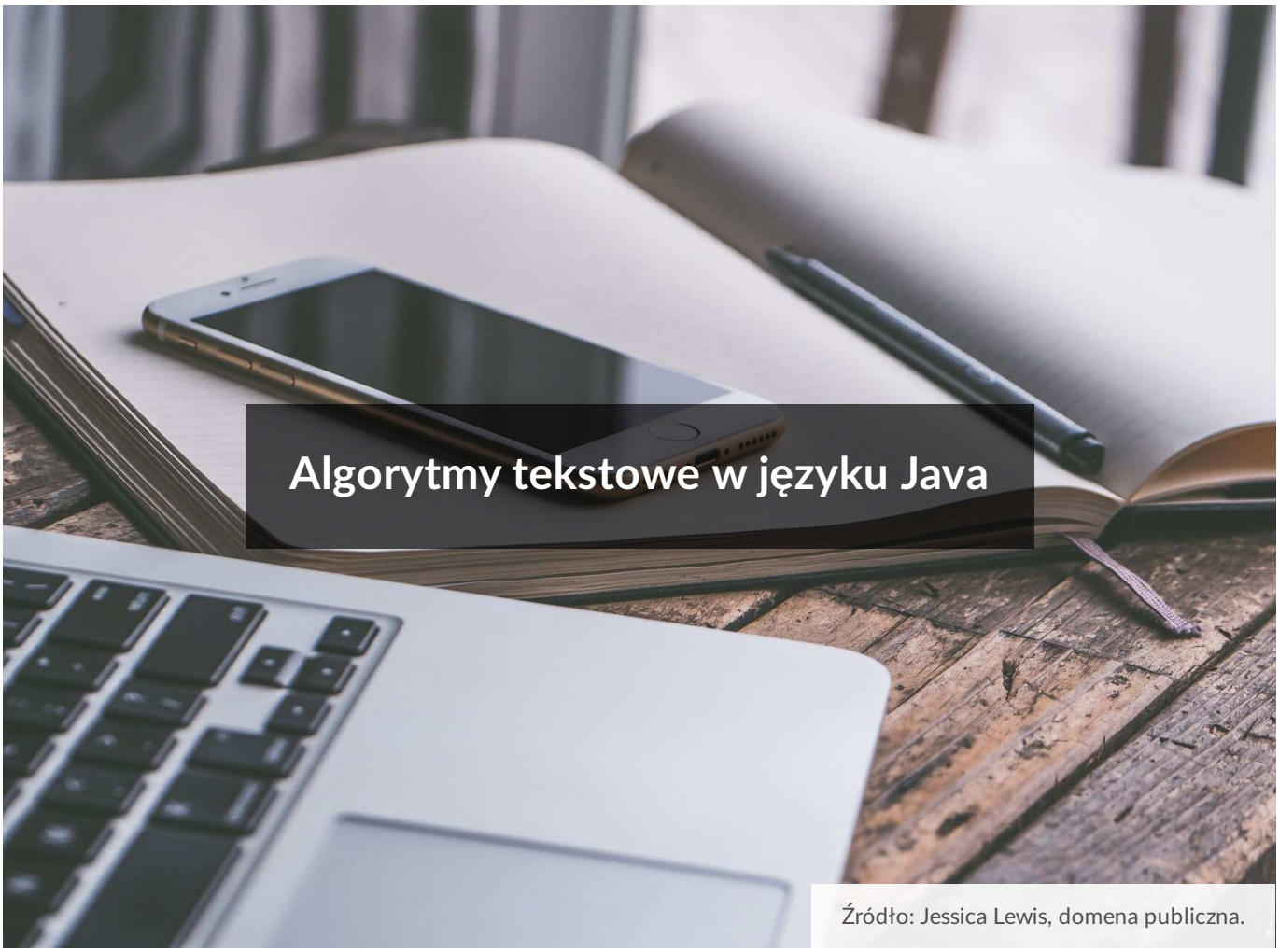
Algorytmy tekstowe w języku Java



Źródło: Jessica Lewis, domena publiczna.

Algorytmów tekstowych używamy do przetwarzania oraz przeszukiwania danych tekstowych. Jeden z nich już poznaliśmy, to algorytm Knutha-Morrisa-Pratta.

Używamy algorytmów tekstowych, korzystając z edytora tekstu czy klienta e-mail. Znajdują one również zastosowanie w grach. Możemy je wykorzystać, projektując implementację popularnej gry w wisielca. W e-materiale Algorytmy tekstowe omówiliśmy jej zasady oraz opracowaliśmy odpowiedni pseudokod.

W tym e-materiale zajmiemy się implementacją tej gry w języku Java.

Implementację algorytmów tekstowych w pozostałych językach programowania znajdziesz w e-materiałach:

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

Twoje cele

- Przeanalizujesz i utrwalisz zasady gry w wisielca.
- Zaimplementujesz algorytm gry w wisielca, przy wykorzystaniu języka Java.
- Wykonasz ćwiczenia z programowania w języku Java, w tematyce algorytmów tekstowych.

- Scharakteryzujesz wybrane algorytmy tekstowe.

Animacja

Problem 1

Napisz program symulujący grę w wisielca.

Specyfikacja:

Założenia:

- nie rysuj szubienicy – liczbę szans przechowuj w zmiennej;
- komputer losuje słowo z wcześniej przygotowanej tablicy;
- gracz zgaduje słowo poprzez wprowadzanie kolejnych liter;
- gra kończy się, gdy gracz zgadnie słowo w całości lub gdy zabraknie mu szans.

Dane:

- `slowa` – tablica łańcuchów znaków
- `liczbaBledow` – liczba naturalna dodatnia

Wynik:

Program w przypadku podania przez użytkownika litery występującej w słowie będzie wyświetlał niepełny napis uzupełniony o podaną literę lub odejmował pozostałe szanse w przypadku litery niewystępującej w danym słowie.

```
1 public static void main(String[] args){
2     // Tutaj dodaj własny kod. Użyj funkcji
3     // System.out.print();
4   }
```

```
1
```

Polecenie 1

Porównaj swoje rozwiązanie z programem przedstawionym w filmie.

Film dostępny pod adresem /preview/resource/RvzRr5dzYWoPh

Film nawiązujący do treści materiału: Algorytmy tekstowe - gra w wisielca.

Kod programu zaprezentowanego w filmie:

Plik o rozmiarze 1.96 KB w języku polskim

Problem 2

Zaimplementuj algorytm wyszukujący w zadanym przez użytkownika ciągu znaków wszystkie podciągi o zadanej długości, w których wszystkie litery są różne.

Specyfikacja:

Dane:

- `ciagZnakow` – łańcuch znaków
- `dlPodlancucha` – liczba naturalna; zmienna typu *int*

Wynik:

Program na wyjściu standardowym wypisze podciągi o zadanej długości, w których wszystkie litery są różne.

```
public static void main(String[] args){
    // Tutaj dodaj własny kod. Użyj funkcji
    // System.out.print();
  }
```

```
```

Polecenie 2

Porównaj swoje rozwiązanie z animacją.

Film dostępny pod adresem /preview/resource/R16MxBlrHWSdv

Film nawiązujący do treści materiału: Algorytmy tekstowe.

Przeczytaj

Implementacja algorytmu w języku Java – podsumowanie

> **Ważne!**
>
> W tej sekcji zapoznasz się z alternatywną (do zaprezentowanej w poprzedniej sekcji) implementacją gry w wisielca.

Pierwszym krokiem jest zadeklarowanie dwóch zmiennych:

```
1 int pozostaleProby = 8;
2 String wymyslone = "";
```

Pierwsza z nich o nazwie `pozostaleProby` to zmienna całkowita `integer`, która określa, ile prób pozostało graczowi na odgadnięcie słowa wymyślonego przez innego gracza (łańcuch znaków `String wymyslone`).

Kolejnym krokiem jest pobranie od użytkownika (w tym przypadku gracza) słowa, które drugi gracz będzie miał za zadanie odgadnąć. Ciąg znaków, który uzyskujemy za pomocą klasy `Scanner` oraz funkcji wbudowanej `nextLine()`, zapisujemy do zmiennej `wymyslone`.

```
1 Scanner scanner = new Scanner(System.in);
2 wymyslone = scanner.nextLine();
```

> **Ważne!**
>
> Ponieważ użyta została zewnętrza klasa `Scanner`, należy pamiętać o dodaniu:
>
> ```
> 1 import java.util.Scanner;
> ```

Deklarujemy tablicę znaków `char[]` o nazwie `odgadywane`, która będzie przechowywać ciąg znaków zawierający litery (w przypadku gdy gracz odgadł daną literę) oraz znaki podkreślenia (jeżeli dana litera nie została jeszcze odgadnięta).

Na samym początku gry nie są jeszcze odgadnięte żadne litery, więc wypełniamy zmienną `odgadywane` znakami podkreślenia.

Zmienna następnie jest wypisywana na ekran, aby drugi gracz znał długość słowa.

```
char[] odgadywane = new char[wymyslone.length()];

for(int i = 0; i < wymyslone.length(); i++) {
    odgadywane[i] = '_';
}
```

Następnie deklarujemy kolejne dwie zmienne:

- `boolean czyOdgadniete = false;` - zmienna logiczna, określająca, czy gracz odgadł już pełne słowo (przyjmie wartość `true`), czy nie (przyjmie wartość `false`),
- `String proba;` - łańcuch znaków, do którego zapisywać będziemy to, co wprowadzi drugi gracz - literę lub całe słowo, które będzie chciał sprawdzić.

```
boolean czyOdgadniete = false;
String proba;
```

Deklarujemy pętlę `while`, w której będziemy pobierać od użytkownika kolejne litery (bądź całe słowa) i sprawdzać czy występują w wymyślonym przez pierwszego gracza wyrazie.

```
while(pozostaleProby > 0) {
    System.out.println("Podaj literę lub całe słowo do sprawdzeni
    proba = scanner.nextLine();
    boolean czyZawiera = false;
```

Tworzymy blok instrukcji warunkowej, która sprawdza, czy użytkownik wprowadził pojedynczy znak (długość równa 1), czy ciąg znaków (długość większa od 1).

W przypadku, gdy został wprowadzony pojedynczy znak, w pętli zostaje sprawdzone, czy wymyślone słowo zawiera podany znak – jeżeli tak, w kolejnej instrukcji warunkowej w tablicy znaków `odgadywane` zamieniane są znaki podkreślenia (w odpowiednich miejscach) na rzeczywiste litery. W przeciwnym przypadku (gdy użytkownik podał literę, która nie znajduje się wymyślonym przez pierwszego gracza wyrazie) wartość zmiennej `pozostaleProby` zostaje zmniejszona o 1.

```
while(pozostaleProby > 0) {
    System.out.println("Podaj literę lub całe słowo do sprawdzeni
    proba = scanner.nextLine();
    boolean czyZawiera = false;

```

```
    if(proba.length() == 1) {

        for(int i = 0; i < wymyslone.length(); i++) {
            if(wymyslone.charAt(i) == proba.charAt(0)) {
                czyZawiera = true;
                break;
            }
        }

        if(czyZawiera) {
            for (int i = 0; i < wymyslone.length(); i++) {
                if(wymyslone.charAt(i) == proba.charAt(0)) {
                    odgadywane[i] = proba.charAt(0);
                }

            }
        } else {
            pozostaleProby--;
            System.out.println("Słowo nie zawiera litery " + prob
        }
    }
```

Następnie należy sprawdzić, czy słowo już zostało odgadnięte w całości. Jeżeli nie zawiera ono znaków podkreślenia, wszystkie litery są już znane i można zakończyć proces odgadywania kolejnych liter.

```
while(pozostaleProby > 0) {
    System.out.println("Podaj literę lub całe słowo do sprawdzeni
    proba = scanner.nextLine();
    boolean czyZawiera = false;

    if(proba.length() == 1) {

        for(int i = 0; i < wymyslone.length(); i++) {
            if(wymyslone.charAt(i) == proba.charAt(0)) {
                czyZawiera = true;
                break;
            }
        }

        if(czyZawiera) {
```

```
            for (int i = 0; i < wymyslone.length(); i++) {
                if(wymyslone.charAt(i) == proba.charAt(0)) {
                    odgadywane[i] = proba.charAt(0);
                }

            }
        } else {
            pozostaleProby--;
            System.out.println("Słowo nie zawiera litery " + prob
        }

        czyOdgadniete = true;

        for (int i = 0; i < odgadywane.length; i++) {
            if (odgadywane[i] == '_') {
                czyOdgadniete = false;
                break;
            }
        }

        if (czyOdgadniete == true) {
            break;
        }
    } else if(proba.length() > 1) {

    }
}
```

Zdefiniujmy teraz operacje, które mają zostać wykonane w przypadku, gdy nie została wprowadzona pojedyncza litera, tylko ciąg znaków.

```
while(pozostaleProby > 0) {
    System.out.println("Podaj literę lub całe słowo do sprawdzeni
    proba = scanner.nextLine();
    boolean czyZawiera = false;

    if(proba.length() == 1) {

        for(int i = 0; i < wymyslone.length(); i++) {
            if(wymyslone.charAt(i) == proba.charAt(0)) {
                czyZawiera = true;
```

```
                break;
            }
        }

        if(czyZawiera) {
            for (int i = 0; i < wymyslone.length(); i++) {
                if(wymyslone.charAt(i) == proba.charAt(0)) {
                    odgadywane[i] = proba.charAt(0);
                }

            }
        } else {
            pozostaleProby--;
            System.out.println("Słowo nie zawiera litery " + prob
        }

        czyOdgadniete = true;

        for (int i = 0; i < odgadywane.length; i++) {
            if (odgadywane[i] == '_') {
                czyOdgadniete = false;
                break;
            }
        }

        if (czyOdgadniete == true) {
            break;
        }
    } else if(proba.length() > 1) {
        if(proba.equals(wymyslone)) {
            czyOdgadniete = true;
            break;
        } else {
            pozostaleProby--;
            System.out.println("Pozostałe życia: " + pozostalePro
        }
    }
}
```

Sprawdzane jest, czy ciąg znaków, który podał gracz, zgadza się z wymyślonym słowem. Jeżeli tak, algorytm kończy działanie. W przeciwnym przypadku zmniejsza się liczba

pozostałych prób.

Po wyjściu z pętli `while` wyświetlany jest końcowy komunikat o wynikach rozgrywki:

```
if (!czyOdgadniete) {
    System.out.println("Niestety nie udało ci się odgadnąć słowa"
} else {
    System.out.println("Udało ci się odgadnąć słowo!");
}
```

Oto kod całego programu:

```
import java.util.Scanner;

public class GraWisielec {

    public static void main(String[] args) {

        int pozostaleProby = 8;
        String wymyslone = "";

        Scanner scanner = new Scanner(System.in);
        wymyslone = scanner.nextLine();

        char[] odgadywane = new char[wymyslone.length()];

        for(int i = 0; i < wymyslone.length(); i++) {
            odgadywane[i] = '_';
        }

        System.out.println(odgadywane);
        boolean czyOdgadniete = false;
        String proba;

        while(pozostaleProby > 0) {
            System.out.println("Podaj literę lub całe słowo do sp
            proba = scanner.nextLine();
            boolean czyZawiera = false;

            if(proba.length() == 1) {
```

```
29
30                  for(int i = 0; i < wymyslone.length(); i++) {
31                      if(wymyslone.charAt(i) == proba.charAt(0)) {
32                          czyZawiera = true;
33                          break;
34                      }
35                  }
36
37                  if(czyZawiera) {
38                      for (int i = 0; i < wymyslone.length(); i++)
39                          if(wymyslone.charAt(i) == proba.charAt(0)
40                              odgadywane[i] = proba.charAt(0);
41                          }
42
43                      }
44                  } else {
45                      pozostaleProby--;
46                      System.out.println("Słowo nie zawiera litery
47                  }
48
49                  czyOdgadniete = true;
50
51                  for (int i = 0; i < odgadywane.length; i++) {
52                      if (odgadywane[i] == '_') {
53                          czyOdgadniete = false;
54                          break;
55                      }
56                  }
57
58                  if (czyOdgadniete == true) {
59                      break;
60                  }
61              } else if(proba.length() > 1) {
62                  if(proba.equals(wymyslone)) {
63                      czyOdgadniete = true;
64                      break;
65                  } else {
66                      pozostaleProby--;
67                      System.out.println("Pozostałe życia: " + pozo
68                  }
69              }
70
```

```
71              System.out.println(odgadywane);
72              System.out.println("Pozostałe życia: " + pozostalePro
73          }
74
75          if (!czyOdgadniete) {
76              System.out.println("Niestety nie udało ci się odgadną
77          } else {
78              System.out.println("Udało ci się odgadnąć słowo!");
79          }
80      }
81  }
```

Słownik

instrukcja warunkowa

element algorytmu pozwalający na sprawdzenie jednego lub kilku warunków, a następnie zdefiniowanie, jakie czynności mają być wykonane, jeśli dane warunki są spełnione lub niespełnione (służy do sterowania programem)

łańcuch znaków

ciąg znaków o ustalonej wcześniej lub dowolnej długości, gdzie znaki to kolejno zapisane bajty w pamięci

Sprawdź się

Pokaż ćwiczenia: 

Ćwiczenie 1

Wyobraźmy sobie następującą grę: dwóch graczy naprzemiennie wypowiada słowo. Musi ono rozpoczynać się literą, którą kończyło się poprzednie słowo. Napisz program, który będzie symulował zachowanie jednego z graczy, wykorzystując podaną tablicę znanych słów, z których wybierze słowo, którego pierwsza litera będzie zarazem ostatnią literą słowa podanego przez gracza zapisanego w zmiennej `lancuchSlow`.

Przetestuj działanie programu dla następujących danych:

```
znaneSlowa = {"arbuz", "lokomotywa", "wiatrak", "krowa", "mleko
lancuchSlow = "wiadro"
lancuchSlow = "rebus"
```

Specyfikacja:

Dane:

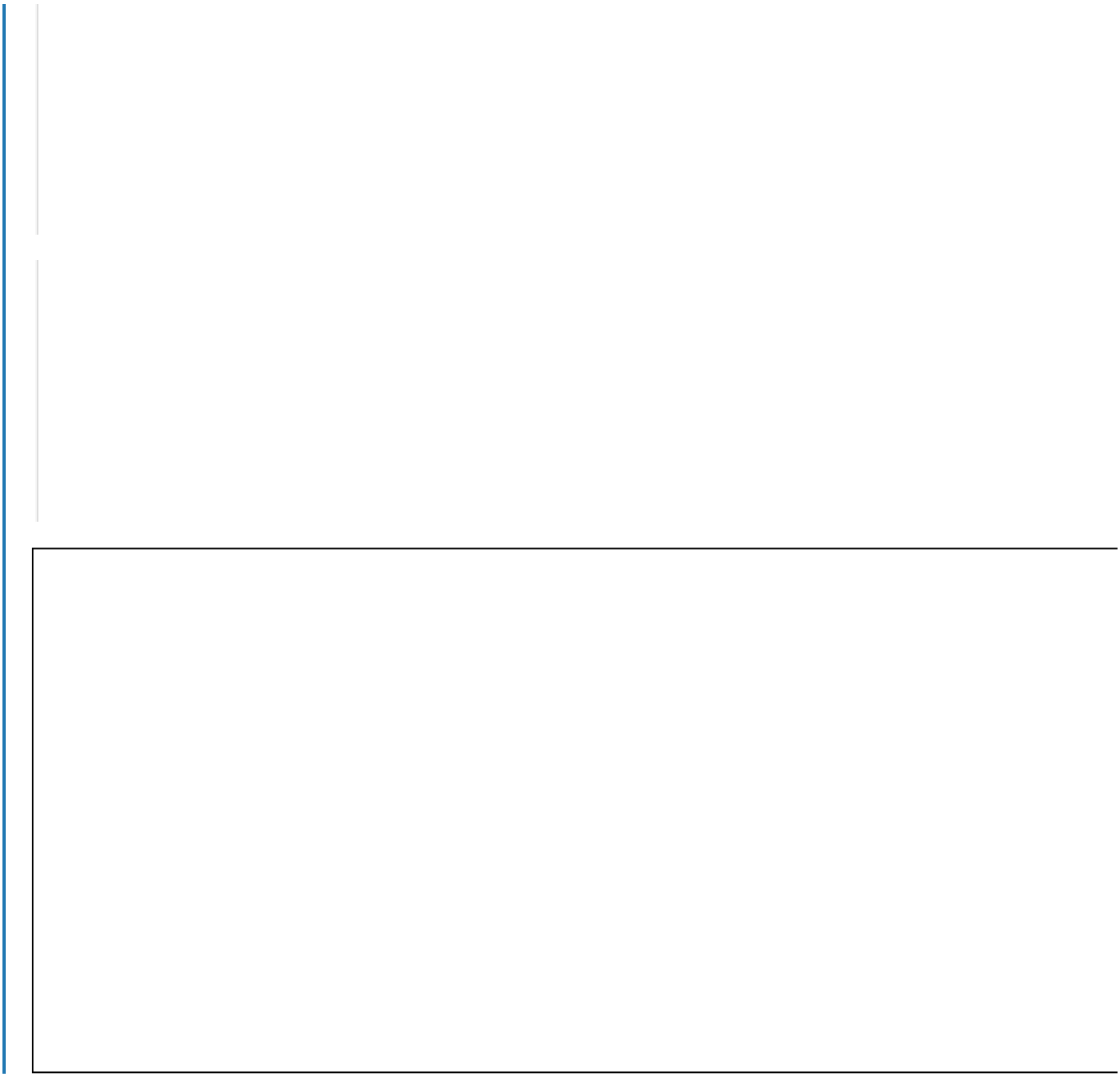
- `znaneSlowa` – tablica łancuchów znaków

Wynik:

Program wyświetla łańcuch znaków zgodny z regułami gry (pobrany z tablicy) lub łańcuch znaków `PRZEGRYWAM`, jeśli tablica nie zawiera łańcucha znaków zgodnego z regułami gry.

Twoje zadania

1. Program drukuje wynik działania funkcji `lancuchSlow` dla wartości `wiadro`, `rebus`.



Ćwiczenie 2

Napisz program, który sprawdzi, czy ze znaków danego łańcucha znaków (`slowoBazowe`) można ułożyć inny łańcuch znaków (`slowo`). Funkcja sprawdzająca powinna przyjmować dwa argumenty. Program powinien wyświetlać wartość `true`, jeśli możliwe jest ułożenie danego łańcucha, lub `false` w przeciwnym wypadku.

Działanie programu przetestuj dla następujących par słów (pierwsze słowo to `slowo`, drugie słowo to `slowoBazowe`).

```
motyka, lokomotywa
loki, lokomotywa
wakat, lokomotywa
```

Specyfikacja:

Dane:

- `slowo, slowoBazowe` – łancuchy znaków

Wynik:

Program drukuje na standardowe wyjście wynik funkcji `sprawdz` (`true` lub `false`) dla przykładowych łańcuchów znaków.

Twoje zadania

1. Program drukuje na standardowe wyjście wartości `true` lub `false` dla przykładowych słów.

Ćwiczenie 3

Przeprowadzana jest gra w „wisielca” – gracz próbuje odgadnąć słowo, którego litery są niewidoczne. Aby tego dokonać, może wybrać literę, która zostanie odsłonięta. Jeśli litera nie występuje w danym słowie, gracz traci jeden punkt życia. Dany jest algorytm, który gra w wisielca według następujących zasad:

1. Pierwsza sprawdzana litera to A (pozycja w alfabecie 0).
2. Jeżeli sprawdzana litera, której pozycja w alfabecie to n, występuje w słowie k razy, następną sprawdzaną literą będzie litera na pozycji $n \cdot 2 + k + 1$.
3. Jeśli dana litera nie występuje w słowie, sprawdzana jest litera na pozycji $n + 1$.
4. Jeżeli pozycja kolejnej litery przekracza zakres alfabetu (Z występuje na pozycji 25), wykonujemy dzielenie modulo 26 aktualnej pozycji.
5. W przypadku natrafienia na już sprawdzoną literę, nie sprawdzamy jej ponownie, a następną sprawdzaną literą będzie litera na pozycji $n + 1$.

Sprawdź, ile razy dla podanych w tablicy słów, algorytm będzie w stanie odgadnąć słowo. Zakładamy, że do dyspozycji jest 16 punktów życia.

Specyfikacja:

Dane:

- `slowa` – tablica łańcuchów znaków

Wynik:

Program przeprowadza symulację gry w „wisielca” oraz liczy, ile razy udało się wygrać określonym algorytmem.

Twoje zadania

1. Program przeprowadza symulację gry w „wisielca” oraz liczy, ile razy udało się wygrać określonym algorytmem.

Dla nauczyciela

Autor: zespół autorski Contentplus.pl sp. z o.o.
Przedmiot: Informatyka

Temat: Algorytmy tekstowe 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 podstawowy. Uczeń:

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

b) na tekstach: porównywania tekstów, wyszukiwania wzorca w tekście metodą naiwną, szyfrowania tekstu metodą Cezara i przestawieniową,

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

2) do realizacji rozwiązania problemu dobiera odpowiednią metodę lub technikę algorytmiczną i struktury danych;

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

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

2) stosuje zasady programowania strukturalnego i obiektowego w rozwiązywaniu problemów;

3) sprawnie posługuje się zintegrowanym środowiskiem programistycznym przy pisaniu, uruchamianiu i testowaniu programów;

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):

- Przeanalizujesz i utrwalisz zasady gry w wisielca.
- Zaimplementujesz algorytm gry w wisielca, przy wykorzystaniu języka Java.
- Wykonasz ćwiczenia z programowania w języku Java, w tematyce algorytmów tekstowych.
- Scharakteryzujesz wybrane algorytmy tekstowe.

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 tekstowe w języku Java". Uczniowie mają zapoznać się z treściami w sekcji „Przeczytaj".

Faza wstępna:

1. Nauczyciel wyświetla temat zajęć oraz cele zawarte w sekcji „Wprowadzenie". Prosi, by na ich podstawie uczniowie sformułowali kryteria sukcesu.
2. Uczniowie dyskutują na temat wykorzystania algorytmów tekstowych.

Faza realizacyjna:

1. **Praca z tekstem.** Nauczyciel ocenia, na podstawie informacji na platformie, stan przygotowania uczniów do zajęć. Jeżeli jest ono niewystarczające, prosi wybraną osobę o przedstawienie najważniejszych informacji z sekcji „Przeczytaj".
2. **Praca z multimedium**. Uczniowie w parach rozwiązują problem 1 z sekcji „Animacja". W grupach porównują swoje rozwiązania. Chętna lub wybrana osoba prezentuje kod na forum klasy. Następnie uczniowie porównują swoje rozwiązanie z przedstawionym w filmie. W kolejnym kroku uczniowie indywidualnie zapoznają się z problemem 2. Próbują napisać samodzielnie kod, w razie problemów prosząc nauczyciela o pomoc. Następnie porównują swoje rozwiązanie z przedstawionym w animacji.
3. **Ćwiczenie umiejętności.** Uczniowie szukają najefektywniejszego rozwiązania problemu. Uczniowie wykonują w parach ćwiczenie nr 1 z sekcji „Sprawdź się", a następnie porównują swój kod, omawiając go wspólnie na forum. Nauczyciel ocenia efektywność zastosowanego rozwiązania.
4. Liga zadaniowa – uczniowie pracując w parach, wykonują ćwiczenie nr 2 z sekcji „Sprawdź się", a następnie dzielą się swoimi wynikami przez porównywanie napisanego kodu z inną grupą, która również zakończyła zadanie.

Faza podsumowująca:

1. Nauczyciel ponownie wyświetla na tablicy temat lekcji zawarty w sekcji „Wprowadzenie" i inicjuje krótką rozmowę na temat zrealizowanych celów (czego uczniowie się nauczyli).

Praca domowa:

1. Uczniowie wykonują ćwiczenie 3 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 „Animacja” można wykorzystać jako podsumowanie i utrwalenie wiedzy uczniów.