

§1. ВВЕДЕНИЕ. АРХИТЕКТУРА ФОН НЕЙМАНА.

ЯЗЫК ПРОГРАММИРОВАНИЯ JAVA

Цель занятия: познакомиться с архитектурой компьютера как с исполнителем задачи, запустить первую программу на Java и научиться решать простейшие задачи.

Введение в архитектуру персонального компьютера

Компьютер (от англ. compute — вычислять) сегодня — это машина для автоматизированного решения прикладных задач.

Нам — пользователям персональных компьютеров — за счет стремительного развития множества замечательных технологий стало возможно быстро искать информацию в Интернете, монтировать видеоролики и играть онлайн вместе с друзьями. Однако мы не часто задумываемся о том, что происходит у компьютера “под капотом”.

Давайте рассмотрим то, что происходит, когда мы играем GTA V Online. Для взаимодействия с виртуальным миром мы используем **устройства ввода** (input) — клавиатуру, мышь или джойстик.



Рис. 1 Устройства ввода для компьютерных игр

Когда мы нажимаем на клавишу или кнопку, происходит замыкание контакта и по соединительному кабелю USB или через Bluetooth передается сигнал на разъем компьютера. Предположим, мы на протяжении пяти секунд непрерывно отправляли сигнал W (вперед), находясь в виртуальном автомобиле Comet.



Рис. 2 Процесс взаимодействия пользователя с виртуальной средой

В игру вступает сердце компьютера — **центральный процессор** (или CPU, т. е. Central Processing Unit), который очень быстро исполняет **последовательные инструкции**, которые запрограммировали разработчики игры из Rockstar и в зависимости от смысла инструкций объединили в **потoki** (от англ. threads).

Экземпляр GTA V на центральном процессоре (CPU)				
<p>Поток ввода</p> <p>Принимает сигналы с устройств ввода: клавиатуры, джойстика, мыши и пр.</p>	<p>Поток логический</p> <p>Обрабатывает игровые события, реализует всю игровую логику</p>	<p>Поток сетевой</p> <p>С помощью сетевой карты отправляет запросы/ответы на сервер и поддерживает виртуальный мир в согласованном представлении</p>	<p>Поток графический</p> <p>Получает задания на рендер (т. е. отрисовку) и рисует ее с помощью графической карты</p>	<p>Поток вывода</p> <p>Передает готовую игровую сцену на экран</p>

Рис. 3 Упрощенное представление потоков игровых приложений

Сигнал с клавиатуры будет обработан **потокком ввода** (input) и передан в качестве игрового события на **логический поток**. На нем будут проанализированы возможность продвижения вперед (может быть, мы уже врезались в стену) и игровой контекст (идем ли мы пешком или находимся в транспортном средстве). **Сетевой поток** передаст запрос на сервер, который уведомит остальных игроков о нашем перемещении в пространстве и согласует состояние игрового мира.

В текущий момент времени необходимо “отрендерить” игровую сцену, т. е. отрисовать ее, за что отвечает **графический поток**. Это самая вычислительно сложная задача, поэтому процессору для нее требуется в команду отдельное устройство — **графическая карта**.

В основе такой масштабной игры, как GTA V, лежит огромная математическая модель, данные для которой необходимо “держать в голове”. Это помогает делать **оперативная**

память (или RAM, т. е. Random Access Memory), т. е. информация для оперативного использования. Вся игра находится в **постоянной памяти** (на жестком диске) и занимает около 60 ГБ, когда у большинства пользователей есть всего лишь 4 или 8 ГБ оперативной памяти. Именно поэтому заполнение оперативной памяти стараются минимизировать, поскольку она не бесконечна.



Рис. 4 Взаимодействие пользователя с ПК

Когда модель рассчитана и сцена отрисована, остается вывести последнюю на экран, за что отвечает **поток вывода** (output). Так, наша машина переместилась в игровом мире, и мы можем наблюдать результат: мы едем.

Архитектура фон Неймана

На примере огромного приложения GTA V мы рассмотрели то, как пользователи в принципе взаимодействуют с компьютером, но остается добавить немного обобщения и формализации, что и сделал в 1940-х гг. ученый Джон фон Нейман из Принстонского университета.



Рис. 5 Компьютер как черный ящик

С должной степенью уверенности можно сказать, что компьютер по своей сущности принимает входные данные, преобразует их и возвращает выходные данные. В этом кибернетическом смысле мы очень похожи на компьютер.

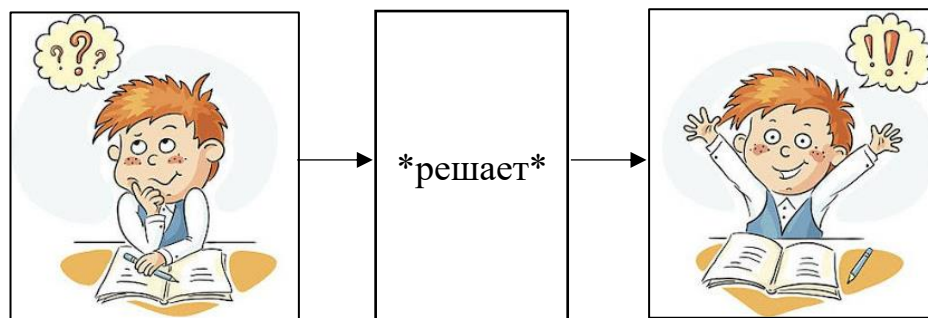


Рис. 6 Решая задачи, мы тоже начинаем с “Дано:” и заканчиваем с “Ответ:”

Но люди и машины, очевидно, отличаются. Человек с помощью разума способен находить новые ответы, а компьютер лишь **исполняет** заложенную в него программу. Одна из классических моделей, в рамках которой проектируют компьютер для исполнения программ, называется **архитектура фон Неймана**.

Давайте рассмотрим такое фундаментальное понятие, как **память** — набор ячеек с адресами фиксированного размера.

По фон Нейману, память состоит из **областей данных** и **областей инструкций**. Мы используем области данных для хранения значений, а области инструкций — для записи команд над ячейками со значениями.

Оперативная память (RAM)	Адрес	Область	Значение ячейки памяти	Комментарий
	0x0000	Область данных		Число А
	0x0001	Область инструкций	Считать число в 0x0000	⇐ Точка входа
	0x0002		Перейти на 0x0004	
	0x0003	Область данных		Число В
	0x0004	Область инструкций	Считать число в 0x0003	
	0x0005		Сложить числа 0x0000 и 0x0003 в ячейку 0x0008	
	0x0006		Вывести число 0x0008	
	0x0007		Конец программы ☺	⇒ Точка останова
	0x0008	Область данных		Число С = А + В

Рис. 7 Программа для сложения двух чисел

Действительно, мы можем разделить ячейки на несколько типов и записывать сколь угодно сложные программы. Это всё, конечно, замечательно, но нам нужен кто-то, кто сможет прийти и исполнить эту программу. Кто это и что ему требуется?

Это **процессор (или CPU)**, и он умеет выполнять простые команды над ячейками. Например:

- Считывать числа в ячейку и выводить числа из ячейки
- Производить простые арифметические операции над числами в ячейках
- Выполнять логические операции над ячейками
- Перемещаться по командам программы по адресу

Для исполнения классическому процессору требуется две важных вещи:

- Знать, откуда ему начинать читать программу, — **точку входа**
- Верить, что когда-нибудь программа закончится, — встретить **точку останова**

Оперативная память (RAM)	0x0000	Область данных		Число А
	0x0001	Область инструкций	Считать число в 0x0000	← Точка входа
	0x0002		Перейти на 0x0004	
	0x0003	Область данных		Число В
	0x0004	Область инструкций	Считать число в 0x0003	
	0x0005		Сложить числа 0x0000 и 0x0003 в ячейку 0x0008	
	0x0006		Вывести число 0x0008	
	0x0007		Конец программы ☺	⇒ Точка останова
	0x0008	Область данных		Число C = A + B

Рис. 8 Исполнение программы процессором

С некоторым приближением можно сказать, что современные компьютеры продолжают работать (во всяком случае, для программистов) именно по этой схеме: в память помещают программу и отдают ее на исполнение процессору.

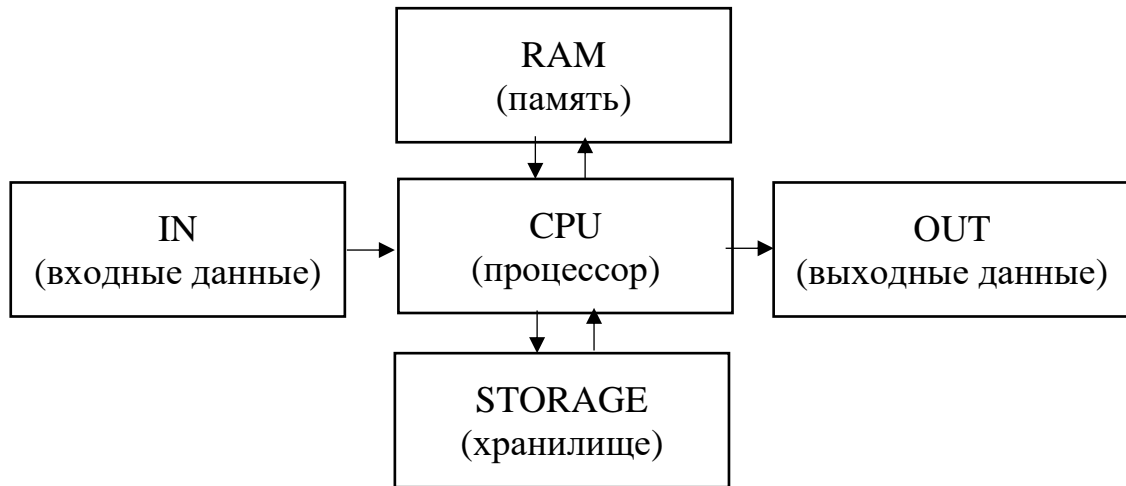


Рис. 9 Схема архитектуры фон Неймана

Hello, world!

Не кажется ли вам, что все эти разговоры про таинственные адреса вида 0x0007 очень запутанные? Вы не одиноки!

Язык программирования — формальная знаковая система для записи программ, которая позволяет избежать проблемы:

- **Непосредственной работы с адресами**

На замену ячейкам памяти с адресами приходят **переменные** — именованные области данных для хранения значений определенного типа

- **Физического выделения памяти**

Нам не нужно делать разметку памяти на уровне адресов: мы вводим переменные, а всю остальную работу делают автоматические инструменты

Давайте рассмотрим, как будет выглядеть самая простая программа на языке Java:

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

Код 1. Программа “Hello, world!” на языке Java

Все равно выглядит сложновато? Давайте пока не будем углубляться в то, что делают строки, кроме подчеркнутой, но ограничимся определением, что они определяют **точку входа** программы в языке Java. Это значит, что исполнение кода по соглашению всегда начинается с **метода main**. Здесь отсутствует какой-то особый смысл, но мы почувствуем красоту этого кода чуть позже в рамках нашего курса.

Чтобы запустить программу на Java, можно воспользоваться средой разработки IntelliJ. Рекомендуется установить следующие настройки для проекта:

- В Project Explorer отметить папку проекта как Sources Root (правой кнопкой мыши)
- File => Project Structure...
 - Project SDK: 11 (или та Java SDK, которая у вас установлена)
 - Project Language Level: 11
 - Project compiler output: на один уровень выше папки проекта
Например, проект лежит в D:\JavaCourse\, тогда можно взять D:\
- Run/Debug Configurations
 - Шаблон Application
 - Main class: Main
 - JRE: 11

Остается запустить с помощью меню в правом верхнем углу:



1. *Run. Запустить программу для выбранной конфигурации*
2. *Debug. Запустить программу в режиме отладки (чуть медленнее, но зато очень пригодится для анализа наших программ)*
3. *Build. Собрать, но не запускать (взрослым разработчикам иногда такое нужно, что же поделать — из песни слов не выкинешь)*

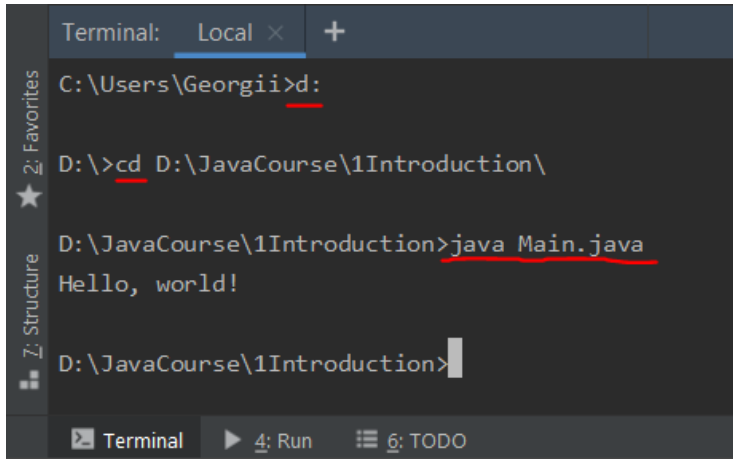
Рис. 10 Запуск программ в IntelliJ

Результатом выполнения программы является выведенная в стандартный поток классическая фраза “Hello, world!”, с которой свой путь по традиции начинают все программисты. Поздравляю, мы на верном пути!

Hello, world!

Код 2. Результат исполнения программы 1

Для быстрого запуска программ можно также использовать терминал IntelliJ или командную строку Windows:



```
Terminal: Local x +
C:\Users\Georgii>d:
D:\>cd D:\JavaCourse\1Introduction\
D:\JavaCourse\1Introduction>java Main.java
Hello, world!
D:\JavaCourse\1Introduction>
```

1. Команда `d:` переключает текущий диск
2. Команда `cd` изменяет текущую директорию
3. Команда `java` запускает программу в текущем терминале

Рис. 11 Запуск программ в терминале IntelliJ

К сожалению, всех этих технических подробностей сложно избежать, так как они могут понадобиться прямо сейчас, но хорошая новость в том, что они подходят к концу.

Давайте вернемся к сути нашей первой команды:

```
System.out.println("Hello, world!");
```

Код 3. Действующая команда программы 1

Помните, мы обсуждали то, что для программы существуют входные и выходные данные? В Java мы называем их `System.in` и `System.out` или **стандартными потоками ввода/вывода**.

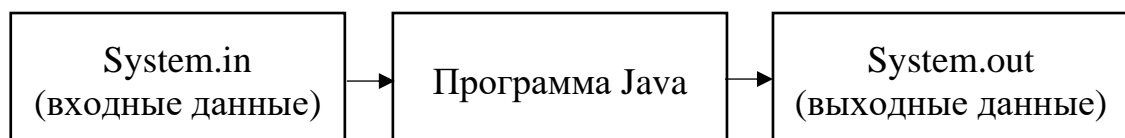


Рис. 12 Упрощенное представление программ на Java

Команда `System.out.println()` выводит строку в стандартный вывод, и это прекрасно. Но давайте посмотрим на несколько других команд и приступим к задачам.

Начальные приемы на Java

Так повелось, что в Java проще выводить данные, чем читать. Для чтения принято использовать обертку Scanner. Пусть это будет последней вещью, которую мы пока примем на веру.

```
import java.util.Scanner;

public class Sum {
    public static void main(String[] args) {
        Scanner in = new Scanner(System.in);    // 1
        int a = in.nextInt();                    // 2
        int b = in.nextInt();                    // 3
        int c = a + b;                           // 4
        System.out.print(c);                     // 5
    }
}
```

Код 4. Программа для сложения двух чисел (аналог программы на рис. 6)

Простейшим типом в Java является целочисленный тип — `int` (от англ. integer). На строках 2 и 3 мы вводим две переменные, в которые положим числа из стандартного входа. На строке 4 происходит объявление новой переменной и задание ей значения суммы двух переменных. Строка 5 выводит результат в стандартный вывод.

Функция оператора	Формат использования	Комментарий
Присваивание	<code>a = b</code>	Значение переменной <code>a</code> становится равно значению <code>b</code>
Сложение	<code>a + b</code>	
Вычитание	<code>a — b</code>	
Умножение	<code>a * b</code>	
Деление	<code>a / b</code>	
Взятие по модулю	<code>a % b</code>	Возвращает такое число <code>x</code> , что существует <code>k</code> , для которого <code>a = k * b + x</code>
Прибавление числа	<code>a += b</code>	Эквивалентно <code>a = a + b</code>
<code>import java.lang.*;</code>		
Квадратный корень	<code>Math.sqrt(a)</code>	Возвращает <code>double</code> (см. задачу D)

Максимальный элемент	Math.max(a, b)	Возвращает неменьшее из двух значение
Минимальный элемент	Math.min(a, b)	Возвращает неольшее из двух значений
Модуль числа $ a $	Math.abs(a)	

Рис. 13 Операции с числами

Задачи

А. Ввод-вывод

В стандартном входе даны целые длины катетов прямоугольного треугольника.

Входные данные: длины катетов $a, b \in \mathbb{Z}$ и $a, b < 10000$

Выходные данные: с помощью команды `println()` выведите

- Площадь треугольника
- Квадрат длины гипотенузы

System.in	System.out
3 4	12 25

В. Замена (swap)

Запишите два числовых значения в переменные `a` и `b` и поменяйте значения этих переменных местами.

Входные данные: $a, b \in \mathbb{Z}$ и $a, b < 10000$

Выходные данные: с помощью команды `println()` выведите эти два числа в порядке после замены.

System.in	System.out
16 20	20 16

С. Замена для любителей арифметики

Решите предыдущую задачу, не используя **дополнительных переменных**.

Д. Квадратное уравнение

Известно, что с выбранными коэффициентами a, b, c уравнение $ax^2 + bx + c = 0$ имеет ровно два различных решения. Найдите и выведите эти решения.

Примечание. В языке Java дробные значения (т. е. с плавающей точкой) хранят не в `int`, а в **`double`** или **`float`**. Такие числа записываются в коде как `0.95`, но могут быть поданы в стандартный ввод как `"0.95"` или `"0,95"` в зависимости от платформы. Чтобы читать числа, записанные с помощью точки, используйте следующий код:

```
Scanner in = new Scanner(System.in).useLocale(Locale.US);
```

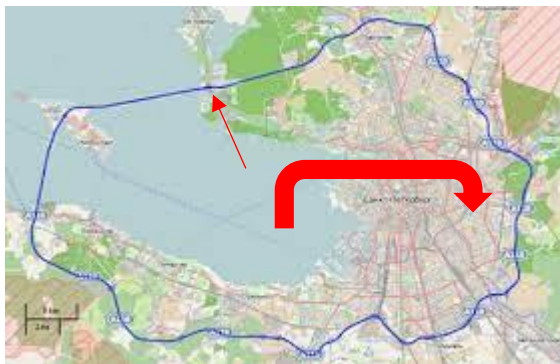
Входные данные: целые коэффициенты a, b и $c \in \mathbb{Z}$ и < 10000 .

Выходные данные: два решения уравнения в порядке убывания через `println()`.

System.in	System.out
1 3 -4	1 -4

Е. Путешествие по КАД

Санкт-Петербург славен своей Кольцевой Автомобильной Дорогой, нулевой километр которой начинается в поселке Лисий нос Приморского района.



Дмитрий получил водительские права и хочет начать свой водительский опыт с поездки по КАД, длина которой — 142 км. Помогите ему решить несколько задач.

Примечание. Говорят, что кто-то находится на i — ом километре, если его координата находится между i — ым километром и $i + 1$ — ым километром.

Пример. Если мы проехали 400 м к Кронштадту, то мы окажемся на 141-ом километре. Если мы проедем 600 м от Кронштадта, то мы окажемся на 0-ом километре.

Задача 1. Дмитрий начинает движение с нулевого километра с постоянной целочисленной скоростью v км/ч и двигается по кругу на протяжении неотрицательного рационального времени t ч. Найдите, на каком километре КАД окажется Дмитрий через указанное время.

Входные данные: $v \in \mathbb{Z}$ в км/ч, $|v| \leq 10000$, $0 \leq t \in \mathbb{Q} \leq 10000$ ч

Выходные данные: километр КАД от 0 до 141.

System.in	System.out
60 5.5	46
-40 2	62

Задача 2. Дмитрий попросил своего опытного друга-водителя Даниила составить ему компанию и поехать навстречу по КАД в обратном направлении. Даны две целочисленные скорости по модулю v_1 и v_2 км/ч. Найдите время в ч, через которое друзья встретятся.

Входные данные: даны $v_1 \in \mathbb{Q}$ км/ч (скорость Дмитрия) и $v_2 \in \mathbb{Q}$ км/ч (скорость Даниила), $0 \leq v_1$ и $v_2 \leq 10000$.

Выходные данные: время встречи в ч.

System.in	System.out
63 79	1

Задача 3. Дмитрий и Даниил устали ездить навстречу друг-другу. Они выпили кофе в Сестрорецке и начали движение из нулевого километра по часовой стрелке на своих машинах. Выяснилось, что машины отличаются скоростями, и Даниил, сам того не заметив, уехал далеко вперед. Но Дмитрий, будучи неуверенным водителем, боится разворачиваться на эстакадах и намерен продолжать движение только вперед, дождавшись, пока Даниил сделает круг и нагонит его сзади. Определите, сколько времени в минутах необходимо для этого.

Входные данные: даны $v_1 \in \mathbb{Q}$ км/ч (скорость Дмитрия) и $v_2 \in \mathbb{Q}$ км/ч (скорость Даниила), $0 \leq v_1$ и $v_2 \leq 10000$.

Выходные данные: время в минутах до их первой встречи.

System.in	System.out
20 80	142

Вопросы

1. Что такое **переменная**?

Назовите не менее трех преимуществ использования переменных (например, **int x**) вместо непосредственной адресации к ячейкам памяти (например, **0x0007**).

2. Рассмотрите следующие программы. Определите:

- Являются ли они корректными программами. Почему?
- Если являются, то что они делают?

Программа А

0x0000		
0x0001	Считать число в 0x0000	⇐ Точка входа
0x0002	Прибавить 1 к числу в ячейке 0x0000	
0x0003	Прибавить 1 к числу в ячейке 0x0000	
0x0004	Прибавить 1 к числу в ячейке 0x0000	
0x0005	Вывести число 0x0000	
0x0006	Конец программы ☺	

Программа В

0x0000	
0x0001	Считать число в 0x0000
0x0002	Прибавить 1 к числу в ячейке 0x0000

0x0003	Вывести число 0x0000
0x0004	Конец программы ☺

Программа С

0x0000		
0x0001	Считать число в 0x0000	⇐ Точка входа
0x0002	Прибавить 1 к числу в ячейке 0x0000	
0x0003	Перейти на 0x0002	
0x0004	Вывести число 0x0000	
0x0005	Конец программы ☺	