

ALGORYTMY

Algorytm – to przepis; zestawienie kolejnych kroków prowadzących do wykonania określonego zadania; to uporządkowany sposób postępowania przy rozwiązywaniu zadania, problemu, z uwzględnieniem opisu danych oraz opisu kolejnych czynności prowadzących do jego rozwiązania w skończonym czasie. Każdy program komputerowy realizuje jakiś algorytm zapisany w zrozumiałym dla komputera języku programowania. Istnieją zadania „niealgorytmiczne”, dla których nie można opracować algorytmu rozwiązania.

Algorytm musi być:

- **poprawny** (dla każdego poprawnego zestawu danych – otrzymujemy poprawny wynik)
- **jednoznaczny** (dla tych samych danych – uzyskujemy ten sam wynik)
- **szczegółowy** (aby jego wykonawca rozumiał opisane czynności i potrafił je wykonać)
- **uniwersalny** (ogólny, aby służył do rozwiązywania pewnej grupy zadań, a nie tylko jednego zadania np. sumy dwóch dowolnych liczb naturalnych, a nie tylko 3+2)

inne cechy algorytmu to:

- **skończoność** - dla każdego zestawu poprawnych danych wejściowych, algorytm powinien dawać wyniki w skończonej liczbie kroków
- **efektywność (sprawność)** – powinien prowadzić do rozwiązania problemu jak najniższym kosztem, czyli w jak najmniejszej liczbie kroków. Należy zoptymalizować pamięć zajętą przez struktury danych wykorzystywane w algorytmie oraz doprowadzić do optymalizacji złożoności obliczeniowej, czyli liczby wykonanych operacji.

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)

Specyfikacja algorytmu - obejmuje podanie:

- danych wejściowych (czyli nazwy używanych zmiennych i ich typ – np. liczba całkowita, rzeczywista, wartość logiczna)
- wyniku, który algorytm powinien otrzymać
- zmiennych pomocniczych niezbędnych do realizacji programu.

Uniwersalny algorytm operuje nie na liczbach, a na **zmiennych** (czyli „pojemnikach na dane” oznaczonych dowolną literą lub łańcuchem znaków).

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. Liczba powtórzeń może być z góry określona - tzw. pętla „for”; dana instrukcja jest powtarzana aż do spełnienia jakiegoś określonego warunku – tzw. pętla „do while”; najpierw jest sprawdzany warunek a jego spełnienie umożliwia wykonanie instrukcji – tzw. pętla „while do”)

Znaczenie klocków. Blok:

– **startowy** (rozpoczęcie wykonywania algorytmu)

START

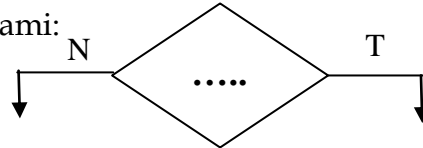
– **wejścia** (wczytywanie danych, przypisywanie ich zmiennym)

czytaj

– **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)

pisz

– **końcowy** („stop”, koniec działania algorytmu)

STOP

Pojęcia:

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

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

Tablica – to jeden ze strukturalnych typów danych. Pod jedną nazwą zmiennej można umieścić więcej danych. Tablica składa się z ustalonej liczby elementów tego samego typu. Za ich pomocą można reprezentować regularne struktury danych, jak np. macierze czy wektory. Dostęp do poszczególnych elementów tablicy uzyskuje się za pomocą indeksów. Np. zmienna tablicowa „tab” ma wartości:

23	1111	10	3	78888
----	------	----	---	-------

Do kolejnych wartości odwołujemy się używając indeksów. Np. wartość 10 wskażemy używając indeksu $\text{tab}[3]$, wartość 78888 – $\text{tab}[5]$, itd..

PRZYKŁADY ALGORYTMÓW

(specyfikacja, **schemat blokowy**, układ klocków w języku ELI, pseudokod)

Uwaga !!! Oznaczenia operatorów relacyjnych:

> większe

< mniejsze

> = większe bądź równe (\geq)

< = mniejsze bądź równe (\leq)

<> różne (\neq)

:= instrukcja przypisania - przypisywanie zmiennej wartości podanej z prawej strony (np.: $y := x*x*x$, oznacza, że zmienna („literka”) y ma teraz wartość x^3)

Czytaj \leftrightarrow wczytaj \leftrightarrow podaj

Pisz \leftrightarrow wypisz

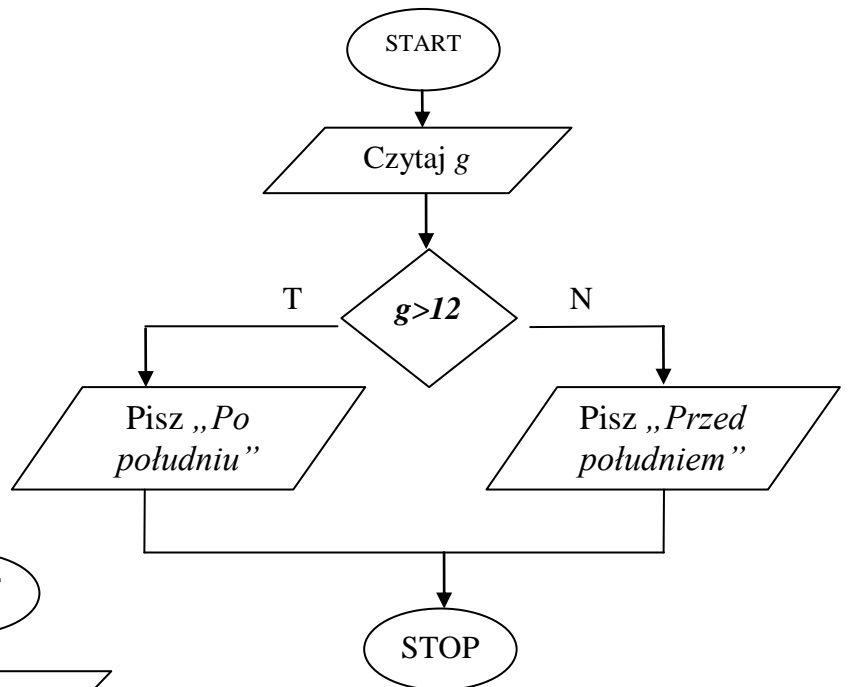
Funkcje, operatory:

- **mod** - zwraca resztę z dzielenia całkowitego, np. $5 \bmod 2 = 1$, $4 \bmod 2 = 0$,

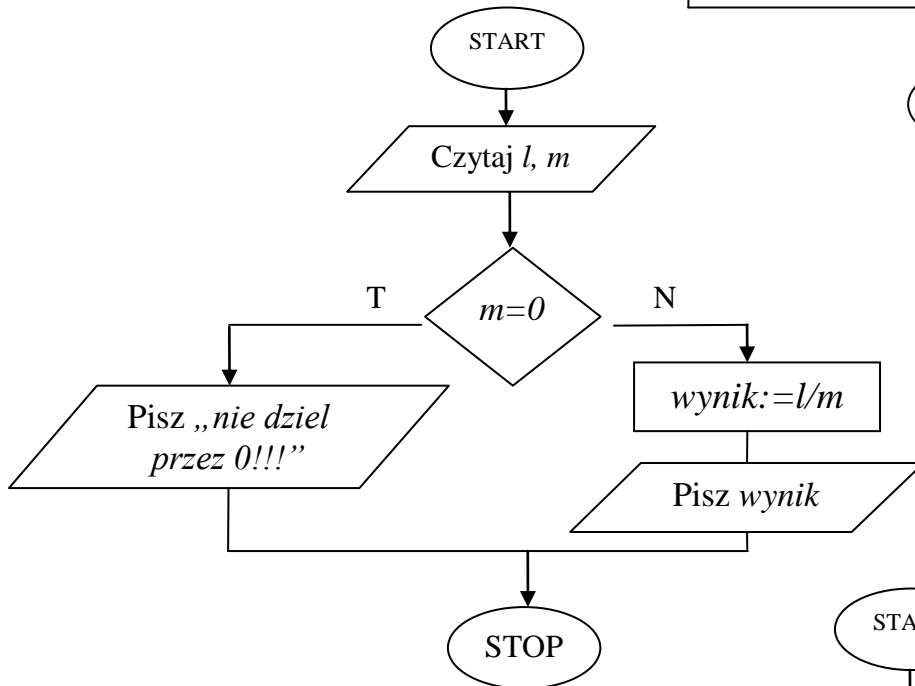
- **div** - zwraca część całkowitą z dzielenia, np. $5 \text{ div } 2 = 2$, $5 \text{ div } 3 = 1$,

- **random(n)** - zwraca losową liczbę naturalną z zakresu 0 do „n”,

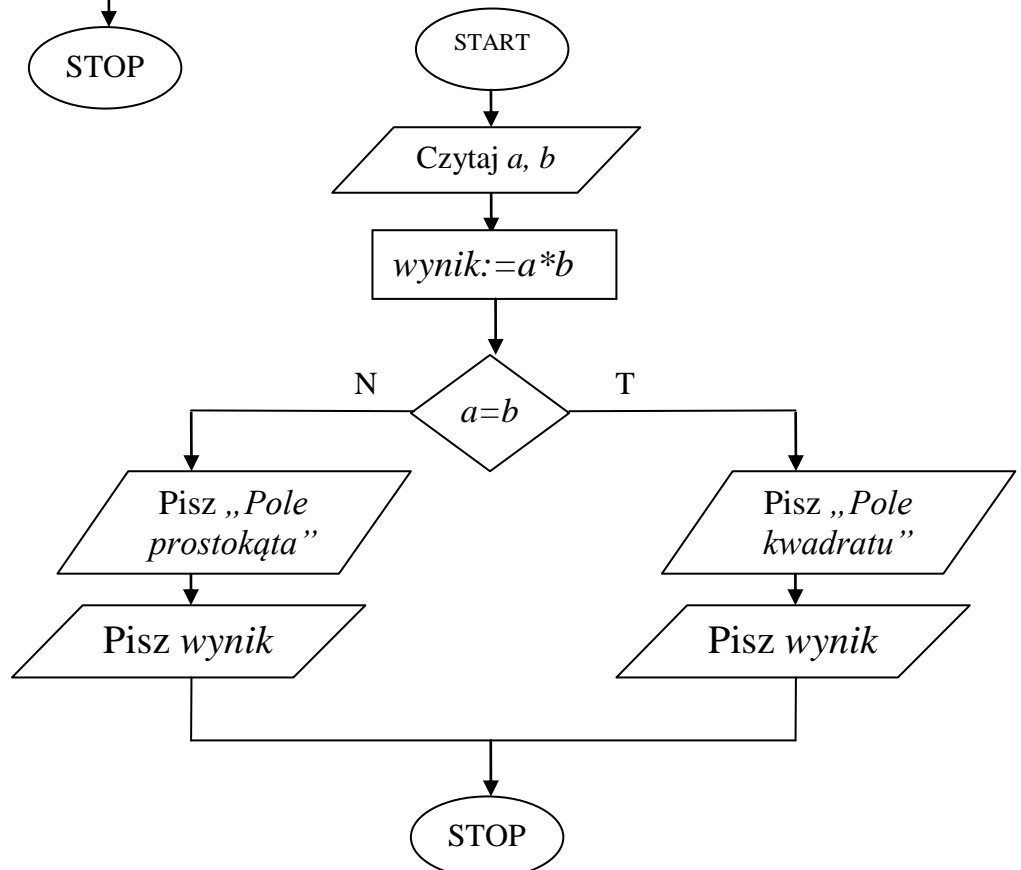
1. Algorytm na podstawie wprowadzonej godziny rozpoznaje czy podana przez użytkownika godzina („g”) jest przedpołudniowa, czy popołudniowa.



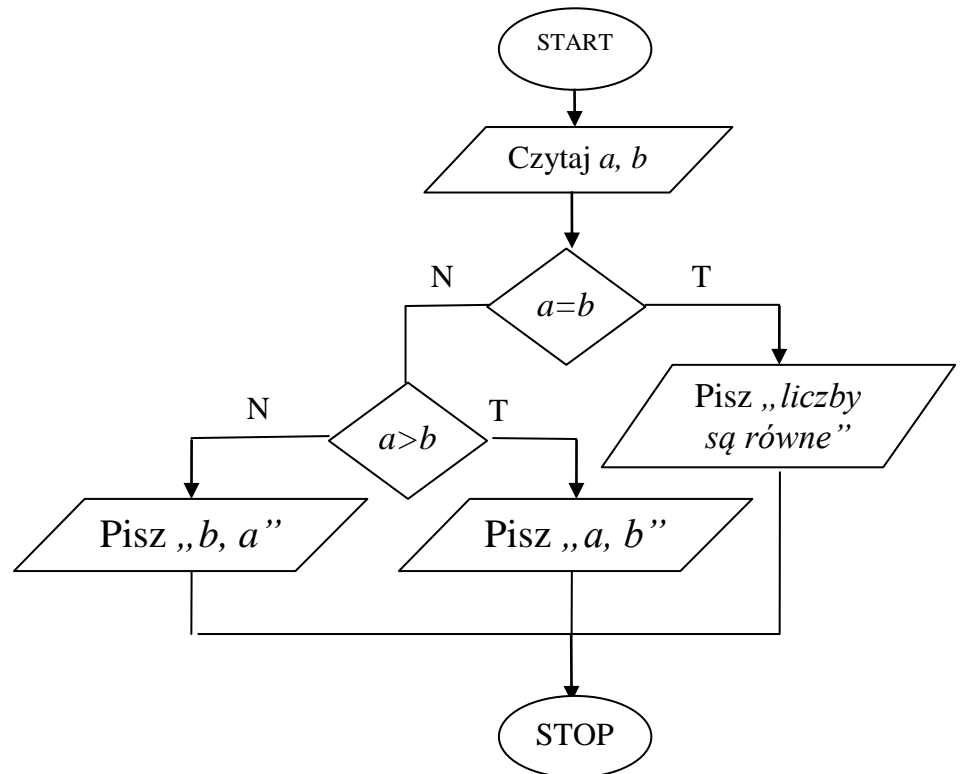
2. Wykonuje dzielenie dwóch liczb. Jeśli mianownik jest równy 0 - wyświetla się komunikat „dzielenie przez zero!”, w przeciwnym wypadku wyświetla się wynik dzielenia.



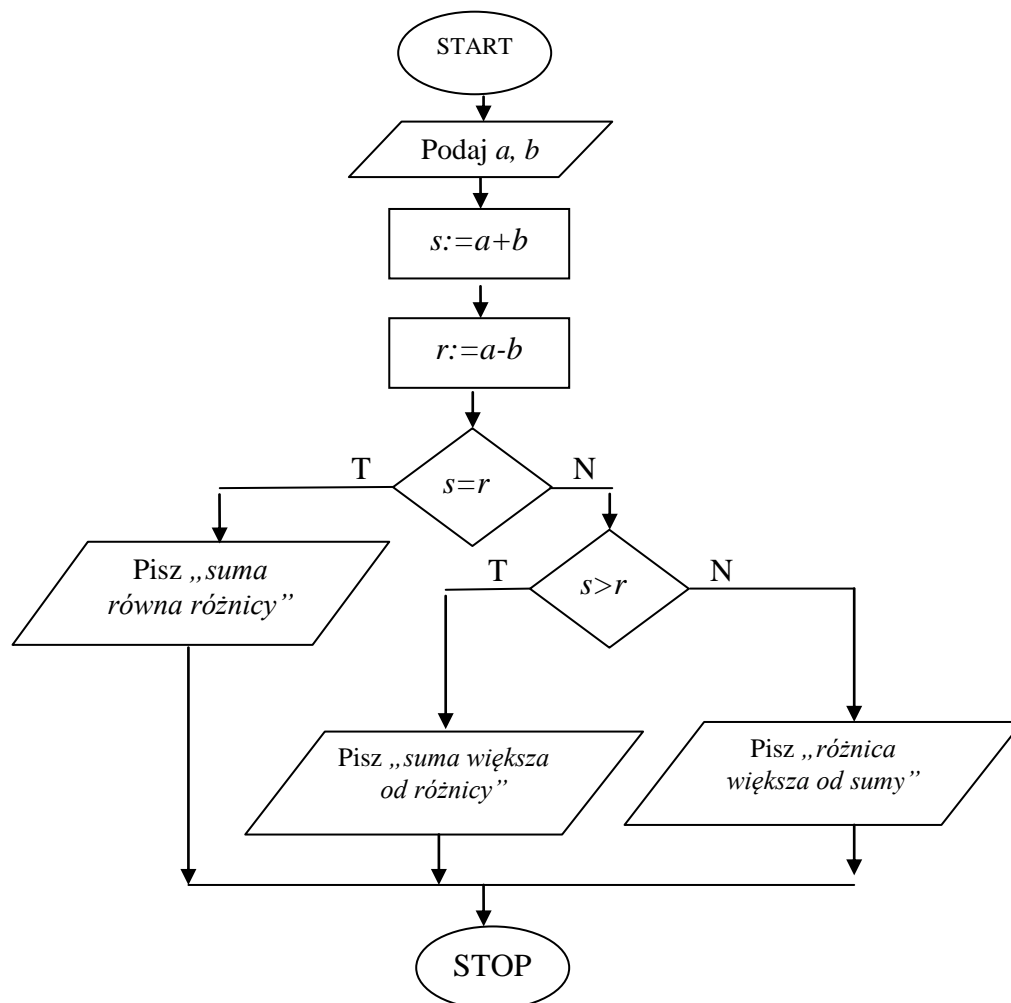
3. Oblicza pole prostokąta. Rozpoznaje czy prostokąt jest kwadratem. Jeżeli tak wyświetla się komunikat „Pole kwadratu wynosi”, w przeciwnym wypadku „Pole prostokąta wynosi” i podaje wynik.



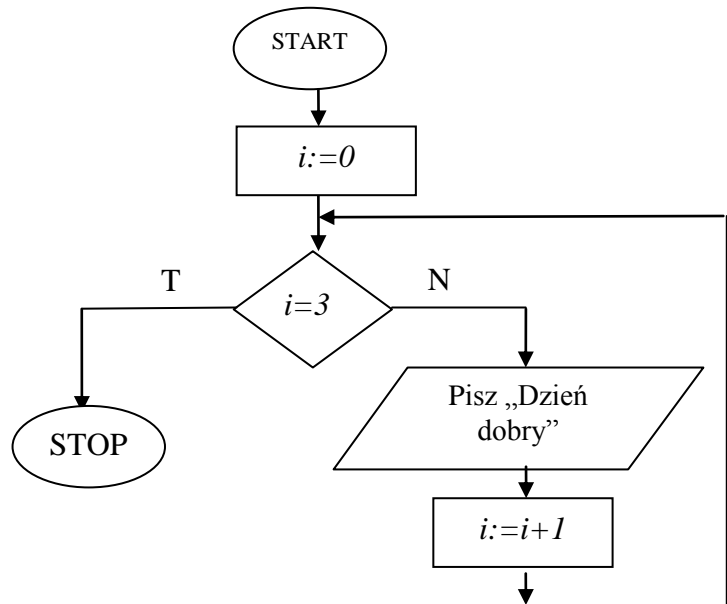
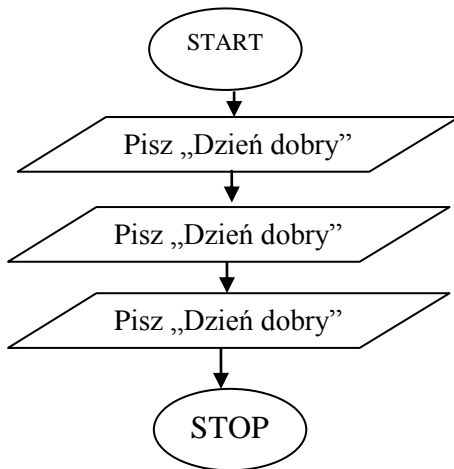
4. Wczytuje dwie liczby i wypisuje je w kolejności malejącej. Jeśli są równe, wyświetla komunikat „liczby są równe”.



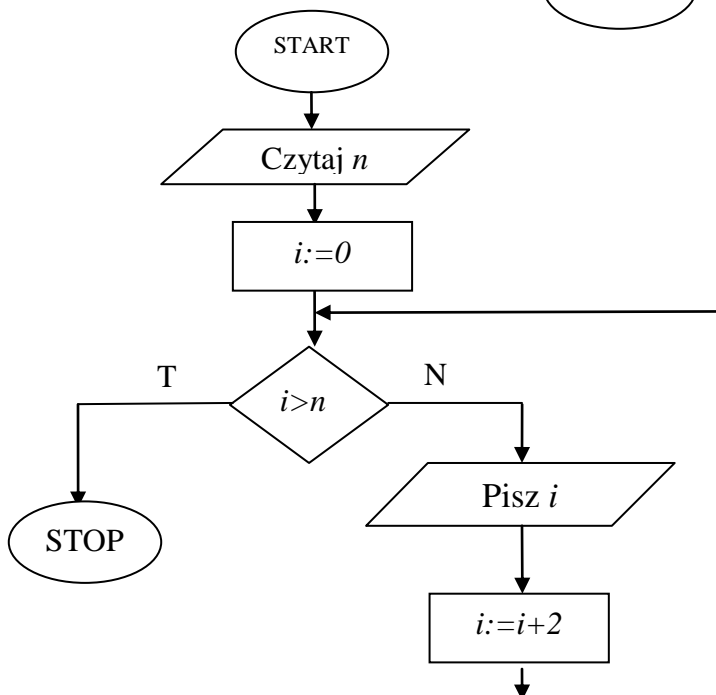
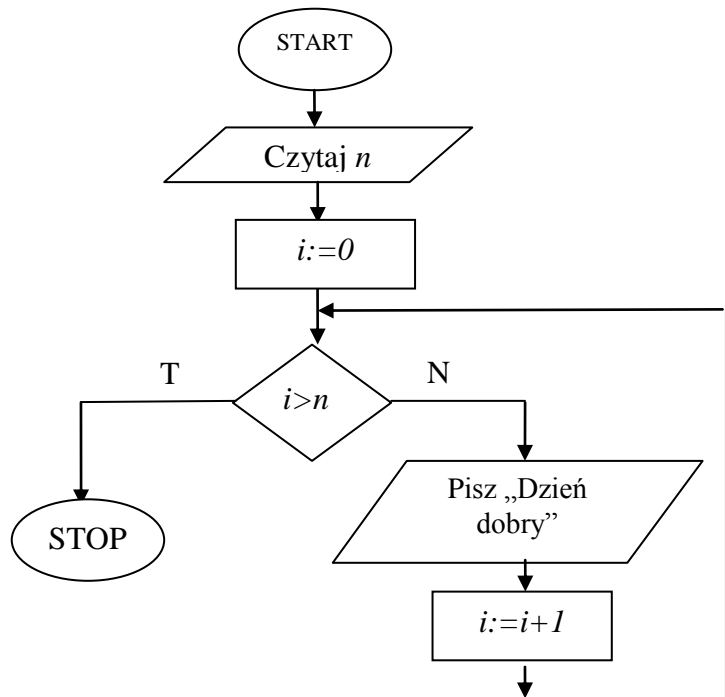
5. Wczytuje dwie liczby całkowite i oblicza ich sumę oraz różnicę, następnie w zależności od wyników, wyświetla komunikat „suma większa od różnicy” lub „różnica większa od sumy”.



6. Wypisuje trzy razy „dzień dobry”.
(algorytm liniowy i z użyciem iteracji, „i” - licznik pętli).

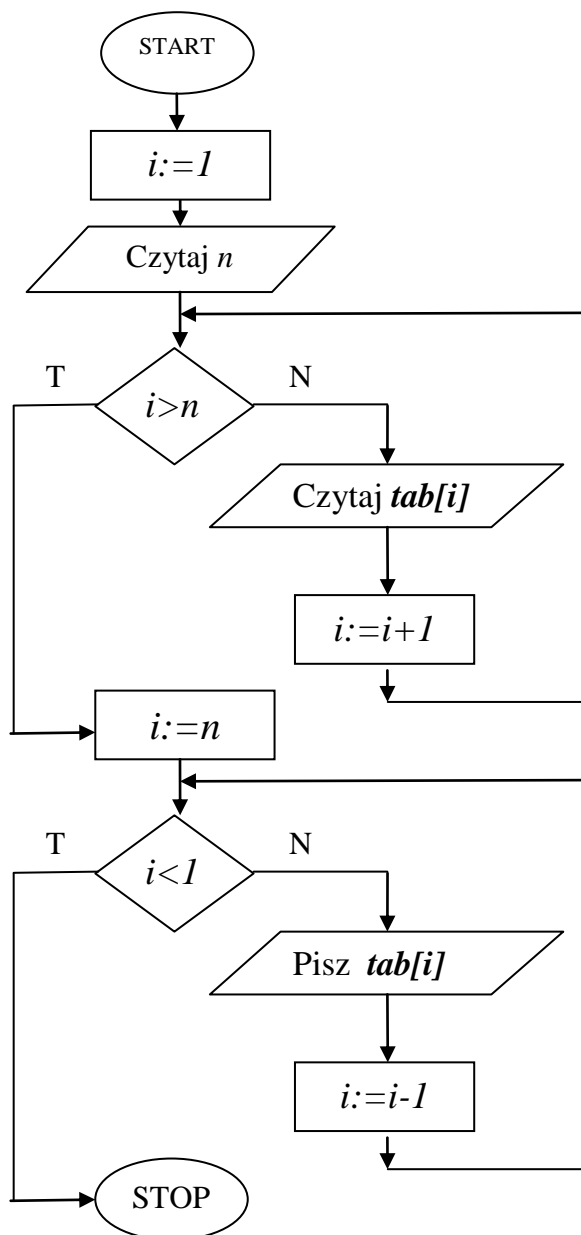
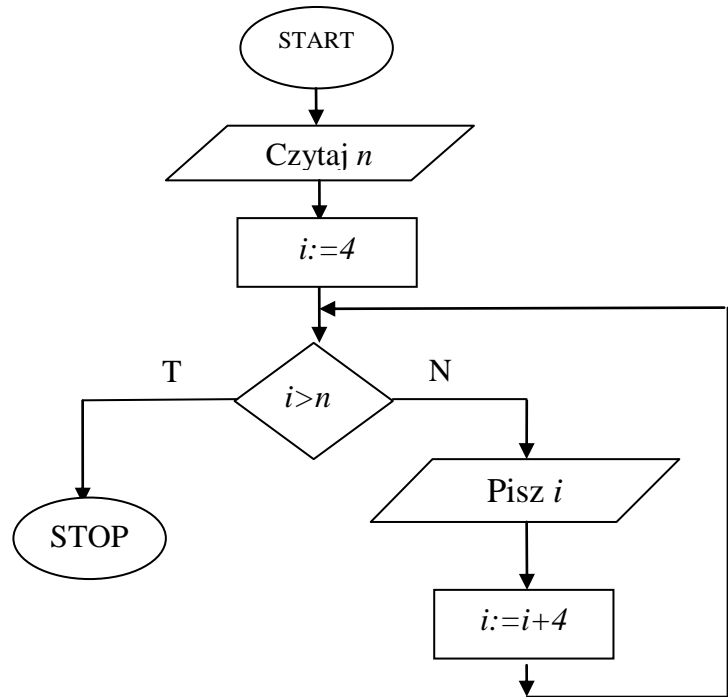


7. Wypisuje „n” razy „dzień dobry”
(„n” - liczba naturalna podana przez użytkownika, „i” - licznik pętli).



8. Wypisuje liczby parzyste z przedziału 0 do „n” („n” - liczba naturalna podana przez użytkownika).

9. Wypisuje liczby z przedziału 0 do „n”, które są podzielne przez 4.



10. TABLICA. Wczytuje do tablicy liczby, a następnie wyświetla je w odwrotnej kolejności.

11. 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

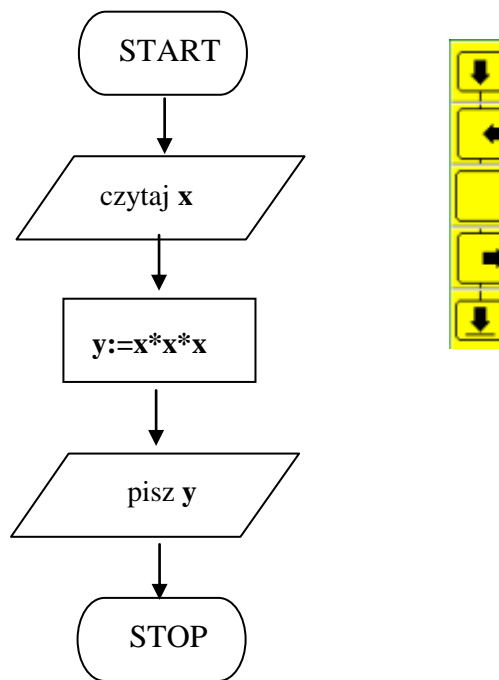
Początek

czytaj(x)

$y:=x*x*x$

pisz(y)

koniec



12. Wybór większej liczby - algorytm warunkowy

Specyfikacja:

Dane:

x, y - liczby całkowite podane przez użytkownika

Wynik:

Większa z podanych liczb

Pseudokod:

Program wieksza

Zmienne:

x, y : całkowite

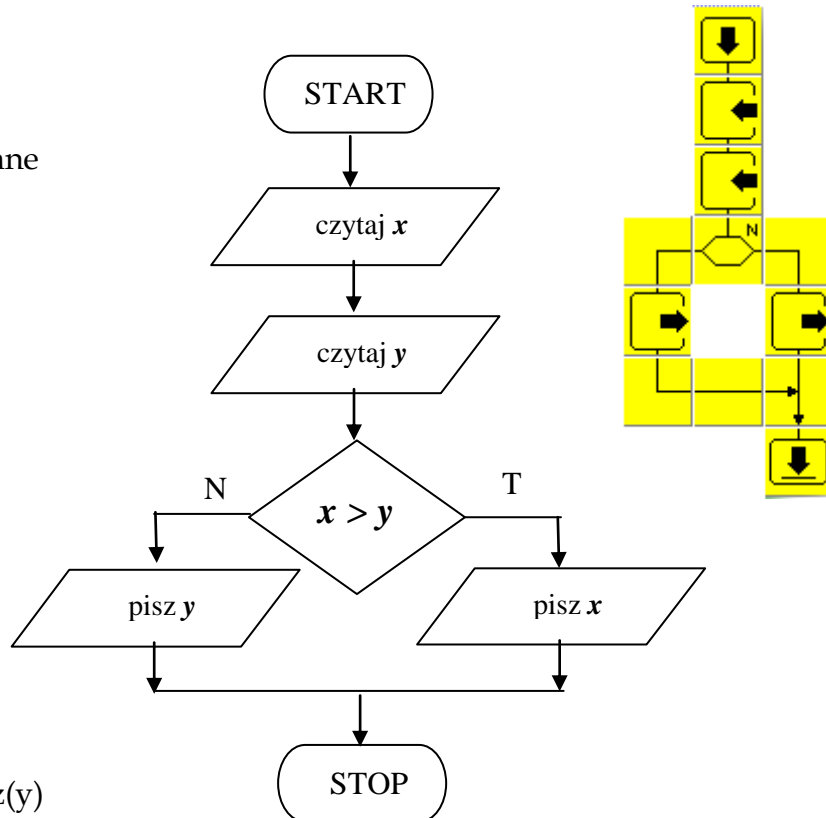
Początek

czytaj(x, y)

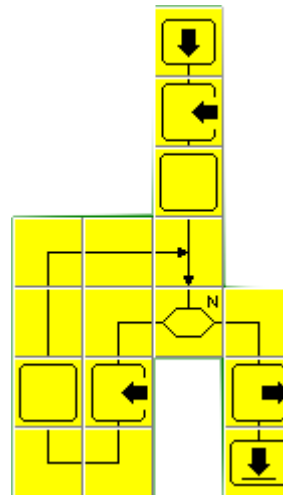
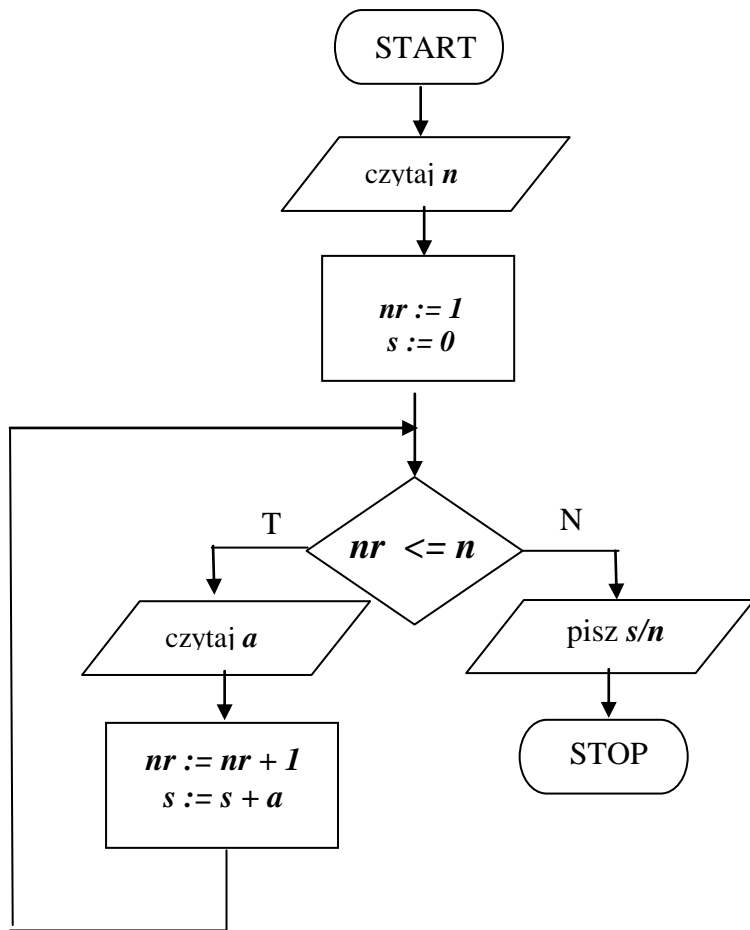
jeśli $x > y$ to pisz(x)

w przeciwnym wypadku pisz(y)

koniec

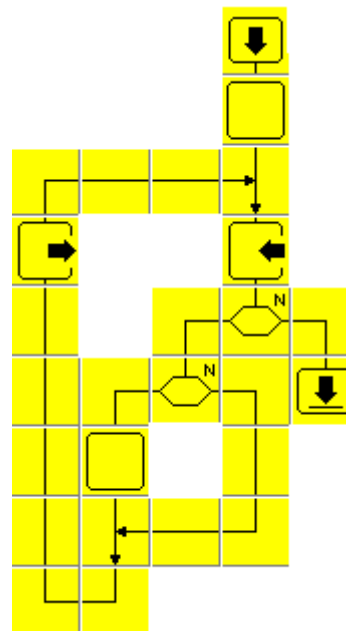
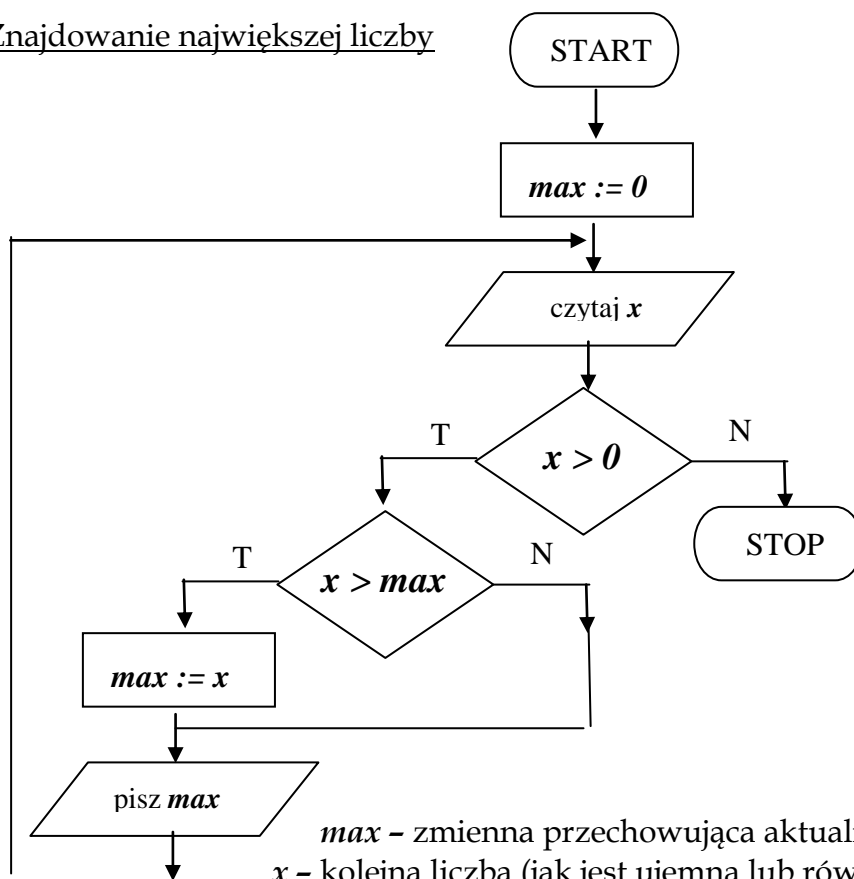


13. Sumowanie i obliczanie średniej



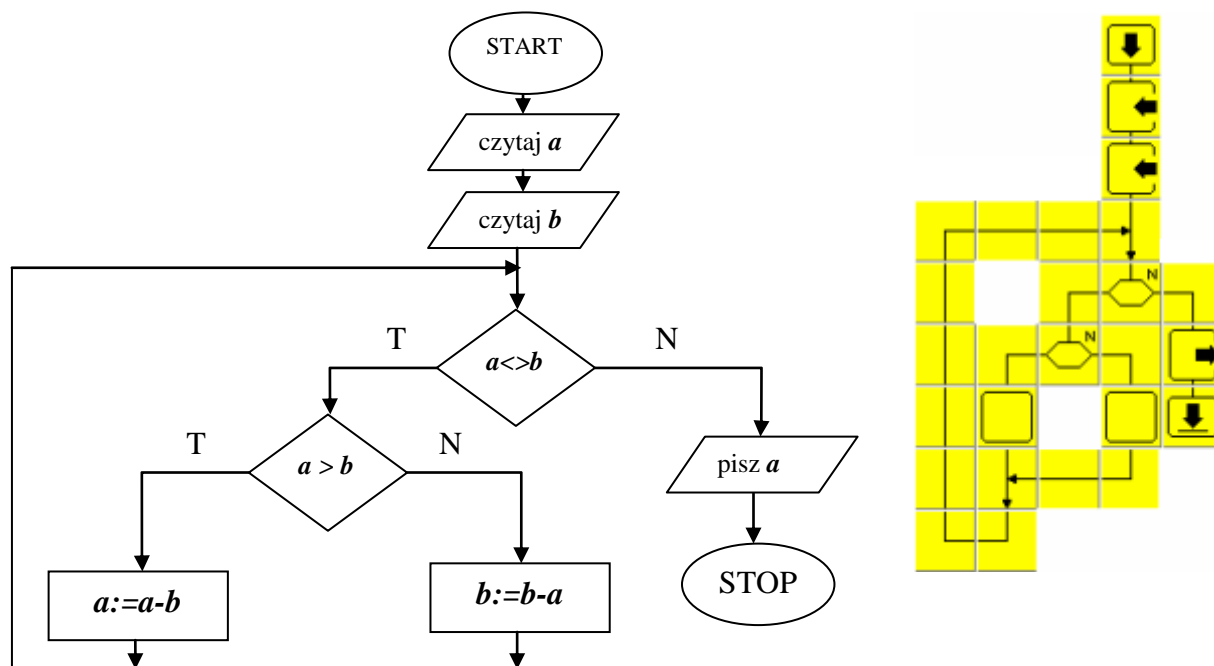
n - ilość liczb, które będziemy sumować
 nr - numer kolejnego obliczenia
 (kolejnej wprowadzanej liczby)
 s - suma
 a - kolejna liczba
 s/n - średnia

14. Znajdowanie największej liczby

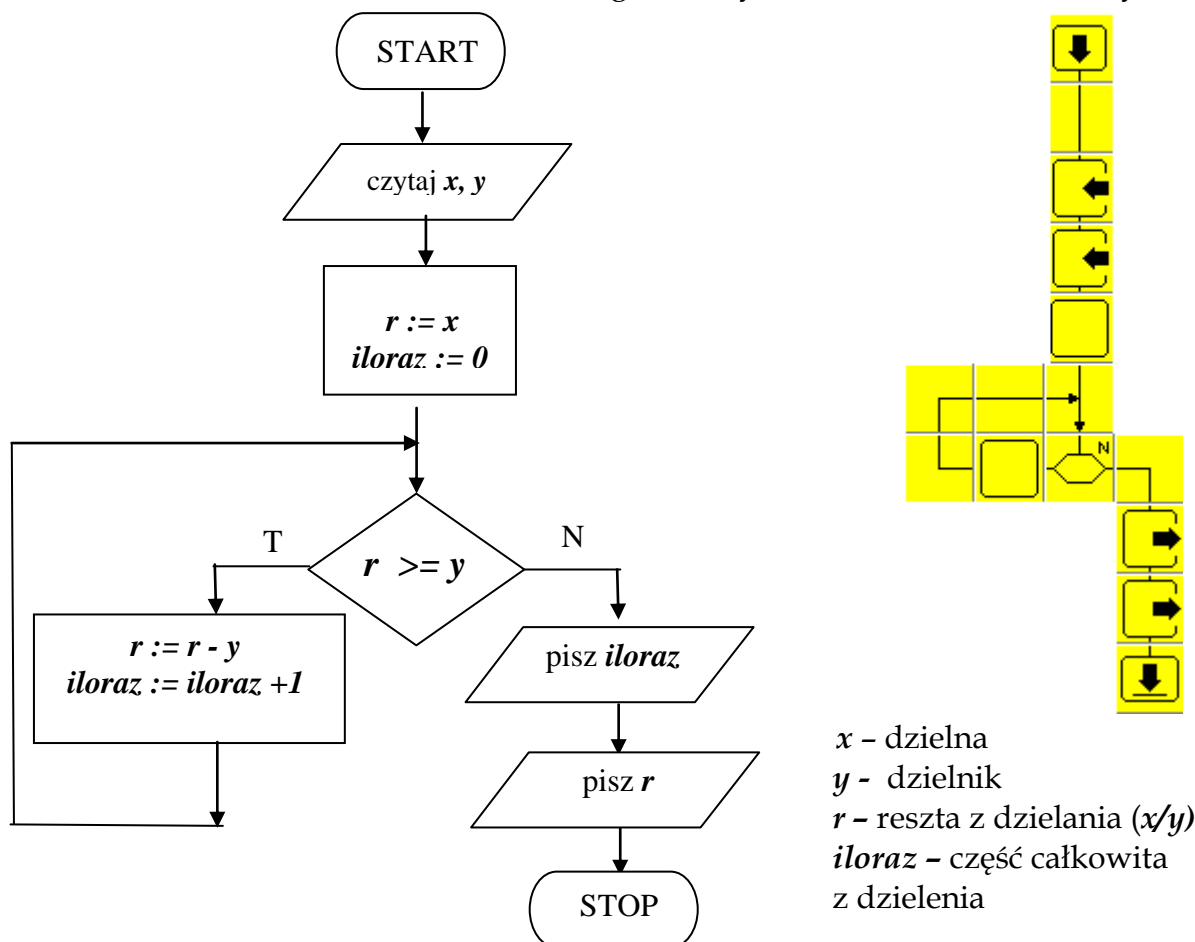


max - zmienna przechowująca aktualnie największą liczbę
 x - kolejna liczba (jak jest ujemna lub równa zero - to koniec algorytmu)

15. **Algorytm Euklidesa** - obliczanie NWD dla dwóch podanych dodatnich liczb całkowitych (wynik z odejmowania)



16. Pętla DOPÓKI. Obliczanie ilorazu całkowitego i reszty z dzielenia liczb naturalnych.

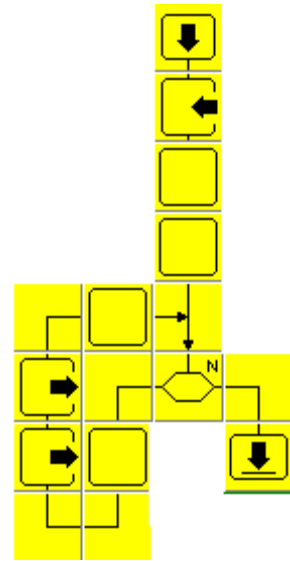
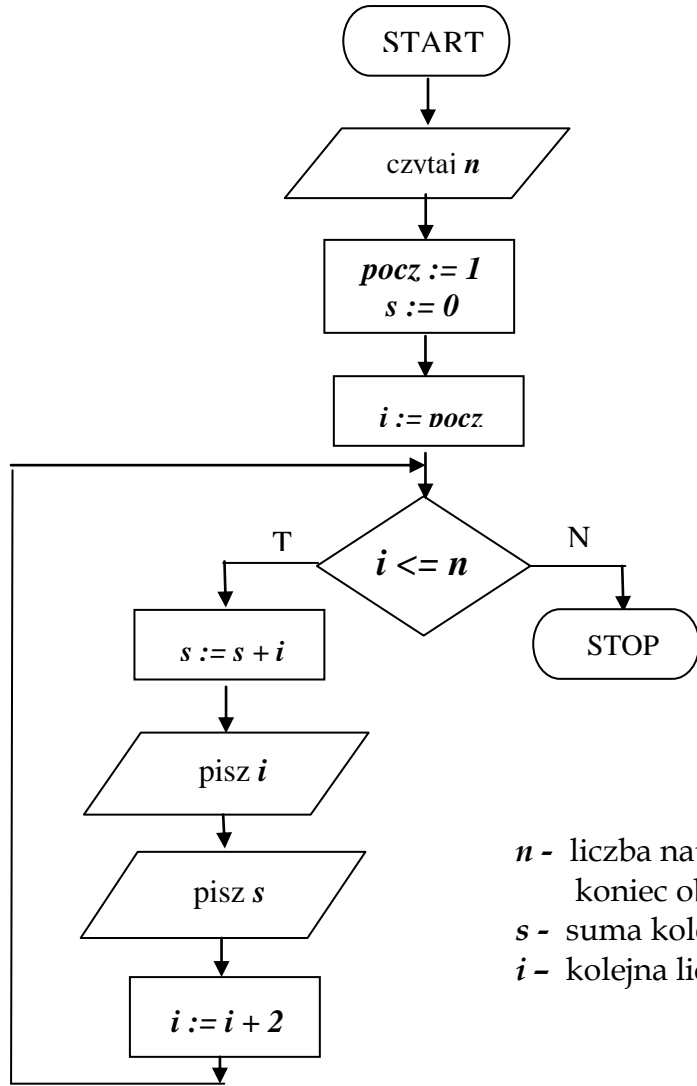


np.:

<i>x</i>	<i>y</i>	<i>r</i>	<i>iloraz</i>
5	3	5	0
		2	1

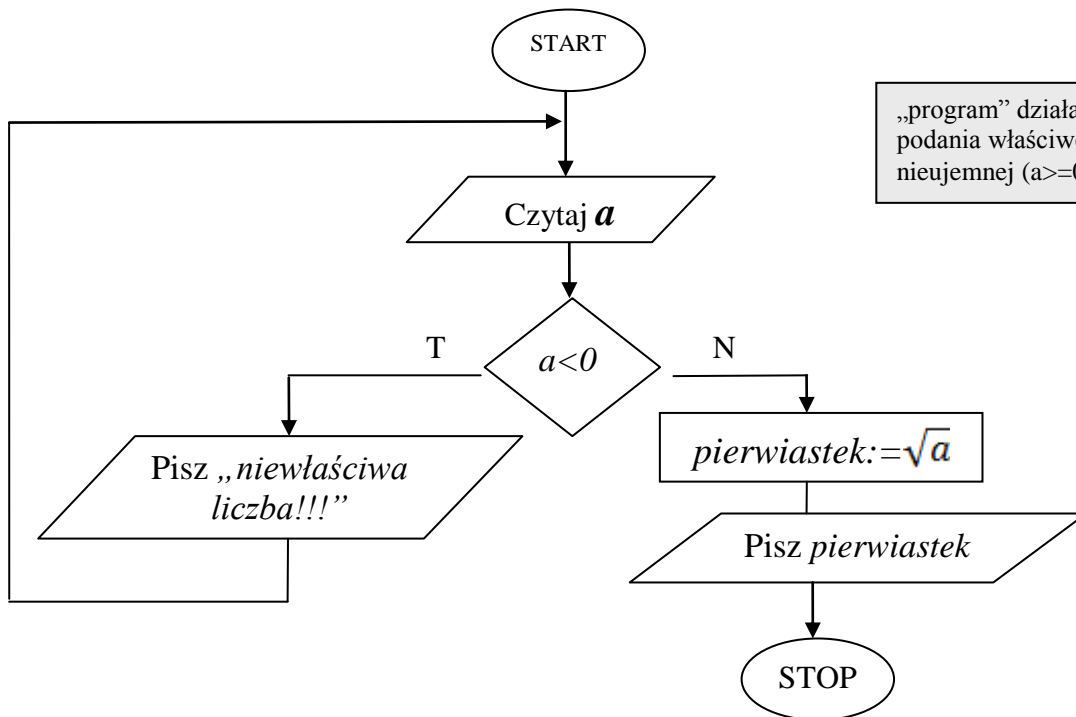
<i>x</i>	<i>y</i>	<i>r</i>	<i>iloraz</i>
8	3	8	0
		5	1
		2	2

17. Pętla DLA. Dodawanie kolejnych liczb nieparzystych.

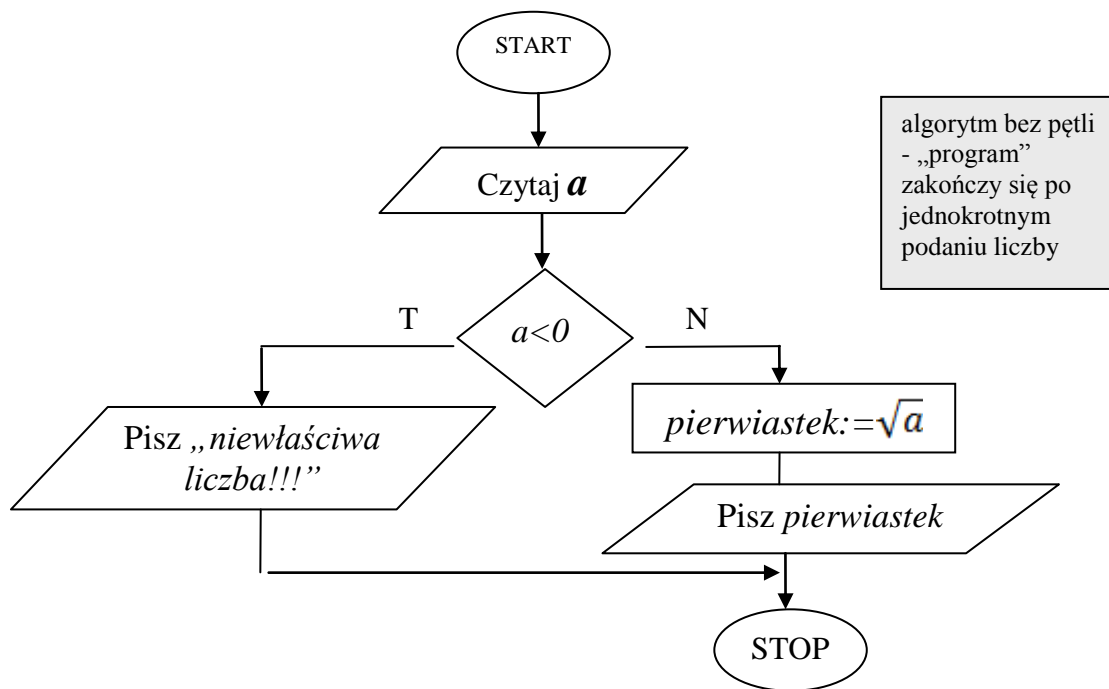


n - liczba naturalna, nieparzysta, na której ma być koniec obliczeń
 s - suma kolejnych liczb nieparzystych aż do n -tej
 i - kolejna liczba nieparzysta - licznik (1, 3, 5, ..., n)

18. Pierwiastek z danej liczby (dwie możliwości zapisu algorytmu)

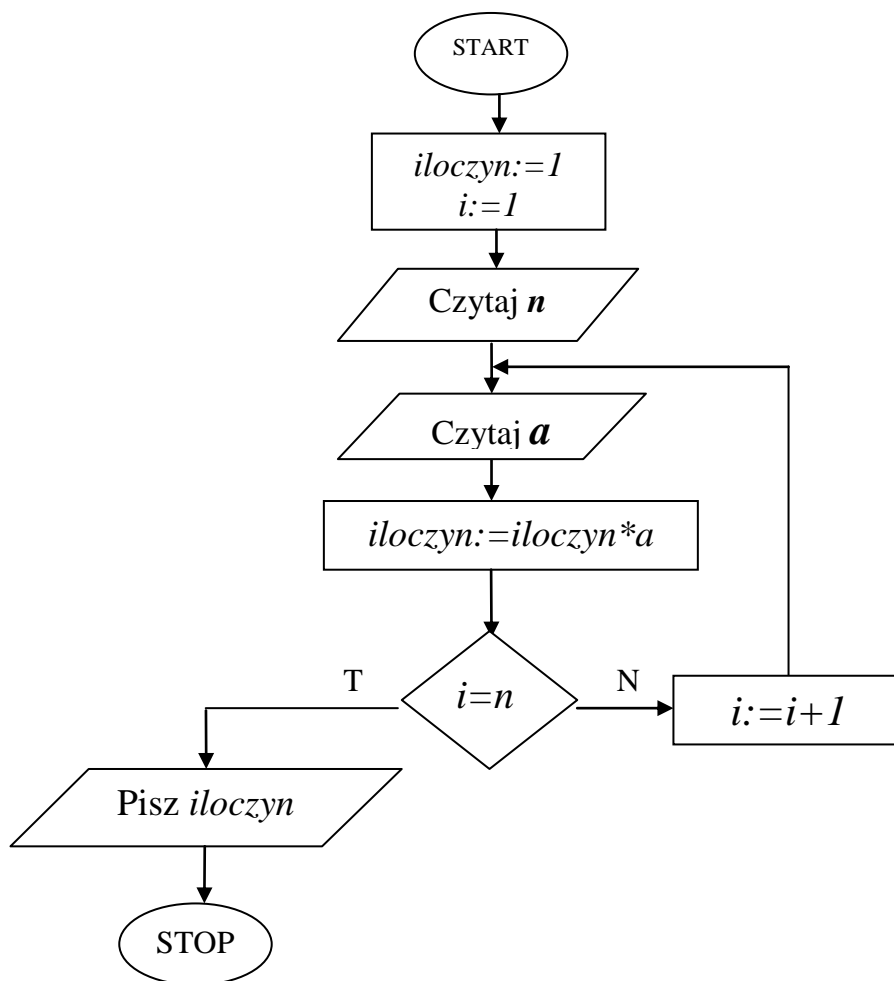


„program” działa do momentu podania właściwej liczby, czyli nieujemnej ($a \geq 0$)

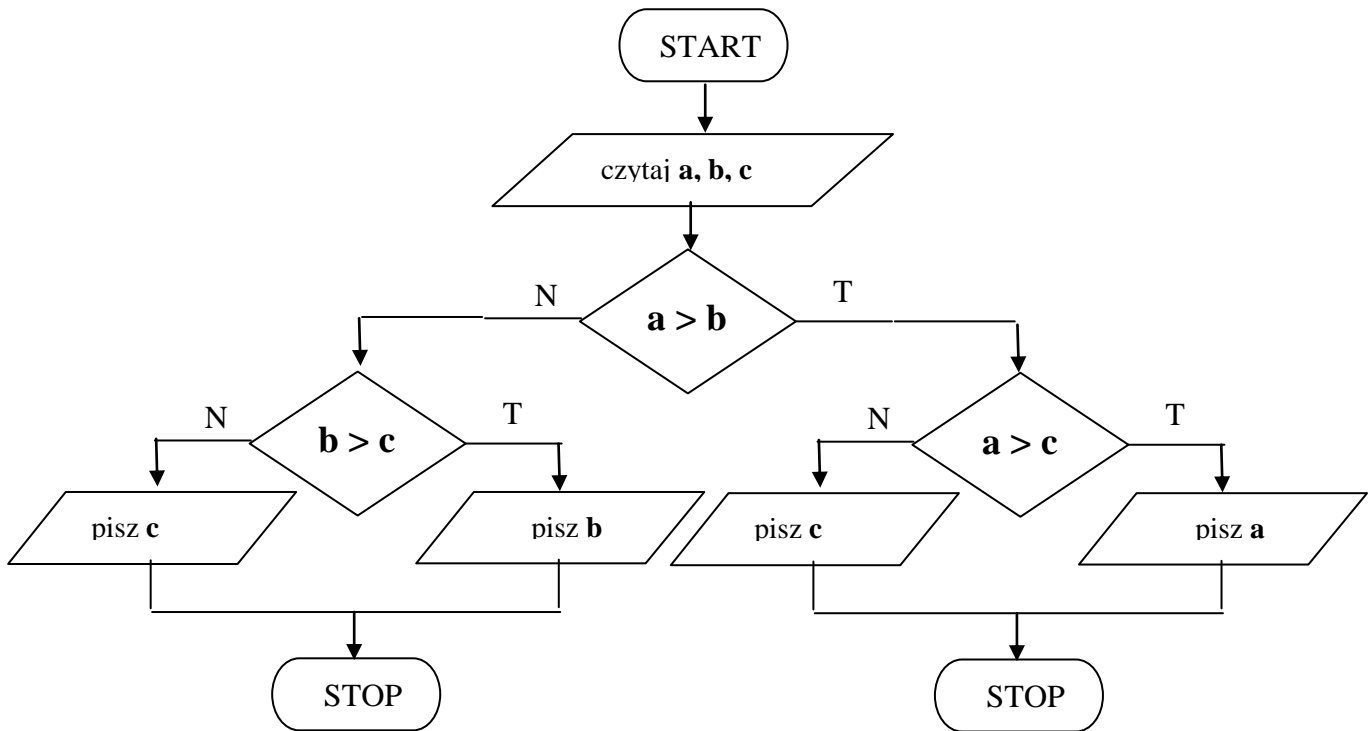


19. Mnożenie określonej liczby (n) dowolnych liczb naturalnych (a)
(iteracja algorytmu z określoną ilością powtórzeń, „for”)

i - licznik (zmienia się od 1 do n)

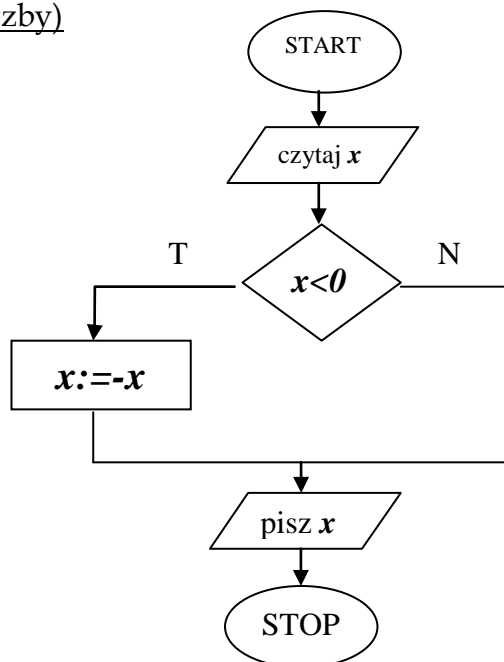


20. Porównywanie trzech liczb (a, b, c - trzy liczby) - wybór największej.



21. Wartość bezwzględna liczby (moduł liczby)

$$|x| = \begin{cases} x, & \text{dla } x \geq 0 \\ -x, & \text{dla } x < 0 \end{cases}$$

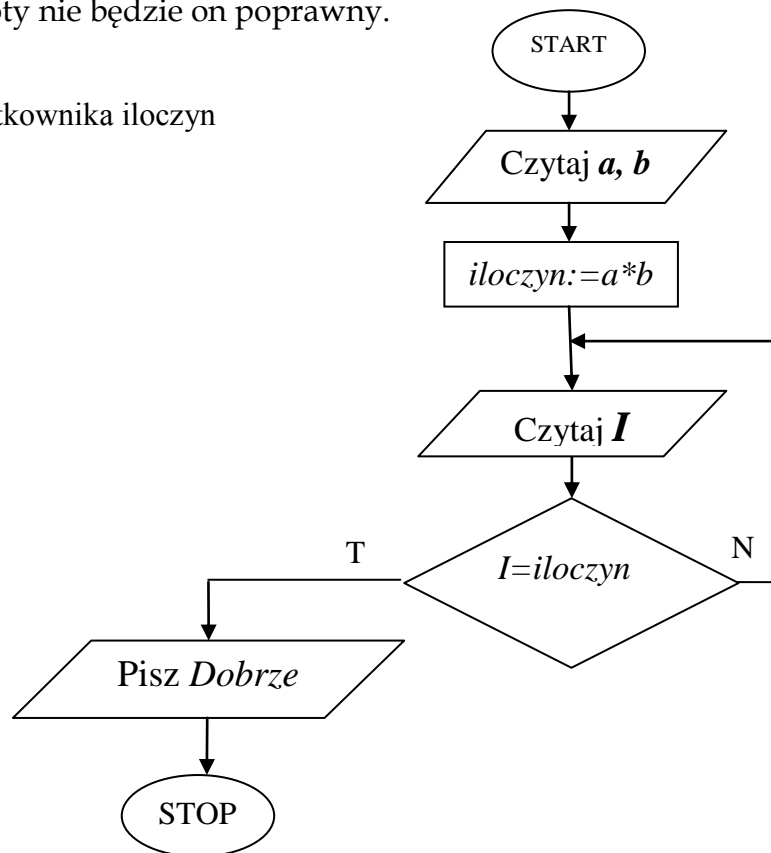


KILKA CIEKAWYCH ZADAŃ 😊

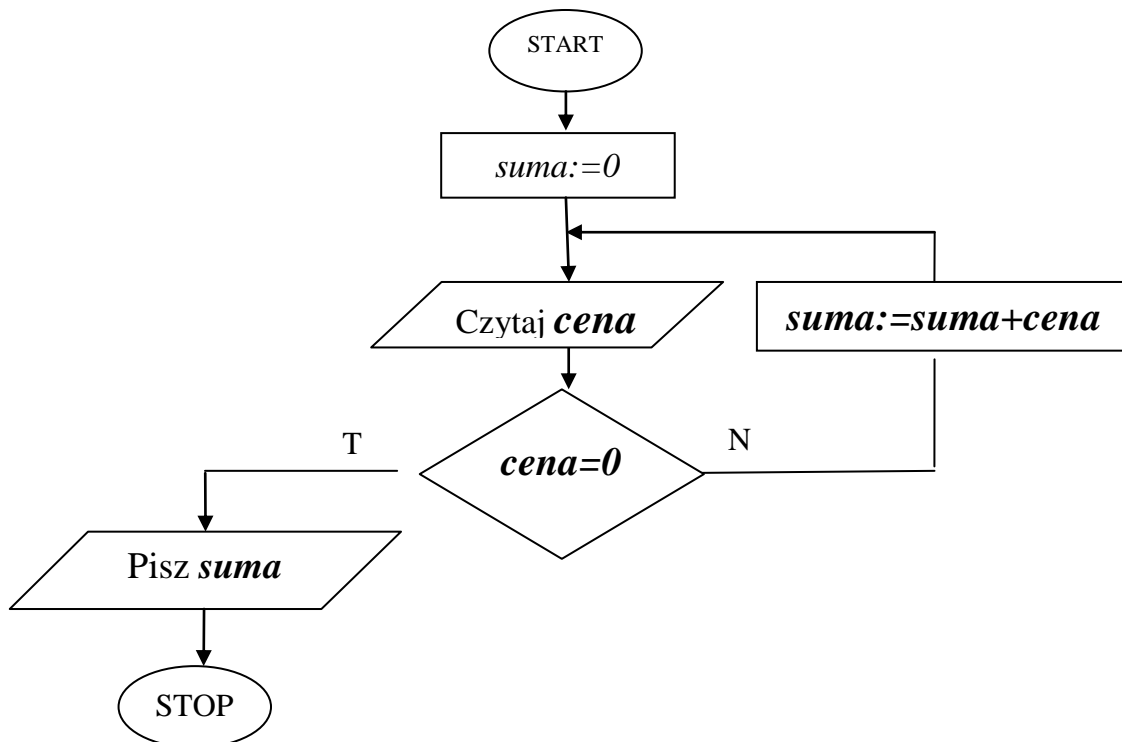
1. **TABLICZKA MNOŻENIA.** Algorytm ma wczytać czynniki mnożenia, czyli dwie dowolne liczby naturalne (a, b), oraz proponowany przez użytkownika wynik tego mnożenia (*iloczyn*). Jeśli wynik będzie poprawny ma pojawić się komunikat - „Dobrze”, a jeśli wynik będzie błędny, to „program” powinien ponowić prośbę o wynik. Wczytywanie proponowanego iloczynu ma odbywać się dopóty nie będzie on poprawny.

a, b - czynniki

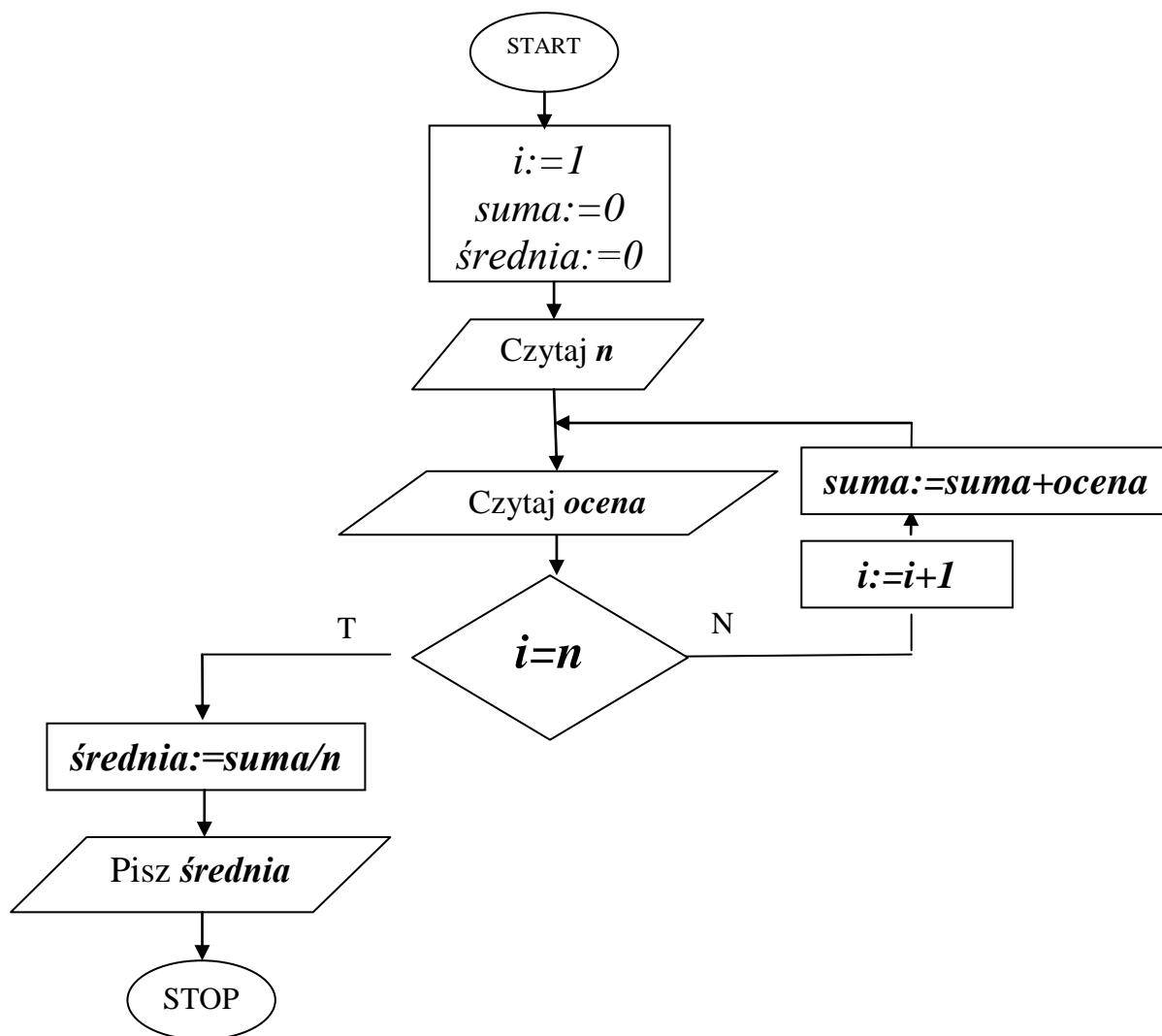
I – proponowany przez użytkownika iloczyn



2. **KASA.** Algorytm ma działać, jak „kasa fiskalna”, czyli ma sumować (w pętli) zakupione towary. Użytkownik podaje *cenę* dowolnej ilości towarów. Obliczanie sumy kończy się, gdy podana zostanie liczba 0. Program wyświetla *sumę* końcową (całkowitą).



3. **ŚREDNIA OCEN.** Algorytm ma za zadanie obliczanie średniej ocen. Użytkownik na początku podaje ilość ocen (n), których średnią będzie chciał policzyć, a następnie podaje kolejne oceny ($ocena$). Wprowadzanie ocen kończy się z chwilą podania przez niego 0. Wtedy pojawia się informacja o średniej ocen danego ucznia ($średnia$).



4. **PIRAMIDA.** Aby zbudować piramidę o podstawie 5 kwadratów jak na rysunku obok, potrzeba 15 elementów. Przygotuj algorytm, który będzie liczył, ile potrzeba kwadratów do zbudowania piramidy o podstawie 10, 15 bądź 20 kwadratów. Przyjmij, że „ n ” – ilość kwadratów w podstawie piramidy – to liczba naturalna z zakresu 1-256, a „ S ” - ilość wszystkich kwadratów potrzebnych do zbudowania piramidy będzie liczbą naturalną.

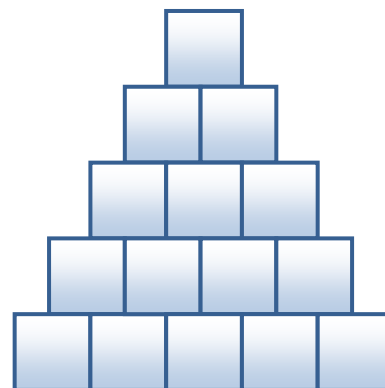
$n \in \mathbb{N} \cap \{1, \dots, 256\}$ - podstawa piramidy
 $s \in \mathbb{N}$ - ilość wszystkich kwadratów

Uwaga!

Znaczenie symbolu Σ


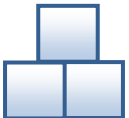
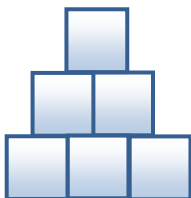
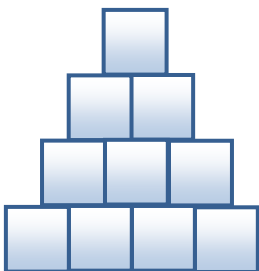
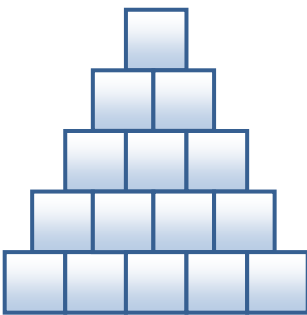
Σ - czyt. sigma - oznacza sumę, np.:

$$\sum_{i=1}^5 i = 1 + 2 + 3 + 4 + 5$$



$$\sum_{i=1}^5 i^2 = 1^2 + 2^2 + 3^2 + 4^2 + 5^2$$

$$\sum_{i=1}^3 (i-1) = (1-1) + (2-1) + (3-1)$$

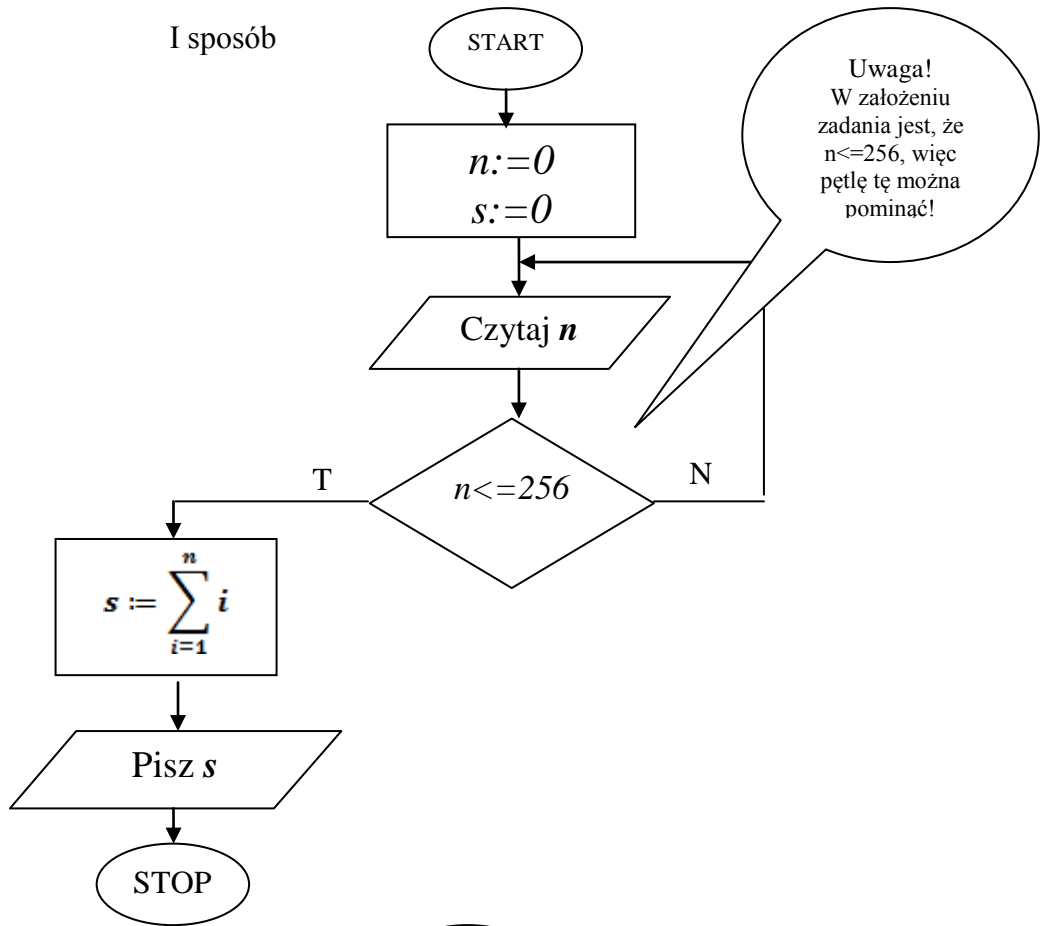
n	Rysunek	s
1		1
2		$3=1+2$
3		$6=1+2+3=3+3$
4		$10=1+2+3+4=6+4$
5		$15=1+2+3+4+5=10+5$
...
		$s_n = s_{n-1} + n$

n

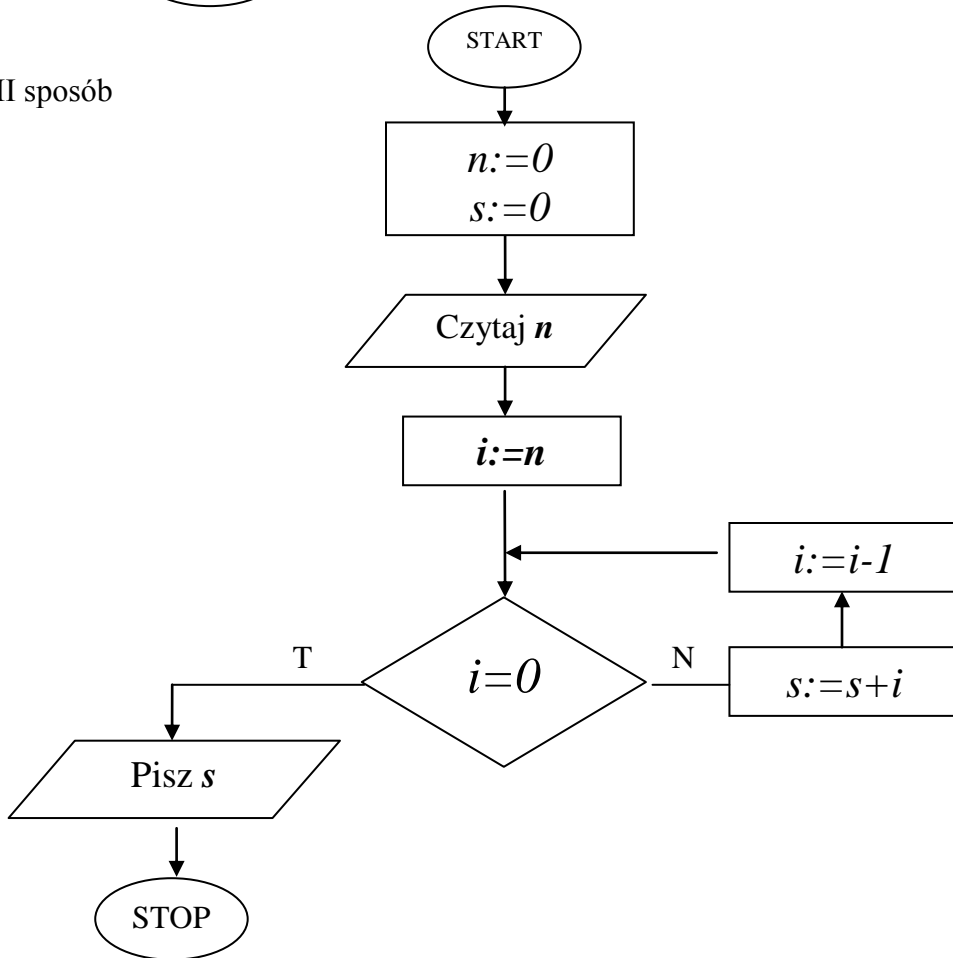
albo

$$s_n = \sum_{i=1}^n i = 1 + 2 + 3 + \dots + n$$

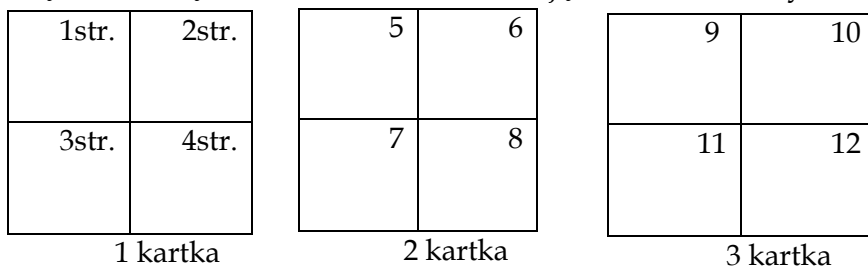
I sposób



II sposób



5. **KARTKI.** Na każdej kartce mieszczą się 4 strony z zadaniami. Napisz algorytm, który po wczytaniu numeru strony wypisze, na której kartce się ona znajduje. Zobacz rysunek. Niech „ n ” – liczbę naturalną z zakresu 1-100 oznaczającą numer strony, „ k ” – numer kartki.

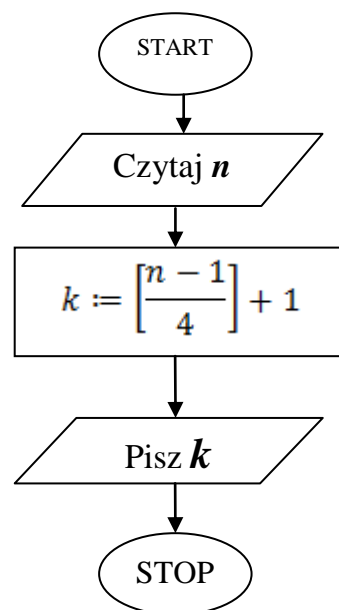


$n \in \mathbb{N} \cap \{1, \dots, 100\}$ – nr strony,
 $k \in \mathbb{N}$ – nr kartki

Uwaga! Znaczenie symbolu $[x]$

$[x]$ – czyt. „cecha” – oznacza największą liczbę całkowitą jej nieprzekraczającą (czyli liczbę całkowitą $\leq x$). Np.: $[4\frac{2}{3}] = 4$, $[-4\frac{2}{3}] = -3$, $[3] = 3$, itd.

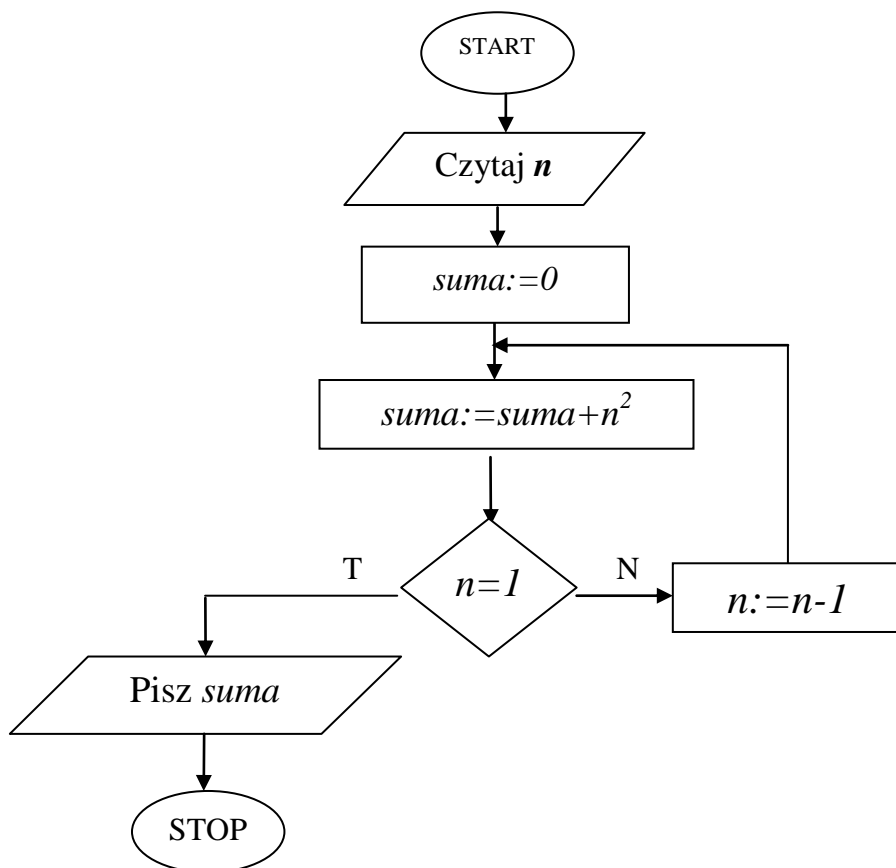
n	k	<i>rysunek</i>				
2	$[\frac{2-1}{4}] + 1 = 0 + 1 = 1$	<table border="1" style="width: 100%; text-align: center;"> <tr><td>1str.</td><td>2str.</td></tr> <tr><td>3str.</td><td>4str.</td></tr> </table>	1str.	2str.	3str.	4str.
1str.	2str.					
3str.	4str.					
3	$[\frac{3-1}{4}] + 1 = 0 + 1 = 1$	<table border="1" style="width: 100%; text-align: center;"> <tr><td>1str.</td><td>2str.</td></tr> <tr><td>3str.</td><td>4str.</td></tr> </table>	1str.	2str.	3str.	4str.
1str.	2str.					
3str.	4str.					
4	$[\frac{4-1}{4}] + 1 = 0 + 1 = 1$					
...						
5	$[\frac{5-1}{4}] + 1 = 1 + 1 = 2$	<table border="1" style="width: 100%; text-align: center;"> <tr><td>5</td><td>6</td></tr> <tr><td>7</td><td>8</td></tr> </table>	5	6	7	8
5	6					
7	8					
...				
8	$[\frac{8-1}{4}] + 1 = 1 + 1 = 2$	<table border="1" style="width: 100%; text-align: center;"> <tr><td>5</td><td>6</td></tr> <tr><td>7</td><td>8</td></tr> </table>	5	6	7	8
5	6					
7	8					
...						
12	$[\frac{12-1}{4}] + 1 = 2 + 1 = 3$	<table border="1" style="width: 100%; text-align: center;"> <tr><td>9</td><td>10</td></tr> <tr><td>11</td><td>12</td></tr> </table>	9	10	11	12
9	10					
11	12					
n	$[\frac{n-1}{4}] + 1$	Itd.				



6. **SUMA POTĘG.** Algorytm ma obliczać sumę n liczb spełniających regułę: 1, 4, 9, 16, 25, 36, ... Niech „ n ” – liczba naturalna z zakresu 1-20 oznaczająca ilość liczb, „ $suma$ ” – suma n liczb (potęg).

$n \in \mathbb{N} \cap \{1, \dots, 20\}$ – ilość kolejnych potęg,

np. Dla $n=5$ mamy $suma=1^2+2^2+3^2+4^2+5^2=1+4+9+16+25=55$.

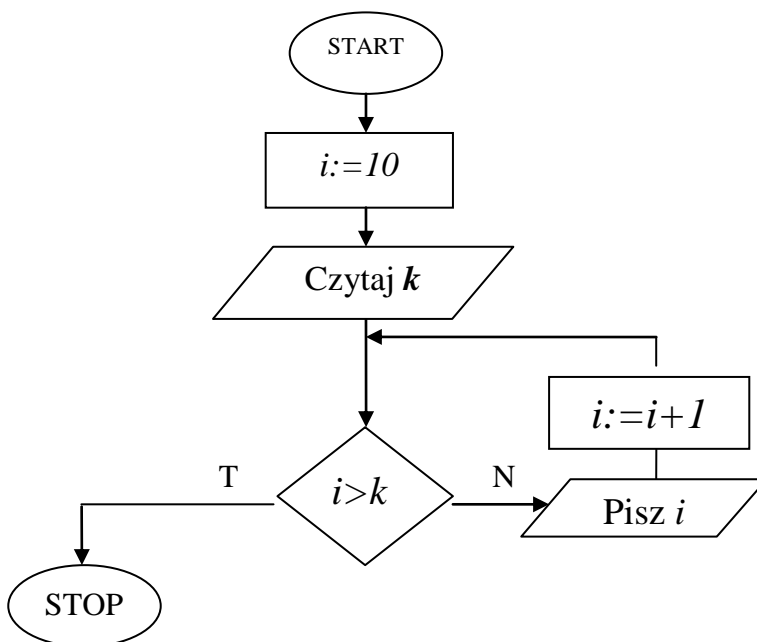


7. **LICZBY DWUCYFROWE.** Dana jest liczba dwucyfrowa k . utwórz algorytm, który wypisze wszystkie liczby dwucyfrowe nie większe niż k w kolejności rosnącej.

$k \in \mathbb{N} \cap \{10, \dots, 99\}$

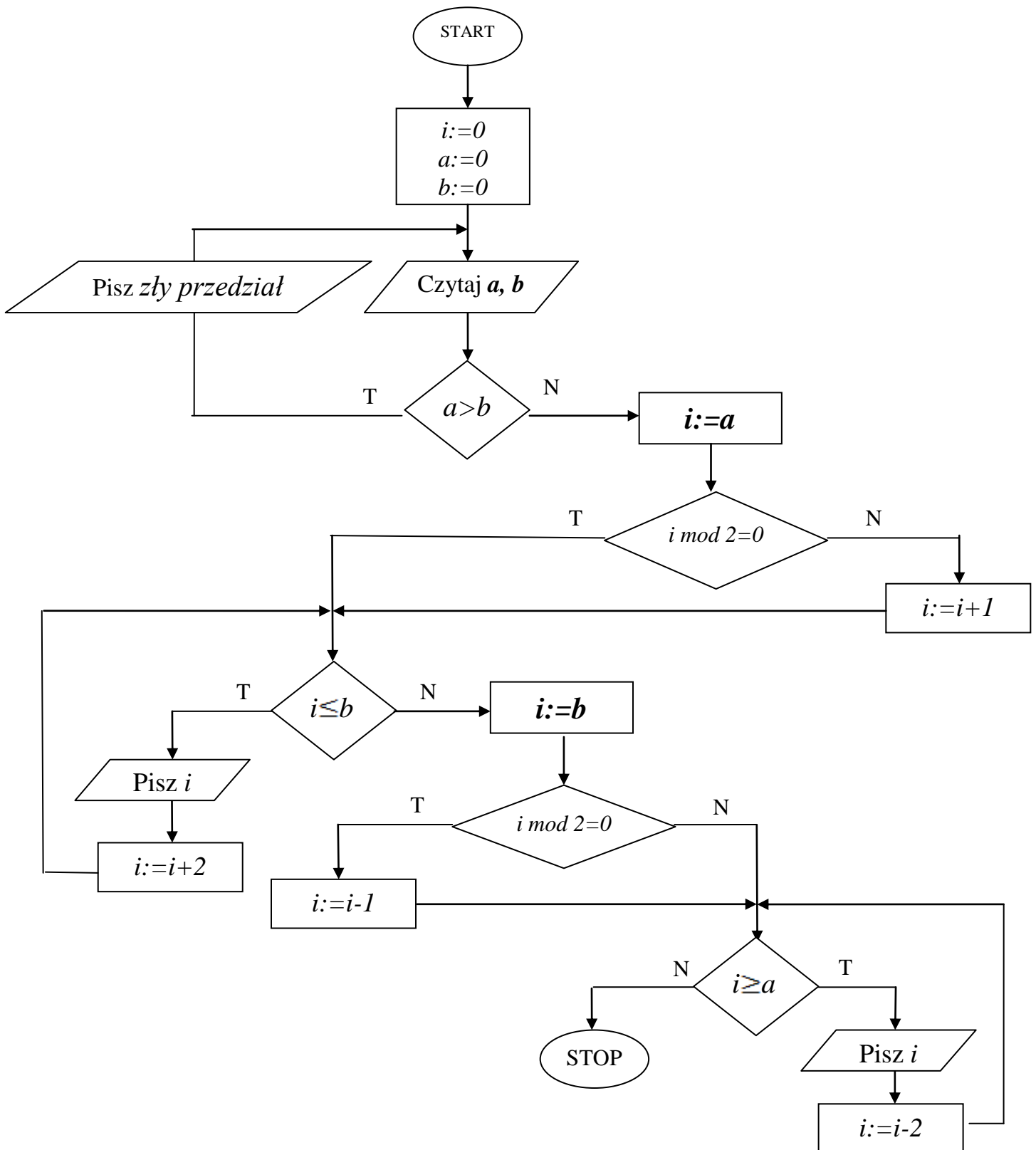
Np.: dla $k=17$

mamy ciąg: 10, 11, 12, 13, 14, 15, 16, 17.



8. **LICZBY.** Użytkownik podaje dwie liczby całkowite a, b . algorytm ma za zadanie wypisać wszystkie parzyste liczby w kolejności rosnącej, a następnie wszystkie liczby nieparzyste w kolejności malejącej z przedziału $\langle a;b \rangle$. niech a, b – liczby całkowite z zakresu 0-255. Np. dla danych wejściowych $a=3, b=8$, otrzymujemy plik wynikowy: 4, 6, 8, 7, 5, 3.

$a, b \in \mathbb{C} \cap \{0, \dots, 255\}$



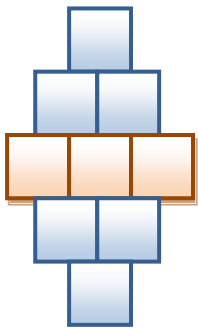
9. **SEKUNDY.** Algorytm ma obliczać ile sekund stanowi G godzin M minut i S sekund. Niech G, M, S – liczby naturalne z zakresu $0 - 60$. W wyniku powinniśmy uzyskać czas w sekundach.

Pamiętamy!!!

$$1\text{h}=60\text{min}=3600\text{s}$$

$$1\text{min}=60\text{s}$$

10. **WRZECIONO.** Przygotuj algorytm, który wyznaczy ilość kwadratów potrzebnych do utworzenia figury podobnej do tej z rysunku poniżej. Niech „ n ” – będzie ilością rzędów wrzeciona (liczba nieparzysta). „ S ” – sumą kwadratów potrzebnych do utworzenia figury. Np. dla $n=5, S=9$ (rysunek obok).



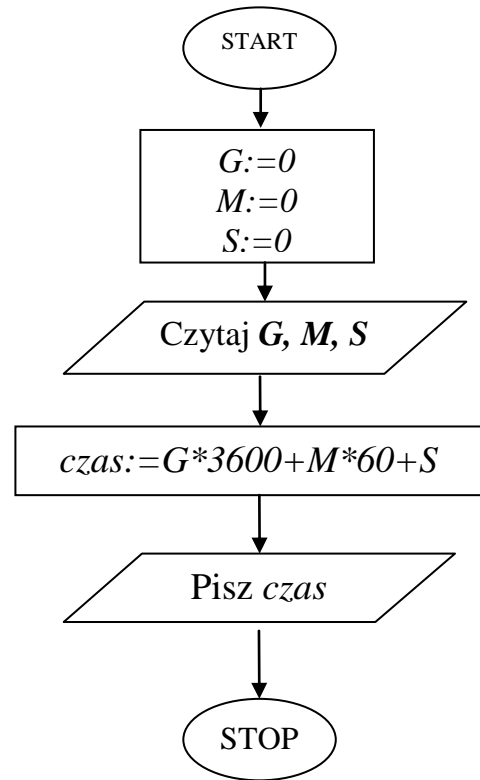
Uwaga!

Tylko dla liczb nieparzystych zachodzi równość:

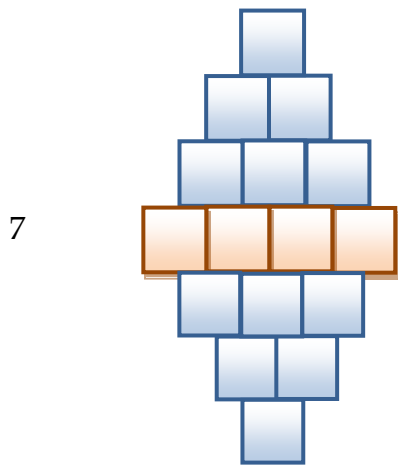
$$\left\lfloor \frac{n}{2} \right\rfloor = \frac{n-1}{2}$$

Zatem x (zmienną pomocniczą) możemy liczyć na dwa sposoby

$$x = \left\lfloor \frac{n}{2} \right\rfloor + 1 = \frac{n-1}{2} + 1$$



n	Rysunek	x	s
1		$\left\lfloor \frac{1}{2} \right\rfloor + 1 = 0 + 1 = 1$	$1=1^2$
3		$\left\lfloor \frac{3}{2} \right\rfloor + 1 = 1 + 1 = 2$	$4=2^2$
5		$\left\lfloor \frac{5}{2} \right\rfloor + 1 = 2 + 1 = 3$	$9=3^2$



7

$$\left\lfloor \frac{7}{2} \right\rfloor + 1 = 3 + 1 = 4$$

$$16 = 4^2$$

9

Itd.

$$\left\lfloor \frac{9}{2} \right\rfloor + 1 = 4 + 1 = 5$$

$$25 = 5^2$$

...

...

...

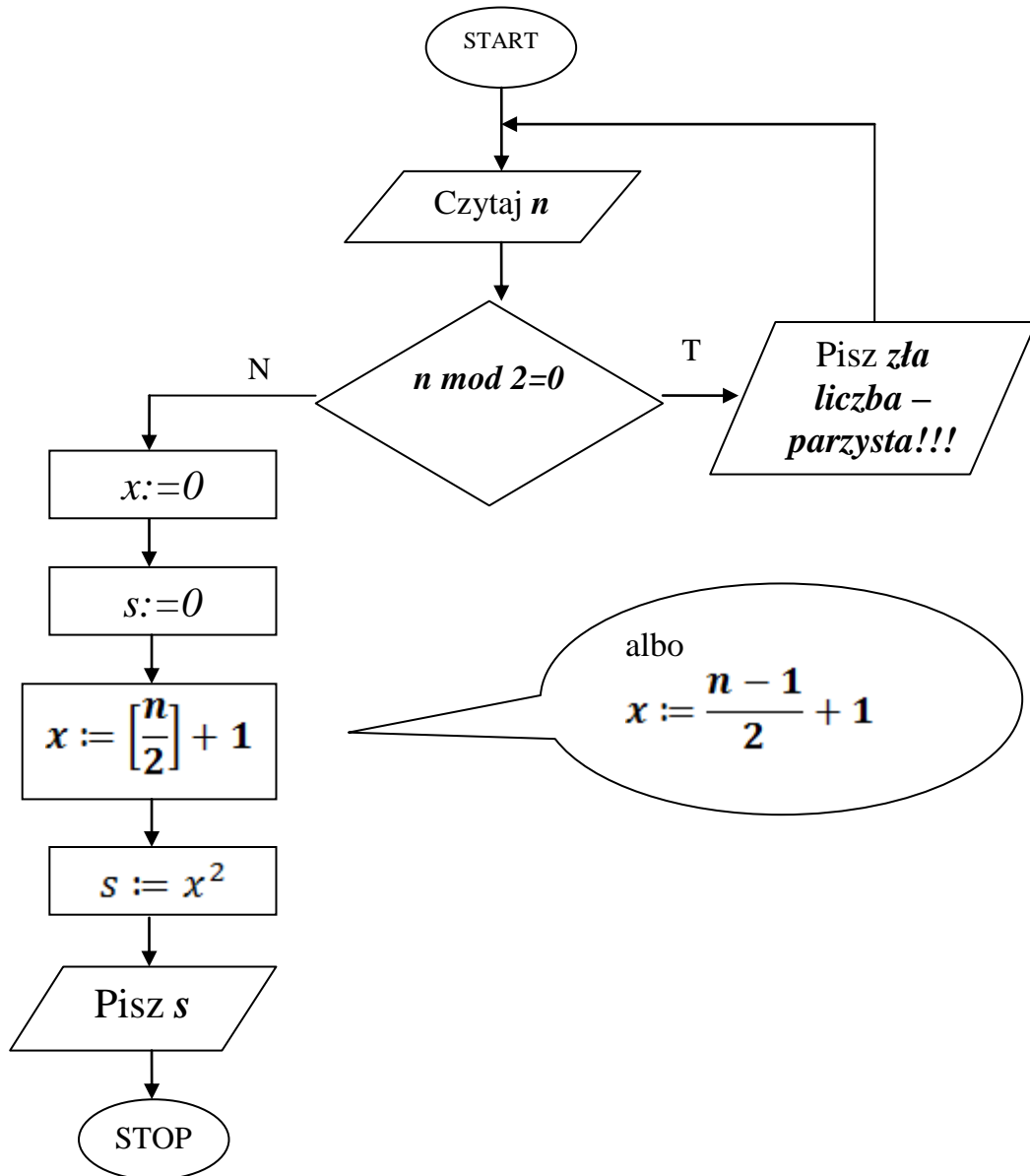
...

n

...

$$x := \left\lfloor \frac{n}{2} \right\rfloor + 1$$

$$s := x^2$$



11. **EGZAMIN.** Pewien uczeń obiecał sobie, że będzie pilnie przygotowywała się do egzaminu t minut każdego dnia. Niestety trudno było mu wytrwać w tym postanowieniu i już następnego dnia czas nauki był o połowę krótszy. Kolejnego dnia czas nauki znowu zmniejszył się o połowę. Sytuacja ta powtarzała się, aż do dnia sprawdzianu. Przygotuj algorytm, który dla podanego czasu nauki pierwszego dnia (t – podany w minutach) i n ilości dni do egzaminu, wyznaczy sumaryczny czas przygotowania się ucznia. Np. dla $t=64$, $n=3$ otrzymujemy odpowiedź: 112 (bo $64+32+16=112$).

t – czas nauki pierwszego dnia (w minutach)

n – ilość dni do egzaminu ($n \in \mathbb{N}$)

czas – całkowity (sumaryczny) czas przygotowywania się do egzaminu.

$$\begin{aligned} \text{czas} &= t + \frac{1}{2}t + \frac{1}{4}t + \frac{1}{8}t + \dots + \frac{1}{2^{n-1}}t = \\ &= t \left(1 + \frac{1}{2} + \frac{1}{2^2} + \frac{1}{2^3} + \dots + \frac{1}{2^{n-1}} \right) = \\ &= t * S_n = t * 1 * \frac{1 - \left(\frac{1}{2}\right)^n}{1 - \frac{1}{2}} = \\ &= t \left(1 - \frac{1}{2^n} \right) * 2 = \\ &= t(1 - 2^{-n}) * 2 = \\ &= (2 - 2^{1-n})t \end{aligned}$$

Czyli $\text{czas} := (2 - 2^{1-n})t$

Np.

dla $n=3$, $t=64$ mamy

$$\text{czas} = (2 - 2^{1-3}) * 64 = \left(2 - \frac{1}{4}\right) * 64 = 128 - 16 = 112$$

Ciąg geometryczny:
 a_1, a_2, a_3, \dots , gdzie
 a_1 - pierwszy wyraz ciągu,
 a_n - n -ty wyraz ciągu, $a_n = a_1 * q^{n-1}$
 q - iloraz, $q = \frac{a_{n+1}}{a_n}$
 S_n - suma n początkowych wyrazów ciągu
 $S_n = a_1 + a_2 + a_3 + \dots + a_n$

$$S_n = \begin{cases} a_1 \frac{1 - q^n}{1 - q} & \text{dla } q \neq 1 \\ a_1 n & \text{dla } q = 1 \end{cases}$$
