Пермский филиал федерального государственного автономного образовательного учреждения высшего образования «Национальный исследовательский университет «Высшая школа экономики»

Факультет экономики, менеджмента и бизнес-информатики

Образовательная программа бакалавриата «Программная инженерия»

**ОТЧЕТ**

**по учебной практике**

Выполнила студентка группы ПИ-16-1

Спасская Полина Алексеевна

\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_

(подпись)

Проверил:

Руководитель практики

преподаватель кафедры информационных

технологий в бизнесе

Гриневич Татьяна Валерьевна

\_\_\_\_\_\_\_\_\_\_ (оценка) \_\_\_\_\_\_\_\_\_\_ (подпись)

\_\_\_\_\_\_\_\_\_\_\_ (дата)

# Отчет по учебной практике

Введение, в котором описываются цели и задачи практики, краткое введение в предметную область, назначение и базовые функции разработанной системы.

 Введение.

Основную часть, отражающую результаты выполнения заданий учебной практики и включающую:   постановку задачи, формат входных и выходных данных;  описание алгоритма решаемой задачи и комментарии к нему;  описание процесса реализации системы;  описание результатов тестирования системы.

Заключение, в котором студент объективно отражает результаты прохождения практики, достигнутые цели, решенные задачи, варианты развития, доработки системы.

Список литературы, которая была использована студентом при прохождении практики.

Приложение, содержащее:  прилагаемые к отчету документы, справочные материалы, иллюстрации;  листинги программ.

# Введение

*Целью учебной практики* является формирование алгоритмического мышления; закрепление, расширение, углубление и систематизация теоретических знаний, полученных при изучении дисциплин «Введение в программную инженерию», «Дискретная математика», «Компьютерный практикум по основам алгоритмизации и методам программирования», «Программирование»; приобретение навыков и опыта алгоритмизации задач, реализации построенных алгоритмов на языках высокого уровня, публичного выступления при защите отчета о прохождении практики.

*Задачами учебной практики являются:*

* развитие и закрепление практических навыков построения и описания алгоритмов для решения задач из разных предметных областей (численные методы, дискретная математика, структуры данных и др.);
* развитие и закрепление практических навыков использования языков высокого уровня и современных сред разработки для реализации построенных алгоритмов;
* развитие и закрепление практических навыков объектно-ориентированного программирования;
* развитие практических навыков оформления отчетов о проделанной работе, публичного выступления с защитой проекта;
* развитие интереса к научно-исследовательской деятельности.

*Предметной областью практики являются* 10 основных и 2 дополнительных заданий, которые необходимо выполнить с применением знаний, полученных во время изучения следующих дисциплин: Введение в программную инженерию

*В основной части будут представлены* решения 12 поставленных задач, содержащие следующие этапы: анализ задачи, разработку алгоритма, реализацию программной системы и тестирование и отладка приложения.

# Основная часть

Основную часть, отражающую результаты выполнения заданий учебной практики и включающую:

  постановку задачи,

формат входных и выходных данных; 

описание алгоритма решаемой задачи и комментарии к нему;  описание процесса реализации системы;

 описание результатов тестирования системы.

## Задание 1. Прямая и квадраты.

**Анализ**

**Постановка задачи:**

В прямоугольной декартовой системе координат прямая задана двумя принадлежащими ей точками (0, W) и (100•N, E). Также заданы N2 квадратов со сторонами, параллельными осям координат. Квадрат Si,jимеет координаты углов (100•i, 100•j) и (100•i - 100, 100•j - 100), i, j = 1, 2, ..., N.

Требуется найти количество квадратов, имеющих общую точку с прямой.

Ограничения: *Время: 1 сек. Память: 16 Мб*

**Формат входных данных:**

Входной файл INPUT.TXT содержит в одной строке числа N, W и E, разделенные пробелами. (1 ≤ N ≤ 100, 0 ≤ W, E ≤ 100•N). Либо ввод производится с консоли. Проверка корректности входных данных не требуется.

В условии не указано, могут ли входные данные десятичными дробями, это необходимо учитывать при решении задачи, кроме того числа могут быть очень большими.

Выбранный тип входных данных: decimal n, w, e.

**Формат выходных данных:**

В выходной файл OUTPUT.TXT выведите количество квадратов, имеющих общую точку с заданной прямой. Возможен консольный вывол.

Переменныые для вывода: decimal sum.

**Алгоритм решаемой задачи.**

Программа считывает входные данные с консоли и преобразует данные в переменные указанного типа.

Для каждого квадрата. Переменным x1, x2, y1, y2 присваиваются координаты углов квадрата Si, Проверяется, лежит ли ордината прямой для точек x1 и x2 в промежутке значений y1 и y2. Если лежит, то значение переменной для подсчета квадратов увеличивается на 1. После чего вышеперечисленные действия выполняются для следующего квадрата.

После рассмотрения всех квадратов программа выводит результат вычислений на консоль.

Уравнение прямой y = w + x \* (e - w) / (100 \* n).

Координаты углов x1 = 100 \* i - 100, x2 = 100 \* i, y1 = 100 \* j - 100, y2 = 100 \* j.

Подробное описание алгоритма в блок-схеме ниже.



**Процесс реализации системы.**

Для всех переменных, используется тип decimal - точный дробный или целочисленный тип, который может представлять десятичные числа с 29 значащими цифрами. При использовании других типов данных, программа проходит не все тесты.

В качестве входных данных считывается строка, которая и преобразуется в три используемые переменные. Для рассмотрения каждого квадрата используется цикл по переменной i и вложенный цикл по переменной j. Каждый цикл выполняется, пока переменная не достигает значения n.

Так как программа тестируется автоматически, дружественныйинтерфейс не требуется. Листинг программы представлен по ссылке <https://acmp.ru/index.asp?main=source&id=6840854>, в репозитории и ниже.

**Листинг программы:**

public static void Main(string[] args)

{

/\*В прямоугольной декартовой системе координат прямая задана двумя принадлежащими ей точками (0, W) и (100•N, E).

Также заданы N2 квадратов со сторонами, параллельными осям координат.

Квадрат Si,jимеет координаты углов (100•i, 100•j) и (100•i - 100, 100•j - 100), i, j = 1, 2, ..., N.

Требуется найти количество квадратов, имеющих общую точку с прямой.\*/

decimal n, w, e; // входные данные

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

//Console.WriteLine("Введите через пробел n w e");

n = decimal.Parse(s[0]);

w = decimal.Parse(s[1]);

e = decimal.Parse(s[2]);

decimal sum = 0;

for (decimal i = 1; i <= n; i++)

for (decimal j = 1; j <= n; j++)

{

decimal x1 = 100 \* i - 100, x2 = 100 \* i, y1 = 100 \* j - 100,

y2 = 100 \* j, y = w + x1 \* (e - w) / (100 \* n);

if ((y - y1) \* (y - y2) > 0)

{

y = w + x2 \* (e - w) / (100 \* n);

if ((y - y1) \* (y - y2) <= 0)

sum++;

}

else sum++;

}

//Console.WriteLine("Количество квадратов, имеющих общую точку с прямой:");

Console.WriteLine(sum); // выходные данные

Console.ReadLine();

}

**Описание результатов тестирования системы**

Программа проходила тестирование на ресурсе acmp.ru . Разработанный код успешно прошео 10 из 10 предложенных тестов. Подробности представлены в таблице ниже.

|  |  |  |  |
| --- | --- | --- | --- |
| Тест | Результат | Время | Память |
| 1 | Accepted | 0,062 | 2229 Кб |
| 2 | Accepted | 0,062 | 2233 Кб |
| 3 | Accepted | 0,092 | 2229 Кб |
| 4 | Accepted | 0,062 | 2233 Кб |
| 5 | Accepted | 0,092 | 2233 Кб |
| 6 | Accepted | 0,092 | 2229 Кб |
| 7 | Accepted | 0,124 | 2229 Кб |
| 8 | Accepted | 0,092 | 2229 Кб |
| 9 | Accepted | 0,062 | 2229 Кб |
| 10 | Accepted | 0,092 | 2229 Кб |

## Задание 2. Друзья - 2.

**Постановка задачи:**

**Не закончена**

Несколько человек решили поехать отдохнуть на природе, подышать свежим воздухом и т.п. Как это часто бывает, некоторые из них дружат друг с другом, а некоторые - нет. Для того, чтобы не испортить никому настроение, они решили разделиться на несколько групп. При этом, в каждой группе должно быть не более 5 человек и они должны дружить друг с другом.

Найдите такое разбиение людей на группы, в котором размер наибольшей группы был бы максимальным (среди всех разбиений).

Ограничения: *Время: 1 сек. Память: 16 Мб*

**Формат входных данных:**

Первая строка входного файла INPUT.TXT содержит целое число n (1 ≤ n ≤ 15) - количество людей. Следующие n строк содержат по n чисел. Если i-ый и j-ый люди дружат, то j-ое число i + 1-ой строки равно 1, иначе - 0.

Также ввод может производиться с консоли. Проверка корректности входных данных не требуется.

По условию задачи все числа целые. Выбранный тип int.

«Взаимоотношения» людей будут представлены в формате массива int[n,n].

**Формат выходных данных:**

В первой строке выходного файла OUTPUT.TXT выведите число групп. Во второй строке выходного файла выведите n чисел (i-ое число - номер группы, в которой находится i-ый человек). Так как в любом случае количество групп не превзойдет n, нумеруйте группы целыми числами от 1 до n. Если решений несколько, то выведите любое.

!

Переменныые для вывода: decimal sum.

**Алгоритм решаемой задачи.**

Программа считывает входные данные с консоли и преобразует данные в переменные указанного типа. //

**Процесс реализации системы.**

Для всех переменных, используется тип decimal - точный дробный или целочисленный тип, который может представлять десятичные числа с 29 значащими цифрами.

Листинг программы представлен по ссылке .

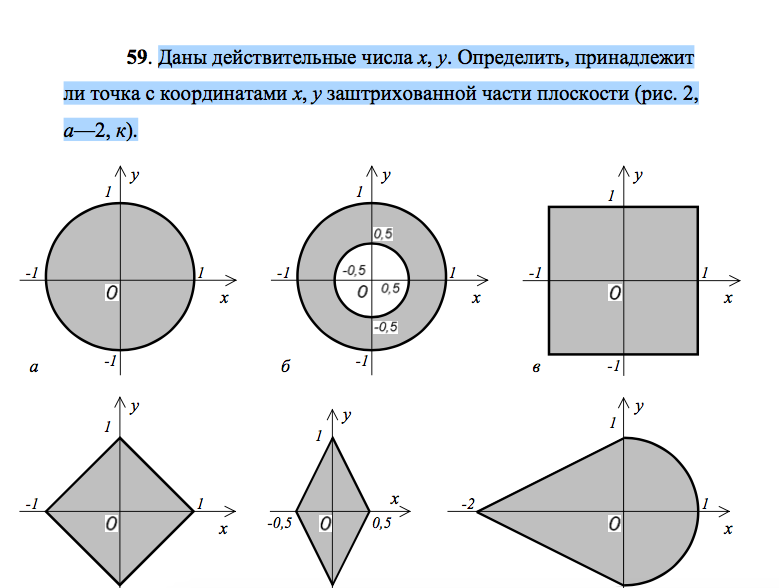
**Описание результатов тестирования системы**

Программа проходила тестирование на ресурсе acmp.ru . Разработанный код успешно прошео 10 из 10 предложенных тестов. Подробности представлены в таблице ниже.

## Задание 3.

**Постановка задачи:**

Даны действительные числа x, y. Определить, принадлежит ли точка с координатами x, y заштрихованной части плоскости.

****

*Комментарий:* Точки, принадлежащие заштрихованной части плоскости, - точки, сумма модулей абсциссы и ординаты которых меньше или равна 1.

**Формат входных данных:**

Действительные числа x, y. Выбранный тип double. Необходимо осуществить проверку корректности ввода числа.

**Формат выходных данных:**

Строка. «Да», когда точка принадлежит области, «Нет» - в обратном случае.

**Алгоритм решаемой задачи.**

Необходимо ввести данные координаты точки. Сложить их модули и сравнить полученную сумму с 1. Если сумма меньше, вывести положительный ответ, иначе отрицательный. Блок-схема алгоритма представлена ниже.



**Процесс реализации системы.**

Программа считывает входные данные с консоли и преобразует данные в переменные типа double. Ввод и проверка корректности данных осуществляется при помощи функции VVOD(), возвращающей значение типа double. Принадлежит ли точка заштрихованной области проверяет функция IsPointInSquare(x, y), в качестве аргументов принимающая значения координат точки и возвращающая значение типа bool. В зависимости от результата работы функции программа возвращает пользователю ответ.

Листинг программы представлен в репозитории и ниже.

**Листинг программы:**

public static void Main(string[] args)

{

Console.WriteLine("Введите значение x (ожидается действительное число)");

double x = VVOD();

Console.WriteLine("Введите значение y (ожидается действительное число)");

double y = VVOD();

Console.WriteLine("Данная точка принадлежит заштрихованной части плоскости?");

if (IsPointInSquare(x, y))

Console.WriteLine("Да");

else

Console.WriteLine("Нет");

Console.ReadLine();

}

static bool IsPointInSquare(double x1, double y1)

{

return (Math.Abs(y1) <= -Math.Abs(x1) + 1);

}

static double VVOD()

{

double value;

bool rightValue;

do

{

string inputValue = Console.ReadLine();

rightValue = double.TryParse(inputValue, out value);

if (!rightValue)

{

Console.WriteLine("Неверное значение для дробного числа вида 0,0 , повторите.");

}

}

while (!rightValue);

return value;

}

**Описание результатов тестирования системы**

// - ???

Корректность ввода

Число внутри области

Число на границе

Число вне области

<> 0

Тестирование проведено по критериям черного ящика.

Тестирование входных данных: коректные/некорректные. Сообщение об ошибке и запрос повторного ввода, если данные некорректны. Продолжение выполнения программы после получения корректных данных.

Тестирование выходных данных. Программа может выводить «Да», когда точка принадлежит области, «Нет» - в обратном случае. Если точка принадлежит заштрихованной области или лежит на ее границе, программа должна вывести «Да», если точка лежит вне области программа должна вывести «Нет». Корректная работа программы не должна зависеть от знака x и y. Также программа должна выводить сообщение об ошибке, если входные данные некорректны.

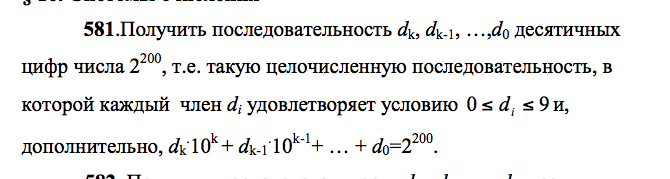
Тестирование функций: VVOD(), IsPointInSquare(x, y).

Основные сценарии тестирования представлены в таблице ниже.

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
| Номер теста | Входные данные | Ожидаемый результат | Реальный результат | Критерий (сценарий) тестирования |
| 1 | x = sss, y = 5 | Неверное значение для дробного числа вида 0,0 , повторите. | + | Тестирование входных данных, выходных данных, функции VVOD().  Некорректный x |
| 2 | x = 1,1 , y = 5.5 | Неверное значение для дробного числа вида 0,0 , повторите. | + | Тестирование входных данных, выходных данных, функции VVOD()  Некорректный y |
| 3 | x = 1, y = 1 | Нет | + | Тестирование входных данных, выходных данных, функций VVOD(), IsPointInSquare(x, y).  Точка не принадлежит области, x и y положительны |
| 4 | x = -1, y = -1 | Нет | + | Тестирование входных данных, выходных данных, функций VVOD(), IsPointInSquare(x, y).  Точка не принадлежит области, x и y отрицательны |
| 5 | x = 1, y = -1 | Нет | + | Тестирование входных данных, выходных данных, функций VVOD(), IsPointInSquare(x, y).  Точка не принадлежит области, x и y разных знаков |
| 5 | x = 0, y = -1 | Да | + | Тестирование входных данных, выходных данных, функций VVOD(), IsPointInSquare(x, y).  Точка лежит на границе области |
| 5 | x = -0,2, y = 0,2 | Да | + | Тестирование входных данных, выходных данных, функций VVOD(), IsPointInSquare(x, y).  Точка лежит внутри области |

## Задание 4.

**Постановка задачи:**

****

**Формат входных данных:**

Ввод данных с консоли или из файла не осуществляется.

**Формат выходных данных:**

Последовательность чисел, каждый элемент которой представляет цифру одного из разрядов 2 в 200 степени. Выходные данные всегда одно и то же число.

*Комментарий:* Длина данной последовательность в несколько раз превышает размерность любой из переменных, существующих в языке C#, поэтому необходимо использовать длинную арифметику. Длинная арифметика — выполняемые с помощью вычислительной машины арифметические операции над числами, разрядность которых превышает длину машинного слова данной вычислительной машины.

**Алгоритм решаемой задачи.**

Необходимо реализовать длинную арифметику, в которой цифры числа – элементы динамической структуры, способной увеличиваться в 2 раза. Изначальное значение 1, необходимо 200 раз увеличить в 2 раза. Алгоритм работы программы представлен в блок-схеме ниже.



**Процесс реализации системы.**

Реализованы два варианта программы: с двухсторонним списком Mas, реализованным вручную и классом коллекции List<T>. Выбор классов обусловлен тем, что для реализации алгоритма необходим динамично увеличивающийся класс связанных элементов. Ниже представлен листинг класса Mas.

public class Mas

{

public Mas next = null;

public int elem;

public Mas last = null;

public Mas(int a)

{

elem = a;

}

public Mas(int a, Mas b)

{

elem = a; last = b;

}

}

Изначально создается класс, содержащий один элемент 1. После чего к нему 200 раз применяется функция Degree(), удваивающая число. Цифры числа в коллекциях располагаются от младших разрядов к старшим, так как старшие разряды добавляются постепенно. В результате число получается «перевернутым». В случае с List<T> используется метод Reverse(), двухсторонний список выводится от последнего элемента к первому.

Функция Degree() сходна для обоих реализаций. Она поочередно удваивает каждый элемент структуры, контролируя, чтобы полученное число не превышало 9 и увеличивая (добавляя) старшие разряды по необходимости.

Листинг программы представлен в репозитории и ниже.

Листинг программы с двухсторонним списком:

static Mas newarr = new Mas(1); //возводимое число - 2^0, представление - длинная арифметика

static void Main(string[] args)

{

Console.WriteLine("2 в 200 степени =");

int degree = 200; // степень, в которую необходимо возвести 2

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

Degree();

while (newarr.next != null) newarr = newarr.next;

while (newarr != null)

{

Console.Write(newarr.elem);

newarr = newarr.last;

}

Console.ReadLine();

}

static void Degree() //функция удваивающая число

{

Mas head = newarr, first = newarr;

int k = 0;

while (newarr != null)

{

newarr.elem = newarr.elem \* 2 + k;

k = 0;

if (newarr.elem >= 10)

{

k = 1;

newarr.elem = newarr.elem - 10;

}

first = newarr;

newarr = newarr.next;

}

if (k == 1)

first.next = new Mas(1, first);

newarr = head;

}

}

Листинг программы с стандартным списком:

static void Main(string[] args)

{

int degree = 200; // степень, в которую необходимо возвести 2

List<int> arr = new List<int> { 1 }; //возводимое число - 2^0, представление - длинная арифметика

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

Degree(ref arr);

arr.Reverse();

foreach (int a in arr)

Console.Write(a);

Console.ReadLine();

}

static void Degree(ref List<int> arr) //функция удваивающая число

{

int c = arr.Count, k = 0;

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

{

arr[i] = arr[i] \* 2 + k;

k = 0;

if (arr[i] >= 10)

{

k = 1;

arr[i] = arr[i] - 10;

}

}

if (k == 1)

arr.Add(1);

}

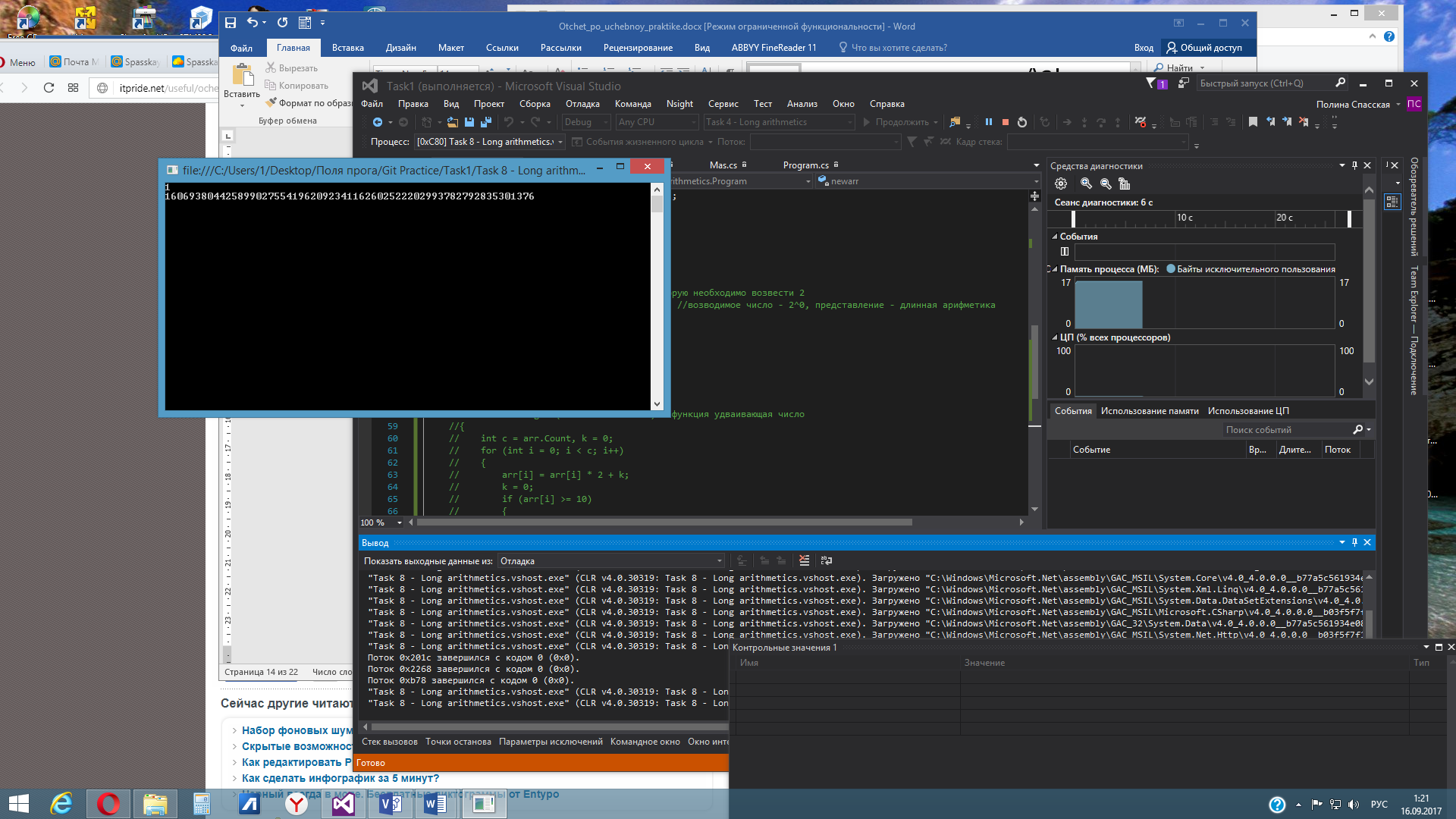
**Описание результатов тестирования системы**

Число, полученное на выходе должно совпадать с табличным значение 2 в 200.

Табличное значение:

1606938044258990275541962092341162602522202993782792835301376

Полученное значение:

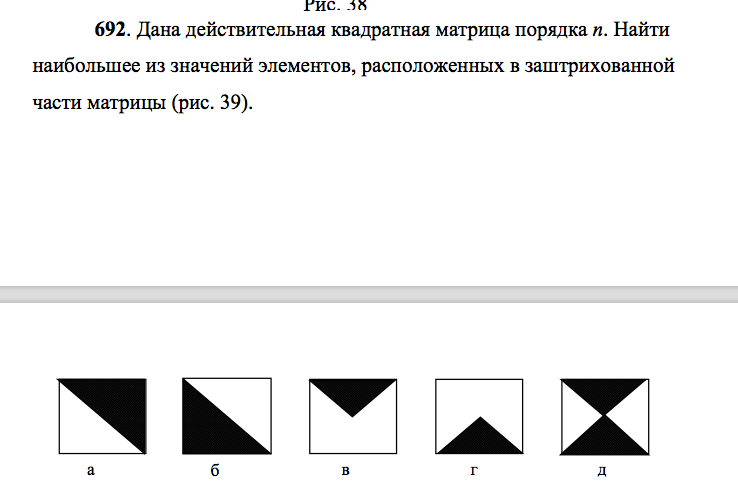


Программа работает корректно.

## Задание 5.

**Постановка задачи:**

Дана действительная квадратная матрица порядка n. Найти наибольшее из значений элементов, расположенных в заштрихованной части матрицы (рис. 13).

****

**Формат входных данных:**

Порядок матрицы n – натуральное число, которое пользователю необходимо ввести с клавиатуры. Требуется проверка ввода натурального числа.

Элементы матрицы – n\*n действительных чисел. Пользователю предложен выбор – ввести числа с клавиауры (требуется проверка ввода действительного числа) или сгенерировать автоматически (в этом случае, сгенерированные числа будут принадлежать промежутку < <).

**Формат выходных данных:**

Необходимо вывести наибольшее найденное значение, принадлежащее заштрихованной области – действительное число. Если их несколько - ?

Формат вывода: «».

**Алгоритм решаемой задачи.**

Отношение строк к столбцам элементов заштрихованной области выражается отношением i j, где i – строка элемента, j – столбец элемента. Необходимо осуществить перебор элементов, отвечающих данному условию и найти среди них максимальный. i <= j

**Процесс реализации системы.**

////

Листинг программы представлен по ссылке .

**Описание результатов тестирования системы**

Введено некорректное значение порядка и/или элементов

Все элементы положительные

Отрицательные

Положительные и отрицательные

Наибольший элемент матрицы вне области

В области

Наибольших элементов несколько (комментрарий, что мы ищем значение, а не порядковый номер).

## Задание 6.

**Постановка задачи:**

**Используя рекурсию**, реализовать программу, решающую задачу, соответствующую Вашему варианту:

Ввести а1, а2, а3, М, N. Построить последовательность чисел ак = 2 \* | ак–1 – ак-2 | + ак–3. Построить N элементов последовательности, либо найти первые M ее элементов, кратных трем (в зависимости от того, что выполнится раньше). Напечатать последовательность и причину остановки.

**Формат входных данных:**

Входные данные: а1, а2, а3, М, N.

а1, а2, а3 – действительные числа, вводятся пользователем с клавиатуры. Необходимо осуществить проверку ввода действительного числа.

М, N – натуральные числа, вводятся пользователем с клавиатуры. Необходимо осуществить проверку ввода натурального числа.

**Формат выходных данных:**

Выходные данные: элементы последовательности – действительные числа. Строка, объясняющая причину остановки (обстановки).

Пример:

**Алгоритм решаемой задачи.**

Программа считывает входные данные с консоли

. //

**Процесс реализации системы.**

Для всех переменных, используется тип decimal - точный дробный или целочисленный тип, который может представлять десятичные числа с 29 значащими цифрами.

//

Листинг программы представлен по ссылке .

**Описание результатов тестирования системы**

Программа проходила тестирование на ресурсе acmp.ru . Разработанный код успешно прошео 10 из 10 предложенных тестов. Подробности представлены в таблице ниже.

//

## Задание 7.

**Постановка задачи:**

Реализовать программу, решающую задачу, соответствующую Вашему варианту:

Заданы частоты символов входного алфавита. Построить двоичный префиксный код Хаффмана. Кодовые слова выписать в лексикографическом порядке.

**Формат входных данных:**

Количество символов в алфавите n – натуральное число. Необходимо осуществить проверку ввода натурального числа.

Частоты символов алфавита – n действительных чисел. Необходимо осуществить проверку ввода действительных чисел.

**Формат выходных данных:**

n кодовых слов Хаффмана (двоичных), расположенных в лексикографическом порядке.

**Алгоритм решаемой задачи.**

Программа считывает входные данные с консоли и преобразует данные в переменные указанного типа. //

**Процесс реализации системы.**

Для всех переменных, используется тип decimal - точный дробный или целочисленный тип, который может представлять десятичные числа с 29 значащими цифрами.

//

Листинг программы представлен по ссылке .

**Описание результатов тестирования системы**

Программа проходила тестирование на ресурсе acmp.ru . Разработанный код успешно прошео 10 из 10 предложенных тестов. Подробности представлены в таблице ниже.

//

## Задание 8.

**Постановка задачи:**

Реализовать программу, решающую задачу, соответствующую Вашему варианту. Для тестирования программы разработать генератор тестов, который позволит сгенерировать набор входных данных, используемых при тестировании.

Граф задан матрицей смежности. Найти в нем какую-либо клику из K вершин.

**Формат входных данных:**

Число вершин графа n – натуральное число.

Матрица смежности - ? число ребер - ?

Матрица смежности – двоичный массив размернасти n\*n (m?), заполненный натуральными (целыми/1 и 0 - ?) числами.

К число вершин в искомой клике – натуральное число.

Число вершин и матрица смежности графа (К - ?) генерируются автоматически, это числа в указанном диапазоне. Также существует возможность ввода параметров через консоль. В этом случае необходимо осуществить проверку ввода натуральных чисел.

Генератор тестов - !!!!!!

**Формат выходных данных:**

Вывести номера K вершин, состовляющих клику графа.

**Алгоритм решаемой задачи.**

Программа считывает входные данные с консоли и преобразует данные в переменные указанного типа. //

А хз

**Процесс реализации системы.**

Для всех переменных, используется тип decimal - точный дробный или целочисленный тип, который может представлять десятичные числа с 29 значащими цифрами.

//

Листинг программы представлен по ссылке .

**Описание результатов тестирования системы**

Программа проходила тестирование на ресурсе acmp.ru . Разработанный код успешно прошео 10 из 10 предложенных тестов. Подробности представлены в таблице ниже.

//

## Задание 9.

**Постановка задачи:**

Выполнить задание, реализовав динамические структуры данных «вручную», **без использования коллекций** языка C#.

В программе построен циклический список. Напишите процедуру подсчета двух сумм: всех положительных и всех отрицательных значений, записанных в информационные поля элементов списка.

**Формат входных данных:**

Действительные числа x, y. Выбранный тип double. Необходимо осуществить проверку корректности ввода числа. //

**Формат выходных данных:**

Строка. «Yes» - когда точка принадлежит области, «No» - в обратном случае. //

**Алгоритм решаемой задачи.**

Программа считывает входные данные с консоли и преобразует данные в переменные указанного типа. //

**Процесс реализации системы.**

Для всех переменных, используется тип decimal - точный дробный или целочисленный тип, который может представлять десятичные числа с 29 значащими цифрами.

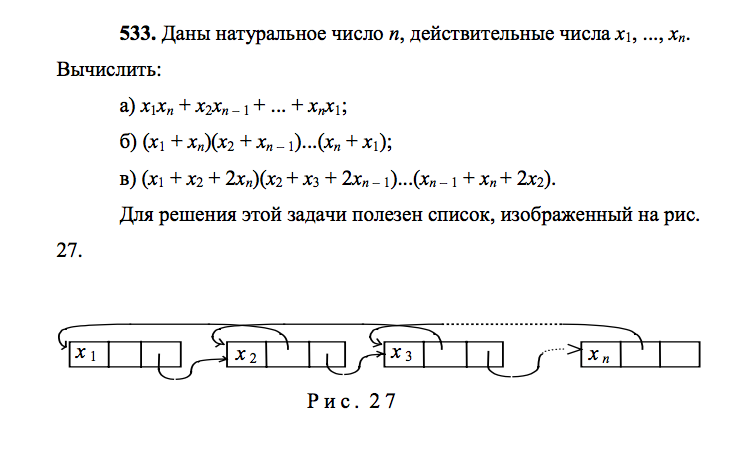
Листинг программы представлен по ссылке .

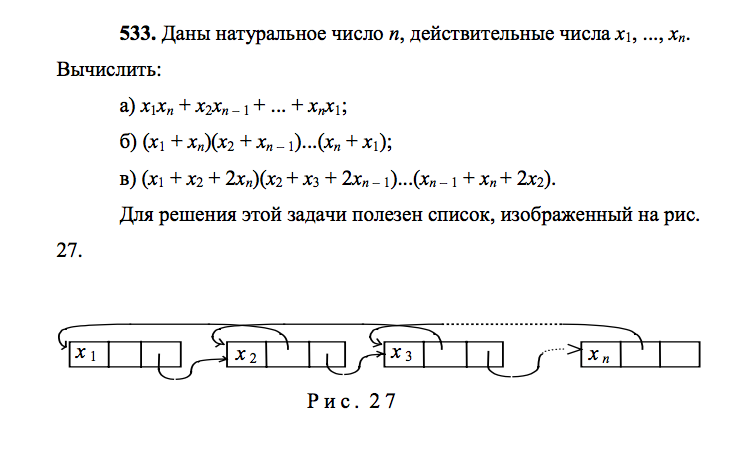
**Описание результатов тестирования системы**

Программа проходила тестирование на ресурсе acmp.ru . Разработанный код успешно прошео 10 из 10 предложенных тестов. Подробности представлены в таблице ниже.

## Задание 10.

**Постановка задачи:**

****

****

*Комментарий:* необходимо найти произведение сумм концов сначала сходящейся потом расходящейся (по номерам элементов) последовательности.

**Формат входных данных:**

Натуральное число n – количество элементов последовательности. Необходимо осуществить проверку корректности ввода натурального числа.

Последовательность из n действительных чисел – элементы последовательности. В условии не указан тип элементов, поэтому по умолчанию считаем их вещественными. Необходимо осуществить проверку корректности ввода вещественного числа.

**Формат выходных данных:**

Строка. "Результат, полученный по формуле (x1 + xn)(x2 + xn-1)...(xn + x1): " и вывод результата sum – действительное число.

**Алгоритм решаемой задачи.**

Необходимо найти произведение сумм концов сначала сходящейся потом расходящейся (по номерам элементов) последовательности, поэтому элементы запоминаются в двухсторонний список. В отдельные переменные запоминаются начало и конец этого списка, после чего элементы перебираются с двух разных сторон, складываются между собой и перемножаются на предыдущие перемноженные суммы. Алгоритм представлен в блок-схеме ниже.



**Процесс реализации системы.**

Программа считывает входные данные с консоли и преобразует данные в переменные типа int и double. Ввод и проверка корректности данных осуществляется при помощи функций VVODN() (возвращает значение типа int > 0) и VVOD () (возвращает значение типа double).

Ниже представлен листинг класса Mas, реализующего двухсторонний список.

public class Mas

{

public Mas next = null;

public int elem;

public Mas last = null;

public Mas(int a)

{

elem = a;

}

public Mas(int a, Mas b)

{

elem = a; last = b;

}

}

Начало и конец списка запоминается в переменные head и tail (типа Mas). head переходит к следующему элементу, tail – к предыдущему. При прохождении списка с двух сторон фиксируется, чтобы ни одна из переменных не вышла за его пределы. Промежуточный результат вычислений запоминается в переменную sum (начальное значение 1), которая в последствии и выводится.

Листинг программы представлен в репозитории и ниже.

class Program

{

static Mas newarr = null;

static void Main(string[] args)

{// читаем n

// читаем все элементы и загружаем в динамический массив

// сохраняем начало и конец массива

// указываем двойное условие и идем с двух концов массива

Console.WriteLine("Введите кол-во элементов (целое число)");

int n = VVODN();

Console.WriteLine("Введите 1 элемент (действительное число)");

newarr = new Mas(VVOD());

Mas head = newarr;

for (int i = 2; i <= n; i++)

{

Console.WriteLine("Введите " + i + " элемент (действительное число)");

newarr.next = new Mas(VVOD(), newarr);

newarr = newarr.next;

}

Mas tail = newarr;

double sum = 1; // самый большой вещественный тип данных

while (head != null || tail != null)

{

sum \*= head.elem + tail.elem;

head = head.next;

tail = tail.last;

}

Console.Write("Результат, полученный по формуле (x1 + xn)(x2 + xn-1)...(xn + x1): " + sum);

Console.ReadLine();

}

static double VVOD()

{

double value;

bool rightValue;

do

{

string inputValue = Console.ReadLine();

rightValue = double.TryParse(inputValue, out value);

if (!rightValue)

{

Console.WriteLine("Неверное значение для дробного числа вида 0,0 , повторите.");

}

}

while (!rightValue);

return value;

}

static int VVODN()

{

int value;

bool rightValue;

do

{

string inputValue = Console.ReadLine();

rightValue = int.TryParse(inputValue, out value);

if (!rightValue || value < 1)

{

Console.WriteLine("Неверное значение, ожидалось натуральное число. повторите.");

}

}

while (!rightValue || value < 1);

return value;

}

}

**Описание результатов тестирования системы**

//

## Задание 11.

**Постановка задачи:**

Один из простейших способов шифровки текста состоит в табличной замене каждого символа другим символом – его шрифтом.

Выбрав некоторую таблицу, разработать способ ее представления, затем

1. зашифровать данный текст;
2. расшифровать данный текст.

**Формат входных данных:**

Входные данные: строка, содержащая текст для шифрования. Программа принимает символы….

**Формат выходных данных:**

Строка. «Yes» - когда точка принадлежит области, «No» - в обратном случае. //

**Алгоритм решаемой задачи.**

Программа считывает входные данные с консоли и преобразует данные в переменные указанного типа. //

**Процесс реализации системы.**

Для всех переменных, используется тип decimal - точный дробный или целочисленный тип, который может представлять десятичные числа с 29 значащими цифрами.

//

Листинг программы представлен по ссылке .

**Описание результатов тестирования системы**

Программа проходила тестирование на ресурсе acmp.ru . Разработанный код успешно прошео 10 из 10 предложенных тестов. Подробности представлены в таблице ниже.

//

## Задание 12.

**Постановка задачи:**

Выполнить сравнение двух предложенных методов сортировки одномерных массивов, содержащих n элементов, по количеству пересылок и сравнений.

Для этого необходимо выполнить программную реализацию двух методов сортировки, включив в нее подсчет количества пересылок (т.е. перемещений элементов с одного места на другое) и сравнений.

Провести анализ методов сортировки для трех массивов: упорядоченного по возрастанию, упорядоченного по убыванию и неупорядоченного.

Все три массива следует отсортировать обоими методами сортировки.

Найти в литературе теоретические оценки сложности каждого из методов и сравнить их с оценками, полученными на практике.

Сделать выводы о том, насколько отличаются теоретические и практические оценки количества операций, объяснить почему это происходит. Сравнить оценки сложности двух алгоритмов.

Вариант задания определяется парой (X, Y), где X, Y – порядковые номера методов сортировки из приведенного списка:

1. Пирамидальная сортировка.

2. Сортировка подсчётом.

**Формат входных данных:**

Действительные числа x, y. Выбранный тип double. Необходимо осуществить проверку корректности ввода числа.

**Формат выходных данных:**

Строка. «Yes» - когда точка принадлежит области, «No» - в обратном случае.

**Алгоритм решаемой задачи.**

Программа считывает входные данные с консоли и преобразует данные в переменные указанного типа. //

**Процесс реализации системы.**

Для всех переменных, используется тип decimal - точный дробный или целочисленный тип, который может представлять десятичные числа с 29 значащими цифрами.

Листинг программы представлен по ссылке .

**Описание результатов тестирования системы**

Программа проходила тестирование на ресурсе acmp.ru . Разработанный код успешно прошео 10 из 10 предложенных тестов. Подробности представлены в таблице ниже.