составляя различные документы вручную. Именно поэтому существует потребность в автоматизации работы некоторых детских садов.

Автоматизация деятельности дает возможность объединить все рабочие процессы в одном информационном пространстве.

Одним из плюсов автоматизации дошкольных образовательных учреждений является то, что повышается производительность труда и оптимизируется работа над такими рутинными вещами, как создание отчетов, оформление договоров.

Целью разработки является создание автоматизированной информационной системы учета посещаемости детского сада. Использование информационных технологий в дошкольных образовательных учреждениях позволяет реализовать принципиально новые и наиболее рациональные методы управления, вывести на более высокий уровень качество государственных услуг.

Для реализации цели необходимо решить следующие задачи:

* проанализировать предметную область;
* сформировать требования к системной разработке;
* произвести выбор инструментальных и программных средств;
* реализовать автоматизированную информационную систему;
* протестировать автоматизированную информационную систему;
* разработать техническую и эксплуатационную документацию.

1. **Разработка системного проекта**

**1.1 Назначение разработки**

Автоматизированная информационная система учета посещаемости детского сада предназначена для минимизации затрат времени, материальных, трудовых ресурсов в ходе деятельности детского сада.

* 1. **Требования к функциональным характеристикам**

Автоматизированная информационная система учета посещения детского сада должна обеспечивать выполнение следующих функций:

* продуманный интерфейс в соответствии с руководством по стилям;
* авторизация должна обеспечивать вход по двум категориям: воспитатель - имеет ограниченный функционал ведения базы данных, заведующий - имеет полный доступ к ведению базы данных;
* ведение и редактирование данных: добавление, удаление, редактирование;
* просмотр информации: о воспитанниках, о сотрудниках, о родителях;
* контроль переполнения группы;
* оформление договора об оплате услуг;
* оформление отчета об посещаемости: день, месяц;
* поиск информации с использованием фильтра: по статусу, по дате.

Входная информация:

* личные данные: воспитанников, родителей, персонала;

Выходная информация:

* договор оплаты услуг;
* отчеты посещаемости.

**3.2 Требования к надежности и безопасности**

Разрабатываемое ПО должно иметь:

* контроль ввода данных;
* резервное копирование БД;
* разграничение прав доступа.

**1.4 Требования к составу и параметрам технических средств**

Минимальные системные требования:

* не ниже Windows 7;
* тактовая частота процессора – 2 ГГц;
* объем оперативной памяти 2 Гб;
* объем свободного дискового пространства не менее 500 МБ;
* разрешение монитора 1366х768;
* наличие устройства чтения компакт-дисков.

**1.5 Требование к информационной и программной совместимости**

Для корректной работы системы на компьютере должно быть установлено следующее программное обеспечение: Microsoft Word 2007, пакет .NET Framework 4.0 или выше

1. **Разработка технического проекта**
   1. **Анализ требований и определение спецификаций ПО**
      1. **Выбор технологии проектирования**

При проектировании программного продукта применяются два подхода: структурный и объектно-ориентированный.

Метод структурного проектирования - это регламентированная последовательность действий, которая позволяет разработать структуру аппаратных и программных средств встраиваемой системы, удовлетворяющих техническим требованиям к проектируемому устройству.

Первым шагом этой последовательности действия является как можно более полное описание технических требований к будущей системе. В подавляющем большинстве случаев технические требования формулирует не тот, кто потом реализует систему. Поэтому технические требования должны быть как можно более точно доведены до исполнителя. Исполнитель обязан подробно исследовать предложенные ему технические требования, понять их обоснованность и согласованность. Вывод: структурное проектирование использует определение проблемы как путь к определению решения этой проблемы.

Существует несколько типичных методов структурного проектирования:

- нисходящее проектирование;

- модульное проектирование.

Основная суть структурного проектирования заключается в том, что логика решения задачи должна соответствовать логике организации программного кода, алгоритм решения задачи должен быть ясен из исходного текста программы. Идеи структурного проектирования и поддержку подпрограмм реализуют процедурные языки программирования. Как показала практика, данный метод эффективен при разработке программ, содержащих до нескольких тысяч строк программного кода.

Достоинства структурного подхода:

* является реализация структурного подхода к проектированию ИС по принципу "сверху-вниз", когда каждый функциональный блок может быть декомпозирован на множество подфункций и т.д., выполняя, таким образом, модульное проектирование ИС;
* функциональные методики в целом лучше дают представление о существующих функциях в организации, о методах их реализации, причем чем выше степень детализации исследуемого процесса, тем лучше они позволяют описать систему.

Недостатки структурного подхода:

* функциональные модели, как правило, не пригодны для повторного использования;
* модульная структура программ;
* нисходящем проектировании, используемом при выделении объектов.

Объектно-ориентированный подход использует объектную декомпозицию. При этом статическая структура системы описывается в терминах объектов и связей между ними, а поведение системы описывается в терминах обмена сообщениями между объектами. Каждый объект системы обладает своим собственным поведением, моделирующим поведение объекта реального мира.

Достоинства ООП:

* объектный подход дает возможность создавать модели меньшего размера путем использования общих механизмов, обеспечивающих необходимую экономию выразительных средств;
* использование объектного подхода существенно повышает уровень унификации разработки и пригодность для повторного использования, что ведет к созданию среды разработки и переходу к сборочному созданию моделей;
* объектный подход позволяет хорошо описать систему на уровне общего описания системы, т.к. он основан на понятии сценария использования.

Недостатки ООП:

К недостаткам объектно-ориентированного подхода относятся высокие начальные затраты. Этот подход не дает немедленной отдачи. Эффект от его применения сказывается после разработки двух–трех проектов и накопления повторно используемых компонентов. Структурный подход не требует высоких начальных затрат.

При проектировании автоматизированной информационной системы по ведению базы данных детского сада были использованы структурное проектирование (построение ER-диаграммы) и объектно-ориентированное проектировании (построение диаграммы вариантов использования и диаграммы деятельности).

* + 1. **Построение моделей**

При объектно-ориентированном проектировании были разработаны диаграмма вариантов использования, диаграмма деятельности, ER-диаграмма.

Диаграмма вариантов использования позволяет наглядно представить ожидаемое поведение программной системы. Основные понятия диаграммы вариантов использования являются: действующее лицо, вариант использования и связь.

Условные обозначения диаграммы вариантов использования:

Актёр – действующее лицо, которое может взаимодействовать с системой. Он выступает как роль, которая определенно относится к программной системе.

Актёр представлен в соответствии с рисунком 2.1.2.1.

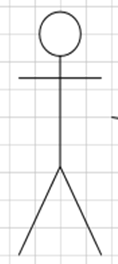


Рисунок 2.1.2.1 – Актёр

Вариант использования - внешняя спецификация последовательности действий, которые система или другая сущность могут выполнять в процессе взаимодействия с актерами. Вариант использования представлен в соответствии с рисунком 2.1.2.2.



Рисунок 2.1.2.2 – Вариант использования

Связь – взаимодействие действующих лиц или соответствующих вариантов использования. Представлено в соответствии с рисунком 2.1.2.3.



Рисунок 2.1.2.3 – Связь

Диаграмма вариантов использования представлена в соответствии с рисунком 2.1.2.4.

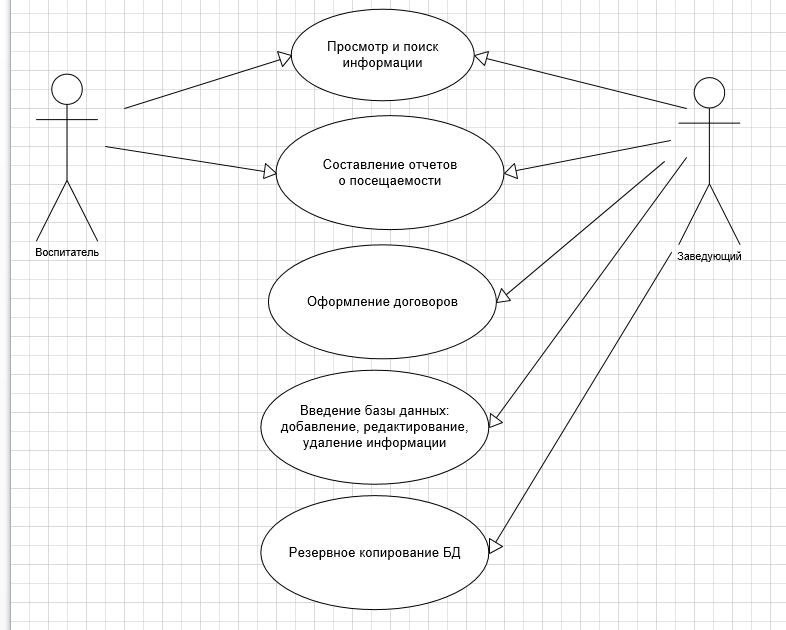


Рисунок 2.1.2.4 – Диаграмма вариантов использования

Сценарий диаграммы использования дополняет диаграмму, раскрывая содержание и логическую последовательность отдельных действий, которые выполняются системой и актёрами.

Сценарий диаграммы вариантов использования представлен в соответствии с таблицей 2.1.

Таблица А.1 – Сценарий диаграммы вариантов использования

|  |  |
| --- | --- |
| Вариант использования | Оформление договора купли-продажи |
| Актеры | Заведующий, воспитатель |
| Краткое описание | Заведующий – может добавлять, удалять, редактировать информацию в БД, производить поиск информации, составлять договора. Воспитатель может просматривать информацию, оформление отчетов о посещаемости. |
| Цель | Учет посещаемости детского сада |
| Тип | Базовый |
| Ссылки на другие варианты использования | Ведение базы данных, просмотр базы данных, поиск информации по базам данных, составление договоров, составление отчетов, резервное копирование БД. |

Диаграмма деятельности - диаграмма, на которой показаны действия, состояния которых описаны на [диаграмме состояний](https://ru.wikipedia.org/wiki/%D0%94%D0%B8%D0%B0%D0%B3%D1%80%D0%B0%D0%BC%D0%BC%D0%B0_%D1%81%D0%BE%D1%81%D1%82%D0%BE%D1%8F%D0%BD%D0%B8%D0%B9_(UML)). Под деятельностью понимается спецификация исполняемого поведения в виде координированного последовательного и параллельного выполнения подчинённых элементов — вложенных видов деятельности и отдельных действий, соединённых между собой потоками, которые идут от выходов одного узла ко входам другого.

Условные обозначения диаграммы деятельности:

Под началом понимается включение программы, представлено в соответствии с рисунком 2.1.2.5.



Рисунок 2.1.2.5 – Начало

Под условие понимается выбор какого-либо действия, представлено в соответствии с рисунком 2.1.2.6.

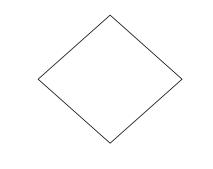


Рисунок 2.1.2.6 – Условие

Под действием понимается выполнение какой-либо операции программой, представлено в соответствии с рисунком 2.1.2.7.



Рисунок 2.1.2.7 – Действие

Под связью понимается соединение определенных операций, представлено в соответствии с рисунком 2.1.2.8.



Рисунок 2.1.2.8 – Связь

Под разделением связей понимается выбор одного из нескольких действий, представлено в соответствии с рисунком 2.1.2.9.



Рисунок 2.1.2.9 – Разделение связей

Под концом понимается завершение программы, либо выход из неё, представлено в соответствии с рисунком 2.1.2.10.



Рисунок 2.1.2.10 – Конец

Диаграмма деятельности представлена в соответствии с рисунком 2.1.2.11.

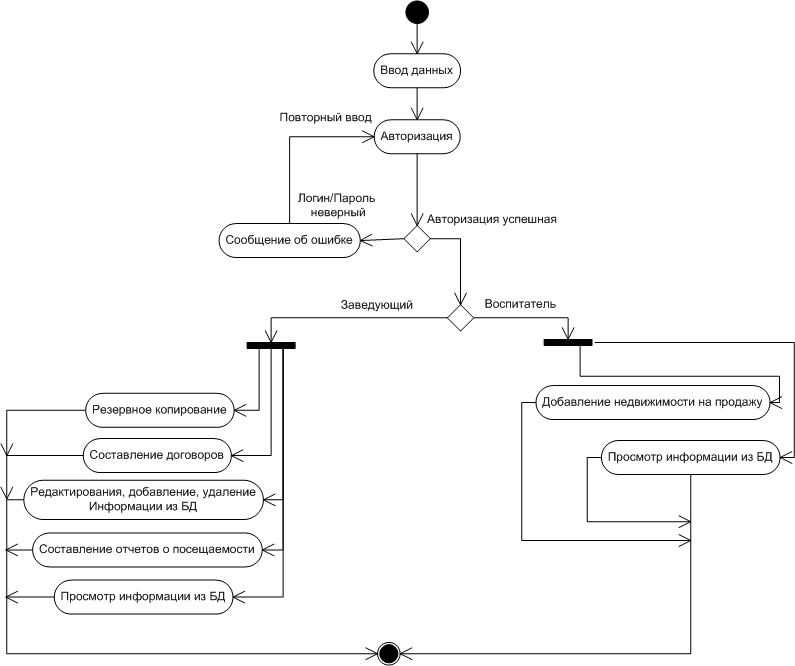


Рисунок 2.1.2.11 – Диаграмма деятельности

ER – диаграмма обеспечивает стандартный способ определения данных и отношений между ними. Она включает сущности и взаимосвязи, отражающие основные бизнес-правила предметной области.

Основные понятия ER – диаграмм.

Сущность – это класс однотипных объектов, информация о которых должна быть учтена в модели.

Представлено в соответствии с рисунком 2.1.2.11.



Рисунок 2.1.2.11 – Сущность

Атрибут сущности – это именованная характеристика, являющаяся некоторым свойством сущности, представлен в соответствии с рисунком 2.1.2.12.



Рисунок 2.1.2.12 – Сущность с атрибутами

Ключ сущности – это не избыточный набор атрибутов, значения которых в совокупности являются уникальными для каждого экземпляра сущности. Ключевые атрибуты изображаются подчёркиванием, представлено в соответствии с рисунком 2.1.2.13.



Рисунок 2.1.2.13 – Сущность с ключом

Связь – это некоторая ассоциация между двумя сущностями. Представлено в соответствии с рисунком 2.1.2.14.

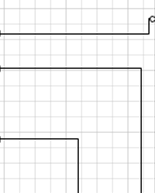


Рисунок 2.1.2.14 – Связи

ER – диаграмма представлена в соответствии с рисунком 2.1.2.15.

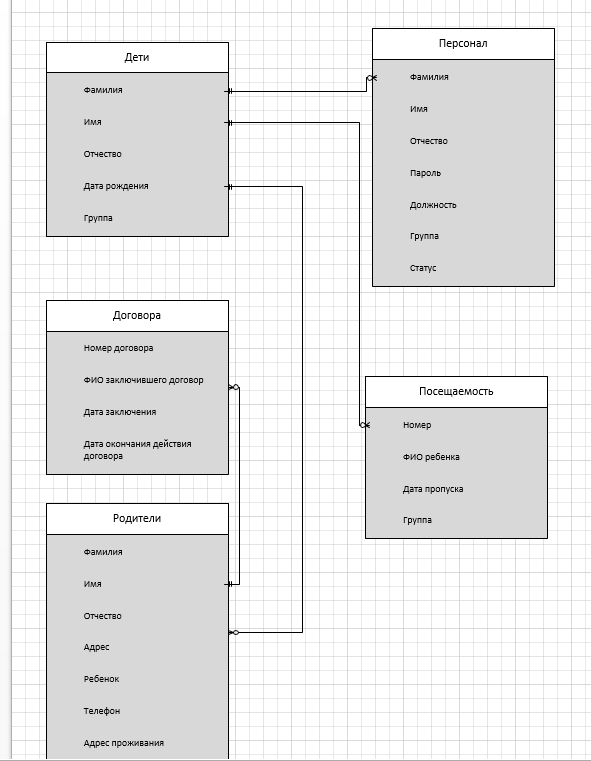


Рисунок 2.1.2.15 - ER – диаграмма

* 1. **Детальное проектирование ПО**

Прототипы могут быть горизонтальные и вертикальные, одноразовые и эволюционные, бумажные и электронные.

Горизонтальный прототип моделирует интерфейс пользователя приложения, не затрагивая логики обработки и структур данных.

Вертикальный прототип направлен не столько на проектирование интерфейса пользователя, сколько на реализацию вертикального «среза» системы, затрагивая все уровни ее реализации.

Одноразовый прототип создается, когда нужно быстро получить маркер разрабатываемой программной системы, те или иные ее аспекты и компоненты.

Эволюционный прототип создается как первое приближение системы, призванное стать впоследствии самой системой.

Бумажный прототип наброски интерфейсов на бумаге.

При проектировании ПП продукта был выбран горизонтальный прототип, так как его используют, когда необходимо прояснить нечёткие требования, имеющие многоальтернативную реализацию. Для горизонтальных прототипов не обязательно использовать программную среду реализации, в которой будет разрабатываться конечный вариант системы.

Прототип авторизации пользователя представлен в соответствии с рисунком 2.1.1.

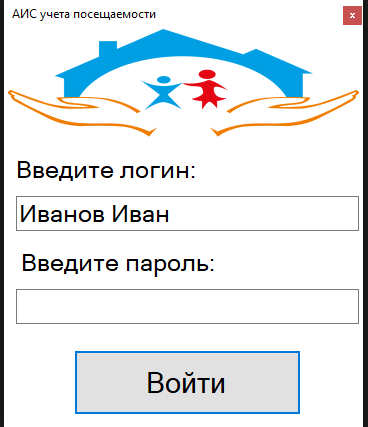


Рисунок 2.1.1 – Прототип окна авторизации

Прототип главной страницы представлен в соответствии с рисунком 2.1.2.

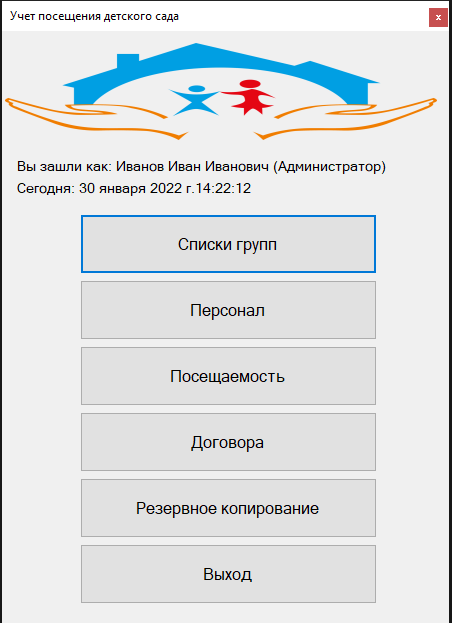


Рисунок 2.1.2 – Прототип главного окна

Горизонтальный прототип списков групп представлен в соответствии с рисунком 2.1.3.

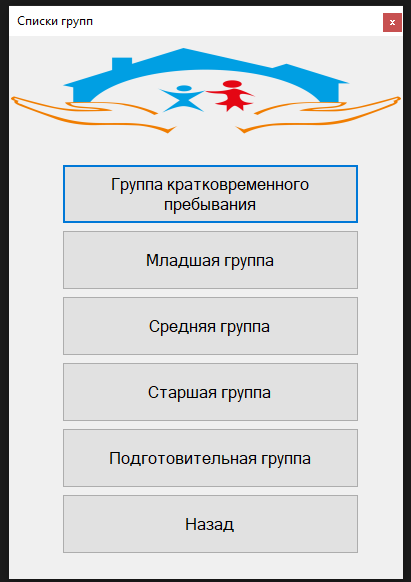


Рисунок 2.1.3. – Прототип списков групп

Горизонтальный прототип окна списка воспитанников группы представлен в соответствии с рисунком 2.1.4.

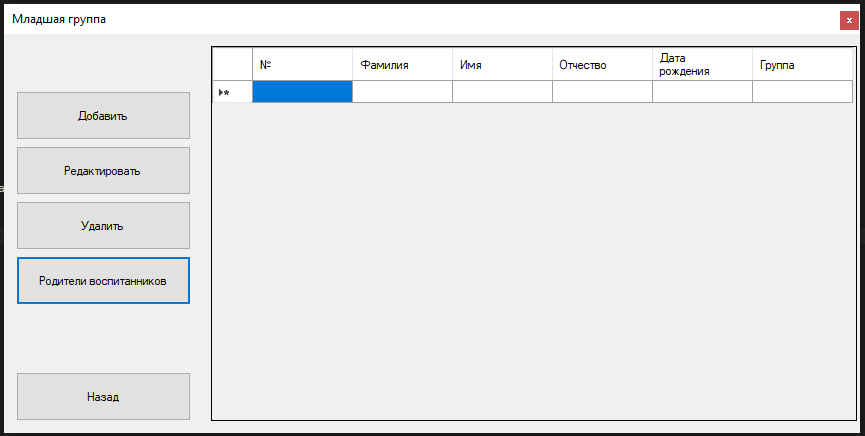


Рисунок 2.1.4 – Прототип формы списка воспитанников группы Горизонтальный прототип окна добавления нового воспитанника в группу представлен в соответствии с рисунком 2.1.5.

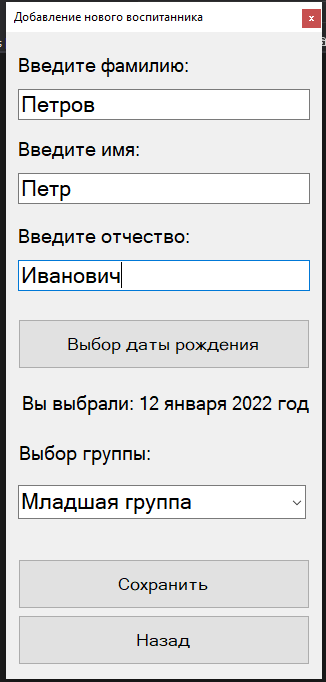


Рисунок 2.1.5– Прототип окна добавления воспитанника

Горизонтальный прототип окна редактирования информации о воспитаннике представлен в соответствии с рисунком 2.1.6.

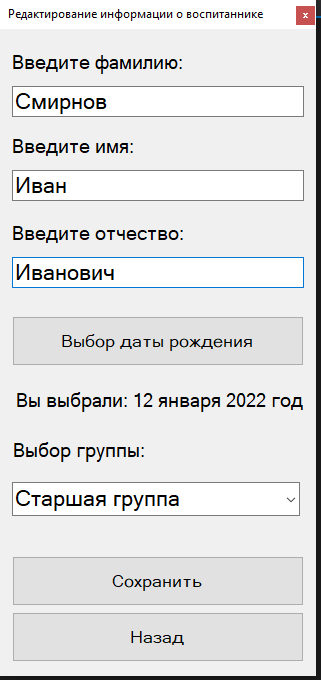


Рисунок 2.1.6– Прототип окна редактирования воспитанников

Прототип таблицы клиентов представлен в соответствии с рисунком 2.1.7.

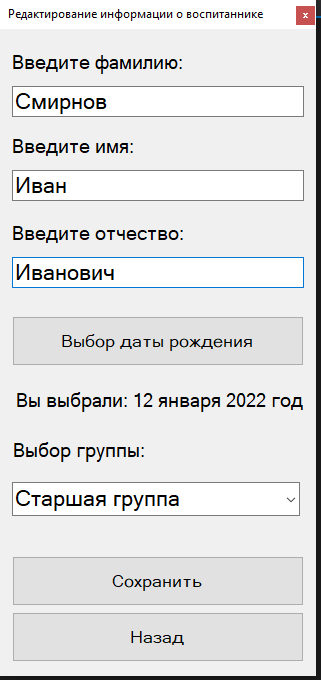


Рисунок 2.1.7– Прототип окна редактирования информации о воспитаннике

Прототип таблицы сотрудников представлен в соответствии с рисунком 2.1.8.

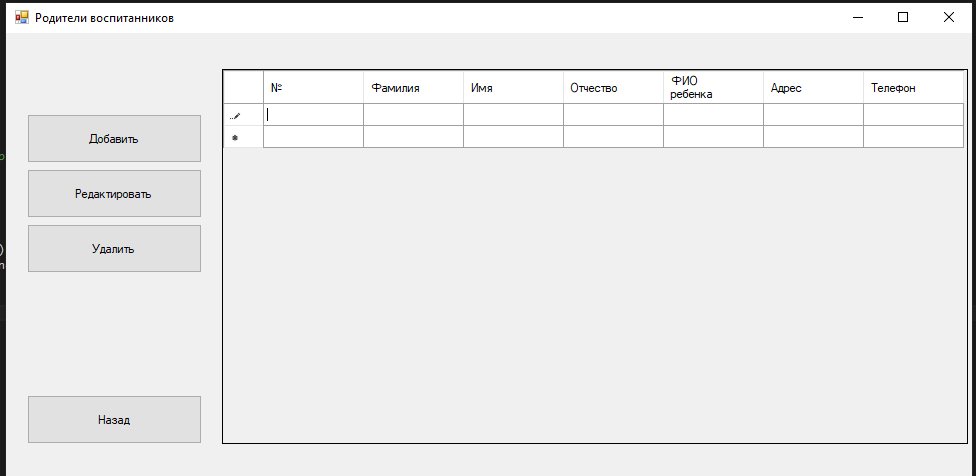


Рисунок 2.1.8– Прототип окна просмотра информации о родителях Прототип таблицы договоров представлен в соответствии с рисунком 2.1.9.

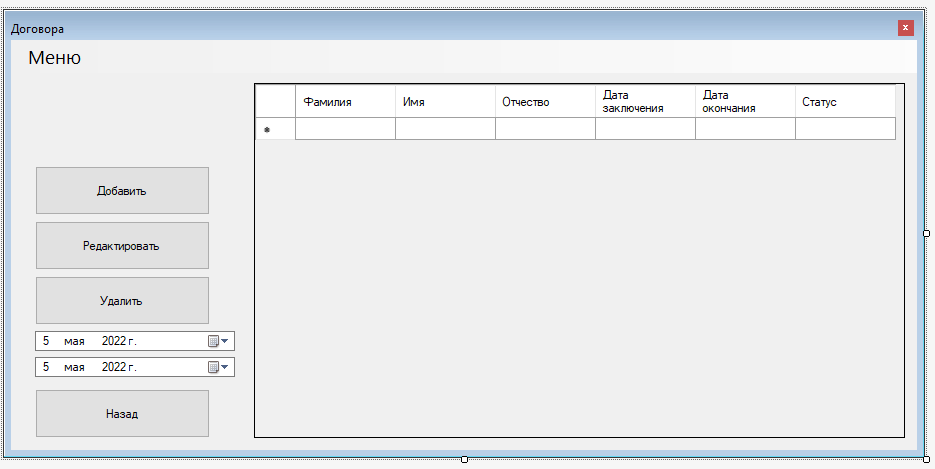


Рисунок 2.1.9 – Прототип окна просмотра таблицы договоров

Прототип окна добавления пропуска представлен в соответствии с рисунком 2.1.10.

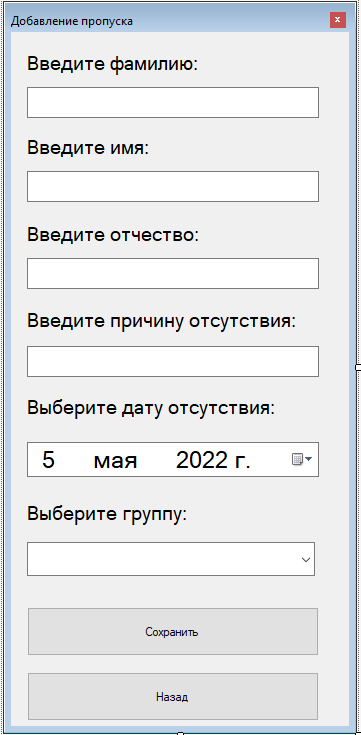


Рисунок 2.1.10 – Прототип окна просмотра добавления пропуска

Прототип печати посещаемости детского сада за месяц представлен в соответствии с рисунком 2.1.11.

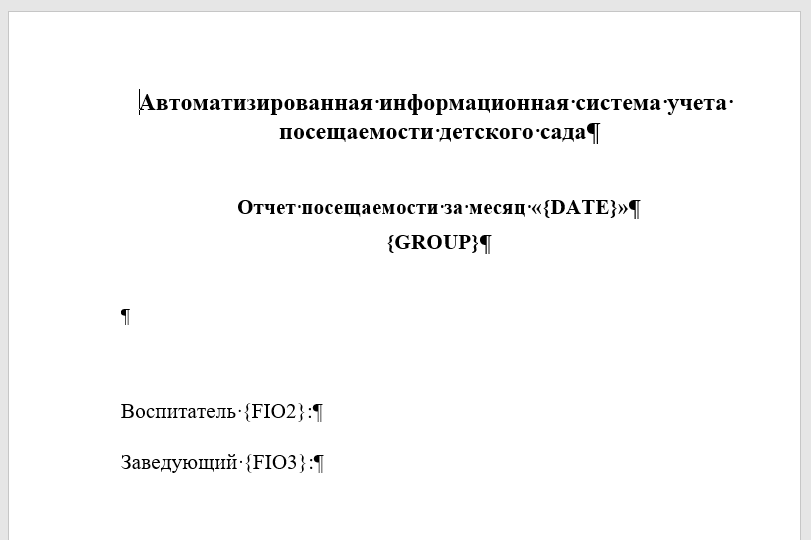


Рисунок 2.1.11 – Прототип печати посещаемости детского сада за месяц

Прототип печати посещаемости детского сада за день представлен в соответствии с рисунком 2.1.11.

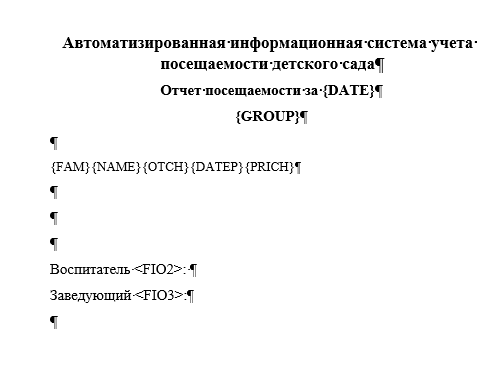


Рисунок 2.1.11 – Прототип печати посещаемости детского сада за день

1. **Реализация**
   1. **Обоснование выбора средств разработки**

При проектировании приложения было выбрано: операционная система, среда разработки, язык программирования и база данных.

К основным ОС относятся: MS Windows, Linux, MacOS.

MS Windows – семейство [коммерческих](https://ru.wikipedia.org/wiki/%D0%9A%D0%BE%D0%BC%D0%BC%D0%B5%D1%80%D1%87%D0%B5%D1%81%D0%BA%D0%BE%D0%B5_%D0%BF%D1%80%D0%BE%D0%B3%D1%80%D0%B0%D0%BC%D0%BC%D0%BD%D0%BE%D0%B5_%D0%BE%D0%B1%D0%B5%D1%81%D0%BF%D0%B5%D1%87%D0%B5%D0%BD%D0%B8%D0%B5) [операционных систем](https://ru.wikipedia.org/wiki/%D0%9E%D0%BF%D0%B5%D1%80%D0%B0%D1%86%D0%B8%D0%BE%D0%BD%D0%BD%D0%B0%D1%8F_%D1%81%D0%B8%D1%81%D1%82%D0%B5%D0%BC%D0%B0)(OC) корпорации [Microsoft](https://ru.wikipedia.org/wiki/Microsoft), ориентированных на управление с помощью [графического интерфейса](https://ru.wikipedia.org/wiki/%D0%93%D1%80%D0%B0%D1%84%D0%B8%D1%87%D0%B5%D1%81%D0%BA%D0%B8%D0%B9_%D0%B8%D0%BD%D1%82%D0%B5%D1%80%D1%84%D0%B5%D0%B9%D1%81_%D0%BF%D0%BE%D0%BB%D1%8C%D0%B7%D0%BE%D0%B2%D0%B0%D1%82%D0%B5%D0%BB%D1%8F).

Плюсы ОС Windows:

* система достаточно проста в использовании;
* большинство программ как платных, так и бесплатных поддерживают именно эту операционную систему;
* поддержка различного оборудования (веб-камеры, принтеры и т.д.);
* стабильность работы.

Минусы ОС Windows:

* платная ОС;
* большинство вирусов написано под эту ОС;
* довольно частые сбои.

MacOS — операционная система от компании Apple. Установлена как основная операционная система на продуктах компании.

Плюсы ОС MacOS:

* удобный пользовательский интерфейс;
* стоимость входит в стоимость оборудования;
* высокая производительность;
* низкая возможность заражения вирусами.

Минусы ОС MacOS:

* дорогое программное обеспечение;
* нельзя установить на оборудование другой компании;

Linux — это **операционная система, ядро которой распространяется на бесплатной основе.** Она состоит из ядра системы и набора небольших программ, взаимодействующих с этим ядром.

Плюсы ОС Linux:

* бесплатное свободно-распространяемое ПО;
* практически нет вирусов;
* не зависит от разработчика.

Минусы ОС Linux:

* поддержка небольшого количества оборудования;
* небольшое количество прикладных программ.

Для разработки была выбрана ОС Windows, т.к. система достаточно проста в использовании, поддерживает большое количество программ, как платных, так и бесплатны и обеспечивает стабильность работы.

К основным IDE относятся Visual Studio, NetBeans.

Visual Studio – это интегрированная среда разработки программного обеспечения от компании Microsoft. С помощью Visual Studio можно создавать приложения для Windows, iOS, Android и других платформ.

Плюсы:

* поддержка множества языков при разработке;
* меньше кода для написания;
* интуитивный стиль кодирования;
* возможности отладки;
* поддержка Microsoft.

NetBeans – бесплатная среда разработки с открытым исходным кодом. Подходит для редактирования существующих проектов или создания новых. NetBeans предлагает простой интерфейс, который поставляется с большим количеством удобных шаблонов проектов.

Плюсы:

* интуитивный интерфейс;
* динамические и статические библиотеки;
* возможность осуществлять удаленное развертывание;
* совместимость с платформами Windows, Linux, OS X и Solaris.

В качестве среды программирования была выбрана Visual Studio, потому что присутствует поддержка Microsoft, русифицирование и поддержка графического редактора для создания проекта.

К основным языкам программирования относятся С#, Python, C++.

C# — объектно-ориентированный язык программирования, который имеет статическую типизацию, перегрузку операторов (в том числе операторов явного и неявного приведения типа), делегаты, атрибуты, события, свойства, обобщённые типы и методы, итераторы, анонимные функции с поддержкой замыканий, комментарии в формате XML.

Python — высокоуровневый язык программирования общего назначения, ориентированный на повышение производительности разработчика и читаемости кода. В то же время стандартная библиотека включает большой объём полезных функций.

C++ — компилируемый, статически типизированный язык программирования общего назначения. Язык имеет богатую стандартную библиотеку, которая включает в себя распространённые контейнеры и алгоритмы, ввод-вывод, регулярные выражения, поддержку многопоточности и другие возможности. C++ сочетает свойства как высокоуровневых, так и низкоуровневых языков.

В качестве языка программирования был выбран язык программирования C#, т.к. это высокоуровневый язык программирования, он удобен в использовании и поддерживает графический интерфейс.

К основным базам данных относятся SQLite, MySQL и Microsoft Access.

SQLite - библиотека, встраиваемая в приложение, которое её использует. Будучи файловой БД, она предоставляет отличный набор инструментов для более простой обработки любых видов данных. Когда приложение использует SQLite, их связь производится с помощью функциональных и прямых вызовов файлов, содержащих данные, а не какого-то интерфейса, что повышает скорость и производительность операций.

Плюсы:

* файловая;
* стандартизированная;
* отлично подходит для разработки и даже тестирования.

MySQL — это самая популярная из всех крупных серверных БД. MySQL - это СУБД, которая поддерживает SQL. SQL — это универсальный язык, который поддерживают все системы управлением базами данных, поэтому, зная его, вы сможете работать с любыми СУБД. MySQL предлагает широкий функционал. Приложения общаются с базой данных через процесс-демон.

Плюсы:

* легко устанавливается;
* поддерживает большую часть функционала SQL;
* безопасность;
* может работать с действительно большими объёмами данных, и неплохо походит для масштабируемых приложений.

Microsoft Access – система управления базами данных, имеющая широкий спектр функций, включая связанные запросы, связь с внешними таблицами и базами данных. Благодаря встроенному языку VBA, в самом Access можно писать приложения, работающие с базами данных.

Плюсы:

* простой графический интерфейс, который позволяет не только создавать собственную базу данных, но и разрабатывать приложения, используя встроенные средства;
* хранит все данные в одном файле, хотя и распределяет их по разным таблицам, как и положено реляционной СУБД. К этим данным относится не только информация в таблицах, но и другие объекты базы данных;
* распространенность, которая обусловлена тем, что Access является продуктом компании Microsoft.

В качестве базы данных была выбрана MySQL, т.к она удобна в использовании, может работать с большим количеством информации и проста в использовании. Имеет широкий спектр функций, включая связанные запросы, связь с внешними таблицами и базами данных.

**3.2 Описание основных программных модулей**

Основные программные модули представлены в таблице 3.2.1.

Таблица 3.2.1 – Основные программные модули

|  |  |
| --- | --- |
| **Название модуля** | **Описание** |
| 1 | 2 |
| AuthF.cs | Форма авторизации |
| public AuthF() | Инициализация формы авторизации |
| private void button1\_Click(object sender, EventArgs e) | Кнопка авторизации |
| AddChildrenF.cs | Форма добавления воспитанника |
| public AddChildrenF(string NameF) | Инициализация формы добавления воспитанника |
| private void button2\_Click(object sender, EventArgs e) | Кнопка добавления воспитанника в базу данных |
| private void button3\_Click(object sender, EventArgs e) | Кнопка возвращения на главное меню |
| private void textBox1\_KeyPress(object sender, KeyPressEventArgs e) | Контроль ввода данных |
| private void textBox2\_KeyPress(object sender, KeyPressEventArgs e) |

Продолжение таблица 3.2.1.

Продолжение таблица 3.2.1.

|  |  |
| --- | --- |
| private void textBox3\_KeyPress(object sender, KeyPressEventArgs e) |  |
| AddParents.cs | Форма добавления родителей воспитанника |
| public AddParents () | Инициализация формы добавления родителя |
| private void button1\_Click(object sender, EventArgs e) | Кнопка возвращения на главное меню |
| private void AddParents\_Load(object sender, EventArgs e) | Отслеживание загрузки формы добавления родителя |
| private void button2\_Click(object sender, EventArgs e) | Кнопка добавления родителя |
| private void textBox1\_KeyPress(object sender, KeyPressEventArgs e) | Контроль ввода данных |
| private void textBox2\_KeyPress(object sender, KeyPressEventArgs e) |
| private void textBox3\_KeyPress(object sender, KeyPressEventArgs e) |
| private void textBox4\_KeyPress(object sender, KeyPressEventArgs e) |
| private void textBox5\_KeyPress(object sender, KeyPressEventArgs e) | Контроль ввода данных |
| private void maskedTextBox2\_KeyPress(object sender, KeyPressEventArgs e) |
| Attendance.cs | Форма просмотра пропусков |
| private void button4\_Click(object sender, EventArgs e) | Кнопка возвращения на главную форму |
| private void button6\_Click(object sender, EventArgs e) | Кнопка перехода на форму добавления пропуска |
| private void Form6\_Load(object sender, EventArgs e) | Отслеживание загрузки формы пропусков |
| private void группаКраткосрочногоВремяПребыванияToolStripMenuItem\_Click(object sender, EventArgs e) | Кнопка для оформления отчета для группы краткосрочного время пребывания |

Продолжение таблица 3.2.1.

|  |  |
| --- | --- |
| private void младшаяГруппаToolStripMenuItem\_Click(object sender, EventArgs e) | Кнопка для оформления отчета для младшей группы |
| private void button3\_Click(object sender, EventArgs e) | Кнопка удаления пропуска |
| private void средняяГруппаToolStripMenuItem1\_Click(object sender, EventArgs e) | Кнопка для оформления отчета для средней группы |
| private void старшаяГруппаToolStripMenuItem1\_Click(object sender, EventArgs e) | Кнопка для оформления отчета для старшей группы |
| private void подготовительнаяГруппаToolStripMenuItem1\_Click(object sender, EventArgs e) | Кнопка для оформления отчета для младшей группы |
| private void button2\_Click(object sender, EventArgs e) | Кнопка перехода на форму для редактирования пропуска |
| private void button2\_Click(object sender, EventArgs e) | Кнопка сохранения заключения договора |
| private void button1\_Click(object sender, EventArgs e) | Кнопка перехода на форму добавления договора |
| private void button2\_Click(object sender, EventArgs e) | Кнопка редактирования информации о воспитаннике |
| private void button1\_Click\_1(object sender, EventArgs e) | Кнопка редактирования информации о пропуске воспитанника |
| private void button7\_Click(object sender, EventArgs e) | Кнопка выхода из программы |
| private void button1\_Click(object sender, EventArgs e) | Кнопка отображения основной формы меню группы |
| private void button5\_Click(object sender, EventArgs e) | Кнопка резервного копирования БД за администратора |

**4 Тестирование программного продукта**

Тестирование программного обеспечения – процесс исследования, испытания [программного продукта](https://ru.wikipedia.org/wiki/%D0%9F%D1%80%D0%BE%D0%B3%D1%80%D0%B0%D0%BC%D0%BC%D0%BD%D0%BE%D0%B5_%D0%BE%D0%B1%D0%B5%D1%81%D0%BF%D0%B5%D1%87%D0%B5%D0%BD%D0%B8%D0%B5), имеющий своей целью проверку соответствия между реальным поведением программы и её ожидаемым поведением на конечном наборе тестов, выбранных определенным образом.

Существует несколько признаков, по которым принято производить классификацию видов тестирования. Обычно выделяют следующие:

- тестирование переходов между состояниями – тестирование состояния программы и меню программы;

- условия гонок – вмешательство в работу программы, когда она выполняет переход между состояниями, обработку данных, ввод-вывод данных;

- нагрузочные испытания – один из видов тестирования граничных условий. Сначала запускают программу в нормальных условиях, а затем в условиях, для которой она не предназначена;

- тестирование функциональной эквивалентности – при тестировании функциональной эквивалентности сравниваются результаты вычислений разными программами одной и той же математической функции.

Тестирование программного продукта представлено в таблице 4.1

Таблица 4.1- Тестирование

|  |  |  |
| --- | --- | --- |
| **№** | **Испытание** | **Результат** |
| **Форма «AuthF»** | | |
| **1** | Ввод логина и пароля и нажатие на кнопку «Войти» | При правильном вводе пароля и логина происходит переход на главную форму при неправильном сообщение о вводе неверных данных. |
| **Форма «MainF»** | | |
| **2** | Нажатие на кнопку «Списки групп» | Закрытие формы главного меню и открытие формы списка групп. |

Продолжение таблицы 4.1

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
| **3** | | Нажатие на кнопку «Персонал» | Закрытие формы главного меню и открытие формы просмотра персонала | |
| **4** | | Нажатие на кнопку «Посещаемость» | Закрытие формы главного меню и открытие формы просмотра посещаемости | |
| **5** | | Нажатие на кнопку «Договоры» | Закрытие формы главного меню и открытие формы просмотра договоров | |
| **3** | | Нажатие на кнопку «Резервное копирование» | Выводится сообщение с подтверждением действия. После подтверждения делается резервная копия БД. | |
| **4** | | Нажатие на кнопку «Выход» | Выводится сообщение с подтверждением действия. После подтверждения форма главного меню закрывается. | |
| **Форма «GroupsF»** | | | | |
| **5** | Нажатие на выбранную кнопку соответствующей группы | | | Закрытие формы со списком групп, открытие формы с воспитанниками группы. |
| **Форма «GroupF»** | | | | |
| **6** | Нажатие на кнопку «Добавить» | | | Закрытие формы со списком воспитанников, открытие формы добавления нового воспитанника. |
| **7** | Нажатие на кнопку «Редактировать» | | | Закрытие формы со списком воспитанников, открытие формы редактирования воспитанника. |
| **8** | Нажатие на кнопку «Удалить» | | | Выводиться сообщение с подтверждением действия, в случае подтверждения действия запись удаляется. |
| **9** | Нажатие на кнопку «Родители воспитанников» | | | Закрытие формы со списком воспитанников, открытие формы со списком родителей воспитанников. |
| **Форма «Attedance»** | | | | |
| **10** | Нажатие на кнопку «Добавить» | | | Закрытие формы со списком пропусков, открытие формы добавления нового пропуска. |
| **11** | Нажатие на кнопку «Редактировать» | | | Закрытие формы со списком пропусков, открытие формы редактирования пропуска. |

Продолжение таблицы 4.1

|  |  |  |
| --- | --- | --- |
| **12** | Нажатие на кнопку «Удалить» | Выводиться сообщение с подтверждением действия, в случае подтверждения действия запись удаляется. |
| **13** | Нажатие на кнопку «Посещаемость» | Выпадает меню с выбором |
| **14** | Нажатие на кнопку «Отчет за день» | Выводится отчет по посещаемости за текущий день в Word |
| **15** | Нажатие на кнопку «Отчет за месяц» | Выводится отчет по посещаемости за выбранный месяц в Word |
| **Форма «Staff»** | | |
| **16** | Нажатие на кнопку «Добавить» | Закрытие формы со списком сотрудников, открытие формы добавления нового сотрудника. |
| **17** | Нажатие на кнопку «Редактировать» | Закрытие формы со списком сотрудников, открытие формы редактирования информации о сотруднике. |
| **18** | Нажатие на кнопку «Удалить» | Выводиться сообщение с подтверждением действия, в случае подтверждения действия запись удаляется. |
| **19** | Нажатие на кнопку «Фильтр» | Выпадает меню с выбором |
| **20** | Нажатие на кнопку «Показать только администраторов» | Информация фильтруется, выводятся список администраторов |
| **21** | Нажатие на кнопку «Показать только пользователей» | Информация фильтруется, выводятся список пользователей |
| **22** | Нажатие на кнопку «Сбросить фильтры» | Фильтры сбрасываются, информация обновляется |
| **Форма «ParentsF»** | | |
| **23** | Нажатие на кнопку «Добавить» | Закрытие формы со списком родителей, открытие формы добавления нового родителя. |
| **24** | Нажатие на кнопку «Редактировать» | Закрытие формы со списком родителей, открытие формы редактирования информации о родителе. |
| **25** | Нажатие на кнопку «Удалить» | Выводиться сообщение с подтверждением действия, в случае подтверждения запись удаляется. |

Продолжение таблицы 4.1

|  |  |  |
| --- | --- | --- |
| **Форма «ContractF»** | | |
| **26** | Нажатие на кнопку «Добавить» | Закрытие формы со списком договоров, открытие формы добавления нового договора. |
| **27** | Нажатие на кнопку «Редактировать» | Закрытие формы со списком договоров, открытие формы редактирования договора. |
| **28** | Нажатие на кнопку «Удалить» | Выводиться сообщение с подтверждением действия, в случае подтверждения действия запись удаляется. |

**Заключение**

В ходе выполнение курсового проекта была разработана автоматизированная информационная система учета посещаемости детского сада. Предназначена минимизации затрат времени, материальных, трудовых ресурсов в ходе деятельности детского сада, за счет облегчения оформления и заключения договора оплаты услуг, учета новых прогулов, а также вывод на печать учет посещаемости за выбранный период времени.

Автоматизированная информационная система выполняет:

- просмотр личных данных: воспитанников, родителей, персонала;

- учет новых прогулов персонала;

- заключение договора оплаты услуг детского сада;

- создание отчета прогулов за выбранный период времени.

Программный продукт содержит все заявленные требования, которые были поставлены на этапе постановки задачи:

- авторизация администратора и пользователя;

- добавление, удаление, редактирование и резервное копирование базы данных администратором;

- поиск информации с использованием фильтра: по статусу, по дате.

- возможность просмотра информации баз данных для администратора;

- возможность просмотра заказов для пользователя;

- заключение договора оплаты услуг;

- создание отчета прогулов за выбранный период времени: день, месяц.

В дальнейшем возможна доработка программы, добавление новых функций по ведению базы данных.

**Литература**

1. Рудаков А.В. "Технология разработки программного обеспечения" -М.:Академия, 2018 – 206 c.
2. Агуров, Павел C#. Сборник рецептов / Павел Агуров. - М.: "БХВ-Петербург", 2018 - 432 c.
3. Бишоп, Дж. C# в кратком изложении / Дж. Бишоп, Н. Хорспул. - М.: Бином. Лаборатория знаний, 2019 - 472 c.
4. Марченко, А. Л. Основы программирования на С# 2.0 / А.Л. Марченко. - М.: Интернет-университет информационных технологий, Бином. Лаборатория знаний, 2021 - 552 c.
5. Смоленцев, Н. К. MATLAB. Программирование на Visual С#, Borland JBuilder, VBA (+ CD-ROM) / Н.К. Смоленцев. - М.: ДМК Пресс, 2011. - 456 c.