Министерство науки и высшего образования РФ

ФГАОУ ВО Пермский национальный исследовательский

политехнический университет

Кафедра «Вычислительная математика, механика и биомеханика»

Отчет по лабораторной работе № 8

тема «Массивы и матрицы»

по дисциплине «Информатика»

Выполнил: студент группы ИСТ-21-1б Евсеев В.С.

Проверил: доцент каф. ВММБ Каменских А.А.

Пермь, 2022

Содержание

[Задание 1 4](#_Toc99279290)

[1.1. Общая постановка задачи 4](#_Toc99279291)

[1.2. Решение задачи, код программы 4](#_Toc99279292)

[1.3. Тестирование программы с проверкой 6](#_Toc99279293)

[Задание 2 9](#_Toc99279294)

[2.1. Общая постановка задачи 9](#_Toc99279295)

[2.2. Решение задачи, код программы 9](#_Toc99279296)

[2.3. Тестирование программы с проверкой 9](#_Toc99279297)

[Задание 3 11](#_Toc99279298)

[3.1. Общая постановка задачи 11](#_Toc99279299)

[3.2. Решение задачи, код программы 11](#_Toc99279300)

[3.3. Тестирование работы программы с проверкой 12](#_Toc99279301)

[Задание 4 15](#_Toc99279302)

[4.1. Общая постановка задачи 15](#_Toc99279303)

[4.2. Решение задачи, код программы 15](#_Toc99279304)

[4.3. Тестирование работы программы с проверкой 16](#_Toc99279305)

[Задание 5 19](#_Toc99279306)

[5.1. Общая постановка задачи 19](#_Toc99279307)

[5.2. Решение задачи, код программы 19](#_Toc99279308)

[5.3. Тестирование программы с проверкой 20](#_Toc99279309)

[Задание 6 23](#_Toc99279310)

[6.1. Общая постановка задачи 23](#_Toc99279311)

[6.2. Решение задачи, код программы 23](#_Toc99279312)

[6.3. Тестирование программы с проверкой 24](#_Toc99279313)

[Задание 7 26](#_Toc99279314)

[7.1. Общая постановка задачи 26](#_Toc99279315)

[7.2. Решение задачи, код программы 26](#_Toc99279316)

[7.3. Тестирование программы с проверкой 27](#_Toc99279317)

[Задание 8 29](#_Toc99279318)

[8.1. Общая постановка задачи 29](#_Toc99279319)

[8.2. Решение задачи, код программы 29](#_Toc99279320)

[8.3. Тестирование программы с проверкой 30](#_Toc99279321)

[Задание 9 33](#_Toc99279322)

[9.1. Общая постановка задачи 33](#_Toc99279323)

[9.2. Решение задачи, код программы 33](#_Toc99279324)

[9.3. Тестирование программы с проверкой 34](#_Toc99279325)

[Задание 10 38](#_Toc99279326)

[10.1. Общая постановка задачи 38](#_Toc99279327)

[10.2. Решение задачи, код программы 38](#_Toc99279328)

[10.3. Тестирование программы с проверкой 40](#_Toc99279329)

[Задание 12 43](#_Toc99279330)

[12.1. Общая постановка задачи 43](#_Toc99279331)

[12.2. Решение задачи, код программы 43](#_Toc99279332)

[12.3. Тестирование программы с проверкой 44](#_Toc99279333)

# Задание 1

## 1.1. Общая постановка задачи

Ввести массив, состоящий из N элементов целого типа. Массив является элементами целого дробной части числа в троичной системе счисления. Проверить правильность введения элементов числа, т.е. в массиве должны быть только 0, 1 и 2. Считается, что в массиве ведены только те цифры, что стоят после запятой. Если число введено верно, то перевести его в десятичную систему счисления и поэлементно записать число в массив, так же только цифры после запятой (ограничиваясь 6 цифрами после запятой, если число при переводе не конечно). Пример:

|  |  |
| --- | --- |
| В 3-ой | В 10-ой |
| |  |  | | --- | --- | | 2 | 1 | | |  |  |  |  |  |  | | --- | --- | --- | --- | --- | --- | | 7 | 7 | 7 | 7 | 7 | 7 | |
| 0,21 | 0, 777777....... |

## 1.2. Решение задачи, код программы

**import** java.util.Scanner;  
  
**import static** java.lang.System.***out***;  
  
**public class** Task81 {  
  
 **static** Scanner *scanner* = **new** Scanner(System.***in***);  
  
 **public static void** main(String[] args) {  
 **try** {  
 Scanner scannerAns = **new** Scanner(System.***in***);  
 ***out***.println(**"Если вы желаете перевести целое число, введите true."** +  
 **"\nЕсли ваш выбор пал на дробное число, введите false."**);  
 **boolean** ans = scannerAns.nextBoolean();  
 *basic*(ans);  
 }  
 **catch** (Exception e) {  
 ***out***.println(**"Ошибка!"**);  
 *main*(args);  
 }  
 }  
  
 **public static void** basic(**boolean** ans) {  
 ***out***.println(**"Введите размерность массива N:"**);  
 **int** N = *scanner*.nextInt();  
 ***out***.println(**"Введите основание системы счисления base:"**);  
 **int** base = *scanner*.nextInt();  
 **while** (base < 2 || base > 9) {  
 ***out***.println(**"Вы ввели не то, 2 <= base <= 9"**);  
 ***out***.println(**"Введите еще раз"**);  
 base = *scanner*.nextInt();  
 }  
 **int**[] A = **new int**[N];  
 ***out***.println(**"Введите элементы массива, как компоненты "** + base + **"-ого числа"**);  
 ***out***.println(**"Это цифры от "** + 0 + **" до "** + (base - 1) + **":"**);  
 **if** (ans) {  
 **int** tempInteger = 0;  
 *masInteger*(N, base, A, tempInteger);  
 }  
 **else** {  
 **double** tempFloat = 0;  
 *masFloat*(N, base, A, tempFloat);  
 }  
 }  
  
 **public static void** masInteger(**int** N, **int** base, **int**[] A, **int** tempInteger) {  
 **for** (**int** i = 0; i < N; i++) {  
 A[i] = *inputInteger*(i, N, base);  
 }  
 ***out***.println();  
 ***out***.println(**"Массив "** + base + **"-ой формы числа имеет вид:"**);  
 *printMas*(A);  
 **for** (**int** i = N - 1, j = 0; i >= 0; i--, j++) {  
 tempInteger += A[i] \* Math.*pow*(base, j);  
 }  
 ***out***.println(**""**);  
 ***out***.println(**"В 10-ой системе счисления число = "** + tempInteger);  
 }  
  
 **public static void** printMas(**int**[] A) {  
 ***out***.print(**"[ "**);  
 **for** (**int** i = 0; i <= A.**length**-1; i++) {  
 ***out***.print(A[i] + **" "**);  
 }  
 ***out***.println(**"]"**);  
 }  
  
 **public static int** inputInteger(**int** i, **int** N, **int** base) {  
 ***out***.println(**"A[ "** + i + **" ] = "**);  
 **int** A = *scanner*.nextInt();  
 **if** ((A >= base || A < 0) & i > 0) {  
 ***out***.println(**"Вы ввели число не из интервала от 0 до "** + (base - 1));  
 **while** (A >= base || A < 0) {  
 ***out***.println(**"Повторите попытку A[i] от 0 до "** + (base - 1));  
 ***out***.println(**"A[ "** + i + **" ] = "**);  
 A = *scanner*.nextInt();  
 }  
 }  
 **else if** ((A >= base || A < 0) & i == 0) {  
 ***out***.println(**"Вы ввели число не из интервала от 1 до "** + (base - 1));  
 **while** (A == 0 || A >= base || A < 0) {  
 ***out***.println(**"Повторите попытку A[i] от 1 до "** + (base - 1));  
 ***out***.println(**"A[ "** + i + **" ] = "**);  
 A = *scanner*.nextInt();  
 }  
 }  
 **else if** (A == 0 & (i == 0 || i == N-1)) {  
 **while** (A == 0 || A >= base || A < 0) {  
 ***out***.println(**"Вы ввели не верно A[ "** + i + **" ], оно может быть от 1 до "** + (base - 1));  
 ***out***.println(**"A[ "** + i + **" ] = "**);  
 A = *scanner*.nextInt();  
 }  
 }  
 **return** A;  
 }  
  
 **public static void** masFloat(**int** N, **int** base, **int**[] A, **double** tempFloat) {  
 **for** (**int** i = 0; i < N; i++) {  
 A[i] = *inputFloat*(i, N, base);  
 }  
 ***out***.println();  
 ***out***.println(**"Массив "** + base + **"-ой формы числа имеет вид:"**);  
 *printMas*(A);  
 **for** (**int** i = 0, j = -1; i < N; i++, j--) {  
 tempFloat += A[i] \* Math.*pow*(base, j);  
 }  
 ***out***.println(**""**);  
 ***out***.printf(**"В 10-ой системе счисления число = %.6f"**, tempFloat);  
 }  
  
 **public static int** inputFloat(**int** i, **int** N, **int** base) {  
 ***out***.println(**"A[ "** + i + **" ] = "**);  
 **int** A = *scanner*.nextInt();  
 **if** ((A >= base || A < 0)) {  
 ***out***.println(**"Вы ввели число не из интервала от 0 до "** + (base - 1));  
 **while** (A >= base || A < 0) {  
 ***out***.println(**"Повторите попытку A[i] от 0 до "** + (base - 1));  
 ***out***.println(**"A[ "** + i + **" ] = "**);  
 A = *scanner*.nextInt();  
 }  
 }  
 **else if** (A == 0 & i == N-1) {  
 **while** (A == 0 || A >= base || A < 0) {  
 ***out***.println(**"Вы ввели не верно A[ "** + i + **" ], оно может быть от 1 до "** + (base - 1));  
 ***out***.println(**"A[ "** + i + **" ] = "**);  
 A = *scanner*.nextInt();  
 }  
 }  
 **return** A;  
 }  
}

## 1.3. Тестирование программы с проверкой

В таблице 1 представлено тестирование работы программы с проверкой решения задачи на языке Java с решением в онлайн калькуляторе.

Таблица 1

Тестирование работы программы и проверка решения задачи 1

|  |  |  |
| --- | --- | --- |
| № п/п | Решение Java | Калькулятор |
| 1 |  |  |
| 2 |  |  |
| 3 |  |  |

<https://calculatori.ru/perevod-chisel.html>

Сравнение решения задачи с использованием трех прикладных пакетов показало, что решения задачи в Java и онлайн-калькулятора совпадает. Данный факт подтверждает правильность написанного кода программы. Неопределенностей при решении задачи выявлено не было, возможно данная ситуация связанна с малым количеством проверок данных.

# Задание 2

## 2.1. Общая постановка задачи

Ввести массив, состоящий из N элементов целого типа. Найти количество элементов четных по значению, которые стоят в четных по номерам ячейках массива.

## 2.2. Решение задачи, код программы

**import static** java.lang.System.***out***;  
  
**import** java.util.Scanner;  
  
**public class** Task82 {  
 **public static void** main(String[] args) {  
 Scanner scanner = **new** Scanner(System.***in***);  
 ***out***.println(**"Введите количество элементов массива:"**);  
 **int** n = scanner.nextInt();  
 **int**[] A = **new int** [n];  
 ***out***.println(**"Введите элементы массива - целые числа:"**);  
 **for** (**int** i = 0; i < A.**length**; i++) {  
 A[i] = scanner.nextInt();  
 }  
 *checkNumber*(A);  
 }  
 **public static void** checkNumber(**int**[] A) {  
 **int** count = 0;  
 **for** (**int** i = 0; i < A.**length**; i++) {  
 **if** ((i % 2 == 0) & (A[i] % 2 == 0)) {  
 count++;  
 }  
 }  
 **if**(count == 0) {  
 ***out***.println(**"Четных элементов, которые стоят на четных по номеру ячейках, не найдено"**);  
 }  
 **else** {  
 ***out***.println(**"Количество четных элементов, которые стоят на четных по номерам ячейках, равно "** + count);  
 }  
 }  
}

## 2.3. Тестирование программы с проверкой

Для проверки решения в MS Excel создана таблица, в которой в столбец A, начиная с ячейки A2, вписаны номера элементов массива, аналогично для столбца B вписаны элементы массива, в ячейку E2 вписана формула проверки четности номера элемента и самого элемента, в ячейку F2 вписана формула для подсчета четных элементов, которые находятся в четных по номеру ячейках массива.

E2=ЕСЛИ(И((ОСТАТ(A2;2)=0); (ОСТАТ(B2;2)=0));1;0)

F2=СУММ(E:E)

На рис.1 представлен вид решения в MS Excel.

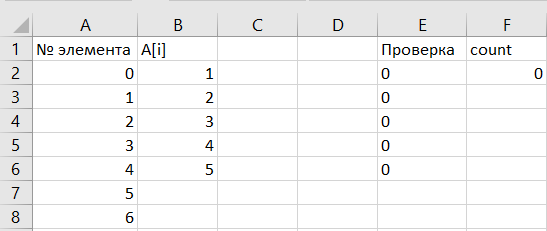


Рисунок 1. Вид решения в MS Excel задания 2

В таблице 2 представлено тестирование работы программы с проверкой решения задачи на языке Java с решением в MS Excel.

Таблица 2

Тестирование работы программы и проверка решения задачи 2

|  |  |  |
| --- | --- | --- |
| № п/п | Решение Java | Решение MS Excel |
| 1 |  |  |
| 2 |  |  |
| 3 |  |  |

Сравнение решения задачи с использованием трех прикладных пакетов показало, что решения задачи в Java и в MS Excel совпадает. Данный факт подтверждает правильность написанного кода программы. Неопределенностей при решении задачи выявлено не было, возможно данная ситуация связанна с малым количеством проверок данных.

# Задание 3

## 3.1. Общая постановка задачи

В одномерном массиве с четным количеством элементов (2N) находятся координаты N точек плоскости. Они располагаются в следующем порядке: x1, y1, х2, y2, x3, y3, и т.д. Определить номера точек, которые могут являться вершинами квадрата.

## 3.2. Решение задачи, код программы

**import** java.util.Scanner;  
  
**import static** java.lang.System.***out***;  
  
**import static** java.lang.Math.\*;  
  
**public class** Task83 {  
 **public static void** main(String[] args) {  
 Scanner scanner = **new** Scanner(System.***in***);  
 ***out***.println(**"Введите размерность массива:"**);  
 **int** n = scanner.nextInt();  
 **if** ((n % 2 != 0) | (n < 8) | (n == 0))  
 **throw new** IllegalArgumentException(**"Вы должны задать для массива четное количество точек, >= 8"**);  
 ***out***.println(**"Введите координаты x1, y1, х2, y2, x3, y3, и т.д"**);  
 **double**[] A = **new double**[n];  
 **for** (**int** i = 0; i < n; i++) {  
 A[i] = scanner.nextDouble();  
 }  
 *coordinates*(A);  
 }  
  
 **public static void** coordinates(**double**[] A) {  
 **double**[] B = **new double**[8]; *//новый массив,чтобы записать туда первые 8 элементов* **for** (**int** i = 0; i < A.**length** - 6; i += 2) {  
 B[0] = A[i];  
 B[1] = A[i + 1];  
 **for** (**int** j = i + 2; j < A.**length** - 4; j += 2) {  
 B[2] = A[j];  
 B[3] = A[j + 1];  
 **for** (**int** k = j + 2; k < A.**length** - 2; k += 2) {  
 B[4] = A[k];  
 B[5] = A[k + 1];  
 **for** (**int** l = k + 2; l < A.**length**; l += 2) {  
 B[6] = A[l];  
 B[7] = A[l + 1];  
 **if** (*isSquare*(B)) {  
 **for** (**int** coord = 0; coord < B.**length**; coord++)  
 ***out***.print(coord + **" "**);  
 ***out***.println(**""**);  
 ***out***.println(**"Номера точек, которые могут являться вершинами квадрата, представлены выше"**);  
 }  
 **else  
 *out***.println(**"Точек не найдено"**);  
 }  
 }  
 }  
 }  
 }  
 **public static boolean** isSquare(**double**[] B) {  
 **double** AB = *sqrt*((*pow*((B[0] - B[2]), 2)) + *pow*((B[1] - B[3]), 2));  
 **double** CD = *sqrt*((*pow*((B[4] - B[6]), 2)) + *pow*((B[5] - B[7]), 2));  
 **double** BC = *sqrt*((*pow*((B[2] - B[4]), 2)) + *pow*((B[3] - B[5]), 2));  
 **double** AD = *sqrt*((*pow*((B[0] - B[6]), 2)) + *pow*((B[1] - B[7]), 2));  
 **return** ((AB == CD) & (AB == BC) & (AB == AD));  
 }  
}

## 3.3. Тестирование работы программы с проверкой

В таблице 3 представлено тестирование работы программы с проверкой решения задачи на языке Java.

Таблица 3

Тестирование работы программы в Java

|  |  |
| --- | --- |
| № п/п | Решение Java |
| 1 |  |
| 2 |  |
| 3 |  |

На рис.2 представлено аналитическое решение

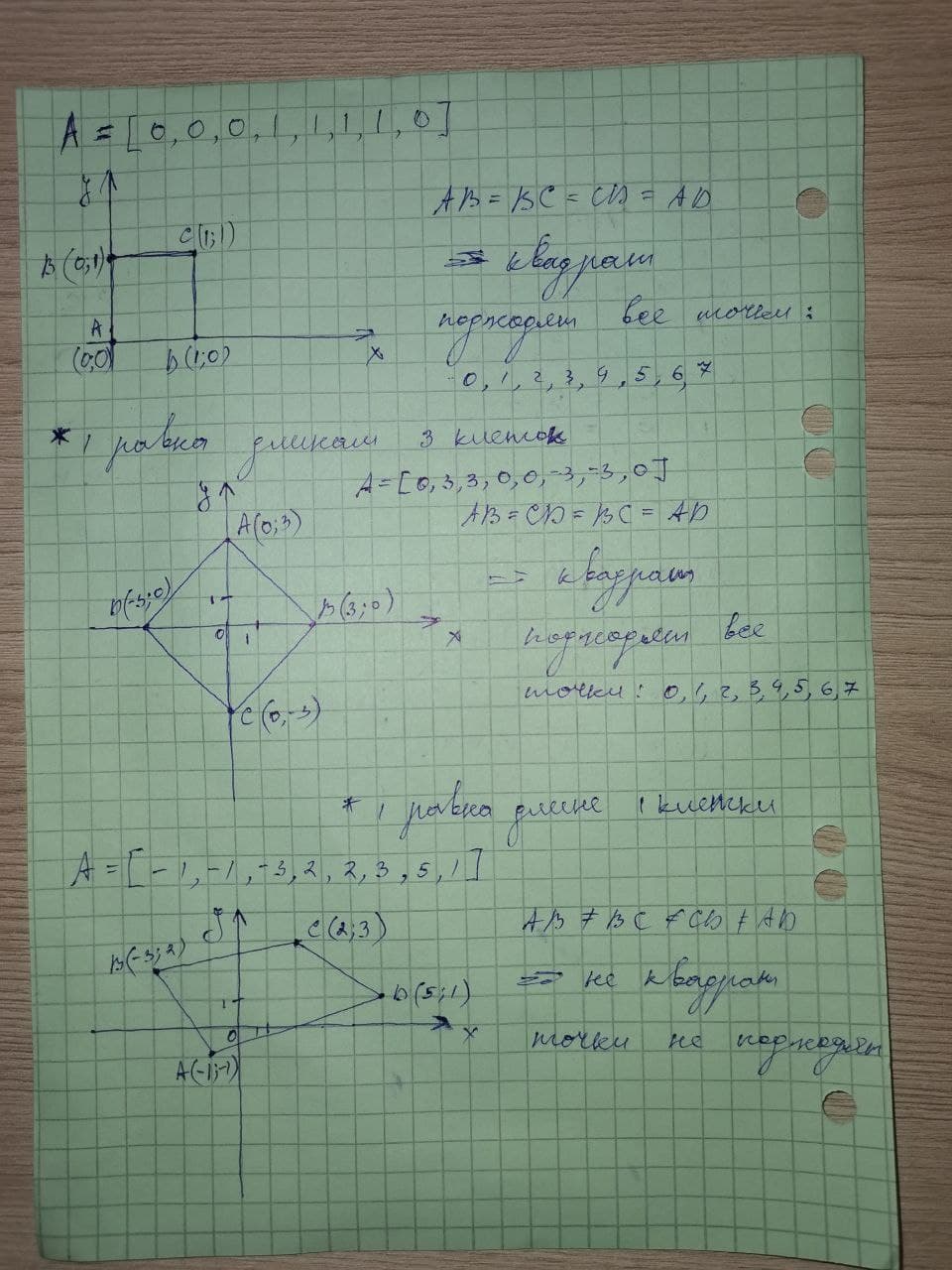


Рисунок 2. Аналитическое решение задания 3

Сравнение решения задачи с использованием трех прикладных пакетов показала, что решения задачи в Java и MS Excel совпадает. Данный факт подтверждает правильность написанного кода программы. Неопределенностей при решении задачи выявлено не было, возможно данная ситуация связанна с малым количеством проверок данных.

## Задание 4

## 4.1. Общая постановка задачи

Задать массив из произвольных чисел и реализовать алгоритм «шейкер-сортировки». Образно алгоритм можно описать так: на каждом шаге основного цикла рассматривается массив данных A[left,right], по двум внутренним циклам: 1) от left к right A[i]>=A[i+1] тогда они меняются местами, иначе ничего не происходит, после окончания цикла right = right-1; 2) от right к left A[i-1]>=A[i] тогда они меняются местами, иначе ничего не происходит, после окончания массива left = left+1. После 1 шага минимальный и максимальный элемент в исходном массиве перетекают к краям, минимальный в — A[left], максимальный – в A[right]. Остановкой итерационной процедуры является равенство 0 перестановок элементов в массиве. Выполнить сравнение «шейкер-сортировки» с сортировкой «пузырек», рассмотрев количество итераций, а так же промежуточные результаты сортировки.

## 4.2. Решение задачи, код программы

**import static** java.lang.System.***out***;  
  
**import** java.util.Scanner;  
  
**public class** Task84 {  
 **public static void** main(String[] args) {  
 Scanner scanner = **new** Scanner(System.***in***);  
 ***out***.println(**"Введите размерность массива:"**);  
 **int** n = scanner.nextInt();  
 **if** ((n == 0) | (n == 1))  
 **throw new** IllegalArgumentException(**"Пожалуйста, введите размерность, хотя бы равную 2"**);  
 **double**[] A = **new double**[n];  
 **double**[] B = **new double**[n];  
 ***out***.println(**"Введите элементы массива:"**);  
 **for** (**int** i = 0; i < n; i++) {  
 ***out***.println(**"[A"** + i + **"]="**);  
 A[i] = scanner.nextDouble();  
 B[i] = A[i];  
 }  
 *shakerSorting*(A, B);  
 }  
  
 **public static void** shakerSorting(**double**[] A, **double**[] B) {  
 **int** left = 0;  
 **int** right = A.**length** - 1;  
 **int** k = 1;  
 **int** countMas1 = 0;  
 **while** ((left < right) & (k > 0)) {  
 k = 0;  
 **for** (**int** i = left; i < right; i++) {  
 **if** (A[i] >= A[i+1]) {  
 **double** C = A[i]; *//чтобы они поменялись местами, а не оставались такими же какие и были* A[i] = A[i+1];  
 A[i+1] = C;  
 k = 1;  
 }  
 }  
 right--;  
 **for** (**int** j = right; j > left; j--) {  
 **if** (A[j-1] >= A[j]) {  
 **double** C = A[j]; *//чтобы они поменялись местами, а не оставались такими же какие и были* A[j] = A[j-1];  
 A[j-1] = C;  
 k = 1;  
 }  
 }  
 left++;  
 countMas1++;  
 **for** (**double** v : A) {  
 ***out***.print(**"\t"** + v);  
 }  
 ***out***.println();  
 }  
 ***out***.printf(**"Сортирование шейкер-сортировкой заняло итераций: %d"**, countMas1);  
 *bubbleSorting*(B);  
 }  
  
 **public static void** bubbleSorting(**double**[] B) {  
 ***out***.println();  
 **int** countMas2 = 0;  
 **for** (**int** i = B.**length** - 1; i >= 1; i--) {  
 **if** (B[i - 1] > B[i]) {  
 **double** temp= B[i - 1];  
 B[i - 1] = B[i];  
 B[i] = temp;  
 i = B.**length**;  
 countMas2++;  
 **for** (**int** j = 0; j < B.**length**; j++) {  
 **if** (j == 0)  
 ***out***.println();  
 ***out***.print(**"\t"** + B[j]);  
 }  
 }  
 }  
 ***out***.println();  
 ***out***.printf(**"Сортирование пузырьком заняло итераций: %d"**, countMas2);  
 }  
}

## 4.3. Тестирование работы программы с проверкой

В таблице 4 представлено тестирование работы программы с проверкой решения задачи на языке Java с MS Excel.

Таблица 4

Тестирование работы программы и проверка решения задания 4

|  |  |
| --- | --- |
| № п/п | Решение Java |
| 1 |  |
| 2 |  |
| 3 |  |
| № п/п | Решение MS Excel |
| 1 |  |
| 2 |  |
| 3 |  |

Сравнение решения задачи с использованием трех прикладных пакетов показала, что решения задачи в Java и MS Excel совпадает. Данный факт подтверждает правильность написанного кода программы. Неопределенностей при решении задачи выявлено не было, возможно данная ситуация связанна с малым количеством проверок данных.

# Задание 5

## 5.1. Общая постановка задачи

В заданной матрице поменять местами последний столбец и столбец, содержащий наименьший элемент матрицы. Если столбец, содержащий наименьший элемент матрицы, совпадает с последним столбцом, тогда выполнить генерацию двумерного массива еще раз и т.д. пока столбец, содержащий наименьший элемент матрицы, не будет совпадать с последним столбцом.

## 5.2. Решение задачи, код программы

**import** java.util.Scanner;  
  
**import static** java.lang.System.***out***;  
  
**public class** Task85 {  
 **public static void** main(String[] args) {  
 Scanner scanner = **new** Scanner(System.***in***);  
 ***out***.println(**"Введите размер (m x n) двумерного массива"** );  
 ***out***.println(**"Введите m:"**);  
 **int** m = scanner.nextInt();  
 ***out***.println(**"Введите n:"**);  
 **int** n = scanner.nextInt();  
 ***out***.println(**"Введите элементы матрицы:"**);  
 **double**[][] A = **new double**[m][n];  
 **for** (**int** i = 0; i < m; i++) {  
 **for** (**int** j = 0; j < n; j++) {  
 A[i][j] = scanner.nextDouble();  
 }  
 }  
 **int** count = 0;  
 *printMas*(A, m, n, count);  
 *minElement*(A, n, args, m, count);  
 }  
  
 **public static void** minElement(**double**[][] A, **int** n, String[] args, **int** m, **int** count) {  
 **double** minA = A[0][0];  
 **int** minJ = 0;  
 **for** (**int** i = 0; i < A.**length**; i++) {  
 **for** (**int** j = 0; j < A.**length**; j++) {  
 **if** (minA >= A[i][j]) {  
 minA = A[i][j];  
 minJ = j; *//столбец с наименьшим элементом* }  
 }  
 }  
 *check*(A, n, minJ, args, m, count);  
 }  
  
 **public static void** check(**double**[][] A,**int** n, **int** minJ, String[] args, **int** m, **int** count) {  
 **if** (minJ == n - 1) {  
 ***out***.println(**"Ошибка!"** +  
 **"\nНаименьший элемент массива находится в последнем столбце"**);  
 *main*(args);  
 }  
 **else** *replacement*(A, minJ, n, m, count);  
 }  
  
 **public static void** replacement(**double**[][] A, **int** minJ, **int** n, **int** m, **int** count) {  
 **for**(**int** i = 0; i < A.**length**; i++) {  
 **double** B = A[i][minJ]; *//чтобы они поменялись местами, а не оставались такими же, какие и были* A[i][minJ] = A[i][n-1];  
 A[i][n-1] = B;  
 }  
 count++;  
 *printMas*(A, m, n, count);  
 }  
  
 **public static void** printMas(**double**[][] A, **int** m, **int** n, **int** count) {  
 **if** (count == 0)  
 ***out***.println(**"Исходная матрица имеет вид:"**);  
 **else  
 *out***.println(**"Матрица после перестановки имеет вид:"**);  
 **for** (**int** i = 0; i < m; i++) {  
 **for** (**int** j = 0; j < n; j++) {  
 ***out***.print(A[i][j] + **"\t"**);  
 }  
 ***out***.println();  
 }  
 }  
}

## 5.3. Тестирование программы с проверкой

В таблице 5 представлено тестирование работы программы с проверкой решения задачи на языке Java с MS Excel.

Таблица 5

Тестирование работы программы с проверкой задания 5

|  |  |
| --- | --- |
| № п/п | Решение Java |
| 1 |  |
| 2 |  |
| 3 |  |
| № п/п | Решение MS Excel |
| 1 |  |
| 2 |  |
| 3 |  |

Сравнение решения задачи с использованием трех прикладных пакетов показала, что решения задачи в Java и MS Excel совпадает. Данный факт подтверждает правильность написанного кода программы. Неопределенностей при решении задачи выявлено не было, возможно данная ситуация связанна с малым количеством проверок данных.

# Задание 6

## 6.1. Общая постановка задачи

Пусть дана действительная матрица размером n × m. Требуется преобразовать матрицу: если в строчке есть отрицательные элементы, то поэлементно вычесть последнюю строку, если в строке нет отрицательных элементов, тогда заменить каждый четный по столбцу элемент на противоположный по знаку.

## 6.2. Решение задачи, код программы

**import static** java.lang.System.***out***;  
  
**import** java.util.Scanner;  
  
**public class** Task86 {  
 **public static void** main(String[] args) {  
 Scanner scanner = **new** Scanner(System.***in***);  
 ***out***.println(**"Введите размерность матрицы n х m"** +  
 **"\nn - число строк, m - число стлобцов"**);  
 ***out***.println(**"Введите n:"**);  
 **int** n = scanner.nextInt();  
 **if** (n < 2) {  
 ***out***.println(**"Введите размер строки хотя бы 2"**);  
 **while**(n < 2) {  
 ***out***.println(**"Повторите попытку, n должно быть >= 2"**);  
 n = scanner.nextInt();  
 }  
 }  
 ***out***.println(**"Введите m:"**);  
 **int** m = scanner.nextInt();  
 **if** (m < 2) {  
 ***out***.println(**"Введите размер столбца хотя бы 2"**);  
 **while**(m < 2) {  
 ***out***.println(**"Повторите попытку, m должно быть >= 2"**);  
 m = scanner.nextInt();  
 }  
 }  
 **double**[][] A = **new double**[n][m];  
 ***out***.println(**"Введите элементы матрицы:"**);  
 **for** (**int** i = 0; i < n; i++) {  
 **for** (**int** j = 0; j < m; j++) {  
 A[i][j] = scanner.nextDouble();  
 }  
 }  
 **int** check = 0;  
 *printMas*(A, check, n, m);  
 *replacement*(A, n, m, check);  
 }  
  
 **public static void** replacement(**double**[][] A, **int** n, **int** m, **int** check) {  
 **for** (**int** i = 0; i < n; i++) {  
 **int** count = 0;  
 **for** (**int** j = 0; j < m; j++) {  
 **if** (A[i][j] < 0) { *// проверяем есть ли отрицаетльный элемент, есть - увеличиваем count* count++;  
 }  
 }  
 **if** (count >= 1) { *// count >= 1 - вычитываем поэлементно последнюю строку* **for** (**int** h = 0; h < m; h++) {  
 A[n-1][h]--;  
 }  
 }  
 **else** { *// count == 0 - четные по номеру столбца элементы меняем на противоположные по знаку* **for** (**int** k = 0; k < n; k++ ) {  
 **if** (k % 2 == 0) {  
 A[i][k] = A[i][k] \* (-1);  
 }  
 }  
 }  
 }  
 check++;  
 *printMas*(A, check, n, m);  
 }  
  
 **public static void** printMas(**double**[][] A, **int** check, **int** n, **int** m) {  
 **if** (check == 0)  
 ***out***.println(**"Исходная матрица имеет вид:"**);  
 **else  
 *out***.println(**"Преобразованаая матрица имеет вид:"**);  
 **for** (**int** i = 0; i < n; i++) {  
 **for** (**int** j = 0; j < m; j++) {  
 ***out***.printf(**"\t%.2f"**, A[i][j]);  
 }  
 ***out***.println();  
 }  
 }  
}

## 6.3. Тестирование программы с проверкой

В таблице 6 представлено тестирование работы программы с проверкой решения задачи на языке Java с MS Excel.

Таблица 6

Тестирование программы с проверкой задания 6

|  |  |
| --- | --- |
| № п/п | Решение Java |
| 1 |  |
| 2 |  |
| 3 |  |
| № п/п | Решение MS Excel |
| 1 |  |
| 2 |  |
| 3 |  |

Сравнение решения задачи с использованием трех прикладных пакетов показала, что решения задачи в Java и MS Excel совпадает. Данный факт подтверждает правильность написанного кода программы. Неопределенностей при решении задачи выявлено не было, возможно данная ситуация связанна с малым количеством проверок данных.

# Задание 7

## 7.1. Общая постановка задачи

Заполнить двумерный массив n×n по образцу. Образец заполнения и пример:

|  |  |
| --- | --- |
| Образец заполнения массива | Пример |
|  |  |

## 7.2. Решение задачи, код программы

**import static** java.lang.System.***out***;  
  
**import** java.util.Scanner;  
  
**public class** Task87 {  
 **public static void** main(String[] args) {  
 Scanner scanner = **new** Scanner(System.***in***);  
 ***out***.println(**"Введите размерность матрицы n х n"**);  
 ***out***.println(**"Введите n:"**);  
 **int** n = scanner.nextInt();  
 **if** (n < 2) {  
 ***out***.println(**"Введите размерность хотя бы 2"**);  
 **while** (n < 2) {  
 ***out***.println(**"Повторите попытку, n должно быть > 2"**);  
 n = scanner.nextInt();  
 }  
 }  
 **double**[][] A = **new double**[n][n];  
 *permutation*(A, n);  
 }  
  
 **public static void** permutation(**double**[][] A, **double** n) {  
 **int** count = 1;  
 **for** (**int** i = 0; i < n; i++) {  
 **if** (i % 2 == 0) { *// строчка чётная - идём по диагонали сверху вниз* **for** (**int** h = 0, j = i; h <= i; h++, j--) {  
 A[h][j] = count++;  
 }  
 }  
 **else** { *// строчка нечётная - идём по диагонали снизу вверх* **for** (**int** h = i, j = 0; h >= 0; h--, j++) {  
 A[h][j] = count++;  
 }  
 }  
 }  
 *printMas*(A);  
 }  
  
 **public static void** printMas(**double**[][] A) {  
 ***out***.println(**"Преобразованная матриц имеет вид:"**);  
 **for** (**int** i = 0; i < A.**length**; i++) {  
 **for** (**int** j = 0; j < A.**length**; j++) {  
 ***out***.print(A[i][j] + **" "**);  
 }  
 ***out***.println();  
 }  
 }  
}

## 7.3. Тестирование программы с проверкой

В таблице 7 представлено тестирование работы программы с проверкой решения задачи на языке Java с MS Excel.

Таблица 7

Тестирование программы с проверкой задания 7

|  |  |
| --- | --- |
| № п/п | Решение Java |
| 1 |  |
| 2 |  |
| 3 |  |
| № п/п | Решение MS Excel |
| 1 |  |
| 2 |  |
| 3 |  |

Сравнение решения задачи с использованием трех прикладных пакетов показала, что решения задачи в Java и MS Excel совпадает. Данный факт подтверждает правильность написанного кода программы. Неопределенностей при решении задачи выявлено не было, возможно данная ситуация связанна с малым количеством проверок данных.

# Задание 8

## 8.1. Общая постановка задачи

Заполнить двумерный массив n×n по образцу. Матрица заполняется автоматически для любого n. Создать метод заполнения блока матрицы и отпараметризировав заполнение заполнить при помощи метода всю матрицу. Образец заполнения:



## 8.2. Решение задачи, код программы

**import static** java.lang.System.***out***;  
  
**import** java.util.Scanner;  
  
**public class** Task88 {  
 **public static void** main(String[] args) {  
 Scanner scanner = **new** Scanner(System.***in***);  
 ***out***.println(**"Введите размерность матрицы n х n"**);  
 ***out***.println(**"Введите n:"**);  
 **int** n = scanner.nextInt();  
 **if** (n < 2) {  
 ***out***.println(**"Введите размерность хотя бы 2"**);  
 **while** (n < 2) {  
 ***out***.println(**"Повторите попытку, n должно быть > 2"**);  
 n = scanner.nextInt();  
 }  
 }  
 **double**[][] A = **new double**[n][n];  
 **int** m = (**int**) Math.*ceil*(n/2);  
 *split*(A, m, n);  
 }  
  
 **public static void** split(**double**[][] A, **int** m, **int** n) {  
 *block*(A, m, 0, 0);  
 *block*(A, m, 0, m);  
 *block*(A, m, m, 0);  
 *block*(A, m, m, m);  
 *printMas*(A);  
 }  
  
 **public static void** block(**double**[][] A, **int** m, **int** shiftI, **int** shiftJ) {  
 **int** count = 1;  
 **for** (**int** k = 0; k <= m; k++) {  
 **for** (**int** j = k; j < m - k; j++) {  
 A[m - 1 - k + shiftI][j + shiftJ] = count++;  
 }  
 **for** (**int** i = m - 2 - k; i >= k; i--) {  
 A[i + shiftI][m - 1 - k + shiftJ] = count++;  
 }  
 **for** (**int** i = m - 2 - k; i >= k; i--) {  
 A[i + shiftI][k + shiftJ] = count++;  
 }  
 **for** (**int** j = k + 1; j <= m - 2 -k; j++) {  
 A[k + shiftI][j + shiftJ] = count++;  
 }  
 }  
 }  
  
 **public static void** printMas(**double**[][] A) {  
 ***out***.println(**"Преобразованная матрица имеет вид:"**);  
 **for** (**int** i = 0; i < A.**length**; i++) {  
 **for** (**int** j = 0; j < A.**length**; j++) {  
 ***out***.print(A[i][j] + **" "**);  
 }  
 ***out***.println();  
 }  
 }  
}

## 8.3. Тестирование программы с проверкой

В таблице 8 представлено тестирование работы программы с проверкой решения задачи на языке Java с MS Excel.

Таблица 8

Тестирование программы с проверкой задания 8

|  |  |
| --- | --- |
| № п/п | Решение Java |
| 1 |  |
| 2 |  |
| 3 |  |
| № п/п | Решение MS Excel |
| 1 |  |
| 2 |  |
| 3 |  |

Сравнение решения задачи с использованием трех прикладных пакетов показала, что решения задачи в Java и MS Excel совпадает. Данный факт подтверждает правильность написанного кода программы. Неопределенностей при решении задачи выявлено не было, возможно данная ситуация связанна с малым количеством проверок данных.

# Задание 9

## 9.1. Общая постановка задачи

Заполнить двумерный массив 2n×2n по образцу. Матрица заполняется автоматически для любого n. Создать метод заполнения блоков матрицы и отпараметризировав выполнить заполнение всей матрицы при помощи методов. Образец заполнения:



## 9.2. Решение задачи, код программы

**import static** java.lang.System.***out***;  
  
**import** java.util.Scanner;  
  
**public class** Task89 {  
 **public static void** main(String[] args) {  
 Scanner scanner = **new** Scanner(System.***in***);  
 ***out***.println(**"Введите размерность матрицы 2n х 2n"**);  
 ***out***.println(**"Введите n:"**);  
 **int** n = scanner.nextInt();  
 **if** (n < 2) {  
 ***out***.println(**"Введите размерность хотя бы 2"**);  
 **while** (n < 2) {  
 ***out***.println(**"Повторите попытку, n должно быть > 2"**);  
 n = scanner.nextInt();  
 }  
 }  
 **double**[][] A = **new double**[2 \* n][2 \* n];  
 *split*(A, n);  
 *printMas*(A);  
 }  
  
 **public static void** split(**double**[][] A, **int** n) {  
 *snake*(A, n, 0, 0);  
 *spiral*(A, n, 0, n);  
 *snake*(A, n, n, 0);  
 *spiral*(A, n, n, n);  
 }  
  
 **public static void** snake(**double**[][] A, **int** n, **int** shiftI, **int** shiftJ) {  
 **int** count = 1;  
 **for** (**int** k = 0; k < 2 \* n - 1; k++) {  
 **if** (k <= n - 1) { *// первая половина матрицы (верхний треугольник)* **if** (k % 2 == 0) { *// k чётное - идём по диагонали снизу вверх* **for** (**int** i = k, j = n - 1; i >= 0; i--, j--) {  
 A[i + shiftI][j + shiftJ] = count++;  
 }  
 } **else** { *// k нечётное - идём по диагонали сверху вниз* **for** (**int** i = 0, j = n - k - 1; i <= k; i++, j++) {  
 A[i + shiftI][j + shiftJ] = count++;  
 }  
 }  
 }  
 **else** { *//вторая половина матрицы (нижний треугольник)* **if** (k % 2 == 0) { *// k чётное - идём по диагонали снизу вверх* **for** (**int** i = n - 1, j = 2 \* n - k - 2; i >= k - n + 1; i--, j--) {  
 A[i + shiftI][j + shiftJ] = count++;  
 }  
 }  
 **else** { *// k нечётное - идём по диагонали сверху вниз* **for** (**int** i = k - n + 1, j = 0; i <= n - 1; i++, j++) {  
 A[i + shiftI][j + shiftJ] = count++;  
 }  
 }  
 }  
 }  
 }  
  
 **public static void** spiral(**double**[][] A, **int** n, **int** shiftI, **int** shiftJ) {  
 **int** count = 1;  
 **double** m = Math.*ceil*(n / 2.0);  
 **for** (**int** k = 0; k < m; k++) {  
 **for** (**int** j = k; j <= n - k - 1; j++) {  
 A[k + shiftI][j + shiftJ] = count++;  
 }  
 **for** (**int** i = k + 1; i <= n - k - 1; i++) {  
 A[i + shiftI][n - k - 1 + shiftJ] = count++;  
 }  
 **for** (**int** j = n - k - 2; j >= k; j--) {  
 A[n - k - 1 + shiftI][j + shiftJ] = count++;  
 }  
 **for** (**int** i = n - k - 2; i >= k + 1; i--) {  
 A[i + shiftI][k + shiftJ] = count++;  
 }  
 }  
  
 }  
  
 **public static void** printMas(**double**[][] A) {  
 ***out***.println(**"Преобразованная матрица имеет вид:"**);  
 **for** (**double**[] doubles : A) {  
 **for** (**int** j = 0; j < A.**length**; j++) {  
 ***out***.print(doubles[j] + **" "**);  
 }  
 ***out***.println();  
 }  
 }  
}

## 9.3. Тестирование программы с проверкой

В таблице 9 представлено тестирование работы программы с проверкой решения задачи на языке Java с MS Excel.

Таблица 9

Тестирование программы с проверкой задания 9

|  |  |
| --- | --- |
| № п/п | Решение Java |
| 1 |  |
| 2 |  |
| 3 |  |
| № п/п | Решение MS Excel |
| 1 |  |
| 2 |  |
| 3 |  |

Сравнение решения задачи с использованием трех прикладных пакетов показала, что решения задачи в Java и MS Excel совпадает. Данный факт подтверждает правильность написанного кода программы. Неопределенностей при решении задачи выявлено не было, возможно данная ситуация связанна с малым количеством проверок данных.

# Задание 10

## 10.1. Общая постановка задачи

Дана матрица с N точками в пространстве, 0-я строчка координаты х, 1-я строчка координаты y. Определить для каждой точки попала ли она в закрашенную область, с использованием массива N элементов (1 – точка попала внутрь области, 0 – точка на границе области, -1 – точка вне области). Области взять из лабораторной работы № 2 задание 5 (условия проверки реализовать как метод). Создать метод для стилизованного вывода ответов по попаданию точки в область в соответствии с массивом данных о попадании точки.

## 10.2. Решение задачи, код программы

**import** java.util.Scanner;  
  
**import static** java.lang.System.***out***;  
  
**public class** Task810 {  
 **public static void** main(String[] args) {  
 Scanner scanner = **new** Scanner(System.***in***);  
 ***out***.println(**"Введите количество точек на проверку:"**);  
 **int** n = scanner.nextInt();  
 **double**[][] points = **new double**[2][n];  
 **for** (**int** j = 0; j < n; j++) {  
 ***out***.printf(**"Введите координату x%d:"**, j+1);  
 ***out***.println();  
 points[0][j] = scanner.nextDouble();  
 ***out***.printf(**"Введите координату y%d:"**, j+1);  
 ***out***.println();  
 points[1][j] = scanner.nextDouble();  
 }  
 *printCheckHitting*(points, n);  
 }  
  
 **public static void** printCheckHitting(**double**[][] points, **int** n) {  
 **for** (**int** j = 0; j < n; j++) {  
 **if** (*checkHitting*(points[0][j], points[1][j]) == 1) {  
 ***out***.println(**"Точка "** + **"("** + points[0][j] + **","** + points[1][j] + **")"** +  
 **" попала в область первой фигуры"**);  
 }  
 **if** (*checkHitting*(points[0][j], points[1][j]) == 10) {  
 ***out***.println(**"Точка "** + **"("** + points[0][j] + **","** + points[1][j] + **")"** +  
 **" попала на границу первой фигуры"**);  
 }  
 **if** (*checkHitting*(points[0][j], points[1][j]) == 2) {  
 ***out***.println(**"Точка "** + **"("** + points[0][j] + **","** + points[1][j] + **")"** +  
 **" попала в область второй фигуры"**);  
 }  
 **if** (*checkHitting*(points[0][j], points[1][j]) == 20) {  
 ***out***.println(**"Точка "** + **"("** + points[0][j] + **","** + points[1][j] + **")"** +  
 **" попала на границу второй фигуры"**);  
 }  
 **if** (*checkHitting*(points[0][j], points[1][j]) == -1) {  
 ***out***.println(**"Точка "** + **"("** + points[0][j] + **","** + points[1][j] + **")"** +  
 **" не попала ни в область, ни на границу фигур"**);  
 }  
 }  
 }  
  
 **public static int** checkHitting(**double** x, **double** y) {  
 **if** (y >= x - 2) { *// попадание в область первой фигуры* **int** Count1 = 0;  
 **if** ((x >= -5) & (x <= -3) & (y < x + 4) & (y > -x - 6) & (y > 2 \* x + 6)) *//1 область* Count1++;  
 **if** ((x >= -3) & (x <= -1) & (y < Math.*sqrt*(4 - Math.*pow*((x + 1), 2)) + 1) & (y > x + 4)) *//2 область* Count1++;  
 **if** ((x >= -1) & (x <= 2) & (y < 2.0 / 3 \* x + 11.0 / 3) & (y > 5.0 / 3 \* x + 5.0 / 3)) *//3 область* Count1++;  
 **if** ((x >= -3) & (x <= -1) & (y < x + 4) & (y > -2 \* x - 6) & (y > x)) *//4 область* Count1++;  
 **if** ((x >= 1) & (x <= 6) & (y < x + 1) & (y < -x / 2 + 7) &  
 (y > x - 2) & (y > x / 3 + 2.0 / 3)) { *//5 область* Count1++;  
 }  
 **if** ((x >= -2) & (x <= 1) & (y < 5 \* x / 3 + 5.0 / 3) & (y > x) & (y > x + 1)) *//6 область* Count1++;  
 **if** ((x >= -1) & (x <= 1) & (y < x) & (y > -Math.*sqrt*(4 - Math.*pow*((x + 1), 2)) + 1)) *//7 область* Count1++;  
 **if** (Count1 > 0)  
 **return** 1;  
 **else** { *// попадание на границу первой фигуры* **int** count1 = 0;  
 **if** ((x >= -5) & (x <= -3) & ((y == x + 4) | (y == -x - 6) | (y == 2 \* x + 6))) *//1 область* count1++;  
 **if** ((x >= -3) & (x <= -1) & ((y == Math.*sqrt*(4 - Math.*pow*((x + 1), 2)) + 1) | (y == x + 4))) *//2 область* count1++;  
 **if** ((x >= -1) & (x <= 2) & ((y == 2.0 / 3 \* x + 11.0 / 3) | (y == 5.0 / 3 \* x + 5.0 / 3))) *//3 область* count1++;  
 **if** ((x >= -3) & (x <= -1) & ((y == x + 4) | (y == -2 \* x - 6) | (y == x))) *//4 область* count1++;  
 **if** ((x >= 1) & (x <= 6) & ((y == x + 1) | (y == -x / 2 + 7) |  
 (y == x - 2) | (y == x / 3 + 2.0 / 3))) { *//5 область* count1++;  
 }  
 **if** ((x >= -2) & (x <= 1) & ((y == 5 \* x / 3 + 5.0 / 3) | (y == x) | (y == x + 1))) *//6 область* count1++;  
 **if** ((x >= -1) & (x <= 1) & ((y == x) | (y == -Math.*sqrt*(4 - Math.*pow*((x + 1), 2)) + 1))) *//7 область* count1++;  
 **if** (count1 > 0)  
 **return** 10;  
 }  
 } **else if** (y < x - 2) { *//попадание в область второй фигуры* **int** Count2 = 0;  
 **if** ((x >= 0) & (x <= 4) & (y < x / 2 - 5) & (y > x / 4 - 5)) { *//1 область* Count2++;  
 }  
 **if** ((x >= 2) & (x <= 4) & (y > -Math.*sqrt*(4 - Math.*pow*((x - 4), 2)) - 1) & (y < 1 - x)) { *//2 область* Count2++;  
 }  
 **if** ((x >= 3) & (x <= 4) & (y > 1 - x) & (y < x - 5) & (y > 1 - x)) { *//3 область* Count2++;  
 }  
 **if** ((x >= 4) & (x <= 6) & (y < Math.*sqrt*(4 - Math.*pow*((x - 4), 2)) - 1) & (y > 5 - x)) { *//4 область* Count2++;  
 }  
 **if** ((x >= 4) & (x <= 7) & (y < 0) & (y < -x + 5) & (y > -2)) { *//5 область* Count2++;  
 }  
 **if** ((x >= 4) & (x >= 5) & (y < -3 \* x + 10) & (y > -x)) { *//6 область* Count2++;  
 }  
 **if** (Count2 > 0) {  
 **return** 2;  
 }  
 **else** {  
 **int** count2 = 0; *// попадание на границу второй фигуры* **if** ((x >= 0) & (x <= 4) & ((y == x / 2 - 5) | (y == x / 4 - 5))) { *//1 область* count2++;  
 }  
 **if** ((x >= 2) & (x <= 4) & ((y == -Math.*sqrt*(4 - Math.*pow*((x - 4), 2)) - 1) | (y < 1 - x))) { *//2 область* count2++;  
 }  
 **if** ((x >= 3) & (x <= 4) & ((y == 1 - x) | (y == x - 5) | (y == 1 - x))) { *//3 область* count2++;  
 }  
 **if** ((x >= 4) & (x <= 6) & ((y == Math.*sqrt*(4 - Math.*pow*((x - 4), 2)) - 1) | (y == 5 - x))) { *//4 область* count2++;  
 }  
 **if** ((x >= 4) & (x <= 7) & ((y == 0) | (y == -x + 5) | (y == -2))) { *//5 область* count2++;  
 }  
 **if** ((x >= 4) & (x >= 5) & ((y == -3 \* x + 10) | (y == -x))) { *//6 область* count2++;  
 }  
 **if** (count2 > 0) {  
 **return** 20;  
 }  
 }  
 }  
 **return** -1;  
 }  
}

## 10.3. Тестирование программы с проверкой

В таблице 10 представлено тестирование работы программы с проверкой решения задачи на языке Java с MS Excel.

Таблица 10

Тестирование программы с проверкой задания 10

|  |  |
| --- | --- |
| № п/п | Решение Java |
| 1 |  |
| № п/п | Решение MS Excel |
| 1 |  |

Сравнение решения задачи с использованием одного прикладного пакета показала, что решения задачи в Java и MS Excel совпадает. Данный факт подтверждает правильность написанного кода программы. Неопределенностей при решении задачи выявлено не было, возможно данная ситуация связанна с малым количеством проверок данных.

# Задание 12

## 12.1. Общая постановка задачи

Дан двумерный квадратный массив A и двумерный квадратный массив обратный к первому A-1. При этом пользователь вводит размерность массива и данные сам и может допустить ошибку при вычислении обратной матрицы или при вводе. Поэтому выполнить проверку соответствуют ли массивы свойству A\* A-1=E, где E – это единичная матрица.

## 12.2. Решение задачи, код программы

**import** java.util.Scanner;  
  
**import static** java.lang.System.***out***;  
  
**public class** Task812 {  
 **public static void** main(String[] args) {  
 Scanner scanner = **new** Scanner (System.***in***);  
 ***out***.println(**"Введите размерность матрицы n x n"**);  
 ***out***.println(**"Введите размерность строки n:"**);  
 **int** n = scanner.nextInt();  
 **double**[][] A = **new double**[n][n];  
 ***out***.println(**"Введите исходную матрицу"**);  
 **for** (**int** i = 0; i < n; i++) {  
 **for** (**int** j = 0; j < n; j++) {  
 A[i][j] = scanner.nextDouble();  
 }  
 }  
 ***out***.println(**"Введите обратную матрицу:"**);  
 **double**[][] inverseA = **new double**[n][n];  
 **for** (**int** i = 0; i < n; i++) {  
 **for** (**int** j = 0; j < n; j++) {  
 inverseA[i][j] = scanner.nextDouble();  
 }  
 }  
 *check*(A, inverseA, n);  
 }  
  
 **public static void** check(**double**[][] A, **double**[][] inverseA, **int** n) {  
 **double**[][] E = **new double**[n][n];  
 **double**[][] C = **new double**[n][n];  
 **int** countE = 1;  
  
 **for**(**int** k = 0; k < n; k++) { *// единичная матрица* E[k][k] = countE;  
 }  
  
 **for** (**int** i = 0; i < n; i++) { *// матрица, которая получается после перемножения исходной и обратной* **for** (**int** j = 0; j < n; j++) {  
 **for** (**int** k = 0; k < n; k++) {  
 C[i][j] += A[i][k] \* inverseA[k][j];  
 }  
 }  
 }  
  
 **int** countCE = 0;  
 **for** (**int** i = 0; i < n; i++) { *// сравнение с единичной матрицей* **for** (**int** j = 0; j < n; j++) {  
 **if** (C[i][j] == E[i][j]) {  
 countCE++;  
 }  
 }  
 }  
  
 **if** (countCE == n \* n) {  
 ***out***.println(**"Исходная матрица и обратная к ней матрица введены верно"**);  
 }  
 **else** {  
 ***out***.println(**"Вы ввели неверно обратную матрицу"**);  
 *printMas*(C);  
 }  
 }  
  
 **public static void** printMas(**double**[][] C) {  
 ***out***.println(**"Матрица, полученная путем перемножения исходной и обратной матрицы, имеет вид:"**);  
 **for** (**double**[] doubles : C) {  
 **for** (**int** j = 0; j < C.**length**; j++) {  
 ***out***.print(doubles[j] + **" "**);  
 }  
 ***out***.println();  
 }  
 }  
}

## 12.3. Тестирование программы с проверкой

Для проверки решения в MS Excel созданы таблицы. В ячейки A2:B3 вписана исходная матрица, в ячейки E2:F3 – обратная матрица, в ячейки I2:J3 – единичная матрица, в ячейки C6:D7 – матрица, полученная перемножением исходной (A2:B3) и обратной (E2:F3) с помощью формулы МУМНОЖ(массив 1; массив 2).

С6=МУМНОЖ(A2:B3;E2:F3)

На рис. 3 представлен вид решения в MS Excel.

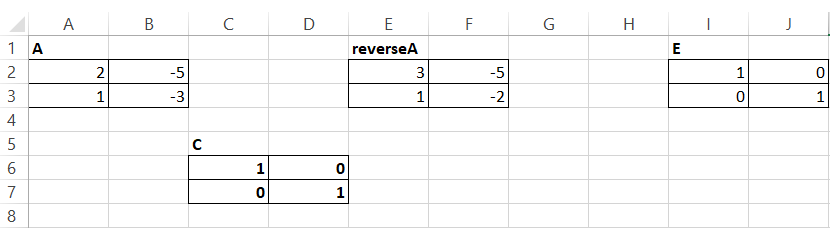


Рисунок 3. Вид решения в MS Excel задания 12

В таблице 11 представлено тестирование работы программы с проверкой решения задачи на языке Java с MS Excel.

Таблица 11

Тестирование программы с проверкой задания 12

|  |  |
| --- | --- |
| № п/п | Решение Java |
| 1 |  |
| 2 |  |
| 3 |  |
| № п/п | Решение MS Excel |
| 1 |  |
| 2 |  |
| 3 |  |

Сравнение решения задачи с использованием трех прикладных пакетов показала, что решения задачи в Java и MS Excel совпадает. Данный факт подтверждает правильность написанного кода программы. Неопределенностей при решении задачи выявлено не было, возможно данная ситуация связанна с малым количеством проверок данных.