Algorytmy numeryczne i przybliżone w języku Java

Źródło: Aswin Anand, domena publiczna.

Wyznaczanie miejsca zerowego funkcji z wykorzystaniem metody bisekcji (przeczytasz o niej więcej w e-materiale Algorytmy numeryczne i przybliżone), polega na dzieleniu na pół zadanego przedziału argumentów funkcji i sprawdzaniu, czy dla argumentu znajdującego się pośrodku przedziału wartość funkcji wynosi zero. Ten e-materiał dotyczy implementacji rozwiązania problemu w języku Java.

Więcej informacji znajduje się w e-materiale:

- Algorytmy numeryczne i przybliżone.

Ciekawi cię, jak wyglądają implementacje w innych językach programowania? Możesz się z nimi zapoznać w dwóch pozostałych e-materiałach z tej serii:

- Algorytmy numeryczne i przybliżone w języku C++,
- Algorytmy numeryczne i przybliżone w języku Python.

Więcej zadań? Sięgnij do zadań maturalnych z tego tematu:

- Algorytmy numeryczne i przybliżone – zadania maturalne.

> **Twoje cele**
>
> - Wyznaczysz miejsce zerowe funkcji, wykorzystując metodę bisekcji.

- Zastosujesz metody numeryczne, by uzyskać przybliżone wartości rozwiązań w zadaniach dotyczących bisekcji.
- Wyjaśnisz, na czym polega podejście iteracyjne i rekurencyjne przy stosowaniu bisekcji.

Przeczytaj

Algorytm bisekcji

Metoda bisekcji (lub metoda równego podziału) jest metodą szacunkową, która pozwala znaleźć miejsce zerowe funkcji w danym przedziale $\langle a, b\rangle$ gdzie $a < b$. Aby tak się stało, muszą zostać spełnione pewne warunki:

- obszar poszukiwania należy ograniczyć do przedziału $\langle a, b\rangle$ tak, aby funkcja na krańcach przedziałów przyjmowała wartości różnych znaków,
- funkcja w tym przedziale musi być ciągła.

Proces szukania miejsca zerowego polega na podzieleniu wybranego przedziału na dwie równe części, a następnie wybraniu jednej z nich (tej, która będzie dalszym obszarem poszukiwania). Wybór zależy od tego, który podprzedział dalej będzie spełniał warunek różności funkcji na swoich krańcach. Proces ten kontynuujemy do momentu, gdy osiągniemy określony warunek stopu.

Przykładowym warunkiem stopu mogą być następujące warunki:

1. Wartość funkcji w znalezionym punkcie, będącym potencjalnym miejscem zerowym funkcji, czyli wartość funkcji w tym punkcie wynosi, co do wartości bezwzględnej, <= ε, gdzie ε jest określone na początku algorytmu.
2. Różnica pomiędzy punktami będącymi potencjalnymi miejscami zerowymi w kolejnych iteracjach wynosi $\pm\varepsilon$, gdzie ε jest określone na początku algorytmu.
3. Osiągnięta została określona liczba iteracji algorytmu.

Bisekcja w języku Java – podejście rekurencyjne

Problem bisekcji może być zrealizowany w postaci rekurencji, od której zaczniemy implementację. Funkcja, której miejsca zerowego będziemy szukać, to $\sin(x)$ w przedziale $\langle 1;\ 5\rangle$. Rekurencja opiera się na założeniu istnienia stanu końcowego, do którego będziemy próbowali dotrzeć. W omawianym przypadku stan ten będzie określony przez wybrany warunek stopu.

Zrealizujmy zatem pierwszy z określonych warunków. Do funkcji rekurencyjnej przekazywać będziemy parametry `a` oraz `b`, które określają kraniec przedziału, a także `epsilon`, czyli dokładność. Wówczas program może wyglądać następująco:

```
1 public class Main {
2     public static double f(double x) {
```

```
        return Math.sin(x);
    }

    public static double bisekcja(double a, double b, double epsi
        double x = (a + b) / 2;
        if (Math.abs(f(x)) < epsilon) {
            return x;
        }
        else if (f(a) * f(x) < 0) {
            return bisekcja(a, x, epsilon);
        }
        return bisekcja(x, b, epsilon);
    }

    public static void main(String[] args) {
        System.out.println(bisekcja(1, 5, 0.00001));
    }
}
```

W wyniku uruchomienia programu na ekran wypisany zostaje wynik `3.1416015625`. Jest to zgodne z prawdą, gdyż jedynym miejscem zerowym funkcji sinus w podanym przedziale jest liczba $\pi \approx 3.1415926535$.

Bisekcja w języku Java – podejście iteracyjne

Podejście iteracyjne do bisekcji również wymaga określenia warunku stopu. Tym razem zaimplementujemy drugi z wymienionych na wstępie przykładowych warunków. Sprawdzenie warunku musi się wykonać co najmniej raz, aby uzyskać różnicę pomiędzy kolejnymi przybliżeniami miejsca zerowego. Odpowiednią konstrukcją informatyczną będzie zatem pętla `do ... while` (oczywiście jest to możliwe do zrealizowania również za pomocą pętli `while`).

```
public class Main {
    public static double f(double x) {
        return Math.sin(x);
    }

    public static double bisekcja(double a, double b, double epsi
        double x0, x1 = (a + b) / 2;

```

```
        do {
            x0 = x1;

            if (f(a) * f(x0) < 0) {
                b = x0;
            }
            else {
                a = x0;
            }

            x1 = (a + b) / 2;
        }
        while (Math.abs(x1 - x0) > epsilon);

        return x1;
    }

    public static void main(String[] args) {
        System.out.println(bisekcja(1, 5, 0.00001));
    }
}
```

W wyniku uruchomienia programu na ekran wypisany zostaje wynik `3.1415939331054688`. Jest to rozwiązanie dużo dokładniejsze, jednak należy pamiętać, że wymagało wykonania większej liczby operacji.

Słownik

> **funkcja ciągła**
>
> funkcja jest ciągła, gdy jest ciągła w każdym punkcie swojej dziedziny
> **warunek stopu**
>
> moment zakończenia wykonywania algorytmu

Symulacja interaktywna

Polecenie 1

Przeanalizuj symulację interaktywną, przedstawiającą sposób działania algorytmu bisekcji. Zapisz algorytm w postaci listy kroków.

Zasób interaktywny dostępny pod adresem https://zpe.gov.pl/a/Dkw039BO6

Polecenie 2

Przygotuj notatkę ze swoimi spostrzeżeniami dotyczącymi symulacji.

Sprawdź się

Pokaż ćwiczenia:

W tym e-materiale pojawiają się treści dotyczące wyznaczania przybliżonej wartości liczby π oraz przybliżonej wartości pierwiastka kwadratowego. Więcej na ten temat znajdziesz w następujących e-materiałach:

- Obliczanie przybliżonej wartości pierwiastka kwadratowego z liczby nieujemnej,
- Wyznaczanie liczby π metodą Monte Carlo.

Ćwiczenie 1

Napisz program, który za pomocą algorytmu bisekcji wypisze wartość pierwiastka kwadratowego z liczby naturalnej dodatniej n. Wynik wypisz z dokładnością do trzech miejsc po przecinku (nie zaokrąglaj). Przetestuj działanie programu dla $n = 2$.

Specyfikacja:

Dane:

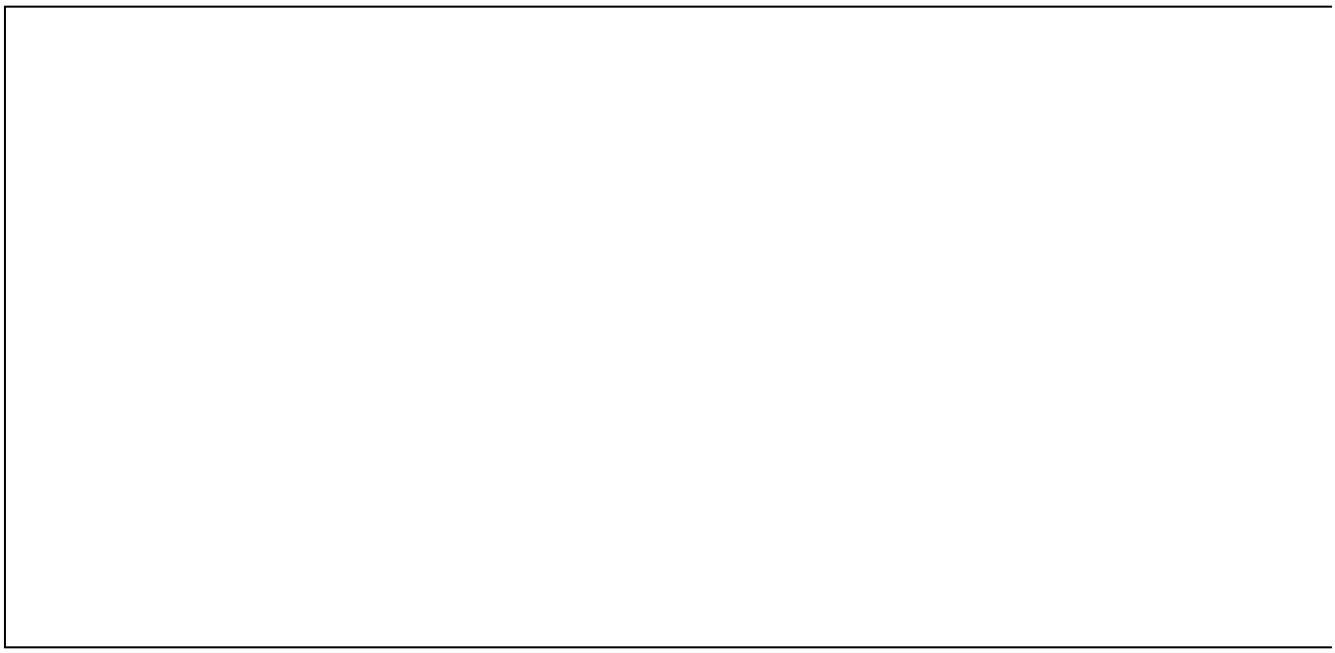
- n – liczba naturalna dodatnia

Wynik:

Program wypisuje wartość pierwiastka liczby naturalnej n obliczonej za pomocą metody bisekcji z dokładnością do trzech miejsc po przecinku (bez zaokrąglania).

Twoje zadania

1. Program wyznacza pierwiastek z liczby naturalnej dodatniej n metodą bisekcji.



Ćwiczenie 2

Napisz program, który wypisze liczbę iteracji potrzebnych do znalezienia przybliżenia pierwiastka z liczby naturalnej dodatniej n za pomocą algorytmu bisekcji. Przyjmij dokładność do trzech miejsc po przecinku. Przetestuj działanie programu dla n = 31.

Specyfikacja:

Dane:

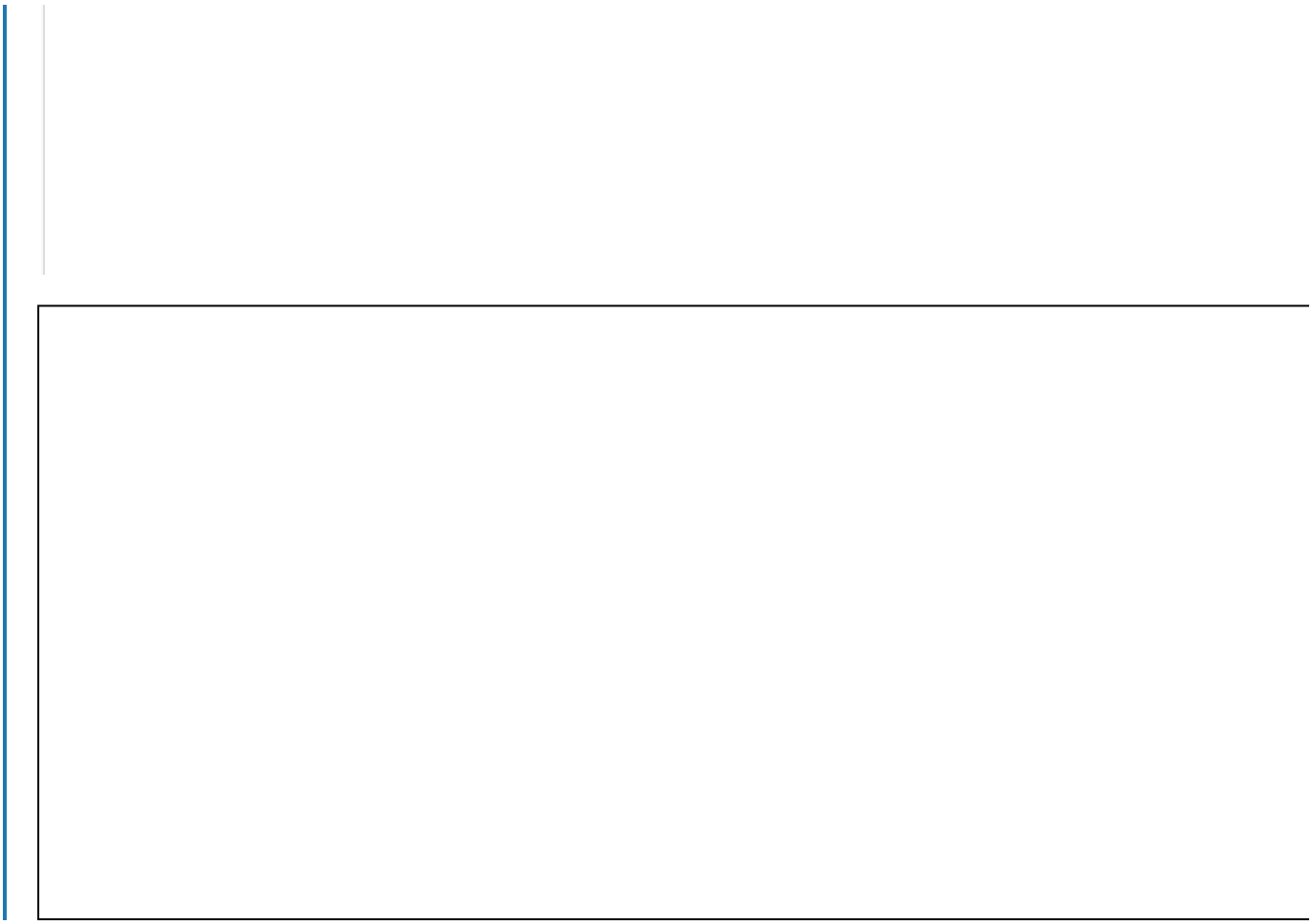
- n – liczba naturalna dodatnia

Wynik:

Program wypisuje liczbę iteracji potrzebnych do znalezienia pierwiastka liczby n metodą bisekcji z dokładnością do trzech miejsc po przecinku.

Twoje zadania

1. Program wypisuje liczbę iteracji potrzebnych do znalezienia wartości pierwiastka z liczby naturalnej n.



Ćwiczenie 3

Napisz program, który na podstawie algorytmu bisekcji wyznaczy wartość liczby π ,
a następnie obliczy pole koła o promieniu `n` i wypisze wynik z dokładnością do trzech miejsc po przecinku (bez zaokrąglania). Przetestuj działanie programu dla `n = 19`.

Specyfikacja:

Dane:

- `r` – liczba naturalna dodatnia, promień koła

Wynik:

Program wypisuje wartość pola koła dla zadanego promienia `n` z dokładnością do trzech miejsc po przecinku (bez zaokrąglania).

Twoje zadania

1. Program oblicza pole koła dla zadanego promienia `n`.

Dla nauczyciela

Autor: Maurycy Gast
Przedmiot: Informatyka

Temat: Algorytmy numeryczne i przybliżone w języku Java

Grupa docelowa:

Liceum ogólnokształcące i technikum, liceum ogólnokształcące, technikum, zakres rozszerzony

Podstawa programowa:

Cele kształcenia – wymagania ogólne

I. Rozumienie, analizowanie i rozwiązywanie problemów na bazie logicznego i abstrakcyjnego myślenia, myślenia algorytmicznego i sposobów reprezentowania informacji.

II. Programowanie i rozwiązywanie problemów z wykorzystaniem komputera oraz innych urządzeń cyfrowych: układanie i programowanie algorytmów, organizowanie, wyszukiwanie i udostępnianie informacji, posługiwanie się aplikacjami komputerowymi.

Treści nauczania – wymagania szczegółowe

I. Rozumienie, analizowanie i rozwiązywanie problemów.

Zakres rozszerzony. Uczeń spełnia wymagania określone dla zakresu podstawowego, a ponadto:

6) objaśnia sposoby wykonywania przez komputer działań arytmetycznych i operacji logicznych;

I + II. Zakres rozszerzony. Uczeń spełnia wymagania określone dla zakresu podstawowego, a ponadto:

1) zapisuje za pomocą listy kroków, schematu blokowego lub pseudokodu, i implementuje w wybranym języku programowania, algorytmy poznane na wcześniejszych etapach oraz algorytmy:

f) wyznaczania miejsc zerowych funkcji metodą połowienia,

Kształtowane kompetencje kluczowe:

- kompetencje cyfrowe;
- kompetencje osobiste, społeczne i w zakresie umiejętności uczenia się;
- kompetencje matematyczne oraz kompetencje w zakresie nauk przyrodniczych, technologii i inżynierii.

Cele operacyjne (językiem ucznia):

- Wyznaczysz miejsce zerowe funkcji, wykorzystując metodę bisekcji.
- Zastosujesz metody numeryczne, by uzyskać przybliżone wartości rozwiązań w zadaniach dotyczących bisekcji.
- Wyjaśnisz, na czym polega podejście iteracyjne i rekurencyjne przy stosowaniu bisekcji.

Strategie nauczania:

- konstruktywizm;
- konektywizm.

Metody i techniki nauczania:

- dyskusja;
- rozmowa nauczająca z wykorzystaniem multimedium i ćwiczeń interaktywnych;
- ćwiczenia praktyczne.

Formy pracy:

- praca indywidualna;
- praca w parach;
- praca w grupach;
- praca całego zespołu klasowego.

Środki dydaktyczne:

- komputery z głośnikami, słuchawkami i dostępem do internetu;
- zasoby multimedialne zawarte w e-materiale;
- tablica interaktywna/tablica, pisak/kreda;
- oprogramowanie dla języka Java SE 8 (lub nowszej wersji), w tym Eclipse 4.4 (lub nowszej wersji).

Przebieg lekcji

Przed lekcją:

1. **Przygotowanie do zajęć.** Nauczyciel loguje się na platformie i udostępnia e-materiał: „Algorytmy numeryczne i przybliżone w języku Java". Uczniowie zapoznają się z treściami w sekcji „Przeczytaj" w kontekście programowania.

Faza wstępna:

1. Wyświetlenie przez nauczyciela tematu i celów zajęć, przejście do wspólnego ustalenia kryteriów sukcesu.

Faza realizacyjna:

1. **Praca z tekstem.** Jeżeli przygotowanie uczniów do lekcji jest niewystarczające, nauczyciel prosi o indywidualne zapoznanie się z treścią zawartą w sekcji „Przeczytaj". Każdy uczestnik zajęć podczas cichego czytania wynotowuje najważniejsze kwestie poruszane w tekście.
2. **Praca z multimedium.** Nauczyciel wyświetla zawartość sekcji „Symulacja interaktywna". Uczniowie wspólnie analizują symulację interaktywną przedstawiającą sposób działania algorytmu bisekcji. Następnie próbują dokonać implementacji tego algorytmu w wybranym języku programowania.
3. **Ćwiczenie umiejętności.** Prowadzący zapowiada uczniom, że będą rozwiązywać ćwiczenie nr 1 z sekcji „Sprawdź się". Uczniowie wykonują je w parach. Po ustalonym czasie następuje porównanie napisanych kodów podczas wspólnego omówienia rozwiązań.

Faza podsumowująca:

1. Nauczyciel wyświetla na tablicy temat lekcji i cele zawarte w sekcji „Wprowadzenie". W kontekście ich realizacji podsumowuje przebieg zajęć, a także wskazuje mocne i słabe strony pracy uczniów.
2. Nauczyciel prosi uczniów o podsumowanie zgromadzonej wiedzy w zakresie programowania w języku Java.

Praca domowa:

1. Uczniowie wykonują ćwiczenie nr 2 z sekcji „Sprawdź się".

Materiały pomocnicze:

- Oficjalna dokumentacja techniczna dla języka Java SE 8 (lub nowszej wersji).
- Oficjalna dokumentacja techniczna dla oprogramowania Eclipse 4.4 (lub nowszej wersji).

Wskazówki metodyczne:

- Treści w sekcji „Przeczytaj" można wykorzystać jako podsumowanie i utrwalenie wiedzy uczniów.