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 «Одномерные массивы и матрицы в Java»

по дисциплине «Информатика»

Выполнил: студент группы ИСТ-22-1б Братчиков З.С.

Проверил: Нетбай Георгий Владимирович

Пермь, 2023

**Задание 1**

* 1. ***Постановка задачи***

Ввести массив, состоящий из N элементов целого типа. Массив является элементами целого числа в троичной системе счисления. Проверить правильность введения элементов числа, т.е. в массиве должны быть только 0, 1 и 2. Если число введено верно, то перевести его в десятичную систему счисления и поэлементно записать число в массив. Пример:

|  |  |
| --- | --- |
| В 3-ой | В 10-ой |
| |  |  |  |  | | --- | --- | --- | --- | | 1 | 2 | 0 | 1 | | |  |  | | --- | --- | | 4 | 6 | |

***1.2. Решение задачи, код программы***

import *java.util.*\*;  
  
import static *java.lang.Math*.\*;  
  
public class *z1* {  
 public static void main(*String*[] *args*) {  
 *Scanner* input = new Scanner(*System*.in);  
 *System*.out.println("введите количество цифр в числе");  
 int N = input.nextInt();  
 int tz[] = new int[N];  
 *ArrayList*<int[]> C10 = new ArrayList<int[]>();  
 int counter = 0;  
 *System*.out.println("ведите число в троичной системе счисления по цифрам");  
 for (int i = 0; i < N; i++) {  
 tz[i] = input.nextInt();  
 while (tz[0] == 0 || tz[i] > 2 || tz[i] < 0) {  
 *System*.out.println("Вы ввели неверное число. Попробуйте еще раз");  
 tz[i] = input.nextInt();  
 }  
 }  
 int tmpc10 = 0;  
 for (int i = N - 1, j = 0; i >= 0; i--, j++) {  
 tmpc10 += (int) (tz[i] \* *pow*(3, j));  
 }  
 while (tmpc10 > 0) {  
 C10.add(new int[1]);  
 C10.get(counter)[0] = tmpc10 % 10;  
 tmpc10 /= 10;  
 counter++;  
 }  
 int result[] = new int[counter];  
 for (int i = counter - 1, j = 0; i >= 0; i--, j++){  
 result[j] = C10.get(i)[0];  
 }  
 *System*.out.print("ваше число в 3-ой системе счисления\n\t");  
 for (int i = 0; i<N;i++){  
 *System*.out.print(tz[i]);  
 }  
 *System*.out.print("\nВаше число в 10-ой системе счисления\n\t");  
 for (int i = 0; i<N;i++) {  
 *System*.out.print(result[i]);  
 }  
 }  
}

***1.3. Тестирование работы программы с проверкой***

В таблице 1 представлено тестирование работы программы с проверкой решения задачи на языке Java.

Таблица 1

Тестирование работы программы и проверка результатов решения

|  |  |  |
| --- | --- | --- |
| № п.п. | Решение Java | Проверка |
| 1 |  |  |
| 2 |  |  |

**Задание 2**

***2.1. Постановка задачи***

Ввести массив, состоящий из N элементов. Упорядочить массив так, чтобы все отрицательные числа были расположены вначале по возрастанию, а все положительные – в конце по убыванию, между ними нулевые числа.

***2.2. Решение задачи, код программы***

import *java.util.*\*;  
  
public class *z2* {  
 public static void main(*String*[] *args*) {  
 *Scanner* input = new Scanner(*System*.in);  
 *System*.out.println("Введите длину массива");  
 int N = input.nextInt();  
 int mass[] = new int[N];  
 *System*.out.println("Введите элементы массива");  
 for (int i = 0; i < N; i++) {  
 mass[i] = input.nextInt();  
 }  
 *Arrays*.*sort*(mass);  
 for (int i = 0; i < N - 1; i++) {  
 for (int j = 0; j < N - 1; j++) {  
 if ((mass[j] > 0) & (mass[j] < mass[j + 1])) {  
 int h = mass[j];  
 mass[j] = mass[j + 1];  
 mass[j + 1] = h;  
 }  
 }  
 }  
 *System*.out.println(*Arrays*.*toString*(mass));  
 }  
}

***2.3. Тестирование работы программы с проверкой***

Далее в таблице 2 представлено тестирование работы программы с проверкой решения задач на языке Java

Таблица 2

Тестирование работы программы

|  |  |
| --- | --- |
| № п.п. | Решение Java |
| 1 |  |
| 2 |  |

**Задание 3**

***3.1. 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 прямоугольника.

***3.2. Решение задачи, код программы***

import *java.util.*\*;  
  
public class *z3* {  
 public static void main(*String*[] *args*) {  
 *Scanner* input = new Scanner(*System*.in);  
 *System*.out.println("введите количество точек");  
 int N = input.nextInt();  
 int coord[] = new int[2 \* N];  
 for (int i = 0; i < 2 \* N; i++) {  
 if (i % 2 == 0) {  
 *System*.out.println("введите x-координату точки " + ((i / 2) + 1));  
 coord[i] = input.nextInt();  
 } else {  
 *System*.out.println("введите y-координату точки " + ((i / 2) + 1));  
 coord[i] = input.nextInt();  
 }  
 }  
 for (int i = 0; i < (2 \* N - 4); i += 2) {  
 for (int j = i + 2; j < (2 \* N - 3); j += 2) {  
 for (int k = j + 2; k < (2 \* N - 2); k += 2) {  
 for (int l = k + 2; l < (2 \* N - 1); l += 2) {  
 if (((coord[i] == coord[j]) & (coord[k] == coord[l]) & (coord[i + 1] == coord[l + 1]) & (coord[j + 1] == coord[k + 1])) ||  
 ((coord[i] == coord[l]) & (coord[k] == coord[l]) & (coord[i + 1] == coord[j + 1]) & (coord[l + 1] == coord[k + 1])) ||  
 ((coord[i] == coord[l]) & (coord[k] == coord[j]) & (coord[i + 1] == coord[k + 1]) & (coord[j + 1] == coord[l + 1]))){  
 *System*.out.println("Точки с координатами " + "(" + coord[i] + ";" + coord[i + 1] + "), (" + coord[j] + ";" + coord[j + 1] + "), ("  
 + coord[k] + ";" + coord[k + 1] + "), (" + coord[l] + ";" + coord[l + 1] + ") являются вершинами прямоугольника");  
 }  
 }  
 }  
 }  
 }  
 }  
}

***3.3. Тестирование работы программы с проверкой***

Далее в таблице 3 представлено тестирование работы программы с проверкой решения задач на языке Java.

Таблица 3

Тестирование работы программы и проверка результатов решения

|  |  |
| --- | --- |
| № п.п. | Решение Java |
| 1 |  |
| 2 |  |

**Задание 4**

***4.1. Постановка задачи***

Алгоритм быстрой сортировки. Отличительной особенностью быстрой сортировки является операция разбиения массива на две части относительно опорного элемента. Пусть крайний левый элемент опорный pivot. Установим указатель left на следующий за ним элемент; right – на последний. Алгоритм должен определить правильное положение элемента pivot. Указатель left движется вправо, пока элементы, на которые он показывает, остаются меньше опорного. Указатель right движется влево, пока элементы, на которые он показывает, остаются больше опорного. Указатель left перемещается до тех пор, пока не покажет элемент больше pivot; right движется, пока не покажет элемент меньше pivot. Элементы left и right меняются местами. И движением возобновляется снова. Процесс продолжается до тех пор, пока right не окажется слева от left. Тем самым будет определено правильное место опорного элемента. Осуществляется перестановка опорного элемента с элементом, на который указывает right. Опорный элемент находится в нужном месте: элементы слева от него имеют меньшие значения; справа – большие. Алгоритм рекурсивно вызывается для сортировки подмассивов слева от разрешающего и справа от него. Реализовать алгоритм быстрой сортировки. Выполнить сравнение быстрой сортировки с сортировкой «пузырек», рассмотрев количество итераций, а так же промежуточные результаты сортировки.

***4.2. Решение задачи, код программы***

import *java.util.Scanner*;  
import *java.util.Arrays*;  
public class *z4* {  
 public static int *a* = 0;  
 public static void main(*String*[] *args*) {  
 *Scanner* scanner = new Scanner(*System*.in);  
 *System*.out.println("Введите количество элементов массива");  
 int N = scanner.nextInt();  
 int l = 0;  
 int quicksort[];  
 int puzir[];  
 quicksort = new int[N];  
 puzir = new int[N];  
 for (int i = 0; i < N; i++) {  
 *System*.out.println("Введите элемент массива");  
 int a = scanner.nextInt();  
 quicksort[i] = a;  
 puzir[i] = a;  
 }  
 int left = 0;  
 int right = N-1;  
 *quick\_sort*(left,right,quicksort);  
 *System*.out.println();  
 *System*.out.println("Количество итераций при методе быстрой сортировки: " + *a*);  
 *System*.out.println("Преобразование массива методом быстрой сортировки:");  
 *System*.out.println(*Arrays*.*toString*(quicksort));  
 *System*.out.println();  
 *System*.out.println();  
 *puzirec*(puzir, N);  
 *System*.out.println("Преобразование массива методом пузырек:");  
 *System*.out.println(*Arrays*.*toString*(puzir));  
 }  
  
 public static void puzirec (int *el*[], int *N*) {  
 int c = 0;  
 for (int i = 1; i<*N*; i++){  
 for (int j = 1; j<*N*; j++){  
 if (*el*[j] < *el*[j-1]) {  
 int el2 = *el*[j];  
 *el*[j] = *el*[j-1];  
 *el*[j-1] = el2;  
 c = c +1;  
 *System*.out.println(*Arrays*.*toString*(*el*));  
 }  
 }  
 }  
 *System*.out.println();  
 *System*.out.println("Количество итераций при сортировке пузырьком: " + c);  
 }  
  
 public static void quick\_sort(int *left*, int *right*, int *quick*[]) {  
 if (*left* >= *right*)  
 return;  
  
 int middle = *left* + (*right* - *left*)/2;  
 int pivot = *quick*[middle];  
 int i = *left*;  
 int j = *right*;  
 while (i <= j) {  
 while (*quick*[i] < pivot) {  
 i = i + 1;  
 }  
 while (*quick*[j] > pivot) {  
 j = j - 1;  
 }  
 if (i <= j) {  
 int el2 = *quick*[i];  
 *quick*[i] = *quick*[j];  
 *quick*[j] = el2;  
 i = i + 1;  
 j = j - 1;  
  
 }  
 if (*left* < j) {  
 *a*=*a*+1;  
 *System*.out.println(*Arrays*.*toString*(*quick*));  
 *quick\_sort*(*left*,j,*quick*);  
 }  
 if (*right* > i) {  
 *a*=*a*+1;  
 *System*.out.println(*Arrays*.*toString*(*quick*));  
 *quick\_sort*(i,*right*,*quick*);  
 }  
 }  
 }  
}

***4.3. Тестирование работы программы с проверкой***

Далее в таблице 4 представлено тестирование работы программы с проверкой решения задач на языке Java.

Таблица 4

Тестирование работы программы и проверка результатов решения

|  |  |
| --- | --- |
| № п.п. | Решение Java |
| 1 |  |
| 2 |  |

**Задание 7**

***5.1. Постановка задачи***

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

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

***5.2. Решение задачи, код программы***

import *java.util.*\*;  
  
public class *z7* {  
 public static void main(*String*[] *args*) {  
 *Scanner* in = new Scanner(*System*.in);  
 *System*.out.println("Введите порядок матрицы");  
 int m = in.nextInt();  
 *Integer*[][] matrix = new Integer[m][m];  
 int n = 1;  
 for (int i = 0; i < m; i++) {  
 for (int j = 0; j < m; j++) {  
 matrix[i][j] = 0;  
 }  
 }  
 for (int k = 0; k < m / 2; k++) {  
 for (int i = k, j = k; i < m - 1 - k; i++, n++) {  
 matrix[j][i] = n;  
 }  
 for (int i = m - 1 - k, j = k; j < m - k; j++, n++) {  
 matrix[j][i] = n;  
 }  
 for (int i = k + 1, j = k; i < m - 1 - k; i++, n++) {  
 matrix[i][j] = n;  
 }  
 for (int i = m - 1 - k, j = k; j < m - k - 1; j++, n++) {  
 matrix[i][j] = n;  
 }  
 }  
 if (m % 2 != 0) {  
 matrix[m / 2][m / 2] = n;  
 }  
 for (int i = 0; i < m; i++) {  
 for (int j = 0; j < m; j++) {  
 *System*.out.print(*String*.*format*("%3.0f", (double) matrix[i][j]) + " ");  
 }  
 *System*.out.println("");  
 }  
 }  
}

***5.3. Тестирование работы программы с проверкой***

Далее в таблице 5 представлено тестирование работы программы с проверкой решения задач на языке Java.

Таблица 5

Тестирование работы программы и проверка результатов решения

|  |  |
| --- | --- |
| № п.п. | Решение Java |
| 1 |  |
| 2 |  |

**Задание 8**

***6.1. 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:



***6.2. Решение задачи, код программы***

import *java.util.*\*;  
  
public class *z8* {  
 public static void vivod(*Integer*[][] *matrix*) {  
 for (int i = 0; i < *matrix*.length; i++) {  
 for (int j = 0; j < *matrix*.length; j++) {  
 *System*.out.print(*String*.*format*("%3.0f", (double) *matrix*[i][j]) + " ");  
 }  
 *System*.out.println("");  
 }  
 *System*.out.println("");  
 }  
  
 public static void nuli(*Integer*[][] *matrix*) {  
 int n = *matrix*.length;  
 int p = 1;  
 for (int i = 0; i < n; i++) {  
 for (int j = 0; j < n; j++) {  
 *matrix*[i][j] = 0;  
 }  
 }  
 }  
  
 public static void block(*Integer*[][] *matrix*, int *ni*, int *nj*) {  
 int n = *matrix*.length / 2;  
 int p = 1;  
 for (int k = 0; k < n / 2; k++) {  
 for (int i = k + *ni*, j = k + *nj*; i < n - 1 - k + *ni*; i++, p++) {  
 *matrix*[j][i] = p;  
 }  
 for (int i = n - 1 - k + *ni*, j = k + *nj*; j < n - k + *nj*; j++, p++) {  
 *matrix*[j][i] = p;  
 }  
 if (n%2!=0) {  
 *matrix*[n / 2 + *nj*][n / 2 + *ni*] = p;  
 }  
 }  
 }  
  
 public static void main(*String*[] *args*) {  
 *Scanner* in = new Scanner(*System*.in);  
 *System*.out.println("Введите порядок матрицы");  
 int n = in.nextInt();  
 *Integer*[][] matrix = new Integer[2 \* n][2 \* n];  
 *nuli*(matrix);  
 *block*(matrix, 0, 0);  
 *block*(matrix, 0, n);  
 *block*(matrix, n, n);  
 *block*(matrix, n, 0);  
 *vivod*(matrix);  
 }  
}

***6.3. Тестирование работы программы с проверкой***

Далее в таблице 6 представлено тестирование работы программы с проверкой решения задач на языке Java.

Таблица 6

Тестирование работы программы и проверка результатов решения

|  |  |
| --- | --- |
| № п.п. | Решение Java |
| 1 |  |
| 2 |  |

**Задание 11**

***7.1. Постановка задачи***

Дан двумерный массив 2×4N, где N – количество предполагаемых четырехугольников. Пользователь вводит данные по координатам вершин четырехугольников в двумерный массив. Создать двумерный массив ответов для N предполагаемых четырехугольников: 0-я строчка результаты проверки о существовании четырехугольника с введенными вершинами (1 – существует; 0 – не существует); 1-я строчка классификация четырехугольников (1 – прямоугольник; 2 – квадрат; 3 – ромб; 4 – трапеция; 5 – произвольный; 0 – если такого четырехугольника не существует); 2-я строчка площадь четырехугольника, если он существует, иначе 0; 3-я строчка сообщает результаты проверки можно ли вписать четырехугольник в окружность (-1 – если нельзя; 1 – если можно; 0 – если четырехугольника не существует). Создать метод для стилизованного вывода ответов.

***7.2. Решение задачи, код программы***

import *java.util.*\*;  
import static *java.lang.Math*.\*;  
  
public class *z11* {  
 public static void main(*String*[] *args*){  
 *Scanner* input = new Scanner(*System*.in);  
 *System*.out.println("Введите n");  
 int n = input.nextInt(); double ab, bc, cd, ad, ac, bd;  
 int [][] coord = new int[2][4 \* n]; int [][] answer = new int[4][n];  
 *getArr*(coord);  
 *PrintOut*(coord);  
 for (int i = 0, j = 0; j < n \* 4; j+=4, i++){  
 ab = *sqrt*(*pow*(coord[0][j] - coord[0][j + 1], 2) + *pow*(coord[1][j] - coord[1][j + 1], 2));  
 bc = *sqrt*(*pow*(coord[0][j + 1] - coord[0][j + 2], 2) + *pow*(coord[1][j + 1] - coord[1][j + 2], 2));  
 cd = *sqrt*(*pow*(coord[0][j + 2] - coord[0][j + 3], 2) + *pow*(coord[1][j + 2] - coord[1][j + 3], 2));  
 ad = *sqrt*(*pow*(coord[0][j + 3] - coord[0][j], 2) + *pow*(coord[1][j + 3] - coord[1][j], 2));  
 ac = *sqrt*(*pow*(coord[0][j] - coord[0][j + 2], 2) + *pow*(coord[1][j] - coord[1][j + 2], 2));  
 bd = *sqrt*(*pow*(coord[0][j + 1] - coord[0][j + 3], 2) + *pow*(coord[1][j + 1] - coord[1][j + 3], 2));  
 if ((coord[0][j + 2] - coord[0][j]) / ((coord[0][j + 1] - coord[0][j])) == (coord[1][j + 2] - coord[1][j]) / ((coord[1][j + 1] - coord[1][j]))){  
 answer[0][i] = 0;  
 answer[1][i] = 0;  
 answer[2][i] = 0;  
 answer[3][i] = 0;  
 }  
 else {  
 answer[0][i] = 1;  
 if (ab == bc & bc == cd & cd == ad & ac == bd){  
 answer[1][i] = 2;  
 }  
 else if (ab == cd & bc == ad & ac == bd){  
 answer[1][i] = 1;  
 }  
 else if (ab == bc & bc == cd & cd == ad){  
 answer[1][i] = 3;  
 }  
 else if ((coord[0][j + 1] - coord[0][j]) / (coord[0][j + 3] - coord[0][j + 2]) == (coord[1][j + 1] - coord[1][j]) / (coord[1][j + 3] - coord[1][j + 2])){  
 answer[1][i] = 4;  
 }  
 else answer[1][i] = 5;  
 answer[2][i] = *abs*((coord[0][j] - coord[0][j + 1]) \* (coord[1][j] + coord[1][j + 1]) +  
 (coord[0][j + 1] - coord[0][j + 2]) \* (coord[1][j + 1] + coord[1][j + 2]) +  
 (coord[0][j + 2] - coord[0][j + 3]) \* (coord[1][j + 2] + coord[1][j + 3]) +  
 (coord[0][j + 3] - coord[0][j]) \* (coord[1][j + 3] + coord[1][j]))/2;  
 if (ab == cd & bc == ad) answer[3][i] = 1;  
 else answer[3][i] = -1;  
 }  
 }  
 *PrintOut*(answer);  
 }  
 public static void getArr(int [][] *arr*){  
 *Random* r = new Random();  
 for (int i = 0; i < *arr*.length; i++) {  
 for (int j = 0; j < *arr*[i].length; j++) {  
 *arr*[i][j] = r.nextInt(10);  
 }  
 }  
 }  
 public static void PrintOut(int[][] *arr*){  
 for (int[] ints : *arr*) {  
 for (int anInt : ints) {  
 *System*.out.print(anInt + " | \t");  
 }  
 *System*.out.println();  
 }  
 }  
}

***7.3. Тестирование работы программы с проверкой***

Далее в таблице 1 представлено тестирование работы программы с проверкой решения задачи на языке Java.

Таблица 1

Тестирование работы программы и проверка результатов решения

|  |  |
| --- | --- |
| № п.п. | Решение Java |
| 1 |  |
| 2 |  |

**Задание 12**

***8.1. Постановка задачи***

Дан двумерный квадратный массив коэффициентов системы линейных алгебраических уравнений A, одномерный массив столбца правой части СЛАУ F и одномерный массив столбца полученного решения X. При этом пользователь вводит размерность массива и данные сам и может допустить ошибку при вычислении решения СЛАУ или при вводе. Поэтому выполнить проверку соответствует ли полученное решение данной СЛАУ A\*X=F.

***8.2. Решение задачи, код программы***

import *java.util.*\*;  
  
import static *java.lang.Math*.\*;  
  
  
public class *z\_12* {  
 public static void main(*String*[] *args*) {  
 *Scanner* input = new Scanner(*System*.in);  
 *System*.out.println("Введите размерность СЛАУ");  
 int N = input.nextInt();  
 int A[][] = new int[N][N];  
 int F[] = new int[N];  
 int X[] = new int[N];  
 *System*.out.println("Введите коэффициенты СЛАУ");  
 for (int i = 0; i < N; i++) {  
 for (int j = 0; j < N; j++) {  
 A[i][j] = input.nextInt();  
 }  
 }  
 *System*.out.println("Введите столбец решений F");  
 for (int i = 0; i < N; i++) {  
 F[i] = input.nextInt();  
 }  
 *System*.out.println("Введите столбец полученных решений X");  
 for (int i = 0; i < N; i++) {  
 X[i] = input.nextInt();  
 }  
 int answer[] = new int[N];  
 for (int i = 0; i < N; i++) {  
 answer[i] = 0;  
 }  
 for (int i = 0; i < N; i++) {  
 for (int j = 0; j < N; j++) {  
 answer[i] += A[i][j] \* X[j];  
 }  
 }  
 if (*Arrays*.*equals*(answer, F)) {  
 *System*.out.println("СЛАУ решена верно");  
 } else *System*.out.println("СЛАУ решена с ошибкой");  
 }  
}

***8.3. Тестирование работы программы***



