Введение в языки программирования. Базовые типы данных

Цели: общее знакомство с языками программирования.
Изучить базовые типы данных.

План урока:

- для чего нужен язык программирования?
- высокоуровневые и низкоуровневые ЯП
- компилируемые, интерпретируемые ЯП
- типы данных
- работа со строками
- типизация статическая и динамическая
- операторы

Процессоры и программы

Процессор компьютера — это сложная интегральная микросхема. Все команды и данные он получает в виде электрических сигналов.

Команды, поступающие в процессор, являются электрическими сигналами, которые представляют из себя совокупность нулей и единиц, то есть числа. Разным командам соответствуют разные числа.

Программа, с которой работает процессор, — это последовательность чисел, называемая машинным кодом.

Языки программирования

Самому написать программу в машинном коде весьма сложно, причем эта сложность резко возрастает с увеличением размера программы и трудоемкости решения нужной задачи.

Условно можно считать, что машинный код приемлем, если размер программы не превышает нескольких десятков байтов и нет потребности в операциях ввода/вывода данных. Поэтому сегодня практически все программы создаются с помощью языков программирования.

Языки программирования

Языки программирования — это искусственные языки, отличающиеся от естественных ограниченным числом «слов», значение которых понятно транслятору, и очень строгими правилами записи команд (операторов). Совокупность подобных требований образует синтаксис языка программирования, а смысл каждой команды и других конструкций языка — его семантику.

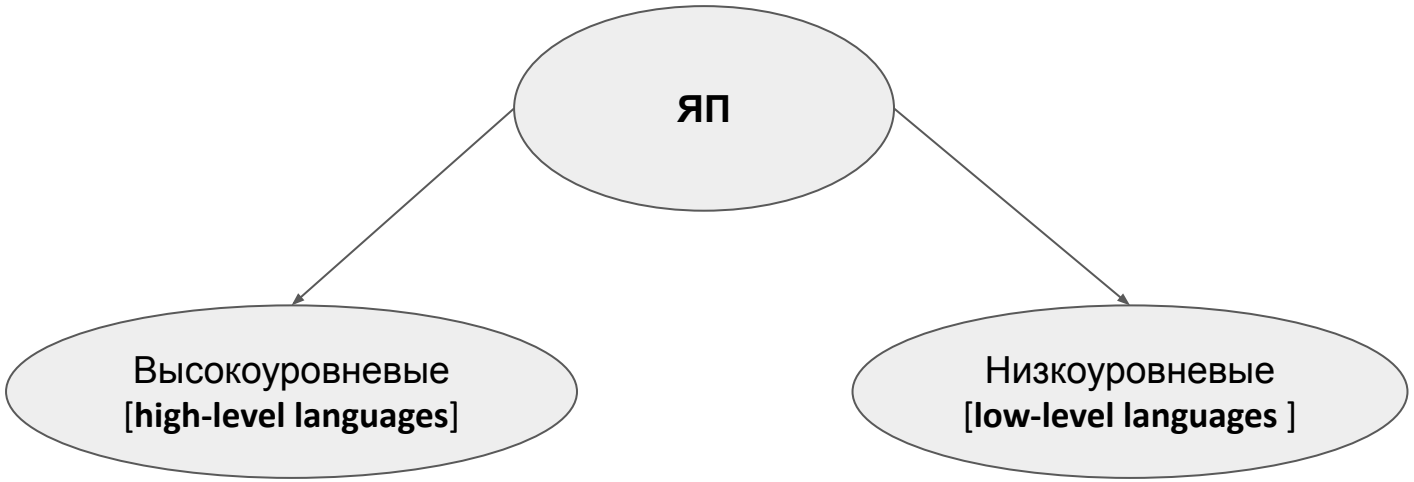
Нарушение формы записи программы приводит к тому, что транслятор не может понять назначение оператора и выдает сообщение о синтаксической ошибке, а правильно написанное, но не отвечающее алгоритму использование команд языка приводит к семантическим ошибкам.

Процесс устранения ошибок называется отладкой.

Языки программирования

Языки программирования бывают **высокоуровневыми** и **низкоуровневыми**.

Низкоуровневые: Assembler, CIL,
Высокоуровневые: любые объектно-ориентированные или поддерживающие сложные типы данных языки.



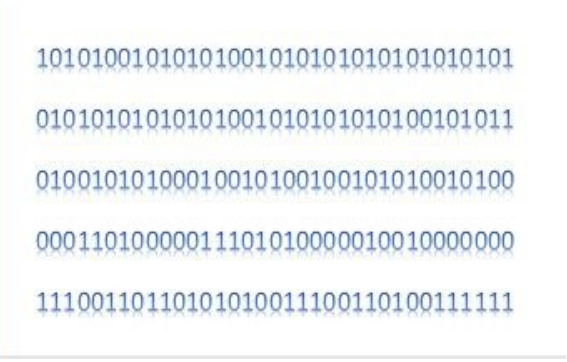
Дружественный к программисту язык, который обеспечивает высокий уровень абстракции от оборудования

python:
print("This line will be printed.")

Язык, **дружественный к компьютеру** и не обеспечивающий абстракции от оборудования или без него.

машинный код: «10110000 01100001» язык ассемблера может упростить это как **«MOV AL, 61h».**

Машинный код

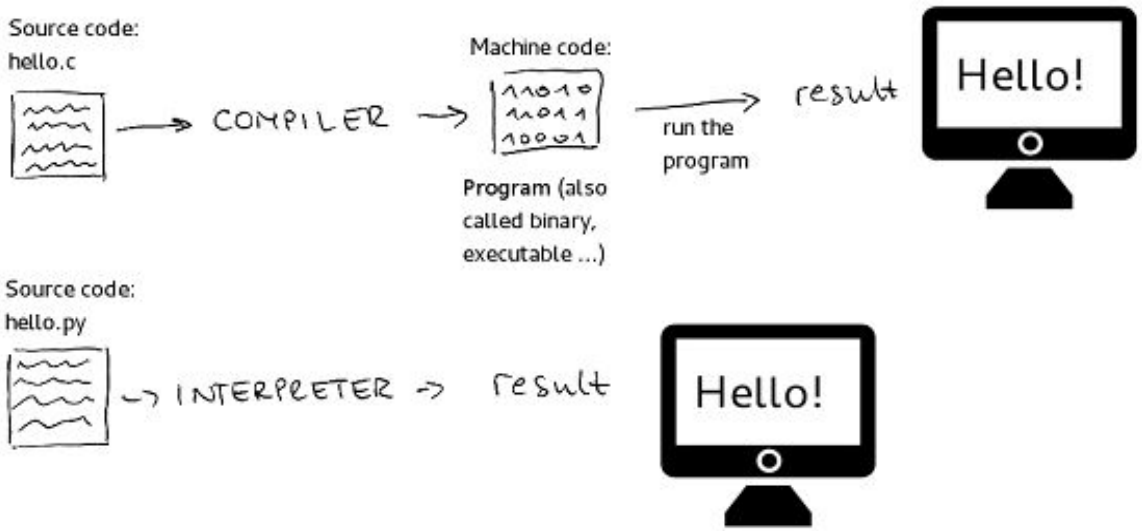


Языки программирования высокого и низкого уровней

Компиляторы и интерпретаторы

С помощью языка программирования создается не готовая программа, а только ее текст, описывающий ранее разработанный алгоритм.

Чтобы получить работающую программу, надо этот текст либо автоматически перевести в машинный код (с помощью компилятора) и затем использовать отдельно от исходного текста, либо сразу выполнять команды языка, указанные в тексте программы (с помощью интерпретатора).



Компиляторы и интерпретаторы

Компилируемые языки:

- Программа конвертируется непосредственно в машинный код
- Высокая скорость исполнения программ
- Большие возможности управления аппаратным обеспечением
- Дополнительные операции (компиляция) перед запуском
- Платформозависимость

C, C++, Erlang, Haskell, Rust, Go и др.

Интерпретируемые языки:

- Программу построчно исполняет интерпретатор
- Низкая скорость исполнения программ
- Удобство отладки программ
- Меньший размер исполняемых файлов
- Платформонезависимость

PHP, JavaScript, Perl, Ruby и др.

Байткод

Байткод-языки — это такие языки, которые используют для исполнения кода как компиляцию, так и интерпретацию. Java (Oracle), Python и .NET (Microsoft) — это типичные примеры байткод-языков.

- В байткод-языке сперва происходит компиляция программы из человекочитаемого языка в байткод.
- Затем байткод передаётся в виртуальную машину (Java Virtual Machine (Java), Common Language Runtime (.NET)), которая затем интерпретирует код также, как и обычный интерпретатор.

Байткод — это набор инструкций, созданный для эффективного исполнения интерпретатором и состоящий из компактных числовых кодов, констант и ссылок на память.

Байткод

При компиляции кода в байткод происходит задержка, но дальнейшая скорость исполнения значительно возрастает в силу оптимизации байткода.

Байткод-языки:

- Программа сначала компилируется, затем полученный набор инструкций интерпретируется
- Достаточно высокая скорость исполнения программ
- Платформонезависимость
- Меньший размер исполняемых файлов

Java, Python, C#, Fortran #, Basic .NET, C++ (CLR) и др.

Java

Java – это строго типизированный объектно-ориентированный язык программирования, который в свое время разработала компания Sun Microsystems.

Изначально язык назывался Oak («Дуб»), но это название оказалось уже зарегистрированным другой компанией, поэтому назвали Java в честь кофе, который в свою очередь был связан с островом Ява.

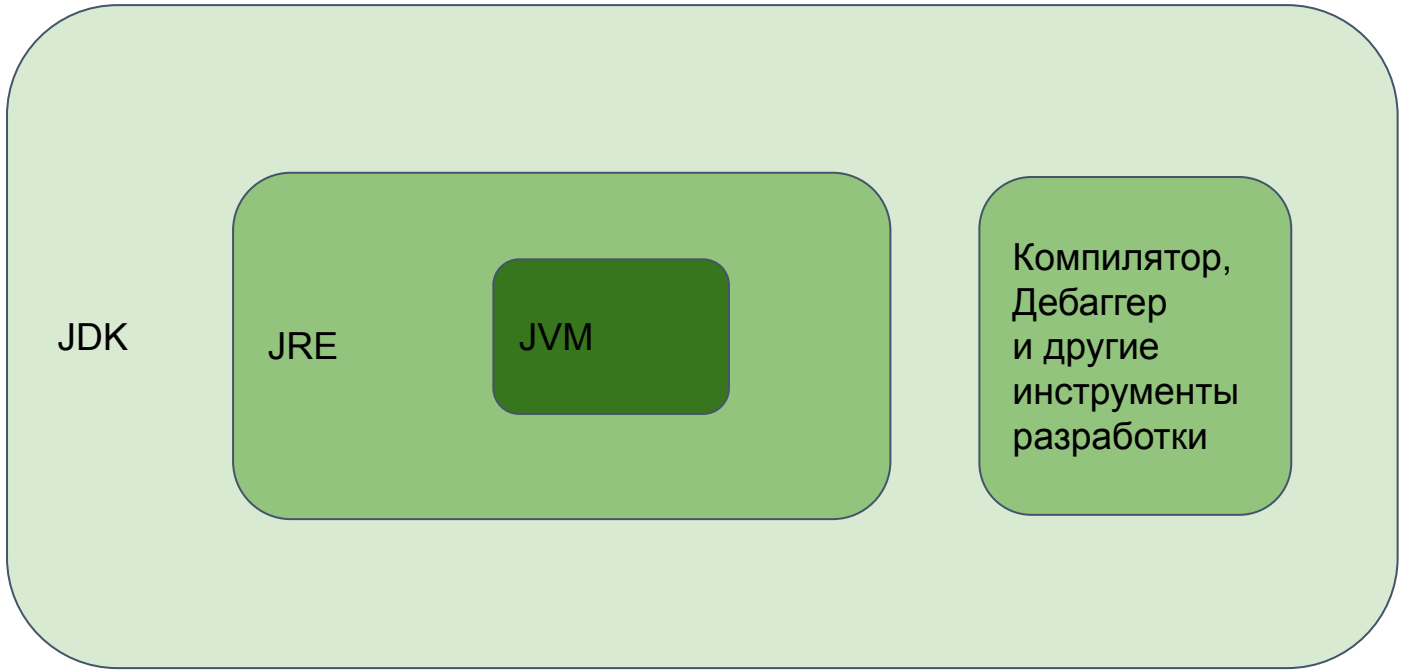
Из чего состоит Java

JVM (Java Virtual Machine) - Виртуальная машина отвечает за само выполнение кода. Она работает с байткодом (тем, что находится внутри файлов с расширением .class).

JRE (Java Runtime Environment) - окружение, необходимое для запуска Java-программ. Включает в себя стандартную библиотеку. В нее входят, как базовые пакеты lang, util, так и пакеты для работы с различными форматами, базами данных, пользовательским интерфейсом. JVM тоже часть JRE.

JDK (Java Development Kit) - набор программ для разработки. Именно его мы (или редактор) устанавливаем к себе на компьютер, чтобы заниматься разработкой на Java. Он включает в себя JRE, загрузчик кода java, компилятор javac, архиватор jar, генератор документации javadoc и другие утилиты, нужные во время разработки.

Из чего состоит Java



Создание переменных

Переменная — именованная область памяти, адрес которой можно использовать для осуществления доступа к данным. Данные, находящиеся в переменной, называются значением этой переменной.

Переменная – это коробка, в которой мы храним значение.
У каждой коробки есть имя

Переменная – variable;

Объявляем переменную
var balance = 100;

Базовые типы данных

Тип данных:

1. Рассказывает, какие значения могут быть
    a. **boolean**: хранит значение true или false
    b. **byte**: хранит целое число от -128 до 127 и занимает 1 байт
    c. **short**: хранит целое число от -32768 до 32767 и занимает 2 байта
    d. **int**: хранит целое число от -2147483648 до 2147483647 и занимает 4 байта
    e. **long**: хранит целое число от –9 223 372 036 854 775 808 до 9 223 372 036 854 775 807 и занимает 8 байт
    f. **double**: хранит число с плавающей точкой от ±4.9*10-324 до ±1.8*10308 и занимает 8 байт. В качестве разделителя целой и дробной части в дробных литералах используется точка.
    g. **float**: хранит число с плавающей точкой от -3.4*1038 до 3.4*1038 и занимает 4 байта
    h. **char**: хранит одиночный символ в кодировке UTF-16 и занимает 2 байта, поэтому диапазон хранимых значений от 0 до 65536
    i. **String**: строковые переменные представляют объект, который в отличие от char или int не является базовым типом.
2. Рассказывает, какие действия с данными можно выполнятЬ

Базовые типы данных

Тип	Возможные значения	Операции
boolean	true|false	Булева алгебра
byte	-128 — 127	Арифметика и сравнение чисел
short	-32768 — 32767	
int	-2147483648 — 2147483647	
long	–9 223 372 036 854 775 808 — 9 223 372 036 854 775 807	
String	все, что внутри “” ● “Москва” ● “Нижний Новгород” ● “5” ● “true” ● “ “ (пробел) ● “” (пустая строка)	Ох..

Переменные

Статическая типизация — когда знаем тип переменной до компиляции.

Примеры статически типизированных языков — Ada, C++, C#, D, Java, ML, Pascal, Nim.

Динамическая типизация — тип данных определяется в момент выполнения программы. Потенциально, могут быть ошибки, связанные с неправильным использованием типа.

Примеры языков с динамической типизацией — Smalltalk, Python, Objective-C, Ruby, PHP, Perl, JavaScript, Lisp, xBase, Erlang, Visual Basic.

Java:

```
int x; //объявление переменной
x = 10; //присваивание значения переменной
x = "Hi!"; //замена значени переменной
```

```
error: incompatible types: String
cannot be converted to int
```

JavaScript:

```
let x = 10; //объявление переменной и присваивание значения
x = "123"; //замена значения
```

Вывод информации на консоль

Управляющие литералы

\n - перевод строки

\t - табуляция

' – одинарная кавычка

” – двойная кавычка

```
System.out.println("\'Одинарные кавычки\'");
System.out.println("\"Двойные кавычки\"");
System.out.println("\\ Обратная косая черта");
System.out.println("Следующая строка\n");
System.out.println("\tТабулятор");

System.out.print("Hello world! \n");
System.out.println("Hello world!");

int x=5;
int y=6;
System.out.println("x=" + x + "; y=" + y);
System.out.printf("x=%d; y=%d \n", x, y);
```

Форматированный вывод

%d - для вывода целых чисел

%f2 - для вывода вещественных чисел с заданной точностью

%e - для вывода чисел в экспоненциальной форме, 1.3e+01

%c - для вывода одиночного символа (в Java)

%s - для вывода строковых значений

Базовые операторы

+ (операция сложения)

- (операция вычитания)

***** (умножение)

/ (деление)

% (получение остатка от деления)

++ (инкремент, постфиксный и инфиксный)

-- (декремент, постфиксный и инфиксный)

= (приравнивает переменной определенное значение)

+= (сложение с последующим присвоением результата)

-= (вычитание с последующим присвоением результата)

***=** (умножение с последующим присвоением результата)

/= (деление с последующим присвоением результата)

%= (получение остатка от деления с последующим присвоением результата)

Математические операции

Для выполнения стандартных математических операций используется библиотека **Math**.

К примеру, с её помощью можно получить число Пи, экспоненту, получить значение тригонометрической функции для какого-либо угла, посчитать квадратный корень(sqrt) и степень от числа(pow), и получить абсолютную величину числа(abs).

```
double pi = Math.PI;
System.out.println(pi); //3.141592653589793

double e = Math.E;
System.out.println(e); //2.718281828459045

double sin0 = Math.sin(0);
System.out.println(sin0); //0.0

double cos0 = Math.cos(0);
System.out.println(cos0); //1.0
```

```
double tan0 = Math.tan(0);
System.out.println(tan0); //0.0

double sqrt = Math.sqrt(4);
System.out.println(sqrt); //2.0

double abs = Math.abs(-10);
System.out.println(abs); //10.0

double pow = Math.pow(2, 3);
System.out.println(pow); //8.0
```

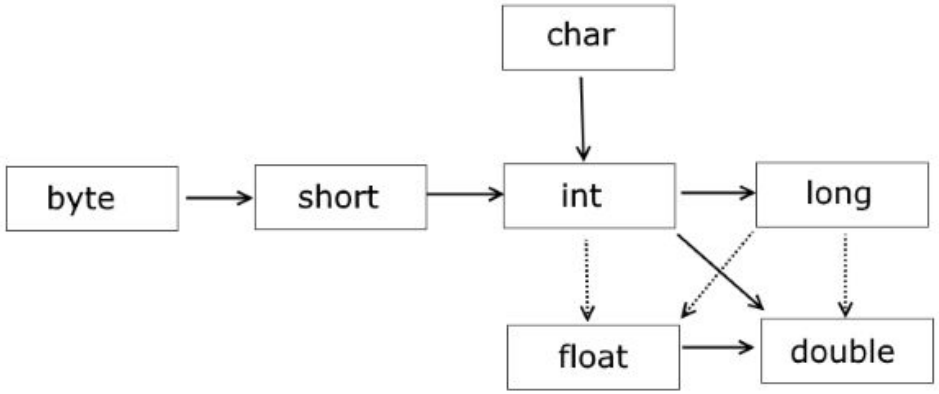
Ключевые слова

Java:

abstract	continue	for	new	switch
assert	default	goto*	package	synchronized
boolean	do	if	private	this
break	double	implements	protected	throw
byte	else	import	public	throws
case	enum	instanceof	return	transient
catch	extends	int	short	try
char	final	interface	static	void
class	finally	long	strictfp	volatile
const*	float	native	super	while

Приведение типов данных

Автоматическое преобразование



Стрелками на рисунке показано, какие преобразования типов могут выполняться автоматически. Пунктирными стрелками показаны автоматические преобразования с потерей точности.

Преобразования при операциях

Нередки ситуации, когда приходится применять различные операции, например, сложение и произведение, над значениями разных типов. Здесь также действуют некоторые правила:

- если один из операндов операции относится к типу **double**, то и второй операнд преобразуется к типу **double**
- *если предыдущее условие не соблюдено, а один из операндов операции относится к типу **float**, то и второй операнд преобразуется к типу **float** (для Java)*
- *если предыдущие условия не соблюдены, один из операндов операции относится к типу **long**, то и второй операнд преобразуется к типу **long** (для Java)*
- иначе все операнды операции преобразуются к типу **int**

Явное преобразование

```
double d1 = 56.9898;
int i7 = (int)d1;
System.out.println(i7);      // 56
```

Операторы сравнения

== сравнивает два операнда на равенство и возвращает true (если операнды равны) и false (если операнды не равны)

!= сравнивает два операнда и возвращает true, если операнды НЕ равны, и false, если операнды равны

< (меньше чем) возвращает true, если первый операнд меньше второго, иначе возвращает false

> (больше чем) возвращает true, если первый операнд больше второго, иначе возвращает false

>= (больше или равно) возвращает true, если первый операнд больше второго или равен второму, иначе возвращает false

<= (меньше или равно) возвращает true, если первый операнд меньше второго или равен второму, иначе возвращает false

Операторы сравнения

```
boolean a = true;
boolean b = false;
boolean c = false;


System.out.println(a == b); //false
System.out.println(c == b); //true

System.out.println(a != b); //true
System.out.println(c != b); //false

System.out.println(a > b); //false
System.out.println(a < b); //true

System.out.println(a <= b); //true
System.out.println(c >= b); //true
```

Логические операторы

В рамках курса мы не будем сталкиваться с побитовыми операциями, поэтому рассмотрим приоритет только логических операций:

1. &&
2. ||
3. !

```
boolean a = true;
boolean b = false;

System.out.println(a && !(b || a));
System.out.println(a && !b || a);
```

A	B	A & B
0	0	0
1	0	0
0	1	0
1	1	1

A	B	A || B
0	0	0
1	0	1
0	1	1
1	1	1

A	¬A
0	1
1	0

Вопросы?