Министерство образования Республики Беларусь

Учреждение образования «Белорусский государственный университет информатики и радиоэлектроники»

Факультет компьютерных систем и сетей

Кафедра программного обеспечения информационных технологий

Дисциплина: Языки программирования (ЯП)

ПОЯСНИТЕЛЬНАЯ ЗАПИСКА

к курсовой работе

на тему:

«БАЗА ДАННЫХ КЛИЕНТОВ ИНТЕРНЕТ-МАГАЗИНА»

БГУИР КП 1-40 01 01 008  ПЗ

Студент: гр. 951005 Доведько Д.Ю.

Руководитель: асс. Болтак С.В.

Минск 2020

**СОДЕРЖАНИЕ**

Введение ........................................................................................................ 4

1 Анализ прототипов, литературных источников и формирование

требований к проектируемому программному средству........................ 6

1.1 Понятие базы данных……………...................................................... 6

1.2 Анализ существующих программных продуктов............................. 10

1.3 Формирование требований к разрабатываемому ПС....................... 14

2 Разработка структуры данных................................................................... 16

2.1 Описание используемых структур данных........................................ 18

3 Разработка программного средства........................................................... 22

3.1 Схема программы.................................................................................. 22

3.2 Проектирование окна «Выбор файла, используемого программой» 27

3.3 Проектирование окна списка клиентов............................................... 27

3.4 Проектирование окна статистики добавленных клиентов................ 28

4 Обоснование технических приемов программирования ......................... 30

5 Тестирование................................................................................................ 32

5.1 Результаты тестирования программы................................................. 32

5.2 Подтверждение корректности тестирования..................................... 35

6 Руководство пользователя программы..................................................... 44

Заключение ..................................................................................................... 46

Список использованных источников ........................................................... 47

Приложение. Исходный код программы ..................................................... 47

Ведомость......................................................................................................... 75

**ВВЕДЕНИЕ**

Внедрение информационных технологий во все сферы деятельности обусловлено стремительным развитием науки, качественным скачком возможностей человека, постоянно растущими объемами информации, сложностью происходящих процессов и явлений. Не обошла стороной информатизация и продажу товара.

Интернет предоставляет беспрецедентные возможности повышения продуктивности работы, продажи товаров и услуг на новых быстро расширяющихся рынках, а также реализует недорогой способ глобальных коммуникаций, как внутри любой организации, так и вне ее. Интернет магазин-очень удобная система демонстрации и продажи товаров и услуг, т.к. подходит для размещения большого количества информации, позволяет оперативно обновлять ассортимент, четко контролировать рабочие процессы. В организационной системе наиболее трудоемкими являются процессы, связанные с обработкой информации - сбор, накопление, преобразование, отображение, хранение, передача и вывод.

Цель курсового проектирования - создание базы данных, позволяющей персоналу интернет-магазина регистрировать и обрабатывать заявки клиентов с использованием принципов объектно-ориентированного программирования, принципов объективности, целенаправленности, системности научного исследования и динамических структур данных.

Пояснительная записка содержит несколько разделов:

* Первый раздел пояснительной записки содержит понятие базы данных, анализ существующих программных систем с примерами и постановку общей задачи на курсовую работу.
* Второй раздел пояснительной записки посвящён алгоритму разработки структуры данных.
* Третий раздел посвящён непосредственно разработке программного средства. Он включает разработку схемы и структуры программы, алгоритмическое конструирование и конструирование пользовательского интерфейса.
* В четвёртом разделе пояснительной записки обосновываются технические приёмы программирования, то есть обоснование архитектурного построения программы и использования конкретных возможностей языка программирования.
* Пятый раздел посвящён тестированию отдельных функций и всей системы в целом. Проверяется корректность выполнения программы и её устойчивость к возможным ошибкам, оговаривается анализ результатов.
* Шестой раздел содержит руководство пользователя, в котором отображена последовательность действий, выполняемых пользователем для успешной работы системы.
* В заключении перечислены итоги выполнения курсовой работы.

После всех вышеперечисленных разделов далее размещены:

* список использованной в ходе работы литературы;
* обязательные приложения с текстом программы.

**1 АНАЛИТИЧЕСКИЙ ОБЗОР ЛИТЕРАТУРЫ И СУЩЕСТВУЮЩИХ**

**АНАЛОГОВ**

**1.1** **Понятие базы данных**

Современный Интернет — это огромное хранилище, в котором каждый по своему желанию может найти нужный текст, музыку, учебные материалы, а с некоторого времени и просто купить все, что необходимо - от продуктов, до программного обеспечения. В данное время многие дальновидные руководители компаний совершенно справедливо считают необходимым создание небольшого Интернет-представительства компании, другими словами, все больше появляется коммерческих сайтов, основной целью которых является продвижение товаров и услуг с помощью Интернет.

Однако, что же представляет собой Интернет-магазин, как он создается, в чем его преимущества и в чем недостатки? Прежде всего, Интернет-магазин - это сайт, содержащий подробный каталог товаров с описанием и изображением. Основное отличие от обычного интернет-каталога состоит в том, что товары, представленные в интернет-магазине можно не только увидеть, но и заказать, не вставая с места и не прерывая увлекательного путешествия по просторам Интернет.

Для всех Интернет-магазин свойственен определенный обязательный набор элементов. Но каждый Интернет-магазин работает с огромным количеством данных, для упрощения работы с которыми, а именно для обработки, упорядочивания, хранения, систематизации и т.д. используются базы дынных.

**База данных** — это упорядоченный набор структурированной информации или данных, которые обычно хранятся в электронном виде в компьютерной системе. База данных обычно управляется системой управления базами данных (СУБД). Данные вместе с СУБД, а также приложения, которые с ними связаны, называются системой баз данных, или, для краткости, просто базой данных.

Данные в наиболее распространенных типах современных баз данных обычно формируются в виде строк и столбцов в ряде таблиц, чтобы обеспечить эффективность обработки и запросов данных. Затем можно легко получать доступ к данным, управлять ими, изменять, обновлять, контролировать и упорядочивать.

В большинстве баз данных для записи и запросов данных используется язык структурированных запросов (SQL).

Существует множество различных типов баз данных. Выбор наилучшей базы данных для конкретной организации зависит от того, как эта организация намеревается использовать данные.

* Реляционные базы данных. Реляционные базы данных стали преобладать в 1980-х годах. Элементы в реляционной базе данных организованы в виде набора таблиц со столбцами и строками. Технология реляционных баз данных обеспечивает наиболее эффективный и гибкий способ доступа к структурированной информации.
* Объектно-ориентированные базы данных. Информация в объектно-ориентированной базе данных представлена в форме объекта, как в объектно-ориентированном программировании.
* Распределенные базы данных. Распределенная база данных состоит из двух или более файлов, расположенных на разных узлах. Такая база данных может храниться на нескольких компьютерах, расположенных в одном физическом месте или распределенных по разным сетям.

Хранилища данных. Будучи централизованным репозиторием для данных, хранилище данных представляет собой тип базы данных, специально предназначенной для быстрого выполнения запросов и анализа.

* Базы данных NoSQL. База данных NoSQL, или нереляционная база данных, позволяет хранить и обрабатывать неструктурированные или слабоструктурированные данные (в отличие от реляционной базы данных, задающей структуру содержащихся в ней данных). Популярность баз данных NoSQL растет по мере распространения и усложнения веб-приложений.
* Графовые базы данных. Графовая база данных хранит данные в контексте сущностей и связей между сущностями.
* Базы данных OLTP. База данных OLTP — это быстрая база данных аналитического типа, предназначенная для большого объема транзакций, выполняемых множеством пользователей.

Это лишь некоторые из десятков типов баз данных, используемых в настоящее время. Другие, менее распространенные базы данных, предназначены для очень специфических научных, финансовых и иных задач. Помимо появления новых типов, базы данных развиваются в абсолютно новых направлениях — изменяются подходы к разработке технологий, происходят значительные сдвиги, такие как внедрение облачных технологий и автоматизации. В частности, в последнее время появились следующие базы данных.

* Базы данных с открытым исходным кодом. Такие базы данных имеют открытый исходный код и могут управляться средствами как SQL, так и NoSQL.
* Облачные базы данных. Облачная база данных представляет собой набор структурированных или неструктурированных данных, размещенный на частной, публичной или гибридной платформе облачных вычислений. Существует два типа моделей облачных баз данных: традиционная база данных и база данных как услуга (DBaaS). В модели DBaaS административные задачи и обслуживание выполняются поставщиком услуг.
* Многомодельные базы данных. Многомодельная база данных объединяет разные типы моделей баз данных в единую интегрированную серверную СУБД. Это означает, что она может содержать различные типы данных.
* Документные базы данных/JSON. Базы данных документов предназначены для хранения, извлечения и обработки документоориентированной информации и предоставляют современный способ хранения данных в формате JSON, а не в виде строк и столбцов.
* Автономные базы данных. Самоуправляемые базы данных (также называемые автономными) – это новейшие и самые революционные облачные базы данных, которые используют машинное обучение для автоматизации настройки, защиты, резервного копирования, обновления и других стандартных задач обслуживания, обычно выполняемых администраторами баз данных.

Для базы данных обычно требуется комплексное программное обеспечение, которое называется системой управления базами данных (СУБД). СУБД служит интерфейсом между базой данных и пользователями или программами, предоставляя пользователям возможность получать и обновлять информацию, а также управлять ее упорядочением и оптимизацией. СУБД также упрощает контроль и управление базами данных, позволяя выполнять различные административные операции, такие как мониторинг производительности, настройка, а также резервное копирование и восстановление.

В качестве примеров популярного программного обеспечения для управления базами данных, или СУБД, можно назвать MySQL, Microsoft Access, Microsoft SQL Server, FileMaker Pro, Oracle Database и dBASE.

Обширный сбор данных из Интернета вещей меняет действительность и производственный сектор по всему миру — современные компании имеют доступ к большему количеству данных, чем когда-либо прежде. Прогрессивные организации теперь могут использовать базы данных, чтобы от обычного хранения данных и базовых транзакций перейти к анализу огромных объемов данных из множества систем. Благодаря базам данных и другим средствам вычислений и бизнес-аналитики современные организации могут использовать собираемые ими данные для более эффективной работы, эффективного принятия решений, гибкости и масштабируемости.

Современные крупные корпоративные базы данных нередко поддерживают очень сложные запросы, и предполагается, что они должны предоставлять почти мгновенные ответы на эти запросы. В результате администраторов баз данных постоянно просят использовать самые разные методы для повышения производительности. Вот некоторые из наиболее распространенных вызовов, с которыми они сталкиваются.

* Восприятие значительно возросших объемов данных. Стремительный рост данных от датчиков, подключенных приборов и десятков других источников заставляет администраторов искать способы эффективного управления и упорядочивания данных своих компаний.
* Обеспечение безопасности данных. В наши дни регулярно случаются утечки данных, и хакеры становятся все более изобретательными. Сейчас как никогда важно обеспечить защиту данных, но в то же время их легкую доступность для пользователей.
* Удовлетворение растущих потребностей. В современной, динамичной бизнес-среде компаниям необходим доступ к данным в режиме реального времени – для своевременного принятия решений и использования новых возможностей.
* Управление и обслуживание базы данных и инфраструктуры. Администраторы базы данных должны осуществлять постоянный мониторинг базы данных на наличие проблем, выполнять профилактическое обслуживание, а также устанавливать обновления и исправления программного обеспечения. Но базы данных становятся все более сложными, объемы данных растут, и компании сталкиваются с необходимостью привлечения дополнительных специалистов для мониторинга и настройки баз данных.
* Устранение границ масштабируемости. Если бизнес хочет выжить, он должен развиваться, и возможности управления данными должны расти вместе с ним. Но администраторам баз данных очень сложно предугадать, какие мощности потребуются компании, особенно при использовании локальных баз данных.

Решение всех этих задач может занимать много времени и отвлекать администраторов баз данных от решения стратегических задач.

* 1. **Анализ существующих программных продуктов**

Для базы данных обычно требуется комплексное программное обеспечение, которое называется системой управления базами данных (СУБД). СУБД служит интерфейсом между базой данных и пользователями или программами, предоставляя пользователям возможность получать и обновлять информацию, а также управлять ее упорядочением и оптимизацией. СУБД также упрощает контроль и управление базами данных, позволяя выполнять различные административные операции, такие как мониторинг производительности, настройка, а также резервное копирование и восстановление.

В качестве примеров популярного программного обеспечения для управления базами данных, или СУБД, можно назвать MySQL, Microsoft Access, Microsoft SQL Server, FileMaker Pro, Oracle Database и dBASE.

**MySQL** — это реляционная система управления базами данных с открытым исходным кодом на основе языка SQL. Она была разработана и оптимизирована для веб-приложений и может работать на любой платформе. Так как с развитием Интернете появились новые требования, веб-разработчики предпочитают использовать для веб-приложений платформу MySQL. База данных MySQL предназначена для обработки миллионов запросов и тысяч транзакций, поэтому ее часто выбирают компании электронной коммерции, которым требуется управлять большим количеством денежных переводов. Гибкость по мере необходимости — основная характеристика MySQL.

MySQL — это реляционная система управления базами данных с открытым исходным кодом. В настоящее время эта СУБД одна из наиболее популярных в веб-приложениях — подавляющее большинство CMS использует именно MySQL (часто только её, без альтернатив), а почти все веб-фреймворки поддерживают MySQL уже на уровне базовой конфигурации (без дополнительных модулей).

Из преимуществ СУБД MySQL стоит отметить простоту использования, гибкость, низкую стоимость владения (относительно платных СУБД), а также масштабируемость и производительность.

MySQL позволяет хранить целочисленные значения со знаком и беззнаковые, длиной в 1, 2, 3, 4 и 8 байтов, работает со строковыми и текстовыми данными фиксированной и переменной длины, позволяет осуществлять SQL-команды SELECT, DELETE, INSERT, REPLACE и UPDATE, обеспечивает полную поддержку операторов и функций в SELECT- и WHERE- частях запросов, работает с GROUP BY и ORDER BY, поддерживает групповые функции COUNT(), AVG(), STD(), SUM(), MAX() и MIN(), позволяет использовать JOIN в запросах, в т.ч. LEFT OUTER JOIN и RIGHT OUTER JOIN, поддерживает репликацию, транзакции, работу с внешними ключами и каскадные изменения на их основе, а также обеспечивает многие другие функциональные возможности.

Гибкость СУБД MySQL обеспечивается поддержкой большого количества типов таблиц: пользователи могут выбрать как таблицы типа MyISAM, поддерживающие полнотекстовый поиск, так и таблицы InnoDB, поддерживающие транзакции на уровне отдельных записей. Есть и другие типы таблиц, разработанные сообществом.

СУБД MySQL появилась в 1995. Написана на C и C++, протестирована на множестве различных компиляторов и работает на различных платформах. C 2010 года разработку и поддержку MySQL осуществляет корпорация Oracle. Продукт распространяется как под GNU GPL, так и под собственной коммерческой лицензией. Однако по условиям GPL, если какая-либо программа включает исходные коды MySQL, то и эта программа тоже должна распространяться по лицензии GPL. Для нежелающих открывать исходные тексты своих программ как раз предусмотрена коммерческая лицензия, которая, в дополнение к возможности разработки под «закрытой» лицензией, обеспечивает качественную сервисную поддержку. Сообществом разработчиков MySQL созданы различные ответвления — Drizzle, OurDelta, Percona Server и MariaDB, все эти ответвления уже существовали на момент получения прав на MySQL корпорацией Oracle.

Многие ведущие веб-сайты и веб-приложения используют СУБД MySQL, в том числе Airbnb, Uber, LinkedIn, Facebook, Twitter и YouTube.

Базы данных — это инструмент для сбора и структурирования информации. В базе могут храниться данные о людях, товарах, заказах и о многом другом. Многие базы данных изначально представляют собой небольшой список в текстовом редакторе или электронной таблице. По мере увеличения объема данных в списке постепенно появляются несоответствия и излишняя информация. Информация, отображенная в виде списка, становится непонятной. Кроме того, ограничены способы, с помощью которых можно искать и отображать подмножества данных. Как только начинают появляться эти проблемы, мы рекомендуем перенести всю информацию в базу данных, созданную в системе управления базами данных (СУБД), такой как Access.

Компьютерная база данных — это хранилище объектов. В одной базе данных может быть больше одной таблицы. Например, система отслеживания складских запасов, в которой используются три таблицы, — это не три базы данных, а одна. В **базе данных Access** (если ее специально не настраивали для работы с данными или кодом, принадлежащими другому источнику) все таблицы хранятся в одном файле вместе с другими объектами, такими как формы, отчеты, макросы и модули. Для файлов баз данных, созданных в формате Access 2007 (который также используется в Access 2016, Access 2013 и Access 2010), используется расширение ACCDB, а для баз данных, созданных в более ранних версиях Access, — MDB. С помощью Access 2016, Access 2013, Access 2010 и Access 2007 можно создавать файлы в форматах более ранних версий приложения (например, Access 2000 и Access 2002–2003).

Использование Access позволяет:

* добавлять новую информацию в базу данных, например новый артикул складских запасов;
* изменять информацию, уже находящуюся в базе, например перемещать артикул;
* удалять информацию, например если артикул был продан или утилизирован;
* упорядочивать и просматривать данные различными способами;
* обмениваться данными с другими людьми с помощью отчетов, сообщений электронной почты, внутренней сети или Интернета.

Ниже приведены краткие описания элементов стандартной базы данных Access.

* Таблица базы данных похожа на электронную таблицу — и там, и там информация расположена в строках и столбцах. Поэтому импортировать электронную таблицу в таблицу базы данных обычно довольно легко. Основное различие заключается в том, как данные структурированы. Чтобы база данных была как можно более гибкой и чтобы в ней не появлялось излишней информации, данные должны быть структурированы в виде таблиц. Например, если речь идет о таблице с информацией о сотрудниках компании, больше одного раза вводить данные об одном и том же сотруднике не нужно. Данные о товарах должны храниться в отдельной таблице, как и данные о филиалах компании. Этот процесс называется нормализацией.
* Строки в таблице называются записями. В записи содержатся блоки информации. Каждая запись состоит по крайней мере из одного поля. Поля соответствуют столбцам в таблице. Например, в таблице под названием "Сотрудники" в каждой записи находится информация об одном сотруднике, а в каждом поле — отдельная категория информации, например имя, фамилия, адрес и т. д. Поля выделяются под определенные типы данных, например текстовые, цифровые или иные данные. Записи и поля можно описать по-другому. Представьте старый библиотечный карточный каталог. Каждой карточке в шкафу соответствует запись в базе данных. Блоки информации на карточке (автор, название книги и т. д.) соответствуют полям в базе данных.
* С помощью форм создается пользовательский интерфейс для ввода и редактирования данных. Формы часто содержат кнопки команд и другие элементы управления, предназначенные для выполнения различных функций. Можно создать базу данных, не используя формы, если просто отредактировать уже имеющуюся информацию в таблицах Access. Тем не менее, большинство пользователей предпочитает использовать формы для просмотра, ввода и редактирования информации в таблицах.
* С помощью кнопок команд задаются данные, которые должны появляться в форме, открываются прочие формы и отчеты и выполняется ряд других задач. Например, есть "Форма клиента", в которой вы работаете с данными о клиентах. И в ней может быть кнопка, нажатием которой открывается форма заказа, с помощью которой вы вносите информацию о заказе, сделанном определенным клиентом.
* Формы также дают возможность контролировать взаимодействие пользователей с информацией базы данных. Например, можно создать форму, в которой отображаются только определенные поля и с помощью которой можно выполнять только ограниченное число операций. Таким образом обеспечивается защита и корректный ввод данных.
* Отчеты используются для форматирования, сведения и показа данных. Обычно отчет позволяет найти ответ на определенный вопрос, например "Какую прибыль в этом году принесли нам наши клиенты?" или "В каких городах живут наши клиенты?" Отчеты можно форматировать таким образом, чтобы информация отображалась в наиболее читабельном виде. Отчет можно сформировать в любое время, и в нем всегда будет отображена текущая информация базы данных. Отчеты обычно форматируются таким образом, чтобы их можно было распечатать, но их также можно просматривать на экране, экспортировать в другие программы или вкладывать в сообщения электронной почты.
* Запросы могут выполнять множество функций в базе данных. Одна из их основных функций — находить информацию в таблицах. Нужная информация обычно содержится в нескольких таблицах, но, если использовать запросы, ее можно просматривать в одной. Кроме того, запросы дают возможность фильтровать данные (для этого задаются критерии поиска), чтобы отображались только нужные записи. Используются и так называемые "обновляемые" запросы, которые дают возможность редактировать данные, найденные в основных таблицах. При работе с обновляемым запросом помните, что правки вносятся в основные таблицы, а не только в таблицу запроса. Есть два основных вида запросов: запросы на выборку и на изменение. Запрос на выборку только находит данные и предоставляет к ним доступ. Результаты такого запроса можно просмотреть на экране, распечатать или скопировать в буфер обмена, а также использовать в качестве источника записей для формы или отчета. С помощью запроса на изменение, как видно из названия, можно выполнять определенные операции с найденными данными: создавать таблицы, добавлять информацию в уже существующие таблицы, а также обновлять или удалять данные.
* Макросы в Access — это нечто вроде упрощенного языка программирования, с помощью которого можно сделать базу данных более функциональной. Например, если к кнопке команды в форме добавить макрос, то он будет запускаться всякий раз при нажатии этой кнопки. Макросы состоят из команд, с помощью которых выполняются определенные задачи: открываются отчеты, выполняются запросы, закрывается база данных и т. д. Используя макросы, можно автоматизировать большинство операций, которые в базе данных вы делаете вручную, и, таким образом, значительно сэкономить время.
* Подобно макросам, модули — это объекты, с помощью которых базу данных можно сделать более функциональной. Но если макросы в Access составляются путем выбора из списка макрокоманд, модули создаются на языке Visual Basic для приложений (VBA). Модули представляют собой наборы описаний, инструкций и процедур. Существуют модули класса и стандартные модули. Модули класса связаны с конкретными формами или отчетами и обычно включают в себя процедуры, которые работают только с этими формами или отчетами. В стандартных модулях содержатся общие процедуры, не связанные ни с каким объектом.

## Формирование требований к разрабатываемому ПС

Создать в среде C++ приложение наподобие базы данных, предназначенное для учета клиентов интернет-магазина. Реализовать автоматическое заполнение полей дата и время посещения при вводе новых клиентов.

Входные данные:

* типизированный файл с информацией о клиентах;
* типизированный файл с информацией о клиентах («черный» список и другие подобные списки);

Выходные данные:

* типизированные файлы с информацией;
* вывести на экран, а также в текстовый файл списки клиентов, предпочтения клиентов (наиболее часто просматриваемые товары);
* построить график посещаемости магазина за период времени (месяц, год и т.д.)

Условия написания:

Программа должна формировать динамические списки клиентов с возможностью добавления новых и удаления имеющихся, а также с возможностью просмотра списков. Записывать информацию в файл и загружать из файла, сортировать списки по выбранному полю. Формировать «черный» список клиентов; список клиентов, имеющих скидочную карту магазина; на основе предпочтений клиентов, выводить наиболее часто просматриваемые товары; список самых активных покупателей магазина.

Поля списка клиентов – ФИО, паспортные данные, особенности работы с данным клиентом (наличие его в определённом списке), дата регистрации на сайте магазина, наиболее часто просматриваемые данным клиентом товары.

1. **РАЗРАБОТКА СТРУКТУРЫ ДАННЫХ**

## 2.1 Описание используемых структур данных

Для реализации поставленной задачи, программа должна выполнять несколько основных функций, реализуемых с помощью следующих структур данных.

**2.1.1 Связные списки**

Хранение данных в массивах имеет ряд недостатков. В неупорядоченном массиве поиск выполняется относительно медленно, тогда как в упорядоченном массиве медленно выполняется вставка. Удаление выполняется медленно в обеих разновидностях массивов. Кроме того, размер массива невозможно изменить после его создания.

Связанные списки — структура данных, которая решает некоторые из этих проблем. Гибкость связанных списков хорошо подходит для многих общих задач хранения данных. Кроме того, связанный список может заменить массив в качестве базы для других структур хранения данных (таких, как стеки и очереди). Более того, связанные списки часто могут использоваться вместо массивов (исключение составляют ситуации с частым произвольным доступом к отдельным элементам по индексу). Связанные списки не решают всех проблем хранения данных, но они на редкость универсальны, а на концептуальном уровне более просты, чем другие структуры данных (например, деревья).

**Строение связного списка**

В связанном списке каждый элемент данных встраивается в специальный объект, называемый элементом списка. Так как список содержит много однотипных элементов, для них удобно создать отдельный класс, отличный от класса самого связанного списка. Каждый элемент содержит ссылку на следующий элемент списка; поле, в котором эта ссылка хранится, обычно называется next. Объект списка содержит ссылку на первый элемент first. Отношения между объектами в этой архитектуре представлены на рис. 2.

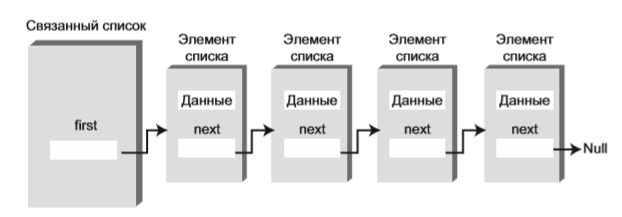


Рис. 2 Связный список

Подобные определения классов иногда называются самоотносимыми (self-referential), потому что объект класса содержит поле (next в данном случае) со ссылкой на объект того же типа.

Метод insertFirst() класса LinkList вставляет новый элемент в начало списка.

В этой позиции вставка выполняется проще всего, потому что first уже указывает на первый элемент. Чтобы вставить в список новый элемент, достаточно присвоить полю next созданного объекта ссылку на предыдущий первый элемент, а затем из-менить поле first так, чтобы оно указывало на только что вставленный элемент. Ситуация показана на рис. 2.1.

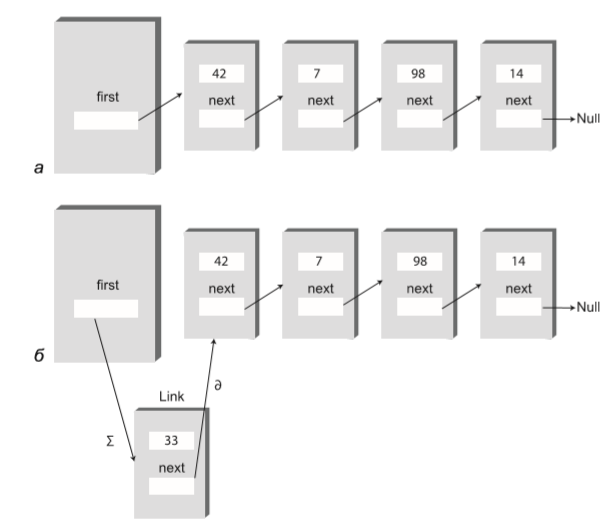


Рис. 2.1. Вставка элемента: а- до вставки; б-после вставки

Метод deleteFirst() является противоположностью insertFirst(). Он отсоединяет первый элемент, для чего в поле first заносится ссылка на второй элемент (который находится по значению поля next в первом элементе). На рис. 2.2 показано, как изменяется значение first при удалении объекта.

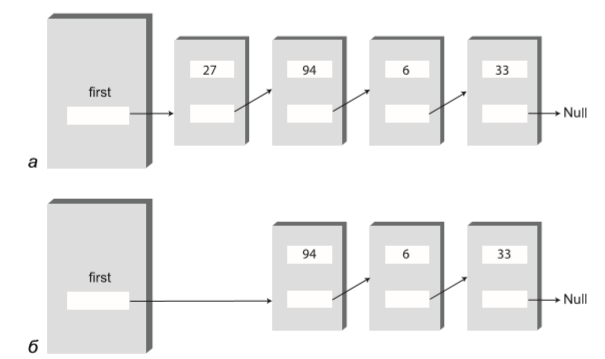


Рис. 2.2 Удаление элемента: а- до удаления; б- после удаления

**2.1.2 Двусвязные списки**

Рассмотрим следующую разновидность связанных списков: **двусвязный список.**

**В разрабатываемой программе на языке C ++ будет использована структура List. Это структура данных, которая построена на двусвязных списках.**

Потенциальным недостатком обычных связанных списков является сложность перемещения по списку в обратном направлении. Команда вида «current=current.next» позволяет легко перейти к следующему элементу списка, но соответствующего способа перехода к предыдущему элементу не существует. В некоторых ситуациях это ограничение создает проблемы. Допустим, в текстовом редакторе связанный список используется для хранения текста. Каждая строка на экране хранится в виде объекта String, встроенного в элемент списка. Когда пользователь перемещает курсор вниз по экрану, программа переходит к следующему элементу списка для обработки или вывода следующей строки. Но что произойдет, если пользователь переместит курсор вверх? При использовании обычного связанного списка придется вернуть переменную current (или ее аналог) к началу списка, а затем перебрать все элементы до предыдущего элемента. Конечно, такой способ неэффективен — перемещение к предыдущей строке должно осуществляться за один шаг.

Двусвязный список предоставляет такую возможность. Он позволяет перемещаться по списку как в прямом, так и в обратном направлении. Дело в том, что каждый элемент хранит ссылки на два других элемента вместо одного. Первая ссылка указывает на следующий элемент, как и в обычных списках. Вторая ссылка указывает на предыдущий элемент. Структура такого списка изображена на рис. 2.1

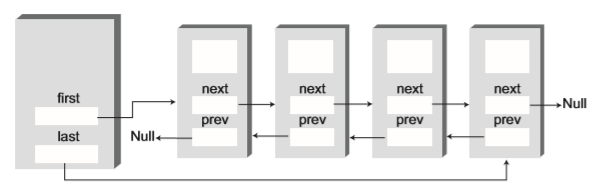


Рис. 2.3. Двусвязный список

К недостаткам двусвязных списков следует отнести то, что при каждой вставке или удалении ссылки вам приходится изменять четыре ссылки вместо двух: две связи с предыдущим элементом и две связи со следующим элементом. И конечно, каждый элемент списка занимает чуть больше места из-за дополнительной ссылки.

## Вставка элемента в двусвязный список

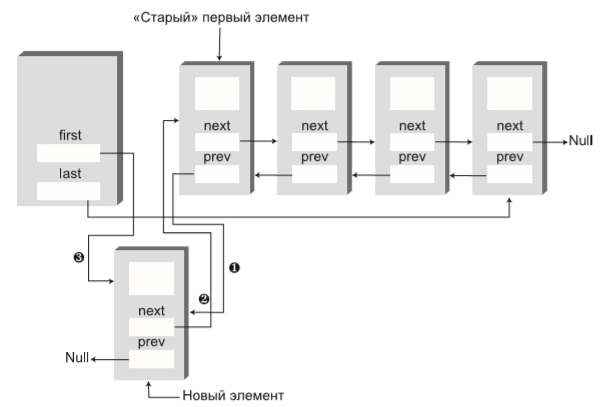


Рис. 2.4. Вставка в начале

## Класс двусвязного списка содержит несколько методов вставки. Метод insertFirst() вставляет новый элемент в начале списка, метод insertLast() — в конце, а метод insert() — по заданному ключу. Убедившись в том, что список не пуст, метод insertFirst() записывает ссылку на новый элемент в поле previous «старого» первого элемента, а ссылку на «старый» первый элемент — в поле next нового элемента. Наконец, в поле first заносится ссылка на новый элемент.

Метод insertLast() выполняет те же действия в конце списка; он является своего рода «зеркальным отражением» insertFirst(). Метод insertAfter() вставляет новый элемент после элемента с заданным ключом. Операция несколько усложняется, потому что в этой ситуации необходимо изменить четыре ссылки. Прежде всего следует найти элемент с заданным ключом, затем, если позиция вставки находится не в конце списка, необходимо создать две связи между новым и следующим элементом, и еще две — между current и новым элементом.

Метод insertAfter() вставляет новый элемент после элемента с заданным клю-чом. Операция несколько усложняется, потому что в этой ситуации необходимо изменить четыре ссылки. Прежде всего следует найти элемент с заданным ключом. Затем, если позиция вставки находится не в конце списка, необходимо создать две связи между новым и следующим элементом, и еще две — между current и новым элементом. Процесс показан на рис. 2.3

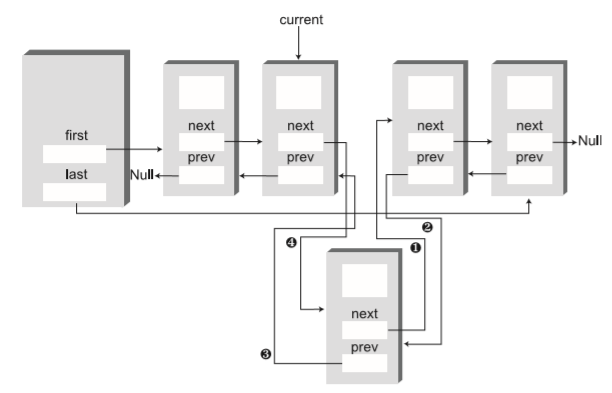


Рис. 2.5. Вставка в произвольной позиции

Если новый элемент должен вставляться в конце списка, то его поле next должно содержать null, а поле last — ссылку на новый элемент.

**Удаление**

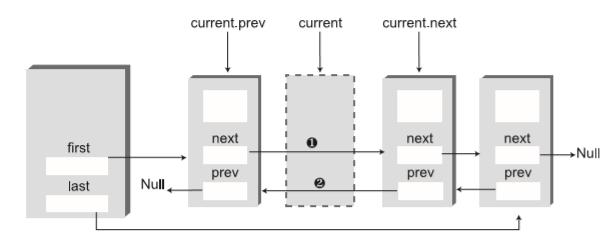
****

Рис. 2.6. Удаление произвольного элемента

Класс также содержит три метода удаления: deleteFirst(), deleteLast() и delete(). Первые два относительно тривиальны; в методе delete() удаляется элемент current. Если удаляемый элемент не является ни первым, ни последним в списке, то в поле next элемента current.getPrevious() (элемент, предшествующий удаляемому) заносится ссылка на current.getNext() (элемент, следующий после удаляемого), а в поле previous элемента current.getNext() заносится ссылка на current.getPrevious(). В результате элемент current исключается из списка.

Удаление реализуется следующими двумя командами: current.getPrevious().getNext()=current.getNext(); current.getNext().getPrevious() = current.getPrevious().

Если удаляемый элемент находится в первой или последней позиции списка, это особый случай, потому что ссылка на следующий или предыдущий элемент должна быть сохранена в поле first или last.

**3 РАЗРАБОТКА ПРОГРАММНОГО СРЕДСТВА**

**3.1 Схема программы**

Схема работы программы изображена на рис. 3.1(а-ж)

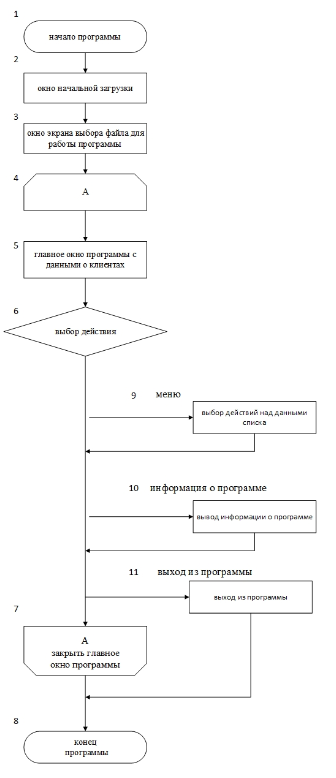


Рис. 3.1(а) Схема работы программы

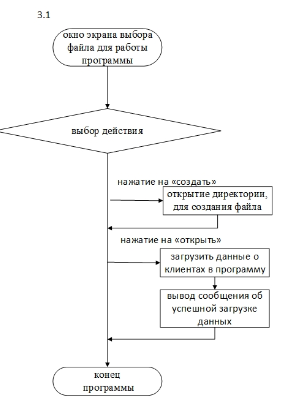


Рис.3.1(б)

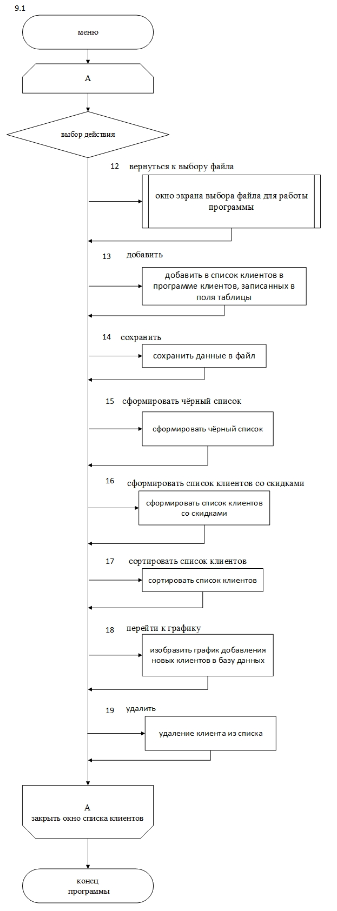


Рис. 3.1(в)

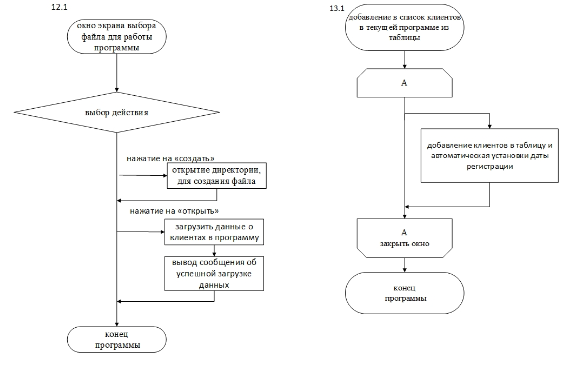


Рис. 3.1(г)

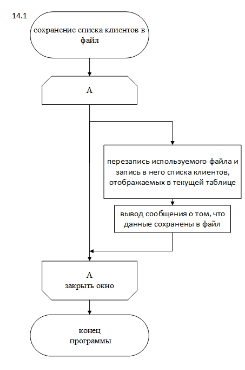


Рис. 3.1(д)

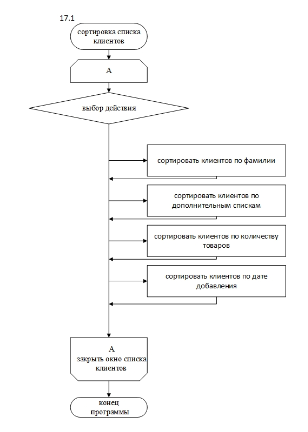


Рис. 3.1(е)

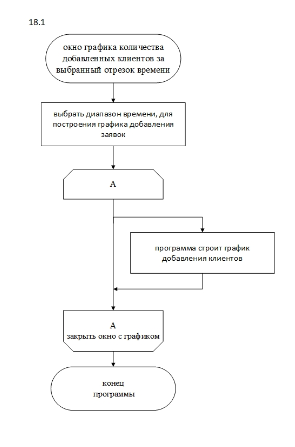


Рис. 3.1(ж)

* 1. **Проектирование окна «Выбор файла, используемого программой»**

При запуске программа сразу предлагает пользователю выбрать необходимый ему файл рис. 3.2.1

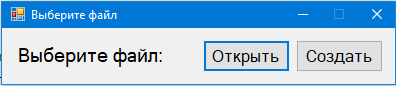


Рис. 3.2.1 окно «Выбор файла, с которым будет работать программы»

Разница между кнопками «Открыть» и «Создать» только в том, что программа будет либо загружать все данные, которые есть в некотором файле, либо не загружать, а создавать новый файл. Если мы нажмём на кнопку «Создать» мы можем либо создать новый документ, либо выбрать уже существующий, в котором что-то есть, но тем не менее программа из него ничего не загрузит. Таким образом данные будут либо перезаписаны, либо записаны по новой.

Программа в данный момент времени видит только текстовые документы (т.к. стоит специальный фильтр, что мы можем работать только с текстовыми документами).

Если данные были успешно загружены программа выводит соответствующее сообщение (Рис. 3.2.2)

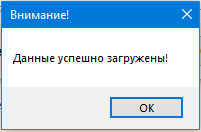
****

Рис. 3.2.2 вывод сообщения об успешной загрузке данных

**3.3 Проектирование окна списка клиентов**

Окно списка клиентов сети должно в первую очередь отображать список клиентов, что очень удобно сделать с помощью соответствующей таблицы. Также у пользователя есть возможность добавить клиента и удалить и т.д. Когда пользователь выполняет некоторые манипуляции со списком клиентов, соответственно изменённый список будет отображаться на экране после нажатия кнопки «Добавить», которая находится в разделе закреплённого вверху экрана меню. Также, пользователя есть возможность редактировать информацию о клиентах и после записывать её в файл, также сортировать списки и затем формировать списки. рис. 3.3, рис. 3.3.1



Рис. 3.3 Окно списка клиентов

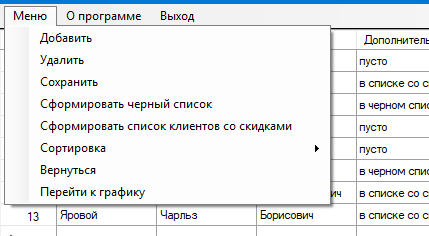


Рис. 3.3.1 меню

Если данные пользователя корректно приняты программой, то ему будет выведено соответствующее сообщение. Пример рис. 3.3.2

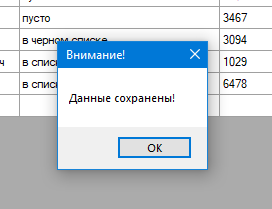
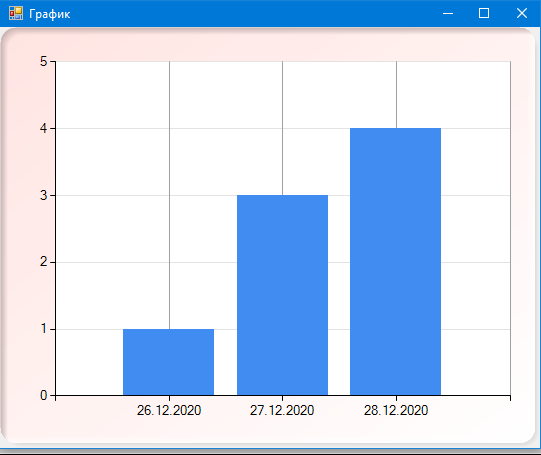


Рис. 3.3.2 Сообщение об успешном сохранении данных

**3.5 Проектирование окна статистики добавленных клиентов**

Программное средство также может отображать на экран статистику количества добавленных в базу клиентов за некоторый (любой, выбранный пользователем) диапазон времени рис. 3.5



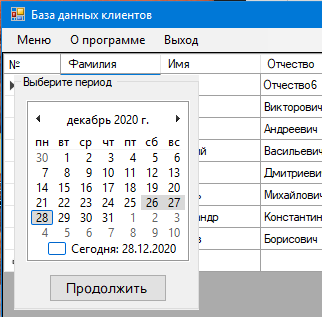


Рис. 3.5 Статистика добавленных в базу клиентов

1. **ОБОСНОВАНИЕ ТЕХНИЧЕСКИХ ПРИЕМОВ**

**ПРОГРАММИРОВАНИЯ**

Данная программа состоит из следующих модулей:

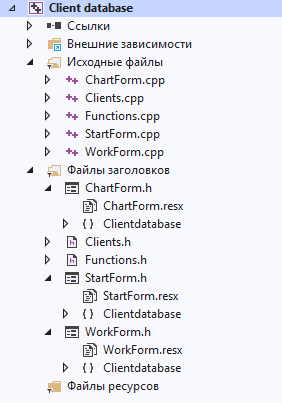


Рис. 4 Файлы программы

Созданная программа включает в себя 11 различных модулей:

* ChartForm.cpp

Эта форма предоставляет возможность просмотреть статистику за указанный промежуток времени. На форме расположены имеется поле, с помощью которых можно выбрать нужный промежуток времени для просмотра статистики. На данном окне есть график, который позволяет пользователю увидеть количество поступивших заявок.

* Clients.cpp

Модуль, отвечающий чтение данных клиентов из текстовых документов и также запись в них. Также считывание товаров, проверку индексов, для проверки на соответствие клиента и списка, в котором он находится. А также там имеются методы для сортировки данных клиентов.

* Functions.cpp

В данном файле находятся некоторые реализации функций конвертации, тк в программе я работаю с с++, но из-за необходимости использовать CLR (windowsForms) некоторые стандартные методы работают не совсем корректно, поэтому необходимы функции преобразования некоторые типов данных. Также там есть парсер даты.

* StartForm.cpp

Эта форма отвечает за выбор файла, с которым программа будет работать. Это форма открывается самой первой в программе.

* WorkForm.cpp

Модуль, отвечающий за вывод на экран списка всех заявок. На фоне находится таблица, отвечающая за вывод списка, и также кнопки для удаления из списка и для обработки клиента. Также на форме имеются разные кнопки, одна из которых отвечает за сохранение клиентов в файл, другая за вывод клиентов на экран.

* ChartForm.h

Эта форма соответственно является заголовочным файлом, используемым при работе с окном графика (статистики) добавленных клиентов за выбранный промежуток времени.

* Clients.h

Этот заголовочный файл содержит структуру клиента и также класс клиента

* Functions.h

Содержит структуру даты и прототипы некоторых функций.

* StartForm.h
* WorkForm.h

Две формы соответственно отвечающие за работу с выбором файла для программы и за сам основной список клиентов.

**5** **ТЕСТИРОВАНИЕ**

**5.1 Результаты тестирования программы**

Таблица 1 – Тест-кейс наличия главного меню

|  |  |  |
| --- | --- | --- |
| Тест[1] | Ожидаемый результат | Результат |
| **Открытие окна для выбора файла, с которым будет работать программа**   1. Запустить программу 2. Проверить элементы первого окна | 1. При запуске программы попадаем на окно выбора вариантов работы с файлом. 2. Отображаются следующие элементы:   - кнопка «Открыть», кнопка для создания нового файла «Создать» и текст Выберите файл. | Успех |

Таблица 2 – Тест-кейс работы кнопок главного меню

|  |  |  |
| --- | --- | --- |
| Тест[2] | Ожидаемый результат | Результат |
| **Работа с кнопками окна выбора файла**   1. Запуск программы 2. Нажать на кнопку «Открыть» | 1. Открывается окно выбора работы с файлом. 2. Кнопки активны и кликабельны во всей области. При нажатии происходит переход на соответствующее окно приложения. При наведении мыши на кнопки, у них появляется эффект «тени». | Успех |

Таблица 3 – Тест-кейс отображения окна «Список клиентов»

|  |  |  |
| --- | --- | --- |
| Тест[3] | Ожидаемый результат | Результат |
| **Отображение окна «Список клиентов»**   1. Открыть окно «Список клиентов» 2. Проверить элементы окна | 1. Перейти на окно «Список клиентов». 2. Отображаются следующие элементы:  * кнопка «меню»; * кнопка «выход»; * кнопка «информация о программе»; * таблица со списком всех клиентов. | Успех |

Таблица 4 – Тест-кейс отображения окна «Добавить клиентов»

|  |  |  |
| --- | --- | --- |
| Тест[4] | Ожидаемый результат | Результат |
| **Отображение меню программы**   1. Открыть окно «список клиентов» 2. Проверить элементы окна   Проверить работу меню программы на примере сортировки списка | 1. Перейти на окно «список клиентов». 2. Отображаются следующие сообщения:  * если сортировка работает корректно, то можем видеть отсортированный список. | Успех |

Таблица 5 – Тест-кейс контента окна «Новая заявка»

|  |  |  |
| --- | --- | --- |
| Тест[5] | Ожидаемый результат | Результат |
| **Составление чёрного списка клиентов**   1. Перейти на окно «Список клиентов» 2. Проверить элементы окна 3. Проверка корректности оставление списка | 1. Открывается окно «Список клиентов».  * Если программа работает корректно, то в файле списка клиентов, будут клиенты из чёрного списка  1. Отображаются следующие элементы:  * если список корректно составлен, то можем просмотреть его в текстовом документе | Успех |

Таблица 6 – Тест-кейс контента окна «Статистика поступления заявок»

|  |  |  |
| --- | --- | --- |
| Тест[6] | Ожидаемый результат | Результат |
| **Отображение контента окна «Статистика поступления заявок»**   1. Перейти на окно «Статистика» 2. Проверить элементы окна | 1. Открывается окно «Статистика поступления заявок». 2. Отображается график статистики | Успех |

Таблица 7 – Тест-кейс сохранения данных о клиенте в файл

|  |  |  |
| --- | --- | --- |
| Тест[7] | Ожидаемый результат | Результат |
| **Сохранение клиента в файл**   1. Открыть окно «Добавить клиента» 2. Ввести данные 3. Нажать кнопку «добавить клиента» 4. Проверить, сохранился ли клиент в файл 5. Проверить, отображается ли он в списке клиентов | 1. Открывается окно «Добавить клиента». 2. Элементы окна кликабельны и работают корректно. 3. Клиент записывается в список клиентов. 4. В файле клиент находится последним в списке. 5. В окне «список клиентов» последним отображается добавленный клиент. | Успех |

Таблица 8 – Тест-кейс сохранения добавленной заявки в файл

|  |  |  |
| --- | --- | --- |
| Тест[8] | Ожидаемый результат | Результат |
| **Сохранение в файл списка клиентов со скидками**   1. Открыть окно «Список клиентов» 2. Заполнить поля клиентов 3. Нажать кнопку «составить список клиентов со скидками» 4. Проверить, сохранилась ли запись | 1. Открывается окно «Добавить заявку». 2. В полях отображается введенная информация. 3. Заявка добавлена в случае корректного заполнения полей. 4. В файле с заявками сохранились данные о добавленной заявке. | Успех |

Таблица 9 – Тест-кейс открытие окна заявок, выполненных за текущие сутки

|  |  |  |
| --- | --- | --- |
| Тест[9] | Ожидаемый результат | Результат |
| **Удаление клиента из списка**   1. На окне списка клиентов нажать кнопку меню а после кнопку удалить | 1. Из таблицы будет удалён клиент, у которого выделена строка | Успех |

Таблица 10 – Тест-кейс корректной работы графиков

|  |  |  |
| --- | --- | --- |
| Тест[10] | Ожидаемый результат | Результат |
| **Корректная работа графиков**   1. Перейти на окно «статистика поступления заявок» 2. Поменять промежуток времени для выборки статистики 3. Сверить данные с хранящимися в файлах | 1. Открывается окно «Статистика поступления заявок». 2. Данные на графиках изменяются согласно промежутку времени. 3. Данные, отображенные на графиках, совпадают с расчетными по всем файлам. | Успех |

* 1. **Подтверждение корректности тестирования**

**Тест [1]- Рис. 5.1**

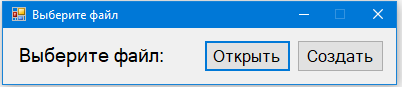
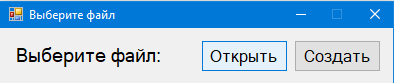


Рис. 5.1 Наличие начального окна и возможность работы с ним

**Тест [2]- Рис. 5.2**



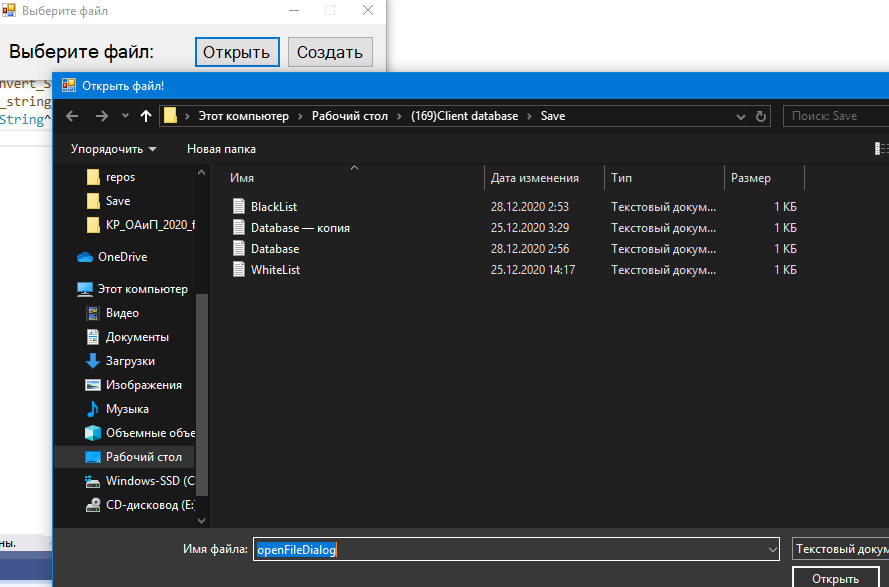


Рис. 5.2 Работа с кнопками начального окна

**Тест [3]- Рис. 5.3**

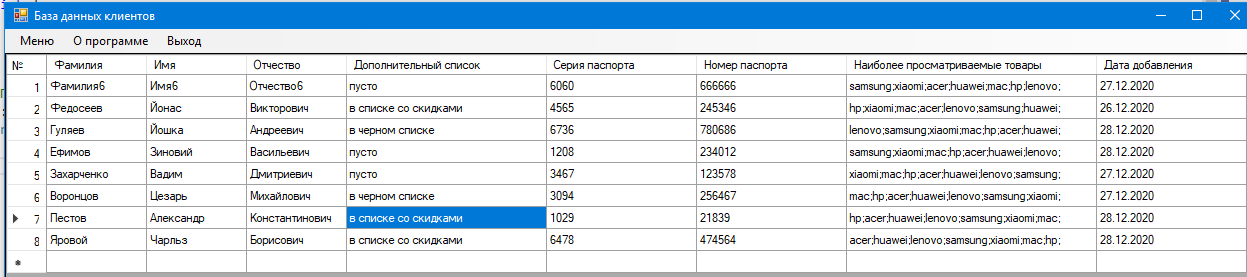
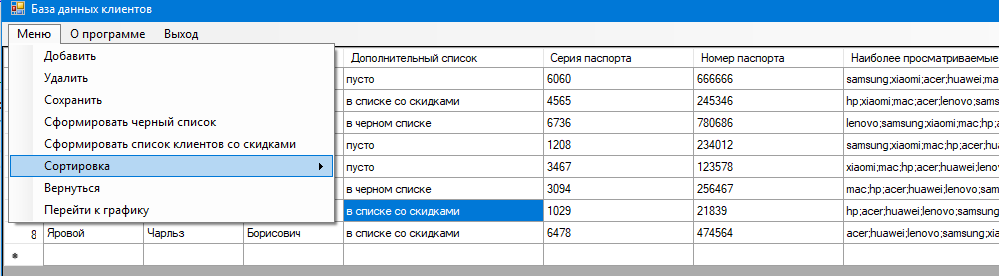


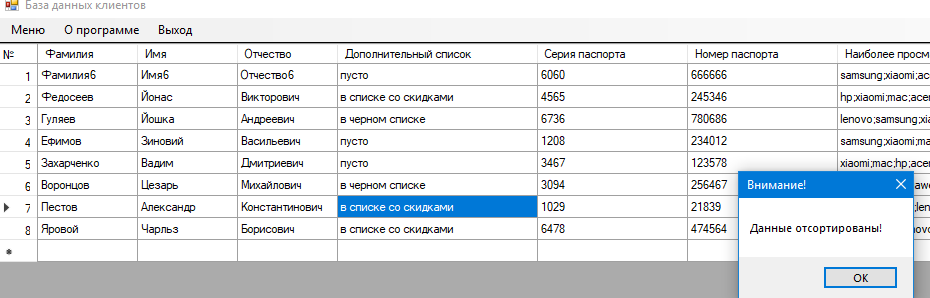
Рис. 5.3 Отображение окна «Список клиентов»

**Тест [4]- Рис. 5.4 ,**

Рис. 5.4 Отображение окна «Добавить клиента»

Рис. 5.4 (3-ий пункт) (а) Отображение окна «Добавить клиента»





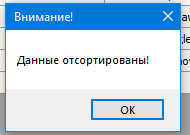
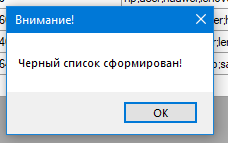


Рис. 5.4

**Тест [5]- Рис. 5.5**



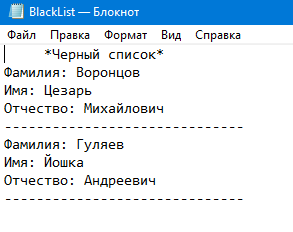
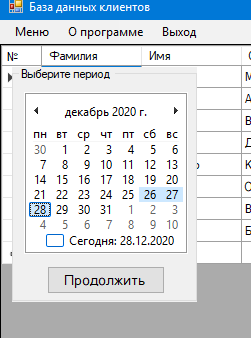


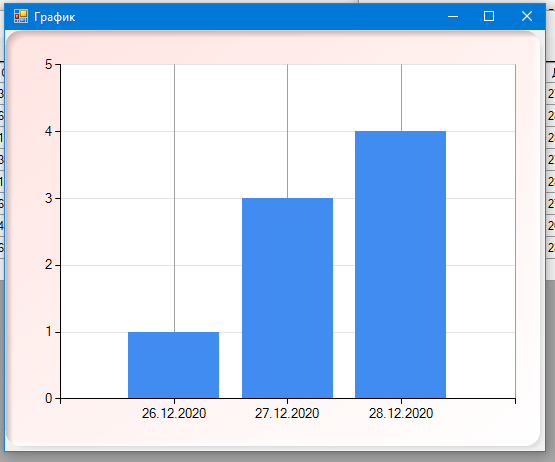
Рис. 5.5

**Тест [6]- Рис. 5.6**

Рис. 5.6 Отображение контента окна «Статистика»

**Тест [7]- Рис. 5.7.(1-4)**





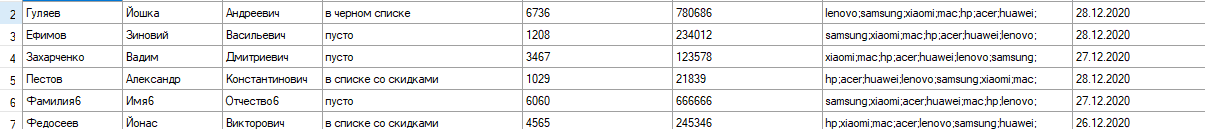
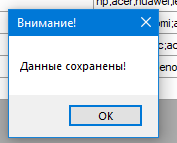


Рис. 5.7.3 Данные отображаются в таблице «Списка клиентов»



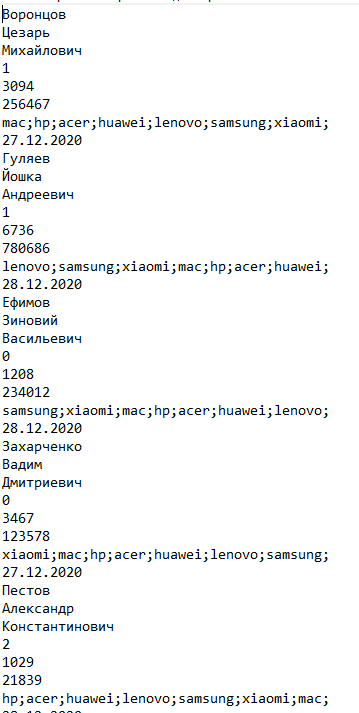


Рис. 5.7.4 Данные отображаются в файле

**Тест [8]- Рис. 5.8.**



Рис. 5.8.1 Клиент добавлен

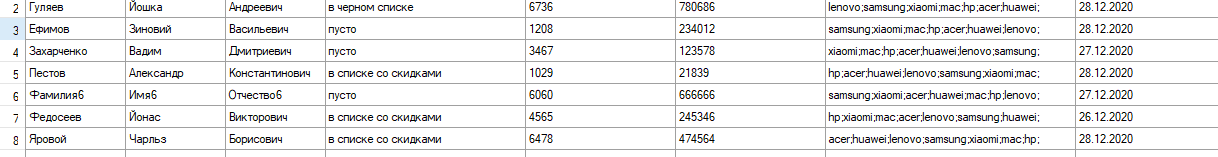
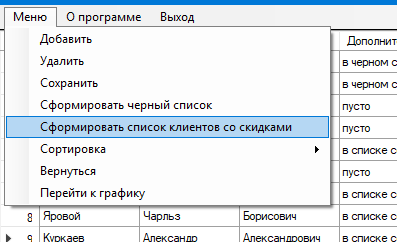
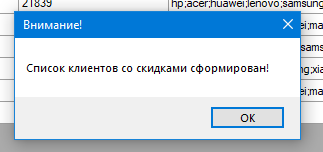
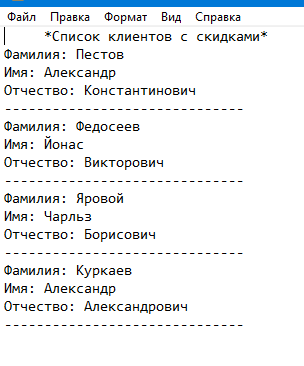


Рис. 5.8.2

****

****

****

**Тест [9]- Рис. 5.9**

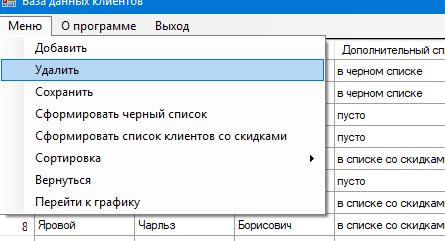
****

Рис. 5.9 Удаление

**Тест [10]- Рис. 5.10**

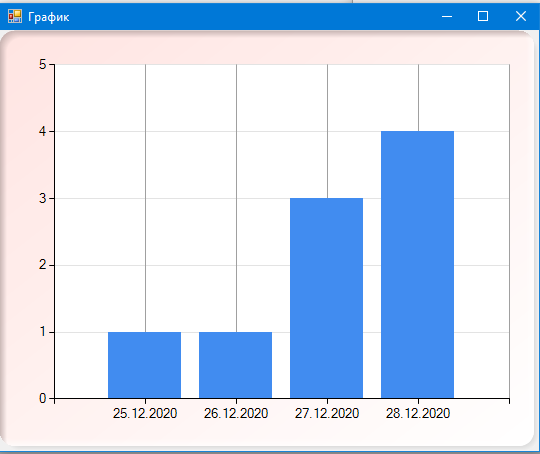


Рис. 5.10 Корректность отображаемой статистики поступления заявок

1. **РУКОВОДСТВО ПОЛЬЗОВАТЕЛЯ ПРОГРАММЫ**
2. Порядок добавления нового клиента интернет-магазина:

* Пользователю нужно сначала выбрать или создать в любой директории файл, для списка клиентов, с которым программа будет работать
* После этого, пользователю необходимо перейти на окно «Список клиентов»;
* Необходимо кликнуть кнопкой мыши на строку в таблице и корректно заполнить её;
* Необходимо заполнить все пустые поля корректными данными;
* После этого пользователю необходимо кликнуть кнопкой мыши на кнопку «Добавить»;
* Таким образом автоматически будет заполнена дата регистрации клиента и ему будут присвоены товары, которые он часто просматривает;
* Нажав сначала на строку с клиентом, а затем кнопку «Удалить клиента», можно удалить клиента из таблицы;
* Если все данные заполнены корректно, то система выдаст оповещение об успешной записи клиента в файл, при нажатии на кнопку «Добавить».

1. Порядок сортировки списка:

* В окне «Список клиентов» нужно нажать на поле меню;
* После этого нажать кнопку сортировка;
* Там пользователю будет предложены поля, на основании которых можно отсортировать список клиентов;
* После нажатия на выбранную кнопку в таблице данные отобразятся нужным образом в уже отсортированном виде

1. Порядок формирования особых («чёрный список», «список клиентов со скидками») списков клиентов:

* Перейти на окно «Список клиентов» и в данном окне отображается список всех заявок;
* В данном окне нужно нажать на кнопку меню, затем на кнопку на выбор, либо «Сформировать чёрный список», либо «Сформировать список клиентов со скидками»;
* Также, нажав сначала на строку с клиентом, а затем кнопку «Удалить клиента», можно удалить клиента из таблицы(DataGridView);
* После этого будет дан выбор, в каком файле нужно сформировать данный список;
* Выбрав текстовый файл, пользователь потом может открыть его и удостовериться, что в данном файле будет сформирован выбранный список.

1. Порядок просмотра статистики поступивших заявок за выбранный диапазон времени:

* Статистика будет отображаться в графика;
* Выбрать промежуток времени, за который необходимо вывести статистику поступивших заявок и, после, нажав на кнопку «Применить»;
* После этого на экран будет выведена статистика заявок, поступивших за выбранный диапазон времени. Статистика будет отображаться в виде графика.

1. Возможные ошибки:

* Когда программа предлагает выбрать файл для работы, пользователь может выбирать только текстовый файл (.txt) остальные файлы даже не будут видны;
* В списке товаров клиентов, если в текстовом файле оставить указатель мыши на последней строке (то есть оставить последнюю строку пустой, то товар, который был записан последним будет дублироваться);
* Если при открытии файла для работы на протяжении всей программы, не вывелось сообщение о том, что данные успешно загружены, то вероятна ошибка.
* Если при заполнении поля «тип списка» пользователь ввёл название списка, который не предусмотрен в программе, то программа автоматически присвоит ему значение «пусто» то есть клиент не находится ни в каком списке.

# ЗАКЛЮЧЕНИЕ

В результате выполнения курсового проекта **проведен анализ предметной области, исходя из которого разработано программное обеспечение для создания базы данных клиентов интернет-магазина.** Для этого был использован язык программирования C++.

В начале работы были изучены прототипы данного программного обеспечения, проведен анализ поставленной задачи. В соответствии с ними сформулированы требования к программному обеспечению.

На этапе разработки программного средства были изучены варианты использования программного обеспечения и в соответствии с этим реализован основной функционал системы.

На следующем этапе было дано обоснование приемов программирования, используемых в построении системы, а также был до конца реализован функционал программного обеспечения, включающий интерфейс системы и его дизайн.

На этапе тестирования была испытана работа системы в различных условиях и на различных наборах данных.

Далее было составлено руководство пользователя, в котором было перечислено: способы использования системы для того, чтобы система работала правильно и корректно.

В результате была построена система с помощью языка C++ согласно сформулированным на начальном этапе требованиям. В ней были использованы различные программные решения, что позволило нам детально спроектировать программу.

**СПИСОК ИСПОЛЬЗОВАННЫХ ИСТОЧНИКОВ**

[1] Library of Congress Cataloging-in-Publication Data Sedgewick, Robert, 1946 Algorithms in C++ / Robert Sedgewick.—3d ed.- 528 c.

[2] C++ Programming Robert Sedgewick - Princeton University Addison Wesley Professional Algorithms in C++, Parts 1A4: Fundamentals, Data Structure, Sorting, Searching, Third Edition – 752 с.

[3] Керниган Б., Ритчи Д. Язык программирования Си ДПер. с англ., 3-е изд., испр. — СПб.: "Невский Диалект", 2001. - 352 с.

[4] Library of Congress Cataloging-in-Publication Data Stroustrup,Bjarne The C++ Programming Language / Bjarne Stroustrup. — 3rd. ed. – 1022 с

[5] Эккель Б. Философия С++. Введение в стандартный С++. 2-е изд. – СПб.: Питер, 2004. – 672 с.

[6] Эккель Б., Эллисон Ч. Философия С++. Практическое программирование. – СПб: Питер, 2004. – 608 с.

[7] Домашняя страница документации и учебных ресурсов Майкрософт для разработчиков и технических специалистов [Электронный ресурс].

URL: <https://docs.microsoft.com/ru-ru/>

[8] Дистанционная подготовка по информатике [Электронный ресурс].

URL: https://informatics.mccme.ru/mod/statements/view3.php?chapterid=2783#1

[9] ГОСТ 19.701–90 (ИСО 5807–85) [Текст]. – Единая система программной документации: Сб. ГОСТов. - М.: Стандартинформ, 2005 с.

[10] Лафоре Р. Объектно-ориентированное программирование С++ — СПб.: Питер, 2013. —702 c.

[11] Интернет-ресурс для изучения программирования на С++ [Электронный ресурс]. – Режим доступа: https://ravesli.com/uroki-po-qt5/

[12] Бахтизин В. В., Оношко Д. Е. Языки программирования – Минск, 2015. – 106 с.

**ПРИЛОЖЕНИЕ**

**(обязательное)**

**Листинг кода**

**ChartForm.cpp**

#include "ChartForm.h"

void Сlientdatabase::ChartForm::AddPointString(String^ s)

{

pointsString.Add(s);

}

void Сlientdatabase::ChartForm::AddPointInt(int s)

{

pointsInt.Add(s);

}

System::Void Сlientdatabase::ChartForm::ChartForm\_Load(System::Object^ sender, System::EventArgs^ e)

{

//Общие свойства

Chart^ myChart = gcnew Chart();

myChart->Parent = this;

myChart->Dock = Windows::Forms::DockStyle::Fill;

myChart->Left = 10; myChart->Top = 10;

myChart->Width = (this->ClientSize.Width - 10);

myChart->Height = (this->ClientSize.Height - 20);

myChart->BringToFront(); //или chart1->Visible = false;

ChartArea^ myChartArea = gcnew ChartArea();

//Область в которой будет построен график (их может быть несколько)

myChartArea->Name = "myChartArea";

myChart->ChartAreas->Add(myChartArea);

//График (их может быть несколько)

Series^ mySeries1 = gcnew Series();

mySeries1->ChartType = SeriesChartType::Column;

mySeries1->ChartArea = "myChartArea";

myChart->Series->Add(mySeries1);

//Исходные данные для графика

array<int>^ y\_values = gcnew array<int>(pointsInt.Count);

for (int i = 0; i < pointsInt.Count; i++) {

y\_values[i] = pointsInt[i];

}

array <System::String^>^ x\_values = gcnew array<String^>(pointsString.Count);

for (int i = 0; i < pointsString.Count; i++) {

x\_values[i] = pointsString[i];

}

//Задаем точки на графике

mySeries1->Points->DataBindXY(x\_values, y\_values);

//фон градиентом

myChart->BackColor = System::Drawing::Color::MistyRose;

myChart->BackGradientStyle = GradientStyle::DiagonalLeft;

//границы в современном стиле

myChart->BorderSkin->SkinStyle = BorderSkinStyle::Sunken;

myChart->BorderSkin->PageColor = this->BackColor;

//линии сетки покажем разными цветами

myChartArea->AxisX->MajorGrid->LineColor = System::Drawing::SystemColors::ControlDark;

myChartArea->AxisY->MajorGrid->LineColor = System::Drawing::SystemColors::ControlLight;

}

**Clients.cpp**

#include "Clients.h"

Clients::Clients()

{

}

Clients::~Clients()

{

}

bool Clients::Load(std::string filename)

{

//Создаем поток для чтения

std::ifstream load(filename);

//Проверяем открылся ли файл

if(!load)

return false;

//Очищаем данные, если они были

data.clear();

//Временные данные

Client buf;

int num;

std::string name;

//Пока не конец файла

while (!load.eof()) {

load >> buf.surname;

load >> buf.name;

load >> buf.patronymic;

load >> buf.mode\_list;

load >> buf.series;

load >> buf.number;

load >> buf.name\_of\_the\_goods;

load >> buf.date;

//Добавляем данные

Add(buf);

}

load.close();

return true;

}

bool Clients::Save(std::string filename)

{

//Создаем поток для перезаписи в файл

std::ofstream save(filename, std::ios::out);

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

save << data[i].surname << std::endl;

save << data[i].name << std::endl;

save << data[i].patronymic << std::endl;

save << data[i].mode\_list << std::endl;

save << data[i].series << std::endl;

save << data[i].number << std::endl;

save << data[i].name\_of\_the\_goods << std::endl;

save << data[i].date;

if (i < data.size() - 1)

save << std::endl;

}

save.close();

return true;

}

void Clients::Add(Client c)

{

data.push\_back(c);

}

bool Clients::Del(int i)

{

//Проверка индекса

if(i < 0 || i>data.size()-1)

return false;

data.erase(data.begin() + i);

return true;

}

bool Clients::Set(int i, Client c)

{

//Проверка индекса

if (i < 0 || i>data.size() - 1)

return false;

data[i] = c;

return true;

}

Client Clients::Get(int i)

{

//Проверка индекса

if (i < 0 || i>data.size() - 1)

return Client();

return data[i];

}

bool Clients::CreateBlackList(std::string filename)

{

//Создаем поток для перезаписи в файл

std::ofstream save(filename, std::ios::out);

save << " \*Черный список\*" << std::endl;

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

if (data[i].mode\_list == 1) {

save << "Фамилия: " << data[i].surname << std::endl;

save << "Имя: " << data[i].name << std::endl;

save << "Отчество: " << data[i].patronymic << std::endl;

save << "------------------------------" << std::endl;

}

}

save.close();

return true;

}

bool Clients::CreateListWithDiscounts(std::string filename)

{

//Создаем поток для перезаписи в файл

std::ofstream save(filename, std::ios::out);

save << " \*Список клиентов с скидками\*" << std::endl;

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

if (data[i].mode\_list == 2) {

save << "Фамилия: " << data[i].surname << std::endl;

save << "Имя: " << data[i].name << std::endl;

save << "Отчество: " << data[i].patronymic << std::endl;

save << "------------------------------" << std::endl;

}

}

save.close();

return true;

}

bool Clients::CreateGoods(int i)

{

//Проверка индекса

if (i < 0 || i>data.size() - 1)

return false;

//Создаем поток для чтения

std::ifstream load("Goods.txt");

//Проверяем открылся ли файл

if (!load)

return false;

std::string name;

int num;

std::vector<std::string> goods;

//Считываем все товары из файла

while (!load.eof()) {

load >> name;

goods.push\_back(name);

}

//Перемешиваем массив

std::random\_shuffle(goods.begin(), goods.end());

//Случайно получаем количество товаров

srand(time(NULL));

num = 1 + rand() % goods.size();

data[i].name\_of\_the\_goods = "";

for (int q = 0; q < goods.size(); q++) {

data[i].name\_of\_the\_goods += goods[q] + ";";

}

load.close();

return true;

}

bool Clients::Sort(int mode)

{

//Проверяем выбранное поле для сортировки

switch (mode)

{

case 1://Фамилия

for (int i = 0; i < data.size() - 1; i++) {

for (int j = i + 1; j < data.size(); j++) {

//Условие сортировки

if (data[i].surname > data[j].surname) {

//Меняем местами

Swap(i, j);

}

}

}

break;

case 2://По дополнительным спискам

for (int i = 0; i < data.size() - 1; i++) {

for (int j = i + 1; j < data.size(); j++) {

//Условие сортировки

if (data[i].mode\_list > data[j].mode\_list) {

//Меняем местами

Swap(i, j);

}

}

}

break;

case 3://По количество товаров

for (int i = 0; i < data.size() - 1; i++) {

for (int j = i + 1; j < data.size(); j++) {

//Условие сортировки

if (data[i].name\_of\_the\_goods.size() > data[j].name\_of\_the\_goods.size()) {

//Меняем местами

Swap(i, j);

}

}

}

break;

case 4://По дате добавления

for (int i = 0; i < data.size() - 1; i++) {

for (int j = i + 1; j < data.size(); j++) {

//Условие сортировки

if (data[i].date > data[j].date) {

//Меняем местами

Swap(i, j);

}

}

}

break;

default:

return false;

}

return true;

}

void Clients::Swap(int i, int j)

{

Client buf;

buf = data[i];

data[i] = data[j];

data[j] = buf;

}

**Functions.cpp**

#include "Functions.h"

/\* Реализация функций \*/

//Конвертируем System::string ^ в std::string

std::string& Convert\_String\_to\_string(System::String^ s, std::string& os) {

using namespace Runtime::InteropServices;

const char\* chars =

(const char\*)(Marshal::StringToHGlobalAnsi(s)).ToPointer();

os = chars;

Marshal::FreeHGlobal(IntPtr((void\*)chars));

return os;

}

//Конвертируем std::string в System::string ^

System::String^ Convert\_string\_to\_String(std::string& os, System::String^ s) {

s = gcnew System::String(os.c\_str()); // конвертируем

return s;

}

Date ParseDate(String^ str)

{

Date date;

String^ buf = "";

int rank = 1;

for (int i = 0; i < str->Length; i++) {

if (str[i] != '.') {

buf += str[i];

}

else {

if(rank == 1)

date.day = Convert::ToInt32(buf);

else

date.month = Convert::ToInt32(buf);

buf = "";

rank++;

}

}

date.year = Convert::ToInt32(buf);

return date;

}

**StartForm.cpp**

#include "Functions.h"

using namespace System;

using namespace System::Windows::Forms;

[STAThreadAttribute]

void main(array<String^>^ args) {

Application::EnableVisualStyles();

Application::SetCompatibleTextRenderingDefault(false);

Сlientdatabase::StartForm form;

Application::Run(% form);

}

Сlientdatabase::StartForm::StartForm(void)

{

InitializeComponent();

//Задаем фильт

openFileDialog->Filter = "Текстовый документ| \*.txt";

}

System::Void Сlientdatabase::StartForm::buttonOpen\_Click(System::Object^ sender, System::EventArgs^ e)

{

openFileDialog->Title = "Открыть файл!";

if (openFileDialog->ShowDialog() != Windows::Forms::DialogResult::OK)

return;

String^ filename = openFileDialog->FileName;

//Создаем форму

WorkForm^ form = gcnew WorkForm();

form->filename = filename;

form->load = true;

form->Show();

this->Hide();

}

System::Void Сlientdatabase::StartForm::buttonCreate\_Click(System::Object^ sender, System::EventArgs^ e)

{

openFileDialog->Title = "Выбрать пустой файл!";

if (openFileDialog->ShowDialog() != Windows::Forms::DialogResult::OK)

return;

String^ filename = openFileDialog->FileName;

//Создаем форму

WorkForm^ form = gcnew WorkForm();

form->filename = filename;

form->load = false;

form->Show();

this->Hide();

}

**WorkForm.cpp**

**#include "Functions.h"**

**std::string \_filename;**

**Clients \_clients;**

**System::Void Сlientdatabase::WorkForm::WorkForm\_Load(System::Object^ sender, System::EventArgs^ e)**

**{**

**Convert\_String\_to\_string(filename, \_filename);**

**if (load) {**

**if (\_clients.Load(\_filename)) {**

**MessageBox::Show("Данные успешно загружены!","Внимание!");**

**OutputData();**

**}**

**else {**

**MessageBox::Show("Ошибка загрузки данных!", "Ошибка!");**

**}**

**}**

**//Задаем фильтр**

**openFileDialog->Filter = "Текстовый документ| \*.txt";**

**dataGridView->TopLeftHeaderCell->Value = "№";**

**}**

**System::Void Сlientdatabase::WorkForm::добавитьToolStripMenuItem\_Click\_1(System::Object^ sender, System::EventArgs^ e)**

**{**

**SetData();**

**}**

**System::Void Сlientdatabase::WorkForm::удалитьToolStripMenuItem\_Click(System::Object^ sender, System::EventArgs^ e)**

**{**

**//Проверяем выбранную строку**

**if (dataGridView->SelectedRows->Count != 1) {**

**MessageBox::Show("Выберите одну строку для удаления","Внимание!");**

**}**

**int index = dataGridView->CurrentRow->Index;**

**dataGridView->Rows->RemoveAt(index);**

**if (\_clients.Del(index))**

**MessageBox::Show("Данные удалены!","Внимание!");**

**}**

**System::Void Сlientdatabase::WorkForm::сохранитьToolStripMenuItem\_Click(System::Object^ sender, System::EventArgs^ e)**

**{**

**SetData();**

**if (\_clients.Save(\_filename)) {**

**MessageBox::Show("Данные сохранены!","Внимание!");**

**}**

**else {**

**MessageBox::Show("Ошибка сохранения данных!", "Ошибка!");**

**}**

**}**

**System::Void Сlientdatabase::WorkForm::сформироватьЧерныйСписокToolStripMenuItem\_Click(System::Object^ sender, System::EventArgs^ e)**

**{**

**openFileDialog->Title = "Выберите файл для формирования черного списка!";**

**if (openFileDialog->ShowDialog() != Windows::Forms::DialogResult::OK)**

**return;**

**SetData();**

**String^ file;**

**file = openFileDialog->FileName;**

**std::string \_file;**

**Convert\_String\_to\_string(file, \_file);**

**if (\_clients.CreateBlackList(\_file))**

**MessageBox::Show("Черный список сформирован!","Внимание!");**

**else**

**MessageBox::Show("Ошибка формирования черного списка!", "Ошибка!");**

**}**

**System::Void Сlientdatabase::WorkForm::сформироватьСписокКлиентовСоСкидкамиToolStripMenuItem\_Click(System::Object^ sender, System::EventArgs^ e)**

**{**

**openFileDialog->Title = "Выберите файл для формирования черного списка!";**

**if (openFileDialog->ShowDialog() != Windows::Forms::DialogResult::OK)**

**return;**

**SetData();**

**String^ file;**

**file = openFileDialog->FileName;**

**std::string \_file;**

**Convert\_String\_to\_string(file, \_file);**

**if (\_clients.CreateListWithDiscounts(\_file))**

**MessageBox::Show("Список клиентов со скидками сформирован!", "Внимание!");**

**else**

**MessageBox::Show("Ошибка формирования списка со скидками!", "Ошибка!");**

**}**

**System::Void Сlientdatabase::WorkForm::вернутьсяToolStripMenuItem\_Click(System::Object^ sender, System::EventArgs^ e)**

**{**

**if (MessageBox::Show("Продолжить?", "Внимание!", MessageBoxButtons::YesNo) == Windows::Forms::DialogResult::Yes) {**

**StartForm^ form = gcnew StartForm();**

**form->Show();**

**this->Hide();**

**}**

**}**

**System::Void Сlientdatabase::WorkForm::поФамилииToolStripMenuItem\_Click(System::Object^ sender, System::EventArgs^ e)**

**{**

**SetData();**

**if (\_clients.Sort(1)) {**

**MessageBox::Show("Данные отсортированы!","Внимание!");**

**OutputData();**

**}**

**else {**

**MessageBox::Show("Ошибка сортировки!", "Ошибка!");**

**}**

**}**

**System::Void Сlientdatabase::WorkForm::поДополнительнымСпискамToolStripMenuItem\_Click(System::Object^ sender, System::EventArgs^ e)**

**{**

**SetData();**

**if (\_clients.Sort(2)) {**

**MessageBox::Show("Данные отсортированы!", "Внимание!");**

**OutputData();**

**}**

**else {**

**MessageBox::Show("Ошибка сортировки!", "Ошибка!");**

**}**

**}**

**System::Void Сlientdatabase::WorkForm::поКоличествоТоваровToolStripMenuItem\_Click(System::Object^ sender, System::EventArgs^ e)**

**{**

**SetData();**

**if (\_clients.Sort(3)) {**

**MessageBox::Show("Данные отсортированы!", "Внимание!");**

**OutputData();**

**}**

**else {**

**MessageBox::Show("Ошибка сортировки!", "Ошибка!");**

**}**

**}**

**System::Void Сlientdatabase::WorkForm::поДатеДобавленияToolStripMenuItem\_Click(System::Object^ sender, System::EventArgs^ e)**

**{**

**SetData();**

**if (\_clients.Sort(4)) {**

**MessageBox::Show("Данные отсортированы!", "Внимание!");**

**OutputData();**

**}**

**else {**

**MessageBox::Show("Ошибка сортировки!", "Ошибка!");**

**}**

**}**

**void Сlientdatabase::WorkForm::SetData()**

**{**

**\_clients.Clear();**

**Client buf;**

**for (int i = 0; i < dataGridView->Rows->Count - 1; i++) {**

**if (dataGridView->Rows[i]->HeaderCell->Value == nullptr) {**

**dataGridView->Rows[i]->HeaderCell->Value = Convert::ToString(i + 1);**

**dataGridView->Rows[i]->Cells[7]->Value = GetDateString(DateTime::Now.ToString());**

**}**

**Convert\_String\_to\_string(dataGridView->Rows[i]->Cells[0]->Value->ToString(), buf.surname);**

**Convert\_String\_to\_string(dataGridView->Rows[i]->Cells[1]->Value->ToString(), buf.name);**

**Convert\_String\_to\_string(dataGridView->Rows[i]->Cells[2]->Value->ToString(), buf.patronymic);**

**if (dataGridView->Rows[i]->Cells[3]->Value->ToString() == "пусто")**

**buf.mode\_list = 0;**

**else if(dataGridView->Rows[i]->Cells[3]->Value->ToString() == "в черном списке")**

**buf.mode\_list = 1;**

**else if (dataGridView->Rows[i]->Cells[3]->Value->ToString() == "в списке со скидками")**

**buf.mode\_list = 2;**

**else {**

**buf.mode\_list = 0;**

**dataGridView->Rows[i]->Cells[3]->Value == "пусто";**

**}**

**buf.series = Convert::ToInt32(dataGridView->Rows[i]->Cells[4]->Value);**

**buf.number = Convert::ToInt32(dataGridView->Rows[i]->Cells[5]->Value);**

**if (dataGridView->Rows[i]->Cells[6]->Value != nullptr) {**

**Convert\_String\_to\_string(dataGridView->Rows[i]->Cells[6]->Value->ToString(), buf.name\_of\_the\_goods);**

**}**

**Convert\_String\_to\_string(dataGridView->Rows[i]->Cells[7]->Value->ToString(), buf.date);**

**\_clients.Add(buf);**

**if (dataGridView->Rows[i]->Cells[6]->Value == nullptr) {**

**\_clients.CreateGoods(i);**

**String^ str = Convert\_string\_to\_String(\_clients.Get(i).name\_of\_the\_goods, str);**

**dataGridView->Rows[i]->Cells[6]->Value = str;**

**}**

**}**

**}**

**void Сlientdatabase::WorkForm::OutputData()**

**{**

**dataGridView->Rows->Clear();**

**String^ str;**

**String^ str1;**

**String^ str2;**

**String^ str3;**

**String^ str4;**

**String^ str5;**

**for (int i = 0; i < \_clients.Size(); i++) {**

**str = Convert\_string\_to\_String(\_clients.Get(i).surname, str);**

**str1 = Convert\_string\_to\_String(\_clients.Get(i).name, str1);**

**str2 = Convert\_string\_to\_String(\_clients.Get(i).patronymic, str2);**

**str3 = Convert\_string\_to\_String(\_clients.Get(i).name\_of\_the\_goods, str3);**

**str4 = Convert\_string\_to\_String(\_clients.Get(i).date, str4);**

**if (\_clients.Get(i).mode\_list == 0)**

**str5 = "пусто";**

**else if (\_clients.Get(i).mode\_list == 1)**

**str5 = "в черном списке";**

**else if (\_clients.Get(i).mode\_list == 2)**

**str5 = "в списке со скидками";**

**dataGridView->Rows->Add(str, str1, str2, str5, \_clients.Get(i).series, \_clients.Get(i).number, str3, str4);**

**dataGridView->Rows[i]->HeaderCell->Value = Convert::ToString(i + 1);**

**}**

**}**

**System::Void Сlientdatabase::WorkForm::выходToolStripMenuItem\_Click(System::Object^ sender, System::EventArgs^ e)**

**{**

**if (MessageBox::Show("Продолжить?", "Внимание!", MessageBoxButtons::YesNo) == Windows::Forms::DialogResult::Yes) {**

**Application::Exit();**

**}**

**}**

**System::Void Сlientdatabase::WorkForm::оПрограммеToolStripMenuItem\_Click(System::Object^ sender, System::EventArgs^ e)**

**{**

**MessageBox::Show("Данная программа предназначена для работы данными клиентов!","Информация о программе!");**

**}**

**String^ Сlientdatabase::WorkForm::GetDateString(String^ date)**

**{**

**String^ buf = date;**

**String^ str = "";**

**for (int i = 0; i < buf->Length; i++) {**

**if (buf[i] != ' ') {**

**str += buf[i];**

**}**

**else {**

**break;**

**}**

**}**

**return str;**

**}**

**System::Void Сlientdatabase::WorkForm::перейтиКГрафикуToolStripMenuItem\_Click(System::Object^ sender, System::EventArgs^ e)**

**{**

**if (groupBoxDate->Visible) {**

**groupBoxDate->Visible = false;**

**}**

**else {**

**groupBoxDate->Visible = true;**

**}**

**}**

**System::Void Сlientdatabase::WorkForm::buttonInChart\_Click(System::Object^ sender, System::EventArgs^ e)**

**{**

**SetData();**

**//Получаем выделенные даты**

**List<String^> dates;**

**String^ buf = GetDateString(monthCalendar->SelectionStart.ToString());**

**Date date = ParseDate(buf);**

**int days = (monthCalendar->SelectionRange->End - monthCalendar->SelectionRange->Start).Days + 1;**

**int\* counts = new int[days];**

**for (int i = 0; i < days; i++) {**

**counts[i] = 0;**

**}**

**//Проходим по все выбранным датам**

**buf = date.day.ToString() + "." + date.month.ToString() + "." + date.year.ToString();**

**dates.Add(buf);**

**for (int i = 0; i < days - 1; i++) {**

**if (date.month == 1) {**

**if (date.day <= 31)**

**date.day++;**

**else {**

**date.month++;**

**date.day = 1;**

**}**

**}**

**else if (date.month == 2) {**

**if (date.day <= 28)**

**date.day++;**

**else {**

**date.month++;**

**date.day = 1;**

**}**

**}**

**else if (date.month == 3) {**

**if (date.day <= 31)**

**date.day++;**

**else {**

**date.month++;**

**date.day = 1;**

**}**

**}**

**else if (date.month == 4) {**

**if (date.day <= 30)**

**date.day++;**

**else {**

**date.month++;**

**date.day = 1;**

**}**

**}**

**else if (date.month == 5) {**

**if (date.day <= 31)**

**date.day++;**

**else {**

**date.month++;**

**date.day = 1;**

**}**

**}**

**else if (date.month == 6) {**

**if (date.day <= 30)**

**date.day++;**

**else {**

**date.month++;**

**date.day = 1;**

**}**

**}**

**else if (date.month == 7) {**

**if (date.day <= 31)**

**date.day++;**

**else {**

**date.month++;**

**date.day = 1;**

**}**

**}**

**else if (date.month == 8) {**

**if (date.day <= 31)**

**date.day++;**

**else {**

**date.month++;**

**date.day = 1;**

**}**

**}**

**else if (date.month == 9) {**

**if (date.day <= 30)**

**date.day++;**

**else {**

**date.month++;**

**date.day = 1;**

**}**

**}**

**else if (date.month == 10) {**

**if (date.day <= 31)**

**date.day++;**

**else {**

**date.month++;**

**date.day = 1;**

**}**

**}**

**else if (date.month == 11) {**

**if (date.day <= 30)**

**date.day++;**

**else {**

**date.month++;**

**date.day = 1;**

**}**

**}**

**else if (date.month == 12) {**

**if (date.day <= 30)**

**date.day++;**

**else {**

**date.month++;**

**date.day = 1;**

**}**

**}**

**else {**

**date.year++;**

**date.month = 1;**

**date.day = 1;**

**}**

**buf = date.day.ToString() + "." + date.month.ToString() + "." + date.year.ToString();**

**dates.Add(buf);**

**}**

**//Проверяем сколько добавили в какой день**

**for (int i = 0; i < dataGridView->Rows->Count - 1; i++) {**

**for (int j = 0; j < days; j++) {**

**if (dataGridView->Rows[i]->Cells[7]->Value->ToString() == dates[j])**

**counts[j]++;**

**}**

**}**

**ChartForm^ form = gcnew ChartForm();**

**//Отправляем ключевые точки в форму графика**

**for (int i = 0; i < dates.Count; i++)**

**form->AddPointString(dates[i]);**

**for (int j = 0; j < days; j++)**

**form->AddPointInt(counts[j]);**

**//Показываем созданную форму**

**form->Show();**

**//Скрываем календарь**

**groupBoxDate->Visible = false;**

**}**

**ChartForm.h**

#pragma once

#include "Functions.h"

namespace Сlientdatabase {

using namespace System;

using namespace System::ComponentModel;

using namespace System::Collections;

using namespace System::Collections::Generic;

using namespace System::Windows::Forms;

using namespace System::Data;

using namespace System::Drawing;

using namespace System::Windows::Forms::DataVisualization::Charting;

/// <summary>

/// Сводка для ChartForm

/// </summary>

public ref class ChartForm : public System::Windows::Forms::Form

{

public:

ChartForm(void)

{

InitializeComponent();

//

//TODO: добавьте код конструктора

//

}

protected:

/// <summary>

/// Освободить все используемые ресурсы.

/// </summary>

~ChartForm()

{

if (components)

{

delete components;

}

}

protected:

protected:

private:

/// <summary>

/// Обязательная переменная конструктора.

/// </summary>

System::ComponentModel::Container ^components;

#pragma region Windows Form Designer generated code

/// <summary>

/// Требуемый метод для поддержки конструктора — не изменяйте

/// содержимое этого метода с помощью редактора кода.

/// </summary>

void InitializeComponent(void)

{

this->SuspendLayout();

//

// ChartForm

//

this->AutoScaleDimensions = System::Drawing::SizeF(6, 13);

this->AutoScaleMode = System::Windows::Forms::AutoScaleMode::Font;

this->ClientSize = System::Drawing::Size(540, 421);

this->Name = L"ChartForm";

this->StartPosition = System::Windows::Forms::FormStartPosition::CenterScreen;

this->Text = L"График";

this->Load += gcnew System::EventHandler(this, &ChartForm::ChartForm\_Load);

this->ResumeLayout(false);

}

#pragma endregion

private: List<String^> pointsString;

private: List<int> pointsInt;

public: void AddPointString(String^ );

public: void AddPointInt(int);

private: System::Void ChartForm\_Load(System::Object^ sender, System::EventArgs^ e);

};

}

**Clients.h**

#pragma once

#include <string>

#include <fstream>

#include <vector>

#include <ctime>

#include <algorithm>

//Структура данных клиента

struct Client {

std::string surname;

std::string name;

std::string patronymic;

int mode\_list;//режим дополнительного списка: 0 - пусто, 1 - черный, 2 - есть скидка

int series;

int number;

std::string name\_of\_the\_goods;

std::string date;

};

class Clients

{

//Поля класса

private: //закрытая часть класса

std::vector<Client> data;

//Методы класса

public: //открытая часть класса

Clients();//конструктур по умолчанию - создает пустой объект класса

~Clients();//деструктор - удаляет объект класса

bool Load(std::string filename);

bool Save(std::string filename);

void Add(Client c);

bool Del(int i);

bool Set(int i, Client c);

Client Get(int i);

bool CreateBlackList(std::string filename);

bool CreateListWithDiscounts(std::string filename);

bool CreateGoods(int i);

bool Sort(int mode);

void Swap(int i, int j);

void Clear() { data.clear(); }

int Size() { return data.size(); }

};

**Functions.h**

#pragma once

#include "Clients.h"

#include "StartForm.h"

#include "WorkForm.h"

#include "ChartForm.h"

using namespace System;

struct Date {

int day,

month,

year;

};

/\* Прототипы функций \*/

std::string& Convert\_String\_to\_string(String^ s, std::string& os);//конвертируем System::string ^ в std::string

String^ Convert\_string\_to\_String(std::string& os, String^ s);//конвертируем std::string в System::string ^

Date ParseDate(String^);

**StartForm.h**

#pragma once

namespace Сlientdatabase {

using namespace System;

using namespace System::ComponentModel;

using namespace System::Collections;

using namespace System::Windows::Forms;

using namespace System::Data;

using namespace System::Drawing;

/// <summary>

/// Сводка для StartForm

/// </summary>

public ref class StartForm : public System::Windows::Forms::Form

{

public:

StartForm(void);

protected:

/// <summary>

/// Освободить все используемые ресурсы.

/// </summary>

~StartForm()

{

if (components)

{

delete components;

}

}

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

private: System::Windows::Forms::OpenFileDialog^ openFileDialog;

private: System::Windows::Forms::Button^ buttonOpen;

private: System::Windows::Forms::Button^ buttonCreate;

protected:

private:

/// <summary>

/// Обязательная переменная конструктора.

/// </summary>

System::ComponentModel::Container ^components;

#pragma region Windows Form Designer generated code

/// <summary>

/// Требуемый метод для поддержки конструктора — не изменяйте

/// содержимое этого метода с помощью редактора кода.

/// </summary>

void InitializeComponent(void)

{

this->label1 = (gcnew System::Windows::Forms::Label());

this->openFileDialog = (gcnew System::Windows::Forms::OpenFileDialog());

this->buttonOpen = (gcnew System::Windows::Forms::Button());

this->buttonCreate = (gcnew System::Windows::Forms::Button());

this->SuspendLayout();

//

// label1

//

this->label1->AutoSize = true;

this->label1->Font = (gcnew System::Drawing::Font(L"Microsoft Sans Serif", 14.25F, System::Drawing::FontStyle::Regular, System::Drawing::GraphicsUnit::Point,

static\_cast<System::Byte>(204)));

this->label1->Location = System::Drawing::Point(12, 15);

this->label1->Name = L"label1";

this->label1->Size = System::Drawing::Size(155, 24);

this->label1->TabIndex = 0;

this->label1->Text = L"Выберите файл:";

//

// openFileDialog

//

this->openFileDialog->FileName = L"openFileDialog";

//

// buttonOpen

//

this->buttonOpen->Font = (gcnew System::Drawing::Font(L"Microsoft Sans Serif", 12, System::Drawing::FontStyle::Regular, System::Drawing::GraphicsUnit::Point,

static\_cast<System::Byte>(204)));

this->buttonOpen->Location = System::Drawing::Point(201, 12);

this->buttonOpen->Name = L"buttonOpen";

this->buttonOpen->Size = System::Drawing::Size(87, 32);

this->buttonOpen->TabIndex = 1;

this->buttonOpen->Text = L"Открыть";

this->buttonOpen->UseVisualStyleBackColor = true;

this->buttonOpen->Click += gcnew System::EventHandler(this, &StartForm::buttonOpen\_Click);

//

// buttonCreate

//

this->buttonCreate->Font = (gcnew System::Drawing::Font(L"Microsoft Sans Serif", 12, System::Drawing::FontStyle::Regular, System::Drawing::GraphicsUnit::Point,

static\_cast<System::Byte>(204)));

this->buttonCreate->Location = System::Drawing::Point(294, 12);

this->buttonCreate->Name = L"buttonCreate";

this->buttonCreate->Size = System::Drawing::Size(87, 32);

this->buttonCreate->TabIndex = 2;

this->buttonCreate->Text = L"Создать";

this->buttonCreate->UseVisualStyleBackColor = true;

this->buttonCreate->Click += gcnew System::EventHandler(this, &StartForm::buttonCreate\_Click);

//

// StartForm

//

this->AutoScaleDimensions = System::Drawing::SizeF(6, 13);

this->AutoScaleMode = System::Windows::Forms::AutoScaleMode::Font;

this->ClientSize = System::Drawing::Size(393, 56);

this->Controls->Add(this->buttonCreate);

this->Controls->Add(this->buttonOpen);

this->Controls->Add(this->label1);

this->MaximizeBox = false;

this->MaximumSize = System::Drawing::Size(409, 91);

this->MinimumSize = System::Drawing::Size(409, 91);

this->Name = L"StartForm";

this->StartPosition = System::Windows::Forms::FormStartPosition::CenterScreen;

this->Text = L"Выберите файл";

this->Load += gcnew System::EventHandler(this, &StartForm::StartForm\_Load);

this->ResumeLayout(false);

this->PerformLayout();

}

#pragma endregion

private: System::Void buttonOpen\_Click(System::Object^ sender, System::EventArgs^ e);

private: System::Void buttonCreate\_Click(System::Object^ sender, System::EventArgs^ e);

private: System::Void StartForm\_Load(System::Object^ sender, System::EventArgs^ e) {

}

};

}

**WorkForm.h**

#pragma once

namespace Сlientdatabase {

using namespace System;

using namespace System::ComponentModel;

using namespace System::Collections;

using namespace System::Collections::Generic;

using namespace System::Windows::Forms;

using namespace System::Data;

using namespace System::Drawing;

/// <summary>

/// Сводка для WorkForm

/// </summary>

public ref class WorkForm : public System::Windows::Forms::Form

{

public:

WorkForm(void)

{

InitializeComponent();

//

//TODO: добавьте код конструктора

//

}

protected:

/// <summary>

/// Освободить все используемые ресурсы.

/// </summary>

~WorkForm()

{

if (components)

{

delete components;

}

}

private: System::Windows::Forms::MenuStrip^ menuStrip1;

protected:

private: System::Windows::Forms::ToolStripMenuItem^ менюToolStripMenuItem;

private: System::Windows::Forms::ToolStripMenuItem^ удалитьToolStripMenuItem;

private: System::Windows::Forms::ToolStripMenuItem^ сохранитьToolStripMenuItem;

private: System::Windows::Forms::ToolStripMenuItem^ сформироватьЧерныйСписокToolStripMenuItem;

private: System::Windows::Forms::ToolStripMenuItem^ сформироватьСписокКлиентовСоСкидкамиToolStripMenuItem;

private: System::Windows::Forms::ToolStripMenuItem^ вернутьсяToolStripMenuItem;

private: System::Windows::Forms::ToolStripMenuItem^ сортировкаToolStripMenuItem;

private: System::Windows::Forms::ToolStripMenuItem^ поФамилииToolStripMenuItem;

private: System::Windows::Forms::ToolStripMenuItem^ оПрограммеToolStripMenuItem;

private: System::Windows::Forms::ToolStripMenuItem^ выходToolStripMenuItem;

private: System::Windows::Forms::ToolStripMenuItem^ поДополнительнымСпискамToolStripMenuItem;

private: System::Windows::Forms::ToolStripMenuItem^ поКоличествоТоваровToolStripMenuItem;

private: System::Windows::Forms::ToolStripMenuItem^ поДатеДобавленияToolStripMenuItem;

private: System::Windows::Forms::DataGridView^ dataGridView;

private: System::Windows::Forms::ToolStripMenuItem^ добавитьToolStripMenuItem;

private: System::Windows::Forms::OpenFileDialog^ openFileDialog;

private: System::Windows::Forms::DataGridViewTextBoxColumn^ Column1;

private: System::Windows::Forms::DataGridViewTextBoxColumn^ Column2;

private: System::Windows::Forms::DataGridViewTextBoxColumn^ Column3;

private: System::Windows::Forms::DataGridViewTextBoxColumn^ Column4;

private: System::Windows::Forms::DataGridViewTextBoxColumn^ Column5;

private: System::Windows::Forms::DataGridViewTextBoxColumn^ Column6;

private: System::Windows::Forms::DataGridViewTextBoxColumn^ Column7;

private: System::Windows::Forms::DataGridViewTextBoxColumn^ Column8;

private: System::Windows::Forms::ToolStripMenuItem^ перейтиКГрафикуToolStripMenuItem;

private: System::Windows::Forms::GroupBox^ groupBoxDate;

private: System::Windows::Forms::Button^ buttonInChart;

private: System::Windows::Forms::MonthCalendar^ monthCalendar;

private:

/// <summary>

/// Обязательная переменная конструктора.

/// </summary>

System::ComponentModel::Container ^components;

#pragma region Windows Form Designer generated code

/// <summary>

/// Требуемый метод для поддержки конструктора — не изменяйте

/// содержимое этого метода с помощью редактора кода.

/// </summary>

void InitializeComponent(void)

{

this->menuStrip1 = (gcnew System::Windows::Forms::MenuStrip());

this->менюToolStripMenuItem = (gcnew System::Windows::Forms::ToolStripMenuItem());

this->добавитьToolStripMenuItem = (gcnew System::Windows::Forms::ToolStripMenuItem());

this->удалитьToolStripMenuItem = (gcnew System::Windows::Forms::ToolStripMenuItem());

this->сохранитьToolStripMenuItem = (gcnew System::Windows::Forms::ToolStripMenuItem());

this->сформироватьЧерныйСписокToolStripMenuItem = (gcnew System::Windows::Forms::ToolStripMenuItem());

this->сформироватьСписокКлиентовСоСкидкамиToolStripMenuItem = (gcnew System::Windows::Forms::ToolStripMenuItem());

this->сортировкаToolStripMenuItem = (gcnew System::Windows::Forms::ToolStripMenuItem());

this->поФамилииToolStripMenuItem = (gcnew System::Windows::Forms::ToolStripMenuItem());

this->поДополнительнымСпискамToolStripMenuItem = (gcnew System::Windows::Forms::ToolStripMenuItem());

this->поКоличествоТоваровToolStripMenuItem = (gcnew System::Windows::Forms::ToolStripMenuItem());

this->поДатеДобавленияToolStripMenuItem = (gcnew System::Windows::Forms::ToolStripMenuItem());

this->вернутьсяToolStripMenuItem = (gcnew System::Windows::Forms::ToolStripMenuItem());

this->перейтиКГрафикуToolStripMenuItem = (gcnew System::Windows::Forms::ToolStripMenuItem());

this->оПрограммеToolStripMenuItem = (gcnew System::Windows::Forms::ToolStripMenuItem());

this->выходToolStripMenuItem = (gcnew System::Windows::Forms::ToolStripMenuItem());

this->dataGridView = (gcnew System::Windows::Forms::DataGridView());

this->Column1 = (gcnew System::Windows::Forms::DataGridViewTextBoxColumn());

this->Column2 = (gcnew System::Windows::Forms::DataGridViewTextBoxColumn());

this->Column3 = (gcnew System::Windows::Forms::DataGridViewTextBoxColumn());

this->Column4 = (gcnew System::Windows::Forms::DataGridViewTextBoxColumn());

this->Column5 = (gcnew System::Windows::Forms::DataGridViewTextBoxColumn());

this->Column6 = (gcnew System::Windows::Forms::DataGridViewTextBoxColumn());

this->Column7 = (gcnew System::Windows::Forms::DataGridViewTextBoxColumn());

this->Column8 = (gcnew System::Windows::Forms::DataGridViewTextBoxColumn());

this->openFileDialog = (gcnew System::Windows::Forms::OpenFileDialog());

this->groupBoxDate = (gcnew System::Windows::Forms::GroupBox());

this->buttonInChart = (gcnew System::Windows::Forms::Button());

this->monthCalendar = (gcnew System::Windows::Forms::MonthCalendar());

this->menuStrip1->SuspendLayout();

(cli::safe\_cast<System::ComponentModel::ISupportInitialize^>(this->dataGridView))->BeginInit();

this->groupBoxDate->SuspendLayout();

this->SuspendLayout();

//

// menuStrip1

//

this->menuStrip1->Items->AddRange(gcnew cli::array< System::Windows::Forms::ToolStripItem^ >(3) {

this->менюToolStripMenuItem,

this->оПрограммеToolStripMenuItem, this->выходToolStripMenuItem

});

this->menuStrip1->Location = System::Drawing::Point(0, 0);

this->menuStrip1->Name = L"menuStrip1";

this->menuStrip1->Size = System::Drawing::Size(924, 24);

this->menuStrip1->TabIndex = 0;

this->menuStrip1->Text = L"menuStrip1";

//

// менюToolStripMenuItem

//

this->менюToolStripMenuItem->DropDownItems->AddRange(gcnew cli::array< System::Windows::Forms::ToolStripItem^ >(8) {

this->добавитьToolStripMenuItem,

this->удалитьToolStripMenuItem, this->сохранитьToolStripMenuItem, this->сформироватьЧерныйСписокToolStripMenuItem, this->сформироватьСписокКлиентовСоСкидкамиToolStripMenuItem,

this->сортировкаToolStripMenuItem, this->вернутьсяToolStripMenuItem, this->перейтиКГрафикуToolStripMenuItem

});

this->менюToolStripMenuItem->Name = L"менюToolStripMenuItem";

this->менюToolStripMenuItem->Size = System::Drawing::Size(53, 20);

this->менюToolStripMenuItem->Text = L"Меню";

//

// добавитьToolStripMenuItem

//

this->добавитьToolStripMenuItem->Name = L"добавитьToolStripMenuItem";

this->добавитьToolStripMenuItem->Size = System::Drawing::Size(326, 22);

this->добавитьToolStripMenuItem->Text = L"Добавить";

this->добавитьToolStripMenuItem->Click += gcnew System::EventHandler(this, &WorkForm::добавитьToolStripMenuItem\_Click\_1);

//

// удалитьToolStripMenuItem

//

this->удалитьToolStripMenuItem->Name = L"удалитьToolStripMenuItem";

this->удалитьToolStripMenuItem->Size = System::Drawing::Size(326, 22);

this->удалитьToolStripMenuItem->Text = L"Удалить";

this->удалитьToolStripMenuItem->Click += gcnew System::EventHandler(this, &WorkForm::удалитьToolStripMenuItem\_Click);

//

// сохранитьToolStripMenuItem

//

this->сохранитьToolStripMenuItem->Name = L"сохранитьToolStripMenuItem";

this->сохранитьToolStripMenuItem->Size = System::Drawing::Size(326, 22);

this->сохранитьToolStripMenuItem->Text = L"Сохранить";

this->сохранитьToolStripMenuItem->Click += gcnew System::EventHandler(this, &WorkForm::сохранитьToolStripMenuItem\_Click);

//

// сформироватьЧерныйСписокToolStripMenuItem

//

this->сформироватьЧерныйСписокToolStripMenuItem->Name = L"сформироватьЧерныйСписокToolStripMenuItem";

this->сформироватьЧерныйСписокToolStripMenuItem->Size = System::Drawing::Size(326, 22);

this->сформироватьЧерныйСписокToolStripMenuItem->Text = L"Сформировать черный список";

this->сформироватьЧерныйСписокToolStripMenuItem->Click += gcnew System::EventHandler(this, &WorkForm::сформироватьЧерныйСписокToolStripMenuItem\_Click);

//

// сформироватьСписокКлиентовСоСкидкамиToolStripMenuItem

//

this->сформироватьСписокКлиентовСоСкидкамиToolStripMenuItem->Name = L"сформироватьСписокКлиентовСоСкидкамиToolStripMenuItem";

this->сформироватьСписокКлиентовСоСкидкамиToolStripMenuItem->Size = System::Drawing::Size(326, 22);

this->сформироватьСписокКлиентовСоСкидкамиToolStripMenuItem->Text = L"Сформировать список клиентов со скидками";

this->сформироватьСписокКлиентовСоСкидкамиToolStripMenuItem->Click += gcnew System::EventHandler(this, &WorkForm::сформироватьСписокКлиентовСоСкидкамиToolStripMenuItem\_Click);

//

// сортировкаToolStripMenuItem

//

this->сортировкаToolStripMenuItem->DropDownItems->AddRange(gcnew cli::array< System::Windows::Forms::ToolStripItem^ >(4) {

this->поФамилииToolStripMenuItem,

this->поДополнительнымСпискамToolStripMenuItem, this->поКоличествоТоваровToolStripMenuItem, this->поДатеДобавленияToolStripMenuItem

});

this->сортировкаToolStripMenuItem->Name = L"сортировкаToolStripMenuItem";

this->сортировкаToolStripMenuItem->Size = System::Drawing::Size(326, 22);

this->сортировкаToolStripMenuItem->Text = L"Сортировка";

//

// поФамилииToolStripMenuItem

//

this->поФамилииToolStripMenuItem->Name = L"поФамилииToolStripMenuItem";

this->поФамилииToolStripMenuItem->Size = System::Drawing::Size(240, 22);

this->поФамилииToolStripMenuItem->Text = L"По фамилии";

this->поФамилииToolStripMenuItem->Click += gcnew System::EventHandler(this, &WorkForm::поФамилииToolStripMenuItem\_Click);

//

// поДополнительнымСпискамToolStripMenuItem

//

this->поДополнительнымСпискамToolStripMenuItem->Name = L"поДополнительнымСпискамToolStripMenuItem";

this->поДополнительнымСпискамToolStripMenuItem->Size = System::Drawing::Size(240, 22);

this->поДополнительнымСпискамToolStripMenuItem->Text = L"По дополнительным спискам";

this->поДополнительнымСпискамToolStripMenuItem->Click += gcnew System::EventHandler(this, &WorkForm::поДополнительнымСпискамToolStripMenuItem\_Click);

//

// поКоличествоТоваровToolStripMenuItem

//

this->поКоличествоТоваровToolStripMenuItem->Name = L"поКоличествоТоваровToolStripMenuItem";

this->поКоличествоТоваровToolStripMenuItem->Size = System::Drawing::Size(240, 22);

this->поКоличествоТоваровToolStripMenuItem->Text = L"По количество товаров ";

this->поКоличествоТоваровToolStripMenuItem->Click += gcnew System::EventHandler(this, &WorkForm::поКоличествоТоваровToolStripMenuItem\_Click);

//

// поДатеДобавленияToolStripMenuItem

//

this->поДатеДобавленияToolStripMenuItem->Name = L"поДатеДобавленияToolStripMenuItem";

this->поДатеДобавленияToolStripMenuItem->Size = System::Drawing::Size(240, 22);

this->поДатеДобавленияToolStripMenuItem->Text = L"По дате добавления";

this->поДатеДобавленияToolStripMenuItem->Click += gcnew System::EventHandler(this, &WorkForm::поДатеДобавленияToolStripMenuItem\_Click);

//

// вернутьсяToolStripMenuItem

//

this->вернутьсяToolStripMenuItem->Name = L"вернутьсяToolStripMenuItem";

this->вернутьсяToolStripMenuItem->Size = System::Drawing::Size(326, 22);

this->вернутьсяToolStripMenuItem->Text = L"Вернуться";

this->вернутьсяToolStripMenuItem->Click += gcnew System::EventHandler(this, &WorkForm::вернутьсяToolStripMenuItem\_Click);

//

// перейтиКГрафикуToolStripMenuItem

//

this->перейтиКГрафикуToolStripMenuItem->Name = L"перейтиКГрафикуToolStripMenuItem";

this->перейтиКГрафикуToolStripMenuItem->Size = System::Drawing::Size(326, 22);

this->перейтиКГрафикуToolStripMenuItem->Text = L"Перейти к графику";

this->перейтиКГрафикуToolStripMenuItem->Click += gcnew System::EventHandler(this, &WorkForm::перейтиКГрафикуToolStripMenuItem\_Click);

//

// оПрограммеToolStripMenuItem

//

this->оПрограммеToolStripMenuItem->Name = L"оПрограммеToolStripMenuItem";

this->оПрограммеToolStripMenuItem->Size = System::Drawing::Size(94, 20);

this->оПрограммеToolStripMenuItem->Text = L"О программе";

this->оПрограммеToolStripMenuItem->Click += gcnew System::EventHandler(this, &WorkForm::оПрограммеToolStripMenuItem\_Click);

//

// выходToolStripMenuItem

//

this->выходToolStripMenuItem->Name = L"выходToolStripMenuItem";

this->выходToolStripMenuItem->Size = System::Drawing::Size(53, 20);

this->выходToolStripMenuItem->Text = L"Выход";

this->выходToolStripMenuItem->Click += gcnew System::EventHandler(this, &WorkForm::выходToolStripMenuItem\_Click);

//

// dataGridView

//

this->dataGridView->ColumnHeadersHeightSizeMode = System::Windows::Forms::DataGridViewColumnHeadersHeightSizeMode::AutoSize;

this->dataGridView->Columns->AddRange(gcnew cli::array< System::Windows::Forms::DataGridViewColumn^ >(8) {

this->Column1,

this->Column2, this->Column3, this->Column4, this->Column5, this->Column6, this->Column7, this->Column8

});

this->dataGridView->Dock = System::Windows::Forms::DockStyle::Fill;

this->dataGridView->Location = System::Drawing::Point(0, 24);

this->dataGridView->Name = L"dataGridView";

this->dataGridView->Size = System::Drawing::Size(924, 412);

this->dataGridView->TabIndex = 1;

this->dataGridView->CellContentClick += gcnew System::Windows::Forms::DataGridViewCellEventHandler(this, &WorkForm::dataGridView\_CellContentClick);

//

// Column1

//

this->Column1->HeaderText = L"Фамилия";

this->Column1->Name = L"Column1";

//

// Column2

//

this->Column2->HeaderText = L"Имя";

this->Column2->Name = L"Column2";

//

// Column3

//

this->Column3->HeaderText = L"Отчество";

this->Column3->Name = L"Column3";

//

// Column4

//

this->Column4->HeaderText = L"Дополнительный список";

this->Column4->Name = L"Column4";

this->Column4->Width = 200;

//

// Column5

//

this->Column5->HeaderText = L"Серия паспорта ";

this->Column5->Name = L"Column5";

this->Column5->Width = 150;

//

// Column6

//

this->Column6->HeaderText = L"Номер паспорта";

this->Column6->Name = L"Column6";

this->Column6->Width = 150;

//

// Column7

//

this->Column7->HeaderText = L"Наиболее просматриваемые товары";

this->Column7->Name = L"Column7";

this->Column7->Width = 250;

//

// Column8

//

this->Column8->HeaderText = L"Дата добавления";

this->Column8->Name = L"Column8";

this->Column8->Width = 150;

//

// openFileDialog

//

this->openFileDialog->FileName = L"openFileDialog";

//

// groupBoxDate

//

this->groupBoxDate->Controls->Add(this->buttonInChart);

this->groupBoxDate->Controls->Add(this->monthCalendar);

this->groupBoxDate->Location = System::Drawing::Point(12, 47);

this->groupBoxDate->Name = L"groupBoxDate";

this->groupBoxDate->Size = System::Drawing::Size(185, 235);

this->groupBoxDate->TabIndex = 2;

this->groupBoxDate->TabStop = false;

this->groupBoxDate->Text = L"Выберите период ";

this->groupBoxDate->Visible = false;

//

// buttonInChart

//

this->buttonInChart->Font = (gcnew System::Drawing::Font(L"Microsoft Sans Serif", 9.75F, System::Drawing::FontStyle::Regular, System::Drawing::GraphicsUnit::Point,

static\_cast<System::Byte>(204)));

this->buttonInChart->Location = System::Drawing::Point(35, 199);

this->buttonInChart->Name = L"buttonInChart";

this->buttonInChart->Size = System::Drawing::Size(118, 28);

this->buttonInChart->TabIndex = 1;

this->buttonInChart->Text = L"Продолжить";

this->buttonInChart->UseVisualStyleBackColor = true;

this->buttonInChart->Click += gcnew System::EventHandler(this, &WorkForm::buttonInChart\_Click);

//

// monthCalendar

//

this->monthCalendar->Location = System::Drawing::Point(12, 25);

this->monthCalendar->Name = L"monthCalendar";

this->monthCalendar->TabIndex = 0;

//

// WorkForm

//

this->AutoScaleDimensions = System::Drawing::SizeF(6, 13);

this->AutoScaleMode = System::Windows::Forms::AutoScaleMode::Font;

this->ClientSize = System::Drawing::Size(924, 436);

this->Controls->Add(this->groupBoxDate);

this->Controls->Add(this->dataGridView);

this->Controls->Add(this->menuStrip1);

this->MainMenuStrip = this->menuStrip1;

this->Name = L"WorkForm";

this->StartPosition = System::Windows::Forms::FormStartPosition::CenterScreen;

this->Text = L"База данных клиентов";

this->Load += gcnew System::EventHandler(this, &WorkForm::WorkForm\_Load);

this->menuStrip1->ResumeLayout(false);

this->menuStrip1->PerformLayout();

(cli::safe\_cast<System::ComponentModel::ISupportInitialize^>(this->dataGridView))->EndInit();

this->groupBoxDate->ResumeLayout(false);

this->ResumeLayout(false);

this->PerformLayout();

}

#pragma endregion

public: bool load;

public: String^ filename;

private: System::Void WorkForm\_Load(System::Object^ sender, System::EventArgs^ e);

private: System::Void добавитьToolStripMenuItem\_Click\_1(System::Object^ sender, System::EventArgs^ e);

private: System::Void удалитьToolStripMenuItem\_Click(System::Object^ sender, System::EventArgs^ e);

private: System::Void сохранитьToolStripMenuItem\_Click(System::Object^ sender, System::EventArgs^ e);

private: System::Void сформироватьЧерныйСписокToolStripMenuItem\_Click(System::Object^ sender, System::EventArgs^ e);

private: System::Void сформироватьСписокКлиентовСоСкидкамиToolStripMenuItem\_Click(System::Object^ sender, System::EventArgs^ e);

private: System::Void вернутьсяToolStripMenuItem\_Click(System::Object^ sender, System::EventArgs^ e);

private: System::Void поФамилииToolStripMenuItem\_Click(System::Object^ sender, System::EventArgs^ e);

private: System::Void поДополнительнымСпискамToolStripMenuItem\_Click(System::Object^ sender, System::EventArgs^ e);

private: System::Void поКоличествоТоваровToolStripMenuItem\_Click(System::Object^ sender, System::EventArgs^ e);

private: System::Void поДатеДобавленияToolStripMenuItem\_Click(System::Object^ sender, System::EventArgs^ e);

private: void SetData();

private: void OutputData();

private: System::Void выходToolStripMenuItem\_Click(System::Object^ sender, System::EventArgs^ e);

private: System::Void оПрограммеToolStripMenuItem\_Click(System::Object^ sender, System::EventArgs^ e);

private: String^ GetDateString(String^ date);

private: System::Void перейтиКГрафикуToolStripMenuItem\_Click(System::Object^ sender, System::EventArgs^ e);

private: System::Void buttonInChart\_Click(System::Object^ sender, System::EventArgs^ e);

private: System::Void dataGridView\_CellContentClick(System::Object^ sender, System::Windows::Forms::DataGridViewCellEventArgs^ e) {

}

};

}

|  |  |  |  |  |  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- |
| Обозначение | | | | Наименование | | | | Дополнительные сведения | | | |
|  | | | | Текстовые документы | | | |  | | | |
|  | | | |  | | | |  | | | |
| БГУИР КП 1–40 01 01 008 ПЗ | | | | Пояснительная записка | | | | 75 c. | | | |
|  | | | |  | | | |  | | | |
|  | | | |  | | | |  | | | |
|  | | | | Графические документы | | | |  | | | |
|  | | | |  | | | |  | | | |
| ГУИР 951005 008 ПД | | | | Схема программы | | | | Формат А1 | | | |
|  | | | |  | | | |  | | | |
|  | | | |  | | | |  | | | |
|  | | | |  | | | |  | | | |
|  | | | |  | | | |  | | | |
|  | | | |  | | | |  | | | |
|  | | | |  | | | |  | | | |
|  | | | |  | | | |  | | | |
|  | | | |  | | | |  | | | |
|  | | | |  | | | |  | | | |
|  | | | |  | | | |  | | | |
|  | | | |  | | | |  | | | |
|  | | | |  | | | |  | | | |
|  | | | |  | | | |  | | | |
|  | | | |  | | | |  | | | |
|  | | | |  | | | |  | | | |
|  |  |  |  |  | БГУИР КП 1-40 01 01 008 Д1 | | | | | | |
|  |  |  |  |  |
|  |  |  |  |  |
| Изм. | Л. | № докум. | Подп. | Дата | База данных клиентов  интернет-магазина  Ведомость курсового  проекта |  | | | | Лист | Листов |
| Разраб. | | Доведько Д.Ю. |  | 27.12.20 | Т |  | |  | 75 | 75 |
| Пров. | | Болтак С.В. |  | 27.12.20 | Кафедра ПОИТ  гр. 951005 | | | | | |
|  | |  |  |  |