Temat: Technologia informacyjna a informatyka

1. **Informatyka** - dyscyplina naukowa i techniczna zajmująca się przetwarzaniem informacji.

   Technologia informacyjna *(ang.) Information Technology,* IT – jedna z dziedzin informatyki (włącznie ze sprzętem komputerowym oraz oprogramowaniem) łącząca telekomunikację, narzędzia i inne technologie związane z informacją. Dostarcza narzędzi do pozyskiwania informacji, selekcjonowania ich, analizowania, przetwarzania, zarządzania i przekazywania innym.

2. **System operacyjny** – program umożliwiający użytkownikowi komunikowanie się z komputerem. Kontroluje pracę programów na nim używanych. Kontroluje pracę całego zestawu komputerowego.

3. Typy pamięci:

   a. **Zewnętrzna** (dyskietki, płyty, pendrive, dyski zewnętrzne, karty pamięci, ...)
   b. **Wewnętrzna** (ROM, RAM)

      ROM – (trwała) tylko do odczytu, umożliwia rozpoczęcie pracy komputera; przechowuje **BIOS** (testuje sprzęt, uruchamia system operacyjny i kontroluje komunikację między urządzeniami komputera)

      RAM – „operacyjna" (nietrwała), przechowuje dane powstające podczas pracy na komputerze, giną po wyłączeniu komputera.

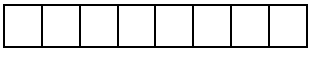
4. **System dziesiątkowy:**
   Ilość cyfr – 10 – {0,1,2, ...,9}
   Podstawa – „10"
   Np. $245_{(10)}$=...

5. **System dwójkowy (binarny)**
   Ilość cyfr – 2 – {0,1}
   Podstawa – „2"
   Np.: $0_{(2)}$, $1_{(2)}$, $101_{(2)}$, $111_{(2)}$,

   Zamiana – ćw.
   a) 2->10 $11101_{(2)}$=
   b) 10 -> 2 $15_{(10)}$ =

6. Jednostki pamięci komputerowej:
   bit – najmniejsza jednostka informacji (0 lub 1)

   Bajt
   1B=8bit

   1kB=1024B (kilobajt)
   1MB=1024kB (megabajt)

1GB=1024MB (gigabajt)
1TB=1024GB (terabajt)
...

Temat: Algorytmy – przykłady.

1. **Algorytm** – to przepis, uporządkowany sposób postępowania prowadzący do rozwiązywania zadania. Każdy program komputerowy realizuje jakiś algorytm zapisany w zrozumiałym dla komputera języku programowania. Istnieją zadania „niealgorytmiczne", dla których nie można opracować algorytmu rozwiązania.

2. Algorytm musi być:
   - **poprawny**
   - **jednoznaczny**
   - **szczegółowy**
   - **uniwersalny**

   inne cechy algorytmu to:
   - **skończoność**
   - **efektywność (sprawność)**

3. Specyfikacja algorytmu - obejmuje podanie:
   - **danych wejściowych** (czyli nazwy używanych zmiennych i ich typ – np. liczba całkowita, rzeczywista, wartość logiczna) np. a, b – liczby naturalne
   - **wyniku**, który algorytm powinien otrzymać, np. suma liczb naturalnych (a+ b)
   - **zmiennych pomocniczych** niezbędnych do realizacji programu. Np. s – suma

Uniwersalny algorytm operuje nie na liczbach, a na **zmiennych** (czyli „pojemnikach na dane" oznaczonych dowolną literą lub łańcuchem znaków). Np.: *a, b, suma, il, iloczyn*, itp.

4. Rodzaje algorytmów:
   - **liniowy** (nie ma w nim żadnych warunków, kolejne czynności są wykonywane jedna po drugiej)
   - **warunkowy** (wykonanie instrukcji uzależnione jest od spełnienia lub niespełnienia warunku; jeśli warunek jest spełniony – to …, a jeśli nie – to …)
   - **iteracyjny** (czyli z pętlą, polegającą na wielokrotnym powtarzaniu instrukcji)

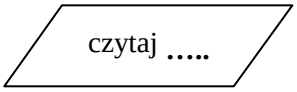
Temat: Sposoby przedstawiania algorytmów.

1. Algorytm może być zadany:
   - opisem słownym
   - wypunktowaną listą kroków
   - schematem blokowym (czyli zapisem graficznym, przy użyciu odpowiednich bloków)
   - pseudokodem, pseudojęzykiem („kroki" od danych wejściowych do wyników, np. czytaj(a), czytaj(b) s:=a+b, pisz(s))
   - określonym językiem programowania (czyli językiem zrozumiałym dla komputera, służącym do zapisywania programów i komunikowania się człowieka z komputerem)

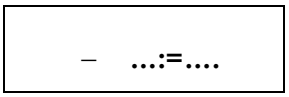
2. **Znaczenie klocków.** Blok:

- **startowy** (rozpoczęcie wykonywania algorytmu)

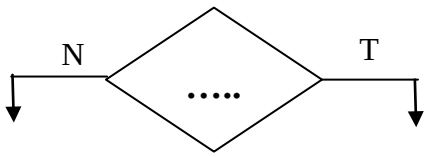
- **wejścia** (wczytywanie danych, przypisywanie ich zmiennym)



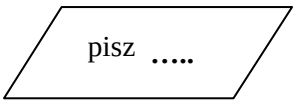
- **operacyjny** (wykonywanie operacji, konkretnych działań, obliczeń, może tu być więcej niż jedno wyrażenie)



- **warunkowy** (tzw. decyzyjny, z dwoma wyjściami: „tak" – jeśli warunek jest spełniony, „nie" – jeśli warunek jest niespełniony)



- **wyjściowy** (wypisanie wyniku, efektu wykonywanych działań)



- **końcowy** („stop", koniec działania algorytmu)

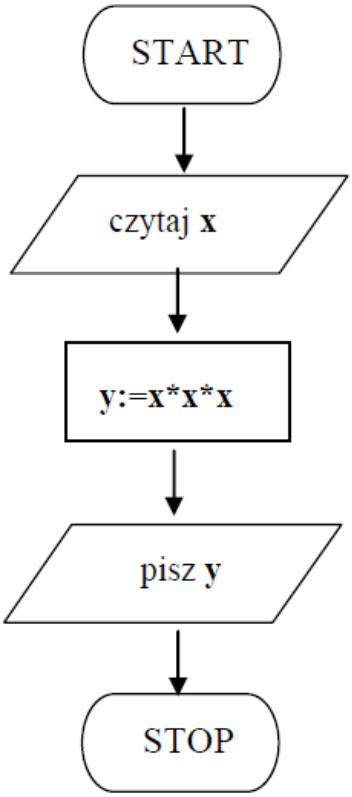
(ćwiczenia ksero – przykłady: 1-8)

1. **Sześcian liczby** (x^3) – algorytm liniowy.

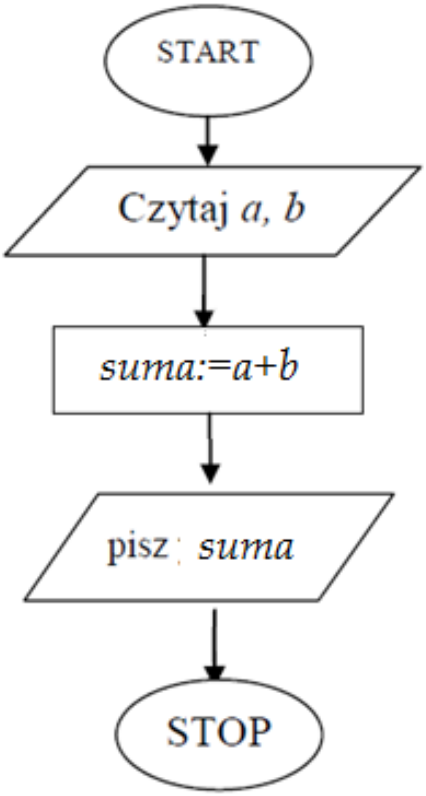
Specyfikacja:
Dane:
x- liczba całkowita podana przez użytkownika

Wynik:
y- sześcian liczby x

Pseudokod:
Program szescian
Zmienne:
x, y: całkowite
Początek
czytaj(x)
y:=x*x*x
pisz(y)
koniec



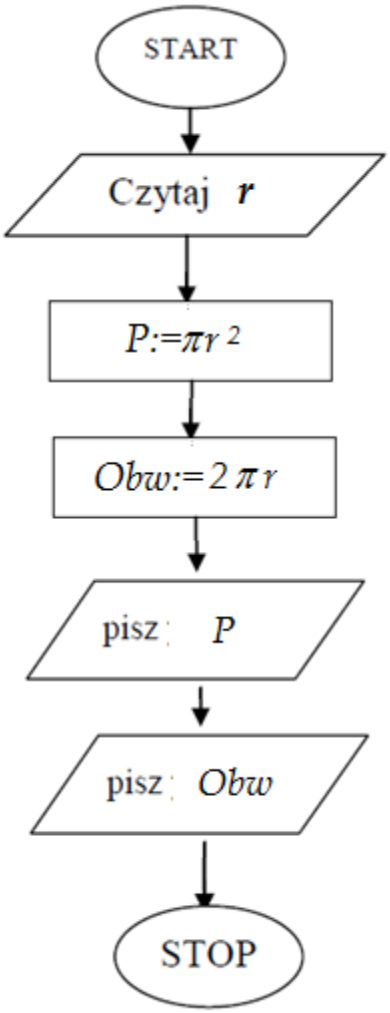
2. Algorytm **dodawania** dwóch liczb.



Specyfikacja:
Dane:
a, b – liczby całkowite podane przez użytkownika

Wynik:
suma – suma liczby *a* i *b*

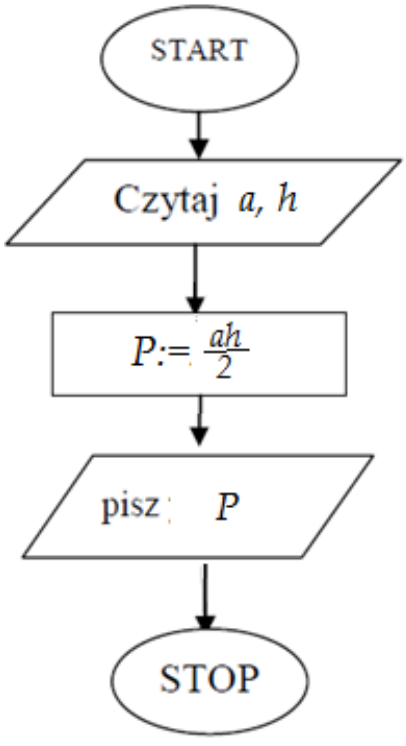
3. Algorytm obliczania **pola i obwodu koła**.



Specyfikacja:
Dane:
r – liczba naturalna – promień koła podany przez użytkownika

Wynik:
P – pole koła o promieniu *r*
Obw - obwód koła o promieniu *r*

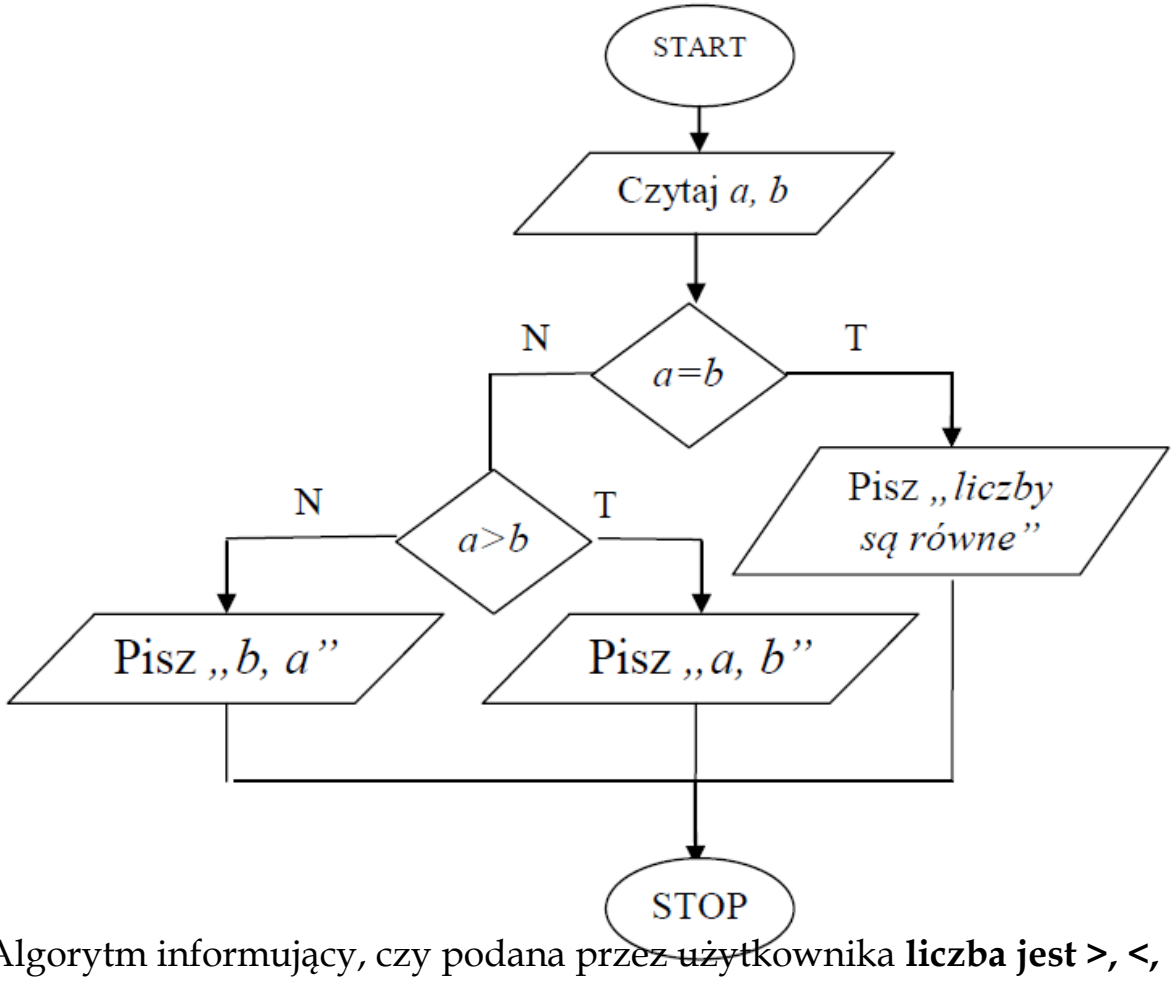
4. Algorytm obliczający **pole trójkąta** (dowolnego).



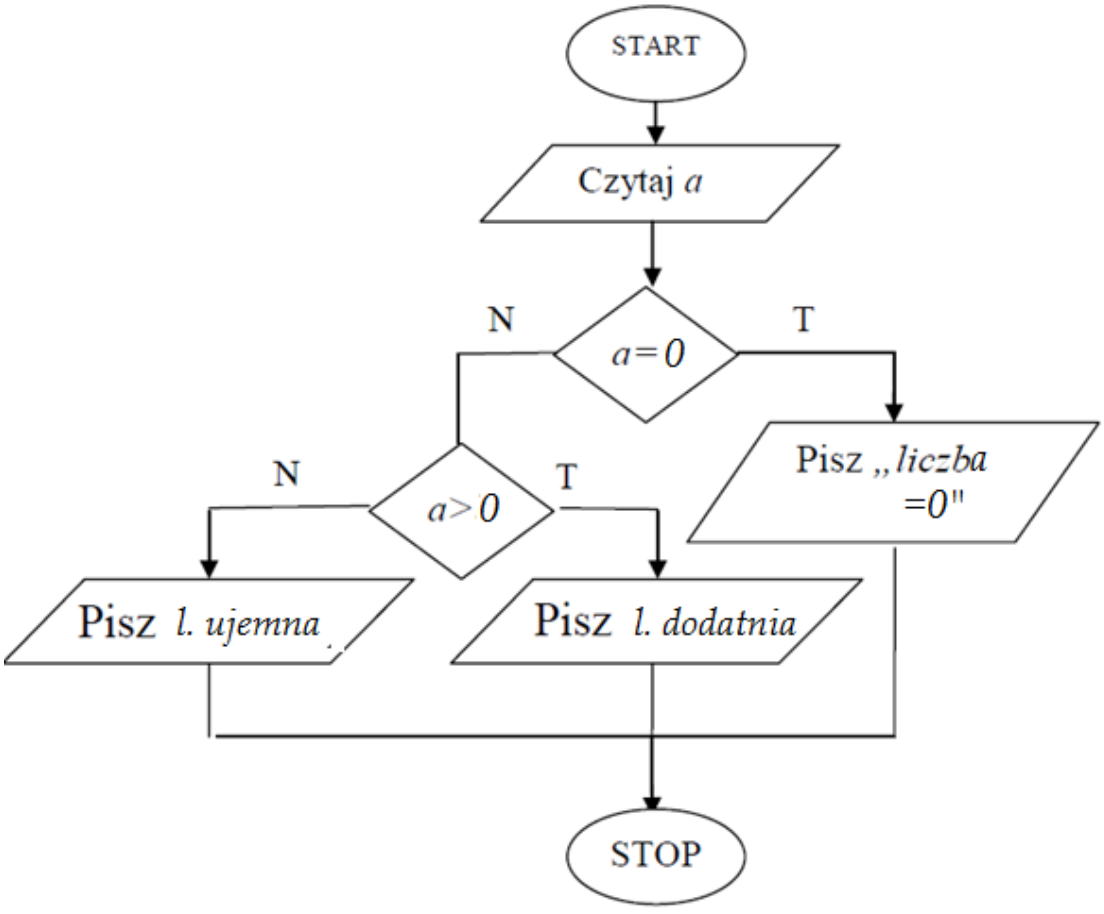
Specyfikacja:
Dane:
a – liczba naturalna – podstawa trójkąta
h – liczba naturalna – wysokość trójkąta (opuszczona na podstawę *a*!!!)

Wynik:
P – pole trójkąta o podstawie *a* i wysokości *h*

5. Wczytuje dwie **liczby** i wypisuje je **w kolejności malejącej** (tzn. większa, mniejsza). Jeśli są równe – wyświetla komunikat „liczby są równe”



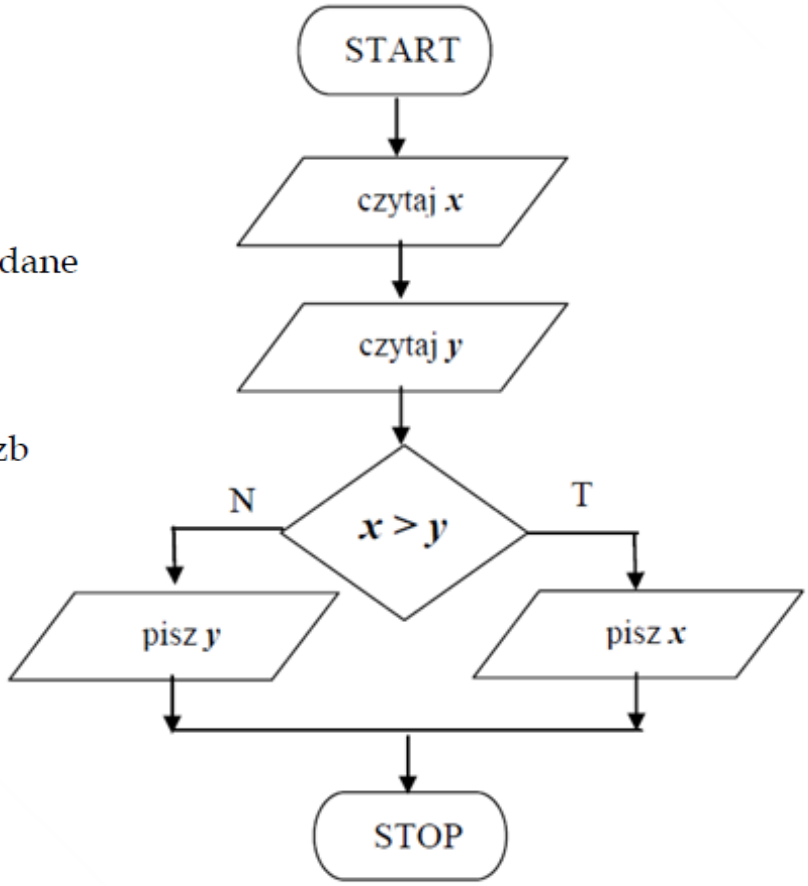
6. Algorytm informujący, czy podana przez użytkownika **liczba jest >, <, czy =0.**



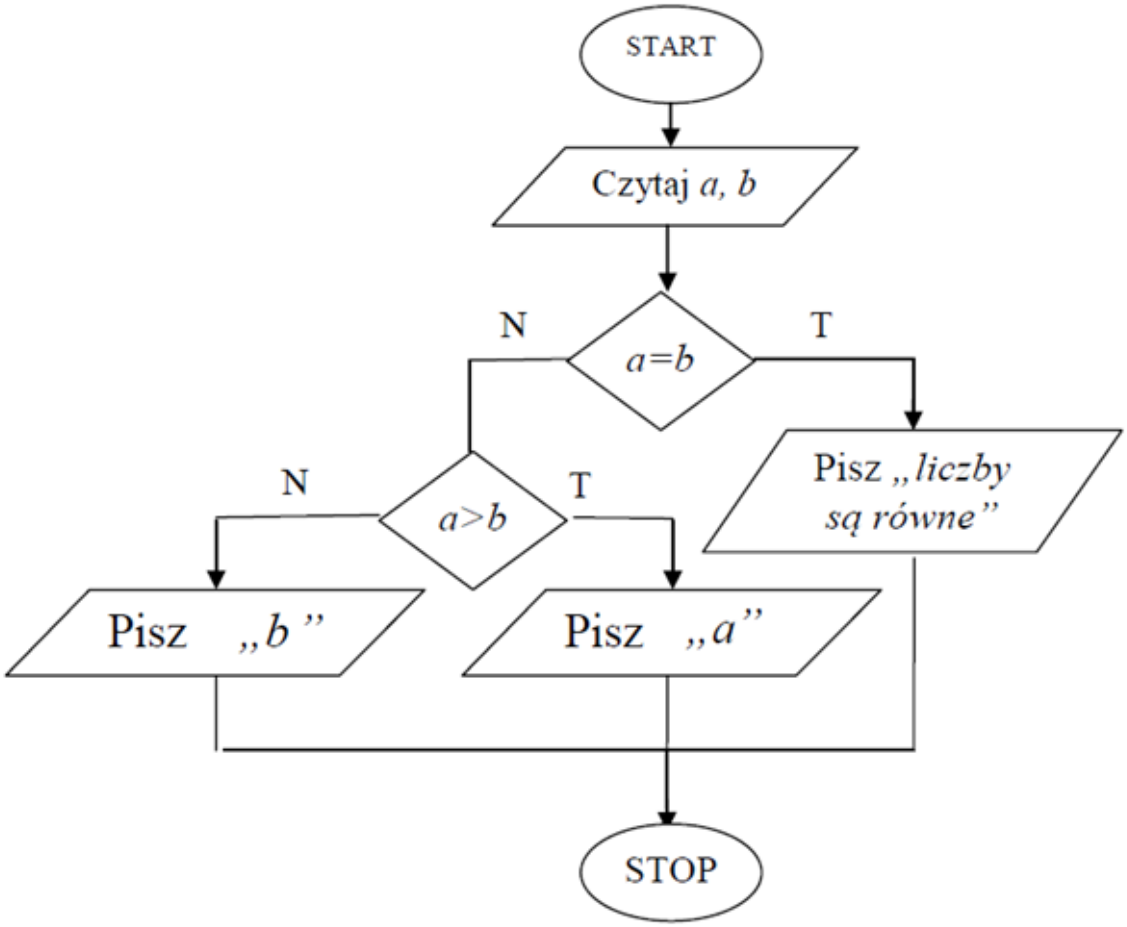
7. **Wybór większej liczby** - algorytm warunkowy (zakładamy, że podawane liczby *a* i *b* będą różne!
   $x \neq y$).

   Specyfikacja:
   Dane:
   x, y- liczby całkowita podane przez użytkownika

   Wynik:
   Większa z podanych liczb



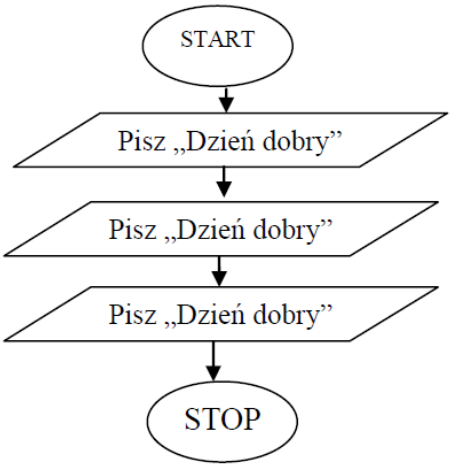
8. **Wybór większej liczby**
   (podawane liczby *a* i *b* mogą być równe!) Algorytm wczytuje dwie liczby i wyświetla większą, lub komunikat „liczby są równe".

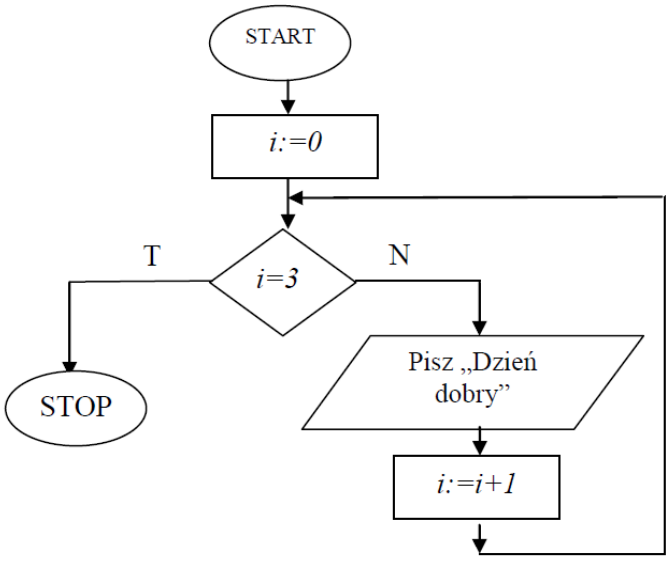


Temat: Algorytm rekurencyjny.

Iteracja – (pętla) powtarzanie danego ciągu operacji. Wymaga zmiennej licznikowej, która sprawdza, ile razy pętla została już powtórzona.

Ćw. 1. Wypisuje **trzy razy „dzień dobry”** (algorytm liniowy i z użyciem iteracji, „i” licznik pętli).

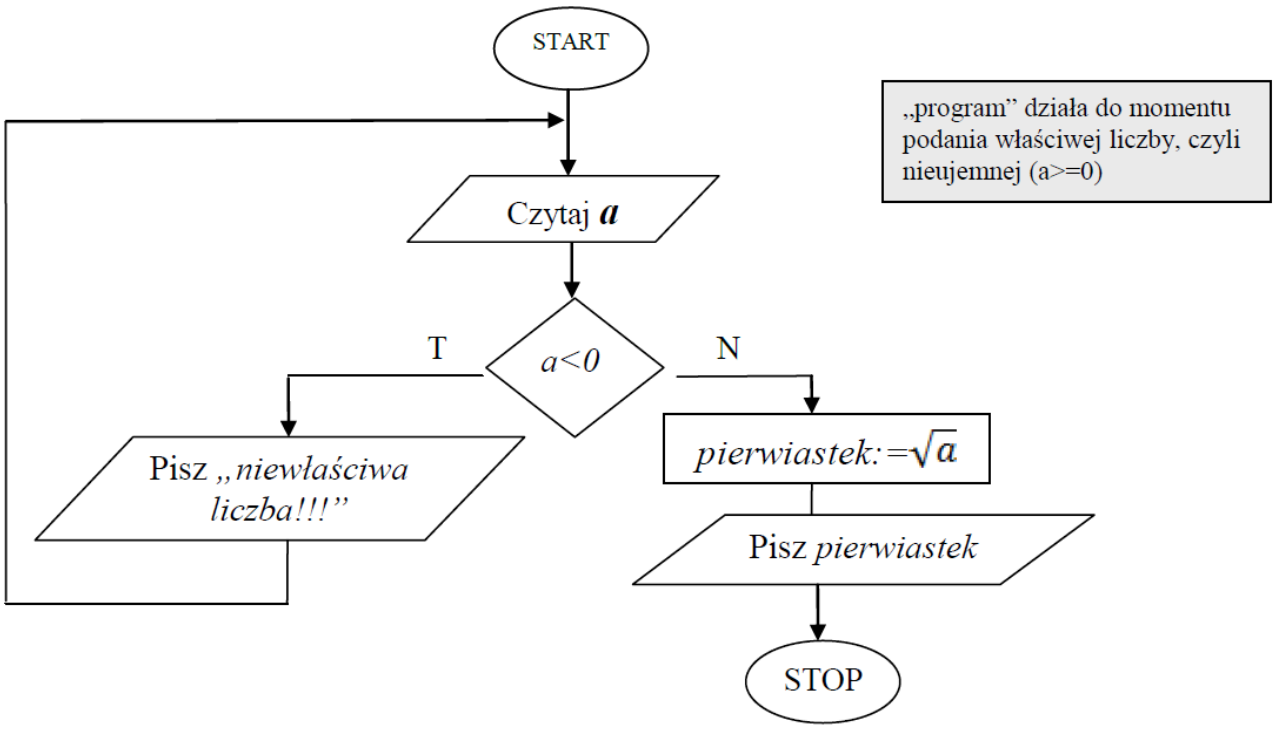




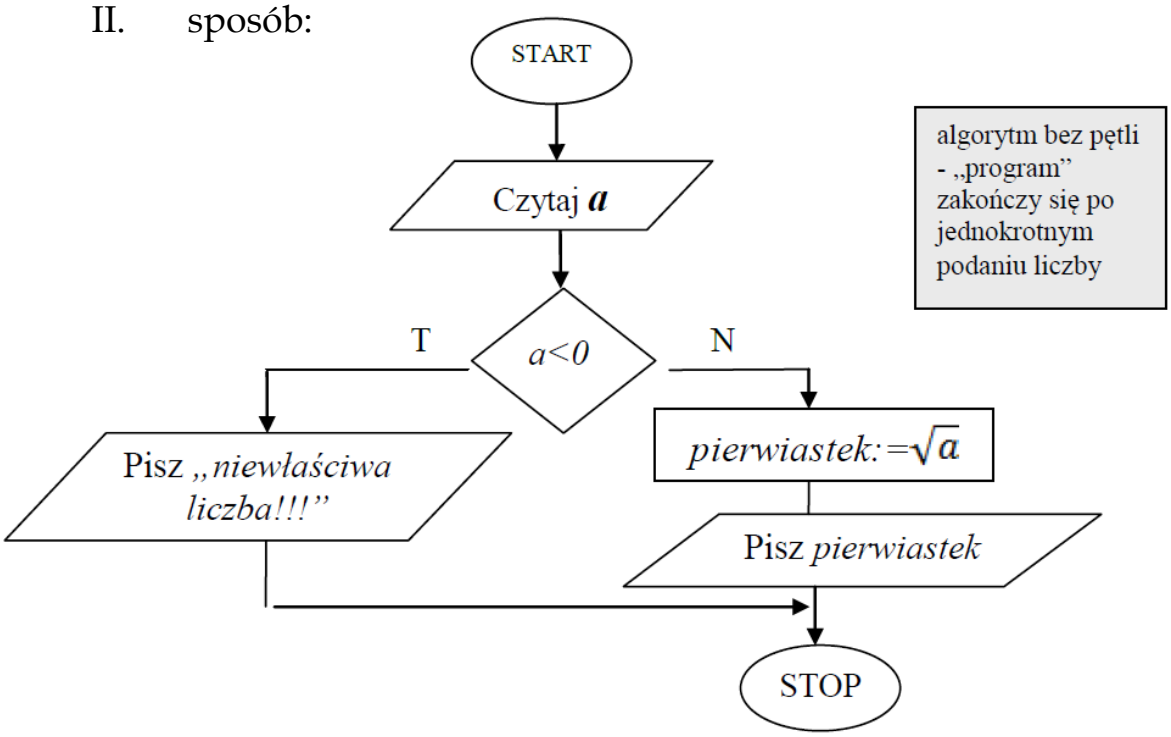
Rekurencja – („wywołanie samego siebie”) to sposób wykonywania obliczeń, polegający na tym, że wydzielony podprogram wywołuje siebie samego.

Ćw. 2. **Pierwiastek** z danej liczby (dwie możliwości zapisu algorytmu)

I. sposób:



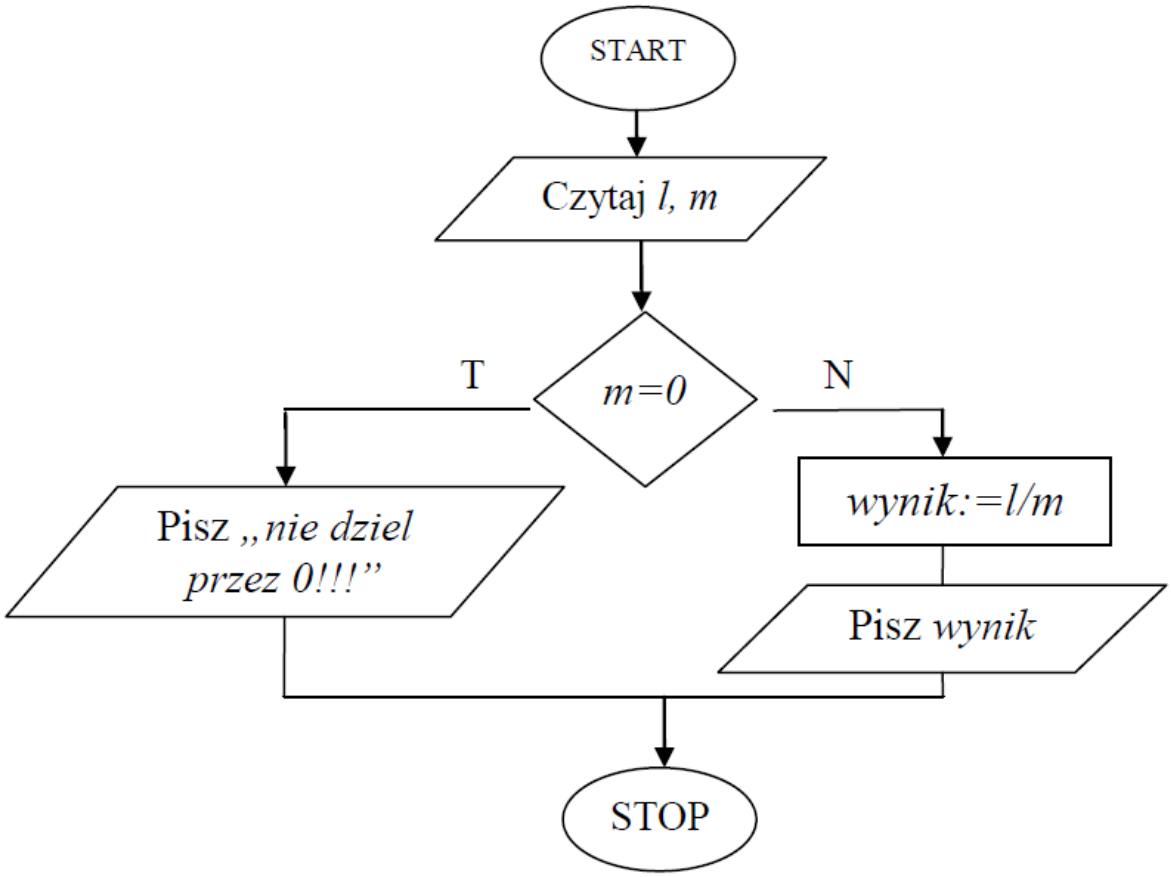
II. sposób:



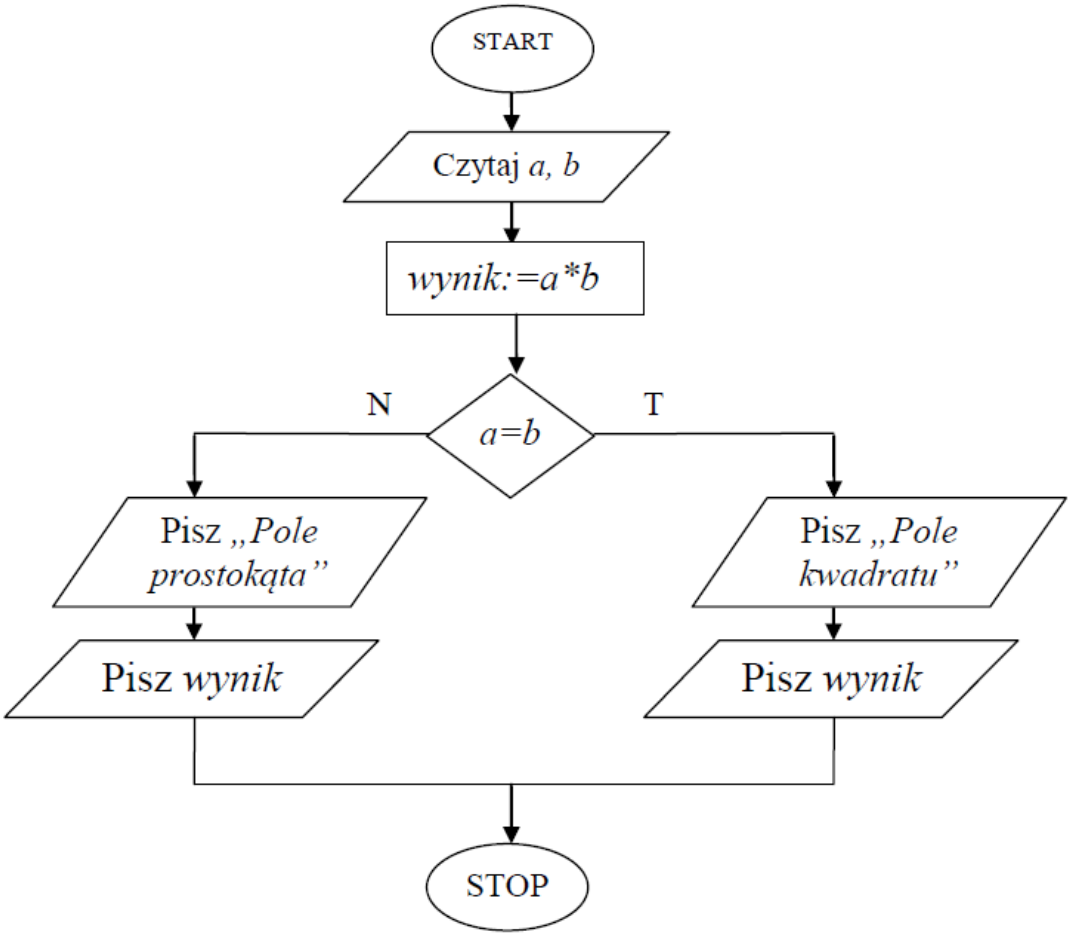
Ćw. 3 – kartki ksero

Wykonuje **dzielenie dwóch liczb.** Jeśli mianownik jest równy 0 – wyświetla się komunikat „dzielenie przez zero!", w przeciwnym wypadku wyświetla się wynik dzielenia.

Rozwiązanie:

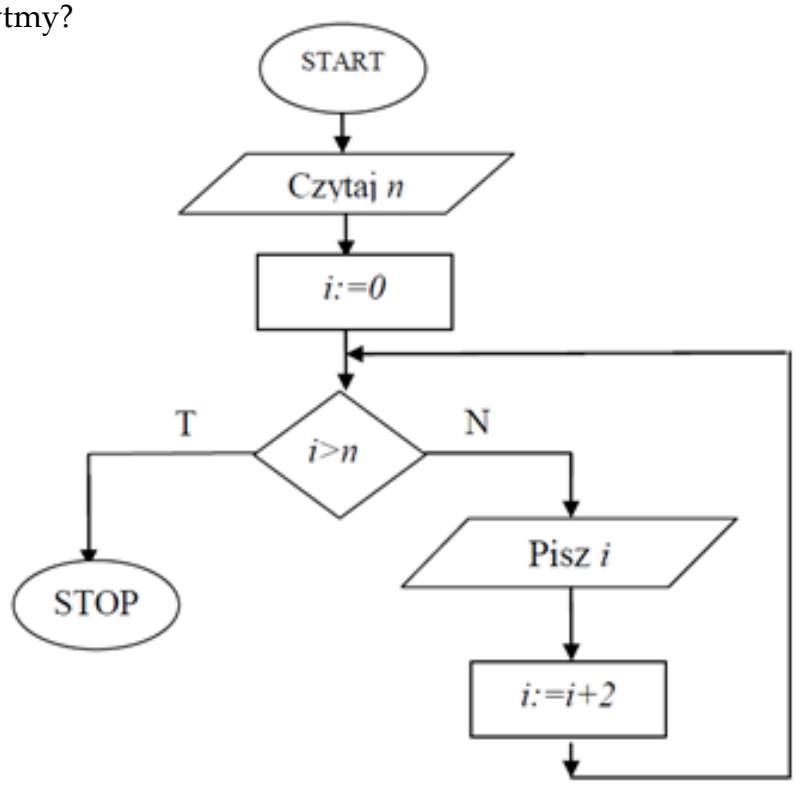


Oblicza **pole prostokąta**. Rozpoznaje czy prostokąt jest kwadratem. Jeżeli tak wyświetla się komunikat „Pole kwadratu wynosi”, w przeciwnym wypadku „Pole prostokąta wynosi” i podaje wynik.

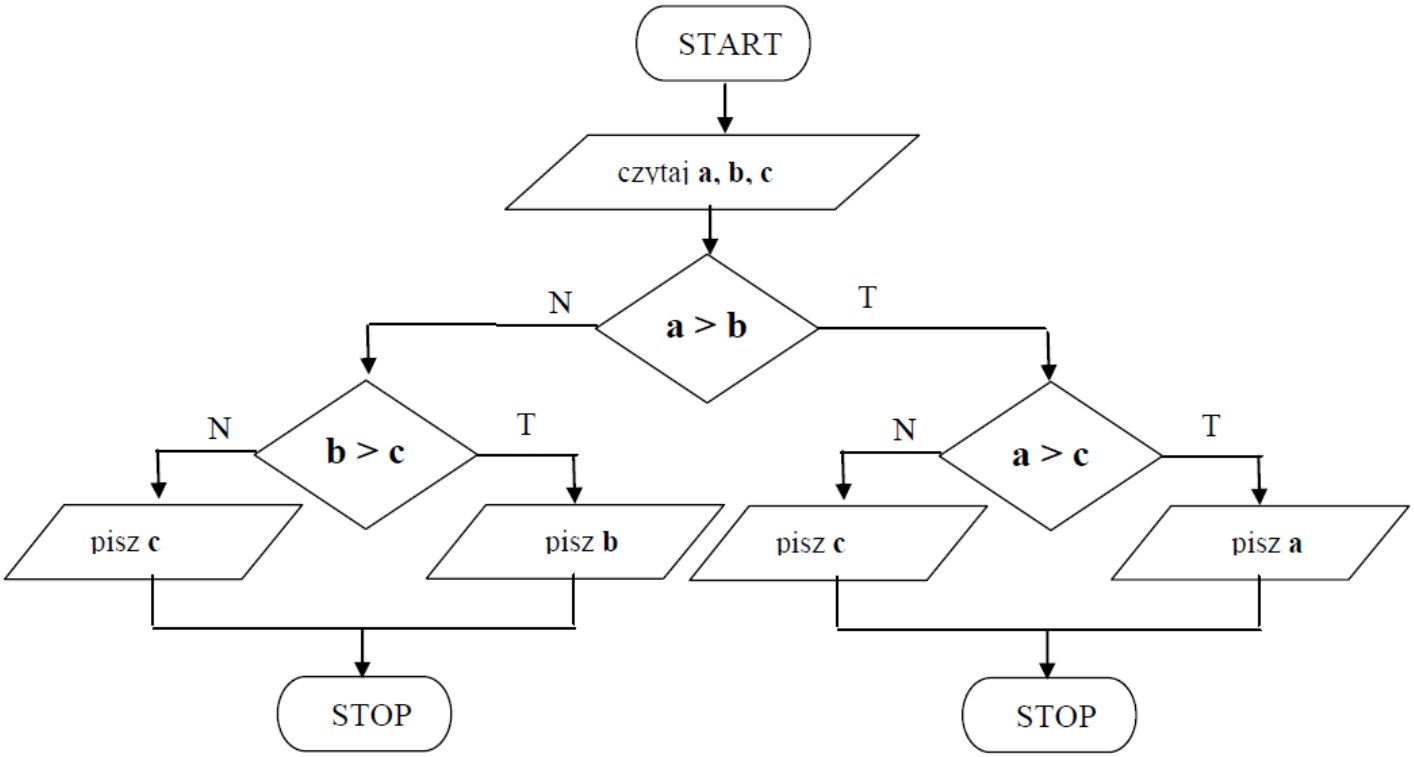


Przeanalizuj poniższe schematy blokowe.
Napisz co wykonują te algorytmy?

I.



II.



Odpowiedź:

I. Schemat blokowy algorytmu wypisującego liczby parzyste z przedziału od 0 do *n*.
Np. dla n=7, efektem działania tego algorytmu będzie ciąg liczb:
0, 2, 4, 6

krok	***n***	***i***	***Wynik działania algorytmu***
Wejście	*7*	*0*	
1	*7*	*2*	*0*
2	*7*	*4*	*0,2*
3	*7*	*6*	*0,2,4*
4	*7*	*8*	***0,2,4,6***
	Stop (bo i>n)		

<u>Uwaga:</u>
Żeby algorytm wypisywał liczby **nieparzyste** z przedziału od *0* do *n*, wystarczy w algorytmie (w trzecim klocku) przypisać ***i:=1***
(nie 0!)

II. Algorytm wybierający największą z trzech liczb.

Temat: Dziś i jutro technologii informacyjnej.

Język Logo Komeniusz

Język Logo – to język strukturalny, umożliwiający dzielenie algorytmu na wyraźnie wyodrębnione problemy, których rozwiązanie opisuje się za pomocą **procedur** (tzn. poleceń zrozumiałych dla tego języka). W programie Logo Komeniusz Procedury pierwotne dzielimy na polecenia (**instrukcje**) i **funkcje** (procedury, które mają wartość).

Procedury pierwotne (polecenia): (po wpisaniu procedury wciskamy klawisz Enter)
pż – (pokaż żółwia) żółw pokazuje się na ekranie
sż – (schowaj żółwia) żółw przestaje być widoczny na ekranie
naprzód *liczba_kroków* lub **np** *liczba_kroków*– przesunięcie żółwia do przodu o podaną liczbę kroków (np. np 100)
wstecz *liczba_kroków* lub **ws** *liczba_kroków* – przesunięcie żółwia do tyłu o podaną liczbę kroków (np. ws 70)
prawo *kąt* lub **pw** *kąt* – obrót żółwia w prawo o podany w stopniach kąt (np. pw 90)
lewo *kąt* lub **lw** *kąt* – obrót żółwia w lewo o podany w stopniach kąt (np. lw 45)
podnieś lub **pod –** powoduje podniesienie pisaka żółwia
opuść lub **opu -** powoduje opuszczenie pisaka żółwia
zmaż – wymazuje wszystkie rysunki z ekranu bez zmiany pozycji żółwia
wróć – ustawia żółwia w pozycji początkowej na środku ekranu
czyść lub **cs** - wymazuje wszystkie rysunki z ekranu i ustawia żółwia w pozycji początkowej
UstalGrubośćPisaka *grubość* lub **ugp** *grubość* -ustala jaką szerokość ma mieć „rysik" pisaka
UstalKolorPisaka *nr_koloru* lub **ukp** *nr_koloru* - ustala kolor pisaka, pod numerami od 1 do 15 są poszczególne kolory
UstalTło *nr_koloru* - ustala kolor tła
zamaluj - Należy podnieść pióro, ustawić się wewnątrz figury zamkniętej i opuścić pióro. Zamaluje (wypełni) wnętrze tej figury kolorem, który jest aktualnie ustalony (poleceniem ukp)
powtórz *n* [*lista_poleceń*] - powoduje n-krotne powtórzenie listy poleceń (np. powtórz 2 [np. 50 pw 90])

Ćwiczenie 1.

Rysowanie figur geometrycznych: (po każdym poleceniu naciskamy Enter lub wszystkie polecenia piszemy w jednym wierszu i na końcu naciskamy Enter)

a) **Kwadrat** o boku długości 100 (np 100 pw 90 np 100 pw 90 np 100 pw 90 np 100 pw 90)
b) **Prostokąt** o bokach 200 na 100
c) **Trójkąt równoboczny** o boku długości 150
d) **Sześciokąt foremny** o boku 40
e) **Pięciokąt foremny** o boku 70

Ćwiczenie 2.

Wykonaj jeszcze raz powyższe ćwiczenia korzystając z polecenia **powtórz**.
Wszystkie rozwiązania notuj w zeszycie !!!

Ćwiczenie 3.
Zdefiniuj procedury: *kwadrat, trójkąt, prostokąt, pięciokąt, sześciokąt* – rysujące odpowiednie figury o wybranej przez Ciebie wielkości.

Rozwiązanie:
Oto *kwadrat*
    Powtórz 4[np 100 pw 90]
Już

Oto *trójkąt*
    Powtórz 3[np 100 pw 120]
Już

Oto *prostokąt*
    Powtórz 2[np 100 pw 90 np 200 pw 90]
Już

360/5

Oto *pięciokąt*
    Powtórz 5[np 100 pw 72]
Już

360/6

Oto *sześciokąt*
    Powtórz 6[np 100 pw 60]
Już

Parametry formalne: *n* i *bok*

Oto *wielokąt* :n :bok
    Powtórz :n[np :bok pw 360/:n]
Już

Parametry aktualne – konkretne wartości

Przykładowe wywołanie:
wielokąt 8 150 (efekt - ośmiokąt foremny o boku 150)

Ćwiczenie 4. Wykonaj rysunek: