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–№ 17 – ПЗ |

|  |  |  |  |  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- |
|  |  |  | **Студента** | | Полещука Егора Юрьевича | | | | | |
|  |  |  |  | | фамилия, имя, отчество полностью | | | | | |
|  |  |  | Курс | 1 |  | Группа | | ФИТ–242 | | |
|  |  |  |  |  |  | |  |  | |  |
|  | | | **Направление (специальность)** | | | | | ***02.03.02*** | | |
|  | | | Фундаментальная информатика и информационные технологии | | | | | | | |
|  |  |  | 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 2024

**Содержание**

[Введение 3](#_Toc185010271)

[Задача 1. Отбор в разведку. 4](#_Toc185010272)

[Задача 2. Кубик Рубика. 8](#_Toc185010273)

[Задача 3. Паук и муха. 12](#_Toc185010274)

[Задача 4. Формы для отливки. 17](#_Toc185010275)

[Заключение 24](#_Toc185010276)

[Список используемой литературы. 25](#_Toc185010277)

# **Введение**

C# - это объектно-ориентированный язык программирования, который был создан в период с 1998 по 2002 год командой инженеров Microsoft. Язык входит в семью C-подобных языков. Синтаксис приближен к Java и C++. Его особенностями являются статическая типизация, поддержка полиморфизма, перегрузки операторов, а также делегация, атрибуты, события, обобщённые типы и анонимные функции. Язык унаследовал многие свои особенности у Delphi, Smalltalk и Java, при этом исключив многие практики, считающимися проблемными.

C# используется для создания широкого спектра различного программного обеспечения: бизнес-приложений, видеоигр, функциональных веб-приложений, приложений для Windows, macOS, мобильных программ для iOS и Android. Благодаря своей гибкости, язык можно считать практически универсальным.

Основными преимуществами C# перед другими языками программирования являются простота, популярность, понятный синтаксис и низкий порог вхождения. За счёт этого C# позволяет быстро начать разрабатывать полноценные программы, в сжатые сроки создавать функциональные и производительные приложения и легко понимать код, написанный другими людьми.

# **Задача 1. Отбор в разведку.**

Из N солдат, выстроенных в шеренгу, требуется отобрать троих в разведку. Для того чтобы сделать это, выполняется следующая операция: если солдат в шеренге больше 3, то шеренга разбивается на две, одна из которых состоит из солдат, стоящие на четных позициях, а вторая – стоящих на нечетных позициях. Эта процедура повторяется для всех полученных шеренг до тех пор, пока в каждой из них не останется 3 или менее солдат. Если солдат осталось трое, то данную группу можно послать в разведку.

Требуется определить, сколько групп по 3 человека может быть сформировано из исходной шеренги.

**Входной файл** содержит число **N** – количество солдат в исходной шеренге. (0 < **N** ≤ 10000000).

**Выходной файл** должен содержать количество вариантов формирования групп разведки.

**Ход решения:**

Заметим, что при изначальном количестве солдат, равном степени двойки 2n, количество конечных групп по три будет равно нулю. Далее количество групп с увеличением количества солдат будет возрастать, пока не достигнет 2n-1, после чего начнёт опять убывать, пока количество солдат не достигнет 2n+1.

Итак, задача заключается в нахождении ближайшей степени двойки, меньшей входного числа. Затем необходимо найти расстояние от числа до ближайшей степени двойки, которая будет равна либо 2n, либо 2n+1. Если число ближе к меньшей степени двойки, либо стоит ровно между двумя степенями, то мы вычитаем из него степень двойки (N - 2n). Иначе вычитаем из следующей степени двойки изначальное число (2n+1 - N). Полученное число является ответом.

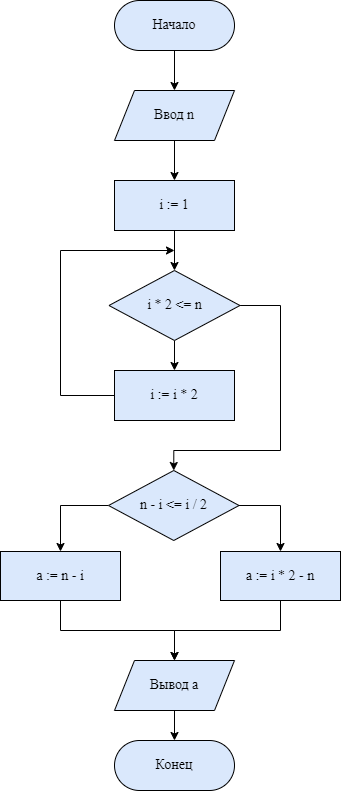


Рисунок 1 – Блок-схема решения задачи 1.

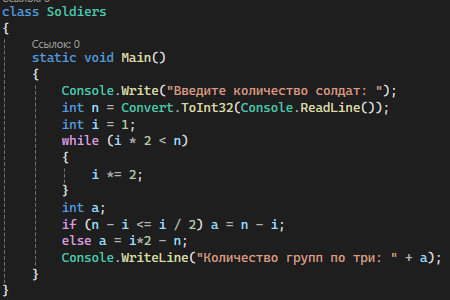


Рисунок 2 – Код для решения задачи 1.



Рисунок 3 – Первый ввод.



Рисунок 4 – Второй ввод.



Рисунок 5 – Третий ввод.

# **Задача 2. Кубик Рубика.**

Имеется кубик Рубика, у которого количество элементов вдоль каждого ребра равно N.

Требуется определить местоположение заданного элемента кубика после M вращений блоков кубика. Каждое вращение выполняется на 90 градусов по часовой стрелке или против часовой стрелки.

**Входной файл**

Первая строка содержит два целых числа **N** и **M**, разделенных пробелом - количество элементов вдоль ребра кубика и количество вращений (1  **N**, **M**  100000).

Вторая строка содержит три целых числа **XN**, **YN** и **ZN**, разделенных пробелами - начальные координаты элемента кубика (1  **XN**, **YN**, **ZN**  **N**).

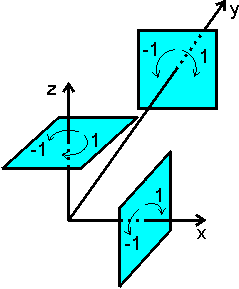


Рисунок 6 – Вращение сторон кубика вокруг осей.

Следующие M строк содержат по данные очередного вращения типа “**A** **K** **S**”, где **A** - ось, вокруг которой проводится вращение (**A**  ['**X**', '**Y**', '**Z**']), K - номер блока кубиков по этой оси, который подвергается вращению (1  **K**  **N**), **S** - направление вращения (-1 - против часовой, 1 - по часовой).

**Выходной файл** должен содержать три целых числа **XK**, **YK** и **ZK**, разделенных пробелами - конечное положение элемента кубика.

**Ход решения:**

Выясним алгоритм изменения координат при повороте по часовой стрелке вокруг оси x:

При повороте вокруг оси x будут изменяться только координаты **Y** и **Z** кубика. При этом новая координаты находятся следующим образом:

**Y`** = **Z**

**Z`** = **N** + **1** – **Y**, где **N** – размер ребра кубика.

Из этого можно вывести общую формулу для поворота по часовой стрелке. Пусть **C1** и **C2** – координаты, изменяющиеся при повороте, тогда **C`1** = **C2**, а **C`2** = **N** + 1 – **C1**.

При повороте против часовой стрелки нужно поменять координаты местами, получаем **C`2**= **C1**, **C`1** = **N** + 1 – **C2**.

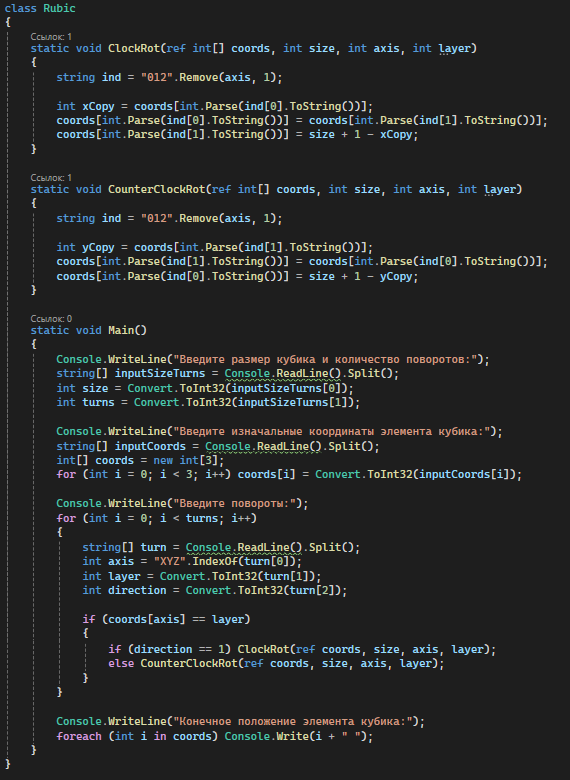


Рисунок 7 – Код решения задачи 2.

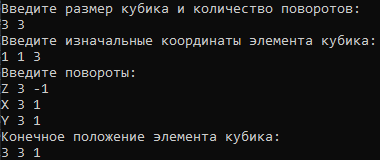


Рисунок 8 – Первый ввод.

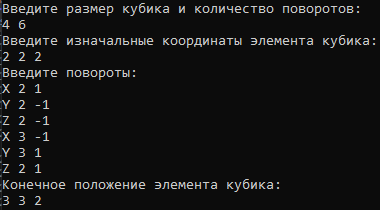


Рисунок 9 – Второй ввод

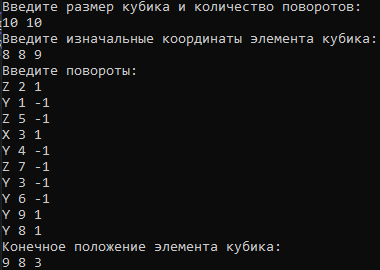


Рисунок 10 – Третий ввод.

# **Задача 3. Паук и муха.**

В комнате, представляющей собой параллелепипед, находятся паук и муха.

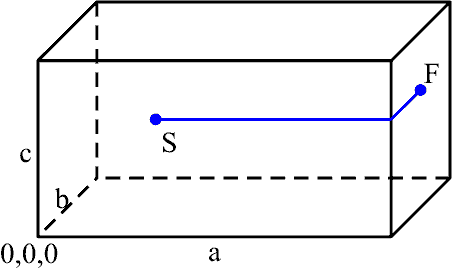


Рисунок 11 – Визуализация комнаты с пауком и мухой.

Паук и муха располагаются на стенах, полу или потолке.

Определите минимальное расстояние, которое необходимо проползти пауку, чтобы добраться до мухи.

**Примечание:** ползать паук может, естественно, только по стенам, полу или потолку.

**Входной файл**

Первая строка содержит три целых числа **a**, **b** и **c**, разделенных пробелами - ширина, глубина и высота комнаты (1  **a, b, c**  100).

Вторая строка содержит три целых числа **Sx**, **Sy**, **Sz**, разделенных пробелами - координаты паука относительно начала комнаты (0, 0).

Третья строка содержит три целых числа **Fx**, **Fy**, **Fz**, разделенных пробелами - координаты мухи относительно начала комнаты (0, 0).

(0  **Sx**, **Fx**  **a**; 0  **Sy**, **Fy**  **b**; 0  **Sz**, **Fz**  **c**), все координаты лежат на стенах, поле или потолке комнаты.

**Выходной файл**

Первая строка должна содержать минимальное расстояние, которое необходимо проползти пауку, чтобы добраться до мухи - число с тремя знаками после запятой.

**Ход решения:**

Существует 3 положения паука и мухи относительно друг друга в комнате:

1. Паук и муха находятся на одной стороне комнаты. В данном случае расстояние между ними будет равно

, где **A** и **B** – координаты, от которых зависит положение паука и мухи на стороне комнаты соответственно.

1. Паук и муха находятся на смежных сторонах комнаты. Для того, чтобы найти расстояние между ними, проведём перпендикуляры к общей грани. Сумма длин этих перпендикуляров станет высотой треугольника, а разность координат на общей оси координат станет шириной треугольника. Тогда расстояние равно

, где **h1** и **h2** – длины перпендикуляров, а **a1** и **a2** – координаты паука и мухи по общей оси соответственно.

1. Паук и муха находится на противоположных сторонах комнаты. Сначала отложим перпендикуляры от паука и мухи ко всем граням. Сумма длин перпендикуляров и длины стороны между пауком и мухой будет высотой треугольника. Шириной треугольника станет разность между координатами по оси, перпендикулярной высоте получившегося треугольника. Тогда расстояние между пауком и мухой выглядит как

, где **h1** и **h2** – длины перпендикуляров, **c** – длина стороны, лежащей между пауком и мухой, а **a1** и **a2** – координаты паука и мухи по оси, перпендикулярной высоте.

Из 4 случаев выбираем наименьшее расстояние.

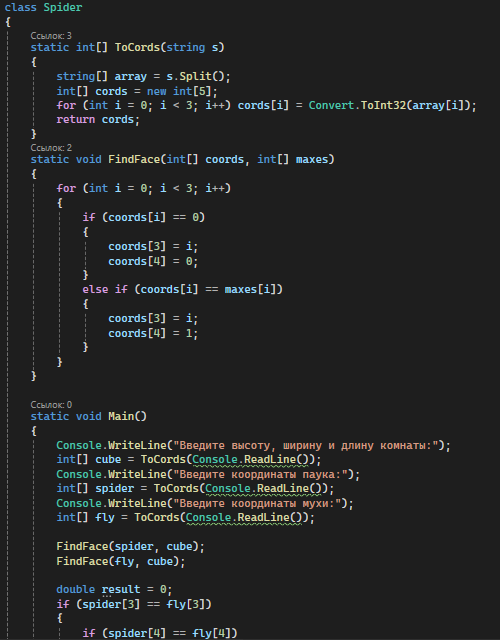


Рисунок 12 – Код для решения задачи 3 (1)

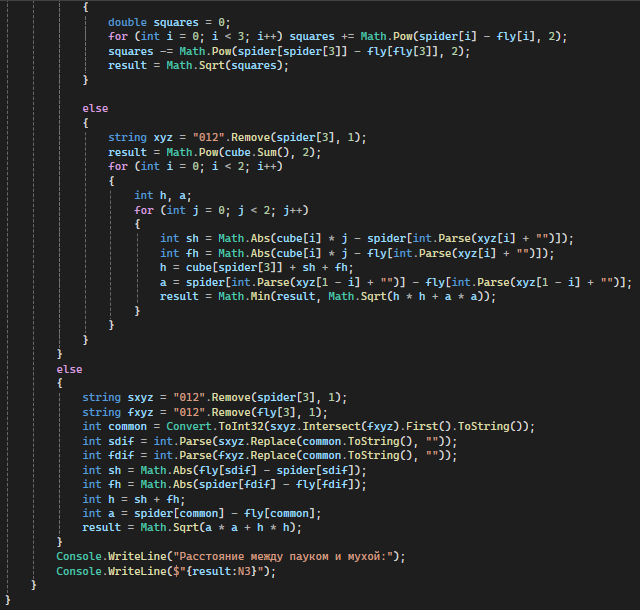


Рисунок 13 – Код для решения задачи 3 (2)

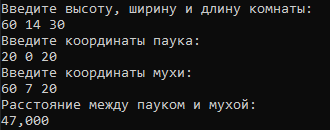


Рисунок 14 – Первый ввод.

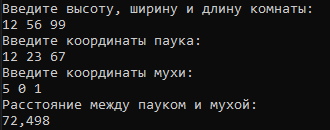


Рисунок 15 – Второй ввод.

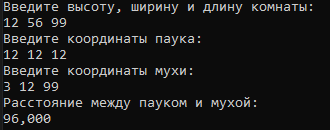


Рисунок 16 – Третий ввод.

# **Задача 4. Формы для отливки.**

Завод “Coding” изготовляет детали для устройства кодирования радиопередач. Детали изготавливаются методом отливки в форму.

Деталь представляет собой цилиндр с выступами в четырех направлениях, расположенных относительно друг друга под углом 90 градусов. В каждом направлении может быть от 0 до 5 выступов. Возможные положения выступов фиксированы.

Форма состоит из двух частей, причем линия соединения частей “режет” пространство для детали по середине двух групп выступов.

При проведении ремонтных работ были перепутаны половинки форм. Требуется по заданным деталям собрать половинки форм.

**Примечания:**

Детали и половинки форм можно вращать;

все детали уникальны, в том числе с учетом вращения;

все половинки форм уникальны, в том числе с учетом вращения;

гарантируется, что имеется единственное решение.

**Входной файл**

Первая строка содержит целое число **N** - количество деталей (1  **N**  1000).

Следующие **N** строк содержат по 20 целых чисел - расположения выступов на детали (4 группы по 5 чисел; 0 - выступа нет, 1 - выступ есть. Группы заданы в условном порядке: верхняя, ближняя, нижняя, дальняя).

Следующие 2\***N** строк содержат по 15 целых чисел - расположение выступов на половинках форм (3 группы по 5 чисел; 0 - выступа нет, 1 - выступ есть. Группы заданы в условном порядке: верхняя, вдавленная, нижняя). Плоскость разреза обращена к наблюдателю.

**Выходной файл**

Должен состоять из **N** строк, каждая из которых содержит по два целых числа, разделенных пробелом - номера половинок для формы **i**-ой детали, причем в каждой строке сначала указывается половинка с наименьшим номером, а потом с наибольшим.

**Ход решения:**

Подберём подходящие половинки перебором. Алгоритм будет заключаться в выборе детали из всех деталей и сравнении с ним всех половинок. Если какая-то половинка подходит, то мы останавливаем поиск (в связи с уникальностью половинок и деталей) и сохраняем её в отдельный массив. Поворачиваем деталь на 90 градусов и повторяем, проверяя каждую деталь. Таким образом мы находим от двух до четырёх подходящих половинок. Из этого вытекает 3 случая:

1. Если подошли лишь 2 половинки, то, очевидно, они обе подходят.
2. Если подошли 3 половинки, то сравниваем поочерёдно подходят ли они друг другу, то есть сопоставляем первую со второй, первую с третьей. Если обе этих проверки оказались неверными, то подходит вторая и третья половинки.
3. Если подошли 4 половинки, то они просто записываются в отдельный массив для сравнения с окончательным массивом всех половинок. Те половинки, которых нет в окончательном массиве, подходят.

После повтора этого алгоритма для всех деталей задача решена.

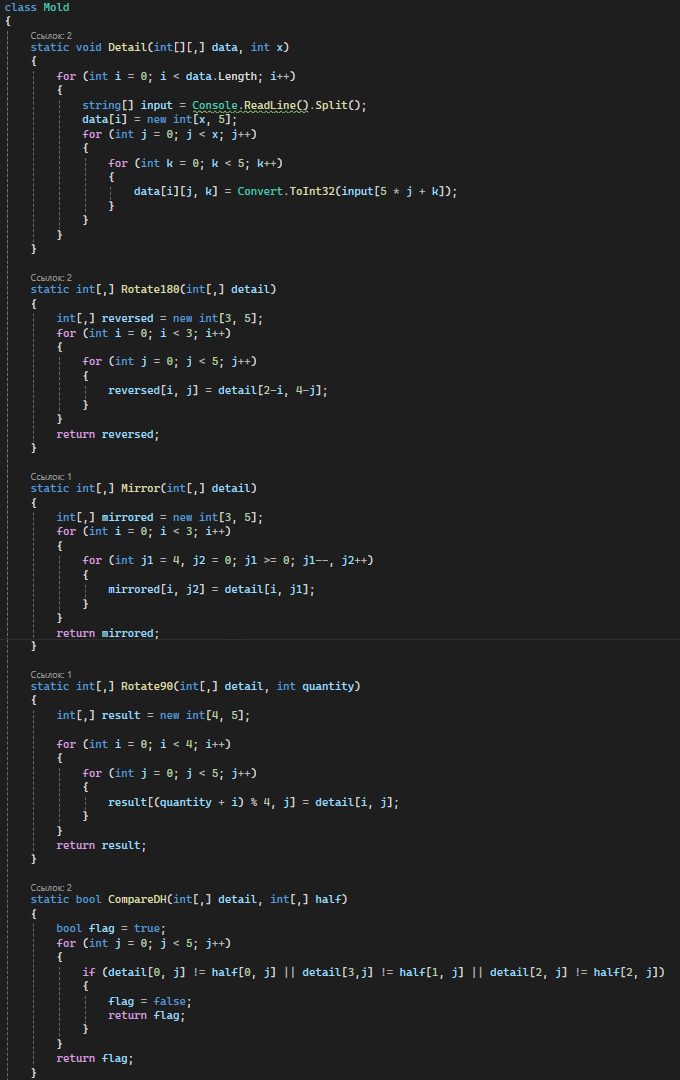


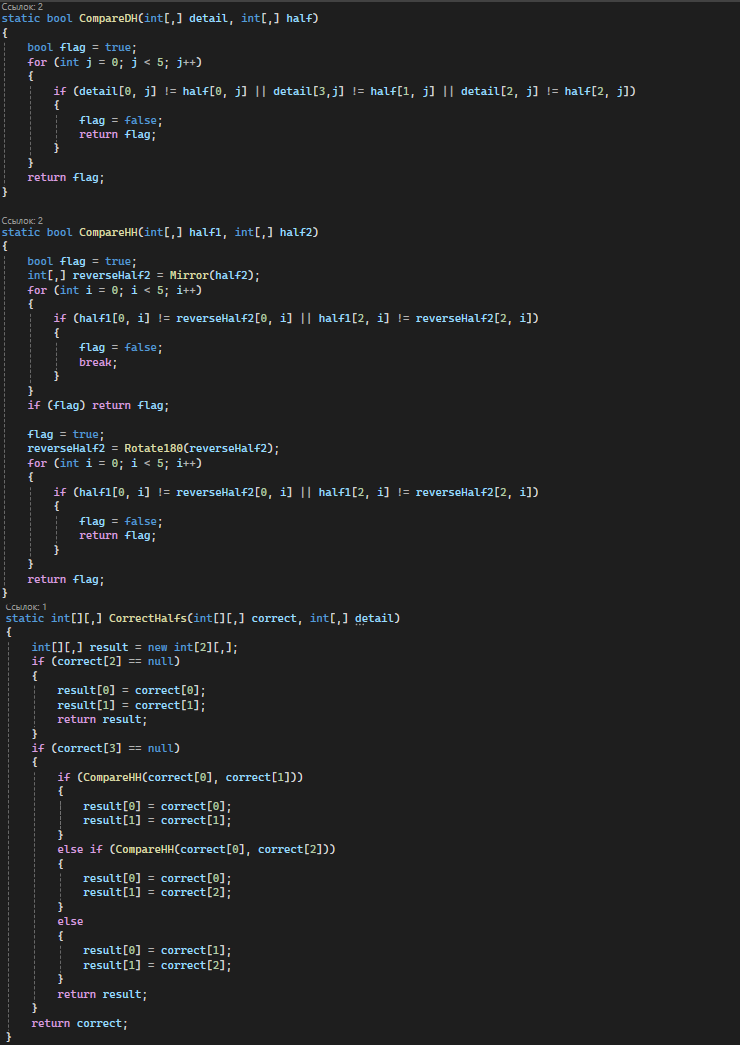
Рисунок 17 – Код для решения задачи 4 (1).

Рисунок 18 – Код решения задачи 4 (2).

Рисунок 19 – Код решения задачи 4 (3).

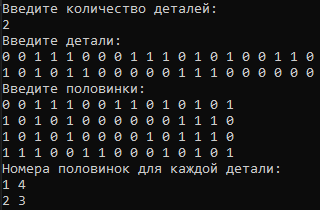


Рисунок 20 – Первый ввод.

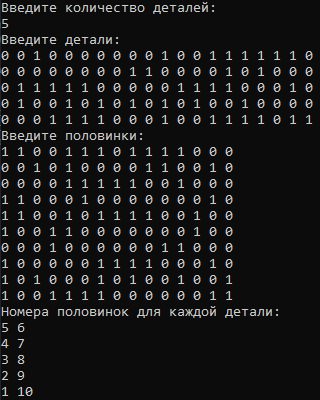


Рисунок 21 – Второй ввод.

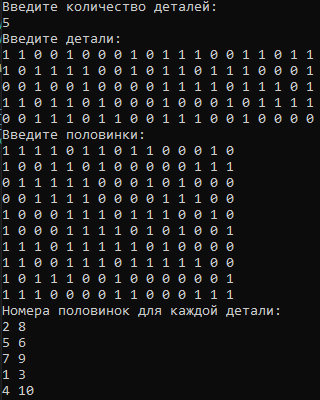


Рисунок 22 – Третий ввод.

# **Заключение**

В ходе выполнения работы были использованы основные конструкции языка C#, включая типы данных int, double и string, условные операторы if, else, switch, циклы for, while, зубчатые и двумерные массивы, функции, библиотека Math. Особое внимание уделялось разработке алгоритмов, обеспечивающих корректность и эффективность решения поставленных задач.

Во время решения задач также были изучены основные аспекты C#, такие как особенности массивов, отличия ступенчатых массивов от многомерных, работа со строками, методы строк, функции с выводом данных и без вывода, различия между циклами for и while, условными операторами if и switch.

# **Список используемой литературы.**

1. Документация Microsoft C# <https://learn.microsoft.com/ru-ru/dotnet/csharp/> дата обращения 24.11.2024.
2. Васильев А.Н. Программирование на C# для начинающих. Основные сведения. – Эксмо, Москва, 2018. – 592 с.
3. Stack Overflow <https://stackoverflow.com> дата обращения 01.12.2024
4. Стиллмен Э., Грин Дж. Head First. Изучаем C#. – Питер, Санкт-Петербург, 2022. – 768 с.
5. Мюллед Дж.П., Семпф Б., Сфер Ч. C# для чайников. – Диалектика, Санкт-Петербург, 2019. – 608 с.