

Мавзу:Рекурсив алгоритмлар.

•

Режа:

- 1.Рекурсия хақида тушунча. Рекурсив алгоритмлар.
- 2.Рекурсив функция тушунчаси.
- 3.Рекурсив функцияларга оид мисоллар.

“Rekursiya nimaligini tushunish uchun oldin rekursiya nimagligini tushunish kerak” — Stephen Hawking



Rekursiya ta'rifi

- **Ta'rif:** *Funksiya o'ziga o'zi to'g'ridan-to'g'ri yoki qandaydir vosita orqali murojaat qilish jarayoniga **rekursiya** deyiladi va bunday funksiya **rekursiv funksiya** deb ataladi.*

Rekursiyani to'g'ri tashkil qilish shartlari

- Har qanday to'g'ri tuzilgan rekursiya asosini ikkita shart tashkil qiladi.
- Rekursiya **asos sharti**
- Funksiyaning o'ziga **o'zgartirilgan argument** bilan murojaat qilish sharti.

- Rekursiv funksiya qaysidir vaqtga kelib o'ziga murojaat qilishni to'xtatishi kerak bo'ladi. Aynan shu narsani **rekursiya asos sharti** ta'minlab beradi.

Рекурсия ҳақида тушунча

Изоҳ

Рекурсия (умуман)— бу объектни мазкур объектга мурожаат қилиш орқали аниқлашдир.

Изоҳ

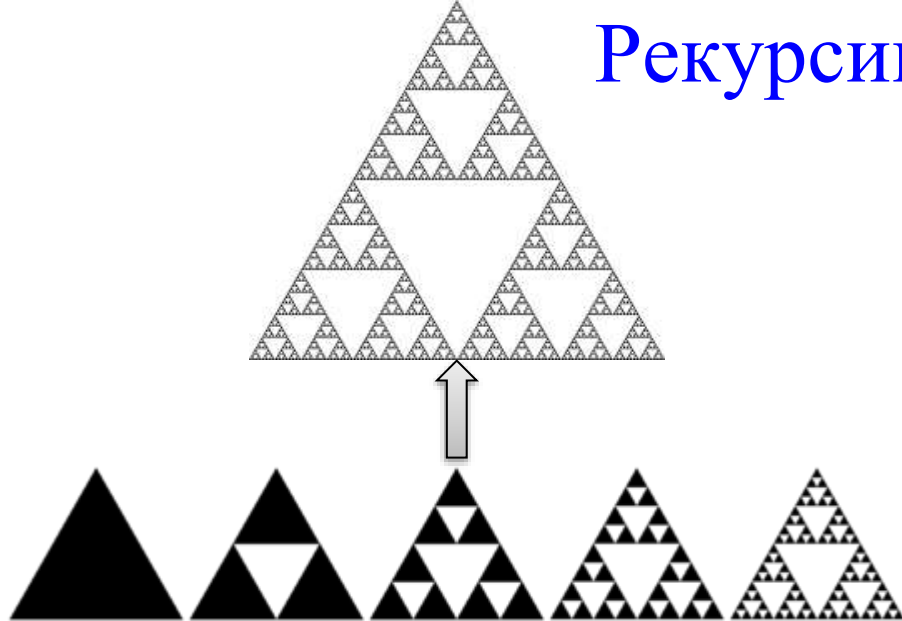
Рекурсив алгоритм — бу алгоритмни аниқлашда ўзига бевосита ёки билвосита мурожаат қилишдир.

Def.1.

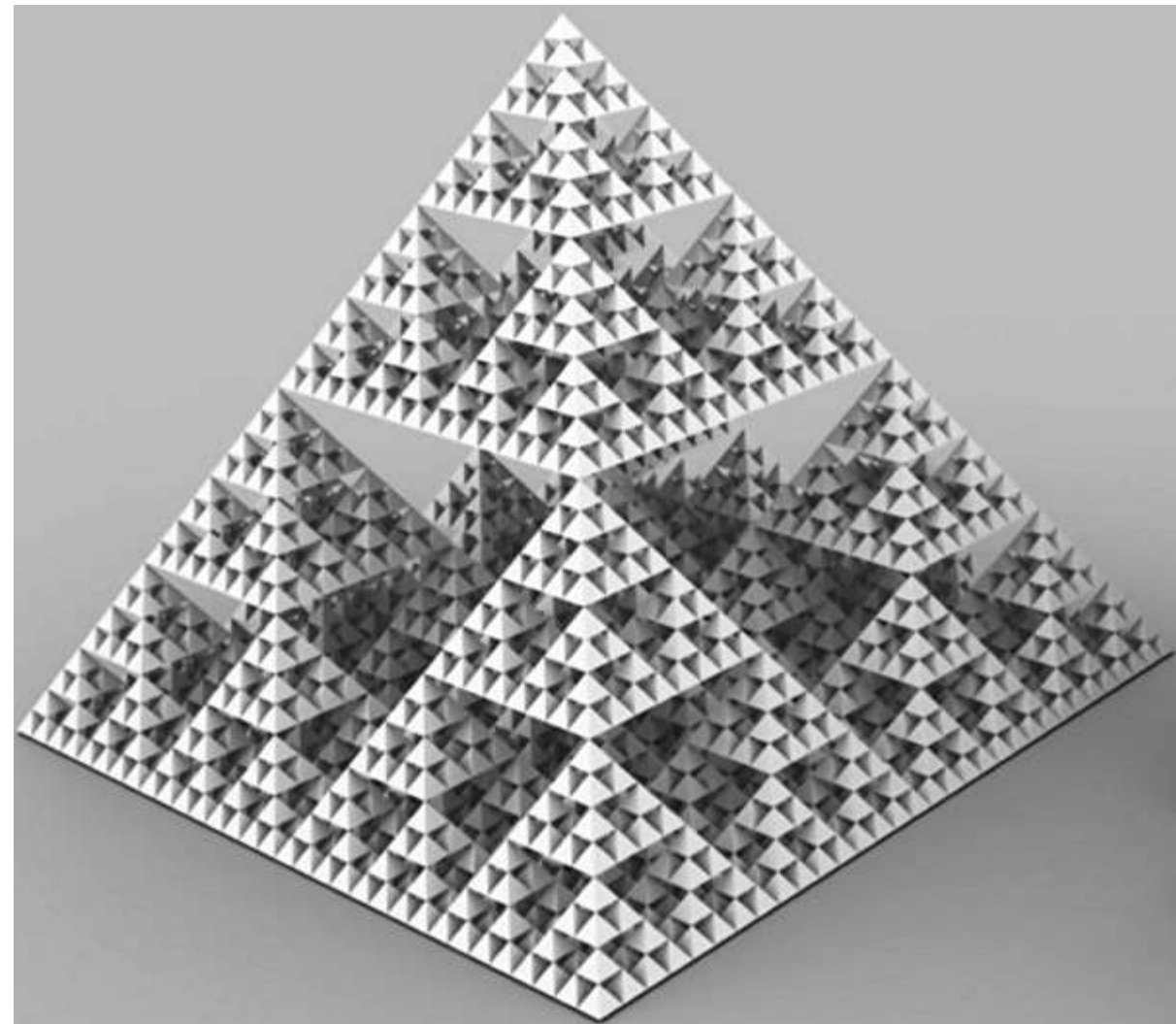
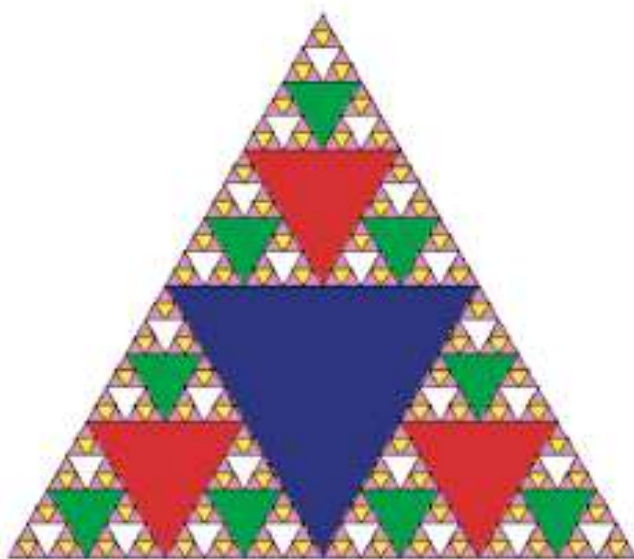
Агар маълумотлар тузилмаси элементлари ҳам мазкур тузилмага ўхшаш тузилма бўлса, у ҳолда бундай тузилмага рекурсив маълумотлар тузилмаси дейилади.

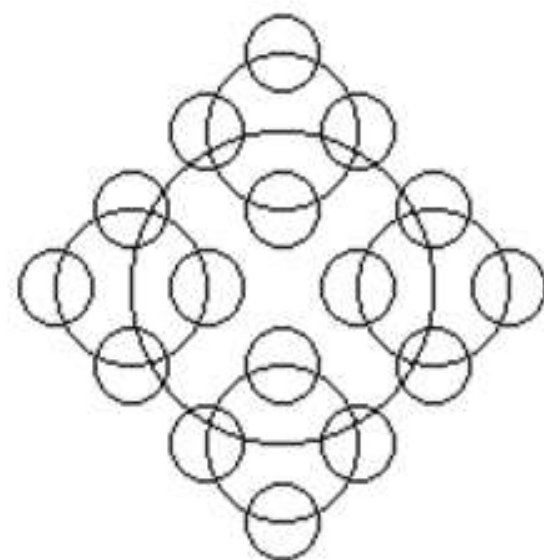
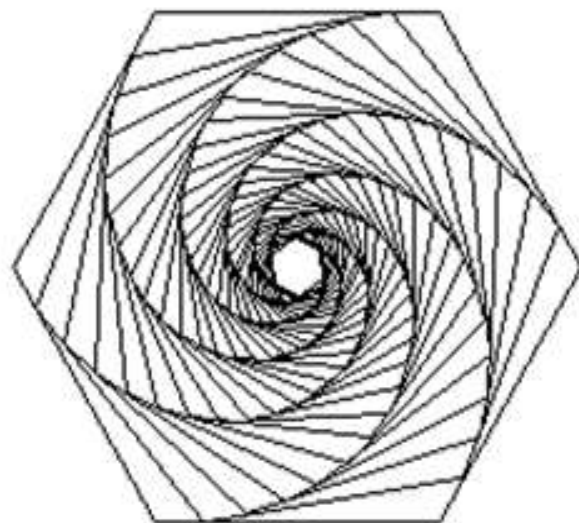
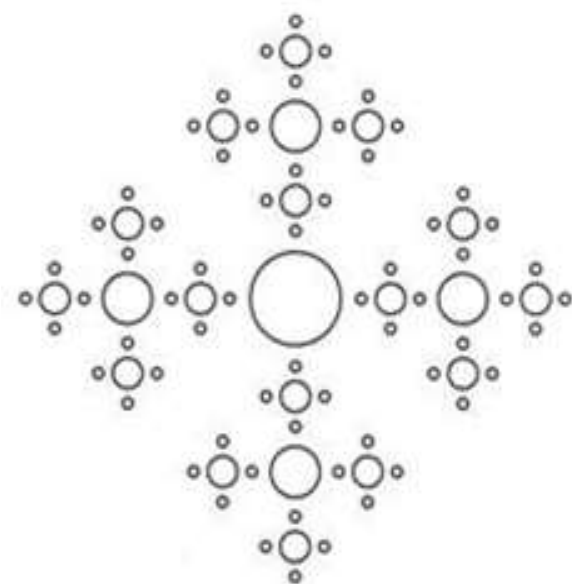
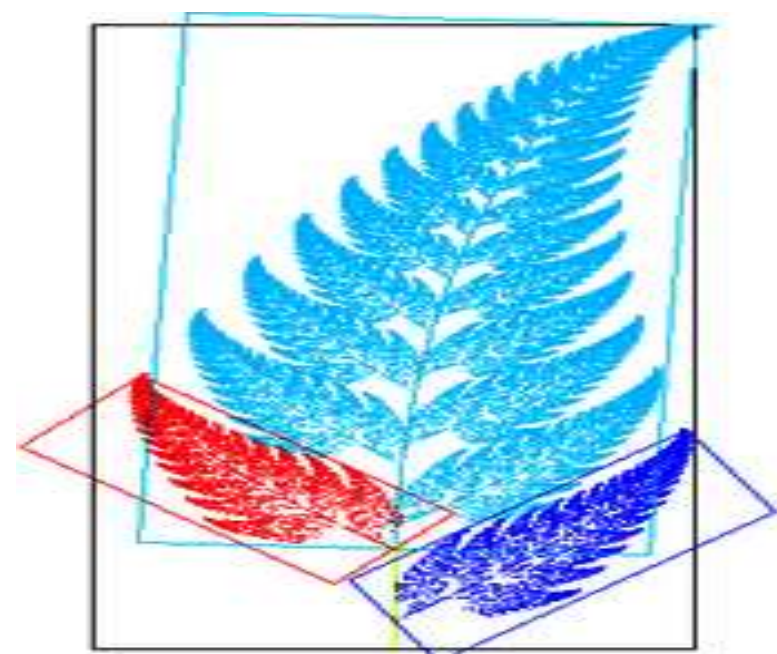
Эслатма: Алгоритм ва дастурларни тузишда рекурсиядан фойдаланиш вақтни тежайди, хажми кичик бўлади, лекин итерацион усулга нисбатан дастур секинроқ ишлайди ҳамда кўпроқ хотира талаб этади.

Рекурсив объектларга мисоллар



Серпинский учбурчаги





Nima uchun rekursiya kerak

- Aslini olganda, *har qanday rekursiv ishlangan masalani iterativ usulda ishlash mumkin va buning aksi ham to'g'ri*. Buning ustiga *rekursiv yechim har doim xotiradan qo'shimcha joy talab qiladi*.
- Shunday ekan, nima uchun unda rekursiya kerak? Albatta, buning yetarlicha sabablari bor:

- **Rekursiya deyarli hamma joyda ishlatiladi.**
- **Ba'zi holatlarda rekursiv yechim ancha soddaroq.** Ayniqsa, ba'zi masalalarning iterativ yechimi juda ham uzun bo'lib ketishi mumkin. Rekursiya esa kodni bir necha barobar qisqartirib berishi mumkin.
- **Aksariyat tuzilmalar va algoritmlarni rekursiyasiz tasavvur qilib bo'lmaydi.** Tree, Graph, Heap, QuickSort, MergeSort, ... Bu ro'yhatni juda uzoq davom ettirish mumkin. Ayniqsa, murakkab tuzilmalar bo'lgan Tree va Graphlarda rekursiya har qadamda uchraydi. Dasturchilikni esa ularsiz tasavvur qilib bo'lmaydi, bu esa o'z o'rnida rekursiya qanchalik muhimligini belgilab beradi.
- **Rekursiya funksional dasturlashning asosiy elementlaridan hisoblanadi.**
- shunday dasturlash tillari borki ularda umuman takrorlanish operatorlari yo'q va bu borada butunlay rekursiyaga tayanadi. **Haskell** va **Erlang** shular jumlasidan.

Рекурсив триада

- **параметризация қилиш** – масала шартини таснифлаш ва уни ҳал этиш учун параметрлар аниқланади (масала кўлаמידан келиб чиқадиб яъни ҳар сафар масала кўлами камайиши керак);
- **рекурсия базаси (асоси)** – масала ечими аниқ бўлган тривиал (тўхтайдиган) ҳолат аниқланади, яъни бу ҳолатда функцияни ўзига мурожаат қилиши талаб этилмайди;
- **декомпозиция** (бутунни қисмларга ажратиш) – умумий ҳолатни нисбатан анча оддий бўлган ўзгарган параметрли қисм масалалар орқали ифодалайди.

Изоҳ

Рекурсив ёки итерацион усул самарадорлиги берилган масалани ҳал қилувчи дастурни турли бошланғич қийматларда таҳлил этиш орқали аниқланади.

Изоҳ

Рекурсив алгоритмларни самарадорлигини ошириш кўпинча триада босқичларини қайта кўриб чиқиш натижасида амалга ошиши мумкин.

Fibonacci soni

$$f(n) = \begin{cases} 1, & n \leq 2 \\ f(n-1) + f(n-2), & n > 2 \end{cases}$$

```
function Fib(n: Integer): Integer;  
begin  
    if (n=1) or (n=2) then  
        Result:=1  
    else  
        Result:=Fib(n-  
1)+Fib(n-2);  
end;
```

Eng kichik umumiy bo'luvchi

$$Ekub(n, m) = \begin{cases} n, & m = 0 \\ Ekub(m, n \bmod m), & m \neq 0 \end{cases}$$

1)

```
#include <iostream>
#include <bits/stdc++.h>
using namespace std;
int ekub(int n, int m)
{
    if (m==0) return n;
    else return ekub(m, n%m);
}

int main()
{
    cout << "EKUB= " << ekub(16, 8) << endl;
    return 0;
}
```

2)

```
#include <iostream>
#include <bits/stdc++.h>

using namespace std;
int ekuk (int n, int m)
{ while ((n!=0) && (m!=0))
    { if (n>=m) n=n%m;
      else m=m%n;
    }
    return n+m;
}

int main()
{ cout << "Ekuk=" << ekuk(12, 8) << endl;
  return 0;
}
```




Рекурсия – простой пример

```
void babushka ()
{
    babushka ();
    printf("k");
}

int main() {
    babushka ();
    return 0;
}
```

Сколько раз будет распечатана буква «k»?

Этот алгоритм поведёт себя как бесконечный цикл. Ничего не будет распечатано. В конце концов программа упадёт из-за недостатка памяти (причины будут рассмотрены более подробно позже).

Первый вызов

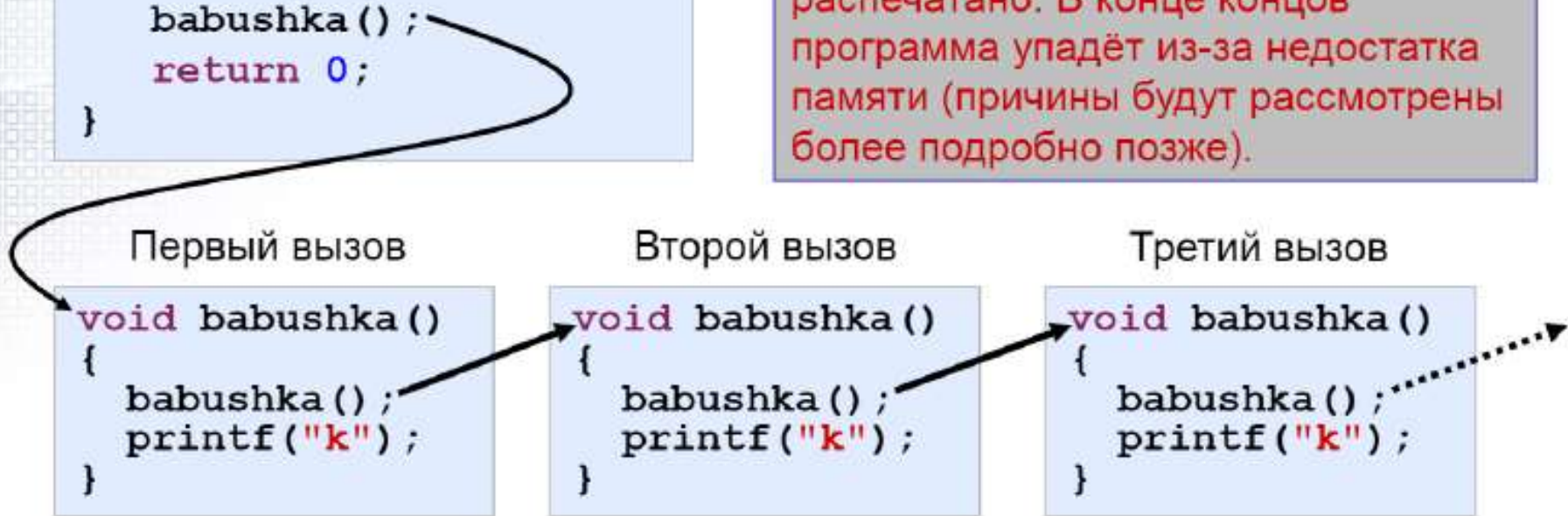
```
void babushka ()
{
    babushka ();
    printf("k");
}
```

Второй вызов

```
void babushka ()
{
    babushka ();
    printf("k");
}
```

Третий вызов

```
void babushka ()
{
    babushka ();
    printf("k");
}
```



O'rin almashtirsa nima bo'ladi?

```
void babushka ()  
{  
    printf("k");  
    babushka();  
}  
  
int main() {  
    babushka();  
    return 0;  
}
```

Сколько раз будет распечатана «k»?

Этот алгоритм тоже поведёт себя как бесконечный цикл. Но на этот раз со множеством распечаток буквы «k». И опять же в конце концов программа упадёт из-за недостатка памяти.

Первый вызов

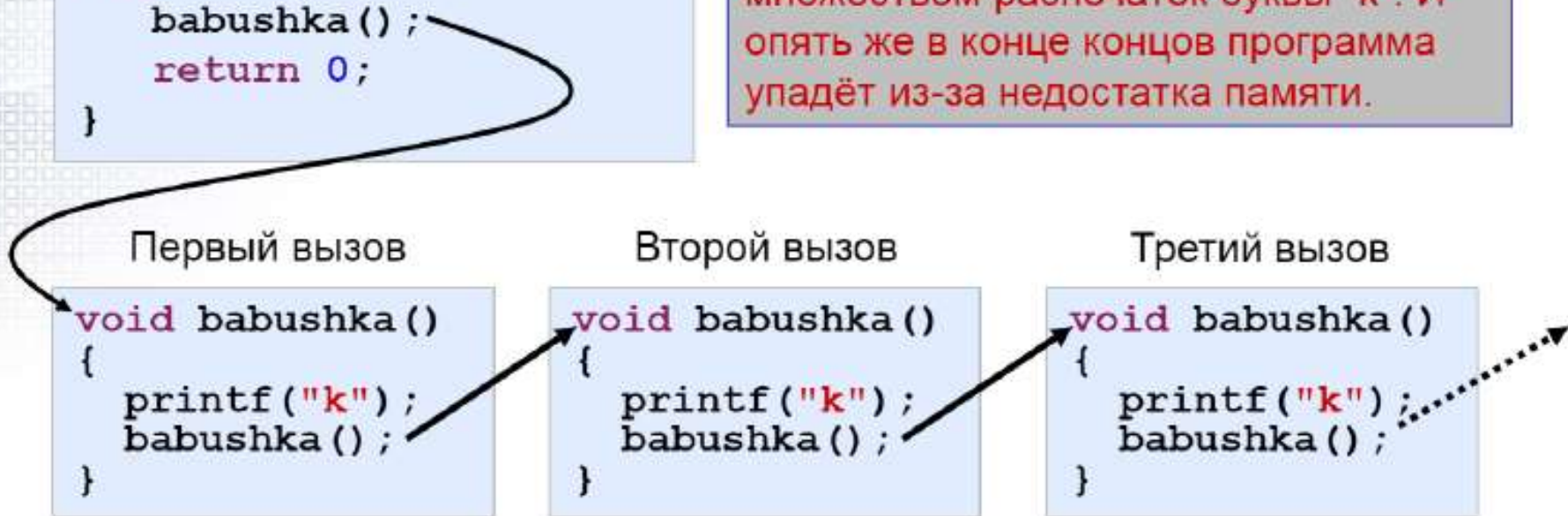
```
void babushka ()  
{  
    printf("k");  
    babushka();  
}
```

Второй вызов

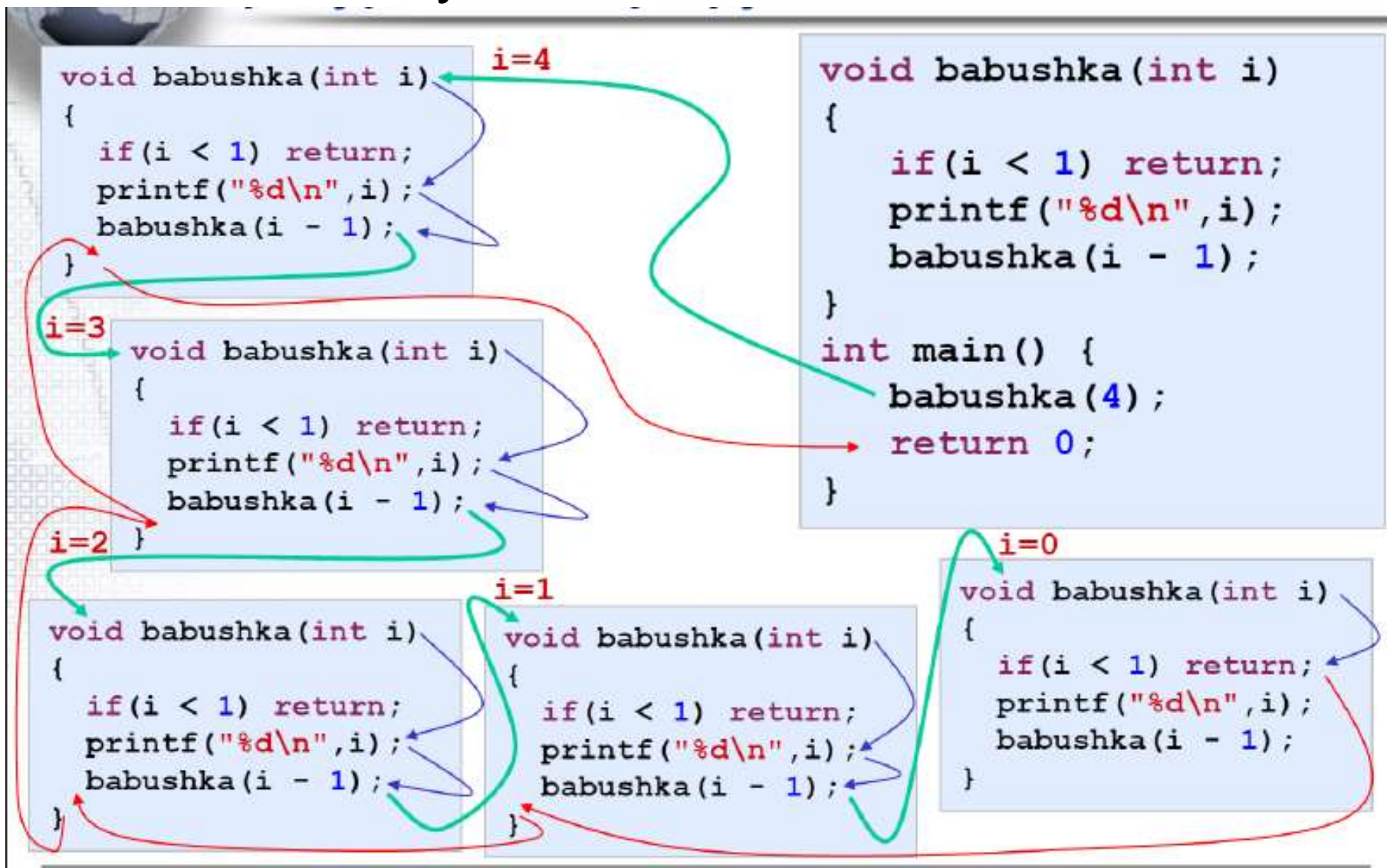
```
void babushka ()  
{  
    printf("k");  
    babushka();  
}
```

Третий вызов

```
void babushka ()  
{  
    printf("k");  
    babushka();  
}
```



Rekursiyani to'xtatish shartini kiritish





Влияние условия остановки рекурсии

```
void babushka(int i)
{
    if(i < 1) {
        return;
    }
    printf("%d\n", i);
    babushka(i - 1);
}

int main() {
    babushka(4);
    return 0;
}
```

0

Будет распечатано:

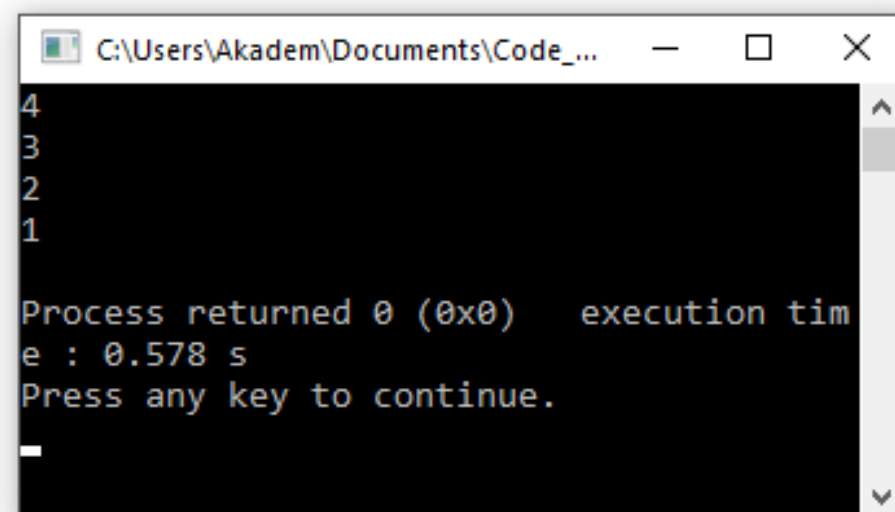
4

3

2

1

```
1  #include <iostream>
2  #include <bits/stdc++.h>
3
4  using namespace std;
5
6  void babushka (int i)
7  {
8      if (i<1) return ;
9      printf("%d\n",i);
10     babushka(i-1);
11
12 }
13 int main()
14 {
15     babushka(4);
16     return 0;
17 }
18
```



```
C:\Users\Akadem\Documents\Code_...
4
3
2
1
Process returned 0 (0x0)   execution time
e : 0.578 s
Press any key to continue.
_
```




Аналогичным образом:

`void babushka(int i)`
{
 `if(i < 1) return;`
 `babushka(i - 1);`
 `printf("%d\n", i);`
}

i=4

i=3

`void babushka(int i)`
{
 `if(i < 1) return;`
 `babushka(i - 1);`
 `printf("%d\n", i);`
}

i=2

`void babushka(int i)`
{
 `if(i < 1) return;`
 `babushka(i - 1);`
 `printf("%d\n", i);`
}

i=1

`void babushka(int i)`
{
 `if(i < 1) return;`
 `babushka(i - 1);`
 `printf("%d\n", i);`
}

i=0

`void babushka(int i)`
{
 `if(i < 1) return;`
 `babushka(i - 1);`
 `printf("%d\n", i);`
}

`void babushka(int i)`
{
 `if(i < 1) return;`
 `babushka(i - 1);`
 `printf("%d\n", i);`
}

`int main() {`
 `babushka(4);`
 `return 0;`
}



Вывод на печать

```
void babushka(int i)
{
    if(i < 1) {
        return;
    }
    babushka(i - 1);
    printf("%d\n", i);
}

int main() {
    babushka(4);
    return 0;
}
```

0

Будет выведено:

1

2

3

4

```

4   int fact(int n)
5   {
6       if (n==0) return 1;
7       if (n==1) return 1;
8       return n*fact(n-1);
9   }
10  int sum(int i, int a[], int n1)
11  {
12      if (i==n1-1) return a[i];
13      else
14          return a[i]+sum(i+1,a,n1);
15  }
16  int ekub(int m, int n)
17  {
18      if (n==0) return m;
19      printf(" a ");
20      return ekub(n,m%n);
21  }
22
23  int main()
24  {
25      int a[]={1,2,3,4,5};
26      cout<<"factorial="<<fact(4)<<endl;
27      cout<<"Yig'indi="<<sum(0,a,5)<<endl;
28      cout<<"EKUB="<<ekub(66,42)<<endl;
29      // cout<<"EKUK="<<ekuk(44,66)<<endl;
30      //cout << "Hello world!" << endl;
31      return 0;
32  }

```

Рекурсив триада:

Параметризация:

n – номанфий бутун

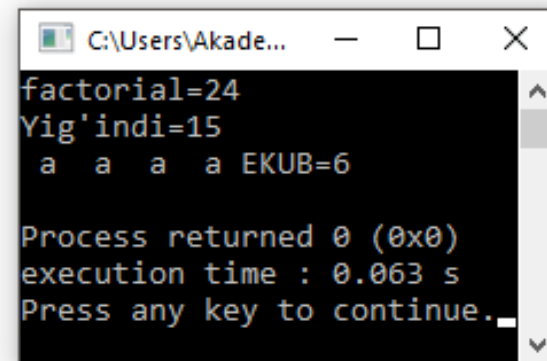
сон.

Рекурсия базаси:

n = 0 ва n = 1 учун факториал 1
га тенг.

Декомпозиция:

$n! = (n-1)! * n$.



```

C:\Users\Akade...
factorial=24
Yig'indi=15
a a a a EKUB=6

Process returned 0 (0x0)
execution time : 0.063 s
Press any key to continue.

```

- Aslida, ko'p hollarda dasturchilar rekursiya ishlatishdan qochishadi. Buning asosiy sabablari esa:
- **Rekursiya har doim xotiradan qo'shimcha joy talab qiladi.**
- **Rekursiv yechimda xato qilib ehtimoli yuqori.** rekursiya juda ham chalg'ituvchi. Shu sababli, uni ishlatishda osongina xato qilib qo'yish mumkin.
- **Rekursiv yechimni xatosini topish qiyin.** Bunday masalalarda xato qilib qo'yish ehtimoli yuqori bo'lishi bilan birga uni topib to'g'irlash ham qiyin bo'lishi mumkin. Buning asosiy sababi, bunday yechimlarni tasavvur qilib olish juda qiyin.
- **Rekursiv algoritmnining murakkabligini hisoblash ko'pincha juda murakkab.** Hattoki, ba'zan to'g'ri yechimni yozishning o'zi ham kam bo'lib qolishi mumkin. Rekursiv algoritmlarda bu ko'pincha juda murakkab va yaxshigina matematika talab qiladi.

5-мавзу бўйича назорат саволлари

1. *Рекурсия нима?*
2. *Рекурсив объект, алгоритм, функция тушунчаси.*
3. *Рекурсив триада.*
4. *Рексурсив алгоритм самарадорлигини аниқлаш ва ошириш йўллари.*