Практическое задание для экзамена (по модулю)

ПМ.02 Осуществление интеграции программных модулей

Вариант №16

Задание №1

1.1 Анализ предметной области:

1. Определение предметной области:

Предметная область — это **логистика грузоперевозок и управление цепочками поставок**. Эта область охватывает процессы планирования, организации и контроля транспортировки товаров от производителей к потребителям. В нашем случае мы фокусируемся на оптимизации процесса транспортировки с учетом определенных ограничений.

2. Основные сущности:

- **Завод-производитель:**
  - **Атрибуты:** название, местоположение, максимальная мощность поставок (количество единиц продукции, которое завод может произвести и отгрузить).
  - **Примеры:** Белоруссия, Урал, Украина
- **Торговый склад:**
  - **Атрибуты:** название, местоположение, потребность в товарах (количество единиц продукции, которое склад хочет получить).
  - **Примеры:** Казань, Рига, Воронеж, Курск, Москва
- **Транспорт:**
  - **Атрибуты:** Тип (например, грузовик, поезд), стоимость перевозки единицы продукции между определенным заводом и складом.
  - **Пример:** Данные в таблице “Затраты на одну перевозку”
- **Груз:**
  - **Атрибуты:** тип, количество (выраженное в каких-либо единицах измерения, например, в тоннах или штуках).
  - **Пример:** Предполагается, что один тип груза перевозится из каждого завода на каждый склад.
- **План перевозок:**
  - **Атрибуты:** Количество груза, перевозимого от каждого завода к каждому складу.
- **Затраты:**
  - **Атрибуты:** Стоимость перевозки всего груза от каждого завода к каждому складу, общая стоимость всех перевозок.

3. Отношения между сущностями:

- **Отношение “производит”:** Завод производит продукцию, которая затем транспортируется. (Завод -> Груз).
- **Отношение “потребляет”:** склад потребляет продукцию, получая ее от заводов. (Склад -> Груз).
- **Отношение “перевозит”:** Транспорт перевозит продукцию с заводов на склады. (Завод -> Транспорт -> Склад)
- **Отношение “определяет”:** План перевозок определяет, какое количество груза необходимо перевезти по каждому маршруту.

4. Процессы:

- **Планирование перевозок:** определение оптимального плана перевозок (какое количество продукции перевозить с какого завода на какой склад) для минимизации общих транспортных расходов.
- **Распределение грузов:** распределение грузов с заводов на склады в соответствии с планом перевозок.
- **Отслеживание перевозок (не входит в эту задачу, но в реальной системе было бы необходимо):** мониторинг движения грузов в процессе транспортировки.
- **Управление запасами (не входит в эту задачу):** контроль запасов на складах и заводах.
- **Формирование отчетов:** Создание отчетов о плане перевозок и затратах.
- **Визуализация:** Отображение плана перевозок в наглядном виде.

5. Ограничения:

- **Ограничение по мощности поставок заводов:** каждый завод не может отгружать больше, чем его максимальная мощность.
- **Ограничение по потребностям складов:** каждый склад хочет получить определенное количество продукции.
- **Ограничение по минимизации затрат:** план перевозок должен быть таким, чтобы общие затраты были минимальными.

6. Цель:

- **Минимизация общих транспортных расходов:** целью модели является поиск такого плана перевозок, при котором суммарные затраты на транспортировку будут наименьшими при условии выполнения поставок и соблюдения ограничений.
- **Удовлетворение потребностей складов:** система должна стремиться к полному удовлетворению потребностей всех складов.

7. Потребители системы:

- **Логисты:** Они будут использовать систему для планирования перевозок.
- **Менеджеры по логистике:** Для принятия стратегических решений.

8. Итог:

Данный анализ предметной области помогает нам лучше понять:

- Какие сущности и процессы участвуют в задаче.
- Какие ограничения нужно учитывать.
- Какова цель модели и кто будет ее использовать.

1.2 Анализ требований:

Функциональные требования (Что система должна делать):

1. **Ввод данных:**
    - Заводы: названия, мощности поставок.
    - Склады: названия, потребности.
    - Стоимость перевозок: для каждой пары “завод-склад”.
2. **Расчет оптимального плана:**

- Найти план перевозок, минимизирующий общие затраты.
- Учитывать ограничения: мощность заводов, потребности складов.

3. **Вывод результатов:**

   - План перевозок: количество груза от каждого завода к каждому складу.
   - Суммарные затраты на перевозку.

4. **Отчетность:**

   - Формирование отчета в табличном виде.

5. **Визуализация:**

   - Отображение плана перевозок в виде диаграммы.

1.3 Нефункциональные требования (Как система должна работать):

1. **Точность:** Обеспечить правильный расчет оптимального плана.
2. **Эффективность:** Расчеты должны выполняться достаточно быстро.
3. **Удобство использования:** Интуитивно понятный ввод данных и вывод результатов.
4. **Надежность:** Система должна стабильно работать и выдавать корректные результаты.
5. **Масштабируемость:** система должна быть готова к добавлению новых заводов и складов.

Краткое описание целей:

- Минимизировать затраты на транспортировку.
- Удовлетворить потребности складов.
- Предоставить удобный инструмент для планирования перевозок.

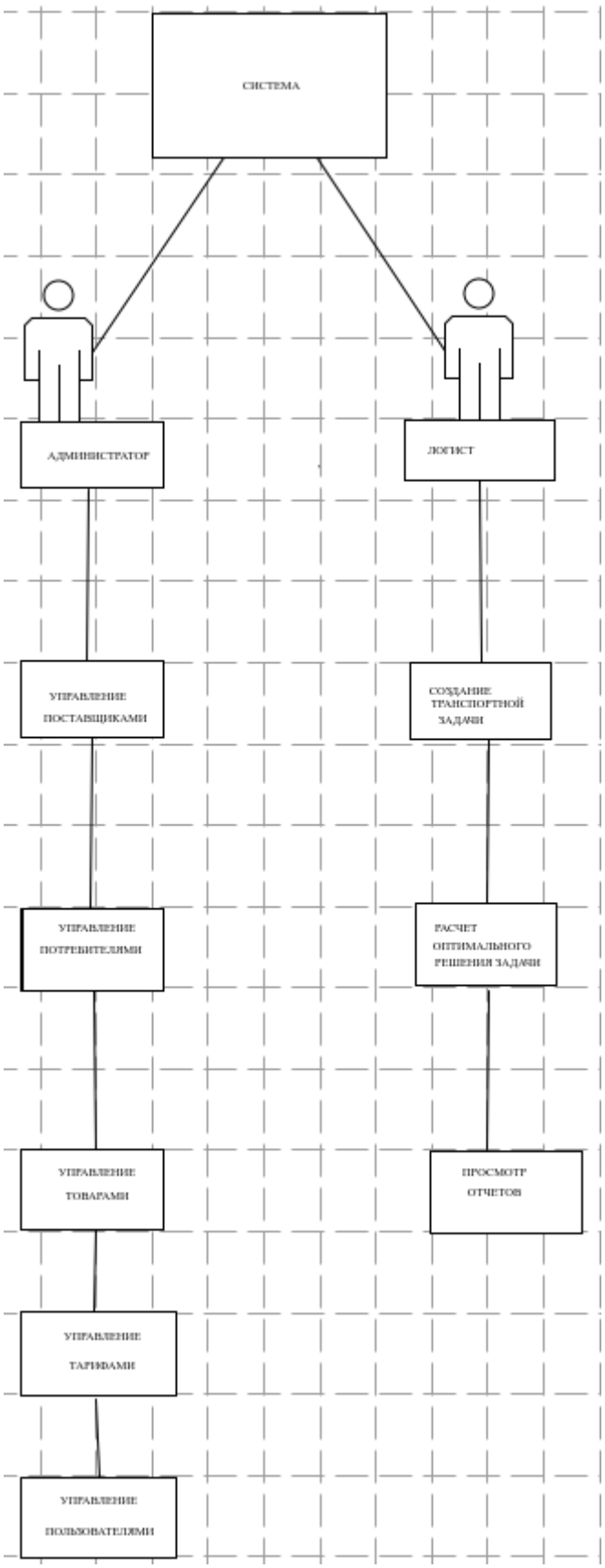
1.4 Диаграмма вариантов использования приложения для решения транспортных задач:

```
graph LR
    A[Логист] --> B(Ввести данные);
    B --> C{Проверить данные};
    C -- Корректны --> D(Рассчитать оптимальный план);
    C -- Ошибки --> B;
    D --> E(Вывести результаты);
    E --> F(Просмотреть план перевозок);
    E --> G(Просмотреть затраты);
     F --> H(Визуализировать план);
    E -->I(Сформировать отчёт);
    A -->J(Сохранить данные);
    A -->K(Загрузить данные);
```

Описание элементов диаграммы:

- **Актер:**
- **A [Логист]:** Основной пользователь приложения.
- **Варианты использования (действия, которые может выполнить пользователь):**
- **B (Ввести данные):** ввод данных о заводах, складах и стоимости перевозок.
- **C (Проверить данные):** Проверка введенных данных на корректность (например, корректность числовых значений, соответствие формату и т. д.).
- **D (Рассчитать оптимальный план):** запуск алгоритма для расчета оптимального плана перевозок (минимизация затрат).

- **E (Вывести результаты):** Вывести рассчитанный план перевозок и общие затраты.
- **F (Просмотреть план перевозок):** Просмотр подробного плана (количество груза от каждого завода до каждого склада).
- **G (Просмотреть затраты):** Просмотр общих затрат и, возможно, стоимости по отдельным маршрутам.
- **H (Визуализировать план):** Отображение плана перевозок в виде графика/диаграммы.
- **I (Сформировать отчёт):** Создание отчета в формате таблицы или текстового файла.
- **J (Сохранить данные):** сохранение введенных данных для дальнейшего использования.
- **K (Загрузить данные):** Загрузка ранее сохраненных данных.


СИСТЕМА
АДМИНИСТРАТОР
ЛОГИСТ
УПРАВЛЕНИЕ ПОСТАВЩИКАМИ
СОЗДАНИЕ ТРАНСПОРТНОЙ ЗАДАЧИ
УПРАВЛЕНИЕ ПОТРЕБИТЕЛЯМИ
РАСЧЕТ ОПТИМАЛЬНОГО РЕШЕНИЯ ЗАДАЧИ
УПРАВЛЕНИЕ ТОВАРАМИ
ПРОСМОТР ОТЧЕТОВ
УПРАВЛЕНИЕ ТАРИФАМИ
УПРАВЛЕНИЕ ПОЛЬЗОВАТЕЛЯМИ

Задание №2

Разработать программное приложение для построения опорного плана транспортной задачи методом минимальных элементов на языке программирования PYTHON

Листинг:

```
import numpy as np


def print_table(supply, demand, cost, allocation):
    print("Таблица распределения:")
    print("    ", end="")
    for j in range(len(demand)):
        print(f"{demand[j]:>5}", end=" ")
    print()

    for i in range(len(supply)):
        print(f"{supply[i]:>3} ", end="")
        for j in range(len(demand)):
            print(f"{allocation[i][j]:>5}", end=" ")
        print()

def min_cost_method(supply, demand, cost):
    # Инициализация аллокации
    allocation = np.zeros((len(supply), len(demand)), dtype=int)

    # Копирование значений
    supply = supply.copy()
    demand = demand.copy()

    while np.any(supply) and np.any(demand):
        # Нахождение минимального элемента в таблице затрат
        min_cost = np.inf
        min_i, min_j = -1, -1

        for i in range(len(supply)):
            for j in range(len(demand)):
```

```
            if supply[i] > 0 and demand[j] > 0 and cost[i][j] < min_cost:
                min_cost = cost[i][j]
                min_i, min_j = i, j

        # Устанавливаем максимальное возможное количество поставок
        quantity = min(supply[min_i], demand[min_j])
        allocation[min_i][min_j] = quantity
        supply[min_i] -= quantity
        demand[min_j] -= quantity

    return allocation

def main():
    # Входные данные
    supply = np.array([310, 260, 280])  # Поставки
    demand = np.array([180, 80, 200, 160, 220])  # Потребности
    cost = np.array([[10, 8, 6, 5, 4],  # Затраты
                     [6, 5, 4, 3, 6],
                     [3, 4, 5, 5, 9]])

    # Вычисление опорного плана
    allocation = min_cost_method(supply, demand, cost)

    # Печать результатов
    print_table(supply, demand, cost, allocation)

if __name__ == "__main__":
    main()
```

Работа Кода

```python
import numpy as np

def print_table(supply, demand, cost, allocation):
    print("Таблица распределения:")
    print("    ", end="")
    for j in range(len(demand)):
        print(f"{demand[j]:>5}", end=" ")
    print()

    for i in range(len(supply)):
        print(f"{supply[i]:>3} ", end="")
        for j in range(len(demand)):
            print(f"{allocation[i][j]:>5}", end=" ")
        print()

def min_cost_method(supply, demand, cost):
    # Инициализация аллокации
    allocation = np.zeros((len(supply), len(demand)), dtype=int)

    # Копирование значений
    supply = supply.copy()
    demand = demand.copy()

    while np.any(supply) and np.any(demand):
        # Нахождение минимального элемента в таблице затрат
        min_cost = np.inf
        min_i, min_j = -1, -1

        for i in range(len(supply)):
            for j in range(len(demand)):
                if supply[i] > 0 and demand[j] > 0 and cost[i][j] < min_cost:
                    min_cost = cost[i][j]
                    min_i, min_j = i, j

        # Устанавливаем максимальное возможное количество поставок
        quantity = min(supply[min_i], demand[min_j])
        allocation[min_i][min_j] = quantity
        supply[min_i] -= quantity
        demand[min_j] -= quantity

    return allocation
```

```python
def main():
    # Входные данные
    supply = np.array([310, 260, 280])  # Поставки
    demand = np.array([180, 80, 200, 160, 220])  # Потребности
    cost = np.array([[10, 8, 6, 5, 4],  # Затраты
                     [6, 5, 4, 3, 6],
                     [3, 4, 5, 5, 9]])

    # Вычисление опорного плана
    allocation = min_cost_method(supply, demand, cost)

    # Печать результатов
    print_table(supply, demand, cost, allocation)

if __name__ == "__main__":
    main()
```

```
Таблица распределения:
      180    80   200   160   220
310     0     0    80     0   220
260     0     0   100   160     0
280   180    80    20     0     0


** Process exited - Return Code: 0 **
Press Enter to exit terminal
```

Задание №3

Тест-Кейсы:

Тестовый случай	Поставки	Потребности	Затраты	Ожидаемая аллокация	Примечания
Тест 1	[310, 260, 280]	[180, 80, 200, 160, 220]	[[10, 8, 6, 5, 4], [6, 5, 4, 3, 6], [3, 4, 5, 5, 9]]	[[180, 0, 130, 0, 0], [0, 80, 0, 160, 0], [0, 0, 70, 0, 220]]	Основной случай
Тест 2	[0, 0, 0]	[100, 200]	[[1, 2], [3, 4], [5, 6]]	[[0, 0], [0, 0]]	Нет поставок
Тест 3	[100, 100]	[0, 0]	[[1, 2], [3, 4]]	[[0, 0], [0, 0]]	Нет потребностей
Тест 4	[100, 200]	[50, 100, 300]	[[1, 2, 3], [4, 5, 6]]	Ошибка или исключение	Несоответствующие суммы
Тест 5	[100, 100]	[100, 100]	[[1, 2], [2, 1]]	[[100, 0], [0, 100]]	Все равные