# Algorytm Knutha-Morrisa-Pratta – zadania maturalne

# Algorytm Knutha-Morrisa-Pratta – zadania maturalne

Źródło: Wallace Chuck, domena publiczna.

W e-materiale Algorytm Knutha-Morrisa-Pratta zapoznaliśmy się wstępnie z tym algorytmem tekstowym – wiemy, czym się charakteryzuje, i w jakich sytuacjach warto go zastosować. Teraz wykorzystamy tę wiedzę podczas rozwiązywania zadań, z jakimi możemy się zetknąć na egzaminie maturalnym.

Implementację algorytmu KMP w wybranych językach programowania znajdziesz w e-materiałach:

- Algorytm Knutha-Morrisa-Pratta w języku C++,
- Algorytm Knutha-Morrisa-Pratta w języku Java,
- Algorytm Knutha-Morrisa-Pratta w języku Python.

**Twoje cele**

- Zastosujesz algorytm KMP w celu wyszukania wzorca w tekście.
- Przeanalizujesz stopień trudności zadań maturalnych wymagających zastosowania algorytmów tekstowych.
- Rozwiążesz problemy wymagające użycia algorytmów tekstowych.
- Prześledzisz przykładowe zadania maturalne opracowane przez CKE.

# Przeczytaj

## Zadanie 1

W pliku `hasla.txt` podanych jest `200` haseł użytkowników pewnego systemu. Każdy użytkownik ma jedno hasło (zapisane w osobnym wierszu), które zawiera od 1 do 20 znaków alfanumerycznych, tzn. cyfr od `0` do `9` lub liter alfabetu łacińskiego (małych lub wielkich). Polityka bezpieczeństwa systemu wymaga, aby hasła były odpowiednio skomplikowane i nie powtarzały się.

`Plik hasla.txt`
Plik o rozmiarze 2.09 KB w języku polskim

> W tym zadaniu zamiast pliku posłużymy się tablicą łańcuchów znaków zawierającą 200 haseł zgodych z warunkami zadania.

### Zadanie 1.1

Podaj liczbę haseł złożonych jedynie ze znaków numerycznych, to znaczy cyfr od 0 do 9.

> Zadanie zostało opracowane przez Centralną Komisję Egzaminacyjną i znajduje się w Maturalnym zbiorze zadań z informatyki jako zadanie nr 74.1. Ze zbiorem można zapoznać się na oficjalnej stronie cke.gov.pl.
> Ponieważ na egzaminie maturalnego uczniowie mogą wybrać język programowania, w przedstawionym tu rozwiązaniu posłużymy się pseudokodem. Twoim zadaniem będzie przełożenie tak zapisanych instrukcji na dowolny język.

Aby rozwiązać zadanie, musimy znaleźć sposób na policzenie haseł spełniających odpowiednie kryteria. Pisząc kod, założymy, że mamy do dyspozycji dwie funkcje:

1. `pobierz_kod_ASCII()` – zwracającą kod ASCII znaku podanego jako argument;
2. `długość()` – zwracającą liczbę znaków składających się na łańcuch podany jako parametr.

Zaczniemy od zadeklarowania tablicy łańcuchów znaków:

```
1 hasła[] = tablica 200 łańcuchów znaków zawierająca hasła użytkown
```

Naszym zadaniem jest policzenie haseł numerycznych, dlatego potrzebujemy zmiennej, której wartość będziemy zwiększać za każdym razem, gdy natrafimy na ciąg złożony wyłącznie z cyfr. Początkowo wartość tej zmiennej wynosi `0`:

```
```

```
hasła[] = tablica 200 łańcuchów znaków zawierająca hasła użytkown
licznik = 0
```

Następnie utworzymy pętlę, w której zmienna licznikowa będzie przyjmowała wartości od 0 do 199. Wskaże ona kolejno wszystkie elementy tablicy:

```
hasła[] = tablica 200 łańcuchów znaków zawierająca hasła użytkown
licznik = 0

dla i = 0, 1, ..., 199 wykonuj
```

Pętla `dla` posłuży do zbadania każdego łańcucha znaków, zapisanego w tablicy `hasła`.

Na początku pętli definiujemy zmienną logiczną `czy_numeryczne` i nadajemy jej wartość `true`. Użyjemy jej później, aby zadecydować, czy zmienna `licznik` powinna być inkrementowana:

```
hasła[] = tablica 200 łańcuchów znaków zawierająca hasła użytkown
licznik = 0

dla i = 0, 1, ..., 199 wykonuj
    czy_numeryczne = true
```

Następnie tworzymy kolejną pętlę `dla`. Jej zmienna licznikowa (`j`) wskaże – po kolei – znaki składające się na wszystkie hasła. Dzięki funkcji `pobierz_kod_ASCII()` będziemy badać wartość liczbową poszczególnych znaków łańcucha `hasła[i]`. Wynik przypiszemy zmiennej `n`.

Korzystając z instrukcji warunkowej `jeżeli`, sprawdzimy, czy `n` jest mniejsze od 48 lub większe od 57. Jeśli warunek taki jest spełniony, znak nie należy do przedziału opisującego cyfry w tablicy ASCII (innymi słowy, nie jest znakiem numerycznym).
Zmienna `czy_numeryczne` przybierze wówczas wartość `false`, a pętla – służąca do kontrolowania bieżącego hasła – zostanie przerwana:

```
hasła[] = tablica 200 łańcuchów znaków zawierająca hasła użytkown
licznik = 0

dla i = 0, 1, ..., 199 wykonuj
    czy_numeryczne = true
```

```
    dla j = 0, 1, ..., długość(hasła[i] - 1) wykonuj
        n = pobierz_kod_ASCII(hasła[i][j])

        jeżeli n < 48 lub n > 57 to
            czy_numeryczne = false
            break
```

W językach programowania, z których można korzystać na egzaminie maturalnym, istnieją gotowe funkcje sprawdzające, czy dany znak jest cyfrą.

W języku Python jest to funkcja `isdigit()`. Zwraca ona wartość `True`, jeżeli znak jest liczbą, lub `False`, jeżeli nie jest. Oto przykład zastosowania:

```python
znak = "6"

if znak.isdigit():
    print("znak jest liczbą")
else:
    print("znak nie jest liczbą")
```

W języku C++ jest to funkcja `isdigit()`. Zwraca ona wartość niezerową, jeżeli znak jest cyfrą; w przeciwnym wypadku zwraca `0`. Przykład zastosowania:

```cpp
#include <iostream>
using namespace std;

int main() {
    char znak = 'a';

    if (isdigit(znak)) {
        cout << "znak jest liczbą" << endl;
    } else {
        cout << "znak nie jest liczbą" << endl;
    }
}
```

W języku Java jest to funkcja `isDigit()` z klasy `Character`. Zwraca ona wartość `true`, jeżeli znak jest cyfrą; w przeciwnym wypadku zwraca `false`. Przykład zastosowania:

```java
public class Test {
    public static void main(String args[]) {
        char znak = '5';

        if (Character.isDigit(znak)) {
            System.out.println("znak jest liczbą");
        } else {
            System.out.println("znak nie jest liczbą");
        }
    }
}
```

Pod koniec zewnętrznej pętli `dla` dodamy kolejną instrukcję warunkową `jeżeli`, która – zależnie od wartości logicznej `czy_numeryczne` – zwiększy zmienną `licznik` lub tego nie zrobi.

Po zakończeniu pętli podamy liczbę haseł, które zawierają wyłącznie znaki numeryczne:

```
hasła[] = tablica 200 łańcuchów znaków zawierająca hasła użytkown
licznik = 0

dla i = 0, 1, ..., 199 wykonuj
    czy_numeryczne = true

    dla j = 0, 1, ..., długość(hasła[i] - 1) wykonuj
        n = pobierz_kod_ASCII(hasła[i][j])

        jeżeli n < 48 lub n > 57 to
            czy_numeryczne = false
            break

    jeżeli czy_numeryczne to
        licznik = licznik + 1

wypisz licznik
```

W ten sposób rozwiązaliśmy zadanie. Zgodnie ze schematem oceniania punkty otrzymamy tylko wtedy, gdy podany wynik będzie prawidłowy. Zwróć uwagę, że ewentualne błędy mogą pojawić się wskutek pominięcia liczby 48 lub 57 podczas zapisywania warunku instrukcji `jeżeli` w wewnętrznej pętli. Warto poświęcić więc trochę czasu na

sprawdzenie przedziałów w warunkach instrukcji warunkowej `jeżeli`. Jeśli się pomylimy i np. zamiast otwartego przedziału wpiszemy zamknięty, stracimy punkt.

Oto plik z wynikiem do zadania 1.1.

Plik wyniki_hasla.txt
Plik o rozmiarze 31.00 B w języku polskim

## Słownik

> **kod ASCII**
>
> kod unikatowy dla każdego znaku; zestawienie znaków i ich kodów ASCII nazywane jest tablicą ASCII
> **znaki alfanumeryczne**
>
> znaki alfabetu łacińskiego oraz cyfry

# Prezentacja multimedialna

**Polecenie 1**

Przeanalizuj prezentację, a następnie – w wybranym języku programowania – napisz program rozwiązujący zaprezentowany problem.

Plik `hasla.txt` użyty w prezentacji
Plik o rozmiarze 2.09 KB w języku polskim

Plik `wyniki_hasla.txt` z wynikami zadania z prezentacji dla danych z pliku hasla.txt
Plik o rozmiarze 106.00 B w języku polskim

1

Donald E. Knuth – jeden z twórców algorytmu
Źródło: Alex Handy, en.wikipedia.org, dostęp: 21.12.2022, CC BY-SA 2.0.

Materiał audio dostępny pod adresem:

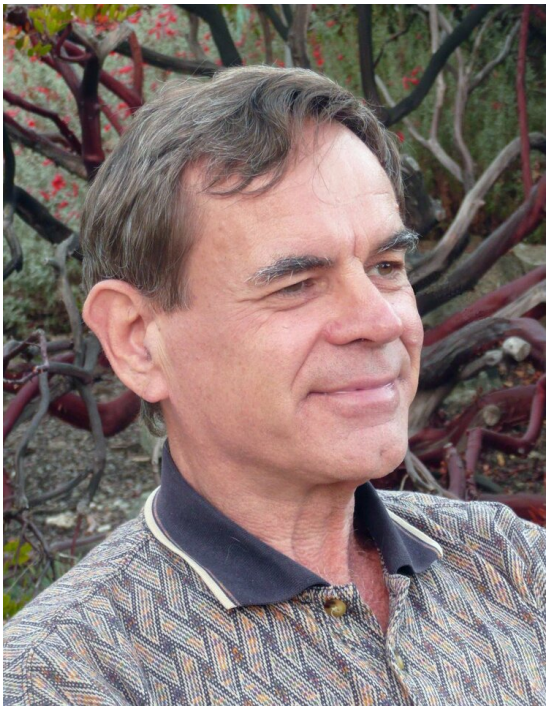
https://zpe.gov.pl/b/PbZpElT2K

W pliku `hasla.txt` danych jest `200` haseł użytkowników pewnego systemu. Każdy użytkownik ma jedno hasło (zapisane w osobnym wierszu), które zawiera od 1 do 20

znaków alfanumerycznych, tzn. cyfr od 0 do 9 lub liter alfabetu łacińskiego (małych lub wielkich). Polityka bezpieczeństwa systemu wymaga, aby hasła były odpowiednio skomplikowane i nie powtarzały się.

Hasła te należy wczytać do 200-elementowej tablicy.

2



Vaughan Pratt – jeden z twórców algorytmu
Źródło: Vaughan Pratt , en.wikipedia.org, dostęp: 21.12.2022, CC BY-SA 3.0.

Materiał audio dostępny pod adresem:

https://zpe.gov.pl/b/PbZpElT2K

Wypisz hasła, które zostały użyte przez co najmniej dwóch różnych użytkowników, tzn. występujące w dwóch różnych wierszach. Hasła wypisz (bez powtórzeń) w kolejności leksykograficznej.

Zadanie zostało opracowane przez Centralną Komisję Egzaminacyjną i znajduje się w Maturalnym zbiorze zadań z informatyki jako zadanie nr 74.2. Ze zbiorem można zapoznać się na oficjalnej stronie cke.gov.pl.

3

James H. Morris – jeden z twórców algorytmu
Źródło: Stanford University, web.stanford.edu, dostęp: 21.12.2022, tylko do użytku edukacyjnego.

Materiał audio dostępny pod adresem:

https://zpe.gov.pl/b/PbZpElT2K

Pisząc kod, użyjemy funkcji `sortuj()`, która uporządkuje zawartość tablicy metodą słownikową (leksykograficznie).
W dozwolonych na egzaminie maturalnym językach programowania można posortować elementy wg ich kodów ASCII – będzie to jednoznaczne z posortowaniem ich w kolejności leksykograficznej.

Zaczniemy od wczytania danych:

4

Materiał audio dostępny pod adresem:

https://zpe.gov.pl/b/PbZpElT2K

Następnie posortujemy zawartość tablicy:

W językach programowania możemy użyć wbudowanej funkcji sortującej:

- w języku Python użyjemy `hasła.sort()`
- w języku C++ użyjemy `sort(hasła, hasła + długość(hasła));`
- w języku Java użyjemy `Arrays.sort(hasła)`

W każdym z tych języków zmodyfikowana zostaje tablica wejściowa.

5

Materiał audio dostępny pod adresem:

https://zpe.gov.pl/b/PbZpElT2K

Kolejną czynnością będzie uruchomienie pętli `dla`. Jej licznik przyjmie wartości od 0 do 199, wskazując kolejne indeksy tablicy:

6

Materiał audio dostępny pod adresem:

https://zpe.gov.pl/b/PbZpElT2K

Zdefiniujemy zmienną `j`, której przypiszemy wartość 0. Posłuży nam ona do sprawdzenia, ile kolejnych haseł jest takich samych.

7

Materiał audio dostępny pod adresem:

https://zpe.gov.pl/b/PbZpElT2K

Użyjemy pętli `dopóki`, która będzie działać tak długo, jak długo wyrażenie `(i + j + 1)` będzie mniejsze niż 199 (gwarantuje to, że sprawdzimy 200 haseł, czyli nie wykroczymy poza zakres tablicy i nie odwołamy się do nieistniejącego elementu).

Drugi warunek zapisany w pętli `(hasła[i + j + 1] == hasła[i])` pozwala sprawdzić, czy dwa kolejne hasła są identyczne. Posortowaliśmy wcześniej zawartość tablicy alfabetycznie, więc dwa takie same elementy pojawią się obok siebie:

Po każdym cyklu pętli zwiększamy wartość zmiennej `j`.

8

Materiał audio dostępny pod adresem:

https://zpe.gov.pl/b/PbZpElT2K

Następnie zastosujemy instrukcję warunkową `jeżeli`, która zostanie wykonana, gdy zmienna `j` będzie większa od 0 (czyli gdy hasło się powtórzy):

9

Materiał audio dostępny pod adresem:

https://zpe.gov.pl/b/PbZpElT2K

Jeżeli warunek zostanie spełniony, wyświetlamy hasło oraz zwiększamy zmienną `i` o wartość zmiennej `j`:

Przedstawiliśmy sposób rozwiązania zadania w postaci pseudokodu. Według schematu oceniania punkty przyznawane są wyłącznie w przypadku podania wszystkich powtarzających się haseł.

10

Materiał audio dostępny pod adresem:

https://zpe.gov.pl/b/PbZpElT2K

Zapoznaj się z rozwiązaniami, które odnoszą się nie do pliku, a jego fragmentu zamieszczonego w tablicy.

Oto implementacja kodu w języku C++:

Implementacja w języku Java:

Implementacja w języku Python:



**Polecenie 2**

Zastanów się, w jaki sposób należy zmodyfikować program, by pobierał dane z pliku i zapisywał je do tablicy.

# Sprawdź się

Pokaż ćwiczenia:

## Testy genetyczne

Naukowcy Bajtolandii odkryli, że mieszkańcy będący nosicielami konkretnego genu, są znacznie bardziej podatni na alergię. Bajtolandzkie Ministerstwo Zdrowia postanowiło zatem, że to oni w pierwszej kolejności będą brać udział w finansowanym przez państwo programie odczulania.

Dany jest plik `DNA.txt` zawierający `100` łańcuchów znaków. Każdy ciąg składa się ze 100 znaków.

Plik o rozmiarze 9.96 KB w języku polskim

Napisz program wyznaczający, którzy obywatele Bajtolandii są nosicielami genu odpowiedzialnego za alergię. Program powinien drukować TAK, jeśli dany gen występuje u osobnika, lub NIE jeżeli gen nie występuje. Wynik zapisz do pliku `DNA2.txt`.

**Do oceny oddajesz:**

- plik `DNA.txt` zawierający odpowiedź (wypisane jeden pod drugim łańcuchy znaków TAK i NIE)
- plik(i) z komputerową realizacją zadania (kodem programu).

> JĘZYK C++

**Ćwiczenie 1**

## Twoje zadania

1. Program sprawdza, które łańcuchy DNA zawierają mutację odpowiedzialną za alergię.

**Ćwiczenie 2**

## Twoje zadania

1. Program sprawdza, które łańcuchy DNA zawierają mutację odpowiedzialną za alergię.

**Ćwiczenie 3**

## Twoje zadania

1. Program sprawdza, które łańcuchy DNA zawierają mutację odpowiedzialną za alergię.

Plik DNA2.txt z danymi spełniającymi warunki zadań
Plik o rozmiarze 502.00 B w języku polskim

## Zadanie 1.2

W pliku `hasla.txt` danych jest 200 haseł użytkowników pewnego systemu. Każdy użytkownik posiada jedno hasło (każde zapisane jest w osobnym wierszu), które zawiera od 1 do 20 znaków alfanumerycznych, tzn. cyfr od 0 do 9 lub liter alfabetu łacińskiego (małych lub dużych). Polityka bezpieczeństwa systemu wymaga, aby hasła były odpowiednio skomplikowane i nie powtarzały się.

Plik o rozmiarze 2.09 KB w języku polskim

Poniżej podano pierwsze pięć haseł zapisanych w pliku `hasla.txt`:

```
ZXUhkPLcjKo
ikfLDegQXj
8Y7JGYXXR5
603624722555
50q4252ax5
```

Napisz program, który da odpowiedzi do poniższych zadań. Odpowiedzi do poszczególnych zadań zapisz w pliku tekstowym `wyniki_hasla.txt`. Wyniki do każdego zadania poprzedź numerem oznaczającym to zadanie.

Podaj liczbę użytkowników posiadających hasła, w których występuje fragment złożony z czterech kolejnych znaków ASCII (w dowolnej kolejności).

Przykłady haseł zawierających taki fragment to:

```
A5mnpoR89cd
A5876RRcg
As45FGHFEk90nba
```

> Zadanie zostało opracowane przez Centralną Komisję Egzaminacyjną i znajduje się w Maturalnym zbiorze zadań z informatyki jako zadanie nr 74.3. Ze zbiorem można zapoznać się na oficjalnej stronie cke.gov.pl.
> Rozwiąż zadanie, posługując się dowolnym językiem programowania. Dane, na których twój program powinien działać, znajdziesz w pliku tekstowym. W testerce wykorzystaj dane umieszczone w tablicy.
>
> Dla danych z pliku poprawna odpowiedź to 39.

**Specyfikacja problemu:**

*Dane*:

- `hasla` – tablica ciągów znaków

*Wynik:*

- `x` – liczba całkowita określająca liczbę użytkowników mających hasła, w których występuje fragment złożony z czterech kolejnych znaków ASCII

JĘZYK C++

**Ćwiczenie 4**

## Twoje zadania

1. Program podaje liczbę haseł, w których pojawia się sekwencja złożona z czterech kolejnych znaków ASCII (ułożonych w dowolnej kolejności).

**Ćwiczenie 5**

## Twoje zadania

1. Program podaje liczbę haseł, w których pojawia się sekwencja złożona z czterech kolejnych znaków ASCII (ułożonych w dowolnej kolejności).

**Ćwiczenie 6**

## Twoje zadania

1. Program podaje liczbę haseł, w których pojawia się sekwencja złożona z czterech kolejnych znaków ASCII (ułożonych w dowolnej kolejności).

## Zadanie 1.3

Podaj liczbę haseł, które spełniają jednocześnie następujace warunki:

- hasło zawiera co najmniej jeden znak numeryczny, tzn. cyfrę od 0 do 9;
- hasło zawiera co najmniej jedną małą literę;
- hasło zawiera co najmniej jedną wielką literę.

> Zadanie zostało opracowane przez Centralną Komisję Egzaminacyjną i znajduje się w Maturalnym zbiorze zadań z informatyki jako zadanie nr 74.4. Ze zbiorem można zapoznać się na oficjalnej stronie cke.gov.pl.
> Rozwiąż zadanie, posługując się dowolnym językiem programowania. Dane, na których twój program powinien działać, znajdziesz w pliku tekstowym. W testerce wykorzystaj dane umieszczone w tablicy.
>
> Dla danych z pliku poprawna odpowiedź to 40.
>
> **Specyfikacja problemu:**
>
> *Dane*:
>
> - `hasla` - tablica ciągów znaków
>
> *Wynik:*
>
> - `x` - liczba całkowita; liczba haseł spełniających określone warunki

> JĘZYK C++

**Ćwiczenie 7**

## Twoje zadania

1. Program podaje, ile w tablicy `hasla` jest haseł, które spełniają wszystkie podane kryteria.

**Ćwiczenie 8**

## Twoje zadania

1. Program podaje, ile w tablicy `hasla` jest haseł, które spełniają wszystkie podane kryteria.

**Ćwiczenie 9**

## Twoje zadania

1. Program podaje, ile w tablicy `hasla` jest haseł, które spełniają wszystkie podane kryteria.

# Dla nauczyciela

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

**Temat: Algorytm Knutha-Morrisa-Pratta – 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.

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.

IV. Rozwijanie kompetencji społecznych, takich jak: komunikacja i współpraca w grupie, w tym w środowiskach wirtualnych, udział w projektach zespołowych oraz zarządzanie projektami.

Treści nauczania – wymagania szczegółowe

I. Rozumienie, analizowanie i rozwiązywanie problemów.

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;

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

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

3) objaśnia, a także porównuje podstawowe metody i techniki algorytmiczne oraz struktury danych, wykorzystując przy tym przykłady problemów i algorytmów, w szczególności:

g) metodę haszowania (wyszukiwanie wzorca w tekście),

**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):**

- Zastosujesz algorytm KMP w celu wyszukania wzorca w tekście.
- Przeanalizujesz stopień trudności zadań maturalnych wymagających zastosowania algorytmów tekstowych.
- Rozwiążesz problemy wymagające użycia algorytmów tekstowych.
- Prześledzisz przykładowe zadania maturalne opracowane przez CKE.

**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 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. **Przygotowanie do zajęć.** Nauczyciel loguje się na platformie i udostępnia e-materiał: „Algorytm Knutha-Morrisa-Pratta – zadania maturalne". Uczniowie zapoznają się z treściami w sekcji „Przeczytaj" w kontekście programowania.

**Faza wstępna:**

1. Przedstawienie tematu zajęć oraz wspólne z uczniami ustalenie kryteriów sukcesu.

**Faza realizacyjna:**

1. Uczniowie w parach przypominają sobie najważniejsze informacje dotyczące algorytmu Knutha–Morrisa–Pratta.
2. **Praca z multimedium.** Uczniowie pracują w parach. Analizują treść polecenia nr 1: „Przeanalizuj prezentację, a następnie w wybranym języku programowania napisz program rozwiązujący zaprezentowany problem." z sekcji „Prezentacja multimedialna". Wybrana grupa omawia rozwiązanie na forum klasy.
3. **Ćwiczenie umiejętności.** Uczniowie indywidualnie wykonują zadanie 1.2 z sekcji „Sprawdź się".

**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).
2. Na koniec zajęć z programowania nauczyciel prosi uczniów o rozwinięcie zdania: „Na dzisiejszych zajęciach nauczyłam/łem się jak…".

**Praca domowa:**

1. Uczniowie wykonują ćwiczenie 1.3 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.
- Dane umieszczone w pliku `hasła` można wykorzystać do lekcji o bezpiecznych hasłach. Temat zadania można wykorzystać jako inspirację do napisania generatora haseł lub do obliczania siły hasła.