Anagramy

Źródło: Pixabay, domena publiczna.

Anagramy są wyrazami, wyrażeniami lub całymi zdaniami, które powstają wskutek przestawienia liter bądź sylab innych wyrazów lub zdań (z wykorzystaniem wszystkich liter wchodzących w skład tekstu oryginalnego). Przykładowymi anagramami są wyrazy „arbuz” i „burza”. Przestawiając litery w słowie „arbuz” ułożymy wyraz „burza” (nie dodając ani nie pomijając żadnego znaku).

Implementację w wybranych językach programowania znajdziesz w e-materiałach:

- Anagramy w języku C++,
- Anagramy w języku Java,
- Anagramy w języku Python.

Więcej zadań? Przejdź do e-materiału Anagramy – zadania maturalne.

> **Twoje cele**
>
> - Nauczysz się rozpoznawać anagramy.
> - Przeanalizujesz algorytmy wykorzystywane do sprawdzania, czy dwa wyrazy są anagramami.
> - Rozwiążesz zadania polegające na sprawdzeniu, czy podane słowa są anagramami.

Przeczytaj

Anagramy – definicje i sposób analizowania

Prostym sposobem sprawdzenia, czy dwa wyrazy są anagramami, jest próba ułożenia wszystkich liter składających się na jeden wyraz w taki sposób, aby otrzymać drugie słowo. Jeżeli próba zakończy się powodzeniem, uzyskujemy pewność, że sprawdzane wyrazy są anagramami.

Wykorzystując przedstawioną metodę sprawdź, czy w zapisanych niżej parach jeden wyraz jest anagramem drugiego:

- wektor – wtorek
- makrela – reklama
- łoś – stolik
- bryka – rybak

Trzy spośród czterech podanych par wyrazów są anagramami. Nie dotyczy to tylko wyrazów „łoś" i „stolik". W przypadku tej właśnie pary od razu stwierdzimy, że jeden wyraz nie może być anagramem drugiego. Powód jest oczywisty: liczba liter w obu słowach jest różna. Nie ma znaczenia, w jaki sposób będziemy je przestawiać – na pewno nie uda nam się przekształcić jednego wyrazu w drugi.

Słowa będące anagramami zawierają taką samą liczbę identycznych liter. Wykorzystamy to spostrzeżenie podczas projektowania algorytmu pozwalającego sprawdzić, czy dwa wyrazy są anagramami.

Anagramami mogą być nie tylko pary wyrazów, ale także pary zdań. Muszą one spełniać te same warunki co słowa, jednak dwa zdania są anagramami nawet wtedy, gdy liczba składających się na nie wyrazów jest inna. Jest to istotne, ponieważ oznacza, że dwa ciągi znaków mogą zawierać różną liczbę spacji, a zarazem być anagramami.

Algorytm do analizowania anagramów: krok 1

Napiszmy algorytm pozwalający sprawdzić, czy dwa wyrazy są anagramami. Wiemy już, że aby tak było, słowa muszą mieć identyczną długość.

Sprawdzimy to od razu – gdyby okazało się, że wyrazy zawierają różną liczbę znaków, na pewno nie są anagramami:

```
1 pierwszyWyraz = "arbuz"
2 drugiWyraz = "kiwi"
```

```
3
4 jeżeli długość(pierwszyWyraz) != długość(drugiWyraz) wykonaj:
5     wypisz "Słowa nie są anagramem"
6     zakończ program
```

Zwróć uwagę na linię numer 4. Została w niej zapisana instrukcja warunkowa, która pozwala wykluczyć słowa o różnej długości.

Na razie nie wiemy jednak jak sprawdzić, czy dwa wyrazy o identycznej długości są anagramami. Wiemy natomiast, że anagramy zawierają tę samą liczbę tych samych liter. Wykorzystajmy tę cechę anagramów.

Algorytm do analizowania anagramów: krok 2

Posłużymy się parą wyrazów „arbuz – burza". Posortujemy alfabetycznie litery w obydwu słowach:

arbuz → abruz

burza → abruz

Wynikiem sortowania jest przekształcenie obydwu wyrazów w dwa ciągi znaków (liter) ułożonych w porządku alfabetycznym. Jeżeli – tak jak w naszym przykładzie – sekwencje te są są identyczne, mamy pewność, że dwa sprawdzane wyrazy są anagramami.

Dopiszmy zatem do algorytmu instrukcje sortowania i porównywania otrzymanych ciągów liter:

```
1  pierwszyWyraz = "arbuz"
2  drugiWyraz = "kiwi"
3
4  jeżeli długość(pierwszyWyraz) != długość(drugiWyraz) to
5      wypisz "Słowa nie są anagramem"
6      zakończ program
7
8  jeżeli sortujAlfabetycznie(pierwszyWyraz) == sortujAlfabetycznie(d
9      wypisz "Słowa są anagramem"
10     zakończ program
```

Ponieważ w czasie bieżącej lekcji nie zajmujemy się algorytmami sortującymi, nie będziemy pisać własnej implementacji mechanizmu sortowania alfabetycznego. W pseudokodzie

użyliśmy funkcji `sortujAlfabetycznie()`, która jako argument przyjmuje badany wyraz, a jako wynik zwraca ciąg liter posortowanych w kolejności alfabetycznej.

Nasz algorytm jest już prawie gotowy. Aby jednak można go było wykorzystać nie tylko w przypadku wyrazów, ale również całych zdań, musimy wprowadzić kilka usprawnień.

Różnica między sprawdzaniem wyrazów a sprawdzaniem zdań polega na tym, że w zdaniach pojawiają się spacje (a także znaki interpunkcyjne, ale w tej implementacji algorytmu je pominiemy). Spacje nie mają wpływu na to, czy dwa zdania są anagramami. Możemy więc usunąć takie znaki. Wykorzystamy funkcję `usunSpacje()`, która zwraca przekazany jej ciąg wyrazów pozbawiony znaków spacji.

Nadamy ponadto inne nazwy używanym dotychczas zmiennym `pierwszyWyraz` i `drugiWyraz`. Zastąpią je zmienne `pierwszyCiagZnakow` i `drugiCiagZnakow` – takie nazwy opisują zarówno wyrazy, jak i całe zdania. Jest to modyfikacja czysto kosmetyczna, ale mimo to warta zastosowania.

Oto zapis gotowego algorytmu:

```
pierwszyCiagZnakow = usunSpacje("arbuz")
drugiCiagZnakow = usunSpacje("kiwi")

jeżeli długość(pierwszyCiagZnakow) != długość(drugiCiagZnakow) wyk
    wypisz "Słowa nie są anagramem"
    zakończ program

jeżeli sortujAlfabetycznie(pierwszyCiagZnakow) == sortujAlfabetycz
    wypisz "Słowa są anagramem"
    zakończ program
```

Sortowanie alfabetyczne i porównywanie otrzymanych ciągów liter nie jest jedyną metodą sprawdzenia, czy dwa zdania lub wyrazy są anagramami. W dalszej części lekcji poznasz implementację innego algorytmu. Zanim do niej przejdziesz, przypomnij sobie jednak, czym jest kod ASCII. Wiedza ta okaże się bowiem niezbędna podczas omawiania kolejnej metody znajdowania anagramów.

Pozostałe materiały z serii


Anagramy w języku C++

Anagramy w języku Java

Anagramy w języku Python

Anagramy – zadania maturalne

Słownik

kod ASCII

(ang. *American Standard Code for Information Interchange*) siedmiobitowy system kodowania znaków używany w systemach komputerowych; liczbom z zakresu 0–127 przyporządkowane są litery alfabetu łacińskiego, cyfry, znaki przestankowe i inne symbole

algorytm sortujący

algorytm wykorzystywany w celu uporządkowania danych w określony sposób

anagram

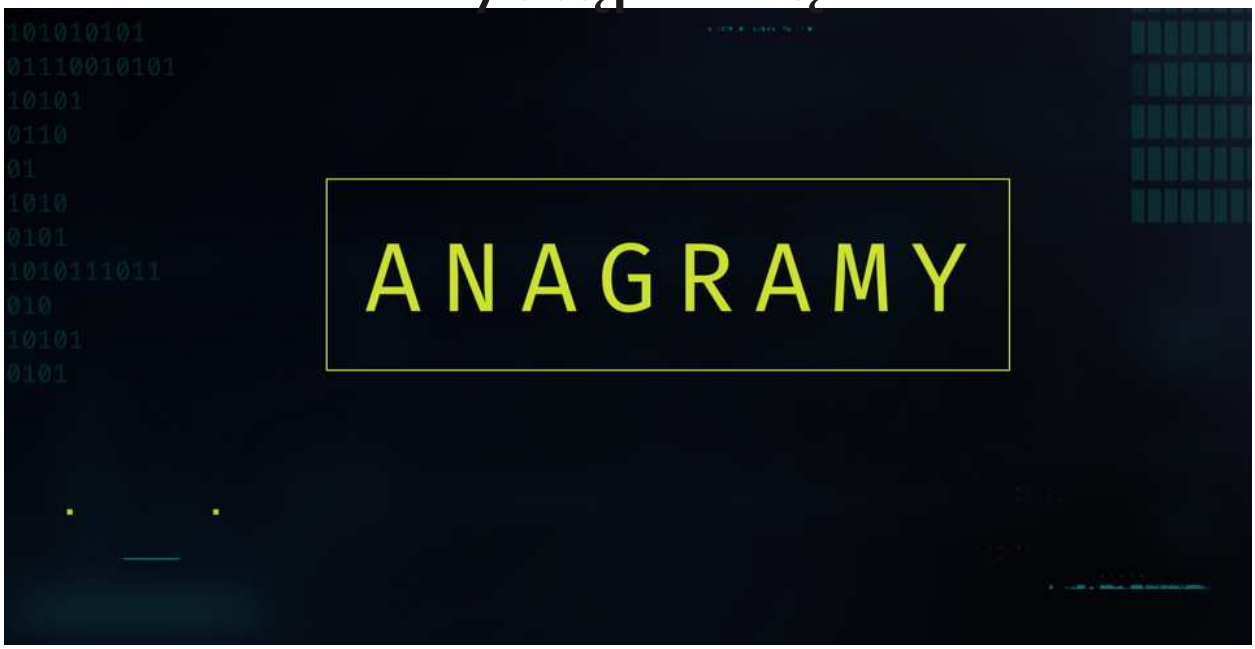
wyraz, wyrażenie bądź całe zdanie powstałe dzięki przestawieniu liter bądź sylab innego wyrazu lub zdania, wykorzystujące wszystkie litery tekstu oryginalnego

Animacja

Polecenie 1

Zapoznaj się z animacją. W dostępnych źródłach poszukaj informacji na temat słynnych anagramów.

Wystąpił błąd



Film dostępny pod adresem /preview/resource/Rhh04CyZMMze5

Film nawiązujący do treści materiału

Polecenie 2

Polecenie 3

Sprawdź się

Pokaż ćwiczenia:

- Ćwiczenie 1
- Ćwiczenie 2
- Ćwiczenie 3
- Ćwiczenie 4
- Ćwiczenie 5
- Ćwiczenie 6
- Ćwiczenie 7
- Ćwiczenie 8

Dla nauczyciela

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

Temat: Anagramy

Grupa docelowa:

Liceum ogólnokształcące i technikum, liceum ogólnokształcące, technikum

Podstawa programowa:

Cele kształcenia – wymagania ogólne

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

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,

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):

- Nauczysz się rozpoznawać anagramy.
- Przeanalizujesz algorytmy wykorzystywane do sprawdzania, czy dwa wyrazy są anagramami.
- Rozwiążesz zadania polegające na sprawdzeniu, czy podane słowa są anagramami.

Strategie nauczania:

- konstruktywizm;
- konektywizm.

Metody i techniki nauczania:

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

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.

Przebieg lekcji

Przed lekcją:

1. **Przygotowanie do zajęć.** Nauczyciel loguje się na platformie i udostępnia e-materiał: „Anagramy". Nauczyciel prosi uczniów o zapoznanie się z treściami w sekcji „Przeczytaj".

Faza wstępna:

1. Nauczyciel inicjuje rozmowę wprowadzającą w temat lekcji. Przedstawia cele zajęć oraz kryteria sukcesu.
2. **Rozpoznanie wiedzy uczniów.** Nauczyciel zadaje uczniom pytania dotyczące ich aktualnego stanu wiedzy w obszarze poruszanego tematu i programowania, np.
   – jak sprawdzić czy dana para wyrazów to anagramy?
   – co to jest algorytm sortujący?
   – co to jest kod ASCII?
   Chętni uczniowie udzielają na nie odpowiedzi.

Faza realizacyjna:

1. Uczniowie analizują przykład z sekcji „Przeczytaj" i powtarzają zaprezentowane rozwiązanie na swoim komputerze.

2. Uczniowie w parach wykonują ćwiczenia nr 1-8 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”.

Praca domowa:

1. Uczniowie opracowują FAQ (minimum 3 pytania i odpowiedzi) do tematu lekcji („Anagramy”).

Wskazówki metodyczne:

- Uczniowie mogą wykorzystać treści w sekcjach: „Przeczytaj”, „Animacja”, „Sprawdź się” jako materiał do lekcji powtórkowej.