Учреждение образования

«БЕЛОРУССКИЙ ГОСУДАРСТВЕННЫЙ ТЕХНОЛОГИЧЕСКИЙ УНИВЕРСИТЕТ»

**МАТЕМАТИЧЕСКОЕ ПРОГРАММИРОВАНИЕ**

**Отчет по лабораторной работе №1**

**Вспомогательные функции**

Выполнила:

Клишевич Светлана Александровна

студентка 2 курса 4 группы

факультета ФИТ

Минск 2025

**ЦЕЛЬ РАБОТЫ:** приобретение навыков составления и отладки программ с использованием пользовательских функций для замера продолжительности процесса вычисления.

***Задание 1.***

Были разработаны три функции (start, dget и iget), в файле «Auxil.cpp», используя представленные в лабораторной работе спецификации (рисунок 1).

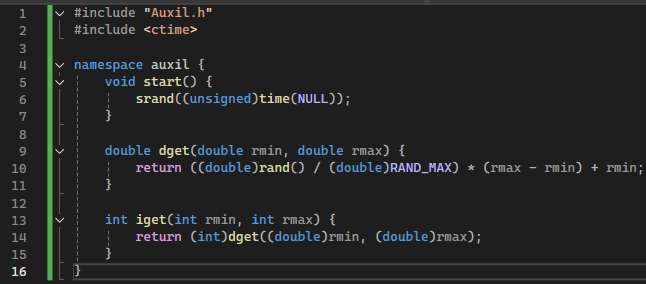


Рисунок 1 – Функции start, dget и iget

В заголовочном файле «Auxil.h» были разработаны прототипы функций (рисунок 2).

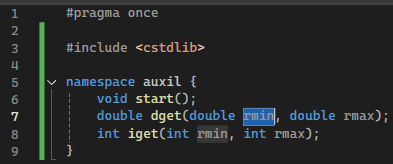


Рисунок 2 − Прототипы функций

И в основном файле «1задание» была произведена реализация (рисунок 3) (также представлен результат (рисунок 4)).

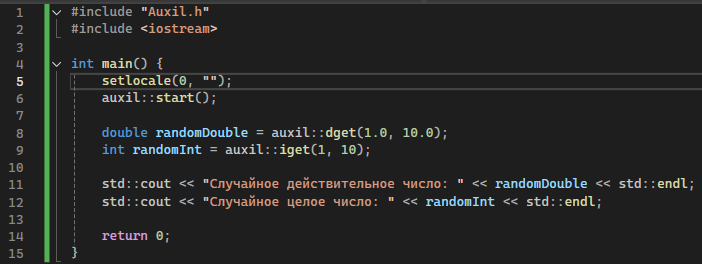


Рисунок 3 − Реализация



Рисунок 4 − Результат первого задания

***Задание 2***

Написан пример из условия в файле «2задание» (рисунок 5), реализован код (рисунок 6) для проверки работоспособности разработанных функций и замера продолжительности процесса вычислений.

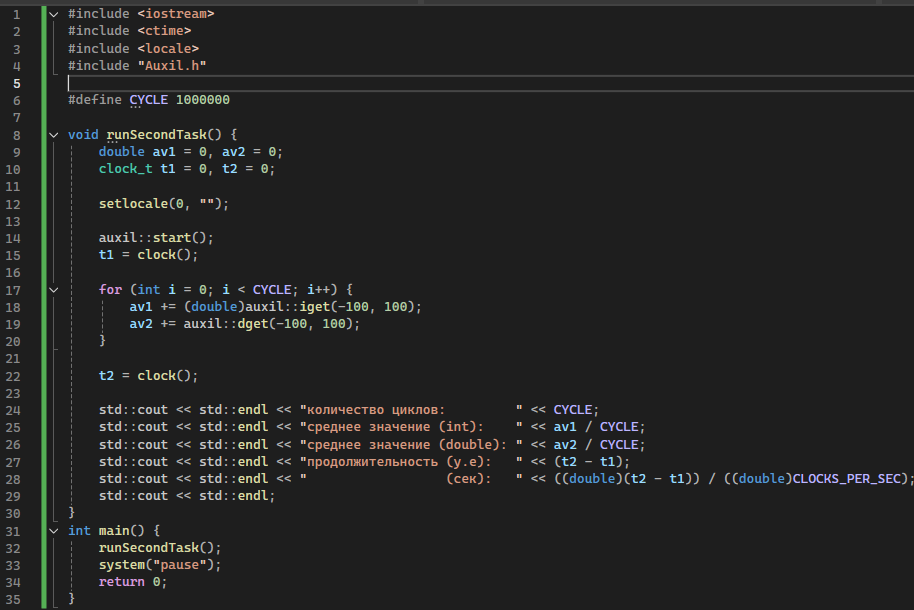


Рисунок 5 − Код второго задания

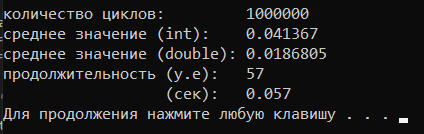


Рисунок 6 − Результат второго задания

**Задание 3**

Был построен график зависимости по второму заданию с изменениями количества циклов (рисунок 7).



Рисунок 7 − График зависимости

Для построения второго графика зависимости будет рассмотрено вычисление факториала (рисунок 8). Для вычисления факториала был написан соответствующий код.

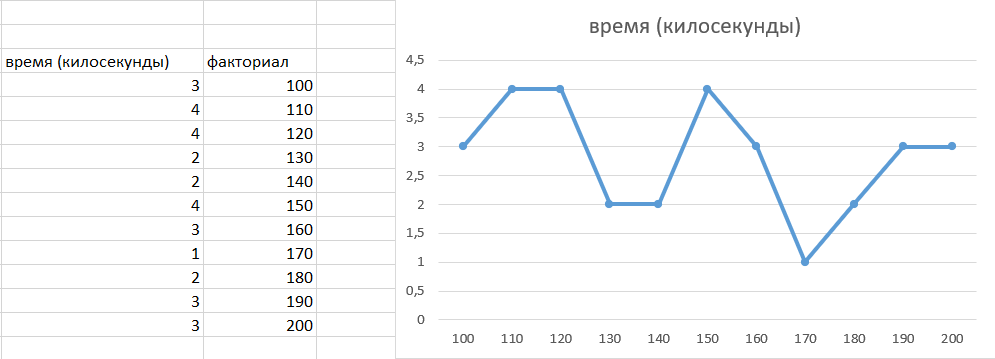


Рисунок 8 − График зависимости

# Лабораторная работа №2. Комбинаторные алгоритмы решения оптимизационных задач

**Цель работы:** приобрести навыки разработки генераторов подмножеств, перестановок, сочетаний и размещений на С++; научиться применять разработанные генераторы для решения задач о рюкзаке.

**Задание 1.** Разобрать и разработать генератор подмножеств заданного множества.

Ниже в листингах представлены заголовочные файлы и файлы .cpp с реализацией поставленной задачи.

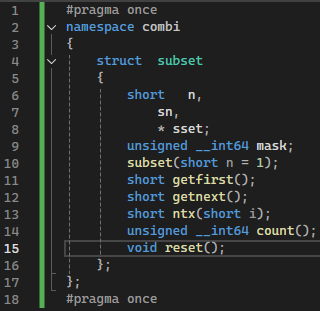


Рисунок 2.1 — файл Combi.h

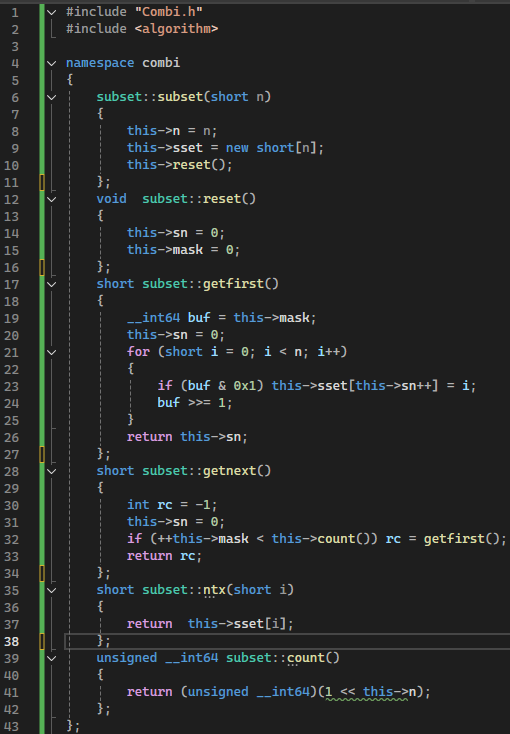


Рисунок 2.2 — файл Combi.cpp

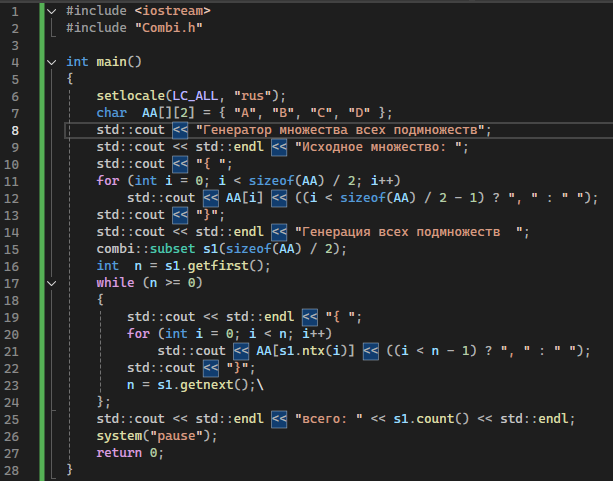


Рисунок 2.3 — файл Main.cpp

Результат работы генератора подмножеств заданного множества предоставлен на рисунке 2.4.

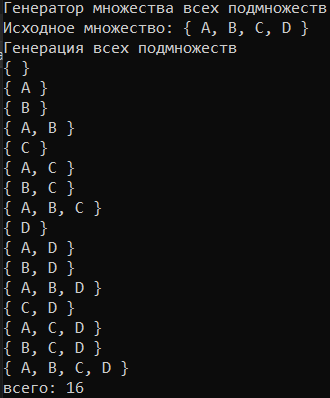
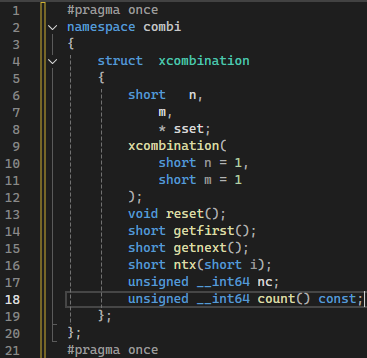


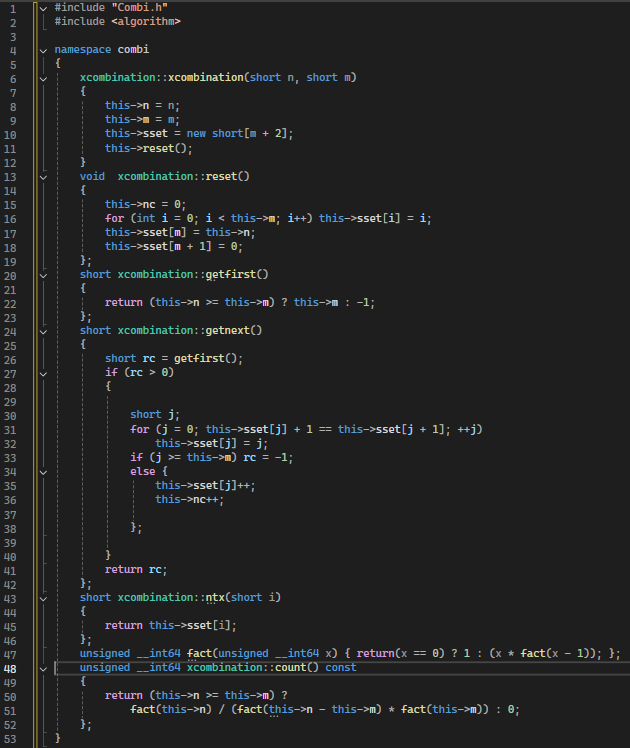
Рисунок 2.4 — результат работы генератора подмножеств заданного множества

**Задание 2.** Разобрать и разработать генератор сочетаний.

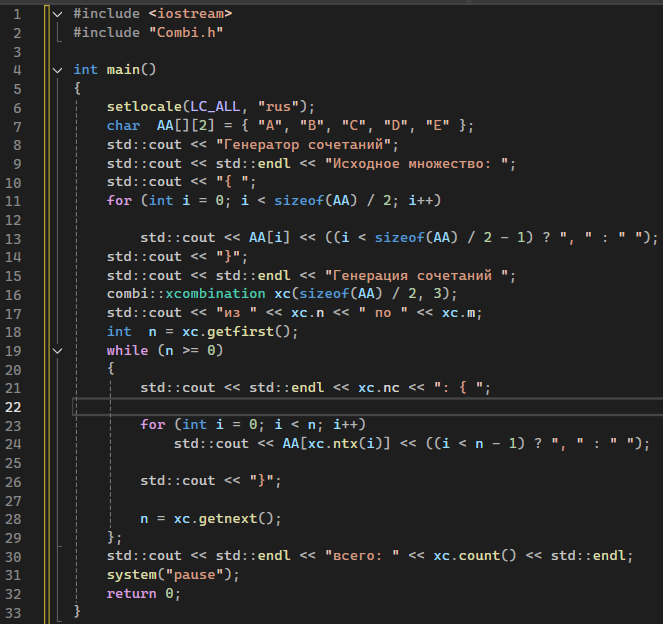
Ниже в листингах представлены заголовочные файлы и файлы .cpp с реализацией поставленной задачи.



Листинг 2.4 — файл Combi.h



Листинг 2.5 — файл Combi.cpp



Листинг 2.6 — файл Main.cpp

Результат работы генератора сочетаний предоставлен на рисунке 2.2.

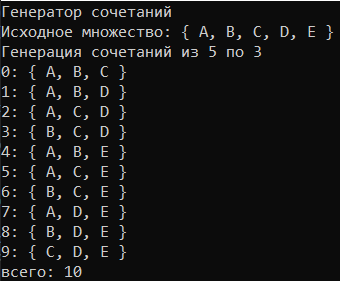
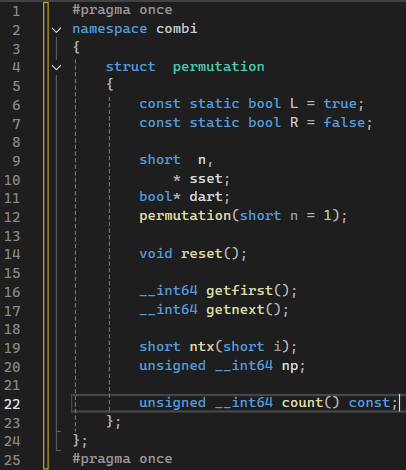


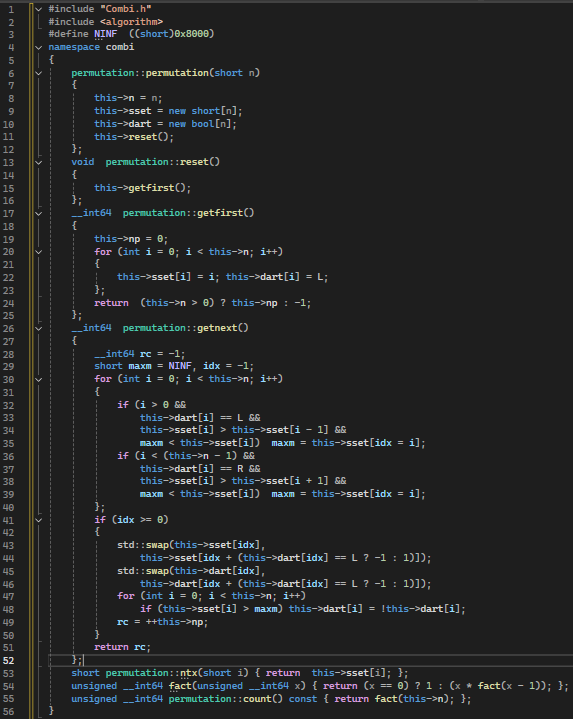
Рисунок 2.2 — результат работы генератора сочетаний

**Задание 3.** Разобрать и разработать генератор перестановок.

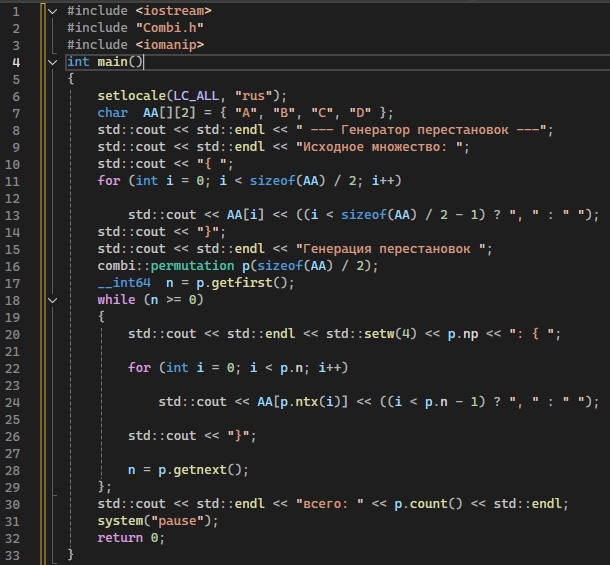
Ниже в листингах представлены заголовочные файлы и файлы .cpp с реализацией поставленной задачи.



Листинг 2.7 — файл Combi.h



Листинг 2.8 — файл Combi.cpp



Листинг 2.9 — файл Main.cpp

Результат работы генератора перестановок предоставлен на рисунке 2.3.

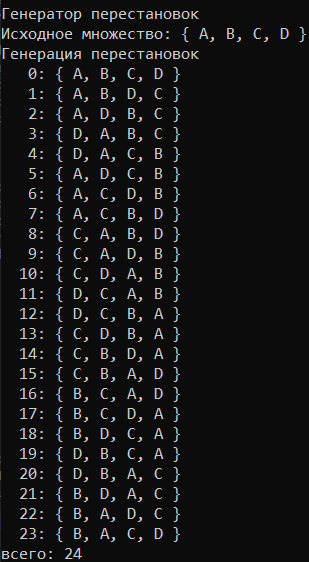
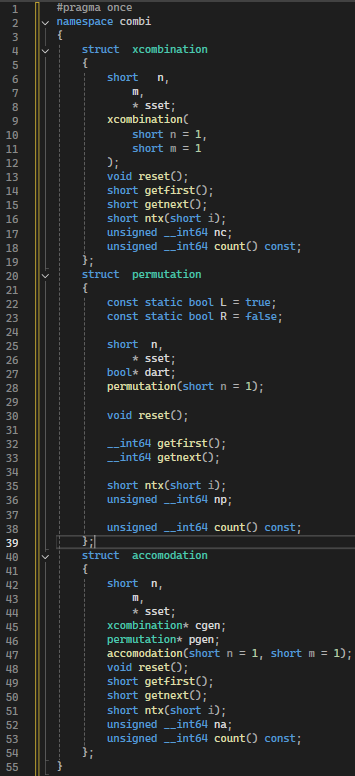


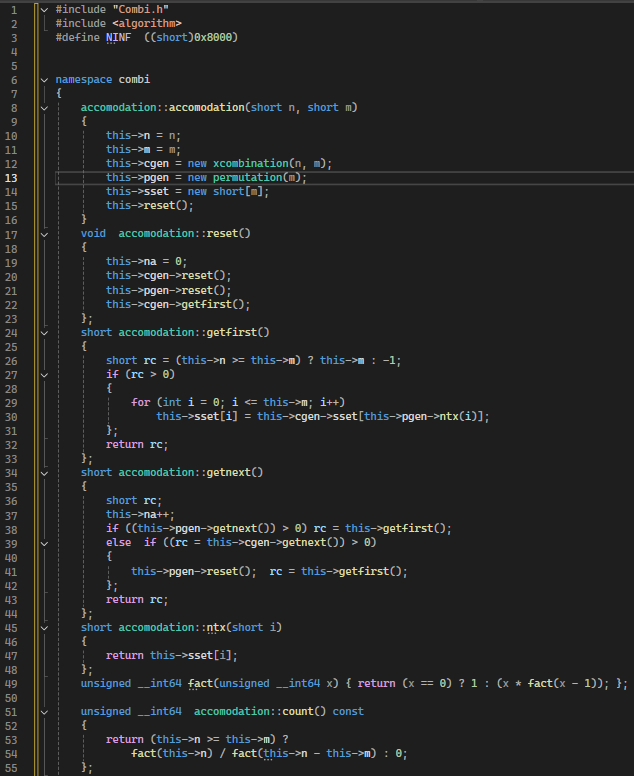
Рисунок 2.3 — результат работы генератора перестановок

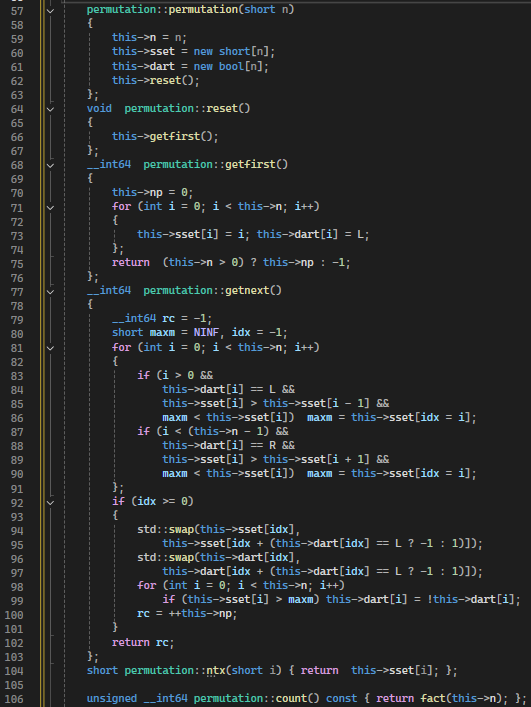
**Задание 4.** Разобрать и разработать генератор размещений.

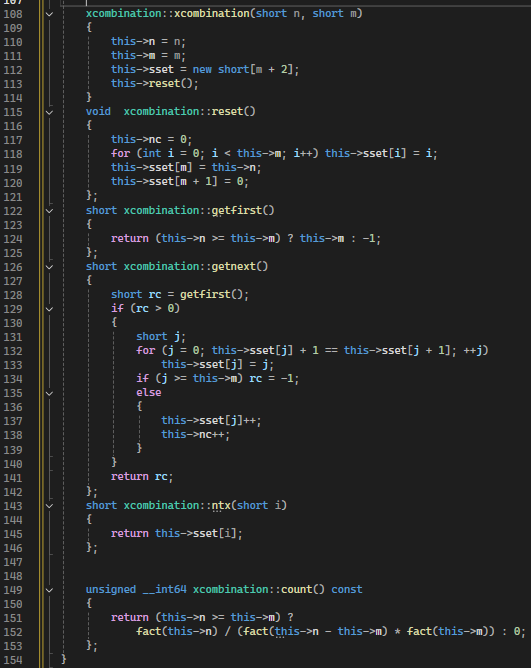
Ниже в листингах представлены заголовочные файлы и файлы .cpp с реализацией поставленной задачи.



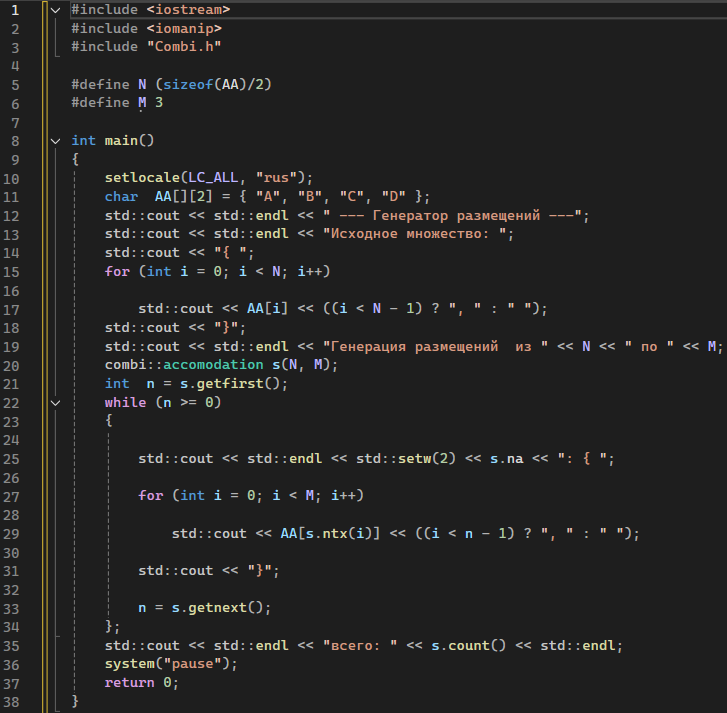
Листинг 2.10 — файл Combi.h







Листинг 2.11 — файл Combi.cpp



Листинг 2.12 — файл Main.cpp

Результат работы генератора размещений предоставлен на рисунке 2.4.

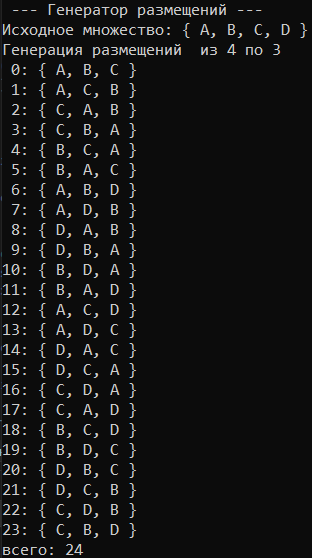


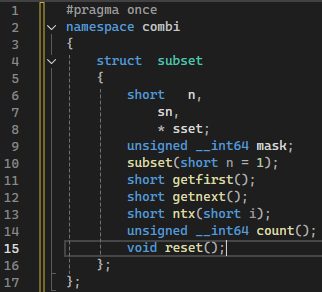
Рисунок 2.4 — результат работы генератора размещений

**Задание 5.**  Решить в соответствии с вариантом задачу и результат занести в отчет (Вариант распределяется по списку):

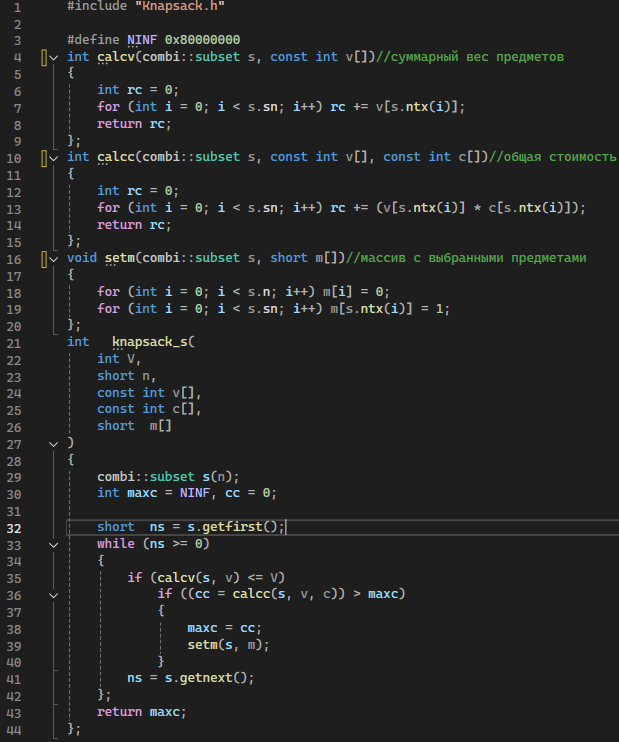
2, 6, 10, 14) упрощенную о рюкзаке (веса предметов и их стоимость сгенерировать случайным образом: вместимость рюкзака 300 кг, веса предметов 10 – 300 кг, стоимость предметов 5 – 55 у.е.; количество предметов – 18 шт.);

Схема решения задачи о рюкзаке с применением генератора множества всех подмножеств

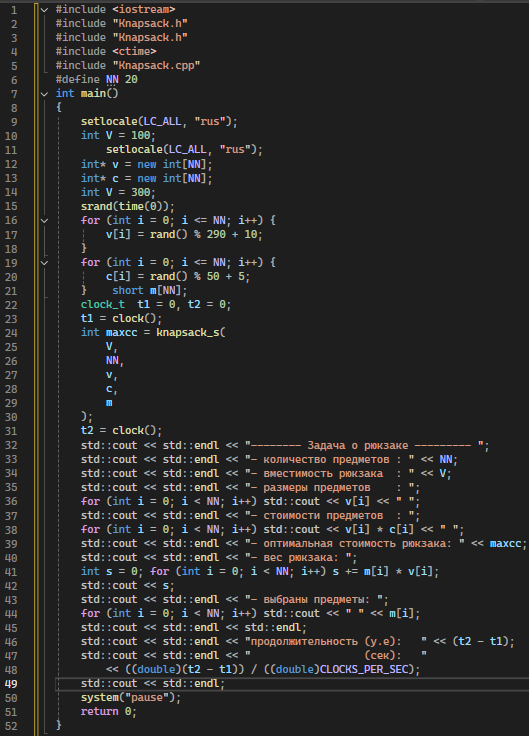
Ниже в листингах представлены заголовочные файлы и файлы .cpp с реализацией поставленной задачи.



Листинг 2.13 — файл Knapsack.h



Листинг 2.14 — файл Knapsack.cpp



Листинг 2.15 — файл Main.cpp

Решение задачи об оптимальной загрузке рюкзака показан на рисунке 2.5.

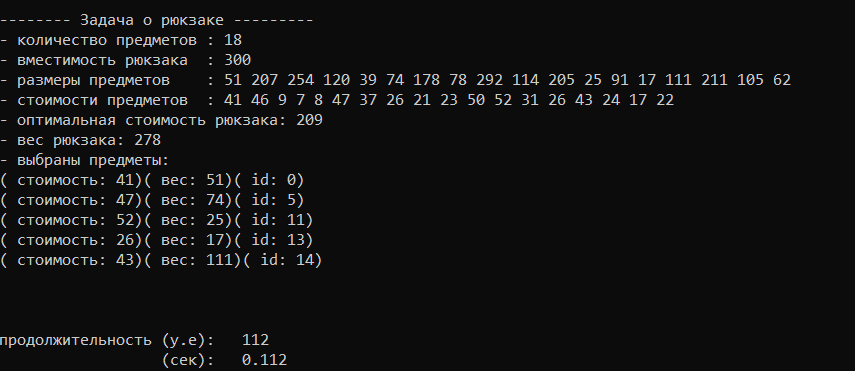


Рисунок 2.5 — решение задачи об оптимальной загрузке рюкзака

**Задание 6.** Исследовать зависимость времени вычисления необходимое для решения задачи (в соответствии с вариантом) от размерности задачи и результат в виде графика с небольшим пояснением занести в отчет:

(2, 6, 10, 14) упрощенную о рюкзаке (количество предметов 12 – 20 шт.);

Зависимость времени вычисления от количества предметов представлено на Рисунке 2.16.



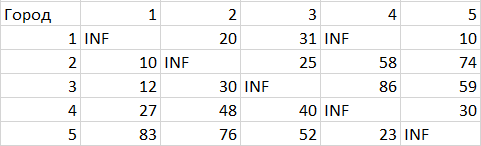
Рисунок 2.6 — Зависимость времени вычисления от количества предметов

Вывод: исходя из полученных данных и графика, можно заметить, что скорость выполнения программы плавно возрастает при добавлении количества предметов.

**Лабораторная работа 3. Метод ветвей и границ. Задача коммивояжера и методы её решения.**

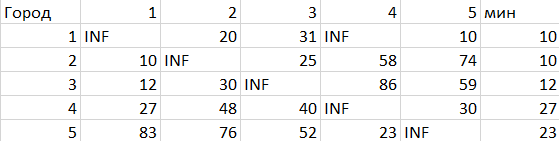
**Цель работы:** освоить общие принципы решения задач методом ветвей и границ, решить задачу о коммивояжере данным методом, сравнить полученное решение задачи с комбинаторным методом перестановок. 315423

***Задание 1***

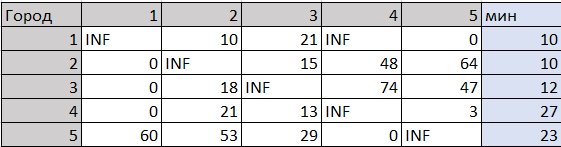


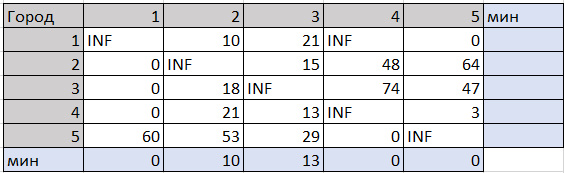
Составили условие, используя номер варианта по подгруппе.

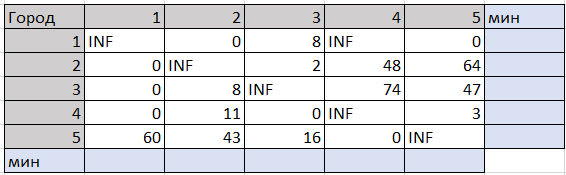
***Задание 2***



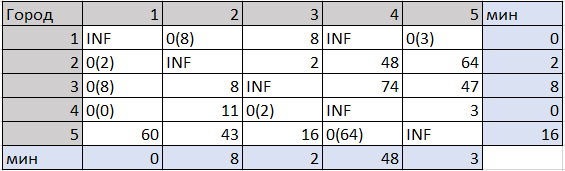
Изначально воспользуемся операцией редукции, то есть находим минимальное значение в каждой строке, после чего вычитаем из каждого элемента той же строки. В последующем делаем то же самое и со столбцами.



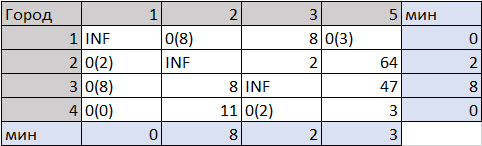




Исходя из этого поняли, что на данный момент минимальный путь равен 106.

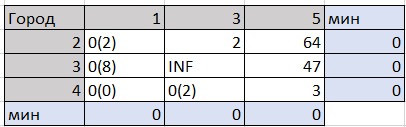


Теперь в ячейках, где значение равно нулю, расставили необходимые уровни. (5,4), (1,2), (3,1),(2,3)

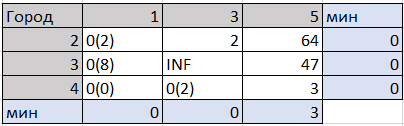


Удаляем строку и столбец, соответствующие ячейке с максимальным уровнем (5,4). После делаем то же самое, что делали до, то есть выставляем необходимые уровни и находим, в какой ячейки он наибольший: (1,2) и (3,1).

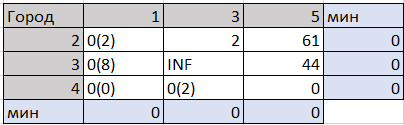
315423

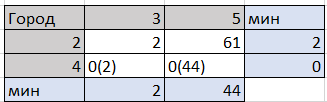


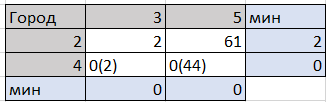
Удалили первую строку и второй столбец.

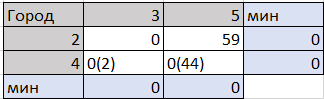


Опять воспользуемся операцией редукции, и прибавим соответствующее значение в минимальный путь (93+3=96).

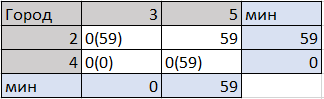


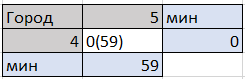


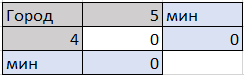




96+2=98



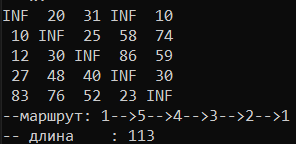




Ответ: минимальный путь, проходящий через все города без повтора, в котором начальная и конечная точки равны – 113.

(5,4), (4,3), (3,2), (2,1), (1,5),

***Задание 3***



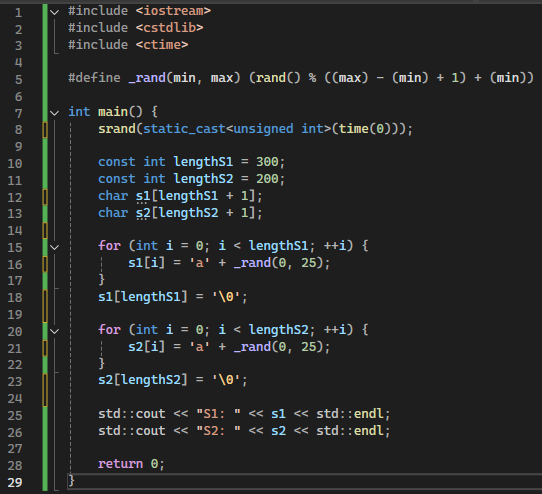
При помощи перестановок находится самый оптимальный путь, начинающийся из города 1, поэтому ответы разные.

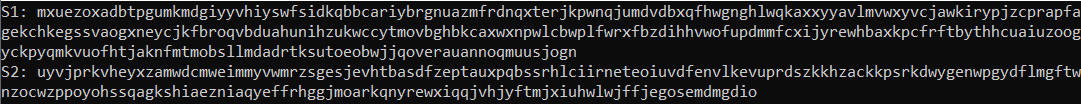
**Вывод.** Таким образом, в данной лабораторной работе были освоены общие принципы решения задач методом ветвей и границ, была решена задача о коммивояжере данным методом, полученное решение задачи сравнивалось с комбинаторным методом перестановок, в результате чего значения совпали.

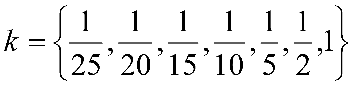
**Лабораторная работа 4. Динамическое программирование.**

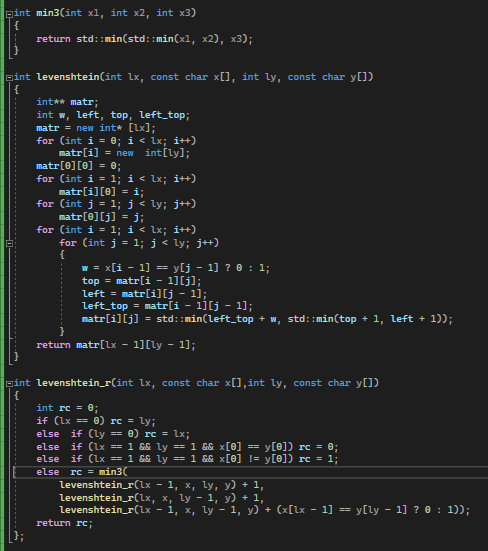
**Цель работы**: освоить общие принципы, решения задач методом динамического программирования, сравнить полученные решения задач с рекурсивным методом.

***Задание 1.*** На языке С++ сгенерировать случайным образом строку букв латинского алфавита S1 длиной 300 символов и S2длиной 200.



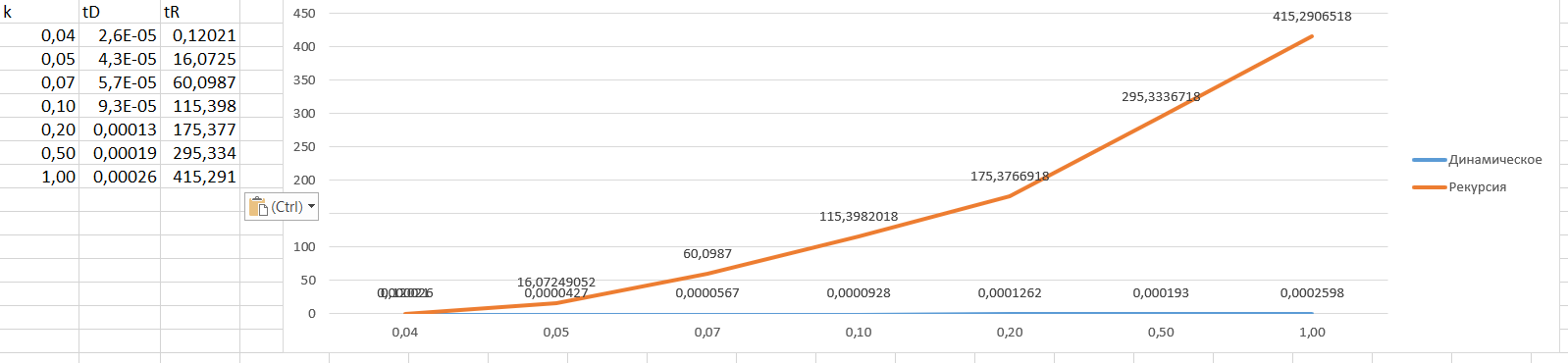


***Задание 2.*** Вычислить двумя способами (рекурсивно и с помощью динамического программирования)  – дистанцию Левенштейна для , где - длина строки ,  − строка, состоящая из первых  символов строки . (копии экрана и код вставить в отчет).



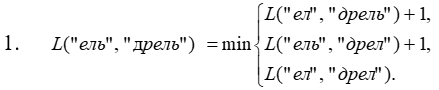


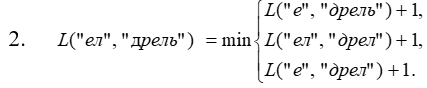
***Задание 3.*** Выполнить сравнительный анализ времени, затраченного на вычисление дистанции Левенштейна для двух методов решения. Построить графики зависимости времени вычисления от . (копии экрана и график вставить в отчет).

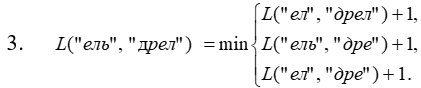


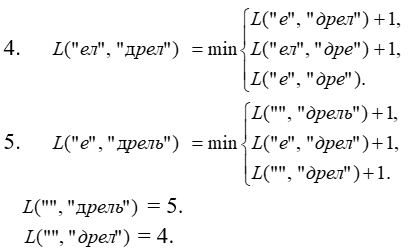
***Задание 4.*** Реализовать вручную пример вычисления дистанции Левенштейна при помощи рекурсивного алгоритма (в соответствии с вариантом) (каждый шаг алгоритма по примеру из лекции вставить в отчет).

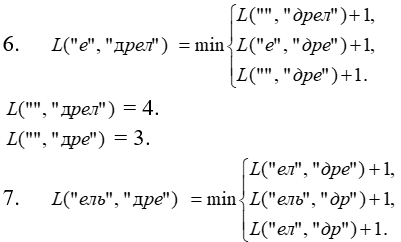


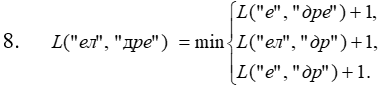


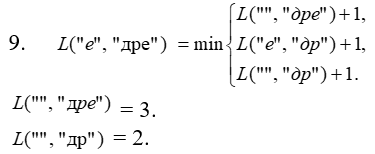


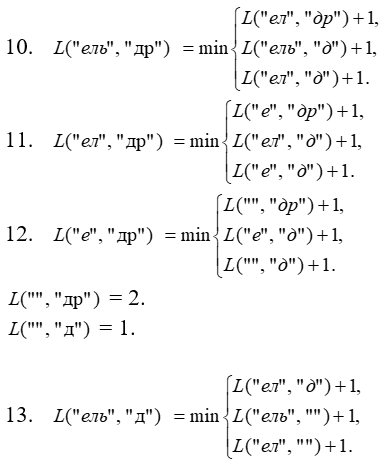


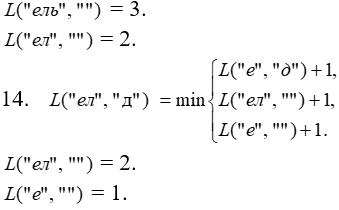


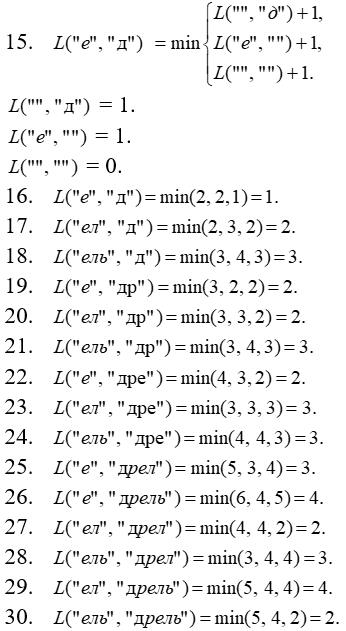






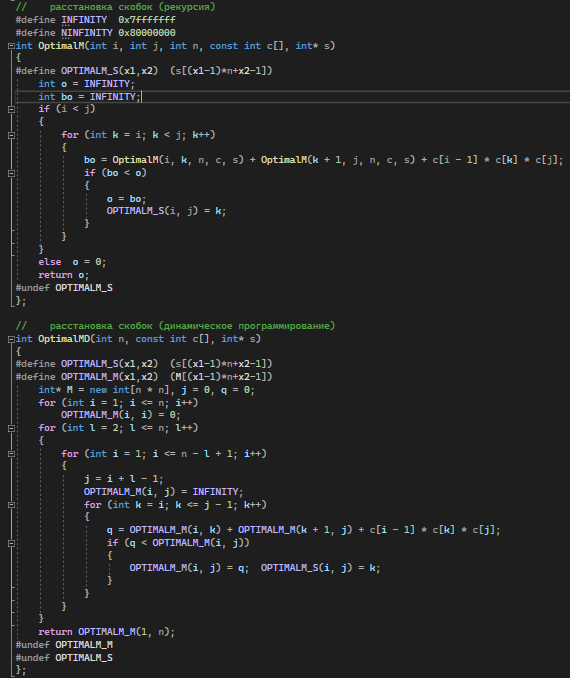


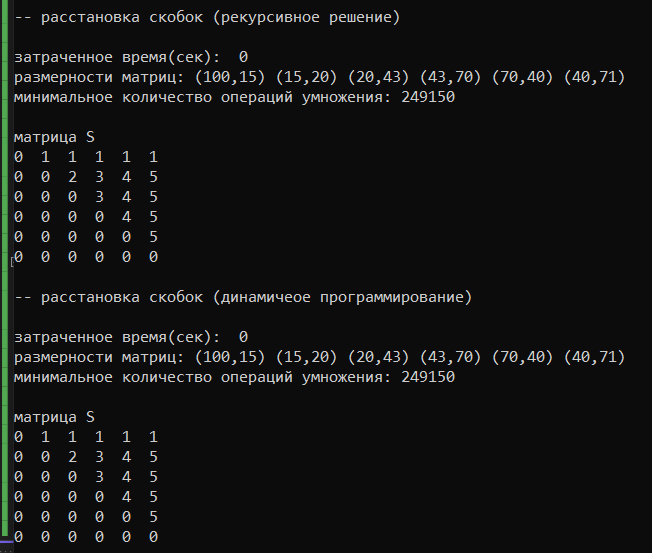




***Задание 5.*** **Четные варианты**. Выполнить сравнительный анализ времени, затраченного на решение задачи об оптимальной расстановке скобок при умножении нескольких матриц для двух методов решения (рекурсивное решение, динамическое программирование). Размерность матриц взять в соответствии с вариантом. Объяснить в отчете принцип расставления скобок по итоговой матрице + код + копии экрана.

q=m[i][k]+m[k+1][j]+p[i−1]×p[k]×p[j]





Принцип расстановки скобок по итоговой матрице:

Скобки расставляются по принципу «сначала внешние – затем внутренние». Имеется 6 матриц, вот их размерность.

А1=100\*15,

А2=15\*20,

А3=20\*43,

А4 =43\*70,

А5 =70\*40,

А6 =40\*71.

Матрица S:

**1 2 3 4 5 6**

**1** 0 1 1 1 1 1

**2** 0 0 2 3 4 5

**3** 0 0 0 3 4 5

**4** 0 0 0 0 4 5

**5** 0 0 0 0 0 5

**6** 0 0 0 0 0 0

Найдем элемент (1,6) в матрице S, он равен 1. Это означает, что точка разрыва между 1-ой и 6-ой матрицей находится после 1-ой матрицы. Что позволяет расставить скобки следующим образом:

A1\*(A2\*A3\*A4\*A5\*A6)

Точку разрыва между второй и шестой матрицей определяет элемент (2,6). Он равен 5. Следовательно разрыв будет после 5-ой матрицы.

A1\*((A2\*A3\*A4\*A5) \*A6)

Далее берем элемент (2,5) и получаем, что он равен 4. Следовательно получаем:

A1\*(((A2\*A3\*A4) \*A5) \*A6)

И на последнем шаге мы возьмем элемент (2,4) и он равен 3:

A1\*((((A2\*A3) \*A4) \*A5) \*A6)

Это выражение и есть конечное.

Полученная расстановка скобок позволяет получить минимальное количество операций умножения, равное 15125.

1. Динамическое программирование применяется там, где задача разбивается на пересекающиеся подзадачи. Основные области:

* Оптимизация (экономика, логистика, управление)
* Биоинформатика (например, выравнивание ДНК)
* Теория графов (поиск кратчайших путей)
* Машинное обучение (некоторые алгоритмы, например, Viterbi)
* Игры и искусственный интеллект (вычисление выигрышных стратегий)
* Финансовое моделирование

2. Задача состоит в **поиске оптимального решения** путём разбиения исходной задачи на **меньшие подзадачи**, **запоминания** их решений (мемоизация), и **повторного использования** этих решений, чтобы не решать одну и ту же подзадачу несколько раз.

* 3 . **Аддитивная** функция: результат считается **суммой** компонентов. Пример: f(x) = a + b + c.
* **Мультипликативная** функция: результат получается как **произведение** компонентов. Пример: f(x) = a \* b \* c.

Обе функции часто встречаются в задачах динамического программирования, где надо складывать или умножать частичные результаты.

4. Иными словами, **оптимальное решение задачи содержит в себе оптимальные решения её подзадач**.

5. Рекурсивный алгоритм — это алгоритм, который **вызывает сам себя** для решения **подзадачи**, пока не достигнет **базового случая**, решаемого напрямую.

6. Рекурсивная функция — это функция, которая **внутри себя вызывает саму себя** с новыми (обычно уменьшенными) аргументами.

7. Системный стек (stack) — это область памяти, где хранятся:

* локальные переменные
* адрес возврата из функции
* данные о текущем состоянии вызова

Каждый вызов функции помещается в стек, а после завершения — удаляется (снимается со стека).

8. Глубина рекурсии — это **максимальное количество вложенных вызовов функции**, прежде чем будет достигнуто завершение (базовый случай).  
Если глубина слишком большая, может произойти **переполнение стека**.

9. Принцип **"разделяй и властвуй"** работает так:

1. **Раздели** задачу на несколько более мелких (обычно одинакового типа).
2. **Реши** каждую мелкую задачу (рекурсивно или напрямую).
3. **Объедини** решения, чтобы получить итоговый ответ.

10. Редакционное расстояние (или расстояние Левенштейна) — это **минимальное количество операций (вставка, удаление, замена)**, необходимых для преобразования одной строки в другую.

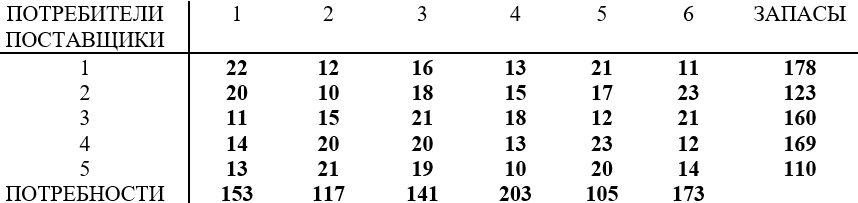
Пример: из "кот" в "крот" — 1 вставка ("р").

11. **Подпоследовательность** — это последовательность, полученная из другой путём **удаления некоторых (или ни одного) элементов**, **без изменения порядка** оставшихся.

Пример: из "abcdef" → подпоследовательность "ace".

**Лабораторная работа №5**

Условие:



Для разрешимости транспортной задачи необходимо, чтобы суммарные запасы продукции у поставщиков равнялись суммарной потребности потребителей. Проверим это условие.

∑a = 178 + 123 + 160 + 169 + 110 = 740

∑b = 153 + 117 + 141 + 203 + 105 + 173 = 892

Так как запасы поставщиков меньше потребности потребителей, введем фиктивного поставщика 6, с запасом продукции равным 892-740=152. Стоимость доставки единицы продукции от фиктивного поставщика ко всем потребителям примем равной нулю.

|  |  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- | --- |
| Поставщик | Потребитель | | | | | | Запас |
| 1 | 2 | 3 | 4 | 5 | 6 |
| 1 | 22 | 12 | 16 | 13 | 21 | 11 | 178 |
| 2 | 20 | 10 | 18 | 15 | 17 | 23 | 123 |
| 3 | 11 | 15 | 21 | 18 | 12 | 21 | 160 |
| 4 | 14 | 20 | 20 | 13 | 23 | 12 | 169 |
| 5 | 13 | 21 | 19 | 10 | 20 | 14 | 110 |
| 6 | 0 | 0 | 0 | 0 | 0 | 0 | 152 |
| Потребность | 153 | 117 | 141 | 203 | 105 | 173 |  |

Теперь выполняется условие.

Этап I

*Метод наименьшей стоимости*

Суть метода заключается в том, что из всей таблицы стоимостей выбирают клетку с наименьшей стоимостью, для этой ячейки присваиваем меньшее из чисел ai, или bj. Затем, из рассмотрения исключают либо строку, соответствующую поставщику, запасы которого полностью израсходованы, либо столбец, соответствующий потребителю, потребности которого полностью удовлетворены, либо строку и столбец, если израсходованы запасы поставщика и удовлетворены потребности потребителя. Повторяем, пока все запасы не будут распределены, а потребности удовлетворены.

Искомый элемент равен c22=10.

x22 = min(123,117) = 117.

|  |  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- | --- |
| Поставщик | Потребитель | | | | | | Запас |
| 1 | 2 | 3 | 4 | 5 | 6 |
| 1 | 22 | 12 | 16 | 13 | 21 | 11 | 178 |
| 2 | 20 | **10** | 18 | 15 | 17 | 23 | 123-117=6 |
| 3 | 11 | 15 | 21 | 18 | 12 | 21 | 160 |
| 4 | 14 | 20 | 20 | 13 | 23 | 12 | 169 |
| 5 | 13 | 21 | 19 | 10 | 20 | 14 | 110 |
| 6 | 0 | 0 | 0 | 0 | 0 | 0 | 152 |
| Потребность | 153 | 117-117=0 | 141 | 203 | 105 | 173 |  |

Искомый элемент равен c54=10.

x54 = min(110,203) = 110.

|  |  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- | --- |
| Поставщик | Потребитель | | | | | | Запас |
| 1 | 2 | 3 | 4 | 5 | 6 |
| 1 | 22 | 12 | 16 | 13 | 21 | 11 | 178 |
| 2 | 20 | 10|117 | 18 | 15 | 17 | 23 | 6 |
| 3 | 11 | 15 | 21 | 18 | 12 | 21 | 160 |
| 4 | 14 | 20 | 20 | 13 | 23 | 12 | 169 |
| 5 | 13 | 21 | 19 | **10** | 20 | 14 | 110-110=0 |
| 6 | 0 | 0 | 0 | 0 | 0 | 0 | 152 |
| Потребность | 153 | 0 | 141 | 203-110=93 | 105 | 173 |  |

Искомый элемент равен c16=11.

x16 = min(178,173) = 173.

|  |  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- | --- |
| Поставщик | Потребитель | | | | | | Запас |
| 22 | 2 | 3 | 4 | 5 | 6 |
| 1 | 20 | 12 | 16 | 13 | 21 | **11** | 178-173=5 |
| 2 | 11 | 10|117 | 18 | 15 | 17 | 23 | 6 |
| 3 | 14 | 15 | 21 | 18 | 12 | 21 | 160 |
| 4 | 13 | 20 | 20 | 13 | 23 | 12 | 169 |
| 5 | 0 | 21 | 19 | 10|110 | 20 | 14 | 0 |
| 6 | 22 | 0 | 0 | 0 | 0 | 0 | 152 |
| Потребность | 153 | 0 | 141 | 203 | 105 | 173-173=0 |  |

Искомый элемент равен c31=11.

x31 = min(160,153) = 153.

|  |  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- | --- |
| Поставщик | Потребитель | | | | | | Запас |
| 1 | 2 | 3 | 4 | 5 | 6 |
| 1 | 22 | 12 | 16 | 13 | 21 | 11|173 | 5 |
| 2 | 20 | 10|117 | 18 | 15 | 17 | 23 | 6 |
| 3 | **11** | 15 | 21 | 18 | 12 | 21 | 160-153=7 |
| 4 | 14 | 20 | 20 | 13 | 23 | 12 | 169 |
| 5 | 13 | 21 | 19 | 10|110 | 20 | 14 | 0 |
| 6 | 0 | 0 | 0 | 0 | 0 | 0 | 152 |
| Потребность | 153-153=0 | 0 | 141 | 93 | 105 | 0 |  |

Искомый элемент равен c35=12.

x35 = min(7,105) = 7.

|  |  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- | --- |
| Поставщик | Потребитель | | | | | | Запас |
| 1 | 2 | 3 | 4 | 5 | 6 |
| 1 | 22 | 12 | 16 | 13 | 21 | 11|173 | 5 |
| 2 | 20 | 10|117 | 18 | 15 | 17 | 23 | 6 |
| 3 | 11|153 | 15 | 21 | 18 | **12** | 21 | 7-7=0 |
| 4 | 14 | 20 | 20 | 13 | 23 | 12 | 169 |
| 5 | 13 | 21 | 19 | 10|110 | 20 | 14 | 0 |
| 6 | 0 | 0 | 0 | 0 | 0 | 0 | 152 |
| Потребность | 0 | 0 | 141 | 93 | 105-7=98 | 0 |  |

Искомый элемент равен c14=13.

x14 = min(5,93) = 5.

|  |  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- | --- |
| Поставщик | Потребитель | | | | | | Запас |
| 1 | 2 | 3 | 4 | 5 | 6 |
| 1 | 22 | 12 | 16 | **13** | 21 | 11|173 | 5-5=0 |
| 2 | 20 | 10|117 | 18 | 15 | 17 | 23 | 6 |
| 3 | 11|153 | 15 | 21 | 18 | 12|7 | 21 | 0 |
| 4 | 14 | 20 | 20 | 13 | 23 | 12 | 169 |
| 5 | 13 | 21 | 19 | 10|110 | 20 | 14 | 0 |
| 6 | 0 | 0 | 0 | 0 | 0 | 0 | 152 |
| Потребность | 0 | 0 | 141 | m | 98 | 0 |  |

Искомый элемент равен c44=13.

x44 = min(169,88) = 88.

|  |  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- | --- |
| Поставщик | Потребитель | | | | | | Запас |
| 1 | 2 | 3 | 4 | 5 | 6 |
| 1 | 22 | 12 | 16 | 13|5 | 21 | 11|173 | 0 |
| 2 | 20 | 10|117 | 18 | 15 | 17 | 23 | 6 |
| 3 | 11|153 | 15 | 21 | 18 | 12|7 | 21 | 0 |
| 4 | 14 | 20 | 20 | **13** | 23 | 12 | 169-88=81 |
| 5 | 13 | 21 | 19 | 10|110 | 20 | 14 | 0 |
| 6 | 0 | 0 | 0 | 0 | 0 | 0 | 152 |
| Потребность | 0 | 0 | 141 | 88-88=0 | 98 | 0 |  |

Искомый элемент равен c25=17.

x25 = min(6,98) = 6.

|  |  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- | --- |
| Поставщик | Потребитель | | | | | | Запас |
| 1 | 2 | 3 | 4 | 5 | 6 |
| 1 | 22 | 12 | 16 | 13|5 | 21 | 11|173 | 0 |
| 2 | 20 | 10|117 | 18 | 15 | **17** | 23 | 6-6=0 |
| 3 | 11|153 | 15 | 21 | 18 | 12|7 | 21 | 0 |
| 4 | 14 | 20 | 20 | 13|88 | 23 | 12 | 81 |
| 5 | 13 | 21 | 19 | 10|110 | 20 | 14 | 0 |
| 6 | 0 | 0 | 0 | 0 | 0 | 0 | 152 |
| Потребность | 0 | 0 | 141 | 0 | 98-6=92 | 0 |  |

Искомый элемент равен c43=20.

x43 = min(81,141) = 81.

|  |  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- | --- |
| Поставщик | Потребитель | | | | | | Запас |
| 1 | 2 | 3 | 4 | 5 | 6 |
| 1 | 22 | 12 | 16 | 13|5 | 21 | 11|173 | 0 |
| 2 | 20 | 10|117 | 18 | 15 | 17|6 | 23 | 0 |
| 3 | 11|153 | 15 | 21 | 18 | 12|7 | 21 | 0 |
| 4 | 14 | 20 | **20** | 13|88 | 23 | 12 | 81-81=0 |
| 5 | 13 | 21 | 19 | 10|110 | 20 | 14 | 0 |
| 6 | 0 | 0 | 0 | 0 | 0 | 0 | 152 |
| Потребность | 0 | 0 | 141-81=60 | 0 | 92 | 0 |  |

Искомый элемент равен c63=0.

x63 = min(152,60) = 60.

|  |  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- | --- |
| Поставщик | Потребитель | | | | | | Запас |
| 1 | 2 | 3 | 4 | 5 | 6 |
| 1 | 22 | 12 | 16 | 13|5 | 21 | 11|173 | 0 |
| 2 | 20 | 10|117 | 18 | 15 | 17|6 | 23 | 0 |
| 3 | 11|153 | 15 | 21 | 18 | 12|7 | 21 | 0 |
| 4 | 14 | 20 | 20|81 | 13|88 | 23 | 12 | 0 |
| 5 | 13 | 21 | 19 | 10|110 | 20 | 14 | 0 |
| 6 | 0 | 0 | **0** | 0 | 0 | 0 | 152-60=92 |
| Потребность | 0 | 0 | 60-60=0 | 0 | 92 | 0 |  |

Искомый элемент равен c65=0.

x65 = min(92,92) = 92.

|  |  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- | --- |
| Поставщик | Потребитель | | | | | | Запас |
| 1 | 2 | 3 | 4 | 5 | 6 |
| 1 | 22 | 12 | 16 | 13|5 | 21 | 11|173 | 0 |
| 2 | 20 | 10|117 | 18 | 15 | 17|6 | 23 | 0 |
| 3 | 11|153 | 15 | 21 | 18 | 12|7 | 21 | 0 |
| 4 | 14 | 20 | 20|81 | 13|88 | 23 | 12 | 0 |
| 5 | 13 | 21 | 19 | 10|110 | 20 | 14 | 0 |
| 6 | 0 | 0 | 0|60 | 0 | **0** | 0 | 92-92=0 |
| Потребность | 0 | 0 | 0 | 0 | 92-92=0 | 0 |  |

|  |  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- | --- |
| Поставщик | Потребитель | | | | | | Запас |
| 1 | 2 | 3 | 4 | 5 | 6 |
| 1 | 22 | 12 | 16 | 13|5 | 21 | 11|173 | 0 |
| 2 | 20 | 10|117 | 18 | 15 | 17|6 | 23 | 0 |
| 3 | 11|153 | 15 | 21 | 18 | 12|7 | 21 | 0 |
| 4 | 14 | 20 | 20|81 | 13|88 | 23 | 12 | 0 |
| 5 | 13 | 21 | 19 | 10|110 | 20 | 14 | 0 |
| 6 | 0 | 0 | 0|60 | 0 | 0|92 | 0 | 0 |
| Потребность | 0 | 0 | 0 | 0 | 0 | 0 |  |

В результате получен первый опорный план, который является допустимым, так как все грузы из баз вывезены, потребность потребителей удовлетворена, а план соответствует системе ограничений транспортной задачи.

Подсчитаем число базисных переменных. Их должно быть m + n - 1 = 11

Значение целевой функции для этого опорного плана равно:

F(x) = 13\*5 + 11\*173 + 10\*117 + 17\*6 + 11\*153 + 12\*7 + 20\*81 + 13\*88 + 10\*110 + 0\*60 + 0\*92 = 8871

Этап II

*Метод потенциалов*

Каждому поставщику ai ставим в соответствие некоторое число - ui, называемое потенциалом поставщика. Каждому потребителю bj ставим в соответствие некоторое число - vj, называемое потенциалом потребителя. Для базисной ячейки (задействованного маршрута), сумма потенциалов поставщика и потребителя должна быть равна тарифу данного маршрута.

ui + vj = cij

Найдем предварительные потенциалы ui, vj. по занятым клеткам таблицы, полагая, что u1 = 0.

|  |
| --- |
| u1 + v4 = 13; 0 + v4 = 13; v4 = 13  u4 + v4 = 13; 13 + u4 = 13; u4 = 0  u4 + v3 = 20; 0 + v3 = 20; v3 = 20  u6 + v3 = 0; 20 + u6 = 0; u6 = -20  u6 + v5 = 0; -20 + v5 = 0; v5 = 20  u2 + v5 = 17; 20 + u2 = 17; u2 = -3  u2 + v2 = 10; -3 + v2 = 10; v2 = 13  u3 + v5 = 12; 20 + u3 = 12; u3 = -8  u3 + v1 = 11; -8 + v1 = 11; v1 = 19  u5 + v4 = 10; 13 + u5 = 10; u5 = -3  u1 + v6 = 11; 0 + v6 = 11; v6 = 11 |
| |  |  |  |  |  |  |  |  | | --- | --- | --- | --- | --- | --- | --- | --- | | Поставщик | Потребитель | | | | | | U | | b 1 | b 2 | b3 | b 4 | b 5 | b 6 | | a 1 | 22 | 12 | 16 | 13|5 | 21 | 11|173 | u1 = 0 | | a 2 | 20 | 10|117 | 18 | 15 | 17|6 | 23 | u2 = -3 | | a 3 | 11|153 | 15 | 21 | 18 | 12|7 | 21 | u3 = -8 | | a 4 | 14 | 20 | 20|81 | 13|88 | 23 | 12 | u4 = 0 | | a 5 | 13 | 21 | 19 | 10|110 | 20 | 14 | u5 = -3 | | a 6 | 0 | 0 | 0|60 | 0 | 0|92 | 0 | u6 = -20 | | V | v1 = 19 | v2 = 13 | v3 = 20 | v4 = 13 | v5 = 20 | v6 = 11 |  | |

Опорный план не является оптимальным, так как существуют оценки свободных клеток, для которых ui + vj > cij

(1;2): ∆12 = 0 + 13 - 12 = 1 > 0

(1;3): ∆13 = 0 + 20 - 16 = 4 > 0

(4;1): ∆41 = 0 + 19 - 14 = 5 > 0

(5;1): ∆51 = -3 + 19 - 13 = 3 > 0

max(1,4,5,3) = 5

Выбираем максимальную оценку свободной клетки (4;1): 14. Для этого в перспективную клетку (4;1) поставим знак «+», а в остальных вершинах многоугольника чередующиеся знаки «-», «+», «-».

|  |  |  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- | --- | --- |
| Поставщик | Потребитель | | | | | | Запас | |
| 1 | 2 | 3 | 4 | 5 | 6 |
| 1 | 22 | 12 | 16 | 13|5 | 21 | 11|173 | | 0 |
| 2 | 20 | 10|117 | 18 | 15 | 17|6 | 23 | 0 | |
| 3 | 11|153[-] | 15 | 21 | 18 | 12|7[+] | 21 | 0 | |
| 4 | 14[+] | 20 | 20|81[-] | 13|88 | 23 | 12 | 0 | |
| 5 | 13 | 21 | 19 | 10|110 | 20 | 14 | 0 | |
| 6 | 0 | 0 | 0|60[+] | 0 | 0|92[-] | 0 | 0 | |
| Потребность | 0 | 0 | 0 | 0 | 0 | 0 |  | |

Из грузов хij стоящих в минусовых клетках, выбираем наименьшее, т.е. Прибавляем 81 к объемам грузов, стоящих в плюсовых и вычитаем 81 из xij, стоящих в минусовых клетках. В результате получим новый опорный план.

|  |  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- | --- |
| Поставщик | Потребитель | | | | | | Запас |
| 1 | 2 | 3 | 4 | 5 | 6 |
| 1 | 22 | 12 | 16 | 13|5 | 21 | 11|173 | 0 |
| 2 | 20 | 10|117 | 18 | 15 | 17|6 | 23 | 0 |
| 3 | 11|72 | 15 | 21 | 18 | 12|88 | 21 | 0 |
| 4 | 14|81 | 20 | 20 | 13|88 | 23 | 12 | 0 |
| 5 | 13 | 21 | 19 | 10|110 | 20 | 14 | 0 |
| 6 | 0 | 0 | 0|141 | 0 | 0|11 | 0 | 0 |
| Потребность | 0 | 0 | 0 | 0 | 0 | 0 |  |

Проверим оптимальность опорного плана. Найдем предварительные потенциалы ui, vj. по занятым клеткам таблицы, в которых ui + vj = cij, полагая, что u1 = 0.

u1 + v4 = 13; 0 + v4 = 13; v4 = 13

u4 + v4 = 13; 13 + u4 = 13; u4 = 0

u4 + v1 = 14; 0 + v1 = 14; v1 = 14

u3 + v1 = 11; 14 + u3 = 11; u3 = -3

u3 + v5 = 12; -3 + v5 = 12; v5 = 15

u2 + v5 = 17; 15 + u2 = 17; u2 = 2

u2 + v2 = 10; 2 + v2 = 10; v2 = 8

u6 + v5 = 0; 15 + u6 = 0; u6 = -15

u6 + v3 = 0; -15 + v3 = 0; v3 = 15

u5 + v4 = 10; 13 + u5 = 10; u5 = -3

u1 + v6 = 11; 0 + v6 = 11; v6 = 11

|  |  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- | --- |
| Поставщик | Потребитель | | | | | | U |
| b 1 | b 2 | b3 | b 4 | b 5 | b 6 |
| a 1 | 22 | 12 | 16 | 13|5 | 21 | 11|173 | u1 = 0 |
| a 2 | 20 | 10|117 | 18 | 15 | 17|6 | 23 | u2 = 2 |
| a 3 | 11|72 | 15 | 21 | 18 | 12|88 | 21 | u3 = -3 |
| a 4 | 14|81 | 20 | 20 | 13|88 | 23 | 12 | u4 = 0 |
| a 5 | 13 | 21 | 19 | 10|110 | 20 | 14 | u5 = -3 |
| a 6 | 0 | 0 | 0 | 0|141 | 0|11 | 0 | u6 = -15 |
| V | v1 = 14 | v2 = 8 | v3 = 15 | v4 = 13 | v5 = 15 | v6 = 11 |  |

Опорный план является оптимальным, так все оценки свободных клеток удовлетворяют условию ui + vj ≤ cij.

Минимальные затраты составят: F(x) = 13\*5 + 11\*173 + 10\*117 + 17\*6 + 11\*72 + 12\*88 + 14\*81 + 13\*88 + 10\*110 + 0\*141 + 0\*11 = 8466

*Анализ оптимального плана.*

Из 1-го склада необходимо доставить 5 ед. товара 4-му потребителю и 173 ед. товара 6-му потребителю.

Из 2-го склада необходимо доставить 177 ед. товара 2-му потребителю и 6 ед. товара 5-му потребителю.

Из 3-го склада необходимо доставить 72 ед. товара 1-му потребителю и 88 ед. товара 5-му потребителю.

Из 4-го склада необходимо доставить 81 ед. товара 1-му потребителю и 88 ед. товара 4-му потребителю.

Из 5-го склада необходимо доставить 110 ед. товара 4-му потребителю.

Потребность 3-го потребителя остается неудовлетворенной на 141 ед.

Потребность 5-го потребителя остается неудовлетворенной на 11 ед.

Минимальные затраты составят 8466 денежных единиц.

1. **Транспортная задача** — это задача оптимального распределения ресурсов (товаров, услуг и т.п.) от поставщиков к потребителям с минимальными затратами на перевозку.

**Закрытая транспортная задача:**

* Сумма **всех поставок** = Сумма **всех потребностей**.
* То есть весь товар можно полностью распределить.
* Решается стандартными методами без добавления "фиктивных" пунктов.

**Открытая транспортная задача:**

* Сумма **всех поставок** ≠ Сумма **всех потребностей**.
* Решение требует **добавления фиктивного поставщика или потребителя**:
  + Если запасов **больше**, добавляется **фиктивный потребитель**.
  + Если спроса **больше**, добавляется **фиктивный поставщик**.

2. Решение транспортной задачи состоит из **двух этапов**:

***I. Поиск начального опорного плана* (первичное распределение):**

* **Метод северо-западного угла**
* **Метод минимального элемента (минимальных затрат)**
* **Метод потенциалов** (используется в коррекции, но иногда применяют и в построении начального плана)

***II. Оптимизация плана*:**

* **Метод потенциалов** (основной метод оптимизации)
* **Метод циклических улучшений**
* **MODI (Modified Distribution Method)** — суть та же, что метод потенциалов

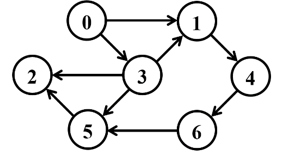
3. Транспортная задача широко используется в:

* **Логистике** — оптимизация перевозок товаров между складами и магазинами
* **Промышленности** — распределение сырья по заводам
* **Энергетике** — распределение электроэнергии между станциями и потребителями
* **Военном деле** — снабжение войск
* **Финансах** — распределение денежных потоков
* **Сельском хозяйстве** — доставка урожая с ферм на переработку

В целом — **везде, где нужно что-то распределить с минимальными затратами**

**Лабораторная работа №6. Алгоритмы на графах**

***Задание 1.*** Ориентированный граф **G** взять в соответствии с вариантом. Представить его в отчете в виде матрицы смежности, матрицы инцидентности, списка смежных вершин.



Матрица смежности:

|  | **0** | **1** | **2** | **3** | **4** | **5** | **6** |
| --- | --- | --- | --- | --- | --- | --- | --- |
| 0 | 0 | 1 | 0 | 1 | 0 | 0 | 0 |
| 1 | 0 | 0 | 0 | 0 | 1 | 0 | 0 |
| 2 | 0 | 0 | 0 | 0 | 0 | 0 | 0 |
| 3 | 0 | 1 | 1 | 0 | 0 | 1 | 0 |
| 4 | 0 | 0 | 0 | 0 | 0 | 0 | 1 |
| 5 | 0 | 0 | 1 | 0 | 0 | 0 | 0 |
| 6 | 0 | 0 | 0 | 0 | 0 | 1 | 0 |

Матрица идентичности:

|  | **e1** | **e2** | **e3** | **e4** | **e5** | **e6** | **e7** | **e8** | **e9** |
| --- | --- | --- | --- | --- | --- | --- | --- | --- | --- |
| 0 | -1 | -1 | 0 | 0 | 0 | 0 | 0 | 0 | 0 |
| 1 | 1 | 0 | -1 | -1 | 0 | 0 | 0 | 0 | 0 |
| 2 | 0 | 0 | 0 | 0 | 1 | 0 | 0 | 1 | 0 |
| 3 | 0 | 1 | 1 | 0 | -1 | -1 | 0 | 0 | 0 |
| 4 | 0 | 0 | 0 | 1 | 0 | 0 | -1 | 0 | 0 |
| 5 | 0 | 0 | 0 | 0 | 0 | 1 | 0 | -1 | 1 |
| 6 | 0 | 0 | 0 | 0 | 0 | 0 | 1 | 0 | -1 |

e1 = 0→1  
e2 = 0→3  
e3 = 3→1  
e4 = 1→4  
e5 = 3→2  
e6 = 3→5  
e7 = 4→6  
e8 = 5→2  
e9 = 6→5

Список смежных вершин:

0: [1, 3]

1: [4]

2: []

3: [1, 2, 5]

4: [6]

5: [2]

6: [5]

***Задание 2.*** Осуществить алгоритмы поиска в ширину и глубину, а также алгоритма топологической сортировки аналогично примерам, рассмотренным на лекциях. Оформить отчет, включив в него **каждый** шаг выполнения алгоритмов.

**1. Поиск в ширину (BFS)**

(начнем из вершины 0)

Алгоритм:

Заводим очередь.

Пишем, что вершина посещена, когда она встала в очередь.

Ход выполнения:

| **Шаг** | **Очередь** | **Посещенные вершины** | **Действие** |
| --- | --- | --- | --- |
| 1 | [0] | {0} | Начинаем с 0 |
| 2 | [1, 3] | {0, 1, 3} | Из 0 добавляем 1 и 3 |
| 3 | [3, 4] | {0, 1, 3, 4} | Из 1 добавляем 4 |
| 4 | [4, 1, 2, 5] | {0, 1, 2, 3, 4, 5} | Из 3 добавляем 1 (уже был), 2 и 5 |
| 5 | [1, 2, 5, 6] | {0, 1, 2, 3, 4, 5, 6} | Из 4 добавляем 6 |
| 6 | [2, 5, 6] | - | 1 уже посещен |
| 7 | [5, 6] | - | 2 уже посещен |
| 8 | [6] | - | 5 уже посещен |
| 9 | [] | - | 6 уже посещен |

Результат обхода: 0 → 1 → 3 → 4 → 2 → 5 → 6

**2. Поиск в глубину (DFS)**

(начинаем с вершины 0)

Алгоритм:

Идём в самую первую доступную вершину, потом от неё дальше, пока можно.

Ход выполнения:

| **Шаг** | **Стек** | **Посещенные вершины** | **Действие** |
| --- | --- | --- | --- |
| 1 | [0] | {0} | Начинаем с 0 |
| 2 | [0, 1] | {0, 1} | Из 0 идём в 1 |
| 3 | [0, 1, 4] | {0, 1, 4} | Из 1 идём в 4 |
| 4 | [0, 1, 4, 6] | {0, 1, 4, 6} | Из 4 идём в 6 |
| 5 | [0, 1, 4, 6, 5] | {0, 1, 4, 5, 6} | Из 6 идём в 5 |
| 6 | [0, 1, 4, 6, 5, 2] | {0, 1, 2, 4, 5, 6} | Из 5 идём в 2 |
| 7 | [0, 1, 4, 6, 5, 2] | - | 2 тупик — назад |
| 8 | [0, 1, 4, 6, 5] | - | 5 тупик — назад |
| 9 | [0, 1, 4, 6] | - | 6 тупик — назад |
| 10 | [0, 1, 4] | - | 4 тупик — назад |
| 11 | [0, 1] | - | 1 тупик — назад |
| 12 | [0, 3] | {0, 1, 2, 3, 4, 5, 6} | Из 0 ещё не обошли 3 |
| 13 | [0, 3] | - | 3 → 1 (уже посещен), 3 → 2 (уже посещен), 3 → 5 (уже посещен) |

Результат обхода: 0 → 1 → 4 → 6 → 5 → 2 → 3

**3. Топологическая сортировка**

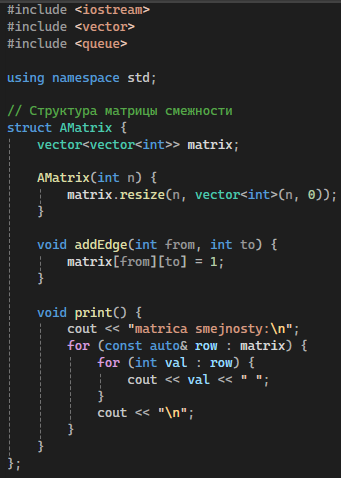
Строим стек обратного выхода:

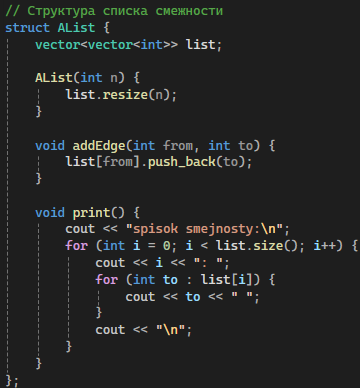
| **Стек** | **Действие** |
| --- | --- |
| 2 | 2 некуда идти |
| 5, 2 | 5 → 2 |
| 6, 5, 2 | 6 → 5 |
| 4, 6, 5, 2 | 4 → 6 |
| 1, 4, 6, 5, 2 | 1 → 4 |
| 3, 1, 4, 6, 5, 2 | 3 → 1, 3 → 2, 3 → 5 |
| 0, 3, 1, 4, 6, 5, 2 | 0 → 1, 0 → 3 |

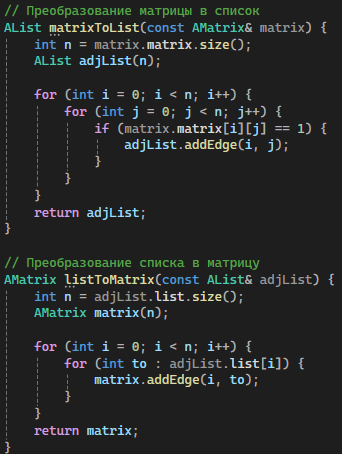
Результат топологической сортировки: 0 → 3 → 1 → 4 → 6 → 5 → 2

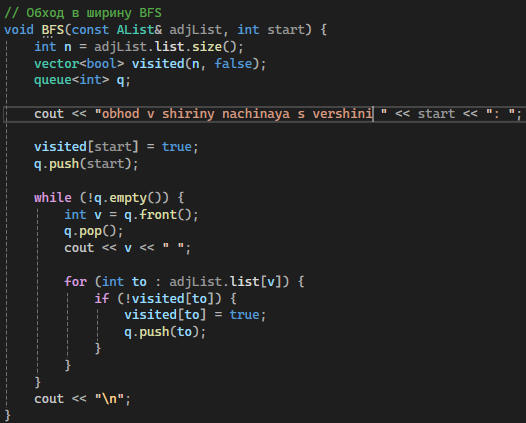
***Задание 3.*** Осуществить программную реализацию алгоритмов на C++. Разработать структуры **AMatrix** и **АList** для представления ориентированного графа матричным и списковым способом. Разработать функции преобразования из одного способа представления в другой. Разработать функцию **BFS** обхода вершин графа, используя метод поиска в ширину. Продемонстрировать работу функции. Копии экрана вставить в отчет.

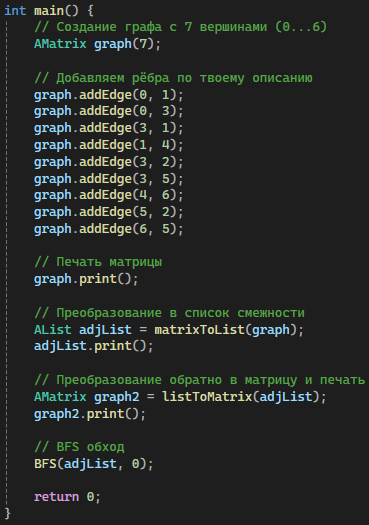
По коду: создали граф в виде **матрицы смежности**, перевели его в **список смежности**, вывели оба представления на экран, делали обход **BFS** начиная с вершины 0, вывели результат на экран.

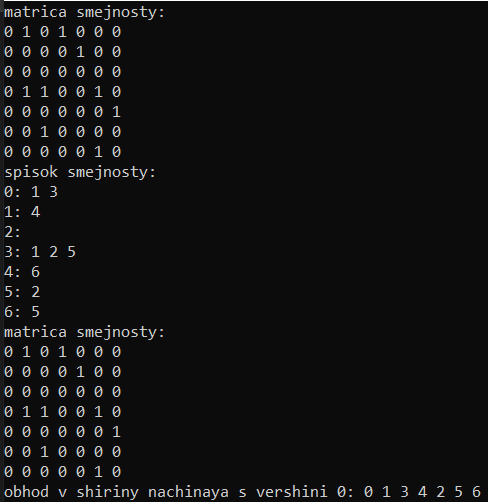




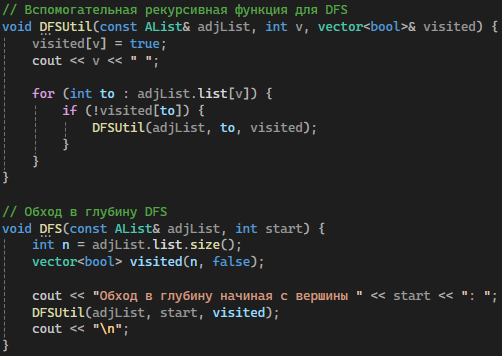








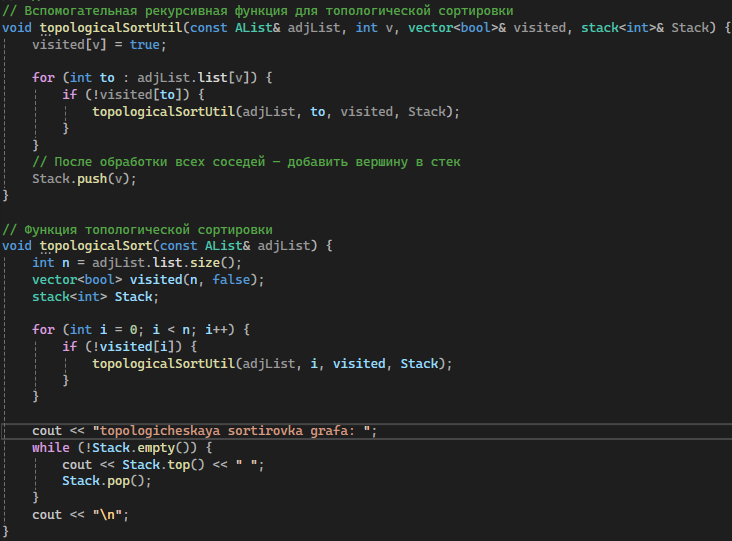
***Задание 4.*** Разработать функцию **DFS** обхода вершин графа, используя метод поиска глубину. Продемонстрировать работу функции. Копии экрана вставить в отчет.







***Задание 5.*** Доработайте функцию **DFS**,для выполнения топологической сортировки графа. Продемонстрировать работу функции. Копии экрана вставить в отчет.







***Задание 6.*** По графу, соответствующему варианту составить минимальное остовное дерево по алгоритму Прима. Шаги построения отразить в отчете.

Веса ребер принять:

W:

W(e0,1)=8; W(e1,0)=5;

W(e0,2)=1; W(e2,0)=3;

W(e0,3)=2; W(e3,0)=8;

W(e1,3)=11; W(e3,1)=4;

W(e1,4)=5; W(e4,1)=3;

W(e2,3)=7; W(e3,2)=9;

W(e2,5)=11; W(e5,2)=10;

W(e4,3)=4; W(e3,4)=1;

W(e4,6)=10; W(e6,4)=2;

W(e5,6)=2; W(e6,5)=6;

W(e5,3)=3; W(e3,5)=6;

W(e6,3)=7; W(e3,6)=9;

**Шаг 1:**

Рассматриваем рёбра от 0:

(0,1) − 8

(0,3) − 2

Минимальный вес 2: ребро (0,3)

Добавляем 3 в дерево.

Дерево: {(0,3)}. Множество вершин: {0,3}

**Шаг 2:**

Рассматриваем рёбра от {0,3}:

(0,1) − 8

(3,1) − 4

(3,2) − 9

(3,5) − 6

Минимальный вес: 4 − (3,1)

Добавляем 1 в дерево.

Дерево: {(0,3), (3,1)}. Множество вершин: {0,1,3}

**Шаг 3:**

Рассматриваем рёбра от {0,1,3}:

(0,1) − (оба выбраны)

(3,1) − (оба выбраны)

(3,2) − 9

(3,5) − 6

(1,4) − 5

Минимальный вес: 5 − (1,4)

Добавляем 4 в дерево.

Дерево: {(0,3), (3,1), (1,4)}. Множество вершин: {0,1,3,4}

**Шаг 4:**

Рассматриваем рёбра от {0,1,3,4}:

(3,2) − 9

(3,5) − 6

(4,6) − 2

Минимальный вес: 2 − (4,6)

Добавляем 6 в дерево.

Дерево: {(0,3), (3,1), (1,4), (4,6)}. Множество вершин: {0,1,3,4,6}

**Шаг 5:**

Рассматриваем рёбра от {0,1,3,4,6}:

(3,2) − 9

(3,5) − 6

(6,5) − 2

Минимальный вес: 2 − (6,5)

Добавляем 5 в дерево.

Дерево: {(0,3), (3,1), (1,4), (4,6), (6,5)}. Множество вершин: {0,1,3,4,5,6}

**Шаг 6:**

Рассматриваем рёбра от {0,1,3,4,5,6}:

(3,2) − 9

(5,2) − 10

Минимальный вес: 9 − (3,2)

Добавляем 2 в дерево.

Дерево: {(0,3), (3,1), (1,4), (4,6), (6,5), (3,2)}. Множество вершин: {0,1,2,3,4,5,6}

Все вершины покрыты.

**Минимальное остовное дерево:**

| **Ребро** | **Вес** |
| --- | --- |
| (0,3) | 2 |
| (3,1) | 4 |
| (1,4) | 5 |
| (4,6) | 2 |
| (6,5) | 2 |
| (3,2) | 9 |

**Суммарный вес остовного дерева:**

2+4+5+2+2+9=242 + 4 + 5 + 2 + 2 + 9 = 242+4+5+2+2+9=24

**Ответ:** Суммарный вес = 24.

**Задание 7.** По графу, соответствующему варианту составить минимальное остовное дерево по алгоритму Крускала. Шаги построения отразить в отчете.

**Шаг 0: сортируем рёбра по весу**

| **Ребро** | **Вес** |
| --- | --- |
| (0,3) | 2 |
| (4,6) | 2 |
| (6,5) | 2 |
| (3,1) | 4 |
| (1,4) | 5 |
| (3,5) | 6 |
| (0,1) | 8 |
| (3,2) | 9 |
| (5,2) | 10 |

**Шаг 1:**

Берём ребро (0,3) с весом 2.

0 и 3 находятся в разных компонентах.

Добавляем ребро (0,3).

Текущее дерево: {(0,3)}

**Шаг 2:**

Берём ребро (4,6) с весом 2.

4 и 6 в разных компонентах.

Добавляем ребро (4,6).

Текущее дерево: {(0,3), (4,6)}

**Шаг 3:**

Берём ребро (6,5) с весом 2.

6 и 5 в разных компонентах.

Добавляем ребро (6,5).

Текущее дерево: {(0,3), (4,6), (6,5)}

**Шаг 4:**

Берём ребро (3,1) с весом 4.

3 и 1 в разных компонентах.

Добавляем ребро (3,1).

Текущее дерево: {(0,3), (4,6), (6,5), (3,1)}

**Шаг 5:**

Берём ребро (1,4) с весом 5.

1 (группа с 0,3,1) и 4 (группа с 4,6,5) в разных компонентах.

Добавляем ребро (1,4).

Текущее дерево: {(0,3), (4,6), (6,5), (3,1), (1,4)}

**Шаг 6:**

Берём ребро (3,5) с весом 6.

3 (в одной группе с 0,1,4,6,5) и 5 (уже в группе с 6) → оба уже связаны через 1,4,6,5.

Значит, (3,5) образует цикл − **не добавляем**.

**Шаг 7:**

Берём ребро (0,1) с весом 8.

0 и 1 уже в одной компоненте − цикл. **Не добавляем**.

**Шаг 8:**

Берём ребро (3,2) с весом 9.

3 (группа с 0,1,4,5,6) и 2 − в разных компонентах.

Добавляем ребро (3,2).

Текущее дерево: {(0,3), (4,6), (6,5), (3,1), (1,4), (3,2)}

Все 7 вершин (0,1,2,3,4,5,6) связаны.

**ИТОГ:**

| **Ребро** | **Вес** |
| --- | --- |
| (0,3) | 2 |
| (4,6) | 2 |
| (6,5) | 2 |
| (3,1) | 4 |
| (1,4) | 5 |
| (3,2) | 9 |

**Суммарный вес:** 2+2+2+4+5+9=24

Совпадает с результатом алгоритма Прима.

**Лабораторная работа 7. Метод ветвей и границ. Задача коммивояжера и методы её решения.**

***Задание 1.* Структурное планирование.**

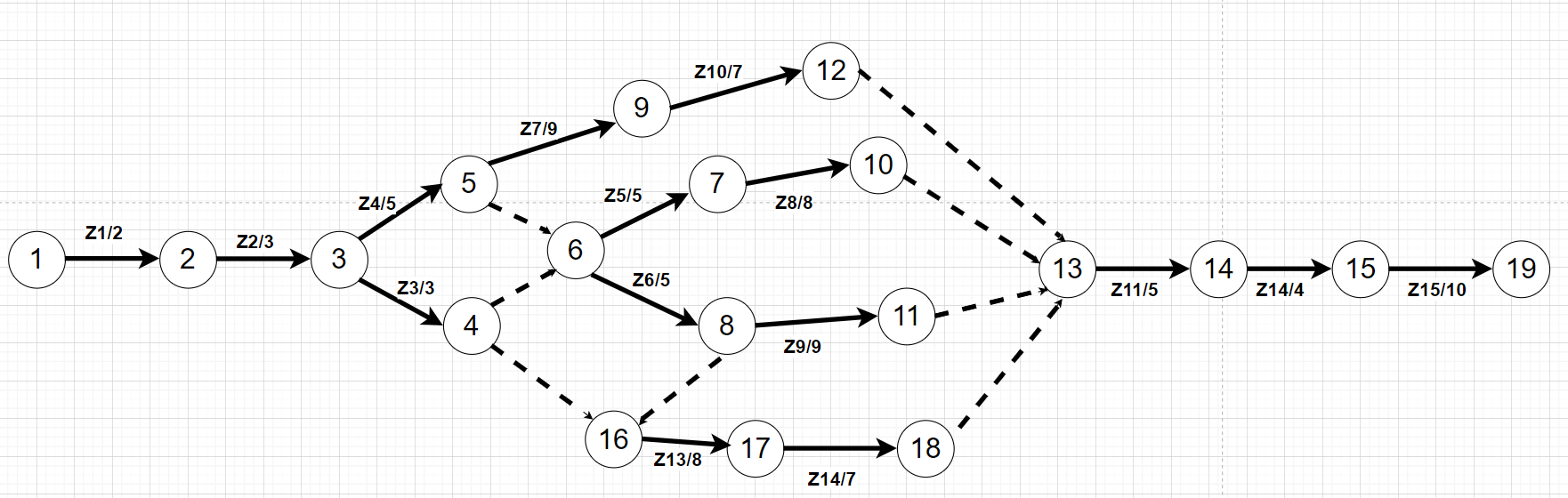
Подумайте и выделите в проекте, согласно вашему варианту не менее 4 этапов работ. Также разбейте полученные этапы на задачи, их количество в совокупности по этапам должно быть не менее 12. Пример оформления задания смотрите в приложении ниже и в лекционном материале по теме.

***Задание 2.*** **Календарное планирование.**

Распределите время, отпущенное на ваш проект согласно вариантам, на выделенные вами этапы. Скорректируйте сформулированные вами задачи, если это необходимо.

Создание компьюерной игры

|  |  |  |  |
| --- | --- | --- | --- |
| Код операции | Наименование операции | Предшествующие операции | t |
| I. АНАЛИЗ | | | | |
| Z1 | Исследование рынка и анализ требований |  | 2 |
| Z2 | Формирование концепции игры | Z1 | 3 |
| Z3 | Планирование проекта и ресурсов | Z2 | 3 |
| II. ПРОЕКТИРОВАНИЕ | | | | |
| Z4 | Проектирование UI/UX | Z2 | 5 |
| Z5 | Разработка механики игры | Z3, Z4 | 5 |
| Z6 | Создание графики и звуковой дизайн | Z3, Z4 | 5 |
| III. КОДИРОВАНИЕ | | | | |
| Z7 | Программирование игровой механики | Z4 | 9 |
| Z8 | Написание интерфейсного кода | Z5 | 8 |
| Z9 | Создание графических и звуковых эффектов | Z6 | 9 |
| IV. ТЕСТИРОВАНИЕ | | | | |
| Z10 | Проведение тестирования игры | Z7 | 7 |
| Z11 | Отладка и устранение ошибок | Z8, Z9, Z10, Z14 | 5 |
| Z12 | Подготовка к выпуску и оптимизация | Z11 | 4 |
|  | V. МАРКЕТИНГ |  |  |
| Z13 | Разработка маркетинговой системы | Z3, Z6 | 8 |
| Z14 | Создание рекламных материалов | Z13 | 7 |
|  | VI. Поддержка после выпуска |  |  |
| Z15 | Поддержка игры после выпуска | Z12 | 10 |



Критический путь в данной сети задач будет следующим:

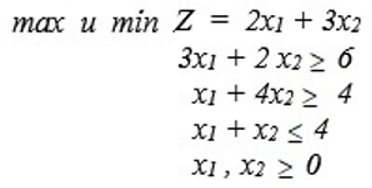
Z1->Z2->Z3->Z5->Z8->Z11->Z12->Z15

Сложив все эти значения, получим общее время выполнения всех задач по критическому пути: 2 + 3 + 3 + 5 + 8 + 5 + 4 + 10 = 40 дней.

**Критический путь:** Z1->Z2->Z3->Z5->Z8->Z11->Z12->Z15 =40 дней

**Лабораторная работа 8. Графический метод решения оптимизационных задач**

**Цель работы:** освоить решение задач графическим методом.



1. Построение прямых:

Нарисуем прямую 2\*x1 + 3\*x2 >= 6. Для этого найдем две точки на этой прямой, подставив значения x1 = 0 и x2 = 0 в уравнение и решив его. Получим точки (0, 3) и (2, 0). Проведем прямую через эти две точки и штрихуем полуплоскость, где 2\*x1 + 3\*x2 >= 6.

Нарисуем прямую x1 + 4\*x2 = 4. Найдем две точки на этой прямой, подставив значения x1 = 0 и x1 = 0 в уравнение и решив его. Получим точки (0, 1) и (4, 0). Проведем прямую через эти две точки и штрихуем полуплоскость,

x1 + 4\*x2 >= 4.

Нарисуем прямую x1 + x2 = 4. Найдем две точки на этой прямой, подставив значения x1 = 0 и x2 = 0 в уравнение и решив его. Получим точки

(0, 4) и (4, 0). Проведем прямую через эти две точки и штрихуем полуплоскость, где x1 + x2 <= 4

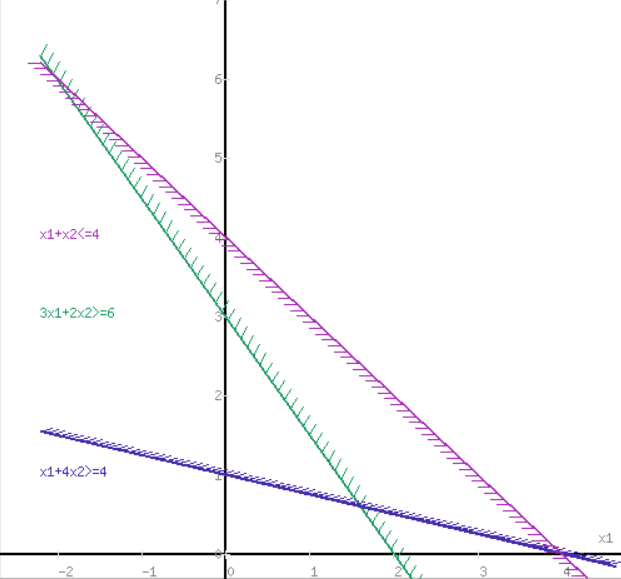


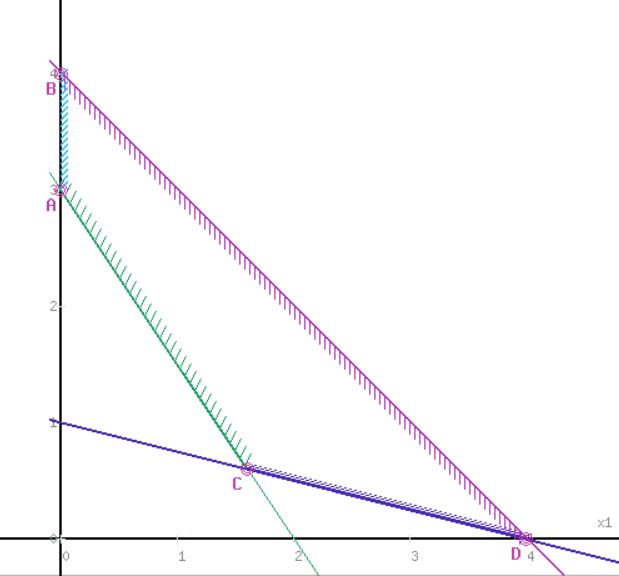
Рисунок 1 – Построенный график всех функций

Рисунок 2 – Полученная полуплоскость

2. Построение целевой функции:

Нарисуем прямую 2\*x1 + 3\*x2 = 0. Для этого найдем две точки на этой прямой, подставив значения x1 = 0 и x2 = 0 в уравнение и решив его. Получим точки (0, 0) и (-1.5, 1). Проведем прямую через эти две точки.

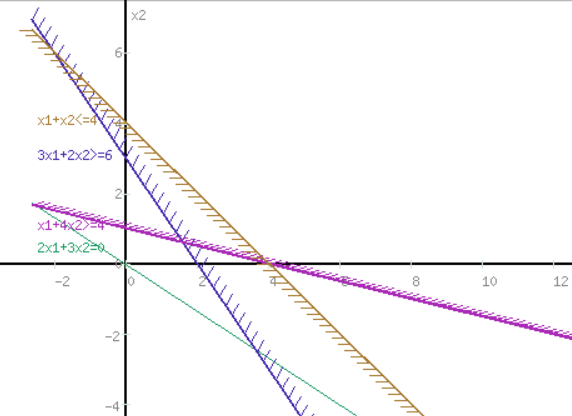


Рисунок 3 – Построение целевой функции

Теперь, чтобы найти максимальное и минимальное значение Z = 2\*x1 + 3\*x2 в области ограничений, мы проверяем каждую вершину этой области (точки пересечения прямых).

Вычислим значение Z для каждой из вершин:

* Для точки (0, 3) => Z = 2\*0 + 3\*3 = 9
* Для точки (2, 0) => Z = 2\*2 + 3\*0 = 4
* Для точки (4, 0) => Z = 2\*4 + 3\*0 = 8
* Для точки (0, 4) => Z = 2\*0 + 3\*4 = 12

Таким образом, максимальное значение функции Z равно 12, а минимальное значение Z равно 4.

1. Поиск в ширину (BFS)

Начинаем с вершины 0.

Шаги:

Очередь: [0]

Посещенные: {0}

Извлекаем 0: добавляем соседей 3, 2

Очередь: [3, 2]

Посещенные: {0, 3, 2}

Извлекаем 3: добавляем 5, 6, 4 (2 уже посещена)

Очередь: [2, 5, 6, 4]

Посещенные: {0, 2, 3, 4, 5, 6}

Извлекаем 2: 5 уже посещена

Извлекаем 5: нет соседей

Извлекаем 6: 5 уже посещена

Извлекаем 4: добавляем 1

Очередь: [1]

Посещенные: {0, 1, 2, 3, 4, 5, 6}

Извлекаем 1: нет соседей

Итоговый порядок BFS:

0 → 3 → 2 → 5 → 6 → 4 → 1

🔍 2. Поиск в глубину (DFS)

Начинаем с вершины 0. Используем стек.

Шаги:

0 → 3

3 → 2

2 → 5

5 (конец)

Назад к 2, 3 → 5 уже посещён

3 → 6

6 → 5 уже посещён

3 → 4

4 → 1

1 (конец)

4 → 6 уже посещён

Итоговый порядок обхода (в момент захода в вершину):

0 → 3 → 2 → 5 → 6 → 4 → 1

📐 3. Топологическая сортировка (DFS + стек)

Шаги (в обратном порядке завершения вызовов):

Начинаем с 0

0 → 3 → 2 → 5

Завершили 5, кладем в стек

Завершили 2, кладем в стек

3 → 6 → 5 уже в стеке

Завершили 6, кладем в стек

3 → 4 → 1

Завершили 1, кладем в стек

4 → 6 уже в стеке

Завершили 4, кладем в стек

Завершили 3, кладем в стек

Завершили 0, кладем в стек

Стек (в порядке добавления):

5, 2, 6, 1, 4, 3, 0

Топологический порядок (из стека в обратном порядке):

0 → 3 → 4 → 1 → 6 → 2 → 5