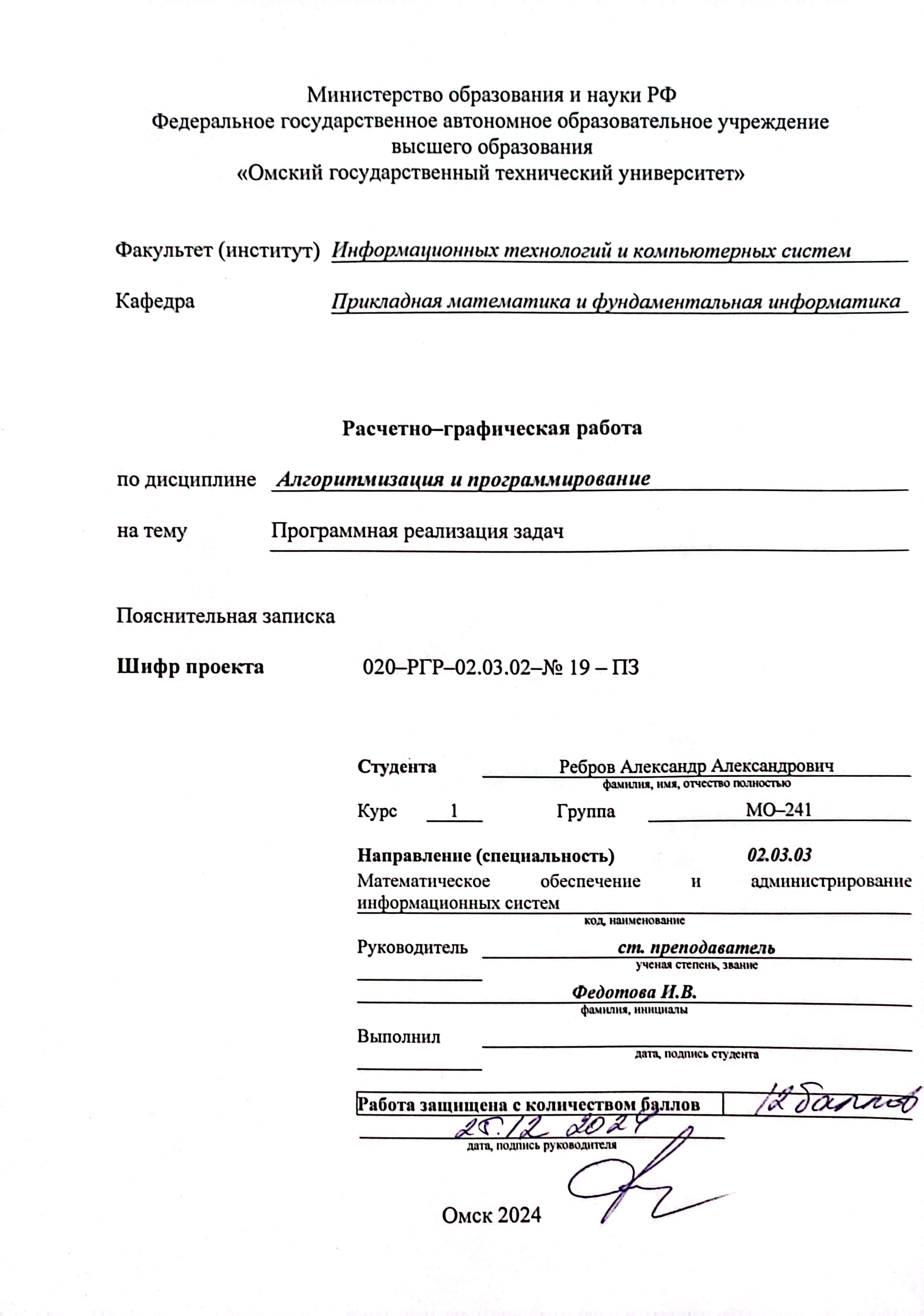
****

**Содержание**

[Введение 3](#_Toc185614637)

[Постановка задачи «Упаковки молока» 4](#_Toc185614638)

[Ход решения задачи «Упаковки молока» 5](#_Toc185614639)

[Постановка задачи «Зельеварение» 9](#_Toc185614640)

[Ход решения задачи «Зельеварение» 11](#_Toc185614641)

[Постановка задачи «Интенсификация производства» 13](#_Toc185614642)

[Ход решения задачи «Интенсификация производства» 14](#_Toc185614643)

[Постановка задачи «Крестьянин и черт» 17](#_Toc185614644)

[Ход решения задачи «Крестьянин и черт» 18](#_Toc185614645)

[Заключение 21](#_Toc185614646)

[Список литературы 22](#_Toc185614647)

**Введение**

В современном мире информационные технологии играют ключевую роль во многих сферах деятельности. Разработка программного обеспечения является одним из наиболее востребованных направлений, а язык программирования C# занимает лидирующие позиции среди разработчиков.

C# — объектно-ориентированный язык программирования, разработанный компанией Microsoft и используемый для создания приложений, работающих на платформе .NET. Он сочетает в себе простоту и мощь, что делает его идеальным инструментом для решения разнообразных задач.

Целью данной расчётно-графической работы является изучение основ языка программирования C#, а также применение его на практике при решении олимпиадных задач. В ходе работы будут рассмотрены основные понятия и концепции C#, такие как работа со строками и числами, использование логических конструкций и функций. Также будет уделено внимание тестированию, отладке и оптимизации программного кода.

**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.

**Ход решения задачи «Упаковки молока»**

Если сформулировать задачу более конкретно, то для её решения необходимо найти стоимость одного литра молока каждой отдельной фирмы.

Для каждой фирмы программа считывает размеры двух видов упаковок (X1,Y1,Z1 для первой упаковки и X2,Y2,Z2 для второй упаковки) и их стоимости (C1 и C2) из консольного ввода. Затем вычисляется объём каждой упаковки (V1 и V2) и площадь поверхности каждой упаковки (S1 и S2).

Далее, по выведенной формуле, представленной на рисунке 1.1, вычисляется стоимость одного литра молока для каждой конкретной фирмы.

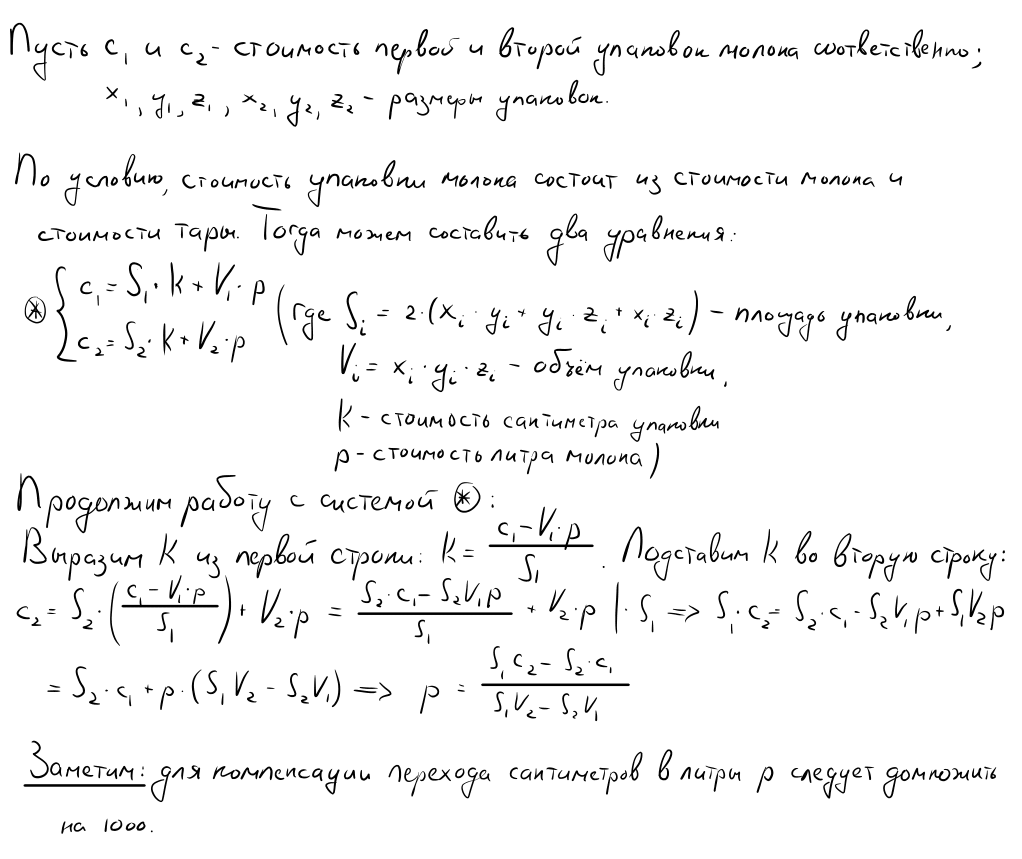
****

Рисунок №1.1 – Вывод формулы расчёта стоимости одного литра молока.

Следующий шаг включает в себя проверку, является ли вычисленная стоимость меньше текущего минимального значения. Если это так, то минимальная стоимость обновляется.

В результате программа осуществляет последовательную обработку всех входных данных и в результате выдаёт минимальную стоимость литра молока и номер фирмы, у которой продаётся самое дешёвое молоко за литр.

Опишем работу этого алгоритма в виде блок-схемы (рисунок 1.2):

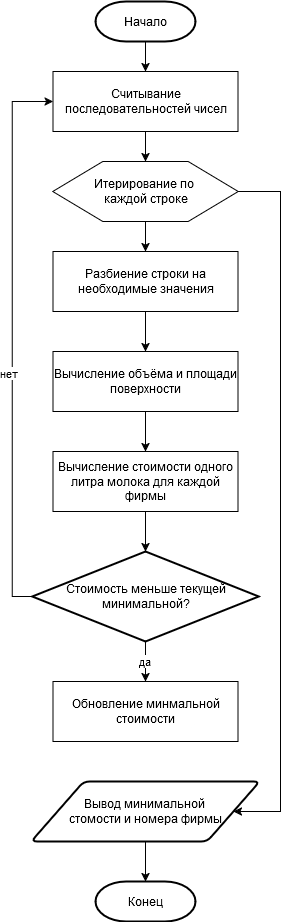
****

Рисунок №1.2 – Блок-схема «Алгоритм решения задачи».

Итоговый код программы представлен на рисунке 1.3.

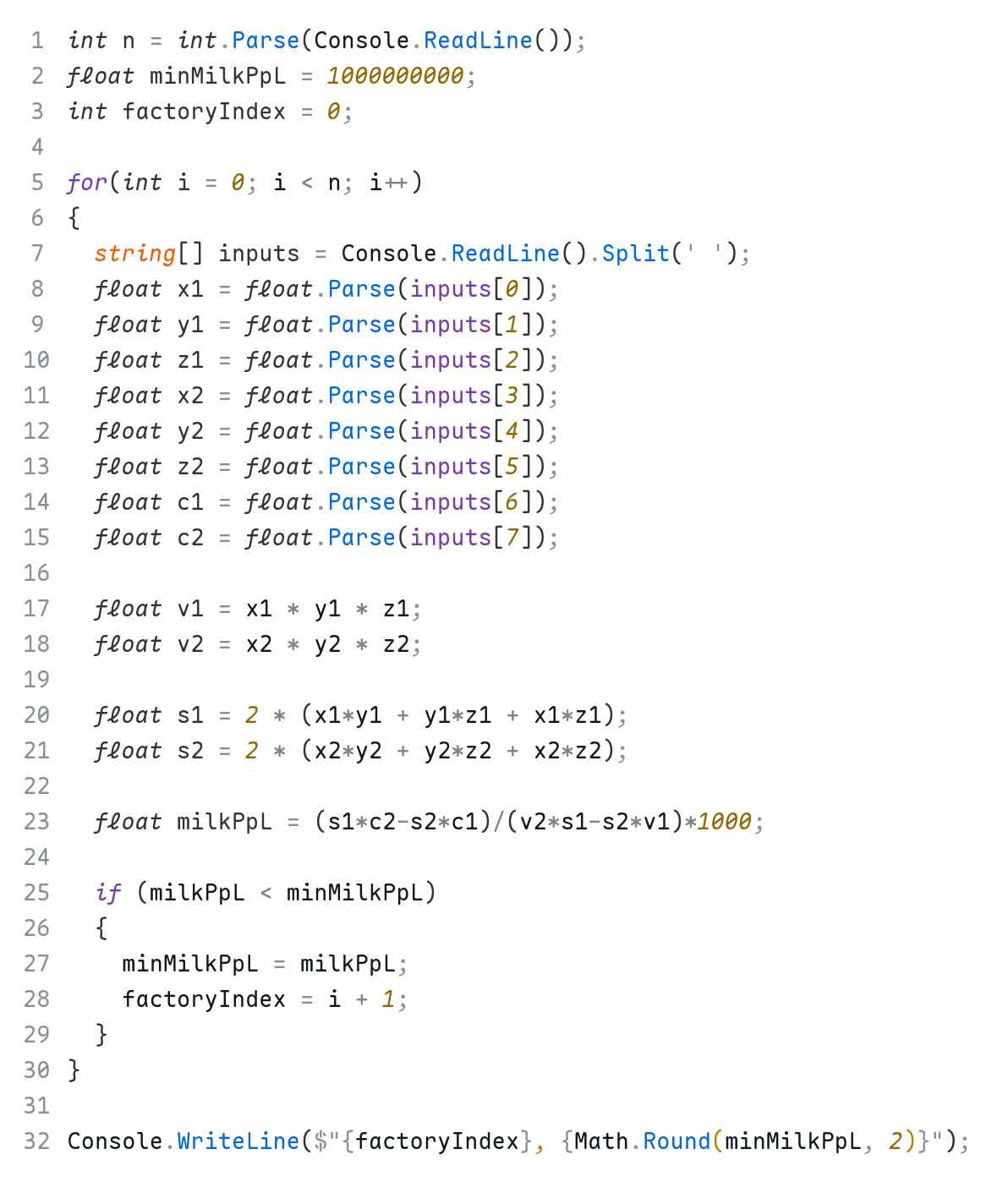


Рисунок №1.3 – Код программы.

Проверим правильность работы программы на тестах (рис 1.4).

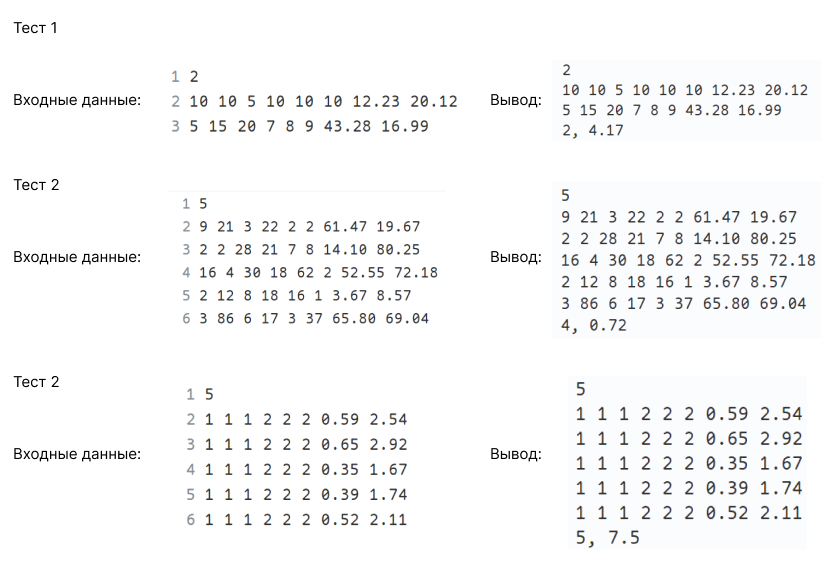


Рисунок №1.4 – Тесты.

**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.

**Ход решения задачи «Зельеварение»**

Если свести задачу к конкретной цели, то необходимо научиться запоминать результат выполнения определённого алгоритма для каждой строки.

Для решения задачи необходимо реализовать два действия: присоединение строки и вставка предыдущего действия.

Заведём цикл, который будет итерироваться по каждой строке текста, полученного на этапе ввода. Затем будем итерироваться по каждому слову строки.

Каждая строка начинается с метода обработки действия. Получая метод обработки, сразу записываем начало и конец будущей строки. После действий начинаются ингредиенты.

У ингредиента проверяем тип данных и в зависимости от результата выполняем конкретное действие. Если ингредиент является строкой, то к итоговой строке добавляем ингредиент. Если же ингредиент является числом, то из списка предыдущих действий достаём нужное и дописываем к строке.

Так поступаем с каждой строкой и в конце получаем верный результат.

Итоговый код программы представлен на рисунке 2.1.



Рисунок №2.1 – Код программы.

Проверим правильность работы программы на тестах (рис 2.2).



Рисунок №2.2 – Тесты.

**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.

**Ход решения задачи «Интенсификация производства»**

Для успешного решения этой задачи нужно научиться считать количество дней между двумя датами. Для этого нужно уметь считать количество дней от начала исчисления до определённой даты.

Расчёт количества дней затрудняют високосные года. В примечании сказано, что високосным считается тот год, который делится нацело на 4. Создадим функцию, которая будет принимать год и возвращать булево значение, указывающее, является ли год високосным.

Затем нужно научиться считать количество дней в определённом месяце. Создадим массив, содержащий количество дней в каждом месяце с учётом возможного високосного года. Запишем это в виде функции, которая будет возвращать количество дней в месяце.

Теперь, используя эти две функции, можно посчитать количество дней от начала исчисления до определённой даты. Создадим счётчик дней и два цикла, которые будут считать количество дней в полных месяцах и количество оставшихся дней соответственно.

Если посчитать таким образом две даты и вычесть вторую из первой, мы получим необходимое количество дней между датами. Суммарный объём продукции можно найти, используя цикл, который будет выполняться столько раз, сколько дней между датами.

Итоговый код программы представлен на рисунке 3.1.



Рисунок №3.1 – Код программы.

Проверим правильность работы программы на тестах (рис 3.2).

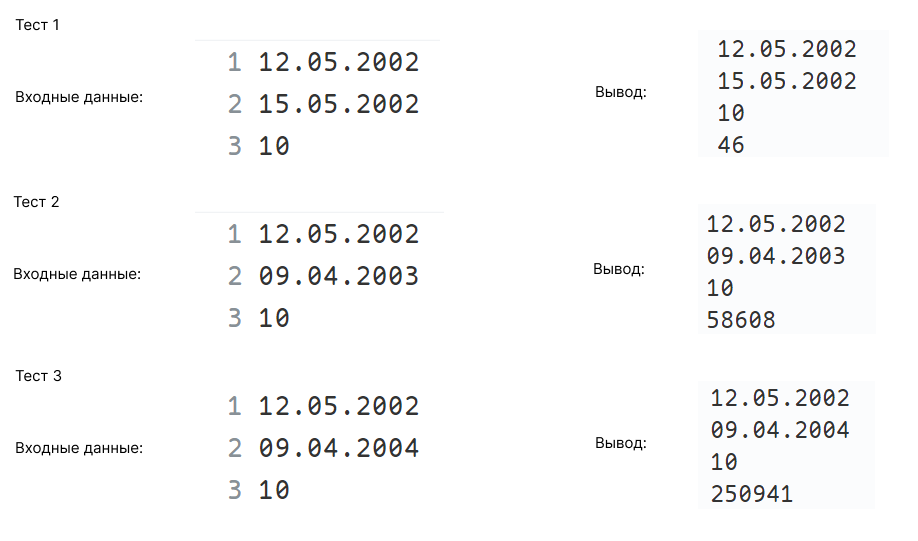


Рисунок №3.2 – Код программы.

**Постановка задачи «Крестьянин и черт»**

Идет крестьянин и плачется: "Эхма! Жизнь моя горькая! Заела нужда совсем! Вот в кармане только несколько монет, да и те сейчас нужно отдать. И как это у других бывает, что на всякие свои деньги они еще деньги получают? Хоть бы кто помочь мне захотел".

Только успел это сказать, как глядь, а перед ним черт стоит и говорит: "Вот видишь этот мост через реку. Стоит тебе перейти через мост, и у тебя будет вдвое больше денег, чем есть. Перейдешь опять, и снова станет вдвое больше. Но за то, что я у тебя деньги удваиваю, после каждого перехода ты мне должен отдавать по **K** монет".

"Ой ли," - сказал крестьянин -"ну-ка, попробуем". Перешел мост, и деньги у него удвоились. Отдал он черту **K** монет, перешел мост еще раз, и опять деньги удвоились. Снова отдал крестьянин черту **K** монет.

Однако после **Z** переходов и отдач черту по **K** монет оказалось, что у крестьянина не осталось ни одной монеты.

Требуется определить, сколько комбинаций условий перехода через мост может быть, если известно, что у крестьянина изначально было не более **MaxN** монет. Комбинацией условий перехода является тройка чисел **N**, **K**, **Z**, где **N** - начальное количество монет у крестьянина, **K** - количество монет, отдаваемых черту после каждого перехода, **Z** - количество переходов. Естественно, что для этой тройки должно выполняться условие, что после **Z** циклов у крестьянина не должно остаться монет.

**Входной файл** содержит целое число **MaxN** - максимальное количество, которое может быть изначально у крестьянина (1 ≤ **MaxN** ≤ 2000000000).

**Выходной файл** должен содержать одно целое число - количество комбинаций условий перехода через мост.

**Ход решения задачи «Крестьянин и черт»**

Решение сводится к тому, чтобы научиться считать число K для любого Z.

Можно выработать два подхода для решения. Один из подходов подразумевает три вложенных цикла, которые будут просчитывать количество комбинаций условий перехода. Проблема этого подхода заключается в том, что он будет выполняться значительное количество времени даже на мелких MaxN.

Попробуем подойти к задаче с точки зрения математики. Если обратить внимание на то, как меняется количество монет с каждым ходом, то можно заметить, что оно образует геометрическую прогрессию. Тогда можно легко найти общую формулу для подсчёта количества монет на любом из переходов (рис. 4.1).

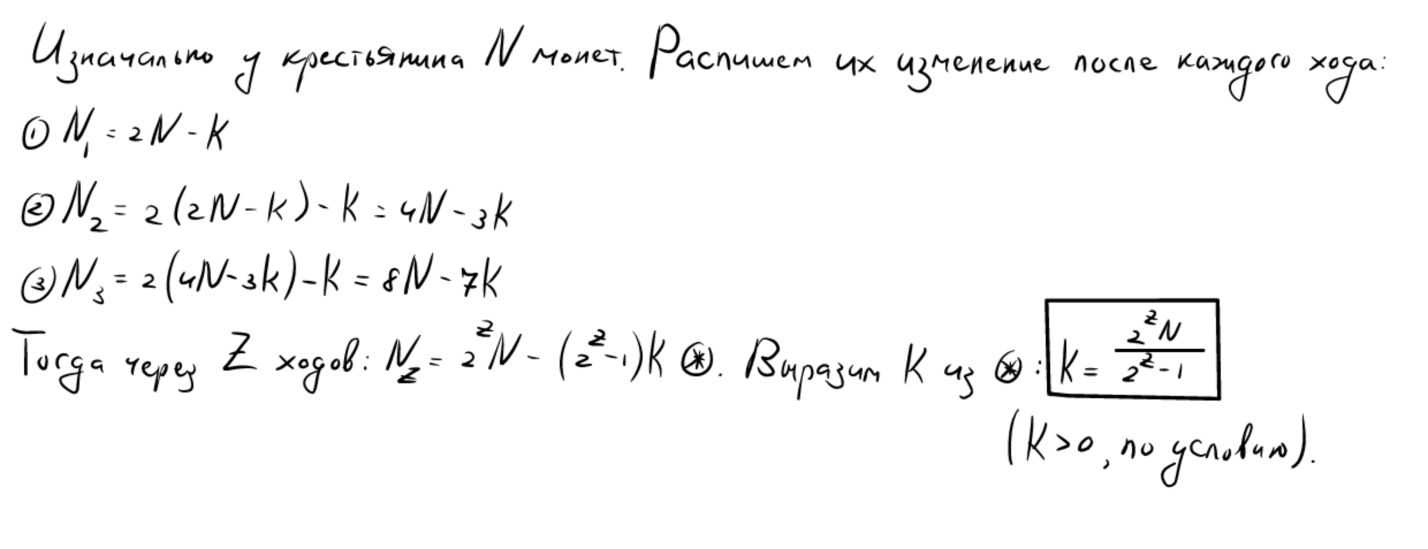


Рисунок №4.1 – Вывод формулы для подсчёта количества монет.

Теперь задачу можно решить средствами комбинаторики. Заведём цикл, который начинается с Z = 1 и длится, условно, бесконечно. На каждой его итерации будем считать значение 2Z и 2Z - 1.

Проверим, меньше или равно 2Z – 1 максимальному количеству монет. Эта проверка нужна для того, чтобы оптимизировать работу программы. Без этой проверки алгоритм будем выполняться существенно дольше.

Теперь можно найти количество допустимых комбинаций с K. Это число прибавим к корневому счётчику. После выхода из цикла в консоль будет выведен результат выполнения программы.

Итоговый код программы представлен на рисунке 4.2.

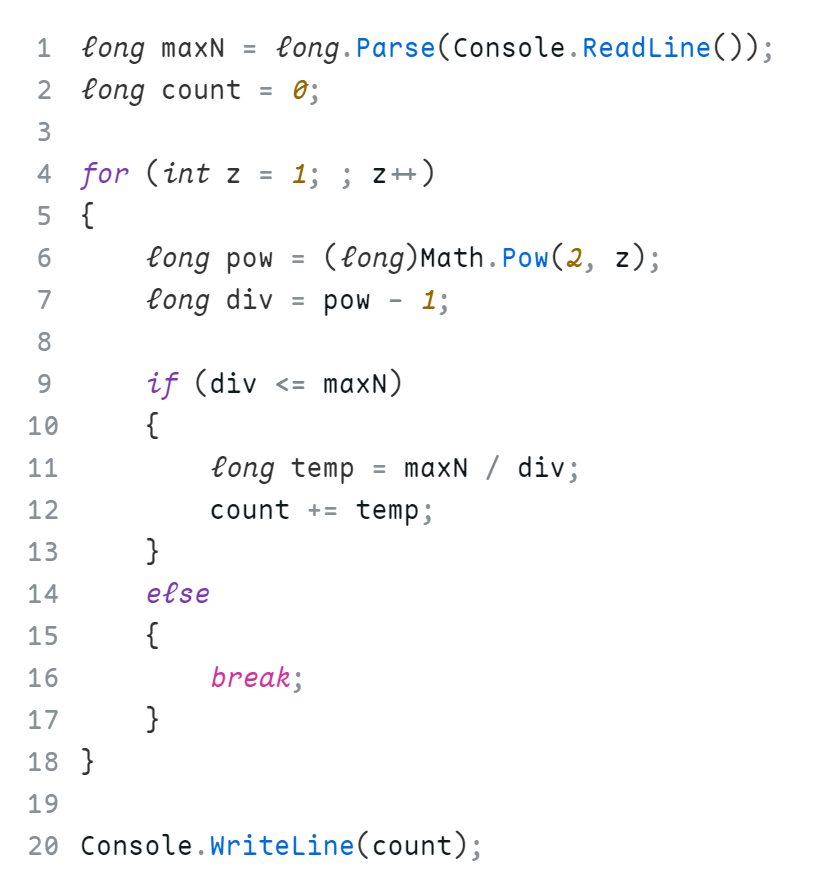


Рисунок №4.2 – Код программы.

Проверим правильность работы программы на тестах (рис.4.3).

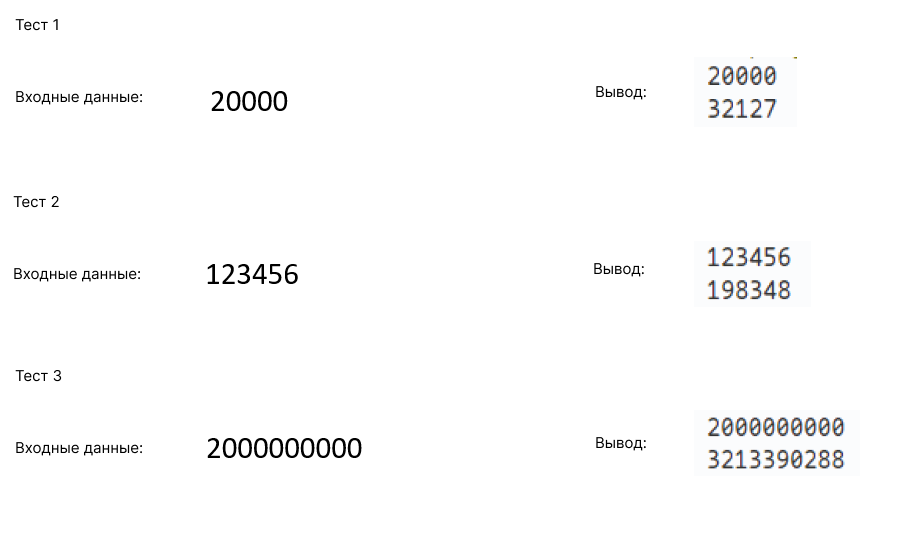


Рисунок №4.3 – Тесты.

**Заключение**

В ходе выполнения работы были изучены основы языка программирования C# и применены полученные знания для решения поставленных задач. Были рассмотрены основные концепции данного языка программирования, проведена работа с классами, объектами и методами. Также были изучены возможности использования стандартных библиотек и работа в среде Visual Studio.

В процессе выполнения работы были созданы несколько программ, демонстрирующих различные аспекты применения языка C# и его возможностей. Были проведены тестирование и отладка созданных программ, что позволило выявить и устранить ошибки и недочёты.

В результате выполнения работы были получены навыки программирования на языке C#, а также опыт разработки и тестирования программного обеспечения. Полученные знания и умения могут быть применены в дальнейшей профессиональной деятельности или при выполнении других проектов.

**Список литературы**

1. Флёнов М. Е. Библия C#. 6-е издание. – БХВ, Санкт-Петербург, 2024 – 512 с.
2. Албахари Д. C#12. Справочник. Полное описание языка. – Диалектика, Санкт-Петербург, 2024 – 272 с.
3. Полное руководство по языку программирования С# 13 и платформе .NET 9: <https://metanit.com/sharp/tutorial/> (дата обращения: 14.11.2024).
4. Что нужно знать при написании алгоритмов на .NET: <https://habr.com/ru/articles/863950/> (дата обращения: 25.10.2024).
5. Особенности строк в .NET: <https://habr.com/ru/articles/172627/> (дата обращения: 05.12.2024).