Руководство по языку Logo

1. Команды перемещения

Разные команды делают разные вещи, одни из них требуют дополнительную информацию, некоторые возвращают её. Далее мы опишем каждую из поддерживаемых команд. Заметьте, что такие команды выделяются тёмно-синим в редакторе кода.

Двигаем Черепашку.

Существует несколько команд для перемещения Черепашки по экрану.

forward (fw)
forward X
перемещает Черепашку на **X** шагов вперёд. Когда перо опущено Черепашка будет оставлять за собой след. Эта команда также может записываться как **fw**.

backward (bw)
backward X
перемещает Черепашку назад на **X** шагов. Когда перо опущено Черепашку будет оставлять за собой след. Может так же записываться как **bw**.

center
Перемещает Черепашку в центр холста.

go
go X,Y
Предписывают Черепашке занять определённое место на холсте. Это место находится на **X** шагов от левой границы и на **Y** шагов от верхней границы холста. Примечание: при перемещении Черепашка не будет оставлять след.

gox
gox X
Используется для перемещения Черепашки на **X** шагов по горизонтали от левой границы холста, высота остается неизменной.

goy
goy Y
Используется для перемещения Черепашки на **Y** шагов по вертикали от верхней границы холста, положение относительно левой границы остаётся неизменным.

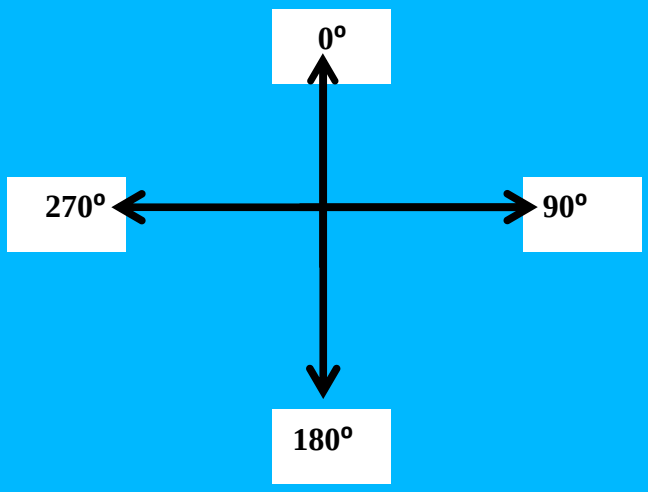
Поворачиваем Черепашку

turnleft (tl)
turnleft X
Предписывает Черепашке повернуть на **X** градусов налево. Может записываться и как **tl**.

turnright (tr)
turnright X
Предписывает Черепашке повернуть на **X** градусов направо. Может записываться и как **tr**.

direction (dir)
direction X
Устанавливает направление Черепашки на X градусов относительно 0, а не относительно предыдущего направления. Может записываться как **dir**.

Направления установлены так:



2. У Черепашки есть перо

У Черепашки есть перо, которым она рисует линию во время перемещения. Есть несколько команд для управления пером. В данном разделе они будут описаны подробно.

penup (pu)

penup отрывает перо от холста. Пока перо оторвано, Черепашка не будет рисовать линию во время перемещений. Может записываться и как **pu**. См. также **pendown**.

pendown (pd)

pendown опускает перо на холст. Когда перо опущено, Черепашка рисует линию при перемещениях. Может записываться и как **pd**. См. также **penup**.

penwidth (pw)

penwidth X

Устанавливает ширину пера (ширину линии) в X шагов. Может записываться и как **pw**.

pencolor (pc)

pencolor R,G,B

Устанавливает цвет пера. В качестве параметров указывается интенсивность красной (R), зеленой (G) и синей (B) составляющих цвета (0..255). Может записываться и как **pc**.

3.Команды для работы с холстом.

Существует несколько команд для работы с холстом.

canvassize (cs)

canvassize X,Y

С помощью этой команды можно поменять размер холста. В качестве входных параметров задаются ширина X и высота Y в Черепашьих шагах. Может записываться и как **cs**.

canvascolor (cc)

canvascolor R,G,B

canvascolor устанавливает цвет холста. Входными параметрами является комбинация RGB. Может записываться и как **cc.**

4.Команды очистки

Существуют две команды очистки холста.

clear

Этой командой вы можете очистить холст от всех следов. Все остальное останется по-

прежнему: позиция и угол направления Черепашки, цвет холста, видимость Черепашки и размер холста.

reset

reset Очищает более объёмно, чем команда clear. После выполнения этой команды всё будет выглядеть так, как будто вы только что запустили KTurtle. Черепашка будет расположена в центре экрана, цвет холста будет белым, цвет пера Черепашки также будет белым.

spriteshow

Делает Черепашку видимой после того, как она была скрыта.

spritehide

Скрывает Черепашку. Это полезно, когда Черепашка не уместна в ваших рисунках.

5.Может ли Черепашка печатать на холсте?

Черепашка может написать всё, что вы ей прикажете.

print

print X

Команда **print** используется для указания Черепашки написать что-либо на холсте. В качестве входных параметров можно передавать строки или числа. Вы можете печатать различные числа и строки, комбинируя их вместе оператором +. Вот маленький пример

```
$year = 2003
$avtor= "Cies"
print $avtor + " started the KTurtle project in " + $year + " and still enjoys working on it!"
```

fontsize

fontsize X

Устанавливает размер шрифта, используемого для печати. Входной параметр один, он должен быть числом.

6.Команды для получения случайных чисел.

Команда, которая бросает игральную кость для вас.

random

random X,Y

random – команда, которая имеет входные и выходные параметры. На входе требуются два числа, первое (X) задает нижний порог получаемых чисел (минимум), второе (Y) задает верхний порог (максимум). Выходной параметр – это псевдослучайное число, которое не меньше минимума и не больше максимума. Вот маленький пример:

```
repeat 500 {
  x = random 1,20
  fw x
  tl 10 - x
}
```

Используя случайное число, вы можете привнести немного хаоса в вашу программу.

7.Контейнеры.

Контейнеры - это символы или слова, которые используются программистом для хранения чисел или текста. Контейнеры, содержащие числа, называются переменными, а содержащие текст - строками.

Контейнеры до первого их использования содержат 0 по умолчанию. Вот пример:

```
print $n
```

Не будет напечатано ничего. Если мы будет использовать математические операции с пустыми контейнерами, то получим ошибки.

Переменные: числовые контейнеры

Давайте начнём с небольшого примера:

```
$x = 3
print $x
```

В первой строке символ $x объявляется переменной (числовым контейнером). Как вы можете увидеть, значение переменной $x устанавливается равной трём. На второй строке это значение выводится на печать.

Помните, что если мы хотим напечатать «x», мы должны написать

```
print "x"
```

Это было совсем просто, вот пример посложнее:

```
$A = 2004
$B = 25
$C = $A + $B
# следующая команда напечатает "2029"
print $C
bw 30
# следующая команда напечатает "2004 плюс 25"
print $A + " плюс " + $B
bw 30
# следующая команда напечатает "1979"
print $A - $B
```

В первых двух строках переменные $A и $B устанавливаются равными 2004 и 25.В третьей строке переменной $C присваивается результат $A + $B, который равен 2029. Остальное в примере – три команды печати на холсте с командой назад 30 между ними. Команда bw 30 используется здесь для уверенности в том, что каждая новая печать на холст будет на новой строке. В данном примере также демонстрируется, как переменные могут быть использованы в математических расчётах.

8.Текстовые контейнеры

Обычный текст заключается в кавычки, например:

```
print "Привет, программист!"
```

то, что между кавычками, называется строкой.

Строки во многом похожи на переменные. Главное же отличие состоит в том, что строки не могут быть использованы в математических выражениях и вопросах. Пример использования строк:

```
$x = "Привет "
$name = ask "Введите своё имя..."
print $x + $name + ", как дела?"
```

В первой строке $x присваивается текст «Привет». Затем строке $name присваивается результат выполнения команды ask и на холсте печатается комбинация полученных трёх строк.

Эта программа спрашивает ваше имя. Когда вы вводите, к примеру, имя «Павел», программа выводит на печать «Привет Павел, как дела?». Пожалуйста, не забывайте, что «+» - единственная математическая операция, которая может использоваться при работе со строками.

9.Может ли Черепашка делать вычисления?

Да, Черепашке под силу заниматься математическими вычислениями. Вы можете складывать (+), вычитать (-), умножать (*), делить (/) и возводить число в степень (^). Вот пример, демонстрирующий использование всех этих команд.

```
$a = 20 - 5
$b = 15 * 2
$c = 30 / 30
$d = 1 + 1
$f = 2 ^ 3
print "a: "+$a+", b: "+$b+", c: "+$c+", d: "+$d+", f: "+$f
```

Знаете ли вы, какие значения примут a, b, c, d и f? Обратите внимание на использование символа присваивания =.

Если в программе вам нужно вычислить простое выражение, вы можете поступать следующим образом:

```
print 2004-12
```

Вот пример с приоритетом вычисления:

```
print ( ( 20 - 5 ) * 2 / 30 ) + 1
```

Выражение в скобках будет вычислено первым. В данном примере сначала будет получена разность 20 – 5, затем полученное значение будет умножено на 2, поделено на 30, и напоследок будет добавлена единица (результат равен 2).

10.Задаём вопросы, получаем ответы...

В следующем разделе мы обсудим команды контроля выполнения если и пока. В этом разделе мы будем использовать команду если для объяснения вопросов.

10.1.Вопросы

Простой пример с вопросом:

```
$x = 6
if $x > 5 {
  print "hello"
}
```

В данном примере вопросом является x > 5. Если будет получен ответ “истина” (true), будут выполнены команды в скобках. Вопросы - важная часть программирования. Чаще всего они используются вместе с операторами контроля, такими как if. Все числа и переменные (числовые контейнеры) могут сравниваться друг с другом с помощью вопросов.

Вот возможные вопросы:

Таблица 1. Типы вопросов

$a = = $b	равенство	ответ «истина» («true»), если a равно b
$a != $b	неравенство	ответ «истина» («true»), если a не равно b
$a > $b	больше	ответ «истина» («true»), если a больше b
$a < $b	меньше	ответ «истина» («true»), если a меньше b
$a >= $b	больше или равно	ответ «истина» («true»), если a больше или равно b
$a <= $b	меньше или равно	ответ «истина» («true»), если a меньше или равно b

Вопросы подсвечиваются (выделяются) в редакторе кода.

10.2.Вопросный клей

Вопросы также могут быть совмещены друг с другом с помощью «вопросного клея». Это позволяет использовать несколько вопросов как один большой.

```
$a = 1
$b = 5
if ($a < 5) and ($b == 5) {
      print "hello"
   }
```

В этом вопросе “склеивающее слово” - **and**, оно используется, чтобы совместить два вопроса ($a < 5, $b==5) вместе. Если с одной стороны результатом будет «ложь» (false), то и ответ на весь вопрос будет «false», потому, что с этим “склеивающим словом” обе стороны должны быть равны «истина» («true»)для получения ответа на весь вопрос «истина»(«true»).

Вот краткий обзор управляющих операторов, более детально они описаны ниже:

Таблица 2. “Склеивающие слова” для вопросов

and	Обе стороны должны быть равны «true» для получения ответа «true» на весь вопрос.
or	Если хотя бы одна сторона равна «true», то и ответ будет «true».
not	только если обе стороны «false», ответ будет «false».

Склеивающие слова подсвечиваются фиолетовым в редакторе кода.

And

Если вопросы соединяются при помощи **and**, для получения общего ответа "true", они все должны быть истинными, например:

```
$a = 1
$b = 5
if (($a < 10) and ($b == 5)) and ($a < $b) {
  print "хм"
}
```

or

Если ответ хотя бы на один из вопросов - "true", то и общий ответ будет таким же, например:

```
$a = 1
$b = 5
if (($a < 10) or ($b == 10)) or ($a == 0) {
 print "хм"
}
```

not

Это специальная приставка, меняющая ответ на противоположный, например:

```
$a = 1
$b = 5
if not (($a < 10) and ($b == 5)) {
  print "хм"
    }
else
{
  print "не хм :-)"
}
```

В этом примере ответ на заданный вопрос - "true", а приставка не изменяет это на "false", так что на холсте будет напечатано"не хм :-)".

11. Может ли Черепашка ждать?

Если вы уже немного попрактиковались в программировании в KTurtle, вы могли заметить, что Черепашка может рисовать чересчур быстро. Следующая команда позволяет избежать этого.

wait

wait X
wait указывает Черепашке подождать X секунд.

```
repeat 36 {
  forward 5
  turnright 10
  wait 0.5
}
```

Данный код рисует круг, при этом после каждого шага Черепашка будет ждать пол секунды. Это создаёт впечатление неторопливого движения.

12. Условное выполнение

if
if вопрос { ... }

Код, расположенный между скобками, будет выполнен только тогда, когда ответом на вопрос будет «true». Для более подробной информации прочитайте, пожалуйста, раздел Вопросы.

```
$x = 6
if $x > 5 {
  print "x больше пяти!"
}
```

В первой строке контейнеру $x присваивается 6. Во второй задаётся вопрос $x > 5. Если ответом на него будет «true», управляющий оператор if позволит выполнить код, расположенный между скобками.

13.Цикл "пока"

while
while вопрос { ... }

Управляющий оператор while очень похож на if. Разница в том, что пока будет повторять код, расположенный между скобками, до тех пор, пока ответом на вопрос не станет «false».

```
$x = 1
while $x < 5 {
  forward 10
  wait 1
  $x = $x + 1
}
```

В первой строке $x присваивается 1. На второй строке задаётся вопрос $x < 5. Так как ответ на этот вопрос - «true», оператор while начнёт выполнять код между скобками, пока ответом на вопрос не станет «false». В данном случае код между скобками будет выполнен 4 раза потому, что на каждом прогоне в пятой строке $x будет увеличиваться на 1.

else
if вопрос { ... } else { ... }

Может быть дополнением к оператору условного выполнения if. Код между скобками, расположенными после else будет выполнен тогда, когда ответом на вопрос будет «false».

```
reset
$x = 4
if $x > 5 { print "x больше пяти!" }
else { print "x меньше пяти!" }
```

If ставит вопрос, больше ли x чем 5. Т.к. x равно 4, ответ на вопрос - «false». Поэтому выполняется код в скобках после else.

14.Считающий цикл "для"

for начальное число **to** конечное число { ... }

For - это цикл «со счётчиком».

```
for $x = 1 до 10 { print $x * 7 forward 15 }
```

Каждый раз, когда выполняется код в скобках, значение x увеличивается на 1, и так до тех пор, пока x не станет равным 10. Код в скобках выводит на печать произведение x и 7. После завершения выполнения программы вы увидите таблицу умножения на 7.

15.Создавайте свои собственные команды!

Learn - это особенная команда, потому что она предназначена для создания ваших собственных команд. Создаваемые вами команды могут принимать входные параметры и возвращать различные значения. Давайте посмотрим, как создаются собственные команды:

```
learn circle $x {
  repeat 36 { forward $x turnleft 10 }
}
```

Новая команда называется **circle**. Её входным параметром является число - размер круга. Теперь команду circle можно использовать как же, как и обычные команды:

```
go 30,30
circle 20
go 40,40
circle 50
```

В следующем примере будет создана команда, возвращающая значение.

```
reset
learn square $n {
  $r = $n * $n
  return $r
}
$i = ask "Введите число и нажмите OK"
print $i + ", помноженное на себя: " + square $i
```

В данном примере создана новая команда с именем square. Ей передаётся число, а она возвращает его квадрат.

16. Математические, и другие функции.

ИмяПеременной=**round** Значение — округляет Значение до ближайшего целого числа. Например: в результате выполнения команды $a=round 3.54 переменная $a получит значение = 4, а в результате выполнения команды $b=round 3.49 переменная $b получит значение = 3.

ИмяПеременной=**sqrt** Число — вычисляет значение квадратного корня из Числа.

ИмяПеременной=**mod** Число1,Число2 – вычисляет остаток от деления Числа1 на Число2. Например: в результате выполнения команды $k=mod 10,3 переменная $k получит значение = 1.

pi — константа, равная числу **π**

17. Функции, обрабатывающие положение черепашки:

ИмяПеременной=**getx —** присваивает переменной значение текущей горизонтальной координаты (x) черепашки

ИмяПеременной=**gety —** присваивает переменной значение текущей вертикальной координаты (y) черепашки