Задачи за масиви и методи

Велико Дончев

Задачи за линейно търсене и обхождане в масив

Задача 1. Да се напише програма на Java , която чете масив с n (n < 20) от цели числа в интервала [10 ; 25] и извежда най- голямото число от тях , както и индексът му.

Забележка: Въвеждането на n от клавиатурата да става докато входът не стане коректен. Освен това , всички въведени n числа трябва да са от 10 до 25. Ако потребителя въведе некоректно число на екрана да излиза подходящо съобщение и да се въведе ново число.

Вход: с << означаваме изход , с >> вход

<< Въведете n от 1 до 20 !:
>> 30
<< Некоректен вход!Въведете отново!
>> 3
<< Въведете 3 цели числа в интервала [10;25] !
>> 11
>> 12
>>9
<< 9 е некоректно число! Въведете ново!
>> 15
Изход :
<< 15, 2

Задача 2. Да се напише метод на Java, който при подаден масив от реални числа връща масив от положителните елементи на първия.
Спецификация: public double[] returnPOSITIVE(double[] realArray) {}

Задача 3. Хотел
Галактическия пътешественик Йон Тихи веднъж посетил един хотел с много, много стаи, които били номерирани с естествените числа 1,2,3... . Първоначално вратите на стаите били затворени. Йон Тихи преминал покрай всичките стаи, като на отворил вратите им. След това преминал втори път , но само покрай стаите с номера 2,4,6... и т.н. , като затварял вратите им. Преминал трети път, но само покрай стаите с номера 3,6,9... и т.н. като отварял вратите , които били затворени, и затварял вратите, които били отворени. Йон Тихи продължил своето странно занимание , започвайки всеки път от врата с все по- голям номер, като при к-тото си преминаване се спирал само при всяка к-та стая и ако вратата и била отворена, той я затварял, и обратното , ако била затворена – той я отварял. Напишете програма , която въвежда от клавиатурата едно цяло положително число n и отпечатва на екрана номерата на отворените стаи след

К-тото преминаване на Йон.
Вход: n= 7 , k=3
Изход на програмата:

	1	2	3	4	5	6	7
	затворена	затворена	затворена	затворена	затворена	затворена	затворена
К=1	отворена	отворена	отворена	отворена	отворена	отворена	отворена
К=2	отворена	затворена	отворена	затворена	отворена	затворена	отворена
К=3	отворена	затворена	затворена	затворена	отворена	отворена	отворена

Изход:
1 5 6 7

Задача 4. Улица (за разглеждане вкъщи или решаване в час без да се поглежда решението)

Оъ едната страна на улица „Небостъргачева“ има N сгради с плоски покриви (3<=N<=200) всяка с височина цяло число – между 3 и 100 метра. Сградите на улицата започват с номер 0 до номер N-1. Съседните , еднакво високи сгради образуват чрез покривите си площадки , подходящи за летища за хеликоптери. Помогнете на инженерите от улица „Небостъргачева“ да съобразят колко подходящи площадки има за да се построят летища. За подходща се счита площадка с повече от 3 (или 3) покрива.

Да се напише програма на Java, която въвежда от клавиатурата n и извежда в подходящ вид всички подходящи площадки, заедно с началото и краят им.

Вход:
N= 20
0 1 2 3 4 5 6 7 8 9 10 11 12 13 14 15 16 17 18 19
3 4 4 4 4 6 5 5 8 8 8 9 7 7 13 13 13 13 13 13
Изход:
От 1 до 4 с височина 4 метра
От 8 до 10 с височина 8 метра
От 14 до 19 с височина 13 метра
3 площадки

Решение на задачата:

Височините на сградите ще пазим в масив. Ще го обходим линейно , като същевременно ще извеждаме площадка ако е „подходяща“. Така няма да се наложи да пазим намерените площадки в отделен масив.

```
import java.util.Scanner;
public class Ulica {
   public static int[] Heights;
   public static void readHeights(int n)
         {   Heights=new int[n]; // инициализиране на глобалния, статичен масив Heights
            Scanner sc=new Scanner(System.in);
            for(int i=0;i<Heights.length ;i++)
                        {
                  System.out.println("Сграда с номер "+i+"е висока : ");
```

```
        do // изискване за коректен вход
                Heights[i]=     sc.nextInt();
                        while(Heights[i]<3 || Heights[i]>100);
                        }
                }
   public static void main(String []args)
   {
          System.out.println("Въведете брой на сгради на улицата между 3 и 200 :");
           Scanner sc=new Scanner(System.in);
           int broiSgradi;
         do // изискване за коректен вход
   broiSgradi=sc.nextInt();
            while(broiSgradi<3 || broiSgradi >200);
          System.out.println("Въвдете височината на всяка сграда");
           readHeights(broiSgradi); // извикване на метода readHeights , прочитане на
височините
           int br1=0; // брой на сградите с еднаква височина в дадена площака
           int height=0;// височина на площадка
           int br2=0; // начало на площадка
           int broi=0; // брой площадки
         for(int i=1;i<Heights.length;i++)
         {
                if(Heights[i]==Heights[i-1]) // намерили сме две съседни сгради с еднаква
височина
                {   // тук има площадка!
                        // да проверим дали има дължина, по - голяма от 3 ?
                height=Heights[i-1]; //височината и е колкото първата сграда
                        br2=i-1; // започва от i-1
                        br1=2; // за сега, дължината и е 2
                        i++; /* мини на следващата сграда
                    и започни да проверядаш дали има още еднакво високи сгради
                            */
                        while(i<Heights.length &&  Heights[i]==Heights[i-1])
                                // докато съседните сгради са с еднаква височина
                        {
                        br1++; //увеличавай дължината на площадката
                        i++; // мести и i
                        }
                        if(br1>=3) // ако намерената площадка е "подходяща"
                                {
                                broi++;
System.out.println("От " + br2 +" до " + (br2+br1-1)+ " с височина " +height + " и дължина
" + br1 );
                    }
                        i--; // нагласяне на индекса :) :) :)
         }}
         System.out.println("Площадки: " +broi);
}
}
```

Методи за сортиране на масив

Метода на "мехурчето"
Сортирането по метода на "мехурчето" се извършва по следния начин:

- обхождането на елементите става от 1 към N, като всеки два съседни елемента се сравняват;
- Ако долният е по-голям от горния, то те сменят местата си;
- Този процес продължава до пълното сортиране на масива.
- След завършване на к-тото обхождане, k- тия по големина елемент е на мястото си. Следващото обхождане стига до N-k-я елемент

Сложност:
В най-добрия случай : O(n)
В най-лошия случай : O(n^2)
Средна сложност: O(n^2)

Пример:

Ако несортираният масив има вид: 5 1 4 2 8 и искаме да получим елементите, сортирани във възходящ ред:

- Вземат се първите два елемента от масива: 5 и 1
- Тъй като 1 < 5 се разменят местата им. Текущ масив: 1 5 4 2 8
- Вземат се следващите два елемента от масива: 5 и 4
- Тъй като 4 < 5 се разменят местата им. Текущ масив: 1 4 5 2 8
- Вземат се следващите два елемента от масива: 5 и 2
- Тъй като 2 < 5 се разменят местата им. Текущ масив: 1 4 2 5 8
- Вземат се следващите два елемента от масива: 5 и 8
- Тъй като 5 < 8 не се извършва размяна. Тук спира първото обхождане на масива.
- Масивът след първото обхождане има вида: 1 4 2 5 8. Следващото обхождане ще бъде до четвъртият елемент, 5. Аналогично се извършват по-нататъшни обхождания, докато се стигне до сортирания масив 1 2 4 5 8.

ПРОГРАМА НА JAVA, реализираща метода :

```
public class Bubblesort {


        public static  void main(String[] args)
                int[] array = new int[]{6,5,4,3,5,1,42,-1};
                int element;
                int size=array.length;

                for (int i=0;i<size;i++)
                        for (int j=1; j<size-i;j++)
                        {
                                if (array[j-1]>array[j])
                                {
                                element = array[j-1];
                                array[j-1]=array[j];
```

```
                    array[j]=element;
                    }
                }
            for (int i=0;i<size;i++)
            {
                System.out.print(" "+array[i]);
            }
        }
    }
```

Пряка селекция

Сортирането чрез пряка селекция се извършва по следния начин:

- намира се минималния елемент след което той и първият разменят местата си;
- намира се минималният елемент измежду всички елементи с изключение на първия и след което той и вторият разменят местата;
- този процес продължава до пълното сортиране на масива.

Пример:

31 34 12 22 11	Най-малкият елемент е 11. Разменят се местата на 11 и 31.
11 34 12 22 31	Най-малкият елемент от останалия списък е 12. Разменят се местата на 12 и 34.
11 12 34 22 31	Вече имаме една част от списъка, която е сортирана. Разменят се местата на 22 и 34.
11 12 22 34 31	Разменят се местата на 31 и 34.
11 12 22 31 34	Списъкът е сортиран

Сложност:
В най-добрия случай : O(n)
В най-лошия случай : O(n^2)
Средна сложност: O(n^2)

ПРОГРАМА НА JAVA, реализираща метода :

```
public class Insertion{
        public static void main(String [] args)
        {
            int[] SomeArray= new int[]{6,5,4,3,5,1,42,-1};
        int x,n,k;
            n=SomeArray.length;
              for(int i=0;i < n-1;i++) {
                k=i;
                x=SomeArray[i];
                  for(int j=i+1;j < n;j++) {
```

```
                    if(SomeArray[j] < x) {
                        k=j;
                    x=SomeArray[j];
                }
                    }
                    SomeArray[k]=SomeArray[i];
                    SomeArray[i]=x;
                }
                for (int i=0;i<SomeArray.length;i++)
                        {
                    System.out.print(" "+SomeArray[i]);
                        }
            }
}
```

Многомерни Масиви

Задача 1.

Да се напише програма, която въвежда елементите на правоъгълна матрица $\mathbf{a}_{nxm}$ и намира и извежда матрицата, получена от дадената като всеки от нейните елементи е увеличен с 1.

Задача 2.

Да се напише програма, която намира и извежда сумата от елементите на всеки стълб на квадратната матрица $\mathbf{a}_{nxn}$.

Задача 3.

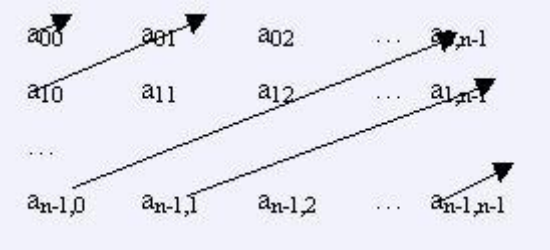
Да се напише програмен фрагмент, който намира номерата на редовете на целочислената квадратна матрица $\mathbf{a}_{nxn}$, в които има елемент, равен на цялото число x и проверява дали матрицата е симетрична

А)относно главния си диагонал.

Б)относно второстепенния си диагонал.

Задача 4.

Да се напише програмен фрагмент, който обхожда кватратната матрица $\mathbf{a}_{nxn}$ по диагонали, започвайки от елемента $\mathbf{a}_{0,0}$, както е показано по-долу:



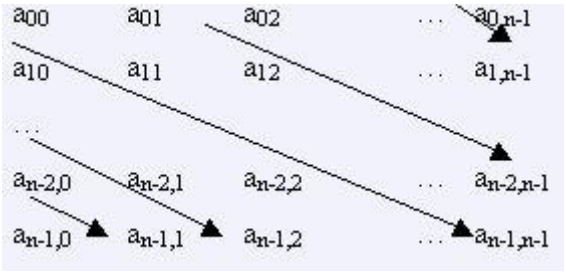
РЕШЕНИЕ:

```
import java.util.Scanner;
public class matrici {

	public static void readMATR(int [][]A)
	{
		Scanner sc=new Scanner(System.in);
		for(int i=0;i<A.length;i++)
			for(int j=0;j<A[0].length;j++)
			A[i][j]=	sc.nextInt();

	}
	public static void writeMATR(int [][]A)
	{

		for(int i=0;i<A.length;i++)
			{for(int j=0;j<A[0].length;j++)
			System.out.print(A[i][j]+" ");
			System.out.println();
			}

	}

	public static void main(String [] args)
{
	Scanner sc=new Scanner(System.in);
	int n=sc.nextInt();

	int[][]A=new int[n][n];
	readMATR(A);
	writeMATR(A);
	//Над второстепенния диагонал
   for(int i=0;i<=n-1;i++)
	{	for(int j=0;j<=i;j++)
			System.out.print(A[i-j][j]+ " ");

	System.out.println();
	}
    //Под второстепенния диагонал

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

```
                    System.out.print(A[j][n-(j-i)-1]+" ");

            System.out.println();
            }
}
}
```

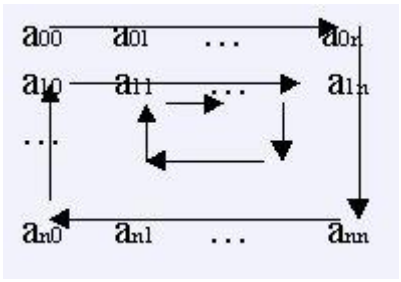
Задача 5. (ДОМАШНО – по аналогия с предната)

Да се напише програмен фрагмент, който обхожда квадратната матрица $\mathbf{a}_{nxn}$ по диагонали, започвайки от елемента $\mathbf{a}_{n-1,0}$, както е показано по-долу:



Задача 6.

Да се напише програмен фрагмент, който обхожда квадратната матрица $\mathbf{a}_{nxn}$ по спирала, започвайки от елемента $\mathbf{a}_{0,0}$, както е показано по-долу:



Вход:

1 2 3 4
5 6 7 8
9 8 7 6
5 4 3 2

Изход:

1 2 3 4 8 6 2 3 4 5 9 5 6 7 7 8

Решение:

Забележка по решението:

Обходените елементи се нулират. Заради това решението е коректно само ако няма въведена 0 от потребителя. Проблема може да бъде лесно решен чрез използване на алгернативна булева матрица ,първоначално инициализирана с false за всеки елемент, чиито елемент (i,j) става true когато е обходен (i,j) тия елемент на нашата матрица.Коригирайте кода.

```
public static void main(String [] args)
        {Scanner sc=new Scanner(System.in);
                int n=sc.nextInt();
                int[][]A=new int[n][n];
                readMATR(A);
                writeMATR(A);
                int dir=0;
                int done=0;
                int x=0,y=0;

                do
                {

        System.out.println(A[x][y]+ " ");
                                A[x][y]=0;
                                switch(dir)
                                {
                                case 0: if(y+1>=n || A[x][y+1]==0) dir=1;break;
                            case 1: if(x+1>=n || A[x+1][y]==0) dir=2;break;
                                case 2: if(y-1<0  || A[x][y-1]==0) dir=3;break;
                                case 3: if(x-1<0  || A[x-1][y]==0) dir=0;break;
                                }
                                switch(dir)
                                {
                                case 0: y++;break;
                                case 1: x++;break;
                                case 2: y--;break;
                                case 3: x--;break;
                                }
                                done++;
                                        }
          while(done!=n*n);
        }}
```

Задача 8.

Да се напише програма, която:

а) въвежда по редове елементите на квадратната реална матрица A с размерност n x n;

б) от матрицата A конструира редицата B: b_0, b_2,..., b_{m-1}, където m = n.n, при което първите n елемента на B съвпадат с елементите на първия стълб на A, вторите n елемента на B съвпадат с елементите на втория стълб на A и т.н., последните n елемента на B съвпадат с елементите на последния стълб на A;

в) сортира във възходящ ред елементите на редицата B;

г) образува нова квадратна матрица A с размерност n x n, като елементите от първия ред на A съвпадат с първите n елемента на B, елементите от втория ред на A съвпадат с вторите n елемента на B и т.н. елементите от n - тия ред на A съвпадат с последните n елемента на B;

д) извежда по редове новата матрица A.