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–№ 21 – ПЗ |

|  |  |  |  |  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- |
|  |  |  | **Студента** | | Шустова Якова Дмитриевича | | | | | |
|  |  |  |  | | 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 2](#_Toc185628072)

[Постановка задачи 1 3](#_Toc185628073)

[Решение задачи 1 5](#_Toc185628074)

[Постановка задачи 2 9](#_Toc185628075)

[Решение задачи 2 11](#_Toc185628076)

[Постановка задачи 3 14](#_Toc185628077)

[Решение задачи 3 16](#_Toc185628078)

[Постановка задачи 4 19](#_Toc185628079)

[Решение задачи 4 21](#_Toc185628080)

[Заключение 24](#_Toc185628081)

[Список использованной литературы 25](#_Toc185628082)

Введение

C# — это современный объектно-ориентированный язык программирования, разработанный компанией Microsoft в рамках платформы .NET. Он был представлен в 2000 году как часть проекта .NET и быстро завоевал популярность благодаря своей гибкости, мощным возможностям и удобству синтаксиса.

C# поддерживает широкий спектр парадигм программирования, включая объектно-ориентированное, функциональное и императивное. Он активно используется для создания приложений для Windows, веб-сервисов, приложений и игр через Unity. Язык обладает сильной типизацией, что помогает уменьшить количество ошибок на этапе компиляции. C# также активно поддерживает асинхронное программирование, благодаря чему разработчики могут эффективно работать с многозадачностью и масштабируемыми приложениями.

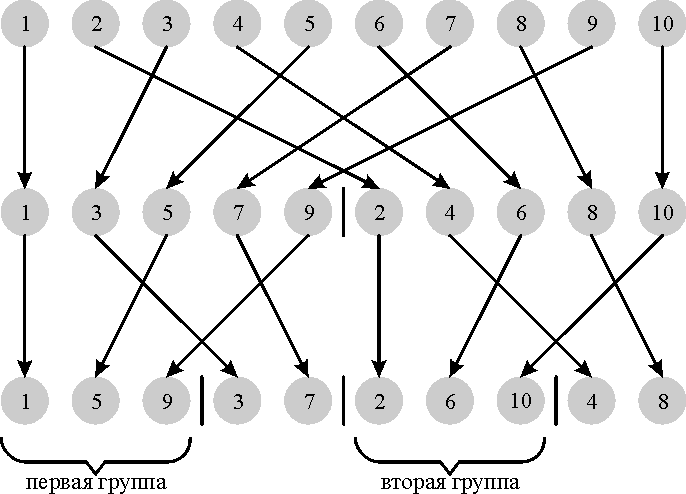
С момента своего появления C# продолжает развиваться, оставаясь одним из самых востребованных языков в разработке программного обеспечения.

Постановка задачи 1

**Отбор в разведку**

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.

*Пример 1:*

*Input.txt*

*10*

*Output.txt*

*2*

*Пример 2:*

*Input.txt*

*4*

*Output.txt*

*0*

Решение задачи 1

Ход программы представлен на рисунках ниже:

Начало

number <= 3

number == 3

count++

Конец

int group\_one = 0

int group\_two = 0

number%2==0

group\_one = number/2

group\_two = number/2

group\_one = number/2

group\_two = number/2 + 1

Divide\_g(group\_one)

Divide\_g(group\_two)

нет

да

да

нет

да

нет

Divide\_g(int number)  
деление на две группы

Рисунок 1 – Блок-схема Divide\_g()

Рисунок 2 – Блок-схема Main()

Начало

Ввод

data

Divide\_g(data)

Конец

Код программы:

using System;

using System.ComponentModel;

using System.Diagnostics.Metrics;

namespace MyApp

{

internal class Recon

{

static void Main(string[] args)

{

int data = Convert.ToInt32(Console.ReadLine());

Divide\_g(data);

Console.WriteLine(count);

Console.ReadLine();

}

static int count = 0;

static void Divide\_g(int number)

{

if (number <= 3)

{

if (number == 3)

{

count++;

}

return;

}

int group\_one = 0;

int group\_two = 0;

if (number % 2 == 0)

{

group\_one = number / 2;

group\_two = number / 2;

}

else

{

group\_one = (number / 2);

group\_two = (number / 2) + 1;

}

Divide\_g(group\_one);

Divide\_g(group\_two);

}

}

}

На рисунках 3 – 5 показаны результаты работы программы.

Рисунок 3 – пример работы программы

Рисунок 4 – пример работы программы

Рисунок 5 – пример работа программы

Для решения данной задачи была использована рекурсивная функция. Divide\_g() принимает на вход количество человек, и продолжает делить их на две подгруппы до тех пор, пока в них не останется по три или меньше человека. Если разведчиков ровно 3, то группа счётчик увеличивается на 1.

Постановка задачи 2

**Упаковки молока**

*Максимальное время: 0,2 с.*

*Максимальная память: 16 MB*

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*

Решение задачи 2

Код решения предоставлен ниже:

using System;

using System.ComponentModel.DataAnnotations;

using System.Globalization;

using System.Runtime.CompilerServices;

using System.Runtime.Intrinsics.Arm;

namespace MyApp

{

internal class Program

{

static void Main(string[] args)

{

int number = Convert.ToInt32(Console.ReadLine());

Run(number);

Console.ReadKey();

}

static void Run(int number)

{

string[] lines = File.ReadAllLines($"input/input{number}.txt");

int firms\_num = Convert.ToInt32(lines[0]);

double min\_milk = Double.MaxValue;

int best\_firm = 0;

int count = 0;

for (int i = 0; i < lines.Length; i++)

{

string[] temp\_line = lines[i].Split();

if (i > 0)

{

double prev\_milk = min\_milk;

double X1 = Convert.ToDouble(temp\_line[0]);

double Y1 = Convert.ToDouble(temp\_line[1]);

double Z1 = Convert.ToDouble(temp\_line[2]);

double X2 = Convert.ToDouble(temp\_line[3]);

double Y2 = Convert.ToDouble(temp\_line[4]);

double Z2 = Convert.ToDouble(temp\_line[5]);

double C1 = Convert.ToDouble(temp\_line[6]);

double C2 = Convert.ToDouble(temp\_line[7]);

double k = (X2 \* Y2 \* Z2) / (X1\*Y1\*Z1);

double S1 = 2 \* (X1 \* Y1 + X1 \* Z1 + Z1\* Y1);

double S2 = 2 \* (X2 \* Y2 + X2 \* Z2 + Z2\* Y2);

min\_milk = Math.Min(((C1 - (S1 \* Math.Abs((C2 - k \* C1) / (S2 - k \* S1)))) / (X1\*Y1\*Z1))\* 1000, min\_milk);

if (min\_milk < prev\_milk)

{

best\_firm = i;

}

count++;

}

}

Console.WriteLine($"{best\_firm}, {Math.Round(min\_milk, 2).ToString("0.00")}");

}

}

}

Результаты работы программы представлены на рисунках 6 – 8.

Рисунок 6 – результат работы программы



Рисунок 7 – результат работы программы



Рисунок 8 – результат работы программы

На вход подаётся номер файла с данными, записанными согласно условию задачи.

*Файл 1*

*2*

*10 10 5 10 10 10 12.23 20.12*

*5 15 20 7 8 9 43.28 16.99*

*Файл 2*

*5*

*9 21 3 22 2 2 61.47 19.67*

*2 2 28 21 7 8 14.10 80.25*

*16 4 30 18 62 2 52.55 72.18*

*2 12 8 18 16 1 3.67 8.57*

*3 86 6 17 3 37 65.80 69.04*

*Файл 3*

*5*

*1 1 1 2 2 2 0.59 2.54*

*1 1 1 2 2 2 0.65 2.92*

*1 1 1 2 2 2 0.35 1.67*

*1 1 1 2 2 2 0.39 1.74*

*1 1 1 2 2 2 0.52 2.11*

Для решения данной задачи была использована следующая формула:

Которая была получена из формул объёма параллелепипеда и площади его поверхности:

Постановка задачи 3

**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*

Решение задачи 3

Код решения зачади:

using System;

using System.Linq;

using System.Reflection.PortableExecutable;

namespace MyApp

{

internal class Program

{

static void Main(string[] args)

{

string number = Console.ReadLine();

Solution(number);

Console.ReadKey();

}

static double CoefficientX(string x)

{

string cf = x.Replace("x", "");

if (cf == "")

{

return 1.0d;

}

else if (cf == "-")

{

return -1.0d;

}

else

{

return Convert.ToDouble(cf);

}

}

static double Ops(string k, string k1, string op)

{

double kf = CoefficientX(k);

double a = CoefficientX(k1);

if (op == "+")

{

kf += a;

}

else if (op == "-")

{

kf -= a;

}

else if (op == "\*")

{

kf \*= a;

}

else if (op == "/")

{

kf /= a;

}

return kf;

}

static void Solution(string number)

{

string[] lines = File.ReadAllLines($"input/input{number}.txt");

double result = Convert.ToDouble(lines[lines.Length-1]);

double[] pair = [1, 0];

for (int i = 1; i < lines.Length-1; i++)

{

string[] temp\_line = lines[i].Split();

if (temp\_line[1].Count(f => f == 'x') != 0)

{

pair[0] = Ops(Convert.ToString(pair[0]), temp\_line[1], temp\_line[0]);

}

else

{

if (temp\_line[0] == "+" || temp\_line[0] == "-")

{

pair[1] = Ops(Convert.ToString(pair[1]), temp\_line[1], temp\_line[0]);

}

else

{

pair[0] = Ops(Convert.ToString(pair[0]), temp\_line[1], temp\_line[0]);

pair[1] = Ops(Convert.ToString(pair[1]), temp\_line[1], temp\_line[0]);

}

}

}

double answer = (result - pair[1]) / pair[0];

Console.WriteLine(answer);

}

}

}

Результаты работы программы приведены на рисунках 9 – 11.

Рисунок 9 – Результат работы программы

Рисунок 10 – Результат работы программы

Рисунок 11 – Результат работы программы

На вход подаётся номер файла с данными, записанными согласно условию задачи.

*Файл 1*

*4*

*+ 2*

*\* 3*

*- 5*

*- x*

*7*

*Файл 2*

*5*

*+ x*

*+ 5*

*\* 2*

*- 20*

*+ x*

*15*

*Файл 3*

*5*

*+ 3*

*\* 2*

*- x*

*- x*

*- x*

*2*

Решение задачи основано на нахождении исходного числа, зная конечный результат и операции, приводящие к нему. Таким образом, над коэффициентом при *x* и свободным членом производятся операции, обратные к исходным. Это реализуется благодаря методам и массивам.

Постановка задачи 4

**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*

Решение задачи 4

Код решения зачади:

using System;

using System.ComponentModel;

namespace MyApp

{

internal class Program

{

static void Main(string[] args)

{

string number = Console.ReadLine();

Run(number);

Console.ReadKey();

}

public static string Reverse( string s )

{

char[] charArray = s.ToCharArray();

Array.Reverse(charArray);

return new string(charArray);

}

public static string Dict(string a, Dictionary<string, string> brew, string[] array)

{

if (a.Any(char.IsDigit))

{

if (brew.ContainsKey(a) == false)

{

brew[a] = array[Convert.ToInt32(a)-1];

return brew[a];

}

else

{

return brew[Convert.ToString(Convert.ToInt32(a) - 1)];

}

}

else

{

return a;

}

}

public static void Run(string number)

{

string[] lines = File.ReadAllLines($"input/input{number}.txt");

string[] potions = new string[lines.Length];

var brew = new Dictionary<string, string>()

{

{ "MIX", "MX"},

{ "WATER", "WT"},

{ "DUST", "DT"},

{ "FIRE", "FR"}

};

for (int i = 0; i < lines.Length; i++)

{

string[] temp\_line = lines[i].Split();

for (int k = 1; k < temp\_line.Length; k++)

{

potions[i] += Dict(temp\_line[k], brew, potions);

}

potions[i] = brew[temp\_line[0]] + potions[i] + Reverse(brew[temp\_line[0]]);

brew[Convert.ToString(i)] = potions[i];

}

Console.WriteLine(potions.Last());

}

}

}

Результаты работы программы приведены на рисунках 12 – 14.

Рисунок 12 – результат работы программы

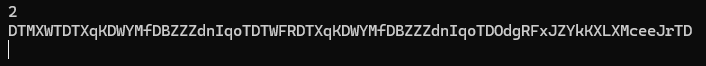


Рисунок 13 – результат работы программы



Рисунок 14 – результат работы программы

На вход подаётся номер файла с данными, записанными согласно условию задачи.

*Файл 1*

*DUST root tooth*

*WATER 1 tear*

*Файл 2*

*DUST XqK DWYMfDBZ ZZdnIqo*

*FIRE 1 Odg*

*WATER 1*

*MIX 3 2 xJZYkKXL*

*DUST 4 ceeJr*

*Файл 3*

*DUST aa*

*FIRE 1*

*WATER 2*

*MIX 3*

*DUST 4*

Решение данной задачи основано на записи зелий в словарь с целыми числами в качестве ключей и последующим обращении (добавлении в рецепт) к ним при встрече нужного индекса в файле.

Заключение

В ходе выполнения данной расчётно-графической работы были изучены на практике основы C#. А именно: разные типы данных, конвертация одного типа в другой, операторы ветвления, цикла, массивы, словари, встроенные методы строк и массивов для разбиения входных данных на символы, применялась интерполяция.

Использовались методы встроенных классов Console (для ввода-вывода информации через консоль), Math (для разных математических операций, например, взятия модуля Abs()), File (для работы с файлами), а также написаны собственные методы, выполняющие самые разные функции (например, рекурсивный метод).

Помимо базы языка программирования были использованы и математические формулы, позволяющие вычислять объём и площадь поверхности параллелепипеда.

Список использованной литературы

1. Stack Overflow – <https://stackoverflow.com/> (01.12.2024)
2. Microsoft Learn – <https://learn.microsoft.com/ru-ru/dotnet/csharp/> (01.12.2024)
3. W3Schools – <https://www.w3schools.com/cs/index.php> (03.12.2024)
4. METAINT.COM Сайт о программировании – <https://metanit.com/> (04.12.2024)
5. GeeksforGeeks – <https://www.geeksforgeeks.org/> (04.12.2024)