Министерство образования и науки РФ
Государственное образовательное учреждение
Высшего и профессионального образования
Иркутский Государственный университет
Институт математики, экономики и информатики

Программирование на языке Java в примерах

учебное пособие

2014

Учебное пособие предназначено для студентов, обучающихся по направлению подготовки бакалавров 0100400.62 «прикладная математика и информатика». Может быть использовано в курсах «Основы информатики», «Компьютерные науки».

Составитель: Мезенцев А.В.

Оглавление

1. Консольные приложения 5
1.1. Приложение без циклов 5
1.2. Приложение с циклом for 7
1.3. Приложение с циклом while 8
1.4. Приложения со строками 9
1.5. Приложения с одномерными массивами 12
1.5.1. Суммирование элементов одномерного массива 12
1.5.2. Поиск максимума 13
1.5.3. Удовлетворяют ли все элементы массива условию 14
1.5.4. Инвертирование массива 15
1.5.5. Сортировка 16
1.5.6. Поиск подпоследовательности в последовательности 19
1.6. Приложения с двумерными массивами 22
1.6.1. Суммирование 22
1.6.2. Поиск максимума (минимума) 25
1.6.3. Части матрицы 26
1.6.4. Транспонирование матрицы 37
1.6.5. Подматрица матрицы 38
2. Приложения с визуальным интерфейсом 42
2.1. Простейшее приложение с визуальным интерфейсом 42
2.2. Ввод/вывод вектора и матрицы 42
2.3. Многострочное текстовое окно 44
2.4. Одномерный массив текстовых окон 45
2.5. Двумерный массив текстовых окон 47
2.6. Графика в Java 50
2.6.1. Вывод текста по центру области рисования 50

2.6.2. График функции 51
2.6.3. Интерактивная графика. Ломаная 52
2.6.4. Моделирование текстового курсора в области рисования 55
3. Файлы 58
3.1. Файлы последовательного доступа 58
3.2. Файлы прямого доступа 60
4. Параллельные процессы 64
5. Классы 66

1. Консольные приложения

1.1. Приложение без циклов

Даны целые коэффициенты квадратного уравнения. Вывести действительные корни квадратного уравнения.

```
(01)import java.io.*;
(02)
(03)public class QuadraticEquationRoots {
(04)
(05)  public static void main(String[] args)throws IOException {
(06)    BufferedReader br = new BufferedReader(new
(07)      InputStreamReader(System.in));
(08)    int a, b, c;
(09)    double d, x1, x2;
(10)    System.out.print("a = ");
(11)    a = Integer.parseInt(br.readLine());
(12)    System.out.print("b = ");
(13)    b = Integer.parseInt(br.readLine());
(14)    System.out.print("c = ");
(15)    c = Integer.parseInt(br.readLine());
(16)    System.out.println(a+"x^2+"+b+"x+"+c+"=0");
(17)    d = b * b - 4 * a * c;
(18)    if(d>0){
(19)      x1 = (-b + Math.sqrt(d)) / (2 * a);
(20)      x2 = (-b - Math.sqrt(d)) / (2 * a);
(21)      System.out.println("x1 ="+x1+"x2 ="+x2);
(22)    }else
(23)      if(d==0){
(24)        x1 = -b / (2 * a);
(25)        System.out.println("x ="+x1);
(26)      }else
(27)        System.out.println("Нет корней");
```

```
(28)  }
(29)
(30)}
```

(01) – оператор `import java.io.*;` нужен для того, чтобы мы могли кратко обращаться к классу `BufferedReader` , в противном случае пришлось бы писать `java.io.BufferedReader` (вообще говоря, `*` означает, что мы можем обращаться кратко ко всем классам из пакета `java.io` , можно было написать `import java.io.BufferedReader;` указав явно тот класс, который нам нужен).

(03) – любое приложение Java состоит как минимум из одного класса.

(05) – простейшее приложение Java состоит из класса содержащего статический метод `main` (статический – значит, к этому методу можно обращаться, не создавая экземпляра класса). Именно этот метод вызывается Java-машиной, когда запускается наше приложение. Мы обязаны вставить в заголовок метода `throws IOException` . Так мы собираемся воспользоваться вводом с клавиатуры, во время выполнения которого могут возникнуть ошибки (исключительная ситуация – `Exception` по терминологии Java), у нас есть две возможности: - первая – обработать эту исключительную ситуацию самим, или вторая – «пробросить» ее, т. е., указать компилятору, что исключительную ситуацию будет обрабатывать метод вызвавший наш метод (в нашем случае – это сама Java-машина).

(06) - объявляется и инициализируется переменная `br` – буферизованный поток ввода с клавиатуры. В отличии от стандартного потока `System.in`, он позволяет вводить цепочку символов за один раз с помощью метода `br.readLine()`.

(11) - ввод с клавиатуры значения `a`. Метод `br.readLine()` возвращает строку (предположительно из цифр) набранную на клавиатуре, вплоть до нажатия клавиши ENTER. Метод `Integer.parseInt(br.readLine())` пытается преобразовать эту строку в целое число. Это число присваивается переменной `a`. (13) и (15) аналогично.

(19) – в Java есть класс `Math` содержащий много статических методов для вычисления математических функций, одна из них `sqrt()` – корень квадратный.

1.2. Приложение с циклом for

Вычислить сумму и среднее *n* дробей: *a*/1, *a*/2, …, *a*/*n*

```
(01)import java.io.*;
(02)
(03)public class SumAndAverage {
(04)
(05)  public static void main(String[] args) throws IOException{
(06)    int a, i, n;
(07)    double sum, avg;
(08)    BufferedReader br = new BufferedReader(new
(09)      InputStreamReader(System.in));
(10)    System.out.print("a = ");
(11)    a=Integer.parseInt(br.readLine());
(12)    System.out.print("n = ");
(13)    n=Integer.parseInt(br.readLine());
(14)    sum=0;
(15)    for(i=1;i<=n;i++)
(16)      sum=sum+(double)a/n;
(17)    avg=sum/n;
(18)    System.out.println("Сумма равна "+sum+"\nСреднее равно "+avg);
(19)  }
(20)
(21)}
```

(06) – объявляются переменные целого типа.

(07) – сумма и тем более среднее дробей могут быть и не целыми, поэтому мы объявляем переменные `sum` и `avg`, как вещественные двойной точности (можно было и одинарной точности - `float`).

(16) – отметим одну особенность языка Java: в нем только одна операция деления / которая интерпретируется как деление нацело, если оба аргумента целые, и вещественное деление, если хотя бы один аргумент вещественный. Поэтому нам

пришлось «принудительно» преобразовать значение `a` в вещественный тип (`(double)a`) иначе бы результат был целый.

(17) – здесь оба аргумента вещественные, преобразование не потребовалось.

(18) – здесь в одной из строковых констант используется управляющий символ `\n` – переход на начало следующей строки при выполнении вывода.

1.3. Приложение с циклом while

```
(01)package infiniteseriessum;
(02)import java.io.*;
(03)
(04)public class InfiniteSeriesSum {
(05)
(06)  public static void main(String[] args) throws IOException{
(07)    double x, p, n, sum, f, eps;
(08)    BufferedReader br = new BufferedReader(new
(09)      InputStreamReader(System.in,"CP866"));
(10)    System.out.print("Введите x(0<|x|<1):");
(11)    x = Double.parseDouble(br.readLine());
(12)    System.out.print("Введите eps:");
(13)    eps = Double.parseDouble(br.readLine());
(14)    f = Math.exp(x);
(15)    sum = 1;
(16)    p = 1;
(17)    n = 1;
(18)    while(Math.abs(p) >= eps){
(19)      p *= x/n;
(20)      sum += p;
(21)      n++;
(22)    }
(23)    System.out.printf(
(24)          "x = \t%.2f \neps = \t%g\nn = \t%.0f\nf = "+
(25)          "\t%.10f\nsum = \t%.10f\n", x, eps, n, f, sum);
```

```
(26)   }
(27)
(28)}
```

1.4. Приложения со строками

Рассмотрим для справки самые главные методы класса `String` (ниже переменные `s`, `s1`, `s2` – имеют тип `String`):

1. `s.length()` – возвращает целое значение – длина строки (количество символов в строке).

2. `s.charAt(int index)` – возвращает значение типа `char` – символ в строке под номером `index` (первый символ имеет номер 0).

3. `s.substring(int begin, int end)` – возвращает строку являющуюся подстрокой `s` , начиная с символа с номером `begin` и заканчивая символом с номером `end-1`.

`s.substring(int begin)` – до конца строки.

4. `s.split(String delimits)` – возвращает одномерный массив строк, полученный разбиением исходной строки на подстроки с разделителем `delimits`.

5. `s1==s2` – истинно только в том случае, когда обе переменные указывают на один и тот же объект (строку). Вместо этого надо использовать методы:

`s1.equals(s2)` – возвращает `true`, если обе переменные указывают на строки с одинаковыми кодами символов.

`s1.equalsIgnoreCase(s2)` – игнорирует регистр, т.е. большие и маленькие буквы считаются одинаковыми.

6. `s1.indexOf(s2)` – возвращает номер символа в `s1`, начиная с которого располагается подстрока равная `s2` (возвращает `-1`, если `s2` не является подстрокой `s1`).

7. `s1.trim()` – возвращает строку равную `s1`, за исключением того, что в ней отсутствуют начальные и конечные пробелы.

```
(01)//Заменить в предложении одно заданное слово другим
(02)package stringexample;
(03)import java.io.*;
(04)import java.util.*;
(05)
(06)public class StringExample {
(07)
(08)  public static void main(String[] args) throws IOException {
(09)    BufferedReader br = new BufferedReader(new
(10)      InputStreamReader(System.in,"CP1251"));
(11)    String s, w1, w2;
(12)    int i=0;
(13)    System.out.print("Введите предложение: ");
(14)    s = br.readLine();
(15)    System.out.print("Введите слово для поиска: ");
(16)    w1= br.readLine();
(17)    System.out.print("Введите слово для замены: ");
(18)    w2 = br.readLine();
(19)    i = s.indexOf(w1); //Находим номер символа в строке s
(20)                       // с которого подстрока w1 входит
(21)                       // в строку s. (-1 если нет).
(22)  /*
(23)    if(i != -1){
(24)      s = s.substring(0, i) + w2 +
(25)          s.substring(i+w1.length());
(26)    }
(27)  */
(28)    s = s.replaceFirst(w1, w2);
(29)    System.out.println("В результате замены получили: " + s);
(30)    // Поменять местами самое длинное и самое короткое слова
(31)    String wmin=s, wmax="", w="";
(32)    char c;
(33)    s=s.trim()+" ";
(34)    int imax=0, imin=0;
(35)    for(i=0;i<s.length();i++){
(36)      c = s.charAt(i);
```

```
(37)      if(c!=' '){
(38)        w=w+Character.toString(c);
(39)      }else{
(40)        if(wmin.length()>w.length()){
(41)          wmin=w;
(42)          imin=i-w.length();
(43)        }
(44)        if(wmax.length()<w.length()){
(45)          wmax=w;
(46)          imax=i-w.length();
(47)        }
(48)        w="";
(49)      }
(50)    }
(51)  /*
(52)    StringTokenizer st = new StringTokenizer(s, " .,!-?;:");
(53)    while(st.hasMoreTokens()){ //Если есть еще слова
(54)      w=st.nextToken();          //Получаем очередное слово
(55)      if(wmin.length()>w.length())wmin=w;
(56)      if(wmax.length()<w.length())wmax=w;
(57)    }
(58)    imax=s.indexOf(wmax);
(59)    imin=s.indexOf(wmin);
(60)  */
(61)  /*
(62)    String[] tokens=s.split(" ");
(63)    for(i=0;i<tokens.length;i++){
(64)      System.out.println("["+tokens[i]+"]");
(65)      if(wmin.length()>tokens[i].length()){
(66)        wmin=tokens[i];
(67)        imin=i;
(68)      }
(69)      if(wmax.length()<tokens[i].length()){
(70)        wmax=tokens[i];
(71)        imax=i;
(72)      }
```

```
(73)    }
(74)    s="";
(75)    for(i=0;i<tokens.length;i++){
(76)      if(i==imax)
(77)        s=s+" "+wmin;
(78)      else
(79)        if(i==imin)
(80)          s=s+" "+wmax;
(81)        else
(82)          s=s+" "+tokens[i];
(83)    }
(84)  */
(85)
(86)    if(imax>imin){
(87)      s=s.substring(0, imin)+wmax+
(88)        s.substring(imin+wmin.length(), imax)+
(89)        wmin+s.substring(imax+wmax.length());
(90)    }
(91)    if(imax<imin){
(92)      s=s.substring(0, imax)+wmin+
(93)        s.substring(imax+wmax.length(), imin)+
(94)        wmax+s.substring(imin+wmin.length());
(95)    }
(96)
(97)    System.out.println("В результате замены получили: " +
(98)                       "["+s+"]");
(99)  }
(100)}
```

1.5. Приложения с одномерными массивами

1.5.1. Суммирование элементов одномерного массива

```
(01)package arraysum;
(02)import java.io.*;
(03)
(04)public class ArraySum {
(05)
(06)  public static void main(String[] args) throws IOException{
(07)    BufferedReader br = new BufferedReader(new
(08)      InputStreamReader(System.in));
(09)    int[] a;
(10)    int n, i, sum;
(11)    System.out.println("Введите длину массива");
(12)    n=Integer.parseInt(br.readLine());
(13)    a=new int[n];
(14)    for(i=0; i<n; i++){
(15)      System.out,print("a["+i+"]=");
(16)      a[i]=Integer.parseInt(sa[i]);
(17)    }
(18)
(19)    sum=0;
(20)    for(i=0;i<n;i++)
(21)      sum=sum+a[i];
(22)      System.out.print("Сумма элементов массива равна "+sum);
(23)  }
(24)
(25)}
```

1.5.2. Поиск максимума

```
(01)package maximuminarray;
(02)import java.io.*;
(03)
(04)public class MaximumInArray {
```

```
(05)
(06)  public static void main(String[] args) throws IOException{
(07)    BufferedReader br = new BufferedReader(new
(08)      InputStreamReader(System.in));
(09)    int[] a;
(10)    int n, i, max;
(11)    System.out.println("Введите длину массива");
(12)    n=Integer.parseInt(br.readLine());
(13)    a=new int[n];
(14)    for(i=0; i<n; i++){
(15)      System.out,print("a["+i+"]=");
(16)      a[i]=Integer.parseInt(sa[i]);
(17)    }
(18)
(19)    max=a[0];
(20)    for(i=1;i<n;i++)
(21)      if(max<a[i]) max=a[i];
(22)      System.out.print("Максимум в массиве равен "+max);
(23)  }
(24)
(25)}
```

1.5.3. Удовлетворяют ли все элементы массива условию

```
(01)package nondecreaseordered;
(02)import java.io.*;
(03)
(04)public class NonDecreaseOrdered {
(05)
(06)  public static void main(String[] args) throws IOException{
(07)    BufferedReader br = new BufferedReader(new
(08)      InputStreamReader(System.in));
```

```
(09)    int[] a;
(10)    int n, i;
(11)    boolean f=true;
(12)    System.out.println("Введите длину массива");
(13)    n=Integer.parseInt(br.readLine());
(14)    a=new int[n];
(15)    for(i=0; i<n; i++){
(16)      System.out,print("a["+i+"]=");
(17)      a[i]=Integer.parseInt(sa[i]);
(18)    }
(19)
(20)    for(i=0;i<n-1;i++)
(21)      if(a[i]>a[i+1]){
(22)        f=false;
(23)        break;
(24)      }
(25)    if(f)
(26)      System.out.println("Массив упорядочен по неубыванию");
(27)    else
(28)      System.out.println("Массив не упорядочен по неубыванию");
(23)  }
(24)
(25)}
```

1.5.4. Инвертирование массива

```
(01)package arrayinversing;
(02)import java.io.*;
(03)
(04)public class ArrayInversing {
(05)
(06)  public static void main(String[] args) throws IOException{
```

```
(07)    BufferedReader br = new BufferedReader(new
(08)      InputStreamReader(System.in));
(09)    int[] a;
(10)    int n, i, t;
(11)    System.out.println("Введите целые числа через пробел");
(12)    String s = br.readLine();
(13)    String[] sa = s.split(" ");
(14)    n = sa.length;
(15)    a = new int[n];
(16)    for(i=0; i<n; i++)
(17)      a[i] = Integer.parseInt(sa[i]);
(18)
(19)    for(i=0;i<n/2;i++){
(20)      t=a[n-i-1];
(21)      a[n-i-1]=a[i];
(22)      a[i]=t;
(23)    }
(24)    for(i=0;i<n;i++)
(25)      System.out.print(a[i]+ " ");
(19)  }
(20)}
```

1.5.5. Сортировка

Сортировка одномерного массива.

```
(01)package arraysorting;
(02)import java.io.*;
(03)
(04)public class ArraySorting {
(05)
(06)  public static void selectionSort(int[] array){
(07)    int i, j, n, min, imin;
(08)    n = array.length;
```

```
(09)    for(i=0; i<n-1; i++){
(10)      min = array[i];
(11)      imin = i;
(12)      for(j=i+1; j<n; j++)
(13)        if(min > array[j]){
(14)          min = array[j];
(15)          imin = j;
(16)        }
(17)      array[imin] = array[i];
(18)      array[i] = min;
(19)    }
(20)  }
(21)
(22)  public static void bubbleSort(int[] array){
(23)    int i, n, t;
(24)    boolean f;
(25)    n = array.length;
(26)    do{
(27)      f = false;
(28)      for(i=0; i<n-1; i++){
(29)        if(array[i] > array[i+1]){
(30)          t = array[i];
(31)          array[i] = array[i+1];
(32)          array[i+1] = t;
(33)          f = true;
(34)        }
(35)      }
(36)    }while(f);
(37)  }
(38)
(39)  public static void bubbleExtSort(int[] array){
(40)    int i, n, t, j;
(41)    n = array.length;
(42)    for(i=0; i<n-1; i++){
(43)      j = i+1;
(44)      while(j > 0 && array[j] < array[j-1]){
```

```
(45)        t = array[j];
(46)        array[j] = array[j-1];
(47)        array[j-1] = t;
(48)        j--;
(49)      }
(50)    }
(51)  }
(52)
(53)  public static void main(String[] args) throws IOException{
(54)    BufferedReader br = new BufferedReader(new
(55)      InputStreamReader(System.in));
(56)    int[] a;
(57)    int n, i;
(58)    long start, end;
(59)    System.out.println("Введите целые числа через пробел");
(60)    String s = br.readLine();
(61)    String[] sa = s.split(" ");
(62)    n = sa.length;
(63)    a = new int[n];
(64)    for(i=0; i<n; i++)
(65)      a[i] = Integer.parseInt(sa[i]);
(66)
(67)    start = System.nanoTime();
(68)    selectionSort(a);
(69)    end = System.nanoTime();
(70)    System.out.println("Сортировка выбором:  ");
(71)    System.out.print((end-start)+ " наносекунд");
(72)
(73)    start = System.nanoTime();
(74)    bubbleSort(a);
(75)    end = System.nanoTime();
(76)    System.out.println("Сортировка пузырьком:  ");
(77)    System.out.print((end-start)+ " наносекунд");
(78)
(79)    start = System.nanoTime();
(80)    bubbleExtSort(a);
```

```
(81)     end = System.nanoTime();
(82)     System.out.println("Сортировка улучшенным пузырьком:  ");
(83)     System.out.print((end-start)+ " наносекунд");
(84)   }
(85)}
```

Результаты работы:

```
Исходный массив:
1 10 2 9 3 8 4 7 5 6
Сортировка выбором: 17536 наносекунд
Сортировка пузырьком: 21386 наносекунд
Сортировка улучшенным пузырьком: 19248 наносекунд

Исходный массив:
10 9 8 7 6 5 4 3 2 1
Сортировка выбором: 17537 наносекунд
Сортировка пузырьком: 36357 наносекунд
Сортировка улучшенным пузырьком: 27375 наносекунд

Исходный массив:
1 2 3 4 5 6 7 8 9 10
Сортировка выбором: 15398 наносекунд
Сортировка пузырьком:  8983 наносекунд
Сортировка улучшенным пузырьком:  8983 наносекунд
```

1.5.6. Поиск подпоследовательности в последовательности

В последовательности целых чисел, заданной в виде одномерного массива, подпоследовательности разделяются локальными максимумами в массиве (a_i – локальный максимум, если $a_{i-1}<a_i>a_{i+1}$, для $0<i<n-1$. a_0 – локальный максимум, если $a_0>a_1$. a_{n-1} – локальный максимум, если $a_{n-1}>a_{n-2}$). Найти подпоследовательность наибольшей длины

```
(01)package subsequence;
(02)import java.io.*;
(03)
(04)public class SubSequence {
(05)
(06)  public static void main(String[] args)throws IOException {
(07)    int[] a, b;
(08)    int n, k=0;
(09)    BufferedReader br = new BufferedReader(new
(10)      InputStreamReader(System.in));
(11)    System.out.print("Введите последовательность"+
(12)                     " целых чисел через пробел\n>");
(13)    String s = br.readLine();
(14)    String[] sa = s.split(" ");
(15)    n = sa.length;
(16)    a = new int[n];
(17)    for(int i=0; i<n; i++)
(18)      a[i] = Integer.parseInt(sa[i]);
(19)    b = new int[n/2];
(20)    if(a[0] > a[1]){
(21)      b[k] = 0;
(22)      k++;
(23)    }
(24)    for(int i=1; i<n-1; i++)
(25)      if(a[i] > a[i-1] && a[i] > a[i+1]){
(26)        b[k] = i;
(27)        k++;
(28)      }
(29)    if(a[n-1] > a[n-2]){
(30)      b[k] = n-1;
(31)      k++;
(32)    }
(33)
(34)    System.out.println("Порядковые номера разделителей:");
(35)    for(int i=0; i<k; i++)
(36)      System.out.print(b[i] + " ");
```

```
(37)    System.out.println();
(38)
(39)    int curbeg, curlen, maxbeg, maxlen;
(40)    if(b[0] != 0){
(41)      maxbeg = 0;
(42)      maxlen = b[0];
(43)    }else{
(44)      maxbeg = b[0]+1;
(45)      maxlen = b[1]-b[0]-1;
(46)    }
(47)    for(int i=1; i<k-1; i++){
(48)      curbeg = b[i]+1;
(49)      curlen = b[i+1]-curbeg;
(50)      if(maxlen < curlen){
(51)        maxbeg = curbeg;
(52)        maxlen = curlen;
(53)      }
(54)    }
(55)    if(b[k-1] != n-1){
(56)      curbeg = b[k-1]+1;
(57)      curlen = n-1-curbeg;
(58)      if(maxlen < curlen){
(59)        maxbeg = curbeg;
(60)        maxlen = curlen;
(61)      }
(62)    }
(63)    System.out.println("Подпоследовательность "+
(64)                       "наибольшей длины:");
(65)    for(int i=0; i<maxlen; i++)
(66)      System.out.print(a[maxbeg+i] + " ");
(67)    System.out.println();
(68)  }
(70)}
```

Идея состоит в том, что строится вспомогательный массив `b`, в который последовательно, в порядке нахождения при просмотре исходного массива `a`,

заносятся индексы разделителей подпоследовательностей в последовательности (в нашем случае – локальные максимумы в массиве `a`). Длину массива b мы не знаем, но, по крайней мере, она не может быть больше длины исходного массива (вырожденный случай – исходный массив состоит из одних разделителей пустых подпоследовательностей). В нашем случае разделители не могут находится рядом.

(19) создаем массив `b` длиной `n/2`.

(20) – (23) – проверяем, не является ли разделителем a[0].

(24) – (28) – проверяем, не является ли разделителем a[i] для i от 1 до n-2.

(29) – (32) – проверяем, не является ли разделителем a[n-1].

В каждом из этих случаев запоминаем индекс разделителя в текущем элементе массива `b` (`b[k]`) и увеличиваем `k` (22), (27), (31). После окончания просмотра массива `a`, `k` – длина массива разделителей `b`.

(34) – (37) – вспомогательный вывод.

Так как в этой задаче может быть два варианта условия: подпоследовательности находятся между разделителями, и разделители находятся между подпоследовательностями, мы рассматриваем второй вариант как более сложный.

То есть, нам надо проверить случай, когда `b[0]=0` (левее `b[0]` нет подпоследовательности). и когда `b[k-1]=n-1` (правее `b[k-1]` нет подпоследовательности). В остальных случаях `b[i]` полностью задает очередную подпоследовательность: `b[i]+1` - индекс первого элемента подпоследовательности в `a`, `b[i+1]-1` - индекс последнего элемента подпоследовательности, `b[i+1]- b[i]+1` - длина подпоследовательности

Осталось найти подпоследовательность с наибольшей длиной `maxlen` сохраняя при этом индекс начала подпоследовательности `maxbeg`.

1.6. Приложения с двумерными массивами

1.6.1. Суммирование

Дана матрица целых чисел *a* размерности *n*×*m*. Вычислить сумму всех элементов матрицы.

```
(01)public static void main(String[] args)throws IOException {
(02)  int[][] a;
(03)  int n, m, i, j, sum;
(04)  BufferedReader br = new BufferedReader(new
(05)                InputStreamReader(System.in));
(06)  System.out.print("Введите количество строк матрицы:");
(07)  n=Integer.parseInt(br.readLine());
(08)  System.out.print("Введите количество столбцов матрицы:");
(09)  m=Integer.parseInt(br.readLine());
(10)  a=new int[n][m];
(11)  for(i=0;i<n;i++)
(12)    for(j=0;j<m;j++){
(13)      System.out.print("a["+i+","+j+"]=");
(14)      a[i][j]=Integer.parseInt(br.readLine());
(15)  sum=0;
(16)  for(i=0;i<n;i++)
(17)    for(j=0;j<m;j++)
(18)      sum=sum+a[i][j];
(19)  System.out.println("сумма все элементов матрицы равна "+sum);
(20)}
```

В отличие от одномерных массивов, элементы которых мы можем последовательно перебирать только двумя способами, элементы матрицы можно перебирать восьмью различными способами:

Вариант 1. По строкам начиная с первой, каждую строку с начала.

```
(15)  sum=0;
(16)  for(i=0;i<n;i++)
(17)    for(j=0;j<m;j++)
(18)      sum=sum+a[i][j];
```

Вариант 2. По строкам начиная с первой, каждую строку с конца.

```
(15)  sum=0;
(16)  for(i=0;i<n;i++)
(17)    for(j=m-1;j<=0;j--)
(18)      sum=sum+a[i][j];
```

Вариант 3. По строкам начиная с последней, каждую строку с начала.

```
(15)  sum=0;
(16)  for(i=n-1;i<=0;i--)
(17)    for(j=0;j<m;j++)
(18)      sum=sum+a[i][j];
```

Вариант 4. По строкам начиная с последней, каждую строку с конца.

```
(15)  sum=0;
(16)  for(i=n-1;i<=0;i--)
(17)    for(j=m-1;j<=0;j--)
(18)      sum=sum+a[i][j];
```

Вариант 5. По столбцам начиная с первого, каждый столбец с начала.

```
(15)  sum=0;
(16)  for(j=0;j<m;j++)
(17)    for(i=0;i<n;i++)
(18)      sum=sum+a[i][j];
```

Вариант 6. По столбцам начиная с первого, каждый столбец с конца.

```
(15)  sum=0;
(16)  for(j=0;j<m;j++)
(17)    for(i=n-1;i<=0;i--)
(18)      sum=sum+a[i][j];
```

Вариант 7. По столбцам начиная с последнего, каждый столбец с начала.

```
(15)  sum=0;
(16)  for(j=n-1;j<=0;j--)
(17)    for(i=0;i<m;i++)
(18)      sum=sum+a[i][j];
```

Вариант 8. По столбцам начиная с последнего, каждый столбец с конца.

```
(15)  sum=0;
(16)  for(j=m-1;j<=0;j--)
(17)    for(i=n-1;i<=0;i--)
(18)      sum=sum+a[i][j];
```

1.6.2. Поиск максимума (минимума)

Дана матрица целых чисел *a* размерности *n*×*m*. Вычислить наибольшее значение среди всех элементов матрицы.

```
(01)public static void main(String[] args)throws IOException {
(02)  int[][] a;
(03)  int n, m, i, j, max;
(04)  BufferedReader br = new BufferedReader(new
(05)                InputStreamReader(System.in));
(06)  System.out.print("Введите количество строк матрицы:");
(07)  n=Integer.parseInt(br.readLine());
(08)  System.out.print("Введите количество столбцов матрицы:");
(09)  m=Integer.parseInt(br.readLine());
(10)  a=new int[n][m];
(11)  for(i=0;i<n;i++)
(12)    for(j=0;j<m;j++){
(13)      System.out.print("a["+i+","+j+"]=");
(14)      a[i][j]=Integer.parseInt(br.readLine());
(15)  max=a[0][0];
(16)  for(i=0;i<n;i++)
(17)    for(j=0;j<m;j++)
(18)      if(max<a[i][j]) max=a[i][j];
(19)  System.out.println("наибольшее значение в матрицы равно "+max);
(20)}
```

Перебирать элементы матрицы можно различными способами, так же как в пункте 1.6.1. Для этого нужно изменить только строки (16) и (17).

Дана матрица целых чисел *a* размерности *n*×*m*. Вычислить наименьшее значение среди всех элементов матрицы.

```
(01)public static void main(String[] args)throws IOException {
(02)  int[][] a;
(03)  int n, m, i, j, min;
(04)  BufferedReader br = new BufferedReader(new
(05)                InputStreamReader(System.in));
(06)  System.out.print("Введите количество строк матрицы:");
(07)  n=Integer.parseInt(br.readLine());
(08)  System.out.print("Введите количество столбцов матрицы:");
(09)  m=Integer.parseInt(br.readLine());
(10)  a=new int[n][m];
(11)  for(i=0;i<n;i++)
(12)    for(j=0;j<m;j++){
(13)      System.out.print("a["+i+","+j+"]=");
(14)      a[i][j]=Integer.parseInt(br.readLine());
(15)    }
(16)  min=a[0][0];
(17)  for(i=0;i<n;i++)
(18)    for(j=0;j<m;j++)
(19)      if(min>a[i][j]) min=a[i][j];
(20)  System.out.println("наименьшее значение в матрицы равно "+min);
(21)}
```

Перебирать элементы матрицы можно различными способами, так же как в пункте 1.6.1. Для этого нужно изменить только строки (17) и (18).

1.6.3. Части матрицы

Кроме строк и столбцов при рассмотрении частей матрицы используются так называемые диагонали. Обычно диагонали рассматриваются только для квадратных матриц. Среди всех диагоналей особо выделяют две, главную:

$\boldsymbol{a_{00}}$	a_{01}	a_{02}	a_{03}	a_{04}
a_{10}	$\boldsymbol{a_{11}}$	a_{12}	a_{13}	a_{14}
a_{20}	a_{21}	$\boldsymbol{a_{22}}$	a_{23}	a_{24}

a_{30}	a_{31}	a_{32}	$\boldsymbol{a_{33}}$	a_{34}
a_{40}	a_{41}	a_{42}	a_{43}	$\boldsymbol{a_{44}}$

и побочная диагональ:

a_{00}	a_{01}	a_{02}	a_{03}	$\boldsymbol{a_{04}}$
a_{10}	a_{11}	a_{12}	$\boldsymbol{a_{13}}$	a_{14}
a_{20}	a_{21}	$\boldsymbol{a_{22}}$	a_{23}	a_{24}
a_{30}	$\boldsymbol{a_{31}}$	a_{32}	a_{33}	a_{34}
$\boldsymbol{a_{40}}$	a_{41}	a_{42}	a_{43}	a_{44}

Мы можем записать условие на индексы матрицы, при котором элемент будет находиться на диагонали. Если $i=j$, то элемент $a_{i\,j}$ находится на главной диагонали. Если $i+j=n-1$, то элемент $a_{i\,j}$ находится на побочной диагонали. Эти условия дают нам возможность записать алгоритм вычисления суммы всех элементов матрицы стоящих на главной диагонали, например, так:

```
for(i=0;i<n;i++)
  for(j=0;j<n;j++)
    if(i==j)s=s+a[i][j];
```

Вполне работоспособный, но очень не эффективный алгоритм. Количество элементов матрицы расположенных на главной диагонали равно `n`. Таким образом, нам необходим только один цикл:

```
for(i=0;i<n;i++)
  s=s+a[i][i];
```

Та же задача для побочной диагонали.

```
for(i=0;i<n;i++)
  for(j=0;j<n;j++)
    if(i+j==n-1)s=s+a[i][j];
```

Заметим, что количество элементов матрицы расположенных на побочной диагонали равно n, также как и для главной диагонали. Поэтому использовать алгоритм с двумя вложенными циклами с общим числом итераций n^2 крайне не эффективно. Достаточно одного цикла пробегающего по всем номерам строк i. Второй индекс легко находится из условия $i+j=n-1$, откуда $j=n-i-1$. Таким образом,

более эффективный алгоритм для нахождения суммы элементов матрицы стоящих на побочной диагонали имеет следующий вид:

```
for(i=0;i<n;i++)
  s=s+a[i][n-i-1];
```

Своими диагоналями матрица делится на треугольные части. Сами диагонали могут включаться или исключаться из этих треугольных частей матрицы. Найдем условие на индексы, при выполнении которого элемент $a_{i\,j}$ будет находиться выше главной диагонали. Это условие достаточно легко получить:

a_{00}	$\boldsymbol{a_{01}}$	$\boldsymbol{a_{02}}$	$\boldsymbol{a_{03}}$	$\boldsymbol{a_{04}}$
a_{10}	a_{11}	$\boldsymbol{a_{12}}$	$\boldsymbol{a_{13}}$	$\boldsymbol{a_{14}}$
a_{20}	a_{21}	a_{22}	$\boldsymbol{a_{23}}$	$\boldsymbol{a_{24}}$
a_{30}	a_{31}	a_{32}	a_{33}	$\boldsymbol{a_{34}}$
a_{40}	a_{41}	a_{42}	a_{43}	a_{44}

Замечаем, что у всех элементов расположенных выше главной диагонали первый индекс меньше второго: $i<j$. Более того, мы видим, что у первой диагонали расположенной выше главной a_{01}, a_{12}, a_{23}, a_{34} условие на индексы $i+1=j$. У второй диагонали a_{02}, a_{13}, a_{24} - $i+2=j$. И так далее. Мы воспользуемся этими условиями в дальнейшем, когда нам потребуется перебрать все диагонали параллельные главной. А сейчас запишем алгоритм, позволяющий перебрать все элементы матрицы расположенные выше главной диагонали:

```
for(i=0;i<n;i++)
  for(j=0;j<n;j++)
    if(i<j)s=s+a[i][j];
```

Недостатком этого алгоритма является то, что он выполняет более чем в два раза больше работы, чем требуется. Более изощренный, но и более эффективный алгоритм выглядит так:

```
for(i=0;i<n-1;i++)
  for(j=i+1;j<n;j++)
    s=s+a[i][j];
```

Индексные выражения в заголовках циклов находятся достаточно легко. Глядя на рисунок примера матрицы, замечаем, что последняя строка в нужную нам часть матрицы не входит. Поэтому заголовок цикла по номерам строк – `for(i=1;i<n-1;i++)`. Во внутреннем цикле начальное значение параметра зависит от номера строки, и на единицу больше его – `for(j=i+1;j<n;j++)`.

Рассмотрим теперь ту часть матрицы, которая расположена ниже главной диагонали:

a_{00}	a_{01}	a_{02}	a_{03}	a_{04}
$\boldsymbol{a_{10}}$	a_{11}	a_{12}	a_{13}	a_{14}
$\boldsymbol{a_{20}}$	$\boldsymbol{a_{21}}$	a_{22}	a_{23}	a_{24}
$\boldsymbol{a_{30}}$	$\boldsymbol{a_{31}}$	$\boldsymbol{a_{32}}$	a_{33}	a_{34}
$\boldsymbol{a_{40}}$	$\boldsymbol{a_{41}}$	$\boldsymbol{a_{42}}$	$\boldsymbol{a_{43}}$	a_{44}

Условие на индексы очевидно: $i>j$. Заметим так же, что для первой диагонали расположенной ниже главной a_{10}, a_{21}, a_{32}, a_{43} условие на индексы $i=j+1$. Для второй диагонали a_{20}, a_{31}, a_{42} условие – $i=j+2$. И так далее.

Первый, не очень эффективный алгоритм:

```
for(i=0;i<n;i++)
  for(j=0;j<n;j++)
    if(i>j)s=s+a[i][j];
```

и второй, более эффективный алгоритм:

```
for(i=1;i<n;i++)
  for(j=0;j<i;j++)
    s=s+a[i][j];
```

строятся аналогично рассмотренному выше примеру.

Рассмотрим теперь те части матрицы, которые получаются при делении ее побочной диагональю.

Найдем условие на индексы, при выполнении которого элемент $a_{i\,j}$ будет находиться выше побочной диагонали. Для этого внимательно посмотрим на матрицу:

$\boldsymbol{a_{00}}$	$\boldsymbol{a_{01}}$	$\boldsymbol{a_{02}}$	$\boldsymbol{a_{03}}$	a_{04}

$\boldsymbol{a_{10}}$	$\boldsymbol{a_{11}}$	$\boldsymbol{a_{12}}$	a_{13}	a_{14}
$\boldsymbol{a_{20}}$	$\boldsymbol{a_{21}}$	a_{22}	a_{23}	a_{24}
$\boldsymbol{a_{30}}$	a_{31}	a_{32}	a_{33}	a_{34}
a_{40}	a_{41}	a_{42}	a_{43}	a_{44}

Здесь условие не так очевидно. Однако, если вспомнить, что условие на индексы для самой побочной диагонали $i+j=n-1$, легко заметить, что это – $i+j<n-1$. Заодно находим условия на индексы для диагоналей расположенных параллельно побочной и выше нее. Для первой a_{03}, a_{12}, a_{21}, a_{30} условие $i+j=n-2$. Для второй a_{02}, a_{11}, a_{20} – $i+j=n-3$. И так далее.

Простой алгоритм:

```
for(i=0;i<n;i++)
  for(j=0;j<n;j++)
    if(i+j<n-1)s=s+a[i][j];
```

Более эффективный алгоритм:

```
for(i=0;i<n-1;i++)
  for(j=0;j<n-i-1;j++)
    s=s+a[i][j];
```

Рассмотрим теперь ту часть матрицы, которая расположена ниже побочной диагонали:

a_{00}	a_{01}	a_{02}	a_{03}	a_{04}
a_{10}	a_{11}	a_{12}	a_{13}	$\boldsymbol{a_{14}}$
a_{20}	a_{21}	a_{22}	$\boldsymbol{a_{23}}$	$\boldsymbol{a_{24}}$
a_{30}	a_{31}	$\boldsymbol{a_{32}}$	$\boldsymbol{a_{33}}$	$\boldsymbol{a_{34}}$
a_{40}	$\boldsymbol{a_{41}}$	$\boldsymbol{a_{42}}$	$\boldsymbol{a_{43}}$	$\boldsymbol{a_{44}}$

Теперь условие на индексы очевидно: $i+j>n-1$. Соответствующие алгоритмы строим аналогично. Первый:

```
for(i=0;i<n;i++)
  for(j=0;j<n;j++)
    if(i+j>n-1)s=s+a[i][j];
```

Второй:

```
for(i=1;i<n;i++)
  for(j=n-i;j<n;j++)
```

```
s=s+a[i][j];
```

Здесь некоторое затруднение может вызвать поиск начального значения для внутреннего цикла `for(j=n-i;j<n;j++)`. Для того, чтобы найти его вид воспользуемся рассмотренным выше методом. Имеем таблицу значений функции:

i	1	2	3	4
$f(i)$	4	3	2	1

Здесь n=5. Значение функции для i=1 равно n-1, и уменьшается на единицу при увеличении аргумента на единицу. Таким образом, $f(i)=n-i+k$. Анализируя таблицу, находим, что k=0.

В некоторых задачах требуется рассматривать более мелкие части матрицы расположенные: выше главной и выше побочной диагоналей; выше главной, но ниже побочной диагоналей; ниже главной и ниже побочной диагоналей; ниже главной, но выше побочной диагоналей:

a_{00}	$\boldsymbol{a_{01}}$	$\boldsymbol{a_{02}}$	$\boldsymbol{a_{03}}$	a_{04}
$\boldsymbol{a_{10}}$	a_{11}	$\boldsymbol{a_{12}}$	a_{13}	$\boldsymbol{a_{14}}$
$\boldsymbol{a_{20}}$	$\boldsymbol{a_{21}}$	a_{22}	$\boldsymbol{a_{23}}$	$\boldsymbol{a_{24}}$
$\boldsymbol{a_{30}}$	a_{31}	$\boldsymbol{a_{32}}$	a_{33}	$\boldsymbol{a_{34}}$
a_{40}	$\boldsymbol{a_{41}}$	$\boldsymbol{a_{42}}$	$\boldsymbol{a_{43}}$	a_{44}

Для поиска общего случая здесь нам одного примера будет не достаточно, поэтому рассмотрим еще один, для четного n:

a_{00}	$\boldsymbol{a_{01}}$	$\boldsymbol{a_{02}}$	$\boldsymbol{a_{03}}$	$\boldsymbol{a_{04}}$	a_{05}
$\boldsymbol{a_{10}}$	a_{11}	$\boldsymbol{a_{12}}$	$\boldsymbol{a_{13}}$	a_{14}	$\boldsymbol{a_{15}}$
$\boldsymbol{a_{20}}$	$\boldsymbol{a_{21}}$	a_{22}	a_{23}	$\boldsymbol{a_{24}}$	$\boldsymbol{a_{25}}$
$\boldsymbol{a_{30}}$	$\boldsymbol{a_{31}}$	a_{32}	a_{33}	$\boldsymbol{a_{34}}$	$\boldsymbol{a_{35}}$
$\boldsymbol{a_{40}}$	a_{41}	$\boldsymbol{a_{42}}$	$\boldsymbol{a_{43}}$	a_{44}	$\boldsymbol{a_{45}}$
a_{50}	$\boldsymbol{a_{51}}$	$\boldsymbol{a_{52}}$	$\boldsymbol{a_{53}}$	$\boldsymbol{a_{54}}$	a_{55}

Не эффективные алгоритмы получить достаточно просто. Достаточно объединить в условном операторе два условия. Например, для случая выше главной и выше побочной диагонали это – $i<j$ и $i+j<n$-1:

```
for(i=0;i<n;i++)
  for(j=0;j<n;j++)
```

```
    if(i<j && i+j<n-1)s=s+a[i][j];
```

Однако теперь не эффективный алгоритм хуже эффективного не в два раза, а в четыре.

Найдем более эффективный алгоритм для того же случая (выше главной и выше побочной диагоналей). Схема алгоритма такова:

```
for(i=0;i<f(n);i++)
  for(j=g(i);j<h(i);j++)
    s=s+a[i][j];
```

Для того, чтобы получить окончательный вид алгоритма, нам необходимо найти вид трех функций: $f(n)$, $g(i)$ и $h(i)$. Вторая и третья ищутся достаточно легко. Кроме того, мы их уже находили выше: $g(i)=i+1$ и $h(i)=n-i-1$. С поиском вида функции $f(n)$ придется повозиться. Для него не достаточно рассмотреть два примера при $n=5$ и $n=6$, для которых $f(5)=2$ и $f(6)=2$. Построим таблицу значений функции f для n от 3 до 10:

n	3	4	5	6	7	8	9	10
$f(n)$	0	0	1	1	2	2	3	3

Из этой таблицы видно, что искомая функция это $f(n)=(n-1)/2-1$. И окончательный вид алгоритма:

```
for(i=0;i<(n-1)/2-1;i++)
  for(j=i+1;j<n-i-1;j++)
    s=s+a[i][j];
```

Рассмотрим построение эффективного алгоритма для левого треугольника (ниже главной, но выше побочной диагоналей). На первый взгляд, кажется, что придется использовать две пары вложенных циклов:

```
for(int i=1;i<n/2;i++)
  for(int j=0;j<i;j++)
    s+=a[i][j];
for(int i=n/2;i<n-1:i++)
  for(int j=0;j<n-i-1;j++)
    s+=a[i][j];
```

Но если внимательно присмотреться к матрице, то видно, что левый треугольник становится верхним, если матрицу транспонировать. Для транспонирования квадратной матрицы достаточно поменять местами индексы строк и столбцов у всех элементов. На самом деле мы конечно матрицу транспонировать не будем, но эта мысль наталкивает нас на следующий алгоритм:

```
for(i=0;i<(n-1)/2-1;i++)
  for(j=i+1;j<n-i-1;j++)
    s=s+a[j][i];
```

Фактически в этом алгоритме `i` это номер столбца, а `j` – номер строки. Этот алгоритм конечно не намного эффективнее предыдущего, но зато он в два раза короче!

Оставляем в качестве самостоятельного задания разработку алгоритмов для нижнего и правого треугольников.

Рассмотрим теперь задачи, в которых требуется перебрать элементы матрицы стоящие на диагоналях параллельных главной (побочной). Заметим, что в квадратной матрице количество диагоналей параллельных главной (включая саму главную диагональ) равно $2n-1$. Столько же диагоналей параллельных побочной диагонали. Выше мы нашли условия на индексы для этих диагоналей. Суммируем их в следующей таблице:

Диагонали параллельные главной диагонали

1	*2*	…	*n-1*	***n***	*n+1*	…	*2n-2*	*2n-1*
i+n-1=j	*i+n-2=j*	…	*i+1=j*	***i=j***	*i-1=j*	…	*i-n+2=j*	*i-n+1=j*

Диагонали параллельные побочной диагонали

1	*2*	…	*n-1*	***n***	*n+1*	…	*2n-2*	*2n-1*
i+j=0	*i+j=1*	…	*i+j=n-2*	***i+j=n-1***	*i+j=n*	…	*i+j=2n-3*	*i+j=2n-2*

Эти условия можно использовать в алгоритме, который будет не очень эффективным, мягко говоря. Попытаемся построить более эффективный алгоритм перебора диагоналей параллельных побочной диагонали для следующей задачи. В квадратной матрице размерности n×n найти номер диагонали параллельной

побочной с наибольшей суммой элементов. Структура алгоритма выглядит следующим образом:

```
max=a[0][0];
imax=1;
for(k=2;k<=n;k++){
  sum=0;
  for(i=0;i<f1(k);i++)
    sum+=a[i][f2(i)];
  if(max<sum){
    max=sum;
    imax=k;
  }
}
for(k=n+1;k<=2*n-1;k++){
  sum=0;
  for(i=f3(k);i<n;i++)
    sum+=a[i][f4(i)];
  if(max<sum){
    max=sum;
    imax=k;
  }
}
```

Мы разделили просмотр всех диагоналей на два цикла: сначала первые n, затем оставшиеся n-1 потому, что номера строк в них изменяются по-разному. Нам не потребовалось вычислять сумму элементов первой диагонали, которую мы взяли в качестве начального значения максимума потому, что первая диагональ состоит из одного элемента a_{00}. Для того, чтобы получить окончательный вид алгоритма необходимо найти вид четырех функций: $f1(k)$, $f2(i)$, $f3(k)$ и $f4(i)$. Проще всего находится вид функции $f1(k)=k$. Для остальных будем анализировать таблицы значений.

i	0	1	2	…	k-2	k-1
$f2(i)$	k-1	k-2	k-3	…	1	0

Из таблицы видно, что $f2(i)=k-i-1$.

k	$n+1$	$n+2$	…	$2n-2$	$2n-1$
$f3(k)$	1	2	…	$n-2$	$n-1$

Из таблицы видно, что $f3(k)=k-n$.

Вид последней функции $f4(i)$ найдем без таблицы. Выше мы выписали условия на индексы для всех диагоналей параллельных побочной диагонали. В общем случае это условие выглядит так: $i+j=k-1$. Откуда $j=k-i-1$, то есть $f4(i)= k-i-1$. Кстати, таким же образом мы могли найти и вид функции $f2(i)$. Обе эти функции и $f2(i)$, и $f4(i)$ задают значение номера столбца по номеру строки для диагонали параллельной побочной диагонали.

В окончательном виде решение нашей задачи: среди диагоналей параллельных побочной диагонали квадратной матрицы размерности $n \times n$ найти диагональ с максимальной суммой компонентов, будет выглядеть так:

```
(01)public static void main(String[] args) throws IOException{
(02)  int [][] a;
(03)  int n, k, max, imax, sum;
(04)  BufferedReader br=new BufferedReader(new
(05)    InputStreamReader(System.in));
(06)  System.out.print("Введите размерность кв. матрицы");
(07)  n=Integer.parseInt(br.readLine);
(08)  a=new int[n][n];
(09)  for(int i=0;i<n;i++)
(10)    for(int j=0;j<n;j++){
(11)      System.out.print("a["+i+","+j+"]=");
(12)      a[i][j]=Integer.parseInt(br.readLine());
(13)    }
(14)  max=a[0][0];
(15)  imax=1;
(16)  for(k=2;k<=n;k++){
(17)    sum=0;
(18)    for(int i=0;i<k;i++)
```

```
(19)       sum+=a[i][k-i-1];
(20)     if(max<sum){
(21)       max=sum;
(22)       imax=k;
(23)     }
(24)   }
(25)   for(k=n+1;k<=2*n-1;k++){
(26)     sum=0;
(27)     for(int i=k-n;i<n;i++)
(28)       sum+=a[i][k-i-1];
(29)     if(max<sum){
(30)       max=sum;
(31)       imax=k;
(32)     }
(33)   }
(34)   System.out.println("диагональ "+imax+" имеет макс. сумму");
(35)}
```

Для задачи: среди диагоналей параллельных главной диагонали квадратной матрицы размерности $n \times n$ найти диагональ с максимальной суммой компонентов, решение выглядит так:

```
(01)public static void main(String[] args) throws IOException{
(02)   int [][] a;
(03)   int n, k, max, imax, sum;
(04)   BufferedReader br=new BufferedReader(new
(05)     InputStreamReader(System.in));
(06)   System.out.print("Введите размерность кв. матрицы");
(07)   n=Integer.parseInt(br.readLine);
(08)   a=new int[n][n];
(09)   for(int i=0;i<n;i++)
(10)     for(int j=0;j<n;j++){
(11)       System.out.print("a["+i+","+j+"]=");
(12)       a[i][j]=Integer.parseInt(br.readLine());
(13)     }
(14)   max=a[0][n-1];
```

```
(15)  imax=1;
(16)  for(k=2;k<=n;k++){
(17)    sum=0;
(18)    for(int i=0;i<k;i++)
(19)      sum+=a[i][n-k+i];
(20)    if(max<sum){
(21)      max=sum;
(22)      imax=k;
(23)    }
(24)  }
(25)  for(k=n+1;k<=2*n-1;k++){
(26)    sum=0;
(27)    for(int i=k-n;i<n;i++)
(28)      sum+=a[i][n-k+i];
(29)    if(max<sum){
(30)      max=sum;
(31)      imax=k;
(32)    }
(33)  }
(34)  System.out.println("диагональ "+imax+" имеет макс. сумму");
(35)}
```

1.6.4. Транспонирование матрицы

Дана квадратная матрица целых чисел *a* размерности *n*×*n*. Транспонировать матрицу (т.е. поменять местами строки и столбцы) не используя вспомогательного массива.

```
(01)public static void main(String[] args)throws IOException {
(02)  int[][] a;
(03)  int n, i, j, t;
(04)  BufferedReader br = new BufferedReader(new
(05)                InputStreamReader(System.in));
(06)  System.out.print("Введите размерность кв. матрицы:");
(07)  n=Integer.parseInt(br.readLine());
(08)  a=new int[n][т];
```

```
(09)  for(i=0;i<n;i++)
(10)    for(j=0;j<n;j++){
(11)      System.out.print("a["+i+","+j+"]=");
(12)      a[i][j]=Integer.parseInt(br.readLine());
(13)    }
(14)  for(i=1;i<n;i++)
(15)    for(j=0;j<i;j++){
(16)      t=a[i][j];
(17)      a[i][j]=a[j][i];
(18)      a[j][i]=t;
(19)    }
(20)  for(i=0;i<n;i++){
(21)    for(j=0;j<n;j++)
(22)      System.out.print(a[i][j]+" ");
(23)    System.out.println();
(24)  }
(25)}
```

1.6.5. Подматрица матрицы

Даны две матрицы целых чисел: *a* размерности $n \times m$ и *b* размерности $p \times q$ ($n>p$, $m>q$). Выяснить, является ли вторая матрица подматрицей первой.

```
(01)public static void main(String[] args)throws IOException {
(02)  int[][] a, b;
(03)  int n, m, p, q, i, j, i1, j1;
(04)  String s;
(05)  String[]as;
(06)  BufferedReader br = new BufferedReader(new
(07)                InputStreamReader(System.in));
(08)  System.out.print("Введите количество строк"+
(09)                " первой матрицы:");
(10)  n=Integer.parseInt(br.readLine());
(11)  System.out.print("Введите количество столбцов"+
(12)                " первой матрицы:");
```

```
(13)  m=Integer.parseInt(br.readLine());
(14)  a=new int[n][m];
(15)  for(i=0;i<n;i++){
(16)    System.out.println("Введите "+i+"-ую строку"+
(17)                       " через пробел");
(18)    s=br.readLine();
(19)    as=s.split(" ");
(20)    for(j=0;j<m;j++)
(21)      a[i][j]=Integer.parseInt(as[j]);
(22)  }
(23)  System.out.print("Введите количество строк"+
(24)                " второй матрицы:");
(25)  p=Integer.parseInt(br.readLine());
(26)  System.out.print("Введите количество столбцов"+
(27)                " второй матрицы:");
(28)  q=Integer.parseInt(br.readLine());
(29)  b=new int[p][q];
(30)  for(i=0;i<p;i++){
(31)    System.out.println("Введите "+i+"-ую строку"+
(32)                       " через пробел");
(33)    s=br.readLine();
(34)    as=s.split(" ");
(35)    for(j=0;j<q;j++)
(36)      b[i][j]=Integer.parseInt(as[j]);
(37)  }
(38)  boolean f=false;
(39)  m1:for(i=0;i<n-p+1;i++)
(40)    for(j=0;j<m-q+1;j++){
(41)      m2:for(i1=0;i1<p;i1++)
(42)        for(j1=0;j1<q;j1++)
(43)          if(a[i+i1][j+j1]!=b[i1][j1])break m2;
(44)      f=true;
(45)      break m1;
(46)    }
(47)  if(f)s="является ";else ="не является ";
(48)  System.out.println("Вторая матрица "+s+
```

```
(49)                          "подматрицей первой");
(50)}
```

(02) – объявляются две переменные типа двумерного массива целых чисел

(03) – объявляются переменные целого типа. `n` – количество строк матрицы `a`, `m` – количество столбцов матрицы `a`. `p` – количество строк матрицы `b`, `q` – количество столбцов матрицы `b`. `i` – номер строки матрицы `a`, `j` – номер столбца матрицы `a`, `i1` – номер строки матрицы `b`, `j1` – номер столбца матрицы `b`.

(06) – объявляется и инициализируется переменная `br` – буферизованный поток ввода с клавиатуры. В отличии от стандартного потока `System.in`, он позволяет вводить цепочку символов за один раз с помощью метода `br.readLine()`.

(10) - ввод с клавиатуры количества строк матрицы `a`. Метод `br.readLine()` возвращает строку (предположительно из цифр) набранную на клавиатуре, вплоть до нажатия клавиши ENTER. Метод `Integer.parseInt(br.readLine())` пытается преобразовать эту строку в целое число. Это число присваивается переменной `n`.

(13) - аналогично (10), только для переменной `m`.

(14) - переменная `a` доопределяется – в памяти создается матрица.

(15) – (22) – ввод значений элементов матрицы `a` с клавиатуры.

(18) – ввод с клавиатуры очередной `i` строки матрицы в переменную `s`.

(19) – с помощью метода `s.split(" ")` создаем массив строк `as`, в каждом элементе которого находится строковое представление элемента матрицы.

(21) преобразуем это строковое представление в целое число, и присваиваем переменной `a[i][j]`.

(23) – (37) – аналогично для матрицы `b`.

(38) переменной `f` присваиваем `false` – признак того, что `b` не является подматрицей `a`.

(39) - (40) – два вложенных цикла, задающих индексы начального элемента в матрице `a`, начиная с которого будет проводиться поиск подматрицы совпадающей с `b` (левый верхний элемент). Просматриваем не все строки и столбцы. Останавливаемся тогда, когда ниже и левее уже не может поместиться подматрица размерности как у `b`. Заголовок цикла `for(i=0;i<n-p+1;i++)` помечен для того, чтобы можно было преждевременно выйти из вложенных циклов с помощью оператора `break m1`.

(41) - (42) – два вложенных цикла, задающих индексы текущего элемента в матрице b. Эти же индексные переменные используются для задания смещения в матрице `a` (`a[i+i1][j+j1]` – `i` и `j` задают начальную точку, `i1` и `j1` задают смещение). Заголовок цикла `for(i1=0;i1<p;i1++)` помечен для того, чтобы можно было преждевременно выйти из вложенных циклов с помощью оператора `break m2`.

(43) – если `a[i+i1][j+j1]` не равен `b[i1][j1]`, то очередной претендент на подматрицу не подошел, надо сдвинуть начальную точку по матрице `a`.

(44) если циклы по `i1` и `j1` завершились нормально (не сработал преждевременный выход), то подматрица найдена. Фиксируем это событие в переменной `f` и выходим из вложенных циклов (45).

(47) - (49) – выводим результат

2. Приложения с визуальным интерфейсом

2.1. Простейшее приложение с визуальным интерфейсом

```
(01)package simpleguidemo;
(02)import java.util.Random;
(03)/* Простое приложение с графическим интерфейсом */
(04)public class SimpleGUIDemo extends javax.swing.JFrame {
(05)// Кнопка заполнить
(06)  private void jButton1ActionPerformed(ActionEvent evt) {
(07)    Random r = new Random();
(08)    Integer x;
(09)    x = r.nextInt(100)-50;
(10)    jTextField1.setText(x.toString());
(11)    x = r.nextInt();
(12)    jTextField2.setText(x.toString());
(13)  }
(14)// Кнопка вычислить
(15)  private void jButton2ActionPerformed(ActionEvent evt){
(16)    int x, y, sum;
(17)    x=Integer.parseInt(jTextField1.getText());
(18)    y=Integer.parseInt(jTextField2.getText());
(19)    sum=x+y;
(20)    //jTextArea1.append(x + "\t+ "+y+"\t = "+sum+"\n");
(21)    jTextArea1.setText(x + "\t+ "+y+"\t = "+
(22)                       sum+"\n"+jTextArea1.getText());
(23)  }
(24)}
```

2.2. Примеры визуального интерфейса для ввода/вывода массивов

Пример визуального интерфейса для ввода/вывода вектора.

```
private void jButton1ActionPerformed(ActionEvent evt){
  String strIn, strOut="";
  strIn=jTextArea1.getText();
  String as[]=strIn.split(" ");
  int n=as.length;
  int a[]=new int[n];
  for(int i=0;i<n;i++)
    a[i]=Integer.parseInt(as[i]);
  for(int i=0;i<n;i++)
    strOut=strOut+a[n-i-1]+" ";
  strOut=strOut+"\n";
  jTextArea2.setText(jTextArea2.getText()+strOut);
}
```

Пример визуального интерфейса для ввода/вывода матрицы.

```
private void jButton2ActionPerformed(ActionEvent evt){
  String strIn, strOut="";
  strIn=jTextArea1.getText();
  String as[]=strIn.split("\n");
  int n=as.length;
  String row[]=as[0].split(" ");
  int m=row.length;
  int a[][]=new int[n][m];
  for(int i=0;i<n;i++){
    row=as[i].split(" ");
    for(int j=0;j<m;j++)
      a[i][j]=Integer.parseInt(row[j]);
  }
  for(int i=0;i<n;i++){
    for(int j=0;j<m;j++)
      strOut=strOut+a[i][j]+" ";
    strOut=strOut+"\n";
  }
  jTextArea2.setText(jTextArea2.getText()+strOut);
```

```
}
```

2.3. Многострочное текстовое окно

В визуальном интерфейсе этого приложения два управляющих элемента: многострочное текстовое окно (`jTextArea1`) и командная кнопка (`jButton1`). Многострочное текстовое окно позволяет вводить и выводить текст из нескольких строк (т. е. в тексте правильно интерпретируются непечатаемые символы перехода на начало следующей строки `"\n"` (которые появляются в тексте принажатии клавиши ENTER).

В приложении иллюстрируется использование стандартного выделения части текста Windows. При каждом нажатии кнопки (`jButton1`) выделяется очередная строка текста в текстовом окне (`jTextArea1`).

```
(01)int start;
(02)
(03)private void jButton1ActionPerformed(ActionEvent evt){
(04)  int end;
(05)  String s=jTextField1.getText();
(06)  end=s.indexOf("\n",start);
(07)  jTextArea1.requestFocus();
(08)  jTextArea1.setSelectionStart(start);
(09)  jTextArea1.setSelectionEnd(end);
(10)  start=end+1;
(11)  if(start>s.length()) start=0;
(12)}
(13)
(14)private void formWindowOpened(WindowEvent evt){
(15)  start=0;
(16)}
```

(01) – переменная `start` служит для указания номера символа в тексте, начиная с которого будет осуществляться поиск ответа (в нашем простом случае – начало

ответа). Объявляется она вне метода обработчика события «нажатие кнопки», так как ее значение должно сохраняться до следующего вызова метода.

(03) – заголовок обработчика события «нажатие кнопки».

(04) – переменная `end` служит для указания номера символа в тексте, которым закачивается ответ.

(05) – присваиваем строковой переменной `s` содержимое текстового окна (это значение типа `String` , которое содержит все символы из текстового окна, включая непечатаемые символы перехода на начало следующей строки).

(06) – присваиваем переменной `end` номер первого найденного символа `"\n"` в `jTextArea1` , если начинать поиск с символа с номером `start`.

(07) – устанавливаем фокус клавиатуры на текстовое окно. Если окно будет не в фокусе, выделение не будет видно.

(08) – задаем начало выделения.

(09) – задаем конец выделения.

(10) – готовим начало поиска ответа для следующего вызова метода.

(11) – если текущий ответ последний в тексте, готовимся в следующий раз искать ответ с начала.

(14) – заголовок обработчика события «начало загрузки окна приложения». Он нужен для инициализации переменной `start`.

2.4. Одномерный массив текстовых окон

Визуальный интерфейс. Одномерный массив, представленный массивом текстовых окон.

```
(01)package visualvectorproject;
(02)import java.awt.Color;
(03)import javax.swing.*;
(04)import java.util.Random;
(05)
(06)public class VisualVectorJFrame extends javax.swing.JFrame {
(07)
(08)  private JTextField[] visVector;
```

```
(09)
(10)  private void formWindowOpened(WindowEvent evt) {
(11)    jButton2.setVisible(false);
(12)    jButton3.setVisible(false);
(13)  }
(14)
(15)  private void jButton1ActionPerformed(ActionEvent evt) {
(16)    // Кнопка "Ввод" - создать массив окон
(17)    int n = Integer.parseInt(jTextField1.getText());
(18)    if(visVector!=null){
(19)      if(visVector.length!=0){
(20)        for(int i=0; i<visVector.length; i++){
(21)          visVector[i].setVisible(false);
(22)          visVector[i]=null;
(23)        }
(24)      }
(25)    }
(26)    visVector = new JTextField[n];
(27)    for(int i=0; i<n;i++){
(28)      visVector[i] = new JTextField();
(29)      jPanel2.add(visVector[i]);
(30)      visVector[i].setBounds(5+i*30, 5, 30, 25);
(31)      visVector[i].setHorizontalAlignment(JTextField.RIGHT);
(32)      visVector[i].setText(Integer.toString(i+1));
(33)    }
(34)    jButton2.setVisible(true);
(35)    jButton3.setVisible(true);
(36)  }
(37)  private void jButton2ActionPerformed(ActionEvent evt) {
(38)    // Кнопка "Заполнить"
(39)    Random r = new Random();
(40)    Integer x;
(41)    for(int i=0; i<visVector.length; i++){
(42)      x = r.nextInt(100)-50;
(43)      visVector[i].setText(x.toString());
(44)    }
```

```
(45)  }
(46)
(47)  private void jButton3ActionPerformed(ActionEvent evt) {
(48)    // Кропка "Вычислить"
(49)    int n = visVector.length;
(50)    int a[], max, imax;
(51)    a = new int[n];
(52)    for(int i=0; i<n; i++){
(53)      visVector[i].setBackground(
(54)        jTextField1.getBackground());
(55)      visVector[i].setForeground(
(56)        jTextField1.getForeground());
(57)      a[i] = Integer.parseInt(visVector[i].getText());
(58)    }
(59)    max = a[0]; imax = 0;
(60)    for(int i=1; i<n; i++)
(61)      if(max < a[i]){
(62)        max = a[i];
(63)        imax = i;
(64)      }
(65)    visVector[imax].setBackground(Color.BLUE);
(66)    visVector[imax].setForeground(Color.WHITE);
(67)  }
(68)}
```

2.5. Двумерный массив текстовых окон

```
(01)package visualmatrixproject;
(02)import java.awt.Color;
(03)import javax.swing.*;
(04)import java.util.Random;
(05)
(06)public class VisualMatrixJFrame extends javax.swing.JFrame {
```

```
(07)
(08)  private JTextField[][] visMatrix;
(09)
(10)  private void formWindowOpened(WindowEvent evt) {
(11)    jButton2.setVisible(false);
(12)    jButton3.setVisible(false);
(13)  }
(14)
(15)  private void jButton1ActionPerformed(ActionEvent evt) {
(16)    // Кнопка "Ввод" - создать массив окон
(17)    int n = Integer.parseInt(jTextField1.getText());
(18)    int m = Integer.parseInt(jTextField2.getText());
(19)    if(visMatrix!=null){
(20)      if(visMatrix.length!=0){
(21)        for(int i=0; i<visMatrix.length; i++)
(22)          for(int j=0;j<visMatrix[i].length;j++){
(23)            visMatrix[i][j].setVisible(false);
(24)            visMatrix[i][j]=null;
(25)          }
(26)      }
(27)    }
(28)    visMatrix = new JTextField[n][m];
(29)    for(int i=0; i<n;i++)
(30)      for(int j=0;j<m;j++){
(31)        visMatrix[i][j] = new JTextField();
(32)        jPanel2.add(visMatrix[i][j]);
(33)        visMatrix[i][j].setBounds(5+j*30, 5+i*25, 30, 25);
(34)        visMatrix[i][j].setHorizontalAlignment(JTextField.RIGHT);
(35)        visMatrix[i][j].setText(Integer.toString(i*m+j+1));
(36)      }
(37)    jButton2.setVisible(true);
(38)    jButton3.setVisible(true);
(39)  }
(40)
(41)  private void jButton2ActionPerformed(ActionEvent evt) {
(42)    // Кнопка "Заполнить"
```

```
(43)    Random r = new Random();
(44)    Integer x;
(45)    for(int i=0;i<visMatrix.length;i++)
(46)      for(int j=0;j<visMatrix[i].length;j++){
(47)        x = r.nextInt(100)-50;
(48)        visMatrix[i][j].setText(x.toString());
(49)      }
(50)  }
(51)
(52)  private void jButton3ActionPerformed(ActionEvent evt) {
(53)    // Кропка "Вычислить"
(54)    int n = visMatrix.length;
(55)    int m=visMatrix[0].lengtf;
(56)    int a[][], max, imax, jmax;
(57)    a = new int[n][m];
(58)    for(int i=0;i<n;i++)
(59)      for(int j=0;j<m;j++){
(60)        visMatrix[i][j].setBackground(
(61)                          jTextField1.getBackground());
(62)        visMatrix[i][j].setForeground(
(63)                          jTextField1.getForeground());
(64)        a[i][j] = Integer.parseInt(visMatrix[i][j].getText());
(65)      }
(66)    max = a[0][0]; imax = 0; jmax=0;
(67)    for(int i=0;i<n;i++)
(68)      for(int j=0;j<m;j++)
(69)        if(max < a[i][j]){
(70)          max=a[i][j];
(71)          imax=i;
(72)          jmax=j;
(73)        }
(74)    visMatrix[imax][jmax].setBackground(Color.BLUE);
(75)    visMatrix[imax][jmax].setForeground(Color.WHITE);
(76)  }
(77)
(78)}
```

2.6. Графика в Java

2.6.1. Вывод текста по центру области рисования

Вывод текста по центру области рисования

```
(01)private Graphics g;
(02)
(03)private void jButton1ActionPerformed(ActionEvent evt){
(04)  //Кнопка «Вывести»
(05)  Dimension d=jPanel1.getSize();
(06)  g.clearRect(0, 0, d.width, d.height);
(07)  g.drawRect(1, 1, d.width-1, d.height-1);
(08)  g.drawLine(0, d.height/2, d.width, d.height/2);
(09)  g.drawLine(d.width/2, 0, d.width/2, d.height);
(10)  Font f=g.getFont();
(11)  g.setFont(new Font(f.getFontName(), f.getStyle(), 36));
(12)  FontMetrics fm=g.getFontMetrics();
(13)  String s=jTextField1.getText();
(14)  int h=fm.getHeight();
(15)  int w=fm.stringWidth(s);
(16)  g.drawString(s, (d.width-w)/2, (d.height-h)/2);
(17)}
(18)
(19)private void formWindowOpened(WindowEvent evt){
(20)  g=jPanel1.getGraphics();
(21)}
```

Этот пример иллюстрирует использование класса `FontMetrics` в котором есть методы позволяющие получить различные размеры текущего шрифта области рисования, и размеры текста набранного этим шрифтом.

2.6.2. График функции

Визуальный интерфейс. Вывод графика функции.

```
(01)import java.awt.*;
(02)/* График функции */
(03)public class FunctionGraphJFrame extends javax.swing.JFrame {
(04)  private Graphics g;
(05)
(06)  jButton1ActionPerformed(java.awt.event.ActionEvent evt) {
(07)    //Кнопка - Вывести график функции
(08)    g = jPanel1.getGraphics();
(09)    Dimension d = jPanel1.getSize();
(10)    Color bg = jPanel1.getBackground();
(11)    Color fg = jPanel1.getForeground();
(12)    g.setColor(bg);
(13)    g.fillRect(0, 0, d.width, d.height);
(14)    g.setColor(fg);
(15)    //Рисуем оси
(16)    //Ось X
(17)    g.drawLine(0, d.height/2, d.width, d.height/2);
(18)    //Ось У
(19)    g.drawLine(d.width/2, 0, d.width/2, d.height);
(20)    int a = Integer.parseInt(jTextField1.getText());
(21)    int w = 2*a;
(22)    //Шаг приращения по оси X на один пиксель
(23)    double h = (double)w/d.width;
(24)    //Сколько пикселей на одну единицу масштаба
(25)    double hh = (double)d.width/w;
(26)    double x = (double)(-a);
(27)    double y=-Math.sin(x), y1;
(28)    for(int i=0; i<d.width;i++){
(29)      y1=y;
(30)      y=-Math.sin(x);
(31)      g.drawLine(i,(int)(y1*hh)+d.height/2,i,
```

```
(32)                    (int)(y*hh)+d.height/2);
(33)       x += h;
(34)     }
(35)   }
(36)}
```

2.6.3. Интерактивная графика. Ломаная

```
(01)int[] xx, yy;
(02)Graphics g;
(03)
(04)private void jPanel1MousePressed(java.awt.event.MouseEvent evt){
(05)   increase();
(06)   int x=xx.length-1;
(07)   xx[n]=evt.getX();
(08)   yy[n]=evt.getY();
(09)   draw();
(10)}
(11)
(12)private void draw(){
(13)   int i, n=xx.length-1;
(14)   g.setColor(Color.RED);
(15)   g.fillOval(xx[0]-2,yy[0]-2,4,4);
(16)   for(i=0;i<n;i++){
(17)     g.setColor(Color.BLACK);
(18)     g.drawLine(xx[i],yy[i],xx[i+1],yy[i+1]);
(19)     g.setColor(Color.RED);
(20)     g.fillOval(xx[i+1]-2,yy[i+1]-2,4,4);
(21)   }
(22)}
(23)
(24)private void increase(){
(25)   //Увеличить длину xx и yy на 1
```

```
(26)  int [] x0, y0;
(27)  int i, n;
(28)  if(xx==null){
(29)    xx=new int[1];
(30)    yy=new int[1];
(31)  }else{
(32)    n=xx.length;
(33)    x0=new int[n];
(34)    y0=new int[n];
(35)    for(i=0;i<n;i++){
(36)      x0[i]=xx[i];
(37)      y0[i]=yy[i];
(38)    }
(39)    xx=new int[n+1];
(40)    yy=new int[n+1];
(41)    for(i=0;i<n;i++){
(42)      xx[i]=x0[i];
(43)      yy[i]=y0[i];
(44)    }
(45)  }
(46)}
(47)
(48)private void decrease(){
(49)  //Уменьшить длину xx и yy на 1
(50)  int [] x0, y0;
(51)  int i, n;
(52)  if(xx==null)return;
(53)  n=xx.length-1;
(54)  x0=new int[n];
(55)  y0=new int[n];
(56)  for(i=0;i<n;i++){
(57)    x0[i]=xx[i];
(58)    y0[i]=yy[i];
(59)  }
(60)  xx=new int[n];
(61)  yy=new int[n];
```

```
(62)  for(i=0;i<n;i++){
(63)    xx[i]=x0[i];
(64)    yy[i]=y0[i];
(65)  }
(66)}
(67)
(68)private void deleteByNumber(int num){
(69)  //Удалить из xx[num] и yy[num]
(70)  int n=xx.length;
(71)  if(num<0||num>n) return;
(72)  for(int i=num;i<n-1;i++){
(73)    xx[i]=xx[i+1];
(74)    yy[i]=yy[i+1];
(75)  }
(76)  decrease();
(77)}
(78)
(79)private void addByNumber(int num, int x, int y){
(80)  //Добавить xx[num]=x и yy[num]=y
(81)  increase();
(82)  int n=xx.length;
(83)  for(int i=n-1;i>num;i--){
(84)    xx[i]=xx[i-1];
(85)    yy[i]=yy[i-1];
(86)  }
(87)  xx[num]=x;
(88)  yy[num]=y;
(89)}
(90)
(91)private void formWindowOpened(java.awt.event.WindowEvent evt){
(92)  g=jPanel1.getGraphics();
(93)}
```

2.6.4. Моделирование текстового курсора в области рисования

Графический интерфейс. Ввод символов в область рисования с клавиатуры.

Моделирование собственного текстового курсора.

```
//Перемещение прямоугольного курсора по Панели
//Ввод цифр в курсор с клавиатуры
//Курсор появляется когда Панель получает фокус клавиатуры и исчезает
//когда теряет
//В текстовое окно можно ввести размер шрифта (по умолчанию – 12 пт)

(01)import java.awt.Graphics;
(02)import java.awt.Color;
(03)import java.awt.FontMetrics;
(04)import java.awt.Dimension;
(05)import java.awt.event.KeyEvent;
(06)import java.awt.Font;
(07)
(08)public class GraphicsCursorJFrame extends javax.swing.JFrame{
(09)
(10)  private Graphics g;
(11)  private int x, y;
(12)
(13)  private void formWindowOpened(WindowEvent evt){
(14)    //Инициализации при открытии jFrame
(15)    g=jPanel1.getGraphics();
(16)    x-0;
(17)    y=0;
(18)    jPanel1.setFocusable(true);
(19)    Font ft=g.getFont();
(20)    Integer fs=ft.getSize();
(21)    jFextField1.setText(fs.toString());
(22)  }
(23)
(24)  private void jPanel1KeyPressed(KeyEvent evt){
(25)    //Панель в фокусе. Нажата клавиша на клавиатуре
```

```
(26)    boolean f=true
(27)    String s="", ss;
(28)    Color bg=jPanel1.getBackground();
(29)    FontMetrics metrics=g.getFontMetrics();
(30)    int w=metrics.charWidth('9');
(31)    int h=metrics.getHeight();
(32)    Dimension d=jPanel1.getSize();
(33)    Integer code=evt.getKeyCode();
(34)    Character cchar=evt.getKeyChar();
(35)    s=cchar.toString();
(36)    ss=s.substring(0,1);
(37)    if(!Character.isLetterOrDigit(cchar)) f=false;
(38)    g.setColor(bg);
(39)    g.drawRect(x, y, w+2, h);
(40)    g.setColor(Color.BLUE);
(41)    switch(code){
(42)      case KeyEvent.VK_DOWN:
(43)        if(y<d.height-h) y+=h;
(44)        f=false;
(45)        break;
(46)      case KeyEvent.VK_RIGHT:
(47)        if(x<d.width-(w+2)) x+=w+2;
(48)        f=false;
(49)        break;
(50)      case KeyEvent.VK_UP:
(51)        if(!(y<h)) y-=h;
(52)        f=false;
(53)        break;
(54)      case KeyEvent.VK_LEFT:
(55)        if(!(x<w)) x-=w+2;
(56)        f=false;
(57)    }
(58)    g.drawRect(x, y, w+2, h);
(59)    if(f){
(60)      g.setColor(bg);
(61)      g.fillRect(x+1, y+1, w+1, h-1);
```

```
(62)       g.setColor(Color.BLACK);
(63)       g.drawString(ss, x+1, y+metrics.getAscent());
(64)     }
(65)   }
(66)
(67)   private void jPanel1FocusGained(FocusEvent evt){
(68)     //Панель получила фокус – рисуем прямоугольный курсор
(69)     FontMetrics metrics=g.getFontMetrics();
(70)     int w=metrics.charWidth('9');
(71)     int h=metrics.getHeight();
(72)     g.setColor(Color.BLUE);
(73)     g.drawRect(x, y, w+2, h);
(74)   }
(75)
(76)   private void jPanel1FocusLost(FocusEvent evt){
(77)     //Панель теряет фокус – стираем курсор
(78)     FontMetrics metrics=g.getFontMetrics();
(79)     int w=metrics.charWidth('9');
(80)     int h=metrics.getHeight();
(81)     g.setColor(jPanel1.getBackground());
(82)     g.drawRect(x, y, w+2, h);
(83)   }
(84)
(85)   private void jTextField1FocusLost(FocusEvent evt){
(86)     //Текстовое окно теряет фокус
(87)     //Берем из него число и меняем размер шрифта
(88)     Font ft=g.getFont();
(89)     Font ft1=new Font(ft.getFontName(), ft.getStyle(),
(90)       Integer.parseInt(jTextField1.getText()));
(91)     g.setFont(ft1);
(92)   }
(93)}
```

3. Файлы

3.1. Файл последовательного доступа

```
(01)private void jButton1ActionPerformed(ActionEvent evt){
(02)  // Кнопка "Записать"
(03)  try{
(04)    BufferedWriter bw = new BufferedWriter(
(05)      new OutputStreamWriter(new FileOutputStream("data.txt")));
(06)    String s = jTextArea1.getText();
(07)    bw.write(s);
(08)    bw.close();
(09)    jTextArea1.setText(jTextArea1.getText()+
(10)      "\nВывод в файл завершен...");
(11)  }catch (Exception ex){
(12)    Sysem.out.println(ex.toString());
(13)  }
(14)}
(15)
(16)private void jButton2ActionPerformed(ActionEvent evt){
(17)  // Кнопка "Прочитать"
(18)  try{
(19)    BufferedReader br = new BufferedReader(
(20)      new InputStreamReader(new FileInputStream("data.txt")));
(21)    String s;
(22)    jTextArea1.setText("Содержимое файла:");
(23)    while((s=br.readLine())!=null)
(24)      jTextArea1.setText(jTextArea1.getText()+"\n"+s);
(25)  }catch (Exception ex){
(26)    Sysem.out.println(ex.toString());
(27)  }
(28)}
```

```
(29)
(30)private void jButton3ActionPerformed(ActionEvent evt){
(31)  // Кнопка "Вывести"
(32)  String s;
(33)  int n, i, count;
(34)  try{
(35)    BufferedReader br = new BufferedReader(
(36)      new InputStreamReader(new FileInputStream("data.txt")));
(37)    String []as;
(38)    int []a;
(39)  /*jText Area1.setText("Вывод\nСодержимого файла:");
(40)    while((s=br.readLine())!=null)
(41)      jTextArea1.setText(jTextArea1.getText()+"\n"+s);
(42)      br.reset();
(43)  */
(44)    jTextArea1.setText(jTextArea1.getText()+"\nОтвет:");
(45)    s = br.readLine();
(46)    as = s.split(" ");
(47)    n = as.length;
(48)    a = new int[n];
(49)    count = 0;
(50)    for(i=0;i<n;i++)
(51)    a[i] = Integer.parseInt(as[i]);
(52)    for(i=0;i<n;i++)
(53)      if(a[i] > 0) count += 1;
(54)    jTextArea1.setText(jTextArea1.getText()+"\n"+count);
(55)  }catch (Exception ex){
(56)    Sysem.out.println(ex.toString());
(57)  }
(58)}
```

3.2. Файл прямого доступа

Файл прямого доступа

```
(001)finale int LEN = 75;     //Длина записи
(002)
(003)private String space(int n){
(004)  //Возвращает строку из n пробелов
(005)  char [] c = new char [n];
(006)  for(int i=0;i<n;i++)
(007)    c[i] = ' ';
(008)  return new String(c);
(009)}
(010)
(011)private void writeToFile(int num){
(012)  //Записать в файл запись с номером num
(013)  RandomAccessFile f;
(014)  long fPointer;
(015)  String s = jTextField3.getText()+" "+//Фамилия
(016)    jTextField4.getText()+" "+              //Имя
(017)    jTextField5.getText()+" "+              //Отчество
(018)    jTextField6.getText()+" "+              //Пол
(019)    jTextField7.getText()+" "+              //Год рождения
(020)    jTextField8.getText()+" "+              //Оценка 1
(021)    jTextField9.getText()+" "+              //Оценка 2
(022)    jTextField10.getText()+" "+"\n";    //Оценка 3
(023)  s = s + space(75).subString(0, 75-s.length);
(024)  try{
(025)    f = new RandomAccessFile("group2121.dat", "rw");
(026)    fPointer = (num-1)*LEN;
(027)    f.seek(fPointer);
(028)    f.writeBytes(s);
(029)  }catch(Exception ex){
(030)    Sysem.out.println(ex.toString());
(031)  }
(032)}
```

```
(033)
(034)private void readFromFile(int num){
(035)  //Прочитать из файла запись с номером num
(036)  RandomAccessFile f;
(037)  long fPointer = 0;
(038)  String s;
(039)  try{
(040)    f = new RandomAccessFile("group2121.dat", "r");
(041)    fPointer = (num-1)*LEN;
(042)    f.seek(fPointer);
(043)    s = f.readLine();
(044)    String []ss = s.split(" ");
(045)    jTextField3.setText(ss[0]);
(046)    jTextField4.setText(ss[1]);
(047)    jTextField5.setText(ss[2]);
(048)    jTextField6.setText(ss[3]);
(049)    jTextField7.setText(ss[4]);
(050)    jTextField8.setText(ss[5]);
(051)    jTextField9.setText(ss[6]);
(052)    jTextField10.setText(ss[7]);
(053)  }catch(Exception ex){
(054)    System.out.println(ex.toString());
(055)  }
(056)}
(057)
(058)private void jButton1ActionPerformed(ActionEvent evt){
(059)  // Кнопка "Записать"
(060)  int num = Integer.parseInt(jTextField1.getText());
(061)  writeToFile(num);
(062)}
(063)
(064)private void jButton2ActionPerformed(ActionEvent evt){
(065)  // Кнопка "Прочитать"
(066)  int num = Integer.parseInt(jTextField1.getText());
(067)  readFromFile(num);
(068)}
```

```
(069)
(070)private void jButton3ActionPerformed(ActionEvent evt){
(071)  // Кнопка "Вывести"
(072)  RandomAccessFile f;
(073)  long fPointer = 0;
(074)  String s;
(075)  long n;
(076)  int num = Integer.parseInt(jTextField1.getText());
(077)  try{
(078)    f = new RandomAccessFile("group2121.dat", "rw");
(079)    n = f.length()/LEN;  //Количество записей в файле
(080)    jTextField2.setText(Integer.toString(n));
(081)    while(n > 0){
(082)      s = f.readLine()+"\n"+
(083)          f.readLine()+"\n"+
(084)          f.readLine()+"\n"+
(085)          f.readLine()+"\n"+
(086)          f.readLine()+"\n"+
(087)          f.readLine()+"\n"+
(088)          f.readLine()+"\n\n";
(089)      jTextArea1.setText(jTextArea1.getText()+s);
(090)      n--;
(091)    }
(092)  }catch(Exception ex){
(093)    System.out.println(ex.toString());
(094)  }
(095)}
(096)
(097)private void jButton4ActionPerformed(ActionEvent evt){
(098)  //Кнопка "Вправо"
(099)  int maxNum = Integer.parseInt(jTextField2.getText());
(100)  int num = Integer.parseInt(jTextField1.getText());
(101)  if(num < maxNum){
(102)    num++;
(103)    jTextField1.setText(Integer.toString(num));
(104)    readFromFile(num);
```

```
(105)  }
(106)}
(107)
(108)private void jButton5ActionPerformed(ActionEvent evt){
(109)  //Кнопка "Влево"
(110)  int num = Integer.parseInt(jTextField1.getText());
(111)  num--;
(112)  if(num < 1) num = 1;
(113)  jTextField1.setText(Integer.toString(num));
(114)  readFromFile(num);
(115)}
(116)
(117)private void jButton6ActionPerformed(ActionEvent evt){
(118)  //Кнопка "На первую запись"
(119)  int num = 1;
(120)  jTextField1.setText(Integer.toString(num));
(121)  readFromFile(num);
(122)}
(123)
(124)private void jButton7ActionPerformed(ActionEvent evt){
(125)  //Кнопка "На последнюю запись"
(126)  int num = Integer.parseInt(jTextField2.getText());
(127)  jTextField1.setText(Integer.toString(num));
(128)  readFromFile(num);
(129)}
(130)
(131)private void jButton8ActionPerformed(ActionEvent evt){
(132)  //Кнопка "Добавить запись"
(133)  int num = Integer.parseInt(jTextField2.getText());
(134)  num++;
(135)  writeToFile(num);
(136)  jTextField1.setText(Integer.toString(num));
(137)  jTextField2.setText(Integer.toString(num));
(138)}
```

4. Параллельные процессы

Рассмотрим простой пример использования процессов в Java, механизма который позволяет реализовать параллельное программирование.

Создадим электронный секундомер:

```
(01)OutThread th;
(02)Integer ind=0;
(03)
(04)private void jButton1ActionPerformed(ActionEvent evt){
(05)  //Кнопка Запустить
(06)  th=new OutThread();
(07)  th.start();
(08)}
(09)
(10)private void jButton2ActionPerformed(ActionEvent evt){
(11)  //Кнопка Остановить
(12)  th.stop;
(13)}
(14)
(15)private void jButton3ActionPerformed(ActionEvent evr){
(16)  //Кнопка Сбросить
(17)  ind=0;
(18)}
(19)
(20)//Класс процесса – СЕКУНДОМЕР
(21)class OutThread extends Thread{
(22)
(23)  public void run(){
(24)    while(true){
(25)      try{
(26)        Thread.sleep(1000);
(27)        ind++;
(28)        jTextField1.setText(ind.toString());
(29)      }catch(InterruptedException ie){}
(30)    }
```

```
(31)   }
(32)}
```

5. Классы

Классы.

Файл `Pair.java` – объявление класса «Пара чисел»

```
(01)package classdemo;
(02)public class Pair{
(03)    private float x;
(04)    private float y;
(05)
(06)    Pair(){
(07)        x=0;
(08)        y=0;
(09)    }
(10)
(11)    Pair(float a, float b){
(12)        x=a;
(13)        y=b;
(14)    }
(15)
(16)    public float getX(){
(17)        return x;
(18)    }
(19)
(20)    public void setX(float a){
(21)        x=a;
(22)    }
(23)
(24)    public float getY(){
(25)        return y;
(26)    }
(27)
(28)    public void setY(float a){
(30)        y=a;
```

```
(31)    }
(32)
(33)    public String toSting(){
(34)        return "("+x+","+y+")";
(35)    }
(36)
(37)    public void reverse(){
(38)        float t;
(39)        t=x;
(40)        x=y;
(41)        y=t;
(42)    }
(43)}
```

Файл `Segment.java` – объявление класса «Отрезок на оси X»

```
(01)package classdemo;
(02)public class Segment extends Pair{
(03)    Segment(){
(04)        super;
(05)    }
(06)
(07)    Segment(float a, float b){
(08)        if(a < b){
(09)            super.setX(a);
(10)            super.setY(b);
(11)        }else{
(12)            super.setX(b);
(13)            super.setY(a);
(14)        }
(15)    }
(16)
(17)    public float length(){
(18)        return super.getY()-super.getX();
(19)    }
(20)
(21)    public void shiftLeft(float a){
```

```
(22)         super.setX(super.getX()-a);
(23)         super.setY(super.getY()-b);
(24)     }
(25)
(26)     public void shiftRight(float a){
(27)         super.setX(super.getX()+a);
(28)         super.setY(super.getY()+b);
(29)     }
(30)
(31)     public void reverse(){
(32)     }
(33)
(34)     public void stretch(int a){
(35)         float b = this.lenght():
(36)         if(b==0) return;
(37)         if(a>0){
(38)             b=b/2;
(39)             super.setX(super.getX()-b);
(40)             super.setY(super.getY()+b);
(41)         }else{
(42)             b=b/4;
(43)             super.setX(super.getX()+b);
(44)             super.setY(super.getY()-b);
(45)         }
(46)     }
(47)}
```

Файл `Point.java` – объявление класса «Точка на плоскости»

```
(01)package classdemo;
(02)public class Point extends Pair{
(03)     Point(){
(04)         super():
(05)     }
(06)
(07)     Point(float a, float b){
(08)         super(a, b);
```

```
(09)    }
(10)
(11)    public float distance(){
(12)        return (float)Math.sqrt(super.getX()*super.getX()+
(13)                super.getY()*super.getY());
(14)    }
(15)
(16)    public float distance(Point p){
(17)        return (float)Math.sqrt(
(18)               (super.getX()-p.getX())*(super.getX()-p.getX())+
(19)               (super.getY()-p.getY())*(super.getY()-p.getY()));
(20)    }
(21)
(22)    public void shift(float x, float y){
(23)        this.setX(this.getX()+x);
(24)        this.setY(this.getY()+y);
(25)    }
(26)}
```

Файл `LineSegment.java` – объявление класса «Отрезок прямой на плоскости»

```
(01)package classdemo;
(02)public class LineSegment{
(03)   private Point a;
(04)   private Point b;
(05)
(06)   LineSegment(){
(07)      a=new Point();
(08)      b=new Point();
(09)   }
(10)
(11)   LineSegment(Point first, Point second){
(12)      a=first;
(13)      b=second;
(14)   }
(15)
(16)   LineSegment(float x1, float y1, float x2, float y2){
```

```
(17)       a=new Point(x1,y1);
(18)       b=new Point(x2,y2);
(19)    }
(20)
(21)    public Point getFirstPoint(){
(22)       return a;
(23)    }
(24)
(25)    public void setFirstPoint(Point x){
(26)       a=x;
(27)    }
(28)
(29)    public Point getSecondPoint(){
(30)       return b;
(31)    }
(32)
(33)    public void setSecondPoint(Point x){
(34)       b=x;
(35)    }
(36)
(37)    public float length(){
(38)       return a.distance(b);
(39)    }
(40)
(41)    public void rotate(float phi, Point p){
(42)       phi=(float)(phi*Math.PI/180);
(43)       float x1=a.getX(), y1=a.getY(), x2=b.getX(), y2=b.getY();
(44)       float x0=p.getX(), y0=p.getY();
(45)       float x=(float)((x1-x0)*Math.cos(phi)-
(46)                (y1-y0)*Math.sin(phi)+x0);
(47)       float y=(float)((x1-x0)*Math.sin(phi)+
(48)                (y1-y0)*Math.sin(phi)+x0);
(49)       a.setX(x);
(50)       a.setY(y);
(51)       x=(float)((x2-x0)*Math.cos(phi)-(y2-y0)*Math.sin(phi)+x0);
(52)       y=(float)((x2-x0)*Math.sin(phi)+(y2-y0)*Math.sin(phi)+x0);
```

```
(53)       b.setX(x);
(54)       b.setY(y);
(55)    }
(56)
(57)    public Point center(){
(58)       Point c=new Point();
(59)       float x1=a.getX(), y1=a.getY(), x2=b.getX(), y2=b.getY();
(60)       float x0, y0, min, max;
(61)       int s;
(62)
(63)    }
(64)}
```

Файл `ClassDemo.java` – демонстрация использования классов

```
(001)package extendedclassdemo;
(002)
(003)public class ExtendedClassDemo {
(004)
(005)  public static void main(String[] args) {
(006)    Pair first = new Pair(), second;
(007)    System.out.println(first.toString());
(008)    second = new Pair(-3, 5);
(009)    System.out.println(second.toString());
(010)    second.reverse();
(011)    System.out.println(second.toString());
(012)    Segment seg1, seg2, seg3;
(013)    seg1=new Segment();
(014)    seg2=new Segment(-2,2);
(015)    seg3=new Segment(3,0);
(016)    System.out.println("отрезок 1: "+seg1.toString());
(017)    System.out.println("отрезок 2: "+seg2.toString());
(018)    System.out.println("отрезок 3: "+seg3.toString());
(019)    seg1.reverse();
(020)    seg2.reverse();
(021)    seg3.reverse();
(022)    System.out.println("отрезок 1 после обращения: "
```

```
(023)                       +seg1.toString());
(024)    System.out.println("отрезок 2 после обращения: "
(025)                       +seg2.toString());
(026)    System.out.println("длина отрезка 2: "+seg2.length());
(027)    System.out.println("отрезок 3 после обращения: "
(028)                       +seg3.toString());
(029)    System.out.println("длина отрезка 3: "+seg3.length());
(030)    seg1.stretch(1);
(031)    seg2.stretch(1);
(032)    seg3.stretch(-1);
(033)    System.out.println("отрезок 1 после растяжения: "
(034)                       +seg1.toString());
(035)    System.out.println("отрезок 2 после растяжения: "
(036)                       +seg2.toString());
(037)    System.out.println("длина отрезка 2: "+seg2.length());
(038)    System.out.println("отрезок 3 после растяжения: "
(039)                       +seg3.toString());
(040)    System.out.println("длина отрезка 3: "+seg3.length());
(041)    seg1.shiftLeft(2);
(042)    seg2.shiftRight(3);
(043)    seg3.shiftLeft(4);
(044)    System.out.println("отрезок 1 после сдвига влево на 2: "
(045)                       +seg1.toString());
(046)    System.out.println("отрезок 2 после сдвига вправо на 3: "
(047)                       +seg2.toString());
(048)    System.out.println("отрезок 3 после сдвига влево на 4: "
(049)                       +seg3.toString());
(050)    Pair a, b;
(051)    a=new Pair(1,2);
(052)    b=new Segment(1,2);
(053)    System.out.println("a= "+a.toString());
(054)    System.out.println("b= "+b.toString());
(055)    a.reverse();
(056)    b.reverse();
(057)    System.out.println("После обращения");
(058)    System.out.println("a= "+a.toString());
```

```
(059)    System.out.println("b= "+b.toString());
(060)    Point p1,p2;
(061)    p1=new Point(4,2);
(062)    p2=new Point(-2,2);
(063)    System.out.println("точка 1: "+p1.toString()+
(064)             "\nрасстояние от начала координат: "+p1.distance());
(065)    System.out.println("точка 2: "+p2.toString()+
(066)                "\nрасстояние от начала координат: "
(067)                +p2.distance());
(068)    System.out.println("расстояние от точки 1 до точки 2: "+
(069)                p1.distance(p2));
(070)    p1.shift(p2.getX(), p2.getY());
(071)    System.out.println("точка 1 после сдвига по X на: "
(072)                +p2.getX()+" по Y на: "+p2.getY()+" : "
(073)                +p1.toString());
(074)
(075)    LineSegment w=new LineSegment(1,1,3,3);
(076)    System.out.println("отрезок: первая точка: "+
(077)            w.getBeginPoint().toString()+" вторая точка: "+
(078)            w.getEndPoint().toString());
(079)    System.out.println("длина отрезка: "+w.length());
(080)    w.rotateEnd(90);
(081)    System.out.println("после поворота на 90 гр. "+
(082)                "относительно 1 точки: первая точка: "+
(083)                w.getBeginPoint().toString()+" вторая точка: "+
(084)                w.getEndPoint().toString());
(085)    System.out.println("длина отрезка: "+w.length());
(086)    w.rotate(90,new Point(0,0));
(087)    System.out.println("после поворота на 90 гр. "+
(088)                "относительно начала координат:\nпервая точка: "+
(089)                w.getBeginPoint().toString()+" вторая точка: "+
(090)                w.getEndPoint().toString());
(091)    System.out.println("длина отрезка: "+w.length());
(092)    Point c=w.center();
(093)    System.out.println("центральная точка: "+c.toString());
(094)    w.rotate(45,c);
```

```
(095)    System.out.println("после поворота на 45 гр. "+
(096)                  "относительно центра отрезка:\nпервая точка: "+
(097)                  w.getBeginPoint().toString()+" вторая точка: "+
(098)                  w.getEndPoint().toString());
(099)    System.out.println("длина отрезка: "+w.length());
(100)  }
(101)}
```

Результаты работы программы:

```
run:
(01)(0.0,0.0)
(02)(-3.0,5.0)
(03)(5.0,-3.0)
(04)отрезок 1: (0.0,0.0)
(05)отрезок 2: (-2.0,2.0)
(06)отрезок 3: (0.0,3.0)
(07)отрезок 1 после обращения: (0.0,0.0)
(08)отрезок 2 после обращения: (-2.0,2.0)
(09)длина отрезка 2: 4.0
(10)отрезок 3 после обращения: (0.0,3.0)
(11)длина отрезка 3: 3.0
(12)отрезок 1 после растяжения: (0.0,0.0)
(13)отрезок 2 после растяжения: (-4.0,4.0)
(14)длина отрезка 2: 8.0
(15)отрезок 3 после растяжения: (0.75,2.25)
(16)длина отрезка 3: 1.5
(17)отрезок 1 после сдвига влево на 2: (-2.0,-2.0)
(18)отрезок 2 после сдвига вправо на 3: (-1.0,7.0)
(19)отрезок 3 после сдвига влево на 4: (-3.25,-1.75)
(20)a= (1.0,2.0)
(21)b= (1.0,2.0)
(22)После обращения
(23)a= (2.0,1.0)
(24)b= (1.0,2.0)
(25)точка 1: (4.0,2.0)
(26)расстояние от начала координат: 4.472136
```

```
(27)точка 2: (-2.0,2.0)
(28)расстояние от начала координат: 2.828427
(29)расстояние от точки 1 до точки 2: 6.0
(30)точка 1 после сдвига по X на: -2.0 по Y на: 2.0 : (2.0,4.0)
(31)отрезок: первая точка: (1.0,1.0) вторая точка: (3.0,3.0)
(32)длина отрезка: 2.828427
(33)после поворота на 90 гр. относительно 1 точки:
(34)первая точка: (1.0,1.0) вторая точка: (-1.0000001,3.0)
(35)длина отрезка: 2.828427
(36)после поворота на 90 гр. относительно начала координат:
(37)первая точка: (-1.0,0.99999994)
(38)вторая точка: (-3.0,-1.0000002)
(39)длина отрезка: 2.8284273
(40)центральная точка: (-2.0,-2.9802322E-7)
(41)после поворота на 45 гр. относительно центра отрезка:
(42)первая точка: (-2.0000002,1.4142134)
(43)вторая точка: (-2.0,-1.4142138)
(44)длина отрезка: 2.8284273
СБОРКА УСПЕШНО ЗАВЕРШЕНА (общее время: 1 секунда)
```