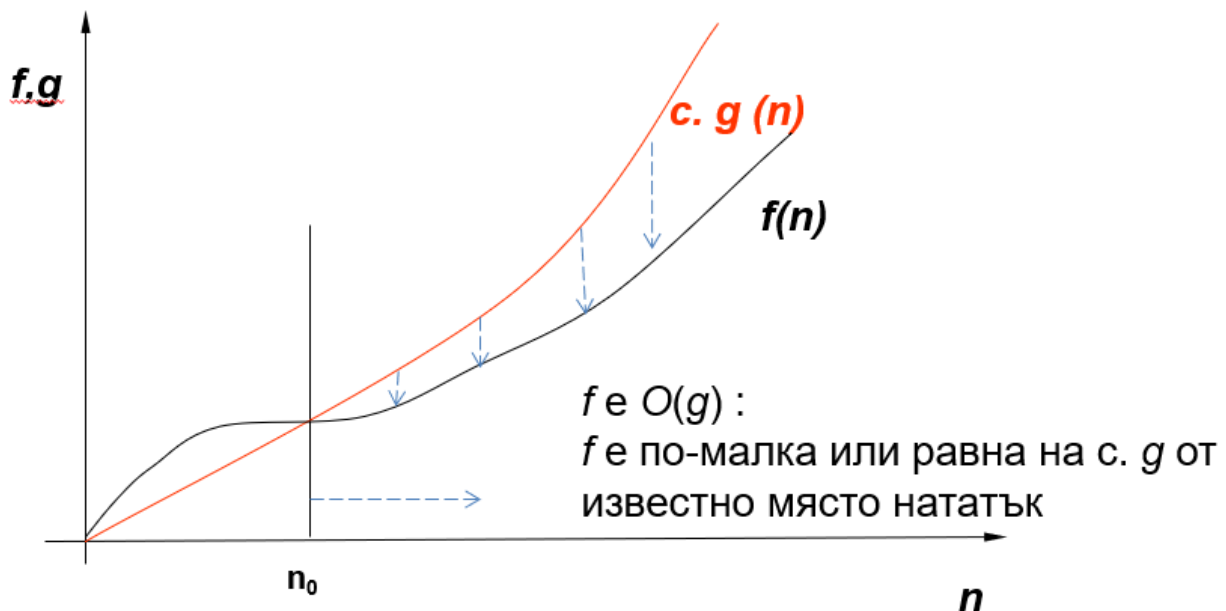


1. Запишете дефинициите на асимптотичните означения O , Ω и θ така, както са описани в литературата и обяснени на лекции. Придружете дефинициите с графики и означения, съответстващи на дефинициите. Опишете с по едно изречение след всяка дефиниция, в свободен текст, как я разбирате.

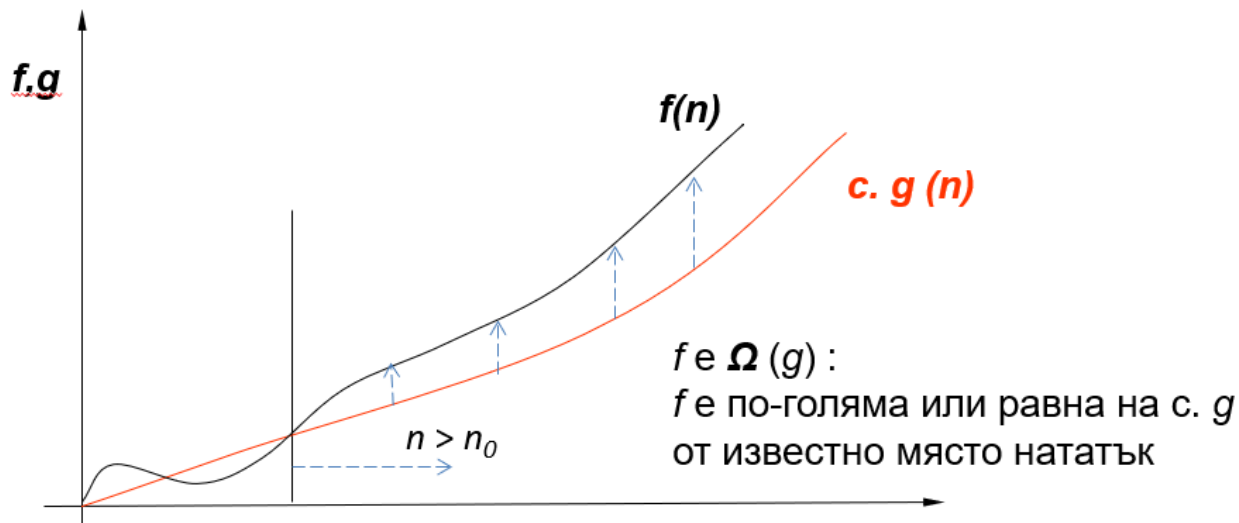
Искаме да оценим функцията f посредством „еталонната“ функция g , като тя може да попадне в тези три класа O -главно, тита, омега

O -главно : Ако съществуват две положителни константи n_0 и c такива, че (от даден момент нататък) за всички $n > n_0$ функцията $f(n) \leq c \cdot g(n)$ (функцията която оценяваме да натиска отгоре $f(n)$) то асимптотично(гранично/почти) $f(n)$ е от класа O . Тя от гледна точка на алгоритми символизира най-лошия случай.



Ω -омега

Аналогично, за други две функции f и g ако съществуват такива две положителни константи n_0 и c така че за всички числа $n > n_0$ функцията $f(n)$ да е по-голяма от $c \cdot g(n)$ съответно $\Rightarrow \Omega(g(n))$ или още казано $g(n)$ е от класа тита. От алгоритмична гледна точка това е най-добрият случай.

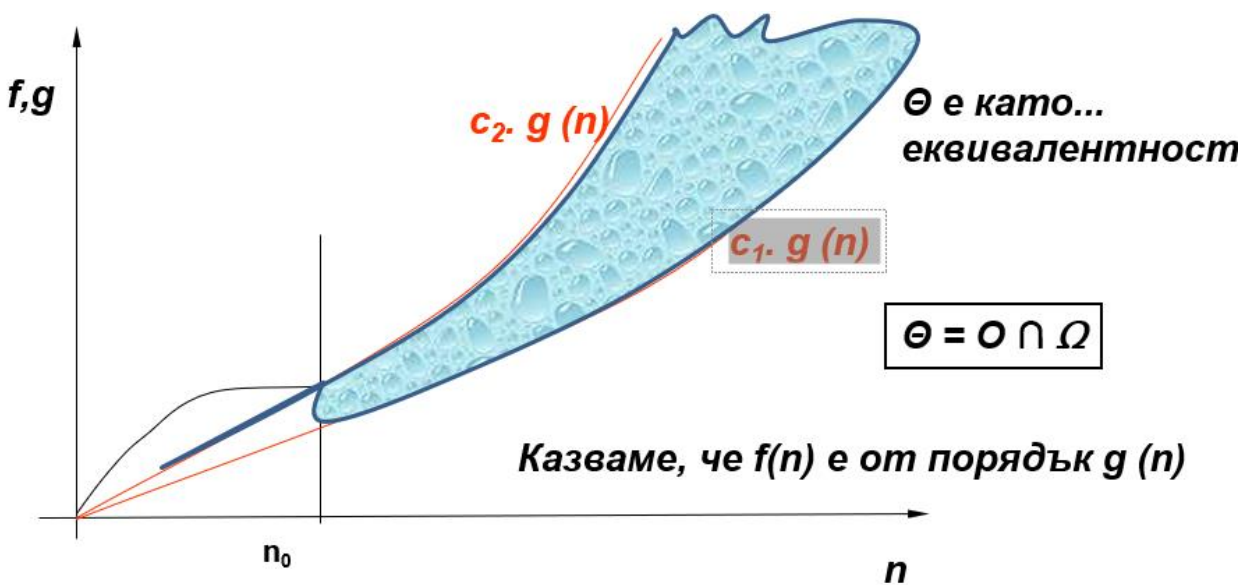


Θ - тита

Една функция е от клас Омега тогава когато съществуват ТРИ положителни константи , n_0 , c_1 и c_2 такива че за всяко $n > n_0$ (от даден момент нататък) $c_1 \cdot g(n) \leq f(n) \leq c_2 \cdot g(n)$

По точно казано когато омега и О-голямо се приложат едновременно или още казано тяхното сечение.

Може да се ограничи между двете $c_2 \cdot g(n)$ и $c_1 \cdot g(n)$



- Запишете какво наричаме сложност на алгоритъм по време. Означете ясно какво има на осите на графиката на $T(n)$.

Сложност на алгоритъма по време е това каква е зависимостта между броя на стъпките „ $T(n)$ “ според това каква е големината на входа (колко е голям).

На графиката има вертикално означено броя на стъпките а хоризонтално големината на входа.

3. Съставете алгоритъм и програма (може да използвате псевдокод) за сортиране чрез вмъкване, като мястото на вмъкване се търси дихотомично. Съставете опорна схема (масивът и какво се прави с него), означете имената на променливите, които ползвате.

java

```
void insertionBinary(int arr[], long arraySize){

    for(int i = 1; i <= arraySize-1 ; i++){ // обхождаме всяко едно число в масива почвайки от позиция 1 (2 елемент)

        int current = arr[i]; // слагаме текущия елемент

        left = 0; // най лявата част от масива (преди да се дели binary

        right = i; // дясната част на масива (съответно големината

        while(left <= right){ // след всяко разглеждане и минаване на while търсенето на мястото
намалява наполовина ,докато лявата страна не стане равна на дясната или съответно не остане 1 елемент

            mid = (left+ right)/2; // взима се средата на масива

            if(arr[mid] == current && arr[mid]<current<arr[mid+1] && arr[mid-1]<current<arr[mid]){ //
проверява се дали стойността на средното(mid) поле в масива е равно на текущото число или е между само и единствено две
(едно по-голямо, едно по-малко)

                for(int j = mid ; j<i-1 ; j++){ //ако е вярно съответно се обхожда отново целия масив от
намерената позиция до края на подредения масив като

                    arr[j+1] = arr[j]; //...се сменят местана на елемента в дясно с този в
ляво докато не стигне до края

                }

                arr[mid] = current; // в следствие текущия елемент се слага на средната позиция (още
казано старата позиция на последно сравнения елемент

            }

            if(arr[mid] > current){ // проверява дали стойността е по малка от средния елемент

                right = mid-1; // слага края на масива(right) да е на предишната позиция от
досегашната среда №пример { 1,2,5, <-| 6 ,9 , 2,6}

            }

            if(arr[mid] < current){ //проверява дали стойността е по голяма от средния елемент

                left = mid+1; // слага началото на масива (left) да е на позицията след досегашното
начало №пример { 1,2,5, |-> 6 ,9 , 2,6}

            }

        }

    }

}
```

}

4. Напишете с ваши думи кога този алгоритъм работи с най-малко сравнения на стойности, колко са те и защо са толкова. Напишете и кога този алгоритъм прави най-много сравнения на стойности, колко са те и защо са толкова.

Ако алгоритъма се подобри така че да проверява дали числото което сравняваме е по-голямо от най-голямото в подредения масив или съответно по-малко от най малкия елемент (още if statement-и) и директно поставя елемента на тази позиция , то тогава алгоритъма би се държал като полином от 1ва степен „ n-1 “ повторения което е най добрия случай, но засега може да я оценим с еталонната функция $\log_2(n*n) = n$ при вход от масив съдържащ поне един елемент. Тя се държи най добре $\log_2(n)$, а най-зле като n-1 като тя е от клас Тита.

Георги Филев – F104081

Домашно 1.Б 2021-22есен.

Използвайки дадения в презентацията Lab 1 - 2020 примерен код, реализирайте измерване на времето на изпълнение на алгоритмите за сортиране пряка селекция, пряко вмъкване и мехурче. Измерете времената на всеки от тях за 10 000, 20 000, ... 100 000 случайно наредени елементи и ги въведете в електронна таблица, за да изчертаете графиката за тях.

```
#include <iostream>
#include <stdlib.h>
#include <time.h>
#include <sys/types.h>
using namespace std;
```

```
void swap(long int *xp, long int *yp)
```

```
{
```

```

    int temp = *xp;

    *xp = *yp;

    *yp = temp;

}

// A function to implement bubble sort
void bubbleSort(long int arr[], long int n)
{
    int i, j;

    for (i = 0; i < n - 1; i++)

        // Last i elements are already in place
        for (j = 0; j < n - i - 1; j++)
            if (arr[j] > arr[j + 1])
                swap(&arr[j], &arr[j + 1]);

}

//
void selectionSort(long int arr[], long int n)
{
    int i, j, min_idx;

    // One by one move boundary of unsorted subarray
    for (i = 0; i < n - 1; i++)
    {
        // Find the minimum element in unsorted array
        min_idx = i;
        for (j = i + 1; j < n; j++)
            if (arr[j] < arr[min_idx])
                min_idx = j;

        // Swap the found minimum element with the first element

```

```

        swap(&arr[min_idx], &arr[i]);
    }
}

//
void insertionSort(long int arr[], long int n)
{
    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;
    }
}

/* Function to print an array */
void printArray(long int arr[], long int n)
{
    int i;
    for (i = 0; i < n; i++)
        cout << arr[i] << " ";
    cout << endl;
}

```

```

int main()
{
    time_t t0, t1;
    clock_t c0, c1;
    srand(time(NULL));
    const long int ARRAY_SIZE = 10000;
    long int arr[ARRAY_SIZE];
    for (int i = 0; i < ARRAY_SIZE; i++) {
        arr[i] = rand() % 100 + 1;
    }

    t0 = time(NULL);
    c0 = clock();

    bubbleSort(arr, ARRAY_SIZE);
    c1 = clock();
    t1 = time(NULL);
    //cout << "Sorted array bubble: \n";
    //printArray(arr, n);
    cout << "elapsed wall clock time: " << ((long)(t1 - t0)) << endl;
    cout << "elapsed wall clock time:" << (float)(c1 - c0) / CLOCKS_PER_SEC
<< endl;
    //cout << "is sorted" << std::is_sorted(a, ARRAY_SIZE);

    //selectionSort(arr, ARRAY_SIZE);
    //cout << "Sorted array selection: \n";
    //printArray(arr, n);

    //insertionSort(arr, ARRAY_SIZE);
    //cout << "Sorted array insertion: \n";
    //printArray(arr, n);

```

}

Bubble sort /time

The screenshot shows a C++ development environment with two windows. The left window displays the source code for three sorting algorithms: bubbleSort, selectionSort, and insertionSort. The right window shows the program's output, which includes timing information for each algorithm and a confirmation message from the operating system.

main.cpp

```
73 }  
74  
75 int main()  
76 {  
77     time_t t0, t1;  
78     clock_t c0, c1;  
79     srand(time(NULL));  
80     const long int ARRAY_SIZE = 1000;  
81     long int arr[ARRAY_SIZE];  
82     for (int i = 0; i < ARRAY_SIZE; i++)  
83         arr[i] = rand() % 1000;  
84  
85     t0 = time(NULL);  
86     c0 = clock();  
87  
88     bubbleSort(arr, ARRAY_SIZE);  
89     c1 = clock();  
90     t1 = time(NULL);  
91     cout << "Sorted array by bubble sort\n";  
92     printArray(arr, ARRAY_SIZE);  
93     cout << "Elapsed wall clock time: " << t1 - t0 << "\n";  
94     cout << "Elapsed wall clock time: " << (c1 - c0) / CLK_TCK << "\n";  
95  
96     //selectionSort(arr, n);  
97     cout << "Sorted array by selection sort\n";  
98     printArray(arr, n);  
99     cout << "Elapsed wall clock time: " << t1 - t0 << "\n";  
100    cout << "Elapsed wall clock time: " << (c1 - c0) / CLK_TCK << "\n";  
101  
102    //insertionSort(arr, n);  
103    cout << "Sorted array by insertion sort\n";  
104    printArray(arr, n);  
105    cout << "Elapsed wall clock time: " << t1 - t0 << "\n";  
106    cout << "Elapsed wall clock time: " << (c1 - c0) / CLK_TCK << "\n";  
107 }
```

C:\Users\Georgj\Favorites\domashno2\bin\Debug\domashno2.exe

```
Process returned 0 (0x0)   execution time : 1.423 s  
Press any key to continue.
```

Selection sort /time

The screenshot displays a C++ development environment with a source file named main.cpp and its corresponding output.

```

79 srand(time(NULL));
80 const long int ARRAY_SIZE = 10000;
81 long
82 for(
83     a
84 }
85
86 t0 =
87 c0 =
88
89 bubb
90 cl =
91 tl =
92 //so
93 //pr
94
95 //so
96
97 select
98 cout
99 print
100 cout
101 cout
102
103 //ins
104 //so
105 //pr
106
107 elapsed wall clock time: 0
108 return elapsed wall clock time:0.286

```

The output window at the bottom indicates the program executed successfully:

```

Process returned 0 (0x0)   execution time : 1.586 s
Press any key to continue.

```


Insertion sort /time

```
11 long int arr[ARRAY_SIZE];
12 for (int i = 0; i < ARRAY_SIZE; i++)
13     arr[i] = rand() % 100 + 1;
14
15 t0 = time(NULL);
16 c0 = clock();
17
18 bubbleSort(arr, ARRAY_SIZE);
19 c1 = clock();
20 t1 = time(NULL);
21 //cout << "Sorted array bubbleSort\n";
22 //printArray(arr, n);
23
24 //cout << "is sorted" << std::endl;
25
26 //selectionSort(arr, ARRAY_SIZE);
27 //cout << "Sorted array selectionSort\n";
28 //printArray(arr, ARRAY_SIZE);
29
30 insertionSort(arr, ARRAY_SIZE);
31 cout << "Sorted array insertionSort\n";
32 printArray(arr, ARRAY_SIZE);
33 cout << "elapsed wall clock time bubbleSort: " << t1 - t0 << endl;
34 cout << "elapsed wall clock time insertionSort: " << t1 - t0 << endl;
35
36 return 0;
```

Output:

```
elapsed wall clock time: 0
elapsed wall clock time: 0.287
Process returned 0 (0x0)   execution time : 1.442 s
Press any key to continue.
```

2021 Структури данни

F104081

Георги Филев

Домашно 2.А. РЕКУРСИЯ

1. Запишете рекурсивните дефиниции на най-голям общ делител, дърво и факториел, както са дадени в лекциите.
GCD(най-голям общ делител) на числата A и B които са естествени е равно на числото B ако B се нанася цяло число пъти в A, $A = B \cdot x + \text{остатък}$

Fct на числото n е числото n-1 факториел умножено по n.

Граф(дърво) е това, което се състои от *краен брой дървета D*, всяко от които е свързано с по едно ребро с един (общ за всички свързвания) възел v.

2. Запишете кои са етапите на рекурсивно изпълнение на програмата (от машина) и ги подкрепете със схема.

Условие за дъно. Най-напред се формулира условието α - дъно, осигуряващо възможност на рекурсивния процес да спре да потъва

Лявата част на рекурсивната дефиниция помага да се определят **входните параметри** на рекурсивната процедура.

Точка 2 на рекурсивната дефиниция служи за ориентир при организирането на обмяна на информация между копията.

3. Развийте на ръка схема на рекурсивно изпълнение на Евклидовия алгоритъм, подобна на тази долу. Числото А образувайте от последните четири цифри от факултетния ви номер и числото В - първите две цифри от вашето ЕГН.

$$A = x \cdot B + \text{остатък}$$

$$4081 = 94 \cdot 43 + 39$$

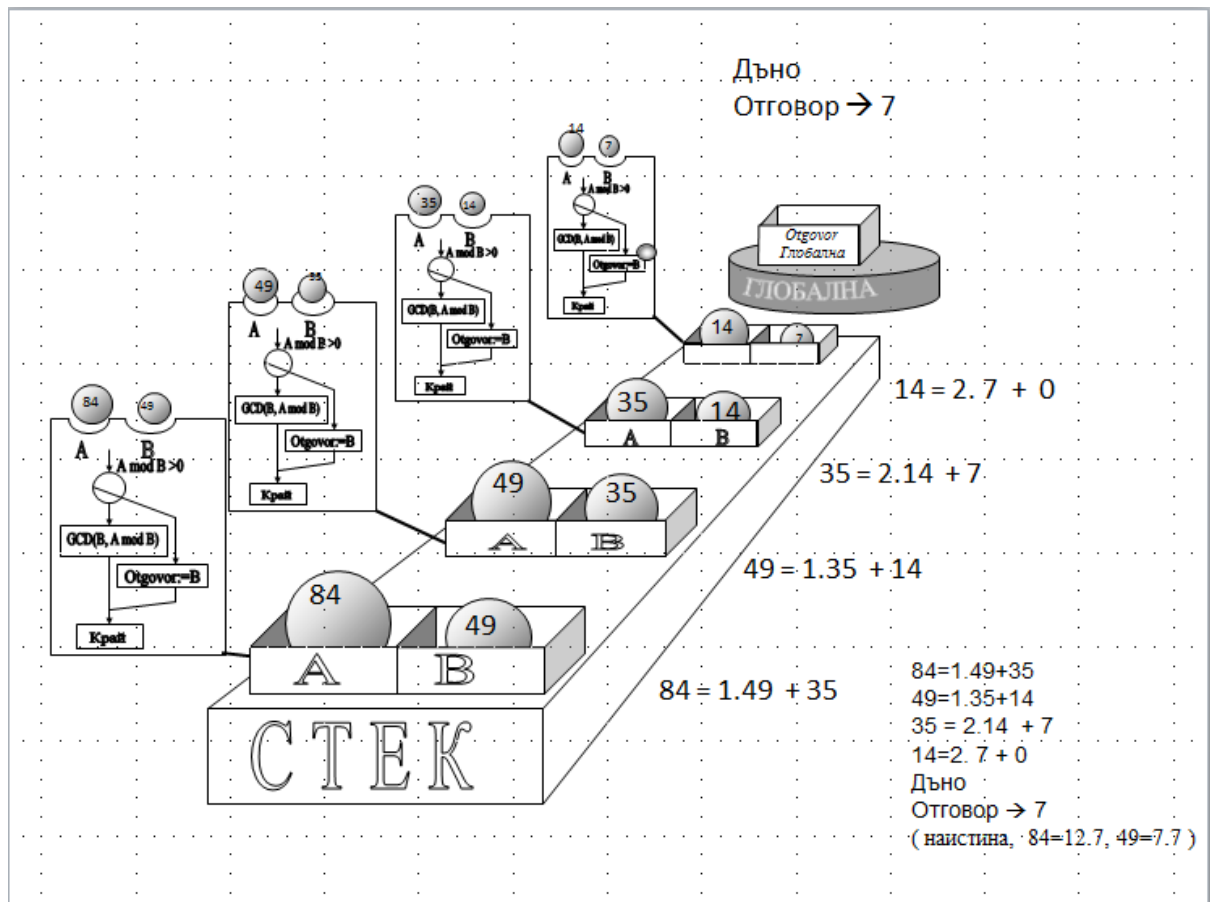
$$43 = 1 \cdot 39 + 4$$

$$39 = 9 \cdot 4 + 3$$

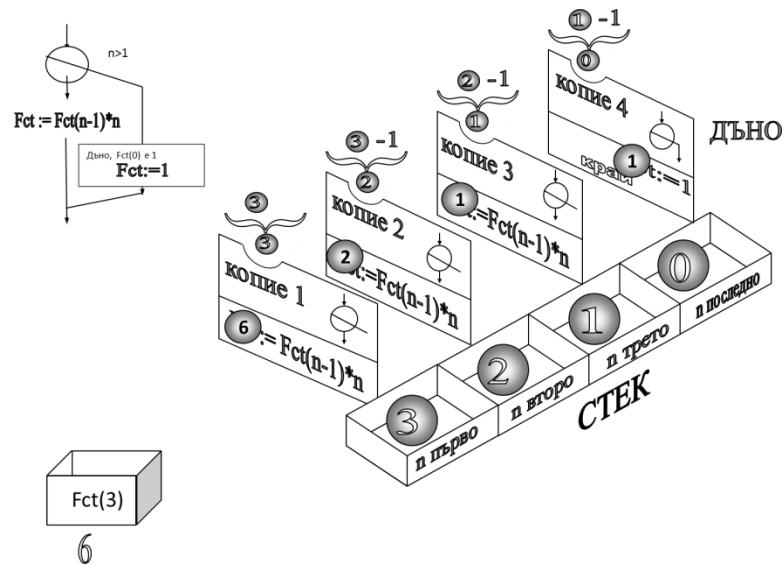
$$4 = 1 \cdot 3 + 1$$

$$3 = 3 \cdot 1 + 0$$

⇒ Най-голям общ делител -> 1



4. Развийте на ръка схема на рекурсивно изпълнение на функция, изчисляваща факториел на числото образувано от последната цифра на вашия факултетен номер. Ако то е нула или едно, приемете числото 6. Използвайте за опора схемата, дадена в лекциите и по-долу, като означите със стрелки как става преносът и изчислението на „изплуване“.



```

Int Fct(int n){
    If(n=1){
        Return 1;
    }
    Else{
        Return Fct(n.(n-1));
    }
}

```

$6 * (6-1) * (6-2) * (6-3) * (6-4) * (6-5)$

ДОМАШНОТО СЕ ПРЕДАВА С ТОЧКИТЕ НА ЗАДАНИЕТО!

Георги Филев – F104081

Домашно 2Б - лаб, 2021

Задача 1. Реализирайте рекурсивно алгоритъма на Евклид посредством процедура, която записва резултата от изчислението в глобална променлива. Поставете в текста оператори, които да разпечатват на екран локалните променливи за всяко копие за да разпечатате развитието на рекурсивния процес по копия: **а. на потъване и б. на изплуване**. Предайте прогресивен текст и снимка на екран - изхода от изпълнение.

```

void functionEvklid(int A , int B){

    int stoinost = A % B;

    if(stoinost > 0){

        cout<< "potuvane  " << "a ="<<A << " b = " << B<< endl;

        functionEvklid(B , stoinost);

        cout<< "izpluvane " << "a ="<<A << " b = " << B<< endl;

    }

    else{

        cout<<"duno"<<endl;

        globalValue = B;

    }

}

```

Задача 2. Отново Евклид рекурсивно, но с връщане на резултата от изчисленията с Return на изплуване, оформено като функция (int). Премежете и сравнете времето на изпълнение между тази и реализацията по задача 1, като поставите за вход и на двете едни и същи стойности, които да са две съседни числа от редицата на Фибоначи (по-големички). Анализирайте резултата от сравнението на времената и го обяснете с едно изречение. Предайте програмен текст и снимка на екран - изхода от изпълнение.

```

int euclid(int A, int B){

    int stoinost = 0;

    if(A % B > 0){

        cout<< "izpluvane " << "a ="<<A << " b = " << B<< endl;

        stoinost = euclid(B , A % B);

        cout<< "potuvane  " << "a ="<<A << " b = " << B<< endl;

    }

    else{

        cout<<"duno " << B<<endl;

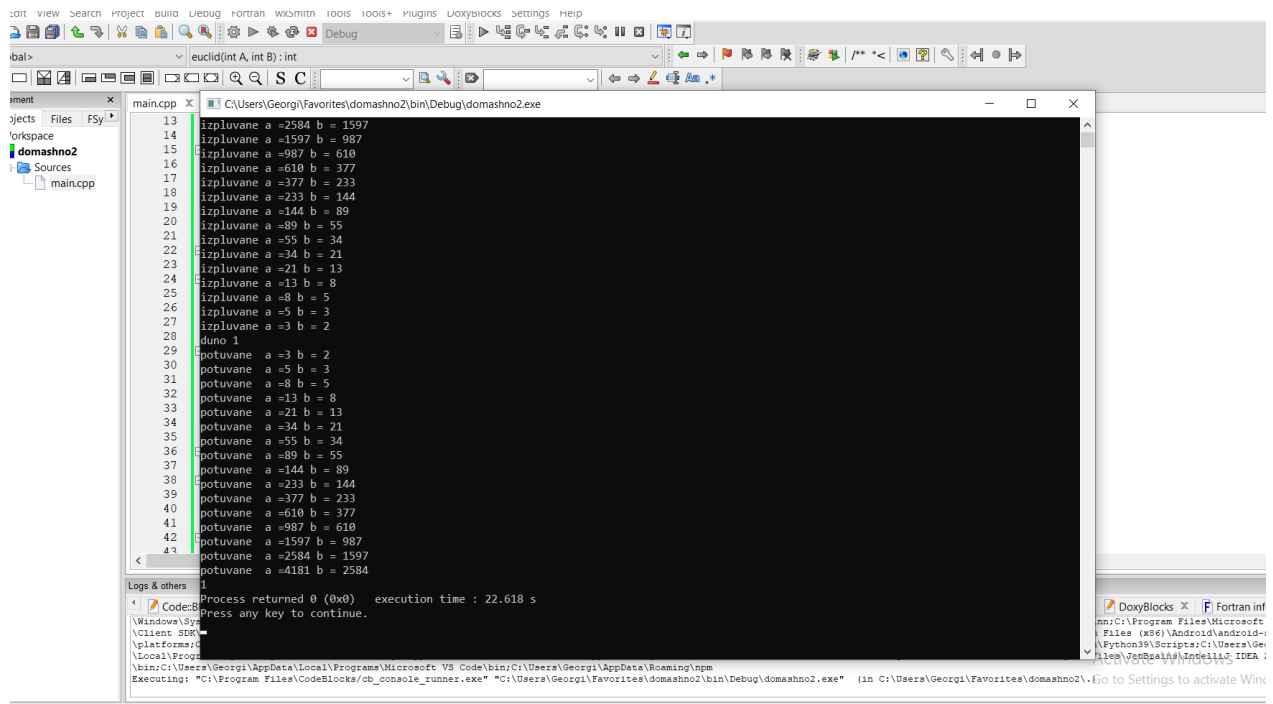
        stoinost =B;

    }

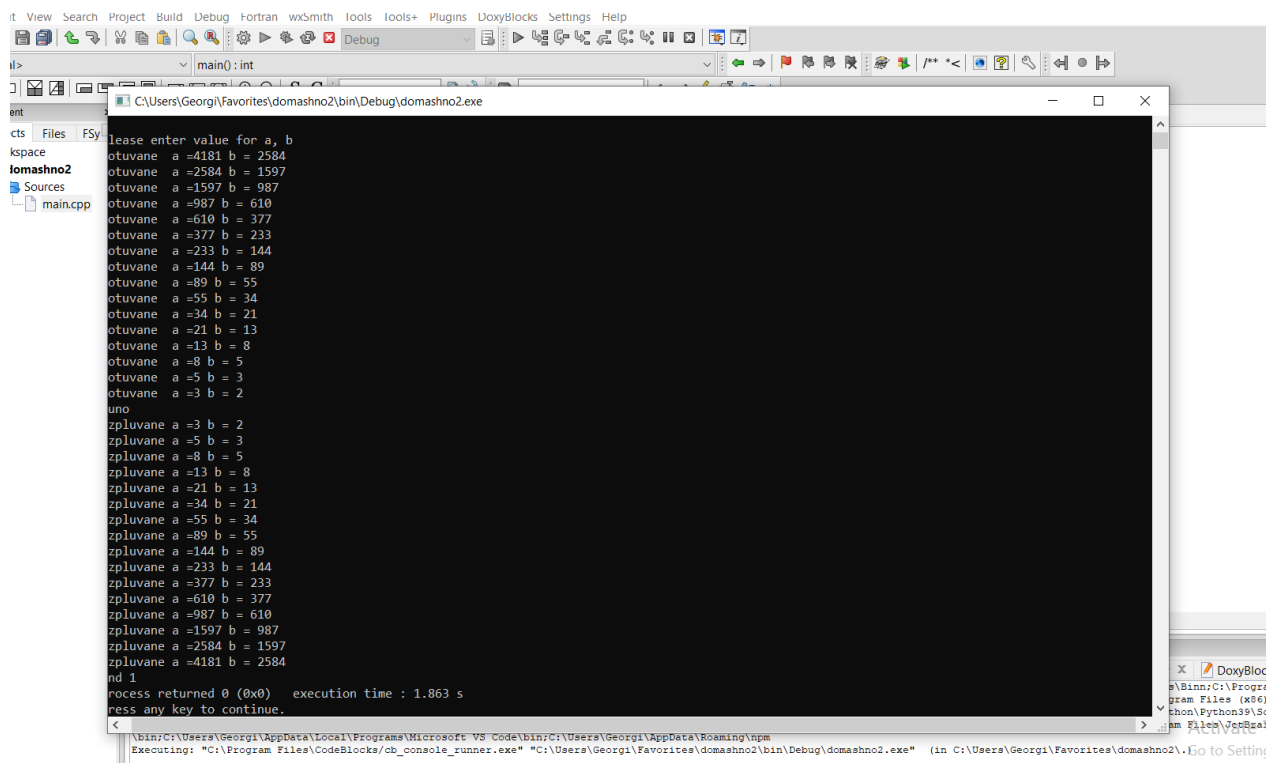
    return stoinost;

}

```



22sec



От първа задача със Void – 1.8 секунда

Задача 3. Факториел - реализирайте рекурсивна функция и разпечатайте локалните на потъване и на изплуване. Предайте програмен текст и снимка на екран - изхода от изпълнение.

```
int fac(int number){
```

```

int gosho = 0;

if(number){

    cout<< "Duno" <<endl;

    gosho =1;

}

else{

    cout<<"potuvam" << "number " << number << endl;

    gosho = fac(number -1)*number;

}

}

```

```

C:\Users\Georgi\Favorites\domashno2\bin\Debug\domashno2.exe
Please enter value for a, b
a=5
end potuvamnumber 5
potuvamnumber 4
potuvamnumber 3
potuvamnumber 2
potuvamnumber 1
Duno
izpluvame! n = 1 smetka = 1
izpluvame! n = 2 smetka = 2
izpluvame! n = 3 smetka = 6
izpluvame! n = 4 smetka = 24
izpluvame! n = 5 smetka = 120
120
Process returned 0 (0x0)   execution time : 0.532 s
Press any key to continue.

```

Задача 4. Дихотомично търсене, програма. Рекурсивно, естествено. Предайте програмен текст и снимка на екран - изхода от изпълнение при търсене на стойност, налична в масива и на стойност, която не е записана в масива.

```

int dihotomic(int koi, int nachalo , int krai){

    int iBeg= nachalo, iMid , iEnd = krai;

    iMid = (iBeg + iEnd)/2;

    if(koi == arr[iMid]){

        cout<<"namerih go! Duno!"<< endl;

```

```

        return iMid;
    }

    else if(iBeg >= iEnd){

        cout<<"nqma go! Duno!"<<endl;

    }

    else{

        if(koi > arr[iMid]){

            iBeg = iMid + 1;

        }

        else {

            iEnd = iMid -1;

        }

        return dihotomic(koi ,iBeg , iEnd);

    }

}

```

The screenshot shows a C++ IDE with a file named `main.cpp` and a console window. The code implements a binary search function `dihtomic` (sic) that searches for a value `koi` in an array `arr` between indices `iBeg` and `iEnd`. The console output shows the program running and finding the value 4 at index 4.

```

main.cpp
43     cout<<"potuyam" << "number " << number << endl;
44     gosho = fac(number -1)*number;
45     cout << "izpluyame!" << "n = " << number << endl;
46
47     return gosho;
48
49
50
51 int dihtomic(int koi, int nachalo , int krai){
52     int iBeg= nachalo, iMid , iEnd = krai;
53     iMid = (iBeg + iEnd)/2;
54     if(koi == arr[iMid]){
55         cout<<"namerih go! Duno!"<< endl;
56         return iMid;
57     }
58     else if(iBeg >= iEnd){
59         cout<<"nqma go! Duno!"<<endl;
60     }
61     else{
62         if(koi > arr[iMid]){
63             cout<<"maha nachaloto"<<endl;
64             iBeg = iMid + 1;
65         }
66         else {
67             cout<<"maha kraq"<<endl;
68             iEnd = iMid -1;
69         }
70         return dihtomic(koi ,iBeg , iEnd);
71     }
72 }
73 int main()

```

```

C:\Users\Georgi\Favorites\domashno2\bin\Debug\domashno2.exe
Please enter a value to search:
4
maha kraq
maha nachaloto
maha kraq
namerih go! Duno!
4
Process returned 0 (0x0)   execution time : 1.944 s
Press any key to continue.

```

2021 Структури данни

Георги Филев F104081

Домашно ЗА. Двуклонова Рекурсия

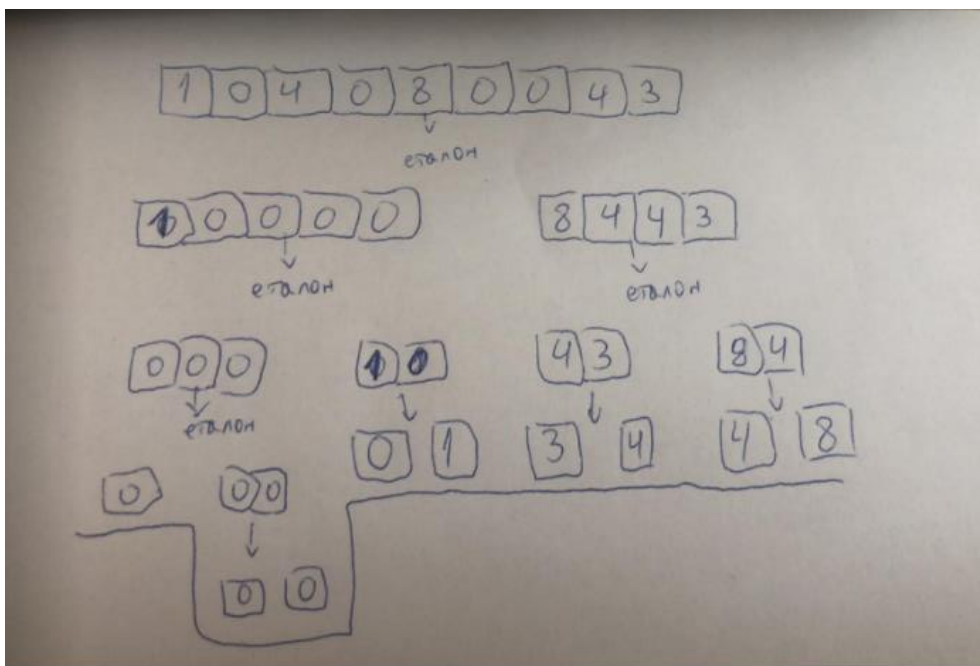
1. Сортиране по Дялове. Запишете рекурсивна дефиниция на нареден масив, както е дадена в лекциите. Един масив е нареден ако се състои от един елемент който е „нареден“, $n=1$, И аналогично нареден масив от n елемента е такъв който се състои от 2 залепени „наредени“ масива (ляв и десен) за всеки елемент на левия масив който е по-малък от десния.
2. Съставете следния седем-местен масив:

	FN					ЕГН първи 4				
Масив										
		1	2	3		4	5	6	7	

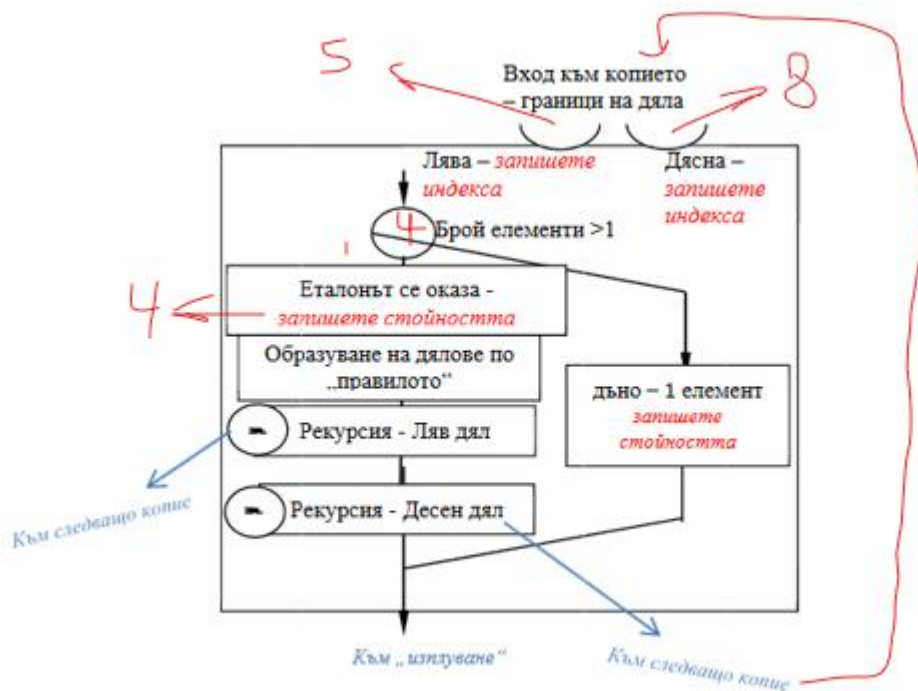
104080043

Проиграйте **принципната схема** на алгоритъма Сортиране по Дялове за получения масив, като използвате схема подобна на тази от лекциите. Проиграйте разделянето на масива на „потъване“, като на всяка стъпка от илюстрираното потъване приемате като еталон за сравнение един **произволен** елемент от разделяния масив. Това, че ЕГН-то на мнозина започва с нули не пречи по никакъв начин на работата.

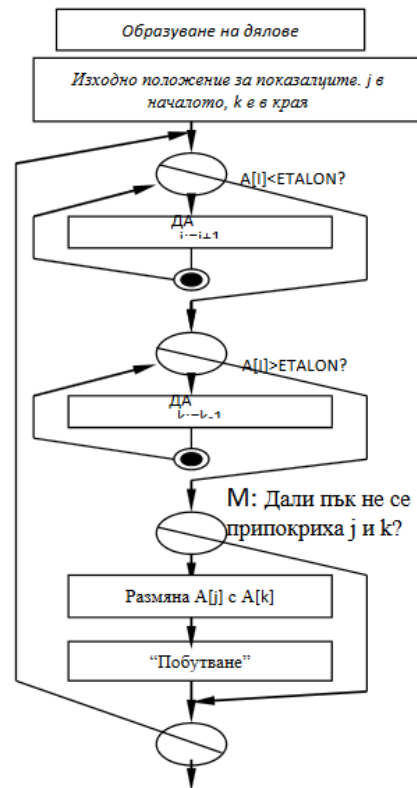




3. Съставете обща схема на развитието на този рекурсивен процес като програма, т.е. всички повикани копия за вашите данни. Изберете за еталон средата на масива. Едно копие да изглежда така:



4. Проиграйте на ръка алгоритъма за образуване на дялове за най-началното, т.е. първото извикване на рекурсивния процес, като ползвате схема, подобна на тази:



Дефинираме коя е лявата и коя е дясната страна.

После първо проверяваме дали лявата страна е по-малка от лявата , след това ако лявата е по-голяма следва че това е Дъно и трябва да обърнем стойностите. Обаче ако не е следва че потъваме и съставяме нашия еталон (примерно някои от средните елементи на масива тоест $\text{arr}[(\text{десен индекс} + \text{ляв индекс}) / 2]$). Така образувахме дялове.

След което докато левия дял и неговия индекс(по точно стойностите на индексите) са по-малки от еталона, алгоритъма минава през тях като ги проверява докато не намери такъв елемент ,че той да е по голям от еталона. След което алгоритъма чака да се случи аналогично същото нещо със десния дял, но той трябва да намери по малка стойност сред индексите спрямо еталона.

След което стойностите на двата отбелязани(„виновни“) индекса се разменят. Съответено алгоритъма продължава докато индекса на левия дял не навлезе в дясната част или обратното.

В последствие се извиква QSort на подмасивите(дяловете) образувани впоследствие на разделянето с цел и те да бъдат подредени.

ДОМАШНОТО СЕ ПРЕДАВА С ТОЧКИТЕ НА ЗАДАНИЕТО

Георги Филев- F104081

2020 Структури данни

Домашно 3Б. Лабораторно упражнение към Сортиране по дялове - трасиране.

Съставете **14-местен масив**, като залепите следните данни едно след друго:

	FN				ЕГН първи 4				
Масив									
	1	2	3	4	5	6	7		

Съставете **програма за сортиране по дялове**, която да ползва за вход този масив (приемете за еталон елемента в средата на обработвания дял) и да разпечатва следното, включително ТЕКСТА:

Начално състояние на масива : ---масив - стойностите--

Копие, граници на дяла от -индекс-- до---индекс

Еталон ---стойност

Размяна на - стойност -- и -- стойност -,

Размяна на - стойност -- и - стойност --,

Размяна на - стойност -- и - стойност -- ...

Край на размените

Потъване наляво

Копие, граници на дяла от - индекс -- до-- индекс -

Еталон --- стойност

Размяна на - стойност -- и -- стойност -,

Размяна на - стойност -- и - стойност --,

Размяна на - стойност -- и - стойност -- ...

Край на размените

Потъване наляво

Дъно.

Изплуване

Потъване надясно

Копие, граници на дяла от -индекс-- до---индекс

Еталон ---стойност

Размяна на - стойност -- и -- стойност -,

Размяна на - стойност -- и - стойност --,

Край на размените

Потъване наляво

Дъно.
Изплуване
Потъване надясно
Дъно.
Изплуване
.....
.....

Крайно състояние на масива : -стойностите, наредени----

Предайте програмен текст и изход от изпълнение.

Решение:

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

void quickSort(int *arr, int li, int di)
{
    int j = li, k = di;
    cout<<"Potuvane !"<<endl;
    cout << "From where to where to enter " << li << " " << di << endl;
    if (di - li >= 1)
    {
        int etalon = arr [(li+di) / 2];
        //Образуване на dqlove
        do{
            while (arr[j] < etalon)
                j++;
            while (arr[k] > etalon)
                k--;
            if (j <= k){
                cout << "Razmqna " << arr[j] << " s " << arr[k] << endl;
                int temp = arr[k];
                arr[k] = arr[j];
                arr[j] = temp;
                k--;
                j++;
            }
        }
        while(j<=k);
    }
}
```

```

    quickSort(arr, li, k);
    quickSort(arr, j, di);
    cout<<" Izpluvane !"<<endl;
}
else
cout << "Duno" << endl;
}
int main()
{
    int arr[14] = {16,1,0,4,0,8,1, 6, 34, 17, 54, 4, 228, 40 };
    quickSort(arr, 0, 14);
    cout<<"Podreden masiw : ";
    for (int i = 0; i < 15; i++){
        cout << arr[i]<<" ";
    }

}

```

The screenshot shows a Windows desktop environment. A Visual Studio Code window is open, displaying a C++ program. The program is a quicksort implementation. The console output shows the array being sorted and the final sorted array. The taskbar at the bottom shows the system clock as 15:43 on 19.10.2021.

```

C:\Users\Georgi\Favorites\QSort\bin\Debug\QSort.exe
Razmqna 4 s 4
From where to where to enter 4 4
Duno
From where to where to enter 5 5
Duno
From where to where to enter 6 6
Duno
From where to where to enter 7 14
Razmqna 54 s 16
Razmqna 228 s 40
From where to where to enter 7 12
Razmqna 34 s 16
Razmqna 17 s 16
From where to where to enter 7 9
Razmqna 16 s 16
From where to where to enter 7 8
Razmqna 8 s 8
From where to where to enter 7 6
Duno
From where to where to enter 8 8
Duno
From where to where to enter 9 9
Duno
From where to where to enter 10 12
Razmqna 34 s 34
From where to where to enter 10 10
Duno
From where to where to enter 12 12
Duno
From where to where to enter 13 14
Razmqna 228 s 54
From where to where to enter 13 13
Duno
From where to where to enter 14 14
Duno
Podreden masiw : 00114468161617344054228
Process returned 0 (0x0)   execution time : 0.062 s
Press any key to continue.

```

Activate Windows
Go to Settings to activate Windows.

15:43
19.10.2021

ДОМАШНОТО СЕ ПРЕДАВА С ТОЧКИТЕ НА ЗАДАНИЕТО!

2021 Структури данни

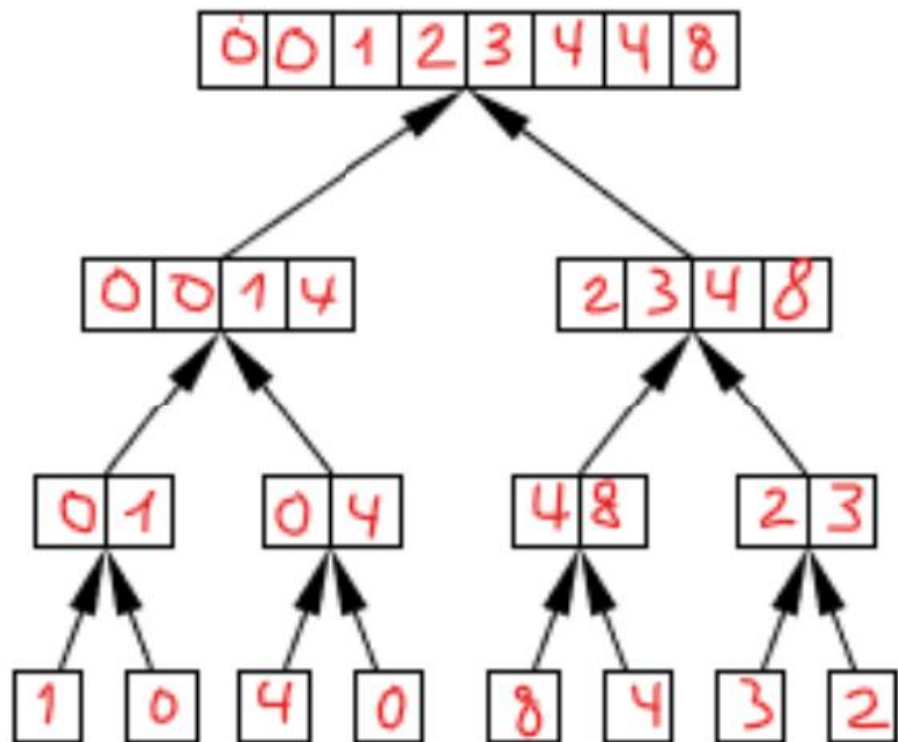
Георги Филев F104081

Домашно 4. Двуклонова Рекурсия

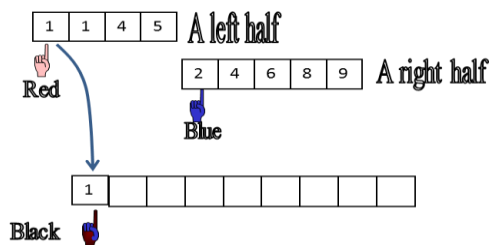
- Сортиране чрез Сливане. Запишете рекурсивна дефиниция на нареден масив, както е дадена в лекциите.
Един масив се води „сортиран“, когато се състои от елементите на два сортирани масива , които са „сортирани“ и пренесени в друг , така че да няма инверсии. При $n=1$ масива е също сортиран.
- Проиграйте принципната схема на рекурсивния алгоритъм за Сортиране чрез Сливане, като използвате за вход следния седем-местен масив:

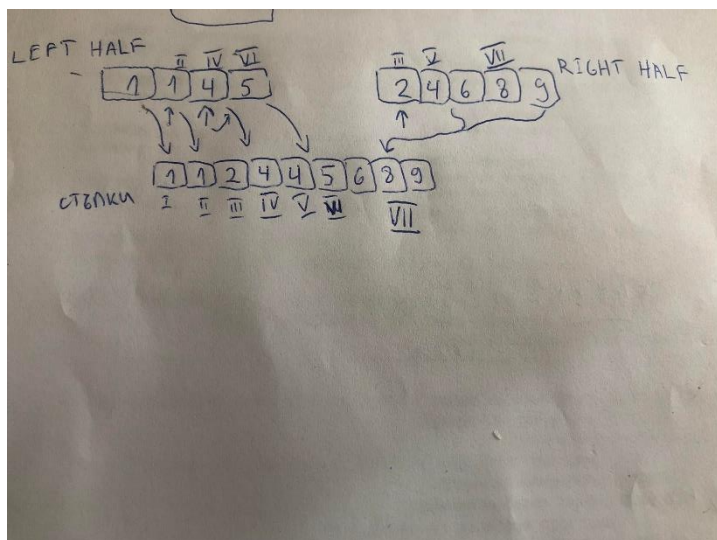
	FN			ЕГН първи 4			
Масив							
	1	2	3	4	5	6	7

- Съставете схема подобна на тази от лекциите и проиграйте сливането на масива на „изплуване“:
- От долу на горе върви нареждането , след като вече масива е разделен на малки сортирани с по един елемент.



7. Проиграйте работата на алгоритъма за Сливане на два наредени масива, като използвате за вход стойностите от лявата и дясната половини на вашия седемместен масив (наредени, очевидно). Долу е дадена примерна схема.





Отговорете, в свободен текст, на следните два въпроси:

- Колко най-малко сравнения на стойности биха се направили **и при какви особености** на входните масиви.
 -Най-малко сравнения ще се правят когато се подават такива наредени масиви ,така че всеки път да нарежда лявата страна първа и после директно да транслира десния нареден масив , тогава ще прави n-сравнения
- Колко най-много сравнения на стойности биха се направили **и при какви особености** на входните масиви.
 Най-много сравнения ще се правят когато трябва всеки път да проверява и нарежда двата масива (ляв и десен) в третия, като стойностите са такива че трябва да се итерира до края и през двата масива.

ДОМАШНОТО СЕ ПРЕДАВА С ТОЧКИТЕ НА ЗАДАНИЕТО!

Георги Филев F104081

Домашна работа към лабораторно и практически реализации – тема 4.

1. Съставете обща графика на времето за изпълнение на два алгоритъма: 1. сортиране по дялове (Quick Sort) и 2. сортиране с пряк избор (Selection sort), базирана на мерене и записване в електронна таблица на времето за изпълнение по двата алгоритъма за входни масиви с 200, 300, 400, ... 1000 елемента със случайни стойности (един и същи случаен набор се подава на двата алгоритъма за всяко генериране на случаен масив).

```
#include <iostream>
```

```
#include <stdlib.h>
```

```
#include <time.h>
```

```
#include <sys/types.h>
```



```
using namespace std;
```

```
void insertionSort(int arr[], int n){
```

```
    int i,key,j;
```

```
    for(int 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;
```

```
    }
```

```
}
```

```
void QSort(int arr[], int li ,int di){
```

```
    int j=li , k = di;
```

```
    if(di > li){
```

```
        int etalon = arr[(di+li)/2];
```

```
        do{
```

```
            while(arr[j]<etalon){
```

```
                j++;
```

```
            }
```

```
            while(arr[k] > etalon){
```

```
                k--;
```

```
            }
```

```
            if(j<=k){
```

```
                int temp = arr[k];
```

```

        arr[k]= arr[j];

        arr[j]= temp;

        k--;

        j++;

    }

    }while(j<=k);

    QSort(arr,li,k);

    QSort(arr,j,di);

}

else{

    cout<<" napuskame cikula"<<endl;

}

}

```

```

int main()

{

    time_t t0, t1;

    clock_t c0, c1;

    srand(time(NULL));


    int arrSize = 2000;

    int arr[arrSize] ;

    for(int i =0; i<2000; i++){

        arr[i] = rand() %100 + 1 ;

        cout<< arr[i];

    }

    t0 = time(NULL);

    c0 =clock();

    for(int i =0 ;i <=2000; i++){

```

```

        insertionSort(arr,i);

        i+=500;
    }

c1 = clock();
t1 = time(NULL);
cout << endl;
cout << "elapsed wall clock time: Insertion " << ((long)(t1 - t0)) << endl;
    cout << "elapsed wall clock time:" << (float)(c1 - c0) / CLOCKS_PER_SEC << endl;


    for(int i =0; i<2000; i++){
        arr[i] = rand() %100 + 1 ;
        cout<< arr[i];
    }

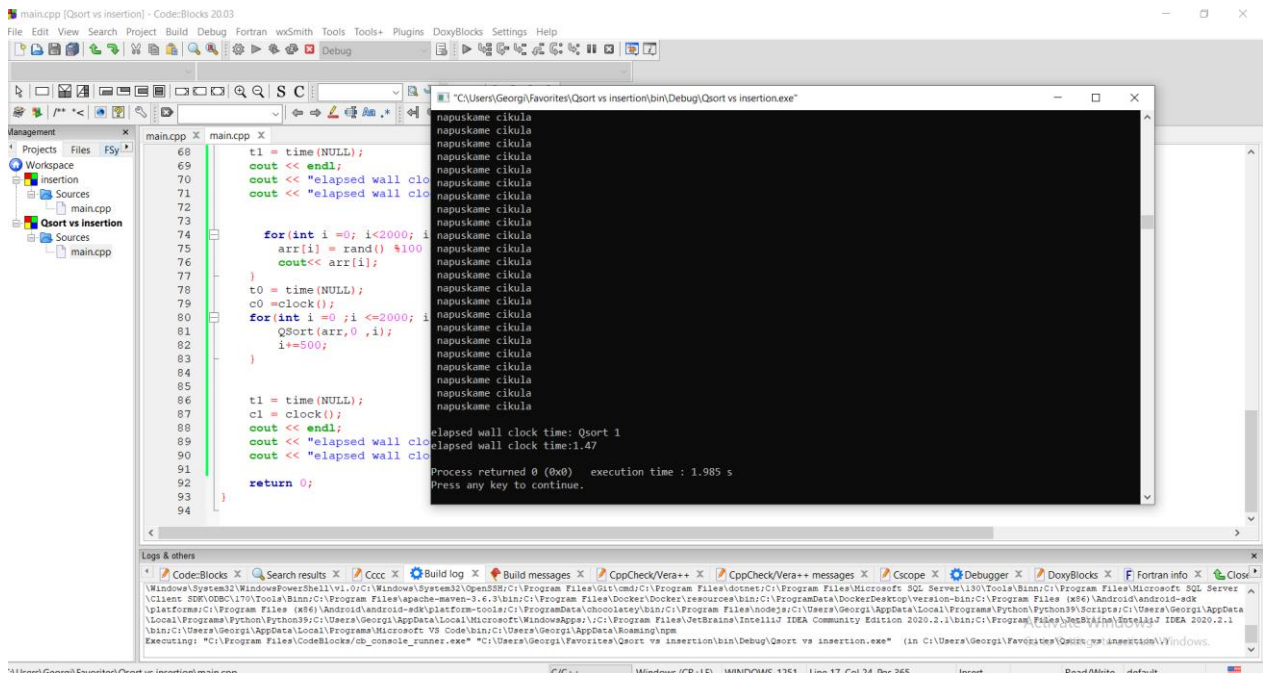
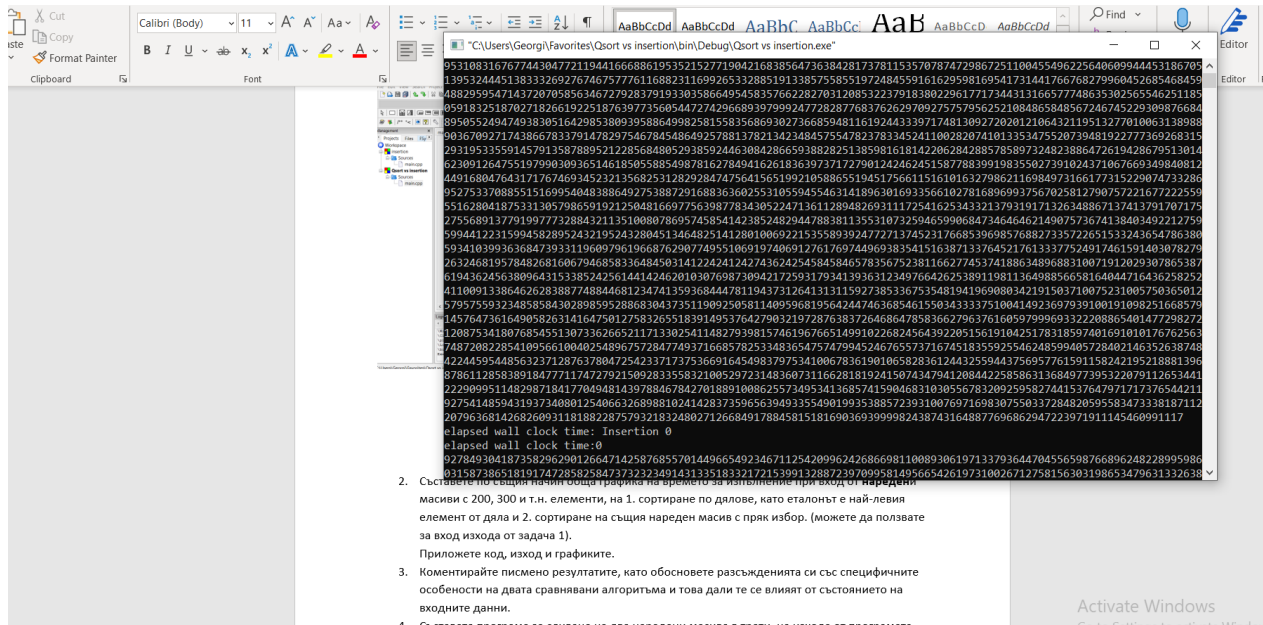
t0 = time(NULL);
c0 =clock();
for(int i =0 ;i <=2000; i++){
    QSort(arr,0 ,i);
    i+=500;
}

t1 = time(NULL);
c1 = clock();
cout << endl;
cout << "elapsed wall clock time: Qsort " << ((long)(t1 - t0)) << endl;
    cout << "elapsed wall clock time:" << (float)(c1 - c0) / CLOCKS_PER_SEC << endl;

```

return 0;

}



- [illegible]

Коментирайте писмено резултатите, като обосновайте разсъжденията си със специфичните особености на двата сравнявани алгоритъма и това дали те се влияят от състоянието на входните данни.

Съставете програма за сливане на два наредени масива в трети, на изхода от програмата да се разпечатават сливаните масиви, как последователно се вземат стойностите от единия и от другия масив , както и слетият и нареден масив.

```
void Merge(int li , int di){
    int j = li, k = (li+di)/2 + 1;
    int r = li;

    while(j<(li + di)/2 + 1 && k<di + 1){
        if(arr[j] < arr[k]){
            Carr[r] = arr[j];
```

```

        j++;
    }
    else{
        Carr[r] = arr[k];
        k++;
    }
    r++;
}
for(int i = j; i < (li+di)/2 + 1; i++){
    Carr[r] = arr[i];
    r++;
}
for(int i = k; i < di+1; i++){
    Carr[r] = arr[i];
}
for(int i = li; di+1; i++){
    cout << Carr[i] << " " << endl;
    arr[i] = Carr[i];
}
}

int main()
{

    Merge(0,9);
    return 0;
}
(не успявам да засека времето за изпълнение)посоаянно е 0 дори с по големи
стойности

```

Съставете слеването по два варианта – 1. с междинна проверка за това кой от двата масива се е изчерпал и 2. Направао два последователни цикъла за пренасяна на стойностите от неизчелпания масив в слетия, като измерите време за изпълнение на двата варианта.

2021 Структури данни

Домашно 5. Анализ и трасиране на двуклонова рекурсия

8. Съставете следния седем-местен масив:

	FN				ЕГН първи 4			
Масив								
	1	2	3	4	5	6	7	

Съставете схеми подобни на тези от лекциите, за да проигравате с този входен масив развитието на рекурсивните процеси на 1. сортиране по дялове с еталон в средата на масива и 2. сортиране чрез сливане (от домашно 4).

9. Отговорете, в свободен текст, на следните два въпроса:
- Защо дълбочината на рекурсията (нива на разгъване на копията) в двата случая (QSort и MergeSort) обичайно не е еднаква при еднакви входни данни.
 - При какво начално нареждане на числата от ВАШИЯ масив по позициите на масива тя би била еднаква. (Това няма да стане лесно без да направите и схема)

ДОМАШНОТО СЕ ПРЕДАВА С ТОЧКИТЕ НА ЗАДАНИЕТО!

Георги Филев F104081

Домашно към тема 5. Сортиране чрез Сливане.

5. Съставете рекурсивната програма за Сортиране чрез сливане.

На изхода от програмата да се разпечатават - на потъване коментарите “потъвам наляво – граници ляв ХХ и десен УУ” и “потъвам надясно - граници ляв ХХ и десен УУ”. Сливането да разпечата “Сливам в С”и “Прехварлям в А”. На изплуване да се разпечата „слях ги и са наредени от индекс // до индекс VV“.

```
#include <iostream>
using namespace std;
int mergeSort(int, int);
int merge(int*, int, int, int*);
int a[20] = { 99,5,6,1,2,3,1,23,34,56,78,45,63,42,78,45,95,864,652,777};
int c[20];
int main() {
    cout << "Purvonachalno sustoqnie na A: " << endl;
    for (size_t i = 0; i < 20; i++)
    {
        cout << a[i] << " ";
    }
    cout << endl;
    mergeSort(0, 19);
}
```

```

for (int i = 0; i < 20; i++)
{
    cout << a[i] << " ";
}
cout << endl;
return 0;
}

int mergeSort(int leftIndex, int rightIndex) {
    if (leftIndex < rightIndex)
    {
        cout << "---Potuvame nalqvo---" << endl;
        cout << "leftIndex = " << leftIndex << " rightIndex = " << rightIndex << endl;
        mergeSort(leftIndex, (leftIndex + rightIndex) / 2);

        cout << "---Potuvame nadqsno---" << endl;
        cout << "leftIndex = " << leftIndex << " rightIndex = " << rightIndex << endl;
        mergeSort(((leftIndex + rightIndex) / 2)+1, rightIndex);

        cout << "Dostignah funkcij merge!" << endl;
        merge(a, leftIndex, rightIndex, c);
    }
    else {
        cout << "--- DUNO ---" << endl;
        cout << "tekush element - a = " << a[leftIndex] << endl;
    }
    return 0;
}

int merge(int* a, int leftIndex, int rightIndex, int* c) {

    int aBeg = leftIndex, bBeg = ((leftIndex + rightIndex) / 2) + 1, cPlace = leftIndex;
    while (aBeg < ((leftIndex+rightIndex)/2)+1 && bBeg < rightIndex+1) {
        if (a[aBeg] < a[bBeg]) {
            c[cPlace] = a[aBeg];
            aBeg++;
        }
        else {
            c[cPlace] = a[bBeg];
            bBeg++;
        }
        cPlace++;
    }
    for (aBeg; aBeg < ((leftIndex + rightIndex) / 2) + 1; aBeg++, cPlace++)
    {
        c[cPlace] = a[aBeg];
    }
    for (bBeg; bBeg < rightIndex+1; bBeg++, cPlace++)

```



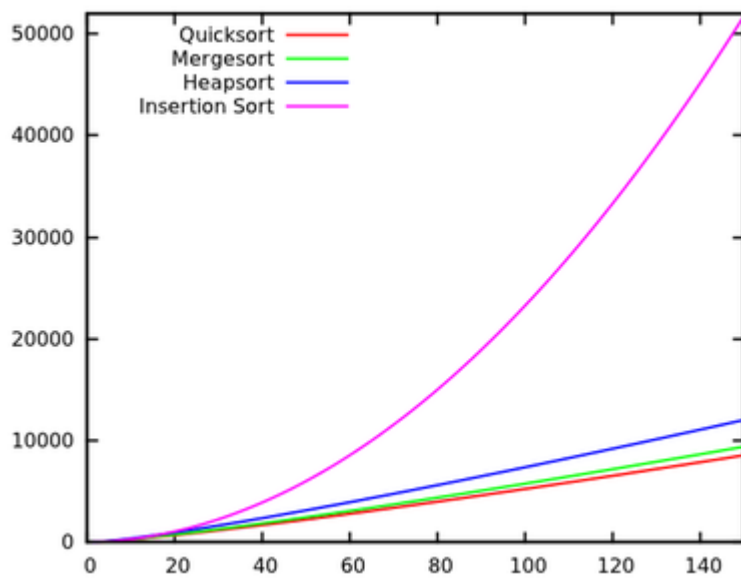
```

{
c[cPlace] = a[bBeg];
}
cout << "Naredih elementite v C!" << endl;
cout << "Premestvam gi v A" << endl;
for (int i = leftIndex; i < rightIndex + 1 ; i++)
{
a[i] = c[i];
}
return 0;
}

```

6. Така съставената програма, изчистена от коментарите, пуснете за сравнение със Сортиране по дялове, при еднакви за двата алгоритъма поредици от входни масиви от случайни стойности (нарастващи постъпково по размер) и, по известната методика, съставете графиката и ги съпоставете.

Basis for comparison	Quick Sort	Merge Sort
Efficiency	Inefficient for larger arrays	More efficient
Sorting method	Internal	External
Stability	Not Stable	Stable
Preferred for	for Arrays	for Linked Lists



Заданието е неразделна част от домашното.

2020 Структури данни

Домашно 6Б 2020. Ханойски кули

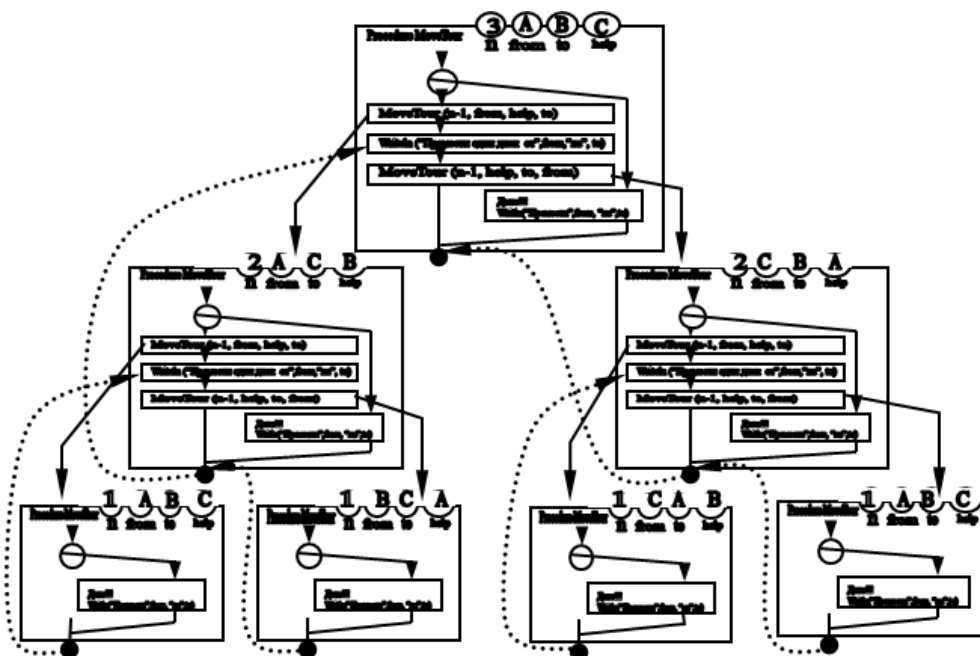
1. Реализирайте Ханойски кули рекурсивно, с разпечатка на последователните инструкции за местене на единични дискове – и я предайте. Проверете на ръка как работи, като изпълните инструкциите, разпечатани на изхода с "истински дискове" (max 5) от <https://www.mathsisfun.com/games/towerofhanoi.html>, предайте снимка на един от екраните.
2. Пуснете работещата програма с 10, 15, 20, 25, 30, 35 и 40 диска. Съставете графика на времето за изпълнение по познатия ви начин – и я предайте заедно с кода.

ДОМАШНОТО СЕ ПРЕДАВА С ТОЧКИТЕ НА ЗАДАНИЕТО!

2020 Структури данни

Домашно 6А. Разделяй и Владей

10. Напишете рекурсивната дефиниция на Ханойска кула (дадена е в лекциите)
11. Разпишете стъпките на рекурсивна процедура за местене на кула, следващи от рекурсивната дефиниция, в псевдокод, като означите параметрите на обмен (в скобите).
12. Съставете пълна схема на развитието на рекурсивния процес, подобна на тази от лекционния материал, за
 - Брой дискове – 4
 - Шишове: Първа Втора и Трета буква от вашето фамилно име.



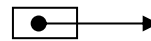
- Разпишете каква последователност от единични преместваня на диск генерира този рекурсивен процес, подобно на разгледаното в час и налично в лекционния материал.

ДОМАШНОТО СЕ ПРЕДАВА С ТОЧКИТЕ НА ЗАДАНИЕТО!

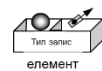
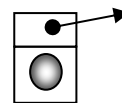
Георги Филев F104081

Домашно 7А. Динамично отреждане на памет, указатели. Образуване на Стек.

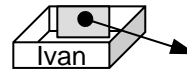
Означение 1. Указател.



Означение 2. Елемент на структура



Означение 3. Променлива – указател.

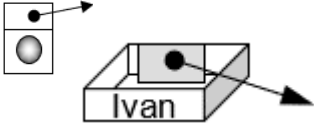
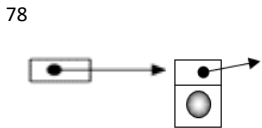
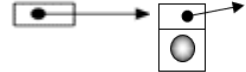
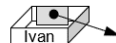



Означение 4. Нулев адрес.



13. Като използвате графични означения подобни на примерните, дадени горе, посочете графично и с текст (на български език) какво прави всеки от следните оператори:

оператори	графична изобразение	текст
<pre>typedef struct Element * po; struct Element { int Data; po Next; };</pre>		
po q;		Инициализираме указател от тип po с име q
po p;		Инициализираме указател от тип po с име p
q=NULL; и		Указателят q няма стойност и не сочи към нищо
p= New Element;		Правим P от тип нов елемент
p-->data = 78;		Стойността на мястото към което сочи указателя е 78

		
<code>p-->Next = q;</code>		Указателя на елемента p сочи към q
<code>q = p;</code>		Елементът q става равен на p ,тоест сочат към едно и също място
<code>p= New Element;</code>		Създаваме нов елемент с име p
<code>p-->data = 38;</code>		Датата на p е равна на 38
<code>p-->Next = q;</code>		Указателят на елемента p сочи към q
<code>q = p;</code>		Q е равно на p
<code>p = p-->Next ;</code>		P е равно на указателя на p

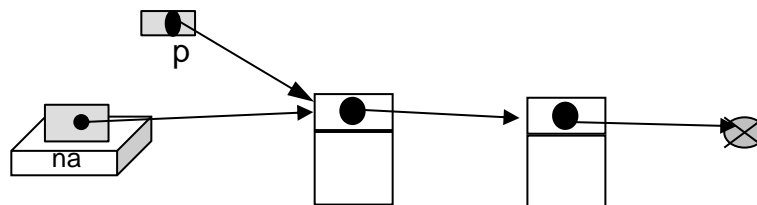
14. Напишете в кода долу смислени коментари на екран, на български, на мястото на дадените в червено (очевидно, трябва първо да разберете какво прави кодът):

```
#include<iostream>
#include<locale>
using namespace std;
typedef struct Element * po;
struct Element{
    int Pole1;
    float Pole2;
    char Pole3;
};
void main() {
    setlocale(LC_ALL, "Bulgarian");
    po p;
    p = new Element;
    cout << "създаваме нов елемент от тип Елемент" << endl;
    cin >> p->Pole1;
    cout << "въвеждаме стойност за Поле1 което е int" << endl;
    cin >> p->Pole2;
    cout << " въвеждаме стойност за Поле2 което е float" << endl;
    cin >> p->Pole3;
    cout << " въвеждаме стойност за Поле3 което е char" << endl;
    cout << p->Pole1 << " " << p->Pole2 << " " << p->Pole3 << endl;
    cout << endl;
    po q;
    q = new Element;
    cout << " създаваме нов елемент от тип Елемент с име q" << endl;
    cin >> p->Pole1;
```

```

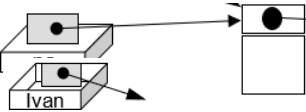

cout << " въвеждаме стойност за Поле1 което е int" " << endl;
cin >> p->Pole2;
cout << " въвеждаме стойност за Поле2 което е float" " << endl;
cin >> p->Pole3;
cout << " въвеждаме стойност за Поле3 което е char" " << endl;
cout << p->Pole1 << " " << p->Pole2 << " " << p->Pole3 << endl;
cout << "изписваме всички стойности на елемента p" << endl;
q = p;
cout << endl;
cout << "стойността на Element q е равна на p" << endl;
cout << p->Pole1 << " " << p->Pole2 << " " << p->Pole3 << endl;
cout << " изписваме всички стойности на елемента p " << endl;
cout << p->Pole1 << " " << p->Pole2 << " " << p->Pole3 << endl;
p->Pole1 = int(p->Pole2);
cout << "Поле1 от елемента p е равно на Поле2 от елемента p като е typecast-нато като int" <<
endl;
cout << p->Pole1 << " " << p->Pole2 << " " << p->Pole3 << endl;
}

```



15. Като използвате графични означения подобни на примерните горе, нарисуйте какво прави **ВСЕКИ** от операторите при две преминавания през първия и после – през втория цикъл:

оператори	графична изображение	
<pre> struct Element { int Data; po Next; }; void main() {po na; po p; na = NULL; </pre>	<pre> graph LR Node[] --> Null((X)) </pre>	
while (потребителят иска да въведе още данни)	графична изображение първо преминаване	графична изображение второ преминаване

<p><code>p = new Element;</code></p> <p><code>p->data = вход от клавиатура</code></p> <p><code>p->next = na;</code></p> <p><code>na = p;</code></p> <p>край на while</p>	 <p>Указателят Р сочи към na</p> <p>Na= p</p>	<p>Вторият път това което се случва е че сегашният елемент сочи към предишния</p>
<p><code>p = na;</code></p>	<p>графична изображение</p>	
<p>while (p не сочи нулев адрес)</p> <p>печат на <code>p->data</code></p> <p><code>p = p->next;</code></p> <p>край на while</p> <p>}</p>	<p>графична изображение първо преминаване</p>  <p>Указателят на p сочи към предишния</p>	<p>графична изображение второ преминаване</p>

16. Въпрос, зададен на тест по алгоритми и структури, е:

Обяснете в свободен текст кога рекурсивният алгоритъм quick sort работи със сложност точно $n \cdot \log(n)$?				
ПОСОЧЕТЕ ЗА ВСЕКИ ОТГОВОР ИМА ЛИ, и ако има - КОЕ Е:				
Представен отговор	Неточен или неясен термин(и)	Невярно или неясно твърдение	Важно допълнение към текста, за да стане верен	Напишете тук ваш коментар
QuickSort работи със сложност $n \cdot \log(n)$, когато избраният еталон е точно средната по големина стойност в масива.			ДА	
В най-добрия случай, когато еталонът е средният по големина елемент.		ДА		

когато се изпълнява рекурсия и почва да се дели масивът всеки път на 2 и за двете части (при търсенето на нов еталон)	да			
Когато масивът е удобен да си го разделяме на по 2 равни части и с правилен еталон.		ДА		
Най-лошият случай е, когато една от двете половини върнати се пада с размер $n-1$. $n \cdot \log(n)$ имаме, когато и двете половинки са равни части или поне близо до това да са равни на брой.	ДА			
Когато елементът е избран за 'еталон', се пада и среден по големина. Тогава се получават две половини с равна дължина.			ДА	
Quick sort ще работи със сложност от $n \cdot \log(n)$ когато всеки път при разделяне на дялове бъде избран за еталон точно средният елемент, т.е. когато разделяме масива на две половини. Това ще бъде и най-добрият случай за quick sort.			ДА	
Всяка операция за разделяне отнема $O(n)$, но всяко делене прави масива два пъти по-малък, следователно имаме $\log(n)$ операции. Обобщено това води до: $O(n \cdot \log(n))$		неясно	Нужно да се спомене	
Сложността е $n \cdot \log(n)$, когато разглеждаме възможно най-добрия случай и се случва, когато разделянето води до равни подмасиви.			ДА	
Когато за еталон е взето средното число на масива.	ДА			
Алгоритъмът работи със сложност $n \cdot \log(n)$ в най-добрият си случай, когато за еталон винаги взимаме средната стойност за разглеждания дял.		ДА		
$n \cdot \log(n)$ е най - благоприятният случай на рекурсивният алгоритъм quick sort. Това се получава когато за всеки елемент позицията на дяла се намира в средата, което води до балансирано дърво с височина $\log(n)$, следователно общата стойност става. $O(n \cdot \log n)$. Казано с други думи еталонът винаги трябва да е средният елемент.			ДА	
Когато еталонът е винаги така избран и данните са така наредени, че разделянето на масивът при потъване е винаги на две равни части. Пример: при наредени данни.			ДА	

В таблицата има и отговори, които са коректни и пълни.

ДОМАШНОТО СЕ ПРЕДАВА С ТОЧКИТЕ НА ЗАДАНИЕТО!

Домашно 7Б 2021

Задача. Съставете програма за образуване на стек посредством указатели, който да поема произволен брой букви и да ги разпечатва ноабратно. Например, като въведете трите си имена, те да се изписват в ред, обратен на четенето. Приложете програмния текст, входа и изхода.

```
#include <iostream>
```

```
using namespace std;
```

```
typedef struct Element * strelka;
```

```
struct Element
```

```
{
```

```
    char Data;
```

```
    strelka Next;
```

```
};
```

```
int main()
```

```
{
```

```
    strelka start = NULL, p = NULL, obj = NULL;
```

```
    char x = 0;
```

```
    cin>>x;
```

```
    while(x != '1')
```

```
    {
```

```
        p = new Element;
```

```
        p->Data = x;
```

```
        p->Next = start;
```

```
        start = p;
```

```
        cin>>x;
```

```
    }
```

```
    obj = start;
```

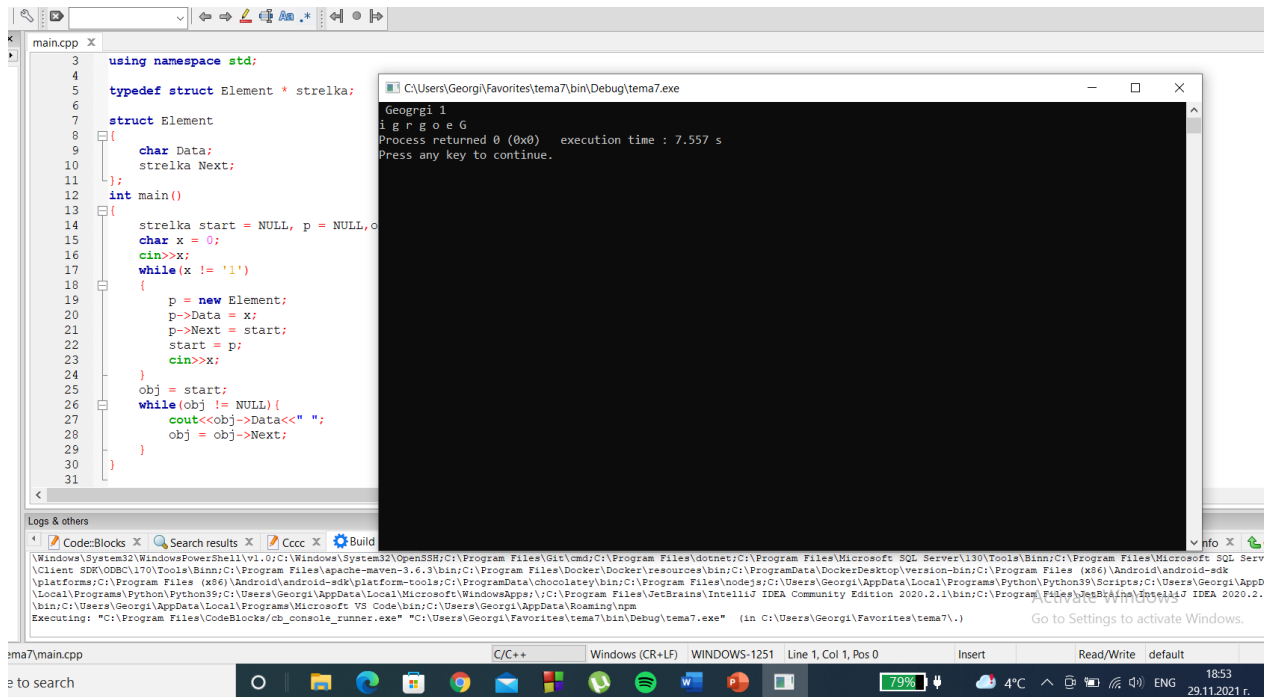
```
    while(obj != NULL){
```

```
        cout<<obj->Data<<" ";
```

```
        obj = obj->Next;
```

```
    }
```

```
}
```

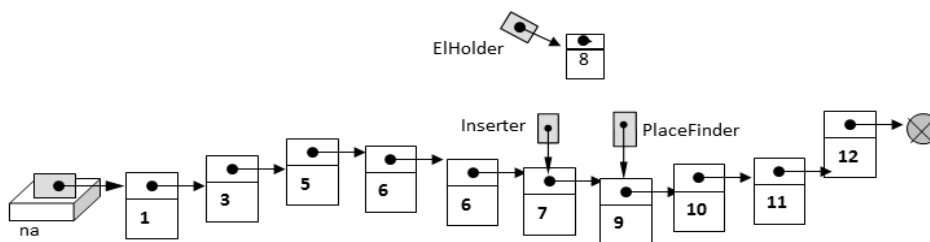



Георги Филев F104081

2020 Структури данни

Домашно 8A.

1. Като използвате означенията от графичното изображение долу, напишете примерна последователност от оператори и означете графично какво прави всеки от операторите при нарастване на нареден линеен списък.



последователност на действията	Примерен програмен текст	графична изображение
<pre> typedef struct Element *po; struct Element { int Data; po Next; }; void main() { po na; po p; ??;? na = NULL; </pre>	<pre> typedef struct Element *po; struct Element{ po Next; int data; }; void main(){ po na = NULL, p, contr; int x; </pre>	

while (потребителят иска да въведе
още данни)

? = new Element;

вход от клавиатура

?->data = входа от клавиатура

If (това и/или онова е вярно)

?

?->next = na;

na = ?;

иначе

? = na; ?=na:

Обхождане за намиране на място
за вмъкване

```
cin>>x;
```

```
p = New Element;
```

```
p->data =x;
```

```
p->next = na;
```

```
if(na ==NULL || na->data >=x){
```

```
na = p;
```

```
}
```

```
else{
```

```
do{contr = p->next
```

```
p->next = p->next->next;
```

```
}
```

```
while(p->next !=NULL &&  
p->next->data<x);
```

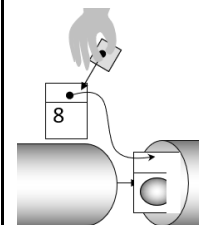
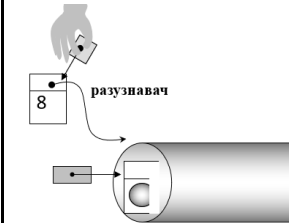
```
contr->next = p;
```

```
}
```

създаваме нов елемент

и въвеждаме стойност на елемент с указател
p ,който ще търсим(ще търсим първия
по-голям елемент от него)

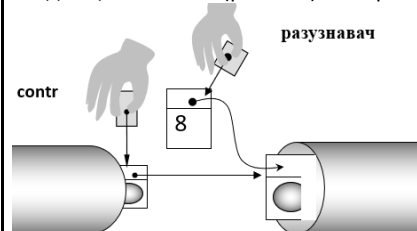
na е елемента към който сочи p (следващия)



ако na е нищо или
елемента е по голям или равен на търсеното
число то този елемент става na

Прави p да сочи към контролора

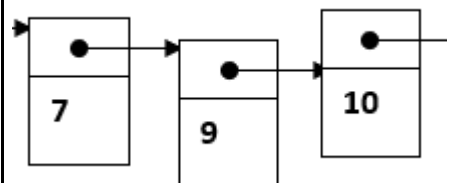
и указателя на елемента който сочи
следващия елемент (p ->next) сочи p



contr->next

p->next->next

p->next

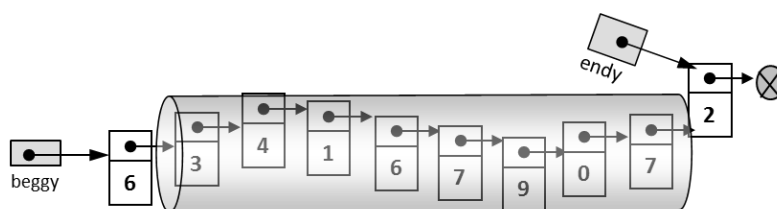


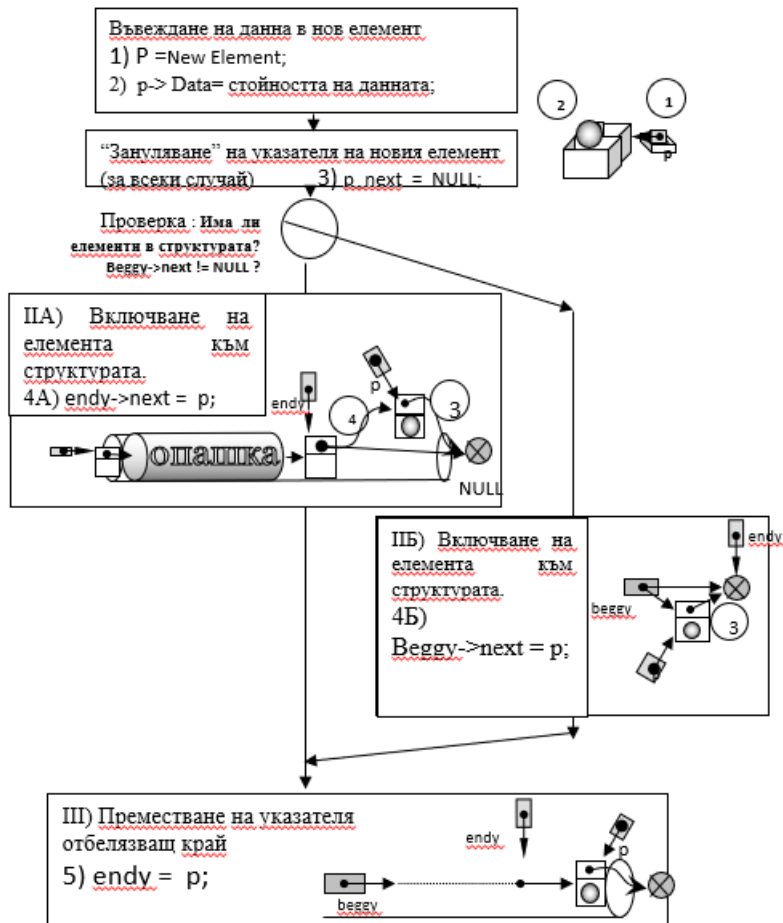
спира докато следващия елемент не е нищо
или на търсената позиция контролора става
следващия елемент

край на обхождането Вмъкване: ? = ? ? = ? Край на while (потребителят иска да въведе още данни) Край	}	
---	---	--

2. Като проследите схемата и означенията от графичните изображения долу, попълнете операторите, отбелязани с „??“ в схемата за нарастване на линеен динамичен списък (реализиран с указатели) като опашка.

Схема, удобна за директно попълване е дадена на последния слайд от анимацията за опашка. Можете да попълните там и да копирате отговора си в домашното.

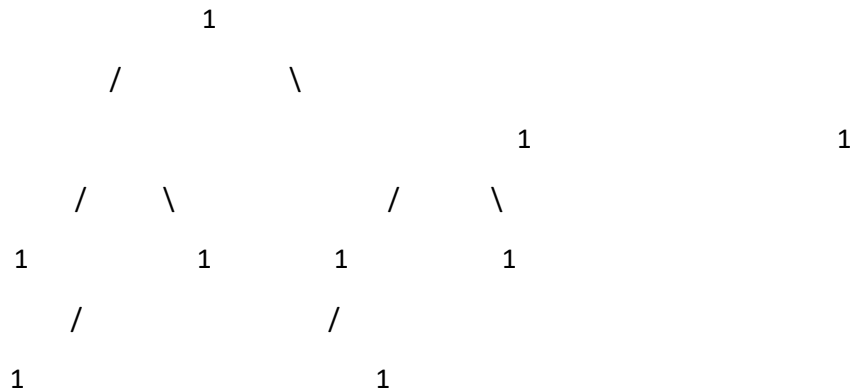




3. А. Запишете дефиницията за ИБД с n възела и Б. нарисуйте ИБД с толкова възли, колкото е сумата от последните две цифри на факултетния ви номер (взети като числа). Ака тя се получи по-малка от 6, работете с 6.

Идеално Балансирано Дърво е когато или няма никого или има един възъл на когото има две идеално балансирани дървета. Лява част $n/2$, дясна част $n - n/2 - 1$

F104081



ДОМАШНОТО СЕ ПРЕДАВА С ТОЧКИТЕ НА ЗАДАНИЕТО!

Георги Филев F104081

2020 Структури данни

Домашно 8Б.

1. Съставете програма за поддържане на *нареден линеен списък* посредством указатели. Подайте на входа трите си имена, в реда на изписване, и изведете на екран съдържанието на списъка.

```
#include <iostream>
```

```
#include <string>
```

```
#include <sstream>
```

```
using namespace std;
```

```
typedef struct Element* pointer;
```

```
struct Element {
```

```
    char data;
```

```
    pointer next;
```

```
};
```

```
void insert(pointer elem, char bukva){
```

```
    pointer p;
```

```
    p = new Element;
```

```
    p->data = bukva;
```

```
    p->next = NULL;
```

```

if( elem == NULL || elem->data >= буква){
    p->next = elem;
    elem = p;
}
else{
    pointer endy = elem;

    while(endy->next != NULL && endy->next->data < буква){
        endy = endy->next;
    }
    p->next = endy->next;
    endy->next= p;
}
}

int main()
{
    string input;
    while (getline(cin, input)){
        pointer head ;
        head = NULL;
        char letter;
        istringstream s(input);
        while(s >> letter){
            insert(head, tolower(letter));
        }

        cout<< "Output :";
        while(head != NULL){
            cout<< head->data;
            head = head->next;
        }
        cout<<endl;
    }
    return 0;
}

```

2. Съставете програма за поддържане на *линеен динамичен списък като опашка* посредством указатели. Подайте на входа трите си имена, в реда на изписване, и разпечатайте съдържанието на линейния списък. Включете в програмния текст част за намаляване на опашката. „Извадете “ от линейната структура „първия влязъл“, т.е. името ви, и отново разпечатайте съдържанието на линейния списък.
(Предаватے програмен текст и снимки на екраните с изход.)

```

#include <iostream>
#include <string>
#include <sstream>

using namespace std;

typedef struct Element* po;

struct Element{
    string data;
    po next;
};

bool isEmpty(po beg){
    if(beg == NULL ) return true;

    return false;
}

void push(po beg, po endy ,string data){
    po p = new Element;
    p->data = data;
    p->next = nullptr;
    if(endy == nullptr){
        endy = p;
        beg = endy;
    }
    else{
        endy->next = p;
        endy = p;
    }
}

string pop(po beg, po endy){
    if(beg == nullptr){
        return "";
    }
    string out = beg->data;
    po tmp = beg;
    beg = beg->next;
    delete tmp;
}

```

```

        if(beg == nullptr){
            endy = beg;
        }

        return out;
    }

int main (){
    po beg , endy;
    beg = endy = nullptr;

    string input;
    while(getline(cin, input)){
        istringstream s(input);
        string word;
        while (s >> word){
            push(beg, endy , word);
        }
        pop(beg , endy);

        cout<< "Output: ";
        while(isEmpty(beg) == false){
            cout << pop(beg, endy) << " ";
        }
        cout<<endl;
    }
    return 0;
}

```

ДОМАШНОТО СЕ ПРЕДАВА С ТОЧКИТЕ НА ЗАДАНИЕТО!

Georgi Filev F104081

Домашно 9.а. Построяване на ИБД по брой възли. Обхождане на дърво.

1. Долу е дадена схемата на управление на алгоритъма за построявена на ИБД с ен възела по рекурсивната дефиниция, както е обяснена на лекции.
 - А. Запишете дефиницията, като отразите факта, че връзките в ИБД в този случай се реализират от указатели.

ИБД с n възела е или празна структура, тоест $n=0$, или има един възел-корен за който има закачени две ИБД-та – или ляво ИБД с $n/2$ възела и дясно с $n - n/2 - 1$ възела

- Б. Встрани от схемата, успоредно на изобразените графично оператори, запишете на български език какво се извършва от всеки от тях.



2. Преработете кода за построяване на ИБД, даден в лекциите, така, че при подаване на числа от клавиатура в **нарастващ** ред, при инфиксното обхождане на дървото, се получава нареден в **нарастващ** ред изход на екран. Предайте снимки на екрани на кода, входа и изхода. Коментирайте с текст в какво, от гледна точка на алгоритмите за построяване и за обхождане, се състои преработката, която сте направили.

```
typedef int DataT;
```

```
typedef struct node *po;
```

```
struct node { Data data ;
```

```
    Po left ;
```

```
    Po right;
```

```
}
```

```
Po ibd(int n){
```

```

Po darj;
Data x;
}
If (n > 0){
    Int nl = n/2 , nd = n-nl-1;
darj = new node;
cin>> x ; darj -> data = x;
darj -> left = ibd(nl);
darj -> right = ibd(nd);
return darj;
}
Else return NULL;
}

```

ДОМАШНОТО СЕ ПРЕДАВА С ТОЧКИТЕ НА ЗАДАНИЕТО

Georgi Filev F104081

Домашно 96. Построяване на ИБД по брой възли. Обхождане на дърво.

1. Съставете програма за построяване на идеално балансирано дърво от n възли, като включите разпечатка на дървото на екран, като структура. Помощен файл за разпечатване е даден към тази тема. Реализирайте програмата с l и m за броя възли на входа, като l и m са съответно цифрите на факултетния ви номер, взети по двойки отзад напред и интерпетирани като двуцифрени числа. Стойностите на възлите са числата от 1 до l и от 1 до m , подавани във възходящ ред.



2. Разпечайте (като списък) обходените възли:

а. префиксно;

б. инфиксно;

в. постфиксно.

```

#include <iostream>
using namespace std;
int COUNT = 10;
typedef char dataT;
typedef struct node* po;
struct node {
    dataT data;
    po left;
    po right;
};
po ibd(int n) {
    po darj;
    if (n == 0) {
        return NULL;
    }
    else {
        darj = new node;
        cout << "Data?";
    }
}

```

```

cin >> darj->data;
darj->left = ibd(n/2);
darj->right = ibd((n - n / 2) - 1);
return darj;
}
}

```

```

void print(po root, int space)
{

```

```

    if (root == NULL)
        return;

    space += COUNT;
    print(root->right, space);
    cout << endl;
    for (int i = COUNT; i < space; i++)
        cout << " ";
    cout << root->data << "\n";
    print(root->left, space);
}

```

```

void inf(po root) {
    po help = root;
    if (help != NULL) {
        cout << "(";
        inf(help->left);
        cout << help->data;
        inf(help->right);
        cout << ")";
    }
}

```

```

void pre(po root) {
    po help = root;
    if (help != NULL) {
        cout << help->data;
        pre(help->left);
        pre(help->right);
    }
}

```

```

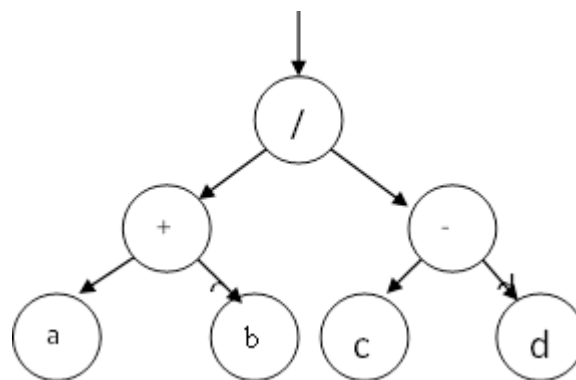
void post(po root) {
    po help = root;
    if (help != NULL) {
        post(help->left);
        post(help->right);
        cout << help->data;
    }
}

```

```
}  
}
```

```
int main()  
{  
    po root;  
    int n;  
    cout << "Number of elements?";  
    cin >> n;  
  
    root = ibd(n);  
    print(root, 10);  
    cout << "\n";  
  
    cout << "Infix: ";  
    inf(root);  
    cout << "\n";  
  
    cout << "Prefix: ";  
    pre(root);  
    cout << "\n";  
  
    cout << "Postfix: ";  
    post(root);  
    cout << "\n";  
  
    return 0;  
}
```

3. Съставете програма за построяване на ИБД с n възела, която след като построи дървото го обхожда с операцията “разпечатване на полето с данни”. Как да бъдат подадени данните (в какъв ред?), така, че да се получи следното дърво:



Трябва да е инфиксно

И което, като обходите инфиксно, да разпечатате в следния вид:

$((a)+(b))/((c)-(d))$

Внимавайте със скобите, коя да се разпечатва на потъване и коя на изплуване. Скобите в горния запис НЕ СА част от входните данни.

Предайте работеща версия на кода със съответно екрани и на изходите.

ДОМАШНОТО СЕ ПРЕДАВА С ТОЧКИТЕ НА ЗАДАНИЕТО

Georgi Filev F104081

Структури данни

Домашно 10А. Дървета – растеж, обхождане, нареденост, баланс.

Рекурсивно:

```
Void Srch(int x, po loc){
```

```
    If(loc== NULL || loc->data ==x){
```

```
        Here = loc;
```

```
        If(here == NULL) cout<<"not in";
```

```
    }
```

```
    Else
```

```
        If(loc->data > x){
```

```
        Srch(x,loc->left);
```

```
        }
```

```
        Else Srch(x , loc-> right);
```

```
}
```

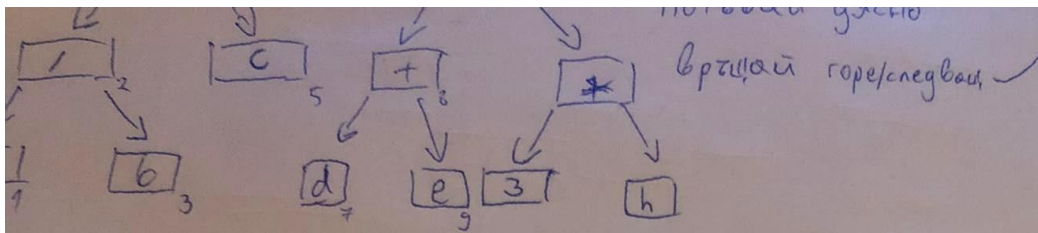
Нерекурсивно с while и после съответно loc = loc ляво или дясно (зависимо дали е > или < от x)

Задача 1. Образувайте списък от числа:

FN1, 2 , FN2, 8, FN3, 12, FN4, 3, FN5, 7, EGN1, 0, EGN2, 9

Премахнете от горния списък повтарящите се.

Те пристигат в този ред на входа на процедура за образуване на ДДП. Изобразете графично как изглежда дървото. Запишете по какъв начин трябва да се обходи дървото с оператор за разпечатване, за да се получат числа, наредени във възходящ ред. Отбележете върху дървото как, в каква последователност се развива рекурсивният процес в този случай, по копия, и разпишете отделно последователността на разпечатаните стойности.



Постфиксно: $ab/c + de + 3h * - *$

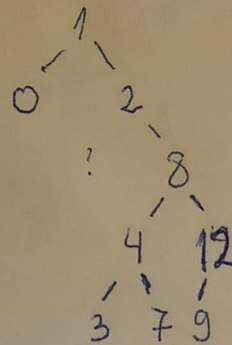
F104081

1, 2, 0, 8, 4, 12, 0, 3, 8, 7, 0, 0, 9

без повтарящи 1, 2, 0, 8, 4, 12, 3, 7, 9



първо горното
→ трябва



за да са
взходящи
префиксно \times
постфикс \times
инфиксно: \checkmark

0 → 1 → (в ляво от 2 няма
ответно връща) 2 →
→ (отива до най ляво) 3 →
→ връща 4 → 7 → 8 ...

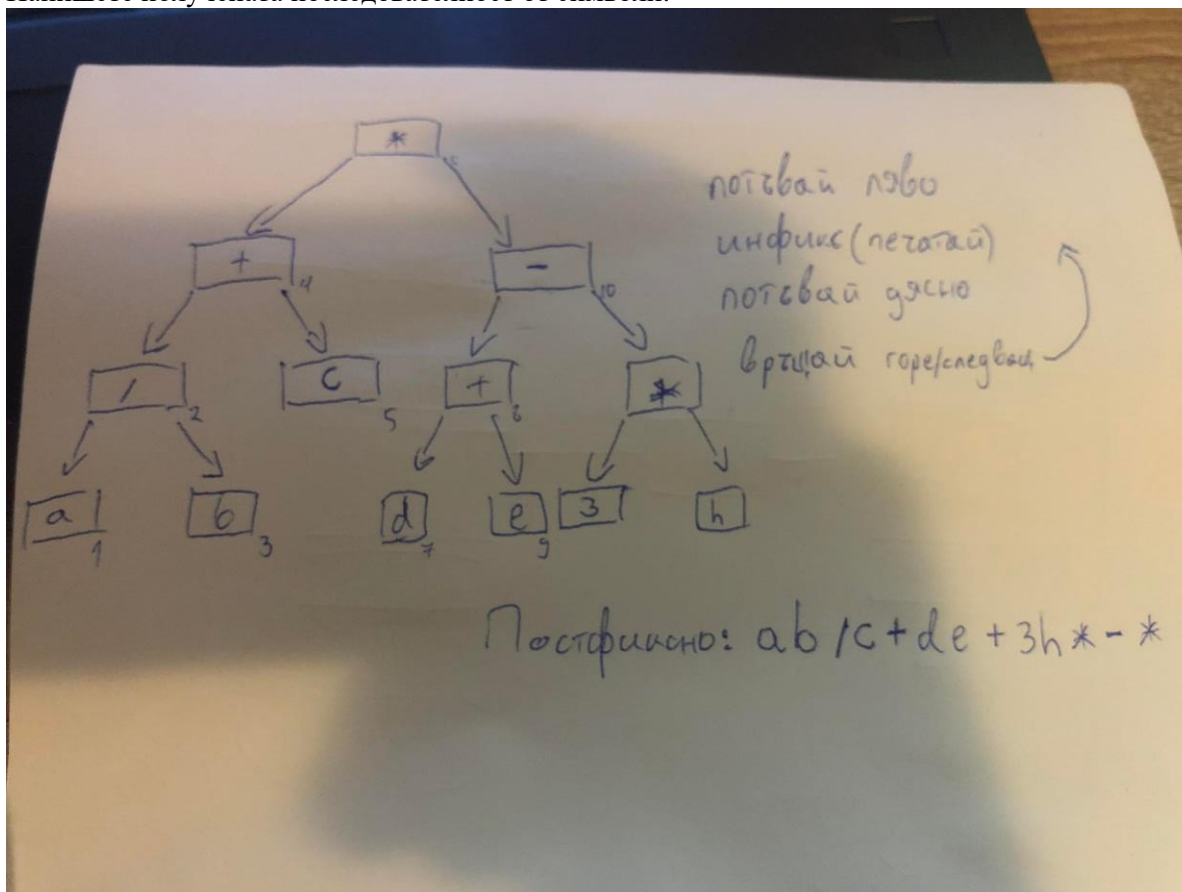
Задача 2. За формулата с инфиксен запис:

$$(a / b + c) * ((d + e) - 3 * h)$$

постройте (нарисувайте) двоично дърво с инфиксна наредба.

Като се базирате на рекурсивния алгоритъм за постфиксно обхождане на дърво, отбележете върху дървото (схемата) с поредни номера последователните рекурсивни копия при обхождането, като посочите пътя на «потъване» и «изплуване» на рекурсивния процес.

Напишете получената последователност от символи.



ДОМАШНОТО СЕ ПРЕДАВА С ТОЧКИТЕ НА ЗАДАНИЕТО!

Георги Филев F104081

Домашно 10Б 2020 ДДП. ИБДП

Задача 1.

Предайте работеща програма (с вход и изход) за образуване на двоично дърво за претърсване. Програмата да докладва след колко проверки установява, че а) търсената стойност е вече записано в дървото или б) стойността я няма, но се включва като лист.

След последователно въвеждане на стойности, които са: вашия факултетен номер, взет като пет едноцифрени числа и после като две двуцифрени числа, със залепени за него ЕГН -

първи четири, взети като едноцифрени и после като двуцифрени (отляво надясно), разпечатайте изход за следното:

- Как изглежда дървото
- Изход от инфиксно обхождане.

```
#include <iostream>
```

```
using namespace std;
```

```
const int COUNT = 10;
```

```
typedef struct Node* po;
```

```
struct Node {  
    int data;  
    po left;  
    po right;  
};
```

```
// двоично дърво за претърсване
```

```
po ddp = nullptr;
```

```
// binary tree build
```

```
void btb(int x) {  
    po insert, loc;  
    insert = ddp;  
    loc = ddp;
```

```
if (ddp == nullptr) {
    ddp = new Node;
    ddp->data = x;
    ddp->left = nullptr;
    ddp->right = nullptr;
}
else {
    while (loc != nullptr && loc->data != x) {
        insert = loc;

        if (x > loc->data) {
            loc = loc->right;
        }
        else {
            loc = loc->left;
        }
    }

    if (loc != nullptr) {
        cout << "found!" << endl;
    }
    else {
        loc = new Node;
        loc->data = x;
        loc->left = nullptr;
        loc->right = nullptr;

        if (x > insert->data) {
```

```
insert->right = loc;
```

```
}
```

```
else {
```

```
insert->left = loc;
```

```
}
```

```
}
```

```
}
```

```
}
```

```
void print(po root, int space)
```

```
{
```

```
if (root == NULL)
```

```
return;
```

```
space += COUNT;
```

```
print(root->right, space);
```

```
cout << endl;
```

```
for (int i = COUNT; i < space; i++)
```

```
cout << " ";
```

```
cout << root->data << "\n";
```

```
print(root->left, space);
```

```
}
```

```
int main() {
```

```
int n = 20;
```

```
cin >> n;
```

```
while (n != -1866) {
```

```
btb(n);
```

```
cin >> n;
```

```
}
```

```
print(ddp, 10);
```

```
return 0;
```

```
}
```

The screenshot shows a C++ IDE with a code editor on the left and a console window on the right. The code in the editor is as follows:

```
61  
62 void print(po root, int space)  
63 {  
64     if (root == NULL)  
65         return;  
66  
67     space += COUNT;  
68     print(root->right, space);  
69     cout << endl;  
70     for (int i = COUNT; i < space; i++)  
71         cout << " ";  
72     cout << root->data << "\n";  
73     print(root->left, space);  
74 }  
75  
76 int main() {  
77     int n = 20;  
78     cin >> n;  
79     while (n != -1866) {  
80         btb(n);  
81         cin >> n;  
82     }  
83  
84     print(ddp, 10);  
85     return 0;  
86 }  
87  
88  
89
```

The console window on the right shows the output of the program, which is a binary tree structure printed in infix notation. The output is as follows:

```
found!  
0  
4  
0  
found!  
8  
1  
found!  
10  
40  
81  
0  
found!  
0  
found!  
4  
found!  
3  
00  
found!  
43  
-1866  
  
81  
43  
40  
10  
8  
4  
3  
1  
0
```

Infix : 0 1 3 4 8 10 40 43 81

Задача 2.

Предайте работеща програма (с вход и изход), която образува Идеално Балансирано Двоично Дълво, но то е Наредено (т.е. то е дърво за претърсване). Програмата получава на входа нареден масив и връща указател към изграденото наредено ИБД.

Въведете на входа **НАРЕДЕНИ** следните стойности, вашия факултетен номер, взет като пет едноцифрени числа и после като две двуцифрени числа, със залепени за него ЕГН -

първи четири, взети като едноцифрени и после като двуцифрени (отляво надясно). (От нас да мине - тази поредица вие получавате от изхода на задача 1.)

Разпечатайте на изход за задача 2 как изглежда ИБД-то.

2020 Структури данни

Домашно 11. Дървета – премахване на възли. Нареденост и баланс.

Задача 1. Като се базирате на материала от лекциите, направете на ръка схема на алгоритъма за премахване на възли от ДДП. Трите случая. Направете опорна схема на обхождането, т.е. самото дърво и означете и кръстете с имена указателите, които ви трябва, за да премахнете възел с ключ X. Встрани от алгоритъма - псевдокод, може и на български.

Задача 2. Изобразете чрез схема ДДП - дърво на Фибоначи, образувано от елементи със следните ключове:

12, FN1, 42, FN2, 55, FN3, 6, FN4, 9, FN5, 19, 21

Забележете, че дървото, като форма, е дърво на Фибоначи, но данните в него са наредени, защото то е ДДП.

Задача 3. Изобразете чрез схема ИБД, което е наредено и съдържа горните ключове.

ДОМАШНОТО СЕ ПРЕДАВА С ТОЧКИТЕ НА ЗАДАНИЕТО!