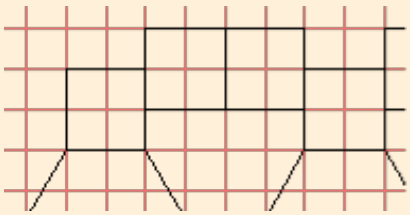
Zadanie Ogniwa – miniLOGIA 16 (2017/18), etap 3

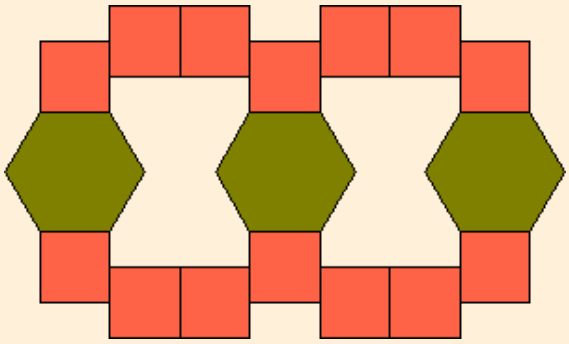
Treść zadania

Napisz dwuparametrową procedurę/funkcję **ogniwa**, po wywołaniu której na środku ekranu powstanie rysunek łańcuszka złożonego z dwukolorowych ogniw powstałych z kwadratowych i sześciokątnych koralików. Pierwszy parametr określa liczbę sześciokątnych koralików w łańcuszku i może przyjmować wartości od **3** do **10**. Drugi parametr określa liczbę kwadratowych koralików w najdłuższym rządku ogniwa i może przyjmować wartości od **2** do **14**. Szerokość rysunku wynosi **700**.

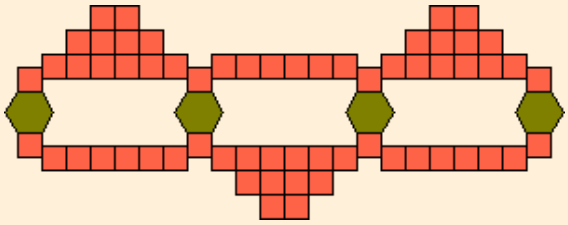


Rysunek pomocniczy

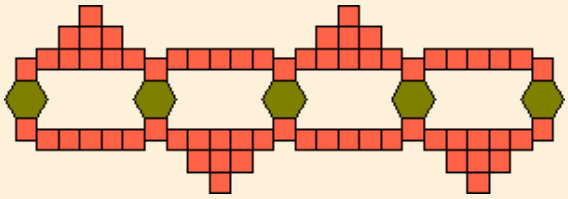
Przykłady:



efekt wywołania:
Logo – **ogniwa 3 2**
Python – **ogniwa (3, 2)**



efekt wywołania:
Logo – **ogniwa 4 6**
Python – **ogniwa (4, 6)**



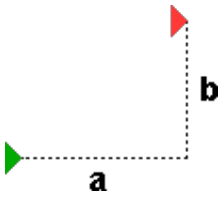
efekt wywołania:
Logo – **ogniwa 5 5**
Python – **ogniwa (5, 5)**

Omówienie rozwiązania

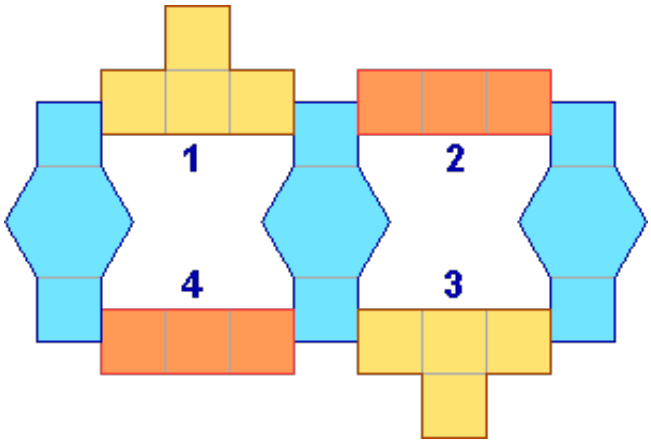
Przyjrzyjmy się uważnie treści zadania. Rysowane są dwa rodzaje figur: kwadraty i sześciokąty o takiej samej długości boku. Długość boków figur zależy od wartości obu parametrów funkcji. Szerokość rysunku wynosi 700, liczba tworzących łańcuszek ogniw jest o jeden mniejsza od liczby sześciokątów. Aby otrzymać długość boku kwadratu należy policzyć ich liczbę na całej długości łańcuszka. W tym celu liczbę ogniw mnożymy przez liczbę kwadratów w najdłuższym rządku ogniwa zwiększoną o 1 (kwadrat przy lewym sześciokącie ogniwa). Do otrzymanej wartości dodajemy 2, biorąc pod uwagę kwadrat rysowany przy ostatnim sześciokącie po prawej stronie łańcuszka oraz wystające fragmenty

sześciokątów na końcach łańcuszka. Następnie dzielimy 700 przez uzyskaną liczbę. Znając długość boku kwadratu i sześciokąta możemy przystąpić do rysowania.

Zauważmy, że przydatna może okazać się funkcja **skok**, przemieszczająca żółwia o podaną liczbę kroków w poziomie i pionie. Na rysunku pomocniczym widzimy wynik działania funkcji **skok(a, b)** w Pythonie. Korzystanie z tej funkcji znacznie uprasza opis poruszania się żółwia, jednak wymaga dokładnego wyliczenia wartości, o jakie chcemy go przemieścić.



Rysunek ogniw opisanych w treści zadania warto podzielić na powtarzające się fragmenty. Zaczynamy od napisania i przetestowania pomocniczych funkcji rysujących kwadrat i sześciokąt. Kolejnym krokiem będzie napisanie funkcji rysującej układ złożony z sześciokąta i dwóch kwadratów (kolor niebieski na rysunku pomocniczym). Przydatna może się także okazać funkcja rysująca pas o podanej długości oraz piramidę złożone z kwadratów (zaznaczone odpowiednio na żółto i pomarańczowo).



Przykładowy podział rysunku na powtarzające się fragmenty

Pisząc funkcję **piramida()** wygodnie jest wykorzystać przygotowaną wcześniej funkcję rysującą pasek z kwadratów, pamiętając, że każdy kolejny rządek kwadratów jest o 2 krótszy, a przed narysowaniem każdego rządka należy odpowiednio przemieścić żółwia. Ostatnim krokiem będzie zbudowanie całego rysunku z poszczególnych elementów. Najpierw możemy narysować wszystkie układy sześciokątów i kwadratów, następnie leżące pomiędzy nimi pasy i piramidy. Zauważmy, że, jeśli zaczniemy od lewego górnego elementu idąc najpierw w prawo, a następnie w dolnej części ogniw od prawej do lewej – pasy i piramidy występują na przemian. Zatem do narysowania ich wystarczy jedna pętla, w której w połowie przemieścimy żółwia do początku pierwszego dolnego elementu. Kolejność rysowania została pokazana na przykładowym rysunku. Wybór elementu do rysowania możemy uzależnić od parzystości zmiennej sterującej pętlą.

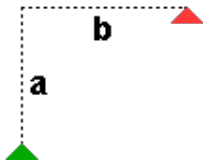
Należy pamiętać, by pisząc funkcję **ogniwa** opisaną w treści zadania zadbać nie tylko o prawidłową wielkość rysunku, ale także o jego wyśrodkowanie – zarówno w poziomie (co jest proste ze względu na stałą szerokość rysunku), jak i w pionie.

Bardzo istotne w tego typu zadaniach jest także ustalenie punktu początkowego, od którego zaczynamy rysowanie danego elementu. Wpływa to na miejsce, do którego będziemy przemieszczać żółwia, składając poszczególne elementy w całość. Najwygodniej jest zwykle kończyć rysowanie elementu w tym samym punkcie, w którym zaczynaliśmy.

Rozwiązanie w języku Python

```
1.  from turtle import *
2.  from math import *
3.
4.  def skok(a, b):
5.      pu(); fd(a); lt(90); fd(b); rt(90); pd()
6.
7.  def kwadrat(a):
8.      fillcolor("tomato");
9.      begin_fill()
10.     for i in range(4):
11.         fd(a); lt(90)
12.     end_fill()
13.
14. def szesc(a):
15.     fillcolor("olive");
16.     begin_fill()
17.     for i in range(6):
18.         fd(a); lt(60)
19.     end_fill()
20.
21. def pion(a):
22.     kwadrat(a)
23.     skok(0, a)
24.     szesc(a)
25.     skok(0, sqrt(3) * a)
26.     kwadrat(a)
27.     skok(0, -(1 + sqrt(3)) * a)
28.
29. def pas(a, n):
30.     for i in range(n):
31.         kwadrat(a)
32.         skok(a, 0)
33.     skok(-n * a, 0)
34.
35. def piramida(a, n):
36.     for i in range(n, 0, -2):
37.         pas(a,i)
38.         skok(a, a)
39.     x = n//2 + n%2
40.     skok(-x * a, -x * a)
41.
42. def ogniwa(n, m):
43.     a = 700 / ((n - 1) * (m + 1) + 2)
44.     skok(-350 + a / 2, -a - sqrt(3) * a / 2)
45.     for i in range(n):
46.         pion(a); skok((m + 1) * a, 0)
47.     skok(-n * ( m + 1) * a, 0)
48.     skok(a, (3 / 2 + sqrt(3)) * a)
49.     for i in range(2 * n - 2):
50.         if i%2 == 0:
51.             piramida(a, m)
52.         else:
53.             pas(a, m)
54.         skok((m + 1) * a, 0)
55.         if i == n - 2:
56.             lt(180); skok(a, (1 + sqrt(3)) * a)
57.     skok(-350 + a / 2, -a / 2 - sqrt(3) * a / 2); lt(180)
```

Rozwiązanie w języku Logo




```
oto skok :a :b
    pod np :a pw 90 np :b lw 90 opu
już

oto kwadrat :a
    powtórz 4 [np :a pw 90]
    skok :a/2 :a/2 ukm "jasnoczerwony zamaluj skok (-:a/2) (-:a/2)
już

oto szesc :a
    powtórz 6 [np :a pw 60]
    skok (pwk 3)*:a/2 :a/2 ukm "oliwkowy zamaluj skok (-(pwk 3)*:a/2) (-:a/2)
już

oto pion :a
    kwadrat :a
    skok :a 0
    lw 30 szesc :a pw 30
    skok (pwk 3)*:a 0
    kwadrat :a
    skok (-(1+pwk 3)*:a) 0
już

oto pas :a :n
    powtórz :n [kwadrat :a skok 0 :a]
    skok 0 (-:n*:a)
już

oto piramida :a :n
    niech "ile (ilorazc :n 2)+reszta :n 2
    powtórz :ile [pas :a :n  skok :a :a niech "n :n-2]
    skok (-:ile*:a) (-:ile*:a)
już

oto ogniwa1 :n :m
    niech "a 700/((:n-1)*(:m+1)+2)
    skok (-:a-(pwk 3)*:a/2) (-350+:a/2)
    powtórz :n [pion :a skok 0 (:m+1)*:a] skok 0 (-:n*(:m+1)*:a)
    skok (3/2+pwk 3)*:a :a
    powtórz 2*:n-2 [
                    jeżeli (reszta npw 2)=1 [piramida :a :m]
                                            [pas :a :m]
                    skok 0 (:m+1)*:a
                    jeżeli npw=:n-1 [pw 180 skok (1+pwk 3)*:a :a]
                   ]
    pw 180
    skok ((1+pwk 3)*:a/2) (350-:a/2)
już
```

Różnice w przemieszczaniu żółwia między rozwiązaniem w języku Logo i Python wynikają z innego jego położenia startowego. Na początku, po uruchomieniu programu, żółw w Pythonie jest skierowany w prawo, w Logo do góry. Także kwadrat i sześciokąt w Logo są rysowane w prawą stronę, a w Pythonie były w lewą. Możliwe jest dosłowne przeniesienie definicji z jednego języka do drugiego, należy jednak wtedy pamiętać, by przed rozpoczęciem rysowania ogniw dodać dodatkowy obrót żółwia o 90 stopni.

Testy

Testowanie rozwiązania rozpoczynamy od przykładów zawartych w treści zadania. Potem testujemy działanie programu dla wartości skrajnych parametrów, wartości parzystych i nieparzystych itp. Warto sprawdzić poprawność rozwiązania dla wszystkich możliwych kombinacji wartości parametrów określonych w treści zadania: pierwszy parametr od 3 do 10, drugi od 2 do 14.

W języku Python, aby przyspieszyć tworzenie rysunku przez żółwia, stosujemy wywołanie złożone z funkcji **tracer()** – rysownie w pamięci, właściwego wywołania funkcji **ogniwa()** i na końcu uaktualniamy ekran za pomocą funkcji **update()**. Przykład:

```
tracer(0)
ogniwa(3, 4)
update()
```

Powrót do standardowego trybu rysowania uzyskamy wywołując funkcję **tracer()** z parametrem równym 1.