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–№2– ПЗ |

|  |  |  |  |  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- |
|  |  |  | **Студента** | | Белого Владимира Александровича | | | | | |
|  |  |  |  | | 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

**Оглавление**

[Введение 3](#_Toc187600071)

[Задача 1. Отбор в разведку. 4](#_Toc187600072)

[Ход решения. Отбор в разведку 5](#_Toc187600073)

[Задача 2. Кубик Рубика. 8](#_Toc187600074)

[Ход решения. Кубик Рубика. 9](#_Toc187600075)

[Задача 3. Шестеренки. 12](#_Toc187600076)

[Ход решения. Шестеренки 14](#_Toc187600077)

[Задача 4. Интенсификация производства. 17](#_Toc187600078)

[Ход решения. Интенсификация производства. 18](#_Toc187600079)

[Заключение 19](#_Toc187600080)

[Список используемой литературы. 20](#_Toc187600081)

# **Введение**

C# – это объектно-ориентированный язык программирования, который был разработан командой инженеров Microsoft в период с 1998 по 2002 год.

C# относится к семейству C-подобных языков. Его синтаксис похож на Java и C++.

C# поддерживает статическую типизацию, полиморфизм, перегрузку операторов, а также такие концепции, как делегаты, атрибуты, события, обобщённые типы и анонимные функции.

C# унаследовал многие особенности от Delphi, Smalltalk и Java, но при этом исключил некоторые практики, которые считаются проблематичными.

C# используется для создания различных программ, включая бизнес-приложения, видеоигры, функциональные веб-приложения, приложения для Windows, macOS, мобильных устройств на iOS и Android. Благодаря своей гибкости, C# можно считать универсальным языком программирования.

Преимущества C# перед другими языками программирования включают простоту, популярность, понятный синтаксис и низкий порог вхождения. Это позволяет быстро разрабатывать полноценные программы, создавать функциональные и производительные приложения, а также легко понимать код, написанный другими разработчиками.

# **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. Отбор в разведку**

Обратите внимание, что если изначально у нас есть количество солдат, которое можно представить в виде степени двойки 2n, то количество групп по три человека будет равно нулю.

Затем, с увеличением количества солдат, количество групп будет расти, пока не достигнет значения 2n - 1. После этого оно начнёт уменьшаться, пока количество солдат не достигнет значения 2n + 1.

Итак, задача состоит в том, чтобы найти ближайшую степень двойки, которая меньше входного числа. Затем необходимо определить расстояние от числа до этой степени двойки.

Если число находится ближе к меньшей степени двойки или находится ровно между двумя степенями, то мы вычитаем из него степень двойки (N - 2n). В противном случае мы вычитаем из следующей степени двойки исходное число (2n + 1 - N). Полученное число и будет ответом.

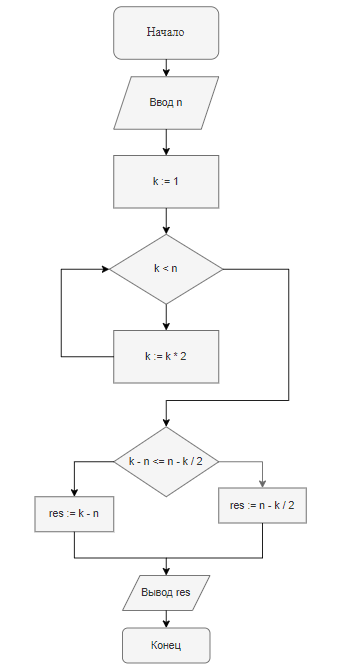


Рисунок 1 – Блок-схема решения задачи 1.

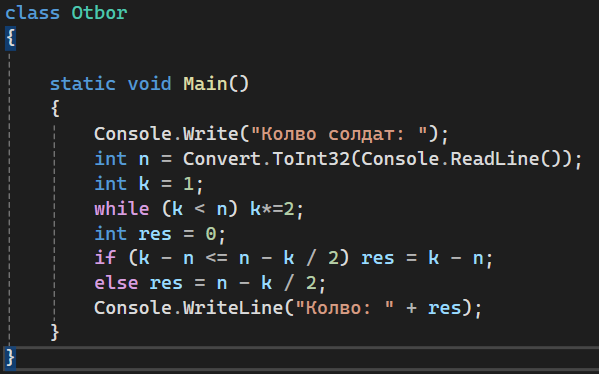


Рисунок 2 – Код для решения задачи 1.



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**).

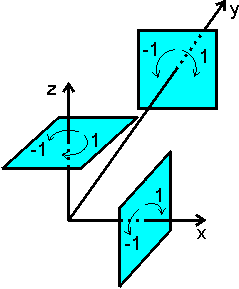


Рисунок 6 – Вращение сторон кубика вокруг осей.

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. Кубик Рубика.**

Определим последовательность действий для изменения координат при повороте вокруг оси x по часовой стрелке:

При повороте вокруг оси x изменяются только координаты Y и Z кубика. Новые координаты вычисляются по следующим формулам:

Y = Z

Z = N + 1 - Y,

где N – длина ребра кубика.

На основе этих формул можно вывести общую формулу для поворота по часовой стрелке. Пусть C1 и C2 – координаты, которые изменяются при повороте. Тогда C1 = C2, а C2 = N + 1 - C1.

При повороте против часовой стрелки координаты меняются местами: C2 = C1, C1 = N + 1 - C2.

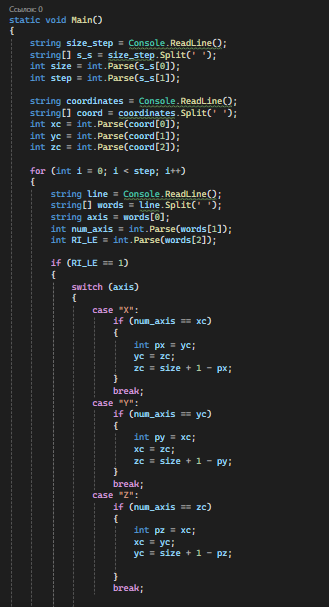


Рисунок 7 – Код решения задачи 2(1).

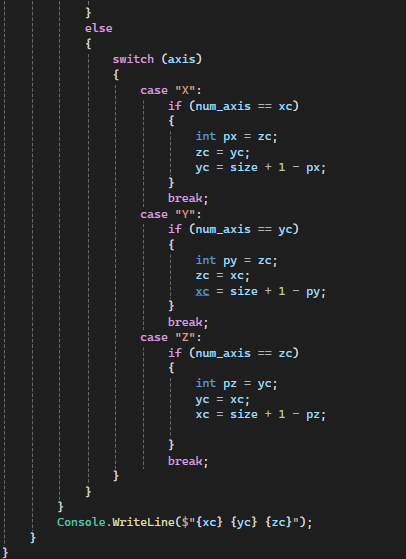


Рисунок 8 – Код решения задачи 2 (2).

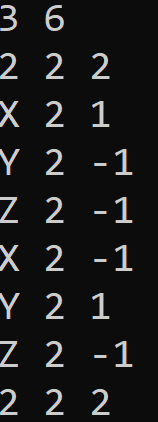


Рисунок 9 – Первый ввод.

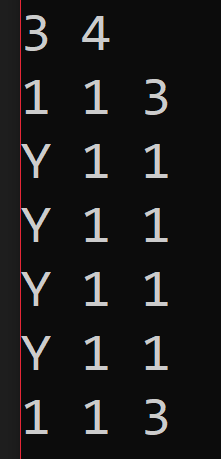


Рисунок 10 – Второй ввод

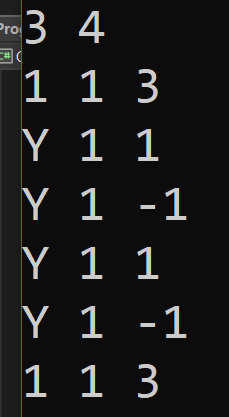


Рисунок 11 – Третий ввод

# **Задача 3. Шестеренки.**

Редуктор привода канатной дороги состоит из **N** шестеренок, каждая из которых имеет определенное количество зубцов. Шестеренки соединены между собой в схему произвольной структуры. Одна из шестеренок надета на вал двигателя; другая – на вал кольца канатной дороги.

Требуется для заданной схемы определить возможность работы такого редуктора и, в случае возможной работы, параметры вращения шестеренки на вале кольца канатной дороги.

**Примечание**

 возможность работы определяется только согласованностью направлений и угловых скоростей вращения шестеренок. Геометрическую возможность их размещения, усилия на проворачивание и т.д. не учитывать;

 каждая шестеренка имеет хотя бы одно соединение с другими шестеренками;

 скорость вращения шестеренок определять без погрешности.

**Входной файл**

Первая строка содержит два целых числа **N** и **M** – количество шестеренок и количество описаний соединения шестеренок (2 ≤ **N** ≤ 100; **N**-1 ≤ **M** ≤ **N**\*(**N**-1)/2).

Следующие **N** строк содержат по два целых числа **K** и **Rk** – номер шестеренки и количество зубьев у нее (1  **K**  **N**; 10  **Rk**  1000).

Следующие **M** строк содержат по два целых числа **S1** и **S2** - номера двух соединенных шестеренок (1  **S1**, **S2**  **N**).

Предпоследняя строка содержит два целых числа **Z1** и **Z2** – номера шестеренок, надетых на валы двигателя и кольца канатной дороги, соответственно (1  **Z1**, **Z2**  **N**).

Последняя строка содержит целое число **V** – направление вращения шестеренки, надетой на вал двигателя. Значение -1 соответствует вращению по часовой стрелки, а 1 – против часовой.

**Выходной файл**

Первая строка должна содержать одно целое число – признак возможности работы редуктора. Значение 1 соответствует возможности работы; значение -1 соответствует невозможности работы.

Если работа редуктора возможна, то

 вторая строка должна содержать целое число **W** – направление вращения шестеренки, надетой на вал кольца канатной дороги. Значение -1 соответствует вращению по часовой стрелки, а 1 – против часовой;

 третья строка должна содержать вещественное число **O** –количество оборотов, которое сделает шестеренка, надетая на вал кольца канатной дороги, если шестеренка, надетая на вал двигателя сделает 1 полный оборот. Число выводится с тремя знаками после запятой.

# **Ход решения. Шестеренки**

**Принимаются и распределяются полученные данные по массивам.**

1.Принимаются и распределяются полученные данные по массивам.

2.Создаётся массив с информацией о взаимодействиях.

3.Даётся стартовой шестерёнке «заряд», равный 1 или -1 в зависимости от стороны поворота.

4.Создаётся цикл, внутри которого каждый раз каждая шестерёнка, у которой есть «заряд», передаёт его всем шестерёнкам, которые с ней контактируют. Причём если у изначальной заряд 1, то у второй -1 и наоборот.

5.Если у шестерёнки уже есть заряд, проверяется случай 1 к 1 и -1 к -1. Если таковой наблюдается, то вращение невозможно.

6.Скорость вращения равна величине стартовой, делённой на величину последней.

7.Выводится возможность вращения, скорость и направление.

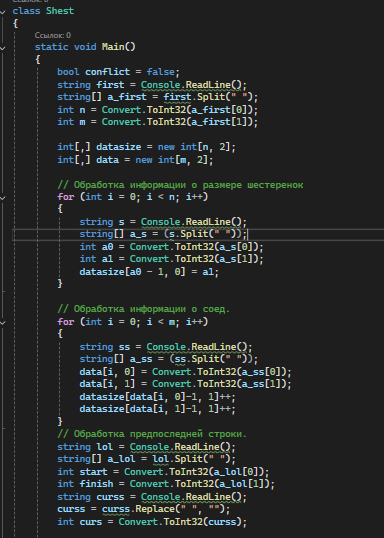


Рисунок 12 – Код для решения задачи 3 (1)

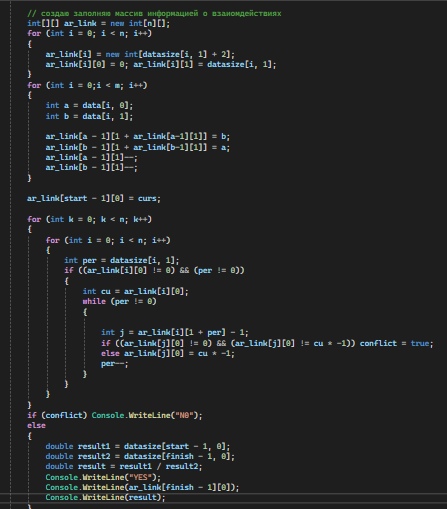


Рисунок 13 – Код для решения задачи 3 (2)

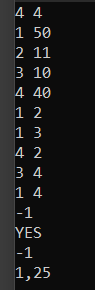


Рисунок 14 – Первый ввод.

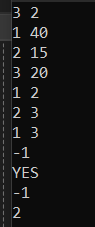


Рисунок 15 – Второй ввод.

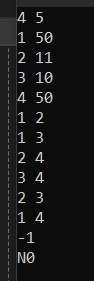


Рисунок 16 – Третий ввод

# **Задача 4. Интенсификация производства.**

Перед коллективом предприятия “Ни шагу назад” была поставлена задача наращивать каждый день производство продукции на 1.

Требуется определить, какой суммарный объем продукции будет выпущен предприятием за заданный период, если в первый день периода предприятие выпускало **P** единиц продукции.

**Примечания:**

1. период задается в виде двух календарных дат;
2. длительность периода лежит в диапазоне от 1 до 60000;
3. високосные годы учитываются по упрощенному правилу: високосным считается год, делящийся нацело на 4;
4. день начала периода и день его окончания учитываются при подсчете суммарного объема продукции и длительности периода;
5. все даты заданы корректно.

**Входной файл** содержит:

1. в первой строке – дата начала периода в формате ДД.ММ.ГГГГ;
2. во второй строке – дата окончания периода в формате ДД.ММ.ГГГГ;
3. в третьей строке целое число – начальный выпуск продукции **P** (0  **P**  5000).

**Выходной файл** должен содержать суммарный объем продукции.

# **Ход решения. Интенсификация производства.**

Пользуясь классом DateTime, вычисляем промежуток времени между двумя датами с помощью класса TimeSpan. Затем создаём переменную типа лонг, которой задаём значения поля days промежутка времени**.**

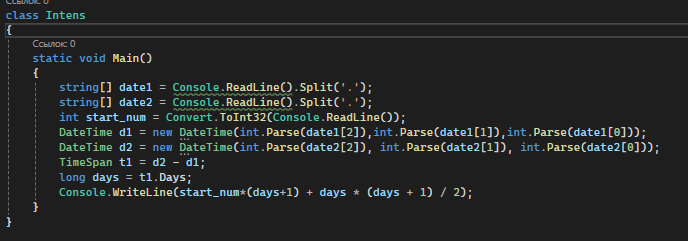


Рисунок 17 – Код для решения задачи 4

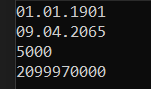


Рисунок 18 – Первый ввод.

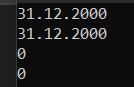


Рисунок 19 – Второй ввод.

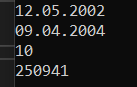


Рисунок 20 – Третий ввод.

# **Заключение**

В процессе работы были применены ключевые конструкции языка C#, включая типы данных int, double и string, а также условные операторы if, else и switch. Кроме того, были использованы циклы for и while, а также зубчатые и двумерные массивы.

Особое внимание уделялось разработке алгоритмов, которые обеспечивали корректность и эффективность решения поставленных задач.

В ходе работы были изучены основные аспекты C#, такие как особенности массивов, различия между ступенчатыми и многомерными массивами, работа со строками, методы строк, функции с выводом данных и без вывода, а также 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. Мюллед Дж.П., Семпф Б., Сфер Ч. C# для чайников. – Диалектика, Санкт-Петербург, 2019. – 608 с.
3. Фленов, Михаил Библия C# **–** БХВ-Петербург, Санкт-Петербург, 2021 **–** 560 c.
4. Рихтер, Джеффри CLR via C#. Программирование на платформе Microsoft.NET Framework 4.5 на языке C# **–** Питер, Санкт-Петербург, 2018. **–** 896 c.
5. Сафонов, В. О. Параметризованные типы данных. История, теория, реализация и применение **–** Издательство Санкт-Петербургского университета, Санкт-Петербург, 2020. **–** 116 c.