

**Домашни работи по Увод в
алгоритмите и програмирането
на**

**Сте́ла Стойчева Га́девска F91114
Пролетен семестър 2019**

Протокол No: 6335

Домашно 1. 2019

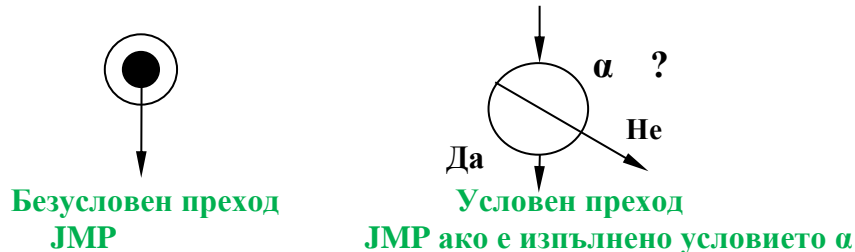
1. Препишете определението за алгоритъм от лекцията.

Алгоритъмът е такова крайно, дискретно (постъпково), детерминирано преобразование, което, приложено над произволен допустим набор от стойности на входното множество, довежда до получаването на единствен набор от допустими стойности на изходното множество.

При това полученото на изхода представлява правилен “отговор” на решаваната задача.

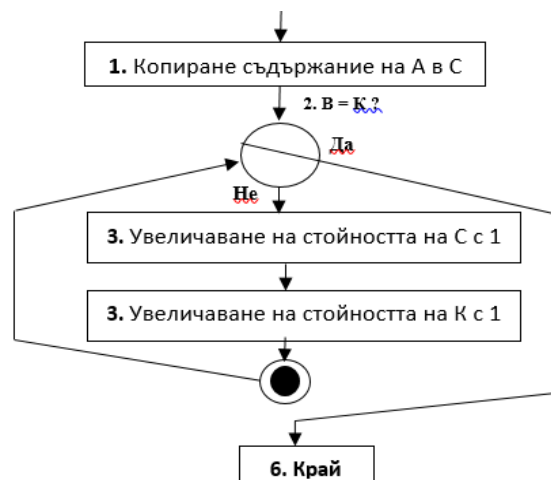
Опитайте се да го разберете добре. Ще обсъждаме всяко свойство на лекциите.

2. Съставете схеми на Базовите Елементи на Управление с техните наименования.



3. Съставете **схемата на управление** за алгоритъма за RAM машина от лекцията (събиране на две цели положителни числа) и съответния програмен текста на прото-асемблер. Обърнете внимание на това как решаването на задачата става чрез въвеждане на допълнителни променливи в паметта за данните, в случая - брояч. Обърнете внимание на двата базови елемента на управление - безусловен и условен преход и как те се използват, за да се програмира повторение (цикъл).

1. Сру СА
2. Jmp BK6
3. Inc C
4. Inc K
5. Jmp 2
6. Continue



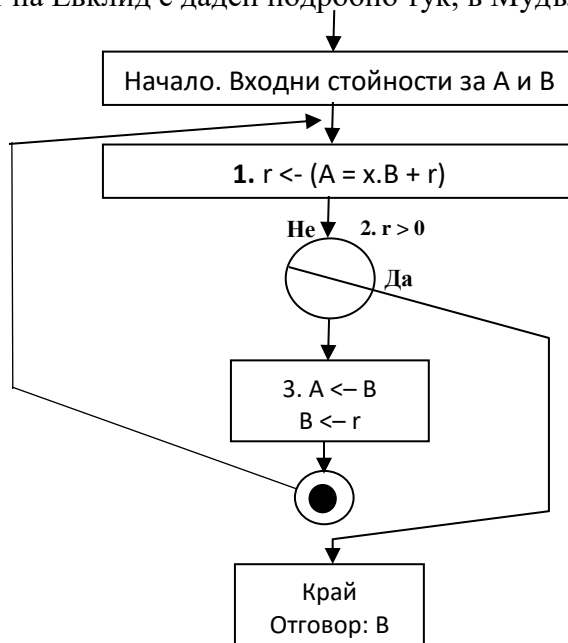
4. Съставете **схема на управление** за алгоритъма на Евклид, така, както той е описан в оригинала (Евклид). Това означава, че трябва да ползвате условни и безусловни преходи, за да организирате управлението. За тези от вас, които не са имали лекции през първия семестър по тази тема, алгоритъмът на Евклид е даден подробно тук, в Мудъл.

1. Намиране на остатъка r от целочисленото деление на A с B

2. Проверка на остатъка:
- няма \rightarrow отг. B
- има \rightarrow стъпка 3

3. Смяна на стойностите (делителят става делимо, а намер. остатък – делител)

4. Връщане към стъпка 1

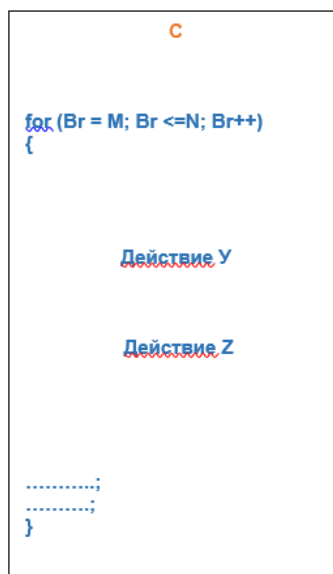
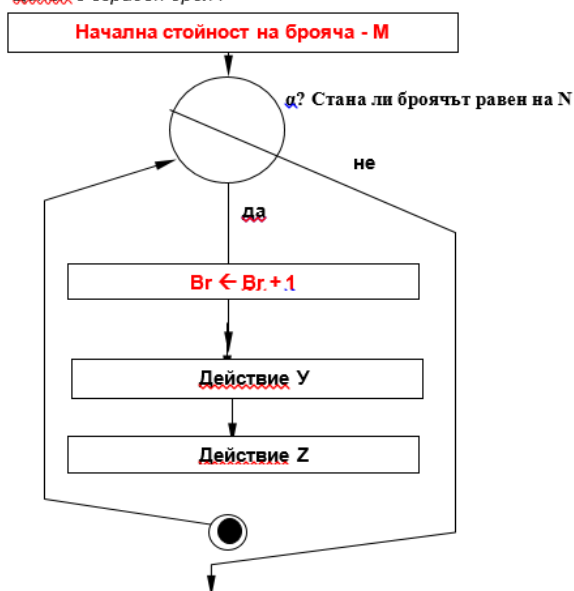


ДОМАШНОТО СЕ ПРЕДСТАВЯ ЗА ЗАЩИТА НА КРАЯ НА СЕМЕСТЪРА С ТОЧКИТЕ НА ЗАДАНИЕТО.

Домашно 2, 2019.

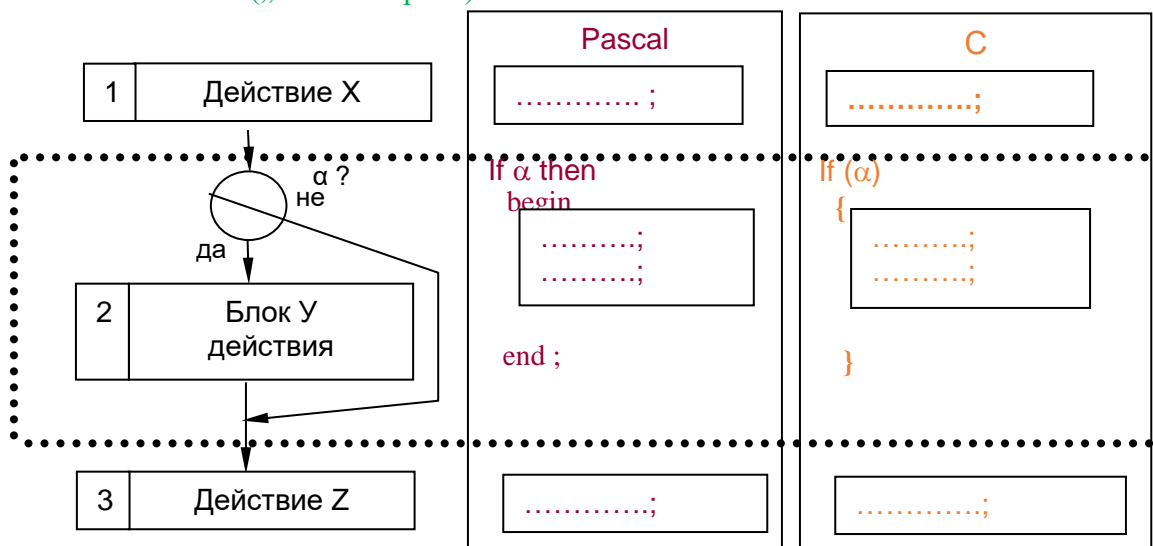
1. Като се базирате на основните елементи на управление, дадени в презентациите към тази тема, съставете петте схеми на основните Конструкции на Управление - Клоните (*двуклон* и *байпас*) и на Повторенията (цикли с предусловие, със следусловие и по брояч). Може на ръка, може да копирате схемите, както ви е по-удобно. Тези конструкции са изучавани в уводните курсове по програмиране, сега задачата е да видите мястото на условните и безусловни преходи като елементи на управление. Прегледайте анимациите към тази тема, за да си припомните изученото по програмиране. Успоредно на схемите, маркирайте програмни оператори на познат език, както е показано в презентациите и тук:

Цикъл с вграден брояч

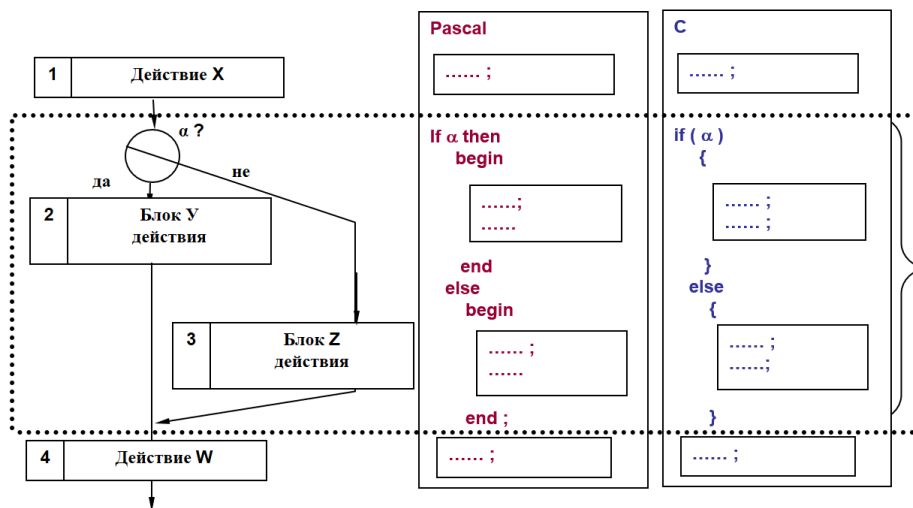


1. Клонове (Двуклон и байпас)

БАЙПАС („мини покрай“)



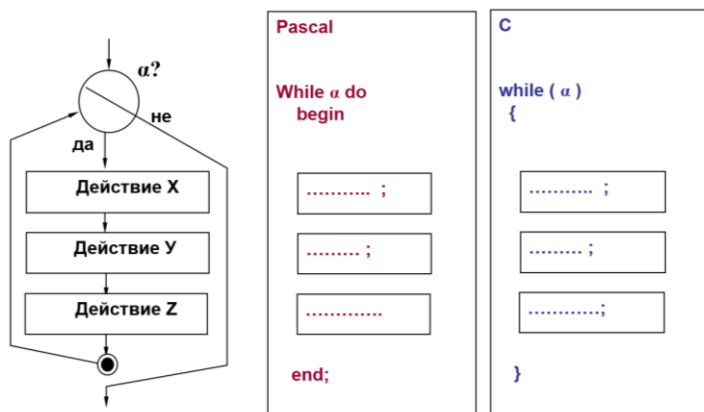
ДВУКЛОН



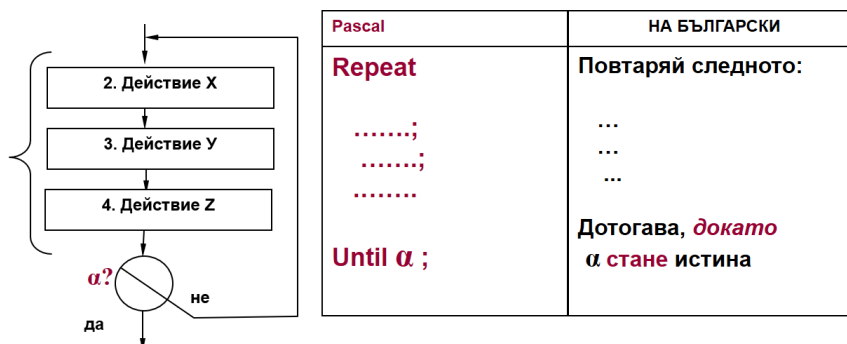
2. Повторения (Цикли с предусловие, със следусловие и по брояч)

* характерно – предоставена е възможност изпълнението на алгоритъмът да става в различни последователности в зависимост от стойностите, получени в средата.
Условен преход – основен.

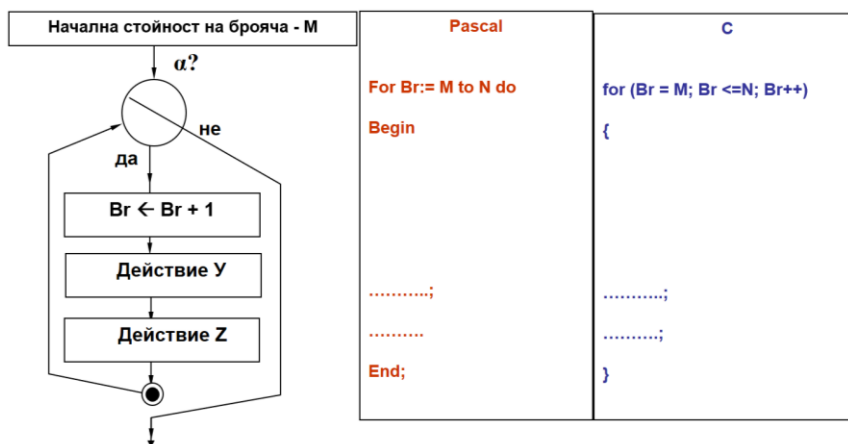
ЦИКЪЛ С ПРЕДУСЛОВИЕ



ЦИКЪЛ СЪС СЛЕДУСЛОВИЕ



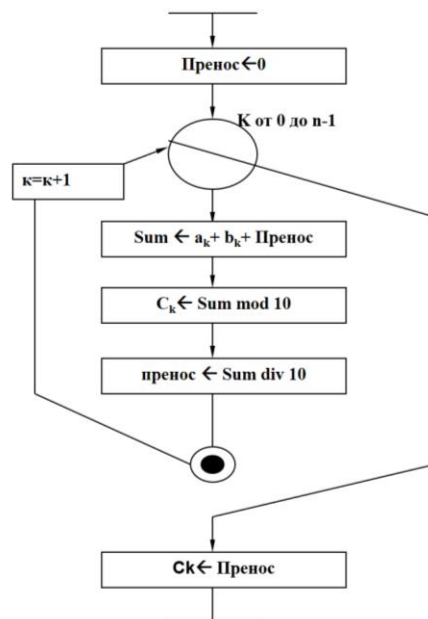
ЦИКЪЛ С ВГРАДЕН БРОЯ (нарастващ)



2. Съставете алгоритмичната схема за събиране на две числа (зададени с два вектора) в позиционна бройна система (използване на операциите с цели числа, по-точно - делене с остатък). Схемата е дадена в презентацията към тази тема, а алгоритъмът е подробно разработван в час. Встрани от схемата напишете програмен текст. Проверете дали програмата работи за събиране на числа в десетичен код.

Пример: Десетична бр. с-ма
("mod 10" и "div 10")

	0	2	3	5	a
	9	7	8	4	b
	10	10	11	9	Sum
	0	0	1	9	c
	1	1	1	0	прено c
$K=4$	3	2	1	0	



- Нулираме като начална стойност преносът.
- В началото, стойността на k е 0; изпълняваме стъпки от цикъла последователно, докато k достигне стойност $n-1$ (n е бр. размерността на елементите във векторите).

3. Сборът от двете числа (зададени с вектори) плюс хипотетичния пренос се приемат от Sum.
4. Остатъкът от делението на Sum с 10 е стойността, която приема съответстващият елемент по ред на сбора.
5. Преносът приема полученото при делението на Sum с 10.
6. Увеличаваме стойността на индекса k с 1.
 - > $k < n-1$ -> изпълняваме цикъла отново
 - > $k > n-1$ -> записваме стойността на преноса в поредния елемент на сбора

```
#include <iostream>
#include <vector>
using namespace std;

int main() {

    vector<int> v1;
    vector<int> v2;
    vector<int> v3;
    int prenos = 0;

    for(int i = v1.size() - 1; i >= 0; i--){
        int sum = v1[i] + v2[i] + prenos;
        int a = sum % 10;
        v3.push_back(a);
        prenos = sum / 10;
    }

    return 0;
}
```

```
import java.util.Vector;

public class VectorDemo {
    public static void main(String[] args) {
        Vector<Integer> v1 = new Vector<>();
        Vector<Integer> v2 = new Vector<>();
        Vector<Integer> v3 = new Vector<>();
        int prenos = 0;

        for (int i = v1.size()-1; i >= 0; i--) {
            int sum = v1.get(i) + v2.get(i) + prenos;
            int a = sum % 10;
            v3.add(a);
            prenos = sum / 10;
        }
    }
}
```

Ако сте любопитни, проверете дали това работи и за числа, кодирани при база шестдесет, както в първата известна система за кодиране – тази на Шумерите <https://en.wikipedia.org/wiki/Sexagesimal>.

Домашно 3, 2019.

Напишете формулата на Гаус за сумата на първите n естествени числа. Формулата е дадена в презентацията към тази тема. Под нея разпишете проверка - дали формулата е вярна за сумата на първите k естествени числа, където k е последната цифра от факултетния ви номер. Ако номерът ви завършва с нула, приемете $k=6$.

За първите n естествени числа.

$$a_1 + a_2 + a_3 + \dots + a_n = \sum_{i=1}^n a_i$$

Проверка:

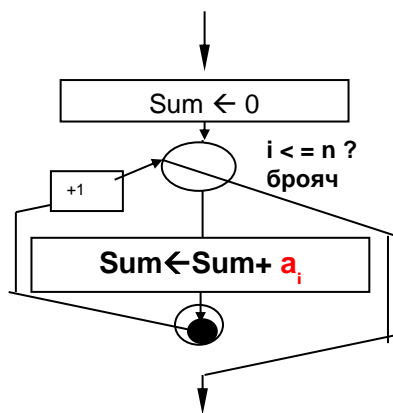
$$\sum_{i=1}^4 i = 1 + 2 + 3 + 4 = \frac{4(4+1)}{2} = \frac{4 \cdot 5}{2} = 10$$

1. Натрупване на суми и произведения. Множество на целите числа, цикли по брояч. Прегледайте материала от лекцията.

Задача. Съставете трите схеми на управление, успоредно на тях - програмния текст на познат за вас език и пуснете на машина една от програмите за натрупване на следните три суми:

$$\sum_{i=1}^n i^2$$

Модел:



$$\sum_{i=2}^n (i^2 - 2i)$$

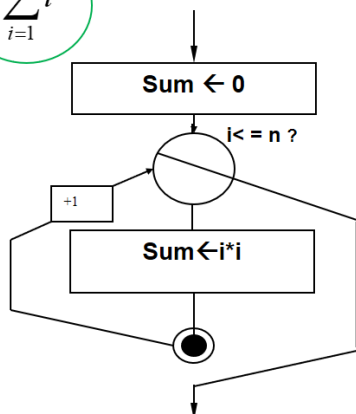
$$\sum_{i=3}^n (i^3 - 2i)$$

Няма обекти, това е просто програма от пет реда, която чака въвеждане на n от клавиатура (напишете оператора) и смята:

```
....
Sum = 0;
for (i = 1; i <= n; i++)
    Sum = Sum + ai;
```

След завършване на цикъла по брояч, разпечатва намерената сума на екран (напишете оператора).

$$\sum_{i=1}^n i^2$$

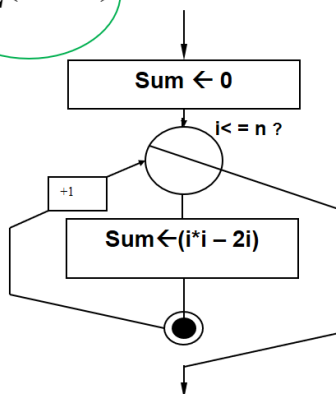


```

Sum = 0;

for (i = 1; i <= n; i++) {
  Sum = Sum + i * i;
}
  
```

$$\sum_{i=2}^n (i^2 - 2i)$$

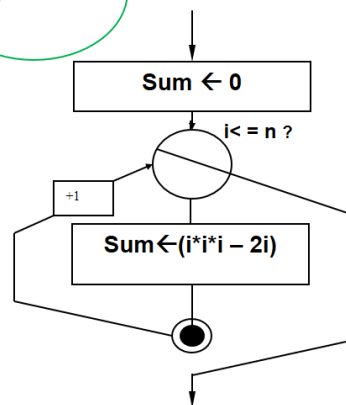


```

Sum = 0;

for (i = 2; i <= n; i++) {
  Sum = Sum + (i * i - 2i);
}
  
```

$$\sum_{i=3}^n (i^3 - 2i)$$



```

Sum = 0;

for (i = 1; i <= n; i++) {
  Sum = Sum + (i * i * i - 2i);
}
  
```

Напомниме – a_i се изразява с i . То е, например, $i * i$.

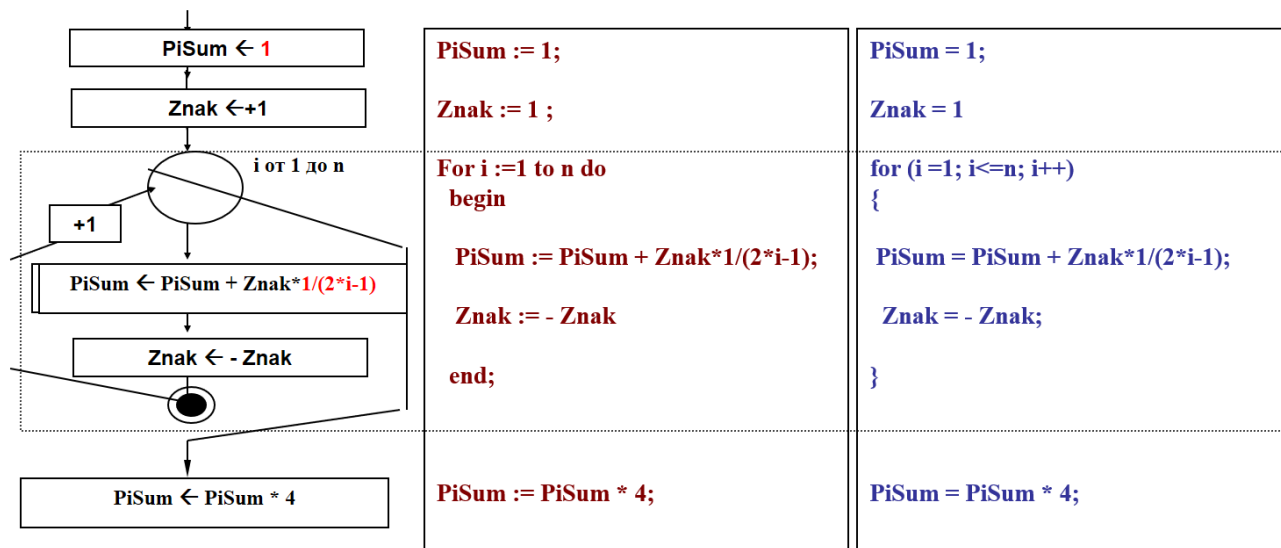
2. Понятие алтернативни редове. Прегледайте материала от лекцията и проследете линковете за функциите \sin и \cos . Ново за вас е само свързаното с представяне на функция в ред на Тейлор и Маклорен, т.е. пресмятане на стойността на функцията посредством нейните производни. Без да се съсредоточавате над формулите, сега запомнете само, че стойностите на тези (и много други) функции могат да се представят като Сума на Безкраен ред. Въпросите на безкрайността коментирахме като свързани с нещо, наричано Точност на Решението.

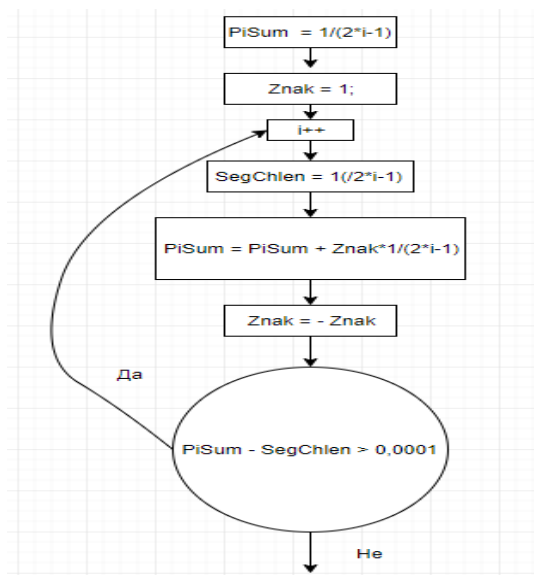
Задача. Съставете схемата на управление (има я в лекцията) и успоредно на нея – програмата за пресмятане на π с алтернативен ред. Съставете програмата по ДВА начина:

- Сумата се пресмята 10 000 пъти, по брояч.
- Сумата се пресмята ДОТОГАВА, ДОКАТО разликата между две стойности, получени последователно, се получи по-малка от 0.0001.

Обърнете внимание, че π е реална константа, за която няма точен израз като рационално число. Формулата, разписана в лекцията, следва от $\tan(\pi/4)$, което е 1. (а \tan е \sin/\cos , нали?).

$$\left(\frac{\pi}{4}\right) = \underset{i: 1}{\frac{1}{1}} - \underset{2}{\frac{1}{3}} + \underset{3}{\frac{1}{5}} - \underset{4}{\frac{1}{7}} + \underset{5}{\frac{1}{9}} - \underset{6}{\frac{1}{11}} + \underset{7}{\frac{1}{13}} - \dots = \sum_1^{\infty} (-1)^{i+1} \frac{1}{2i-1}$$





```
PiSum = 1/(2*i-1);
Znak = 1;
do{
    i++;
    SegChlen = 1/(2*i-1);
    PiSum += Znak * SegChlen;
    Znak = - Znak;
}while(PiSum - SegChlen > 0.0001)
```

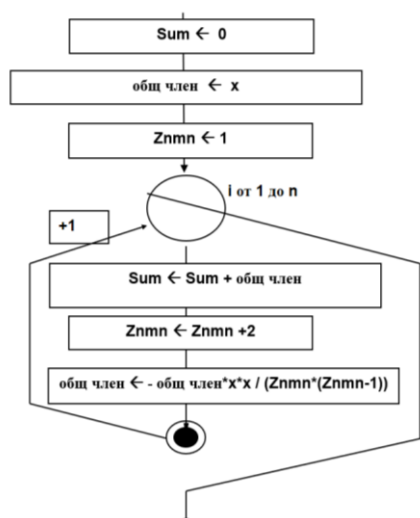
ПРИ ЗАЩИТА, ЗАДАНИЕТО СЕ ПРЕДАВА КАТО ПЪРВА СТРАНИЦА НА ДОМАШНОТО.

Домашно 4, 2019

Задача 1.

Да се състави алгоритъм (схемата на управление) и **програма** (в съответствие със схемата) за пресмятане **по два начина** на функцията $\sin(x)$ с *точност* 0.000001 , както е дадена в лекциите и на семинар (развита в ред на Маклорен). Първи начин – с пресмятане на факториел (в знаменателя) и втори начин – без пресмятане на факториел. Обърнете внимание, че аргументът е ъгъл в радиани. Прегледайте материала от семинара, схемите са дадени. Приложете към домашното код и снимки на екран с входа и изхода.

$$\sin(x) = +\frac{x}{1!} - \frac{x^3}{3!} + \frac{x^5}{5!} - \frac{x^7}{7!} + \dots = \sum_{i=1}^{\infty} (-1)^{i+1} \frac{x^{(2i-1)}}{(2i-1)!}$$



```
"C:\Program Files\Java\jdk-11.0.5\bin\javac
```

```
180
```

```
1,123308
```

```
Process finished with exit code 0
```

```
|
```

```
import java.util.Scanner;

public class IntegerNumbersSum {

    public static double factorial(double n) {
        double x = 1;
        for (int i = 1; i <= n; i++) {
            x *= i;
        }
        return x;
    }

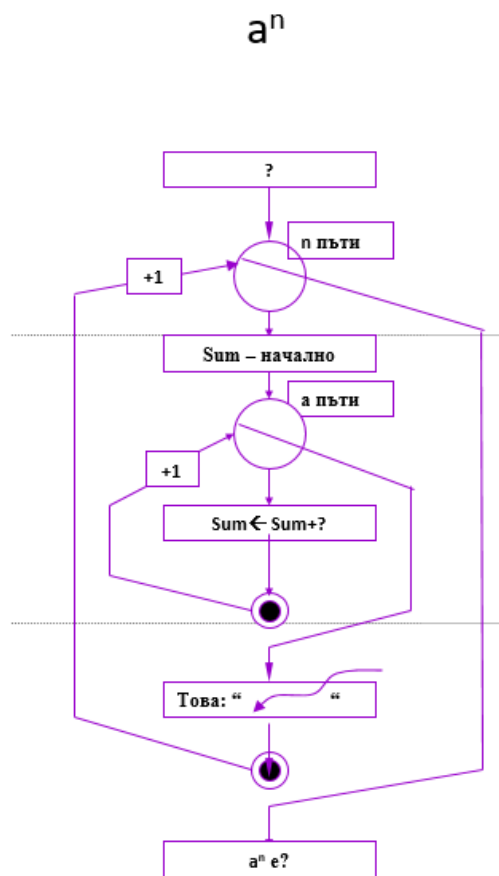
    public static void main(String[] args) {
        Scanner scanner = new Scanner(System.in);
        double pi = Math.PI;
        double x = Double.parseDouble(scanner.nextLine());
        x *= pi / 180;

        double sinX = x - Math.pow(x, 3) / factorial(n: 3)
            + Math.pow(x, 5) / factorial(n: 5)
            + Math.pow(x, 7) / factorial(n: 7);

        System.out.printf("%.6f", sinX);
    }
}
```

Задача 2.

Да се състави алгоритъм (схемата на управление) и програма (в съответствие със схемата) за повдигане на цяло положително число a на цяла положителна степен n , при използване само на адитивни операции. Прегледайте още веднъж материала от семинара. Там и схемите са дадени.



```
"C:\Program Files\Java\jdk-11.0.5\
```

```
2
```

```
3
```

```
8
```

```
Process finished with exit code 0
```

```
import java.util.Scanner;

public class Power {
    public static void main(String[] args) {
        Scanner scanner = new Scanner(System.in);
        int a = Integer.parseInt(scanner.nextLine());
        int n = Integer.parseInt(scanner.nextLine());

        int sum = a;
        int increment = a;

        for (int i = 1; i < n; i++) {
            for (int j = 1; j < a; j++) {
                sum += increment;
            }
            increment = sum;
        }

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

Заданието е неразделна първа страница на домашното.

Домашно 5, 2019

Задача 1. Табулиране на функция, решаване на уравнение чрез сканиране на интервал.

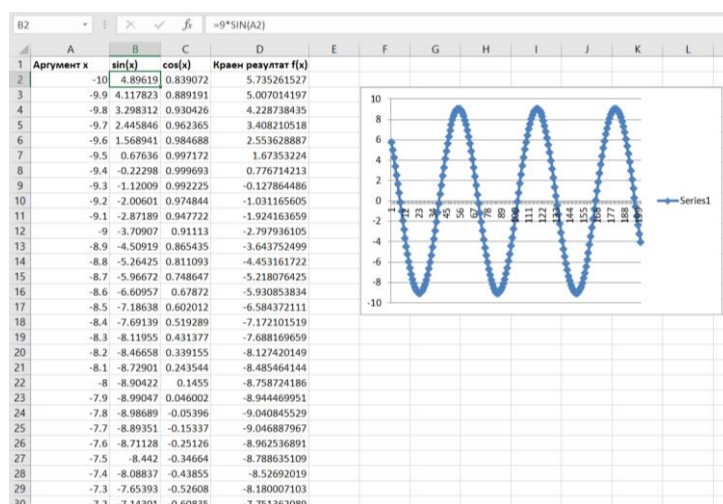
- В ексел, със стъпка за аргумента 0.1, табулирайте в интервала -10, 10

функцията:

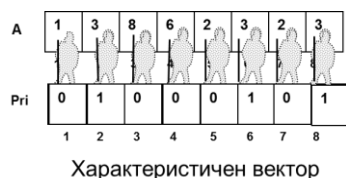
$$f(x) = \Phi 1. \sin(x) - \Phi 2. \cos(x),$$

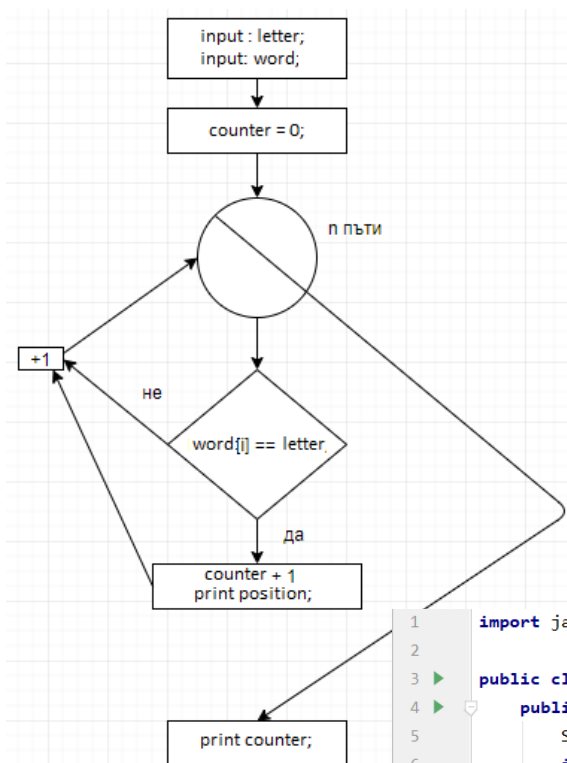
където $\Phi 1$ и $\Phi 2$ са първата и втората цифра на факултетния ви номер. Ако някое от тях е нула, заместете го с 5, а ако и двете са нула, заместете ги съответно с 5 и 6.

- Направете необходимото, за да се види къде $f(x)$ си сменя знака и отбележете тези места с коментар „корен на уравнението $f(x) = 0$ “.



Задача 2. Претърсване чрез обхождане. Съставете алгоритъм (схема на управление) и програма (успоредно на схемата) за претърсване на въведена от клавиатура дума за наличие на дадена буква. Програмата да „докладва“ колко пъти се среща търсената буква и на кои точно позиции.





```

1  import java.util.Scanner;
2
3  public class Word {
4      public static void main(String[] args) {
5          Scanner scanner = new Scanner(System.in);
6          int counter = 0;
7          char letter = scanner.nextLine().charAt(0);
8          String word = scanner.nextLine();
9
10         System.out.println("Positions: ");
11         for (int i = 0; i < word.length(); i++) {
12             if (word.charAt(i) == letter) {
13                 counter++;
14                 System.out.println("Searched letter (" + letter + ") is found in position: " + i);
15             }
16         }
17
18         System.out.println("Number of times found: " + counter);
19     }
20 }
  
```

a

Matematika

Positions:

Searched letter (a) is found in position: 1

Searched letter (a) is found in position: 5

Searched letter (a) is found in position: 9

Number of times found: 3

Process finished with exit code 0

Използвайте Характеристичен вектор.

Заданието е неразделна част от домашното

Домашно 6, 2019

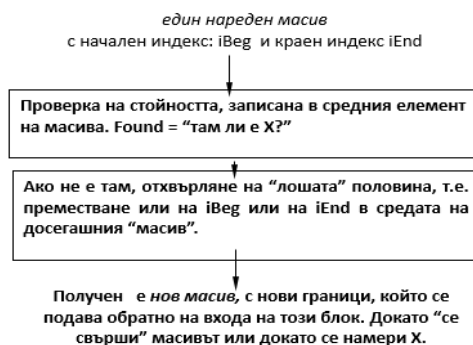
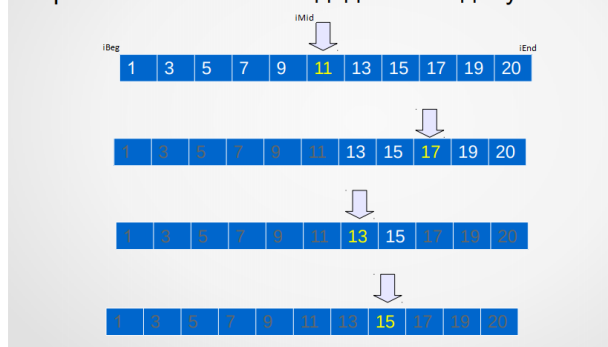
Задача 1. **Претърсване чрез дихотомия.** Опишете с думи, с най-много две изречения и без подробности, алгоритъма за дихотомично претърсване на нареден масив. В отделно изречение напишете кога, при какви условия, алгоритъмът СПИРА да обработва масива. Съставете опорна схема на метода:

Взимаме средния елемент на масива и го сравняваме с елемента, който търсим и в зависимост дали е по-голям или по-малък взимаме лявата или дясната страна на масива и повтаряме процеса. Той приключва когато индекса на лявата граница стане по-голям от този на дясната.

1. Направете обща схема на масив от n елемента.
2. Отбележете къде е средният индекс на масива и с каква променлива го бележите.
3. Отбележете коя част от масива се отхвърля (примерно) при първото изпълнение на итеративната част на алгоритъма за дихотомично търсене
4. Отбележете със стрелки как се сменят стойностите на променливите, съдържащи индексите на масива, в който претърсването ще продължава. Дайте имена на тези променливи и ги отбележете на схемата.
5. Отбележете къде е средният индекс на масива, в който търсенето продължава след тази първа стъпка и напишете формулата, по която се получава неговата стойност

Като спазвате въведените от вас означения, съставете схема на управление и програмата (успоредно на схемата) за претърсване на нареден масив по дихотомичен принцип.

Търсим стойността 15 в дадения по-долу масив




```

1  import java.util.Scanner;
2
3  public class DichoSearch {
4      public static void main(String[] args) {
5          Scanner scanner = new Scanner(System.in);
6          int[] array = new int[15];
7          int iBeg = 0;
8          int iEnd = 14;
9          int iMid = (iBeg+iEnd)/2;
10
11          int x = Integer.parseInt(scanner.nextLine());
12
13          for (int i = 0; i < 15; i++) {
14              array[i] = i + 1;
15          }
16
17          while (iEnd > iBeg) {
18              if (x < array[iMid]) {
19                  iEnd = iMid - 1;
20                  iMid = (iBeg+iEnd)/2;
21              } else if (x > array[iMid]) {
22                  iBeg = iMid + 1;
23                  iMid = (iBeg + iEnd)/2;
24              } else {
25                  iBeg = iEnd;
26              }
27
28              if (x == array[iMid]) {
29                  System.out.println("Index: " + iMid);
30                  System.out.println("Number: " + array[iMid]);
31                  break;
32              } else {
33                  System.out.println("Number doesn't exist!");
34                  break;
35              }
36          }
37      }
38  }

```

"C:\Program Files\Java\jdk-11.0.5\bin"
12
Index: 11
Number: 12

Process finished with exit code 0

Задача 2. Напишете програмата за намиране на **елемент с минимална стойност** и на неговия индекс, в едномерен масив. Материалът е изложен в Мудъл. Оградете с правоъгълник оператора, който осигурява запазване на индекса на елемента с минимална стойност. Запишете колко сравнения на стойности прави тази програма.

"C:\Program Files\Java\jdk-11.0.5\bin"
Minimal value: 1 at index: 1
Number of times: 4

Process finished with exit code 0

```

1  public class Minimal {
2      public static void main(String[] args) {
3
4          int[] array = {9, 1, 1, 1, 4};
5          int minimal = array[0];
6          int counter = 0;
7          int index = 0;
8
9          for (int i = 1; i < array.length; i++) {
10             if (array[i] < minimal) {
11                 minimal = array[i];
12                 index = i;
13             }
14             counter++;
15         }
16
17         System.out.println("Minimal value: " + minimal + " at index: " + index);
18         System.out.println("Number of times: " + counter);
19     }
20 }
21

```

Задача 3. Разучете добре алгоритъма за сортиране на един масив в друг масив (даван в час и изложен в Мудъл).

а. Опишете с думи, с най-много три изречения, как работи този алгоритъм. Не използвайте думи като *Фор* или *Джей*. Използвайте термини като : *пъти*, *обхождане*, *попълване*, *следващ*, *минимална стойност* и *дезактивиране на изтеглената стойност*.

Оригиналният масив се обхожда за да се намери максималния елемент, след което се записва на последната позиция на новия масив. След това се обхожда отново за минимални като се запазва в променлива и след това на позицията в оригиналния масив се записва максималната стойност. После се записва в новия масив минималната стойност в началото и след това целия процес се повтаря.

б. Напишете с думи какво прави операторът, който позволява да се изтегля всеки път следващата по ред минимална стойност.

Сравнява текущата стойност на променливата с тези в масива докато не намери минималната стойност след което я записва в променливата, а в началния масив записва максималната стойност.

в. Напишете колко пъти се изтегля минимална стойност по този алгоритъм, ако масивът е n – местен.

N- 1 пъти

3.

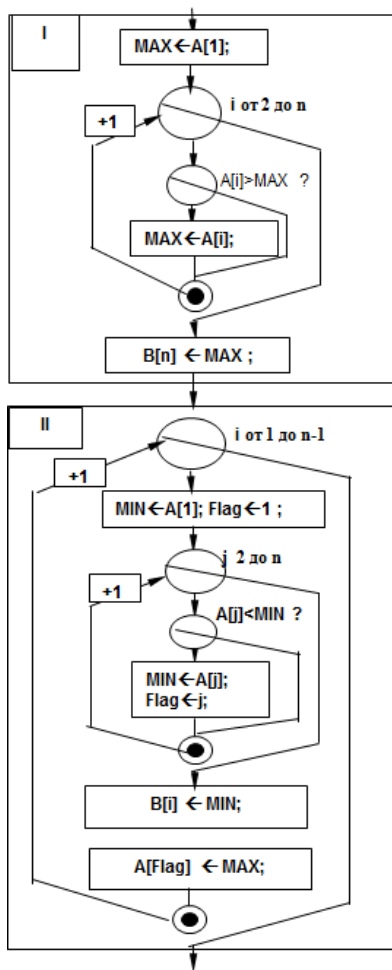
8	7
7	9
6	1
5	1
4	6
3	1
2	4
1	1
0	2

Напишете на свободните позиции първите четири цифри от факултетния си номер. Това е масиват А, стойностите на който трябва да се появят сортирани в масива В.

На следващата страница е дадена схема на управление и схема за трасиране на алгоритъма за сортирани в друг масив

Г. Напишете успоресно на схемата за управление оператори на избран от вас език за програмиране.

Е. На схемата за трасиране - Трасирайте алгоритъма при тези входни данни. Отбележете със стрелки коя стойност къде отива. Отбележете какви стойности се получават за всяко състояние на масива А до пълното му ... унищожаване..



```

1 public class Maximum {
2     public static void main(String[] args) {
3         int[] a = {2, 1, 4, 1, 6, 1, 1, 9, 7};
4         int maximum = a[0];
5         int minimum = a[0];
6         int index = 0;
7         int[] b = new int[a.length];
8
9         for (int i = 0; i < 9; i++) {
10             if (a[i] > maximum) {
11                 maximum = a[i];
12             }
13         }
14
15         b[8] = maximum;
16
17         for (int i = 0; i < 9; i++) {
18             for (int j = 0; j < 9; j++) {
19                 if (a[j] < minimum) {
20                     minimum = a[j];
21                     index = j;
22                 }
23             }
24             b[i] = minimum;
25             minimum = maximum;
26             a[index] = maximum;
27         }
28
29         for (int i = 0; i < 9; i++) {
30             System.out.print(b[i] + " ");
31         }
32     }
33 }
  
```

																																																																																																																																																																																																																																																																																																																																																																																																																																																																																																																																																																																																																																																																																																																																																																																																																																																																																																																																																																																																																																																																																																																																																																																																																																																																																																																																																																																																																					</
--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	----

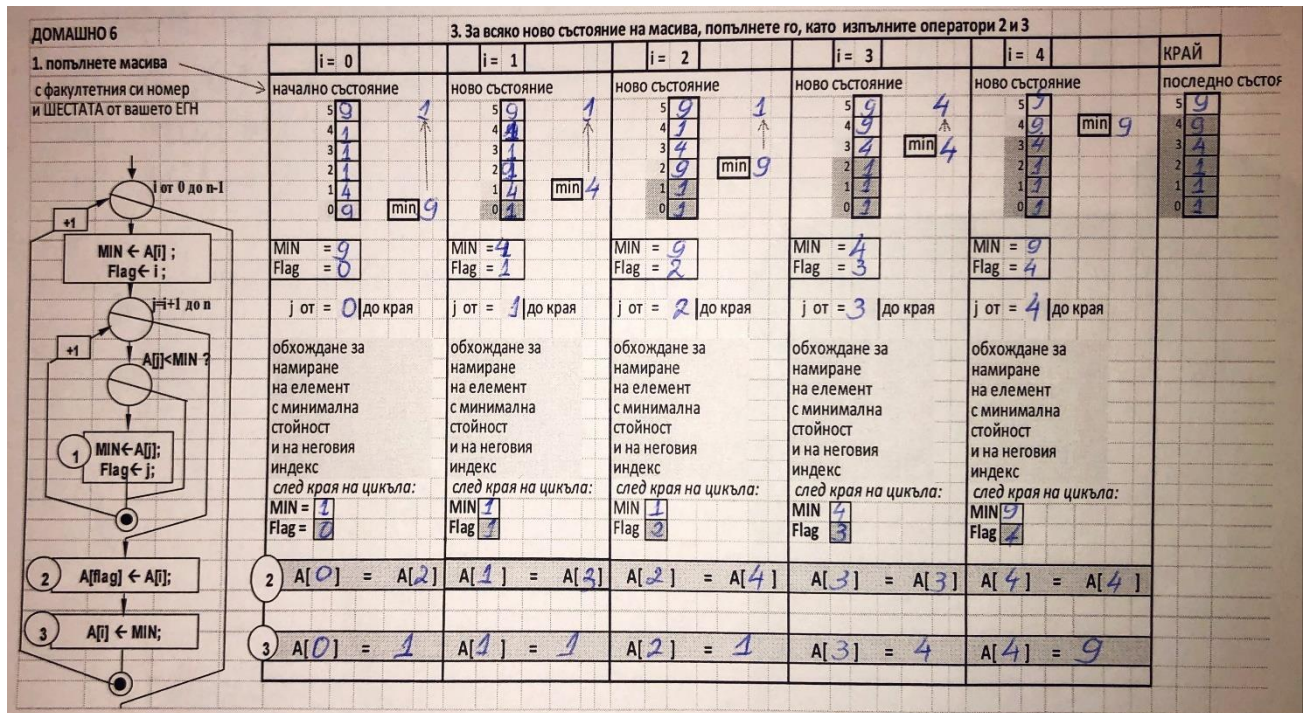
Домашно 7, 2019.

На защита се представя заедно със заданието – т.е. тази страница.

1. Пряка селекция.

Базирайки се на разгледаното в час, на материалите в Мудъл и като погледнете и разучите Екселския файл наречен *Трасиране - Пряка Селекция*, трасирайте на ръка, на схемата долу, алгоритъма с входен масив, образуван от факултетния ви номер и шестата цифра на вашето ЕГН.

ТРАСИРАНЕ



2. Пряка селекция. Първа задача съдържа *схема на управление* (това вляво) и *схема на алгоритмичните преобразувания* на метода на пряката селекция (това което попълвате, за да проследите какво прави методът с данните).

Като се базирате на схемана на управление, напишете програмата, в точно съответствие със схемата, за сортиране на масив по метода на пряката селекция.

```

1  public class Tracing {
2  public static void main(String[] args) {
3      int[] a = {9,4,1,1,1,9};
4      int flag = 0;
5      int min = 0;
6
7      for (int i = 0; i < a.length; i++) {
8          flag = i;
9          min = a[i];
10         for (int j = i+1; j < a.length; j++) {
11             if (min > a[j]) {
12                 flag = j;
13                 min = a[j];
14             }
15         }
16         a[flag] = a[i];
17         a[i] = min;
18     }
19
20     for (int i = 0; i < a.length; i++) {
21         System.out.println(a[i]);
22     }
23 }
24 }

```

"C:\Program Files\Java\jdk-11.0.5\bin"

1

1

1

4

9

9

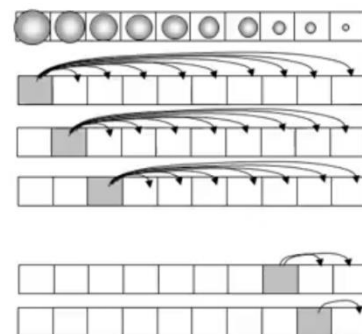
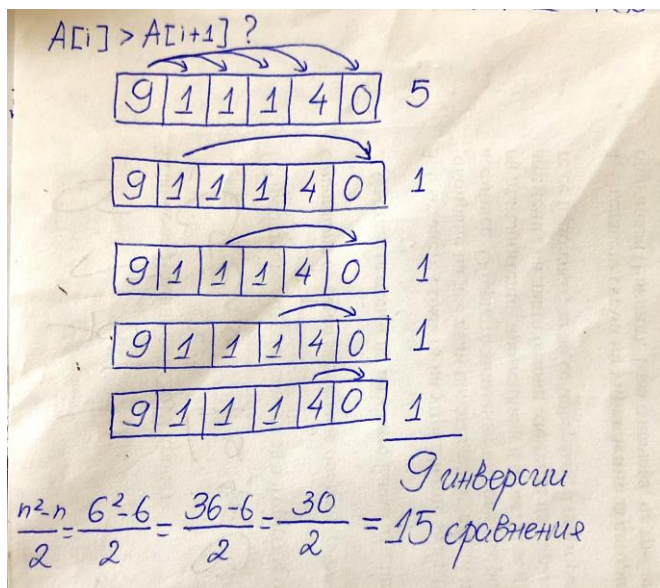
Process finished with exit code 0

3. Броене. Сложност по време.
Запишете формулата на Гаус за сумата на първите n естествени числа, приложете я и разпишете колко сравнения на стойности прави този алгоритъм. $(n^2 - n)/2$

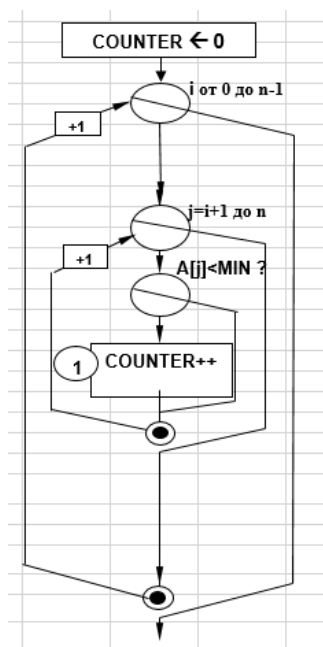
$$a_1 + a_2 + a_3 + \dots + a_n = \sum_{i=1}^n a_i$$

$$\sum_{i=1}^4 i = 1 + 2 + 3 + 4 = \frac{4(4+1)}{2} = \frac{4 \cdot 5}{2} = 10$$

4. Инверсии. Съставете масив от цифрите на факултетния си номер и втората цифра от вашето ЕГН. Колко са инверсиите? 9 инверсии.
А. Изобразете масива графично. [9, 1, 1, 1, 4, 0]
Б. Съставете схемата на преброяване, както е дадена в материала.
В. Пребройте инверсиите за ВСЯКА позиция, както е на схемата.
Съберете ги.



5. Инверсии. Съставете сами алгоритъм и програма за преброяване на инверсиите в масив. Колко сравнения прави вашият алгоритъм?



```

1 public class Inversions {
2     public static void main(String[] args) {
3         int[] a = {9,1,1,1,4,0};
4         int counter = 0;
5
6         for (int i = 0; i < a.length; i++) {
7             for (int j = i+1; j < a.length; j++) {
8                 if (a[i] > a[j]) {
9                     counter++;
10                }
11            }
12        }
13
14        System.out.println("Number of inversions: " + counter);
15    }
16 }
17
  
```

15 сравнения

Защо? - обяснете с едно изречение. Обяснете това са думи.

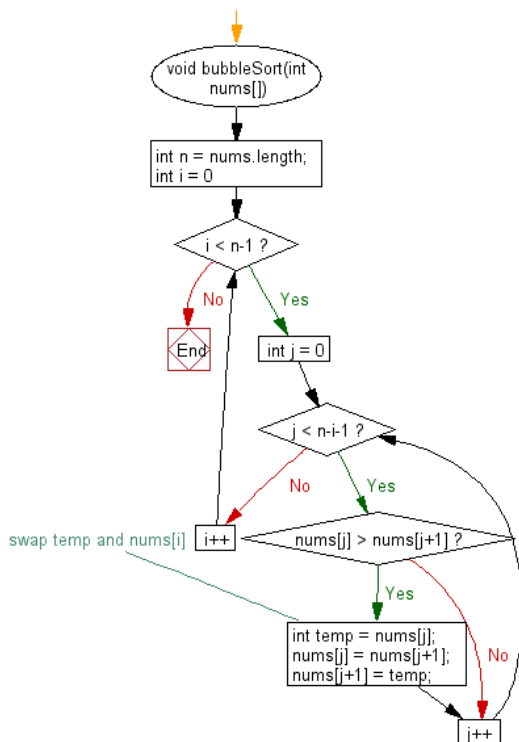
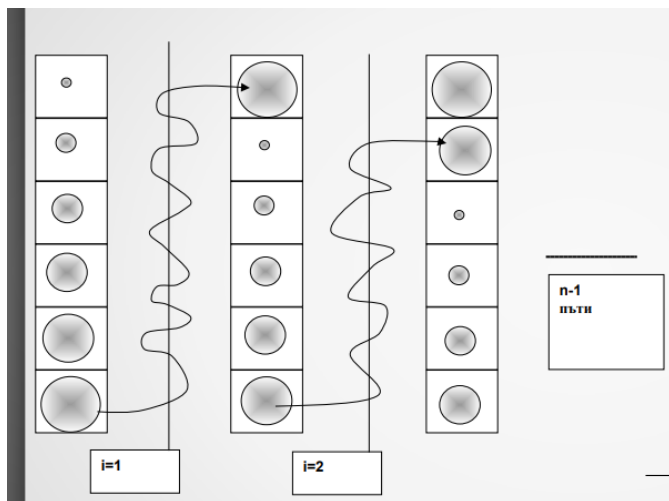
$(n^2 - n)/2$ - масива се обхожда от началния до крайния елемент и със всяко следващо обхождане лявата позиция се измества с едно напред. Ако започнем да събираме всички инверсии и се загледаме в написаното, ще установим, че ако обърнем уравнението, получаваме формулата за сумата на първите n числа – Формулата на Гаус.

Домашно 8, 2019

1. Сортиране. Метод на пряката размяна (мехурчето).

Съставете опорна схема (масивът и какво се прави с него, с означени имена на променливи и т.н.),

Съставете схема на управление със спиране по брой инверсии (почти е готова в Moodle) и успоредно на нея – програма.



```
1  import java.util.Arrays;
2
3  public class BubbleSortTest {
4      @ void bubbleSort(int nums[]) {
5          int n = nums.length;
6          for (int i = 0; i < n - 1; i++)
7              for (int j = 0; j < n - i - 1; j++)
8                  if (nums[j] > nums[j + 1]) {
9                      // swap temp and nums[i]
10                     int temp = nums[j];
11                     nums[j] = nums[j + 1];
12                     nums[j + 1] = temp;
13                 }
14     }
```


2. Отговорете писмено на всеки от следните въпроси (обсъждани в час).

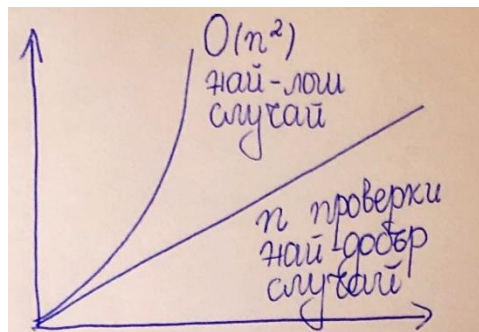
А) Колко сравнения на стойности **най-много** се правят при сортиране по метода на мехурчето. При какви входни данни стави това. Съставете схемата на преброяване и приложете подходяща формула, за да получите броя им.

При N елемента най-много сравнения се правят когато масивът е напълно обърнат спрямо приетата подредба, което е равно на $(N^2 - N)/2$ сравнения.

Б. Колко сравнения на стойности **най-малко** се правят при сортиране по метода на мехурчето. При какви входни данни стави това.

При N елемента най-малко сравнения се правят когато масивът е подреден спрямо приетата подредба, което е равно на N сравнения.

3. Направете на ръка как изглеждат теоретично двете графики (за най-добрия и най-лошия случай, посочени в задача 2) на зависимостта между големината на входа (броят на елементите в масива) и броя на сравненията. Означете на графиките за кой случай те се отнасят.



4. Направете експериментални графики (на ръка) на зависимостта между големината на входа (броят на елементите в масива) и времето за изпълнение в най-добрия случай, най-лошия случай и при случайно подредени елементи.

Времето за изпълнение измервайте чрез кода, качен в Moodle. За да получите добра графика, трябва да го направите за няколко размера на масива, колкото повече, толкова по-добре.

случайно подредени елементи

elapsed wall clock time: 2

elapsed wall clock time: 2.334

най-лошия случай

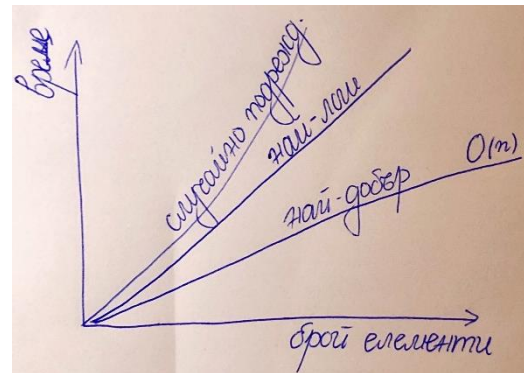
elapsed wall clock time: 0

elapsed wall clock time: 0

най-добрия случай

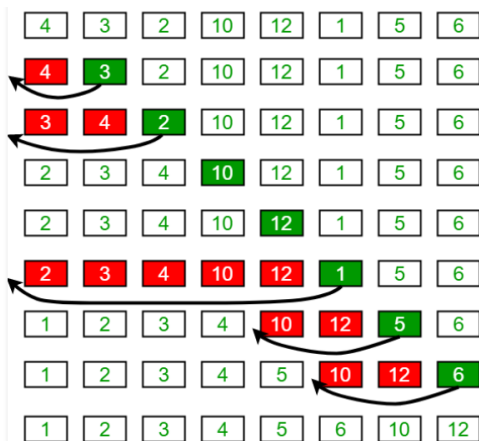
elapsed wall clock time: 1

elapsed wall clock time: 1.719



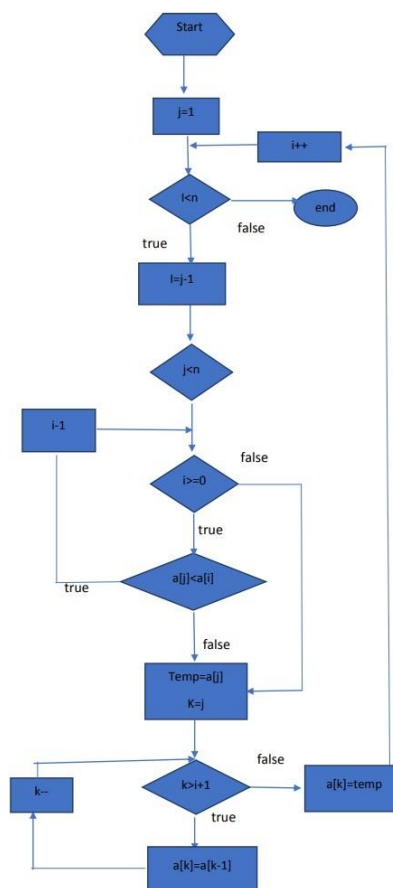
Домашно 9. Пролет 2019.

1. Сортиране. Пряко вмъкване – съставете опорна схема (масивът и какво се прави с него, с означени имена на променливи и т.н.), схема на управление и успоредно на нея – програмен текст на избран от вас език.



```
int i, key, j;
for (i = 1; i < n; i++)
{
    key = arr[i];
    j = i - 1;

    while (j >= 0 && arr[j] > key)
    {
        arr[j + 1] = arr[j];
        j = j - 1;
    }
    arr[j + 1] = key;
}
```



2. Помислете дали този метод работи по-добре (с по-малка сравнения), ако във входния масив няма инверсии. Обяснете с думи и със схема на преброяване **защо**.
Колкото по-малко инверсии има толкова по-малко сравнения има.



3. Пуснете програмата на машина и приложете снимки на екран на кода и на изхода от изпълнение.

```
int i, key, j;
for (i = 1; i < n; i++)
{
    key = arr[i];
    j = i - 1;

    while (j >= 0 && arr[j] > key)
    {
        arr[j + 1] = arr[j];
        j = j - 1;
    }
    arr[j + 1] = key;
}

/* Driver code */
int main()
{
    int arr[] = { 12, 11, 13, 5, 6 };
    int n = sizeof(arr) / sizeof(arr[0]);

    insertionSort(arr, n);
    printArray(arr, n);
}
```

C:\Users\stoya\Documents\Homework\9\1.exe
5 6 11 12 13
Process returned 0 (0x0) execution time : 0.094 s
Press any key to continue.

4. Пряко вмъкване с дихотомично търсене на мястото на вмъкване. Съставете опорна схема (масивът и как се вмъква дихотомично, с означени имена на променливи и т.н.), схема на управление и успоредно на нея – програмен текст на избран от вас език.

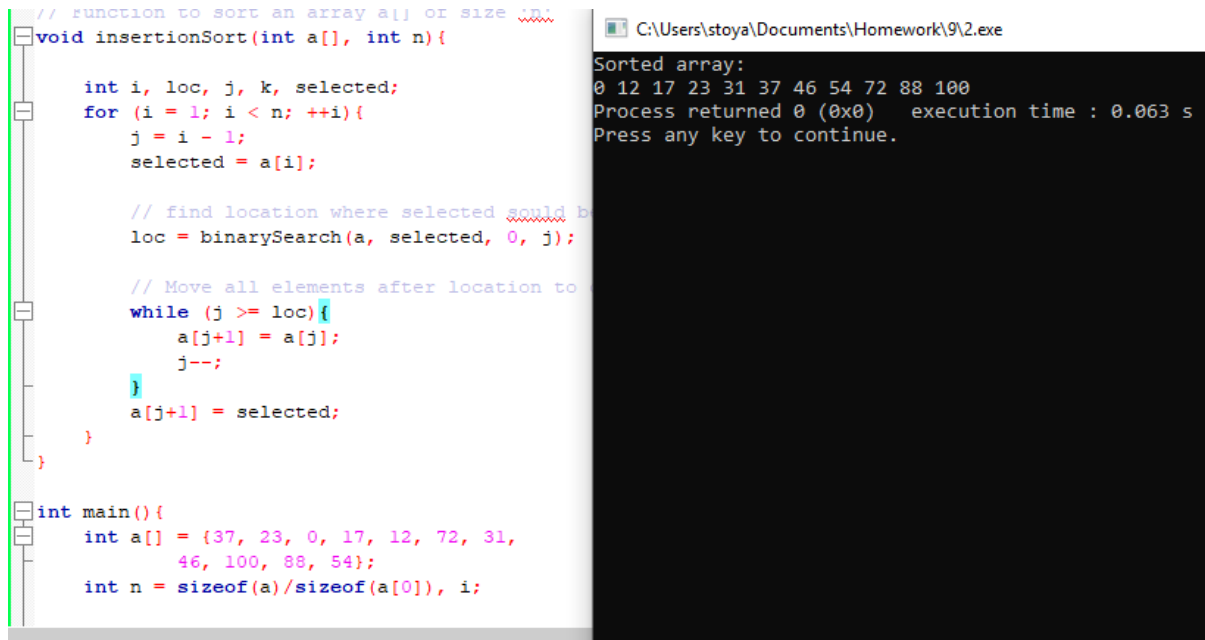
```
// Function to sort an array a[] of size 'n'
void insertionSort(int a[], int n) {

    int i, loc, j, k, selected;
    for (i = 1; i < n; ++i) {
        j = i - 1;
        selected = a[i];

        // find location where selected should be inserted
        loc = binarySearch(a, selected, 0, j);

        // Move all elements after location to create space
        while (j >= loc) {
            a[j+1] = a[j];
            j--;
        }
        a[j+1] = selected;
    }
}
```

5. Пуснете програмата на машина и приложете снимки на екран на кода и на изхода от изпълнение.



The image shows a C++ program in a code editor. The program implements an insertion sort algorithm. The `insertionSort` function takes an array `a` and its size `n`. It iterates through the array, selecting each element and finding its correct position in the sorted portion of the array using `binarySearch`. Elements greater than the selected element are shifted one position to the right, and the selected element is then inserted at the found location. The `main` function initializes an array `a` with the values {37, 23, 0, 17, 12, 72, 31, 46, 100, 88, 54} and calls `insertionSort` with `n = sizeof(a)/sizeof(a[0])`.

The output window shows the sorted array: 0 12 17 23 31 37 46 54 72 88 100. It also displays the process return code as 0 (0x0) and the execution time as 0.063 s. A prompt 'Press any key to continue.' is visible at the bottom of the output window.

```
// function to sort an array a[] of size 'n':
void insertionSort(int a[], int n){

    int i, loc, j, k, selected;
    for (i = 1; i < n; ++i){
        j = i - 1;
        selected = a[i];

        // find location where selected should be
        loc = binarySearch(a, selected, 0, j);

        // Move all elements after location to
        while (j >= loc){
            a[j+1] = a[j];
            j--;
        }
        a[j+1] = selected;
    }
}

int main(){
    int a[] = {37, 23, 0, 17, 12, 72, 31,
               46, 100, 88, 54};
    int n = sizeof(a)/sizeof(a[0]), i;
```

Sorted array:
0 12 17 23 31 37 46 54 72 88 100
Process returned 0 (0x0) execution time : 0.063 s
Press any key to continue.

6. Пуснете двата кода (по задачи 3 и 4) с еднакви входни данни и измерете разликата във времето за изпълнение, ползвайки опита си от дамошно 8.

Нормално пряко вмъкване - elapsed wall clock time:0.736

Пряко вмъкване с дихотомично търсене - elapsed wall clock time:0.596

Заданието е неразделна част от домашното и се представя при защита.

Домашно 10, 2019

- Попълнете на празните позиции в матрицата А последните три цифри от факултетния си номер, а в матрицата В – втората и третата цифра от вашето ЕГН.

	1	8	9	4		0	3
A:	3	1	1	7	*B:	8	4
	6	4	4	2		9	1
						4	7

A(nxm)

B(mxp)

А. Изобразете графично, на тази схема матрицата $C = A * B$, като празна таблица. Запишете с думи колко реда има С. Запишете колко стълба има С.

С има 3 реда и 2 стълба

	Матрица А:						Матрица В:				
	j=0	j=1	j=2	j=3			j=0	j=1			
i=0	1	8	9	4	*		i=0	0	3		
i=1	3	1	1	7			i=1	8	4		
i=2	6	4	4	2			i=2	9	1		
							i=3	4	7		
	Матрица С:										
	j=0	j=1									
i=0	161	72									
i=1	45	63									
i=2	76	52									

Б. Отбележете на схемата как бележите индексите на редовете и на стълбовете на С. Запишете ги отделно с думи, например така: с индекс i бележа редовете на С, а с индекс j - стълбовете.

В. Запишете как изчислявате **всеки** от елементите на С. Пресметнете ги. Например така

$$C_{22} = 3 \cdot 3 + 1 \cdot 8 + 1 \cdot 1 + 7 \cdot 7 = \dots = \text{Стойност}$$

$$C_{11} = 1 \times 0 + 8 \times 8 + 9 \times 9 + 4 \times 4 = 0 + 64 + 81 + 16 = 161$$

$$C_{12} = 3 \times 0 + 1 \times 8 + 1 \times 9 + 7 \times 4 = 0 + 8 + 9 + 28 = 45$$

$$C_{21} = 6 \times 0 + 4 \times 8 + 4 \times 9 + 2 \times 4 = 0 + 32 + 36 + 8 = 76$$

$$C_{22} = 1 \times 3 + 8 \times 4 + 9 \times 1 + 4 \times 7 = 3 + 32 + 9 + 28 = 72$$

$$C_{32} = 3 \times 3 + 1 \times 4 + 1 \times 1 + 7 \times 7 = 9 + 4 + 1 + 49 = 63$$

$$C_{33} = 6 \times 3 + 4 \times 4 + 4 \times 1 + 2 \times 7 = 18 + 16 + 4 + 14 = 52$$

и попълнете матрицата C със стойностите, които получавате

Г. Запишете управляващия оператор на цикъла, който обхожда по клетки всеки от редовете на C - `For (int j = 1; j < 369; j++)`. Запишете числови стойности за границите на цикъла, например `j <= 369`.

Д. Запишете управляващия оператор на цикъла, който обхожда всички редове на C.

`For (int i = 1; i < 369; i++)`

Е. Запишете вложени един в друг двата оператора за циклите, които обхождат всички клетки на C.

```
for (int j = 1; j < 3; j++) {
    for (int k = 1; k < 3; k++) {
    }
}
```

Ж. Запишете на схемата горе с какви индекси бележите редовете и стълбовете на A и на B.

i – редове, *j* – стълбове

З. Запишете общия вид на Сумата – скалярно произведение, с която изчислите всеки елемент на C.

$$C_{\text{индекс1 индекс2}} = \sum_{\text{индекс?}}^{\text{до}} A_{\text{индекс? индекс?}} \cdot B_{\text{индекс? индекс?}}$$

$$C_{\text{индекс1 индекс2}} = A_{\text{индекс? индекс?}} \cdot B_{\text{индекс? индекс?}}$$

$$C_{11} = A_{11} \cdot B_{11} + A_{12} \cdot B_{21} + A_{13} \cdot B_{31}$$

$$C_{12} = A_{11} \cdot B_{12} + A_{12} \cdot B_{22} + A_{13} \cdot B_{32}$$

$$C_{13} = A_{11} \cdot B_{13} + A_{12} \cdot B_{23} + A_{13} \cdot B_{33}$$

$$C_{21} = A_{21} \cdot B_{11} + A_{22} \cdot B_{21} + A_{23} \cdot B_{31}$$

$$C_{22} = A_{21} \cdot B_{12} + A_{22} \cdot B_{22} + A_{23} \cdot B_{32}$$

$$C_{23} = A_{21} \cdot B_{13} + A_{22} \cdot B_{23} + A_{23} \cdot B_{33}$$

$$C_{31} = A_{31} \cdot B_{11} + A_{32} \cdot B_{21} + A_{33} \cdot B_{31}$$

$$C_{32} = A_{31} \cdot B_{12} + A_{32} \cdot B_{22} + A_{33} \cdot B_{32}$$

$$C_{33} = A_{31} \cdot B_{13} + A_{32} \cdot B_{23} + A_{33} \cdot B_{33}$$

И. Запишете цикъла, който натрупва тази сума.

`For (int k = 1; k < 369; k++)`

`C[i][j] = C[i][j] + A[i][k] * B[k][j];`

2. Съставете цялата програма, като спазвате всички означения от опорната ви схема, запишете я тук и я пуснете с вашите входни данни. Приложете резпечатка на изхода.

```
"C:\Program Files\Java\jdk-11.0.5\
```

```
Product of two matrices is:
```

```
161    72
```

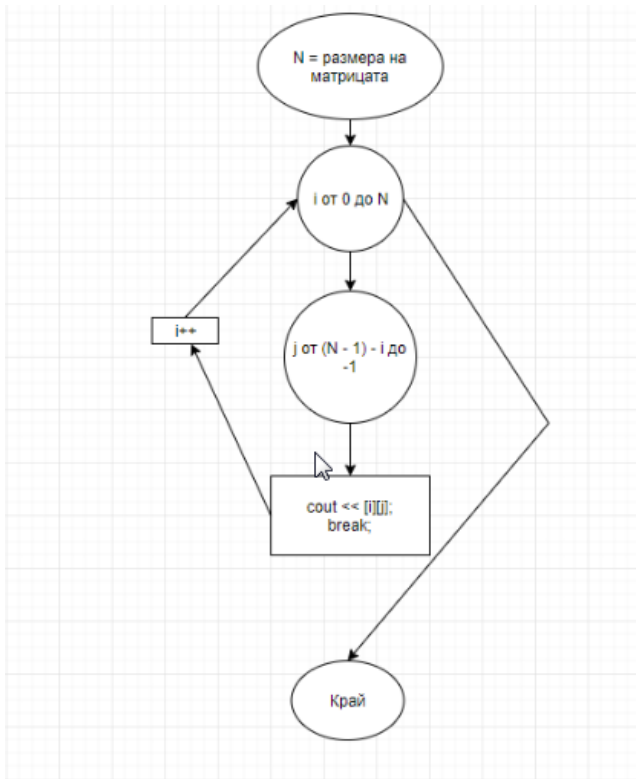
```
45     63
```

```
76     52
```

```
Process finished with exit code 0
```

```
1  ▶ public class MultiplyingMatrix {
2  ▶     public static void main(String[] args) {
3      int r1 = 3, c1 = 4;
4      int r2 = 4, c2 = 2;
5      int[][] firstMatrix = {{1, 8, 9, 4},
6                             {3, 1, 1, 7},
7                             {6, 4, 4, 2}};
8      int[][] secondMatrix = {{0, 3},
9                              {8, 4},
10                             {9, 1},
11                             {4, 7}};
12
13      // Multiplying Two matrices
14      int[][] product = multiplyMatrices(firstMatrix, secondMatrix, r1, c1, c2);
15
16      // Displaying the result
17      displayProduct(product);
18  }
19
20  @ public static int[][] multiplyMatrices(int[][] firstMatrix, int[][] secondMatrix, int r1, int c1, int c2) {
21      int[][] product = new int[r1][c2];
22      for (int i = 0; i < r1; i++) {
23          for (int j = 0; j < c2; j++) {
24              for (int k = 0; k < c1; k++) {
25                  product[i][j] += firstMatrix[i][k] * secondMatrix[k][j];
26              }
27          }
28      }
29
30      return product;
31  }
32
33  @ public static void displayProduct(int[][] product) {
34      System.out.println("Product of two matrices is: ");
```

3. Съставете 1. Опорна схема. 2. Алгоритъм 3. Програма за обхождане на всички елементи от неглавния диагонал на квадратна матрица.



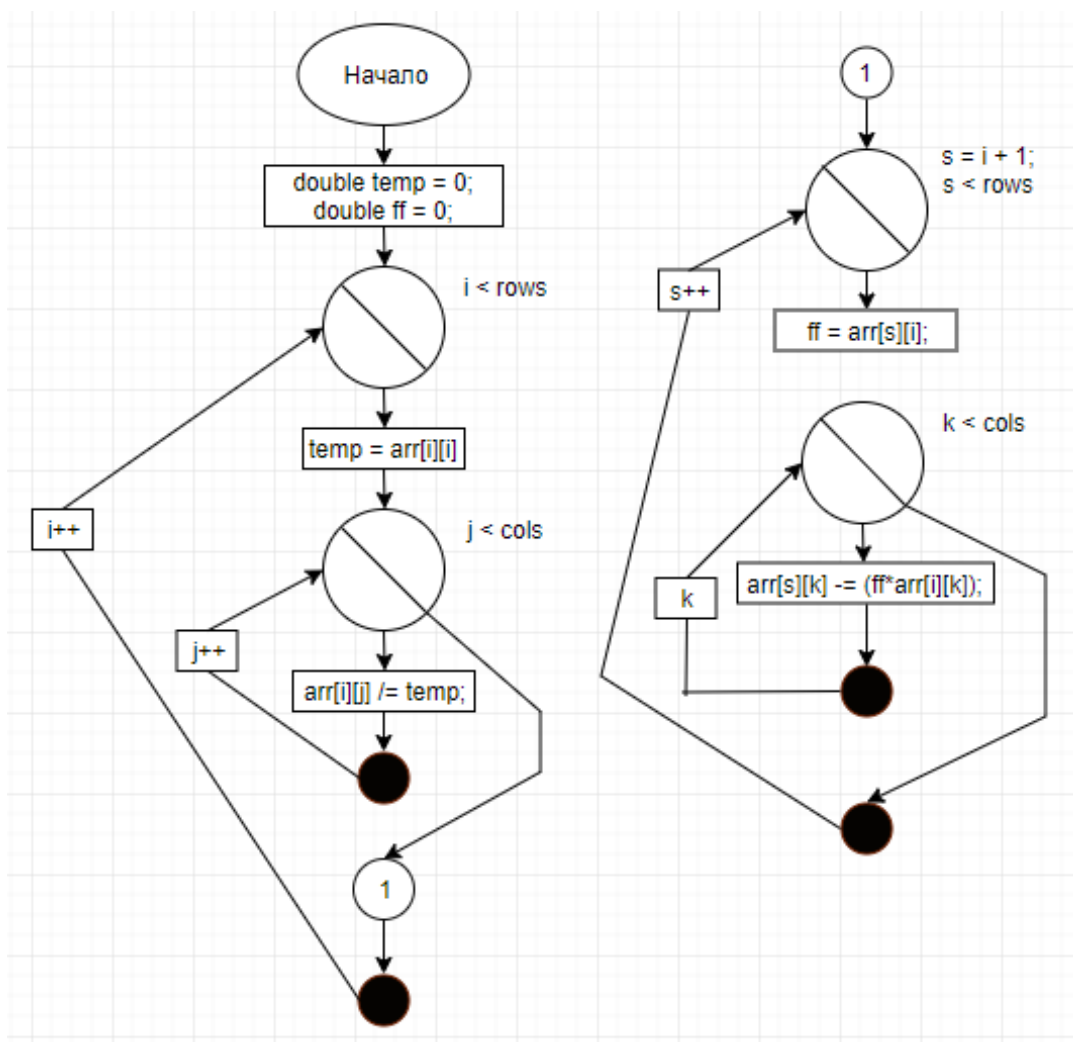
```
1 public class Diagonal {
2     static void printSecondaryDiagonal(int[][] mat, int n)
3     {
4         System.out.print("Secondary Diagonal: ");
5
6         for (int i = 0; i < n; i++) {
7             for (int j = 0; j < n; j++) {
8
9                 // Condition for secondary diagonal
10                if ((i + j) == (n - 1)) {
11                    System.out.print(mat[i][j] + ", ");
12                }
13            }
14        }
15        System.out.println("");
16    }
17
18    public static void main(String[] args)
19    {
20        int n = 4;
21        int[][] a = { { 1, 2, 3, 4 },
22                      { 5, 6, 7, 8 },
23                      { 1, 2, 3, 4 },
24                      { 5, 6, 7, 8 } };
25
26        printSecondaryDiagonal(a, n);
27    }
28 }
```

"C:\Program Files\Java\jdk-11.0.5\bin
Secondary Diagonal: 4, 7, 2, 5,

Process finished with exit code 0

Домашно 11, 2019

1. В модул е поставен файл, съдържащ схема на постъпково изпълнение на метода на Гаус за решаване на СЛУ. Попълнете го докрай на ръка с посочените там входни данни за коефициентите.
2. Като ползвате схемите от лекционния материал към темата, съставете програмата за правия ход (тя е почти написана) за решаване на СЛУ по метода на Гаус и я пуснете с примера от вашите входни данни по задача 1. Приложете към домашното алгоритмичната схема на метода и програмата (успоредно на схемата). Приложете снимка на екрани от изпълнението – вход и изход. Напишете извода за получените резултати – сравнение на вашето решение от задача 1 и решението, получено при изпълнение на програмата.



```

1  ► public class Gaus {
2  ►  public static void main(String[] args) {
3      double[][] arr = {{2, 9, 2, 134},
4                          {9, -3, 7, 12},
5                          {0, 6, 5, -36}};
6
7      int rows = 3;
8      int cols = 4;
9      double temp = 0;
10     double ff = 0;
11
12     for (int i = 0; i < rows; i++) {
13         temp = arr[i][i];
14
15         for (int j = i; j < cols; j++) {
16             arr[i][j] /= temp;
17         }
18
19         for (int s = i+1; s < rows; s++) {
20             ff = arr[s][i];
21             for (int k = 0; k < cols; k++) {
22                 arr[s][k] -= (ff*arr[i][k]);
23             }
24         }
25     }
26 }
27 }

```

"C:\Program Files\Java\jdk-11.0.5\bin\java.exe"

1 4.5 1 67

0 1 0.045977 13.5862

0 0 1 -24.8759

Process finished with exit code 0