Оглавление

[Разветвляющиеся программы 1](#_Toc182156694)

[Циклические программы 3](#_Toc182156695)

[Массивы 5](#_Toc182156696)

[Строки 8](#_Toc182156697)

[Методы и функции 9](#_Toc182156698)

[Классы. Объекты 12](#_Toc182156699)

# Разветвляющиеся программы

2.1 - 20

Дан номер года (положительное целое число). Определить количество дней в этом году, учитывая, что обычный год насчитывает 365 дней, а високосный — 366 дней. Високосным считается год, делящийся на 4, за исключением тех годов, которые делятся на 100 и не делятся на 400 (например, годы 300, 1300 и 1900 не являются високосными, а 1200 и 2000 — являются).

**Котлин:**

fun main() {  
 val year = *readLine*()!!.*toInt*()  
 when {  
 year % 400 == 0 -> *println*(366)  
 year % 100 == 0 -> *println*(365)  
 year % 4 == 0 -> *println*(366)  
 else -> *println*(365)  
 }  
}

**Джава:**

import java.util.Scanner;  
  
public class Main {  
 public static void main(String[] args) {  
 Scanner scanner = new Scanner(System.*in*);  
 int year = scanner.nextInt();  
  
 if (year % 400 == 0) {  
 System.*out*.println(366);  
 } else if (year % 100 == 0) {  
 System.*out*.println(365);  
 } else if (year % 4 == 0) {  
 System.*out*.println(366);  
 } else {  
 System.*out*.println(365);  
 }  
 }  
}

2.3 - 20

Написать программу, которая бы по введенному номеру времени года (1 — зима, 2 — весна, 3 — лето, 4 — осень) выдавала соответствующие этому времени года месяцы, количество дней в каждом из месяцев.

Котлин:

fun main() {  
 val seasonNumber = *readLine*()!!.*toInt*()  
  
 when (seasonNumber) {  
 1 -> {  
 *println*("Зима")  
 *println*("Декабрь: 31 день")  
 *println*("Январь: 31 день")  
 *println*("Февраль: 28 дней")  
 }  
 2 -> {  
 *println*("Весна")  
 *println*("Март: 31 день")  
 *println*("Апрель: 30 дней")  
 *println*("Май: 31 день")  
 }  
 3 -> {  
 *println*("Лето")  
 *println*("Июнь: 30 дней")  
 *println*("Июль: 31 день")  
 *println*("Август: 31 день")  
 }  
 4 -> {  
 *println*("Осень")  
 *println*("Сентябрь: 30 дней")  
 *println*("Октябрь: 31 день")  
 *println*("Ноябрь: 30 дней")  
 }  
 else -> *println*("Неверный номер времени года")  
 }  
}

Джава:

import java.util.Scanner;  
  
public class Main {  
 public static void main(String[] args) {  
 Scanner scanner = new Scanner(System.*in*);  
 System.*out*.print("Введите номер времени года (1 - зима, 2 - весна, 3 - лето, 4 - осень): ");  
 int seasonNumber = scanner.nextInt();  
  
 switch (seasonNumber) {  
 case 1:  
 System.*out*.println("Зима");  
 System.*out*.println("Декабрь: 31 день");  
 System.*out*.println("Январь: 31 день");  
 System.*out*.println("Февраль: 28 дней");  
 break;  
 case 2:  
 System.*out*.println("Весна");  
 System.*out*.println("Март: 31 день");  
 System.*out*.println("Апрель: 30 дней");  
 System.*out*.println("Май: 31 день");  
 break;  
 case 3:  
 System.*out*.println("Лето");  
 System.*out*.println("Июнь: 30 дней");  
 System.*out*.println("Июль: 31 день");  
 System.*out*.println("Август: 31 день");  
 break;  
 case 4:  
 System.*out*.println("Осень");  
 System.*out*.println("Сентябрь: 30 дней");  
 System.*out*.println("Октябрь: 31 день");  
 System.*out*.println("Ноябрь: 30 дней");  
 break;  
 default:  
 System.*out*.println("Неверный номер времени года");  
 break;  
 }  
 }  
}

# Циклические программы

3.3 - 20

Дано целое число N (> 0). С помощью операций деления нацело и взятия остатка от деления определить, имеется ли в записи числа N цифра «2». Если имеется, то вывести True, если нет — вывести False

**Котлин:**

fun main() {  
 val n = *readLine*()!!.*toInt*()  
  
 if (n <= 0) {  
 *println*("Число должно быть больше 0")  
 return  
 }  
  
 var number = n  
 var hasDigitTwo = false  
  
 while (number > 0) {  
 val digit = number % 10  
 if (digit == 2) {  
 hasDigitTwo = true  
 break  
 }  
 number /= 10  
 }  
  
 *println*(hasDigitTwo)  
}

**Джава:**

import java.util.Scanner;  
  
public class Main {  
 public static void main(String[] args) {  
 Scanner scanner = new Scanner(System.*in*);  
 int n = scanner.nextInt();  
  
 if (n <= 0) {  
 System.*out*.println("Число должно быть больше 0");  
 return;  
 }  
  
 int number = n;  
 boolean hasDigitTwo = false;  
  
 while (number > 0) {  
 int digit = number % 10;  
 if (digit == 2) {  
 hasDigitTwo = true;  
 break;  
 }  
 number /= 10;  
 }  
  
 System.*out*.println(hasDigitTwo);  
 }  
}

3.4 - 20

Дано целое число N (> 0). Используя один цикл, найти сумму 1! + 2! + 3! + … + N! (выражение N! — N–факториал — обозначает произведение всех целых чисел от 1 до N: N! = 1·2·…·N). Чтобы избежать целочисленного переполнения, проводить вычисления с помощью вещественных переменных и вывести результат как вещественное число.

**Котлин:**

fun main() {  
 val n = *readLine*()!!.*toInt*()  
  
 if (n <= 0) {  
 *println*("Число должно быть больше 0")  
 return  
 }  
  
 var sum = 0.0  
 var factorial = 1.0  
  
 for (i in 1..n) {  
 factorial \*= i  
 sum += factorial  
 }  
  
 *println*(sum)  
}

**Джава:**

import java.util.Scanner;  
  
public class Main {  
 public static void main(String[] args) {  
 Scanner scanner = new Scanner(System.*in*);  
 int n = scanner.nextInt();  
  
 if (n <= 0) {  
 System.*out*.println("Число должно быть больше 0");  
 return;  
 }  
  
 double sum = 0.0;  
 double factorial = 1.0;  
  
 for (int i = 1; i <= n; i++) {  
 factorial \*= i;  
 sum += factorial;  
 }  
  
 System.*out*.println(sum);  
 }  
}

# Массивы

4.1 16

Дано целое число K и набор ненулевых целых чисел; признак его завершения — число 0. Вывести номер последнего числа в наборе, большего K. Если таких чисел нет, то вывести 0.

**Котлин:**

fun main() {  
 val k = *readLine*()!!.*toInt*()  
 val nums = *mutableListOf*<Int>()  
  
 while (true) {  
 val n = *readLine*()!!.*toInt*()  
 if (n == 0) break  
 nums.add(n)  
 }  
  
 var last = 0  
 for (i in nums.*indices*.*reversed*()) {  
 if (nums[i] > k) {  
 last = i + 1  
 break  
 }  
 }  
  
 *println*(last)  
}

**Джава:**

import java.util.ArrayList;  
import java.util.Scanner;  
  
public class Main {  
 public static void main(String[] args) {  
 Scanner scanner = new Scanner(System.in);  
 int k = scanner.nextInt();  
 ArrayList<Integer> nums = new ArrayList<>();  
  
 while (true) {  
 int n = scanner.nextInt();  
 if (n == 0) break;  
 nums.add(n);  
 }  
  
 int last = 0;  
 for (int i = nums.size() - 1; i >= 0; i--) {  
 if (nums.get(i) > k) {  
 last = i + 1;  
 break;  
 }  
 }  
  
 System.out.println(last);  
 }  
}

4.6 12

Дан массив размера N и целое число K (1 < K < N). Перед элементом массива с порядковым номером K вставить новый элемент с нулевым значением.

**Котлин:**

fun main() {  
 val n = *readLine*()!!.*toInt*()  
 val k = *readLine*()!!.*toInt*()  
  
 val array = IntArray(n) **{** *readLine*()!!.*toInt*() **}** val newArray = IntArray(n + 1)  
 for (i in 0 *until* k - 1) {  
 newArray[i] = array[i]  
 }  
 newArray[k - 1] = 0  
 for (i in k - 1 *until* n) {  
 newArray[i + 1] = array[i]  
 }  
  
 *println*(newArray.*joinToString*("\n"))  
}

**Джава:**

import java.util.Scanner;  
  
public class Main {  
 public static void main(String[] args) {  
 Scanner scanner = new Scanner(System.in);  
 int n = scanner.nextInt();  
 int k = scanner.nextInt();  
  
 int[] array = new int[n];  
 for (int i = 0; i < n; i++) {  
 array[i] = scanner.nextInt();  
 }  
  
 int[] newArray = new int[n + 1];  
 for (int i = 0; i < k - 1; i++) {  
 newArray[i] = array[i];  
 }  
 newArray[k - 1] = 0;  
 for (int i = k - 1; i < n; i++) {  
 newArray[i + 1] = array[i];  
 }  
  
 for (int num : newArray) {  
 System.out.println(num);  
 }  
 }  
}

5.3 20

Дана матрица размера M × N. Удалить ее последний столбец, содержащий только отрицательные элементы. Если требуемых столбцов нет, то вывести матрицу без изменений

**Котлин:**

fun main() {  
 val m = *readLine*()!!.*toInt*()  
 val n = *readLine*()!!.*toInt*()  
  
 val mat = Array(m) **{** IntArray(n) **{** *readLine*()!!.*toInt*() **} }** var lastNegCol = -1  
  
 for (j in 0 *until* n) {  
 var allNeg = true  
 for (i in 0 *until* m) {  
 if (mat[i][j] >= 0) {  
 allNeg = false  
 break  
 }  
 }  
 if (allNeg) lastNegCol = j  
 }  
  
 if (lastNegCol != -1) {  
 val newMat = Array(m) **{** IntArray(n - 1) **}** for (i in 0 *until* m) {  
 for (j in 0 *until* lastNegCol) {  
 newMat[i][j] = mat[i][j]  
 }  
 for (j in lastNegCol *until* n - 1) {  
 newMat[i][j] = mat[i][j + 1]  
 }  
 }  
 *printMat*(newMat)  
 } else {  
 *printMat*(mat)  
 }  
}  
  
fun printMat(mat: Array<IntArray>) {  
 for (row in mat) {  
 *println*(row.*joinToString*(" "))  
 }  
}

**Джава:**

import java.util.Scanner;  
  
public class Main {  
 public static void main(String[] args) {  
 Scanner scanner = new Scanner(System.in);  
 int m = scanner.nextInt();  
 int n = scanner.nextInt();  
  
 int[][] mat = new int[m][n];  
 for (int i = 0; i < m; i++) {  
 for (int j = 0; j < n; j++) {  
 mat[i][j] = scanner.nextInt();  
 }  
 }  
  
 int lastNegCol = -1;  
  
 for (int j = 0; j < n; j++) {  
 boolean allNeg = true;  
 for (int i = 0; i < m; i++) {  
 if (mat[i][j] >= 0) {  
 allNeg = false;  
 break;  
 }  
 }  
 if (allNeg) lastNegCol = j;  
 }  
  
 if (lastNegCol != -1) {  
 int[][] newMat = new int[m][n - 1];  
 for (int i = 0; i < m; i++) {  
 for (int j = 0; j < lastNegCol; j++) {  
 newMat[i][j] = mat[i][j];  
 }  
 for (int j = lastNegCol; j < n - 1; j++) {  
 newMat[i][j] = mat[i][j + 1];  
 }  
 }  
 printMat(newMat);  
 } else {  
 printMat(mat);  
 }  
 }  
  
 public static void printMat(int[][] mat) {  
 for (int[] row : mat) {  
 for (int num : row) {  
 System.out.print(num + " ");  
 }  
 System.out.println();  
 }  
 }  
}

# Строки

6.3 10

Даны строки S и S0. Удалить из строки S все подстроки, совпадающие с S0. Если совпадающих подстрок нет, то вывести строку S без изменений

**Котлин:**

fun main() {  
 val s = *readLine*()!!  
 val s0 = *readLine*()!!  
  
 val result = s.*replace*(s0, "")  
  
 *println*(result)  
}

**Джава:**

import java.util.Scanner;  
  
public class Main {  
 public static void main(String[] args) {  
 Scanner scanner = new Scanner(System.*in*);  
 String s = scanner.nextLine();  
 String s0 = scanner.nextLine();  
  
 String result = s.replace(s0, "");  
  
 System.*out*.println(result);  
 }  
}

6.4 10

Дана строка, состоящая из русских слов, разделенных пробелами (одним или несколькими). Вывести строку, содержащую эти же слова, разделенные одним пробелом и расположенные в обратном порядке.

**Котлин:**

fun main() {  
 val input = *readLine*()!!  
 val words = input.*split*(" ").*filter* **{ it**.*isNotEmpty*() **}** val result = words.*reversed*().*joinToString*(" ")  
 *println*(result)  
}

**Джава:**

import java.util.Scanner;  
  
public class Main {  
 public static void main(String[] args) {  
 Scanner scanner = new Scanner(System.*in*);  
 String input = scanner.nextLine();  
 String[] words = input.split("\\s+");  
  
 StringBuilder result = new StringBuilder();  
 for (int i = words.length - 1; i >= 0; i--) {  
 if (!words[i].isEmpty()) {  
 result.append(words[i]).append(" ");  
 }  
 }  
  
 System.*out*.println(result.toString().trim());  
 }  
}

# Методы и функции

7.1 11

Разработать функцию MakeStr(ch1,ch2:char; N:integer):string, возвращающую строку, в которой символы ch1 и ch2 чередуются, и это чередование повторяется N раз. Вводятся пары символов. Используя разработанную функцию, сгенерировать строки из повторений данных символов, причем первая пара должна повторяться 1 раз, вторая пара 2 раза, пара номер i должна повторяться i раз.

**Котлин:**

fun MakeStr(ch1: Char, ch2: Char, N: Int): String {  
 val sb = StringBuilder()  
 for (i in 0 *until* N) {  
 sb.append(ch1)  
 sb.append(ch2)  
 }  
 return sb.toString()  
}  
  
fun main() {  
 val numPairs = *readLine*()!!.*toInt*()  
  
 for (i in 1..numPairs) {  
 val (ch1, ch2) = *readLine*()!!.*split*(" ").*map* **{ it**[0] **}** val result = *MakeStr*(ch1, ch2, i)  
 *println*(result)  
 }  
}

**Джава:**

import java.util.Scanner;  
  
public class Main {  
 public static String MakeStr(char ch1, char ch2, int N) {  
 StringBuilder sb = new StringBuilder();  
 for (int i = 0; i < N; i++) {  
 sb.append(ch1);  
 sb.append(ch2);  
 }  
 return sb.toString();  
 }  
  
 public static void main(String[] args) {  
 Scanner scanner = new Scanner(System.*in*);  
 int numPairs = scanner.nextInt();  
 scanner.nextLine();  
  
 for (int i = 1; i <= numPairs; i++) {  
 String[] chars = scanner.nextLine().split(" ");  
 char ch1 = chars[0].charAt(0);  
 char ch2 = chars[1].charAt(0);  
 String result = *MakeStr*(ch1, ch2, i);  
 System.*out*.println(result);  
 }  
 }  
}

7.2 11

Описать процедуру Minmax(X, Y), записывающую в переменную X минимальное из значений X и Y, а в переменную Y — максимальное из этих значений (X и Y — вещественные параметры, являющиеся одновременно входными и выходными). Используя четыре вызова этой процедуры, найти минимальное и максимальное из данных чисел A, B, C, D.

**Котлин:**

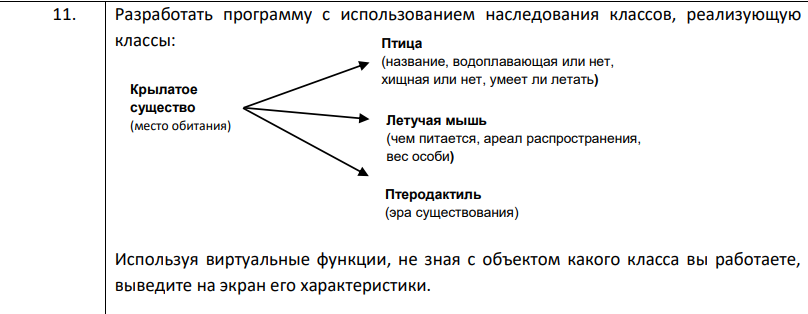
fun Minmax(x: Double, y: Double, min: DoubleArray, max: DoubleArray) {  
 if (x < y) {  
 min[0] = x  
 max[0] = y  
 } else {  
 min[0] = y  
 max[0] = x  
 }  
}  
  
fun main() {  
 // Считываем числа A, B, C, D  
 val A = *readLine*()!!.*toDouble*()  
 val B = *readLine*()!!.*toDouble*()  
 val C = *readLine*()!!.*toDouble*()  
 val D = *readLine*()!!.*toDouble*()  
  
 val min1 = DoubleArray(1)  
 val max1 = DoubleArray(1)  
 val min2 = DoubleArray(1)  
 val max2 = DoubleArray(1)  
  
 *Minmax*(A, B, min1, max1)  
 *Minmax*(C, D, min2, max2)  
  
 val finalMin = DoubleArray(1)  
 val finalMax = DoubleArray(1)  
 *Minmax*(min1[0], min2[0], finalMin, finalMax)  
 *Minmax*(max1[0], max2[0], finalMin, finalMax)  
  
 *println*("Минимальное значение: ${finalMin[0]}")  
 *println*("Максимальное значение: ${finalMax[0]}")  
}

**Джава:**

import java.util.Scanner;  
  
public class Main {  
 public static void Minmax(double x, double y, double[] min, double[] max) {  
 if (x < y) {  
 min[0] = x;  
 max[0] = y;  
 } else {  
 min[0] = y;  
 max[0] = x;  
 }  
 }  
  
 public static void main(String[] args) {  
 Scanner scanner = new Scanner(System.in);  
  
 // Считываем числа A, B, C, D  
 double A = scanner.nextDouble();  
 double B = scanner.nextDouble();  
 double C = scanner.nextDouble();  
 double D = scanner.nextDouble();  
  
 double[] min1 = new double[1];  
 double[] max1 = new double[1];  
 double[] min2 = new double[1];  
 double[] max2 = new double[1];  
  
 Minmax(A, B, min1, max1);  
 Minmax(C, D, min2, max2);  
  
 double[] finalMin = new double[1];  
 double[] finalMax = new double[1];  
 Minmax(min1[0], min2[0], finalMin, finalMax);  
 Minmax(max1[0], max2[0], finalMin, finalMax);  
  
 System.out.println("Минимальное значение: " + finalMin[0]);  
 System.out.println("Максимальное значение: " + finalMax[0]);  
 }  
}

# Классы. Объекты

8.2 11



**Котлин:**

open class WingedEssence(val placeResident: String) {  
 open fun printInfo() {  
 *println*("Место обитания: $placeResident")  
 }  
}  
  
open class Bird(  
 placeResident: String,  
 val name: String,  
 val water: String,  
 val predatory: String,  
 val canFly: String) : WingedEssence(placeResident) {  
 override fun printInfo() {  
 super.printInfo()  
 *println*("Название: $name, водоплавающая: $water, хищник: $predatory, умение летать: $canFly")  
 }  
}  
  
open class FlyMouse(  
 placeResident: String,  
 val food: String,  
 val areal: String,  
 val weight: String) : WingedEssence(placeResident) {  
 override fun printInfo() {  
 super.printInfo()  
 *println*("Чем питается: $food, ареал: $areal, вес: $weight")  
 }  
}  
  
fun main() {  
 val creatures = *listOf*(  
 Bird("Лес", "Сова", "Нет", "Да", "Да"),  
 FlyMouse("Пещера", "Насекомые", "Европа", "0.1 кг")  
 )  
  
 creatures.*forEach* **{** creature **->** *println*("Информация о существе:")  
 creature.printInfo()  
 *println*()  
 **}**}

**Джава:**

import java.util.List;  
import java.util.Arrays;  
  
abstract class WingedEssence {  
 protected String placeResident;  
  
 public WingedEssence(String placeResident) {  
 this.placeResident = placeResident;  
 }  
  
 public void printInfo() {  
 System.*out*.println("Место обитания: " + placeResident);  
 }  
}  
  
class Bird extends WingedEssence {  
 private String name;  
 private String water;  
 private String predatory;  
 private String canFly;  
  
 public Bird(String placeResident, String name, String water, String predatory, String canFly) {  
 super(placeResident);  
 this.name = name;  
 this.water = water;  
 this.predatory = predatory;  
 this.canFly = canFly;  
 }  
  
 @Override  
 public void printInfo() {  
 super.printInfo();  
 System.*out*.println("Название: " + name + ", водоплавающая: " + water + ", хищник: " + predatory + ", умение летать: " + canFly);  
 }  
}  
  
class FlyMouse extends WingedEssence {  
 private String food;  
 private String areal;  
 private String weight;  
  
 public FlyMouse(String placeResident, String food, String areal, String weight) {  
 super(placeResident);  
 this.food = food;  
 this.areal = areal;  
 this.weight = weight;  
 }  
  
 @Override  
 public void printInfo() {  
 super.printInfo();  
 System.*out*.println("Чем питается: " + food + ", ареал: " + areal + ", вес: " + weight);  
 }  
}  
  
public class Main {  
 public static void main(String[] args) {  
 List<WingedEssence> creatures = Arrays.*asList*(  
 new Bird("Лес", "Сова", "Нет", "Да", "Да"),  
 new FlyMouse("Пещера", "Насекомые", "Европа", "0.1 кг")  
 );  
  
 for (WingedEssence creature : creatures) {  
 System.*out*.println("Информация о существе:");  
 creature.printInfo();  
 System.*out*.println();  
 }  
 }  
}