Algorytmy – wprowadzenie

Pojęcie słowa „algorytm”

Intuicyjnie:

Algorytm jako *przepis*, *proces*, *metoda*, *technika*, *procedura*.

Przykłady: przepis kucharski, instrukcja składania mebla/urządzenia/, zapis nutowy,

wykonywanie pisemne dodawania/mnożenia/dzielenia

Precyzyjniej:

Algorytm – skończony zbiór reguł wskazujący kolejność operacji dla rozwiązania problemu danego typu.

Sposób postępowania (przepis) umożliwiający rozwiązanie określonego zadania (klasy zadań), podany w postaci skończonego zestawu czynności do wykonania, ze wskazaniem ich następstwa.

Istotne cechy algorytmu

- **Definicja zadania** = co algorytm ma zrobić
- **Opis ciągu czynności**, które po kolei mają być wykonane
- Czynności te muszą być na tyle proste (i możliwe do wykonania), aby wykonawca algorytmu mógł je bez dodatkowego tłumaczenia, wykonać (operacje elementarne, odpowiednio dobrany poziom szczegółowości)
- Skończona liczba operacji elementarnych (**skończony czas działania**)
- Algorytm dostaje pewne informacje (**dane wejściowe**) i zwraca pewne (oczekiwane) wyniki — **dane wyjściowe**
- **Może istnieć kilka przepisów**, które dają w wyniku te same wyniki

Algorytm

- **Pochodzenie nazwy od nazwiska w wersji łacińskiej Algorithmus, Algorismus, perskiego matematyka Muhammeda ibn Musy zwanego al Chuwarismi, żyjącego w IX w; podał on algorytmy wykonywania działań arytmetycznych na liczbach dziesiętnych**
- **Algorytmika – dział wiedzy zajmujący się badaniem algorytmów**
- **Sposoby zapisu algorytmu**
  - **słowami**
  - **za pomocą schematu blokowego**
  - **w pseudokodzie**
  - **w jednym z języków programowania**

Formalnie spisana wersja algorytmu to program

Algorytm – formalnie

Algorytm - ściśle określony ciąg kroków obliczeniowych, prowadzący do przekształcenia danych wejściowych w wyjściowe

Algorytm – formalnie

Cechy dobrego algorytmu

- Skończoność. Wykonanie algorytmu zawsze kończy się po skończonej liczbie kroków.
- Poprawne zdefiniowanie. Każdy krok algorytmu opisany jest precyzyjnie i jednoznacznie.
- Dane wejściowe. Są to wartości znane przed rozpoczęciem działania algorytmu lub dostarczane w czasie jego wykonywania.
- Dane wyjściowe – wynik działania algorytmu. Algorytm generuje dane wyjściowe, powiązane w pewien sposób z danymi wejściowymi.
- Efektywne zdefiniowanie. Operacje algorytmu powinny być jak najprostsze, dające wykonać się w jak najkrótszym możliwym czasie.

Zadanie algorytmiczne

- **Postawienie problemu (specyfikacja zadania algorytmicznego)**
  - **Dane wejściowe — poprawność i zakres danych wejściowych**
  - **Dane wyjściowe (wyniki) — charakterystyka oczekiwanych wyników jako funkcji danych wejściowych**

Np. chcemy zrobić herbatę

-dysponujemy wodą, herbata, cukrem, oraz czajnikiem, łyżeczką i kubkiem.

-oczekujemy, że powstanie odpowiednio słodka herbata

Zadanie algorytmiczne c.d.

- **Celem zadania algorytmicznego jest znalezienie algorytmu przekształcającego dane wejściowe w wyjściowe, zgodnie z zadanymi założeniami**

Np. Kombinujemy jak z dostępnych narzędzi i środków (dane wejściowe) zrobić pyszny napar z herbaty. Staramy się wykonać ciąg czynności które doprowadzą do realizacji postawionego zadania.

Woda do czajnika -> gotujemy wodę -> wsypujemy herbate do kubka

->zalewamy wrzątkiem -> dodajemy cukier -> mieszamy -> w razie potrzeby dosypujemy cukru…

Zadanie algorytmiczne c.d.

- **Algorytm = rozwiązanie zadania algorytmicznego**
  Algorytm powinien działać dla dowolnego zestawu danych ze zbioru poprawnych danych wejściowych

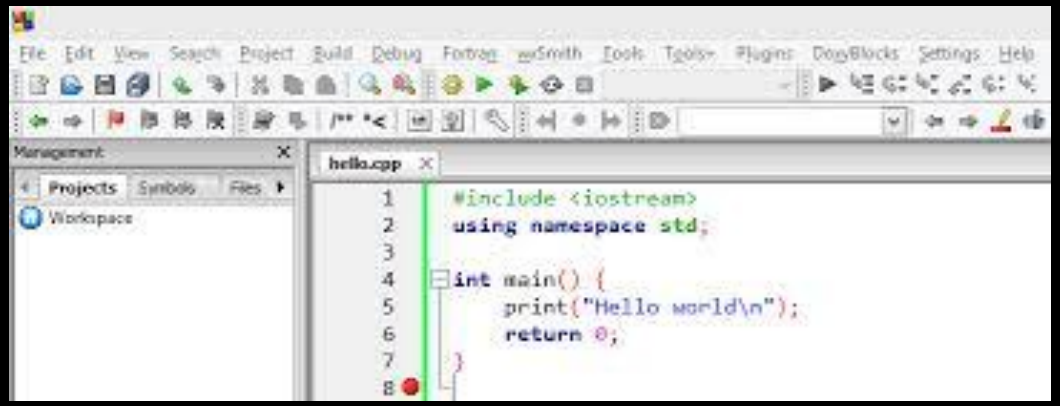
Np. Nie ma znaczenia, że dysponowaliśmy na wejściu kilogramem cukru i herbaty -> dobieramy taką ilość produktów by herbata spełniła określone wymagania. Temperatura początkowa wody również nie miała znaczenia -> należało ją zagotować i dopiero użyć do sporządzenia naparu.

Wniosek -> ważne były odpowiednie kryteria i warunki…

Program komputerowy

Pojęcie nierozerwalnie związane z tematem algorytmów…

To ciąg instrukcji języka programowania (zrozumiały dla komputera) realizujący zadany algorytm.



Język programowania

Zbiór zasad określających, kiedy ciąg symboli tworzy program komputerowy oraz jakie obliczenia opisuje.

Lista kilku najpopularniejszych języków programowania:

Java	C/C++	C#	Python
Visual Basic .NET	PHP		JavaScript

Relacja problem – algorytm – program

1 - INFORMATYK

Problem

Układa

lub wybiera

Algorytm

zapisuje

realizuje

Program

wykonuje

Rozwiązywanie problemu (zagadnienia)

- Sformułowanie zadania
- Określenie danych wejściowych
- Określenie celu, czyli wyniku
- Określenie metody rozwiązania (wybór algorytmu)
- Przedstawienie algorytmu (w wybranej postaci):
  - Opis słowny
  - Lista kroków
  - Schemat blokowy
  - Program w wybranym języku programowania
- Analiza poprawności
- Testowanie rozwiązania dla różnych danych – ocena efektywności

Specyfikacja problemu

Specyfikacja problemu to szczegółowy opis zadania, w którym wymieniamy dane wejściowe oraz wyniki i związek jaki zachodzi między nimi (warunki jakie muszą być spełnione).

Specyfikacja problemu – przykład

Zadanie 1. Uporządkuj rosnąco zbiór nazwisk uczniów klas pierwszych.

Dane: nieuporządkowany zbiór nazwisk

Wynik: uporządkowany alfabetycznie rosnąco zbiór nazwisk

Zadanie 2. Obliczyć sumę dwóch liczb całkowitych.

Dane: dwie dowolne liczby całkowite A i B

Wynik: wartość W będąca sumą liczb A i B

Specyfikacja problemu – ćwiczenie

Napisz specyfikacje do zadań:

Zadanie 1. Zebrano daje o wzroście uczniów klas pierwszych. Uporządkuj informacje o wzroście malejąco.

Specyfikacja problemu – ćwiczenie

Rozwiązanie:

Zadanie 1. Zebrano daje o wzroście uczniów klas pierwszych. Uporządkuj informacje o wzroście malejąco.

DANE: nieuporządkowany zbiór danych o wzroście uczniów klas pierwszych

WYNIK: uporządkowany zbiór danych (malejąco) o wzroście uczniów

Specyfikacja problemu – ćwiczenie

Napisz specyfikacje do zadań:

Zadanie 2. Znajdź wśród danych uczniów twojej klasy liczbę określającą najniższego i największego ucznia.

Specyfikacja problemu – ćwiczenie

Rozwiązanie:

Zadanie 1. Znajdź wśród danych uczniów twojej klasy liczbę określającą najniższego i największego ucznia.

DANE: nieuporządkowany zbiór danych o wzroście uczniów klasy

WYNIK: wartość minimalna i maksymalna określająca ucznia najniższego i najwyższego

Sposoby przedstawiania algorytmów

Lista kroków – przedstawienie algorytmu w kolejnych punktach (krokach) za pomocą słów (języka naturalnego).

Każdy punkt takiej listy zawiera opis wykonywanej czynności.

Kolejność punktów nie może być przypadkowa – musi być zgodna z działaniem algorytmu. Kolejność wykonywania poleceń jest bardzo istotna i należy ją precyzyjnie określić.

Uwaga: Przebieg algorytmu nie zawsze musi odpowiadać kolejności kroków – może się zdarzyć, że polecenie będzie określało konieczność przejścia do innego niż kolejny krok.

Przykład 1: chcemy przedstawić algorytm dzielenia liczb całkowitych, jedna przez drugą.

Dane: dowolne liczby całkowite A i B.

Wynik: wartość dzielenia liczby A przez B.

Lista kroków:

1. Zaczynamy algorytm
2. Wprowadź wartość liczby A i B
3. Oblicz wartość wyrażenia W:= A / B
4. Wyprowadź wynik W
5. Zakończ algorytm

Uwaga: symbol := oznacza przypisanie wartości

Ćwiczenie: **Napisz w postaci listy kroków algorytm obliczania sumy dwóch liczb całkowitych.**

Rozwiązanie: Napisz w postaci listy kroków algorytm obliczania sumy dwóch liczb całkowitych.

Dane: dowolne liczby całkowite A i B.

Wynik: wartość sumy liczby A i B.

Lista kroków:

1.Zaczynamy algorytm

2.Wprowadź wartość liczby A i B

3.Oblicz wartość wyrażenia W:= A + B

4.Wyprowadź wynik W

5.Zakończ algorytm

Operacje warunkowe

Bywa, że w algorytmie zachodzi potrzeba wykonania pewnych czynności w wypadku zaistnienia pewnych okoliczności.

Np. Jeśli słodzimy herbatę to wsypujemy określoną ilość cukru…

…jeśli nie – nie robimy tego.

Miejsce w którym następuje podjęcie określonej decyzji o podjęciu stosownych kroków w oparciu o zdefiniowane kryteria to

OPERACJA WARUNKOWA.

(instrukcja warunkowa/blok warunkowy)

Operacje warunkowe

Czyli…

wykonanie pewnych kroków/czynności

w przypadku zaistnienia określonego warunku/warunków.

Ważne by WARUNEK był jednoznaczny – tzn. dawał jednoznaczną odpowiedź

TAK lub NIE

Operacje warunkowe

Pytanie: Dlaczego są tak ważne?

Odpowiedź: Bez nich program nie byłby w stanie adaptować się do zmieniających warunków i realizować zadań zgodnie z określonymi kryteriami.

Przykład: Albo słodzimy herbatę wszystkim po równo (bez względu na to czy tego chcą czy nie), albo nie robimy tego wcale…

Operacje warunkowe

Jak zatem działają operacje warunkowe?

…

2.Instrukcja kroku drugiego

„3.Jeśli ZACHODZI WARUNEK idź do kroku 6”

4.Instrukcja kroku czwartego

…

czyli jeśli powyższy warunek jest spełniony wówczas idź do kroku 6, w przeciwnym razie idź dalej, czyli do kroku 4.

Operacje warunkowe

Jak definiować warunki?

W zasadzie niezbędna wiedza to ta wyniesiona z matematyki. Obowiązują te same zasady co w matematyce/logice i te same operatory porównania:

A = B - A równe B

A > B - A większe od B

A < B - A mniejsze od B

A >= B - A większe lub równe B

A <= B - A mniejsze lub równe B

A <> B - A różne od B

Operacje warunkowe

Kiedy warunek jest spełniony…?

Przykład:

1.Rozpocznij algorytm

2.Wczytaj X

3.Jeśli X > 0 idź do kroku 5

4.Wypisz X

5.Zakończ algorytm

Wczytana zostaje wartość X

Jeśli X jest większe od 0 to program

się zakończy bez wypisywanie X.

W przeciwnym razie X zostanie wypisane.

To czy warunek będzie spełniony czy nie zależy od tego jaka wartość X wczytano: Np. dla wartości 3 lub 123 warunek jest prawdziwy ale dla wartości -111 lub 0 warunek nie jest spełniony.

Przykład 2: chcemy przedstawić algorytm dzielenia liczb całkowitych, jedna przez drugą. UWAGA – nie dzielimy przez zero.

Dane: dowolne liczby całkowite A i B.

Wynik: wartość dzielenia liczby A przez B, chyba, że B równa jest zero wówczas nie dzielimy i brak wyniku.

Lista kroków:

1. Zaczynamy algorytm
2. Wprowadź wartość liczby A i B
3. Jeśli B = 0 idź do kroku 6
4. Oblicz wartość wyrażenia W:= A / B
5. Wyprowadź wynik W
6. Zakończ algorytm

Przykład 2.1: chcemy przedstawić algorytm który informuje nas o tym czy wczytana dowolna wartość liczby całkowitej jest większa od zera.

Dane: dowolna liczba całkowita A.

Wynik: Informacja czy A jest większe od 0 czy nie.

Lista kroków:

1. Zaczynamy algorytm
2. Wprowadź wartość liczby A
3. Jeśli A = 0 idź do kroku 5
4. Wypisz „A jest większe od 0” i idź do kroku 6.
5. Wypisz „A nie jest większe od 0”
6. Zakończ algorytm

Algorytm ma błąd…?

Ćwiczenie: **chcemy przedstawić algorytm który informuje nas o tym czy wczytana dowolna wartość liczby całkowitej jest większa od zera.**

Popraw błąd z przykładu 2.1

Rozwiązanie: chcemy przedstawić algorytm który informuje nas o tym czy wczytana dowolna wartość liczby całkowitej jest większa od zera.

Dane: dowolna liczba całkowita A.

Wynik: Informacja czy A jest większe od 0 czy nie.

Lista kroków:

1. Zaczynamy algorytm
2. Wprowadź wartość liczby A
3. Jeśli A <= 0 idź do kroku 5
4. Wypisz „A jest większe od 0” i idź do kroku 6.
5. Wypisz „A nie jest większe od 0”
6. Zakończ algorytm

Ćwiczenie: **Napisz w postaci listy kroków algorytm znajdujący wartość MAX spośród dwóch liczb całkowitych A i B.**

Uwaga: nie trzeba sprawdzać czy A=B – jeśli tak jest to prawdą będzie że A jest Max jak również że B jest Max.

Rozwiązanie: Napisz w postaci listy kroków algorytm znajdujący wartość MAX spośród dwóch liczb całkowitych A i B.

Dane: dowolne liczby całkowite A i B.

Wynik: wartość MAX określająca wartość większej liczby spośród A i B.

Lista kroków:

1.Zaczynamy algorytm

2.Wprowadź wartość liczby A i B

3.Jeśli A < B idź do kroku 6

4.MAX := A;

5.Idź do kroku 7

6.MAX := B

7.Wyprowadź wynik MAX

8.Zakończ algorytm

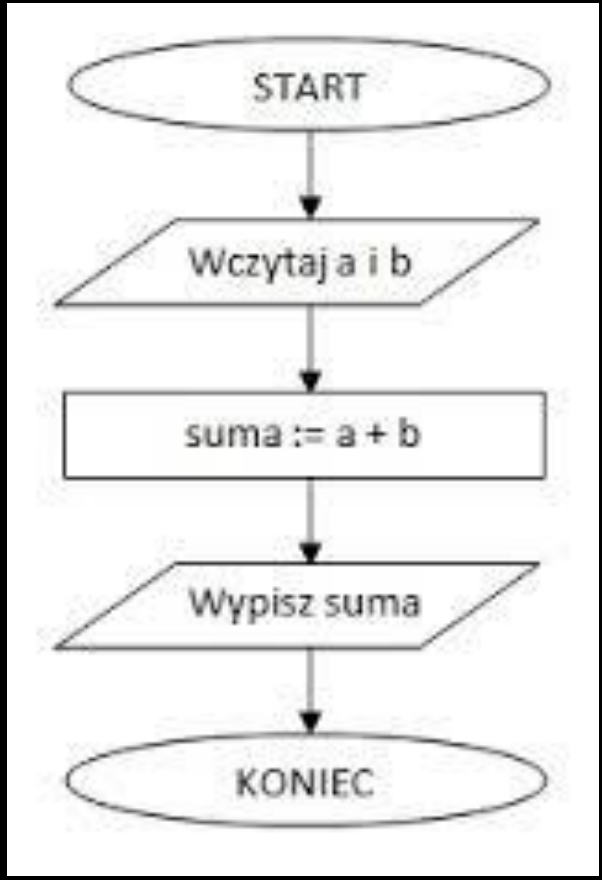
Zadanie domowe: Napisz w postaci listy kroków algorytm znajdujący wartość MAX spośród dwóch liczb całkowitych A i B. Jeśli liczby A i B są równe należy wypisać komunikat „wartości równe”.

Zadanie domowe 2: Napisz w postaci listy kroków algorytm znajdujący wartość MAX spośród trzech liczb całkowitych A, B i C. Jeśli liczby A, B i C są równe należy wypisać komunikat „wartości równe”.

Schemat blokowy

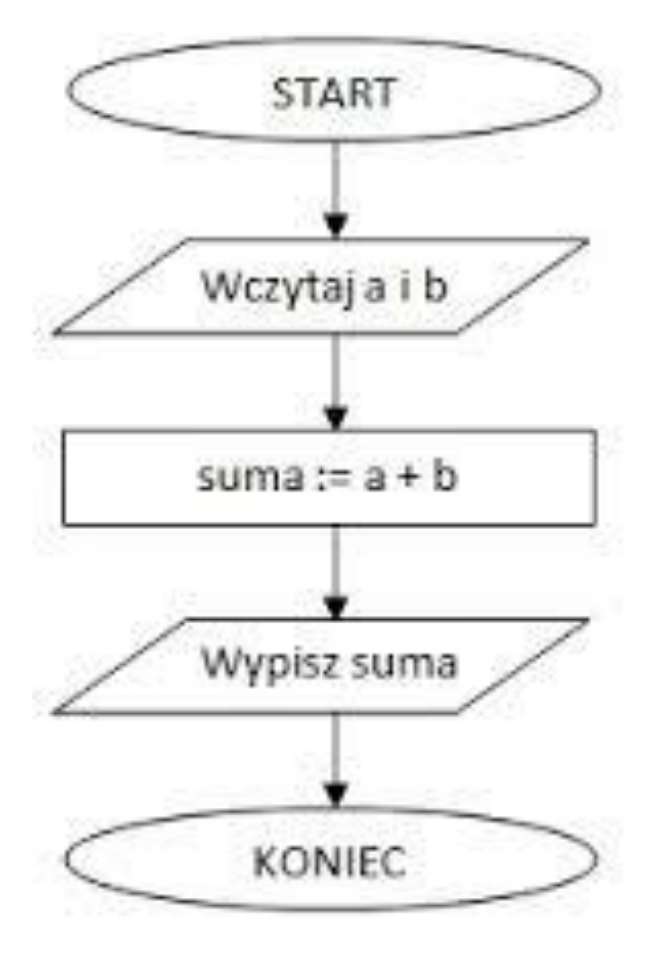
Schemat blokowy

Schemat blokowy przedstawia kolejne operacje za pomocą połączonych symboli (figur - bloków). Kształt i opis bloku określa rodzaj wykonywanej operacji. Kolejność wykonywania operacji wyznaczają połączenia między blokami.



Schemat blokowy vs Lista kroków

Napisz algorytm obliczający wartość sumy dwóch liczb całkowitych A i B.



Dane: dowolne liczby całkowite A i B.

Wynik: wartość MAX określająca wartość większej liczby spośród A i B.

Lista kroków:

1.Zaczynamy algorytm

2.Wprowadź wartość liczby A i B

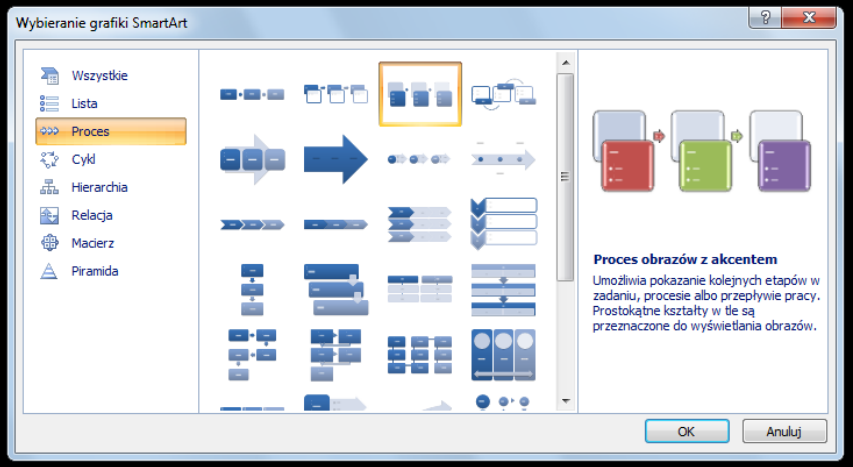
3.Suma := A + B

4.Wyprowadź Suma

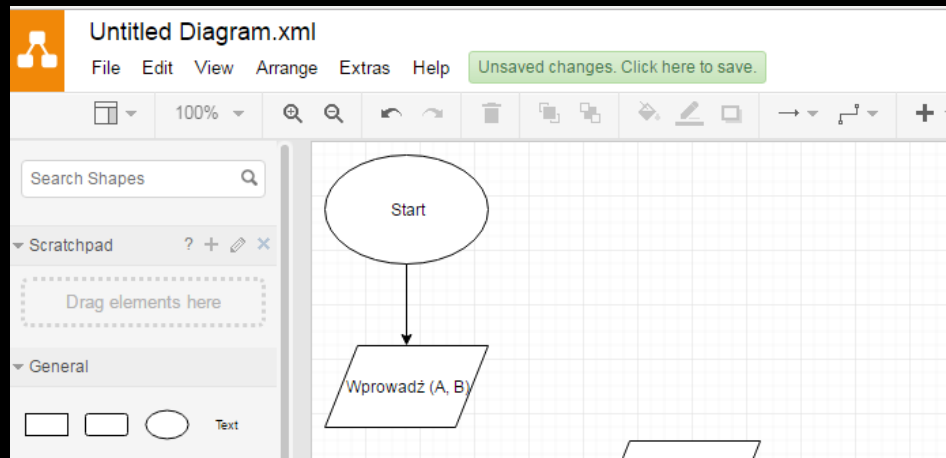
5.Zakończ algorytm

Schemat blokowy – bloki

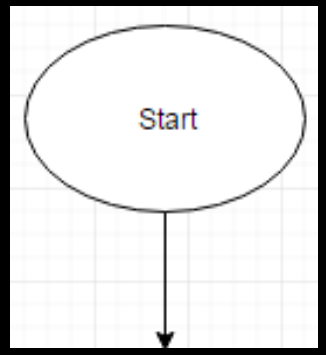
Do tworzenia schematów blokowych można wykorzystać program MS Word (a w nim moduł smartART) lub jeden z wielu dostępnych na rynku edytorów schematów blokowych (w tym te w wersji online: www.draw.io)



lub np. www.draw.io

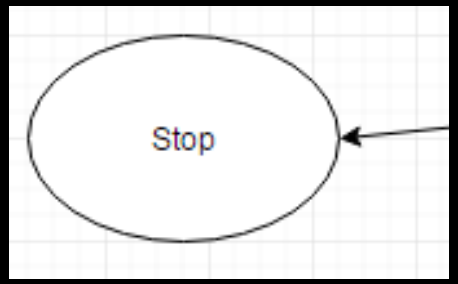


Schemat blokowy – bloki



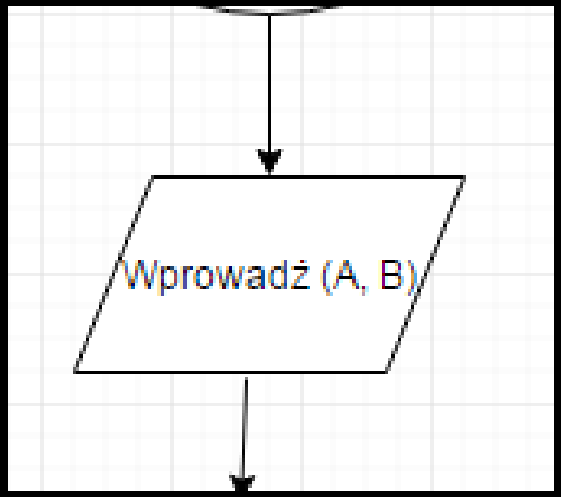
Początek algorytmu – wychodzi z niego tylko jedno połączenie i żadne do niego nie wchodzi. W każdym schemacie może być tylko jeden taki blok

Schemat blokowy – bloki



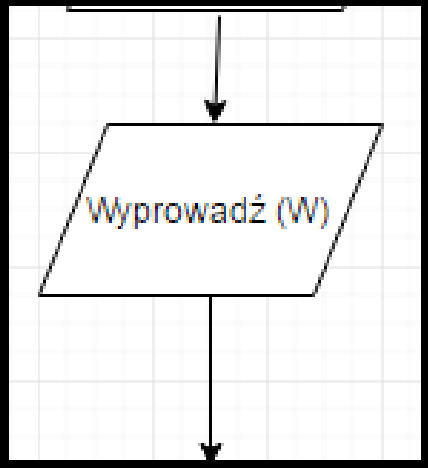
Zakończenie algorytmu – Wchodzi do niego jedno połączenie i żadne nie wychodzi. W jednym schemacie może być wiele takich bloków.

Schemat blokowy – bloki



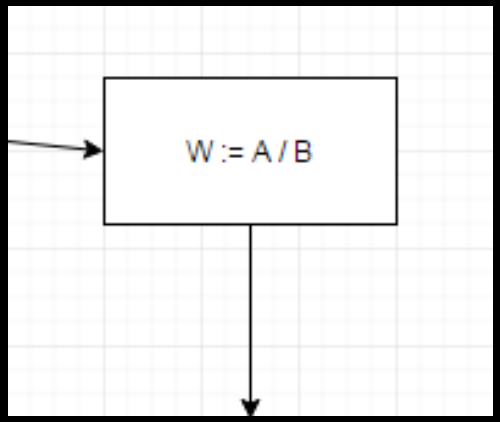
Wprowadzanie danych (blok wejścia) – służy do wprowadzania danych wejściowych. Ma jedno połączenie wchodzące i jedno wychodzące. Może być wiele bloków wejścia w jednym diagramie.

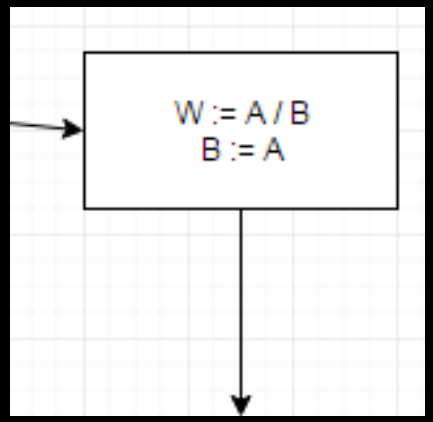
Schemat blokowy – bloki



Wyprowadzenie danych (blok wyjścia) – służy do wyprowadzania danych wyjściowych. Ma jedno połączenie wchodzące i jedno wychodzące. Może być wiele bloków wejścia w jednym diagramie.

Schemat blokowy – bloki

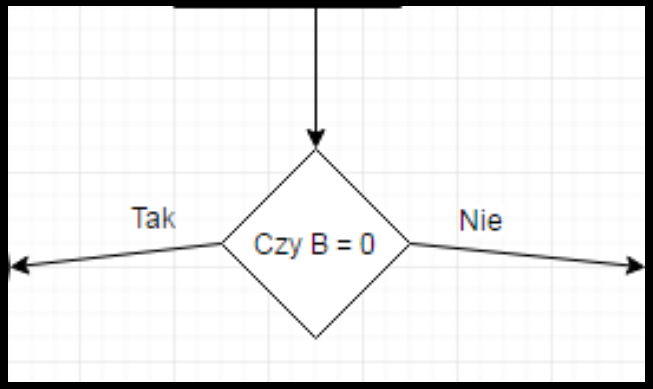




Wykonywanie działań (blok operacyjny) – służy do wykonywania różnych działań (obliczeń etc). Ma jedno połączenie wchodzące i jedno wychodzące. Może być wiele bloków wejścia w jednym diagramie.

W jednym bloku może być więcej niż jedno działanie.

Schemat blokowy – bloki



Sprawdzenie warunku (blok warunkowy lub decyzyjny) – służy do podejmowania decyzji. Ma jedno połączenie wchodzące i dwa wychodzące.

Jedno połączenie wychodzące opisane jako „TAK” określa, co zrobić gdy warunek jest spełniony.

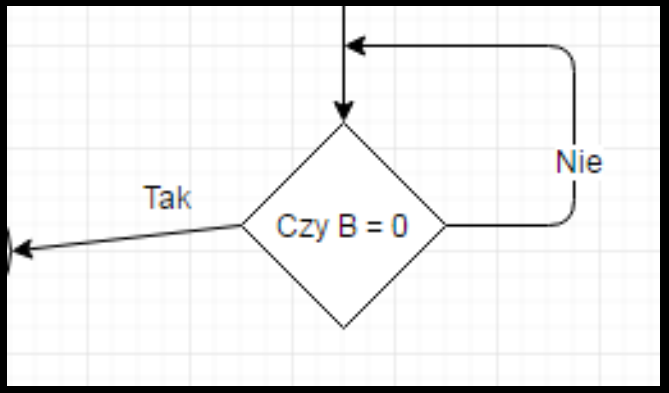
Jedno połączenie wychodzące opisane jako „NIE” określa, co zrobić gdy warunek nie jest spełniony.

Może być wiele bloków wejścia w jednym diagramie.

Schemat blokowy – bloki

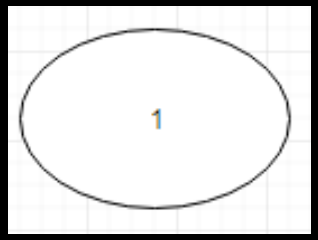
Połączenie – służy do łączenia poszczególnych bloków.

Połączenie może dochodzić do innego połączenia:



Połączeń w jednym schemacie może być wiele.

Schemat blokowy – bloki

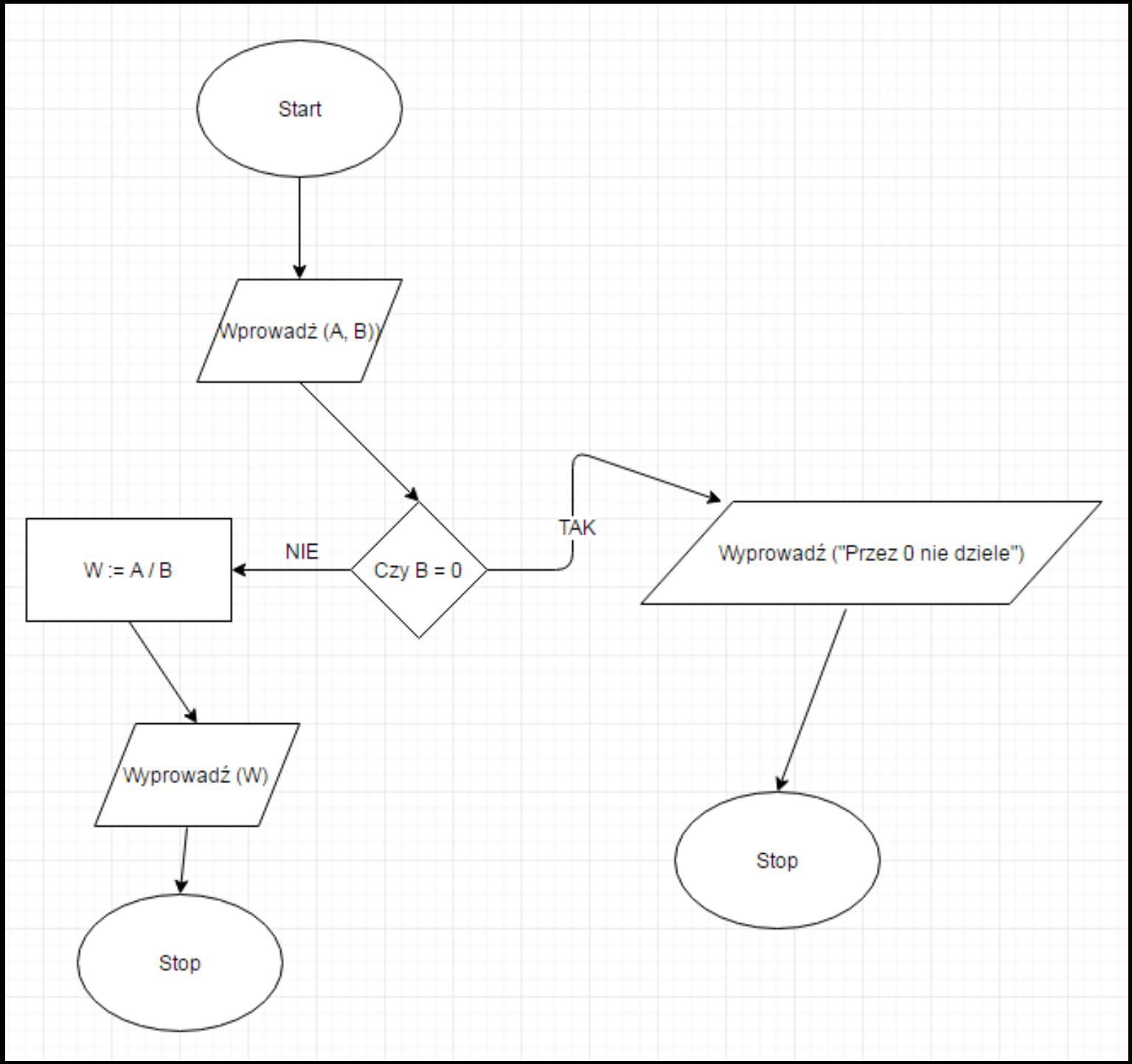


Łącznik – stosuje się go gdy schemat jest tak duży, że trzeba go przedstawić na więcej niż jednej stronie

UWAGA: umieszczony w środku numer powinien być ten sam na obu łączonych częściach.

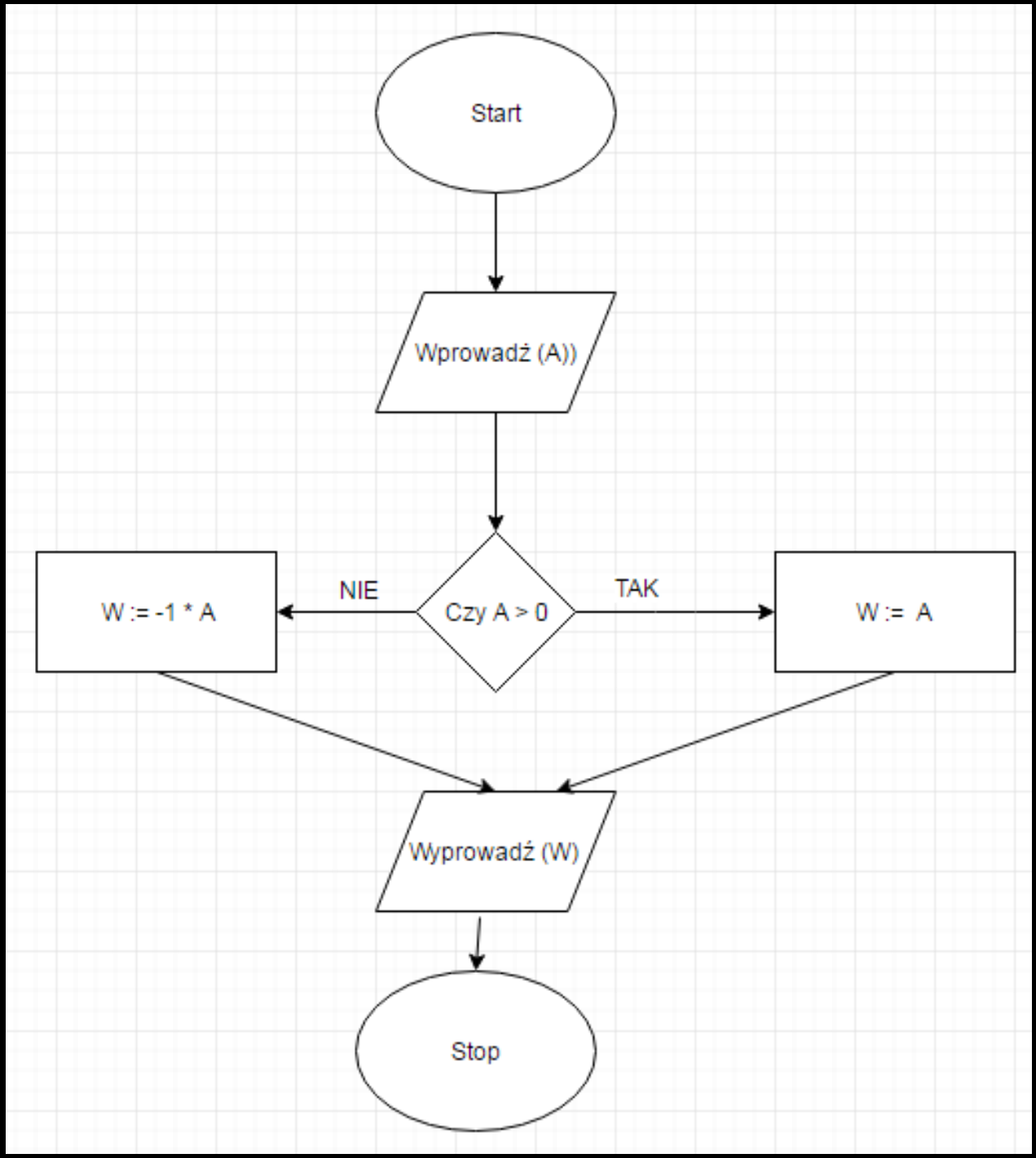
Przykład:

Co przedstawia?



Przykład 2:

Obliczanie wartości bezwzględnej liczby A



Zadanie: narysuj schemat blokowy dla algorytmu realizującego (obliczającego) następującą obliczenie: A*B-C

A, B i C to dowolne liczby rzeczywiste.

Zadanie: narysuj schemat blokowy dla algorytmu opisanego poniżej za pomocą listy kroków.

Algorytm znajdujący wartość MAX spośród dwóch liczb całkowitych A i B.

Dane: dowolne liczby całkowite A i B.

Wynik: wartość MAX określająca wartość większej liczby spośród A i B.

Lista kroków:

1.Zaczynamy algorytm

2.Wprowadź wartość liczby A i B

3.Jeśli A < B idź do kroku 6

4.MAX := A;

5.Idź do kroku 7

6.MAX := B

7.Wyprowadź wynik MAX

8.Zakończ algorytm

Zadanie: **Narysuj schemat blokowy wypisujący (wyznaczający) wartość MIN (najmniejszą) spośród 3 wprowadzonych dowolnych wartości liczb całkowitych A, B i C.**

Zadanie: **Narysuj schemat blokowy wypisujący (wyznaczający) wartość (bezwzględną) |A – B|.**

A i B to dowolne liczby rzeczywiste

Zadanie domowe: **Narysuj schemat blokowy obliczający pole powierzchni prostokąta dla podanych długości boków A i B. Uwaga: należy sprawdzić, czy dane wejściowe nie są błędne (muszą być większe od zera). Jeśli któraś z danych wejściowych jest błędna należy wypisać komunikat „błędne dane”.**

Zadanie domowe: Narysuj schemat blokowy przedstawiający opisany poniżej algorytm:

Algorytm rozwiązywania równania liniowego AX+B=0.

Dane: dowolne liczby rzeczywiste A i B

Wynik: wartość X

Lista kroków:

1.Zacznij algorytm

2.Wprowadź wartości A i B

3.Jeśli A=0 to idź do kroku 6

4.X := -B/A

5.Wypisz X i idź do kroku 9

6.Jeśli B=0 to idź do kroku 8

7.Wyprowadź napis „równanie sprzeczne” i idź do kroku 9

8. Wyprowadź napis „nieskończenie wiele rozwiązań” i idź do kroku 9

9.Zakończ algorytm

Algorytmy iteracyjne

Algorytm iteracyjny - algorytm, który uzyskuje wynik przez iterację, czyli powtarzanie danej operacji początkowo określoną liczbę razy lub aż do spełnienia określonego warunku.

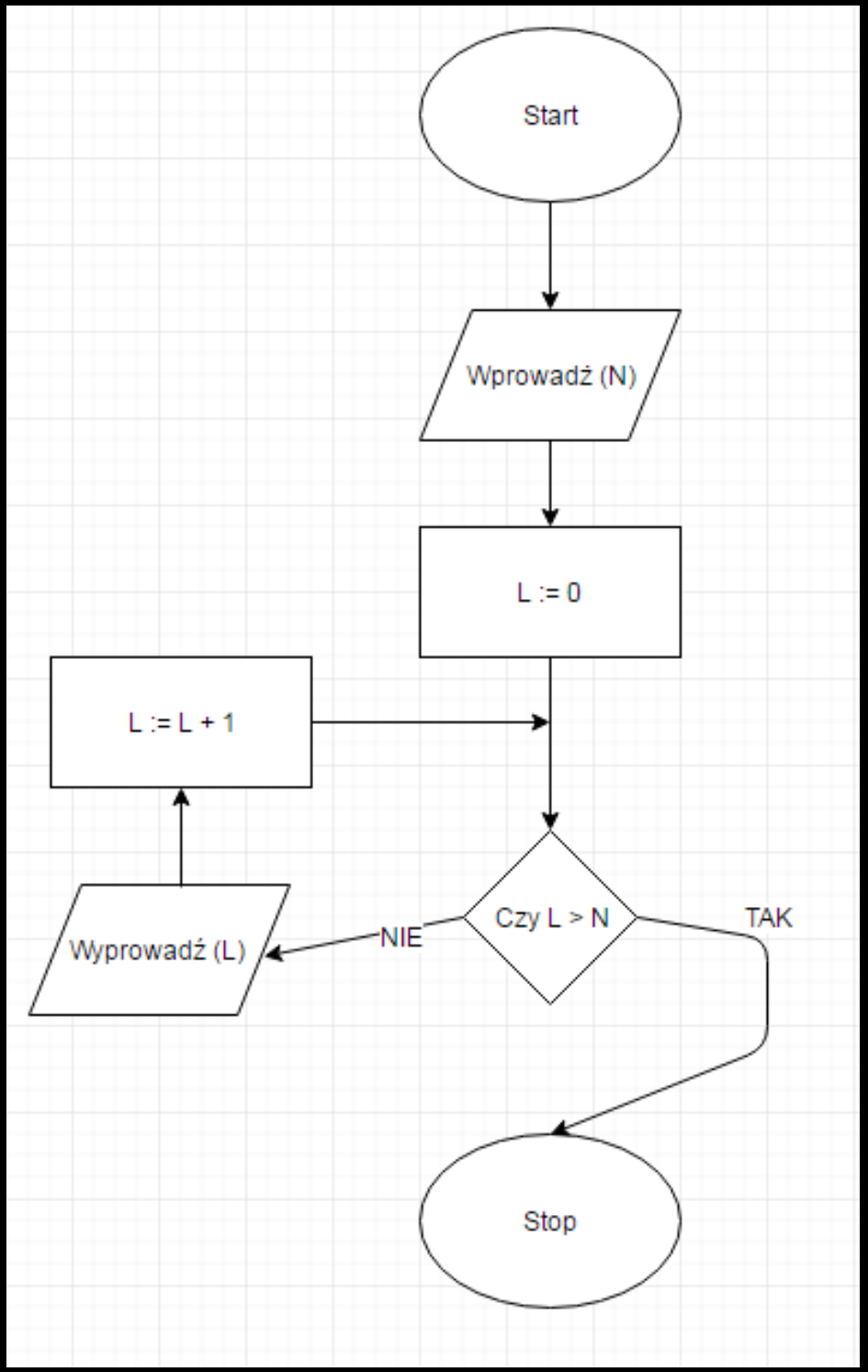
Przykłady:

-obliczanie sumy kolejnych cyfr

-obliczanie wartości A^n

Sam mechanizm iteracji nazywany jest potocznie pętlą…

Przykład: algorytm wypisujący N kolejnych cyfr całkowitych zaczynając od zera.



Licznik

Warto zauważyć, że w powyższym algorytmie pojawił się licznik. Pełni on zazwyczaj w algorytmach iteracyjnych kluczową rolę.

Licznik pętli to pewna zmienna, która kontroluje zachowanie się pętli, określa ona ile razy zostanie powtórzony dany ciąg instrukcji.

Pamiętaj, że licznik określa przejścia pętli, czyli musi to być liczba całkowita. Pętla nie może przecież wykonać się np. 2,2 razy!

Warunek pętli

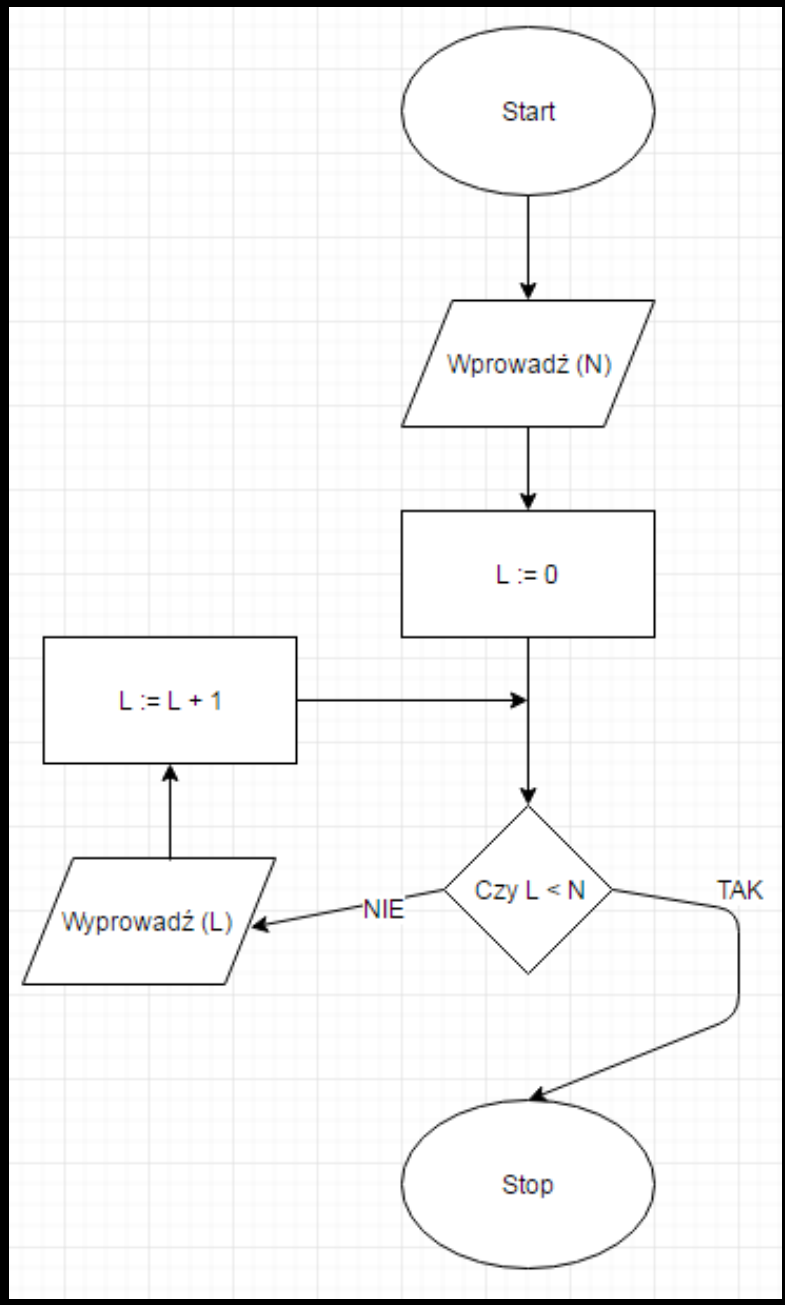
Jest to bardzo ważna cześć algorytmu iteracyjnego i źle dobrany warunek może zaważyć na poprawnym jego działaniu.

Warunek pętli

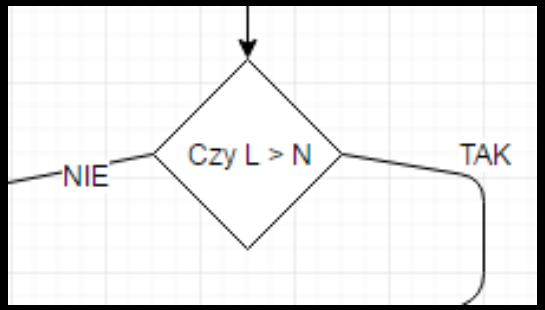
Przykład:

Ten sam algorytm generowania N kolejnych liczb całkowitych zaczynając od zera. Jednak warunek pętli jest zły.

W tym wypadku przy wpisaniu dowolnej wartości N większej od zera, mówiącej ile liczb ma być wypisanych, spowoduje, że żadna liczba nie będzie wypisana…



Powinno być:



Warunek pętli

W skrajnych przypadkach źle dobrany warunek działania/końca pętli może doprowadzić do sytuacji w której pętla nigdy się nie skończy…

Efekt – program realizujący taki algorytm po prostu się zawiesi.

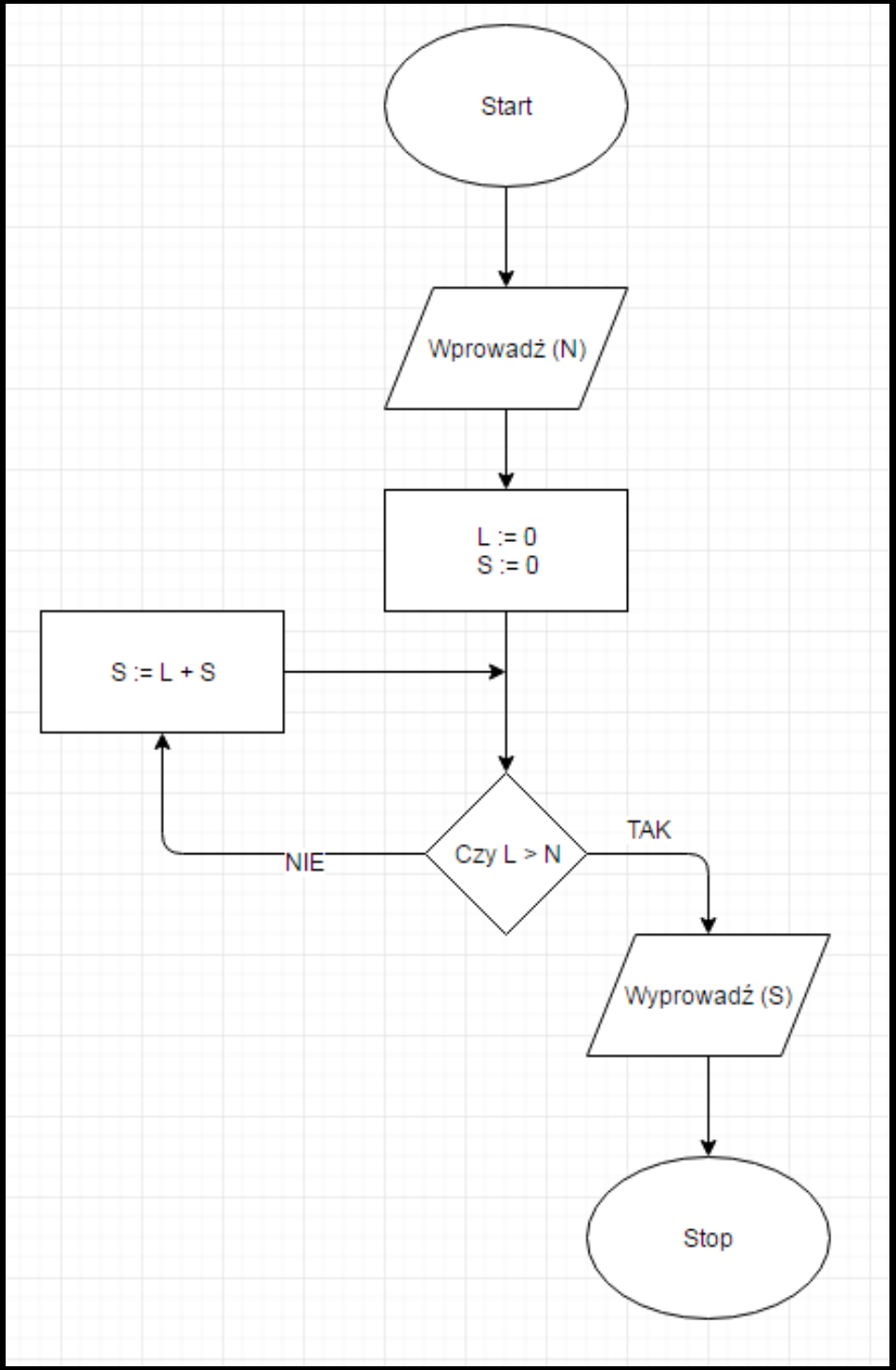
Warunek pętli – Licznik

Wniosek:

Zarówno dobrze dobrany warunek działania pętli jak również odpowiednio dobrana wartość początkowa licznika (jeśli jest konieczność jego stosowania) są gwarantem poprawnie działającego algorytmu iteracyjnego.

Inny przykład algorytmu iteracyjnego.

Co tym razem jest liczone?



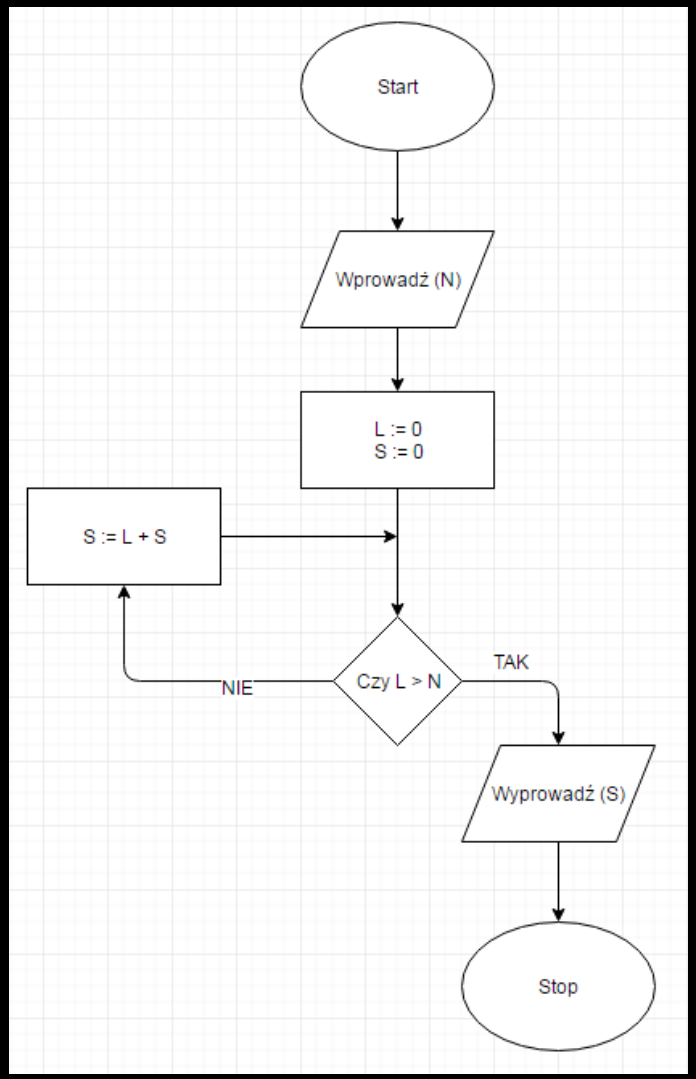
Zadanie: narysuj schemat blokowy algorytmu wypisującego na ekranie kolejne liczby parzyste z przedziału od 2 do N.

N to dowolna liczba całkowita dodatnia.

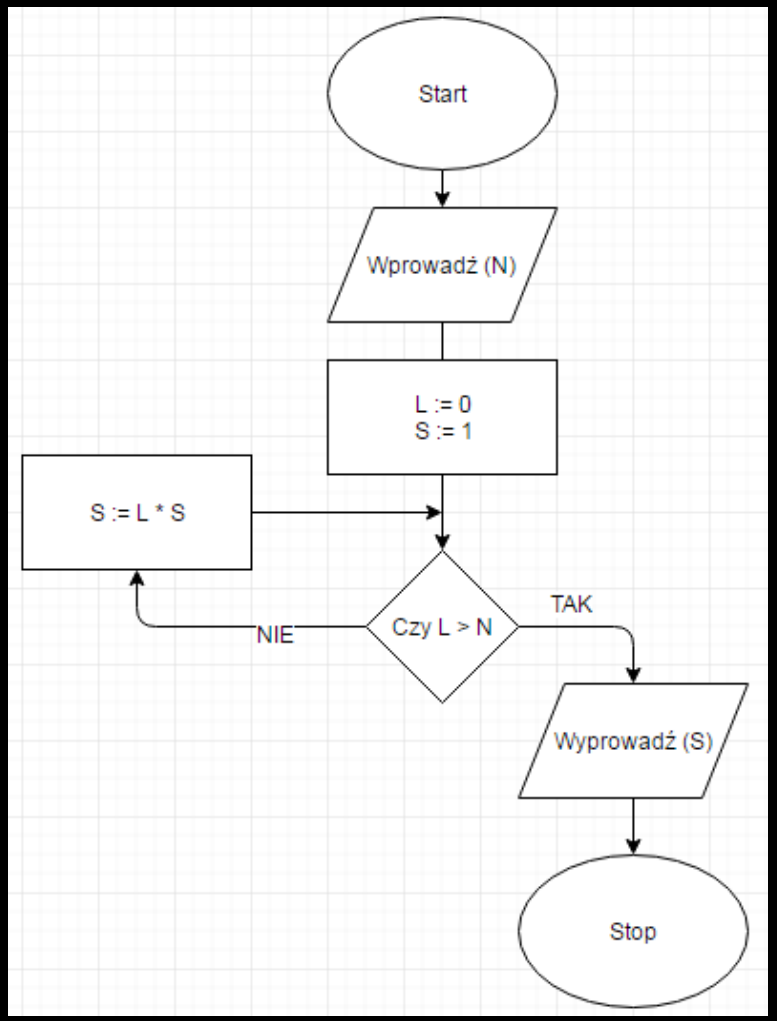
Suma / Iloczyn

Bardzo często algorytmy iteracyjne są stosowane do obliczania sumy lub iloczynu pewnej ilości liczb. Prezentują to poniższe przykłady:

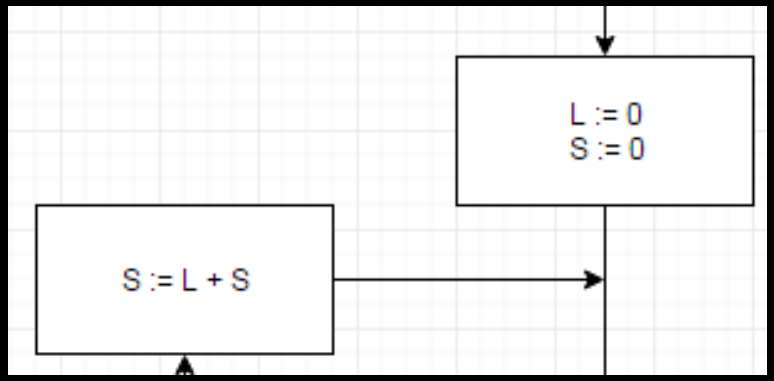
Dodawanie N liczb

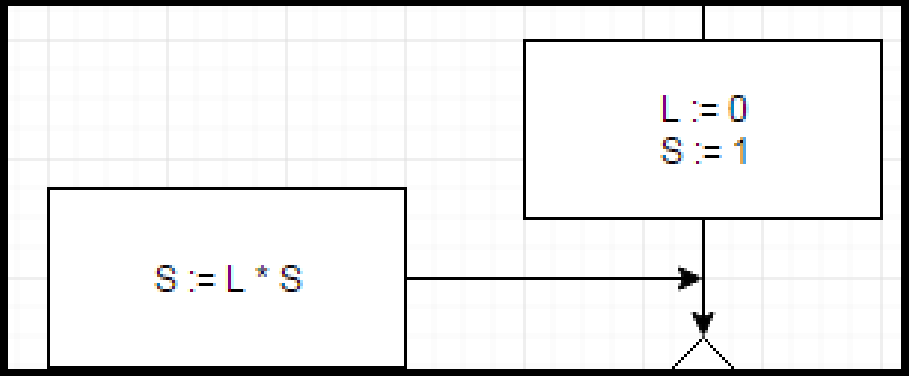


Mnożenie N liczb



Różnice:





1. **W przypadku dodawania wykonywane jest dodawanie (L+S), a w przypadku mnożenia mnożenie (L*S).**
2. **W przypadku dodawania początkowa wartość sumy S wynosi ZERO, a w przypadku mnożenia początkowa wartość sumy S wynosi JEDEN.**

Pierwsza różnica jest oczywista. Druga już niekoniecznie – jednak jest bardzo istotna. Wartość początkowa sumy przy takich działaniach musi być odpowiednia do działania.

Zadanie: narysuj schemat blokowy algorytmu obliczającego wartość A^n, gdzie A i n to liczby naturalne.

Podpowiedź: A^n to nic innego jak $\underbrace{A * A * \ldots * A}_{\text{n razy}}$

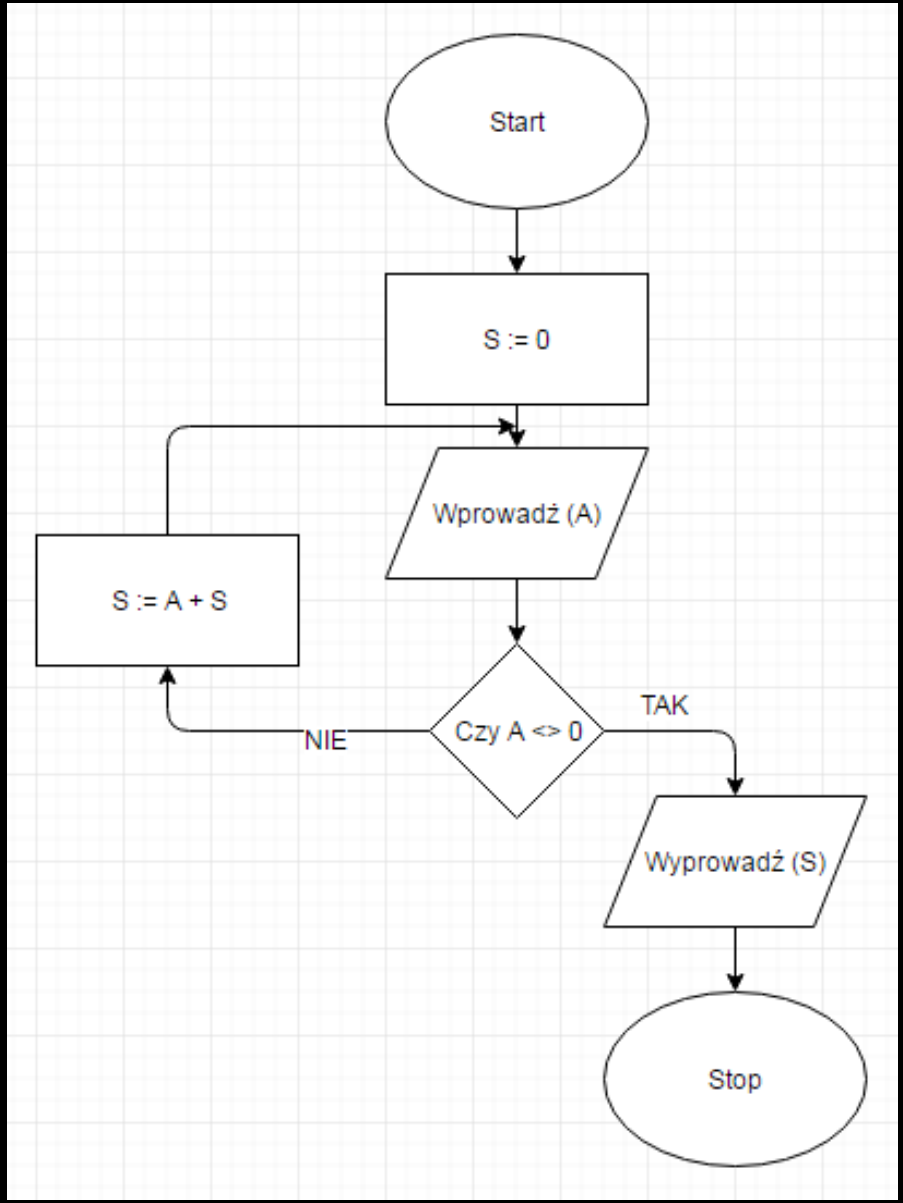
Czyli 2^3 to $2*2*2$

Zadanie: narysuj schemat blokowy algorytmu obliczającego wartość średnią N kolejno wprowadzanych wartości liczby X.

Algorytmy iteracyjne nie zawsze muszą zakładać powtarzanie pewnej czynności określoną z góry ilość razy. Czasem wykonywanie czynności jest powtarzana do czasu osiągnięci pewnego warunku.

Przykład: Algorytm sumuje kolejne wprowadzane liczby do momentu wprowadzenia wartości ZERO.

Nie wiadomo ile razy wykona się operacja dodawania S:=A+S…



Zadanie domowe: narysuj schemat blokowy algorytmu obliczającego wartość N! (silnia), gdzie N to liczba naturalna.

Funkcję N! definiuje się następująco:

$$n! = \prod_{k=1}^{n} k \qquad \text{dla } n \geqslant 1$$

Wartość 0! określa się osobno:

$$0! = 1$$

Definicja rekurencyjna silni ma postać:

$$n! = \begin{cases} 1 & \text{dla } n = 0 \\ n \cdot (n-1)! & \text{dla } n \geqslant 1 \end{cases}$$

Przykłady:

$$4! = 1 \cdot 2 \cdot 3 \cdot 4 = 24$$
$$5! = 1 \cdot 2 \cdot 3 \cdot 4 \cdot 5 = 120$$
$$6! = 1 \cdot 2 \cdot 3 \cdot 4 \cdot 5 \cdot 6 = 720$$

KONIEC

Więcej przykładów i ćwiczeń na kolejnych zajęciach…