

In [7]:

```
()
favoritelang = ("Python", "Julia", "R")
# кортеж из целых чисел:
x1 = (1, 2, 3)
# кортеж из элементов разных типов:
x2 = (1, 2.0, "tmp")
# именованный кортеж:
x3 = (a=2, b=1+2)
# Примеры операций над кортежами:
# длина кортежа x2:
@show length(x2)
# обратиться к элементам кортежа x2:
@show x2[1], x2[2], x2[3]
# произвести какую-либо операцию (сложение)
# с вторым и третьим элементами кортежа x1:
@show c = x1[2] + x1[3]
# обращение к элементам именованного кортежа x3:
@show x3.a, x3.b, x3[2]
# проверка вхождения элементов tmp и 0 в кортеж x2
# (два способа обращения к методу in()):
@show in("tmp", x2), 0 in x2
```

```
length(x2) = 3
(x2[1], x2[2], x2[3]) = (1, 2.0, "tmp")
c = x1[2] + x1[3] = 5
(x3.a, x3.b, x3[2]) = (2, 3, 3)
("tmp" in x2, 0 in x2) = (true, false)
```

Out[7]:

(true, false)

In [10]:

```
phonebook = Dict{"Иванов И.И." => ("867-5309", "333-5544"),
  "Бухгалтерия" => "555-2368"}
# вывести ключи словаря:
@show keys(phonebook)
# вывести значения элементов словаря:
@show values(phonebook)
# вывести заданные в словаре пары "ключ - значение":
@show pairs(phonebook)
# проверка вхождения ключа в словарь:
@show haskey(phonebook, "Иванов И.И.")
# добавить элемент в словарь:
phonebook["Сидоров П.С."] = "555-3344"
# удалить ключ и связанные с ним значения из словаря
pop!(phonebook, "Иванов И.И.")
@show phonebook
# Объединение словарей (функция merge()):
a = Dict{"foo" => 0.0, "bar" => 42.0};
b = Dict{"baz" => 17, "bar" => 13.0};
@show merge(a, b), merge(b, a)
```

```
keys(phonebook) = ["Бухгалтерия", "Иванов И.И."]
values(phonebook) = Any["555-2368", ("867-5309", "333-5544")]
pairs(phonebook) = Dict{String, Any}{"Бухгалтерия" => "555-2368", "Иванов И.И." => ("867-5309", "333-5544")}
haskey(phonebook, "Иванов И.И.") = true
phonebook = Dict{String, Any}{"Сидоров П.С." => "555-3344", "Бухгалтерия" => "555-2368"}
(merge(a, b), merge(b, a)) = (Dict{String, Real}{"bar" => 13.0, "baz" => 17, "foo" => 0.0}, Dict{String, Real}{"bar" => 42.0, "baz" => 17, "foo" => 0.0})
```

Out[10]:

(Dict{String, Real}{"bar" => 13.0, "baz" => 17, "foo" => 0.0}, Dict{String, Real}{"bar" => 42.0, "baz" => 17, "foo" => 0.0})

In [11]:

```
@show A = Set{[1, 3, 4, 5]}
@show B = Set{"abracadabra"}
# проверка эквивалентности двух множеств:
S1 = Set{[1,2]};
S2 = Set{[3,4]};
@show issetequal(S1,S2)
S3 = Set{[1,2,2,3,1,2,3,2,1]};
```

```

S4 = Set([2,3,1]);
@show issetequal(S3,S4)
#объединение
@show C=union(S1,S2)
# пересечение множеств:
@show D = intersect(S1,S3)
# разность множеств:
@show E = setdiff(S3,S1)
# проверка вхождения элементов одного множества в другое:
@show issubset(S1,S4)
# добавление элемента в множество:
push!(S4, 99)
# удаление последнего элемента множества:
pop!(S4)
@show S4

```

```

A = Set([1, 3, 4, 5]) = Set([5, 4, 3, 1])
B = Set("abracadabra") = Set(['a', 'd', 'r', 'k', 'b'])
issetequal(S1, S2) = false
issetequal(S3, S4) = true
C = union(S1, S2) = Set([4, 2, 3, 1])
D = intersect(S1, S3) = Set([2, 1])
E = setdiff(S3, S1) = Set([3])
issubset(S1, S4) = true
S4 = Set([99, 3, 1])

```

Out[11]:

Set{Int64} with 3 elements:

```

99
3
1

```

In [14]:

```

# создание пустого массива с абстрактным типом:
empty_array_1 = []
# создание пустого массива с конкретным типом:
empty_array_2 = (Int64)[]
empty_array_3 = (Float64)[]
# вектор-столбец:
a = [1, 2, 3]
# вектор-строка:
b = [1 2 3]
# многомерные массивы (матрицы):
@show A = [[1, 2, 3] [4, 5, 6] [7, 8, 9]]
@show B = [[1 2 3]; [4 5 6]; [7 8 9]]
# одномерный массив из 8 элементов (массив $1 \times 8$)
# со значениями, случайно распределёнными на интервале [0, 1):
@show c = rand(1,8)
# многомерный массив $2 \times 3$ (2 строки, 3 столбца) элементов
# со значениями, случайно распределёнными на интервале [0, 1):
@show C = rand(2,3);
# трёхмерный массив:
@show D = rand(4, 3, 2)

```

```

A = [[1, 2, 3] [4, 5, 6] [7, 8, 9]] = [1 4 7; 2 5 8; 3 6 9]
B = [[1 2 3]; [4 5 6]; [7 8 9]] = [1 2 3; 4 5 6; 7 8 9]
c = rand(1, 8) = [0.7431163477569965 0.5917663480873426 0.6547780837178668 0.2924931265349273 0.1479220481624941 0.6383714366202
929 0.30431868916510985 0.8067359589816482]
C = rand(2, 3) = [0.9960040577543992 0.4790935502078558 0.49399717997309156; 0.042224840105775985 0.7266008349008092 0.6687541932
896643]
D = rand(4, 3, 2) = [0.27122464286936665 0.8955774915302851 0.5542817480941896; 0.36970259221788127 0.6561096692058381 0.30891183
35977675; 0.9284402105276804 0.3673174887644949 0.4890547942151139; 0.9890080919803317 0.2155065409945104 0.22973585782228834;
; 0.3669813842874181 0.17581989162059863 0.26430218206490197; 0.09239673325404774 0.1351048422617671 0.059443540045023724; 0.9
219737263316139 0.7509666161625167 0.8814504910621536; 0.7430303528296531 0.5533996366824883 0.6940130050298862]

```

Out[14]:

4×3×2 Array{Float64, 3}:

```

[:, :, 1] =
0.271225 0.895577 0.554282
0.369703 0.65611 0.308912
0.92844 0.367317 0.489055
0.989008 0.215507 0.229736

```

```

[:, :, 2] =
0.366981 0.17582 0.264302
0.0923967 0.135105 0.0594435
0.921974 0.750967 0.88145

```

0.74303 0.5534 0.694013

In [17]:

```
# массив из квадратных корней всех целых чисел от 1 до 10:
@show roots = [sqrt(i) for i in 1:10]
# массив с элементами вида 3*x^2,
# где x - нечётное число от 1 до 9 (включительно)
@show ar_1 = [3*i^2 for i in 1:2:9]
# массив квадратов элементов, если квадрат не делится на 5 или 4:
@show ar_2 = [i^2 for i = 1:10 if i^2 % 5 != 0 && i^2 % 4 != 0]
```

```
roots = [sqrt(i) for i = 1:10] = [1.0, 1.4142135623730951, 1.7320508075688772, 2.0, 2.23606797749979, 2.449489742783178, 2.6457513110645907, 2.8284271247461903, 3.0, 3.1622776601683795]
ar_1 = [3 * i ^ 2 for i = 1:2:9] = [3, 27, 75, 147, 243]
ar_2 = [i ^ 2 for i = 1:10 if i ^ 2 % 5 != 0 && i ^ 2 % 4 != 0] = [1, 9, 49, 81]
```

Out[17]:

4-element Vector{Int64}:

```
1
9
49
81
```

In [22]:

```
# одномерный массив из пяти единиц:
ones(5)
# двумерный массив 2x3 из единиц:
@show ones(2,3)
# одномерный массив из 4 нулей:
@show zeros(4)
# заполнить массив 3x2 цифрами 3.5
@show fill(3.5,(3,2))
# заполнение массива посредством функции repeat():
@show repeat([1,2],3,3)
@show repeat([1 2],3,3)
# преобразование одномерного массива из целых чисел от 1 до 12
# в двумерный массив 2x6
a = collect(1:12)
b = reshape(a,(2,6))
# транспонирование
@show b"
# транспонирование
@show c = transpose(b)
```

```
ones(2, 3) = [1.0 1.0 1.0; 1.0 1.0 1.0]
zeros(4) = [0.0, 0.0, 0.0, 0.0]
fill(3.5, (3, 2)) = [3.5 3.5; 3.5 3.5; 3.5 3.5]
repeat([1, 2], 3, 3) = [1 1 1; 2 2 2; 1 1 1; 2 2 2; 1 1 1; 2 2 2]
repeat([1 2], 3, 3) = [1 2 1 2 1 2; 1 2 1 2 1 2; 1 2 1 2 1 2]
(b)' = [1 3 5 7 9 11; 2 4 6 8 10 12]
c = transpose(b) = [1 2; 3 4; 5 6; 7 8; 9 10; 11 12]
```

Out[22]:

6×2 transpose{::Matrix{Int64}} with eltype Int64:

```
1 2
3 4
5 6
7 8
9 10
11 12
```

In [27]:

```
# массив 10x5 целых чисел в диапазоне [10, 20]:
@show ar = rand(10:20, 10, 5)
# выбор всех значений строки в столбце 2:
ar[:, 2]
# выбор всех значений в столбцах 2 и 5:
ar[:, [2, 5]]
# все значения строк в столбцах 2, 3 и 4:
ar[:, 2:4]
# значения в строках 2, 4, 6 и в столбцах 1 и 5:
ar[[2, 4, 6], [1, 5]]
# значения в строке 1 столбца 2 до последнего столбца:
```

```

# значения в строке 1 от столбца 5 до последнего столбца.
@show ar[1, 3:end]
# сортировка по столбцам:
@show sort(ar,dims=1)
# сортировка по строкам:
@show sort(ar,dims=2)
# поэлементное сравнение с числом
# (результат - массив логических значений):
@show ar .> 14
# возврат индексов элементов массива, удовлетворяющих условию:
@show findall(ar .> 14)

```

```

ar = rand(10:20, 10, 5) = [15 16 20 10 13; 17 10 15 13 18; 18 10 20 13 12; 15 11 20 14 16; 11 14 16 20 19; 10 15 13 15 15; 13 13 10 18 12; 12 18 16
11; 11 12 15 18 13; 13 11 20 14 17]
ar[1, 3:end] = [20, 10, 13]
sort(ar, dims = 1) = [10 10 10 10 11; 11 10 13 13 12; 11 11 15 13 12; 12 11 15 13 13; 13 12 16 14 13; 13 13 16 14 15; 15 14 20 15 16; 15 15 20 18 17
17 16 20 18 18; 18 18 20 20 19]
sort(ar, dims = 2) = [10 13 15 16 20; 10 13 15 17 18; 10 12 13 18 20; 11 14 15 16 20; 11 14 16 19 20; 10 13 15 15 15; 10 12 13 13 18; 11 12 13 16 18
11 12 13 15 18; 11 13 14 17 20]
ar .> 14 = Bool[1 1 1 0 0; 1 0 1 0 1; 1 0 1 0 0; 1 0 1 0 1; 0 0 1 1 1; 0 1 0 1 1; 0 0 0 1 0; 0 1 1 0 0; 0 0 1 1 0; 0 0 1 0 1]
findall(ar .> 14) = CartesianIndex{2}[CartesianIndex(1, 1), CartesianIndex(2, 1), CartesianIndex(3, 1), CartesianIndex(4, 1), CartesianIndex(1, 2), Cart
esianIndex(6, 2), CartesianIndex(8, 2), CartesianIndex(1, 3), CartesianIndex(2, 3), CartesianIndex(3, 3), CartesianIndex(4, 3), CartesianIndex(5, 3), C
artesianIndex(8, 3), CartesianIndex(9, 3), CartesianIndex(10, 3), CartesianIndex(5, 4), CartesianIndex(6, 4), CartesianIndex(7, 4), CartesianIndex(9,
4), CartesianIndex(2, 5), CartesianIndex(4, 5), CartesianIndex(5, 5), CartesianIndex(6, 5), CartesianIndex(10, 5)]

```

Out[27]:

```

24-element Vector{CartesianIndex{2}}:
 CartesianIndex{1, 1}
 CartesianIndex{2, 1}
 CartesianIndex{3, 1}
 CartesianIndex{4, 1}
 CartesianIndex{1, 2}
 CartesianIndex{6, 2}
 CartesianIndex{8, 2}
 CartesianIndex{1, 3}
 CartesianIndex{2, 3}
 CartesianIndex{3, 3}
 CartesianIndex{4, 3}
 CartesianIndex{5, 3}
 CartesianIndex{8, 3}
 CartesianIndex{9, 3}
 CartesianIndex{10, 3}
 CartesianIndex{5, 4}
 CartesianIndex{6, 4}
 CartesianIndex{7, 4}
 CartesianIndex{9, 4}
 CartesianIndex{2, 5}
 CartesianIndex{4, 5}
 CartesianIndex{5, 5}
 CartesianIndex{6, 5}
 CartesianIndex{10, 5}

```

1. Даны множества: $A = \{0, 3, 4, 9\}$, $B = \{1, 3, 4, 7\}$, $C = \{0, 1, 2, 4, 7, 8, 9\}$. Найти $A \cap B \cup A \cap C \cup B \cap C$.

In [33]:

```

A = Set{([0,3,4,9])}
B = Set{([1, 3, 4, 7])}
C = Set{([0, 1, 2, 4, 7, 8, 9])}
@show P = union(intersect(A, B), intersect(A, C), intersect(B, C))

```

```

P = union(intersect(A, B), intersect(A, C), intersect(B, C)) = Set{([0, 4, 7, 9, 3, 1])}

```

Out[33]:

```

Set{Int64} with 6 elements:
 0
 4
 7
 9
 3
 1

```

1. Приведите свои примеры с выполнением операций над множествами элементов разных типов.

In [42]:

In [42]:

```
A = Set([1, 3, 4, 5])
B = Set("abracadabra1")
@show union(A, B)
@show intersect(A, B)
@show setdiff(A, B)
@show issubset(A, union(A, B))
```

```
union(A, B) = Set{Any}[5, 4, '1', 'a', 'd', 'r', 'k', 3, 1, 'b']
intersect(A, B) = Set{Any}{}
setdiff(A, B) = Set{Any}[5, 4, 3, 1]
issubset(A, union(A, B)) = true
```

Out[42]:

true

In [40]:

```
@show union(A, B)
@show intersect(A, B)
@show setdiff(A, B)
@show issubset(A, union(A, B))
```

```
union(A, B) = Set{Any}[5, 4, '1', 'a', 'd', 'r', 'k', 3, 1, 'b']
intersect(A, B) = Set{Any}{}
setdiff(A, B) = Set{Any}[5, 4, 3, 1]
issubset(A, union(A, B)) = true
```

Out[40]:

true

Создайте разными способами:

3.1) массив (1, 2, 3, ..., -1,), выберите больше 20;

3.2) массив (, -1 ..., 2, 1), выберите больше 20;

3.3) массив (1, 2, 3, ..., -1, , -1, ..., 2, 1), выберите больше 20;

In [52]:

```
@show a1 = collect(1:21)
@show a2 = collect(21:-1:1)
@show array_3_3 = [1:21; 21:-1:-1:2]
```

```
a1 = collect(1:21) = [1, 2, 3, 4, 5, 6, 7, 8, 9, 10, 11, 12, 13, 14, 15, 16, 17, 18, 19, 20, 21]
a2 = collect(21:-1:1) = [21, 20, 19, 18, 17, 16, 15, 14, 13, 12, 11, 10, 9, 8, 7, 6, 5, 4, 3, 2, 1]
array_3_3 = [1:21; 21:-1:-1:2] = [1, 2, 3, 4, 5, 6, 7, 8, 9, 10, 11, 12, 13, 14, 15, 16, 17, 18, 19, 20, 21, 20, 19, 18, 17, 16, 15, 14, 13, 12, 11, 10, 9, 8, 7, 6, 5, 4, 3, 2]
```

Out[52]:

40-element Vector{Int64}:

```
1
2
3
4
5
6
7
8
9
10
11
12
13
⋮
13
12
11
10
9
8
7
6
```

"fn1"
"fn2"
"fn3"
"fn4"
"fn5"
"fn6"
"fn7"
"fn8"
"fn9"
"fn10"
"fn11"

```
"fn12"
"fn13"
:
"fn19"
"fn20"
"fn21"
"fn22"
"fn23"
"fn24"
"fn25"
"fn26"
"fn27"
"fn28"
"fn29"
"fn30"
```

векторы и целочисленного типа длины = 250 как случайные выборки из совокупности 0, 1, ..., 999; на его основе:

- сформируйте вектор $(x_2 - x_1, \dots, x_n - x_{n-1})$;
- сформируйте вектор $(x_1 + 2x_2 - x_3, x_2 + 2x_3 - x_4, \dots, x_{n-2} + 2x_{n-1} - x_n)$;
- сформируйте вектор $(\frac{\sin(x_1)}{\cos(x_2)}, \dots, \frac{\sin(x_{n-1})}{\cos(x_n)})$;
- вычислите $\sum_{i=1}^{n-1} \frac{e^{-x_{i+1}}}{x_i + 10}$

In [90]:

```
n = 250
x = rand(0:999, n)
y = rand(0:999, n)
v1 = [y[i]-x[i-1] for i in 2:n]
v2 = [x[i] + 2*x[i+1] - x[i+2] for i in 1:n-2]
v3 = [sin(y[i]) / cos(x[i+1]) for i in 1:n-1]
S = sum([exp(-x[i+1]) / (x[i] + 10) for i in 1:n-1])
```

Out[90]:

0.0025233553067527583

- выберите элементы вектора y , значения которых больше 600, и выведите на экран; определите индексы этих элементов;
- определите значения вектора x , соответствующие значениям вектора y , значения которых больше 600 (под соответствием понимается расположение на аналогичных индексных позициях);
- сформируйте вектор $(|x_1 - \bar{x}|^{0.5}, \dots, |x_n - \bar{x}|^{0.5})$, где \bar{x} обозначает среднее значение вектора x ;
- определите, сколько элементов вектора y отстоят от максимального значения не более, чем на 200;

In [101]:

```
@show y[y .> 600]
@show findall(y .> 600)
@show x[y .> 600]
v4 = [abs(x[i] - sum(x)/n)^0.5 for i in 1:n]
@show sum([abs(y[i] - maximum(y)) for i in 1:n] .<= 200)
```

```
y[y .> 600] = [984, 650, 869, 619, 714, 920, 750, 672, 805, 752, 696, 640, 926, 863, 787, 681, 776, 976, 949, 711, 852, 892, 707, 847, 622, 889, 739,
983, 923, 794, 874, 847, 689, 950, 855, 852, 948, 657, 850, 971, 670, 869, 604, 752, 702, 634, 669, 920, 913, 769, 999, 968, 736, 722, 858, 850, 804,
976, 850, 764, 684, 886, 889, 623, 717, 918, 758, 756, 787, 817, 747, 620, 962, 837, 841, 777, 765, 721, 829, 868, 763, 989, 877, 992, 888, 826, 946,
901, 640, 896, 702, 816, 736, 665, 860, 721, 904, 934, 960, 955, 739, 921, 945, 651, 997, 862]
findall(y .> 600) = [2, 3, 8, 9, 11, 12, 15, 18, 19, 22, 30, 33, 34, 35, 36, 42, 45, 46, 52, 53, 54, 56, 58, 61, 62, 63, 66, 67, 70, 73, 75, 77, 79, 82, 85, 88,
0, 100, 103, 107, 108, 109, 113, 114, 118, 119, 120, 121, 127, 128, 129, 130, 131, 133, 134, 140, 142, 143, 144, 145, 149, 151, 153, 155, 157, 161, 165, 166, 167, 171, 173, 174, 182, 183, 185, 189, 190, 191, 192, 193, 194, 195, 196, 201, 203, 204, 207, 208, 209, 212, 214, 215, 216, 217, 218, 220,
222, 226, 229, 230, 232, 237, 239, 240, 245, 248]
x[y .> 600] = [352, 348, 394, 102, 598, 649, 848, 368, 89, 914, 927, 589, 704, 29, 221, 16, 188, 2, 41, 931, 685, 93, 361, 36, 497, 563, 417, 594, 128,
344, 339, 182, 367, 865, 118, 97, 667, 369, 402, 383, 836, 927, 867, 451, 877, 823, 380, 481, 841, 767, 607, 733, 738, 482, 189, 50, 214, 116, 651, 412, 80, 115, 913, 180, 175, 932, 335, 812, 177, 503, 458, 525, 553, 760, 368, 943, 64, 561, 485, 737, 750, 22, 745, 814, 668, 336, 600, 943, 859, 85,
761, 816, 766, 747, 410, 435, 479, 277, 491, 635, 375, 19, 586, 294, 750, 463]
sum([abs(y[i] - maximum(y)) for i in 1:n] .<= 200) = 59
```

Out[101]:

- определите, сколько чётных и нечётных элементов вектора ;
- определите, сколько элементов вектора кратны 7;
- отсортируйте элементы вектора в порядке возрастания элементов вектора ;
- выведите элементы вектора , которые входят в десятку наибольших (top-10)?
- сформируйте вектор, содержащий только уникальные (неповторяющиеся) элементы вектора .

In [118]:

```
println(count(x -> x % 2 == 0, x), " четные")
println(count(x -> x % 2 == 1, x), " нечетные")
println(count(x -> x % 7 == 0, x), " делятся на 7")
@show x[sortperm(y)]
println("10 максимальных: ", sort(x, rev=true)[1:10])
unique_x = unique(x)
```

```
120 четные
130 нечетные
29 делятся на 7
x[sortperm(y)] = [564, 244, 889, 988, 88, 152, 491, 606, 933, 266, 21, 318, 668, 575, 815, 501, 769, 725, 221, 805, 610, 599, 861, 592, 718, 331, 233,
623, 286, 886, 871, 410, 671, 257, 596, 783, 747, 221, 86, 285, 277, 360, 280, 859, 360, 906, 79, 668, 397, 919, 522, 941, 711, 207, 645, 656, 410, 4
56, 577, 279, 524, 45, 968, 804, 90, 682, 66, 434, 481, 453, 377, 289, 258, 10, 365, 956, 359, 494, 745, 50, 976, 538, 526, 746, 963, 568, 374, 2, 392
, 670, 357, 755, 662, 204, 14, 423, 775, 790, 364, 681, 572, 639, 401, 276, 552, 400, 678, 865, 153, 49, 926, 557, 323, 859, 454, 307, 1, 106, 296, 77
3, 673, 750, 308, 41, 171, 624, 979, 899, 22, 142, 674, 363, 650, 197, 980, 928, 774, 558, 733, 637, 905, 434, 35, 659, 867, 102, 525, 497, 180, 823,
589, 859, 348, 294, 369, 747, 380, 836, 368, 16, 80, 367, 927, 877, 761, 361, 931, 598, 175, 561, 435, 482, 738, 766, 417, 375, 458, 848, 914, 451, 8
12, 335, 750, 412, 64, 767, 188, 943, 221, 177, 344, 214, 89, 816, 503, 336, 485, 760, 368, 36, 182, 402, 50, 651, 685, 97, 118, 189, 410, 463, 29, 73
7, 394, 927, 339, 745, 115, 668, 563, 913, 93, 85, 943, 479, 841, 932, 649, 481, 19, 128, 704, 277, 586, 600, 667, 41, 865, 635, 491, 553, 733, 383, 2
, 116, 594, 352, 22, 814, 750, 607]
10 максимальных: [988, 980, 979, 976, 968, 963, 956, 943, 943, 941]
```

Out[118]:

224-element Vector{Int64}:

```
221
352
348
364
88
871
552
394
102
360
598
649
90
:
933
19
568
586
294
577
171
233
899
463
673
592
```

Создайте массив squares, в котором будут храниться квадраты всех целых чисел от 1 до 100.

In [119]:

```
squares = [x^2 for x in 1:100]
```

Out[119]:

100-element Vector{Int64}:

```
1
4
9
16
25
```


25
36
49
64
81
100
121
144
169
⋮
7921
8100
8281
8464
8649
8836
9025
9216
9409
9604
9801
10000

Подключите пакет Primes (функции для вычисления простых чисел). Сгенерируйте массив myprimes, в котором будут храниться первые 168 простых чисел. Определите 89-е наименьшее простое число. Получите срез массива с 89-го до 99-го элемента включительно, содержащий наименьшие простые числа.

In [121]:

```
import Pkg; Pkg.add("Primes")

Warning: could not download https://pkg.julialang.org/registries
exception = InterruptException:
@ Pkg.Registry C:\workdir\usr\share\julia\stdlib\v1.8\Pkg\src\Registry\Registry.jl:68
Updating registry at `D:\julia\depot\registries\General.toml`
Resolving package versions...
Installed IntegerMathUtils — v0.1.2
Installed Primes — v0.5.4
Updating `D:\julia\depot\environments\v1.8\Project.toml`
[27ebfcd6] + Primes v0.5.4
Updating `D:\julia\depot\environments\v1.8\Manifest.toml`
[18e54dd8] + IntegerMathUtils v0.1.2
[27ebfcd6] + Primes v0.5.4
Precompiling project...
✓ IntegerMathUtils
✓ Primes
2 dependencies successfully precompiled in 5 seconds. 272 already precompiled. 2 skipped during auto due to previous errors.
```

In [122]:

```
using Primes
```

In [129]:

```
myprimes = primes(1000)[1:168]
println(myprimes[89])
println(myprimes[89:99])
```

461
[461, 463, 467, 479, 487, 491, 499, 503, 509, 521, 523]

Вычислите следующие выражения:

6.1) $\sum_{i=10}^{100} (\beta + 4\beta^2)$

6.2) $\sum_{i=1}^{25} \left(\frac{2^i}{7} + \frac{3^i}{\beta} \right)$

6.2) $1 + \frac{2}{3} + \frac{2 \cdot 4}{3 \cdot 5} + \frac{2 \cdot 4 \dots 38}{3 \cdot 5 \dots 39}$

In [152]:

```
println(sum((1/3 - 1/4) for i in 10:100))
```

```
println(sum([i^3+4*i^2 for i in 10:100]))
println(sum([2^i/i + 3^i/i^2 for i in 1:25]))
print(1 + sum([prod(BigInt, 2:2:2*i) / prod(BigInt, 3:2:2*i+1) for i in 1:19]))
```

```
26852735
2.1291704368143802e9
6.976346137897620220974672781027668100260836150955607356439198582642989471833141
```

In [154]:

```
[prod(BigInt, 2:2:2*i) // prod(BigInt, 3:2:2*i+1) for i in 1:19]
```

Out[154]:

```
19-element Vector{Rational{BigInt}}:
 2//3
 8//15
16//35
128//315
256//693
1024//3003
2048//6435
32768//109395
65536//230945
262144//969969
524288//2028117
4194304//16900975
8388608//35102025
33554432//145422675
67108864//300540195
2147483648//9917826435
4294967296//20419054425
17179869184//83945001525
34359738368//172308161025
```

In []: