Урок 2

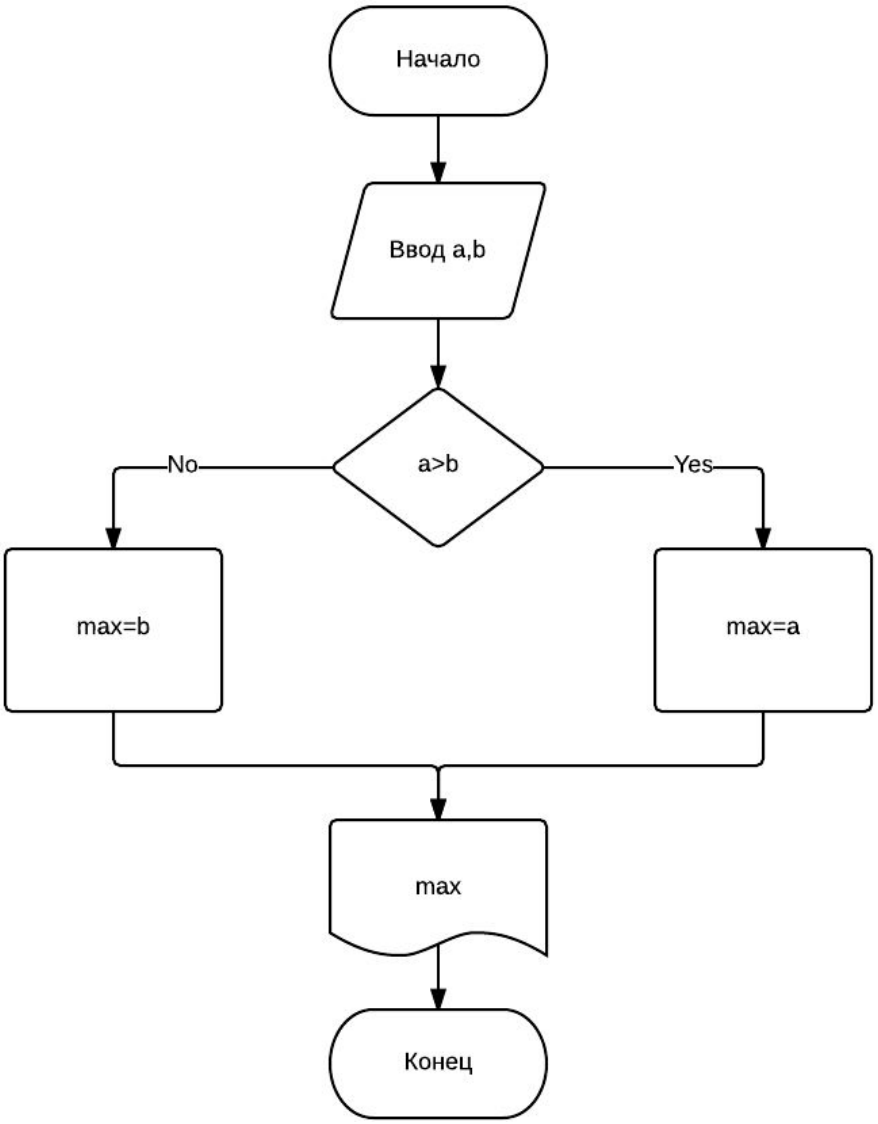
Ветвления. Циклы.

Ветвления. Циклы.

Разветвляющиеся алгоритмы

Решим задачу нахождения наибольшего из двух чисел.

Идея решения. Вводим два числа. Сравниваем числа, если первое число больше второго, то запоминаем первое число, иначе, запоминаем второе число.



Ветвление в JavaScript реализовано условным оператором

```
if (condition)
{
    //что делать, если условие верно
} else
```

```
{
    //что делать, если условие неверно
}
```

где condition – это какое то выражение равное true или false.

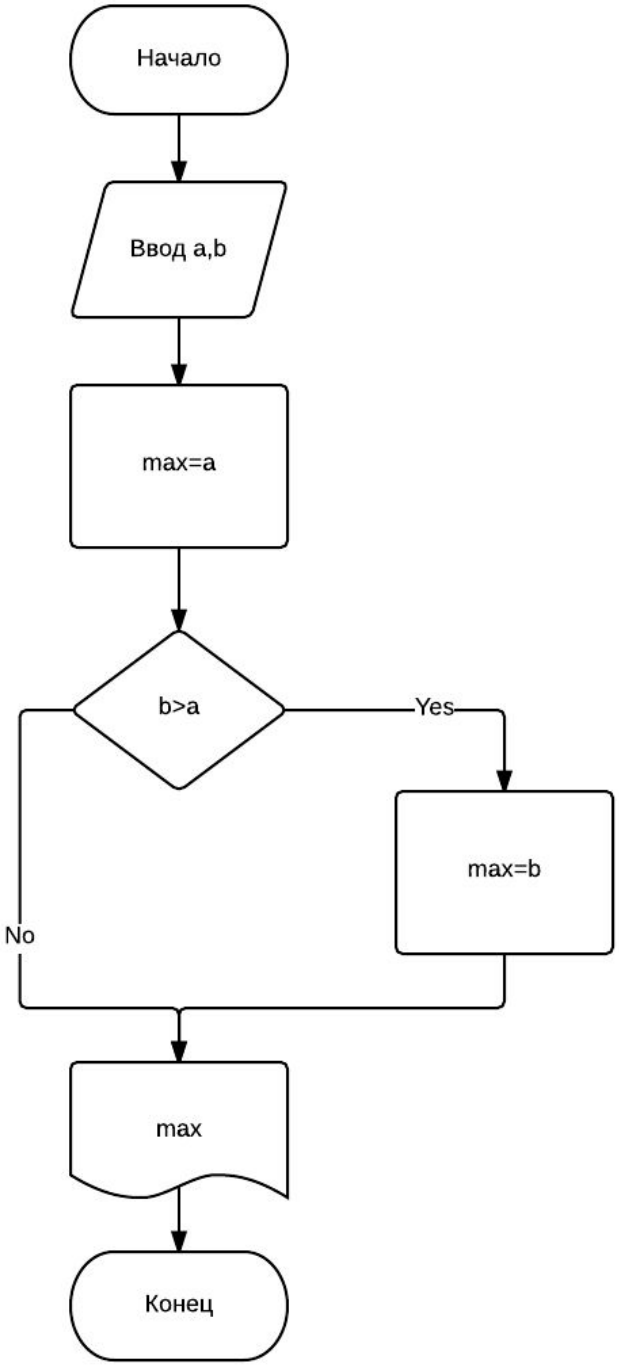
Особенности:

- вторая часть (else) может отсутствовать (неполная форма условного оператора);
- если в блоке один оператор, можно скобки { и } не ставить.

Реализация алгоритма на JavaScript

```
<script>
    var a = +prompt("a:");
    var b = +prompt("b:");
    var max;
    if (a > b) {
        max = a;
    } else {
        max = b;
    }
    alert(max);
</script>
```

Решим задачу нахождения наибольшего числа, используя неполную форму условного оператора



Реализация на JavaScript

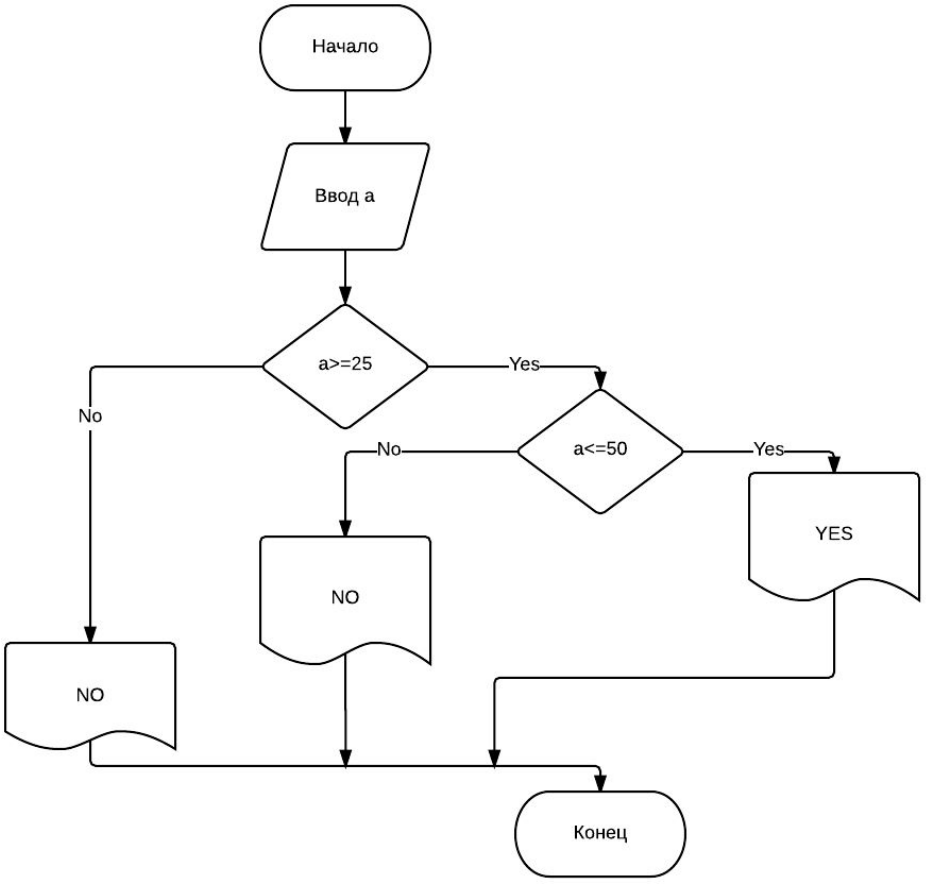
```
<script>
    var a = +prompt("a:");
    var b = +prompt("b:");
    var max = a;
    if (b > a)
        max = b;
    alert(max);
</script>
```

Вложенные условия и программирование “лесенкой”

Задача. Фирма набирает сотрудников от 25 до 50 лет включительно. Ввести возраст человека и определить, подходит ли он фирме (вывести ответ «YES» или «NO»).

Особенность: надо проверить, выполняются ли два условия одновременно.

Можно ли решить задачу уже известными методами? Да, просто нужно будет вложить одно условие в другое.




```
<script>


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



if (a>=25)
{

    if (a<=50)
    {
      alert("YES");
    }
    else
    {
       alert("NO");
    }
}
else
{
 alert("NO");
}

</script>
```

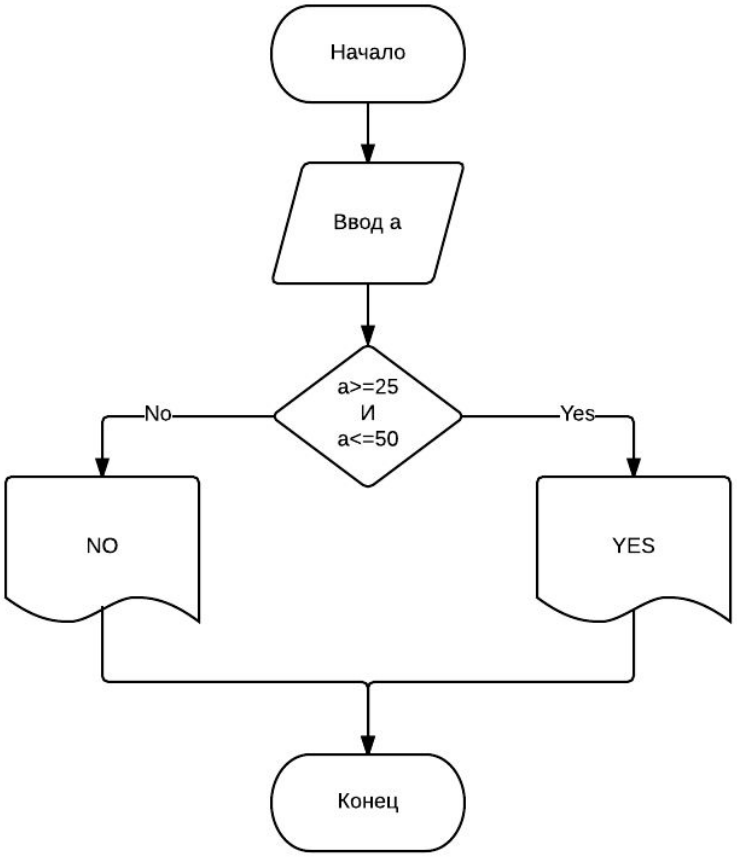
Эта конструкция называется “вложенное условие”. Вложенные условия часто используются программистами, когда одним условием не обойтись. Здесь условие a<=50 будет выполняться, только если истинно условие a>=25. Обратите внимание, как написан код программы - “лесенкой”. Мы и раньше использовали такой стиль, немного отодвигая некоторые строчки правее от края. Но это первая программа, которая демонстрирует для чего программисты используют такой стиль. “Лесенка”

помогает увидеть вложенность блоков кода друг в друга. Чем сильнее отступ тем больше вложенность.

Благодаря “лесенке” код становится более читабельным. Поначалу вам может быть не привычно писать, используя такой стиль, но, чем более сложные программы вам придется писать, тем более очевидным будет использование такого стиля.

Сложные условия

Согласитесь, что было бы проще предыдущую задачу решить, если бы у нас была возможность поместить оба условия в одно, как на блок-схеме:



Здесь условие состоит из двух операций сравнения, объединённых логической операцией **И**. В языках программирования условия, которые состоят из нескольких отношений, объединенных логическими операциями, называются *сложными условиями*.

Рассмотрим логические операции в порядке приоритетов выполнения:

- **!** - Отрицание - если отношение истинно, то отрицание делает его ложным и наоборот;
- **&&** - И - сложное условие истинно, когда оба отношения истины;
- **||** - ИЛИ - условие истинно, когда истинно хотя бы одно из условий;

Таблица истинности

A	B	A && B	A || B	!A
false	false	false	false	true

false	true	false	true	true
true	false	false	true	false
true	true	true	true	false

Тогда программа, использующая сложное условие, будет выглядеть так:

```
<script>
    var a = +prompt("a:");
    if (a >= 25 && a <= 50) {
        alert("YES");
    } else {
        alert("NO");
    }
</script>
```

Сравнение строк

Часто нужно получить от пользователя ответ на вопрос. В примере показано, как можно сравнить строковые значения.

```
<script>
    var answer = prompt("Вы уверены?");
    if (answer == "Да" || answer == "да" || answer == "ДА") {
        alert("Хорошо, сделайте это!");
    } else {
        alert("Возможно позже...");
    }
</script>
```

Циклы

Вывод без OK

Рассмотрим простую программу:

```
<script>
    document.write("Hello!");
</script>
```

Запустите и убедитесь, что в окне браузера появилась надпись Hello!

Теперь познакомимся с новым способом вывода информации на экран и заодно начнем знакомство (весьма беглое) с его величеством Объектно-Ориентированным Программированием. До этого момента большинство из вас использовала команду alert для вывода информации на экран. Преимущества этой команды в её простоте, но главный недостаток в том, что нужно обязательно

щелкнуть на OK, чтобы продолжить выполнение скрипта. Давайте познакомимся с ещё одной командой вывода на экран, которая лишена этого недостатка. Это команда **document.write**

Ее синтаксис следующий:

```
function(html:string)
```

Это означает, что это команда принимает на вход строку и не возвращает значение. Слово html - это подсказка программисту, что это команда может принимать HTML-код, который будет передан браузеру на обработку.

Эту команду нужно писать, указывая в начале слово document, что означает, что команда write находится в объекте document. На самом деле и команда alert была в объекте window, и мы могли бы писать window.alert(), но window можно опускать при написании, а document нет.

Рассмотрим, что означает подсказка html. Наберите программу:

```
<script>
    document.write("<strong>Hello!</strong><br>");
    document.write("I am <em>HTML!</em>");
</script>
```

Здесь мы использовали теги HTML, и браузер их обработал. сделал “Hello!” более жирным, а HTML наклонным.
 - это переход на следующую строку.

Цикл while

Решим задачу. Вывести на экран числа от 1 до 5. Вот ее решение:

```
<script>
    document.write("1<br>");
    document.write("2<br>");
    document.write("3<br>");
    document.write("4<br>");
    document.write("5<br>");
</script>
```

Вот вывод:

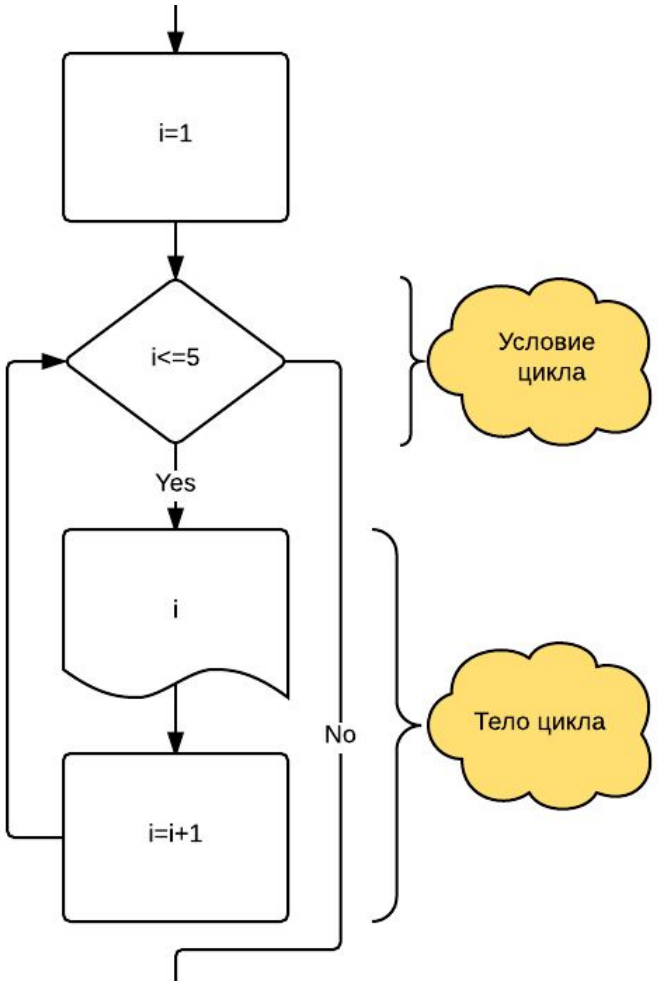
1
2
3
4
5

Мы добились результата, но как быть, если нам нужно вывести 100 чисел? Воспользуемся циклом.

```
<script>
    var i = 1;
    while (i <= 5) {
        document.write(i + "<br>");
```

```
        i = i + 1;
    }
</script>
```

Эту программу легко переделать, чтобы она выводила хоть 100, хоть 1 000 000 чисел. Для подробного разбора цикла запишем алгоритм программы в виде блок-схемы.



Этот цикл будет выполняться, пока условие цикла истинно. Условие стоит перед телом цикла. Отсюда его название - *цикл с предусловием*

Формат записи цикла while

```
while (условие)
{
    //тело цикла
}
```

Особенности:

- Условие может быть сложным;
- Операторные скобки могут отсутствовать, если тело цикла состоит из одного оператора

Подсчет суммы арифметической прогрессии

Решим задачу: Подсчитать сумму чисел от 10 до N с шагом 2 и вывести получившуюся сумму на экран.

```
<script>
    var start = 10;       // Начало отсчета(включительно)
    var n = 16;           // до какого числа считаем
    var step = 2;         // Шаг
    var sum = 0;          // здесь будем хранить сумму всех чисел от start до n
    var next = start;     // Следующее число

    while (next <= n) {
        sum = sum + next;
        next = next + step;
    }
    document.write("Начало отсчета:" + start + "<br>");
    document.write("Число, до которого считаем:" + n + "<br>");
    document.write("Шаг:" + step + "<br>");
    document.write("Итоговая сумма:" + sum + "<br>");
</script>
```

Цикл for

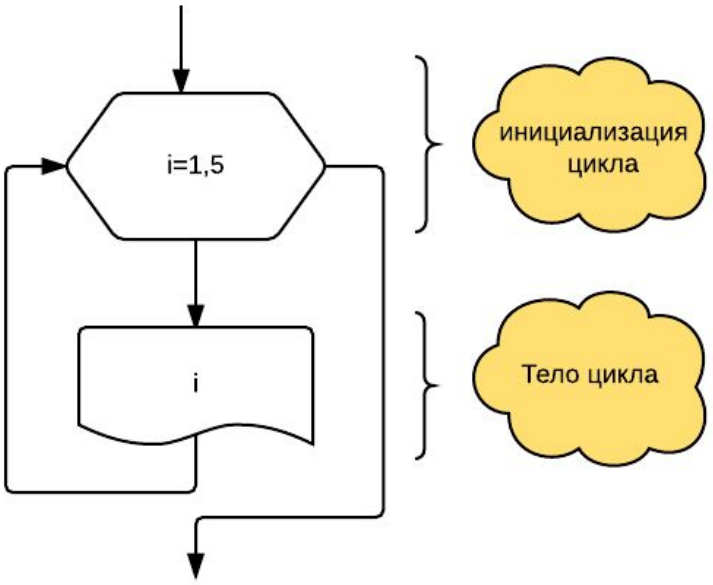
Программистам весьма часто приходится решать задачи, в которых что-то подсчитывается, а для подсчета используется специальная переменная - счётчик. Для таких задач удобно использовать специальный цикл - цикл со счётчиком

Рассмотрим программу:

```
<script>
    for (var i = 1; i <= 5; i++) {
        document.write(i + "<br>");
    }
</script>
```

Здесь мы выводим числа от 1 до 5 с использованием цикла for. В этом цикле условие инициализации начальным значением и изменения счетчика описывается непосредственно в заголовке цикла. Для решения многих задач такое описание может быть более удобным, поэтому этот цикл довольно часто заменяет цикл while.

Блок-схема цикла for



Это цикл использует переменную счетчик для подсчета количества итераций, поэтому этот цикл еще называют *цикл со счетчиком или цикл с параметром.*

В качестве упражнения, можете попробовать самостоятельно переписать программу подсчета арифметической прогрессии с использованием цикла for или посмотреть на пример ниже:

```
<script>
    var n = 5;          // до какого числа считаем
    var sum = 0;        // Сумма
    for (var i = 1; i <= n; i++) {
        sum = sum + i;
    }
    document.write("Арифметическая прогрессия до числа: " + n + "<br>");
    document.write("Сумма всех чисел от 0 до " + n + " = " + sum + "<br>");
</script>
```

Формат записи цикла for

```
for (переменная = начальное значение; условие; изменение переменной)
{
    //тело цикла
}
```

Особенности:

- Условие может быть сложным;
- Операторные скобки могут отсутствовать, если тело цикла состоит из одного оператора.

Цикл do while

Предположим нам нужно попросить пользователя ввести число, но при этом ограничить ввод только положительными значениями. Программа с использованием цикла while будет выглядеть вот так:

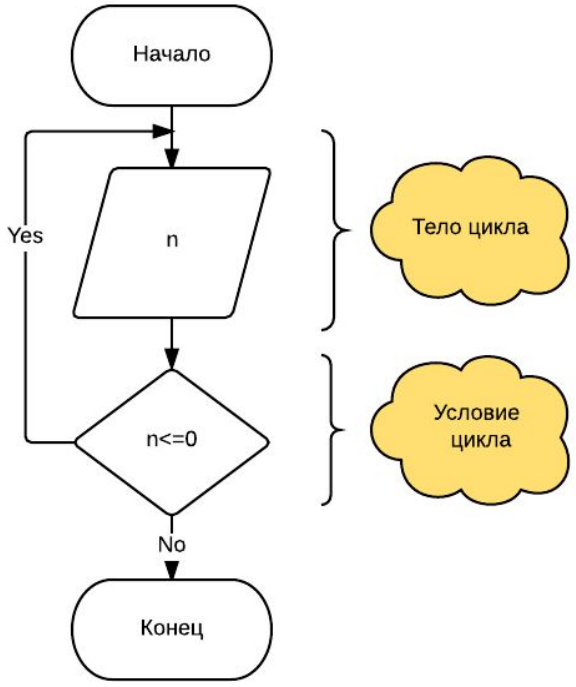
```
<script>
    var number;
    while ((number = +prompt("Введите пожалуйста число больше нуля:")) <= 0) {

    }
</script>
```

Также это можно решить с помощью цикла do while:

```
<script>
    var number;
    do {
        number = +prompt("Введите пожалуйста число больше нуля:");
    } while (number <= 0);
</script>
```

Блок-схема программы



Этот цикл так же называется *циклом с постусловием*, так как условие расположено после тела цикла

Формат записи цикла do while

```
do {
    //тело цикла
} while (условие);
```

Особенности:

- Условие может быть сложным;
- Операторные скобки { и } обязательны

Пора применить полученные знания для написания вашей первой интерактивной веб-страницы.

Веб-страница игры “Угадай число”

Эта простая игра поможет нам закрепить полученные знания. Смысл игры довольно простой. Компьютер загадывает число в заданном диапазоне чисел от 1 до 100. Пользователь делает попытку угадать число, вводя его с клавиатуры. Если введённое число оказывается больше или меньше загаданного, компьютер сообщает об этом и дает ещё одну попытку. Если же введённое число совпадает с загаданным, то игра заканчивается, и компьютер сообщает о победе пользователя. Для большего интереса, ограничим количество попыток для угадывания числа 7 попытками.

```
<!DOCTYPE html>
<html>
<head>
    <meta charset="utf-8">
    <title>Угадай число</title>
</head>
<body>
```

```
    <script>
        var answer = parseInt(Math.random() * 100);
        var attempt = 1;
        var attemptsCount = 7;

        while (attempt <= attemptsCount) {
            var userAnswer = prompt("Угадайте число от 0 до 100, у вас 7
попыток. Попытка № " + attempt);
            userAnswer = parseInt(userAnswer);

            if (userAnswer > answer) {
                alert("Ваш ответ слишком большой.");
            } else if (userAnswer < answer) {
                alert("Ваш ответ слишком маленький.");
            } else if (userAnswer == answer) {
                alert("Вы угадали!");
                break;
            } else {
                alert("Необходимо ввести число!");
            }
            attempt++;
        }

        if (attempt > attemptsCount) {
            alert("К сожалению вы не угадали. Было загадано число: " + answer);
        }
    </script>
</body>
</html>
```

Домашнее задание

1. **Игра в загадки.**
    a. Загадать загадку. Если ответ верен – поздравить пользователя. Затем сообщить, что игра окончена.
    b. Если ответ неверный – написать пользователю, что он не угадал.
    c. Добавить еще 3 загадки. Подсчитать количество правильных ответов, сообщить пользователю.
    d. (по желанию). В качестве верного ответа принимать несколько вариантов ответов. Например, “Стул”, “стул” и “табуретка”.
1. **Банковская программа.** Пользователь вводит сумму вклада и процент, который будет начисляться ежегодно. Отобразить размер вклада поочередно на ближайшие 5 лет.

2. **Дописать игру “Угадай число” (по желанию).** Сделать игру с двумя игроками с бесконечным числом попыток. Сделайте возможность выйти из игры по желанию одного из игроков.