Anagramy – zadania maturalne

Źródło: Pixabay, domena publiczna.

Algorytm tekstowy pozwalający sprawdzić, czy podany wyraz jest anagramem innego słowa, został omówiony w e-materiale Anagramy. Wiedzę dotyczącą zagadnień związanych z anagramami wykorzystamy teraz do rozwiązywania zadań typu maturalnego.

Implementację algorytmów sprawdzających, czy dane wyrazy są anagramami, znajdziesz w e-materiałach:

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

> **Twoje cele**
>
> - Prześledzisz konstrukcję przykładowego zadania maturalnego, związanego z algorytmami tekstowymi.
> - Rozwiążesz zadanie maturalne, wykorzystując algorytmy tekstowe.
> - Zapoznasz się z przykładowym zadaniem maturalnym.
> - Utrwalisz swoją wiedzę dotyczącą anagramów.

Przeczytaj

Zadanie 1. Anagram

Anagram to słowo powstałe z innego słowa przez przestawienie liter. Przez słowo rozumiemy w tym zadaniu dowolny ciąg liter alfabetu łacińskiego.

Przykłady anagramów:

- dla słowa: *barok* – *korba*, *robak*, *arobk*, *rokab*, *orkab*...
- dla słowa: *ranty* – *tyran*, *narty*, *utyra*, *natyr*, *ytnar*...

W pliku tekstowym `anagram.txt` znajduje się 200 wierszy zawierających po 5 słów w każdym wierszu. Słowa oddzielone są znakiem odstępu. Długość każdego ze słów wynosi od 1 do 20 znaków.

Plik `anagramy.txt`.
Plik o rozmiarze 6.37 KB w języku polskim

Przykład:

```
abcd cdba dbac cbad dcba
barbakan xle ala foto otof
smok ayszkm lampa ayszkm bakara
skok arabanta oko agnieba dyskietka
```

> Zadanie zostało opracowane przez Centralną Komisję Egzaminacyjną i pojawiło się na egzaminie maturalnym z informatyki w maju 2010 roku (egzamin w tzw. starej formule, poziom rozszerzony, cz. 2). Cały arkusz można znaleźć na stronie internetowej CKE.

Zadanie 1.1

Napisz program w wybranym przez siebie języku programowania, za pomocą którego wykonasz poniższe polecenie:

a) Wyszukaj w pliku `anagram.txt` te wiersze, w których wszystkie słowa znajdujące się w danym wierszu mają taką samą liczbę znaków. Zapisz te wiersze w pliku `odp_4a.txt`.

Do oceny oddajesz:

- plik `odp_4a.txt` z odpowiedzią,
- plik(i) z komputerową realizacją zadania (kodem programu).

Rozwiązanie

> Rozwiązanie zadania przedstawimy w postaci pseudokodu, ponieważ na egzaminie maturalnym można korzystać z wybranego języka programowania: C++, Java lub Python.

Rozwiązanie rozpoczniemy od wczytania danych z pliku do dwuwymiarowej tablicy `anagramy` przechowującej łańcuchy znaków składającej się z 200 wierszy oraz 5 kolumn.

Zaimplementujemy pętlę `dla` wykonującą się 200 razy (tyle, ile jest wierszy w pliku). W pętli zmiennej logicznej `czyRownaDlugosc` nadajemy wartość `Prawda`, zakładamy zatem, że wszystkie słowa w danym wierszu mają taką samą długość. W każdym wierszu mamy po pięć słów – w pętli wewnętrznej porównamy długość pierwszego słowa z długościami pozostałych. Możemy przypisać długość pierwszego słowa do zmiennej `n`. Dzięki temu funkcja `długość` nie będzie wywoływana niepotrzebnie kilka razy. Skorzystamy z tej funkcji, ponieważ jej implementacja jest dostępna w każdym z języków programowania używanych na egzaminie maturalnym.

```
anagramy[0..199] ← wczytaj dane z pliku "anagram.txt"

dla i = 0, 1, ..., 199 wykonuj:
    czyRownaDlugosc ← PRAWDA
    n ← dlugosc(anagramy[i][0])
    dla j = 1, 2, ..., 4 wykonuj:
```

Jeżeli którekolwiek ze słów będzie miało inną długość niż pierwsze, ustawimy wartość zmiennej `czyRownaDlugosc` na `FAŁSZ` i przerwiemy pętlę:

```
anagramy[0..199] ← wczytaj dane z pliku "anagram.txt"

dla i = 0, 1, ..., 199 wykonuj:
    czyRownaDlugosc ← PRAWDA
    n ← dlugosc(anagramy[i][0])
    dla j = 1, 2, ..., 4 wykonuj:
        jeżeli n != dlugosc(anagramy[i][j]):
            czyRownaDlugosc ← FAŁSZ
            przerwij pętlę
```

Po zakończeniu wewnętrznej pętli sprawdzamy wartość zmiennej `czyRownaDlugosc`. Jeśli jest równa `PRAWDA`, to wszystkie słowa z danego wiersza mają tę samą długość i możemy go zapisać do pliku wynikowego. Po przetworzeniu wszystkich danych z pliku zamykamy pliki `anagram.txt` i `odp_4a.txt`:

```
anagramy[0..199] ← wczytaj dane z pliku "anagram.txt"

dla i = 0, 1, ..., 199 wykonuj:
    czyRownaDlugosc ← PRAWDA
    n ← dlugosc(anagramy[i][0])
    dla j = 1, 2, ..., 4 wykonuj:
        jeżeli n != dlugosc(anagramy[i][j]):
            czyRownaDlugosc ← FAŁSZ
            przerwij pętlę
    jeżeli czyRownaDlugosc = PRAWDA:
        dopisz anagramy[i] do pliku "odp_4a.txt"

zamknij plik "anagram.txt"
zamknij plik "odp_4a.txt"
```

Schemat oceniania

- **4 pkt** – za poprawną zawartość pliku `odp_4a.txt` uwzględniającą wszystkie wiersze,
- **2 pkt** – za zawartość pliku `odp_4a.txt` z jednym błędem (błędny jeden wiersz lub brak jednego wiersza),
- **0 pkt** – za zawartość pliku `odp_4a.txt` z dwoma błędami lub więcej albo za brak odpowiedzi.

Odpowiedź do zadania

Plik `odp_4a.txt`
Plik o rozmiarze 774.00 B w języku polskim

Słownik

> **anagram**
>
> słowo utworzone w wyniku przestawienia kolejności liter innego słowa, zawierające wszystkie jego litery

Prezentacja multimedialna

> **Ważne!**
>
> Zadanie jest kontynuacją zadania z sekcji „Przeczytaj”.

Zadanie 1.2

Polecenie 1

Napisz program w wybranym przez siebie języku programowania, za pomocą którego wykonasz poniższe polecenie:

b) Wyszukaj w pliku `anagram.txt` wszystkie wiersze tekstu, w których wszystkie słowa są anagramami pierwszego słowa w danym wierszu. Zapisz te wiersze w pliku `odp_4b.txt`.

```
1
```

Do oceny oddajesz:

- plik `odp_4b.txt` z odpowiedzią,
- plik(i) z komputerową realizacją zadania (kodem programu).

Rozwiązanie

Polecenie 2

Prześledź kolejne kroki prezentacji i porównaj je ze swoim rozwiązaniem.

Rozwiązanie zadania przedstawimy w postaci pseudokodu, ponieważ na egzaminie maturalnym można korzystać z wybranego języka programowania: C++, Java lub Python.

Ważne!

Możemy zauważyć, że w pliku z danymi wszystkie słowa zapisane są małymi literami. W związku z tym w programie nie musimy uwzględniać wielkości liter

1

Inspiracja anagramami widoczna jest w wielu dziełach kultury. Ilustracja przedstawia fragment portalu kościoła w Meppen. Inskrypcja zawiera dwie łacińskie sekwencje będące anagramami.
Źródło: de.wikipedia.org, domena publiczna.

Materiał audio dostępny pod adresem:

https://zpe.gov.pl/b/PD7egCT2D

Aby rozwiązać zadanie 1.2, wykorzystamy część kodu z rozwiązania dla podpunktu 1.1. Wiemy, że anagramy muszą się składać z jednakowej liczby tych samych znaków. Słowa będące anagramami muszą mieć jednakową długość:

```
1 anagramy[0..199] ← wczytaj
  dane z pliku "anagram.txt"
2
3 dla i = 0, 1, ..., 199
  wykonuj:
4     czyAnagramy ← PRAWDA
```

```
    n ← dlugosc(anagramy[i][0])
    dla j = 1, 2, ..., 4 wykonuj:
        jeżeli n != dlugosc(anagramy[i][j]):
            czyAnagramy ← FAŁSZ
            przerwij pętlę
```

2

Materiał audio dostępny pod adresem:

https://zpe.gov.pl/b/PD7egCT2D

Jeśli wykluczyliśmy przypadek różnych długości słów, możemy przejść dalej. Sprawdzenie, czy dwa słowa są anagramami, możemy wykonać przez posortowanie ich znaków w kolejności alfabetycznej i stwierdzenie, czy obydwa powstałe łańcuchy są sobie równe. Skorzystamy z funkcji `sortuj()`, zwracającej posortowany łańcuch znaków, której implementację przedstawimy po głównej części zadania. Analogicznie jak przy sprawdzaniu długości, przerwiemy pętlę i zmienimy wartość zmiennej `czyAnagramy`, jeśli posortowane słowa będą różne:

```
anagramy[0..199] ← wczytaj dane z pliku "anagram.txt"

dla i = 0, 1, ..., 199 wykonuj:
    czyAnagramy ← PRAWDA
    n ← dlugosc(anagramy[i][0])
    dla j = 1, 2, ..., 4 wykonuj:
        jeżeli n != dlugosc(anagramy[i][j]):
```

```
            czyAnagramy ← FAŁSZ
            przerwij pętlę
    jeżeli czyAnagramy = PRAWDA:
        pierwsze ← sortuj(anagramy[i][0])
        dla j = 1, 2, ..., 4 wykonuj:
            jeżeli pierwsze != sortuj(anagramy[i][j]):

czyAnagramy ← FAŁSZ
                przerwij pętlę
```

3

Materiał audio dostępny pod adresem:

https://zpe.gov.pl/b/PD7egCT2D

Jeśli pętla wykona się bez przerwania, to znaczy, że wszystkie słowa są anagramami pierwszego słowa w danym wierszu, a zmienna `czyAnagramy` dalej ma wartość prawdy. Zapisujemy wtedy cały wiersz do pliku wynikowego i powtarzamy procedurę dla pozostałych wierszy tablicy `anagramy`, wykorzystując zewnętrzną pętlę `for` (`dla`):

```
anagramy[0..199] ← wczytaj dane z pliku "anagram.txt"

dla i = 0, 1, ..., 199 wykonuj:
    czyAnagramy ← PRAWDA
    n ← dlugosc(anagramy[i][0])
    dla j = 1, 2, ..., 4 wykonuj:
```

```
        jeżeli n != dlugosc(anagramy[i][j]):
            czyAnagramy ← FAŁSZ
            przerwij pętlę
    jeżeli czyAnagramy = PRAWDA:
        pierwsze ← sortuj(anagramy[i][0])
        dla j = 1, 2, ..., 4 wykonuj:
            jeżeli pierwsze != sortuj(anagramy[i][j]):

czyAnagramy ← FAŁSZ
                przerwij pętlę
    jeżeli czyAnagramy = PRAWDA:
        dopisz anagramy[i] na koniec pliku "odp_4b.txt"
```

4

Materiał audio dostępny pod adresem:

https://zpe.gov.pl/b/PD7egCT2D

Do posortowania liter w słowie wykorzystamy sortowanie metodą wyboru. Algorytm sortowania zapiszemy za pomocą funkcji `sortuj` z jednym parametrem `wyraz`. Zadaniem funkcji będzie posortowanie liter w łańcuchu znaków przekazanych w argumencie funkcji. Przy implementacji algorytmu wykorzystamy fakt, że łańcuch znaków możemy potraktować jako jednowymiarową tablicę znaków.

Zapiszmy zatem pętlę przechodzącą po wszystkich elementach łańcucha znaków. Po raz

kolejny skorzystamy z funkcji `dlugosc`. Funkcja jest dostępna w każdym z języków programowania dopuszczonych na egzaminie maturalnym.

```
1 funkcja sortuj(wyraz)
2
3     dla i = 0, 1, ..., dlugosc(wyraz) - 1 wykonuj:
4         najmniejszy ← i
```

5

Materiał audio dostępny pod adresem:

https://zpe.gov.pl/b/PD7egCT2D

W każdej iteracji będziemy przeszukiwać nieuporządkowaną część tablicy w celu znalezienia najmniejszego elementu i zamienienia go z elementem o indeksie `i`. Zadeklarujmy kolejną pętlę `dla`:

```
1 funkcja sortuj(wyraz)
2
3     dla i = 0, 1, ..., dlugosc(wyraz) - 1 wykonuj:
4         najmniejszy ← i
5         dla j = i + 1, i + 2, ..., dlugosc(wyraz) - 1 wykonuj:
```

6

Materiał audio dostępny pod adresem:

https://zpe.gov.pl/b/PD7egCT2D

W celu porównywania ze sobą znaków skorzystamy z funkcji `ascii()`, zwracającej kod ASCII danego znaku (kolejność znaków w kodzie ASCII odpowiada tej alfabetycznej). Na egzaminie maturalnym implementacja funkcji jest dostępna w każdym języku programowania. Jeżeli znak pod indeksem `j` ma mniejszy kod ASCII, oznacza to, że występuje wcześniej w alfabecie i należy zaktualizować zmienną `najmniejszy`:

```
funkcja sortuj(wyraz)

    dla i = 0, 1, ..., dlugosc(wyraz) - 1 wykonuj:
        najmniejszy ← i
        dla j = i + 1, i + 2, ..., dlugosc(wyraz) - 1 wykonuj:
            jeżeli ascii(wyraz[j]) < ascii(wyraz[najmniejszy]):

najmniejszy ← j
```

7

Materiał audio dostępny pod adresem:

https://zpe.gov.pl/b/PD7egCT2D

Po zakończeniu wewnętrznej pętli zamieniamy elementy pod indeksami `i` oraz `najmniejszy` i kontynuujemy sortowanie. Na koniec zwracamy uporządkowany alfabetycznie łańcuch:

```
funkcja sortuj(wyraz)

    dla i = 0, 1, ..., dlugosc(wyraz) - 1 wykonuj:
        najmniejszy ← i
```

```
5         dla j = i + 1, i +
2, ..., dlugosc(wyraz) - 1
wykonuj:
6             jeżeli
ascii(wyraz[j]) <
ascii(wyraz[najmniejszy]):
7
najmniejszy ← j
8         zamień wyraz[i] z
wyraz[najmniejszy]
9
10    zwróć wyraz
```

8

Konstruowanie z liter słowa kolejnych wyrazów jest podstawą różnego typu zabaw słownych i gier, jedną z nich jest Boggle .
Źródło: Rich Brooks, en.wikipedia.org, CC BY-SA 2.0.

Materiał audio dostępny pod adresem:

https://zpe.gov.pl/b/PD7egCT2D

Cały kod wygląda następująco:

```
1 funkcja sortuj(wyraz)
2
3     dla i = 0, 1, ...,
dlugosc(wyraz) - 1
wykonuj:
4         najmniejszy ← i
5         dla j = i + 1, i +
2, ..., dlugosc(wyraz) - 1
wykonuj:
```

```
            jeżeli
ascii(wyraz[j]) <
ascii(wyraz[najmniejszy]):

najmniejszy ← j
        zamień wyraz[i] z
wyraz[najmniejszy]

    zwróć wyraz

anagramy[0..199] ← wczytaj
dane z pliku "anagram.txt"

dla i = 0, 1, ..., 199
wykonuj:
    czyAnagramy ← PRAWDA
    n ←
dlugosc(anagramy[i][0])
    dla j = 1, 2, ..., 4
wykonuj:
        jeżeli n !=
dlugosc(anagramy[i][j]):
            czyAnagramy ←
FAŁSZ
            przerwij pętlę
    jeżeli czyAnagramy =
PRAWDA:
        pierwsze ←
sortuj(anagramy[i][0])
        dla j = 1, 2, ...,
4 wykonuj:
            jeżeli
pierwsze !=
sortuj(anagramy[i][j]):

czyAnagramy ← FAŁSZ
                przerwij
pętlę
    jeżeli czyAnagramy =
PRAWDA:
        dopisz anagramy[i]
na koniec pliku
"odp_4b.txt"
```

Schemat oceniania

- **6 pkt** – za poprawną zawartość pliku `odp_4b.txt` uwzględniającą wszystkie wiersze z anagramami,
- **2 pkt** – za zawartość pliku `odp_4b.txt` z jednym błędem (błędny jeden wiersz lub brak jednego wiersza),
- **0 pkt** – za zawartość pliku `odp_4b.txt` z dwoma błędami lub więcej albo za brak odpowiedzi.

Odpowiedź do zadania

Plik `odp_4b.txt`.
Plik o rozmiarze 229.00 B w języku polskim

> **Polecenie 3**
>
> Dodaj do opracowanego przez siebie kodu programu komentarze, by nawet osoba, która nie potrafi programować, mogła go zrozumieć.

Sprawdź się

Pokaż ćwiczenia:

Zadanie 2

W Bajtolandii ostatnim krzykiem mody są tkaniny we wzór anagramów. Najpopularniejszy wzór to ten, który składa się z trzech czteroznakowych anagramów.

Plik `tkaniny.txt` zawiera 100 dostępnych wzorów. Każdy ze wzorów składa się z trzech słów. Największy sklep z tkaninami w Bajtolandii chciałby złożyć zamówienie wyłącznie na te wzory, które są aktualnie w modzie, czyli anagramy składające się z czterech znaków.

W pliku tekstowym `tkaniny.txt` znajduje się 100 wierszy zawierających po 3 słowa w każdym wierszu. Słowa oddzielone są znakiem odstępu. Długość każdego ze słów wynosi od 2 do 14 znaków.

Plik `tkanininy.txt`.
Plik o rozmiarze 2.56 KB w języku polskim

Zadanie 2.1

Napisz program, który wypisze wiersze z pliku `tkaniny.txt`, w których znajdują się wyrazy będące anagramami składającymi się z czterech znaków. Odpowiedzi zapisz w pliku `modne_tkaniny.txt`.

Do oceny oddajesz:

- plik `modne_tkaniny.txt` z odpowiedzią do zadania (wiersze, w których znajdują się wyrazy będące anagramami składającymi się z czterech znaków),
- plik(i) z programem (komputerową realizacją zadania).

Ważne!

Informację na temat wbudowanych funkcji sortujących w językach programowania znajdziesz w e-materiale Algorytmy sortowania w bibliotekach standardowych.

Przykład 1

Poprawny wynik dla danych podanych w testerkach:

```
TSST TSST TSST
ABBA BAAB BBAA
SRSR SRSR SRSR
```

```
4 GPWD WPDG WPDG
```

Ćwiczenie 1

JĘZYK C++

Twoje zadania

1. Program wypisuje wiersze, które zawierają wyłącznie czteroznakowe anagramy.

```cpp
#include <iostream>
#include <string>

using namespace std;

string sortuj(string wyraz) {
    int n = wyraz.size();

    for (int i = 0; i < n; i++)
        for (int j = 1; j < n - i; j++)
            if (wyraz[j - 1] > wyraz[j])
                swap(wyraz[j - 1], wyraz[j]);

    return wyraz;
```

```cpp

```

Ćwiczenie 2

JĘZYK JAVA

Twoje zadania

1. Program wypisuje wiersze, które zawierają wyłącznie czteroznakowe anagramy.

```java
public class Main {
    public static String sortuj(String wyraz) {
        int n = wyraz.length();

        for (int i = 0; i < n; i++)
            for (int j = 1; j < n - i; j++)
                if (wyraz.charAt(j - 1) > wyraz.charAt(j)) {
                    char[] c = wyraz.toCharArray();
                    char temp = c[j - 1];
                    c[j - 1] = c[j];
                    c[j] = temp;
                    wyraz = new String(c);
                }

```

```
1
```

Ćwiczenie 3

JĘZYK PYTHON

Twoje zadania

1. Program wypisuje wiersze, które zawierają wyłącznie czteroznakowe anagramy.

```python
def sortuj(wyraz):
    tab = []
    n = len(wyraz)

    for i in range(0, n):
        tab.append(wyraz[i])

    for i in range(0, n):
        for j in range(1, n - i):
            if tab[j - 1] < tab[j]:
                pom = tab[j - 1]
                tab[j - 1] = tab[j]
                tab[j] = pom

```

```
1
```

Odpowiedź dla danych z pliku.

Plik `modne_tkaniny.txt`.
Plik o rozmiarze 238.00 B w języku polskim

Zadanie 2.2

Moda w Bajtolandii zmienia się szybko – popularność straciły tkaniny ze słowami powtarzającymi się we wzorach.

Sklep chciałby złożyć nowe zamówienie, tym razem wyłącznie na te tkaniny, których wzory to czteroliterowe różne słowa, będące anagramami.

Napisz program podający liczbę wierszy w pliku `tkaniny.txt`, w których wszystkie słowa to czteroliterowe anagramy, które się nie powtarzają.

Do dyspozycji masz funkcje sortujące `sort()` w języku C++, `Arrays.sort()` w języku Java oraz `sorted()` w języku Python.

Wynik programu zapisz do pliku `najmodniejsze_tkaniny.txt`.

Do oceny oddajesz:

- plik `najmodniejsze_tkaniny.txt` z odpowiedzią do zadania (liczba naturalna oznaczającą liczbę wierszy, w których znajdują się czteroliterowe, niepowtarzające się anagramy),
- plik(i) z programem (komputerową realizacją zadania).

Ćwiczenie 4

JĘZYK C++

Twoje zadania

1. Program podaje liczbę wierszy, w których wszystkie słowa to czteroliterowe anagramy i żadne z nich się nie powtarza.

```cpp
#include <iostream>
#include <string>
#include <algorithm>

using namespace std;

string sortuj(string wyraz) {
    string wyraz_kopia = wyraz;
    sort(wyraz_kopia.begin(), wyraz_kopia.end());
    return wyraz_kopia;
}

int main() {
    int m = 5;
```

```
1
```

Ćwiczenie 5

JĘZYK JAVA

Twoje zadania

1. Program podaje liczbę wierszy, w których wszystkie słowa to czteroliterowe anagramy i żadne z nich się nie powtarza.

```java
public class Main {
    public static String sortuj(String wyraz) {
        char wyraz_kopia[] = wyraz.toCharArray();
        Arrays.sort(wyraz_kopia);
        return new String(wyraz_kopia);
    }

    public static void main(String[] args) {
        int m = 5;
        String [][] slowa = {
            {"TSST", "TSST", "TSST"},
            {"GDGGDEGDGG","AAEDEGDCDC","CGCGBEFCAF"},
            {"ABBA", "BAAB", "BBAA"},
            {"SRSR", "SRSR", "SRSR"},
```

```
1
```

Ćwiczenie 6

JĘZYK PYTHON

Twoje zadania

1. Program podaje liczbę wierszy, w których wszystkie słowa to czteroliterowe anagramy i żadne z nich się nie powtarza.

```python
def sortuj(slowo):
    slowo_kopia = slowo
    slowo_kopia = sorted(slowo_kopia)
    return "".join(slowo_kopia)

lista_slow = [
    ["TSST", "TSST", "TSST"],
    ["GDGGDEGDGG","AAEDEGDCDC","CGCGBEFCAF"],
    ["ABBA", "BAAB", "BBAA"],
    ["SRSR", "SRSR", "SRSR"],
    ["GPWD", "WPDG", "WPDG"]
]

# Tutaj dodaj kod. Do wypisywania użyj instrukcji print().
```

```python

```

Odpowiedź dla danych z pliku:

Plik `najmodniejsze_tkaniny.txt`.
Plik o rozmiarze 2.00 B w języku polskim

Dla nauczyciela

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

Temat: Anagramy – zadania maturalne

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.

Treści nauczania – wymagania szczegółowe

I. Rozumienie, analizowanie i rozwiązywanie problemów.

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

1) w zależności od problemu rozwiązuje go, stosując metodę wstępującą lub zstępującą;

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

3) objaśnia dobrany algorytm, uzasadnia poprawność rozwiązania na wybranych przykładach danych i ocenia jego efektywność;

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

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

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):

- Prześledzisz konstrukcję przykładowego zadania maturalnego, związanego z algorytmami tekstowymi.
- Rozwiążesz zadanie maturalne, wykorzystując algorytmy tekstowe.
- Zapoznasz się z przykładowym zadaniem maturalnym.
- Utrwalisz swoją wiedzę dotyczącą anagramów.

Strategie nauczania:

- konstruktywizm;
- konektywizm.

Metody i techniki nauczania:

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

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 C++, w tym kompilator GCC/G++ 4.5 (lub nowszej wersji) i Code::Blocks 16.01 (lub nowszej wersji), Orwell Dev-C++ 5.11 (lub nowszej wersji) lub Microsoft Visual Studio;
- oprogramowanie dla języka Java SE 8 (lub nowszej wersji), w tym Eclipse 4.4 (lub nowszej wersji);
- oprogramowanie dla języka Python 3 (lub nowszej wersji), w tym PyCharm lub IDLE.

Przebieg lekcji

Przed lekcją:

1. Nauczyciel prosi uczniów o przypomnienie sobie najważniejszych informacji dotyczących anagramów.
2. **Przygotowanie do zajęć.** Nauczyciel loguje się na platformie i udostępnia e-materiał: „Anagramy – zadania maturalne". Nauczyciel prosi uczniów o zapoznanie się z treściami w sekcji „Przeczytaj".

Faza wstępna:

1. **Rozpoznanie wiedzy uczniów**. Chętna lub wybrana osoba referuje najważniejsze informacje dotyczące anagramów. Nauczyciel w razie potrzeby uzupełnia wypowiedź.
2. Wyświetlenie przez nauczyciela tematu i celów lekcji. Określenie wiążących dla uczniów kryteriów sukcesu.

Faza realizacyjna:

1. **Praca z tekstem.** Uczniowie w parach analizują zaprezentowane w sekcji „Przeczytaj" Zadanie 1.1 oraz jego rozwiązanie. Nauczyciel odpowiada na ewentualne pytania.
2. **Praca z multimedium**. Uczniowie zapoznają się z prezentacją. W parach przygotowują program, wykorzystując wybrany język programowania. Chętna lub wybrana para prezentuje swoje rozwiązanie. Nauczyciel w razie potrzeby uzupełnia je.
3. **Ćwiczenie umiejętności.** Uczniowie wykonują indywidualnie zadanie 2.1. Następnie w parach prezentują sobie swoje rozwiązania, tłumacząc je krok po kroku.

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. Wybrany uczeń podsumowuje zajęcia, zwracając uwagę na nabyte umiejętności, omawia ewentualne problemy podczas rozwiązania ćwiczeń z programowania.

Praca domowa:

1. Uczniowie wykonują zadanie 2.2 z sekcji „Sprawdź się".

Materiały pomocnicze:

- Oficjalna dokumentacja techniczna dla języka C++.
- Oficjalna dokumentacja techniczna dla kompilatora GCC/G++ 4.5 (lub nowszej wersji).
- Oficjalna dokumentacja techniczna dla oprogramowania Code::Blocks 16.01 (lub nowszej wersji), Orwell Dev-C++ 5.11 (lub nowszej wersji) lub Microsoft Visual Studio.
- Oficjalna dokumentacja techniczna dla języka Java SE 8 (lub nowszej wersji).

- Oficjalna dokumentacja techniczna dla oprogramowania Eclipse 4.4 (lub nowszej wersji).
- Oficjalna dokumentacja techniczna dla języka Python 3 (lub nowszej wersji).
- Oficjalna dokumentacja techniczna dla oprogramowania PyCharm lub IDLE.

Wskazówki metodyczne:

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