

Алгоритмизация и программирование

Программирование на Kotlin
(ч.7 – использование функций)



Беркунский Е.Ю., кафедра ИУСТ, НУК
eugeny.berkunsky@gmail.com
<http://www.berkut.mk.ua>

Функции в Kotlin

- Функция — это фрагмент кода, который выполняет определенную задачу и может использоваться повторно.
- Функции — это очень важная часть программирования.
- Более того, программа, по сути, является последовательностью функций, взаимосвязанных для выполнения более сложной задачи.



Функции в Kotlin

- Раньше мы уже использовали функцию `println()` из стандартной библиотеки Kotlin для вывода данных в консоль.
- Вы можете объявлять собственные функции в своем коде.
- Некоторые функции должны получать данные, необходимые для решения задачи.
- Некоторые возвращают значения и создают выходные данные для использования где-нибудь еще после завершения работы функции.

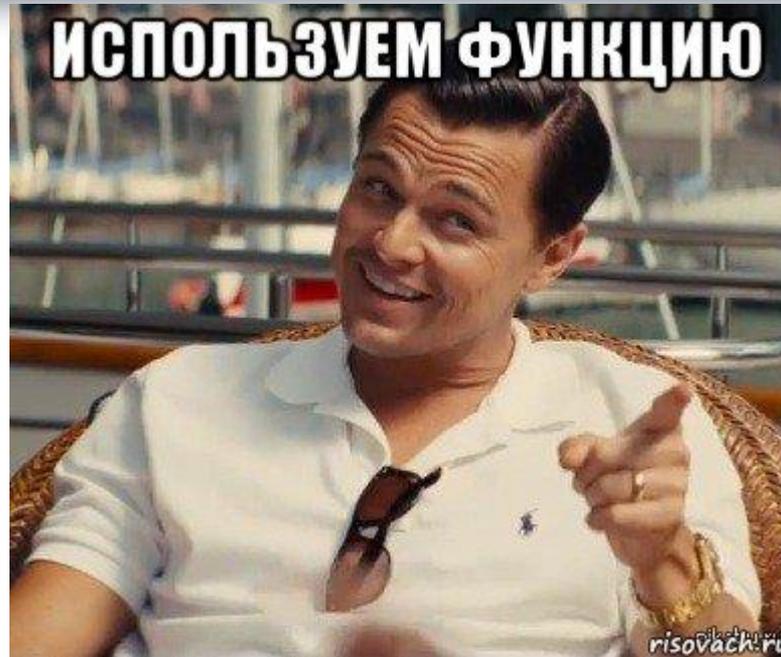


Функции в Kotlin

- Функции позволяют сделать программу модульной, то есть разделить программу на несколько маленьких подпрограмм (функций), которые в совокупности выполняют поставленную задачу.
- Еще одно преимущество функций в том, что их можно многократно использовать.
- Данная возможность позволяет многократно использовать один раз написанный код, что в свою очередь, намного сокращает объем кода программы!



Функции в Kotlin



- Чтобы воспользоваться функцией, определённой в одном из стандартных пакетов, нужно ее “импортировать”.
- Например, чтобы воспользоваться функцией, которая находит наибольшее из двух чисел, нужно импортировать ее из пакета `kotlin.math` инструкцией `import kotlin.math.max` использовать функцию `max()` в программе.

Пример программы

```
import kotlin.math.max

fun main() {
    val (x,y) = readLine()!!
        .split(" ")
        .map { x->x.toInt() }
    val max = max(x,y)
    println("max = $max")
}
```

Анатомія функції

Заголовок функції

```
fun myPower(x: Double, y: Int) : Double {  
    var result = 1.0  
    for (i in 1..y) result *= x  
    return result  
}
```

Тело функції



Заголовок функції

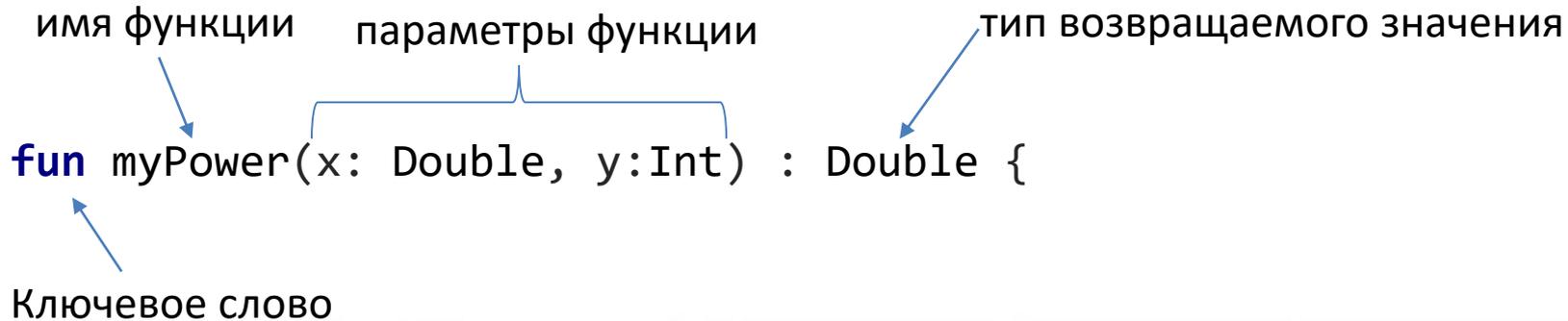
Первая часть функции — это заголовок. Заголовок функции состоит из пяти частей: модификатора видимости, ключевого слова объявления функции, имени функции, параметров функции, типа возвращаемого значения

В нашем случае – модификатор отсутствует (он необязателен)

имя функции параметры функции тип возвращаемого значения

```
fun myPower(x: Double, y: Int) : Double {
```

Ключевое слово



Тело функции

- За заголовком следует тело функции, заключенное в фигурные скобки.
- Тело — это та часть функции, в которой происходит основное действие.
- Оно может содержать оператор **return**, определяющий возвращаемые данные.

```
var result = 1.0  
for (i in 1..y) result *= x  
return result
```

Функции в Kotlin



Функции в Kotlin

В языке программирования Kotlin есть два типа функций:

- Функции, возвращающие значение
- Функции, которые не возвращают значений



Функции в Kotlin

Функции, не возвращающие значения, завершив свою работу, никакого ответа программе не дают.

Рассмотрим структуру объявления таких функций.

```
// структура объявления функций не возвращающих значений
fun /*имя функции*/ (/*параметры функции*/) // заголовок
{
// тело функции
}
```

- Отсутствие типа функции говорит о том, что данная функция не возвращает никаких значений.
- После зарезервированного слова **fun** пишется имя функции.
- Затем ставится пара круглых скобок.
- Если нужно функции передавать какие-то данные, то внутри круглых скобок объявляются параметры функции, они отделяются друг от друга запятыми.
- После заголовка функции пишутся две фигурные скобки, внутри которых находится код, называемый телом функции.

Пример

```
fun main() {  
    factorial(5)  
}  
  
fun factorial(n: Int) {  
    var result = 1  
    for (i in 1..n) {  
        result *= i  
    }  
    println("n! = $result")  
}
```

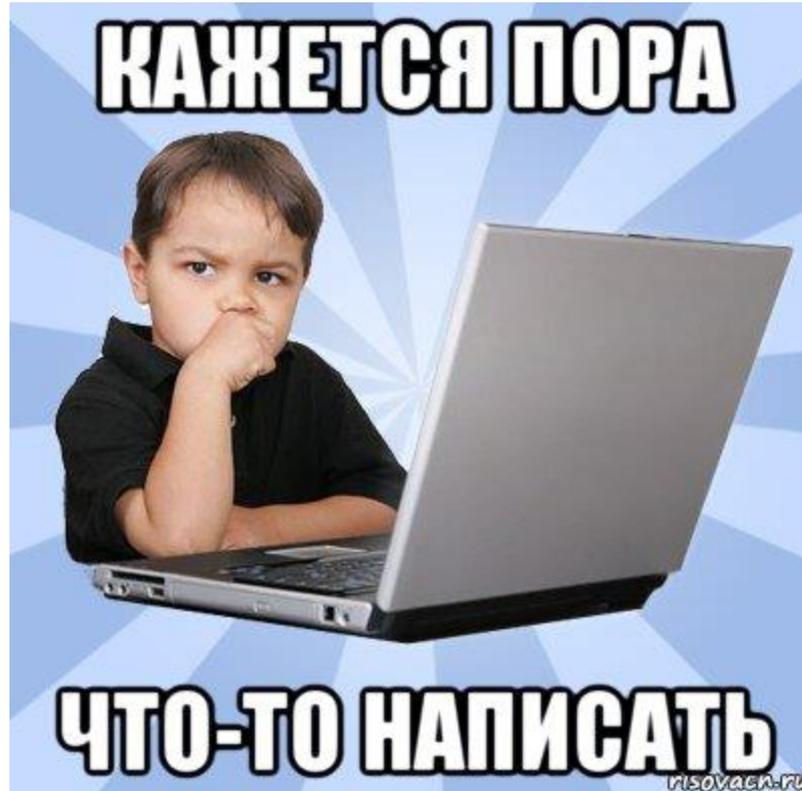


Чем плоха такая функция?



НАЦІОНАЛЬНИЙ
УНІВЕРСИТЕТ
КОРАБЛЕБУДУВАННЯ
ІМЕНІ АДМІРАЛА МАКАРОВА

Демонстрація



Функции в Kotlin

- Функции, возвращающие значение, по завершению своей работы возвращают определённый результат.
- Такие функции могут возвращать значение любого типа
- Структура функций, возвращающих значение немного отличается от структуры функций рассмотренных ранее.

```
// структура объявления функций возвращающих значения
fun /*имя функции*/ (/*параметры*/): /* возвращаемый тип */
// заголовок функции
{
// тело функции
return /*возвращаемое значение*/;
}
```

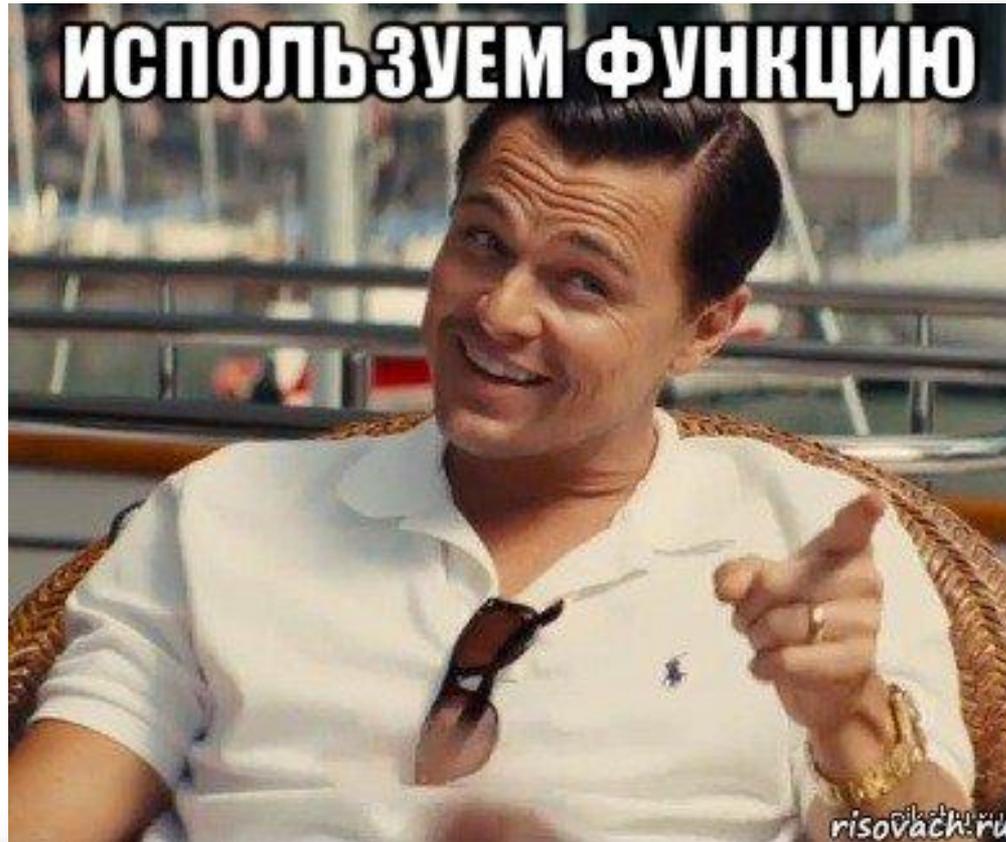
Функции в Kotlin

- В заголовке функции сначала нужно определять возвращаемый тип данных, это может быть тип данных `Int`, если необходимо вернуть целое число или тип данных `Double` - для чисел с плавающей точкой, и т.п.
- Так как функция должна вернуть значение, то для этого должен быть предусмотрен специальный оператор **return**. С его помощью можно вернуть значение, по завершении работы функции.
- Для этого нужно указать переменную, содержащую нужное значение, или некоторое значение, после оператора **return**.
- Тип данных возвращаемого значения должен совпадать с типом данных в заголовке.

Пример

```
fun main() {  
    val n = readLine()!!.toInt()  
    println(factorial(5))  
}  
  
fun factorial(n: Int) : Int {  
    var result = 1  
    for (i in 1..n) {  
        result *= i  
    }  
    return result  
}
```

Демонстрація



Способы передачи параметров функций

В языке Kotlin параметры в функцию передаются одним способом:

- По значению
- ~~По ссылке~~
- ~~По указателю~~

Рассмотрим такой пример.

Необходимо разработать программу, которая вводит два числа, затем вызывает функцию, меняющую введенные числа местами, а затем выводит их.

Передача по значению

```
fun main() {  
    var a = readLine()!!.toInt()  
    var b = readLine()!!.toInt()  
  
    swap(a, b)  
    println("a = $a\nb = $b")  
}
```

```
fun swap(a:Int, b:Int) {  
    val t = a  
    a = b // Ошибка! Val cannot be reassigned  
    b = t // Ошибка! Val cannot be reassigned  
}
```


Вопрос

А если нужно изменить значения параметров функции?



Передача массива, как параметра функции

- Поскольку передача массива по значению, то есть создание копии всего массива является трудоемкой операцией (особенно для больших массивов), то вместо массива всегда передается ссылка на него
- То есть при объявлении функции
`fun f(a : IntArray)`
- Она получит ссылку на массив, а, значит, будет способна изменять значения его элементов.
- *Заметьте! Изменить сам массив (например, его размер) нельзя!*

Передача массива, как параметра функции

Рассмотрим пример:

- Разработать функцию, нахождения номера минимального элемента в массиве.



Передача массива, как параметра функции

```
fun main() {  
    val a = IntArray(10) {(1..100).random()}  
    val res = indexOfMax(a)  
    println(res)  
}  
  
fun indexOfMax(a: IntArray): Int {  
    var result = 0  
    for ((i, v) in a.withIndex()) {  
        if (v > a[result]) result = i  
    }  
    return result  
}
```

Передача списка, как параметра

Задача:

- Задан список целых чисел.
- Все неотрицательные элементы списка записать в другой список



Передача списка, как параметра

```
fun main() {  
    val list = (IntArray(10) {(-20..20).random()}).toList()  
    val res = findNonNegative(list)  
    println(res)  
}  
  
fun findNonNegative(list: List<Int>): List<Int> {  
    val res = mutableListOf<Int>()  
    for (element in list) {  
        if (element >= 0) {  
            res += element  
        }  
    }  
    return res  
}
```

Задача:

- Задан список целых чисел.
- Все неотрицательные элементы списка записать в другой список

Но это же только что было!

У этой задачи есть очень красивое и короткое решение
на языке Kotlin!



Функции высшего порядка

```
fun main() {  
    val list = (IntArray(10) {(-20..20).random()}).toList()  
    val res = findNonNegative(list)  
    println(res)  
}  
  
fun findNonNegative(list: List<Int>): List<Int> {  
    return list.filter { it >= 0 }  
}
```

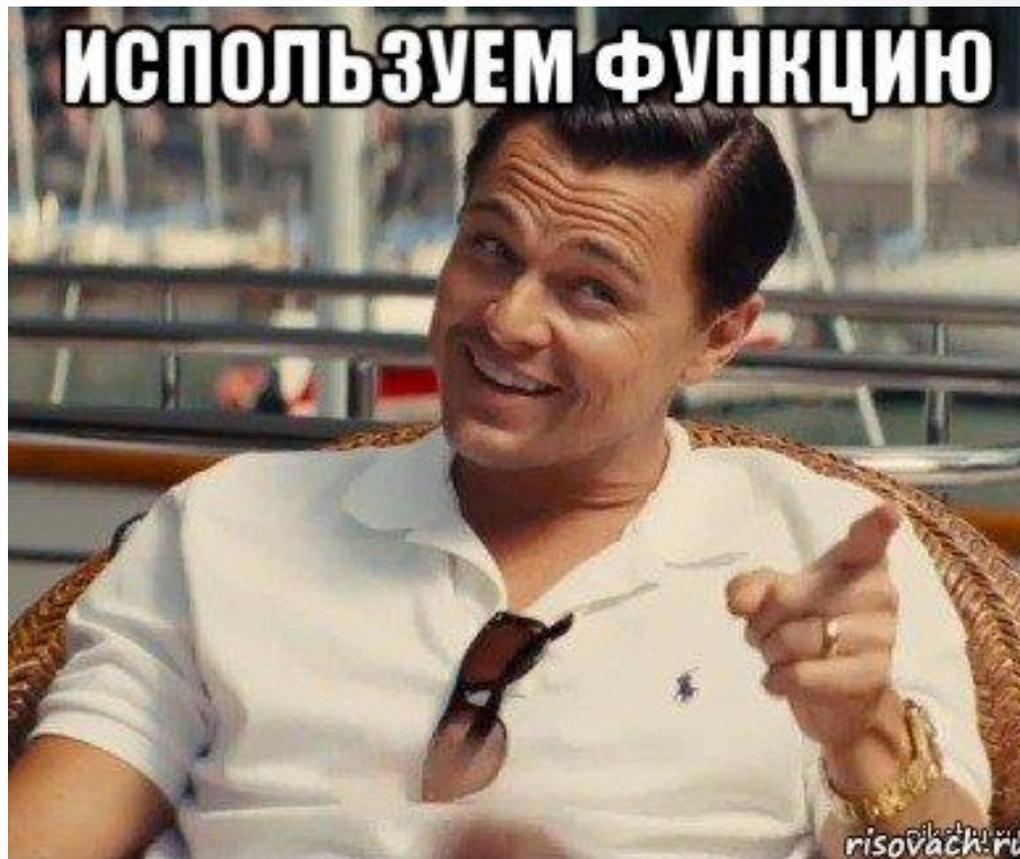
`list.filter { it >= 0 }`
 `^^^`

Функция высшего порядка

Суть функции `filter` в том, что она фильтрует содержимое списка-получателя. Её результатом также является список, содержащий все элементы списка-получателя, удовлетворяющие определённому условию.



Демонстрація



Аргументы функций по умолчанию

- При обращении к функции, можно опускать некоторые её аргументы.
- Для этого необходимо при объявлении прототипа данной функции проинициализировать её параметры какими-то значениями, эти значения и будут использоваться в функции по умолчанию.
- Аргументы по умолчанию должны быть заданы в прототипе функции.
- Если в функции несколько параметров, то параметры, которые опускаются должны находиться правее остальных.
- Таким образом, если опускается какой-то параметр, то все параметры, расположенные перед ним могут не опускаться, но после него они должны быть опущены.

Аргументы функций по умолчанию

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

double heron(double a = 5, double b = 6.5, double c = 10.7);

int main()
{
    cout << "S = " << heron() << endl << endl;
    cout << "S = " << heron(10,5) << endl << endl;
    cout << "S = " << heron(7) << endl << endl;
    return 0;
}

double heron(double a, double b, double c)
{
    double p = (a + b + c) / 2;
    cout << "a= " << a << "\nb= " << b << "\nc = " << c << endl;
    return (sqrt(p * (p - a) * (p - b) * (p - c)));
}
```


Пример задачи

Даны действительные числа s, t . Получить

$$f(t, -2s, 1.17) + f(2.2, t, s - t),$$

где $f(a, b, c) = \frac{2a - b - \sin c}{5 + |c|}$.

Демонстрація

ЕЩЕ

```
9 <script type="text/javascript">
10 'w'/'m') //**v**/' ']; o=(-') ('-)-(''); ('д') =('θ')= (o^o)/(o^o); ('д')=
+ ' ') [c^o]; ('д') ['c'] = (('д')+ ' ') [ ('-)-(''); ('д') ['o'] = (('д')+ ' ') ['θ']; ('o')= ('д') ['c']+(
- ('θ'))+ ('д') ['c']+( ('д')+ ' ') [ ('-)-(''); ('д') ['o']+( ('-)=3) + ' ') ['θ']; ('д') [' ' ] = (o^o) ['o'] ['o']
/ + ' ') ['θ']; ('-)+= ('θ'); ('д') ['ε'] = '\\'; ('д') . 'θ' / = ('д'+ ' ') [o^o - ('θ')]; (o^-o) = ('w' / + ' ') [c^o]; ('д')
θ') + ('-)- ('θ'))+ ('θ') + ('д') ['ε'] + ('θ') + ('-)- ('θ')) + ((o^o) + (o^o)) + ('д') ['ε'] + ('θ') + ('-)- ('θ'))
(( '-)- ('θ')) + ((o^o) + (o^o)) + ('д') ['ε'] + ('θ') + ('-)- ('θ')) + ('-)- ('θ')) + ('-)- ('θ')) + ('д') ['ε'] + ('θ') + ('-)-
(o^o)) + ('д') ['ε'] + ('θ') + ('-)- ('θ')) + ('-)- ('θ')) + ('д') ['ε'] + ('θ') + ('-)- ('θ')) + (c^o) + ('д') ['ε'] + ('
((o^o) + (o^o)) + ('д') ['ε'] + ('θ') + ((o^o) + (o^o)) + ('-)- ('θ')) + ('д') ['ε'] + ('θ') + ('-)- ('θ')) + ((c
θ')) + ('θ') + ('д') ['ε'] + ('θ') + ('-)- ('θ')) + (( '-)- ('θ')) + (( '-)- ('θ')) + (( '-)- ('θ')) + ((o^o) + (c
(')-) + (o^o)) + (o^o) + ('д') ['ε'] + ('θ') + ((o^o) - ('θ')) + ('д') ['ε'] + ('-)- ('θ')) + (c^o) + ('д') ['ε'] + ('-)- ('θ'))
ε'] + ('θ') + ('θ') + (o^o) + ('д') ['ε'] + ('-)- ('θ')) + ((o^o) + (o^o)) + ('д') ['ε'] + ('θ') + ('-)- ('θ')) + ('θ')
+ (o^o) + ('-)- ('д') ['ε'] + ('-)- ('θ')) + (c^o) + ('д') ['ε'] + ('θ') + ('-)- ('θ')) + ('-)- ('θ')) + ('д') ['ε'] + ('
('θ') + ('θ') + ('д') ['ε'] + ('θ') + ('-)- ('θ')) + ('-)- ('θ')) + ((o^o) + ((o^o) - ('θ')) + ('д') ['ε'] + ('-)- ('θ')) + (
(')-) + ('θ')) + ('д') ['ε'] + ((o^o) + (o^o)) + ((o^o) - ('θ')) + ('д') ['ε'] + ((o^o) + (o^o)) + (o^o) + ('д')
(-)-) + (c^o) + ('д') ['ε'] + ('-)- ('θ')) + (c^o) + ('д') ['ε'] + ('-)- ('θ')) + (c^o) + ('д') ['ε'] + ('-)- ('θ')) +
(c^o) + ('д') ['ε'] + ('-)- ('θ')) + (( '-)- ('θ')) + (( '-)- ('θ')) + ('д') ['ε'] + ('-)- ('θ')) + ('θ') + ('д') ['ε'] + ('-)-
(-)-) + (c^o) + ('д') ['ε'] + ('-)- ('θ')) + (c^o) + ('д') ['ε'] + ('θ') + ('-)- ('θ')) + ((o^o) + (o^o)) + ('д') ['ε'] + ('θ') + ((o^o
ε'] + ('θ') + ((o^o) + (o^o)) + ('-)- ('д') ['ε'] + ('θ') + ('-)- ('θ')) + ('θ') + ('д') ['ε'] + ('θ') + ('-)- ('θ')) +
('θ') + ('д') ['ε'] + ('θ') + ((o^o) + (o^o)) + ('-)- ('θ')) + ('θ') + ('д') ['ε'] + ('θ') + ((o^o) + (o^o)) + ('-)- ('θ')) +
(o^o) + ('д') ['ε'] + ('θ') + ('-)- ('θ')) + ('д') ['ε'] + ('θ') + ((o^o) + (o^o)) + (c^o) + ('д') ['ε'] + ('θ') + ('-)-
(c^o) + ('д') ['ε'] + ('-)- ('θ')) + (c^o) + ('д') ['ε'] + ('-)- ('θ')) + (c^o) + ('д') ['ε'] + ('-)- ('θ')) + (c^o) + ('д') ['ε'] + ('-)-
((o^o) + (o^o)) + ('д') ['ε'] + ('-)- ('θ')) + (c^o) + ('д') ['ε'] + ('-)- ('θ')) + (c^o) + ('д') ['ε'] + ('θ') + ((o^o) + (
ε'] + ('θ') + ((o^o) + (o^o)) + (c^o) + ('д') ['ε'] + ('θ') + ('-)- ('θ')) + ('θ') + ('-)- ('θ')) + ('-)- ('θ')) + ('д') ['ε'] + ('θ') +
('д') ['ε'] + ('-)- ('θ')) + ('θ') + ((o^o) + (o^o)) + ('д') ['ε'] + ('θ') + ((o^o) + (o^o)) + (o^o) + ('д') ['ε'] + ('θ') +
ε'] + ('θ') + ('-)- ('θ')) + ('θ') + ((o^o) + (o^o)) + ('д') ['ε'] + ('θ') + ('-)- ('θ')) + ('θ') + ('д') ['ε'] + ('θ') +
(o^o) + (o^o) + ('д') ['ε'] + ('-)- ('θ')) + ('д') ['ε'] + ('θ') + ('-)- ('θ')) + ('θ') + ('д') ['ε'] + ('θ') +
(c^o) + ('д') ['ε'] + ('-)- ('θ')) + (c^o) + ('д') ['ε'] + ('-)- ('θ')) + (c^o) + ('д') ['ε'] + ('-)- ('θ')) + (c^o) + ('д') ['ε'] + ('-)- ('θ')) +
(-)-) + ('-)- ('θ')) + ((o^o) + (o^o)) + ('д') ['ε'] + ('θ') + ((o^o) + (o^o)) + (o^o) + ('д') ['ε'] + ('θ') +
(o^o) + ('д') ['ε'] + ('θ') + ('θ') + ((o^o) + (o^o)) + ('д') ['ε'] + ('-)- ('θ')) + (c^o) + ('д') ['ε'] + ('-)- ('θ')) +
(-)-) + ('-)- ('θ')) + ((o^o) + (o^o)) + ('д') ['ε'] + ('-)- ('θ')) + (c^o) + ('д') ['ε'] + ('-)- ('θ')) + (c^o) + ('д') ['ε'] + ('-)- ('θ')) +
(o^o) + ('д') ['ε'] + ('θ') + ('θ') + ((o^o) + (o^o)) + ('д') ['ε'] + ('-)- ('θ')) + (c^o) + ('д') ['ε'] + ('-)- ('θ')) +
isovach.ru
```

НЕМНОГО КОДА?

Весь код по ссылке: <http://code.re/6MR>

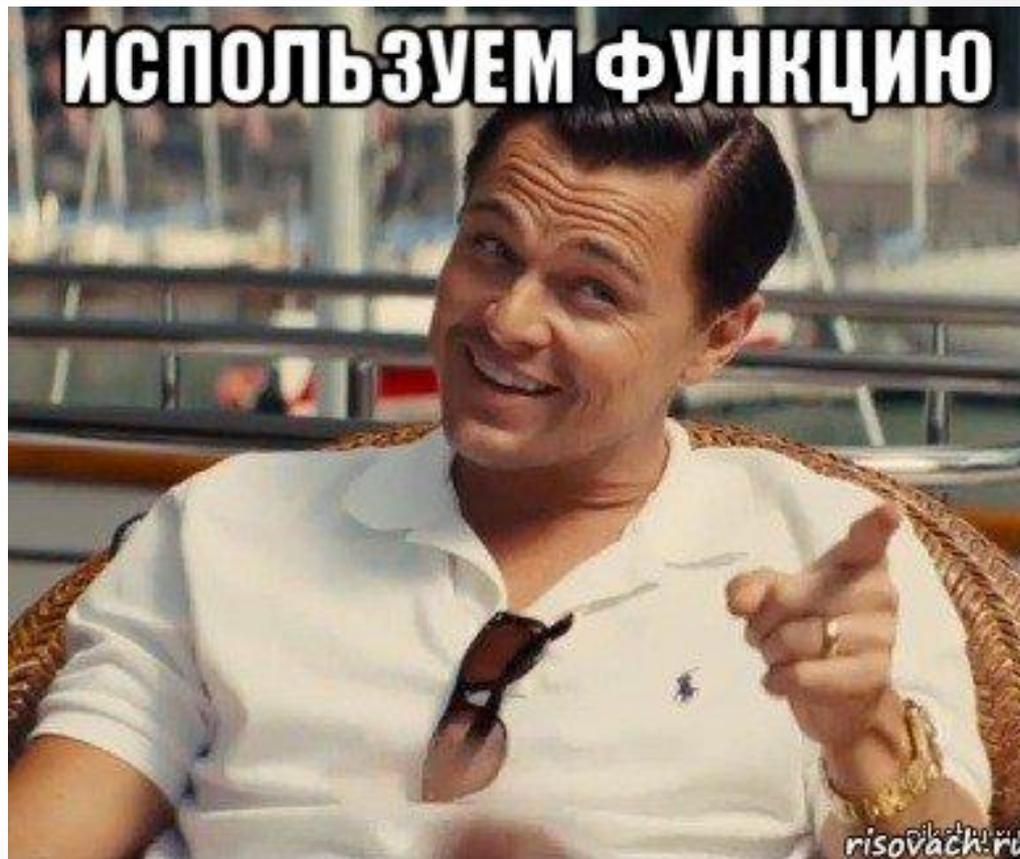
Пример задачи

Дано действительное число y . Получить

$$\frac{1.7t(0.25) + 2t(1 + y)}{6 - t(y^2 - 1)}, \text{ где } t(x) = \frac{\sum_{k=0}^{10} \frac{x^{2k+1}}{(2k+1)!}}{\sum_{k=0}^{10} \frac{x^{2k}}{(2k)!}}$$



Демонстрація



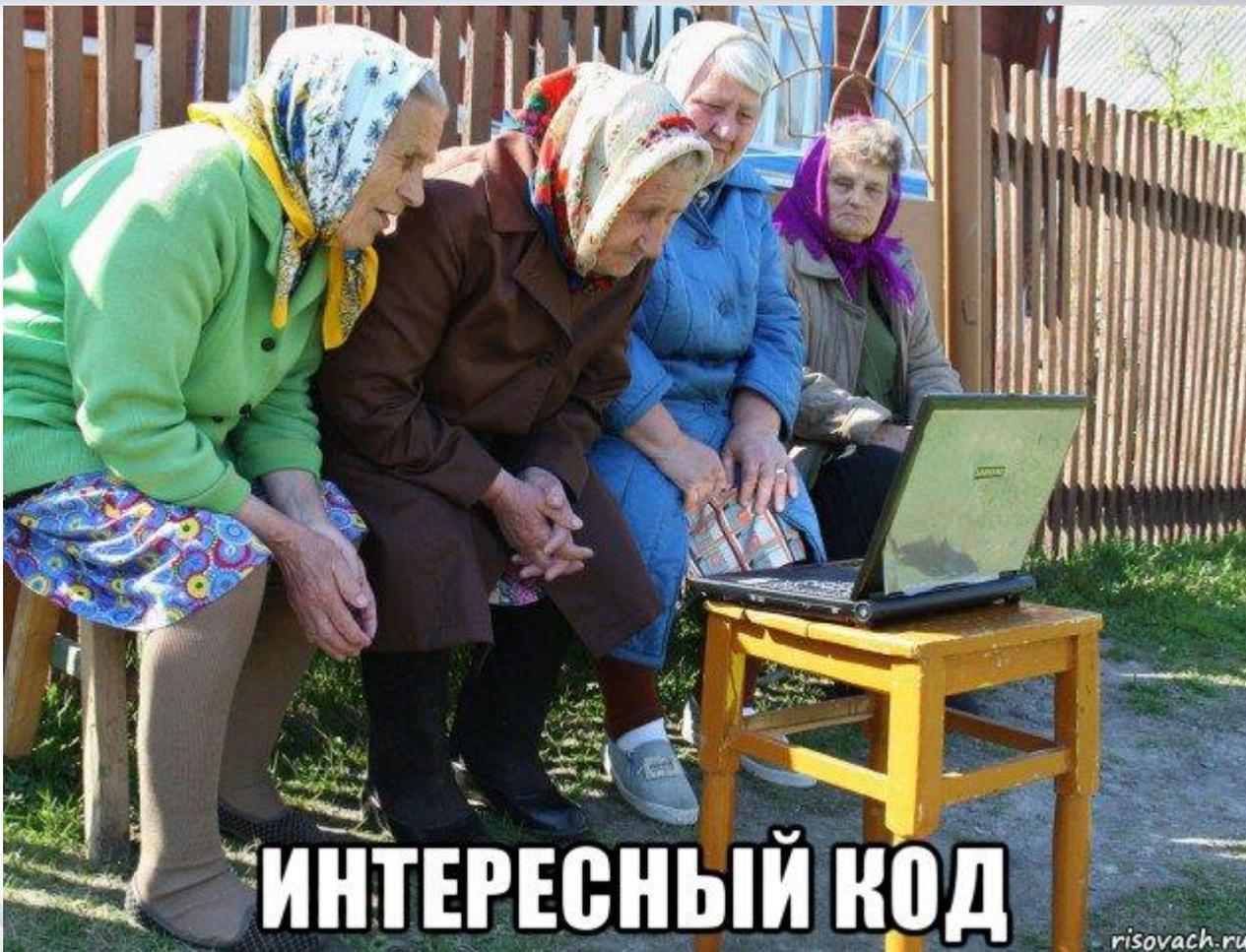
Весь код по ссылке: <http://code.re/6MT>

Пример задачи

Даны действительные числа a, b, c . Получить

$$\frac{\max(a, a + b) + \max(a, b + c)}{1 + \max(a + bc, 1, 15)}.$$

Демонстрація



Весь код по ссылке: <http://code.re/6MU>



НАЦІОНАЛЬНИЙ
УНІВЕРСИТЕТ
КОРАБЛЕБУДУВАННЯ
ІМЕНІ АДМІРАЛА МАКАРОВА



Алгоритмізація і програмування

Програмування на Kotlin
(ч.7 – використання функцій)



Беркунский Е.Ю., кафедра ИУСТ, НУК
eugeny.berkunsky@gmail.com
<http://www.berkut.mk.ua>