Урок 1

Начинаем программировать

Переменные. Типы данных. Ветвления. HTML+CSS. Устройство компьютера

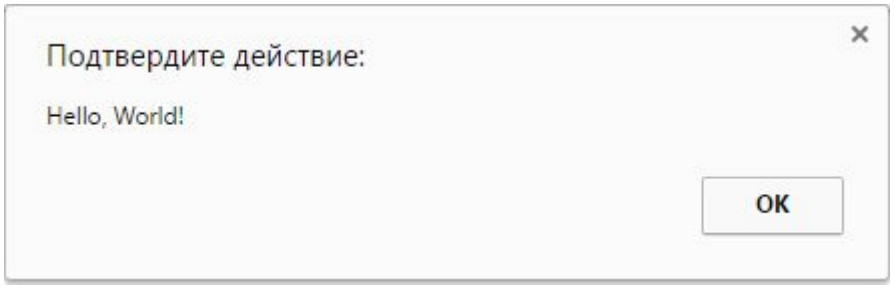
Первая программа

Запустите Блокнот. Наберите в редакторе следующий текст

```
<script>
    alert("Hello, World!");
</script>
```

Сохраните файл на Рабочий стол под названием 1.html. Расширение .html необходимо, чтобы данный файл открывался в браузере. Теперь запустите этот файл, щелкнув по нему мышкой два раза.

Если вы все сделали правильно, то у вас запустится ваш браузер и появится следующее окошко.



Вид окна может отличаться в разных версиях браузера. На курсе мы, как правило, используем браузер Chrome, хотя вы можете использовать браузер, к которому вы больше привыкли.

Это простая программа знакомит вас тремя понятиями:

- теги;
- строка;
- команда языка программирования;
- параметры команды.

Теги `<script></script>` не относятся к языку JavaScript. Это указание браузеру, что внутри них заключена программа, которую следует выполнить. Про теги мы поговорим попозже.

Строка. Любая последовательность символов заключенная в двойные кавычки является строкой в JavaScript. Строка - это одна из разновидностей данных, с которыми умеет обращаться язык JavaScript. В дальнейшем мы познакомимся с другими типами данных.

Слово alert это команда, которая заставляет браузер выводить окошко с кнопкой OK и текстом, который мы указали в скобках.

Параметры команды. У команд бывают параметры, с помощью которых мы сообщаем, как они должны работать. Описание команды alert выглядит так:

```
alert(message:string)
```

message - это подсказка программисту, что alert может обработать текст сообщения, а string - это подсказка программисту, что это сообщение должно быть строкой. Учтите, что это описание команды

alert. При использовании команды мы в скобки просто передаем строку, не указывая его тип. В конце строки можно поставить точку с запятой, но она не является обязательной.

Числовой тип и автоматическое преобразование в строковой тип

Теперь давайте изменим нашу программу следующим образом:

```
<script>
    alert(2016);
</script>
```

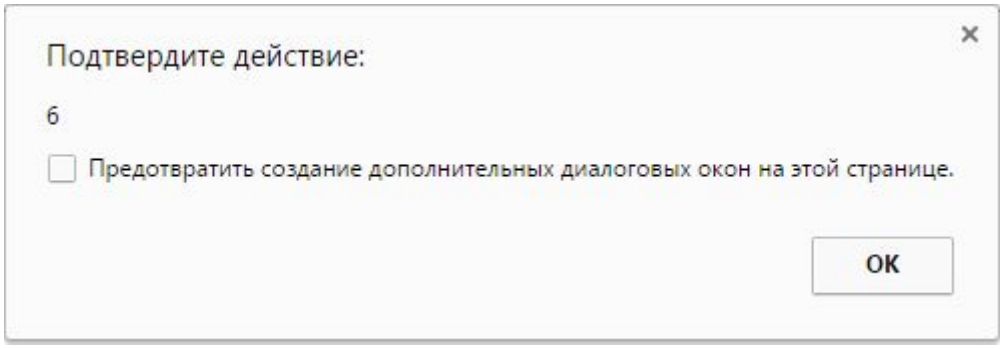
Эта программа выведет на экран число 2016. Но ведь 2016 не является строкой, скажете вы и будете правы. 2016 - это число и относится к числовому типу данных. Тогда почему alert вывел 2016 на экран, хотя в его описании указывается, что он выводит строковой тип данных? Дело в том, что JavaScript автоматически преобразовал 2016 в строковой тип. Чтобы в этом разобраться, нужно изучить, как компьютер хранит в памяти различные данные. Познакомиться с двоичным кодированием. Пока достаточно понять, что при необходимости JavaScript автоматически преобразовывает данные из одного типа данных в другой.

Вычисления

Измените программу, поместив в скобки alert арифметическое выражение:

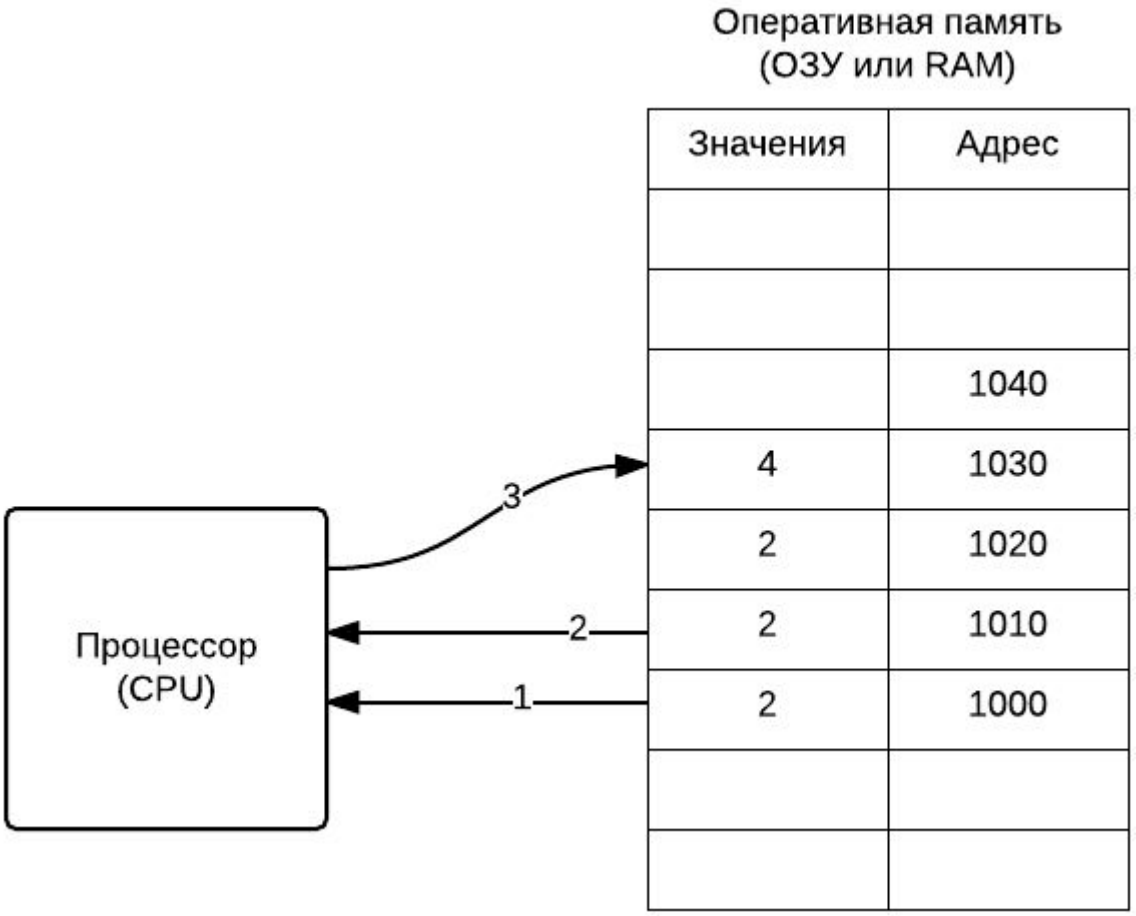
```
<script>
    alert(2 + 2 * 2);
</script>
```

Сохраните и посмотрите результат в браузере



Как вы и ожидали, прежде, чем вывести результат на экран, компьютер вычислил выражение в скобках, преобразовал результат в текст и выполнил команду alert. Компьютер достаточно умен, чтобы понимать, какую операцию выполнять в первую очередь. Если нужно изменить приоритет операции, то используйте скобки, например: alert((2+2)*2).

Полезно понять, что для вычисления компьютер анализирует программу и сохраняет числа в оперативной памяти, каждое число в своей ячейке. Для вычисления числа передаются в процессор, где складываются (умножаются, вычитаются, делятся...), и результат помещается обратно в оперативную память. Процесс вычисления 2+2 схематически показан на рисунке ниже.

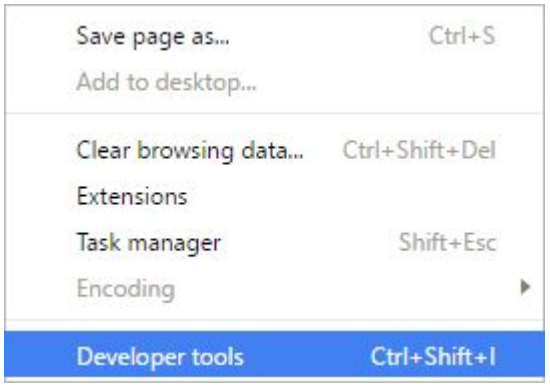


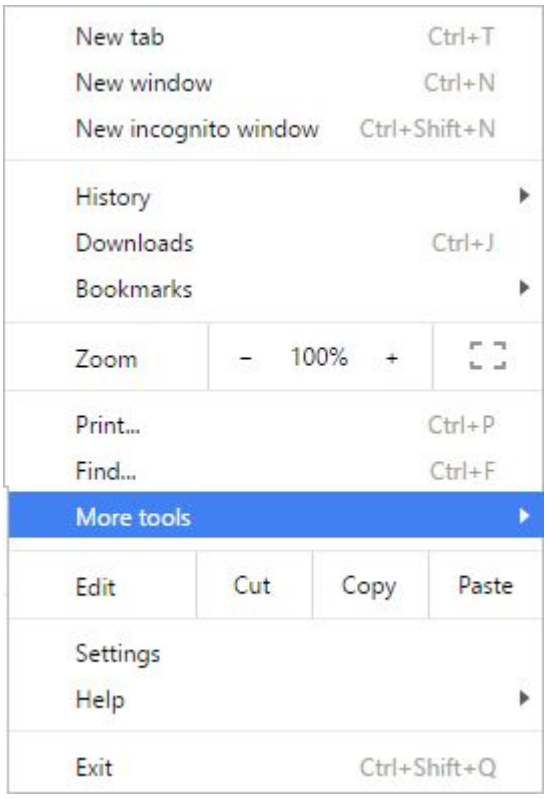
Отладка программы

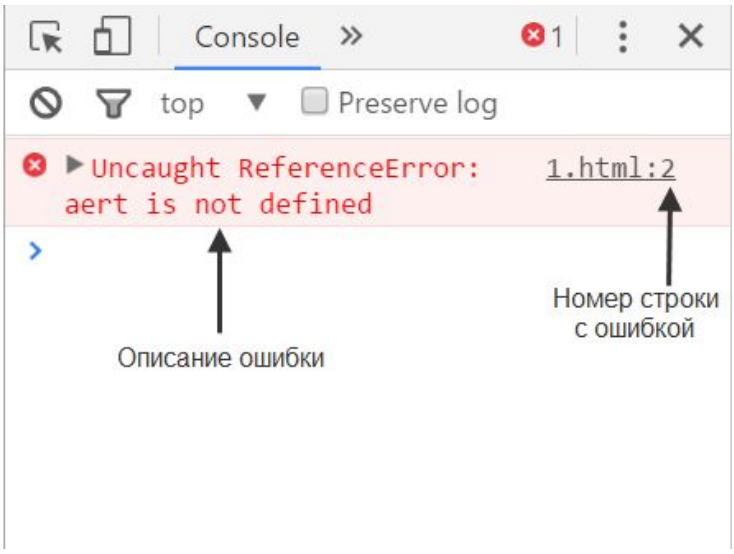
Удалите в предыдущей программе в команде alert какую-нибудь букву. Запустите программу.

```
<script>
    aert(2 + 2 * 2);
</script>
```

В этом случае программа не запустится, так как она написана с ошибкой. В браузерах есть специальные инструменты, которые помогают программистам искать ошибки в программах. В браузере Chrome выберите Дополнительно-Инструменты разработчика и щелкните на вкладке Console (или нажмите Ctrl+Shift+I и перейдите на вкладку Console).







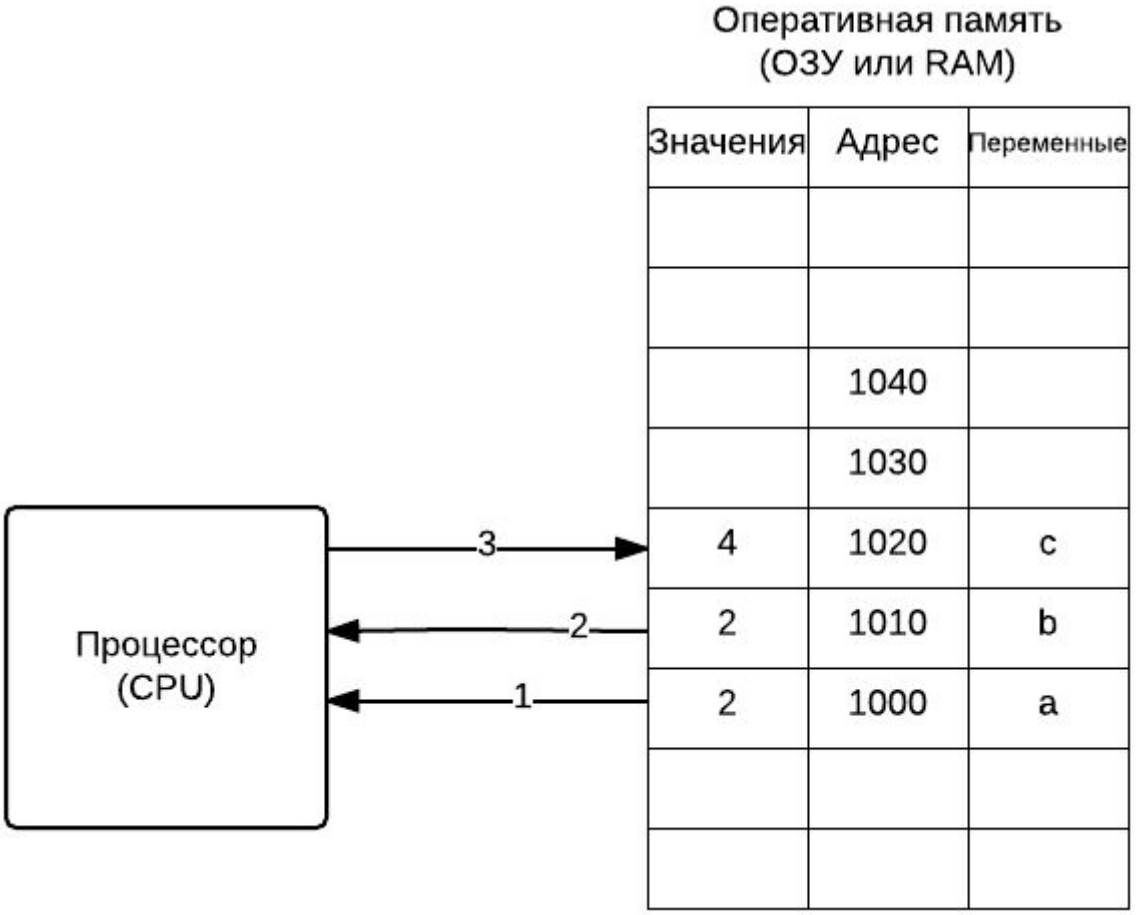
Если программа написана с ошибкой, то в консоли выводится сообщение о том, в какой строчке допущена ошибка и описание ошибки. Начинающим программистам описание ошибки может и не поможет, но хотя бы увидите, к какой строке нужно присмотреться. Исправьте ошибку, сохраните программу и перезагрузите страницу в браузере.

Переменные. Операция присваивания

Наберите и запустите следующую программу

```
<script>
    var a = 2;
    var b = 2;
    var c = a + b;
    alert(c);
</script>
```

В общем-то, ничего особенного не произошло. Программа вывела на экран сумму двух чисел. Но теперь для хранения чисел используются переменные. Переменные нужны для хранения изменяющихся данных.



На самом деле переменные это именованные адреса ячеек, то есть, обращаясь к переменной **a,** мы обращаемся к ячейке в памяти компьютера, и написав команду a=2, мы указываем компьютеру, что хотим положить в ячейку, с которой связана переменная **a,** значение 2. Первые программисты были настолько суровы, что писали программы без использования переменных.

Для изменения значения переменной используется операция **присваивания**, которая в JavaScript обозначается знаком равенство (=). Несмотря на свой незамысловатый вид, это одна из самых

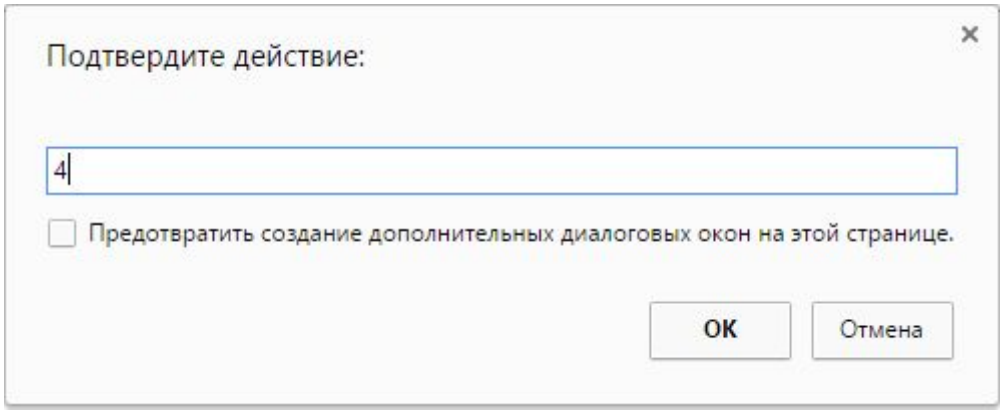
важных операций в программировании, поэтому важно понимать что происходит, когда вы ее используете.

Ввод данных

Ввод данных в программу может производиться разными способами. Это может быть клавиатура, мышь, касание экрана, считывание данных из файла или из базы данных. Но в каждом языке есть команда, с которой начинают изучать ввод данных. В JavaScript это команда **prompt**. Наберите следующую программу и запустите её на выполнение:

```
<script>
    var a;
    a = prompt();
    alert(a);
</script>
```

Эта программа выводит окно со строкой, куда пользователь может вводить данные.



Введенные данные сохраняются в переменной, и теперь мы можем вывести их на экран.

Склеивание строк

Давайте рассмотрим следующий пример:

```
<script>
    var a;
    a = prompt();
    alert(a + " is a good number!");
</script>
```

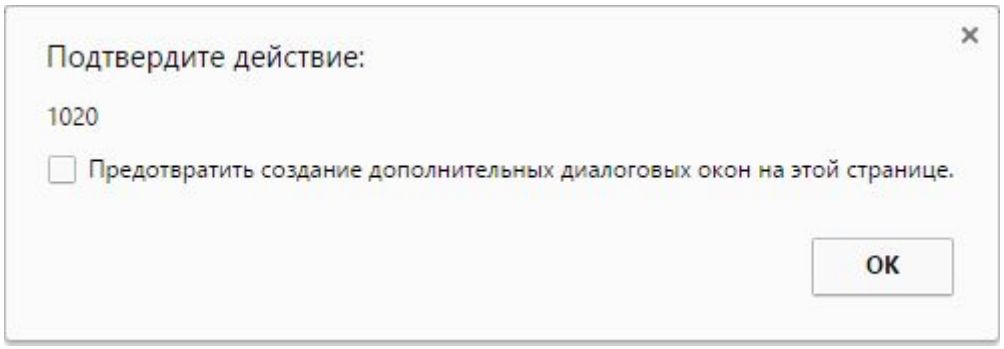
Здесь мы демонстрируем операцию склеивания строк. Для строк знак плюс означает, что должна получится новая строка, состоящая из нескольких строк, стоящих слева и справа от знака +.

Преобразование из строки в число

Мы уже демонстрировали, что, при необходимости, JavaScript переводит число в строку. Что делать, если нужно сделать наоборот? Рассмотрим пример:

```
<script>
    var a = prompt("a:");
    var b = prompt("b:");
    var c = a + b;
    alert(c);
</script>
```

Запустите программу и введите числа 10 и 20. Вместо того, чтобы сложить два числа, программа склеит две строки. В этом нет ничего удивительного, если знать, что результатом выполнения команды prompt является строка.



Посмотрим на описание команды prompt:

```
function(message:string,value:string):string
```

Описание команды еще называют синтаксисом.

Ничего страшного, что вы пока не понимаете это описание. Обратите внимание, что в конце записи стоит :string. Это означает, что prompt возвращает строковое значение. (Смысл слова “возвращает” мы разберём подробнее , когда будем изучать функции.) Значит компьютер принял от prompt строчку и поступил правильно, когда склеил две строки, вместо того, чтобы сложить числа. Для того, чтобы он записал в переменные числа, а не строки, нужно указать ему на это. Самый простой способ - поставить перед prompt символ +.

```
<script>
    var a = +prompt("a:");
    var b = +prompt("b:");
    var c = a + b;
    alert(c);
</script>
```

Другой способ преобразования из строки в число - использование функции parseInt

```
<script>
    var a = parseInt(prompt("a:"));
    var b = parseInt(prompt("b:"));
    var c = a + b;
    alert(c);
</script>
```

Логический тип данных

Рассмотрим программу.

```
<script>
    var a = 2 * 2 == 4;
    alert(a);
</script>
```

Это программа при запуске выведет true, что в переводе с английского означает “истина”. Здесь мы с вами знакомимся с еще одним типом данных - логическим. В переменную **a** поместился результат операции **отношения** равенство (==). Сначала посчиталась левая часть, операция 2*2, и сравнилась с правой частью 4. Так как результаты равны, результат операции отношения “истина”. Результат операции присвоился в переменную a.

Логический тип данных может принимать всего два значения: истина и ложь. В JavaScript используется специальные значения для их обозначения: true и false.

В JavaScript существуют следующие операции отношения:

- == - равенство
- < - меньше
- > - больше
- >= - больше или равно
- <= - меньше или равно
- != - не равно

Алгоритмы

В программистском сообществе постоянно идут споры: нужны алгоритмы или нет. Точнее, обычно все признают, что алгоритмы важны, но вот стоит ли записывать их или можно сразу писать программу - это вызывает бурные дискуссии. Многие “профессиональные” программисты высокомерно заявляют, что они пишут свои программы вообще не используя алгоритмы или, точнее, не записывая их на бумаге. Не будем с ними спорить. Мы считаем, что если изучать язык без алгоритмов, то программист в будущем будет строить программы, используя конструкции первого языка. Вообще-то, вряд ли от этого получится избавиться и при изучении алгоритмов, но, все-таки, написание алгоритма позволяет мыслить более широко, давая возможность сосредоточиться на решении задачи, а не на реализации её на конкретном языке.

Слово “алгоритм” произошло от имени Мухаммеда ал-Хорезми, средневекового ученого, написавшего трактат о вычислениях с использованием десятичной системы счисления.

> Алгоритм - это точное описание порядка действий, которые должен выполнить исполнитель для решения задачи за конечное время.

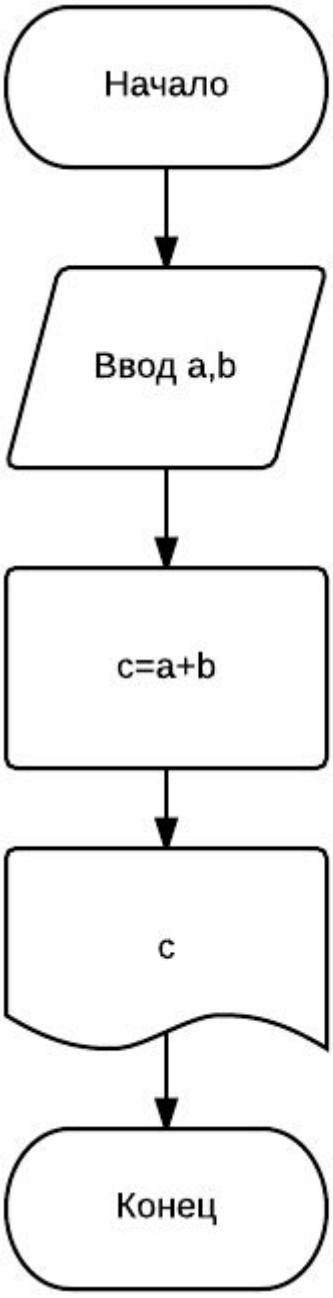
Алгоритм получает на вход некоторый дискретный входной объект (например, набор чисел или слово) и обрабатывает входной объект по шагам (дискретно), строя промежуточные дискретные объекты. Этот процесс может закончиться или не закончиться. Если процесс выполнения алгоритма заканчивается, то объект, полученный на последнем шаге работы, является результатом работы алгоритма при данном входе. Если процесс выполнения не заканчивается, говорят, что алгоритм зациклился. В этом случае результат его работы не определен.

Действительно, операции над десятичными числами, хоть мы об этом и не задумываемся, происходят по определенному, заранее заданному, порядку действий, и, если все действия произвести правильно, то это приведет к определенному результату.

Для описания алгоритма любой сложности необходимо и достаточно иметь всего три алгоритмические конструкции: следование (линейный алгоритм), ветвление и цикл.

Линейный алгоритм

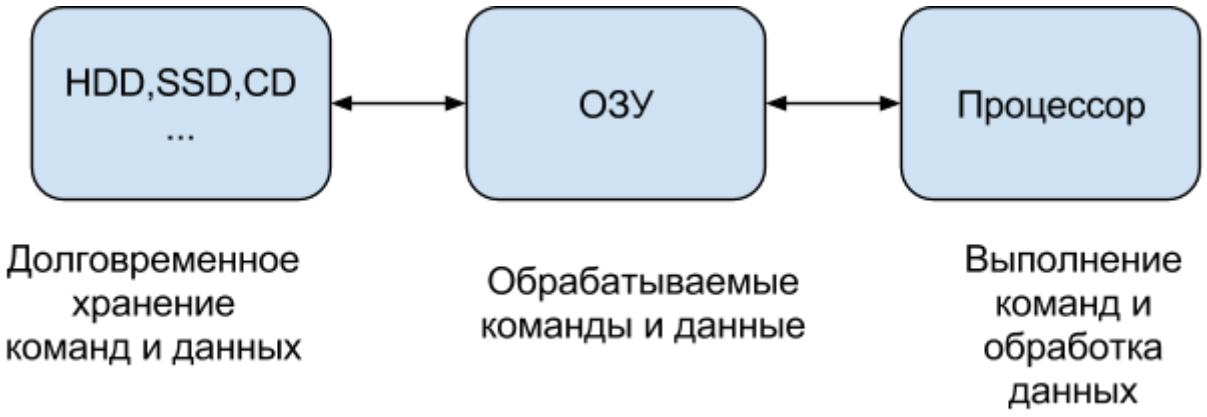
Линейный алгоритм описывает последовательность команд, выполняющихся строго друг за другом. Рассмотрим запись линейного алгоритма с помощью блок-схем.




```
<script>

var a=+prompt("a:");
var b=+prompt("b:");

var c=a+b;

alert(c);

</script>
```

Устройство компьютера

Современные компьютеры, если досконально разбираться в их устройстве, весьма сложны и чтобы понять, как он функционирует, может потребоваться прочитать не одну книгу. Но откроем вам страшную тайну: программисту, даже профессиональному, не обязательно очень подробно знать его устройство. Но общую схему функционирования программист, конечно, знать обязан. Она не сложная.



Необходимо познакомиться и понять взаимосвязь всего трех элементов компьютера.

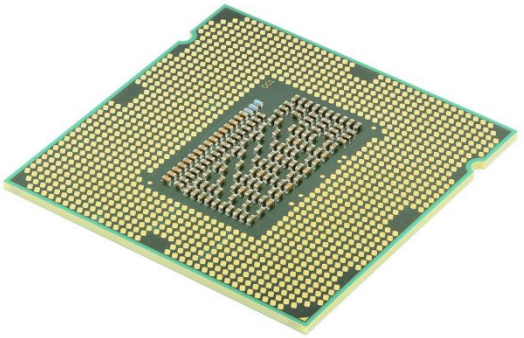
1. Процессор (CPU)
2. Оперативная память (ОЗУ,RAM)
3. Жесткий диск (HDD,SSD)

Процессор

Процессор - это устройство, которое обрабатывает данные и управляет остальными устройствами компьютера. Процессор обладает следующими характеристиками:

- Частота;
- Разрядность;
- Объем кэш-памяти;
- Многоядерность.

Чтобы понять, на что влияют эти характеристики, сравним процессор с мельницей, которая перемалывает зерно в муку. Тогда частота - это скорость вращения жернова, который перемалывает муку. Разрядность - это ширина мола. Чем больше частота и разрядность, тем больше муки можно перемолоть. Или, в нашем случае, данных. Тогда объем кэш-памяти - зерно, которое лежит рядом с валом, за которым не нужно бегать в амбар (в ОЗУ). Тогда многоядерность - это мельница с несколькими молами для перемалывания.

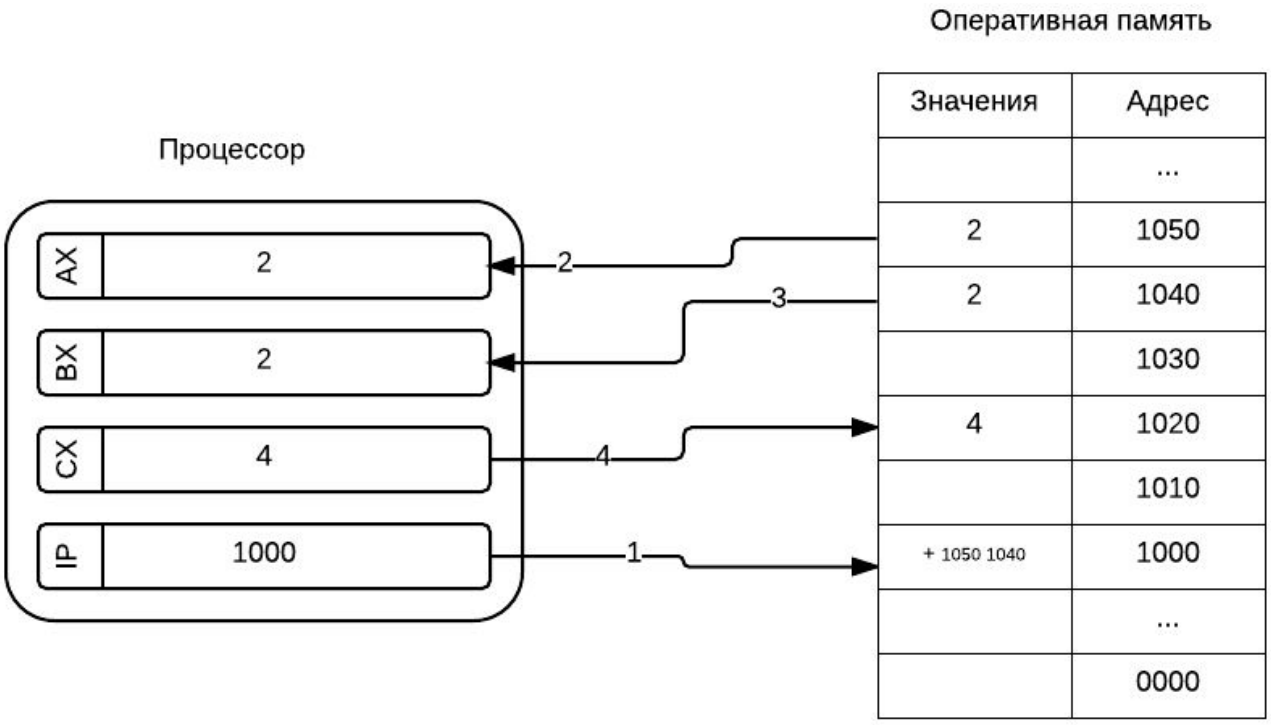


ОЗУ

В обиходе “оперативка” - это микросхемы, на которых хранятся данные и команды, предназначенные для обработки. Оперативная память так устроена, что работает намного быстрее (в сотни тысяч раз) жесткого диска, но для хранения данных в оперативной памяти требуется электричество. Если питание пропадет даже на доли секунды, данные в оперативной памяти стираются.

Процессор и ОЗУ

Данные и команды, которые обрабатываются в текущий момент, времени хранятся в ОЗУ. Из ОЗУ данные и команды загружаются в процессор, где происходит обработка данных в соответствии с командами. В процессоре присутствуют собственные ячейки памяти, называемые *регистрами*. Все вычисления производятся в регистрах процессора. Это самая быстрая часть компьютера.



В процессоре существует специальный регистр команд IP. Он хранит в себе адрес ячейк в которой записана команда, которую необходимо выполнить.

Жесткий диск

Жесткий диск, а также “флешки”, CD, DVD и другие относят к внешним запоминающим устройствам. Это связано с тем, что они играют второстепенную роль в обработки информации. То есть непосредственно в обработке информации эти устройства не участвуют, но при этом они выполняют другую важную роль. Их устройство позволяет им хранить информацию долговременно без источника питания.

Домашнее задание

1. **Конвертер валют.** Программа хранит в переменных курс доллара и евро. Пользователь вводит сумму в рублях и получает информацию о том, сколько эта сумма составляет в долларах и евро. В одной переменной у вас хранится стоимость одного евро в рублях, в другой стоимость одного доллара в рублях. Затем вы спрашиваете у пользователя сколько рублей он хочет сконвертировать, получаете это число и считаете. Результат выводите на страницу с помощью alert.
2. **Подсчет площади трапеции.** Пользователь вводит длину оснований трапеции (a и b), а также высоту трапеции h. Программа выводит сообщение: “Площадь трапеции будет равна <значение>”. Площадь вычисляется по формуле, указанной на картинке.

трапеция

$$S=\frac{a+b}{2}\cdot h$$

a, b - основания
h - высота