Учреждение образования

«БЕЛОРУССКИЙ ГОСУДАРСТВЕННЫЙ ТЕХНОЛОГИЧЕСКИЙ УНИВЕРСИТЕТ»

Дисциплина «Математическое программирование»

Отчёт по лабораторным работам

Студент: Турчинович Н. А.

ФИТ 2 курс 2 группа

.

Минск 2024

**Содержание**

[Лабораторная работа 1. Вспомогательные функции 3](#_Toc163562447)

[Лабораторная работа 2. Комбинаторные алгоритмы решения оптимизационных задач 7](#_Toc163562448)

[Лабораторная работа 3. Метод ветвей и границ 18](#_Toc163562449)

[Лабораторная работа 4. Динамическое программирование 23](#_Toc163562450)

# Лабораторная работа 1. Вспомогательные функции

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

***Задание 1.*** Разработайте три функции (start, dget и iget), используя следующие спецификации:

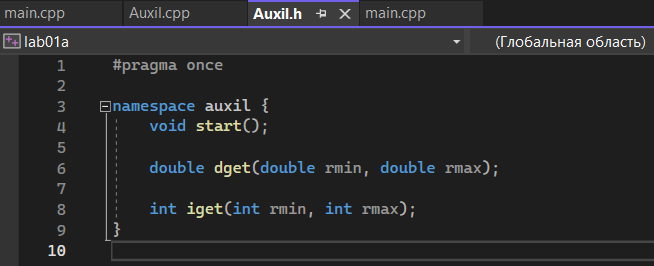


Рисунок 1.1 – Файл Auxio.h

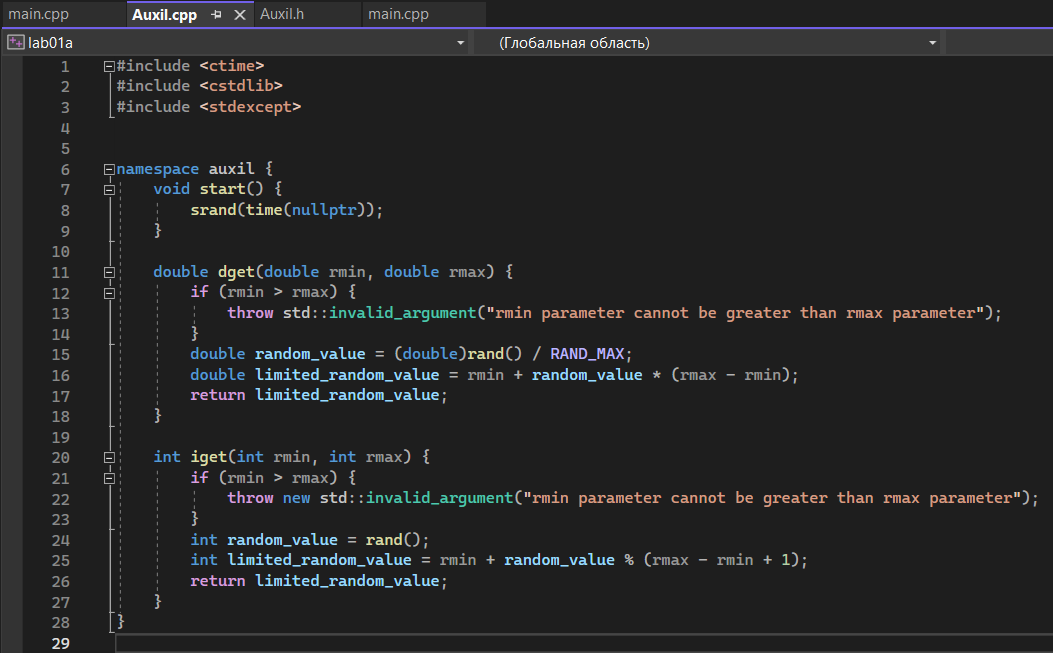


Рисунок 1.1 – Файл Auxio.cpp

***Задание 2***

1. Реализовать пример 2. Представлено на рисунке 1.3.

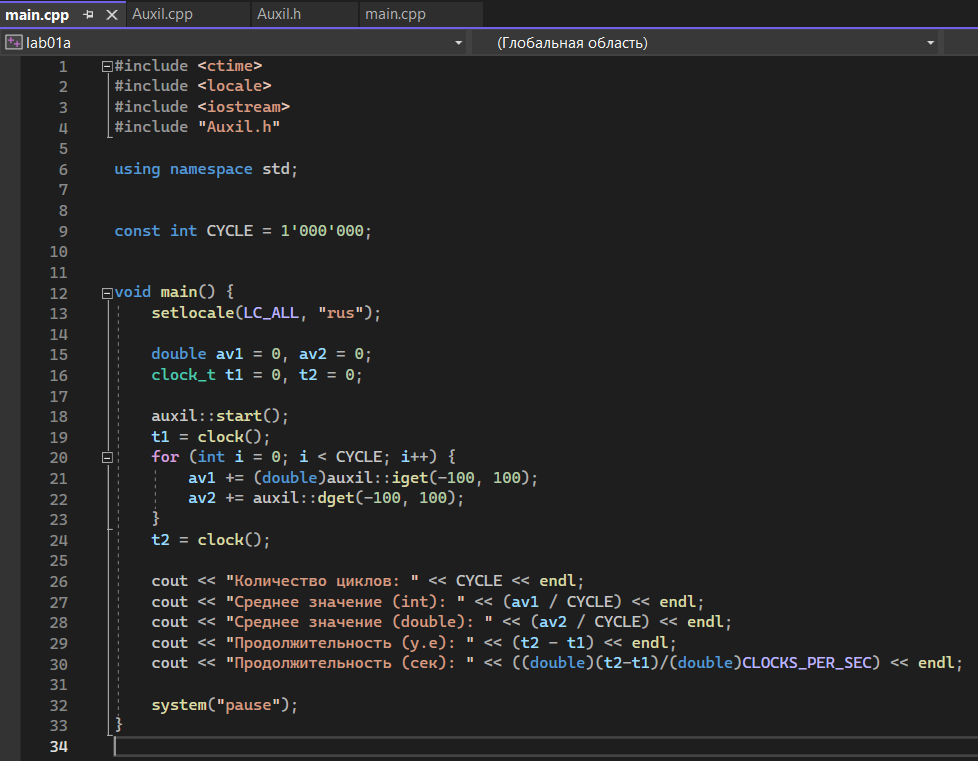


Рисунок 1.3 – Файл Lab1.cpp

1. Для проверки работоспособности разработанных функций и приобретения навыков замера продолжительности процесса вычисления реализуйте программу, приведенную в примере 2. Представлено на рисунке 1.4

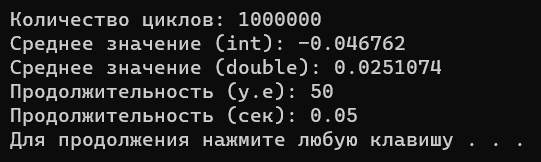


Рисунок 1.4 – Результат замера 1 000 000 циклов

**Задание 3**

Проведите необходимые эксперименты и постройте график зависимости (Excel) продолжительности процесса вычисления от количества циклов в примере 2. Проанализируйте характер зависимости. Результаты измерений и соответствующий график линейной зависимости приведены на рисунке 1.5.

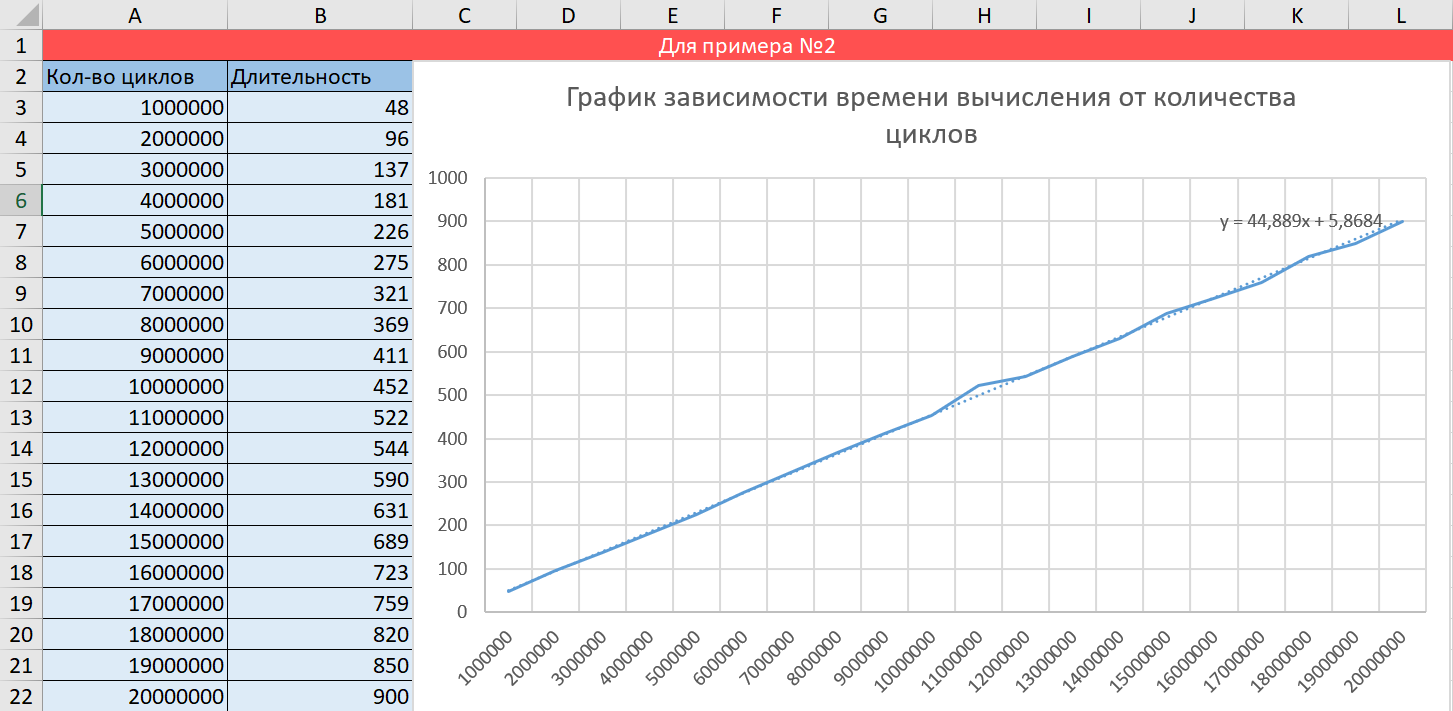


Рисунок 1.5 – Данные и график в Excel

Проведите исследование любого другого рекурсивного алгоритма, например, вычисления факториала или генератора чисел Фибоначчи (прим. – например вычислите каким будет 100-е, 200-е, 300-е и т.д число), и включите в отчет график.

Для примера был взят алгоритм вычисления чисел Фибоначчи. Рабочий вариант кода представлен на рисунке 1.6.

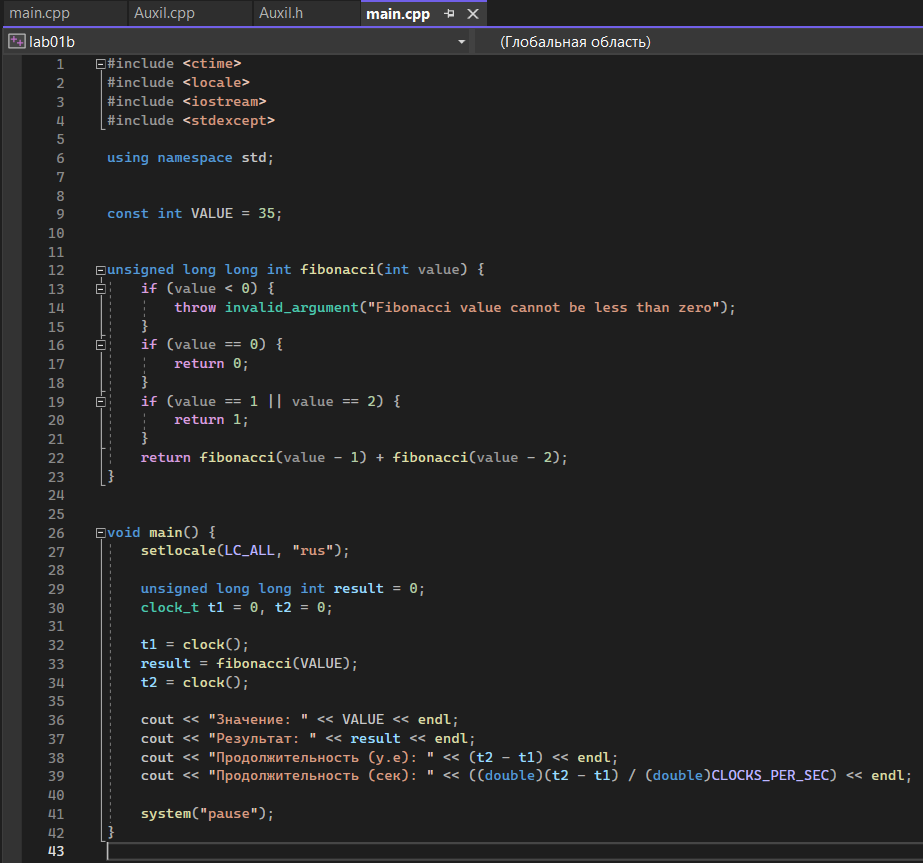


Рисунок 1.6 – Алгоритм вычисления

На рисунке 1.7 представлен результат выполнения программы.

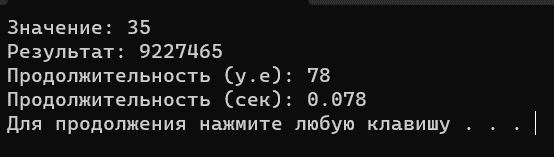


Рисунок 1.7 – Результат выполнения

На рисунке 1.8 представлен график с таблицей значений и времени, потраченного на вычисления чисел от 30 до 50.



Рисунок 1.8 – Таблица чисел

В результате по графику определено, что чем больше число Фибоначчи, тем больше времени нужно для его подсчета в экспоненциальной зависимости.

**Вывод:**

Продолжительность выполнения программы линейно зависит от количества итерация и циклов выполнения, что видно на графике рисунка 1.5.

Продолжительность подсчета чисел Фиббоначчи имеет экспоненциальную зависимость, что видно на графике рисунка 1.8.

# 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 (упрощенную, коммивояжера, об оптимальной загрузке судна и об оптимальной загрузке судна с центровкой.

**Задание 1:** Разработать генератор подмножеств заданного множества. Генератор подмножеств заданного множества будет на рисунке 2.1.

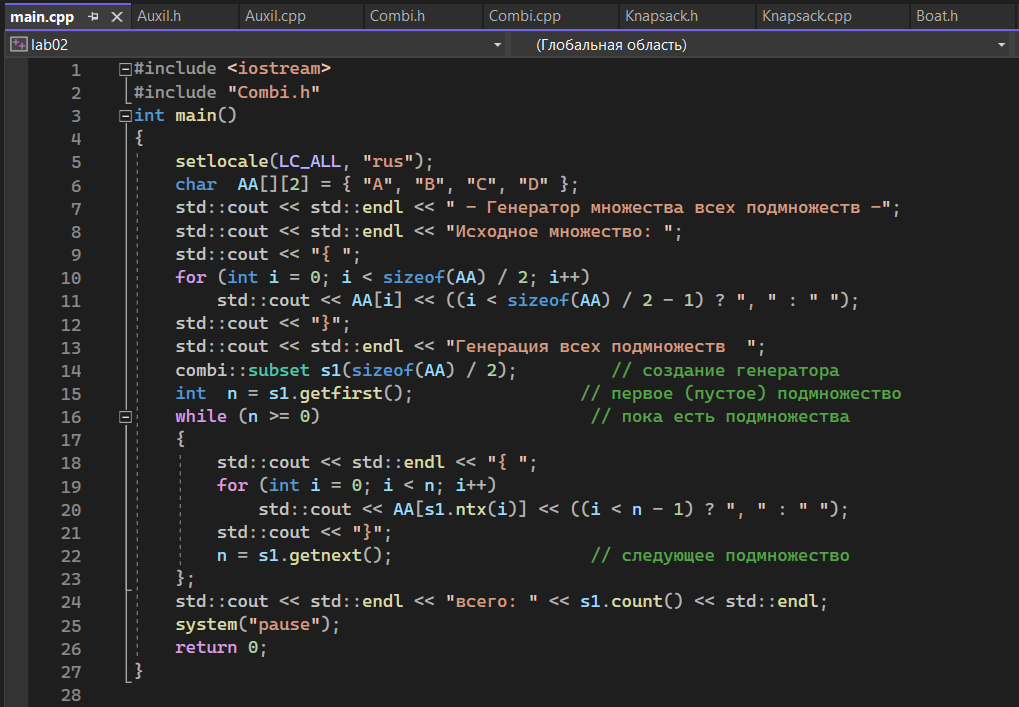


Рисунок 2.1 – Генератор подмножеств

Результат выполнения будет представлен на рисунке 2.2.

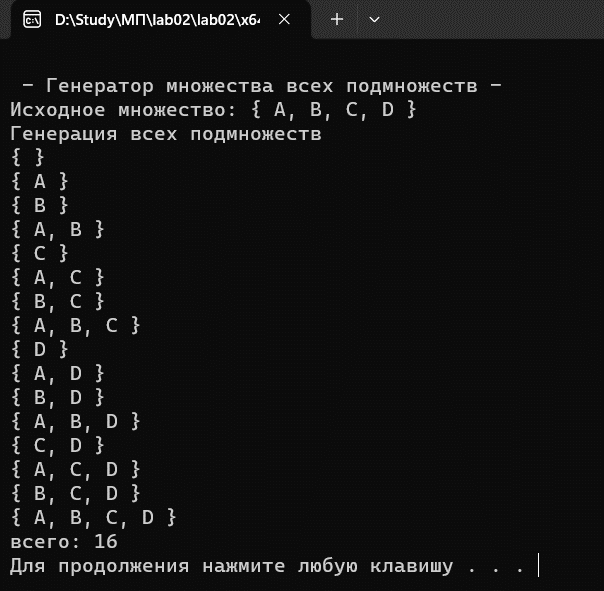


Рисунок 2.2 – Генератор подмножеств заданного множества

Алгоритм будет представлен на рисунке 2.3.



Рисунок 2.3 – Генератор подмножеств заданного множества

**Задание 2:** Разработать генератор сочетаний. Код будет представлен на рисунке 2.4.

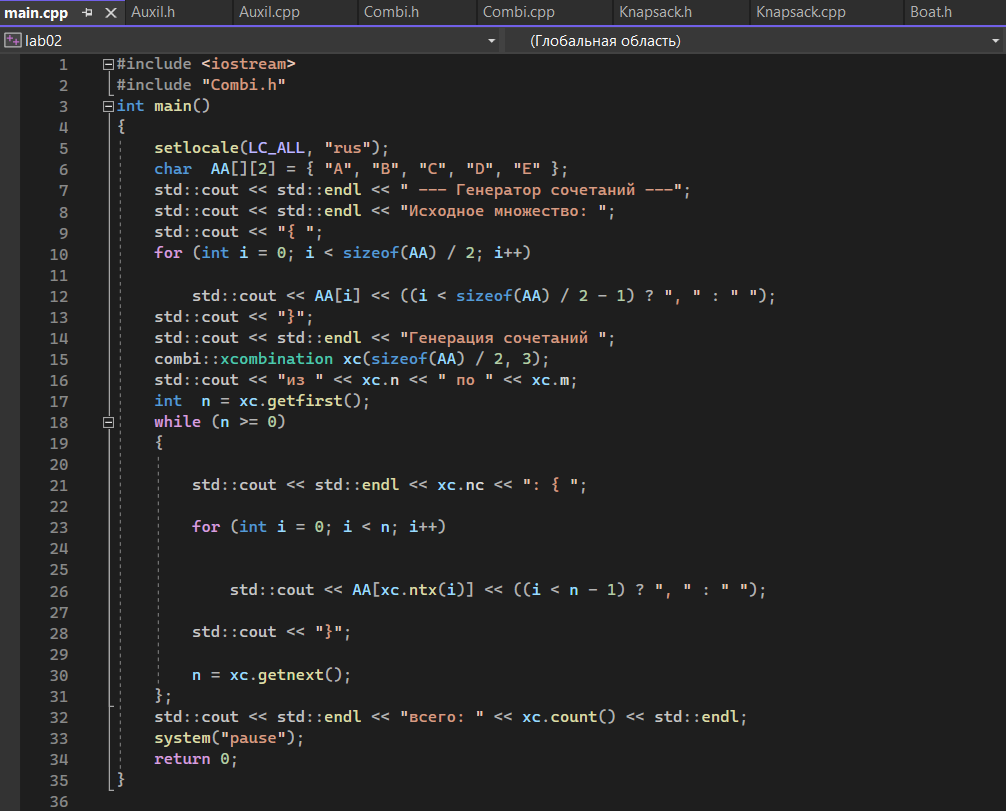


Рисунок 2.4 – Код генератора сочетаний

Результат выполнения будет представлен на рисунке 2.5.

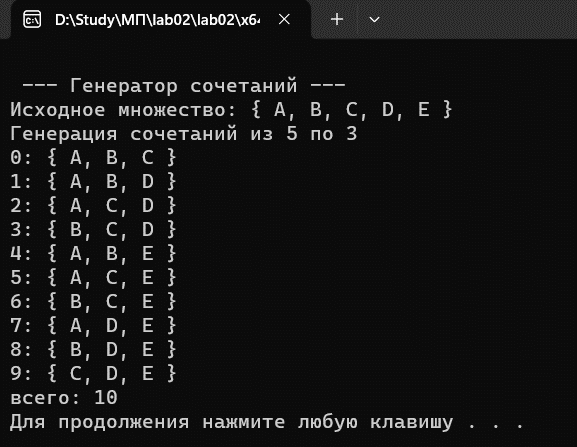


Рисунок 2.5 –Результат генератора сочетаний

Алгоритм будет представлен на рисунке 2.6

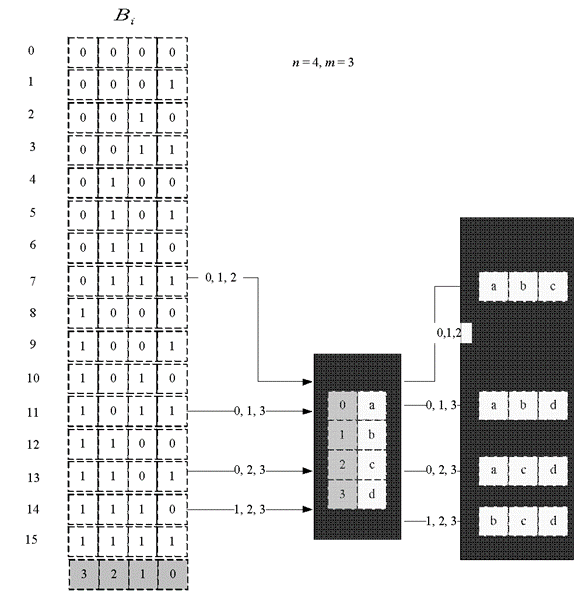


Рисунок 2.6 – Генератор сочетаний

**Задание 3:** Разработать генератор перестановок. Код будет представлен на рисунке 2.7.

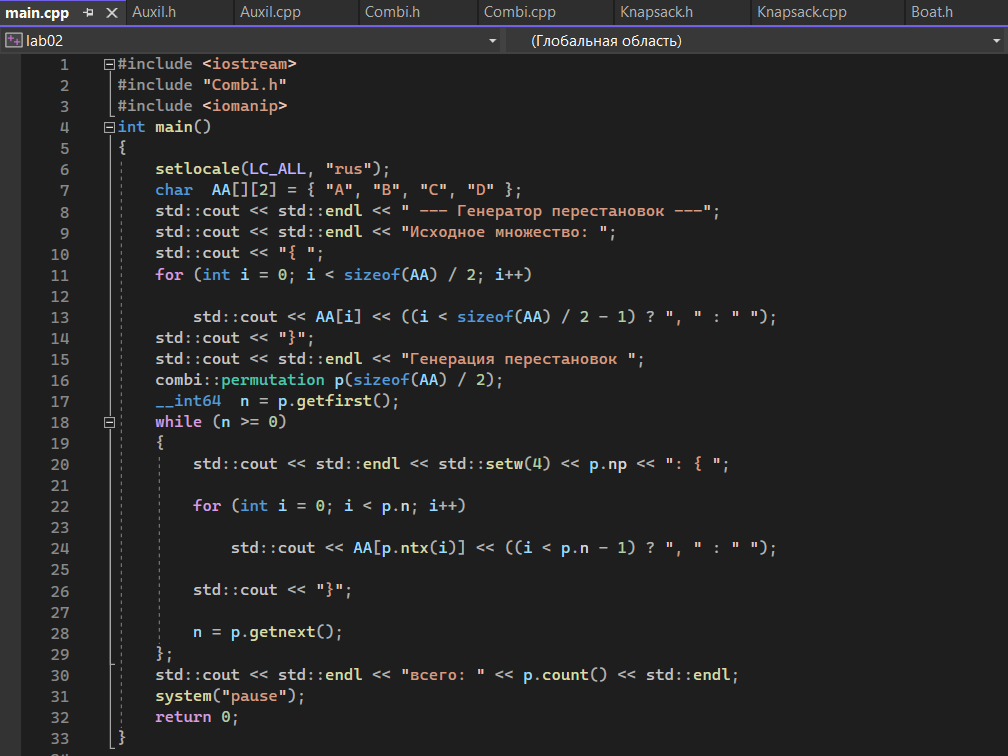


Рисунок 2.7 – Код генератора перестановок

Результат выполнения будет представлен на рисунке 2.8.

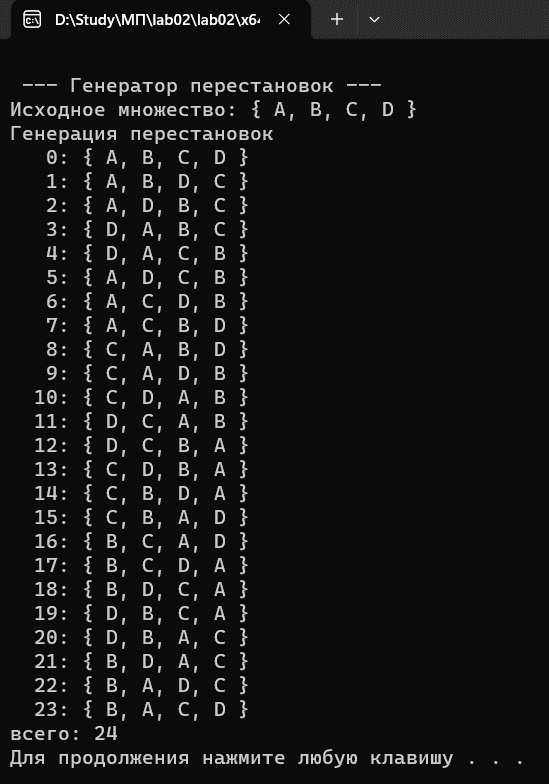


Рисунок 2.8 –Результат генератора перестановок

Алгоритм будет представлен на рисунке 2.9



Рисунок 2.9 – Генератор перестановок

**Задание 4:** Разработать генератор размещений. Код будет представлен на рисунке 2.10.

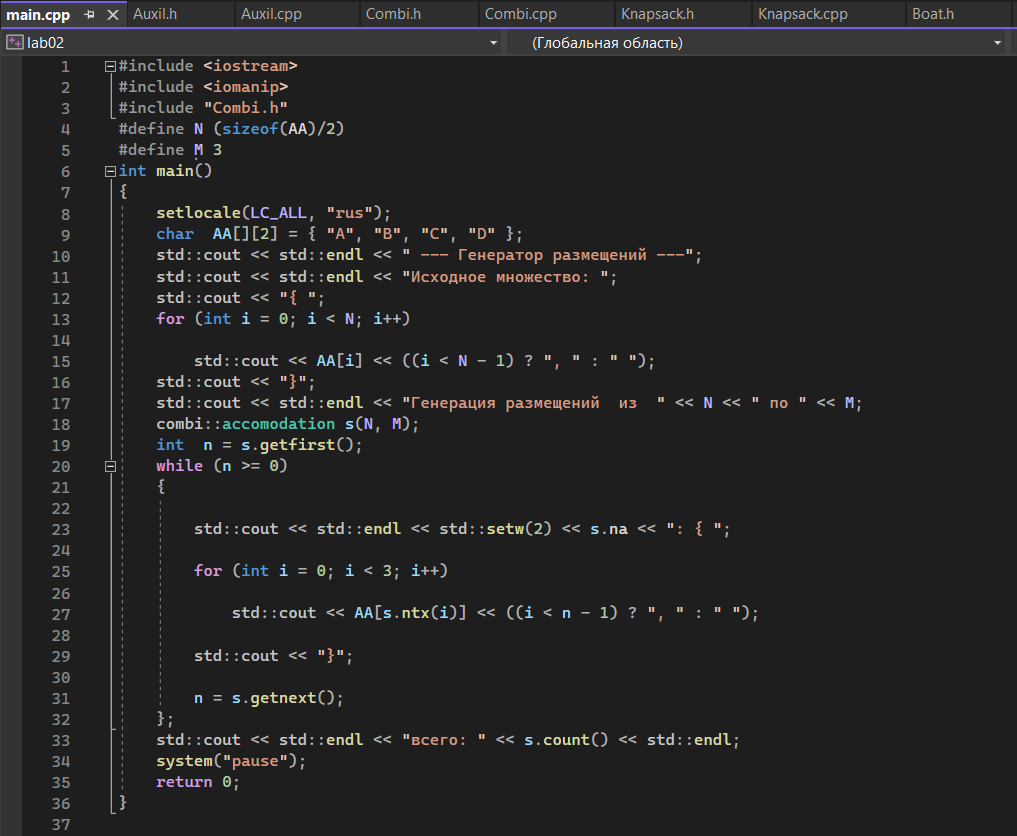


Рисунок 2.10 – Код генератора размещений

Результат выполнения будет представлен на рисунке 2.11.

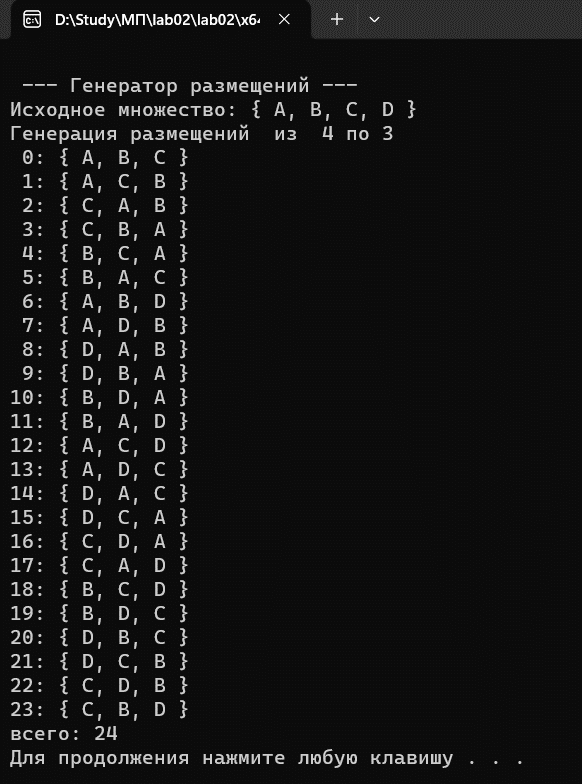


Рисунок 2.11 –Результат генератора размещений

Алгоритм будет представлен на рисунке 2.12.



Рисунок 2.12– Генератор размещений

**Задание 5:** Решить в соответствии с вариантом задачу и результат занести в отчет (Вариант распределяется по списку): (14) упрощенную о рюкзаке (веса предметов и их стоимость сгенерировать случайным образом: вместимость рюкзака 300 кг, веса предметов 10 – 300 кг, предметов 5 – 55 у.е., количество предметов – 18 шт.).

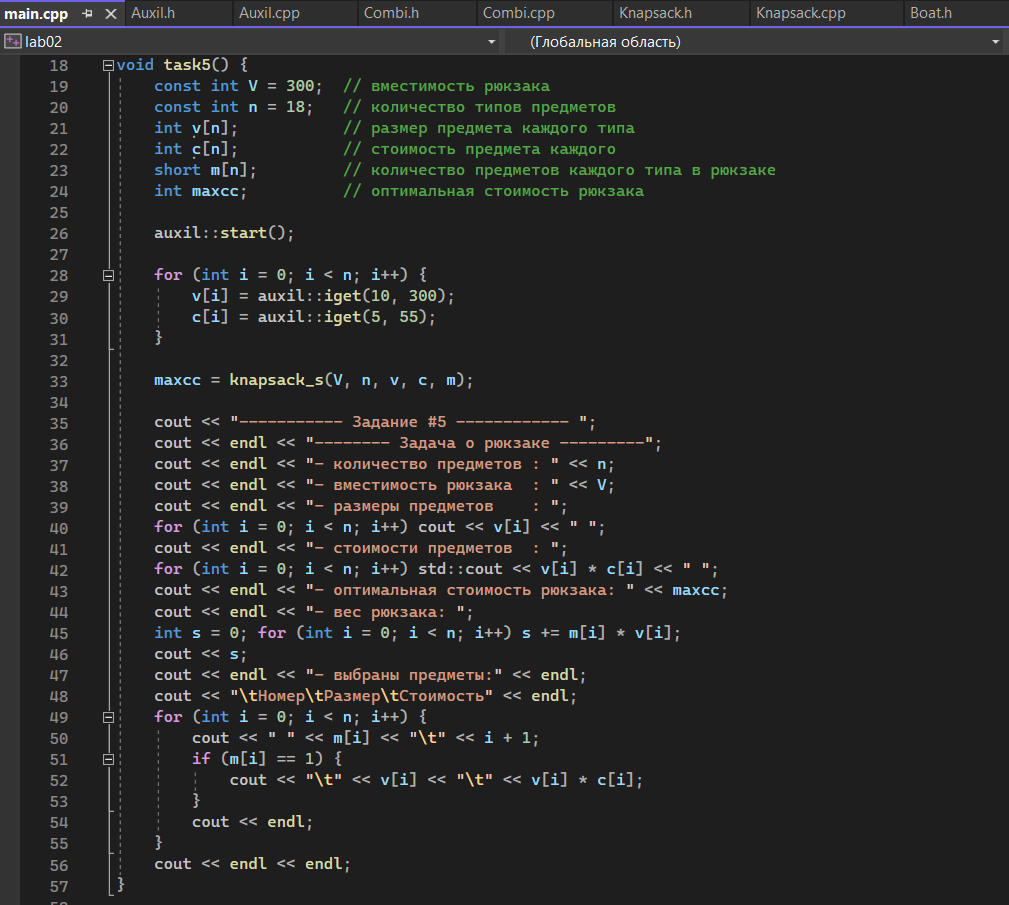


Рисунок 2.13 – Код задания 5

Результат выполнения будет представлен на рисунке 2.14.

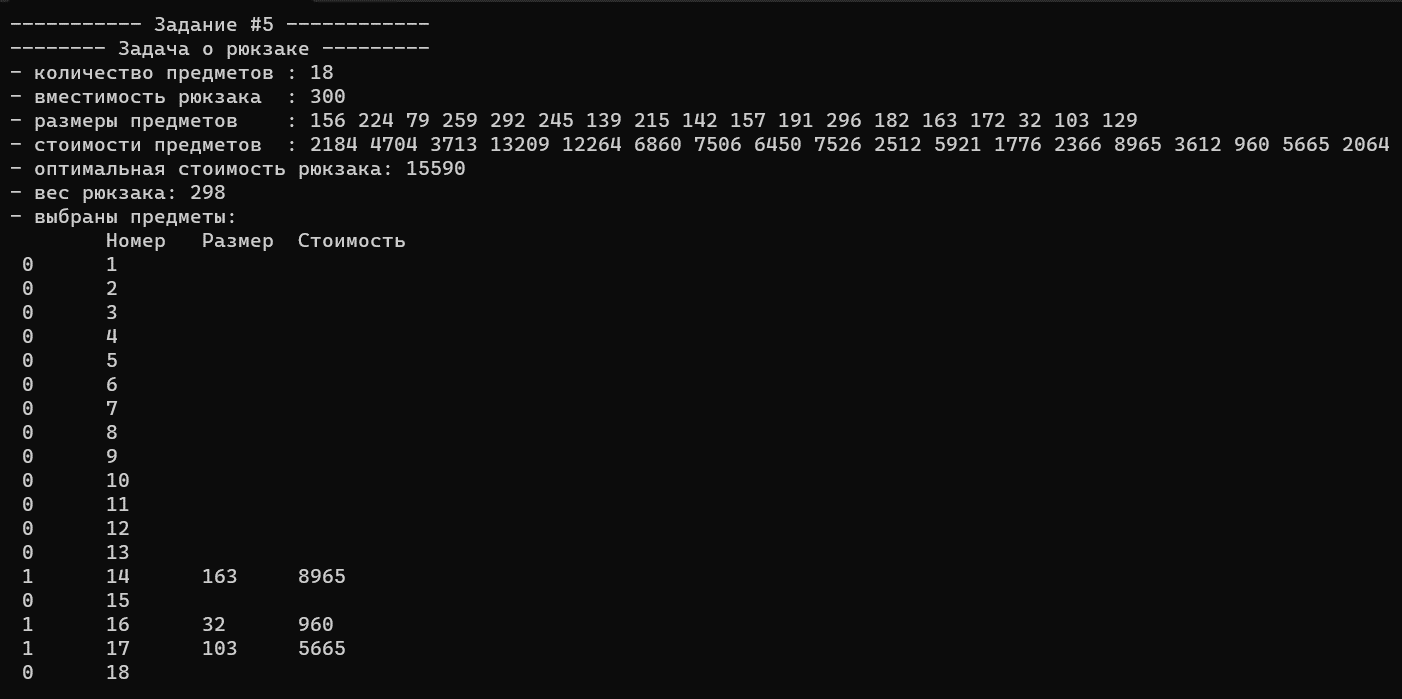


Рисунок 2.14 –Результат задания 5

**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: (14) упрощенную о рюкзаке (количество предметов 12 – 20 шт.)

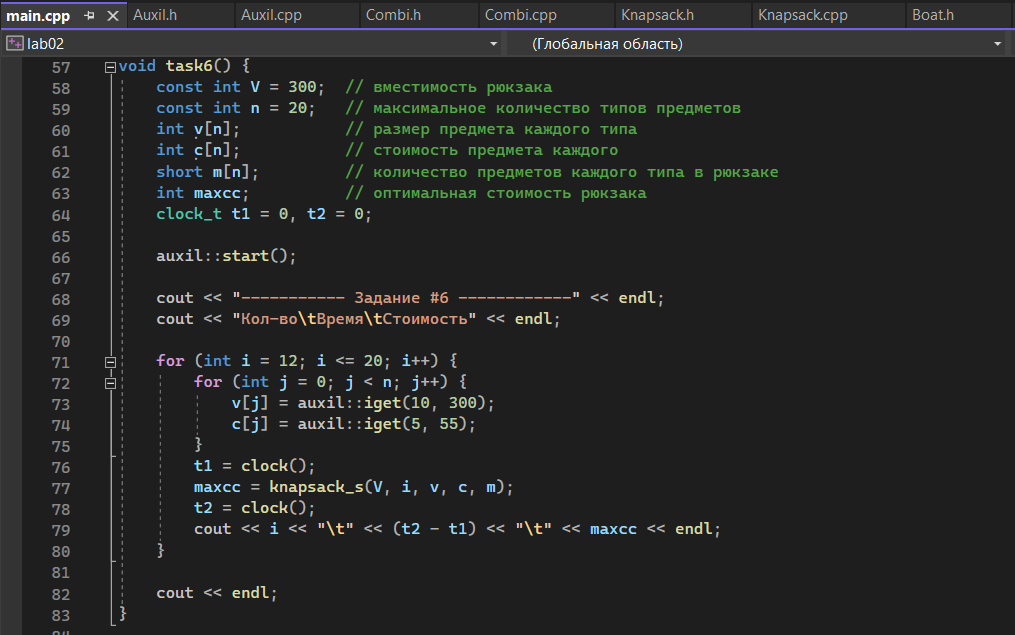


Рисунок 2.15 – Код задания 6

Результат выполнения будет представлен на рисунке 2.16.

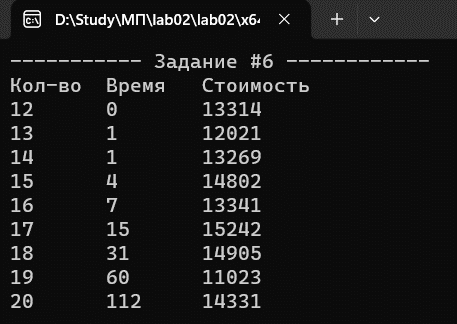


Рисунок 2.16–Результат задания 6

График будет представлен на рисунке 2.17.



Рисунок 2.17 – График

Исходя из данного графика, можно сделать вывод о том, что зависимость времени решения задачи от количества элементов является экспоненциальной. Алгоритм будет представлен на рисунке 2.18.



Рисунок 2.18– Схема решения задачи о рюкзаке

**Вывод:** В ходе работы были приобретены навыки разработки генераторов подмножеств, перестановок, сочетаний и размещений на С++; применены разработанные генераторы для решения упрощенной задачи о рюкзаке, а также построен график зависимости времени решения задачи от количества элементов.

# Лабораторная работа 3. Метод ветвей и границ

**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.

**Задание 1.**

Сформулировать условие задачи коммивояжера с параметром. Для этого:

* принять элементы матрицы расстояний равными:

|  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- |
| **Город** | **1** | **2** | **3** | **4** | **5** |
| **1** |  | 2 \* n | 21 + n |  | n |
| **2** | n |  | 15 + n | 68 - n | 84 - n |
| **3** | 2 + n | 3 \* n |  | 86 | 49 + n |
| **4** | 17 + n | 58 - n | 4 \* n |  | 3 \* n |
| **5** | 93 - n | 66 + n | 52 | 13 + n |  |

где *n* – номер варианта или номер по журналу;

Вариант №14.

|  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- |
| Город | 1 | 2 | 3 | 4 | 5 |
| 1 | INF | 28 | 35 | INF | 14 |
| 2 | 14 | INF | 29 | 54 | 70 |
| 3 | 16 | 42 | INF | 86 | 63 |
| 4 | 31 | 44 | 56 | INF | 42 |
| 5 | 79 | 80 | 52 | 27 | INF |

**Задание 2.**

Решить сформулированную задачу методом ветвей и границ.

Для определения нижней границы множества приведем матрицу по строкам, для чего необходимо в каждой строке матрицы найти минимальный элемент и вычесть его из всех элементов строки. Такую же операцию проводим по столбцам, для чего в каждом столбце находим минимальный элемент.

1. Приведение по строкам.

|  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- |
| Город | 1 | 2 | 3 | 4 | 5 | MIN |
| 1 | INF | 14 | 21 | INF | 0 | 14 |
| 2 | 0 | INF | 15 | 40 | 56 | 14 |
| 3 | 0 | 26 | INF | 70 | 47 | 16 |
| 4 | 0 | 13 | 25 | INF | 11 | 31 |
| 5 | 52 | 53 | 25 | 0 | INF | 27 |

2. Приведение по столбцам.

|  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- |
| Город | 1 | 2 | 3 | 4 | 5 | MIN |
| 1 | INF | 1 | 6 | INF | 0 | 14 |
| 2 | 0 | INF | 0 | 40 | 56 | 14 |
| 3 | 0 | 13 | INF | 70 | 47 | 16 |
| 4 | 0 | 0 | 10 | INF | 11 | 31 |
| 5 | 52 | 40 | 10 | 0 | INF | 27 |
| MIN | 0 | 13 | 15 | 0 | 0 |  |

3. Определение нижней границы.

После вычитания минимальных элементов получаем полностью приведенную матрицу, где величины MIN называются **константами приведения**. Сумма констант приведения определяет нижнюю границу H = 14 + 14 + 16 + 31 + 27 + 0 + 13 + 15 + 0 + 0 = 130.

|  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- |
| Город | 1 | 2 | 3 | 4 | 5 | MIN |
| 1 | INF | 1 | 6 | INF | 0 | 14 |
| 2 | 0 | INF | 0 | 40 | 56 | 14 |
| 3 | 0 | 13 | INF | 70 | 47 | 16 |
| 4 | 0 | 0 | 10 | INF | 11 | 31 |
| 5 | 52 | 40 | 10 | 0 | INF | 27 |
| MIN | 0 | 13 | 15 | 0 | 0 | 130 |

Полностью приведенная матрица:

|  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- |
| Город | 1 | 2 | 3 | 4 | 5 |
| 1 | INF | 1 | 6 | INF | 0 |
| 2 | 0 | INF | 0 | 40 | 56 |
| 3 | 0 | 13 | INF | 70 | 47 |
| 4 | 0 | 0 | 10 | INF | 11 |
| 5 | 52 | 40 | 10 | 0 | INF |

4. Матрица 5х5

**Определяем ребро ветвления** и разобьем все множество маршрутов относительно этого ребра на два подмножества. С этой целью для всех клеток матрицы с нулевыми элементами заменяем поочередно нули на бесконечность и определяем для них сумму образовавшихся констант приведения, они приведены в скобках.

|  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- |
| Город | 1 | 2 | 3 | 4 | 5 |
| 1 | INF | 1 | 6 | INF | 0 (12) |
| 2 | 0 (0) | INF | 0 (6) | 40 | 56 |
| 3 | 0 (13) | 13 | INF | 70 | 47 |
| 4 | 0 (0) | 0 (1) | 10 | INF | 11 |
| 5 | 52 | 40 | 10 | 0 (50) | INF |

Наибольшая сумма констант приведения равна 50 для ребра (5,4), множество разбивается на два подмножества (5,4) и (5\*,4\*).

**Исключение ребра** (5,4) проводим путем замены элемента d54 = 0 на бесконечность, после чего осуществляем очередное приведение матрицы расстояний для образовавшегося подмножества (5\*,4\*), в результате получим приведенную матрицу. Нижняя граница гамильтоновых циклов этого подмножества: H(5\*,4\*) = 130 + 50 = 180.

**Включение ребра** (5,4) проводится путем исключения всех элементов 5-ой строки и 4-го столбца, в которой элемент d45 заменяем на бесконечность, для исключения образования негамильтонова цикла.

В результате получим другую сокращенную матрицу (4 x 4), которая подлежит операции приведения. В итоге получаем приведенную матрицу. Сумма констант приведения сокращенной матрицы 0. Нижняя граница подмножества (5,4) равна: H(5,4) = 130 + 0 = 130 ≤ 180. Поскольку нижняя граница этого подмножества (5,4) меньше, чем подмножества (5\*,4\*), то ребро (5,4) включаем в маршрут с новой границей H = 130.

|  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- |
| Город | 1 | 2 | 3 | 4 | 5 | MIN |
| 1 | INF | 1 | 6 | INF | 0 | 0 |
| 2 | 0 | INF | 0 | 40 | 56 | 0 |
| 3 | 0 | 13 | INF | 70 | 47 | 0 |
| 4 | 0 | 0 | 10 | INF | INF | 0 |
| 5 | 52 | 40 | 10 | 0 | INF |  |
| MIN | 0 | 0 | 0 |  | 0 | 0 |

Полностью приведенная матрица:

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
| Город | 1 | 2 | 3 | 5 |
| 1 | INF | 1 | 6 | 0 |
| 2 | 0 | INF | 0 | 56 |
| 3 | 0 | 13 | INF | 47 |
| 4 | 0 | 0 | 10 | INF |

Аналогичные действия повторяются для матриц размеров 4x4 и 5x5.

5. Матрица 4x4

Определение ребра ветвления.

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
| Город | 1 | 2 | 3 | 5 |
| 1 | INF | 1 | 6 | 0 (48) |
| 2 | 0 (0) | INF | 0 (6) | 56 |
| 3 | 0 (13) | 13 | INF | 47 |
| 4 | 0 (0) | 0 (1) | 10 | INF |

Включение ребра (1, 5).

|  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- |
| Город | 1 | 2 | 3 | 5 | MIN |
| 1 | INF | 1 | 6 | 0 |  |
| 2 | 0 | INF | 0 | 56 | 0 |
| 3 | 0 | 13 | INF | 47 | 0 |
| 4 | 0 | 0 | 10 | INF | 0 |
| MIN | 0 | 0 | 0 |  | 0 |

Полностью приведенная матрица:

|  |  |  |  |
| --- | --- | --- | --- |
| Город | 1 | 2 | 3 |
| 2 | 0 | INF | 0 |
| 3 | 0 | 13 | INF |
| 4 | 0 | 0 | 10 |

6. Матрица 3x3.

Определение ребра ветвления.

|  |  |  |  |
| --- | --- | --- | --- |
| Город | 1 | 2 | 3 |
| 2 | 0 (0) | INF | 0 (10) |
| 3 | 0 (13) | 13 | INF |
| 4 | 0 (0) | 0 (13) | 10 |

Включение ребра (4, 2).

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
| Город | 1 | 2 | 3 | MIN |
| 2 | 0 | INF | 0 | 0 |
| 3 | 0 | 13 | INF | 0 |
| 4 | 0 | 0 | 10 |  |
| MIN | 0 |  | 0 | 0 |

Полностью приведенная матрица:

|  |  |  |
| --- | --- | --- |
| Город | 1 | 3 |
| 2 | 0 | 0 |
| 3 | 0 | INF |

7. Матрица 2x2.

В соответствии с этой матрицей включаем в гамильтонов маршрут ребра (2, 3) и(3, 1).

|  |  |  |
| --- | --- | --- |
| Город | 1 | 3 |
| 2 | 0 | 0 |
| 3 | 0 | INF |

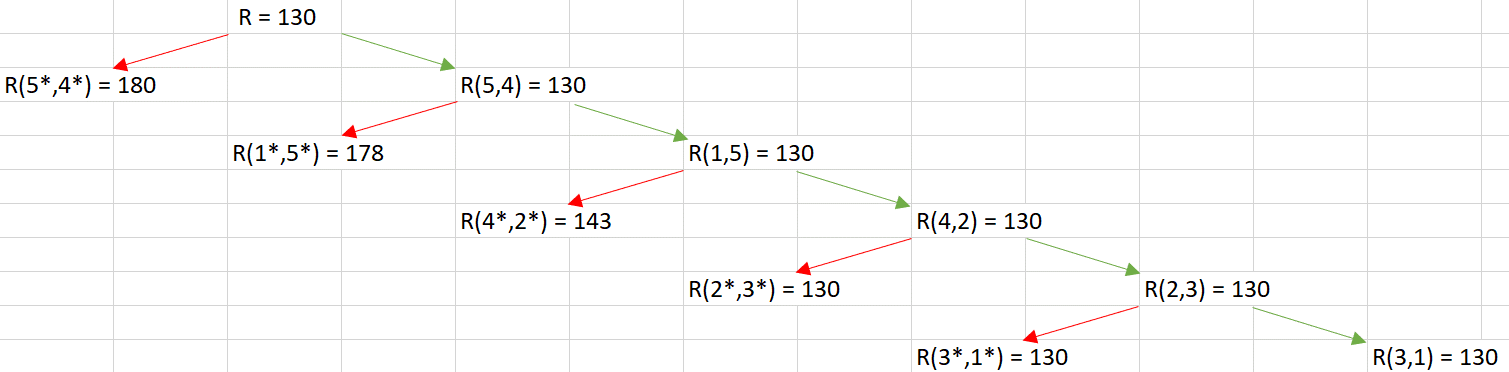
8. Длина маршрута.

Длина маршрута равна F(∞) = 130.

|  |  |
| --- | --- |
| Отрезок | Длина |
| 5 -> 4 | 27 |
| 1 -> 5 | 14 |
| 4 -> 2 | 44 |
| 2 -> 3 | 29 |
| 3 -> 1 | 16 |
| Всего | 130 |

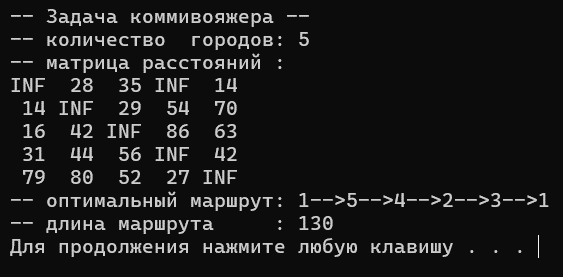
9. Дерево ветвлений.

В результате по дереву ветвлений гамильтонов цикл образуют ребра:  
(5, 4), (1, 5), (4, 2), (2, 3), (3, 1). Оптимальный маршрут 1->5->4->2->3->1.



**Задание 3.**

Вывод программы.



**Вывод**: решение, полученное при помощи генератора перестановок, совпадает с решением, полученным при использовании метода ветвей и границ.

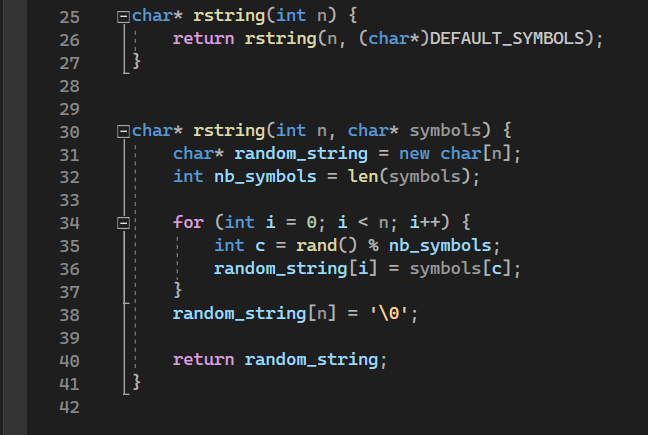
# Лабораторная работа 4. Динамическое программирование

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

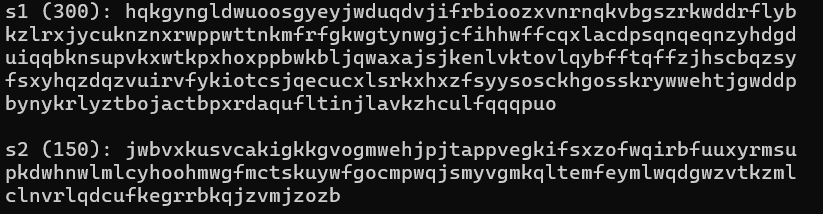
**Задание 1.**

На языке С++ сгенерировать случайным образом строку букв латинского алфавита S1 длиной 300 символов и S2 длиной 200.

Код генерации случайных строк заданной длины:



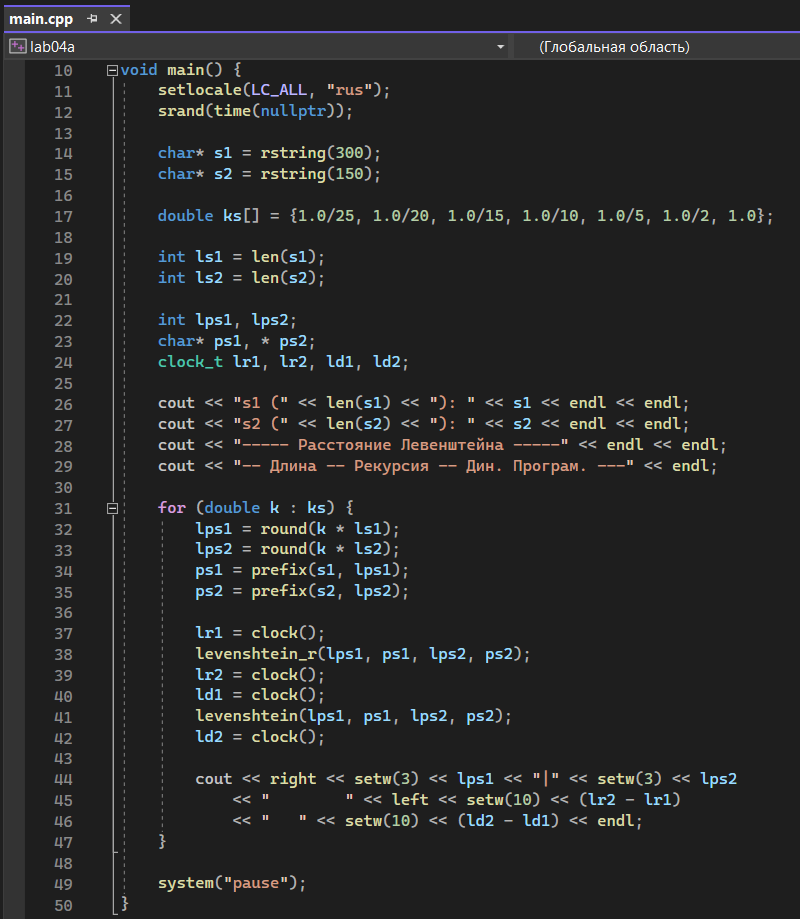
Пример генерации случайных строк заданной длины:



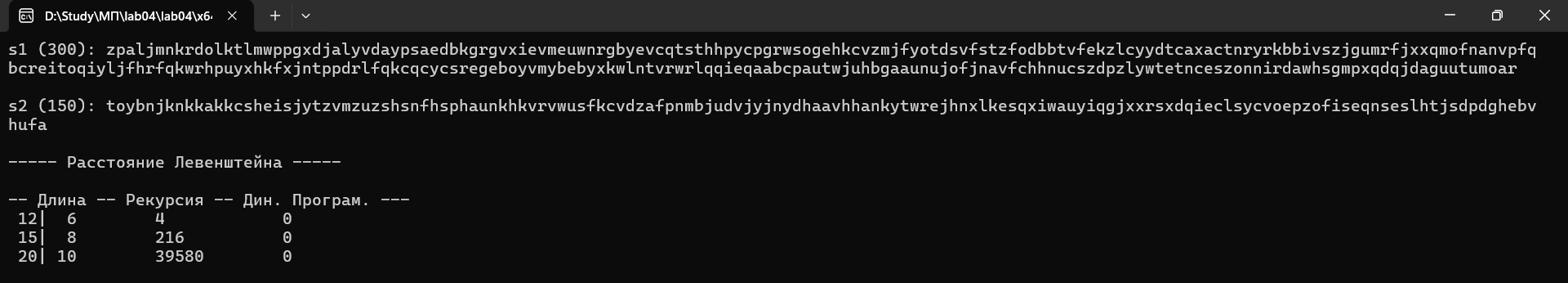
**Задание 2.**

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).

Код программы:



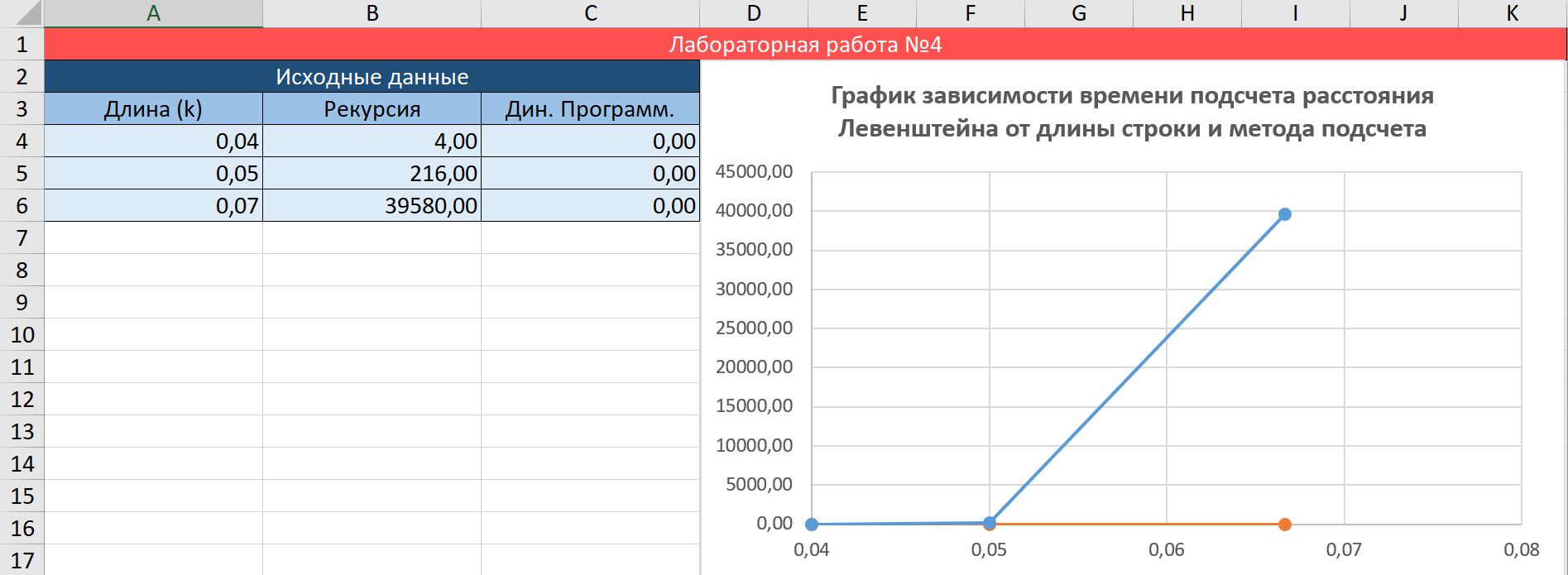
Окно вывода:



**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).

График зависимости времени вычисления от длин строк:



Исходя из графика зависимости видно, что метод рекурсии выполняется значительно медленнее метода динамического программирования, на большей длине строк этот эффект проявляется сильнее.

**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).

Первое слово: Сан; Второе слово: Сонар (Вариант 14).

* + - 1. L(“сан”, “сонар”) =
      2. L(“са”, “сонар”) =
      3. L(“сан”, “сона”) =
      4. L(“са”, “сона”) =
      5. L(“с”, “сонар”) =

L(“”, “сонар”) = 5

L(“”, “сона”) = 4

* + - 1. L(“с”, “сона”) =

L(“”, “сона”) = 4

L(“”, “сон”) = 3

* + - 1. L(“сан”, “сон”) =
      2. L(“са”, “сон”) =
      3. L(“сан”, “со”) =
      4. L(“сан”, “с”) =

L(“сан”, “”) = 3

L(“са”, “”) = 2

* + - 1. L(“с”, “сон”) =

L(“”, “сон”) = 3

L(“”, “со”) = 2

* + - 1. L(“са”, “со”) =

L(“с”, “с”) = 0

* + - 1. L(“с”, “со”) =

L(“”, “со”) = 2

L(“с”, “с”) = 0

L(“”, “с”) = 1

* + - 1. L(“са”, “с”) =

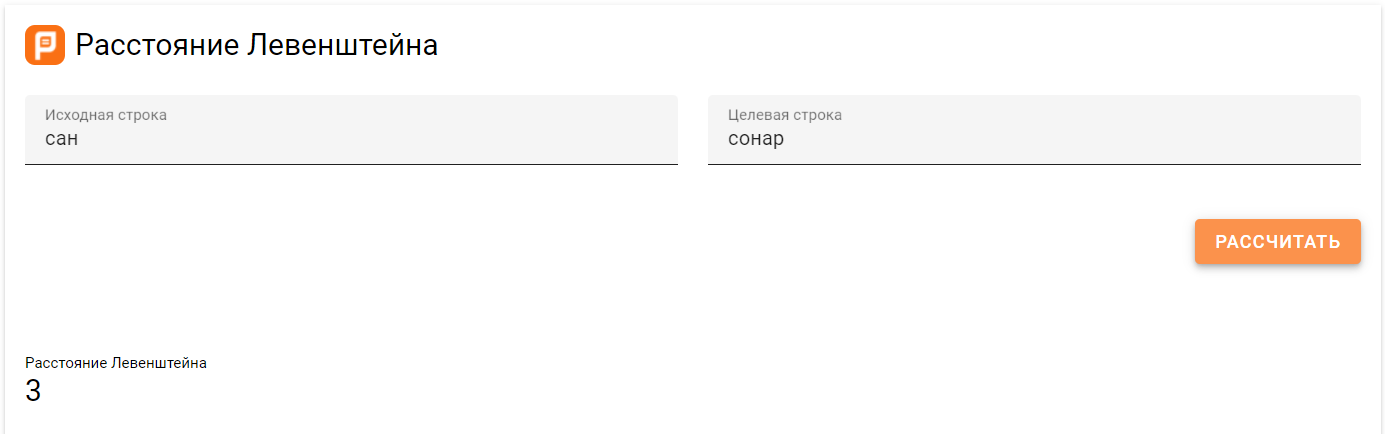
L(“с”, “с”) = 0

L(“са”, “”) = 2

L(“с”, “”) = 1

* + - 1. L(“са”, “с”) = min(1, 3, 2) = 1
      2. L(“с”, “со”) = min(3, 1, 2) = 1
      3. L(“са”, “со”) = min(2, 2, 1) = 1
      4. L(“с”, “сон”) = min(4, 2, 3) = 2
      5. L(“сан”, “с”) = min(2, 4, 3) = 2
      6. L(“сан”, “со”) = min(2, 3, 2) = 2
      7. L(“са”, “сон”) = min(3, 2, 2) = 2
      8. L(“сан”, “сон”) = min(3, 3, 1) = 1
      9. L(“с”, “сона”) = min(5, 3, 4) = 3
      10. L(“с”, “сонар”) = min(6, 4, 5) = 4
      11. L(“са”, “сона”) = min(4, 3, 3) = 3
      12. L(“сан”, “сона”) = min(4, 2, 3) = 2
      13. L(“са”, “сонар”) = min(5, 4, 4) = 4
      14. L(“сан”, “сонар”) = min(5, 3, 4) = 3

Проверка



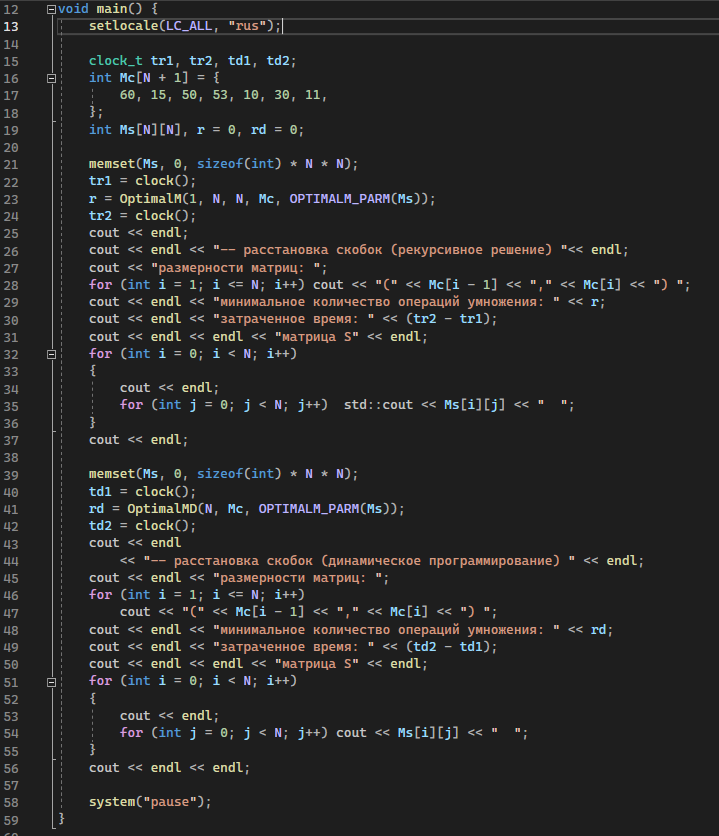
Таким образом, ручное решение рекурсивным способом дало правильный результат.

**Задание 5.**

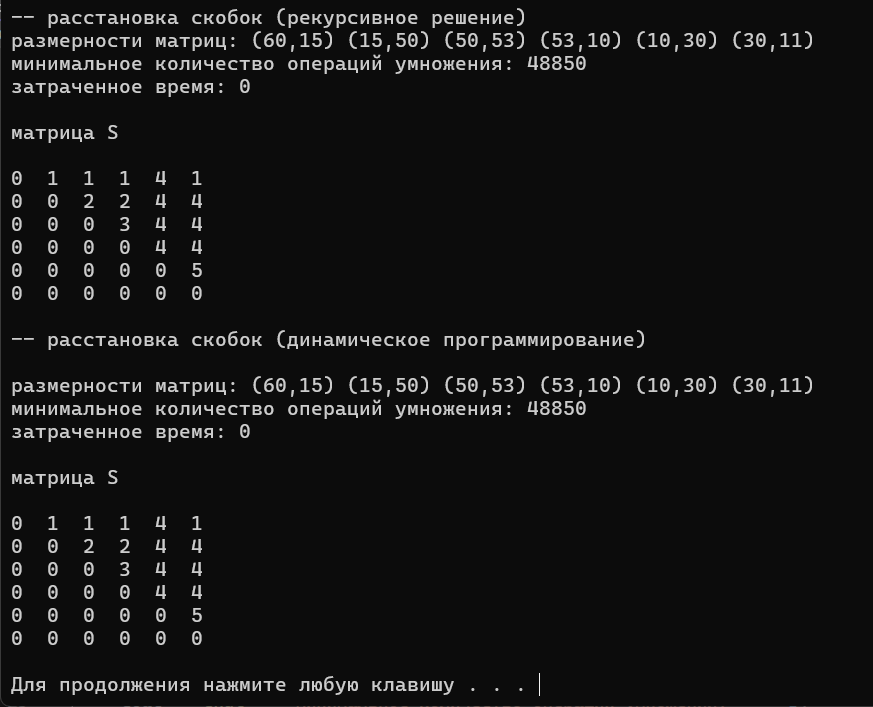
Выполнить сравнительный анализ времени, затраченного на решение задачи об оптимальной расстановке скобок при умножении нескольких матриц для двух методов решения (рекурсивное решение, динамическое программирование). Размерность матриц взять в соответствии с вариантом. Объяснить в отчете принцип расставления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

Условие: 60\*15, 15\*50, 50\*53, 53\*10, 10\*30, 30\*11 (Вариант 14)

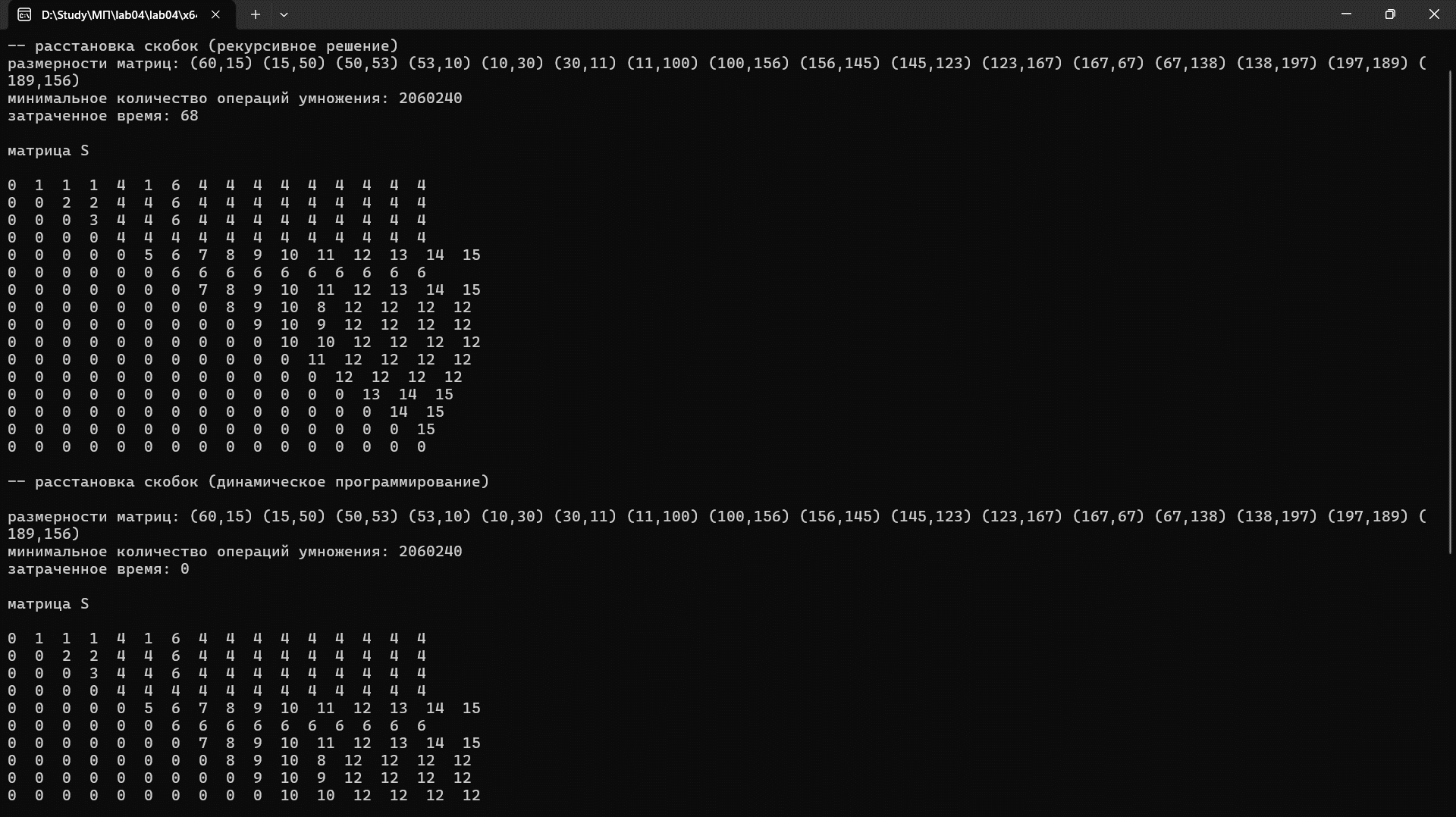
Код программы:



Окно вывода:



Для относительно небольшого количества матриц оба метода выполняются достаточно быстро. При увеличении количества матриц видно, что метод рекурсии выполняется существенно медленнее, в то время как метод динамического программирования не теряет в своей производительности:



Таким образом, для решения данной задачи с большим количеством матриц предпочтительнее использовать метод динамического программирования, хотя использование рекурсивного метода и является допустимым в некоторых случаях.

Объяснение принципа расстановки скобок.



(Индексы в матрице начинаются с 1)

Исходные матрицы: (60,15); (15,50); (50,53); (53,10); (10,30); (30,11).

Скобки расставляются по принципу «сначала внешние – затем внутренние». Элемент матрицы, который имеет индекс (1, 5) – это число 4, оно означает, что точка разрыва скобок находится между 1 и 5 матрицей после 4 матрицы, что позволяет расставить скобки следующим образом:

(A1 x A2 x A3 x A4) x A5 x A6

Точка разрыва между 2 и 6 матрицей находится в элементе матрицы под индексом (2, 6) – это число 4, что означает, что точка разрыва скобок находится между 2 и 6 матрицей после 4 матрицы, что позволяет расставить скобки следующим образом:

(A1 x (A2 x A3 x A4)) x A5 x A6

Совершая те же действия, получается следующая последовательность перемножений:

(A1 x (A2 x (A3 x A4))) x A5 x A6

**Вывод**: в ходе работы были освоены общие принципы решения задач методом динамического программирования, выполнено сравнение полученных решений задач с рекурсивным методом. Рекурсивный метод при малом количестве данных высчитывает результат довольно быстро, однако при большом количестве резко возрастает время ожидания. Однако рекурсивный метод более прост в понимании и реализации. Метод динамического программирования заключается в разбиении задач на подзадачи и работает всегда очень быстро и гораздо быстрее рекурсивного метода. Однако этот метод более сложен в понимании и реализации.