Возможности графического модуляTurtle (Черепаха) при обучении информатике в профильных классах

Мирная Мария Ивановна

МОАНУ СОШ №17
им.К.В.Навальневой
МО Кореновский район

Для работы с исполнителем Черепаха сначала нужно подключить модуль turtle

```
import turtle
или
from turtle import *
```

Принцип организации библиотеки графического вывода построен на метафоре Черепахи, воображаемого роботоподобного устройства, которое перемещается по экрану или бумаге и поворачивается в заданных направлениях, при этом оставляя (или, по выбору, не оставляя) за собой нарисованный след заданного цвета и ширины.

Проще: **черепашка ползает по экрану и рисует. Мы управляем черепашкой на плоскости при помощи программы.**

Функции на примере turtle

Функция – фрагмент программного кода, к которому можно обратиться по имени. Иногда функции бывают безымянными.

У функции есть входные и выходные параметры. Функция fd(150) – фрагмент программного кода, который двигает курсор вперед на заданное во входном значении количество пикселей (150). Выходного значения у функции fd() нет.

Когда функцию надо выполнить, после ее названия пишут круглые скобки. fd – просто название, ничего не происходит. fd(100) – функция выполняется с входным параметром 100. Обычно названия функций пишут с маленькой буквы.

Классы на примере turtle

Класс – программный шаблон для создания объектов, заготовка для чего-то, имеющего собственное состояние. Мы можем нарисовать прямоугольник и назвать его кнопкой, но это еще не кнопка, потому что у нее нет собственных свойств и поведения. Прямоугольник надо научить быть самостоятельной, отличной от других, кнопкой.

Turtle – класс, его имя пишется с большой буквы. через оператор присваивания = мы создаем экземпляр класса: circ = turtle.Turtle(). Turtle – класс (шаблон, трафарет, заготовка), circ – его экземпляр (рисунок, набор уникальных цветов, штрихов и свойств). На картинке выше видно, что экземпляр класса circ богат установленными свойствами, а экземпляр t обладает свойствами по умолчанию: тонкая черная линия, треугольный курсор.

Программирование с использованием классов и их экземпляров будем называть объектно-ориентированным программированием, ООП. объектно-ориентированный подход необходим при построении графического интерфейса пользователя, GUI.

Список основных команд черепашки

Команды перемещения черепашки	
`forward(n)/fd(n)`	Проползти вперед n шагов (пикселей)
`backward(n)`	Проползти назад n шагов (пикселей)
`left(angle)`	Повернуться налево на angle градусов
`right(angle)`	Повернуться направо на angle градусов
`circle(r)`	Нарисовать окружность радиуса |r|, центр которой находится слева от черепашки, если r>0 и справа, если r<0
`circle(r,angle)`	Нарисовать дугу радиуса |r| и градусной мерой angle. Дуга рисуется против часовой стрелки, если r>0 и по часовой стрелке, если r<0
`goto(x,y)`	Переместить черепашку в точку с координатами (x,y)

Команды рисования	
`down()`	Опустить перо. После этой команды черепашка начнет оставлять след при любом своем передвижении
`up()`	Поднять перо
`width(n)`	Установить ширину следа черепашки в n пикселей
`color(s)`	Установить цвет следа черепашки в s. s должно быть текстовой строкой, заключенной в кавычки, с названием цвета (по-английски), например: "red", "yellow", "green" и т.д.
`Bgcolor(s)`	Устанавливает цвет фона в s. s должно быть текстовой строкой, заключенной в кавычки, с названием цвета (по-английски), например: "red", "yellow", "green" и т.д.
`fill(f)`	Используется для рисования закрашенных областей. Начиная рисовать закрашенную область, дайте команду turtle.fill(1), а закончив рисование области — turtle.fill(0)
`stamp()`	После выполнения этой команды в окне для графики в месте, на котором была черепашка, останется рисунок этой черепашки.
`setheading(x)`	где x – угол поворота в градусах относительно начального положения

Прочие команды	
`reset()`	Возврат черепашки в исходное состояние: очищается экран, сбрасываются все параметры, черепашка устанавливается в начало координат, глядя вправо
`write(s)`	Вывести текстовую строку s в точке нахождения черепашки
`dot(r, color)`	Команда ставит точку, где r – радиус точки и color – цвет точки
`radians()`	Установить меру измерения углов (во всех командах черепашки) в радианы
`degrees()`	Установить меру измерения углов (во всех командах черепашки) в градусы. Этот режим включен по умолчанию
`tracer(f)`	Включить режим отладки (трассировки) программы черепашки, если значение f равно 1. Если значение f равно 0, отключить режим отладки. В режиме отладки черепашка перемещается медленнее и на экране нарисована сама черепашка. По умолчанию режим отладки включен

Исполнитель Черепаха действует на плоскости с декартовой системой координат. В начальный момент Черепаха находится в начале координат, её голова направлена вдоль положительного направления оси ординат, хвост опущен. При опущенном хвосте Черепаха оставляет на поле след в виде линии. В каждый конкретный момент известно положение исполнителя и направление его движения. У исполнителя существует две команды: **Вперёд n** (где n – целое число), вызывающая передвижение Черепахи на n единиц в том направлении, куда указывает её голова, и **Направо m** (где m – целое число), вызывающая изменение направления движения на m градусов по часовой стрелке. Запись **Повтори k [Команда1 Команда2 ... КомандаS]** означает, что последовательность из S команд повторится k раз.

Задача 1

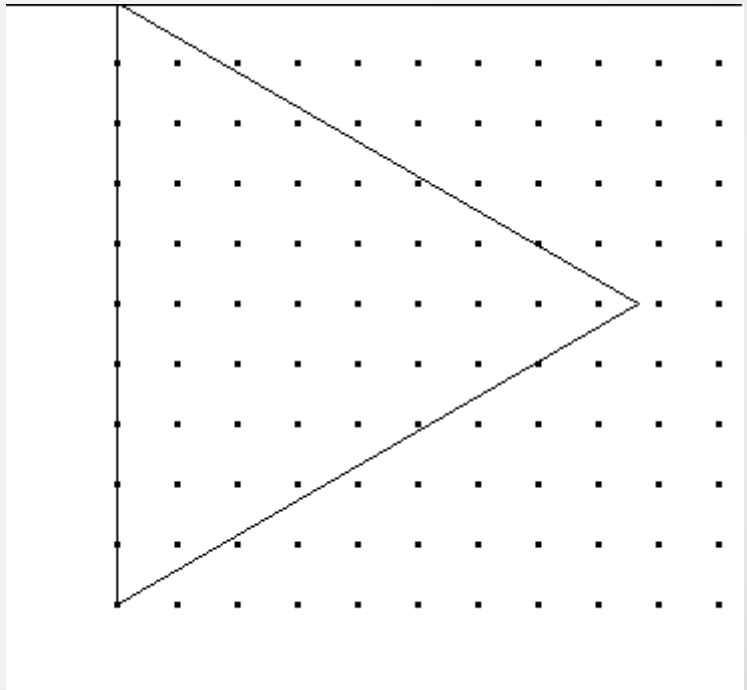
Черепахе был дан для исполнения следующий алгоритм:

Повтори 7 [Вперёд 10 Направо 120]

Определите, сколько точек с целочисленными координатами будут находиться внутри области, ограниченной линией, заданной данным алгоритмом. Точки на линии учитывать не следует.

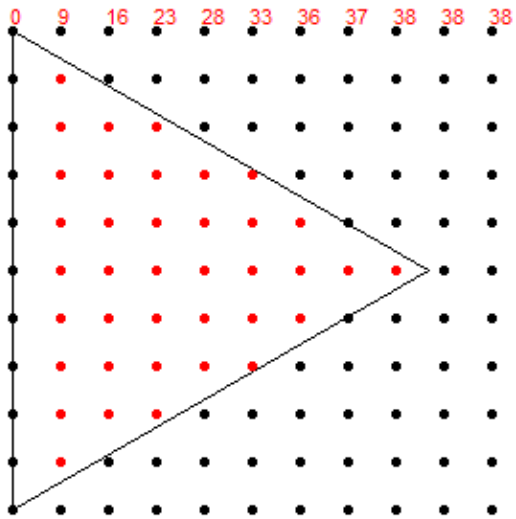
Решение

```
import turtle as t
k = 30 # масштаб
t.left( 90 ) # развернуть Черепаху "на север"
for i in range(7):
    t.forward( 10*k )
    t.right( 120 )
t.up()
for x in range(0, 11):
  for y in range(0, 11):
    t.goto( x*k, y*k )
    t.dot( 4 )
```



Решение с модулем math

```
'''
from turtle import *
from math import *
tracer(2)               # speed(0) команды ускорения рисования
hideturtle()            # ht() скрыть изображение черепашки
m = 25                  # масштаб
left(90)                # lt(90)
for i in range(7):
  forward(10 * m)       # fd(10*m)
  right(120)            # rt(120)
penup()                 # up() поднять перо Черепашки
count = 0               # количество точек, попавших внутрь треугольника
k = tan(pi / 6)         # тангенс угла пи/6 радиан (угла 30 градусов)
for x in range(0, 11):
  for y in range(0, 11):
    goto(x * m, y * m)
    if (y < -k * x + 10  # ниже верхней линии треугольника?
      and y > k * x      # выше нижней линии треугольника?
        and x > 0):      # правее вертикальной линии треугольника?
      dot(5, 'red')      # красная точка (точка попала внутрь треугольника)
      count += 1
    else:
      dot(5, 'black') # черная точка (точка не попала внутрь треугольника)
  color('red')
  write(count)          # количество красных точек по колонкам нарастающим итогом
mainloop()              # done()
```



Задача 2

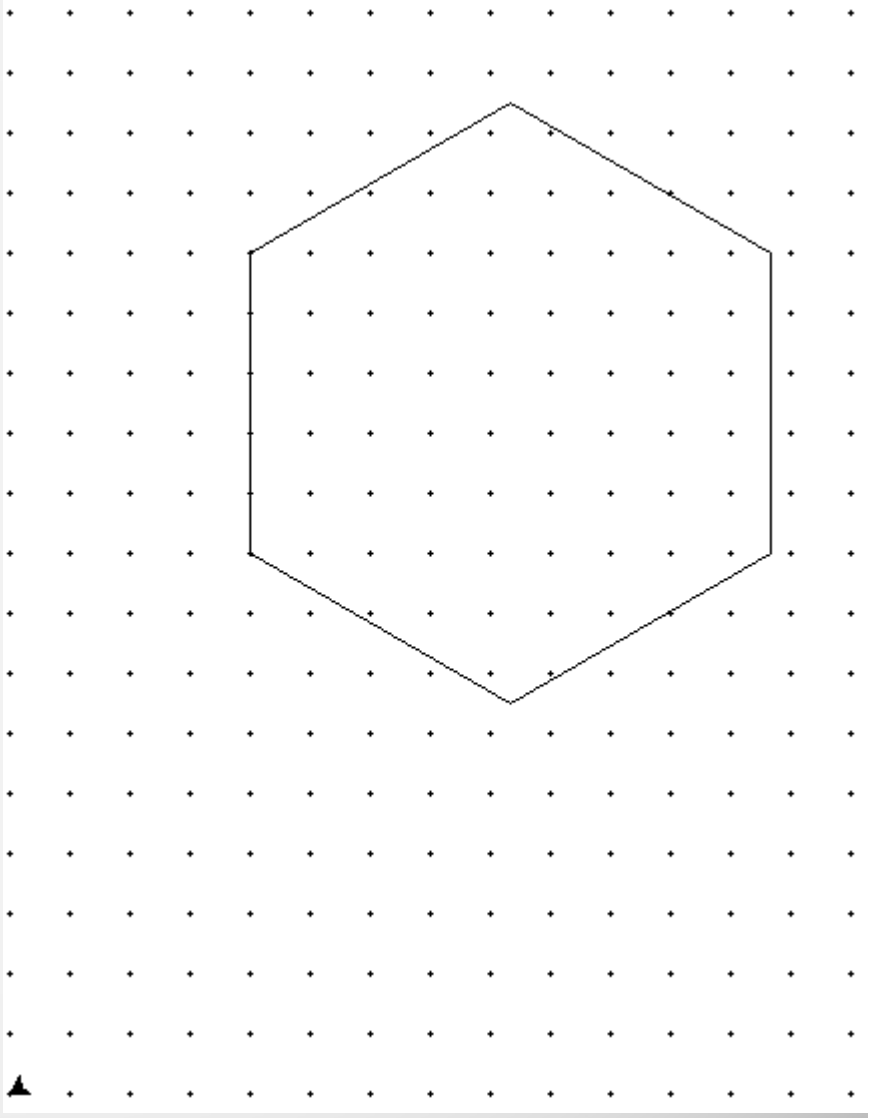
Черепахе был дан для исполнения следующий алгоритм:

Повтори 10 [Вперёд 5 Направо 60]

Определите, сколько точек с целочисленными координатами будут находиться внутри области, ограниченной линией, заданной данным алгоритмом. Точки на линии учитывать не следует.

Решение

```
import turtle as t  # Подключим модуль черепашка
k = 30
t.left(90)
t.speed(10)
for i in range(10):  # пропишем алгоритм построения фигуры по условию
    t.forward(5 * k)
    t.right(60)
t.up()
t.speed(10)  # Увеличим скорость черепашки
for x in range(10, -5, - 1):  # Алгоритм построения точек
    for y in range(10, -10, - 1):
        t.goto(x * k, y * k)
        t.dot(3)  # точки размером 4 пикселя
t.done()
```



Исполнитель Цапля

Исполнитель Цапля действует на плоскости с декартовой системой координат. В начальный момент Цапля находится в начале координат, её клюв направлен вдоль положительного направления оси ординат, клюв опущен. При опущенном клюве Цапля оставляет на поле след в виде линии. В каждый конкретный момент известно положение исполнителя и направление его движения. У исполнителя существует три команды: Вперёд *n* (где *n* — целое число), вызывающая передвижение Цапли на *n* единиц в том направлении, куда указывает её клюв; Направо *m* (где *m* — целое число), вызывающая изменение направления движения на *m* градусов по часовой стрелке; Дута *r*, *a*, *b*, α (где *r*, *a*, *b*, α — целые числа), вызывающая передвижение Цапли из текущей точки с координатами (*x*, *y*) по дуге окружности с центром в точке с координатами (a+x, b+y) и радиусом *r*, градусная мера дуги равна α, движение по дуге идёт по часовой стрелке.

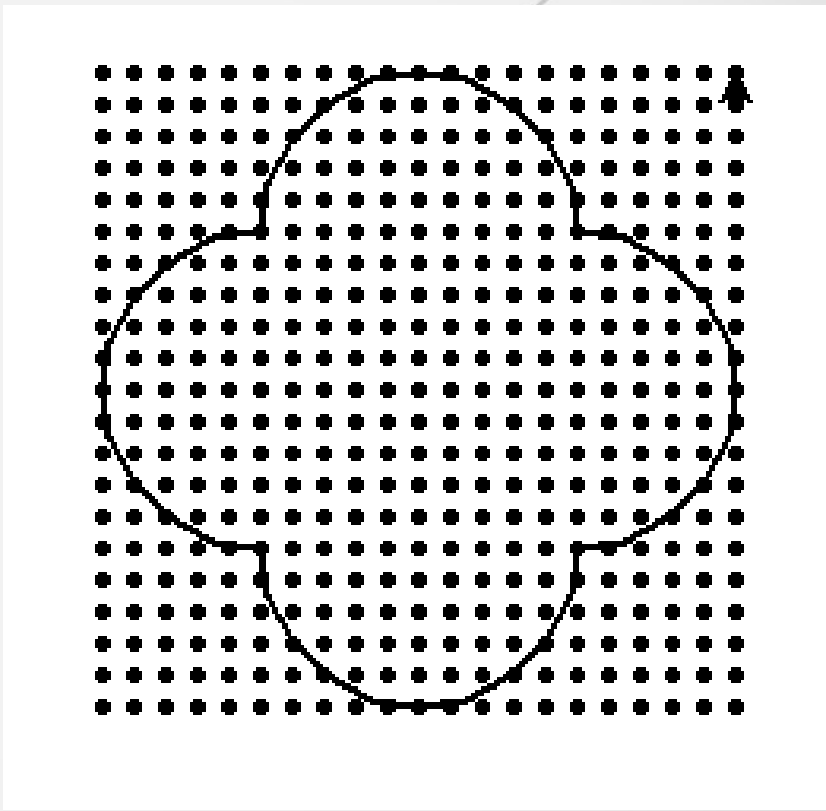
Запись **Повтори *k* [Команда 1 Команда 2 … Команда *S*]** означает, что последовательность из *S* команд повторится *k* раз.

Задача 3

Цапле был дан для исполнения следующий алгоритм:
Повтори 5 [Дуга 5, 0, 5, 180 Дуга 5, 5, 0, 180 Дуга 5, 0, -5, 180 Дуга 5, -5, 0, 180].
Определите, сколько точек с целочисленными координатами будут находиться внутри области, ограниченной линией, заданной данным алгоритмом. Точки на линии учитывать не следует.

Решение

```
import turtle
t=turtle.Turtle()
t.reset()
t.seth(90)
t.width(2)
t.speed(20)
k = 10 #коэффициент для увеличения масштаба
for i in range(5):
    t.seth(0)
    t.circle(5*k,180)
    t.seth(90)
    t.circle(5*k,180)
    t.seth(180)
    t.circle(5*k,180)
    t.seth(270)
    t.circle(5*k,180)
t.penup()
for x in range(-15,6,1):
    for y in range(-5,16):
        t.goto(x*k , y*k )
        t.dot(5)
t.penup()
turtle.mainloop()
```



Возможности модуля Turtle:

- рисование в Python Turtle Graphics
- тренировка методов объектно-ориентированного программирования
- отработка умения привязывать действия Черепахи к ее положению на координатной плоскости
- тренировка работы с цветом
- использование команд, классов и методов в интерактивном режиме