Zadanie Gwiazdy – LOGIA 19 SP (2018/19), etap 2

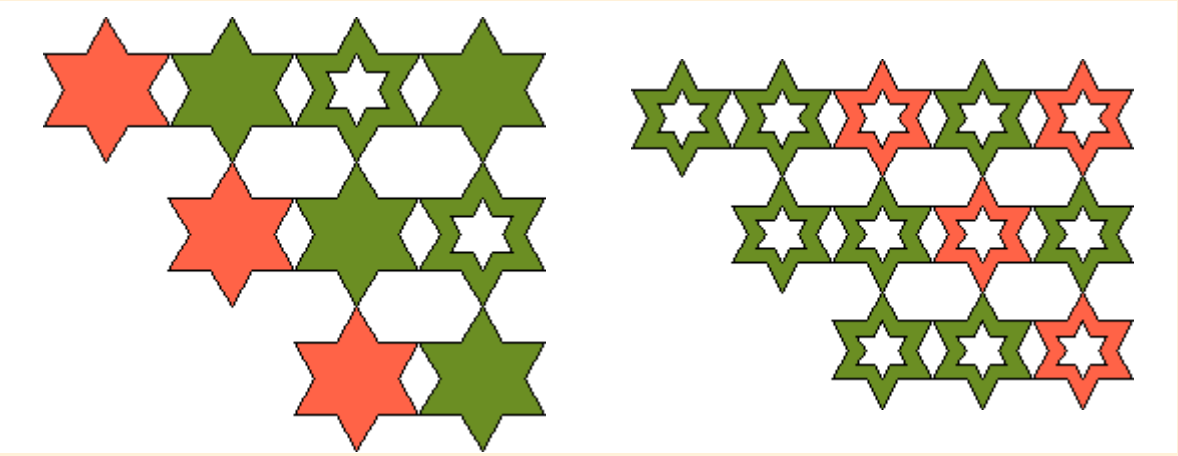
Treść zadania

Firma produkująca ozdoby zakupiła maszynę sterowaną za pomocą kodów liczbowych. Maszyna produkuje ozdoby ułożone w trzech wierszach. Pierwszy wiersz zawiera elementy ozdoby (gwiazdki) odpowiadające kolejnym cyfrom kodu liczbowego, drugi odpowiada kodowi bez ostatniej cyfry, trzeci – kodowi bez dwóch ostatnich cyfr. Maszyna wykorzystuje cztery rodzaje gwiazdek odpowiadające cyfrom. Długość boku wewnętrznej gwiazdki jest równa połowie długości boku zewnętrznej.

0 1 2 3 4 5 6 7 8 9

Napisz jednoparametrową procedurę/funkcję **gwiazdy**, po wywołaniu której na środku ekranu powstanie rysunek układu dwukolorowych gwiazdek. Parametrem jest liczba z zakresu od 1000 do 1999999999. Szerokość rysunku wynosi 560.

Przykłady:



gwiazdy(2490) **gwiazdy(11357)**

Omówienie rozwiązania

Zadanie polega na rysowaniu układów złożonych z czterech rodzajów gwiazdek: czerwonych i zielonych, z wycięciem w środku lub bez. Gwiazdki rysowane są w trzech rządkach wyrównanych do prawej, przy czym każdy kolejny rządek jest o jedną gwiazdkę krótszy. Rodzaj kolejnych gwiazdek w pierwszym wierszu układu został opisany za pomocą kodu liczbowego – każdej cyfrze przyporządkowana została pewna gwiazdka zgodnie z systemem zdefiniowanym w treści zadania. Możemy zauważyć, że układ gwiazdek odpowiadających kolejnym cyfrom powtarza się cyklicznie co 4 pozycje – można to wykorzystać pisząc funkcję rysującą kolejne gwiazdki, typ gwiazdki określimy za pomocą operacji modulo (reszta z dzielenia przez 4). Warto przygotować funkcje pomocnicze: **gw()** rysującą kolorową gwiazdkę oraz **gwgw()** tworzącą całą ozdobę. Ich parametrem będzie szerokość elementu oraz odpowiednio kolor gwiazdki i typ (numer) ozdoby. Żółw rozpoczyna i kończy rysowanie w środku elementu.

Kolejne wiersze rysowanego układu wymagają analizowania liczby, będącej parametrem funkcji, cyfra po cyfrze. W tym celu badamy kolejno resztę z dzielenie liczby przez 10 oraz iloraz całkowity liczby przez 10. By nie powtarzać tych czynności wielokrotnie przy rysowaniu kolejnych rzędów możemy utworzyć listę złożoną z cyfr liczby, przy czym kolejność cyfr na liście powinna być taka sama, jak w badanej liczbie.

Szerokość tworzonego rysunku jest stała i wynosi 560. Wysokość zależy od liczby cyfr w kodzie opisującym układ gwiazdek. Przy wyliczaniu wysokości rysunku należy wziąć pod uwagę odległość między czubkami przeciwległych ramion gwiazdki, czyli skorzystać z wzoru na wysokość w trójkącie równobocznym. Obie te wartości, szerokość i wysokość rysunku, wykorzystujemy do jego wyśrodkowania.

Rozwiązanie w języku Python

```
from turtle import *
from math import sqrt

def gw(a, k):
    pu(); lt(60); bk(a/3); rt(60); pd()
    fillcolor(k);
    begin_fill()
    for i in range(6):
        rt(60); fd(a/3); lt(120); fd(a/3);
    end_fill()
    pu(); lt(60); fd(a/3); rt(60); pd()

def gwgw(a, n):
    # dla danej cyfry liczę resztę z dzielenia przez 4
    # by wybrać jeden z 4 typów ozdób
    n = n % 4
    if n>1:
        k = "tomato"
    else:
        k = "olivedrab"
    gw(a, k);
    if (n==1 or n==3):
        gw(a/2, "white")

def gwiazdy(kod):
    szer_rys = 560
    # przygotowuję listę złożoną z cyfr kodu
    cyfry = []
    while kod >0:
        cyfry.append(kod % 10);
        kod = kod // 10
    # odwracam kolejność cyfr na liście
    cyfry = cyfry[::-1]
    # wyliczam szerokość ozdoby
    szer_gw = szer_rys / (len(cyfry))
    # przemieszczam żółwia, by rysunek był wyśrodkowany
    pu(); bk(szer_rys/2 - szer_gw/2); lt(90)
    fd(2* szer_gw*sqrt(3)/3); rt(90); pd()
    #rysuję trzy rzędy ozdób
    for a in range(3):
        # oglądam kolejne cyfry kodu i rysuję odpowiednie gwiazdki
        for i in cyfry:
            gwgw(szer_gw, i)
            pu(); fd(szer_gw); pd()
        # usuwam ostatnią cyfrę liczby - kolejny rząd ma być krótszy
        cyfry.pop()
        # przemieszczam żółwia na początek kolejnego rzędu
        pu(); bk(len(cyfry)* szer_gw); rt(90)
        fd(2* szer_gw*sqrt(3)/3); lt(90); pd()
```

Testy

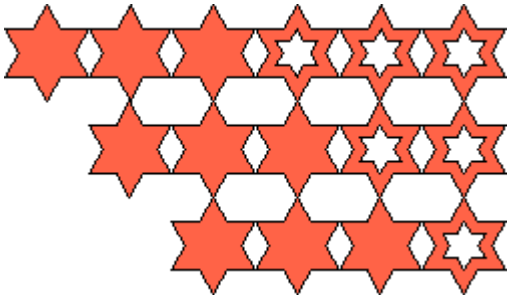
Testowanie rozwiązania rozpoczynamy od przykładów zawartych w treści zadania. Potem testujemy działanie programu dla różnych kodów liczbowych opisujących rysunek – różna liczba cyfr, wszystkie możliwe cyfry (ozdoby), różna kolejność cyfr, cyfry powtarzające się itp. Sprawdzamy, czy szerokość rysunku jest prawidłowa oraz czy jest on wyśrodkowany.

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

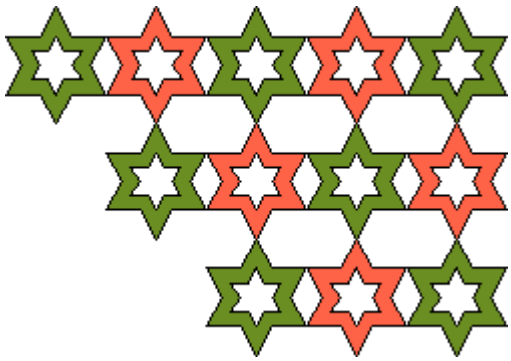
```
tracer(0)
gwiazdy(1234567890)
update()
```

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

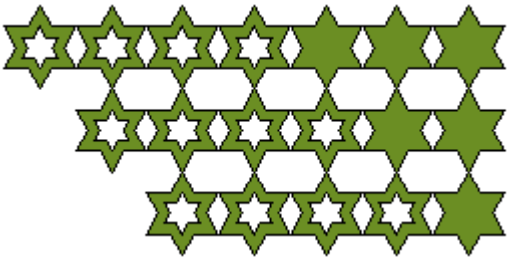
Przykładowe testy:



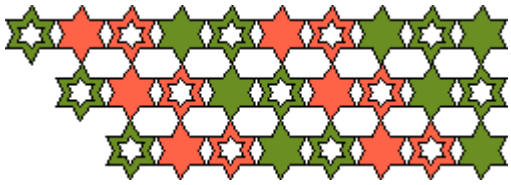
gwiazdy 262333



gwiazdy 97531



gwiazdy 1591408



gwiazdy 1234567890