**Как работает компьютер**

**Что надо знать программисту при написании программ**

Сегодня мы поговорим о базовых знаниях из области информатики и программирования.

*Если материал кажется сложным или, наоборот, простым, можно промотать ниже — там начинается чистое программирование.*

**Принципы работы современного компьютера на основе двоичной логики**

Практически любая современная вычислительная машина, будь то настольный компьютер, ноутбук, планшет или смартфон, работает на основе двоичной логики.

У человека 10 пальцев рук, потому нам знакома и понятна десятичная [система счисления](https://ru.wikipedia.org/wiki/%D0%A1%D0%B8%D1%81%D1%82%D0%B5%D0%BC%D0%B0_%D1%81%D1%87%D0%B8%D1%81%D0%BB%D0%B5%D0%BD%D0%B8%D1%8F). То есть, мы считаем десятками: счет после 9 начинается с нового десятка, точнее – порядка. Потому при записи используется 10 цифр: от 0 до 9.

Если бы у людей было 16 пальцев, метот записи был бы совсем другим. В [16-ричной системе счисления](https://ru.wikipedia.org/wiki/%D0%A8%D0%B5%D1%81%D1%82%D0%BD%D0%B0%D0%B4%D1%86%D0%B0%D1%82%D0%B5%D1%80%D0%B8%D1%87%D0%BD%D0%B0%D1%8F_%D1%81%D0%B8%D1%81%D1%82%D0%B5%D0%BC%D0%B0_%D1%81%D1%87%D0%B8%D1%81%D0%BB%D0%B5%D0%BD%D0%B8%D1%8F) для недостающих символов используются буквы. Сначала ситаем от 0 до 9, а потом идут буквы латинского алфавита: A, B, C, D, E и F.

Из математики известно, что это далеко не самые экономичные системы, и для вычислительных машин гораздо эффективнее будет [троичная система счисления](https://ru.wikipedia.org/wiki/%D0%A2%D1%80%D0%BE%D0%B8%D1%87%D0%BD%D0%B0%D1%8F_%D1%81%D0%B8%D1%81%D1%82%D0%B5%D0%BC%D0%B0_%D1%81%D1%87%D0%B8%D1%81%D0%BB%D0%B5%D0%BD%D0%B8%D1%8F). Она устроена так: счет после 0, 1 и 2 начинается с нового порядка: следующими числами будут 10, 11 и 12. Затем – новый порядок: 20, 21, 22. И так далее. Подробнее о троичной системе счисления можно прочитать в [статье википедии](https://ru.wikipedia.org/wiki/%D0%A2%D1%80%D0%BE%D0%B8%D1%87%D0%BD%D0%B0%D1%8F_%D1%81%D0%B8%D1%81%D1%82%D0%B5%D0%BC%D0%B0_%D1%81%D1%87%D0%B8%D1%81%D0%BB%D0%B5%D0%BD%D0%B8%D1%8F).

Но современный компьютер работает на [двоичной логике](https://ru.wikipedia.org/wiki/%D0%94%D0%B2%D0%BE%D0%B8%D1%87%D0%BD%D0%B0%D1%8F_%D1%81%D0%B8%D1%81%D1%82%D0%B5%D0%BC%D0%B0_%D1%81%D1%87%D0%B8%D1%81%D0%BB%D0%B5%D0%BD%D0%B8%D1%8F)! При поразрядном сравнении троичная система счисления оказывается более ёмкой, чем двоичная система счисления. Но двоичный компьютер гораздо проще при проектировании и изготовлении. Да и программистам это удобнее. Несмотря на то, что в 50-х и 60-х годах XX века был произведен не один десяток троичных компьютеров (посмотрите статью про [ЭВМ «Сетунь»](https://ru.wikipedia.org/wiki/%D0%A1%D0%B5%D1%82%D1%83%D0%BD%D1%8C_(%D0%BA%D0%BE%D0%BC%D0%BF%D1%8C%D1%8E%D1%82%D0%B5%D1%80))!), сегодня мы найдем только вычислительные устройства, основанные на двоичной логике.

Как работает современный компьютер? Он использует [булеву алгебру](https://ru.wikipedia.org/wiki/%D0%91%D1%83%D0%BB%D0%B5%D0%B2%D0%B0_%D0%B0%D0%BB%D0%B3%D0%B5%D0%B1%D1%80%D0%B0). Она названа в честь [Джорджа Буля](https://ru.wikipedia.org/wiki/%D0%91%D1%83%D0%BB%D1%8C,_%D0%94%D0%B6%D0%BE%D1%80%D0%B4%D0%B6), английского математика и логика.

**Булева алгебра**

В логике современного компьютера на двоичной логике есть только два состояния, два элемента: Истина (1) или Ложь (0). Если мы хотип записать число 4 в двоичной системе, мы будем считать так: сначала в этой системе, как и вдругих, идут 0 и 1, а потом цифры заканчиваются и начинается новый порядок. Число 2 — это 10, а 3 — 11. Дальше — снова новый порядок. Число 4 в двоичной записи — это 100.

Конечно, на одних числах компьютер не построишь. С ними необходимо совершать операции. И х всего три: это

1) **отрицание**; оно делает из 1 (Истина) 0 (Ложь) и наоборот.

2) две [бинарных операции](https://ru.wikipedia.org/wiki/%D0%91%D0%B8%D0%BD%D0%B0%D1%80%D0%BD%D0%B0%D1%8F_%D0%BE%D0%BF%D0%B5%D1%80%D0%B0%D1%86%D0%B8%D1%8F): [логическое И](https://ru.wikipedia.org/wiki/%D0%9A%D0%BE%D0%BD%D1%8A%D1%8E%D0%BD%D0%BA%D1%86%D0%B8%D1%8F) и

3) [логическое ИЛИ](https://ru.wikipedia.org/wiki/%D0%94%D0%B8%D0%B7%D1%8A%D1%8E%D0%BD%D0%BA%D1%86%D0%B8%D1%8F).

Работают они просто. Если мы сравниваем **1 И 1**, получим **1**:выражение истинно, если истинны оба значения. В остальных случаях (0 И 1, 1 И 0, 0 И 0) получим 0, или Ложь.

Если мы применим операцию ИЛИ, то **a ИЛИ b**будет истинным даже тогда, когда один из [операндов](https://ru.wikipedia.org/wiki/%D0%9E%D0%BF%D0%B5%D1%80%D0%B0%D0%BD%D0%B4) — 0: **1 ИЛИ 0 = 1**, **0 ИЛИ 1 = 1**.

В программировании операция сравнения — применяется очень часто. Допустим, мы хотим сравнить два набора: «101» и «100». Много у них общего?

Воспользуемся операцией И. первый разряд: 1 И 1 = 1. Второй разряд: 0 И 0 = 1 (операнды одинаковы, значит результат сравнения — Истина). Третий разряд: 1И 0 = 0 (операнды не равны, результат — ложь).

Из трех операций одна дала ложь. Значит, два набора не равны. Вместе с тем, два сравнения дали результат 1. Значит, у этих наборов есть одинаковые элементы.

**Архитектура вычислительной машины**

Для работы компьютера необходимо, чтобы он умел:

1) хранить различные значения (в двоичной системе — набор нулей и единиц),

2) производить операции с данными (наборами нулей и единиц),

3) Позволять пользователю компьютера вводить данные и команды,

4) Выводить данные на устройство вывода.

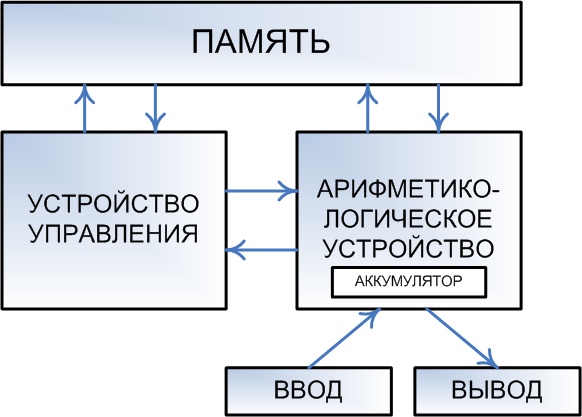
Нам это хорошо знакомо. Устройство вывода — это экран компьютера или смартфона. Устройство ввода — это клавиатура или сенсорный экран смартфона.

**Все, о чем мы говорим — наборы данных, команды, ввод и вывод — просто потоки нулей и единиц!**

*Только представьте — просмотр видео с Ютюба на экране — это результат обработки колоссального количества нулей и единиц и их вывод на миллионы отдельных точек экрана, которые, получая свой набор нулей и единиц, загораются и гаснут в нужное время!*

Все это возможно благодаря тому, что ввод, вывод и выполнение операций с данными производится с огромной скоростью — миллиарды операций в секунду!

В блок-схеме компьютера выделяют отдельные блоки. Это:



1) память, в ней хранятся данные и команды — операции, которые следует выполнить;

2) арифметико-логическое устройство, которое производит операции над данными;

3) устройство управления, которое заведует всеми процессами работы компьютера. В самом деле, кто-то же должен управлять этими бесконечными рядами нулей и единиц!

4) устройства ввода и вывода.

Современный компьютер основан на [архитектуре фон Неймана](https://ru.wikipedia.org/wiki/%D0%90%D1%80%D1%85%D0%B8%D1%82%D0%B5%D0%BA%D1%82%D1%83%D1%80%D0%B0_%D1%84%D0%BE%D0%BD_%D0%9D%D0%B5%D0%B9%D0%BC%D0%B0%D0%BD%D0%B0). Главная ее особенность — данные и команды хранятся в одной и той же памяти. Поэтому, если случайно записать данные в то место, где должна была быть записана команда, все ломается, и компьютер выдает ошибку. Исторически, это — одна из самых частых причин прекращения работы запущенной программы.

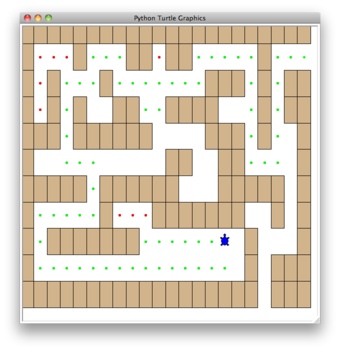
**Файлы и файловые системы**

Но человеку неудобно работать с нулями и единицами. Поэтому данные в памяти компьютера хранят в *файлах*, а эти файлы распределены по отдельным *директориям*или *папкам.*Все вместе составляет [файловую систему](https://ru.wikipedia.org/wiki/%D0%A4%D0%B0%D0%B9%D0%BB%D0%BE%D0%B2%D0%B0%D1%8F_%D1%81%D0%B8%D1%81%D1%82%D0%B5%D0%BC%D0%B0). На самом деле, чуть сложнее, но наам достаточно этого.

**Программирование: инструкции**

Теперь, наконец, поговорим о программировании. В самом простом случае, язык программирования (мы будем использовать [Python](https://ru.wikipedia.org/wiki/Python), или Питон) позволяет создать программу в виде набора инструкций. Эти инструкции будут выполняться последовательно: когда выполнена одна инструкция, компьютер приступает к выполнению следующей.

Допустим, что мы пишем программу по движению черепашки по лабиринту. Необходимо **решить задачу** — нарисовать линию пути черепашки по лабиринту.



Зная, как устроен лабиринт, мы напишем последовательность действий:

1) поверни налево (черепашка смотрит вверх),

2) ползи вперед 6 шагов,

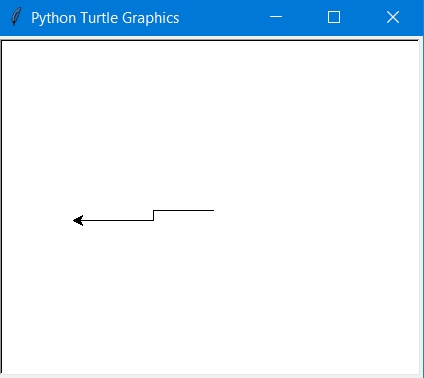
3) поверни на 90 градусов налево,

4) ползи 1 шаг,

5) поверни на 90 градусов направо,

6) ползи 8 шагов,

и так далее. **Программа приведет черепашку к выходу, только если шаги будут выполняться последовательно!**



Последовательность действий здесь — набор инструкций, [алгоритм](https://ru.wikipedia.org/wiki/%D0%90%D0%BB%D0%B3%D0%BE%D1%80%D0%B8%D1%82%D0%BC).

Мы говорили в тексте выше об инструкциях (командах) и данных. В нашем алгоритме **данные — это**:

1) углы поворота (90 градусов),

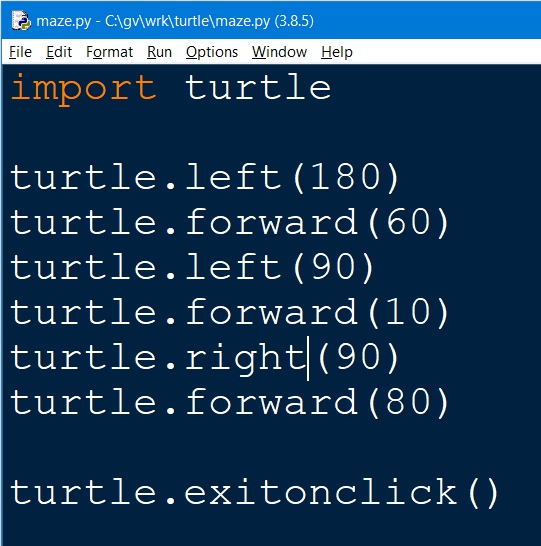
2) длина пути черепашки на текущем шаге.

**Инструкции — это**:

1) поверни на заданный угол,

2) ползи заданное количество шагов.

Для того, наш алгоритм заработал, его надо написать на выбранном языке программирования.



Вот наша программа на языке Python с [библиотекой turtle](https://ru.wikipedia.org/wiki/%D0%A7%D0%B5%D1%80%D0%B5%D0%BF%D0%B0%D1%88%D1%8C%D1%8F_%D0%B3%D1%80%D0%B0%D1%84%D0%B8%D0%BA%D0%B0) — та самая черепашка! Почитайте статью о черепашьей графике по ссылке выше, там описаны базовые команды.

Наша программа — текстовый **файл**, состоящий из 10 строк (две строки — пустые). Его расположение в **файловой системе** указано в заголовке окна: диск C, затем вложенные папки gv, wrk, turtle. Затем название файла — [maze](https://ru.wikipedia.org/wiki/%D0%9B%D0%B0%D0%B1%D0%B8%D1%80%D0%B8%D0%BD%D1%82).py

Первая строка — импорт нужных команд. turtle — название библиотеки,

следующие команды — начало движения черепашки (для удобства мы умножили количество шагов на 10). Последней командой мы просим закрыть окно только тогда, когда кликнем мышкой по нему.

Для выполнения алгоритма необходимо установить дистрибутив языка программирования Python. Его можно скачать по [этой ссылке](https://www.python.org/).

Затем следует запустить встроенную среду разработки [IDLE](https://ru.wikipedia.org/wiki/IDLE). В ней создать файл с текстом программы, сохранить его и запустить.

**Самостоятельная работа**

Попробуйте продолжить алгоритм движения черепашки до выхода!

*Главная цель — освоить процесс разработки на языке программирования Python.*

Позже мы вернемся к логическим операциям и двоичной логике, когда будем использовать *условные выражения*.