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–02.03.03–№ 1 – ПЗ |

|  |  |  |  |  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- |
|  |  |  | **Студента** | | Аристовой Ирины Витальевны | | | | | |
|  |  |  |  | | 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 2024

**Содержание**

[Введение 3](#_Toc184552134)

[Постановка задачи «Упаковки молока» 4](#_Toc184552135)

[Ход решения задачи «Упаковки молока» 6](#_Toc184552136)

[Постановка задачи «Отгадай число» 10](#_Toc184552137)

[Ход решения задачи «Отгадай число» 12](#_Toc184552138)

[Постановка задачи «Интенсификация производства» 15](#_Toc184552139)

[Ход решения задачи «Интенсификация производства» 16](#_Toc184552140)

[Постановка задачи «Зельеварение» 18](#_Toc184552141)

[Ход решения задачи «Зельеварение» 21](#_Toc184552142)

[Заключение 24](#_Toc184552143)

[Список используемой литературы 25](#_Toc184552144)

Введение

Язык программирования C# был разработан компанией Microsoft и стал одним из самых популярных языков для создания приложений на платформе .NET. Благодаря своей простоте и мощным возможностям C# позволяет разработчикам эффективно создавать разнообразные приложения — от настольных до веб- и мобильных.

В данной расчётно-графической работе мы рассмотрим четыре задачи, решаемые с использованием языка C#. Эти задачи охватывают различные аспекты, включая работу с базовыми структурами данных, алгоритмами.

Для разработки использовалась среда разработки Visual Studio Code, которая предоставляет инструменты для написания, отладки и тестирования кода на C#. Она облегчает процесс разработки благодаря своим возможностям автозаполнения, встроенной документации.

Решение этих задач не только продемонстрирует практическое применение C#, но и углубит понимание его возможностей и особенностей.

Постановка задачи «Упаковки молока»

На оптовой базе имеется молоко, выпущенное несколькими фирмами.

Молоко каждой фирмы расфасовано в два вида упаковок, представляющих собой параллелепипеды. Для каждого вида упаковки каждой из фирм известна стоимость, которая включает как стоимость материала тары, так и стоимость собственно молока.

Требуется определить фирму, у которой стоимость одного литра собственно молока минимальна, а также эту стоимость.

Примечание

Считать, что материал тары абсолютно тонкий и все плоскости параллелепипеда состоят из одного слоя материала.

Считать, что у двух упаковок одной фирмы стоимость единицы площади материала одинакова.

Считать, что у двух упаковок одной фирмы стоимость одного литра собственно молока одинакова.

Входной файл

Первая строка содержит целое число N - количество фирм (1  N  100).

Следующие N строк содержат шесть целых чисел Xi1, Yi1, Zi1, Xi2, Yi2, Zi2 - размеры двух видов упаковок i-ой фирмы в сантиметрах (0 < Xi1, Yi1, Zi1, Xi2, Yi2, Zi2  100; 1  i  N), а также два вещественных числа Ci1 и Ci2 - стоимости первой и второй упаковок соответственно у i-ой фирмы в рублях (0 < Ci1, Ci2  1000.0). В стоимости упаковок включаются как стоимость материала тары, так и стоимость собственно молока.

Выходной файл

Должен содержать одну строку, состоящую из целого и вещественного чисел, разделенных пробелом - номер фирмы, у которой стоимость одного литра собственно молока минимальна, а также эту стоимость в рублях (стоимость выводить с двумя знаками после запятой).

Если имеется несколько фирм с одинаковой минимальной стоимостью собственно молока, то вывести ту из них, номер которой минимален.

Пример:

Input.txt

2

10 10 5 10 10 10 12.23 20.12

5 15 20 7 8 9 43.28 16.99

Output.txt

2 4.17

Ход решения задачи «Упаковки молока»

Нахождение номера фирмы с минимальной стоимостью

Вывод номера фирмы и минимальной стоимости

Вычисление площади поверхности 1-ой упаковки

Вычисление объема 1-ой упаковки

Ввод информации о 2-ой упаковке молока

Вычисление площади поверхности 1-ой упаковки

Вычисление объема 1-ой упаковки

Вычисление стоимости за 1 литр молока

Вычисление минимальной стоимости молока

Ввод информации о 1-ой упаковке молока

Ввод числа фирм (n)

Цикл от i до n

Рисунок 1 – Блок-схема алгоритма

Чтобы определить минимальную цену, следует узнать стоимость 1 литра молока каждой фирмы. Для этого необходимо решить систему уравнений.

Пусть g – стоимость за литр, m – стоимость материала, c1 - стоимость 1-ой упаковки, c2 - стоимость 2-ой упаковки, S1 – площадь поверхности 1-ой упаковки, S2 – площадь поверхности 2-ой упаковки, V1 – объём 1-ой упаковки, V2 – объём 2-ой упаковки.

Тогда, решая систему , получим .

После математической обработки задачи следует продумать программную реализацию. Пользователь вводит число фирм, среди которых будем искать минимальную стоимость молока. Создаем массив для стоимостей. Создаем цикл, в котором вводятся данные о двух пакетах молока одной и той же фирмы, но разных размеров и стоимостей. Используя формулу для определения стоимости, находим её и кладем в массив. Ищем в массиве минимальную стоимость 1 литра молока и к индексу этой стоимости прибавляем 1, чем находим первый номер фирмы с минимальной стоимостью молока.

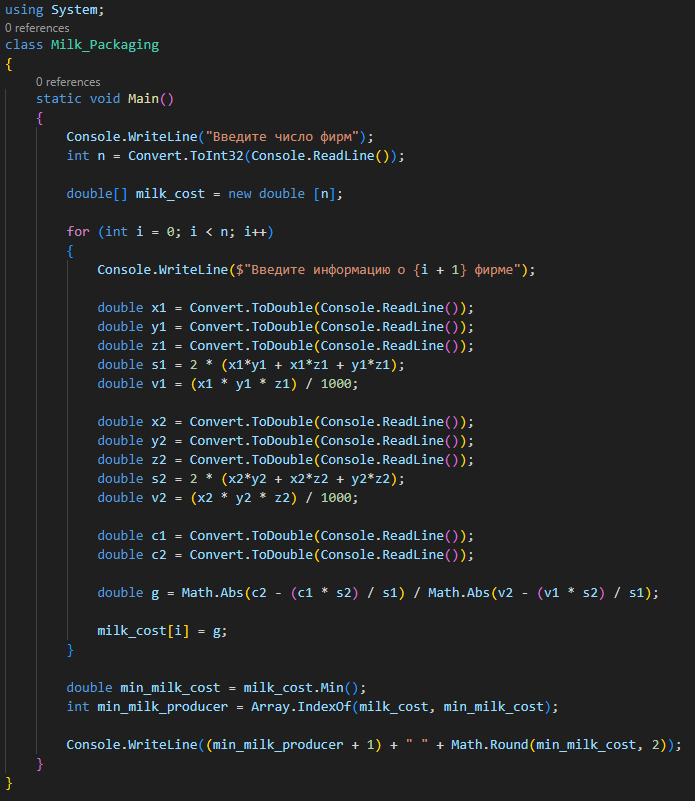
Код программы представлен ниже.

Рисунок 2 – Код задачи «Упаковки молока»

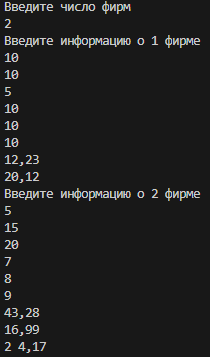
Примеры работы тестовых данных представлены ниже.

Рисунок 3 – Пример работы тестовых данных 1

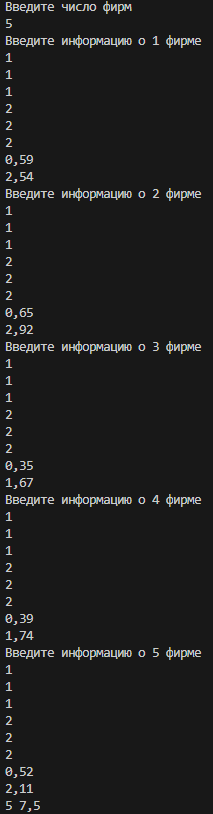


Рисунок 4 – Пример работы тестовых данных 2

Постановка задачи «Отгадай число»

Известен следующий фокус. Фокусник предлагает выполнить действия следующего характера: задумайте число, прибавьте 2, умножьте на 3, отнимите 5, отнимите задуманное число и т.д. После этого по названному полученному результату фокусник определяет задуманное число.

Пусть задумано некоторое целое число X. Требуется после выполнения ряда действий по известному результату R определить это число.

Примечание:

• гарантируется, что имеется только один ответ;

• гарантируется, что во время выполнения действий какие-либо промежуточные результаты не превышают по модулю 2 000 000 000.

Входной файл

Первая строка содержит количество действий N (0 ≤ N ≤ 100).

Следующие N строк содержат описания действий в последовательности их выполнения, причем в каждой строке указывается одно действие в формате S V, где:

• S - тип действия, состоящий из одного символа: "\*" - умножить; "-" - отнять; "+" - прибавить;

• V - аргумент действия. Может быть целым числом (|V| ≤ 100) либо символом "x". Символ "x" может применяться только в действиях "-" и "+" и обозначает, что нужно отнять или прибавить задуманное число, соответственно.

Последняя строка содержит результат R (|R| ≤ 2 000 000 000).

Выходной файл должен содержать одно целое число - задуманное число X.

Пример:

Input.txt

4

+ 2

\* 3

- 5

- x

7

Output.txt

3

Ход решения задачи «Отгадай число»

Число, которое нужно отгадать можно представить в виде линейного уравнения: , где a – коэффициент перед задуманным числом, x – задуманное число, b – свободные переменные, result – конечный результат проведения операций.

Пользователь вводит число операций. Создаем массив equation = [1, 0], который представляет собой a = 1 (так как число точно загадано), b = 0 (так как еще не проведены никакие операции).

Далее идет цикл, в котором обрабатывается каждая операция. Пользователь вводит операцию и число. Эта строка разбивается по пробелу и кладется в новый массив operation. Если на этом этапе получаем массив operation = [“+”, “x”], то equation[0] + 1, если operation = [“-”, “x”], то equation[0] – 1. Если же имеем дело с операциями сложения, умножения, вычитания с числами, то действуем по-другому. В зависимости от введенной операции происходит аналогично описанной выше обработка с помощью switch и его вызова.

После обработки всех операций пользователь вводит результат. Вычисляем загаданное число как .

Код программы представлен ниже.



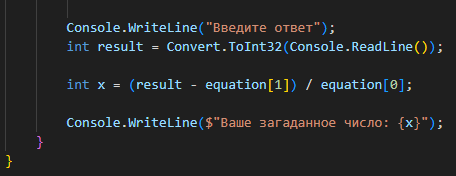
Рисунок 5 - Код задачи «Отгадай число»

Рисунок 6 – Код задачи «Отгадай число» (продолжение)

Примеры работы тестовых данных представлены ниже.

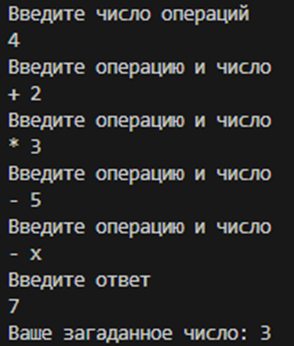


Рисунок 7 - Пример работы тестовых данных 1

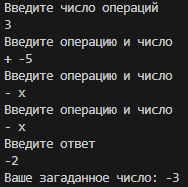


Рисунок 8 - Пример работы тестовых данных 2

Постановка задачи «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:

• период задается в виде двух календарных дат;

• длительность периода лежит в диапазоне от 1 до 60000;

• високосные годы учитываются по упрощенному правилу: високосным считается год, делящийся нацело на 4;

• день начала периода и день его окончания учитываются при подсчете суммарного объема продукции и длительности периода;

• все даты заданы корректно.

Входной файл содержит:

• в первой строке – дата начала периода в формате ДД.ММ.ГГГГ;

• во второй строке – дата окончания периода в формате ДД.ММ.ГГГГ;

• в третьей строке целое число – начальный выпуск продукции P (0 ≤ P ≤ 5000).

Выходной файл должен содержать суммарный объем продукции.

Пример:

Input.txt

12.05.2002

15.05.2002

10

Output.txt

46

Ход решения задачи «Интенсификация производства»

Пользователь вводит начальную дату, которая «разбивается» по знаку “.” и кладется в соответствующий массив. Далее каждый элемент этого массива через цикл переводится в int. Аналогично поступаем с конечной датой.

Используя структуру DateTime и метод TimeSpan вычисляем разницу между датами.

Пользователь вводит начальный размер выпуска продукции. С помощью цикла, прибавляя за каждый день по +1, вычисляем конечный выпуск продукции и выводим ответ.

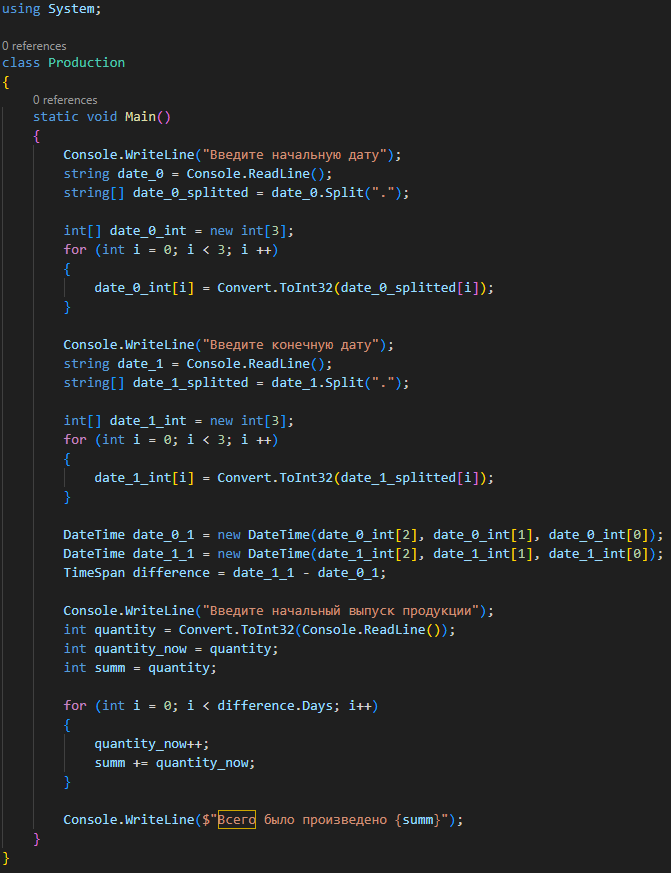
Код программы представлен ниже.

Рисунок 9 – Код задачи «Интенсификация производства»

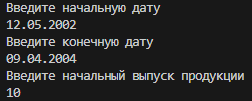
Примеры работы тестовых данных представлены ниже.

Рисунок 10 – Пример работы тестовых данных 1

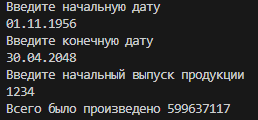


Рисунок 11 - Пример работы тестовых данных 2

Постановка задачи «Зельеварение»

Одним из нелюбимых предметов Невилла Долгопупса, товарища Гарри Поттера с факультета Гриффиндор школы чародейства и волшебства “Хогвартс”, было зельеварение.

Чтобы помочь Невиллу в совершенствовании навыков зельеварения, Гарри придумал зельеварочный комбайн, изготавливающий зелья с помощью определенных заклинаний. Каждое заклинание представляет собой одно слово, формируемое в зависимости от последовательности и методов приготовления зелья.

Последовательность приготовления зелья описывается в виде набора действий, каждое из которых указывает на метод обработки заданного списка ингредиентов:

• смешивание описывается MIX < ингредиент1 ингредиент2, …>;

• растворение в воде описывается WATER < ингредиент1 ингредиент2, …>;

• измельчение описывается DUST < ингредиент1 ингредиент2, …>;

• обжиг описывается FIRE < ингредиент1 ингредиент2, …>,

при этом в качестве любого ингредиента может выступать либо некоторое вещество, задаваемое строковой константой, либо результат выполнения любого из предыдущих действий, задаваемый с помощью номера действия. Название действия и названия ингредиентов разделяются пробелами. В действии участвует как минимум один ингредиент.

Каждое действие переводится в слово по следующему правилу:

• смешивание задается в формируемом заклинании словом MX<список ингредиентов>XM;

• растворение в воде - слово WT<список ингредиентов>TW;

• измельчение - слово DT<список ингредиентов>TD;

• обжиг - слово FR<список ингредиентов>RF,

где <список ингредиентов> - единое слово, сформированное путем сложения названий ингредиентов или слов, описывающих предыдущие действия.

Последнее действие явно или неявно использует результаты выполнения всех предыдущих действий и является основой для заклинания.

Помогите Невиллу по заданной последовательности действий сформировать заклинание.

Примечание:

• при формировании заклинания учитывается регистр названий действий и ингредиентов;

• порядок названий ингредиентов в заклинании должен соответствовать их порядку в действии, т.е. для действия “DUST root tooth” в заклинании правильным считается слово “DTroottoothTD”, а слово “DTtoothrootTD” считается неправильным;

• в названии веществ нет цифр, а используются только английские буквы;

• каждое действие может быть несколько раз использовано в последующих действиях;

• гарантируется, что длина строки, содержащей сформированное заклинание, не превышает 50000 символов.

Входной файл содержит набор строк, каждая из которых описывает отдельное действие. Строки расположены в порядке выполнения действий. Длина каждой строки не превышает 255 символов. Количество строк не превышает 100.

Выходной файл должен содержать строку, содержащую сформированное заклинание.

Пример:

Input.txt

DUST root tooth

WATER 1 tear

Output.txt

WTDTroottoothTDtearTW

Ход решения задачи «Зельеварение»

Из входных данных довольно легко понять, как формируется конечное заклинание. В начале и конце ставится “DT” и “TD”, если “DUST”; “WT” и “TW”, если “WATER”; “FR” и “RF”, если “FIRE”; “MX” и “XM”, если “MIX”. Между этими буквами идет все то, что было после описания действий во входных данных без пробела. Стоит учитывать, если внутри попадается цифра, то она показывается число повторов ингредиента, но сама не пишется в заклинании.

Реализация данной задачи так же будет выполнена через switch. Пользователь вводит число заклинаний и заклинания, которые «разбиваются» по пробелу и кладутся в массив spell\_now\_splitted. Первый элемент массива представляет собой действие (“DUST”, “WATER”, “FIRE”, “MIX”), которое далее будет использовано в switch.

Определим, что будет внутри данного заклинания. В цикле проверяем, содержат ли ингредиенты цифры. Если цифры не найдены (это проверяется с помощью Char.IsDigit), ингредиент добавляется к строке ingredients. Таким образом, из ингредиентов удаляются все элементы, содержащие цифры. Далее, программа формирует новое заклинание, добавляя к нему префиксы и суффиксы. Эти новые заклинания строятся в методе Spell\_Now\_Done, который использует switch для выбора нужного префикса и суффикса в зависимости от действия. После обработки всех заклинаний, результат (конечное заклинание) выводится на экран.

Код программы представлен ниже.

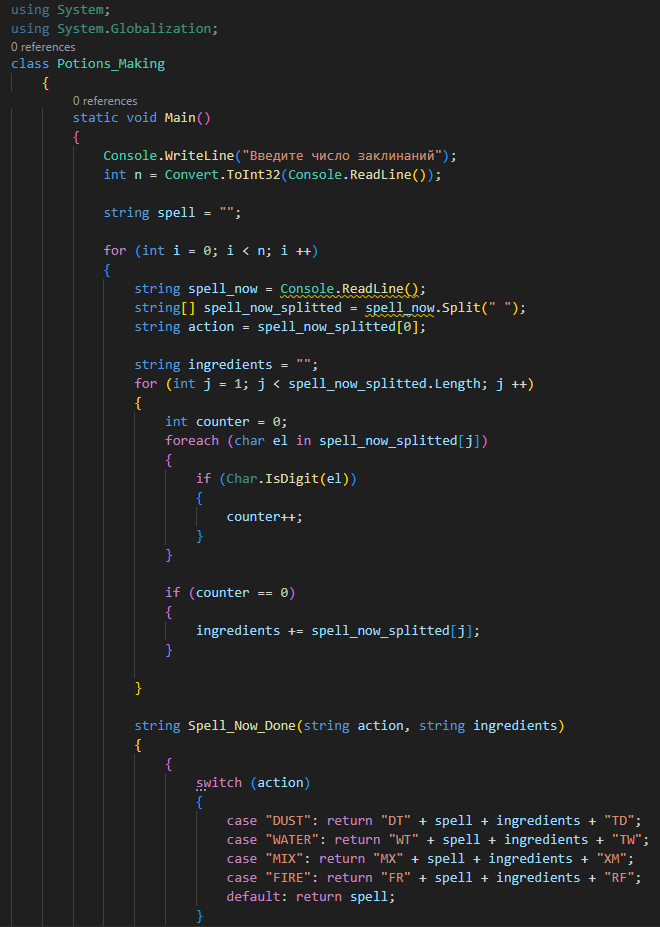


Рисунок 12 - Код задачи «Зельеварение»

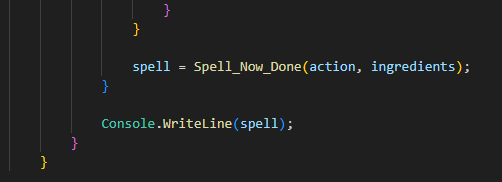


Рисунок 13 - Код задачи «Зельеварение» (продолжение)

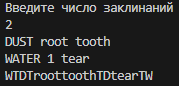
Примеры работы тестовых данных представлены ниже.

Рисунок 14 – Пример работы тестовых данных 1

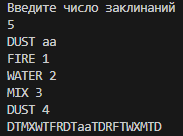


Рисунок 15 - Пример работы тестовых данных 2

Заключение

В ходе выполнения работы, посвященной программной реализации задач на языке C#, были разработаны различные решения основы алгоритмизации, Язык C# оказался мощным инструментом для реализации эффективных решений благодаря своей синтаксической простоте, богатой стандартной библиотеке и мощным встроенным средствам для работы с объектами.

В процессе разработки были продемонстрированы ключевые принципы программирования: использование условных операторов и циклов для обработки данных, методы работы с текстовой и числовой информацией. Также была изучена работа с массивами и строками, что позволило лучше понять концепции обработки данных и оптимизации алгоритмов.

В результате работы были получены практические навыки программирования, что поспособствовало более глубокому пониманию теоретических концепций и облегчило решение более сложных задач.

Таким образом, проделанная работа позволила не только освоить основы языка C#, но и укрепить навыки логического мышления и алгоритмической обработки данных, что является важным этапом в становлении программиста и разработчика.

Список используемой литературы

1. «Microsoft Learn» URL: https://learn.microsoft.com/ru-ru/dotnet/api/system.timespan?view=net-8.0 (Дата обращения: 30.11.2024)
2. «Microsoft Learn» URL: https://learn.microsoft.com/ru-ru/dotnet/api/system.char.isdigit?view=net-9.0 (Дата обращения: 21.11.2024)
3. «Microsoft Learn» URL: https://learn.microsoft.com/ru-ru/dotnet/api/system.convert.todouble?view=net-8.0 (Дата обращения: 30.11.2024)
4. «Microsoft Learn» URL: https://learn.microsoft.com/ru-ru/dotnet/api/system.string.split?view=net-8.0 (Дата обращения: 21.11.2024)
5. «METANIT.COM» URL: https://metanit.com/sharp/tutorial/19.1.php (Дата обращения: 30.11.2024)