PCEN
Doskonalimy z pasją!

Algorytmy, programowanie i myślenie komputacyjne

wykład

Algorytm

- **Algorytm** - w matematyce oraz informatyce to **skończony**, **ciąg** **jasno zdefiniowanych** **czynności**, **koniecznych** **do wykonania pewnego rodzaju** **zadań** (np. obliczeniowych) zwanych problemami.
- Algorytm ma przeprowadzić określony stan rzeczy **z pewnego stanu początkowego** do **pożądanego stanu wynikowego** **w sposób powtarzalny** tj. dla tego samego stanu początkowego musi zawsze dawać ten sam stan wynikowy (właściwość ta nie dotyczy tzw. algorytmów probabilistycznych).
- Oznacza to, że problem musi być rozwiązany za pomocą danego algorytmu z wykorzystaniem dostępnych zasobów (np. obliczeniowych) **w skończonym czasie**.

Algorytm – cechy oczywiste

Jakkolwiek algorytm musi zapewnić osiągnięcie stanu wynikowego (inaczej: rozwiązania) **poprzez wykonanie czynności w skończonej liczbie kroków** (a więc też w skończonym czasie)**, to:**

- skończona liczba kroków nie oznacza, że z góry wiadomo po ilu krokach algorytm zakończy działanie;
- komunikat o błędzie lub braku rozwiązania też jest jednym z możliwych poprawnych zakończeń realizacji algorytmu!
- określenie kolejności wykonywania czynności jest krytyczne dla osiągnięcia rozwiązania;
- musi istnieć mechanizm rozgałęziania algorytmu, tj. decydowania o kolejności w trakcie wykonywania algorytmu na podstawie zaistniałych warunków.

Dalsze cechy algorytmów

Pod względem poprawności występują:

1. Algorytmy poprawne (są też częściowo poprawne)
2. Algorytmy częściowo poprawne (nie są poprawne)
3. Algorytmy niepoprawne (te, które nie są częściowo poprawne)

Dokładne znaczenie powyższych pojęć **nie jest potoczne**. Ich zdefiniowanie wymaga wprowadzenia:

- dziedziny i przeciwdziedziny algorytmicznej
- kryterium stopu algorytmu

Cechy algorytmu

- **Jednoznaczność**
- **Skończoność**
- **Kompletność**
- **Stabilność**

Miary efektywności algorytmu

- **Złożoność pamięciowa**
- **Złożoność czasowa**

Informacje, dane, struktury danych

- *Informacja* = byt abstrakcyjny bez definicji (pojęcie pierwotne)
- *Dana* = fizyczna reprezentacja informacji
- *Struktura danych* = sposób ulokowania lub właściwości informacji w *danych* determinujący m.in. zakres operacji przetwarzania (działania) jakie na *danej* można wykonać

Przykładowe struktury danych

- rekord lub struktura (ang. *record*, *struct*), logiczny odpowiednik to krotka
- tablica
- lista
- stos
- kolejka
- drzewo i jego liczne odmiany (np. drzewo binarne)
- graf
- kopiec

Zmienne jako dane

- Zmienna = jednostka pamięci identyfikowana nazwą do której można wprowadzać różne *informacje* (wartości) o ustalonej (lub nie) *strukturze danych*

Zmienna to pojęcie powszechnie stosowane w matematyce i w jej nauczaniu od pierwszych klas szkoły podstawowej !

Fundamentalne instrukcje algorytmiczne

- **Instrukcja przypisania wartości do zmiennej**
- **Instrukcja warunkowa**
- **Instrukcja pętli**

Ponadto dla algorytmów wykonywanych autonomicznie (np. przez komputer):

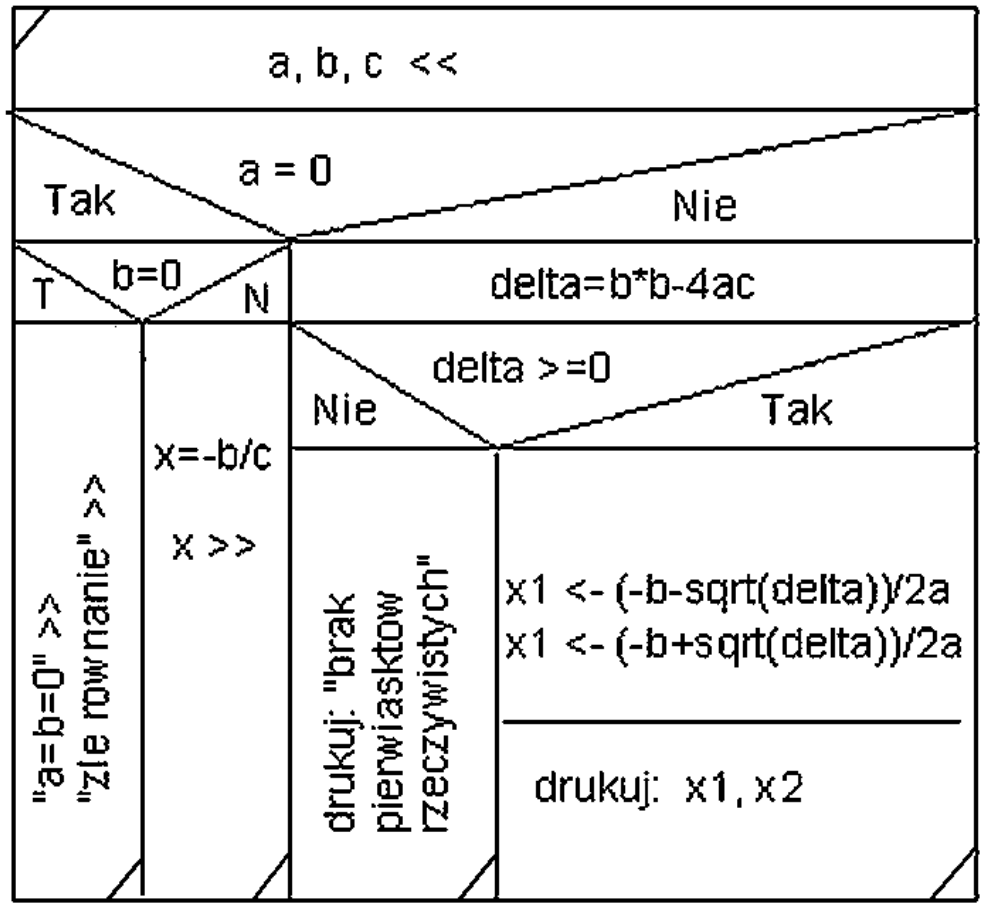
- **Instrukcja wprowadzenia wartości zmiennej - instrukcja wejścia**
- **Instrukcja wyprowadzenia wartości zmiennej - instrukcja wyjścia**

Mechanizm składania (konkatenacji) instrukcji.

Notacje algorytmów

- W języku naturalnym
- W języku symbolicznym
- W postaci graficznej np.
  - Schematy blokowe (strzałkowe)
  - Schematy blokowe Nassi-Schneidermana (N-S; na bazie prostokątu)

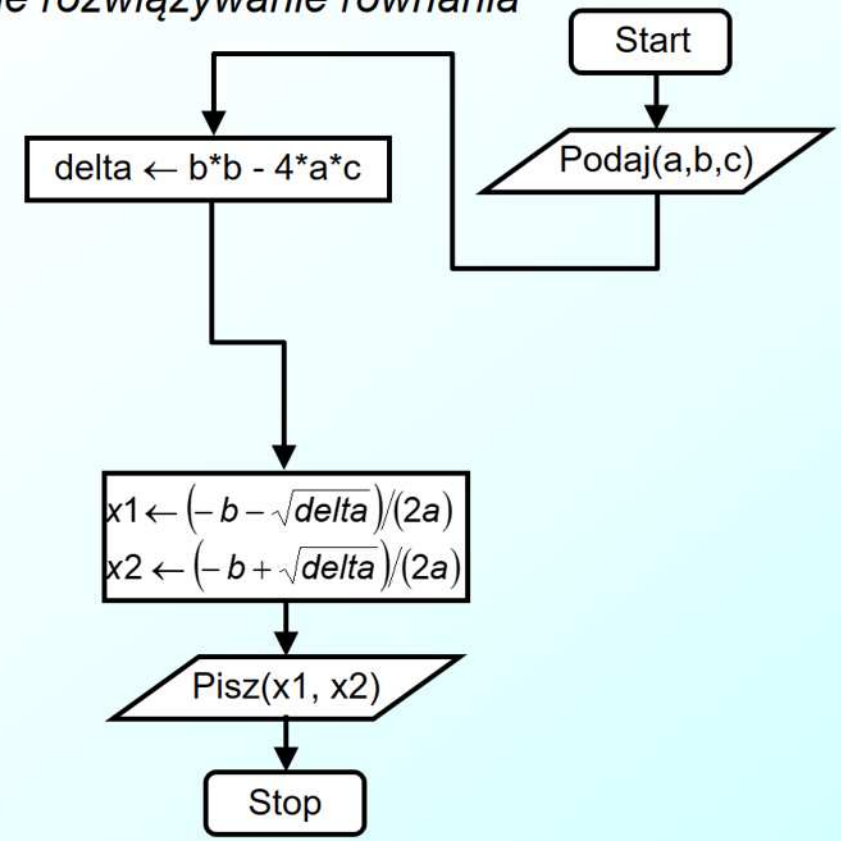
Przykład notacji N-S



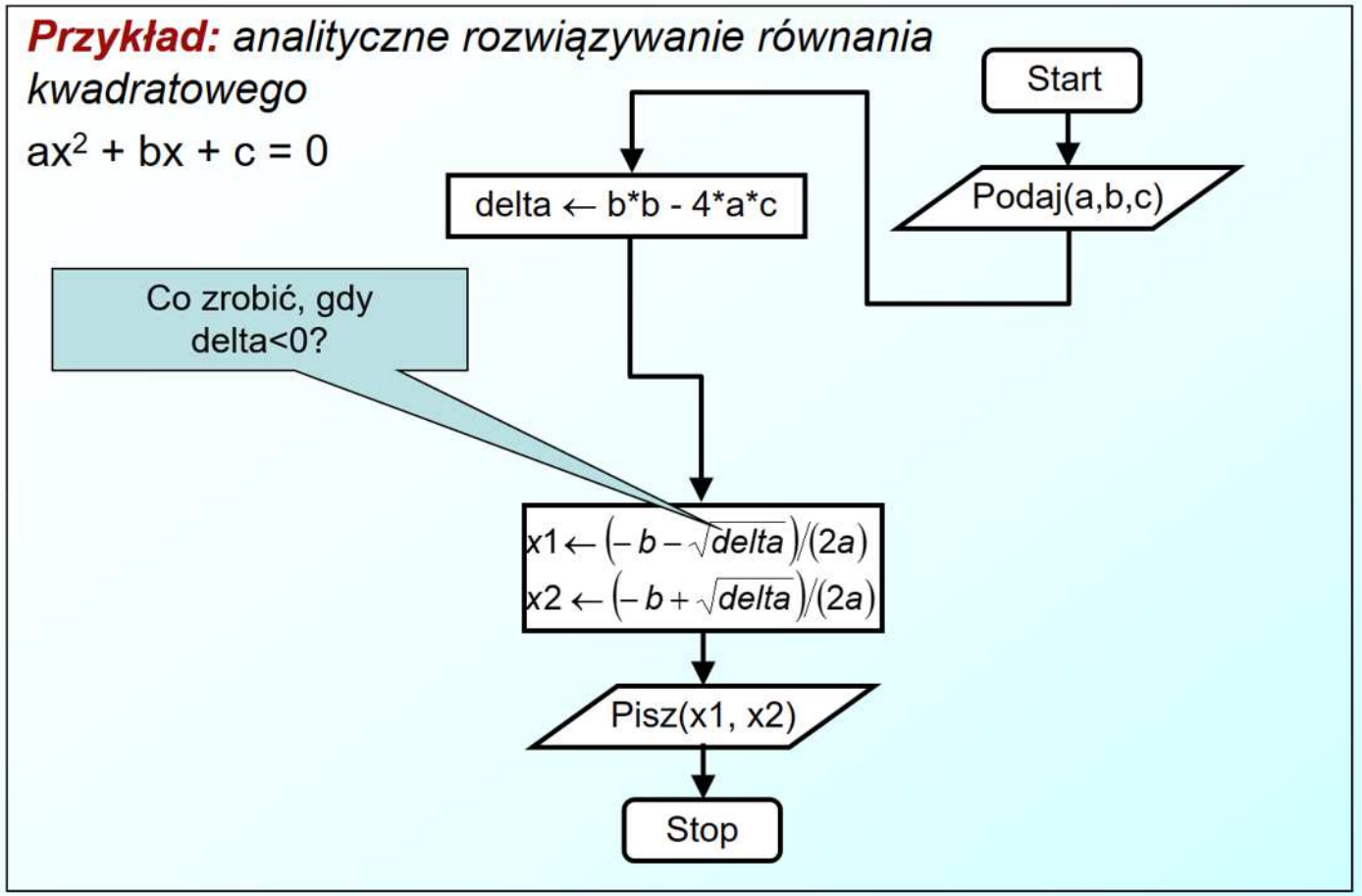
Przykład algorytmu – kompletność cz.1

Przykład: *analityczne rozwiązywanie równania kwadratowego*

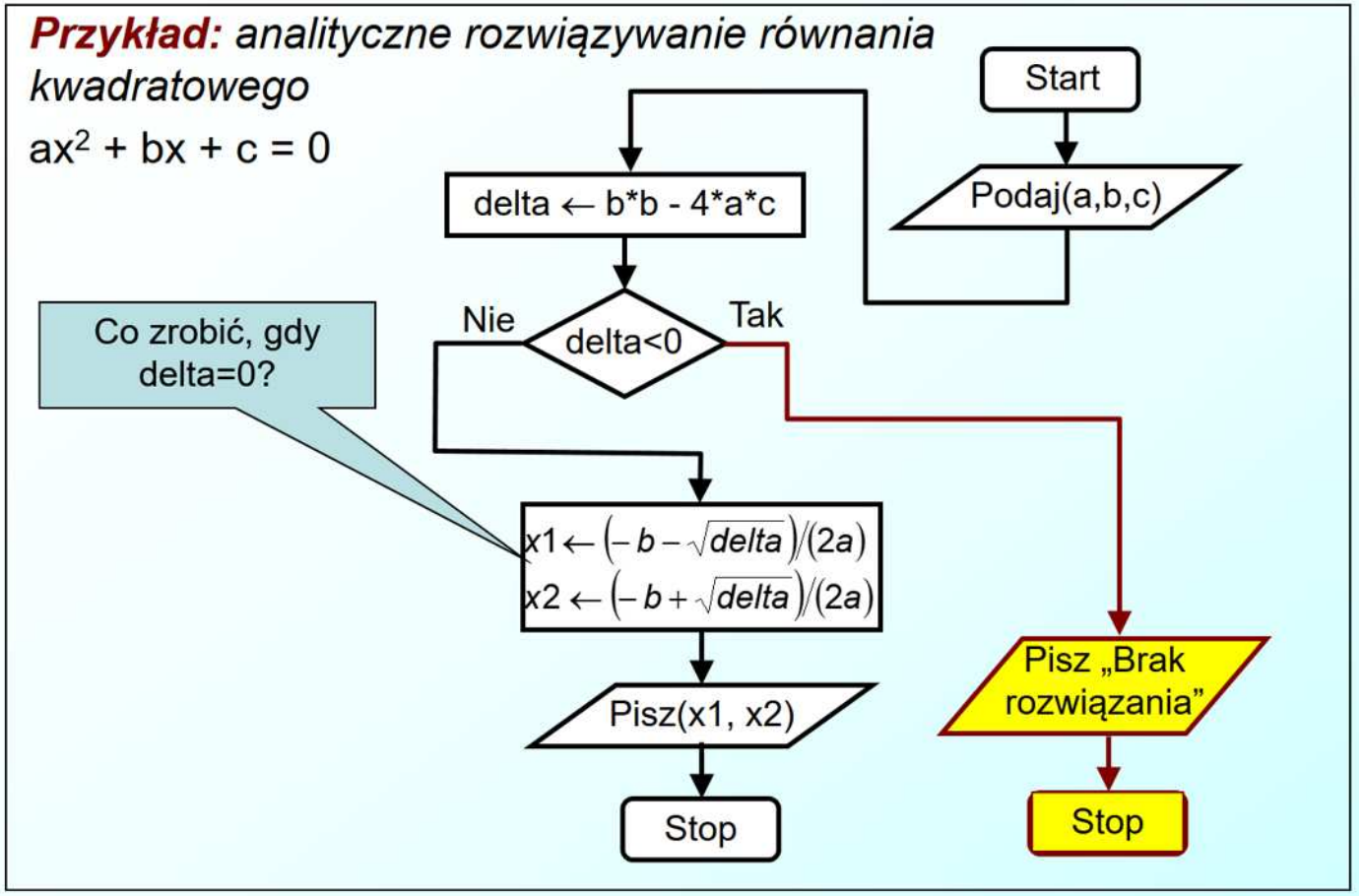
$ax^2 + bx + c = 0$



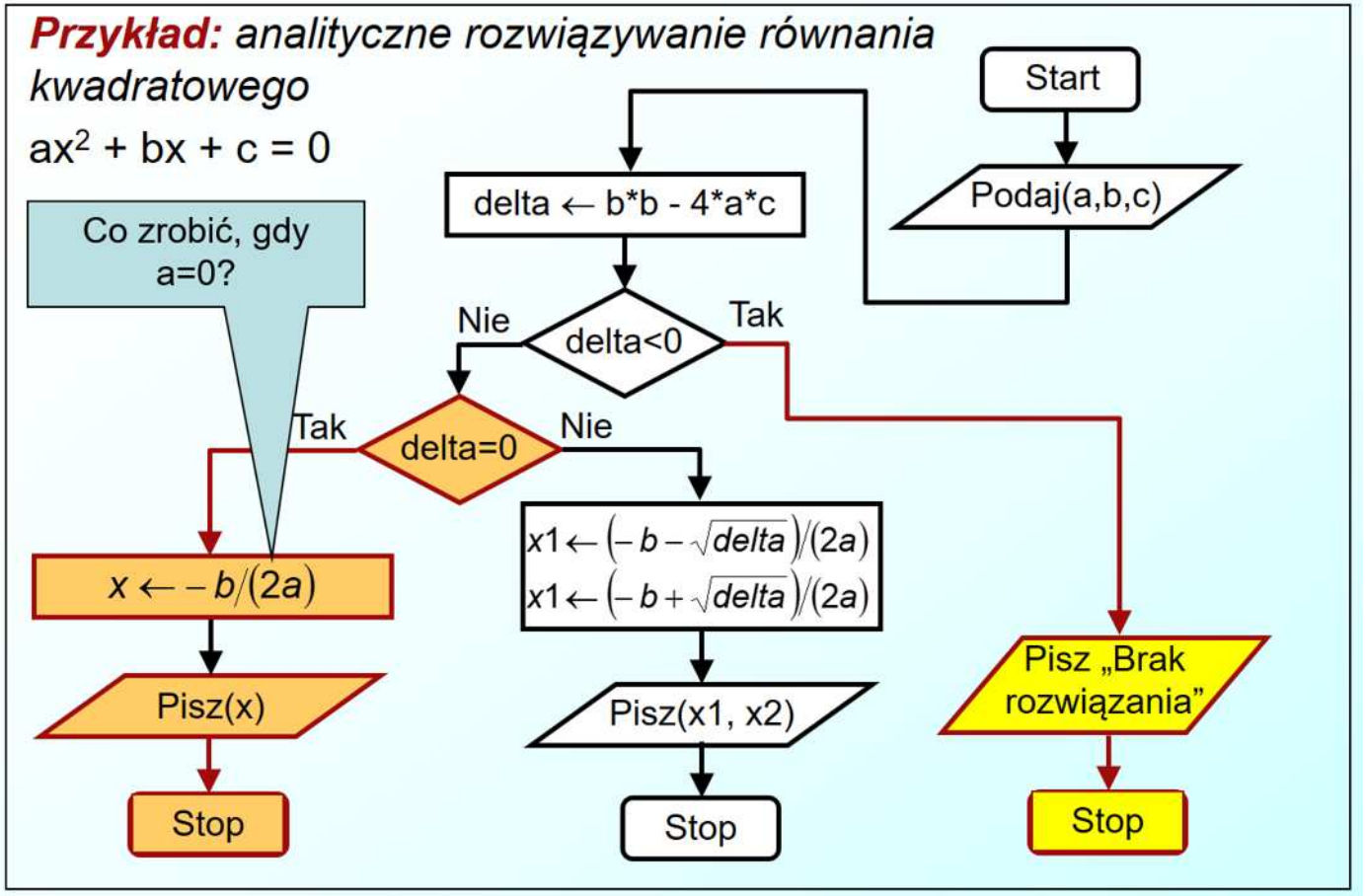
Przykład algorytmu – kompletność cz.1



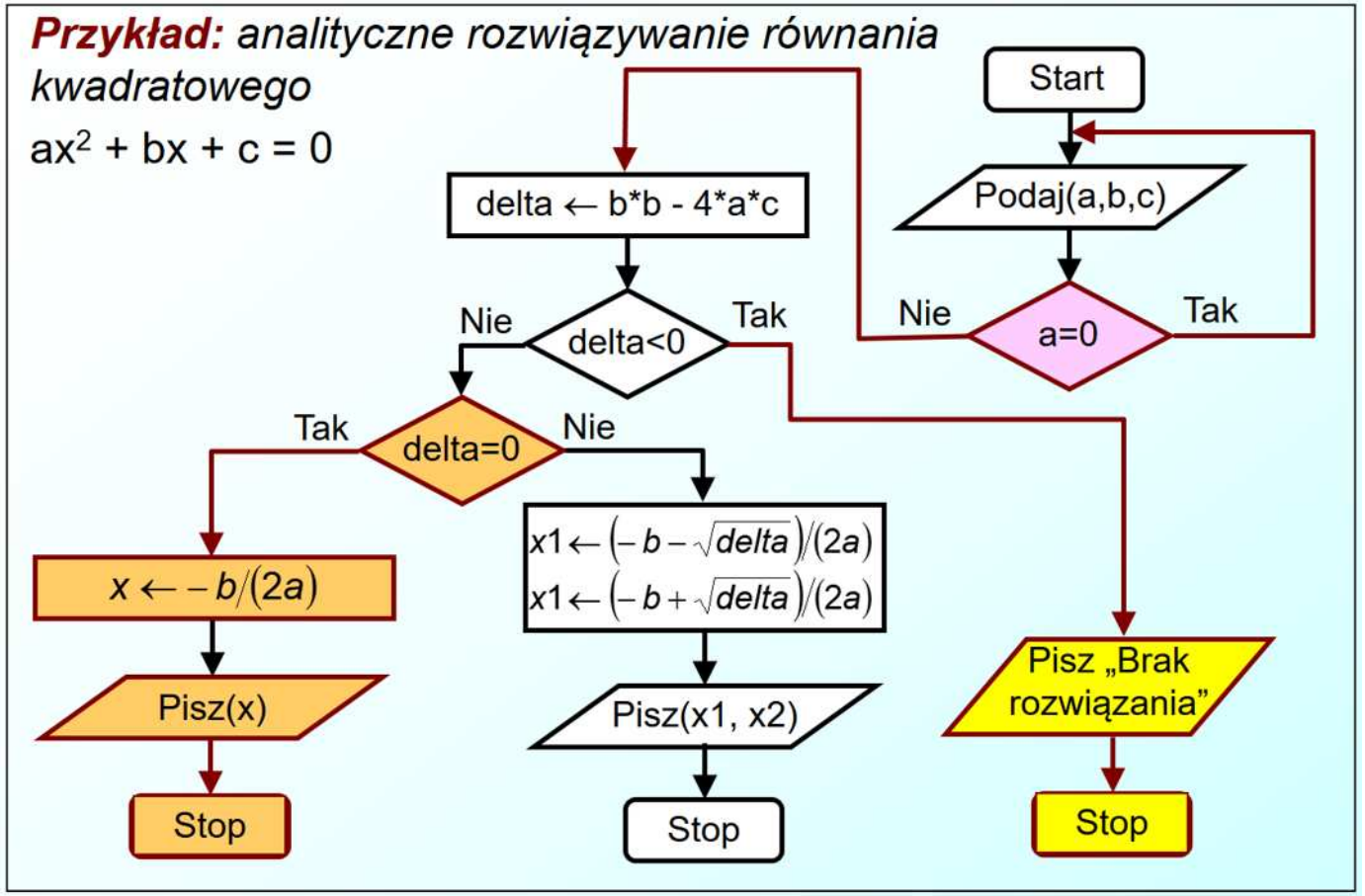
Przykład algorytmu – kompletność cz.2



Przykład algorytmu – kompletność cz.3



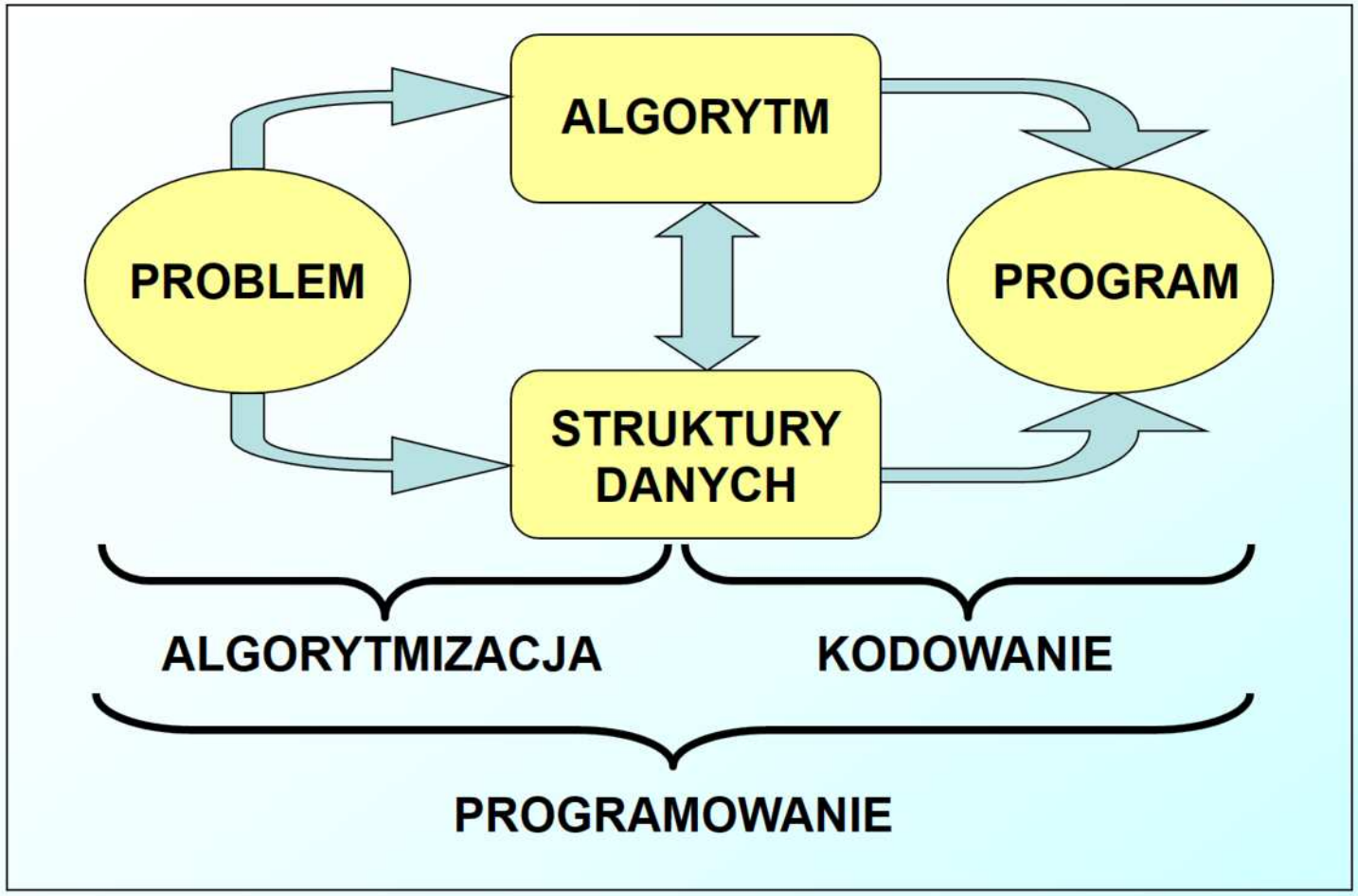
Przykład algorytmu – kompletność cz.4



Algorytmika i algorytmizacja

- **Algorytmika** to **nauka o algorytmach** (m.in. zajmuje się badaniem własności algorytmów).
- Algorytmika jest potężnym działem informatyki, a także służy, dla większości nauk matematyczno-przyrodniczych, ekonomii i techniki.
- Częścią algorytmiki jest **algorytmizacja**, czyli **proces budowy konkretnego algorytmu**.

Programowanie informatyczne = algorytmizacja + kodowanie



Zmiana systemu zapisu liczby

W – wartość liczbowa; p>=2 – podstawa systemu pozycyjnego

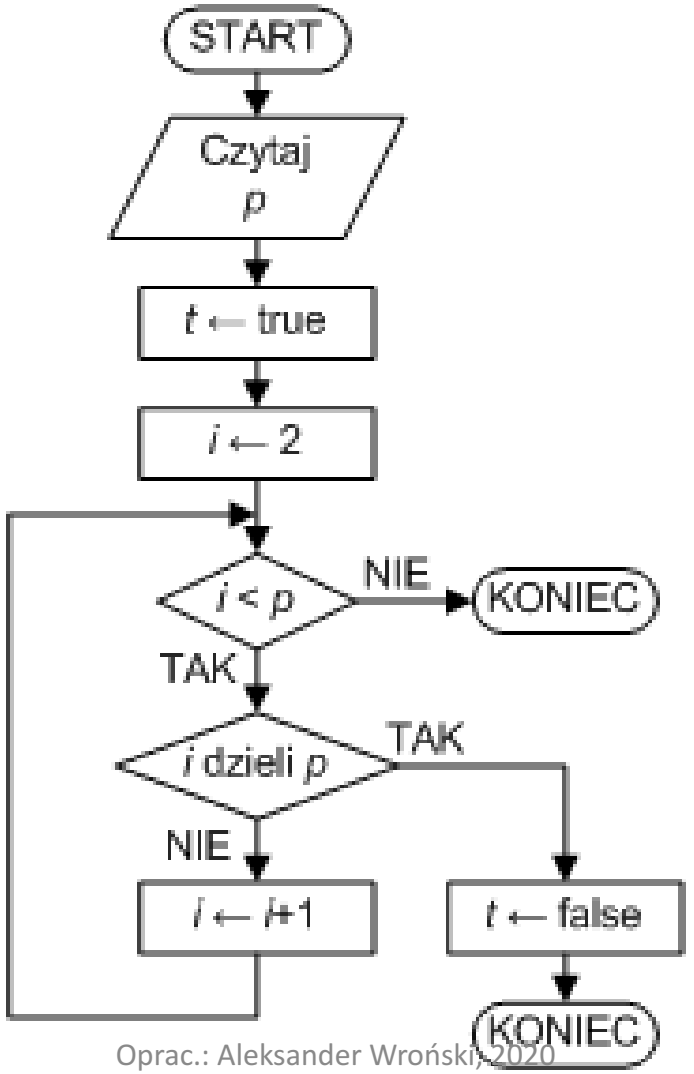
Napis<-””

Dopóki W>0 dopóty:

- Napis<- znak(reszta z W div p) + Napis
- W<- W div p

gdzie „div” to operacja dzielenia całkowitego liczb

Liczba pierwsza – algorytm naiwny



Algorytm naiwny pierwszości rekurencyjnie

Założenie: P – badana na pierwszość liczba; p>=2

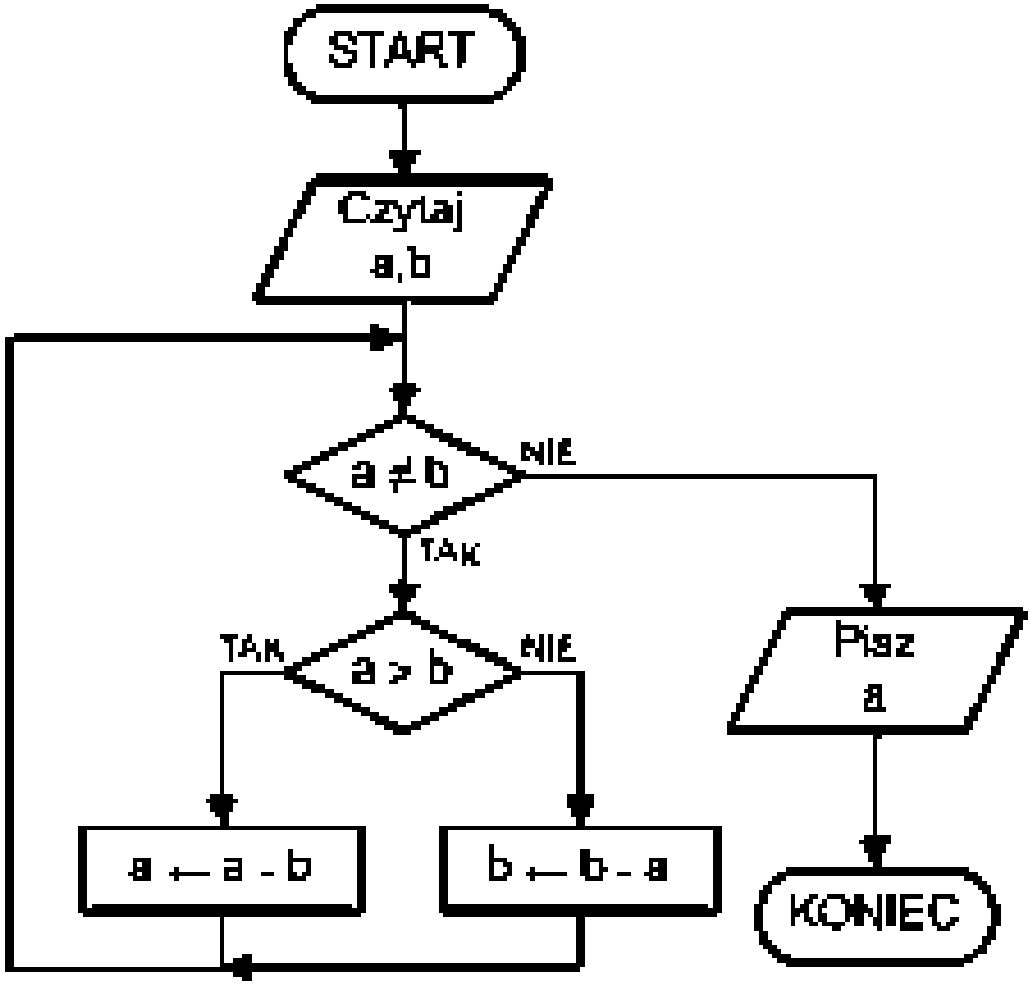
- Funkcja MaDzielnik(n): {n>=2}
  - jeżeli n=2 to MaDzielnik<-(reszta z P div n =0) w p.p.
    - jeżeli (reszta z P div n=0) to MaDzielnik<-prawda w p.p. MaDzielnik<- MaDzielnik(n-1);

Badanie pierwszości P:

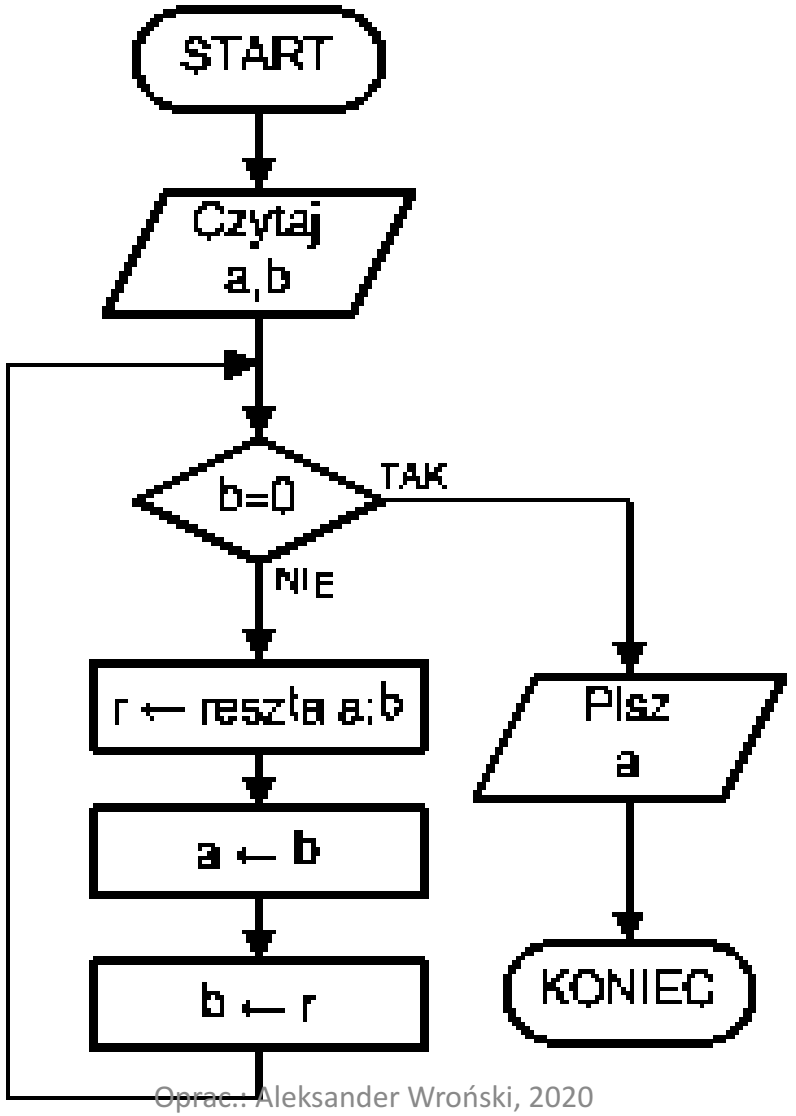
Wczytaj P; P>=2.

- jeżeli MaDzielnik(P-1)=prawda to „P jest liczbą pierwszą”

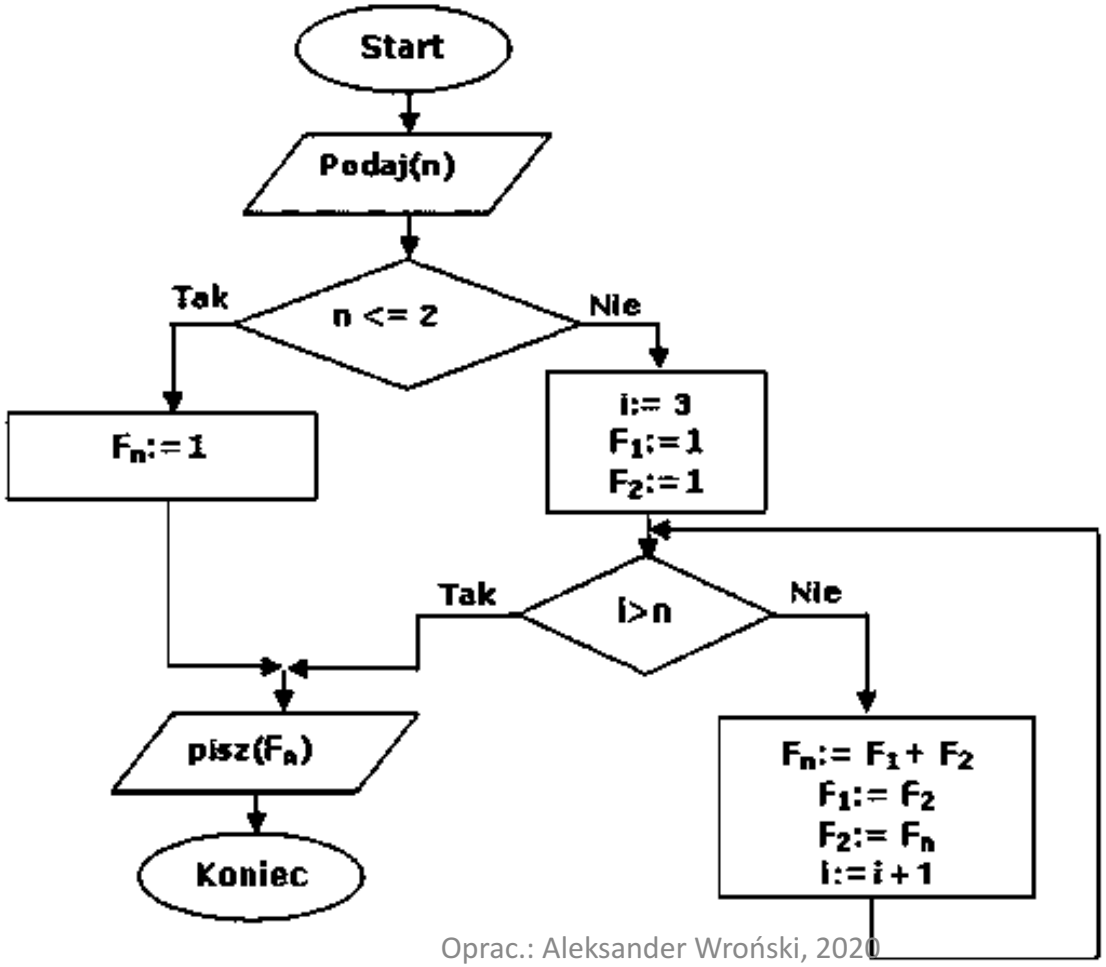
Algorytm NWP Euklidesa - różnicowy



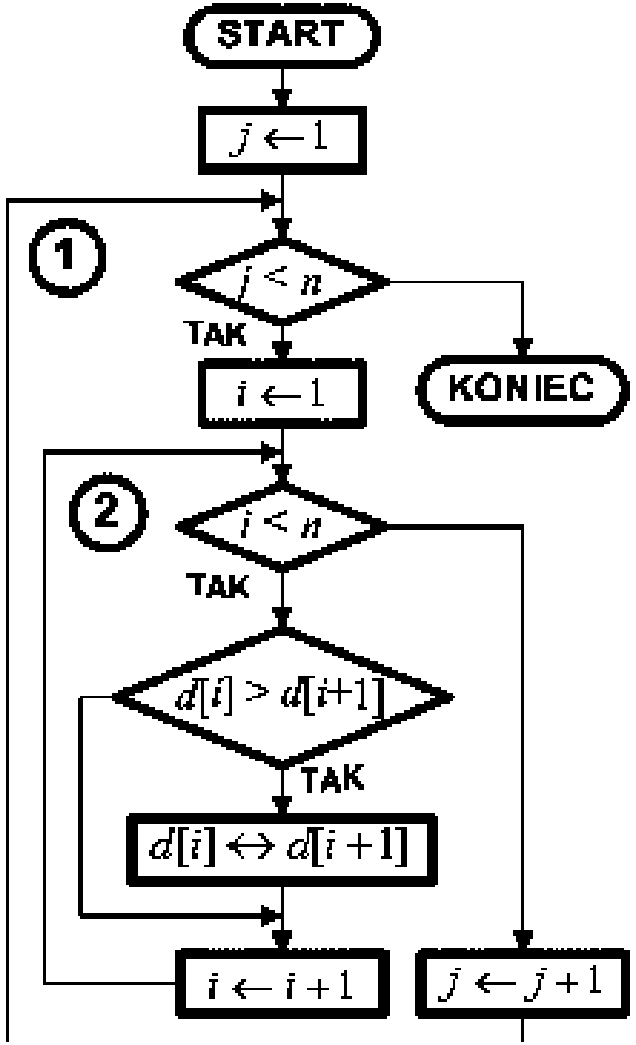
Algorytm NWP Euklidesa - dzieleniowy



Liczby Fibonacciego - algorytm



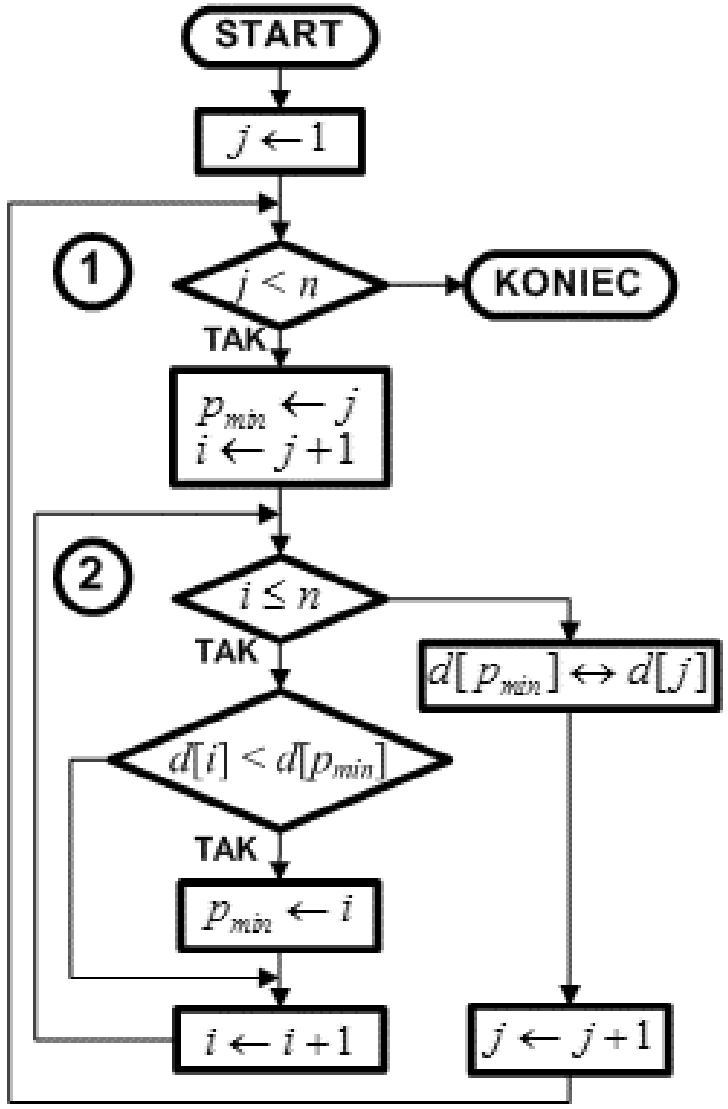
Sortowanie bąbelkowe




```
// Sortujemy

for(j = 0; j < N - 1; j++)
  for(i = 0; i < N - 1; i++)
    if(d[i] > d[i + 1])
    {
      x = d[i]; d[i] = d[i + 1]; d[i + 1] = x;
    };
```

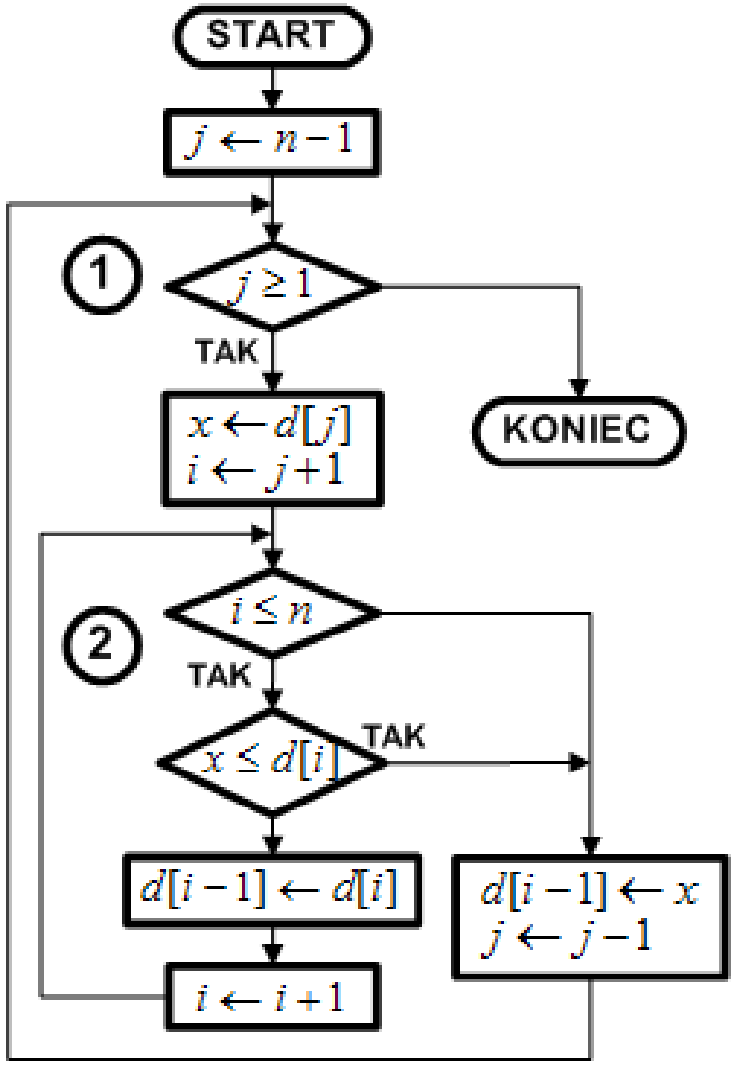
Sortowanie przez proste wybieranie




```
// Sortujemy

for(j = 0; j < N - 1; j++)
{
  pmin = j;
  for(i = j + 1; i < N; i++)
    if(d[i] < d[pmin]) pmin = i;
  x = d[pmin]; d[pmin] = d[j]; d[j] = x;
}
```

Sortowanie przez proste wstawianie




```
// Sortujemy

for(j = N - 2; j >= 0; j--)
{
  x = d[j];
  i = j + 1;
  while((i < N) && (x > d[i]))
  {
    d[i - 1] = d[i];
    i++;
  }
  d[i - 1] = x;
}
```

Myślenie komputacyjne

W najogólniejszym ujęciu, to powtarzalny wielostopniowy proces myślowy polegający na efektywnym znajdowaniu rozwiązań dla złożonych problemów metodami wzorowanych na tych stosowanych w informatyce (algorytmice).

W takim rozumieniu myślenie komputacyjne bliskie jest założeniom pedagogicznym Johna Deweya (1988), który uważał nabywanie umiejętności rozwiązywania problemów za najważniejsze w rozwoju myślenia.

Proces myślenia komputacyjnego składa się z następujących etapów:

- 1. Dekompozycja: sformułowanie problemu i rozłożenie go na części składowe.
- 2. Analiza: rozpoznanie prawidłowości właściwych problemowi.
- 3. Abstrahowanie: eliminowanie elementów nieistotnych przez uogólnianie.
- 4. Tworzenie algorytmu: ustalenie czynności prowadzących do rozwiązania problemu.

Tak rozumiane myślenie komputacyjne jest odbiciem niemal każdego procesu myślowego skierowanego na rozwiązanie problemu i koresponduje z rozwiązaniami problemów (zadań) informatycznych.

Myślenie komputacyjne (wg M.Sysło)

Rodzaj myślenia na drodze rozwiązywania problemów pochodzących z różnych dziedzin, odnoszący się jednocześnie do jego korzeni tkwiących w informatyce, w szczególności w programowaniu.

Należy podkreślić, że z jednej strony myślenie komputacyjne znacznie wykracza poza myślenie algorytmiczne, czyli poza informatykę, z drugiej zaś – że kształcenie informatyczne nie jest tożsame z nauką programowania.