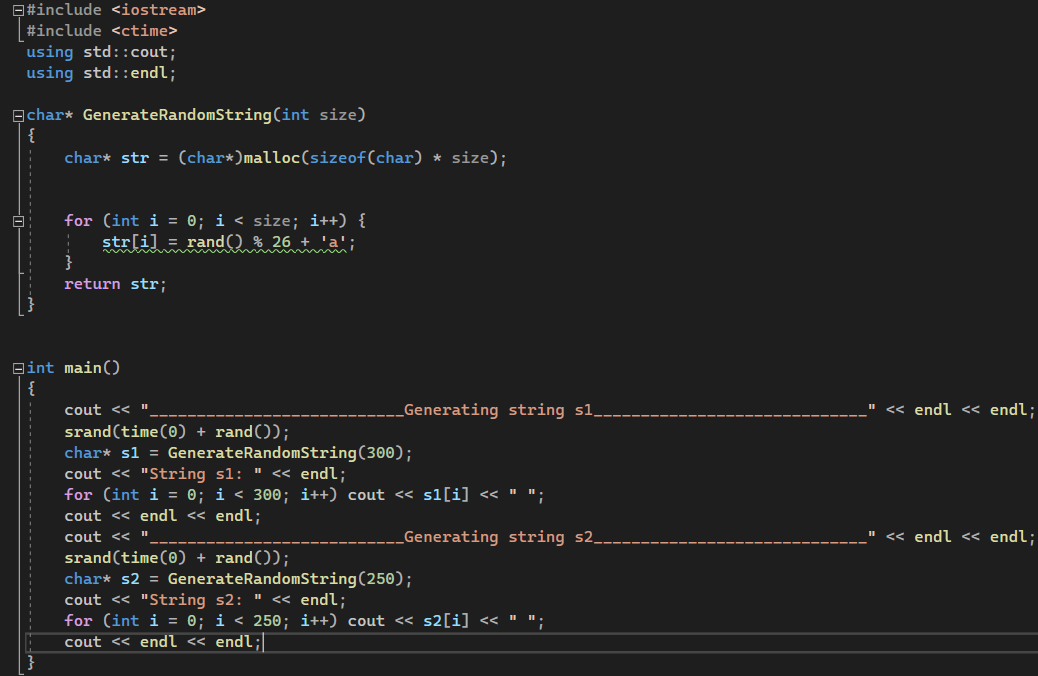
**Лабораторная работа №4. Динамическое программирование**

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

**Ход работы**

## **1 Сгенерировать две строки размером 300 и 250 символов латинского алфавита.**

Код программы:



Листинг 1. Функция генерации случайной строки

Результат:

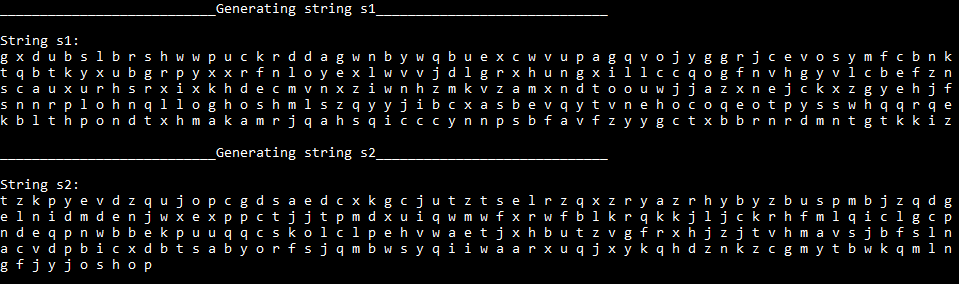
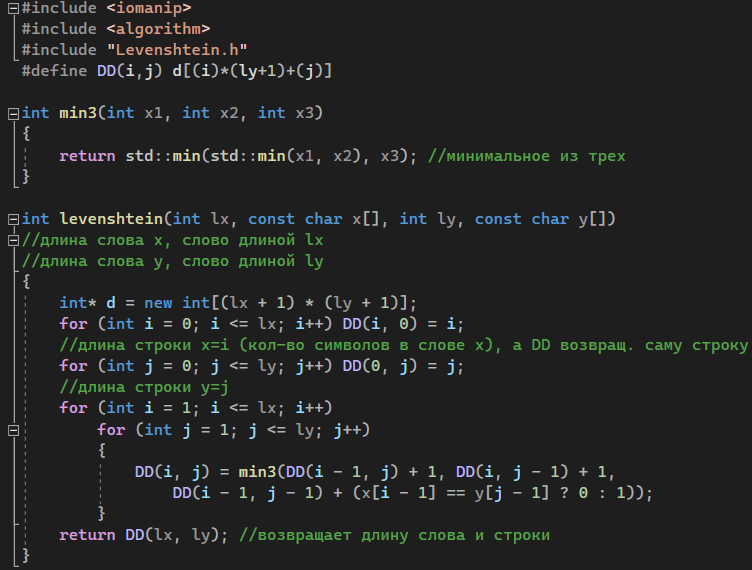


Рисунок 1. Результат генерации строк

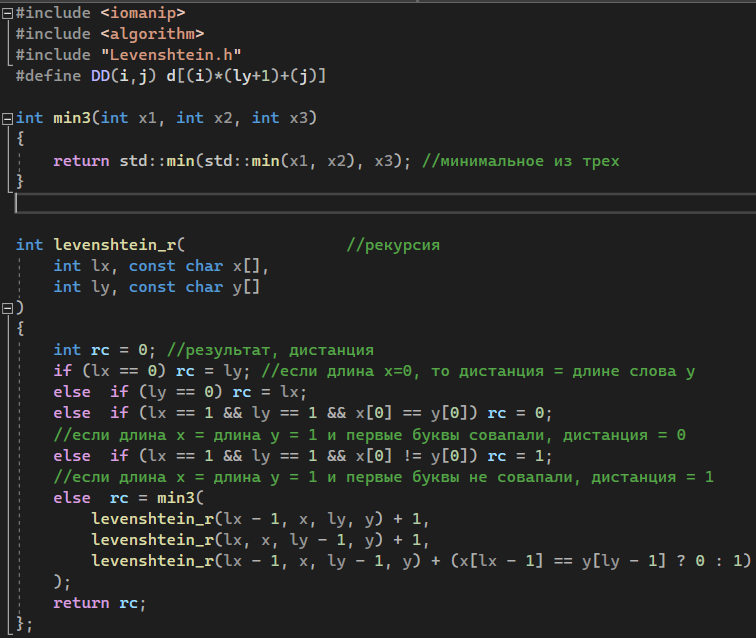
**2 Вычислить двумя способами (рекурсивно и с помощью динамического программирования)  – дистанцию Левенштейна для , где - длина строки ,  - строка состоящая из первых  символов строки . (копии экрана и код вставить в отчет).**

Реализация через динамическое программирование:



Листинг 2.1. Реализация через динамическое программирование

Реализация через рекурсию:



Листинг 2.2. Реализация через рекурсию

Результат:

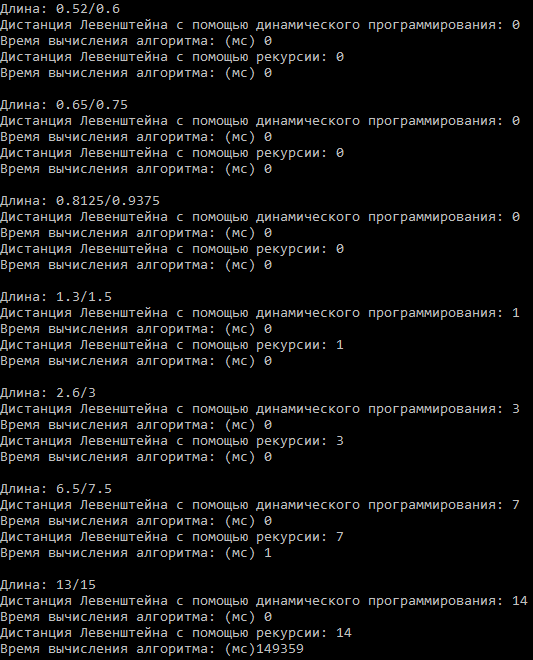


Рисунок 2. Результаты динамического программирования и рекурсии

**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).**

Метод динамического программирования значительно эффективнее, чем рекурсивный метод так, как выполняется значительно быстрее. График построен на результате из рисунка 2.

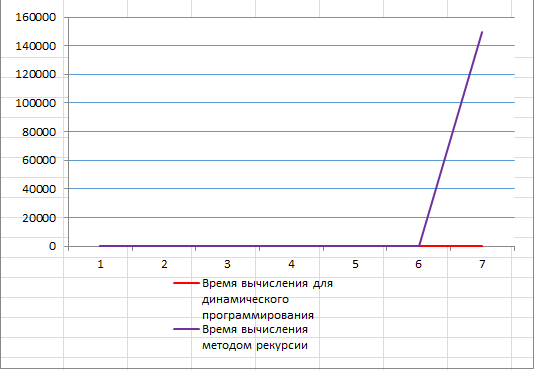


Рисунок 3. График времени выполнения алгоритмов

**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).**

Найти расстояние Левенштейна между словами «Акр» и «Якорь».

1. L(«Акр», «Якорь») = min

2. L(«Ак», «Якорь») = min

3. L(«Акр», «Якор») = min

4. L(«Ак», «Якор») = min

5. L(«А», «Якорь») = min

L(«», «Якорь») = 5,

L(«», «Якор») = 4

6. L(«А», «Якор») = min

L(«», «Якор») = 4,

L(«», «Яко») = 3

7. L(«Акр», «Яко») = min

8. L(«Ак», «Яко») = min

9. L(«Акр», «Як») = min

10. L(«Акр», «Я») = min

L(«Акр», «») = 3,

L(«Ак», «») = 2,

11. L(«Ак», «Як») = min

12. L(«А», «Як») = min

L(«», «Як») = 2,

L(«», «Я») = 1,

13. L(«Ак», «Я») = min

L(«Ак», «») = 2,

L(«А», «») = 1

14. L(«А», «Я») = min

L(«», «Я») = 1,

L(«А», «») = 1,

15. L(«Ак», «Я») = min (2, 3, 2) = 2

16. L(«А», «Як») = min (3, 2, 2) = 2

17. L(«Ак», «Як») = min (3, 3, 1) = 1

18. L(«А», «Яко») = min (4, 3, 3) = 3

19. L(«Акр», «Я») = min (3, 4, 3) = 3

20. L(«Акр», «Як») = min (2, 5, 4) = 2

21. L(«Ак», «Яко») = min (4, 2, 3) = 2

22. L(«Акр», «Яко») = min (3, 3, 2) = 2

23. L(«А», «Якор») = min (5, 4, 4) = 4

24. L(«А», «Якорь») = min (6, 5, 5) = 5

25. L(«Ак», «Якор») = min (5, 3, 4) = 4

26. L(«Акр», «Якор») = min (4, 3, 2) = 2

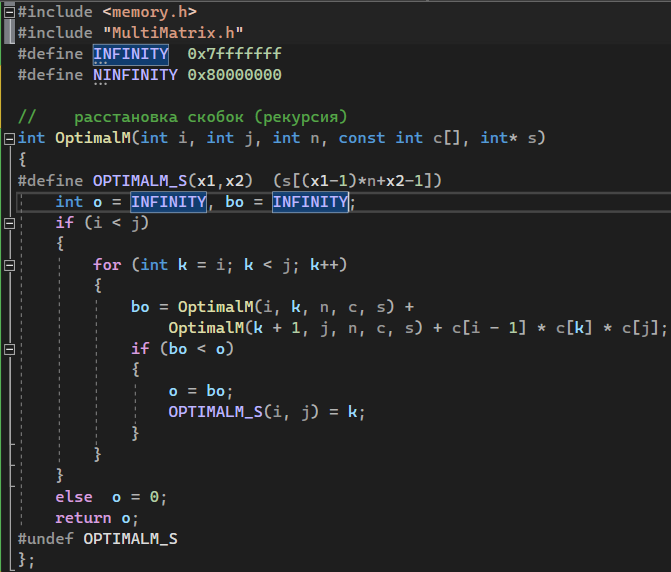
27. L(«Ак», «Якорь») = min (6, 4, 5) = 4

28. L(«Акр», «Якорь») = min (5, 3, 4) = 3

Дистанция Левенштейна для слов «Акр» и «Якорь»: 3.

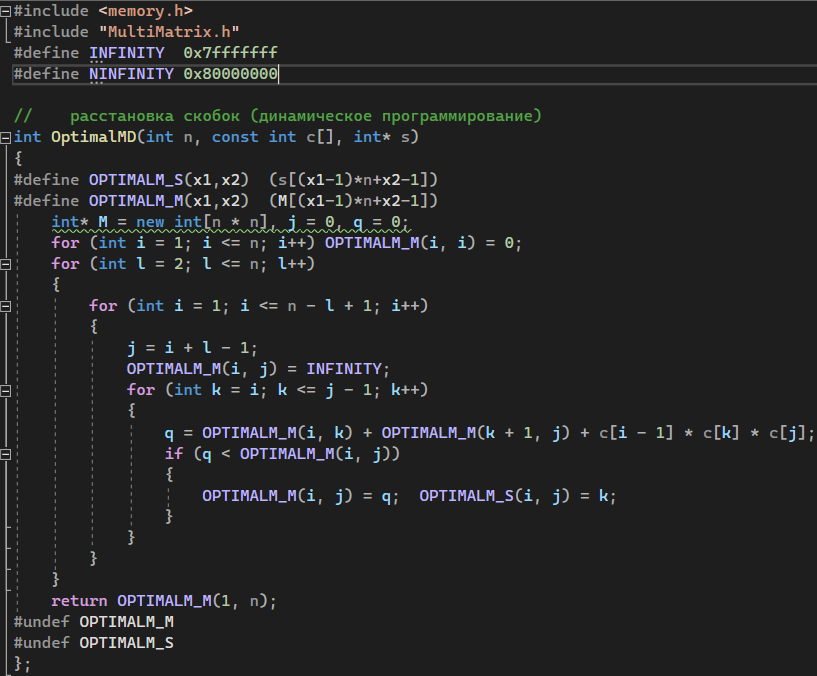
**5 Выполнить сравнительный анализ времени затраченного на решение задачи об оптимальной расстановке скобок при умножении нескольких матриц для двух методов решения (рекурсивное решение, динамическое программирование). Размерность матриц взять в соответствии с вариантом. Объяснить в отчете принцип расставления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

Реализация через рекурсию:



Листинг 5.1. Реализация через рекурсию

Реализация через динамическое программирование:



Листинг 5.2. Реализация через динамическое программирование

Результат:

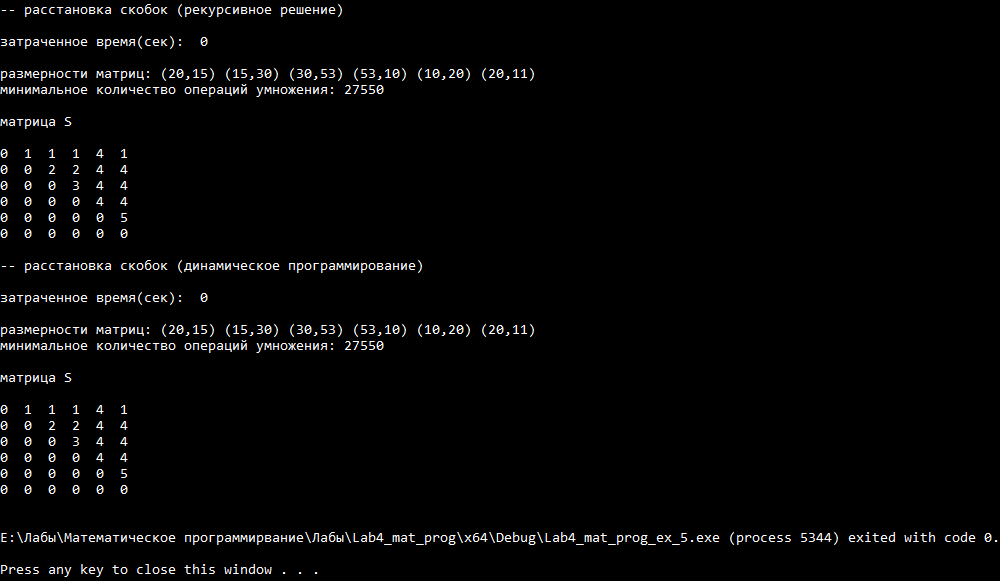


Рисунок 3. Результаты динамического программирования и рекурсии

Принцип расстановки скобок по итоговой матрице:

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

А1=20\*15,

А2=15\*30,

А3=30\*53,

А4 =53\*10,

А5 =10\*20,

А6 =20\*11.

Матрица S:

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

Далее берем элемент (2,3) и получаем, что он равен 2. Следовательно получаем:

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

Далее берем элемент (3,4) и получаем, что он равен 3. Следовательно получаем:

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

И на последнем шаге мы возьмем элемент (4,5) и он равен 4:

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

Это выражение и есть конечное.

Полученная расстановка скобок позволяет получить минимальное количество операций умножения, равное 27550.

**Вывод:** освоил общие принципы решения задач методом динамического программирования, сравнил полученные решения задач с рекурсивным методом.