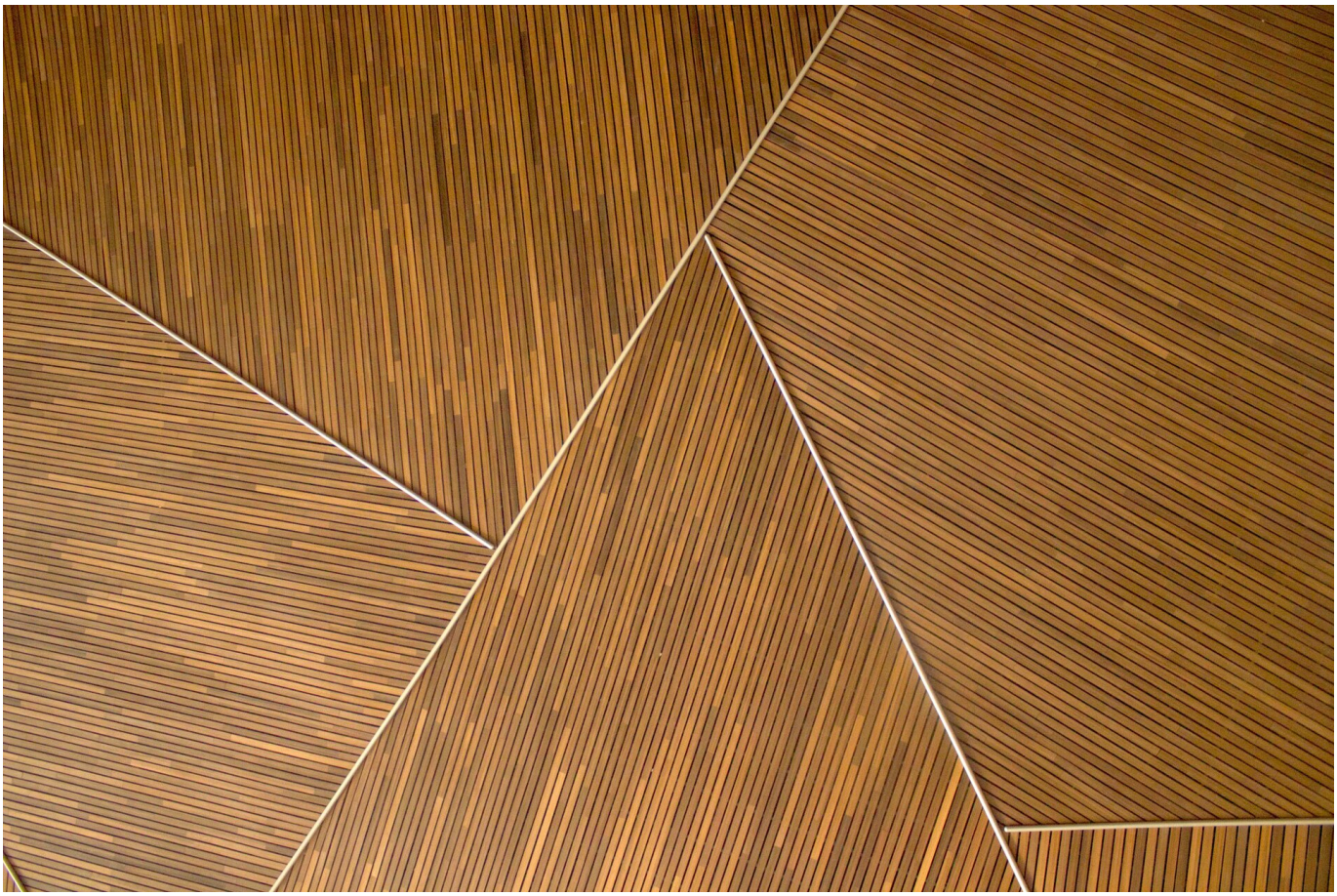
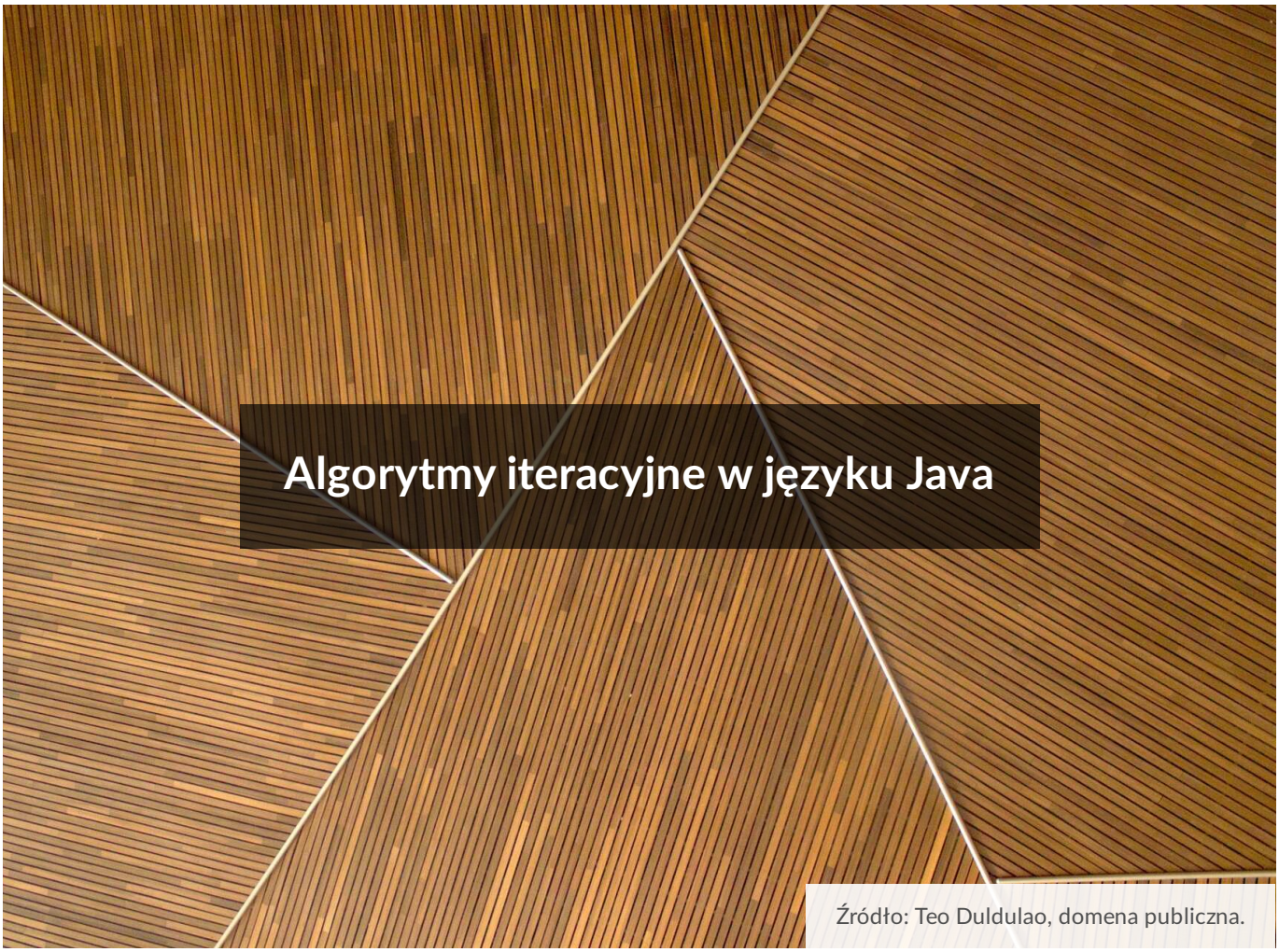


Algorytmy iteracyjne w języku Java



Źródło: Teo Duldulao, domena publiczna.

> W tym e-materiale powtarzamy wiadomości ze szkoły podstawowej.

Jak już wiemy z e-materiału Algorytmy iteracyjne, iteracja to powtarzanie zapisanych w programie instrukcji w pętli z góry określoną liczbę razy lub aż do spełnienia określonego warunku. Mechanizm ten jest podstawową operacją wykorzystywaną w programowaniu.

W tym e-materiale zapoznamy się z algorytmami iteracyjnymi w języku Java.

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

- Algorytmy iteracyjne w języku C++,
- Algorytmy iteracyjne w języku Python.

Więcej zadań? Sięgnij do Algorytmy iteracyjne – zadania maturalne.

> **Twoje cele**
>
> - Wyjaśnisz, czym jest iteracja i w jaki sposób zaimplementować ją w języku Java.
> - Przeanalizujesz składnie różnych rodzajów pętli języka Java.
> - Rozwiążesz kilka zadań, w których sprawdzisz swoje umiejętności z zakresu iteracji w języku Java.

Przeczytaj

Pętle w języku Java

W większości języków programowania to właśnie pętle służą do wykonywania iteracji. Umożliwiają one cykliczne powtarzanie zadanych operacji określoną liczbę razy lub do momentu spełnienia określonego warunku.

> **Ciekawostka**
>
> Istnieją języki programowania, w których nie ma pętli. Przykładem takiego języka jest Erlang. Zamiast pętli używa się w nim rekurencji.

> **Ciekawostka**
>
> Można także stworzyć pętlę wykonującą się nieskończenie wiele razy. Zazwyczaj taką pętlę konstruuje się, podając warunek, który zawsze będzie spełniony.

Suma parzystych wartości z przedziału

> **Problem 1**
>
> Napisz program w języku Java, który pobierze od użytkownika dwie liczby całkowite a i b, oznaczające granice przedziału $\langle a, b \rangle$. Zadaniem programu jest obliczyć sumę liczb parzystych z zadanego zakresu.
>
> **Specyfikacja problemu:**
>
> *Dane:*
>
> - a, b – liczby całkowite; kolejno: lewy i prawy kraniec przedziału, a $\leq$ b
>
> *Wynik:*
>
> Na standardowym wyjściu program wypisuje sumę liczb parzystych z zadanego przedziału.

Implementację algorytmu zaczynamy od zadeklarowania klasy `Rozwiazanie`. Definiujemy w niej metodę `main`.

```
1 public class Rozwiazanie {
2     public static void main(String[] args) {
```

```
3
4       }
5   }
```

Tworzymy nowy obiekt typu `Scanner`, za pomocą którego pobieramy od użytkownika lewy i prawy kraniec przedziału. Pamiętajmy o odpowiednim zaimportowaniu danych!

```
1  import java.util.Scanner;
2
3  public class Rozwiazanie {
4      public static void main(String[] args) {
5          Scanner sc = new Scanner(System.in);
6          System.out.println("Podaj lewy kraniec przedziału: ");
7          int a = sc.nextInt();
8
9          System.out.println("Podaj prawy kraniec przedziału: ");
10         int b = sc.nextInt();
11     }
12 }
```

Tworzymy zmienną pomocniczą `suma`, którą inicjalizujemy wartością 0.

```
1  import java.util.Scanner;
2
3  public class Rozwiazanie {
4      public static void main(String[] args) {
5          Scanner sc = new Scanner(System.in);
6          System.out.println("Podaj lewy kraniec przedziału: ");
7          int a = sc.nextInt();
8
9          System.out.println("Podaj prawy kraniec przedziału: ");
10         int b = sc.nextInt();
11
12         int suma = 0;
13     }
14 }
```

Tworzymy pętlę `for`. Jej iterator `i` inicjalizujemy wartością pobraną ze zmiennej `a`. Pętla będzie się wykonywać, dopóki wartość iteratora będzie mniejsza lub równa wartości

zapisanej w zmiennej `b`. Każdy przebieg pętli kończymy, zwiększając wartość zmiennej `i` o 1.

```
import java.util.Scanner;

public class Rozwiazanie {
    public static void main(String[] args) {
        Scanner sc = new Scanner(System.in);
        System.out.println("Podaj lewy kraniec przedziału: ");
        int a = sc.nextInt();

        System.out.println("Podaj prawy kraniec przedziału: ");
        int b = sc.nextInt();

        int suma = 0;
        for (int i = a; i <= b; i++) {

        }
    }
}
```

Wewnątrz pętli sprawdzamy, czy wartość zmiennej `i` jest podzielna przez 2 (czyli: czy jest parzysta). Jeżeli warunek jest prawdziwy, wówczas zwiększamy wartość zmiennej `suma` o `i`.

```
import java.util.Scanner;

public class Rozwiazanie {
    public static void main(String[] args) {
        Scanner sc = new Scanner(System.in);
        System.out.println("Podaj lewy kraniec przedziału: ");
        int a = sc.nextInt();

        System.out.println("Podaj prawy kraniec przedziału: ");
        int b = sc.nextInt();

        int suma = 0;
        for (int i = a; i <= b; i++) {
            if (i % 2 == 0) {
                suma += i;
            }
        }
```

```
    }
}
```

Po zakończeniu pętli wypisujemy wynik. W tym celu korzystamy z funkcji `System.out.println()`.

```
import java.util.Scanner;

public class Rozwiazanie {
    public static void main(String[] args) {
        Scanner sc = new Scanner(System.in);
        System.out.println("Podaj lewy kraniec przedziału: ");
        int a = sc.nextInt();

        System.out.println("Podaj prawy kraniec przedziału: ");
        int b = sc.nextInt();

        int suma = 0;
        for (int i = a; i <= b; i++) {
            if (i % 2 == 0) {
                suma += i;
            }
        }
        System.out.println(suma);
    }
}
```

Spróbujmy nieco zmodyfikować algorytm. Tym razem w naszej implementacji wykorzystamy pętlę `do ... while`.

> **Problem 2**
>
> Napisz program w języku Java, który pobierze od użytkownika dwie liczby całkowite a i b, oznaczające granice przedziału $\langle a, b \rangle$. Zadaniem programu jest obliczyć sumę liczb parzystych z zadanego zakresu. W swoim rozwiązaniu wykorzystaj pętlę `do ... while`.
>
> **Specyfikacja problemu:**
>
> *Dane:*
>
> - a, b – liczby całkowite; kolejno: lewy i prawy kraniec przedziału, a ≤ b
>
> *Wynik:*
>
> Na standardowym wyjściu program wypisuje sumę liczb parzystych z zadanego przedziału.

Ponownie deklarujemy klasę `Rozwiazanie`. We wnętrzu klasy tworzymy statyczną metodę `main`, wczytujemy klasę `Scanner` oraz pobieramy od użytkownika prawy i lewy zakres przedziału. Te wartości zapisujemy do zmiennych `a` i `b`.

```
1  import java.util.Scanner;
2
3  public class Rozwiazanie {
4      public static void main(String[] args) {
5          Scanner sc = new Scanner(System.in);
6          System.out.println("Podaj lewy kraniec przedziału: ");
7          int a = sc.nextInt();
8
9          System.out.println("Podaj prawy kraniec przedziału: ");
10         int b = sc.nextInt();
11     }
12 }
```

Tworzymy dwie zmienne pomocnicze – obie typu całkowitoliczbowego. Pierwsza: `suma` posłuży nam do przechowywania wszystkich dodanych do siebie liczb parzystych; druga: `i` będzie pełnić funkcje iteratora – wskaże, który obieg pętli jest aktualnie wykonywany. Zmienną `suma` inicjalizujemy wartością 0, z kolei do zmiennej `i` przypisujemy wartość zmiennej `a` (lewego krańca przedziału pobranego od użytkownika).

```java
import java.util.Scanner;

public class Rozwiazanie {
    public static void main(String[] args) {
        Scanner sc = new Scanner(System.in);
        System.out.println("Podaj lewy kraniec przedziału: ");
        int a = sc.nextInt();

        System.out.println("Podaj prawy kraniec przedziału: ");
        int b = sc.nextInt();

        int suma = 0;

        int i = a;
    }
}
```

Wykorzystujemy pętlę `do ... while`. Pętla ma się wykonywać, dopóki `i` będzie mniejsze lub równe `b`. Przypomnijmy, że wspomniana pętla sprawdza warunek kolejnego wykonania, po zakończeniu wszystkich operacji w jej wnętrzu.

```java
import java.util.Scanner;

public class Rozwiazanie {
    public static void main(String[] args) {
        Scanner sc = new Scanner(System.in);
        System.out.println("Podaj lewy kraniec przedziału: ");
        int a = sc.nextInt();

        System.out.println("Podaj prawy kraniec przedziału: ");
        int b = sc.nextInt();

        int suma = 0;

        int i = a;

        do {

        } while (i <= b);
    }
```

```
}
```

We wnętrzu pętli sprawdzamy, czy liczba `i` jest parzysta. Jeżeli warunek jest prawdziwy, wówczas dodajemy do zmiennej `suma` wartość zmiennej `i`. Pamiętajmy o zwiększeniu wartości zmiennej `i` po instrukcji warunkowej.

```
import java.util.Scanner;

public class Rozwiazanie {
    public static void main(String[] args) {
        Scanner sc = new Scanner(System.in);
        System.out.println("Podaj lewy kraniec przedziału: ");
        int a = sc.nextInt();

        System.out.println("Podaj prawy kraniec przedziału: ");
        int b = sc.nextInt();

        int suma = 0;

        int i = a;

        do {
            if (i % 2 == 0) {
                suma += i;
            }
            i++;
        } while (i <= b);
    }
}
```

Działanie algorytmu kończymy, wypisując wartość zmiennej `suma`.

```
import java.util.Scanner;

public class Main {
    public static void main(String[] args) {
        Scanner sc = new Scanner(System.in);
        System.out.println("Podaj lewy kraniec przedziału: ");
        int a = sc.nextInt();

```

```
        System.out.println("Podaj prawy kraniec przedziału: ");
        int b = sc.nextInt();

        int suma = 0;

        int i = a;

        do {
            if (i % 2 == 0) {
                suma += i;
            }
            i++;
        } while (i <= b);
        System.out.println(suma);
    }
}
```

Możemy poprawić efektywność rozwiązania. Jesteśmy w stanie znacznie zmniejszyć liczbę wykonań pętli, jeżeli się upewnimy, że lewy kraniec przedziału (liczba `a`) jest parzysty. Sprawdzenie tego warunku pozwala na zwiększenie iteratora `i` o 2, co przekłada się na redukcję liczby iteracji o połowę. Tym razem w naszej implementacji wykorzystamy pętlę `while`.

> **Problem 3**
>
> Napisz program w języku Java, który pobierze od użytkownika dwie liczby całkowite a i b, oznaczające granice przedziału $\langle a, b \rangle$. Zadaniem programu jest obliczyć sumę liczb parzystych z zadanego zakresu. W swoim rozwiązaniu wykorzystaj pętlę `while`. Spróbuj zredukować liczbę wykonań pętli.
>
> **Specyfikacja problemu:**
>
> *Dane:*
>
> - a, b – liczby całkowite; kolejno: lewy i prawy kraniec przedziału, a ≤ b
>
> *Wynik:*
>
> Na standardowym wyjściu program wypisuje sumę liczb parzystych z zadanego przedziału.

Nowy program zaczynamy od stworzenia klasy `Rozwiazanie`, umieszczamy w niej statyczną metodę `main` oraz pobieramy od użytkownika liczby `a` i `b`.

```
import java.util.Scanner;

public class Rozwiazanie {
    public static void main(String[] args) {
        Scanner sc = new Scanner(System.in);
        System.out.println("Podaj lewy kraniec przedziału: ");
        int a = sc.nextInt();

        System.out.println("Podaj prawy kraniec przedziału: ");
        int b = sc.nextInt();
    }
}
```

Następnie sprawdzamy, czy liczba `a` (lewy kraniec przedziału) jest nieparzysta. Jeżeli warunek okaże się prawdziwy, wówczas zwiększamy wartość `a` o 1.

```
import java.util.Scanner;

public class Rozwiazanie {
    public static void main(String[] args) {
        Scanner sc = new Scanner(System.in);
        System.out.println("Podaj lewy kraniec przedziału: ");
        int a = sc.nextInt();

        System.out.println("Podaj prawy kraniec przedziału: ");
        int b = sc.nextInt();

        if (a % 2 == 1) {
            a = a + 1;
        }
    }
}
```

Tworzymy dwie zmienne pomocnicze: `suma`, którą inicjalizujemy wartością 0, oraz `i`, do której przypisujemy wartość zmiennej `a`.

```

```

```java
import java.util.Scanner;

public class Rozwiazanie {
    public static void main(String[] args) {
        Scanner sc = new Scanner(System.in);
        System.out.println("Podaj lewy kraniec przedziału: ");
        int a = sc.nextInt();

        System.out.println("Podaj prawy kraniec przedziału: ");
        int b = sc.nextInt();

        if (a % 2 == 1) {
            a = a + 1;
        }

        int suma = 0;

        int i = a;
    }
}
```

Zgodnie z treścią zadania, wykorzystujemy pętlę `while`. Ma ona się wykonywać, dopóki wartość iteratora `i` będzie mniejsza lub równa `b`. We wnętrzu pętli dodajemy do zmiennej `suma` wartość zmiennej `i`. Ją samą zwiększamy o 2. Działanie programu kończymy, wypisując wartość zmiennej `suma`.

```java
import java.util.Scanner;

public class Rozwiazanie {
    public static void main(String[] args) {
        Scanner sc = new Scanner(System.in);
        System.out.println("Podaj lewy kraniec przedziału: ");
        int a = sc.nextInt();

        System.out.println("Podaj prawy kraniec przedziału: ");
        int b = sc.nextInt();

        if (a % 2 == 1) {
            a = a + 1;
        }

```

```
16          int suma = 0;
17
18          int i = a;
19
20          while (i <= b) {
21              suma += i;
22              i = i + 2;
23          }
24
25          System.out.println(suma);
26      }
27  }
```

Najszybszym sposobem na policzenie sumy liczb z przedziału jest wykorzystanie wzoru na sumę ciągu arytmetycznego. W omawianym przypadku suma S_a^b tylko liczb parzystych zawartych między parzystymi elementami a i b, gdzie a jest początkiem, zaś b końcem przedziału, jest równa:

$$S_a^b = \frac{(a+b)(b-a+2)}{4}$$

> **Problem 4**
>
> Napisz program w języku Java, który pobierze od użytkownika dwie liczby całkowite a i b, oznaczające granice przedziału $\langle a, b \rangle$. Zadaniem programu jest obliczyć sumę liczb parzystych z zadanego zakresu. W swoim rozwiązaniu wykorzystaj wzór na sumę ciągu arytmetycznego.
>
> **Specyfikacja problemu:**
>
> *Dane:*
>
> - a, b – liczby całkowite; kolejno: lewy i prawy kraniec przedziału, a $\leq$ b
>
> *Wynik:*
>
> Na standardowym wyjściu program wypisuje sumę liczb parzystych z zadanego przedziału.

Ponownie implementację algorytmu rozpoczynamy od stworzenia klasy `Rozwiazanie`, zadeklarowania w niej metody `main` oraz pobrania od użytkownika liczb `a` i `b`.

```
```

```java
import java.util.Scanner;

public class Rozwiazanie {
    public static void main(String[] args) {
        Scanner sc = new Scanner(System.in);
        System.out.println("Podaj lewy kraniec przedziału: ");
        int a = sc.nextInt();

        System.out.println("Podaj prawy kraniec przedziału: ");
        int b = sc.nextInt();
    }
}
```

Aby użyć wcześniej przedstawionego wzoru, krańce przedziału muszą być liczbami parzystymi. Sprawdzamy, czy zmienna `a` jest nieparzysta. Jeżeli warunek jest prawdziwy, wówczas zwiększamy jej wartość o 1. Podobną operację wykonujemy dla zmiennej `b` – w tym wypadku wartość zmniejszamy o 1.

```java
import java.util.Scanner;

public class Rozwiazanie {
    public static void main(String[] args) {
        Scanner sc = new Scanner(System.in);
        System.out.println("Podaj lewy kraniec przedziału: ");
        int a = sc.nextInt();

        System.out.println("Podaj prawy kraniec przedziału: ");
        int b = sc.nextInt();

        if (a % 2 == 1) {
            a = a + 1;
        }

        if (b % 2 == 1) {
            b = b - 1;
        }
    }
}
```

Tworzymy zmienną `suma`. Przypisujemy jej wartość obliczoną zgodnie z wcześniej przedstawionym wzorem. Działanie programu kończymy, wypisując wartość zmiennej `suma`.

```
import java.util.Scanner;

public class Rozwiazanie {
    public static void main(String[] args) {
        Scanner sc = new Scanner(System.in);
        System.out.println("Podaj lewy kraniec przedziału: ");
        int a = sc.nextInt();

        System.out.println("Podaj prawy kraniec przedziału: ");
        int b = sc.nextInt();

        if (a % 2 == 1) {
            a = a + 1;
        }

        if (b % 2 == 1) {
            b = b - 1;
        }

        int suma = ((a + b) * (b - a + 2)) / 4;

        System.out.println(suma);
    }
}
```

Wyszukiwanie minimum w zbiorze

> **Problem 5**
>
> Napisz program w języku Java, który pobierze od użytkownika n liczb całkowitych i wyłoni spośród nich najmniejszą.
>
> **Specyfikacja problemu:**
>
> *Dane:*
>
> - x1 , x2 , ... xn – kolejne liczby naturalne; liczba elementów wprowadzanych przez użytkownika; n $\geq$ 1
>
> *Wynik:*
>
> Na standardowym wyjściu program wypisuje najmniejszą spośród wprowadzonych liczb.

Implementację algorytmu zaczynamy od zadeklarowania klasy `Rozwiazanie`. Definiujemy w niej metodę `main`, wczytujemy klasę `Scanner` i za jej pomocą zapisujemy w zmiennej `n`, ile liczb chcemy wprowadzić.

```
1 import java.util.Scanner;
2
3 public class Rozwiazanie {
4     public static void main(String[] args) {
5         Scanner sc = new Scanner(System.in);
6         System.out.println("Podaj ile liczb chcesz wprowadzić: ")
7         int n = sc.nextInt();
8     }
9 }
```

Pobieramy od użytkownika pierwszą liczbę. Zapisujemy ją w zmiennej `x`. Deklarujemy zmienną `min` – będziemy w niej przechowywać aktualnie najmniejszą liczbę. Ponieważ jedyną wartością dotychczas wprowadzoną przez użytkownika jest liczba `x`, to właśnie ją przypisujemy do zmiennej `min`.

```
1 import java.util.Scanner;
2
3 public class Rozwiazanie {
```

```
    public static void main(String[] args) {
        Scanner sc = new Scanner(System.in);
        System.out.println("Podaj ile liczb chcesz wprowadzić: ")
        int n = sc.nextInt();

        System.out.println("Podaj liczbę:");
        int x = sc.nextInt();
        int min = x;
    }
}
```

Tworzymy pętlę `for`, której iterator `i` przyjmie wartość początkową 1 (pierwszą liczbę już pobraliśmy). Pętla będzie się wykonywać, dopóki `i < n`. Po każdym przebiegu pętli zwiększamy wartość iteratora `i` o 1.

```
import java.util.Scanner;

public class Rozwiazanie {
    public static void main(String[] args) {
        Scanner sc = new Scanner(System.in);
        System.out.println("Podaj ile liczb chcesz wprowadzić: ")
        int n = sc.nextInt();

        System.out.println("Podaj liczbę:");
        int x = sc.nextInt();
        int min = x;

        for (int i = 1; i < n; i++) {

        }
    }
}
```

Wewnątrz pętli pobieramy od użytkownika kolejne liczby. Jeżeli aktualnie wprowadzona liczba jest mniejsza od wartości zapisanej w zmiennej `min`, to do zmiennej `min` zapisujemy tę liczbę – staje się ona aktualnie najmniejszą. Działanie programu kończymy, wypisując wartość zmiennej `min`.

```
import java.util.Scanner;

```

```java
public class Rozwiazanie {
    public static void main(String[] args) {
        Scanner sc = new Scanner(System.in);
        System.out.println("Podaj ile liczb chcesz wprowadzić: ")
        int n = sc.nextInt();

        System.out.println("Podaj liczbę:");
        int x = sc.nextInt();
        int min = x;

        for (int i = 1; i < n; i++) {
            System.out.println("Podaj liczbę:");
            x = sc.nextInt();

            if (x < min) {
                min = x;
            }
        }

        System.out.println(min);
    }
}
```

Iteracyjne podejście do operacji wykonywanych na ciągach liczbowych

> **Problem 6**
>
> Napisz program w języku Java, który pobierze od użytkownika cztery liczby a, b, r i x. Liczby a i b będą oznaczać granice przedziału $\langle a, b \rangle$, w którym mają się zawierać elementy ciągu. Pierwszym elementem ciągu ma być liczba a. Każdy kolejny element ma różnić się od poprzedniego o wartość r. Zadaniem programu jest wypisanie kwadratów elementów ciągu, jeżeli są one większe od zadanego x, a następnie obliczenie i wypisanie ich sumy.
>
> **Specyfikacja problemu:**
>
> *Dane:*
>
> - a, b – liczby całkowite; kolejno: lewy i prawy kraniec przedziału, a ≤ b
> - x – liczba naturalna
> - r – różnica ciągu; liczba naturalna dodatnia
>
> *Wynik:*
>
> Na standardowym wyjściu program wypisze kwadraty elementów ciągu większe od x oraz ich sumę.

Implementację rozwiązania problemu rozpoczniemy od zadeklarowania wewnątrz klasy `Rozwiazanie` metody `main`. Wczytujemy klasę `Scanner` oraz pobieramy od użytkownika lewy i prawy kraniec przedziału (wartości te zapisujemy w zmiennych `a` i `b`), różnicę ciągu `r`, a także zmienną `x`, z którą będziemy porównywać kwadraty liczb. Jeśli będą one większe od `x`, będziemy je wypisywać i dodawać do zmiennej `suma`.

```
1 import java.util.Scanner;
2
3 public class Rozwiazanie {
4     public static void main(String[] args) {
5         Scanner sc = new Scanner(System.in);
6         System.out.println("Podaj lewy kraniec przedziału: ");
7         int a = sc.nextInt();
8
9         System.out.println("Podaj prawy kraniec przedziału: ");
```

```
        int b = sc.nextInt();

        System.out.println("Podaj r: ");
        int r = sc.nextInt();

        System.out.println("Podaj x: ");
        int x = sc.nextInt();
    }
}
```

Tworzymy zmienną pomocniczą `suma`. Inicjalizujemy ją wartością 0. Deklarujemy iterator `i`, któremu przypisujemy wartość `a`.

```
import java.util.Scanner;

public class Rozwiazanie {
    public static void main(String[] args) {
        Scanner sc = new Scanner(System.in);
        System.out.println("Podaj lewy kraniec przedziału: ");
        int a = sc.nextInt();

        System.out.println("Podaj prawy kraniec przedziału: ");
        int b = sc.nextInt();

        System.out.println("Podaj r: ");
        int r = sc.nextInt();

        System.out.println("Podaj x: ");
        int x = sc.nextInt();

        int suma = 0;

        int i = a;
    }
}
```

Deklarujemy pętlę `while`. Pętla ta ma się wykonywać, dopóki `i` będzie mniejsze bądź równe `b`.

```
import java.util.Scanner;
```

```
2
3  public class Rozwiazanie {
4      public static void main(String[] args) {
5          Scanner sc = new Scanner(System.in);
6          System.out.println("Podaj lewy kraniec przedziału: ");
7          int a = sc.nextInt();
8
9          System.out.println("Podaj prawy kraniec przedziału: ");
10         int b = sc.nextInt();
11
12         System.out.println("Podaj r: ");
13         int r = sc.nextInt();
14
15         System.out.println("Podaj x: ");
16         int x = sc.nextInt();
17
18         int suma = 0;
19
20         int i = a;
21
22         while (i <= b) {
23
24         }
25     }
26 }
```

Wewnątrz pętli deklarujemy zmienną `kwadrat`, której przypisujemy kwadrat liczby `i`. Jeżeli wartość zmiennej `kwadrat` jest większa od wartości `x`, to wypisujemy ją na standardowe wyjście oraz dodajemy do zmiennej `suma`. Na końcu bloku pętli zwiększamy wartość `i` o `r`.

```
1  import java.util.Scanner;
2
3  public class Rozwiazanie {
4      public static void main(String[] args) {
5          Scanner sc = new Scanner(System.in);
6          System.out.println("Podaj lewy kraniec przedziału: ");
7          int a = sc.nextInt();
8
9          System.out.println("Podaj prawy kraniec przedziału: ");
10         int b = sc.nextInt();
11
```

```
        System.out.println("Podaj r: ");
        int r = sc.nextInt();

        System.out.println("Podaj x: ");
        int x = sc.nextInt();

        int suma = 0;

        int i = a;

        while (i <= b) {
            int kwadrat = i * i;

            if (kwadrat > x) {
                System.out.println(kwadrat);
                suma += kwadrat;
            }

            i += r;
        }
    }
}
```

Implementację algorytmu kończymy, wypisując wartość zmiennej `suma`.

```
import java.util.Scanner;

public class Rozwiazanie {
    public static void main(String[] args) {
        Scanner sc = new Scanner(System.in);
        System.out.println("Podaj lewy kraniec przedziału: ");
        int a = sc.nextInt();

        System.out.println("Podaj prawy kraniec przedziału: ");
        int b = sc.nextInt();

        System.out.println("Podaj r: ");
        int r = sc.nextInt();

        System.out.println("Podaj x: ");
        int x = sc.nextInt();
```

```
17
18          int suma = 0;
19
20          int i = a;
21
22          while (i <= b) {
23              int kwadrat = i * i;
24
25              if (kwadrat > x) {
26                  System.out.println(kwadrat);
27                  suma += kwadrat;
28              }
29
30              i += r;
31          }
32
33          System.out.println(suma);
34      }
35  }
```

Rozwiązanie możemy jeszcze poprawić. Zauważmy, że po znalezieniu pierwszego kwadratu większego od `x`, kwadrat każdej kolejnej liczby `i` również będzie spełniał ten warunek. Możemy wówczas zrezygnować z każdorazowego sprawdzania warunku. Przykład:

```
1  import java.util.Scanner;
2
3  public class Rozwiazanie {
4      public static void main(String[] args) {
5          Scanner sc = new Scanner(System.in);
6          System.out.println("Podaj lewy kraniec przedziału: ");
7          int a = sc.nextInt();
8
9          System.out.println("Podaj prawy kraniec przedziału: ");
10         int b = sc.nextInt();
11
12         System.out.println("Podaj r: ");
13         int r = sc.nextInt();
14
15         System.out.println("Podaj x: ");
16         int x = sc.nextInt();
17
```

```
18          int suma = 0;
19
20          int i = a;
21
22          while (i <= b) {
23              int kwadrat = i * i;
24
25              if (kwadrat > x) {
26                  System.out.println(kwadrat);
27                  suma += kwadrat;
28                  i += r;
29                  break;
30              }
31
32              i += r;
33          }
34
35          for (; i <= b; i+=r) {
36              int kwadrat = i * i;
37              System.out.println(kwadrat);
38              suma += kwadrat;
39          }
40
41          System.out.println(suma);
42      }
43  }
```

Słownik

iteracja

czynność powtarzania tej samej operacji w pętli z góry określoną liczbę razy lub aż do spełnienia określonego warunku

pętla

jedna z konstrukcji programowania strukturalnego; pozwala na cykliczne wykonywanie instrukcji określoną liczbę razy, do momentu zajścia pewnych warunków, dla każdego elementu zbioru lub w nieskończoność

słowo kluczowe

w kontekście programowania oznacza słowo zdeklarowane w składni określonego języka programowania, mające szczególne znaczenie, oznaczające rozkaz, instrukcję, definicję lub deklarację w programie komputerowym; lista słów kluczowych jest zazwyczaj

> specyficzna dla konkretnego języka programowania; w języku Java przykładowe słowa kluczowe to: `int`, `for`, `static`, `public`, `private`, `class`

Gra edukacyjna

Polecenie 1

Sprawdź swoją wiedzę, biorąc udział w grze.

Test

Sprawdź swoją wiedzę o algorytmach iteracyjnych w języku Java, biorąc udział w grze.

Poziom trudności:

łatwy

Limit czasu:

7 min

Twój ostatni wynik:

-

Uruchom

Polecenie 2

Zastanów się, które pytania sprawiły ci trudność. Wróć do fragmentów, których dotyczą.

Polecenie 3

Zaproponuj trzy inne pytania, które mogłyby znaleźć się w grze edukacyjnej.

Sprawdź się

Pokaż ćwiczenia: 

Ćwiczenie 1

Napisz program, który wypisze n razy napis `ITERACJA` (każdy w nowej linii). W tym celu użyj pętli `for` lub pętli `while`. Przetestuj działanie swojego programu dla n równego `10`.

Specyfikacja problemu:

Dane:

- n – liczba naturalna określająca, ile razy napis `ITERACJA` ma zostać wypisany

Wynik:

Program wypisuje n razy napis `ITERACJA`, za każdym razem w nowej linii.

Twoje zadania

1. Program wypisuje napis `ITERACJA` n razy, każdy w nowej linii.

Ćwiczenie 2

Napisz program, który iteracyjnie obliczy n-ty wyraz ciągu a_n określonego następującym wzorem:

$$a_n = \begin{cases} 1 & \text{dla } n = 0 \\ 3a_{n-1} + 3 & \text{dla } n > 0 \end{cases}$$

Przetestuj działanie swojego programu dla n równego 7.

Specyfikacja problemu:

Dane:

- n – numer wyrazu ciągu; liczba naturalna

Wynik:

Program wypisuje wartość n-tego wyrazu ciągu.

Twoje zadania

1. Program ma wypisać wartość n-tego wyrazu ciągu a_n określonego wzorem:
   $$a_n = \begin{cases} 1 & \text{dla } n = 0 \\ 3a_{n-1} + 3 & \text{dla } n > 0 \end{cases}$$

Ćwiczenie 3

Napisz program, który obliczy wartość potęgi x^y, gdzie x jest sumą liczb podzielnych przez 5, a $\langle a, b \rangle$. Jeżeli potęga jest symbolem nieoznaczonym 0^0, program powininen wypisać komunikat „symbol nieoznaczony”. Przetestuj działanie swojego programu dla przedziału $\langle 2, 8 \rangle$.

Specyfikacja problemu:

Dane:

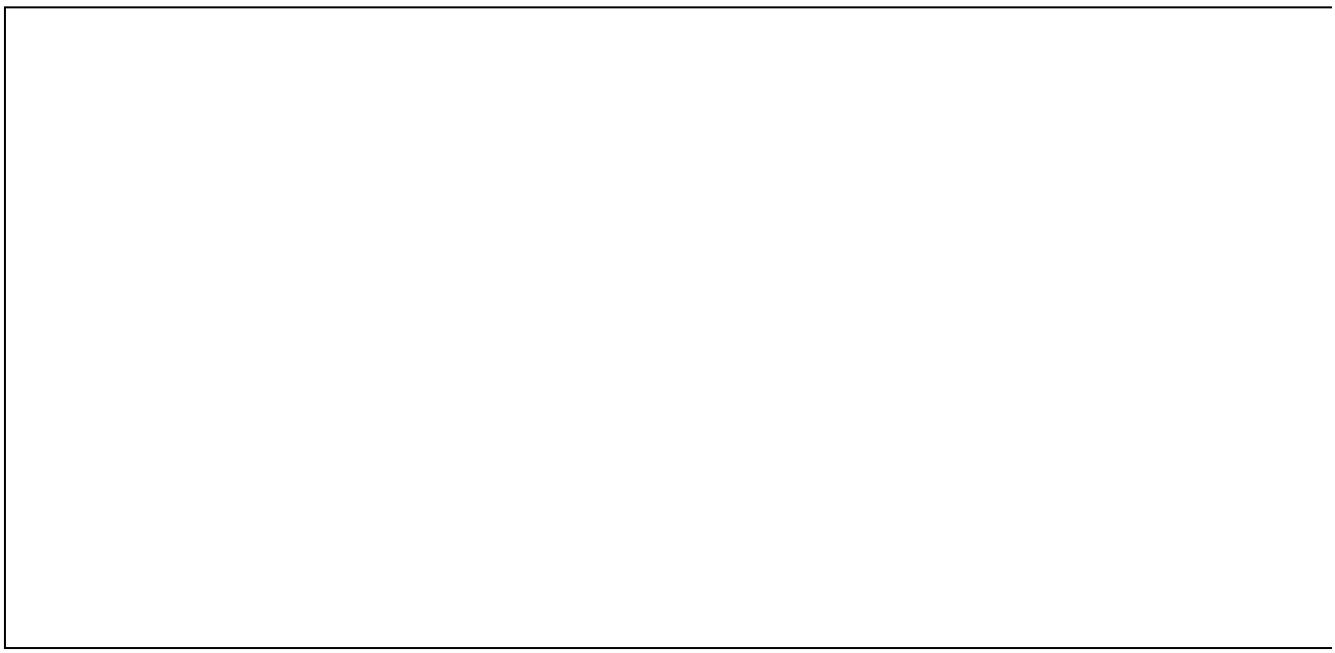
- a, b – krańce przedziału; liczby naturalne dodatnie

Wynik:

Program wypisuje wartość potęgi x^y, gdzie x jest sumą liczb podzielnych przez 5, a y największą liczbą podzielną przez 3 z zadanego przedziału $\langle a, b \rangle$ lub komunikat „symbol nieoznaczony”, gdy potęga jest symbolem nieoznaczonym 0^0.

Twoje zadania

1. Program wypisuje wartość potęgi x^y, gdzie x jest sumą liczb podzielnych przez 5, a y największą liczbą podzielną przez 3 z zadanego przedziału $<a, b>$, lub komunikat „symbol nieoznaczony” gdy potęga jest symbolem nieoznaczonym 0^0.



Dla nauczyciela

Autor: Maurycy Gast
Przedmiot: Informatyka

Temat: Algorytmy iteracyjne w języku Java

Grupa docelowa:

Liceum ogólnokształcące i technikum, liceum ogólnokształcące, technikum, zakres podstawowy

Podstawa programowa:

Cele kształcenia – wymagania ogólne

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 podstawowy. Uczeń:

2) stosuje przy rozwiązywaniu problemów z różnych dziedzin algorytmy poznane w szkole podstawowej oraz algorytmy:

e) obliczania wartości elementów ciągu metodą iteracyjną i rekurencyjną, w tym wartości elementów ciągu Fibonacciego.

II. Programowanie i rozwiązywanie problemów z wykorzystaniem komputera i innych urządzeń cyfrowych.

Zakres podstawowy. Uczeń:

1) projektuje i programuje rozwiązania problemów z różnych dziedzin, stosuje przy tym: instrukcje wejścia/wyjścia, wyrażenia arytmetyczne i logiczne, instrukcje warunkowe, instrukcje iteracyjne, funkcje z parametrami i bez parametrów, testuje poprawność programów dla różnych danych; w szczególności programuje algorytmy z punktu I.2);

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):

- Wyjaśnisz, czym jest iteracja i w jaki sposób zaimplementować ją w języku Java.
- Przeanalizujesz składnie różnych rodzajów pętli języka Java.
- Rozwiążesz kilka zadań, w których sprawdzisz swoje umiejętności z zakresu iteracji w języku Java.

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 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 iteracyjne w języku Java". Uczniowie zapoznają się z treściami w sekcji „Przeczytaj".

Faza wstępna:

1. Nauczyciel wyświetla i odczytuje temat lekcji oraz cele zajęć. Prosi uczniów o sformułowanie kryteriów sukcesu.
2. **Rozpoznanie wiedzy uczniów.** Uczniowie tworzą pytania dotyczące tematu zajęć, na które odpowiedzą w trakcie lekcji.

Faza realizacyjna:

1. Odnosząc się do treści zawartej w sekcji „Przeczytaj", uczniowie w parach konstruują alternatywny przykład, definiując samodzielnie problem, rozwiązanie i ewentualną implementację. Rezultaty omawiane są na forum klasy.
2. **Praca z multimedium.** Uczniowie indywidualnie zapoznają się z treścią polecenia nr 1 z sekcji „Gra edukacyjna" i odpowiadają na kolejne pytania gry.
3. **Ćwiczenie umiejętności.** Uczniowie wykonują ćwiczenia nr 1-2 z sekcji „Sprawdź się". Nauczyciel sprawdza poprawność wykonanych zadań, omawiając je wraz z uczniami.

Faza podsumowująca:

1. Nauczyciel ponownie wyświetla na tablicy temat i cele lekcji zawarte w sekcji „Wprowadzenie". W kontekście ich realizacji następuje omówienie ewentualnych problemów z rozwiązaniem ćwiczeń z sekcji „Sprawdź się".
2. Wybrany uczeń podsumowuje zajęcia, zwracając uwagę na nabyte umiejętności, omawia ewentualne problemy podczas rozwiązania ćwiczeń z programowania w języku Java.

Praca domowa:

1. Uczniowie wykonują ćwiczenie 3 z sekcji „Sprawdź się".
2. Uczniowie wykonują polecenie 2 z sekcji „Gra edukacyjna".

Materiały pomocnicze:

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

Wskazówki metodyczne:

- Uczniowie mogą wykorzystać treści w sekcjach: „Przeczytaj", „Gra edukacyjna", „Sprawdź się" jako materiał do lekcji powtórkowej.