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  (Фамилия, Имя, Отчество)  (подпись) |
| Проверил  Руководитель практики:  Преподаватель кафедры информационных технологий в бизнесе  (должность, учёная степень, учёное звание)  Гриневич Татьяна Валерьевна  (Фамилия, Имя, Отчество) | |  |
| (оценка) | (подпись)  (дата) |  |

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

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 не доделана

Постановка задачи:

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].

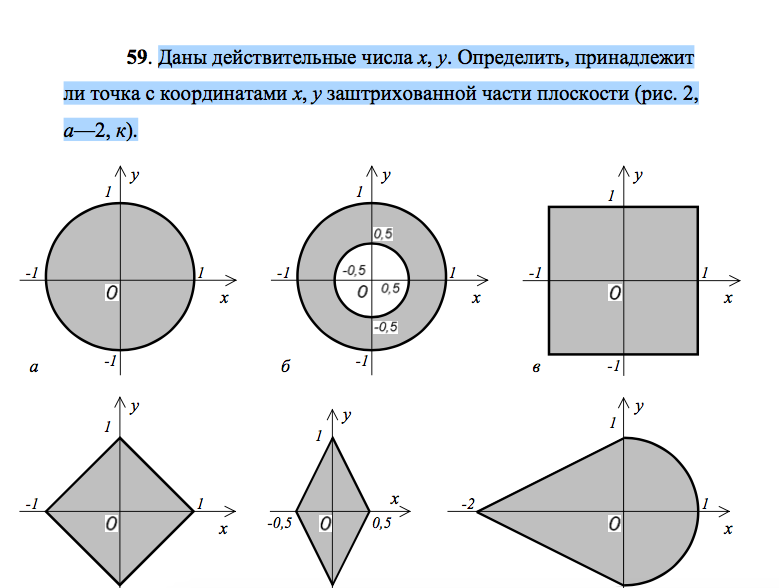
Формат выходных данных:

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.

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

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.

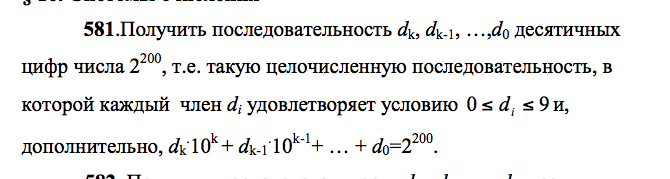
Тестирование функций: VVOD(), IsPointInSquare(x, y).

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);

}

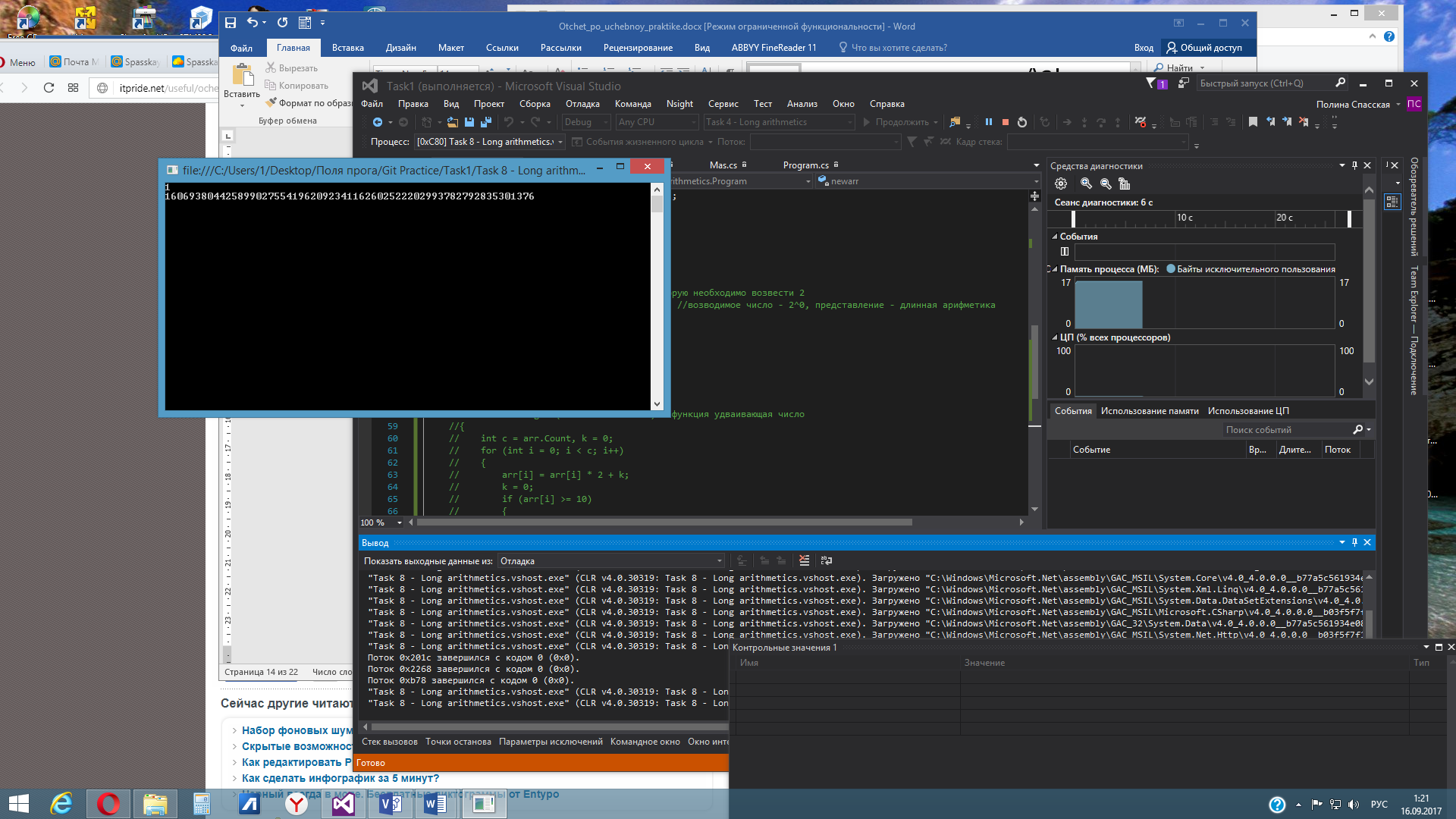
Описание результатов тестирования системы

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

Табличное значение:

1606938044258990275541962092341162602522202993782792835301376

Полученное значение:

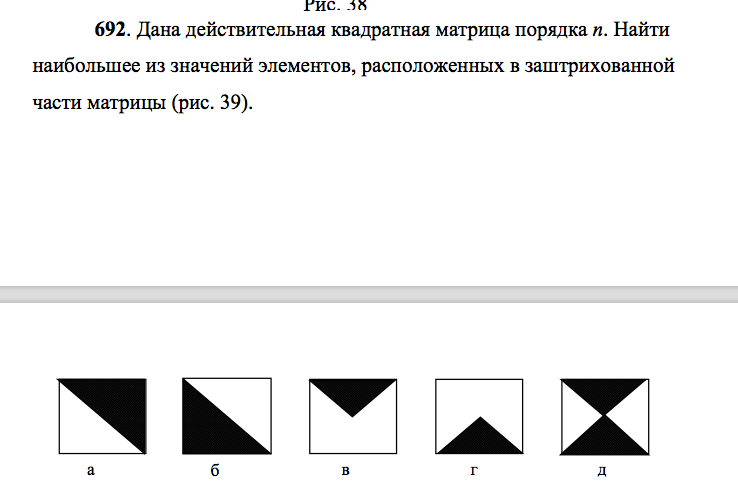


Программа работает корректно.

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 клавиатуры (требуется проверка ввода действительного числа) или сгенерировать автоматически (в этом случае, сгенерированные числа будут принадлежать промежутку от 0 до 11).

Формат выходных данных:

Необходимо вывести наибольшее найденное значение, принадлежащее заштрихованной области – действительное число max. Если таких значений несколько, то выходные данные не изменятся, так как требуется вывести именно значение, а не номер элемента.

Если максимальный элемент матрицы расположен вне области, выводится максимальный элемент заштрихованной области. Если в или на границе области, он и выводится.

Перед выводом наибольшего значения области, осуществляется вывод матрицы, с точностью вывода значений два знака после запятой и раскрашенными в зеленый цвет элементами заштрихованной области (основной цвет элементов белый).

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. Строки и столбцы элементов заштрихованной области относятся как i <= j, поэтому для поиска максимального значения используется неполный перебор элементов: цикл по i от 0 до n, вложенный цикл по j от i n до n (j >= i). Алгоритм работы программы 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). Либо элементы матрицы генерируются автоматически (в этом случае, сгенерированные числа будут принадлежать промежутку от 0 до 11). Пользователю предоставлен выбор.

Перебор элементов осуществляется при помощи вложенных циклов for.

Листинг программы представлен в репозитории и ниже.

Листинг программы:

static void Main(string[] args)

{

// read or genirate

//Дана действительная квадратная матрица порядка n.

//Найти наибольшее из значений элементов, расположенных в заштрихованной части матрицы

// read M

Console.WriteLine("Задание 5");

Console.WriteLine("Введите размерность матрцы (целое число)");

int n = VVODN();

double[,] M = new double[n, n];

Console.WriteLine("Если вы хотите ввести элементы вручную, нажмите 1. В противном случае элементы будут сгенерированы автоматически.");

if (Console.ReadLine() == "1")

{

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

{

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

{

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

M[i, j] = VVOD();

}

}

//вывести всю матрицу

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

{

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

{

if (i <= j)

{

Console.ForegroundColor = ConsoleColor.Green;

Console.Write("{0:0.00} ", M[i, j]);

Console.ForegroundColor = ConsoleColor.White;

}

else

Console.Write("{0:0.00} ", M[i, j]);

}

Console.WriteLine();

}

}

else

{//genirate M:

Random rand = new Random();

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

{

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

{

M[i, j] = rand.NextDouble() + rand.Next(0, 10);

if (i <= j)

{

Console.ForegroundColor = ConsoleColor.Green;

Console.Write("{0:0.00} ", M[i, j]);

Console.ForegroundColor = ConsoleColor.White;

}

else

Console.Write("{0:0.00} ", M[i, j]);

}

Console.WriteLine();

}

}

double max = M[0, 0];

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

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

if (i <= j && M[i, j] > max)

max = M[i, j];

Console.ForegroundColor = ConsoleColor.DarkCyan;

Console.WriteLine("Наибольшее из значений, расположенных в заштрихованной части:");

Console.WriteLine(" {0:0.00}", max);

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

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. Возможность генерации элементов.

Тестирование выходных данных. Программа возвращает наибольшее найденное значение, принадлежащее заштрихованной области. Если таких значений несколько, то выходные данные не изменятся, так как требуется вывести именно значение, а не номер элемента. Корректная работа программы не должна от местоположения максимального элемента. Также программа должна выводить сообщение об ошибке, если входные данные некорректны.

Тестирование матриц разной размерности.

Тестирование функций: VVOD(), VVODN()

Основные сценарии тестирования представлены в таблице ниже.

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
| Номер теста | Входные данные | Ожидаемый результат | Реальный результат | Критерий (сценарий) тестирования |
| 1 | n = 0 | Неверное значение, ожидалось натуральное число. повторите. | + | Тестирование входных данных, функции функции VVODN().  Некорректный n |
| 2 | n = 3  1 1 3  1 1 1  1 1 1.1 | Неверное значение для дробного числа вида 0,0 , повторите. | + | Тестирование входных данных, функции функции VVODN(), функции VVOD()  Некорректный элемент |
| 3 | n = 3  1 1 3  1 1 1  1 1 1 | Max = 3. | + | Тестирование входных данных, выходных данных, функции VVOD(),функции VVODN().  Максимальный элемент матрицы совпадает с максимальным элементом заштрихованной области |
| 4 | n = 3  1 1 3  1 1 1  5 1 1 | Max = 3. | + | Тестирование входных данных, выходных данных, функции VVOD(),функции VVODN().  Максимальный элемент матрицы не совпадает с максимальным элементом заштрихованной области |
| 5 | n = 3  1 1 3  1 1 3  1 1 1 | Max = 3. Максимальных элементов в заштрихованной области несколько | + | Тестирование входных данных, выходных данных, функции VVOD(),функции VVODN().  Максимальных элементов в заштрихованной области несколько |
| 6 | n = 1  3,3 | Max = 3,3 | + | Тестирование входных данных, выходных данных, функции VVOD(),функции VVODN().  Матрица маленькой размерности – все элементы в заштрихованной области |
| 7 | n = 20 | Max = 9,57 | + | Тестирование входных данных, выходных данных, функции VVODN().  Автоматическая генерация элементов матрицы |

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 – целые числа (так как могут быть кратными 3), вводятся пользователем с клавиатуры. Необходимо осуществить проверку ввода целого 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(ы) остановки.

Примеры: «Построены первые Н элементов последовательности», «Найдены первы М элементов, кратные трем».

Алгоритм решаемой задачи.

Программа считывает входные данные с консоли. Создаются целочисленные переменные n (считает количество выведенных элементов) и m (считает количество выведенных элементов кратных трем). Запускается рекурсивная функция. Новое значение элемента вычисляется по формуле ак = 2 \* | ак–1 – ак-2 | + ак–3 и значение выводится на экран. Изменяется n и m по необходимости, переменные проверяются на достижение значений N и M. Если ни одно из условий выхода не выполнено, вызывается рекурсивная функция от последних трех элементов последовательности, включая новый. Иначе, выводится сообщение о причине остановки и происходит выход из функции.



Процесс реализации системы.

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

Рекурсивная функция Rec(int a1, int a2, int a3) принимает три аргумента и не возвращает значений. Перед первым запуском функции также осуществляется проверка выполнения условий выхода.

Листинг программы представлен в репозитории и ниже.

Листинг программы:

class Program

{

static int M, N, m, n;

public static void Main(string[] args)

{

Console.WriteLine("Задание 6");

//print("Введите а1 а2 а3 М, N через пробел (целые числа, М, N - натуральные)");

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

//M = Convert.ToInt32(s[3]); N = Convert.ToInt32(s[4]);

M = VVODN("M"); N = VVODN("N");

m = 0; n = 3;

int a1 = VVOD("a1"), a2 = VVOD("a2"), a3 = VVOD("a3");

Console.WriteLine("Последовательность:");

p(a1); p(a2); p(a3);

if (a1 % 3 == 0) m++;

if (a2 % 3 == 0) m++;

if (a3 % 3 == 0) m++;

//Rec(Convert.ToInt32(s[0]), Convert.ToInt32(s[1]), Convert.ToInt32(s[2]));

if (n >= N)

print("Построены первые N элементов последовательности");

if (m >= M)

print("Найдены первы M элементов, кратные трем.");

if (m < M && n < N)

Rec(a1, a2, a3);

Console.ReadLine();

}

static void Rec(int a1, int a2, int a3)

{

int ak = 2 \* Math.Abs(a1 - a2) + a3;

p(ak); n++;

if (ak % 3 == 0) m++;

if (n >= N)

print("Построены первые Н элементов последовательности");

if (m >= M)

print("Найдены первы М элементов, кратные трем.");

if (m < M && n < N)

Rec(ak, a1, a2);

}

static int VVODN(string s)

{

int value;

bool rightValue;

Console.WriteLine("Введите натуральное число " + s);

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;

}

static int VVOD(string s)

{

int value;

bool rightValue;

Console.WriteLine("Введите целое число " + s);

do

{

string inputValue = Console.ReadLine();

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

if (!rightValue)

{

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

}

}

while (!rightValue || value < 1);

return value;

}

static void p(int a)

{

Console.Write(a + " ");

}

static void print(string a)

{

Console.WriteLine();

Console.WriteLine(a);

}

}

Описание результатов тестирования системы

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. При n <= 3 или при m кратных 3 элементов среди первых трех элементов последовательности, программа не заходит в рекурсивную функцию.

Тестирование выходных данных: выполнения одного/обоих условий. Проверка каждого из возможных вариантов вывода. Также программа должна выводить сообщение об ошибке, если входные данные некорректны.

Тестирование функций: VVOD(), VVODN(), Rec(int a1, int a2, int a3).

Основные сценарии тестирования представлены в таблице ниже.

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
| Номер теста | Входные данные | Ожидаемый результат | Реальный результат | Критерий (сценарий) тестирования |
| 1 | n = -1 | Неверное значение, ожидалось натуральное число. повторите. | + | Тестирование входных данных, функции VVODN().  Некорректный n |
| 2 | a2 = 5/5 | Неверное значение, ожидалось целое число. повторите. | + | Тестирование входных данных, функции VVOD()  Некорректный элемент |
| 3 | n = 3  m = 50  a1 = 1  a2 = 0  a3 = 5 | 1 0 5  Построены первые Н элементов последовательности | + | Тестирование входных данных, выходных данных, функции VVOD(),функции VVODN().  Выполнение одного условия, рекурсия не выполняется |
| 4 | n = 10  m = 4  a1 = 1  a2 = 2  a3 = 3 | 1 2 3 5 10 7 18 33 37  Построены первые Н элементов последовательности | + | Тестирование входных данных, выходных данных, функции VVOD(),функции VVODN(),Rec  Выполнение одного условия |
| 5 | n = 10  m = 2  a1 = 1  a2 = 3  a3 = 5 | 1 3 5 9  Найдены первые М элементов, кратные трем. | + | Тестирование входных данных, выходных данных, функции VVOD(),функции VVODN(),Rec  Выполнение одного условия |
| 6 | n = 3  m = 2  a1 = 1  a2 = 3  a3 = 9 | 1 3 9  Построены первые Н элементов последовательности  Найдены первы М элементов, кратные трем. | + | Тестирование входных данных, выходных данных, функции VVOD(),функции VVODN().  Выполнение обоих условий, рекурсия не выполняется |

Задание 7.

Постановка задачи:

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) – подстрок (string), расположенных в лексикографическом порядке. Либо сообщение об ошибке.

Алгоритм решаемой задачи.

Прежде всего, по данному набору частот **найдем длины кодовых слов** кода Хаффмана. Для этого запишем в список (или иную структуру) исходные частоты в порядке их убывания и будем преобразовывать этот список, на каждом шаге заменяя две самых маленьких частоты их суммой и располагая в новом списке полученные частоты в порядке убывания. Процесс преобразования частот завершится, как только мы придем к списку с двумя частотами (их сумма равна единице).

Теперь **построим бинарное дерево**, корень которого располагается в самом верхнем ярусе, а потомки – частоты из последнего списка. Каждая вершина, кроме корня, помечается числом из набора списков. Построение дерева и расстановка пометок его вершин выполняется рекурсивно. Сначала к единственной вершине- корню дерева добавляются две смежные с ней вершины, которые помечаются числами из последнего списка. Рекурсивный шаг состоит в следующем: к вершине, сложенной из двух других, добавляются они в качестве потомков. Они помечаются частотами из предыдущего списка, которые в сумме дают число из этой выделенной вершины. рекурсивный шаг выполняется многократно и всякий раз применяется к вершине, сложенной из других (то есть с частотой, не являющейся начальной). Таким образом, получаем дерево. Количество «родителей» каждой частоты определяет длину кодового слова.

Каждая вершина дерева кроме частоты **определяется кодовым словом** Хаффмана, они получаются методом, приведенном ниже. Корень дерева помечается пустым словом, все остальные вершины определены словами в алфавите {0, 1}. Принцип их образования следующий: при переходе от вершины к потомкам к слову вершины добавляется 0 и новое слово присваивается левому потомку и 1 при переходе к правому потомку. «Конечные» вершины с заданными частотами не являются началом никаких других, это гарантирует уникальность полученных слов и делает код префиксным.

Блок-схема алгоритма представлена 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). Если сумма введенных частот не равна 1, программа выдаст сообщение об ошибке.

Определение длины кодовых слов и построение дерева происходит одновременно. Каждый элемент представляет и себя корень, содержащий частоту, информационное поле, в котором будет записано кодовое слово, и двух потомков такого же типа. Ниже представлен листинг класса элементов Tree.

class Tree

{

public Tree left = null;

public Tree right = null;

public double frequency;

public string key = "";

public Tree(double f)

{

frequency = f;

}

}

Сначала в список добавляются все частоты. Потом над двумя наименьшими надстраивается новый элемент и добавляется в список вместо них. Действия повторяются, пока в списке не останется всего 1 элемент. После чего согласно алгоритму заполняются информационные поля элементов дерева. Слова ключевых элементов добавляются в список слов, который впоследствии сортируется и выводится на экран.

Листинг программы представлен в репозитории и ниже.

Листинг:

static List<Tree> l = new List<Tree>();

static List<string> s = new List<string>();

public static void Main(string[] args)

{

Console.WriteLine("Задание 7");

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

int n = VVODN(); // проверка на 0 и на отрицательные плюс проверка ввода

Console.WriteLine("Поочередно введите все элементы в формате 0,0...");

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

l.Add(new Tree(VVOD(i))); // читаем и сразу добавляем

while (l.Count > 1)

{

Tree min1 = Min(), min2 = Min(); // определяются две наименьшие частоты в списке и удаление элементов из списка

Tree elem = new Tree(min1.frequency + min2.frequency); // над наименьшими элементами надстраивается новый

elem.left = min1; elem.right = min2;

l.Add(elem);

}

Tree head = l[0];

if (head.frequency != 1)

Console.WriteLine("Программа вынуждена завершить работу в связи с тем, что сумма всех частот не равна 1");

else

{

MakeKey(ref head);

s.Sort();

Console.WriteLine();

Console.WriteLine("Полученные слова Хаффмана:");

foreach (string str in s)

Console.Write(str + " ");

}

Console.ReadLine();

}

static Tree Min()

{

Tree min = l[0];// необходимо ли здесь глубинное копирование?

foreach (Tree x in l)

if (x.frequency < min.frequency)

min = x; // ????????????????

l.Remove(min);

return min;

}

static void MakeKey(ref Tree x) // определение кода числа и добавление его в список слов

{

if (x != null)

{

if (x.left != null)

{

x.left.key = x.key + "1";

MakeKey(ref x.left);

}

if (x.right != null)

{

x.right.key = x.key + "0";

MakeKey(ref x.right);

}

if (x.left == null && x.right == null)

s.Add(x.key);

}

}

static double VVOD(int k)

{

double value;

bool rightValue;

Console.WriteLine("Введите частоту элемента " + (k + 1));

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

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. Если сумма введенных частот меньше или больше 1, программа работать не будет и сообщит об ошибке.

Тестирование выходных данных: программа возвращает полученный префиксный код в лексикографическом порядке. Также программа должна выводить сообщение об ошибке, если входные данные некорректны или сумма частот не 1.

Тестирование функций: VVOD(), VVODN(), MakeKey(ref Tree x), Min().

Основные сценарии тестирования представлены в таблице ниже.

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
| Номер теста | Входные данные | Ожидаемый результат | Реальный результат | Критерий (сценарий) тестирования |
| 1 | n = -1 | Неверное значение, ожидалось натуральное число. повторите. | + | Тестирование входных данных, функции VVODN().  Некорректный n |
| 2 | n = 3  0.5 0,2 0,3 | Неверное значение для дробного числа вида 0,0 , повторите. | + | Тестирование входных данных, функции VVOD()  Некорректный элемент |
| 3 | n = 3  0,7 0,2 0,3 | Программа вынуждена завершить работу в связи с тем, что сумма всех частот не равна 1 | + | Тестирование входных данных, выходных данных, функции VVOD(),функции VVODN()  сумма частот не 1 |
| 4 | n =5  0,4 0,25 0,2 0,1 0,05 | Полученные слова Хаффмана:  000 0010 0011 01 1 | + | Тестирование входных данных, выходных данных, функции VVOD(),функции VVODN(), MakeKey(ref Tree x), Min().  Корректная работа программы |

Задание 8. Не доделана

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.

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:

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.

Либо автоматическая генерация вещественных элементов последовательности в диапазоне от -100 до 101.

Формат выходных данных:

Вывод двух сумм: всех положительных и всех отрицательных значений, записанных в информационные поля элементов списка. Суммы типа double и комментарии к ним – строки.

Алгоритм решаемой задачи.

Создается класс, который реализует односторонний циклический список, в нем создаются постоянные переменные, запоминающие кол-во всех элементов, сумму положительных и сумму отрицательных. Переменные изменяются при создании нового элемента класса в зависимости от его значения. После окончания ввода (генерации) элементов, выводятся значения сумм.

Алгоритм добавления нового элемента 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). Либо элементы последовательности генерируются автоматически. Пользователю предоставлен выбор.

Первый элемент списка ссылается сам на себя, впоследствии последний элемент будет замыкаться на первый. При добавлении каждого нового элемента, проверяется его знак и по необходимости увеличивается (уменьшается) переменная, считающая сумму положительных (отрицательных) элементов.

Ниже представлен листинг класса Mas, реализующего циклический список.

class Mas

{

public Mas next = null;

public double elem;

public Mas(double a)

{

elem = a;

next = this;

kol++;

if (a > 0) GoodElem++;

if (a < 0) BadElem++;

}

public Mas(double a, Mas b)

{

elem = a;

next = b.next;

b.next = this;

kol++;

if (a > 0) GoodElem++;

if (a < 0) BadElem++;

}

public static int kol = 0, GoodElem = 0, BadElem = 0;

}

Листинг программы представлен в репозитории и ниже.

Листинг программы:

class MainClass

{

//В программе построен циклический список.

//Напишите процедуру подсчета двух сумм: всех положительных и всех отрицательных значений,

//записанных в информационные поля элементов списка.

public static void Main(string[] args)

{

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

int n = VVODN();

Console.WriteLine("Если вы хотите ввести элементы вручную, нажмите 1. В противном случае элементы будут сгенерированы автоматически.");

if (Console.ReadLine() == "1")

{

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

Mas a = new Mas(VVOD());

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

{

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

a = new Mas(VVOD(), a);

}

}

else

{

Random rand = new Random();

Mas a = new Mas(rand.NextDouble() + rand.Next(-100, 100));

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

a = new Mas(rand.NextDouble() + rand.Next(-100, 100), a);

Console.WriteLine("Сгенерированные элементы:");

for (int i = 0; i < Mas.kol; i++)

{

Console.Write(a.elem + " ");

a = a.next;

}

}

Console.WriteLine();

Console.WriteLine("Сумма положительных элементов: " + Mas.GoodElem);

Console.WriteLine("Сумма отрицательных элементов: " + Mas.BadElem);

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

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 возвращает две суммы. Также программа должна выводить сообщение об ошибке, если входные данные некорректны. Корректная работа программы при генерации элементов. Форматированный вывод.

Тестирование алгоритма: одна(обе) из сумм равна 0 (нет положительных и/или отрицательных элементов)

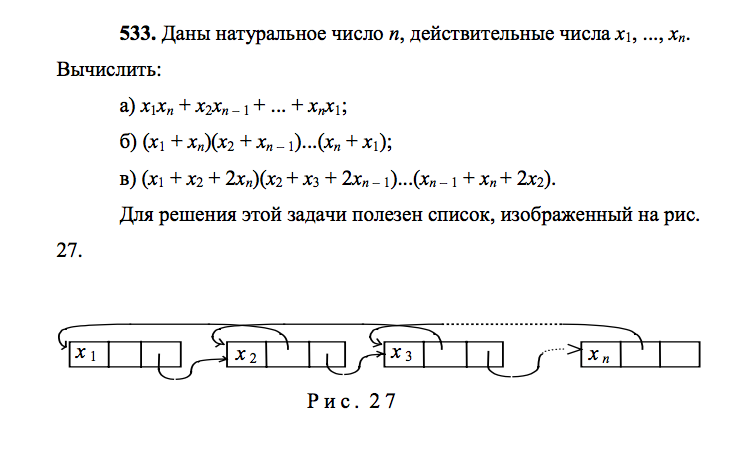
Тестирование функций: VVOD(), VVODN().

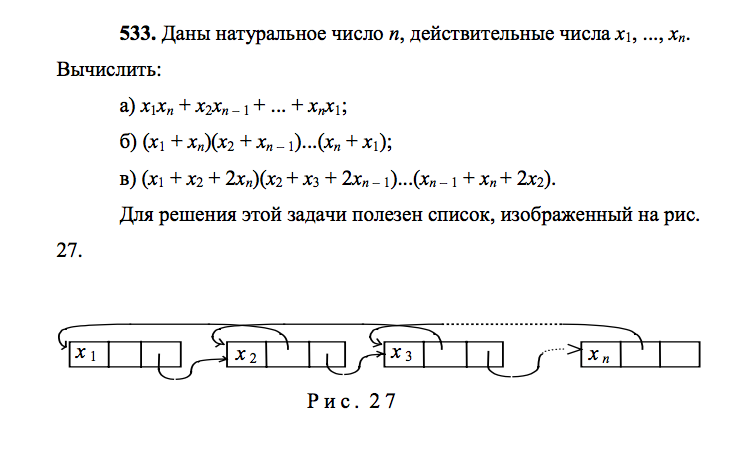
Основные сценарии тестирования представлены в таблице ниже.

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
| Номер теста | Входные данные | Ожидаемый результат | Реальный результат | Критерий (сценарий) тестирования |
| 1 | n = -1 | Неверное значение, ожидалось натуральное число. повторите. | + | Тестирование входных данных, функции VVODN().  Некорректный n |
| 2 | n = 3  0.5 0,2 0,3 | Неверное значение для дробного числа вида 0,0 , повторите. | + | Тестирование входных данных, функции VVOD()  Некорректный элемент |
| 3 | n = 3  0,7 -0,2 0,3 | 1  -0,2 | + | Тестирование входных данных, выходных данных, функции VVOD(),функции VVODN(), тестирование алгоритма  Обе суммы отличны от 0 |
| 4 | n = 2  0  -0,7 | 0  -0,7 | + | Тестирование входных данных, выходных данных, функции VVOD(),функции VVODN(), тестирование алгоритма  Нет положительных элементов |
| 5 | n = 3  0 1  0,9 | 1,9  0 | + | Тестирование входных данных, выходных данных, функции VVOD(),функции VVODN(), тестирование алгоритма  Нет отрицательных элементов |
| 5 | n = 1  0 | 0  0 | + | Тестирование входных данных, выходных данных, функции VVOD(),функции VVODN()  Все элементы 0 |
| 6 | n = 10 | 174,95  -42,94 | + | Тестирование входных данных, выходных данных, функции VVOD(),функции VVODN()  Генерация элементов и форматированный вывод |

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

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 возвращает полученный результат. Также программа должна выводить сообщение об ошибке, если входные данные некорректны.

Тестирование алгоритма: элементов четное/нечетное число (при нечетном один из элементов складывается сам с собой). Один из множителей равен 0. Есть отрицательные множители.

Тестирование функций: VVOD(), VVODN().

Основные сценарии тестирования представлены в таблице ниже.

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
| Номер теста | Входные данные | Ожидаемый результат | Реальный результат | Критерий (сценарий) тестирования |
| 1 | n = -1 | Неверное значение, ожидалось натуральное число. повторите. | + | Тестирование входных данных, функции VVODN().  Некорректный n |
| 2 | n = 3  0.5 0,2 0,3 | Неверное значение для дробного числа вида 0,0 , повторите. | + | Тестирование входных данных, функции VVOD()  Некорректный элемент |
| 3 | n = 3  0,7 0,2 0,3 | 0,4 | + | Тестирование входных данных, выходных данных, функции VVOD(),функции VVODN()  Нечетное число элементов |
| 4 | n = 3  0,7 0,2  -0,7 | 0 | + | Тестирование входных данных, выходных данных, функции VVOD(),функции VVODN()  Нечетное число элементов, один из множителей 0 |
| 5 | n = 3  0 1  -0,9 | 0,81 | + | Тестирование входных данных, выходных данных, функции VVOD(),функции VVODN()  Нечетное число элементов, один из множителей отрицательный |
| 5 | n = 3  0 -1  0,9 | -0,81 | + | Тестирование входных данных, выходных данных, функции VVOD(),функции VVODN()  Нечетное число элементов, средний множитель отрицательный |

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

зашифровать данный текст;

расшифровать данный текст.

Формат входных данных:

Входные данные: строка, содержащая текст для шифрования. Программа принимает символы русского алфавита, английского алфавита и наиболее популярные знаки (></?\\*! и т. д.).

Формат выходных данных:

Строки. Программа возвращает зашифрованное слово и расшифрованное из него слово с комментариями.

Алгоритм решаемой задачи.

Программа считывает входные данные с консоли. Для шифрования и дешифрования создается два словаря: прямой и обратный. Каждый символ заменяется своим четырехзначным номером (кодом символа). В словаре шифрования учтены заглавные и строчные буквы английского и русского алфавита, пробел, цифры и основные знаки. После чего выводится на экран. Новая строка делится на четырех символьные подстроки и дешифруется в обратную стороны. Вывод осуществляется повторно. Алгоритм представлен в блок-схеме ниже.



Процесс реализации системы.

Функция Tables() заполняет оба словаря: From – прямой, To – обратный. Прямой словарь формируется из кодов символов, обратный создается из прямого словаря.

Функция Encryption(string s) осуществляет шифрование. Если код символа имеет меньше четырех разрядов, то полученная строка дополняется нулями слева до четырех символов.

Функция Decryption(string s) осуществляет дешифрование закодированной строки, которая делится на подстроки по 4 цифры преобразуется в символы.

Если был введен не существующий символ в словаре, он кодируется как 0000 и выводится как §.

Листинг программы представлен в репозитории и ниже .

Листинг:

static Dictionary<Char, string> From = new Dictionary<char, string> { };

static Dictionary<string, Char> To = new Dictionary<string, Char> { };

static bool Check1 = true, Check2 = true;

static void Main(string[] args)

{

// Заполнение таблиц

Tables();

string word = "Полина";

Console.WriteLine("Пожалуйста, введите строку, которую необходимо зашифровать");

word = Console.ReadLine();

// шифрование

string answer = Encryption(word);

Console.WriteLine("Шифрование:");

if (!Check1) Console.WriteLine("ВНИМАНИЕ! При шифровании были найдены символы, которых нет в таблице. Они будут зашифрованы как 0000");

Console.WriteLine(answer);

// дешифрование

Console.WriteLine("Дешифрование:");

if(!Check2) Console.WriteLine("ВНИМАНИЕ! При дешифровании были найдены шифры, которых нет в таблице. Они будут расшифрованы как §");

Console.WriteLine(Decryption(answer));

Console.ReadLine();

}

static void Tables() // Заполнение таблиц

{

// Заполнение таблицы шифрования

for (int i = 32; i < 123; i++) // английский алфавит и символы (цифры)

From.Add((Char)i, String.Format("{0,0:D4}", i));

for (int i = 1040; i < 1104; i++) // русский алфавит

From.Add((Char)i, String.Format("{0,0:D4}", i));

// Заполнение таблицы дешифрования

foreach (var x in From)

To.Add(x.Value, x.Key);

}

static string Decryption(string wordto) // дешифрование

{

string word = "";

int size = 0; string subs = "";

foreach (Char c in wordto)

{

size++; subs = subs + c;

if (size == 4)

{

if (To.ContainsKey(subs))

word = word + To[subs];

else // если код не найден

{

Check2 = false;

word = word + "§";

}

size = 0; subs = "";

}

}

if (size != 0)

{

if (To.ContainsKey(subs))

word = word + To[subs];

else // если код не найден

{

Check2 = false;

word = word + "§";

}

}

return word;

}

static string Encryption(string word) // шифрование

{

string answer = "";

foreach (Char s in word)

{

if (From.ContainsKey(s))

answer = answer + From[s];

else // если символ не найден

{

Check1 = false;

answer = answer + "0000";

}

}

return answer;

}

Описание результатов тестирования системы

Тестирование проведено по критериям черного ящика.

Тестирование входных данных: корректные/некорректные. Если какой-то из введенных символов отсутствует в словаре, его нужно заменить на специальный и вывести пользователю предупреждение.

Тестирование выходных данных: стандартный вывод/ вывод со специальным символом и комментарием. Если какой-то из введенных символов отсутствует в словаре, его нужно заменить на специальный и вывести пользователю предупреждение (например, символ Ё).

Тестирование функций: Tables(), Encryption(string s), Decryption(string s).

Основные сценарии тестирования представлены в таблице ниже.

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
| Номер теста | Входные данные | Ожидаемый результат | Реальный результат | Критерий (сценарий) тестирования |
| 1 | Полина | 105510861083108010851072  Полина | + | Тестирование входных данных, выходных данных, функции Tables(), Encryption(string s), Decryption(string s).  Строка полностью преобразуется |
| 2 | ПолинаЁ | 1055108610831080108510720000  Полина§ | + | Тестирование входных данных, выходных данных, функции Tables(), Encryption(string s), Decryption(string s).  Строка неполностью преобразуется |

Задание 12. Не доделана

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.

Заключение

Подводя итоги, были спроектированы, разработаны и протестированы двенадцать задач; развиты и закреплены практические навыки построения и описания алгоритмов для решения задач из разных предметных областей (численные методы, дискретная математика, структуры данных и др). Были развиты и закреплены практические навыки использования языков высокого уровня и современных сред разработки для реализации построенных алгоритмов. Были развиты и закреплены практические навыки объектно-ориентированного программирования. Были развиты практические навыки оформления отчетов о проделанной работе и публичного выступления с защитой проекта. По окончанию работы получены приложения выполняющие поставленные задачи. Таким образом все задачи были выполнены, и как следствие – цель достигнута.

Библиографический список

1. Мацяшек Л.А. Практическая программная инженерия на основе учебного примера: пер. с англ. – М.: БИНОМ. Лаборатория знаний, 2011. — 948 с.
2. Терехов А.Н. Технология программирования: учебное пособие. / А.Н. Терехов: – 2-е изд. – М.: БИНОМ. Лаборатория знаний, 2012. — 149 с.
3. Троелсен Э. Язык программирования C# 2010 и платформа .NET 4: пер. с англ. – М.: Издательский дом "Вильямс", 2012. — 1392 c.
4. Абрамов С.А., Гнездилова Г.Г., Капустина Е.Н., Селюн М.И. Задачи по программированию. М.: Наука, 1988. — 569 c.
5. Плаксин М.А., Тестирование и отладка программ для профессионалов будущих и настоящих / М.А. Плаксин — М.: Бином. Лаборатория знаний, 2007. — 169 c.