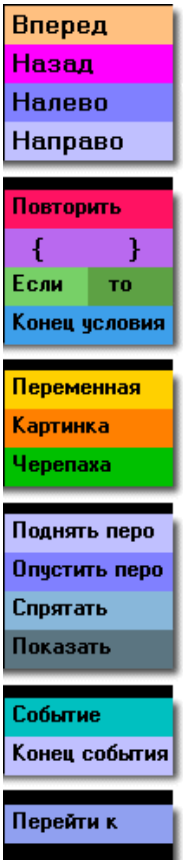
Хрунов К.А. школа при Посольстве России в Великобритании

ПРОГРАММИРУЕМ на GAME LOGO

Учебник GameLogo

1.

1. Исполнитель черепашка. Первые шаги



Первоначальные команды: "вперед", "назад", "налево", "направо". Команда "повторить". Правило 360 градусов. Рисование многоугольников.

Вначале программист дает черепашке простые приказы, например **ВПЕРЕД 100**, что означает "передвинуться вперед на 100 шагов", или **НАЛЕВО 60**, т. е. "сделать поворот влево на 60 градусов". Эти команды можно использовать для создания программ, рисующих геометрические фигуры, необходимо только запомнить, что каждая команда пишется на отдельной строке.

Вставлять в программу команды удобно, используя кнопки в левой части экрана.

Шаги черепашки очень маленькие - равны расстоянию между двумя соседними точками на экране, поэтому действие команды "ВПЕРЕД 1" можно и не заметить.

Выполняя команды НАЛЕВО или НАПРАВО, черепашка поворачивается на заданный угол (при этом она считает, что угол задан в градусах). При повороте черепашка остается на месте, не смещаясь ни в какую сторону, меняется только ее направление. Не забывайте, что направление поворота (налево или направо) определяется "с точки зрения" черепашки.

Запускают программу на выполнение с помощью большой красной кнопки с надписью **Выполнить.**

Попробуем вместе начертить квадрат. Чтобы заставить черепашку двигаться по квадрату, мысленно проделайте за нее весь путь и опишите получившееся.

Если сторона квадрата будет длиной в 100 черепашьих шагов, можно составить такую последовательность команд:

вперед 100
налево 90
вперед 100
налево 90
вперед 100
налево 90
вперед 100
налево 90

В этой последовательности 4 раза повторяется группа команд ВПЕРЕД 100 НАЛЕВО 90. Чтобы не писать одни и те же команды четыре раза подряд, познакомимся с еще одной командой, которую знает черепашка, - командой **ПОВТОРИТЬ**. Наша программа, рисующая квадрат, будет выглядеть так:

повторить 4 {
вперед 100
налево 90
}

В фигурных скобках в нужной последовательности записываются те команды, которые необходимо сделать черепашке несколько раз.

Фигурные скобки можно ставить и так:

повторить 4
{
вперед 100
налево 90
}

или так:

повторить 4{
вперед 100
налево 90}

Для того чтобы увидеть выполнение команд в более медленном режиме, можно воспользоваться командой ПАУЗА, после которой необходимо указать время в миллисекундах. В одной секунде - 1000 миллисекунд. То есть, если мы хотим сделать задержку в полсекунды, следует дать команду **пауза 500**.

Теперь подумаем, как нарисовать равносторонний треугольник?

При рисовании равностороннего треугольника черепашка должна нарисовать три одинаковых линии и сделать три одинаковых поворота. При создании программы нам поможет одно простое правило, которое называется "Правило 360 градусов".

Правило 360 градусов

Обратите внимание, что при рисовании квадрата черепашка начинает и заканчивает свой путь в одной и той же точке и смотрит в одну и ту же сторону, т.е. ее начальное и конечное положения совпадают. Рисуя квадрат, она совершает полный оборот, т.е. поворачивается на 360 градусов. Разделив 360 на 4 (равных поворота), мы получим 90 - размер угла поворота. При рисовании треугольника черепашка должна сделать три одинаковых поворота и вернуться в начальное положение. Следовательно, каждый из углов поворота равен 360 : 3 = 120.

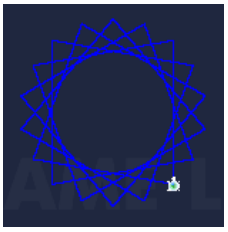
Напишем программу:

повторить 3 {
вперед 100
налево 120
}

Можно легко научить черепашку рисовать правильный пятиугольник. Каждый раз черепашка поворачивается на угол, равный 72 градусам (360:5=72). Шестиугольник (360:6=60)

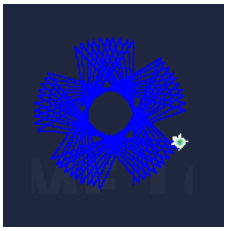
Итак, теперь черепашка умеет рисовать правильные многоугольники.

Попробуем еще что-нибудь нарисовать:



повторить 18 {

вперед 200
налево 100}
Или еще:



повторить 50 {
вперед 200
налево 145
}

А теперь попробуйте сами.

2. Черепашья графика

Выбор цвета и толщины пера черепашки. Команды "поднять перо" и "опустить перо". Как вернуть черепашку в исходное положение. Перемещение в точку с заданными координатами. Команды "спрятать черепаху" и "показать черепаху". Изменение фона.

Научим черепашку выбирать разные цвета для пера, которым она рисует.

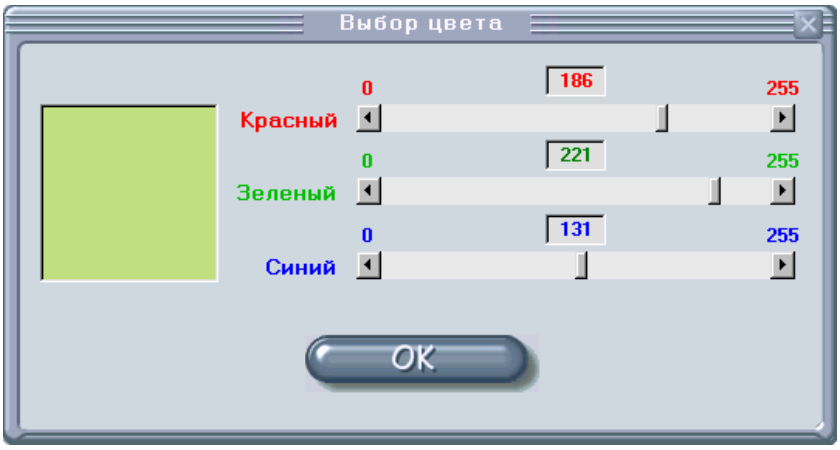
Перед тем как начать что-либо рисовать, дадим команду ЦВЕТ и укажем номер цвета, тогда черепашка возьмет перо нужного цвета.

Вставлять в программу команду выбора цвета удобно, используя цветные кнопки с числами цветов в нижней части экрана. Всего черепашка знает 16 цветов (черный цвет имеет номер 0).

Попробуем написать программу с командами выбора цвета.

цвет 12
вперед 50
цвет 14
вперед 50
цвет 10
вперед 50

Следующая за белым цветом (15) кнопка  помогает воспользоваться функцией RGB (Red - Красный, Green - Зеленый, Blue - Синий).



Функция RGB возвращает один из 16 миллионов цветов 24-битной палитры. Значения R, G и B должны лежать в пределах от 0 до 255.

Для управления толщиной пера служит команда ПЕРО.

Попробуем:

```
перо 5
цвет RGB (186, 221, 131)

повторить 18 {
  вперед 250
  налево 140
}
```

Для управления пером служат две команды:

ПОДНЯТЬ ПЕРО - поднимает перо у черепашки, после чего черепашка перестает оставлять след при движении.

ОПУСТИТЬ ПЕРО - опускает перо у черепашки, после чего черепашка оставляет след при движении.

В заключение еще несколько полезных команд:

СПРЯТАТЬ ЧЕРЕПАХУ - делает черепашку невидимой.

ПОКАЗАТЬ ЧЕРЕПАХУ - делает черепашку видимой.

ДОМОЙ - по этой команде черепашка возвратится в исходное положение, в центр экрана, головой вверх.

МЕСТО - передвигает черепашку в место с указанными координатами. Размеры экранного поля составляют 800 точек по горизонтали и 600 точек по вертикали. Точка с координатами (0, 0) находится в верхнем левом углу. Точка с координатами (800, 600) - в нижнем правом углу. Центр экрана - (400, 300).

Пример:
место 150, 200

КРУГ - рисует круг заданного радиуса с центром в месте, в котором стоит черепаха.

Пример:
круг 250

ЗАКРАСЬ - закрашивает замкнутую область, в которую входит точка с заданными координатами.

Пример:
круг 200
закрась 400, 300

или

повторить 4 {
вперед 100
налево 90 }
закрась черепаха.х - 10 , черепаха.у - 10

ОЧИСТИТЬ ФОН - очищает фон, стирает все нарисованное.

Фон рабочего поля можно менять командой **ФОН =** с указанием имени файла, лежащего в папке "Фоны". Для того чтобы выбрать файл фона, напишем команду фон=, затем нажмем на закладку меню фонов  в правой части экрана и сделаем двойной клик на каком-либо изображении фона.

Пример:
фон = gamelogo.jpg

3. Переменные

Что такое переменная. Объявление переменных. Присваивание значений. Основные арифметические действия. Случайные числа. Команда "пиши".

Для хранения различных значений в языках программирования используются переменные. Переменной называется область памяти, имеющая имя. Само слово "переменная" подразумевает, что содержимое этой области памяти может изменяться. Имя переменной может состоять как из одного, так и из нескольких символов. В именах переменных разрешается использовать строчные и прописные буквы, цифры и символ подчёркивания, который тоже считается буквой. Имя переменной не должно содержать пробелов. Первым символом обязательно должна быть буква. Имя переменной не должно совпадать с зарезервированными в Лого словами.

В GameLogo все переменные должны быть объявлены. Это означает, что в начале программы должен быть приведен список всех используемых переменных.

3.1. Объявление переменных

Переменные объявляют с помощью ключевого слова ПЕРЕМЕННАЯ, после которого пишут имя переменной. В GameLogo каждая переменная объявляется в отдельной строке.

Пример:

```
переменная х
переменная БУМ
переменная моё_число
```

Таким образом, будут объявлены переменные х, БУМ, моё_число.

3.2. Присваивание

Для присваивания значений переменным в GameLogo служит знак "=". Выражение, стоящее справа от знака присваивания, вычисляется, и полученное значение присваивается переменной, стоящей слева от знака присваивания. При этом предыдущее значение, хранящееся в переменной, стирается и заменяется на новое.

Оператор "=" не следует понимать как равенство. Например, выражение **a = 5** следует читать как "присвоить переменной a значение 5".

Примеры:

x = 5 + 3 'сложить значения 5 и 3, результат присвоить переменной x (записать в переменную x)

b = a + 4 'прибавить 4 к значению, хранящемуся в переменной a, полученный результат присвоить переменной b (записать в переменную b)

b = b + 2 'прибавить 2 к значению, хранящемуся в переменной b, полученный результат присвоить переменной b (записать в переменную b)

В правой части значение переменной может использоваться несколько раз:

c = b * b + 3 * b

3.3. Арифметические операции

Арифметические операции записываются в выражениях следующим образом:

+ (плюс)
- (минус)
*** (умножение)**
/ (деление)

Пример:

x = 3 'переменной x будет присвоено значение 3
y = x + 5 'к значению, хранящемуся в переменной x, будет прибавлено число 5,
'полученный результат будет записан в переменную y
z = x * y 'значения переменных x и y будут перемножены,
'результат будет записан в переменную z
z = z - 1 'от значения, хранящегося в переменной z, будет отнято 1
'результат будет записан в переменную z

Таким образом, в переменной **z** будет храниться число 23.

3.4. Переменные в качестве аргументов

Переменные или выражения, состоящие из переменных, могут использоваться в качестве аргументов различных команд.

Пример:

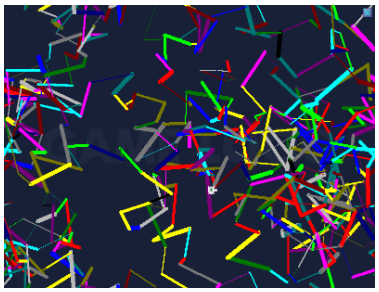
цвет x
вперед a + 50

3.5. Случайные числа

С помощью ключевого слова СЛУЧАЙНЫЙ можно получать случайные значения, которые представляют из себя дробные числа, лежащие в диапазоне от 0 до 1. Если нам требуется случайное число в диапазоне от 0 до 100, то возвращаемое значение нужно умножить на 100. Если, к примеру, нам потребуется случайное значение в диапазоне от 40 до 70, то возвращаемое значение нужно умножить на 30 и прибавить к результату 40.

Попробуем написать программу, создающую абстрактное компьютерное искусство.

```
повторить 500 {
  перо случайный * 10
  цвет случайный * 15
  вперед случайный * 100
  налево случайный * 360
}
```



3.6. Команда вывода

Чтобы узнать число, хранящееся в переменной, его можно вывести на экран с помощью команды ПИШИ.

Пример:

пиши x

С помощью команды ПИШИ можно писать на экране не только значения переменных, но и различные слова. Чтобы вывести на экран какие-либо слова, их следует взять в кавычки.

Пример:
пиши "Здравствуй, мир!"

В команде ПИШИ можно сочетать вывод значений переменных и печать каких-либо слов. Для этого их необходимо соединить с помощью соединяющего знака "+".

Пример:
пиши "Ответ: " + x

Цвет текста можно задавать командой РУЧКА, указав номер цвета из палитры 15 цветов или с помощью функции **RGB**.

Пример:

ручка 12
ручка RGB (186, 221, 131)

Кроме того, в качестве аргумента команды РУЧКА, также как и у команды ЦВЕТ, можно использовать переменные и выражения.

Пример:

ручка x

3.7. Команда ввода

Для ввода значений используется команда СПРОСИ.

Пример:

переменная x
спроси x
пиши x

Пример с добавлением сообщения:

переменная x
спроси "сколько яблок?", x
пиши x

Или просто:

переменная x

```
спроси сколько яблок?, x
пиши x
```

или вот так:

```
переменная x

переменная N

N = 5

спроси "сколько яблок у " + N + " детей?", x
пиши "У "+ N + " детей " + x + " яблок"
```

4. Циклы

Цикл со счетчиком "повторить для". Построение спирали. Цикл с условием "повторять пока". Вложенные циклы. Оформление программы и комментарии.

4.1. Что такое циклы?

Одно из ценнейших свойств компьютера, да и любого автомата, - способность много-много раз без устали повторять одни и те же действия.

Многократное повторение последовательности каких-либо действий в программе называют циклом, а саму последовательность действий - телом цикла. Когда необходимо создать цикл, программисты говорят: "Организуем цикл".

Процессы, при реализации которых повторяется выполнение одних и тех же действий, называют циклическими.

Мы уже знакомились с самыми простыми циклами, которые организовывали с помощью команды ПОВТОРИ. Теперь обратимся к более сложным видам циклов - циклу со счетчиком и циклу с условием.

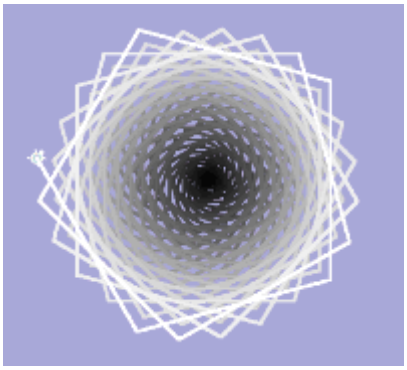
4.2. Цикл со счетчиком

Представим себе, что нам необходимо нарисовать спираль. Наша спираль должна будет раскручиваться из центра, все более и более увеличиваясь в размерах. При этом угол, на который должна поворачиваться черепашка после того, как нарисует очередной сегмент

спирали, будет все время одинаковым. А вот для того, чтобы спираль "раскручивалась", черепашке нужно будет каждый раз идти вперед хотя бы на шаг больше, чем в предыдущий.

Мы можем написать следующую программу:

```
переменная x

повторить 255 {
  x = x + 1
  цвет RGB( x, x, x)
  вперед x
  направо 75
}
```



На каждом шаге цикла (именно так говорят о каждом повторении) переменная x будет увеличиваться на 1. Мы используем значение этой переменной в функции **RGB** и в команде **вперед**. Сначала x будет равен 0, так как никакое значение ему не присвоено. На первом шаге цикла x станет равен 1, потом 2, потом 3 и т. д., пока не достигнет значения 255 на последнем шаге цикла. В данном случае переменная x в цикле будет своеобразным счетчиком. Таким образом, мы организовали цикл со счетчиком.

Для организации циклов со счетчиком существует специальная форма записи. Попробуем написать нашу программу, используя эту форму.

```
переменная x

повторить для x = 1 до 255 {
  цвет RGB( x, x, x)
  вперед x
  направо 75
}
```

В данном случае в команде *повторить* мы видим следующую запись

"для x = 1 до 255", которая означает, что в качестве переменной для счетчика используется переменная "x". Сначала ей присваивается начальное значение **"x = 1".** Затем указывается, до какого значения x повторять цикл.

Циклы со счетчиком чаще всего используются, когда заранее известно количество повторений, которые должны быть выполнены.

Еще одним параметром для цикла со счетчиком является **ШАГ**. Это необязательный параметр, но иногда он может быть очень полезным. Если этот параметр не задан, то переменная, используемая в качестве счетчика, на каждом шаге цикла увеличивается на 1. А если шаг задан, то переменная изменяется на величину шага.

Для примера рассмотрим задачу о нахождении суммы четных чисел, лежащих в интервале от 100 до 200. В данной задаче нам потребуется последовательно перебирать четные числа в заданном интервале. Первым числом будет 100, вторым - 102, третьим - 104. То есть шаг, с которым будет изменяться число, равен 2. Таким образом мы можем написать программу, в которой организуем цикл со счетчиком и использованием шага.

```
переменная x
переменная S
повторить для x = 100 до 200 шаг 2 {
  S = S + x
}
пиши S
```

В GameLogo параметры цикла со счетчиком (начальное значение, конечное значение, шаг) могут быть только целочисленными.

4.3. Цикл с условием

Теперь рассмотрим другую задачу. В банк положили 1000 рублей под проценты. Каждый месяц на эту сумму начислялось 5%. Причем на следующий месяц проценты начислялись уже на сумму вместе с начисленными в предыдущий месяц поцентами (процент на процент). Через сколько месяцев сумма на счету достигла 2000 рублей?

Для решения данной задачи лучше всего применить цикл с условием. Такой цикл повторяется, пока выполняется условие, заданное в команде повторить.

```
переменная S
переменная m

S = 1000
```

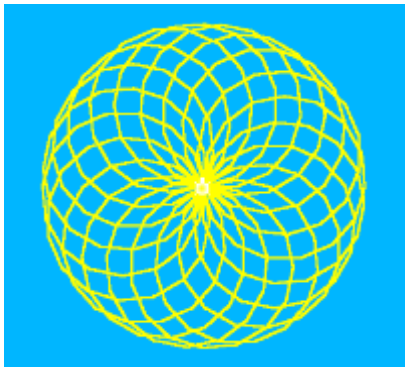
```
повторять пока S < 2000 {
  S = S + S * 0.05
  m = m + 1
}

пиши m
```

Циклы с условием чаще всего используются, когда заранее не известно количество повторений и цикл должен работать до тех пор, пока выполняется условие.

4.4. Вложенные циклы

Один цикл повторения может находиться внутри другого цикла.
При использовании вложенных циклов следует соблюдать простое правило. Один цикл внутри другого должен находиться так же, как одна матрешка внутри другой матрешки.

Рассмотрим следующий пример:



```
повторить 20 {

    повторить 15 {
    направо 24
    вперед 40
    }
    направо 18
}
```

В данном примере цикл, рисующий правильный 15-угольник, повторяется 20 раз, причем черепашка после того, как нарисует очередной 15-угольник поворачивается на 18 градусов. Этот поворот необходим для того, чтобы каждый последующий многоугольник рисовался не на том же месте, где и предыдущий, а с небольшим сдвигом. Чтобы расположить 15-угольники красиво по кругу, необходимо каждый раз поворачиваться на угол, равный 360:20, что составит 18 градусов.

4.5. Бесконечный цикл

Организовать бесконечный цикл можно, воспользовавшись циклом с условием, в котором условие будет выполняться всегда.
Например:

```
повторять пока 1 = 1 {
}
```

4.6. Оформление программы

Для того, чтобы ваша программа была понятна и легче воспринималась, следует придерживаться двух простых правил.

- Для отделения одной части программы от другой используйте пустые строки.
- Пользуйтесь отступами для выделения блоков команд, стоящих внутри условий и циклов.
- Добавить отступ можно с помощью сочетания клавиш Ctrl + Tab.
- Проиллюстрируем применение данных правил.

Запись 1.

```
переменная x
повторить 4 {
повторить для x = -1 до 100 {
вперед 10
направо 100 - x }
налево 20 }
```

Запись 2.

```
переменная x
  повторить 4 {
      повторить для x = -1 до 100 {
         вперед 10
         направо 100 - x
       }
      налево 20
  }
```

4.7. Комментарии

Еще одной важной частью оформления программы являются комментарии. Комментарии служат для того, чтобы снабдить программу ясными и понятными пояснениями. В GameLogo комментарий начинается со знака опострофа ' и может располагаться как на отдельной строке, так и после какой-либо команды.

Пример программы с комментариями.

```
'------------------------------------------------------------------------

' Программа "Бесконечная спираль"

'------------------------------------------------------------------------

переменная x
переменная p
спрятать черепаху
перо 4
повторять пока 1 = 1 { ' начало бесконечного цикла

  p = случайный * 10 + 65 ' задаем угол поворота для спирали

  повторить 255 {
    x = x + 1
    цвет RGB(x, x, x)
    вперед x
    направо p
  }
  домой

  x = 0 ' обнуляем переменную для количества шагов и цвета

} ' конец бесконечного цикла
```

5. Условия

5.1. Условия в программах. Способы записи условий.

Очень часто наши действия зависят от того или иного условия. Например:
если на улице хорошая погода, то можно совершить прогулку;
если дождь, то берём зонт, иначе зонт не берём.
В программах также можно пользоваться условиями для того, чтобы определить, следует ли выполнять то или иное действие.

Для того чтобы задать какое-либо условие, в GameLogo применяют конструкцию следующего вида:

если *<условие>* **то** *<команда>*

При истинности условия будет выполняться команда, стоящая после **"то"**, в противном случае команда, стоящая после **"то",** выполнена не будет.

В выражениях, создающих условия, используются знаки сравнения:

=	равно
>	больше
<	меньше
>=	больше или равно
<=	меньше или равно
<>	не равно

Если необходимо выполнить разные действия в зависимости от того, выполняется условие или нет, применяют следующую конструкцию:

если *<условие>* **то** *<команда 1>* **иначе** *<команда 2>*

При выполнении условия будет выполнена *<команда 1>,* если же условие не выполнится, то будет выполнена *<команда 2>.*

Пример:

Выдержит ли мост, если по нему проедет грузовик весом 2000 кг., на который погрузили 50 коробок весом 80 кг. каждая?

Предел нагрузки моста - 5000 кг.

```
переменная m
m = 2000 + 50 * 80
если m > 5000 то пиши "не выдержит" иначе пиши "выдержит"
```

В том случае, если при выполнении условия требуется выполнение нескольких команд, используется следующий тип записи условия:

если *<условие>* **то**
<команда>
<команда>
<команда>
...
конец условия

Ключевое слово **"конец условия"** обозначает конец многострочной конструкции, и его наличие в этом случае обязательно.

Конструкция **если ... то ... иначе** аналогична конструкции **если ... то**, но позволяет задать действия, выполняемые как при выполнении условия, так и в случае его невыполнения.

если *<условие>* **то**
  <команда>
  <команда>
  <команда>
  ...
иначе

  <команда>
  <команда>
  <команда>
  ...
конец условия

Если заданное условие не выполняется и запись условия содержит ключевое слово **"иначе",** выполнится последовательность команд, расположенных следом за **"иначе".** После чего управление перейдет к командам, следующим после строки **"конец условия".**
В случае использования многострочных способов записи условий количество команд может быть любым (в том числе и одна команда). Кроме того, следует помнить, что **"если ... то", "иначе" и "конец условия"** пишутся на отдельных строках.

Попробуем написать программу из предыдущего примера с использованием многострочной записи условия.

```
переменная m
  m = 2000 + 50 * 80
  если m > 5000 то
     пиши "не выдержит"
  иначе
     пиши "выдержит"
  конец условия
```

Так же, как и в случае применения циклов, условия могут быть вложены одно в другое.

5.2. Безусловный переход

Команда безусловного перехода **ПЕРЕЙТИ** К осуществляет переход исполнения программы к указанной метке. Метка ставится на отдельной строке. Имя метки может быть любым, но не должно содержать пробелы и знаки препинания. После имени метки всегда ставится двоеточие. Пробел между именем метки и двоеточием не ставится.

Пример:

СТАРТ:

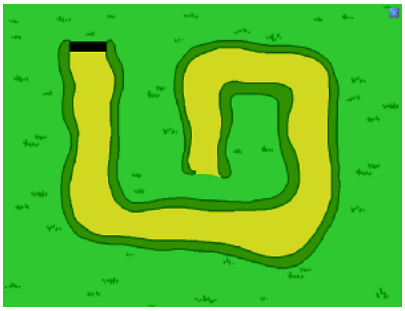
вперед 10
направо 5
перейти к СТАРТ

В данном примере с помощью безусловного перехода организовано бесконечное повторение команд, находящихся между меткой **СТАРТ** и командой **перейти к СТАРТ**.

6. Датчик

6.1. Значения датчика. Пример использования датчика.

Черепашка имеет датчик, который расположен точно в ее центре. Датчик может реагировать на поверхность, по которой движется черепашка.
Значения, возвращаемые датчиком, находятся в диапазоне **от 0 до 100**. Если черепашка движется по черной поверхности, то датчик возвратит значение 0. При движении по белой поверхности значение датчика будет равно 100.
Датчик реагирует не на цвет поверхности, а на ее тон. Значение датчика зависит от того, насколько светлой или темной является поверхность под черепашкой. При движении по светлой поверхности значения датчика будут больше 50, на темной поверхности - меньше 50.
Текущее значение датчика можно подставлять в любое выражение или команду, используя слово **"датчик".**



В стандартной поставке GameLogo есть фон **path.gif**. Попробуем провести черепашку по дорожке до черного прямоугольника.
Установим фон командой **фон = path.gif** и поднимем перо у черепашки. В основе программы будет цикл с условием, который будет повторяться до тех пор, пока черепашка не достигнет черного квадрата, то есть пока значение датчика не равно нулю **(повторять пока датчик <> 0).**
На каждом шаге цикла с помощью условия мы будем проверять, на светлой или темной поверхности находится черепаха **(если датчик > 50 то).** Если черепашка находится на

светлой дорожке, то дадим ей задание продвинуться вперед на несколько шагов **(вперед 7).** Иначе, если черепашка находится над темной поверхностью, дадим ей задание вернуться назад на два-три десятка шагов **(назад 30)** и повернуться на 90 градусов в правую сторону **(направо 90).**

```
фон = path.gif
поднять перо
повторять пока датчик <> 0
{
  если датчик > 50 то
    вперед 7
  иначе
    назад 30
    направо 90
  конец условия
}
```

Об использовании датчика для прохождения по лабиринту, правилах и алгоритмах поиска выхода вы можете прочитать в статье

6.2. "Прохождение лабиринта: моделирование робота в среде GameLogo".

Одним из самых простых правил для прохождения лабиринта является правило "одной руки": двигаясь по лабиринту, надо все время касаться правой или левой рукой его стены. Этот алгоритм, вероятно, был известен еще древним грекам. Придется пройти долгий путь, заходя во все тупики, но в итоге цель будет достигнута. Хотя у этого правила и есть один недостаток, но о нем мы поговорим позже.

Попробуем описать робота, действующего в соответствии с правилом "правой руки".

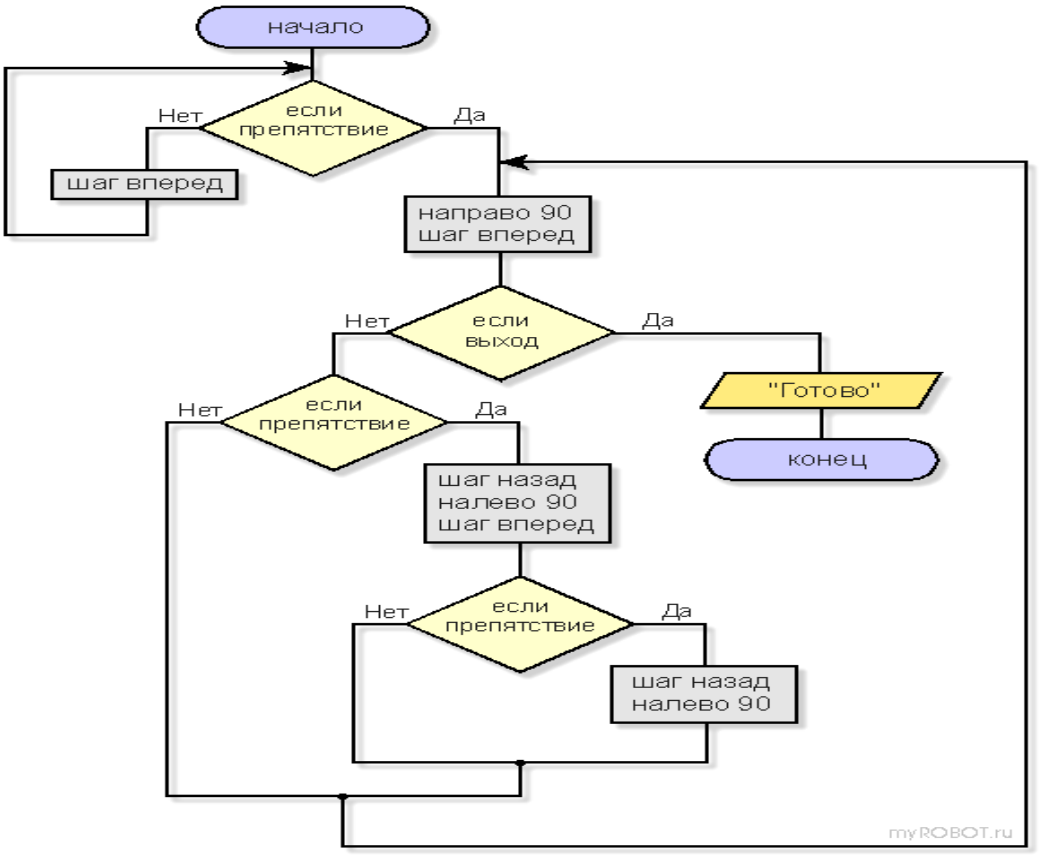
В начале своей работы робот должен найти стену, по которой он будет следовать. Для этого он может просто двигаться вперед, пока не упрется в преграду.

После того как робот наткнулся на препятствие, он начинает передвигаться в соответствии с правилом "правой руки".

Двигаясь вдоль стены, робот следит, есть ли проход справа. Если проход есть, робот должен идти по нему, чтобы не оторваться от стены справа.

Если прохода нет - впереди стена - робот поворачивает налево. Если прохода снова нет, он еще раз поворачивает налево, таким образом разворачиваясь на 180 градусов, и идет в обратном направлении.

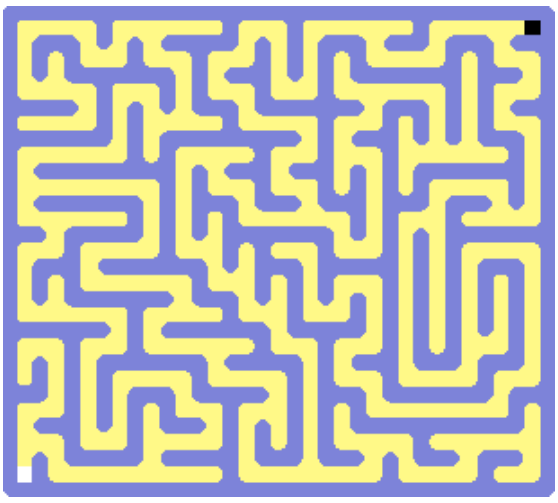
Блок-схема алгоритма для робота, работающего по правилу "правой руки", представлена на рисунке.



Попробуем проверить работу данного алгоритма и напишем для него программу. Для этой цели обратимся к среде программирования GameLogo. Эта среда является удобным средством для моделирования различных алгоритмов, связанных с управлением роботами. В ней есть исполнитель черепаха, который по своей сути является не чем иным, как самым настоящим роботом. Черепаха располагает очень удобным набором команд - вперед, направо, налево, назад. Кроме того, в центре черепахи есть датчик, принимающий значение от 0 до 100, в зависимости от тона поверхности, на которой она находится.

Диалект языка Лого, который мы будем использовать, очень прост и похож на Basic. Познакомиться с командами языка можно здесь.

В архиве с GameLogo есть фоны с лабиринтами, одним из которых мы и воспользуемся.



В самом начале программы дадим команду черепахе, чтобы она подняла перо (по умолчанию черепаха оставляет после себя след).

Размер поля - 800 на 600 точек. Исходное положение для черепахи находится в месте с координатами 115, 545 (белый квадрат).

Цвет дорожек лабиринта - светлый, на них датчик будет принимать значения больше 50. Цвет стен лабиринта - темный, значение датчика будет меньше 50. Выход из лабиринта представлен черным квадратом, значение

датчика над которым будет равно 0.

Объявим переменную флаг, с помощью которой будем контролировать, достигнут ли выход из лабиринта.

Напишем программу и запустим ее с помощью большой красной кнопки с надписью "Выполнить".

```
переменная флаг

фон = maze1.gif
   поднять перо
      место  115, 545

' поиск первой стены

повторять пока датчик > 50 {
      вперед 12
}
' правило правой руки

повторять пока флаг = 0 {

      направо 90
      вперед 12

      если датчик = 0 то

            флаг = 1

      иначе

            если датчик < 50 то

                  назад 12
                  налево 90
                  вперед 12

                  если датчик < 50 то
                        назад 12
                        налево 90
                  конец условия

            конец условия

      конец условия
}

пиши "цель достигнута"
```

Робот для прохождения лабиринта
на базе ATmega8
(соревнования Micromouse competition)

Если известно, что у лабиринта нет отдельно стоящих стенок, то есть нет замкнутых маршрутов, по которым можно возвращаться в исходную точку, то такой лабиринт называют односвязным и его всегда можно обойти полностью, применив правило "одной руки".

Если же лабиринт содержит отдельно стоящие стенки, то, применяя правило "одной руки", не всегда можно пройти все коридоры и тупики. Лабиринты с отдельно стоящими стенками и с замкнутыми маршрутами называются многосвязными. При этом многосвязные лабиринты можно разделить на две группы: без "петли" вокруг цели (замкнутый маршрут не проходит вокруг цели) и с замкнутой "петлей" вокруг цели (цель можно обойти по замкнутому маршруту).

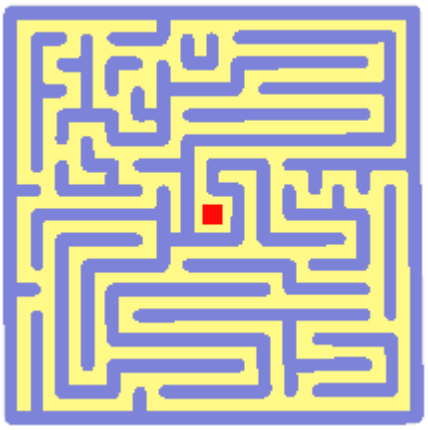
В многосвязных лабиринтах второй группы правило "одной руки" не работает и, применяя его, достичь цели невозможно. Но и эти лабиринты можно пройти, полагаясь на точный алгоритм.

Решение задачи о таких лабиринтах принадлежит сравнительно позднему времени, и начало ему положено Леонардом Эйлером. Эйлер не без оснований полагал, что выход из любого лабиринта может быть найден, и притом сравнительно простым путем.

Универсальный алгоритм прохождения любых лабиринтов был описан только через столетие в книге французского математика Э. Люка "Recreations matematiques", изданной в 1882 году. Интересно, что Люка при описании алгоритма указал на первенство другого французского математика М. Тремо. Таким образом, алгоритм стал известен как **алгоритм Люка-Тремо.**

Тремо предлагает следующие правила: выйдя из любой точки лабиринта, надо сделать отметку на его стене (крест) и двигаться в произвольном направлении до тупика или перекрестка; в первом случае вернуться назад, поставить второй крест, свидетельствующий, что путь пройден дважды - туда и назад, и идти в направлении, не пройденном ни разу, или пройденном один раз; во втором - идти по произвольному направлению, отмечая каждый перекресток на входе и на выходе одним крестом; если на перекресте один крест уже имеется, то следует идти новым путем, если нет - то пройденным путем, отметив его вторым крестом.

Зная алгоритм Тремо, можно скорректировать поведение легендарного Тесея. Вдохновленный подарком любимой Ариадны, он уверенно идет по лабиринту. Вдруг перед ним возникает ход, по которому уже протянута нить... Что делать? Ни в коем случае не пересекать ее, а вернуться по уже известному пути, сдваивая нить, пока не найдется еще один непройденный ход.



Применив вариант алгоритма Тремо, отец теории информации Клод Шеннон (Claude Elwood Shannon) построил одного из первых самообучающихся роботов. Шеннон дал ему звучное имя "Тесей", но в истории "Тесей" стал больше известен как "мышь" Шеннона. "Мышь" сначала обследовала весь лабиринт, а затем (во второй раз) проходила весь путь значительно быстрее, избегая участков, пройденных дважды.

В наши дни роботы, проходящие лабиринт, являются участниками одного из самых интересных состязаний думающих машинок, которое проходит в нескольких странах мира. Эти соревнования носят общее название Micromouse competition и по своим техническим новациям относятся к лидерам робототехнического спорта.

На первой Российской Олимпиаде Роботов проводились соревнования, целью которых было прохождение своеобразного лабиринта: за наиболее короткое время, двигаясь через "открытые двери" в стенках, робот должен был добраться от места старта до места финиша. Контролировать свое движение робот мог по черным линиям, нанесенным на пол лабиринта.

7. Объекты

Кроме объекта черепашка, в GameLogo возможно использование спрайтовых объектов, имеющих тип **картинка**. Объект **картинка** имеет сходный с черепашкой набор свойств и методов.

7.1. Методы и свойства объектов

Когда мы говорим о методах объекта, то подразумеваем действия, которые этот объект способен выполнять. Например, такой объект, как **черепаха**, может выполнять команды **вперед, назад, налево, направо**. Так как это главный объект в GameLogo, то, когда мы используем команды черепахи, мы не указываем название объекта. Но если бы нам понадобилось написать эти команды в полном виде, то они выглядели бы следующим образом:

черепаха.вперед 100
черепаха.налево 90

В данном случае после имени объекта (черепаха) через точку пишется метод, который поддерживается этим объектом.

Кроме методов, черепаха обладает набором свойств. К свойствам черепах относятся цвет пера, его толщина, координаты черепахи и т. д. Свойства так же, как и методы, пишутся через точку после имени объекта. Для изменения таких свойств черепахи, как цвет и толщина пера, мы можем написать следующий вариант записи:

черепаха.цвет 14
черепаха.перо 5

7.2. Координаты и угол

В набор свойств черепахи входят координаты и угол поворота.

Координата по горизонтали (ось X) может быть задана следующей командой:

черепаха.х = 200

Координата по вертикали (ось Y) может быть задана командой:

черепаха.у = 500

При задании координат могут использоваться как английские, так и русские символы. Например, при обращении к свойству, задающему координату по горизонтали, мы можем воспользоваться как английской буквой **x** ("экс"), так и русской буквой **х** ("ха").

Начало координат находится в вернем левом углу поля. Таким образом, ось X направлена слева направо, а ось Y - сверху вниз.

Помимо координат можно задавать угол поворота черепахи командой следующего вида:

черепаха.угол = 50

Такие свойства черепахи, как координаты и угол, могут использоваться в выражениях или в качестве аргумента команд.

К координатам и углу черепахи можно обращаться в краткой форме:

.х - координата по оси X
.у - координата по оси Y
.угол - угол поворота черепахи

7.3. Объект картинка

Чтобы использовать в программе объект **картинка**, его следует объявить, дав ему имя. Имя может быть любым, но не должно совпадать с зарезервированными ключевыми словами, не должно содержать пробелов и знаков препинания. После объявления картинке следует сопоставить файл с рисунком, который будет отображаться на экране. Файл можно выбрать из меню с картинками с помощью двойного клика.

Попробуем объявить объект **картинка** и сопоставить ему файл с изображением.

```
картинка мяч
мяч = BALL.BMP
```

Запустив подобный пример, мы сможем наблюдать появившуюся картинку в левом верхнем углу экрана.

Картинка обладает схожим с черепашкой набором методов и свойств. Первоначально координаты картинки равны нулю. Картинка имеет ширину и высоту. Узнать их можно, подведя мышь к изображению в меню с картинками в правой части экрана. Например, для файла BALL.BMP это будут размеры 50x50 точек.

Попробуем поставить нашу картинку в центр поля. Координаты картинки считаются по ее верхнему левому углу. Таким образом, если центр поля имеет координаты 400 по горизонтали и 300 по вертикали, то, чтобы поставить картинку точно в центр, мы должны отнять от координат центра поля половину ширины и высоты нашей картинки.

```
мяч.х = 400 - 25
мяч.у = 300 - 25
```

Теперь попробуем запустить нашу картинку, подобно черепашке, используя методы вперед, назад, направо, налево.

```
повторить 100 {
мяч.вперед 20
мяч.направо 10
}
```

Мы можем спрятать нашу картинку командой

```
спрятать мяч
```

Показать картинку мяч можно командой

показать мяч

Кроме того, у **картинки** есть свойство **видимость**. Если присвоить этому свойству значение 0, картинка станет невидимой. Сделать ее снова видимой можно, присвоив свойству **видимость** значение **1**.

мяч.видимость = 0
мяч.видимость = 1

Изменить угол поворота движения картинки с помощью изменения свойства угол.

мяч.угол = 90

Также можно использовать команду **место**.

мяч.место 500, 200

В программе может существовать несколько объектов типа картинка. Следует только помнить, что к каждому объекту следует обращаться по присвоенному ему имени.

8. События

8.1. Событие "нажата клавиша". Коды клавиш. Управление с помощью клавиатуры.

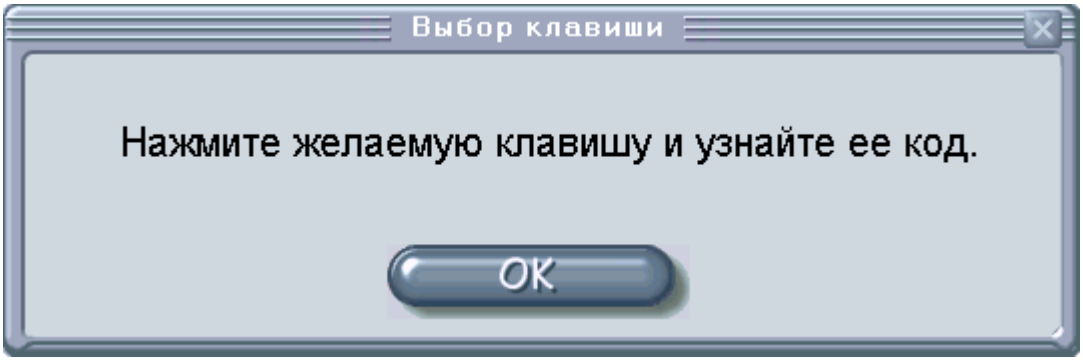
Если мы хотим, чтобы наша программа реагировала на нажатие какой-либо клавиши на клавиатуре, то мы можем воспользоваться программированием событий клавиатуры.
Каждая клавиша на клавиатуре имеет свой код. Например, клавиша **"стрелка вверх"** имеет код **38**. Попробуем написать программу, реагирующую на нажатие этой клавиши.

```
событие нажата клавиша 38
  вперед 5
конец события
```

Запустим программу и попробуем нажать на клавиатуре клавишу **"стрелка вверх".** Черепаха продвинется вперед на 5 шагов.

Любые команды, расположенные между строкой **событие нажата клавиша** и строкой **конец события**, будут выполнены тогда, когда будет нажата соответствующая клавиша.

Узнать код клавиши можно, нажав на кнопку  .



Попробуем добавить в нашу программу события для других клавиш со стрелками.

```
событие нажата клавиша 40
  назад 5
конец события

событие нажата клавиша 37
  налево 5
конец события

событие нажата клавиша 39
  направо 5
конец события
```

Программный код, отвечающий за события, лучше располагать после основной программы.

Переменные, объявленные как в начале основной программы, так и внутри кода события, являются общими для всей программы.

Попробуем добавить еще какой-нибудь код, например для клавиши "**ц**".

```
событие нажата клавиша 87
  повторить 10 {
      повторить 9 {
        вперед 15
        направо 40
      }
      направо 36
  }
```

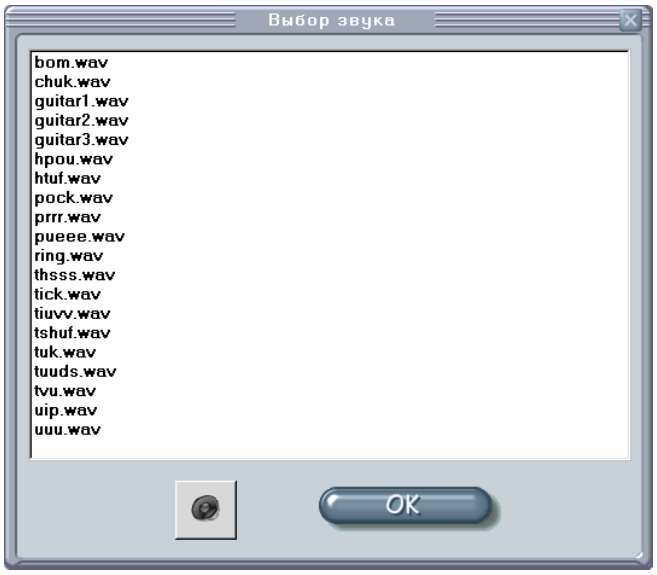
конец события

9. Мультимедиа

9.1. Команда "звук". Использование звуков в программах.

Мультимедийные возможности в GameLogo представлены командой воспроизведения звуковых файлов в формате **wav.**

Чтобы воспроизвести звук в программе, необходимо вставить команду **звук =** , нажать на кнопку выбора звуков  и выбрать звуковой файл.



Получившаяся команда должна будет выглядеть подобным образом:

звук = guitar1.wav

Если необходимо воспроизвести несколько звуков, следующих один за другим, то следует поставить команду пауза между ними. Пауза необходима для того, чтобы предыдущий звуковой файл успел прозвучать прежде, чем начнется воспроизведение следующего.

Пример:

```
повторить 2
{
  звук = guitar3.wav
  пауза 300
}
```

```
повторить 3
{
  звук = tuuds.wav
  пауза 300
}

повторить 2
{
  звук = prrr.wav
  пауза 300
}
```

10. Черепашка считает

10.1. Математические действия и функции, работа с дробными числами.

Черепашка понимает следующие математические действия и функции:

+	плюс
-	минус
*	умножить
/	разделить
^	возвести в степень
%	остаток при целом делении двух чисел
int	целая часть числа
sin	синус
cos	косинус
tan	тангенс
atn	арктангенс
log	логарифм

Рассмотрим два примера.

Пример 1:
Дано 5,7 мили. Задание: перевести мили и ярды.
(В одной миле 1760 ярдов.)

```
переменная х
переменная м
переменная я

  х = 5.7
```

```
  м = int(x)
  я = 1760 * (x - int(x))

пиши "миль - " + м + "; ярдов - " + я
```

Пример 2:
Найти факториал числа 10.

```
переменная x
переменная ф

  x = 10
  ф = 1

  повторить пока x > 0 {
    ф = ф * x
    x = x - 1
  }

пиши ф
```

11.Графики функций

11.1. Команда "точка". Построение графиков функций.

Для построения графиков функций удобно воспользоваться командой ТОЧКА. Команда точка ставит точку в месте с заданными координатами. Цвет точкисовпадает с текущим цветом пера черепашки. Размер точки зависит от текущей толщины пера черепашки.
Попробуем поставить желтую точку размером 5 пикселов.

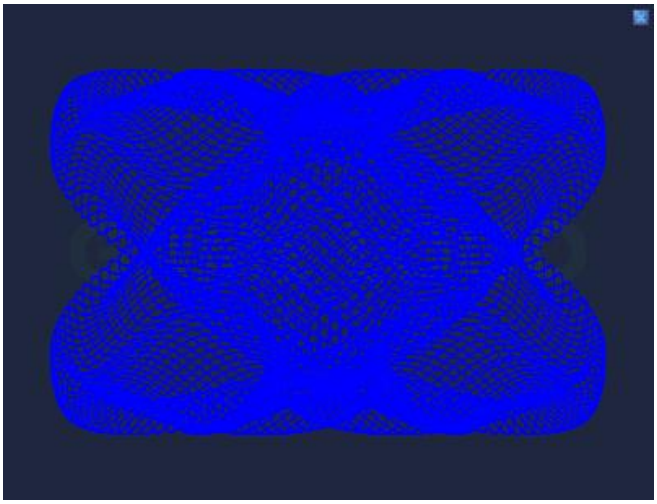
```
цвет 14
перо 5
точка 200, 100
```

Теперь рассмотрим программу построения графика функции **y = x * Sin (x).** Нам понадобятся три переменных: x и y - для кординат; t - для приращения координаты по оси X. Установим толщину пера. Спрячем черепаху и зададим начальное смещение по оси X, равное -20. После этого создадим цикл для построения 800 точек графика заданной функции. При вычислении координаты по оси Y увеличим получившийся результат в 15 раз, чтобы амплитуда графика выглядела более красивой, и добавим смещение по оси Y до середины экрана, прибавив 300.

```
переменная x
```

```
переменная y
переменная t

перо 3
спрятать черепаху

  t = -20

  повторить для x = 1 до 800
  {
    t = t + 0.05
    y = Sin(t) * t * 15 + 300
    точка x, y
  }
```

Попробуем построить еще один график.



```
переменная z
переменная t
переменная x
переменная y
переменная k
переменная m
переменная s

спрятать черепаху

  s = 0.001 * pi

  повторить для z = 0 до 100000
  {
    t = t + s
    x = Sin(0.99 * t) - 0.7 * Cos(3.01 * t)
    y = Cos(1.01 * t) + 0.1 * Sin(15.03 * t)
    k = 200 * x + 400
    m = 200 * y + 300
    точка k, m
  }
```