Algorytmy

Wykład nr 1 z Podstaw Informatyki

Politechnika Śląska
Robert Tutajewicz
Katedra Informatyki Stosowanej

Pojęcie algorytmu

- Jedno z najważniejszych pojęć informatyki
- Przepis określający działania umożliwiające osiągnięcie pewnego celu (nieścisłe)
- Przykłady: przepis kulinarny (?), sposób rozwiązywania równań kwadratowych z użyciem wyznacznika Δ
- Pierwszy algorytm: NWD Euklidesa (III w. p.n.e.)
- Nazwa pochodzi od matematyka Muhammada al-Chuwarizmi (łac. Algorismus)

Definicja algorytmu

- Brak jednej ogólnie akceptowanej definicji
- A.A. Markow: Precyzyjny przepis definiujący proces obliczeniowy, prowadzący dla różnorodnych danych wejściowych do pożądanego wyniku
- H.S. Stone: Zbiór reguł precyzyjnie określających sekwencję czynności, takich, że każda reguła jest efektywna a sekwencja kończy się w skończonym czasie
- H. Rogers: Efektywna metoda, wyrażona jako skończona lista ściśle określonych instrukcji przeznaczona do obliczania funkcji

Definicja algorytmu – cd.

- D.E. Knuth: Przeprowadzenie systemu od pewnego stanu początkowego poprzez skończoną liczbę stanów pośrednich, do stanu końcowego
- S. Węgrzyn: Zbiór operacji, po wykonaniu których otrzymujemy rozwiązanie dowolnego zadania z pewnej klasy zadań
- M.F. Sipser: Zbiór prostych instrukcji realizujących pewne zadanie
- J. Erickson: Precyzyjna i jednoznaczna sekwencja mechanicznie wykonywanych instrukcji elementarnych

Cechy algorytmu

- Ogólność – może być stosowany dla dowolnego zestawu określonej klasy danych wejściowych
- Skończoność – dla poprawnych danych kończy pracę z poprawnymi wynikami po skończonej liczbie kroków
- Określoność – w każdej chwili na podstawie aktualnego stanu systemu możemy jednoznacznie określić następny krok algorytmu. Tym samym, dla tych samych danych, zawsze mamy ten sam wynik
- Efektywność – „jakość” algorytmu

Sposoby zapisu algorytmu

- Język naturalny
- Funkcje matematyczne
- Schematy blokowe
- Języki formalne (w tym języki programowania)
- Pseudokod

Przykład 1 – język naturalny

Podać algorytm znajdowania wartości $y = Max\{x_i\}$, gdzie $1 \leq i \leq n$

1. $i \leftarrow 1$, idź do 2
2. $y \leftarrow x_i$ idź do 3
3. Czy i = n ? Jeśli tak – koniec, jeśli nie – idź do 4
4. $i \leftarrow i + 1$ idź do 5
5. Czy $x_i > y$? Jeśli tak – idź do 2, jeśli nie – idź o 3

Przykład 1 – realizacja algorytmu

- Ciąg x: 3 4 2 4 5
- Aktualne maksimum y: ?
- Indeks przetwarzanego elementu i: ?

1. i ← 1, idź do 2
2. y ← x_i idź do 3
3. Czy i = n ? Jeśli tak – koniec, jeśli nie – idź do 4
4. i ← i + 1 idź do 5
5. Czy x_i > y ? Jeśli tak – idź do 2, jeśli nie – idź o 3

Przykład 1 – realizacja algorytmu

- Ciąg x: 3 4 2 4 5
- Aktualne maksimum y: ?
- Indeks przetwarzanego elementu i: 1

1. i ← 1, idź do 2
2. y ← x_i idź do 3
3. Czy i = n ? Jeśli tak – koniec, jeśli nie – idź do 4
4. i ← i + 1 idź do 5
5. Czy x_i > y ? Jeśli tak – idź do 2, jeśli nie – idź o 3

Przykład 1 – realizacja algorytmu

- Ciąg x: 3 4 2 4 5
- Aktualne maksimum y: 3
- Indeks przetwarzanego elementu i: 1

1. i ← 1, idź do 2
2. y ← x_i idź do 3
3. Czy i = n ? Jeśli tak – koniec, jeśli nie – idź do 4
4. i ← i + 1 idź do 5
5. Czy x_i > y ? Jeśli tak – idź do 2, jeśli nie – idź o 3

Przykład 1 – realizacja algorytmu

- Ciąg x: 3 4 2 4 5
- Aktualne maksimum y: 3
- Indeks przetwarzanego elementu i: 1

1. i ← 1, idź do 2
2. y ← x_i idź do 3
3. Czy i = n ? Jeśli tak – koniec, jeśli nie – idź do 4
4. i ← i + 1 idź do 5
5. Czy x_i > y ? Jeśli tak – idź do 2, jeśli nie – idź o 3

Przykład 1 – realizacja algorytmu

- Ciąg x: 3 4 2 4 5
- Aktualne maksimum y: 3
- Indeks przetwarzanego elementu i: 2

1. i ← 1, idź do 2
2. y ← x_i idź do 3
3. Czy i = n ? Jeśli tak – koniec, jeśli nie – idź do 4
4. i ← i + 1 idź do 5
5. Czy x_i > y ? Jeśli tak – idź do 2, jeśli nie – idź o 3

Przykład 1 – realizacja algorytmu

- Ciąg x: 3 4 2 4 5
- Aktualne maksimum y: 3
- Indeks przetwarzanego elementu i: 2

1. i ← 1, idź do 2
2. y ← x_i idź do 3
3. Czy i = n ? Jeśli tak – koniec, jeśli nie – idź do 4
4. i ← i + 1 idź do 5
5. Czy x_i > y ? Jeśli tak – idź do 2, jeśli nie – idź o 3

Przykład 1 – realizacja algorytmu

- Ciąg x: 3 4 2 4 5
- Aktualne maksimum y: 4
- Indeks przetwarzanego elementu i: 2

1. i ← 1, idź do 2
2. y ← x_i idź do 3
3. Czy i = n ? Jeśli tak – koniec, jeśli nie – idź do 4
4. i ← i + 1 idź do 5
5. Czy x_i > y ? Jeśli tak – idź do 2, jeśli nie – idź o 3

Przykład 1 – realizacja algorytmu

- Ciąg x: 3 4 2 4 5
- Aktualne maksimum y: 4
- Indeks przetwarzanego elementu i: 2

1. i ← 1, idź do 2
2. y ← x_i idź do 3
3. Czy i = n ? Jeśli tak – koniec, jeśli nie – idź do 4
4. i ← i + 1 idź do 5
5. Czy x_i > y ? Jeśli tak – idź do 2, jeśli nie – idź o 3

Przykład 1 – realizacja algorytmu

- Ciąg x: 3 4 2 4 5
- Aktualne maksimum y: 4
- Indeks przetwarzanego elementu i: 3

1. i ← 1, idź do 2
2. y ← x_i idź do 3
3. Czy i = n ? Jeśli tak – koniec, jeśli nie – idź do 4
4. i ← i + 1 idź do 5
5. Czy x_i > y ? Jeśli tak – idź do 2, jeśli nie – idź o 3

Przykład 1 – realizacja algorytmu

- Ciąg x: 3 4 2 4 5
- Aktualne maksimum y: 4
- Indeks przetwarzanego elementu i: 3

1. i ← 1, idź do 2
2. y ← x_i idź do 3
3. Czy i = n ? Jeśli tak – koniec, jeśli nie – idź do 4
4. i ← i + 1 idź do 5
5. Czy x_i > y ? Jeśli tak – idź do 2, jeśli nie – idź o 3

Przykład 1 – realizacja algorytmu

- Ciąg x: 3 4 2 4 5
- Aktualne maksimum y: 4
- Indeks przetwarzanego elementu i: 3

1. i ← 1, idź do 2
2. y ← x_i idź do 3
3. Czy i = n ? Jeśli tak – koniec, jeśli nie – idź do 4
4. i ← i + 1 idź do 5
5. Czy x_i > y ? Jeśli tak – idź do 2, jeśli nie – idź o 3

Przykład 1 – realizacja algorytmu

- Ciąg x: 3 4 2 4 5
- Aktualne maksimum y: 4
- Indeks przetwarzanego elementu i: 4

1. i ← 1, idź do 2
2. y ← x_i idź do 3
3. Czy i = n ? Jeśli tak – koniec, jeśli nie – idź do 4
4. i ← i + 1 idź do 5
5. Czy x_i > y ? Jeśli tak – idź do 2, jeśli nie – idź o 3

Przykład 1 – realizacja algorytmu

- Ciąg x: 3 4 2 4 5
- Aktualne maksimum y: 4
- Indeks przetwarzanego elementu i: 4

1. i ← 1, idź do 2
2. y ← x_i idź do 3
3. Czy i = n ? Jeśli tak – koniec, jeśli nie – idź do 4
4. i ← i + 1 idź do 5
5. Czy x_i > y ? Jeśli tak – idź do 2, jeśli nie – idź o 3

Przykład 1 – realizacja algorytmu

- Ciąg x: 3 4 2 4 5
- Aktualne maksimum y: 4
- Indeks przetwarzanego elementu i: 4

1. i ← 1, idź do 2
2. y ← x_i idź do 3
3. Czy i = n ? Jeśli tak – koniec, jeśli nie – idź do 4
4. i ← i + 1 idź do 5
5. Czy x_i > y ? Jeśli tak – idź do 2, jeśli nie – idź o 3

Przykład 1 – realizacja algorytmu

- Ciąg x: 3 4 2 4 5
- Aktualne maksimum y: 4
- Indeks przetwarzanego elementu i: 5

1. i ← 1, idź do 2
2. y ← x_i idź do 3
3. Czy i = n ? Jeśli tak – koniec, jeśli nie – idź do 4
4. i ← i + 1 idź do 5
5. Czy x_i > y ? Jeśli tak – idź do 2, jeśli nie – idź o 3

Przykład 1 – realizacja algorytmu

- Ciąg x: 3 4 2 4 5
- Aktualne maksimum y: 4
- Indeks przetwarzanego elementu i: 5

1. i ← 1, idź do 2
2. y ← x_i idź do 3
3. Czy i = n ? Jeśli tak – koniec, jeśli nie – idź do 4
4. i ← i + 1 idź do 5
5. Czy x_i > y ? Jeśli tak – idź do 2, jeśli nie – idź o 3

Przykład 1 – realizacja algorytmu

- Ciąg x: 3 4 2 4 5
- Aktualne maksimum y: 5
- Indeks przetwarzanego elementu i: 5

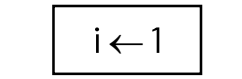
1. i ← 1, idź do 2
2. y ← x_i idź do 3
3. Czy i = n ? Jeśli tak – koniec, jeśli nie – idź do 4
4. i ← i + 1 idź do 5
5. Czy x_i > y ? Jeśli tak – idź do 2, jeśli nie – idź o 3

Przykład 1 – realizacja algorytmu

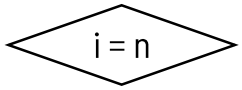
- Ciąg x: 3 4 2 4 5
- Aktualne maksimum y: 5
- Indeks przetwarzanego elementu i: 5

1. i ← 1, idź do 2
2. y ← x_i idź do 3
3. Czy i = n ? Jeśli tak – koniec, jeśli nie – idź do 4
4. i ← i + 1 idź do 5
5. Czy x_i > y ? Jeśli tak – idź do 2, jeśli nie – idź o 3

Zapis algorytmu za pomocą schematów blokowych

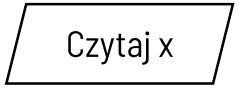


Blok operacyjny (obliczenia)



Blok decyzyjny (testowanie warunku)

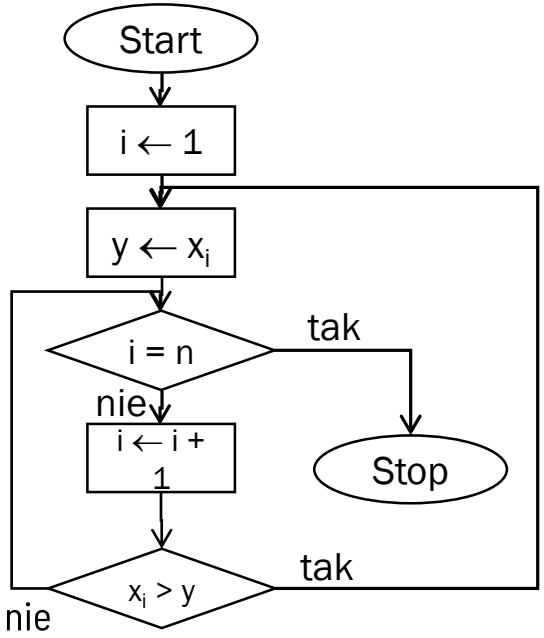
Blok graniczny (początek, koniec)



Blok wejścia/wyjścia

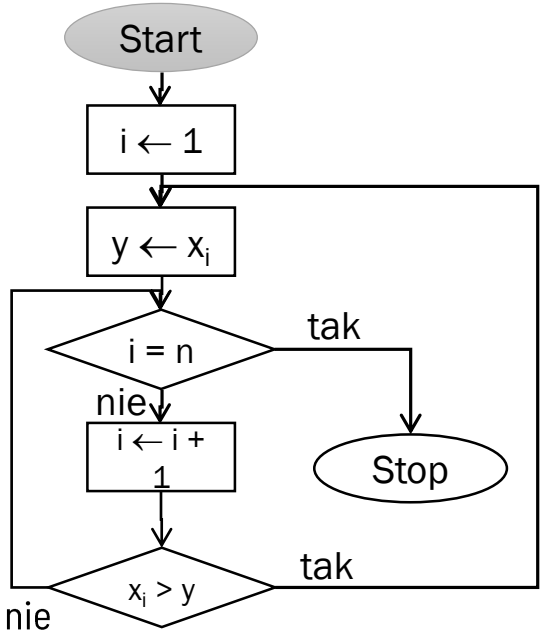
Strzałka (powiązania między blokami)

Przykład 1 – schemat blokowy



1. i ← 1, idź do 2
2. y ← xi, idź do 3
3. Czy i = n ? Jeśli tak – koniec, jeśli nie – idź do 4
4. i ← i + 1, idź do 5
5. Czy xi > y ? Jeśli tak – idź do 2, jeśli nie – idź do 3

Przykład 1 – schemat blokowy



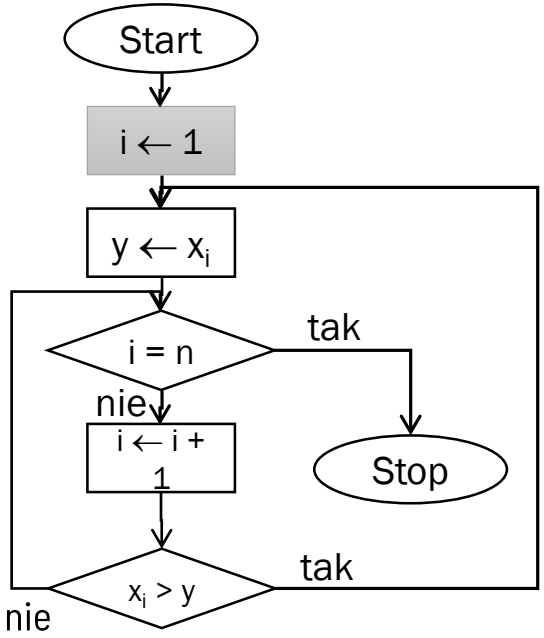
Ciąg x: 3 4 2

y = ?

i = ?

n = 3

Przykład 1 – schemat blokowy



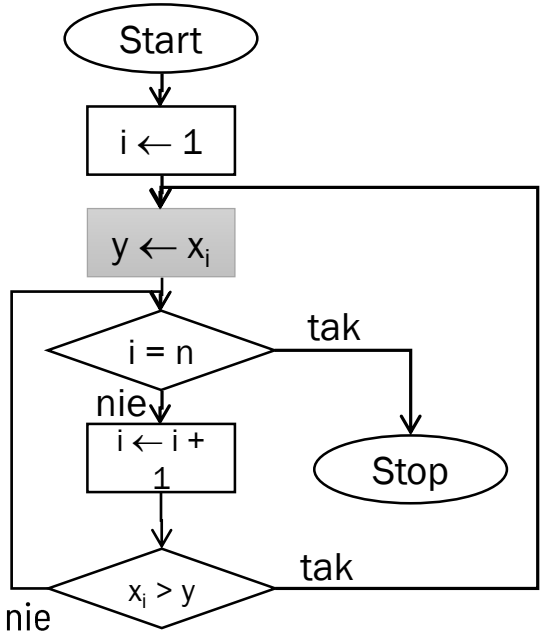
Ciąg x: 3 4 2

y = ?

i = 1

n = 3

Przykład 1 – schemat blokowy



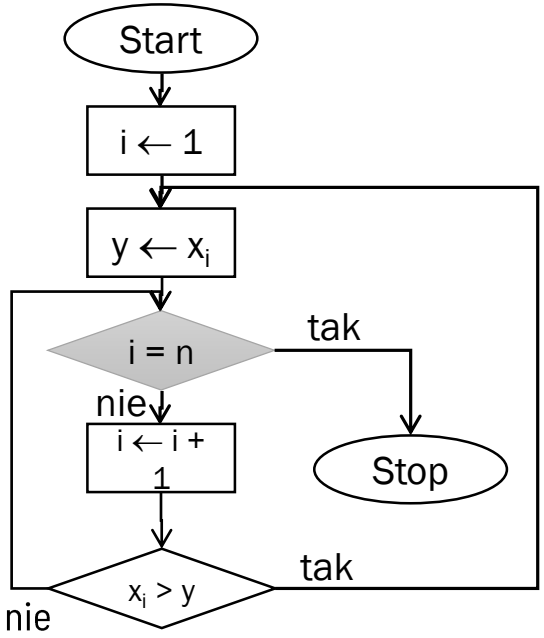
Ciąg x: 3 4 2

y = 3

i = 1

n = 3

Przykład 1 – schemat blokowy



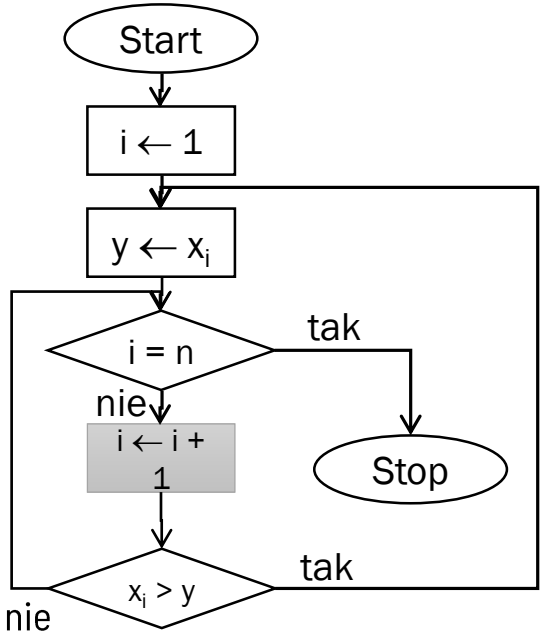
Ciąg x: 3 4 2

y = 3

i = 1

n = 3

Przykład 1 – schemat blokowy



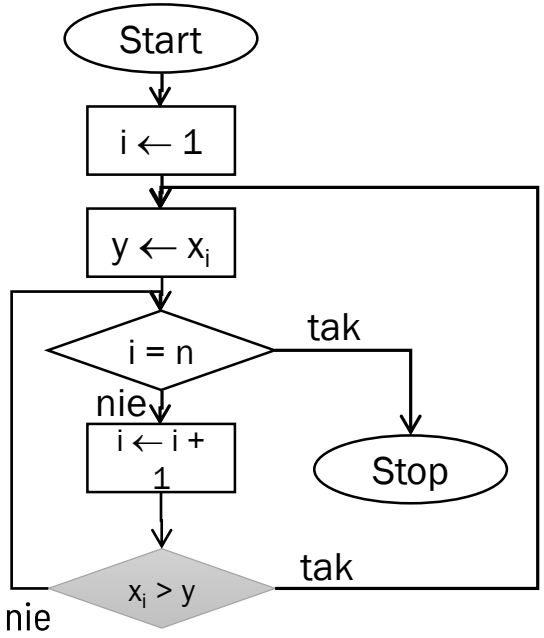
Ciąg x: 3 4 2

y = 3

i = 2

n = 3

Przykład 1 – schemat blokowy



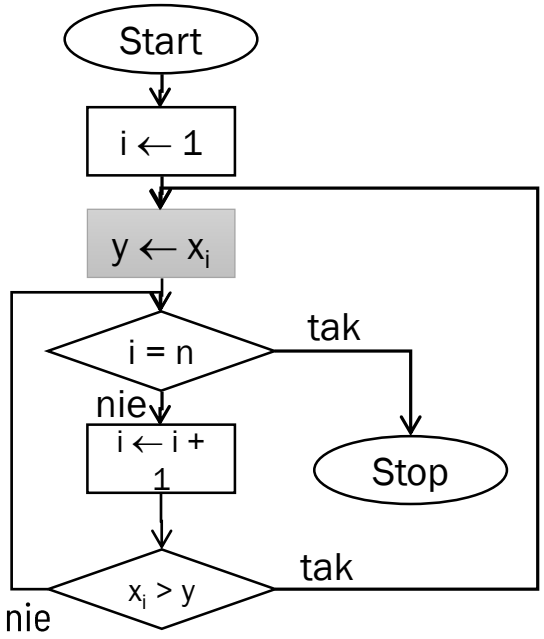
Ciąg x: 3 4 2

y = 3

i = 2

n = 3

Przykład 1 – schemat blokowy



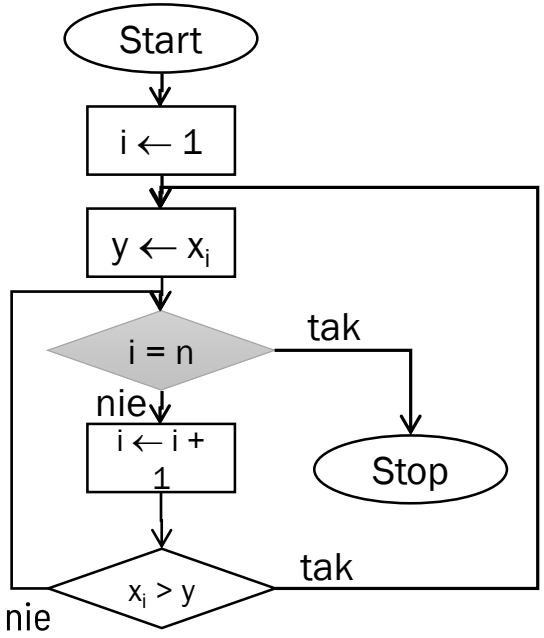
Ciąg x: 3 4 2

y = 4

i = 2

n = 3

Przykład 1 – schemat blokowy



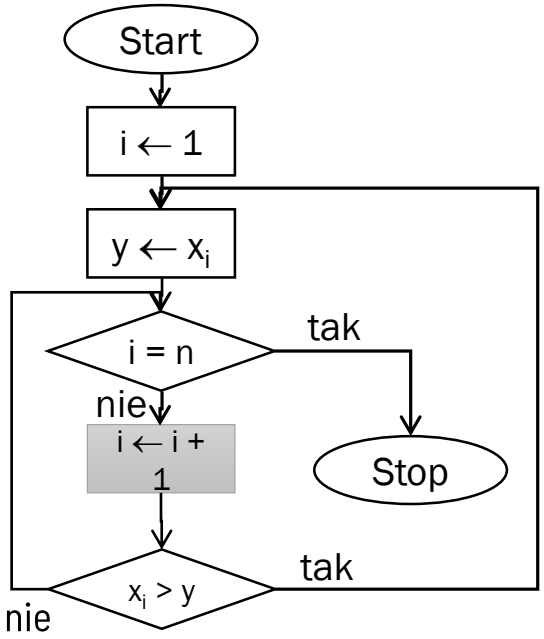
Ciąg x: 3 4 2

y = 4

i = 2

n = 3

Przykład 1 – schemat blokowy



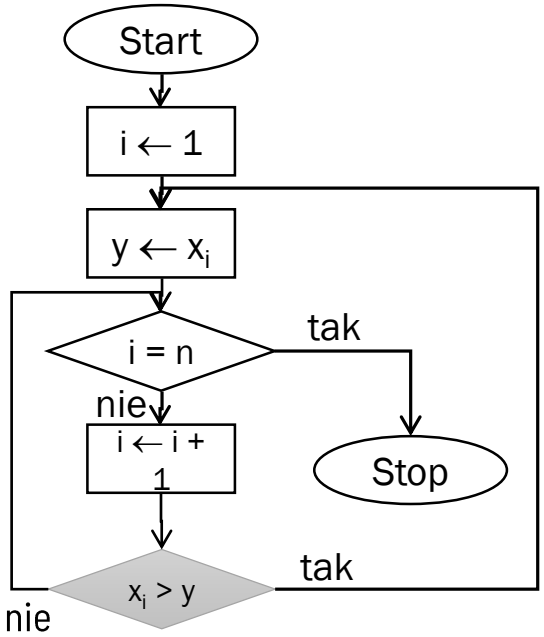
Ciąg x: 3 4 2

y = 4

i = 3

n = 3

Przykład 1 – schemat blokowy



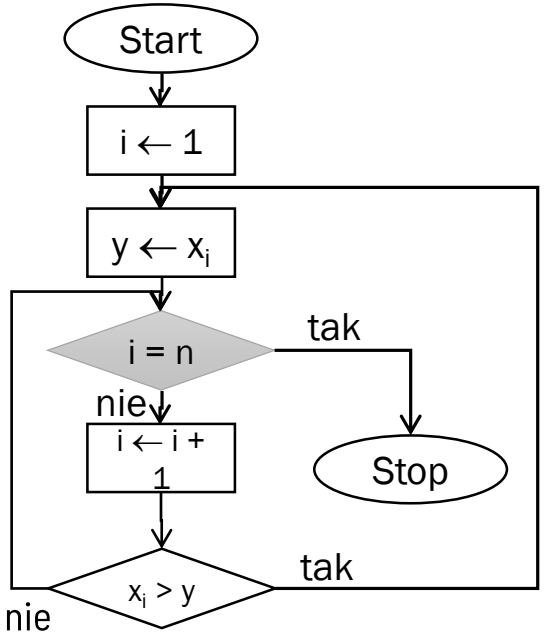
Ciąg x: 3 4 2

y = 4

i = 3

n = 3

Przykład 1 – schemat blokowy



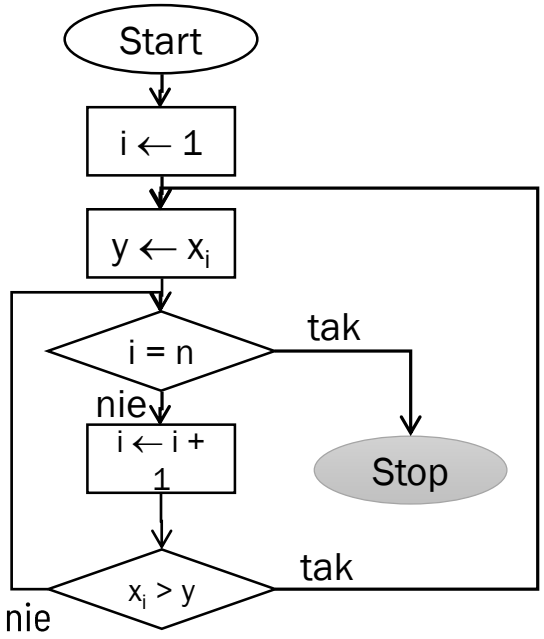
Ciąg x: 3 4 2

y = 4

i = 3

n = 3

Przykład 1 – schemat blokowy



Ciąg x: 3 4 2

y = 4

i = 3

n = 3

Przykład 1 – język programowania (Pascal)

```
function Max( x : array of integer ) : integer;
var
  i, y : integer;
begin
  y := x[ 1 ];
  for i := 2 to n do
    if x[ i ] > y then
      y := x[ i ];
  Max := y;
end;
```

Przykład 1 – język programowania (C)

```
int Max( int *x )
{
  int i;
  int y = x[0];
  for( i = 1; i < n; i++ )
  if( x[i] > y )
    y = x[i];
  return y;
}
```

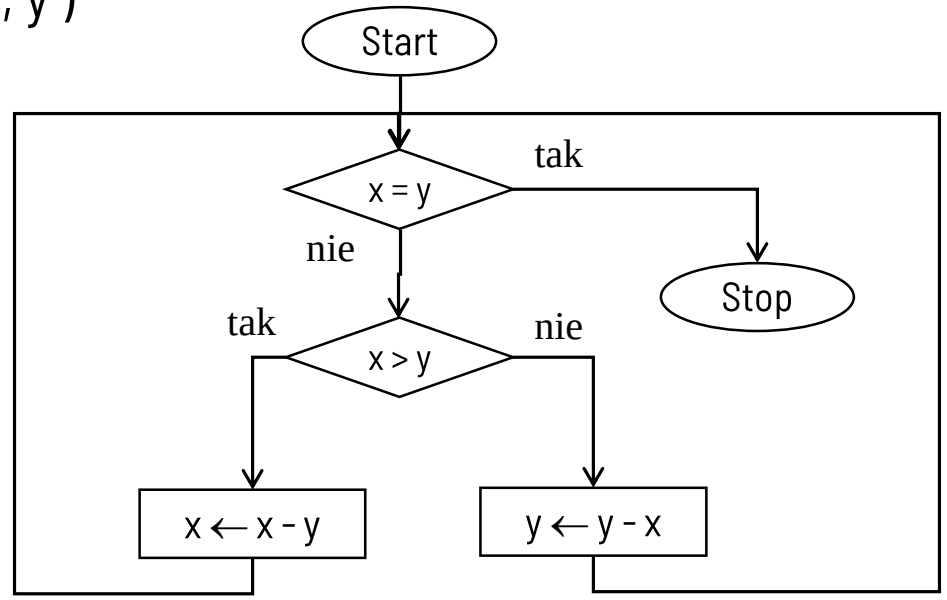
Przykład 1 – język programowania (Python)

```
def Max(x):
    y = x[0]
    for i in range(len(x)):
        if x[i] > y:
            y = x[i]
    return y
```

Przykład 2 – algorytm Euklidesa

Gdy x > y:

NWD(x, y) = NWD(x – y, y)



Algorytm Euklidesa – cechy

- Ogólność – oblicza NWD dla dowolnej pary liczb naturalnych x, y
- Skończoność – w każdym przebiegu pętli zmniejsza się wartość jednej ze zmiennych, więc w końcu (w najgorszym przypadku) $x = y = 1$
- Określoność – wszystkie operacje jednoznacznie zdefiniowane
- Efektywność – na następnym wykładzie

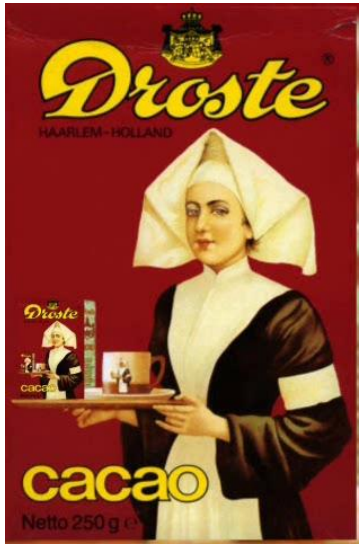
Przykład 2 – algorytm Euklidesa

$$NWD(x, y) = NWD(x - y, y) \text{ gdy } x > y$$
$$NWD(x, y) = x = y \text{ gdy } x = y$$
$$NWD(x, y) = NWD(y - x, x) \text{ gdy } x < y$$

```
int NWD( int x, y )
{
  if ( x == y ) return x;
  else if ( x > y ) return NWD( x - y, y );
  else return NWD( x, y - x );
};
```

Rekurencja

W logice, matematyce i programowaniu odwoływanie się np. funkcji lub definicji do samej siebie (łac. *recurrere* – przybiec z powrotem)



Ciąg Fibonacciego

- Ciąg, którego pierwsze 2 wyrazy wynoszą 1 a kolejne powstają wg rekurencyjnej formuły: F(n) = F(n – 2) + F(n – 1)
- Początkowe wyrazy ciągu: 1 1 2 3 5 8 13 21 34 55 89 144 233 …

```
int Fib( int n)
{
  if ( n < 3 ) return 1;
  else
    return Fib( n - 2 ) + Fib( n – 1 );
};
```

Problem Collatza

- Weźmy dowolną liczbę naturalną $c_0 > 0$
- Jeśli jest ona parzysta, to $c_1 = c_0 / 2$
- Jeśli jest nieparzysta, to $c_1 = 3 \cdot c_0 + 1$
- Następnie z c_1 postępujemy tak jak z c_0 i kontynuujemy ten proces
- Np. dla $c_0 = 11$ mamy: 11, 34, 17, 52, 26, 13, 40, 20, 10, 5, 16, 8, 4, 2, 1, 4, 2, 1, 4, 2, 1
- Dla $c_0 = 15$ mamy: 15, 46, 23, 70, 35, 106, 53, 160, 80, 40, 20, 10, 5, 16, 8, 4, 2, 1
- Czy zawsze w końcu osiągniemy 1 ?

Skończoność algorytmów –problem stopu

• Czy poniższy „algorytm” zakończy się dla dowolnego n?

```
void collatz( int n )
{
  do {
    if ( n % 2 == 0 ) n = n / 2;
    else n = 3 * n + 1;
  while ( n > 1 );
};
```

(tak dla $20 \cdot 2^{58} \approx 5.764 \cdot 10^{18}$)

Pal Erdős: *mathematics is not yet ready for such problems*

Algorytmy szeregowe

- Instrukcje tworzące algorytm wykonywane są sekwencyjnie, jedna po drugiej

$n+1$ – liczba operacji w algorytmie

O_i – operacja i-ta

t(x) – chwila czasu kiedy jest wykonywana operacja x

$$\forall_i \; t(\text{koniec}(O_i)) \leq t(\text{początek}(O_{i+1}))$$

Algorytmy równoległe

- Co najmniej 2 instrukcje mogą być wykonywane równolegle

$n+1$ – liczba operacji w algorytmie

O_i – operacja i-ta

t(x) – chwila czasu kiedy jest wykonywana operacja x

$$\exists_i \; t\big(\text{koniec}(O_i)\big) > t(\text{początek}(O_{i+1}))$$

Przykład algorytmu równoległego

Opracować algorytm wyznaczający całkowite pierwiastki wielomianu

$$W(x) = x^n + a_{n-1}x^{n-1} + .. + a_2x^2 + a_1x + a_0 = 0$$

gdzie a_0, a_1, .., a_{n-1} są całkowite dodatnie.

$$x(x^{n-1} + a_{n-1}x^{n-2} + .. + a_2x + a_1) = -a_0$$

Jeżeli istnieje całkowity pierwiastek W(x) to musi on być dzielnikiem wyrazu wolnego a_0

Przykład 3 – ciąg dalszy

- Znajdź dzielniki wyrazu wolnego a_0
- Sprawdź, które z nich są pierwiastkami wielomianu

- Np. równanie $x^2 + x - 2 = 0$
- Dzielniki wyrazu wolnego: -2, -1, 1, 2

$$(-2)^2+(-2) - 2 = 4 - 2 - 2 = 0$$

-2 jest pierwiastkiem

Przykład 3 – ciąg dalszy

- Znajdź dzielniki wyrazu wolnego a_0
- Sprawdź, które z nich są pierwiastkami wielomianu

- Np. równanie $x^2 + x - 2 = 0$
- Dzielniki wyrazu wolnego: -2, -1, 1, 2

$$(-1)^2+(-1) - 2 = 1 - 1 - 2 = -2$$

-1 nie jest pierwiastkiem

Przykład 3 – ciąg dalszy

- Znajdź dzielniki wyrazu wolnego a_0
- Sprawdź, które z nich są pierwiastkami wielomianu

- Np. równanie $x^2 + x - 2 = 0$
- Dzielniki wyrazu wolnego: -2, -1, 1, 2

$$1^2 + 1 - 2 = 1 + 1 - 2 = 0$$

1 jest pierwiastkiem

Przykład 3 – ciąg dalszy

- Znajdź dzielniki wyrazu wolnego a_0
- Sprawdź, które z nich są pierwiastkami wielomianu

- Np. równanie $x^2 + x - 2 = 0$
- Dzielniki wyrazu wolnego: -2, -1, 1, 2

$$2^2 + 2 - 2 = 4 + 2 - 2 = 4$$

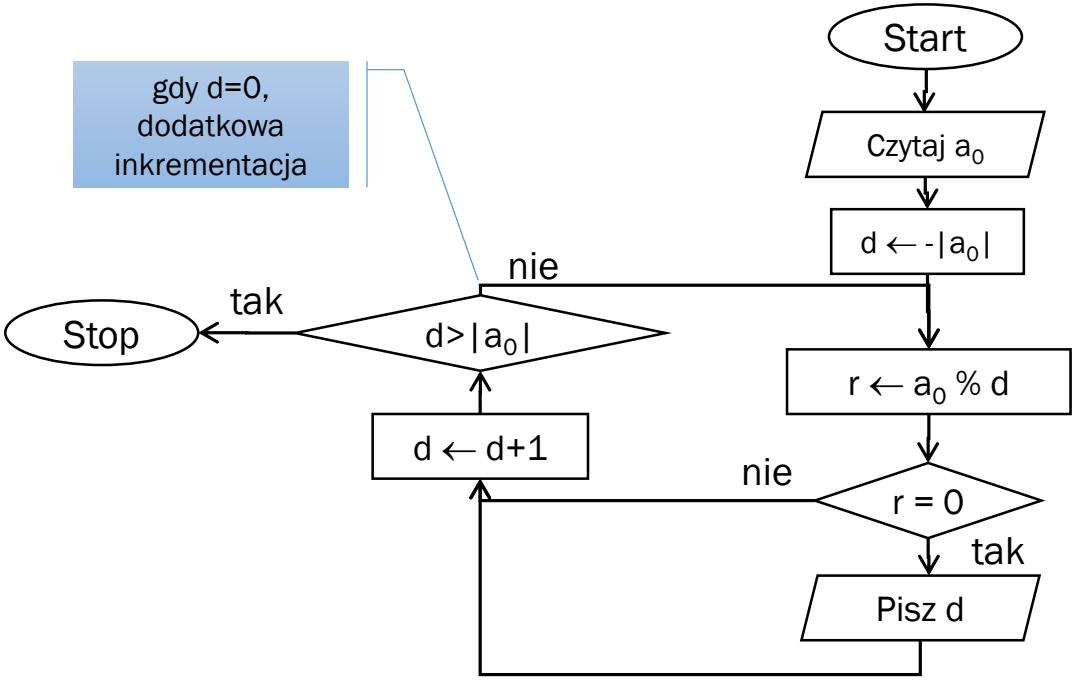
2 nie jest pierwiastkiem

Przykład 3 – ciąg dalszy

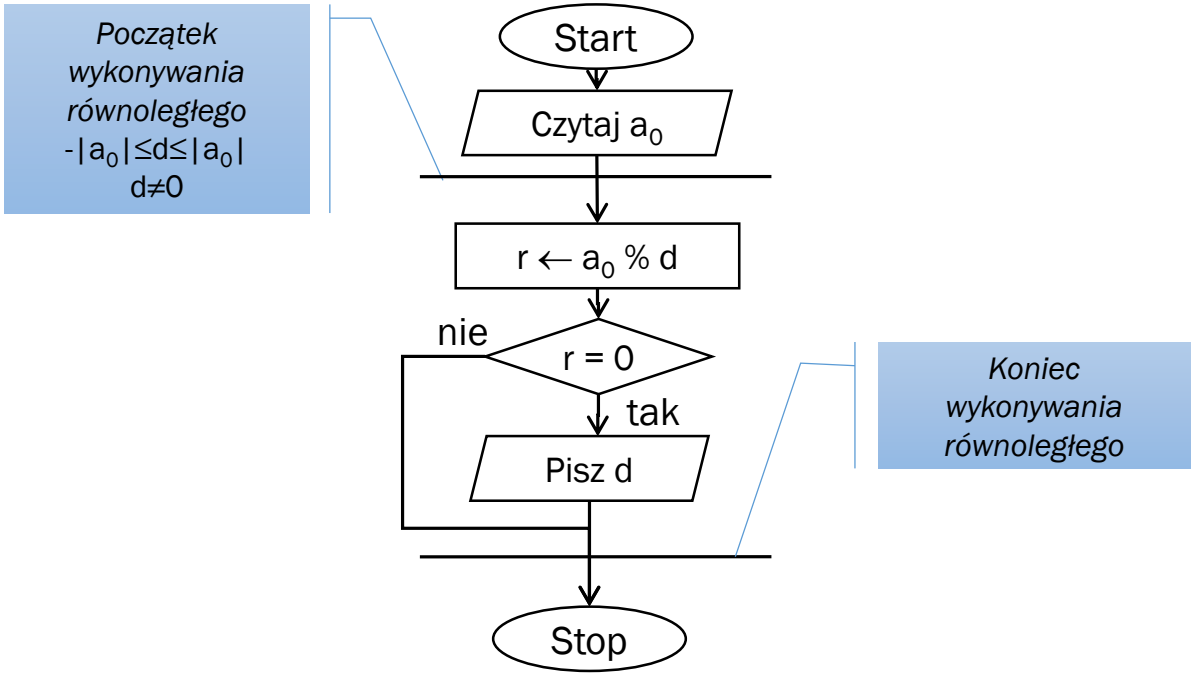
- Znajdź dzielniki wyrazu wolnego a_0
- Sprawdź, które z nich są pierwiastkami wielomianu

- Np. równanie $x^2 + x - 2 = 0$
- Dzielniki wyrazu wolnego: -2, -1, 1, 2
- Pierwiastki: -2, 1
- Problem: jak znaleźć dzielniki wyrazu wolnego ?

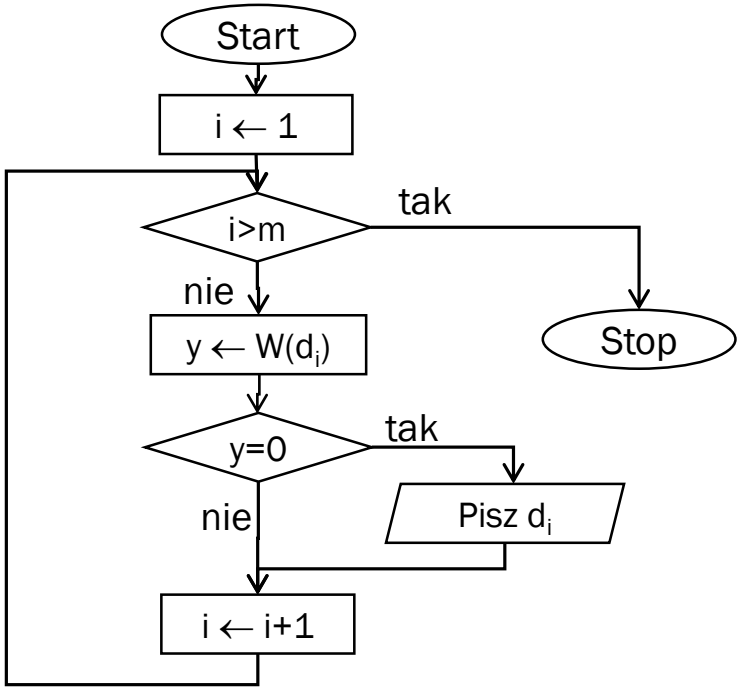
Znajdowanie dzielników a_0



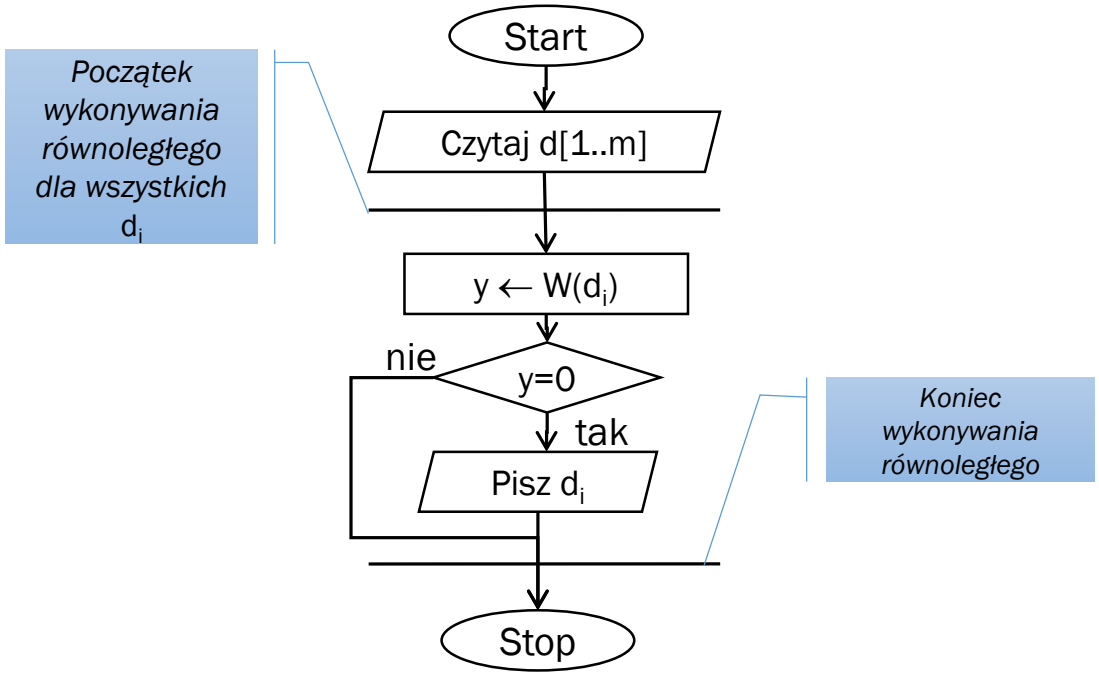
Znajdowanie dzielników a_0 – równolegle



Szukanie pierwiastków wśród dzielników a_0 – szeregowo



Szukanie pierwiastków wśród dzielników a_0 – równolegle



Podsumowanie

- Definicja algorytmu i jego cechy
- Sposoby zapisu algorytmów
- Przykłady algorytmów
- Algorytmy szeregowe i równoległe

Algorytmy

Wykład nr 1 z Podstaw Informatyki

Politechnika Śląska
Robert Tutajewicz
Katedra Informatyki Stosowanej