АЗЫ ПРОГРАММИРОВАНИЯ

(программируем на Logo и Python)

В.В.Михайлова

(ГОУ СОШ № 572 г.Москвы)

2008 г.

Содержание

Данное пособие является копией дистанционного курса, разработанного в среде Moodle, для учащихся 5 — 7 классов. Курс включает в себя изучение основ языков программирования Logo и Python и является факультативным.

В курсе предусмотрена преемственность между изучением указанных языков программирования, используются однотипные задачи. Основной упор пока делается на задачи, предполагающие рисование различных фигур. Математическая часть курса находится в стадии разработки.

Курс начинается с работы в программе Kturtle, включенную в ОС Linux. При этом особое внимание, в отличии от ЛогоМиров, уделяется не зрелищности и анимационным вопросам, а именно терминологии и грамотному написанию программ.

В дальнейшем изучаются основы языка Python. При этом основной упор делается на библиотеку turtle. Однако, дети знакомятся с такими серьезными понятиями, как переменная, параметр, цикл, функция и т.д.

Работа над курсом только начата. Из предполагаемых 34 уроков написаны пока только 13. Но, уже после их прохождения учащиеся могут писать довольно интересные и сложные программы.

В своей работе я использовала:

Книги

1. А.Г.Юдина «Практикум по информатике в среде LogoWriter», Москва, 1999 г.

2. А.Н.Чаплыгин «Учимся программировать вместе с Питоном».

Сайты:

1. http://myrobot.ru/logo

2. http://www.python.org/doc/2.5.2/lib/module-turtle.html

3. http://www.python.ru/

4. http://hostinfo.ru/articles/web/rubric48/rubric55/rubric59/1358/

5. http://sleepy.cs.surrey.sfu.ca/cmpt/courses/cmpt120/notes/

Ч А С Т Ь 1.

Программируем на языке Logo

- **"Лого - это философия образования и непрерывно развивающееся семейство языков, реализующих эту философию."**
  Harold Abelson, Apple Logo, 1982.

Приглашаю вас изучить основы языка Logo при помощи программы KTurtle

НИКОГДА НЕ ЗАБЫВАЙ СОХРАНЯТЬ ФАЙЛЫ И ВО ВРЕМЯ РАБОТЫ НАЖИМАТЬ CTRL+S!!!

Урок 1. Квадраты.

Привет! Меня зовут Черепашка.

Начнем! Запускай KTurle. Слева окно Кода программы, справа на холсте - исполнитель, то есть я, черепашка.
Давай, напишем нашу первую программу для рисования ***квадрата***.
Сначала мы будем писать короткие программы, а потом - длинные.
Чтобы не забыть, что мы хотели запрограммировать, в программу нужно включать *комментарии*, то есть пояснения. Установи курсор мыши в окне Кода программы. Нажми CTRL+D - команда для вставки комментария. Видишь, появился значок комментария #. Строка с таким значком не является исполняемой командой. Напиши комментарий:
Квадрат. По умолчанию, я всегда стою в центре холста. Для того, чтобы не делать ошибок, заглядывай в справочник. На каждой строке должна размещаться одна команда. Для удобства, ты можешь пронумеровать строки - F11. А теперь набери код программы. Пусть это бдет квадрат 100 на 100. пикселей. 1 *черепаший шаг равен одному пикселю.* Попробуй сначала сам, а потом проверь себя:

```
# Квадрат
вп 100
пр 90
вп 100
пр 90
вп 100
пр 90
вп 100
пр 90
```

Согласись, что программа выглядит довольно некрасиво. В ней 4 раза повторяется одно и тоже. Такой повтор одних и тех же действий называется циклом. Для записи цикла у меня есть команда *повтори*. Код цикла заключается в квадратные скобки. Теперь наша программа будет

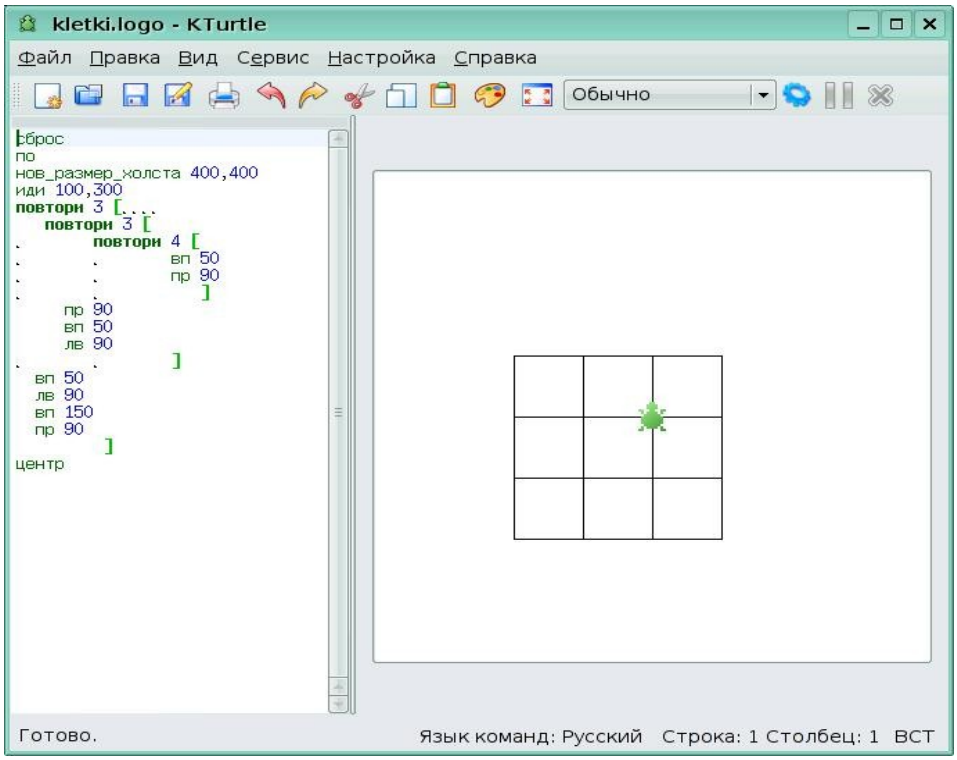
выглядеть так:

```
# Квадрат
сброс
повтори 4 [
вперёд 100
направо 90
]
```

Правда, красивее? Заметь, удобно начинать программу со слова "сброс", чтобы автоматически очищать экран перед следующим запуском программы.

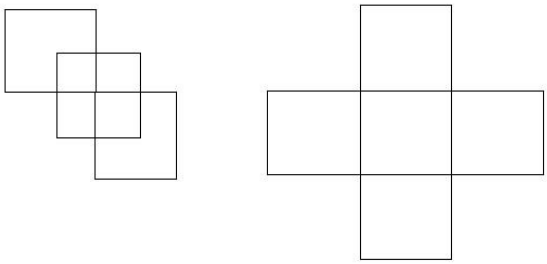
Для выполнения программы нужно нажать ALT+ENTER или синюю шестеренку на панели инструментов. Но, мне кажется, что ALT+ENTER удобнее. Обрати внимание, ты можешь задать разную скорость выполнения программы. Медленный режим хорош для *отладки* программы. При выполнении программы в медленном режиме, в окне Кода подсвечиваются выполняемые в данный момент команды. Понаблюдай это, задав медленную скорость. На панели инструментов есть кнопка "пауза" и кнопка "стоп", которая прекращает выполнение программы.

А теперь задание посложнее. Нарисуй дорожку из трех квадратов. Получилось? Тогда, составь программу для рисования решетки. Проверь себя:



Задание 1.

Составь программы для рисования следующих фигур и пришли на проверку.

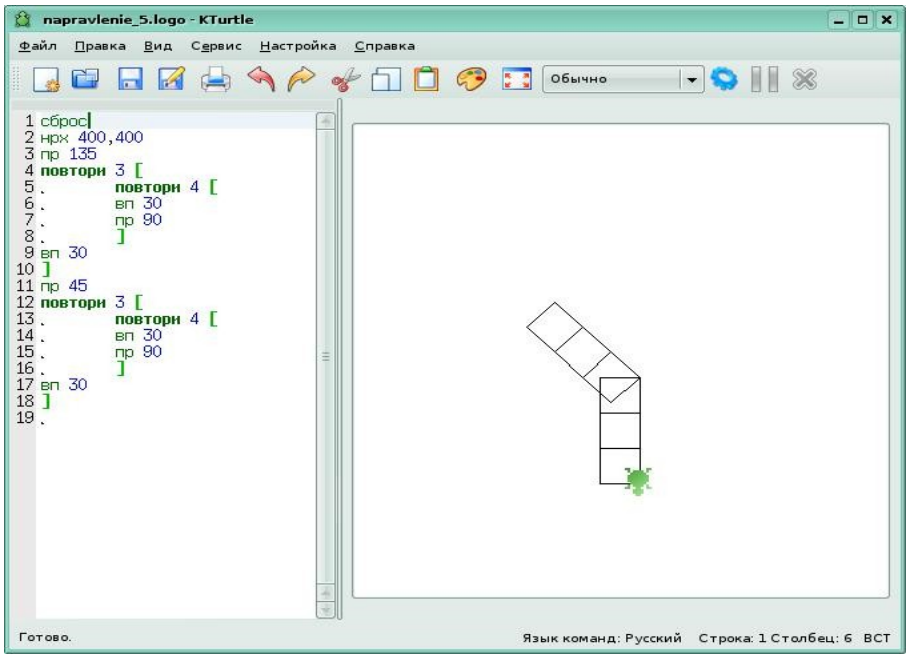


Урок 2. Опятьквадраты.

Привет! Ты успешно справился с первым заданием? Тогда двинемся дальше.

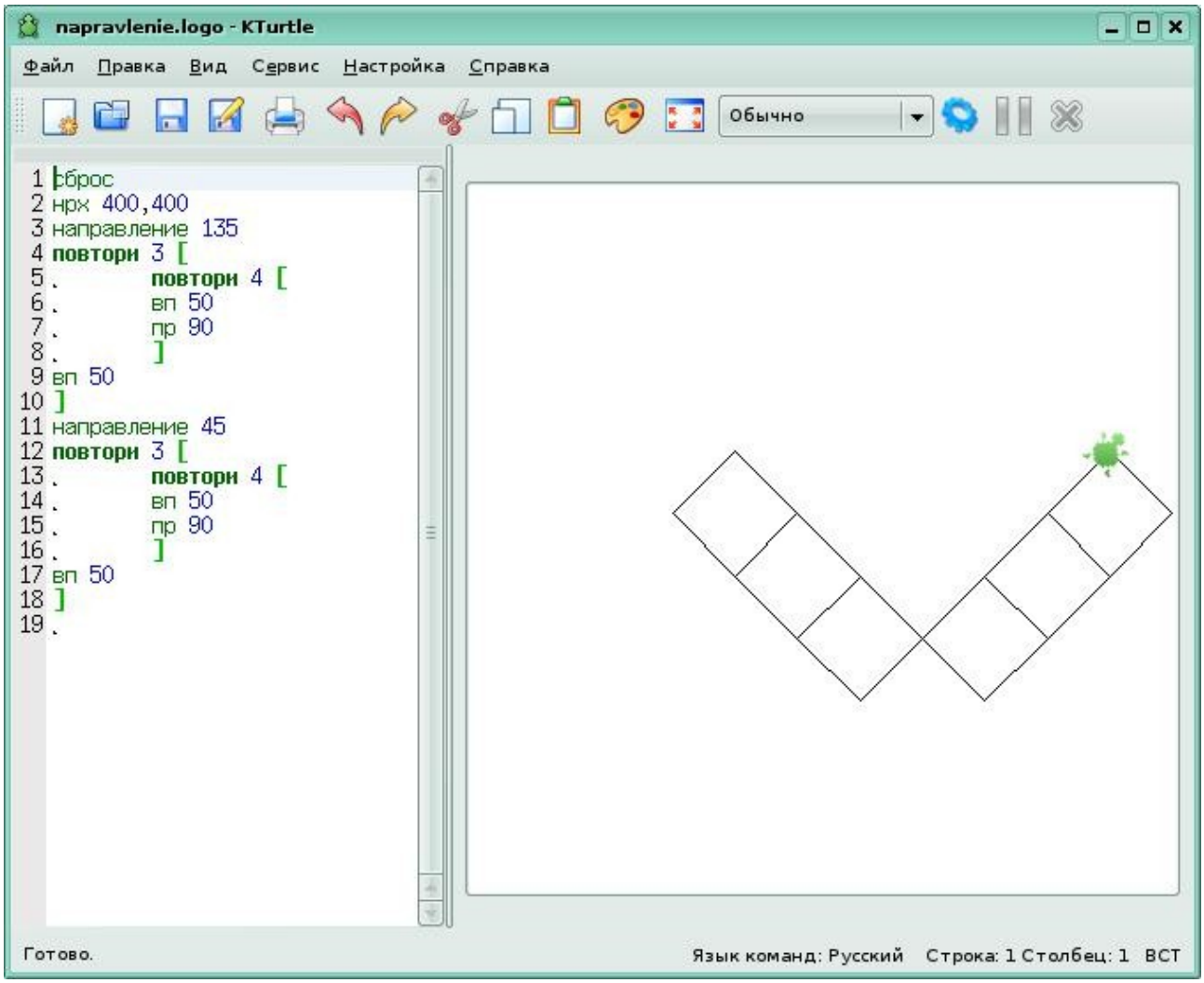
Давай подробнее посмотрим на команды *иди* и *направление*. Чем они отличаются от команд *направо*, *налево* и *вперед*, *назад*?

Команды направо и налево поворачивают черепашку на определенное количество градусов относительно ее текущего положения. Набери код и проследи на медленной скорости за действиями черепашки:



Черепашка стояла как обычно и по команде *пр 135* повернулась относительно своей оси на 135 градусов.

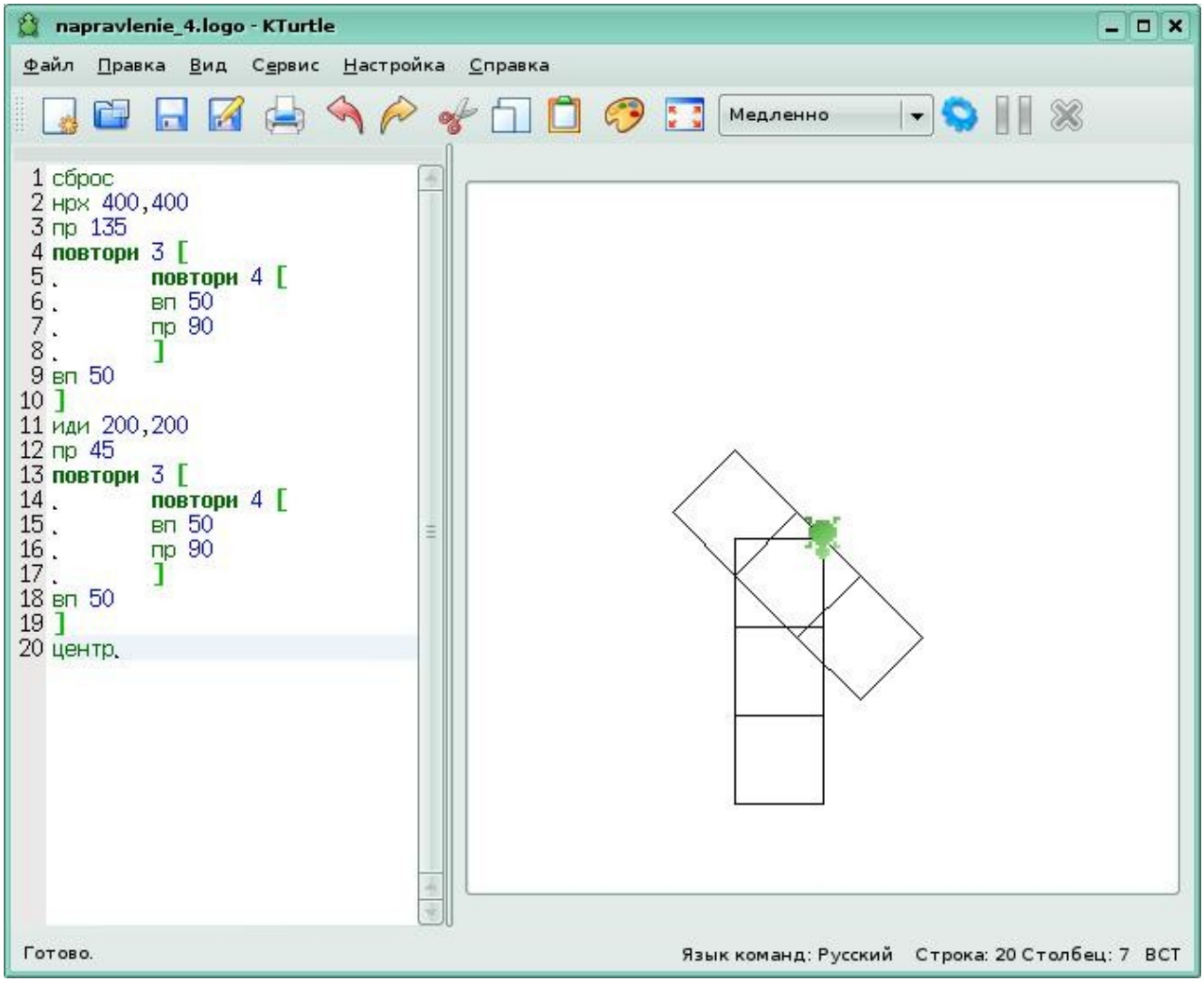
Посмотрим другой пример.



Как видишь, команда направление работает иначе. Она всегда поворачивает черепашку на определенное количество градусов относительно условного первоначального положения, то есть того, когда черепашка стоит головой вверх. В этом примере черепашка сначала повернулась на 135 градусов, а перед рисованием второй дорожки повернулась на 45 градусов, представляя, что она стоит в условно первоначальном положении.

Теперь поговорим о команде *иди*. Размер холста определяется в пикселях. Командой *нрх x,y* ты можешь задать новый размер холста. Ширина его будет x пикселей, а длина y пикселей. Нулевой точкой считается верхний левый угол холста. Команда *иди x,y* отправляет черепашку в определенную точку холста, координаты которой ты задал. Поэксперементируй с этой командой.

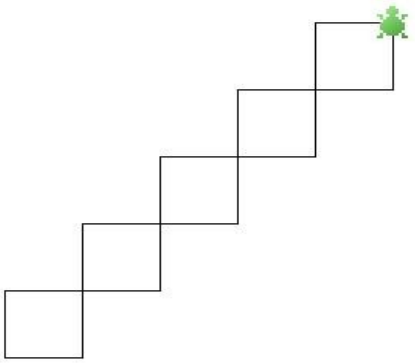
Посмотри пример.



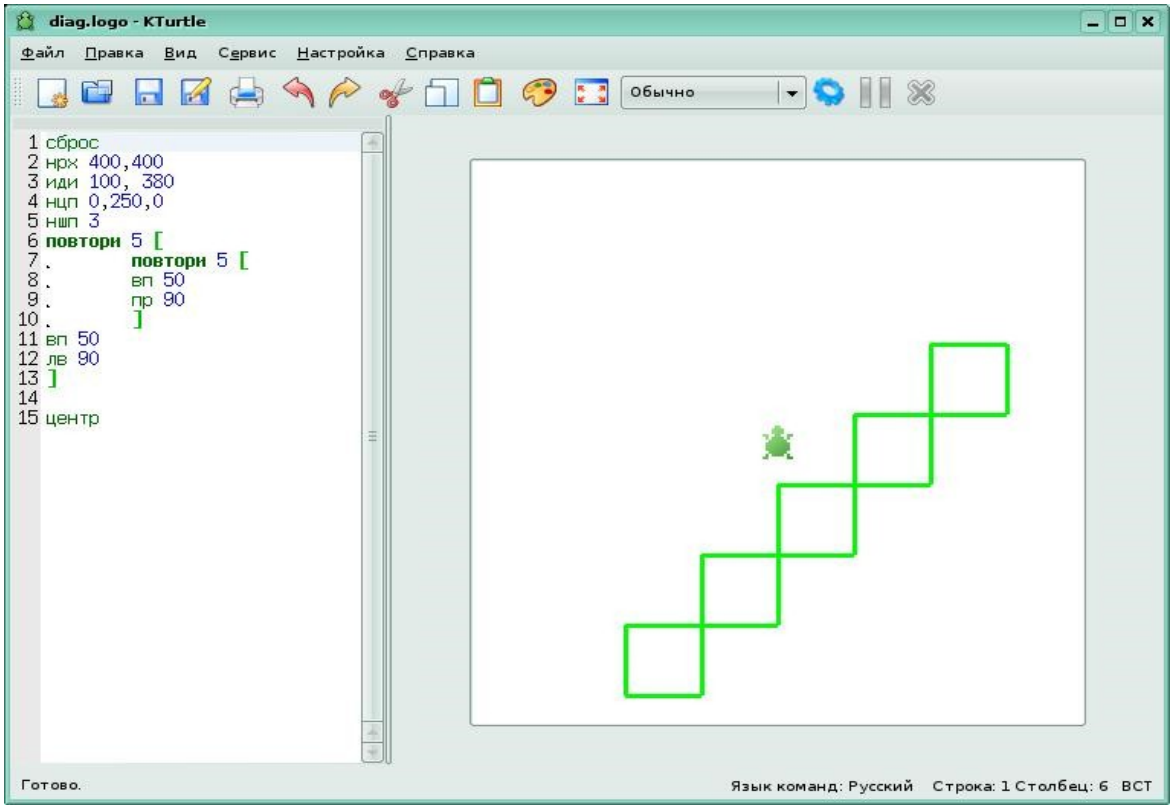
Размер холста 400x400 пикселей, значит команда *иди 200,200* отправит черепашку в середину холста.

Ты заметил, что во всех предыдущих примерах применялась конструкция Цикл - в - Цикле? Это очень удобно! Только не забывай сосчитать - количество открывающих и закрывающих скобок должно быть одинаково.

Попробуй написать код программы для рисования следующей картинки. Используй команды иди, направление и конструкцию Цикл-в Цикле.

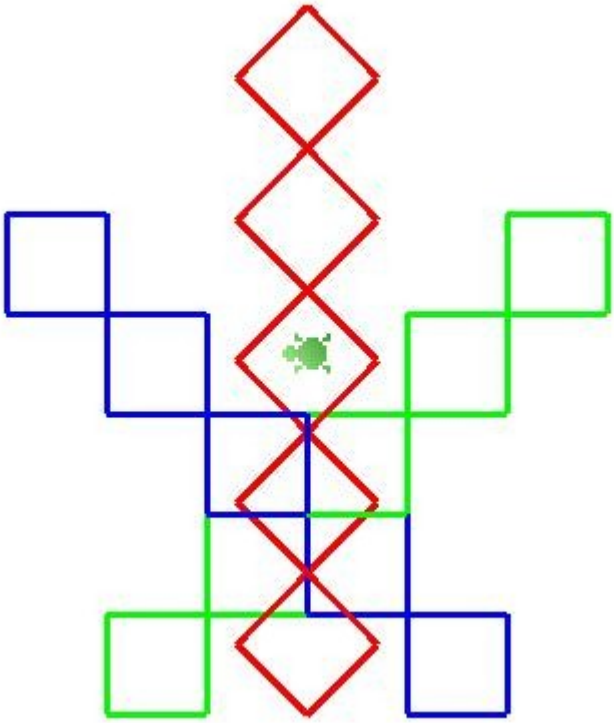


Проверь себя.



Задание 2.

Составь программы для рисования следующих фигур и пришли на проверку.

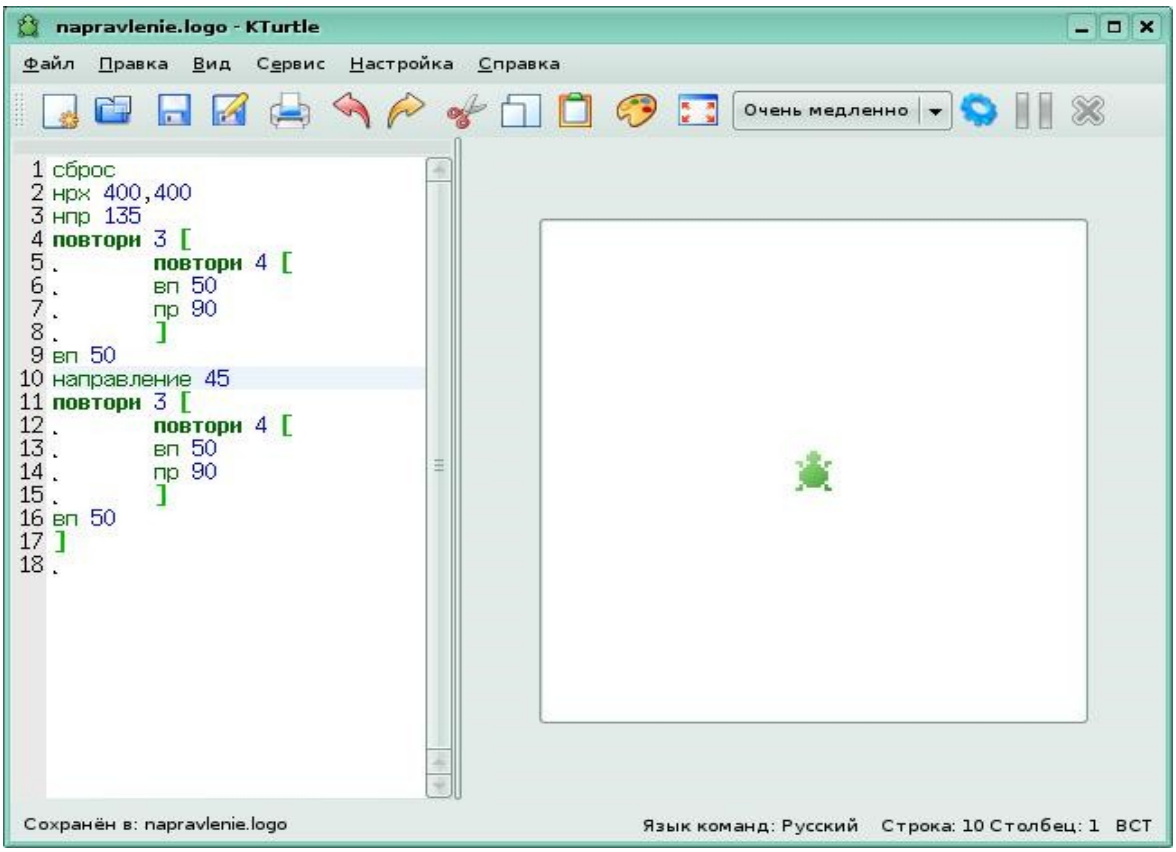


Урок 3. Как распознать ошибку.

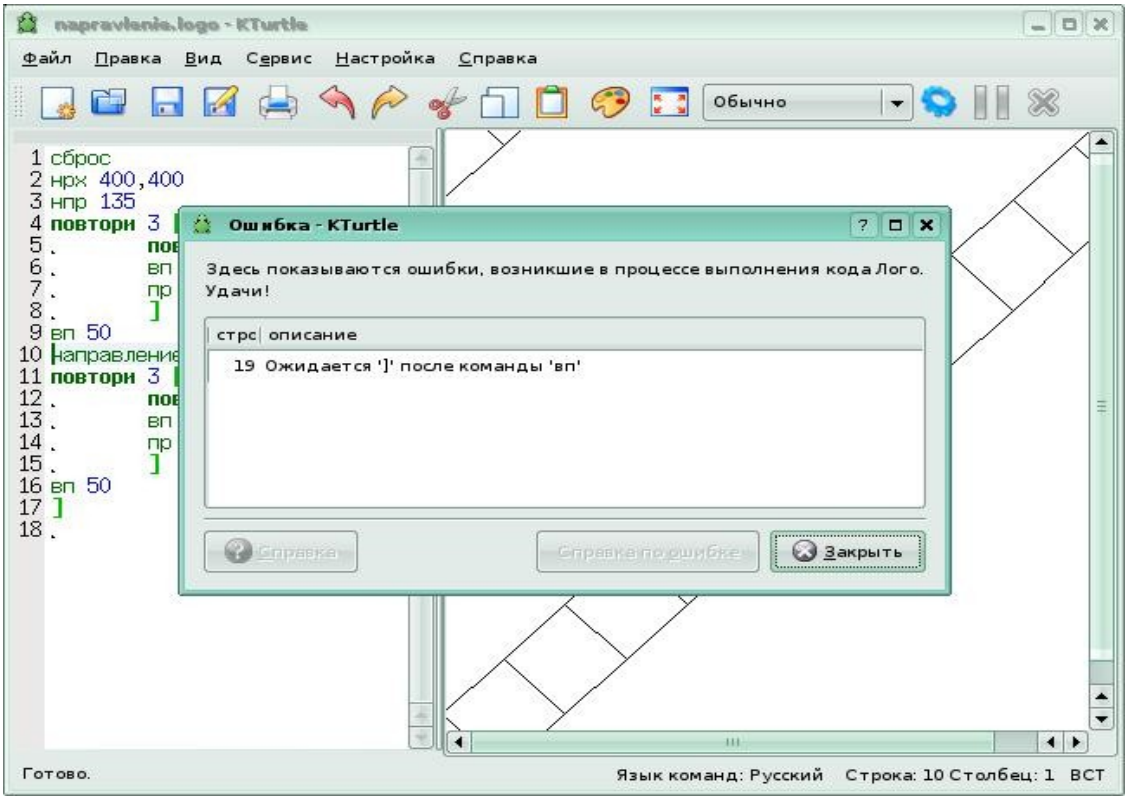
Как часто учитель слышит "Я все сделал правильно, а программа не работает!" Как же распознать ошибку? Поговорим об этом.

Программирование - довольно сложный процесс, и вполне естественно, когда программист допускает ошибку. Так повелось, что программные ошибки называют "багами" (от англ. bug - жучок). В сленге российских программистов чаще встречается слово "глюк". Процесс поиска и устранения ошибок мы будем называть отладкой.

Код программы должен быть записан предельно точно и только на языке команд исполнителя. В противном случае я подам сигнал об ошибке. Посмотри на следующий пример:

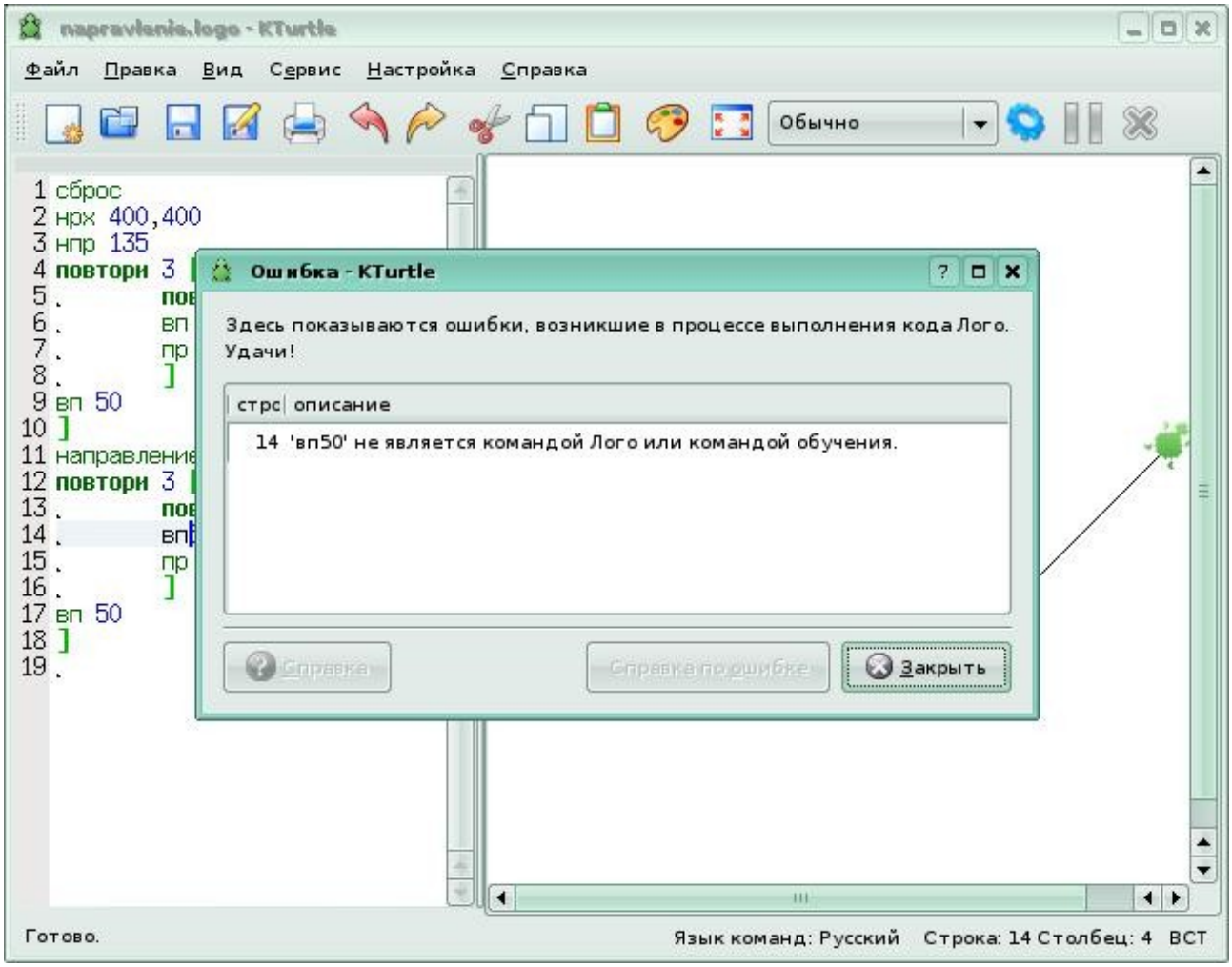


Как ты думаешь есть в коде программы ошибки? Если объем программы довольно большой, найти ошибки крайне трудно. Помогает пошаговое (очень медленное выполнение) программы и так называемые сообщения об ошибках. Попробуй выполнить эту программу и вот что ты получишь:



Как видишь, один из циклов не закрыт. В сообщении указывается, что не хватает одной скобки и даже сказано после какой команды ее недостает, а строка в которой есть ошибка подсвечивается.

Еще один пример.

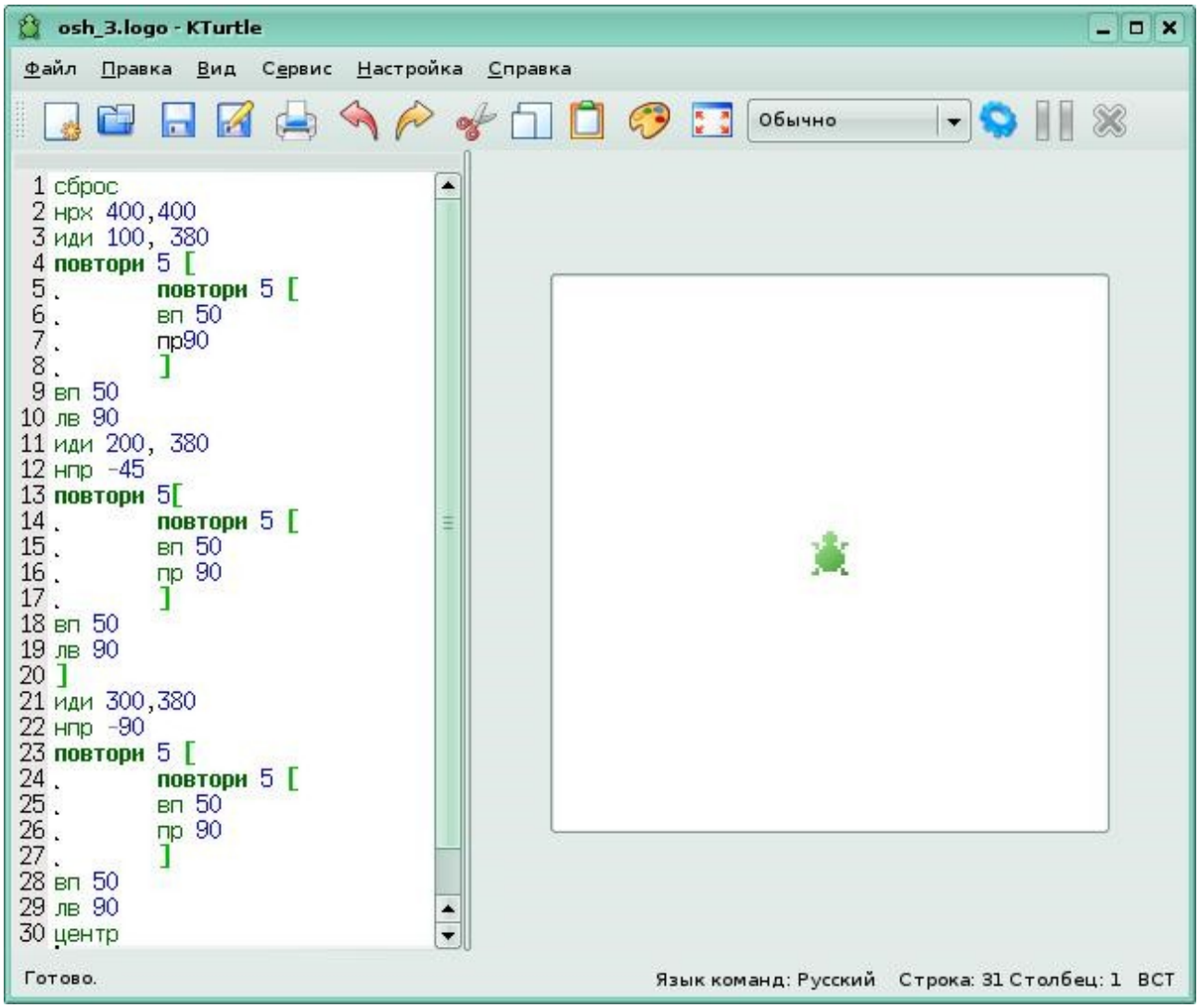


Сразу видно, что ошибка в 14 строке - не хватает пробела между вп и 50.

Попробуй сам сделать ошибки в своих программах и посмотреть, какое сообщение будет выдавать черепашка.

Задание 3.

Найди ошибки в коде программы и пришли на проверку в виде текстового файла.



Урок 4. Команда «выучи».

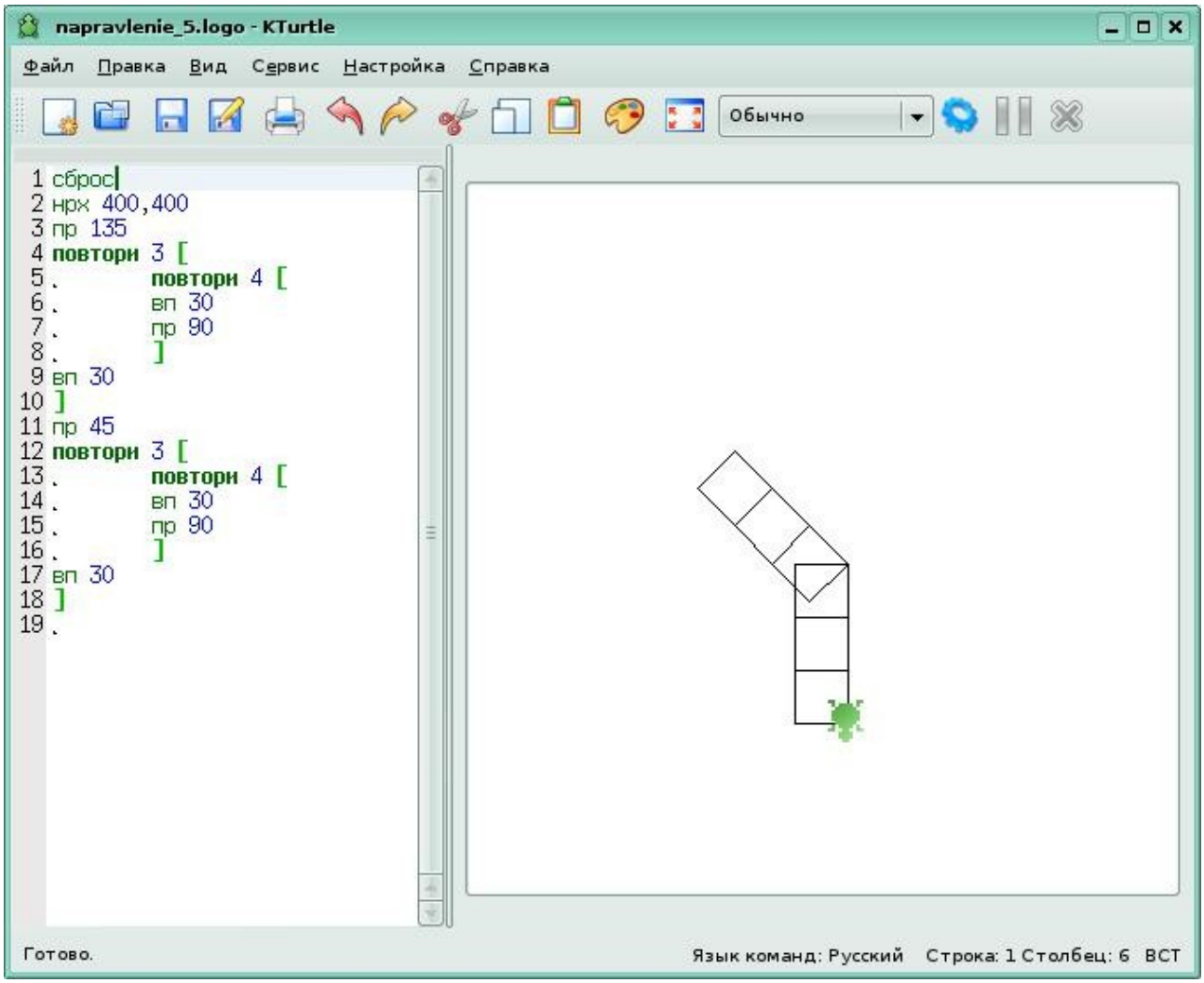
Выучи - особенная команда, потому что она предназначена для создания так называемых подпрограмм или процедур, то есть ваших собственных команд. Давай посмотрим как это делается.

Уже давно программистами было замечено, что некоторые части программы можно использовать более одного раза, если изменить значение какого-то параметра (например, длину стороны квадрата).

Бывает, что в программе может несколько раз вычисляться значение по определенной формуле, но каждый раз значения, подставляемые в формулу, меняются.

Программисты стали обобщать решения небольших повторяющихся подзадач и обращаться к частям программы с этими решениями там, где это было нужно.

Посмотри на программу из урока 3.



Чтобы нарисовать несколько квадратов, мы два раза повторили один и тот же фрагмент программы. Можно сделать проще.
Использовать команду выучи.
Зададим Черепашке такую программу:

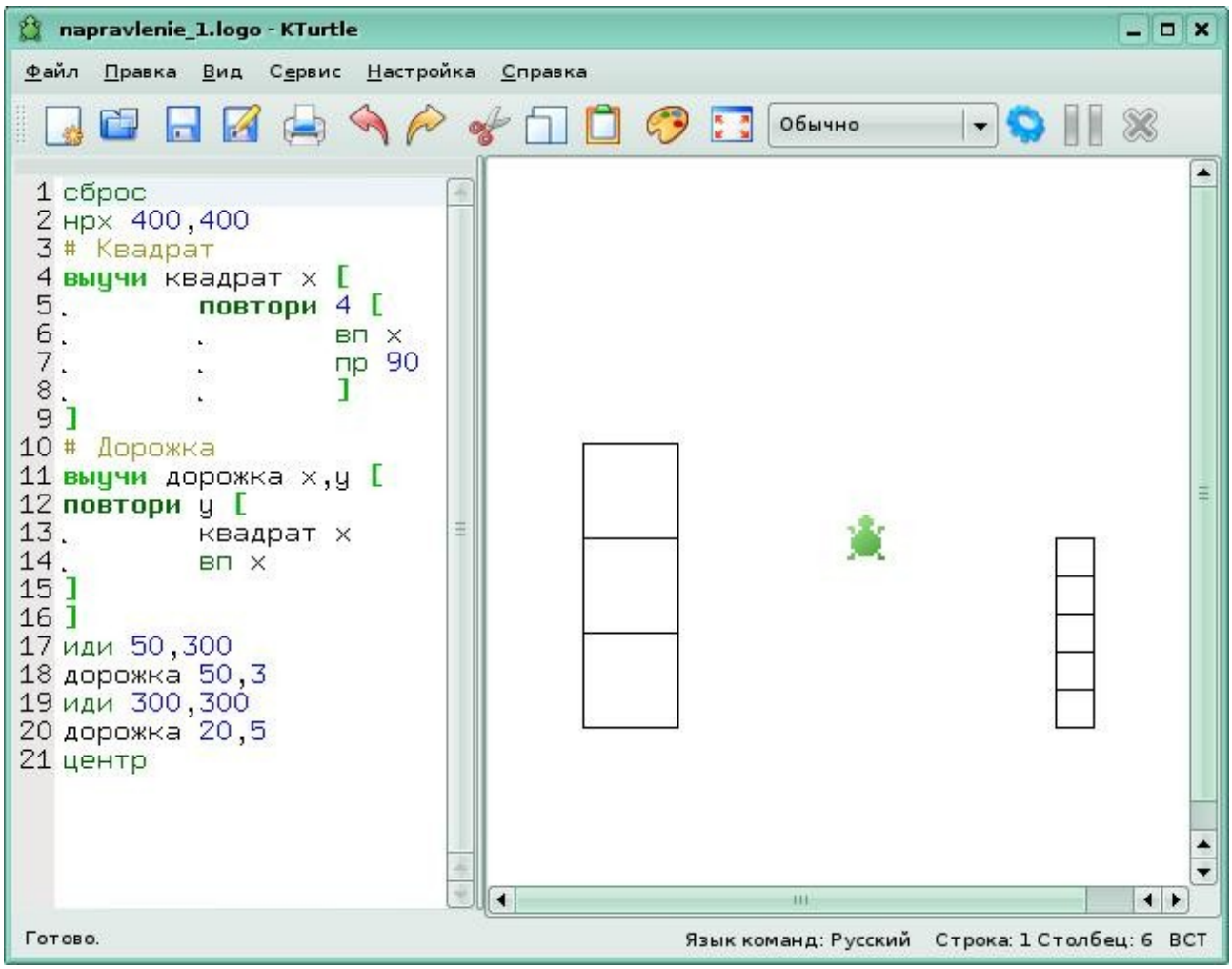
```
# Квадрат
выучи квадрат 50 [
повтори 4 [
вп 50
пр 90
]
]
```

Теперь ты можешь отправить Черепашку в любую точку холста и задать

ей команду квадрат 50. Черепашка сразу нарисует квадрат со стороной 50. Также ты можешь создать подпрограмму "Дорожка" и нарисовать дорожку из нескольких квадратов.

Если же ты вместо конкретного указания количества шагов поставишь ***параметр*** (то есть переменную), то сможешь рисовать квадраты различной величины.

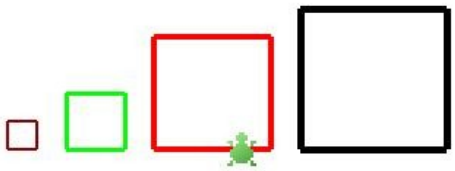
Рассмотри следующий пример:

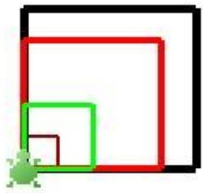


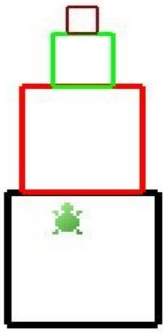
Ты видишь здесь две *процедуры с параметром*. Одна позволяет нарисовать квадрат. Чтобы задать длину стороны квадрата, надо вместо x поставить число шагов. Вторая рисует дорожку. В ней два параметра. Один задает длину стороны квадрата, другой - количество квадратов.

Задание 4.

Попробуй, используя команду "выучи", составить программы для следующих рисунков и пришли их на проверку.







Урок 5. Переменные и их типы.

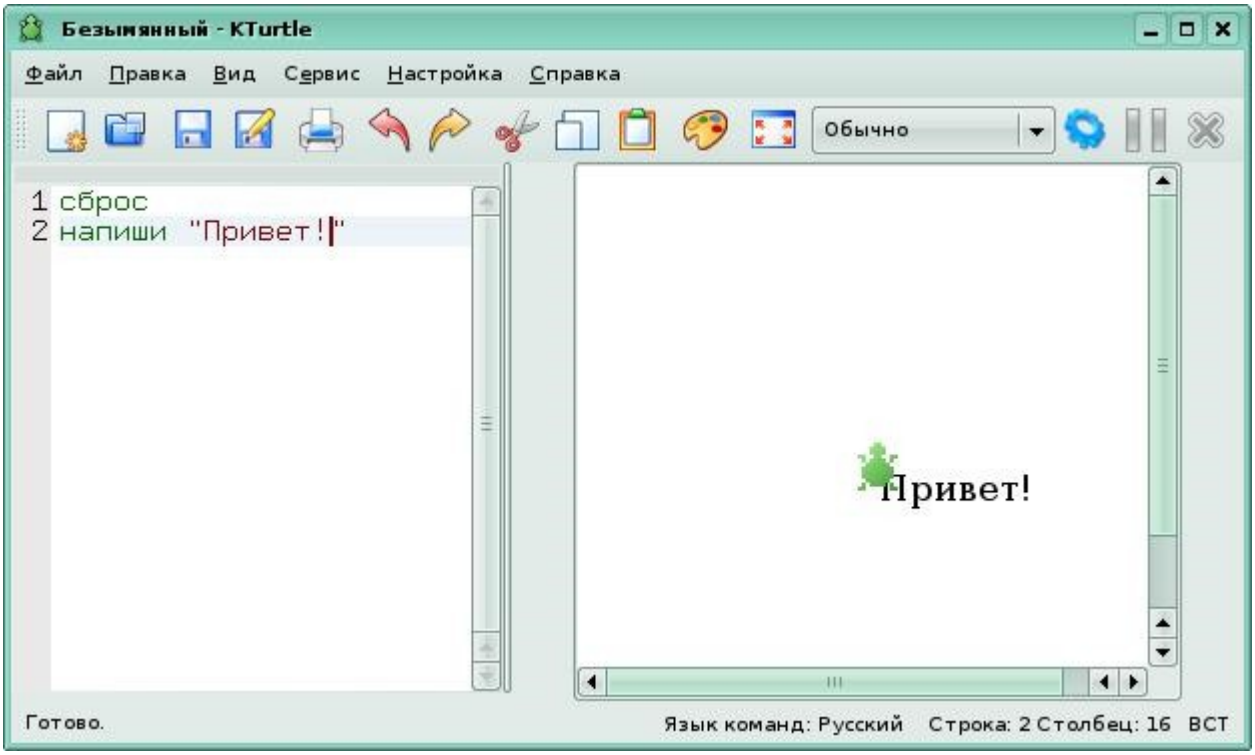
Все программы работают с переменными. Переменной может быть число или строка.

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. В KTurtle переменные делятся на числа и строки.

Числа, используемые в 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.

У Черепашки есть команда ***напиши***. Например, если ты дашь команду

напиши "Привет!", то значением переменной этой команды будет ***строка***, которая должна заключаться в кавычки.



Используя команду новый_размер_шрифта x, ты можешь задать новый размер шрифта в пикселях.

Для присваивания значений переменным в Logo служит знак "=". Это не тот знак "равно", к которому мы привыкли. В языках программирования этот знак читается "присвоить значение".
При этом значением может быть как число, так и строка.

Например, выражение **a = 5** следует читать как "присвоить переменной **a** значение 5. Или, выражение **имя = "Вера"** означает, что переменной "имя" было присвоено значение "Вера".

При этом после вычисления или выполнения какой-то операции, значение переменной может меняться, а предыдущее значение будет стерто из памяти.

Имена переменных не могут совпадать с именами команд. Нельзя, например, назвать переменную "вперёд", потому что у Черепашки есть такая команда.

Имена переменных могут состоять из букв, цифр и символа

подчеркивания (_), но первой всегда должна быть буква.

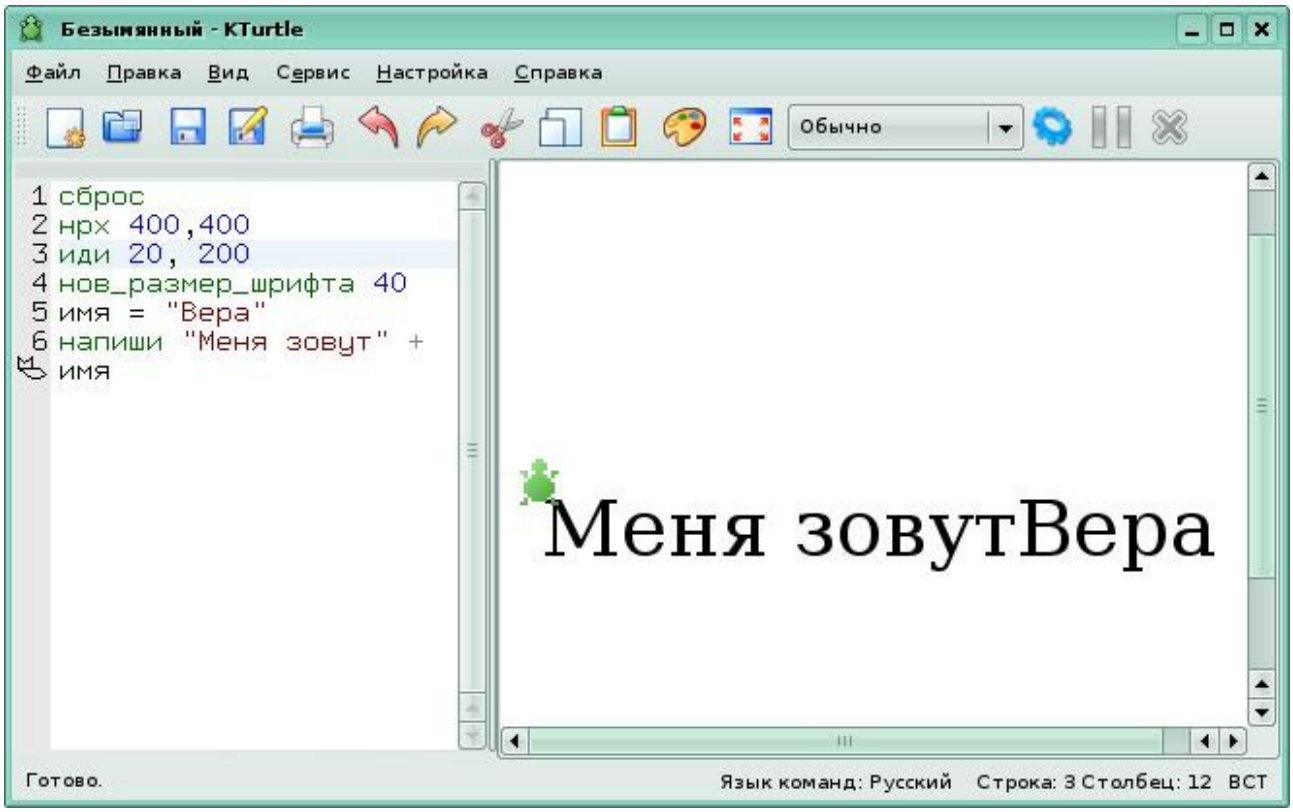
Рассмотрим некоторые операции со строковыми переменными.

Такой операцией является общение с Черепашкой через диалоги.

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

Рассмотрим отличия этих команд.

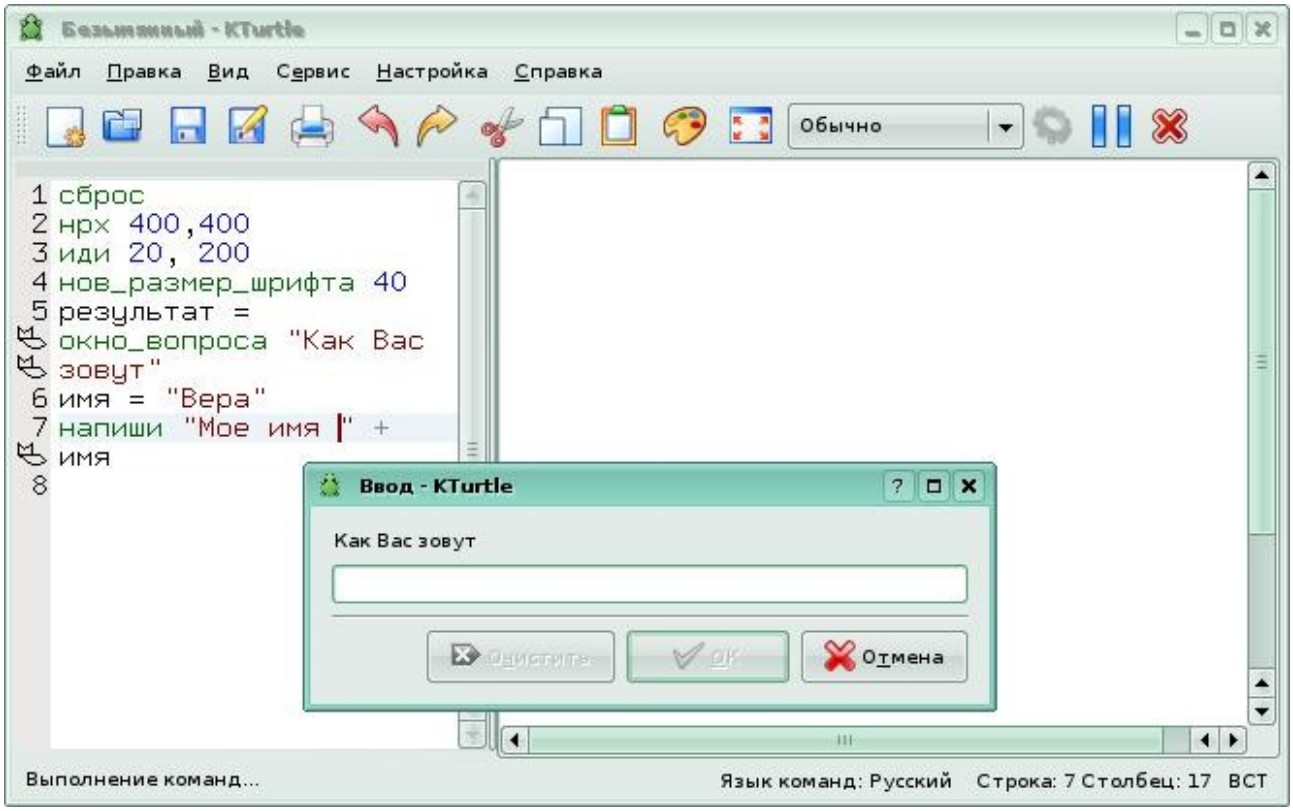
Сначала как выглядит сообщение.



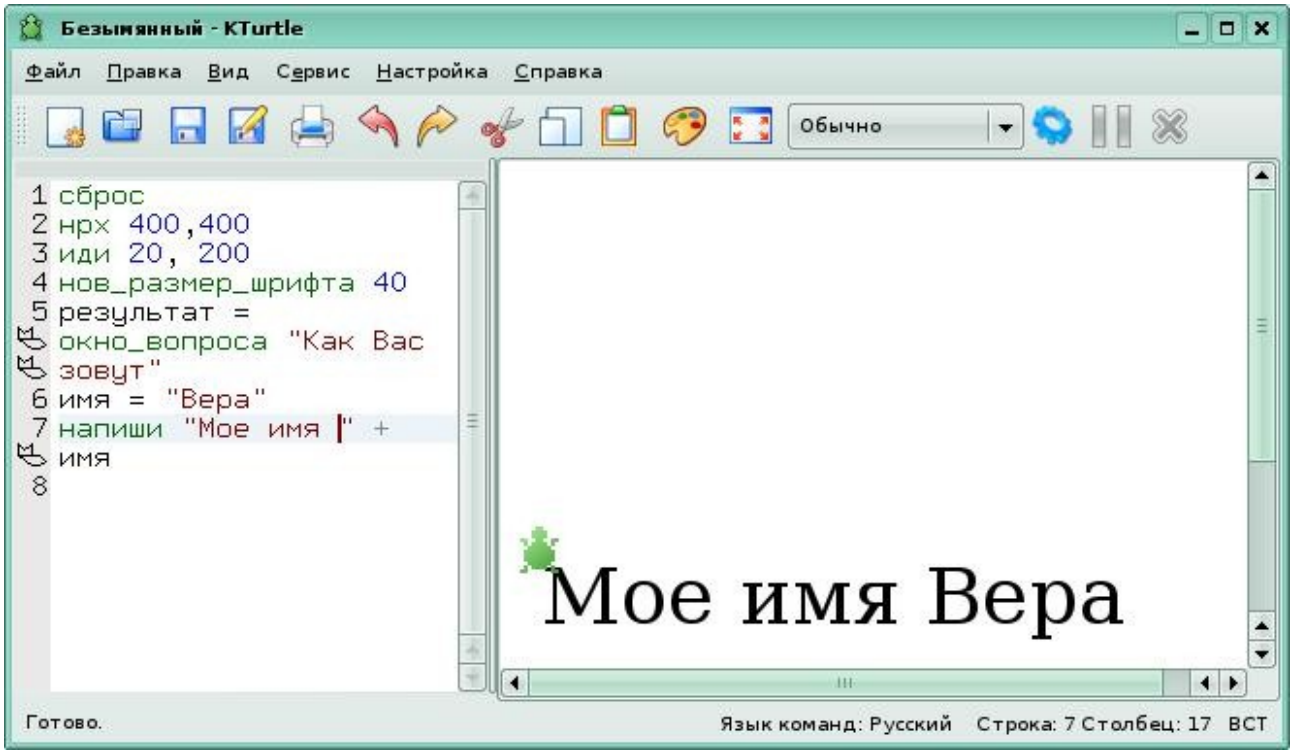
4 строка устанавливает новый размер шрифта.
В 5 строке переменной имя присваивается строковое значение "Вера".
В 6 строке к надписи "Меня зовут" прибавляется значение переменной, которое мы лекго можем поменять, не затрагивая структуры самой программы.

Теперь разберемся с командой ***окно_вопроса***.
Команде окно_вопроса требуется передать строку, она будет показана в появившемся окне, аналогично команде сообщение. Но, кроме этого, в

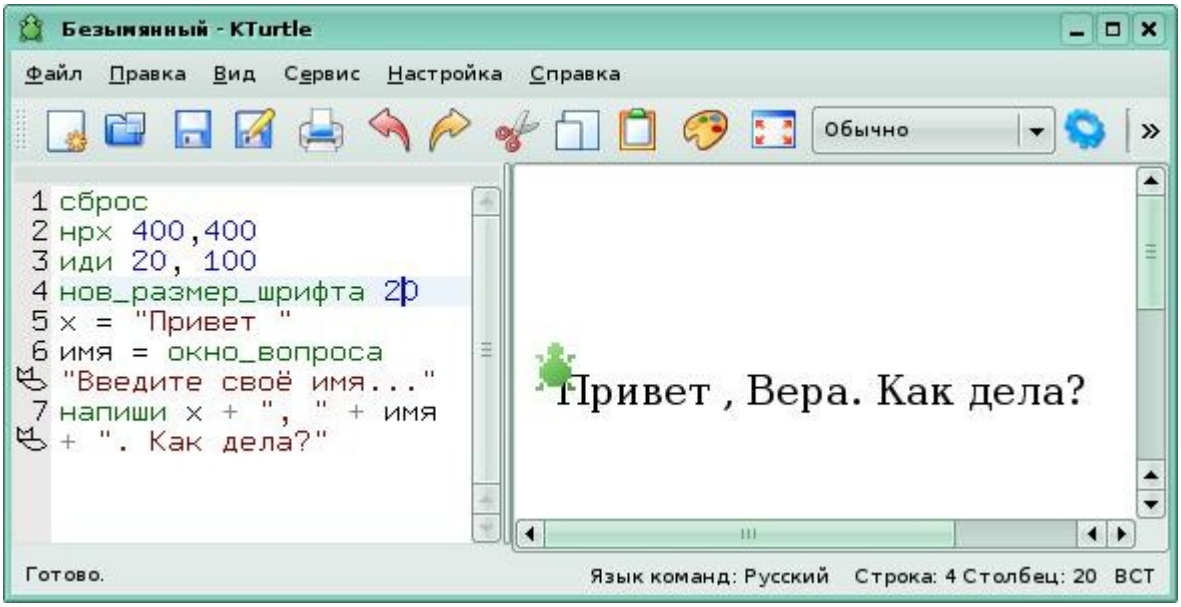
окне будет поле для ввода числа или текста. Например,



После ввода имени, Черепашка выведет его на экран.



Рассмотрим еще один пример.



Здесь, в 5 строке переменной х присваивается строковое значение "Привет", затем выводится окно диалога, в которое ты записываешь свое имя. В 6 строке показано, что вместо переменной "имя" будет внесено, то строковое значение, которое ты ввел в окне диалога. И,наконец, в 7 строке будет сформирована надпись. Удобство в том, что не меняя тела программы, ты можешь заменить значение переменной х на любое другое (например, "Пока!"), а в окне диалога ввести любое имя. Потренируйся!

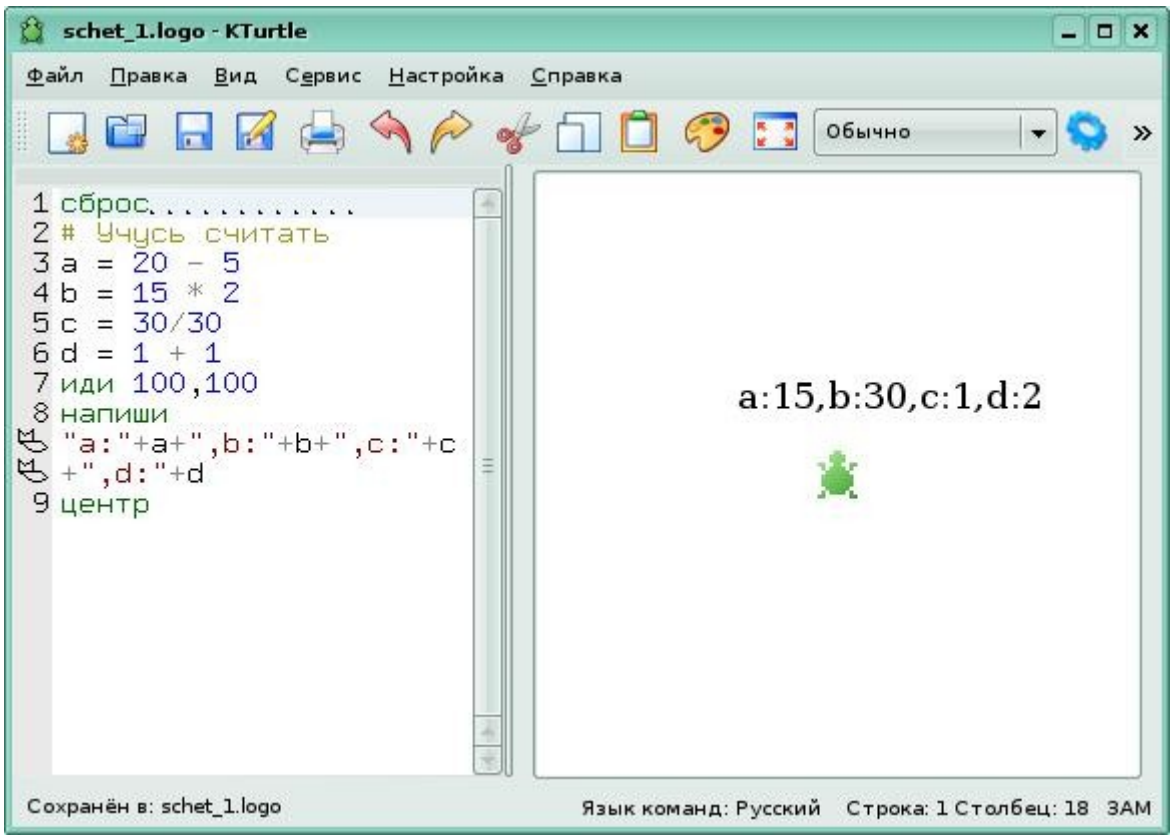
Задание 5.

Создай программу, которая будет содержать не менее трех окон диалога, и пришли ее на проверку.

Урок 6. Учим черепашку считать.

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

Черепашке под силу заниматься математическими вычислениями. Рассмотрим следующий пример:



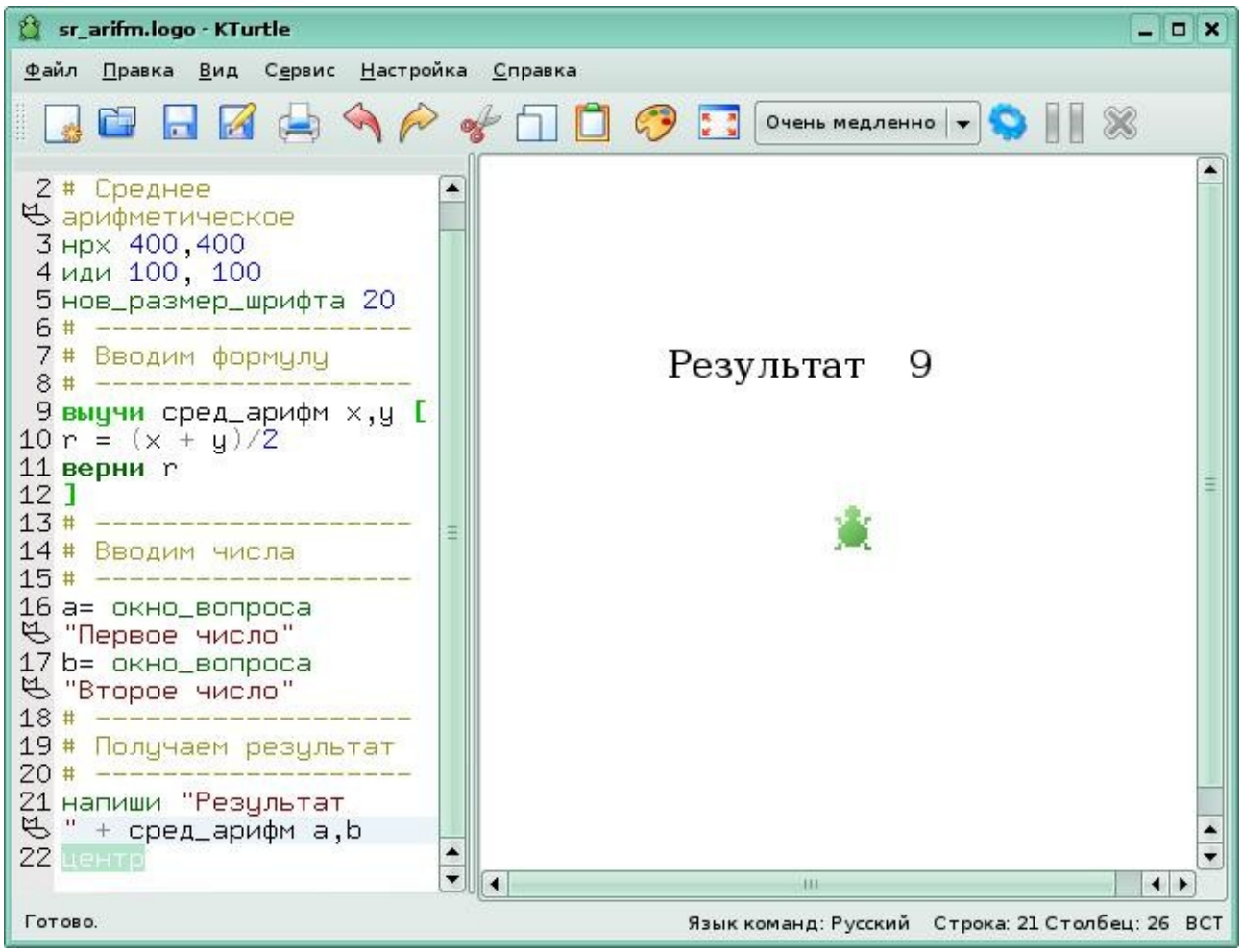
Еще раз напомню, что оператор "=" не следует понимать как равенство.

Выражение a = 20 - 5 следует читать так: вычесть значение 5 из 20, результат присвоить переменной a (записать в ячейку памяти для переменной a). Попробуй сам прочитать выражения в строках 4,5,6. В строке 8 мы даем черепашке команду вывести на экран результаты вычислений.

Если в программе нужно вычислить значение простого выражения можно поступить так:
напиши ((20 - 5) * 2/30) + 1

Выражение в скобках будет вычислено первым. Черепашка соблюдает приоритет математических операций.

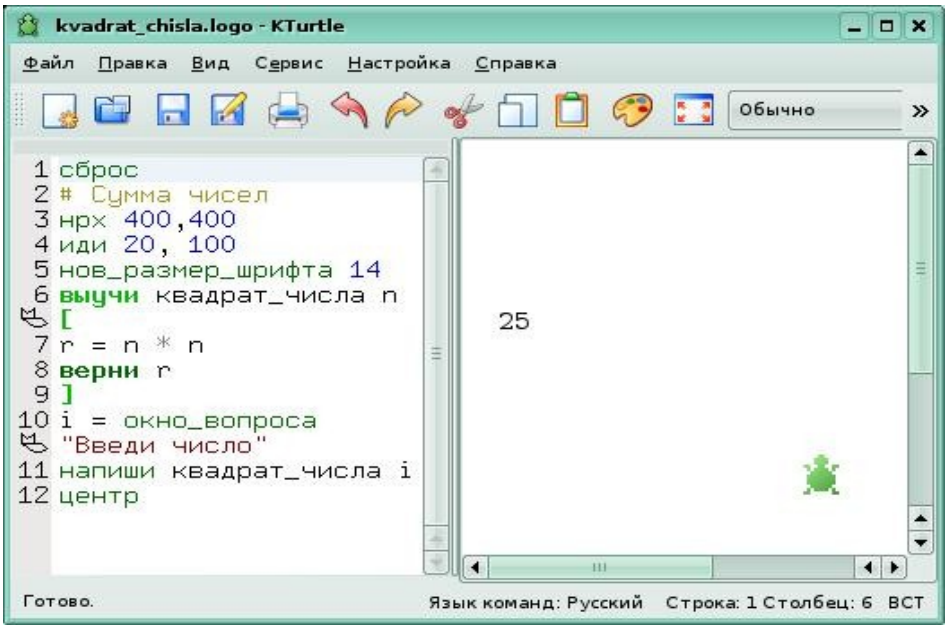
Давай научим черепашку находить среднее арифметическое двух чисел. Для этого нам понадобится формула для нахождения среднего арифметического, записанная на "черепашьем" языке: r = (a + b)/2, где a и b - некоторые числа. Воспользовавшись переменными, создадим программу, которая позволит нам вводить любые значения a и b.



Итак, после задания размера холста, отправки черепашки в удобное для записи место и установки шрифта, мы задаем непосредственно формулу для вычисления среднего арифметического. Эта процедура с параметрами. То есть, мы пишем формулу в общем виде, а числа подставлять будем потом.

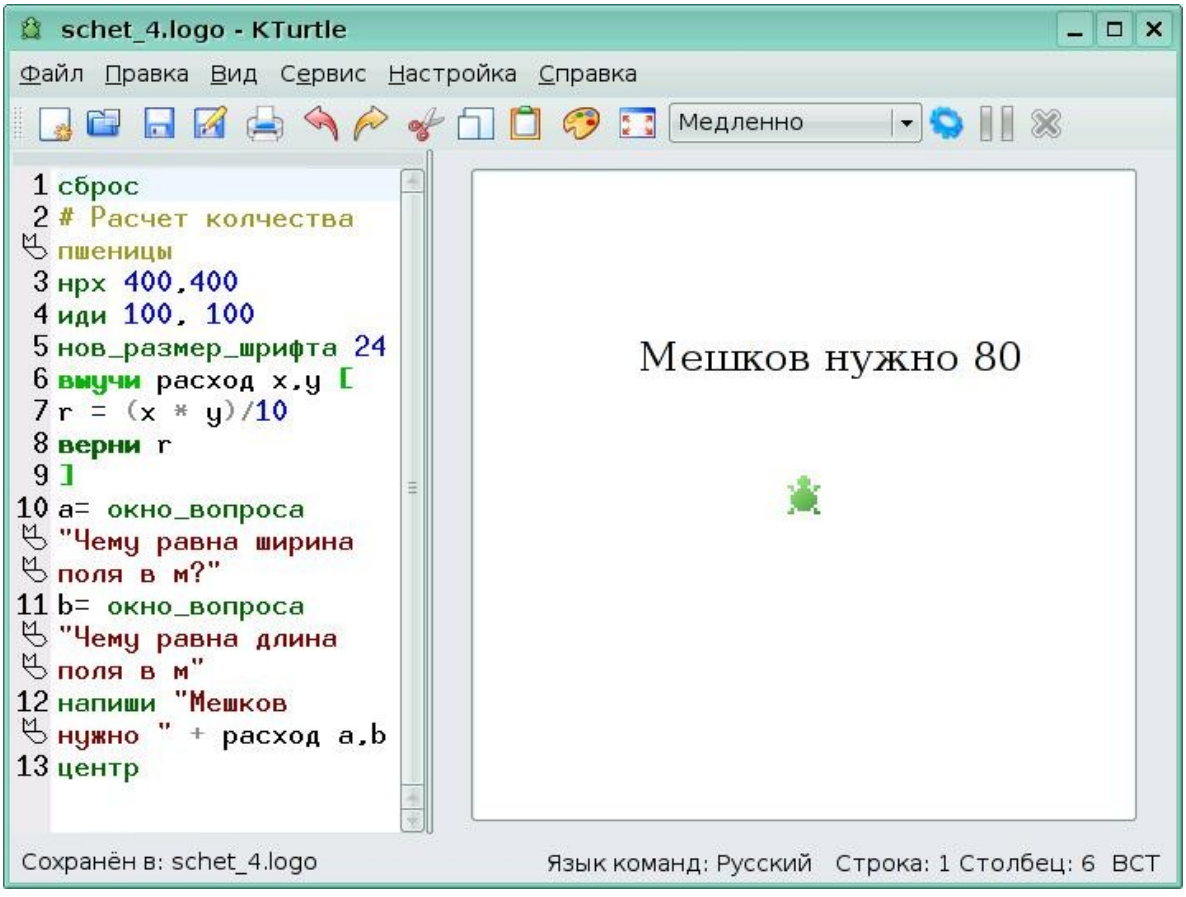
Мы говорили о том, что оператор присваивания передает ячейке памяти новое значение. Так вот в 11 строке команда "верни" напротив возвращает в ячейку памяти найденное ранее значение переменной r. После этого при помощи окон диалога мы вводим нужные числа, а черепашка находит нам результат.

Попробуй написать программу для вычисления квадрата числа.
Проверь.

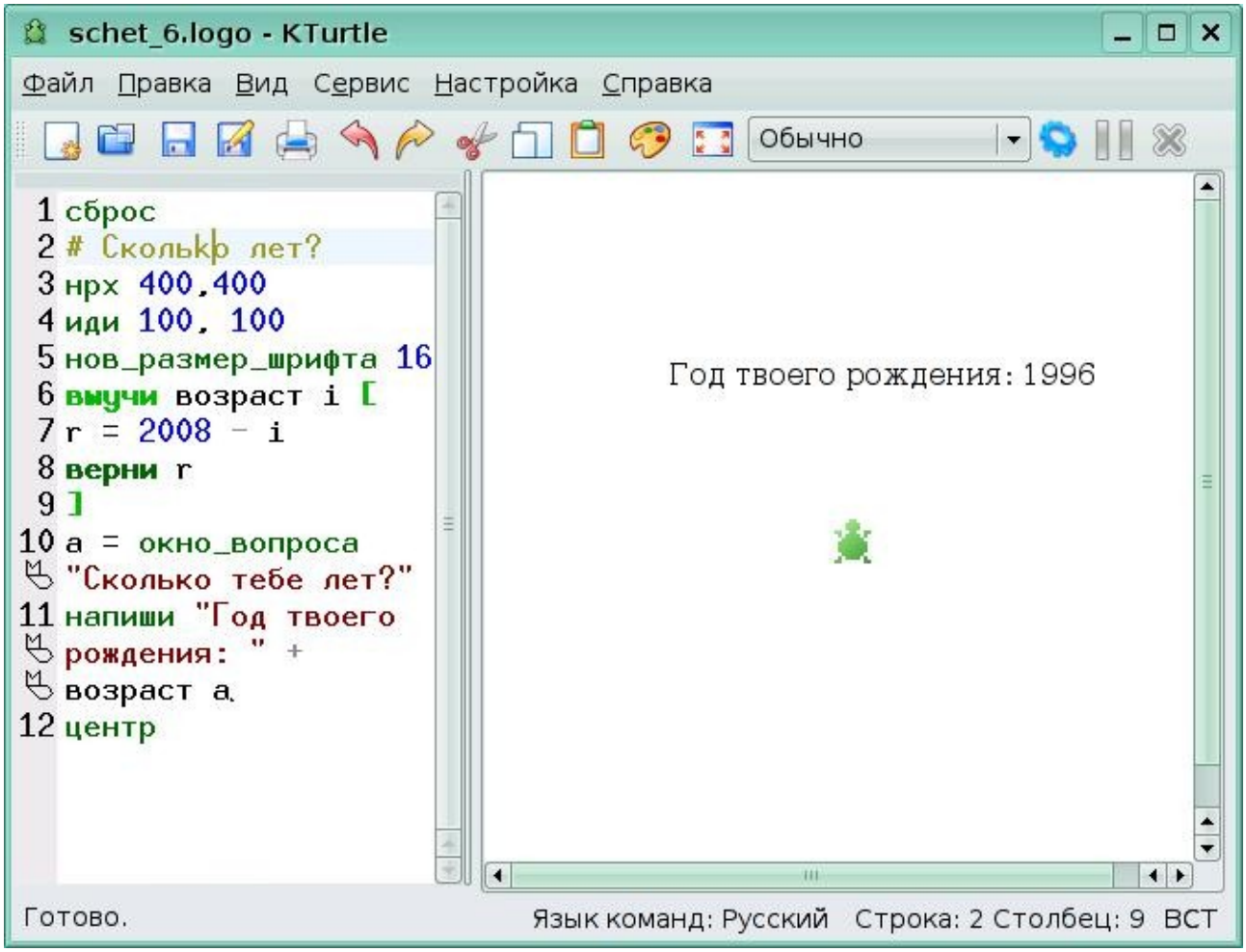


Рассмотри еще одну программу.

Вводимые параметры - длина и ширна поля. Фиксированное число - 10 кг пшеницы в мешке на 1 кв.м поля. Расчет - необходимое количество мешков. Подумай, какие комментарии ты добавил бы в программу.



Напиши программу, которая по введенному тобой в диалоговое окно возрасту, вычисляет год рождения. Проверь себя.



Задание 6.

Создай программу для расчета необходимого количества рулонов обоев для поклейки комнаты. Ширина рулона - 60 см. Результат пришли на проверку.

Урок 7. Подробнее о циклах. Цикл со счетчиком.

Одно из ценнейших свойств компьютера, как и любого автомата, - способность многократно повторять одни и те же действия.

Многократное повторение последовательности каких-либо действий в программе называют циклом, а саму последовательность действий - телом цикла.

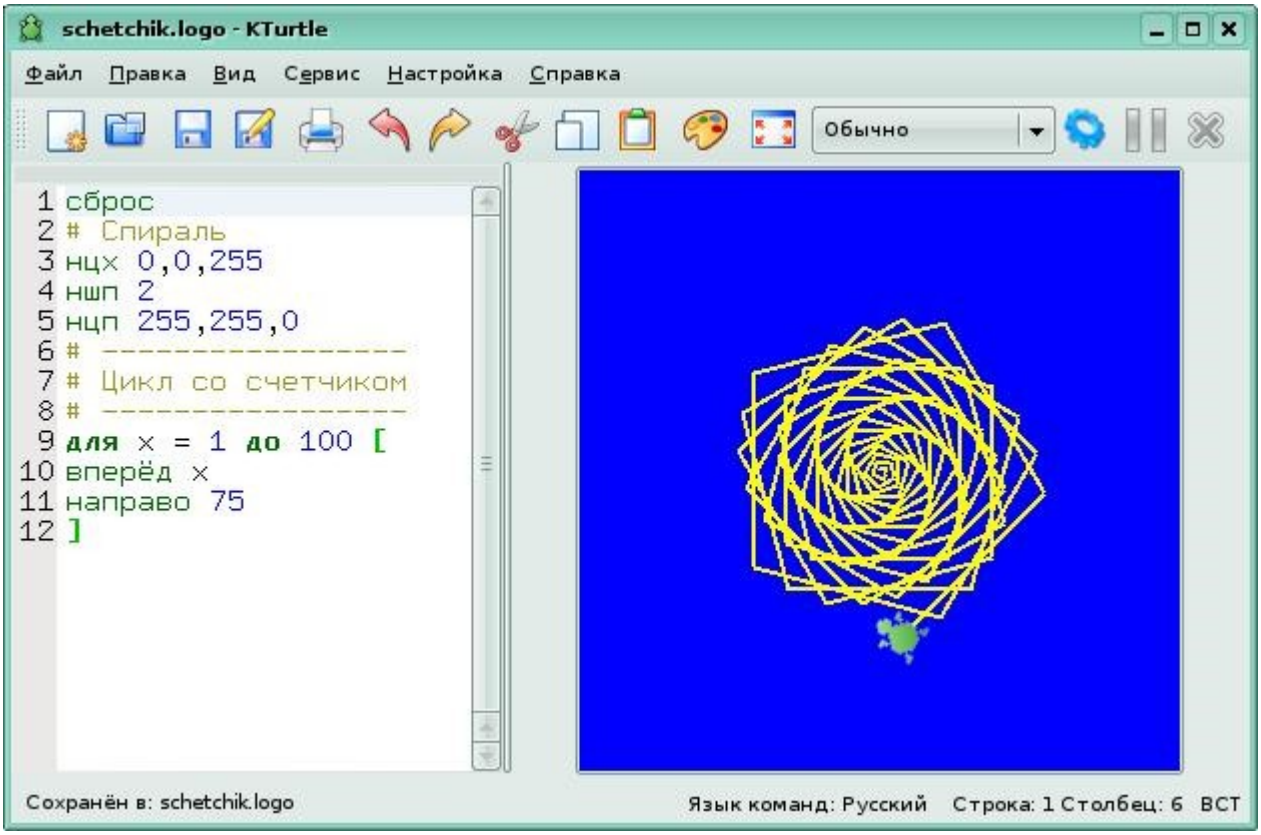
Процессы, при реализации которых повторяется выполнение одних и тех же действий, называют циклическими.

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.

Цикл со счетчиком

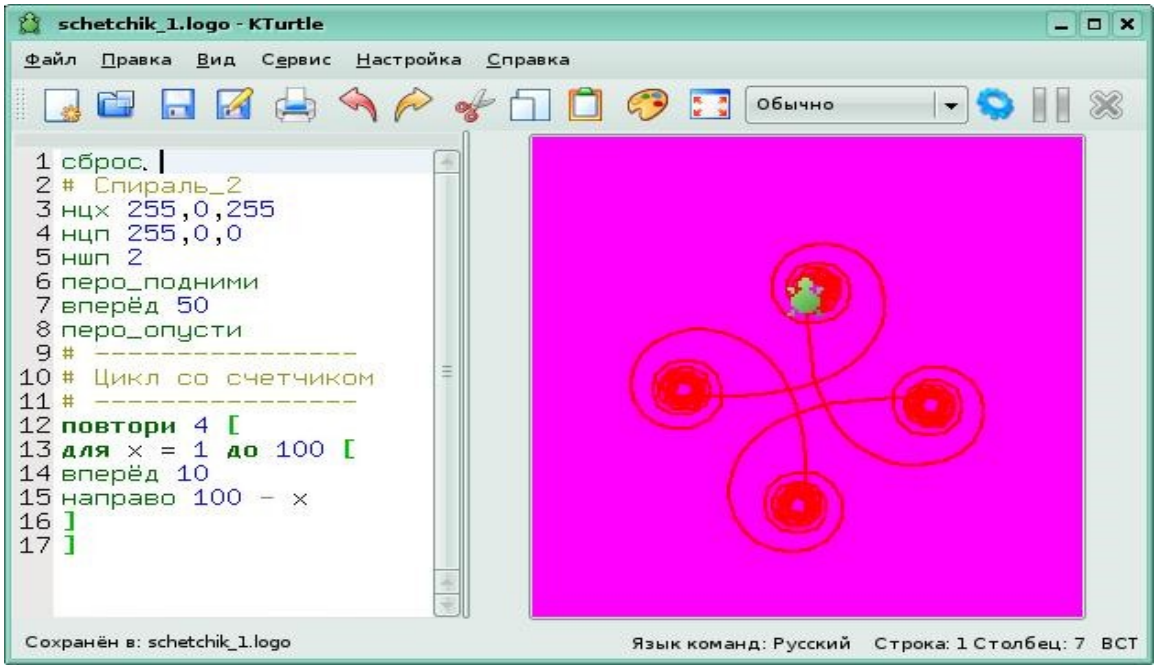
Представим себе, что нам необходимо нарисовать спираль. Наша спираль должна будет раскручиваться из центра, все более и более увеличиваясь в размерах. При этом угол, на который должна поворачиваться черепашка после того, как нарисует очередной сегмент спирали, будет все время одинаковым. А вот для того, чтобы спираль "раскручивалась", черепашке нужно будет каждый раз идти вперед хотя бы на шаг больше, чем в предыдущий.

Мы можем написать следующую программу:

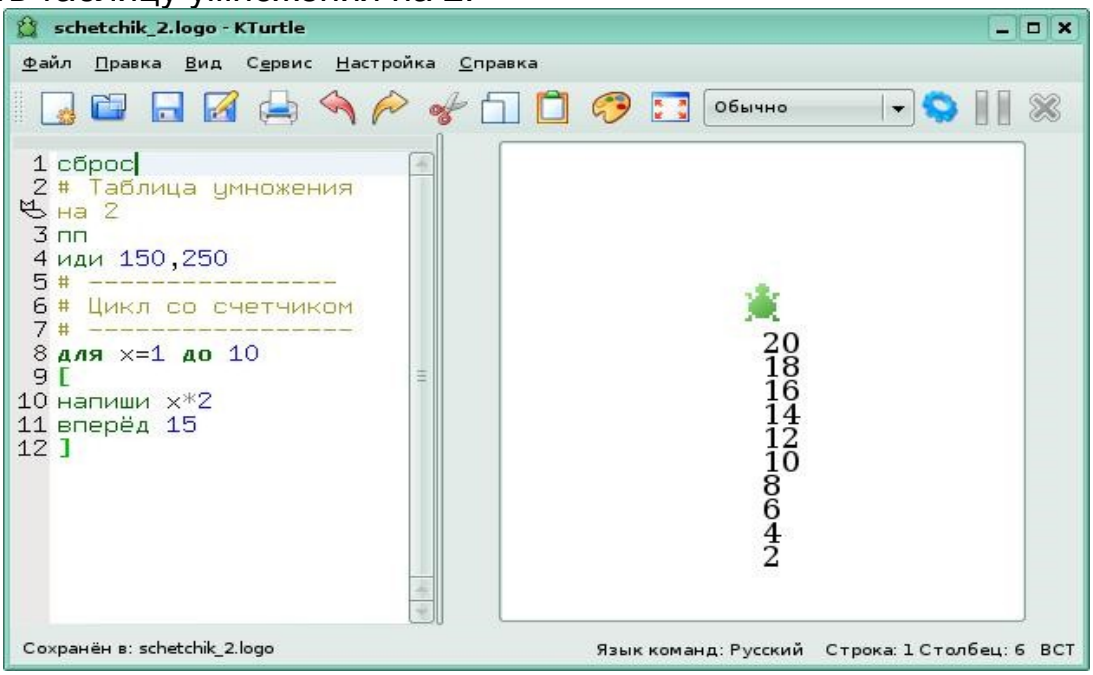


В данном случае в 9 строке мы видим запись "для x = 1 до 100", которая означает, что в качестве переменной для счетчика используется переменная "x". Сначала ей присваивается начальное значение "x = 1". Затем указывается, до какого значения x повторять цикл. Циклы со счетчиком чаще всего используются, когда заранее известно

количество повторений, которые должны быть выполнены. В нашем случае черепашка будет 100 раз продвигаться на шаг на единицу больше предыдущего и поворачиваться на 75 градусов. Еще один пример цикла со счетчиком приведен в самой программе KTurtle:



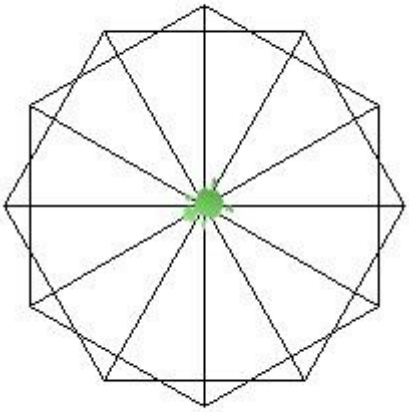
В данном примере мы видим конструкцию цикл-в-цикле. Цикл со счетчиком можно использовать для вычислений. Давай напишем программу, которая каждое натуральное число от 1 до 10 умножает на 2, то есть таблицу умножения на 2.



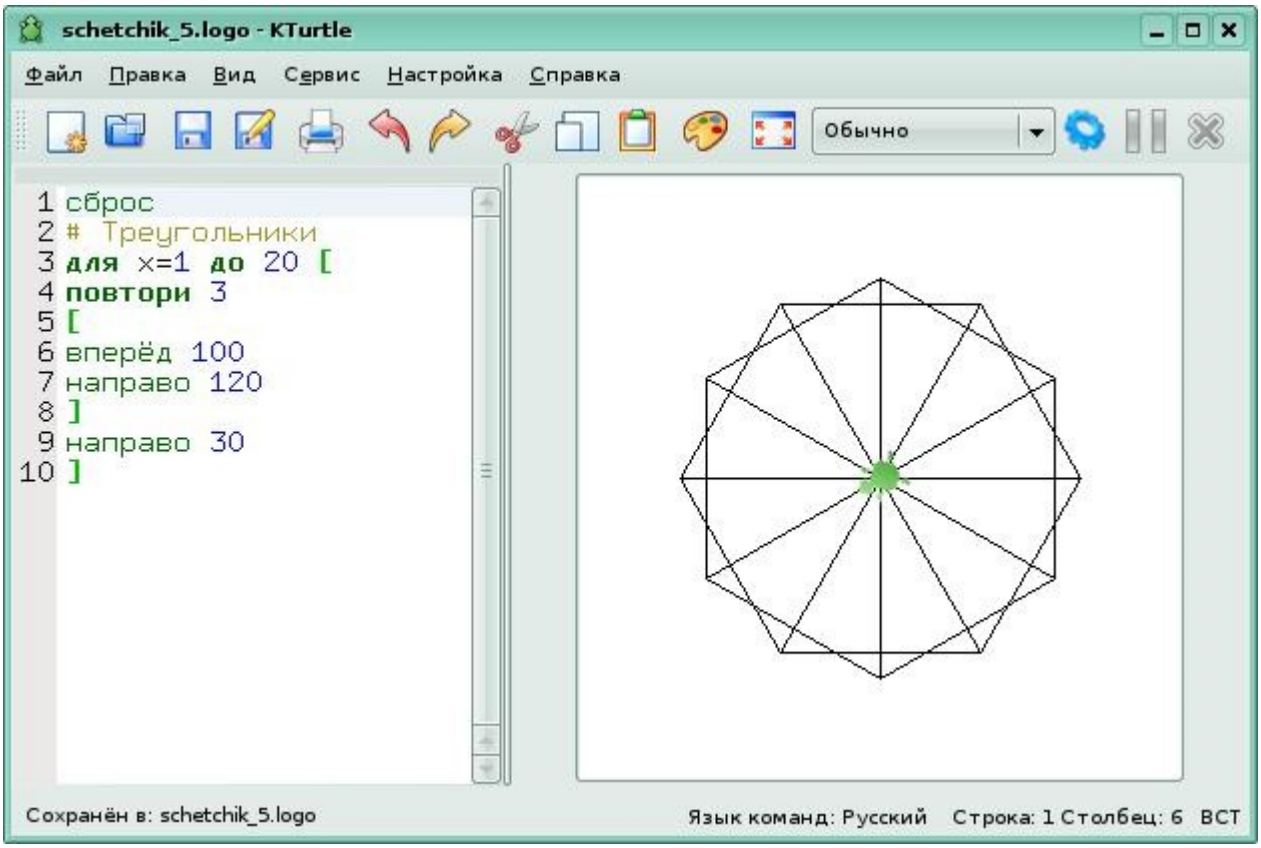
До сих пор, мы рисовали квадраты. Попробуй написать программу для рисования равностороннего треугольника. У него все углы по 60

градусов. На сколько градусов должна поворачиваться черепашка?

Если получилось нарисовать треугольник, заставь черепашку нарисовать более интересную картинку, используя цикл со счетчиком. Например, такую (из 20 треугольников):

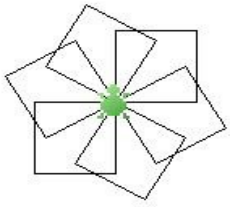


Проверь себя:



Задание 7.

1. *Создай программу для рисования такой картинки:*



2. *Напиши программу, которая выводит на экран результат таблицы умножения на какое-либо число. В программу включи диалоговое окно для ввода числа.*

Урок 8. Подробнее о циклах. Цикл с условием.

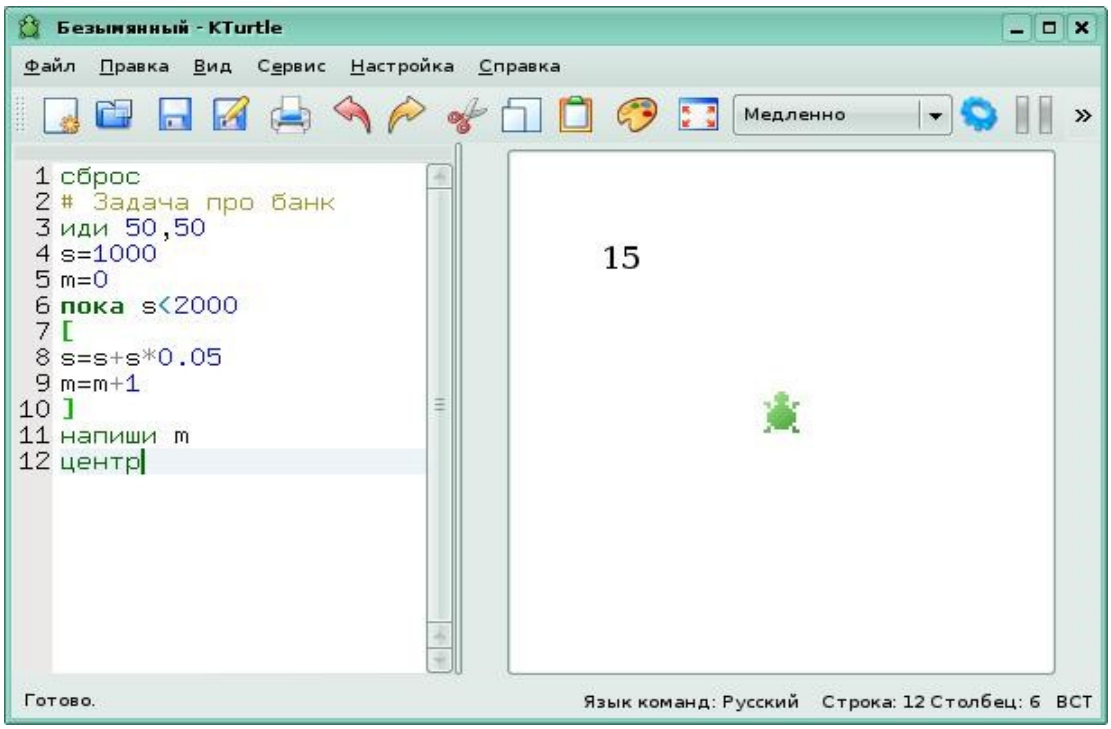
Другой разновидностью циклов является цикл с условием.

Рассмотрим такую 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 заданное условие.

Пусть s - вычисляемая сумма, а m - количество месяцев.



Задание 8.

Используя цикл с условием ("пока") напиши программу для рисования круга и пришли на проверку.

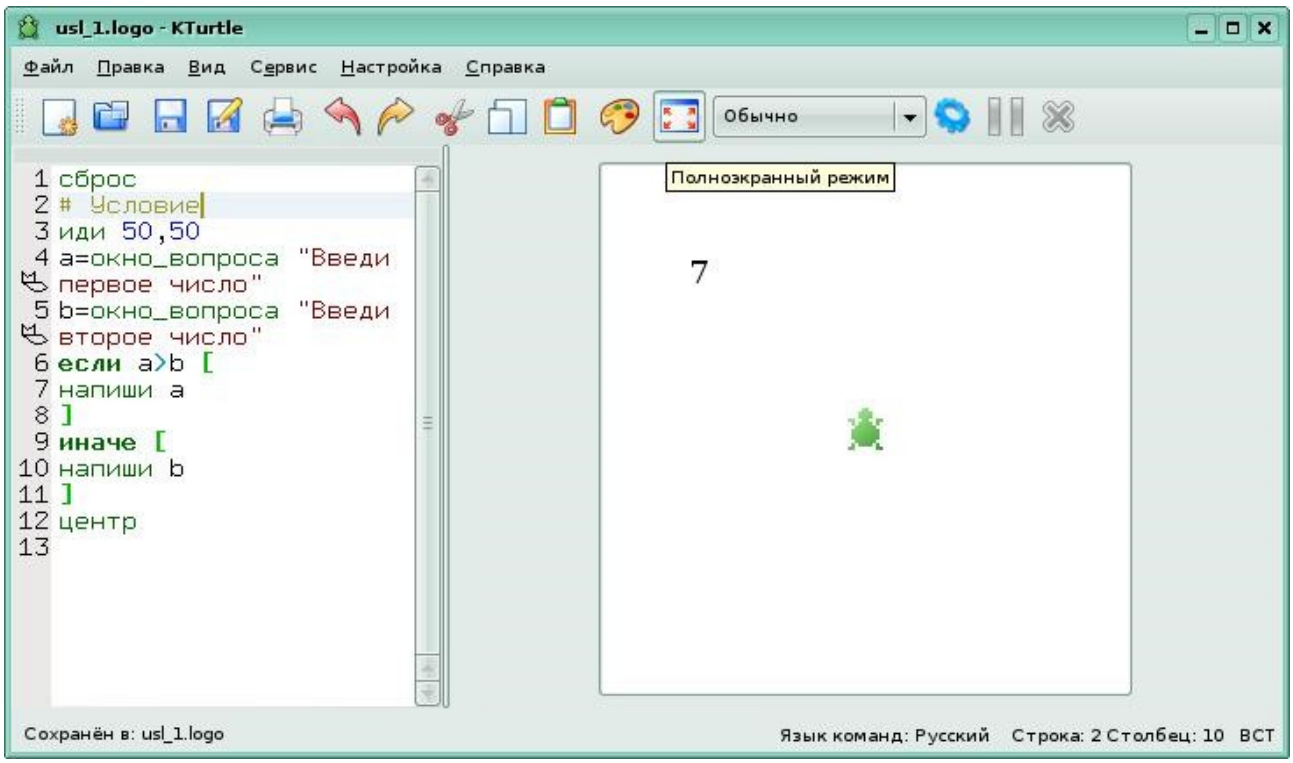
Урок 9. Условие.

Иногда в программе необходимо выполнить ту или иную последовательность действий, в зависимости от некоторого условия.

Если условие выполняется, будет произведена одна группа действий, а если не выполняется — то другая. Очевидно, что невозможно выполнить сразу обе группы действий, в каждом конкретном случае будет выполнена только одна группа.

Для того, чтобы организовать в программе выбор в зависимости от условия, используется команда если.

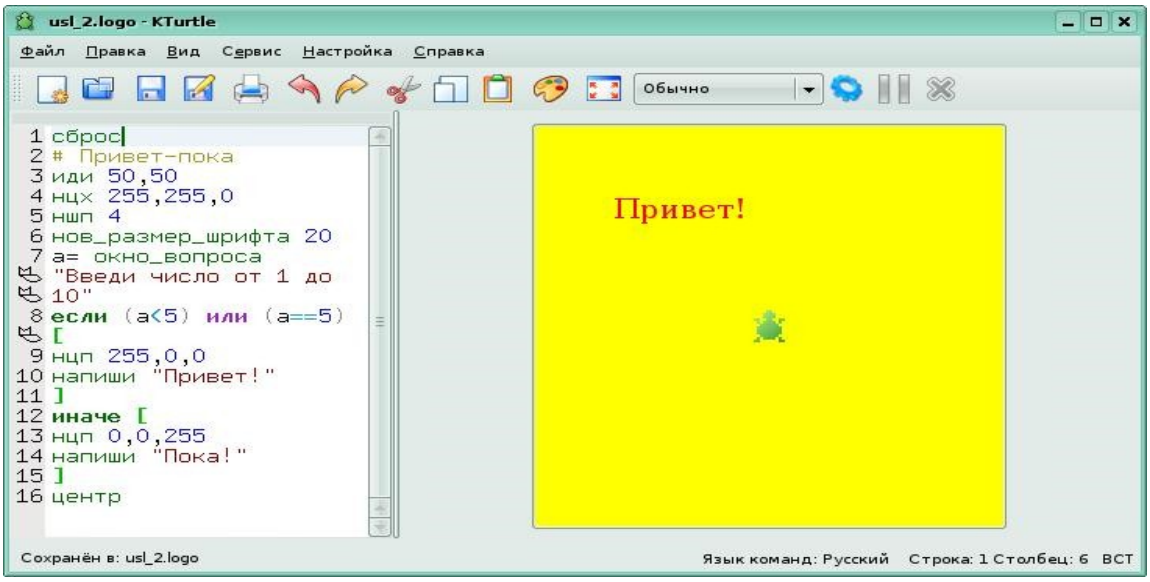
Рассмотрим пример:



В данном случае черепашка сравнивает два числа и записывает большее. Для этого используется конструкция

если (условие) [
действие 1
]
иначе [
действие 2
]

Можно организовать программу "склеив" несколько условий при помощи слов **и** и **или**. Рассмотри приведенный ниже пример. В зависимости от введенного числа черепашка пишет "Привет!" или "Пока!" и меняет цвет. Условие "склеено" словом **или**.



Попробуем изменить программу, добавив после слова "если" слово не.

```
сброс
# Привет-пока
иди 50,50
нцх 255,255,0
ншп 4
нов_размер_шрифта 20
а= окно_вопроса
"Введи число от 1 до
10"
если не (а<5) или
(а==5) [
нцп 255,0,0
напиши "Привет!"
]
иначе [
нцп 0,0,255
напиши "Пока!"
]
центр
```

При выполнении этой программы черепашка будет действовать обратно предыдущему примеру. Слово не меняет ответ на противоположный. То есть теперь при введении числа меньшего или равного 5, она будет писать слово "Пока!".

В выражениях, создающих условия, используются знаки сравнения.

a = = b a равно b
a ! = b a не равно b
a > b a больше b
a< b a меньше b
a >= b a больше или равно b
a <= b a меньше или равно b

Таким образом, в последних двух примерах мы могли двойное условие записать короче, не используя слово "или": a<=5.

Попробуй составить программу для решения следующей задачи:

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.

Проверь себя.

```
сброс
# Про мост
иди 50, 50
нов_размер_шрифта 20
m=2000 + 50*80
если m>5000 [
напиши "Не выдержит!"
]
иначе [
напиши "Выдержит!"
]
центр
```

Справочник Лого

Горячие клавиши:

Создать *(создание нового документа)*	CTRL + N
Открыть *(открывает файл Logo)*	CTRL + O
Сохранить *(сохраняет текущий файл)*	CTRL + S
Выполнить сценарий	ALT + ENTER
Остановить выполнение	ESC
Печать	CTRL + P
Выход *(завершение работы Kturtle)*	CTRL + Q
Откат *(отмена действия)*	CTRL + Z
Вырезать *(вырезает выделенный текст)*	CTRL + X
Копировать *(копирует выделенный текст)*	CTRL + C
Вставить	CTRL + V
Показать номера строк	F11
Выбор цвета	ALT + C
Увеличить отступ	CTRL + I
Уменьшить отступ	CTRL + SHIFT + I
Добавить комментарий	CTRL + D
Вызвать справку	F1

Команды управления

нрх 200,200 - устанавливает ширину и высоту холста в 200 пикселей. Ширина и высота одинаковы, холст будет квадратным.

нцх 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.

нцп 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», вызвав Справку (F1).

ншп 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.

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.

сброс - щчищает 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.

пп — перо подними

по — перо опусти

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

обертка выкл - этой командой вы отключаете обёртку холста. Это значит, что Черепашка может выйти за границы холста и «исчезнуть».

пж — Покажи. Делает Черепашку видимой после того, как она была скрыта.

сч — Спрячь. Скрывает Черепашку. Это полезно, когда Черепашка неуместна в ваших рисунках.

Команды движения:

вперёд x — вперед на x пикселей
назад x — назад на x пикселей
налево x — повернуть налево на x градусов
направо x - повернуть направо на x градусов
нпр x — устанавливает направление Черепашки на x градусов относительно 0 (левый верхний угол), а не относительно предыдущего направления
центр — перемещает Черепашку в центр холста
иди x,y - предписывает 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 x - используется для перемещения Черепашки на X пикселей от левой границы холста, высота остаётся неизменной
иди_верт x - используется для перемещения Черепашки на Y пикселей от верхней границы холста, положение относительно левой границы остаётся неизменным

Ч А С Т Ь 2.
Python

Почему именно Питон?

Что такое Python? Простой и в то же время мощный язык программирования. Если вы хотите быстро научиться создавать свои программы, обратите пристальное внимание на язык Python.

Потому что Python:

- · гибкий — подходит для разработки программ самого разного назначения и сложности: от простых учебных программ до профессиональных проектов.
- · не изобилует сложными для освоения конструкциями — в отличие от других языков, о которых вы могли слышать (C, PHP, Pascal).
- · не требует компиляции, т. е. программы на Python можно запускать на любых платформах (Windows, Linux, MacOS, …).
- · поддерживает множество стилей программирования, на его примере можно освоить любую парадигму (ООП, структурное программирование, функциональное программирование).
- · содержит ещё множество замечательных возможностей.

Язык программирования Python используется при разработке таких известных интернет-сервисов, как YouTube, Google, Yahoo!; Python применяют при разработке программ в NASA (космическое агентство США) и CERN (Европейская организация по ядерным исследованиям).

При этом знание языка Python служит хорошей основой для изучения других языков программирования.

История языка загибает второй десяток пальцев. Питон был создан в начале 90-х годов сотрудником голландского института CWI по имени **Гвидо ван Россум** (Guido van Rossum), который сегодня работает в Google. Для операционной системы Amoeba требовался расширяемый скриптовый язык, именно по этой причине Гвидо начал писать Питон на досуге, и это занятие изначально было всего лишь хобби. Первоначальная версия языка программирования заимствовала наработки из языка ABC, в разработке которого Гвидо опять-таки принимал участие. После этого язык появился в сети, и его по достоинству оценили другие программисты. Рабочие версии языка появились достаточно поздно, по этой причине в Питоне чувствуется влияние множества других языков программирования.

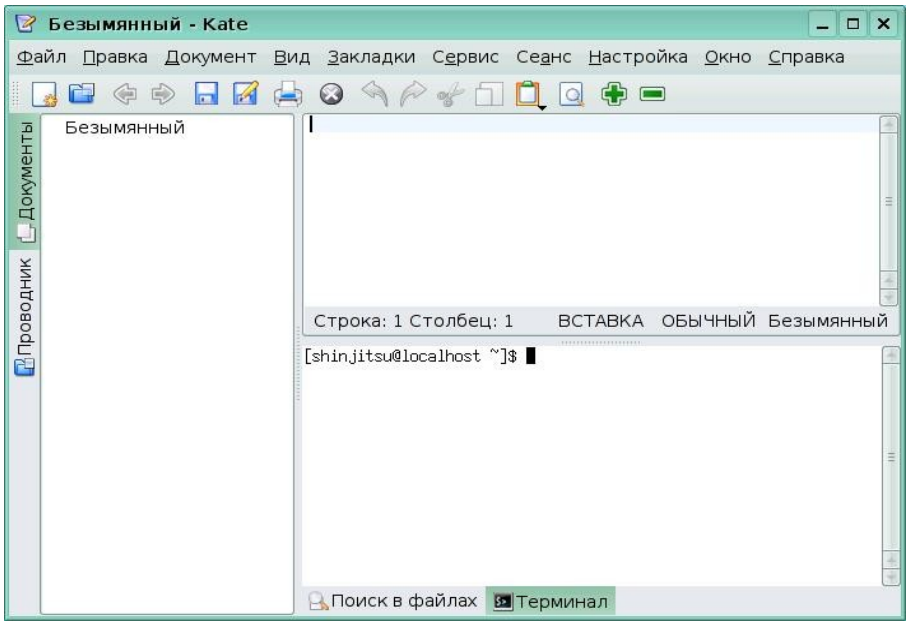
Урок 10.Текстовый редактор Kate.

НИКОГДА НЕ ЗАБЫВАЙ СОХРАНЯТЬ ФАЙЛЫ И ВО ВРЕМЯ РАБОТЫ НАЖИМАТЬ CTRL+S!!!

Программа на языке Питон — это обычный текстовый файл, который можно создать в любом текстовом редакторе.

Каждая строчка программы содержит ровно одну инструкцию для компьютера: вывести что-либо на экран, выполнить какую-либо арифметическую операцию и т.д.

Мы будем использовать текстовый редактор для написания программ Kate.



Так как ты уже умеешь пользоваться текстовыми редакторами, у тебя не возникнет проблем с использованем Kate.

Чтобы запустить редактор Kate, открой меню программ KDE. В появившемся меню запуска выбери подменю Служебные --> Редакторы. Появится список доступных редакторов. Выбери Kate.
Когда ты впервые запустишь Kate,то увидишь две области.

Область слева - это боковая панель. В ней доступны список открытых файлов (вкладка "Документы") и проводник ("Проводник") в виде "служебных панелей". Переключаться между ними можно при помощи ярлыков, которые находятся вверху боковой панели.

В правой области ты увидишь текст редактируемого файла, а в списке файлов, который находится на боковой панели -- его имя.

Над ними, как и в других текстовых редакторах, находится панель инструментов с наиболее часто используемыми командами. Ещё выше - строка меню.

Внизу ты можешь видеть два ярлыка, один из которых называется "Терминал". В этом окне мы будем запускать программу для выполнения.

Давай попробуем набрать нашу первую программу. Очень многие действия будут похожи на язык Лого, но команды Питона могут быть написаны только на английском языке.

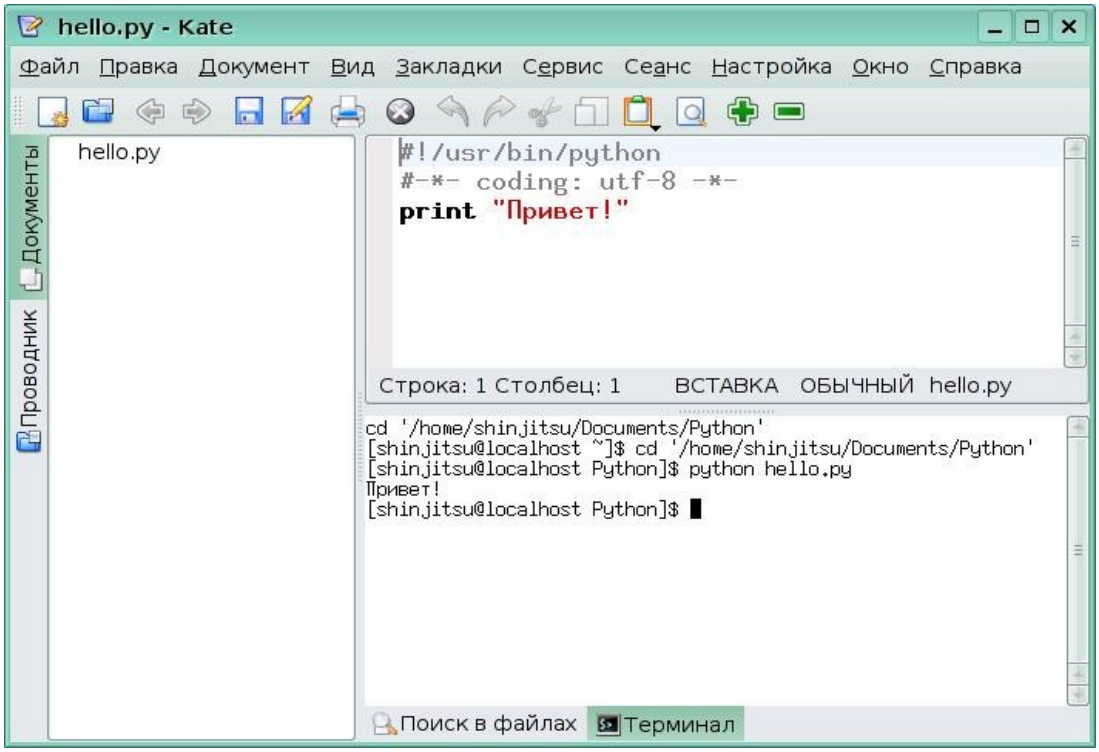
Пока не вдаваясь в подробности, давай договоримся, что первые две строки любой программы, написанной на Питоне должны начинаться так:

```
#!/usr/bin/python
#-*- coding: utf-8 -*-
```

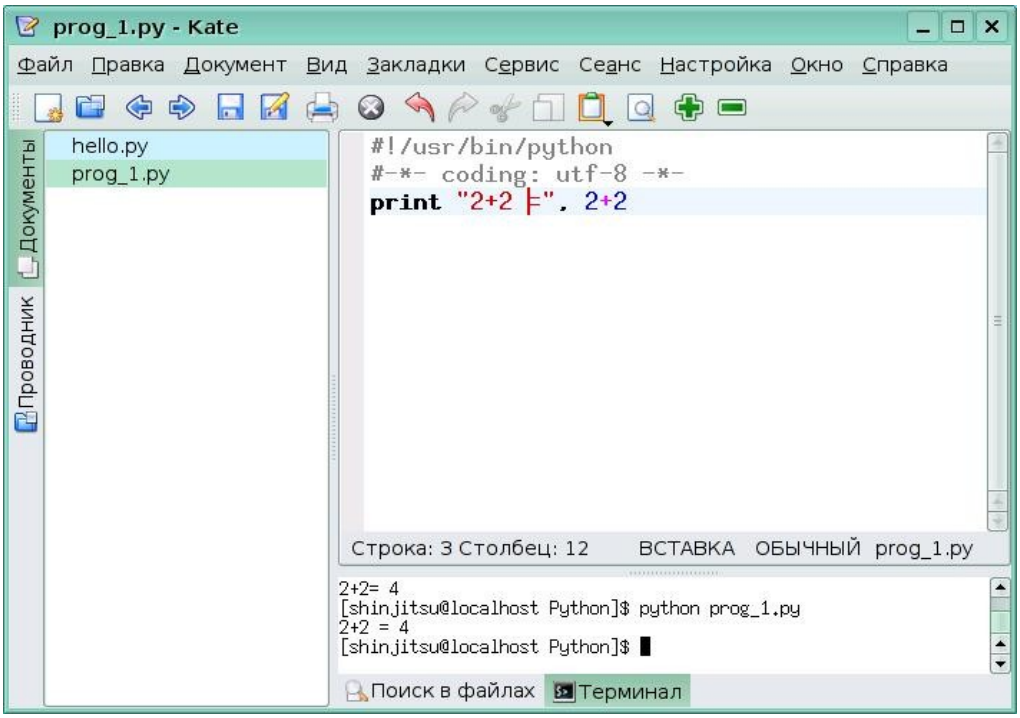
На самом деле первая строка говорит о том, что мы используем Python, а вторая указывает кодировку. Но об этом позже.

Третья строка аналогична команде Лого "напиши". Только на английском языке она задается как **print**. Набрав программу, сохрани файл с именем hello.py. Обрати внимание, расширение у файлов Питона **.py**. Теперь перейди в окно Терминала и около приглашения командной строки, которое обозначается знаком **$** напиши **python hello.py** и нажми

клавишу Enter. Результатом выполнения программы станет написанное строчкой ниже слово Привет!



Ты можешь выводить на печать не только фразы, но и результаты вычислений. Посмотри на следующую програму:



Запись в кавычках воспринимается как строка, а запись без кавычек, как математическая операция. Поэтому после выполнения программы, ты видишь: *2+2 = 4* .

В одной инструкции **`print`** можно напечатать несколько текстовых строк или выражений. Для этого их нужно перечислить через запятую.

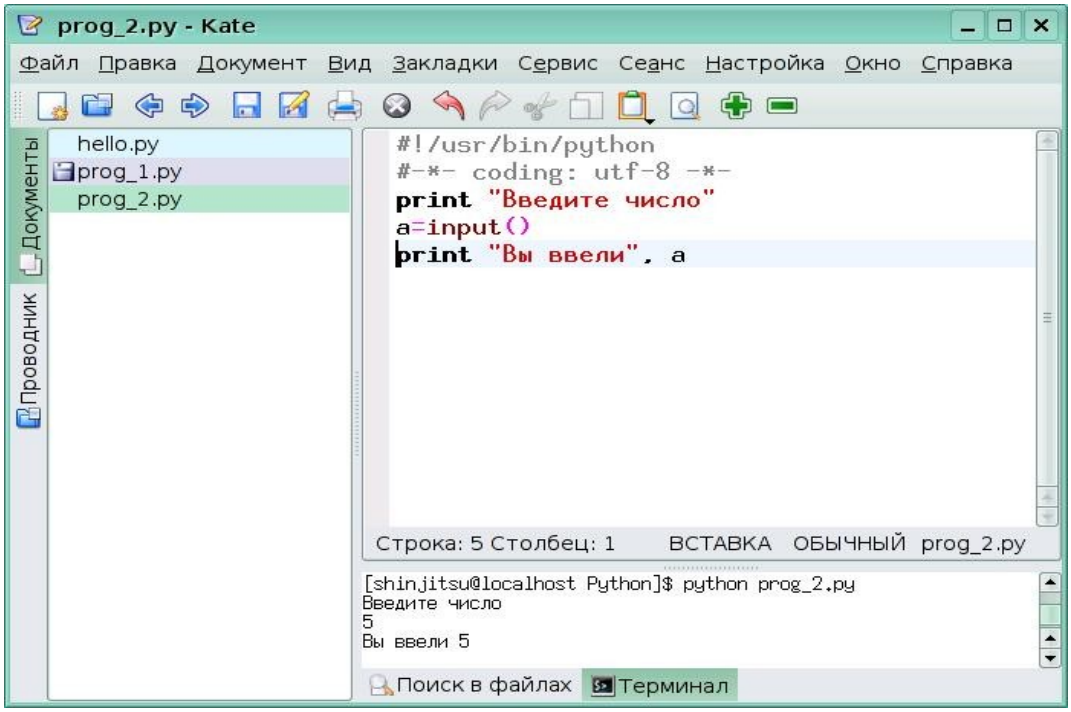
Урок 11. Ввод-вывод.

Предыдущая программа не обладала свойством интерактивности, то есть она только считывала наши данные.

Для того, чтобы программа при каждом новом запуске выводила на экран что-то новое, необходимо

1. Считать данные (например, с клавиатуры).
2. Выполнить над данными действия.
3. Вывести результат (например, на экран).

Рассмотрим следующую программу:



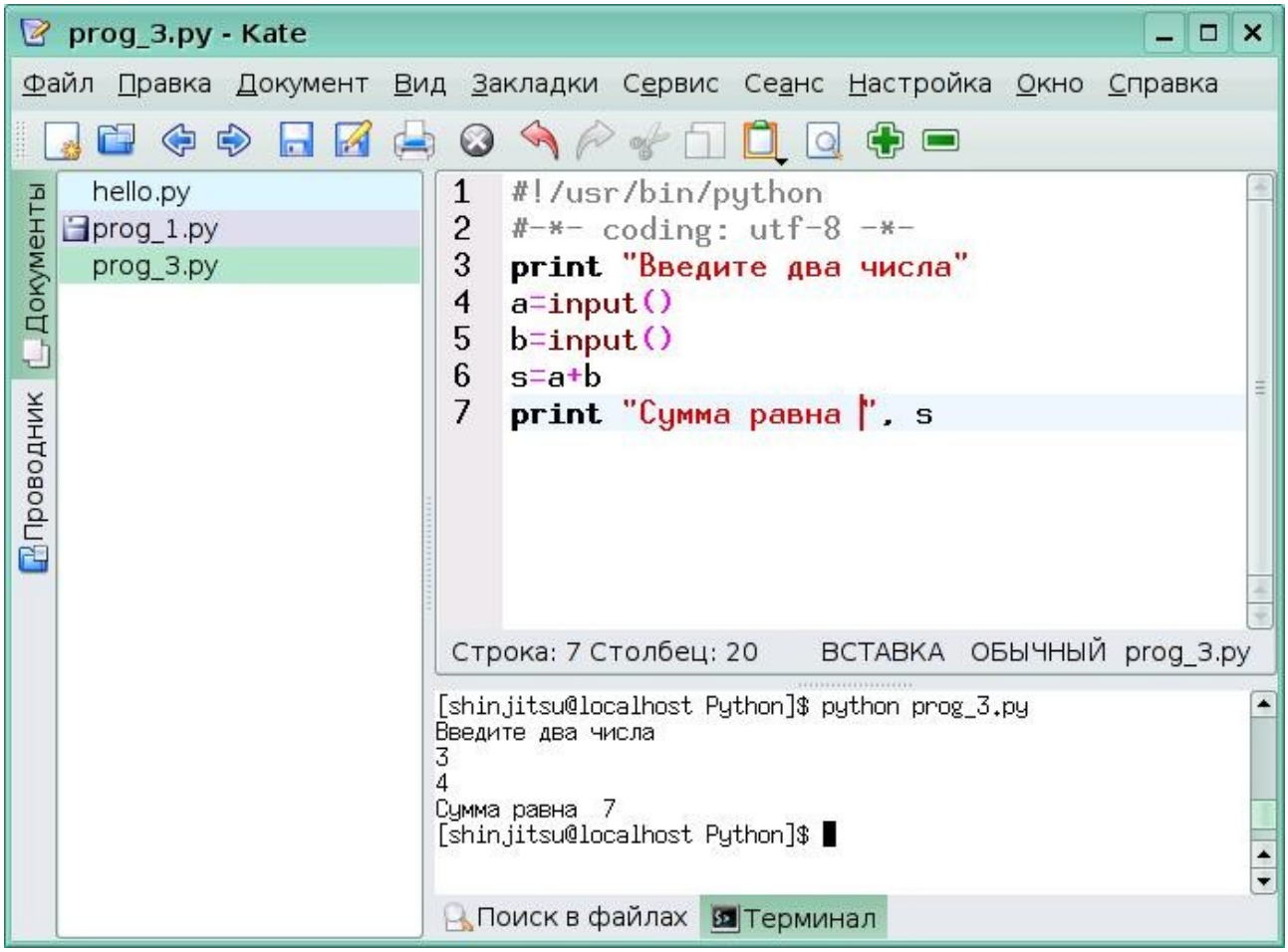
Она состоит из трех инструкций. Первая инструкция выводит на экран приглашение ввести число. Вторая инструкция `input()` ожидает ввода с клавиатуры выражения и присваивает результат ввода переменной `a`. Третья инструкция выводит на экран строку `"Вы ввели"`, а затем

значение переменной `a`.

Запустите эту программу. После появления строки **`Введите число'** наберите **5<Enter>**. Переменной `a` будет присвоено значение `5`, и в результате выполнения последней инструкции на экран будет выведено **`Вы ввели 5'**.

Теперь рассмотрим программу, которая находит сумму двух введенных чисел. У нас будет три переменные: в переменных `a` и `b` мы будем хранить два введенных пользователем слагаемых, а в переменной `s` — их сумму. Наша программа должна:

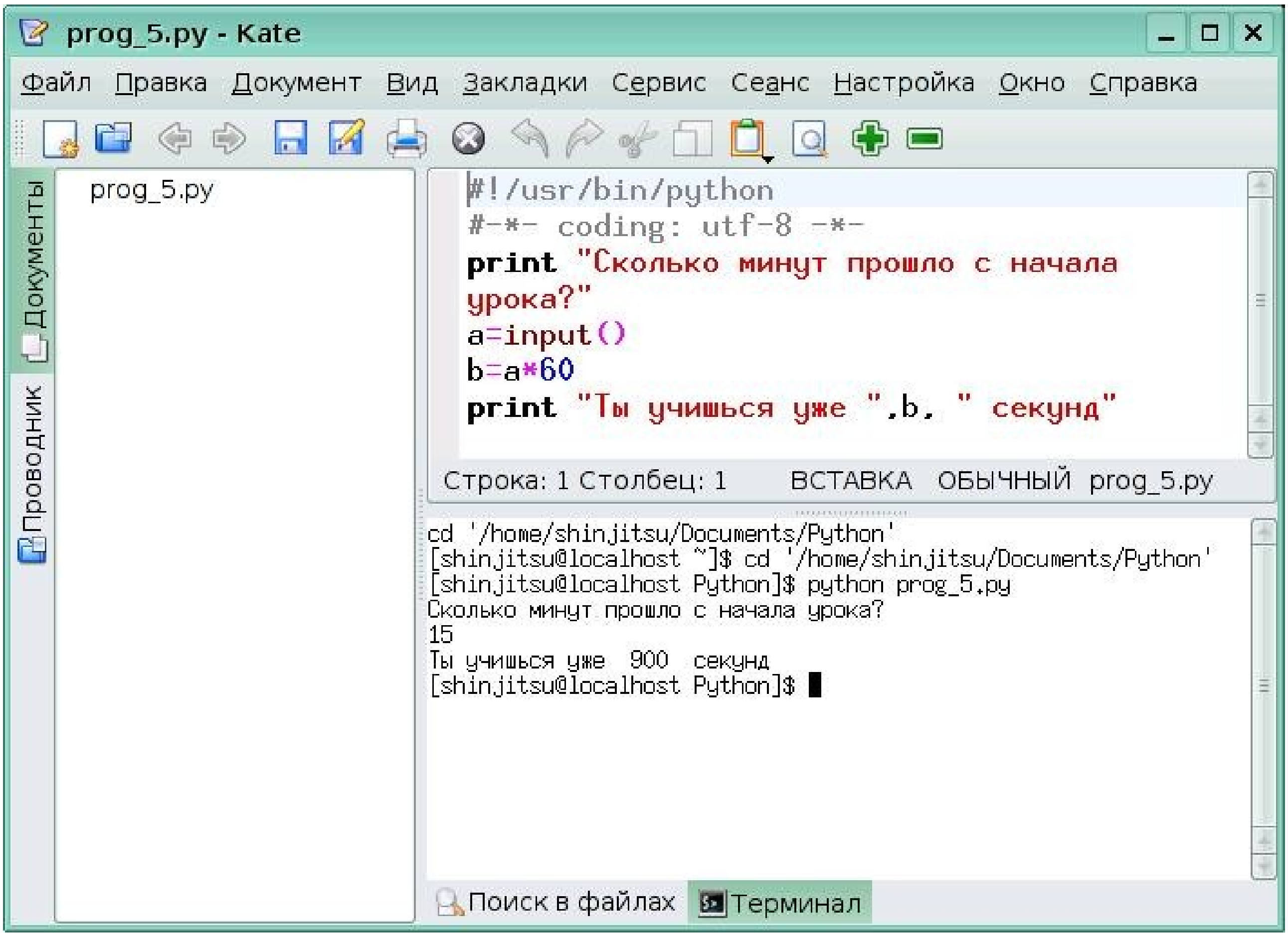
1. Попросить пользователя ввести значения двух слагаемых.
2. Считать с клавиатуры значения двух переменных `a` и `b`.
3. Присвоить переменной `s` их сумму.
4. Вывести на экран результат вычислений.



vvvvvvvvvvvvvvvvvvvvvvvvvvvvvvvvvvvvvvvvvvvvvvvvvvvvvvvvvvvv

Напиши программу, которая спрашивает "Сколько минут прошло от начала урока?", пересчитывает это время в секунды и выводит сообщение "Ты учишься уже ... секунд!". Функция умножения аналогична языку Лого. Проверь себя.

vvvvvvvvvvvvvvvvvvvvvvvvvvvvvvvvvvvvvvvvvvvvvvvvvvvvvvvvvvvvvvvvv



Урок 12. Библиотека turtle.

Библиотека `turtle` *– это расширения языка Питон, позволяющее рисовать на экране несложные рисунки.*

В языке Лого мы писали программы для черепашки. Аналогичные программы можно записать и на Питоне. Для этого нужно подключить библиотеку turtle. Программу, использующую этот графический модуль надо начинать командами (не забудь про первые две строки!):

import turtle
turtle.reset()

Первая команда подключает библиотеку turtle, вторая аналогична команде "сброс" в Лого.

Для того, чтобы задержать графическое окно на экране, необходимо заканчивать все программы, использующие модуль turtle командой:

turtle._root.mainloop()

Надо отметить, что на экране мы увидим не черепашку, а треугольничек, но по привычке будем называть его "черепашкой".

В Питоне, как в Лого и других языках программирования, необходимо использовать комментарии. Они тоже обозначаются знаком #, но в отличии от Лого, комментарии можно писать не только в отдельной строке, но и в той же строке, правее команды.

Пронумеровать строки можно клавишей F11.

Нам придется выучить команды для черепашки на английском языке.

Вот необходимый набор команд.

forward (a)	вперёд на a шагов
backward (a)	назад на a шагов
left (ß)	налево на ß градусов
right (ß)	направо на ß градусов
circle (r)	нарисовать окружность радиуса r, центр слева - r>0, справа - r>0
circle (r,ß)	нарисовать дугу радиуса r и градусной мерой ß
goto (x,y)	"иди" в точку с координатами (x,y)
down ()	перо подними

up ()	перо опусти
width (a)	новая ширина пера
color (s)	цвет пера, указывается по-английски в кавычках, "red" - красный, "green" - зеленый, "blue" - синий, "white" - белый, "black" - черный, "yellow" - желтый, "pink" - розовый
fill (f)	закрашивает замкнутую область, перед закрашиванием дай команду *turtle.fill(1)*, а закончив раскрашивание - *turtle.fill(0)*.
reset ()	сброс
clear ()	очистка экрана
write (s)	вывод текста (=напиши)

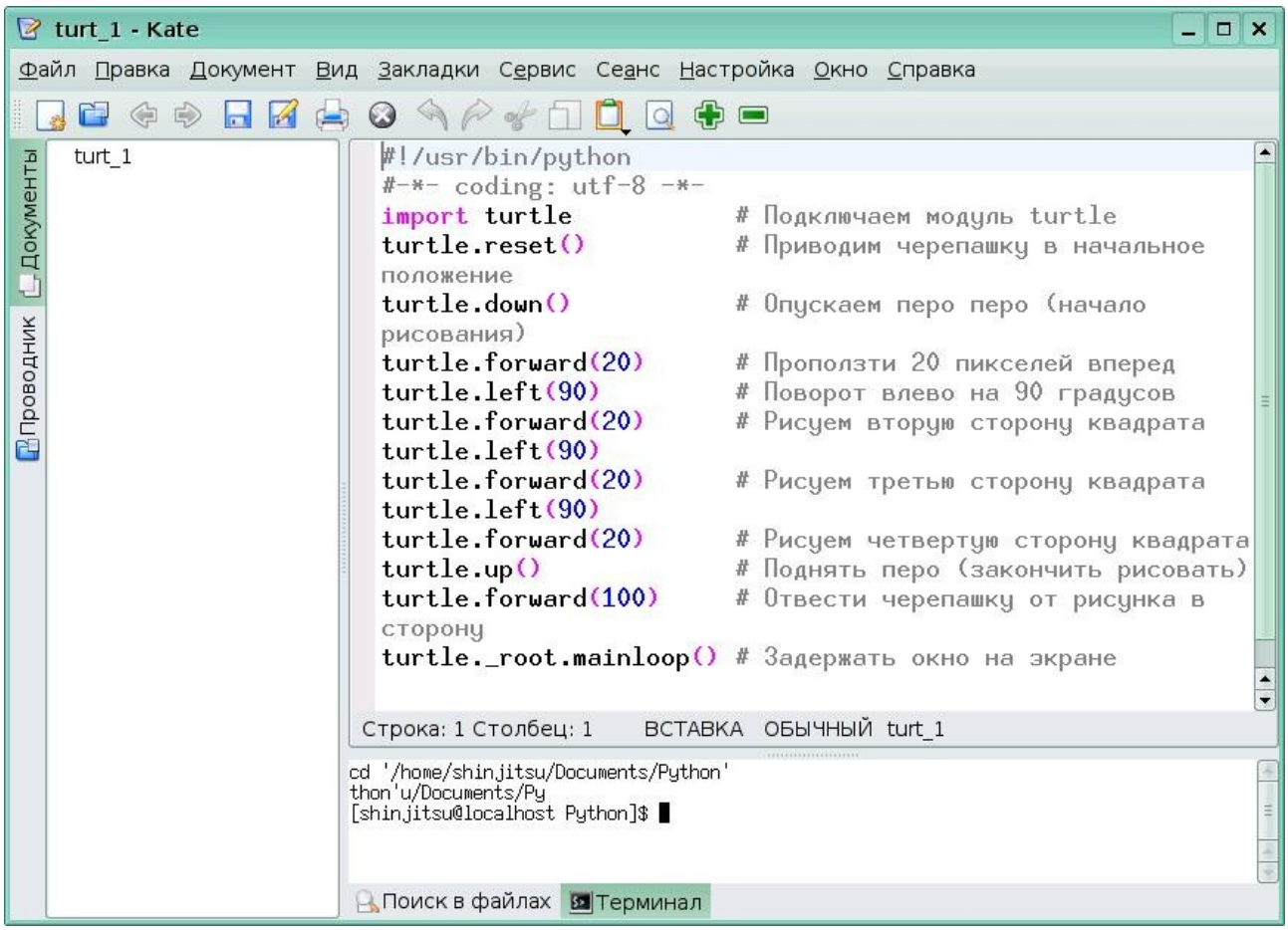
Перед каждой командой этой библиотеки записывается ее название, то есть команда "вперёд на 50" шагов будет выглядеть так:
turtle.forward(20)

vvvvvvvvvvvvvvvvvvvvvvvvvvvvvvvvvvvvvvvvvvvvvvvvvvvvvvvvvvvvv vvvvvvvvvvvvvvvv

Напиши и выполни программу для рисования квадрата, как первый раз в Лого. Пока она не будет содержать цикла.

vvvvvvvvvvvvvvvvvvvvvvvvvvvvvvvvvvvvvvvvvvvvvvvvvvvvvvvvvvvvv vvvvvvvvvvvvvvvv

Проверь себя!



Мы начинали изучать Лого с рисования квадратов. У тебя получилось написать программу для рисования квадрата на Python?

А теперь, как и в Лого, попробуем организовать цикл. Вот как будет выглядеть программа для рисования квадрата на Питоне, записанная при помощи цикла:

```
#!/usr/bin/python
#-*- coding: utf-8 -*-
import turtle
for i in range(4):
        turtle.forward(100)
        turtle.right(90)
turtle._root.mainloop()
```

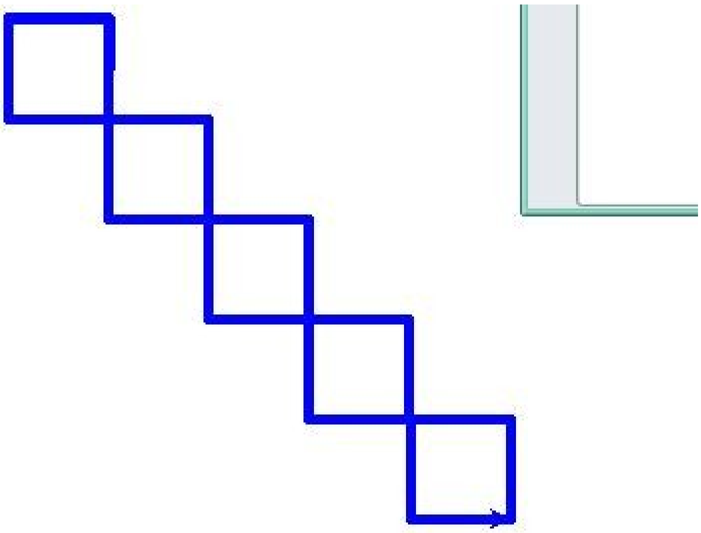
Здесь третья строка - добавление библиотеки turtle. В четвертой строке мы организуем цикл, который заставляет черепашку 4 раза повторять одно и то же. Тело цикла, то есть те команды, которые выполняются в цикле, смещаются при записи программы клавишей <Tab>. Последняя строка нужна для фиксации окна на экране.

Vvvvvvvvvvvvvvvvvvvvvvvvvvvvvvvvvvvvvvvvvvvvvvvvvvvvvvvvvvvvvvvv

Напиши программу для рисования квадрата, в которой предусмотрен ввод длины стороны с экрана, подсчет площади квадрата и вывод ее черепашкой на экран.

vvvvvvvvvvvvvvvvvvvvvvvvvvvvvvvvvvvvvvvvvvvvvvvvvvvvvvvvvvvvvvvvvvv

Напиши программу для рисования рисунка из 5 квадратов, который ты уже рисовал в Лого.



Проверь себя.

```
#!/usr/bin/python
#-*- coding: utf-8 -*-
import turtle
turtle.up()
turtle.goto(-150,130)
turtle.down()
turtle.width(5)
turtle.color("blue")
for i in range(5):
        for i in range(5):
                turtle.forward(50)
                turtle.right(90)
        turtle.forward(50)
        turtle.left(90)

turtle._root.mainloop()
```

Урок 13. Функции.

Черепашке была нужна команда "выучи" для написания подпрограмм. В Питоне такие подпрограммы называются функциями. Посмотрим как организовать функцию в Python.

Для начала напишем программу для рисования треугольника.

```
#!/usr/bin/python
#-*- coding: utf-8 -*-
# Рисуем зеленый треугольник
import turtle
turtle.down()
turtle.width(5)
turtle.color("green")
turtle.fill(1)
for i in range(3):
        turtle.forward(100)
        turtle.left(120)
turtle.fill(0)
turtle._root.mainloop()
```

Как видишь, в программе организован цикл, в котором трижды повторяется движение вперед и поворот на 120 градусов.

Теперь представим себе, что нам нужно нарисовать ни один, а несколько треугольников. Для этого создадим функцию treug() и будем ее использовать в программе.

Для создания функции используется следующая конструкция:

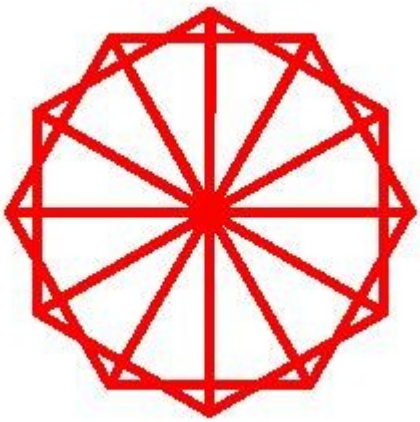
```
def название_функции():
описание действий
```

Разберем программу для рисования двух зеленых треугольников.

```
#!/usr/bin/python
#-*- coding: utf-8 -*-
# Рисуем два треугольника
import turtle
turtle.down()
turtle.width(5)
turtle.color("green")
def treug():
        for i in range(3):
                turtle.right(120)
                turtle.forward(100)
turtle.reset
treug()
turtle.forward(50)
treug()
turtle._root.mainloop()
```

В этой программе мы создаем функцию treug(). При этом описание действий должно быть смещено. В нашем случае - это цикл, рисующий треугольник, в котором команды также смещаются. После создания функции, мы перегружаем черепашку (равносильно команде "сброс") turtle.reset. Далее мы дважды вызываем созданную функцию для рисования двух треугольников на расстоянии 50 шагов между ними.

В программе может быть создана ни одна функция. Функция может использоваться и в теле цикла. Еще одним важным моментом является то, что при задании количества шагов или угла поворота может использоваться параметр или арифметическое выражение. Создадим программу для рисования красного треугольного колеса:



Для рисования нашего треугольного колеса нам понадобится функция

treug(). Всего в колесе 12 треугольников. Значит, нужно будет организовать цикл из 12 повторений и использовать созданную функцию. Каждый раз черепашка должна поворачиваться на один и тот же угол. Так как, у нас 12 треугольников, а вся окружность составляет 360 градусов, то этот угол мы найдем деление 360 на 12. Это действие запишем так: *turtle.right(360.0 / 12)*.

```
#!/usr/bin/python
#-*- coding: utf-8 -*-
# Рисуем красное треугольное колесо
import turtle
turtle.down()
turtle.width(5)
turtle.color("red")
# Создаем функцию для рисования треугольника
def treug():
        for i in range(3):
                turtle.right(120)
                turtle.forward(100)
turtle.reset
# Используем созданную функцию
for i in range(12):
        turtle.right(360.0 / 12)
        treug()
turtle._root.mainloop()
```

Теперь посмотрим, как создается функция с параметром. Предположим, что нам нужно нарисовать два треугольника разного размера. Для этого в названии функции мы зададим параметр (переменную величину). Мы таким же образом поступали и в Лого. Посмотри в названии функции мы добавили параметр size, который потом указали вместо определенного количества шагов в команде *turtle.forward(size)*. Потом мы указываем значение параметра ри вызове функции, например, treug(50), черепашка нарисует треугольник заданного размера.

```
#-*- coding: utf-8 -*-
# Рисуем треугольники разного размера
import turtle
turtle.down()
turtle.width(3)
def treug(size):
        for i in range(3):
                turtle.right(120)
                turtle.forward(size)
turtle.reset
treug(50)
turtle.forward(50)
treug(100)
turtle._root.mainloop()
```

vvvvvvvvvvvvvvvvvvvvvvvvvvvvvvvvvvvvvvvvvvvvvvvvvvvvvvvvvvvvv

Напиши программу для рисования рисунка из 5 квадратов, который ты уже рисовал. Только теперь создай и используй функцию с параметром и предусмотри ввод параметра (размера квадрата) самим пользователем.

<u>Программа без параметра и диалога может выглядеть так!</u>

```
turtle.up()
turtle.goto(-150,130)
turtle.width(5)
turtle.color("blue")
turtle.down()
def kvadrat():
        for i in range(5):
                turtle.forward(20)
                turtle.right(90)
turtle.reset
for i in range(5):
        kvadrat()
        turtle.forward(20)
        turtle.left(90)
turtle._root.mainloop()
```

vvvvvvvvvvvvvvvvvvvvvvvvvvvvvvvvvvvvvvvvvvvvvvvvvvvvvvvvvvvvv