Основы программирования

Блок-схемы и ветвления

[В какой версии ПО был написан код. Если актуально]

# На этом уроке

1. Узнаем как можно визуализировать алгоритмы с помощью блок-схем;
2. Узнаем что такое операторы сравнения;
3. Научимся использовать ветвление в программах.

# Введение

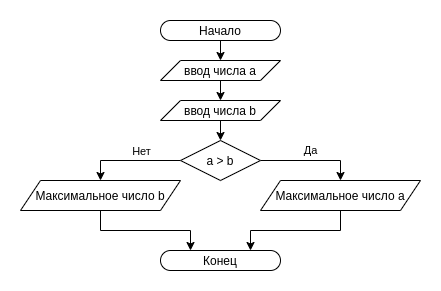
На прошлом уроке мы узнали самые базовые вещи в программировании: как получать данные от пользователя, как эти данные хранить в переменных и изменять. Чтобы двигаться дальше, нам нужно научиться писать программу, которая способна по-разному реагировать на поступающие данные. Точно так же, как мы по-разному реагируем на обстоятельства, например не переходим дорогу на красный свет, а ждём зелёного, программе тоже важно «уметь» принимать решения в зависимости от обстоятельств. Слово «уметь» взято в кавычки потому, что в действительности программа лишена свободы воли, она работает по тем правилам, которые придумал программист.

Мы подошли к важному вопросу: как задавать правила выполнения алгоритма, как в программе делать выбор?

# Ветвление

Ветвление — это конструкция языка программирования, обеспечивающая выполнение определённой команды (набора команд) только при условии истинности некоторого логического выражения.

Для того, чтобы визуализировать какой-то алгоритм, принято использовать блок-схемы. Давайте построим блок-схему алгоритма, отвечающего на вопрос «какое из двух чисел, введённых пользователем, больше?»:



Как можно видеть, блок-схема наглядно и доходчиво описывает логику программы. Давайте теперь напишем код по этой блок-схеме:

|  |
| --- |
| let a = prompt('Введите число a'); let b = prompt('Введите число b');  if (a > b) {  console.log('Максимальное число: a'); } else {  console.log('Максимальное число: b');  } |

В этом коде появились новые операторы и логические выражения:

* **if** переводится с английского как «если», это условный оператор;
* Смысл выражения **a > b** очевиден — это проверка на то, является ли значение переменной a больше значения переменной b;
* **else** переводится как «иначе», определяет последнюю ветвь условного оператора.

Этот код

|  |
| --- |
| if (a > b) {  console.log('Максимальное число: a'); } else {  console.log('Максимальное число: b');  } |

На русском можно было бы записать как:

|  |
| --- |
| если (a > b) {  вывести\_текст('Максимальное число: a'); } иначе {  вывести\_текст('Максимальное число: b'); } |

***Обратите внимание, что код, который должен выполняться при соблюдении определённого условия, помещается в блок, который выделяется фигурными скобками { }.***

Наверняка у вас возникло желание где-то помечать комментариями трудные для понимания места в коде, чтобы не забыть что это значит.

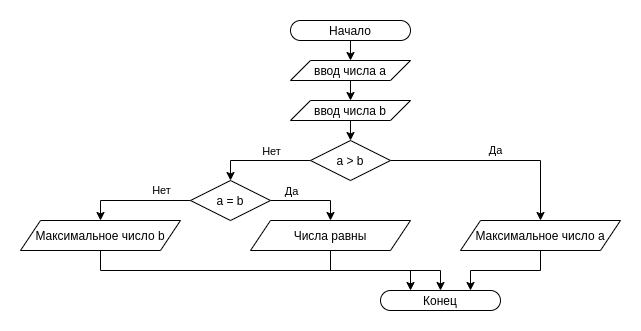
Сопроводим наш код комментариями:

|  |
| --- |
| /\*  Эта программа определяет какое из двух  введённых пользователем чисел больше \*/ let a = prompt('Введите число a'); // Пользователь вводит первое число let b = prompt('Введите число b'); // Пользователь вводит второе число  if (a > b) { // Если первое число больше второго  console.log('Максимальное число: a'); } else { // В противном случае  console.log('Максимальное число: b');  } |

Теперь работать с кодом стало проще и нам и тем, кто будет его читать. Конечно, такое подробное комментирование в большинстве случаев не требуется, но поскольку сейчас мы только учимся программированию, комментарии лишними не будут. Заметьте, что есть два типа комментариев:

* Однострочные, которые начинаются с двух символов «слеш»: //
* Многострочные, которые начинаются с символов /\* и заканчиваются символами \*/

Давайте разберём одну проблему в составленном нами алгоритме: что будет, если число a окажется равным числу b? Условие a > b не выполняется, значит будет выполнен код в блоке else, то есть, пользователю будет сказано, что максимальным является число b, что не совсем корректно. Нам нужно модифицировать алгоритм таким образом, чтобы в случае равенства чисел пользователю выводилось сообщение «Числа равны». Давайте начнём с блок-схемы:



Ветвление стало сложнее потому, что добавилось ещё одно условие. Теперь, после того, как мы проверяем a > b, в случае, если a не больше b, нужно дополнительно проверять, не равны ли a и b. Дополним код программы:

|  |
| --- |
| let a = prompt('Введите число a'); let b = prompt('Введите число b');  if (a > b) { // Если первое число больше второго  console.log('Максимальное число: a'); } else { // В противном случае  if (a == b) { // Если первое число равно второму  console.log('Числа равны');   } else { // В противном случае  console.log('Максимальное число: b');   } } |

В полном соответствии с блок-схемой появился вложенный оператор if. Сама по себе вложенность (если не является слишком многоуровневой, что мешает восприятию кода) не является плохим приёмом, но в большинстве языков программирования есть специальный оператор, позволяющий избавится от ненужной вложенности:

|  |
| --- |
| let a = prompt('Введите число a'); let b = prompt('Введите число b');  if (a > b) { // Если первое число больше второго  console.log('Максимальное число: a'); } else if (a == b) { // В другом случае, если первое число равно второму  console.log('Числа равны');  } else { // Во всех остальных случаях  console.log('Максимальное число: b');  } |

Правила, которые нужно помнить при использовании операторов **if**, **else if** и **else**:

* ветвление должно всегда начинаться с **if**;
* **else if** требует обязательного указания условия, так же как и **if**;
* каждому оператору **if** может соответствовать несколько операторов **else if**;
* **else** — последний блок ветвления;
* каждому оператору **if** может соответствовать только один оператор **else**;
* **else if** и **else** — необязательные блоки в ветвлении.

Обратите внимание на то, как происходит сравнение a и b на равенство: используется оператор ==. Дело в том, что в программировании оператор присваивания (=) и оператор равенства (==) — это два разных оператора, использующихся для разных задач.

# Операторы сравнения

В JavaScript, операторы сравнения записываются так:

* Больше-меньше: a > b, a < b;
* Больше или равно, меньше или равно: a >= b, a <= b;
* Равно: a == b;
* Не равно: a != b.

Результат сравнения — это всегда булево значение (логический тип, о котором мы говорили в прошлом уроке), которое можно назначить переменной, поэтому код

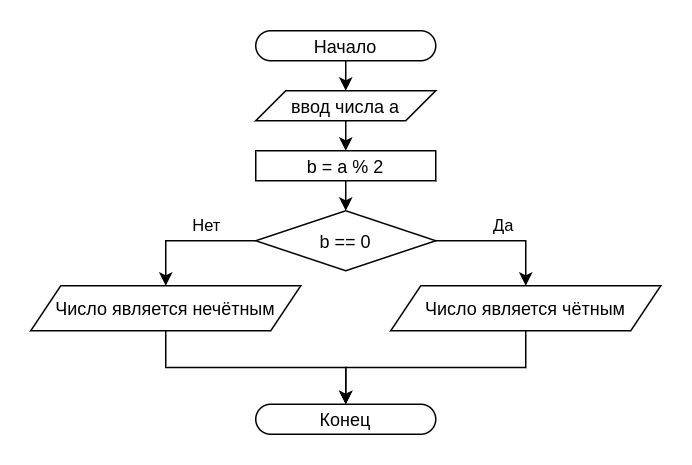
|  |
| --- |
| if (a > b) {  console.log('Максимальное число: a'); } |

Можно записать так:

|  |
| --- |
| let result = a > b; // true, если a > b и false в противном случае if (result) {  console.log('Максимальное число: a'); } |

# Определение чётности числа

Применим полученные знания для решения следующей задачи: пользователь вводит число, а мы выводим результат: является ли число чётным или нет.



Если посмотреть на блок вычисления числа b, можно заметить новый оператор %. Это **оператор вычисления остатка от деления** и работает он следующим образом:

|  |
| --- |
| console.log(3 % 2); // 1 console.log(10 % 2); // 0 console.log(10 % 15); // 10 |

Теперь можно перейти к реализации алгоритма, отражённого на блок-схеме:

|  |
| --- |
| let a = prompt('Укажите число'); let b = a % 2; // Чётное число - это число, которое делится на 2 без остатка  if (b == 0) {  console.log('Число является чётным'); } else {  console.log('Число является нечётным'); } |

Кстати, получившийся код можно упростить:

|  |
| --- |
| let a = prompt('Укажите число');  if (a % 2 == 0) {  console.log('Число является чётным'); } else {  console.log('Число является нечётным'); } |

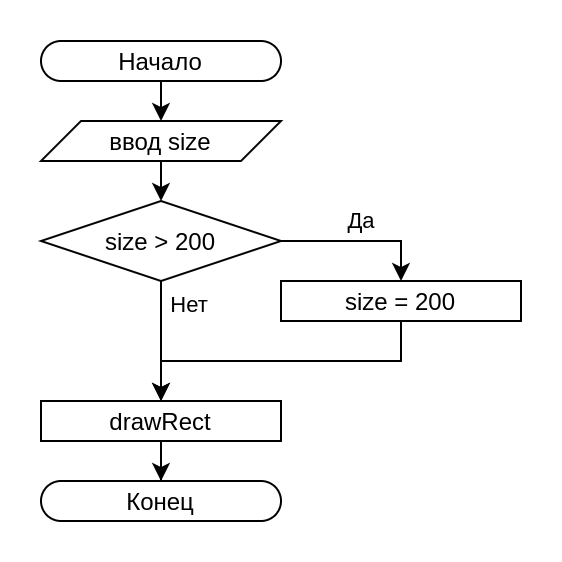
Мы избавились от ненужной переменной b и поместили выражение a % 2 слева от оператора сравнения.

# Рисование квадрата с настраиваемым размером

Применим полученные знания к решению задачи, связанной с графикой. Решим следующую задачу: пользователь указывает размер квадрата, после чего необходимо нарисовать квадрат заданного размера. Если пользователь ввёл число больше 200, нужно нарисовать квадрат размером 200 (ограничим максимальный размер).

|  |
| --- |
| let size = prompt('Укажите размер квадрата');  if (size > 200) {  size = 200;  }  drawRect(10, 10, size, size, 'black'); |

Примечательно, что **if** здесь используется без **else**, так тоже можно делать, когда в else нет необходимости, на блок-схеме это выглядело бы так:



Из нового здесь только функция drawRect, которая может нарисовать не только квадрат, но и прямоугольник. Эта функция принимает следующие аргументы:

1. Позиция прямоугольника слева;
2. Позиция прямоугольника сверху;
3. Ширина прямоугольника;
4. Высота прямоугольника.
5. Цвет прямоугольника.

***Не забывайте, что функция drawRect не является стандартной для JavaScript, мы добавили её в тренажёр специально для иллюстрации того, как программирование позволяет работать с графикой.***

# Домашнее задание

Необходимо написать программу, которая запрашивает у пользователя год (например, 974, 1988, 2020) и выводит ответ на вопрос является ли год високосным или нет. Алгоритм определения того, является ли год високосным следующий:

1. Если год делится на 4 без остатка, перейдите на шаг 2. В противном случае год **не високосный**.
2. Если год делится на 100 без остатка, перейдите на шаг 3. В противном случае год **високосный**.
3. Если год делится на 400 без остатка, то он **високосный**, если нет, то **не високосный**.

let year = prompt('Введите год'); //пользователь вводит год

if (year % 4 > 0) { //если год делится на 4 с остатком, то он не високосный

console.log('Год не является високосным');

} else if (year % 100 > 0) { //если год делится на 100 с остатком, то он високосный

console.log('Год является високосным');

} else if (year % 400 == 0) { //если год делится на 400 без остатка, то он високосный

console.log('Год является високосным');

} else { //в противном случае год не високосный

console.log('Год не является високосным');

}

## Дополнительно

Написать программу, которая запрашивает у пользователя какую фигуру он хочет нарисовать (можно использовать ответ пользователя 1 - «круг», 2 - «квадрат»), в зависимости от ответа запрашивает радиус или ширину и рисует соответствующую фигуру. Расположение фигуры можно оставить на своё усмотрение, либо тоже запросить у пользователя.

let figure = prompt('Какую фигуры вы хотите нарисовать? (1 - круг, 2 - квадрат)');

if (figure == 1) {

let radius = prompt('Введите радиус');

drawCircle(200, 150, radius, 'black');

} else {

let width = prompt('Введите ширину');

drawRect(10, 10, width, width, 'black');

}

# Глоссарий

* Ветвление — конструкция языка программирования, обеспечивающая выполнение определённой команды (набора команд) только при условии истинности некоторого логического выражения, либо выполнение одной из нескольких команд (наборов команд) в зависимости от значения некоторого выражения;
* Логическое выражение — конструкция языка программирования, результатом вычисления которой является «истина» или «ложь»;
* Логическое равенство — это логическое выражение, которое является истинным тогда, когда оба простых логических выражения имеют одинаковую истинность;
* Чётность числа — характеристика целого числа, определяющая его способность делиться нацело на два;
* Прямоугольник — четырёхугольник, у которого все углы прямые (равны 90 градусам).
* Квадрат — четырёхугольник, у которого все углы равны и все стороны равны;
* Логический тип — примитивный тип данных в информатике, принимающий два возможных значения, иногда называемых истиной (true) и ложью (false).

# Дополнительные материалы

* Методический материал школьного урока по делению с остатком — <https://math-prosto.ru/?page=pages/division_with_remainder/division_with_remainder.php>;

# Используемые источники

* <https://wikipedia.org>