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–№ 23 – ПЗ |

|  |  |  |  |  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- |
|  |  |  | **Студента** | | Григорьева Романа Павловича | | | | | |
|  |  |  |  | | фамилия, имя, отчество полностью | | | | | |
|  |  |  | Курс | 1 |  | Группа | | ФИТ–241 | | |
|  |  |  |  |  |  | |  |  | |  |
|  | | | **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

**Содержание**

[Введение 3](#_Toc185607578)

[Постановка задачи 1 4](#_Toc185607579)

[Ход решения задачи 1 5](#_Toc185607580)

[Постановка задачи 2 8](#_Toc185607581)

[Ход решения задачи 2 9](#_Toc185607582)

[Постановка задачи 3 11](#_Toc185607583)

[Ход решения задачи 3 12](#_Toc185607584)

[Постановка задачи 4 15](#_Toc185607585)

[Ход решения задачи 4 16](#_Toc185607586)

[Заключение 17](#_Toc185607587)

[Список используемой литературы 19](#_Toc185607588)

**Введение**

Высокопроизводительные языки программирования предназначены для создания приложений, требующих максимальной скорости выполнения и эффективного использования ресурсов. Эти языки обеспечивают низкоуровневый доступ к аппаратуре, минимальные накладные расходы, а также оптимизацию памяти и процессорного времени. Для выполнения данной расчётно-графической работы мною был выбран язык C#, который позволил эффективно решить поставленные задачи, обеспечив оптимальное сочетание производительности и простоты реализации.

**Постановка задачи 1**

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 задачи 1**

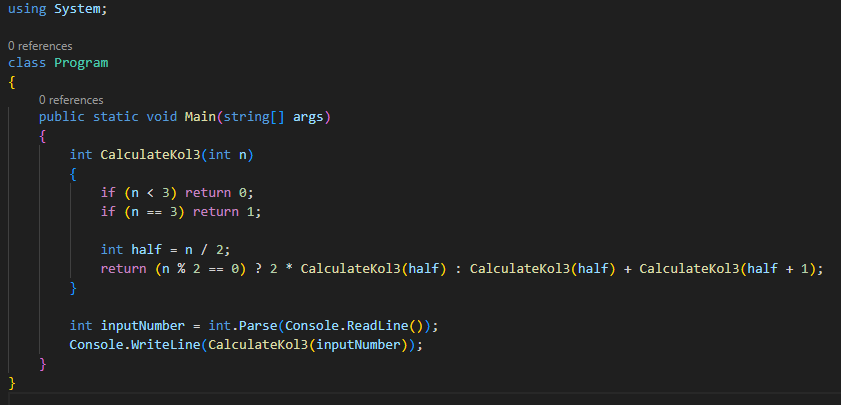
Для решения задачи применяется рекурсивный подход, при котором общее количество солдат разделяется на две подгруппы — с четными и нечетными позициями. Рекурсия завершается, если в группе остается меньше трёх солдат. В случае, когда солдат ровно три, формируется одна группа. Для каждой из подгрупп рекурсивно вызывается функция, которая подсчитывает и суммирует количество возможных групп. Пример реализации данного метода представлен в коде ниже:

Рисунок 1

Примеры работы на тестовых данных приведены на Рисунок 2 – Рисунок 4:



Рисунок 2



Рисунок 3



Рисунок 4

нет

Ввод N

N<3?

Вернуть 0

N=3?

Вернуть 1

N четное?

Вернуть 2 \* CalculateKol3(N /2)

Вернуть CalculateKol3(N) + CalculateKol3(N+1)

да

да

нет

да

нет

**Постановка задачи 2**

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**:

1. гарантируется, что имеется только один ответ;
2. 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:

1. **S** - тип действия, состоящий из одного символа: "\*" - умножить; "-" - отнять; "+" - прибавить;
2. **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.
3. Последняя строка содержит результат **R** (|**R**| <= 2 000 000 000).
4. **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 2**

Метод решения задачи реализован с использованием конструкции `switch-case` для обработки последовательности операций. Каждая команда, модифицирующая коэффициенты уравнения t=ax+b, обрабатывается в соответствующем блоке `case`. После выполнения всех операций вычисляется исходное значение x на основе итогового значения r. Код программы представлен ниже:

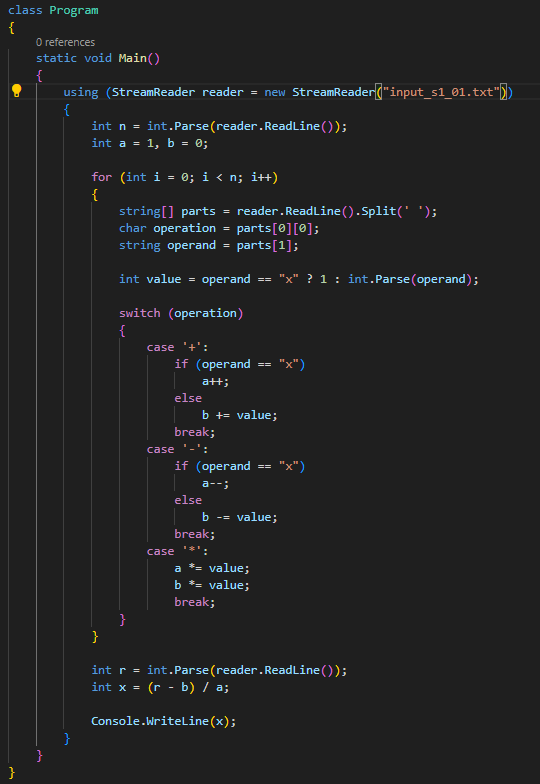


Рисунок 5

Примеры работы на тестовых данных приведены на рисунок 6 – рисунок 8:

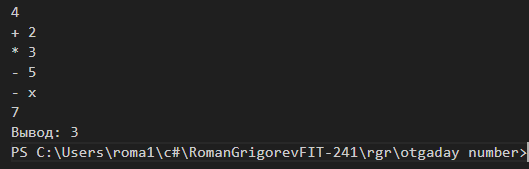


Рисунок 6

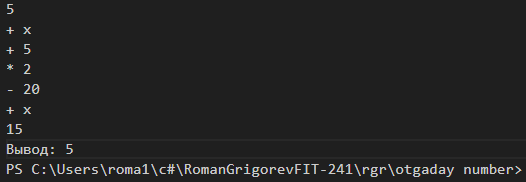


Рисунок 7

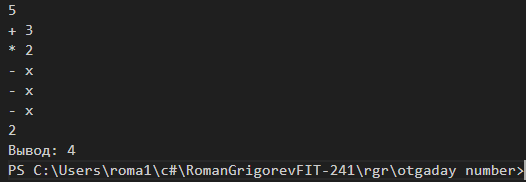


Рисунок 8

**Постановка задачи 3**

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 3**

Программа перебирает все возможные значения начальных монет N и стоимости K для каждого перехода. Для каждого значения она вычисляет результат после каждого перехода, проверяя, что после Z переходов у крестьянина остаётся 0 монет. Количество таких комбинаций и выводится. Так как количество монет, которые отдает крестьянин (k) не может быть больше 1n + 1 (Исходя из того, что по условию задачи k<=2n, чтобы при первом проходе крестьянин не был должен больше, чем у него есть. Заметим, что k может принимать целые значения только до 1n, поэтому сокращаем цикл до 1n и в конце прибавляем к ответу maxn(=по одному прибавлению к count с каждого k=2n), так как такое условие выполняется всегда), было использовано сокращение цикла k, для более быстрой работы программы на больших значениях

Код программы представлен ниже:

Программа перебирает все возможные значения начальных монет N и стоимости K для каждого перехода. Для каждого сочетания вычисляется результат после каждого перехода, проверяя, что после Z переходов у крестьянина остаётся ровно 0 монет. Выводится количество таких комбинаций.

Так как количество монет, отдаваемых крестьянином (K), не может превышать 1n + 1 (по условию задачи k<=2n, чтобы при первом переходе крестьянин не был должен больше, чем у него есть), перебор значений K ограничивается 1n. Цикл был сокращён для ускорения программы на больших значениях.

Пример реализации программы приведён ниже.

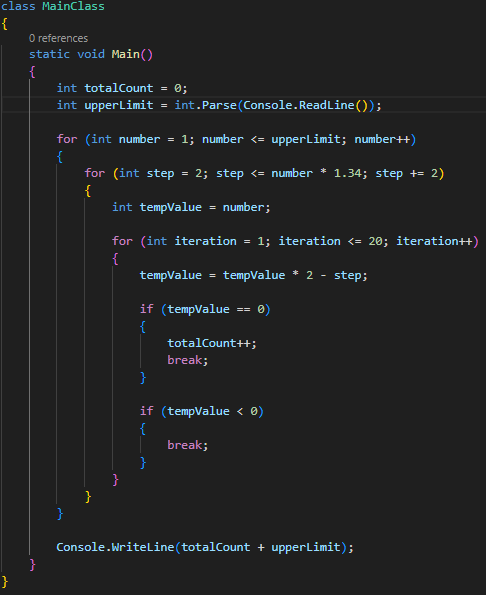


Рисунок 9

Примеры работы на тестовых данных приведены на рисунок 10 – рисунок 12:



Рисунок 10



Рисунок 11



Рисунок 12

**Постановка задачи 4**

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 4**

Решение задачи основано на соотношении площадей поверхности, объемов и цен упаковок. Программа вычисляет стоимость одного литра молока для каждой фирмы и сравнивает её с минимальной ценой, найденной ранее, обновляя минимальное значение, если текущая цена меньше.

Код программы представлен ниже:

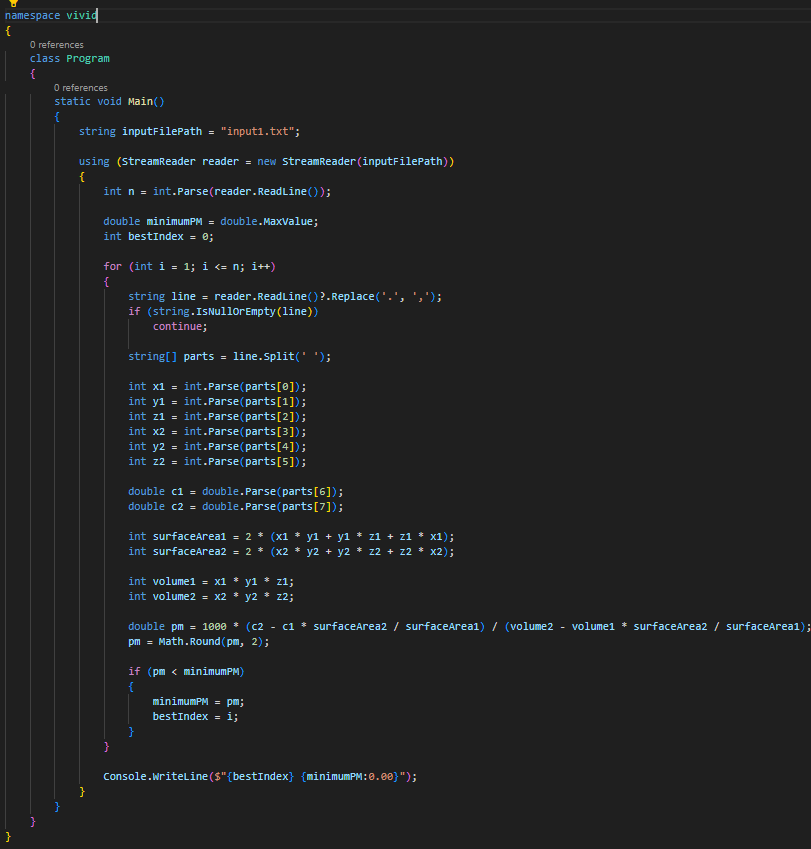


Рисунок 13

Примеры работы на тестовых данных приведены на рисунок 14 – рисунок 16:

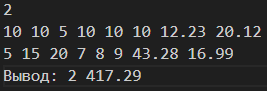


Рисунок 14

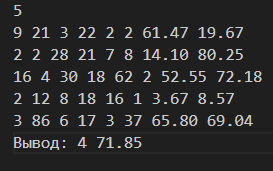


Рисунок 15

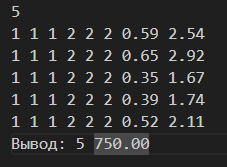


Рисунок 16

**Заключение**

В ходе выполнения расчётно-графической работы были изучены возможности языка программирования C#, что позволило глубже понять его применение для решения разнообразных задач. Были исследованы и использованы необходимые формулы и алгоритмы, обеспечившие достижение поставленных целей, а также рассмотрены ключевые особенности синтаксиса и структуры языка. Практическая реализация задач на C# продемонстрировала его эффективность и удобство при работе с различными типами данных и решении программных задач. Полученные знания и навыки создают основу для разработки более сложных проектов и приложений в будущем.

**Список используемой литературы**

Microsoft, C# Documentation, url: <https://learn.microsoft.com/en-us/dotnet/csharp/> (дата обращения: 28 декабря 2024)

Andrew Troelsen, Philip Japikse, Pro C# 9 with .NET 5: Foundational Principles and Practices in Programming. – Apress, Berkeley, 2021. – 1080 с.

Joseph Albahari, Ben Albahari, C# 10 in a Nutshell: The Definitive Reference. – O'Reilly Media, Sebastopol, 2021. – 1050 с.

Christian Nagel, Professional C# and .NET. – Wiley, Indianapolis, 2021. – 1560 с.