LOGOMOCJA

GRAFIKA ŻÓŁWIA

CZĘŚĆ 1

OPRACOWAŁ:

Wojciech Rogowicz

Opis środowiska Logomocja

Program komputerowy to zrozumiały dla komputera ciąg instrukcji. Każdy program napisany jest w jakimś języku programowania. Język ten służy wyłącznie do zapisywania algorytmów wykonywania programów. Zestaw instrukcji opisujących całą aplikację nazywamy kodem źródłowym programu. Tali kod źródłowy nie jest bezpośrednio zrozumiały dla komputera. Aby maszyna go wykonała, musi zostać najpierw przetłumaczony na język wewnętrzny komputera, który jest ciągiem operacji elementarnych realizowanych przez komputer. Taki program nazywamy programem wynikowym. Aby zamienić kod źródłowy programu na sam program wynikowy musimy użyć innego programu zwanego translatorem (od ang. *translation* **- tłumaczenie). Operacja ta nazywana jest translacją. Wyróżniamy przy tym dwa rodzaje translatorów w zależności od zastosowanej metody tłumaczenia programu:**

- **Kompilator - wczytuje cały kod źródłowy programu, dokonuje jego optymalizacji i tłumaczy na kod wynikowy. W rezultacie otrzymujemy program wynikowy gotowy w każdej chwili do wykonania. Wynik kompilacji jest zapamiętywany w postaci pliku wykonywalnego (stąd rozszerzenie dla wielu programów zapisanych za pomocą pliku wykonywalnego *.exe od ang.** *executable*).
  Skompilowanie programu umożliwia jego wykonanie w dowolnym momencie na dowolnym komputerze. Kompilator jest nam potrzebny tylko raz (podczas kompilacji).
- **Interpretator - wczytuje fragment kodu źródłowego programu, tłumaczy na kod wynikowy i od razu wykonuje. Operacja ta jest powtarzana dla kolejnych elementów programu. Przy każdym uruchomieniu programu trzeba od początku zamieniać jego kod źródłowy na kod wynikowy zrozumiały dla komputera. Zatem do każdego wykonania programu musimy użyć interpretatora.**

Podczas pracy z językiem logo w środowisku Logomocja będziesz używał interpretatora. Dlatego aby uruchomić jakikolwiek program napisany w logo musisz posiadać i uruchomić program Logomocja.

Programy napisane w logo mogą wykonywać obliczenia, pisać teksty oraz tworzyć rysunki na ekranie. Rysunki na ekranie tworzone są przez żółwia (bądź wiele żółwi) na ekranie graficznym. Wszystkie polecenia rysowania będziesz kierował właśnie do żółwia! się po ekranie według twoich poleceń będzie on pozostawiał ślady, które z kolei utworzą obrazy, często dość skomplikowane. Plecenia dla żółwia wprowadzisz z klawiatury używając do tego celu ekranu tekstowego. Najwygodniej korzystać jest zatem z trybu graficzno-tekstowego, w którym jednocześnie możesz wpisywać polecenia i obserwować rezultaty ich wykonania.

Kreski, kwadraty i koła

Kreski

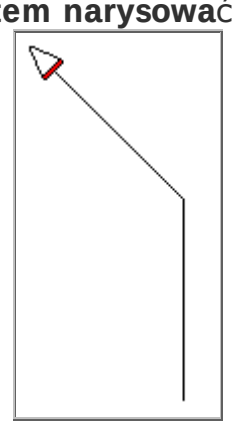
To, jaką czynność właśnie wykonuje żółw, zależy od trybu jego pracy. może rysować, ścierać lub po prostu się przemieszczać po ekranie. Aby żółw przeszedł do odpowiedniego trybu, korzystamy z następujących komend:

- **POD - żółw przesuwa się po ekranie nie zostawiając za sobą śladu;**
- **OPU - żółw przesuwa się po ekranie zostawiając za sobą linię;**
- **ŚĆIER - żółw przesuwa się po ekranie ścierając istniejące elementy.**

Spróbuj teraz przećwiczyć kilka najprostszych konstrukcji. Posłuż się do tego najbardziej elementarnymi procedurami:

- **PŻ - pokazuje symbol żółwia na ekranie;**
- **SŻ -chowa symbol żółwia na ekranie;**
- **CS - czyści ekran i ustawia żółwia na pozycji (0,0);**
- **NP X - porusza żółwiem o X pikseli do przodu;**
- **PW X - obraca żółwia o X° w prawo;**
- **LW X - obraca żółwia o X° w lewo.**

Nic trudnego, prawda? Jak zatem narysować poniższy rysunek?



Nic prostszego! Należy pokazać żółwia, następnie przejść żółwiem o 100 pikseli do przodu, obrócić go o 45° w lewo i na koniec przejść nim o 90 pikseli do przodu. Zatem Twój "program" będzie wyglądał następująco:

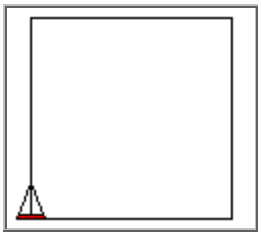
```
? pż
? np 100
? lw 45
? np 90
```

Kwadraty

Narysowanie kwadratu o określonym przez Ciebie boku jest stosunkowo łatwe. Musisz postawić cztery kreski obrócone względem siebie o 90°.

```
? cs
? np 100
? pw 90
? np 100
? pw 90
? np 100
? pw 90
? np 100
? pw 90
```

A oto i efekt:

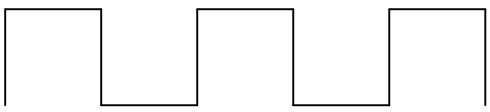


Zauważ, że w powyższym zadaniu czterokrotnie wykonałeś tą samą czynność: poruszałeś się naprzód, a następnie obracałeś żółwia o 90° w prawo. Program znacznie by się uprościł, gdyby można było nakazać żółwiowi pewne czynności wykonać kilkakrotnie. Spróbuj:

```
? cs
? powtórz 4 [ np 100 pw 90 ]
```

Efekt będzie taki sam, jak na poprzednim rysunku. Stosując plecenie powtórz wypisujemy ilość powtarzanych czynności natomiast w nawiasach zaznaczamy zakres kroków np.:

Rys. 1



Przykład rozwiązania zadania z *Rys. 1* :

```
?Powtórz 4[np 50 pw 90 np 50 pw 90 np 50 lw 90 np 50 lw 90]
lub
?Powtórz 4[Powtórz 2[np 50 pw 90] np 50 lw 90 np 50 lw 90]
```

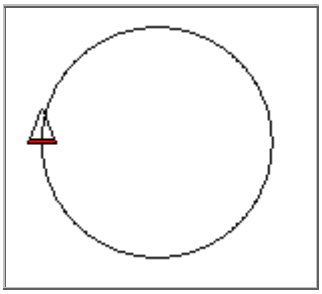
Koła

Mam nadzieję, że dotychczasowe czynności nie sprawiły Ci zbyt wiele kłopotu. Spróbuj więc teraz czegoś trudniejszego. Co sądzisz o narysowaniu koła?

W logomocji nie ma procedury pozwalającej na bezpośredni rysowanie okręgów czy łuków. Aby udało Ci się narysować okrąg musisz żółwia trochę oszukać. Spróbuj narysować trzystuseśćdziesięciobok o boku 1. Będzie on wyglądał niemal idealnie jak koło.

```
? cs
? powtórz 360 [ np 1 pw 1 ]
```

W wyniku otrzymasz taki obrazek:



Tak naprawdę do koła wystarczająco podobny jest 36-bok i nim będziemy się w dalszej części kursu posługiwać.

```
? cs
? powtórz 36 [ np 10 pw 10 ]
```

Procedury bez parametrów

Programowanie w logo polega na opisywaniu czynności wykonywanych przez żółwia za pomocą procedur, czyli zrozumiałych dla żółwia. Najprostsze czynności zostały zdefiniowane przez twórców języka logo.

Nazywamy je procedurami pierwotnymi. Niektóre z nich już poznałeś (na przykład NP, WS, CS PW czy LW). Dość obszerną listę procedur pierwotnych znajdziesz w dziale różne.

Wykorzystując istniejące procedury możesz budować własne, odpowiadające chwilowym potrzebom programów.

Definicję procedury rozpoczynasz od słowa oto. Następnie podaj nazwę procedury (nie może ona zawierać spacji), potem dodaj treść procedury. Koniec sygnalizujesz słowem już.

Aby zdefiniować procedurę można wykonać to w pasku pleceń ale także korzystając ze schowka. Druga metoda jest bardziej praktyczna.

```
? oto nazwa_procedury
? treść procedury
? już
```

Jak to działa? Spróbuj zdefiniować sobie procedurę rysującą kwadrat o boku 100 punktów.

```
? oto kwadrat
? powtórz 4 [ np 100 pw 90 ]
? już
```

Teraz wystarczy wywołać procedurę...

```
? kwadrat
```

...i na ekranie pojawi się żądany kwadrat.

Zadeklarowany w ten sposób każdy rysowany kwadrat będzie miał bok długości 100 punktów. Jeżeli będziesz potrzebował do rysowania innego kwadratu, np. o boku 20 punktów, będziesz musiał zdefiniować nową procedurę pod nową nazwą:

```
? oto kwadrat20
? powtórz 4 [ np 20 pw 90 ]
? już
```

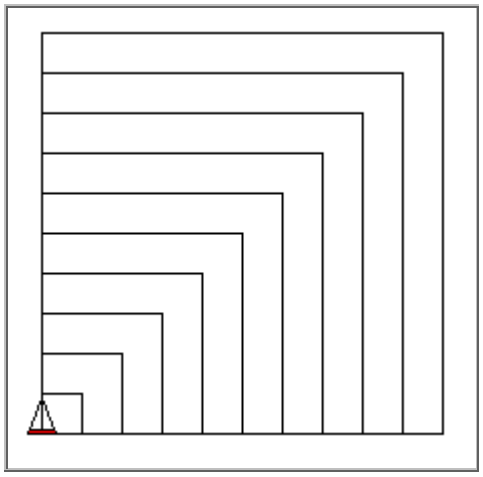
W ten sposób możesz zdefiniować także inne figury, np. koło czy trójkąt:

```
? oto koło
```

```
? powtórz 36 [ np 10 pw 10 ]
? już
?
? oto trójkąt
? powtórz 3 [ np 100 pw 120 ]
? już
```

Procedury z parametrami

Zauważ, że definiowanie procedury bez parametru jest mało elastyczne. Właściwie do każdej potrzebnej nam figury będziesz potrzebował nowej procedury. Żeby narysować poniższy rysunek, musiałbyś zdefiniować 10 różnych procedur kwadratów o bokach od 20 do 200 punktów!



Znacznie lepiej byłoby zdefiniować jedną uniwersalną procedurę, która sprawdzałaby się każdej sytuacji. Struktura takiej procedury wygląda następująco:

```
? oto nazwa_procedury :parametr1 :parametr2 itd.
? treść procedury
? już
```

Spróbuj teraz zdefiniować procedurę, w której parametrem będzie długość boku kwadratu:

```
? oto kwadrat :bok
? powtórz 4 [np :bok pw 90]
? już
```

Teraz wystarczy wywołanie: `kwadrat 70` i już kwadrat o boku 70 pojawi się na Twoim ekranie.

Parametrem Twojej procedury jest :bok. Nazwa poprzedzona jest : (dwukropkiem), który nazywamy operatorem wartościowania. Będziesz go stosował, aby odróżnić nazwę parametru (bądź zmiennej) od nazwy procedury.

Spójrz i wypróbuj także poniższe procedury. Pierwsza narysuje na ekranie koło o zadanym promieniu, druga wielobok o określonym przy wywoływaniu boku i liczbie kątów.

Dlaczego takie skomplikowane obliczenia przy kole? Otóż obwód 36-boku równy jest 36 * długość boku. Natomiast obwód koła równy jest 2 * Pi * r. Przyrównajmy te dwie warości do siebie:

2 * Pi * r = 36 * bok

Znając promień otrzymujemy:

bok = (2 * Pi * r) / 36 ˜= 0,175 * r

```
? oto koło :promień
? powtórz 36 [np :promień * 0,175 pw 10]
? już

? oto wielobok :bok :ileboków
? powtórz :ileboków [np :bok pw 360 / :ileboków]
? już
```

Spróbuj przygotować własne procedury rysujące kwadrat, romb, prostokąt i równoległobok o zadanych przez Ciebie parametrach (długości boków i kąty między bokami).

Powodzenia!

Tworząc bardziej skomplikowane rysunki możemy łączyć ze sobą procedury. Tak więc aby wykonać poniższy rysunek należy:

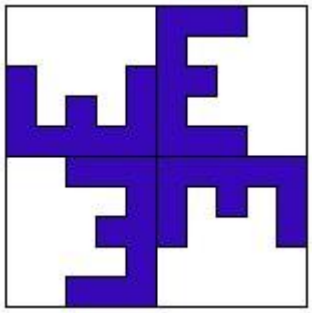
***w tym zadaniu wykorzystywana jest procedura niech. Oto przykład jej zastosowania.**

Zawsze piszę procedury z parametrem, ponieważ często je wykorzystuję do tworzenia innych, bardziej skomplikowanych. Jednak czasem z treści zadania wynika, że procedura ma być wywoływana bez parametru. Jak mogę szybko przerobić procedurę z parametrem na procedurę bez parametru?

Można to zrobić na kilka sposobów.
Jednym z nich jest użycie polecenia **niech**. Oto przykład

procedura z parametrem	procedura bez parametru
oto kwadrat :bok powtórz 4 [np :bok pw 90] już	oto kwadrat niech "bok 100 powtórz 4 [np :bok pw 90] już

Polecenie **niech** tworzy w procedurze **kwadrat** zmienną o nazwie bok (nazwę poprzedzamy znakiem dosłowności ") i nadaje mu wartość 100. Jeśli chcemy zmienić długość boku wystarczy przypisać zmiennej nową wartość. No tak, ale czy nie prościej jest w procedurze **kwadrat** zastąpić zmienną liczbą 100 lub inną?
W tym przypadku zgoda, ale jeśli procedura jest dłuższa to narażamy się na konieczność dokonywania wielu zmian. Łatwo więc o pomyłkę

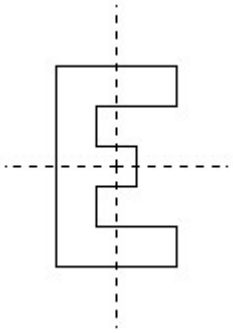


Zdefiniować procedurę rysująca literkę E. No to po kolei.

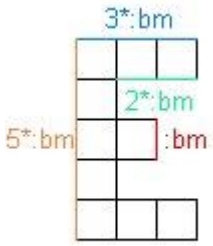
Ułóż procedurę o nazwie E, która utworzy na środku ekranu, stosunkowo duży, następujący rysunek:

wskazówki:

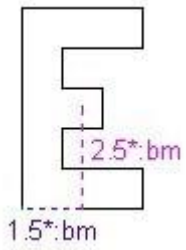
1. środek rysunku wyznaczamy dzieląc literę na pół w pionie i w poziomie:



2. przyjmujemy za parametr najmniejszy z boków litery (:bm) **i wyznaczamy długości pozostałych boków:**



3. określamy długość skoku żółwia ze środka ekranu do punktu, z którego zaczynamy rysować literę:



4. ustalamy kolor malowania i rysujemy zamalowaną literkę używając polecenia `wielokąt`;

5**. na koniec wracamy żółwiem na środek ekranu**

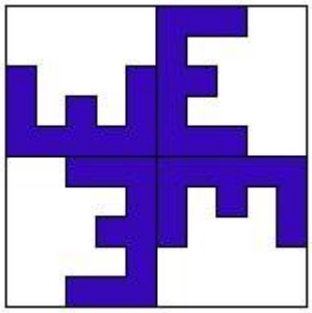
Procedura E

```
oto E
 niech "bm 70
 skok2 (- 2.5 * :bm) (- 1.5 * :bm)
 ukm "ciemnoniebieski5
 wielokąt [np (5 * :bm) pw 90
           np (3 * :bm) pw 90
           np :bm pw 90
           np (2 * :bm) lw 90
           np :bm lw 90
           np :bm pw 90
           np :bm pw 90
           np :bm lw 90
           np :bm lw 90
           np (2 * :bm) pw 90
           np :bm pw 90
           np (3 * :bm) pw 90]
 skok2 2.5 * :bm 1.5 * :bm
już

oto skok2 :a :b
 pod
  np :a pw 90
  np :b lw 90
 opu
już
```

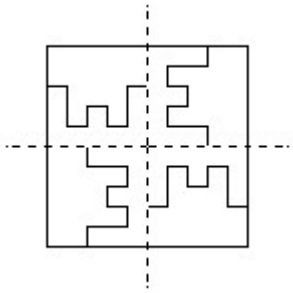
Stwórzmy teraz naszą szukaną figurę.

Ułóż procedurę o nazwie E2, która utworzy na środku ekranu, stosunkowo duży, następujący rysunek:



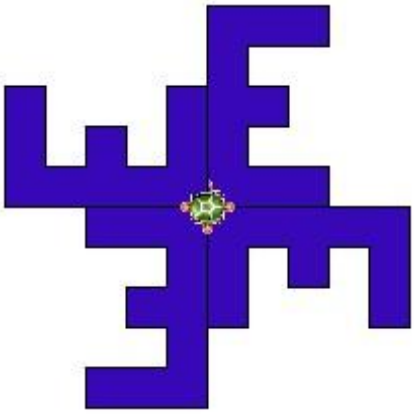
wskazówki:

1.wyznaczamy środek rysunku (żółw będzie startował z tego punktu) - dzielimy kwadrat w pionie i w poziomie na pół:

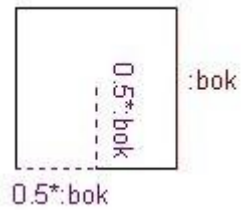


2. przerabiamy procedurę E - zmieniając jej nazwę na np. E1 - tak, aby dolny, lewy róg litery znajdował się na środku ekranu:

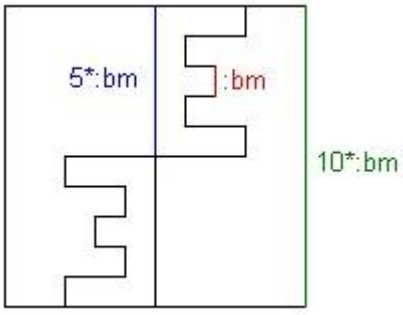
3. **powtarzamy cztery razy: wywołanie procedury E1 i obrót w prawo o 90°:**



4**. układamy procedurę z parametrem, która tworzy na środku ekranu kwadrat o dowolnym boku:**



5. wyznaczamy długość boku kwadratu, pamiętając o założeniach poczynionych przy pisaniu procedury E1:



```
oto E2
 niech "bm 40
 powtórz 4 [e1 :bm pw 90]
 kwadrat 10 * :bm
już

oto e1 :bm
 ukm "ciemnoniebieski5
 wielokąt [np (5 * :bm) pw 90
           np (3 * :bm) pw 90
           np :bm pw 90
           np (2 * :bm) lw 90
           np :bm lw 90
           np :bm pw 90
           np :bm pw 90
           np :bm lw 90
           np :bm lw 90
           np (2 * :bm) pw 90
           np :bm pw 90
           np (3 * :bm) pw 90]
już

oto kwadrat :bok
 skok2 ( - :bok / 2 ) ( - :bok / 2 )
  powtórz 4 [np :bok pw 90]
 skok2 :bok / 2 :bok / 2
już

oto skok2 :a :b
 pod
  np :a pw 90
  np :b lw 90
 opu
już
```

Co to jest rekurencja?

Rekurencja jest to wywołanie podprogramu (procedury) samej przez siebie. W logo zapis rekurencji będzie wyglądał następująco:

```
? oto nazwa_funkcji
? czynności_wykonywane_przez_procedurę
? nazwa_funkcji
? już
```

Zauważ, że tak skonstruowana procedura działałaby w nieskończoność. Dlatego przygotowując funkcję rekurencyjną zawsze musisz zadbać o istnienie parametru warunkującego kolejne wykonanie procedury. Skorzystaj w tym celu z procedury warunkowej jeśli. Ma ona następującą strukturę:

jeśli warunek [co_zrobić_gdy_prawda] [co_zrobić_gdy_fałsz]

A oto dwa przykłady:

```
? oto nazwa_funkcji :parametr
? jeśli :parametr = 0 [stop]
? czynności_wykonywane_przez_procedurę
? nazwa_funkcji :parametr - 1
? już
```

```
? oto nazwa_funkcji :parametr
? czynności_wykonywane_przez_procedurę
? jeśli :parametr > 10 [nazwa_funkcji :parametr / 2]
? już
```

Jak działają powyższe funkcje?

W pierwszym wypadku procedura zostanie wykonana w zależności od wielkości parametru [dokładnie tyle razy ile wynosi wartość :parametr].

W drugim wypadku procedura będzie wykonywana nieokreśloną ilość razy do momentu, gdy wartość parametru spadnie poniżej założonej wielkości (w naszym wypadku 10).

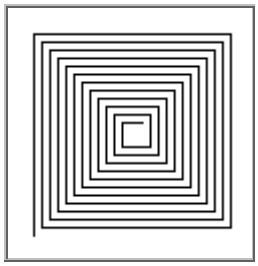
Proste rekurencje

Spróbujmy zatem coś narysować.

Spirala

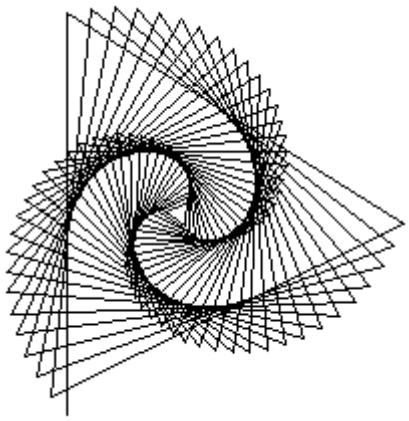
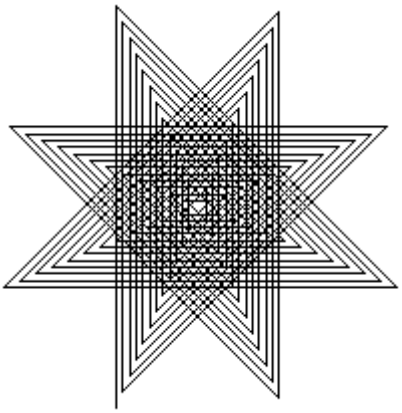
Najprostszym przykładem rekurencji będzie dla Ciebie narysowanie spirali. Jak powstaje spirala? Spróbuj narysować ją poprzez poruszanie się żółwiem do przodu za każdym razem zmniejszając krok o 2 punkty aż do wielkości kroku 5. Za każdym razem obracaj żółwia o 90 stopni w prawo. Procedurę i wynik możesz obejrzeć poniżej.

```
? oto spirala :bok
? jeśli :bok < 10 [stop]
? np :bok pw 90
? spirala :bok - 2
? już
```



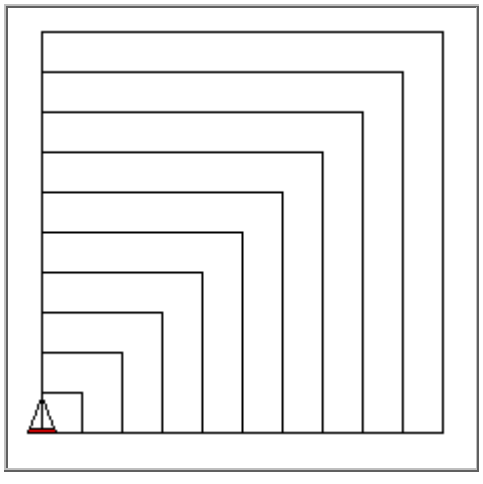
Spróbuj zdefiniować podobną procedurę. Dodaj jednak parametr określający kąt obrotu żółwia zamiast jednostajnego 90 stopni. Procedurę oraz wyniki jej wywołania możesz zobaczyć poniżej.

```
? oto spirala :bok :kąt
? jeśli :bok < 10 [stop]
? np :bok pw 90
? spirala :bok - 2 :kąt
? już
```

spirala 200 122	spirala 200 135
	

Kwadraty

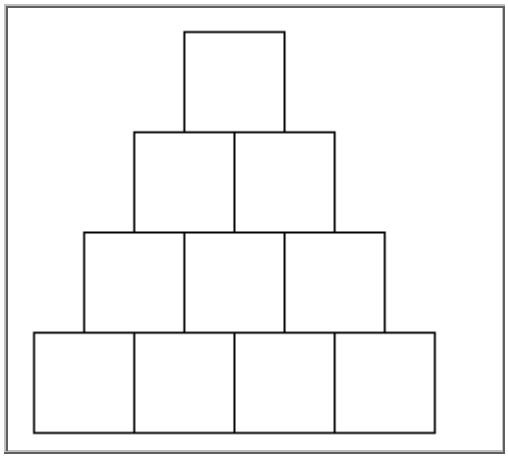
Zobacz, jak łatwo za pomocą rekurencji narysować kilka kolenych kwadratów.



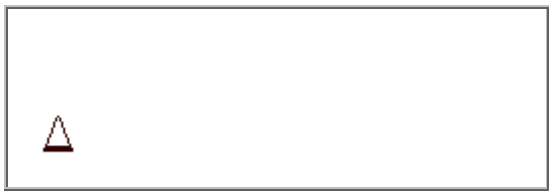
Aby narysować taki obrazek za pomocą zwykłej procedury z parametrem, musiałbyś ją wywoływać dziesięć razy! Zastąpisz ją jedną procedurą rekurencyjną. Po prostu za każdym razem będziesz zmniejszał wielkość boku o 20 aż uzyskasz wartość mniejszą niż 10.

```
? oto kwadraty :bok
? jeśli :bok < 10 [stop]
? powtórz 4 [np :bok pw 90]
? kwadraty :bok - 20
? już
```

Spróbuj teraz narysować piramidę z kwadratów. Napisz procedurę piramida :n, która wywołana z parametrem narysuje piramidę o *n* rzędach, przy czym w pierwszym rzędzie będzie *n* kwadratów, natomiast w każdym kolejnym rzędzie będzie stopniowo o jeden kwadrat mniej. Przykładowo, w wyniku wywołania piramida 4 otrzymasz figurę przedstawioną na poniższym rysunku:



Jak to działa? Najpierw musisz narysować kwadrat i ustawić się do rysowania kolejnego kwadratu. Czynność tę 4 powtarzasz razy.



W tym celu musisz wykonać następujące polecenia:

```
powtórz :n [powtórz 4 [np 50 pw 90] pw 90 np 50 lw 90]
```

Następnie wracasz do punktu początkowego:

```
pw 90 ws :n * 50 lw 90
```

Teraz ustawiasz się na miejscu rysowania kolejnej warstwy piramidy:

```
np 50 pw 90 np 25 lw 90
```

Nie pozostało Ci nic innego, niż wywołać całą procedurę rekurencyjnie:

```
piramida :n - 1
```

A oto i cała treść procedury. Nie zapominaj o warunku przerwania pętli!

```
? oto piramida :n
? jeśli :n = 0 [stop]
? powtórz :n [powtórz 4 [np 50 pw 90] pw 90 np 50 lw 90]
? pw 90 ws :n * 50 lw 90 np 50 pw 90 np 25 lw 90
? piramida :n - 1
? już
```

Dla AmbitnychJ

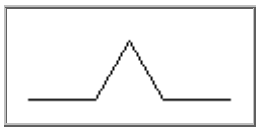
Fraktale

Fraktale to, najprościej ujmując, wielokrotnie powtarzane motywy w postaci krzywych. Charakteryzują się tym, że ich części składowe są pomniejszonym obrazem całości.

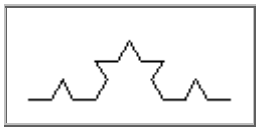
Płatek Kocha

Najprostszym fraktalem jest płatek Kocha. Jako stopień zerowy tego fraktala przyjmujemy odcinek o długości :n (wywołanie koch 0 100).

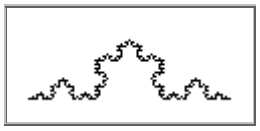
Stopień pierwszy powstaje przez podział odcinka na trzy równe części, a następnie zastąpienie środkowej części dwoma złączonymi odcinkami równymi części usuniętej (wywołanie koch 1 100).



Aby utworzyć stopień drugi, musisz podzielić każdy z odcinków na 4 nowe (wywołanie koch 2 100).



Tak będzie się prezentować się wywołanie koch 4 100.



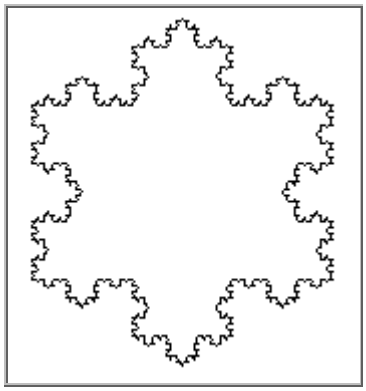
A oto procedura, która pozwoli Ci narysować pojedynczy płatek:

```
oto koch :stopień :długość
 jeśli :stopień = 0 [np :długość stop]
 koch :stopień - 1 :długość / 3 pw 60
 koch :stopień - 1 :długość / 3 lw 120
 koch :stopień - 1 :długość / 3 pw 60
 koch :stopień - 1 :długość / 3
już
```

Pełny płatek Kocha będziesz mógł zobaczyć trzykrotnie powtarzając rysowanie pojedynczego "listka".

```
oto gwiazdka :stopień :długość
 powtórz 3 [koch :stopień :długość lw 120]
już
```

Wynik wywołania gwiazdka 4 150.

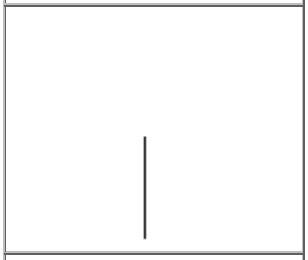


Drzewo binarne

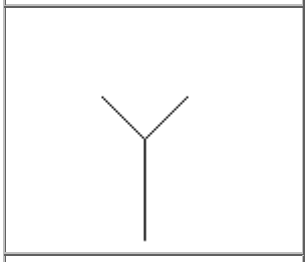
To także przykład bardzo prostego fraktala. Jak go otrzymasz? Musisz narysować kreskę, na jej końcu dwie kolejne kreski położone względem siebie pod kątem 45°, na ich końcach kolejne dwie kreski...

```
oto dbin :wiek :pien
 jeśli :wiek = 0 [np :pien pw 180 np :pien stop]
 np :pien
 lw 45
 dbin :wiek - 1 :pien * 0,6
 lw 90
 dbin :wiek - 1 :pien * 0,6
 lw 45
 np :pien
już
```

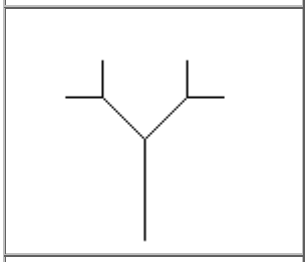
dbin 0 50



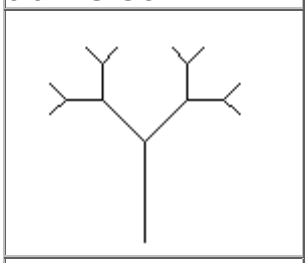
dbin 1 50



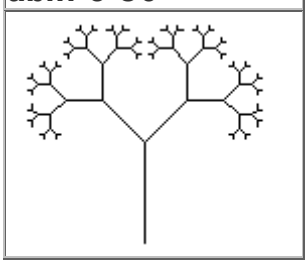
dbin 2 50



dbin 3 50



dbin 6 50

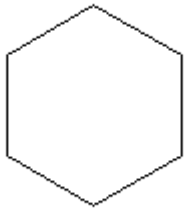


Plaster miodu

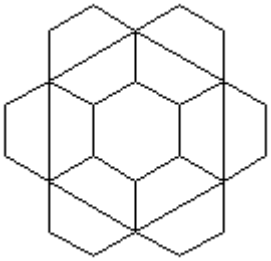
Ten fraktal otrzymasz rysując sześcian, a następnie rysując sześć mniejszych sześcianów na początku każdego boku, potem rysujesz sześcian na boku każdego z małych sześcianów...

```
oto plaster :poziom :bok
 jeśli :poziom = 0 [stop]
 powtórz 6 [np :bok plaster :poziom - 1 :bok / 2 pw 60]
już
```

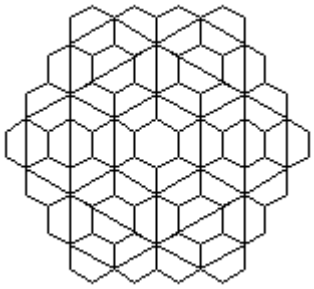
plaster 1 50



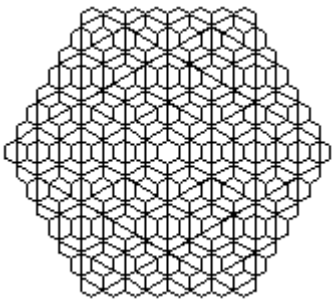
plaster 2 50



plaster 3 50



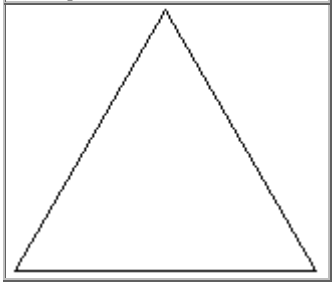
plaster 4 50



Trójkąt Sierpińskiego

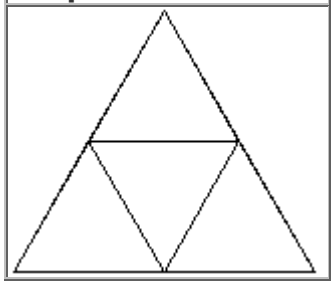
Ten fraktal otrzymasz rysując trójkąt.

sierpinski 0 150



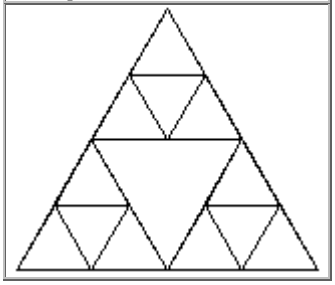
Kolejna czynność to narysowanie wewnątrz następnego trójkąta o boku równym połowie pierwszego.

sierpinski 1 150



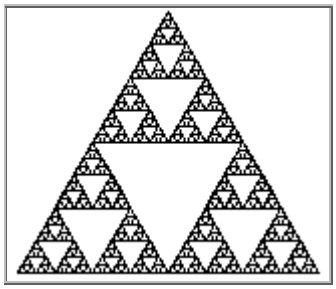
Następnie wrysowujesz wewnątrz trzech trójkątów podobnych kolejny trójkąt.

sierpinski 2 150



Można tak jeszcze długo..

sierpinski 5 150



```
oto troj :a
 powtórz 3 [np :a pw 120]
już

oto sierp :n :a
 jeśli :n = 0 [troj :a stop]
 troj :a
 sierp :n - 1 :a / 2
 np :a / 2
 sierp :n - 1 :a / 2
 pw 60
 np :a / 2 pw 60
 sierp :n - 1 :a / 2
 lw 60 ws :a / 2
 lw 60 ws :a / 2
już

oto sierpinski :n :a
 cs
 pw 30
 sierp :n :a
już
```

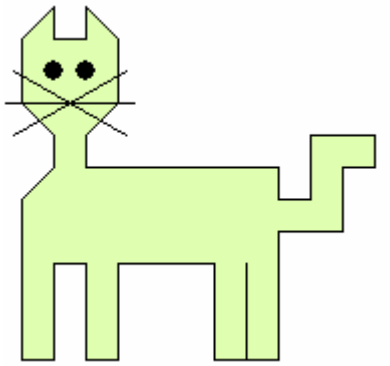
Zadania 1 etapu konkursu miniLOGIA 5

– przedmiotowego konkursu informatycznego dla uczniów szkół podstawowych województwa mazowieckiego

9 listopada – 6 grudnia 2006 roku

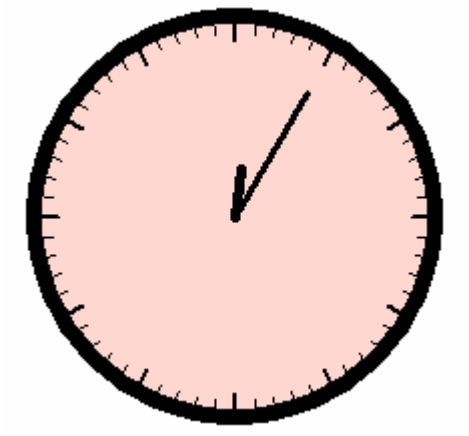
Zadanie 1.

Napisz procedurę **KOT**, po wywołaniu której powstanie rysunek taki jak poniżej:



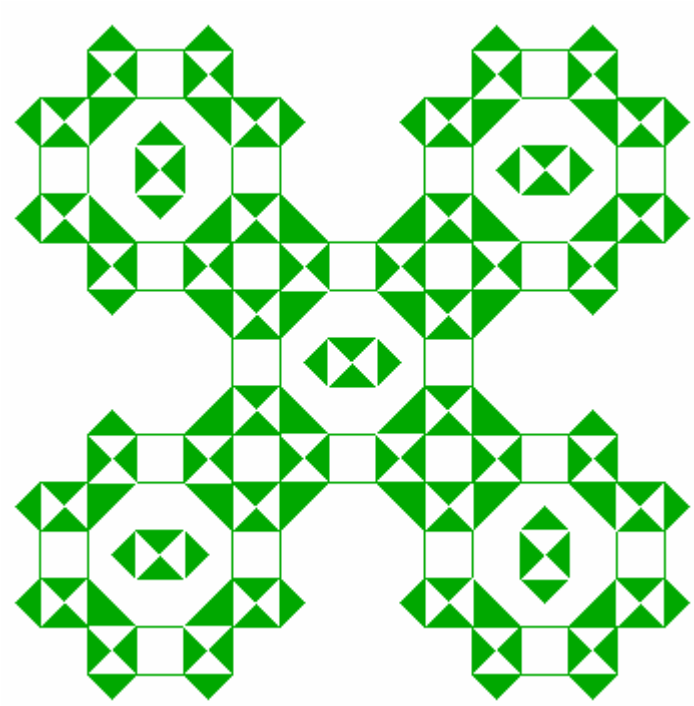
Zadanie 2

Napisz procedurę **ZEGAR**, po wywołaniu której powstanie rysunek zegara jak na rysunku poniżej. Zegar wskazuje godzinę 12^{05}. Rysunek powinien być możliwie duży i na środku ekranu.



Zadanie 3

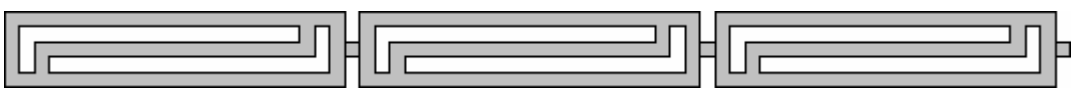
Napisz procedurę **MOZAIKA**, po wywołaniu której powstanie rysunek taki jak poniżej. Rysunek powinien być możliwie duży i na środku ekranu.



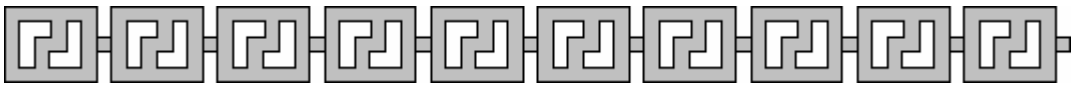
Zadanie 4

Napisz procedurę **BRANSOLETKA :ile**, po wywołaniu której powstanie rysunek taki jak poniżej. Bransoletka ma zawsze taką samą szerokość, równą 700 i taką samą wysokość, równą 50 (niezależnie od wartości parametru :ile). Parametr :ile określa liczbę powtarzających się elementów (może przyjmować wartości od 1 do 10).

BRANSOLETKA 1



BRANSOLETKA 3



BRANSOLETKA 10