

Algorytmy i struktury danych - pseudokod

Marcin Żurowski

1 marca 2024

Zaliczenie

- Obecność:
 - minimalnie 8 obecności
 - 3 zajęcia nieusprawiedliwione
 - tydzień na przyniesienie usprawiedliwienia
- Punkty:
- Oceny:

Zaliczenie

- Obecność:
 - minimalnie 8 obecności
 - 3 zajęcia nieusprawiedliwione
 - tydzień na przyniesienie usprawiedliwienia
- Punkty:
- Oceny:

Zaliczenie

- Obecność:
 - minimalnie 8 obecności
 - 3 zajęcia nieusprawiedliwione
 - tydzień na przyniesienie usprawiedliwienia
- Punkty:
 - 2 kolokwia po 25 punktów (poprawka średnia arytmetyczna)
 - 10 zadań programistycznych po 1 lub 1.5 punktu (0.5 punktu po terminie)
- Oceny:

Zaliczenie

- Obecność:
 - minimalnie 8 obecności
 - 3 zajęcia nieusprawiedliwione
 - tydzień na przyniesienie usprawiedliwienia
- Punkty:
 - 2 kolokwia po 25 punktów (poprawka średnia arytmetyczna)
 - 10 zadań programistycznych po 1 lub 1.5 punktu (0.5 punktu po terminie)
- Oceny:

Zaliczenie

- Obecność:
 - minimalnie 8 obecności
 - 3 zajęcia nieusprawiedliwione
 - tydzień na przyniesienie usprawiedliwienia
- Punkty:
 - 2 kolokwia po 25 punktów (poprawka średnia arytmetyczna)
 - 10 zadań programistycznych po 1 lub 1.5 punktu (0.5 punktu po terminie)
- Oceny:

Zaliczenie

- Obecność:
 - minimalnie 8 obecności
 - 3 zajęcia nieusprawiedliwione
 - tydzień na przyniesienie usprawiedliwienia
- Punkty:
 - 2 kolokwia po 25 punktów (poprawka średnia arytmetyczna)
 - 10 zadań programistycznych po 1 lub 1.5 punktu (0.5 punktu po terminie)
- Oceny:
 - $50 + \% \text{ punktów} \cdot 3$
 - $60 + \% \text{ punktów} \cdot 3.5$

Zaliczenie

- Obecność:
 - minimalnie 8 obecności
 - 3 zajęcia nieusprawiedliwione
 - tydzień na przyniesienie usprawiedliwienia
- Punkty:
 - 2 kolokwia po 25 punktów (poprawka średnia arytmetyczna)
 - 10 zadań programistycznych po 1 lub 1.5 punktu (0.5 punktu po terminie)
- Oceny:
 - 50 + % punktów - 3
 - 60 + % punktów - 3.5
 - 70 + % punktów - 4
 - 80 + % punktów - 4.5
 - 90 + % punktów - 5

Zaliczenie

- Obecność:
 - minimalnie 8 obecności
 - 3 zajęcia nieusprawiedliwione
 - tydzień na przyniesienie usprawiedliwienia
- Punkty:
 - 2 kolokwia po 25 punktów (poprawka średnia arytmetyczna)
 - 10 zadań programistycznych po 1 lub 1.5 punktu (0.5 punktu po terminie)
- Oceny:
 - 50 + % punktów - 3
 - 60 + % punktów - 3.5
 - 70 + % punktów - 4
 - 80 + % punktów - 4.5
 - 90 + % punktów - 5

Zaliczenie

- Obecność:
 - minimalnie 8 obecności
 - 3 zajęcia nieusprawiedliwione
 - tydzień na przyniesienie usprawiedliwienia
- Punkty:
 - 2 kolokwia po 25 punktów (poprawka średnia arytmetyczna)
 - 10 zadań programistycznych po 1 lub 1.5 punktu (0.5 punktu po terminie)
- Oceny:
 - 50 + % punktów - 3
 - 60 + % punktów - 3.5
 - 70 + % punktów - 4
 - 80 + % punktów - 4.5
 - 90 + % punktów - 5

Zaliczenie

- Obecność:
 - minimalnie 8 obecności
 - 3 zajęcia nieusprawiedliwione
 - tydzień na przyniesienie usprawiedliwienia
- Punkty:
 - 2 kolokwia po 25 punktów (poprawka średnia arytmetyczna)
 - 10 zadań programistycznych po 1 lub 1.5 punktu (0.5 punktu po terminie)
- Oceny:
 - 50 + % punktów - 3
 - 60 + % punktów - 3.5
 - 70 + % punktów - 4
 - 80 + % punktów - 4.5
 - 90 + % punktów - 5

Zaliczenie

- Obecność:
 - minimalnie 8 obecności
 - 3 zajęcia nieusprawiedliwione
 - tydzień na przyniesienie usprawiedliwienia
- Punkty:
 - 2 kolokwia po 25 punktów (poprawka średnia arytmetyczna)
 - 10 zadań programistycznych po 1 lub 1.5 punktu (0.5 punktu po terminie)
- Oceny:
 - 50 + % punktów - 3
 - 60 + % punktów - 3.5
 - 70 + % punktów - 4
 - 80 + % punktów - 4.5
 - 90 + % punktów - 5

Zaliczenie

- Obecność:
 - minimalnie 8 obecności
 - 3 zajęcia nieusprawiedliwione
 - tydzień na przyniesienie usprawiedliwienia
- Punkty:
 - 2 kolokwia po 25 punktów (poprawka średnia arytmetyczna)
 - 10 zadań programistycznych po 1 lub 1.5 punktu (0.5 punktu po terminie)
- Oceny:
 - 50 + % punktów - 3
 - 60 + % punktów - 3.5
 - 70 + % punktów - 4
 - 80 + % punktów - 4.5
 - 90 + % punktów - 5

Problemy

- decyzyjne
- optymalizacyjne
- wyszukiwania

Problemy

- decyzyjne
- optymalizacyjne
- wyszukiwania

Problemy

- decyzyjne
- optymalizacyjne
- wyszukiwania

Problemy

- decyzyjne
- optymalizacyjne
- wyszukiwania

Problem definicja

Problem:

- parametry
- pytanie

Przykład:

Jeżeli parametry mamy dane są, wówczas otrzymujemy
instancję problemu

Jeżeli do wszystkich zmiennych wstawimy wartości to uzyskamy
instancję problemu

Problem definicja

Problem:

- parametry
- pytanie

Przykład:

- parametry: mamy dane ciąg n liczb naturalnych (n_1, n_2, \dots, n_n)
- pytanie: czy istnieje...

Jeżeli do wszystkich zmiennych wstawimy wartości to uzyskamy instancję problemu

Problem definicja

Problem:

- parametry
- pytanie

Przykład:

- parametry: mamy dane ciąg n liczb naturalnych (n_1, n_2, \dots, n_n)
- pytanie:

Jeżeli do wszystkich zmiennych wstawimy wartości to uzyskamy instancję problemu

Problem definicja

Problem:

- parametry
- pytanie

Przykład:

- parametry: mamy dane ciąg n liczb naturalnych (n_1, n_2, \dots, n_n)
- pytanie:
 - czy istnieje najmniejszy element?
 - znajdź pierwszy najmniejszy element
 - znajdź najmniejszy element

Jeżeli do wszystkich zmiennych wstawimy wartości to uzyskamy instancję problemu

Problem definicja

Problem:

- parametry
- pytanie

Przykład:

- parametry: mamy dane ciąg n liczb naturalnych (n_1, n_2, \dots, n_n)
- pytanie:
 - czy istnieje najmniejszy element?
 - znajdź pierwszy najmniejszy element
 - znajdź najmniejszy element

Jeżeli do wszystkich zmiennych wstawimy wartości to uzyskamy instancję problemu

Problem definicja

Problem:

- parametry
- pytanie

Przykład:

- parametry: mamy dane ciąg n liczb naturalnych (n_1, n_2, \dots, n_n)
- pytanie:
 - czy istnieje najmniejszy element?
 - znajdź pierwszy najmniejszy element
 - znajdź najmniejszy element

Jeżeli do wszystkich zmiennych wstawimy wartości to uzyskamy instancję problemu

Problem definicja

Problem:

- parametry
- pytanie

Przykład:

- parametry: mamy dane ciąg n liczb naturalnych (n_1, n_2, \dots, n_n)
- pytanie:
 - czy istnieje najmniejszy element?
 - znajdź pierwszy najmniejszy element
 - znajdź najmniejszy element

Jeżeli do wszystkich zmiennych wstawimy wartości to uzyskamy instancję problemu

Problem definicja

Problem:

- parametry
- pytanie

Przykład:

- parametry: mamy dane ciąg n liczb naturalnych (n_1, n_2, \dots, n_n)
- pytanie:
 - czy istnieje najmniejszy element?
 - znajdź pierwszy najmniejszy element
 - znajdź najmniejszy element

Jeżeli do wszystkich zmiennych wstawimy wartości to uzyskamy instancję problemu

Problem definicja

Problem:

- parametry
- pytanie

Przykład:

- parametry: mamy dane ciąg n liczb naturalnych (n_1, n_2, \dots, n_n)
- pytanie:
 - czy istnieje najmniejszy element?
 - znajdź pierwszy najmniejszy element
 - znajdź najmniejszy element

Jeżeli do wszystkich zmiennych wstawimy wartości to uzyskamy instancję problemu

Problem definicja

Problem:

- parametry
- pytanie

Przykład:

- parametry: mamy dane ciąg n liczb naturalnych (n_1, n_2, \dots, n_n)
- pytanie:
 - czy istnieje najmniejszy element?
 - znajdź pierwszy najmniejszy element
 - znajdź najmniejszy element

Jeżeli do wszystkich zmiennych wstawimy wartości to uzyskamy **instancję problemu**

Algorytm

Sekwencja instrukcji elementarnych rozwiązująca dowolną instancję problemu

Własności algorytmu:

- precyzyjny opis
- determinizm (każdy krok algorytmu jest jednoznacznie określony)
- skończoność
- efektywność

Algorytm

Sekwencja instrukcji elementarnych rozwiązująca dowolną instancję problemu

Własności algorytmu:

- precyzyjny opis
- determinizm (z wyjątkiem algorytmów randomizowanych)
- skończoność
- poprawność
- efektywność

Algorytm

Sekwencja instrukcji elementarnych rozwiązująca dowolną instancję problemu

Własności algorytmu:

- precyzyjny opis
- determinizm (z wyjątkiem algorytmów randomizowanych)
- skończoność
- poprawność
- ogólność

Algorytm

Sekwencja instrukcji elementarnych rozwiązująca dowolną instancję problemu

Własności algorytmu:

- precyzyjny opis
- determinizm (z wyjątkiem algorytmów randomizowanych)
- skończoność
- poprawność
- ogólność

Algorytm

Sekwencja instrukcji elementarnych rozwiązująca dowolną instancję problemu

Własności algorytmu:

- precyzyjny opis
- determinizm (z wyjątkiem algorytmów randomizowanych)
- skończoność
- poprawność
- ogólność

Algorytm

Sekwencja instrukcji elementarnych rozwiązująca dowolną instancję problemu

Własności algorytmu:

- precyzyjny opis
- determinizm (z wyjątkiem algorytmów randomizowanych)
- skończoność
- poprawność
- ogólność

Algorytm

Sekwencja instrukcji elementarnych rozwiązująca dowolną instancję problemu

Własności algorytmu:

- precyzyjny opis
- determinizm (z wyjątkiem algorytmów randomizowanych)
- skończoność
- poprawność
- ogólność

Reprezentacja algorytmów

- język naturalny
- lista kroków
- schemat blokowy
- pseudokod
- język programowania

Reprezentacja algorytmów

- język naturalny
- lista kroków
- schemat blokowy
- pseudokod
- język programowania

Reprezentacja algorytmów

- język naturalny
- lista kroków
- schemat blokowy
- pseudokod
- język programowania

Reprezentacja algorytmów

- język naturalny
- lista kroków
- schemat blokowy
- pseudokod
- język programowania

Reprezentacja algorytmów

- język naturalny
- lista kroków
- schemat blokowy
- pseudokod
- język programowania

Zmienna

- integer `i, j`
- real `x, y`
- Boolean `war1 = false, war2 = true`

Przypisanie wartości

Zmienna

- integer `i, j`
- real `x, y`
- Boolean `war1 = false, war2 = true`

Przypisanie wartości

Zmienna

- integer `i, j`
- real `x, y`
- Boolean `war1 = false, war2 = true`

Przypisanie wartości

Zmienna

- integer i, j
- real x, y
- Boolean war1 = false, war2 = true

Przypisanie wartości

● a = 3

● b = 4

Zmienna

- integer i, j
- real x, y
- Boolean $war1 = \text{false}, war2 = \text{true}$

Przypisanie wartości

- $a = 3$
- $Z = W$

Zmienna

- integer i, j
- real x, y
- Boolean $war1 = false, war2 = true$

Przypisanie wartości

- $a = 3$
- $Z = W$

Zmienna

- integer i, j
- real x, y
- Boolean $war1 = false, war2 = true$

Przypisanie wartości

- $a = 3$
- $Z = W$

Operatory

- NOT
- * / DIV MOD
- + -
- < <= > >=
- = !=
- AND
- OR
- =

Operatory

- NOT
- * / DIV MOD
- + -
- < <= > >=
- = !=
- AND
- OR
- =

Operatory

- NOT
- * / DIV MOD
- + -
- < <= > >=
- = !=
- AND
- OR
- =

Operatory

- NOT
- * / DIV MOD
- + -
- < <= > >=
- = !=
- AND
- OR
- =

Operatory

- NOT
- * / DIV MOD
- + -
- < <= > >=
- = !=
- AND
- OR
- =

Operatory

- NOT
- * / DIV MOD
- + -
- < <= > >=
- = !=
- AND
- OR
- =

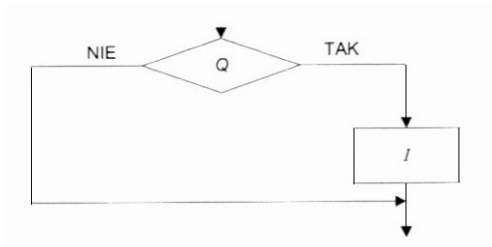
Operatory

- NOT
- * / DIV MOD
- + -
- < <= > >=
- = !=
- AND
- OR
- =

Operatory

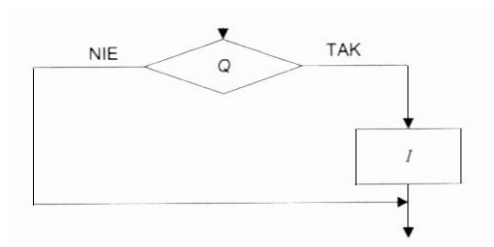
- NOT
- * / DIV MOD
- + -
- < <= > >=
- = !=
- AND
- OR
- =

Instrukcja warunkowa



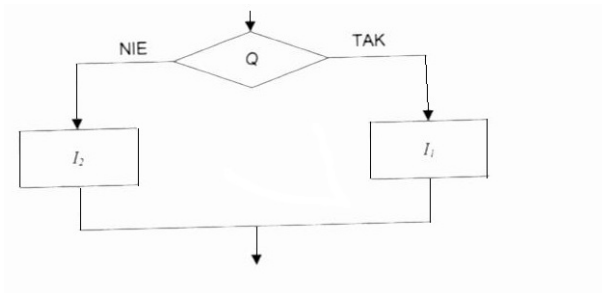
```
if Q  
  I
```

Instrukcja warunkowa



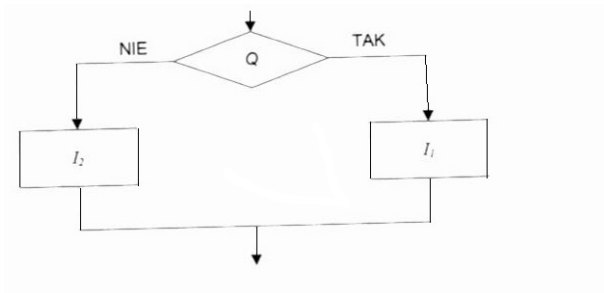
```
if Q  
  I
```

Instrukcja warunkowa



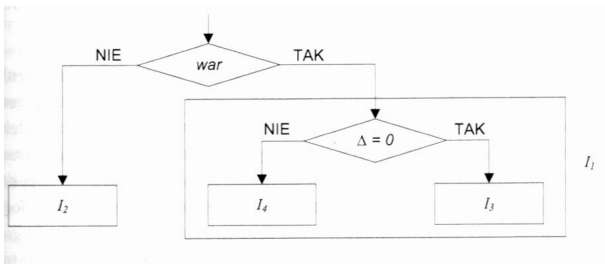
```
if Q
  I1
else
  I2
```

Instrukcja warunkowa



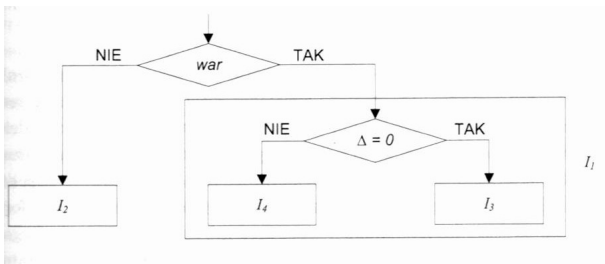
```
if Q
  I1
else
  I2
```

Instrukcja warunkowa



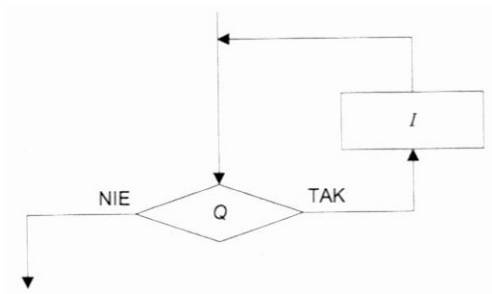
```
if war
  if delta = 0
    I3
  else
    I4
else
  I2
```

Instrukcja warunkowa



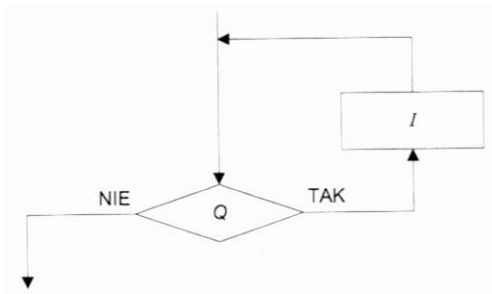
```
if war
  if delta = 0
    I3
  else
    I4
else
  I2
```

Iteracje - while



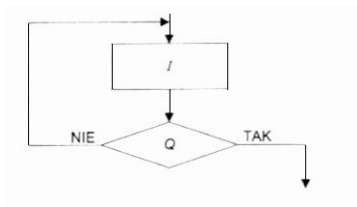
```
while Q  
  I
```

Iteracje - while



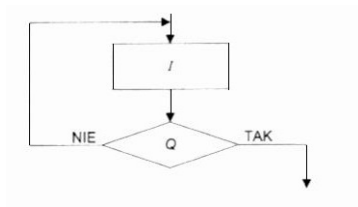
```
while Q  
  I
```


repeat



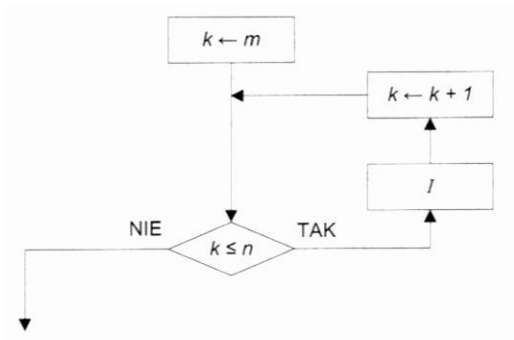
```
repeat  
  I  
until Q
```

repeat



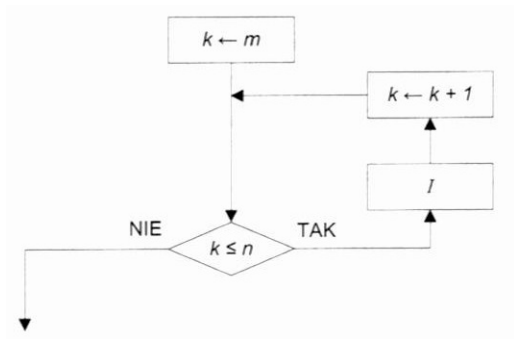
```
repeat  
  I  
until Q
```

for



```
for k = m to n  
  I
```

for



```
for k = m to n  
  I
```

Tablice

- integer array $T[1..100]$
- $T[50] = 4$
- real array $T2[1..10,1..10]$
- $T2[1,1] = 2.0$

Tablice

- integer array $T[1..100]$
- $T[50] = 4$
- real array $T2[1..10,1..10]$
- $T2[1,1] = 2.0$

Tablice

- integer array $T[1..100]$
- $T[50] = 4$
- real array $T2[1..10, 1..10]$
- $T2[1, 1] = 2.0$

Tablice

- integer array $T[1..100]$
- $T[50] = 4$
- real array $T2[1..10, 1..10]$
- $T2[1, 1] = 2.0$

Procedura

```
procedure N(PF)
```

```
  S
```

```
  I
```

Przykład:

```
procedure MIN(a,b)
```

```
  integer a, b
```

```
  if a < b
```

```
    MIN = a
```

```
  else
```

```
    MIN = b
```

Procedura

```
procedure N(PF)
```

```
  S
```

```
  I
```

Przykład:

```
procedure MIN(a,b)
```

```
  integer a, b
```

```
  if a < b
```

```
    MIN = a
```

```
  else
```

```
    MIN = b
```

Wejście i wyjście

- `read(a,b)`
- `write(c)`

Wejście i wyjście

- `read(a,b)`
- `write(c)`

1

Wyznacz wartości zmiennych i , j , k w następujących przypadkach:

1 $i = 10$

$$k = i + 6$$

$$i = 2 * i - 5$$

$$j = i + k - 3$$

2 $i = 10$

$$i = 2 * i - 5$$

$$k = i + 6$$

$$j = i + k - 3$$

3 $k = i + 6$

$$i = 10$$

$$i = 2 * i - 5$$

$$j = i + k - 3$$

1

Wyznacz wartości zmiennych i , j , k w następujących przypadkach:

1 $i = 10$

$$k = i + 6$$

$$i = 2 * i - 5$$

$$j = i + k - 3$$

2 $i = 10$

$$i = 2 * i - 5$$

$$k = i + 6$$

$$j = i + k - 3$$

3 $k = i + 6$

$$i = 10$$

$$i = 2 * i - 5$$

$$j = i + k - 3$$

1

Wyznacz wartości zmiennych i , j , k w następujących przypadkach:

① $i = 10$

$$k = i + 6$$

$$i = 2 * i - 5$$

$$j = i + k - 3$$

② $i = 10$

$$i = 2 * i - 5$$

$$k = i + 6$$

$$j = i + k - 3$$

③ $k = i + 6$

$$i = 10$$

$$i = 2 * i - 5$$

$$j = i + k - 3$$

Zakładając, że zmiennym a , b , c , x zostały przypisane wartości.
Zapisz ciąg instrukcji wyliczających wartość wielomianu:
 $w(x) = ax^2 + bx + c$.

3

Dane są dwie zmienne a i b , którym wcześniej przypisano liczby. Napisz ciąg instrukcji "zamieniający" wartość tych zmiennych. Dokładnie, liczba, która jest przypisana a , ma stać się wartością zmiennej b , a liczba, która jest przypisana zmiennej b , ma stać się wartością zmiennej a .

Wyobraźmy sobie hipotetyczną sytuację, że każdy klient w supermarkecie podchodząc do kasy, mówi, ile ma w koszyku towarów. Kasjerka z kodu kreskowego wprowadza cenę każdego produktu. Skonstruujemy algorytm, który jako wynik poda ostateczną kwotę, jaką musi zapłacić klient.

Podać algorytm wczytujący liczby różne od zera i obliczający ich sumę. Nie wiemy przy tym, ile liczb jest do wczytania.