

Министерство образования Республики Беларусь
Учреждение образования «Белорусский государственный университет
информатики и радиоэлектроники»

Факультет компьютерных систем и сетей

Кафедра программного обеспечения информационных технологий

Дисциплина: Основы алгоритмизации и программирования

ПОЯСНИТЕЛЬНАЯ ЗАПИСКА
к курсовой работе
на тему

ПРОГРАММНОЕ СРЕДСТВО МОНИТОРИНГА И ОТОБРАЖЕНИЯ
СЕТЕВОЙ СТРУКТУРЫ ОТГРУЗКИ ТОВАРОВ СО СКЛАДОВ В
МАГАЗИНЫ

БГУИР КР 6-05-0612-01 122 ПЗ

Студент: гр. 451001 Соболев Н.Г.

Руководитель:
асс. Фадеева Е.Е.

Минск 2025

Учреждение образования

«Белорусский государственный университет информатики и
радиоэлектроники»

Факультет компьютерных систем и сетей

УТВЕРЖДАЮ

Заведующий кафедрой ПОИТ

Лапицкая Н.В.

(подпись)

11.02.2025

ЗАДАНИЕ

по курсовому проектированию

Студенту Соболю Никите Глебовичу 451001

1. Тема работы Программное средство мониторинга и отображения сетевой структуры отгрузки товаров со складов в магазины.

2. Срок сдачи студентом законченной работы 02.06.2025 г.

3. Исходные данные к работе язык программирования Delphi. Программное средство на вход получают данные о товарах, складах, магазинах, транспортных маршрутах, заказах. На выход отдаёт графическое отображение сетевой структуры отгрузок, отчеты и аналитику по отгрузке товаров, уровень запасов в магазинах и на складах.

4. Содержание расчётно-пояснительной записки (перечень вопросов, которые подлежат разработке)
Введение.

1. Анализ прототипов, литературных источников и формирование требований к проектируемому программному средству;

2. Анализ требований к программному средству и разработка функциональных требований;

3. Проектирование программного средства;

4. Создание (конструирование) программного средства;

5. Тестирование, проверка работоспособности и анализ полученных результатов;

6. Руководство по установке и использованию;

Список используемой литературы

Заключение

5. Перечень графического материала (с точным обозначением обязательных чертежей и графиков)

1. "Программное средство мониторинга и отображения сетевой структуры отгрузки товаров со складов в магазины", А1, схема программы, чертеж

6. Консультант по курсовой работе

Фадеева Е.Е.

7. Дата выдачи задания 11.02.2025

8. Календарный график работы над курсовой работой на весь период проектирования (с обозначением сроков выполнения и процентом от общего объёма работы):

раздел 1 к 24.02.2025 – 15 % готовности работы;

разделы 2, 3 к 20.03.2025 – 30 % готовности работы;

разделы 4, 5 к 21.04.2025 – 60 % готовности работы;

раздел 6 к 19.05.2025 – 90 % готовности работы;

оформление пояснительной записки и графического материала к 02.06.2025 – 100 % готовности работы.

Защита курсовой работы с 02.06.2025 по 06.06.2025 г.

РУКОВОДИТЕЛЬ Е.Е.Фадеева
(подпись)

Задание принял к исполнению _____
(дата и подпись студента)

СОДЕРЖАНИЕ

СОДЕРЖАНИЕ	4
ВВЕДЕНИЕ	Ошибка! Закладка не определена.
1. АНАЛИЗ ПРОТОТИПОВ, ЛИТЕРАТУРНЫХ ИСТОЧНИКОВ И ФОРМИРОВАНИЕ ТРЕБОВАНИЙ К ПРОЕКТИРУЕМОМУ	7
1.1. Обзор литературы	Ошибка! Закладка не определена.
1.2. Примеры решения аналогичных задач, анализ достоинств и недостатков известных решений	7
1.3. Требования к разрабатываемому программному средству	8
2. АНАЛИЗ ТРЕБОВАНИЙ К ПРОГРАММНОМУ СРЕДСТВУ И РАЗРАБОТКА ФУНКЦИОНАЛЬНЫХ ТРЕБОВАНИЙ	12
2.1. Теоретический анализ, математическое обоснование и доказательства, модели технических объектов и результаты моделирования	12
2.2. Описание функциональности программного средства	12
3. ПРОЕКТИРОВАНИЕ ПРОГРАММНОГО СРЕДСТВА	16
3.1. Проектирование главного модуля	16
3.2. Проектирование алгоритмов работы с декартовыми деревьями	18
3.3. Проектирование алгоритмов хеширования	23
3.4. Проектирование алгоритмов фильтрации	24
3.5. Проектирование алгоритмов валидации	26
3.6. Проектирование алгоритмов управления отгрузками	30
3.7. Проектирование алгоритмов управления стрелками	31
3.8. Проектирование алгоритмов диалогового взаимодействия с пользователем	33
4. КОНСТРУИРОВАНИЕ ПРОГРАММНОГО СРЕДСТВА	34
4.1. Структура программного средства	34
4.2. Разработка модуля BalanceUnit	35
4.3. Разработка модуля CartesianTree	36
4.4. Разработка модуля CartesianTreeByName	37
4.5. Разработка модуля CartesianTreeItem	38
4.6. Разработка модуля Filter	40
4.7. Разработка модуля GetKeys	40
4.8. Разработка модуля Hash	40
4.9. Разработка модуля MainUnit	40

4.10. Разработка модуля Messages	43
4.11. Разработка модуля SelectShipmentsUnit	44
4.12. Разработка модуля Shipments	45
4.13. Разработка модуля ShipmentsTableUnit.....	45
4.14. Разработка модуля TableUnit.....	45
4.15. Разработка модуля Validation	46
4.16. Разработка модуля ArrowsUnit.....	47
5. ТЕСТИРОВАНИЕ, ПРОВЕРКА РАБОТОСПОСОБНОСТИ И АНАЛИЗ ПОЛУЧЕННЫХ РЕЗУЛЬТАТОВ	48
5.1. Описание тестов, результаты тестирования	48
6. РУКОВОДСТВО ПО УСТАНОВКЕ И ИСПОЛЬЗОВАНИЮ	54
ЗАКЛЮЧЕНИЕ	65
СПИСОК ИСПОЛЬЗУЕМОЙ ЛИТЕРАТУРЫ	66
ПРИЛОЖЕНИЕ А	67
ПРИЛОЖЕНИЕ Б.....	68
Содержание модуля BalanceUnit.....	68
Содержание модуля CartesianTree	71
Содержание модуля CartesianTreeByName	74
Содержание модуля CartesianTreeItem	76
Содержание модуля Filter	79
Содержание модуля GetKeys.....	81
Содержание модуля Hash.....	81
Содержание модуля MainUnit	82
Содержание модуля Messages	118
Содержание модуля SelectShipmentsUnit.....	120
Содержание модуля shipments.....	125
Содержание модуля ShipmentsTableUnit.....	127
Содержание модуля TableUnit.....	130
Содержание модуля Validation	132
Содержание модуля ArrowsUnit.....	134
Содержание модуля Types	135
Содержание модуля Vars	137
ВЕДОМОСТЬ	138

ВВЕДЕНИЕ

Целью данного проекта является создание программного средства мониторинга и визуализации сетевой структуры отгрузок товаров со складов в магазины, которое позволит в реальном времени отслеживать движение грузов, формировать отчёты по остаткам и анализировать эффективность выполнения логистических операций. Основные функции будущего программного обеспечения включают обработку данных о складах, магазинах, товарах и отгрузках; автоматическую валидацию вводимых данных на корректность; хранение информации в локальных структурах, обеспечивающих быстрый поиск и обновление; а также сохранение и восстановление состояния в файлах.

При проектировании использован ряд принципов, позволяющих сочетать производительность и удобство: хранение объектов (складов и магазинов) и товаров в сбалансированных структурах данных (например, в виде декартовых деревьев), что создает возможность осуществлять большинство операций над объектами за логарифмическое время. Есть применение модулей визуализации, адаптирующих карту к любому разрешению экрана и отображающих склады и магазины условными символами бордового и синего цвета соответственно, а маршруты – в виде чёрных линий.

Для разработки выбран Delphi с применением RAD Studio, что обеспечивает быстрое создание графического интерфейса, поддержку работы на Windows и возможность расширения функционала за счёт встроенных компонентных библиотек.

В результате пользователю будет доступна система, которая в одном окне объединяет карту сети, таблицы с актуальным состоянием складов и магазинов, а также списки отгрузок с возможностью фильтрации по различным критериям. Пользователь получит интуитивно понятный интерфейс, позволяющий быстро создавать, редактировать и удалять склады, магазины, товары и отгрузки, а также видеть на карте, как изменяется логистическая сеть в реальном времени.

1. АНАЛИЗ ПРОТОТИПОВ, ЛИТЕРАТУРНЫХ ИСТОЧНИКОВ И ФОРМИРОВАНИЕ ТРЕБОВАНИЙ К ПРОЕКТИРУЕМОМУ

1.1 Обзор литературы

1.1.1 [1] algorithmica [Электронный ресурс]. – Режим доступа: <https://ru.algorithmica.org/>

Подробное описание множества алгоритмов, в том числе хеширования и декартового дерева.

1.1.2 [2] e-maxx [Электронный ресурс]. – Режим доступа: <http://e-maxx.ru/algo>

Подробное описание множества алгоритмов, в том числе хеширования и декартового дерева.

1.1.3 [3] docwiki.embarcadero [Электронный ресурс]. – Режим доступа: <https://docwiki.embarcadero.com/>

Подробное описание работы с delphi и RAD Studio. Документация по процедурам и функциям в языке delphi. Подробное описание создания оконных приложений при помощи RAD Studio.

1.2 Примеры решения аналогичных задач, анализ достоинств и недостатков известных решений

Рассмотрим уже существующие программные средства, решающие задачу мониторинга и отображения сетевой структуры отгрузки товаров со складов в магазины:

1) Zabbix

Достоинства:

- А) Бесплатное программное обеспечение;
- Б) Мощные возможности мониторинга и уведомлений;
- В) Есть API;
- Г) Гибкая настройка.

Недостатки:

- А) Высокий порог входа;
- Б) Требуется тяжёлая настройка.

2) Nagios

Достоинства:

- А) Высокая гибкость и настраиваемость;
- Б) Множество плагинов.

Недостатки:

А) Сложная настройка и управление.

3) SAP ERP

Достоинства:

А) Высокая функциональность;

Б) Интеграция с другими модулями ERP.

Недостатки:

А) Высокая стоимость;

Б) Сложность внедрения.

4) Oracle NetSuite

Достоинства:

А) Облачная платформа;

Б) Простота использования.

Недостатки:

А) Ограниченные возможности кастомизации.

1.3 Требования к проектируемому программному средству

1.3.1 Назначение разработки. Проектируемое программное средство предназначено для мониторинга и отображения сетевой структуры отгрузки товаров со складов в магазины в реальном времени. Оно должно обеспечивать отслеживание отгрузок, а также генерировать таблицы для анализа логистической цепочки. Система должна обеспечивать высокую степень автоматизации, снижать вероятность ошибок и ускорять принятие управленческих решений.

1.3.2 Состав выполняемых функций:

1.3.2.1 Управление складами и магазинами:

1) Создание нового склада или магазина с вводом: название, улица, дом, корпус (опционально), общая вместимость. При сохранении проводится проверка ввода, создаётся узел в treap-структуре, вычисляются координаты X, Y (по данным пользователя) и добавляется графическое отображения объекта на карту;

2) Редактирование и удаление существующих объектов изменение адресных и ёмкостных параметров, проверка корректности данных и обновление treap-структур;

3) Отображение детальной информации об объекте: статистика по уровню занятости, списку товаров, возможностям отгрузок.

1.3.2.2 Управление товарами в рамках каждого узла. Добавление товара: ввод наименования, категории, объёма единицы (количество условных

единиц) и текущего количества. Реализована авторизация по существующим категориям, проверка уникальности ключа товара в trear-структуре.

1.3.2.3 Управление отгрузками:

1) Создание новой отгрузки с выбором отправителя (склада или магазина), получателя (склад или магазин), наименования товара и количества. Система проверяет наличие достаточного запаса и обновляет поле, отвечающее за количество товара, которое необходимо отправить;

2) При создании отгрузки создаётся структура с полями: имя отгрузки, уникальный ID, указатели на отправителя и получателя, наименование товара и количество. Данные об отгрузке добавляются в связный список отгрузок и сразу же создаётся объект для визуализации отгрузки на карте.

1.3.2.4 Фильтрация объектов. Пользователь может задавать параметры фильтрации: тип объекта (склад/магазин), улица, дом, корпус, диапазон общей вместимости, диапазон текущей занятости. При применении фильтра вызывается функция, скрывающая все графические элементы объектов, не подходящих под критерии. Карта перерисовывается после каждой фильтрации.

1.3.2.5 Визуализация сетевой структуры отгрузок. Компонент TPaintBox загружает фон и рисует стрелки между координатами узлов. Если у отправителя или получателя свойство видимости отключено, соответствующая стрелка не рисуется.

1.3.2.6 Формирование таблиц:

1) Таблица остатков товаров во всех магазинах и на складах. При открытии формы данные извлекаются из trear-структур и отображаются в TStringGrid;

2) Таблица существующих отгрузок. Информация о каждой отгрузке читается из связного списка отгрузок и отображается с указанием: имя отгрузки, ID, отправитель, адрес отправителя, получатель, адрес получателя, товар, артикул, количество;

3) Таблицы выборочно по остаткам товаров в конкретном магазине или складе. Пользователь выбирает узел, и вызывается соответствующая процедура, проходящая по списку товаров данного узла и формирующая таблицу.

1.3.2.7 Сохранение данных:

1) Все данные хранятся в трёх текстовых файлах: shops.txt, warehouses.txt, shipments.txt;

2) При сохранении: для каждого склада/магазина записываются строки с полями: количество объектов, описание каждого объекта. Поля описания каждого объекта: название, улица, дом, корпус, вместимость, занятая вместимость, зарезервированное место, идентификатор, координаты, количество различных позиций, описание позиций. Каждая позиция содержит поля: название, категория, объём единицы товара, количество, идентификатор, количество, которое необходимо отправить.

3) При сохранении отгрузок формируется файл с полями: количество отгрузок, описание каждой отгрузки. Описание каждой отгрузки содержит: название, идентификатор, идентификаторы отправителя и получателя, название товара, количество.

1.3.3 Входные данные:

1) Информация о складах: название склада, адрес (улица, дом), вместимость.

2) Информация о магазинах: название склада, адрес (улица, дом), вместимость.

3) Данные о товарах: название, категория, объём единицы товара (количество условных единиц)

4) Данные об отгрузках: название, тип отправителя, название отправителя, идентификатор отправителя, тип получателя, название получателя, идентификатор получателя, название товара, артикул товара, количество товара для отгрузки;

1.3.4 Выходные данные:

1) Графическое отображение сетевой структуры отгрузок в виде карты;

2) Формирование таблиц по остаткам товаров во всех магазинах и на складах;

3) Формирование таблиц с существующими отгрузками;

4) Формирование таблиц по остаткам товаров в конкретном магазине или в конкретном складе.

1.3.5 Требования к временным характеристикам:

1) Обновление данных в режиме реального времени;

2) Задержка между обновлением данных и их отображением на экране не должна превышать 1 секунды.

1.3.6 Требования к надёжности. Программное средство должно быть устойчиво к программным ошибкам и сбоям, в том числе проверять все входные данные на корректность.

1.3.7 Условия эксплуатации. Программное средство должно поддерживать работу на ОС Windows.

1.3.8 Язык и среда разработки. Язык разработки: Delphi, благодаря его возможности создания высокопроизводительных и надежных приложений с графическим интерфейсом. Среда разработки: Embarcadero RAD Studio, так как она предлагает мощные инструменты для визуального проектирования и отладки приложений на Delphi.

1.3.9 Дополнительные требования. Интерфейс должен быть простым и понятным для пользователей с разным уровнем технической подготовки. Это включает в себя использование понятных терминов, логичную структуру меню и подсказки для сложных операций. Для отображения сетевой структуры отгрузок необходимо использовать графические элементы (карта, линии на карте, множество панелей), которые позволяют быстро анализировать информацию.

2. АНАЛИЗ ТРЕБОВАНИЙ К ПРОГРАММНОМУ СРЕДСТВУ И РАЗРАБОТКА ФУНКЦИОНАЛЬНЫХ ТРЕБОВАНИЙ

2.1 Теоретический анализ, математическое обоснование и доказательства, модели технических объектов и результаты моделирования

2.1.1 Декартово дерево (treap, дерамида). Декартово дерево – структура данных, которое объединило в себе бинарное дерево поиска (tree) и бинарную кучу (heap). Эта структура данных хранит в себе пары (X, Y), так, что дерамида является бинарным деревом поиска по X и бинарной пирамидой по Y. Таким образом, если в текущей вершине лежит значение (X0, Y0), то в левом поддереве будут лежать элементы со значением $X < X_0$, а в правом со значением $X > X_0$. А также и в левом, и в правом поддереве будут располагаться $Y < Y_0$. В дальнейшей реализации X – ключ, Y – приоритет. Приоритет будет генерироваться случайно, обеспечивая сложность работы $O(\log N)$ в среднем. В итоге декартово дерево позволяет выполнять следующие операции:

- 1) Insert(X, Y) – Вставка элемента в дерево. $O(\log N)$ в среднем;
- 2) Find(X) – Поиск элемента в дереве. $O(\log N)$ в среднем;
- 3) Erase(X) – Удаление элемента из дерева. $O(\log N)$ в среднем;
- 4) Merge(Root1, Root2) – Объединение деревьев. $O(M \log(N/M))$ в среднем;
- 5) Split(Root, X) – Разделение одного дерева на 2 по ключу X. $O(\log N)$ в среднем [2].

2.2 Описание функциональности программного средства

2.2.1 В системе добавление любого нового логистического объекта начинается с того, что пользователь открывает соответствующую форму и вводит его наименование, адрес (улица, дом, корпус), общую вместимость и координаты на карте. Система проверяет, что название не пустое, не нарушает правил длины и символов, и что оно ещё не встречается среди существующих записей; после этого создаётся запись в общем treap-дереве «объекты», где ей присваивается уникальный ключ и случайный приоритет. Одновременно на карте появляется круг определённого цвета (бордовый для склада, синий для магазина), размещённый в указанных координатах. При редактировании объекта пользователь выбирает его на карте, изменяет нужные поля (название, адрес, вместимость, координаты), и система снова проверяет корректность

ввода: если изменилось имя, убеждается в его уникальности; если изменена вместимость, убеждается, что текущее использование не превосходит новое значение. После успешной обработки изменённая запись остаётся в том же месте treap-дерева, а круг на карте перемещается или меняет подпись. Удаление происходит лишь в том случае, если для объекта нет активных отгрузок: система просматривает список маршрутов и, если не находит ни одной связи, удаляет запись из treap, освобождает вложенные товары, и круг исчезает с карты.

Фильтрация работает так: пользователь задаёт условия по типу «склад/магазин», строковому адресу и числовым диапазонам вместимости и занятости, после чего программа обходит все узлы treap-дерева и скрывает или показывает соответствующие круги; вместе с этим анализируются маршруты, и, если объект скрыт, все линии, связанные с ним, тоже временно исчезают. Таким образом, поиск и управление объектами реализованы через единое treap-дерево, а визуализация и фильтрация происходят «на лету» без перезагрузки.

2.2.2 Управление ассортиментом товаров. В рамках управления ассортиментом у каждого склада или магазина имеется собственный treap-список товаров. Чтобы добавить новый товар, пользователь выбирает нужный логистический объект, затем вводит название, категорию, объём одной единицы и количество. Система проверяет, что название корректно по формату и не повторяется среди товаров этого же объекта, а объём и количество являются целыми числами. После этого создаётся новая запись в вложенном treap «товаров» конкретного узла, и одновременно пересчитывается занятость пространства этого склада или магазина (к «текущей занятости» прибавляется произведение объёма на количество), а «доступная вместимость» уменьшается соответственно.

Для просмотра всех товаров в системе предусмотрен табличный вывод, в котором отображаются: объект, в котором лежит товар, тип объекта, вместимость этого объекта, количество занятого и зарезервированного места, название товара, категория товара, артикул, количество, место, занятое товаром.

2.2.3 Планирование и исполнение отгрузок. Пользователь может создать новую отгрузку, выбрав отправителя (склад или магазин), получателя (склад или магазин), товар и количество условных единиц. Перед созданием, система проверяет, что у отправителя есть достаточное количество желаемого товара на балансе. После подтверждения отгрузка записывается в список отгрузок.

Также для каждой действующей отгрузки система визуально показывает маршрут на карте, связывая отправную и конечную точки.

2.2.4 Пользователь может ограничить отображение объектов на карте по нескольким критериям: вид «склад» или «магазин», адресные параметры (улица, дом, корпус), а также диапазоны общей вместимости и текущей занятости. После ввода необходимых условий и нажатия «Применить» система последовательно обходит все записи trear-дерева логистических объектов. Каждый объект сравнивается с условиями фильтра: если хотя бы одно ограничение не выполняется, его круг на карте скрывается. При этом одновременно скрываются все маршруты, связанные с этим объектом, чтобы не оставалось «висячих» линий. При сбросе фильтра все круги и линии снова становятся видимыми, а карта обновляется. Такой подход обеспечивает быстрое и прозрачное управление тем, какие склады и магазины (и их маршруты) в данный момент отображаются пользователю.

2.2.5 Визуализация логистической сети. На основной карте отображаются все склады и магазины условными символами (разный цвет для складов и магазинов). При наведении мыши на объект на карте можно увидеть его краткое описание.

Для каждой активной отгрузки на карте рисуется линия с направлением от точки отправления до точки прибытия. При наведении курсора мыши на отгрузку можно увидеть её подробные сведения: кто отправитель, кто получатель, какой товар, сколько штук было отправлено.

Вся визуализация автоматически обновляется при создании, обработке или удалении отгрузок, а также при добавлении или удалении объектов, чтобы карта всегда отражала актуальное состояние сети.

2.2.6 Сохранение и восстановление состояния. Система поддерживает сохранение полного состояния базы данных (списки объектов, товары, отгрузки, балансы) в файлы. При следующем запуске приложения можно загрузить ранее сохранённые файлы, и система восстановит все объекты, товары, отгрузки и отобразит их на карте.

При загрузке данных прежние списки очищаются, затем последовательно восстанавливаются объекты с их товарами и балансовыми остатками, а затем восстанавливается список отгрузок с соответствующими маршрутами. После завершения загрузки карта автоматически перерисовывается, чтобы отобразить все объекты и текущие маршруты.

2.3 Спецификация функциональных требований

2.3.1 Визуализация сетевой структуры отгрузок. Сетевая структура отгрузок (склады, магазины, маршруты) отображается в виде карты. Карта автоматически подстраивается под разрешение экрана пользователя.

Элементы карты:

- склады, которые обозначаются бордовыми кругами;
- магазины, которые обозначаются синими кругами;
- маршруты, которые отображаются черными линиями.

2.3.2 Построение маршрута начинается в момент, когда система получает новую запись об отгрузке. В этот момент создаётся структура, в которой сохраняются ссылки на объекты-отправитель и -получатель, наименование товара, количество и уникальный идентификатор. После этого в список активных отгрузок добавляется только что созданная. А на карту добавляется прямая линия, соединяющая отправителя и получателя. В процессе работы, когда пользователь перемещает курсор по области карты, система для каждой линии вычисляет минимальное расстояние от указателя до отрезка по стандартной формуле «расстояние от точки до прямой». Если это расстояние оказывается меньше заранее заданного порога, рядом с курсором показывается всплывающая подсказка с подробными данными о соответствующей отгрузке. Такое поведение обеспечивает удобный интерактивный доступ к информации о любой отгрузке прямо на карте.

2.3.3 Добавление / редактирование / удаление объектов. Должно быть реализовано несколько функций, каждая из которых будет отвечать за добавление, редактирование или удаление определенного объекта (магазина / склада). На ввод подаётся информация о новом или уже существующем объекте, в зависимости от того, какая операция будет выполняться. Все данные проходят валидацию. Функция ничего не возвращает, а только добавляет объект в список, редактирует уже существующий объект или удаляет объект из списка.

3. ПРОЕКТИРОВАНИЕ ПРОГРАММНОГО СРЕДСТВА

Проектирование программного средства состоит из проектирования следующих алгоритмов:

- Алгоритмы работы с декартовыми деревьями
- Алгоритмы хеширования
- Алгоритмы фильтрации объектов
- Алгоритмы валидации пользовательского ввода
- Алгоритмы управления отгрузками
- Алгоритмы управления стрелками
- Алгоритмы диалогового взаимодействия с пользователем

3.1 Проектирование главного модуля

Схема программы представлена в Приложении А.

После запуска инициализации в обработчике создания формы (FormCreate) главная единица (MainUnit) сохраняет исходные размеры окна, обнуляет корни деревьев магазинов, складов и отгрузок, настраивает глобальные списки стрелок, устанавливает фильтр по умолчанию и помечает данные как сохранённые. Далее программа переходит в режим ожидания событий.

Когда пользователь выбирает пункт меню «Загрузить файл», выполняется очистка существующих структур (разрушаются деревья, удаляются узлы отгрузок и стрелок) и затем из файла вновь строятся деревья имён и объектов, заполняются списки отгрузок и стрелок, после чего форма возвращается в ожидание.

При перемещении курсора по области карты происходит вычисление попадания в границы отрисованных стрелок (с помощью функции IsPointNearLine), и при совпадении отображается соответствующая панель свойств.

При наведении курсора на существующий объект появляется панель с полной информацией о данном объекте.

Наведение курсора над уже существующим объектом или стрелкой приводит к визуальному изменению курсора, что даёт пользователю обратную связь о возможности взаимодействия.

Нажатие кнопки «Фильтр» открывает панель настройки критериев — после того как пользователь вводит параметры и подтверждает их, модуль читает значения, проверяет их корректность, вызывает CreateFilter, а затем

рекурсивно применяет фильтр к каждому узлу дерева через `ApplyFilter`, скрывая или показывая объекты на карте.

При выборе «Показать остатки» создаётся и отображается окно баланса, где вычисляются ёмкости и остатки для всех магазинов и складов и заполняется таблица, после чего управление вновь возвращается `MainUnit`.

Создание отгрузки через соответствующую кнопку запускает валидацию введённых данных, формирует новый узел `PShipment`, добавляет его в связный список отгрузок и регистрирует стрелку через `AddArrow`.

Кнопка «Выполнить все» последовательно перебирает все отгрузки, для каждой вызывает `doShipment` — удаляет стрелки, корректирует деревья товаров и ёмкости пунктов, освобождает память узла и сбрасывает флаг отметки. После полного выполнения отображается результат через `showMessage`, таблица отгрузок обновляется, и программа возвращается к ожиданию. Аналогично, при выборе выполнения лишь отдельных отгрузок открывается форма с чекбоксами, где отмеченные строки обрабатываются функцией `btnSelectConfirmClick`, выполняются через `doShipment`, а затем список снова перерисовывается.

Добавление товара инициируется через панель ввода, где после валидации создаётся новый узел `PItem` и вставляется в дерево существующих товаров при помощи `InsertTreapItem`.

Просмотр всех отгрузок открывает форму с таблицей, заполняемой процедурой `LoadData` из модуля `ShipmentsTableUnit`, а наведение курсора на строку или на соответствующую стрелку на карте синхронно подсвечивает связанные элементы.

Редактирование или удаление существующих объектов вызывают соответствующие панели, используют `EraseTreapName`, `EraseTreap` и `EraseTreapItem` для корректного удаления из деревьев.

При попытке закрыть главное окно (`FormClose`) проверяется флаг сохранения: если данные изменены, пользователю предлагается подтвердить или отказаться от сохранения через `getConfirmation` и, при согласии, выполнить процедуру сохранения. После этого приложение завершает цикл ожидания и закрывается. `MainUnit` выступает координатором: оно вызывает специализированные модули для хеширования, генерации ключей, фильтрации, валидации, управления отгрузками, отрисовки таблиц и запуска диалогов, обеспечивая целостную работу всей системы.

3.2 Проектирование алгоритмов работы с декартовыми деревьями

InitTree ставит корень дерева в значение nil, дерево готово к работе.

Создание нового узла дерева выполняется функцией CreateNewNode, которая выделяет память под структуру PTreapNode, сохраняет в ней указатель на данные (PLocation), обнуляет ссылки на левого и правого потомка и генерирует случайный приоритет.

Поиск узла по ключу происходит рекурсивно в функции FindTreap: если текущий корень равен nil, возвращается nil; если ключ совпадает — возвращается этот узел; если ключ меньше, поиск продолжается в левом поддереве, иначе в правом.

Разделение дерева на два по заданному ключу осуществляется процедурой SplitTreap. Если дерево пусто, оба выходных поддерева получают nil. В противном случае, если ключ корня меньше порогового, рекурсивно разбивается правое поддерево — полученные левые части присоединяются к исходному корню, иначе аналогично обрабатывается левое поддерево.

Слияние двух корректных деревьев (MergeTreap) выполняется сравнением приоритетов корней: корень с большим приоритетом становится корнем результата, а второе дерево рекурсивно встраивается в соответствующее под-дерево; при равных приоритетах дополнительным критерием выбирается меньший ключ. Если одно из деревьев пусто, возвращается другое.

Вставка нового узла (InsertTreap) проверяет, пуст ли корень: если да, делает SplitTreap по ключу нового узла, присоединяет полученные поддерева к его левому и правому детям и устанавливает его в корень. Если корневой узел уже есть, рекурсивно спускается в левое или правое поддерево в зависимости от сравнения ключей.

Удаление узла по ключу производится процедурой EraseTreap: происходит поиск узла аналогично функции поиска, и когда целевой узел найден, его левое и правое поддерева соединяются через MergeTreap, исходный узел освобождается, а результат присоединяется к родительскому указателю.

Полная очистка дерева (ClearTreap) реализована через рекурсивный обход: сначала очищаются левые и правые поддерева, затем для каждого узла удаляются связанные с ним данные (деревья товаров, списки стрелок, графические объекты) и сама структура узла, после чего корень устанавливается в nil.

Декартовы деревья, реализованные в модулях CartesianTreeByName и CartesianTreeItem реализованы таким же образом, но хранят в своих вершинах другие структуры.

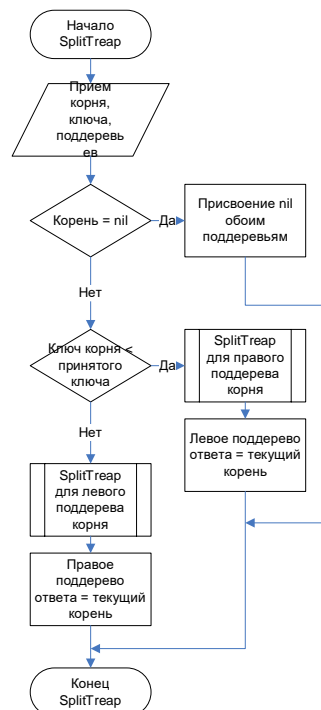


Рисунок 3.1 – Процедура SplitTgear

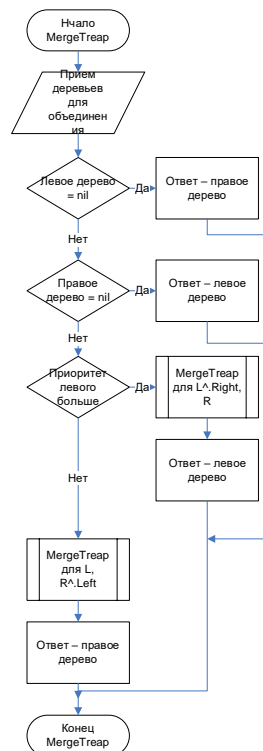


Рисунок 3.2 – Функция MergeTgear

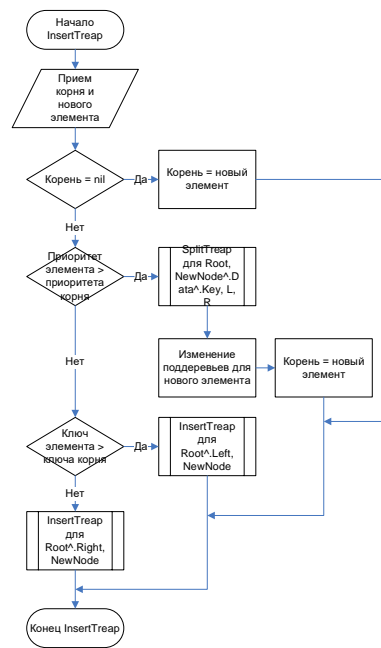


Рисунок 3.3 – Процедура InsertTreap

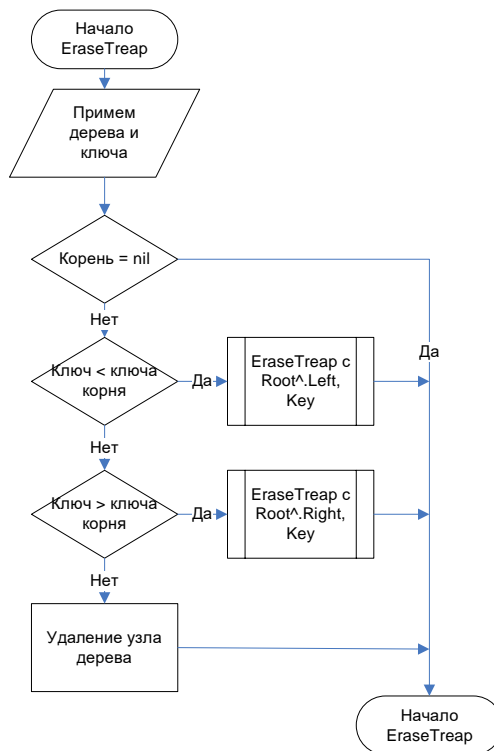


Рисунок 3.4 – Процедура EraseTreap

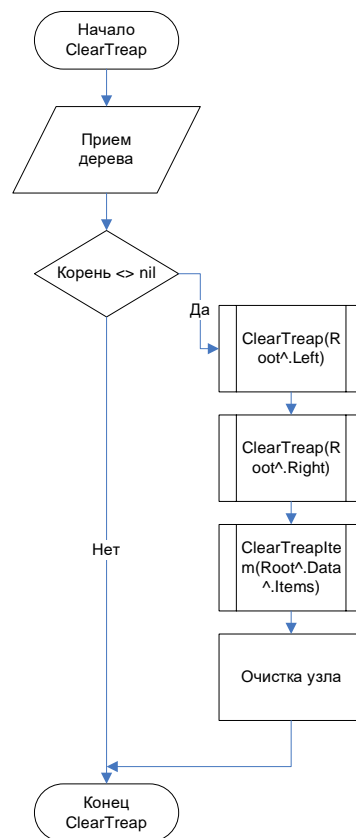


Рисунок 3.5 – Процедура ClearTreap

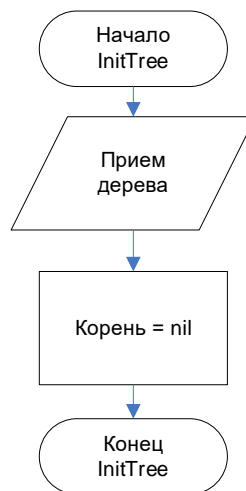


Рисунок 3.6 – Процедура InitTree

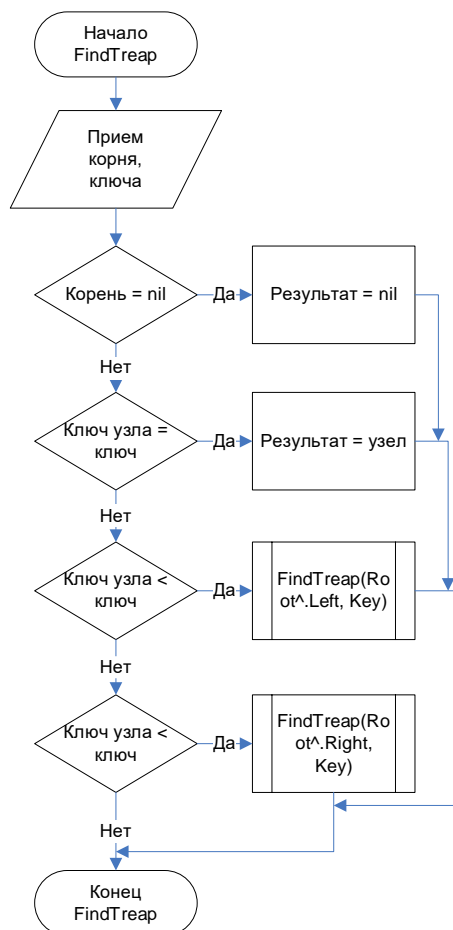


Рисунок 3.7 – Функция FindTreap



Рисунок 3.8 – Функция CreateNewNode

3.3 Проектирование алгоритмов хеширования

Процедура `initHash` — получает два целых параметра (основание и модуль), сохраняет их во внутренних переменных и один раз заполняет массив степеней основания по модулю, чтобы ускорить последующие вычисления.

Функция `getHash` — принимает строку, последовательно обрабатывает каждый символ, умножая его код на соответствующее заранее вычисленное значение из массива и аккумулируя результат с учётом модуля, после чего возвращает готовый хеш.

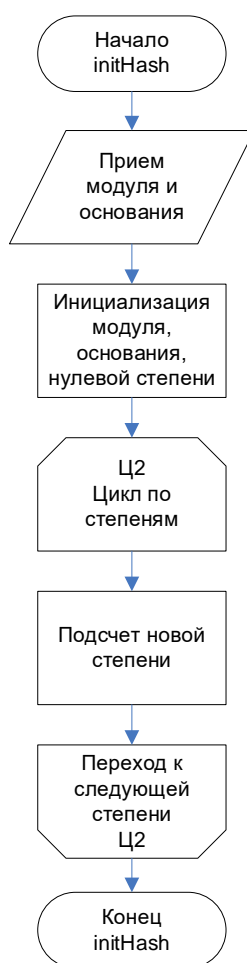


Рисунок 3.9 – Процедура `initHash`

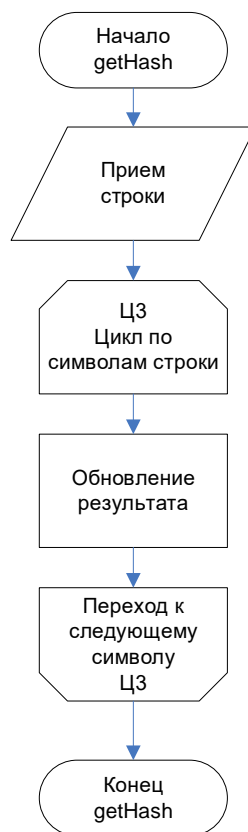


Рисунок 3.10 – Функция getHash

3.4 Проектирование алгоритмов фильтрации

Процедура `InitFilter` — устанавливает фильтр в начальное состояние, разрешая все типы объектов, сбрасывая строковые поля в пустую строку и числовые в -1 .

Процедура `CreateFilter` — принимает параметры типа объектов, улицы, дома, корпуса и границ по вместимости и занятости, записывает их в структуру фильтра для последующего применения.

Процедура `ApplyFilter` — обходит все узлы декартового дерева, сравнивает каждый объект с заданными критериями фильтра и устанавливает видимость фигур на основе результата, затем рекурсивно вызывает себя для левого и правого поддеревьев.

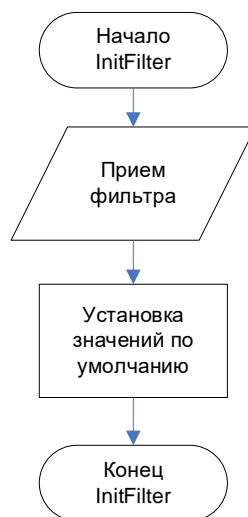


Рисунок 3.11 – Процедура InitFilter

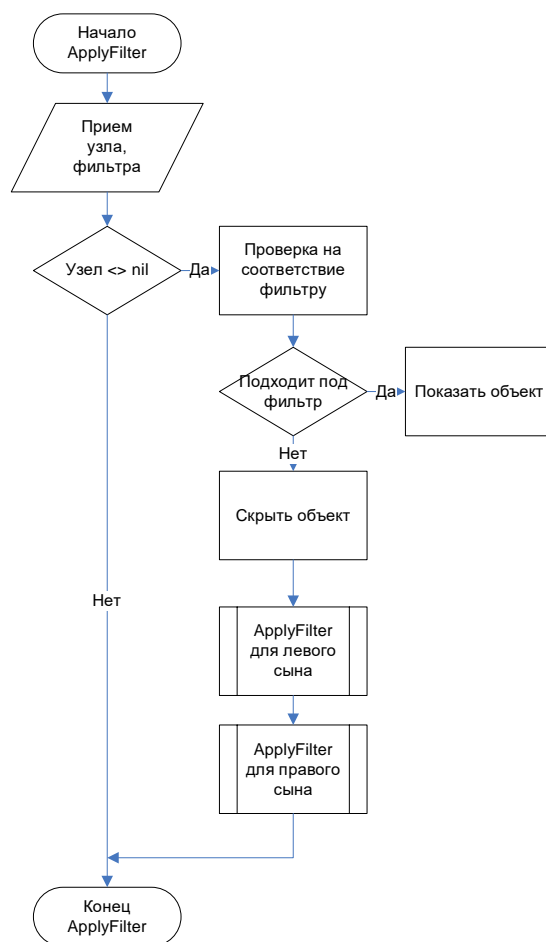


Рисунок 3.12 – Процедура ApplyFilter

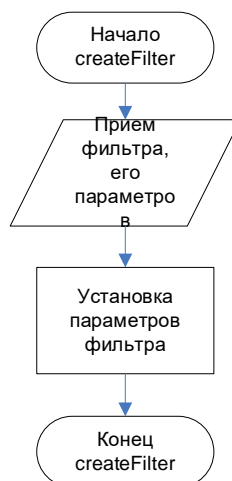


Рисунок 3.13 – Процедура createFilter

3.5 Проектирование алгоритмов валидации

Процедура `validateLengthLess70` — отслеживает текст в поле ввода, и если его длина превышает 70 символов, удаляет последний введенный символ и выводит окно с предупреждением о максимально допустимой длине.

Процедура `validateIntegerInput` — обрабатывает ввод числового значения в поле: удаляет ведущие нули, ограничивает длину десятью цифрами, не допускает превышения максимального целого, и при нарушении условий корректирует текст и выводит предупреждающее сообщение.

Функция `validateLength` — убирает ведущие пробелы, устанавливает курсор в конец текста и проверяет, что поле не пустое; если пустое, окрашивает фон в красный и возвращает `False`, иначе — `True`.

Функция `validateLetters` — проверяет каждый символ текста, разрешая только буквы (латинские и кириллические), цифры и пробел; при обнаружении постороннего символа окрашивает фон в красный и возвращает `False`, иначе — `True`.

Функция `validateAll` — последовательно вызывает `validateLength` и `validateLetters` для одного поля ввода и возвращает `True` только если обе проверки прошли успешно.

Функция `validateFromTo` — сравнивает два поля ввода с числовыми значениями и возвращает `False`, если оба заполнены и значение первого больше второго, в противном случае — `True`.

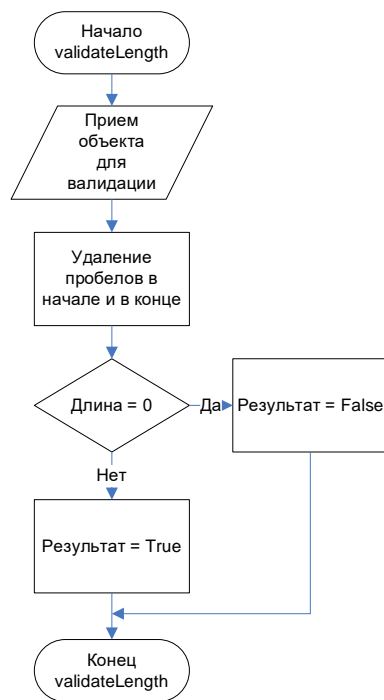


Рисунок 3.14 – Функция `validateLength`

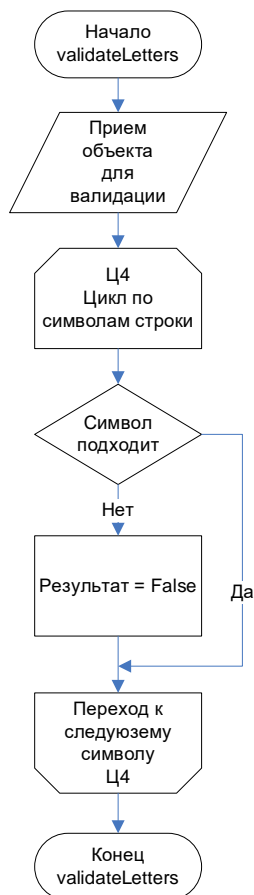


Рисунок 3.15 – Функция `validateLetters`

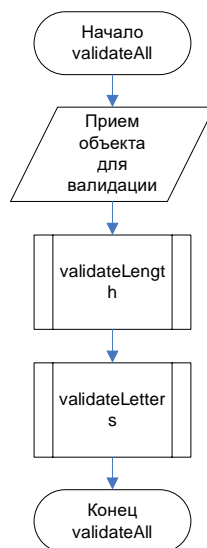


Рисунок 3.16 – Функция `validateAll`

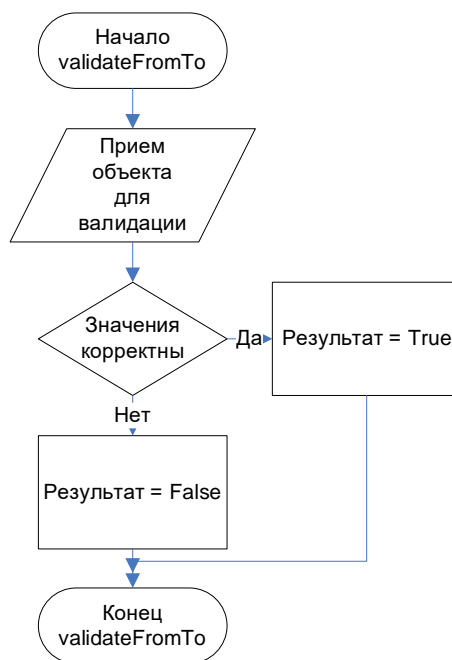


Рисунок 3.17 – Функция `validateFromTo`

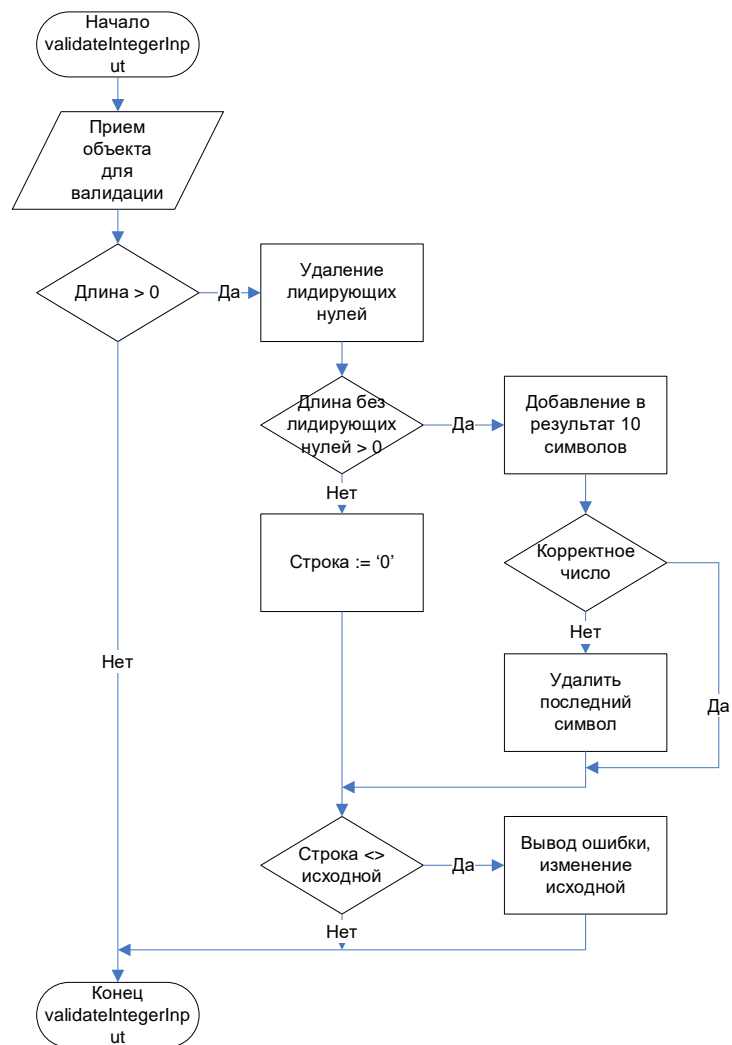


Рисунок 3.18 – Процедура validateIntegerInput

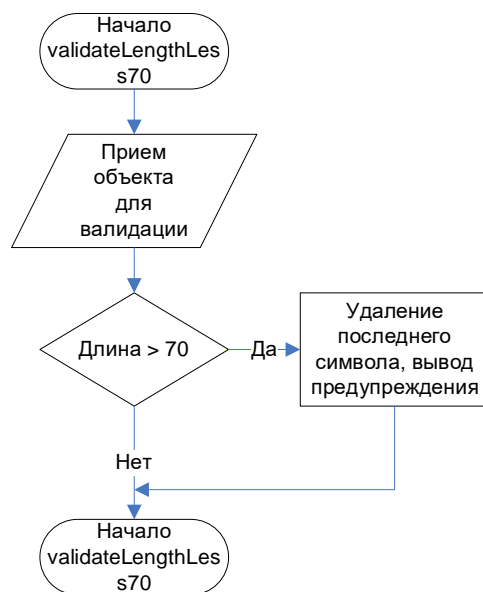


Рисунок 3.19 – Процедура validateLengthLess70

3.6 Проектирование алгоритмов управления отгрузками

Процедура `ClearShipments` — принимает указатель на начало односвязного списка отгрузок и последовательно освобождает память каждого узла: сохраняет текущий узел во временную переменную, переводит указатель на следующий элемент и удаляет освобождённый узел, пока список не окажется пустым.

Функция `doShipment` — выполняет одну отгрузку, сначала удаляя из списков стрелок отправителя, получателя и из глобального списка `Arrows` все ссылки на текущую отгрузку, затем находит в деревьях товаров узлы отправления и назначения. Если в пункте назначения товара нет, создаётся новый узел с теми же свойствами, что и у отправителя, и вставляется в его дерево. После этого корректируются счётчики объёма и количества: у отправителя уменьшаются показатели запланированной отправки, занятого пространства и остаточного числа товаров, у получателя — резерв и занятое пространство, при необходимости удаляется пустой узел отправителя. В случае любой ошибки функция возвращает `False`, иначе возвращает `True`.

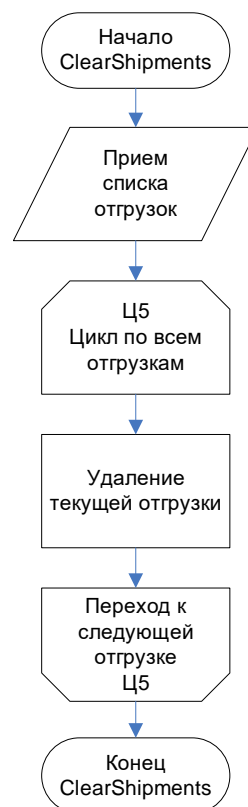


Рисунок 3.20 – Процедура `ClearShipments`

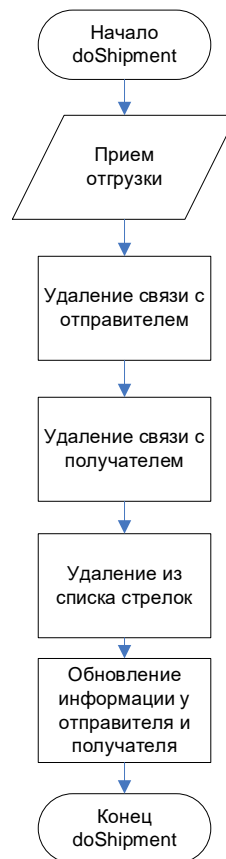


Рисунок 3.21 – Функция doShipment

3.7 Проектирование алгоритмов управления стрелками

Процедура AddArrow — создаёт новый объект стрелки, связывает его с переданной отгрузкой, определяет его видимость по состоянию отправителя и получателя, добавляет в глобальный список Arrows и в списки OutgoingArrows отправителя и IncomingArrows получателя, создавая их при необходимости.

Функция IsPointNearLine — проверяет, лежит ли точка в пределах отрезка между двумя точками, сначала убеждаясь, что она попадает в прямоугольник, ограничивающий отрезок, а затем оценивая расстояние от точки до линии и сравнивая его с допуском.

Процедура RemoveArrow — удаляет указанную стрелку из списков OutgoingArrows отправителя, IncomingArrows получателя и глобального списка Arrows, после чего освобождает память структуры стрелки.

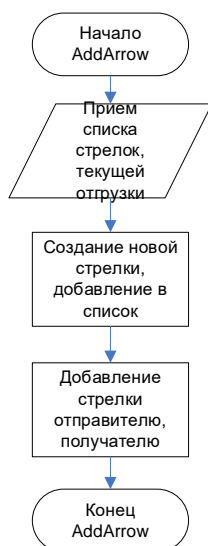


Рисунок 3.22 – Процедура AddArrow

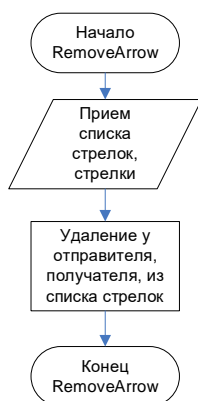


Рисунок 3.23 – Процедура RemoveArrow

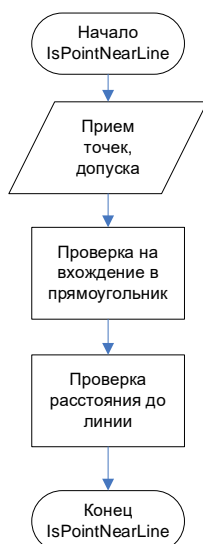


Рисунок 3.24 – Функция IsPointNearLine

3.8 Проектирование алгоритмов диалогового взаимодействия с пользователем

Функция `getConfirmation` — создаёт диалог подтверждения с кнопками «Да» и «Нет», устанавливает заголовок и текст, переводит стандартные подписи кнопок на русский, отображает окно и возвращает `True`, если пользователь выбрал «Да».

Процедура `showMessage` — создаёт информационное окно с кнопкой «Ок», устанавливает заголовок и текст, переименовывает кнопку в «Ок», показывает модальное окно и освобождает ресурсы по закрытию.

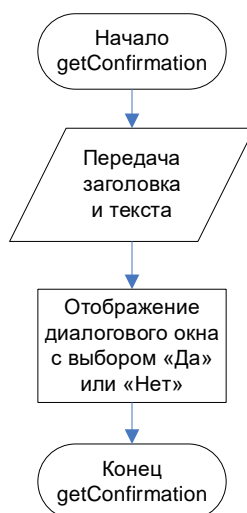


Рисунок 3.25 – Функция `getConfirmation`

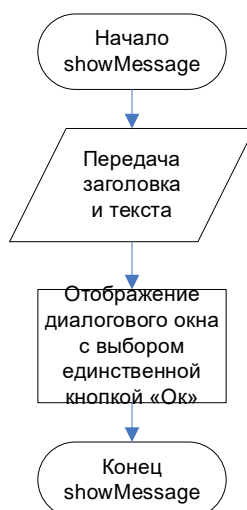


Рисунок 3.26 – Процедура `showMessage`

4. КОНСТРУИРОВАНИЕ ПРОГРАММНОГО СРЕДСТВА

4.1. Структура программного средства

На основании функциональных требований из предыдущих разделов разрабатываемое программное средство состоит из следующих модулей и форм:

Модуль `CartesianTree` – содержит реализацию декартового дерева, которое хранит объекты (склады/магазины). Поиск в дереве производится по идентификатору объекта.

Модуль `CartesianTreeByName` – содержит реализацию декартового дерева, которое хранит имена существующих складов/магазинов. Поиск в дереве производится по хешу названия объекта.

Модуль `CartesianTreeItem` – содержит реализацию декартового дерева, которое хранит товары в конкретном объекте. Поиск в дереве производится по хешу названия товара.

Модуль `Filter` – Содержит подпрограммы, которые отвечают за создание фильтра по определенным параметрам и применение этого фильтра к объектам.

Модуль `GetKeys` – Содержит актуальные идентификаторы для нового объекта. Отвечает за получение и изменение уникальных идентификаторов.

Модуль `Hash` – Содержит подпрограммы для получения хеша какой-либо строки.

Модуль `Messages` – Содержит подпрограммы, которые можно вызвать для отображения на экране уведомлений. Всего 2 варианта уведомлений. Первый – с возможностью выбора да/нет. Второй – с единственной кнопкой «Ок».

Модуль `shipments` – Содержит подпрограммы, отвечающие за выполнение конкретной отгрузки и очистки всех отгрузок.

Модуль `Validation` – Содержит подпрограммы, которые отвечают за проверку введенных данных.

Модуль `ArrowsUnit` – Содержит подпрограммы, которые позволяют добавлять линии, удалять линии, проверять, находится ли заданная точка рядом с линией.

Модуль `Types` – Содержит множество типов, используемых в различных модулях.

Модуль `Vars` – Содержит множество необходимых переменных, используемых в различных модулях.

Форма `MainUnit` – Содержит основную логику программного средства.

Форма BalanceUnit – Форма, показывающая остаток всех товаров во всех объектах.

Форма SelectShipmentsUnit – Форма, позволяющая выбирать отдельные отгрузки и выполнять их.

Форма ShipmentsTableUnit – Форма, отображающая на экране все существующие отгрузки.

Форма TableUnit – Форма, отображающая список всех товаров, которые хранятся в определенном объекте.

4.2 Разработка модуля BalanceUnit

4.2.2 FormClose. При получении события закрытия формы выполняет освобождение ресурсов самой формы, устанавливая действие на saFree, чтобы окно было уничтожено.

4.2.3 FormCreate. При создании формы сохраняет её начальные размеры, настраивает таблицу, задаёт число столбцов, вычисляет их ширины в зависимости от ширины окна, заполняет заголовки (имя объекта, тип, вместимость и т. д.) и отключает штатную отрисовку ячеек, чтобы дальше управлять её содержимым вручную.

4.2.4 SetData. Получив два корня деревьев (магазины и склады), подсчитывает общее число товарных записей, настраивает количество строк таблицы и затем последовательно проходит по всем узлам этих деревьев, выводя для каждого объекта и каждого его товара соответствующие значения в строки таблицы.

4.2.5 getSizObject. Рекурсивно обходит дерево объектов и суммирует количество всех вложенных товарных узлов, возвращая общее число товарных записей в данном поддереве.

4.2.6 getSizItems. Рекурсивно обходит дерево товарных узлов и считает общее число элементов, возвращая количество узлов в поддереве товаров.

4.2.7 showDataObject. Для переданного узла объекта (склад или магазин) вызывает вывод товарных записей: сначала обрабатывает все товары данного объекта, затем рекурсивно переходит к левому и правому поддеревьям объектов, постепенно заполняя таблицу строками.

4.2.8 showDataItem. Для одного товарного узла записывает в текущую строку таблицы имя родительского объекта, его тип, вместимость, занятое и зарезервированное место, а затем детали товара – название, категорию, артикул, количество и итоговый объём. После записи строки переходит к левому и правому потомкам, увеличивая счётчик строк.

4.2.9 FormResize. При попытке изменить размер окна сразу возвращает форму к тем размерам, которые были при создании, тем самым фиксируя её статический размер.

4.2.10 sgBalanceTableDrawCell. При отрисовке каждой ячейки вручную заполняет фон и выводит текст из таблицы, обеспечивая перенос слов и выравнивание по левому краю без использования штатной механики компонента.

4.3 Разработка модуля CartesianTree

4.3.1 InitTree. Принимает указатель на корень дерева (переменную) и инициализирует его, устанавливая в `nil`, то есть создаёт пустое декартово дерево.

4.3.2 CreateNewNode. Принимает указатель на данные (структуру с информацией об объекте) и создаёт новый узел дерева: выделяет память, сохраняет данные, обнуляет ссылки на дочерние узлы и генерирует случайный приоритет, необходимый для балансировки.

4.3.3 FindTreap. Принимает корень дерева и ключ, затем рекурсивно ищет узел с указанным ключом: сравнивает значение ключа в текущем узле и спускается в левое или правое поддерево до тех пор, пока не найдёт нужный узел либо не дойдёт до конца дерева. Возвращает либо найденный узел, либо `nil`, если ключ отсутствует.

4.3.4 SplitTreap. Принимает корень дерева и пороговый ключ, а также две переменные для корней выходных поддеревьев. Разделяет исходное дерево на две части: в одной оказываются все узлы с ключом меньше порога, в другой – все остальные. Делается это рекурсивно, корректно перенастраивая ссылки потомков.

4.3.5 MergeTreap. Принимает два корня деревьев (левого и правого) и объединяет их в одно корректное декартово дерево, соблюдая приоритеты

узлов. Если одно из деревьев пусто, сразу возвращает другое; иначе сравнивает приоритеты корней и рекурсивно встраивает меньший поддереву в соответствующее место.

4.3.6 InsertTreap. Принимает корень дерева и новый узел для вставки. Если дерево пусто, сразу подкладывает новый узел и выполняет разбиение по ключу. Иначе спускается по дереву «слева» или «справа», сравнивая ключи до тех пор, пока не найдёт место для вставки нового узла. Так гарантируется сохранение свойств бинарного поиска по ключу.

4.3.7 EraseTreap. Принимает корень дерева и ключ узла, который нужно удалить. Рекурсивно находит узел: если ключ меньше или больше текущего, идёт в соответствующее поддерево; если равен – объединяет левое и правое поддерева удаляемого узла с помощью MergeTreap, освобождает память и переподключает результат к родительскому указателю.

4.3.8 ClearTreap. Принимает корень дерева и рекурсивно очищает всё дерево: сначала обходит левые и правые поддерева, вызывая ClearTreap для каждого узла, затем освобождает ресурсы, связанные с данными узла (списки товаров, списки стрелок, графические элементы) и сам узел, после чего устанавливает корень в nil.

4.4 Разработка модуля CartesianTreeByName

4.4.1 InitTreeName. Принимает переменную-указатель на корень дерева имен и инициализирует его, устанавливая в nil, создавая пустое дерево.

4.4.2 CreateNewNameNode. Принимает строку с именем и целочисленный идентификатор; вычисляет хеш от имени, создаёт новый узел, сохраняет имя, ключ (хеш), ID и случайный приоритет, после чего возвращает указатель на созданный узел.

4.4.3 FindTreapName. Принимает корень дерева имен и ключ (хеш-значение). Рекурсивно сравнивает ключи в узлах, спускаясь в левое или правое поддерево. Возвращает указатель на узел с данным ключом или nil, если узел не найден.

4.4.4 SplitTreapName. Принимает корень дерева имен и пороговый ключ, а также две переменные для выходных поддеревьев. Делит исходное дерево на две части: в левом поддереве окажутся узлы с ключами меньше порога, в

правом — все остальные. Присваивает полученные корни переданным переменным.

4.4.5 MergeTreapName. Принимает два корня деревьев имен (левое и правое). Объединяет их в одно сбалансированное дерево, сохраняя свойства декартового дерева: выбирает в качестве нового корня узел с бóльшим приоритетом и рекурсивно соединяет поддеревья. Возвращает корень объединённого дерева.

4.4.6 InsertTreapName. Принимает корень дерева имен и новый узел. Если дерево пусто, разделяет его по ключу и вставляет узел. Иначе рекурсивно спускается в левое или правое поддерево, сравнивая ключи, и в конечном счёте вставляет узел так, чтобы сохранить свойства бинарного поиска.

4.4.7 EraseTreapName. Принимает корень дерева имен и ключ удаляемого узла. Рекурсивно ищет узел с указанным ключом: если текущее значение меньше или больше, уходит в соответствующее поддерево; если ключ совпал, объединяет два поддерева удаляемого узла через MergeTreapName, освобождает память узла и переподключает результат к родительскому указателю.

4.4.8 ClearTreapName. Принимает корень дерева имен и полностью освобождает память всего дерева: рекурсивно очищает левое и правое поддеревья, затем удаляет текущий узел и устанавливает корень в nil.

4.5 Разработка модуля CartesianTreeItem

4.5.1 InitTreeItem. Принимает ссылку на переменную, содержащую корень дерева товарных узлов, и обнуляет её, то есть после выполнения этого метода дерево считается пустым.

4.5.2 CreateNewItemNode. Получает указатель на структуру данных PItem (с информацией о конкретном товаре), выделяет память для нового узла, сохраняет в нём переданные данные, устанавливает ссылки на левого и правого потомков в nil и генерирует случайный приоритет. В результате возвращается указатель на созданный узел, готовый для вставки в дерево.

4.5.3 FindTreapItem. Принимает корень дерева товарных узлов и ключ (целочисленное значение). Метод рекурсивно сравнивает переданный ключ с ключом текущего узла: если совпадает, возвращает этот узел; если меньше —

продолжает поиск в левом поддереве, иначе — в правом. Если узел с таким ключом не найден, возвращается `nil`.

4.5.4 SplitTreapItem. Принимает корень исходного дерева товаров и пороговый ключ, а также две переменные-указателя для выходных деревьев (`L` и `R`). В результате выполнения все узлы с ключами меньше порогового оказываются в дереве `L`, а узлы с ключами, большими или равными порогу, — в дереве `R`. При этом рекурсивно перенастраиваются указатели на потомков, чтобы оба полученных поддерева сохранили корректную структуру.

4.5.5 MergeTreapItem. Получает два небнутых корня деревьев товаров (`L` и `R`), причём гарантируется, что все ключи в `L` меньше ключей в `R`. Если одно из деревьев пусто, сразу возвращается указатель на другое. В противном случае сравниваются приоритеты корней: узел с более высоким приоритетом становится корнем результирующего дерева, а второе дерево рекурсивно встраивается в соответствующее поддерево этого корня. В результате возвращается указатель на корень нового объединённого дерева.

4.5.6 InsertTreapItem. Получает корень дерева товаров и указатель на новый узел для вставки. Если исходное дерево пусто, вызывается `SplitTreapItem` (чтобы разделить `nil` по ключу нового узла), затем новый узел сразу становится корнем с подходящими левым и правым ссылками. Если дерево непусто, метод сравнивает ключ нового узла и ключ текущего узла и рекурсивно спускается в левое или правое поддерево, пока не найдёт место для вставки, тем самым сохраняя свойства двоичного поиска.

4.5.7 EraseTreapItem. Принимает корень дерева товаров и ключ удаляемого узла. Он рекурсивно ищет узел с данным ключом: если ключ меньше или больше текущего, продолжает спускаться соответственно в левое или правое поддерево. Как только находит узел с совпадающим ключом, объединяет его левое и правое поддерева с помощью `MergeTreapItem`, освобождает память удаляемого узла и переподключает результат объединения к родительскому указателю, тем самым сохраняя балансировку дерева.

4.5.8 ClearTreapItem. Получает корень дерева товаров и полностью очищает его: сначала рекурсивно удаляет все узлы левого и правого поддерева, затем освобождает память текущего узла и устанавливает корень в `nil`, что означает, что дерево полностью очищено.

4.6 Разработка модуля Filter

4.6.1 InitFilter. Устанавливает фильтр в начальное состояние без ограничений (оба типа объектов разрешены, числовые поля = -1, строковые – пустые).

4.6.2 CreateFilter. Записывает в фильтр заданные значения типа объектов, улицы, дома, строения и диапазонов вместимости и занятой площади.

4.6.3 ApplyFilter. Обходит все узлы дерева, сравнивает их атрибуты (тип, улица, дом, строение, вместимость, занятость) с фильтром и скрывает те, что не соответствуют.

4.7 Разработка модуля GetKeys

4.7.1 getShopKey. Возвращает текущее значение глобальной переменной shopKey и увеличивает её на единицу.

4.7.2 getWarehouseKey. Возвращает текущее значение глобальной переменной warehouseKey и увеличивает её на единицу.

4.8 Разработка модуля Hash

4.8.1 initHash. Принимает два целых числа (p и m), сохраняет их и заполняет массив степеней $p \pmod m$ от 0 до верхней границы.

4.8.2 getHash. Принимает строку, для каждого символа умножает его код на соответствующую степень p , суммирует все результаты по модулю m и возвращает итоговое значение.

4.9 Разработка модуля MainUnit

4.9.1 FormClose. При попытке закрыть форму проверяет флаг Saved. Если данные не сохранены, показывает диалог подтверждения (вопрос «Сохранить изменения?»). При положительном ответе вызывает метод сохранения (Save1Click).

4.9.2 FormCreate. В момент создания формы выполняет инициализацию: запоминает исходные размеры окна, инициализирует декартовы деревья, списки отгрузок и стрелок.

4.9.3 FormDestroy. При разрушении формы освобождает всю загруженную в ходе работы память, вызывая ClearAllData для деревьев магазинов, складов, отгрузок, списков имён и стрелок.

4.9.4 FormResize. При изменении размеров окна сбрасывает ширину и высоту клиентской области к исходным значениям, зафиксированным в FormCreate.

4.9.5 btnFilterClick. Открывает панель установки фильтра (делает видимым блок pnFilter), заполняет поля текущими значениями фильтра.

4.9.6 btnFilterConfirmClick. Считывает значения из элементов управления на панели фильтра и вызывает CreateFilter, затем применяет фильтр ко всем узлам дерева через ApplyFilter, скрывая неподходящие объекты.

4.9.7 btnFilterCancelClick. Закрывает панель фильтра (скрывает pnFilter), не меняя текущие настройки.

4.9.8 btnFilterDefaultClick. Сбрасывает фильтр к исходному состоянию (все значения по умолчанию) через InitFilter и сразу применяет его.

4.9.9 btnCreateSelectShopClick. Открывает панель выбора типа создаваемого объекта и переключает режим на «Магазин» (ставит соответствующий radio-button).

4.9.10 btnCreateSelectWarehouseClick. То же, что и btnCreateSelectShopClick, но переключается на «Склад».

4.9.11 btnCreateSelectCancelClick. Закрывает панель выбора типа создаваемого объекта, не переходя к вводу данных.

4.9.12 btnCreateObjConfirmClick. Считывает введённые пользователем данные (название, адрес, вместимость), проверяет их корректность через модуль Validation. Затем получает новый ключ (getShopKey или getWarehouseKey), формирует структуру данных, создаёт новый узел и

вставляет его в соответствующее дерево (InsertTreapName + InsertTreapItem или только InsertTreapItem). Обновляет визуализацию.

4.9.13 btnCreateObjCancelClick. Отменяет создание объекта, очищает поля ввода и скрывает панель pnCreateObj.

4.9.14 btnSelectObjEditClick. При выборе существующего объекта (магазин или склад) открывает панель редактирования, заполняя её полями текущих значений из узла дерева.

4.9.15 btnEditObjConfirmClick. После правки сохраняет изменённые поля обратно в структуру узла: обновляет имя, адрес, вместимость. Обновляет отображение на карте и в списках.

4.9.16 btnEditObjCancelClick. Отменяет редактирование, сбрасывает изменения в поле ввода и закрывает панель pnEditObj.

4.9.17 btnSelectObjDeleteClick. Удаляет выбранный объект из дерева: ищет узел по ключу и вызывает EraseTreapName (для имён) и EraseTreapItem (для самого узла), освобождает память, перестраивает дерево.

4.9.18 btnSelectObjItemListClick. Открывает окно со списком отгрузок, в которых участвует выбранный объект (магазин или склад), формируя фильтр по ID объекта.

4.9.19 ClearAddItem. Сбрасывает все поля на панели «Добавить товар» (pnAddItem): очищает текстовые поля, сбрасывает цвета, переключает тип приёма на «Магазин» по умолчанию.

4.9.20 btnAddItemConfirmClick. Валидация введённых данных (название товара, количество, объём, пункт приёма/назначения). Создаёт новую структуру PItem, назначает приоритет, вставляет узел в дерево shipments через InsertTreapItem.

4.9.21 btnAddItemCancelClick. Отменяет добавление товара, очищает поля и скрывает панель «Добавить товар».

4.9.22 ClearCreateShipment. Сбрасывает поля панели «Создать отгрузку»: очищает ID и наименования отправителя и получателя, товарные данные, переключает радиокнопки по умолчанию.

4.9.23 btnCreateShipmentConfirmClick. Валидация данных отправителя, получателя, товара и количества. Формирует новую запись «отгрузка», вставляет её в структуру данных (специальное дерево отгрузок).

4.9.24 btnCreateShipmentCancelClick. Отменяет ввод новой отгрузки, очищает все поля и скрывает панель.

4.9.25 updateID. Утилита для полей на формах: принимает поле ввода ID (TEdit), группу радиокнопок (магазин/склад) и поле имени. По тексту в поле имени вычисляет хеш, ищет узел в дереве имён (FindTreapName) и, при нахождении, записывает в ID-поле строку ключа (с учётом маски); иначе очищает поле.

4.9.26 updateName. Обратная утилита: по вводу ID (число) и группе радиокнопок ищет узел в основном дереве (FindTreap) и, если найден, подставляет в поле имени значение Data^.name; иначе очищает имя.

4.9.27 rbAddItemTypeShopClick / rbAddItemTypeWarehouseClick. При переключении типа пункта (магазин/склад) на форме «Добавить товар» вызывают updateID или updateName для соответствующего поля, чтобы синхронизировать ID и имя пункта.

4.9.28 Остальные подпрограммы. Все остальные обработчики событий (кликов по кнопкам и переключения радиокнопок) реализованы аналогично: они либо открывают/скрывают панели, либо очищают поля, либо вызывают соответствующие вспомогательные процедуры валидации, генерации ключей и работы с деревьями.

4.10 Разработка модуля Messages

4.10.1 getConfirmation. Принимает заголовок (capt) и текст сообщения (text). Создаёт диалоговое окно типа подтверждения с кнопками «Да» и «Нет» на русском языке, устанавливает заголовок, отображает переданный текст. Ждёт ответа пользователя и возвращает True, если был выбран «Да», и False — если «Нет». После выбора освобождает ресурсы формы.

4.10.2 showMessage. Принимает заголовок (capt) и текст сообщения (text). Создаёт информационное окно с единственной кнопкой «Ок» на

русском языке, устанавливает заголовок, выводит текст. После нажатия «Ок» (или любого закрытия) освобождает ресурсы формы.

4.11 Разработка модуля SelectShipmentsUnit

4.11.1 FormCreate. Запоминает исходные размеры формы, настраивает сетку sgSelectShipmentsTable (число и ширины колонок, заголовки, двойная буферизация), отключает стандартную отрисовку и назначает обработчики.

4.11.2 FormClose. При закрытии устанавливает Action := saFree, чтобы форма освобождала память.

4.11.3 FormResize. При изменении размеров возвращает клиентскую область к исходным startWidth/startHeight.

4.11.4 LoadData. Сохраняет указатель на список отгрузок в FShipmentsPtr, вычисляет число записей, настраивает RowCount и заполняет строки данными (Название, ID, Отправитель, Адреса, Товар, Хеш, Количество).

4.11.5 sgSelectShipmentsTableDrawCell. Кастомная отрисовка ячеек: в столбце чекбоксов рисует рамку и «+» при отмеченном состоянии, в остальных выводит текст с переносами.

4.11.6 sgSelectShipmentsTableMouseDown. При клике в области чекбокса текущей ячейки вызывает ToggleCheckbox.

4.11.7 ToggleCheckbox. Переключает значение в ячейке чекбокса ('0'↔'1') и перерисовывает форму.

4.11.8 btnSelectAllClick. Устанавливает во всех строках (кроме заголовка) флажки в '1' и перерисовывает форму.

4.11.9 btnSelectResetClick. Сбрасывает во всех строках флажки в '0' и перерисовывает форму.

4.11.10 btnSelectConfirmClick. Для каждой отмеченной отгрузки вызывает doShipment, удаляет узел из списка, освобождает память, показывает итог через showMessage («Успешно»/«Ошибка»), обновляет таблицу и перерисовывает форму.

4.12 Разработка модуля Shipments

4.12.1 ClearShipments. Очищает список отгрузок: пока указатель shipment ненулевой, сохраняет текущий элемент в prev, двигает shipment на следующий узел и освобождает память prev.

4.12.2 doShipment. Выполняет отгрузку shipment: удаляет ссылки на неё из списков стрелок отправителя, получателя и глобального списка Arrows; находит или создаёт узел товара в дереве пункта назначения; корректирует счётчики needToSend, usedCapacity и Count в узлах отправителя и получателя; при пустом остатке удаляет узел из дерева отправителя; отмечает данные как несохранённые; возвращает True при успешном выполнении, False при ошибке.

4.13 Разработка модуля ShipmentsTableUnit

4.13.1 FormCreate. Запоминает исходные размеры формы, настраивает стиль и позицию окна, задаёт число и ширины столбцов сетки sgShipmentsTable, заполняет заголовки и отключает стандартную отрисовку.

4.13.2 FormClose. При закрытии устанавливает Action := saFree для освобождения памяти формы.

4.13.3 FormResize. При изменении размеров возвращает клиентскую область к исходным startWidth/startHeight.

4.13.4 LoadData. Подсчитывает число узлов в списке shipment, устанавливает RowCount, затем заполняет строки: название отгрузки, ID, имя отправителя, адрес отправителя, имя получателя, адрес получателя, наименование товара, артикул (хеш) и количество.

4.13.5 sgShipmentsTableDrawCell. Кастомная отрисовка ячейки: очищает фон и выводит текст с переносами по словам через DrawText.

4.14 Разработка модуля TableUnit

4.14.1 FormCreate. Запоминает исходные размеры формы, настраивает стиль и позицию окна, задаёт число и ширины столбцов сетки sgItemsTable, заполняет заголовки и отключает стандартную отрисовку.

4.14.2 FormClose. При закрытии устанавливает Action := saFree для освобождения памяти формы.

4.14.3 FormResize. При изменении размеров возвращает клиентскую область к исходным startWidth/startHeight.

4.14.4 GetTreeSize. Рекурсивно подсчитывает число узлов в дереве PTreapItemNode, возвращая 0 для nil или сумму 1 + размеры левого и правого поддерева.

4.14.5 SetDataToTable. Выполняет обход дерева в порядке корень-лево-право, в каждой строке заполняя ячейки: имя, категорию, объём, количество и ключ, увеличивая счётчик строк i.

4.14.6 LoadData. Определяет размер таблицы через GetTreeSize, устанавливает RowCount, задаёт заголовок панели pnItemsTableName с указанием «в магазине» или «на складе» и имени объекта, форматирует шрифт и вызывает SetDataToTable для заполнения строк.

4.14.7 sgItemsTableDrawCell. Кастомная отрисовка ячеек: очищает фон и выводит текст с переносами по словам через DrawText.

4.15 Разработка модуля Validation

4.15.1 validateLengthLess70. Проверяет длину текста в TEdit и если она превышает 70 символов, удаляет последний символ, возвращает курсор в конец и показывает сообщение «Длина строки должна быть не более 70 символов!».

4.15.2 validateAll. Вызывает validateLength и validateLetters, возвращает True только если обе проверки прошли успешно.

4.15.3 validateIntegerInput. При вводе числа удаляет ведущие нули, обеспечивая корректность; при нарушении условий обрезает последний символ, обновляет TEdit.Text и показывает сообщение об ошибке.

4.15.4 validateLetters. Перебирает символы текста, проверяет, что каждый — пробел, цифра, латинская/кириллическая буква (включая «ё»), иначе устанавливает TEdit.Color := clRed и возвращает False.

4.15.5 `validateLength`. Обрезает ведущие пробелы текста, устанавливает курсор в конец; если поле пустое, красит фон TEdit в красный и возвращает False.

4.15.6 `validateFromTo`. Сравнивает два TEdit: если оба заполнены и значение первого больше второго, возвращает False, иначе — True.

4.16 Разработка модуля ArrowsUnit

4.16.1 `AddArrow`. Принимает список стрелок Arrows и указатель Shipment, создаёт новый узел PArrow, устанавливает ему поле shipment, вычисляет видимость (оба конца видимы → Visible := true, иначе false), добавляет в глобальный список Arrows и в списки OutgoingArrows у отправителя и IncomingArrows у получателя.

4.16.2 `RemoveArrow`. Принимает список стрелок arrowsList и указатель Arrow, удаляет Arrow из списков OutgoingArrows и IncomingArrows соответствующих пунктов, убирает из arrowsList и освобождает память узла.

4.16.3 `IsPointNearLine`. Принимает точку P, концы отрезка A, B и допуск Tolerance; проверяет попадание P в прямоугольник, ограниченный A–B, затем рассчитывает расстояние от точки до прямой и возвращает True, если оба условия выполняются.

Полный код программного средства размещен в Приложении Б.

5. ТЕСТИРОВАНИЕ, ПРОВЕРКА РАБОТОСПОСОБНОСТИ И АНАЛИЗ ПОЛУЧЕННЫХ РЕЗУЛЬТАТОВ

5.1 Описание тестов, результаты тестирования

Таблица 5.1 – Тестирование функционала программного средства

Специфика тестирования	Номер теста	Вводимые данные	Ожидаемый результат	Полученный результат
Запуск программы	1	Двойной щелчок левой кнопкой мыши по программе	Появление окна программы с картой	Тест пройден
Фильтрация	2	Нажатие на кнопку “фильтр” на главном экране	Появление окна со значениями для фильтра по центру экрана	Тест пройден
Создание магазина	3	Нажатие на кнопку "Создать магазин" и ввод имени, улицы, дома, емкости	Появление магазина на карте с соответствующими данными	Тест пройден
Создание склада	4	Нажатие на кнопку "Создать склад" и ввод имени, улицы, дома, емкости	Появление склада на карте с соответствующими данными	Тест пройден
Редактирование магазина	5	Выбор магазина для редактирования, изменение данных (имя, улица, дом)	Обновление данных магазина на карте	Тест пройден
Редактирование склада	6	Выбор склада для редактирования, изменение данных (имя, улица, дом)	Обновление данных склада на карте	Тест пройден

Продолжение таблицы 5.1

Валидация данных при создании объекта	7	Ввод неправильных данных (недопустимые символы в имени или улице)	Появление ошибки и запрет на создание объекта	Тест пройден
Валидация данных при фильтрации	8	Ввод некорректных значений в фильтре (нечисловые значения в поле «Вместимость»)	Появление сообщения об ошибке или запрет на фильтрацию с ошибочными данными	Тест пройден
Выделение объекта на карте	9	Наведение курсором по объекту (магазин/склад) на карте	Появление панели с подробной информацией о выбранном объекте	Тест пройден
Фильтрация объектов по типу	10	Установка фильтра «Только магазины» и нажатие кнопки «Применить»	Отображение только объектов типа «Магазин» на карте	Тест пройден
Фильтрация объектов по адресу	11	Ввод улицы и дома в фильтре, нажатие кнопки «Применить»	Отображение объектов, которые соответствуют фильтру по адресу	Тест пройден
Сброс фильтра	12	Нажатие на кнопку «Сброс»	Очистка всех фильтров, возврат к состоянию с отображением всех объектов	Тест пройден

Продолжение таблицы 5.1

Работа с полями фильтрации (интервал емкости)	13	Ввод минимального и максимального значения емкости в фильтре и нажатие кнопки «Применить»	Отображение объектов, которые соответствуют диапазону емкости	Тест пройден
Успешное создание объекта с уникальным именем	14	Ввод уникального имени для магазина/склада и нажатие кнопки «Подтвердить»	Создание нового объекта на карте с уникальным именем и соответствующими данными	Тест пройден
Проверка существующего имени объекта	15	Ввод существующего имени магазина/склада и попытка создать новый объект	Появление сообщения об ошибке и отказ в создании нового объекта	Тест пройден
Проверка обновления объектов после редактирования	16	Редактирование данных объекта (например, изменения вместимости склада)	Обновление отображаемых данных на карте и на панели информации	Тест пройден
Тестирование кнопки «Отмена» в процессе создания объекта	17	Нажатие кнопки «Отмена» в процессе создания объекта	Очистка введенных данных и возврат в начальное состояние формы	Тест пройден
Проверка работы кнопки «Удалить»	18	Нажатие на кнопку «Удалить» на объекте	Удаление объекта с карты и его исчезновение из базы данных	Тест пройден

Продолжение таблицы 5.1

Попытка создать объект с именем длиной ровно 70 символов (максимально допустимая длина)	19	При создании магазина ввести имя, состоящее из 70 букв «А» подряд	Магазин успешно создаётся (длина имени на границе допустимого), отображается на карте	Тест пройден
Сохранение и загрузка при отсутствии объектов и отгрузок	20	Запустить приложение сразу после установки, без создания магазинов, складов или отгрузок; нажать «Сохранить»; затем «Загрузить»	В файлах сохранения ничего не будет записано. После загрузки структура остаётся пустой: на карте нет ни одного круга, список отгрузок пуст.	Тест пройден
Сохранение после редактирования объектов и перезапуск с загрузкой	21	Создать магазин, затем отредактировать его, нажать «Сохранить», закрыть приложение, снова «Загрузить»	После первой сохранённой сессии файл объектов должен содержать магазин с изменёнными параметрами	Тест пройден
При фильтрации оставить пустыми все поля (никаких условий) и нажать «Применить»	22	В фильтре не вводить ничего, оба чекбокса «склад» и «магазин» отмечены, диапазоны пусты; нажать «Применить»	Отображаются все склады и магазины без изменений	Тест пройден

Продолжение таблицы 5.1

При фильтрации ввести в поле «вместимость от» больше, чем в поле «вместимость до»	23	В фильтре ввести «вместимость от» = 200, «вместимость до» = 100	Применение фильтра становится невозможным, поле «вместимость до» меняет цвет на красный	Тест пройден
Попытка создать объект с именем длиной 71 символ	24	При создании магазина ввести имя из 71 символа «А» подряд	Появление сообщения об ошибке («Имя должно быть не более 70 символов»)	Тест пройден
Ввод недопустимых символов в поле «улица»	25	При создании магазина ввести улицу «St@r!»	Становится невозможным нажатие кнопки «Создать», поле улицы меняет цвет на красный	Тест пройден
Отмена создания нового объекта	26	Нажать «Создать склад», ввести его данные, нажать «Отмена»	Все введенные данные сброшены, панель закрывается, склад не появляется на карте	Тест пройден
Удаление склада, в котором есть товары, но нет отгрузок	27	В складе «S1» есть несколько товаров, но нет активных отгрузок; нажать «Удалить»	Склад «S1» удаляется вместе со всеми товарами, круг исчезает с карты	Тест пройден
Проверка удаления визуального маркера после удаления объекта	28	Создать магазин, нажать «Удалить»	Круг магазина исчезает с карты и больше не доступен при наведении	Тест пройден

Продолжение таблицы 5.1

Создание объекта, не подходящего под текущий фильтр	29	Включить фильтр на отображение магазинов, создать склад	Созданный склад не отображается на карте	Тест пройден
Попытка ввода отрицательного числа в любое поле, которое ожидает число	30	Ввод «-10» в поле «Вместимость» при создании объекта	Вывод сообщения об ошибке	Тест пройден
Попытка создания отгрузки без выбора получателя	31	Указать только склад, не выбрать магазин	Появляется предупреждение, отгрузка не создаётся	Тест пройден
Отмена редактирования объекта	32	Открыть редактирование, изменить данные, нажать «Отмена»	Изменения не сохраняются, данные объекта остаются прежними	Тест пройден
Завершение программы	33	Нажатие на «X» в правом верхнем углу окна	Появление предупреждающего окна, завершение программы	Тест пройден

Все тесты прошли успешно. Сбои в работе программы при прохождении тестов не были обнаружены. Программа полностью исправна и готова к использованию.

6. РУКОВОДСТВО ПО УСТАНОВКЕ И ИСПОЛЬЗОВАНИЮ

Чтобы начать использование программы, нужно использовать установщик. Для этого достаточно запустить файл `setup.exe`, который представлен на рисунке 6.1.



Рисунок 6.1 – Установщик программы

После запуска установщика пользователь должен выбрать режим установки в окне, представленном на рисунке 6.2.

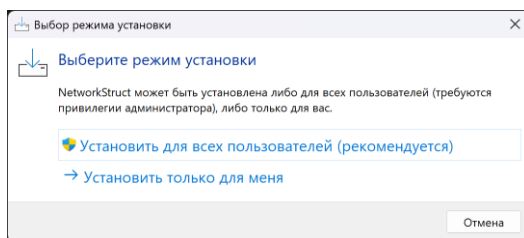


Рисунок 6.2 – Выбор режима установки

В следующем окне пользователь должен выбрать язык установщика. Выбрать можно либо русский, либо английский. Окно выбора языка установщика представлено на рисунке 6.3.

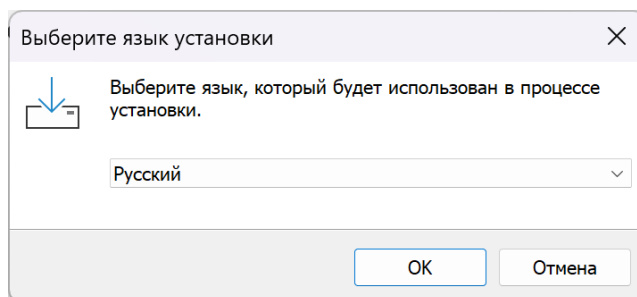


Рисунок 6.3 – Выбор языка установщика

После выбора языка пользователю предлагается выбрать место для установки приложения. Путь по умолчанию - `C:\Program Files (x86)\NetworkStruct`. Окно выбора пути представлено на рисунке 6.4.

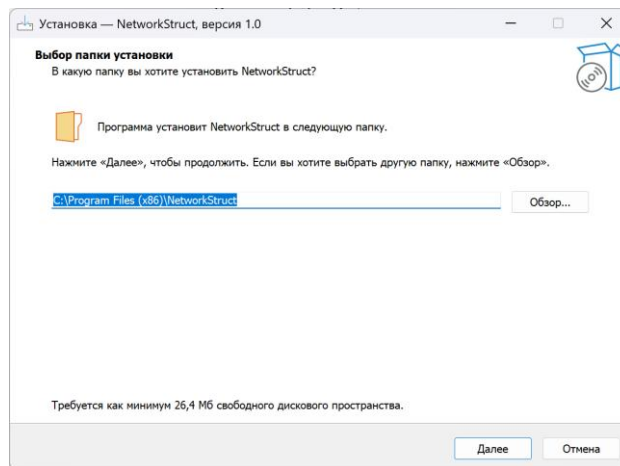


Рисунок 6.4 – Выбор пути для установки

После выбора пути открывается окно, в котором предлагается создать значок на Рабочем столе. Данное окно представлено на рисунке 6.5.

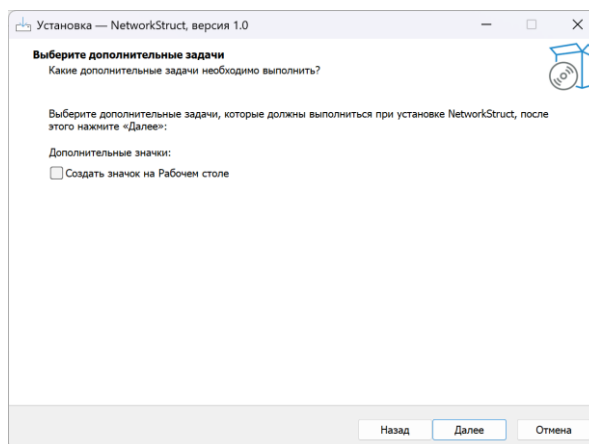


Рисунок 6.5 – Выбор дополнительных задач

Далее пользователь видит окно подтверждения установки, которое представлено на рисунке 6.6.

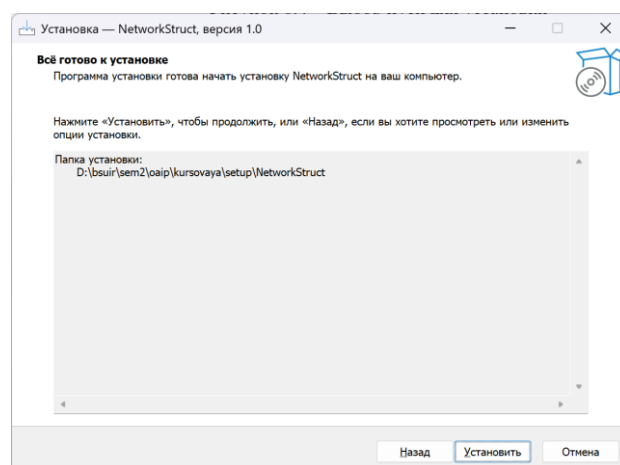


Рисунок 6.6 – Окно подтверждения установки

После установки пользователь увидит окно, представленное на рисунке 6.7, которое говорит о том, что программа успешно завершила установку. После этого ее можно запускать.

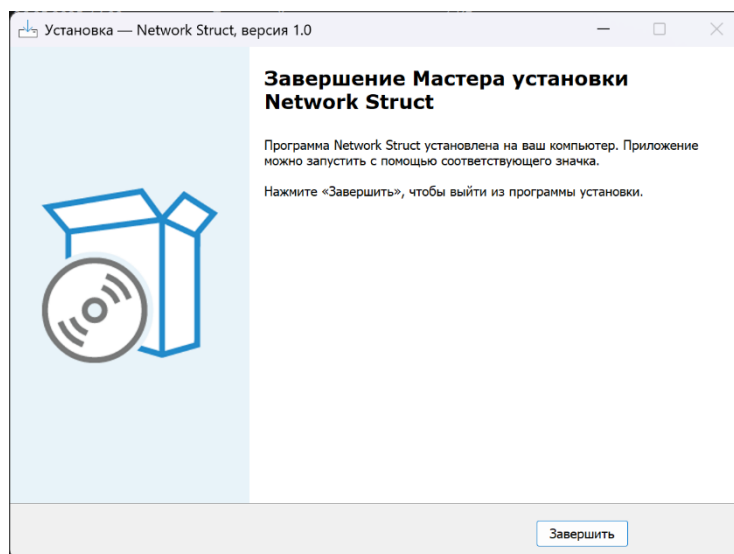


Рисунок 6.7 – Окно завершения установки

Папка, в которую была произведена установка должна содержать 7 элементов: map.bmp, Project1.exe, shipments.txt, shops.txt, unins000.dat, unins000.exe, warehouses.txt. Данная структура представлена на рисунке 6.8.

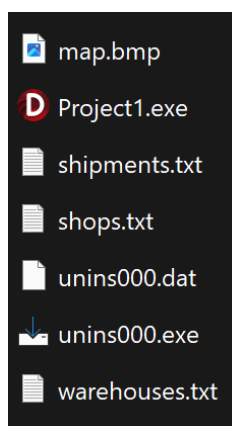


Рисунок 6.8 – Содержимое папки с программой

Map.bmp – фон (схематичное изображение карты), который отображается на главной странице программы. Project1.exe – исполняемый файл программы. Shipments.txt, shops.txt, warehouses.txt – текстовые файлы, которые хранят информацию сохраненных складов, магазинов, отгрузок. Кодировка данных в файле – ANSI. Программа не гарантирует корректную работу, при ручном изменении (если данные введены неверно) какого-либо из файлов: shipments.txt, warehouses.txt, shops.txt. Рекомендуется сохранять данные при помощи комбинации клавиш Ctrl+S или через вкладку программы

«Файл», а затем «Сохранить». Данные в файле shipments.txt должны быть расположены в определенном формате: первая строка должна содержать количество отгрузок, последующие строки содержат информацию о каждой отгрузке в соответствующем формате. Первая строка каждой отгрузки – ее название, вторая строка – ее идентификатор, третья – идентификатор отправителя, четвертая – идентификатор получателя, пятая – название продукта, участвующего в отгрузке, шестая – количество условных единиц товара, которое участвует в отгрузке. Данные в файле shops.txt и в файле warehouses.txt должны быть также должны быть записаны в определенном формате. Первая строка файла – количество объектов (магазинов или складов), последующие строки хранят информацию о каждом объекте в определенном формате. Первая строка информации об объекте содержит название объекта, вторая – улицу, на которой расположен объект, третья – номер дома, четвертая – корпус (-1 если корпус отсутствует), пятая – вместимость объекта, шестая – занятое место, седьмая – место, зарезервированное под дальнейшие доставки, восьмая – идентификатор объекта, девятая строка – количество товаров, которые хранятся в этом объекте, далее идет описание товаров в определенном формате. Первая строка описания каждого товара - название товара, вторая – категория (пустая строка, если категория отсутствует), третья – место, занимаемое единицей товара, четвертая – количество товара, пятая – идентификатор товара, шестая – количество товара, которое отложено для дальнейших доставок.

После запуска приложения пользователь видит главное меню, которое показано на рисунке 6.9.

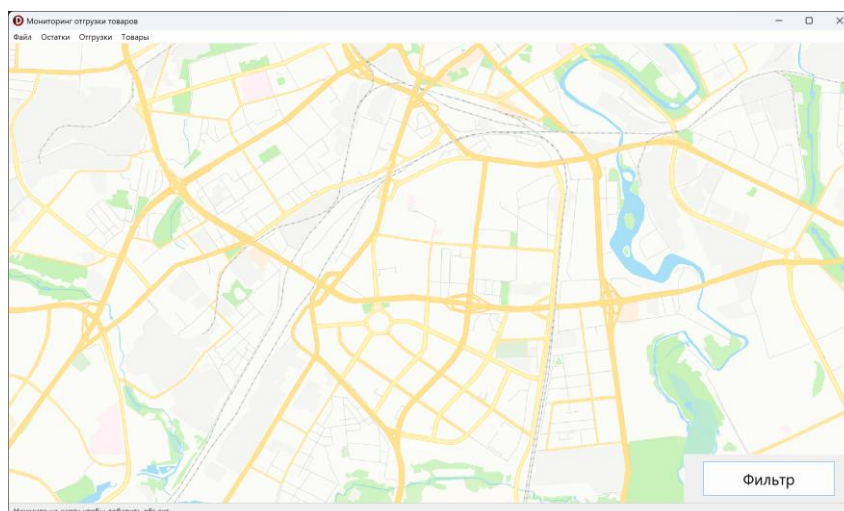


Рисунок 6.9 – Главное меню

Для загрузки данных из файла пользователь должен воспользоваться сочетанием клавиш Ctrl+O, либо вкладкой меню «Файл», а затем «Загрузить»,

изображенной на рисунке 6.10. При этом все данные, которые уже были внесены в приложение будут удалены.

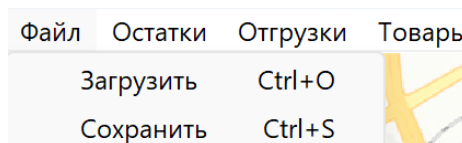


Рисунок 6.10 – Вкладка файл

После нажатия на кнопку загрузки данных появляется окно подтверждения, изображенное на рисунке 6.11.

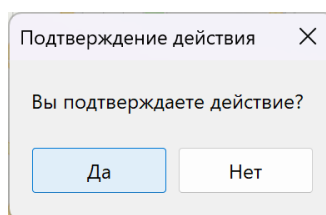


Рисунок 6.11 – Подтверждение загрузки данных

После загрузки данных пользователь увидит сообщение, изображенное на рисунке 6.12.

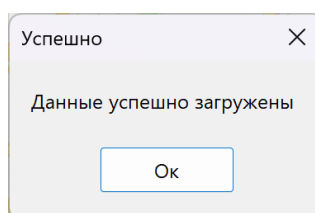


Рисунок 6.12 – Успешная загрузка данных

Для того, чтобы добавить объект на карту, пользователь должен нажать на место, где будет создан новый объект. Появится окно, показанное на рисунке 6.13.

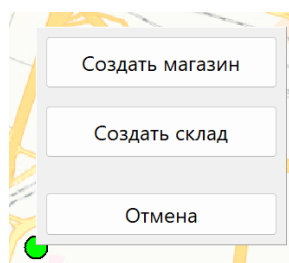
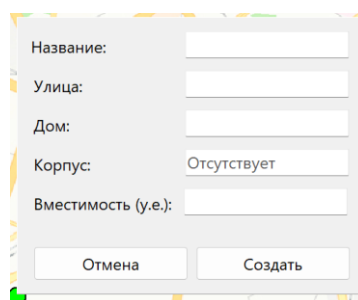


Рисунок 6.13 – Окно выбора типа объекта

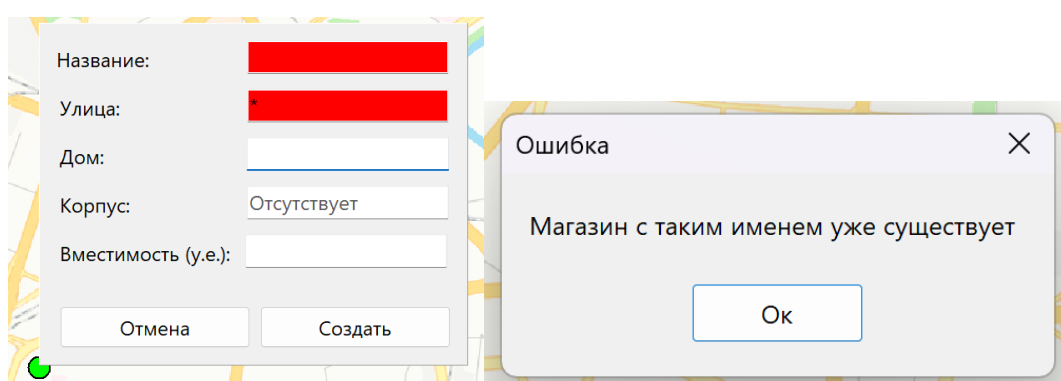
После выбора типа объекта пользователь должен ввести данные для нового объекта. Окно ввода данных показано на рисунке 6.14.



Название:	<input type="text"/>
Улица:	<input type="text"/>
Дом:	<input type="text"/>
Корпус:	Отсутствует
Вместимость (у.е.):	<input type="text"/>
<input type="button" value="Отмена"/> <input type="button" value="Создать"/>	

Рисунок 6.14 – Ввод данных объекта

Все данные проходят проверку при вводе пользователя. При неверном вводе пользователь увидит, какие именно данные введены неверно. Примеры представлены на рисунке 6.15.



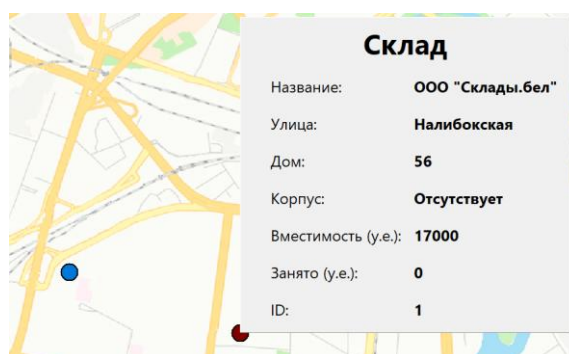
Название:	<input type="text"/>
Улица:	<input type="text"/>
Дом:	<input type="text"/>
Корпус:	Отсутствует
Вместимость (у.е.):	<input type="text"/>
<input type="button" value="Отмена"/> <input type="button" value="Создать"/>	

Ошибка

Магазин с таким именем уже существует

Рисунок 6.15 – Проверка данных ввода

Склады в программе отображаются бордовым цветом, магазины – синим. При наведении на объект появляется вся информация о нем. Примеры представлены на рисунке 6.16.



Склад	
Название:	ООО "Склады.бел"
Улица:	Налибокская
Дом:	56
Корпус:	Отсутствует
Вместимость (у.е.):	17000
Занято (у.е.):	0
ID:	1

Рисунок 6.16 – Представление объектов на карте

Чтобы добавить товар в объект, на панели управления нужно нажать «Товары», затем «Добавить». Появится панель, показанная на рисунке 6.17.

Добавление товара

Название товара:

Категория:

Тип получателя: ☒ Магазин ☐ Склад

Получатель (Название):

Получатель (ID):

Объем единицы товара (y.e.):

Количество единиц товара:

Рисунок 6.17 – Панель добавления товара

В поля панели необходимо ввести все данные товара и объекта, куда добавляется товар. Все поля проходят проверку, а в случае неверного ввода поле меняет цвет на красный, либо появляется окно ошибки, показанное на рисунке 6.15.

Для того, чтобы добавить отгрузку, необходимо нажать «Отгрузки» на панели управления, затем «Создать». Появится окно, показанное на рисунке 6.18.

Добавление отгрузки

Название отгрузки:

Тип отправителя: ☐ Магазин ☒ Склад

Отправитель (Название):

Отправитель (ID):

Тип получателя: ☒ Магазин ☐ Склад

Получатель (Название):

Получатель (ID):

Название товара:

Артикул товара:

Количество:

Рисунок 6.18 – Панель добавления отгрузки

Все поля проходят проверку. При неверном вводе возникает одна из ошибок, которые показаны на рисунке 6.15. Поля связаны между собой: при корректном заполнении одного поля ввода, может автоматически заполняться другое. Созданные отгрузки показаны на карте черными линиями, при наведении на которые пользователь видит информацию об отгрузке. Пример показан на рисунке 6.19.

Отгрузка iPhone

Тип отправителя:	Склад
Отправитель:	ООО "Склады.бел"
ID отправителя:	1
Тип получателя:	Магазин
Получатель:	ТЦ Бонус
ID получателя:	1
Товар:	Отгрузка iPhone
Артикул:	1
Количество:	2
Объем отгрузки:	2

Рисунок 6.19 – Пример отгрузки

Чтобы выполнить отгрузки, можно воспользоваться пунктом меню «Отгрузки», а затем «Выполнить». Далее пользователю дается выбор между «Все», для выполнения всех существующих отгрузок и «Выбрать», если нужно выполнить какие-то конкретные отгрузки. При нажатии «Все» появляется окно с подтверждением, а затем уведомление об успешном выполнении. При нажатии «Выбрать» появляется новая форма, на которой изображена информация о существующих отгрузках, пользователь может настроить, какие именно отгрузки он хочет выполнить. Пример показан на рисунке 6.20.

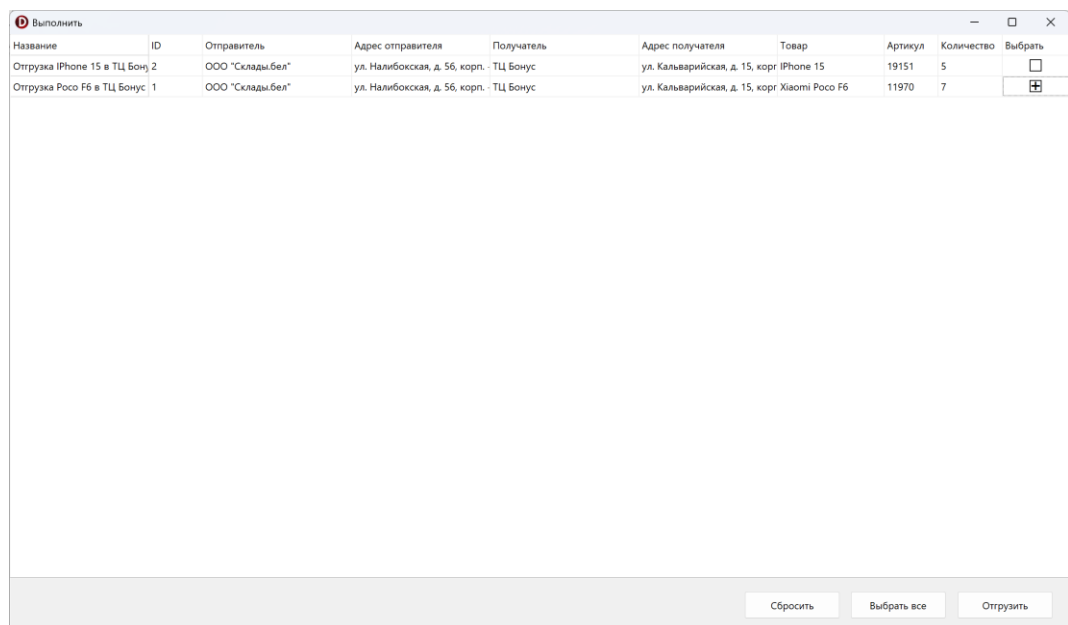


Рисунок 6.20 – Окно выполнения отгрузок

Для того, чтобы просто посмотреть список актуальных отгрузок, пользователь может воспользоваться вкладкой меню «Отгрузки», а затем «Показать». Появляется окно, показанное на рисунке 6.21.

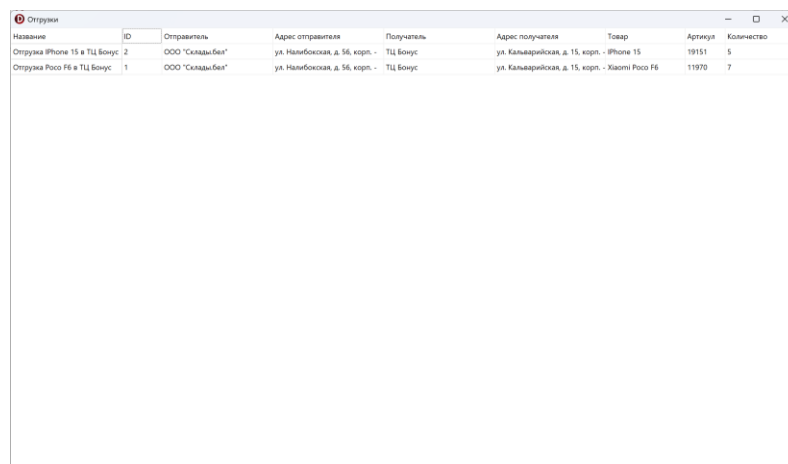
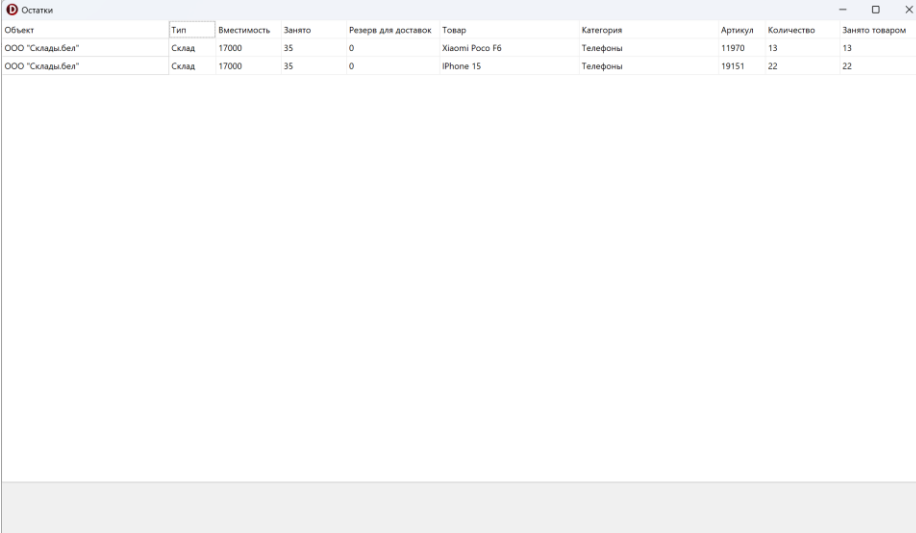


Рисунок 6.21 – Просмотр отгрузок

Для просмотра остатков в объектах, пользователь может воспользоваться вкладкой «Остатки», а затем «Показать». Появляется окно, изображенное на рисунке 6.22, в котором пользователь видит информацию о всех остатках.



Объект	Тип	Вместимость	Занято	Резерв для доставки	Товар	Категория	Артикул	Количество	Занято товаром
ООО "Склады.бел"	Склад	17000	35	0	Xiaomi Poco F6	Телефоны	11970	13	13
ООО "Склады.бел"	Склад	17000	35	0	iPhone 15	Телефоны	19151	22	22

Рисунок 6.22 – Остатки в объектах

При нажатии на объект на карте появляется панель, показанная на рисунке 6.23 в которой можно посмотреть список товаров в данном объекте, удалить или изменить этот объект.

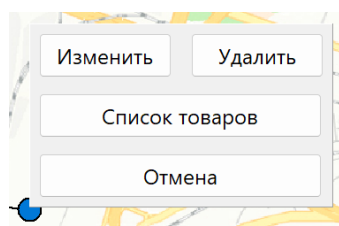


Рисунок 6.23 – Панель конкретного объекта

При нажатии кнопки «Список товаров» пользователь увидит окно, показанное на рисунке 6.24, в котором будут отображены все товары, которые в данный момент есть в этом объекте.



Товары на складе ООО "Склады.бел"				
Название	Категория	Объем единицы товара (л/м)	Количество	Артикул
Xiaomi Poco F6	Телефоны	1	13	11970
iPhone 15	Телефоны	1	22	19151

Рисунок 6.24 – Список товаров в объекте

На главном есть кнопка «Фильтр», при нажатии на которую появляется панель, показанная на рисунке 6.25.

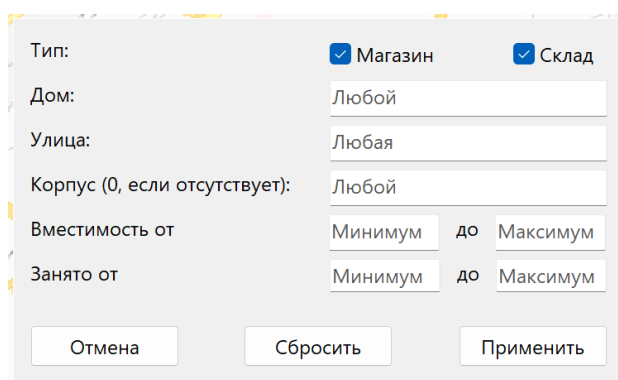


Рисунок 6.25 – Панель фильтров

При нажатии на кнопку «Сбросить» фильтр будет сброшен до исходного состояния, показанного на рисунке 6.25. Для применения фильтров пользователю необходимо ввести нужные ему значения и нажать кнопку «Применить». После применения фильтров на карте останутся только объекты, которые подходят под текущий фильтр, а количество применяемых фильтров будет отображаться рядом с кнопкой «Фильтр» как показано на рисунке 6.26

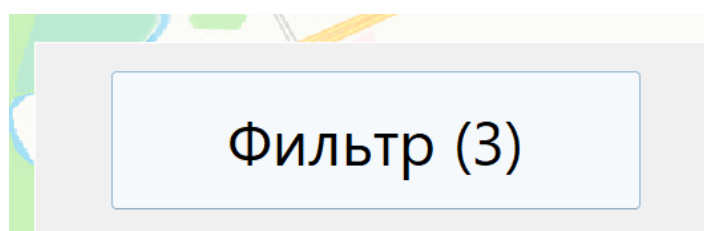


Рисунок 6.26 – Количество применяемых фильтров

Заккрыть программу можно при помощи крестика, который расположен в правом верхнем углу окна и показан на рисунке 6.27.



Рисунок 6.27 – Кнопка закрытия программы

Если есть данные, которые не были сохранены, пользователь увидит окно подтверждения, которое показано на рисунке 6.28.

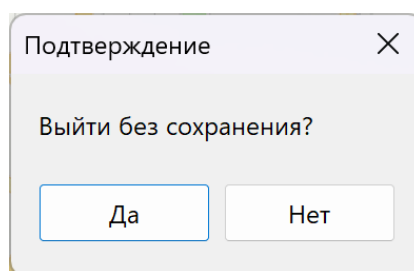


Рисунок 6.28 – Подтверждение выхода

ЗАКЛЮЧЕНИЕ

В рамках курсового проектирования было разработано программное средство для мониторинга и визуализации сетевой структуры отгрузок товаров со складов в магазины в реальном времени.

Изучены и исследованы требования к системе мониторинга логистической цепочки, включая операции добавления, редактирования и удаления складов и магазинов, а также управления ассортиментом товаров и отгрузками.

Разработано программное средство на языке Delphi в среде Embarcadero RAD Studio, что обеспечило высокую производительность и надёжность графического интерфейса с визуализацией на основе векторных элементов.

Разработана структура данных на основе декартового дерева (treap) для хранения списка объектов и вложенных списков товаров, что обеспечило эффективные операции поиска, вставки и удаления за $O(\log N)$.

Предложена реализация механизма фильтрации по типу объекта, адресу, диапазонам вместимости и занятости с динамическим скрыванием/показыванием узлов и маршрутов на карте.

Подготовлены функции сохранения и загрузки состояния системы в текстовые файлы: один для описания объектов с вложенными товарами, другой — для списка отгрузок, что позволяет полностью восстановить рабочее состояние после перезапуска.

Проведена отладка и испытана корректность работы всех основных функций.

Разработанное программное средство может быть использовано специалистами логистических служб для контроля потоков товаров между складами и магазинами, а также преподавателями служб складской логистики.

СПИСОК ИСПОЛЬЗУЕМОЙ ЛИТЕРАТУРЫ

- [1] algorithmica [Электронный ресурс]. – Режим доступа: <https://ru.algorithmica.org/>
- [2] e-maxx [Электронный ресурс]. – Режим доступа: <http://e-maxx.ru/algo>
- [3] docwiki.embarcadero [Электронный ресурс]. – Режим доступа: <https://docwiki.embarcadero.com/>

ПРИЛОЖЕНИЕ А

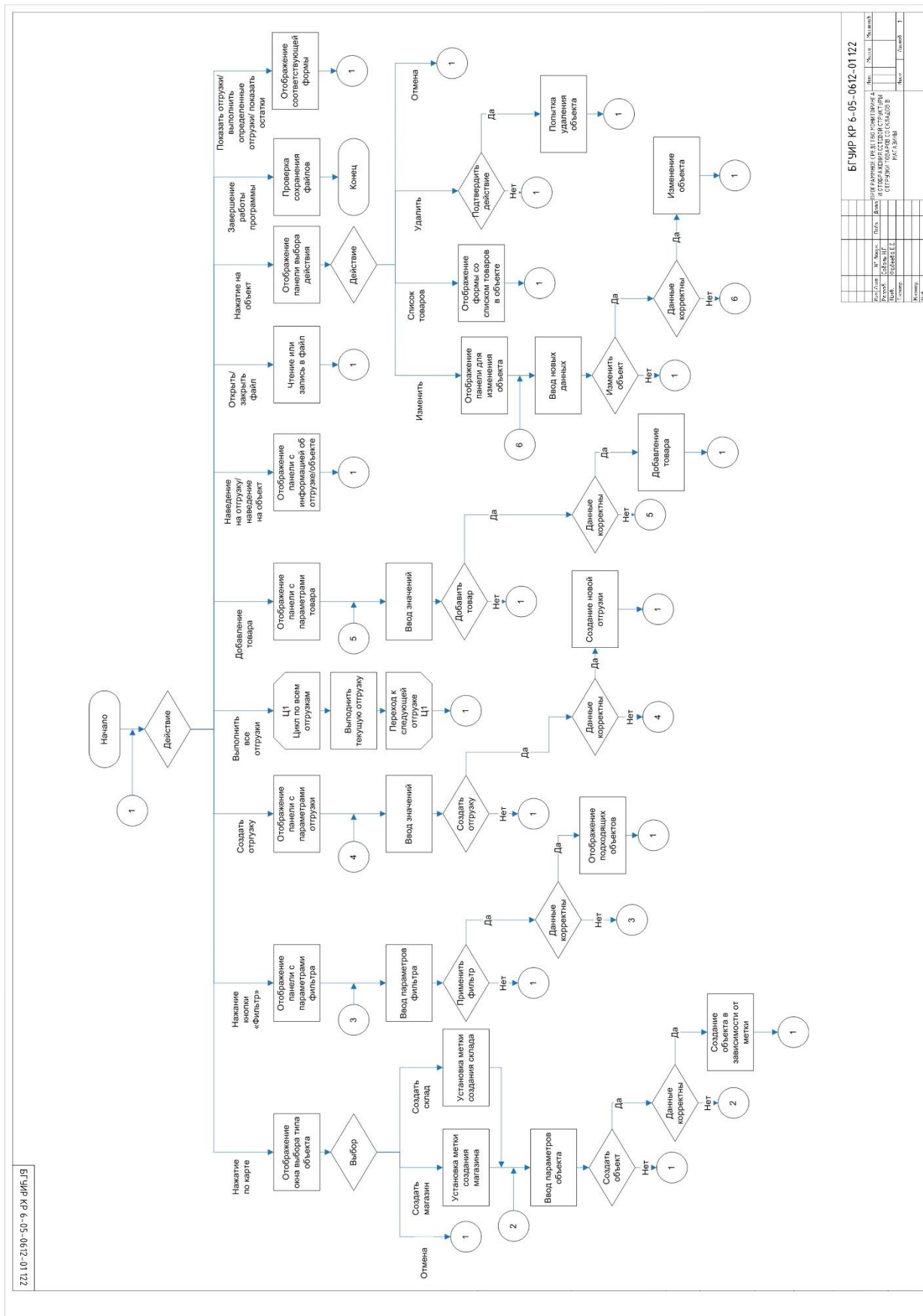


Рисунок А.1 – Схема программного средства

ПРИЛОЖЕНИЕ Б

Содержание модуля BalanceUnit

```
unit BalanceUnit;

interface

uses
  Winapi.Windows, Winapi.Messages, System.SysUtils, System.Variants, System.Classes, Vcl.Graphics,
  Vcl.Controls, Vcl.Forms, Vcl.Dialogs, Vcl.Grids, Vcl.ExtCtrls,
  Types, Vars;

type
  TfrBalance = class(TForm)
    pnBalance: TPanel;
    sgBalanceTable: TStringGrid;
    procedure FormClose(Sender: TObject; var Action: TCloseAction);
    procedure FormCreate(Sender: TObject);
    procedure SetData(const shops, warehouses: PTreapNode);
    function getSizObject(const item: PTreapNode): integer;
    function getSizItems(const item: PTreapItemNode): integer;
    procedure showDataObject(const item: PTreapNode; var i: integer);
    procedure showDataItem(const curObject: PTreapNode; const item: PTreapItemNode; var i: integer);
    procedure FormResize(Sender: TObject);
    procedure sgBalanceTableDrawCell(Sender: TObject; ACol, ARow: LongInt;
      Rect: TRect; State: TGridDrawState);
    procedure sgBalanceTableTopLeftChanged(Sender: TObject);
  private
    { Private declarations }
  public
    { Public declarations }
  end;

var
  frBalance: TfrBalance;

implementation

{$R *.dfm}

var
  startHeight, startWidth: integer;

procedure TfrBalance.FormClose(Sender: TObject; var Action: TCloseAction);
begin
  Action := caFree;
end;

function TfrBalance.getSizItems(const item: PTreapItemNode): integer;
begin
  Result := 0;
  if item <> nil then
  begin
    Inc(Result);
    Result := Result + getSizItems(item^.Left);
    Result := Result + getSizItems(item^.Right);
  end;
end;

function TfrBalance.getSizObject(const item: PTreapNode): integer;
begin
```

```

Result := 0;
if item <> nil then
begin
    Result := Result + getSizItems(item^.Data^.Items);
    Result := Result + getSizObject(item^.Left);
    Result := Result + getSizObject(item^.Right);
end;
end;

procedure TfrBalance.sgBalanceTableDrawCell(Sender: TObject; ACol,
ARow: LongInt; Rect: TRect; State: TGridDrawState);
var
    S: string;
begin
    S := sgBalanceTable.Cells[ACol, ARow];
    sgBalanceTable.Canvas.FillRect(Rect);

    DrawText(
        sgBalanceTable.Canvas.Handle,
        PChar(S), Length(S),
        Rect,
        DT_WORDBREAK or DT_NOPREFIX or DT_LEFT
    );
end;

procedure TfrBalance.sgBalanceTableTopLeftChanged(Sender: TObject);
begin
    sgBalanceTable.LeftCol := 1;
end;

procedure TfrBalance.showDataItem(const curObject: PTreapNode; const item: PTreapItemNode; var i: integer);
begin
    if item <> nil then
    begin
        sgBalanceTable.Cells[0, i] := string(curObject^.Data^.name);
        if (curObject^.Data^.Key and mask) <> 0 then
            sgBalanceTable.Cells[1, i] := 'Магазин'
        else
            sgBalanceTable.Cells[1, i] := 'Склад';
        sgBalanceTable.Cells[2, i] := intToStr(curObject^.Data^.capacity);
        sgBalanceTable.Cells[3, i] := intToStr(curObject^.Data^.usedCapacity);
        sgBalanceTable.Cells[4, i] := intToStr(curObject^.Data^.shipmentCapacity);
        sgBalanceTable.Cells[5, i] := string(item^.Data^.name);
        sgBalanceTable.Cells[6, i] := string(item^.Data^.category);
        sgBalanceTable.Cells[7, i] := string(intToStr(item^.Data^.Key));
        sgBalanceTable.Cells[8, i] := intToStr(item^.Data^.Count);
        sgBalanceTable.Cells[9, i] := intToStr(item^.Data^.Count * item^.Data^.Volume);

        Inc(i);
        showDataItem(curObject, item^.Left, i);
        showDataItem(curObject, item^.Right, i);
    end;
end;

procedure TfrBalance.showDataObject(const item: PTreapNode; var i: integer);
begin
    if item <> nil then
    begin
        showDataItem(item, item^.Data^.Items, i);
        showDataObject(item^.Left, i);
        showDataObject(item^.Right, i);
    end;
end;

```

```

end;
end;

procedure TfrBalance.SetData(const shops, warehouses: PTreapNode);
var
  siz, i: integer;
begin
  siz := getSizObject(shops) + getSizObject(warehouses);
  sgBalanceTable.RowCount := siz + 1;
  i := 1;
  showDataObject(shops, i);
  showDataObject(warehouses, i);
end;

procedure TfrBalance.FormCreate(Sender: TObject);
begin
  startHeight := frBalance.ClientHeight;
  startWidth := frBalance.ClientWidth;
  FormStyle := fsNormal;
  Position := poMainFormCenter;

  sgBalanceTable.ColCount := 10;

  sgBalanceTable.ColWidths[0] := trunc(0.18 * frBalance.clientWidth);
  sgBalanceTable.ColWidths[1] := trunc(0.05 * frBalance.clientWidth);
  sgBalanceTable.ColWidths[2] := trunc(0.07 * frBalance.clientWidth);
  sgBalanceTable.ColWidths[3] := trunc(0.07 * frBalance.clientWidth);
  sgBalanceTable.ColWidths[4] := trunc(0.10 * frBalance.clientWidth);
  sgBalanceTable.ColWidths[5] := trunc(0.15 * frBalance.clientWidth);
  sgBalanceTable.ColWidths[6] := trunc(0.15 * frBalance.clientWidth);
  sgBalanceTable.ColWidths[7] := trunc(0.05 * frBalance.clientWidth);
  sgBalanceTable.ColWidths[8] := trunc(0.08 * frBalance.clientWidth);
  sgBalanceTable.ColWidths[9] := frBalance.clientWidth
    -sgBalanceTable.ColWidths[0]
    -sgBalanceTable.ColWidths[1]
    -sgBalanceTable.ColWidths[2]
    -sgBalanceTable.ColWidths[3]
    -sgBalanceTable.ColWidths[4]
    -sgBalanceTable.ColWidths[5]
    -sgBalanceTable.ColWidths[6]
    -sgBalanceTable.ColWidths[7]
    -sgBalanceTable.ColWidths[8]
    -sgBalanceTable.ColCount * sgBalanceTable.GridLineWidth;

  sgBalanceTable.Cells[0, 0] := 'Объект';
  sgBalanceTable.Cells[1, 0] := 'Тип';
  sgBalanceTable.Cells[2, 0] := 'Вместимость';
  sgBalanceTable.Cells[3, 0] := 'Занято';
  sgBalanceTable.Cells[4, 0] := 'Резерв для доставок';
  sgBalanceTable.Cells[5, 0] := 'Товар';
  sgBalanceTable.Cells[6, 0] := 'Категория';
  sgBalanceTable.Cells[7, 0] := 'Артикул';
  sgBalanceTable.Cells[8, 0] := 'Количество';
  sgBalanceTable.Cells[9, 0] := 'Занято товаром';

  sgBalanceTable.RowCount := 0;

  sgBalanceTable.DefaultDrawing := false;
end;

procedure TfrBalance.FormResize(Sender: TObject);
begin

```

```

frBalance.ClientHeight := startHeight;
frBalance.ClientWidth := startWidth;
end;

end.

```

Содержание модуля CartesianTree

```

unit CartesianTree;

interface

uses Vcl.ExtCtrls, CartesianTreeItem, System.Generics.Collections, Types;

procedure SplitTreap(t: PTreapNode; const key: Integer; var L, R: PTreapNode);
function MergeTreap(var L, R: PTreapNode): PTreapNode;
procedure InsertTreap(var Root: PTreapNode; var NewNode: PTreapNode);
procedure EraseTreap(var Root: PTreapNode; const Key: Integer);
procedure ClearTreap(var Root: PTreapNode);
procedure InitTree(var root: PTreapNode);
function FindTreap(var Root: PTreapNode; const Key: Integer): PTreapNode;
function CreateNewNode(var Data: PLocation): PTreapNode;

implementation

procedure InitTree(var root: PTreapNode);
begin
    root := nil;
end;

function CreateNewNode(var Data: PLocation): PTreapNode;
begin
    New(Result);
    Result^.Data := Data;
    Result^.Left := nil;
    Result^.Right := nil;
    Result^.Priority := Random(MaxInt);
end;

function FindTreap(var Root: PTreapNode; const Key: Integer): PTreapNode;
begin
    if Root = nil then
        begin
            Result := nil;
        end
    else if Root^.Data^.Key = Key then
        begin
            Result := Root;
        end
    else if Key < Root^.Data^.Key then
        begin
            Result := FindTreap(Root^.Left, Key);
        end
    else
        begin
            Result := FindTreap(Root^.Right, Key);
        end
    end;
end;

procedure SplitTreap(t: PTreapNode; const key: Integer; var L, R: PTreapNode);
begin
    if t = nil then
        begin
            L := nil;

```

```

    R := nil;
end
else if t^.Data^.Key < key then
begin
    SplitTreap(t^.Right, key, t^.Right, R);
    L := t;
end
else
begin
    SplitTreap(t^.Left, key, L, t^.Left);
    R := t;
end;
end;
end;

```

```

function MergeTreap(var L, R: PTreapNode): PTreapNode;
begin
    if L = nil then
        Exit(R);
    if R = nil then
        Exit(L);

    if (L^.Priority > R^.Priority)
        or ((L^.Priority = R^.Priority) and (L^.Data^.Key < R^.Data^.Key)) then
    begin
        L^.Right := MergeTreap(L^.Right, R);
        Result := L;
    end
    else
    begin
        R^.Left := MergeTreap(L, R^.Left);
        Result := R;
    end;
end;
end;

```

```

procedure InsertTreap(var Root: PTreapNode; var NewNode: PTreapNode);
var
    L, R: PTreapNode;
begin
    if Root = nil then
    begin
        NewNode^.Left := nil;
        NewNode^.Right := nil;
        Root := NewNode;
    end
    else if NewNode^.Priority > Root^.Priority then
    begin
        SplitTreap(Root, NewNode^.Data^.Key, L, R);
        NewNode^.Left := L;
        NewNode^.Right := R;
        Root := NewNode;
    end
    else
    begin
        if NewNode^.Data^.Key < Root^.Data^.Key then
            InsertTreap(Root^.Left, NewNode)
        else
            InsertTreap(Root^.Right, NewNode);
        end;
    end;
end;
end;

```

```

procedure EraseTreap(var Root: PTreapNode; const Key: Integer);

```



```

var
  Temp: PTreapNode;
begin
  if Root = nil then
    begin
      { Узла не существует }
    end
  else if Key < Root^.Data^.Key then
    begin
      EraseTreap(Root^.Left, Key);
    end
  else if Key > Root^.Data^.Key then
    begin
      EraseTreap(Root^.Right, Key);
    end
  else
    begin
      Temp := Root;
      Root := MergeTreap(Root^.Left, Root^.Right);
      Dispose(Temp);
    end;
end;

procedure ClearTreap(var Root: PTreapNode);
begin
  if Root <> nil then
    begin
      ClearTreap(Root^.Left);
      ClearTreap(Root^.Right);
      ClearTreapItem(Root^.Data^.Items);
      root^.Data^.OutgoingArrows.Free;
      root^.Data^.IncomingArrows.Free;
      Root^.Data^.shape.Free;
      Dispose(Root);
      Root := nil;
    end;
end;

end.

```

Содержание модуля CartesianTreeByName

```
unit CartesianTreeByName;

interface

uses Vcl.ExtCtrls, Hash, Types;

function CreateNewNameNode(const name: string; const ID: integer): PTreapNameNode;
procedure SplitTreapName(t: PTreapNameNode; const key: Integer; var L, R: PTreapNameNode);
function MergeTreapName(var L, R: PTreapNameNode): PTreapNameNode;
procedure InsertTreapName(var Root, NewNode: PTreapNameNode);
procedure EraseTreapName(var Root: PTreapNameNode; const Key: Integer);
procedure ClearTreapName(var Root: PTreapNameNode);
procedure InitTreeName(var root: PTreapNameNode);
function FindTreapName(var Root: PTreapNameNode; const Key: Integer): PTreapNameNode;

implementation

procedure InitTreeName(var root: PTreapNameNode);
begin
    root := nil;
end;

function CreateNewNode(var Data: PLocation): PTreapNode;
begin
    New(Result);
    Result^.Data := Data;
    Result^.Left := nil;
    Result^.Right := nil;
    Result^.Priority := Random(MaxInt);
end;

function CreateNewNameNode(const name: string; const ID: integer): PTreapNameNode;
begin
    New(Result);
    New(Result^.Data);
    Result^.Data^.name := shortString(name);
    Result^.Data^.Key := getHash(name);
    Result^.Data^.ID := ID;
    Result^.Left := nil;
    Result^.Right := nil;
    Result^.Priority := Random(MaxInt);
end;

function FindTreapName(var Root: PTreapNameNode; const Key: Integer): PTreapNameNode;
begin
    if Root = nil then
        begin
            Result := nil;
        end
    else if Root^.Data^.Key = Key then
        begin
            Result := Root;
        end
    else if Key < Root^.Data^.Key then
        begin
            Result := FindTreapName(Root^.Left, Key);
        end
    else
        begin
            Result := FindTreapName(Root^.Right, Key);
        end
    end;
end;
```

```

end;

function ExistsPriorityName(var Node: PTreapNameNode; const pr: Integer): Boolean;
begin
  if Node = nil then
    begin
      Result := False;
    end
  else if Node^.Priority = pr then
    begin
      Result := True;
    end
  else
    begin
      Result := ExistsPriorityName(Node^.Left, pr) or ExistsPriorityName(Node^.Right, pr);
    end;
  end;
end;

procedure SplitTreapName(t: PTreapNameNode; const key: Integer; var L, R: PTreapNameNode);
begin
  if t = nil then
    begin
      L := nil;
      R := nil;
    end
  else if t^.Data^.Key < key then
    begin
      SplitTreapName(t^.Right, key, t^.Right, R);
      L := t;
    end
  else
    begin
      SplitTreapName(t^.Left, key, L, t^.Left);
      R := t;
    end;
  end;
end;

function MergeTreapName(var L, R: PTreapNameNode): PTreapNameNode;
begin
  if L = nil then
    Exit(R);
  if R = nil then
    Exit(L);

  if (L^.Priority > R^.Priority)
    or ((L^.Priority = R^.Priority) and (L^.Data^.Key < R^.Data^.Key)) then
    begin
      L^.Right := MergeTreapName(L^.Right, R);
      Result := L;
    end
  else
    begin
      R^.Left := MergeTreapName(L, R^.Left);
      Result := R;
    end;
  end;
end;

procedure InsertTreapName(var Root, NewNode: PTreapNameNode);
var
  L, R: PTreapNameNode;

```

```

begin
  if Root = nil then
    begin
      NewNode^.Left := nil;
      NewNode^.Right := nil;
      Root := NewNode;
    end
  else if NewNode^.Priority > Root^.Priority then
    begin
      SplitTreapName(Root, NewNode^.Data^.Key, L, R);
      NewNode^.Left := L;
      NewNode^.Right := R;
      Root := NewNode;
    end
  else
    begin
      if NewNode^.Data^.Key < Root^.Data^.Key then
        InsertTreapName(Root^.Left, NewNode)
      else
        InsertTreapName(Root^.Right, NewNode);
      end;
    end;
end;

```

```

procedure EraseTreapName(var Root: PTreapNameNode; const Key: Integer);
var
  Temp: PTreapNameNode;
begin
  if Root = nil then
    begin
      { Узел с таким ключом отсутствует }
    end
  else if Key < Root^.Data^.Key then
    begin
      EraseTreapName(Root^.Left, Key);
    end
  else if Key > Root^.Data^.Key then
    begin
      EraseTreapName(Root^.Right, Key);
    end
  else
    begin
      Temp := Root;
      Root := MergeTreapName(Root^.Left, Root^.Right);
      Dispose(Temp);
    end;
end;

```

```

procedure ClearTreapName(var Root: PTreapNameNode);
begin
  if Root <> nil then
    begin
      ClearTreapName(Root^.Left);
      ClearTreapName(Root^.Right);
      Dispose(Root);
      Root := nil;
    end;
end;

end.

```

Содержание модуля CartesianTreeItem

```
unit CartesianTreeItem;
```

```

interface
uses Vcl.ExtCtrls, Types;

function CreateNewItemNode(const Data: PItem): PTreapItemNode;
procedure SplitTreapItem(t: PTreapItemNode; const key: Integer; var L, R: PTreapItemNode);
function MergeTreapItem(var L, R: PTreapItemNode): PTreapItemNode;
procedure InsertTreapItem(var Root, NewNode: PTreapItemNode);
procedure EraseTreapItem(var Root: PTreapItemNode; const Key: Integer);
procedure ClearTreapItem(var Root: PTreapItemNode);
procedure InitTreeItem(var root: PTreapItemNode);
function FindTreapItem(var Root: PTreapItemNode; const Key: Integer): PTreapItemNode;

implementation

procedure InitTreeItem(var root: PTreapItemNode);
begin
    root := nil;
end;

function CreateNewItemNode(const Data: PItem): PTreapItemNode;
begin
    New(Result);
    Result^.Data := Data;
    Result^.Left := nil;
    Result^.Right := nil;
    Result^.Priority := Random(MaxInt);
end;

function FindTreapItem(var Root: PTreapItemNode; const Key: Integer): PTreapItemNode;
begin
    if Root = nil then
        begin
            Result := nil;
        end
    else if Root^.Data^.Key = Key then
        begin
            Result := Root;
        end
    else if Key < Root^.Data^.Key then
        begin
            Result := FindTreapItem(Root^.Left, Key);
        end
    else
        begin
            Result := FindTreapItem(Root^.Right, Key);
        end
    end;
end;

procedure SplitTreapItem(t: PTreapItemNode; const key: Integer; var L, R: PTreapItemNode);
begin
    if t = nil then
        begin
            L := nil;
            R := nil;
        end
    else if t^.Data^.Key < key then
        begin
            SplitTreapItem(t^.Right, key, t^.Right, R);
            L := t;
        end
    end
end

```

```

else
begin
  SplitTreapItem(t^.Left, key, L, t^.Left);
  R := t;
end;
end;

function MergeTreapItem(var L, R: PTreapItemNode): PTreapItemNode;
begin
  if L = nil then
    Exit(R);
  if R = nil then
    Exit(L);

  if (L^.Priority > R^.Priority)
    or ((L^.Priority = R^.Priority) and (L^.Data^.Key < R^.Data^.Key)) then
  begin
    L^.Right := MergeTreapItem(L^.Right, R);
    Result := L;
  end
  else
  begin
    R^.Left := MergeTreapItem(L, R^.Left);
    Result := R;
  end;
end;

procedure InsertTreapItem(var Root, NewNode: PTreapItemNode);
var
  L, R: PTreapItemNode;
begin
  if Root = nil then
  begin
    NewNode^.Left := nil;
    NewNode^.Right := nil;
    Root := NewNode;
  end
  else if NewNode^.Priority > Root^.Priority then
  begin
    SplitTreapItem(Root, NewNode^.Data^.Key, L, R);
    NewNode^.Left := L;
    NewNode^.Right := R;
    Root := NewNode;
  end
  else
  begin
    if NewNode^.Data^.Key < Root^.Data^.Key then
      InsertTreapItem(Root^.Left, NewNode)
    else
      InsertTreapItem(Root^.Right, NewNode);
  end;
end;

procedure EraseTreapItem(var Root: PTreapItemNode; const Key: Integer);
var
  Temp: PTreapItemNode;
begin
  if Root = nil then
  begin
    { Узел с таким ключом отсутствует }
  end;
end;

```

```

end
else if Key < Root^.Data^.Key then
begin
  EraseTreapItem(Root^.Left, Key);
end
else if Key > Root^.Data^.Key then
begin
  EraseTreapItem(Root^.Right, Key);
end
else
begin
  Temp := Root;
  Root := MergeTreapItem(Root^.Left, Root^.Right);
  Dispose(Temp);
end;
end;

procedure ClearTreapItem(var Root: PTreapItemNode);
begin
  if Root <> nil then
  begin
    ClearTreapItem(Root^.Left);
    ClearTreapItem(Root^.Right);
    Dispose(Root);
    Root := nil;
  end;
end;

end.

```

Содержание модуля Filter

```

unit Filter;

interface

uses Types, Vars;

procedure InitFilter(var resObj: TFilter);
procedure CreateFilter(var resObj: TFilter; const objType: integer; const objStreet: string;
  const house, building,
  capacityFrom, capacityTo,
  usedCapacityFrom, usedCapacityTo: integer);
procedure ApplyFilter(const root: PTreapNode; const filter: TFilter);

implementation

procedure createFilter(var resObj: TFilter; const objType: integer; const objStreet: string;
  const house, building,
  capacityFrom, capacityTo,
  usedCapacityFrom, usedCapacityTo: integer);
//objType = 0 (00b): none, 1(01b): shop, 2(10b): warehouse, 3(11b): both
begin
  resObj.buildingType := objType;
  resObj.street := shortString(objStreet);
  resObj.house := house;
  resObj.building := building;
  resObj.capacityFrom := capacityFrom;
  resObj.capacityTo := capacityTo;
  resObj.usedCapacityFrom := usedCapacityFrom;
  resObj.usedCapacityTo := usedCapacityTo;
end;

procedure InitFilter(var resObj: TFilter);

```

```

begin
  resObj.buildingType := 3;
  resObj.street := "";
  resObj.house := -1;
  resObj.building := -1;
  resObj.capacityFrom := -1;
  resObj.capacityTo := -1;
  resObj.usedCapacityFrom := -1;
  resObj.usedCapacityTo := -1;
end;

procedure ApplyFilter(const root: PTreapNode; const filter: TFilter);
var
  curType: integer;
  show: boolean;
begin
  if root <> nil then
    begin
      show := true;

      if (root^.Data^.Key and mask) <> 0 then
        curType := 1
      else
        curType := 2;

      if (curType and filter.buildingType) = 0 then
        show := false;
      if True then

        if (Length(filter.street) > 0) and (filter.street <> root^.Data^.street) then
          show := false;

        if (filter.house <> -1) and (filter.house <> root^.Data^.house) then
          show := false;

        if (filter.building <> -1) and (filter.building <> root^.Data^.building) then
          show := false;

        if (filter.capacityFrom <> -1) and (filter.capacityFrom > root^.Data^.capacity) then
          show := false;

        if (filter.capacityTo <> -1) and (filter.capacityTo < root^.Data^.capacity) then
          show := false;

        if (filter.usedCapacityFrom <> -1) and (filter.usedCapacityFrom > root^.Data^.usedCapacity) then
          show := false;

        if (filter.usedCapacityTo <> -1) and (filter.usedCapacityTo < root^.Data^.usedCapacity) then
          show := false;

        if not show then
          root^.Data^.shape.Visible := false
        else
          root^.Data^.shape.Visible := true;

        if root^.Left <> nil then
          ApplyFilter(root^.Left, filter);
        if root^.Right <> nil then
          ApplyFilter(root^.Right, filter);
        end;
      end;
    end;
end;

```


end.

Содержание модуля GetKeys

unit GetKeys;

interface

uses Vars;

function getShopKey: integer;

function getWarehouseKey: integer;

implementation

function getShopKey: integer;

begin

Result := shopKey;

shopKey := shopKey + 1;

end;

function getWarehouseKey: integer;

begin

Result := warehouseKey;

warehouseKey := warehouseKey + 1;

end;

end.

Содержание модуля Hash

unit Hash;

interface

uses Vars;

procedure initHash(const userP, userM: integer);

function getHash(const str: string): integer;

implementation

procedure initHash(const userP, userM: integer);

var

i: integer;

begin

p := userP;

m := userM;

pows[0] := 1;

for i := 1 to High(pows) do

begin

pows[i] := pows[i-1] * p;

pows[i] := pows[i] mod m;

end;

end;

function getHash(const str: string): integer;

var

i: Integer;

begin

Result := 0;

for i := Low(str) to High(str) do

begin

Result := Result + ord(str[i]) * pows[i];

```

    Result := Result mod m;
end;
end;

end.

```

Содержание модуля MainUnit

```

unit MainUnit;

interface

uses
    Winapi.Windows, Winapi.Messages, System.SysUtils, System.Variants, System.Classes, Vcl.Graphics,
    Vcl.Controls, Vcl.Forms, Vcl.Dialogs, Vcl.ExtCtrls, Vcl.StdCtrls,
    Vcl.Imaging.jpeg, System.UITypes, System.Types, Vcl.Menus, Vcl.NumberBox,
    CartesianTreeByName, CartesianTreeItem, Validation,
    GetKeys, Hash, Messages, Filter, Data.FMTBcd, Data.DB,
    Data.SqlExpr, Vcl.Grids,
    TableUnit, ShipmentsTableUnit, shipments, BalanceUnit, SelectShipmentsUnit,
    ArrowsUnit, CartesianTree,
    System.Generics.Collections,
    Types, Vars;

type

TfrMainForm = class(TForm)
    pnFilter: TPanel;
    btnFilter: TButton;
    pnMapWrap: TPanel;
    pnCreateSelect: TPanel;
    btnCreateSelectShop: TButton;
    btnCreateSelectWarehouse: TButton;
    btnCreateSelectCancel: TButton;
    spMapPoint: TShape;
    pnCreateObj: TPanel;
    lbCreateObjName: TLabel;
    edCreateObjName: TEdit;
    lbCreateObjStreet: TLabel;
    edCreateObjStreet: TEdit;
    lbCreateObjHouse: TLabel;
    edCreateObjHouse: TEdit;
    btnCreateObjConfirm: TButton;
    btnCreateObjCancel: TButton;
    lbCreateObjBuilding: TLabel;
    edCreateObjBuilding: TEdit;
    edCreateObjCapacity: TEdit;
    lbCreateObjCapacity: TLabel;
    pnSelectObject: TPanel;
    btnSelectObjEdit: TButton;
    btnSelectObjDelete: TButton;
    btnSelectObjCancel: TButton;
    pnEditObj: TPanel;
    lbEditObjName: TLabel;
    lbEditObjStreet: TLabel;
    lbEditObjHouse: TLabel;
    lbEditObjBuilding: TLabel;
    lbEditObjCapacity: TLabel;
    edEditObjName: TEdit;
    edEditObjStreet: TEdit;
    edEditObjHouse: TEdit;
    btnEditObjConfirm: TButton;
    btnEditObjCancel: TButton;
    edEditObjBuilding: TEdit;

```

```

edEditObjCapacity: TEdit;
pnObjectInfo: TPanel;
lbObjInfoName: TLabel;
lbObjInfoStreet: TLabel;
lbObjInfoHouse: TLabel;
lbObjInfoBuilding: TLabel;
lbObjInfoCapacity: TLabel;
lbObjInfoNameVal: TLabel;
lbObjInfoStreetVal: TLabel;
lbObjInfoHouseVal: TLabel;
lbObjInfoBuildingVal: TLabel;
lbObjInfoCapacityVal: TLabel;
lbObjInfoTitle: TLabel;
lbObjInfoUsedCapacity: TLabel;
lbObjInfoUsedCapacityVal: TLabel;
MainMenu1: TMainMenu;
N4: TMenuItem;
N6: TMenuItem;
N10: TMenuItem;
N11: TMenuItem;
pnHints: TPanel;
lbHints: TLabel;
pnFilterParams: TPanel;
btnFilterDefault: TButton;
btnFilterCancel: TButton;
pnFilterButtons: TPanel;
btnFilterConfirm: TButton;
lbFilterStreet: TLabel;
lbFilterHouse: TLabel;
lbFilterBuilding: TLabel;
lbFilterCapacityFrom: TLabel;
lbFilterCapacityTo: TLabel;
lbFilterType: TLabel;
cbFilterTypeWarehouse: TCheckBox;
pnFilterParamsType: TPanel;
pnFilterParamsStreet: TPanel;
edFilterStreetVal: TEdit;
pnFilterParamsHouse: TPanel;
edFilterHouseVal: TEdit;
pnFilterParamsBuilding: TPanel;
edFilterBuildingVal: TEdit;
pnFilterParamsCapacity: TPanel;
edFilterCapacityFromVal: TEdit;
edFilterCapacityToVal: TEdit;
pnFilterParamsUsedCapacity: TPanel;
lbFilterUsedCapacityFrom: TLabel;
lbFilterUsedCapacityTo: TLabel;
edFilterUsedCapacityFromVal: TEdit;
edFilterUsedCapacityToVal: TEdit;
cbFilterTypeShop: TCheckBox;
pnCreateShipment: TPanel;
lbCreateShipment: TLabel;
lbObjInfoId: TLabel;
lbObjInfoIdVal: TLabel;
lbCreateShipmentName: TLabel;
N12: TMenuItem;
N13: TMenuItem;
lbCreateShipmentSenderName: TLabel;
lbCreateShipmentDestName: TLabel;
lbCreateShipmentItemName: TLabel;
lbCreateShipmentCnt: TLabel;
lbCreateShipmentSenderID: TLabel;

```

lbCreateShipmentDestID: TLabel;
 lbCreateShipmentItemID: TLabel;
 edCreateShipmentSenderName: TEdit;
 edCreateShipmentSenderID: TEdit;
 edCreateShipmentDestName: TEdit;
 edCreateShipmentDestID: TEdit;
 edCreateShipmentItemName: TEdit;
 edCreateShipmentItemID: TEdit;
 edCreateShipmentCnt: TEdit;
 edCreateShipmentName: TEdit;
 btnCreateShipmentCancel: TButton;
 btnCreateShipmentConfirm: TButton;
 lbCreateShipmentSenderType: TLabel;
 rbCreateShipmentSenderShop: TRadioButton;
 rbCreateShipmentSenderWarehouse: TRadioButton;
 rbCreateShipmentDestShop: TRadioButton;
 rbCreateShipmentDestWarehouse: TRadioButton;
 lbCreateShipmentDestType: TLabel;
 pnCreateShipmentSenderType: TPanel;
 pnCreateShipmentDestType: TPanel;
 pnAddItem: TPanel;
 pnAddItemType: TPanel;
 rbAddItemTypeShop: TRadioButton;
 rbAddItemTypeWarehouse: TRadioButton;
 lbAddItemType: TLabel;
 lbAddItemName: TLabel;
 edAddItemName: TEdit;
 lbAddItemDestName: TLabel;
 edAddItemDestName: TEdit;
 lbAddItemDestID: TLabel;
 edAddItemDestID: TEdit;
 lbAddItemVol: TLabel;
 lbAddItemCnt: TLabel;
 edAddItemVol: TEdit;
 edAddItemCnt: TEdit;
 btnAddItemCancel: TButton;
 btnAddItemConfirm: TButton;
 lbAddItemCategory: TLabel;
 edAddItemCategory: TEdit;
 File1: TMenuItem;
 Open1: TMenuItem;
 Save1: TMenuItem;
 N1: TMenuItem;
 N2: TMenuItem;
 N3: TMenuItem;
 N14: TMenuItem;
 btnSelectObjItemList: TButton;
 N15: TMenuItem;
 N16: TMenuItem;
 pbMap: TPaintBox;
 pnArrowInfo: TPanel;
 lbArrowInfoShipmentName: TLabel;
 lbArrowInfoSenderType: TLabel;
 lbArrowInfoSenderNameVal: TLabel;
 lbArrowInfoSenderName: TLabel;
 lbArrowInfoDestTypeVal: TLabel;
 lbArrowInfoSenderID: TLabel;
 lbArrowInfoSenderIDVal: TLabel;
 lbArrowInfoDestName: TLabel;
 lbArrowInfoSenderTypeVal: TLabel;
 lbArrowInfoDestType: TLabel;
 lbArrowInfoDestNameVal: TLabel;

```

lbArrowInfoDestID: TLabel;
lbArrowInfoDestIDVal: TLabel;
lbArrowInfoItemID: TLabel;
lbArrowInfoItemIDVal: TLabel;
lbArrowInfoItemName: TLabel;
lbArrowInfoItemNameVal: TLabel;
lbArrowInfoItemCnt: TLabel;
lbArrowInfoItemCntVal: TLabel;
lbArrowInfoItemVol: TLabel;
lbArrowInfoItemVolVal: TLabel;
lbAddItem: TLabel;

procedure createNewObj(var newObj: PLocation; const isShop: boolean);

procedure imgMapMouseUp(Sender: TObject; Button: TMouseButton;
    Shift: TShiftState; X, Y: Integer);

procedure updateID(const editID: TEdit;
    const isShopRadio: TRadioButton;
    const editName: TEdit);
procedure updateName(const editID: TEdit;
    const isShopRadio: TRadioButton;
    const editName: TEdit);

procedure btnCreateSelectCancelClick(Sender: TObject);
procedure btnCreateObjCancelClick(Sender: TObject);
procedure btnCreateSelectClick(Sender: TObject);
procedure btnCreateObjConfirmClick(Sender: TObject);

procedure OnClickValidateLength(Sender: TObject);
procedure OnClickValidateLetters(Sender: TObject);
procedure OnClickvalidateAll(Sender: TObject);

procedure createShop(Sender: TObject);
procedure createWarehouse(Sender: TObject);

function validateCreateObj: boolean;
function validateEditObj: boolean;
function validateNumberFromText(const curText: string): integer;
function validateAddItem: boolean;
function validateCreateSipment: boolean;

procedure FormCreate(Sender: TObject);

procedure hideAllPanels;
procedure showPanel(const panel: TPanel; const x, y: integer);
procedure resetPnCreateObj;
procedure resetPnEditObj;

procedure pnSelectObjectShow(Sender: TObject; Button: TMouseButton;
    Shift: TShiftState; X, Y: Integer);
procedure pnObjectInfoShow(Sender: TObject);
procedure pnObjectInfoHide(Sender: TObject);

procedure setFilterPanel;

function cntFilteredItems: integer;

procedure btnSelectObjDeleteClick(Sender: TObject);
procedure btnSelectObjCancelClick(Sender: TObject);
procedure btnSelectObjEditClick(Sender: TObject);

```

```

procedure btnEditObjConfirmClick(Sender: TObject);
procedure btnFilterClick(Sender: TObject);
procedure btnFilterCancelClick(Sender: TObject);
procedure btnFilterDefaultClick(Sender: TObject);
procedure btnFilterConfirmClick(Sender: TObject);
procedure edFilterCapacityToValExit(Sender: TObject);
procedure edFilterCapacityToValChange(Sender: TObject);
procedure edFilterUsedCapacityToValChange(Sender: TObject);
procedure edFilterUsedCapacityToValExit(Sender: TObject);
procedure edFilterCapacityFromValChange(Sender: TObject);
procedure edFilterUsedCapacityFromValChange(Sender: TObject);
procedure edFilterCapacityFromValExit(Sender: TObject);
procedure edFilterUsedCapacityFromValExit(Sender: TObject);
procedure N11Click(Sender: TObject);
procedure btnCreateShipmentCancelClick(Sender: TObject);
procedure edCreateShipmentSenderIDExit(Sender: TObject);
procedure rbCreateShipmentSenderWarehouseClick(Sender: TObject);
procedure rbCreateShipmentSenderShopClick(Sender: TObject);
procedure edCreateShipmentDestIDExit(Sender: TObject);
procedure rbCreateShipmentDestWarehouseClick(Sender: TObject);
procedure rbCreateShipmentDestShopClick(Sender: TObject);
procedure N13Click(Sender: TObject);
procedure edCreateShipmentSenderNameExit(Sender: TObject);
procedure edCreateShipmentDestNameExit(Sender: TObject);
procedure edAddItemDestNameExit(Sender: TObject);
procedure edAddItemDestIDExit(Sender: TObject);
procedure rbAddItemTypeShopClick(Sender: TObject);
procedure rbAddItemTypeWarehouseClick(Sender: TObject);
procedure btnAddItemCancelClick(Sender: TObject);
procedure btnAddItemConfirmClick(Sender: TObject);
procedure edAddItemNameExit(Sender: TObject);
procedure edAddItemVolExit(Sender: TObject);
procedure edAddItemCntExit(Sender: TObject);
procedure ClearAddItem;
procedure btnCreateShipmentConfirmClick(Sender: TObject);
procedure ClearCreateShipment;
procedure edCreateShipmentItemNameExit(Sender: TObject);
procedure edCreateShipmentItemIDExit(Sender: TObject);
procedure N3Click(Sender: TObject);

procedure btnSelectObjItemListClick(Sender: TObject);
procedure N16Click(Sender: TObject);
procedure N6Click(Sender: TObject);
procedure btnEditObjCancelClick(Sender: TObject);
procedure N14Click(Sender: TObject);
procedure Save1Click(Sender: TObject);
procedure Open1Click(Sender: TObject);
procedure pbMapPaint(Sender: TObject);
procedure pbMapMouseMove(Sender: TObject; Shift: TShiftState; X,
    Y: Integer);
procedure FormDestroy(Sender: TObject);

procedure showArrowInfo(var arrow: PArrow; const x, y: integer);
procedure FormResize(Sender: TObject);
procedure FormClose(Sender: TObject; var Action: TCloseAction);
procedure edCreateShipmentSenderIDChange(Sender: TObject);
procedure edCreateShipmentDestIDChange(Sender: TObject);
procedure edCreateShipmentItemIDChange(Sender: TObject);
procedure edCreateShipmentCntChange(Sender: TObject);
procedure edCreateObjHouseChange(Sender: TObject);
procedure edCreateObjBuildingChange(Sender: TObject);
procedure edCreateObjCapacityChange(Sender: TObject);

```

```

procedure edFilterHouseValChange(Sender: TObject);
procedure edFilterBuildingValChange(Sender: TObject);
procedure edEditObjHouseChange(Sender: TObject);
procedure edEditObjBuildingChange(Sender: TObject);
procedure edEditObjCapacityChange(Sender: TObject);
procedure edAddItemDestIDChange(Sender: TObject);
procedure edAddItemVolChange(Sender: TObject);
procedure edAddItemCntChange(Sender: TObject);
procedure edCreateShipmentNameExit(Sender: TObject);
procedure edFilterStreetValExit(Sender: TObject);
procedure edAddItemCategoryExit(Sender: TObject);
procedure edCreateShipmentNameChange(Sender: TObject);
procedure edCreateShipmentSenderNameChange(Sender: TObject);
procedure edCreateShipmentDestNameChange(Sender: TObject);
procedure edCreateShipmentItemNameChange(Sender: TObject);
procedure edCreateObjNameChange(Sender: TObject);
procedure edCreateObjStreetChange(Sender: TObject);
procedure edEditObjNameChange(Sender: TObject);
procedure edEditObjStreetChange(Sender: TObject);
procedure edAddItemNameChange(Sender: TObject);
procedure edAddItemCategoryChange(Sender: TObject);
procedure edAddItemDestNameChange(Sender: TObject);
procedure CreateParams(var Params: TCreateParams); override;
private
{ Private declarations }
xPos, yPos: integer;
shops, warehouses: PTreapNode;
shopsNames, warehousesNames: PTreapNameNode;
filter: TFilter;
shipments: PShipment;

function getSiz(const curNode: PTreapNode): integer;
procedure writeObjData(const curFile: TextFile; const curObj: PTreapNode);
function getItemsSiz(const curItem: PTreapItemNode): integer;
procedure WriteItemData(const curFile: TextFile;
    const curObj: PTreapItemNode);
procedure createNewObjFile(const fil: textFile);
procedure ClearAllData(var shops, warehouses: PTreapNode;
    var shipments: PShipment; var shopsNames, warehousesNames: PTreapNameNode;
    var Arrows: TList<PArrow>);

public
{ Public declarations }

end;

var
frMainForm: TfrMainForm;
frTableForm: TfrTableForm;
frShipmentTableForm: TfrShipmentsTable;
frBalanceForm: TfrBalance;
frSelectShipments: TfrSelectShipments;
startWidth, startHeight: integer;
implementation

{$R *.dfm}
procedure TfrMainForm.CreateParams(var Params: TCreateParams);
begin
    inherited CreateParams(Params);
    Params.ExStyle := Params.ExStyle or WS_EX_COMPOSITED;
end;

```

```

procedure TfrMainForm.FormClose(Sender: TObject; var Action: TCloseAction);
begin
  if not Saved then
    if getConfirmation('Подтверждение', 'Сохранить изменения?') then
      begin
        Save1Click(self);
      end;
    Action := caFree;
end;

procedure TfrMainForm.FormCreate(Sender: TObject);
begin
  Randomize;
  Saved := true;
  startWidth := frMainForm.ClientWidth;
  startHeight := frMainForm.ClientHeight;
  Arrows := TList<PArrow>.Create;
  curShipmentID := 1;
  shopKey := 1;
  warehouseKey := 1;
  InitFilter(filter);
  InitTree(shops);
  InitTree(wareHouses);
  InitTreeName(shopsNames);
  InitTreeName(warehousesNames);
  InitHash(47, 40009);
  shipments := nil;
end;

procedure TfrMainForm.FormDestroy(Sender: TObject);
begin
  ClearAllData(shops, warehouses, shipments, shopsNames, warehousesNames, Arrows);
end;

procedure TfrMainForm.FormResize(Sender: TObject);
begin
  frMainForm.ClientHeight := startHeight;
  frMainForm.ClientWidth := startWidth;
end;

procedure TfrMainForm.ClearAddItem;
begin
  edAddItemName.Color := clWindow;
  edAddItemName.Text := '';
  edAddItemCategory.Text := '';
  edAddItemDestName.Text := '';
  edAddItemDestID.Text := '';
  edAddItemVol.Color := clWindow;
  edAddItemVol.Text := '';
  edAddItemCnt.Color := clWindow;
  edAddItemCnt.Text := '';
  rbAddItemTypeShop.Checked := true;
  rbAddItemTypeWarehouse.Checked := false;
end;

procedure TfrMainForm.ClearCreateShipment;
begin
  edCreateShipmentCnt.color := clWindow;
  edCreateShipmentName.Text := '';
  edCreateShipmentSenderName.Text := '';
  edCreateShipmentSenderID.Text := '';

```



```

edCreateShipmentDestName.Text := "";
edCreateShipmentDestID.Text := "";
edCreateShipmentItemName.Text := "";
edCreateShipmentItemID.Text := "";
edCreateShipmentCnt.Text := "";

rbCreateShipmentDestShop.Checked := true;
rbCreateShipmentDestWarehouse.Checked := false;
rbCreateShipmentSenderShop.Checked := false;
rbCreateShipmentSenderWarehouse.Checked := true;
end;

procedure TfrmMainForm.updateID(const editID: TEdit;
                                const isShopRadio: TRadioButton;
                                const editName: TEdit);
var
    node: PTreapNameNode;
begin
    //editName.text := "";

    if isShopRadio.Checked then
    begin
        node := FindTreapName(shopsNames, gethash(editName.text));
        if node <> nil then
            editID.text := intToStr(node^.Data^.ID xor mask)
        else
            editID.text := "";
        end
    else
    begin
        node := FindTreapName(warehousesNames, gethash(editName.text));
        if node <> nil then
            editID.text := intToStr(node^.Data^.ID)
        else
            editID.text := "";
        end;
    end;
end;

procedure TfrmMainForm.updateName(const editID: TEdit;
                                const isShopRadio: TRadioButton;
                                const editName: TEdit);
var
    node: PTreapNode;
begin
    //editID.text := "";
    if Length(editID.Text) > 0 then
    begin
        if isShopRadio.Checked then
            node := FindTreap(shops, strToInt(editID.Text) or mask)
        else
            node := FindTreap(warehouses, strToInt(editID.Text));
        if node <> nil then
            editName.text := string(node^.Data^.name)
        else
            editName.text := "";
        end;
    end;
end;

procedure TfrmMainForm.rbAddItemTypeShopClick(Sender: TObject);
begin
    if Length(edAddItemDestName.Text) > 0 then

```

```

        updateID(edAddItemDestID,
            rbAddItemTypeShop,
            edAddItemDestName)
    else
        updateName(edAddItemDestID,
            rbAddItemTypeShop,
            edAddItemDestName)
    end;

procedure TfrMainForm.rbAddItemTypeWarehouseClick(Sender: TObject);
begin
    if Length(edAddItemDestName.Text) > 0 then
        updateID(edAddItemDestID,
            rbAddItemTypeShop,
            edAddItemDestName)
    else
        updateName(edAddItemDestID,
            rbAddItemTypeShop,
            edAddItemDestName)
    end;

procedure TfrMainForm.rbCreateShipmentDestShopClick(Sender: TObject);
begin
    if Length(edCreateShipmentDestName.Text) > 0 then
        updateID(edCreateShipmentDestID,
            rbCreateShipmentDestShop,
            edCreateShipmentDestName)
    else
        updateName(edCreateShipmentDestID,
            rbCreateShipmentDestShop,
            edCreateShipmentDestName)
    end;

procedure TfrMainForm.rbCreateShipmentDestWarehouseClick(Sender: TObject);
begin
    if Length(edCreateShipmentDestName.Text) > 0 then
        updateID(edCreateShipmentDestID,
            rbCreateShipmentDestShop,
            edCreateShipmentDestName)
    else
        updateName(edCreateShipmentDestID,
            rbCreateShipmentDestShop,
            edCreateShipmentDestName)
    end;

procedure TfrMainForm.rbCreateShipmentSenderShopClick(Sender: TObject);
begin
    if Length(edCreateShipmentSenderName.Text) > 0 then
        updateID(edCreateShipmentSenderID,
            rbCreateShipmentSenderShop,
            edCreateShipmentSenderName)
    else
        updateName(edCreateShipmentSenderID,
            rbCreateShipmentSenderShop,
            edCreateShipmentSenderName);
    edCreateShipmentItemNameExit(self);
end;

procedure TfrMainForm.rbCreateShipmentSenderWarehouseClick(Sender: TObject);
begin
    if Length(edCreateShipmentSenderName.Text) > 0 then
        updateID(edCreateShipmentSenderID,

```

```

        rbCreateShipmentSenderShop,
        edCreateShipmentSenderName)
else
    updateName(edCreateShipmentSenderID,
        rbCreateShipmentSenderShop,
        edCreateShipmentSenderName);
edCreateShipmentItemNameExit(self);
end;

procedure TfrMainForm.resetPnCreateObj;
begin
    edCreateObjName.Text := "";
    edCreateObjStreet.Text := "";
    edCreateObjHouse.Text := "";
    edCreateObjBuilding.Text := "";
    edCreateObjCapacity.Text := "";
end;

procedure TfrMainForm.resetPnEditObj;
begin
    edEditObjName.Text := "";
    edEditObjStreet.Text := "";
    edEditObjHouse.Text := "";
    edEditObjBuilding.Text := "";
    edEditObjCapacity.Text := "";
end;

function TfrMainForm.getSiz(const curNode: PTreapNode) : integer;
begin
    if curNode = nil then
        Result := 0
    else
        begin
            Result := 1 + getSiz(curNode^.Left) + getSiz(curNode^.Right);
        end;
    end;
end;

function TfrMainForm.getItemsSiz(const curItem: PTreapItemNode): integer;
begin
    if curItem = nil then
        Result := 0
    else
        begin
            Result := 1 + getItemsSiz(curItem^.Left) + getItemsSiz(curItem^.Right);
        end;
    end;
end;

procedure TfrMainForm.WriteItemData(const curFile: TextFile; const curObj: PTreapItemNode);
begin
    if curObj <> nil then
        begin
            writeln(curFile, curObj^.Data^.name);
            writeln(curFile, curObj^.Data^.category);
            writeln(curFile, curObj^.Data^.Volume);
            writeln(curFile, curObj^.Data^.Count);
            writeln(curFile, curObj^.Data^.Key);
            writeln(curFile, curObj^.Data^.needToSend);
            WriteItemData(curFile, curObj^.Left);
            WriteItemData(curFile, curObj^.Right);
        end;
    end;
end;

```

```

procedure TfrMainForm.writeObjData(const curFile: TextFile; const curObj: PTreapNode);
begin
  if curObj <> nil then
    begin
      writeln(curFile, curObj^.Data^.name);
      writeln(curFile, curObj^.Data^.street);
      writeln(curFile, curObj^.Data^.house);
      writeln(curFile, curObj^.Data^.building);
      writeln(curFile, curObj^.Data^.capacity);
      writeln(curFile, curObj^.Data^.usedCapacity);
      writeln(curFile, curObj^.Data^.shipmentCapacity);
      writeln(curFile, curObj^.Data^.Key);
      writeln(curFile, curObj^.Data^.X);
      writeln(curFile, curObj^.Data^.Y);
      writeln(curFile, getItemsSiz(curObj^.Data^.Items));
      WriteItemData(curFile, curObj^.Data^.Items);
      writeObjData(curFile, curObj^.Left);
      writeObjData(curFile, curObj^.Right);
    end;
end;

procedure TfrMainForm.Save1Click(Sender: TObject);
var
  shopsFile, warehousesFile, shipmentsFile: TextFile;
  curObject: PTreapNode;
  curShip: PShipment;
  siz: integer;
begin
  if getconfirmation('Подтверждение действия', 'Сохранить данные?') then
    begin
      Saved := true;

      AssignFile(shopsFile, 'shops.txt');
      Rewrite(shopsFile);
      curObject := shops;
      writeln(shopsFile, getSiz(shops));
      writeObjData(shopsFile, curObject);
      CloseFile(shopsFile);

      AssignFile(warehousesFile, 'warehouses.txt');
      Rewrite(warehousesFile);
      curObject := warehouses;
      writeln(warehousesFile, getSiz(warehouses));
      writeObjData(warehousesFile, curObject);
      CloseFile(warehousesFile);

      AssignFile(shipmentsFile, 'shipments.txt');
      Rewrite(shipmentsFile);
      curShip := shipments;
      siz := 0;
      while curShip <> nil do
        begin
          Inc(siz);
          curShip := curShip^.next;
        end;
      writeln(shipmentsFile, siz);
      curShip := shipments;
      while curShip <> nil do
        begin
          writeln(shipmentsFile, curShip^.ShipmentName);
          writeln(shipmentsFile, curShip^.ID);
          writeln(shipmentsFile, curShip^.SourceID^.Key);
        end;
      end;
    end;
end;

```

```

        writeln(shipmentsFile, curShip^.DestinationID^.Key);
        writeln(shipmentsFile, curShip^.ProductName);
        writeln(shipmentsFile, curShip^.Count);
        curShip := curShip^.next;
    end;
    CloseFile(shipmentsFile);
    showMessage('Успешно', 'Успешно сохранено');
end;
end;

procedure TfrMainForm.showArrowInfo(var arrow: PArrow; const x, y: integer);
var
    curItem: PTreapItemNode;
begin
    lbArrowInfoShipmentName.Caption := arrow^.shipment^.ShipmentName;

    if (arrow^.shipment^.SourceID^.key and mask) <> 0 then
    begin
        lbArrowInfoSenderTypeVal.Caption := 'Магазин' ;
        lbArrowInfoSenderIDVal.Caption := intToStr(arrow^.shipment^.SourceID^.Key xor mask);
    end
    else
    begin
        lbArrowInfoSenderTypeVal.Caption := 'Склад';
        lbArrowInfoSenderIDVal.Caption := intToStr(arrow^.shipment^.SourceID^.Key);
    end;
    lbArrowInfoSenderNameVal.Caption := string(arrow^.shipment^.SourceID^.name);

    if (arrow^.shipment^.DestinationID^.key and mask) <> 0 then
    begin
        lbArrowInfoDestTypeVal.Caption := 'Магазин' ;
        lbArrowInfoDestIDVal.Caption := intToStr(arrow^.shipment^.DestinationID^.Key xor mask);
    end
    else
    begin
        lbArrowInfoDestTypeVal.Caption := 'Склад';
        lbArrowInfoDestIDVal.Caption := intToStr(arrow^.shipment^.DestinationID^.Key);
    end;
    lbArrowInfoDestNameVal.Caption := string(arrow^.shipment^.DestinationID^.name);

    lbArrowInfoItemNameVal.Caption := arrow^.shipment^.ShipmentName;
    lbArrowInfoItemIDVal.Caption := intToStr(arrow^.shipment^.ID);
    lbArrowInfoItemCntVal.Caption := intToStr(arrow^.shipment^.Count);
    curItem := FindTreapItem(arrow^.shipment^.SourceID^.Items, getHash(arrow^.shipment^.ProductName));
    lbArrowInfoItemVolVal.Caption := intToStr(arrow^.shipment^.Count * curItem^.Data^.Volume);
    showPanel(pnArrowInfo, x, y);
end;

procedure TfrMainForm.showPanel(const panel: TPanel; const x, y: integer);
begin
    //check y pos
    panel.top := y;
    if Y >= panel.height then
        panel.top := panel.top - panel.height;

    //check x pos
    panel.left := x;
    if X + panel.width > pbMap.width then
        panel.left := panel.left - panel.width;

    panel.visible := true;
end;

```

```

procedure TfrMainForm.hideAllPanels;
begin
  pnCreateObj.Visible := false;
  pnCreateSelect.Visible := false;
  pnSelectObject.Visible := false;
  pnEditObj.Visible := false;
  pnFilterParams.Visible := false;
  pnCreateShipment.Visible := false;
  pnAddItem.Visible := false;
  pnArrowInfo.Visible := false;
end;

procedure TfrMainForm.btnSelectObjCancelClick(Sender: TObject);
begin
  pnSelectObject.Visible := false;
end;

procedure TfrMainForm.btnSelectObjEditClick(Sender: TObject);
var
  curNode: PTreapNode;
begin
  hideAllPanels;

  pnEditObj.tag := pnSelectObject.tag;

  if (pnSelectObject.tag and mask) <> 0 then
  begin
    //shop
    curNode := FindTreap(shops, pnSelectObject.tag);
  end
  else
  begin
    //warehouse
    curNode := FindTreap(warehouses, pnSelectObject.tag);
  end;

  showPanel(pnEditObj,
    curNode^.Data^.shape.left + (curNode^.Data^.shape.Width shr 1),
    curNode^.Data^.shape.top + (curNode^.Data^.shape.Height shr 1)
  );

  edEditObjName.Text := string(curNode^.Data^.name);
  edEditObjStreet.Text := string(curNode^.Data^.street);
  edEditObjHouse.Text := intToStr(curNode^.Data^.house);
  if curNode^.Data^.building <> -1 then
    edEditObjBuilding.Text := intToStr(curNode^.Data^.building);
  edEditObjCapacity.Text := intToStr(curNode^.Data^.capacity);
end;

procedure TfrMainForm.btnSelectObjItemListClick(Sender: TObject);
var
  curNode: PTreapNode;
begin
  frtableForm := TfrTableForm.Create(Application);
  if (pnSelectObject.tag and mask) <> 0 then
  begin
    //shop
    curNode := FindTreap(shops, pnSelectObject.tag);
  end
  else

```

```

begin
  //warehouse
  curNode := FindTreap(warehouses, pnSelectObject.tag);
end;
frtableForm.Location := curNode^.Data;
frtableForm.LoadData;
pnSelectObject.Visible := false;
frtableForm.ShowModal;
end;

procedure TfrMainForm.btnSelectObjDeleteClick(Sender: TObject);
var
  curNode: PTreapNode;
begin
  if getconfirmation('Подтверждение действия', 'Вы подтверждаете действие?') then
  begin
    hideAllPanels;
    if (pnSelectObject.tag and mask) <> 0 then
    begin
      //shop
      curNode := FindTreap(shops, pnSelectObject.tag);
      if (curNode^.Data^.OutgoingArrows.count = 0)
        and (curNode^.Data^.IncomingArrows.count = 0) then
      begin
        FreeAndNil(curNode^.Data^.shape);
        EraseTreap(shops, pnSelectObject.tag);
        EraseTreapName(shopsNames, getHash(string(curNode^.Data^.name)));
        Saved := false;
        showMessage('Успешно', 'Магазин был удалён');
      end
      else
      begin
        showMessage('Ошибка', 'Невозможно удалить. Есть отгрузки, связанные с этим магазином!');
      end;
    end
    else
    begin
      //warehouse
      curNode := FindTreap(warehouses, pnSelectObject.tag);
      if (curNode^.Data^.OutgoingArrows.count = 0)
        and (curNode^.Data^.IncomingArrows.count = 0) then
      begin
        FreeAndNil(curNode^.Data^.shape);
        EraseTreap(warehouses, pnSelectObject.tag);
        Saved := false;
        EraseTreapName(warehousesNames, getHash(string(curNode^.Data^.name)));
        showMessage('Успешно', 'Склад был удалён');
      end
      else
      begin
        showMessage('Ошибка', 'Невозможно удалить. Есть отгрузки, связанные с этим складом!');
      end;
    end;
  end;
end;

procedure TfrMainForm.pnSelectObjectShow(Sender: TObject; Button: TMouseButton;
  Shift: TShiftState; X, Y: Integer);
begin
  hideAllPanels;

```

```

spMapPoint.Visible := false;
pnObjectInfo.Visible := false;

X := (Sender as TShape).left + ((Sender as TShape).width shr 1);
Y := (Sender as TShape).top + ((Sender as TShape).height shr 1);
showPanel(pnSelectObject, X, Y);

pnSelectObject.tag := (Sender as TShape).tag;
end;

procedure TfrMainForm.pnObjectInfoShow(Sender: TObject);
var
  curNode: PTreapNode;
begin
  pnArrowInfo.Visible := false;
  if ((Sender as TShape).tag and mask) <> 0 then
  begin
    //shop
    curNode := FindTreap(shops, (Sender as TShape).tag);
    lbObjInfoTitle.Caption := 'Магазин';
    lbObjInfoIdVal.Caption := intToStr(curNode^.Data^.Key xor mask);
  end
  else
  begin
    //warehouse
    curNode := FindTreap(warehouses, (Sender as TShape).tag);
    lbObjInfoTitle.Caption := 'Склад';
    lbObjInfoIdVal.Caption := intToStr(curNode^.Data^.Key);
  end;

  lbObjInfoNameVal.Caption := string(curNode^.Data^.name);
  lbObjInfoStreetVal.Caption := string(curNode^.Data^.street);
  lbObjInfoHouseVal.Caption := intToStr(curNode^.Data^.House);

  if curNode^.Data^.Building <> -1 then
    lbObjInfoBuildingVal.Caption := intToStr(curNode^.Data^.Building)
  else
    lbObjInfoBuildingVal.Caption := 'Отсутствует';
  lbObjInfoCapacityVal.Caption := intToStr(curNode^.Data^.Capacity);
  lbObjInfoUsedCapacityVal.Caption := intToStr(curNode^.Data^.UsedCapacity);
  showPanel(pnObjectInfo,
    (Sender as TShape).left + ((Sender as TShape).width shr 1),
    (Sender as TShape).top + ((Sender as TShape).height shr 1)
  );
end;

procedure TfrMainForm.pbMapMouseMove(Sender: TObject; Shift: TShiftState; X,
  Y: Integer);
var
  i: Integer;
  Arrow: PArrow;
  Pt: TPoint;
  nearLine: PArrow;
begin
  Pt := Point(X, Y);
  nearLine := nil;
  i := Arrows.Count - 1;
  while (i >= 0) and (nearLine = nil) do
  begin
    Arrow := Arrows[i];

```



```

    if Arrow^.Visible and IsPointNearLine(Pt, Point(Arrow^.Shipment^.SourceID^.X,
Arrow^.Shipment^.SourceID^.Y),
        Point(Arrow^.Shipment^.DestinationID^.X, Arrow^.Shipment^.DestinationID^.Y), 5) then
    begin
        Screen.Cursor := crHandPoint;
        nearLine := Arrow;
    end;
    Dec(i);
end;
if nearLine = nil then
begin
    Screen.Cursor := crDefault;
    pnArrowInfo.Visible := false;
end
else
begin
    showArrowInfo(nearLine, X, Y);
end;
end;

procedure TfrMainForm.pbMapPaint(Sender: TObject);
var
    Background: TBitmap;
    i: Integer;
    Arrow: PArrow;
    FromPt, ToPt: TPoint;
begin
    Background := TBitmap.Create;
    try
        Background.LoadFromFile('map.bmp');
        pbMap.Canvas.StretchDraw(pbMap.ClientRect, Background);

        pbMap.Canvas.Pen.Color := clBlack;
        pbMap.Canvas.Pen.Width := 2;
        for i := 0 to Arrows.Count - 1 do
            begin
                Arrow := Arrows[i];
                FromPt := Point(Arrow^.Shipment^.SourceID^.X, Arrow^.Shipment^.SourceID^.Y);
                ToPt := Point(Arrow^.Shipment^.DestinationID^.X, Arrow^.Shipment^.DestinationID^.Y);
                if Arrow^.Shipment^.SourceID^.shape.Visible and Arrow^.Shipment^.DestinationID^.shape.Visible then
                begin
                    Arrow^.Visible := true;
                    pbMap.Canvas.MoveTo(FromPt.X, FromPt.Y);
                    pbMap.Canvas.LineTo(ToPt.X, ToPt.Y);
                end
                else
                begin
                    Arrow^.Visible := false;
                end;
            end;
        finally
            Background.Free;
        end;
    end;

procedure TfrMainForm.pnObjectInfoHide(Sender: TObject);
begin
    pnObjectInfo.Visible := false;
end;

procedure TfrMainForm.createNewObj(var newObj: PLocation; const isShop: boolean);

```

```

begin
  New(newObj);
  newObj^.Items := nil;
  newObj^.name := shortString(edCreateObjName.Text);
  newObj^.street := shortString(edCreateObjStreet.Text);
  newObj^.house := strToInt(edCreateObjHouse.Text);
  newObj^.building := -1;
  if Length(edCreateObjBuilding.Text) > 0 then
    newObj^.building := strToInt(edCreateObjBuilding.Text);
  newObj^.capacity := strToInt(edCreateObjCapacity.Text);
  newObj^.usedCapacity := 0;
  newObj^.shipmentCapacity := 0;

  newObj^.X := xPos;
  newObj^.Y := yPos;

  //shape
  newObj^.shape := TShape.Create(self);
  newObj^.shape.Parent := spMapPoint.Parent;

  newObj^.shape.Width := spMapPoint.Width;
  newObj^.shape.Height := spMapPoint.Height;

  newObj^.shape.Left := xPos - newObj^.shape.width shr 1;
  newObj^.shape.Top := yPos - newObj^.shape.height shr 1;

  newObj^.shape.Shape := stCircle;

  newObj^.shape.Cursor := crHandPoint;

  if isShop then
    begin
      newObj^.key := getShopKey or mask;
      newObj^.shape.Brush.Color := shopColor;
    end
  else
    begin
      newObj^.key := getWarehouseKey;
      newObj^.shape.Brush.Color := warehouseColor;
    end;

  newObj^.shape.Tag := newObj^.key;

  newObj^.OutgoingArrows := TList<PArrow>.Create;
  newObj^.IncomingArrows := TList<PArrow>.Create;

  newObj^.shape.onMouseUp := pnSelectObjectShow;
  newObj^.shape.OnMouseEnter := pnObjectInfoShow;
  newObj^.shape.OnMouseLeave := pnObjectInfoHide;

  newObj^.shape.Visible := true;
  newObj^.shape.BringToFront;
  //endshape

end;

procedure TfrMainForm.createShop(Sender: TObject);
var
  newObj: PLocation;
  NewNode: PTreapNode;

```

```

    NewNameNode: PTreapNameNode;
begin
    createNewObj(newObj, true);

    NewNode := CreateNewNode(newObj);

    InsertTreap(shops, NewNode);

    NewNameNode := CreateNewNameNode(string(newObj^.name), newObj^.Key);
    InsertTreapName(shopsNames, NewNameNode);
end;

procedure TfrMainForm.createWarehouse(Sender: TObject);
var
    newObj: PLocation;
    NewNode: PTreapNode;
    NewNameNode: PTreapNameNode;
begin
    createNewObj(newObj, false);
    NewNode := CreateNewNode(newObj);
    InsertTreap(warehouses, NewNode);

    NewNameNode := CreateNewNameNode(string(newObj^.name), newObj^.Key);
    InsertTreapName(warehousesNames, NewNameNode);
end;

procedure TfrMainForm.edAddItemCategoryChange(Sender: TObject);
begin
    validateLengthLess70(Sender);
end;

procedure TfrMainForm.edAddItemCategoryExit(Sender: TObject);
begin
    (Sender as TEdit).Text := Trim((Sender as TEdit).Text);
end;

procedure TfrMainForm.edAddItemCntChange(Sender: TObject);
begin
    validateIntegerInput(Sender);
end;

procedure TfrMainForm.edAddItemCntExit(Sender: TObject);
begin
    if validateLength(edAddItemCnt) then
        (Sender as TEdit).color := clWindow
    end;
end;

procedure TfrMainForm.edAddItemDestIDChange(Sender: TObject);
begin
    validateIntegerInput(Sender);
end;

procedure TfrMainForm.edAddItemDestIDExit(Sender: TObject);
begin
    updateName(edAddItemDestID,
        rbAddItemTypeShop,
        edAddItemDestName);
end;

procedure TfrMainForm.edAddItemDestNameChange(Sender: TObject);
begin

```

```

    validateLengthLess70(Sender);
end;

procedure TfrMainForm.edAddItemDestNameExit(Sender: TObject);
begin
    edAddItemDestName.Text := Trim(edAddItemDestName.Text);
    updateID(edAddItemDestID,
        rbAddItemTypeShop,
        edAddItemDestName);
end;

procedure TfrMainForm.edAddItemNameChange(Sender: TObject);
begin
    validateLengthLess70(Sender);
end;

procedure TfrMainForm.edAddItemNameExit(Sender: TObject);
begin
    edAddItemName.Text := Trim(edAddItemName.Text);
    if validateLength(edAddItemName) then
        (Sender as TEdit).color := clWindow
    end;
end;

procedure TfrMainForm.edAddItemVolChange(Sender: TObject);
begin
    validateIntegerInput(Sender);
end;

procedure TfrMainForm.edAddItemVolExit(Sender: TObject);
begin
    if validateLength(edAddItemVol) then
        (Sender as TEdit).color := clWindow
    end;
end;

procedure TfrMainForm.edCreateObjBuildingChange(Sender: TObject);
begin
    validateIntegerInput(Sender);
end;

procedure TfrMainForm.edCreateObjCapacityChange(Sender: TObject);
begin
    validateIntegerInput(Sender);
end;

procedure TfrMainForm.edCreateObjHouseChange(Sender: TObject);
begin
    validateIntegerInput(Sender);
end;

procedure TfrMainForm.edCreateObjNameChange(Sender: TObject);
begin
    validateLengthLess70(Sender);
end;

procedure TfrMainForm.edCreateObjStreetChange(Sender: TObject);
begin
    validateLengthLess70(Sender);
end;

```

```

procedure TfrMainForm.edCreateShipmentCntChange(Sender: TObject);
begin
    validateIntegerInput(Sender);
end;

procedure TfrMainForm.edCreateShipmentDestIDChange(Sender: TObject);
begin
    validateIntegerInput(Sender);
end;

procedure TfrMainForm.edCreateShipmentDestIDExit(Sender: TObject);
begin
    updateName(edCreateShipmentDestID,
        rbCreateShipmentDestShop,
        edCreateShipmentDestName);
end;

procedure TfrMainForm.edCreateShipmentDestNameChange(Sender: TObject);
begin
    validateLengthLess70(Sender);
end;

procedure TfrMainForm.edCreateShipmentDestNameExit(Sender: TObject);
begin
    edCreateShipmentDestName.Text := Trim(edCreateShipmentDestName.Text);
    updateID(edCreateShipmentDestID,
        rbCreateShipmentDestShop,
        edCreateShipmentDestName);
end;

procedure TfrMainForm.edCreateShipmentItemIDChange(Sender: TObject);
begin
    validateIntegerInput(Sender);
end;

procedure TfrMainForm.edCreateShipmentItemIDExit(Sender: TObject);
var
    senderNode: PTreapNode;
    itemNode: PTreapItemNode;
begin
    if Length(edCreateShipmentSenderID.Text) > 0 then
        begin
            if rbCreateShipmentSenderShop.Checked then
                senderNode := FindTreap(shops, strToInt(edCreateShipmentSenderID.Text) or mask)
            else
                senderNode := FindTreap(warehouses, strToInt(edCreateShipmentSenderID.Text));
            itemNode := nil;
            if (Length(edCreateShipmentItemID.Text) > 0) and (senderNode <> nil) then
                itemNode := FindTreapItem(senderNode^.Data^.Items, strToInt(edCreateShipmentItemID.Text));
            if itemNode <> nil then
                edCreateShipmentItemName.Text := string(itemNode^.Data^.name)
            else
                edCreateShipmentItemName.Text := "";
            end
        end
    else
        begin
            edCreateShipmentItemName.Text := "";
        end
    end;
end;

```

```

procedure TfrMainForm.edCreateShipmentItemNameChange(Sender: TObject);
begin
    validateLengthLess70(Sender);
end;

procedure TfrMainForm.edCreateShipmentItemNameExit(Sender: TObject);
var
    senderNode: PTreapNode;
    itemNode: PTreapItemNode;
begin
    edCreateShipmentItemName.Text := Trim(edCreateShipmentItemName.Text);
    if Length(edCreateShipmentSenderID.Text) > 0 then
    begin
        if rbCreateShipmentSenderShop.Checked then
            senderNode := FindTreap(shops, strToInt(edCreateShipmentSenderID.Text) or mask)
        else
            senderNode := FindTreap(warehouses, strToInt(edCreateShipmentSenderID.Text));
        itemNode := nil;
        if senderNode <> nil then
            itemNode := FindTreapItem(senderNode^.Data^.Items, getHash(edCreateShipmentItemName.Text));
        if itemNode <> nil then
            edCreateShipmentItemID.Text := intToStr(getHash(edCreateShipmentItemName.Text))
        else
            edCreateShipmentItemID.Text := "";
        end
        else
        begin
            edCreateShipmentItemID.Text := "";
        end;
    end;
end;

procedure TfrMainForm.edCreateShipmentNameChange(Sender: TObject);
begin
    validateLengthLess70(Sender);
end;

procedure TfrMainForm.edCreateShipmentNameExit(Sender: TObject);
begin
    edCreateShipmentName.Text := Trim(edCreateShipmentName.Text);
end;

procedure TfrMainForm.edCreateShipmentSenderIDChange(Sender: TObject);
begin
    validateIntegerInput(Sender);
end;

procedure TfrMainForm.edCreateShipmentSenderIDExit(Sender: TObject);
begin
    updateName(edCreateShipmentSenderID,
        rbCreateShipmentSenderShop,
        edCreateShipmentSenderName);
    edCreateShipmentItemNameExit(self);
end;

procedure TfrMainForm.edCreateShipmentSenderNameChange(Sender: TObject);
begin
    validateLengthLess70(Sender);
end;

```

```

procedure TfrMainForm.edCreateShipmentSenderNameExit(Sender: TObject);
begin
    edCreateShipmentSenderName.Text := Trim(edCreateShipmentSenderName.Text);
    updateID(edCreateShipmentSenderID,
        rbCreateShipmentSenderShop,
        edCreateShipmentSenderName);
    edCreateShipmentItemNameExit(self);
end;

procedure TfrMainForm.edEditObjBuildingChange(Sender: TObject);
begin
    validateIntegerInput(Sender);
end;

procedure TfrMainForm.edEditObjCapacityChange(Sender: TObject);
begin
    validateIntegerInput(Sender);
end;

procedure TfrMainForm.edEditObjHouseChange(Sender: TObject);
begin
    validateIntegerInput(Sender);
end;

procedure TfrMainForm.edEditObjNameChange(Sender: TObject);
begin
    validateLengthLess70(Sender);
end;

procedure TfrMainForm.edEditObjStreetChange(Sender: TObject);
begin
    validateLengthLess70(Sender);
end;

procedure TfrMainForm.edFilterBuildingValChange(Sender: TObject);
begin
    validateIntegerInput(Sender);
end;

procedure TfrMainForm.edFilterCapacityFromValChange(Sender: TObject);
begin
    validateIntegerInput(Sender);
    if validateFromTo(edFilterCapacityFromVal, edFilterCapacityToVal) then
        edFilterCapacityToVal.color := clWindow;
end;

procedure TfrMainForm.edFilterCapacityFromValExit(Sender: TObject);
begin
    if not validateFromTo(edFilterCapacityFromVal, edFilterCapacityToVal) then
        edFilterCapacityToVal.color := clRed;
end;

procedure TfrMainForm.edFilterCapacityToValChange(Sender: TObject);
begin
    validateIntegerInput(Sender);
    if validateFromTo(edFilterCapacityFromVal, edFilterCapacityToVal) then
        edFilterCapacityToVal.color := clWindow;
end;

procedure TfrMainForm.edFilterCapacityToValExit(Sender: TObject);
begin
    if not validateFromTo(edFilterCapacityFromVal, edFilterCapacityToVal) then

```

```

    edFilterCapacityToVal.color := clRed;
end;

procedure TfrMainForm.edFilterHouseValChange(Sender: TObject);
begin
    validateIntegerInput(Sender);
end;

procedure TfrMainForm.edFilterStreetValExit(Sender: TObject);
begin
    edFilterStreetVal.Text := Trim(edFilterStreetVal.Text);
end;

procedure TfrMainForm.edFilterUsedCapacityFromValChange(Sender: TObject);
begin
    validateIntegerInput(Sender);
    if validateFromTo(edFilterUsedCapacityFromVal, edFilterUsedCapacityToVal) then
        edFilterUsedCapacityToVal.color := clWindow;
end;

procedure TfrMainForm.edFilterUsedCapacityFromValExit(Sender: TObject);
begin
    if not validateFromTo(edFilterUsedCapacityFromVal, edFilterUsedCapacityToVal) then
        edFilterUsedCapacityToVal.color := clRed;
end;

procedure TfrMainForm.edFilterUsedCapacityToValChange(Sender: TObject);
begin
    validateIntegerInput(Sender);
    if validateFromTo(edFilterUsedCapacityFromVal, edFilterUsedCapacityToVal) then
        edFilterUsedCapacityToVal.color := clWindow;
end;

procedure TfrMainForm.edFilterUsedCapacityToValExit(Sender: TObject);
begin
    if not validateFromTo(edFilterUsedCapacityFromVal, edFilterUsedCapacityToVal) then
        edFilterUsedCapacityToVal.color := clRed;
end;

procedure TfrMainForm.OnClickValidateLetters(Sender: TObject);
begin
    validateLengthLess70(Sender);
    if validateLetters(Sender) then
        begin
            (Sender as TEdit).color := clWindow;
        end;
end;

procedure TfrMainForm.createNewObjFile(const fil: textFile);
var
    i, j, cntObj, cntItems: integer;
    newObj: PLocation;
    curItem: PItem;
    newNode: PTreapNode;
    newNameNode: PTreapNameNode;
    newItemNode: PTreapItemNode;
begin
    readln(fil, cntObj);
    for i := 1 to cntObj do
        begin
            newObj := new(PLocation);

```



```

newObj^.Items := nil;
readln(fil, newObj^.name);
readln(fil, newObj^.street);
readln(fil, newObj^.house);
readln(fil, newObj^.building);
readln(fil, newObj^.capacity);
readln(fil, newObj^.usedCapacity);
readln(fil, newObj^.shipmentCapacity);
readln(fil, newObj^.key);
readln(fil, newObj^.x);
readln(fil, newObj^.y);
readln(fil, cntItems);
newObj^.shape := TShape.Create(self);

newObj^.shape.Parent := spMapPoint.Parent;
newObj^.shape.Width := spMapPoint.Width;
newObj^.shape.Height := spMapPoint.Height;

newObj^.shape.Left := newObj^.x - newObj^.shape.width shr 1;
newObj^.shape.Top := newObj^.y - newObj^.shape.height shr 1;

newObj^.shape.Shape := stCircle;

newObj^.shape.Cursor := crHandPoint;

if (newObj^.Key and mask) = 0 then
begin
  if newObj^.Key >= warehouseKey then
    warehouseKey := newObj^.Key + 1;
    newObj^.shape.Brush.Color := warehouseColor;
  end
else
begin
  if (newObj^.Key xor mask) >= shopKey then
    shopKey := (newObj^.Key xor mask) + 1;
    newObj^.shape.Brush.Color := shopColor;
  end;
end;

newObj^.shape.Tag := newObj^.key;

newObj^.OutgoingArrows := TList<PArrow>.Create;
newObj^.IncomingArrows := TList<PArrow>.Create;

newObj^.shape.onMouseUp := pnSelectObjectShow;
newObj^.shape.OnMouseEnter := pnObjectInfoShow;
newObj^.shape.OnMouseLeave := pnObjectInfoHide;

newObj^.shape.Visible := true;
newObj^.shape.BringToFront;
for j := 1 to cntItems do
begin
  curItem := new(PItem);
  readln(fil, curItem^.name);
  readln(fil, curItem^.category);
  readln(fil, curItem^.Volume);
  readln(fil, curItem^.Count);
  readln(fil, curItem^.Key);
  readln(fil, curItem^.needToSend);
  newItemNode := CreateNewItemNode(curItem);
  InsertTreapItem(newObj^.Items, newItemNode);
end;

```

```

    if (newObj^.Key and mask) <> 0 then
    begin
        NewNode := CreateNewNode(newObj);
        InsertTreap(shops, NewNode);

        NewNameNode := CreateNewNameNode(string(newObj^.name), newObj^.Key);
        InsertTreapName(shopsNames, NewNameNode);
    end
    else
    begin
        NewNode := CreateNewNode(newObj);
        InsertTreap(warehouses, NewNode);

        NewNameNode := CreateNewNameNode(string(newObj^.name), newObj^.Key);
        InsertTreapName(warehousesNames, NewNameNode);
    end;
end;

procedure TfrMainForm.ClearAllData(var shops, warehouses: PTreapNode;
    var shipments: PShipment;
    var shopsNames, warehousesNames: PTreapNameNode;
    var Arrows: TList<PArrow>);
begin
    ClearTreap(shops);
    ClearTreap(warehouses);
    ClearShipments(shipments);
    ClearTreapName(shopsNames);
    ClearTreapName(warehousesNames);
    Arrows.Free;
end;

procedure TfrMainForm.Open1Click(Sender: TObject);
var
    shopsFile, warehousesFile, shipmentsFile: textFile;
    cntShipments, i, sourceKey, destKey: integer;
    curShipment: PShipment;
begin
    if not FileExists('warehouses.txt')
    or not FileExists('shops.txt')
    or not FileExists('shipments.txt') then
    begin
        showMessage('Ошибка', 'Ошибка загрузки. Одного/нескольких файлов не существует!');
    end
    else
    begin
        if getconfirmation('Подтверждение действия', 'Вы подтверждаете действие?') then
        begin
            curShipmentID := 1;
            shopKey := 1;
            warehouseKey := 1;

            ClearAllData(shops, warehouses, shipments, shopsNames, warehousesNames, Arrows);

            shops := nil;
            warehouses := nil;
            shipments := nil;
            shopsNames := nil;
            warehousesNames := nil;
            AssignFile(warehousesFile, 'warehouses.txt');
            Reset(warehousesFile);
            createNewObjFile(warehousesFile);
        end
        else
            exit;
        end
    end
end;

```

```

CloseFile(warehousesFile);

AssignFile(shopsFile, 'shops.txt');
Reset(shopsFile);
createNewObjFile(shopsFile);
CloseFile(shopsFile);

AssignFile(shipmentsFile, 'shipments.txt');
Reset(shipmentsFile);
readln(shipmentsFile, cntShipments);
curShipmentID := cntShipments + 1;

Arrows := TList<PArrow>.Create;

for i := 1 to cntShipments do
begin
  curShipment := new(PShipment);
  readln(shipmentsFile, curShipment^.ShipmentName);
  readln(shipmentsFile, curShipment^.ID);
  readln(shipmentsFile, sourceKey);
  readln(shipmentsFile, destKey);
  if (sourceKey and mask) <> 0 then
    curShipment^.SourceID := FindTreap(shops, sourceKey).Data
  else
    curShipment^.SourceID := FindTreap(warehouses, sourceKey).Data;

  if (destKey and mask) <> 0 then
    curShipment^.DestinationID := FindTreap(shops, destKey).Data
  else
    curShipment^.DestinationID := FindTreap(warehouses, destKey).Data;

  AddArrow(Arrows, curShipment);
  pbMap.Invalidate;

  readln(shipmentsFile, curShipment^.ProductName);
  readln(shipmentsFile, curShipment^.Count);
  curShipment^.next := shipments;
  shipments := curShipment;
end;
CloseFile(shipmentsFile);
spMapPoint.Parent.Invalidate;
showMessage('Успешно', 'Данные успешно загружены');
end;
end;
end;

function TfrMainForm.validateNumberFromText(const curText: string): integer;
begin
  if Length(curText) > 0 then
    Result := strToInt(curText)
  else
    Result := -1;
end;

procedure TfrMainForm.OnClickValidateLength(Sender: TObject);
begin
  edCreateObjName.Text := Trim(edCreateObjName.Text);
  edEditObjName.Text := Trim(edEditObjName.Text);

  if validateLength(Sender) then
    (Sender as TEdit).color := clWindow;
end;

```

```

procedure TfrMainForm.OnClickvalidateAll(Sender: TObject);
begin
  edCreateObjStreet.Text := Trim(edCreateObjStreet.Text);
  edEditObjStreet.Text := trim(edEditObjStreet.Text);

  if (validateLength(Sender)) and (validateLetters(Sender)) then
  begin
    (Sender as TEdit).color := clWindow;
  end;
end;

function TfrMainForm.validateCreateObj: boolean;
begin
  Result := validateLength(edCreateObjName);

  Result := validateLength(edCreateObjStreet) and Result;

  Result := validateLetters(edCreateObjStreet) and Result;

  Result := validateLength(edCreateObjHouse) and Result;

  Result := validateLength(edCreateObjCapacity) and Result;
end;

function TfrMainForm.validateEditObj: boolean;
begin
  Result := validateLength(edEditObjName);

  Result := validateLength(edEditObjStreet) and Result;

  Result := validateLetters(edEditObjStreet) and Result;

  Result := validateLength(edEditObjHouse) and Result;

  Result := validateLength(edEditObjCapacity) and Result;
end;

function TfrMainForm.validateAddItem: Boolean;
var
  node: PTreapNode;
  itemNode, senderItemNode: PTreapItemNode;
  category: string;
  shipmentCapacity: integer;
  curShip: PShipment;
begin
  Result := true;
  Result := validateLength(edAddItemName) and Result;
  Result := validateLength(edAddItemVol) and Result;
  Result := validateLength(edAddItemCnt) and Result;
  if (Length(edAddItemDestName.Text) = 0) or (Length(edAddItemDestID.Text) = 0) then
  begin
    showMessage('Ошибка', 'Получателя с таким названием/ID не существует!');
    Result := false;
  end;

  if Result then
  begin
    if rbAddItemTypeShop.Checked then
      node := FindTreap(shops, strToInt(edAddItemDestID.Text) or mask)
    else
      node := FindTreap(warehouses, strToInt(edAddItemDestID.Text));
  end;
end;

```

```

itemNode := FindTreapItem(node^.Data^.Items, getHash(edAddItemName.Text));
if itemNode <> nil then
begin
  if strToInt(edAddItemVol.Text) <> itemNode^.Data^.Volume then
  begin
    Result := false;
    showMessage('Ошибка', 'Товар уже существует в магазине,'
      + ' но объем за единицу товара отличается!'
      + ' Объем уже существующего товара: '
      + intToStr(itemNode^.Data^.Volume));
  end;
  if edAddItemCategory.Text <> string(itemNode^.Data^.category) then
  begin
    if Length(edAddItemCategory.Text) = 0 then
      category := 'Отсутствует'
    else
      category := edAddItemCategory.Text;
    Result := false;
    showMessage('Ошибка', 'Товар уже существует в магазине,'
      + ' но категория отличается!'
      + ' Категория уже существующего товара: '
      + category);
  end
end;

curShip := Shipments;
shipmentCapacity := 0;
while curShip <> nil do
begin
  if curShip^.DestinationID = node^.Data then
  begin
    senderItemNode := FindTreapItem(curShip^.SourceID^.Items, getHash(curShip^.ProductName));
    shipmentCapacity := shipmentCapacity + curShip^.Count
      * senderItemNode^.Data^.Volume;
  end;
  curShip := curShip^.next;
end;

if Result and
  (node^.Data^.usedCapacity
  + strToInt(edAddItemCnt.Text) * strToInt(edAddItemVol.Text)
  + shipmentCapacity
  > node^.Data^.capacity) then
begin
  Result := false;
  showMessage('Ошибка', 'У получателя недостаточно свободного места!'
    + ' Свободное место: '
    + intToStr(node^.Data^.capacity - node^.Data^.usedCapacity)
    + ' у.е. Из них '
    + intToStr(node^.Data^.shipmentCapacity)
    + ' зарезервировано под отгрузку');
end;

end;
end;

function TfrMainForm.validateCreateSipment: boolean;
var
  senderNode, destNode: PTreapNode;
  curItem, senderItemNode, destItemNode: PTreapItemNode;
begin

```

```

Result := true;

if (Length(edCreateShipmentSenderName.Text) = 0)
or (Length(edCreateShipmentSenderID.Text) = 0) then
begin
  showMessage('Ошибка', 'Отправителя с таким названием/ID не существует!');
  Result := false;
end;

if Result and ((Length(edCreateShipmentDestName.Text) = 0)
or (Length(edCreateShipmentDestID.Text) = 0)) then
begin
  showMessage('Ошибка', 'Получателя с таким названием/ID не существует!');
  Result := false;
end;

if Result and ((Length(edCreateShipmentItemName.Text) = 0)
or (Length(edCreateShipmentItemID.Text) = 0)) then
begin
  showMessage('Ошибка', 'Товара с таким названием/артикулом не существует!');
  Result := false;
end;

if Result and (edCreateShipmentDestID.Text = edCreateShipmentSenderID.Text)
and ((rbCreateShipmentSenderWarehouse.Checked and rbCreateShipmentDestWarehouse.checked)
or (not rbCreateShipmentSenderWarehouse.Checked and not rbCreateShipmentDestWarehouse.checked)) then
begin
  showMessage('Ошибка', 'Отправитель и получатель не должны совпадать!');
  Result := false;
end;

senderNode := nil;
if Result then
begin
  if rbCreateShipmentSenderShop.Checked then
    senderNode := FindTreap(shops, strToInt(edCreateShipmentSenderID.Text) or mask)
  else
    senderNode := FindTreap(warehouses, strToInt(edCreateShipmentSenderID.Text));

  curItem := FindTreapItem(senderNode^.Data^.Items, strToInt(edCreateShipmentItemID.Text));
  if curItem^.Data^.Count - curItem^.Data^.needToSend
    < strToInt(edCreateShipmentCnt.Text) then
    begin
      showMessage('Ошибка', 'У отправителя недостаточное количество товара!');
      Result := false;
    end;
end;

destNode := nil;

if (senderNode <> nil) and Result then
begin
  if rbCreateShipmentDestShop.Checked then
    destNode := FindTreap(shops, strToInt(edCreateShipmentDestID.Text) or mask)
  else
    destNode := FindTreap(warehouses, strToInt(edCreateShipmentDestID.Text));

  if destNode^.Data^.usedCapacity + destNode^.Data^.shipmentCapacity
    + strToInt(edCreateShipmentCnt.Text) * FindTreapItem(senderNode^.Data^.Items,
strToInt(edCreateShipmentItemID.Text))^Data^.Volume
    > destNode^.Data^.capacity then
    begin

```

```

    showMessage('Ошибка', 'У получателя не хватает места!');
    Result := false;
end;
end;

if Result and (senderNode <> nil) and (destNode <> nil) then
begin
    senderItemNode := nil;
    destItemNode := nil;
    if Length(edCreateShipmentItemID.Text) > 0 then
    begin
        senderItemNode := FindTreapItem(senderNode^.Data^.Items, strToInt(edCreateShipmentItemID.Text));
        destItemNode := FindTreapItem(destNode^.Data^.Items, strToInt(edCreateShipmentItemID.Text));
    end;
    if (senderItemNode <> nil) and (destItemNode <> nil) then
    begin
        if senderItemNode^.Data^.category <> destItemNode^.Data^.category then
        begin
            showMessage('Ошибка', 'Категории текущего товара у данных объектов не совпадают!');
            Result := false;
        end;
        if Result and (senderItemNode^.Data^.Volume <> destItemNode^.Data^.Volume) then
        begin
            showMessage('Ошибка', 'Объемы единицы текущего товара у данных объектов не совпадают!');
            Result := false;
        end;
    end;
end;
Result := validateLength(edCreateShipmentCnt) and Result;
end;

procedure TfrMainForm.btnCreateSelectClick(Sender: TObject);
begin
    hideAllPanels;

    pnCreateObj.BringToFront;

    edCreateObjName.color := clWindow;
    edCreateObjStreet.color := clWindow;
    edCreateObjHouse.color := clWindow;
    edCreateObjCapacity.color := clWindow;

    //check y pos
    pnCreateObj.top := yPos;
    if yPos >= pnCreateObj.height then
        pnCreateObj.top := pnCreateObj.top - pnCreateObj.height;

    //check x pos
    pnCreateObj.left := xPos;
    if xPos + pnCreateObj.width > pbMap.width then
        pnCreateObj.left := pnCreateObj.left - pnCreateObj.width;

    pnCreateObj.tag := (Sender as TButton).tag; //tag=1 - shop, tag=2 - Warehouse
    pnCreateObj.visible := true;
end;

procedure TfrMainForm.btnCreateShipmentCancelClick(Sender: TObject);
begin
    pnCreateShipment.Visible := false;
    ClearCreateShipment;
end;

```

```

procedure TfrMainForm.btnCreateShipmentConfirmClick(Sender: TObject);
var
  newShipment: PShipment;
  curItem: PTreapItemNode;
  senderNode: PTreapNode;
begin
  if validateCreateSipment then
  begin
    newShipment := new(PShipment);
    if rbCreateShipmentSenderShop.Checked then
      senderNode := FindTreap(shops, strToInt(edCreateShipmentSenderID.Text) or mask)
    else
      senderNode := FindTreap(warehouses, strToInt(edCreateShipmentSenderID.Text));
    curItem := FindTreapItem(senderNode^.Data^.Items, strToInt(edCreateShipmentItemID.Text));
    Inc(curItem^.Data^.needToSend, strToInt(edCreateShipmentCnt.Text));
    if rbCreateShipmentSenderShop.Checked then
      newShipment^.SourceID := FindTreap(shops, strToInt(edCreateShipmentSenderID.Text) or mask)^.Data
    else
      newShipment^.SourceID := FindTreap(warehouses, strToInt(edCreateShipmentSenderID.Text))^.Data;

    if rbCreateShipmentDestShop.Checked then
      newShipment^.DestinationID := FindTreap(shops, strToInt(edCreateShipmentDestID.Text) or mask)^.Data
    else
      newShipment^.DestinationID := FindTreap(warehouses, strToInt(edCreateShipmentDestID.Text))^.Data;

    newShipment^.ID := curShipmentId;
    Inc(curShipmentId);
    newShipment^.ShipmentName := edCreateShipmentName.Text;
    newShipment^.ProductName := edCreateShipmentItemName.Text;
    newShipment^.Count := strToInt(edCreateShipmentCnt.Text);
    newShipment^.next := shipments;
    shipments := newShipment;
    pnCreateShipment.Visible := false;
    ClearCreateShipment;
    AddArrow(Arrows, newShipment);

    Inc(newShipment^.DestinationID^.shipmentCapacity, newShipment^.Count
      * curItem^.Data^.Volume);

    Saved := false;

    pbMap.Invalidate;
  end;

end;

procedure TfrMainForm.btnEditObjCancelClick(Sender: TObject);
begin
  pnEditObj.Visible := false;
end;

procedure TfrMainForm.btnEditObjConfirmClick(Sender: TObject);
var
  curNode: PTreapNode;
begin
  if (pnEditObj.tag and mask) <> 0 then
  begin
    //shop
    curNode := FindTreap(shops, pnSelectObject.tag);
  end
  else

```



```

begin
  //warehouse
  curNode := FindTreap(warehouses, pnSelectObject.tag);
end;

if validateEditObj then
begin
  if strToInt(edEditObjCapacity.Text) < curNode^.Data^.usedCapacity
    + curNode^.Data^.shipmentCapacity then
    begin
      showMessage('Ошибка', 'Новая вместимость не позволяет вместить существующие товары!');
    end
  else if getconfirmation('Подтверждение действия', 'Вы подтверждаете действие?') then
    begin
      edEditObjName.Text := trim(edEditObjName.Text);
      edEditObjStreet.Text := trim(edEditObjStreet.Text);

      curNode^.Data^.name := shortString(edEditObjName.Text);
      curNode^.Data^.street := shortString(edEditObjStreet.Text);
      curNode^.Data^.house := strToInt(edEditObjHouse.Text);
      curNode^.Data^.building := -1;
      if Length(edEditObjBuilding.Text) > 0 then
        curNode^.Data^.building := strToInt(edEditObjBuilding.Text);
      curNode^.Data^.capacity := strToInt(edEditObjCapacity.Text);

      edEditObjName.Text := "";
      edEditObjStreet.Text := "";
      edEditObjHouse.Text := "";
      edEditObjBuilding.Text := "";
      edEditObjCapacity.Text := "";
      hideAllPanels;

      Saved := false;
    end;
  end;
end;

procedure TfrMainForm.setFilterPanel;
begin
  edFilterStreetVal.Text := "";
  edFilterHouseVal.Text := "";
  edFilterBuildingVal.Text := "";
  edFilterCapacityFromVal.Text := "";
  edFilterCapacityToVal.Text := "";
  edFilterUsedCapacityFromVal.Text := "";
  edFilterUsedCapacityToVal.Text := "";

  cbFilterTypeShop.Checked := true;
  cbFilterTypeWarehouse.Checked := true;
end;

procedure TfrMainForm.btnFilterCancelClick(Sender: TObject);
begin
  pnFilterParams.Visible := false;
end;

procedure TfrMainForm.btnFilterClick(Sender: TObject);
begin
  hideAllPanels;
  spMapPoint.Visible := false;

  cbFilterTypeShop.Checked := false;

```

```

cbFilterTypeWarehouse.Checked := false;
edFilterStreetVal.Text := "";
edFilterHouseVal.Text := "";
edFilterBuildingVal.Text := "";
edFilterCapacityFromVal.Text := "";
edFilterCapacityToVal.Text := "";
edFilterUsedCapacityFromVal.Text := "";
edFilterUsedCapacityToVal.Text := "";

if (filter.buildingType and 1) <> 0 then
    cbFilterTypeShop.Checked := true;
if (filter.buildingType and 2) <> 0 then
    cbFilterTypeWarehouse.Checked := true;
if Length(filter.street) > 0 then
    edFilterStreetVal.Text := string(filter.street);
if filter.house <> -1 then
    edFilterHouseVal.Text := intToStr(filter.house);
if filter.building <> -1 then
    edFilterBuildingVal.Text := intToStr(filter.building);
if filter.capacityFrom <> -1 then
    edFilterCapacityFromVal.Text := intToStr(filter.capacityFrom);
if filter.capacityTo <> -1 then
    edFilterCapacityToVal.Text := intToStr(filter.capacityTo);
if filter.usedCapacityFrom <> -1 then
    edFilterUsedCapacityFromVal.Text := intToStr(filter.usedCapacityFrom);
if filter.usedCapacityTo <> -1 then
    edFilterUsedCapacityToVal.Text := intToStr(filter.usedCapacityTo);
pnFilterParams.left := (pnMapWrap.width - pnFilterParams.width) shr 1;
pnFilterParams.top := (pnMapWrap.height - pnFilterParams.height) shr 1;
pnFilterParams.Visible := true;
end;

function TfrMainForm.cntFilteredItems: integer;
begin
    Result := 0;
    if not cbFilterTypeShop.Checked then
        Inc(Result);
    if not cbFilterTypeWarehouse.Checked then
        Inc(Result);
    if Length(edFilterStreetVal.Text) > 0 then
        Inc(Result);
    if Length(edFilterHouseVal.Text) > 0 then
        Inc(Result);
    if Length(edFilterBuildingVal.Text) > 0 then
        Inc(Result);
    if Length(edFilterCapacityFromVal.Text) > 0 then
        Inc(Result);
    if Length(edFilterCapacityToVal.Text) > 0 then
        Inc(Result);
    if Length(edFilterUsedCapacityFromVal.Text) > 0 then
        Inc(Result);
    if Length(edFilterUsedCapacityToVal.Text) > 0 then
        Inc(Result);
end;

procedure TfrMainForm.btnFilterConfirmClick(Sender: TObject);
var
    cntFilter: integer;
    objType: integer;
begin
    if (validateLetters(edFilterStreetVal))
        and ((validateFromTo(edFilterCapacityFromVal, edFilterCapacityToVal))

```

```

    and (validateFromTo(edFilterUsedCapacityFromVal, edFilterUsedCapacityToVal)))
then
begin
    cntFilter := cntFilteredItems;
    objType := 0;
    if cbFilterTypeShop.checked then
        objType := objType or 1;
    if cbFilterTypeWarehouse.checked then
        objType := objType or 2;

    //передать -1 в числовое поле, если пусто
    createFilter(filter, objType,
        edFilterStreetVal.text,
        validateNumberFromText(edFilterHouseVal.Text),
        validateNumberFromText(edFilterBuildingVal.Text),
        validateNumberFromText(edFilterCapacityFromVal.Text),
        validateNumberFromText(edFilterCapacityToVal.Text),
        validateNumberFromText(edFilterUsedCapacityFromVal.Text),
        validateNumberFromText(edFilterUsedCapacityToVal.Text)
    );
    ApplyFilter(shops, filter);
    ApplyFilter(warehouses, filter);
    pbMap.Invalidate;
    if cntFilter <> 0 then
        btnFilter.Caption := 'Фильтр (' + intToStr(cntFilter) + ')'
    else
        btnFilter.Caption := 'Фильтр';
    hideAllPanels;
end;
end;

procedure TfrMainForm.btnFilterDefaultClick(Sender: TObject);
begin
    setFilterPanel;
end;

procedure TfrMainForm.btnCreateObjConfirmClick(Sender: TObject);
begin
    if validateCreateObj then
    begin
        if (
            (pnCreateObj.tag = 1) and (FindTreapName(shopsNames, getHash(edCreateObjName.Text)) <> nil)
            or
            (pnCreateObj.tag = 2) and (FindTreapName(warehousesNames, getHash(edCreateObjName.Text)) <> nil)
        ) then
        begin
            if pnCreateObj.tag = 1 then
            begin
                showMessage('Ошибка', 'Магазин с таким именем уже существует');
            end
            else
            begin
                showMessage('Ошибка', 'Склад с таким именем уже существует');
            end;
        end
        else
        begin
            edCreateObjName.Text := trim(edCreateObjName.Text);
            edCreateObjStreet.Text := trim(edCreateObjStreet.Text);

            if pnCreateObj.tag = 1 then
            begin

```

```

        //create shop
        createShop(Sender);
        Saved := false;
    end
    else if pnCreateObj.tag = 2 then
    begin
        //create warehouse
        createWarehouse(Sender);
        Saved := false;
    end
    else
    begin
        //error
    end;
    resetPnCreateObj;
    hideAllPanels;
    spMapPoint.visible := false;

    ApplyFilter(shops, filter);
    ApplyFilter(warehouses, filter);
    pbMap.Invalidate;
end;
end;
end;

procedure TfrMainForm.btnCreateSelectCancelClick(Sender: TObject);
begin
    pnCreateSelect.Visible := false;
    spMapPoint.visible := false;
end;

procedure TfrMainForm.btnAddItemCancelClick(Sender: TObject);
begin
    pnAddItem.Visible := false;
    ClearAddItem;
end;

procedure TfrMainForm.btnAddItemConfirmClick(Sender: TObject);
var
    node: PTreapNode;
    newItem: PItem;
    itemNode, newItemNode: PTreapItemNode;
begin
    if validateAddItem then
    begin
        if rbAddItemTypeShop.Checked then
            node := FindTreap(shops, strToInt(edAddItemDestID.Text) or mask)
        else
            node := FindTreap(warehouses, strToInt(edAddItemDestID.Text));

        itemNode := FindTreapItem(node^.Data^.Items, getHash(edAddItemName.Text));

        if itemNode <> nil then
            itemNode^.Data^.Count := itemNode^.Data^.Count + strToInt(edAddItemCnt.Text)
        else
        begin
            newItem := new(PItem);
            newItem^.name := shortString(edAddItemName.Text);
            newItem^.needToSend := 0;
            newItem^.category := shortString(edAddItemCategory.Text);
            newItem^.Volume := strToInt(edAddItemVol.Text);

```

```

    newItem^.Count := strToInt(edAddItemCnt.Text);
    newItem^.Key := getHash(string(newItem^.name));

    newItemNode := CreateNewItemNode(newItem);
    InsertTreapItem(node^.Data^.Items, newItemNode);
end;

node^.Data^.usedCapacity := node^.Data^.usedCapacity +
    strToInt(edAddItemCnt.Text)
    * strToInt(edAddItemVol.Text);
pnAddItem.Visible := false;
Saved := false;
ClearAddItem;
end;
end;

procedure TfrMainForm.btnCreateObjCancelClick(Sender: TObject);
begin
    pnCreateObj.Visible := false;
    edCreateObjName.Text := "";
    edCreateObjStreet.Text := "";
    edCreateObjHouse.Text := "";
    edCreateObjBuilding.Text := "";
    edCreateObjCapacity.Text := "";
    spMapPoint.visible := false;
end;

procedure TfrMainForm.imgMapMouseDown(Sender: TObject; Button: TMouseButton;
    Shift: TShiftState; X, Y: Integer);
begin
    hideAllPanels;
    resetPnCreateObj;
    resetPnEditObj;

    xPos := X;
    yPos := Y;
    pnCreateSelect.BringToFront;
    showPanel(pnCreateSelect, xPos, yPos);

    //print green point
    spMapPoint.top := Y - spMapPoint.height shr 1;
    spMapPoint.left := X - spMapPoint.width shr 1;
    spMapPoint.BringToFront;
    spMapPoint.visible := true;
end;

procedure TfrMainForm.N11Click(Sender: TObject);
begin
    hideAllPanels;

    pnCreateShipment.left := (pnMapWrap.width - pnCreateShipment.width) shr 1;
    pnCreateShipment.top := (pnMapWrap.height - pnCreateShipment.height) shr 1;

    rbCreateShipmentSenderWarehouse.Checked := true;
    rbCreateShipmentDestShop.Checked := true;

    spMapPoint.visible := false;
    pnCreateShipment.Visible := true;
end;

```

```

procedure TfrMainForm.N13Click(Sender: TObject);
begin
  hideAllPanels;
  spMapPoint.Visible := false;
  pnAddItem.Visible := true;
  pnAddItem.Left := (pnMapWrap.Width - pnAddItem.Width) shr 1;
  pnAddItem.Top := (pnMapWrap.Height - pnAddItem.Height) shr 1;
end;

procedure TfrMainForm.N14Click(Sender: TObject);
begin
  frSelectShipments := TfrSelectShipments.Create(Application);
  frSelectShipments.LoadData(@shipments);
  frSelectShipments.ShowModal;
  pbMap.Invalidate;
end;

procedure TfrMainForm.N16Click(Sender: TObject);
begin
  frShipmentTableForm := TfrShipmentsTable.Create(Application);
  frShipmentTableForm.LoadData(shipments);
  frShipmentTableForm.ShowModal;
end;

procedure TfrMainForm.N3Click(Sender: TObject);
var
  curShipment, prev: PShipment;
  allDone: boolean;
begin
  allDone := true;
  if getconfirmation('Подтверждение действия', 'Вы подтверждаете действие?') then
  begin
    curShipment := shipments;
    while curShipment <> nil do
    begin
      allDone := doShipment(curShipment) and allDone;
      prev := curShipment;
      curShipment := curShipment^.next;
      Dispose(prev);
    end;
    pbMap.Invalidate;
    if not allDone then
      ShowMessage('Ошибка', 'Произошла ошибка!')
    else
      showMessage('Успешно', 'Все отгрузки выполнены');
    shipments := nil;
  end;
end;

procedure TfrMainForm.N6Click(Sender: TObject);
begin
  frBalanceForm := TfrBalance.Create(Application);
  frBalanceForm.SetData(shops, warehouses);
  frBalanceForm.ShowModal;
end;

end.

```

Содержание модуля Messages

```
unit Messages;
```

```

interface
uses Vcl.Forms, Vcl.Dialogs, Vcl.StdCtrls, System.UITypes;

function getConfirmation(const capt, text: string): boolean;
procedure showMessage(const capt, text: string);

implementation

function getConfirmation(const capt, text: string): boolean;
var
  Dlg: TForm;
  i: integer;
begin
  Dlg := CreateMessageDialog(text,
                             mtConfirmation, [mbYes, mbNo]);
  try
    Dlg.Caption := capt;
    for i := 0 to Dlg.ComponentCount - 1 do
    begin
      if Dlg.Components[i] is TButton then
      begin
        with TButton(Dlg.Components[i]) do
        begin
          if ModalResult = mrYes then
            Caption := 'Да'
          else if ModalResult = mrNo then
            Caption := 'Нет';
          end;
        end;
      end;
    end;
    Result := Dlg.ShowModal = mrYes;
  finally
    Dlg.Free;
  end;
end;

procedure showMessage(const capt: string; const text: string);
var
  Dlg: TForm;
  i: integer;
begin
  Dlg := CreateMessageDialog(text,
                             mtConfirmation, [mbYes]);
  Dlg.Caption := capt;
  for i := 0 to Dlg.ComponentCount - 1 do
  begin
    if Dlg.Components[i] is TButton then
    begin
      with TButton(Dlg.Components[i]) do
      begin
        if ModalResult = mrYes then
          Caption := 'Ок'
        end;
      end;
    end;
  end;
  if (Dlg.ShowModal = mrYes) or true then
    Dlg.Free;
  end;
end.

```

Содержание модуля SelectShipmentsUnit

```
unit SelectShipmentsUnit;

interface

uses
  Winapi.Windows, Winapi.Messages, System.SysUtils, System.Variants, System.Classes, Vcl.Graphics,
  Vcl.Controls, Vcl.Forms, Vcl.Dialogs, Vcl.ExtCtrls, Vcl.Grids, Vcl.StdCtrls,
  Shipments, Hash, CartesianTree, Types, Messages;

type
  TfrSelectShipments = class(TForm)
    sgSelectShipmentsTable: TStringGrid;
    pnSelectShipments: TPanel;
    btnSelectConfirm: TButton;
    btnSelectAll: TButton;
    btnSelectReset: TButton;
    procedure FormCreate(Sender: TObject);
    procedure FormClose(Sender: TObject; var Action: TCloseAction);
    procedure LoadData(shipmentsPtr: PPShipment);

    procedure sgSelectShipmentsTableDrawCell(Sender: TObject; ACol, ARow: Integer;
      Rect: TRect; State: TGridDrawState);
    procedure sgSelectShipmentsTableMouseDown(Sender: TObject;
      Button: TMouseButton; Shift: TShiftState; X, Y: Integer);
    procedure btnSelectAllClick(Sender: TObject);
    procedure btnSelectResetClick(Sender: TObject);
    procedure btnSelectConfirmClick(Sender: TObject);
    procedure FormResize(Sender: TObject);
    procedure sgSelectShipmentsTableTopLeftChanged(Sender: TObject);
  private
    { Private declarations }
    procedure ToggleCheckbox(const ARow: Integer);
  var
    siz: integer;
    FShipmentsPtr: PPShipment;
  public
    { Public declarations }
  end;

var
  frSelectShipments: TfrSelectShipments;

implementation

{$R *.dfm}

const
  CHECKBOX_COL = 9;
  CHECKBOX_SIZE = 28;

var
  startHeight, startWidth: integer;

procedure TfrSelectShipments.btnSelectAllClick(Sender: TObject);
var
  i: integer;
begin
  for i := 1 to siz - 1 do
  begin
    sgSelectShipmentsTable.Cells[CHECKBOX_COL, i] := '1'
```



```

end;
frSelectShipments.Invalidate;
end;

procedure TfrSelectShipments.btnSelectConfirmClick(Sender: TObject);
var
  i: integer;
  curShipment, prev, temp: PShipment;
  allCorrect: boolean;
begin
  curShipment := FShipmentsPtr^;
  prev := nil;
  allCorrect := true;
  for i := 1 to siz-1 do
    begin
      if sgSelectShipmentsTable.Cells[CHECKBOX_COL, i] = '1' then
        begin
          allCorrect := doShipment(curShipment) and allCorrect;
          if prev = nil then
            FShipmentsPtr^ := curShipment^.Next
          else
            begin
              prev^.Next := curShipment^.Next;
            end;
          temp := curShipment;
          curShipment := curShipment^.Next;
          Dispose(temp);
        end
      else
        begin
          prev := curShipment;
          curShipment := curShipment^.next;
        end;
      sgSelectShipmentsTable.Cells[CHECKBOX_COL, i] := '0';
    end;
  if allCorrect then
    begin
      showMessage('Успешно', 'Все отгрузки были выполнены успешно!');
    end
  else
    begin
      showMessage('Ошибка', 'Произошла ошибка!');
    end;
  loadData(FShipmentsPtr);
  frSelectShipments.Invalidate;
end;

procedure TfrSelectShipments.btnSelectResetClick(Sender: TObject);
var
  i: integer;
begin
  for i := 1 to siz - 1 do
    begin
      sgSelectShipmentsTable.Cells[CHECKBOX_COL, i] := '0'
    end;
  frSelectShipments.Invalidate;
end;

procedure TfrSelectShipments.FormClose(Sender: TObject;
  var Action: TCloseAction);
begin
  Action := caFree;
end;

```

```

end;

procedure TfrSelectShipments.LoadData(shipmentsPtr: PPShipment);
var
  i: integer;
  curShipment: PShipment;
begin
  FShipmentsPtr := shipmentsPtr;
  siz := 1; //header
  curShipment := FShipmentsPtr^;
  while curShipment <> nil do
    begin
      Inc(siz);
      curShipment := curShipment^.next;
    end;
  sgSelectShipmentsTable.RowCount := siz;
  curShipment := FShipmentsPtr^;
  i := 1;
  while curShipment <> nil do
    begin
      sgSelectShipmentsTable.Cells[0, i] := curShipment^.ShipmentName;
      sgSelectShipmentsTable.Cells[1, i] := intToStr(curShipment^.ID);
      sgSelectShipmentsTable.Cells[2, i] := string(curShipment^.SourceID^.name);
      sgSelectShipmentsTable.Cells[3, i] := 'ул. ' + string(curShipment^.SourceID^.street)
        + ', д. ' + intToStr(curShipment^.SourceID^.house)
        + ', корп. ';
      if curShipment^.SourceID^.building <> -1 then
        sgSelectShipmentsTable.Cells[3, i] := sgSelectShipmentsTable.Cells[3, i] +
          intToStr(curShipment^.SourceID^.building)
      else
        sgSelectShipmentsTable.Cells[3, i] := sgSelectShipmentsTable.Cells[3, i] + '-';

      sgSelectShipmentsTable.Cells[4, i] := string(curShipment^.DestinationID^.name);

      sgSelectShipmentsTable.Cells[5, i] := 'ул. ' + string(curShipment^.DestinationID^.street)
        + ', д. ' + intToStr(curShipment^.DestinationID^.house)
        + ', корп. ';
      if curShipment^.DestinationID^.building <> -1 then
        sgSelectShipmentsTable.Cells[5, i] := sgSelectShipmentsTable.Cells[5, i] +
          intToStr(curShipment^.DestinationID^.building)
      else
        sgSelectShipmentsTable.Cells[5, i] := sgSelectShipmentsTable.Cells[5, i] + '-';

      sgSelectShipmentsTable.Cells[6, i] := curShipment^.ProductName;
      sgSelectShipmentsTable.Cells[7, i] := intToStr(getHash(curShipment^.ProductName));
      sgSelectShipmentsTable.Cells[8, i] := intToStr(curShipment^.Count);
      Inc(i);
      curShipment := curShipment^.next;
    end;
  end;

procedure TfrSelectShipments.sgSelectShipmentsTableDrawCell(Sender: TObject;
  ACol, ARow: Integer; Rect: TRect; State: TGridDrawState);
var
  CheckRect: TRect;
  S: string;
  OldFontSize: integer;
  OldCharSet: integer;
  OldFontName: string;
begin
  if (ACol = CHECKBOX_COL) and (ARow > 0) then
    begin

```

```

with sgSelectShipmentsTable.Canvas do
begin
    // Очищаем фон
    Brush.Color := clWhite;
    FillRect(Rect);

    // Рассчитываем положение чекбокса
    CheckRect := Rect;
    CheckRect.Left := CheckRect.Left + (CheckRect.Width - CHECKBOX_SIZE) div 2;
    CheckRect.Top := CheckRect.Top + (CheckRect.Height - CHECKBOX_SIZE) div 2;
    CheckRect.Right := CheckRect.Left + CHECKBOX_SIZE;
    CheckRect.Bottom := CheckRect.Top + CHECKBOX_SIZE;

    // Рисуем рамку чекбокса
    Pen.Color := clBlack;
    Brush.Style := bsClear;
    Rectangle(CheckRect);

    // Рисуем галочку если отмечено
    if sgSelectShipmentsTable.Cells[ACol, ARow] = '1' then
    begin
        OldFontSize := Font.Size;
        OldCharSet := DEFAULT_CHARSET;
        OldFontName := Font.Name;

        Font.Size := 16;
        Font.Name := 'Arial';
        Font.CharSet := DEFAULT_CHARSET;
        TextOut(CheckRect.Left + 1, CheckRect.Top - CHECKBOX_SIZE shr 1 + 3, '+');

        Font.Size := OldFontSize;
        Font.CharSet := OldCharSet;
        Font.Name := OldFontName;
    end;
end;

if (ACol <> CHECKBOX_COL) or (ARow = 0) then
begin
    S := sgSelectShipmentsTable.Cells[ACol, ARow];
    sgSelectShipmentsTable.Canvas.FillRect(Rect);

    DrawText(
        sgSelectShipmentsTable.Canvas.Handle,
        PChar(S), Length(S),
        Rect,
        DT_WORDBREAK or DT_NOPREFIX or DT_LEFT
    );
end;

end;

procedure TfrSelectShipments.sgSelectShipmentsTableMouseDown(Sender: TObject;
    Button: TMouseButton; Shift: TShiftState; X, Y: Integer);
var
    ACol, ARow: Integer;
    CheckRect: TRect;
begin
    sgSelectShipmentsTable.MouseToCell(X, Y, ACol, ARow);

    if (ACol = CHECKBOX_COL) and (ARow > 0) then
    begin

```

```

// Получаем область ячейки
CheckRect := sgSelectShipmentsTable.CellRect(ACol, ARow);

// Рассчитываем область чекбокса
CheckRect.Left := CheckRect.Left + (CheckRect.Width - CHECKBOX_SIZE) div 2;
CheckRect.Top := CheckRect.Top + (CheckRect.Height - CHECKBOX_SIZE) div 2;
CheckRect.Right := CheckRect.Left + CHECKBOX_SIZE;
CheckRect.Bottom := CheckRect.Top + CHECKBOX_SIZE;

// Проверяем клик внутри чекбокса
if (X >= CheckRect.Left) and (X <= CheckRect.Right) and
   (Y >= CheckRect.Top) and (Y <= CheckRect.Bottom) then
begin
    ToggleCheckbox(ARow);
end;
end;
end;

procedure TfrSelectShipments.sgSelectShipmentsTableTopLeftChanged(
    Sender: TObject);
begin
    sgSelectShipmentsTable.LeftCol := 1;
end;

procedure TfrSelectShipments.ToggleCheckbox(const ARow: Integer);
begin
    if sgSelectShipmentsTable.Cells[CHECKBOX_COL, ARow] = '1' then
        sgSelectShipmentsTable.Cells[CHECKBOX_COL, ARow] := '0'
    else
        sgSelectShipmentsTable.Cells[CHECKBOX_COL, ARow] := '1';

    frSelectShipments.Invalidate;
end;

procedure TfrSelectShipments.FormCreate(Sender: TObject);
begin
    startWidth := frSelectShipments.ClientWidth;
    startHeight := frSelectShipments.ClientHeight;
    FormStyle := fsNormal;
    Position := poMainFormCenter;

    //Название (15), ID (5), отправитель (15), адрес отправителя (15),
    //получатель (15), адрес получателя (15), товар (10), артикул (5), количество(5)

    sgSelectShipmentsTable.ColCount := 10;

    sgSelectShipmentsTable.ColWidths[0] := trunc(0.13 * frSelectShipments.clientWidth);
    sgSelectShipmentsTable.ColWidths[1] := trunc(0.05 * frSelectShipments.clientWidth);
    sgSelectShipmentsTable.ColWidths[2] := trunc(0.14 * frSelectShipments.clientWidth);
    sgSelectShipmentsTable.ColWidths[3] := trunc(0.13 * frSelectShipments.clientWidth);
    sgSelectShipmentsTable.ColWidths[4] := trunc(0.14 * frSelectShipments.clientWidth);
    sgSelectShipmentsTable.ColWidths[5] := trunc(0.13 * frSelectShipments.clientWidth);
    sgSelectShipmentsTable.ColWidths[6] := trunc(0.10 * frSelectShipments.clientWidth);
    sgSelectShipmentsTable.ColWidths[7] := trunc(0.05 * frSelectShipments.clientWidth);
    sgSelectShipmentsTable.ColWidths[8] := trunc(0.06 * frSelectShipments.clientWidth);
    sgSelectShipmentsTable.ColWidths[9] := frSelectShipments.clientWidth
        -sgSelectShipmentsTable.ColWidths[0]
        -sgSelectShipmentsTable.ColWidths[1]
        -sgSelectShipmentsTable.ColWidths[2]
        -sgSelectShipmentsTable.ColWidths[3]
        -sgSelectShipmentsTable.ColWidths[4]
        -sgSelectShipmentsTable.ColWidths[5]

```

```

-sgSelectShipmentsTable.ColWidths[6]
-sgSelectShipmentsTable.ColWidths[7]
-sgSelectShipmentsTable.ColWidths[8]
-sgSelectShipmentsTable.ColCount * sgSelectShipmentsTable.GridLineWidth;

sgSelectShipmentsTable.DefaultDrawing := False;

sgSelectShipmentsTable.Cells[0, 0] := 'Название';
sgSelectShipmentsTable.Cells[1, 0] := 'ID';
sgSelectShipmentsTable.Cells[2, 0] := 'Отправитель';
sgSelectShipmentsTable.Cells[3, 0] := 'Адрес отправителя';
sgSelectShipmentsTable.Cells[4, 0] := 'Получатель';
sgSelectShipmentsTable.Cells[5, 0] := 'Адрес получателя';
sgSelectShipmentsTable.Cells[6, 0] := 'Товар';
sgSelectShipmentsTable.Cells[7, 0] := 'Артикул';
sgSelectShipmentsTable.Cells[8, 0] := 'Количество';
sgSelectShipmentsTable.Cells[CHECKBOX_COL, 0] := 'Выбрать';

//sgSelectShipmentsTable.Cells[9, 0] := 'Выбрать';

sgSelectShipmentsTable.RowCount := 0;

sgSelectShipmentsTable.DoubleBuffered := True;

sgSelectShipmentsTable.OnDrawCell := sgSelectShipmentsTableDrawCell;
sgSelectShipmentsTable.OnMouseDown := sgSelectShipmentsTableMouseDown;

end;

procedure TfrSelectShipments.FormResize(Sender: TObject);
begin
    frSelectShipments.ClientWidth := startWidth;
    frSelectShipments.ClientHeight := startHeight;
end;

end.

```

Содержание модуля shipments

```

unit shipments;

interface
uses CartesianTreeItem, Hash, Types, Vars;

function doShipment(var shipment: PShipment): boolean;
procedure ClearShipments(var shipment: PShipment);

implementation

procedure ClearShipments(var shipment: PShipment);
var
    prev: PShipment;
begin
    while shipment <> nil do
        begin
            prev := shipment;
            shipment := shipment^.next;
            Dispose(prev);
        end;
    end;
end;

function doShipment(var shipment: PShipment): boolean;

```

```

var
  sendItemNode, destItemNode: PTreapItemNode;
  newNode: PItem;
  found: boolean;
  i: integer;
  newItemNode: PTreapItemNode;
begin
  Result := true;
  try
    begin
      found := false;
      i := 0;
      while (not found) and (i < shipment^.SourceID^.OutgoingArrows.Count) do
        begin
          if shipment^.SourceID^.OutgoingArrows[i]^shipment = shipment then
            begin
              shipment^.SourceID^.OutgoingArrows.Remove(shipment^.SourceID^.OutgoingArrows[i]);
              found := true;
            end;
          Inc(i);
        end;

      found := false;
      i := 0;
      while (not found) and (i < shipment^.DestinationID^.IncomingArrows.Count) do
        begin
          if shipment^.DestinationID^.IncomingArrows[i]^shipment = shipment then
            begin
              shipment^.DestinationID^.IncomingArrows.Remove(shipment^.DestinationID^.IncomingArrows[i]);
              found := true;
            end;
          Inc(i);
        end;

      found := false;
      i := 0;
      while (not found) and (i < Arrows.Count) do
        begin
          if Arrows[i]^shipment = shipment then
            begin
              Arrows.Remove(Arrows[i]);
              found := true;
            end;
          Inc(i);
        end;

      destItemNode := FindTreapItem(shipment^.DestinationID^.Items, getHash(shipment^.ProductName));
      sendItemNode := FindTreapItem(shipment^.SourceID^.Items, getHash(shipment^.ProductName));

      if destItemNode = nil then
        begin
          newNode := new(PItem);
          newNode^.name := shortString(shipment^.ProductName);
          newNode^.category := sendItemNode^.Data^.category;
          newNode^.Volume := sendItemNode^.Data^.volume;
          newNode^.Count := 0;
          newNode^.Key := getHash(shipment^.ProductName);
          newItemNode := CreateNewItemNode(newNode);
          InsertTreapItem(shipment^.DestinationID^.Items, newItemNode);
          destItemNode := FindTreapItem(shipment^.DestinationID^.Items, getHash(shipment^.ProductName));
        end;
    end;
  end;

```

```

if sendItemNode^.Data^.Volume = destItemNode^.Data^.Volume then
begin
  Dec(sendItemNode^.Data^.needToSend, shipment^.Count);
  Dec(shipment^.SourceID^.usedCapacity, sendItemNode^.Data^.Volume * shipment^.Count);
  Dec(shipment^.DestinationID^.shipmentCapacity, sendItemNode^.Data^.Volume * shipment^.Count);
  Inc(shipment^.DestinationID^.usedCapacity, sendItemNode^.Data^.Volume * shipment^.Count);

  Dec(sendItemNode^.Data^.Count, shipment^.Count);

  if sendItemNode^.Data^.Count = 0 then
  begin
    EraseTreapItem(shipment^.SourceID^.Items, getHash(shipment^.ProductName));
  end;

  Inc(destItemNode^.Data^.Count, shipment^.Count);
end;
Saved := false;
end
except
begin
  Result := false;
end;
end;

end;

end.

```

Содержание модуля ShipmentsTableUnit

```

unit ShipmentsTableUnit;

interface

uses
  Winapi.Windows, Winapi.Messages, System.SysUtils, System.Variants, System.Classes, Vcl.Graphics,
  Vcl.Controls, Vcl.Forms, Vcl.Dialogs, shipments, Vcl.Grids, hash, Types;

type
  TfrShipmentsTable = class(TForm)
    sgShipmentsTable: TStringGrid;
    procedure FormClose(Sender: TObject; var Action: TCloseAction);
    procedure LoadData(var shipment: PShipment);
    procedure FormCreate(Sender: TObject);
    procedure FormResize(Sender: TObject);
    procedure sgShipmentsTableDrawCell(Sender: TObject; ACol, ARow: LongInt;
      Rect: TRect; State: TGridDrawState);
    procedure sgShipmentsTableTopLeftChanged(Sender: TObject);
  private
    { Private declarations }
  public
    { Public declarations }
  end;

var
  frShipmentsTable: TfrShipmentsTable;

implementation

var
  startHeight, startWidth: integer;

{$R *.dfm}

```

```

procedure TfrShipmentsTable.FormClose(Sender: TObject;
  var Action: TCloseAction);
begin
  Action := caFree;
end;

procedure TfrShipmentsTable.LoadData(var shipment: PShipment);
var
  siz, i: integer;
  curShipment: PShipment;
begin
  siz := 0;
  curShipment := shipment;
  while curShipment <> nil do
    begin
      Inc(siz);
      curShipment := curShipment^.next;
    end;
  sgShipmentsTable.RowCount := siz + 1;
  curShipment := shipment;
  i := 1;
  while curShipment <> nil do
    begin
      sgShipmentsTable.Cells[0, i] := curShipment^.ShipmentName;
      sgShipmentsTable.Cells[1, i] := intToStr(curShipment^.ID);
      sgShipmentsTable.Cells[2, i] := string(curShipment^.SourceID^.name);
      sgShipmentsTable.Cells[3, i] := 'ул. ' + string(curShipment^.SourceID^.street)
        + ', д. ' + intToStr(curShipment^.SourceID^.house)
        + ', корп. ';
      if curShipment^.SourceID^.building <> -1 then
        sgShipmentsTable.Cells[3, i] := sgShipmentsTable.Cells[3, i] + intToStr(curShipment^.SourceID^.building)
      else
        sgShipmentsTable.Cells[3, i] := sgShipmentsTable.Cells[3, i] + '-';

      sgShipmentsTable.Cells[4, i] := string(curShipment^.DestinationID^.name);

      sgShipmentsTable.Cells[5, i] := 'ул. ' + string(curShipment^.DestinationID^.street)
        + ', д. ' + intToStr(curShipment^.DestinationID^.house)
        + ', корп. ';
      if curShipment^.DestinationID^.building <> -1 then
        sgShipmentsTable.Cells[5, i] := sgShipmentsTable.Cells[5, i] + intToStr(curShipment^.DestinationID^.building)
      else
        sgShipmentsTable.Cells[5, i] := sgShipmentsTable.Cells[5, i] + '-';

      sgShipmentsTable.Cells[6, i] := curShipment^.ProductName;
      sgShipmentsTable.Cells[7, i] := intToStr(getHash(curShipment^.ProductName));
      sgShipmentsTable.Cells[8, i] := intToStr(curShipment^.Count);
      Inc(i);
      curShipment := curShipment^.next;
    end;
  end;

procedure TfrShipmentsTable.sgShipmentsTableDrawCell(Sender: TObject; ACol,
  ARow: LongInt; Rect: TRect; State: TGridDrawState);
var
  S: string;
begin
  S := sgShipmentsTable.Cells[ACol, ARow];
  sgShipmentsTable.Canvas.FillRect(Rect);

  DrawText(
    sgShipmentsTable.Canvas.Handle,

```



```

    PChar(S), Length(S),
    Rect,
    DT_WORDBREAK or DT_NOPREFIX or DT_LEFT
);
end;

procedure TfrShipmentsTable.sgShipmentsTableTopLeftChanged(Sender: TObject);
begin
    sgShipmentsTable.LeftCol := 1;
end;

procedure TfrShipmentsTable.FormCreate(Sender: TObject);
begin
    startWidth := frShipmentsTable.ClientWidth;
    startHeight := frShipmentsTable.ClientHeight;
    FormStyle := fsNormal;
    Position := poMainFormCenter;

    //Название (15), ID (5), отправитель (15), адрес отправителя (15),
    //получатель (15), адрес получателя (15), товар (10), артикул (5), количество(5)

    sgShipmentsTable.ColCount := 9;

    sgShipmentsTable.ColWidths[0] := trunc(0.14 * frShipmentsTable.clientWidth);
    sgShipmentsTable.ColWidths[1] := trunc(0.05 * frShipmentsTable.clientWidth);
    sgShipmentsTable.ColWidths[2] := trunc(0.14 * frShipmentsTable.clientWidth);
    sgShipmentsTable.ColWidths[3] := trunc(0.14 * frShipmentsTable.clientWidth);
    sgShipmentsTable.ColWidths[4] := trunc(0.14 * frShipmentsTable.clientWidth);
    sgShipmentsTable.ColWidths[5] := trunc(0.14 * frShipmentsTable.clientWidth);
    sgShipmentsTable.ColWidths[6] := trunc(0.10 * frShipmentsTable.clientWidth);
    sgShipmentsTable.ColWidths[7] := trunc(0.05 * frShipmentsTable.clientWidth);
    sgShipmentsTable.ColWidths[8] := frShipmentsTable.clientWidth
        -sgShipmentsTable.ColWidths[0]
        -sgShipmentsTable.ColWidths[1]
        -sgShipmentsTable.ColWidths[2]
        -sgShipmentsTable.ColWidths[3]
        -sgShipmentsTable.ColWidths[4]
        -sgShipmentsTable.ColWidths[5]
        -sgShipmentsTable.ColWidths[6]
        -sgShipmentsTable.ColWidths[7]
        -sgShipmentsTable.ColCount * sgShipmentsTable.GridLineWidth;

    sgShipmentsTable.Cells[0, 0] := 'Название';
    sgShipmentsTable.Cells[1, 0] := 'ID';
    sgShipmentsTable.Cells[2, 0] := 'Отправитель';
    sgShipmentsTable.Cells[3, 0] := 'Адрес отправителя';
    sgShipmentsTable.Cells[4, 0] := 'Получатель';
    sgShipmentsTable.Cells[5, 0] := 'Адрес получателя';
    sgShipmentsTable.Cells[6, 0] := 'Товар';
    sgShipmentsTable.Cells[7, 0] := 'Артикул';
    sgShipmentsTable.Cells[8, 0] := 'Количество';

    sgShipmentsTable.RowCount := 0;

    sgShipmentsTable.DefaultDrawing := False;
end;

procedure TfrShipmentsTable.FormResize(Sender: TObject);
begin
    frShipmentsTable.ClientWidth := startWidth;
    frShipmentsTable.ClientHeight := startHeight;

```

end;

end.

Содержание модуля TableUnit

unit TableUnit;

interface

uses

Winapi.Windows, Winapi.Messages, System.SysUtils, System.Variants, System.Classes, System.UITypes,
Vcl.Graphics, Vcl.Controls, Vcl.Forms, Vcl.Dialogs, Vcl.Grids, Vcl.ExtCtrls,
Types, Vars;

type

TfrTableForm = class(TForm)
 pnItemsTableName: TPanel;
 sgItemsTable: TStringGrid;
 procedure FormClose(Sender: TObject; var Action: TCloseAction);
 procedure FormCreate(Sender: TObject);
 function GetTreeSize(const root: PTreapItemNode): integer;
 procedure LoadData;
 procedure SetDataToTable(const root: PTreapItemNode; var i: integer);
 procedure FormResize(Sender: TObject);
 procedure sgItemsTableDrawCell(Sender: TObject; ACol, ARow: LongInt;
 Rect: TRect; State: TGridDrawState);
 procedure sgItemsTableTopLeftChanged(Sender: TObject);
private
 { Private declarations }
public
 Location: PLocation;
 { Public declarations }
end;

var

frTableForm: TfrTableForm;

implementation

var

startWidth, startHeight: integer;

{ \$R *.dfm }

function TfrTableForm.GetTreeSize(const root: PTreapItemNode): integer;

begin

 if root = nil then

 Result := 0

 else

 Result := 1 + GetTreeSize(root^.Left) + GetTreeSize(root^.Right);

end;

procedure TfrTableForm.SetDataToTable(const root: PTreapItemNode; var i: integer);

begin

 if root <> nil then

 begin

 sgItemsTable.Cells[0, i] := string(root^.Data^.name);

 sgItemsTable.Cells[1, i] := string(root^.Data^.category);

 sgItemsTable.Cells[2, i] := intToStr(root^.Data^.Volume);

 sgItemsTable.Cells[3, i] := intToStr(root^.Data^.Count);

 sgItemsTable.Cells[4, i] := intToStr(root^.Data^.Key);

 Inc(i);

```

        SetDataToTable(root^.Left, i);
        SetDataToTable(root^.Right, i);
    end;
end;

procedure TfrTableForm.sgItemsTableDrawCell(Sender: TObject; ACol,
    ARow: LongInt; Rect: TRect; State: TGridDrawState);
var
    S: string;
begin
    S := sgItemsTable.Cells[ACol, ARow];
    sgItemsTable.Canvas.FillRect(Rect);

    DrawText(
        sgItemsTable.Canvas.Handle,
        PChar(S), Length(S),
        Rect,
        DT_WORDBREAK or DT_NOPREFIX or DT_LEFT
    );
end;
procedure TfrTableForm.sgItemsTableTopLeftChanged(Sender: TObject);
begin
    sgItemsTable.LeftCol := 1;
end;

procedure TfrTableForm.LoadData;
var
    siz, i: integer;
begin
    siz := GetTreeSize(Location^.Items);
    sgItemsTable.RowCount := siz + 1;
    i := 1;
    pnItemsTableName.Caption := 'Товары ';
    if (Location^.Key and mask) <> 0 then
        pnItemsTableName.Caption := pnItemsTableName.Caption + 'в магазине'
    else
        pnItemsTableName.Caption := pnItemsTableName.Caption + 'на складе';
    pnItemsTableName.Caption := pnItemsTableName.Caption + ' ' + string(Location^.name);
    pnItemsTableName.Font.Style := pnItemsTableName.Font.Style + [fsBold];
    pnItemsTableName.Font.Size := 16;
    SetDataToTable(Location^.Items, i);
end;

procedure TfrTableForm.FormClose(Sender: TObject; var Action: TCloseAction);
begin
    Action := caFree;
end;

procedure TfrTableForm.FormCreate(Sender: TObject);
begin
    startWidth := frTableForm.ClientWidth;
    startHeight := frTableForm.ClientHeight;

    FormStyle := fsNormal;
    Position := poMainFormCenter;

    sgItemsTable.ColCount := 5;

    sgItemsTable.ColWidths[0] := trunc(0.328 * frTableForm.clientWidth); // Название (32.8%)

```

```

sgItemsTable.ColWidths[1] := trunc(0.266 * frTableForm.clientWidth); // Категория (26.6%)
sgItemsTable.ColWidths[2] := trunc(0.141 * frTableForm.clientWidth); // Объем места (14.1%)
sgItemsTable.ColWidths[3] := trunc(0.094 * frTableForm.clientWidth); // Количество (9.4%)
sgItemsTable.ColWidths[4] := frTableForm.clientWidth
    -sgItemsTable.ColWidths[0]
    -sgItemsTable.ColWidths[1]
    -sgItemsTable.ColWidths[2]
    -sgItemsTable.ColWidths[3]
    -sgItemsTable.ColCount * sgItemsTable.GridLineWidth; // Артикул (17.1%)

sgItemsTable.Cells[0, 0] := 'Название';
sgItemsTable.Cells[1, 0] := 'Категория';
sgItemsTable.Cells[2, 0] := 'Объем единицы товара (у.е.)';
sgItemsTable.Cells[3, 0] := 'Количество';
sgItemsTable.Cells[4, 0] := 'Артикул';

sgItemsTable.RowCount := 0;

sgItemsTable.DefaultDrawing := false;
end;

procedure TfrTableForm.FormResize(Sender: TObject);
begin
    frTableForm.ClientWidth := startWidth;
    frTableForm.ClientHeight := startHeight;
end;

end.

```

Содержание модуля Validation

```

unit Validation;

interface
uses SysUtils, Vcl.Graphics, Vcl.StdCtrls, messages;

function validateLength(Sender: TObject): boolean;
function validateLetters(Sender: TObject): boolean;
function validateAll(Sender: TObject): boolean;
function validateFromTo(const firstField, secondField: TObject): boolean;
procedure validateIntegerInput(Sender: TObject);
procedure validateLengthLess70(Sender: TObject);

implementation

procedure validateLengthLess70(Sender: TObject);
var
    newString: string;
begin
    if Length((Sender as TEdit).Text) > 70 then
    begin
        newString := (Sender as TEdit).Text;
        Delete(newString, Length(newString), 1);
        (Sender as TEdit).Text := newString;
        showMessage('Внимание!', 'Длина строки должна быть не более 70 символов!');
    end;
end;

function validateAll(Sender: TObject): boolean;
begin
    Result := validateLength(Sender);
    Result := validateLetters(Sender) and Result;
end;

```

```

procedure validateIntegerInput(Sender: TObject);
var
  newString: string;
  i: integer;
begin
  // test 0
  if (Length((Sender as TEdit).Text) > 0) and ((Sender as TEdit).Text <> '0') then
  begin
    i := Low((Sender as TEdit).Text);
    while (i <= High((Sender as TEdit).Text)) and ((Sender as TEdit).Text[i] = '0') do
      Inc(i);
    if i > High((Sender as TEdit).Text) then
      newString := '0'
    else
      begin
        newString := '';
        while (i <= High((Sender as TEdit).Text)) and (Length(newString) < 10) do
          begin
            newString := newString + (Sender as TEdit).Text[i];
            Inc(i);
          end;

          if (Length(newString) < 10) or ((Length(newString) = 10)
            and (newString < intToStr(MaxInt))) then
            begin
              //pass
            end
          else
            Delete(newString, Length(newString), 1);
          end;

          if newString <> (Sender as TEdit).Text then
            begin
              showMessage('Внимание!', 'Число должно быть меньше 2147483647 и не содержать лидирующих нулей!');
              (Sender as TEdit).Text := newString;
            end;

          end;
        end;

function validateLetters(Sender: TObject): boolean;
var
  i: integer;
begin
  Result := true;
  for i := Low((Sender as TEdit).Text) to High((Sender as TEdit).Text) do
  begin
    if (((Sender as TEdit).Text[i] <> ' ')
      and
      ((
        (
          (lowerCase((Sender as TEdit).Text[i]) > 'z')
          or (lowerCase((Sender as TEdit).Text[i]) < 'a')
        )
        and
        (
          ((Sender as TEdit).Text[i] > 'я')
          or ((Sender as TEdit).Text[i] < 'а'))
        ))
      and
      ((Sender as TEdit).Text[i] <> 'ё'))

```

```

    and
    ((Sender as TEdit).Text[i] <> 'Ё')
    and
    (
    ((Sender as TEdit).Text[i] > 'Я')
    or ((Sender as TEdit).Text[i] < 'А')
    )
    and
    (
    ((Sender as TEdit).Text[i] > '9')
    or ((Sender as TEdit).Text[i] < '0')
    )
  ) then
begin
  (Sender as TEdit).color := clRed;
  Result := false;
end;
end;
end;

```

```

function validateLength(Sender: TObject): boolean;
begin
  Result := true;
  (Sender as TEdit).Text := trimLeft((Sender as TEdit).Text);
  (Sender as TEdit).SelStart := Length((Sender as TEdit).Text);
  if Length((Sender as TEdit).Text) = 0 then
  begin
    (Sender as TEdit).color := clRed;
    Result := false;
  end
end;

```

```

function validateFromTo(const firstField, secondField: TObject): boolean;
begin
  Result := true;
  if (((Length((firstField as TEdit).Text) > 0) and (Length((secondField as TEdit).Text) > 0))
    and (strToInt((firstField as TEdit).text) > strToInt((secondField as TEdit).text)))
  then
    Result := false;
end;

end.

```

Содержание модуля ArrowsUnit

```
unit ArrowsUnit;
```

```
interface
```

```
uses System.Generics.Collections, System.Types, Types, Vars;
```

```

procedure AddArrow(var Arrows: TList<PArrow>; var Shipment: PShipment);
procedure RemoveArrow(var arrowsList: TList<PArrow>; var Arrow: PArrow);
function IsPointNearLine(P, A, B: TPoint; Tolerance: Integer): Boolean;

```

```
implementation
```

```

procedure AddArrow(var Arrows: TList<PArrow>; var Shipment: PShipment);
var
  NewArrow: PArrow;
begin
  NewArrow := New(PArrow);
  NewArrow^.shipment := Shipment;

```

```

if NewArrow^.shipment^.SourceID^.shape.Visible
  and NewArrow^.shipment^.DestinationID^.shape.Visible then
  NewArrow^.Visible := true
else
  NewArrow^.Visible := false;

Arrows.Add(NewArrow);

if Shipment^.SourceID^.OutgoingArrows = nil then
  Shipment^.SourceID^.OutgoingArrows := TList<PArrow>.Create;
if Shipment^.DestinationID^.IncomingArrows = nil then
  Shipment^.DestinationID^.IncomingArrows := TList<PArrow>.Create;

Shipment^.SourceID^.OutgoingArrows.Add(NewArrow);
Shipment^.DestinationID^.IncomingArrows.Add(NewArrow);
end;

function IsPointNearLine(P, A, B: TPoint; Tolerance: Integer): Boolean;
var
  dx, dy, numerator, denominator: Double;
  maxx, maxy, minx, miny: integer;
begin
  Result := true;

  maxx := A.X;
  minx := A.X;
  maxy := A.Y;
  minY := A.Y;

  if B.X > maxx then
    maxx := B.X;

  if B.X < minx then
    minx := B.X;

  if B.Y > maxy then
    maxy := B.Y;

  if B.Y < miny then
    miny := B.Y;

  Result := Result and (P.X >= minx) and (P.X <= maxx) and (P.Y >= miny) and (P.Y <= maxy);

  dx := B.X - A.X;
  dy := B.Y - A.Y;
  numerator := Abs(dy * P.X - dx * P.Y + B.X * A.Y - B.Y * A.X);
  denominator := Sqrt(dx * dx + dy * dy);
  Result := Result and ((numerator / denominator) <= Tolerance);
end;

procedure RemoveArrow(var arrowsList: TList<PArrow>; var Arrow: PArrow);
begin
  Arrow^.Shipment^.SourceID^.OutgoingArrows.Remove(Arrow);
  Arrow^.Shipment^.DestinationID^.IncomingArrows.Remove(Arrow);
  Arrows.Remove(Arrow);
  Dispose(Arrow);
end;

end.

```

Содержание модуля Types

unit Types;

```

interface

uses Vcl.ExtCtrls, System.Generics.Collections;

type

TItem = record
    name, category: string[255];
    Volume, Count, Key, needToSend: Integer;
end;
PItem = ^TItem;

PTreapItemNode = ^TTreapItemNode;
TTreapItemNode = record
    Data: PItem;
    Left, Right: PTreapItemNode;
    Priority: Integer;
end;

PLocation = ^TLocation;
PPShipment = ^PShipment;
PShipment = ^TShipment;
PTreapNode = ^TTreapNode;

TShipment = record
    ShipmentName: string;
    ID: integer;
    SourceID: PLocation;    // Отправитель
    DestinationID: PLocation; // Получатель
    ProductName: string;
    Count: Integer;
    next: PShipment;
end;

PArrow = ^TArrow;
TArrow = record
    shipment: PShipment;
    Visible: boolean;
end;

TLocation = record
    name, street: string[255];
    house, building, capacity, usedCapacity, shipmentCapacity: integer;
    Key, X, Y: Integer;
    shape: TShape;
    Items: PTreapItemNode;
    OutgoingArrows: TList<PArrow>;
    IncomingArrows: TList<PArrow>;
end;

TTreapNode = record
    Data: PLocation;
    Left, Right: PTreapNode;
    Priority: Integer;
end;

TName = record
    name: string[255];
    ID: integer;
    Key: integer;
end;

```



```

PName = ^TName;

PTreapNameNode = ^TTreapNameNode;
TTreapNameNode = record
  Data: PName;
  Left, Right: PTreapNameNode;
  Priority: Integer;
end;

TFilter = record
  buildingType: integer;
  //objType = 0 (00b): none, 1(01b): shop, 2(10b): warehouse, 3(11b): both

  street: string[255];

  house, building,
  capacityFrom, capacityTo,
  usedCapacityFrom, usedCapacityTo: integer;
end;

implementation

end.

```

Содержание модуля Vars

```

unit Vars;

interface

Uses Vcl.Graphics, System.Generics.Collections, Types;

var
  //ArrowsUnit
  Arrows: TList<PArrow>;

  //GetKeys
  shopKey: integer = 1;
  warehouseKey: integer = 1;
  curShipmentID: integer = 1;

  //Hash
  p: integer = 47;
  m: integer = 40009;
  pows: array[0..255] of integer;

  //MainUnit
  Saved: boolean;

const
  //MainUnitColors
  shopColor = clHighlight;
  warehouseColor = clMaroon;

  //ObjectMask
  mask = 1 shl 30;
implementation

end.

```

Обозначение					Наименование					Дополнительные сведения				
					<u>Текстовые документы</u>									
БГУИР КР 6-05-0612-01 122 ПЗ					Пояснительная записка					138 с.				
					<u>Графические документы</u>									
ГУИР 451001 122 СП					Программное средство мониторинга и отображения сетевой структуры отгрузки товаров со складов в магазины Ведомость курсовой работы					Формат А1				
ГУИР 451001 122 СА					Программное средство мониторинга и отображения сетевой структуры отгрузки товаров со складов в магазины Ведомость курсовой работы					Формат А1				