**Содержание**

[Введение 3](#_Toc186377872)

[Постановка задачи «Упаковки молока» 4](#_Toc186377873)

[Ход решения задачи «Упаковки молока» 5](#_Toc186377874)

[Постановка задачи «Интенсификация производства» 8](#_Toc186377875)

[Ход решения задачи «Интенсификация производства» 9](#_Toc186377876)

[Постановка задачи «Отгадай число» 11](#_Toc186377877)

[Ход решения задачи «Отгадай число» 12](#_Toc186377878)

[Постановка задачи «Зельеварение» 15](#_Toc186377879)

[Ход решения задачи «Зельеварение» 17](#_Toc186377880)

[Заключение 20](#_Toc186377881)

[Список литературы 21](#_Toc186377882)

**Введение**

Наша жизнь всё сильнее связывается с вычислительной техникой, которая автоматизируют различные процессы и упрощают работу, а потому важно быстро и правильно доносить свои мысли до неё. Компьютеры понимают только последовательность из нулей и единиц, а не привычный нам язык, и любая команда, поданная человеком, должна принимать именно такую форму. Для этого придумали языки программирования, которые способны наладить диалог между человеком и машиной. Человек по определенным правилам пишет команды, которые автоматически переводятся в машинный код. Но спектр задач у компьютера огромный, а потому невозможно придумать идеальный язык программирования, который подойдет под все ситуации и будет максимально эффективным. Поэтому появилось множество языков, которые эффективны в своей области. Они постепенно совершенствовались и упрощались, повышая уровень абстракции, чтобы было легче писать программы. Так мы прошли путь от Fortran к C#, JavaScript, Python и другим.

С# является высокоуровневым объектно-ориентированным языком программирования, что делает его относительно простым в изучении и использовании. Он унаследовал лучшие черты от своих предшественников: с и с++ ­–­ что делает его эффективным в работе с аппаратной частью устройства, но при этом был модифицирован: в него добавили автоматическую сборку мусора – контроль памяти и её освобождение от неиспользуемых объектов, что значительно упрощает жизнь программисту.

Целью данной расчётно-графической работы является изучение основ языка программирования C#, а также применение его для решения практических задач. В ходе работы будут изучены основные понятия и принципы C#, работа с основными типами данных, использование логических конструкций и функций. А также будут задействованы основные алгоритмические структуры, и будет уделено внимание тестированию и отладке 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**: cчитать 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 1 литра у i-ой фирмы. Для этого у нас есть параметры Xi1, Yi1, Zi1 и Xi2, Yi2, Zi2– размеры первой и второй упаковок, и их итоговые цены Ci1 и Сi2. Все эти параметры мы считываем из консольного ввода.

Следующим шагом мы вычисляем объем и площадь поверхности j-ой упаковки у i-ой фирмы (формула 1) (1≤j≤2, 0≤i≤N). После этого с помощью формулы 2 можно посчитать стоимость молока, не забыв перевести кубические сантиметры в литры, умножив на 1000. После каждого вычисления необходимо сравнить результат с наименьшим значением. Если он меньше, то присвоить наименьшему значению наш результат. Осталось проделать эту последовательность действий для каждой i-ой фирмы. После всех итераций на выходе получим минимальную стоимость молока.

Формула 1 – нахождение объема и площади поверхности/

Формула 2 – Вывод формулы расчёта стоимости одного литра молока

Опишем работу алгоритма с помощью блок-схемы (рисунок 1).

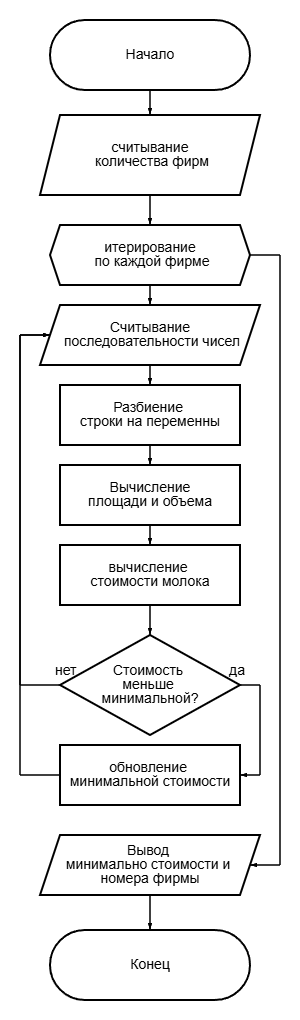


Рисунок 1– Блок схема «Алгоритм решения задачи».

Код программы представлен на рисунке 2, а тесты на рисунке 3.

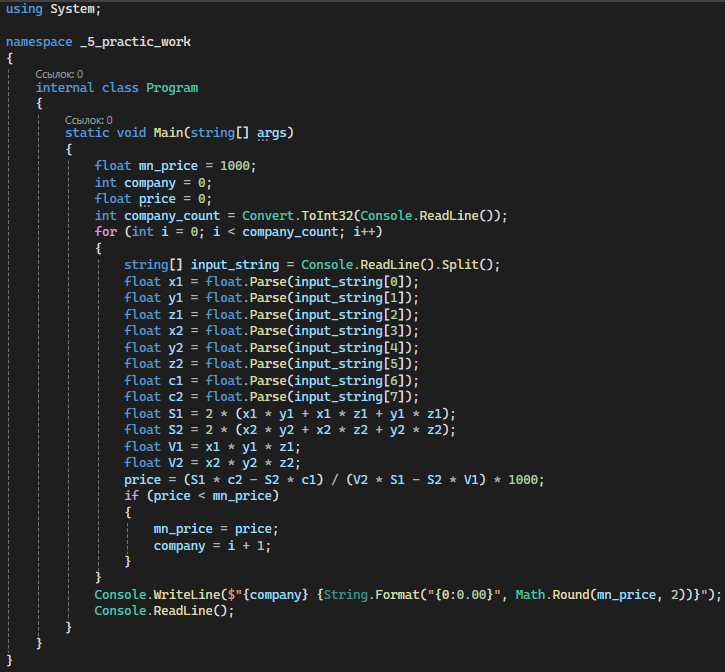


Рисунок 2 – Код программы.

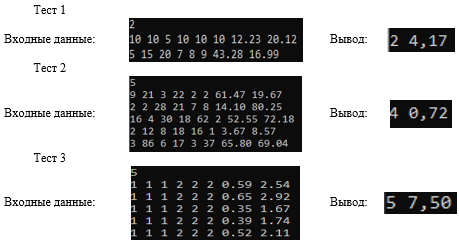


Рисунок 3 – Тесты.

**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»**

Эту задачу удобно решать путём перевода даты в количество дней, прошедших от начала летоисчисления – даты 00.00.00. А после перевода первой и второй даты в этот формат вычесть из большего числа меньшее, тем самым найдём количество дней между ними.

Функция перевода даты в дни считает количество полных лет и переводит их в дни, не забывая учесть високосный год (по условию високосный год – каждый четвертый год). После этого прибавляет количество дней в каждом полном месяце (информация о количестве дней в месяце хранится в массиве). После этого осталось прибавить количество оставшихся дней.

Наращивание производства представляет собой арифметическую прогрессию с шагом один, вводимым в консоль начальным значением и количеством членов, равное количество дней между датами. Зная все эти параметры можно воспользоваться формулой суммы арифметической прогрессии.

Код программы представлен на рисунке 4, а тесты на рисунке 5.

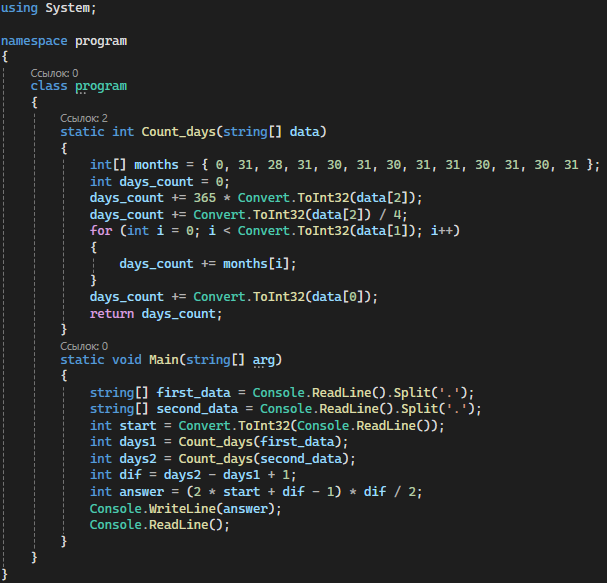


Рисунок 4 – Код программы

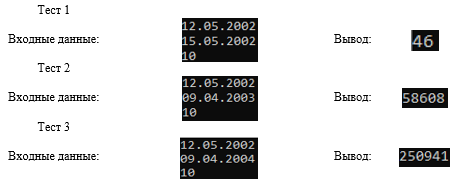


Рисунок 5 – Тесты

**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 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**: содержит одно целое число - задуманное число **X**.

**Ход решения задачи «Отгадай число»**

Решение задачи сводится к решению уравнения (1). A и B изменяется в зависимости от операции. Заведём массив, который будет хранить коэффициенты A и B в виде. В конце, после вычисления A и B и ввода С можно вычислить x по формуле (1).

Сперва считаем количество действий, после этого войдем в цикл. 1 элемент строки всегда операция. Дальше в зависимости от операции вызываем функцию, в которую передаем два аргумента: второй элемент введённой строки (будем называть его k) и массив, хранящий коэффициенты нашего уравнения.

Если мы в функции Minus или Plus, мы проверяем: если k нельзя перевести в число, значит мы ввели x, поэтому 1 либо прибавляем к A, либо вычитаем из A. Иначе мы ввели число, которое надо прибавить либо к B, либо отнять от B. Если наша функция Multiply, то мы и A, и B умножаем на k. В конце выполнения функции возвращаем измененный массив.

После всех итераций подставляем коэффициенты в формулу и находим значение x – загаданное число.

Уравнение 2 – вывод формулы нахождения x.

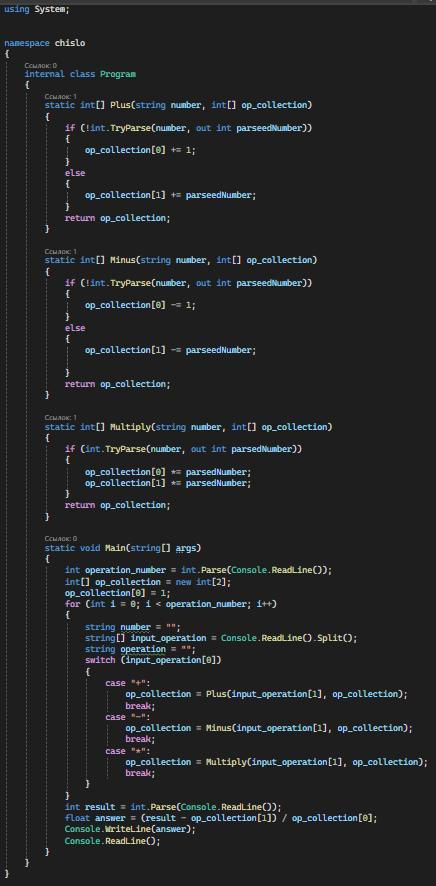
  
Рисунок 6 – Код программы.



Рисунок 7 – 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»**

Решение задачи сводится к трём действиям: присоединение введённой строки, вставка предыдущих строк, и присоединение слева и справа по два символа операции. Запоминание предыдущих операций будем делать через строку, которую будем превращать в массив. Эту строку будем называть «память»

Каждая считываемая строка строится по принципу: «операция» «ингредиенты». Создадим массив из этой строки – где первый элемент – операция, а остальные «ингредиенты». В зависимости от его значения мы сразу запоминаем начало и конец будущей строки. Затем передадим «память» и ингредиенты в функцию.

Функция проверяет тип данных, и если его можно перевести в число, то мы обращаемся к «памяти» с индексом [число-1] и добавляем этот элемент к конечной строке. Иначе мы просто добавляем эту строку к возвращаемому результату. Так надо итерироваться по всем элементам из «ингредиентов». В конце возвращаем полученную строку.

После получения строки из функции необходимо поместить её между запомненными нами ранее началом и концом. Полученный результат записываем в «память». Таким образом повторяем алгоритм пока не получим слово END. Оно означит остановку и вывод последнего шага. Разбиваем «память» в массив и выдаём последний элемент.

Код программы представлен на рисунке 4, а темы на рисунке 5.



Рисунок 8 – Код программы.

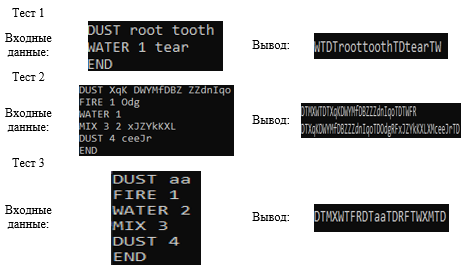


Рисунок 9 – Тесты.

**Заключение**

В ходе выполнения работы были изучены основы языка C# и применены знания для решения практических задач. Были изучены основные типы данных и методы данного языка, освоены основы функционального программирования и сформировалось понимание работы функции. Так же были получены навыки работы в среде Visual Studio.

В процессе выполнения работы были созданы четыре программы, показывающие особенности и возможности применения языка С#. Был получен навык тестирования и отладки программ.

В результате выполнения работы я получил ценные знания и навыки программирования на языке C#. Но помимо этого я так же получил знания об алгоритмах, работе с данными и функциями, что так же полезно при использовании в других языках, поскольку включает общие фундаментальные принципы. Язык С# стал хорошей точкой входа в программирование. И теперь, с полученным опытом, я дальше смогу развиваться в этом направлении.

**Список литературы**

1. Мюллер Д.П. C# для чайников, 3-e издание – Диалектика, Москва, 2019 – 591 с.
2. Полное руководство по языку программирования C# 13 и платформе .NET 9: <https://metanit.com/sharp/tutorial/> (дата обращения 15.12.2024).
3. Руководство по C# – управляемый язык .NET | Microsoft Learn: <https://learn.microsoft.com/ru-ru/dotnet/csharp/> (дата обращения 10.12.2024)
4. C# уроки для чайников - курсы обучения C# для начинающих с нуля на itProger: <https://itproger.com/course/csharp> (дата обращения 16.12.2024)
5. Первая программа на C#: <https://gb.ru/blog/pervaya-programma-na-c-sharp/> (дата обращения 20.12.2024).