

Algorytmy i struktury danych - pseudokod

Marcin Żurowski

12 października 2025

Plan zajęć

- 1 Zaliczenie
- 2 Problem algorytmiczny, instancja
- 3 Algorytm, struktura danych
- 4 Pseudokod

Zaliczenie

- Obecność:

- minimalnie 4 obecności
- 3 zajęcia nieusprawiedliwione
- tydzień na przyniesienie usprawiedliwienia

- Punkty:

- Aby zaliczyć wykłady, należy zaliczyć laboratoria
- Warunkiem zaliczenia przedmiotu jest zdanie sprawdzianu pisemnego (ocena przynajmniej 3.0)
- Oceny:

Zaliczenie

- Obecność:
 - minimalnie 4 obecności
 - 3 zajęcia nieusprawiedliwione
 - tydzień na przyniesienie usprawiedliwienia
- Punkty:
 - Aby zaliczyć wykłady, należy zaliczyć laboratoria
 - Warunkiem zaliczenia przedmiotu jest zdanie sprawdzianu pisemnego (ocena przynajmniej 3.0)
 - Oceny:

Zaliczenie

- Obecność:
 - minimalnie 4 obecności
 - 3 zajęcia nieusprawiedliwione
 - tydzień na przyniesienie usprawiedliwienia
- Punkty:
 - 20 * zaliczenia po 20 punktów (poprawka 3 razy w semestrze)
 - 2 zadania programistyczne po 1 lub 2 punkty (15 punktów po semestrze)
- Aby zaliczyć wykłady, należy zaliczyć laboratoria
- Warunkiem zaliczenia przedmiotu jest zdanie sprawdzianu pisemnego (ocena przynajmniej 3.0)
- Oceny:

Zaliczenie

- Obecność:
 - minimalnie 4 obecności
 - 3 zajęcia nieusprawiedliwione
 - tydzień na przyniesienie usprawiedliwienia
- Punkty:
 - 2 kolokwia po 25 punktów (poprawka średnia arytmetyczna)
 - 7 zadań programistycznych po 1 lub 2 punktu (0.5 punktu po terminie)
- Aby zaliczyć wykłady, należy zaliczyć laboratoria
- Warunkiem zaliczenia przedmiotu jest zdanie sprawdzianu pisemnego (ocena przynajmniej 3.0)
- Oceny:

Zaliczenie

- Obecność:
 - minimalnie 4 obecności
 - 3 zajęcia nieusprawiedliwione
 - tydzień na przyniesienie usprawiedliwienia
- Punkty:
 - 2 kolokwia po 25 punktów (poprawka średnia arytmetyczna)
 - 7 zadań programistycznych po 1 lub 2 punktu (0.5 punktu po terminie)
- Aby zaliczyć wykłady, należy zaliczyć laboratoria
- Warunkiem zaliczenia przedmiotu jest zdanie sprawdzianu pisemnego (ocena przynajmniej 3.0)
- Oceny:

Zaliczenie

- Obecność:
 - minimalnie 4 obecności
 - 3 zajęcia nieusprawiedliwione
 - tydzień na przyniesienie usprawiedliwienia
- Punkty:
 - 2 kolokwia po 25 punktów (poprawka średnia arytmetyczna)
 - 7 zadań programistycznych po 1 lub 2 punktu (0.5 punktu po terminie)
- Aby zaliczyć wykłady, należy zaliczyć laboratoria
- Warunkiem zaliczenia przedmiotu jest zdanie sprawdzianu pisemnego (ocena przynajmniej 3.0)
- Oceny:

Zaliczenie

- Obecność:
 - minimalnie 4 obecności
 - 3 zajęcia nieusprawiedliwione
 - tydzień na przyniesienie usprawiedliwienia
- Punkty:
 - 2 kolokwia po 25 punktów (poprawka średnia arytmetyczna)
 - 7 zadań programistycznych po 1 lub 2 punktu (0.5 punktu po terminie)
- Aby zaliczyć wykłady, należy zaliczyć laboratoria
- Warunkiem zaliczenia przedmiotu jest zdanie sprawdzianu pisemnego (ocena przynajmniej 3.0)
- Oceny:

Zaliczenie

- Obecność:
 - minimalnie 4 obecności
 - 3 zajęcia nieusprawiedliwione
 - tydzień na przyniesienie usprawiedliwienia
- Punkty:
 - 2 kolokwia po 25 punktów (poprawka średnia arytmetyczna)
 - 7 zadań programistycznych po 1 lub 2 punktu (0.5 punktu po terminie)
- Aby zaliczyć wykłady, należy zaliczyć laboratoria
- Warunkiem zaliczenia przedmiotu jest zdanie sprawdzianu pisemnego (ocena przynajmniej 3.0)
- Oceny:

Zaliczenie

- Obecność:
 - minimalnie 4 obecności
 - 3 zajęcia nieusprawiedliwione
 - tydzień na przyniesienie usprawiedliwienia
- Punkty:
 - 2 kolokwia po 25 punktów (poprawka średnia arytmetyczna)
 - 7 zadań programistycznych po 1 lub 2 punktu (0.5 punktu po terminie)
- Aby zaliczyć wykłady, należy zaliczyć laboratoria
- Warunkiem zaliczenia przedmiotu jest zdanie sprawdzianu pisemnego (ocena przynajmniej 3.0)
- Oceny:
 - $50 + \% \text{ punktów} \cdot 3$
 - $50 + \% \text{ punktów} \cdot 3.5$

Zaliczenie

- Obecność:
 - minimalnie 4 obecności
 - 3 zajęcia nieusprawiedliwione
 - tydzień na przyniesienie usprawiedliwienia
- Punkty:
 - 2 kolokwia po 25 punktów (poprawka średnia arytmetyczna)
 - 7 zadań programistycznych po 1 lub 2 punktu (0.5 punktu po terminie)
- Aby zaliczyć wykłady, należy zaliczyć laboratoria
- Warunkiem zaliczenia przedmiotu jest zdanie sprawdzianu pisemnego (ocena przynajmniej 3.0)
- Oceny:
 - 50 + % punktów - 3
 - 60 + % punktów - 3.5
 - 70 + % punktów - 4
 - 80 + % punktów - 4.5
 - 90 + % punktów - 5

Zaliczenie

- Obecność:
 - minimalnie 4 obecności
 - 3 zajęcia nieusprawiedliwione
 - tydzień na przyniesienie usprawiedliwienia
- Punkty:
 - 2 kolokwia po 25 punktów (poprawka średnia arytmetyczna)
 - 7 zadań programistycznych po 1 lub 2 punktu (0.5 punktu po terminie)
- Aby zaliczyć wykłady, należy zaliczyć laboratoria
- Warunkiem zaliczenia przedmiotu jest zdanie sprawdzianu pisemnego (ocena przynajmniej 3.0)
- Oceny:
 - 50 + % punktów - 3
 - 60 + % punktów - 3.5
 - 70 + % punktów - 4
 - 80 + % punktów - 4.5
 - 90 + % punktów - 5

Zaliczenie

- Obecność:
 - minimalnie 4 obecności
 - 3 zajęcia nieusprawiedliwione
 - tydzień na przyniesienie usprawiedliwienia
- Punkty:
 - 2 kolokwia po 25 punktów (poprawka średnia arytmetyczna)
 - 7 zadań programistycznych po 1 lub 2 punktu (0.5 punktu po terminie)
- Aby zaliczyć wykłady, należy zaliczyć laboratoria
- Warunkiem zaliczenia przedmiotu jest zdanie sprawdzianu pisemnego (ocena przynajmniej 3.0)
- Oceny:
 - 50 + % punktów - 3
 - 60 + % punktów - 3.5
 - 70 + % punktów - 4
 - 80 + % punktów - 4.5
 - 90 + % punktów - 5

Zaliczenie

- Obecność:
 - minimalnie 4 obecności
 - 3 zajęcia nieusprawiedliwione
 - tydzień na przyniesienie usprawiedliwienia
- Punkty:
 - 2 kolokwia po 25 punktów (poprawka średnia arytmetyczna)
 - 7 zadań programistycznych po 1 lub 2 punktu (0.5 punktu po terminie)
- Aby zaliczyć wykłady, należy zaliczyć laboratoria
- Warunkiem zaliczenia przedmiotu jest zdanie sprawdzianu pisemnego (ocena przynajmniej 3.0)
- Oceny:
 - 50 + % punktów - 3
 - 60 + % punktów - 3.5
 - 70 + % punktów - 4
 - 80 + % punktów - 4.5
 - 90 + % punktów - 5

Zaliczenie

- Obecność:
 - minimalnie 4 obecności
 - 3 zajęcia nieusprawiedliwione
 - tydzień na przyniesienie usprawiedliwienia
- Punkty:
 - 2 kolokwia po 25 punktów (poprawka średnia arytmetyczna)
 - 7 zadań programistycznych po 1 lub 2 punktu (0.5 punktu po terminie)
- Aby zaliczyć wykłady, należy zaliczyć laboratoria
- Warunkiem zaliczenia przedmiotu jest zdanie sprawdzianu pisemnego (ocena przynajmniej 3.0)
- Oceny:
 - 50 + % punktów - 3
 - 60 + % punktów - 3.5
 - 70 + % punktów - 4
 - 80 + % punktów - 4.5
 - 90 + % punktów - 5

Zaliczenie

- Obecność:
 - minimalnie 4 obecności
 - 3 zajęcia nieusprawiedliwione
 - tydzień na przyniesienie usprawiedliwienia
- Punkty:
 - 2 kolokwia po 25 punktów (poprawka średnia arytmetyczna)
 - 7 zadań programistycznych po 1 lub 2 punktu (0.5 punktu po terminie)
- Aby zaliczyć wykłady, należy zaliczyć laboratoria
- Warunkiem zaliczenia przedmiotu jest zdanie sprawdzianu pisemnego (ocena przynajmniej 3.0)
- Oceny:
 - 50 + % punktów - 3
 - 60 + % punktów - 3.5
 - 70 + % punktów - 4
 - 80 + % punktów - 4.5
 - 90 + % punktów - 5

Literatura

- 1 T. Cormen, Ch. Leiserson, R. Rivest, C. Stein, Wprowadzenie do algorytmów, Wydawnictwa Naukowo-Techniczne, Warszawa 2012.
- 2 E. Palka, Elementy algorytmiki dla początkujących, Wydawnictwo Naukowe UAM, 2012.
- 3 Kenneth A. Ross, Charles R.B. Wright, Matematyka dyskretna, Wydawnictwo Naukowe PWN, Warszawa 2012.
- 4 M. Wirth, Algorytmy + struktury danych – programy, Wydawnictwa Naukowo-Techniczne, Warszawa 2000.
- 5 J. Bentley, Perła oprogramowania, Wydawnictwa Naukowo-Techniczne, Warszawa 2001.
- 6 D. Horst, Y. Feldman, Rozpoczęcie Informatyki, Algorytmika, Wydawnictwa Naukowo-Techniczne, Warszawa 2003.

Literatura

- 1 T. Cormen, Ch. Leiserson, R. Rivest, C. Stein, Wprowadzenie do algorytmów, Wydawnictwa Naukowo-Techniczne, Warszawa 2012.
- 2 E. Palka, Elementy algorytmiki dla początkujących, Wydawnictwo Naukowe UAM, 2012.
- 3 Kenneth A. Ross, Charles R.B. Wright, Matematyka dyskretna, Wydawnictwo Naukowe PWN, Warszawa 2012.
- 4 N. Wirth, Algorytmy + struktury danych = programy, Wydawnictwa Naukowo -Techniczne, Warszawa 2000.
- 5 J. Bentley, Perelki oprogramowania, Wydawnictwa Naukowo-Techniczne, Warszawa 2001.
- 6 D. Harel, Y. Feldman, Rzecz o istocie informatyki. Algorytmika, Wydawnictwa Naukowo-Techniczne, Warszawa 2008.

Literatura

- 1 T. Cormen, Ch. Leiserson, R. Rivest, C. Stein, Wprowadzenie do algorytmów, Wydawnictwa Naukowo-Techniczne, Warszawa 2012.
- 2 E. Palka, Elementy algorytmiki dla początkujących, Wydawnictwo Naukowe UAM, 2012.
- 3 Kenneth A. Ross, Charles R.B. Wright, Matematyka dyskretna, Wydawnictwo Naukowe PWN, Warszawa 2012.
- 4 N. Wirth, Algorytmy + struktury danych = programy, Wydawnictwa Naukowo -Techniczne, Warszawa 2000.
- 5 J. Bentley, Perelki oprogramowania, Wydawnictwa Naukowo-Techniczne, Warszawa 2001.
- 6 D. Harel, Y. Feldman, Rzecz o istocie informatyki. Algorytmika, Wydawnictwa Naukowo-Techniczne, Warszawa 2008.

Literatura

- 1 T. Cormen, Ch. Leiserson, R. Rivest, C. Stein, Wprowadzenie do algorytmów, Wydawnictwa Naukowo-Techniczne, Warszawa 2012.
- 2 E. Palka, Elementy algorytmiki dla początkujących, Wydawnictwo Naukowe UAM, 2012.
- 3 Kenneth A. Ross, Charles R.B. Wright, Matematyka dyskretna, Wydawnictwo Naukowe PWN, Warszawa 2012.
- 4 N. Wirth, Algorytmy + struktury danych = programy, Wydawnictwa Naukowo -Techniczne, Warszawa 2000.
- 5 J. Bentley, Perłki oprogramowania, Wydawnictwa Naukowo-Techniczne, Warszawa 2001.
- 6 D. Harel, Y. Feldman, Rzecz o istocie informatyki. Algorytmika, Wydawnictwa Naukowo-Techniczne, Warszawa 2008.

Literatura

- 1 T. Cormen, Ch. Leiserson, R. Rivest, C. Stein, Wprowadzenie do algorytmów, Wydawnictwa Naukowo-Techniczne, Warszawa 2012.
- 2 E. Palka, Elementy algorytmiki dla początkujących, Wydawnictwo Naukowe UAM, 2012.
- 3 Kenneth A. Ross, Charles R.B. Wright, Matematyka dyskretna, Wydawnictwo Naukowe PWN, Warszawa 2012.
- 4 N. Wirth, Algorytmy + struktury danych = programy, Wydawnictwa Naukowo -Techniczne, Warszawa 2000.
- 5 J. Bentley, Perełki oprogramowania, Wydawnictwa Naukowo-Techniczne, Warszawa 2001.
- 6 D. Harel, Y. Feldman, Rzecz o istocie informatyki. Algorytmika, Wydawnictwa Naukowo-Techniczne, Warszawa 2008.

Literatura

- 1 T. Cormen, Ch. Leiserson, R. Rivest, C. Stein, Wprowadzenie do algorytmów, Wydawnictwa Naukowo-Techniczne, Warszawa 2012.
- 2 E. Palka, Elementy algorytmiki dla początkujących, Wydawnictwo Naukowe UAM, 2012.
- 3 Kenneth A. Ross, Charles R.B. Wright, Matematyka dyskretna, Wydawnictwo Naukowe PWN, Warszawa 2012.
- 4 N. Wirth, Algorytmy + struktury danych = programy, Wydawnictwa Naukowo -Techniczne, Warszawa 2000.
- 5 J. Bentley, Perełki oprogramowania, Wydawnictwa Naukowo-Techniczne, Warszawa 2001.
- 6 D. Harel, Y. Feldman, Rzecz o istocie informatyki. Algorytmika, Wydawnictwa Naukowo-Techniczne, Warszawa 2008.

Literatura

- 1 T. Cormen, Ch. Leiserson, R. Rivest, C. Stein, Wprowadzenie do algorytmów, Wydawnictwa Naukowo-Techniczne, Warszawa 2012.
- 2 E. Palka, Elementy algorytmiki dla początkujących, Wydawnictwo Naukowe UAM, 2012.
- 3 Kenneth A. Ross, Charles R.B. Wright, Matematyka dyskretna, Wydawnictwo Naukowe PWN, Warszawa 2012.
- 4 N. Wirth, Algorytmy + struktury danych = programy, Wydawnictwa Naukowo -Techniczne, Warszawa 2000.
- 5 J. Bentley, Perełki oprogramowania, Wydawnictwa Naukowo-Techniczne, Warszawa 2001.
- 6 D. Harel, Y. Feldman, Rzecz o istocie informatyki. Algorytmika, Wydawnictwa Naukowo-Techniczne, Warszawa 2008.

Problemy algorytmiczny - definicja

Problem algorytmiczny - opis rodziny możliwych:

- parametrów wejściowych
- pożądanых wyników (pytanie)

Problemy algorytmiczny - definicja

Problem algorytmiczny - opis rodziny możliwych:

- parametrów wejściowych
- pożądanych wyników (pytanie)

Problemy algorytmiczny - definicja

Problem algorytmiczny - opis rodziny możliwych:

- parametrów wejściowych
- pożądanych wyników (pytanie)

Problem algorytmiczny - podział

Problemy algorytmiczny dzielimy na:

- decyzyjne
- optymalizacyjne
- wyszukiwania

Przykład:

Jeżeli do wszystkich zmiennych wstawimy wartości to uzyskamy instancję problemu (egzemplarz)

Problem algorytmiczny - podział

Problemy algorytmiczny dzielimy na:

- decyzyjne
- optymalizacyjne
- wyszukiwania

Przykład:

Jeżeli do wszystkich zmiennych wstawimy wartości to uzyskamy instancję problemu (egzemplarz)

Problem algorytmiczny - podział

Problemy algorytmiczny dzielimy na:

- decyzyjne
- optymalizacyjne
- wyszukiwania

Przykład:

W parametrze mamy dane n liczb naturalnych $(a_1, a_2, \dots, a_n)$

Jeżeli do wszystkich zmiennych wstawimy wartości to uzyskamy
instancję problemu (egzemplarz)

Problem algorytmiczny - podział

Problemy algorytmiczny dzielimy na:

- decyzyjne
- optymalizacyjne
- wyszukiwania

Przykład:

- parametry: mamy dane ciąg n liczb naturalnych $(n_1, n_2, \dots, n_n)$
- pytanie: czy istnieje podciąg o sumie równej 0

Jeżeli do wszystkich zmiennych wstawimy wartości to uzyskamy instancję problemu (egzemplarz)

Problem algorytmiczny - podział

Problemy algorytmiczny dzielimy na:

- decyzyjne
- optymalizacyjne
- wyszukiwania

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 (egzemplarz)

Problem algorytmiczny - podział

Problemy algorytmiczny dzielimy na:

- decyzyjne
- optymalizacyjne
- wyszukiwania

Przykład:

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

- czy istnieje najmniejszy element? (decyzyjny)
- znajdź pierwszy najmniejszy element (optymalizacyjny)
- znajdź najmniejszy element (optymalizacyjny)

Jeżeli do wszystkich zmiennych wstawimy wartości to uzyskamy instancję problemu (egzemplarz)

Problem algorytmiczny - podział

Problemy algorytmiczny dzielimy na:

- decyzyjne
- optymalizacyjne
- wyszukiwania

Przykład:

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

Jeżeli do wszystkich zmiennych wstawimy wartości to uzyskamy instancję problemu (egzemplarz)

Problem algorytmiczny - podział

Problemy algorytmiczny dzielimy na:

- decyzyjne
- optymalizacyjne
- wyszukiwania

Przykład:

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

Jeżeli do wszystkich zmiennych wstawimy wartości to uzyskamy instancję problemu (egzemplarz)

Problem algorytmiczny - podział

Problemy algorytmiczny dzielimy na:

- decyzyjne
- optymalizacyjne
- wyszukiwania

Przykład:

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

Jeżeli do wszystkich zmiennych wstawimy wartości to uzyskamy
instancję problemu (egzemplarz)

Problem algorytmiczny - podział

Problemy algorytmiczny dzielimy na:

- decyzyjne
- optymalizacyjne
- wyszukiwania

Przykład:

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

Jeżeli do wszystkich zmiennych wstawimy wartości to uzyskamy
instancję problemu (egzemplarz)

Problem algorytmiczny - podział

Problemy algorytmiczny dzielimy na:

- decyzyjne
- optymalizacyjne
- wyszukiwania

Przykład:

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

Jeżeli do wszystkich zmiennych wstawimy wartości to uzyskamy **instancję problemu (egzemplarz)**

Problem algorytmiczny - podział

Problemy algorytmiczny dzielimy na:

- decyzyjne
- optymalizacyjne
- wyszukiwania

Przykład:

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

Jeżeli do wszystkich zmiennych wstawimy wartości to uzyskamy **instancję problemu (egzemplarz)**

Algorytm

Algorytm - sekwencja instrukcji elementarnych rozwiązująca dowolną instancję problemu

Własności algorytmu:

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

Algorytm

Algorytm - sekwencja instrukcji elementarnych rozwiązująca dowolną instancję problemu

Własności algorytmu:

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

Algorytm

Algorytm - sekwencja instrukcji elementarnych rozwiązująca dowolną instancję problemu

Własności algorytmu:

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

Algorytm

Algorytm - sekwencja instrukcji elementarnych rozwiązująca dowolną instancję problemu

Własności algorytmu:

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

Algorytm

Algorytm - sekwencja instrukcji elementarnych rozwiązująca dowolną instancję problemu

Własności algorytmu:

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

Algorytm

Algorytm - sekwencja instrukcji elementarnych rozwiązująca dowolną instancję problemu

Własności algorytmu:

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

Struktura danych, zmienna

Struktura danych - sposób przechowywania informacji w komputerze.

Zmienna - miejsce, w którym można przechowywać pewną wartość.

Własności zmiennej:

- adres

- wartość

- nazwa

- typ (zmienność)

- adres pamięci

- zakres

Struktura danych, zmienna

Struktura danych - sposób przechowywania informacji w komputerze.

Zmienna - miejsce, w którym można przechowywać pewną wartość.

Własności zmiennej:

- adres
- wartość
- nazwa
- typ (zmienność)
- zakres
- sposób zapisu

Struktura danych, zmienna

Struktura danych - sposób przechowywania informacji w komputerze.

Zmienna - miejsce, w którym można przechowywać pewną wartość.

Własności zmiennej:

- adres
- wartość
- nazwa
- typ (rozmiar)
- czas życia
- dostępność

Struktura danych, zmienna

Struktura danych - sposób przechowywania informacji w komputerze.

Zmienna - miejsce, w którym można przechowywać pewną wartość.

Własności zmiennej:

- adres
- wartość
- nazwa
- typ (rozmiar)
- czas życia
- dostępność

Struktura danych, zmienna

Struktura danych - sposób przechowywania informacji w komputerze.

Zmienna - miejsce, w którym można przechowywać pewną wartość.

Własności zmiennej:

- adres
- wartość
- nazwa
- typ (rozmiar)
- czas życia
- dostępność

Struktura danych, zmienna

Struktura danych - sposób przechowywania informacji w komputerze.

Zmienna - miejsce, w którym można przechowywać pewną wartość.

Własności zmiennej:

- adres
- wartość
- nazwa
- typ (rozmiar)
- czas życia
- dostępność

Struktura danych, zmienna

Struktura danych - sposób przechowywania informacji w komputerze.

Zmienna - miejsce, w którym można przechowywać pewną wartość.

Własności zmiennej:

- adres
- wartość
- nazwa
- typ (rozmiar)
- czas życia
- dostępność

Struktura danych, zmienna

Struktura danych - sposób przechowywania informacji w komputerze.

Zmienna - miejsce, w którym można przechowywać pewną wartość.

Własności zmiennej:

- adres
- wartość
- nazwa
- typ (rozmiar)
- czas życia
- dostępność

- **integer** i, j = 2
- **real** x, y = 3.14
- **Boolean** war1 = false, war2 = true

- **integer** i, j = 2
- **real** x, y = 3.14
- **Boolean** war1 = **false**, war2 = **true**

- **integer** i, j = 2
- **real** x, y = 3.14
- **Boolean** war1 = false, war2 = true

Zmienna

Typy:

- **integer** i, j = 2
- **real** x, y = 3.14
- **Boolean** war1 = **false**, war2 = **true**

Termy:

Zmienna

Typy:

- **integer** `i, j = 2`
- **real** `x, y = 3.14`
- **Boolean** `war1 = false, war2 = true`

Termy:

- stała
- zmienna
- funkcja (operator)

Zmienna

Typy:

- **integer** i, j = 2
- **real** x, y = 3.14
- **Boolean** war1 = **false**, war2 = **true**

Termy:

- stała
- zmienna
- funkcja (operator)

Zmienna

Typy:

- **integer** i, j = 2
- **real** x, y = 3.14
- **Boolean** war1 = **false**, war2 = **true**

Termy:

- stała
- zmienna
- funkcja (operator)

Zmienna

Typy:

- **integer** i, j = 2
- **real** x, y = 3.14
- **Boolean** war1 = **false**, war2 = **true**

Termy:

- stała
- zmienna
- funkcja (operator)

Operatory

Operatory:

- przypisania: `zmienna = wyrażenie`
- arytmetyczne: `+`, `-`, `*`, `/`, **`DIV`**, **`MOD`**
- relacji: `=`, `!=`, `<`, `<=`, `>`, `>=`
- logiczne: **`NOT`**, **`AND`**, **`OR`**

○○●○○○○○○○○○○○○○○○○○○○○○○○○○○○○○○

Operator

Priorytety:

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

○○●○○○○○○○○○○○○○○○○○○○○○○○○○○○○○○

Operator

Priorytety:

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

Operator

Priorytety:

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

Operator

Priorytety:

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

Operator

Priorytety:

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

Operator

Priorytety:

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

Łączność:

- lewostronna $a + b + c \Rightarrow ((a + b) + c)$
- prawostronna $a = b = c \Rightarrow (a + (b + c))$

Operator

Priorytety:

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

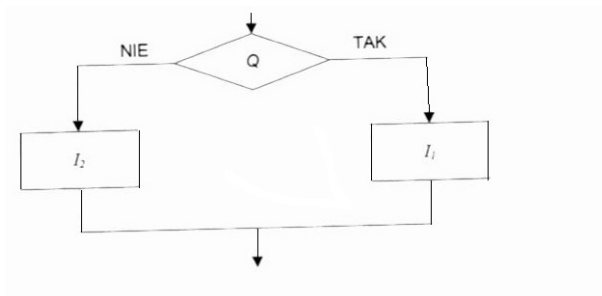
Łączność:

- lewostronna $a + b + c \Rightarrow ((a + b) + c)$
- prawostronna $a = b = c \Rightarrow (a + (b + c))$

Instrukcja warunkowa - przykład

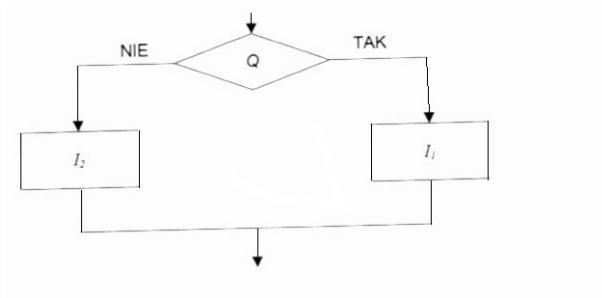
```
real a, b, c, x1, x2, delta
delta = b * b - 4 * a * c
if delta >= 0
    delta = sqrt(delta)
    a = 2 * a
    x1 = (-b - delta) / a
    x2 = (-b + delta) / a
```

Instrukcja warunkowa



```
if Warunek
    Instrukcja1
else
    Instrukcja2
```


Instrukcja warunkowa

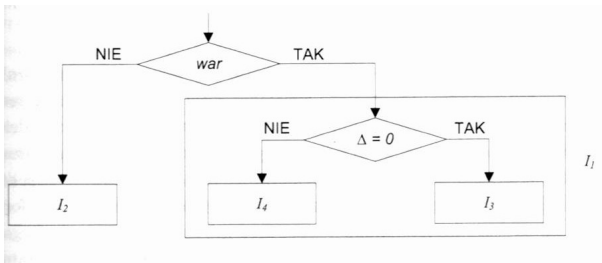


```
if Warunek  
    Instrukcja1  
else  
    Instrukcja2
```

Instrukcja warunkowa - przykład

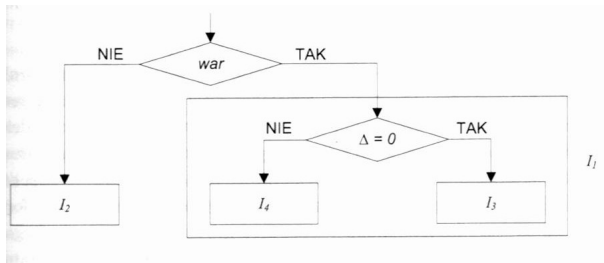
```
real a, b, c, x1, x2, delta
delta = b * b - 4 * a * c
if delta >= 0
    delta = sqrt(delta)
    a = 2 * a
    x1 = (-b - delta) / a
    x2 = (-b + delta) / a
else
    pisz(nie ma pierwiastków rzeczywistych)
```

Instrukcja warunkowa



```
if Warunek1
  if Warunek2
    Instrukcja3
  else
    Instrukcja4
else
  Instrukcja2
```

Instrukcja warunkowa

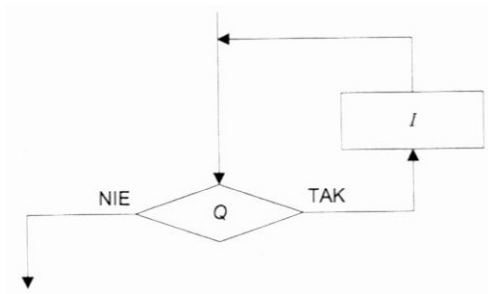


```
if Warunek1
  if Warunek2
    Instrukcja3
  else
    Instrukcja4
else
  Instrukcja2
```

Instrukcja warunkowa - przykład

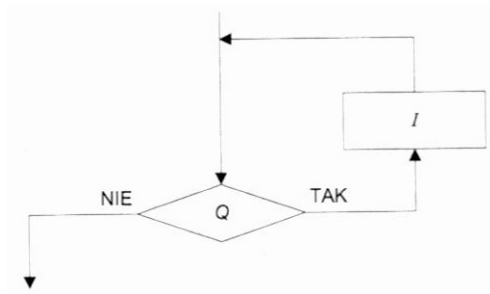
```
real a, b, c, x1, x2, delta
delta = b * b - 4 * a * c
if delta >= 0
    if delta = 0
        delta = sqrt(delta)
        a = 2 * a
        x1 = (-b - delta) / a
        x2 = (-b + delta) / a
    else
        x1 = -b / (2 * a)
        x2 = x1
    pisz x1, x2
else
    pisz(nie ma pierwiastków rzeczywistych)
```

Iteracje - while



while Warunek
Instrukcje

Iteracje - while

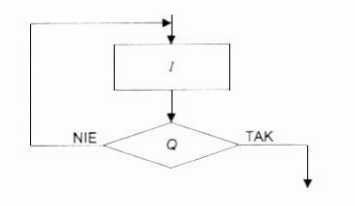


while Warunek
Instrukcje

Iteracje - while - przykład

```
integer a, n, b, k  
czytaj a, n  
b = 1  
k = 1  
while k <= n  
    b = b * a  
    k = k + 1  
pisz b
```


repeat

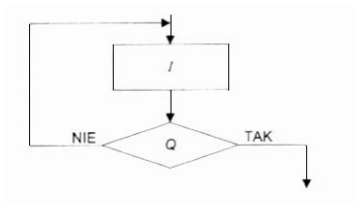


repeat

Instrukcje

until Warunek

repeat



repeat

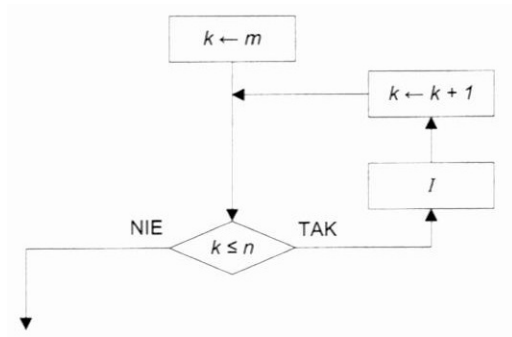
Instrukcje

until Warunek

repeat - przykład

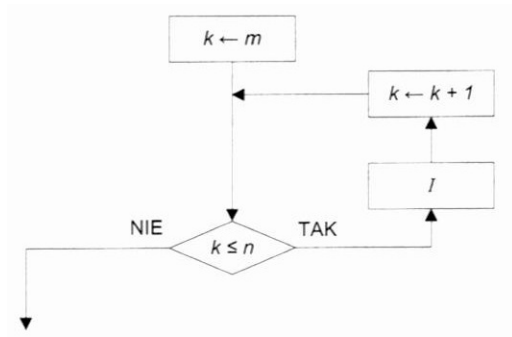
```
real s, a
s = 0
repeat
  czytaj a
  s = s + a
until a = 0
pisz s
```

for



for $k = m$ **to** n
Instrukcje

for



for $k = m$ **to** n
Instrukcje

for - przykład

```
integer a, n, b, k  
czytaj a, n  
b = 1  
for k = 1 to n  
    b = b * a  
pisz b
```

Tablice - jednowymiarowe

Tablica jednowymiarowa (wektor) - struktura danych, do której elementu odwołujemy się za pomocą indeksu

- integer array $T[1..100]$

- $T[50] = 4$

Tablice - jednowymiarowe

Tablica jednowymiarowa (wektor) - struktura danych, do której elementu odwołujemy się za pomocą indeksu

- **integer array** $T[1..100]$
- $T[50] = 4$

Tablice - jednowymiarowe

Tablica jednowymiarowa (wektor) - struktura danych, do której elementu odwołujemy się za pomocą indeksu

- **integer array** $T[1..100]$
- $T[50] = 4$

Tablice - jednowymiarowe - przykład

```
integer i, n, s
integer array A[1..n]
czytaj n, A
s = 0
for i = 1 to n
    s = s + A[i]
pisz s
```

Tablice - dwuwymiarowe

Tablica dwuwymiarowa (macierz) - struktura danych, do której elementu odwołujemy się za pomocą uporządkowanej pary indeksów, oznaczających odpowiednio numer wiersza i numer kolumny macierzy.

• `real array T2[1..10,1..10]`

• `T2[1,1] = 2.0`

Tablice - dwuwymiarowe

Tablica dwuwymiarowa (macierz) - struktura danych, do której elementu odwołujemy się za pomocą uporządkowanej pary indeksów, oznaczających odpowiednio numer wiersza i numer kolumny macierzy.

- **real array** T2[1..10,1..10]
- T2[1,1] = 2.0

Tablice - dwuwymiarowe

Tablica dwuwymiarowa (macierz) - struktura danych, do której elementu odwołujemy się za pomocą uporządkowanej pary indeksów, oznaczających odpowiednio numer wiersza i numer kolumny macierzy.

- **real array** T2[1..10,1..10]
- T2[1,1] = 2.0

Tablice - dwuwymiarowa - przykład

```
integer i, j, n, s, m
integer array A[1..n, 1..m]
czytaj n, m, A
s = 0
for i = 1 to n
  for j = 1 to m
    s = s + A[i,j]
pisz s
```

Funkcja

Funkcja (procedura, procedura funkcyjna, metoda) -
wydzielona część algorytmu wykonująca pewne operacje,
ewentualnie zwracająca ich wyniki i posiadająca nazwę.
do funkcji można przekazać pewne parametry (argumenty):

- Parametry formalne - obiekt występujący w treści funkcji,
nie odpowiada mu bezpośrednio żaden obiekt głównego
algorytmu.
- Parametry aktualne - wartość, która w momencie użycia
funkcji w głównym algorytmie przyporządkowana do
parametru formalnego.

```
procedure NAZWA(ParametryFormalne)  
    SpecyfikacjaParametrów  
    Instrukcje
```

Funkcja

Funkcja (procedura, procedura funkcyjna, metoda) - wydzielona część algorytmu wykonująca pewne operacje, ewentualnie zwracająca ich wyniki i posiadająca nazwę. do funkcji można przekazać pewne parametry (argumenty):

- **Parametry formalne** - obiekt występujący w treści funkcji, nie odpowiada mu bezpośrednio żaden obiekt głównego algorytmu.
- **Parametry aktualne** - wartość, która w momencie użycia funkcji w głównym algorytmie przyporządkowana do parametru formalnego.

```
procedure NAZWA(ParametryFormalne)
    SpecyfikacjaParametrów
    Instrukcje
```


Funkcja

Funkcja (procedura, procedura funkcyjna, metoda) - wydzielona część algorytmu wykonująca pewne operacje, ewentualnie zwracająca ich wyniki i posiadająca nazwę. do funkcji można przekazać pewne parametry (argumenty):

- **Parametry formalne** - obiekt występujący w treści funkcji, nie odpowiada mu bezpośrednio żaden obiekt głównego algorytmu.
- **Parametry aktualne** - wartość, która w momencie użycia funkcji w głównym algorytmie przyporządkowana do parametru formalnego.

```
procedure NAZWA(ParametryFormalne)
    SpecyfikacjaParametrów
Instrukcje
```

Funkcja

Funkcja (procedura, procedura funkcyjna, metoda) - wydzielona część algorytmu wykonująca pewne operacje, ewentualnie zwracająca ich wyniki i posiadająca nazwę. do funkcji można przekazać pewne parametry (argumenty):

- **Parametry formalne** - obiekt występujący w treści funkcji, nie odpowiada mu bezpośrednio żaden obiekt głównego algorytmu.
- **Parametry aktualne** - wartość, która w momencie użycia funkcji w głównym algorytmie przyporządkowana do parametru formalnego.

```
procedure NAZWA(ParametryFormalne)
    SpecyfikacjaParametrów
Instrukcje
```

Funkcja

Funkcja (procedura, procedura funkcyjna, metoda) - wydzielona część algorytmu wykonująca pewne operacje, ewentualnie zwracająca ich wyniki i posiadająca nazwę. do funkcji można przekazać pewne parametry (argumenty):

- **Parametry formalne** - obiekt występujący w treści funkcji, nie odpowiada mu bezpośrednio żaden obiekt głównego algorytmu.
- **Parametry aktualne** - wartość, która w momencie użycia funkcji w głównym algorytmie przyporządkowana do parametru formalnego.

procedure NAZWA(ParametryFormalne)

SpecyfikacjaParametrów

Instrukcje

Funkcja - przykład

```
procedure MIN(a,b)
    integer a, b
    if a < b
        MIN = a //return a
    else
        MIN = b //return b
```

Wejście i wyjście

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

Wejście i wyjście

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