



# Algorytmy

# Definicja



**Algorytm** - ciąg instrukcji tworzący rozwiązanie problemu, gdzie problemem nazywamy zamianę wartości wejściowych na wartości wyjściowe.

Wejście



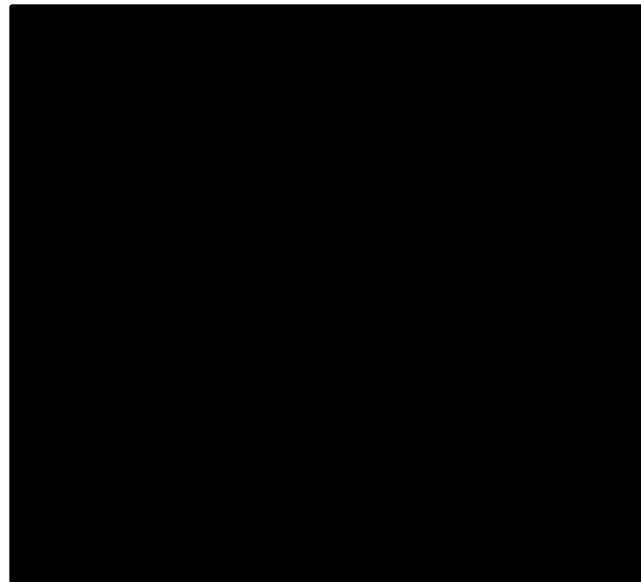
Wejście



Wyjście

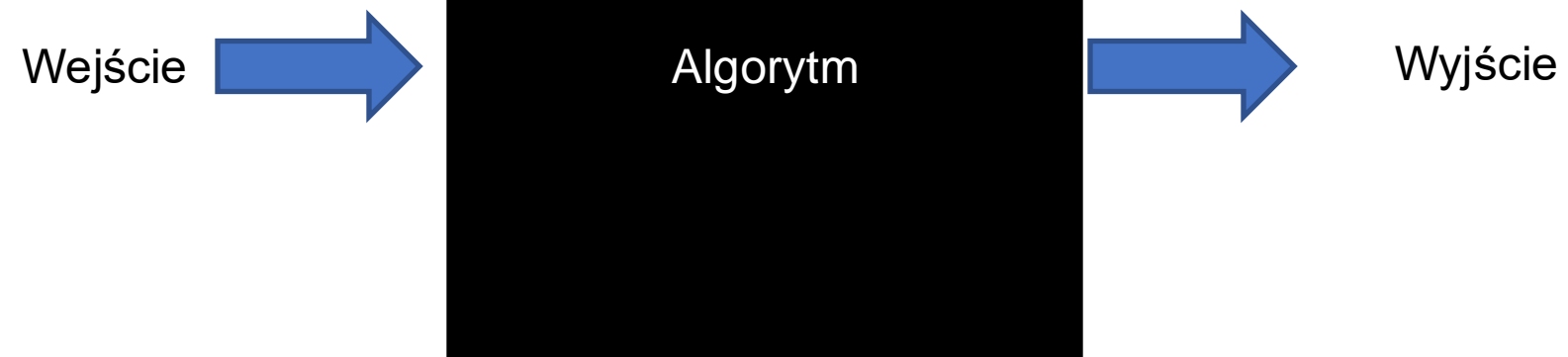


Wejście

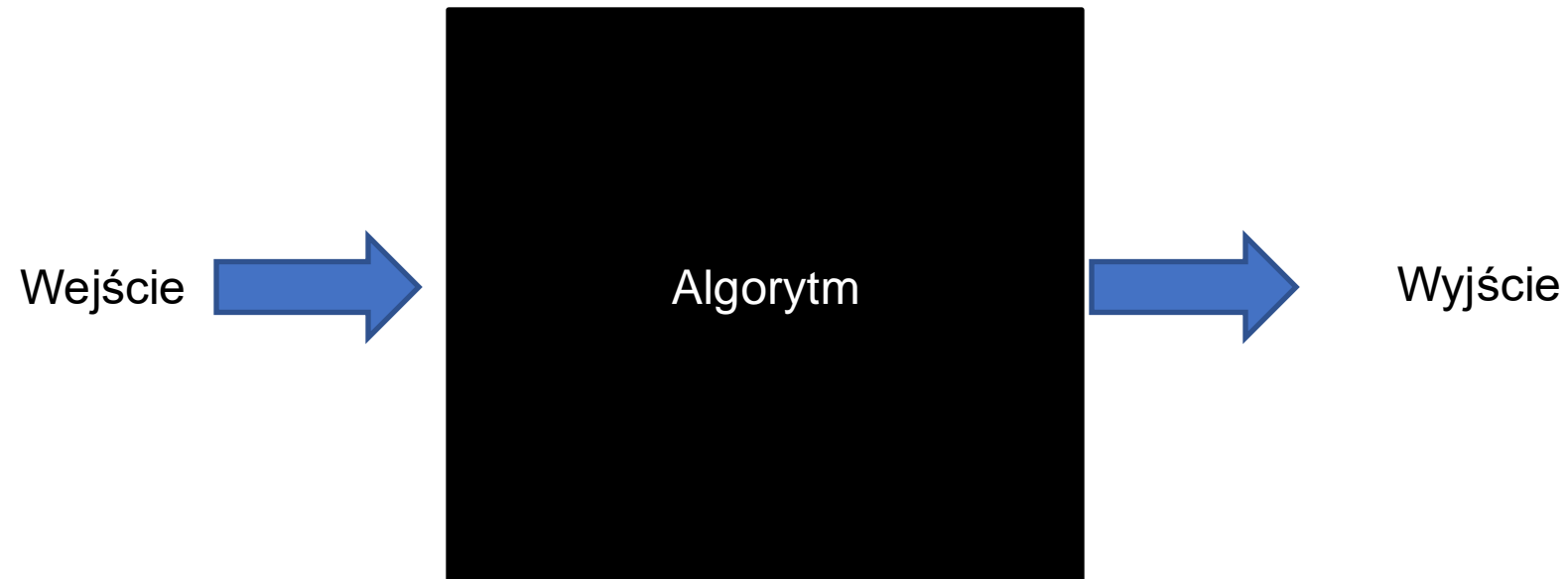


Wyjście





Algorytm - ciąg instrukcji będący rozwiązaniem problemu\*.



\*problem – zamiana parametrów wejściowych na wartości wyjściowe









# Ryba po grecku

sierysuje.pl

## SKŁADNIKI:

dla 4 osób

500 g filetów z dorsza



2 marchewki  
2 pietruszki



2 cebule

100 ml przecieru  
pomidorowego



olej, bulion (może być kostka),  
mąka, 2 liście laurowe, słodka  
papryka w proszku, ziele  
angielskie, sól, pieprz

1

Marchew i pietruszkę zetrzyj  
na grubych oczkach tarki



2

Cebule posiekaj  
i podsmaż na  
oleju



3

Dodaj marchew i pietruszkę, duś 10 minut,  
przypraw. Wlej szklankę bulionu, gotuj  
jeszcze kwadrans. Dodaj przecier, duś  
kilka minut.

4

Rybę obtocz  
w mące, usmaż  
na rumiano



5

Ułóż rybę na półmisku,  
przykryj warzywami.





# Wejście



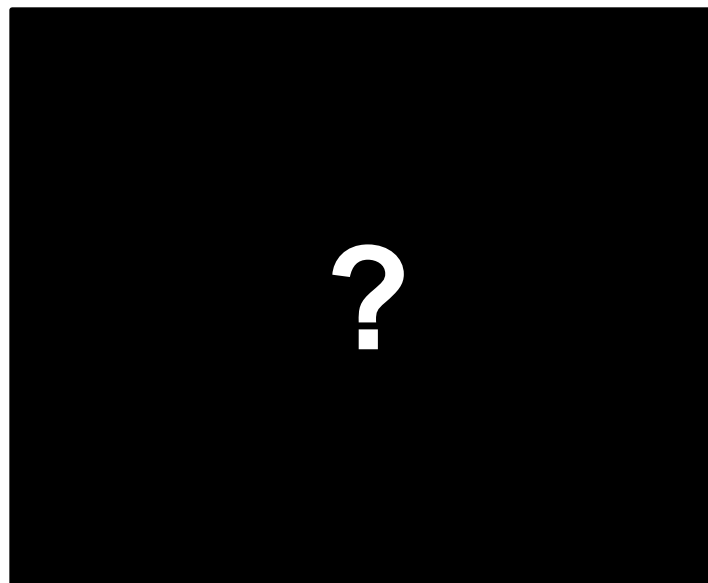
**Wejście**



**Wyjście**



**Wejście**



**Wyjście**



## Wejście



## Algorytm

**Ryba po grecku**  
sieryouje.pl

**SKŁADNIKI:**  
dla 4 osób

- 500 g filetów z dorsza
- 2 marchewki
- 2 pietruszki
- 2 cebule
- 100 ml przecieru pomidorowego
- olej, bulion (może być kostka), mąka, 2 liście laurowe, słodka papryka w proszku, ziele angielskie, sól, pieprz

- 1 Marchew i pietruszkę zetrzyj na grubych oczkach tarki
- 2 Cebule posiekaj i podsmaż na oleju
- 3 Dodaj marchew i pietruszkę, duś 40 minut, przypraw. Wlej szklankę bulionu, gotuj jeszcze kwadrans. Dodaj przecier, duś kilka minut
- 4 Rybę obtocz w mące, usmaż na rumiano
- 5 Ułóż rybę na płemiskach, przykryj warzywami

## Wyjście





# Podstawowe elementy algorytmu

# Instrukcje elementarne





# Instrukcje sterujące



## 1. Bezpośrednie następstwo

- *"wykonaj A, a potem B"*

## 2. Rozgałęzienie (aka wybór warunkowy, warunek)

- *"jeśli Q to wykonaj A, w przeciwnym razie wykonaj B"*

Przy użyciu tylko tych dwóch instrukcji liczba linijek w zapisie algorytmu będzie pokrywała się z liczbą potrzebnych do wykonania kroków.

## 3. Powtórzenia (aka zwroty pętłace, pętle)

- *"dopóki Q wykonuj A"*

**Wejście**

5

6



**Wejście**

**Wyjście**

5



6



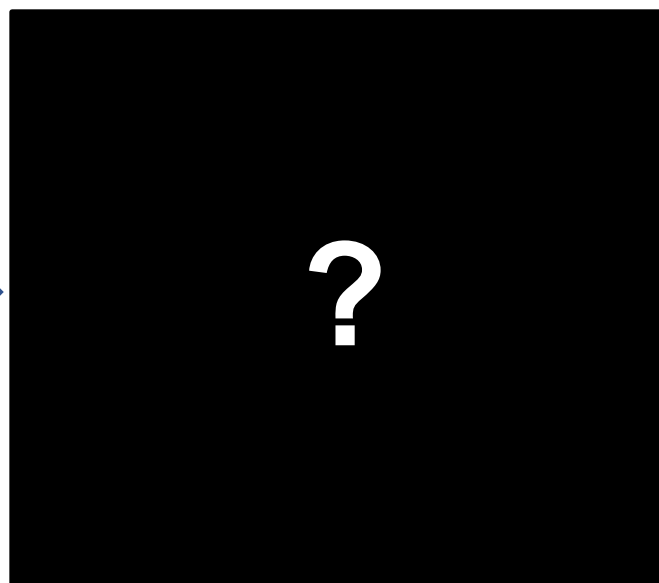
11

Wejście

Wyjście

5

6



11

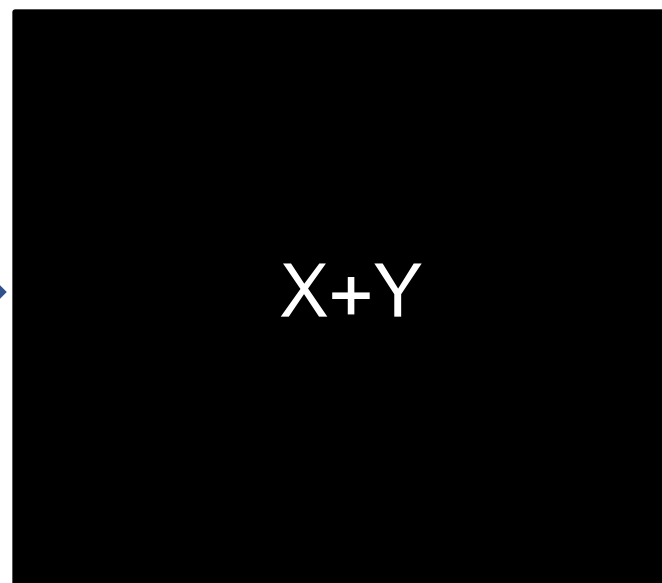
Wejście

Algorytm

Wyjście

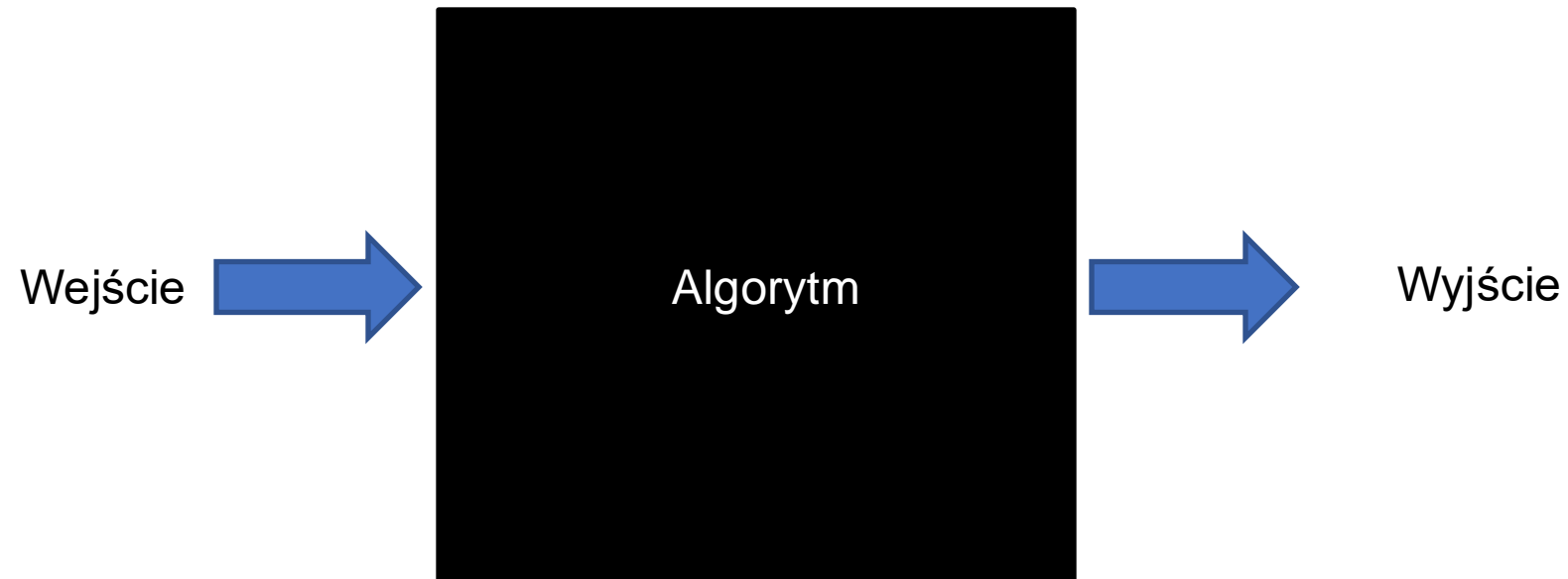
5

6



11

Algorytm - ciąg instrukcji będący rozwiązaniem problemu\*.



\*problem – zamiana parametrów wejściowych na wartości wyjściowe



# Wydajność algorytmu

# Wydajność algorytmu



Wejście



Wyjście



Jan Kowalski 111-222-333





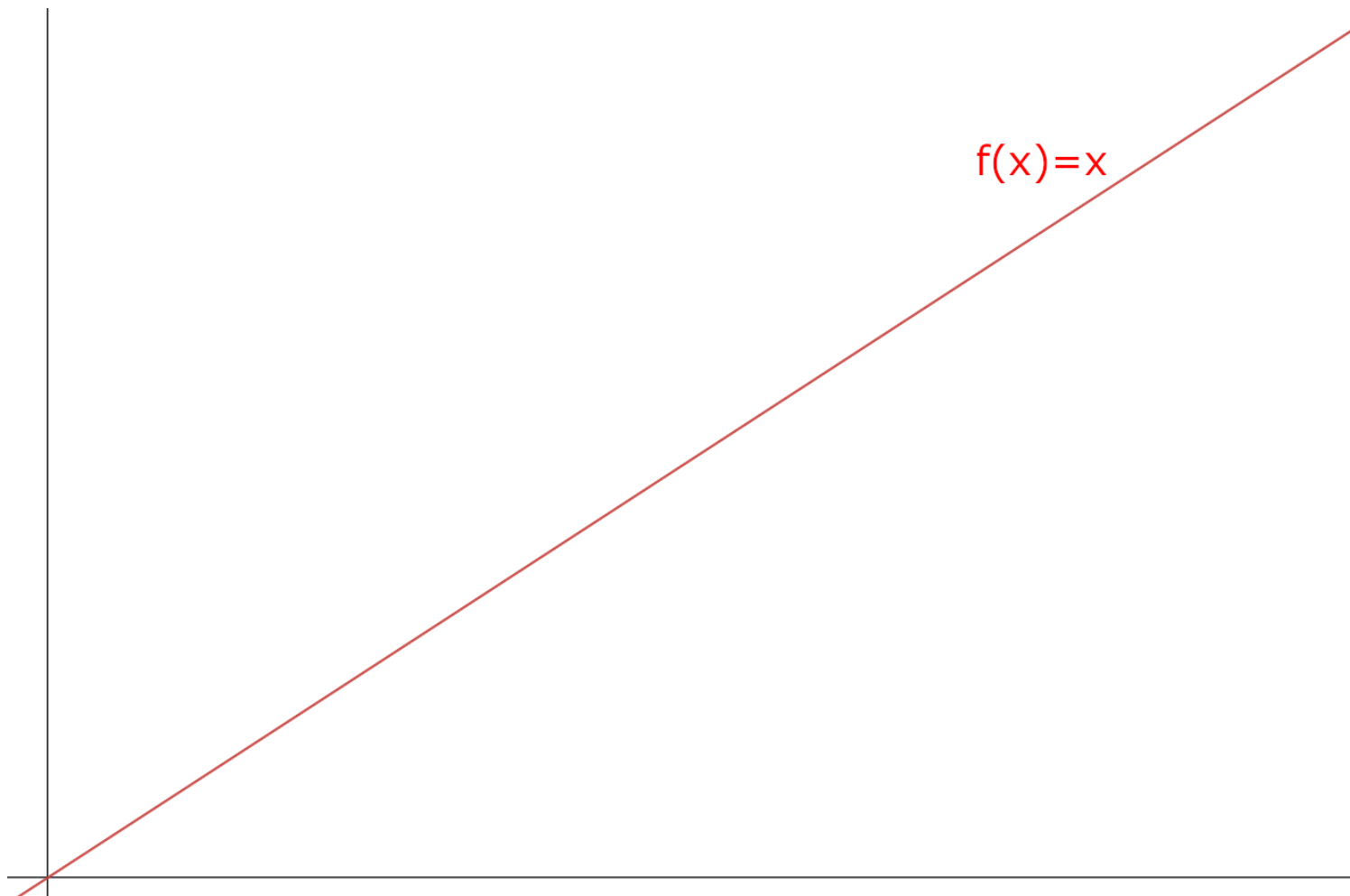
# Złożoność obliczeniowa

Wydajność algorytmu możemy mierzyć **liczbą instrukcji elementarnych** potrzebnych do jego wykonania.

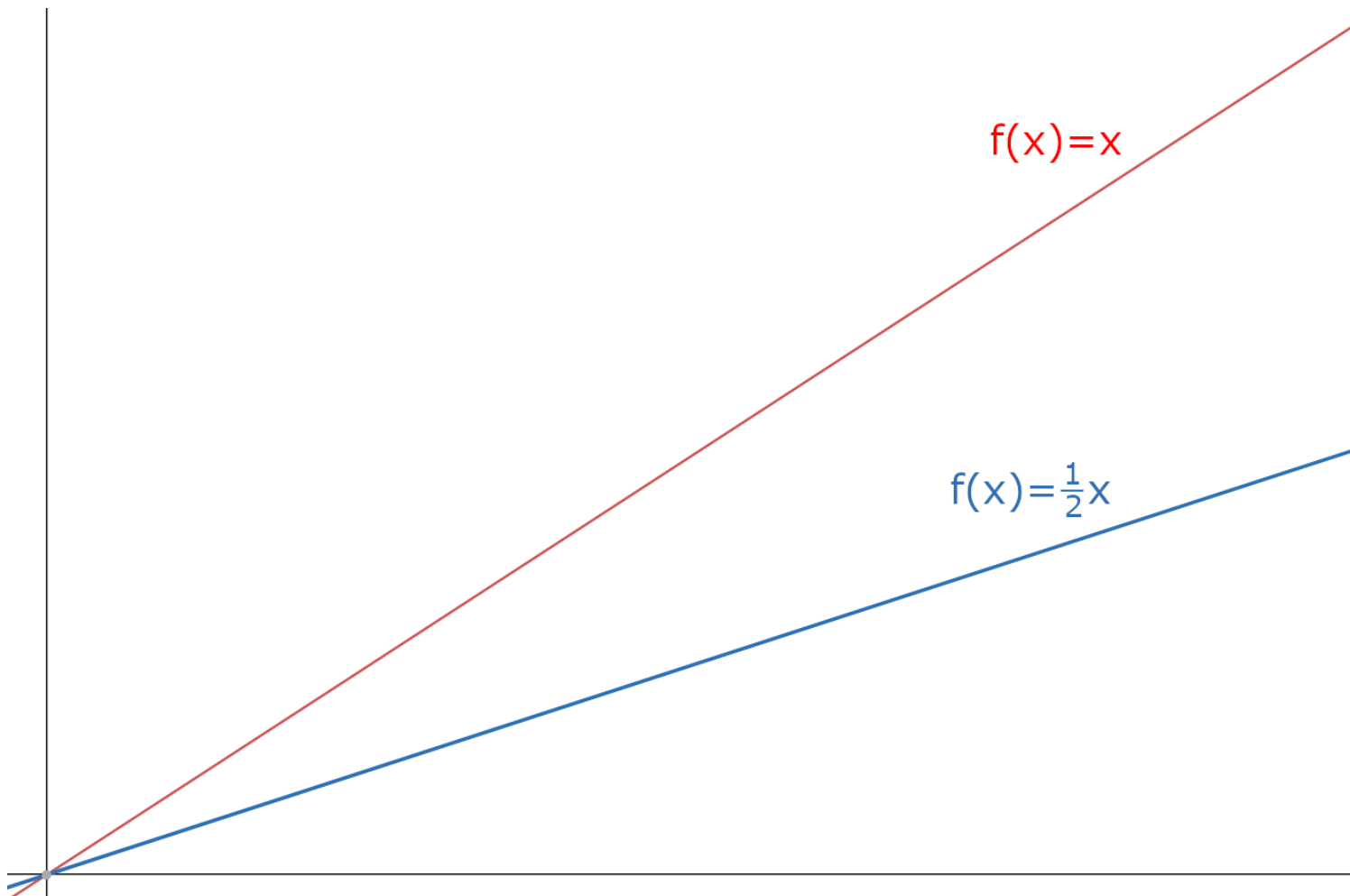
Liczba instrukcji elementarnych może się zmieniać w zależności od rozmiaru danych wejściowych (algorytm sortowania wykona inną liczbę operacji przy porządkowaniu dwuelementowej listy, a inną przy porządkowaniu stu-elementowej listy, chociaż sposób jego działania się nie zmieni). Dlatego szacowanie przeprowadza się **w funkcji rozmiaru danych wejściowych**.

**Złożoność obliczeniowa** – liczba instrukcji elementarnych w funkcji rozmiaru danych wejściowych

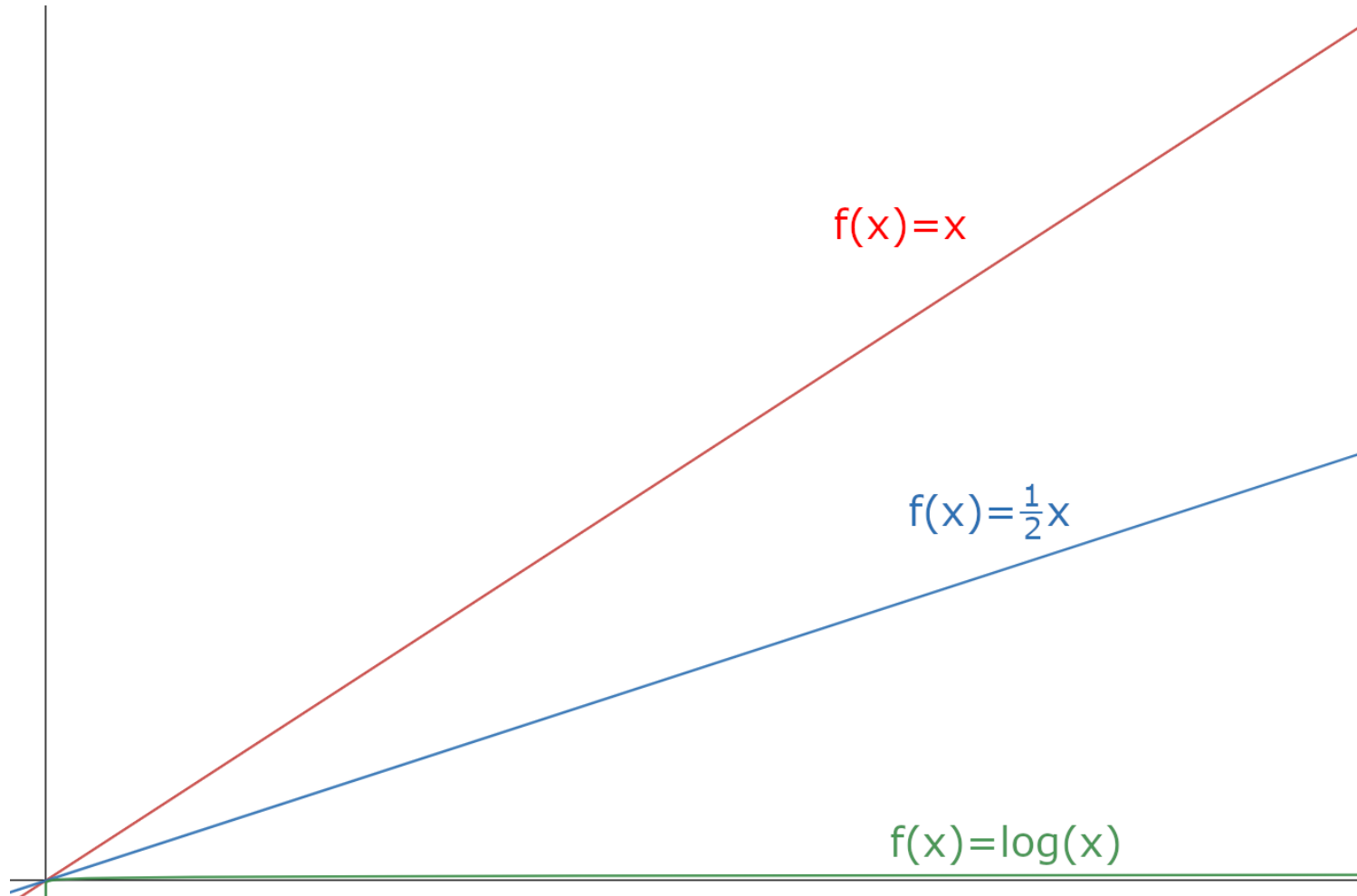
# Złożoność obliczeniowa



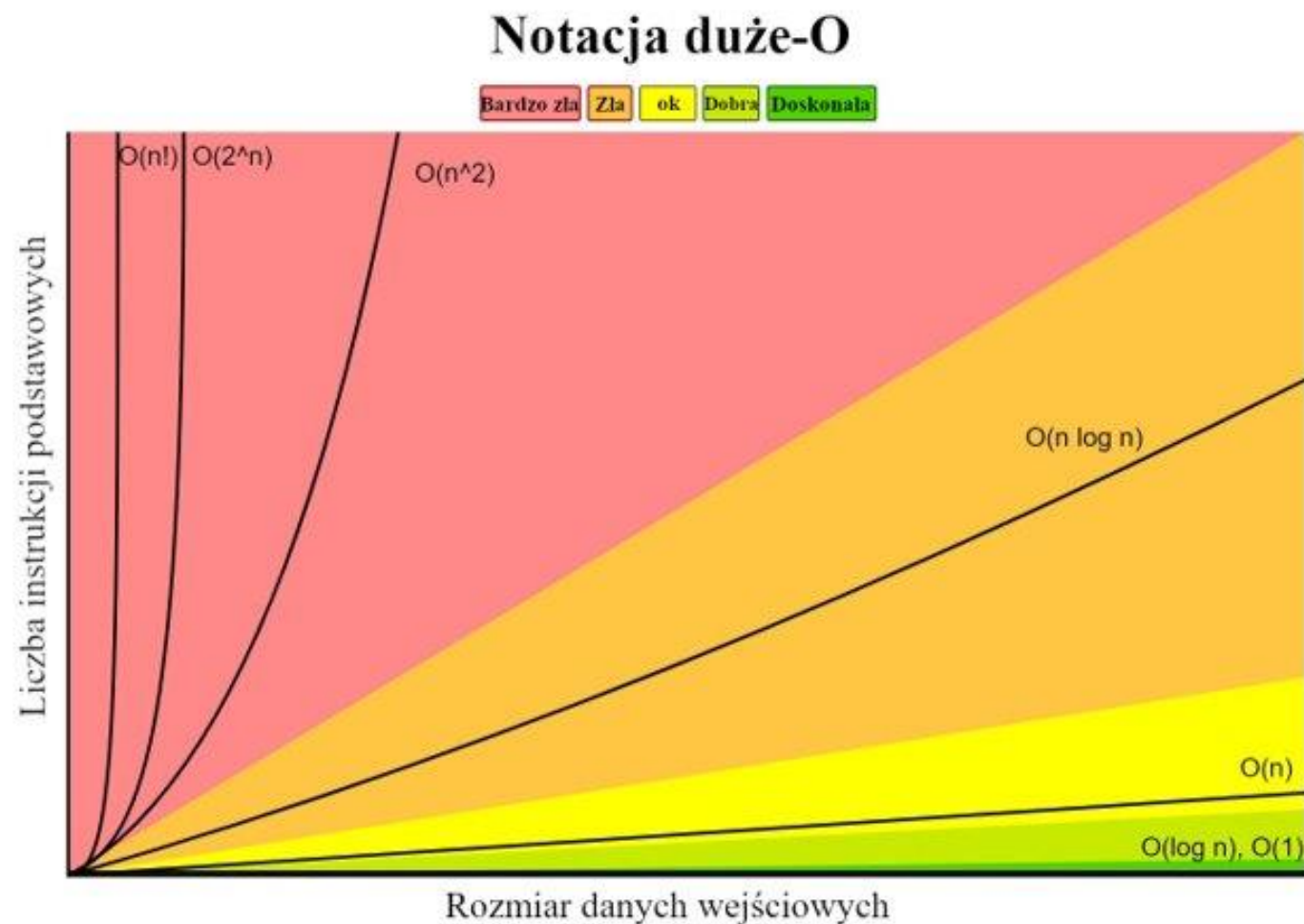
# Złożoność obliczeniowa



# Złożoność obliczeniowa



# Złożoność obliczeniowa

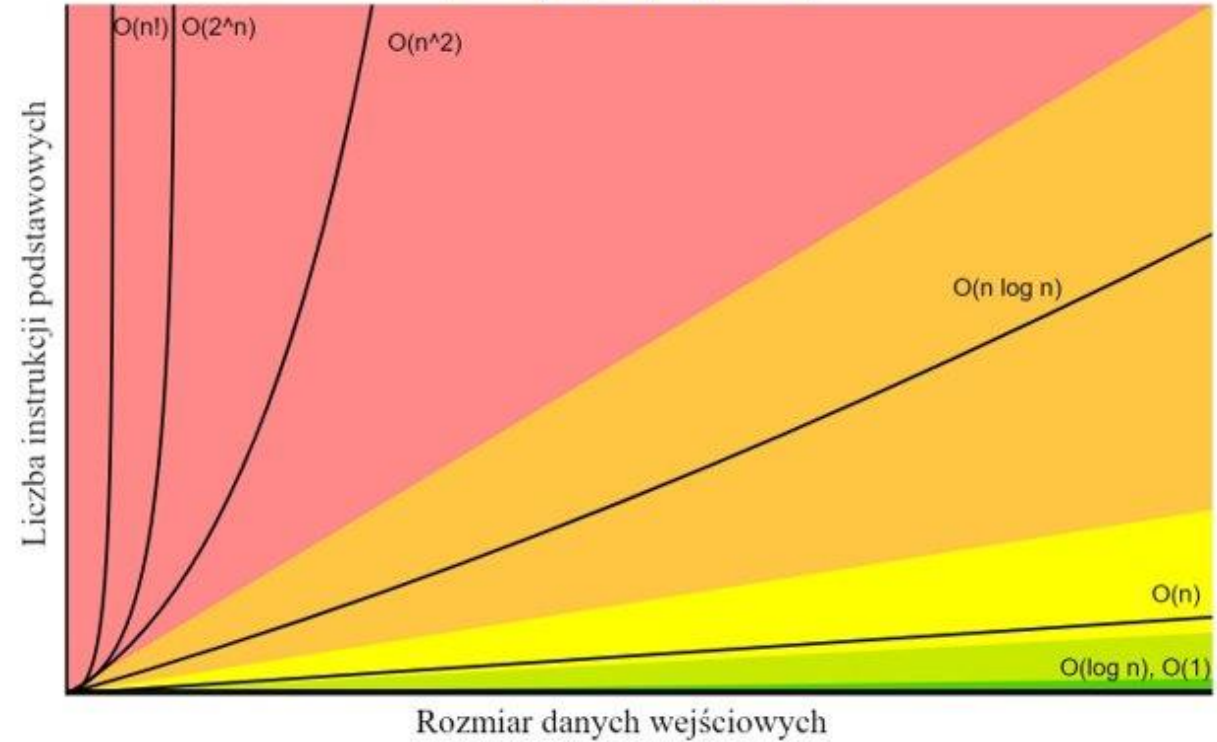




# Złożoność obliczeniowa

## Notacja duże-O

Bardzo zła Zła ok Dobra Doskonała



Oznaczenie	Nazwa
1	stała
$\log(n)$	logarytmiczna
$n$	liniowa
$n \log(n)$	liniowo-logarytmiczna
$n^2$	kwadratowa
$n^c$	wielomianowa
$c^n$	wykładnicza
$n!$	silnia



# Reprezentacja algorytmu

# Reprezentacje algorytmu



Zapis słowny

Lista kroków

Pseudokod

Drzewo algorytmu

Schemat blokowy





Weź książkę telefoniczną, otwórz po środku i przejrzyj wpisy. Jeżeli wśród wpisów znajdziesz Jana Kowalskiego, zapisz jego numer i zakończ. Jeżeli wśród wpisów nie ma Jana Kowalskiego, sprawdź czy pierwszy wpis znajduje się przed Janem Kowalskim w kolejności alfabetycznej. Jeżeli tak, otwórz po środku prawej części i ponów przeglądanie wpisów. W przeciwnym wypadku, ponów przeglądanie po środku lewej części. Zakończ, kiedy prawa i lewa część będą tą samą, jedną kartką.

Zapis słowny

Lista kroków

Pseudokod

Drzewo algorytmu

Schemat blokowy



1. Weź książkę telefoniczną
2. Otwórz po środku
3. Przejrzyj wpisy
4. Jeżeli Jan Kowalski wśród wpisów:
  - A. zapisz numer
5. W przeciwnym razie jeżeli Jan Kowalski wcześniej w książce:
  - A. otwórz po środku lewej części
  - B. wróć do kroku 3
6. W przeciwnym razie jeżeli Jan Kowalski później w książce:
  - A. otwórz po środku prawej części
  - B. wróć do kroku 3
7. W przeciwnym wypadku:
  - A. zakończ

Zapis słowny

Lista kroków

Pseudokod

Drzewo algorytmu

Schemat blokowy



Początek

start = 0

koniec = len(ksiazka)

dopóki start < koniec rób:

    srodek = (start + koniec) / 2

    jeżeli 'Jan Kowalski' w książka[srodek]:

        daj książka[srodek]['Jan Kowalski']

    zakończ

    jeżeli 'Jan Kowalski' > książka[srodek][0]:

        start = srodek

    w przeciwnym razie:

        koniec = srodek

Koniec

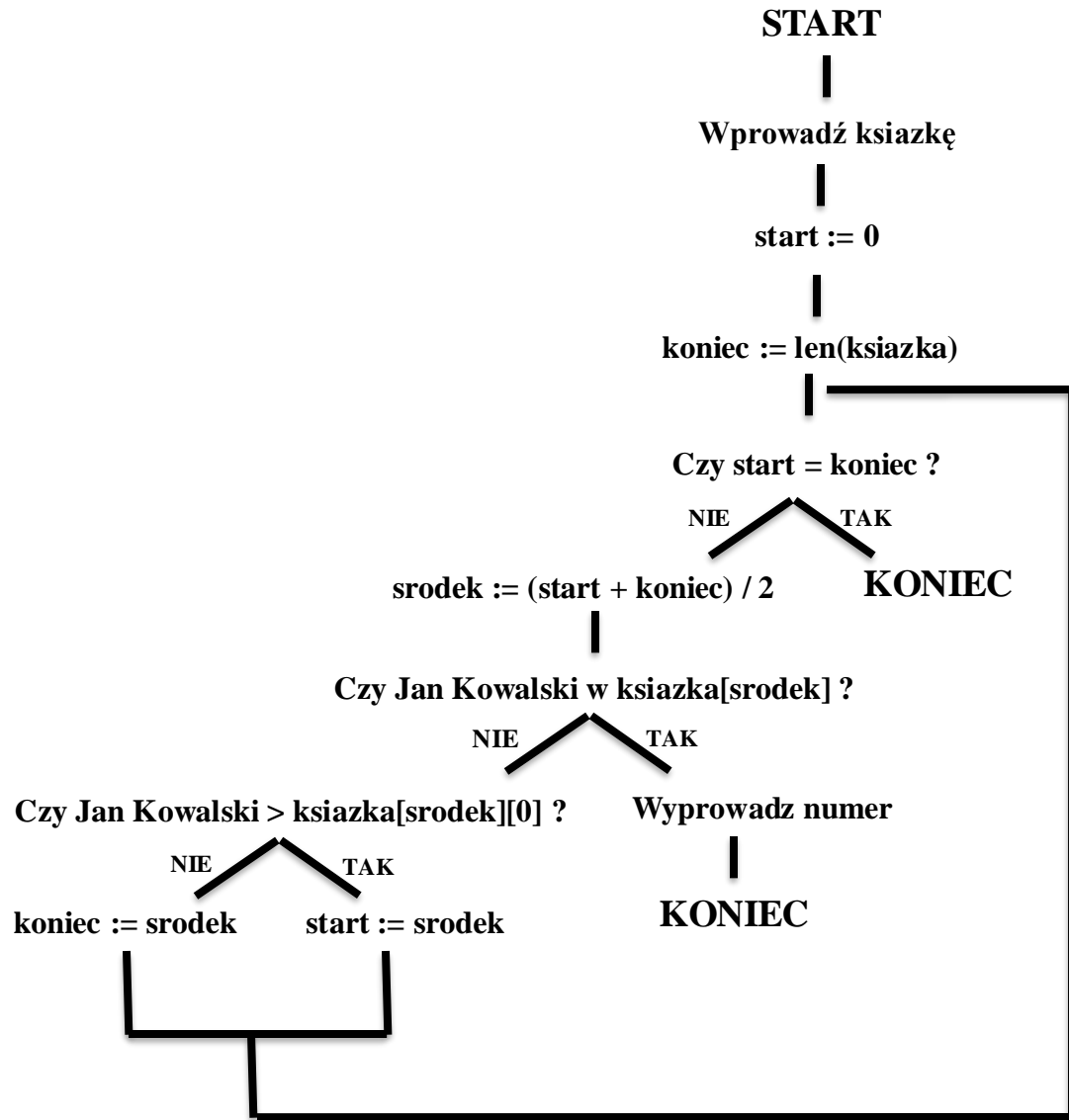
Zapis słowny

Lista kroków

Pseudokod

Drzewo algorytmu

Schemat blokowy



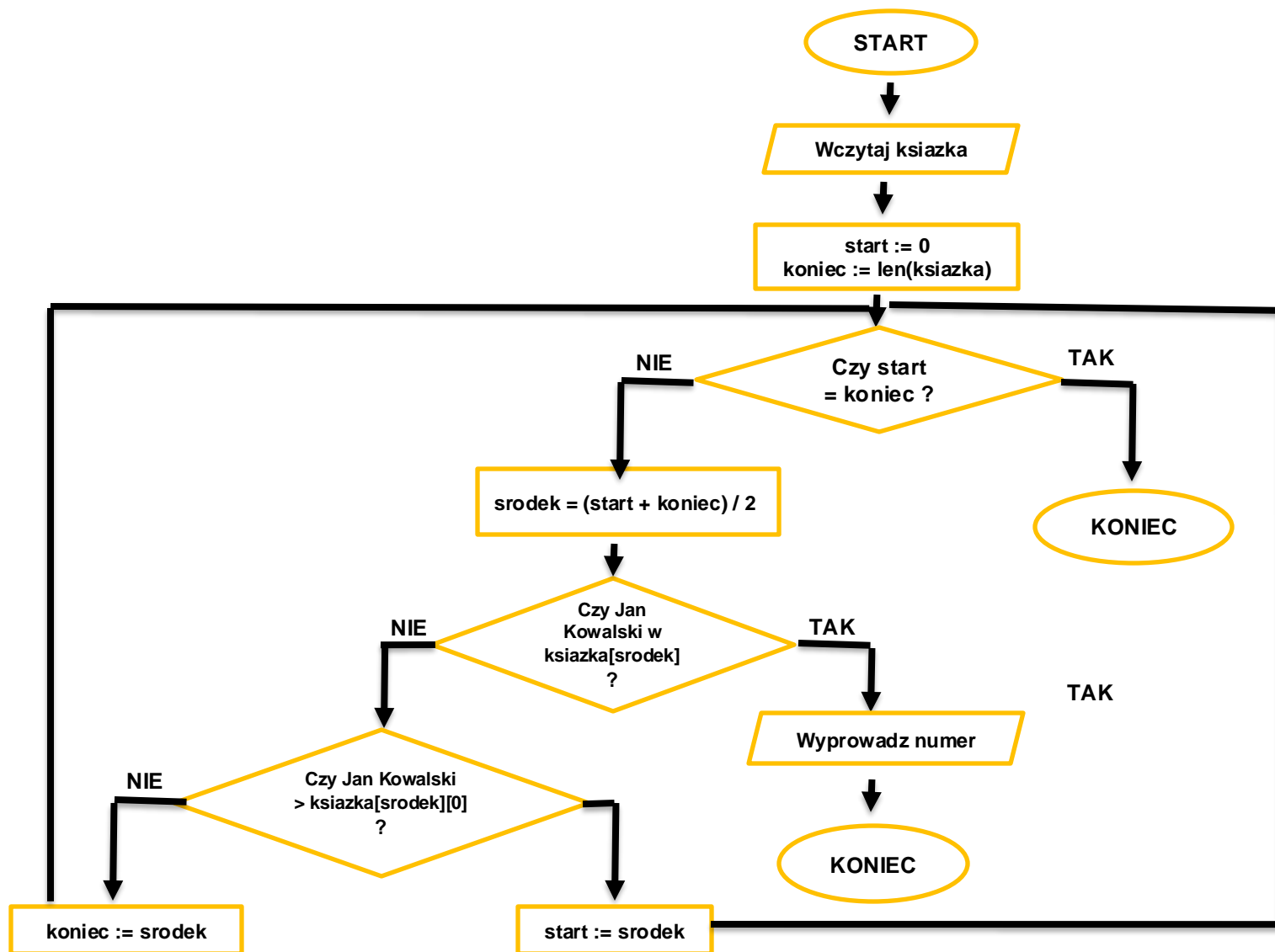
Zapis słowny

Lista kroków

Pseudokod

Drzewo algorytmów

Schemat blokowy



Zapis słowny





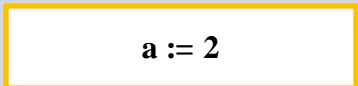
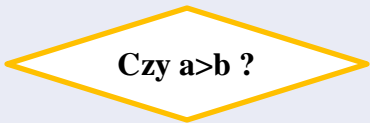

Lista kroków

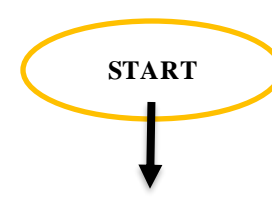
Pseudokod

Drzewo algorytmu

Schemat blokowy

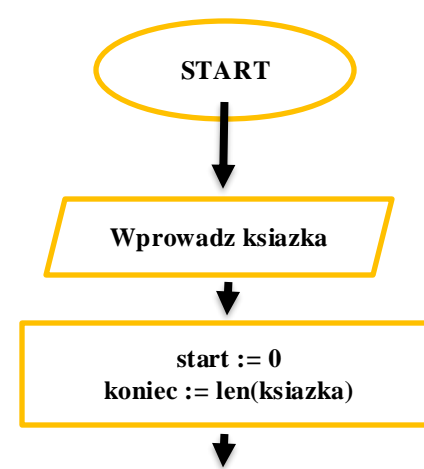


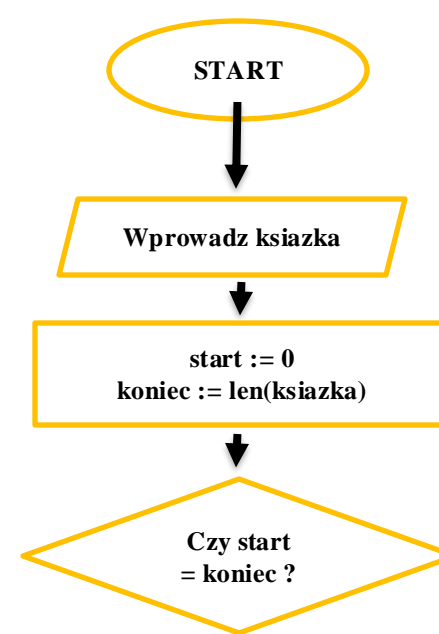
Reprezentacja graficzna	Opis
	Początek algorytmu
	Zakończenie algorytmu
	Blok wejścia
	Blok wyjścia
	Blok operacyjny
	Blok warunkowy (decyzyjny)
	Połączenie

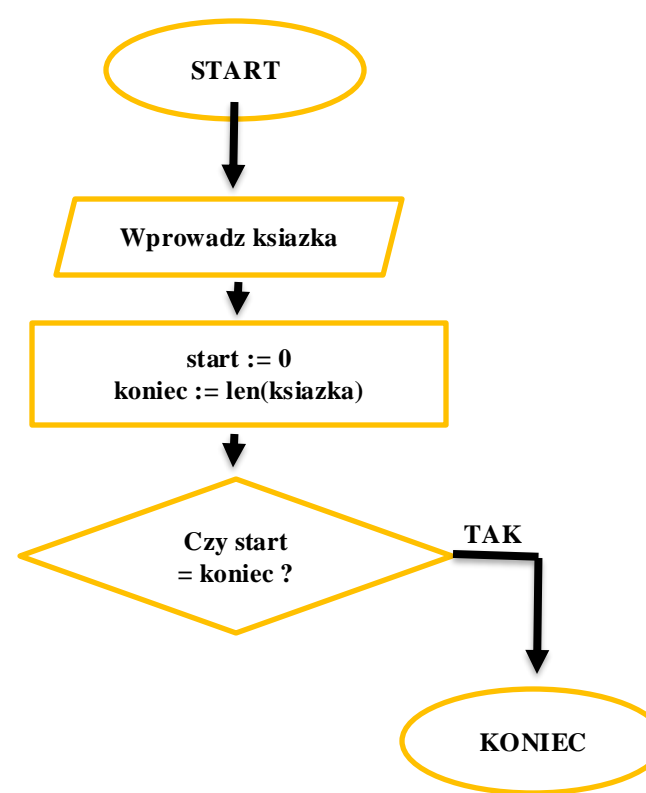


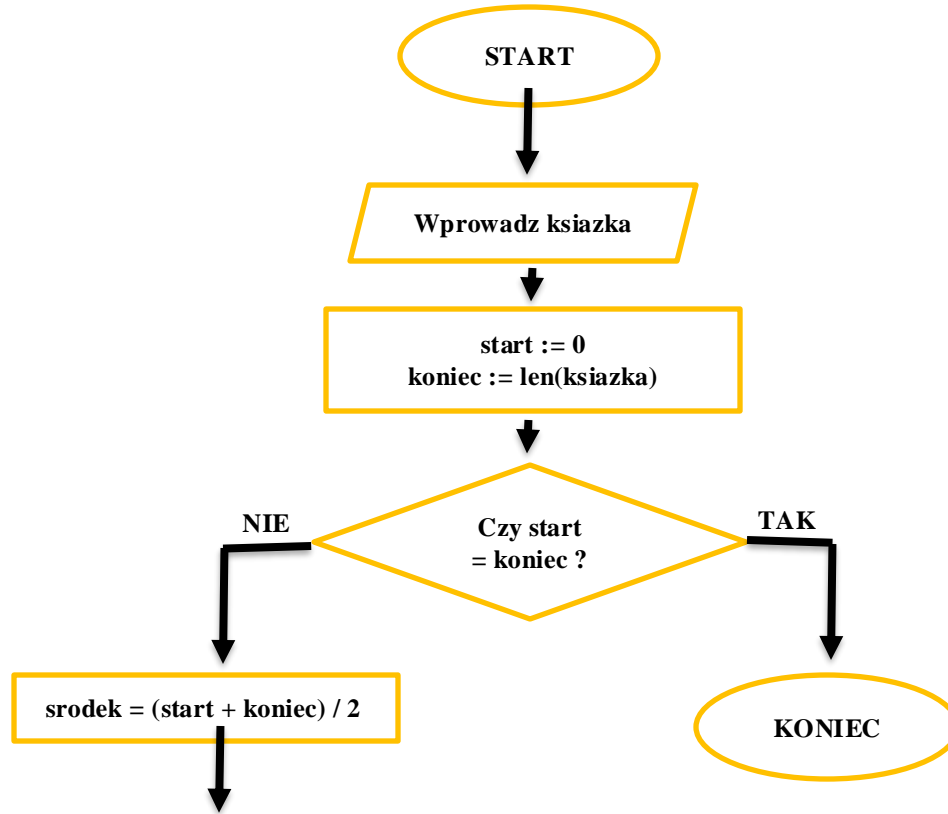


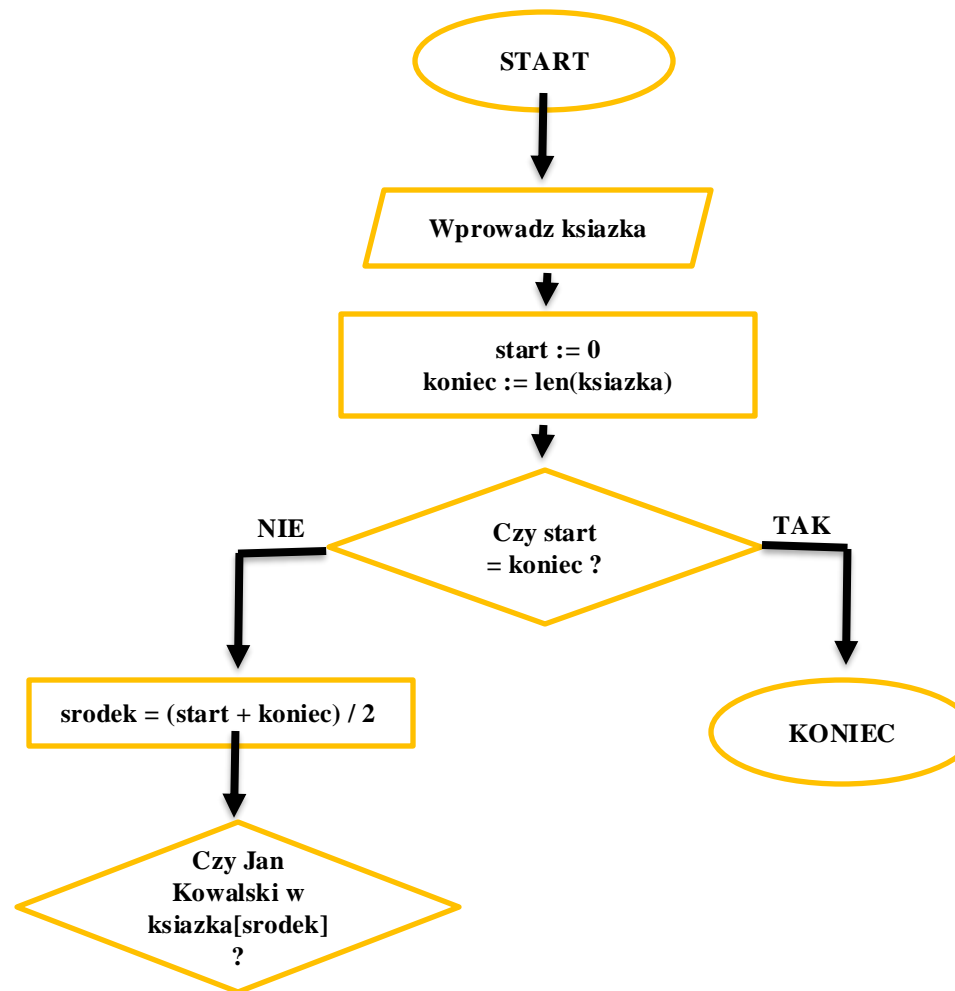


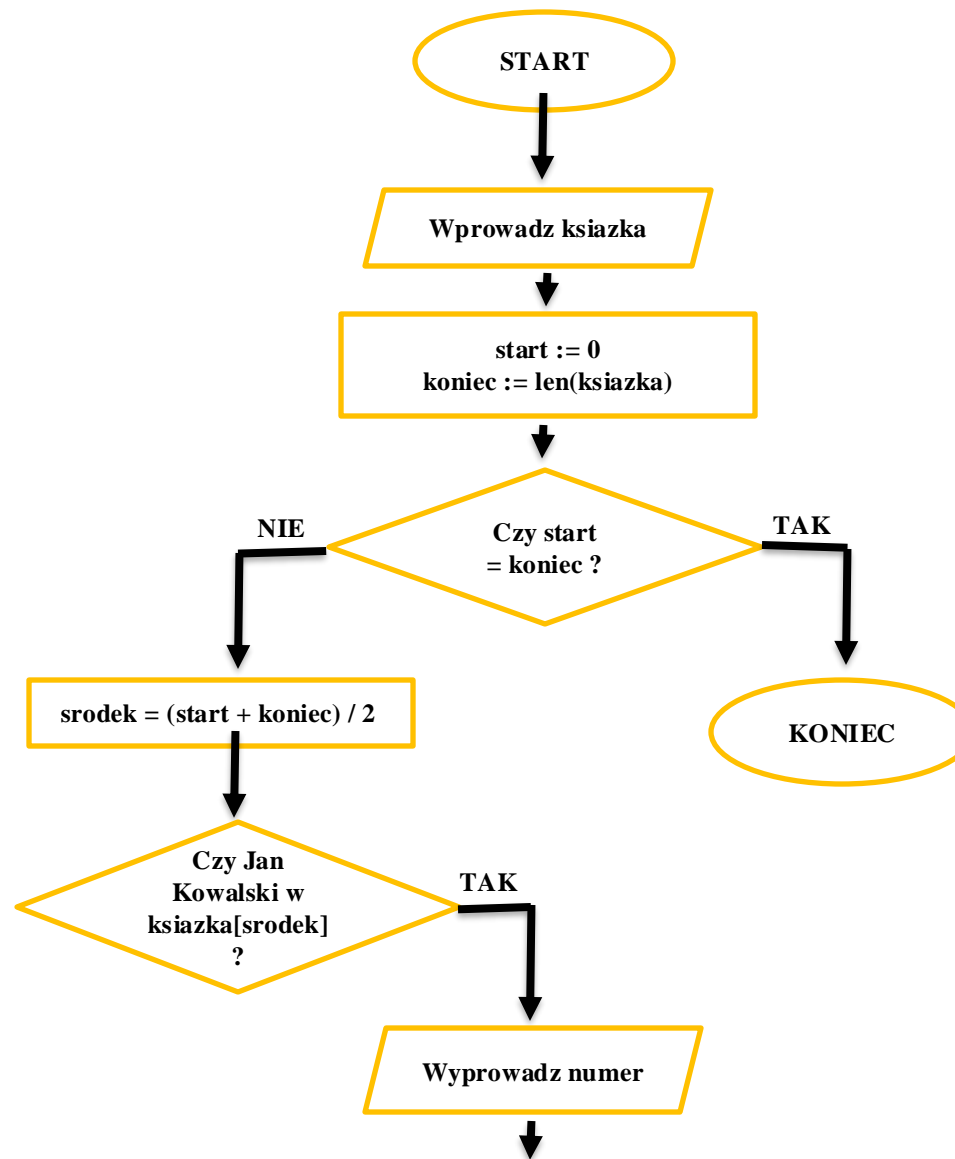


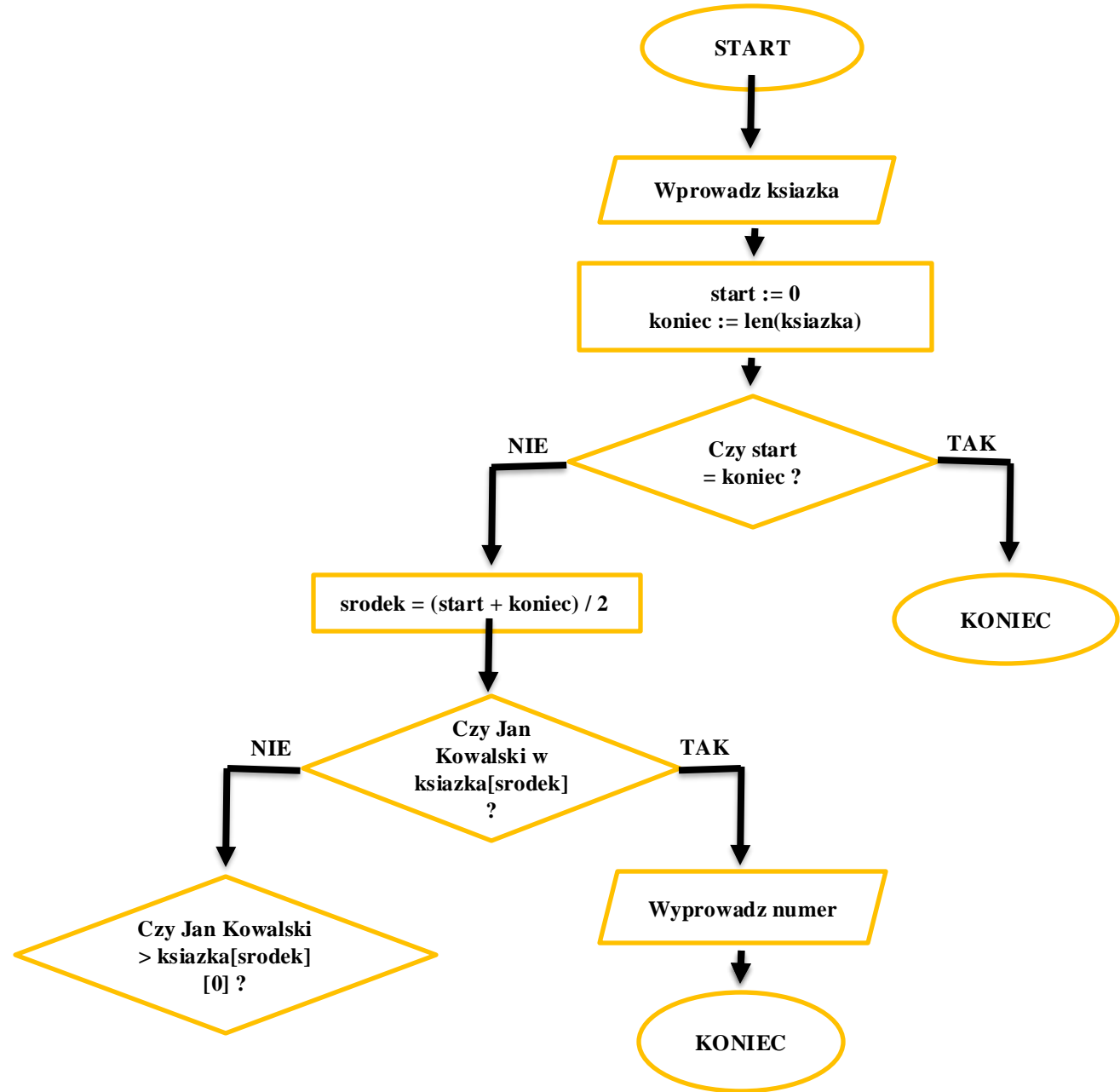


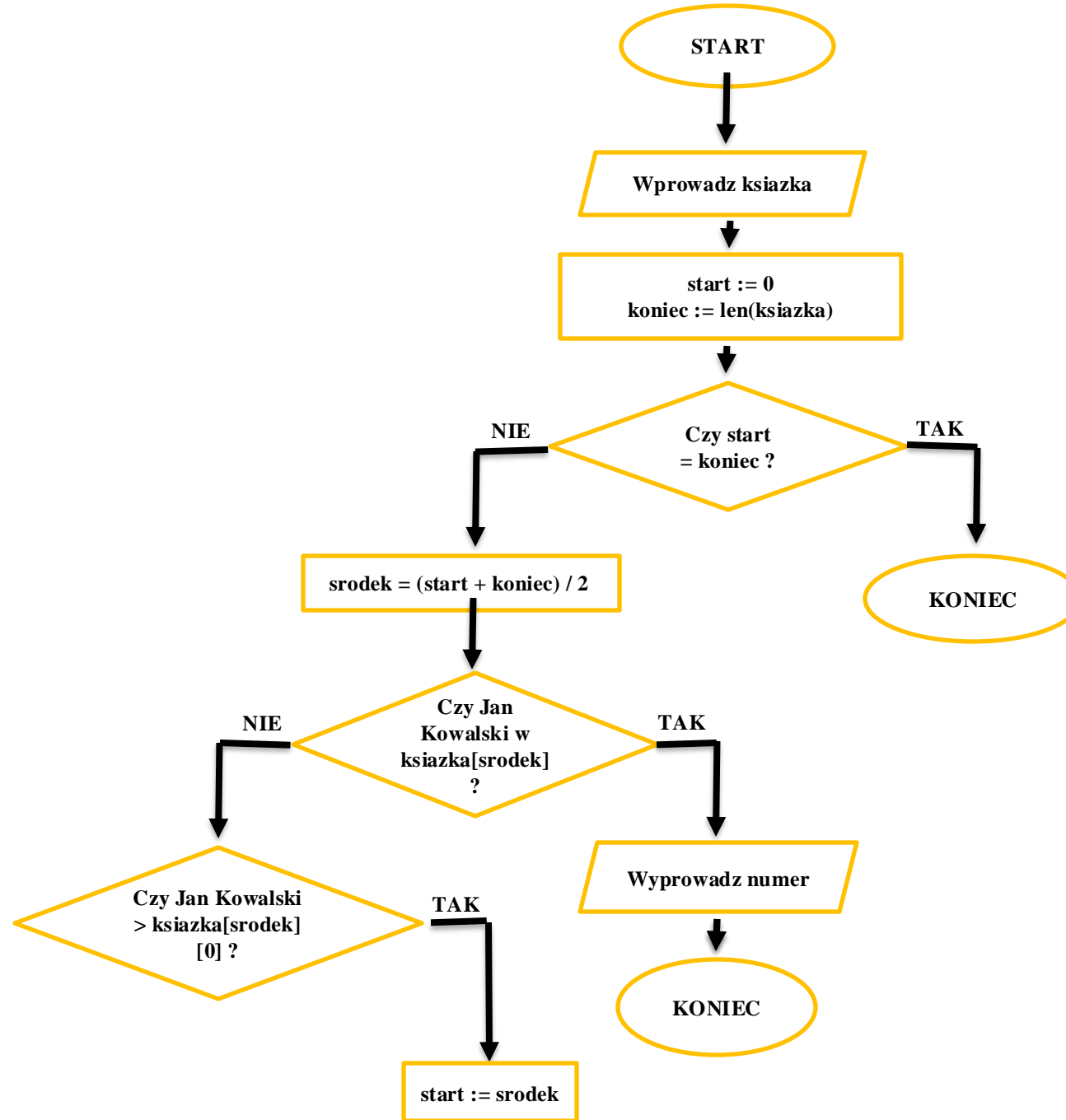




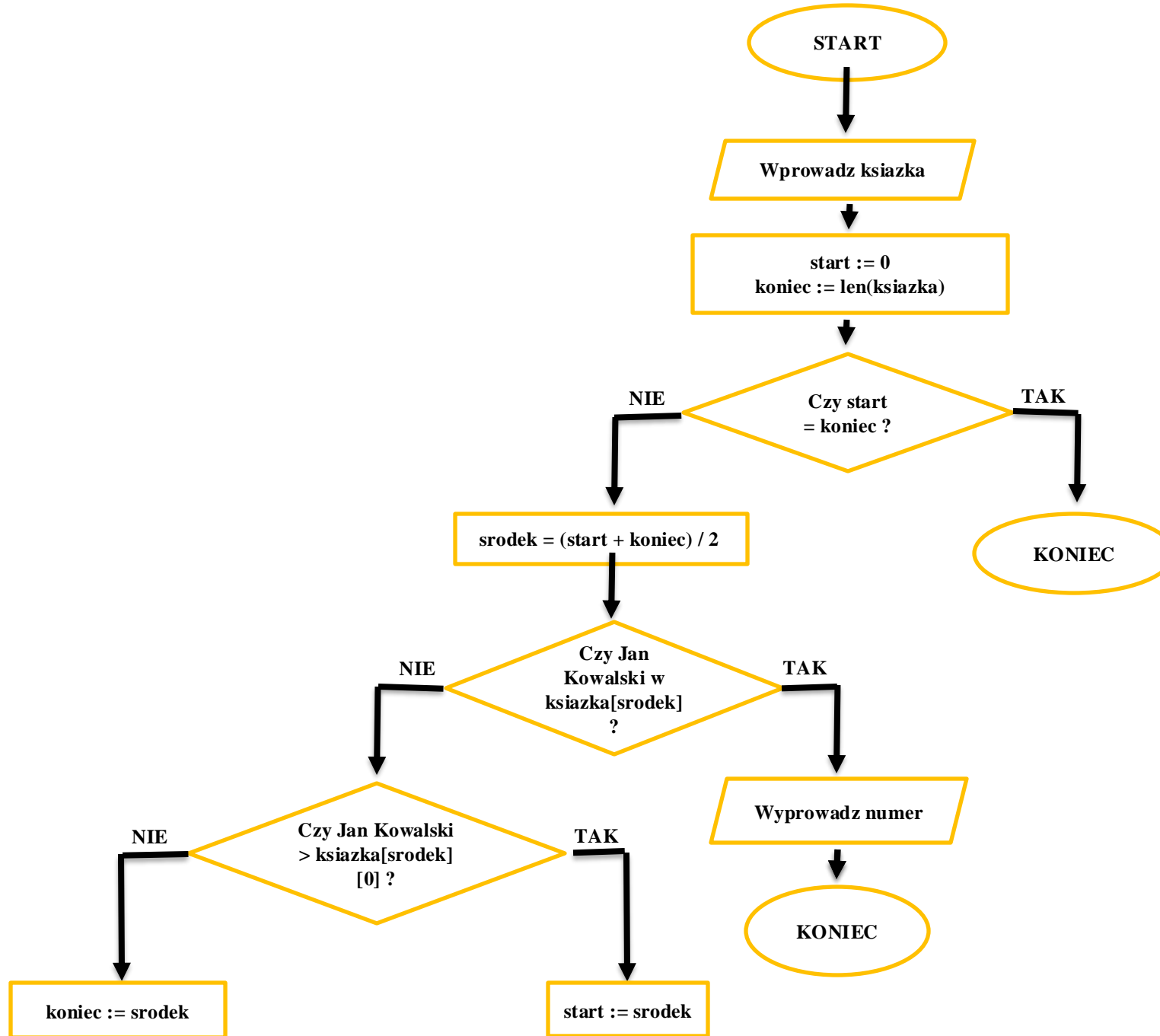


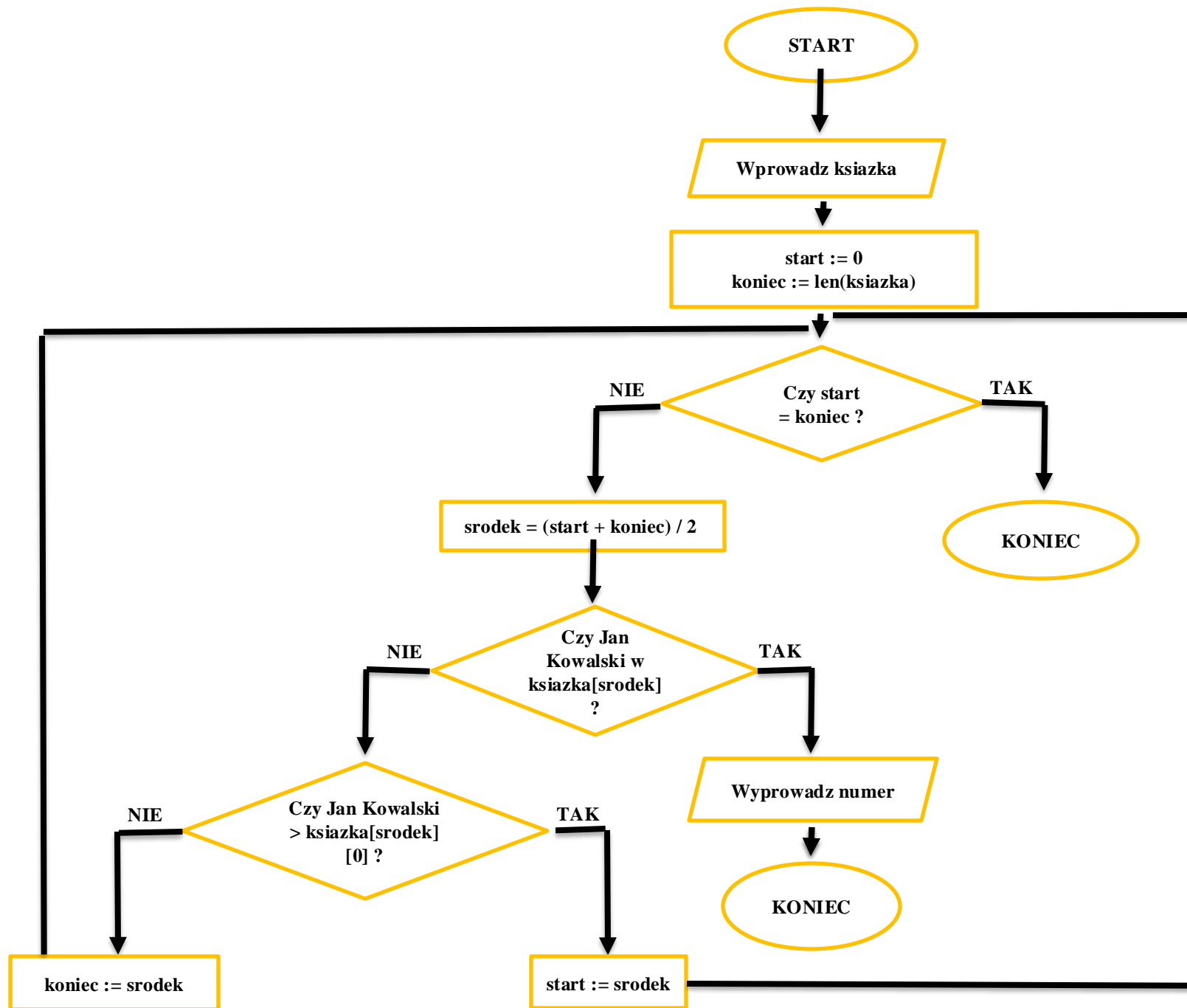














# **Implementacja algorytmu**

(realizacja)

# Definicja



**Język** - zestaw reguł składniowych

**Język programowania** – zestaw reguł składniowych umożliwiający realizację algorytmu.

Algorytm składa się ze zbioru instrukcji elementarnych oraz zestawu instrukcji sterujących. W związku z tym realizacja algorytmu za pomocą wybranego języka jest możliwa, jeżeli ten język implementuje zbiór instrukcji elementarnych oraz zestaw instrukcji sterujących. Taki język nazywamy językiem programowania.

Przykład: Język HTML nie jest nazywany językiem programowania, ponieważ nie posiada instrukcji sterujących. Jest nazywany językiem znaczników.

## Kod maszynowy (binary code)



```
00101110 11110011 00101110 00101110 00101110 00101110 00111100
01010111 01100011 00101110 00101110 00101110 00101110 00101110
00101110 00101110 00101110 00101110 00101110 00101110 00101110
01000000 00101110 00101110 00101110 01110011 00101110 00101110
00101110 00101110 01100100 00101110 00101110 01000111 01001000
01100100 00101110 00101110 01010011 00101000 00101110 00101110
00101110 00101110 01110011 00101110 00101110 00101110 00101110
01001000 01100101 01101100 01101100 01101111 00100000 01010111
01101111 01110010 01101100 01100100 01001110 00101000 00101110
00101110 00101110 00101110 00101000 00101110 00101110 00101110
00101110 00101000 00101110 00101110 00101110 00101110 00101000
00101110 00101110 00101110 00101110 01110011 00101110 00101110
00101110 00101110 01101000 01100101 01101100 01101100 01101111
01110111 01101111 01110010 01101100 01100100 00101110 01110000
01111001 01110100 00101110 00101110 00101110 00101110 00111100
01101101 01101111 01100100 01110101 01101100 01100101 00111110
00101110 00101110 00101110 00101110 01110011 00101110 00101110
00101110 00101110
```

# Assembler



```
01 DATA SEGMENT
02     MESSAGE DB "HELLO WORLD!?!?"
03 ENDS
04
05 CODE SEGMENT
06     ASSUME DS:DATA CS:CODE
07 START:
08     MOV AX,DATA
09     MOV DS,AX
10     LEA DX,MESSAGE
11     MOV AH,9
12     INT 21H
13     MOV AH,4CH
14     INT 21H
15 ENDS
16 END START
17
```

C



```
#include <stdio.h>

int main(void)
{
    printf("hello, world\n");
}
```

Python



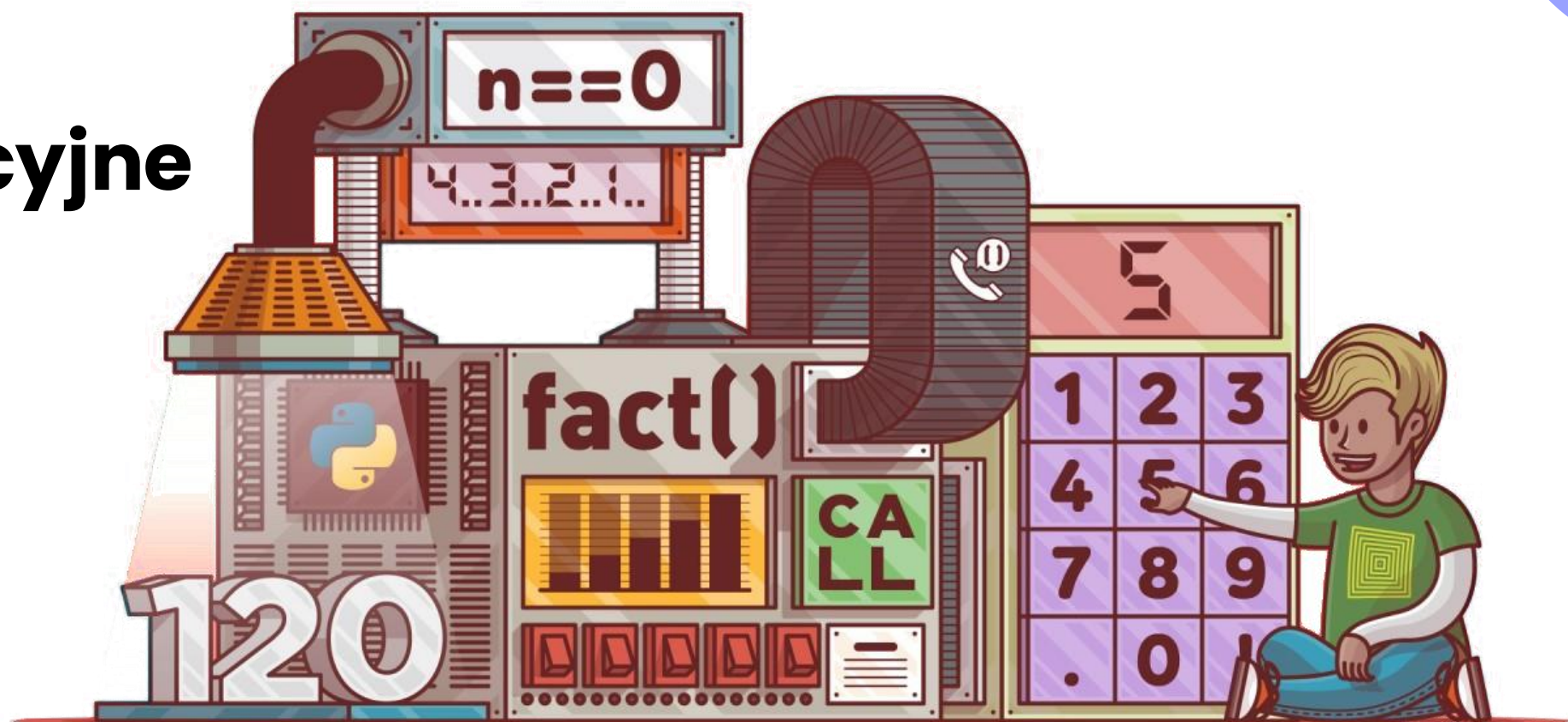
```
print("hello, world")
```



# Scratch



# Funkcje rekurencyjne



Real Python

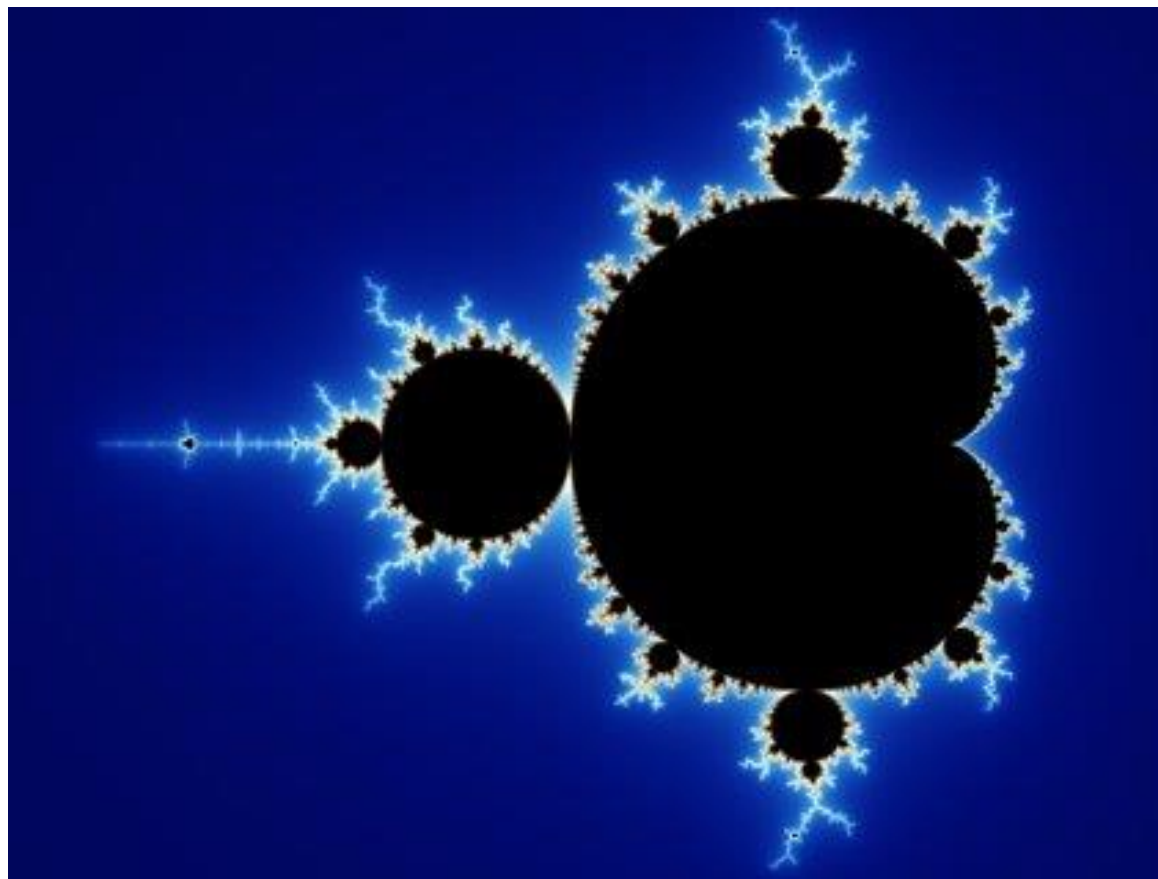
# Rekurencja



# Rekurencja



# Rekurencja







An illustration of a man in a dark suit, light blue shirt, and pink tie standing behind a service counter. He is looking down at a small green card in his hands. In front of him, a queue of four people is waiting, their backs to the viewer. The background is a plain tan wall with vertical lines. Above the man, a small orange rectangular sign is hanging, displaying the text 'x == 1'.

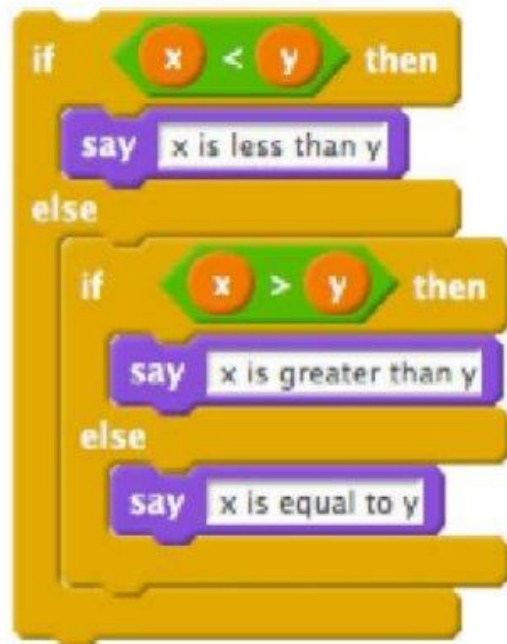
`x == 1`

```
def count(x):  
    if x == 1:  
        return 1  
    else:  
        return x + count(x-1)
```



# **Dodatek: Instrukcje w Scratchu**





# rozgałęzienia (warunki)



# powtórzenia (pętla I)



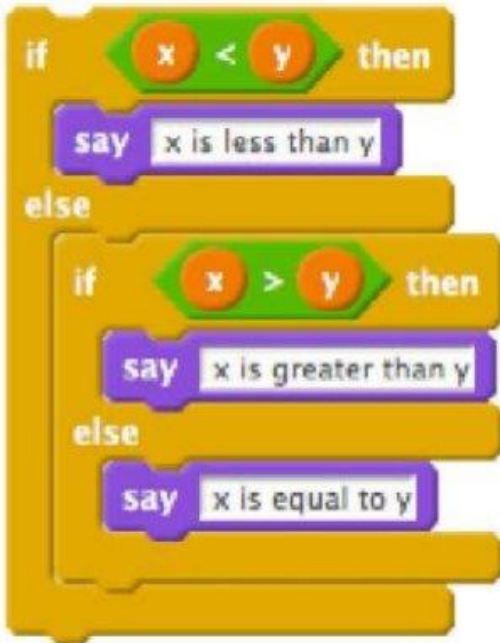
# powtórzenia (pętla II)





# Instrukcje w Pythonie

# rozgałęzienia (warunki)



```
if x < y:  
    print("x jest mniejsze od y")  
elif x > y:  
    print("x jest większe od y")  
else:  
    print("x jest równe y")
```

# powtórzenia (pętla I)



```
while True:  
    print("hello, world")
```

# powtórzenia (pętla II)



```
for i in range(50):  
    print("hello, world")
```

# Dziękujemy!

