Конспект 1

Strim_6_1

Исполнитель "Черепаха" представляет собой устройство, которое перемещается по плоскости и оставляет за собой следы в виде линий. В начальном положении черепаха находится в точке начала координат (0, 0), при этом её голова направлена вдоль положительного направления оси Y, иными словами, вверх. Когда хвост черепахи опущен, она оставляет на поверхности линию.
Есть несколько команд, которые управляют действиями черепахи:
1. «Вперед n» – эта команда заставляет черепаху двигаться вперёд в направлении, куда указывает её голова.
2. «направо m» – данная команда поворачивает голову черепахи на указанное количество градусов по часовой стрелке.
3. «Цикл» – этот оператор позволяет повторять последовательность команд некоторое количество раз.
В задачах этого типа предоставлены конкретные алгоритмы для управления движением черепахи. Необходимо определить, сколько точек с целочисленными координатами будет находиться внутри области или областей, ограниченных линиями, которые рисует черепаха согласно заданному алгоритму. При этом следует обращать внимания на указание о точках, лежащих непосредственно на границе этой области, в некоторых условиях задачи они должны быть учтены, а в некоторых не учитываются. Разберем решение подобных задач на различных примерах. Все эти задачи являются типовыми для задания № 6 ЕГЭ.

Задача № 1 (5500)

(Демо-2023). 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. 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.

Ссылка на видео-разбор с таймингом: https://vk.com/video-205546952_456241239?t=0h4m45s

Решение

Эта задача заключается в определении количества точек с числовыми координатами, находящихся внутри области, ограниченной линией, описанной этим алгоритмом. При этом точки, лежащие

непосредственно на самой линии, учитываться не должны.
Для решения задачи прежде всего, нужно запустить Python и загрузить модуль черепахи (Turtle), чтобы затем с его помощью нарисовать линию, заданную алгоритмом. После этого требуется расставить целочисленные точки и подсчитать их количество.
После того, запущен Python следует импортировать этот модуль, поскольку он не встроен в стандартную библиотеку Python. Для этого используем следующую команду:

```
from turtle import *
```

Этот код импортирует все основные функции и методы модуля turtle, предоставляя доступ ко всем возможностям черепашки.
Теперь давайте перейдем к настройке среды. Один из важных аспектов работы с модулем Turtle – это управление масштабом. По умолчанию черепашка рисует в пикселях, поэтому важно понимать, как управлять размерами рисунка. Однако для простоты можно временно игнорировать масштабирование и сосредоточиться на основных командах.
Давайте рассмотрим несколько базовых команд для управления движением черепашки:
1. Движение вперед (forward):
forward(длина)
Эта команда перемещает черепашку вперед на указанное расстояние. Например, forward(100) переместит черепашку на 100 пикселей вперед.
2. Поворот вправо (right или rt):
right(угол)
Эта команда поворачивает черепашку направо на указанный угол. Например, right(90) повернет черепашку на 90 градусов вправо.
3. Поворот влево (left или lt):
left(угол)
Аналогично команде поворота вправо, эта команда поворачивает черепашку влево на указанный угол. Например, left(45) повернет черепашку на 45 градусов влево.
4. Движение назад (backward или bk):
backward(длина)
Эта команда перемещает черепашку назад на указанное расстояние. Например, backward(50) переместит черепашку на 50 пикселей назад.
5. Поднять перо (up или pu):
up()
Эта команда поднимает "перо" черепашки, благодаря чему при дальнейшем движении линия не будет рисоваться. Таким образом, можно перемещать черепашку, не оставляя следов.
6. Опустить перо (down или pd):
down()
Эта команда опускает "перо", после чего черепашка снова начнет рисовать линию при движении.
Эти команды являются основными для управления движением черепашки. Они позволяют создавать сложные траектории и фигуры, а также контролировать процесс рисования линий.
Есть и другие полезные команды, которые могут облегчить работу с черепашкой:
7. Очистка экрана (clearscreen или cs):
clearscreen()
Эта команда очищает экран от всех рисунков и возвращает черепашку в начальное положение.
8. Установка позиции (goto или setpos):
goto(x, y)
Эта команда перемещает черепашку в указанную точку координат (x, y) на экране.
9. Получение текущих координат (position или pos):
position()

Эта команда возвращает текущие координаты черепашки в виде кортежа (x, y).
10. Задание цвета пера (pencolor или pc):
pencolor("цвет")
Эта команда задает цвет пера черепашки. Цвет может быть задан именем ("red", "blue") или шестнадцатеричным значением ("#FF0000").
Таким образом, используя эти команды, мы можем эффективно управлять черепашкой и решать задачи, связанные с построением графиков и фигур.
Важно!!! Никогда не называйте файл как модуль, Turtle. Если вы это сделаете, у вас черепаха перестанет работать, пока вы этот файл не удалите.
Итак, после того как подключен модуль turtle. Следует:
1. отключить анимацию (tracer(0)), чтобы сразу увидеть конечный результат без промежуточных шагов
2. установить исходную ориентацию Черепахи вверх (lt(90)), чтобы полученная картинка располагалась в соответствии с заданием.
3. задать большой размер экрана (screensize)
4. указать масштаб (r), чтобы увеличить длину шага для лучшего отображения.

```
from turtle import *
```

```
tracer(0)
```

```
lt(90)
```

```
screensize(10000,10000)
```

```
r = 30
```

Затем, напишем основной алгоритм, увеличивая длину шага для лучшего отображения.

```
for i in range(7):
    fd(10*r)
    rt(120)
```

Создадим сетку из точек с целыми координатами.

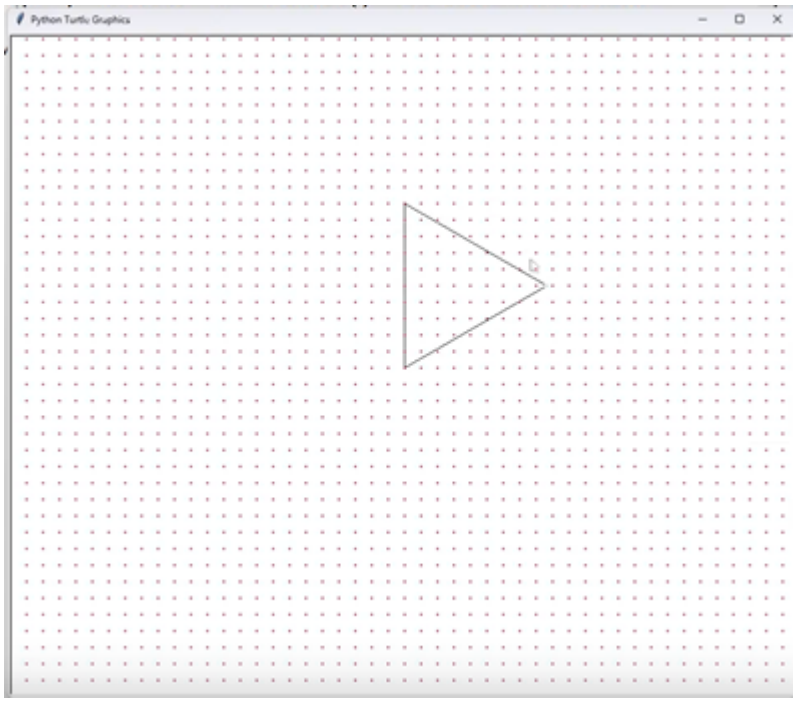
```
up()
for x in range(-50,50):
    for y in range(-50,50):
        goto(x*r,y*r)
        dot(3,'red')
```

Чтобы результат движения черепахи появился на экране следует в самом конце программы записать команду update().
Так будет выглядеть программа целиком:

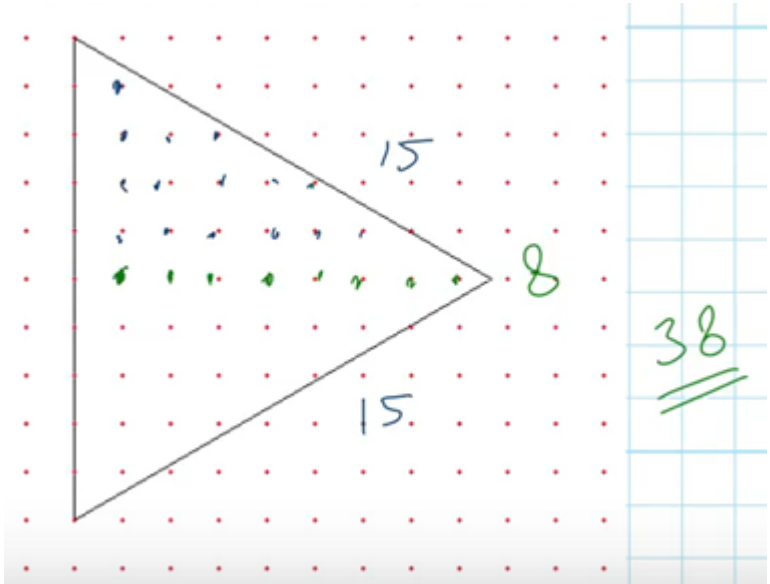
```
from turtle import *
tracer(0) #отключение анимации
lt(90) #направляем голову Черепахи вверх
screensize(10000,10000) #добавляет ползунки
r = 30 #масштаб картинки
#основной алгоритм
```

```
for i in range(7):
    fd(10*r)
    rt(120)
#целочисленная сетка
up()
for x in range(-50,50):
    for y in range(-50,50):
        goto(x*r,y*r)
        dot(3,'red')
update() #Обновление картинки
done() #для PyCharm (или mainloop())
```

Результат работы программы:



После того, как получен рисунок подсчитываем количество точек внутри фигуры.



Ответ: 38

Если вы работаете в PyCharm,чтобы PyCharm не закрывал ваше окно с черепашкой можно последней командой написать done()или main loop(). Main loop – это бесконечный цикл событий.

Задача №2 (5767)

Черепахе был дан для исполнения следующий алгоритм:
Повтори 2 [Вперёд 10 Направо 90 Вперёд 20 Направо 90]
Поднять хвост
Вперёд 3 Направо 90 Вперёд 5 Налево 90
Опустить хвост
Повтори 2 [Вперёд 70 Направо 90 Вперёд 80 Направо 90]
Определите, сколько точек с целочисленными координатами будут находиться внутри пересечения фигур, ограниченных заданными алгоритмом линиями, включая точки на границах этого пересечения.

Ссылка на видео-разбор с таймингом: https://vk.com/video-205546952_456241239?t=0h21m50s

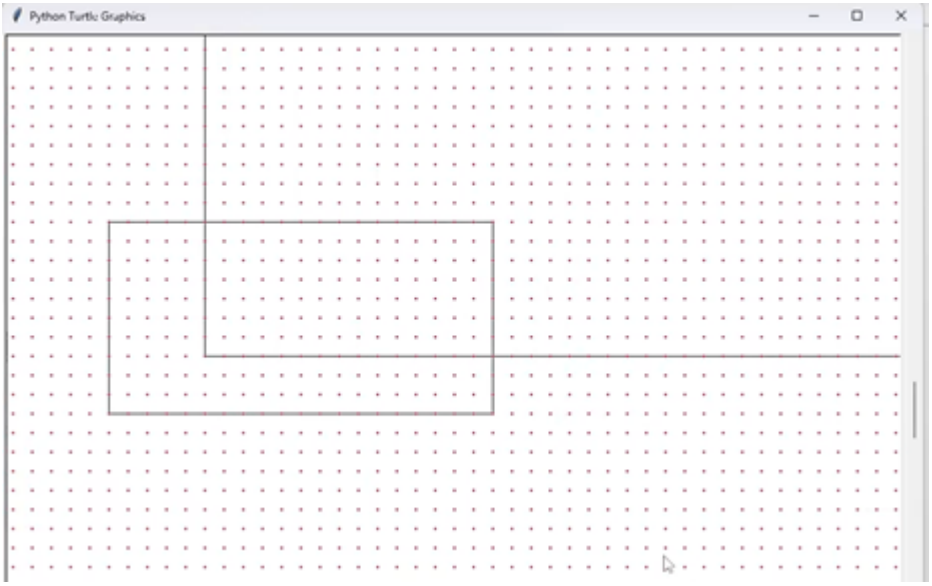
Решение

Сначала импортируем библиотеку turtle. Для начала отключим анимацию (используем команду tracer(0)), чтобы сразу увидеть результат. Затем поворачиваем черепашку на 90 градусов, чтобы она была направлена вверх, и устанавливаем размер экрана (screensize) на 10 000 x 10 000, чтобы удобно просматривать изображение и задаем масштаб равный 20. После этого пишем заданный исполнителю алгоритм. Затем выполнив команду up(), поднять хвост, рисуем координатную сетку. Завершаем программу командой update()

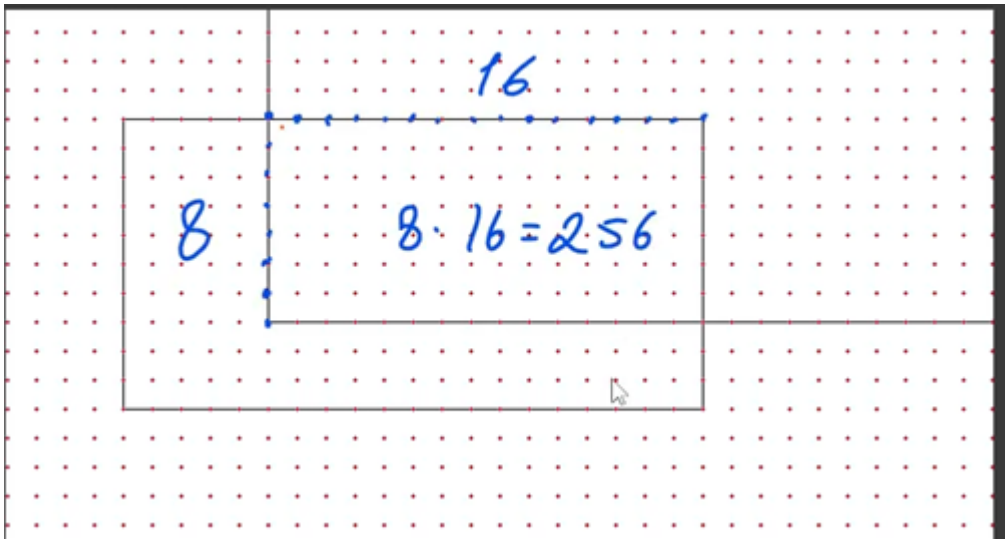
```
from turtle import *
tracer(0)
lt(90)
screensize(10000,10000)
r = 20
for i in range(2):
    fd(10*r)
    rt(90)
    fd(20*r)
    rt(90)
up()
fd(3*r)
rt(90)
fd(5*r)
lt(90)
down()
for i in range(2):
    fd(70*r)
    rt(90)
    fd(80*r)
    rt(90)
up()
for x in range(-50,50):
    for y in range(-50,50):
```

```
        goto(x*r,y*r)
        dot(3,'red')
update()
```

Результат работы программы:



Для получения ответа следует сосчитать количество точек в пересечении фигур, включая точки на линии:



Ответ: 128

Как сосчитать количество точек в объединении фигур?

Ссылка на видео-разбор с таймингом: https://vk.com/video-205546952_456241239?t=0h31m10s

Объединение – это общий набор точек во всех представленных фигурах, будь то маленькая или большая. Обратите внимание, что точек в полученном программой рисунке может быть недостаточно для покрытия всей фигуры.

Как же это сделать? Конечно, не вручную! Хотя пересечение можно посчитать вручную, точки в отдельных прямоугольниках подсчитываются довольно просто. Рассмотрим принцип счета точек. Предположим, у нас есть небольшой прямоугольник размером 3x6 (где 3 – длина, а 6 – ширина). Сколько точек содержится в нем?
Если учитывать все точки, включая те, которые находятся на границах, то вдоль вертикальной оси у нас будет 4 точки, а вдоль горизонтальной – 7. Таким образом, количество точек всегда на единицу больше, чем размеры сторон прямоугольника. Это объясняется тем, что каждая сторона имеет начальную точку, конечную точку и еще одну замыкающую. По аналогии с забором: если у нас есть три жерди, то пролетов между ними будет два.
Теперь, зная эту закономерность, мы можем легко определить количество точек в любом прямоугольнике. В первом прямоугольнике размер составляет 10x20, значит, количество точек равно 11x21, т.е. 231. Во втором прямоугольнике, который значительно больше, размер равен 71x81, что дает нам 5751 точку.
Однако, при сложении количества точек двух прямоугольников возникает проблема: пересечение было учтено дважды. Чтобы исправить это, необходимо вычесть количество точек в пересечении.
Итак, итоговое количество точек составит:
231 + 5751 - 128 = 5854.

Задача № 3 (5770)

Черепахе был дан для исполнения следующий алгоритм:
Повтори 2 [Вперёд 7 Направо 60 Вперёд 12 Направо 120]
Поднять хвост
Вперёд 7 Направо 60
Опустить хвост
Повтори 2 [Вперёд 5 Направо 120 Вперёд 10 Направо 60]
Определите, сколько точек с целочисленными координатами будут находиться внутри пересечения фигур, ограниченных заданными алгоритмом линиями, не включая точки на границах этого пересечения.

Ссылка на видео-разбор с таймингом: https://vk.com/video-205546952_456241239?t=0h36m45s

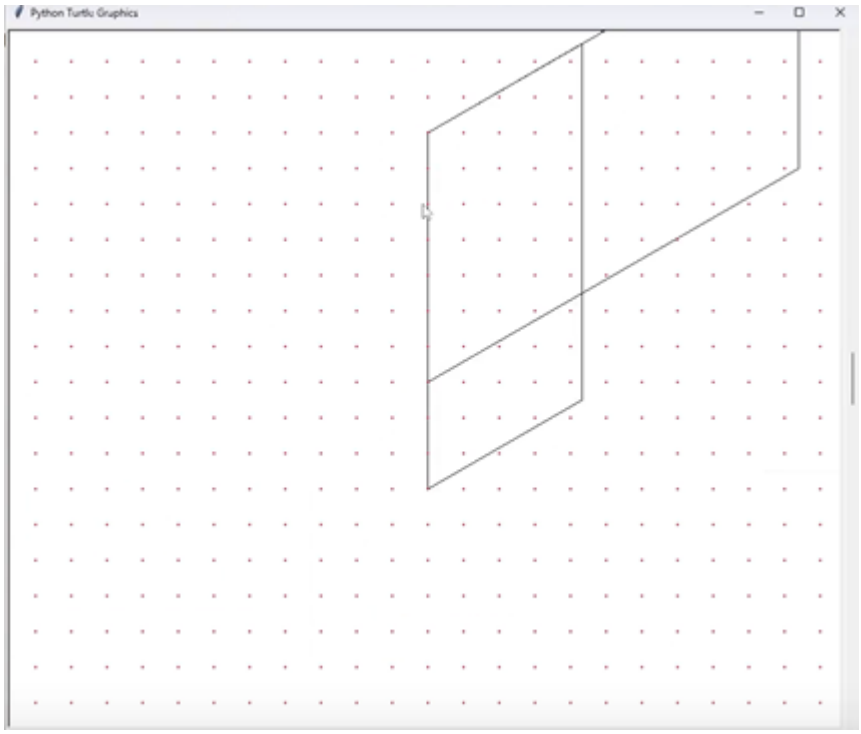
Решение

Импортируем библиотеку turtle. Отключаем анимацию (используем команду tracer(0)), чтобы сразу увидеть результат. Затем поворачиваем черепашку на 90 градусов, чтобы она была направлена вверх, и устанавливаем размер экрана (screensize) на 10 000 x 10 000, чтобы удобно просматривать изображение и задаем масштаб равный 40 (так будет удобнее сосчитать точки). После этого пишем заданный исполнителю алгоритм. Затем выполнив команду up(), поднять хвост, рисуем координатную сетку. Завершаем программу командой update()

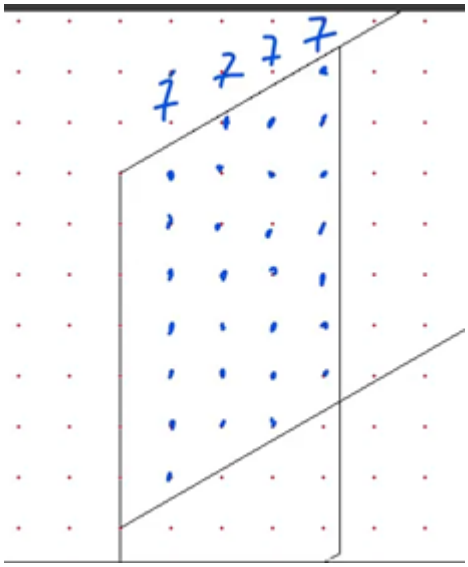
```
from turtle import *

tracer(0)
lt(90)

screensize(10000,10000)

r = 40
```

```
for i in range(2):
    fd(7*r)

    rt(60)

    fd(12*r)

    rt(120)

up()

fd(7*r)

rt(60)

down()

for i in range(2):

    fd(5*r)

    rt(120)

    fd(10*r)

    rt(60)

up()
for x in range(-50,50):

    for y in range(-50,50):

        goto(x*r,y*r)

        dot(3,'red')

update()
```

Результат работы программы:



Сосчитаем точки внутри фигуры, не считая точки на линии



Ответ: 28

Задача № 4 (5937)

Черепахе был дан для исполнения следующий алгоритм:
Повтори 2 [Вперёд 5 Направо 90 Вперёд 8 Направо 90]
Поднять хвост
Вперёд 4 Направо 90 Вперёд 3 Налево 90
Опустить хвост
Повтори 2 [Вперёд 4 Направо 90 Вперёд 9 Направо 90]
Выполняя этот алгоритм, Черепаха рисует одну за другой две фигуры. Определите, сколько точек с целочисленными координатами будут находиться внутри первой нарисованной фигуры, но не внутри второй. Точки на границах указанной области следует учитывать.

Ссылка на видео-разбор с таймингом: https://vk.com/video-205546952_456241239?t=0h42m30s

Решение

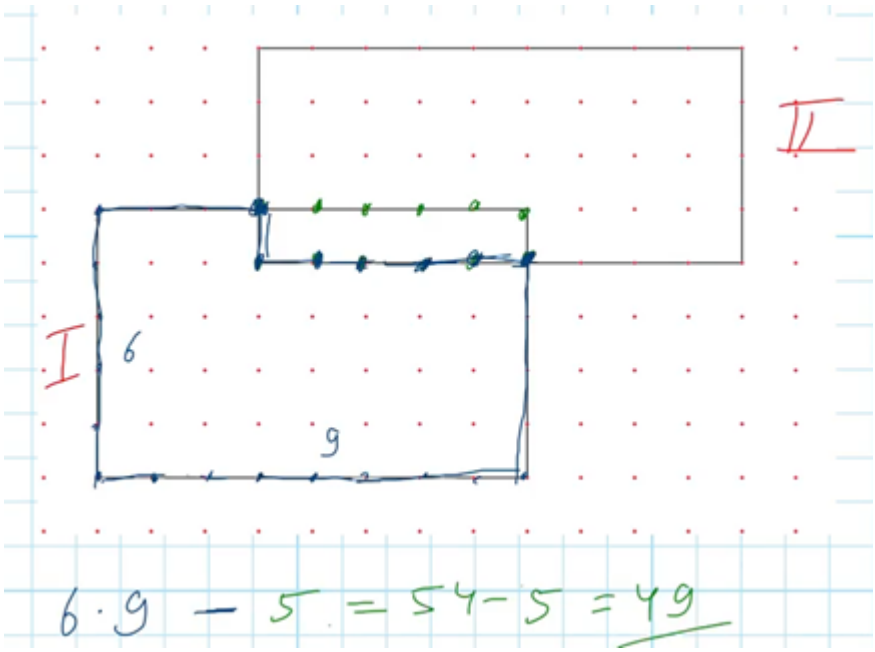
Импортируем библиотеку turtle. Отключаем анимацию (используем команду tracer(0)), чтобы сразу увидеть результат. Затем поворачиваем черепашку на 90 градусов, чтобы она была направлена вверх, и устанавливаем размер экрана (screensize) на 10 000 x 10 000, чтобы удобно просматривать изображение и задаем масштаб равный 40 (так будет удобнее сосчитать точки). После этого пишем заданный исполнителю алгоритм. Затем выполнив команду up(), поднять хвост, рисуем координатную сетку. Завершаем программу командой update()

```
from turtle import *
tracer(0)
lt(90)
screensize(10000,10000)
r = 40
for i in range(2):
    fd(5*r); rt(90); fd(8*r); rt(90)
up()
fd(4*r); rt(90); fd(3*r); lt(90)
down()
for i in range(2):
    fd(4*r); rt(90); fd(9*r); rt(90)
up()
for x in range(-50,50):
    for y in range(-50,50):
        goto(x*r,y*r)
        dot(3,'red')
update()
```

Результат работы программы:


```
from turtle import *
tracer(0)
lt(90)
screensize(10000,10000)
r = 40
for i in range(2):
    fd(5*r); rt(90); fd(8*r); rt(90)
up()
fd(4*r); rt(90); fd(3*r); lt(90)
down()
for i in range(2):
    fd(4*r); rt(90); fd(9*r); rt(90)
up()
for x in range(-50,50):
    for y in range(-50,50):
        goto(x*r,y*r)
        dot(3,'red')
update()
```

Первая фигура имеет размеры 5 на 8, а вторая – 4 на 9. Понятно, что 5 на 8 относится к первой фигуре. Посчитаем: 1, 2, 3, 4, 5 на 8. Таким образом, это наша первая фигура. Вторая фигура, соответственно, будет иметь размеры 4 на 9.
Исходя из контекста и размеров фигур, становится ясно, какая из них первая, а какая вторая. Нам нужно найти точки, которые находятся внутри первой фигуры. Точки, которые располагаются в области второй фигуры должны быть удалены из расчета.



Ответ: 49

Задача №5 (5939)

Черепахе был дан для исполнения следующий алгоритм:
Повтори 2 [Вперёд 5 Направо 90 Вперёд 8 Направо 90]
Поднять хвост
Вперёд 2 Направо 90 Вперёд 4 Налево 90
Опустить хвост
Повтори 2 [Вперёд 4 Направо 90 Вперёд 9 Направо 90]
Выполняя этот алгоритм, Черепаха рисует одну за другой две фигуры. Определите, сколько точек с целочисленными координатами будут находиться внутри области, полученной при объединении двух фигур. Точки на границах указанной области следует учитывать.

Ссылка на видео-разбор с таймингом: https://vk.com/video-205546952_456241239?t=0h53m0s

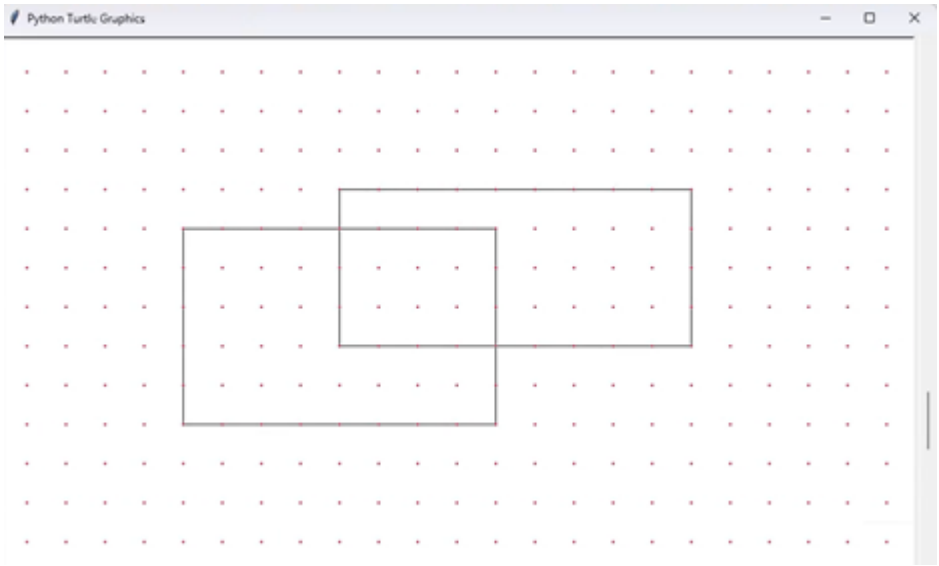
Решение

Импортируем библиотеку turtle. Отключаем анимацию (используем команду tracer(0)), чтобы сразу увидеть результат. Затем поворачиваем черепашку на 90 градусов, чтобы она была направлена вверх, и устанавливаем размер экрана (screensize) на 10 000 x 10 000, чтобы удобно просматривать изображение и задаем масштаб равный 40 (так будет удобнее сосчитать точки). После этого пишем

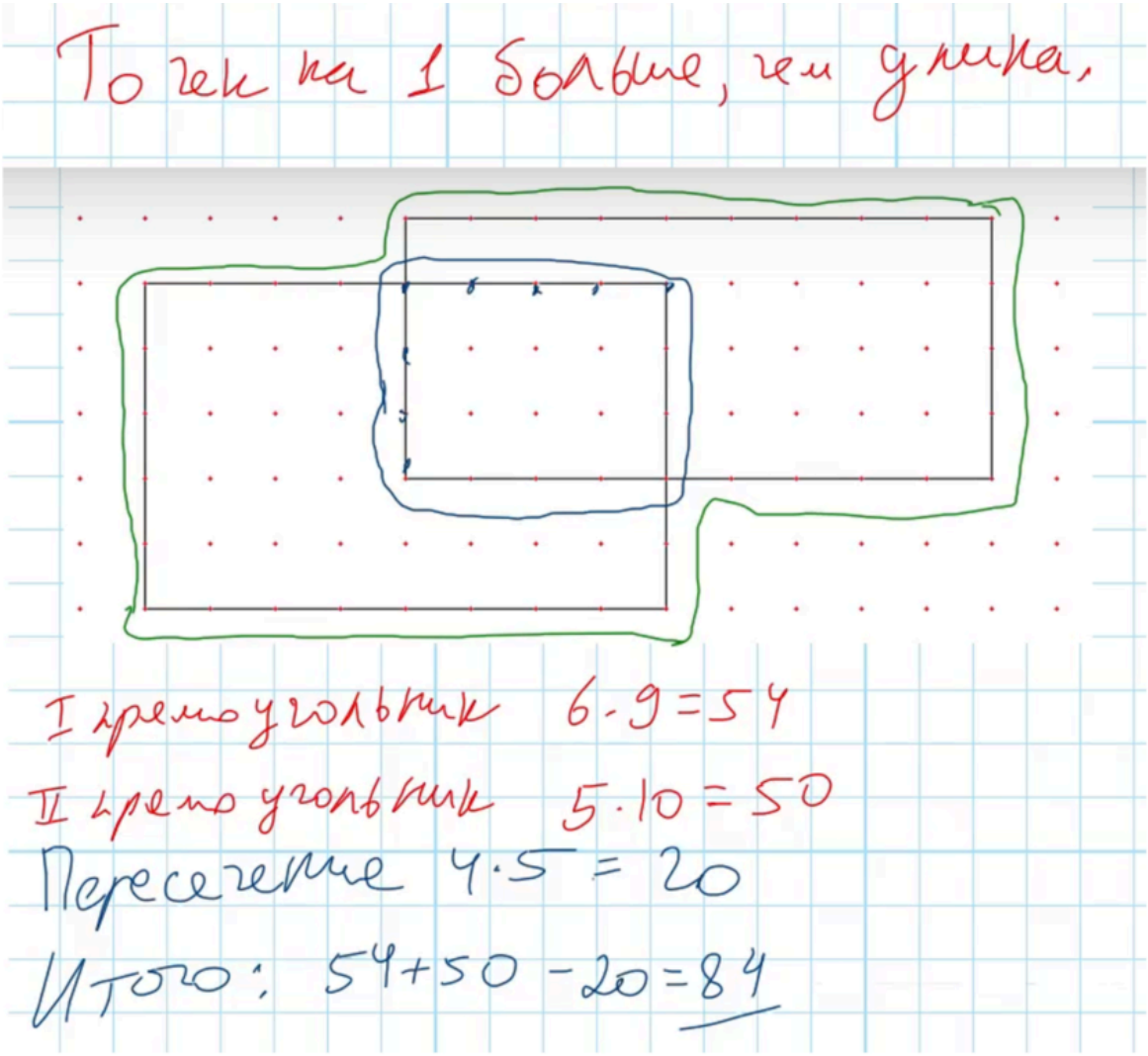
заданный исполнителю алгоритм. Затем выполнив команду up(), поднять хвост, рисуем координатную сетку. Завершаем программу командой update()

```
from turtle import *
tracer(0)
lt(90)
screensize(10000,10000)
r = 40
for i in range(2):
    fd(5*r); rt(90); fd(8*r); rt(90)
up()
fd(4*r); rt(90); fd(3*r); lt(90)
down()
for i in range(2):
    fd(4*r); rt(90); fd(9*r); rt(90)
up()
for x in range(-50,50):
    for y in range(-50,50):
        goto(x*r,y*r)
        dot(3,'red')
update()
```

Результат работы программы:



Сделаем необходимые для получения расчеты:



Ответ: 84

Задание №6 (6911)

Черепахе был дан для исполнения следующий алгоритм:
Повтори 2 [Вперёд 16 Направо 90 Вперёд 9 Направо 90]
Поднять хвост
Вперёд 5 Направо 90 Вперёд 11 Направо 90
Опустить хвост
Повтори 2 [Вперёд 20 Направо 90 Вперёд 8 Направо 90]

Нарисованные Черепахой линии образуют несколько областей, внутри которых нет линий. Определите, сколько точек с целочисленными координатами будут находиться внутри области с наибольшей площадью. Точки, расположенные на контуре области, следует учитывать.

Ссылка на видео-разбор с таймингом: https://vk.com/video-205546952_456241239?t=1h0m15s

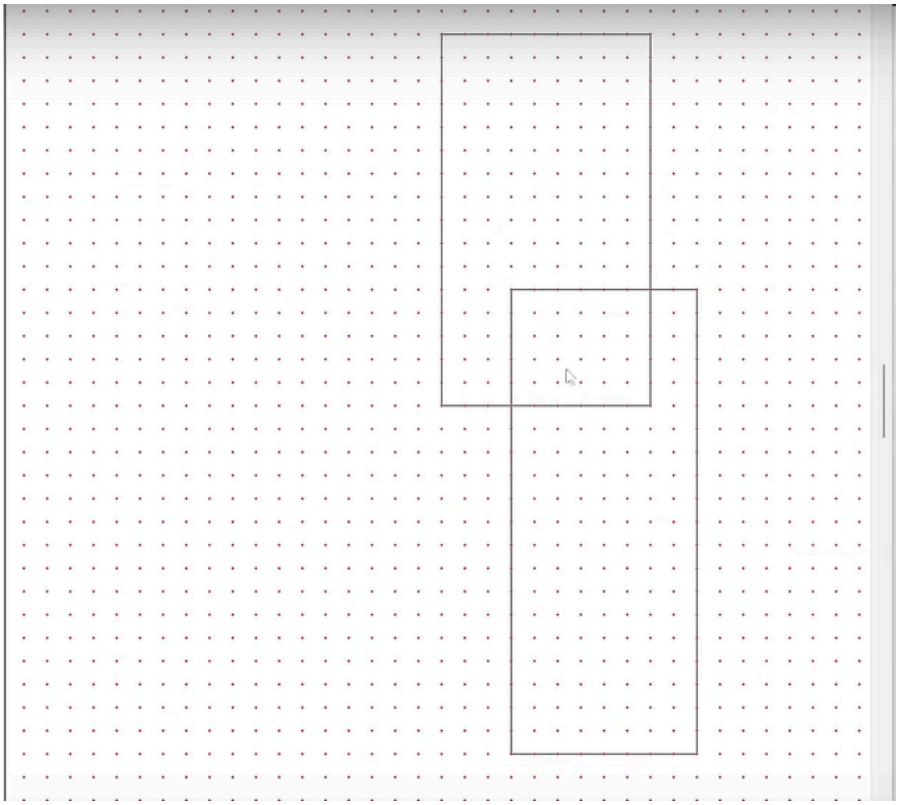
Решение

Импортируем библиотеку turtle. Отключаем анимацию (используем команду tracer(0)), чтобы сразу увидеть результат. Затем поворачиваем черепашку на 90 градусов, чтобы она была направлена вверх, и устанавливаем размер экрана (screensize) на 10 000 x 10 000, чтобы удобно просматривать изображение и задаем масштаб равный 25 (так будет удобнее сосчитать точки). После этого пишем

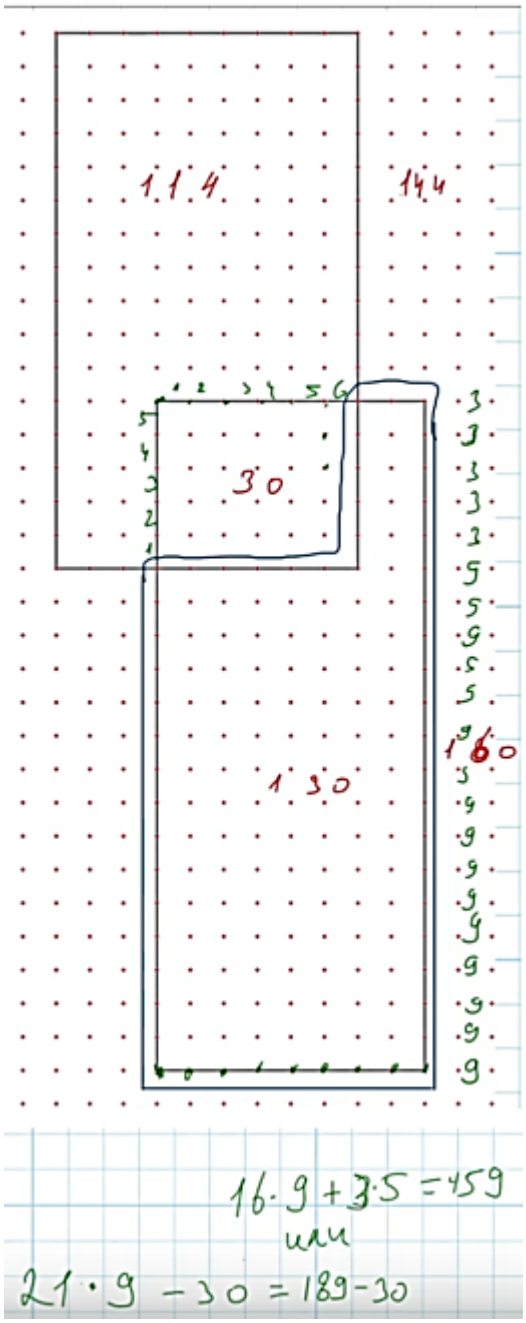
заданный исполнителю алгоритм. Затем выполнив команду up(), поднять хвост, рисуем координатную сетку. Завершаем программу командой update()

```
from turtle import *
tracer(0)
lt(90)
screensize(10000,10000)
r = 25
for i in range(2):
    fd(16*r); rt(90); fd(9*r); rt(90)
up()
fd(5*r); rt(90); fd(11*r); rt(90)
down()
for i in range(2):
    fd(20*r); rt(90); fd(8*r); rt(90)
up()
for x in range(-50,50):
    for y in range(-50,50):
        goto(x*r,y*r)
        dot(3,'red')
update()
```

Результат работы программы:



Обратите внимание, в каких-то задачах просят точки, в каких-то просят площадь или периметр. Важно не путать длину и количество точек. Так, чтобы посчитать площадь, надо посчитать именно длину в отрезках. Очевидно, что самая большая площадь будет у нижней фигуры.
Произведем подсчет точек внутри этой фигуры с учетом точек на её контуре



Ответ: 159

Задание № 7(7358)

Черепахе был дан для исполнения следующий алгоритм:
Повтори 5 [Вправо 45 Вперед 10 Вправо 45]
Повтори 6 [Вперед 20 Вправо 90]
В каждом из двух циклов Черепаха рисует замкнутый контур. Определите площадь области пересечения фигур, ограниченных этими контурами.

Ссылка на видео-разбор с таймингом: https: https://vk.com/video-205546952_456241239?t=1h26m20s

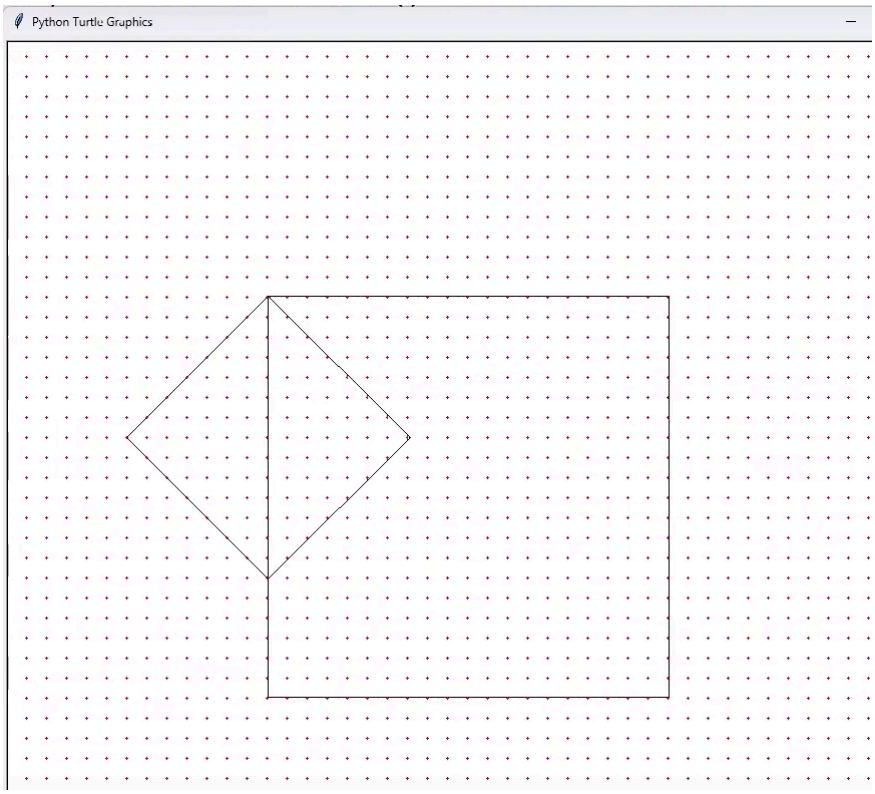
Решение

Импортируем библиотеку turtle. Отключаем анимацию (используем команду tracer(0)), чтобы сразу увидеть результат. Затем поворачиваем черепашку на 90 градусов, чтобы она была направлена вверх, и устанавливаем размер экрана (screensize) на 10 000 x 10 000, чтобы удобно просматривать

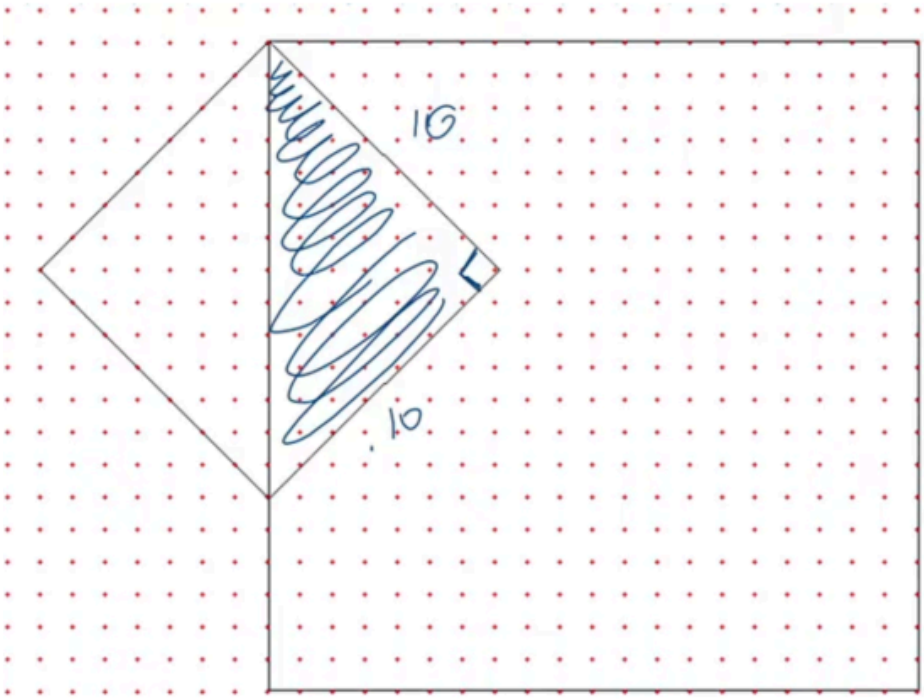
изображение и задаем масштаб равный 20 (так будет удобнее сосчитать точки). После этого пишем заданный исполнителю алгоритм. Затем выполнив команду up(), поднять хвост, рисуем координатную сетку. Завершаем программу командой update().

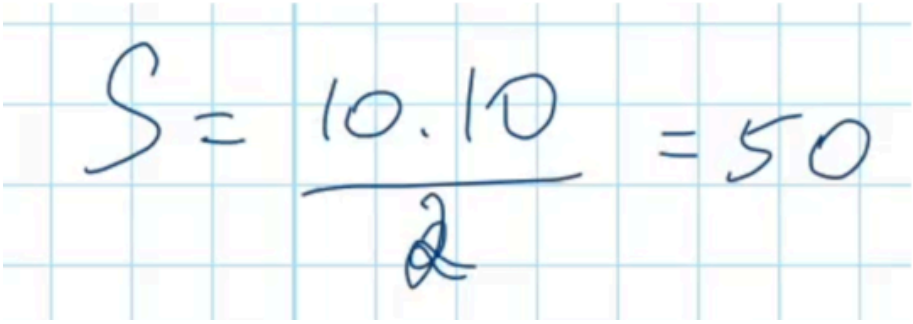
```
from turtle import *
tracer(0)
lt(90)
screensize(10000,10000)
r = 20
for i in range(5):
    rt(45); fd(10*r); rt(45)
for i in range(6):
    fd(20*r); rt(90)
up()
for x in range(-50,50):
    for y in range(-50,50):
        goto(x*r,y*r)
        dot(3,'red')
update()
```

Результат работы программы:



Нам надо найти площадь пересечения этих фигур. Мы можем увидеть, что фигура на пересечении – это половина малого квадрата.





Ответ: 50

Задание №8 (7459)

(ЕГЭ-2024)
Черепахе был дан для исполнения следующий алгоритм:
Повтори 4 [Вперёд 28 Направо 90 Вперёд 26 Направо 90]
Поднять хвост
Вперёд 8 Направо 90 Вперёд 7 Налево 90
Опустить хвост
Повтори 4 [Вперёд 67 Направо 90 Вперёд 98 Направо 90]

Определите площадь пересечения фигур, ограниченных заданными алгоритмом линиями.

Ссылка на видео-разбор с таймингом: https://vk.com/video-205546952_456241239?t=1h26m20s

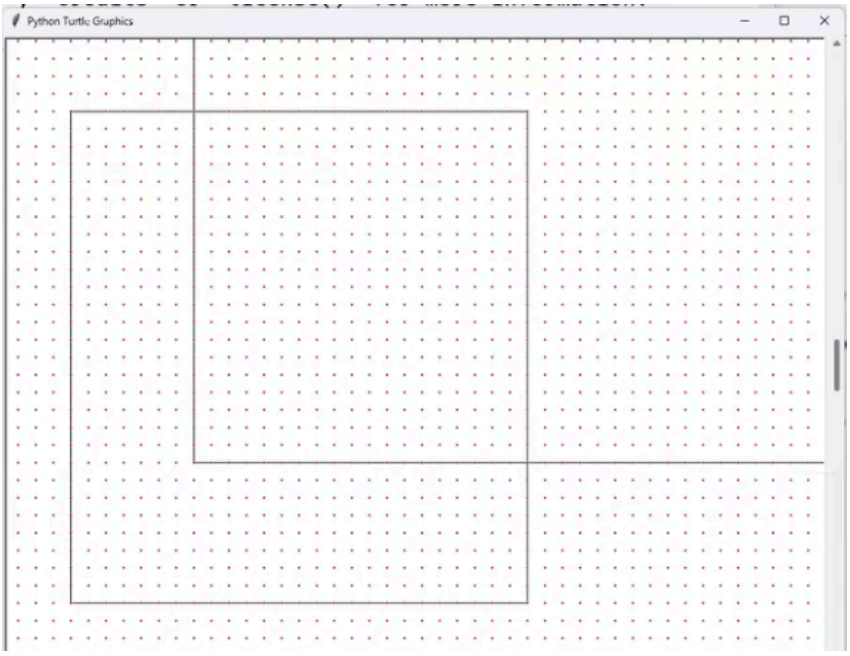
Решение

Импортируем библиотеку turtle. Отключаем анимацию (используем команду tracer(0)), чтобы сразу увидеть результат. Затем поворачиваем черепашку на 90 градусов, чтобы она была направлена вверх, и устанавливаем размер экрана (screensize) на 10 000 x 10 000, чтобы удобно просматривать изображение и задаем масштаб равный 20 (так будет удобнее сосчитать точки). После этого пишем

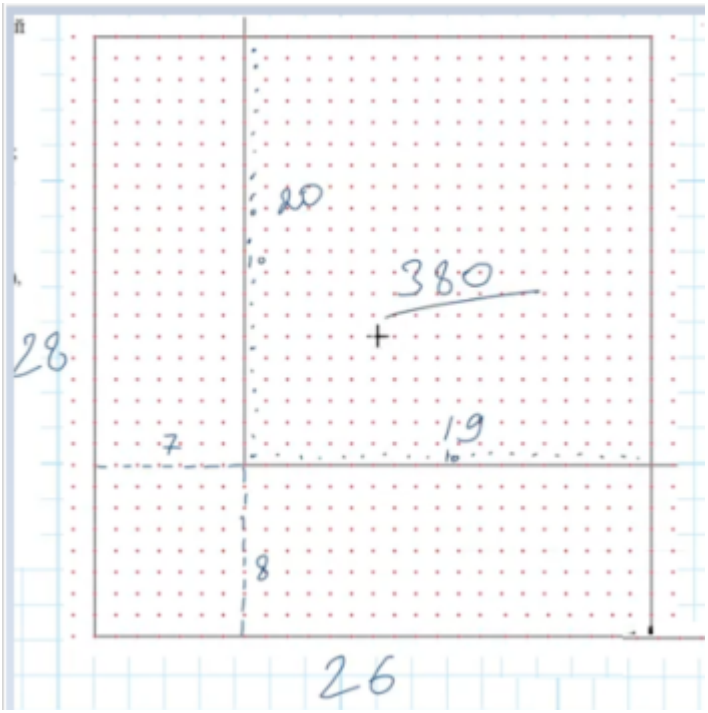
заданный исполнителю алгоритм. Затем выполнив команду up(), поднять хвост, рисуем координатную сетку. Завершаем программу командой update().

```
from turtle import *
tracer(0)
lt(90)
screensize(10000,10000)
r = 20
for i in range(4):
    fd(28*r); rt(90); fd(26*r); rt(90)
up()
fd(8*r); rt(90); fd(7*r); lt(90)
down()
for i in range(4):
    fd(67*r); rt(90); fd(98*r); rt(90)
up()
for x in range(-50,50):
    for y in range(-50,50):
        goto(x*r,y*r)
        dot(3,'red')
update()
```

Результат работы программы:



Найдем площадь области на пересечении фигур.



Ответ: 380

Задание №9(7638)

Черепахе был дан для исполнения следующий алгоритм:
Повтори 9 [Вперёд 12 Направо 90 Вперёд 6 Направо 90]
Поднять хвост
Вперёд 1 Направо 90 Вперёд 3 Налево 90
Опустить хвост
Повтори 9 [Вперёд 53 Направо 90 Вперёд 75 Направо 90]

Определите периметр области пересечения фигур, ограниченных заданными алгоритмом линиями.

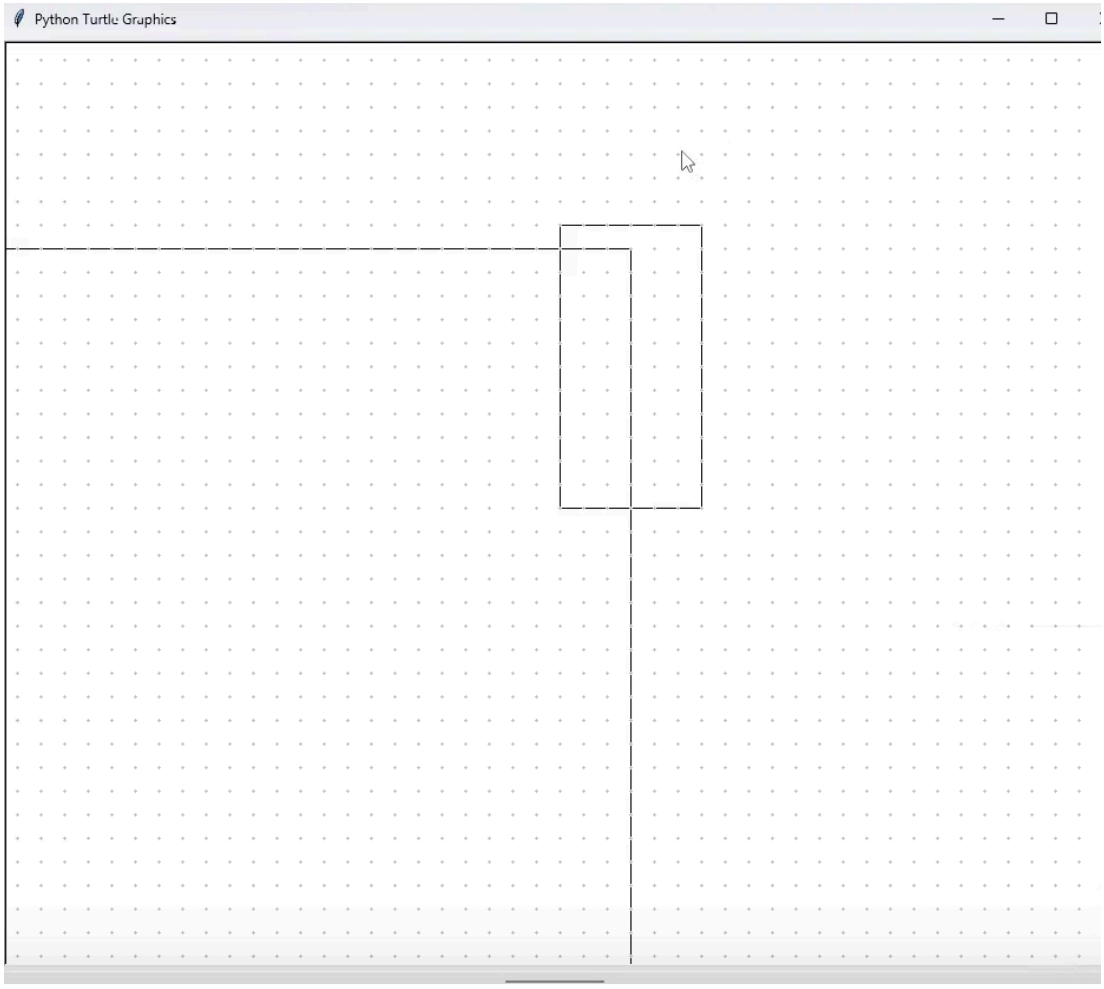
Ссылка на видео-разбор с таймингом: https://vk.com/video-205546952_456241239?t=1h37m35s

Решение
Импортируем библиотеку turtle. Отключаем анимацию (используем команду tracer(0)), чтобы сразу увидеть результат. Затем поворачиваем черепашку на 90 градусов, чтобы она была направлена вверх, и устанавливаем размер экрана (screensize) на 10 000 x 10 000, чтобы удобно просматривать изображение и задаем масштаб равный 20 (так будет удобнее сосчитать точки). После этого пишем заданный исполнителю алгоритм. Затем выполнив команду up(), поднять хвост, рисуем координатную сетку. Завершаем программу командой update().

```
from turtle import *
tracer(0)
lt(90)
screensize(10000,10000)
r = 20
for i in range(9):
    fd(12*r); rt(90); fd(6*r); rt(90)
up()
fd(1*r); rt(90); fd(3*r); lt(90)
down()
```

```
for i in range(9):
    fd(53*r); rt(90); fd(75*r); rt(90)
up()
for x in range(-50,50):
    for y in range(-50,50):
        goto(x*r,y*r)
        dot(3,'silver')
update()
```

Результат работы программы:



Вспомним, что длины считаются в отрезках:

Ответ: 28

Периметр объединения фигур.

Ссылка на видео-разбор с таймингом: https://vk.com/video-205546952_456241239?t=1h42m5s

Нам нужна сумма длин сторон малого прямоугольника и длин сторон большого прямоугольника. При этом нам необходимо вычесть сумму длин линий, которые находятся внутри фигуры, это периметр пересечения, найденный нами выше.

$P_{пересечения} = 2 \cdot (3 + 11) = 28$

$P_{объединения} = 2(12+6) + 2 \cdot (53+75) - P_{пересечения} = 264$