Szyfr Cezara – zadania maturalne

Szyfr Cezara – zadania maturalne

Źródło: Pixabay, domena publiczna.

Poznaliśmy już szyfr Cezara, czyli klasyczny szyfr przesuwający (podstawieniowy). Jego działanie polega na zastąpieniu każdej z liter tekstu jawnego odpowiadającą jej literą oddaloną o określoną liczbę miejsc w alfabecie.

W tym e-materiale zapoznamy się z przykładowymi zadaniami maturalnymi dotyczącymi tego zagadnienia.

Implementacje w poszczególnych językach programowania przedstawiamy w e-materiałach:

- Implementacja szyfru Cezara w języku C++,
- Implementacja szyfru Cezara w języku Java,
- Szyfr Cezara w języku Python.

Twoje cele

- Przeanalizujesz zadania maturalne, których treść jest związana z algorytmami kryptograficznymi.
- Prześledzisz schemat oceniania zadań maturalnych.
- Rozwiążesz przykładowe zadania maturalne, wymagające zastosowania szyfru Cezara.

Przeczytaj

Na egzaminie maturalnym z informatyki od czasu do czasu pojawiają się zadania związane z szyframi – w tym z szyframi przesuwającymi, do których należy szyfr Cezara. Warto umieć z niego korzystać: sprawdźmy się, rozwiązując przykładowe zadania maturalne.

Zadanie maturalne

W pliku `dane_6_1.txt` znajduje się 100 słów. Słowa umieszczono w osobnych wierszach.

Fragment pliku `dane_6_1.txt`:
```
INTERPRETOWANIE
ROZWESELANIE
KONSERWOWANIE
```

Napisz program, który **zaszyfruje** słowa z pliku `dane_6_1.txt` z użyciem klucza k = 107. Wynik zapisz w pliku `wyniki_6_1.txt`. Każde słowo umieść w osobnym wierszu, w porządku odpowiadającym kolejności słów z pliku z danymi.

Uwaga:

Dla pierwszego słowa z pliku `dane_6_1.txt` (`INTERPRETOWANIE`) wynikiem jest słowo `LQWHUSUHWRZDQLH`.

> Przedstawione zadanie zostało opracowane przez Centralną Komisję Egzaminacyjną i pojawiło się na maturze z informatyki w maju 2016 roku (poziom rozszerzony, cześć II). Z pełnym arkuszem można się zapoznać na oficjalnej stronie CKE.

Załóżmy, że słowa – zamiast w pliku – znajdują się w tablicy łańcuchów znaków `słowa`.

Ponieważ podczas egzaminu maturalnego każdy uczeń może wybrać dowolny z dostępnych języków programowania, przedstawimy rozwiązanie zadania w pseudokodzie. Twoją rolą będzie przełożenie proponowanego pseudokodu na język, w którym programujesz.

W kodzie użyjemy funkcji `pobierz_ASCII()`, która zwróci kod ASCII podanego znaku.

Wykorzystamy również funkcję `długość()`, która zwróci długość podanego łańcucha znaków.

Ostatnią z funkcji, które wykorzystamy, będzie `znak_ASCII()`, zamieniająca kod ASCII w znak odpowiadający mu w tablicy ASCII.

Rozwiązywanie zadania zaczniemy od zadeklarowania tablicy łańcuchów znaków zawierającej 100 słów. Zainicjujemy również zmienną `k`, w której umieścimy klucz szyfrowania.

```
słowa[100] = tablica łańcuchów znaków zawierająca 100 słów
k = 107 mod 26
```

Jako klucz wprowadziliśmy resztę z dzielenia 107 przez 26. Wynika to z faktu, że stosując szyfr Cezara nie możemy użyć klucza dłuższego niż liczba znaków składających się na alfabet łaciński. A to właśnie miałoby miejsce, gdyby klucz był większy od 26, czyli od liczby liter w alfabecie łacińskim.

Następnie rozpoczniemy pętlę `dla`, której zmienna iteracyjna będzie przyjmować wartości od 0 do 99. Inaczej mówiąc, przyjmie ona kolejno wartość indeksu każdego elementu w tablicy ze słowami. Pętla ta będzie służyć do pobierania kolejnych słów z tablicy łańcuchów znaków `słowa` w celu ich zaszyfrowania.

```
słowa[100] = tablica łańcuchów znaków zawierająca 100 słów
k = 107 mod 26

dla i = 0, 1, ..., długość(słowa) - 1 wykonuj:
```

Wewnątrz pętli zadeklarujemy łańcuch znaków `szyfrowanie`, któremu przypiszemy poszczególne elementy tablicy `słowa[i]`. Następnie utworzymy kolejną pętlę `dla`, w której będziemy przesuwać znaki z podanego słowa na kolejne pozycje w alfabecie.

```
słowa[100] = tablica łańcuchów znaków zawierająca 100 słów
k = 107 mod 26

dla i = 0, 1, ..., długość(słowa) - 1 wykonuj:
    szyfrowanie = slowa[i]

    dla j = 0, 1, ..., długość(szyfrowanie) - 1 wykonuj:
        kod = pobierzASCII(szyfrowanie[j])
        kod = kod + k

        jeżeli kod > pobierzASCII('Z') wykonaj:
            kod = kod - 26
```

Zaczniemy od pobrania kodu ASCII każdej litery szyfrowanego słowa, a następnie dodamy do niego zmienną `k`, czyli resztę z dzielenia klucza szyfrowania przez 26.

Zastosujemy dalej instrukcję warunkową `jeżeli`, która zostanie wykonana, gdy kod szyfrowanego znaku po przesunięciu o `k` miejsc znajdzie się poza alfabetem łacińskim. W takim przypadku zmniejszymy wartość kodu ASCII o 26. W rezultacie kod ASCII odpowiadający literze znowu znajdzie się w zakresie przypisanym alfabetowi.

Teraz pozostaje przypisać zawartość łańcucha znaków `szyfrowanie`, w której znajduje się zaszyfrowane słowo, do odpowiadającego jej miejsca w tablicy łańcuchów znaków `słowa`.

```
słowa[100] = tablica łańcuchów znaków zawierająca 100 słów
k = 107 mod 26

dla i = 0, 1, ..., długość(słowa) - 1 wykonuj:
    szyfrowanie = slowa[i]

    dla j = 0, 1, ..., długość(szyfrowanie) - 1 wykonuj:
        kod = pobierzASCII(szyfrowanie[j])
        kod = kod + k

        jeżeli kod > pobierzASCII('Z') wykonaj:
            kod = kod - 26

        szyfrowanie[j] = znakASCII(kod)
    slowa[i] = szyfrowanie
```

W ten sposób udało się rozwiązać zadanie. Po wykonaniu przedstawionego wyżej kodu w tablicy `słowa` znajdą się zaszyfrowane wyrazy.

> **Praca domowa**
>
> Zapisz program w wybranym przez siebie języku programowania.

Schemat oceniania

3 p. – za poprawny plik wynikowy

2 p. – za pominięcie ostatniego wiersza

1 p. – za plik z błędnym wykonaniem zawinięcia cyklicznego albo bez zawijania

0 p. – za odpowiedź błędną albo za brak odpowiedzi

Słownik

klucz szyfrowania

w przypadku szyfru przesuwającego (np. szyfru Cezara) – liczba miejsc w alfabecie, o które ma zostać przesunięta każda litera tekstu jawnego

Gra edukacyjna

Polecenie 1

Przeanalizuj prezentację, a następnie zastanów się, na czym polega wspomniany w schemacie oceniania błąd zawijania.

1

Materiał audio dostępny pod adresem:

https://zpe.gov.pl/b/P9N4LtutO

W zadaniu maturalnym, które wykonamy, wykorzystamy szyfr Cezara.

2

Materiał audio dostępny pod adresem:

https://zpe.gov.pl/b/P9N4LtutO

W pliku `dane_6_2.txt` zapisano 3000 szyfrogramów i odpowiadające im klucze szyfrujące. W każdym wierszu znajduje się jeden szyfrogram (zaszyfrowane słowo) oraz, po pojedynczym znaku odstępu, odpowiadający mu klucz (maksymalnie czterocyfrowa liczba).

Napisz program, który odszyfruje słowa zaszyfrowane podanymi kluczami. Wynik zapisz w pliku `wyniki_6_2.txt`: każde odszyfrowane słowo umieść w osobnym wierszu, w porządku odpowiadającym kolejności szyfrogramów z pliku z danymi.

Przedstawione zadanie zostało opracowane przez Centralną Komisję Egzaminacyjną i pojawiło się na egzaminie maturalnym z informatyki w maju 2016 roku (poziom

rozszerzony, część II). Z pełnym arkuszem można zapoznać się na oficjalnej stronie CKE.

3

Materiał audio dostępny pod adresem:

https://zpe.gov.pl/b/P9N4LtutO

Dla potrzeb prezentacji przyjmiemy, że szyfrogramy i ich klucze szyfrujące znajdują się w tablicy łańcuchów znaków `szyfrogramy`. Importowanie danych z pliku omówione zostało w czasie jednej z wcześniejszych lekcji. Będzie ono wyglądało inaczej w przypadku każdego z języków: C++, Java oraz Python.

Rozwiązanie zadania przedstawimy w pseudokodzie.

Użyjemy następujących funkcji:

- `długość()` - zwracającej długość podanego jej ciągu znaków,

- `pobierzASCII()` - zwracającej kod ASCII podanego jej znaku,

- `znakASCII()` - zwracającej znak odpowiadający podanemu jej kodowi z tablicy ASCII.

4

Materiał audio dostępny pod adresem:

https://zpe.gov.pl/b/P9N4LtutO

Zaczniemy od zadeklarowania tablicy z szyfrogramami oraz tablicy `słowa`, w której umieścimy odszyfrowane słowa.

```
1 szyfrogramy[3000][2] =
  dwuwymiarowa tablica
```

```
łańcuchów znaków,
zawierająca 3000 wierszy,
w których znajdują się
szyfrogram oraz klucz,
którym został on
zaszyfrowany
słowa[3000] = tablica
zawierająca 3000 pustych
łańcuchów znaków
```

5

Materiał audio dostępny pod adresem:

https://zpe.gov.pl/b/P9N4LtutO

Teraz zadeklarujemy pętlę `dla`, w której przechodzić będziemy pomiędzy poszczególnymi wierszami tablicy. Jej zmienna iteracyjna przyjmie wartości od 0 do 2999.

```
szyfrogramy[3000][2] =
dwuwymiarowa tablica
łańcuchów znaków,
zawierająca 3000 wierszy,
w których znajdują się
szyfrogram oraz klucz,
którym został on
zaszyfrowany
słowa[3000] = tablica
zawierająca 3000 pustych
łańcuchów znaków

dla i = 0, 1, ...,
długość(szyfrogramy) - 1
wykonuj:
```

6

Materiał audio dostępny pod adresem:

https://zpe.gov.pl/b/P9N4LtutO

Wewnątrz pętli z kolejnych wierszy z szyfrogramami pobierać będziemy szyfrogram oraz odpowiadający mu klucz szyfrowania, a następnie umieszczać je w odpowiednich zmiennych. Oczywiście przy pobieraniu klucza do zmiennej zapiszemy jego resztę z dzielenia przez 26.

```
szyfrogramy[3000][2] = dwuwymiarowa tablica łańcuchów znaków, zawierająca 3000 wierszy, w których znajdują się szyfrogram oraz klucz, którym został on zaszyfrowany
słowa[3000] = tablica zawierająca 3000 pustych łańcuchów znaków

dla i = 0, 1, ..., długość(szyfrogramy) - 1 wykonuj:
    szyfrogram = szyfrogramy[i][0]
    klucz = szyfrogramy[i][1] % 26
```

7

Materiał audio dostępny pod adresem:

https://zpe.gov.pl/b/P9N4LtutO

Teraz wprowadzimy kolejna pętlę `dla`, w której przeprowadzimy proces deszyfrowania za pomocą instrukcji warunkowej `jeżeli` oraz funkcji zadeklarowanych na początku prezentacji.

```
szyfrogramy[3000][2] = dwuwymiarowa tablica łańcuchów znaków, zawierająca 3000 wierszy, w których znajdują się szyfrogram oraz klucz, którym został on zaszyfrowany
słowa[3000] = tablica zawierająca 3000 pustych łańcuchów znaków

dla i = 0, 1, ..., długość(szyfrogramy) - 1 wykonuj:
    szyfrogram = szyfrogramy[i][0]
    klucz = szyfrogramy[i][1] % 26

    dla j = 0, 1, ..., długość(szyfrogram) – 1 wykonuj:
        kod = pobierzASCII(szyfrogram[j])
        kod = kod - klucz

        jeżeli kod < pobierzASCII('A') to
            kod += 26

        szyfrogram[j] = znakASCII(kod)
```

8

Materiał audio dostępny pod adresem:

https://zpe.gov.pl/b/P9N4LtutO

W tym momencie kod odszyfrowuje już słowa za pomocą ich klucza szyfrowania. Pozostaje

dodać kod przypisujący odszyfrowane słowa odpowiednim elementom tablicy łańcuchów `słowa`, w której zamieścimy wyniki.

```
szyfrogramy[3000][2] = dwuwymiarowa tablica łańcuchów znaków, zawierająca 3000 wierszy, w których znajdują się szyfrogram oraz klucz, którym został on zaszyfrowany
słowa[3000] = tablica zawierająca 3000 pustych łańcuchów znaków

dla i = 0, 1, ..., długość(szyfrogramy) - 1 wykonuj:
    szyfrogram = szyfrogramy[i][0]
    klucz = szyfrogramy[i][1] % 26

    dla j = 0, 1, ..., długość(szyfrogram) – 1 wykonuj:
        kod = pobierzASCII(szyfrogram[j])
        kod = kod - klucz

        jeżeli kod < pobierzASCII('A') to
            kod += 26

        szyfrogram[j] = znakASCII(kod)
    słowa[i] = szyfrogram
```

Tym sposobem udało nam się ukończyć zadanie.

9

Materiał audio dostępny pod adresem:

https://zpe.gov.pl/b/P9N4LtutO

Oto schemat oceniania:

4 p. - za poprawny plik wynikowy;

2 p. - za błędne dekodowanie jednej z liter alfabetu lub błędne rozwiązanie wynikające z błędu zawijania lub za błędne rozwiązanie wynikające z przyjęcia błędnej długości alfabetu (25);

0 p. - za odpowiedź błędną lub brak odpowiedzi.

Schemat został przygotowany przez CKE. Można go znaleźć na oficjalnej stronie internetowej CKE.

10

Materiał audio dostępny pod adresem:

https://zpe.gov.pl/b/P9N4LtutO

Możemy otrzymać punkty nawet wtedy, gdy odpowiedź nie będzie w pełni prawidłowa. Błąd zawijania (wspomniany we wstępie) dotyczy przypadku, gdy nie użylibyśmy instrukcji warunkowej `jeżeli` w celu zwiększenia zmiennej `kod` o 26 (kiedy jej kod ASCII jest mniejszy niż ten, który odpowiada literze ‘A’).

Polecenie 2

Sprawdź swoją wiedzę, biorąc udział w grze.

Test

Sprawdź swoją wiedzę o szyfrze Cezara, biorąc udział w grze

Poziom trudności:

łatwy

Limit czasu:

7 min

Twój ostatni wynik:

-

Uruchom

Sprawdź się

> Zadania są inspirowane zadaniem nr 6, które pojawiło się na egzaminie maturalnym w maju 2016 (część II arkusza, poziom rozszerzony).

> W pliku `dane.txt` zapisano 3 000 par słów, po jednej parze w wierszu, oddzielonych pojedynczym znakiem odstępu. Drugie słowo w każdej parze jest szyfrogramem pierwszego z nieznanym kluczem. Słowa zapisane są wielkimi literami.
>
> Fragment pliku `dane.txt`:
>
> ```
> ZAWISLAK EFBNXQFP
>
> KRASZEWSKI XENFMRJFXV
> ```
>
> Napisz program, który dla każdego wiersza tablicy wyświetli zastosowany klucz szyfrowania. Dla każdego wiersza klucz ten powinien być dodatni, a zarazem jak najmniejszy. Wypisz kolejne klucze po znakach spacji.
>
> Plik dane2.txt
> Plik o rozmiarze 52.58 KB w języku polskim

Przetestuj działanie programu dla następującego fragmentu pliku `dane.txt`:

```
ZAWISLAK EFBNXQFP
KRASZEWSKI XENFMRJFXV
WARDA OSJVS
DRAB LZIJ
MOKRZYCKI FHDKSRVDB
LAKOMY FUEIGS
POWROZNIK LKSNKVJEG
BABIARZ DCDKCTB
RACZKA JSURCS
KREZEL JQDYDK
```

Specyfikacja problemu:

Dane:

- `szyfrogram` – tablica łańcuchów znaków

Wynik:

Program wypisuje w jednej linii oddzielone od siebie znakiem spacji kolejne klucze szyfrowania.

Przykładowe wyjście:

```
5 13 18 8 19 20 22 2 18 25
```

Pokaż ćwiczenia: 

> **Język C++**

Ćwiczenie 1

Twoje zadania

1. Program wypisuje klucze użyte do zaszyfrowania słów w wierszach.

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

int main() {
    string szyfrogram[10][2] = {
    {"ZAWISLAK", "EFBNXQFP" },
    {"KRASZEWSKI", "XENFMRJFXV" },
    {"WARDA", "OSJVS" },
    {"DRAB", "LZIJ" },
    {"MOKRZYCKI", "FHDKSRVDB" },
    {"LAKOMY", "FUEIGS" },
    {"POWROZNIK", "LKSNKVJEG" },
    {"BABIARZ", "DCDKCTB" },
```

```cpp

```

Język Java

Ćwiczenie 2

Twoje zadania

1. Program wypisuje klucze użyte do zaszyfrowania słów w wierszach.

```java
public class Main {
    public static void main(String[] args) {
        String [][] szyfrogram = {
            {"ZAWISLAK", "EFBNXQFP" },
            {"KRASZEWSKI", "XENFMRJFXV" },
            {"WARDA", "OSJVS" },
            {"DRAB", "LZIJ" },
            {"MOKRZYCKI", "FHDKSRVDB" },
            {"LAKOMY", "FUEIGS" },
            {"POWROZNIK", "LKSNKVJEG" },
            {"BABIARZ", "DCDKCTB" },
            {"RACZKA", "JSURCS" },
            {"KREZEL", "JQDYDK" },
        };
```

```
```

Język Python

Ćwiczenie 3

Twoje zadania

1. Program wypisuje klucze użyte do zaszyfrowania słów w wierszach.

```
szyfrogram = [
    [ "ZAWISLAK", "EFBNXQFP" ],
    [ "KRASZEWSKI", "XENFMRJFXV" ],
    [ "WARDA", "OSJVS" ],
    [ "DRAB", "LZIJ" ],
    [ "MOKRZYCKI", "FHDKSRVDB" ],
    [ "LAKOMY", "FUEIGS" ],
    [ "POWROZNIK", "LKSNKVJEG" ],
    [ "BABIARZ", "DCDKCTB" ],
    [ "RACZKA", "JSURCS" ],
    [ "KREZEL", "JQDYDK" ],
]
```

```

```

Odpowiedź dla danych z pliku `dane.txt`:

```
5 13 18 8 19 20 22 2 18 25 2 7 15 1 18 9 11 2 25 26 12 7 23 14 7
```

> W pliku `dane2.txt` zapisano 3 000 par słów, po jednej parze w wierszu, oddzielonych pojedynczym znakiem odstępu. Drugie słowo w każdej parze jest szyfrogramem pierwszego z nieznanym kluczem. Słowa zapisane są wielkimi literami. Niektóre szyfrogramy są błędne, co oznacza, że niektóre litery w słowie zakodowano z różnymi przesunięciami. Słowo ma zawsze tę samą długość co odpowiadający mu szyfrogram. Do zaszyfrowania wszystkich słów został użyty klucz szyfrujący zapisany w postaci zmiennej `klucz`.
>
> Napisz program, który obliczy, w ilu wierszach pojawiły się niepoprawne szyfrogramy.
>
> Fragment pliku `dane2.txt`:
>
> ```
> ZAWISLAK EFBNXQFP
>
> KRASZEWSKI XENFMRJFXV
> ```
>
> Plik dane2.txt
> Plik o rozmiarze 52.58 KB w języku polskim

Przetestuj działanie programu dla następującego fragmentu pliku `dane2.txt`:

```
ZAWISLAK KLHTDWLV
KRASZEWSKI VCLDKPHDVT
WARDA JNEQN
DRAB QENO
MOKRZYCKI ZBXEMLPXV
LAKOMY APZDBN
POWROZNIK EDLGDOCXZ
BABIARZ ONOVNEM
RACZKA ENPMXN
KREZEL XERMRY
```

Specyfikacja problemu:

Dane:

- `szyfrogram` – tablica łańcuchów znaków
- `klucz` – liczba naturalna; wartość klucza szyfrującego

Wynik:

- `wynik` – liczba naturalna

> Język C++

Ćwiczenie 4

Twoje zadania

1. Program sprawdza, w ilu wierszach wystąpiło niepoprawne szyfrowanie.

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

int main() {
    int klucz = 13;
    std::string szyfrogram[10][2] = {
        {"ZAWISLAK", "KLHTDWLV" },
        {"KRASZEWSKI", "VCLDKPHDVT" },
        {"WARDA", "JNEQN" },
        {"DRAB", "QENO" },
        {"MOKRZYCKI", "ZBXEMLPXV" },
        {"LAKOMY", "APZDBN" },
        {"POWROZNIK", "EDLGDOCXZ" },
```

```cpp

```

Ćwiczenie 5

Twoje zadania

1. Program sprawdza, w ilu wierszach wystąpiło niepoprawne szyfrowanie.

```
klucz = 13
szyfrogram = [
    ["ZAWISLAK", "KLHTDWLV" ],
    ["KRASZEWSKI", "VCLDKPHDVT" ],
    ["WARDA", "JNEQN" ],
    ["DRAB", "QENO" ],
    ["MOKRZYCKI", "ZBXEMLPXV" ],
    ["LAKOMY", "APZDBN" ],
    ["POWROZNIK", "EDLGDOCXZ" ],
    ["BABIARZ", "ONOVNEM" ],
    ["RACZKA", "ENPMXN" ],
    ["KREZEL", "XERMRY" ]
]

```

```

```

Ćwiczenie 6

Twoje zadania

1. Program sprawdza, w ilu wierszach wystąpiło niepoprawne szyfrowanie.

```java
public class Main {
    public static void main(String[] args) {
        int klucz = 13;
        String [][] szyfrogram = {
            {"ZAWISLAK", "KLHTDWLV" },
            {"KRASZEWSKI", "VCLDKPHDVT" },
            {"WARDA", "JNEQN" },
            {"DRAB", "QENO" },
            {"MOKRZYCKI", "ZBXEMLPXV" },
            {"LAKOMY", "APZDBN" },
            {"POWROZNIK", "EDLGDOCXZ" },
            {"BABIARZ", "ONOVNEM" },
            {"RACZKA", "ENPMXN" },
            {"KREZEL", "XERMRY" },
```

```
1
```

Odpowiedź dla danych z pliku `dane2.txt`:

```
27
```

Dla nauczyciela

Autor: Wojciech Malicki
Przedmiot: Informatyka

Temat: Szyfr Cezara – zadania maturalne

Grupa docelowa:

Szkoła ponadpodstawowa, liceum ogólnokształcące, technikum, zakres podstawowy i 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 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):

- Przeanalizujesz zadania maturalne, których treść jest związana z algorytmami kryptograficznymi.
- Prześledzisz schemat oceniania zadań maturalnych.
- Rozwiążesz przykładowe zadania maturalne, wymagające zastosowania szyfru Cezara.

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;
- 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. Chętna lub wybrana osoba przypomina najważniejsze informacje dotyczące szyfru Cezara.
2. **Przygotowanie do zajęć.** Nauczyciel loguje się na platformie i udostępnia e-materiał: „Szyfr Cezara – zadania maturalne". Nauczyciel prosi uczniów o zapoznanie się z treściami w sekcji „Przeczytaj".

Faza wstępna:

1. Nauczyciel sprawdza przygotowanie grupy do zajęć.
2. Nauczyciel wyświetla uczniom temat, wskazuje cele zajęć oraz ustala z uczestnikami zajęć kryteria sukcesu.

Faza realizacyjna:

1. **Praca z tekstem.** Uczniowie, którzy nie zapoznali się z treścią sekcji „Przeczytaj” w domu, robią to w czasie lekcji. Pozostali uczniowie mogą zacząć wykonywać zadanie domowe zamieszczone w sekcji „Przeczytaj”.
2. Uczniowie indywidualnie rozwiązują ćwiczenia z sekcji „Sprawdź się”. Chętne lub wybrane osoby prezentują swój kod na forum klasy. Nauczyciel omawia rozwiązania.
3. Uczniowie w grupach dyskutują na temat swoich rozwiązań.

Faza podsumowująca:

1. Na koniec zajęć nauczyciel prosi uczniów o rozwinięcie zdania: „Na dzisiejszych zajęciach nauczyłam/łem się jak…”.

Praca domowa:

1. Uczniowie wykonują zadania z sekcji „Gra edukacyjna”. Analizują prezentację a następnie odpowiadają na pytanie: na czym polega wspomniany w schemacie oceniania błąd zawijania? Następnie wykonują interaktywny test „Sprawdź swoja wiedzę”.

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:

- Nauczyciel może wykorzystać multimedium w sekcji „Gra edukacyjna” do pracy przed lekcją. Uczniowie zapoznają się z jego treścią i przygotowują do pracy na zajęciach w ten sposób, żeby móc samodzielnie rozwiązać zadania dołączone do e-materiału „Szyfr Cezara – zadania maturalne”.