Министерство образования и науки РФ

Федеральное государственное автономное образовательное учреждение высшего образования

«Омский государственный технический университет»

|  |  |
| --- | --- |
| Факультет (институт) | *Информационных технологий и компьютерных систем* |
|  |  |
| Кафедра | *Прикладная математика и фундаментальная информатика* |
|  |  |

**Расчетно–графическая работа**

|  |  |
| --- | --- |
| по дисциплине | ***Алгоритмизация и программирование*** |
|  |  |
| на тему | Программная реализация задач |

Пояснительная записка

|  |  |
| --- | --- |
| **Шифр проекта** | 020–РГР–02.03.02–№ 12– ПЗ |

|  |  |  |  |  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- |
|  |  |  | **Студента** | | Лутонина Даниила Михайловича | | | | | |
|  |  |  |  | | фамилия, имя, отчество полностью | | | | | |
|  |  |  | Курс | 1 |  | Группа | | МО–241 | | |
|  |  |  |  |  |  | |  |  | |  |
|  | | | **Направление (специальность)** | | | | | ***02.03.03*** | | |
|  | | | Математическое обеспечение и администрирование информационных систем | | | | | | | |
|  |  |  | код, наименование | | | | | | | |
|  |  |  | Руководитель | | ***ст. преподаватель*** | | | | | |
|  |  |  | ученая степень, звание | | | | | |
|  |  |  | ***Федотова И.В.*** | | | | | | | |
|  |  |  | D:\Учёба\Важно\подпись.pngфамилия, инициалы | | | | | | | |
|  |  |  | Выполнил | | 20.05.2025 | | | | | |
|  |  |  | дата, подпись студента | | | | | |
|  |  |  |  | | | | | | | |
|  |  |  | **Работа защищена с количеством баллов** | | | | | |  | |



|  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- |
|  |  | дата, подпись руководителя |  |  |  |

Омск, 2025

Оглавление

[Введение 3](#_Toc199748868)

[Постановка задачи 1 4](#_Toc199748869)

[Ход решения задачи 1 5](#_Toc199748870)

[Постановка задачи 2 8](#_Toc199748871)

[Ход решения задачи 2 9](#_Toc199748872)

[Постановка задачи 3 13](#_Toc199748873)

[Ход решения задачи 4 14](#_Toc199748874)

[Постановка задачи 4 16](#_Toc199748875)

[Ход решения задачи 4 18](#_Toc199748876)

[Постановка задачи 5 23](#_Toc199748877)

[Ход решения задачи 5 24](#_Toc199748878)

[Заключение 30](#_Toc199748879)

[Список литературы 31](#_Toc199748880)

Введение

Напишем введение в виде небольшого рассказа.

В мире цифровых технологий, где правят биты и байты, появился язык программирования C#. Он унаследовал мощь своего предшественника – C++, но обладал более современным и удобным дизайном.

C# поддерживал объектно–ориентированный подход, что помогало разработчикам строить сложные, но при этом четко организованные программы. Его синтаксис был интуитивно понятным, благодаря чему язык быстро осваивали и новички, и профессионалы.

Изначально C# разрабатывался для платформы .NET, что открывало доступ к обширным библиотекам и инструментам. Это ускоряло процесс создания приложений для разных систем: Windows, веба и мобильных устройств.

Со временем C# завоевал доверие разработчиков и стал одним из самых востребованных языков. С его помощью писали как небольшие утилиты, так и масштабные корпоративные решения. Он славился стабильностью, высокой производительностью и способностью генерировать чистый, качественный код.

Несмотря на стремительное развитие технологий, C# не отставал: в него добавляли новые возможности, такие как асинхронность и элементы функционального программирования. Это делало язык еще более гибким и мощным.

Сегодня C# входит в число ведущих языков программирования. Его применяют в самых разных сферах: от игровой индустрии до искусственного интеллекта и облачных сервисов. Как универсальный и современный инструмент, он продолжает вдохновлять разработчиков на создание инновационных решений.

Конец.

Постановка задачи 1

Города

Максимальное время: 2,5 с.

Максимальная память: 4 MB

В некоторой стране население живет в городах. Города соединены дорогами. Каждая дорога начинается в одном городе и заканчивается в другом. Длины всех дорог известны. Из любого города через систему дорог можно добраться до любого другого города.

Будем называть расстоянием между двумя городами длину кратчайшего пути между ними.

Требуется найти наибольшее расстояние между городами в стране.

Входной файл

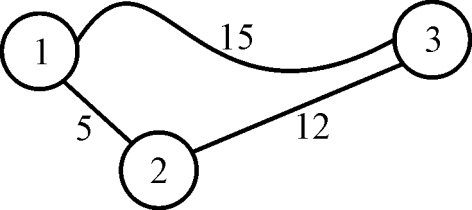
Первая строка содержит два целых числа N и M, разделенных пробелом – количество городов и дорог (2 <=N<= 400, 1 <=M<= 100000).

Следующие M строк содержат по три целых числа I, J, L – номера городов, соединенных дорогой (1 <=I, J<=N) и длина дороги (1 <=L<= 1000).

Выходной файл

Содержит единственное целое число – наибольшее расстояние между городами в стране.

Пример 1:



Ход решения задачи 1

Ввод и вывод данных происходит через консоль. Код программы представлен на рисунках 1, 2, 3.

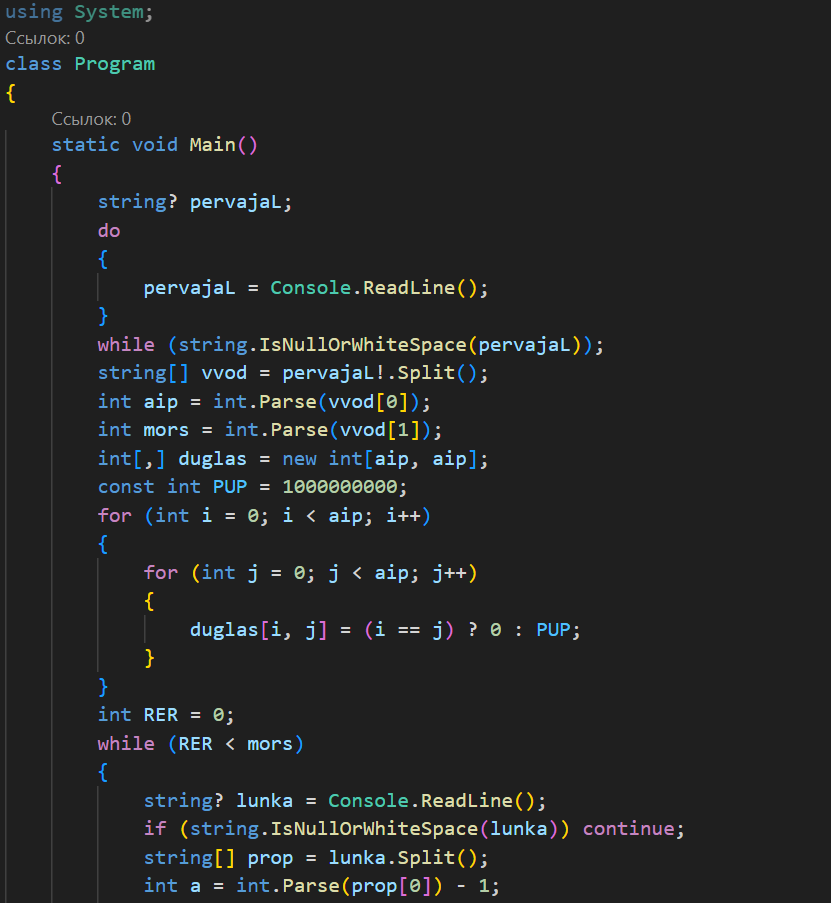


Рисунок 1 – Код программы задачи 1(1)

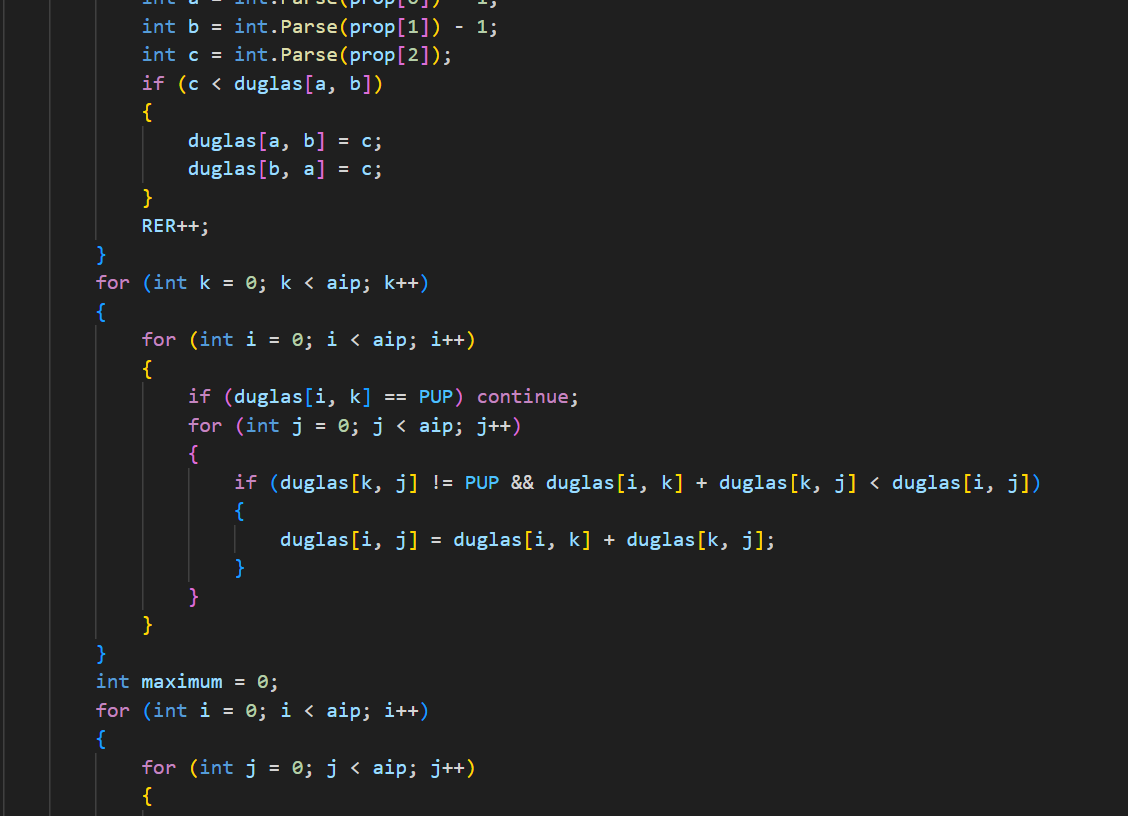


Рисунок 2 – Код программы задачи 1(2)

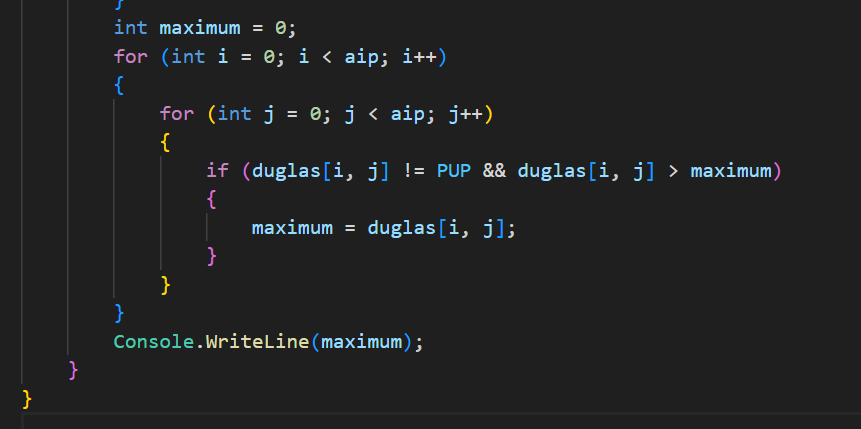


Рисунок 3 – Код программы задачи 1(3)

Данная программа реализует алгоритм Флойда–Уоршелла для нахождения кратчайших путей между всеми парами вершин в графе и вычисляет диаметр этого графа (максимальное из кратчайших расстояний между всеми вершинами). Примеры работы программы представлены на рисунке 4.

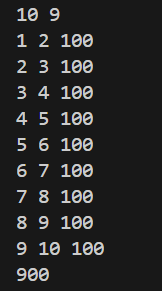
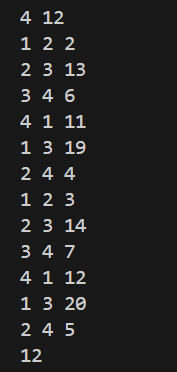
 

Рисунок 4 – Примеры работы программы задачи 1

Постановка задачи 2

Многоугольник

Максимальное время: 0,3 с.

Максимальная память: 4 MB

Дано N точек (3 <= N <= 1000) с целочисленными координатами, некоторые (или все) из которых являются вершинами выпуклого многоугольника, а остальные (или ни одной) находятся внутри него. Любые три точки не лежат на одной прямой. Определить площадь выпуклого многоугольника.

Модули координат не превосходят 30000. Считается, что кроме заданных вершин, других вершин нет.

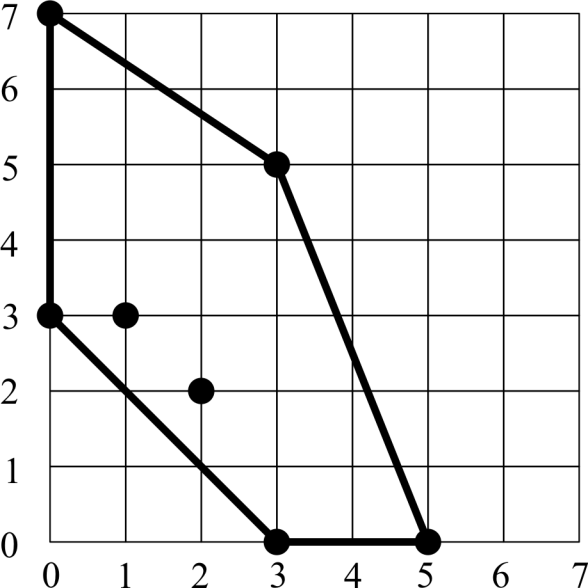
Входной файл

Первая строка содержит число точек N.

Следующие N строк содержат по два числа xi и yi, разделенных пробелом – координаты точек.

Выходной файл

Содержит площадь многоугольника с тремя знаками после запятой.



Ход решения задачи 2

Ввод и вывод данных происходит через консоль. Код программы представлен на рисунках 5, 6, 7.

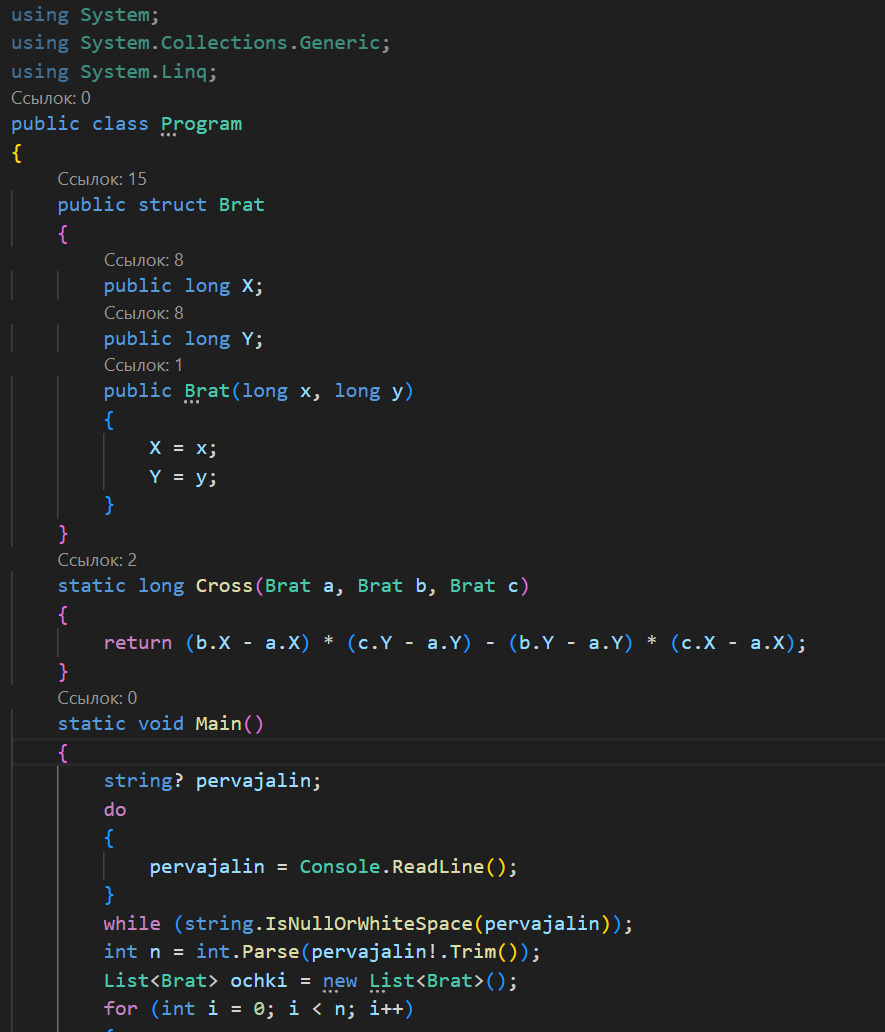


Рисунок 5 – Код программы задачи 2(1)

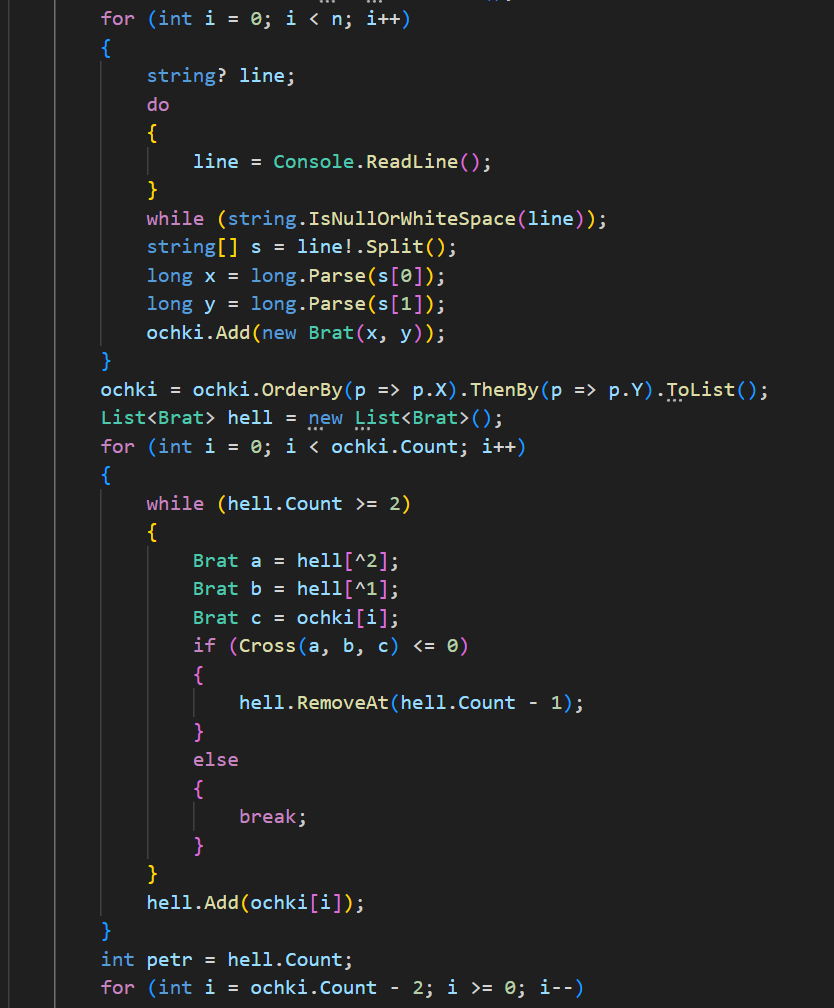


Рисунок 6 – Код программы задачи 2(2)

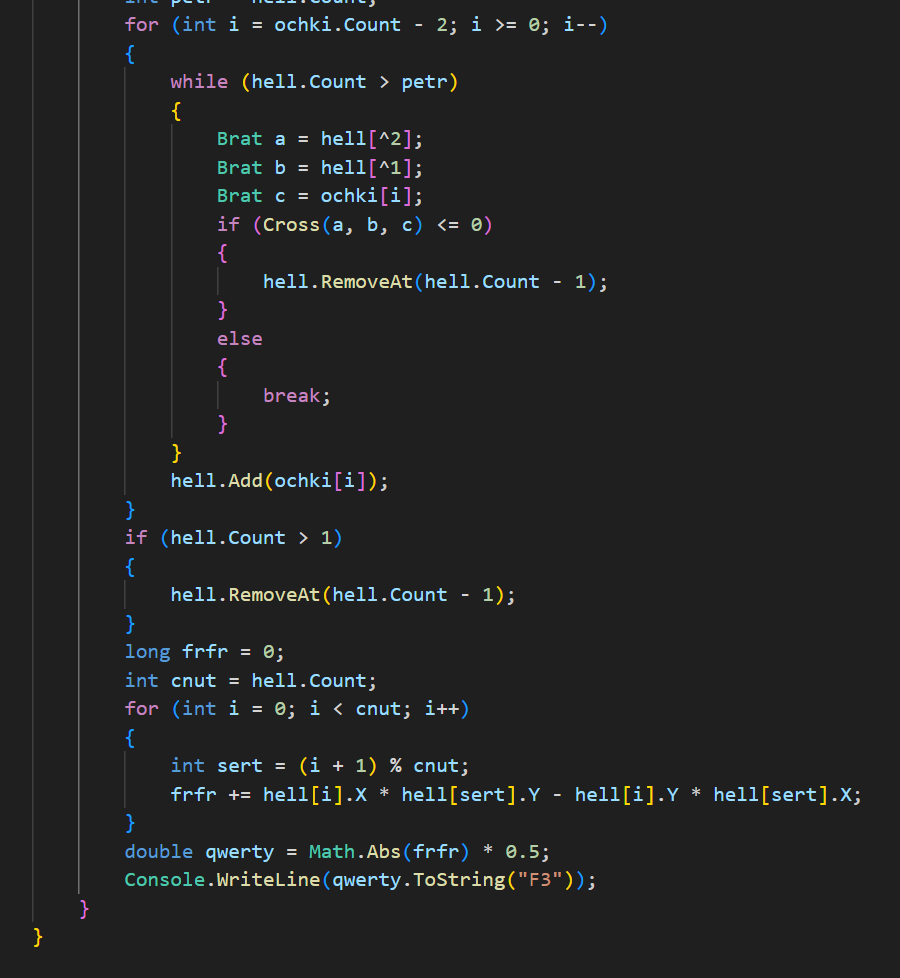


Рисунок 7 – Код программы задачи 2(3)

Данная программа вычисляет площадь выпуклой оболочки (convex hull) заданного множества точек на плоскости, используя алгоритм Эндрю (метод сканирования Грэхема с двумя проходами). Затем она находит площадь полученного многоугольника по формуле шнурования (Shoelace formula). Примеры работы программы представлены на рисунке 8.

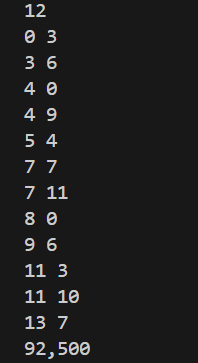
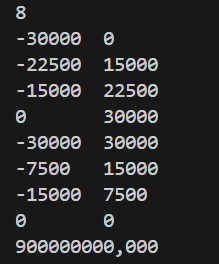
 

Рисунок 8 –Примеры работы программы задачи 2

Постановка задачи 3

Расстояние между городами

Максимальное время: 0,2 с.

Максимальная память: 16 MB

На поверхности некоторой планеты, представляющей собой идеальный шар радиуса R, заданы координаты двух городов в виде двух чисел – широты и долготы.

Требуется определить минимальное расстояние по поверхности планеты между этими городами.

Примечание

Пары (широта, долгота) уникальны.

Входной файл

Первая строка содержит два целых числа S1 и D1 – широту и долготу первого города в градусах (–90 <=S1<= 90; 0 <=D1<= 359).

Вторая строка содержит два целых числа S2 и D2 – широту и долготу второго города в градусах (–90 <=S2<= 90; 0 <=D2<= 359).

Третья строка содержит целое число R – радиус планеты (1 <=R<= 30000).

Выходной файл

Должен содержать одно вещественное число – минимальное расстояние между городами по поверхности планеты, выведенное с тремя знаками после запятой.

Ход решения задачи 4

Ввод и вывод данных происходит через консоль. Код программы представлен на рисунке 9.

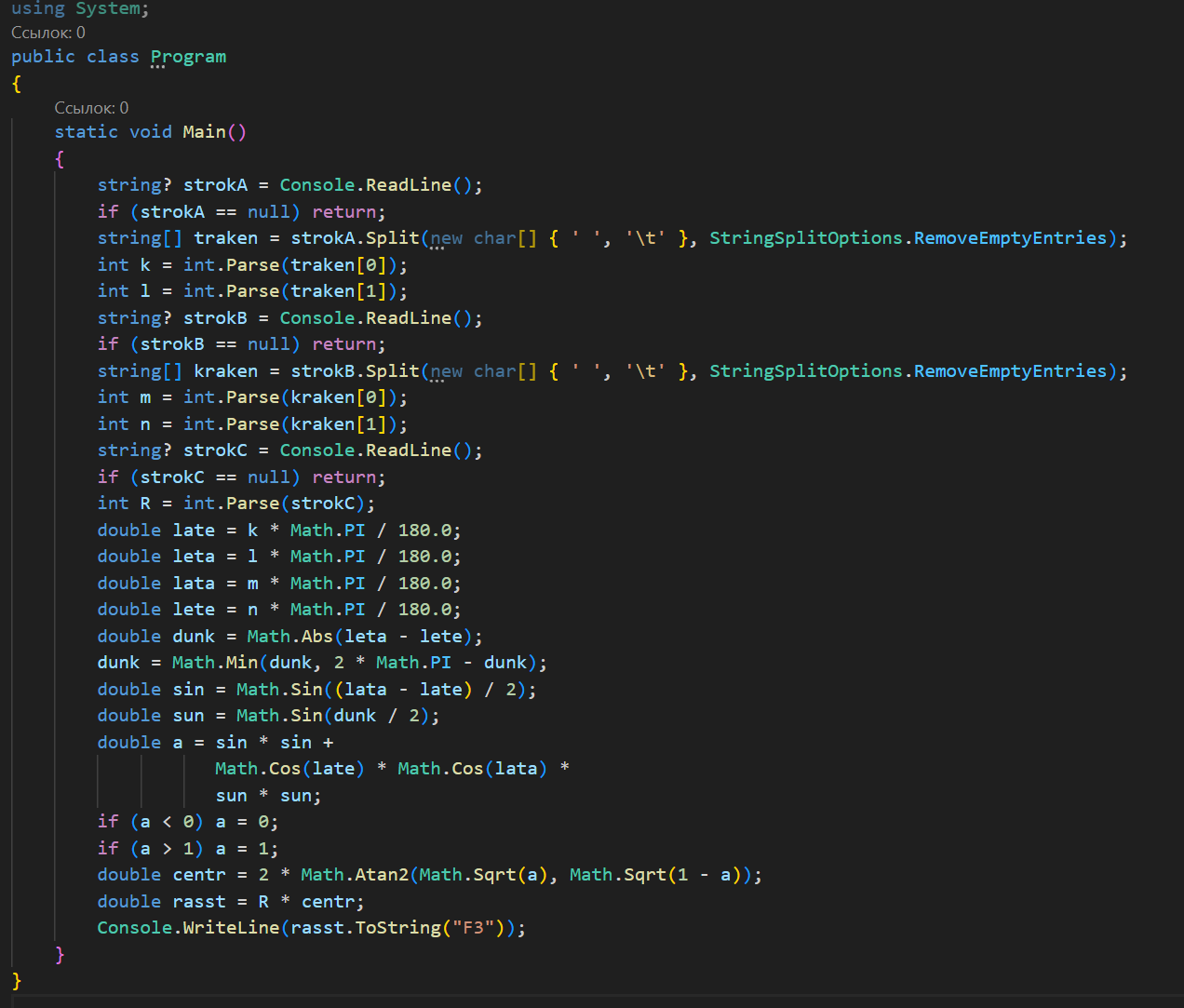


Рисунок 9 – Код программы задачи 3

Данная программа вычисляет расстояние между двумя точками на поверхности сферы (например, Земли) по их географическим координатам (широте и долготе) с использованием формулы гаверсинусов. Результат выводится с точностью до трёх знаков после запятой.

Шаги выполнения программы:

Ввод данных: Первая строка содержит координаты первой точки (широта k и долгота l в градусах). Вторая строка содержит координаты второй точки (широта m и долгота n в градусах). Третья строка содержит радиус сферы R.

Преобразование координат:

Все угловые координаты переводятся из градусов в радианы умножением на π/180.

Вычисление разницы долгот:

Находится абсолютная разница между долготами |l – n|. Корректируется на случай пересечения нулевого меридиана (берется минимальное значение между разницей и 2π – разница).

Применение формулы гаверсинусов:

Вычисляется промежуточная величина a по формуле: a = sin²(Δφ/2) + cos(φ1) \* cos(φ2) \* sin²(Δλ/2) где: Δφ – разница широт, Δλ – разница долгот, φ1, φ2 – широты точек.

Вычисление центрального угла:

Вычисляется центральный угол между точками по формуле: c = 2 \* arctg2(√a, √(1–a))

Расчет расстояния:

Итоговое расстояние вычисляется умножением центрального угла на радиус сферы: distance = R \* c

Вывод результата:

Программа выводит расстояние с точностью до трёх знаков после запятой.

Примеры работы программы представлены на рисунке 10.

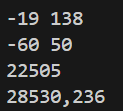
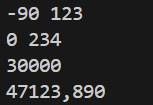
 

Рисунок 10 – Примеры работы программы задачи 3

Постановка задачи 4

Игра "Города"

Максимальное время: 0,8 с.

Максимальная память: 16 MB

Известна игра “в города”, в которой по очереди называются уникальные названия городов, причем каждое следующее название города должно начинаться с буквы, на которую заканчивается предыдущее название города.

Будем называть цепочкой названий городов два и более названия, составленные по описанному выше принципу.

Будем называть закольцованной цепочкой названий городов такую цепочку, в которой название первого города в цепочке начинается на ту же букву, на которую заканчивается название последнего города в цепочке.

Для заданного набора уникальных названий городов требуется определить количество и длины (количество названий в цепочке) закольцованных цепочек, построенных по следующему правилу:

* из исходного набора названий строится первая закольцованная цепочка максимально возможной длины;
* из названий, не вошедших в первую закольцованную цепочку, строится вторая закольцованная цепочка максимально возможной длины;
* из названий, не вошедших в предыдущие закольцованные цепочки, строится очередная закольцованная цепочка максимально возможной длины;
* построение заканчивается, когда все названия городов использованы.

Примечание: гарантируется, что все названия городов входят в какую–нибудь цепочку.

Входной файл

Каждая строка входного файла содержит по одному названию города. Количество названий городов не превышает 30000. Название города содержит не более 15 символов. В название могут входить только маленькие латинские буквы.

Выходной файл

Первая строка должна содержать целое число N – количество построенных закольцованных цепочек.

Следующие N строк должны содержать длины (количество названий в цепочке) закольцованных цепочек, выведенные в порядке построения цепочек (т.е. по убыванию длины).

Ход решения задачи 4

Ввод и вывод данных происходит через консоль. Код программы представлен на рисунке 11, 12, 13.

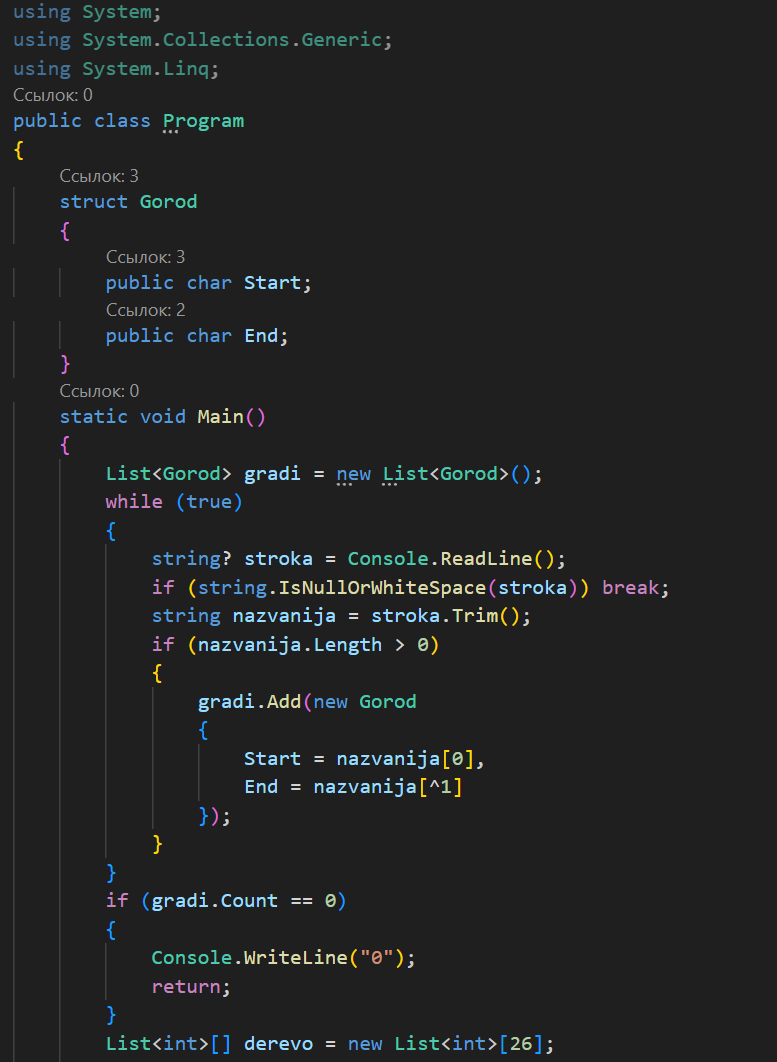


Рисунок 11 – Код программы задачи 4(1)

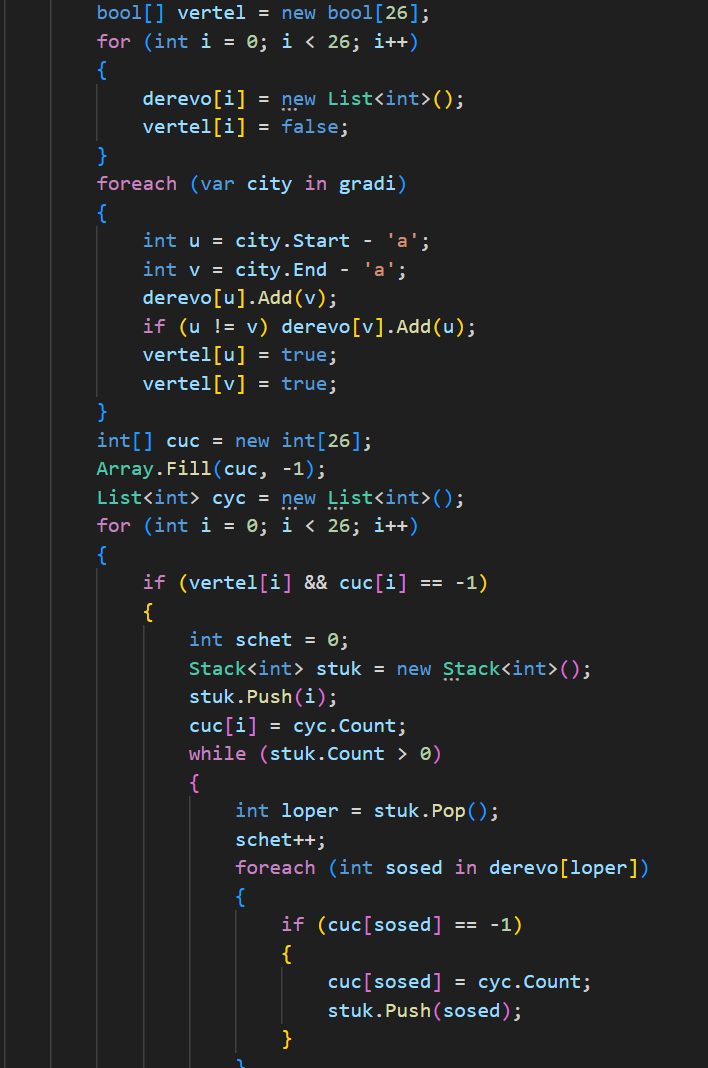


Рисунок 12 – Код программы задачи 4(2)

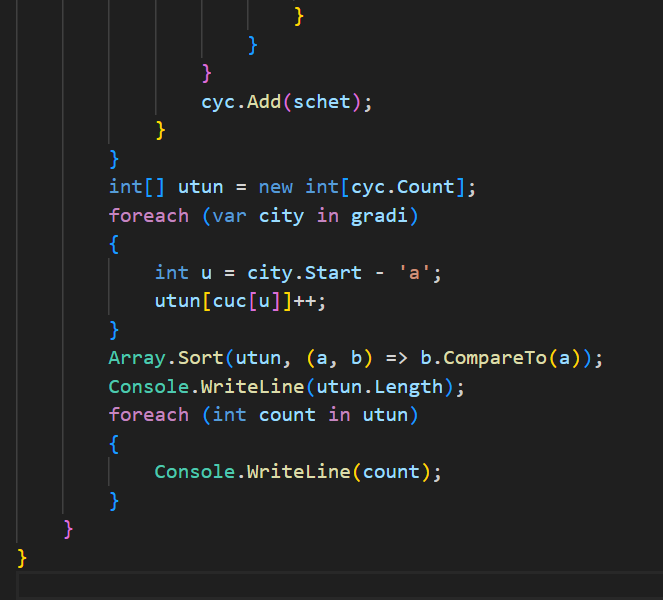


Рисунок 13 – Код программы задачи 4(3)

Данная программа анализирует набор названий городов и строит граф связей между ними, где города представлены как вершины графа, а связи между ними – как рёбра. Программа затем находит количество компонент связности в этом графе и для каждой компоненты выводит количество рёбер (связей между городами).

Шаги выполнения программы:

Ввод данных:

Программа считывает названия городов построчно до пустой строки. Для каждого города запоминается его первая и последняя буква (создаётся структура Gorod).

Построение графа:

Создаётся список смежности derevo (26 элементов по количеству букв английского алфавита). Для каждой пары букв (начало и конец названия города) добавляются рёбра в граф. Граф делается неориентированным (если город A связан с B, то и B связан с A).

Поиск компонент связности:

Используется алгоритм обхода в глубину (DFS) для нахождения всех компонент связности. Каждой компоненте присваивается уникальный идентификатор. Запоминается размер каждой компоненты (количество вершин в ней).

Подсчёт рёбер в компонентах:

Для каждой компоненты подсчитывается количество рёбер (связей между городами). Результаты сортируются по убыванию количества рёбер.

Вывод результата:

Сначала выводится количество компонент связности. Затем для каждой компоненты выводится количество рёбер в ней.

Пример работы представлен на рисунке 14, а на рисунке 15 представлена краткая блок схема задачи.

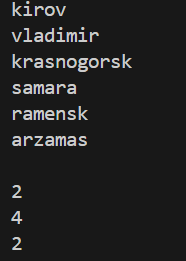


Рисунок 14 – Пример работы программы задачи 4

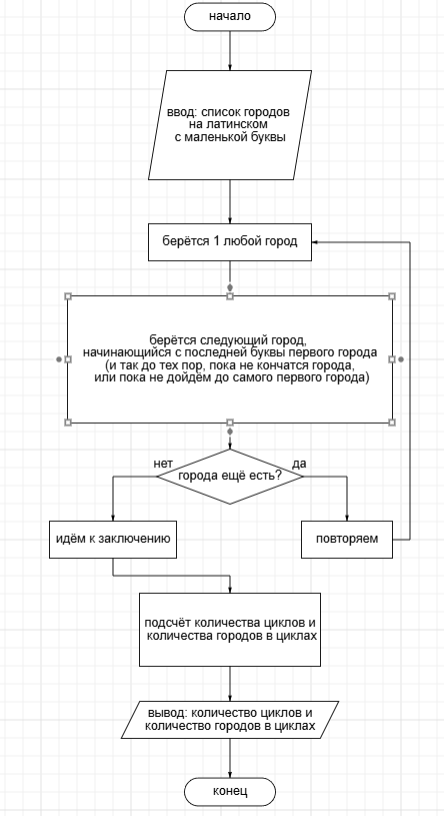


Рисунок 15 –Блок схема функции задачи 4

Постановка задачи 5

Задание 12

Фермер Джон хочет, как можно дешевле организовать свою систему распределения воды, но он не хочет, чтобы его конкурент фермер Плуто мог предсказать маршруты, которые он выбирает. Фермер Джон знает, что такая задача обычно требует самого дешевого способа прокладки труб поэтому он решил использовать второй по стоимости способ.

Дан список всех двунаправленных труб, которые могут соединять множество из W (3 <= W <= 2 000) станций с водой (каждая из которых может быть встроена в колодец). Необходимо найти второй из самых дешевых способов соединить насосные станции, используя не более чем Р (Р <= 20 000) труб с заданной стоимостью каждой трубы. Не должно быть трубы, соединяющей станцию саму с собой. Не должно быть двух труб, соединяющих дважды одну и ту же пару станций. Гарантируется, что есть только один самый дешевый способ распределить воду, и что существует, как минимум, два способа распределить воду. Все стоимости – положительные числа. Водная станция идентифицируется своим номером – целым числом в диапазоне l..W.

Формат входных данных

строка 1 – два разделенных пробелом целых числа, W u P;

строки 2..P + 1 – каждая строка описывает одну трубу и содержит 3 числа, разделенных пробелом, – номера станций начала и конца трубы, а также стоимость этой трубы.

Формат выходных данных

Одна строка, содержащая целое число – вторая минимальная стоимость конструирования системы распределения воды.

Ход решения задачи 5

Ввод и вывод данных происходит через консоль. Код программы представлен на рисунках 1, 2, 3, 4, 5.

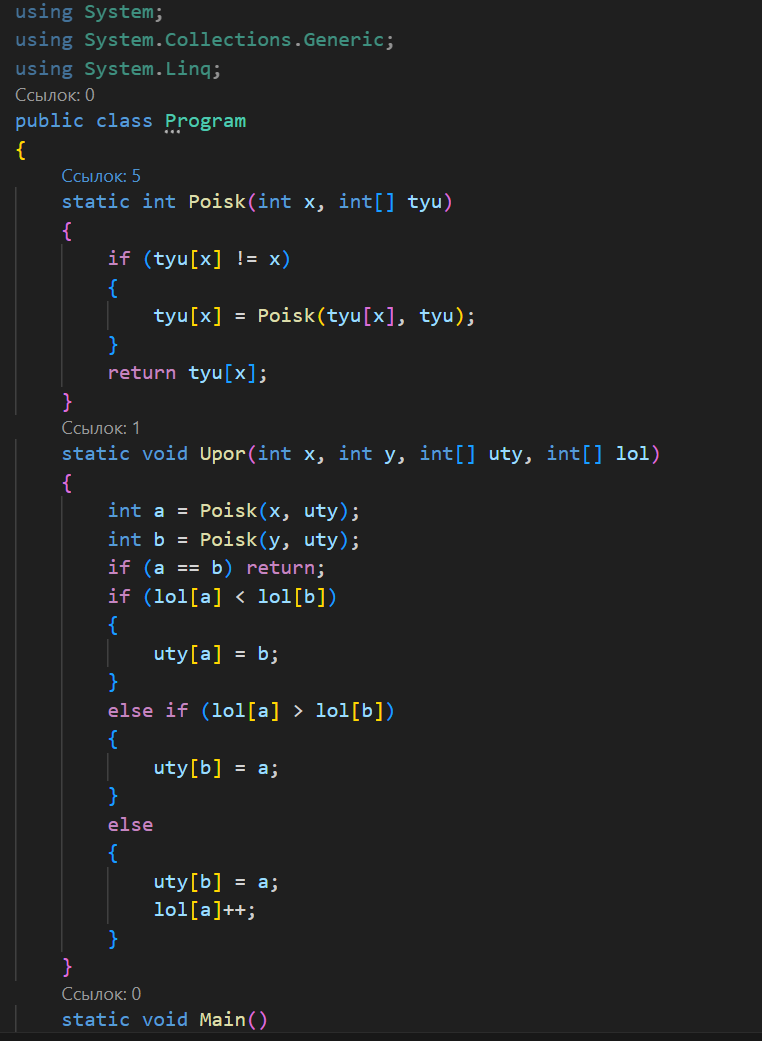


Рисунок 16 – Код программы задачи(1)



Рисунок 17 – Код программы задачи(2)

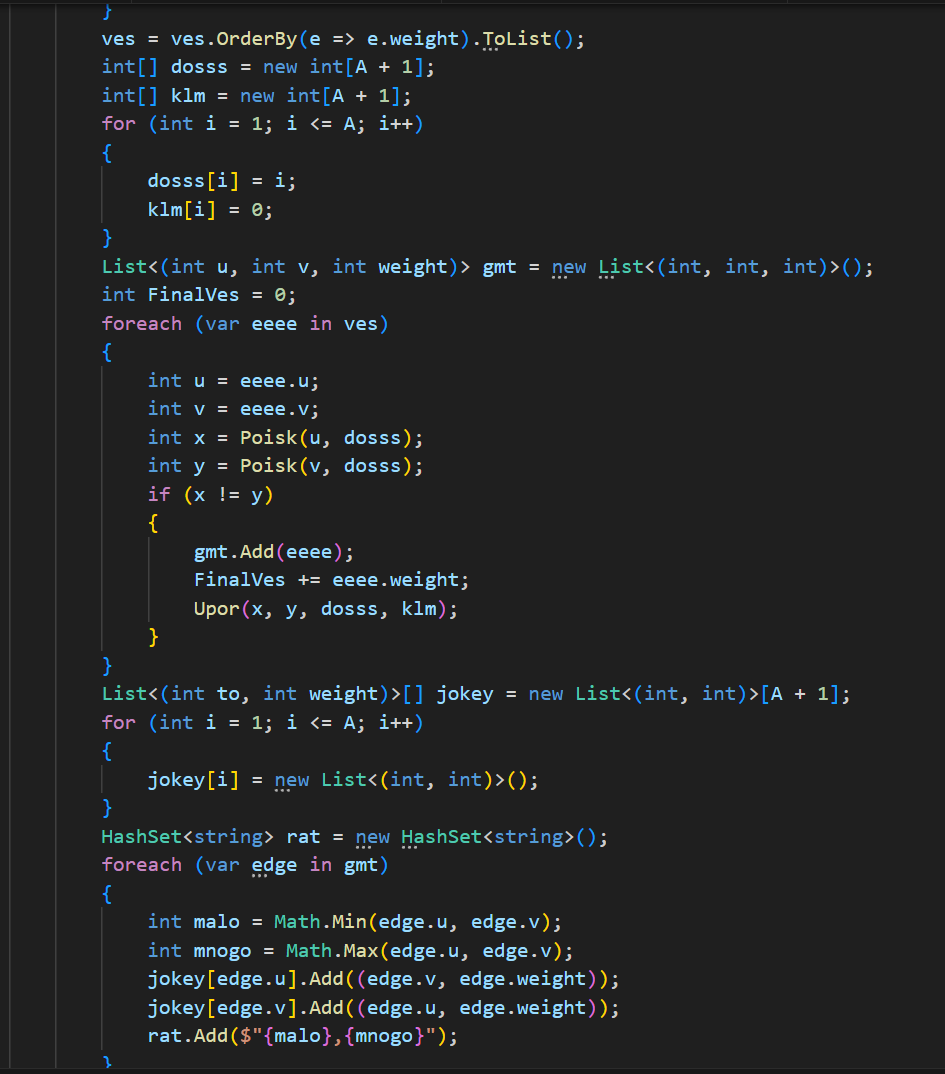


Рисунок 18 – Код программы задачи(3)

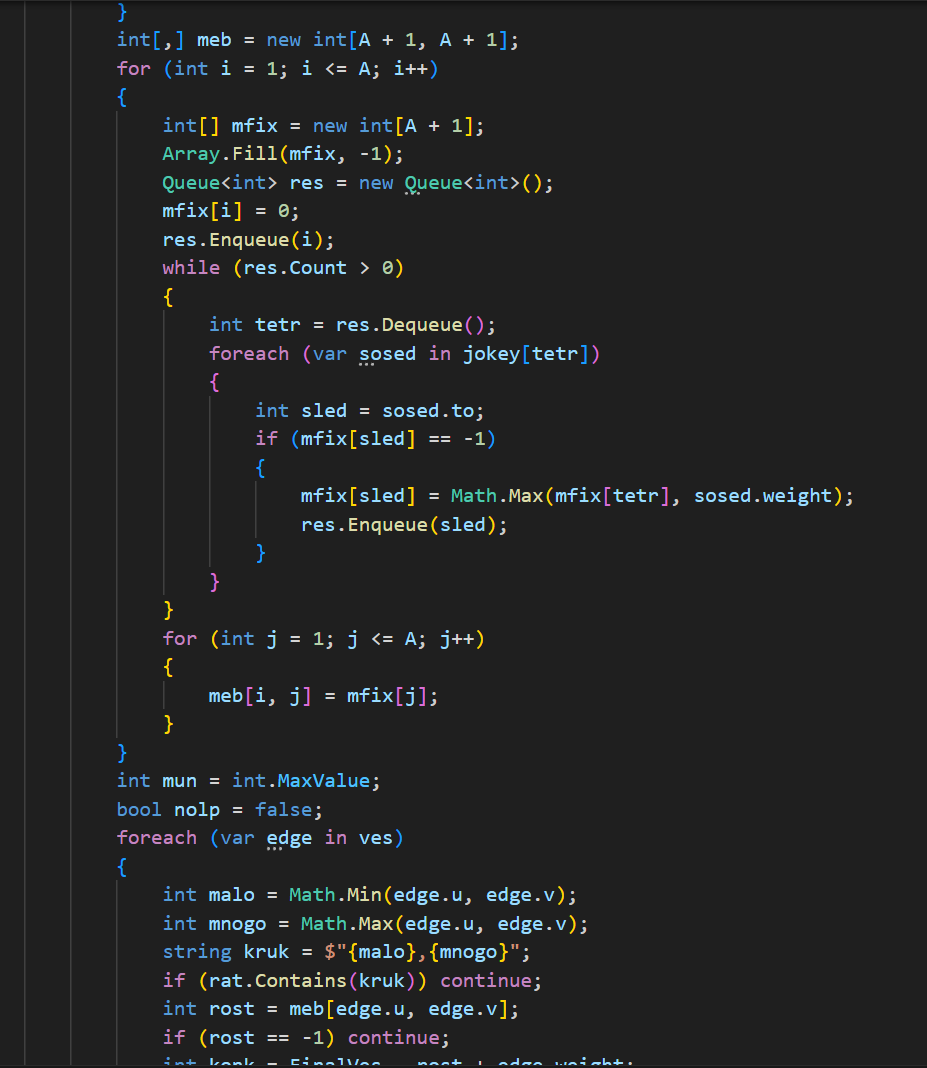


Рисунок 19 – Код программы задачи(4)

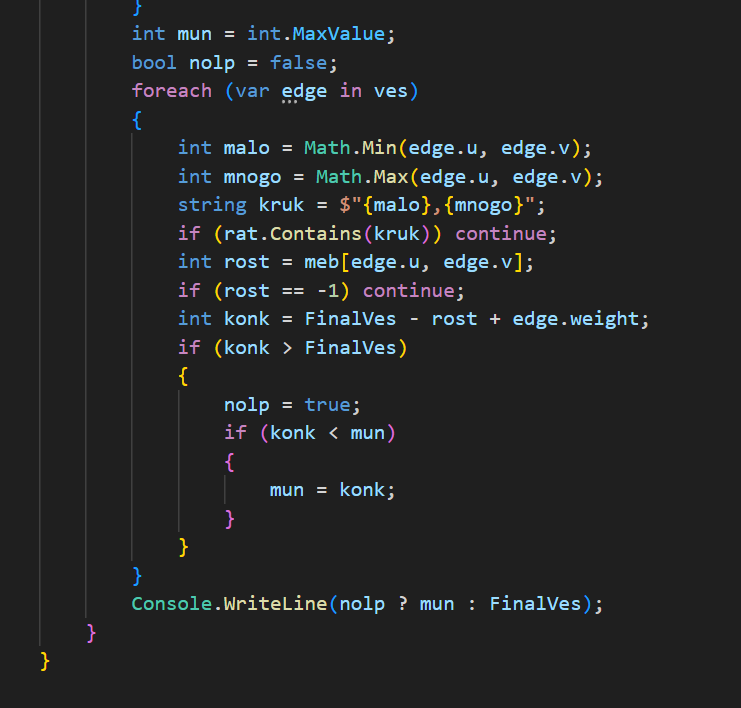


Рисунок 20 – Код программы задачи(5)

Данная программа реализует алгоритм поиска второго минимального остовного дерева (Second–best Minimum Spanning Tree) в связном взвешенном графе. Основной алгоритм состоит из нескольких ключевых этапов:

1. Построение минимального остовного дерева (MST) с использованием алгоритма Крускала с системой непересекающихся множеств (DSU):

– Сортировка всех рёбер по весу;

– Пошаговое добавление рёбер с проверкой на образование циклов;

– Использование эвристик сжатия пути и объединения по рангу.

2. Поиск второго минимального остовного дерева:

– Для каждого ребра, не входящего в MST, вычисляется альтернативное решение;

– Находится максимальное ребро в пути между вершинами текущего ребра в MST;

– Рассчитывается потенциальный вес нового дерева при замене максимального ребра на текущее.

3. Специальные случаи:

– Обработка несвязных графов;

– Проверка на существование второго MST;

– Вывод либо веса второго MST, либо веса основного MST, если второго не существует.

Программа демонстрирует эффективное использование:

– Алгоритмов на графах (Крускал, BFS);

– Структур данных (DSU, очереди, хеш–множества);

– Методов оптимизации (эвристики объединения, кеширование результатов).

Примеры работы программы представлены на рисунке 6.

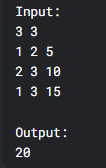
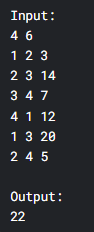
 

Рисунок 21 – Примеры работы программы задачи

Заключение

В процессе решения четырёх различных задач на C# удалось не только успешно применить возможности языка и его стандартных библиотек, но и значительно расширить практические навыки программирования. Каждая задача требовала особого подхода, что позволило глубже освоить принципы разработки программного обеспечения.

Работа с этими задачами дала ценный опыт применения теоретических знаний на практике, способствовала развитию аналитического мышления и алгоритмической логики. Приобретённые умения несомненно будут полезны как в учебном процессе, так и в профессиональной деятельности разработчика.

Постоянное совершенствование навыков и готовность к освоению новых технологий действительно являются важнейшими составляющими успеха в IT–индустрии. Решение практических задач на C# стало отличной возможностью для профессионального роста и закрепления ключевых концепций программирования.

Список литературы

1. C# для начинающих <https://lesmatveev.narod.ru/knigi_proect/ci_sharp.pdf> (дата обращения: 10.05.2025);
2. Metanit <https://metanit.com/> (дата обращения: 16.05.2025);
3. Habr <https://habr.com/ru/articles/> (дата обращения: 19.05.2025);
4. Otus <https://otus.ru/nest/post/1891/> (дата обращения: 26.05.2025);
5. Wikipedia <https://ru.wikipedia.org> (дата обращения 12.05.2025).