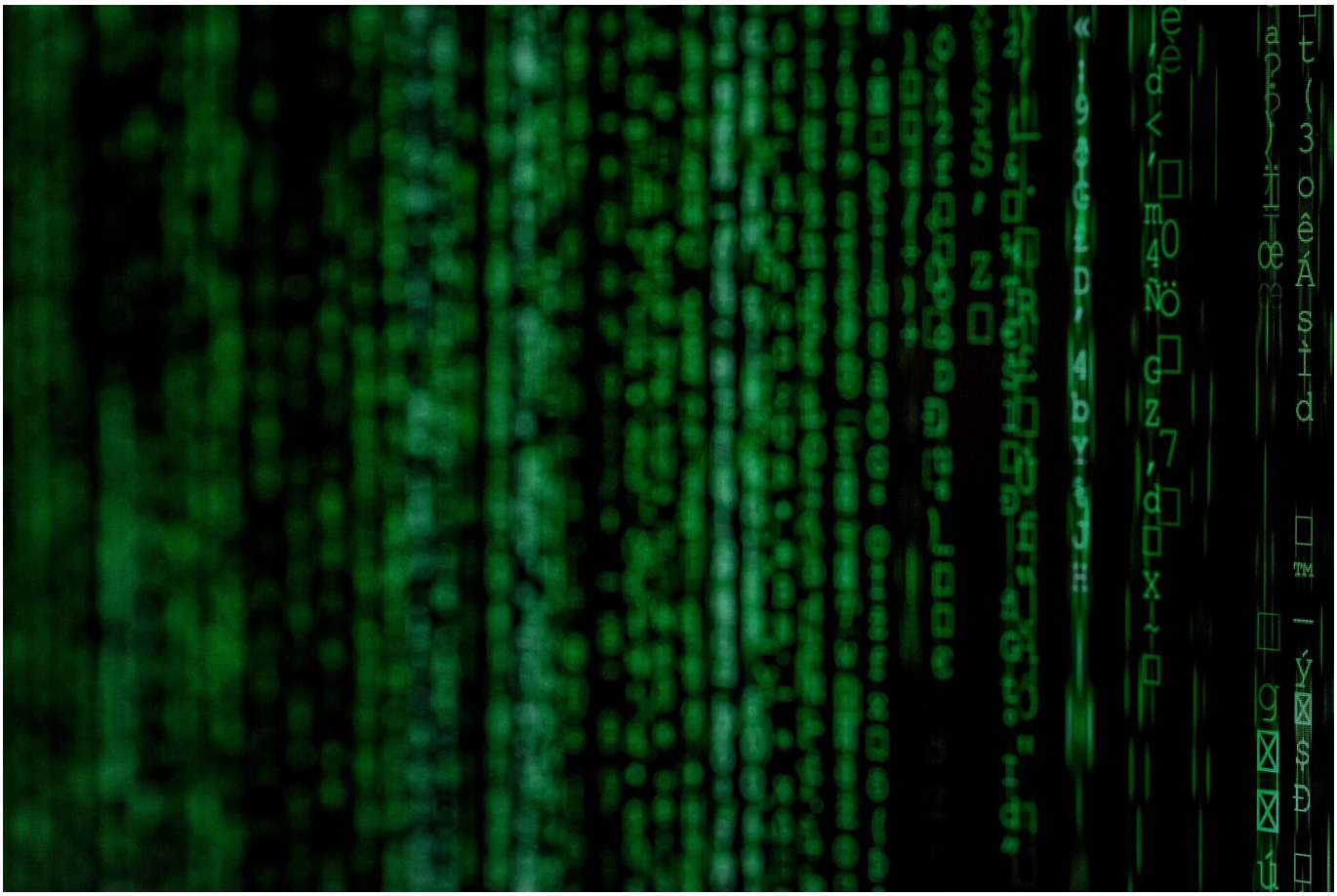
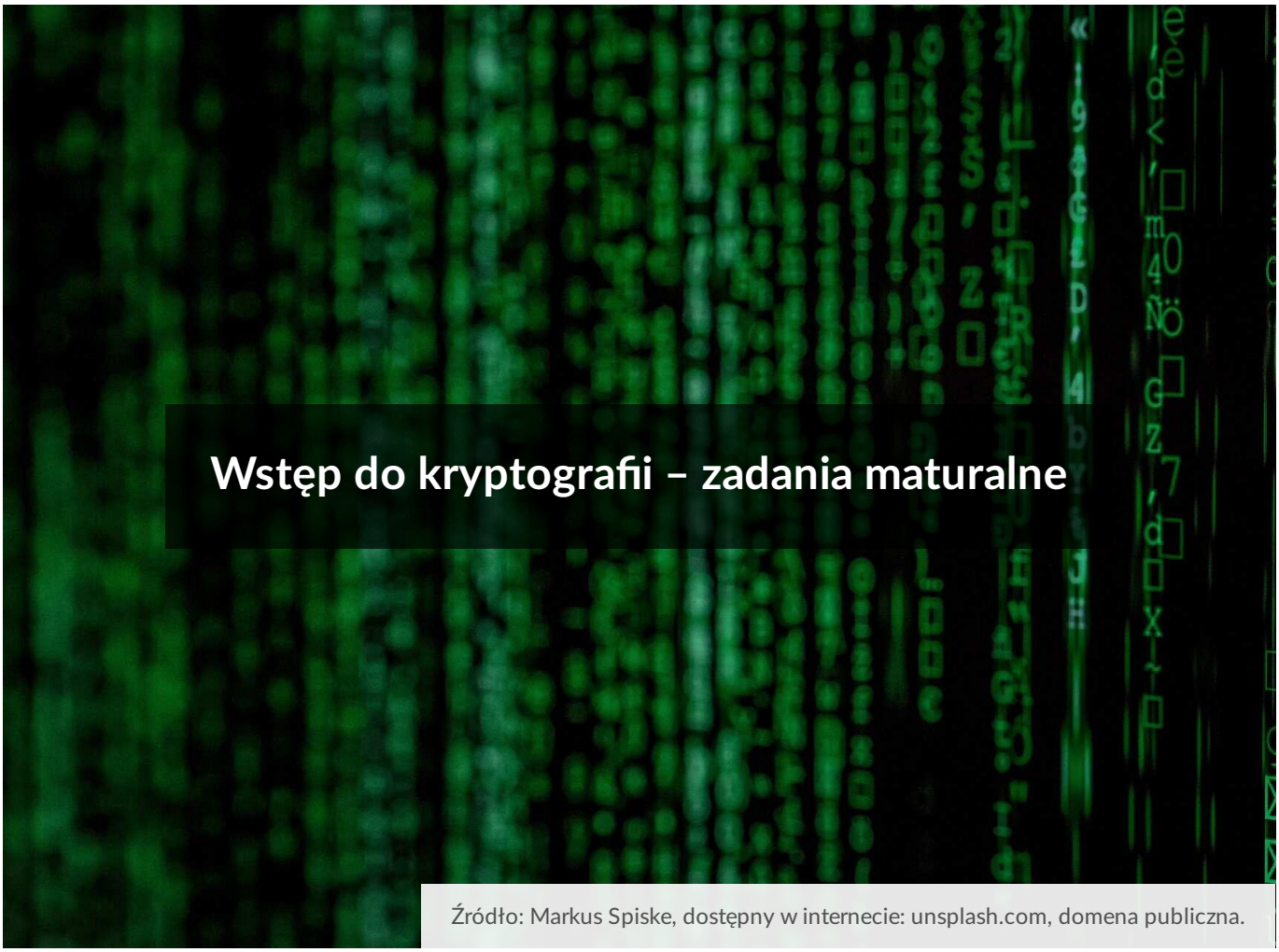


Wstęp do kryptografii – zadania maturalne



Źródło: Markus Spiske, dostępny w internecie: unsplash.com, domena publiczna.

Najlepszym sposobem na przekazanie tajnych informacji, tak aby nie mogły zostać odczytane przez osoby niepowołane, jest szyfrowanie ich. Jednym z prostszych algorytmów szyfrowania jest szyfr płotkowy. W tym e-materiale znajdziesz zadania typu maturalnego wykorzystujące ten algorytm. We Wstępie do kryptografii przedstawiliśmy najważniejsze informacje dotyczące tego zagadnienia.

Implementację zagadnienia w poszczególnych językach programowania znajdziesz w e-materiałach:

- Wstęp do kryptografii w języku C++,
- Wstęp do kryptografii w języku Java,
- Wstęp do kryptografii w języku Python.

Twoje cele

- Przeanalizujesz zadania wykorzystujące szyfr płotkowy.
- Przećwiczysz rozwiązywanie zadań typu maturalnego.
- Napiszesz program wykorzystujący szyfr płotkowy.

Przeczytaj

Zadanie 1

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 = 3`. Wynik zapisz do pliku `dane_6_1.txt`, każde słowo w osobnym wierszu, w porządku odpowiadającym kolejności słów z pliku z danymi.

Specyfikacja:

Dane:

- `n` – liczba słów w pliku
- `wyrazy[n]` – tablica zawierająca słowa znajdujące się w pliku
- `k` – klucz

Wynik:

- lista zaszyfrowanych słów

Poprawny wynik dla podanego fragmentu pliku:

```
IRTNNEPEOAITRWE
REAOWSLNEZEI
KEWEOSROAINWN
```

> Plik z danymi został przygotowany przez CKE i pojawił się egzaminie maturalnym z informatyki w maju 2016 roku (poziom rozszerzony, część II). Cały arkusz wraz z danymi można znaleźć na stronie internetowej Centralnej Komisji Egzaminacyjnej.

Plik o rozmiarze 1.29 KB w języku polskim

Przykładowe rozwiązanie

Dla każdego słowa tworzymy nową pustą tablicę dwuwymiarową `plotek[][]` o wysokości równej wartości klucza i szerokości równej długości rozpatrywanego wyrazu.

Tworzymy dwie zmienne pomocnicze `dol` oraz `wiersz`. Będą one pomagały w nawigacji po tablicy `plotek[][]`, tak aby kolejne litery tekstu jawnego utworzyły zygzak.

Jeżeli zmienna `wiersz` osiągnie wartość równą `klucz - 1`, oznaczać to będzie, że jest to ostatni wiersz tabeli. W takiej sytuacji kolejne litery będą umieszczane na coraz wyższych poziomach.

W przypadku, gdy zmienna `wiersz` osiągnie wartość `0`, oznaczać to będzie, że jest to najwyższy wiersz tabeli. W takiej sytuacji kolejne litery będą umieszczane na coraz niższych poziomach.

Zmienna `dol` informuje, czy tekst będzie zapisywany do coraz niższych, czy coraz wyższych wierszy.

Na koniec zaszyfrowany tekst zostanie zapisany do zmiennej `szyfrogram` z pominięciem spacji i wyświetlony na ekranie.

```
n = 100
dla i = 0, 1, 2, ..., n - 1 wykonuj:
   slowo = wyrazy[i]

   dla j = 0, 1, 2, ..., klucz - 1 wykonuj:
       dla l = 0, 1, 2, ..., długość(słowo) - 1 wykonuj:
           plotek[j][l] = ' '

   wiersz = 0
   dol = 1

   dla j = 0, 1, 2, ..., długość(słowo) - 1 wykonuj:
       plotek[wiersz][j] = slowo[j]

       jeżeli wiersz == klucz - 1 wykonaj:
           dol = 0
       w przeciwnym razie jeżeli wiersz == 0 wykonaj:
           dol = 1

       jeżeli dol == 1 wykonaj:
           wiersz = wiersz - 1
       w przeciwnym razie wykonaj:
           wiersz = wiersz + 1
```

```
24
25      dla j = 0, 1, 2, ..., klucz - 1 wykonuj:
26          dla l = 0, 1, 2, ..., długość(słowo) - 1 wykonuj:
27              jeżeli plotek[j][l] != ' ' wykonaj:
28                  szyfrogram = szyfrogram + plotek[j][l]
29
30      wypisz(szyfrogram)
```

> **Polecenie 1**
>
> Rozwiąż zadanie samodzielnie w wybranym języku programowania.

Schemat punktowania

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.

Zadanie 2

Napisz program, który zaszyfruje łańcuch znaków `zdanie` szyfrem z kluczem o wartości `k`. Pamiętaj, aby wcześniej przygotować dane w taki sposób, aby wynikiem był szyfrogram złożony tylko i wyłącznie z wielkich liter alfabetu.

Przetestuj działanie programu dla łacińskiej sentencji `Finis coronat opus` i klucza o wartości 6.

Specyfikacja:

Dane:

- `zdanie` – łańcuch znaków zawierający zdanie do zaszyfrowania
- `k` – liczba naturalna; klucz

Wynik:

- `szyfrogram` – łańcuch znaków `zdanie` zaszyfrowany szyfrem z kluczem o wartości `k`

Poprawny wynik dla podanych danych:

```
FNIOANRTIOSSCOUP
```

Przykładowe rozwiązanie:

Przed przystąpieniem do szyfrowania odpowiednio przygotowujemy dane. Tworzymy pusty łańcuch znaków `jawny`, który będzie zawierał przekształcone dane wejściowe.

Jeżeli litera znajdująca się w `zdanie[i]` jest mała, to znaczy, że jej kod w tablicy ASCII należy do przedziału <97, 122> (litery: 'a' - 'z'). W takim wypadku odejmujemy od wartości znaku różnicę odległości kodu litery 'a' (97) oraz 'A' (65) i dodajemy do ciągu znaków `jawny`.

Jeżeli litera znajdująca się w `zdanie[i]` jest wielka, to znaczy, że jej kod w tablicy ASCII należy do przedziału <65, 90> (litery: 'A' - 'Z'). W takim wypadku dodajemy znak do ciągu znaków `jawny`.

Szyfrowanie wygląda identycznie jak w poprzednim zadaniu.

```
jawny = ""
dla i = 0, 1, 2, ..., długość(zdanie) - 1 wykonuj:
    jeżeli zdanie[i] >= 'a' ORAZ zdanie[i] <= 'z' wykonaj:
        jawny += zdanie[i] - ('a' - 'A')
    w przeciwnym razie jeżeli zdanie[i] >= 'A' ORAZ zdanie[i] <='
        jawny += zdanie[i]

dla i = 0, 1, 2, ..., klucz - 1 wykonuj:
    dla j = 0, 1, 2, ..., długość(jawny) - 1 wykonuj:
        plotek[i][j] = ' '

wiersz = 0
dol = 1

dla i = 0, 1, 2, ..., długość(jawny) - 1 wykonuj:
    plotek[wiersz][i] = jawny[i]

    jeżeli wiersz == klucz - 1 wykonaj:
        dol = 0
    w przeciwnym razie wykonaj:
        dol = 1

    jeżeli dol == 1 wykonaj:
        wiersz = wiersz - 1
    w przeciwnym razie wykonaj:
```

```
        wiersz = wiersz + 1

szyfrogram = ""
dla i = 0, 1, 2, ..., klucz-1 wykonuj:
    dla j = 0, 1, 2, ..., długość(jawny) - 1 wykonuj:
        jeżeli plotek[i][j] != ' ' wykonaj:
            szyfrogram = szyfrogram + plotek[i][j]

wypisz(szyfrogram)
```

> **Polecenie 2**
>
> Rozwiąż zadanie samodzielnie w wybranym języku programowania.

Słownik

> **iteracja**
>
> technika programowania, która polega na powtarzaniu tej samej operacji określoną liczbę razy lub do momentu, w którym zadany warunek zostanie spełniony
> **klucz**
>
> informacja, która jest wykorzystywana do szyfrowania i/lub deszyfrowania wiadomości
> **klucz prywatny**
>
> tajny klucz wykorzystywany w procesie deszyfrowania w szyfrach asymetrycznych; powinien być znany jedynie adresatowi zaszyfrowanej wiadomości
> **klucz publiczny**
>
> udostępniony publicznie klucz wykorzystywany w procesie szyfrowania w szyfrach asymetrycznych
> **kryptografia**
>
> gałąź wiedzy o zapisywaniu informacji w sposób utrudniający, bądź całkowicie uniemożliwiający jej odczytanie
> **string**
>
> klasa języka C++, której obiekty służą do przechowywania ciągów znaków, takich jak całe zdania (np. „Ala ma kota”) albo pojedyncze wyrazy (np. „kot”)
> **szyfrogram**
>
> zaszyfrowana wiadomość
> **szyfrowanie**

przekształcanie tekstu jawnego w szyfrogram

tablica ASCII

7-bitowy system kodowania znaków, w którym każdy z obsługiwanych symboli jest reprezentowany przez liczbę; 7 bitów umożliwia przechowanie informacji o znakach o kodach z zakresu 0-127; używany m.in. we współczesnych komputerach oraz sieciach komputerowych

zmienna logiczna

zmienna, która przyjmuje wartości 1 (`true` – prawda) lub 0 (`false` – fałsz)

Prezentacja multimedialna

Zadanie 3

W poniższym pliku znajduje się 130 wierszy zawierających dane zaszyfrowane szyfrem płotkowym oraz odpowiadające im klucze szyfrujące.

Przykładowe dane:

```
AAGKNZISE 5
AANJAZSAT 5
ACIETRRHTKUA 2
```

Napisz program, który odszyfruje słowa zaszyfrowane podanymi kluczami.

Specyfikacja:

Dane:

- `n` – liczba szyfrogramów
- `tab[n][2]` – tablica dwuwymiarowa zawierająca `n` wierszy danych, gdzie pierwsza kolumna to zaszyfrowany wyraz, a druga to odpowiadający mu klucz

Wynik:

- lista odszyfrowanych słów

Poprawny wynik dla podanego fragmentu pliku:

```
AGNIESZKA
ANASTAZJA
ARCHITEKTURA
```

Plik o rozmiarze 1.77 KB w języku polskim

W załączniku znajdują się rzeczywiste dane. Spróbuj rozwiązać zadanie samodzielnie w wybranym języku programowania.

1

Materiał audio dostępny pod adresem:

https://zpe.gov.pl/b/PkJnQ2ONe

Naszym zadaniem będzie odszyfrowanie tajnych wiadomości zapisanych w tablicy dwuwymiarowej `tab[][]`. Każdy wiersz tablicy zawiera dwie dane: szyfrogram i klucz. Każdorazowo przypisujemy te wartości do odpowiednich zmiennych: `szyfrogram` i `klucz`.

2

Materiał audio dostępny pod adresem:

https://zpe.gov.pl/b/PkJnQ2ONe

Dla każdego zaszyfrowanego słowa z tablicy `tab[][]` tworzymy wypełnioną spacjami tablicę `plotek[][]` o szerokości równej długości rozpatrywanego wyrazu i wysokości równej wartości klucza.

3

4

Materiał audio dostępny pod adresem:

https://zpe.gov.pl/b/PkJnQ2ONe

Za pomocą zmiennych pomocniczych `wiersz` i `dol` w tablicy `plotek[][]` zostają umieszczone znaki '*' w formie zygzaka.

5

Materiał audio dostępny pod adresem:

https://zpe.gov.pl/b/PkJnQ2ONe

Gdy zmienna `dol` jest równa 1, zmienna `wiersz` zwiększa się o 1, czyli kolejne '*' wpisujemy ku dołowi tablicy. Dzieje się tak, dopóki zmienna wiersz nie osiągnie wartości `klucz - 1`, czyli dopóki nie dotrze do ostatniego wiersza tablicy. Jeżeli taka sytuacja wystąpi, zmiennej `dol` zostaje przypisana wartość 0, zaś zmienna `wiersz` będzie zmniejszała się o 1, czyli kolejne '*' wpisujemy ku górze tablicy.

6

7

Materiał audio dostępny pod adresem:

https://zpe.gov.pl/b/PkJnQ2ONe

Następnym krokiem będzie umieszczenie w miejscu '*' kolejnych liter rozpatrywanej zaszyfrowanej wiadomości. Robimy to za pomocą zmiennej `pomoc`, która zawiera kolejne adresy liter szyfrogramu i zapisuje je do tablicy `plotek[][]` we właściwych miejscach.

8

9

Materiał audio dostępny pod adresem:

https://zpe.gov.pl/b/PkJnQ2ONe

Za pomocą pętli (podobnej jak w przypadku, gdy do tablicy `plotek[ ][ ]` wpisywaliśmy '*' we właściwych miejscach) do zmiennej `jawny` dodajemy kolejne litery odszyfrowanej wiadomości.

10

11

Materiał audio dostępny pod adresem:

https://zpe.gov.pl/b/PkJnQ2ONe

Na koniec wewnętrznej pętli `dla` wypisujemy kolejne odszyfrowane słowa.

Sprawdź się

Zadanie 4

W pliku znajduje się 130 par słów oraz kluczy. Pierwsze słowo to tekst jawny, drugie jest szyfrogramem uzyskanym przy użyciu szyfru płotkowego. Sprawdź, czy wiadomość została zaszyfrowana za pomocą klucza znajdującego się w pliku. Policz, ile jest par takich słów, które nie zostały zaszyfrowane za pomocą klucza danego w pliku.

Przykładowe dane:

```
INTERPRETOWANIE ITRRTWNENEPEOAI 2
ROZWESELANIE REOSLEZEAIWN 4
KONSERWOWANIE KEWEOSROAINWN 6
```

Specyfikacja:

Dane:

- `wyrazy[][]` – tablica dwuwymiarowa zawierająca kolejno tekst jawny i szyfrogram
- `klucze` – tablica kluczy

Wynik:

- `wynik` – liczba naturalna; liczba słów, które nie zostały zaszyfrowane za pomocą klucza danego w pliku

Poprawny wynik dla podanych danych:

1

Poprawny wynik dla całego pliku:

47

Plik o rozmiarze 2.69 KB w języku polskim

> Rozwiąż zadanie lokalnie na swoim komputerze.

Pokaż ćwiczenia:

Ćwiczenie 1

Rozwiązanie w języku C++.

Twoje zadania

1. Program wypisuje liczbę par słów, które nie zostały zaszyfrowane za pomocą klucza podanego w pliku.

Ćwiczenie 2

Rozwiązanie w języku Java.

Twoje zadania

1. Program wypisuje liczbę par słów, które nie zostały zaszyfrowane za pomocą klucza podanego w pliku.

Ćwiczenie 3

Rozwiązanie w języku Python.

Twoje zadania

1. Program wypisuje liczbę par słów, które nie zostały zaszyfrowane za pomocą klucza podanego w pliku.

Dla nauczyciela

Autor: Maurycy Gast
Przedmiot: Informatyka

Temat: Wstęp do kryptografii – 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.

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.

V. Przestrzeganie prawa i zasad bezpieczeństwa. Respektowanie prywatności informacji i ochrony danych, praw własności intelektualnej, etykiety w komunikacji i norm współżycia społecznego, ocena zagrożeń związanych z technologią i ich uwzględnienie dla bezpieczeństwa swojego i innych.

Treści nauczania – wymagania szczegółowe

I. Rozumienie, analizowanie i rozwiązywanie problemów.

Zakres podstawowy. Uczeń:

1) planuje kolejne kroki rozwiązywania problemu, z uwzględnieniem podstawowych etapów myślenia komputacyjnego (określenie problemu, definicja modeli i pojęć, znalezienie rozwiązania, zaprogramowanie i testowanie rozwiązania).

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,

4) porównuje działanie różnych algorytmów dla wybranego problemu, analizuje algorytmy na podstawie ich gotowych implementacji;

5) sprawdza poprawność działania algorytmów dla przykładowych danych.

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;

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;

V. Przestrzeganie prawa i zasad bezpieczeństwa.

Zakres podstawowy. Uczeń:

3) stosuje dobre praktyki w zakresie ochrony informacji wrażliwych (np. hasła, pin), danych i bezpieczeństwa systemu operacyjnego, objaśnia rolę szyfrowania informacji;

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 wykorzystujące szyfr płotkowy.
- Przećwiczysz rozwiązywanie zadań typu maturalnego.
- Napiszesz program wykorzystujący szyfr płotkowy.

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.** Uczniowie powtarzają informacje na temat kryptografii i szyfru płotkowego.

Faza wstępna:

1. Nauczyciel wyświetla temat i cele zajęć. Prosi uczniów, by na podstawie wiadomości zdobytych przed lekcją zaproponowali kryteria sukcesu.
2. **Rozpoznanie wiedzy uczniów.** Nauczyciel prosi wybranego ucznia lub uczniów o przedstawienie sytuacji problemowej związanej z tematem lekcji.

Faza realizacyjna:

1. **Praca z tekstem.** Nauczyciel wyświetla zawartość sekcji „Przeczytaj". Na forum klasy uczniowie analizują rozwiązanie Zadania 1. Następnie w parach implementują rozwiązanie w wybranym języku programowania. Nauczyciel sprawdza poprawność wykonania.

W kolejnym kroku uczniowie przechodzą do Zadania 2. Na forum klasy analizują rozwiązanie zapisane za pomocą pseudokodu.
2. **Praca z multimedium.** Nauczyciel wyświetla zawartość sekcji „Prezentacja multimedialna". Uczniowie wspólnie zapoznają się z przykładowym rozwiązaniem Zadania 3. Następnie pobierają załącznik, w którym znajdują się rzeczywiste dane i próbują indywidualnie rozwiązać zadanie w wybranym języku programowania.
3. **Ćwiczenia umiejętności.** Uczniowie indywidualnie wykonują Zadanie 4 z sekcji „Sprawdź się" w jednym z dostępnych języków programowania. Następnie porównują swoje rozwiązania z wynikiem pracy innej osoby programującej w tym samym języku.

Faza podsumowująca:

1. Na koniec zajęć nauczyciel raz jeszcze wyświetla na tablicy temat lekcji i cele zawarte w sekcji „Wprowadzenie". W odniesieniu do ich realizacji dokonuje szczegółowej oceny rozwiązania zastosowanego przez wybranego ucznia.
2. Na koniec zajęć z programowania w C++ nauczyciel prosi uczniów o rozwinięcie zdania: „Na dzisiejszych zajęciach nauczyłam/łem się jak...".

Praca domowa:

1. Uczniowie implementują rozwiązanie Zadania 2 z sekcji „Przeczytaj" w wybranym języku programowania.

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:

- Treści w sekcji „Przeczytaj" można wykorzystać jako podsumowanie i utrwalenie wiedzy uczniów.