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–№ 20 – ПЗ |

|  |  |  |  |  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- |
|  |  |  | **Студента** | | Чалова Владимира Евгеньевича | | | | | |
|  |  |  |  | | 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.*** | | | | | | | |
|  |  |  | фамилия, инициалы | | | | | | | |
|  |  |  | Выполнил | |  | | | | | |
|  |  |  | дата, подпись студента | | | | | |
|  |  |  |  | | | | | | | |
|  |  |  | **Работа защищена с количеством баллов** | | | | | |  | |

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

Омск 2025

Содержание

[**Введение** 3](#_Toc198999259)

[**Постановка задачи «Прямоугольник»** 4](#_Toc198999260)

[**Ход решения задачи «Прямоугольник»** 5](#_Toc198999261)

[**Постановка задачи «Расстояние между городами»** 8](#_Toc198999262)

[**Ход решения задачи «Расстояние между городами»** 9](#_Toc198999263)

[**Постановка задачи «Минное поле»** 12](#_Toc198999264)

[**Ход решения задачи «Минное поле»** 13](#_Toc198999265)

[**Постановка задачи «Города»** 17](#_Toc198999266)

[**Ход решения задачи «Города»** 18](#_Toc198999267)

[**Заключение** 22](#_Toc198999268)

[**Список литературы** 23](#_Toc198999269)

**Введение**

В современном обществе информационные технологии занимают центральное место во множестве областей. Одним из самых популярных направлений считается разработка программного обеспечения, где язык программирования C# уверенно удерживает свои позиции на пике популярности среди разработчиков.

C# – это объектно-ориентированный язык программирования, созданный компанией Microsoft, который применяется для разработки приложений на платформе .NET. Его сочетание простоты и функциональных возможностей делает его превосходным инструментом для решения широкого спектра задач.

В данной расчетно-графической работе мы сосредоточимся на изучении как базовых, так и более сложных аспектов языка C#, а также на его практическом применении при решении олимпиадных задач. В процессе работы будут освещены ключевые понятия и концепции C#, такие как манипуляции со строками и числами, использование логических конструкций и функций. Кроме того, особое внимание будет уделено процессам тестирования, отладки и оптимизации кода.

**Постановка задачи «Прямоугольник»**

Имеется прямоугольная область размером **N**×**M** клеток, некоторые из которых закрашены.

Требуется найти прямоугольник максимальной площади, не содержащий закрашенные клетки. Стороны прямоугольника должны быть параллельны краям прямоугольной области.

**Входной файл** содержит:

* в первой строке два числа **N** и **M** – число столбцов и строк прямоугольной области соответственно (1 ≤ **N**, **M** ≤ 250);
* во второй строке число **K** – количество закрашенных клеток (0 ≤ **K**  ≤ **(N**\***M**-1));
* следующие **K** строк содержат по 2 числа **X** и **Y** – координаты (столбец и строка соответственно) закрашенной клетки.

**Выходной файл** должен содержать площадь найденного прямоугольника.

**Ход решения задачи «Прямоугольник»**

Для решения задачи необходимо найти прямоугольник максимальной площади, который может быть размещен внутри заданной прямоугольной области, не включающий ни одной закрашенной клетки. При этом стороны прямоугольника должны быть параллельны краям области.

**Принцип работы:**

* Инициализация высот столбцов: Для каждой строки мы поддерживаем массив heights, где heights[col] представляет собой количество последовательных пустых клеток (не закрашенных) над текущей клеткой (row, col) включительно.
* Обновление высот столбцов: Для каждой строки мы обновляем массив heights:
* Если клетка (row, col) закрашена, то heights[col] = 0 (так как над ней не может быть пустого прямоугольника).
* Если клетка пустая, то heights[col] увеличивается на 1 (так как мы можем расширить прямоугольник вверх).
* Поиск максимального прямоугольника в текущей строке: Для каждой строки после обновления heights мы ищем максимальный прямоугольник, который может быть построен из столбцов с текущими высотами:
* Для каждого столбца col мы ищем максимальный прямоугольник, который может быть построен, начиная с этого столбца и расширяясь вправо.
* При этом мы поддерживаем минимальную высоту (minHeight) среди всех рассмотренных столбцов, так как высота прямоугольника определяется самым низким столбцом в нем.
* Площадь текущего прямоугольника вычисляется как minHeight \* (right - col + 1).
* Обновление максимальной площади: На каждом шаге мы сравниваем текущую площадь с максимальной найденной площадью и обновляем ее, если текущая площадь больше.

Код программы представлен на рисунке 1.1.

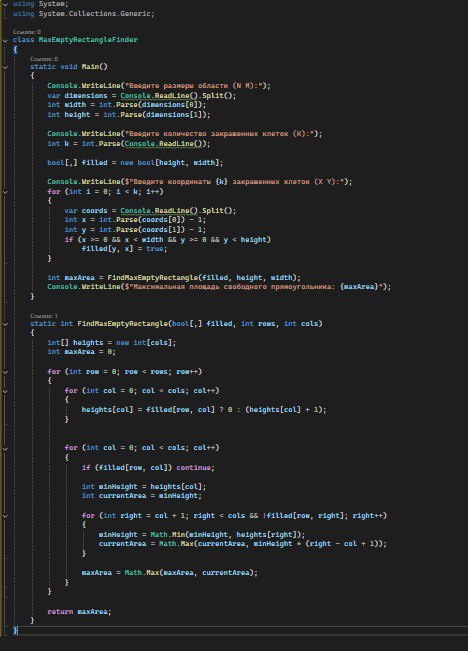


Рисунок №1.1 – Код программы «Прямоугольник».

Проверим работу программы на тестах (рис 1.2).

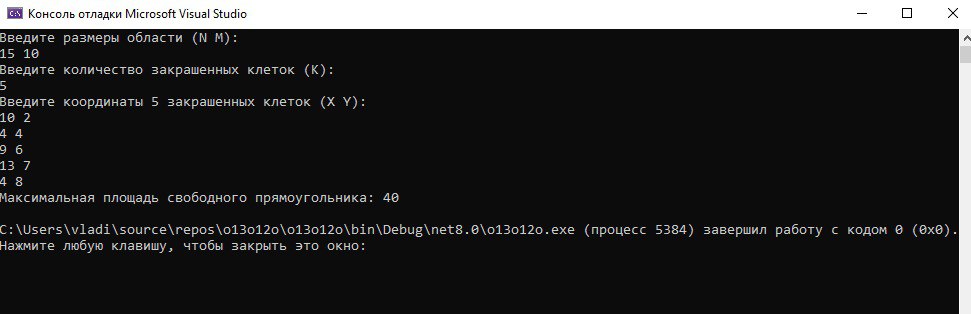


Рисунок №1.2 – Тест для задачи «Прямоугольник».

**Постановка задачи «Расстояние между городами»**

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 «Расстояние между городами»**

Для решения задачи требуется вычислить минимальное расстояние между двумя точками на поверхности сферы, заданными их широтой и долготой. Города расположены на идеальной сфере радиуса R, поэтому расстояние определяется длиной кратчайшей дуги большого круга между ними. Сначала считываются входные данные: две строки с координатами городов в градусах (широта и долгота) и третья строка с радиусом планеты, которые преобразуются в числовой формат для дальнейших вычислений. Затем широта и долгота переводятся из градусов в радианы, так как тригонометрические функции обычно работают с радианами. После этого рассчитывается угол между двумя точками с помощью формулы сферической тригонометрии, где косинус угла равен сумме произведений синусов широт и произведений косинусов широт на косинус разности долгот. Чтобы избежать ошибок округления, значение косинуса ограничивается диапазоном от -1 до 1. Далее вычисляется угол как арккосинус полученного значения, после чего он умножается на радиус сферы для получения искомого расстояния – длины дуги по поверхности планеты между городами.

Результат выводится с точностью до трех знаков после десятичной точки согласно требованиям задачи.

Код программы представлен на рисунке 2.1.



Рисунок №2.1 – Код программы «Расстояние между городами».

Проверим работу программы на тестах (рис 2.2).

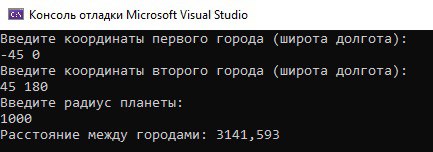


Рисунок №2.2 – Тест для задачи «Расстояние между городами».

**Постановка задачи «Минное поле»**

Минное поле представляет собой прямоугольник, разделенный на квадратные зоны одинакового размера. С помощью средств современной разведки было выяснено, сколько мин заложено в каждой зоне. Для разминирования каждой мины требуется одинаковое время. Для прохода по зоне, необходимо разминировать все мины на ней.

Требуется найти один из вариантов прохода в минном поле, требующий минимального времени для разминирования.

**Примечания:**

* размер поля **N**\***M**, 0 < **N** ≤ 200, 0 < **M** ≤ 400;
* количество мин в каждой зоне от 0 до 100.

**Входной файл** содержит:

* в первой строке числа **N** и **M**, разделенные пробелом (количество рядов на минном поле и зон в ряде соответственно);
* в следующих **N** строках содержится по **M** целых чисел, разделенных пробелом – количество мин в зонах **i**-го ряда (1 ≤ **i** ≤ **N**).

**Выходной файл** должен содержать **N** строк. В каждой строке указывается номер зоны в **i**-м ряду (1 ≤ **i** ≤ **N**), через которую лежит найденный проход.

**Ход решения задачи «Минное поле»**

Для решения данной задачи потребуется определить маршрут через минное поле, минимизирующий суммарное количество мин, которое необходимо устранить. Поле представляет собой прямоугольную сетку размерами N на M, где в каждой ячейке указано число мин. Передвижение осуществляется по последовательным рядам сверху вниз, при этом из текущей ячейки разрешено переходить либо прямо вниз, либо по диагоналям влево-вниз или вправо-вниз. Цель – пройти из любой ячейки первого ряда в любой ячейке последнего ряда так, чтобы сумма мин по выбранному пути была минимальной.

**Принцип работы:**

* На вход подаются размеры поля (N и M), а затем N строк по M чисел, показывающих количество мин в каждой зоне. В результате требуется вывести N строк, в которых указаны номера ячеек в выбранном оптимальном маршруте.
* Для решения задачи применяется динамическое программирование. Создается двумерный массив, одновременно с этим – массив для хранения предыдущих позиций, чтобы потом построить путь назад. Вначале заполняется первый ряд значениями из исходных данных – в них сумма мин равна просто количеству в соответствующих ячейках, так как других вариантов сюда попасть нет.
* Далее для каждого следующего ряда требуется рассчитать минимальную сумму мин для достижения каждой ячейки. Это делается путём анализа всех возможных переходов из ячеек предыдущего ряда: вверх, диагональ влево-вниз или вправо-вниз. При этом выбирается вариант, дающий наименьшее общее количество мин, и запоминается индекс ячейки-отправной точки, чтобы позднее можно было восстановить маршрут.
* После обработки всех рядов, в последнем ряду выбирается ячейка с минимальной суммой мин. Далее с помощью запомненных индексов восстанавливается обратный маршрут – путь минимального количества мин – начиная с выбранной ячейки в последнем ряду и движением назад к первому.
* В конце выводятся номера ячеек, через которые проходит оптимальный маршрут, начиная с первого ряда и заканчивая последним. Индексы ячеек в выводе нумеруются с единицы, чтобы соответствовать формату.

Код программы представлен на рисунке 3.1.

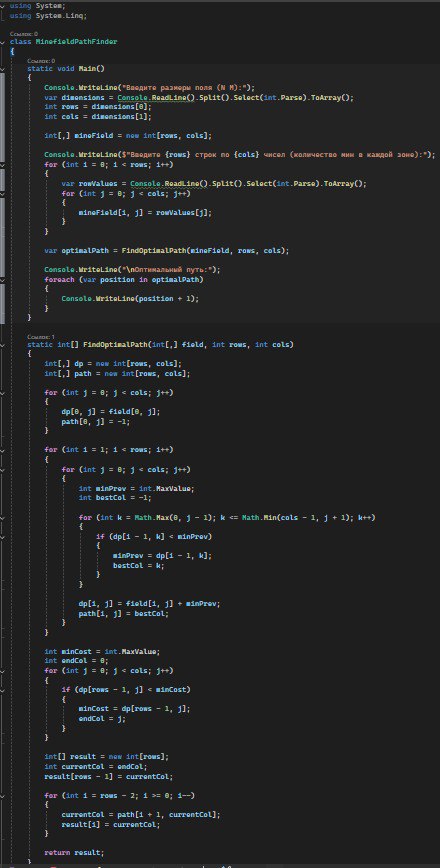


Рисунок №3.1 – Код программы «Минное поле».

Проверим работу программы на тестах (рис 3.2).

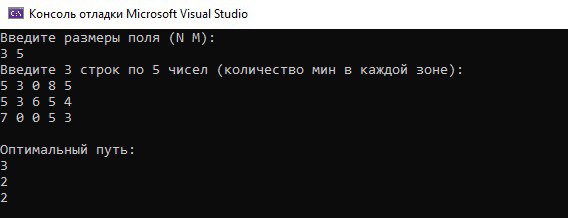


Рисунок №3.2 – Тест для программы «Минное поле».

**Постановка задачи «Города»**

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 расстояние 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.

**Ход решения задачи «Города»**

Для решения задачи необходимо вычислить диаметр графа, представляющего сеть городов, соединённых дорогами. Под диаметром графа понимается наибольшее из кратчайших расстояний между всеми возможными парами городов. Решение основано на алгоритме Флойда для поиска кратчайших путей между всеми парами вершин.

**Принцип работы:**

* Программа начинается с ввода исходных данных: количества городов (вершин графа) и дорог (рёбер графа). Затем создаётся матрица смежности, где каждая ячейка хранит длину дороги между соответствующими городами. Для пар городов, не соединённых напрямую дорогой, устанавливается условно бесконечное значение (достаточно большое число, чтобы исключить его из рассмотрения при поиске минимумов).
* Основная часть решения заключается в последовательном применении алгоритма Флойда. Этот метод динамического программирования постепенно улучшает оценки кратчайших путей между всеми парами городов, рассматривая промежуточные вершины. На каждом шаге алгоритм проверяет, может ли путь между городами i и j быть сокращён путём прохода через промежуточный город k. Если такое сокращение возможно, соответствующее значение в матрице расстояний обновляется.
* После вычисления всех кратчайших путей программа находит максимальное значение в полученной матрице, которое и является диаметром графа. Это значение представляет наибольшее из минимальных расстояний между всеми парами городов в сети, что соответствует самому длинному "прямому" пути, который придётся преодолеть при перемещении между наиболее удалёнными городами.
* Результат работы программы – это целое число, показывающее максимальное расстояние между самой удалённой парой городов в сети с учётом оптимальных маршрутов.

Код программы представлен на рисунке 4.1.



Рисунок №4.1 – Код программы задачи «Города».

Проверим работу программы на тестах (рис.4.2 и 4.3).

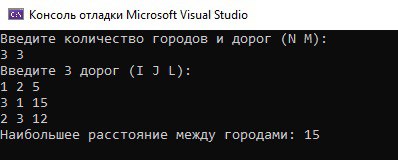


Рисунок №4.2 – Тест 1 для программы «Минное поле».

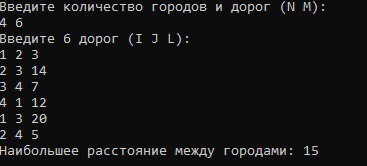


Рисунок №4.3 – Тест 2 для программы «Минное поле».

**Заключение**

В ходе выполнения задания был изучен базовый и продвинутый функционал языка программирования C#, а также применены эти знания для решения конкретных задач. Рассматривались ключевые концепции языка, такие как использование классов, объектов и методов. Также был изучен потенциал стандартных библиотек и освоена среда разработки Visual Studio. В процессе работы был создан ряд программ, демонстрирующих разнообразные возможности и особенности языка C#, с последующей их проверкой и отладкой для устранения ошибок и недочетов.

В результате освоены навыки программирования на C#, приобретен опыт разработки и тестирования программного обеспечения, который может быть полезен в будущем при реализации различных проектов или в профессиональной деятельности.

**Список литературы**

1. Троелсен Э., Джепикс Ф. C# 10 и .NET 6. Современная кроссплатформенная разработка. – М.: Диалектика, 2023. – 1088 с. (дата обращения19.05.2025).  
   (Фундаментальное руководство по современным возможностям языка)
2. Скит Д. C# для профессионалов. Тонкости программирования. – СПб.: Питер, 2023. – 608 с. (дата обращения 18.05.2025).  
   (Глубокий разбор сложных аспектов языка и внутреннего устройства)
3. Лафоре Р**.** Структуры данных и алгоритмы в C#. – М.: Эксмо, 2022. – 992 с. (дата обращения 02.05.2025).  
   (Практическое руководство по реализации алгоритмов на C#)
4. Официальная документация Microsoft по C#  
   <https://learn.microsoft.com/ru-ru/dotnet/csharp/>(дата обращения 22.05.2025).
5. Кормен Т., Лейзерсон Ч., Ривест Р., Штайн К. Алгоритмы. Построение и анализ. – М.: Вильямс, 2022. – 1328 с.  
   (Классический труд по алгоритмам с математически строгим изложением) (дата обращения 11.05.2025).
6. Седжвик Р., Уэйн К. Алгоритмы на C#. – М.: Вильямс, 2021. – 1056 с.  
   (Практическое применение алгоритмов с примерами на C#).(дата обращения 17.05.2025).