Szyfr polialfabetyczny w języku C++

Źródło: Amador Loureiro, domena publiczna.

Poznaliśmy już szyfry oparte na jednym alfabecie szyfrującym, takie jak szyfr Cezara. Trudniejsze do złamania są te, które korzystają z większej liczby alfabetów, czyli szyfry polialfabetyczne. Przykładami takich szyfrów są szyfry Vigenère'a oraz Beauforta. W tym e-materiale dowiesz się, jak zaimplementować je w języku C++.

Jeśli ciekawi cię, jak wyglądają implementacje w innych językach programowania, możesz się z nimi zapoznać w pozostałych e-materiałach z tej serii:

- Szyfr polialfabetyczny w języku Java,
- Szyfr polialfabetyczny w języku Python.

Więcej zadań? Sięgnij do: Szyfr polialfabetyczny – zadania maturalne.

Twoje cele

- Przeanalizujesz implementacje algorytmów szyfru Vigenère'a i szyfru Beauforta w języku C++.
- Napiszesz program wykorzystujący szyfrowanie polialfabetyczne.
- Scharakteryzujesz algorytm szyfrowania za pomocą szyfrów Vigenère'a oraz Beauforta.

Przeczytaj

Problem 1

Specyfikacja problemu:

Napisz program szyfrujący dany komunikat za pomocą szyfru polialfabetycznego (wykorzystaj algorytm szyfru Vigenère'a).

Przetestuj działanie programu dla słowa jawnego OPADY oraz klucza szyfrującego LEJE.

Specyfikacja problemu:

Dane:

`wiadomosc` – słowo jawne; łańcuch znaków

`slowo_klucz` – klucz szyfrujący; łańcuch znaków

Wynik:

`szyfrogram` – słowo tajne; łańcuch znaków

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

int main () {

    // Tutaj dodaj kod. Żeby coś wypisać, użyj polecenia: cout
}
```

```
1
```

Polecenie 1

Porównaj swoje rozwiązanie z przedstawionym w filmie.

Film dostępny pod adresem /preview/resource/Ra28XZjKGIQrK

Film nawiązujący do treści materiału: Szyfr polialfabetyczny Vigenere'a.

Podsumowanie

Zaimplementujmy w języku C++ szyfr Vigenère'a. W pierwszym kroku dodajemy wybrane biblioteki.

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

using namespace std;
```

Kolejnym krokiem jest zdefiniowanie funkcji `generujKlucz()`, która ma za zadanie przygotowanie klucza o odpowiedniej długości, tj. równej długości tekstu szyfrowanego. Funkcja jako argumenty przyjmie słowo-klucz oraz długość szyfrowanego tekstu.

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

using namespace std;

string generujKlucz(string slowoKlucz, int dlugoscWiadomosci)
{
    int iterator = 0;

    while (slowoKlucz.length() < dlugoscWiadomosci)
    {
        slowoKlucz += slowoKlucz[iterator];
        iterator++;
    }

    return slowoKlucz;
}
```

Przedstawiona funkcja tworzy klucz, powielając słowo klucz tak długo, aż klucz osiągnie pożądaną długość. Następnie musimy zdefiniować funkcję szyfrującą. Jej argumentami będą: szyfrowana wiadomość oraz klucz – ten, który został utworzony funkcją `generujKlucz()`. Funkcja szyfrująca wykonuje pętle po wszystkich znakach szyfrowanej wiadomości. Dla każdego znaku (zarówno wiadomości, jak i klucza – mają one taką samą długość) określany jest jego indeks, czyli numer znaku. Otrzymujemy go poprzez odjęcie od kodu ASCII liczby 65 (czyli kodu litery A). Tym samym litera A ma indeks 0, B - 1, C - 2 itd.

Szyfrogram powstaje następująco: suma indeksów litery wiadomości i odpowiadającej jej litery klucza dzielona jest modulo 26. Reszta z dzielenia jest indeksem kolejnej litery szyfrogramu, a więc wystarczy do niego dodać znak o kodzie ASCII, będący tym właśnie indeksem, powiększonym o 65.

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

using namespace std;

string generujKlucz(string slowoKlucz, int dlugoscWiadomosci)
{
    int iterator = 0;

    while (slowoKlucz.length() < dlugoscWiadomosci)
    {
        slowoKlucz += slowoKlucz[iterator];
        iterator++;
    }

    return slowoKlucz;
}

string szyfruj(string wiadomosc, string klucz)
{
    string szyfrogram = "";

    for (int i = 0; i < wiadomosc.length(); i++)
    {
        int indeksLiteryWiadomosci = wiadomosc[i] - 65;
        int indeksLiteryKlucza = klucz[i] - 65;
        int indeksLiterySzyfrogramu = (indeksLiteryWiadomosci + i

        char kodAsciiLiterySzyfrogramu = (char)(indeksLiterySzyfr
        szyfrogram += kodAsciiLiterySzyfrogramu;
    }
    return szyfrogram;
}
```

Kolejnym etapem jest implementacja funkcji deszyfrującej. Będzie ona wyglądała bardzo podobnie. Jej argumentami są: zaszyfrowana wiadomość oraz klucz. Pojawia się tu jednak

dodatkowy problem. Podczas odszyfrowania (odwrotnie niż przy szyfrowaniu) będziemy odejmować indeks litery klucza od indeksu litery szyfrogramu. Wynik takiej operacji może być ujemny, co musimy uwzględnić przy dzieleniu modulo 26. Zdefiniujemy więc dodatkową funkcję `modulo26()`, która w przypadku ujemnego wyniku zwróci nam uzyskaną wartość zwiększoną o 26. W efekcie wynik będzie zawsze dodatni.

```
int modulo26(int a)
{
    int wynik = a % 26;
    return wynik < 0 ? wynik + 26 : wynik;
}

string deszyfruj(string szyfrogram, string klucz)
{
    string wiadomosc = "";

    for (int i = 0; i < szyfrogram.length(); i++)
    {
        int indeksLiterySzyfrogramu = szyfrogram[i] - 65;
        int indeksLiteryKlucza = klucz[i] - 65;
        int indeksLiteryWiadomosci = (indeksLiterySzyfrogramu - i

        char kodAsciiLiteryWiadomosci = (char)(indeksLiteryWiadom
        wiadomosc += kodAsciiLiteryWiadomosci;
    }
    return wiadomosc;
}
```

Teraz wystarczy już tylko w odpowiedni sposób wywołać funkcję dla przykładowej wiadomości oraz klucza. Funkcja `main()` będzie wyglądać następująco:

```
int main()
{
    string wiadomosc = "OPADY";
    string slowoKlucz = "LEJE";

    string klucz = generujKlucz(slowoKlucz, wiadomosc.length());
    string szyfrogram = szyfruj(wiadomosc, klucz);

    cout << szyfrogram << endl;
```

```
10
11      string odszyfrowanaWiadomosc = deszyfruj(szyfrogram, klucz);
12
13      cout << odszyfrowanaWiadomosc << endl;
14
15      return 0;
16  }
```

Gotowy program prezentuje się następująco:

```
1   #include <iostream>
2   #include <string>
3
4   using namespace std;
5
6   string generujKlucz(string slowoKlucz, int dlugoscWiadomosci)
7   {
8       int iterator = 0;
9
10      while (slowoKlucz.length() < dlugoscWiadomosci)
11      {
12          slowoKlucz += slowoKlucz[iterator];
13          iterator++;
14      }
15
16      return slowoKlucz;
17  }
18
19  string szyfruj(string wiadomosc, string klucz)
20  {
21      string szyfrogram = "";
22
23      for (int i = 0; i < wiadomosc.length(); i++)
24      {
25          int indeksLiteryWiadomosci = wiadomosc[i] - 65;
26          int indeksLiteryKlucza = klucz[i] - 65;
27          int indeksLiterySzyfrogramu = (indeksLiteryWiadomosci + i
28
29          char kodAsciiLiterySzyfrogramu = (char)(indeksLiterySzyfr
30          szyfrogram += kodAsciiLiterySzyfrogramu;
31      }
```

```
    return szyfrogram;
}

int modulo26(int a)
{
    int wynik = a % 26;
    return wynik < 0 ? wynik + 26 : wynik;
}

string deszyfruj(string szyfrogram, string klucz)
{
    string wiadomosc = "";

    for (int i = 0; i < szyfrogram.length(); i++)
    {
        int indeksLiterySzyfrogramu = szyfrogram[i] - 65;
        int indeksLiteryKlucza = klucz[i] - 65;
        int indeksLiteryWiadomosci = modulo26(indeksLiterySzyfrog

        char kodAsciiLiteryWiadomosci = (char)(indeksLiteryWiadom
        wiadomosc += kodAsciiLiteryWiadomosci;
    }
    return wiadomosc;
}


int main()
{
    string wiadomosc = "OPADY";
    string slowoKlucz = "LEJE";

    string klucz = generujKlucz(slowoKlucz, wiadomosc.length());
    string szyfrogram = szyfruj(wiadomosc, klucz);

    cout << szyfrogram << endl;

    string odszyfrowanaWiadomosc = deszyfruj(szyfrogram, klucz);

    cout << odszyfrowanaWiadomosc << endl;

    return 0;
}
```

Po uruchomieniu programu najpierw generowany jest klucz na podstawie słowa „LEJE”. Następnie wykonywana jest funkcja `szyfruj()` i powstaje szyfrogram, który zostaje wyświetlony na ekranie. W kolejnym etapie szyfrogram jest z powrotem odszyfrowywany za pomocą funkcji `deszyfruj()`, a powstała wiadomość jest wyświetlana na ekranie.
W przypadku naszych przykładowych słów powstały szyfrogram ma postać „ZTJHJ”. Po odszyfrowaniu uzyskujemy prawidłową postać odszyfrowaną: „OPADY”.

Słownik

alfabet szyfrowy

przekształcony alfabet służący do zaszyfrowywania tekstu jawnego
graficzny interfejs

(ang. Graphical User Interface); interfejs pozwalający komunikować się z programem za pomocą wyświetlanych okien
szyfrogram

zaszyfrowana wiadomość
szyfrowanie

przekształcenie tekstu jawnego w sposób uniemożliwiający odczytanie oryginalnej treści bez dokonania procesu deszyfrowania, do przeprowadzenia którego niezbędna jest znajomość klucza szyfrującego
tekst jawny

tekst przed zaszyfrowaniem lub po odszyfrowaniu; jest to zwyczajny, niezaszyfrowany tekst, który można swobodnie przeczytać

Prezentacja multimedialna

Polecenie 1

Przeanalizuj prezentację przedstawiającą proces tworzenia programu realizującego szyfrowanie metodą Beauforta. Następnie porównaj go z omawianym wcześniej szyfrem Vigenère'a.

1

Materiał audio dostępny pod adresem:

https://zpe.gov.pl/b/PY7WUu8nD

Szyfr Vigenère'a to niejedyny szyfr polialfabetyczny. Innym przykładem może być metoda opracowana przez Francisa Beauforta. Nazwisko Beaufort kojarzymy głównie ze skali opisującej siłę wiatru. Pasją badacza nie była wyłącznie meteorologia. Szyfr przez niego stworzony opiera się praktycznie na tych samych założeniach, jak ten opracowany przez Vigenère'a. Zaimplementujemy go jednak w nieco inny sposób, za pomocą tablicy alfabetów. W podobny sposób można również zaimplementować szyfr Vigenère'a.

2

Materiał audio dostępny pod adresem:

https://zpe.gov.pl/b/PY7WUu8nD

W pierwszym kroku dodajemy wybrane biblioteki oraz tworzymy jednowymiarową tablicę `alfabet` i dwuwymiarową tablicę `tabela`. Stworzymy też funkcję `wypelnij`, której zadaniem będzie wypełnienie zadeklarowanych tablic.

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

char alfabet[26];
char tabela[26][26];

void wypelnij() {

}

```

3

Materiał audio dostępny pod adresem:

https://zpe.gov.pl/b/PY7WUu8nD

Tworzymy pętlę wypełniającą jednowymiarową tablicę `alfabet` kolejnymi literami. Przedstawione rozwiązanie wymagać będzie od nas znajomości tabeli ASCII. Zmienna `i`, zawarta w warunku pętli `for`, zainicjowana zostanie z wartością 97. Jest to odpowiednik pozycji litery „A” we wspomnianej tabeli. Powtórzenia wykonywać będziemy aż do osiągnięcia wartości równej pozycji ostatniej litery alfabetu „Z” (122). Teraz, gdy mamy już odpowiedni przedział wartości typu `int`, wystarczy posłużyć się operacją rzutowania.

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

char alfabet[26];
char tabela[26][26];

void wypelnij() {

    int z = 0;
    // WYPELNIENIE TABLICY LITERAMI ALFABETU
    for(int i = 97; i < 123; i++) {
```

```
        alfabet[z] =
(char)i;
        z++;
    }

}

```

4

Materiał audio dostępny pod adresem:

https://zpe.gov.pl/b/PY7WUu8nD

Teraz pozostaje nam wypełnienie dwuwymiarowej tablicy `tabela`. Znowu posłużymy się pętlami `for`. Rozpoczynać będziemy od wypełnienia pojedynczego wiersza naszej tablicy. Po ukończeniu tej operacji musimy jeszcze zamienić każdą literę naszego alfabetu na literę po niej następującą, realizując w ten sposób niezbędne przesunięcie. Po jego zakończeniu wypełniamy kolejny wiersz zmodyfikowanym już alfabetem. Opisane operacje wykonujemy do momentu zapełnienia całej tabeli.

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

char alfabet[26];
char tabela[26][26];

void wypelnij() {

    int z = 0;
    // WYPELNIENIE TABLICY
LITERAMI ALFABETU
    for(int i = 97; i <
123; i++) {
        alfabet[z] =
(char)i;
        z++;
```

```
    }

    // WYPELNIENIE TABLICY ALFABETÓW
    for(int z = 0; z < 26; z++) {
        for (int x = 0; x < 26; x++) {
            tabela[z][x] = alfabet[x];
        }

        for (int b = 0; b < 26 - 1; ++b) {
            int zamienianaLiczba = alfabet[b];
            alfabet[b] = alfabet[b+1];
            alfabet[b+1] = zamienianaLiczba;
        }

    }

}

```

Materiał audio dostępny pod adresem:

https://zpe.gov.pl/b/PY7WUu8nD

5

Wciąż pozostaje nam sam zabieg szyfrowania. Stworzymy osobną funkcję `szyfruj()`, w której argumentami będą wiadomość oraz klucz.

```
string szyfruj(string wiadomosc, string klucz) {

}
```

4

6

Materiał audio dostępny pod adresem:

https://zpe.gov.pl/b/PY7WUu8nD

Kolejnym krokiem jest wyznaczenie liter szyfrogramu. W szyfrze Beauforta, jeżeli chcemy zaszyfrować dany znak tekstu jawnego, wyszukujemy w tablicy alfabetów kolumnę przez niego wyznaczaną, a następnie znajdujemy w niej odpowiadającą literę klucza.

Wyznacza ona wiersz, na którego pierwszej pozycji (w przypadku tablicy chodzi o pozycję 0) znajduje się zaszyfrowany znak. W ten sposób postępujemy dla każdej kolejnej litery szyfrowanej wiadomości.

Musimy się zastanowić, jak opisany algorytm zaimplementować w praktyce. Wiemy, że nasz znak szyfrogramu znajduje się w zerowej kolumnie, jednak określenie numeru wiersza będzie bardziej skomplikowane. Skorzystamy z operacji matematycznej – różnice pomiędzy pozycją litery klucza a pozycją litery wiadomości dodamy do liczby liter w alfabecie, a następnie całą sumę poddamy operacji modulo. Od naszych liczb odjąć musimy 97, ponieważ poszukujemy wartości w tablicy dwuwymiarowej, a nie – jak poprzednio – w tabeli ASCII. W ten sposób uzyskamy dokładny numer wiersza.

```
1 string szyfruj(string
wiadomosc, string klucz) {
2     string zaszyfrowane =
"";
3
4     for (int i = 0, j = 0;
i < wiadomosc.size(); i++)
{
```

```
        zaszyfrowane += tabela[0][(int)((26+((klucz[j]-97)-(wiadomosc[i]-97)))%26)];

    return zaszyfrowane;
}

```

Materiał audio dostępny pod adresem:

https://zpe.gov.pl/b/PY7WUu8nD

7

Dodatkowo zaimplementować musimy operację uniemożliwiającą sytuację, w której klucz jest za krótki. Gdy dojdziemy do momentu, że podczas szyfrowania litery w kluczu się skończyły, wówczas pod uwagę brana jest ponownie pierwsza litera, a następnie te następujące po niej.

```
string szyfruj(string wiadomosc, string klucz) {
    string zaszyfrowane = "";

    for (int i = 0, j = 0; i < wiadomosc.size(); i++) {
        zaszyfrowane += tabela[0][(int)((26+((klucz[j]-97)-(wiadomosc[i]-97)))%26)];

        if (j == klucz.size() - 1) {
            j = 0;
        } else {
            j++;
        }
    }
```

```
13
14      return zaszyfrowane;
15  }
16
```

8

Materiał audio dostępny pod adresem:

https://zpe.gov.pl/b/PY7WUu8nD

Pozostaje nam już tylko wywołać funkcje w funkcji `main`. Szyfrowanie przeprowadzimy dla wiadomości „WIATR” i klucza „SKALA”.

Wynikiem przeprowadzonej operacji będzie szyfrogram „WCASJ”.

```
1  int main() {
2
3      wypelnij();
4      cout <<
   szyfruj("WIATR", "SKALA");
5
6
7      return 0;
8  }
9
```

Materiał audio dostępny pod adresem:

https://zpe.gov.pl/b/PY7WUu8nD

9

Przygotowany program prezentuje się następująco:

```
1  #include <iostream>
2  using namespace std;
3
4  char alfabet[26];
```

```
5  char tabela[26][26];
6
7  void wypelnij() {
8
9      int z = 0;
10     // WYPELNIENIE TABLICY
   LITERAMI ALFABETU
11     for(int i = 97; i <
   123; i++) {
12         alfabet[z] =
   (char)i;
13         z++;
14     }
15
16     // WYPELNIENIE TABLICY
   ALFABETÓW
17     for(int z = 0; z < 26;
   z++) {
18         for (int x = 0; x
   < 26; x++) {
19             tabela[z][x] =
   alfabet[x];
20         }
21
22         for (int b = 0; b
   < 26 - 1; ++b) {
23             int
   zamienianaLiczba =
   alfabet[b];
24             alfabet[b] =
   alfabet[b+1];
25             alfabet[b+1] =
   zamienianaLiczba;
26         }
27
28     }
29
30 }
31
32 string szyfruj(string
   wiadomosc, string klucz) {
33     string zaszyfrowane =
   "";
34
```

```
35      for (int i = 0, j = 0;
i < wiadomosc.size(); i++)
{
36          zaszyfrowane +=
tabela[0][(int)((26+
((klucz[j]-97)-
(wiadomosc[i]-97)))%26)];
37
38          if (j ==
klucz.size() - 1) {
39              j = 0;
40          } else {
41              j++;
42          }
43      }
44
45      return zaszyfrowane;
46  }
47
48  int main() {
49
50      wypelnij();
51      cout <<
szyfruj("WIATR", "SKALA");
52
53
54      return 0;
55  }
56
```

10

Materiał audio dostępny pod adresem:

https://zpe.gov.pl/b/PY7WUu8nD

W ten sposób zaimplementowaliśmy szyfr Beauforta w języku C++. Jak widzimy, jest on zbliżony do algorytmu opracowanego przez Vigenère'a, chociaż zaimplementowaliśmy go nieco inną metodą. W praktyce oba szyfry można zrealizować na oba opisane sposoby. W ten sposób możemy szybko wywnioskować,

iż opierając się na tych samych narzędziach i nieco modyfikując metodę, otrzymamy całkowicie inny wynik szyfrowania. Może to zmobilizować cię do samodzielnego modyfikowania znanych algorytmów szyfrujących, a w dalszej perspektywie - do opracowania własnego szyfru.

Polecenie 2

Opracuj notatkę podsumowującą najważniejsze informacje przedstawione w prezentacji.

Sprawdź się

Pokaż ćwiczenia: 

Ćwiczenie 1

Znasz już oryginalną wersję szyfru Vigenère'a – dotyczy ona określonych zasad przy tworzeniu klucza szyfrującego. Na podstawie przedstawionego kodu napisz funkcję, która przyjmie jeden parametr – łańcuch znaków do zaszyfrowania, a następnie utworzy odpowiedni klucz (pierwszy znak klucza ma być równy „J") i go zwróci. Przetestuj działanie programu dla łańcucha znaków do zaszyfrowania „tajnytekst".

Specyfikacja problemu:

Dane:

- `doZaszyfrowania` – łańcuch znaków

Wynik:

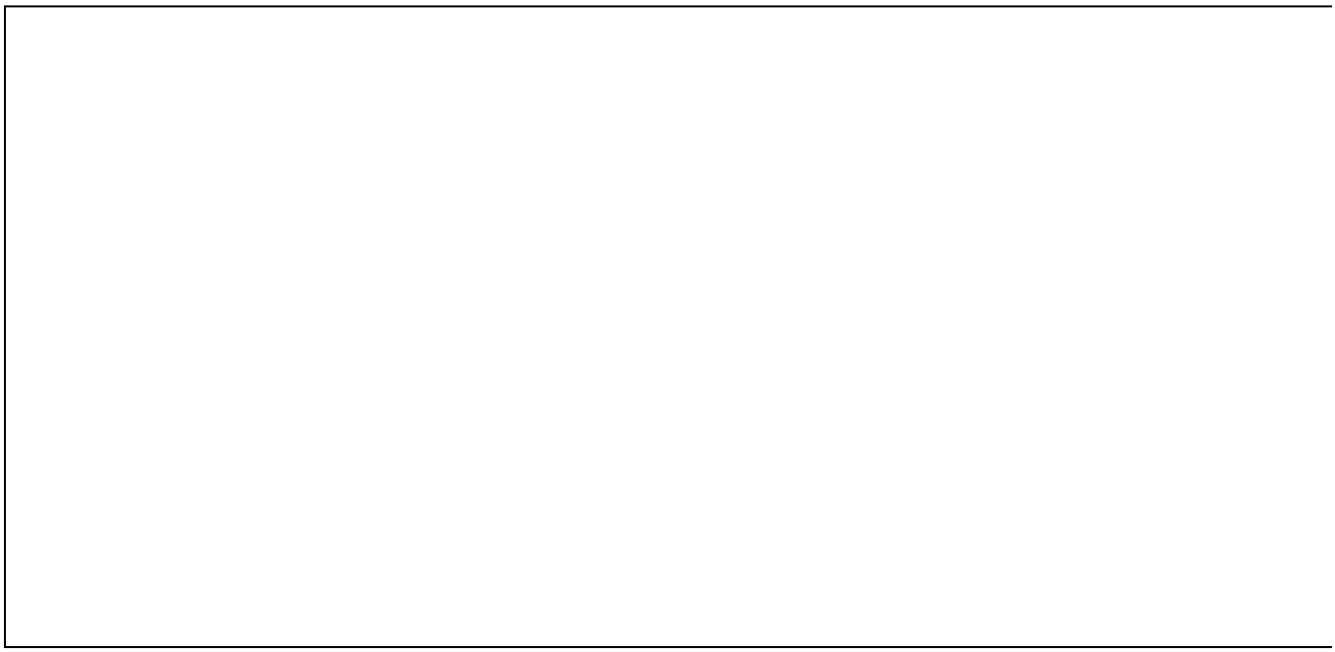
- `klucz` – łańcuch znaków

Twoje zadania

1. Program ma utworzyć klucz szyfrujący (według oryginalnej wersji szyfru Vigenère'a) dla zadanego ciągu znaków „tajnytekst", a następnie wypisać go na ekran konsoli. Za pierwszy znak klucza szyfrującego należy przyjąć znak „J".

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

// twój kod
int main() {

}
```

```
1
```



Ćwiczenie 2

W tym zadaniu musimy rozszerzyć program z **Ćwiczenia 1** o generowanie tablicy szyfrującej oraz wykonanie szyfrowania. Klucz ma być tworzony tak, jak w poprzednim zadaniu (według oryginalnej wersji szyfru Vigenère'a).

Specyfikacja problemu:

Dane:

- `doZaszyfrowania` – łańcuch znaków
- `klucz` – łańcuch znaków

Wynik:

- Program wyświetla utworzoną tablicę szyfrującą, a w nowej linii wynik szyfrowania.

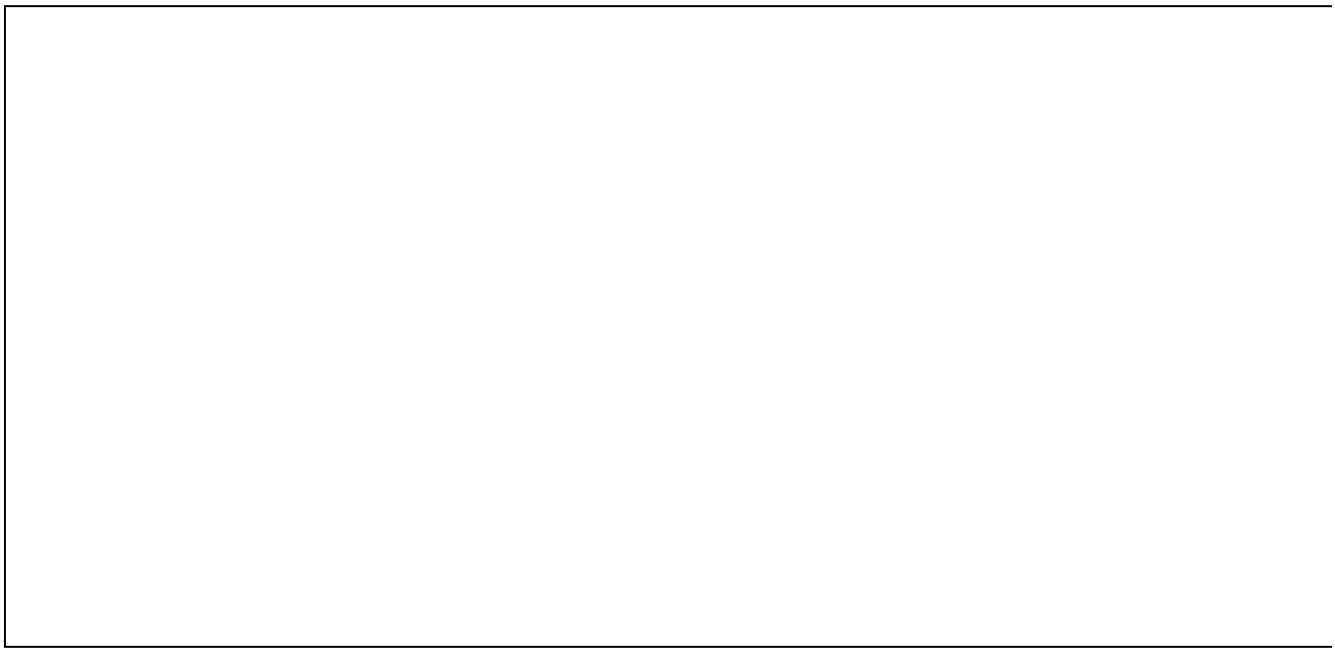
Twoje zadania

1. Program ma utworzyć tablicę szyfrowania (tablica 26 x 26 zawierająca znaki alfabetu łacińskiego, jak w poprzednich sekcjach e-materiału) oraz zaszyfrować (według oryginalnej wersji szyfru Vigenère'a) ciąg znaków „tajnytekst”. Za pierwszą literę klucza szyfrującego należy przyjąć literę „J”. **Należy wyświetlić utworzoną tablicę, a następnie w nowej linii wynik szyfrowania.**

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

char alfabet[26];
char tabela[26][26];

void wypelnijTablice() {

}

string szyfruj(string wiadomosc, string klucz) {

}
```

```
14
```

```
1
```



Dla nauczyciela

Autor: Maurycy Gast
Przedmiot: Informatyka

Temat: Szyfr polialfabetyczny w języku C++

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:

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,

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;

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

2) omawia znaczenie algorytmów szyfrowania i składania podpisu elektronicznego.

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 implementacje algorytmów szyfru Vigenère'a i szyfru Beauforta w języku C++.
- Napiszesz program wykorzystujący szyfrowanie polialfabetyczne.
- Scharakteryzujesz algorytm szyfrowania za pomocą szyfrów Vigenère'a oraz Beauforta.

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.

Przebieg lekcji

Przed lekcją:

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

Faza wstępna:

1. Nauczyciel wyświetla temat i cele zajęć zawarte w sekcji „Wprowadzenie". 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 ocenia, na podstawie informacji na platformie, stan przygotowania uczniów do zajęć. Jeżeli jest ono niewystarczające, prosi wybraną osobę o przedstawienie najważniejszych informacji z sekcji „Przeczytaj".
   Następnie uczniowie w parach analizują Problem 1 z sekcji „Przeczytaj" i piszą program, swoje rozwiązania porównują z zaprezentowanym na filmie.
2. **Praca z multimedium.** Nauczyciel wyświetla zawartość sekcji „Prezentacja multimedialna". Uczniowie wspólnie analizują prezentację przedstawiającą proces tworzenia programu realizującego szyfrowanie metodą Beauforta, a następnie porównują go z omawianym wcześniej szyfrem Vigenère'a.
3. **Ćwiczenie umiejętności.** Uczniowie wykonują indywidualnie ćwiczenie nr 1 z sekcji „Sprawdź się", a następnie porównują swoje odpowiedzi z kolegą lub koleżanką.
4. W kolejnym etapie uczniowie dobierają się w pary i wykonują ćwiczenie nr 2 z sekcji „Sprawdź się". Następnie konsultują swoje rozwiązania z inną parą uczniów i ustalają jedną wersję odpowiedzi.

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. Wybrany uczeń podsumowuje zajęcia z programowania w C++, zwracając uwagę na nabyte umiejętności.

Praca domowa:

1. Uczniowie proponują alternatywny sposób rozwiązania problemu postawionego w sekcji „Prezentacja multimedialna”.

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.

Wskazówki metodyczne:

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