GeekBrains

Основы программирования

Начинаем программировать

Переменные. Типы данных. Устройство компьютера.

Оглавление

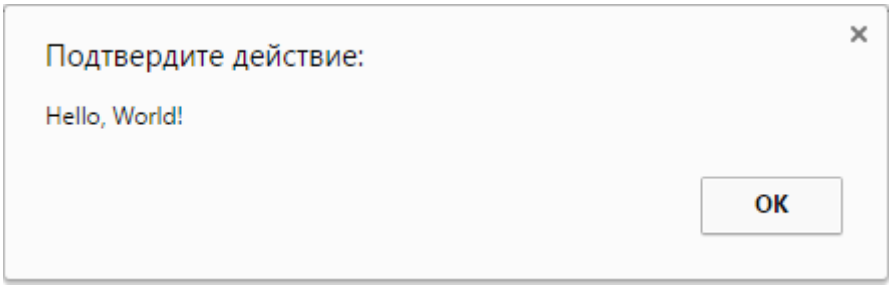
Первая программа

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 он открывался в браузере. Теперь запустите этот файл, щелкнув по нему мышкой два раза.

Если вы все сделали правильно, то запустится браузер и появится такое окно:



Оно может выглядеть по-разному в зависимости от версии браузера. На курсе мы, как правило, используем Chrome, хотя вы можете работать с тем, к которому привыкли.

Эта простая программа знакомит вас с 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 поговорим позже.

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.

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

Параметры команды — с их помощью мы сообщаем командам, как они должны работать. Описание команды **alert**:

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

message — это подсказка программисту, что **alert** может обработать текст сообщения, а **string** означает, что это сообщение должно быть строкой. Учтите, что это описание команды **alert**. Используя команду, в скобки мы просто передаем строку, не указывая тип. В конце строки можно поставить точку с запятой, но это не обязательно.

Числовой тип и автоматическое преобразование в строковой

Изменим программу следующим образом:

```
<script>

     alert(2016);

 </script>
```

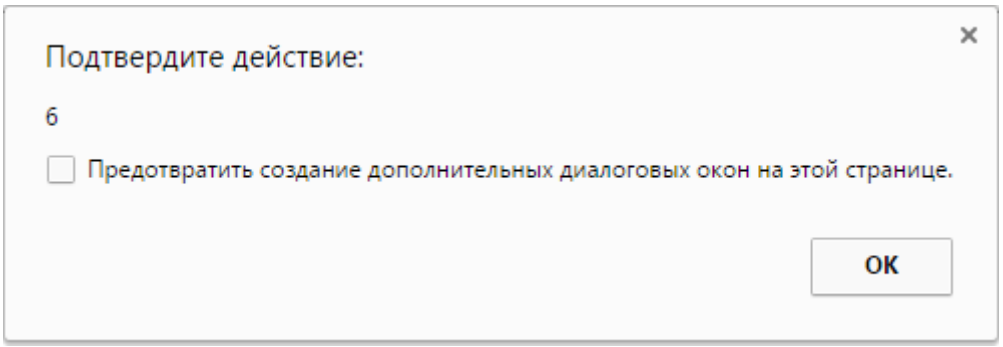
Она выведет на экран число 2016 — но оно не является строкой, а относится к числовому типу данных. Тогда почему **alert** вывел 2016 на экран, хотя в его описании указывается, что он выводит строковой тип данных? Дело в том, что JavaScript автоматически преобразовал 2016 в строковой тип. Чтобы в этом разобраться, нужно изучить, как компьютер хранит в памяти данные, и познакомиться с двоичным кодированием. Пока достаточно понять, что при необходимости JavaScript автоматически преобразовывает данные из одного типа в другой.

Вычисления

Изменим программу, поместив в скобки **alert** арифметическое выражение:

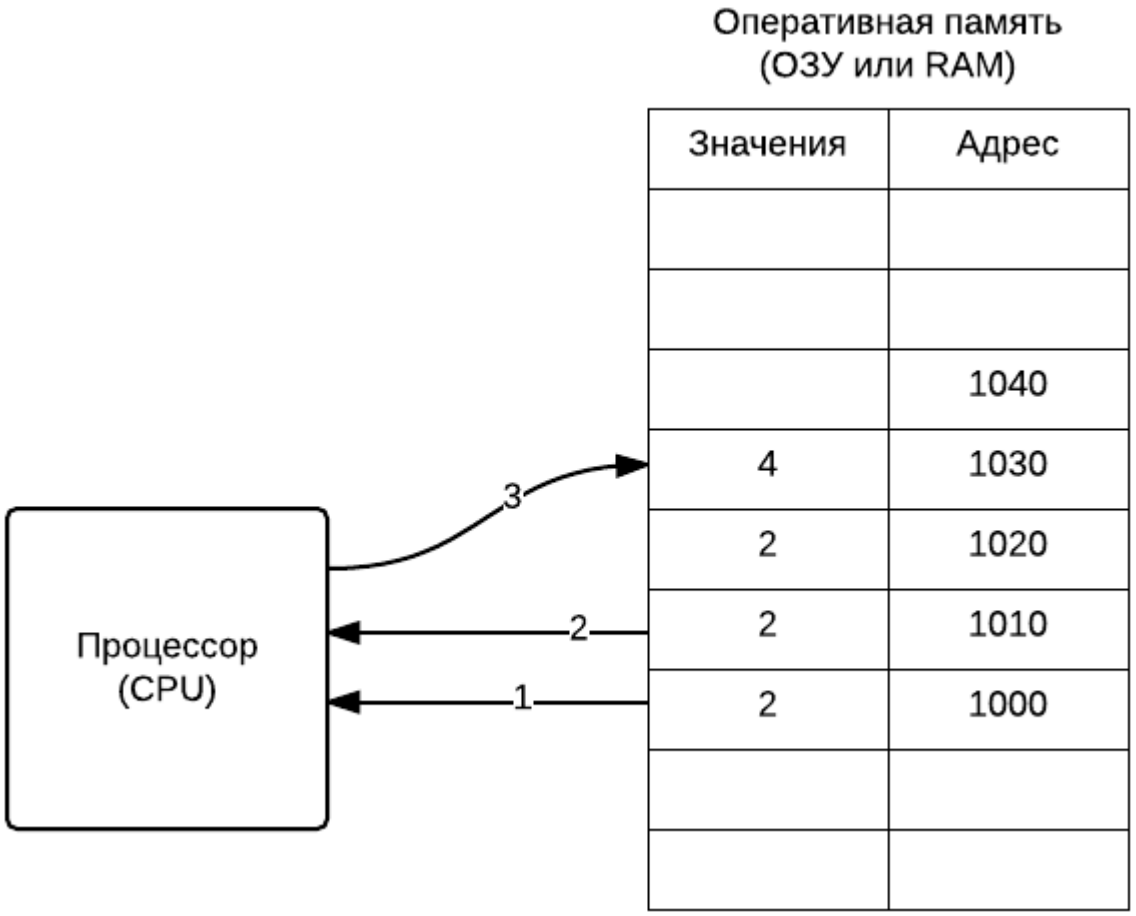
```
<script>

      alert(2 + 2 * 2);

 </script>
```

Сохраним и посмотрим результат в браузере:



Прежде чем вывести результат на экран, компьютер вычислил выражение в скобках, преобразовал полученное число в текст и выполнил команду **alert**. Машина понимает, какую операцию выполнять в первую очередь. Если нужно изменить приоритет операции, используйте скобки: **alert((2+2)*2)**.

Компьютер анализирует программу и сохраняет числа в оперативной памяти, каждое 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:

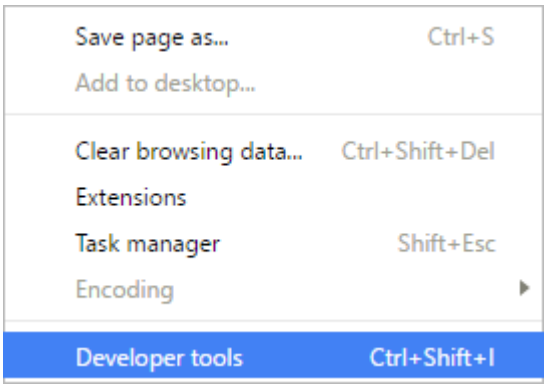


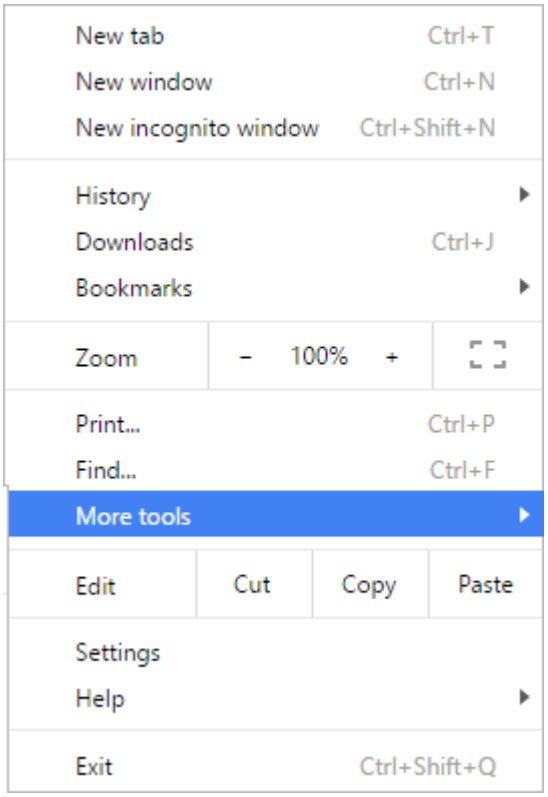
Отладка программы

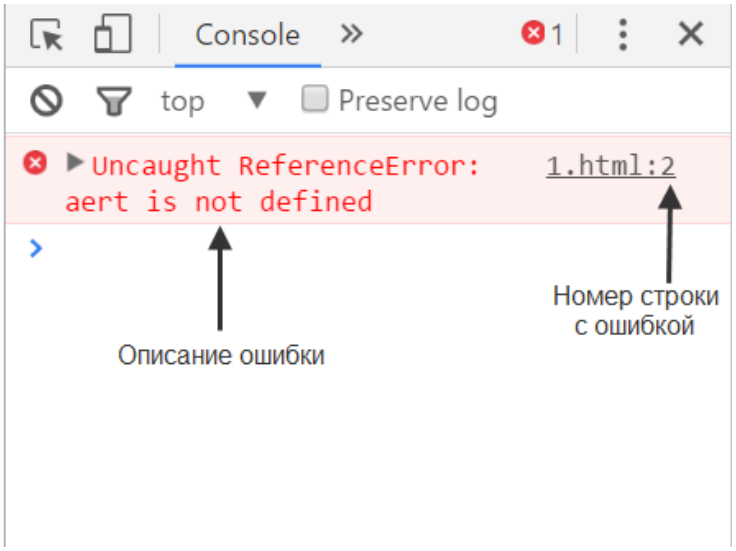
Удалите в предыдущей программе в команде **alert** одну букву и запустите:

```
<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 такие баги. В браузере Chrome выберите «Дополнительно» — «Инструменты разработчика» и щелкните на вкладке Console (или нажмите Ctrl+Shift+I и перейдите на вкладку Console).







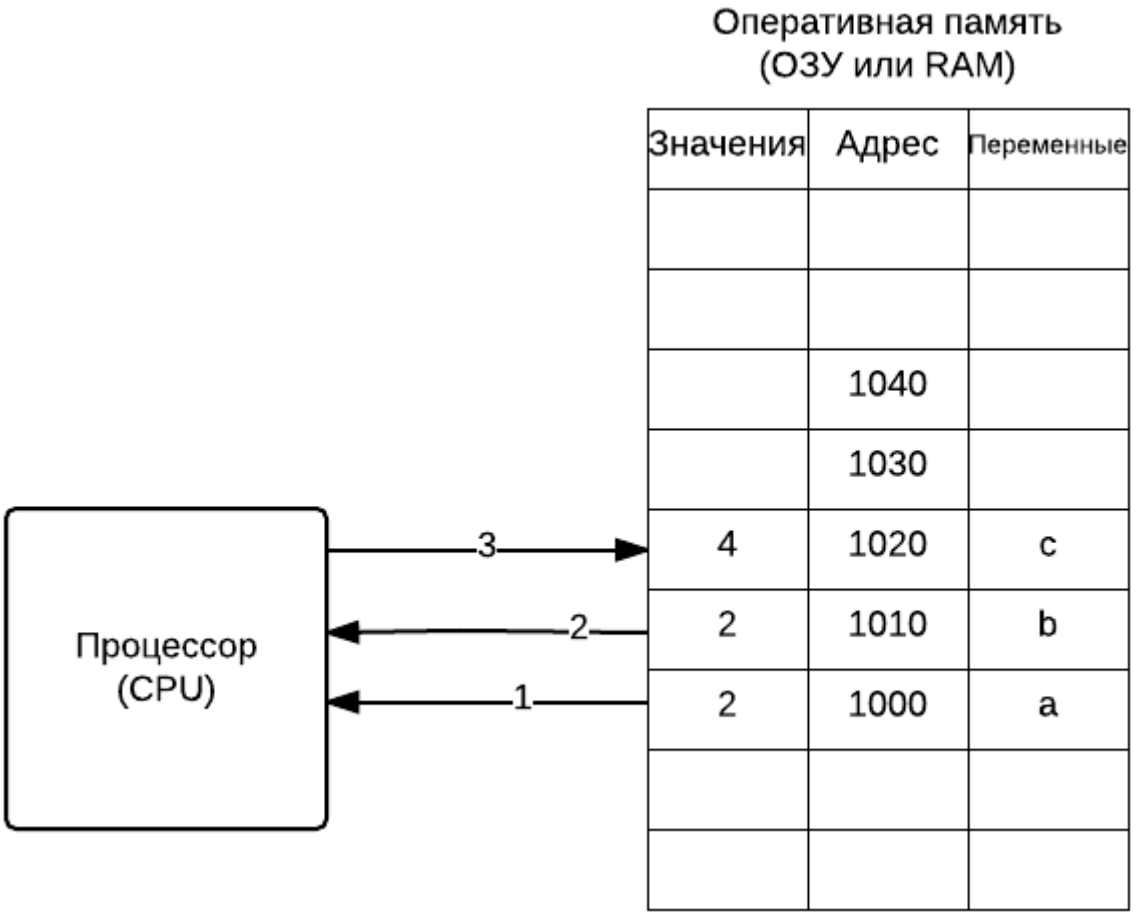
Если в записи программы есть ошибка, в консоли выводится сообщение: в какой строчке она допущена и ее описание. Последнее начинающим программистам, может, и не пригодится, но указание строки поможет присмотреться и найти неисправность. Исправьте ошибку, сохраните программу и перезагрузите страницу в браузере.

Переменные. Операция присваивания

Наберите и запустите следующую программу:

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

Ничего особенного не произошло: программа вывела на экран сумму двух чисел. Но теперь для их хранения используются переменные — они нужны, чтобы хранить изменяющиеся данные.



На самом деле переменные — это именованные адреса ячеек. То есть, обращаясь к переменной **a,** мы обращаемся к ячейке в памяти компьютера. 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, не используя переменные.

Чтобы изменить значение переменной, применяют операцию **присваивания**, которая в JavaScript обозначается знаком равенства (=). Несмотря на свой незамысловатый вид, это одна из самых

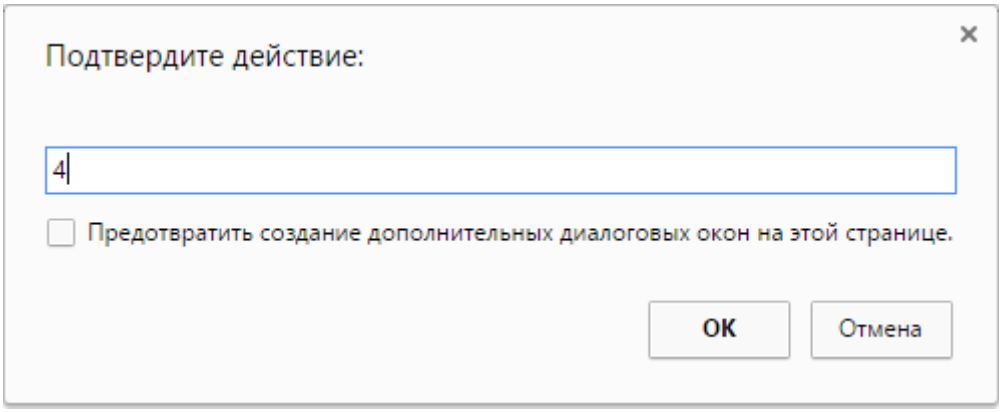
важных операций в программировании — поэтому важно понимать, что происходит, когда вы ее используете.

Ввод данных

Вводить данные в программу можно разными способами: с помощью клавиатуры, мыши, касания экрана, считывания данных из файла или из базы данных. Но в каждом языке есть команда, с которой начинают изучать ввод данных. В JavaScript это **prompt**. Наберите следующую программу и запустите ее на выполнение:

```
<script>

     var a;

     a = prompt();

     alert(a);

 </script>
```

Она выводит окно со строкой, куда пользователь может вводить данные.



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

Склеивание строк

Рассмотрим пример:

```
<script>

     var a;

     a = prompt();

     alert(a + " is a good number!");

 </script>
```

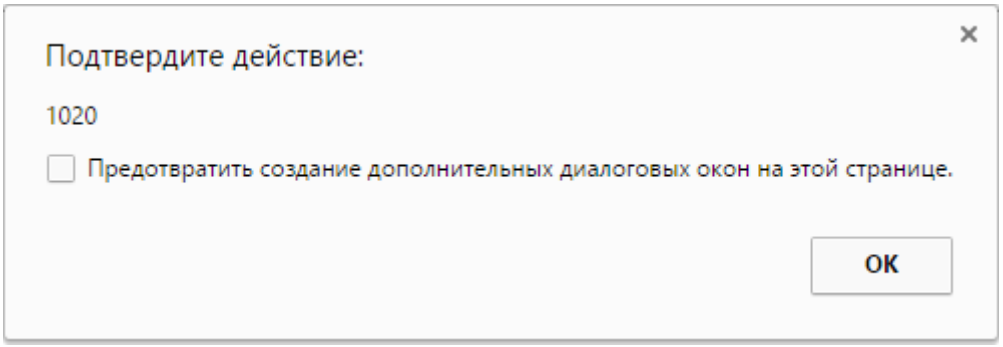
Здесь мы демонстрируем операцию склеивания строк. Для них знак **+** означает, что должна получиться новая строка, состоящая из нескольких строк, стоящих слева и справа от **+**.

Преобразование из строки в число

При необходимости JavaScript переводит число в строку. А если нужно сделать обратное? 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 **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 разберем подробнее, когда будем изучать функции.) Значит, компьютер принял от **prompt** строчку и поступил правильно, когда склеил две строки, вместо того чтобы сложить числа. 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>
```

При запуске она выведет **true** (истина). Знакомимся с еще одним типом данных — логическим. В переменную **a** поместился результат операции **отношения равенства** (==). Сначала посчиталась левая часть, операция 2 * 2, и сравнилась с правой — 4. Так как результаты равны, результат операции — «истина», и присвоился он в переменную **a**.

Логический тип данных может принимать всего два значения: «истина» и «ложь». В JavaScript используются специальные обозначения: **true** и **false**.

В JavaScript существуют следующие операции отношения:

- **==** — равенство;

- **<** — меньше;
- **>** — больше;
- **>=** — больше или равно;
- **<=** — меньше или равно;
- **!=** — не равно.

Алгоритмы

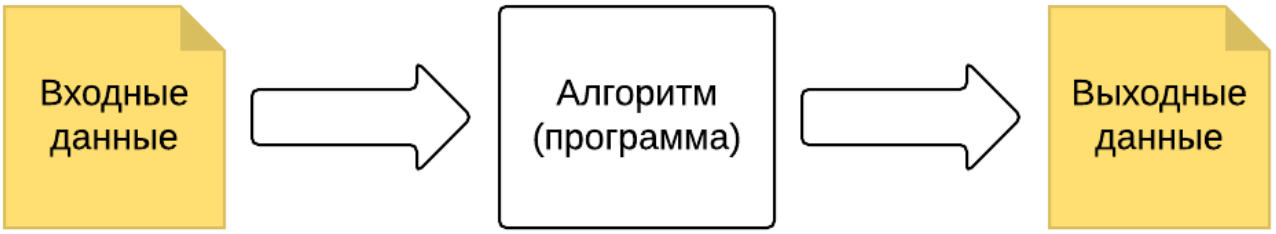
В сообществе программистов постоянно идут споры, нужны ли алгоритмы. Обычно все признают, что они важны, но вот стоит ли записывать их — или можно сразу писать программу? Многие высокомерно заявляют, что пишут программы, вообще не записывая алгоритмы на бумаге.

Не будем с ними спорить — но мы считаем, что если изучать язык без алгоритмов, то разработчик будет строить программы, используя конструкции первого языка. Вряд ли от этого получится избавиться и при изучении алгоритмов. Но их написание все же позволяет мыслить более широко, дает возможность сосредоточиться на решении задачи, а не ее реализации на конкретном языке.

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

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

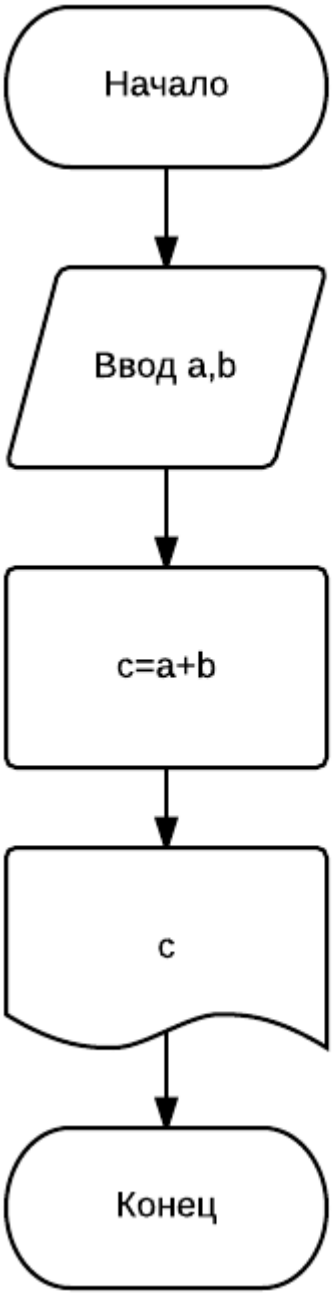
Действительно, операции над десятичными числами происходят по заданному порядку действий. И если все их произвести правильно, это приведет к результату.



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

Линейный алгоритм

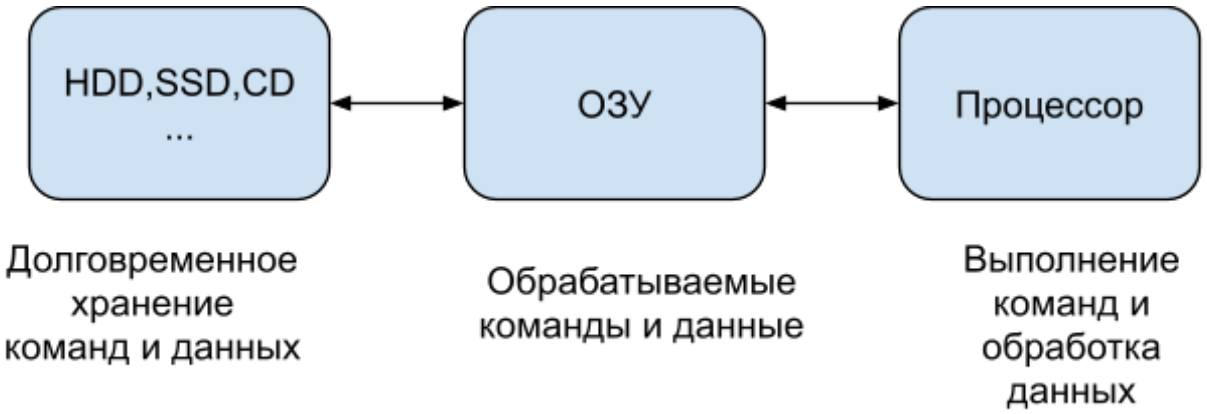
Линейный алгоритм описывает последовательность команд, выполняющихся строго друг за другом. Рассмотрим запись линейного алгоритма с помощью блок-схем:




```
<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, очень сложны. Чтобы понять, как они функционируют, надо прочитать не одну книгу. Но откроем страшную тайну: программисту, даже профессиональному, не обязательно подробно знать это. Достаточно общей схемы функционирования:



Необходимо понять взаимосвязь всего трех элементов компьютера:

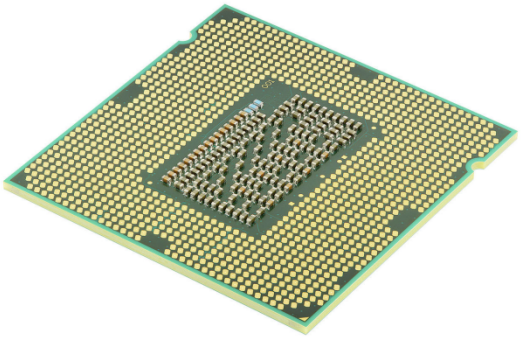
- процессора (CPU);
- оперативной памяти (ОЗУ, RAM);
- жесткого диска (HDD, SSD).

Процессор

Процессор обрабатывает данные и управляет остальными устройствами компьютера. Его характеристики:

- частота;
- разрядность;
- объем кеш-памяти;
- многоядерность.

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

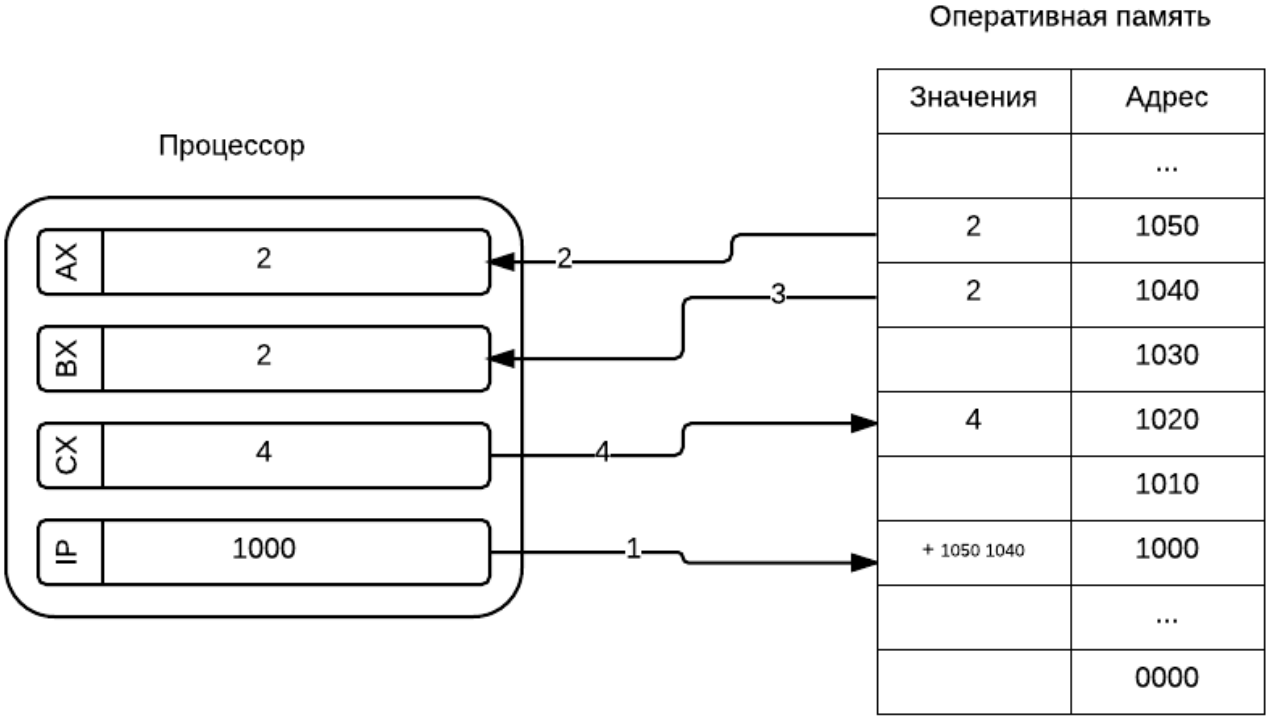


ОЗУ

В обиходе «оперативка» — это микросхемы, на которых хранятся данные и команды, предназначенные для обработки. Оперативная память так устроена, что работает в сотни тысяч раз быстрее жесткого диска, но для хранения данных в ней требуется электричество. Если питание пропадает даже на доли секунды, данные в оперативной памяти стираются.

Процессор и ОЗУ

Данные и команды, которые обрабатываются в текущий момент, хранятся в ОЗУ. Оттуда они загружаются в процессор, где происходит обработка данных в соответствии с командами. В процессоре присутствуют собственные ячейки памяти — *регистры*, в которых производятся все вычисления. Это самая быстрая часть компьютера.



В процессоре есть специальный регистр команд IP. Он хранит адрес ячейки — в ней записана команда, которую необходимо выполнить.

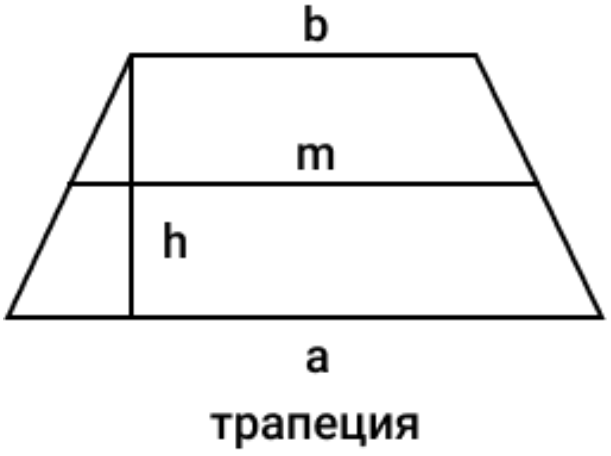
Жесткий диск

Жесткий диск, а также флешки, CD, DVD относят к внешним запоминающим устройствам — они играют второстепенную роль в обработке информации. При этом могут хранить ее долговременно без источника питания.

Практическое задание

1. **Конвертер валют.** Программа хранит в переменных курс доллара и евро. Пользователь вводит сумму в рублях и получает информацию о том, сколько она составляет в других валютах. В одной переменной у вас хранится стоимость одного евро в рублях, в другой — одного доллара. 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 ее высоту **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