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–№ 30 – ПЗ |

|  |  |  |  |  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- |
|  |  |  | **Студента** | | Шушаковой Виктории Александровна | | | | | |
|  |  |  |  | | 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 2](#_Toc185772599)

[Задача 1 3](#_Toc185772600)

[Постановка задачи 1 3](#_Toc185772601)

[Ход решения задачи 1 4](#_Toc185772602)

[Задача 2 8](#_Toc185772603)

[Постановка задачи 2 8](#_Toc185772604)

[Ход решения задачи 2 9](#_Toc185772605)

[Задача 3 12](#_Toc185772606)

[Постановка задачи 3 12](#_Toc185772607)

[Ход решения задачи 3 13](#_Toc185772608)

[Задача 4 16](#_Toc185772609)

[Постановка задачи 4 16](#_Toc185772610)

[Ход решения задачи 4 17](#_Toc185772611)

[Заключение 23](#_Toc185772612)

[Литература 24](#_Toc185772613)

# **Введение**

C# ̶ это современный, инновационный и открытый объектно-ориентированный язык программирования, который занимает одно из первых мест по популярности на GitHub. На сегодняшний день C# стал кросс-платформенным, что позволяет разработчикам создавать приложения не только для Windows, но и для iOS и Linux, обеспечивая таким образом широкие возможности для программирования.

Этот язык широко используется в бэкенд-разработке веб-приложений, включая работу с базами данных, сетевые взаимодействия и реализацию бизнес-логики. Разработчики, владеющие C#, востребованы в таких областях, как банковский сектор, а также в небольших компаниях и стартапах, включая мобильную разработку.

Кроме того, C# является отличным выбором для тех, кто планирует заниматься разработкой видеоигр и VR-приложений с использованием популярного игрового движка Unity. Он также может быть полезен для работы с алгоритмами машинного обучения, открывая новые горизонты для инновационных проектов.

Одним из главных достоинств языка C# является его тесная связь с платформой .NET, что предоставит разработчикам возможность использовать обширную коллекцию уже готовых инструментов и функций.

В рамках данной работы будут рассмотрены четыре задачи, выполненные с применением языка C#. Эти задачи наглядно продемонстрируют многофункциональность языка, а также ключевые возможности, которые делают его одним из лучших и наиболее предпочтительных инструментов в сфере программирования.

# **Задача 1**

# **Постановка задачи 1**

*На месте разрушенного дома, от которого уцелела одна стена, необходимо построить новый дом с основанием в виде прямоугольника размером X х Y м. Длина уцелевшей стены - L м. При постройке дома следует учитывать, что:*

* *ремонт погонного метра уцелевшей стены, для использования ее в новом доме, обходится в C1 руб.;*
* *разбор погонного метра уцелевшей стены обходится в С2 руб.;*
* *строительство погонного метра из материала, полученного при разборе стены, обходится в С3 руб.;*
* *строительство погонного метра из нового материала обходится в С4 руб.;*
* *стоимость погонного метра нового материала составляет С5 руб.;*
* *вывоз на свалку погонного метра материала, образованного при разборе стены, обходится в С6 руб.;*
* *требуется определить минимальную сумму (в руб.), которую необходимо потратить при строительстве дома на постройку стен.*

*Примечание:*

* *после строительства на стройке не должно остаться мусора в виде неиспользуемого куска старой стены и каких либо неиспользуемых материалов.;*
* *считать стены абсолютно тонкими.*

*Входной файл содержит строку, состоящую из девяти целых чисел X, Y, L, С1, C2, С3, С4, С5, С6, разделенных пробелами (1 ≤ X, Y, L, С1, С2, С3, С4, С5, С6 ≤ 100).*

*Выходной файл должен одно целое число - минимальную сумму (в руб.), которую необходимо потратить на строительство стен нового дома.*

*Пример:*

*Ввод: 3 4 5 10 4 7 6 6 3*

*Вывод: 159*

# **Ход решения задачи 1**

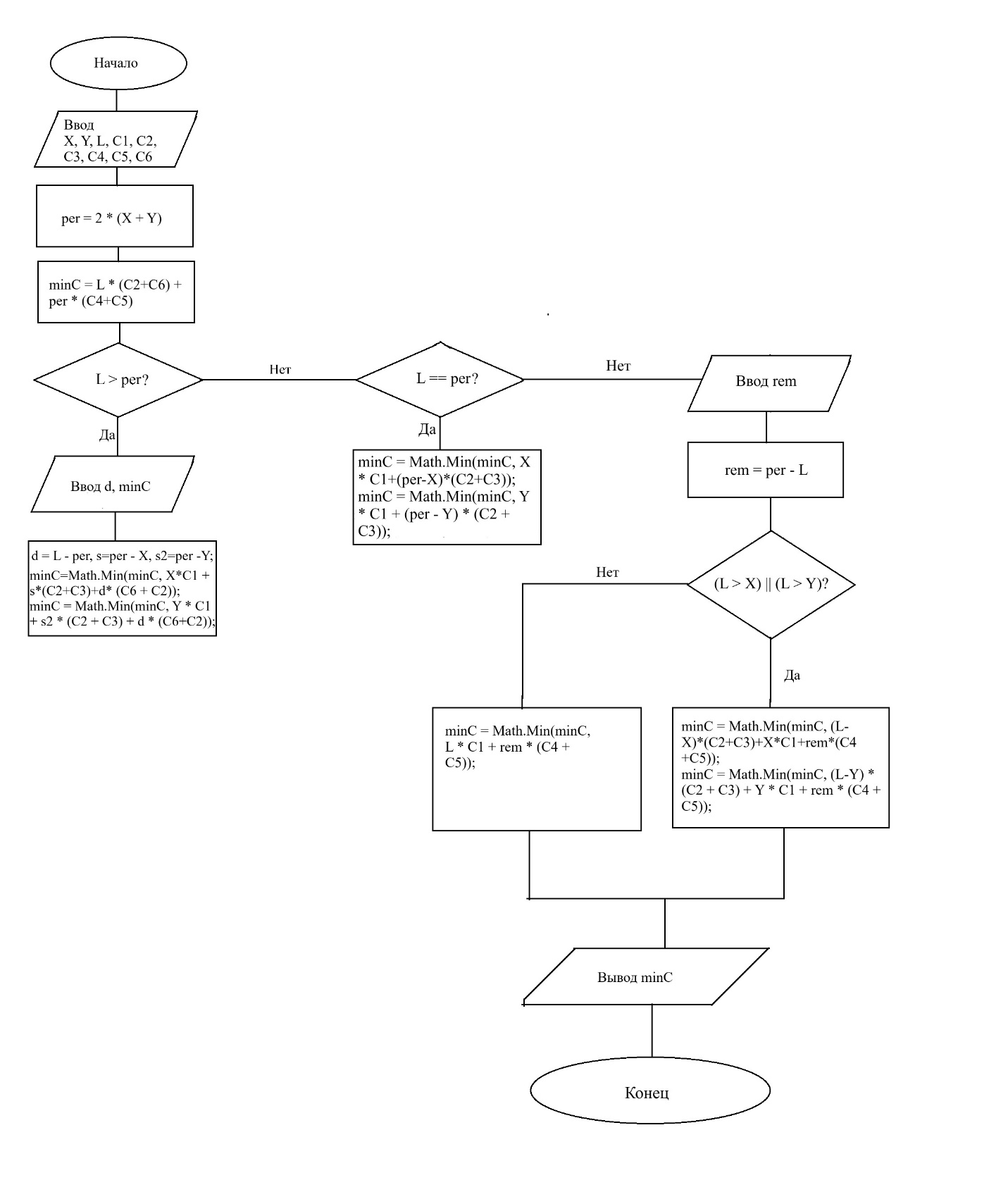
****

Рисунок 1 – Блок-схема алгоритма задачи 1

Алгоритм решает задачу по вычислению минимальной стоимости работ по ремонту и строительству стены с учётом различных факторов. Программа получает от пользователя размеры основания стены, длину уцелевшей части стены и стоимости различных этапов (ремонт, разборка, строительство, вывоз материала).

На основе введённых данных вычисляется периметр стены. Программа вычисляет начальную стоимость, предполагая, что вся стена будет разобрана, вывезена и построена заново из новых материалов. Если длина уцелевшей стены больше периметра, программа проверяет несколько вариантов ремонта и строительства, выбирая минимальную стоимость. Если длина равна периметру, она рассматривает два варианта и снова выбирает наименьшую стоимость. Если длина уцелевшей стены меньше периметра, программа также анализирует варианты, учитывая, какие части стены можно отремонтировать, а какие нужно строить заново. После анализа всех вариантов программа выводит минимальную возможную стоимость.

using System;

class Programm

{

static void Main()

{

Console.WriteLine("Введите размер основания X (м): ");

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

Console.WriteLine("Введите размер основания Y (м): ");

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

Console.WriteLine("Введите длину уцелевшей стены L (м): ");

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

Console.WriteLine("Введите стоимость ремонта 1 метра старой стены (руб.): ");

int C1 = Convert.ToInt32(Console.ReadLine());

Console.WriteLine("Введите стоимость разборки 1 метра старой стены (руб.): ");

int C2 = Convert.ToInt32(Console.ReadLine());

Console.WriteLine("Введите стоимость строительства 1 метра новой стены из старого материала (руб.): ");

int C3 = Convert.ToInt32(Console.ReadLine());

Console.WriteLine("Введите стоимость строительства 1 метра новой стены из нового материала (руб.): ");

int C4 = Convert.ToInt32(Console.ReadLine());

Console.WriteLine("Введите стоимость 1 метра новой стены из нового материала (руб.): ");

int C5 = Convert.ToInt32(Console.ReadLine());

Console.WriteLine("Введите стоимость вывоза 1 метра старого материала (руб.): ");

int C6 = Convert.ToInt32(Console.ReadLine());

int per = 2 \* (X + Y);

int minC = L \* (C2 + C6) + per \* (C4 + C5);

if (L > per)

{

int d = L - per, s = per - X, s2 = per - Y;

minC = Math.Min(minC, X \* C1 + s \* (C2 + C3) + d \* (C6 + C2));

minC = Math.Min(minC, Y \* C1 + s2 \* (C2 + C3) + d \* (C6 + C2));

}

else if (L == per)

{

minC = Math.Min(minC, X \* C1 + (per - X) \* (C2 + C3));

minC = Math.Min(minC, Y \* C1 + (per - Y) \* (C2 + C3));

}

else

{

int rem = per - L;

if ((L > X) || (L > Y))

{

minC = Math.Min(minC, (L - X) \* (C2 + C3) + X \* C1 + rem \* (C4 + C5));

minC = Math.Min(minC, (L - Y) \* (C2 + C3) + Y \* C1 + rem \* (C4 + C5));

}

else

{

minC = Math.Min(minC, L \* C1 + rem \* (C4 + C5));

}

}

Console.WriteLine(minC);

}

}

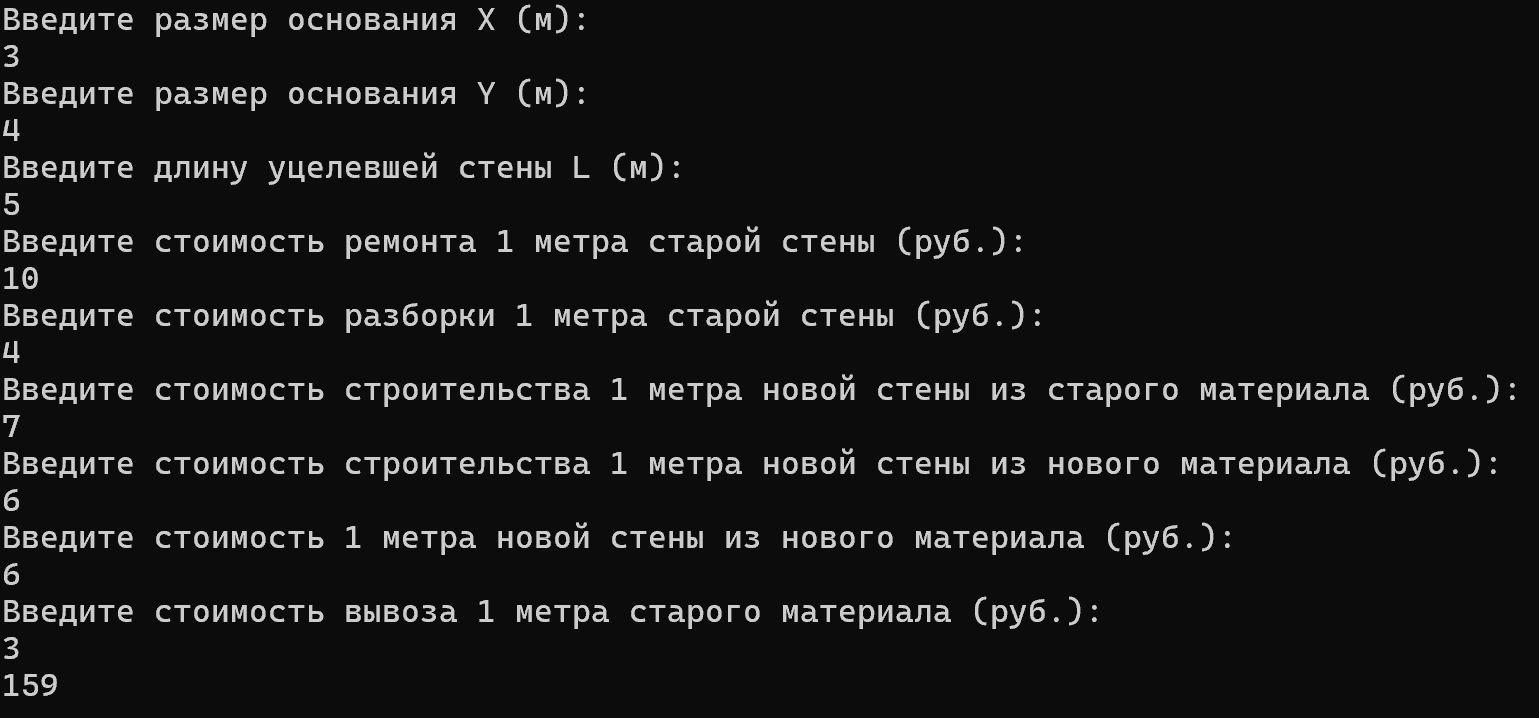


Рисунок 2 – Первый тест задачи 1

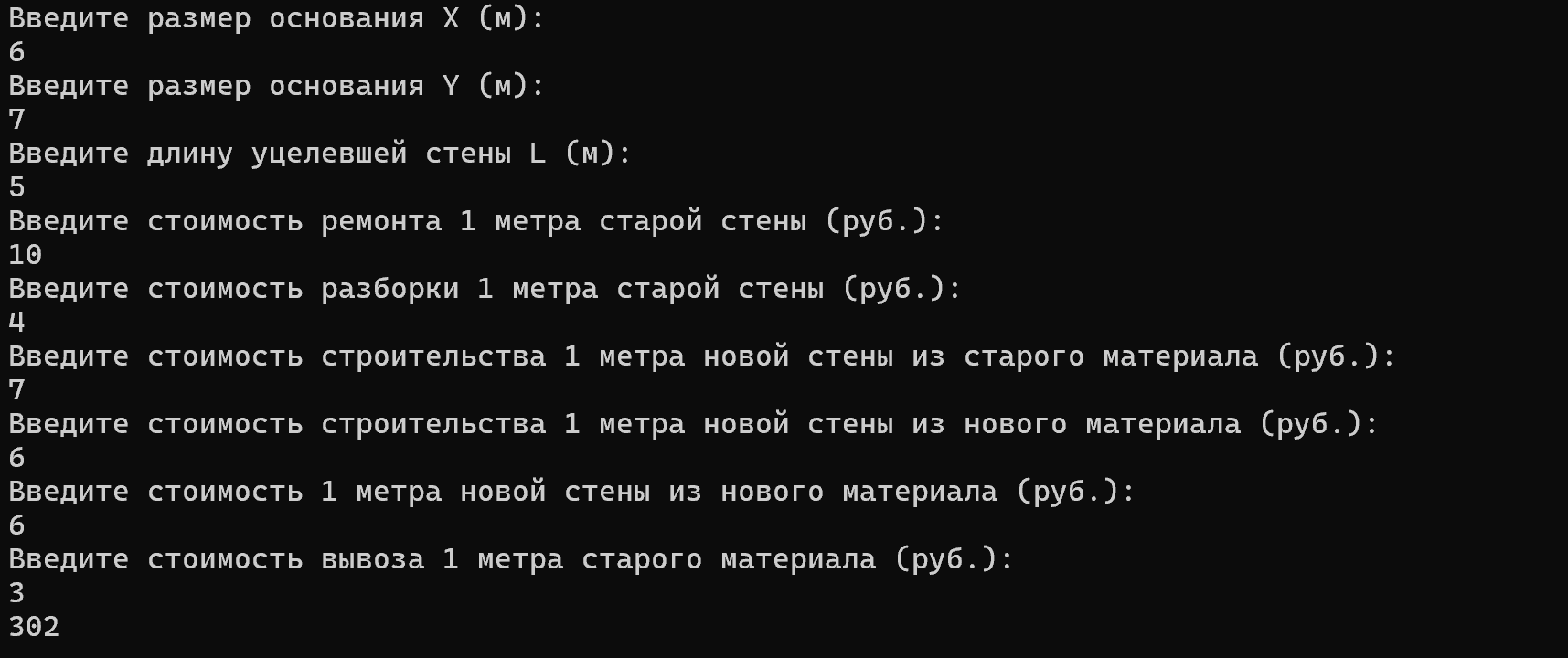


Рисунок 3 – Второй тест задачи 1

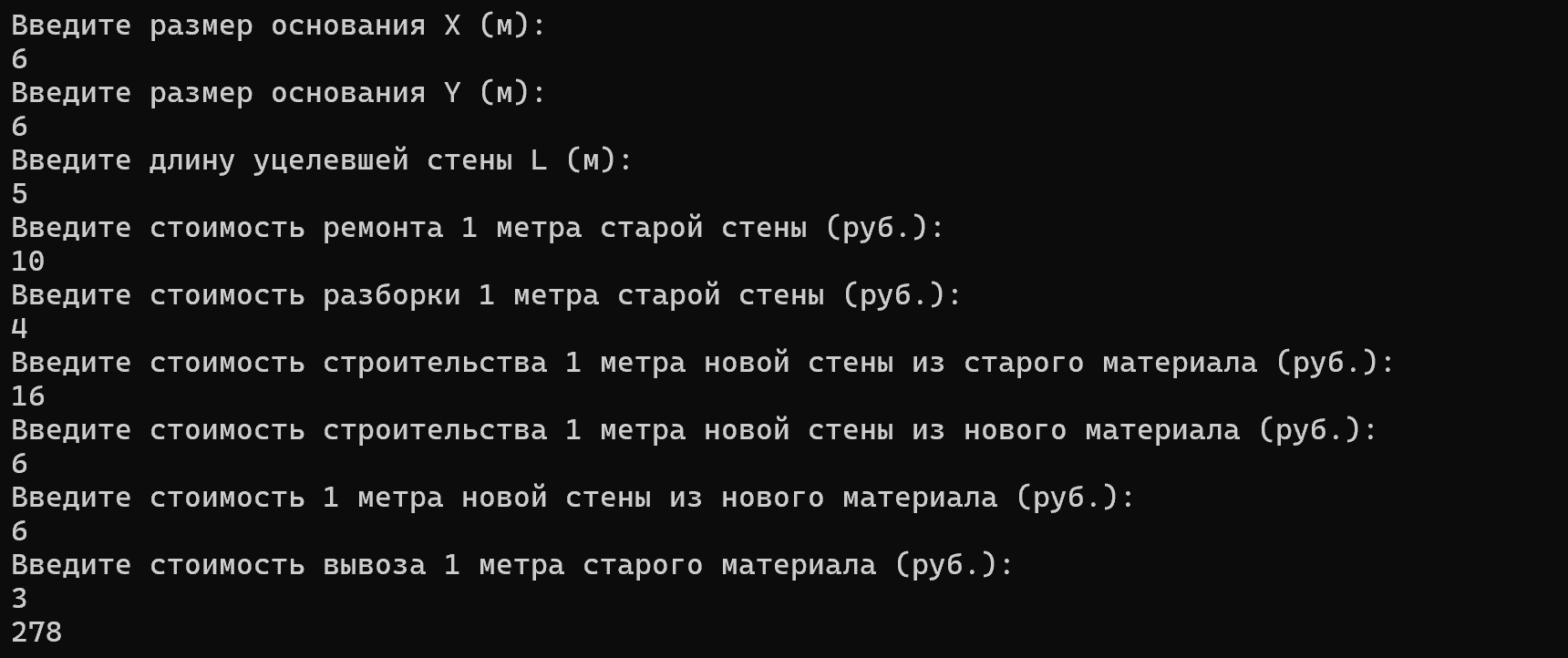


Рисунок 4 – Третий тест задачи 1

# **Задача 2**

# **Постановка задачи 2**

*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 2**

Этот алгоритм решает задачу нахождения минимальной стоимости упаковки молока, вычисляемой на основе данных о двух разных вариантах, сравнивая их по ряду характеристик. Программа сначала считывает количество строк (Fline), которые содержат информацию о разных упаковках. Для каждой строки считываются параметры. Эти данные используются для вычислений, чтобы найти цену для каждой упаковки. Алгоритм находит минимальную цену среди всех вариантов и выводит номер упаковки, которая имеет эту минимальную цену, а также саму цену, округленную до двух знаков. Программа выводит номер упаковки с минимальной ценой и саму минимальную цену, округленную до двух знаков.

using System;

using System.Globalization;

namespace ConsoleApp1

{

internal class Program

{

static void Main(string[] args)

{

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

double res = 10000;

int cnt = 0;

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

{

string line = Console.ReadLine();

string[] varArr = line.Split(' ');

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

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

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

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

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

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

double c1 = double.Parse(varArr[6], CultureInfo.InvariantCulture);

double c2 = double.Parse(varArr[7], CultureInfo.InvariantCulture);

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

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

double v1 = x1 \* y1 \* z1;

double v2 = x2 \* y2 \* z2;

double price = (-c1 + (s1 \* c2) / s2) / (-((v1 - s1) / 1000) + (s1 \* ((v2 - s2) / 1000)) / s2);

if (price < res)

{

res = Math.Min(res, price);

cnt = i + 1;

}

}

Console.WriteLine($"{cnt} {Math.Round(res, 2)}");

}

}

}

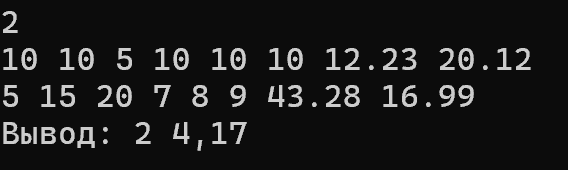


Рисунок 5 – Первый тест задачи 2

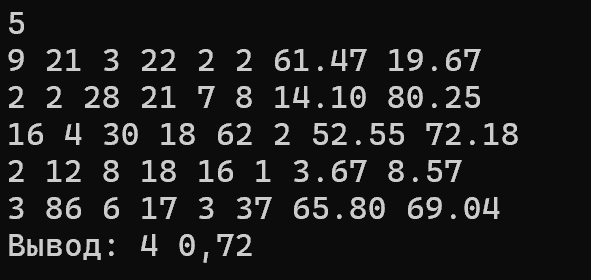


Рисунок 6 – Второй тест задачи 2

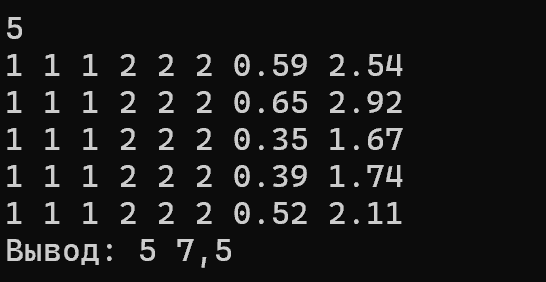


Рисунок 7 – Третий тест задачи 2

# **Задача 3**

# **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. 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*

# **Ход решения задачи 3**

Этот код решает задачу подсчета количества таких троек чисел (i, k, z), которые удовлетворяют определенным математическим условиям, и выводит это количество в консоль.

using System;

using System.Security.Cryptography.X509Certificates;

class Program()

{

static void Main()

{

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

int count=0;

for(int i =1; i<=MaxN; i++)

{

for (int k =i+1; k<=i\*2; k++)

{

for(int z=1; i\*hpo(2,z)-k\*(hpo(2,z)-1)>=0; z++)

{

if(i\*hpo(2,z)-k\*(hpo(2,z)-1)==0)

{

count++;

break;

}

}

}

}Console.WriteLine(count);

}

static int hpo(int a, int b)

{

int result= 1;

for( int g=0;g<b;g++)

{

result\*=a;

}

return result;

}

}

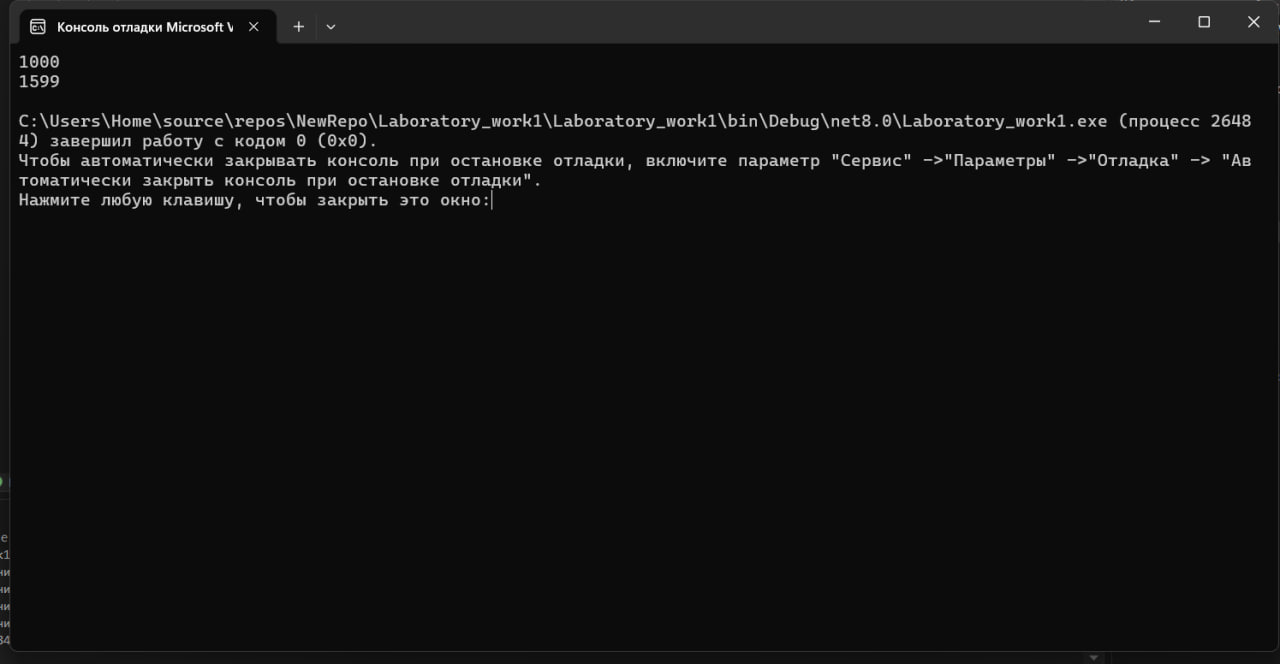


Рисунок 8 – Первый тест задачи 3

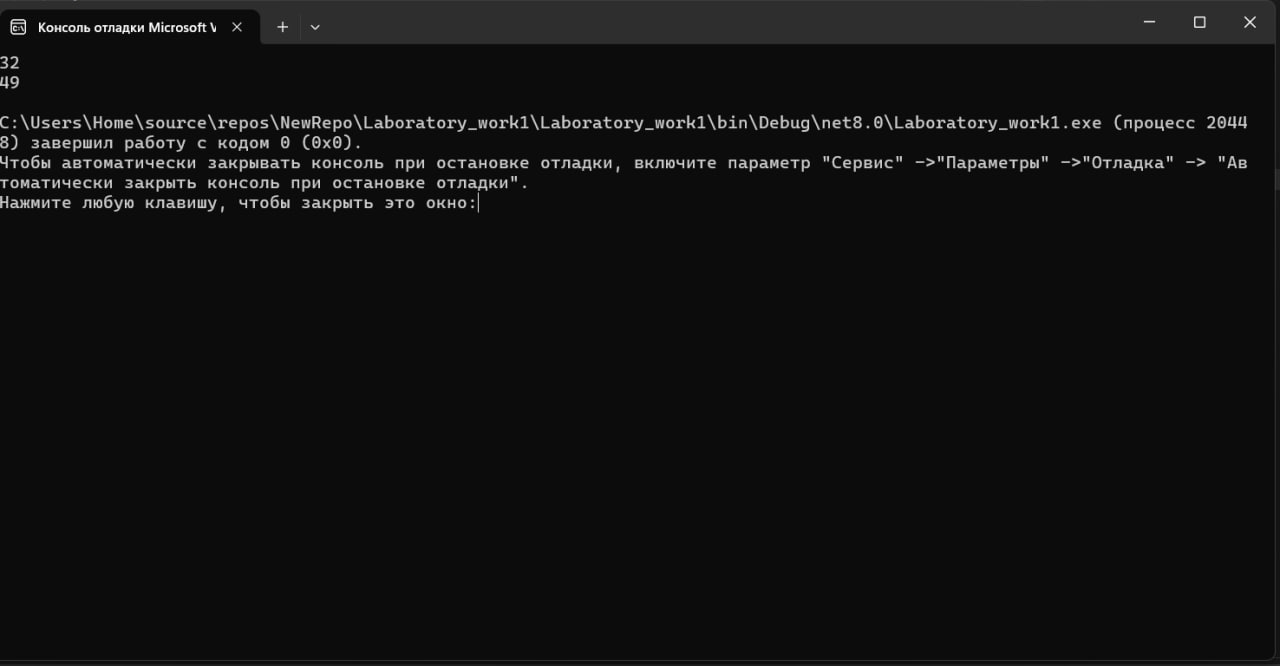


Рисунок 9 – Второй тест задачи 4

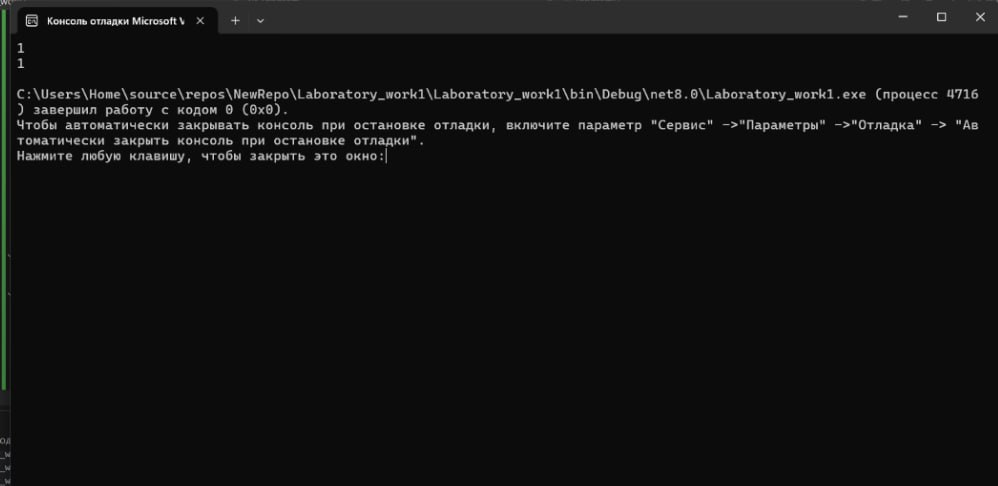


Рисунок 10 – Третий тест задачи 5

# **Задача 4**

# **Постановка задачи 4**

*В компании “ХХХ” каждый служащий имеет уникальный идентификационный номер и может иметь уникальное имя. Между служащими установлены отношения типа “начальник – подчиненный”.*

*Требуется определить всех (т.е. не только непосредственных) подчиненных для заданного начальника. Начальник может быть задан как по идентификационному номеру, так и по уникальному имени.*

*Примечания:*

* *в именах нет различий между строчными и прописными буквами;*
* *служащий не может иметь более одного начальника;*
* *отношения “начальник – подчиненный” не образуют циклов.*

*Входной файл состоит из двух блоков:*

* *в первом блоке перечисляются отношения “начальник – подчиненный” (0 < количество отношений ≤ 5000). Каждое отношение задается двумя строками. В первой строке содержатся данные о начальнике, во второй – о его подчиненном. Каждая строка имеет формат XXXX\_NAME, где XXXX – уникальный идентификационный номер служащего, всегда состоящий из 4 цифр; NAME – уникальное имя служащего, которое может отсутствовать в строке, даже если оно присутствует в других строках; “\_” – пробел. В случае, если имя отсутствует, пробел не ставится. Длина XXXX\_NAME не превышает 30 символов. Между любыми двумя сотрудниками может быть только одно отношение “начальник – подчиненный”.*

*Блок заканчивается словом “END” (без кавычек) в очередной строке;*

* *второй блок содержит одну строку – уникальный номер или уникальное имя начальника, подчиненных которого необходимо вывести. Данный сотрудник обязательно присутствует хотя бы в одном отношении “начальник – подчиненный”.*

*Выходной файл должен содержать найденных подчиненных, выведенных в соответствии со следующими правилами:*

* *подчиненных выводить по одному в строке, указывая для каждого его уникальный идентификационный номер и имя, разделенные пробелом;*
* *если имя подчиненного неизвестно, то вместо него выводить “Unknown Name” (без кавычек);*
* *подчиненные должны быть выведены в порядке возрастания уникального номера;*
* *в случае, если служащий не имеет подчиненных, выводить “NO” (без кавычек).*

*Пример 1:*

*Input.txt*

*0001 Hanh Tran*

*0005 Stanley Wendler*

*0001 Hanh Tran*

*0003*

*0003 Ben Tu*

*0004 Steven Emery*

*0003*

*0007*

*END*

*Hanh Tran*

*Output.txt*

*0003 Ben Tu*

*0004 Steven Emery*

*0005 Stanley Wendler*

*0007 Unknown Name*

# **Ход решения задачи 4**

Этот алгоритм обрабатывает информацию о сотрудниках и их руководителях, введенную с клавиатуры.

Цель программы ̶ найти всех сотрудников, которые подчиняются указанному руководителю, и вывести их в отсортированном списке. Входные данные представляют собой строки, где каждая пара строк состоит из ID или имени руководителя и имени сотрудника.

using System;

using System.Collections.Generic;

class TripleXXX

{

static void Main()

{

List<string> employees = new List<string>();

List<string> lines = new List<string>();

string line;

while ((line = Console.ReadLine()) != "END")

{

lines.Add(line);

}

line = Console.ReadLine();

int requiredBoss;

bool isNumeric = Int32.TryParse(line, out requiredBoss);

int endLine = lines.Count;

if (isNumeric)

{

for (int i = 0; i < endLine - 1; i += 2)

{

string employee = lines[i + 1];

string boss = lines[i];

string[] splittedInfoBoss = boss.Split(" ");

if (Convert.ToInt32(splittedInfoBoss[0]) == requiredBoss)

{

if (employee.Split(" ").Length == 1)

{

employee += " UNKNOWN NAME";

}

employees.Add(employee);

}

else

{

for (int j = 0; j < employees.Count; j++)

{

string em = employees[j];

if (em.Contains(boss) || Convert.ToInt32(em.Split(" ")[0]) == Convert.ToInt32(boss.Split(" ")[0]))

{

if (employee.Split(" ").Length == 1)

{

employee += " UNKNOWN NAME";

}

employees.Add(employee);

break;

}

}

}

}

}

else

{

string requiredBossName = line;

for (int i = 0; i < endLine - 1; i += 2)

{

string employee = lines[i + 1];

string boss = "";

string[] bossSplit = lines[i].Split(" ");

int bossId = Convert.ToInt32(bossSplit[0]);

if (bossSplit.Length > 1)

{

for (int j = 1; j < bossSplit.Length; j++)

{

boss += bossSplit[j] + " ";

}

boss = boss.Trim();

if (boss == requiredBossName)

{

if (employee.Split(" ").Length == 1)

{

employee += " UNKNOWN NAME";

}

employees.Add(employee);

}

else

{

for (int j = 0; j < employees.Count; j++)

{

string em = employees[j];

if (em.Contains(boss) || Convert.ToInt32(em.Split(" ")[0]) == bossId)

{

employees.Add(employee);

}

}

}

}

else

{

for (int j = 0; j < employees.Count; j++)

{

string em = employees[j];

if (em.Contains(boss) || Convert.ToInt32(em.Split(" ")[0]) == bossId)

{

if (employee.Split(" ").Length == 1)

{

employee += " UNKNOWN NAME";

}

employees.Add(employee);

break;

}

}

}

}

}

for (int i = 0; i < employees.Count; i++)

{

if (employees[i].Contains("UNKNOWN NAME"))

{

string id = employees[i].Split(" ")[0];

for (int j = 0; j < lines.Count; j++)

{

if (lines[j].Contains(id) && lines[j].Split(" ").Length > 1)

{

employees[i] = lines[j];

}

}

}

}

employees.Sort((a, b) => a.CompareTo(b));

employees.ForEach(em => Console.WriteLine($"Ответ: {em}"));

}

}

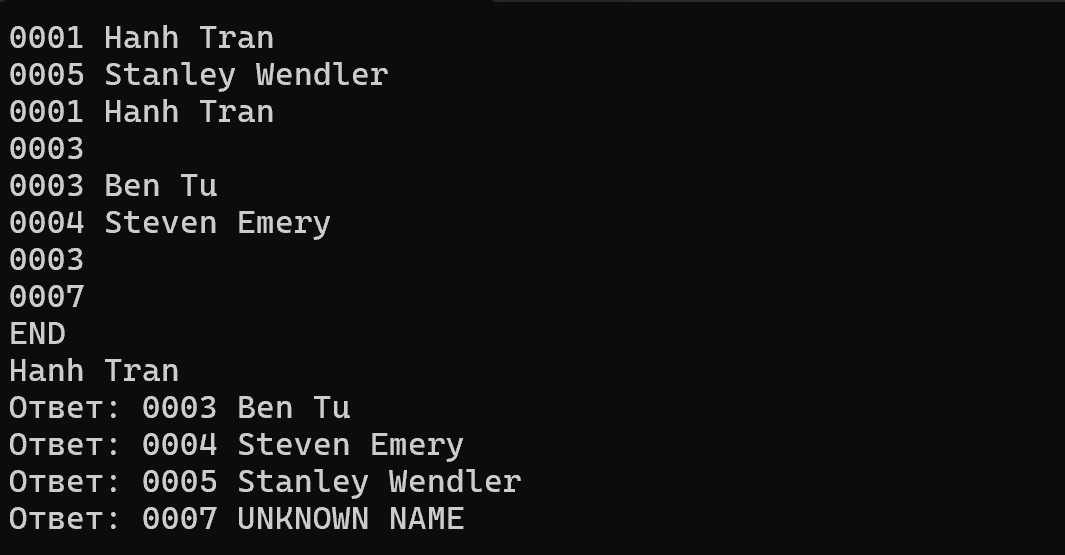


Рисунок 12 – Первый тест задачи 4

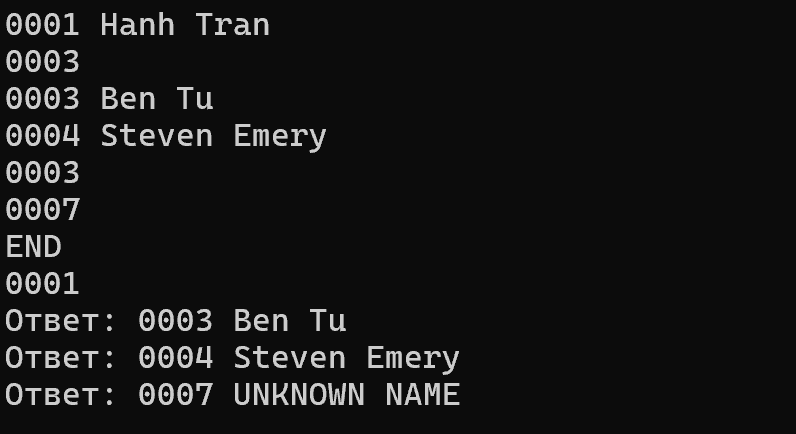


Рисунок 13 – Второй тест задачи 4

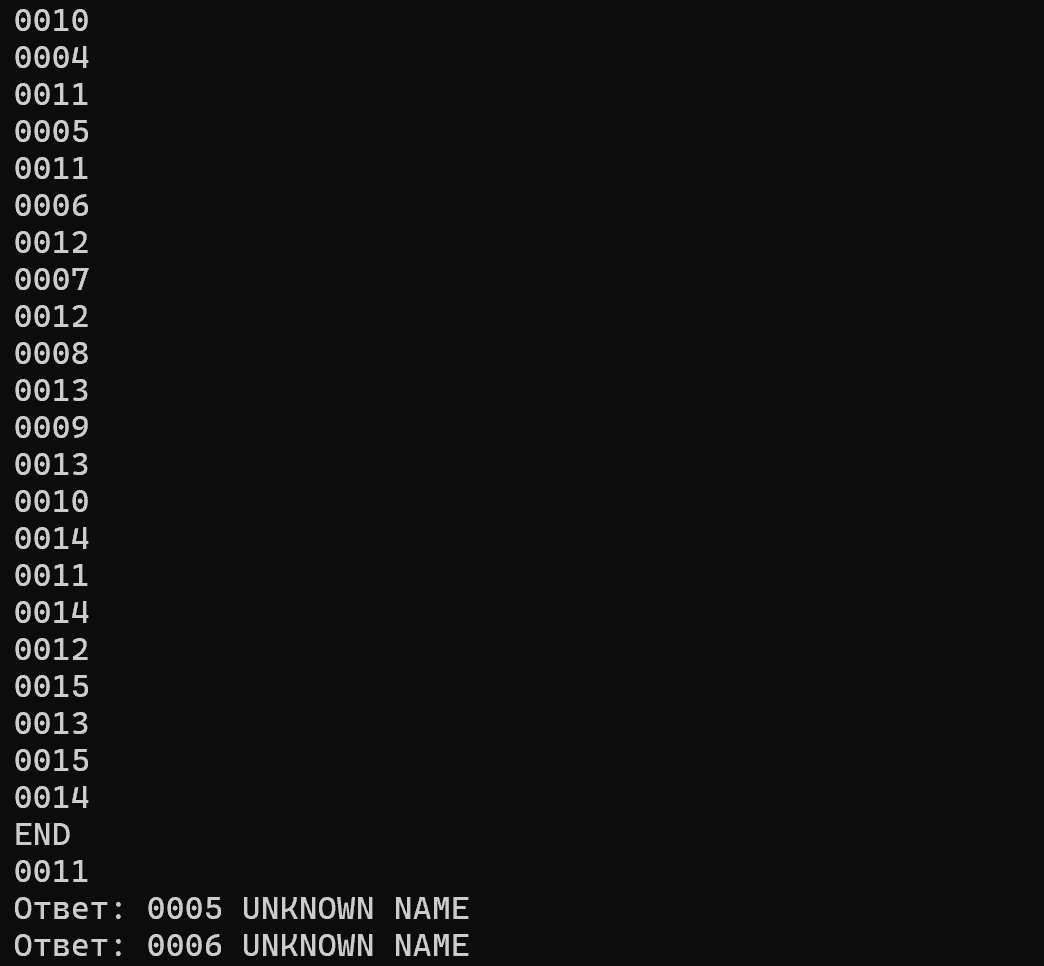


Рисунок 14 – Третий тест задачи 4

# **Заключение**

В ходе выполнения расчетно-графической работы, посвященной олимпиадным заданиям на языке программирования C#, были подробно изучены и применены на практике основные принципы работы с переменными, условиями, циклами, алгоритмами, а также базовыми структурами данных. Эта работа позволила глубже понять и усвоить ключевые аспекты программирования, такие как управление потоком исполнения программы, эффективное использование условий и циклов, а также оптимизацию алгоритмов для решения различных задач.

Кроме того, работа способствовала значительному укреплению навыков программирования на C#, позволив лучше ориентироваться в синтаксисе и особенностях языка, а также повысить общий уровень алгоритмической подготовки. Применение теоретических знаний на практике и решение реальных задач дало возможность развить логическое и аналитическое мышление, что будет полезно как для участия в олимпиадах и конкурсах, так и для решения сложных практических задач в рамках профессиональной деятельности. Этот опыт поможет более уверенно и эффективно подходить к разработке программных решений в различных областях.

# **Литература**

1. <https://skillbox.ru/media/code/yazyk-s-gde-ispolzuyut-chto-pishut-kak-poyavilsya-i-chem-khorosh/.2024> (дата обращения 23.11.2024).
2. Васильев А.Н. Программирование на C# для начинающих – Москва, 2019 – 860 с. (дата обращения 18.11.2024).
3. Троелсен Э., Джапура К. Язык программирования C# 10 и платформа .NET 6. – Питер, Санкт-Петербург, 2022. – 960 с.
4. <https://learn.microsoft.com/ru-ru/dotnet/csharp/> (дата обращения 10.11.2024)
5. https://metanit.com/sharp/tutorial/1.1.php (дата обращения 9.12.2024)