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** | | Передериной Софьи Владимировны | | | | | |
|  |  |  |  | | фамилия, имя, отчество полностью | | | | | |
|  |  |  | Курс | 1 |  | Группа | | ФИТ-242 | | |
|  |  |  |  |  |  | |  |  | |  |
|  | | | **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](#_Toc183430238)

[Постановка задачи 1 4](#_Toc183430239)

[Ход решения задачи 1 6](#_Toc183430240)

[Постановка задачи 2 8](#_Toc183430241)

[Ход решения задачи 2 9](#_Toc183430242)

[Постановка задачи 3 11](#_Toc183430243)

[Ход решения задачи 3 13](#_Toc183430244)

[Постановка задачи 4 15](#_Toc183430245)

[Ход решения задачи 4 17](#_Toc183430246)

[Заключение 19](#_Toc183430247)

[Список используемой литературы 20](#_Toc183430248)

Введение

С# является объектно-ориентированным языком программирования, выпущенным в январе 2002 года командой инженеров компании Microsoft под руководством Андерса Хейлсберга и Скотта Вильтаумота. C# принадлежит к группе С-подобных языков, многие особенности его синтаксиса схожи с Java и C++.

К одним из основных полезных функций языка относятся инкапсуляция, наследование, полиморфизм, автоматическое управление памятью, кроссплатформенность. Кроме того, C# подразумевает использование строгой типизации данных, что помогает предотвратить ошибки выполнения программы, обеспечивает ее надежность.

Объектно-ориентированный подход позволяет разработчику создавать в C# крупные, но в то же время гибкие приложения, которые можно модифицировать по мере необходимости и легко использовать повторно.

Одним из важных преимуществ использования C# является интеграция с .NET платформой. Она обеспечивает возможность высокой совместимости с Windows – приложениями, а также позволяет использовать многообразные библиотеки и компоненты, созданные на других языках .NET.

Благодаря преимуществам языка С# находит широкое применение в различных сферах разработки. Его возможности позволяют создавать игры на DirectX и Unity, использовать в приложениях машинное обучение и нейросети, создавать веб-приложения.

Постановка задачи 1

Максимальное время: 0,1 с.

Максимальная память: 4 MB

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.

*Пример:*

*Input.txt*

*3*

*Output.txt*

*4*

*Пояснение: возможны комбинации условий перехода*

*N K Z*

*1 2 1*

*2 4 1*

*3 6 1*

*3 4 2*

Ход решения задачи 1

Ход решения задачи представлен ниже на рисунке 1.

d

ans +=

Начало

Ввод Max\_N

ans = 0

N =

Z = 1,N

Вывод ans

Рисунок – Блок-схема

﻿

using System;

using System.Collections.Generic;

using System.Linq;

using System.Text;

using System.Threading.Tasks;

namespace ConsoleApp6

{

internal class Задача\_3

{

static void Main(string[] args)

{

Console.WriteLine("Входные данные");

int max\_n = int.Parse(Console.ReadLine());

ulong ans = 0;

int n = (int)(Math.Log(max\_n) / Math.Log(2) + 2);

for (int z = 1; z < n; z++)

{

int d = (int)(Math.Pow(2, z) - 1);

ans += (ulong)(max\_n / d);

}

Console.WriteLine($"Выходные данные: {ans}");

}

}

}

На рисунках 2 – 4 представлены примеры работы программы.

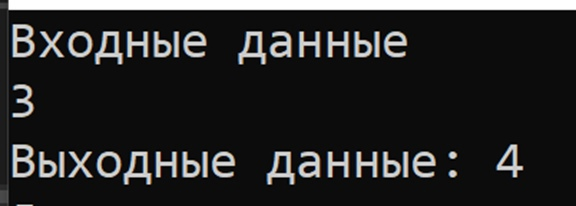


Рисунок – Первый пример работы программы

Изображение выглядит как текст, Шрифт, снимок экрана, число

Автоматически созданное описание

Рисунок 3 – Второй пример работы программы

Изображение выглядит как текст, Шрифт, снимок экрана

Автоматически созданное описание

Рисунок 4 – Третий пример работы программы

Для решения задачи была использована формула перехода логарифма к новому основанию:

.

Постановка задачи 2

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.

*Пример:*

*Input.txt*

*12.05.2002*

*15.05.2002*

*10*

*Output.txt*

*46*

Ход решения задачи 2

Код программы представлен ниже.

using System;

namespace ConsoleApp6

{

internal class Интенсификация\_производства

{

static ulong allDays(string s)

{

ulong allDays = 0;

string[] strings = s.Split('.');

allDays += Convert.ToUInt16(strings[0]);

ulong month = Convert.ToUInt16(strings[1]);

ulong year = Convert.ToUInt16(strings[2]);

allDays += 365 \* (year - 1) + ((year - 1) / 4);

if (year % 4 == 0) allDays++;

switch (month)

{

case 1: allDays += 0; break;

case 2: allDays += 31; break;

case 3: allDays += 59; break;

case 4: allDays += 90; break;

case 5: allDays += 120; break;

case 6: allDays += 151; break;

case 7: allDays += 181; break;

case 8: allDays += 212; break;

case 9: allDays += 243; break;

case 10: allDays += 273; break;

case 11: allDays += 304; break;

case 12: allDays += 334; break;

}

return allDays;

}

static void Main(string[] args)

{

Console.WriteLine("Входные данные");

string data1 = Console.ReadLine();

string data2 = Console.ReadLine();

ulong begin = Convert.ToUInt16((Console.ReadLine()));

ulong workDays = allDays(data2) - allDays(data1) + 1;

ulong result = ((2 \* begin + workDays - 1) \* workDays) / 2;

Console.WriteLine($"Выходные данные {result}");

}

}

}

На рисунках 5 – 7 представлены примеры работы программы.

Изображение выглядит как текст, Шрифт, снимок экрана, число

Автоматически созданное описание

Рисунок – Первый пример работы программы

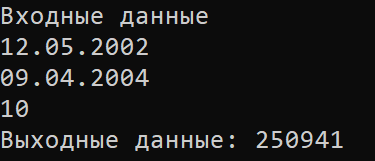


Рисунок – Второй пример работы программы

Рисунок – Третий пример работы программы

Изображение выглядит как текст, Шрифт, снимок экрана

Автоматически созданное описание

Для решения задачи была использована формула суммы членов арифметической прогрессии:

.

Постановка задачи 3

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*

Ход решения задачи 3

Код программы представлен ниже.

using System;

using System.Collections.Generic;

using System.Diagnostics.CodeAnalysis;

using System.Dynamic;

using System.Linq;

using System.Text;

using System.Threading.Tasks;

namespace ConsoleApp6

{

internal class Program

{

static void Main(string[] args)

{

Console.WriteLine("Входные данные");

int n = int.Parse(Console.ReadLine());

int sm = 0;

int k = 1;

for (int i = 0; i < n; i++)

{

string[] s = (Console.ReadLine()).Split(' ');

if (s[0] == "+")

{

if (s[1] == "x") k += 1;

else sm += Convert.ToInt32(s[1]);

}

if (s[0] == "\*")

{

k \*= Convert.ToInt32(s[1]);

sm \*= Convert.ToInt32(s[1]);

}

if (s[0] =="-")

{

if (s[1] == "x") k -= 1;

else sm -= Convert.ToInt32(s[1]);

}

}

int r = int.Parse(Console.ReadLine());

sm = r - sm;

int ans = sm / k;

Console.WriteLine($"Выходные данные:{ans}");

}}}

На рисунках 8 – 10 представлены примеры работы программы.

Изображение выглядит как текст, снимок экрана, Шрифт

Автоматически созданное описание

Рисунок – Первый пример работы программы

Изображение выглядит как текст, снимок экрана, Шрифт

Автоматически созданное описание*Изображение выглядит как текст, снимок экрана, Шрифт

Автоматически созданное описание*

Рисунок – Третий пример работы программы

Рисунок – Второй пример работы программы

Постановка задачи 4

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*

Ход решения задачи 4

Код программы представлен ниже.

using System;

using System.Collections.Generic;

using System.Linq;

using System.Text;

using System.Threading.Tasks;

namespace ConsoleApp6

{

internal class Program

{

static void Main(string[] args)

{

Console.WriteLine("Входные данные");

int n = int.Parse(Console.ReadLine());

int number = -1;

double minPrice = int.MaxValue;

for (int i = 0; i < n; i++)

{

var s = Console.ReadLine().Replace('.',',').Split();

int x1 = int.Parse(s[0]);

int y1 = int.Parse(s[1]);

int z1 = int.Parse(s[2]);

int x2 = int.Parse(s[3]);

int y2 = int.Parse(s[4]);

int z2 = int.Parse(s[5]);

double milkPrice;

float c1 = float.Parse(s[6]);

float c2 = float.Parse(s[7]);

int s1 = 2 \* (x1 \* y1 + x1 \* z1 + y1 \* z1);

int s2 = 2 \* (x2 \* y2 + x2 \* z2 + y2 \* z2);

int v1 = x1 \* y1 \* z1;

int v2 = x2 \* y2 \* z2;

milkPrice = 1000\*(c2 - c1 \* s2 / s1) / (v2 - v1\*s2/s1);

if (milkPrice < minPrice)

{

minPrice = milkPrice;

number = i + 1;

}

else if (milkPrice == minPrice) number = Math.Min(number, i + 1);

}

Console.WriteLine($"Выходные данные {number} {minPrice:0.00}");

}}}

На рисунках 11 – 13 представлены примеры работы программы.

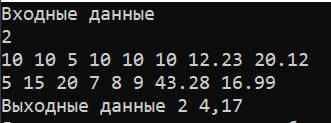


Рисунок – Первый пример работы программы

Изображение выглядит как текст, Шрифт, снимок экрана, типография

Автоматически созданное описаниеИзображение выглядит как текст, Шрифт, снимок экрана, число

Автоматически созданное описание

Рисунок – Третий пример работы программы

Рисунок – Второй пример работы программы

Для решения задачи была использована формула объема прямоугольного параллелепипеда:

*.*

Также была применена формула площади боковой поверхности прямоугольного параллелепипеда:

*.*

Заключение

В ходе выполнения расчетно-графической работы были изучены основы языка программирования C#. Были изучены ввод данных с консоли и вывод данных. Был рассмотрен способ вывода данных при помощи интерполяции. Была изучена инициализация переменных. Были рассмотрены различные типы переменных: int, uint, float, double, string, а также применены возможности преобразования типов данных. Кроме того, были изучены возможности конструкции if – else и цикла for.

Для решения задач были использованы некоторые функции класса Math (например, метод Log). Были рассмотрены некоторые методы строк. Были применены возможности создания собственных методов, рассмотрены способы применения оператора return. Также для решения задач использовалась конструкция switch.

Для выполнения работы также были применены массивы.

Список используемой литературы

1. METANIT.COM. Сайт о программировании – <https://metanit.com/> (дата обращения 23.11.2024).
2. Microsoft Learn – <https://learn.microsoft.com/ru-ru/dotnet/csharp/> (дата обращения 23.11.2024).
3. InterestPrograms.RU – <https://www.interestprograms.ru/article-history-csharp> (дата обращения 23.11.2024).
4. ci-sharp.ru – <https://ci-sharp.ru/> (дата обращения 23.11.2024).
5. Хабр – <https://habr.com/ru/articles/313694/> (дата обращения 23.11.2024).