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 Соболь Н.Г.

Руководитель:

асс. Фадеева Е.Е.

Минск 2025

1. **Анализ прототипов, литературных источников и формирование требований к проектируемому программному средству**
   1. **Обзор литературы**

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

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

Здесь подробно описываются различные алгоритмы, в том числе эффективные алгоритмы поиска кратчайших путей в графе. В программном средстве будет использоваться алгоритм Дейкстры, позволяющий найти кратчайший путь между двумя вершинами.

**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.

**1.1.4** [4] Гаджинский, А. М. Логистика. Учебник – Режим доступа: <https://www.booksite.ru/fulltext/logist/text.pdf>

В учебнике собрата теория по логистике, которую можно применить в программном средстве.

* 1. **Примеры решения аналогичных задач**

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. **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 маршрутов, статусов отгрузки и поставок, а также генерировать отчеты 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. Мониторинг отгрузок в режиме реального времени;
2. Визуализация сетевой структуры отгрузок;
3. Анализ и формирование отчетов по эффективности доставки;
4. Анализ и формирование отчетов по отгрузке товаров;
5. Анализ информации об уровне запасов в магазинах и на складах.

**1.3.3** Входные данные:

1. Информация о складах, участвующих в процессе отгрузки товаров. Название склада, адрес, вместимость, текущий уровень запасов, тип товара, хранящийся на складе.
2. Информация о магазинах, получающих товары со складов. Название магазина, адрес, требуемый уровень запасов, текущий уровень запасов, типы товаров, которые необходимы магазину.
3. Данные о товарах. Название, артикул или id, категория, срок годности, вес
4. Данные о маршрутах доставки;
5. Данные о транспортных средствах;
6. Статус отгрузок и информация о доставке;
7. Информация о заказах магазинов. Номер заказа, дата создания заказа, магазин, который сделал заказ, информация о заказанных товарах, статус заказа.

**1.3.4** Выходные данные:

1. Графическое отображение сетевой структуры отгрузок в виде карты;
2. Отчеты по выполнению планов доставки, с аналитическими данными. Номер отгрузки, дата и время, склад-отправитель, объём, статус, магазин-получатель.
3. Отчеты о текущем состоянии отгрузок. Среднее время отгрузок, загруженность складов, эффективность работы транспорта.
4. Планы отгрузок. Таблица с планами отгрузок на ближайшее время. Указание склада-отправителя, магазина-получателя, даты и объёма отгрузки.
5. Прогнозирование спроса на товары и будущих отгрузок на фоне уже произведенных.

**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, а также производить резервное копирование данных в конце каждой сессии и иметь возможность их восстановления.

* + 1. Условия эксплуатации:

Программное средство должно поддерживать работу на ОС Windows.

* + 1. 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. 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 (графы, диаграммы, карты), которые позволяют быстро анализировать информацию.

**2** **Анализ требований к ПС и** **разработка функциональных требований**

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

* + 1. Описание алгоритма Дейкстры

Алгоритм Дейкстры (англ. Dijkstra’s algorithm) находит кратчайшие пути от заданной вершины до всех остальных во взвешенном графе без ребер отрицательного веса. Пусть s – это начальная вершина. Необходимо задать массив d, в котором для каждой вершины v будет храниться значение кратчайшего расстояния из s в v. Изначально d[s]=0 (расстояние до стартовой вершины равно нулю), а для всех остальных вершин эта длина равна бесконечности (за бесконечность можно принять любое число, заведомо больше максимальной длины пути). Помимо этого, нужно создать массив u, в котором будет храниться информация о том, помечена вершина или нет. Изначально все вершины не помечены. Алгоритм будет производить n итераций, на каждой из них выбирать непомеченную вершина v с наименьшей величиной d[v]. Выбранная таким образом вершина v отмечается помеченной. Далее, на текущей итерации, из вершины v производятся релаксации: просматриваются все рёбра (v, to), исходящие из вершины v, и для каждой вершины to алгоритм пытается улучшить значение d[to]. Пусть длина текущего ребра равна len, тогда мы обновим d[to] как минимум из d[to] и d[v] + len. На этом итерация заканчивается. После всех итераций, все вершины графа станут помеченными, алгоритм завершится. Утверждается, что найденные значения d[v] и есть искомые длины кратчайших путей из s в v. В случае, если какие-либо вершины недостижимы из начальной, значение d[v] в них останется равным бесконечности.

* + 1. Доказательство корректности алгоритма Дейкстры

Основное утверждение, на котором основан алгоритм Дейкстры: утверждается, что после того, как какая-либо вершина v становится помеченной, текущее расстояние до неё d[v] уже является кратчайшим, и больше меняться не будет. Доказательство производится по индукции. Для начальной вершины справедливость очевидна: расстояния меньше нуля быть не может. Нужно предположить, что основное утверждение выполняется для всех предыдущих итераций, т.е. для всех вершин, которые уже были помечены ранее. Теперь нужно доказать, что это же утверждение не нарушается, при текущей итерации. Пусть v — вершина, выбранная на текущей итерации, т.е. вершина, которую алгоритм собирается пометить. Нужно доказать, что d[v] действительно равно длине кратчайшего пути до неё (обозначим эту длину через l[v]). Следующим шагом рассматривается некоторый путь P до вершины v. Очевидно, что этот путь можно разбить на два пути: P1, который состоит только из помеченных вершин (как минимум, в нем будет содержаться начальная вершина, а она уже была помечена) и вторая часть P2, которая обязательно начинается с непомеченной вершины, но содержать может как помеченные, так и не помеченные. Пусть p – это первая вершина пути P2, а q – последняя вершина пути P1.

Сначала приводится доказательство утверждения для вершины p, т.е. доказательство равенства d[p] = l[p]. Но это почти очевидно: на одной из предыдущих итераций мы выбирали вершину q и выполняли релаксацию из неё. Поскольку кратчайший путь до p равен кратчайшему пути до q плюс ребро (p,q), то при выполнении релаксации из q величина d[p] действительно установится в требуемое значение. Вследствие неотрицательности стоимостей рёбер длина кратчайшего пути l[p] = d[p] не превосходит длины l[v] кратчайшего пути до вершины v. Учитывая, что l[v] <=d[v] (так как алгоритм Дейкстры не мог найти более короткого пути, чем это вообще возможно), в итоге получаются соотношения:

С другой стороны, поскольку и p, и v — вершины непомеченные, то так как на текущей итерации была выбрана именно вершина v, а не вершина p, то получается другое неравенство:

Из этих неравенств следует равенство:

А значит, что:

* 1. **Описание функциональности ПС**

**2.2.1** Мониторинг отгрузок в реальном времени

Автоматическое обновление статусов отгрузок. Отображение данных в таблице с фильтрами и возможностью сортировки по складам, магазинам и статусам ("В пути", "Доставлено", "Задержано").

В интерфейсе вкладки “Текущие отгрузки” выводится таблица с колонками: номер отгрузки, склад-отправитель, магазин-получатель, товар, объем, статус, прогнозируемое время доставки.

**2.2.2** Построение оптимального маршрута

Получение выбранных точек от пользователя (склад и магазин). Применение алгоритма Дейкстры для поиска кратчайшего пути по времени или расстоянию на основе полученных точек. Визуализация маршрута на карте с отображением времени и расстояния.

**2.2.3** Формирование отчетов

Обработка данных за какой-либо промежуток времени, который задаёт пользователь (день / неделя / месяц) и генерация отчётов в виде отдельных файлов или вывода на экран. В отчёты система добавляет информацию о среднем времени доставки и проценте выполненных отгрузок.

**2.2.3** Управление данными (склады, магазины, транспорт)

Пользователь заполняет форму (название, адрес, вместимость, тип товара). Система проверяет уникальный идентификатор нового объекта (товара / магазина / склада) и правильность введенных данных. Происходит сохранение информации в файлы и автоматическое обновление карты сети или списка товаров.

При удалении или редактировании склада / магазина / товара система проверяет отсутствие записей, связанных с этим объектом.

**2.2.4** Прогнозирование спроса

Анализ данных за конкретный промежуток времени и построение прогноза спроса на основе предыдущих значений.

* 1. **Спецификация функциональных требований**
     1. Визуализация сетевой структуры отгрузок

Сетевая структура отгрузок (склады, магазины, маршруты) отображается в виде интерактивной карты. Карта автоматически подстраивается под разрешение экрана пользователя. Ось X (горизонтальная): расстояние в километрах. Ось Y (вертикальная): расстояние в километрах.

Элементы карты:

-склады обозначаются иконками с подписью: название, текущий запас (в шт. или кг).

-магазины обозначаются иконками с подписью названия магазина.

-маршруты отображаются линиями с указанием времени доставки (часы) и расстояния (км).

* + 1. Сортировка данных

Функция фильтрации принимает на вход данные, которые нужно отсортировать и параметр, по которому производится сортировка. На выход ничего не отдаётся, сортировка происходит во входном массиве.

* + 1. Построение кратчайшего маршрута

На вход функции подаётся граф, заданный списком смежности, начальная и конечная точка. Функция возвращает массив кратчайшего пути с номерами вершин в порядке обхода от начальной точки до конечной, время и длину пути.

* + 1. Добавление / редактирование / удаление объектов

Должно быть реализовано несколько процедур, каждая из которых будет отвечать за добавление, редактирование или удаление определенного объекта (магазина / склада / товара). На ввод подаётся информация о новом или уже существующем объекте (название товара / склада / магазина, адрес, тип товара, вместимость склада / магазина, категория товара, идентификатор объекта), в зависимости от того, какая операция будет выполняться. Все данные проходят валидацию. Подпрограмма ничего не возвращает, а только добавляет объект в список или удаляет объект из списка.

* + 1. Прогнозирование спроса

На вход подаются данные, по которым будет производиться прогнозирование, и промежуток времени, на основании которого необходимо сделать прогноз. Функция должна возвращать прогноз наиболее вероятных значений.

**3 Проектирование программного средства**

 **3.1 Разработка алгоритма ПС и алгоритмов отдельных модулей**

Рисунок 3.1.1 – Схема работы quickSort

Алгоритм быстрой сортировки реализован рекурсивно. Изначально в процедуру передается массив, параметр для сортировки, левая и правая граница. Если левая граница больше или равна правой, то алгоритм завершает работу. В противном случае алгоритм вызывает функцию выбора опорного элемента и помещает все элементы, которые меньше опорного, слева, остальные – справа. Далее процедура вызывается рекурсивно для левой и правой части массива.



Рисунок 3.1.2 – Схема работы partition

На ввод функции передается массив, границы и параметр, по которому производится сортировка. Далее случайным образом выбирается опорный элемент. Запускаем цикл, пока левая граница меньше правой. Внутри цикла происходит перестановка элементов. Все элементы, которые меньше опорного перемещаются в левую часть, остальные – в правую. В конце функция возвращает индекс левой границы.



Рисунок 3.1.3 – Схема работы dijkstra

На ввод функция получает граф, заданный списком смежности, количество вершин, начальную вершину, конечную вершину. Создается массив кратчайших расстояний, изначально заполненный числами, заведомо большими, чем максимально возможное расстояние в графе. Расстояние до стартовой вершины инициализируется нулём. Создаётся очередь с приоритетом (куча), в вершине которой всегда будет храниться вершина, расстояние до которой на данной итерации наименьшее. В очередь добавляется начальная вершина. Запускается цикл, пока в очереди есть хотя бы 1 элемент. На каждой итерации выбирается элемент с вершины очереди и удаляется из нее. Перебираются все рёбра, которые исходят из выбранной вершины. Для каждой вершины, в которую ведет текущее рассматриваемое ребро, совершается попытка найти ответ, который лучше, чем уже существующий. Если такой ответ существует, мы обновляем массив расстояний и добавляем вершину и расстояние до нее в приоритетную очередь. В конце происходит восстановление ответа и его возврат.



Рисунок 3.1.4 – Схема работы filter

Функция принимает на вход данные и параметр, по которому будет производиться фильтрация. Создаётся список ответа. Далее цикл проходит по каждому элементу в массиве входных данных и, если текущий элемент подходит по параметру, он добавляется в список ответа. В конце функция возвращает сформированный список ответа.



Рисунок 3.1.5 – Схема работы addShop

Процедура начинается с цикла, который проверяет корректность входных данных. Пользователь должен ввести название и расположение магазина. После завершения цикла процедура добавляет магазин в список магазинов.



Рисунок 3.1.6 – Схема работы addWarehouse

Процедура начинается с цикла, который проверяет корректность входных данных. Пользователь должен ввести название, расположение и вместимость склада. После завершения цикла процедура добавляет склад в список складов.

Рисунок 3.1.7 – Схема работы addItem

Процедура начинается с цикла, который проверяет корректность входных данных. Пользователь должен ввести название, место добавления, количество, категорию товара. После завершения цикла процедура добавляет товар в соответствующее место.



Рисунок 3.1.8 – Схема работы rmShop

Процедура начинается с цикла, который проверяет корректность входных данных. Пользователь должен ввести название магазина. После завершения цикла процедура удаляет магазин из списка магазинов.



Рисунок 3.1.9 – Схема работы rmWarehouse

Процедура начинается с цикла, который проверяет корректность входных данных. Пользователь должен ввести название склада. После завершения цикла процедура удаляет склад из списка складов.



Рисунок 3.1.10 – Схема работы linearRegression

Функция принимает на вход массивы координат x и y, коэффициент A и коэффициент B. Функция должна вычислить коэффициенты для уравнения y = Ax + B, по которой и будет производиться прогнозирование. После ввода начинается цикл по значениям массивов. Происходит вычисление четырёх значений: сумма всех X, сумма всех Y, сумма произведений X\*Y, сумма квадратов всех X. После цикла происходит вычисление знаменателя по формуле , где n – количество координат, sumX – сумма всех X, sumX2 – сумма квадратов всех X. Далее происходит вычисление коэффициентов A и B.



Рисунок 3.1.11 – Схема работы editShop

Процедура начинается с цикла, который проверяет корректность входных данных. Пользователь должен ввести название магазина. После завершения цикла происходит выбор поля, которое необходимо поменять. Пользователь вводит новое значение поля и подтверждает или отменяет свой выбор.



Рисунок 3.1.12 – Схема работы editWarehouse

Процедура начинается с цикла, который проверяет корректность входных данных. Пользователь должен ввести название склада. После завершения цикла происходит выбор поля, которое необходимо поменять. Пользователь вводит новое значение поля и подтверждает или отменяет свой выбор.



Рисунок 3.1.13 – Схема работы editItem

Процедура начинается с цикла, который проверяет корректность входных данных. Пользователь должен ввести название товара и место, в котором его нужно поменять (список товаров / конкретный склад / конкретный магазин). После завершения цикла происходит выбор поля, которое необходимо поменять. Пользователь вводит новое значение поля и подтверждает или отменяет свой выбор.