Учреждение образования

«Белорусский государственный технологический университет»

**Лабораторная работа №1**

Вспомогательные функции

Выполнил:

Студент 2 курса 4 группы ФИТ

Станчик Максим Андреевич

2024 г.

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

**Задание 1**

Используя спецификации, приведенные в условии данной лабораторной работы, мной было разработаны три функции (start, dget и iget). Результат выполнения задания приведен на рисунке 1:

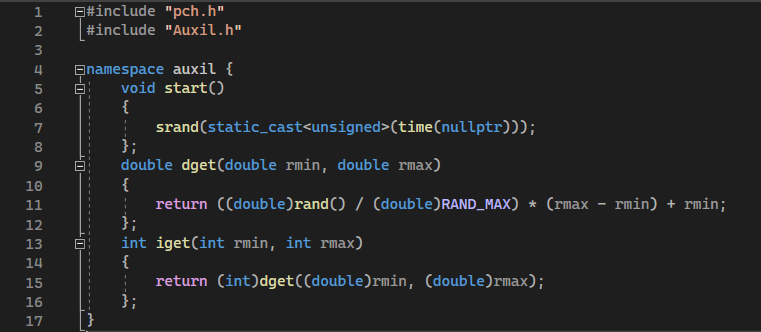


Рисунок 1 – Реализация функций

**Задание 2**

В целях проверки работоспособности разработанных функций и приобретения навыков замера продолжительности процесса вычисления мной также была реализована программа, основанная на примере, приведенном в лабораторной работе. Результат выполнения задания приведен на рисунке 2:

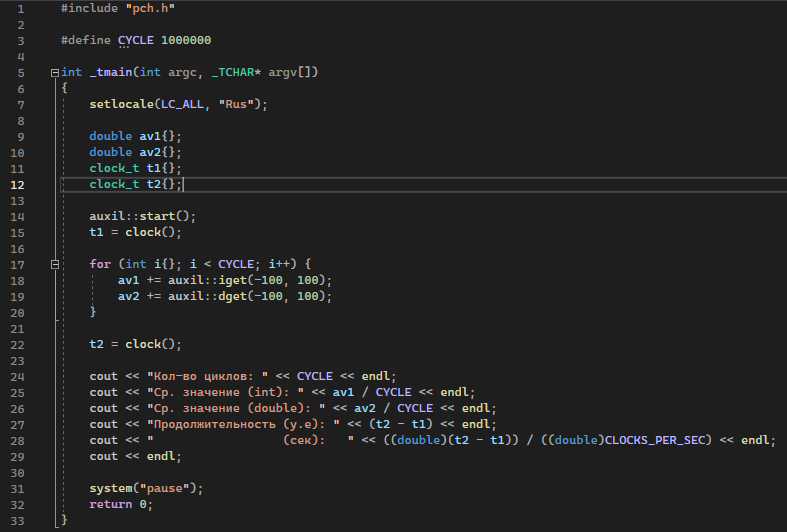


Рисунок 2 – вызов функции и измерение времени выполнения

**Задание 3**

Для выполнения задания 3 мной были проведены необходимые эксперименты и построен график зависимости продолжительности процесса вычисления от количества циклов в программе, реализованной на рисунках 1 и 2. Мной была использована программа для работы с электронными таблицами Microsoft Excel. Проанализировал характер зависимости. Результат выполнения приведен на рисунке 3:

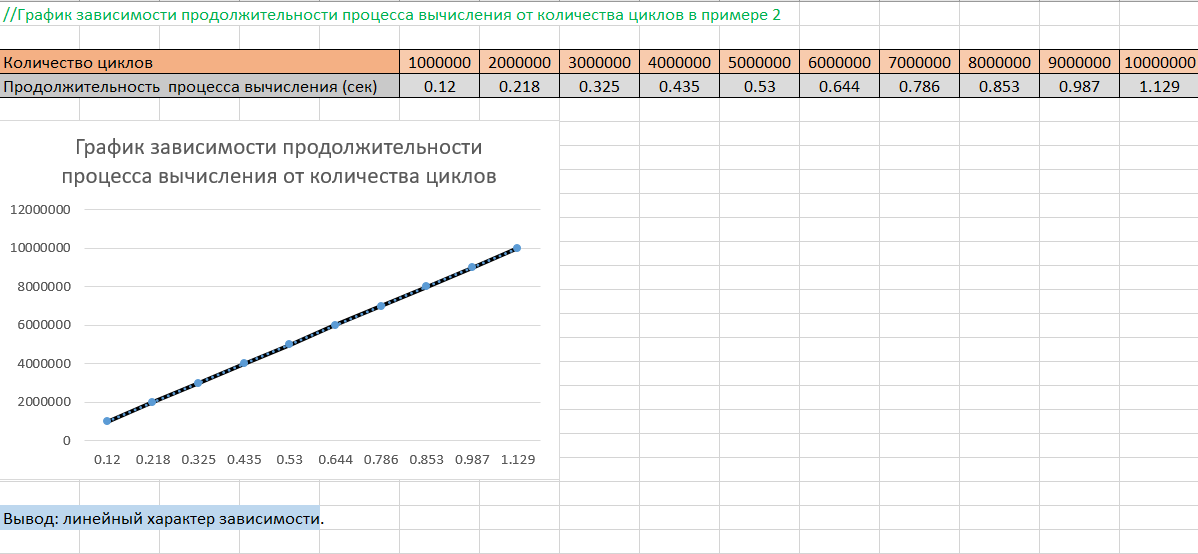


Рисунок 3 – График зависимости продолжительности процесса вычисления от количества циклов в примере 2

Также мной был реализован алгоритм для нахождения факториала. Содержимое файла, где находятся функции для работы данного алгоритма представлен на рисунке 4:

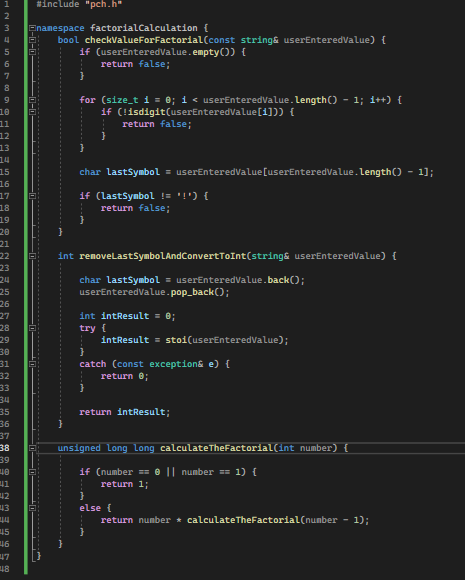


Рисунок 4 – Реализация алгоритма нахождения факториала

Вызов функций и измерение времени выполнения алгоритма находится в файле Main.cpp, после выполнения заданий 1 и 2. Содержимое файла, где находится вызов функций представлен на рисунке 5:

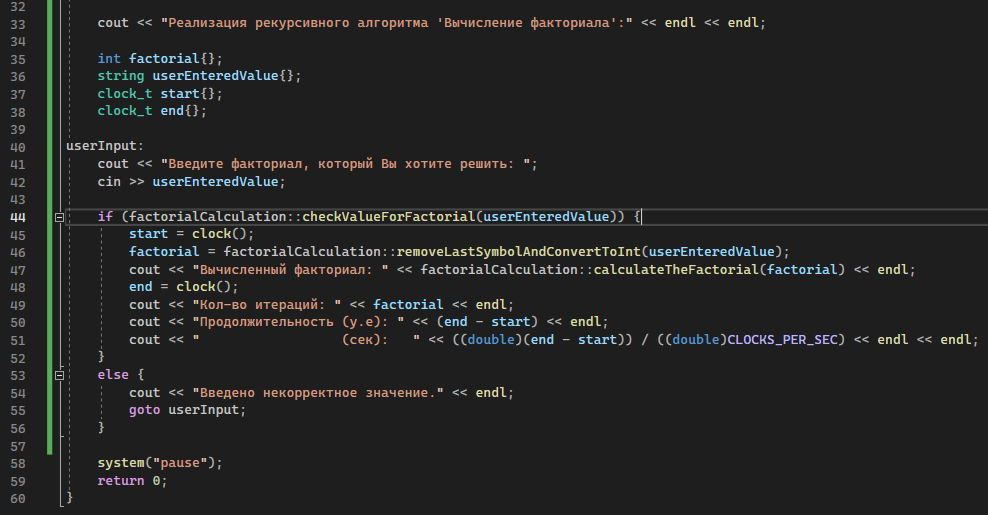


Рисунок 5 – Вызовы функций и измерение времени выполнения алгоритма

Также необходимо было построить график зависимости продолжительности процесса вычисления (сек) от количества итераций. Результат представлен на рисунке 6:

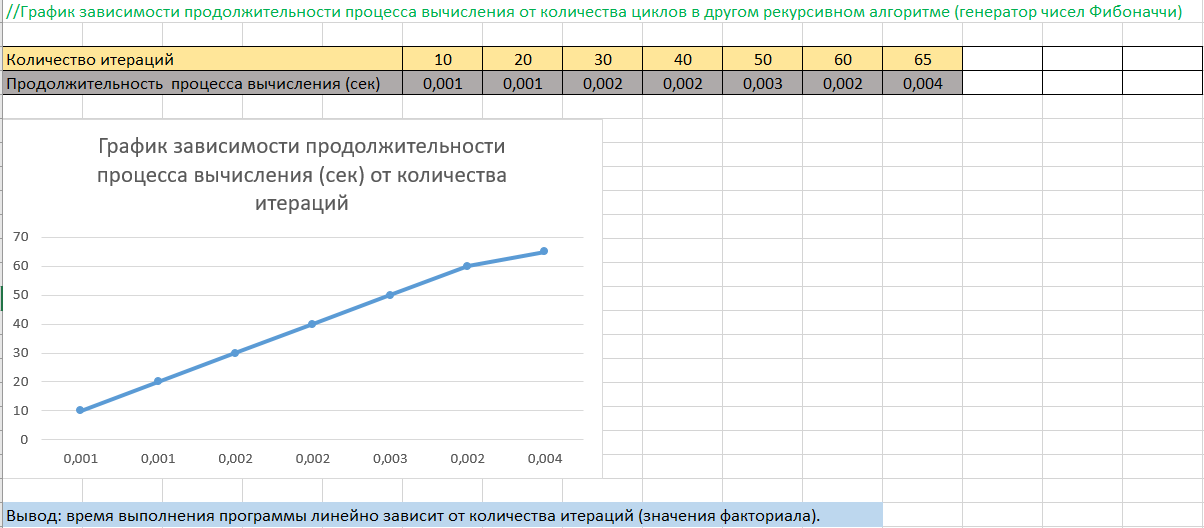


Рисунок 6 – График зависимости продолжительности процесса вычисления (сек) от количества итераций

Работоспособность программы представлена на рисунке 7:

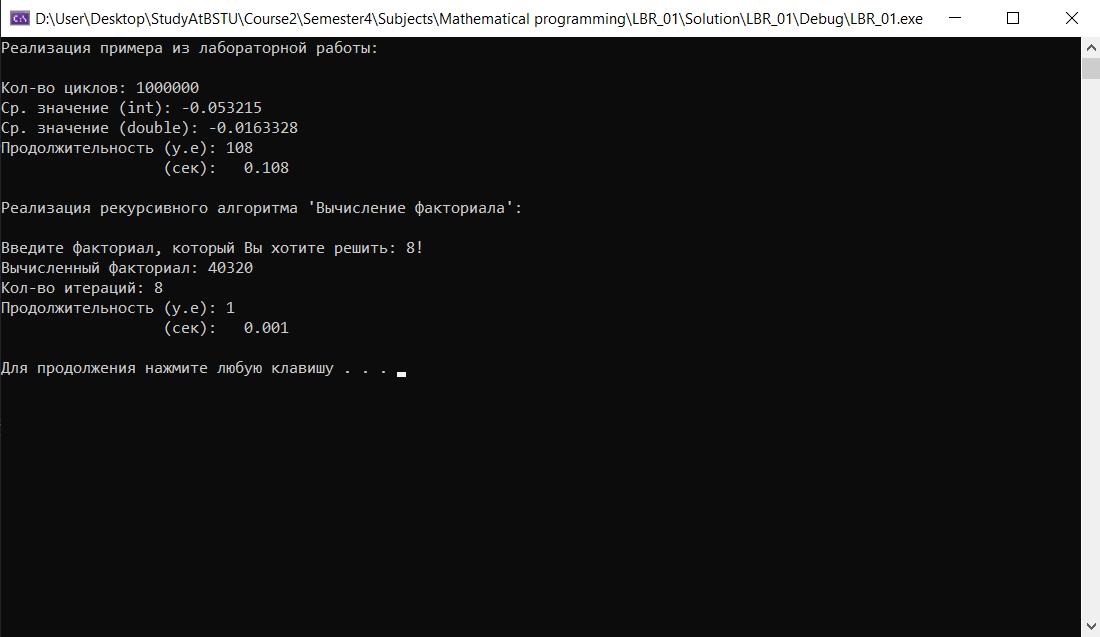


Рисунок 7 – Запуск программы

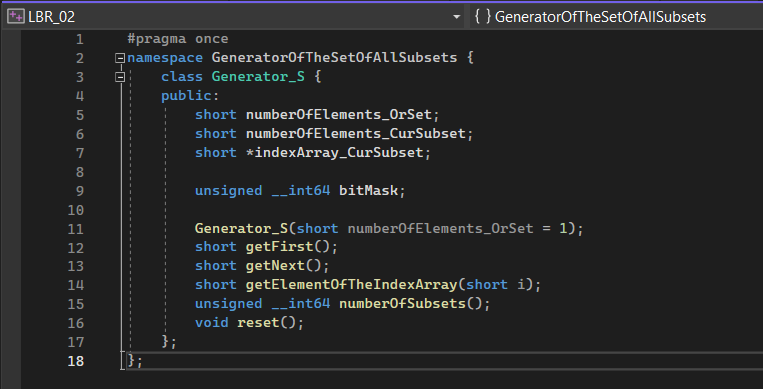
Вывод: в ходе данной лабораторной работы были использованы функции времени, которые нужны для замера чего-либо, например работы алгоритма или программы в целом, разработан алгоритм нахождения факториала, разработаны графики зависимости, первый--график зависимости продолжительности процесса вычисления от количества циклов, второй--график зависимости продолжительности процесса вычисления (сек) от количества итераций.

**Лабораторная работа №2**

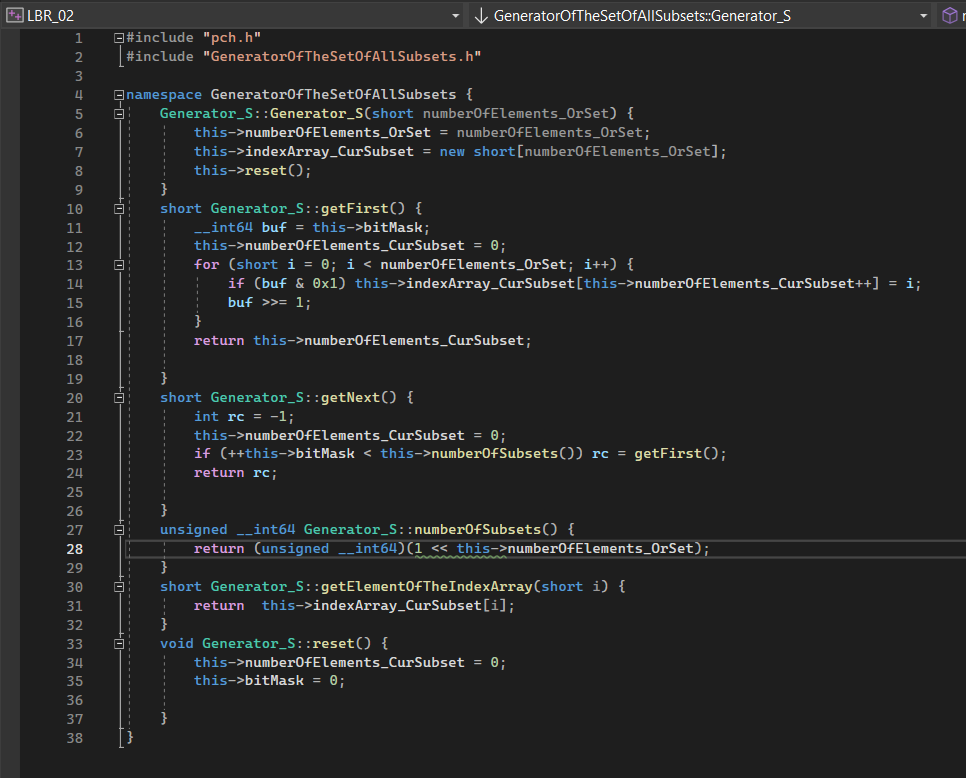
Комбинаторные алгоритмы решения оптимизационных задач

**Задание 1.** Разработать генератор подмножеств заданного множества.

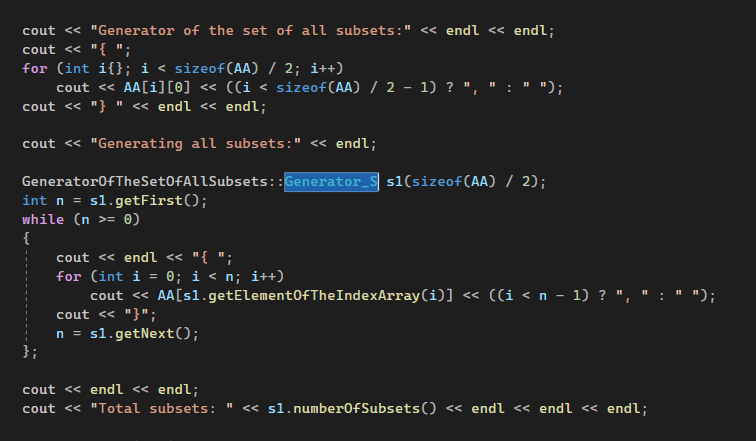
В этом задании было необходимо разработать генератор подмножеств заданного множества. Ниже на рисунках представлены заголовочные файлы и файлы cpp с реализацией поставленной задачи.



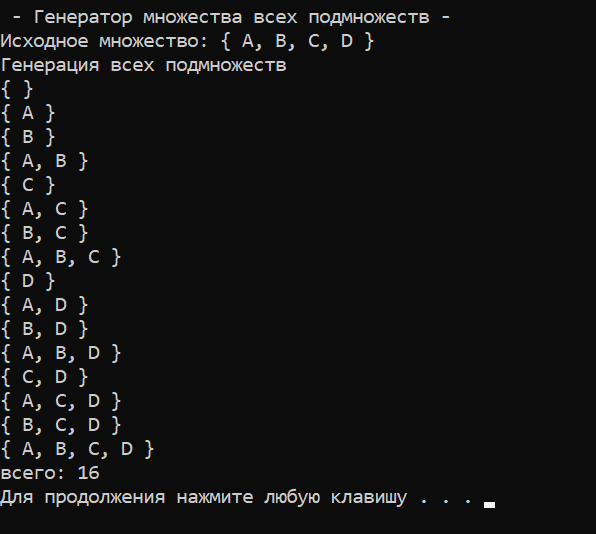
Файл GeneratorOfTheSetOfAllSubsets.h



Файл GeneratorOfTheSetOfAllSubsets.cpp



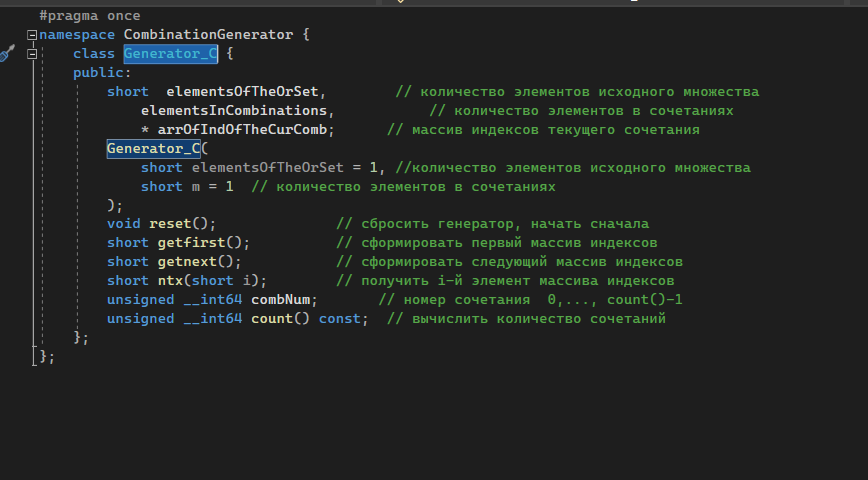
Использование в Main()



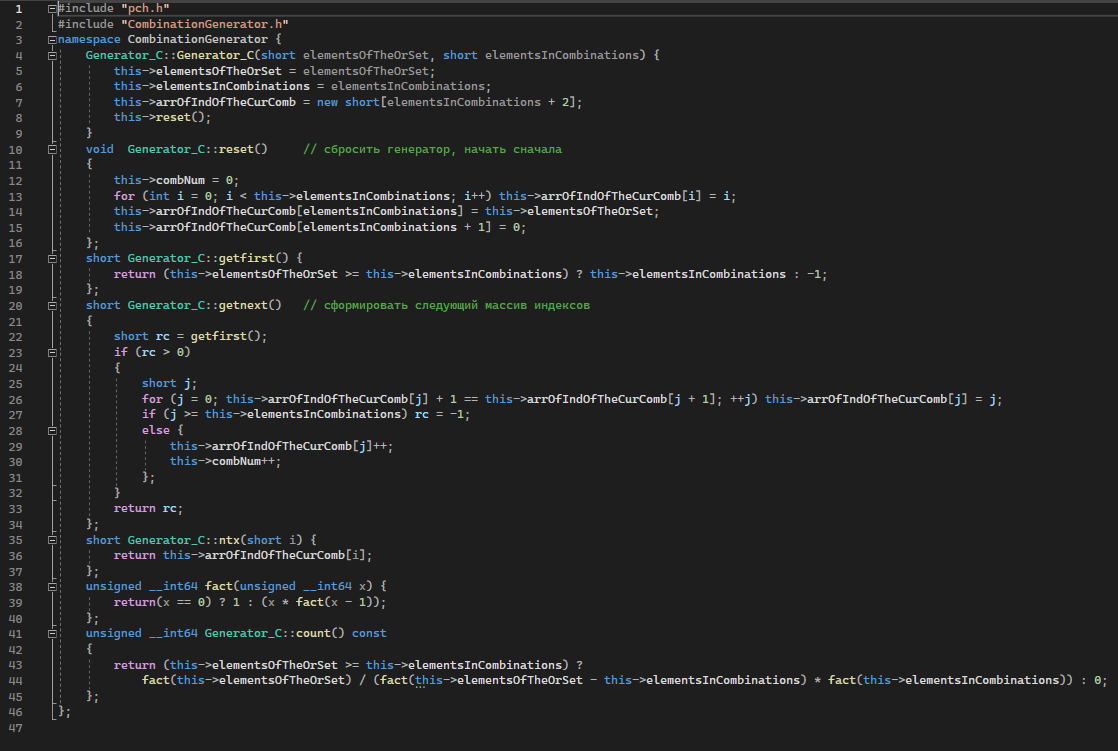
Результат выполнения

**Задание 2.** Разработать генератор сочетаний.

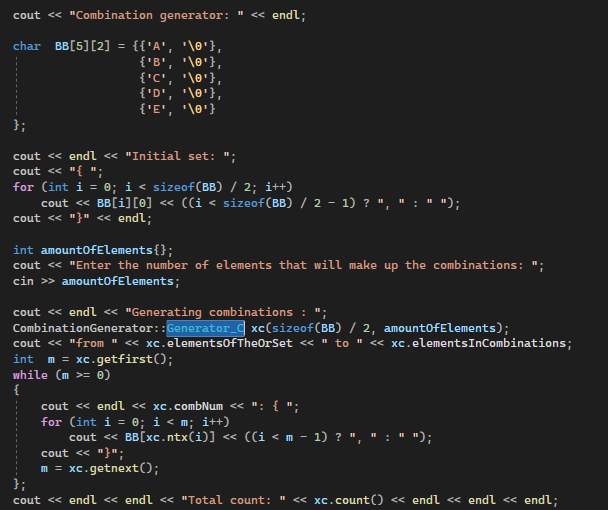
В этом задании было необходимо разработать генератор сочетаний. Ниже на рисунках представлены заголовочные файлы и файлы cpp с реализацией поставленной задачи.



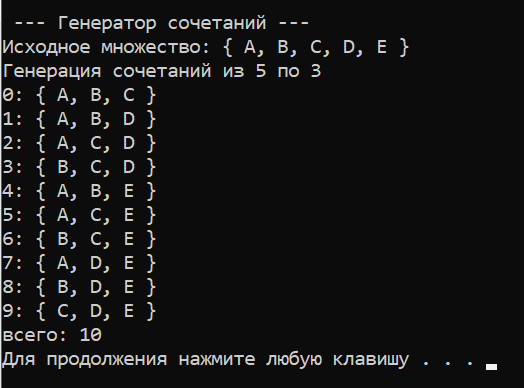
Файл CombinationGenerator.h



Файл CombinationGenerator.cpp



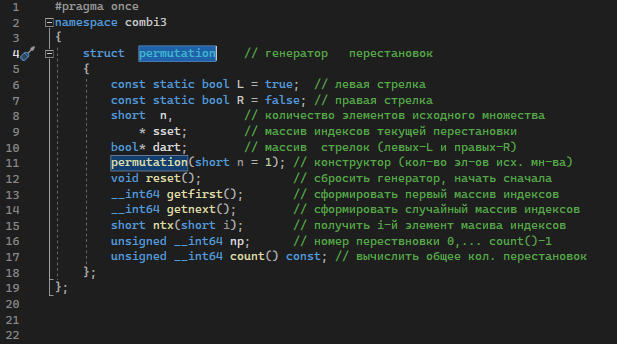
Использование в Main()



Результат выполнения

**Задание 3.** Разработать генератор перестановок.

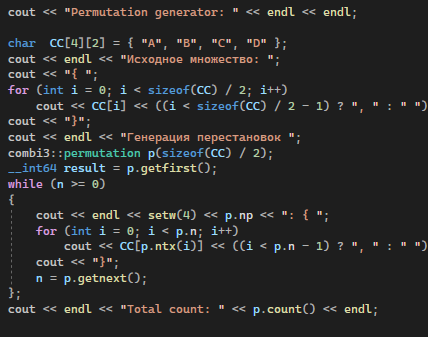
В этом задании было необходимо разработать генератор перестановок. Ниже на рисунках представлены заголовочные файлы и файлы cpp с реализацией поставленной задачи.



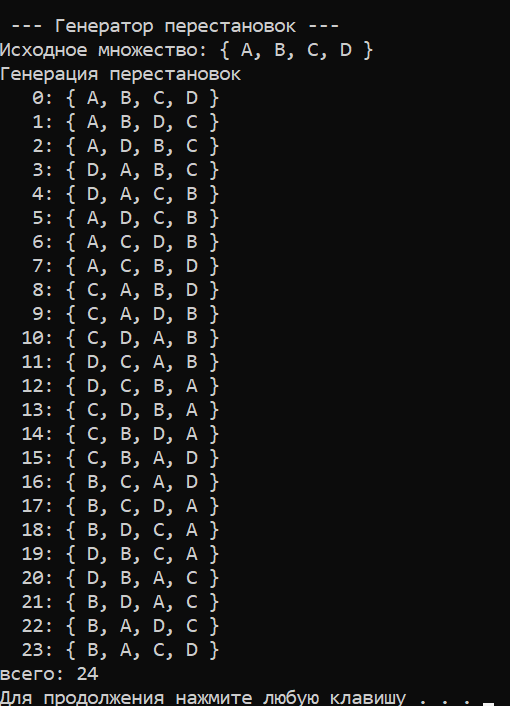
Файл PermutationGenerator.h



Файл PermutationGenerator.cpp



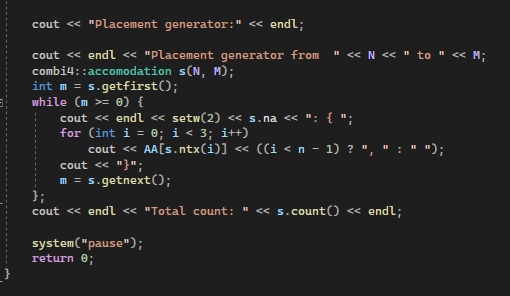
Использование в Main()



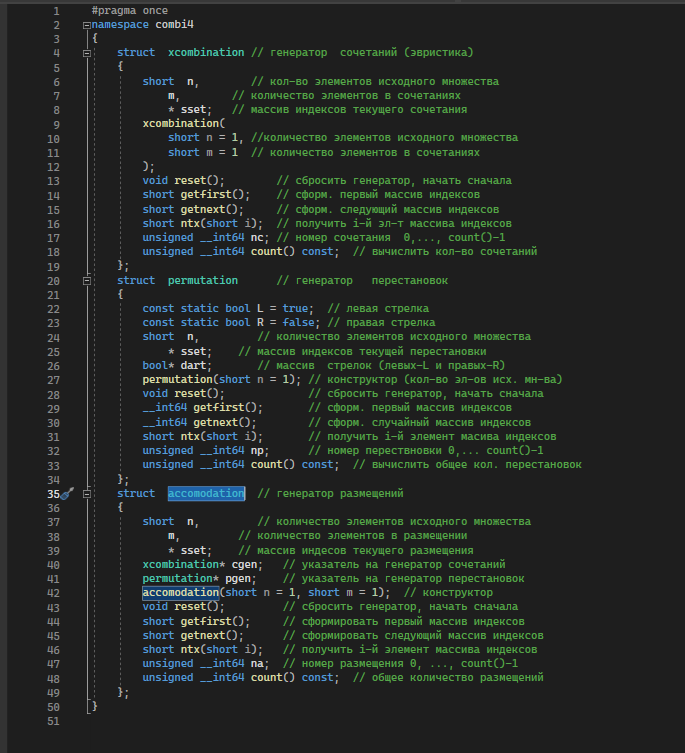
Результат выполнения

**Задание 4.** Разработать генератор размещений.

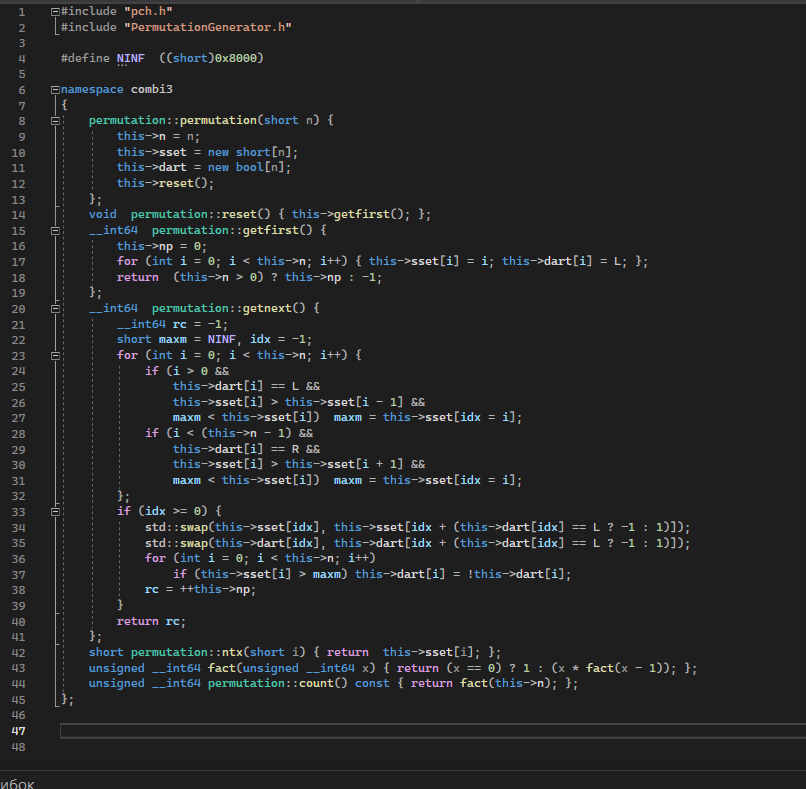
В этом задании было необходимо разработать генератор размещений. Ниже на рисунках представлены заголовочные файлы и файлы cpp с реализацией поставленной задачи.



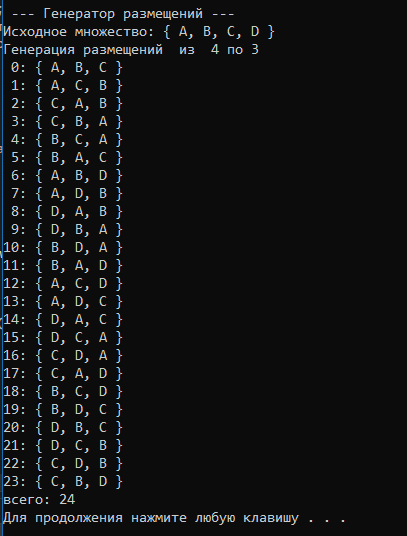
Использование генератора размещений в Main()



Файл PermutationGenerator.h



Файл PermutationGenerator.cpp



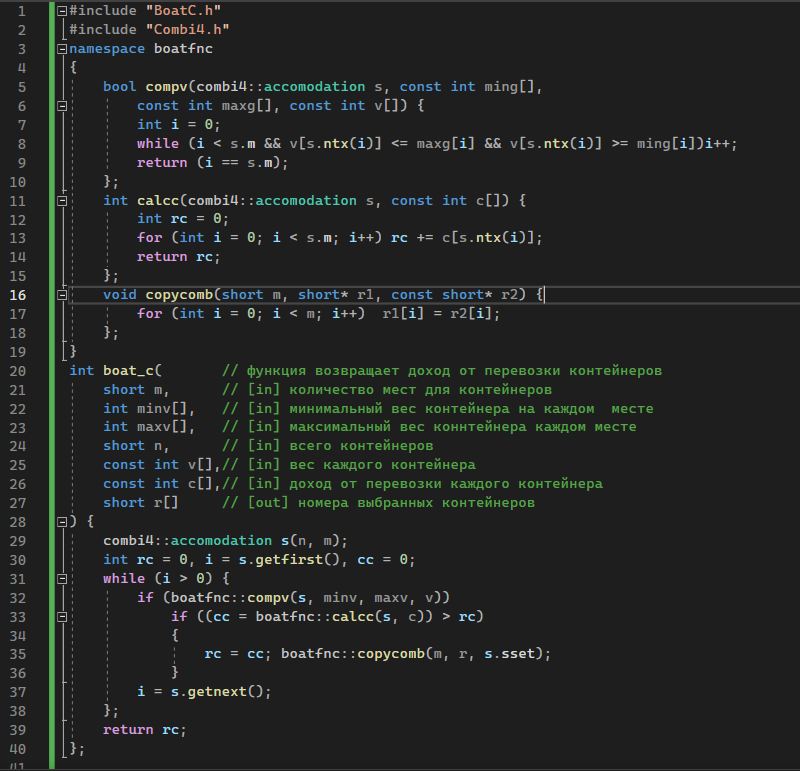
Результат выполнения

**Задание 5.** Решить в соответствии с вариантом задачу и результат занести в отчет. У меня восьмой вариант, поэтому условие задачи следующее:

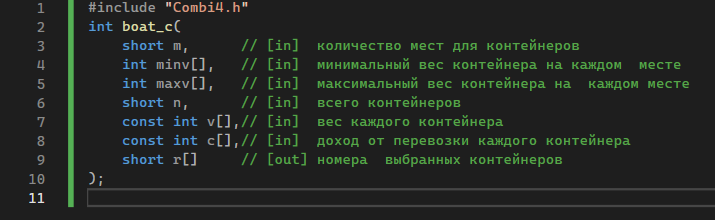
об оптимальной загрузке судна с условием центровки (веса контейнеров сгенерировать случайным образом: количество мест на судне для контейнеров – 5, количество контейнеров 8, веса контейнеров 100 – 200 кг., доход от перевозки 10 – 100 у.е.; минимальный вес контейнера для каждого места 50 – 120 кг, максимальный вес контейнера для каждого места 150 – 850 кг);

Для решения данного задания использовался генератор размещений, схема которого представлена на рисунке:





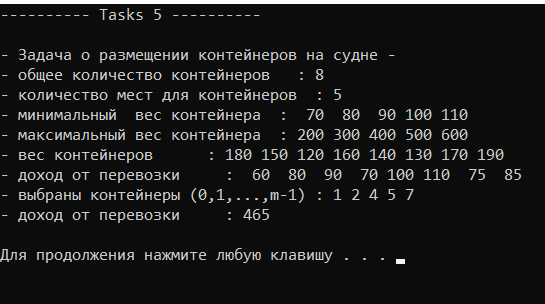
Файл BoatC.cpp



Файл BoatC.h



Использование в Main()



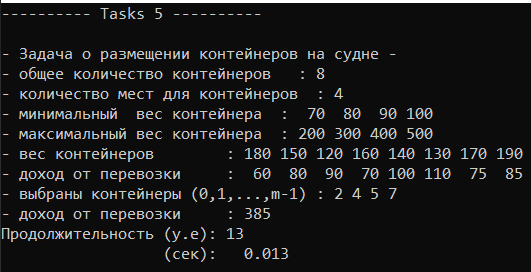
Результат решения задачи

**Задание 6.** Исследовать зависимость времени вычисления необходимое для решения задачи от размерности задачи и результат в виде графика:

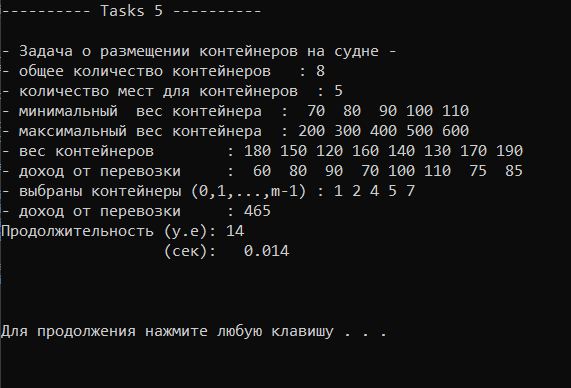
Об оптимальной загрузке судна с условием центровки (количество мест на судне для контейнеров 4 – 8);

Пусть x = количество мест на судне для контейнеров. Тогда:

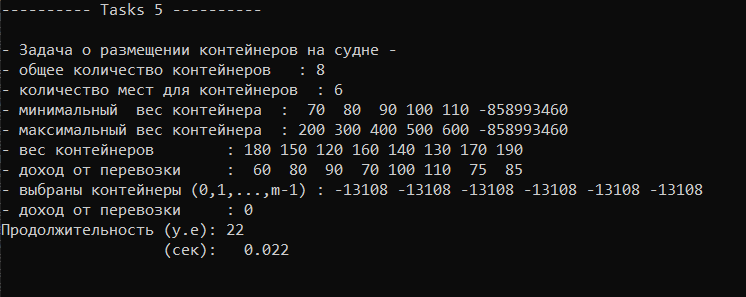
При х = 4:



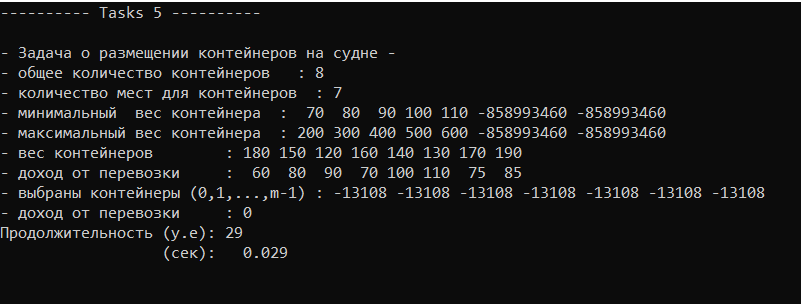
При х = 5:



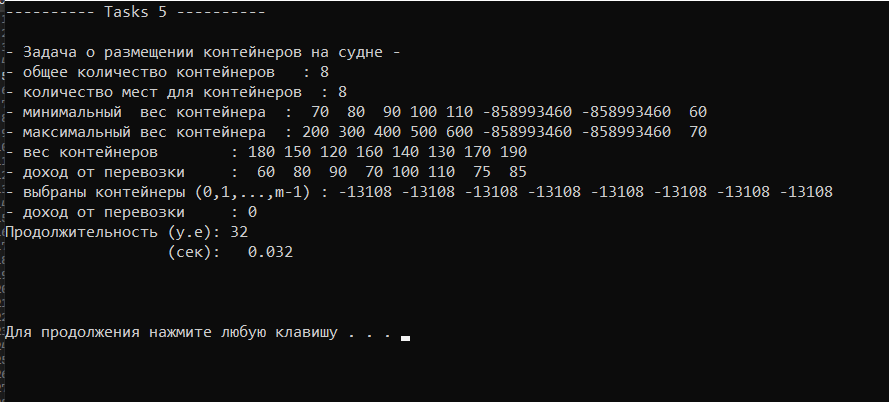
При х = 6:

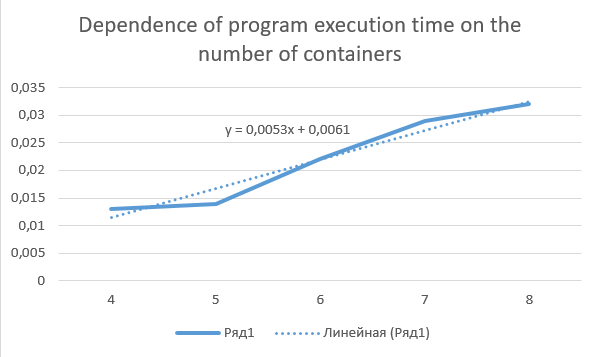


При х = 7:



При х = 8:





Вывод: исходя из полученных данных и графика, можно заметить, что скорость выполнения программы линейно возрастает при добавлении количества мест для контейнеров. Уравнение функции: y = kx +b.

//сочетание размещение перестановки и что такое генератор

Учреждение образования

«Белорусский государственный технологический университет»

**Лабораторная работа №3**

Метод ветвей и границ. Задача коммивояжера и методы её решения.

Выполнил:

Студент 2 курса 4 группы ФИТ

Станчик Максим Андреевич

2024 г.

**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.**

|  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- |
| **Город** | **1** | **2** | **3** | **4** | **5** |
| **1** |  | 16 | 29 |  | 8 |
| **2** | 8 |  | 23 | 60 | 76 |
| **3** | 10 | 24 |  | 86 | 57 |
| **4** | 25 | 50 | 32 |  | 24 |
| **5** | 85 | 74 | 52 | 21 |  |

*Вариант 8*

**Задание 2.**

В каждой строке (городе) необходимо найти минимальное число. После чего от каждого элемента рассматриваемой строки необходимо отнять минимальное число этой строки.

|  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- |
| **Город** | **1** | **2** | **3** | **4** | **5** |
| **1** |  | 16 | 29 |  | 8 |
| **2** | 8 |  | 23 | 60 | 76 |
| **3** | 10 | 24 |  | 86 | 57 |
| **4** | 25 | 50 | 32 |  | 24 |
| **5** | 85 | 74 | 52 | 21 |  |

|  |  |
| --- | --- |
| **Строка** | **Минимальное значение (константа приведения)** |
| **1** | 8 |
| **2** | 8 |
| **3** | 10 |
| **4** | 24 |
| **5** | 21 |

Сумма минимальных значений в строках: 8 + 8 + 10 + 24 + 21 = 71 – константа приведения;

От каждого элемента отняли минимальное значение строки, в которой находится рассматриваемый элемент. Это называется приведение таблицы по строке.

Изменение всех элементов столбца матрицы расстояний на одно и то же число не влияет на выбор оптимального маршрута коммивояжера.

Преобразованная таблица:

|  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- |
| **Город** | **1** | **2** | **3** | **4** | **5** |
| **1** |  | 8 | 21 |  | 0 |
| **2** | inf |  | 15 | 52 | 68 |
| **3** | 0 | 14 |  | 76 | 47 |
| **4** | 1 | 26 | 8 |  | 0 |
| **5** | 64 | 53 | 31 | 0 |  |

В каждом столбце нашли минимальное число. От каждого элемента в столбце отняли это число.

|  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- |
| **Номер столбца** | **1** | **2** | **3** | **4** | **5** |
| **Минимальное значение (константа приведения)** | 0 | 8 | 8 |  | 0 |

Сумма минимальных значений в столбцах: 16 – константа приведения;

От каждого элемента отняли минимальное значение столбца, в котором находится рассматриваемый элемент. Это называется приведение таблицы по столбцу.

Полностью приведенная таблица:

|  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- |
| **Город** | **1** | **2** | **3** | **4** | **5** |
| **1** |  | 0 | 13 |  | 0 |
| **2** | 0 |  | 7 | 52 | 68 |
| **3** | 0 | 6 |  | 76 | 47 |
| **4** | 1 | 18 | 0 |  | 0 |
| **5** | 64 | 45 | 23 | 0 |  |

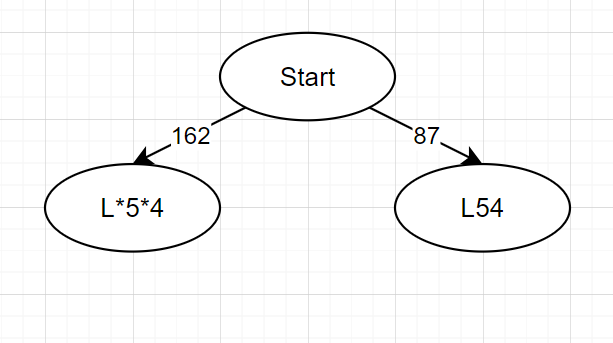
Складывая константы приведения таблицы получаем нижнюю границу

Сумма констант приведения: 71 + 16 = 87 – нижняя граница;

Кол-во расстояний между городами = кол-во нулевых элементов в таблице.

|  |  |  |
| --- | --- | --- |
| **Расстояния между городами** | **Вычисление значения** | **Значение** |
| **L21** | 0 + 7 | 7 |
| **L31** | 0 + 6 | 6 |
| **L12** | 0 + 6 | 6 |
| **L43** | 7 + 0 | 7 |
| **L54** | 52 + 23 | 75 |
| **L15** | 0 + 0 | 0 |
| **L45** | 0 + 0 | 0 |

Самое большое значение имеет L54



Start – начало

Если выберем L54, необходимо будет потратить 87. Если выберем не L54, то есть L\*5\*4, нужно будет потратить 162. Т.к. 87 < 162, выбираем L54. В таком случае удаляем из таблицы строку 5 и столбец 4. Значение на строке 4 и столбце 5 станет INF. Тогда таблица имеет вид:

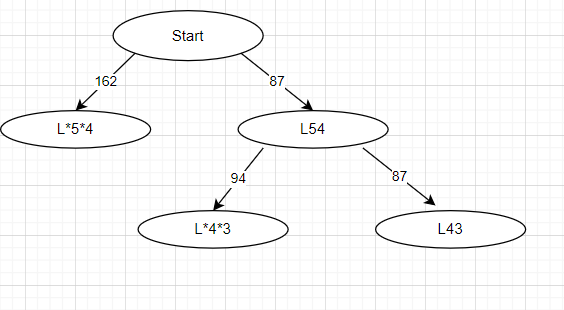
|  |  |  |  |  |
| --- | --- | --- | --- | --- |
| **Город** | **1** | **2** | **3** | **5** |
| **1** |  | 0 | 13 | 0 |
| **2** | 0 |  | 7 | 68 |
| **3** | 0 | 6 |  | 47 |
| **4** | 1 | 18 | 0 | 0 |

|  |  |  |
| --- | --- | --- |
| **Расстояния между городами** | **Вычисление значения** | **Значение** |
| **L21** | 0 + 0 | 0 |
| **L31** | 6 + 0 | 6 |
| **L12** | 6 + 0 | 6 |
| **L43** | 0 + 7 | 7 |
| **L15** | 0 + 0 | 0 |
| **L45** | 0 + 0 | 0 |

Сумма минимальных значений по столбцам: 0;

Сумма минимальных значений по строкам: 0;

Необходимо использовать L43



Получили таблицу:

|  |  |  |  |
| --- | --- | --- | --- |
| **Город** | **1** | **2** | **5** |
| **1** |  | 0 | 0 |
| **2** | 0 |  | 68 |
| **3** | 0 | 6 | 47 |

|  |  |
| --- | --- |
| **Строка** | **Минимальное значение (константа приведения)** |
| **2** | 0 |
| **3** | 0 |
| **4** | 0 |

|  |  |
| --- | --- |
| **Столбец** | **Минимальное значение (константа приведения)** |
| **1** | 0 |
| **2** | 0 |
| **5** | 0 |

Сумма минимальных значений в строках: 0 + 0 + 0 = 0;

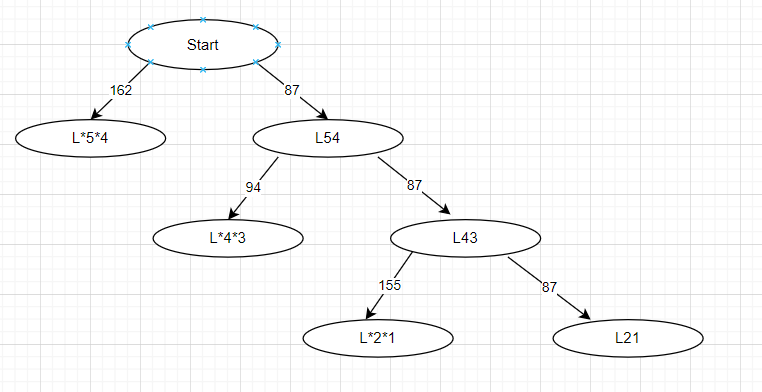
Сумма минимальных значений в столбцах: 0 + 0 + 0 = 0;

Сумма констант приведения: 0 + 0 = 0;

87 + 0 = 87;

|  |  |  |
| --- | --- | --- |
| **Расстояния между городами** | **Вычисление значения** | **Значение** |
| **L21** | 68 + 0 | 68 |
| **L31** | 0 + 6 | 6 |
| **L12** | 0 + 6 | 6 |
| **L15** | 47 + 0 | 47 |

Необходимо использовать L21.



Получили таблицу:

|  |  |  |
| --- | --- | --- |
| **Город** | **2** | **5** |
| **1** | 0 | 0 |
| **3** | 6 | 47 |

|  |  |
| --- | --- |
| **Строка** | **Минимальное значение (константа приведения)** |
| **1** | 0 |
| **3** | 6 |

|  |  |
| --- | --- |
| **Столбец** | **Минимальное значение (константа приведения)** |
| **1** | 0 |
| **2** | 0 |

Сумма минимальных значений в строках: 0 + 6 = 6;

Сумма минимальных значений в столбцах: 0 + 0 = 0

Сумма констант приведения: 0 + 6 = 6;

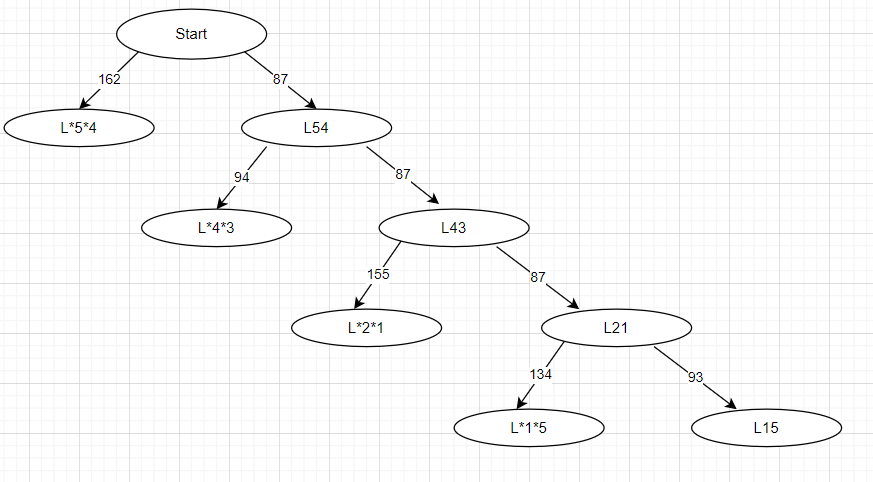
87 + 6 = 93;

Получили таблицу:

|  |  |  |
| --- | --- | --- |
| **Город** | **2** | **5** |
| **1** | 0 | 0 |
| **3** | 0 | 41 |

|  |  |  |
| --- | --- | --- |
| **Расстояния между городами** | **Вычисление значения** | **Значение** |
| **L12** | 0 + 0 | 0 |
| **L15** | 0 + 41 | 41 |
| **L32** | 0 + 41 | 41 |

Выбираем L15.

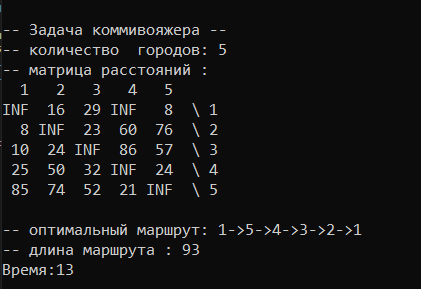


Получили таблицу:

|  |  |
| --- | --- |
| **Город** | **2** |
| **3** | 0 |

Ответ: 93.

После проверки результата при помощи генератора перестановок из лабораторной работы 2, полученные ответы совпадали с исходным решением.



Вывод: алгоритм ветвей и границ подходит для решения общей задачи целочисленного линейного программирования. Метод является вариацией полного перебора с отсевом подмножеств допустимых решений, заведомо не содержащих оптимальных решений. Применяется в решении задачи о коммивояжере, минимаксных задачах о назначениях, задачи календарного планирования, задачи о трех станках (вероятностные задачи).

**Лабораторная работа №4**

Динамическое программирование.

Выполнил:

Студент 2 курса 4 группы ФИТ

Станчик Максим Андреевич

2024 г.

**Задание 1**

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

На рисунке 1 представлена функция для генерации строк:

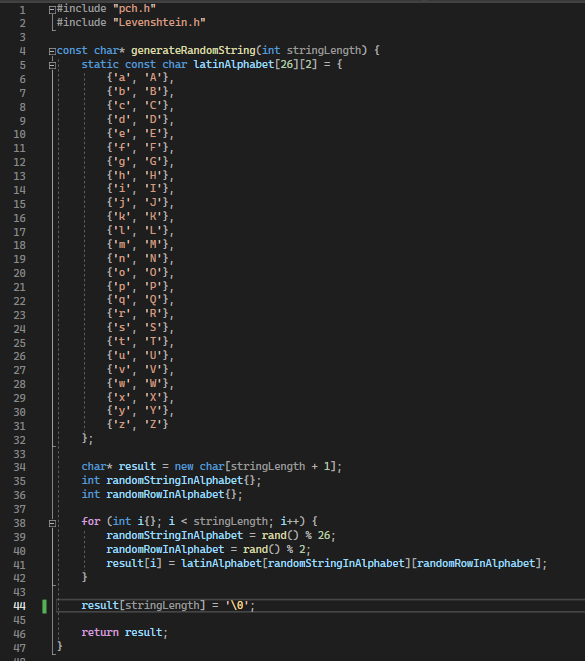


Рисунок 1 – Функция для генерации строк

На рисунке 2 представлен код, где эта функция вызывается для заполнения двух строковых переменных нужными значениями.

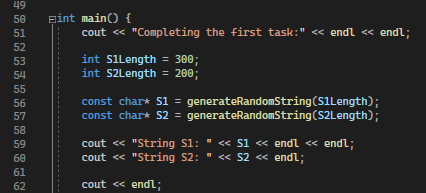


Рисунок 2 – Вызов функций в int main()

**Задание 2**

Необходимо при помощи двух способов вычислить дистанцию Левенштейна для . Код выполнение программы представлен на рисунках ниже.

На рисунке 3 представлено содержание файла Levenshtein.h

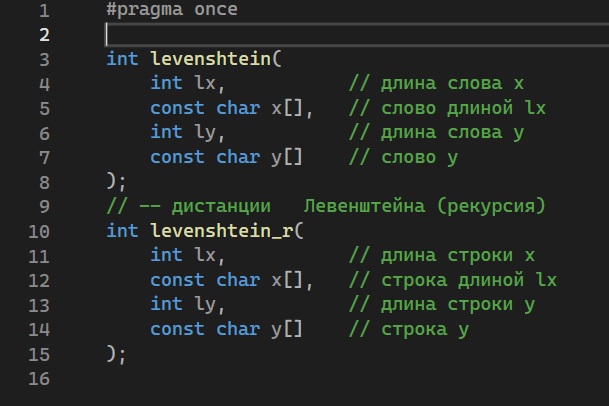


Рисунок 3 – прототипы функций

На рисунке 4 представлено содержание файла Levenshtein.cpp

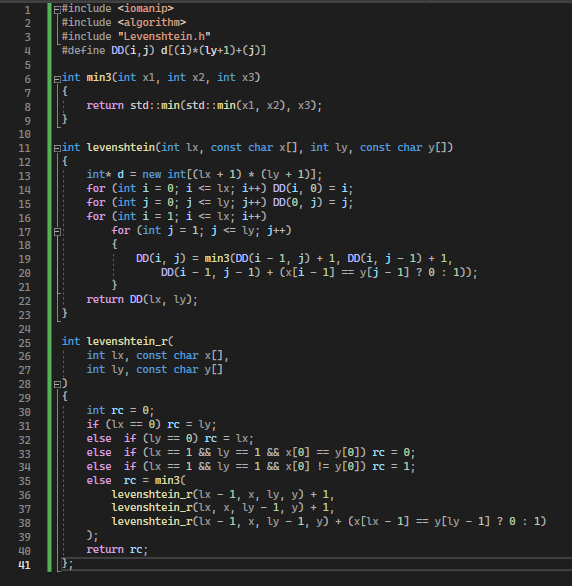


Рисунок 4 – реализация функций для вычисления расстояния Левенштейна

Результат выполнения программы представлен на рисунке 5.

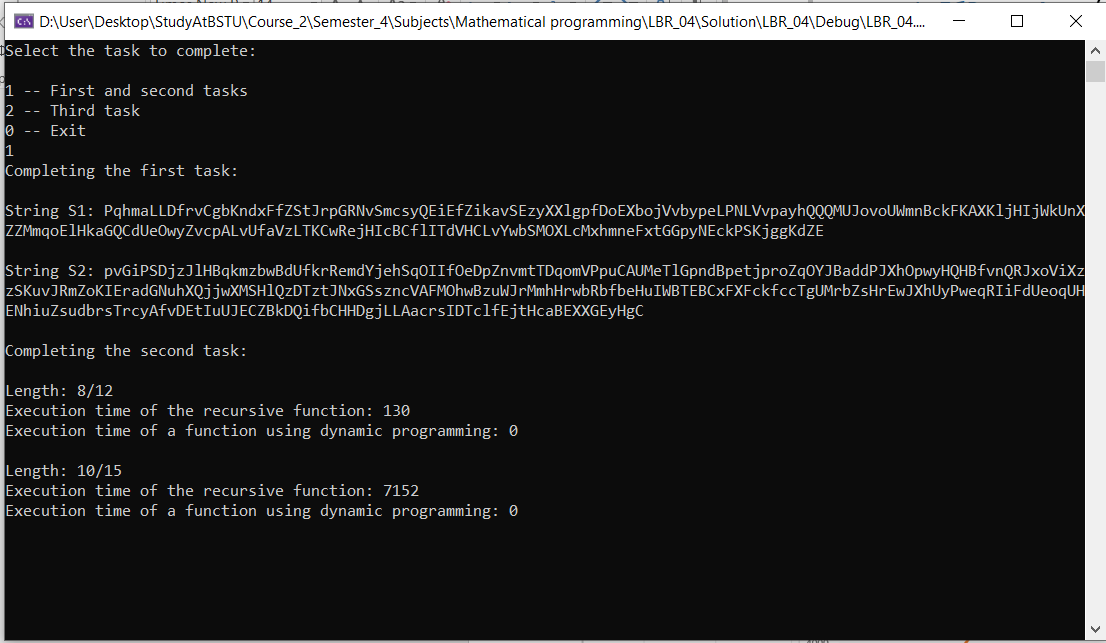


Рисунок 5 – результат выполнения программы

**Задание 3**

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

На рисунке 6 представлен график зависимости выполнения.

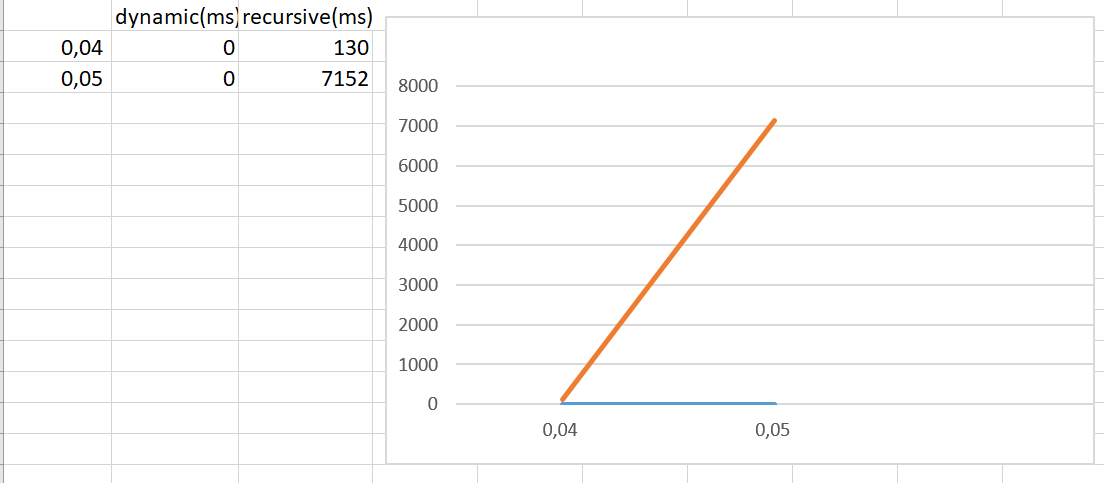


Рисунок 6 – график зависимости выполнения

***Задание 4.***

Реализовать вручную пример вычисления дистанции Левенштейна при помощи рекурсивного алгоритма. Ниже представлен алгоритм решения последовательно:



Рассмотрим 3 варианта:

1. Изменение первого
2. Изменение второго
3. Удаление обоих слов

= 5.

= 4.



= 3.

= 2.

= 2.

= 1.

= 3.

= 2.

= 1.

= 0.

16. L(“B”, “Б”) = min(2,2,1) = 1.

17. L(“Ва”,”Б”) = min(2,3,2) = 2.

18. L(“Вар”,”Б”) = min(3,4,3) = 3.

19. L(“В”,”Ба”) = min(3,2,2) = 2.

20. L(“Ва”,”Ба”) = min(3,3,1) = 1.

21. L(“Вар”,”Ба”) = min(2,4,3) = 2.

22. L(“В”,”Бар”) = min(4,3,3) = 3.

23. L(“Ва”,”Бар”) = min(4,2,3) = 2.

24. L(“Вар”,”Бар”) = min(3,3,1) = 1.

1. L(“В”,”Бара”) = min(5,4,4) = 4.
2. L(“В”,”Баран”) = min(6,5,5) = 5.
3. L(“Ва”,”Бара”) = min(5,3,3) = 3.
4. L(“Вар”,”Бара”) = min(4,2,3) = 2.
5. L(“Ва”,”Баран”) = min(6,4,5) = 4.
6. L(“Вар”,”Баран”) = min(5,3,4) = 3.

Условие: Вар Баран

Решение:

1. Вар Варан
2. Вара Варан
3. Варан Варан

Ответ: 3.

***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.



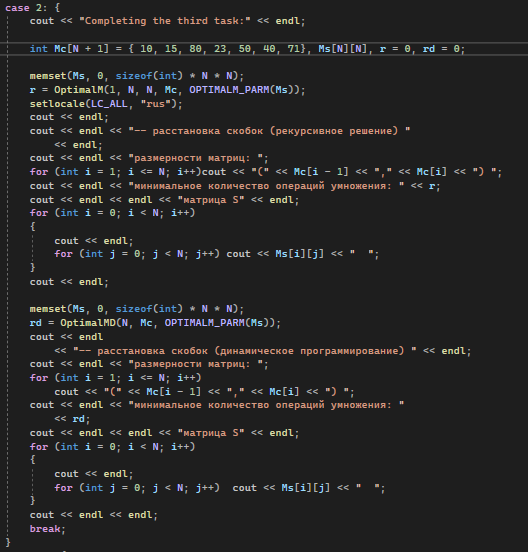


Рисунок 7 – вызов функций в main()

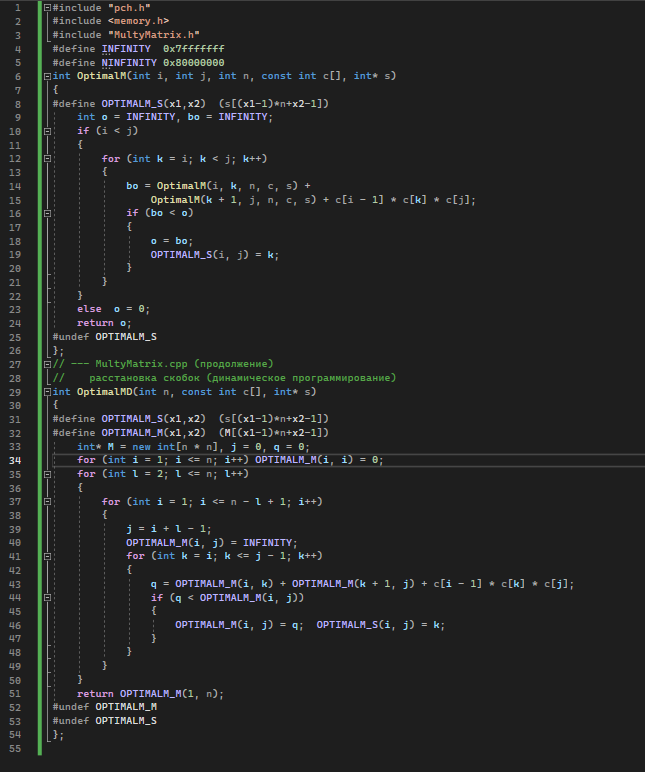


Рисунок 8 – содержание файла MultyMatrix.cpp

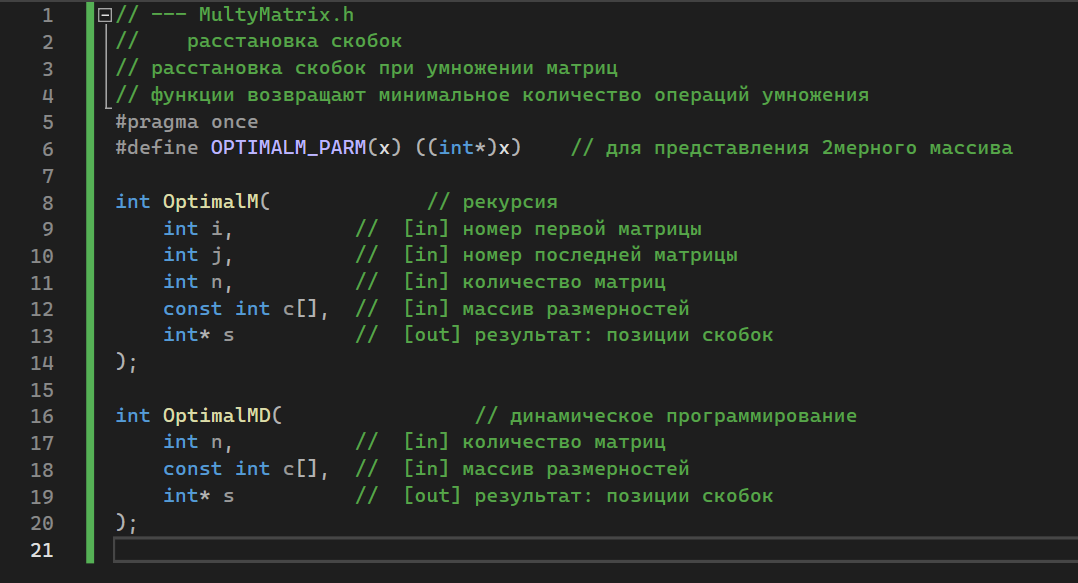


Рисунок 9 – содержание файла MultyMatrix.h

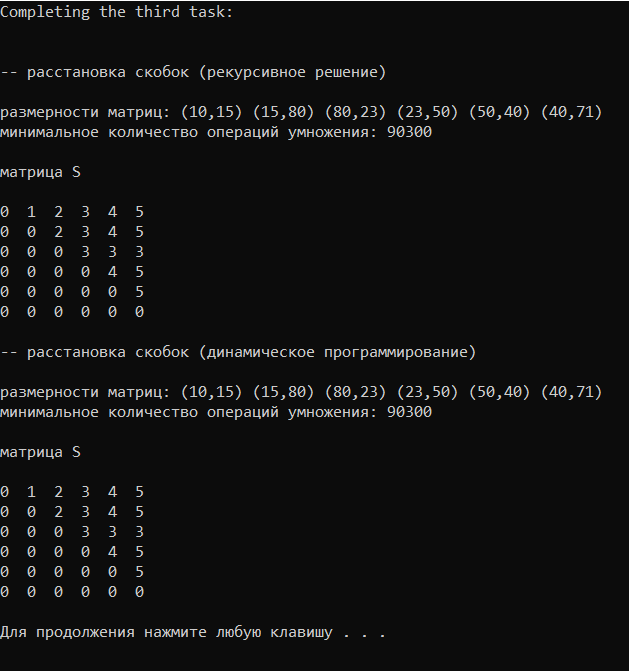


Рисунок 10 – результат выполнения программы

Протестировав программу на работоспособность, а также выполнив сравнительный анализ времени затраченного на решение задачи о расстановке скобок при перемножении матриц для двух методов решения, можно заметить, что динамический алгоритм работает с той же скоростью, что и рекурсивный алгоритм.

**Вывод:** динамический подход к решению задач позволяет выполнять их значительно быстрее, чем рекурсивный, особенно это будет заметно при решении задач с большим объёмом информации. Так же, я сделал следующие выводы:

1. Динамическое программирование - это мощный метод решения задач, позволяющий эффективно решать широкий спектр задач, которые не могут быть решены простыми алгоритмами.

2. Метод динамического программирования заключается в разбиении сложной задачи на более простые подзадачи, решение которых затем комбинируется в общее решение задачи.

3. Решение задач методом динамического программирования отличается высокой скоростью выполнения благодаря использованию кэширования вычислений и быстрой обработке данных.

4. В ходе выполнения лабораторной работы был исследован один из наиболее популярных методов динамического программирования - расстояние Левенштейна. Результаты экспериментов показали, что данная техника может быть очень эффективна для решения задач, связанных с обработкой текстовых данных.

5. Решение задач методом динамического программирования может быть осуществлено не только с помощью программирования на языке C++, но и на других языках, таких как Python или Java.