3 Ден

Рекурзија 1

Вовед во рекурзија:

Постојат 2 начина за составување ПОВТОРУВАЧКИ алгоритми:

-Итерација (Iteration)
Се повторуваат само параметрите на алгоритмот, а не и самиот алгоритам.

-Рекурзија (Recursion)
Повторувачки процес во кој алгоритмот СЕ ПОВИКУВА СЕБЕСИ. Алгоритмот се појавува во сопствената дефиниција .

Рекурзија VS итерација:

-Повторување
- Итерација: експлицитни циклуси
- Рекурзија: функциски повици

-Прекин на повторувањето
- Итерација: условот за повторување повеќе не важи
- Рекурзија: се препознава основниот случај

И во обата случаи можни се појави на бесконечно повторување

Што е рекурзија?

Рекурзивна функција претставува функција која се повикува самата себеси. Кај рекурзивните функции, секогаш (ама секогаш!) мора да наведеме услов за завршување на рекурзијата. Во спротивно, функцијата ќе продолжи да се повикува самата себеси до бесконечност - практично, додека не снема меморија или оперативниот систем не ја сруши нашата програма.

Креирање на рекурнтни алгоритми:

Секоја рекурзивен алгоритам МОРА да има основен(граничен случај):

-Изразот што го решава проблемот
- пр. factoriel(0)

-Остатокот од алгоритмот се нарекува општ случај (рекурентна врска)
- n · factoriel(n-1)

-За да се напише рекурзивен алгоритам:
- Одредете го основниот случај factoriel(0)
- Одредете го општиот случај (рекурентната врска) n factoriel(n-1)

-СЕКОЈ рекурзивен повик морада реши
- Дел од проблемот
- ИЛИ

Да ја редуцира големината на проблемот

Кога не треба да се користи рекурзија?

-Ако се одговори со НЕ на било кое од следните прашања:
Дали алгоритмотили податочните структури природно се зададени со рекурзивна формула?
Дали рекурзивното решение е пократко и поразбирливо?
Дали рекурзивното решение се одвива во прифатливи временски и просторни граници?
- Рекурзивните алгоритмигенерално се ПОСПОРИ од итеративните алгоритми
Функциските повици земаат повеќе време отколку инструкција во циклус

Пример:

Да разгледаме една функција за пресметување на факториел - производ на сите природни броеви помали или еднакви на N. Факториелот на природниот број N (се означува N!) го дава бројот на пермутации на одредено множество со N елементи (на колку начини може да се наредат N различни елементи во листа). На пример, 3! = 1*2*3 = 6 и постојат 6 различни начини да се наредат 3 елементи во листа: [1, 2, 3], [1, 3, 2], [2, 1, 3], [2, 3, 1], [3, 1, 2], [3, 2, 1].

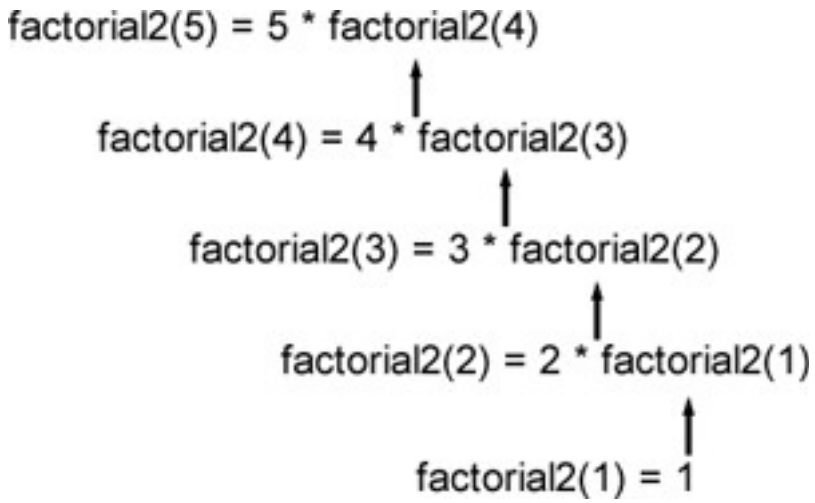
-Итеративен алгоритам
Factoriel(n)=n (n-1) (n-2) … 2 1
Пр. Factoriel(4)=4*3*2*1 = 24

```
int factoriel(int n)
{
    int r = 1;

    for (int i=2; i<=n; i++)
        r = r*i;

    return r;
}
```

-Рекурзивен алгоритам
Начин на пресметување
5! = 5 * 4! (поедноставување на проблемот)
4! = 4 * 3! ...
По дефиниција 1! = 0! = 1 (основен случај)
Тогаш важи
2! = 2 * 1! = 2 * 1 = 2;
3! = 3 * 2! = 3 * 2 = 6;
46

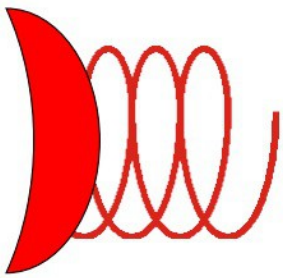
```
int factoriel(int n)
{
	if (n <= 1)
		return 1;
	else
		return n*factoriel(n-1);
}
```

Првата функција е стандардна и неа нема посебно да ја разгледуваме. Но, погледнете ја втората функција - factorial2(int n). Таа се повикува самата себеси и го користи резултатот кој истата го враќа за параметар (n-1). Во конкретниот случај (за пресметување на факториел), ова важи бидејќи N! = N * (N-1)!, или 5! = 5 * (4!) = 5 * 4 * 3 * 2 * 1.

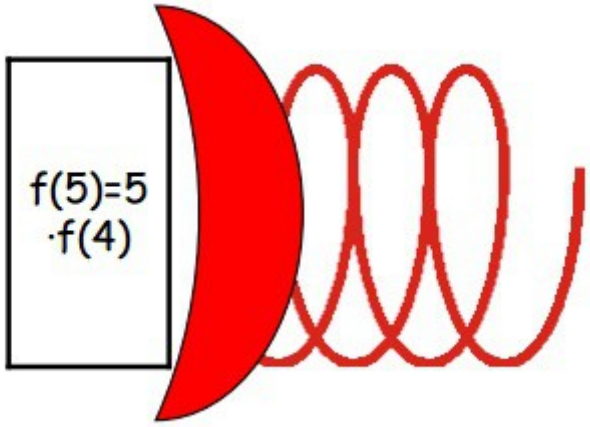


-Патување во две насоки

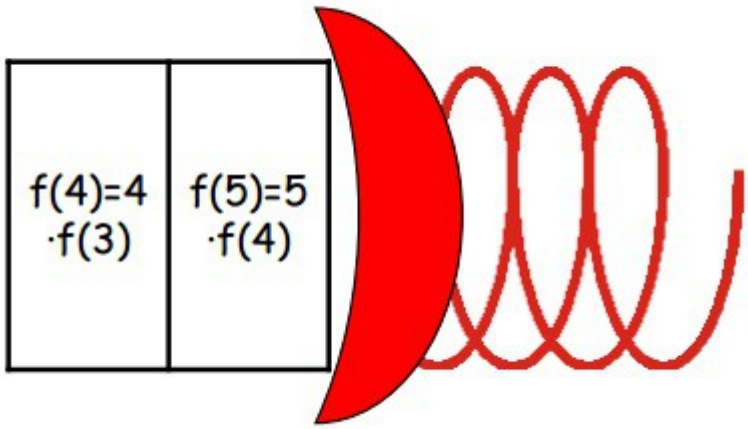
Прво –се разложува од врвот кон дното(од n до 1)
Второ-се решава одејќи од дното кон врвот(од 1 до n)



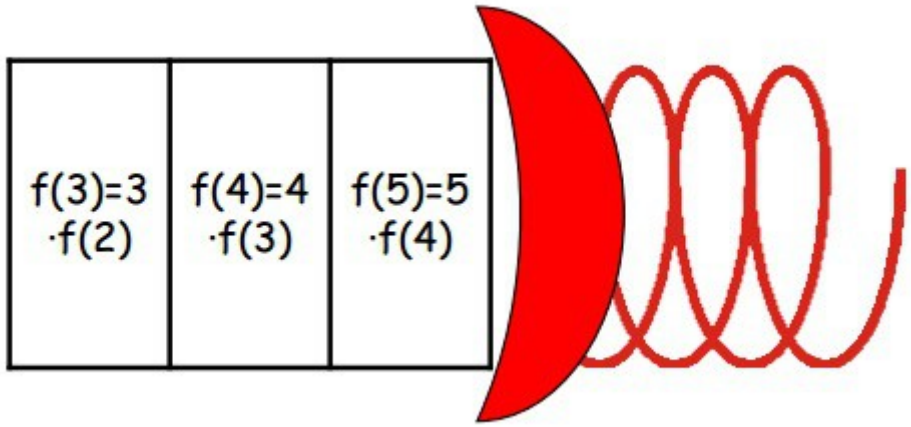
Да разгледаме што се случува при повикот на factoriel(5):



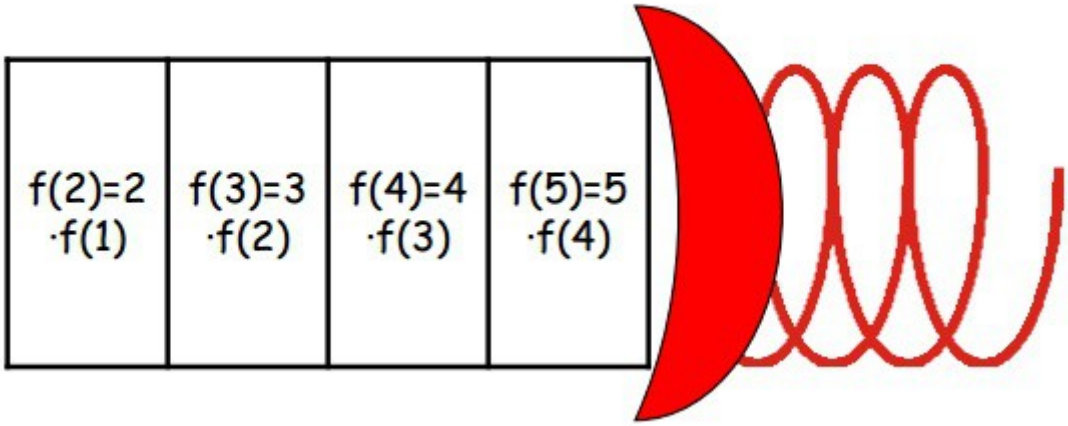
се проверува дали (n=5) <= 1. Бидејќи не е, потребно е да се пресмета колку е факториел од 4, за да може да се пресмета вредноста на 5!. Се упатува повик до factoriel(4).



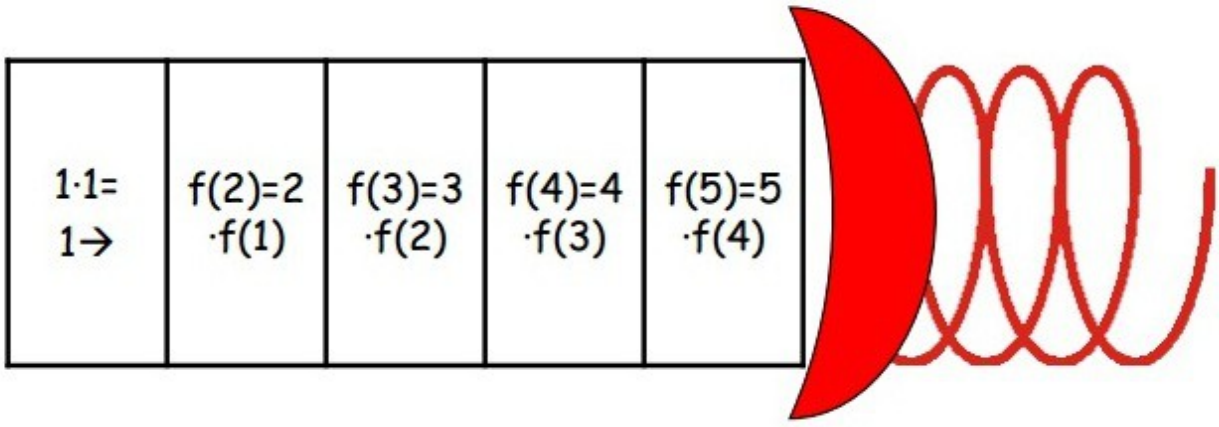
се проверува дали (n=4) <= 1. Бидејќи не е, потребно е да се пресмета колку е факториел од 3, за да може да се пресмета вредноста на 4!. Се упатува повик до factoriel(3).



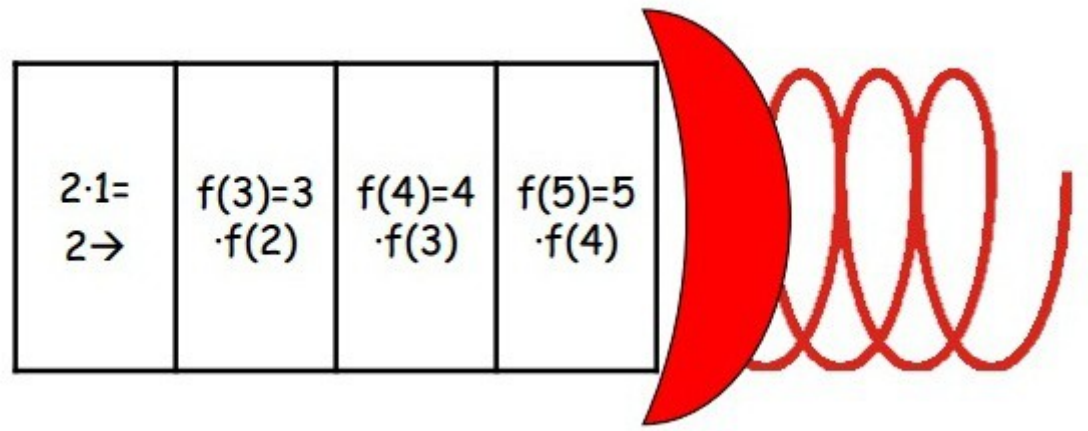
се проверува дали (n=3) <= 1. Бидејќи не е, потребно е да се пресмета колку е факториел од 2, за да може да се пресмета вредноста на 3!. Се упатува повик до factoriel(2).



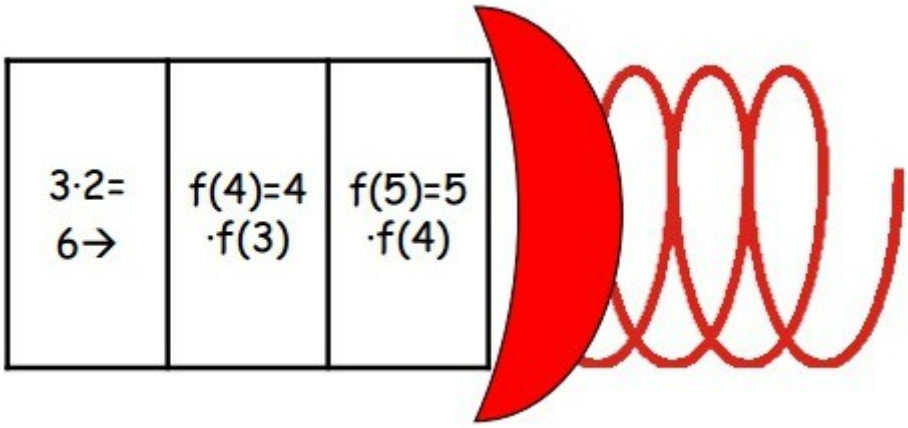
се проверува дали (n=2) <= 1. Бидејќи не е, потребно е да се пресмета колку е факториел од 1, за да може да се пресмета вредноста на 2!. Се упатува повик до factoriel(1).



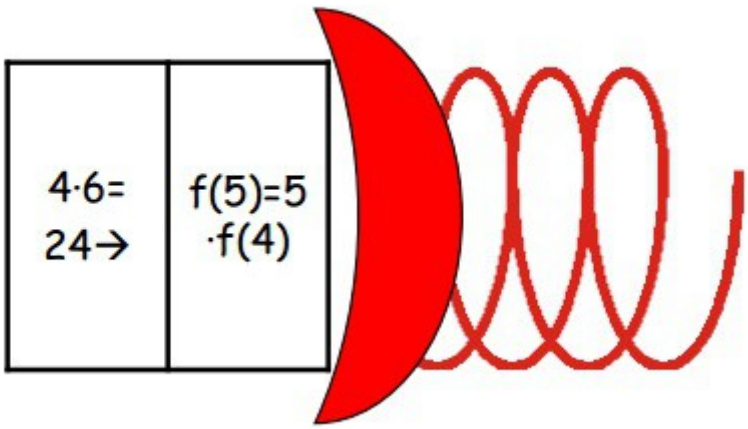
се проверува дали (n=1) <= 1. Бидејќи условот е исполнет, функцијата враќа резултат 1.



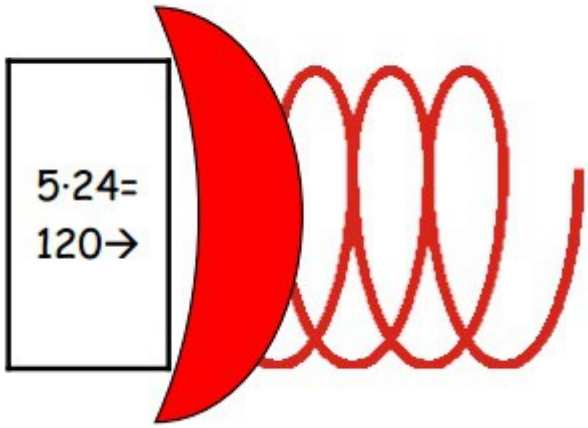
Резултатот вратен во чекор 5, при повикот factoriel(1), се користи при пресметување на factoriel(2), така што 2 се множи со добиената вредност од factoriel(1) = 1, во чекор 5. Добиениот резултат 2*factoriel(1) = 2*1 = 2, се враќа како резултат на повикот factoriel(2).



Резултатот вратен во чекор 6, при повикот factoriel(2), се користи при пресметување на factoriel(3), така што 3 се множи со добиената вредност од factoriel(2) = 2, во чекор 6. Добиениот резултат 3*factoriele(2) = 3*2 = 6, се враќа како резултат на повикот factoriel(3).



Резултатот вратен во чекор 7, при повикот factoriel(3), се користи при пресметување на factoriel(4), така што 4 се множи со добиената вредност од factoriel(3) = 6, во чекор 7. Добиениот резултат 4*factoriel(3) = 4*6 = 24, се враќа како резултат на повикот factoriel(4).



Резултатот вратен во чекор 8, при повикот factoriel(4), се користи при пресметување на factoriel(5), така што 5 се множи со добиената вредност од factoriel(4) = 24, во чекор 8. Добиениот резултат 5*factoriel(4) = 5*24 = 120, се враќа како резултат на повикот factoriel(5), што всушност е и првобитниот повик упатен од страна на функцијата main().