Муниципальное бюджетное общеобразовательное учреждение
средняя общеобразовательная школа № 141
городского округа Самара

Изучение языка программирования Logo в образовательной оболочке KTurtle

Учебное пособие

Составил:

Гальчинский
Владислав
Анатольевич

Самара, 2014

Оглавление

Глава 1. Введение

KTurtle - это образовательная программная оболочка для изучения языка программирования Logo, которая позволяет программировать максимально легко и просто. KTurtle великолепно подходит для обучения детей основам математики, геометрии и программирования. Все команды, используемые для программирования, сделаны в стиле Logo. Отличительной особенностью языка Logo является то, что все его команды переведены на родной язык программиста.

Пакет KTurtle назван так по главному персонажу программы — Черепашке. Пользователь управляет передвижениями Черепашки, используя команды языка Logo, для рисования на холсте.

Что такое Logo?

Первая версия языка программирования Logo была создана Сеймуром Пейпертом (Seymour Papert) в Лаборатории Искусственного Интеллекта Массачусетского Технологического Института в 1967 году как ответвление языка программирования LISP. Впоследствии вышло в свет много его версий. К 1980 Logo становится очень популярным, активно используются его версии для MSX, Commodore, Atari, Apple II и IBM PC компьютеров — главным образом, в образовательных целях. В 1985 году компания LCSI выпустила среду Mac®Logo как профессиональный инструмент программирования, но она не получила распространения. Массачусетский Технологический Институт до сих пор поддерживает сайт, посвящённый Logo, который располагается по адресу http://el.media.mit.edu/logo-foundation/.

На сегодняшний день существует большое количество версий и диалектов Logo, информацию о которых можно найти на сайте МТИ (см. выше) или при помощи любой поисковой системы. Данная версия Logo (KTurtle) изначально нацелена на образовательные нужды и вряд ли пригодится для профессионального программирования.

Возможности Kturtle

KTurtle обладает замечательными особенностями, которые позволят начать программировать легко и непринуждённо. Вот небольшой список особенностей KTurtle:

Встроенный интерпретатор Logo, использующий XML, поддерживает пользовательские функции и динамические типы.

Выполнение можно замедлить и остановить в любое время.

Мощный редактор команд Logo с подсветкой синтаксиса, нумерацией строк и многим другим.

Холст с результатом работы программы может быть сохранено как изображение или распечатано.

Холст имеет функцию переноса Черепашки на другой край при достижении границы, так что вам будет сложно её потерять.

Контекстная подсказка по всем командам Logo, которая вызывается простым нажатием F2.

Все команды Logo могут быть переведены на любой язык.

Диалог с сообщениями об ошибках, знакомящий детей с отладкой.

Технология упрощённого программирования.

Полноэкранный режим.

Множество прилагаемых примеров программ на Logo, переведённых на разные языки, помогут быстрее освоиться новичку.

Глава 2. Использование Kturtle

Главное окно KTurtleГлавное окно KTurtle имеет две основных области: редактор кода, расположенный слева, который предназначен для ввода команд Logo и холст, расположенного справа, на котором выводятся результаты выполнения команд. Холст — место для игр Черепашки, где она перемещается и рисует. Также в главном окне есть меню, из которого производится управление программой, панель инструментов, которая предоставляет быстрый доступ к часто используемым действиям и строка состояния, на которой показывается различная информация о состоянии Kturtle.

Редактор кода

Редактор кода предназначен для ввода команд Logo. Он обладает всеми возможностями, присущими современному редактору. Большинство команд вызывается из меню Правка и Сервис. Редактор кода может быть «прилеплен» к любой границе главного окна или же, будучи выведен как отдельное окно, располагаться в любом месте рабочего стола.

Существуют разные пути получения кода в редакторе. Простейший — использовать готовый пример. Для этого необходимо вызвать пункт Файл->Открыть пример в меню Файл и выбрать необходимый пример в появившемся диалоге. Имя файла говорит о том, что данный пример демонстрирует (например square.logo (англ. — квадрат) предназначен для построения квадрата). Выбранный файл будет открыт в редакторе кода и его можно будет запустить на выполнение, вызвав Файл->Выполнить сценарий.

Вы можете открывать файлы Logo, вызывая пункт Файл->Открыть.

Третий путь — ввод кода в редакторе вручную или вставка его из буфера обмена.

Справа в строке состояния показывается текущая позиция курсора в виде номеров строки и столбца.

Холст

Холст — это область, предназначенная для вывода результатов выполнения команд. Другими словами, это место для игр Черепашки. После ввода каких-либо команд в редакторе кода и вызове пункта Файл->Выполнить сценарий могут произойти две вещи: или программа выполнится успешно и вы увидите некоторые изменения на холсте, или, если были допущены ошибки, вы получите сообщение, которое уведомит вас о характере допущенной ошибки.

Это сообщение поможет устранить ошибки.

Построенное в ходе выполнения команд изображение может быть сохранено в файл (Файл->Сохранить холст) или напечатано (Файл->Печать).

Главное меню

В главном меню вы найдёте все действия, доступные в программе KTurtle. В нём присутствуют следующие группы: Файл, Правка, Вид, Сервис, Настройка и Справка. В данном разделе все они описаны подробно.

Меню Файл

Создать
Файл->Создать (Ctrl-N)
Создаёт новый, пустой файл программы на Logo.

Открыть
Файл->Открыть... (Ctrl-O)
Открывает файл Logo.

Последние файлы
Файл->Последние файлы
Выводит список последних файлов Logo, с которыми работали в программе.

Открыть пример
Файл->Открыть пример... (Ctrl-E)
Показывает папку, в которой расположены файлы примеров Logo. Для того, чтобы команды в этих файлах были на вашем языке, необходимо предварительно выбрать язык в меню Настройка->Настроить Kturtle....

Сохранить
Файл->Сохранить (Ctrl-S)
Сохраняет текущий открытый файл Logo.

Сохранить как
Файл->Сохранить как...
Сохраняет текущий открытый файл Logo в выбранном месте.

Сохранить холст
Файл->Сохранить холст...
Сохраняет текущий рисунок на холсте как файл изображения.

Скорость выполнения
Файл->Скорость выполнения
Если скорость «Обычно» (по умолчанию), трудно уследить за тем, что происходит на холсте. С помощью этого пункта можно замедлить скорость выполнения, чтобы проследить за каждым шагом Черепашки. Кроме «Обычно» вы можете поставить скорость «Медленно», «Медленнее» и «Очень медленно». Когда установлен одна из медленных скоростей, в редакторе кода выделяется выполняемая команда.

Выполнить сценарий
Файл->Выполнить сценарий (Alt-Return)
Запускает на выполнение команды Logo, введённые в редакторе кода.

Пауза
Файл->Пауза (Pause)
Приостанавливает выполнение. Этот пункт меню доступен только во время выполнения сценария.

Остановить выполнение
Файл->Остановить выполнение (Escape)
Прекращает выполнение команд. Этот пункт меню доступен только во время выполнения сценария.

Печать
Файл->Печать... (Ctrl-P)
Печатает код, находящийся в редакторе, или рисунок на холсте.

Выход
Файл->Выход (Ctrl-Q)
Завершение работы Kturtle.

Меню Правка

Правка->Отменить действие (Ctrl-Z)
Отменяет последнее изменение в коде. KTurtle имеет неограниченное число шагов отмены.

Правка->Повторить (Ctrl-Shift-Z)
Возвращает последнее отменённое изменение в коде.

Правка->Вырезать (Ctrl-X)
Вырезает выделенный текст из редактора кода в буфер обмена.

Правка->Копировать (Ctrl-C)
Копирует выделенный текст из редактора кода в буфер обмена.

Правка->Вставить (Ctrl-V)
Вставляет текст из буфера обмена в редактор кода.

Правка->Найти... (Ctrl-F)
Поиск фразы в тексте программы.

Правка->Найти далее (F3)
Поиск следующего вхождения фразы в тексте программы.

Правка->Заменить... (Ctrl-R)
Производит замену фраз в редакторе кода.

Меню Вид

Вид->Полноэкранный режим (Ctrl-Shift-F)
Позволяет переключаться в полноэкранный режим.
Примечание: в полноэкранном режиме во время выполнения команд все, кроме холста, скрыто. Это позволяет создавать «полноэкранные» программы в Kturtle.

Вид->Показывать номера строк (F11)
Позволяет включить или выключить показ номеров строк в редакторе кода. Может быть полезно при поиске ошибок.

Меню Сервис

Сервис->Выбор цвета (Alt-C)
Вызывает диалог выбора цвета. Используя этот диалог, можно легко подобрать нужный цвет и вставить его цифровое представление в редактор кода.

Сервис->Увеличить отступ (Ctrl-I)
Этот пункт увеличивает отступ (добавляет пустое пространство) в строки выделенного фрагмента. Правильная расстановка отступов облегчает чтение кода. Все примеры используют расставленные отступы.

Сервис->Уменьшить отступ (Ctrl-Shift-I)
Этот пункт уменьшает отступ (убирает пустое пространство) из строк выделенного фрагмента.

Сервис->Снять отступ
Снимает все отступы на выделенных строках.

Сервис->Закомментировать (Ctrl-D)
Данная команда добавляет символы комментария (#) в выделенные строки. Весть текст, следующий за символом комментария до конца строки, игнорируется при выполнении кода. Комментарии позволяют программисту пояснить назначение того или иного фрагмента кода или скрыть от интерпретатора часть кода, чтобы он не выполнялся.

Сервис->Раскомментировать (Ctrl-Shift-D)
Удаляет символ комментария с выделенных строк.

Меню Настройка

Настройка->Показать/Скрыть панель инструментов
Переключает показ панели инструментов

Настройка->Показать/Скрыть строку состояния
Переключает показ строки состояния

Настройка->Дополнительная настройка
Здесь можно настроить параметры, которые обычно в настройке не нуждаются.
Подменю Дополнительная настройка имеет три пункта:
Настроить редактор... (стандартный диалог настройки редактора Kate), Комбинации клавиш... (стандартный диалог настройки комбинации клавиш приложения KDE) и Панели инструментов... (стандартный диалог настройки панели инструментов KDE).

Настройка->Настроить KTurtle...
Вызывает диалог настройки KTurtle. Здесь можно поменять язык команд Logo или установить новый размер холста по умолчанию.

Меню Справка

Справка->Руководство KTurtle (F1)
Вызывает руководство, которое вы в данный момент читаете.

Справка->Что это? (Shift-F1)
После активации данного пункта меню курсор приобретёт вид стрелки с вопросительным знаком. После этого, когда вы щёлкните на каком-либо

элементе в окне KTurtle, появится подсказка об этом элементе и его предназначении.

Справка->Справка по... (F2)

Очень полезная функция. Она вызывает справку по команде, рядом с которой находится курсор в редакторе кода. Например, вы используете команду напиши в своей программе и хотите узнать побольше о ней из руководства. Вам достаточно переместить курсор на команду и нажать F2. Руководство по KTurtle откроется на странице с описанием команды напиши.

Контекстная помощь очень важна при обучении программированию.

Справка->Сообщить об ошибке

Используйте данный пункт для уведомления разработчиков об ошибках в KTurtle. Это поможет делать программу лучше от версии к версии.

Справка->О программе KTurtle

Здесь вы найдёте информации об авторах KTurtle и лицензии под которой распространяется программа.

Справка->О KDE

Здесь вы найдёте информацию о KDE. Если вы до сих пор не знаете, что это такое – не проходите мимо.

Панель инструментов

Здесь вы можете быстро добраться до наиболее часто используемых действий. По умолчанию вы найдёте здесь большинство используемых действий, включая Выполнить сценарий и Прекратить выполнение.

Вы можете настроить панель инструментов под себя, используя Настройка->Дополнительная настройка->Панели инструментов...

Строка состояния

В строке состояния выводится информации от KTurtle. Слева показывается последнее действие. Справа показывается текущая позиция курсора (номер строки и столбца). Посередине показывается язык, используемый для ввода команд.

Глава 3. Начало работы

Когда вы запустите KTurtle, вы увидите рабочее поле программы.

KTurtleВ данном руководстве мы будем использовать команды на русском языке. Вы можете поменять язык команд Logo в разделе Язык диалога Настройки->Настроить KTurtle.... Только выбранный язык может применяться для ввода команд.

Первые шаги в Logo: встречаем Черепашку!

Вы, наверное, уже заметили Черепашку в центре холста, осталось только научиться, как управлять ею, используя команды в редакторе кода.

Движения Черепашки

Давайте начнём с изучения движений.

Наша Черепашка может перемещаться тремя способами:

(1) передвигаться **вперёд** и **назад,**
(2) поворачивать **направо** или **налево** или
(3) перескочить сразу к определённой точке холста.

Попробуйте, например, это:

вперёд 100
налево 90

Введите эти строчки в редакторе кода или просто скопируйте отсюда, а потом вставьте в редактор **(обратите внимание на использование буквы «ё»).** После этого запустите на код исполнение (Файл->Выполнить сценарий) и посмотрите результат.

После ввода и запуска команд, подобных приведённым выше, вы можете наблюдать следующее:

После запуска команд на выполнение Черепашка продвинулась вверх, рисуя линию, и затем повернулась на 90 градусов влево. Это произошло потому, что мы использовали команды **вперёд** и **налево.**

Как вы могли заметить, при вводе кода его цвет меняется — это называется подсветка кода (разные типы команд подсвечиваются по-разному). Подсветка делает чтение блоков кода удобнее.

Черепашка рисовала тонкой чёрной линией.

Может быть, вы получили **сообщение об ошибке**. Скорее всего, произошло одно из двух: вы совершили ошибку при копировании команд или же не выставили правильный язык ввода команд Logo (вы можете сделать это в меню Настройки->Настроить KTurtle..., Язык).

Как вы могли понять, команда **вперёд 100** указала Черепашке на то, что ей необходимо двигаться вперёд, оставляя за собой линию, а команда **налево 90** — повернуть на 90 градусов влево.

Обратитесь к справочнику Logo для полного описания следующих команд: вперёд, назад, налево и направо.

Ещё примеры

Первый пример был совсем простенький. Продолжаем.

```
нов_размер_холста 600,400
нов_цвет_холста 0,0,0
нов_цвет_пера 255,0,0
нов_ширина_пера 5
очисти

иди 20,20
направление 135

вперёд 200
налево 135
вперёд 100
налево 135
вперёд 141
налево 135
вперёд 100
налево 45

иди 40, 100
```

Вы можете набрать этот код вручную, либо скопировать его в редактор отсюда, либо открыть файл arrow.logo в папке примеров, выбрав пункт Открыть пример меню Файл. Запустите пример (Файл->Выполнить сценарий) и посмотрите результат.

Как вы могли заметить, второй пример содержит гораздо больше кода. Также появилась пара новых команд. Ниже дано краткое описание новых команд:

нов_размер_холста 600,400 устанавливает ширину в 600 и высоту холста в 400 пикселей. Ширина и высота одинаковы, холст будет квадратным.

нов_цвет_холста 0,0,0 задаёт чёрный цвет холста. 0,0,0 — это комбинация красной, зелёной и синей (по-английски: red, green, blue, или сокращённо RGB) составляющих цвета, если все значения установлены в

0, получится чёрный цвет.

нов_цвет_пера 255,0,0 устанавливает красный цвет для пера. 255,0,0 — это комбинация красной, зелёной и синей составляющих цвета, где красная составляющая равна 255, а все остальные — 0. Результатом будет красный цвет.

Если вы не разбираетесь в задании значений цвета, обратитесь к разделу глоссария «Коды цветов».

нов_ширина_пера 5 устанавливает толщину (размер) пера в 5 пикселей. С этого момента Черепашка будет рисовать линию толщиной 5 до тех пор, пока мы не зададим другую толщину пера.

очисти очищает холст.

иди 20,20 указывает Черепашке перескочить в определённое место холста. Отсчитывается от верхнего левого угла, в данном случае Черепашка должна перескочить на 20 пикселей влево от левой границы холста и на 20 пикселей вниз от верхней границы холста. Примечание: при использовании этой команды, Черепашка не рисует линию.

направление 135 задаёт направление Черепашки. Команды налево и направо изменяют направление Черепашки на заданный угол относительно текущей позиции. Команда **направление** изменяет направление Черепашки на заданный угол относительно 0 и не обращает внимание на предыдущее направление.

После команды направление следует множество команд вперёд и налево. Эти команды выполняют рисование.

В конце используется ещё одна команда **иди** для отвода Черепашки в сторону.

Прочитайте, пожалуйста, описание всех этих команд в справочнике. Там они объяснены более подробно.

Глава 4. Справочник Logo для Kturtle

Эта глава начинается с краткого описания различных типов инструкций. Потом идёт подробное описание каждой команды, затем описываются контейнеры, математические операции, ответы на вопросы и операторы условного выполнения. В конце даётся несколько примеров создания собственных команд с помощью команды **выучи.**

Типы инструкций

Как и любой другой язык, Logo содержит различные типы слов и символов. Ниже вкратце объясняется различие между типами.

Команды

Используя команды, вы предписываете Черепашке или KTurtle выполнить какое-либо действие. Некоторые команды нуждаются во входных параметрах, некоторые возвращают результат. В данном разделе объясняются все команды, используемые в KTurtle.

вперёд - это команда, с параметром, в нашем случае — это 100:
вперёд 100

Подробное описание всех команд, поддерживаемых KTurtle, находится в разделе: **«Команды»**

Числа

Числа, используемые с KTurtle, не отличаются от математических.
В Logo есть натуральные числа: 0, 1, 2, 3, 4, 5, ...; отрицательные: -1, -2, -3, ...; и десятичные дроби: 0.1, 3.14, 33.3333, -5.05, -1.0.
Числа используются в математических расчётах и вопросах. Они могут помещаться в контейнеры. Числа выделяются синим цветом в редакторе кода.

Строки

Пример:
напиши "Привет, я строка."

В этом примере напиши — команда, которой передаётся строка "Привет, я строка.". Строки начинаются и заканчиваются на ", так KTurtle может определить, что это строка.

Строки можно помещать в контейнеры.
Строки выделяются тёмно-красным в редакторе кода.

Имена

Имена можно давать контейнерам (переменным) и новым командам.
Имена не могут совпадать с именами команд. Например, нельзя назвать контейнер вперёд.

```
# попытка использовать вперёд в качестве имени,
# но оно уже занято командой, так что при выполнении получим ошибку
вперёд = 20

# это работает:
вперёд 20
```

Имена могут состоять из букв, цифр и символа подчёркивания (_), но первой всегда должна быть буква.
Подробно это изложено в разделе **«Контейнеры»** и справке по команде **«выучи».**

Присваивание

Присваивание производится символом =. В большинстве языков программирования лучше читать один символ = не как «равно», а как «становится». Слово «равно» больше подходит к ==, это используется в вопросах.

Присваивание служит для помещения значения в контейнер и изменения значения в нём, например:

```
x = 10
# контейнер x теперь содержит число 10
W = "Сейчас мне столько лет: "
# контейнер W теперь содержит строку "Сейчас мне столько лет: "
# это выводит на экран содержимое контейнеров W и x
напиши W + x
```

Другие примеры приведены в разделе **«Контейнеры».**

Математические знаки

KTurtle поддерживает все основные математические операции: **сложение (+), вычитание (-), умножение (*), деление (/), использование скобок (и).**

Описание использования и дополнительные примеры вы найдёте в разделе **«Математические вычисления».**

Вопросы

У программы можно спрашивать вопросы и получать на них ответы (истина или ложь).
Использование вопросов подробно описано в соответствующем разделе.

Склеивающие слова в вопросах

С помощью них можно соединять несколько вопросов. Это могут быть слова **и, или**, а также специальное слово **не**.
Использование склеивающих слов подробно описано в соответствующем разделе.

Комментарии

Комментарии - это строки, начинающиеся с **#.**
Например:
 # это комментарий!
напиши "это не комментарий"
 # напиши "а это - комментарий"

Комментарии могут содержать разъяснения к коду, ещё комментарием можно сделать какую-то команду, чтобы она не выполнялась.
Комментарии выделяются тёмно-жёлтым в редакторе кода.

4.1. Команды

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-зелёным в редакторе кода.**

Двигаем Черепашку

Существует несколько команд для перемещения Черепашки по холсту.
вперёд (вп)

вперёд X
вперёд перемещает Черепашку на X пикселей вперёд. Когда перо опущено, Черепашка будет оставлять за собой след. Эта команда также может записываться как вп.

назад (нд)

назад X
назад перемещает Черепашку назад на X пикселей. Когда перо опущено, Черепашку будет оставлять за собой след. Может так же записываться как нд.

налево (лв)

налево X
Предписывает Черепашке повернуть на X градусов налево. Может записываться как лв.

направо (пр)

направо X
Предписывает Черепашке повернуть на X градусов направо. Может записываться как пр.

направление (нпр)

направление X
Устанавливает направление Черепашки на X градусов относительно 0, а не относительно предыдущего направления. Может записываться как нпр.

центр

Перемещает Черепашку в центр холста.

Иди

иди X,Y

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

иди_гор

иди_гор X

иди_гор используется для перемещения Черепашки на X пикселей от левой границы холста, высота остаётся неизменной.

иди_верт

иди_верт Y

Используется для перемещения Черепашки на Y пикселей от верхней границы холста, положение относительно левой границы остаётся неизменным.

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.

перо_подними (пп)

перо_подними отрывает перо от холста. Пока перо оторвано, Черепашка не будет рисовать линию во время перемещений. Может записываться и как пп.

перо_опусти (по)

перо_опусти опускает перо на холст. Когда перо опущено, Черепашка рисует линию при перемещениях. Может записываться и как по. См. также перо_подними.

нов_ширина_пера (ншп)

нов_ширина_пера X

нов_ширина_пера устанавливает ширину пера (толщину линии) в X пикселей. Может записываться и как ншп.

нов_цвет_пера (нцп)

нов_цвет_пера R,G,B

нов_цвет_пера устанавливает цвет пера. В качестве параметров указывается интенсивность красной, зелёной и синей составляющих цвета (0..255). Может записываться и как нцп.

Команды для работы с холстом

Существует несколько команд для работы с холстом.

нов_размер_холста (нрх)

нов_размер_холста X,Y

С помощью этой команды можно поменять размер холста. В качестве входных параметров задаются ширина X и высота Y в пикселях. Может записываться и как нрх.

нов_цвет_холста (нцх)

нов_цвет_холста R,G,B

нов_цвет_холста устанавливает цвет холста. Входными параметрами является комбинация RGB. Может записываться и как нцх.

обёртка_вкл

Этой командой вы устанавливаете обёртку холста. Это значит, что при достижении края холста Черепашка не исчезнет, а окажется на его противоположной стороне.

обёртка_выкл

Этой командой вы отключаете обёртку холста. Это значит, что Черепашка может выйти за границы холста и «исчезнуть».

Команды очистки

Существуют две команды очистки холста.

очисти (очс)

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. Может записываться и как очс.

Сброс

Очищает более объёмно, нежели команда очисти. 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, Черепашка рисует чёрную линию.

Черепашка — это спрайт

Большинство людей и понятия не имеют, что такое спрайты, так вот, спрайты — это маленькие картинки, которые можно перемещать по экрану. Так что Черепашка — спрайт.

Ниже будет дано подробное описание всех команд работы со спрайтами.

Текущая версия KTurtle пока не поддерживает использование спрайтов, отличных от Черепашки. В ближайшем будущем вы сможете заменить Черепашку на любой другой персонаж какой хотите.

покажи (пж)

Делает Черепашку видимой после того, как она была скрыта. Так же может записываться как пж.

спрячь (сч)

Скрывает Черепашку. Это полезно, когда Черепашка неуместна в ваших рисунках. Может записываться и как сч.

Может ли Черепашка печатать на холсте?

Черепашка может написать всё, что вы ей прикажете.

напиши X

Команда напиши 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.

```
год = 2003
автор = "Сиес"
напиши автор + " начал проект KTurtle в " + год + " году и до сих пор с удовольствием работает над ним!"
```

нов_размер_шрифта X

Устанавливает размер шрифта, используемого для печати. Входной параметр один, он должен быть числом. Размер задаётся в пикселях.

Команды для получения случайных чисел

Команда, которая «бросает игральную кость» для вас — random: полезна для получения неожиданных результатов.

случайное X,Y

случайное — команда, которая имеет входные и выходные параметры. На входе требуются два числа, первое (X) задаёт нижний порог получаемых чисел (минимум), второе (Y) задаёт верхний порог (максимум). Выходной параметр — это псевдослучайное число, которое не меньше минимума и не больше максимума.

Вот маленький пример:

```
повтори 500 [
  x = случайное 1,20
  вперёд x
  налево 10 - x
]
```

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

Взаимодействие через диалоги

Диалог — небольшое всплывающее окно. Оно выводит сообщение или запрашивает что-либо. В KTurtle есть две команды для диалогов: сообщение и окно_вопроса.

сообщение X

Команде сообщение требуется передать строку, она будет показана в появившемся окне.

год = 2003
автор = "Сиес"
напиши автор + " начал проект KTurtle в " + год + " году и до сих пор с удовольствием работает над ним!"

окно_вопроса X

Команде окно_вопроса требуется передать строку, она будет показана в появившемся окне, аналогично команде сообщение. Но, кроме этого, в окне будет поле для ввода числа или текста.

Например

```
результат = окно_вопроса "Сколько вам лет?"
осталось = 18 - результат
напиши "Через " + осталось + " лет вам будет разрешено водить машину."
```

Если пользователь ничего не введёт и просто закроет окно, контейнер **осталось** будет пустой.

4.2. 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 ничего не содержат.

Вот пример:

```
напиши N
```

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

Переменные: числовые контейнеры

Давайте начнём с небольшого примера:

```
x = 3
напиши 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 не содержимое контейнера, а строку «x», мы должны написать

напиши "x"

Это было совсем просто, вот пример посложнее:

```
A = 2004
B = 25
C = A + B

# следующая команда напечатает "2029"
напиши C
назад 30
# следующая команда напечатает "2004 плюс 25"
напиши A + " плюс " + B
назад 30
# следующая команда напечатает "1979"
напиши 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 назад 30 используется здесь для уверенности в том, что текст будет выводится на новой строке. В данном примере также демонстрируется, как переменные могут быть использованы в математических расчётах.

Строки: текстовые контейнеры

Обычный текст заключается в кавычки, например:

напиши "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 "
имя = окно_вопроса "Введите своё имя..."
напиши x + ", " + имя + ". Как дела?"
```

В первой строке **x** присваивается текст «Привет». Затем строке **имя** присваивается результат выполнения команды **окно_вопроса** и на холсте печатается комбинация четырёх 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.

4.3. Может ли Черепашка делать вычисления?

Да, Черепашке под силу заниматься математическими вычислениями. Вы можете складывать (+), вычитать (-), умножать (*) и делить (/).

Вот пример, демонстрирующий использование всех этих команд.

```
a = 20 - 5
b = 15 * 2
c = 30 / 30
d = 1 + 1
напиши "a: "+a+", b: "+b+", c: "+c+", d: "+d
```

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

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

```
напиши 2004-12
```

Вот пример с приоритетом вычислений:

```
напиши ( ( 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).

4.4. 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.

Вопросы

Простой пример с вопросом:

```
x = 6
если x > 5 [
  напиши "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 если. Все числа и переменные (числовые контейнеры) могут сравниваться друг с другом с помощью вопросов.

Вот возможные вопросы:

Таблица 4.1. 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) **голубым** в редакторе кода.

Вопросный клей

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

```
a = 1
b = 5
если (a < 5) и (b = 5) [
  напиши "hello"
]
```

В этом вопросе «склеивающее слово» — **и**, оно используется, чтобы совместить два вопроса (a < 5, b=5) вместе. Если с одной стороны результатом будет «ложь», то и ответ на весь вопрос будет «ложь», потому, что с этим «склеивающим словом» обе стороны должны быть равны «истина» для получения ответа на весь вопрос «истина». **Не забывайте использовать скобки для выставления приоритетов вопросов.**

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

Таблица 4.2. «Склеивающие слова» для вопросов

И

Обе стороны должны быть равны «истина» для получения ответа «истина» на весь вопрос.

Или

Если хотя бы одна сторона равна «истина», то и ответ будет «истина».

Не

Только если обе стороны «ложь», ответ будет «ложь».

Склеивающие слова подсвечиваются **фиолетовым** в редакторе кода.

Примеры:

«И»

Если вопросы соединяются при помощи **и**, для получения общего ответа «истина», они все должны быть истинными, например:

```
a = 1
b = 5
если ((a < 10) и (b = 5)) и (a < b) [
  напиши "хм"
]
```

«Или»

Если ответ хотя бы на один из вопросов — «истина», то и общий ответ будет таким же, например:

```
a = 1
b = 5
если ((a < 10) или (b = 10)) или (a = 0) [
  напиши "хм"
]
```

«Не»

Это специальная приставка, меняющая ответ на противоположный, например:

```
a = 1
b = 5
если не ((a < 10) и (b = 5)) [
  напиши "хм"
]
иначе
[
  напиши "не хм ;-)"
]
```

В этом примере ответ на заданный вопрос — «истина», а приставка не изменяет это на «ложь», так что на холсте будет напечатано "не хм ;-)".

4.5. Контроль над выполнением

Управляющие операторы позволяют вам контролировать процесс выполнения (о чём говорит само их название).

Управляющие операторы выделяются **тёмно-зелёным** цветом и полужирным начертанием шрифта. Квадратные скобки, используемые вместе с ними, выделяются светло-зелёным.

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.

Жди

жди X

жди указывает Черепашке подождать X секунд.

```
повтори 36 [
  вперёд 5
  направо 10
  жди 0.5
]
```

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

Условное выполнение

Если

если вопрос [...]

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

```
x = 6
если x > 5 [
  напиши "x больше пяти!"
]
```

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

Цикл «пока»
Пока

пока вопрос [...]
Управляющий оператор пока немного похож на **если**. Разница в том, что пока будет повторять код (цикл), расположенный между скобками, до тех пор, пока ответом на вопрос не станет «ложь».

```
x = 1
пока x < 5 [
  вперёд 10
  жди 1
  x = x + 1
]
```

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

Если нет, «иначе»
Иначе

если вопрос [...] иначе [...]
Может быть дополнением к оператору условного выполнения **если**. Код между скобками, расположенными после иначе будет выполнен тогда, когда ответом на вопрос будет «ложь».

```
сброс
x = 4
если x > 5 [
  напиши "x больше пяти!"
]
иначе
[
  напиши "x меньше пяти!"
]
```

Оператор если ставит вопрос, больше ли x чем 5. Так как x равно 4, ответ на вопрос — «ложь». Поэтому выполняется код в скобках после иначе.

Считающий цикл «для»
Для

для начальное число до конечное число [...]
Цикл для — это цикл «со счётчиком».

```
для x = 1 до 10 [
  напиши x * 7
  вперёд 15
]
```

Каждый раз, когда выполняется код в скобках, значение x увеличивается на 1. Выполнение цикла продолжиться, пока x не станет равным 10. Код в скобках выводит на печать произведение x и 7. Результат работы этой программы — таблица умножения на 7.

4.6. Создавайте свои собственные команды

Выучи — 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:

```
выучи круг X [
  повтори 36 [
    вперёд X
    налево 10
  ]
]
```

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

```
выучи круг X [
  повтори 36 [
    вперёд X
    налево 10
  ]
]

иди 30,30
круг 20

иди 40,40
круг 50
```

В следующем примере будет создана команда, возвращающая значение.

```
сброс
выучи квадрат_числа n [
  r = n * n
  верни r
]
i = окно_вопроса "Введите число и нажмите OK"
напиши i + ", умноженное на себя: " + квадрат_числа i
```

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

Глава 5. Глоссарий

В данной главе вы найдёте объяснение большинства «непонятных» слов, встречающихся в данном руководстве.

Градусы

Градусы – единицы измерения углов или поворотов. Полный разворот – это 360 градусов, половина разворота - это 180 градусов и четверть разворота - 90 градусов. Входными параметрами команд налево, направо и направление являются углы в градусах.

Входные параметры и возвращаемые значения команд

Некоторым командам необходимы входные параметры, некоторые возвращают значения. Есть такие, которые имеют и вход, и выход, а есть, наоборот, не имеющие ни входных параметров, ни возвращаемых значений.

Вот несколько команд, имеющих только входные параметры:

```
вперёд 50
нов_цвет_пера 255,0,0
напиши "Привет!"
```

Команда вперёд принимает в качестве входного параметра число 50. Данный параметр указывает команде вперёд на сколько пикселей вперёд должна продвинуться Черепашка. Входным параметром для нов_цвет_пера является цвет, а для напиши это будет строка. И не забывайте, что входным параметром также может являться контейнер. Следующий пример продемонстрирует это:

```
икс = 50
напиши икс
строка = "Привет!"
напиши строка
```

Теперь приведём примеры команд, возвращающих значения:

```
х = окно_вопроса "Введите что-нибудь и нажмите ОК... спасибо!"
r = случайное 1,100
```

Команда окно_вопроса принимает в качестве входного параметра строку, а возвращает число или строку. Как вы можете заметить,

возвращаемое окно_вопроса значение помещается в контейнер x. Команда случайное также возвращает значение. В данном случае это будет число от 1 до 100. Как и в случае с предыдущей командой, выходное значение случайное также помещается в контейнер, имеющий имя r. Надо заметить, что контейнеры x и r нигде до этого в коде примера не использовались.

Упомянем и команды, которые ничего не принимают и ничего не возвращают. Вот несколько примеров:

```
очисти
перо_подними
обёртка_вкл
спрячь
```

Подсветка синтаксиса

Это особенность KTurtle позволяет сделать код более наглядным. С подсветкой синтаксиса весь код в редакторе выводится разными цветами, в зависимости от того, для чего предназначен тот или иной его кусок. В следующем списке вы найдёте описание разных типов кода и цветов, которые они получают в редакторе кода.

Таблица 5.1. Разные типы кода и их раскраска

Команды

тёмно-зелёный Обычные команды описаны здесь.

Контроллеры выполнения

чёрный (жирный) Специальные команды контроля выполнения, узнать больше можно здесь.

Комментарии

тёмно-жёлтый Строки комментария начинаются со знака комментария (#). Они игнорируются при выполнении программы. Комментарии необходимы для пояснения программистом того, что он делает в том или ином куске кода, а также для того, чтобы временно не выполнять какие-либо команды.

Скобки [,]

светло-зелёный (жирный) Скобки используются для группировки фрагмента программы. Зачастую скобки используются совместно с командами контроля выполнения.

Команда выучи

светло-зелёный (жирный) Команда выучи используется для создания новых команд.

Числа

голубой Числа..., да вроде бы говорить о них нечего.

Строки

тёмно-красный Единственное, что мы скажем о строках – они должны начинаться и заканчиваться двойными кавычками (").

Математические символы

серый Вот математические символы: +, -, *, /, (, и). Узнайте о них больше здесь.

Символы вопросов

голубой (жирный) Узнайте больше о символах вопросов здесь.

“Склеивающие слова” вопросов

розовый

Обычный текст чёрный

Пиксели

Пиксель – точка на экране. Если вы посмотрите на экран с очень близкого расстояния вы увидите, что ваш монитор использует пиксели. Пиксель – наименьшая частица, которая может быть нарисована на экране.

Множеству команд требуется количество пикселей в качестве входных параметров. Вот эти команды: вперёд, назад, иди, иди_гор, иди_верт, нов_размер_холста и нов_ширина_пера.

RGB комбинации (коды цветов)

RGB-комбинации используются для описания цветов. “R” отвечает за красный, “G” за зел`ный и “B” за синий цвета. Например, рассмотрим комбинацию 255,0,0: первое число, отвечающее за красный, равно 255, а два остальных равны 0, это говорит о том, что данная комбинация передаёт чистейший красный цвет. Каждая составляющая комбинации лежит в диапазоне от 0 до 255. Ниже приведен пример нескольких часто используемых цветов:

Таблица 5.2. Часто используемые RGB-комбинации

0,0,0 чёрный
255,255,255 белый
255,0,0 красный
150,0,0 тёмно-красный
0,255,0 зелёный
0,0,255 голубой
0,255,255 светло-голубой
255,0,255 розовый
255,255,0 жёлтый

Для легкого нахождения RGB комбинаций вы можете использовать диалог выбора цвета. Он вызывается из меню Сервис->Выбор цвета.

RGB комбинации в качестве входных параметров используются в двух командах: нов_цвет_холста и нов_цвет_пера.

Спрайт

Спрайт - это небольшая картинка, перемещаемая по экрану. Наша Черепашка, к слову, является спрайтом.

Примечание: в данной версии KTurtle спрайт не может быть заменён с Черепашки на что-либо другое. В следующих версиях вы обязательно сможете это делать.

Обёртка

Обёртка применяется тогда, когда Черепашка рисует что-то, не умещающееся на холсте (если при этом режим обёртки включен).

Пример обёртки. Когда Черепашка выйдет за границу холста, она немедленно перенесется на его противоположную сторону и сможет продолжать движение. Таким образом, Черепашка всегда будет оставаться на экране. Так будет, пока режим обертки останется включенным.

Включаться/выключаться он может командами **обёртка_вкл** и **обёртка_выкл**.

При запуске KTurtle режим обёртки включен по умолчанию.

Глава 6. Приложения

Рис. 1 (Рабочее поле KTurtle)

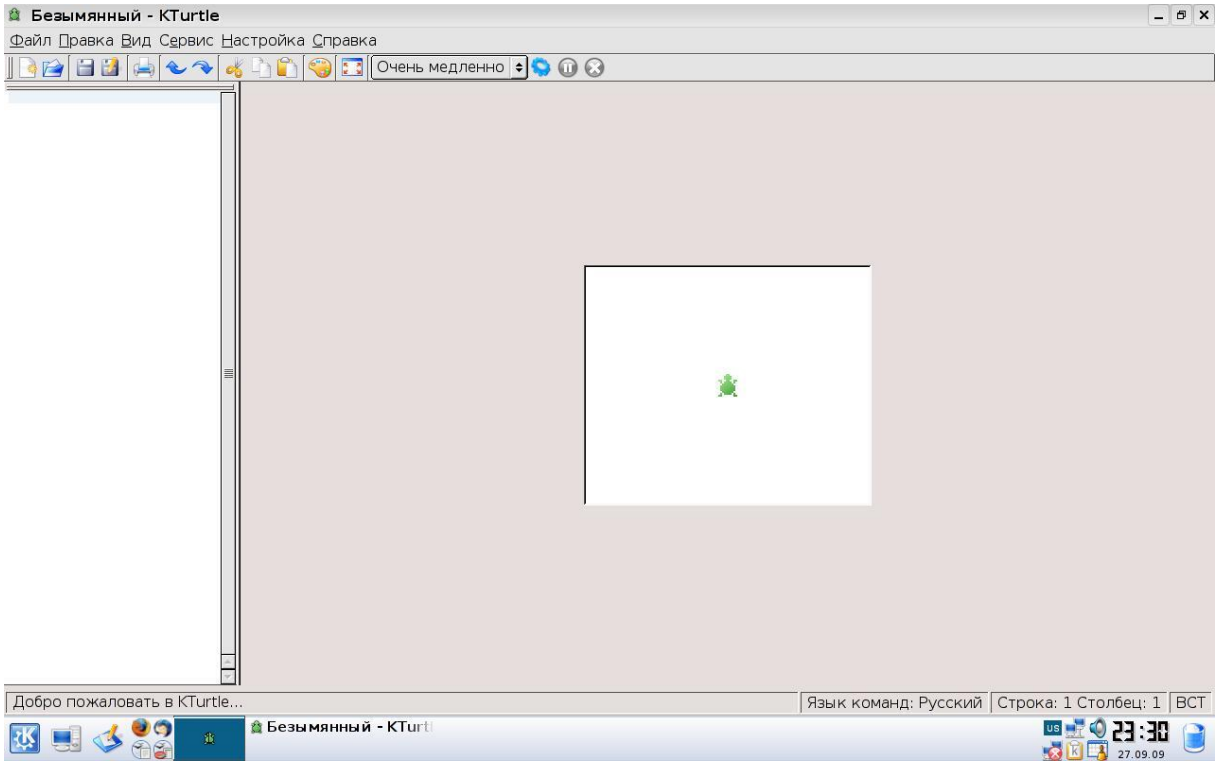

Рис. 2 (Координатная плоскость в KTurtle)

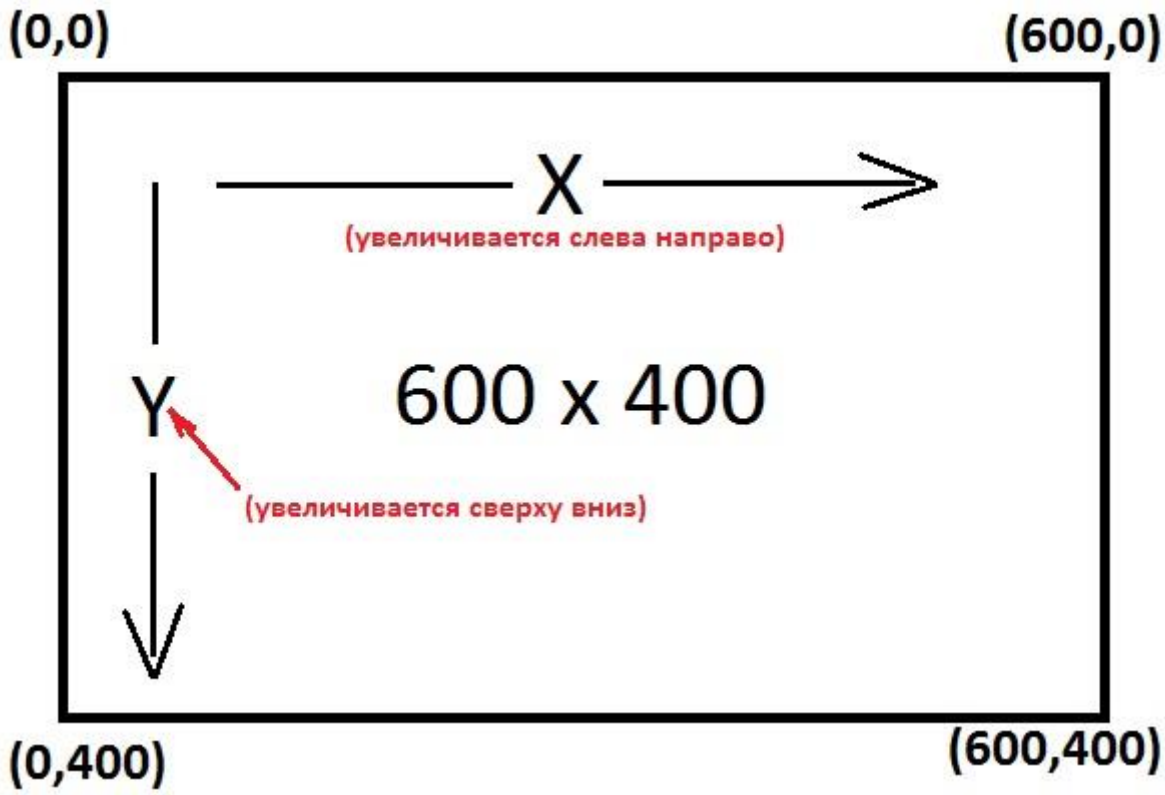

Рис. 3 («Направление» в KTurtle)