

Домашни работи
по
“Увод в алгоритмите и програмирането”

Теодор Мангъров
(**F113621**)

21 май, 2024г.

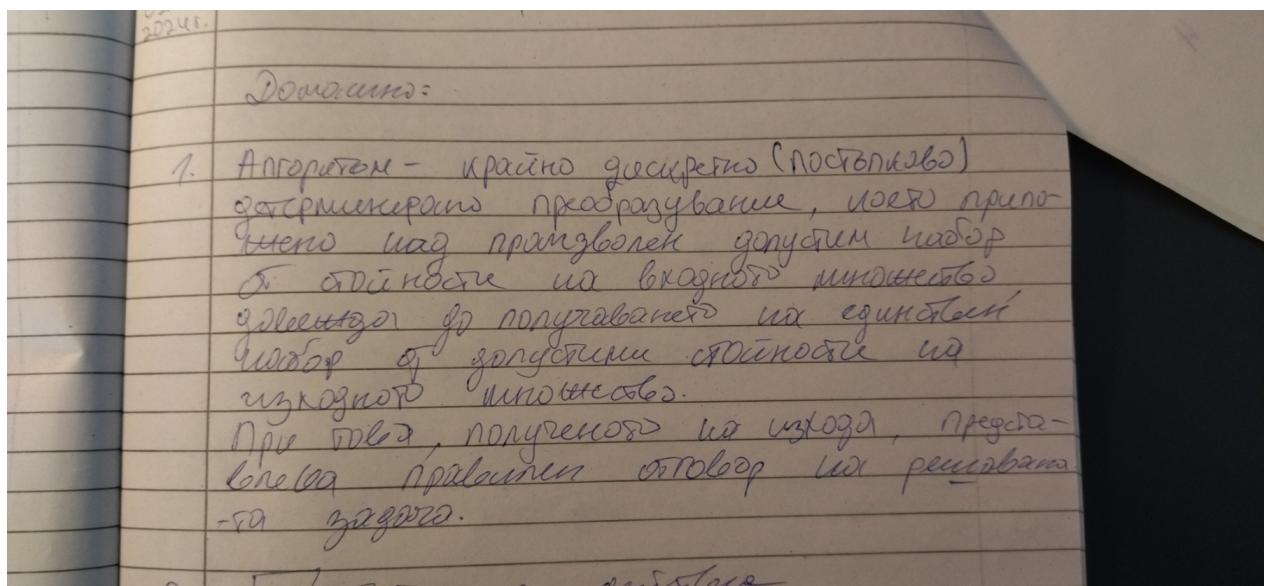
ДОМАШНО 1

Краен срок: сряда, 6 март 2024, 23:50 PM

Предадено на: сряда, 28 февруари 2024, 18:00 PM

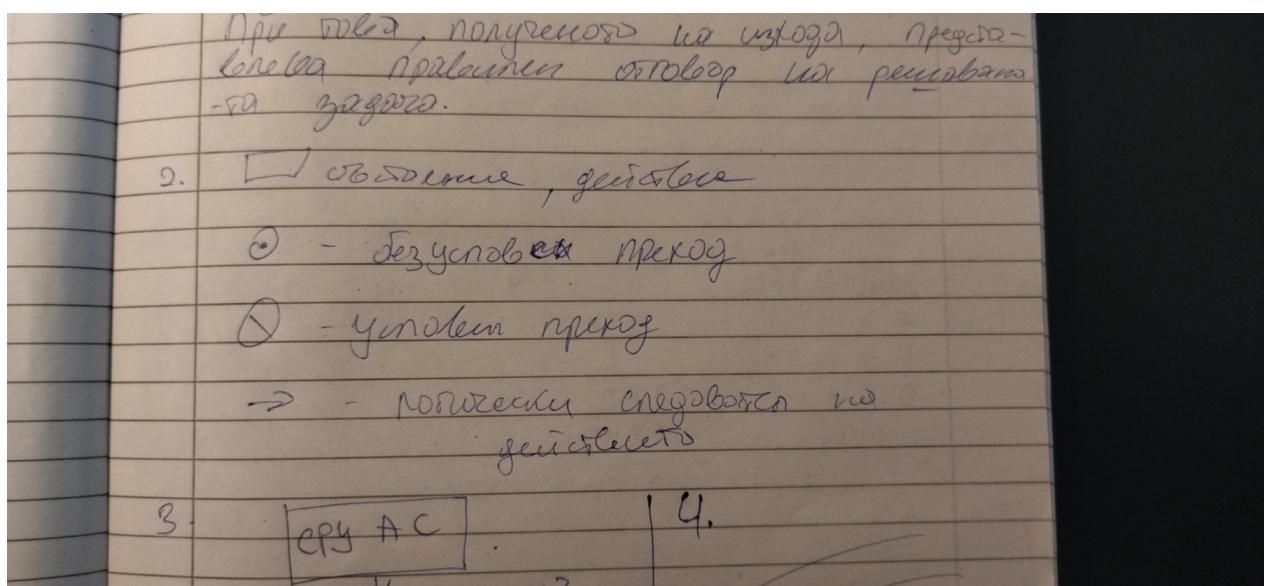
- Препишете определението за алгоритъм от лекцията. Опитайте се да го разберете добре. Ще обсъждаме всяко свойство на лекциите.

1.1 ОТГОВОР



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

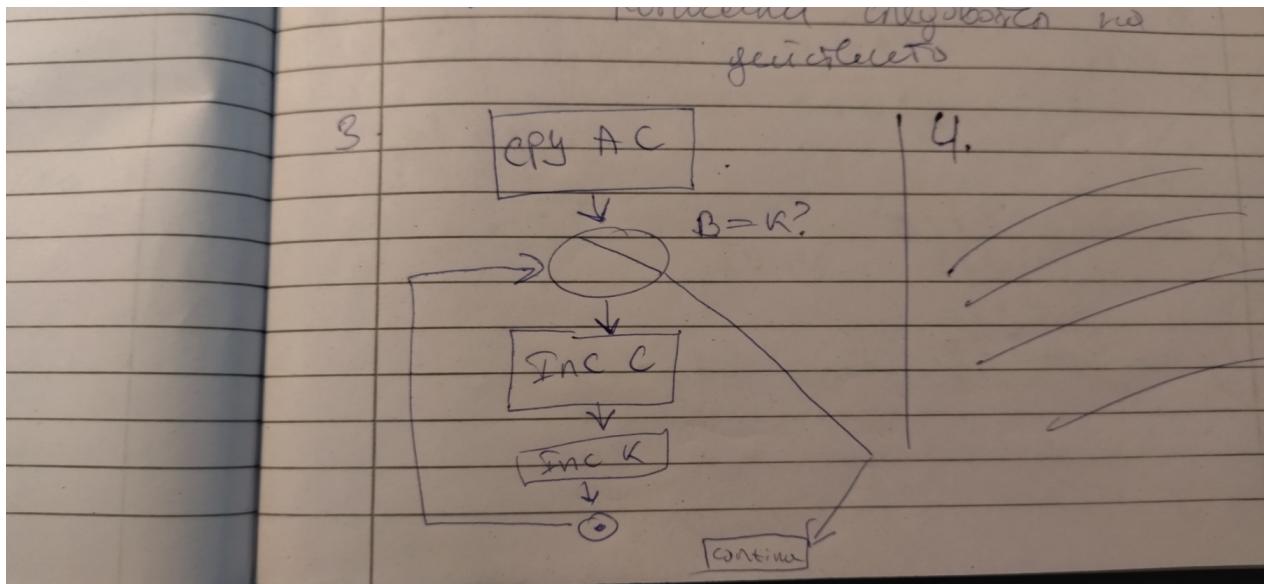
1.2 ОТГОВОР



3. Съставете схемата на управление за алгоритъма за RAM машина от лекцията (събиране на две цели положителни числа) и съответния програмен текст наproto-асемблер.

- (а) Обърнете внимание на това как решаването на задачата става чрез въвеждане на дългото променливи в паметта за данните, в случая - брояч.
- (б) Обърнете внимание на двата базови елемента на управление - безусловен и условен преход и как те се използват, за да се програмира повторение (цикъл).

1.3 ОТГОВОР

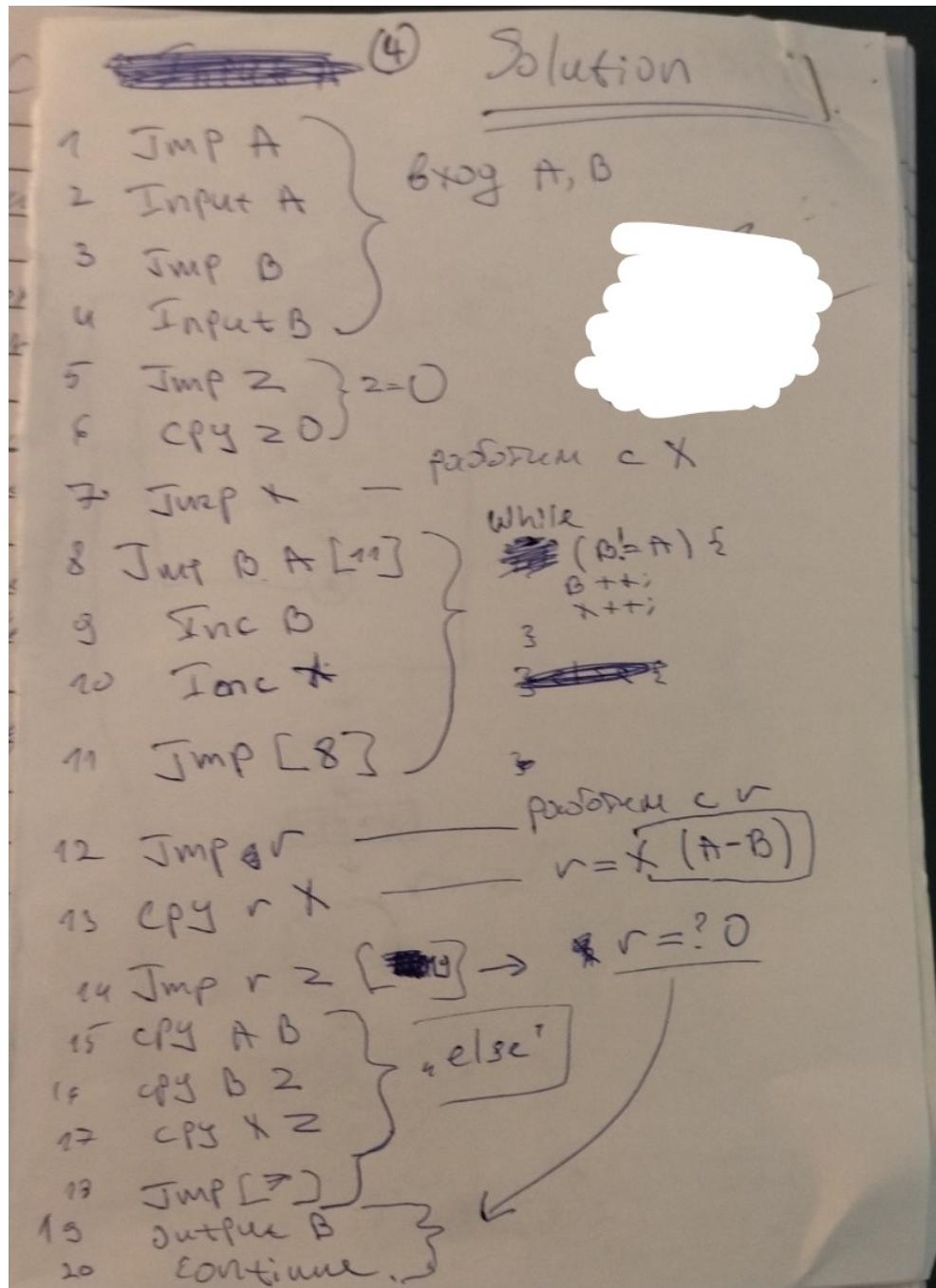


Забележка: Схемата е с презумпцията, че "B" и "K" вече съществуват в паметта

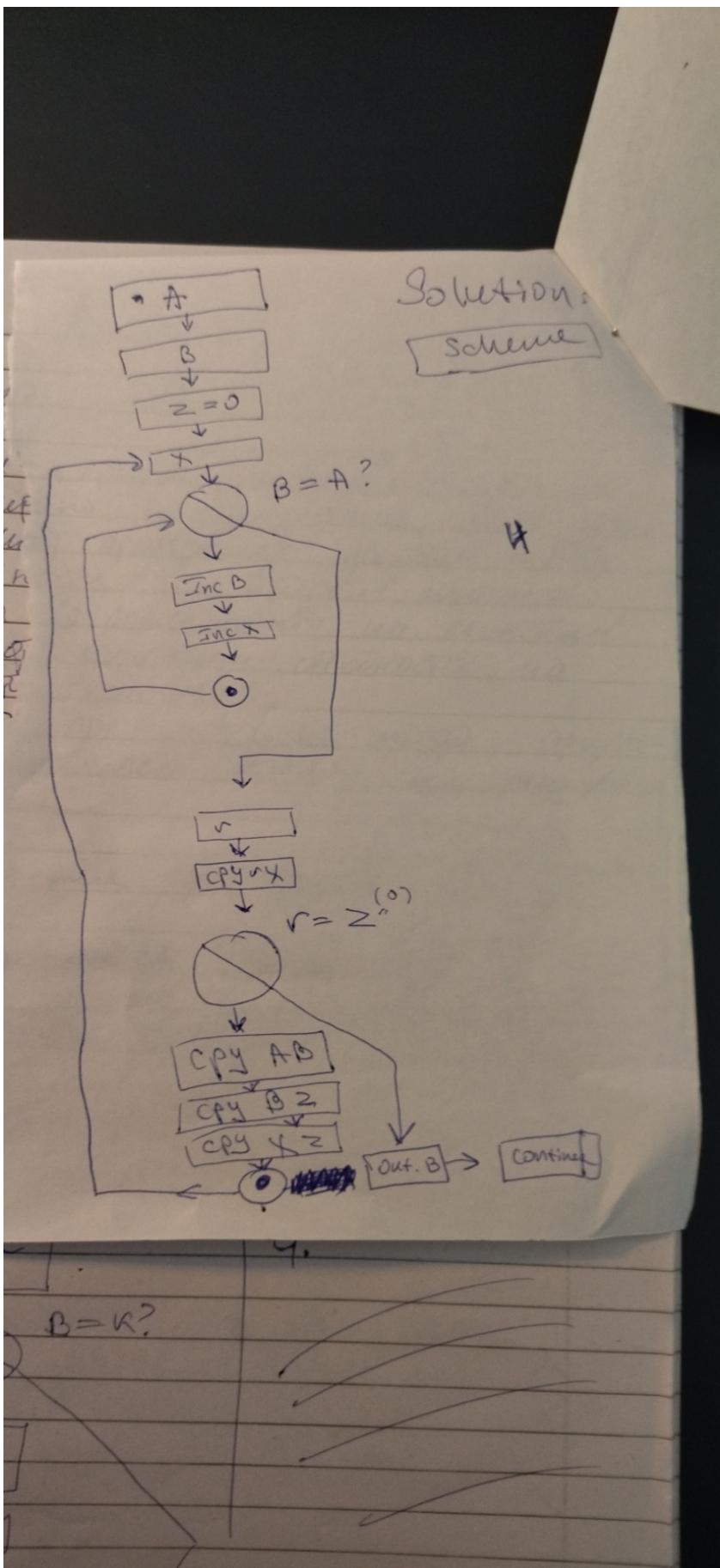
4. Съставете схема на управление за алгоритъма на Евклид, така, както той е описан в оригинала (Евклид). Това означава, че трябва да ползвате условни и безусловни преходи, за да организирате управлението. Приемете, че машината може да изпълнява следните операции над цели числа:

- (а) Намиране на остатък от целочислено делене (A на B) : MOD (A,B)
- (б) Присвояване на стойност, например $C = \text{MOD} (A,B)$
- (в) Условен преход, като условието е дали $\text{Нешо} = 0$

1.4 ОТГОВОР

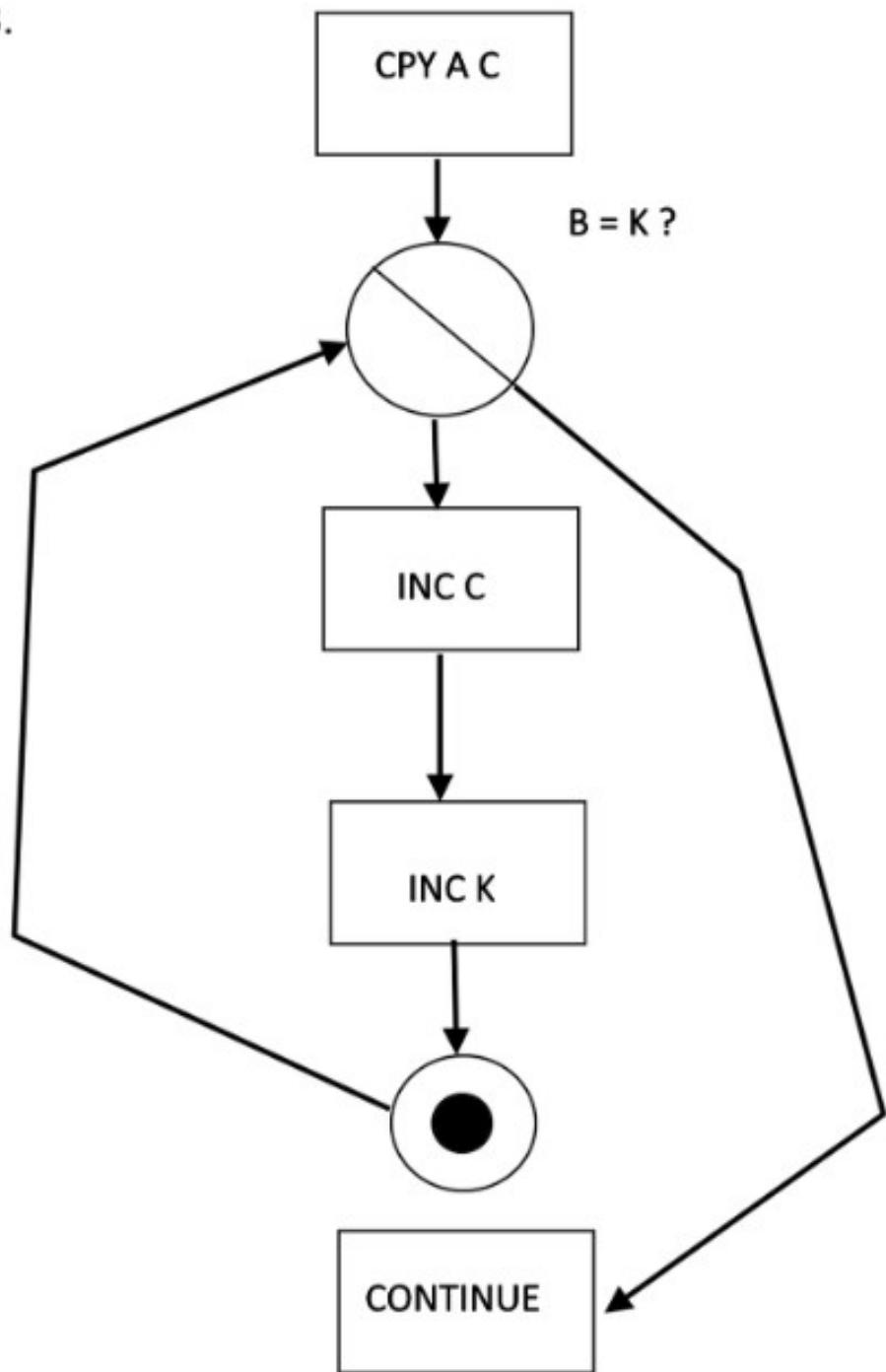


Фигура 1: алгоритъм

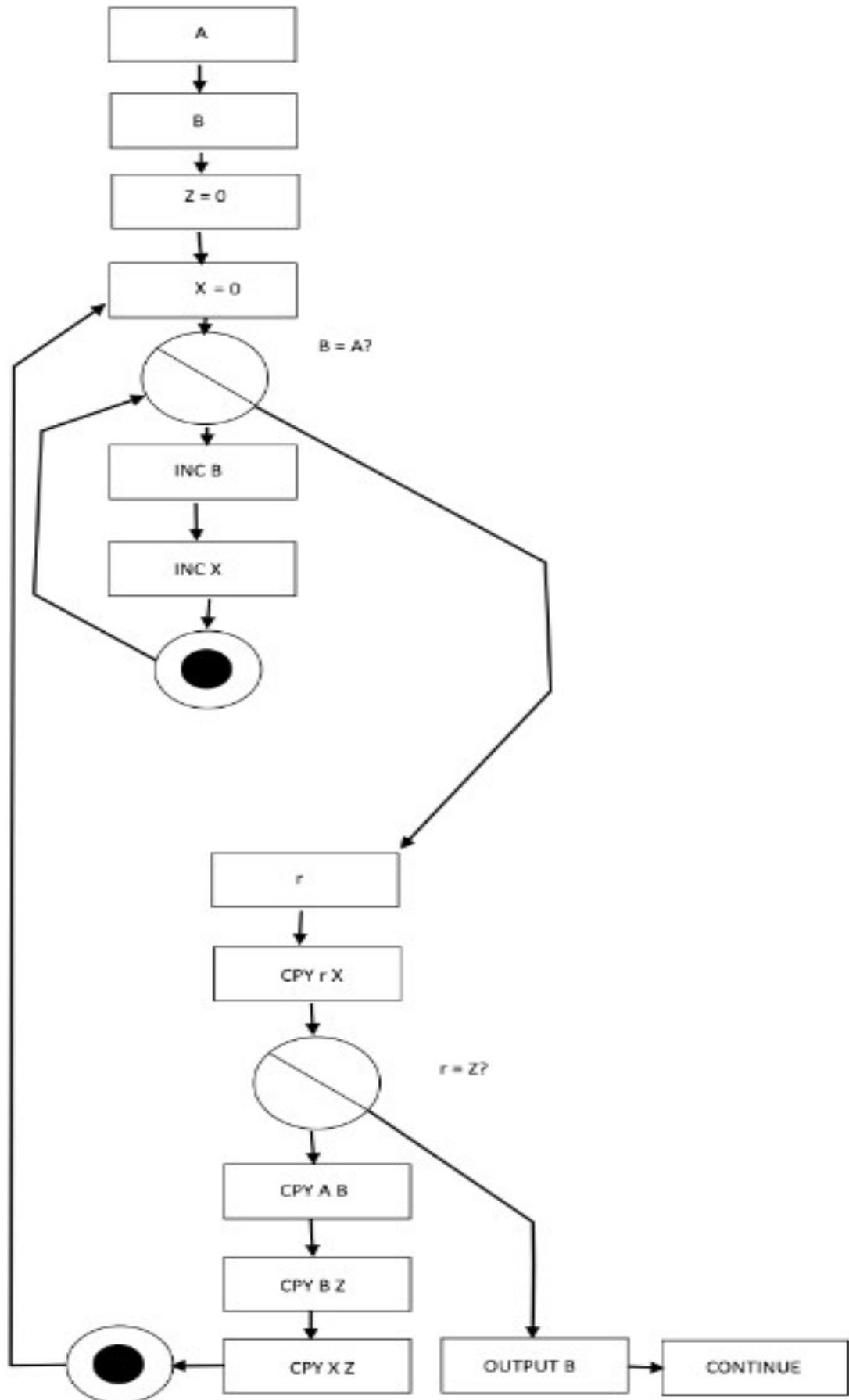


Фигура 2: опорна схема

3.



Фигура 3: опорна схема moodle



Фигура 4: опорна схема moodle

ДОМАШНО 2

Краен срок: сряда, 13 март 2024, 00:00 AM

Предадено на: неделя, 10 март 2024, 17:55 PM

- Съставете програма за алгоритъма на Евклид, в която няма оператор за цикъл. Ползвайте схемата от предходното домашно. Там няма рекурсия! Предайте снимка на еcran с кода и с изхода от изпълнение.

2.1 ОТГОВОР

```
1 #include <iostream>
2
3 int main() {
4
5     std::cout << "HOMEWORK SOLUTION 1" << std::endl;
6     int a, b , r;
7
8     std::cin >> a;
9     std::cin >> b;
10
11    if(a > b){
12        start:
13        r = a % b;
14        std::cout << "r = " << r << std::endl << std::endl;
15        if(r > 0){
16            a = b;
17            std::cout << a << std::endl;
18            b = r;
19            std::cout << b << std::endl;
20            goto start;
21
22        }
23        else{
24            goto end;
25        }
26
27
28    end:
29        std::cout << "result: " << b << std::endl;
30        return 0;
31    }
32    return 1;
33 }
```

HOMEWORK SOLUTION 1

158

16

$r = 14$

16

14

$r = 2$

14

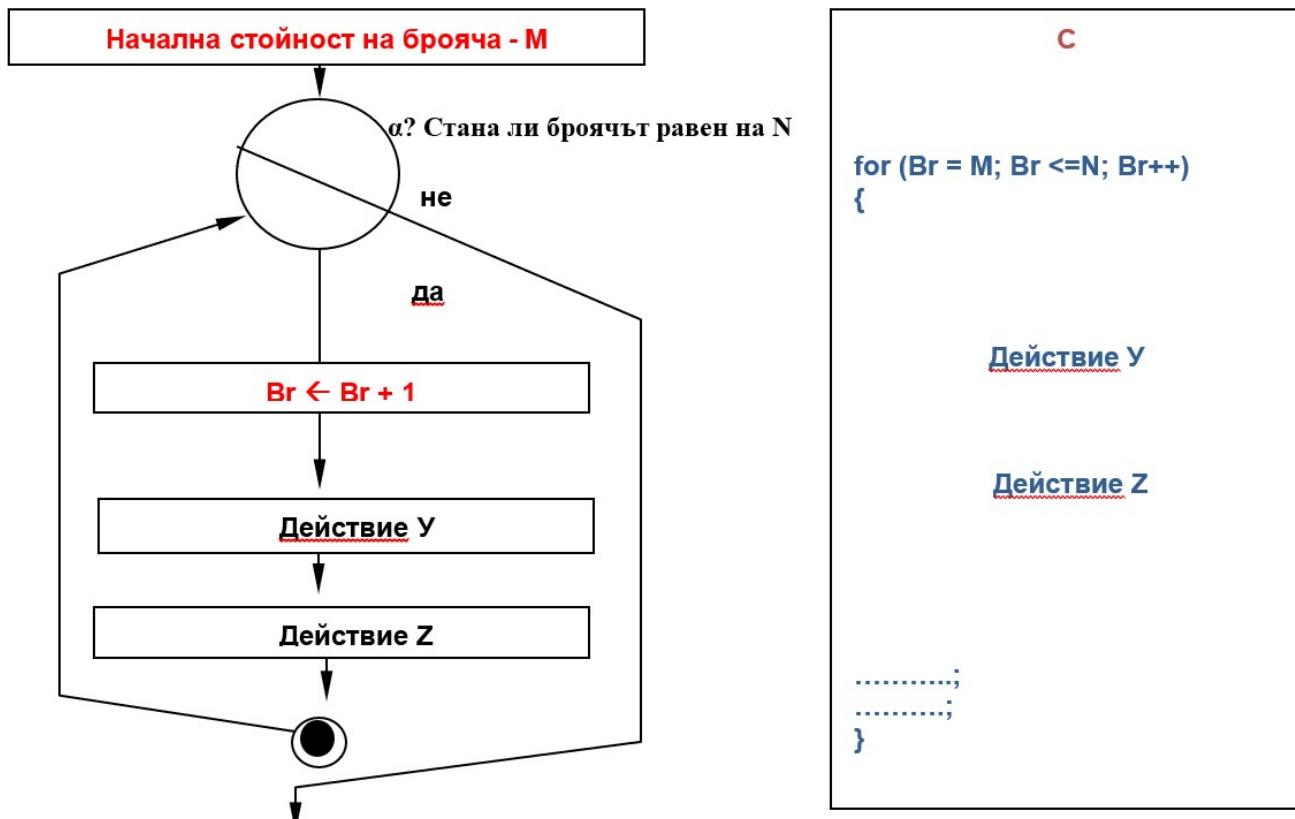
2

$r = \theta$

result: 2

Фигура 5: изход от изпълнение

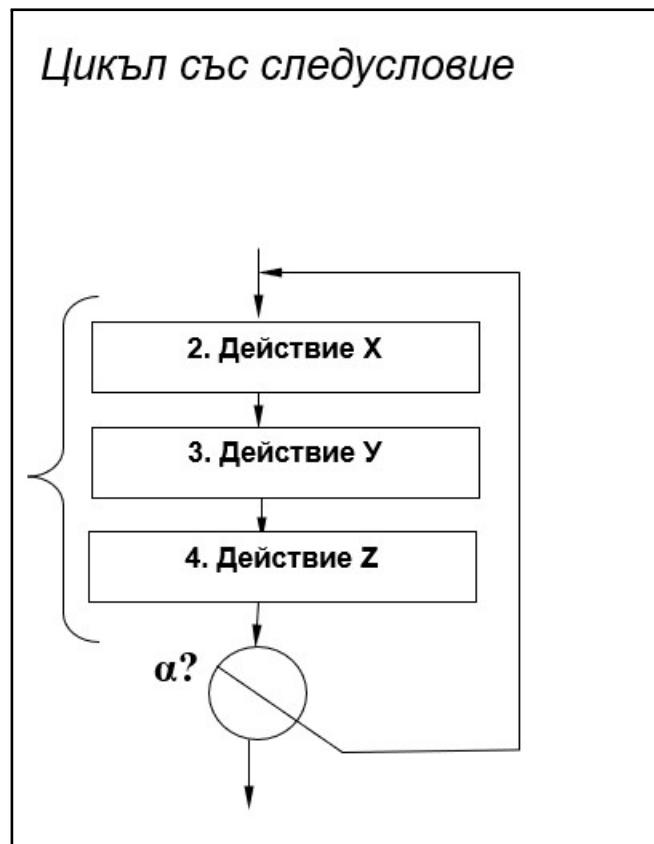
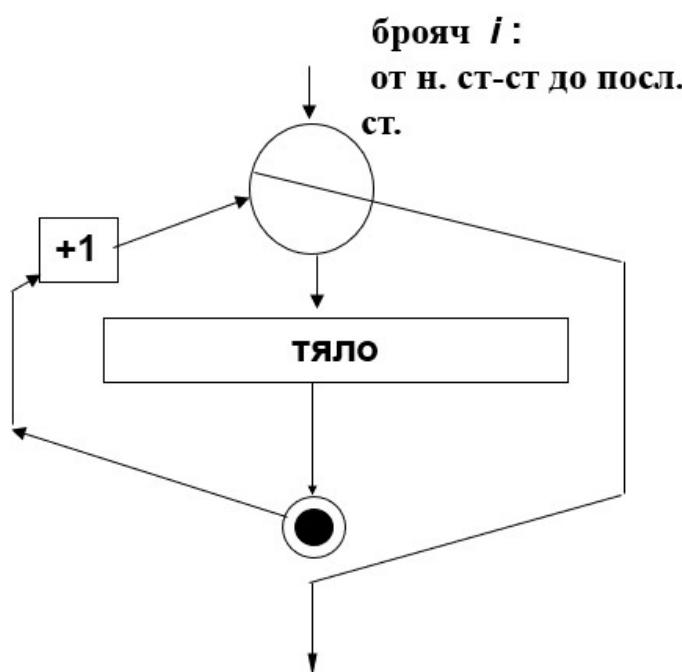
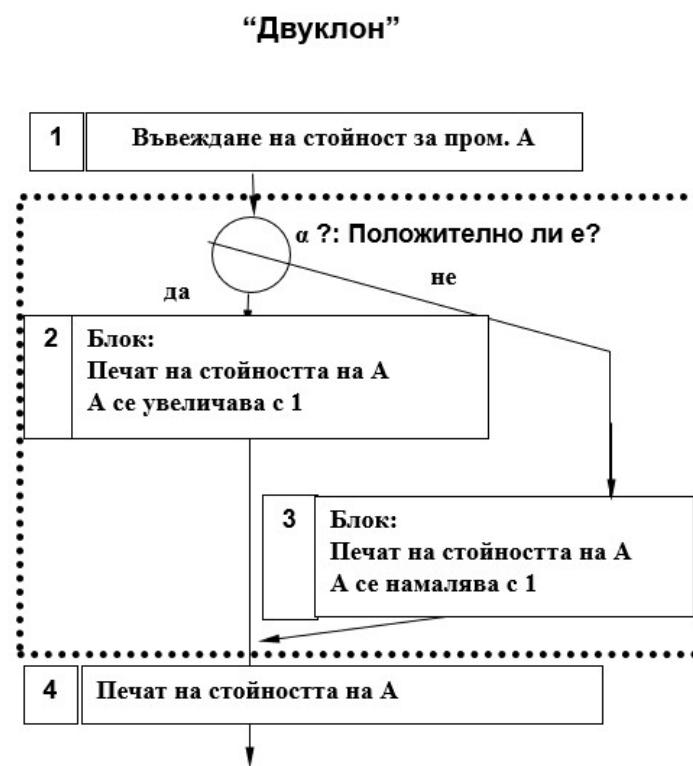
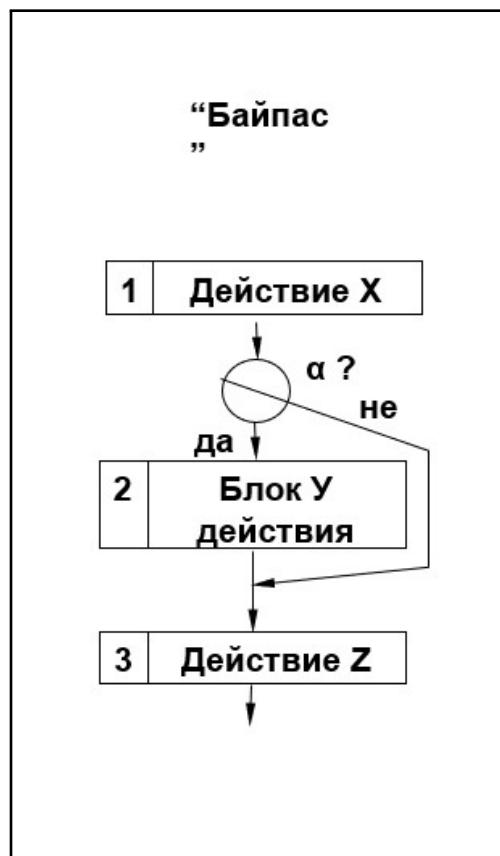
2. Като се базирате на основните елементи на управление, дадени в презентациите към тази тема, съставете петте схеми на основните Конструкти на Управление - Клоновете (двуклон и байпас) и на Повторенията (цикли с предусловие, със следусловие и по брояч). Може на ръка, може да копирате схемите, както ви е по-удобно. Тези конструкти 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:



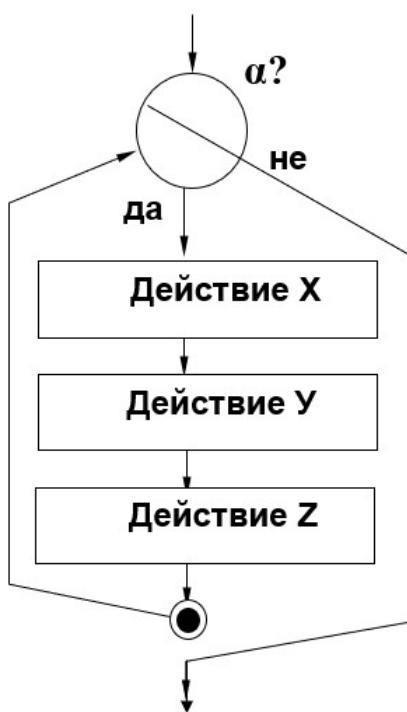
Фигура 6: Цикъл с вграден брояч

Схемите заемат около една страница и половина. Помислете над тях. Ще обсъждаме проблеми, ако се появят, в час.

2.2 ОТГОВОР



Цикъл с предусловие



3. Съставете схемата на управление 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 презентация към тази тема, а алгоритъмът е подробно разработван в час. Встрани от схемата напишете програмен текст на предпоетан от вас език.

2.3 OTTOBOP

```
1 #include <iostream>
2
3 int main(){
4     std::cout << "HOMEWORK SOLUTION 2" << std::endl;
5     const int N = 5;
6
7
8     int a[N] = { 5, 3, 2, 0 };
9     int b[N] = { 4, 8, 7, 9 };
10
11    int c[N];
12
13    int carry = 0;
14    int sum;
15
16    int k = 0;
17
18 start:
19    if(k >=0 && k <= N-1){
20        std::cout << "\nk: " << k << std::endl;
21        std::cout << "-----" << std::endl;
22        sum = a[k] + b[k] + carry;
23        c[k] = sum % 10;
24        carry = sum / 10;
25
26        std::cout
27            << "a: " << a[k]
28            << "\nb: " << b[k]
29            << "\nsum: " << sum
30            << "\nc[" << k << "]: " << c[k]
31            << "\ncarry: " << carry <<
32        std::endl;
33
34
35        k++; // k = k+1
36        goto start;
37
38    } else {
39        goto end;
40    }
41
42 end:
43 // 0 - 4 -> 5 -> 5-1 -> i[4];
44 std::cout << "\n\nRESULT (c[" << k-1 << "]): " << c[k-1] << std::endl;
45     return 0;
46
47     return 1;
48 }
```

4. Реализирайте съставената по предходната тачка програма и проверете дали работи за събиране на числа (представени като вектори) в десетичен код. Предайте снимка на еcran с кода и изход.

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>.

2.4 OTTOBOP

```
1 #include <iostream>
2
3 int main(){
4     std::cout << "HOMEWORK SOLUTION 3" << std::endl;
5     const int N = 5;
6
7
8     int a[N] = { 3, 8, 6, 5 };
9     int b[N] = { 4, 8, 7, 9 };
10
11    int c[N];
12
13    int carry = 0;
14    int sum;
15
16    int k = N-1;
17
18 start:
19    if(k >= 0){
20        std::cout << "\nk: " << k << std::endl;
21        std::cout << "-----" << std::endl;
22
23        sum = a[k] + b[k] + carry;
24        c[k] = sum % 10;
25        carry = sum / 10;
26
27        std::cout
28            << "a: " << a[k]
29            << "\nb: " << b[k]
30            << "\nsum: " << sum
31            << "\nc[" << k << "]": " << c[k]
32            << "\ncarry: " << carry <<
33        std::endl;
34
35
36        k--;
37        goto start;
38
39    } else {
40        goto end;
41    }
42
43 end:
44    std::cout << std::endl;
45    for(int i = 0; i < N; i++)
46        std::cout << c[i];
47    return 0;
48
49    return 1;
50 }
```

HOMWORK SOLUTION 3

```
k: 4
-----
a: 0
b: 0
sum: 0
c[4]: 0
carry: 0

k: 3
-----
a: 5
b: 9
sum: 14
c[3]: 4
carry: 1

k: 2
-----
a: 6
b: 7
sum: 14
c[2]: 4
carry: 1

k: 1
-----
a: 8
b: 8
sum: 17
c[1]: 7
carry: 1

k: 0
-----
a: 3
b: 4
sum: 8
c[0]: 8
carry: 0
```

87440

Фигура 7: изход от изпълнение

ДОМАШНО 3

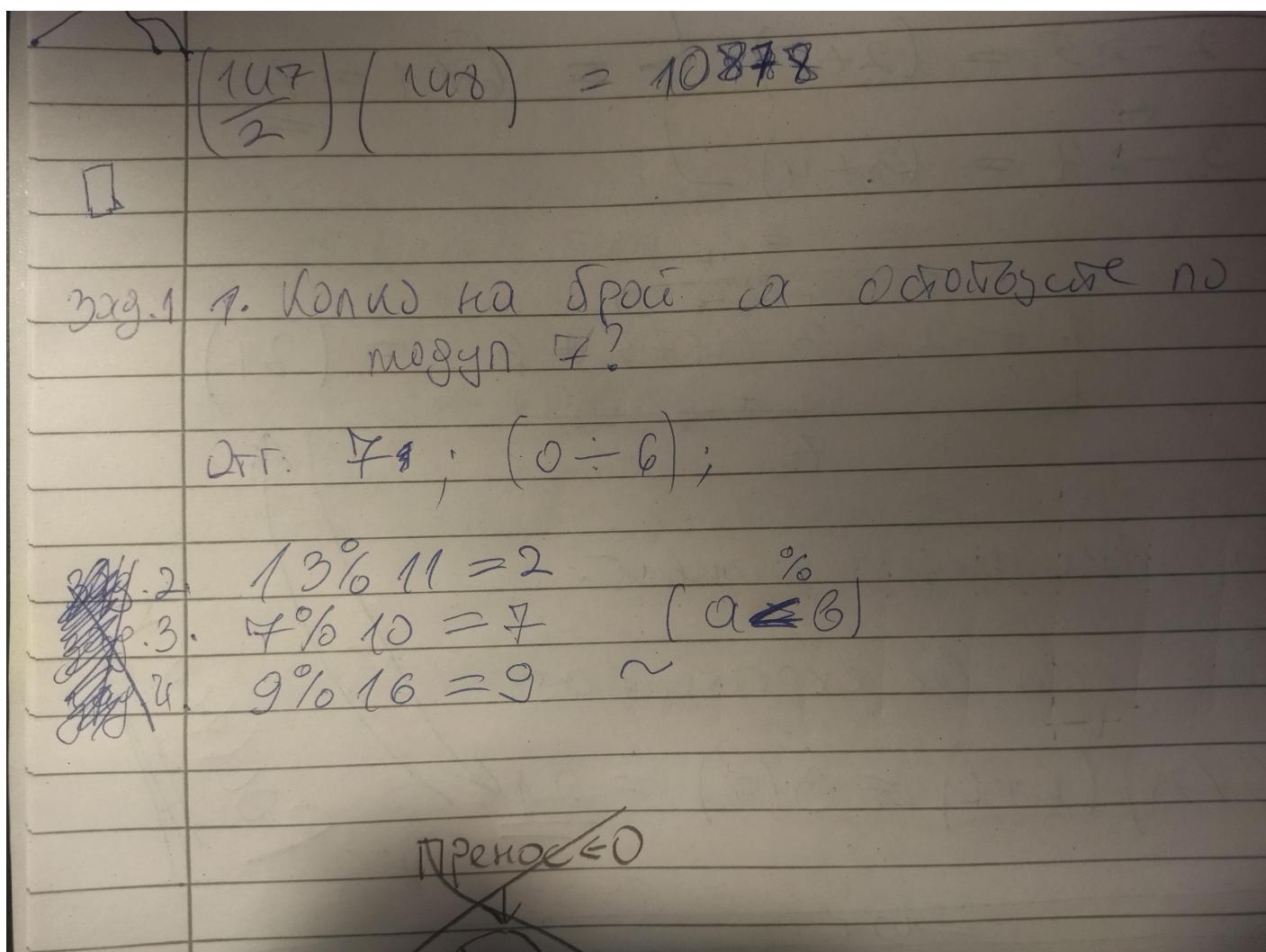
Краен срок: вторник, 19 март 2024, 23:59 PM

Предадено на: четвъртък, 14 март 2024, 15:44 PM

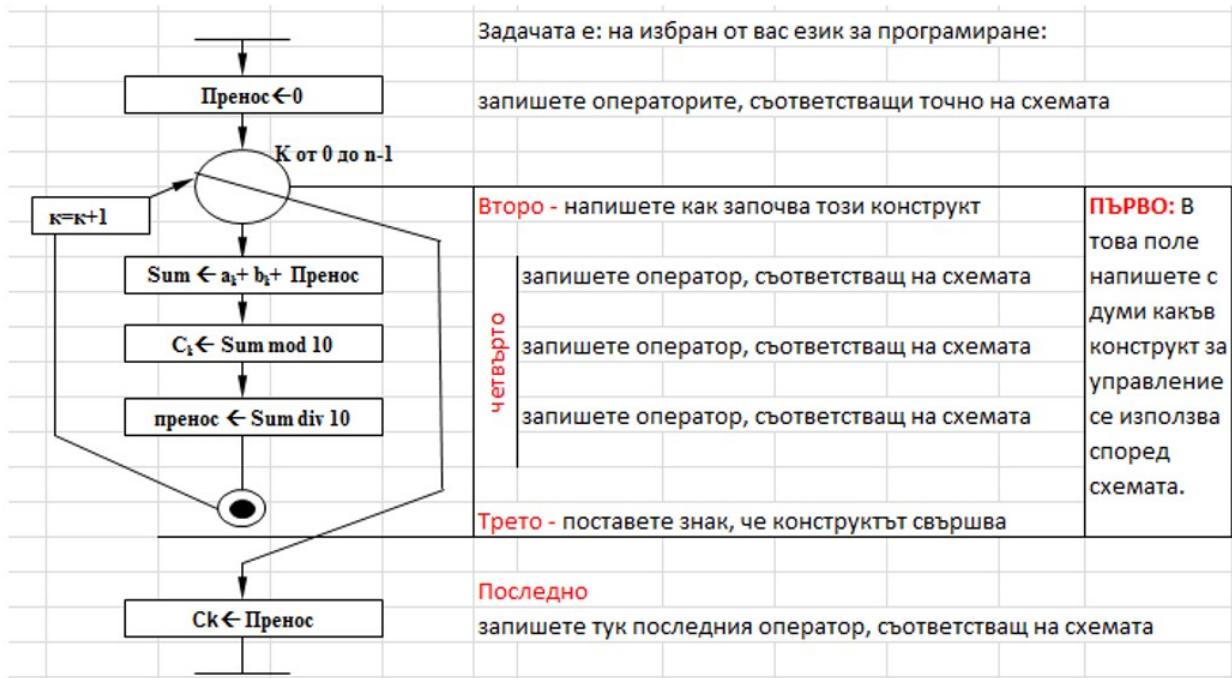
1. Отговорете писмено на следните въпроси:

- (а) Колко на брой са остатъците по модул 7;
- (б) Колко е 13 по модул 11;
- (в) Колко е 7 по модул 10;
- (г) Колко е 9 по модул 16.

3.1 ОТГОВОР

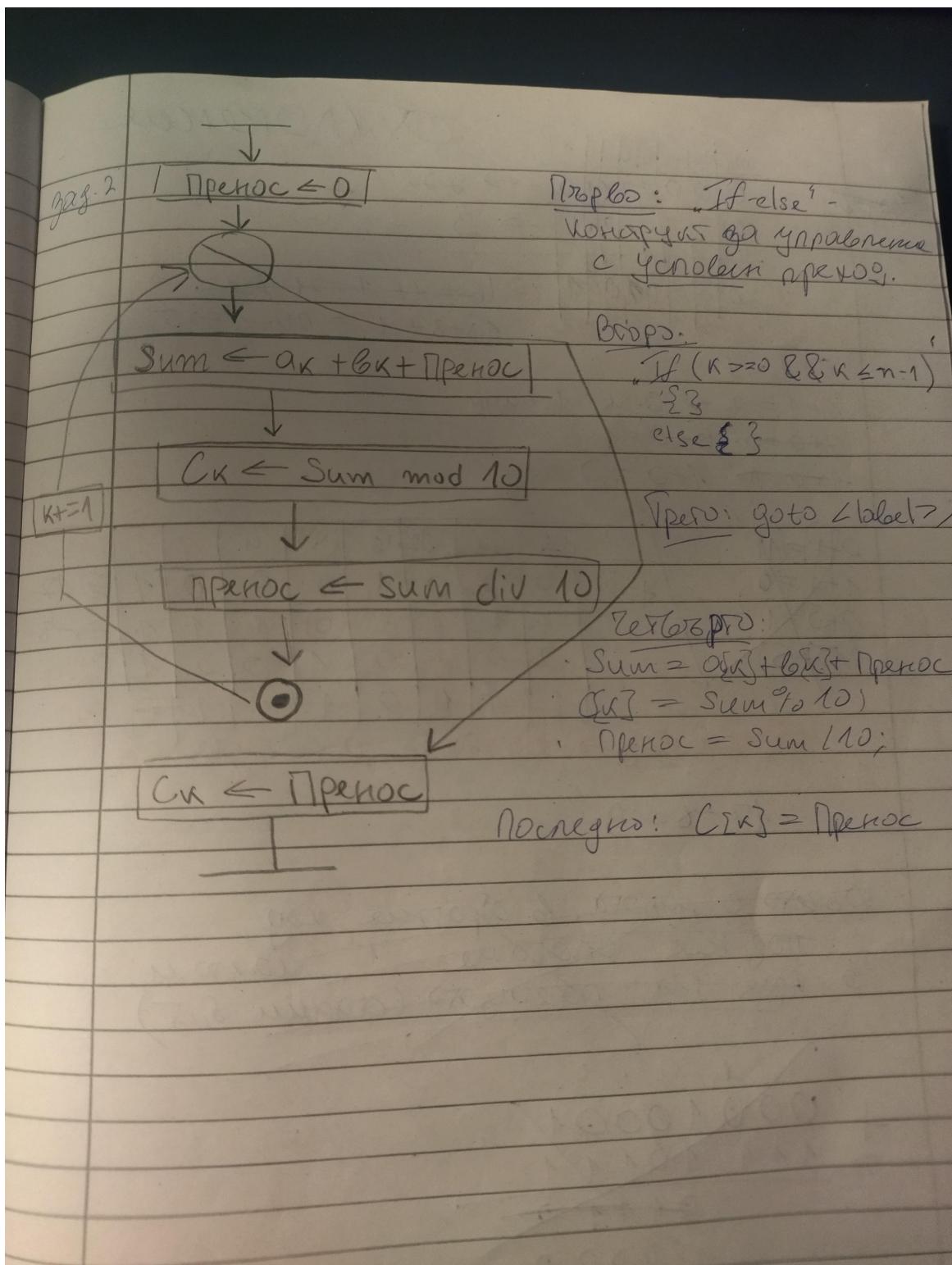


2. Долу е дадена схемата на управление на алгоритъма за събиране в позиционна бройна система, както е обяснена и трасирана с пример в час. Направете схемата на ръка и встрани, успоредно на нея, съставете съответен програмен текст.



Фигура 8: Множество на целите числа, цикли по брояч. Натрупване на суми и произведения

3.2 ОТГОВОР



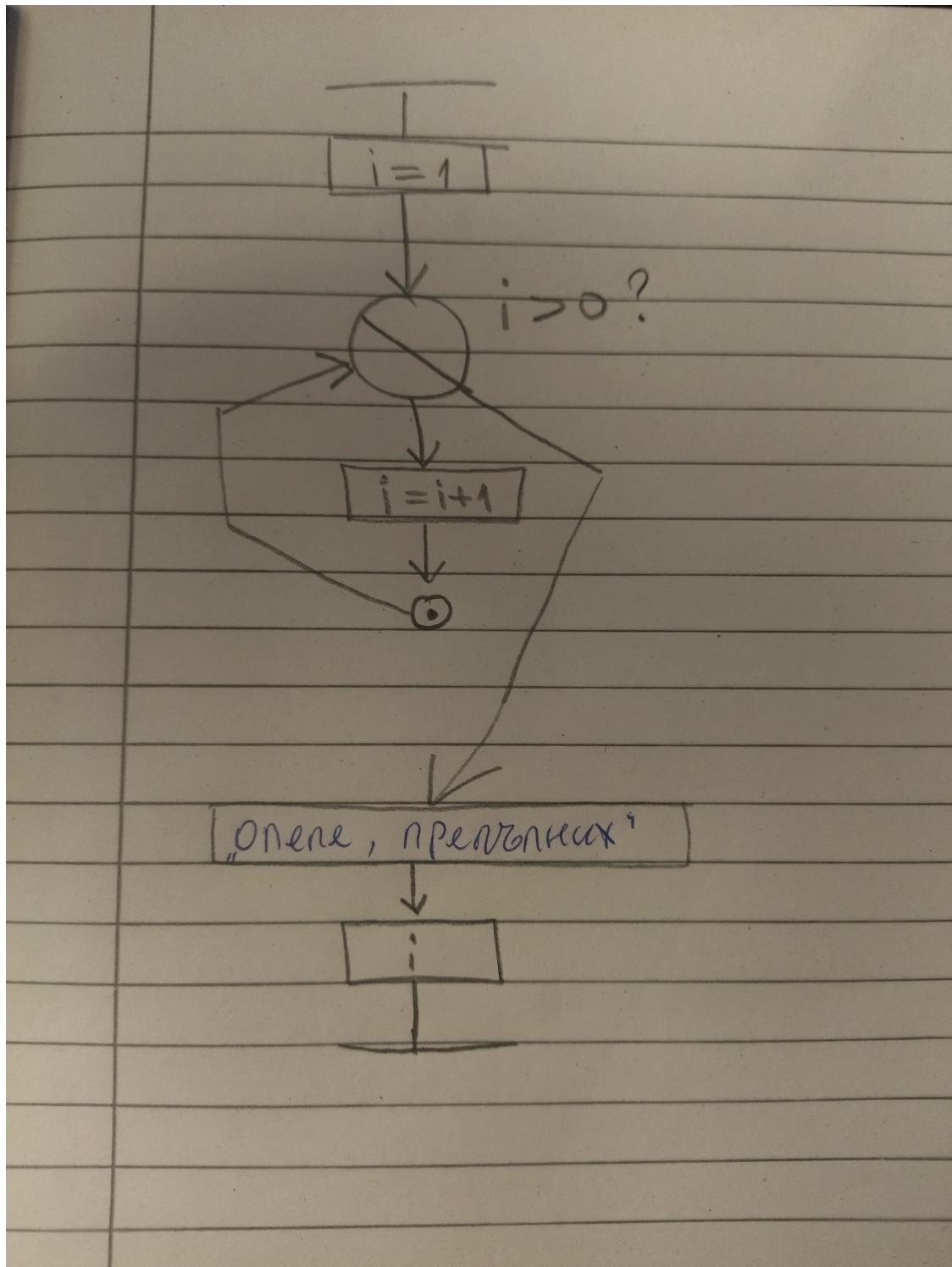
```
1 #include <iostream>
2
3 int main(){
4     std::cout << "HOMEWORK SOLUTION 3" << std::endl;
5     const int N = 5;
6
7
8     int a[N] = { 3, 8, 6, 5 };
9     int b[N] = { 4, 8, 7, 9 };
10
11    int c[N];
12
13    int carry = 0;
14    int sum;
15
16    int k = N-1;
17
18 start:
19    if(k >= 0){
20        std::cout << "\nk: " << k << std::endl;
21        std::cout << "-----" << std::endl;
22
23        sum = a[k] + b[k] + carry;
24        c[k] = sum % 10;
25        carry = sum / 10;
26
27        std::cout
28            << "a: " << a[k]
29            << "\nb: " << b[k]
30            << "\nsum: " << sum
31            << "\nc[" << k << "]": " << c[k]
32            << "\ncarry: " << carry <<
33        std::endl;
34
35
36        k--;
37        goto start;
38
39    } else {
40        goto end;
41    }
42
43 end:
44    std::cout << std::endl;
45    for(int i = 0; i < N; i++)
46        std::cout << c[i];
47    return 0;
48
49    return 1;
50 }
```

3. Препълване на битовете за цяло число.

Съставете:

- (a) Схема на управление

3.3 ОТГОВОР

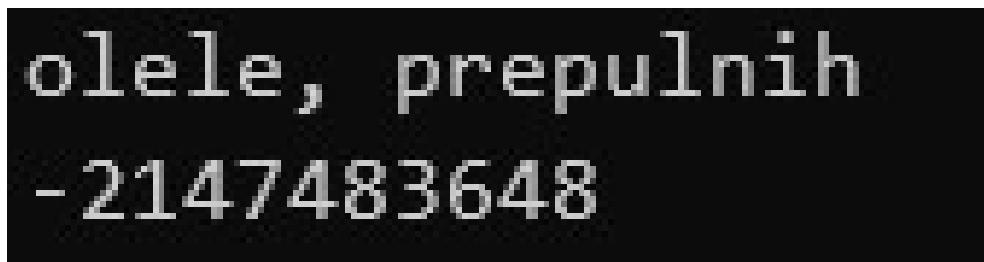


- (6) Успоредно на нея - програма за реализиране на цикъл, като ползвате единствено **условен преход и безусловен преход**. Това в езиците, които ползвате, става например с If условие, тяло на Ifa при Да, и go to етикета на Ifa.

Вътре в тялото на цикъла увеличавайте с единица една променлива i - цяло число, като началната ѝ стойност зададете да е едно. Условието на Ifa да е такова, че когато променливата i стане отрицателна, да се излиза от цикъла и да се извежда съобщение – „олеле, препълних“, като се изведе на екран и стойността на i. Пуснете програвата и приложете разпечатка на програмния текст и на изхода.

3.4 ОТГОВОР

```
1 #include <iostream>
2
3 int main(){
4     int i = 1;
5
6 start:
7     if(i > 0) {
8         i++;
9         goto start;
10    } else {
11        goto end;
12    }
13
14 end:
15     std::cout << "olele, prepulnih" << std::endl;
16     std::cout << i << std::endl;
17     return 0;
18 }
```



```
olele, prepulnih
-2147483648
```

Фигура 9: изход от изпълнение

ДОМАШНО 4

Краен срок: четвъртък, 28 март 2024, 00:00 AM

Предадено на: неделя, 24 март 2024, 11:37 AM

- Напишете формулата на Гаус за сумата на първите n естествене числа. Под нея разпишете проверка - дали формулата е вярна за сумата на първите n естествени числа, когато n е последната цифра от факултетния ви номер. Ако номерът ви завършва с нула, приемете $n = 6$.

Например така:

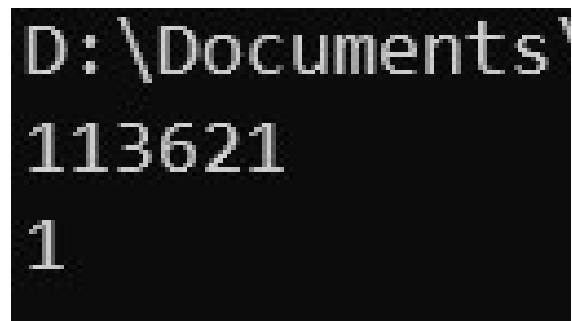
$$\begin{aligned} & F1081174 \\ & n = 4 \\ & 1 + 2 + 3 + 4 = 10 \end{aligned} \tag{1}$$

По формулата на Гаус за сумата на първите n естествени числа, при $n = 4$:

$$\sum_{n=1}^4 i = \frac{(4+1)*4}{2} = \frac{5*4}{2} = 10 \tag{2}$$

4.1 ОТГОВОР

```
1 #include <iostream>
2
3 int main()
4 {
5     char fNumber[6];
6     std::cin >> fNumber;
7     //std::cout << fNumber << std::endl;
8     //std::cout << fNumber[6] << std::endl;
9
10    int n = fNumber[5] - '0';
11    int result = 0;
12    for(int i = 0; i < n; i++){
13
14        result += fNumber[i] - '0';
15    }
16
17    std::cout << result << std::endl;
18
19
20    return 0;
21 }
```



Фигура 10: изход от изпълнение

Заместване на стойностите във формулата на K. Ф. Гаус

$$\sum_{i=1}^n x = \sum_{i=1}^1 \frac{(i+1)*i}{2} = \frac{(1+1)*1}{2} = \frac{2*1}{2} = \frac{2}{2} = 1 \quad (3)$$

2. 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:

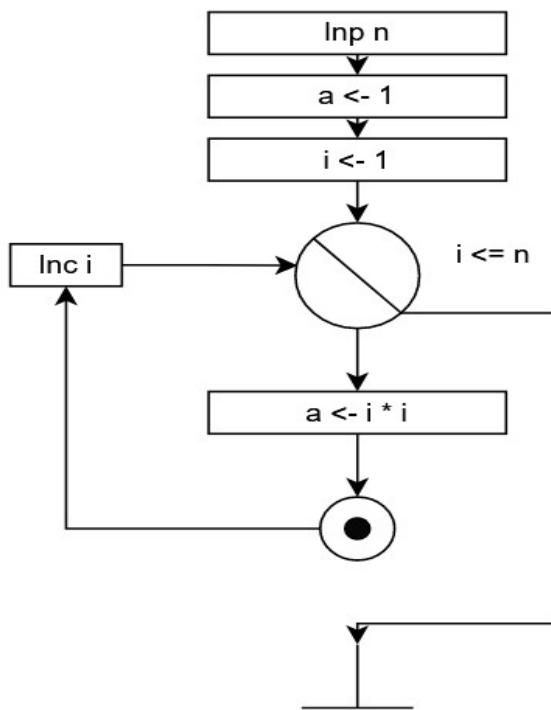
$$\sum_{i=1}^n i^2 \quad (4)$$

4.2 ОТГОВОР

```

1 #include <iostream>
2
3 int main()
4 {
5     int n;
6     std::cin >> n;
7
8     int a = 1;
9     int i = 1;
10
11
12     for(; i <= n; i++){
13
14         a = i * i;
15         std::cout << i << "^2: " << a << std::endl;
16     }
17
18
19     return 0;
20 }
```

2A



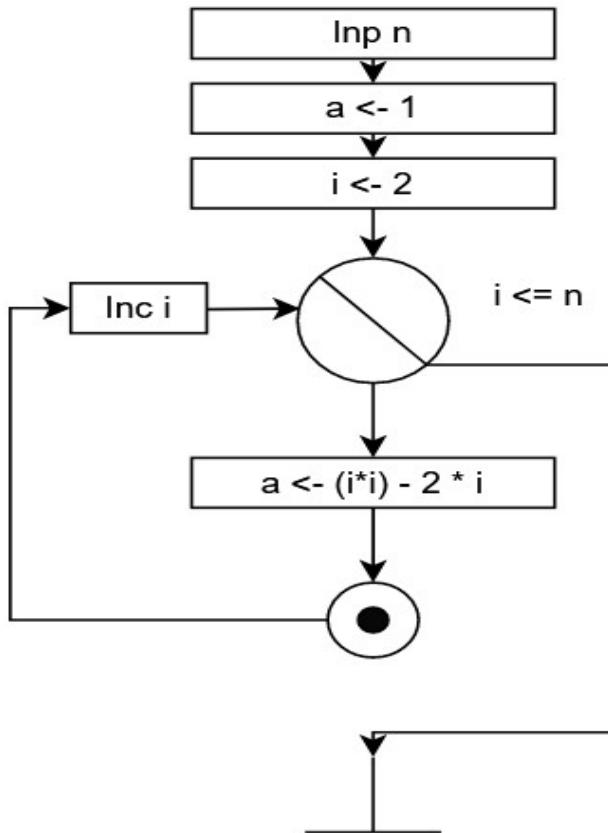
$$\sum_{i=2}^n (i^2 - 2i) \quad (5)$$

4.3 ОТГОВОР

```

1 #include <iostream>
2
3
4
5 int main()
6 {
7     int n;
8     std::cin >> n;
9     int a = 1;
10    int i = 2;
11    for(; i <= n; i++){
12
13        a = (i * i) - 2 * i;
14
15        std::cout << i << " ^2 - 2 * " << i << ":" << a << std::endl;
16    }
17
18    return 0;
19 }
```

2B



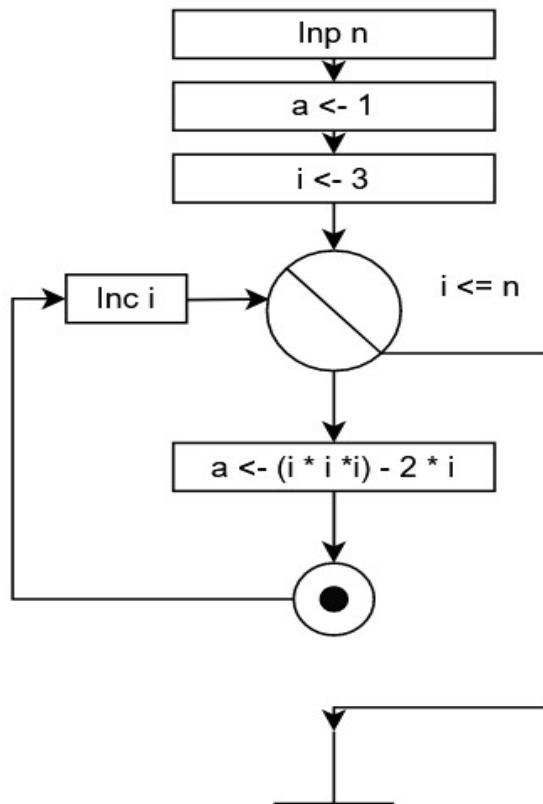
$$\sum_{i=3}^n (i^3 - 2i) \quad (6)$$

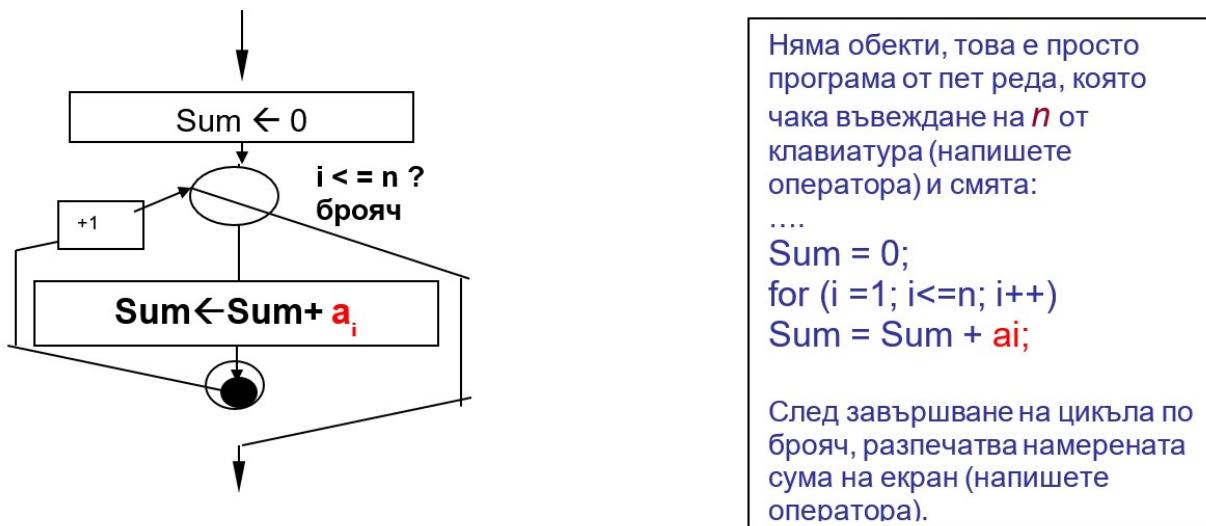
4.4 ОТГОВОР

```

1 #include <iostream>
2
3
4
5 int main()
6 {
7     int n;
8     std::cin >> n;
9     int a = 1;
10    int i = 3;
11    for(; i <= n; i++){
12
13        a = (i * i * i) - 2 * i;
14
15        std::cout << i << " ^3 - 2 * " << i << ":" << a << std::endl;
16    }
17
18    return 0;
19 }
```

2C





Фигура 11: Модел

Напомняме – a_i се изразява с i , т.e. с поредния номер на член на сумата. То е, например, $i * i$.

3. Съставете програма за пресмятане на **факториел** на цяло положително число чрез натрупване на произведение, т.е. с цикъл по брояч. Пуснете я на машина и проверете факториел на кое число довежда вече до препълване на машинното представяне на целити числа.

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 магат да се представят като Сума на ред.

4.5 ОТГОВОР

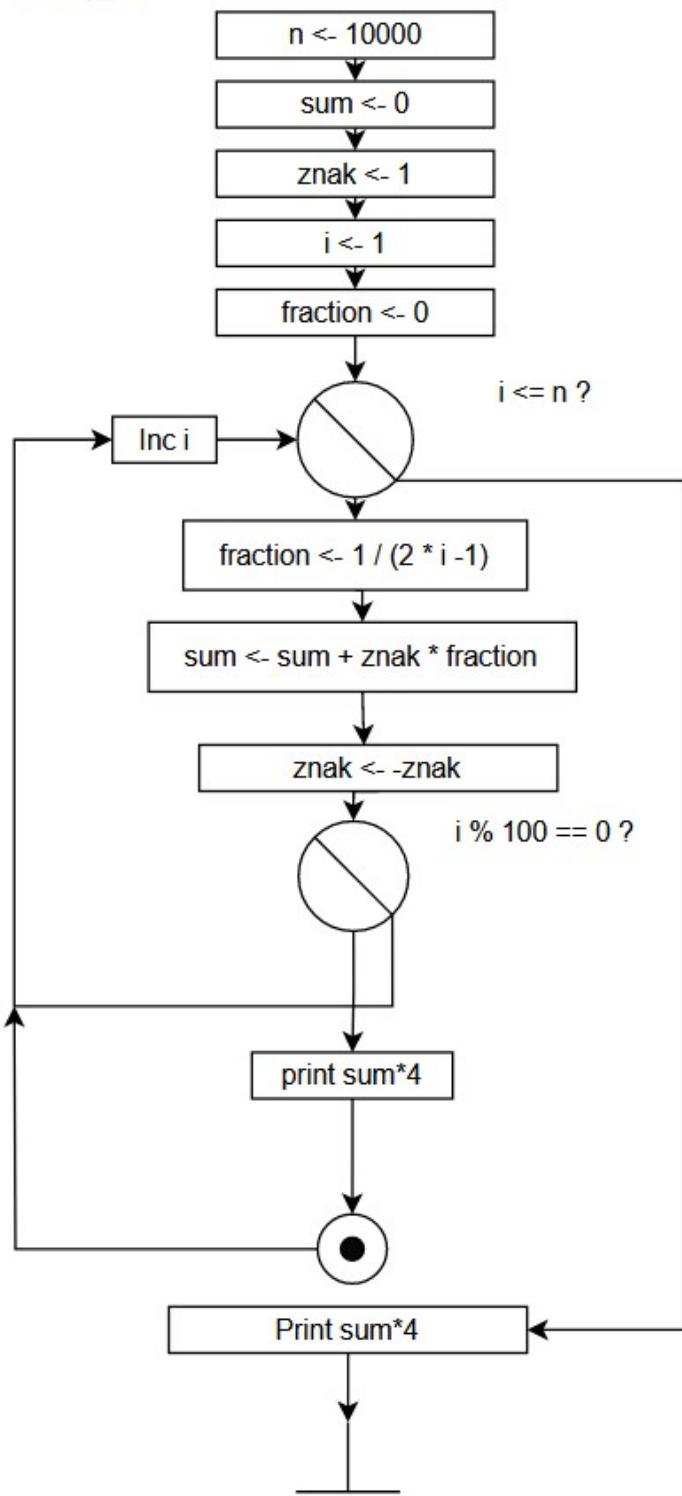
```
1 #include <iostream>
2
3 int main()
4 {
5
6     unsigned int n;
7     std::cin >> n;
8
9
10    int fact = 1;
11    int counter = 1;
12
13    while(counter <= n){
14
15        fact = fact * counter;
16
17        counter++;
18    }
19
20    std::cout << fact << std::endl;
21    return 0;
22 }
```

4. Съставете схемата на управление (има я в пепете) и успоредно на нея – програмата за пресмятане на π с алтернативен ред, като Сумата се пресмята 10 000 пъти, по брояч. Реализирайте програмата с разпечатка на получаваното за π на всеки 100 преминавания през цикъла.

4.6 ОТГОВОР

```
1 #include <iostream>
2
3
4 int main()
5 {
6     int n = 10000;
7
8     float sum = 0;
9     int znak = 1;
10    int i = 1;
11
12    float fraction = 0;
13
14    while(i <= n){
15        fraction = (float)1 / (float)(2*i - 1);
16        sum = sum + znak * fraction;
17        znak = -znak;
18        if(i % 100 == 0)
19            std::cout << i << ":" << sum * 4 << std::endl;
20        i++;
21    }
22    std::cout << "Final: " << sum * 4;
23    return 0;
24 }
```

calc_Pi



Фигура 12: опорна схема

ДОМАШНО 5

Краен срок: *сряда, 3 април 2024, 09:00 AM*

Предадено на: *понеделник, 1 април 2024, 17:16 PM*

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

- Сумата се пресмята 10 000 пъти, по брояч – от миналото домашно 2024.

5.1 ОТГОВОР

```
1 #include <iostream>
2
3
4 int main()
5 {
6     int n = 10000;
7
8     float sum = 0;
9     int znak = 1;
10    int i = 1;
11
12    float fraction = 0;
13
14    while(i <= n){
15        fraction = (float)1 / (float)(2*i - 1);
16        sum = sum + znak * fraction;
17        znak = -znak;
18        if(i % 100 == 0)
19            std::cout << i << ":" << sum * 4 << std::endl;
20        i++;
21    }
22    std::cout << "Final: " << sum * 4;
23    return 0;
24 }
```

- Сумата се пресмята ДОТОГАВА, ДОКАТО разликата между две стойности, получени последователно, се получи по-малка от 0.0001. **Съставете програма** в съответствие с тази схема.
Реализирайте програмата по два начина - с използване на вграден езиков оператор за цикъл и с използване на условен и безусловен преход.

5.2 ОТГОВОР

```

1 #include <iostream>
2 #include <cmath> // fabs()!
3
4
5 int main()
6 {
7
8     float precision = 0.0001;
9
10    float sum = 0;
11    float sum_prev = 0;
12    float diff = 1;
13
14    int znak = 1;
15
16    float fraction = 0;
17
18
19    int i = 1;
20 start:
21     if(fabs(diff) > precision){
22         sum_prev = sum;
23         fraction = (float)1 / (float)(2*i - 1);
24         sum = sum + znak * fraction;
25         znak = -znak;
26
27         diff = sum - sum_prev;
28
29         //std::cout << fabs(diff) << std::endl;
30         //std::cout << sum * 4 << std::endl;
31         //std::cout << sum_prev * 4 << std::endl;
32         //std::cout << std::endl;
33         //std::cout << std::endl;
34         //system("pause");0
35
36         i++;
37         goto start;
38     } else {
39         goto theend;
40     }
41
42
43 theend:
44     std::cout << sum_prev * 4 << std::endl;

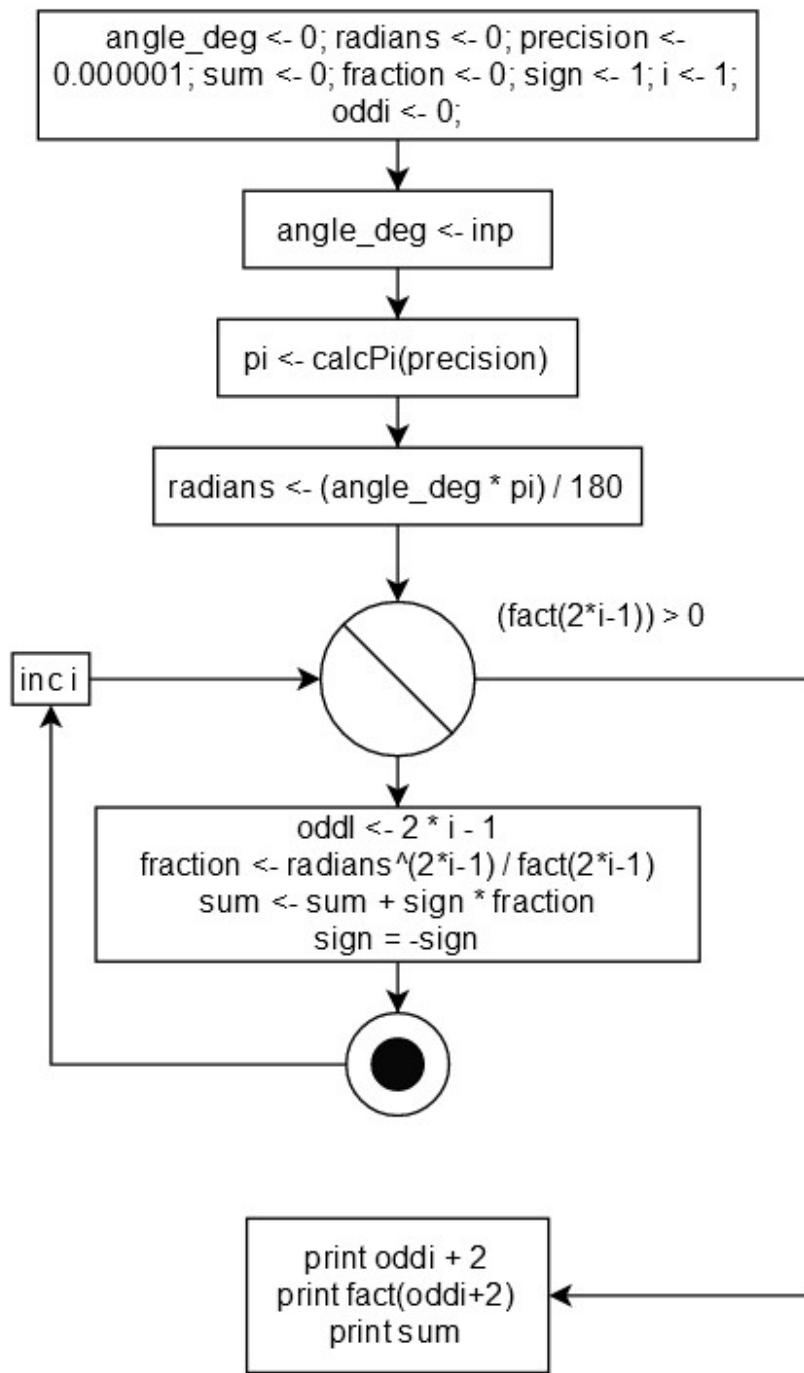
```

```

45
46
47     return 0;
48 }

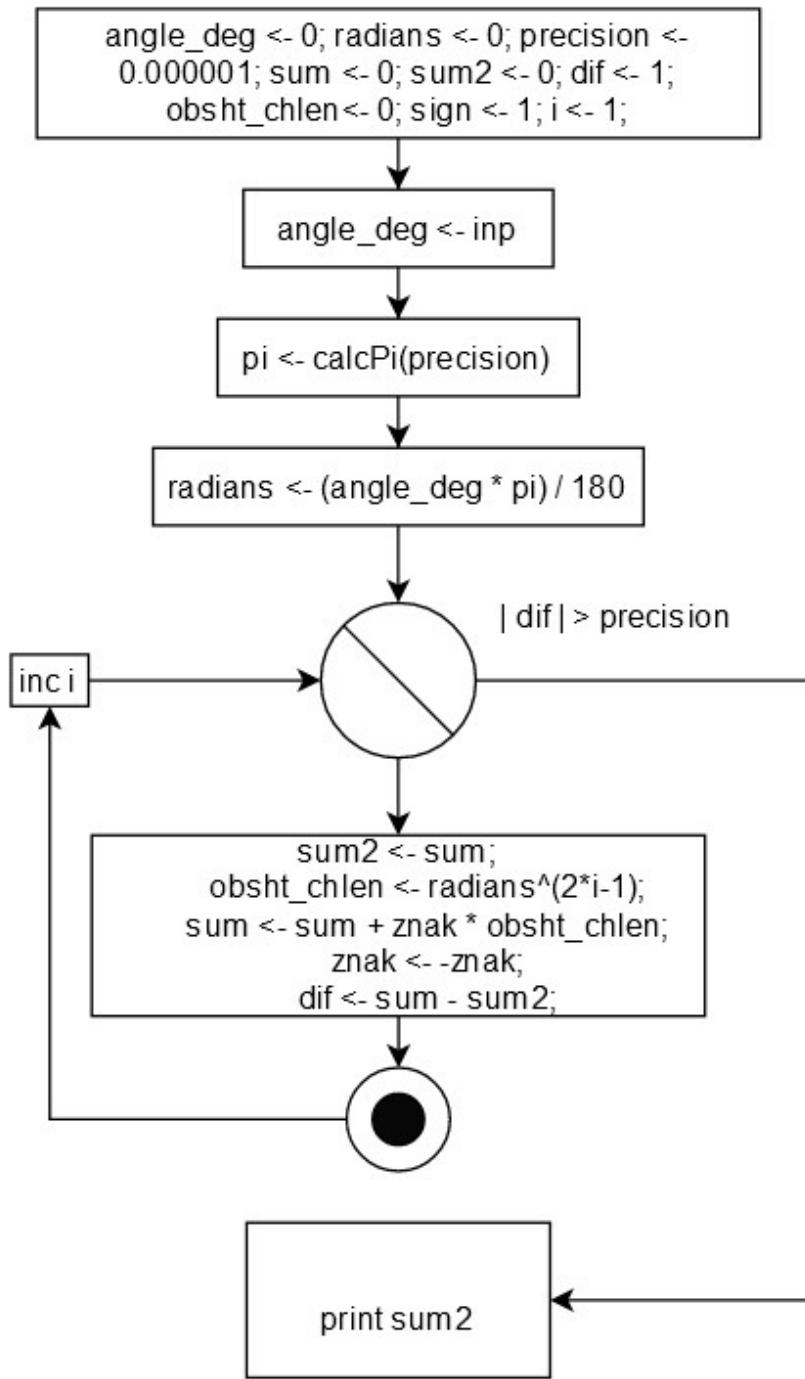
```

Първи начин



Фигура 13: опорна схема

Втори вариант



Фигура 14: опорна схема

2. Да се състави алгоритъм (схемата на управление) и **програма** (в съответствие със схемата) за пресмятане **по два начина** на функцията $\sin(x)$, с итеративен цикъл, както е дадена на семинар (развита в ред на Маклорен).

- Първи начин – с пресмянате на факториел (в знаменателя) до получаване на член с отрицателен знаменател

5.3 ОТГОВОР

```
1
2
3 #include <iostream>
4 #include <cmath>
5
6
7 float calcPi(float prec){
8
9
10    //std::cout << "calcPi()" << std::endl;
11    //std::cout << prec << std::endl;
12
13    float fraction = 0;
14    float sum = 0;
15    float sum_prev = 0;
16
17    int sign = 1;
18
19    float diff = 1;
20
21    int i = 1;
22    while(fabs(diff) > prec){
23        sum_prev = sum;
24        fraction = (float)1 / (float)(2*i - 1);
25        sum = sum + sign * fraction;
26        sign = -sign;
27
28        diff = sum - sum_prev;
29
30        //std::cout << fabs(diff) << std::endl;
31        //std::cout << sum * 4 << std::endl;
32        //std::cout << sum_prev * 4 << std::endl;
33        //std::cout << std::endl;
34        //std::cout << std::endl;
35        //system("pause");0
36
37        i++;
38    }
39
40
41    sum_prev*=4;
42
43    return sum_prev;
44 }
```

```

45
46
47 int fact(int n){
48
49
50     int fact = 1;
51
52     for(int i = 1; i < n+1; i++){
53         fact = fact * i;
54     }
55
56     return fact;
57 }
58
59
60 int main()
61 {
62     double angle_deg = 0; // degrees
63     double radians = 0;
64     float precision = 0.000001;
65
66     float sum = 0;
67     float fraction = 0;
68     int sign = 1;
69     int i = 1;
70
71     int oddI = 0;
72
73     std::cout << "Enter an angle (degrees): ";
74     std::cin >> angle_deg;
75
76
77
78     float pi = calcPi(precision);
79     std::cout << "pi: " << pi << std::endl;
80
81     radians = (angle_deg * pi) / 180;
82     std::cout << "radians: " << radians << std::endl;
83
84     while((fact(2*i-1)) > 0){
85         oddI = 2 * i - 1;
86         std::cout << 2*i-1 << ":" << fact(2*i-1) << std::endl;
87         fraction = pow(radians, 2*i-1) / fact(2*i-1);
88         sum = sum + sign * fraction;
89         sign = -sign;
90         i++;
91     }
92
93     std::cout << "type overflow at i = " << oddI+2 << ":" << fact(oddI+2) <
94
95     std::cout << "sin (" << angle_deg << " deg): " << sum << std::endl;
96
97

```

```
98  
99  
100  
101     return 0;  
102 }
```

```

D:\Documents\Personel\laptop\myDoc\laptop\algorithms\march-26-24\hw5\2\bin\Debug\2.exe
Enter an angle (degrees): 45
pi: 3.14159
radians: 0.785398
1: 1
3: 6
5: 120
7: 5040
9: 362880
11: 39916800
13: 1932053504
15: 2004310016
type overflow at i = 17: -288522240
sin (45 deg): 0.707107
Process returned 0 (0x0)   execution time : 2.871 s
Press any key to continue.

```

Фигура 15: 45 градуса

The image shows two screenshots from a web browser. The top screenshot displays search results for "45 degrees radians". It includes a search bar with the query, navigation links for All, Images, Videos, News, and Maps, and filters for Always private, Bulgaria, Safe search, and Any time. Below the search bar are two boxes: one containing "45" with dropdown menus for Degrees and Radians. The bottom screenshot shows a calculator interface with a numeric keypad and function keys like sin, cos, tan, etc. The calculator displays the result of sin(45 deg) as 0.70710678119.

Фигура 16: проверяваме получените стойности

- втори начин – с точност 0.000001 без пресмятане на факториел.

5.4 ОТГОВОР

```

1
2
3 #include <iostream>
4 #include <cmath>
5
6
7 float calcPi(float prec){
8
9
10    //std::cout << "calcPi()" << std::endl;
11    //std::cout << prec << std::endl;
12
13    float fraction = 0;
14    float sum = 0;
15    float sum_prev = 0;
16
17    int znak = 1;
18
19    float diff = 1;
20
21    int i = 1;
22    while(fabs(diff) > prec){
23        sum_prev = sum;
24        fraction = (float)1 / (float)(2*i - 1);
25        sum = sum + znak * fraction;
26        znak = -znak;
27
28        diff = sum - sum_prev;
29
30        //std::cout << fabs(diff) << std::endl;
31        //std::cout << sum * 4 << std::endl;
32        //std::cout << sum_prev * 4 << std::endl;
33        //std::cout << std::endl;
34        //std::cout << std::endl;
35        //system("pause");0
36
37        i++;
38    }
39
40
41    sum_prev*=4;
42
43    return sum_prev;
44 }
45
46
47 ///no fact
48
49 int main()

```

```

50  {
51      std::cout << "NO FACT CALC!" << std::endl << std::endl;
52      double angle_deg = 0; // degrees
53      double radians = 0;
54      float precision = 0.000001;
55
56      float sum = 0;
57      float sum2 = 0;
58      float dif = 1;
59
60      float obsht_chlen = 0;
61      int znak = 1;
62      int i = 1;
63
64      std::cout << "Enter an angle (degrees): ";
65      std::cin >> angle_deg;
66
67      float pi = calcPi(precision);
68      std::cout << "pi: " << pi << std::endl;
69
70      // 1 degree = pi / 180 radians
71
72
73      radians = (angle_deg * pi) / 180;
74      std::cout << "radians: " << radians << std::endl;
75
76      while(fabs(dif) > precision){
77          //std::cout << std::fixed << fabs(dif) << std::endl;
78
79          sum2 = sum;
80          obsht_chlen = pow(radians, 2*i-1);
81          sum = sum + znak * obsht_chlen;
82          znak = -znak;
83          dif = sum - sum2;
84          i++;
85      }
86      std::cout << "closest value to our precision: [" << fabs(dif) << "], fix
87      std::cout << "sin (" << angle_deg << " deg): " << sum2 << std::endl;
88
89      return 0;
90  }

```

```

D:\Documents\Personal\laptop\myDocs\laptop\algorithms\march-26-24\hw5\22\bin\Debug\22.exe
NO FACT CALC!

Enter an angle (degrees): 30
pi: 3.14159
radians: 0.523599
closest value to our precision: [3.57628e-07], fixed:(0.000000)
sin (30.000000 deg): 0.410938

Process returned 0 (0x0)   execution time : 3.374 s
Press any key to continue.

```

Фигура 17: 30 градуса

The screenshot shows a search results page for "30 deg rad". The top bar includes a logo, a search input field containing "30 deg rad", and a magnifying glass icon. Below the search bar are navigation links: All, Images, Videos, News, Maps, Chat, and Settings. Filter options include "Always private", "Bulgaria", "Safe search: moderate", and "Any time". The main search result displays two boxes: one containing "30" and another containing "0.5235988". Below these boxes are dropdown menus for "Degrees" and "Radians". A "Share Feedback" link is located at the bottom right of the result card.

The screenshot shows a calculator application interface. At the top, there is a search bar with the query "sin 30 degrees" and a magnifying glass icon. Below the search bar are navigation links: All, Images, Videos, News, Maps, Calculator (which is selected), Chat, and Settings. The main area features a numeric keypad with a yellow-highlighted division key (÷). To the right of the keypad, the result "sin(30 deg)" is shown above the value "0.5". The calculator interface includes standard buttons for trigonometric functions (sin, cos, tan), constants (π, e), and mathematical operations (+, -, ×, ÷).

Фигура 18: проверка на получените стойности

```
D:\Documents\Personal\laptop\myDocs\laptop\algorithms\march-26-24\hw5\22\bin\Debug\22.exe
NO FACT CALC!

Enter an angle (degrees): 15
pi: 3.14159
radians: 0.261799
closest value to our precision: [4.02331e-07], fixed:(0.000000)
sin (15.000000 deg): 0.245007

Process returned 0 (0x0)   execution time : 0.980 s
Press any key to continue.
```

Фигура 19: 15 градуса

The figure consists of two screenshots from the DuckDuckGo search engine. The top screenshot shows the search bar with '15 deg rad' and the results page with a calculator widget. The bottom screenshot shows the search bar with 'sin 15' and the results page with a calculator interface.

Top Screenshot (Search for '15 deg rad'):

- Search bar: 15 deg rad
- Results:
 - All Images Videos News Maps
 - Always private (checked)
 - Bulgaria (selected)
 - Safe search: moderate
 - Any time
- Calculator Widget:
 - Input: 15
 - Output: 0.2617994
 - Conversion: Degrees ↔ Radians
- Share Feedback link

Bottom Screenshot (Search for 'sin 15'):

- Search bar: sin 15
- Results:
 - All Images Videos News Maps
 - Calculator (selected)
 - Chat Settings
- Calculator Interface:
 - Display: sin(15)
0.65028784016
 - Buttons:
 - RAD DEG
 - ()
 - C % ÷
 - sin cos tan π 7 8 9 ×
 - x! x² x³ x^y 4 5 6 -
 - 1/x √x x√y EE 1 2 3 +
 - log ln e^x e 0 . =

```

D:\Documents\Personel\laptop\myDocs\laptop\algorithms\march-26-24\hw5\22\bin\Debug\22.exe
NO FACT CALC!

Enter an angle (degrees): 5
pi: 3.14159
radians: 0.0872665
closest value to our precision: [3.72529e-08], fixed:(0.000000)
sin (5.000000 deg): 0.086607

Process returned 0 (0x0)   execution time : 1.507 s
Press any key to continue.

```

Фигура 20: 5 градуса

The screenshot shows a search results page from a web browser. The search query is "5 deg rad". The results include a snippet from a calculator application and a direct link to a calculator interface.

Calculator Snippet:

- Input: 5
- Output: 0.08726646
- Conversion: Degrees ↔ Radians

Calculator Interface:

- Search bar: sin 5 deg
- Calculator buttons:
 - RAD (radio button)
 - DEG (radio button)
 - ()
 - C
 - %
 - ÷
 - sin
 - cos
 - tan
 - π
 - 7
 - 8
 - 9
 - x*
 - x²
 - x³
 - x^y
 - EE
 - 1
 - 2
 - 3
 -
 - 1/x
 - √x
 - x√y
 - EE
 - 0
 - .
 - =
- Output: sin(5 deg)
0.08715574275
- Output (highlighted): 0.08715574275

```
D:\Documents\Personal\laptop\myDocs\laptop\algorithms\march-26-24\hw5\22\bin\Debug\22.exe
NO FACT CALC!
Enter an angle (degrees): 1
pi: 3.14159
radians: 0.0174533
closest value to our precision: [1.86265e-09], fixed:(0.000000)
sin (1.000000 deg): 0.017448

Process returned 0 (0x0)   execution time : 1.230 s
Press any key to continue.
```

Фигура 21: 1 градус

The image shows two screenshots side-by-side. The top screenshot is a command-line application window titled 'NO FACT CALC!' with the following output:

```
D:\Documents\Personal\laptop\myDocs\laptop\algorithms\march-26-24\hw5\22\bin\Debug\22.exe
NO FACT CALC!
Enter an angle (degrees): 1
pi: 3.14159
radians: 0.0174533
closest value to our precision: [1.86265e-09], fixed:(0.000000)
sin (1.000000 deg): 0.017448

Process returned 0 (0x0)   execution time : 1.230 s
Press any key to continue.
```

The bottom screenshot is a search results page from a web browser. The search query '1 deg rad' has been entered. The results show the conversion of 1 degree to radians (0.01745329). The calculator interface below the search results shows the input '1' and the output '0.01745329' with units set to Degrees and Radians respectively. A yellow box highlights the result '0.01745329'.

Извод: Когато изпълняваме метода без пресмятане на факториел, губим от точността в изчисленията. Това прави първият начин по-надежден спрямо втория.

За двета начина съставете код.

Обърнете внимание, че аргументът е ъгъл в радиани. Пуснете вход от клавиатура - ъгъл в градуси, който програмата превръща в радиани.

За първия начин, проследете до кой член сумата се пресмята без препълване и направете програмата да извежда на еcran коментар, когато става препълването.

За втория начин пуснете програмата с различни входове – 30 градуса, 15 градуса, 5 градуса и 1 градус. Поставете в итеративния цикъл брояч и извеждайте стойността на брояча, при която се е достигнала исканата точност.

Приложете към домашното код и снимки на еcran с входа и изхода.

ДОМАШНО 6

Краен срок: *петък, 12 април 2024, 00:00 AM*

Предадено на: *понеделник, 8 април 2024, 16:07 PM*

- Съставете схема на управление и успоредно на нея – програмен текст на претърсване на масив чрез **обхождане** на целия масив. Изведете броя проверки “ти хикс ли си” на еcran. Пуснете програмата над масив от числа - цифрите на факултетния ви номер (зети като числа), като търсите последователно дали стойностите 1, 2, 3 и 4 се намират вътре или не. Предавате програмен текст, вход и изход – снимка на еcran.

6.1 ОТГОВОР

```
1 #include <iostream>
2
3 // F113621
4
5
6 int main()
7 {
8     int a[6] = { 1, 1, 3, 6, 2, 1 };
9     int b[4] = { 1, 2, 3, 4 };
10    int checks = 0;
11    bool state = false;
12    std::string message = "";
13
14    while(checks != 4){ // size of second array
15
16        for(int i = 0; i < 6; i++){ // size of first array
17            std::cout << "ti (" << b[checks] << ") " << a[i] << " li si?" << std::endl;
18            if(b[checks] == a[i]){
19                state = true;
20                std::cout << "-> da" << std::endl;
21                break;
22            } else {
23                std::cout << "-> ne" << std::endl;
24            }
25        }
26        if(state)
27            message = "Ima go!\n";
28        else
29            message = "Ne go namerih.\n";
30        std::cout << b[checks] << ":" " << message << std::endl;
31        checks++;
32
33        state = false;//reset
34    }
35
36    return 0;
37 }
```

```

ti (1) 1 li si?
-> da
1: Ima go!

ti (2) 1 li si?
-> ne
ti (2) 1 li si?
-> ne
ti (2) 3 li si?
-> ne
ti (2) 6 li si?
-> ne
ti (2) 2 li si?
-> da
2: Ima go!

ti (3) 1 li si?
-> ne
ti (3) 1 li si?
-> ne
ti (3) 3 li si?
-> da
3: Ima go!

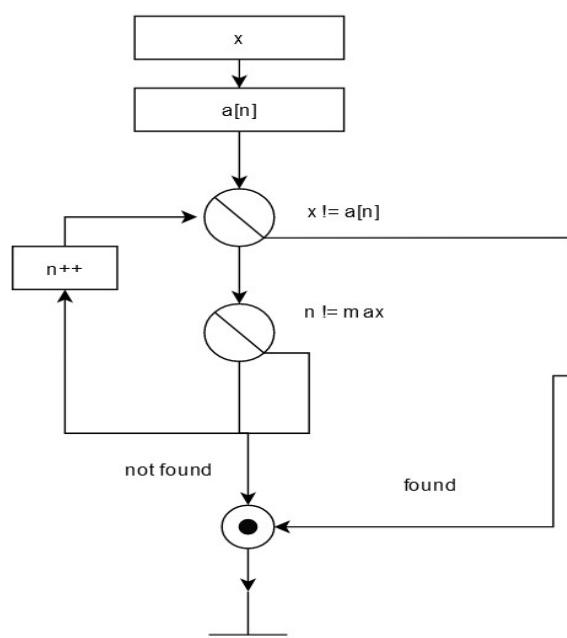
ti (4) 1 li si?
-> ne
ti (4) 1 li si?
-> ne
ti (4) 3 li si?
-> ne
ti (4) 6 li si?
-> ne
ti (4) 2 li si?
-> ne
ti (4) 1 li si?
-> ne
4: Ne go namerih.

```

```

Process returned 0 (0x0)   execution time : 0.038 s
Press any key to continue.
-
```

Линейно търсене



2. Съставете схема на управление и успоредно на нея – програмен текст на търсене в масив **дотогава, докато** се намери търсената стойност, при използване на „**котва**“. Изведете броя проверки “ти хикс ли си” на еcran. Пуснете програмата над масив от букви – буквите от фамилното ви име, като търсите последователно дали стойностите а, б, в и г се намират вътре или не. Предавате програмен текст, вход и изход – снимка на еcran.

6.2 ОТГОВОР

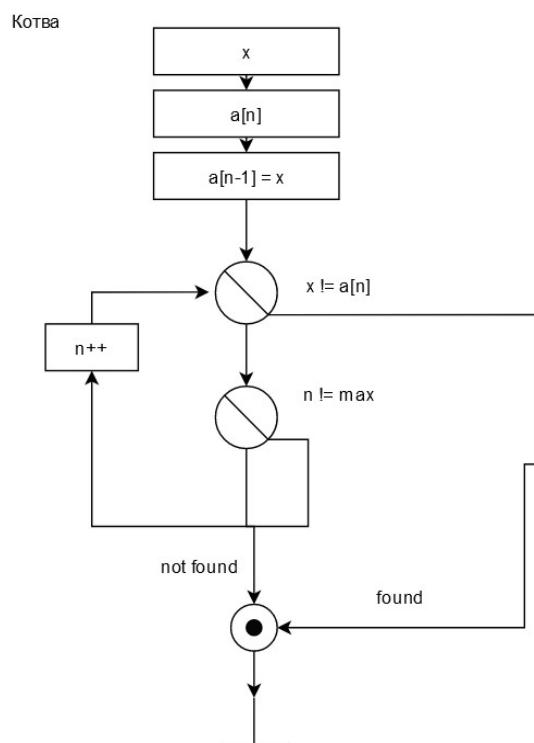
```

1 //include <iostream>
2 #include <windows.h>
3 int main()
4 {
5     SetConsoleOutputCP(1251);
6
7     char a[9] = { 'M', 'a', 'n', 'r', 'b', 'p', 'o', 'b', 'O' };
8     char b[4] = { 'a', 'b', 'v', 'g' };
9
10    int index;
11
12    for(int j = 0; j < 4; j++) {
13        a[0] = b[j];
14        index = 0;
15
16        while (a[index] != b[j]) {
17            std::cout << "ти (" << a[index] << ") \n" << b[j] << "\n" << "ти ли си?: не!";
18            if(a[index] == b[j])
19                std::cout << "да!" << std::endl;
20            else
21                std::cout << "не!" << std::endl;
22
23            index++;
24        }
25
26
27        if (index == 8) // if the index has reached the end (not found)
28            std::cout << "-> \n" << a[index] << "\n" << "не!!" << std::endl;
29        else
30            std::cout << "-> \n" << a[index] << "\n" << "да!!" << std::endl;
31
32    }
33
34
35
36
37
38    return 0;
39
40

```

ти (M) "a" ли си?: не!
-> "a" го има!!!
ти (M) "б" ли си?: не!
ти (a) "б" ли си?: не!
ти (н) "б" ли си?: не!
ти (г) "б" ли си?: не!
ти (ъ) "б" ли си?: не!
ти (р) "б" ли си?: не!
ти (о) "б" ли си?: не!
ти (в) "б" ли си?: не!
-> "б" го няма!!!
ти (M) "в" ли си?: не!
ти (a) "в" ли си?: не!
ти (н) "в" ли си?: не!
ти (г) "в" ли си?: не!
ти (ъ) "в" ли си?: не!
ти (р) "в" ли си?: не!
ти (о) "в" ли си?: не!
-> "в" го има!!!
ти (M) "г" ли си?: не!
ти (a) "г" ли си?: не!
ти (н) "г" ли си?: не!
-> "г" го има!!!

Process returned 0 (0x0) execution time : 0.040 s
Press any key to continue.



3. Съставете

- (а) Опорна схема (това значи схема на масив, означени променливи за начало и край на претърсваната част, среда, местене на индексите - със стрелки);
- (б) схема на управление която да съответства на опорната схема и успоредно на нея – програмен текст на дихотомично претърсване на масив. Материалът е даден подробно с препете в лекциите. Изведете на еcran как се менят стойностите на променливите съхраняващи индексите за начало и край на подмасива, в който търсенето продължава И на броя проверки “ти хикс ли си”. Пуснете програмата над масив от букви и цифри образуван така – факултетният ви номер и фамилното ви име, залепени, като търсите последователно дали стойностите 9, 8, 7 и я, ю, ъ и о се намират вътре или не. Не забравяйте, че методът работи САМО при наредени данни. Наредете ги първо, например на ръка... Предавате програмен текст, вход и изход – снимка на еcran.

6.3 ОТГОВОР

```

#include <iostream>
#include <windows.h>

int main()
{
    // f113621манъров
    SetConsoleOutputCP(1251);
    SetConsoleCP(1251);
    char x;
    std::cout << "Моля въведете търсеният от вас символ: ";
    std::cin >> x;
    int l, mid, r;
    // символи са буквите от кирилицата и малките и големите латински букви
    // захолзванието на всички са залити на всички символи
    char a[13] = { 'а', 'в', 'г', 'м', 'н', 'т', 'р', 'ь', 'и', 'е', 'с', 'з', 'ф' };
    l = 0;
    r = 12; //n-1
    mid = (l+r) / 2;
    while(a[mid] != x && r >= l){
        for(int i = l; i <= r; i++){
            std::cout << a[i] << " ";
        }
        std::cout << std::endl;
        char leftchar = a[l];
        char midchar = a[mid];
        char rightchar = a[r];
        std::cout << "ляв край: \\" << leftchar
        << "\\\" среда: \\" << midchar
        << "\\\" десен край: \\" << rightchar << "\\"
        << std::endl << std::endl;
        if(a[mid] < x)
            l = mid + 1;
        else
            r = mid - 1;
        mid = (l + r) / 2;
    }
    if(l - r == 1){ // ако различавате в 1, не можем да сложим mid. тогава излизаме от цикъла.
        std::cout << "Нямаме среда! Излизам..." << std::endl;
        break;
    }
    if(r >= 1)
        std::cout << "Търсеният символ присъства в множеството!" << std::endl;
    else
        std::cout << "Търсеният символ не беше намерен в множеството." << std::endl;
    return 0;
}

```

Моля въведете търсеният от вас символ: 3
 а в г м н о р ъ 1 2 3 6 f
 ляв край: "а"
 среда: "р"
 десен край: "f"
 ъ 1 2 3 6 f
 ляв край: "ъ"
 среда: "2"
 десен край: "f"
 3 6 f
 ляв край: "3"
 среда: "6"
 десен край: "f"
 Търсеният символ присъства в множеството!
 Process returned 0 (0x0) execution time : 0.861 s
 Press any key to continue.

```

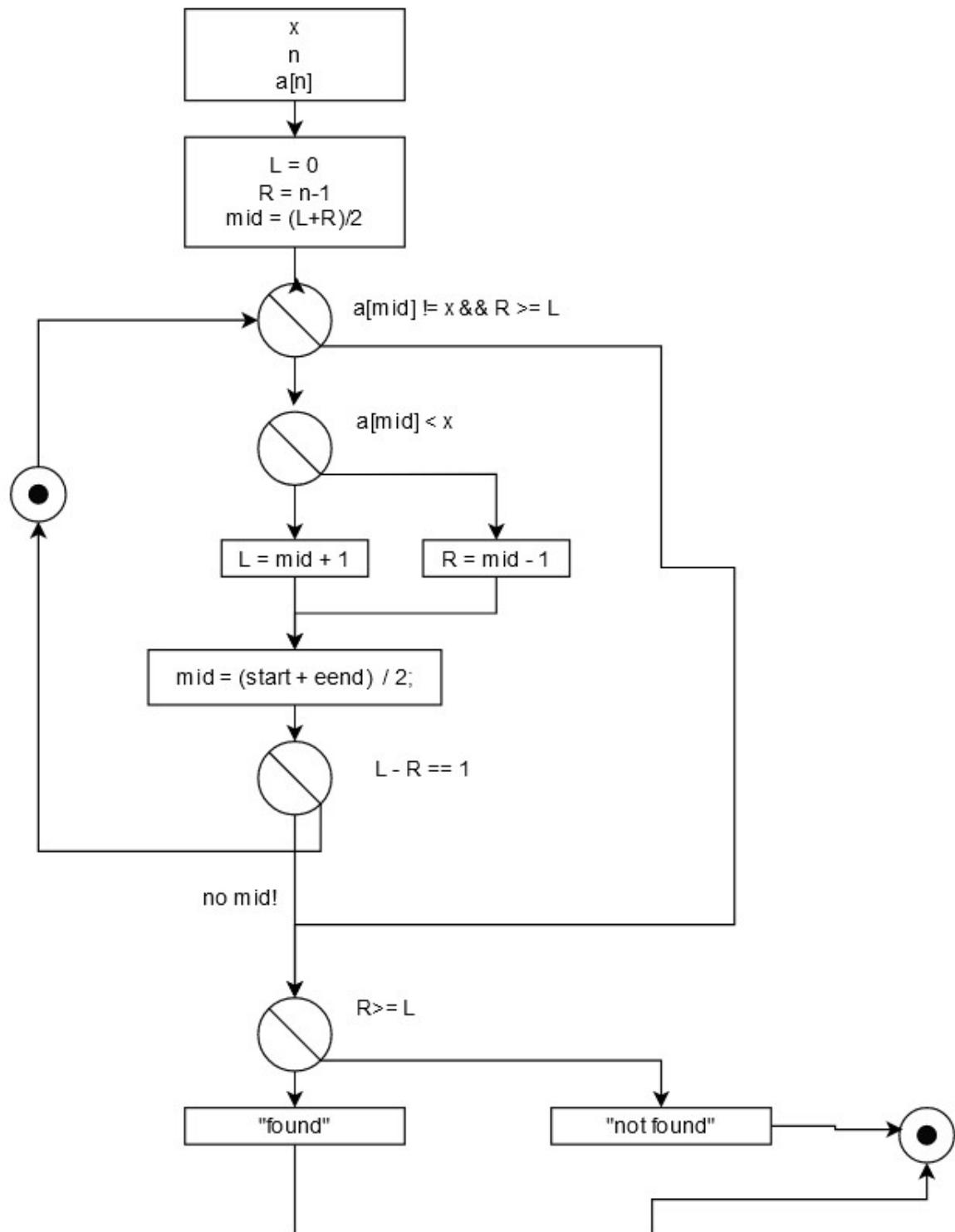
#include <iostream>
#include <windows.h>

int main()
{
    // f113621манъров
    SetConsoleOutputCP(1251);
    SetConsoleCP(1251);
    char x;
    std::cout << "Моля въведете търсеният от вас символ: ";
    std::cin >> x;
    int l, mid, r;
    // символи са буквите от кирилицата и малките и големите латински букви
    // захолзванието на всички са залити на всички символи
    char a[13] = { 'а', 'в', 'г', 'м', 'н', 'т', 'р', 'ь', 'и', 'е', 'с', 'з', 'ф' };
    l = 0;
    r = 12; //n-1
    mid = (l+r) / 2;
    while(a[mid] != x && r >= l){
        for(int i = l; i <= r; i++){
            std::cout << a[i] << " ";
        }
        std::cout << std::endl;
        char leftchar = a[l];
        char midchar = a[mid];
        char rightchar = a[r];
        std::cout << "ляв край: \\" << leftchar
        << "\\\" среда: \\" << midchar
        << "\\\" десен край: \\" << rightchar << "\\"
        << std::endl << std::endl;
        if(a[mid] < x)
            l = mid + 1;
        else
            r = mid - 1;
        mid = (l + r) / 2;
    }
    if(l - r == 1){ // ако различавате в 1, не можем да сложим mid. тогава излизаме от цикъла.
        std::cout << "Нямаме среда! Излизам..." << std::endl;
        break;
    }
    if(r >= 1)
        std::cout << "Търсеният символ присъства в множеството!" << std::endl;
    else
        std::cout << "Търсеният символ не беше намерен в множеството." << std::endl;
    return 0;
}

```

Моля въведете търсеният от вас символ: ф
 а в г м н о р ъ 1 2 3 6 f
 ляв край: "а"
 среда: "р"
 десен край: "f"
 ъ 1 2 3 6 f
 ляв край: "ъ"
 среда: "2"
 десен край: "f"
 Нямаме среда! Излизам...
 Търсеният символ не беше намерен в множеството.
 Process returned 0 (0x0) execution time : 1.860 s
 Press any key to continue.

дихотомичен метод (binary search)



4. Време. Напишете тези три програми като три функции за претърсване на масив за дадена стойност: (1. Която претърсва винаги целия масив; 2. Която реализира търсене с котва; 3. Която реализира алгоритъма за дихотомично търсене).

За всяка от трите направете следното: - На входа на двете подайте един и същи масив със 100 000 случаен генериирани елемента, а за дихотомичното търсене – числата от 1 до 100 000, наредени възходящо. - измерете времето, което отнема на всяка от функциите да намери 5 стойности измежду тези 100 000 елемента. Търсените 5 стойности се задават от клавиатура и се подават едни и същи към трите функции. Времето за търсене е с много малка стойност и се губи точност при записването ѝ във float/double. Затова повторете 10 пъти в цикъл търсенето на една и съща стойност по всеки от методите и измерете общото за десетте пъти време. Извеждайте в конзолата или записвайте във файл измерените времена, заедно с броя елементи на масива от входа. Повторете същото измерване за 200 000, 300 000 и така до 1 000 000 елемента.

Получените серии от стойности за всеки алгоритъм поставете в Excel или друга електронна таблица и изчертайте трите серии на една графика, която показва какво е времето, което отнема на всеки алгоритъм за даден вход. Например така:

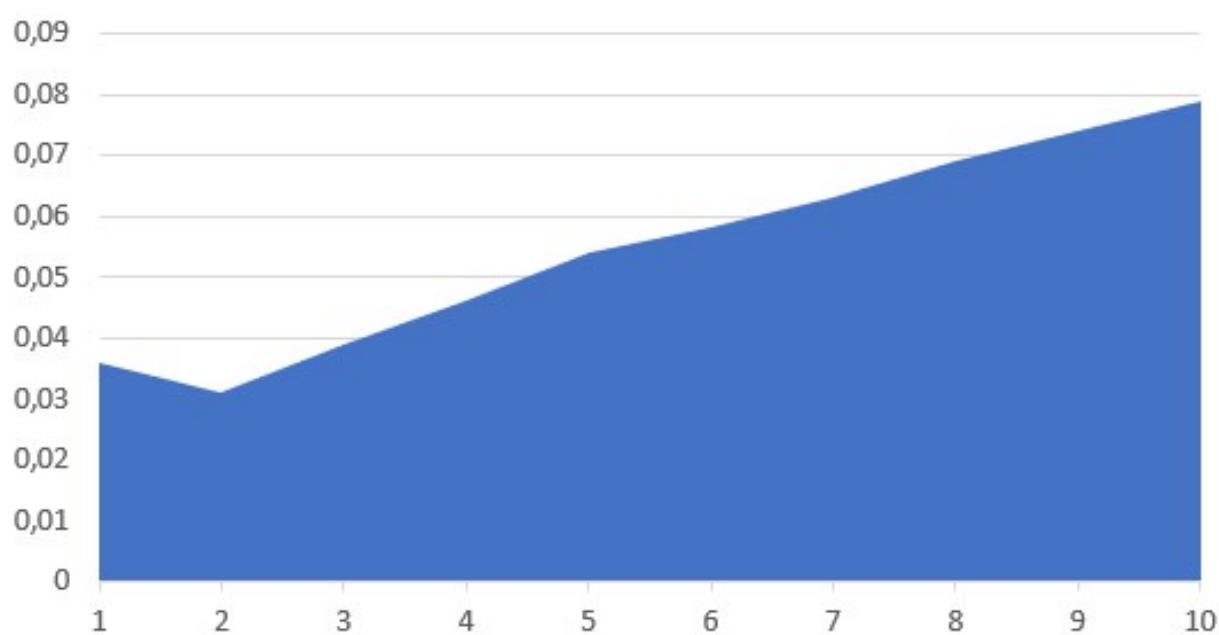
метод/брой подадени елементи	време за изпълнение		
	търсене чрез обхождане на целия масив	търсене до намиране - с котва	дихотомично претърсване
100 000	25	22	10
200 000	45	34	12
300 000	56	47	13
400 000	68	56	13,5
500 000	това не са измерени стойности, това е пример.		
600 000			
700 000			

Фигура 22: таблица

6.4 ОТГОВОР

време за изпълнение			
метод/размер	линейно търсене	котва	дихотомично търсене
100 000	0,036	0,046	0,046
200 000	0,031	0,044	0,046
300 000	0,039	0,045	0,04
400 000	0,046	0,051	0,045
500 000	0,054	0,058	0,045
600 000	0,058	0,061	0,044
700 000	0,063	0,065	0,041
800 000	0,069	0,068	0,044
900 000	0,074	0,078	0,042
1 000 000	0,079	0,081	0,044

линейно търсене



котва



дихотомично търсене



ДОМАШНО 7

Краен срок: четвъртък, 18 април 2024, 00:00 AM

Предадено на: събота, 13 април 2024, 17:38 PM

- Напишете на лист със схема програмата за намиране **на елемент с минимална стойност** и на неговия индекс, в едномерен масив. Оградете с правоъгълник оператора, който осигурява запазване на индекса на елемента с минимална стойност. Запишете колко сравнения на стойности прави тази програма. Пуснете и предайте програмата с вход и изхода – снимки на еcran.

7.1 ОТГОВОР

```
1 #include <iostream>
2
3 int main()
4 {
5     int a[6] = {7, -8, 2, 94, 1, 504};
6
7
8     int minimal = a[0];
9     int minimalIndex = 0;
10
11    for(int i = 1; i < 5; i++){
12        if(minimal > a[i]){// брой сравнения: n-1
13            minimal = a[i];
14            minimalIndex = i;
15        }
16    }
17
18    std::cout << minimal << " @ position " << minimalIndex;
19    std::cin.get(); // фигурира като пауза на програмата
20    return 0;
21
22 }
```

-8 @ position 1

2. Сортиране на един масив в друг масив.

- (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.

7.2 ОТГОВОР

В отделна структура за цикъл обхождаме n на брой пъти нашия разбркан масив, за да извлечем максималната стойност от него, заедно с позицията на която се намира. След което я запазваме в променлива, изпълнявайки това във друг блок от два вложени цикъла с еднакви условия за проверка, но с различни променливи за обозначение. Обхождаме същия масив като този път се стремим да извлечем неговата минимална стойност.

Установявайки минималната стойност, чрез най-младшия цикъл в дадената структура, тя се попълва във втори празен масив с идентична размерност чрез приемане на стойността за място на втория масив от най-старшия цикъл да бъде равна на намерената досегашна минимална стойност.

Позицията на намерената минимална стойност се презаписва с досегашният максимум, следвайки задаването на нов минимум приемащ стойността на максимума.

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

7.3 ОТГОВОР

Задачата на оператора за присвояване на стойност в този случай се използва за задаване на нова минимална стойност на масива, имайки предвид че досегашната винаги ще бъде по-малка от сравняващата се поредна на масива.

За да избегнем запълване с еднакви стойности, презаписваме минималната стойност с намерената максимална, за да можем да осъщесвим сравнението на новия минимум в младшата конструкция за цикъл с поредната такава от масива, стигайки до нова минимална стойност.

- (в) Запишете колко пъти се изтегля минимална стойност по този алгоритъм, ако масивът е n – местен.

7.4 ОТГОВОР

Изтеглянето на минималната стойност се случва n на брой пъти, където “ n ” е размерът на масива, с който работим.

3. Напишете на свободните позиции първите четири цифри от факултетния си номер. Това е масив A, стойностите на който трябва да се появят сортирани в масива B.

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

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

- (a) Напишете успоредно на схемата за управление оператори на избран от вас език за програмиране.

7.5 ОТГОВОР

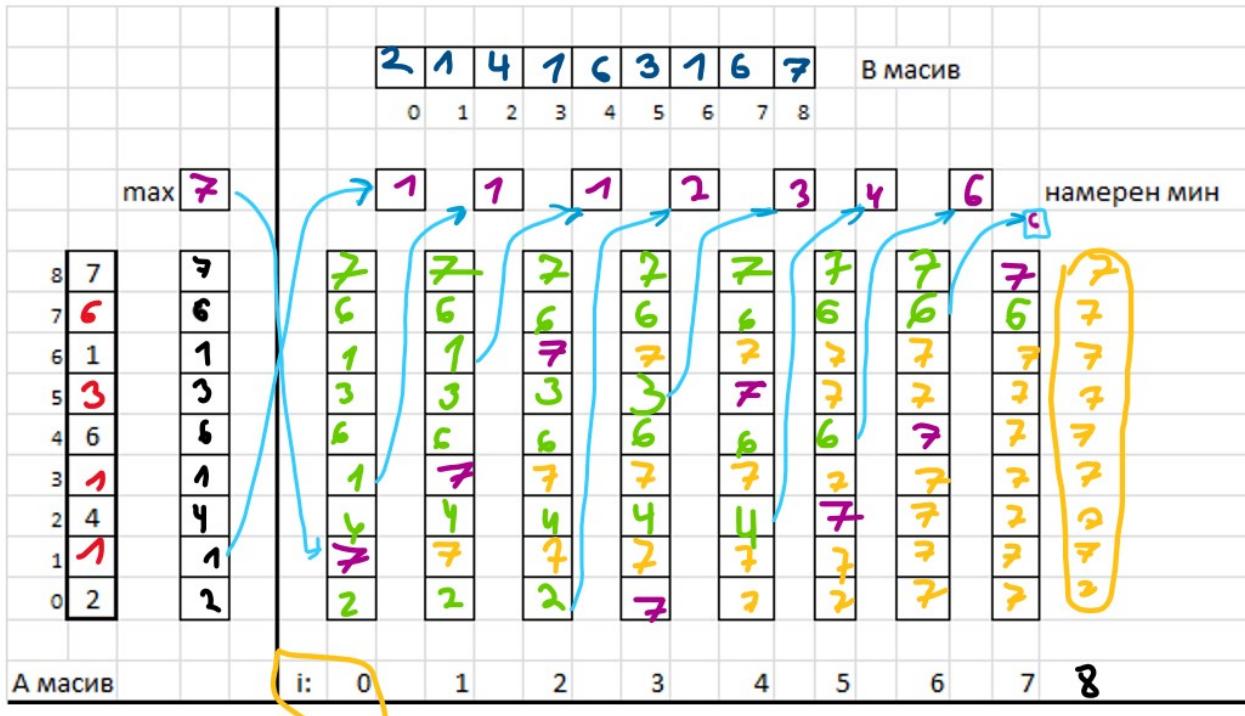
```
1 #include <iostream>
2
3
4 int main()
5 {
6     int random[9] = {2, 1, 4, 1, 6, 3, 1, 6, 7};
7     std::cout << "Original" << std::endl;
8     for(auto& s : random)
9         std::cout << s << " ";
10    std::cout << std::endl << std::endl;
11    int sorted[9];
12
13    int maximal = random[0];
14    int maximalIndex = 0;
15
16    for(int i = 1; i < 9; i++){
17        if(maximal < random[i]){
18            maximal = random[i];
19            maximalIndex = i;
20        }
21    }
22    //std::cout << maximal << ":" " << maximalIndex << std::endl;
23
24    int minimal = random[0];
25    int minimalIndex = 0;
26
27    for(int j = 0; j < 9; j++){
28        for(int i = 0; i < 9; i++){
29            if(minimal > random[i]){
30                minimal = random[i];
31                minimalIndex = i;
32            }
33        }
34
35        sorted[j] = minimal;
36        random[minimalIndex] = maximal;
37        for(int k = 0; k < 9; k++){
38            std::cout << random[k] << " ";
39        }
40        std::cout << ", minimal: " << minimal;
41        minimal = maximal; //!!!  

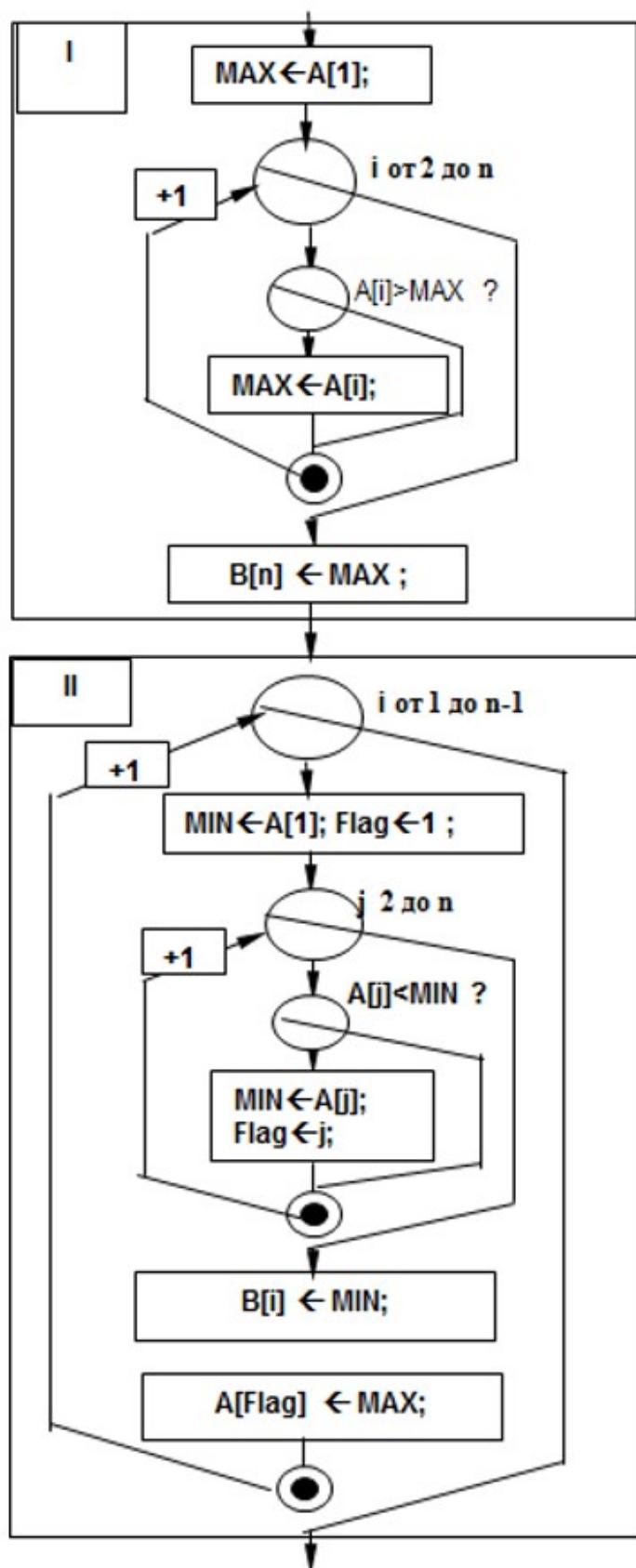
42        std::cout << "; Changed index to max: " << minimalIndex << std::endl;
43
44        std::cout << std::endl;
45    }
46
47    std::cout << std::endl << "Sorted" << std::endl;
48    for(int k = 0; k < 9; k++){
49        std::cout << sorted[k] << " ";
50    }
51 }
```

```
52     return 0;  
53 }
```

- (6) На схемата за трасиране - Трасирайте алгоритъма при тези входни данни. Отбележете с стрелки коя стойност къде отива. Отбележете какви стойности се получават за всяко състояние на масива A до пълното му ... унищожаване..

7.6 ОТГОВОР



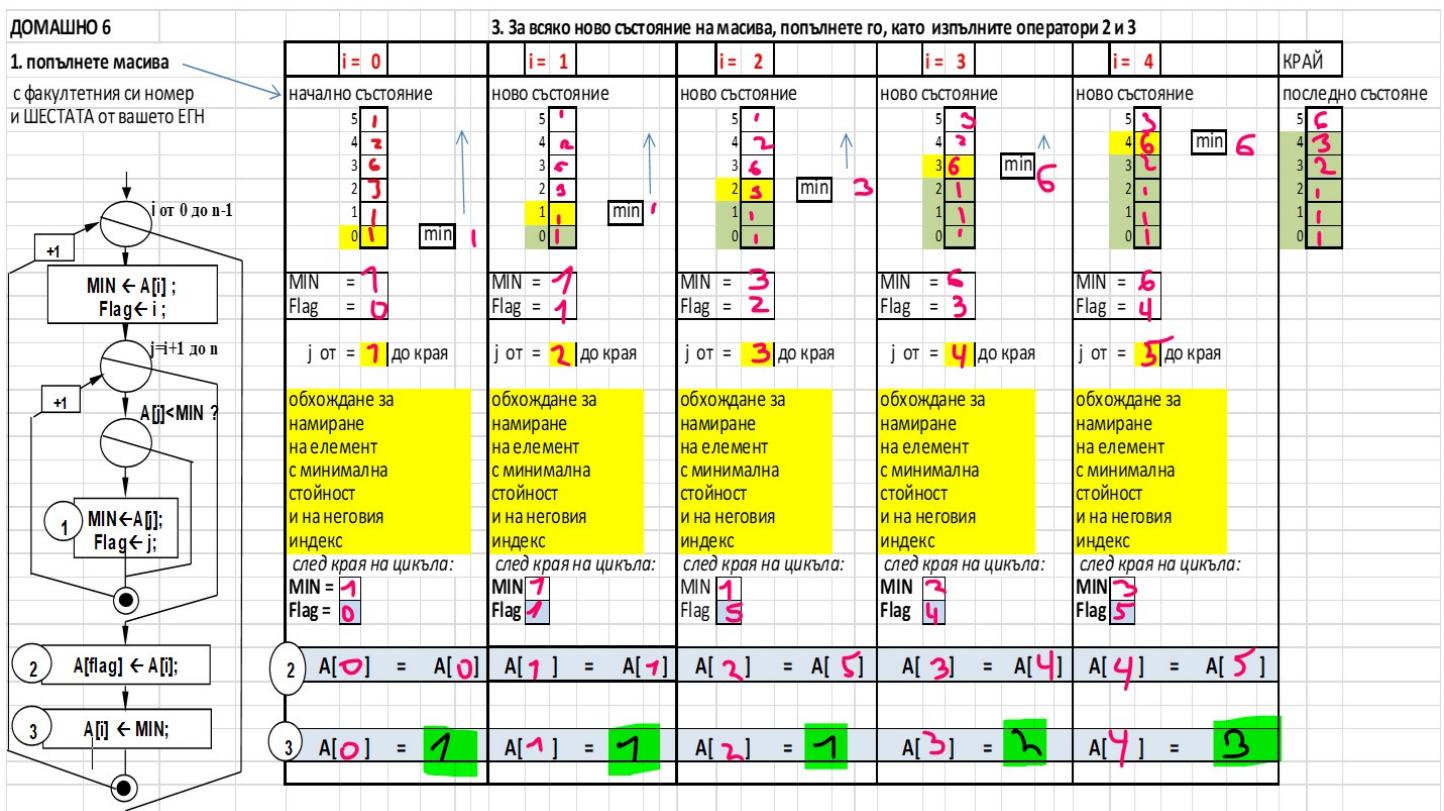


Забележка – в схемата горе с индекс 1 е отбелелязат първият елемент.

4. 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.

7.7 ОТГОВОР



Фигура 23: Трасиране

5. 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 схемата на управление, **напишете програмата**, в точно съответствие със схемата, за сортиране на масив по метода на пряката селекция.

Предайте работеща програма, снимки на еcran с вход – какавто е изискван по задача 3 и изход – нарадения масив. Програмата да извежда на еcran: масив, номер на обхождането за изтегляне на минимум и броя на сравнения извършени във вътрешния цикъл. Например:

*Състояние 0: 6,8,3,7,4,9
обхождане 1, сравнения 5*

*Състояние 1: 3,8,6,7,4,9
обхождане 2, сравнения 4*

*Състояние 2: 3,4,6,7,8,9
обхождане 3, сравнения 3*

и така нататък.

7.8 ОТГОВОР

The screenshot shows a C++ code editor and a terminal window. The code in the editor is a bubble sort algorithm with some additional reporting logic. The terminal window shows the execution of the program with the following output:

```
Състояние 0: 1, 1, 3, 6, 2, 1, 8,  
Обхождане 1, сравнения 6  
  
Състояние 1: 1, 1, 3, 6, 2, 1, 8,  
Обхождане 2, сравнения 5  
  
Състояние 2: 1, 1, 3, 6, 2, 1, 8,  
Обхождане 3, сравнения 4  
  
Състояние 3: 1, 1, 1, 6, 2, 3, 8,  
Обхождане 4, сравнения 3  
  
Състояние 4: 1, 1, 1, 2, 6, 3, 8,  
Обхождане 5, сравнения 2  
  
Състояние 5: 1, 1, 1, 2, 3, 6, 8,  
Обхождане 6, сравнения 1  
  
Състояние 6: 1, 1, 1, 2, 3, 6, 8,  
Обхождане 7, сравнения 0  
  
1 1 1 2 3 6 8  
Process returned 0 (0x0) execution time : 0.018 s  
Press any key to continue.
```

ДОМАШНО 8

Краен срок: вторник, 14 май 2024, 00:00 AM

Предадено на: неделя, 28 април 2024, 12:57 PM

1. Броене. Сложност по време.

Запишете формулата на Гаус за сумата на първите n естествени числа, приложете я и разпишете колко сравнения на стойности прави алгоритъма Пряка Селекция.

8.1 ОТГОВОР

За масив от 555 елемента, алгоритъмът извежда 153 735 на брой сравнения.

```
1 #include <iostream>
2 #include <stdlib.h>
3
4
5 int rand_int(int maxval){
6     return rand() % maxval + 1;
7 }
8
9 const int ARRAY_SIZE = 555;
10
11
12 void selectionSort(int* a, const int& sz){
13     int myMin{};
14     int flag{};
15     int sum{};
16     int f = 1;
17     int s{};
18
19     for(int i = 0; i <= sz-1; i++){
20         myMin = a[i];
21         flag = i;
22         for(int j = i + 1; j < sz; j++){
23             s++;
24             if(a[j] < myMin){
25                 myMin = a[j];
26                 flag = j;
27             }
28         }
29         a[flag] = a[i];
30         a[i] = myMin;
31     }
32 }
33
34
35
36
37 int srav = (sz * (sz - 1)) / 2; // Carl F. Gauss
38 if(srav == s){
39     std::cout << std::endl << std::endl << "Sorted:\n";
```

```
40     for(int i = 0; i < sz; i++){
41         std::cout << a[i] << " ";
42     }
43     std::cout << std::endl << "Sravnenyia: " << '[' << sz << ',' << sz - 1 <<
44 }
45
46 }
47
48 int main(){
49
50
51     int a[ARRAY_SIZE];
52
53     for (int i = 0; i < ARRAY_SIZE; i++)
54     {
55         a[i]=rand_int(100);
56     }
57
58     for(const auto& s : a)
59         std::cout << s << " ";
60
61     selectionSort(a, ARRAY_SIZE);
62
63
64     return 0;
65 }
```

Фигура 24: изход от изпълнение

2. **Инверсии.** Съставете масив от цифрите на факултетния си номер и втората цифра от вашето ЕГН. Колко са инверсиите?

- (а) Изобразете масива графично.
- (б) Съставете схемата на пребояване, както е дадена в материала.
- (в) Пребройте инверсиите за ВСЯКА позиция, както е на схемата. Съберете ги.

8.2 ОТГОВОР

```
1 #include <iostream>
2
3 void doSwap(int& a, int& b){
4     int temp = a;
5     a = b;
6     b = temp;
7 }
8
9 const int sz = 7;
10
11 int main()
12 {
13
14     int a[sz] = {1, 6, 3, 1, 4, 1, 2};
15     int j{};
16     int g{};
17     int og{};
18
19     int sum{};
20
21
22     while(j < sz){
23         for(int i = 0; i < sz; i++){
24             if(a[i] != a[sz-1]){
25                 if(a[i] > a[i+1]){
26                     doSwap(a[i], a[i+1]);
27                     g++;
28                 }
29             }
30         }
31         if(g == og)
32             break;
33
34
35         std::cout << '[';
36         for(const auto& d : a)
37             std::cout << d << " ";
38         std::cout << ']' << std::endl;
39
40         std::cout << "Swaps from position " << j << ":" << g << std::endl;
41         sum += g;
42         og = g;
43         std::cout << "current sum: " << sum << std::endl << std::endl;
44
45         j++;
46     }
47
48
49
50     return 0;
51 }
```

```

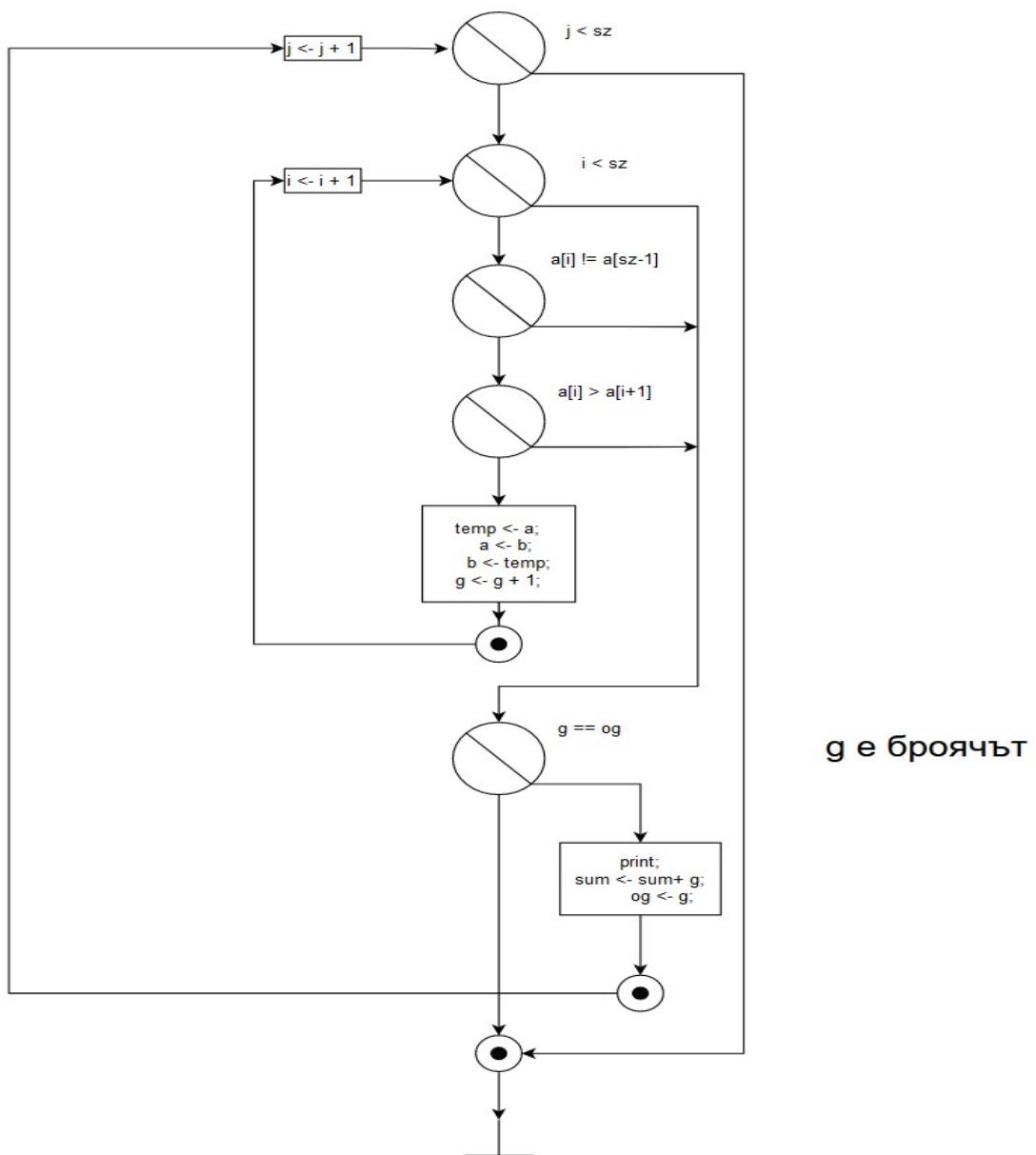
D:\Documents\
[1 3 1 4 1 2 6 ]
Swaps from position 0: 5
current sum: 5

[1 1 3 1 2 4 6 ]
Swaps from position 1: 8
current sum: 13

[1 1 1 2 3 4 6 ]
Swaps from position 2: 10
current sum: 23

```

Фигура 25: изход от изпълнение



Фигура 26: опорна схема на масив [1, 6, 3, 1, 4, 1, 2]

3. **Инверсии.** Съставете сами алгоритъм и програма за пребояване на инверсиите в масив. Колко сравнения прави вашият алгоритъм? Защо? - обяснете с едно изречение. Обяснете това са думи.

8.3 ОТГОВОР

```
1 //personal algorithm
2
3 #include <iostream>
4 #include <stdlib.h>
5 #include <time.h>
6
7
8 int randInt(int);
9 void doSwap(int&, int&);
10 void printArr(int*, const int&);
11 void mySort_bubble(int*, const int&);
12
13
14
15 const int n = 1000;
16
17 int main()
18 {
19
20     char f;
21     clock_t c0, c1;
22     srand ( time(NULL) );
23     int a[n];
24
25     c0 = clock();
26
27     for(int i = 0; i < n; i++)
28         a[i] = randInt(100);
29     std::cout << "original:" << std::endl;
30     printArr(a, n);
31     std::cout << std::endl;
32
33
34     mySort_bubble(a,n); // THE ALGORITHM
35
36     c1 = clock();
37     std::cout << "\nElapsed wall clock time: " << (float) (c1 - c0) / CLOCKS_PER_T
38     std::cout << "NOTE: To have precise measure, please turn off all prints to t
39     std::cin.get(f);
40     if(f == 'y')
41         printArr(a,n); // release me
42     else
43         return 0;
44
45 }
```

```

47
48
49
50 void mySort_bubble(int* a, const int& n){
51
52     int c{};
53     int oks{};
54     int swaps{};
55     int wasted{};
56
57     int t{};
58     int s = t+1;
59
60     while(c != n){
61         while(t < n-1){
62             //printArr(a,n);
63             //std::cout << "a[t] = " << a[t] << " (" << t << ")" << std::endl;
64             //std::cout << "a[s] = " << a[s] << " (" << s << ")" << std::endl;
65             if(a[t] > a[s]){
66                 doSwap(a[t], a[s]);
67                 swaps++;
68                 //std::cout << "\t\t\t\tSWAPPED!\n";
69                 //std::cin.ignore();
70             }
71             else{
72                 //std::cout << "\t\t\t\tOK!\n";
73                 oks++;
74                 //std::cin.ignore();
75             }
76             t++;
77             s = t+1;
78
79         }
80     }
81
82
83     /// RESET
84     //std::cout << "\t\tRESET!!!" << std::endl;
85     //std::cout << "\t\tSWAPS = " << swaps << std::endl;
86     //std::cout << "\t\tOKS = " << oks-1 << std::endl << std::endl;
87     wasted++;
88     if(swaps == 0){
89
90         std::cout << "NO MORE SWAPS LEFT TO BE MADE!!! WE ARE DONE SORTING!";
91         std::cout << "->TOTAL COUNT OF ITERATIONS: " << wasted << std::endl;
92         //printArr(a,n);
93         break;
94     }
95     oks=0;
96     swaps=0;
97     t=0;
98     s=t+1;
99     c++;

```

```

100     }
101 }
102 int randInt(int m){
103     return rand() % m + 1;
104 }
105 void doSwap(int& a, int& b){
106     int temp = a;
107     a = b;
108     b = temp;
109 }
110 void printArr(int* a, const int& sz){
111     std::cout << '[';
112     for(int i = 0; i < sz; i++)
113         std::cout << a[i] << " ";
114     std::cout << ']' << std::endl;
115 }
116 }
```

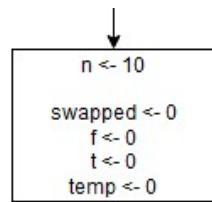
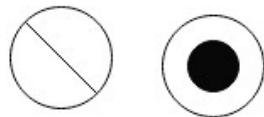
Алогитмът ми изпълнява $\pm 2\%$ по-малък брой проверки, за да изпълни сортирането. Причината за това се дължи на свърху основната проверка за направени инверсии. При проверки след обхождане равни на 0, програмата влиза в условието и приключва своето изпълнение.

Така се освобождава от излишни допълнителни проверки на вече наредени елементи. При тест за 1000 случаенни елемента между 1 и 100, броят сравнения е 966 за момента на провеждането му; Следващ брой проверки показва стойност, която е синаги по-малка от размера на масива.

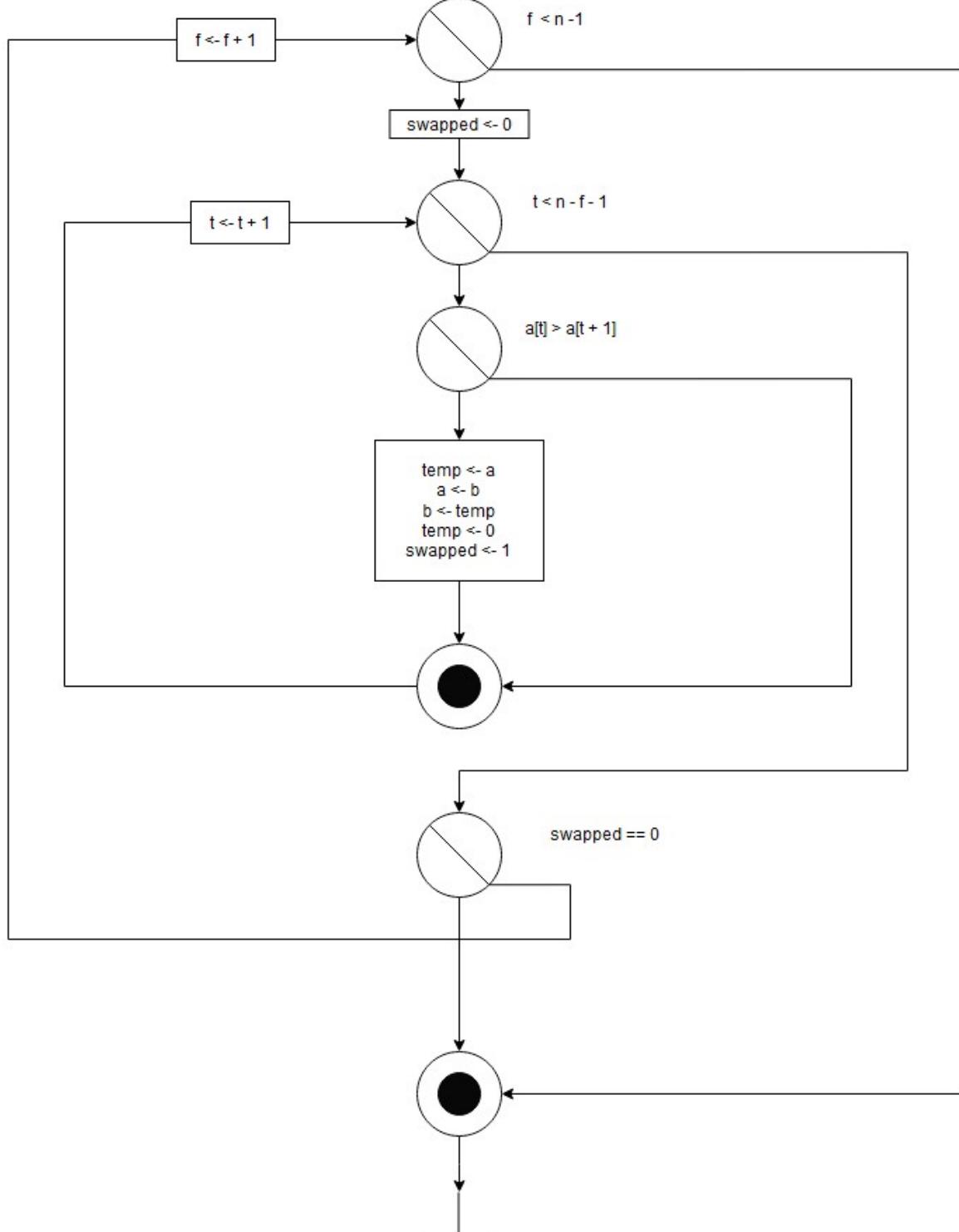
4. **Сортиране.** Метод на проката размяна (мехурчето). Съставете опорна схема (масивът и какво се прави с него, с означени имена на променливи и т.н.), Съставете схема на управление със спиране по брой инверсии (почти е готова в Moodle) и успоредно на нея – програма.

8.4 ОТГОВОР

```
1 #include <iostream>
2
3
4 void mySwap(int& a, int& b){
5     int temp = a;
6     a = b;
7     b = temp;
8     temp = NULL;
9 }
10
11 void bubble_sort(int* a, const int& n)
12 {
13
14     bool swapped;
15     for (int f = 0; f < n - 1; f++) {
16         swapped = false;
17         for (int t = 0; t < n - f - 1; t++) {
18             if (a[t] > a[t + 1]) {
19                 mySwap(a[t], a[t + 1]);
20                 swapped = true;
21             }
22         }
23         if (swapped == false)
24             break;
25     }
26 }
27
28 const int n = 10;
29
30 int main()
31 {
32
33     int a[n] { 3, 2, 7, 1, 0, 6, 4, 8, 5, 9 };
34
35     bubble_sort(a, n);
36
37     for(const auto& s : a)
38         std::cout << s << ',';
39
40     return 0;
41 }
```



зад. 4



5. Отговорете писмено на всеки от следните въпроси .

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

8.5 ОТГОВОР

Най-голям брой сравнения при разглеждане на алгоритъма, водещ се по метода на мехурчето имаме при масив, чиито елементи са наредени в низходящ (т.нар "обратен") ред. Тук сложността, т.е. времето за обхождане и пренарездане, изразяваме като n^2 , имайки в предвид двата вложени цикъла необходими за реализирането на алгоритъма, всеки с отделна сложност n ;

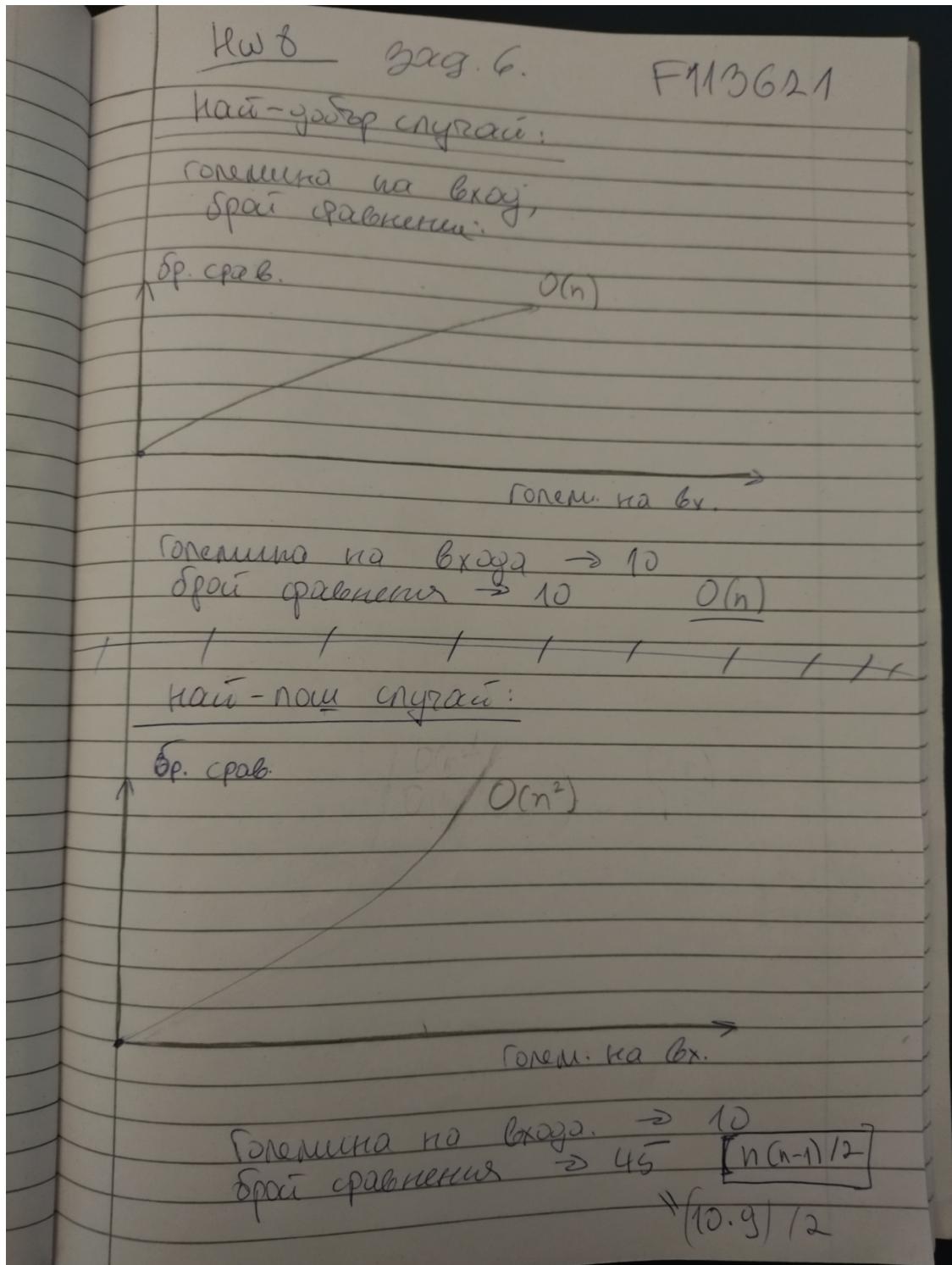
- (б) Колко сравнения на стойности **най-малко** се правят при сортиране по метода на мехурчето. При какви входни данни стави това.

8.6 ОТГОВОР

Най-малък брой сравнения имаме при вече сортирани елементи на масива. Тези са 0 на брой. Сложността тук е n .

6. Направете на ръка как изглеждат теоретично двете графики (за най-добрия и най-лошия случай, посочени в задача 5) на зависимостта между големината на входа (броят на елементите в масива) и броя на сравненията. Означете на графиките за кой случай те се отнасят.

8.7 ОТГОВОР

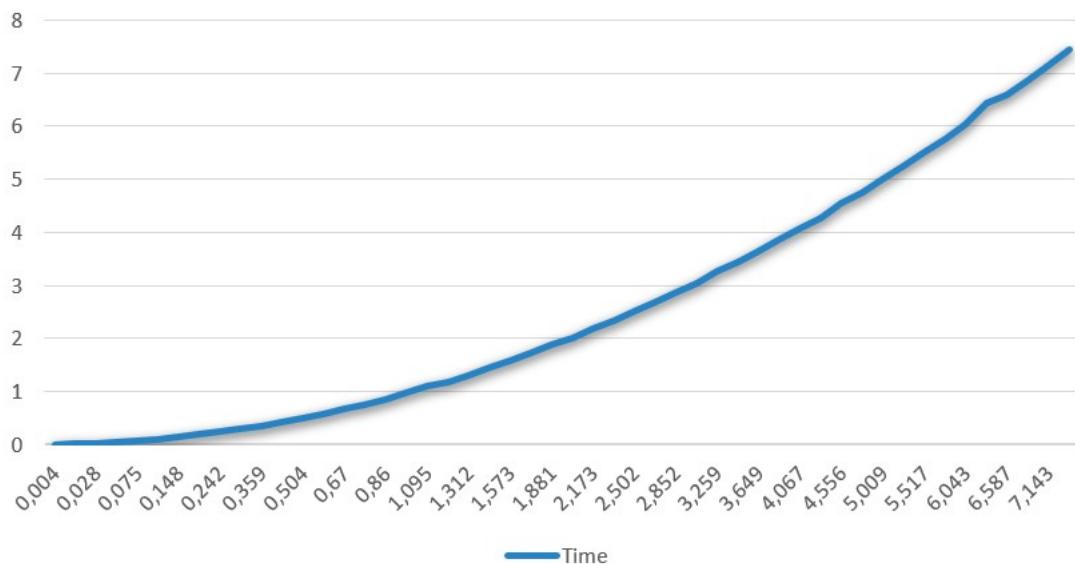


7. Съставете програма, която реализира метода на мехурчето в **два варианта** – когато външният цикъл е в зависимост от броя на елементите в масива и когато външният цикъл е от друг тип – прекратява изпълнението си ако в поредното преминаване през неподредената част на масива не са направени размени. (Не се позволява да прекъсвате цикъла по бояч с насилен изход по допълнително условие.) След като имате тези две процедури, използвайте кода, качен в Moodle (същия, който палзвахте за предишно домашно) за измерване времето на изпълнение, и измерете колко време отнема на всяка от двете процедури да сортира един и същ случаен подреден масив с 1000, 2000, ... 50 000 елемента.

Това означава, че за всеки размер трябва да измерите времето И за едната И за другата процедура за една и същ масив. Използвайте стойностите на тези времена, за да начертаете графика в EXCEL, която показва как нараства времето за изпълнение на двете процедури в зависимост от големината на входа. На графиката трябва да е видно коя линия за коя процедура се отнася, както и да са надписани двете оси – за броя входни елементи на масива и за времето на изпълнение.

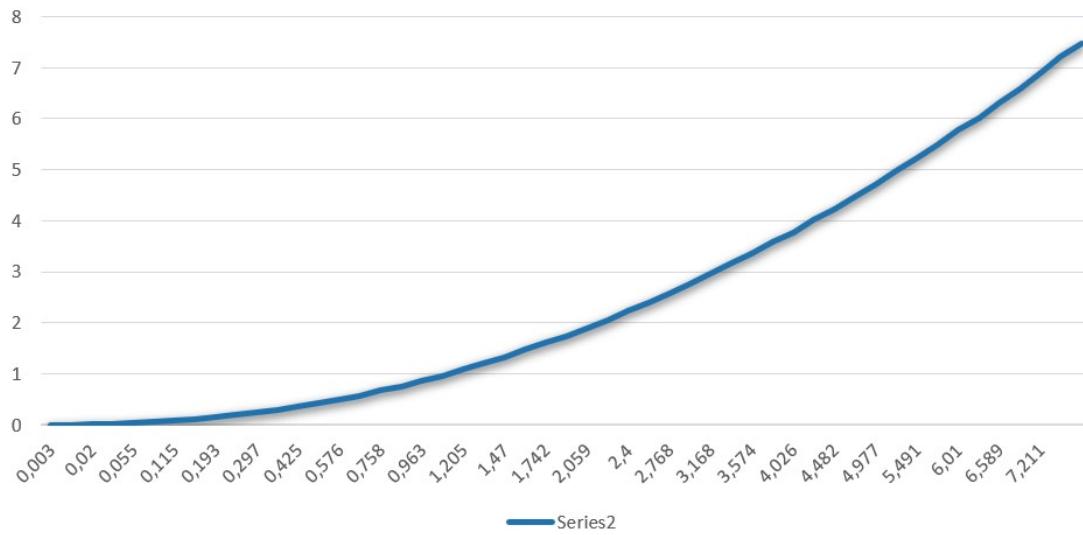
8.8 ОТГОВОР

когато външният цикъл е в зависимост от броя на елементите в масива



Фигура 27: измерване от задача 7.1

когато външният цикъл е от друг тип



Фигура 28: измерване от задача 7.2

ДОМАШНО 9

Краен срок: *събота, 4 май 2024, 00:00 AM*

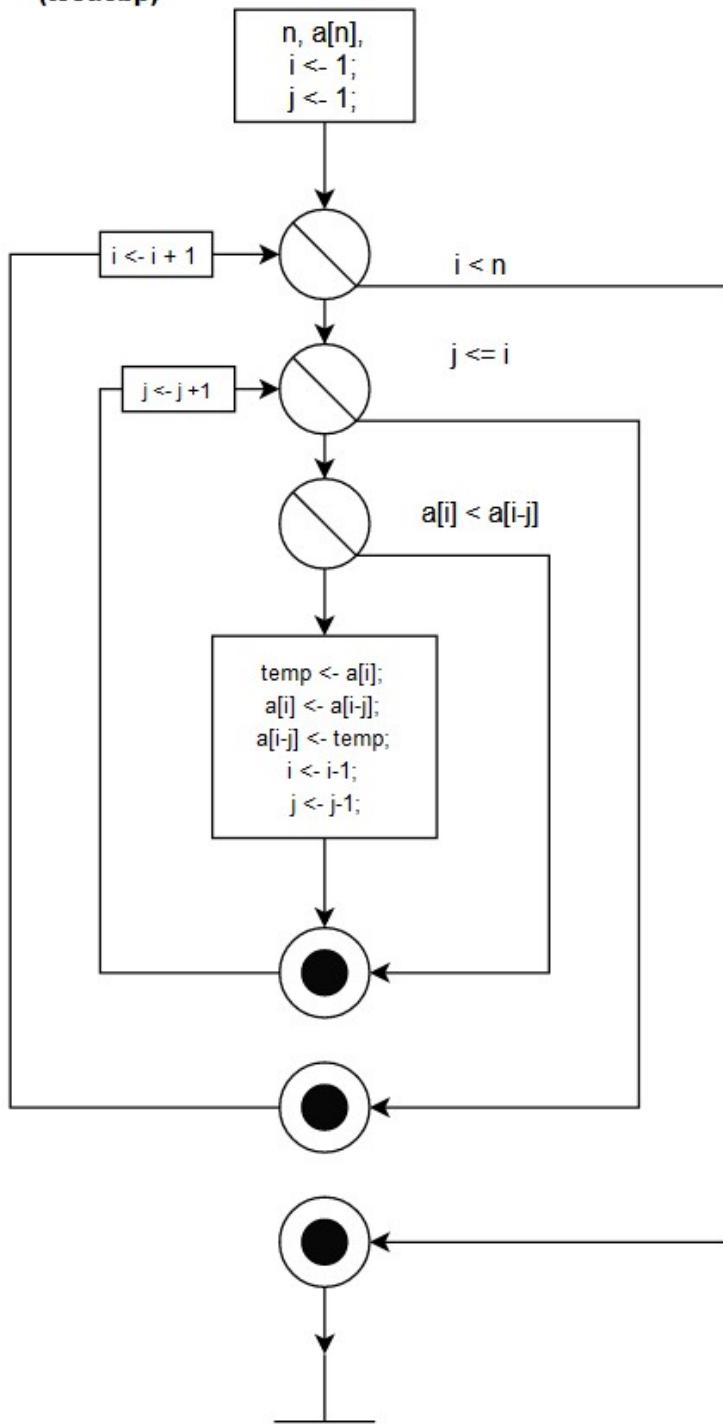
Предадено на: *четвъртък, 2 май 2024, 08:29 AM*

1. Сортиране. Пряко вмъкване – съставете:

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

9.1 ОТГОВОР

зад. 1
(test.cbp)



2. Приложете схеми и примерен програмант текст за цикъла за търсене на място на вмъкване, за следните **два варианта**, придружени съответно със схеми на управление и примерен програмен текст:

- (а) Слизане към началото на масива **докато**: или се стигне до по-малък елемент, или се стигне до началото на масива.

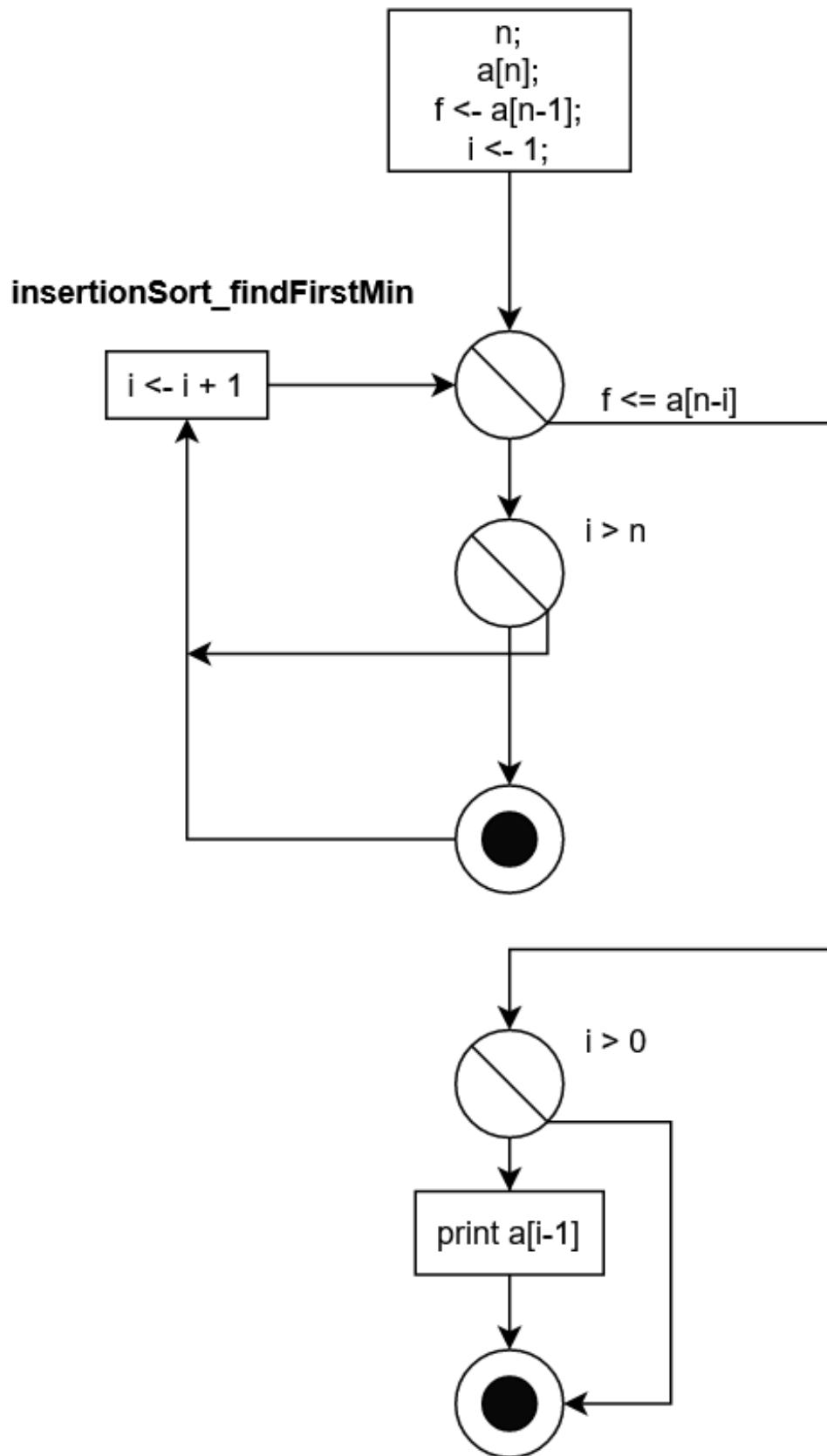
9.2 ОТГОВОР

```
1 #include <iostream>
2 #include <stdlib.h>
3
4
5
6
7 void printA(int*, const int&);
8 void insertionSort_findFirstMin(int*, int);
9 void insertionSort_anchorSort(int*, int&);
10
11 int getPos(int*, int&, int);
12 int findMin(int*, int&, int);
13
14 int randInt(int);
15
16
17
18
19 int main()
20 {
21     int N = 18;
22     int* a = new int[N];
23     for(int i = 0; i < N; i++){
24         a[i] = randInt(100);
25     }
26
27     std::cout << "Original: "; printA(a, N); std::cout << std::endl;
28
29     insertionSort_findFirstMin(a, N);
30     std::cout << std::endl;
31     insertionSort_anchorSort(a, N);
32
33     delete[] a;
34     a = NULL;
35
36     return 0;
37 }
38
39 void printA(int* a, const int& sz){
40     std::cout << '[';
41     for(int i = 0; i < sz; i++){
42         if(i + 1 == sz)
43             std::cout << a[i];
44         else
45             std::cout << a[i] << ", ";
46     }
47     std::cout << ']';
48     std::cout << "\n";
49 }
50
51 int getPos(int* a, int& n, int s){
```

```

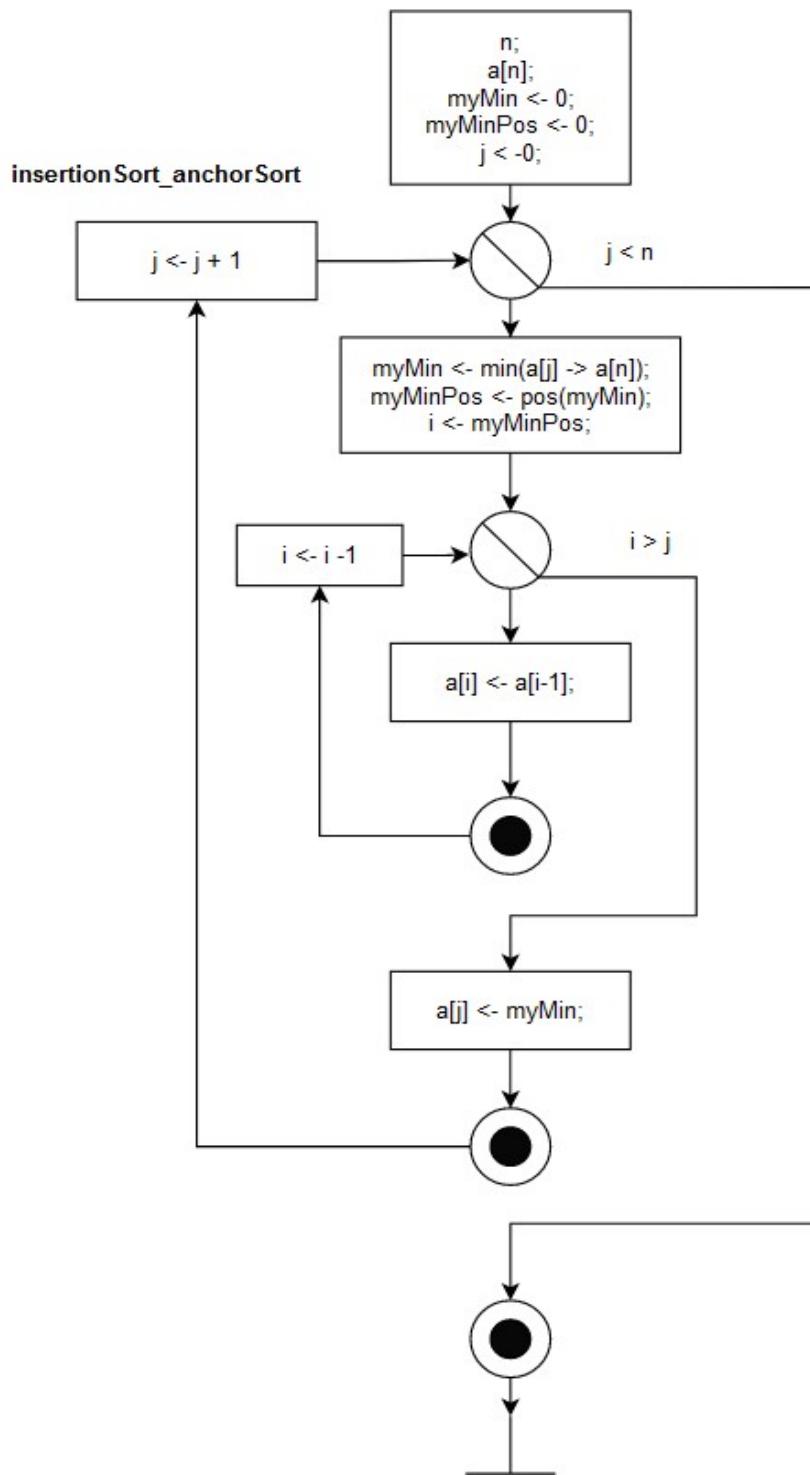
53
54     for(int i = 0; i < n; i++){
55
56         if(a[i] == s){
57             return i;
58
59         }
60
61     }
62
63 }
64 int findMin(int* a, int& n, int s){
65
66     int theMin = a[s];
67
68     for(int i = s; i < n; i++){
69
70         if(a[i] < theMin){
71             theMin = a[i];
72         }
73
74     }
75
76     return theMin;
77 }
78
79
80
81
82
83 void insertionSort_findFirstMin(int* a, int n){
84     std::cout << "_findFirstMin: ";
85     int f = a[n-1];
86     int i = 1;
87     while(f <= a[n-i]){
88         if(i > n)
89             break;
90         //std::cout << f << " > " << a[n-i] << std::endl;
91
92         i++;
93     }
94     if(i > 0){
95         std::cout << a[n-i] << " < " << f << std::endl;
96
97     } else {
98         std::cout << "End reached! Nothing was found." << std::endl;
99     }
100 }
101
102
103 void insertionSort_anchorSort(int* a, int& n){
104     std::cout << "_anchorSort:\n";
105 }
```

```
106     int myMin{};
107     int myMinPos{};
108     int j{};
109
110    while(j < n){
111        //find new min
112        myMin = findMin(a, n, j);
113        myMinPos = getPos(a, n, myMin);
114        for(int i = myMinPos; i > j; i--){
115            a[i] = a[i-1];
116        }
117
118        //move min to j
119        a[j] = myMin;
120        j++;
121
122        printA(a, n);
123    }
124
125 }
126
127 int randInt(int x){
128     return rand() % x + 1;
129 }
```



- (6) **Котва.** Реализирайте „слизането“ с котва. Това означава да поставяте вмъквания елемент „най-отдолу“ на масива, за да спестите от условието за край на „слизането“. (Индексите са цели числа – проверете дали отрицателните индекси са легитимни за езика, в който работите.)

9.3 ОТГОВОР



3. Обясните с думи как става освобождаването на място за вмъкване на вмъквания елемент, посочете с кои оператори от предложенияя програмен текст става това.

9.4 ОТГОВОР

Обхождаме масива докато не намерим минималният му елемент. След като го открием отместваме всички останали елементи с една позиция, стартирайки от края на масива като стигаме до позицията на намереният минимум. Презаписваме останалото място с намереният минимален елемент.

4. Обяснете с думи **зашо** този метод работи по-добре (с по-малка сравнения), ако във входния масив няма инверсии.

9.5 ОТГОВОР

Не изисква извикването на отделна функция.

Работи в едно пространство.

Работи със сложност по-малка от тази при използването на инверсии.

5. Реализирайте на машина двата варианта на програмата, всеки от тях в два случая на входни данни – при нареден масив на входа и при обратно нареден масив на входа (общо четири реализации). Разпечатайте на изхода **броя сравнения** при двата вида входни масиви от 10000, 15000, 20000 и 30000 елемента.

- (a) Приложете снимки на еcran на кода и на изхода от изпълнение.

9.6 ОТГОВОР

```
1
2 #include <iostream>
3 #include <stdlib.h>
4
5
6
7
8 void printA(int*, const int&);
9 void insertionSort_findFirstMin(int*, int);
10 void insertionSort_anchorSort(int*, int&);
11
12
13 int main()
14 {
15     int N{};
16     for(; std::cin >> N;){
17         int* a = new int[N];
18         int* b = new int[N];
19         for(int i = 0; i < N; i++){
20             a[i] = i;
21         }
22         for(int i = N - 1; i >= 0; i--){
23             b[i] = i;
```

```

24 }
25
26 //std::cout << "Original: "; printA(a, N); std::cout << std::endl;
27 //std::cout << "Original: "; printA(b, N); std::cout << std::endl;
28
29     insertionSort_findFirstMin(a, N);
30     insertionSort_findFirstMin(b, N);
31     std::cout << std::endl;
32     insertionSort_anchorSort(a, N);
33     insertionSort_anchorSort(b, N);
34
35
36     delete[] a;
37     a = NULL;
38
39     delete[] b;
40     b = NULL;
41 }
42
43
44     return 0;
45 }
46
47 void printA(int* a, const int& sz){
48     std::cout << '[';
49     for(int i = 0; i < sz; i++){
50         if(i + 1 == sz)
51             std::cout << a[i];
52         else
53             std::cout << a[i] << ", ";
54     }
55     std::cout << ']' ;
56     std::cout << "\n";
57 }
58
59
60 void insertionSort_findFirstMin(int* a, int n){
61     std::cout << "_findFirstMin (compares): ";
62     int f = a[n-1];
63     int i = 1;
64     int comps{};
65     comps++;
66     while(f <= a[n-i]){
67         comps++;
68         if(i > n){
69             break;
70         }
71         //std::cout << f << " > " << a[n-i] << std::endl;
72
73         i++;
74     }
75
76     std::cout << comps << std::endl;

```

```

77
78 }
79
80
81 void insertionSort_anchorSort(int* a, int& n){
82     std::cout << "_anchorSort (compares): ";
83
84     int myMin{};
85     int myMinPos{};
86     int j{};
87     int comps{};
88     int theMin{};
89
90     comps++;
91     while(j < n){
92         //find new min
93
94         theMin = a[j];
95
96         for(int i = j; i < n; i++){
97             comps++;
98             if(a[i] < theMin){
99                 theMin = a[i];
100                myMinPos = i;
101            }
102            comps++;
103        }
104
105        for(int i = myMinPos; i > j; i--){
106            a[i] = a[i-1];
107            comps++;
108        }
109
110        //move min to j
111        a[j] = myMin;
112        j++;
113
114    }
115    std::cout << comps << std::endl;
116 }
117 }
```

```
10
_findFirstMin (compares): 2
_findFirstMin (compares): 2

_anchorSort (compares): 111
_anchorSort (compares): 111
10000
_findFirstMin (compares): 2
_findFirstMin (compares): 2

_anchorSort (compares): 100010001
_anchorSort (compares): 100010001
15000
_findFirstMin (compares): 2
_findFirstMin (compares): 2

_anchorSort (compares): 225015001
_anchorSort (compares): 225015001
20000
_findFirstMin (compares): 2
_findFirstMin (compares): 2

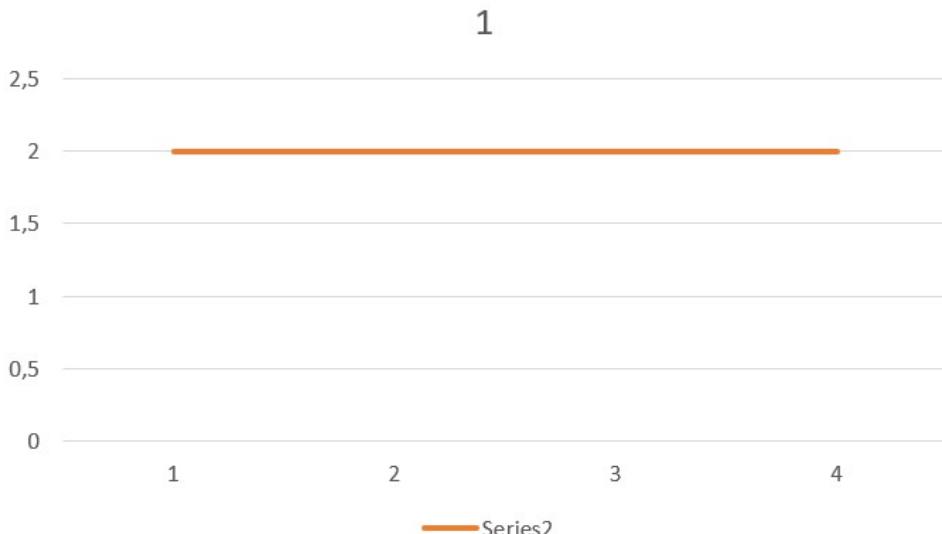
_anchorSort (compares): 400020001
_anchorSort (compares): 400020001
30000
_findFirstMin (compares): 2
_findFirstMin (compares): 2

_anchorSort (compares): 900030001
_anchorSort (compares): 900030001
```

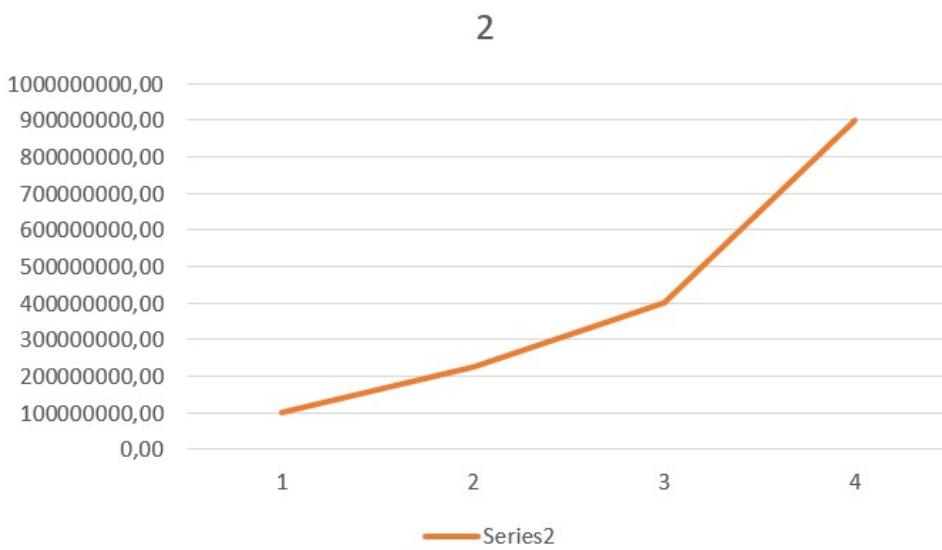
Фигура 29: изход от изпълнение

- (6) Съставете ОБЩА графика, на ексел, на база на големината на входа и броя сравнения за двета варианта, всеки за двета случая на нареденост на масивите.

9.7 ОТГОВОР



Фигура 30: вариант 1



Фигура 31: вариант 2

ДОМАШНО 10

Краен срок: *сряда, 8 май 2024, 00:00 AM*

Предадено на: *сряда, 8 май 2024, 12:43 PM - СЪС ЗАКЪСНЕНИЕ*

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

A:	<table border="1"><tr><td>6</td><td>8</td><td>9</td><td>4</td></tr><tr><td>3</td><td>2</td><td>1</td><td>7</td></tr><tr><td>6</td><td>4</td><td>1</td><td>2</td></tr><tr><td>j</td><td></td><td></td><td></td></tr></table>	6	8	9	4	3	2	1	7	6	4	1	2	j				*	B:	<table border="1"><tr><td>4</td><td>3</td></tr><tr><td>8</td><td>5</td></tr><tr><td>9</td><td>1</td></tr><tr><td>4</td><td>7</td></tr></table>	4	3	8	5	9	1	4	7
6	8	9	4																									
3	2	1	7																									
6	4	1	2																									
j																												
4	3																											
8	5																											
9	1																											
4	7																											
i	<table border="1"><tr><td></td><td></td></tr><tr><td></td><td></td></tr><tr><td></td><td></td></tr><tr><td></td><td></td></tr></table>									C:	<table border="1"><tr><td></td><td></td></tr><tr><td></td><td></td></tr><tr><td></td><td></td></tr><tr><td></td><td></td></tr></table>																	

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

10.1 ОТГОВОР

Матрицата има четири реда и два стълба.

- Отбележете на схемата как бележите индексите на редовете и на стълбовете на C. Запишете ги отделно с думи, например така: *c индекс i бележка редовете на C, а c индекс j - стълбовете.*

10.2 ОТГОВОР

C индекс "i" бележка редовете, а с "j" бележка стълбовете (колоните) на новообразуваната матрица C.

- Запишете как изчислявате всеки от елементите на C. Пресметнете ги. Например така:
 $C_{22} = 3 * 3 + F * E + 1 * 1 + 7 * 7 = \dots = \text{Стойност}$
и попълнете матрицата C със стойностите, които получавате

10.3 ОТГОВОР

Изпълнено по-нататък в документа.

- Запишете управляващия оператор на цикъла, който обхожда по клетки всеки от редовете на C - *For()*. Запишете числови стойности за границите на цикъпа, например $l <= 369$.

10.4 ОТГОВОР

Управляващият оператор, отговарящ за обхождането на всеки от редовете в матрицата C е ' j '.

' j ' е променливата на първият по степен вложен цикъл в серията от цикли умножаващи двете матрици (A и B) в една (матрицата C).

```
1 for(int i = 0; i < Ar; i++){ // main loop
2     for(int j = 0; j < Bc; j++){ // first nested loop
3         ...
4     }
5     ...
6 }
7
```

Зависимостта при увеличаването стойността на '*j*' се определя от променливата "Вс която пази в себе си стойността за броят колони в матрицата *B*".

- (д) Запишете управляващия оператор на цикъла, който обхожда всички редове на *C*. *For()*

10.5 ОТГОВОР

Управляващият оператор, отговарящ за обхождането на редовете на матрицата *C* е '*i*'.

'*i*' е променливата на най-външният цикъл от поредицата цикли умножаващи двете матрици в една (*C*).

```
1 for(int i = 0; i < Ar; i++){}
2
```

Зависимостта при увеличаването на '*i*' се определя от "Ar което обозначава броят редове на матрицата 'A'".

- (е) Запишете вложени един в друг двата оператора за циклите, които обхождат всички клетки на *C*.

10.6 ОТГОВОР

```
1 for(int i = 0; i < Ar; i++){ // main loop
2     for(int j = 0; j < Bc; j++){ // first nested loop
3         ...
4     }
5     ...
6 }
7
```

- (ж) Запишете на схемата горе с какви индекси бележите редовете и стълбовете на *A* и на *B*.

10.7 ОТГОВОР

$$A[i][d]; B[d][j];$$

'*d*' е променлива във втори вложен цикъл със зависимост от "Br отговаряща за броят редове на матрицата *B*".

- (з) Запишете общия вид на Сумата – скаларно произведение, с която изчислихте всеки елемент на *C*.

$$C_{ij} = \sum_{?}^x A_{??} * B_{??}$$

10.8 ОТГОВОР

$$C_{ij} = \sum_{d=0}^{Bc-1} A_{id} * B_{dj}$$

- (и) Запишете цикъла, който натрупва тази сума.

For()
C[][] = C[][] + A[][] * B[][];

10.9 ОТГОВОР (+ 10.3)

```
1 // head diagonal
2 for(int i = 0; i < Ar; i++){
3     for(int j = 0; j < Bc; j++){
4         x=0;
5         for(int d = 0; d < Br; d++){
6             x += A[i][d] * B[d][j];
7         }
8         C[i][j] = x;
9     }
10 }
11 }
12 }
```

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.

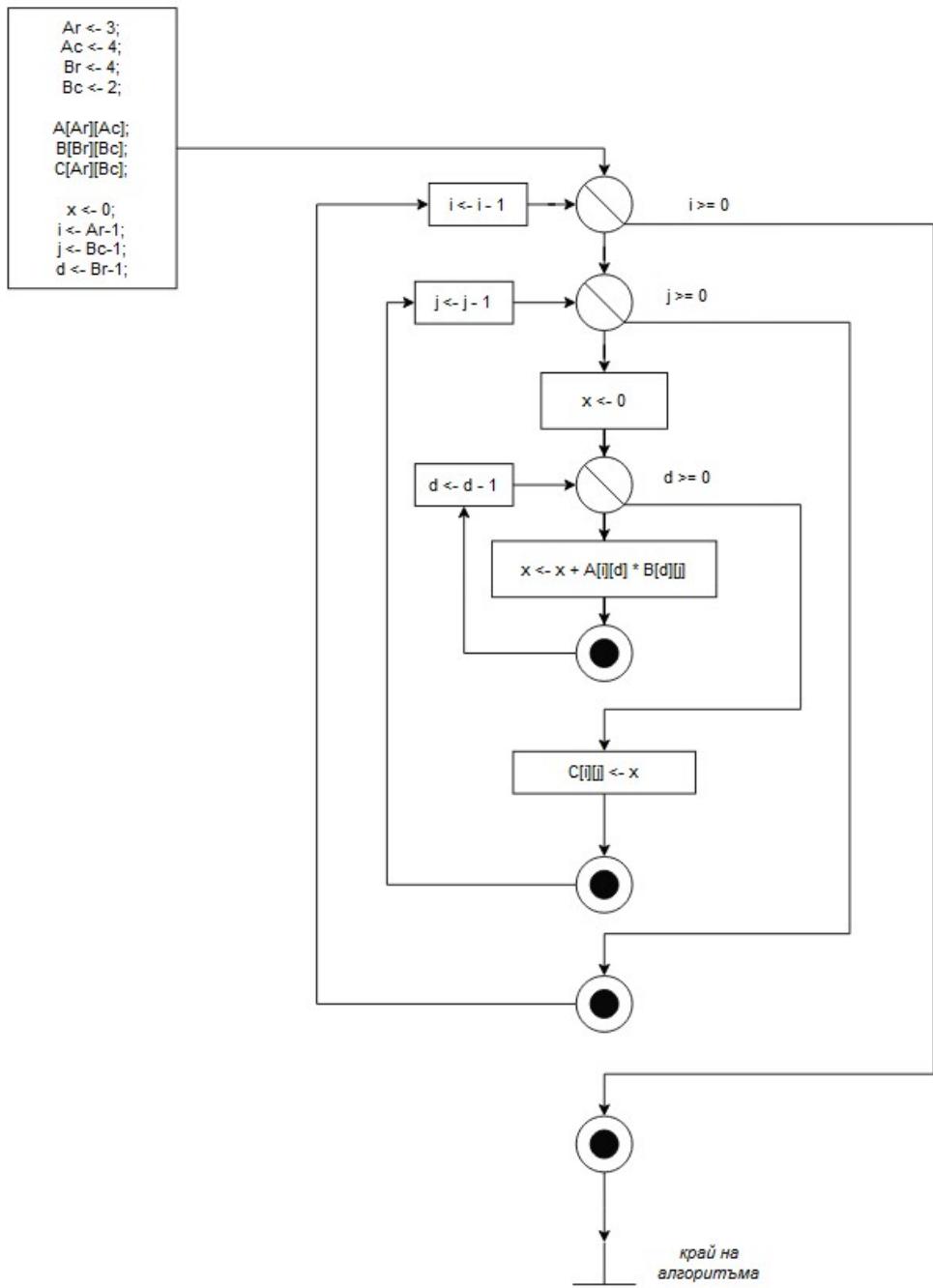
10.10 ОТГОВОР

Изпълнено по-нататък в документа.

3. Съставете

- Опорна схема.
- Алгоритъм
- Програма за обхождане на всички елементи от неглавния диагонал на квадратна матрица.

10.11 ОТГОВОР



Фигура 32: алгоритъм - главен диагонал

(+10.10) Програмен код с алгоритъм водещ се по главния диагонал на матрицата.

```
1 #include <iostream>
2 #include <string>
3
4
5 int main()
{
6     // rows x cols
7
8     const int
9         Ar = 3,
10        Ac = 4,
11
12        Br = 4,
13        Bc = 2;
14
15
16
17
18     int A[Ar][Ac] = {
19                     {6, 8, 9, 4},
20                     {3, 2, 1, 7},
21                     {6, 4, 1, 2}
22                 };
23
24     int B[Br][Bc] = {
25                     {4, 3},
26                     {8, 5},
27                     {9, 1},
28                     {4, 7}
29                 };
30
31
32
33
34     for(int i = 0; i < Ar; i++){
35         for(int j = 0; j < Ac; j++){
36             std::cout << A[i][j] << " ";
37         }
38         std::cout << std::endl;
39     }
40     std::cout << std::endl;
41     for(int i = 0; i < Br; i++){
42         for(int j = 0; j < Bc; j++){
43             std::cout << B[i][j] << " ";
44         }
45         std::cout << std::endl;
46     }
47
48     std::cout << std::endl;
49
50     int C[Ar][Bc]{};
51
```

```

52     for(int i = 0; i < Ar; i++){
53         for(int j = 0; j < Bc; j++){
54             std::cout << C[i][j] << " ";
55         }
56         std::cout << std::endl;
57     }
58 // C pritezhava 4 reda
59 std::cout << "C pritezhava " << Bc << " stulba" << std::endl << std::endl;
60
61     int x{};
62     std::string eq;
63
64     for(int i = 0; i < Ar; i++){
65         for(int j = 0; j < Bc; j++){
66             x=0; //!!!  

67             for(int d = 0; d < Br; d++){
68                 x += A[i][d] * B[d][j];
69                 eq += "(" + std::to_string(A[i][d]) + "*" + std::to_string(B[d][j]);
70                 eq += " + ";
71             }
72             C[i][j] = x;
73             std::cout << "C[" << i << "] [" << j << "] = " << eq << " == " << x << std::endl;
74             eq = "";
75         }
76     }
77
78     std::cout << std::endl;
79     std::cout << "A X B\n->C:\n" << std::endl;
80     for(int i = 0; i < Ar; i++){
81         for(int j = 0; j < Bc; j++){
82             std::cout << C[i][j] << " ";
83         }
84         std::cout << std::endl;
85     }
86 }
87
88
89
90
91
92     return 0;
93 }
```

10.12 ОТГОВОР

Програмен код с алгоритъм водещ се по **обратния** диагонал на матрицата.

```
1 #include <iostream>
2 #include <string>
3
4
5 int main()
6 {
7     // rows x cols
8
9     const int
10    Ar = 3,
11    Ac = 4,
12
13    Br = 4,
14    Bc = 2;
15
16
17
18     int A[Ar][Ac] = {
19                     {6, 8, 9, 4},
20                     {3, 2, 1, 7},
21                     {6, 4, 1, 2}
22                 };
23
24     int B[Br][Bc] = {
25                     {4, 3},
26                     {8, 5},
27                     {9, 1},
28                     {4, 7}
29                 };
30
31
32
33
34     for(int i = 0; i < Ar; i++){
35         for(int j = 0; j < Ac; j++){
36             std::cout << A[i][j] << " ";
37         }
38         std::cout << std::endl;
39     }
40     std::cout << std::endl;
41     for(int i = 0; i < Br; i++){
42         for(int j = 0; j < Bc; j++){
43             std::cout << B[i][j] << " ";
44         }
45         std::cout << std::endl;
46     }
47
48     std::cout << std::endl;
49
50     int C[Ar][Bc]{};
```

```

51
52     for(int i = 0; i < Ar; i++){
53         for(int j = 0; j < Bc; j++){
54             std::cout << C[i][j] << " ";
55         }
56         std::cout << std::endl;
57     }
58 // C pritezhava 4 reda
59 std::cout << "C pritezhava " << Bc << " stulba" << std::endl << std::endl;
60
61     int x{};
62     std::string eq;
63
64     for(int i = Ar-1; i >= 0; i--){
65         for(int j = Bc-1; j >= 0; j--){
66             x=0; //!!!  

67             for(int d = Br-1; d >= 0; d--){
68                 x += A[i][d] * B[d][j];
69                 eq += "(" + std::to_string(A[i][d]) + "*" + std::to_string(B[d][j]);
70                 eq += " + ";
71             }
72             C[i][j] = x;
73             std::cout << "C[" << i << "] [" << j << "] = " << eq << " == " << x << std::endl;
74             eq = "";
75         }
76     }
77
78     std::cout << std::endl;
79     std::cout << "A X B\n->C:\n" << std::endl;
80     for(int i = Ar-1; i >= 0; i--){
81         for(int j = Bc-1; j >= 0; j--){
82             std::cout << C[i][j] << " ";
83         }
84         std::cout << std::endl;
85     }
86
87
88
89
90
91
92     return 0;
93 }
```

ДОМАШНО 11

Краен срок: вторник, 14 май 2024, 09:30 AM

Предадено на: петък, 10 май 2024, 16:14 PM

1. В мудъл 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.

11.1 ОТГОВОР

Задачата е изпълнена по условие - на хартия. Въпреки това, решението е преписано в следващото условие.

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.

11.2 ОТГОВОР

Първият ред се разделя на първия елемент в редицата (в случая на 2):

Целта е да получим единица в горния ляв ъгъл.

$$\left[\begin{array}{ccc|c} 2 & 0 & 1 & 134 \\ 1 & -3 & 2 & 12 \\ 3 & 6 & 5 & -36 \end{array} \right] \sim \left[\begin{array}{ccc|c} 1 & 0 & \frac{1}{2} & 67 \\ 1 & -3 & 2 & 12 \\ 3 & 6 & 5 & -36 \end{array} \right] \quad (7)$$

Вече новият първи ред се изважда от втория, за да получим **новия втори ред**.
Това е част от **зануляването на стълба под единицата**:

$$\left[\begin{array}{ccc|c} 1 & 0 & \frac{1}{2} & 67 \\ 1 & -3 & 2 & 12 \end{array} \right] \sim \left[\begin{array}{ccc|c} 0 & 3 & -\frac{3}{2} & 55 \end{array} \right] \quad (8)$$

Повтаряме процедурата - ред първи се **умножава по 3** и изважда от третия, за да образу-

ваме новия трети ред:

$$\left[\begin{array}{ccc|c} 3 & 0 & \frac{3}{2} & 201 \\ 3 & 6 & 5 & -36 \end{array} \right] \sim \quad (9)$$

$$\sim [0 \ -6 \ -\frac{7}{2} \mid 237]$$

Завършихме с матрицата:

$$\left[\begin{array}{ccc|c} 1 & 0 & \frac{1}{2} & 67 \\ 0 & 3 & -\frac{3}{2} & 55 \\ 0 & -6 & -\frac{7}{2} & 237 \end{array} \right] \quad (10)$$

Преминаваме към обработване ред втори като се стремим се да получим единица на позиция [1][1]. За целта ще разделим реда на 3.

Получаваме:

$$\left[\begin{array}{ccc|c} 1 & 0 & \frac{1}{2} & 67 \\ 0 & 1 & -\frac{1}{2} & 18,33 \\ 0 & -6 & -\frac{7}{2} & 237 \end{array} \right] \quad (11)$$

Новият втори ред умножаваме по -6 и изваждаме от третия:

$$\begin{aligned} \left[\begin{array}{ccc|c} 0 & -6 & 3 & -109,98 \\ 0 & -6 & -\frac{7}{2} & 237 \end{array} \right] \sim \\ \sim [0 \ 0 \ \frac{13}{2} \mid -346,98] \end{aligned} \quad (12)$$

Матрицата с обработени първи и втори ред вече изглежда така:

$$\left[\begin{array}{ccc|c} 1 & 0 & \frac{1}{2} & 67 \\ 0 & 1 & -\frac{1}{2} & 18,33 \\ 0 & 0 & \frac{13}{2} & -346,98 \end{array} \right] \quad (13)$$

Като за последно е нужно само да разделим последният елемент от матрицата на себе си, за да го превърнем в единица:

$$\begin{aligned} \left[\begin{array}{ccc|c} 1 & 0 & \frac{1}{2} & 67 \\ 0 & 1 & -\frac{1}{2} & 18,33 \\ 0 & 0 & \frac{13}{2} & -346,98 \end{array} \right] \sim \\ \sim \left[\begin{array}{ccc|c} 1 & 0 & \frac{1}{2} & 67 \\ 0 & 1 & -\frac{1}{2} & 18,33 \\ 0 & 0 & 1 & \frac{-346,98}{\frac{13}{2}} \end{array} \right] \end{aligned} \quad (14)$$

Матрицата, обработена по метода на **Карл Фридрих Гаус**, придоби следната форма:

$$\left[\begin{array}{ccc|c} 1 & 0 & \frac{1}{2} & 67 \\ 0 & 1 & -\frac{1}{2} & 18,33 \\ 0 & 0 & 1 & -53,38153846 \end{array} \right] \quad (15)$$

Закръгляйки до хилядни, x_3 става:

$$\left[\begin{array}{ccc|c} 1 & 0 & \frac{1}{2} & 67 \\ 0 & 1 & -\frac{1}{2} & 18,33 \\ 0 & 0 & 1 & -53,3846 \end{array} \right] \quad (16)$$

Намираме останалите неизвестни

Знаейки вече стойността на x_3 , се ориентираме към намирането на тази на x_2 :

$$\begin{aligned} 1 * x_2 + \left(\left(-\frac{1}{2} \right) * (-53, 3846) \right) &= 18, 33 \Rightarrow \\ \Rightarrow 1 * x_2 + \left(\left(-\frac{1}{2} \right) * (-53, 3846) \right) - 18, 33 &= 0 \Rightarrow \\ \Rightarrow 1 * x_2 + 26, 6923 - 18, 33 &= 0 \Rightarrow \\ \Rightarrow 1 * x_2 + 8, 3623 &= 0 \Rightarrow \\ \Rightarrow 1 * x_2 = -8, 3623 &\Rightarrow \\ \Rightarrow x_2 = \frac{-8, 3623}{1} &\Rightarrow \end{aligned} \tag{17}$$

$$x_2 = -8, 3623$$

Правим същото и за x_1 :

$$\begin{aligned} 1 * x_1 + 0 * (-8, 3623) + \frac{1}{2} * (-53, 3846) &= 67 \Rightarrow \\ \Rightarrow 1 * x_1 + 0 + (-26, 6923) - 67 &= 0 \Rightarrow \\ \Rightarrow 1 * x_1 + (-26, 6923) - 67 &= 0 \Rightarrow \\ \Rightarrow 1 * x_1 - 26, 6923 - 67 &= 0 \Rightarrow \\ \Rightarrow 1 * x_1 - 93, 6923 &= 0 \Rightarrow \\ \Rightarrow 1 * x_1 = 93, 6923 &\Rightarrow \\ \Rightarrow x_1 = \frac{93, 6923}{1} &\Rightarrow \end{aligned} \tag{18}$$

$$x_1 = 93, 6923$$

Проверка

$$\begin{aligned}
 2 * x_1 + 0 * x_2 + 1 * x_3 &=? 134 \\
 2 * 93,6923 + 0 * (-8,3623) + 1 * (-53,3846) &=? 134 \\
 2 * 93,6923 + 0 + (-53,3846) &=? 134 \\
 2 * 93,6923 - 53,3846 &=? 134
 \end{aligned} \tag{19}$$

$$187,3846 - 53,3846 = 134$$

$$\begin{aligned}
 1 * x_1 + ((-3) * x_2) + 2 * x_3 &=? 12 \\
 1 * 93,6923 + ((-3) * (-8,3623)) + 2 * (-53,3846) &=? 12 \\
 93,6923 + 25,0869 - 106,7692 &=? 12 \\
 118,7792 - 106,7692 &=? 12
 \end{aligned} \tag{20}$$

$$12,01 \approx 12$$

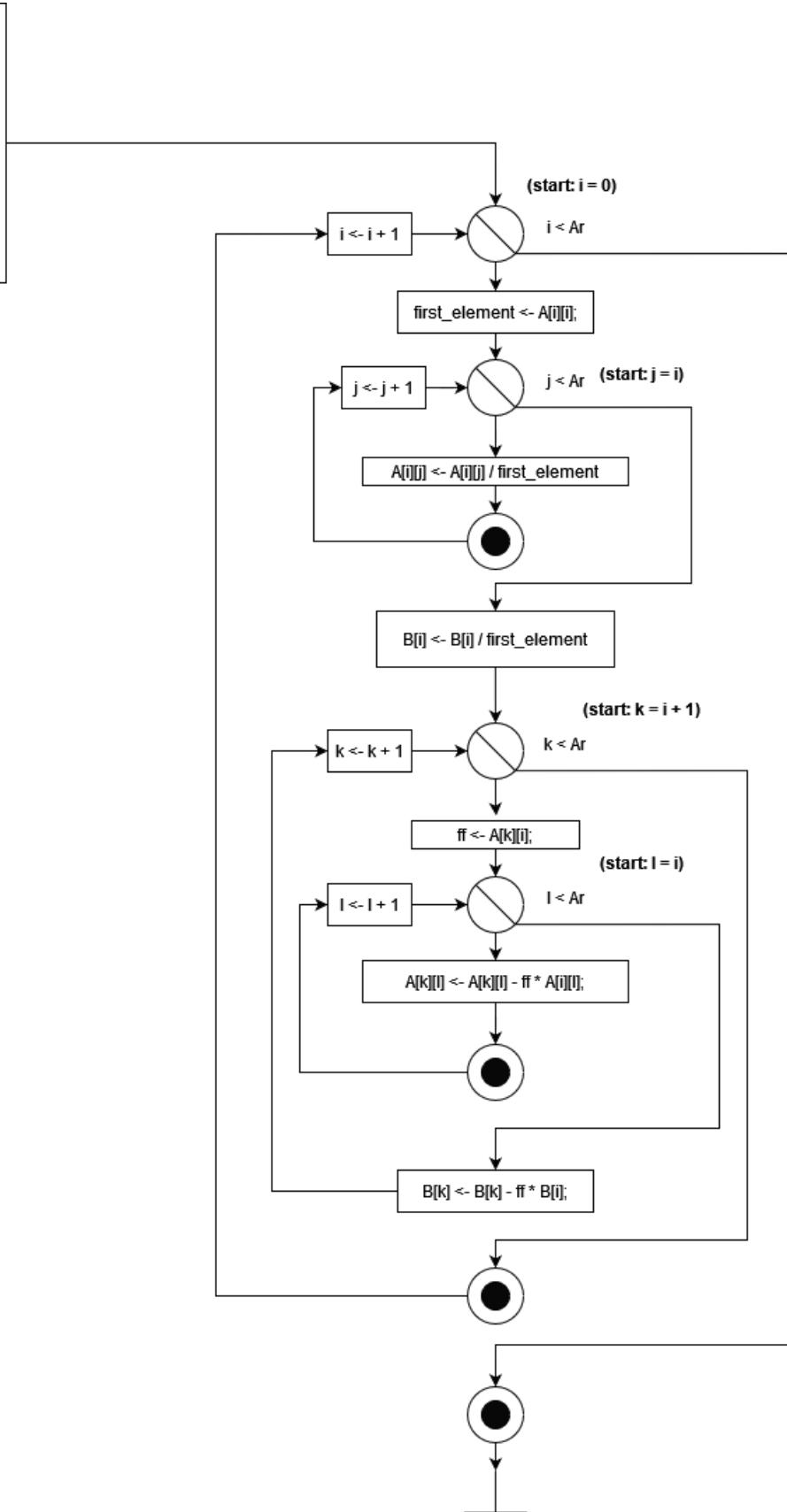
$$\begin{aligned}
 3 * x_1 + 6 * x_2 + 5 * x_3 &=? - 36 \\
 3 * 93,6923 + 6 * (-8,3623) + 5 * (-53,3846) &=? - 36 \\
 281,0769 + (-50,1738) + (-266,923) &=? - 36 \\
 281,0769 - 50,1738 - 266,923 &=? - 36 \\
 230,9031 - 266,923 &=? - 36
 \end{aligned} \tag{21}$$

$$-36,0199 \approx -36$$

```

Ar <- 3
Ac <- 3
Br <- 3
B[Br];
first_element <- 0;
ff <- 0;
i <- 0;
j <- 0;
k <- 0;
l <- 0;

```



Фигура 33: опорна схема на използвания алгоритъм

```
1 #include <iostream>
2 #include <iomanip>
3
4 int main(){
5
6     const int Ar = 3, Ac = 3;
7     const int Br = 3;
8
9     if(Ac != Br){
10         std::cout << "Cannot perform calculations with these values.";
11         return 1;
12     }
13
14     double** A = new double*[Ar];
15     for(int i = 0; i < Ar; i++){
16         A[i] = new double[Ac];
17         for(int j = 0; j < Ac; j++){
18             std::cout << "A[" << i << "][" << j << "] = ";
19             std::cin >> A[i][j];
20         }
21         std::cout << std::endl;
22     }
23
24     double* B = new double[Br];
25     for(int i = 0; i < Br; i++){
26         std::cout << "B[" << i << "] = ";
27         std::cin >> B[i];
28         std::cout << std::endl;
29     }
30
31     for(int i = 0; i < Ar; i++){
32         for(int j = 0; j < Ac; j++){
33             std::cout << std::setw(7) << A[i][j];
34         }
35         std::cout << " | " << B[i] << std::endl;
36     }
37     std::cout << std::endl;
38
39
40 // THE ALGORITHM
41 double first_element{};
42 double ff{};
43 for(int i = 0; i < Ar; i++){
44     first_element = A[i][i];
45
46     for(int j = i; j < Ar; j++){
47         A[i][j] /= first_element;
48     }
49
50     B[i] /= first_element;
51
52     for(int k = i + 1; k < Ar; k++){
```

```

53     ff = A[k][i];
54     for(int l = i; l < Ar; l++){
55         A[k][l] = A[k][l] - ff * A[i][l];
56     }
57     B[k] = B[k] - ff * B[i];
58 }
59 }
60
61 for(int i = 0; i < Ar; i++){
62     for(int j = 0; j < Ac; j++){
63         std::cout << std::setw(7) << A[i][j];
64     }
65     std::cout << " | " << B[i] << std::endl;
66 }
67 std::cout << std::endl;
68
69 std::cout << "B:" << std::endl;
70 for(int i = 0; i < Br; i++){
71     std::cout << B[i] << " ";
72 }
73
74
75 // mm
76 for(int i = 0; i < Ar; i++){
77     delete[] A[i];
78 }
79
80 delete[] A;
81 delete[] B;
82 A = NULL;
83 B = NULL;
84
85
86 return 0;
87 }

```

```
\hw11>g++ solution.cpp -o .\out\sol
```

```
\hw11>.\out\sol.exe
```

```
A[0][0] = 2.0
```

```
A[0][1] = 0.0
```

```
A[0][2] = 1.0
```

```
A[1][0] = 1.0
```

```
A[1][1] = -3.0
```

```
A[1][2] = 2.0
```

```
A[2][0] = 3.0
```

```
A[2][1] = 6.0
```

```
A[2][2] = 5.0
```

```
B[0] = 134.0
```

```
B[1] = 12.0
```

```
B[2] = -36.0
```

2	0	1		134
1	-3	2		12
3	6	5		-36

1	0	0.5		67
0	1	-0.5		18.3333
0	0	1		-53.3846

```
B:
```

```
67 18.3333 -53.3846
```

Фигура 34: изход от изпълнение

*Извод: Алгоритъмът работи по предназначение, изчислявайки крайните стойности с ко-
то можем да намерим търсените x_1 , x_2 и x_3 . Те съответстват на проведените от мен
изчисления спрямо таблицата, включена в заданието от домашната работа.*