МИНОБРНАУКИ РОССИИ

федеральное государственное бюджетное образовательное учреждение

высшего образования

«Кузбасский государственный технический университет   
имени Т.Ф. Горбачева»

ИНСТИТУТ ПРОФЕССИОНАЛЬНОГО ОБРАЗОВАНИЯ

ОТЧЕТ ПО УЧЕБНОЙ ПРАКТИКЕ

ПМ 02 «Осуществление интеграции программных модулей»

*(шифр и название профессионального модуля)*

Выполнил:

студент группы ИСт-193

Зайцев Сергей Алексеевич

Руководитель практики:

преподаватель

Сыркин И.С.

Оценка*\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_*

«\_\_\_»\_\_\_\_\_\_\_\_\_\_\_\_20\_\_\_ г.

\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_ *Сыркин И.С.*

*Подпись (расшифровка подписи)*

Кемерово 2021

СОДЕРЖАНИЕ

[ВВЕДЕНИЕ 4](#_Toc74209637)

[1 АНАЛИЗ ПРЕДМЕТНОЙ ОБЛАСТИ. РАЗРАБОТКА И ОФОРМЛЕНИЕ ТЕХНИЧЕСКОГО ЗАДАНИЯ 6](#_Toc74209638)

[1.1 Задание 6](#_Toc74209639)

[1.2 Результат выполнения 6](#_Toc74209640)

[1.3 Задание 6](#_Toc74209641)

[1.4 Результат выполнения 6](#_Toc74209642)

[2 ПОСТРОЕНИЕ АРХИТЕКТУРЫ ПРОГРАММНОГО СРЕДСТВА, ИЗУЧЕНИЕ РАБОТЫ В СИСТЕМЕ КОНТРОЛЯ ВЕРСИЙ 7](#_Toc74209643)

[2.1 Задание 7](#_Toc74209644)

[2.2 Результат выполнения 7](#_Toc74209645)

[2.3 Задание 7](#_Toc74209646)

[2.4 Результат выполнения 7](#_Toc74209647)

[3 Построение архитектуры программного средства 8](#_Toc74209648)

[3.1 Задание 8](#_Toc74209649)

[3.2 Результат выполнения 8](#_Toc74209650)

[3.3 Задание 8](#_Toc74209651)

[3.4 Результат выполнения 8](#_Toc74209652)

[4 Изучение работы в системе контроля версий 9](#_Toc74209653)

[4.1 Задание 9](#_Toc74209654)

[4.2 Результат выполнения 9](#_Toc74209655)

[4.3 Задание 9](#_Toc74209656)

[4.4 Результат выполнения 9](#_Toc74209657)

[5 Описание и анализ требований 10](#_Toc74209658)

[5.1 Задание 10](#_Toc74209659)

[5.2 Результат выполнения 10](#_Toc74209660)

[5.3 Задание 10](#_Toc74209661)

[5.4 Результат выполнения 10](#_Toc74209662)

[6 Разработка тестового сценария 11](#_Toc74209663)

[6.1 Задание 11](#_Toc74209664)

[6.2 Результат выполнения 11](#_Toc74209665)

[6.3 Задание 11](#_Toc74209666)

[6.4 Результат выполнения 11](#_Toc74209667)

[7 Инспекция программного кода на предмет соответствия стандартам кодирования 12](#_Toc74209668)

[7.1 Задание 12](#_Toc74209669)

[7.2 Результат выполнения 12](#_Toc74209670)

[8 Основы математического моделирования информационных систем 14](#_Toc74209671)

[8.1 Задание 14](#_Toc74209672)

[8.2 Результат выполнения 14](#_Toc74209673)

[8.3 Задание 14](#_Toc74209674)

[8.4 Результат выполнения 14](#_Toc74209675)

[ЗАКЛЮЧЕНИЕ 15](#_Toc74209676)

[СПИСОК ЛИТЕРАТУРЫ 16](#_Toc74209677)

# ВВЕДЕНИЕ

Программа учебной практики является частью программы подготовки специалистов среднего звена в соответствии с ФГОС по специальности СПО 09.02.07 «Информационные системы и программирование» в части освоения основного вида профессиональной деятельности (ВПД). Данная практика направлена на формирование следующих компетенций:

ОК 01 – Выбирать способы решения задач профессиональной деятельности, применительно к различным контекстам.

ОК 02 – Осуществлять поиск, анализ и интерпретацию информации, необходимой для выполнения задач профессиональной деятельности.

ОК 03 – Планировать и реализовывать собственное профессиональное и личностное развитие.

ОК 04 – Работать в коллективе и команде, эффективно взаимодействовать с коллегами, руководством, клиентами.

ОК 05 – Осуществлять устную и письменную коммуникацию на государственном языке с учетом особенностей социального и культурного контекста.

ОК 06 – Проявлять гражданско-патриотическую позицию, демонстрировать осознанное поведение на основе традиционных общечеловеческих ценностей, применять стандарты антикоррупционного поведения.

ОК 07 – Содействовать сохранению окружающей среды, ресурсосбережению, эффективно действовать в черезвычайных ситуациях.

ОК 08 – Использовать средства физической культуры для сохранения и укрепления здоровья в процессе профессиональной деятельности и поддержания необходимого уровня физической подготовленности.

ОК 09 – Использовать информационные технологии в профессиональной деятельности.

ОК 10 – Пользоваться профессиональной документацией на государственном и иностранном языках.

ОК 11 – Использовать знания по финансовой грамотности, планировать предпринимательскую деятельность в профессиональной сфере.

ПК 2.1 – Разрабатывать требования к программным модулям на основе анализа проектной и технической документации на предмет взаимодействия компонент.

ПК 2.2 – Выполнять интеграцию модулей в программном обеспечение.

ПК 2.3 – Выполнять отладку программного модуля с использованием специализированных программных средств.

ПК 2.4 – Осуществлять разработку тестовых наборов и тестовых сценариев для программного обеспечения

ПК 2.5 – Производить инспектирование компонент программного обеспечения на предмет соответствия стандартам кодирования.

# АНАЛИЗ ПРЕДМЕТНОЙ ОБЛАСТИ. РАЗРАБОТКА И ОФОРМЛЕНИЕ ТЕХНИЧЕСКОГО ЗАДАНИЯ

## Задание

Задание представлено на рисунке 1.1.

# Horse-racing Duals

Ипподром Касабланки организует новый вид скачек: парные. В парном разряде в гонке участвуют только две лошади. Чтобы гонка была интересной, необходимо попробовать выбрать две одинаковые по силе закуски. Напишите программу, которая, используя заданное количество сильных сторон, идентифицирует две самые близкие сильные стороны и показывает их разницу с помощью целого числа (≥ 0). Строка 1: Количество лошадей N В следующих N строках: сила Pi каждой лошади. Пи - целое число. Выход Разница D между двумя ближайшими сильными сторонами. D - целое число, большее или равное 0.

Рисунок 1.1

## Результат выполнения

using System;

using System.Linq;

using System.IO;

using System.Text;

using System.Collections;

using System.Collections.Generic;

class Solution

{

static void Main(string[] args)

{

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

List<int> numberOfHorses = new List<int>();

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

{

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

numberOfHorses.Add(pi);

}

List<int> orderedHorses = numberOfHorses.OrderByDescending(i => i).ToList();

int smallestStrengthDifference = int.MaxValue;

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

{

if (i == 0)

{

smallestStrengthDifference = orderedHorses[i] - orderedHorses[i + 1];

}

else if (orderedHorses[i-1] - orderedHorses[i] < smallestStrengthDifference)

{

smallestStrengthDifference = orderedHorses[i - 1] - orderedHorses[i];

}

}

Console.WriteLine(smallestStrengthDifference);

Console.ReadLine();

}

}

## Задание

Задание представлено на рисунке 1.2

# Temperatures

В этом упражнении вы должны проанализировать записи температуры, чтобы найти значение, наиболее близкое к нулю. Напишите программу, которая печатает температуру, ближайшую к 0 среди входных данных. Если два числа одинаково близки к нулю, положительное целое число следует рассматривать как наиболее близкое к нулю (например, если температуры равны -5 и 5, то отобразите 5). Ваша программа должна считывать данные со стандартного ввода и записывать результат на стандартный вывод. Вход Строка 1: N, количество температур для анализа. Строка 2: строка с N температурами, выраженными целыми числами от -273 до 5526. Выход Отображение 0 (ноль), если температура не указана. В противном случае отобразите температуру, наиболее близкую к 0.

Рисунок 1.2

## Результат выполнения

using System;

using System.Linq;

using System.IO;

using System.Text;

using System.Collections;

using System.Collections.Generic;

class Solution

{

static void Main(string[] args)

{

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

string temps = Console.ReadLine();

int[] inputTempsToAnalyze = new int[n];

if (n > 0)

{

inputTempsToAnalyze = temps.Split(' ').Select(s=> int.Parse(s)).ToArray();

}

int lowestTempOutput = 0;

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

{

if (i == 0)

{

lowestTempOutput = inputTempsToAnalyze[i];

}

else if (Math.Abs(inputTempsToAnalyze[i]) < Math.Abs(lowestTempOutput))

{

lowestTempOutput = inputTempsToAnalyze[i];

}

else if (Math.Abs(inputTempsToAnalyze[i]) == Math.Abs(lowestTempOutput) && inputTempsToAnalyze[i] > lowestTempOutput)

{

lowestTempOutput = inputTempsToAnalyze[i];

}

}

Console.WriteLine("{0}", lowestTempOutput);

}

}

# ПОСТРОЕНИЕ АРХИТЕКТУРЫ ПРОГРАММНОГО СРЕДСТВА, ИЗУЧЕНИЕ РАБОТЫ В СИСТЕМЕ КОНТРОЛЯ ВЕРСИЙ

## Задание

Задание представлено на рисунке 2.1.

Power The Thor – Episode 1

Цель: Your program must allow Thor to reach the light of power.

(Ваша программа должна позволить Тору достичь света силы.)

Рисунок 2.1

## Результат выполнения

class Player

{

static void Main(string[] args)

{

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

int lightX = int.Parse(inputs[0]);

int lightY = int.Parse(inputs[1]);

int initialTX = int.Parse(inputs[2]);

int initialTY = int.Parse(inputs[3]);

int thorX = initialTX;

int thorY = initialTY;

while (true)

{

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

string directionX = "";

string directionY = "";

if (thorX > lightX)

{

directionX = "W";

thorX--;

}

else if (thorX < lightX)

{

directionX = "E";

thorX++;

}

if (thorY > lightY)

{

directionY = "N";

thorY--;

}

else if (thorY < lightY)

{

directionY = "S";

thorY++;

}

Console.WriteLine(directionY + directionX);

}

}

}

## Задание

Задание представлено на рисунке 2.2.

(Уничтожьте горы, прежде чем ваш звездолет столкнется с одним из них. Для этого стреляйте в самую высокую гору на своем пути.

В начале каждого игрового хода слева направо указывается высота 8 гор.

К концу игрового хода вы должны стрелять по самой высокой горе, пе-реставив ее индекс (от 0 до 7).

Стрельба по горе уничтожит только ее часть, уменьшив ее высоту. Ваш корабль спускается после каждого прохода.)

Рисунок 2.2

## Результат выполнения

class Player

{

static void Main(string[] args)

{

while (true)

{

int max = 0;

int imax = 0;

for (int i = 0; i < 8; i++)

{

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

if (mountainH > max) {

max = mountainH;

imax = i;

}

}

Console.WriteLine(imax);

}

}

}

# Построение архитектуры программного средства

## Задание

Задание представлено на рисунке 3.1.

Задание: MIME Type

Вы должны написать программу, которая позволяет определять MIME-тип файла по его имени.

Рисунок 3.1

## Результат выполнения

class Solution

{

static void Main(string[] args)

{

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

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

Dictionary<string, string> mimeTypes = new Dictionary<string, string>();

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

{

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

string EXTENSION = inputs[0];

string MIMETYPE = inputs[1];

mimeTypes.Add("." + EXTENSION.ToLower(), MIMETYPE);

}

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

{

string FILENAME = Console.ReadLine();

string FILEEXT = Path.GetExtension(FILENAME).ToLower();

if (mimeTypes.ContainsKey(FILEEXT))

{

Console.WriteLine(mimeTypes[FILEEXT]);

}

else

{

Console.WriteLine("UNKNOWN");

}

}

}

}

## Задание

Задание представлено на рисунке 3.2.

Задание: ASCll Art

# # # ### # # # ### ### # ###

### # # # # # # # # # # # # # #

### ### # # ### ### # # ### # #

# # # # # # # # # # # # # # # #

# # # # # # # # # # # # # # # #

Ваша миссия - написать программу, которая может отображать строку текста в формате ASCII в стиле, который вы вводите.)

Рисунок 3.2

## Результат выполнения

class Solution

{

static void Main(string[] args)

{

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

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

string T = Console.ReadLine().ToUpper();

const int unknownCharIndex = 'Z' - 'A' + 1;

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

{

string asciiLine = Console.ReadLine();

for (int j = 0; j < T.Length; j++)

{

int charIndex = T[j] - 'A';

int letterIndex = char.IsLetter(T[j]) ? charIndex : unknownCharIn-dex;

string asciiPart = asciiLine.Substring(letterIndex \* L, L);

Console.Write(asciiPart);

}

Console.WriteLine();

}

}

}

# Изучение работы в системе контроля версий

## Задание

Задание представлено на рисунке 4.1.

Задание: Bank Robbers

Цель: банда глупых грабителей решает ограбить банк. В банке V хранилищ (индексируется от 0 до V - 1). Грабителям удалось получить у директора банка некоторую информацию: - Комбинация хранилища состоит из символов C (цифр / гласных). - Первые N символов состоят из цифр от 0 до 9. - Остальные символы состоят из гласных (A, E, I, O, U). - C и N могут быть одинаковыми или разными для разных хранилищ. Все грабители работают одновременно. Грабитель может работать в одном хранилище за раз, а в хранилище может работать только один грабитель. Грабители разбираются с различными хранилищами в порядке возрастания. Грабитель пробует комбинации со скоростью 1 комбинацию в секунду. Он пробует все возможные комбинации, то есть продолжает пробовать непроверенные комбинации даже после того, как нашел правильную комбинацию. Закончив одно хранилище, он переходит к следующему доступному хранилищу, то есть хранилищу с наименьшим индексом среди всех хранилищ, над которыми еще не работали. Ограбление окончено, когда грабители проработают все хранилища. Предположим, что переход из одного хранилища в другое не занимает много времени. Вы должны вывести общее время ограбления.

(Вы должны вывести общее время ограбления.)

Рисунок 4.1

## Результат выполнения

class Solution

{

static void Main(string[] args)

{

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

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

int[] T = new int[R];

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

{

string[] line = Console.ReadLine().Split();

int C = int.Parse(line[0]);

int N = int.Parse(line[1]);

T[0] += (int)(Math.Pow(10, N) \* Math.Pow(5, C - N));

Array.Sort(T);

}

Console.WriteLine(T[R - 1]);

}

}

## Задание

Задание представлено на рисунке 4.2.

Brick in wall

Кирпич в стене Старый добрый мастер Мейсон хочет построить стену. У него на полу лежат N кирпичей, каждый разного веса. Высота кирпичей 6,5 см. Стена строится снизу; в каждом ряду может быть не больше кирпичей, чем в ряду под ним. Кирпичи можно класть прямо сверху, если выполнено предыдущее условие. В каждом ряду есть место для максимум X кирпичей. Мастер Мейсон хочет свести к минимуму свою работу. Вспоминая старые школьные времена и уроки физики, он рассчитывает работу следующим образом. Если кирпич встраивается в L-й ряд от пола, работа, необходимая для установки этого кирпича, составляет: W = ((L-1) \* 6.5 / 100) \* г \* м, где m - вес кирпича. измеряется в килограммах, а g = 10 м / с² - это (не слишком точное значение) ускорение свободного падения. (L-1) \* 6.5 / 100 измеряет расстояние, на которое необходимо поднять кирпич, в метрах. Обратите внимание, что самая нижняя строка - первая. Задача состоит в том, чтобы рассчитать минимальную работу, которую Мастер Каменщик может построить из всех кирпичей в стену (не обязательно прямоугольную). Вход Строка 1: целое число X, обозначающее количество кирпичей в ряду. Строка 2: целое число N для количества кирпичей. Строка 3: целые веса кирпичей, разделенные пробелом. Вес указан в кг. Выход Строка 1: Минимальная работа. Напечатано с точностью до 3 знаков после запятой.

Рисунок 4.2

## Результат выполнения

class Solution

{

static void Main(string[] args)

{

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

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

int[] M = Con-sole.ReadLine().Split().Select(int.Parse).OrderByDescending(s => s).ToArray();

double W = 0;

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

{

int L = i / X;

W += L \* .65 \* M[i];

}

Console.WriteLine(Math.Round(W, 3).ToString("0.000"));

}

}

# Описание и анализ требований

## Задание

Задание представлено на рисунке 5.1.

В игре CardFight® игроки сражаются друг с другом, собирая колоду карт и используя стратегию и тактику, чтобы победить своих противников. На каждой карте указано целое число очков здоровья. Карты со слишком низким уровнем здоровья совершенно бесполезны на боевой арене. Однако карты с низким уровнем здоровья по-прежнему имеют свою ценность. Компания CardFight® предлагает услугу конвертации, которая представляет собой машину, позволяющую игрокам вставить две карты, нажать кнопку возврата и получить одну новую карту с очком здоровья, равным сумме двух исходных карт. Кажется, что у карт, производимых машиной, нет верхнего предела здоровья. Ура! Эффективно используя эту машину преобразования, вы можете обменивать свои карты, чтобы получить новые карты с таким высоким показателем здоровья, который вам нужен. Вы знаете правило бизнеса - нет ничего бесплатного. Каждый раз, когда вы получаете новую карту в автомате, вы должны платить комиссию за обслуживание, пропорциональную баллу здоровья на новой карте. Например, если вы вставите старую карту с 1 очком и другую старую карту с 2 очками, вы получите новую карту с 3 очками. CardFight® спишет с вашего счета 3 доллара. Когда у вас есть стопка карт, желающих добавить свои очки в новую карту, вы обнаружите, что порядок имеет значение. Скажем, у вас есть три карты с точками 1, 2 и 3. Вставьте 1 и 2, чтобы получить новую карту на 3 балла. Стоимость 3 доллара. Теперь у вас есть новая карта с 3 очками и ваша оригинальная карта с 3 очками. Вставьте 3 и 3, чтобы получить новую карту на 6 очков. Стоимость 6 долларов. Общая стоимость 9 долларов. Делаем это по-другому. Вставьте 2 и 3, чтобы получить новую карту на 5 очков. Стоимость 5 долларов. Теперь у вас есть новая 5-балльная карта и ваша оригинальная 1-балльная карта. Вставьте 1 и 5, чтобы получить новую карту на 6 очков. Стоимость 6 долларов. Общая стоимость ошибочно ... 11 долларов ?! Будучи умным игроком, вы должны найти лучшую стратегию, чтобы завершить обмен с наименьшими затратами. Строка 1: целое число N - количество карт для обмена. Строка 2: N положительных целых чисел, разделенных пробелом, представляющих количество баллов на каждой карточке. Ваша цель - обменять все карты в списке на одну новую карту. Выход Строка 1. Самая низкая общая стоимость завершения конверсии. (Будучи умным игроком, вы должны найти лучшую стратегию, чтобы завершить обмен с наименьшими затратами.)

Рисунок 5.1

## Результат выполнения

class Solution

{

static void Main(string[] args)

{

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

List<int> X = Console.ReadLine().Split().Select(x => int.Parse(x)).ToList();

int cost = 0;

while (X.Count() > 1)

{

X.Sort();

int minSum = X.ElementAt(0) + X.ElementAt(1);

cost += minSum;

X.RemoveAt(0);

X.RemoveAt(0);

X.Add(minSum);

}

Console.WriteLine(cost);

}

}

## Задание

Задание представлено на рисунке 5.2.

Задание: How time flies

(Ваша программа должна печатать отформатированную строку с коли-чеством полных лет и полных месяцев (если они больше 0) и общим количе-ством дней между НАЧИНАТЬ а также КОНЕЦдаты в формате дд.мм.гггг.)

Рисунок 5.2

## Результат выполнения

class Solution

{

static void Main(string[] args)

{

DateTime begin = DateTime.ParseExact(Console.ReadLine(), "dd.MM.yyyy", null);

DateTime end = DateTime.ParseExact(Console.ReadLine(), "dd.MM.yyyy", null);

int days = (end - begin).Days;

int months = (DateTime.MinValue + (end - begin)).Month - 1;

int years = days / 365;

if (years > 0)

{

Console.Write("{0} year{1}, ", years, years > 1 ? "s" : "");

}

if (months > 0)

{

Console.Write("{0} month{1}, ", months, months > 1 ? "s" : "");

}

Console.WriteLine("total {0} days", days);

}

}

# Разработка тестового сценария

## Задание

Задание представлено на рисунке 6.1.

Задание: Ghost legs

Цель: Ghost Legs - разновидность лотереи, распространенной в Азии. Он начинается с ряда вертикальных линий. Между линиями расположены случайные горизонтальные соединители, связывающие все линии в соединенную диаграмму, как показано ниже. ABC | | | | - | | | | - | | | - | | | | 12 3 Чтобы играть в игру, игрок выбирает линию вверху и следует по ней вниз. Когда встречается горизонтальный соединитель, он должен следовать за соединителем, чтобы перейти к другой вертикальной линии и продолжить движение вниз. Повторяйте это, пока не достигнете нижней части диаграммы. На диаграмме примера, когда вы начнете с A, вы попадете в 2. Начиная с B будет в 1. Начиная с C будет в 3. Гарантируется, что каждая верхняя метка будет отображаться на уникальную нижняя этикетка. Учитывая диаграмму Призрачных ног, выясните, какая верхняя метка связана с какой нижней. Перечислите все подключенные пары. (Учитывая диаграмму Призрачных ног, выясните, какая верхняя метка связана с какой нижней. Перечислите все подключенные пары.)

Рисунок 6.1

## Результат выполнения

class Solution

{

static void Main(string[] args)

{

string[] line = Console.ReadLine().Split();

int W = int.Parse(line[0]);

int H = int.Parse(line[1]);

string[] T = Console.ReadLine().Split(new [] { " " }, StringSplitOp-tions.None);

int[] Ti = new int[T.Length];

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

{

Ti[i] = i;

}

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

{

line = Console.ReadLine().Split('|');

for (int j = 0; j < line.Length; j++)

{

if (line[j] == "--")

{

for (int k = 0; k < Ti.Length; k++)

{

if (Ti[k] == j - 1)

{

Ti[k] += 1;

}

else if (Ti[k] == j)

{

Ti[k] -= 1;

}

}

}

}

}

string[] B = Console.ReadLine().Split(new [] { " " }, StringSplitOp-tions.None);

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

{

Console.WriteLine(T[i] + B[Ti[i]]);

}

}

}

## Задание

Задание представлено на рисунке 6.2

Задание: The river 2

Цель: цифровая река - это последовательность чисел, за каждым числом которой следует одно и то же число плюс сумма его цифр. В такой последовательности за 123 следует 129 (поскольку 1 + 2 + 3 = 6), за которым снова следует 141.

Мы называем цифровую реку K, если она начинается со значения K.

Например, река 7 - это последовательность, начинающаяся с {7, 14, 19, 29, 40, 44, 52, ...}, а река 471 - это последовательность, начинающаяся с {471, 483, 498, 519, ...} .

Цифровые реки могут встречаться. Это происходит, когда две цифровые реки имеют одинаковые ценности. Река 32 встречает реку 47 в точке 47, а река 471 встречает реку 480 в точке 519.

Учитывая число, решите, может ли это быть местом встречи двух или более цифровых рек. Например, легко проверить, что только река 20 содержит число 20 в своей последовательности (в качестве отправной точки). (Имея номер, решите, может ли она быть местом встречи двух или бо-лее цифровых рек.)

Рисунок 6.2

## Результат выполнения

class Solution

{

static void Main(string[] args)

{

int r1 = int.Parse(Console.ReadLine());

bool riversMeetR1 = false;

for (int i = r1 - 1; i > 0 && !riversMeetR1; i--)

{

riversMeetR1 = (GetNextRiverNumber(i) == r1);

}

Console.WriteLine(riversMeetR1 ? "YES" : "NO");

}

private static long GetNextRiverNumber(long riverNumber)

{

long nextRiverNumber = riverNumber;

while (riverNumber > 0)

{

nextRiverNumber += riverNumber % 10;

riverNumber -= riverNumber % 10;

riverNumber /= 10;

}

return nextRiverNumber;

}

}

# Инспекция программного кода на предмет соответствия стандартам кодирования

## Задание

Описать процесс формальной инспекции программного кода.

## Результат выполнения

Процесс формальной инспекции программного кода подчиняется всем правилам, определенным для абстрактной формальной инспекции, однако, имеет некоторые особенности, связанные, в первую очередь со структурой инспектируемого программного кода, а также с тем, что обычно инспектируются участки кода, которые невозможно проверить при помощи автоматизированного тестирования, основанного на тестовых примерах.

Выделение ключевых точек и построение или использование таблиц трассировки. Перед началом просмотра исходного кода рекомендуется отметить пункты требований, на соответствие которым проверяется исходный код, а также записать обоснования того, почему эти требования не бывают проверены в автоматическом режиме. После этого можно переходить к просмотру собственно исходного кода. Все пометки, которые придется вносить в ходе инспектирования в исходный код крайне важно делать не в файле, хранящемся в базе данных проекта͵ а в его копии, которая потом будет подшита к материалам инспекции. Копия должна быть в том же формате, что и исходный файл, либо распечатана на бумаге или выведена в формат DOC, PDF или аналогичный, допускающий комментирование.

При помощи трассировочных таблиц в исходном коде определяются инспектируемые функции или методы, соответствующие необходимым требованиям.

Участки кода выделяются и отмечаются меткой или номером соответствующего требования. В случае если участок кода соответствует требованиям, то крайне важно отметить данный факт либо цветом выделения, либо соответствующим текстовым примечанием. В случае если участок кода имеет проблемы, данный факт должен быть отражен либо цветом выделения, либо ссылкой на соответствующий пункт списка замечаний в бланке инспекции.

В случае отсутствия трассировочных таблиц требований на исходный код рекомендуется делать пометки, поясняющие, почему именно данный участок кода реализует указанные требования. Такие пометки помогут на этапе обсуждения документа.

Проверка стиля кодирования. Отдельным объектом проверки при формальной инспекции программного кода является стиль кодирования. В большинстве проектов существуют стандарты, описывающие правила оформления исходных текстов программ и файлов данных. Неверный стиль кодирования не влияет на работоспособность программы в целом, но значительно затрудняет сопровождение и поддержку изменений в ходе дальнейшего развития системы. По этой причине отклонения от стиля кодирования в инспектируемых участках кода также должны отмечаться в тексте и в списке замечаний.

В некоторых случаях проводят инспекции, целиком направленные на проверку стиля кодирования.

Проверка надежности кода. В некоторых случаях рекомендуется проверять наличие участков, гарантирующих робастность, даже если требования прямо не определяют крайне важности обработки недопустимых значений. В случае, в случае если потенциально возможна некорректная работа программы из-за отсутствия обработчиков неверных значений, рекомендуется отметить это в списке замечаний.

# Основы математического моделирования информационных систем

## Задание

Задание представлено на рисунке 8.1.

Задание: The River I

Цель: цифровая река - это последовательность чисел, за каждым числом которой следует одно и то же число плюс сумма его цифр. В такой последовательности за 123 следует 129 (поскольку 1 + 2 + 3 = 6), за которым снова следует 141.

Мы называем цифровую реку K, если она начинается со значения K.

Например, река 7 - это последовательность, начинающаяся с {7, 14, 19, 29, 40, 44, 52, ...}, а река 471 - это последовательность, начинающаяся с {471, 483, 498, 519, ...} .

Цифровые реки могут встречаться. Это происходит, когда две цифровые реки имеют одинаковые ценности. Река 32 встречает реку 47 в точке 47, а река 471 встречает реку 480 в точке 519.

Учитывая две встречи цифровых рек, распечатайте точку встречи.

(Учитывая две встречи цифровых рек, распечатайте точку встречи.)

Рисунок 8.1

## Результат выполнения

class Solution

{

static void Main(string[] args)

{

long r1 = long.Parse(Console.ReadLine());

long r2 = long.Parse(Console.ReadLine());

while (r1 != r2)

{

if (r1 < r2)

{

r1 = GetNextRiverNumber(r1);

}

else

{

r2 = GetNextRiverNumber(r2);

}

}

Console.WriteLine(r1);

}

private static long GetNextRiverNumber(long riverNumber)

{

long nextRiverNumber = riverNumber;

while (riverNumber > 0)

{

nextRiverNumber += riverNumber % 10;

riverNumber -= riverNumber % 10;

riverNumber /= 10;

}

return nextRiverNumber;

}

}

## Задание

Задание представлено на рисунке 8.2

Задание: Lumen

Цель: ОНИ поместят вас в комнату квадратной формы, по N метров с каждой стороны.

ОНИ хотят знать о тебе все.

ОНИ наблюдают за вами.

ОНИ поставили свечи в комнате.

Каждая свеча заставляет L "светиться" на том месте, где они находятся, и каждое пятно квадратной формы получает на один "свет" меньше, чем следующие. Если к точке прикоснутся две свечи, она будет иметь больший «свет». Каждое пятно имеет базовый свет 0.

Вы можете спрятаться, только если найдете темное пятно, у которого 0 "света".

Сколько у вас темных пятен?

Вы получите карту комнаты с пустыми местами (X) и свечами (C) в N строках, каждый символ разделен пробелом.

Пример для светового рассеяния N = 5, L = 3:

Х Х Х Х Х

Х С Х Х Х

Х Х Х Х Х

Х Х Х Х Х

Х Х Х Х Х

2 2 2 1 0

2 3 2 1 0

2 2 2 1 0

1 1 1 1 0

0 0 0 0 0

Рисунок 8.2

## Результат выполнения

class Solution

{

static void Main(string[] args)

{

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

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

int[,] grid = new int[N, N];

for (int x = 0; x < N; x++)

{

string[] line = Console.ReadLine().Split();

for (int y = 0; y < N; y++)

{

grid[x, y] = line[y] == "C" ? L : 0;

}

}

for (int i = L - 1; i > 0; i--)

{

{

if (grid[x, y] == 0)

{

darkSpots++;

}

}

}

Console.WriteLine(darkSpots);

}

}

# ЗАКЛЮЧЕНИЕ

В результате прохождения учебной практики было разработано техническое задание для заданной предметной области, выбрана и реализована архитектуры программного средства. Получены навыки работы в системе контроля версий, выбора и реализации архитектуры программного средства, рассмотрены вопросы, связанные с описанием и анализом требований к системе и разработкой тестового сценария. Была проведена формальная инспекция разработанного программного кода. В заключении учебной практики была решена поставленная задача с использованием математического моделирования.

# СПИСОК ЛИТЕРАТУРЫ

1. Перлова, О. Н. Соадминистрирование баз данных и серверов : учебник для студентов среднего профессионального образования по специальности 09.02.07 "Информационные системы и программирование" / О. Н. Перлова, О. П. Ляпина ; О. Н. Перлова, О. П. Ляпина. – Москва : Академия, 2018. – 304 с. c. – URL: <http://academia-moscow.ru/catalogue/4831/345911/> (дата обращения: 07.06.2021). – Текст : электронный.

2. Федорова, Г. Н. Разработка, внедрение и адаптация программного обеспечения отраслевой направленности : Учебное пособие / Г. Н. Федорова. – Москва : НИЦ ИНФРА-М, 2020. – 336 c. – ISBN 978-5-906818-41-6. – URL: <http://znanium.com/go.php?id=1047718> (дата обращения: 06.06.2021). – Текст : электронный.

3. Гохберг, Г. С. Информационные технологии : учебник для образовательных организаций, реализующих программы среднего профессионального образования по специальностям "Информационные системы и программирование", "Сетевое и системное администрирование" / Г. С. Гохберг, А. В. Зафиевский, А. А. Короткин ; Г. С. Гохберг, А. В. Зафиевский, А. А. Короткин. – Москва : Академия, 2017. – 240 с. c. – (Профессиональное образование). – URL: <http://academia-moscow.ru/catalogue/4831/297236/>. – Текст : непосредственный + электронный.

4. Чистов, Д. В. Проектирование информационных систем.: учебник и практикум для СПО / Чистов Д. В., Мельников П. П., Золотарюк А. В., Ничепорук Н. Б. ; Под общ. ред. Чистова Д.В.. – Москва : Юрайт, 2020. – 258 c. – ISBN 978-5-534-03173-7. – URL: <https://urait.ru/book/proektirovanie-informacionnyh-sistem-452680> (дата обращения: 06.06.2021). – Текст : электронный.