

# Введение в программирование

## Лекция 2

Лопатин Александр

2015

- 1 Языки программирования
- 2 «Hello world» на всех языках
- 3 Переменные и константы
- 4 Действия (операторы)
- 5 Структурное программирование

# Язык программирования

Знаковая система для написания компьютерных программ

Текст, написанный на таком языке называют текст программы  
или исходный код (source code) или просто код

## Трансляторы языков программирования

Программы, которые понимают исходный код

Бывают двух типов:

- ▶ Компилятор — превращает текст программы в двоичный код
- ▶ Интерпретатор — читает текст программы и выполняет написанное

## Двоичный код

- ▶ **машинный код** (machine/native code) — код, который исполняет аппаратный исполнитель (например микропроцессор)
- ▶ **байт-код виртуальной машины** — код, который исполняет программный исполнитель (например «виртуальная машина .NET» или «виртуальная машина Java (JVM)»)

Понятие «виртуальная машина» — многозначно. Об этом разжевано здесь:

<http://habrahabr.ru/company/intel/blog/254793/>

С ростом популярности JIT-компиляции разделение трансляторов на компиляторы и интерпретаторы не столь актуально

Большинство актуальных интерпретаторов стало, грубо говоря, компиляторами в машинный код)

<https://ru.wikipedia.org/wiki/JIT>

Лучше разделять сами языки (и то как их применяют), а не трансляторы:

- ▶ **Компилируемые** (Assembler, C++, C#, Java, ...)
- ▶ **Скриптовые** (Python, Ruby, JavaScript/ECMAScript, bash/shell, cmd/bat, PowerShell, ...)

Слово «скриптовый» всё же не имеет четкого значения и для многих является синонимом к интерпретируемым языкам

<http://programmers.stackexchange.com/a/46233>

## Компилируемые языки

Исходный код компилируют в двоичный код и (обычно) распространяют в скомпилированном виде (.exe, .dll, .so и т.д.)

Часто используется для:

- ▶ **прикладного программирования** (от текстового редактора до веб-браузера или более сложной системы)
- ▶ **системного программирования** (драйвера устройств и т.д.)

На них часто решают задачи, требующие высокой производительности



## Скриптовые языки (языки сценариев)

Для многих — синоним к «Интерпретируемым языкам»

Программы обычно распространяют в виде исходного кода  
(.py, .js, .bat и т.д.)

Часто используется для:

- ▶ **прототипов** прикладных программ
- ▶ **сценариев** для автоматизации задач, вроде сборки/тестирование/deploy билда
- ▶ написание пользовательских **макросов** для игры или электронной таблицы
- ▶ **плагинов/расширений** (для браузера, скажем)
- ▶ клиентский или серверный **код для веб-сайта**

Популярно там, где хочется быстро увидеть результат

А еще языки классифицируют по тому, на какие  
**парадигмы программирования** сделан основной акцент...

## Wait, wait...

Зачем об этом всё́м знать? — Для осознания того

- ▶ что языки можно очень по-разному классифицировать
- ▶ одни классы эффективно решают одни типы задач, другие классы — другие типы задач
- ▶ «универсального» языка не существует
- ▶ чем более узкоспециализирован язык (SQL для БД, G-Codes для станков, Verilog для спецификации устройств, GLSL для графики...) или комбинация языка и библиотеки/фреймворка (Ruby+RoR или JS+node.js для веб, ...) тем быстрее их можно изучить

Другими словами — чтобы появилось представление о способе выбора следующего языка для изучения


<https://www.youtube.com/watch?v=LR8fQiskYII>

<https://www.youtube.com/watch?v=NvWTnIoQZj4>

## На википедии

Можно увидеть разницу между языками, если обращать внимание на Paradigm и Typing discipline

Python	
	
<b>Paradigm</b>	multi-paradigm: object-oriented, imperative, functional, procedural, reflective
<b>Designed by</b>	Guido van Rossum
<b>Developer</b>	Python Software Foundation
<b>First appeared</b>	1991; 24 years ago
<b>Stable release</b>	3.4.3 / 25 February 2015 <sup>[1]</sup> 2.7.10 / 23 May 2015 <sup>[2]</sup>
<b>Preview release</b>	3.5.0b3 / 5 July 2015 <sup>[3]</sup>
<b>Typing discipline</b>	duck, dynamic, strong, gradual (as of Python 3.5) <sup>[4]</sup>

JavaScript	
	
<b>Paradigm</b>	Multi-paradigm: scripting, object-oriented (prototype-based), imperative, functional <sup>[1]</sup>
<b>Designed by</b>	Brendan Eich
<b>Developer</b>	Netscape Communications Corporation, Mozilla Foundation, Ecma International
<b>First appeared</b>	1995; 20 years ago
<b>Stable release</b>	ECMAScript 6 <sup>[2]</sup> / June 17, 2015; 24 days ago
<b>Typing discipline</b>	dynamic, duck

Java	
	
<b>Paradigm</b>	multi-paradigm: object-oriented (class-based), structured, imperative, functional, generic, reflective, concurrent
<b>Designed by</b>	James Gosling and Sun Microsystems
<b>Developer</b>	Oracle Corporation
<b>First appeared</b>	1995; 20 years ago <sup>[1]</sup>
<b>Stable release</b>	Java Standard Edition 8 Update 45 (1.8.0_45) <sup>[2]</sup> / April 14, 2015; 2 months ago <sup>[3]</sup>
<b>Preview release</b>	Java Standard Edition 9 Early Access b46 (1.9.0-ea-b46) / January 20, 2015; 5 months ago
<b>Typing discipline</b>	Static, strong, safe, nominative, manifest

## Привет, Python 2!

```
print "Hello World!"
```

Попробовать —

[http://tutorialspoint.com/execute\\_python\\_online.php](http://tutorialspoint.com/execute_python_online.php)

## Привет, JavaScript (ECMAScript 6)!

```
console.log("Hello World!")
```

Попробовать — <http://www.es6fiddle.net/>

## Привет, Java 8!

```
public class HelloWorld {  
    public static void main(String[] args) {  
        System.out.println("Hello World!");  
    }  
}
```

Попробовать — <http://www.compilejava.net/>

## Данные в языках — это переменные и константы

**Переменную** можно изменять (время, координаты, ...)

Пример на Java:

```
int x = 2;  
System.out.println("x == " + x);           // x == 2  
x = x + 1;  
System.out.println("x == " + x);           // x == 3
```

**Константу** нельзя изменять (число  $\pi$ , скорость света ...)

Пример на Java:

```
final double pi = 3.14;
```



## Данные могут быть разных типов

Примитивные (простые) типы:

- ▶ **целое** число (int, long, ...): -123, 12345678910L
- ▶ **дробное** число (float, double): 0.123f, 0.123
- ▶ **символ** (char): 'z'

Составные типы:

- ▶ **массивы** (array): [1, 2, 3, 4]
- ▶ **объекты** (string, list, set, dict/map, ...): "Hello",  
{ "hello": "привет" }

## Присваивание (Python)

```
x = 1
y = 2
print x          # 1
x = y
print x          # 2
x = y + 1
print x          # 3
```

Синтаксис: переменная = значение

Читается «переменной присвоить значение»

Или же: переменная = выражение

Читается «переменной присвоить результат вычисления  
выражения»

## Арифметические операторы (Python)

```
x = 2 + 2
print x                # 4
x = x - 1
print x                # 3
x = x * x
print x                # 9
print x % 2            # 1 (остаток от деления)
print x / 2            # 4 (целочисленное деление)
print x / 2.0          # 4.5 (деление целого на дробное)
print float(x) / 2.0   # 4.5
```

В предпоследнем: целое (int) было неявно преобразовано в дробное (float)

## Логические операторы (Python)

```
x = True
y = False
print x and y      # False
print x or y       # True
print not x        # False (может самое, что и x == False)
```

```
a = 2
b = 3
print a < b        # True
print a > b        # False
print a <= b, a >= b # True False
print a == b       # False (читается "a равняется b")
print a == (b - 1) # True
```

Не путать **равенство** (логический оператор) с **присвоением** (действие, которое изменяет значение переменной)

## Логические операторы (JavaScript)

```
var x = true
var y = false
console.log(x && y)    // false
console.log(x || y)    // true
console.log(!x)        // false
```

```
var a = 2
var b = 3
// операторы сравнения везде одинаковые, кроме равенства
console.log(a === b)   // false
```

Порядок вычислений зависит от приоритетов операторов

Например в выражении

$$y = a + b * c$$

сначала будет выполнено умножение, потом — сложение.

На порядок вычислений можно повлиять, расставив скобки

$$y = (a + b) * c$$

Существуют операторы, которые имеют разные приоритеты,  
в зависимости от языка

В случае неуверенности в порядке вычислений — нужно  
проставлять скобки

А в случае со слишком длинным выражением — лучше распилить его на части

Например вместо

```
pageHeight = headerHeight + itemHeight * newsNumber +  
              footerHeight
```

написать что-то вроде

```
newsHeight = itemHeight * newsNumber  
pageHeight = headerHeight + newsHeight + footerHeight
```



## Ввод (Python)

```
line = raw_input("input a number: ") # ввод строки
number = int(line)                   # преобр. в целое
line = raw_input("input something: ")# снова ввод строки
floatNumber = float(line)            # преобр.
                                     # в дробное
```

## Массивы (Python)

```
x = [1, 2, 55, -123]
i = 2
print x[i]                # 55
x[i] = 777
print x                    # [1, 2, 777, -55]
n = len(x)
print n                    # 4
```

$n$  — это **размер** или **длина** массива

$i$  — это **индекс** массива

Индексация (обычно) начинается с нуля

## Массивы (Java)

```
int x[] = {1, 2, 55, -123};  
int y[] = new int[x.length];  
System.out.println("length is " + x.length);  
// length is 4
```

Подробнее —

[http://www.tutorialspoint.com/java/java\\_arrays.htm](http://www.tutorialspoint.com/java/java_arrays.htm)

## Массивы (JavaScript)

```
var x = [1, 2, 55, -123]  
// индексировать, изменять и получать размер  
// - как в Java
```

Подробнее — [http://www.w3schools.com/js/js\\_arrays.asp](http://www.w3schools.com/js/js_arrays.asp)

## Массивы могут быть вложены

Массивы с двумя уровнями вложенности называют двумерными или «массив размерности два»  
(не путать с размерность с размером)

Пример (на Python) массива 2x4 (размерности 2, размера 4; или с 2-мя строками и 4-мя столбцами):

```
x = [[1, 2, 55, -123], [4, 5, 6, 7]]
```

```
x[1][3] = 4444
```

```
print x
```

```
[[1, 2, 55, -123], [4, 5, 6, 4444]]
```

## Ввод (Java)

```
import java.util.Scanner;           // импорт библиотеки
...
Scanner in = new Scanner(System.in); // создание объекта
String line = in.nextLine();         // получение строки
int number = in.nextInt();           // получение целого
float floatNumber = in.nextFloat();  // получение дробного
```

## Ввод (JavaScript)

Там для этого можно использовать HTML-форму

Пока не будем это использовать

## Практика

Поиграться с описанным выше на всех языках



## Условия (JavaScript)

```
var a = true  
var b = false
```

*// с одной веткой*

```
if (a && b) {  
    console.log("both are true")  
}
```

*// с двумя ветками*

```
if (a && b) {  
    console.log("both are true")  
} else {  
    console.log("one of them")  
}
```

## Вложенные условия (JavaScript)

```
var a = true  
var b = false
```

```
if (a && b) {  
    console.log("both are true")  
} else {  
    if (!a) {  
        console.log("a is false")  
    } else {  
        console.log("b is false")  
    }  
}
```

*// более читаемый вариант*

```
if (a && b) {  
    console.log("both are true")  
} else if (!a) {  
    console.log("a is false")  
} else {  
    console.log("b is false")  
}
```

Hint: надо всегда выделять ветки условий в фигурные скобки в языках JS и Java

## Вложенные условия (Python)

```
a = True
b = False

if a and b:
    print "both are true"
elif not a:
    print "a is false"
else:
    print "b is false"
```

## Цикл с предусловием (Python)

```
i = 0
while i < 10:
    print i
    i = i + 1
```

## Цикл с постусловием (Java)

```
int i = 0;  
do {  
    System.out.println("i = " + i)  
    i = i + 1;  
} while (i < 10);
```

## Эмуляция цикла с постусловием (Python)

Настоящего цикла с постусловием в Python нет

```
i = 0
conditionMet = False
while not condMet:
    print "i =", i
    i = i + 1
    if i < 10:
        condMet = True
```

## Цикл со счетчиком (Java)

```
for (int i = 0; i < 10; i = i + 1) {  
    System.out.println("i = " + i)  
}
```

## Цикл со «счетчиком» (Python)

```
for i in range(0, 10, 1):  
    print "i = " + str(i)
```



## Циклы могут быть вложены (Python)

```
a = [[4, 5, 6], [7, 8, 9]]
n = len(a)
m = len(a[0])

for i in range(0, 2, 1):
    j = 0
    while j < m:
        print "i =", i, " j =", j
        print a[i][j]
        j = j + 1
```

## Практика

Поиграться с описанным выше на всех языках

Написать примитивный калькулятор

Написать алгоритм поиска наибольшего элемента в одномерном массиве

## Домашка

На любом из трёх языков программирования: Python, JavaScript или Java (по +1 баллу за реализацию на других языках, из перечисленных):

1. Реализовать задачу из предыдущей лекции (решение квадратного уравнения)
2. Дан одномерный массив из  $n$  целочисленных элементов (реализовывать ввод массива не нужно). Инвертировать порядок элементов в этом массиве
3. Бонусная задача: дан двумерный массив целочисленных элементов (реализовывать ввод массива не нужно) из  $n$  строк и  $m$  столбцов. Вывести номер строки и столбца наименьшего элемента
  - ▶ подсказка: стоит начать с упрощенной версии программы: вывод наименьшего элемента одномерного массива, усложнить до вывода индекса наименьшего элемента одномерного массива и только потом пытаться реализовать тоже самое для двумерного массива