

# Рекурсия

## ИТ Кариера



Учителски екип  
Обучение за ИТ кариера

<https://it-kariera.mon.bg/e-learning>

# Съдържание

- Рекурсивна реализация на цикли
- Опашкова рекурсия



# Рекурсивна реализация на цикли

- В Haskell няма цикли
- Циклите се реализират чрез рекурсия
- В Haskell всички функции са чисти (не могат да променят състояние)
- Итерациите се реализират с рекурсивни извиквания
- Итераторите се реализират като параметри и се променят при всяко рекурсивно извикване

# Рекурсивна реализация на цикли

```
int repeatString(String str, int n) {  
    String result = "";  
    for(int i = 0; i < n; ++i)  
        result += str;  
  
    return result;  
}
```

Java реализация на функция, която използва цикъл, който долепя даден символен низ до себе си n на брой пъти

# Рекурсивна реализация на цикли

```
repeatString str n =  
    if n == 0  
    then ""  
    else str ++ (repeatString str (n-1))
```

реализация на същата  
функция в Haskell



# Рекурсивна реализация на цикли

- За функциите, които зависят от външно състояние се използва помощна функция

```
pow2loop n x i =  
    if i < n  
    then pow2loop n (x*2) (i+1)  
    else x
```

```
pow2 n = pow2loop n 1 0
```

# Рекурсивна реализация на цикли

- За функциите, които зависят от външно състояние се използва помощна функция

параметърът *i* е итератора, който би се използвал при for-цикъл в процедурен език

```
pow2loop n x i =  
    if i < n  
    then pow2loop n (x*2) (i+1)  
    else x
```

```
pow2 n = pow2loop n 1 0
```

# Рекурсивна реализация на цикли

- За функциите, които зависят от външно състояние се използва помощна функция

```
pow2loop n x i =  
  if i < n  
  then pow2loop n (x*2) (i+1)  
  else x
```

параметърът *i* е итератора, който би се използвал при for-цикъл в процедурен език

```
pow2 n = pow2loop n 1 0
```

pow2 приема само един аргумент - на коя степен да се повдигне 2; За начален индекс на цикъла се задава 0, а за начална стойност на произведението се задава 1



## Задача:

- Дефинирайте функция, която намира сбора на първите 10 естествени числа

## Решение:

```
sumNumbers = sumNumbersLoop 0 1
```

```
sumNumbersLoop sum index =
```

```
    if index > 10
```

```
    then sum
```

```
    else (sum + index) + (sumNumbersLoop sum  
    (index + 1))
```

# Опашкова рекурсия

- Опашковата рекурсия е рекурсия, при която последното извършвано действие е рекурсивно извикване
- Оптимизация наречена премахване на опашното извикване (tail call elimination)
- Вместо с последващо връщане рекурсивното обръщение се реализира със преход без връщане
- При тази рекурсия заделената в стека памет се преизползва вместо да се заделя нова
- Намалява разхода на памет и обикновено подобрява бързината на алгоритъма, но по-трудно се откриват грешки

# Опашкова рекурсия

- Обратно към примера с функцията repeatString - тя може да се преобразува като се използва опашкова рекурсия по следния начин:

```
repeatStringLoop string result n =  
    if n == 0  
    then result  
    else repeatStringLoop string (result ++  
string) (n - 1)
```

```
repeatString string n = repeatStringLoop  
string string n
```

# Опашкова рекурсия

- Обратно към примера с функцията repeatString - тя може да се преобразува като се използва опашкова рекурсия по следния начин:

```
repeatStringLoop string result n =  
    if n == 0  
    then result  
    else repeatStringL  
string) (n - 1)
```

отново се използва помощна функция, на която се подават 2 параметъра - стринга и колко повторения трябва да се направят

```
repeatString string n = repeatStringLoop  
string string n
```



# Опашкова рекурсия

- Обратно към примера с функцията `repeatString` - тя може да се преобразува като се използва опашкова рекурсия по следния начин:

```
repeatStringLoop string result n =  
    if n == 0  
    then result  
    else repeatStringL  
string) (n - 1)
```

функцията от своя страна извиква рекурсивния цикъл като задава стойности за низа, който ще се повтаря, текущия низ (в началото със същата стойност) и броя повторения, които се изискват

```
repeatString string n = repeatStringLoop  
string string n
```

# Опашкова рекурсия

- Обратно към примера с функцията преобразува като се използва опашка

```
repeatStringLoop string n
  if n == 0
  then result
  else repeatStringLoop string (result ++
string) (n - 1)
```

```
repeatString string n = repeatStringLoop
string string n
```

забележете как вместо функцията да долепя низа със резултата от рекурсивното извикване на функция за  $(n - 1)$  директно извиква функцията като ѝ подава низа (досега), залепен с началният низ, който трябва да се повтаря

# Министерство на образованието и науката (МОН)

- Настоящият курс (презентации, примери, задачи, упражнения и др.) е разработен за нуждите на Национална програма "**Обучение за ИТ кариера**" на МОН за подготовка по професия "Приложен програмист"



Министерство  
на образованието  
и науката



Национална  
програма  
„Обучение за  
ИТ кариера“

- Курсът се разпространява под свободен лиценз **CC-BY-NC-SA**

