Повторение примеров

In [1]:
```
typeof(3), typeof(3.5), typeof(3/3.55), typeof(sqrt(3+4im)), typeof(pi)
```

Out[1]:
```
(Int64, Float64, Float64, ComplexF64, Irrational{:π})
```

In [2]:
```
1.0/0.0, 1.0/(-0.0), 0.0/0.0
```

Out[2]:
```
(Inf, -Inf, NaN)
```

In [3]:
```
for T in [Int8,Int16,Int32,Int64,Int128,UInt8,UInt16,UInt32,UInt64,UInt128]
println("$(lpad(T,7)): [$(typemin(T)),$(typemax(T))]")
end
```

```
  Int8: [-128,127]
 Int16: [-32768,32767]
 Int32: [-2147483648,2147483647]
 Int64: [-9223372036854775808,9223372036854775807]
Int128: [-170141183460469231731687303715884105728,170141183460469231731687303715884105727]
 UInt8: [0,255]
UInt16: [0,65535]
UInt32: [0,4294967295]
UInt64: [0,18446744073709551615]
UInt128: [0,340282366920938463463374607431768211455]
```

In [4]:
```
Int64(2.0), Char(2), typeof(Char(2))
```

Out[4]:
```
(2, '\x02', Char)
```

In [5]:
```
convert(Int64, 2.0), convert(Char, 2)
```

Out[5]:
```
(2, '\x02')
```

In [6]:
```
typeof(promote(Int8(1), Float16(4.5), Float32(4.1)))
```

Out[6]:
```
Tuple{Float32, Float32, Float32}
```

In [7]:
```
function f(x)
    x^2
end
```

Out[7]:
```
f (generic function with 1 method)
```

In [8]:
```
f(4)
```

Out[8]:
```
16
```

In [9]:
```
g(x) = x^2
```

Out[9]:
```
g (generic function with 1 method)
```

In [10]:
```
g(8)
```

Out[10]:
```
64
```

In [11]:
```
a = [4 7 6]
b = [1, 2, 3]
a[2], b[2]
```

Out[11]:
```
(7, 2)
```

In [12]:
```
a = 1; b = 2; c = 3; d = 4
Am = [a b; c d]
```

Out[12]:
```
2×2 Matrix{Int64}:
 1  2
 3  4
```

In [13]:
```
Am[1,1], Am[1,2], Am[2,1], Am[2,2]
```

Out[13]:
```
(1, 2, 3, 4)
```

In [14]:
```
aa = [1 2]
AA = [1 2; 3 4]
aa*AA*aa'
```

Out[14]:
```
1×1 Matrix{Int64}:
 27
```

In [15]:
```
aa, AA, aa'
```

Out[15]:
```
([1 2], [1 2; 3 4], [1; 2;;])
```

Самостоятельная работа

In [16]:
```
write("my_file.txt", "Удивительное рядом!\nДостаточно просто протянуть руку!")
read("my_file.txt", String)
```

Out[16]:
```
"Удивительное рядом!\nДостаточно просто протянуть руку!"
```

In [17]:
```
readline("my_file.txt")
```

Out[17]:
```
"Удивительное рядом!"
```

In [18]:
```
readlines("my_file.txt")
```

Out[18]:
```
2-element Vector{String}:
 "Удивительное рядом!"
 "Достаточно просто протянуть руку!"
```

In [19]:
```
print("Julia is a programming language")
print("Julia is a programming language")
```

```
Julia is a programming languageJulia is a programming language
```

In [20]:
```
println("Julia is a programming language")
println("Julia is a programming language")
```

```
Julia is a programming language
Julia is a programming language
```

In [28]:
```
struct November
   n::Int
end

Base.show(io::IO, ::MIME"text/plain", d::November) = print(io, d.n, " ноября")
November(11)
```

Out[28]:
```
11 ноября
```

In [22]:
```
open("delim_file.txt", "w") do f
          write(f, "1,2\n3,4\n5,6\n7,8")
end
```

Out[22]:
```
15
```

In [23]:
```
using DelimitedFiles
readdlm("delim_file.txt", ',', Float64)
```

Out[23]:
```
4×2 Matrix{Float64}:
 1.0  2.0
 3.0  4.0
 5.0  6.0
 7.0  8.0
```

In [29]:
```
parse(Int, "afc", base = 16), parse(Float64, "1.2e-3")
```

Out[29]:
```
(2812, 0.0012)
```

In [36]:
```
4+5, [1 2] + [2 3], [1, 2].*3, [1 2] * [1, 2], [10 5]./5, mod(7,3), div(7,3), 10^6, [1, 2] * [1 2]
```

Out[36]:
```
(9, [3 5], [3, 6], [5], [2.0 1.0], 1, 2, 1000000, [1 2; 2 4])
```

In [26]:
```
[1 3]', [2, 4]', [1 2; 3 4]'
```

Out[26]:
```
([1; 3;;], [2 4], [1 3; 2 4])
```

In [ ]: