Содержание

[Введение 3](#_Toc186138662)

[Постановка задачи «Упаковки молока» 4](#_Toc186138663)

[Ход решения задачи «Упаковки молока» 5](#_Toc186138664)

[Постановка задачи «Отгадай число» 8](#_Toc186138665)

[Ход решения задачи «Отгадай число» 9](#_Toc186138666)

[Постановка задачи «Зельеварение» 11](#_Toc186138667)

[Ход решения задачи «Зельеварение» 13](#_Toc186138668)

[Постановка задачи «Интенсификация производства» 16](#_Toc186138669)

[Ход решения задачи «Интенсификация производства» 17](#_Toc186138670)

[Заключение 19](#_Toc186138671)

[Список литературы 20](#_Toc186138672)

# **Введение**

На сегодняшний день, одним из самых популярных и удобных языков программирования является язык C#. Так как это язык общего назначения, его можно использовать для разработки разнообразных приложений: игр**, веб-приложений, мобильных приложений, десктопных приложений, облачных сервисов, применять для искусственного интеллекта и машинного обучения.** С# отличается читаемым и понятным кодом, что облегчает разработку и поддержку приложений.

C# является ключевой частью .NET Framework, предоставляющей огромную библиотеку классов и инструментов для разработки. Это позволяет разработчикам сосредоточиться на создании логики приложения, а не на низкоуровневых деталях. Компания Microsoft активно развивает C# и .NET, регулярно выпуская новые версии с улучшенными функциями и возможностями. Это гарантирует, что C# останется актуальным и востребованным в будущем. С#— это язык, который будет востребован в индустрии, поможет реализовать различные идеи и станет крепким фундаментом для карьеры в IT.

Данная расчётно-графическая работа ставит своей целью исследование основ языка программирования C# и демонстрацию его практической применимости в контексте решения задач олимпиадного типа. В рамках работы будут рассмотрены ключевые аспекты C#, включающие манипуляции с данными типа "строка" и "число", использование логических операторов и реализацию функциональных элементов. Важным компонентом также является отработка навыков тестирования, отладки и оптимизации программного обеспечения.

# **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»**

Для решения задачи необходимо найти стоимость одного литра молока каждой отдельной фирмы.

Программа считывает данные о нескольких фирмах, для каждой из которых заданы размеры двух видов упаковок (X1, Y1, Z1 и X2, Y2, Z2) и их цены (C1 и C2). Исходя из этих размеров, рассчитываются объемы (V1 и V2) и площади поверхностей (S1 и S2) для обеих упаковок. Затем, используя формулу, вычисляется стоимость одного литра молока для каждой фирмы

После этого программа сравнивает вычисленную стоимость с минимальной на данный момент. Если текущая стоимость оказывается меньше, то минимальная стоимость и номер фирмы обновляются. По завершении обработки всех фирм программа выдаёт минимальную стоимость за литр молока и номер фирмы, у которой она была найдена. Представим решение задачи в виде блок-схемы(рис.1).

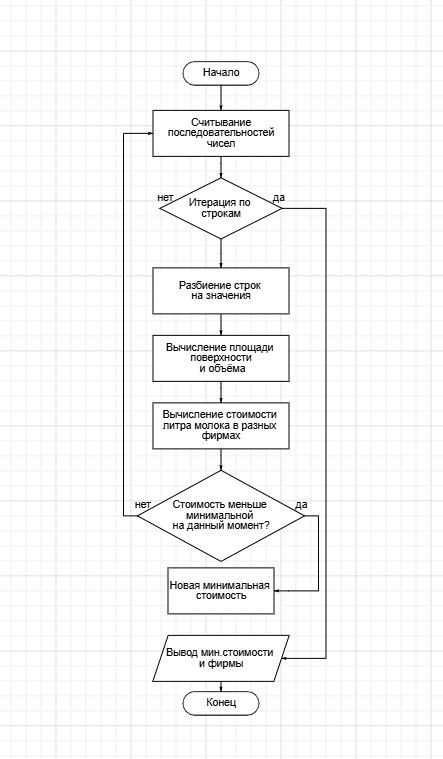


Рисунок 1. Блок-схема.

Итоговый код программы представлен на рисунке 2:

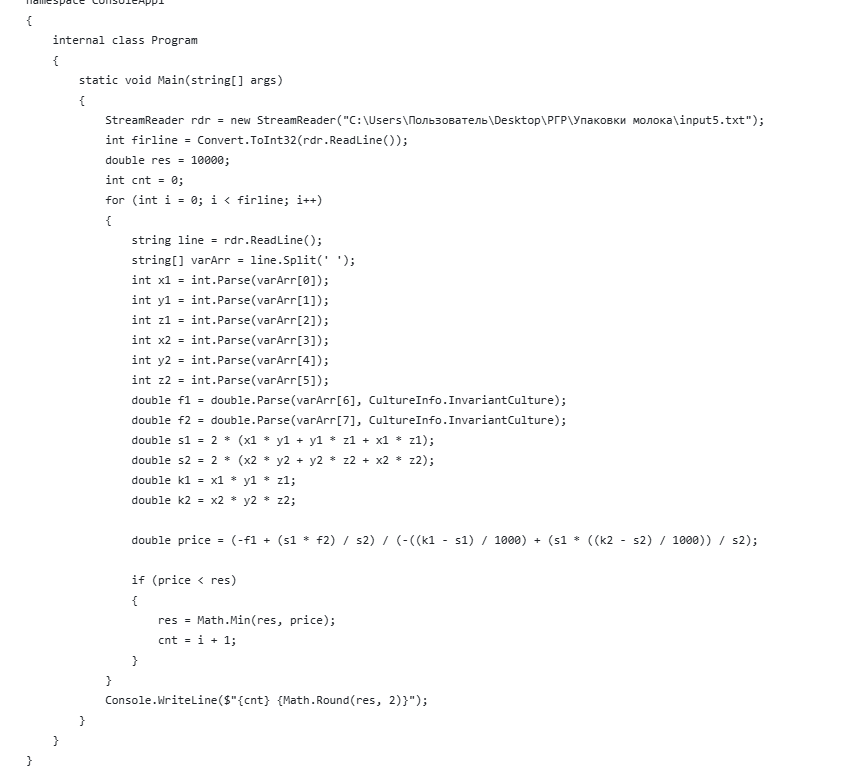


Рисунок 2 – Код программы.

Работа программы на тестах(рис.3):

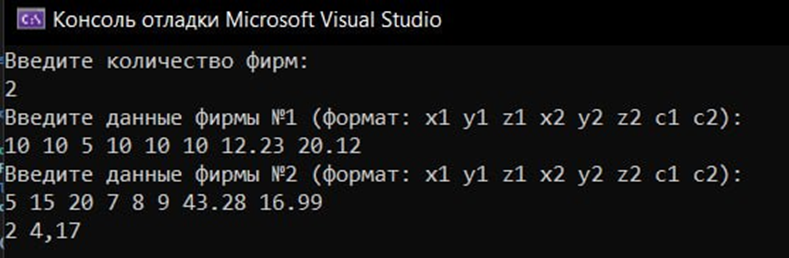


Рисунок 3 Тесты.

# **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

# **Ход решения задачи «Отгадай число»**

Для решения этой задачи необходимо реализовать программу, которая по определенному алгоритму восстанавливает из результата исходное число.

В программе имеется объект для чтения данных из текстового файла и переменная, в которую сохраняется количество строк в файле. Затем идёт строковый массив, предназначенный для хранения строк из файла. Заводим цикл, который считывает строки из файла и сохраняет их в массив. Первая строка перезаписана символом «x». выводим содержимое массива в консоль. Далее инициализируем переменные, используемые для вычислений, и заводим цикл, который обрабатывает элементы массива. Проходим по массиву, анализируя каждую строку. Если встречается число, то оно используется для вычисления промежуточного результата numRes. Если встречается символ "x", то изменяется счетчик cntX. Затем вычисляем результат путём вычитания из последнего элемента массива промежуточного результата и деления этой разности на счётчик. Выводим результат на консоль.

Итоговый код программы (рис.4):

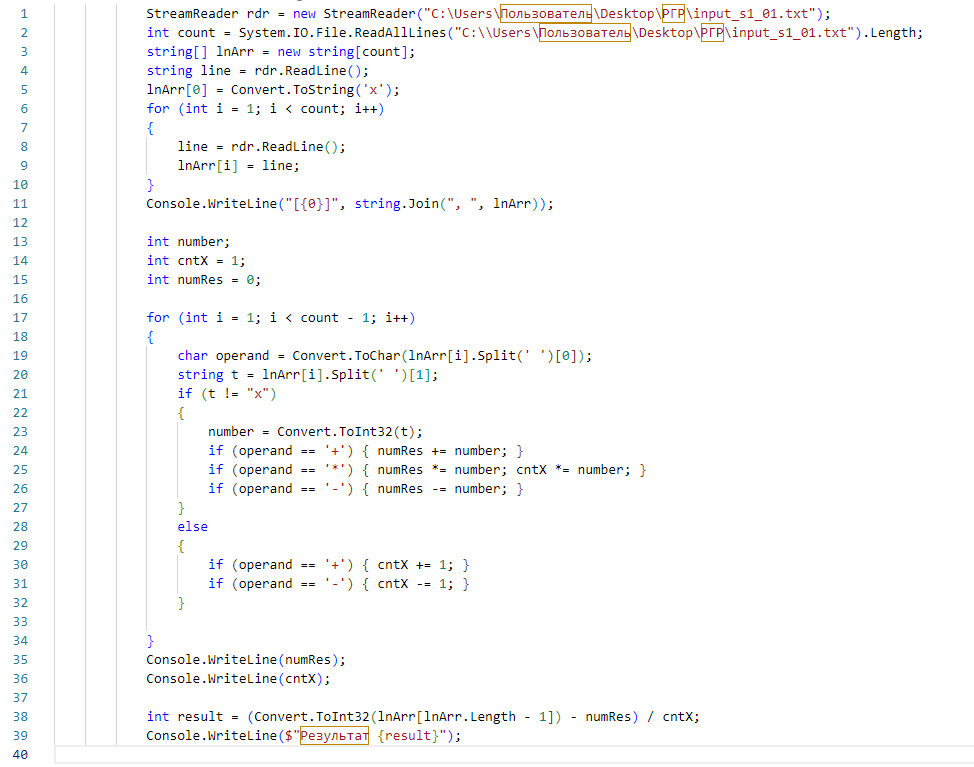


Рисунок 4 – Код программы.

Работа программы показана на тестах(рис.5):

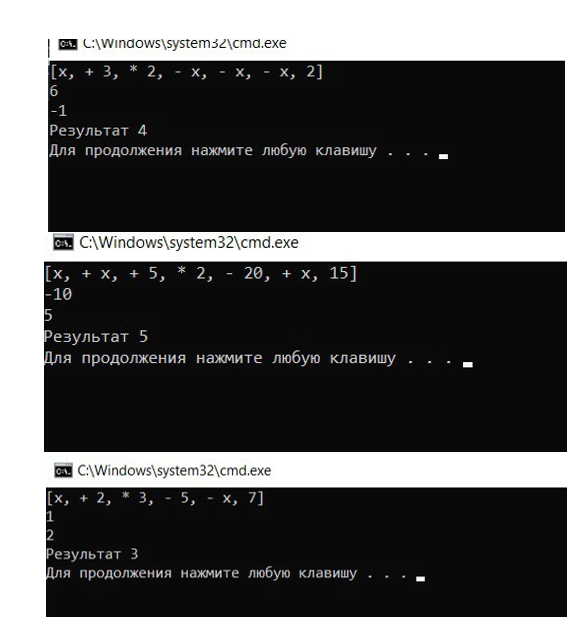


Рисунок 5. Тесты.

# **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»**

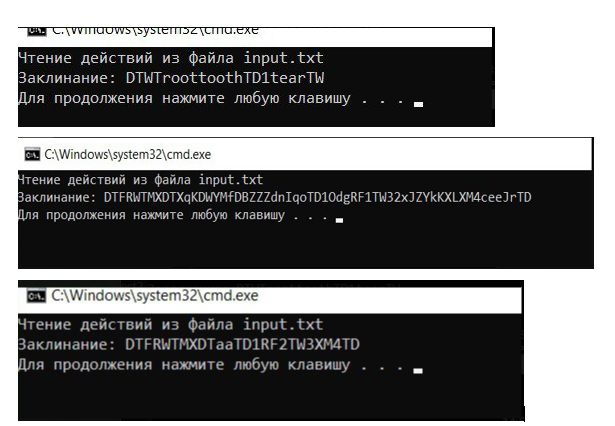
Если свести задачу к конкретной цели, то необходимо научиться запоминать результат выполнения определённого алгоритма для каждой строки.

Программа обрабатывает команды и ингредиенты, вводимые пользователем, для создания итогового "заклинания". Каждая команда (MIX, WATER, DUST, FIRE) добавляет к заклинанию определенный префикс и суффикс, используя предоставленные ингредиенты. Программа считывает команды и ингредиенты до тех пор, пока не будет введено "END". В процессе обработки выполняется проверка корректности ввода команд. Входные данные, разделенные пробелами, разбиваются на отдельные списки и помещаются в зубчатый массив. После определения команды программа формирует строку заклинания, объединяя префикс, ингредиенты и суффикс. В конце выводится результирующая строка, содержащая итоговое "заклинание".

Итоговый код представлен на рисунке:

Рисунок 6 Код программы.

Работа программы на тестах: (рис.7):

Рисунок 7 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»**

Программа вычисляет период времени, за который произведено заданное количество продукции. Сначала определяется разница между начальной и конечной датами, выраженная в днях. Затем, используя эту разницу и заданный объем продукции, программа рассчитывает суммарный объем, предполагая увеличение производства на единицу каждый день.

Входные данные включают начальную и конечную даты, а также объем продукции. Программа проверяет, что разница между датами находится в диапазоне от 1 до 60000 дней. Результатом работы программы является итоговый объем продукции, произведенный за рассчитанный период.

Итоговый код программы представлен на рисунке 8.

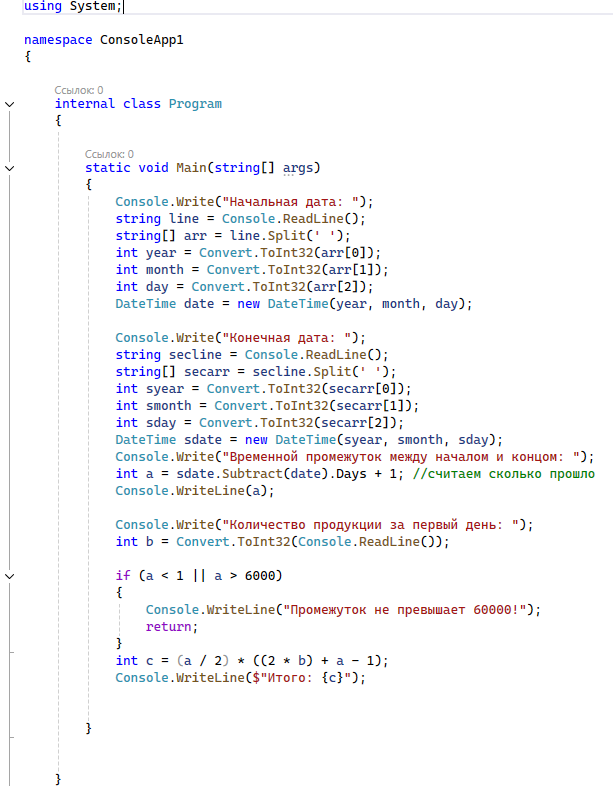


Рисунок 8. Код программы.

Работа программы на тестах: (рис.9):

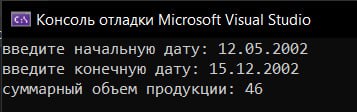


Рисунок 9 Тесты.

# **Заключение**

Данная работа была направлена на изучение основ языка программирования C# и их применение в процессе решения поставленных задач.

В рамках исследования были рассмотрены ключевые концепции языка, включая работу с классами, объектами, методами, а также возможности стандартных библиотек и среды разработки Visual Studio. В ходе выполнения работы были разработаны программные решения, демонстрирующие различные аспекты применения языка C# и его инструментария. Проведена отладка и тестирование программ, что обеспечило выявление и устранение ошибок.

В результате работы были получены навыки программирования на языке C# и опыт разработки и тестирования программного обеспечения, применимые в будущей профессиональной деятельности.

# **Список литературы**

1. Мюллер Д. П. C# для чайников. – Диалектика, Санкт-Петербург, 2019 – 609 с.
2. Стиллмен Э. Head First. Изучаем C#. – O`Reilly, Москва, 2022 – 769 с.
3. Руководство по C# — управляемый язык .NET: [https://learn.microsoft.com/ru-ru/dotnet/csharp/](https://learn.microsoft.com/ru-ru/dotnet/csharp/%20) (дата обращения: 30.10.2024).
4. Полное руководство по языку программирования С# 13 и платформе .NET 9: <https://metanit.com/sharp/tutorial/> (дата обращения: 15.11.2024).
5. C#: Конкатенация: <https://code-basics.com/ru/languages/csharp/lessons/concatenation> (дата обращения: 01.12.2024).