ЗМІСТ

ПЕРЕЛІК УМОВНИХ ПОЗНАЧЕНЬ, СИМВОЛІВ, ОДИНИЦЬ ВИМІРЮВАННЯ, СКОРОЧЕНЬ ТА ТЕРМІНІВ

БД – База даних

ДВВ – ДВВ

ДК – діаграми класів

ДП – діаграма послідовності

ДС – діаграма станів

ОС – операційна система

ПЗ – програмне забезпечення

ПК – персональный компьютер

ПМ – програмний модуль

ПП – програмний продукт

API – Application Programming Interface (интерфейс программирования приложений)

CRM – Customer Relationship Management (Система керування взаєминами із клієнтами)

GIS – Геоінформаційна система

UML – unified modeling language

ВСТУП

Сучасний етап розвитку ринкових відносин в Україні характеризується значним розширенням масштабів господарської діяльності, створенням нових підприємств. У цих умовах підприємства прагнуть підвищити свою конкурентоспроможність. Транспортно-складська логістика – це система організації та оптимізації складських операцій, а саме переміщення та доставка яких-небудь матеріальних предметів з одного місця в інше за оптимальним маршрутом [1,2].

Сьогодні Україна є одним із найбільших у Східній Європі транзитних коридорів по транспортуванню вантажів. Особливо великою, розгалуженою транспортною інфраструктурою є Одеський регіон, що має три найбільші порти в Чорному морі. В даних умовах виявляється досить велика кількість, як дрібних, так і великих підприємств, основним видом діяльності яких є експедирування вантажів, як по території України, так і за її межі.

У зв’язку з цим актуальним є розроблення інтегрованих автоматизованих систем управління логістичними процесами і методів моделювання транспортного процесу з метою оптимізації витрат підприємства.

Для постановки задачі наукового дослідження обрано підприємство, яке має власний складський комплекс, на якому складуються товари різних габаритів і маси. На підприємстві є автопарк вантажних автомобілів. На протязі робочого дня формуються замовлення на доставку товарів в різні точки, кількість, яких заздалегідь невідома. У нічний час на складах підприємства проводиться завантаження необхідного товару на автомобілі, які на наступний день вранці розвозять їх в торгові точки.

Актуальність проблеми полягає в необхідності нового підходу до побудови транспортно-складських логістичних маршрутів: підходу, який зміг би забезпечити підприємству довгострокові конкурентні переваги при найменших втратах та за найкоротший час.

Метою даної роботи є розроблення програмного забезпечення, яке інтегрується у CRM систему підприємства і дозволяє оптимізувати транспортно-складські логістичні потоки. Система надає картографічну інформацію про логістичні потоки підприємства, а також дозволяє автоматизувати його документообіг.

Для досягнення мети магістерської работи необхідно вирішити наступні завдання:

1) Провести аналіз предметної області та обґрунтувати вибір технологічної платформи.
2) Виконати дослідження та обґрунтувати вибір методів для оптимального планування вантажоперевезень.
3) Виконати основні етапи проектування програмного модулю, створити діаграми варіантів використання, діаграми послідовності та діаграми класів.
4) Описати етапи розробки програмного забезпечення, привести керівництво користувача.
5) Виконати тестування програмного забезпечення.

1 АНАЛІТИЧНА ЧАСТИНА

1.1 Аналіз предметної області

Серед безліч проблем обліку на підприємствах оптової торгівлі та підприємствах, що займаються складським зберіганням та доставкою товарів, є проблема збільшення швидкості і зменшення витрат вантажно-завантажувальних робіт, а також визначення оптимальних маршрутів доставки товарів замовникам.

Створення автоматичного виробництва, розумно починати з розробки моделі взаємодії зі споживачами. Система керування взаєминами із клієнтами CRM (скорочення від англ. Customer Relationship Management) – прикладне програмне забезпечення для організацій, призначене для автоматизації стратегій взаємодії із замовниками (клієнтами), зокрема, для підвищення рівня продажів, оптимізації маркетингу і поліпшення обслуговування клієнтів шляхом збереження інформації про клієнтів та історії взаємин з ними. Підтримка цих бізнес-цілей включає в себе збір, зберігання та аналіз інформації про споживачів, постачальників, партнерів, а також про внутрішні процеси компанії. Функції для підтримки бізнес-цілей включають закупівлі, продаж, маркетинг, логістику, підтримку споживачів [3,4].

На даний момент на ринку логістики в Україні можна спостерігати конкуренцію й прискорення процесу консолідації логістичного ринку. На думку фахівців, посилюються тенденції до перерозподілу попиту на користь великих і середніх, а також вузькоспеціалізованих компаній, чому буде сприяти підвищення вимог до якості послуг, надійності підрядників, безпеки, а також відхід з ринку неефективних гравців.

На сьогодення в Україні на підприємствах, що займаються логістикою, складським зберіганням і експедируванням, застосовуються кілька різних програмних продуктів. У більшості випадків для ведення складського (товарно-матеріального) обліку й бухгалтерії застосовуються програми сімейства 1С Підприємство. Фірма 1С здійснює підтримку своїх програмних продуктів залежно від зміни податкового законодавства й стандартів ведення бухгалтерського обліку.

На сьогоднішній день в Україні на підприємствах, що займаються логістикою, складським зберіганням і експедируванням, застосовуються кілька різних програмних продуктів. У більшості випадків для ведення складського (товарно-матеріального) обліку і бухгалтерії застосовуються програми

сімейства 1С Підприємство. Велика мережа франчайзингових компаній та підприємців здійснює підтримку, та розробку як своїх так і офіційних конфігурацій залежно від зміни податкового законодавства і стандартів ведення бухгалтерського обліку, що ж стосується сторонніх продуктів, що спеціалізуються на експедируванні та логістиці – у них не передбачена дана функція, що приводить до додаткових витрат на повторне введення однієї й тієї ж інформації в різні програмні продукти. Тому було ухвалене рішення розробити програмний модуль, який вирішував би проблеми підприємства, що займається вантажоперевезеннями, складською логістикою і був би «вбудований» у найпоширеніший програмний продукт серед підприємств малого і середнього бізнесу на Україні.

Програмний модуль повинен передбачати наявність карти з контрольними пунктами (підприємствами, їх складами та місцями торгівлі), перелік яких вводить користувач. Один запис у таблиці контрагентів, умовно назвемо «карткою контрагента», заповнюється коли підприємства складають договір на поставку товару або продукції. Після чого програмним модулем у певні поля картки контрагента будуть повернуті значення довготи і широти його місця розташування. Також є необхідним передбачити побудову оптимального маршруту між складом і місцями доставки, з метою економії коштів на пально-мастильні матеріали і оплату автотранспорту (при тимчасовій його оренді). Однієї з важливих функцій програмного модулю є автоматичне створення супровідних документів та звітів про маршрут і навантаження транспорту за запитом користувача.

У якості середовища розробки було обрано «1С: Підприємство 8х», оскільки інтегрування модулів, що розроблюються, є можливим в стандартних конфігураціях даної CRM системи. Варто відзначити, що впровадження нових модулів абсолютно не заважає подальшому відновленню конфігурацій, необхідність яких виникає у зв'язку з постійними змінами законодавства та стандартів обліку.

Даний програмний продукт поставляється з відкритим кодом і може бути адаптованим під потреби того або іншого підприємства за допомогою мов програмування 1С і Java.

Для виводу тайлових карт було вирішено використовувати сервіс Google Maps API. Компанія Google постійно підтримує свої карти в актуальному стані, що зменшує витрати на використанні власних карт. На рис. 1.1 представлена архітектура програмного модуля, що розроблюється.

Архітектура кінцевого вирішення 1С ділиться на «платформу», яка містить у собі саме ядро, бібліотеку компонент платформи, а так само засоби для розробки, тобто сам конфигуратор [5].

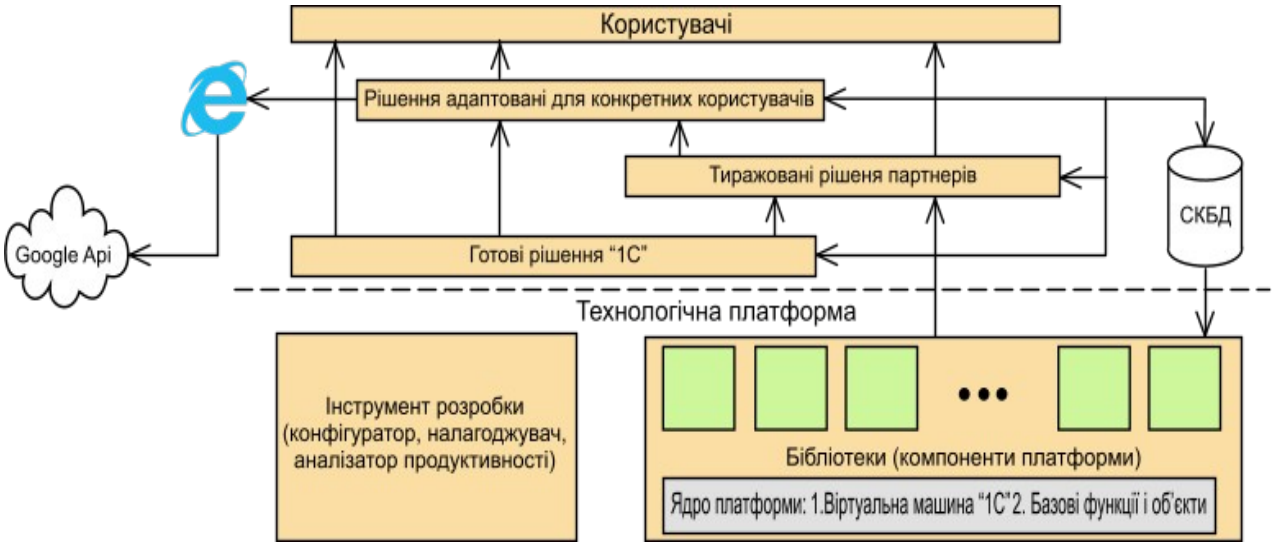


Рисунок 1.1 – Загальна архітектура програмного модуля

Друга частина є прикладним рішенням для кінцевих користувачів. У більшості випадків нові конфігурації, крім тих, які поширюються виробником платформи не пишуться, за винятком тих випадків, коли необхідно одержати таке вирішення, якого раніше не було. Дуже часто розроблювачі змінюють готові рішення під потреби замовника і створюють тиражовані конфігурації, які можуть працювати на більшості підприємств даної галузі. І якщо підприємству необхідно доробити таке тиражоване рішення або готову конфігурацію, воно може запросити програміста, який її доробить під потреби користувачів конкретного підприємства. У запропонованій роботі виконана саме доробка конфігурації, адаптована під потреби конкретного підприємства, з якої цілком можливо створити й тиражовану конфігурацію.

1.2 Обґрунтування вибору технологічної платформи

«1С:Підприємство 8. Бухгалтерія для України» – це багатофункціональне програмне забезпечення економічного призначення, що має безліч різноманітних функцій, які допомагають організувати роботу будь-якого підприємства.

Технологічна платформа «1С:Підприємство» являє собою програмну оболонку над базою даних (БД). Використовуються БД на основі Dbf-файлів, або MsSQL SERVER у версії 7.7, власний формат 1CD з версії 8.0 або будь-яка із СКБД, таких, як Microsoft SQL Server, PostgreSQL і, а з версії 8.2 була додана й. Проекти з вбудованою мовою 1С:Підприємства називаються конфігураціями. Поширення (продаж) і впровадження таких конфігурацій – це основна комерційна діяльність фірм-партнерів 1С. [5] Вбудована мова 1С:8 є об'єктивно - орієнтованою і найбільш подібна за своїм синтаксисом до мови Visual Basic.

Клієнтська частина платформи функціонує в, а починаючи з версії 8.3, також у середовищі Linux і MacOS. Починаючи з версії 8.1, серверна частина платформи може функціонувати на ОС MS Windows і Linux. Платформою надається фіксований набір базових класів, орієнтованих на вирішення типових завдань прикладної області (рис. 1.2)

Даний програмний продукт поставляється з відкритим кодом і може бути легко адаптованим під потреби того або іншого підприємства будь-яким програмістом зі знаннями 1С и Java. Його можна використовувати на середніх і великих підприємствах, що займаються оптовою торгівлею, логістикою, експедируванням або на складських підприємствах, що займаються плануванням доставки товарів до кінцевого покупця.

1.3 Принципи проектування логістичних систем.

З позиції менеджменту організації – логістику можна розглядати, як стратегічне управління матеріальними потоками в процесі постачання: закупівлі, перевезення, продажу та зберігання матеріалів, деталей і готового інвентарю (техніки та іншого). Поняття включає всебе також управління відповідними потоками інформації, а також фінансовими потоками. Логістика спрямована на оптимізацію витрат і раціоналізацію процесу виробництва, збуту і супутнього сервісу як в рамках одного підприємства, так і для групи підприємств.

Транспортна логістика – це система по організації доставки, а саме по переміщенню будь-яких матеріальних предметів, речовин тощо. З однієї точки в іншу за оптимальним маршрутом. Більш детальними функціями даної логістики:

– персонал, який займається здійсненням цих завдань (вантажники, водії);

– класифікація транспортних засобів (за обсягами, м³).

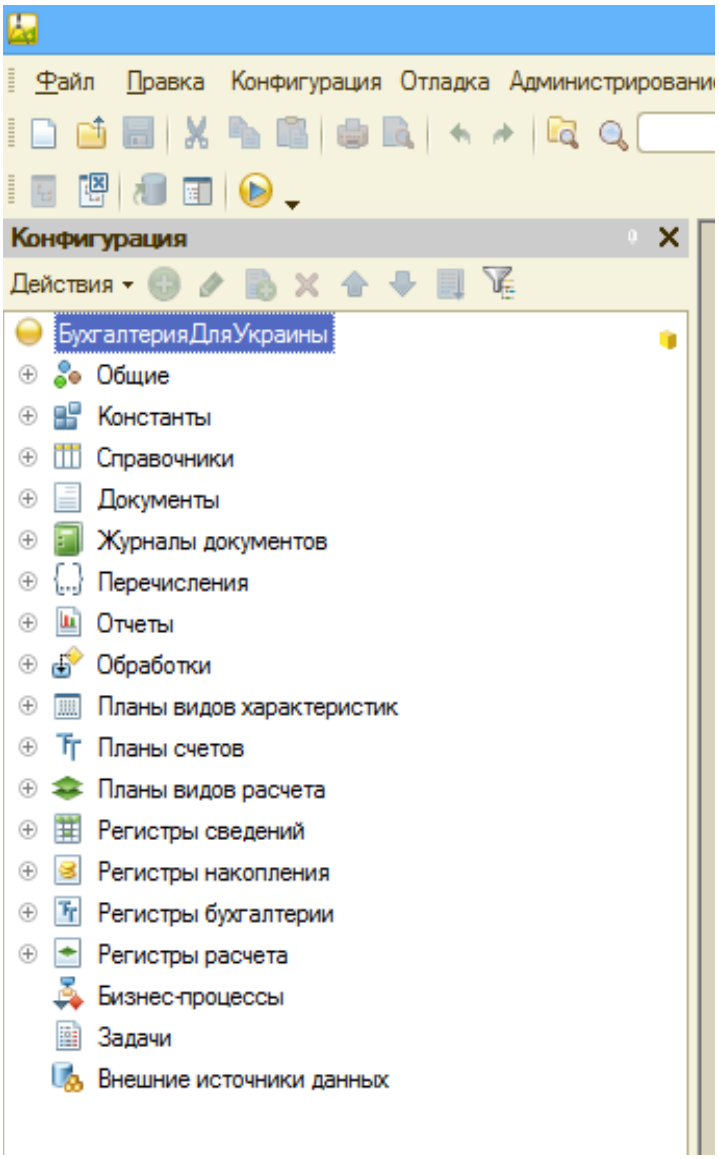


Рисунок 1.2 – Дерево конфігурації «1С: Підприємство: Бухгалтерія для України»

Під транспортно-логістичної системою розуміється сукупність споживачів і виробників послуг, а також використовувані для їх надання системи управління, транспортні засоби, шляхи сполучення, споруди та інше майно. В іншому визначенні говориться про те, що транспортно-логістична система – сукупність об'єктів і суб'єктів транспортної та логістичної інфраструктури разом з матеріальними, фінансовими та інформаційними потоками між ними, що виконує функції транспортування, зберігання, розподілу товарів, а також інформаційного та правового супроводу товарних потоків.

Виходячи з вищесказаного при побудові логістичних систем найбільшу увагу варто приділяти бізнеспроцессам, які працюють на підприємстві, на

підставі їх відстежувати інформаційні потоки, які будуть необхідні для вирішення задачі оптимізації витрат підприємства.

1.3 Огляд найближчих аналогів

Як було визначено раніше, програма автоматизації транспортної логістики – це зручний інструмент логіста й помічник начальника в керуванні підприємством. На даний момент існує дуже невелика кількість ПП, які займаються оптимізацією маршрутів, а також складанням графіків навантаження.

З найближчих аналогів це:

– мурашина логістика: заснована на картах Open Street Map;

– transnavicom: використовують свою інформаційну систему, є одним із постачальників даних для карт «Яндекс» і «Google».

Кожна із запропонованих вище програм має ті або іншими перевагами й недоліками, однак на даний момент не існує сервісу, сумісного з матеріальним обліком товарів на складах і бухгалтерією підприємства, і в якім було б зручно ввести ці види обліку. Уся суть роботи існуючих сервісів полягає тільки в побудові маршруту між підприємствами. Крім того дані сервіси не займаються оптимізацією маршрутів згідно з математичних методів, їх оптимизація полягає лише в оптимізації згідно з завантаженням автотранспорту. Метод Кларка-Райта передбачає у собі наступні критерії для побудови оптимизованого маршруту:

- загрузка автотранспорту;
- чи не знаходиться точка в іншому маршруті;
- початковий пункт пов'язаний з кінцевим пунктом.

1.4 Розгляд найближчих методів для реалізації задачі

Серед багатьох існуючих методів потрібно було обрати метод, завдяки котрому можно було б обрати найвигідніший маршрут руху автотранспорту який повинен проходити один раз через пункти розвозки товарів з подальшим поверненням в початковий пункт. Найголовнішим критерієм при вирішенні даної задачі є: мінімальний пробіг транспортного засобу при максимальному завантажені автомобіля.

Дану задачу ми знаєм, як «задачу комівояжера». Задача комівояжера – одна з найвідоміших задач комбінаторної оптимізації, що полягає в пошуку

самого вигідного маршруту, що проходить через зазначені міста хоча б по одному разу з наступним поверненням в початковий місто. В умовах задачі вказуються критерій вигідності маршруту (найкоротший, найдешевший, сукупний критерій тощо) і відповідні матриці відстаней, вартості тощо. Як правило, вказується, що маршрут повинен проходити через кожне місто тільки один раз – в такому випадку вибір здійснюється серед гамільтонових циклів. Існує кілька окремих випадків загальної постановки завдання, зокрема, геометрична задача комівояжера (також звана планарной або евклідової, коли матриця відстаней відображає відстані між точками на площині), метрична задача комівояжера (коли на матриці вартостей виконується нерівність трикутника), симетрична і асиметрична завдання комівояжера.

NP-повна задача – в теорії алгоритмів завдання з відповіддю «так» або «ні» з класу NP, до якої можна звести будь-яку іншу задачу з цього класу за поліноміальний час (тобто за допомогою операцій, число яких не перевищує деякого полінома в залежності від розміру вихідних даних). Таким чином, NP-повні задачі утворюють в деякому сенсі підмножина «типових» задач в класі NP: якщо для якоїсь з них знайдений «полиномиально швидкий» алгоритм рішення, то і будь-яка інша задача з класу NP може бути вирішена так само «швидко»[6].

Існує багато математичних методів, котрі дають можливість знайти, як і точне, так і приблизне рішення даної задачі. Серед методів, котрі дають більш точне рішення відомо методи повного перебору та гілок і меж.

Розглянемо більш детально дані методи.

1.4.1 Задача вибору методом повного перебору

Задача вибору оптимального пошуку оптимального маршруту, котра вирішується повним перебором і оцінюванням всіх можливих варіантів розміщення розподільчих центрів і виконується на ЕОМ методами математичного програмування. Однак на практиці в умовах розгалужених транспортних мереж метод може виявитися непридатний, тому що число можливих варіантів по міру збільшення масштабів мережі, а з ними і трудомісткість вирішення, зростають за експоненті.

1.4.2 Задача гілок і меж

Результатом роботи алгоритму є знаходження максимуму функції на допустимій множині. При чому множина може бути як дискретною, так і раціональною. В ході роботи алгоритму виконується дві операції: розбиття вихідної множини на підмножини(гілки), та знаходження оцінок(меж). Існує оцінка множини згори та оцінка знизу. Оцінка згори – точка що гарантовано не менша за максимум на заданій підмножині. Оцінка знизу – точка що гарантовано не більша за мінімум на заданій підмножині. Множина що має найбільшу оцінку зверху зветься рекордною. На початку вся множина вважається рекордною[7].

Основним недоліком даних методів є висока часова і ємкісна складність, що важливо враховувати при великій кількості пунктів. Всі ефективні (скорочують повний перебір) методи вирішення «завдання комівояжера» – методи евристичні. З них найбільше застосування знайшли:

- «Метод Кларка-Райта»;
- «Алгоритм мурашиної колонії»;
- «Метод найближчого сусіда»;
- «Метод найдешевшого включення»

1.5 Еврестичні методи

Евристичні методи – послідовність приписів або процедур обробки інформації, яка виконується з метою пошуку більш раціональних і нових конструктивних рішень [8].

Евристичні методи зазвичай протиставляють формальним методам вирішення, що спирається на точні математичні моделі.

Евристичні методи забезпечують виявлення, обробку та впорядкування системи закономірностей, механізмів і методологічних засобів. конструювання нового завдання і цілеспрямованих способів діяльності на основі узагальнення попереднього досвіду і випереджального відображення моделей майбутнього з метою повного задоволення потреб моделей.

1.5.1 Алгоритм мурашиної колонії

В основі алгоритму лежить поведінка мурашиної колонії [9] – маркування більш вдалих шляхів великою кількістю феромона. Робота починається з розміщення мурашок у вершинах графа (містах), потім починається рух

мурашок – напрям визначається імовірнісним методом, на підставі формули виду:

$$P_i = \frac{l_i^q * f_i^p}{\sum_{k=0}^{N} l_k^q * f_k^p}, \quad (1.1)$$

Де:

P_i – ймовірність переходу по шляху і

,

l_i – величина, зворотна вазі (довжині) і-го переходу,

f_i – кількість феромону на і-м переході,

q – величина, що визначає «жадібність» алгоритму,

p – величина, що визначає «стадність» алгоритму і.

$$q + p = 1. \quad (1.2)$$

Рішення не є точним і навіть може бути одним з найгірших, однак, в силу імовірнісний рішення, повторення алгоритму може видавати (досить) точний результат.

1.5.2 Метод найближчого сусіда

Формулюється таким чином: Алгоритм найближчого сусіда [6] починається в довільній точці та поступово відвідує кожну найближчу точку, яка ще не була відвідана. Пункти обходу плану послідовно включаються до маршруту, причому, кожен черговий пункт, що включається до маршруту, повинен бути найближчим до останнього вибраного пункту серед усіх інших, ще не включених до складу маршруту. Алгоритм завершується, коли відвідано всі точки. Остання точка з'єднується з першою[10].

Алгоритм простий у реалізації, швидко виконується, але, як і інші «жадібні» алгоритми, може видавати неоптимальні рішення. Обчислювальна складність алгоритму – O(n2). Результатом виконання алгоритму найближчого сусіда є маршрут, приблизно на 25% довший від оптимального.

1.5.3 Метод найдешевшого включення

Метод найдешевшого включення на кожному кроці алгоритму проводить припустимий маршрут по поточному подмножеству пунктів обходу, вже

включених в маршрут, додаючи до нього новий пункт, включення якого між деякими суміжними пунктами приводить до мінімального збільшення вартості (довжини) маршруту [11].

Метод мінімального остовного дерева становлять 3 послідовно виконуваних кроку.

На 1-му кроці для безлічі пунктів плану, інтерпретованих як вершини повного графа, будується найкоротший кістяк (за допомогою алгоритму Прима). Дублюючи кожне ребро і задаючи кратним ребрах протилежні напрямки кістяк перетворюється в орієнтований мультіграф, який є ейлеровим, тому що всі його вершини з побудови мають парну ступінь.

На 2-му кроці в побудованому мультиграфом виділяється маршрут мінімальної довжини, який проходить через кожен пункт не менше 1 разу, при цьому кожна ланка маршруту обходу плану (пара суміжних вершин) з'являється в ньому рівно один раз. Маршрут такий структури називається ейлеровим циклом і задається послідовністю, яку утворює деяка перестановка з повтореннями на безлічі номерів пунктів обходу для прийнятої нумерації пунктів (зліва-направо і зверху-вниз в координатах плану).

На 3-му кроці з послідовності обходу виключаються всі повторні входження кожного пункту. Таке виключення є коректним, тому, що за визначенням ейлерова циклу кожен пункт обходу зустрічається в маршруті не менше 1 разу. В результаті виключення повторюваних пунктів Ейлером цикл перетворюється в гамільтонів, топологія якого становить шукане наближене рішення задачі комівояжера і утворює допустимий маршрут, можливо, задовольняє вимогам конкретного прикладного завдання.

1.6 Алгоритм Кларка-Райта

При організації кільцевих маршрутів руху автомобіля, тобто коли автомобіль відправляється з тієї ж точки, в яку після розвезення всієї продукції повинен повернутися, необхідно визначення раціональної послідовності об'їзду пунктів, щоб здійснити перевезення з мінімальним пробігом [12].

Для вирішення завдань маршрутизації перевезень дрібно партійних вантажів існує дві групи методів, при застосуванні яких можна отримати точні або приблизні результати. У зв'язку з тим, що застосування точних методів обмежене розмірністю розв'язуваних завдань, то на практиці користуються в основному приблизними методами.

Виконавши аналіз різних способів вирішення завдання, було прийнято рішення використовувати метод Кларка-Райта. Перевагами цього методу є його простота, надійність і гнучкість, що дозволяє враховувати цілий ряд додаткових факторів, які впливають на кінцеве рішення задачі. Цей алгоритм використовує поняття виграшів, щоб оцінити операції злиття між маршрутами. Виграш – міра скорочення вартості, отримана комбінуванням двох маленьких маршрутів в один великий маршрут. Тобто маршрути, які виходять із одного пункту вантажовідправника (ВВ), попарно групуються в кільцеві маршрути за принципом отримання на кожному максимального виграшу від цього об'єднання. Виграш від об'єднання пунктів *i* і *j* маршрутів визначається за формулою (1.3)

$$\Delta ij = l_{i,0} + l_{0,j} - l_{i,j}, \qquad (1.3)$$

Де:

$l_{i,0}$ – найкоротша відстань від пункту *i* до ВВ,

$l_{0,j}$ – найкоротша відстань від ВВ до *j* пункту,

$l_{i,j}$ – найкоротша відстань від пункту і до пункту j.

За оцінкою всіх можливих комбінацій об'єднань пунктів *i* і *j* в пари (в таблиці оцінок), в першу чергу включають в маршрут пару вершин, що мають максимальне значення у виграші. При наступному кроці включення проводиться або на вході в маршрут (в точці *i*), або на виході з нього (в точці *j*).

В даному випадку відшукується максимальний виграш в стовпці *i* і в рядку *j* таблиці оцінок, в залежності від якого здійснюють включення чергового пункту у фрагмент маршруту, що будується.

При побудові маршруту здійснюється перевірка на задоволення обмеження (по вантажомісткості автомобіля, часу знаходження в наряді, термінів доставки вантажу тощо). Формування маршруту закінчується при вичерпанні списку вершин або відсутності можливості включення пункту без порушення заданих обмежень. В останньому випадку приступають до побудови чергового маршруту. Процедура повторюється до отримання всього плану маршрутизації.

Погрішність рішення не перевершує в середньому 5-10%. Однак, з огляду на жадібний характер алгоритму Кларка-Райта, отримані рішення мають часто недостатню якість щодо більш складних підходів. Необхідно також урахувати, що після перших декількох ітерацій у завданнях з багатьма обмеженнями

ймовірність злиттів маршруту може рішуче зменшитися, і не має можливості контролювати кількість маршрутів.

Додамо таким чином 12 пунктів, та повернемося до самого алгоритму. Необхідно розвести товар зі складу до 12 пунктів. На даний момент – це 12 радіальних маршрутів, які представлені на рис.1.3, де «шт» – це кількість вантажу, який необхідно відвезти до кожного з пунктів.

Суть алгоритму полягає у тому, щоб відштовхуючись від початкової схеми перевезення за кроками перейти до оптимальної схеми з кільцевими маршрутами.

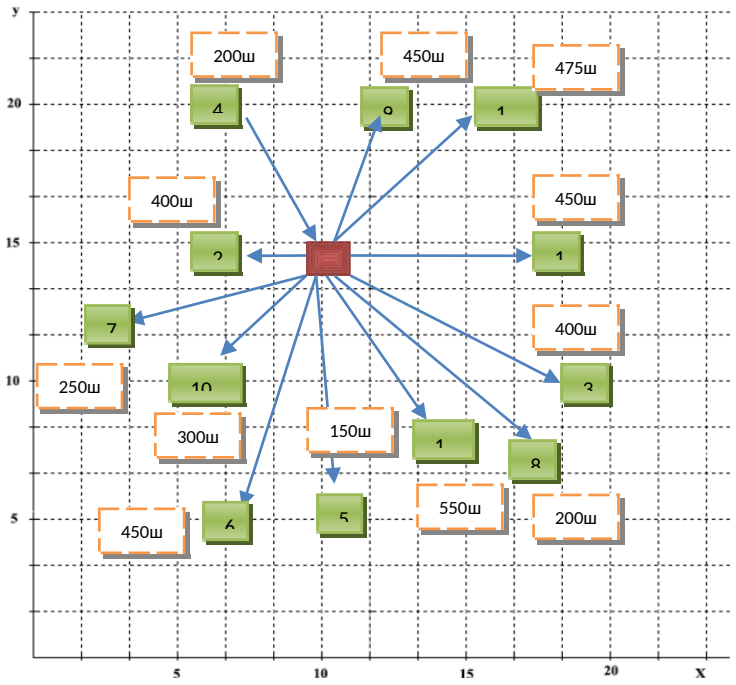


Рисунок 1.3 – Радіальні маршрути

З даною метою введемо таке поняття, як «кілометровий виграш». На рис.1.4 відображено схеми доставки. Схема А забезпечує доставку вантажу за радіальним маршрутом. Схема В – за кільцевим. Пробіг автотранспорту для схеми А розраховується за формулою (1.4), для схеми В – за формулою (1.5).

$$L_a = d_{01} + d_{10} + d_{02} + d_{20} = 2d_{01} + 2d_{02}, \qquad (1.4)$$

$$L_B = d_{01} + d_{12} + d_{02}, \qquad (1.5)$$

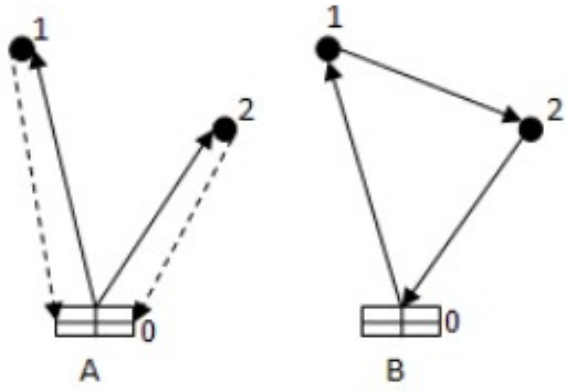


Рисунок 1.4 – Схеми доставки:
А – за радіальним маршрутом; В – за кільцевим маршрутом

Як правило за показниками схема В дає кращий результат ніж А. При переході від схеми А до В – отримаємо кілометровий виграш (1.6):

$$S_{12}=L_a-L_b=d_{01}+d_{02}-d_{12}. \qquad (1.6)$$

Загалом формула для розрахунку кілометрового виграшу виглядає наступним чином (1.7):

$$S_{ij}=d_{0i}+d_{0j}-d_{ij}\ , \qquad (1.7)$$

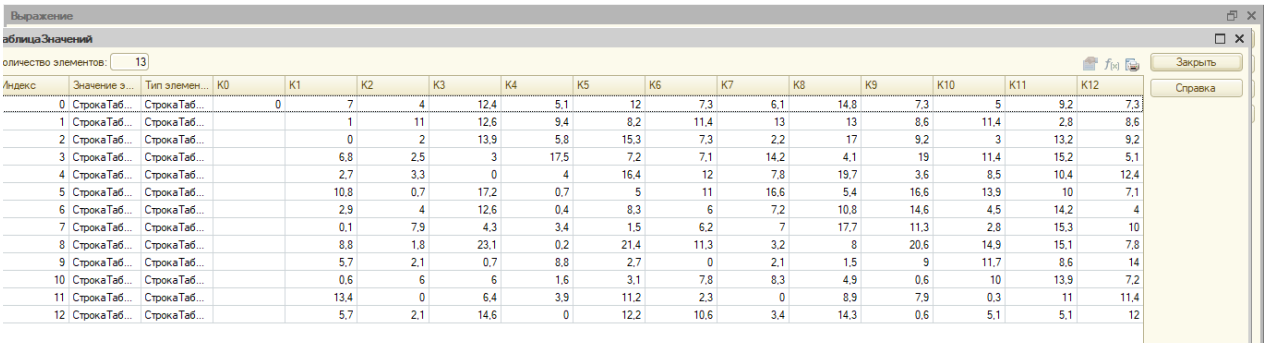
Де S_{ij} – кілометровий виграш отриманий в результаті об’єднання пунктів i та j відповідно, км. d_{ij} – це відстань між пунктами i та j.

Дані значення заносяться в таблицю (табл.1.1).

Таблиця 1.1 - Матриця відстані та кілометрового виграшу

Матриця кілометрових виграшів S_{ij}	0	d_{01}	d_{02}	d_{03}	…	d_{0r}
	S_{10}	1	d_{12}	d_{13}	…	d_{1r}
	S_{20}		2	d_{23}	…	d_{2r}
	S_{30}			3	…	d_{3r}
	…	…	…	…	…	…
	S_{r0}	S_{r1}	S_{r2}	S_{r3}	…	r

Далі отримаємо матрицю кілометрових виграшів (рис.1.5).


Выражение

аблицаЗначений

оличество элементов: 13

Закрыть

Справка

Индекс	Значение э...	Тип элемен...	К0	К1	К2	К3	К4	К5	К6	К7	К8	К9	К10	К11	К12
0	СтрокаТаб...	СтрокаТаб...	0	7	4	12,4	5,1	12	7,3	6,1	14,8	7,3	5	9,2	7,3
1	СтрокаТаб...	СтрокаТаб...		1	11	12,6	9,4	8,2	11,4	13	13	8,6	11,4	2,8	8,6
2	СтрокаТаб...	СтрокаТаб...		0	2	13,9	5,8	15,3	7,3	2,2	17	9,2	3	13,2	9,2
3	СтрокаТаб...	СтрокаТаб...		6,8	2,5	3	17,5	7,2	7,1	14,2	4,1	19	11,4	15,2	5,1
4	СтрокаТаб...	СтрокаТаб...		2,7	3,3	0	4	16,4	12	7,8	19,7	3,6	8,5	10,4	12,4
5	СтрокаТаб...	СтрокаТаб...		10,8	0,7	17,2	0,7	5	11	16,6	5,4	16,6	13,9	10	7,1
6	СтрокаТаб...	СтрокаТаб...		2,9	4	12,6	0,4	8,3	6	7,2	10,8	14,6	4,5	14,2	4
7	СтрокаТаб...	СтрокаТаб...		0,1	7,9	4,3	3,4	1,5	6,2	7	17,7	11,3	2,8	15,3	10
8	СтрокаТаб...	СтрокаТаб...		8,8	1,8	23,1	0,2	21,4	11,3	3,2	8	20,6	14,9	15,1	7,8
9	СтрокаТаб...	СтрокаТаб...		5,7	2,1	0,7	8,8	2,7	0	2,1	1,5	9	11,7	8,6	14
10	СтрокаТаб...	СтрокаТаб...		0,6	6	6	1,6	3,1	7,8	8,3	4,9	0,6	10	13,9	7,2
11	СтрокаТаб...	СтрокаТаб...		13,4	0	6,4	3,9	11,2	2,3	0	8,9	7,9	0,3	11	11,4
12	СтрокаТаб...	СтрокаТаб...		5,7	2,1	14,6	0	12,2	10,6	3,4	14,3	0,6	5,1	5,1	12

Рисунок 1.5 – Розрахункова матриця відстані та кілометрового виграшу.

Згідно алгоритму необхідно знайти комірку (i,j) з максимальним кілометровим виграшем Smax (1.8)

$$S_{max}=max_{i,j}S(i,j)=S\text{¿}. \qquad (1.8)$$

Для правильного рішення задачі повинні виконуватися наступні умови:

– Пункти i та j не повинні входити в склад одного й того ж маршруту;
– Пункти i та j є начальними або кінцевими тих маршрутів до складу котрих входять;
– Комірка i та j не заблокована (розглядалася на попередніх кроках алгоритму).

Якщо не вдалося знайти комірку, котра задовольняє умовам, тоді: маршрут до складу котрого входить i – це маршрут 1. Відповідно для j – 2.

Приймемо наступні умовні значення:

- N = {1, 2, 3,…, n} – множина одержувачів;
- N_1= ($N_1 \subset$ N) – підмножина пунктів у складі маршруту 1;
- N_2= ($N_2 \subset$ N) – підмножина пунктів у складі маршруту 2.

Тепер необхідно розрахувати сумарний об’єм поставок за маршрутами 1 та 2. Розрахунок проводимо за формулою (1.9):

$$q_1=\sum_{k\in N_1} q_k \text{ і } q_2=\sum_{k\in N_2} q_k, \qquad (1.9)$$

Де q_k – це обсяг попиту k-го пункту, шт. Перевіримо на виконання наступну умову за формулою (1.10):

$$q_1 + q_2 \leq c \ . \qquad (1.10)$$

Де:

c – вантажомісткість авто, шт.,

q_1 – є кінцевим пунктом маршрута 1,

q_2 – є початковим пунктом маршрута 2.

Якщо умова виконується, тоді об'єднуємо маршрути 1 та 2 в кільцевий маршрут х.

При обєднанні маршрутів 1, 2 необхідно виконувати наступні умови:

- Послідовність пунктів на маршруті 1 від начала *i* до пункту *j* – не змінюється;
- Пункт *i* зв'язаний з пунктом *j*;
- Послідовність пунктів на маршруті 2 від пункту *j* і до кінця не змінюється.

Якщо умова виконується, тоді розраховуємо сумарний пробіг автотранспорту для 12 пунктів. Весь хід послідовного вирішення задачі представлений у таблиці (рис. 1.6). Де, стовбець 1 – номер ітерації; стовбці 2,3 – номера пунктів *i** і *j**, котрі позначують комірку з максимальним кілометровим виграшем; стовбець 4 – значення максимального кілометрового виграшу Smax; стовбці 5,6,7 – результати перевірки умов 1,2,3. Якщо умова виконується – тоді «+», якщо ні – «–»; стовбці 8, 9 – об'єм перевезень за маршрутом 1, до складу котрого входить пункт *i* (q_1)* , та маршруту 2 до складу котрого входить пункт *j* (q_2)*; стовбець 10 – перевірка на умову: $q_1 + q_2 \leq c$, де c – вантажомісткість транспортного засобу, «+» – позитивний результат перевірки умови, «–» – негативний результат, відповідно; стовбець 11 – порядковий номер кільцевого маршруту; стовбець 12 – структура кільцевого маршруту, що був створений на даній ітерації (рис.1.7).

В ході рішення було отримано 4 кільцевих маршрути (рис.1.7).

Для побудови даної схеми було зроблено 20 ітерацій, сумарний кілометровий виграш котрих склав (1.11):

$$S = 23.0 + 21.4 + 14.6 + 13.4 + 8.8 + 8.3 + 7.9 + 7.8 = 105.3 км. \qquad (1.11)$$

А загальний пробіг автотранспорту (1.12):

$$L_1 = L_0 - S = 195 - 105.3 = 89.7\,км \quad . \qquad (1.12)$$

Результати рішення представленні на рис.1.8

№ п/п	Крок 1						Крок 2		Крок 3	Крок 4		
	i*	j*	Smax	Умови			q1, шт	q2, шт	q1 + q2 ≤ c, c = 1500 шт.	№ маршрута	Маршрут	Об'єм перевезення, шт
				1	2	3						
1	2	3	4	5	6	7	8	9	10	11	12	13
1	8	3	23,0	+	+	+	400	200	+	1	0-8-3-0	600
2	8	5	21,4	+	+	+	600	150	+	1	0-5-8-3-0	750
3	5	3	17,2	-	+	+	-	-	-	-	-	-
4	12	3	14,6	+	+	+	750	550	+	1	0-5-8-3-12-0	1300
5	12	8	14,2	-	-	+	-	-	-	-	-	-
6	11	1	13,4	+	+	+	475	450	+	2	0-1-11-0	925
7	6	3	12,6	+	-	+	-	-	-	-	-	-
8	12	5	12,3	-	+	+	-	-	-	-	-	-
9	11	5	11,3	+	+	+	925	1300	-	-	-	-
10	8	6	11,2	+	-	+	-	-	-	-	-	-
11	5	1	10,8	+	+	+	1300	925	-	-	-	-
12	12	6	10,6	+	+	+	1300	450	-	-	-	-
13	11	8	8,9	+	-	+	-	-	-	-	-	-
14	9	4	8,8	+	+	+	450	200	+	3	0-9-4-0	650
15	8	1	8,8	+	-	+	-	-	-	-	-	-
16	6	5	8,3	+	+	+	450	1300	-	-	-	-
17	10	7	8,3	+	+	+	300	250	+	4	0-10-7-0	550
18	11	9	7,9	+	+	+	925	650	-	-	-	-
19	7	2	7,9	+	+	+	550	400	+	4	0-10-7-2-0	950
20	10	6	7,8	+	+	+	950	450	+	4	0-6-10-7-2-0	1400

Рисунок 1.6 – Розрахункова таблиця проміжних результатів рішення задачі

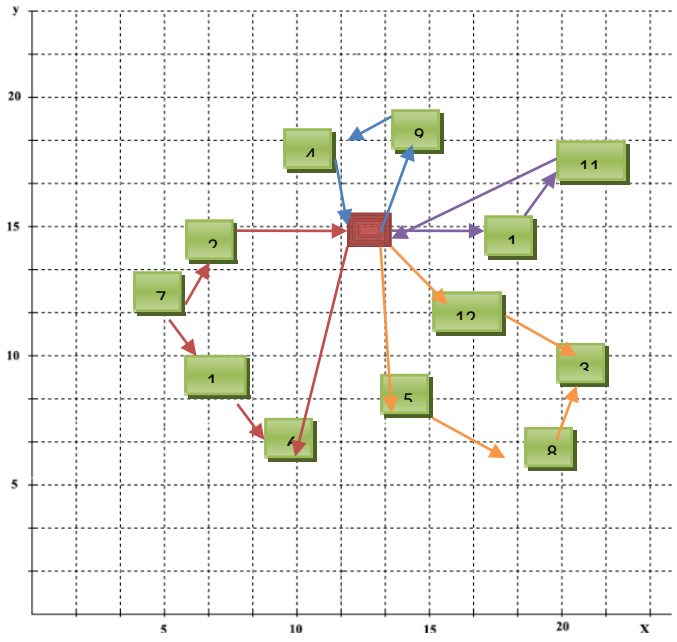


Рисунок 1.7 – Графічне відображення оптимальної схеми доставки

№ п/п	Маршрут	Об'єм поставки, шт	Пробіг, км
1	0-5-8-3-12-0	1300	33,9
2	0-1-11-0	925	19,0
3	0-9-4-0	650	16,0
4	0-6-10-7-2-0	1400	20,8
	Разом:	**4275**	**89,7**

Рисунок 1.8 – Результати рішення задачі методом Кларка-Райта

1.7 Висновки до розділу

У даному розділі дипломної роботи були проаналізовани проблеми логістичних операцій на підприємстві, вибрана технологічна платформа для розробки програмного забезпечення. Сформавана задача та вибран найбільш приємний метод її рішення, а саме спроектовано:

– Сформульована постановка завдання на розробку програмного забезпечення;

– Розроблена загальна архітектура програмного модуля;

– Вибрана технологічна платформа для реалізації задачі;

– Розглянуты найбіль близькі аналогі та методи реалізації задачи;

– Вибран метод реалізації Кларка-Райта, як найбіль оптимальний для рішення данної задачи.

2 ПРОЕКТНА ЧАСТИНА

У проектній частині дипломної роботи розглядаються проекті рішення, що запропоновані розробником. Зміст проектної частини визначається, по-перше, специфікою теми дипломної роботи, по-друге, особливостями конкретних технічних пропозицій до роботи.

2.1 Призначення програмного модулю

Програмний модуль (ПМ), що розроблюється, призначений для створення системи керування складськими і логістичними операціями підприємства. Для роботи з ним, досить базових знань менеджера з складської логістики або просто логіста. Робота з програмним модулем заснована на додаванні та редагуванні у БД інформації відносно контрольних пунктів (торговельних точок), яких в одного покупця може бути необмежена кількість, і на які буде розвозитися товар. На підставі програмної реалізації обраного алгоритму оптимального планування маршруту вантажоперевезень, відбувається промальовування на карті маршруту за самим коротшим шляхом між контрольними точками (підприємствами). Після чого менеджер, який працює із програмним модулем може вивести звіт про маршрут, а також звіт про подорожній лист.

2.2 Основні поняття та позначення, що прийняті в програмному модулі

Для роботи з програмним модулем, користувачеві необхідно володіти спеціальною термінологією, основні поняття якої наведені нижче.

Конфігурація – визначає які дані можна зберігати в БД 1С і яким чином їх обробляти. Отже конфігурація – це словник бази даних.

Контрагенти – це загальне поняття, куди включені постачальники, покупці, організації і приватні особи.

Маршрут – це шлях проходження автомобіля, що враховує напрямок руху щодо географічних орієнтирів або координат, із вказівкою початкової та кінцевої точок.

Трасування маршруту – процес визначення маршруту проходження, дані для якого беруться з бази даних.

Кластер – об'єднання декількох однорідних елементів, яке може розглядатися як самостійна одиниця, що володіє певними властивостями. У цьому випадку в якості кластерів виступають – маркери.

Завантажувальний лист – звіт, що представляє собою, перелік товарів упорядкованих для навантаження у зворотному порядку їх доставки, для прискорення процесу навантаження, доставки і розвантаження товарів за заданим маршрутом.

Оптимальний маршрут – маршрут, за яким можливо доставити логістичний об'єкт, у найкоротший термін (або передбачені строки) з мінімальними витратами.

Транспортування – дія, яка полягає в переміщенні продукції транспортним засобом за певною технологією в ланцюзі поставок і складається з логістичних операцій і функцій.

Транспортна експедиція – діяльність, яка пов'язана з наданням послуг відправникам вантажу і вантажоодержувачам (клієнтам) та організацією доставки вантажів яким-небудь видом транспорту.

2.3 Проектування діаграми варіантів використання

Візуальне моделювання в UML можна уявити, як певний процес порівневого спуску від найбільш загальної і абстрактної концептуальної моделі вихідної системи до логічної, а потім і до фізичної моделі відповідної програмної системи [13]. Для досягнення цих цілей спочатку будується модель у формі, так званої діаграми варіантів використання (ДВВ) (use case diagram), яка описує функціональне призначення системи або, іншими словами, те, що система буде робити в процесі свого функціонування. ДВВ є вихідним концептуальним поданням чи концептуальною моделлю системи в процесі її розробки і впровадження. Розробка діаграми варіантів використання переслідує наступні завдання:

- визначити спільні кордони та контекст модельованої предметної області на початкових етапах проектування системи;
- сформулювати загальні вимоги до функціональної поведінки проектованої системи;
- розробити вихідну концептуальну модель системи для її подальшої деталізації у формі логічних і фізичних моделей;
- підготувати вихідну документацію для взаємодії розробників системи з її замовниками і користувачами.

Суть даної діаграми полягає в наступному: проектована система представляється у вигляді безлічі сутностей або акторів, які взаємодіють з системою за допомогою, так званих варіантів використання. При цьому актором (actor) або дійовою особою називається будь-яка сутність, що взаємодіє з системою ззовні. Це може бути людина, технічний пристрій, програма або будь-яка інша система, яка може служити джерелом впливу на модельовану систему так, як визначить сам розробник. У свою чергу, варіант використання (use case) служить для опису сервісів, які система надає акторові. Кожен варіант використання визначає певний набір дій, який управляє системою при діалозі з актором. Відносинами даного графа можуть бути тільки деякі фіксовані типи взаємозв'язків між акторами і варіантами використання, які в сукупності описують сервіси або функціональні вимоги до системи, що моделюється.

1С: Підприємство – багатофункціональна система. На діаграмі варіантів використання (рис. 2.1) розглядається приклад впровадження розробленої надбудови у конфігурацію 1С Бухгалтерія для України.

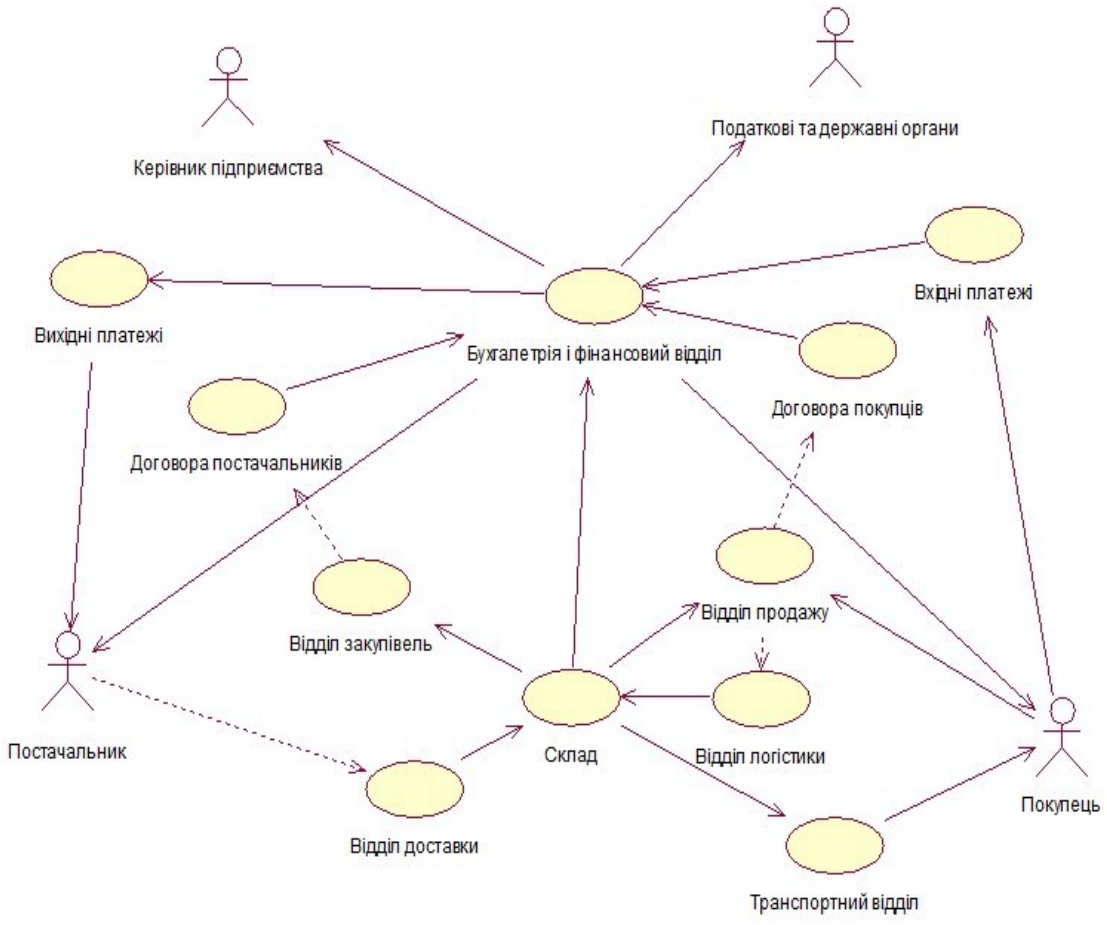


Рисунок 2.1 – Діаграми варіантів використання

«Відділ закупівель» компанії робить замовлення на поставку товарів «Постачальника». «Постачальник» відвантажує товар на «Склад» підприємства через «Відділ доставки». При отриманні товарів склад заповнює документ: «Надходження товарів і послуг», відпрацьовує інформацію для бухгалтерії підприємства і ініціює можливість оплати постачальникам за отриманий товар, згідно за договором на поставку товарів.

За замовленням клієнта на покупку товарів у «Відділі продажів» компанії, постачальник відвантажує товар на «Склад» підприємства через «Відділ доставки». Склад отримує рознарядку на товар, який необхідно відвантажити «Покупцю», згідно документу: «Реалізація товарів і послуг», який заповнили в відділі продажів, відпрацьовує інформацію для бухгалтерії підприємства і ініціює можливість оплати «Постачальнику» за отриманий товар, згідно з договором на поставку товарів. Під час проведення документів цього типу накопичується інформація для «Відділу логістики» про товарну номенклатуру, порядок завантаження цієї номенклатури і маршрут розвезення її «Покупцям», яка після навантаження автотранспорту друкується складським працівником і разом з товарно-транспортними документами передається водієві транспортного засобу.

Решта документів з обов'язкового пакета для передачі покупцеві обробляються бухгалтерією на підставі Документа: «Реалізація товарів і послуг». На підставі всього комплексу документів, що використовуються підприємством «Бухгалтерією» створюється податкова звітність для державних органів та фінансова для керівництва підприємства. Так само на підставі матеріального обліку запасів на складі підприємства «Відділом закупівель» проводиться аналіз товарних залишків для замовлення необхідних товарів «Постачальникам».

2.4 Проектування діаграми послідовності

В UML взаємодія елементів розглядається в інформаційному аспекті їх комунікації, тобто взаємодіючі об'єкти обмінюються між собою деякою інформацією [14]. При цьому інформація приймає форму закінчених повідомлень. Іншими словами, хоча повідомлення і має інформаційний зміст, воно набуває додаткову властивість надавати направлений вплив на свого одержувача. Для моделювання взаємодії об'єктів у мові UML використовуються відповідні діаграми взаємодії. На діаграмі послідовності

зображаються виключно ті об'єкти, які безпосередньо беруть участь у взаємодії і не показуються можливі статичні асоціації з іншими об'єктами.

На рис. 2.2 представлена діаграма послідовності для конфігурації, що розроблюється.

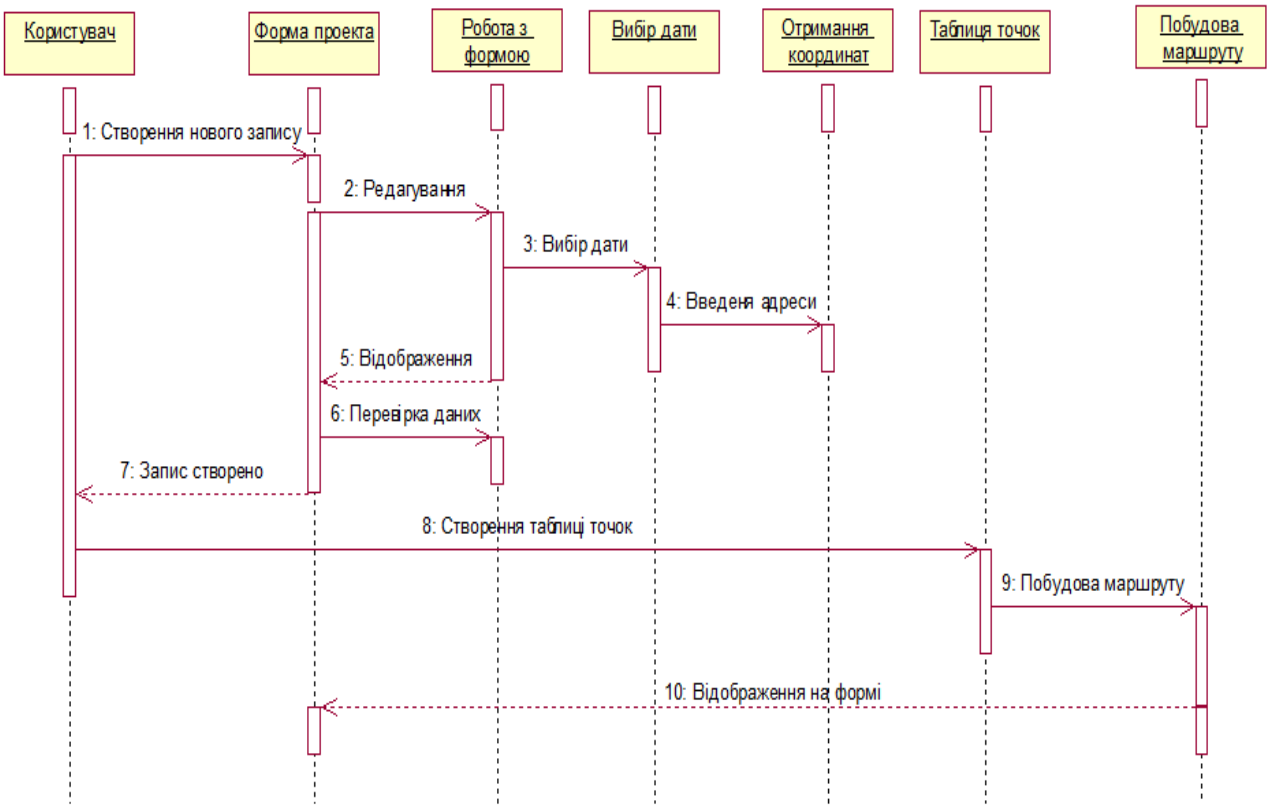


Рисунок 2.2 – Діаграма послідовності

«Користувач» у розробленому модулі створює новий запис, який розташовується на «Формі проекту». Після чого перевіряє свої записи, відбувається «Робота з формою», в цей момент користувач аналізує історію роботи з покупцем. Якщо покупець з якихось причин занесений в «чорний список» покупців, то на цьому робота над замовленням припиняється. Якщо з покупцем підприємство ще не мало відносин і він не занесений до довідника «Контрагенти», то менеджер вводить новий запис («Картку») до довідника. У регістр відомостей «Контактна інформація» – вводиться інформація про адреси всіх торгових точок покупця куди може відвантажуватися товар. Так само в цей регістр можна при допомозі створеної обробки занести координати торгівельних точок. Менеджер в формі проекту створює Документ: «реалізація товарів і послуг», куди заносить контрагента, адреси доставки, товарну номенклатуру (все це вибирається з відповідних довідників і регістрів), а так

само кількість товару, що відвантажується. Також вибирнає дату, та після введеня адреси – отримує координати точок. Після того, як точки сформувалися відбувається їх оптимизація методом Кларку-Райта, після чого будується маршрут. В кінці робочого дня на підставі введеної інформації проводиться завантаження автотранспорту на складі, генерується звіт «завантажувальний лист», «трасування маршруту», «роздрук маршруту» та передача його разом з необхідним пакетом документів водієві – експедитору.

2.5 Проектування діаграми класів

Діаграма класів [15] – статичне представлення структури моделі. Відображає статичні (декларативні) елементи, такі як: класи, типи даних, їх зміст та відношення.

У додатку А наведена UML-діаграма класів для програмного модулю. Клас «Контрагенты» наслідується від класу «Справочники» і містить метод «Проверка уникальности кода», що призначений для запобігання вводу оператором однакових покупців, чи постачальників. Контроль здійснюється за кодом ЄДРПОУ (Єдиний Державний Реєстр Підприємст Організацій України). Клас «Регисты сведений контактной информации», що наслідується від класу «Регистры сведений», містить поле «Объект», через яке кожний запис регістра зв’язаний з довідником «Контрагенты» в реляційному відношенні один до багатьох, де одному запису довідника відноситься багато записів регістру, в яких вказується інформація з адресами доставки тому чи іншому контрагенту.

Клас «Обработка карты», який спадкується від стандартного класу 1С «Обработки», працює у двох режимах: визначення географічних координат за введеною адресою через виклик методів Google Maps Api та побудова оптимального маршруту доставки товарів покупцям. В класі «Обработка карта» викликається «Макет Google» за допомогою Javascript.

Клас «Управление логистикой» є підкласом класу «Общие модули» та містить наступні методи:

- «Алгоритм Кларка-Райта»: в цей метод передаються константи широти та довготи складу, таблиця доступного автотранспорту для перевезення вантажу. На підставі цих даних будується матриця відстаней за допомогою виклику методу «Создать матрицу».
- «Расстояние»: визначає відстань між складом та пунктом доставки;
- «Построение маршрута» – метод визначає максимальні виграші відстані для кожного пункту;

- «Вывести маршрутные листы» – використовується для виведення на друк маршрутних листів.

Під час заповнення користувачем довідника адреси доставки, вона передається в GoogleAPI, який повертає координати об’єкта, за якими він відображається на карті. Після роботи алгоритму Кларка-Райта будуть отримані маршрути доставки, які також за допомогою GoogleAPI з'являться на карті.

2.6 Розрахунок економічної ефективності

На середньому підпрємстві працює три логіста, завдяки даній програмі підприємство може заощадити грощі на двох логістах, так як програма сама будує найоптимальніший маршрут, крім того заощаджується час на побудову маршрута логістом.

Якщо програму продавати серійно її вартість состовлятиме 5000грн, якщо індивідуально для тій чи іншій фірми – 15000грн. Розрахуємо різницю затрат підприємства за даною програмою та одним логістом, та без програми і трьома логістами. По даним work.ua – середня зарплата логіста становить 8000грн/міс. [16]

Розрахуємо економічну ефективність для цього порахуємо скільки підприємство витрачає на сьогодення ща формулою (2.1).

$$В=(Зпл+налог)*чол.. \quad (2.1)$$

Де:

В – витрати,

Зпл – зарплата логіста, налог – 22%,

чол – кількість логістів на підприємстві.

Далі рахуємо, скільки підприємство буде витрачати, якщо придбає серійну конфігурацію та продовже працювати з одним логістом за формулою (2.2):

$$В = (Зпл+налог)+Пс. \quad (2.2)$$

Де:

Пс – вартість програми серійного випуску.

Також розрахуємо, витрати, якщо купувати програми індивідуально для певного підприємства (2.3):

$$В = (Зпл+налог)+Пінд. \quad (2.3)$$

Де:

Пінд – вартість програми індивідуального випуску.

Після розрахунків отримуємо наступну таблицю значень (табл. 2.1).

Таблиця 2.1 – Витрати підприємства

місяць	Всього(грн) з логістами	Всього (грн) з програмою серія	Всього (грн) з програмою не серія
1	29280	15176	25176
2	58560	25352	35352
3	87840	35528	45528
4	117120	45704	55704
5	146400	55880	65880
6	175680	66056	76056
7	204960	76232	86232
8	234240	86408	96408
9	263520	96584	106584
10	292800	106760	116760
11	322080	116936	126936
12	351360	127112	137112

Як ми бачимо з таблиці значень окупність програми відбувається вже у перший місяць її використання, на рік підприємство зможе заощадити більш ніж 2.5 раза свої витрати, крім того зберігається багато часу, в наслідок чого є можливість зробити робітникам більше роботи.

Для наочності побудуємо діаграму (рис. 2.3).

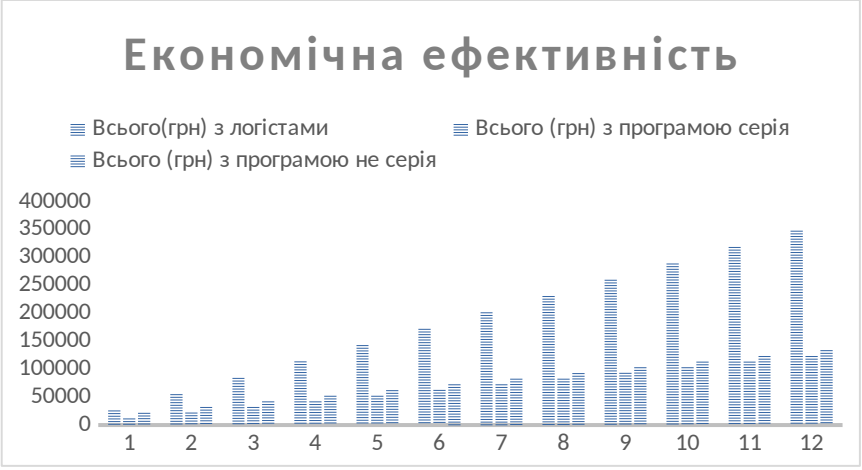


Рисунок 2.3 – Діаграма економічної ефективності

На діаграмі відображено скільки тратить підприємство грошей у тому чи іншому випадку. Сині стовбці – це скільки підприємство витрачає без даної програми, роботаючи з трьома логістами. Красні стовбці – це кошти котрі підприємство буде витрачати з одним логістом та даною програмою при покупці її в серійному вигоотовлені, зелені стовбці – це кошти котрі буде витрачати підприємство при індивідульному впроваджені.

Крім того можливо знизити зарплату директору складу, так як йому більше не має необхідності складати листи загрузки автотранспорту – програма це зробить за нього.

2.7 Висновки до розділу

У даному розділі дипломної роботи змодельована архітектура (проектний «каркас») програмного засобу проектування мережевих графіків організації виробництва та принципу роботи програми, а саме спроектовано:

- діаграму варіантів використання із вкладеною діаграмою-пакетом;
- діаграму послідовності;
- діаграму станів;
- діаграму класів;
- діаграму компонентів.

Також було розраховано економічну ефективність від впровадженної програми.

3 ПРОГРАМНО-МЕТОДИЧНИЙ КОМПЛЕКС

3.1 Призначення програмного засобу

Програмний засіб (ПЗ), що створюється у поданій дипломній роботі, призначено для оптимизації точок розвозки товарів автотранспортом. Для роботи з ним, досить базових знань в галузі логістики та знань звичайного середньо-статистичного адміністратора. Процес створення оптимизованного маршруту досить складний, але для кінцевого користувача вся роботоа ведеться в одній розробленній зручній форм,і тому робота з впровадженим модулем та його освоєння займе набагато часу.

Методика роботи ПЗ заснована на додавані, редагуванні параметрів товару, адрес підприємств, після чого нові параметри оптимизуються згідно з алгоритмом Кларка-Райта. Будь-яка зміна параметрів проводиться в программі: «1С-Предприятие» під правами користувача, тобто в режимі: «Предприятие».

3.2 Розроблення програмного модулю

Перед розробкою програмного модуля був проведений аналіз системи розробки, яка пропонує для розробки різні об'єкти системи.

У зв'язку з тим, що в ході розробки знадобляться функції і процедури до яких будуть відбуватися звернення з різних об'єктів системи було створено спільний модуль «Додаткові функції» в якому будуть розміщуватися процедури і функції призначені для роботи з загальними об'єктами бази даних і в разі ісползованія комерційної версії звернення до GoogleApi функції для отримання ключа гугл GoogleApi.

Так само необхідні будуть процедури і функції, які оптимізують маршрут по методу «Кларка Райта» і процедури звернення до функцій GoogleApi їх доцільно виділити в окремий загальний модуль «Метод Кларка Райта».

Крім цього в базу даних необхідно буде додати елементи зберігання адрес торгівельних точок контрагентів компанії. При цьому кожному контрагенту може відповідати необмежену колічетво торгових точок, так як клієнтами компанії можуть бути мережі роздрібних торгових підприємств. В ході аналізу системи розробки було прийнято рішення використовувати для зберігання адрес торгових точок довідник, підпорядкований довіднику «Контрагенти». Це при подальшій розробці спростить звернення до довідника «Адреси доставки»,

а так само відбори адрес торгових точок, які належать контрагенту з яким в даний момент працює менеджер.

Для зручності роботи менеджера з довідником адрес він повинен мати три форми.

Перша форма – це форма списку. Очевидно в цій формі менеджеру буде зручно оперативно подивитися на карті місце розташування торгової точки, тому екранна форма складається з двох «областей» у верхній області екранна форма містить таблицю списку адрес контрагента, а в нижній – вікно браузера з картою GoogleMaps.

Крім цієї форми менеджеру по роботі з контрагентами знадобиться екранна форма, в якій можливо буде редагувати дані адреси і назви торгової точки, а так само робити запит до GoogleMaps за допомогою скрипта GoogleMapsApi для визначення географічних координат торгової точки і записи їх в базу даних.

Друга форма – буде необхідна для вибору необхідного адреси з вже відібраних адрес поточного контрагента. Ця форма вибору елемента довідника.

Для роботи менеджера логіста зручно розробити одну форму в якій менеджер буде виконувати всі необхідні в його роботі дії. Для цього найбільш підходить об'єкт конфігурації «Обробка». Кількість операцій, які користувач буде виконувати на екранній формі – досить велике, тому форму доцільно було розбити на розділи, для чого був використаний екранний елемент – панель зі сторінками. На першій сторінці розташована таблиця з точками доставки товарів на день розвезення і кнопки формування цієї таблиці та кнопка формування оптимізованого маршруту. На другій закладці розташована таблиця з сформованими маршрутами і оглядачем, на якому відображається маршрут, обраний в таблиці маршрутів. І третя сторінка призначена для формування висновку друкованих форм, як на екран, так і на принтер.

Для зберігання java скриптів GoogleApi найдоцільніше використовувати «Макет», а так, як звернення до скрипту буде відбуватися з різних об'єктів конфігурації, то макет розташували в мекетах конфігурації. При розроблені програмного модулю були розроблені процедури, перелік котрих вказан у розділі 3.2.4. Розглянемо найважніші елементи, котрі були розроблені.

Спочатку необхідно було побудувати розрахункову митрицю відстаней та кілометрового виграшу, описаня алгоритму вказано в пункті 1.6, код представлен в лістингу в процедурі: «Алгоритм Кларка-Райта». В результаті вдалося сформувати матрицю між вказаними нами пунктами. Для того, щоб

переконатися в тому, що матриця будується правильно необхідно поставити крапку в наступному місці (рис. 3.1).

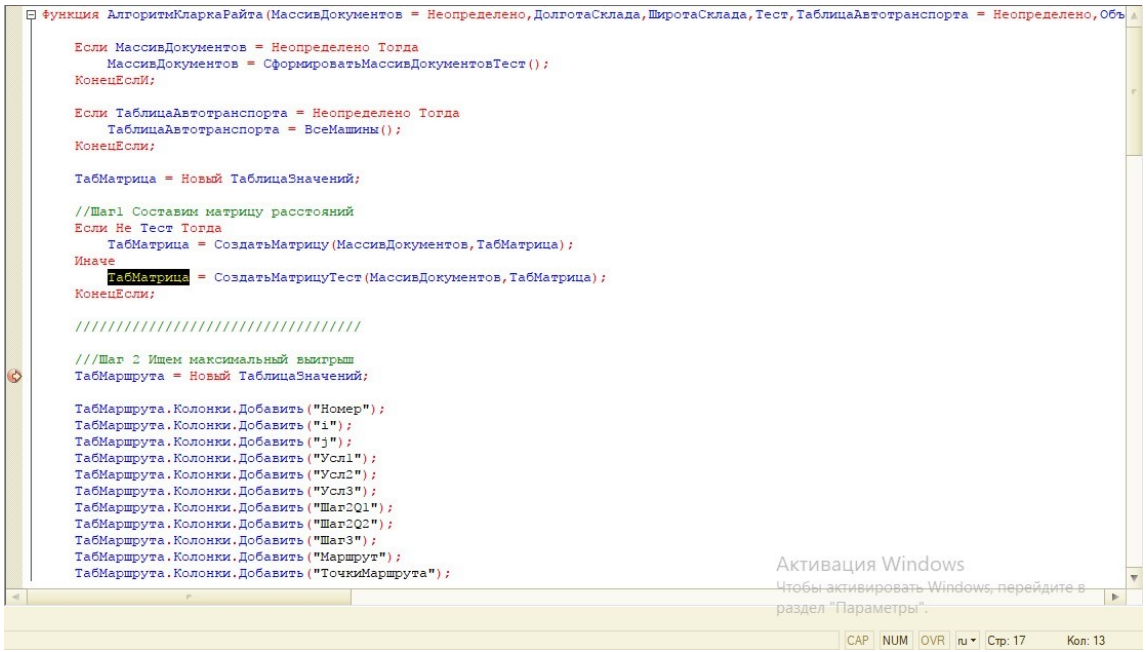


Рисунок 3.1 – Зупинка програми, для перевірки роботи методу

Після того, як програма запуститься, необхідно натискнути на панелі задач на іконку: «Вычислить выражение» (рис. 3.2)

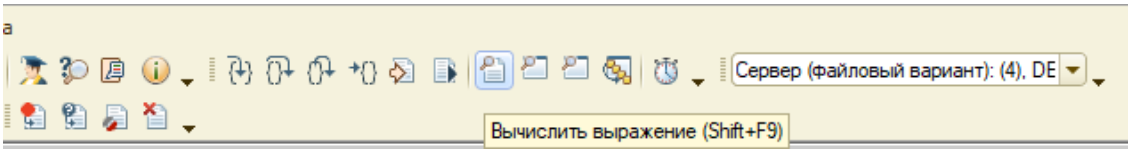


Рисунок 3.2 – Іконка «Вычислить выражение»

В результаті відкривається вікно з значеням, котре нам необхідно натиснути на «ТабМатрица» ПКМ та вибираємо: «Показать значения в отдельном окне» (рис. 3.3)

Перед нами відкрилася таблиця значень та ми можемо перевірити їх вірність, перерахувавши все самостійно (рис 3.4) згідно з алгоритмом Кларка-Райта, опис котрого знаходиться в пункті [1.6]. Значення в таблиці – це кілометрові виграші між точками.

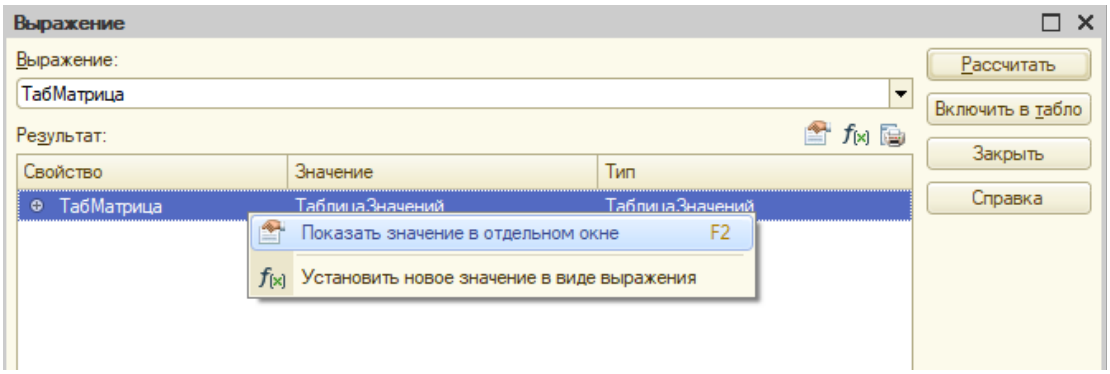


Рисунок 3.3 – Перегляд значення «ТабМатрица»

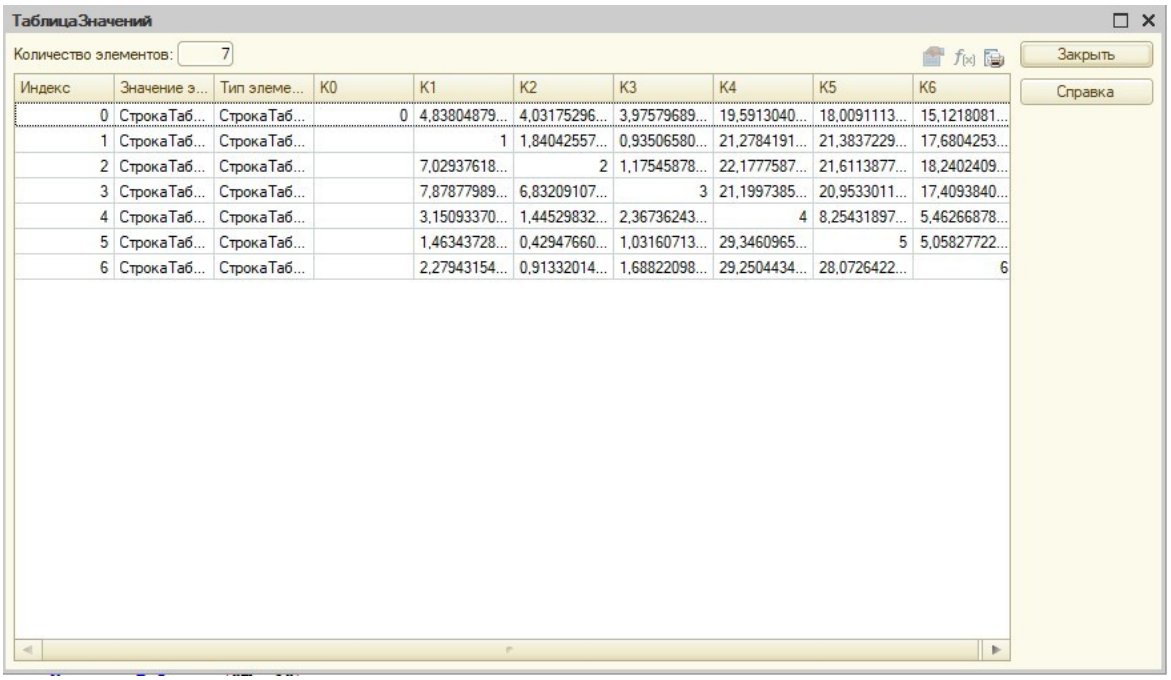


Рисунок 3.4 – Матриця кілометрових виграшів

Після того, як ми отримали координати точок з GoogleMapsApi необхідно було розрахувати відстань між цими точками. Це вдалося реалізувати завдяки сферичній теоремі косинусів.

$$\Delta\sigma = arcos\{\sin\varnothing_1 \sin\varnothing_2 + \cos\varnothing_1 \cos\varnothing_2 \cos\Delta\alpha\}. \qquad (3.1)$$

Де:

$\varnothing_1, \varnothing_2$ – Широта та довгота двух точок;

$\Delta\alpha$ – Різниця координат по довготі;

$\Delta\sigma$ – Углова різниця.

Для перекладу кутового відстані в метричний, потрібно кутову різницю помножити на радіус Землі (6372795 метрів), одиниці кінцевого відстані дорівнюватимуть одиницям, в яких виражений радіус (в даному випадку - метри).

В програмі це виглядає наступним чином:

```
d = acos(sin(ПервыйОбъект.X)*sin(ВторойОбъект.X) +
cos(ПервыйОбъект.X)*cos(ВторойОбъект.X)*cos(ПервыйОбъект.Y-
ВторойОбъект.Y));
Возврат d*111.197;
```

Де:

d – углова різниця,

ПервыйОбъект, ВторойОбъект – координати, котрі ми отримуємо з GoogleMaps.

3.2.1 Обмеження прав доступу

Для обмеження прав доступу в 1С необхідно призначити для того чи іншого користувача. Для уникнення втрати даних і заборони користувачам змінювати текст програми відкритого коду, було вирішено розмежувати права доступу до системи і зробити дві ролі – «Администратор» і «Менеджер логист» (рис. 3.5). Акаунт користувача «Администратор» захищається паролем, щоб менеджер логіст не зміг увійти з його правами [17].

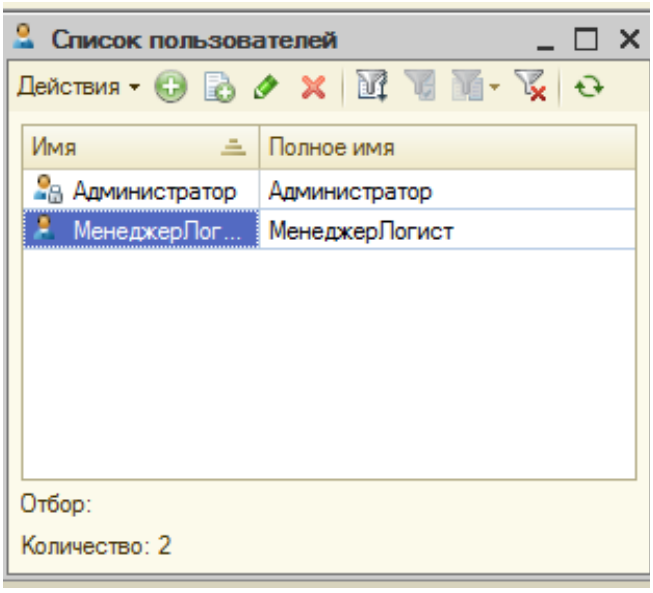


Рисунок 3.5– Вікно розмежування прав доступу

3.2.2 Створеня HTML макету для карт Google Maps

Для роботи зі скриптами Google Map API були створені javascript запроси до баз даних сервісу Google API. Так підключення бібліотек карт було виконано за допомогою наступного скрипта, де вказується посилання до бібліотеки, а також версія API [11] :

<script_type="text/javascript"src="https://maps.googleapis.com/maps/api/js?api=AIzaSyAa7YZpvi09ig92s_BLP2H3QVLTmoqdcQQ&v=3.22.exp&libraries=geometry&sensor=false"></script>

Далі виконується підключення бібліотеки кластерів Google Maps:

<script type="text/javascript" src="http://google-maps_utility_library_v3.22.googlecode.com/svn/trunk/markerclusterer/src/markerclusterer.js"></script>

Також необхідно підключити бібліотеку jquery:

<script_src="http://ajax.googleapis.com/ajax/libs/jquery/1.3.2/jquery.min.js" type="text/javascript"></script>.

Кожну з функцій для роботи з картою Google треба викликати і прописувати окремо, це зроблено для того, щоб програміст мав можливість вибрати лише необхідні йому функції. Нижче, наведена одна з функцій, що була використана – виведення в екстенті карти м. Чорноморськ.

```
function initialize() {
directionsDisplay_=_new_google.maps.DirectionsRenderer();
latlng_=_new_google.maps.LatLng(46.30,_30.65);
                var myOptions = {
                  zoom: 12,
                  center: latlng,
                  mapTypeId: google.maps.MapTypeId.ROADMAP,
                  disableDoubleClickZoom: true,
                  panControl: true,
                  zoomControl: true,
                  mapTypeControl: true,
                  scaleControl: true,
                  streetViewControl: true,
                  overviewMapControl: true
                };
          myMap_=_new_google.maps.Map(document.getElementById("map"),
```

myOptions);

Далі будується карта:

myMap=new_google.maps.Map(document.getElementById("map"),myOptions).google.maps.event.addListener(myMap, 'click', function(event)

І вказуються опції для роботи з менеджером маркерів:

mcOptions = {gridSize: 50, maxZoom: 15};

clusterer = new MarkerClusterer(myMap, markers, mcOptions);

3.2.3 Опис методів класу «Макет Google»

Для роботи з картами Google були розроблені методи, короткий опис яких наведений нижче.

- initialize() – метод ініціалізації карти, виконує додавання маркеру на карту по натисканню кнопки миші;
- addMarker(location) – додає маркер, який з'явився на карті, до масиву, тобто передає його положення (широту та довготу);
- calcRoute(options) – метод містить алгоритм побудови маршруту за контрольними точками;
- Reset() – метод обнуляє маркери на карті;
- FindAdres(Adres) – метод здійснюється пошук адреси за ім'ям;
- ReverseSearchAdres(CoordX, CoordY, Adres) – метод здійснює пошук адреси за координатами;
- addToPointArray(CoordX, CoordY, ID, Text) – метод додає точки до масиву точок. Необхідний для того, щоб для кожного кластеру підкріпилась та чи інша точка;
- drawCluster() – метод здійснює додавання кластера до масиву точок. Після того, як до кожного кластеру додана точка, будується полігон.

3.2.4 Опис процедур, розроблених в «1С: Бухгалтерія для України»

Для того, щоб методи, які розроблені для карт Google почали працювати, їх треба обробити. Для цього в програмі 1С була створена форма для виведення карт. Скориставшись засобом побудови форм, було розроблено табличне поле з колонками «Адреса, Довгота, Широта» куди виводяться значення того чи іншого підприємства, інформація поступає за допомогою javascript запиту розробленого в «Макет Google».

Далі за допомогою процедури в 1С інформація обробляється і переноситься до форми. В колонку «Накладная» виводиться значення накладної товару. Для виводу карти Google було створено поле html документа, яке називається «Эксплорер». На форму обробки також додане поле типу «Дата», яке призначене для створення завантажувального листа і трасування маршруту при натисканні на кнопку «Трассировка», також при натисканні на дану кнопку йде запит до «Макет Google» і всі методи, що були написані передаються до вікна «Эксплорер», виводиться карта, а також будується маршрут. Також було додано кнопку «Погрузочный лист» для виводу звітів – маршрутного листа та завантажувального листа.

Для того, щоб отримати координати була розроблена «Процедура ПолучитьКоординаты()», яка звертається до «Эксплорера» та отримує значення широти та довготи і записує їх у відповідні змінні.

Нижче наводиться короткий опис інших процедур, що були створені в 1С:

- «ИнициализироватьКарту» – створює тимчасовий файл «Карта.html» та вказується шлях до «Макета Google»;
- «ПоискАдреса» – визиває метод Google API «FindAdres» та емулює клік на карті по знайденим координатам для появи маркеру;
- «АдресаПриАктивизацииСтроки(Элемент)» – процедура призначена для пошуку адреси;
- «ПроизвестиГеокодинг_Гугл» – процедура призначена для отримання географічних координат за адресою;
- «ПроложитьМаршрут» – процедура призначена для трасування маршруту між кількома адресами;
- «ПолучитьПараметрыМаршрутаГугл» – визивається із процедури «ПроложитьМаршрут» та повертає параметри маршруту. В даній процедурі за допомогою SQL запиту до БД 1С отримується вибірка усіх адрес за потрібну дату за якими треба доставити товар покупцям. До адресів додається початкова та кінцева точка, якою є склад (маршрут круговий);
- «ПроложитьМаршрут» – передає масив параметрів маршруту у метод «calcRoute» та до серверу Google за допомогою javascript;
- «ПогрузочнийЛист» – призначена для виводу завантажувального листа;
- «СобратьСтроку» – призначен для розбору рядка на складові, розділені символом раделітелем. У метод передається стока і символ роздільник. Метод повертає список значень складових.РазобратьСтроку(Стр,Рзд)

прредназначен для раз бора строки на составляющие, разделенные символом раделителем. В метод передається строка та символ роздільник. Метод повертає список значень составних;

- «ПостроениеМаршрутов» – призначений для складання маршрктов з точок доставки. У метожд передаються таблиця значень точок доставки, порожня таблиця маршрутів, максимально допустимий обсяг завантаження транспортного засобу, первинні документи у вигляді масиву посилань на них. Метод заповнює і повертає таблицю підготовлених маршрутів для розвезення товарів;
- «ПроверитьНаличиеТочекВМаршрутах» – призначений для перевірки входження точок в уже існуючі маршрути і їх статусу (кінцева, початкова точка);
- Метод «ПроверитьНаличиеТочкиВМаршрутах» – призначений для перевірки наявності точки в існуючих маршрутах;
- Метод «СоздатьМатрицу» – призначений для матриці виграшів відстаней. У метод передаються масив з посиланнями документів. Метод повертає заповнену матрицю виграшів у вигляді таблиці значень;
- Функція «ВсеМашины» – робить запит до бази даних і повертає таблицю транспортних засобів;
- Функція «ОпределитьПоКоординатамРасстояние» – призначена для визначення відстаней між точками, заданими географічними координатами;
- Функція «УзнатьКоординатыОбъектаПоСтрокеАдреса» – повертає географічні координати по переданому в функцію адресою;
- Функція «УстановитьКоординатыОбъекта» - записує в базу даних координати згідно Передання в фнкцію Контрагента, його Адрема і структурі координат (широта і довгота).

3.2.5 Справочник адреса доставки

Довідник адреса доставки є підлеглим довідником довідника «Контрагенти». Реляційний зв'язок: «Один багато до багатьох». Кожен елемент довідника «Контрагенти» може бути пов'язаний з безліччю елементів довідника «Адреси доставки». У довіднику реалізовано три екранні форми.

- Форма элемента справочника. У цю форму користувачем вводиться адреса торгової точки контрагента. В формі реалізовано отримання географічних координат від GoogleApi;

- Серверна Процедура «ОтобразитьАдресаДоставки» – містить запит до довідника адреси доставки по поточній посилання і отримує довготу і широту з довідника;
- Клієнтська процедура «ОпределитьКоординаты» – викликає серверну процедуру «ОтобразитьАдресаДоставки» та оновлює відображення даних на формі;
- Серверна процедура «ОбновитьКоординаты» – призначена для запису координат в базу даних;
- «ПроизвестиГеокодингГугл» – У функцію передається адреса, функція повертає структуру географічних координат отриманої адреси.

Група подій змін полів форми призначені для форматування адресного рядка для геокодінга Google.

Екранна форма списку справочника «Адреса доставки». На формі показуються адреси доставки і відображається місцезнаходження адреси на карті вбудованого експлорера. На формі розташована таблиця з адресами доставки. Подія активізація рядка.

- Процедура «СписокПриАктивизацииСтроки» виводить на карту поточний адресу;
- Процедура «ИнициализироватьКарту» – призначена для ініціалізації карти за допомогою звернення до java скрипту;
- Функція «ПолучитьТекстМакета» – отримує і повертає макет java, що знаходиться в загальних макетах;
- Подія «ПриОткрытииФормы» – У момент відкриття викликається функція ініціалізації карти;
- Подія «ЭксплорерПриНажатии» – призначене для отримання координат місця після клацання миші;
- Процедура «ПолучитьКоординаты» – отримує координати встановленої точки;
- Процедура «НайтиАдресНаКарте» – Викликає процедуру пошуку адреси за переданою адресою;
- Процедура «ПоискАдреса» – Показує переданий в процедуру адресу на карті;
- Подія «СписокПриОкончанииРедактирования» – Після закінчення редагування рядка списку оновлює дані і перемальовує карту;
- Екранна форма «ФормаВыбора» – призначена для вибору необхідного адреси з інших форм, наприклад з форми документа «РеализацияТоваровИУслуг»;

– Обработка «Логистика» – це і є робоче місце логіста, де він може оптимізувати маршрути за допомогою методу Кларка Райта, подивитися маршрути на карті, створити вантажний і маршрутні листи;
– Процедура «ЗаполнитьДоставкуНа Сервере» завантажує в таблицю обробки дані з рахунків і документів «РеализацияТоваровУслуг»;
– «СформироватьТаблицуАдресов» – Формує список значень, заповнений адресами точок доставки;
– Процедура «ПоискАдреса» – виводить на карту точок з переданої в процедуру адресою;
– Процедура «СформироватьМаршрутыНаСервере» – Пересилає між клієнтом та сервером таблицю значень з маршрутами у вигляді списку структур рядків;
– Процедура «МаршрутыНаКлиент» – Отримує на клієнті з сервера список значень у вигляді структури рядків і заповнює таблицю значень на клієнті;
– Процедура «ТочкиНаКлиент» – Пересилає між клієнтом і сервером таблицю точок з адресами доставки у вигляді списку структур рядків;
– Процедура «СформироватьМаршруты» – Пересилає між клієнтом і сервером таблицю маршрутів у вигляді списку структур рядків;
– Подія «ЭксплорерПриНажатии» – Отримує координати точки на якій відбулося натискання;
– Подія «ТаблицаМаршрутовПриАктивизацииСтроки» – Виникає при активізації рядка таблиці і отримує впорядкований маршрут в виді точок з відображення точок на карті;
– Событие «ПриОткрытии» – Ініціалізує карту Google;
– Процедура «ОчисткаКарты» – Очищає на карту від точок і промальованих маршрутів;
– Подія «ПриАктивацииМаршрута» – Викликає серверні методи з метою встановлення і промальовування маршруту на карті, що знаходиться у вікні експлорера;
– Функція «ПолучитьКоординатыСклада» – отримує координати власного складу з бази даних;
– Подія «ОбновитьКарту» – Викликає процедури промальовування поточного маршруту на карті в вікні експлорера;
– Процедура «ПодготовитьПараметрыОтчета» – Готує параметри звіту для передачі на сервер до виконання у фоновому режимі;

- Процедури «СформироватьОтчетМаршрутовНаСервере» та «СформироватьОтчетПогрузочныеЛистыНаСервере» – формують табличний документ на формі після виклику в модулі менеджера обробки процедур формування цих документів у фоновому режимі;
- Функція «ЗаданиеВыполнено» Перевіряє чи виконано фонове завдання раз в зазначений час;
- Подія «ОтменитьВыполнениеЗадания» – Перериває виконання побудови звіту;
- Подія «ВычислитьСуммуВыделенныхЯчеек» – Обчислює суму виділених осередків звіту, якщо в них містятся числові значення. Виникає при груповому виділенні одного або декількох полів табличного документа «Результат» на екранній формі обробки;
- Процедура «Подключаемый_РезультатПриАктивизацииОбласти» – Для серверного варіанту бази даних Обчислює суму виділених осередків звіту, якщо в них міститися числові значення. Виникає при груповому виділенні одного або декількох полів табличного документа «Результат» на екранній формі обробки. Підключається подія до загального модулю бухгалтерських звітів;
- Процедура «УправлениеФормой» – Ініціалізує на клієнтській частині програми посилання на об'єкти форми конфігурації;
- Процедура «ОбновитьТекстЗаголовка» – Виводить текст заголовка на форму;
- Процедура «ВыбратьПериодЗавершение» – Встановлює значення Актуальності даних звіту, виведених в табличний документ, в залежності від зміни параметрів, що передаються в модулі формування табличного документа звіту;
- Процедура «РезультатПриАктивизацииОбласти» – Виводить результати при активізації області табличного документа, при цьому відслідковуючи швидкість клієнтського з'єднання. При низькій швидкості час очікування встановлюється 1 секунда, при високій швидкості 0,2 секунди;
- Процедура «ПриСозданииНаСервере» – При створенні на сервері форми обробки встановлює настройки друку за замовчуванням;
- Процедура ПередЗакрытием – Викликає стандартні функції, такі, як очищення кеша і сховища значень при закритті форми обробки;
- Подія «ПриЗакрытии» – Скасовує всі незавершені фонові завдання, закриваємої форми;

- Подія «ПриСохраненииПользовательскихНастроекНаСервере» – Зберігає призначені для користувача настройки форми в сховище налаштувань даного користувача;
- Процедура «ПриЗагрузкеПользовательскихНастроекНаСервере» – Завантажує раніше збережені, якщо такі є, призначене для користувача настройки обробки зі сховища налаштувань користувача;
- Процедура «СформироватьОтчетМаршруты» – Викликає з менеджера об'єкта процедуру формування табличного документа звіту «Маршруты» і виводить, повернутий табличний документ на форму в поле результат;
- Процедура «СформироватьПогрузочныеЛисты» – Викликає з менеджера об'єкта процедуру формування табличного документа звіту «погрузочные листы» і виводить, повернутий табличний документ на форму в поле результат;
- Менеджер Обробки «Логистика» – містить процедури формування відповідних звітів;
- «СформироватьОтчетМаршруты» та «СформироватьОтчетЛист»– при цьому в процедури передається в процедури параметри, необхідні для формування звітів у вигляді структури і унікальний ідентифікатор адреси сховища, через яке буде повернуто табличний документ з результатами звіту.

3.3 Опис роботи з програмою для користувача

Робота з програмним модулем заснована на додаванні і редагуванні бази даних з контрольними пунктами торговими точками, які містять повну адресу, а також географічні координати, одержувані за допомогою звернення до GoogleApi,).

API Google Карти – це абсолютно безкоштовний, програмований, картографічний сервіс, який надається компанією Google. Його можна легко розмістити на своєму сайті і періодично відзначати своє місце положення, щоб клієнти могли легко визначити ваше місце розташування і без зайвого клопоту відшукати офіс.

Сам сервіс представлений у вигляді набору протоколів, за рахунок яких програмісти і веб-розробники можуть як «по цеглинці» збирати різні додатки. Ще зовсім недавно API був тісно пов'язаний з ОС і додатками на робочому столі. Але за останні кілька років тренд настільки виріс, що API став незамінним інструментом у житті багатьох програмістів у різних сферах[18].

Для додавання або редагування адреси підприємства досить у картці контрагента (покупця) перейти на закладку «Контактная информация» і ввести адресу в певному форматі з роздільником «,» або нажавши на кнопку «…» ввести адресу у форму введення для автоматичного форматуванняПри цьому після закінчення введення інформації в будь-який адресне поле форми програмно сформується адресний рядок в необхідному форматі для подальшого використання її геоконгом GoogleApi. Приклад формату: країна, область, місто, вулиця, номер будинку.

Для того щоб розпочати роботу з програмою необхідно запустити «1С-Предприятие», вибрати необхідно конфігурацію, та натиснути на кнопку «1С-Предприятие», вести ім’я та логін. (рис. 3.6)

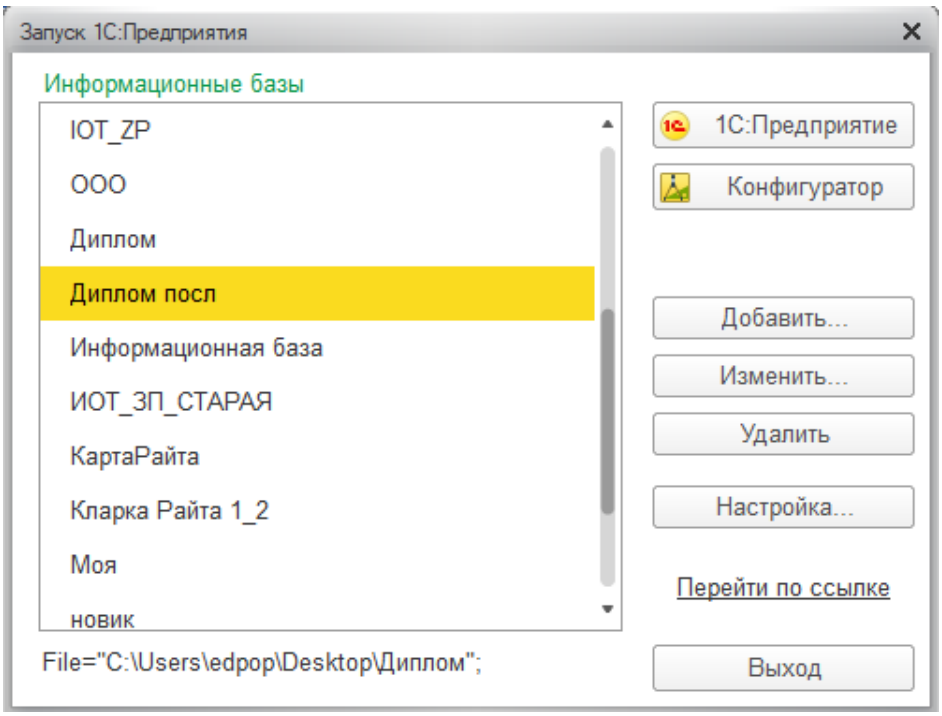


Рисунок 3.6 – Вікно входу до: «1С-Предприятия»

Після того, як увійшли у Конфігурацію, необхідно додати нових контрагентів.

Коли менеджер відділу продажів виписує видаткові документи (видаткову накладну), він вибирає торговельну точку, у яку необхідно доставити товарпри цьому менеджеру програмно виводиться список тргових точок тільки цього контрагента, що полегшує менеджеру роботу, фільтрацією непотрібної інформації, з довідника «Адреси доставки», який є підлеглим довіднику «Контрагенти», тобто відноситься до нього в співвідношенні багато

до одного. Таким чином БД заповнюється видатковими накладними за поточний день. Після закінчення формування замовлень на завтра менеджер служби зі свого робочого місця в програмне запускає обробку натиснувши на робочому столі в розділі продажів посилання Після цього менеджер вказує дату для якої він повинен визначити маршрути доставки і створити навантажувальні листи.По натисканню на кнопку «Трасування маршруту» дані за допомогою запиту SQL з БД передаються на форму юПісля цього менеджер повинен натиснути на кнопкупісля цього програма обробляє оптимізацію маршрутів доставки по методуз урахуванням максимально допустимого обсягу, який може помістити в транспортний засіб. Це можна було б зробити і за допомогою сервісу GoogleMaps, але даний сервіс допускає оптимізацію тільки 23 точок доставки, не враховуючи початкову і кінцеву точки. Тому ми спочатку розбиваємо велику кількість точок по різних маршрутах, використовуючи алгоритм оптимізаціїі тільки потім виробляємо трасування кожного готельного маршруту, а їх за умовою задачі не може бути більше 23, так як максимальний обсяг перевезення самого вантажу Так само за допомогою гугл АПИ можна виробляти трасування з використанням сервісу Коли програмно отримані необхідні маршрути, після вибору необхідного маршруту на закладці форми обробкинатискається кнопкапісля чого на картуиьвіди з картою і відбувається промальовування маршруту за самим коротшим шляхом між контрольними точками (підприємствами). Після чого менеджер, який працює із програмним модулем може вивести звіт про маршрут, а також звіт про завантажувальний лист.Друковані форми будуть сформовані і виведені посторінково відповідно до маршрутами і необхідної завантаженням автотранпорт, згідно вантажним листам маршрутами.

Для прикладу додамо 7 пунктів перевезення товару. Щоб додати нового контрагента необхідно натиснути на кнопку «Создать», заповнити всі поля та записати його у БД (рис.3.7).

Далі натискаємо на вкладку «Адреса», додамо адресу нашого пункту (рис. 3.8) та натискаємо на кнопку «ОК», після чого сформується xml документ для відправки інформації до карт Google Maps.

Потім заходимо до пункту «Адреса доставки». Якщо в нас з’явилася нова адреса, ми можемо також додати її тут (рис.3.9) і вона відобразиться на карті (рис.3.10). Всі дані, котрі вводяться передаються у конфігуратор 1С, а з нього до сервіса GoogleMapsApi.

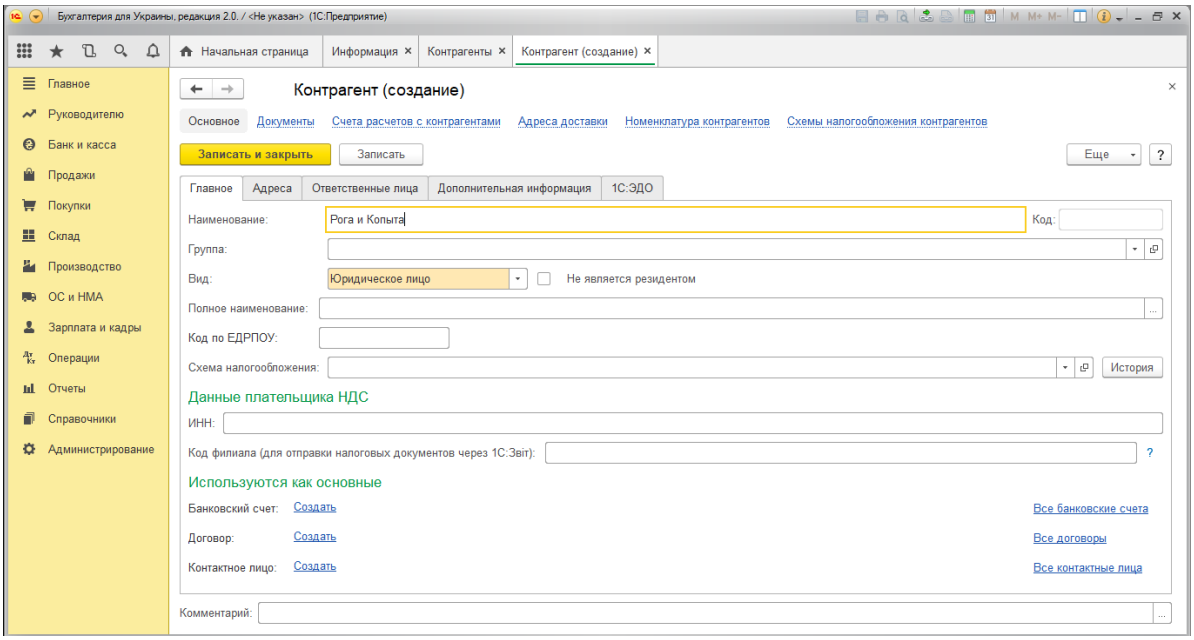


Рисунок 3.7 – Заповнення даних для нового контрагента

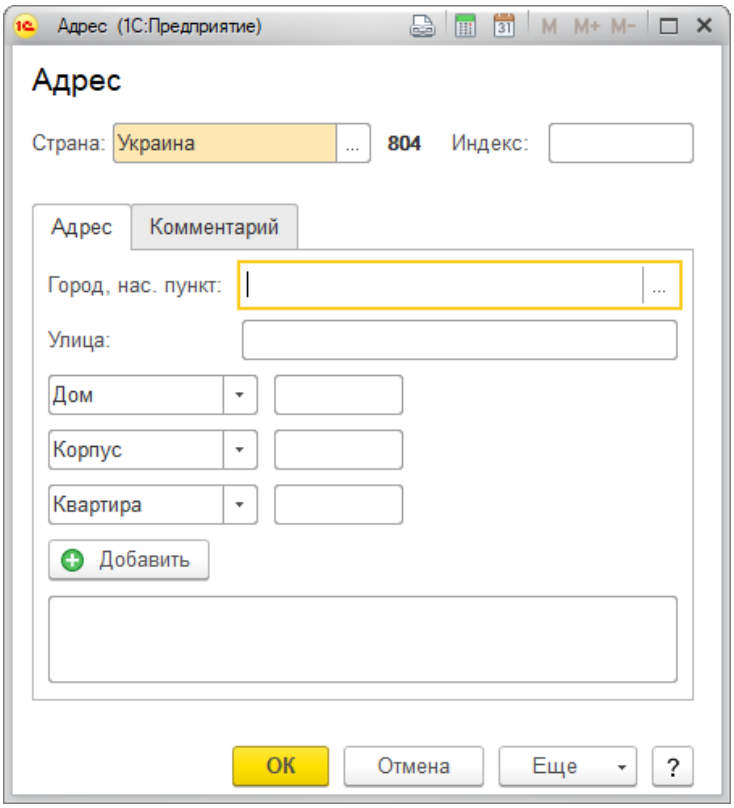


Рисунок 3.8 – Введення адреси підприємства

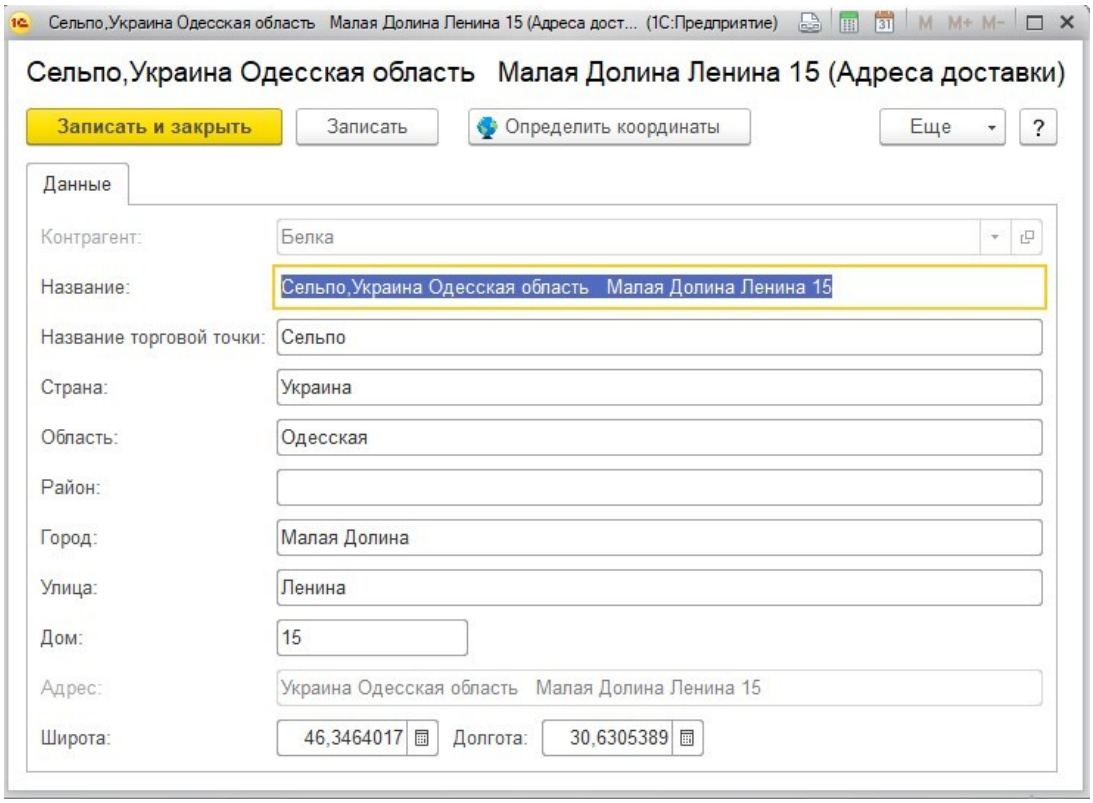


Рисунок 3.9 – Додавання адреси доставки

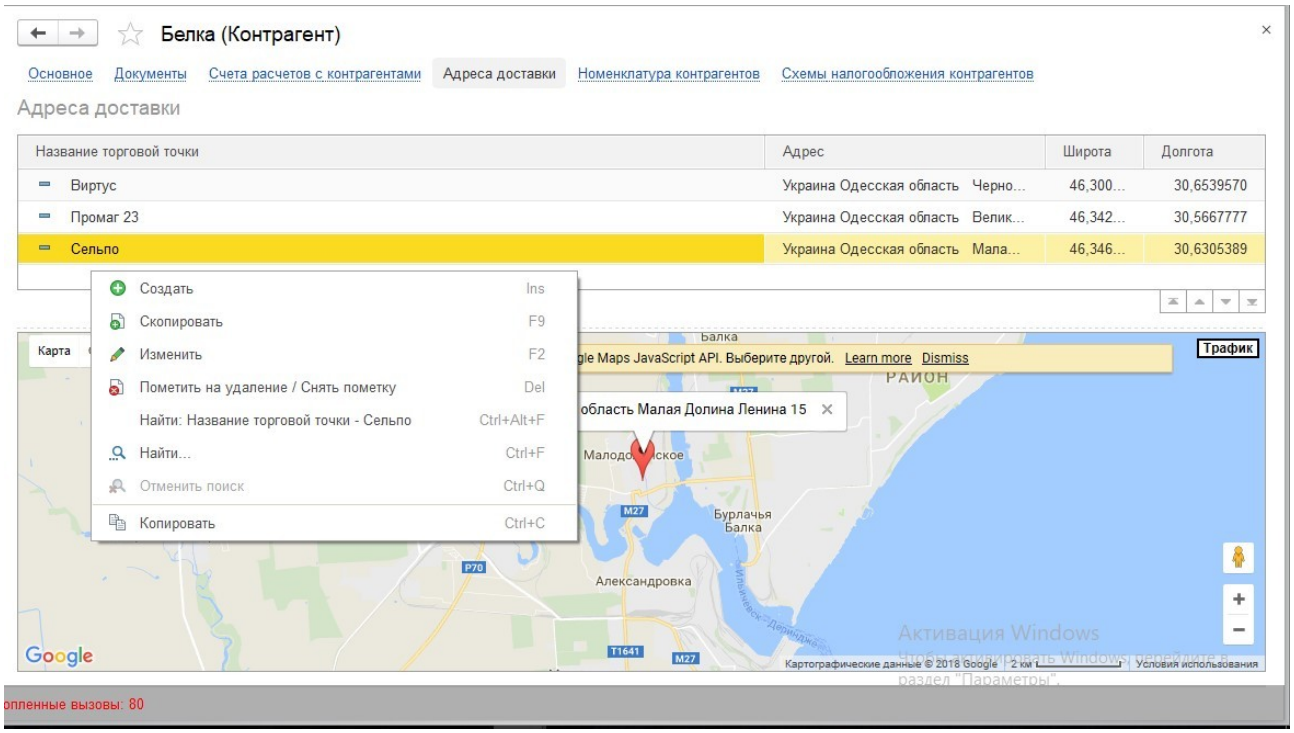


Рисунок 3.10 – Відображення пункту доставки на карті

Тепер необхідно оформити продаж товару. Продаж товарів оформляється в програмі документом Реалізація товарів і послуг. Документ можна провести, тільки якщо є певна кількість товару на складі. В даному вікні вказується список видаткових накладних. Тут вказується дата коли товар був проведений, номер, сума, валюта, контрагент та склад (рис. 3.11).

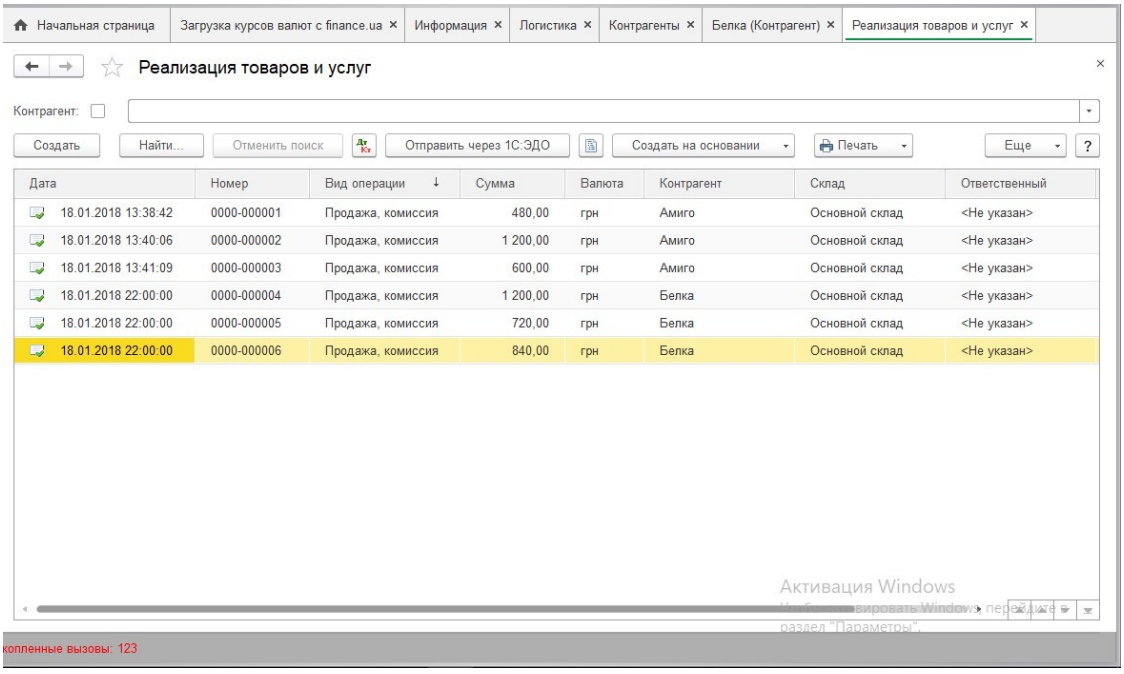


Рисунок 3.11 – Оформлення товарів

Для оформлення продажу нового товару необхідно натискнути на кнопку: «Создать», у вікні вписати необхідну дату, номер, вибрати Контрагента, та договір, склад. Далі натискаємо на вкладку «Товары», натискаємо на кнопку: «Добавить» та вводимо назву товару, кількість, одиниці в котрих вимірюється товар, задаємо його ціну, та отримуємо суму. (рис. 3.12)

Необхідно вибрати місце куди даний товар треба доставити, для цього натискаємо на вкладку «Дополнительно», та вказуємо в випадаючому списку адресу доставки. Далі натискаємо: «Провести и закрыть» (рис. 3.13).

Для того, щоб програма сформувала необхідні маршрути за методом Кларка-Райта, необхідно вийти до головного меню, вибрати пункт «Продажи», в ньому обрати пункт «Логистика» (рис. 3.14).

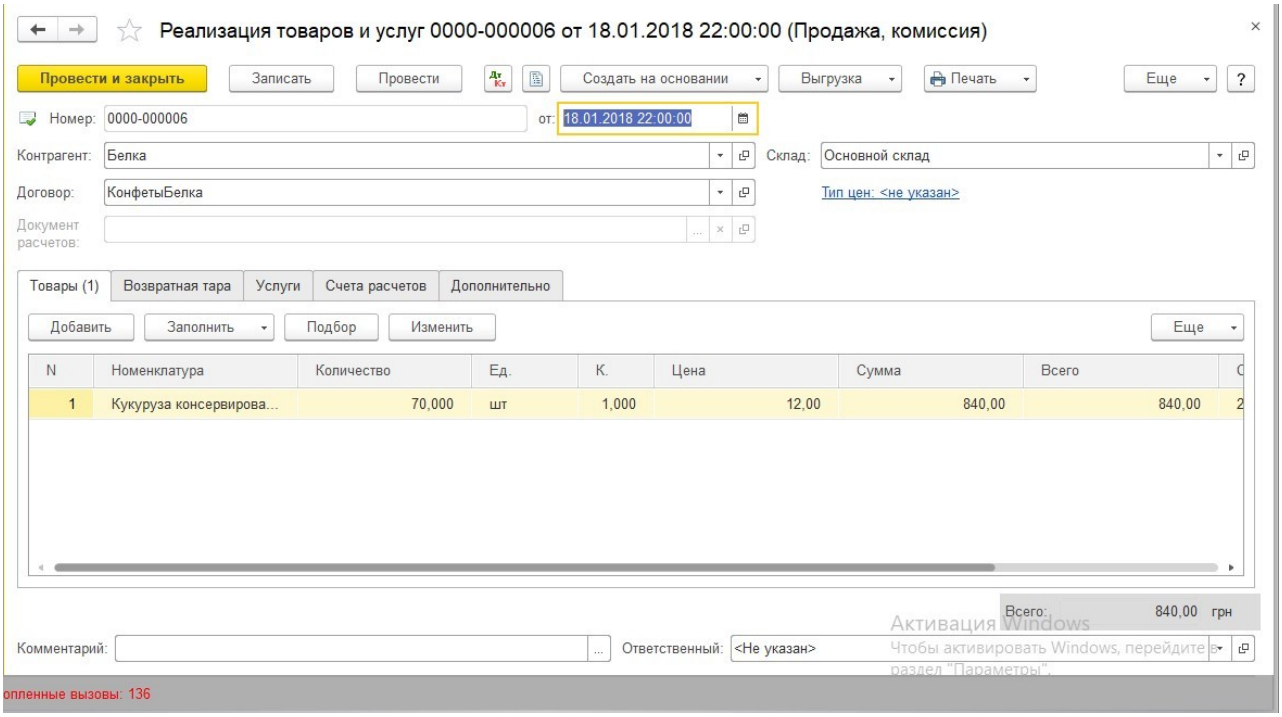


Рисунок 3.12 – Додавання товару в базу

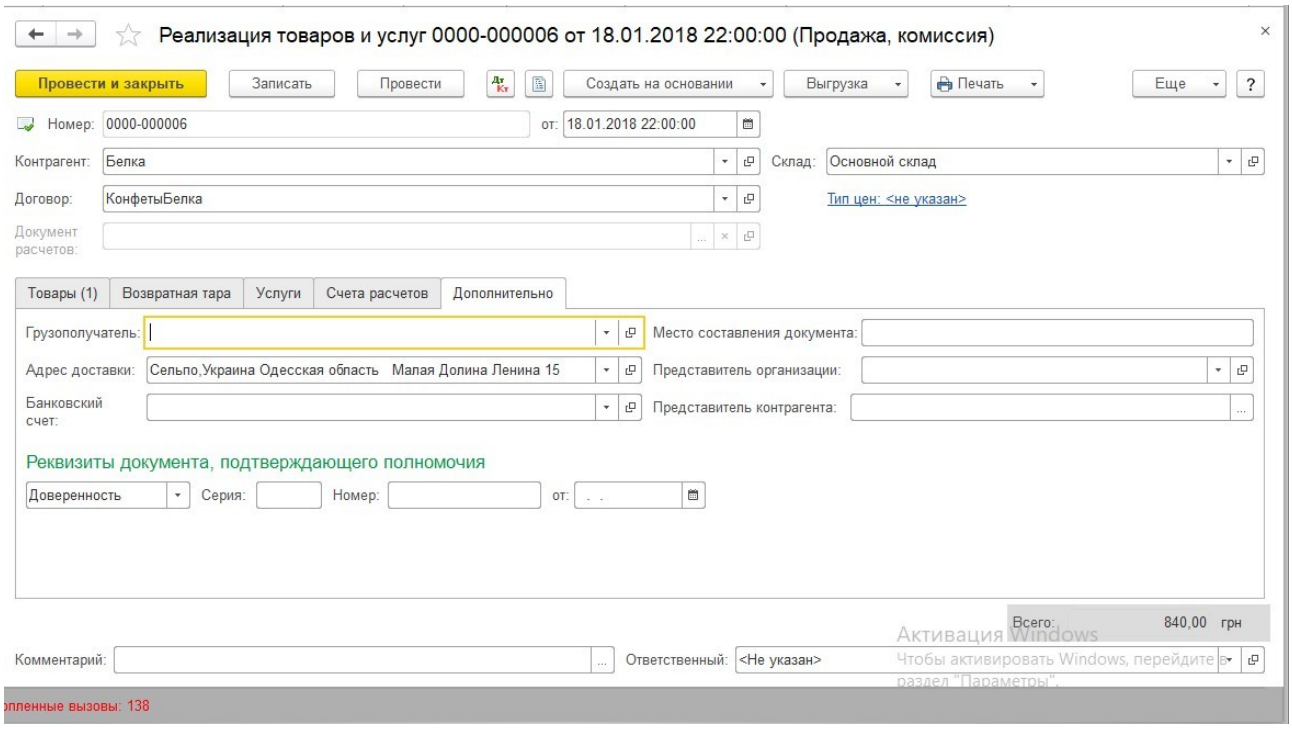


Рисунок 3.13 – Додаємо адресу доставки до вказаного товару

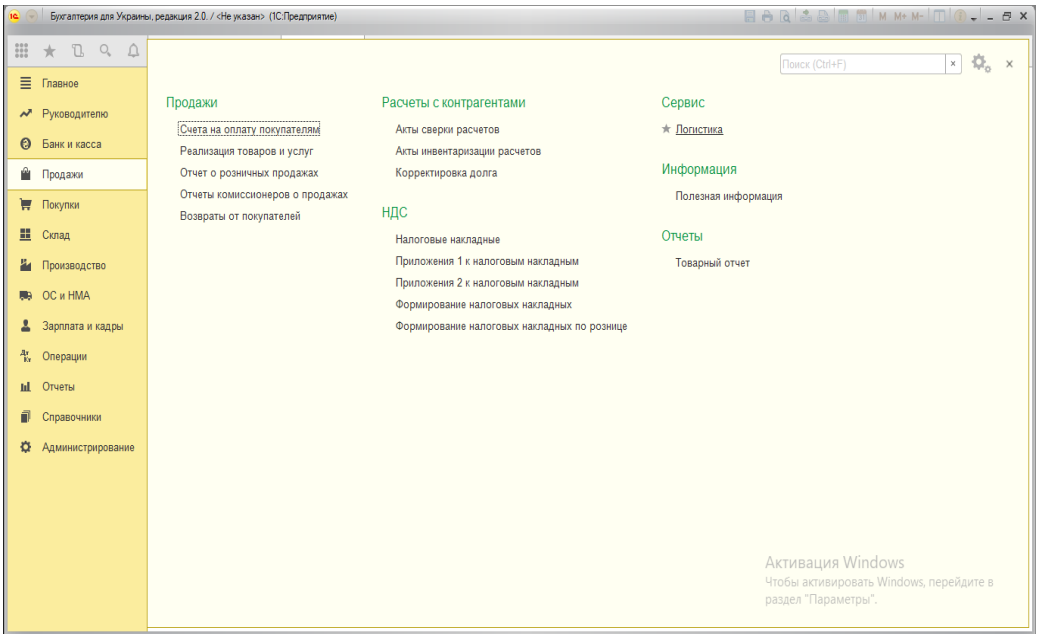


Рисунок 3.14 – Переходимо до розробленого модуля «Логистика»

Після того, як ми обрали пункт «Логістика», перед нами відурівається розроблена нами форма (рис. 3.15). Для того, щоб вивести оптимизовані маршрути за методом Кларка-Райта необхідно спочатку необхідну обрати дату, за яку були проведені товари.

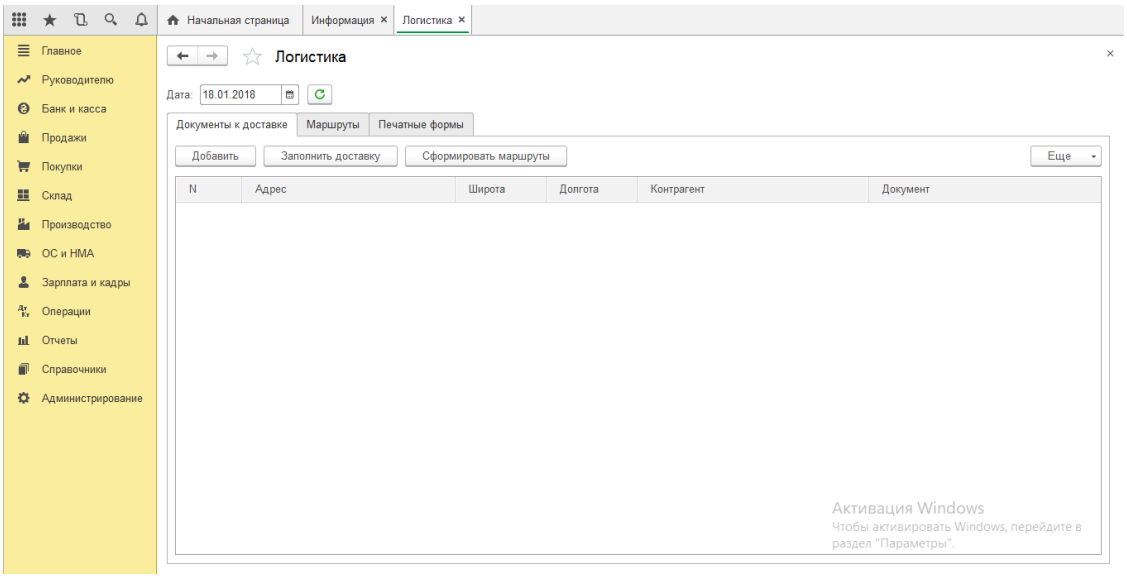


Рисунок 3.15 – Форма розробленого модуля

Після того, як дата обрана необхідно натискнути на кнопку: «Заповнити доставку», в результаті чого формується доставка (рис. 3.16):

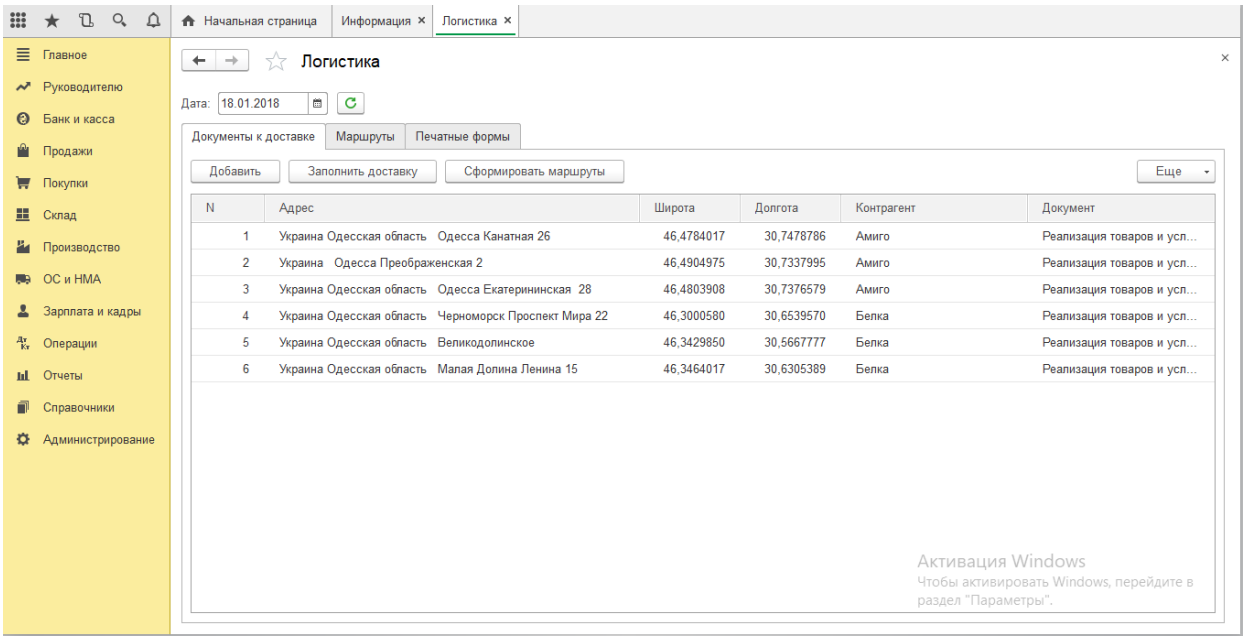


Рисунок 3.16 – Формування доставки

Після того, як доставка сформувалась, необхідно натиснути на кнопку «Сформировать маршрут», перейти на вкладку «Маршруты» та натискнути на кнопку відновити (рис. 3.17)

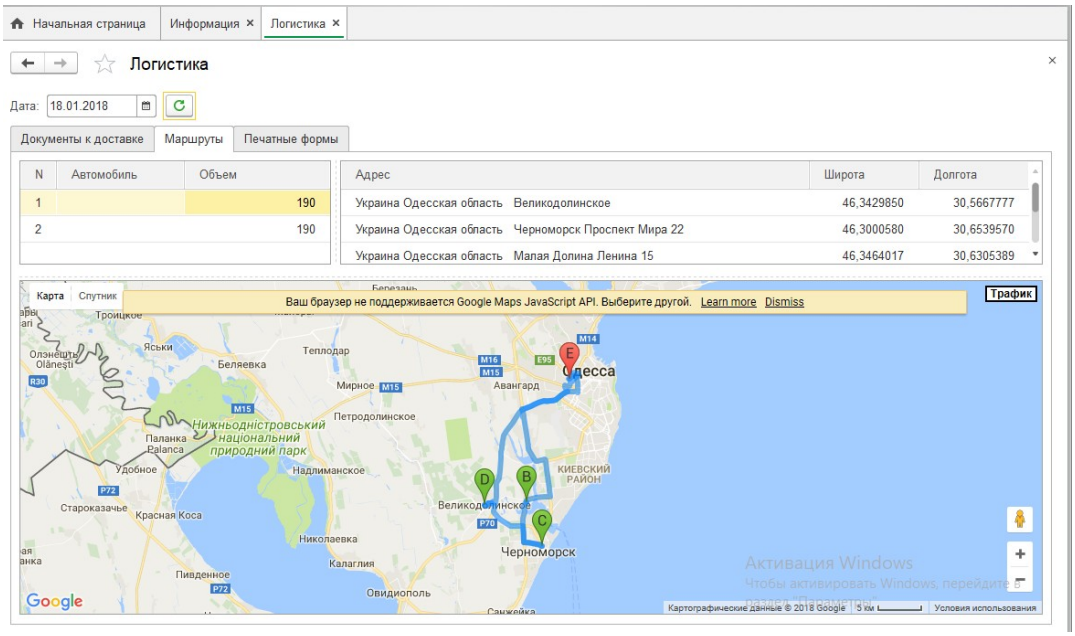


Рисунок 3.17 – Сформований маршрут

На екрані відобразилася карта та сформувався маршрут. Точки оптимизовані за методом Кларка-Райта [1.6]. В таблиці у стовбці «N»

відображається номер маршруту, так як точок було 6, програма сформувала два маршрути, в стовбець «Автомобіль» – передається марка машини, котра буде розвозити товар, до стовбця об’єм передається макисмальний обєм, котрий можливо перевезти за допомогою того чи іншого автотранспорту. До стовбя: «Адрес» – передається адреса точки, у стовбці «Широта», «Довгота» – передаються координати точок згідно з системою координат: WGS84, так як саме цю систему використовує Google Maps.

WGS 84 – всесвітня система геодезичних параметрів Землі 1984 года, в число яких входить система геоцентричних координат. На відміну від локальних систем, є єдиною системою для всієї планети. Попередниками WGS 84 були системи WGS 72, WGS 66 і WGS 60.

WGS 84 визначає координати відносно центру мас Землі, похибка становить менше 2 см. В WGS 84 нульовим меридіаном вважається – Опорний меридіан, що проходить в 5,31 "(~ 100 м) на схід від Гринвічського меридіана. За основу взято еліпсоїд з більшим радіусом – 6 378 137 м (екваторіальний) і меншим – 6 356 752,3142 м (полярний). Практична реалізація ідентична відлікової основі ITRF.

Для того щоб переглянути другий маршрут достатньо натискнути на другий номер та на кнопку відновити, після чого маршрут сформується та відобразиться на екрані(рис.3.18)

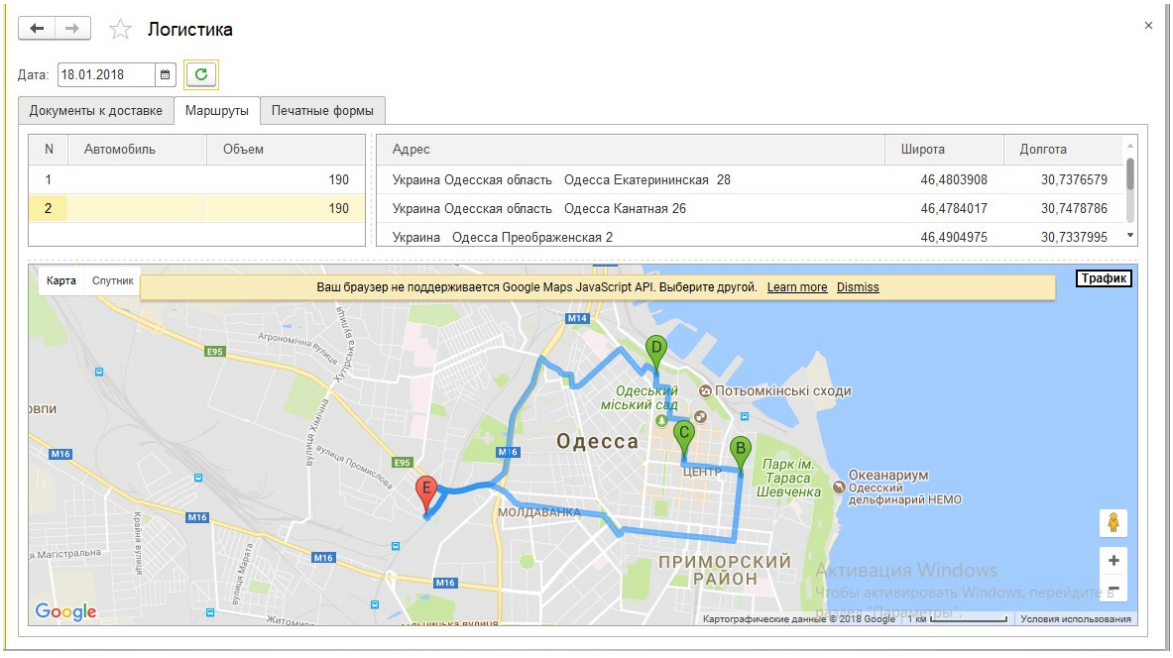


Рисунок 3.18 – Другий побудований машрут

У програмі було реалізовано вивод звіту на форму, для того, щоб отримати звіт з логістики та звіт для загрузки автотранспорту, необхідно натиснути на вкладку «Печатные формы». У відкрившимося вікні необхідно натиснути на кнопку «Сформировать печать маршрутов», на екран виведені всі маршрути, та об’єм котрий необхідно доставити до кожної з точок (рис. 3.19)

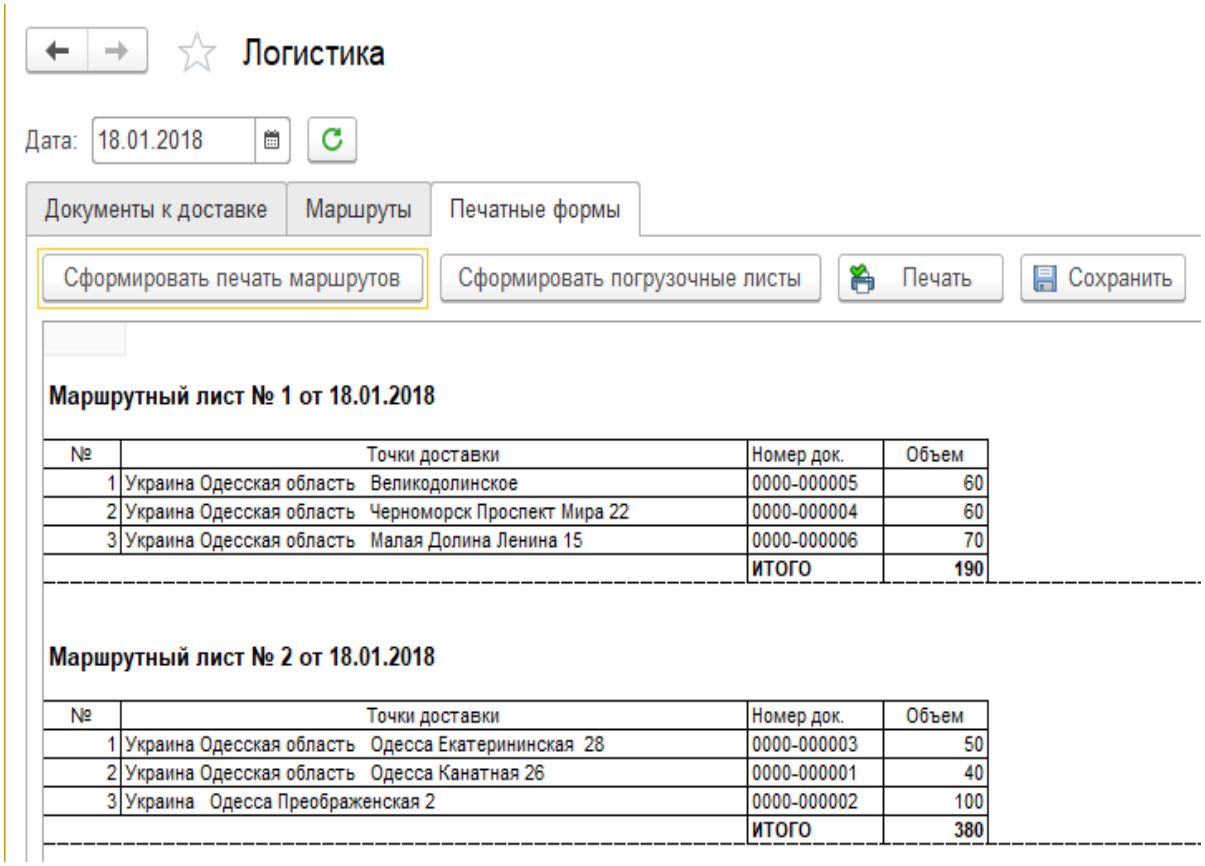


Рисунок 3.19 – Сформовані маршрути

Для того, щоб сформувати лист погрузки необхідно натиснути на «Сформировать погрузочные листы». Після чого формується звіт в котрому вказано як грузити автотрансорт, згідно з пунктами розвозки. Що необхідно доставити пізніше – кладеться першим, а те що в першу чергу – ближче. Також в звіті є ім’я контрагенту, назва документу, його номер, та дата проводки, адреса куди необхідно доставити товар, номер маршруту, найменування товару, та кількість товару (рис. 3.20).

Після чого звіти можно роздрукувати, чи сохранити. Нажавши на відповідні кнопки.

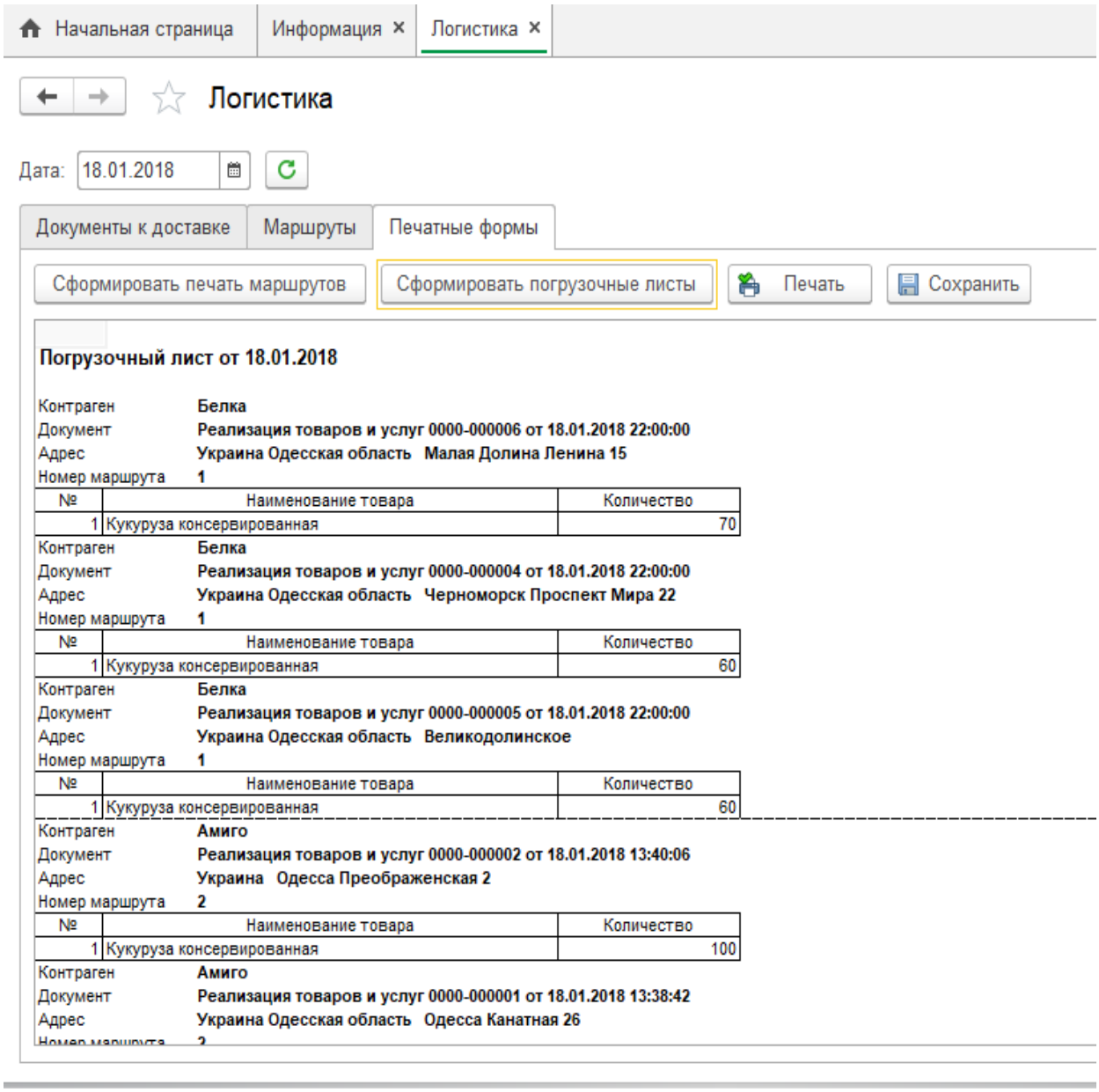


Рисунок 3.20 - Сформовані листи погрузки

3.4 Висновки до розділу

В процесі виконання даного розділу дипломної роботи магістра, було розроблено програмно-методичний комплекс (ПМК) роботи з точками доставки та побудоваю мрашруту, створення якого складалось з наступних етапів:

- Сфера призначення;
- Виділення основних понять та позначень;
- Визначення порядку роботи з ПМК, що складається зі створення нового маршруту, зміни параметрів маршруту, створення зв’язків між товарами, маршрутами за алгоритмом Кларка-Райта, додавання точок товару,

формування листів завантаження автотранспорту згідно з значеннями сформованими в программі та формування звіту з розвозки товарів між точками;

- Генерування таблиць розвозки товарів між пунктами;
- Друк та збереження інформації відображеної у звіті.

Робота з програмаю не потребує спецальних знань та є інтелектуально зрозумілою для кінцевого користувача.

ВИСНОВКИ

В ході виконання роботи була розроблена інтегрована автоматизована система управління логістичними процесами підприємства, яка інтегрується у CRM систему «1С: Підприємство 8х» і дозволяє оптимізувати транспортно-складські логістичні потоки. Система надає картографічну інформацію про логістичні потоки підприємства, розраховує оптимальні послідовності об'їзду пунктів доставки товарів, щоб здійснити перевезення з мінімальним пробігом, а також дозволяє автоматизувати документообіг підприємства.

Для візуального моделювання системи було побудовано діаграми варіантів використання і послідовності. Розроблена UML-діаграма класів програми.

Для виводу тайлових карт був використаний сервіс Google Maps API. Для роботи зі скриптами Google Map API були створені javascript запити до баз даних сервісу Google API і окремий класу «Макет Google» з набором методів, призначених для роботи з картами Google.

Для вирішення завдань маршрутизації перевезень дрібно партійних вантажів був використаний метод Кларка-Райта, який відноситься до приблизних методів, що пояснюється обмеженням застосування точних методів розмірністю розв'язуваних завдань. Перевагами цього методу є його простота, надійність і гнучкість, що дозволяє враховувати цілий ряд додаткових факторів, які впливають на кінцеве рішення задачі. Цей алгоритм використовує поняття виграшів, щоб оцінити операції злиття між маршрутами. Погрішність рішення не перевершує в середньому 5-10%.

За даними статистики в нашій країні програмами 1С користується від 300 000 до 500 000 компаній [19]. Тому впровадження розробленого програмного модуля «надбудови» дозволить отримати велику економію, як трудових, так і матеріальних ресурсів в компаніях, які займаються експедируванням, автоперевезеннями, оптовою торгівлею, складуванням.

ПЕРЕЛІК ПОСИЛАНЬ

1. Проблеми розвитку логістики. URL: http://www.xcomp.biz/problemy-razvitiya-logistiki-v-rossii.html (дата звернення: 18.11.2017)

2. Логістика, як науковий напрямок. Фактори розвитку логістики. URL: http://libraryno.ru/3-1-logistika-kak-nauchnoe-napravlenie-faktory-razvitiya-logisti ki- baseslogist/ (дата звернення: 18.11.2017)

3. Автоматичне виробництво, кероване клієнтами, через інтерфейс. URL: http://www.plm.pw/2016/04/full-automation.html (дата звернення: 18.11.2017)

4. Доля логістики в Україні. URL: http://www.economy.nayka.com.ua/ ? op=1&z=1352 (дата звернення: 18.11.2017)

5. Огляд системи: 1С:Підприємство. URL: http://1c.ua/ua/v8/ (дата звернення: 18.11.2017)

6. NP-повні задачі URL: https://uk.wikipedia.org/wiki/NP-повна_задача

7. Метод гілок і меж URL: https://uk.wikipedia.org/wiki/Метод_гілок і_меж

8. Евристичні методи URL: http://pidruchniki.com/14350120 /menedzhment/evristichni_metodi_obgruntuvannya_priynyattya_rishen (дата звернення: 18.11.2017)

9. Метод мурашиної колонії. URL: https://uk.wikipedia.org/wiki/ /Мурашиний_алгоритм

10.Метод найближчого сусіда https://uk.wikipedia.org/wiki/Метод_ найближчого_сусіда

11. Метод найдешевшого включення. URL:http://ukrbukva.netpage,3,36766 Nepreryvnye-geneticheskie-algoritmy.html

12. Метод Кларка-Райта. URL: Https://infostart1c.pp.ua/public/443585/ (дата звернення: 18.11.2017)

13. Діаграма варіантів використання. URL: http://5fan.ru/wievjob.php ?id=21296 (дата звернення: 18.11.2017)

14. Діаграма послідовності. URL: https://uk.wikipedia.org/wiki/ діаграма_послідовності (дата звернення: 18.11.2017)

15. Діаграма класів. URL: https://uk.wikipedia.org/wiki/діаграма_класів (дата звернення: 18.11.2017)

16. Зарплата логіста URL: https://www.work.ua/jobs-odesa-логіст

17. Ролі в 1С. URL: http://v8.1c.ru/overview/Term_000000150.htm (дата звернення: 18.11.2017)

18. GoogleMapsApi. URL: https://developers.google.com/maps/documentation (дата звернення: 18.11.2017)

19. Логістика в Україні. URL: https://ain.ua/2017/05/16/chto-delat-klientam-1s-posle-sankcij-kommentarij-kompanii (дата звернення: 18.11.2017)

ДОДАТОК А
UML-ДІАГРАМА КЛАСІВ

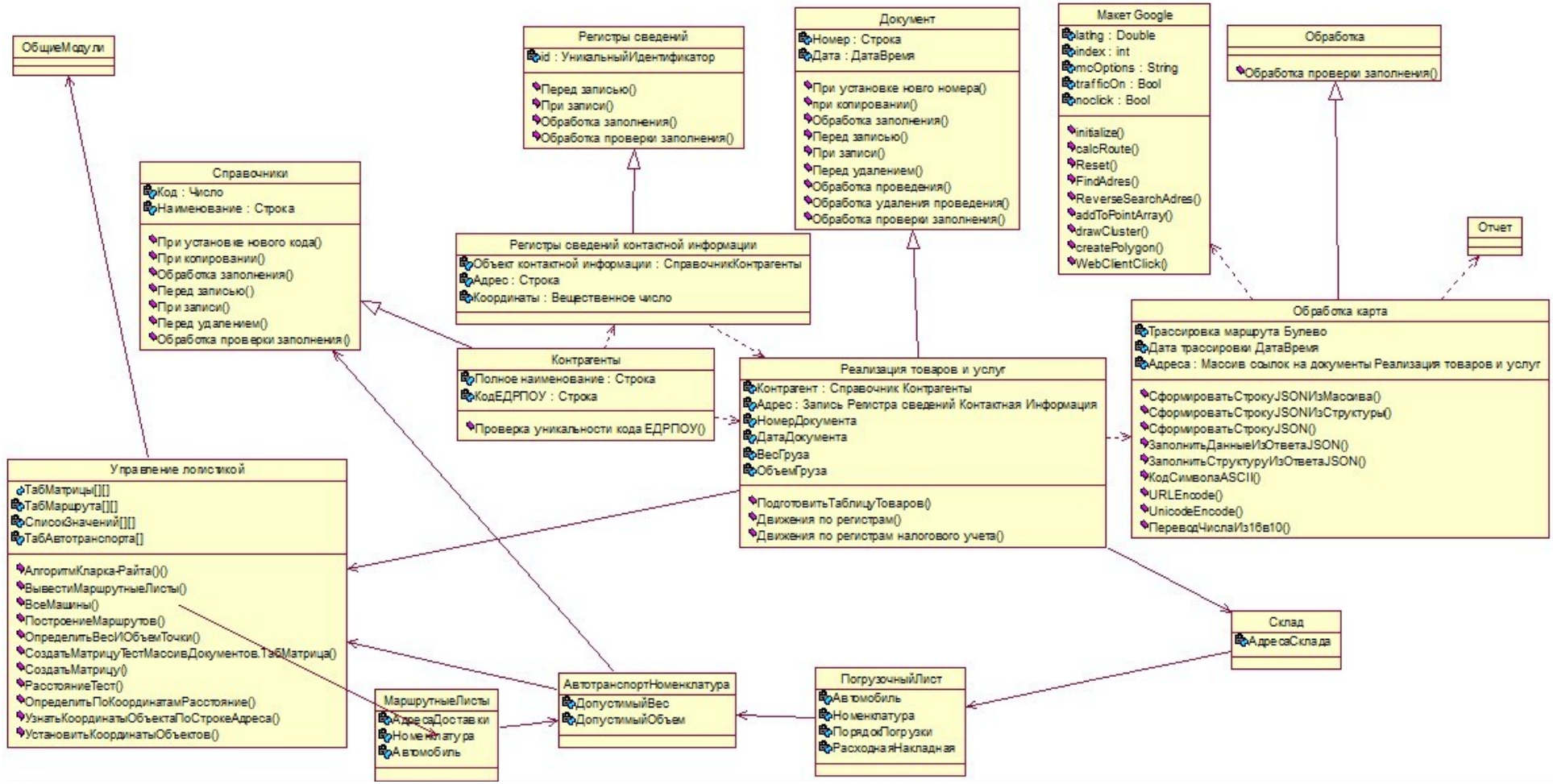


РИСУНОК А.1 - UML-ДІАГРАМА КЛАСІВ

ДОДАТОК Б
ФРАГМЕНТ ЛІСТИНГУ ПРОГРАМНОГО КОДУ

```
    Функция АлгоритмКларкаРайта(МассивДокументов = Неопределено,ДолготаСклада,ШиротаСклада,Тест,ТаблицаАвтотранспорта = Неопределено,Объем = 200) Экспорт

        Если МассивДокументов = Неопределено Тогда
            МассивДокументов = СформироватьМассивДокументовТест();
        КонецЕслИ;

        Если ТаблицаАвтотранспорта = Неопределено Тогда
            ТаблицаАвтотранспорта = ВсеМашины();
        КонецЕсли;

        ТабМатрица = Новый ТаблицаЗначений;

        //Шаг1 Составим матрицу расстояний
        Если Не Тест Тогда
            ТабМатрица = СоздатьМатрицу(МассивДокументов,ТабМатрица);
        Иначе
            ТабМатрица = СоздатьМатрицуТест(МассивДокументов,ТабМатрица);
        КонецЕсли;

        ////////////////////////////////

        ///Шаг 2 Ищем максимальный выигрыш
        ТабМаршрута = Новый ТаблицаЗначений;

        ТабМаршрута.Колонки.Добавить("Номер");
        ТабМаршрута.Колонки.Добавить("i");
        ТабМаршрута.Колонки.Добавить("j");
        ТабМаршрута.Колонки.Добавить("Усл1");
        ТабМаршрута.Колонки.Добавить("Усл2");
        ТабМаршрута.Колонки.Добавить("Усл3");
        ТабМаршрута.Колонки.Добавить("Шаг2Q1");
        ТабМаршрута.Колонки.Добавить("Шаг2Q2");
        ТабМаршрута.Колонки.Добавить("Шаг3");
        ТабМаршрута.Колонки.Добавить("Маршрут");
```

```
        ТабМаршрута.Колонки.Добавить("ТочкиМаршрута");

        спИспользованныхТочек = Новый СписокЗначений;
        Маршрут = 1;

        ТабТочек = Новый ТаблицаЗначений;
        ТабТочек.Колонки.Добавить("Номер");
        ТабТочек.Колонки.Добавить("X");
        ТабТочек.Колонки.Добавить("Y");
        ТабТочек.Колонки.Добавить("Выигрыш");

        //Ищем точку с максимальным выигрышем
        МаксимальныйВыигрыш = 0;
        ТекущийОбъем = 0;
        ТекущийВес   = 0;
        н = 1;
        Для i = 1 По ТабМатрица.Количество() - 1 Цикл
             Для j = 1 По i-1 Цикл
                  Если          ТабМатрица[i][j]          >
МаксимальныйВыигрыш Тогда
                       НоваяСтрока = ТабТочек.Добавить();
                       НоваяСтрока.Выигрыш = ТабМатрица[i]
[j];
                       НоваяСтрока.X = j;
                       НоваяСтрока.Y = i;
                       н = н + 1;
                  КонецЕсли;
             КонецЦикла;
        КонецЦикла;

        ТабТочек.Сортировать("Выигрыш Убыв");

        //Шаг 1
        ТабТочек.Колонки.Добавить("НомерМаршрута");
        ТабТочек.Колонки.Добавить("Маршрут");



        ТаблицаМаршрутов = Новый ТаблицаЗначений;
        ТаблицаМаршрутов.Колонки.Добавить("Маршрут");

ТаблицаМаршрутов.Колонки.Добавить("НомерМаршрута");
        ТаблицаМаршрутов.Колонки.Добавить("Автомобиль");
        ТаблицаМаршрутов.Колонки.Добавить("НачТочка");
```

```
        ТаблицаМаршрутов.Колонки.Добавить("КонТочка");
        ТаблицаМаршрутов.Колонки.Добавить("Объем");
        ТаблицаМаршрутов.Колонки.Добавить("МассивТочек");

ТаблицаМаршрутов.Колонки.Добавить("ТаблицаДокументов");

        Маршрут                                          =
ПостроениеМаршрутов(ТабТочек,ТаблицаМаршрутов,Объем,0,Мас
сивДокументов);


ДополнитьМаршрутТаблицейДокументов(Маршрут,МассивДокумент
ов);

        СтруктураТаблиц = Новый Структура;

        ТаблицаДокументовМаршрута = Новый ТаблицаЗначений;

ТаблицаДокументовМаршрута.Колонки.Добавить("НомерМаршрута
");

ТаблицаДокументовМаршрута.Колонки.Добавить("Адрес");

ТаблицаДокументовМаршрута.Колонки.Добавить("Документ");

        Для Каждого Стр Из Маршрут Цикл
             Для  Каждого  СтрДок  Из  Стр.ТаблицаДокументов
Цикл
                  НоваяСтрока                            =
ТаблицаДокументовМаршрута.Добавить();

                  НоваяСтрока.НомерМаршрута              =
Стр.НомерМаршрута;
                  НоваяСтрока.Адрес                      =
СтрДок.Документ.АдресДоставки.Адрес;
                  НоваяСтрока.Документ                   =
СтрДок.Документ;

             КонецЦикла;
        КонецЦикла;
```

```
СтруктураТаблиц.Вставить("ТаблицаМаршрутов",Маршрут);

СтруктураТаблиц.Вставить("ТаблицаДокументовМаршрута",Табл
ицаДокументовМаршрута);

        Возврат СтруктураТаблиц;
    КонецФункции

    Функция
ДополнитьМаршрутТаблицейДокументов(Маршрут,МассивДокумент
ов)

        Для Каждого Стр Из Маршрут Цикл
            спУпорядоченныйИндексДокументов              =
РазобратьСтроку(Стр.Маршрут,"-");
            ТаблицаДокументов = Новый ТаблицаЗначений;

  ТаблицаДокументов.Колонки.Добавить("НомерМаршрута");
            ТаблицаДокументов.Колонки.Добавить("Номер");
            ТаблицаДокументов.Колонки.Добавить("Адрес");

  ТаблицаДокументов.Колонки.Добавить("Документ");
            Для          н          =          0          По
спУпорядоченныйИндексДокументов.Количество() - 1 Цикл
                ТекИндекс                                    =
Число(спУпорядоченныйИндексДокументов[н].Значение);
                НоваяСтрока                                  =
ТаблицаДокументов.Добавить();
                НоваяСтрока.НомерМаршрута = н+1;
                НоваяСтрока.Номер                            =
спУпорядоченныйИндексДокументов[н];
                НоваяСтрока.Адрес                            =
МассивДокументов[ТекИндекс].Документ.АдресДоставки;
                НоваяСтрока.Документ                         =
МассивДокументов[ТекИндекс].Документ;
            КонецЦикла;
            Стр.ТаблицаДокументов = ТаблицаДокументов;
        КонецЦикла;

        Возврат Маршрут;

    КонецФункции
```

```
Функция РазобратьСтроку(Стр,Рзд)
  СтрРазб = Стр;
  Сп = Новый СписокЗначений;
    // пропускаем разделитель спереди
  // если есть
  Если Сред(СтрРазб, 1, 1) = Рзд Тогда
       СтрРазб = Сред(СтрРазб, 2);
  КонецЕсли;

  Инд = Найти(СтрРазб, Рзд);
  Пока Инд <> 0 Цикл
       ТекСтр = Сред(СтрРазб, 1, Инд - 1);
       Сп.Добавить(СокрЛП(ТекСтр));
       СтрРазб = Сред(СтрРазб, Инд + 1);
       Инд = Найти(СтрРазб, Рзд);
  КонецЦикла;

  Если Не ПустаяСтрока(СтрРазб) Тогда
       Сп.Добавить(СокрЛП(СтрРазб));
  КонецЕсли;

  Возврат Сп;

КонецФункции

//Передается матрица выигрышей
//Номер итерации - номер маршрута
//Список уже использованных в маршрутах точек
//Таблица построенных маршрутов
//Максимальный объем для текущего маршрута
//Максимальнчй вес для текущего маршрута
//Массив документов = исходный массив документов
//
Функция
ПостроениеМаршрутов(ТабТочек,ТаблицаМаршрутов,Максимальны
йОбъем,МаксимальныйВес,МассивДокументов)
  мИспользованныхТочек = Новый Массив;
  Для Каждого Стр Из ТабТочек Цикл
       СтруктураПроверки                              =
ПроверитьНаличиеТочекВМаршрутах(Стр.X,              Стр.Y,
ТаблицаМаршрутов);
```

```
            Если СтруктураПроверки.ТочкиОтсутствуют Тогда
///Новый маршрут обе точки еще не использованы

                мТочек = Новый Массив;
                мТочек.Добавить(Стр.X);
                мТочек.Добавить(Стр.Y);

                СтрокаМаршрута                          =
ТаблицаМаршрутов.Добавить();
                СтрокаМаршрута.МассивТочек   = мТочек;
                СтрокаМаршрута.Маршрут                  =
СокрЛП(Стр.Y)+"-"+СокрЛП(Стр.X);
                СтрокаМаршрута.НачТочка      = Стр.Y;
                СтрокаМаршрута.КонТочка      = Стр.X;
                СтрокаМаршрута.НомерМаршрута            =
ТаблицаМаршрутов.Количество();
                СтрокаМаршрута.Объем                    =
МассивДокументов[Стр.X].Документ.Товары.Итог("Количество"
);
                СтрокаМаршрута.Объем                    =
СтрокаМаршрута.Объем                                    +
МассивДокументов[Стр.Y].Документ.Товары.Итог("Количество"
);
                мИспользованныхТочек.Добавить(Стр.X);
                мИспользованныхТочек.Добавить(Стр.Y);
            Иначе   //Точки как-то присутствуют пробуем
добавить в маршрут или проигнорируем
                Для     Каждого     СтрокаМаршрута     Из
ТаблицаМаршрутов Цикл
                    //Проверяем не крайняя ли точка -
если крайняя попробуем добавить
                    Если            Стр.Y           =
СтрокаМаршрута.НачТочка Тогда ///Пытаемся добавляем слева
                        ///Но проверяем нет ли второй
точки в другом маршруте или в этом
                        Если
СтрокаМаршрута.МассивТочек.Найти(Стр.X)   =   Неопределено
Тогда //В этом маршруте нет
                            Если
ПроверитьНаличиеТочкиВМаршрутах(Стр.X,ТаблицаМаршрутов)
Тогда//Добавляем точку X в маршрут
                                Если
СтрокаМаршрута.Объем                                    +
```

```
МассивДокументов[Стр.X].Документ.Товары.Итог("Количество"
) > МаксимальныйОбъем Тогда
                                                Продолжить;
                                            КонецЕсли;

  СтрокаМаршрута.Маршрут     =   СокрЛП(Стр.X)   +   "-"   +
СтрокаМаршрута.Маршрут;

  СтрокаМаршрута.НачТочка = Стр.X;
                                            СтрокаМаршрута.Объем
=                   СтрокаМаршрута.Объем                    +
МассивДокументов[Стр.X].Документ.Товары.Итог("Количество"
);

  СтрокаМаршрута.МассивТочек.Добавить(Стр.X);

  мИспользованныхТочек.Добавить(Стр.X);
                                        КонецЕсли;
                                    КонецЕсли;
                                ИначеЕсли           Стр.X           =
СтрокаМаршрута.НачТочка Тогда ///Пытаемся добавляем слева
                                    ///Но проверяем нет ли второй
точки в другом маршруте или в этом
                                    Если
СтрокаМаршрута.МассивТочек.Найти(Стр.Y)   =   Неопределено
Тогда //В этом маршруте нет
                                        Если
ПроверитьНаличиеТочкиВМаршрутах(Стр.Y,ТаблицаМаршрутов)
Тогда//Добавляем точку в маршрут
                                            Если
СтрокаМаршрута.Объем                                        +
МассивДокументов[Стр.Y].Документ.Товары.Итог("Количество"
) > МаксимальныйОбъем Тогда
                                                Продолжить;
                                            КонецЕсли;

  СтрокаМаршрута.Маршрут   = СтрокаМаршрута.Маршрут + "-"
+ СокрЛП(Стр.Y);

  СтрокаМаршрута.НачТочка = Стр.Y;
                                            СтрокаМаршрута.Объем
=                   СтрокаМаршрута.Объем                    +
```

```
МассивДокументов[Стр.Y].Документ.Товары.Итог("Количество"
);

  СтрокаМаршрута.МассивТочек.Добавить(Стр.Y);

  мИспользованныхТочек.Добавить(Стр.Y);
                                КонецЕсли;
                            КонецЕсли;
                        ИначеЕсли          Стр.X          =
СтрокаМаршрута.КонТочка   Тогда   ///Пытаемся   добавляем
справа
                            ///Но проверяем нет ли второй
точки в другом маршруте или в этом
                            Если
СтрокаМаршрута.МассивТочек.Найти(Стр.Y)   =   Неопределено
Тогда //В этом маршруте нет
                                Если
ПроверитьНаличиеТочкиВМаршрутах(Стр.Y,ТаблицаМаршрутов)
Тогда//Добавляем точку в маршрут
                                    Если
СтрокаМаршрута.Объем                                    +
МассивДокументов[Стр.Y].Документ.Товары.Итог("Количество"
) > МаксимальныйОбъем Тогда
                                        Продолжить;
                                    КонецЕсли;

  СтрокаМаршрута.Маршрут  = СтрокаМаршрута.Маршрут + "-"
+ СокрЛП(Стр.Y);

  СтрокаМаршрута.КонТочка = Стр.Y;
                                    СтрокаМаршрута.Объем
=                 СтрокаМаршрута.Объем                  +
МассивДокументов[Стр.Y].Документ.Товары.Итог("Количество"
);

  СтрокаМаршрута.МассивТочек.Добавить(Стр.Y);

  мИспользованныхТочек.Добавить(Стр.Y);
                                КонецЕсли;
                            КонецЕсли;
                        ИначеЕсли          Стр.Y          =
СтрокаМаршрута.КонТочка   Тогда   ///Пытаемся   добавляем
справа
```

```
                                ///Но проверяем нет ли второй
точки в другом маршруте или в этом
                                Если
СтрокаМаршрута.МассивТочек.Найти(Стр.Х)   =   Неопределено
Тогда //В этом маршруте нет
                                    Если
ПроверитьНаличиеТочкиВМаршрутах(Стр.Х,ТаблицаМаршрутов)
Тогда//Добавляем точку в маршрут
                                        Если
СтрокаМаршрута.Объем                                    +
МассивДокументов[Стр.Х].Документ.Товары.Итог("Количество"
) > МаксимальныйОбъем Тогда
                                            Продолжить;
                                        КонецЕсли;

  СтрокаМаршрута.Маршрут   = СтрокаМаршрута.Маршрут + "-"
+ СокрЛП(Стр.Х);

  СтрокаМаршрута.КонТочка = Стр.Х;
                                        СтрокаМаршрута.Объем
=                  СтрокаМаршрута.Объем                 +
МассивДокументов[Стр.Х].Документ.Товары.Итог("Количество"
);

  СтрокаМаршрута.МассивТочек.Добавить(Стр.Х);

  мИспользованныхТочек.Добавить(Стр.Х);
                                    КонецЕсли;
                                КонецЕсли;
                            КонецЕсли
                        КонецЦикла;
                КонецЕсли;
                Если    мИспользованныхТочек.Количество()    =
МассивДокументов.Количество()-1 Тогда
                    Прервать;
                КонецЕсли;
        КонецЦикла;

        Возврат ТаблицаМаршрутов;

    КонецФункции
```

```
     Функция
ПроверитьНаличиеТочкиВМаршрутах(Точка,ТаблицаМаршрутов)

        КопияТаблицы = ТаблицаМаршрутов.Скопировать();

        НетТочки = Истина;
        Для Каждого СтрПроверки Из КопияТаблицы Цикл
             Если  Не  СтрПроверки.МассивТочек.Найти(Точка)
= Неопределено Тогда
                  НетТочки = Ложь;
             КонецЕсли;
        КонецЦикла;

        Возврат НетТочки;

     КонецФункции

     Функция
ПроверитьНаличиеТочекВМаршрутах(X,Y,ТаблицаМаршрутов)

        КопияТаблицы = ТаблицаМаршрутов.Скопировать();

        СтруктураПроверки = Новый Структура;

        СтруктураПроверки.Вставить("XНачальный",Ложь);
        СтруктураПроверки.Вставить("YНачальный",Ложь);
        СтруктураПроверки.Вставить("XКонечный",Ложь);
        СтруктураПроверки.Вставить("YКонечный",Ложь);

СтруктураПроверки.Вставить("ТочкиОтсутствуют",Истина);

        Для Каждого СтрКопия Из КопияТаблицы Цикл
             Если СтрКопия.НачТочка = X Тогда
                  СтруктураПроверки.XНачальный = Истина;
                  СтруктураПроверки.ТочкиОтсутствуют     =
Ложь;
             КонецЕсли;
             Если СтрКопия.НачТочка = Y Тогда
                  СтруктураПроверки.YНачальный = Истина;
                  СтруктураПроверки.ТочкиОтсутствуют     =
Ложь;
             КонецЕсли;
```

```
            Если СтрКопия.КонТочка = Y Тогда
                СтруктураПроверки.YКонечный = Истина;
                СтруктураПроверки.ТочкиОтсутствуют     =
Ложь;
            КонецЕсли;
            Если СтрКопия.КонТочка = X Тогда
                СтруктураПроверки.XКонечный = Истина;
                СтруктураПроверки.ТочкиОтсутствуют     =
Ложь;
            КонецЕсли;
            Если   Не   СтрКопия.МассивТочек.Найти(X)   =
Неопределено Тогда
                СтруктураПроверки.ТочкиОтсутствуют     =
Ложь;
            Иначе
                Если  Не  СтрКопия.МассивТочек.Найти(Y)  =
Неопределено Тогда
                    СтруктураПроверки.ТочкиОтсутствуют
= Ложь;
                КонецЕсли;
            КонецЕсли;
        КонецЦикла;

        Возврат СтруктураПроверки;

    КонецФункции

    Функция          СоздатьМатрицуТест(МассивДокументов,
ТабМатрица)

        ////////////////////////////////////////////
        // Формируем пустую матрицу
        //
        ////С 0=й строки и 0-й колонки
        //Колонки

        Для i = 0 По МассивДокументов.Количество()-1 Цикл


ТабМатрица.Колонки.Добавить("К"+СтрЗаменить(СокрЛП(i),Сим
волы.НПП,""));
        КонецЦикла;
        //Теперь строки
```

```
        Для i = 0 По МассивДокументов.Количество()-1 Цикл
             ТабМатрица.Добавить();
        КонецЦикла;
        ////////////////////////////////////////////////

        Таб = Новый ТаблицаЗначений;

        ////Заполним в матрицу растояния
        Для i = 0 По МассивДокументов.Количество()-1 Цикл
             Для j = i По МассивДокументов.Количество()-1
Цикл
                  Если i = j Тогда
                       ТабМатрица[i][j] = i;
                  Иначе
                       Если i = 0 Тогда
                            ЭтоСклад = Истина;
                       Иначе
                            ЭтоСклад = Ложь;
                       КонецЕсли;
                       //РасстояниеТест(Х1,У1,Х2,У2)
                       ТабМатрица[i][j]                    =
РасстояниеТест(МассивДокументов[i].X,МассивДокументов[i].
Y,МассивДокументов[j].X,МассивДокументов[j].Y);
                  КонецЕсли;
             КонецЦикла;
        КонецЦикла;

        ////Дополним матрицу выигрышем
        Для i = 1 По МассивДокументов.Количество()-1 Цикл
             Для j = i По МассивДокументов.Количество()-1
Цикл
                  Если i = j Тогда
                       Продолжить;
                  КонецЕсли;
                  Попытка
                       ТабМатрица[j][i] = ТабМатрица[0][i]
+ ТабМатрица[0][j] - ТабМатрица[i][j];
                  Исключение
                       ТабМатрица[j][i] = 0;
                  КонецПопытки;
             КонецЦикла;
        КонецЦикла;
```

```
        Возврат ТабМатрица;

    КонецФункции

    Функция СоздатьМатрицу(МассивДокументов, ТабМатрица)

        ////////////////////////////////////////////
        // Формируем пустую матрицу
        //
        ////С 0=й строки и 0-й колонки
        //Колонки
        Для i = 0 По МассивДокументов.Количество()-1 Цикл


ТабМатрица.Колонки.Добавить("К"+СтрЗаменить(СокрЛП(i),Сим
волы.НПП,""));
        КонецЦикла;
        //Теперь строки
        Для i = 1 По МассивДокументов.Количество() Цикл
            ТабМатрица.Добавить();
        КонецЦикла;
        ////////////////////////////////////////////////

        Таб = Новый ТаблицаЗначений;

        ////Заполним в матрицу растояния
        Для i = 0 По МассивДокументов.Количество()-1 Цикл
            Для j = i По МассивДокументов.Количество()-1
Цикл
                Если i = j Тогда
                    ТабМатрица[i][j] = i;
                Иначе
                    Если i = 0 Тогда
                        ЭтоСклад = Истина;
                    Иначе
                        ЭтоСклад = Ложь;
                    КонецЕсли;
                    ТабМатрица[i][j]                    =
ОпределитьПоКоординатамРасстояние(МассивДокументов[i],Мас
сивДокументов[j],ЭтоСклад);
                КонецЕсли;
            КонецЦикла;
        КонецЦикла;
```

```
        ////Дополним матрицу выигрышем
        ////Дополним матрицу выигрышем
        Для i = 1 По МассивДокументов.Количество()-1 Цикл
             Для j = i По МассивДокументов.Количество()-1
Цикл
                  Если i = j Тогда
                       Продолжить;
                  КонецЕсли;
                  Попытка
                       ТабМатрица[j][i] = ТабМатрица[0][i]
+ ТабМатрица[0][j] - ТабМатрица[i][j];
                  Исключение
                       ТабМатрица[j][i] = 0;
                  КонецПопытки;
             КонецЦикла;
        КонецЦикла;
        //Для  i  =  1  По  МассивДокументов.Количество()-1
Цикл
        //   Для j = 0 По МассивДокументов.КОличество() -
1 Цикл
        //        Если Не i = j Тогда
        //             ТабМатрица[i][j]  =  ТабМатрица[i-1]
[j] + ТабМатрица[i-1][j+1] - ТабМатрица[i][j+1];
        //        КонецЕсли;
        //   КонецЦикла;
        //КонецЦикла;

        Возврат ТабМатрица;

     КонецФункции

     Функция
СформироватьМассивДокументовТест();////Массив  структур  с
координатами теста для проверки

        МассивДокументов = Новый Массив;

        Структура = Новый Структура;
        Структура.Вставить("Номер",0);
        Структура.Вставить("X",10);
        Структура.Вставить("Y",15);
        Структура.Вставить("Объем",0);
```

```
МассивДокументов.Добавить(Структура);

Структура = Новый Структура;
Структура.Вставить("Номер",1);
Структура.Вставить("X",17);
Структура.Вставить("Y",15);
Структура.Вставить("Объем",450);
МассивДокументов.Добавить(Структура);

Структура = Новый Структура;
Структура.Вставить("Номер",2);
Структура.Вставить("X",6);
Структура.Вставить("Y",15);
Структура.Вставить("Объем",400);
МассивДокументов.Добавить(Структура);

Структура = Новый Структура;
Структура.Вставить("Номер",3);
Структура.Вставить("X",13);
Структура.Вставить("Y",3);
Структура.Вставить("Объем",400);
МассивДокументов.Добавить(Структура);

Структура = Новый Структура;
Структура.Вставить("Номер",4);
Структура.Вставить("X",9);
Структура.Вставить("Y",20);
Структура.Вставить("Объем",200);
МассивДокументов.Добавить(Структура);

Структура = Новый Структура;
Структура.Вставить("Номер",5);
Структура.Вставить("X",19);
Структура.Вставить("Y",7);
Структура.Вставить("Объем",150);
МассивДокументов.Добавить(Структура);

Структура = Новый Структура;
Структура.Вставить("Номер",6);
Структура.Вставить("X",8);
Структура.Вставить("Y",8);
Структура.Вставить("Объем",450);
МассивДокументов.Добавить(Структура);
```

```
Структура = Новый Структура;
Структура.Вставить("Номер",7);
Структура.Вставить("X",4);
Структура.Вставить("Y",14);
Структура.Вставить("Объем",250);
МассивДокументов.Добавить(Структура);

Структура = Новый Структура;
Структура.Вставить("Номер",8);
Структура.Вставить("X",17);
Структура.Вставить("Y",2);
Структура.Вставить("Объем",200);
МассивДокументов.Добавить(Структура);

Структура = Новый Структура;
Структура.Вставить("Номер",9);
Структура.Вставить("X",12);
Структура.Вставить("Y",22);
Структура.Вставить("Объем",450);
МассивДокументов.Добавить(Структура);

Структура = Новый Структура;
Структура.Вставить("Номер",10);
Структура.Вставить("X",6);
Структура.Вставить("Y",12);
Структура.Вставить("Объем",300);
МассивДокументов.Добавить(Структура);

Структура = Новый Структура;
Структура.Вставить("Номер",11);
Структура.Вставить("X",19);
Структура.Вставить("Y",17);
Структура.Вставить("Объем",475);
МассивДокументов.Добавить(Структура);

Структура = Новый Структура;
Структура.Вставить("Номер",12);
Структура.Вставить("X",12);
Структура.Вставить("Y",8);
Структура.Вставить("Объем",550);
МассивДокументов.Добавить(Структура);
```

```
        Возврат МассивДокументов;

    КонецФункции

    Функция ВсеМашины()

        Запрос = Новый Запрос;

        Запрос.Текст = "ВЫБРАТЬ
                       |    ТранспортныеСредства.Ссылка
                       |ИЗ
                                                          |
  Справочник.ТранспортныеСредства                        КАК
ТранспортныеСредства";

        Возврат Запрос.Выполнить().Выгрузить();

    КонецФункции

    //Определяем расстояние между двумя объектами в км
    Функция
ОпределитьПоКоординатамРасстояние(ПервыйОбъект,ВторойОбъе
кт,ЭтоСклад=Ложь);

        Если ПервыйОбъект = ВторойОбъект Тогда
            Возврат  0;  ////  Обе  точки  одинаковы  -
расстояние 0
        КонецЕсли;

        //ОтносительноеРасстояниеШироты                  =
Sqrt(Pow(ПервыйОбъект.X     -     ВторойОбъект.X,2)     +
Pow(ПервыйОбъект.Y - ВторойОбъект.Y,2));

         ///Расчитаем километры
        d = acos(sin(ПервыйОбъект.X)*sin(ВторойОбъект.X) +
cos(ПервыйОбъект.X)*cos(ВторойОбъект.X)*cos(ПервыйОбъект.
Y-ВторойОбъект.Y));

        Возврат d*111.197;

    КонецФункции
```

```
Функция РасстояниеТест(Х1,У1,Х2,У2)

  Возврат Sqrt(Pow(Х1 - Х2,2) + Pow(У1 - У2,2));

КонецФункции

///Из регистра сведений получаем координаты объекта
Функция
УзнатьКоординатыОбъектаПоСтрокеАдреса(Адрес,Контрагент=Не
определено,Склад = Неопределено)

  ////Определяем структуру координат
  СтруктураКоординат = Новый Структура;

  Если Контрагент = Неопределено Тогда //Это склад

      ////Ищем адрес в базе складов
      Запрос = Новый Запрос;

      Запрос.Текст = "ВЫБРАТЬ
                                                    |
  КооржинатыСобственногоСклада.Поле1,
                                                    |
  КооржинатыСобственногоСклада.Поле2
                          |ИЗ
                                                    |
  РегистрСведений.КооржинатыСобственногоСклада      КАК
КооржинатыСобственногоСклада
                          |ГДЕ
                                                    |
  КооржинатыСобственногоСклада.Объект = &Объект";

      Запрос.УстановитьПараметр("Объект",Склад);

      Выборка = Запрос.Выполнить().Выбрать();

      Выборка.Следующий();

      Если СокрЛП(Выборка.Поле1) = "" Тогда
//Координаты не определены - нгадо определить
```

```
                СтруктураКоординат                                =
УстановитьКоординатыОбъекта(Контрагент,Адрес,СтруктураКоо
рдинат);
            Иначе // Все ОК возвращаем координаты

  СтруктураКоординат.Вставить("Широта",Выборка.Поле1);

  СтруктураКоординат.Вставить("Долгота",Выборка.Поле2);
            КонецЕсли;

        Иначе // Это контрагент

            ////Ищем адрес в базе контрагента
            Запрос = Новый Запрос;

            Запрос.Текст = "ВЫБРАТЬ
                                                              |
  КонтактнаяИнформация.Поле1,
                                                              |
  КонтактнаяИнформация.Поле2
                            |ИЗ
                                                              |
  РегистрСведений.КонтактнаяИнформация                  КАК
КонтактнаяИнформация
                            |ГДЕ
                                                              |
  КонтактнаяИнформация.Представление                ПОДОБНО
&Представление
                            |    И
КонтактнаяИнформация.Объект = &Объект";


  Запрос.УстановитьПараметр("Объект",Контрагент);

  Запрос.УстановитьПараметр("Представление",Адрес);

            Выборка = Запрос.Выполнить().Выбрать();

            Выборка.Следующий();

            Если   СокрЛП(Выборка.Поле1)   =   ""   Тогда
//Координаты не определены - нгадо определить
```

```
                СтруктураКоординат =
УстановитьКоординатыОбъекта(Контрагент,Адрес,СтруктураКоо
рдинат);
            Иначе // Все ОК возвращаем координаты

  СтруктураКоординат.Вставить("Широта",Выборка.Поле1);

  СтруктураКоординат.Вставить("Долгота",Выборка.Поле2);
            КонецЕсли;

        КонецЕсли;

        Возврат СтруктураКоординат;

    КонецФункции

    ///Записывает координаты в регистр сведений
    Функция
УстановитьКоординатыОбъекта(Контрагент,Адрес,СтруктураКоо
рдинат)



    КонецФункции
    Перем СтруктураПоставщиковКарт Экспорт;
    Перем СтруктураЧисел;

    //////////////////////////РАБОТА С
JSON ////////////////////////////////////////
    Функция СформироватьСтрокуJSONИзМассива(Объект)

        СтрокаJSON = "[";

        Для каждого Элемент Из Объект Цикл
            СтрокаJSON = СтрокаJSON +
СформироватьСтрокуJSON(Элемент) + ",";
        КонецЦикла;

        Если Прав(СтрокаJSON, 1) = "," Тогда
            СтрокаJSON = Лев(СтрокаJSON,
СтрДлина(СтрокаJSON)-1);
        КонецЕсли;
```

```
        Возврат СтрокаJSON + "]";

    КонецФункции

    Функция СформироватьСтрокуJSONИзСтруктуры(Объект)

        СтрокаJSON = "{";

        Для каждого Элемент Из Объект Цикл

            Если Элемент.Значение = "" Тогда
                Продолжить;
            КонецЕсли;

            СтрокаJSON = СтрокаJSON + """" + Элемент.Ключ
+ """" + ":";

            Если ТипЗнч(Элемент.Значение) = Тип("Строка")
Тогда
                СтрокаJSON = СтрокаJSON + """"
+ URLEncode(Элемент.Значение) + """";

            ИначеЕсли ТипЗнч(Элемент.Значение)
Тип("Число") Тогда
                СтрокаJSON = СтрокаJSON
+ СтрЗаменить(Строка(Элемент.Значение), Символы.НПП, "");

            ИначеЕсли ТипЗнч(Элемент.Значение)
Тип("Булево") Тогда
                СтрокаJSON = СтрокаJSON
+ Формат(Элемент.Значение, "БЛ=false; БИ=true");

            ИначеЕсли ТипЗнч(Элемент.Значение)
Тип("Дата") Тогда
                // преобразование в unixtime
                СтрокаJSON = СтрокаJSON
+ Формат(ТекущаяДата() - Дата(1970,1,1,1,0,0), "ЧГ=0");

            ИначеЕсли ТипЗнч(Элемент.Значение)
Тип("Массив") Тогда
                СтрокаJSON = СтрокаJSON
+ СформироватьСтрокуJSON(Элемент.Значение);
```

```
            ИначеЕсли      ТипЗнч(Элемент.Значение)      =
Тип("Структура") Тогда
                СтрокаJSON      =      СтрокаJSON      +
СформироватьСтрокуJSON(Элемент.Значение);

            ИначеЕсли      ТипЗнч(Элемент.Значение)      =
Тип("ТаблицаЗначений") Тогда
                СтрокаJSON      =      СтрокаJSON      +
СформироватьСтрокуJSON(Элемент.Значение);

            Иначе
                СтрокаJSON   =   СтрокаJSON   +   """"   +
URLEncode(Строка(Элемент.Значение)) + """";

            КонецЕсли;

            СтрокаJSON = СтрокаJSON + ",";

        КонецЦикла;

        Если Прав(СтрокаJSON, 1) = "," Тогда
            СтрокаJSON          =          Лев(СтрокаJSON,
СтрДлина(СтрокаJSON)-1);
        КонецЕсли;

        Возврат СтрокаJSON + "}";

    КонецФункции

    Функция СформироватьСтрокуJSON(Объект) Экспорт

        СтрокаJSON = "";

        Если ТипЗнч(Объект) = Тип("Массив") Тогда
            СтрокаJSON                                    =
СформироватьСтрокуJSONИзМассива(Объект);

        ИначеЕсли ТипЗнч(Объект) = Тип("Структура") Тогда
            СтрокаJSON                                    =
СформироватьСтрокуJSONИзСтруктуры(Объект);
```

```
        ИначеЕсли ТипЗнч(Объект) = Тип("ТаблицаЗначений") Тогда
            // преобразуем таблицу значений в массив структур - работает дольше, но кода меньше
            // если нужна скорость, то нужно отдельно обработать таблицу значений

            СоставСтруктуры = "";
            Для каждого Колонка Из Объект.Колонки Цикл
                СоставСтруктуры = СоставСтруктуры + ?(ЗначениеЗаполнено(СоставСтруктуры), ",", "") + Колонка.Имя;
            КонецЦикла;

            МассивСтрок = Новый Массив;
            Для каждого Строка Из Объект Цикл
                СтруктураКолонок = Новый Структура(СоставСтруктуры);

  ЗаполнитьЗначенияСвойств(СтруктураКолонок, Строка);
                МассивСтрок.Добавить(СтруктураКолонок);
            КонецЦикла;

            СтрокаJSON = СформироватьСтрокуJSONИзМассива(МассивСтрок);

        КонецЕсли;

        Возврат СтрокаJSON;

    КонецФункции

    Процедура ЗаполнитьДанныеИзОтветаJSON(Результат, ТекстJSON, ТипДанных)

        ТекстJSON = СокрЛП(Сред(ТекстJSON, 2)); // удалим открывающий символ структуры(массива)

        НомерЗначения = 0;

        Пока ТекстJSON <> "" Цикл
```

```
            ПервыйСимвол = Лев(ТекстJSON, 1);
            Если ПервыйСимвол = "{" Тогда
                  // вложенная структура
                  Значение = Новый Структура;
                  ЗаполнитьДанныеИзОтветаJSON(Значение,
ТекстJSON, "Структура");

                  Если ТипДанных = "Структура" Тогда
                        Результат.Вставить("Значение"  +  ?
(НомерЗначения = 0, "", НомерЗначения), Значение);
                        НомерЗначения = НомерЗначения + 1;
                  ИначеЕсли ТипДанных = "Массив" Тогда
                        Результат.Добавить(Значение);
                  КонецЕсли;

            ИначеЕсли ПервыйСимвол = "[" Тогда
                  // вложенный массив
                  Значение = Новый Массив;
                  ЗаполнитьДанныеИзОтветаJSON(Значение,
ТекстJSON, "Массив");

                  Если ТипДанных = "Структура" Тогда
                        Результат.Вставить("Значение"  +  ?
(НомерЗначения = 0, "", НомерЗначения), Значение);
                        НомерЗначения = НомерЗначения + 1;
                  Иначе
                        Результат.Добавить(Значение);
                  КонецЕсли;

            ИначеЕсли ПервыйСимвол = "}" И ТипДанных =
"Структура" Тогда
                  // структура закончилась
                  ТекстJSON = СокрЛП(Сред(ТекстJSON, 2));
                  Если Лев(ТекстJSON, 1) = "," Тогда
                        ТекстJSON  =  СокрЛП(Сред(ТекстJSON,
2));
                  КонецЕсли;

                  Возврат;

            ИначеЕсли ПервыйСимвол = "]" И ТипДанных =
"Массив" Тогда
                  // массив закончился
```

```
                    ТекстJSON = СокрЛП(Сред(ТекстJSON, 2));
                    Если Лев(ТекстJSON, 1) = "," Тогда
                         ТекстJSON  =  СокрЛП(Сред(ТекстJSON,
2));
                    КонецЕсли;

                    Возврат;

               Иначе

                    Если ТипДанных = "Структура" Тогда
                         //ПервыйКавычка = Ложь;
                         //Если  Лев(ТекстJSON,  1)  =  """"
Тогда
                         //   ПервыйКавычка = Истина;
                         //КонецЕсли;

                         Поз = Найти(ТекстJSON, ":");
                         Если Поз = 0 Тогда
                              // неверный формат, прервемся
                              Прервать;
                         КонецЕсли;

                         //ПредпоследнийКавычка = Ложь;
                         //Если Сред(ТекстJSON, Поз - 1, 1)
= """" Тогда
                         //   ПредпоследнийКавычка = Истина;
                         //КонецЕсли;


                         ИмяЗначения = СокрЛП(Лев(ТекстJSON,
Поз - 1));
                         ИмяЗначения                        =
СтрЗаменить(ИмяЗначения, """", "");

                         ТекстJSON  =  СокрЛП(Сред(ТекстJSON,
Поз+1));

                         Если Лев(ТекстJSON, 1) = "{" Тогда
                              //      значение      является
структурой
                              Значение = Новый Структура;
```

```
  ЗаполнитьДанныеИзОтветаJSON(Значение,        ТекстJSON,
"Структура");

                        ИначеЕсли  Лев(ТекстJSON,  1)  =  "["
Тогда
                              // значение является массивом
                              Значение = Новый Массив;

  ЗаполнитьДанныеИзОтветаJSON(Значение,        ТекстJSON,
"Массив");

                        Иначе
                              // обычное значение
                              ПервыйКавычка = Ложь;
                              ПредпоследнийКавычка = Ложь;
                              Поз = 0;
                              Для     Сч     =     1     По
СтрДлина(ТекстJSON) Цикл
                                    Символ  =  Сред(ТекстJSON,
Сч, 1);

                                    Если Символ = """" Тогда
                                          Если    ПервыйКавычка
Тогда

  ПредпоследнийКавычка = Истина;
                                                Иначе
                                                      ПервыйКавычка  =
Истина;
                                                КонецЕсли;

                                    КонецЕсли;

                                    Если   (Символ   =   ","   И
((ПервыйКавычка    И    ПредпоследнийКавычка)    Или    (Не
ПервыйКавычка И Не ПредпоследнийКавычка))) ИЛИ Символ =
"]" ИЛИ Символ = "}" Тогда
                                          Поз = Сч;
                                          Прервать;
                                    КонецЕсли;
                              КонецЦикла;
```

```
                                    //ПредпоследнийКавычка = Ложь;
                                    //Если  Сред(ТекстJSON,  Поз  -
1, 1) = """" Тогда
                                    //   ПредпоследнийКавычка     =
Истина;
                                    //КонецЕсли;

                                    Если Поз = 0 Тогда
                                         Значение = ТекстJSON;
                                         ТекстJSON = "";

                                    Иначе
                                         Значение  =  Лев(ТекстJSON,
Поз - 1);
                                         Значение                  =
СтрЗаменить(Значение, """", "");
                                         ТекстJSON                 =
СокрЛП(Сред(ТекстJSON, Поз + ?(Сред(ТекстJSON, Поз, 1) =
",", 1, 0)));
                                    КонецЕсли;

                                    Значение = СокрЛП(Значение);

                               КонецЕсли;

                               Результат.Вставить(ИмяЗначения,
Значение);

                          ИначеЕсли ТипДанных = "Массив" Тогда

                               // обычное значение
                               Поз = 0;
                               Для  Сч  =  1  По  СтрДлина(ТекстJSON)
Цикл
                                    Символ  =  Сред(ТекстJSON,  Сч,
1);
                                    Если Символ = "," ИЛИ Символ =
"]" ИЛИ Символ = "}" Тогда
                                         Поз = Сч;
                                         Прервать;
                                    КонецЕсли;
                               КонецЦикла;
```

```
                        //ПредпоследнийКавычка = Ложь;
                        //Если Сред(ТекстJSON, Поз - 1, 1)
= """" Тогда
                        //   ПредпоследнийКавычка = Истина;
                        //КонецЕсли;

                        Если Поз = 0 Тогда
                             Значение = ТекстJSON;
                             ТекстJSON = "";

                        Иначе
                             Значение  =  Лев(ТекстJSON,  Поз
- 1);
                             Значение                       =
СтрЗаменить(Значение, """", "");
                             ТекстJSON                      =
СокрЛП(Сред(ТекстJSON, Поз + ?(Сред(ТекстJSON, Поз, 1) =
",", 1, 0)));

                        КонецЕсли;

                        Значение = СокрЛП(Значение);

                        Результат.Добавить(Значение);

                   КонецЕсли;


              КонецЕсли;

        КонецЦикла;

     КонецПроцедуры

     Функция           ЗаполнитьСтруктуруИзОтветаJSON(Знач
ТекстJSON) Экспорт

        Результат = Новый Структура;

        ТекстJSON  =  СтрЗаменить(ТекстJSON,  "\""",  """");
  // заменим последовательность \" на "
```

```
        //ТекстJSON = СтрЗаменить(ТекстJSON, """", "");
   // а теперь удалим все кавычки

        Если Лев(ТекстJSON, 1) = "{" Тогда
             // начало структуры
             ЗаполнитьДанныеИзОтветаJSON(Результат,
ТекстJSON, "Структура");

        ИначеЕсли Лев(ТекстJSON, 1) = "[" Тогда
             // начало массива
             МассивДанных = Новый Массив;
             ЗаполнитьДанныеИзОтветаJSON(МассивДанных,
ТекстJSON, "Массив");

             Результат.Вставить("Значение", МассивДанных);

        КонецЕсли;

        Возврат Результат;

     КонецФункции

     Функция КодСимволаASCII(Символ)
         КодUNICODE = КодСимвола(Символ);
         Если ((КодUNICODE > 1039) И (КодUNICODE < 1104))
Тогда
             Возврат (КодUNICODE - 848);
         ИначеЕсли КодUNICODE = 8470 Тогда
             Возврат 185;
         ИначеЕсли КодUNICODE = 1105 Тогда
             Возврат 184;
         ИначеЕсли КодUNICODE = 1025 Тогда
             Возврат 168;
         Иначе
             Возврат КодUNICODE;
         КонецЕсли;
     КонецФункции

     Функция URLEncode(value)

          table = "%00%01%02%03%04%05%06%07%08%09%0A%0B%0C
%0D%0E%0F%10%11%12%13%14" +
```

```
                        "%15%16%17%18%19%1A%1B%1C%1D%1E%1F
%20%21%22%23%24%25%26%27%28" +
                                    "%29%2A%2B%2C%2D%2E%2F
%30%31%32%33%34%35%36%37%38%39%3A%3B%3C" +
                                                "%3D%3E%3F
%40%41%42%43%44%45%46%47%48%49%4A%4B%4C%4D%4E%4F%50" +
                  "%51%52%53%54%55%56%57%58%59%5A%5B%5C%5D
%5E%5F%60%61%62%63%64" +
                        "%65%66%67%68%69%6A%6B%6C%6D%6E%6F
%70%71%72%73%74%75%76%77%78" +
                                    "%79%7A%7B%7C%7D%7E%7F
%80%81%82%83%84%85%86%87%88%89%8A%8B%8C" +
                                                "%8D%8E%8F
%90%91%92%93%94%95%96%97%98%99%9A%9B%9C%9D%9E%9F%A0" +
                  "%A1%A2%A3%A4%A5%A6%A7%A8%A9%AA%AB%AC%AD
%AE%AF%B0%B1%B2%B3%B4" +
                        "%B5%B6%B7%B8%B9%BA%BB%BC%BD%BE%BF
%C0%C1%C2%C3%C4%C5%C6%C7%C8" +
                                    "%C9%CA%CB%CC%CD%CE%CF
%D0%D1%D2%D3%D4%D5%D6%D7%D8%D9%DA%DB%DC" +
                                                "%DD%DE%DF
%E0%E1%E2%E3%E4%E5%E6%E7%E8%E9%EA%EB%EC%ED%EE%EF%F0" +
                  "%F1%F2%F3%F4%F5%F6%F7%F8%F9%FA%FB%FC%FD
%FE%FF";

        result = "";

        length = СтрДлина( value );

        Для i = 1 По  length Цикл
            symbol = Сред( value, i, 1 );
            //code = КодСимвола( symbol );
            code = КодСимволаASCII( symbol );
            result = result + Сред( table, code*3 + 1, 3
);
        КонецЦикла;



        Возврат result;
    КонецФункции
    //////////////////////////РАБОТА                    С
JSON ////////////////////////////////////////
```

```
    //Функция декодирует полученный unicod текст - из
ответа геокодера Рамблера
    // в привычный нам
    //Параметры:
    // Строка
    //Возвращаемое значение:
    // Строка
    Функция UnicodeEncode(Строка) Экспорт
      Результат = Истина;

      Попытка
          //регулярное выражение
          Рег = Новый COMОбъект("VBScript.RegExp");
          Рег.IgnoreCase = Истина;
          Рег.Global = Истина;
          Рег.Multiline = Ложь;
          Рег.Pattern = "u[0-9a-f]+";
          Колекция = Рег.Execute(Строка);
          Для Каждого Элемент Из Колекция Цикл
              Если СтрДЛина(Элемент.value) = 1 Тогда
                  Продолжить;
              КонецЕсли;

              КодСимвола                              =
ПереводЧислаИз16в10(Сред(ВРег(Элемент.value), 2));
              Символ = Символ(КодСимвола);
              Строка   =   СтрЗаменить(Строка,   "\"   +
Элемент.value, Символ);
          КонецЦикла;
      Исключение
          Результат = Ложь;
          Сообщить("Ошибка преобразования из Unicode",
СтатусСообщения.Информация);
      КонецПопытки;

      Возврат Результат;
    КонецФункции

    Функция ПереводЧислаИз16в10(Знач Значение)
      Результат = 0;

      //перевод значения в строку
```

```
        Если ТипЗнч(Значение) <> Тип("Строка") Тогда
            Значение = СокрЛП(Строка(Значение));
        КонецЕсли;

        МаксРазрядЦелых = 0;
        МаксРазрядЦелых = СтрДлина(Значение) - 1;

        н = МаксРазрядЦелых;
        Ин = 1;
        Пока н >= 0 Цикл
            ТекЗначение                                    =
СтруктураЧисел.Получить(Сред(Значение, Ин, 1)) * Pow(16,
н);
            Результат = Результат + ТекЗначение;
            н = н - 1;
            Ин = Ин + 1;
        КонецЦикла;

        Возврат Результат;
    КонецФункции

    ////////////////////////////
    СтруктураПоставщиковКарт = Новый Соответствие;
    СтруктураПоставщиковКарт.Вставить(0, "Яндекс");
    СтруктураПоставщиковКарт.Вставить(1, "Гугл");
    СтруктураПоставщиковКарт.Вставить(2, "2ГИС");
    СтруктураПоставщиковКарт.Вставить(3, "Рамблер");


    СтруктураЧисел = Новый Соответствие;
    СтруктураЧисел.Вставить("0", 0);
    СтруктураЧисел.Вставить("1", 1);
    СтруктураЧисел.Вставить("2", 2);
    СтруктураЧисел.Вставить("3", 3);
    СтруктураЧисел.Вставить("4", 4);
    СтруктураЧисел.Вставить("5", 5);
    СтруктураЧисел.Вставить("6", 6);
    СтруктураЧисел.Вставить("7", 7);
    СтруктураЧисел.Вставить("8", 8);
    СтруктураЧисел.Вставить("9", 9);
    СтруктураЧисел.Вставить("A", 10);
    СтруктураЧисел.Вставить("B", 11);
    СтруктураЧисел.Вставить("C", 12);
```

```
СтруктураЧисел.Вставить("D", 13);
СтруктураЧисел.Вставить("E", 14);
СтруктураЧисел.Вставить("F", 15);
```