Szyfrowanie i deszyfrowanie w języku C++

Szyfrowanie i deszyfrowanie w języku C++

Źródło: Tomasz_Mikolajczyk, dostępny w internecie: pixabay.com, domena publiczna.

Poznaliśmy już kilka algorytmów szyfrujących. Rozumiemy, jak działają i potrafimy dokonać ich implementacji w różnych językach programowania.

W tym e-materiale wykorzystamy zdobytą wiedzę do praktycznego opracowania projektu. Jak zaplanować pracę nad nim, dowiedzieliśmy się w e-materiale Szyfrowanie i deszyfrowanie – projekt. Celem zaimplementowanego programu będzie szyfrowanie oraz deszyfrowanie zadanych ciągów znaków za pomocą różnych algorytmów szyfrowania. Zostanie również stworzone odpowiednie menu wraz z interfejsem, aby ułatwić potencjalnemu użytkownikowi korzystanie z programu.

Ten e-materiał poświęcony jest implementacji w języku C++.

Ciekawi cię, jak wyglądają implementacje w innych językach programowania? Możesz się z nimi zapoznać w dwóch pozostałych lekcjach z tej serii:

- Szyfrowanie i deszyfrowanie w języku Java,
- Szyfrowanie i deszyfrowanie w języku Python.

Więcej zadań? Sięgnij do: Szyfrowanie i deszyfrowanie – zadania maturalne.

> **Twoje cele**

- Opracujesz program, który będzie realizował operacje szyfrowania i deszyfrowania.
- Powtórzysz poznane metody szyfrowania i deszyfrowania tekstu.
- Wykorzystasz zdobytą wiedzę na temat algorytmów szyfrujących do zaimplementowania praktycznego programu.

Film samouczek

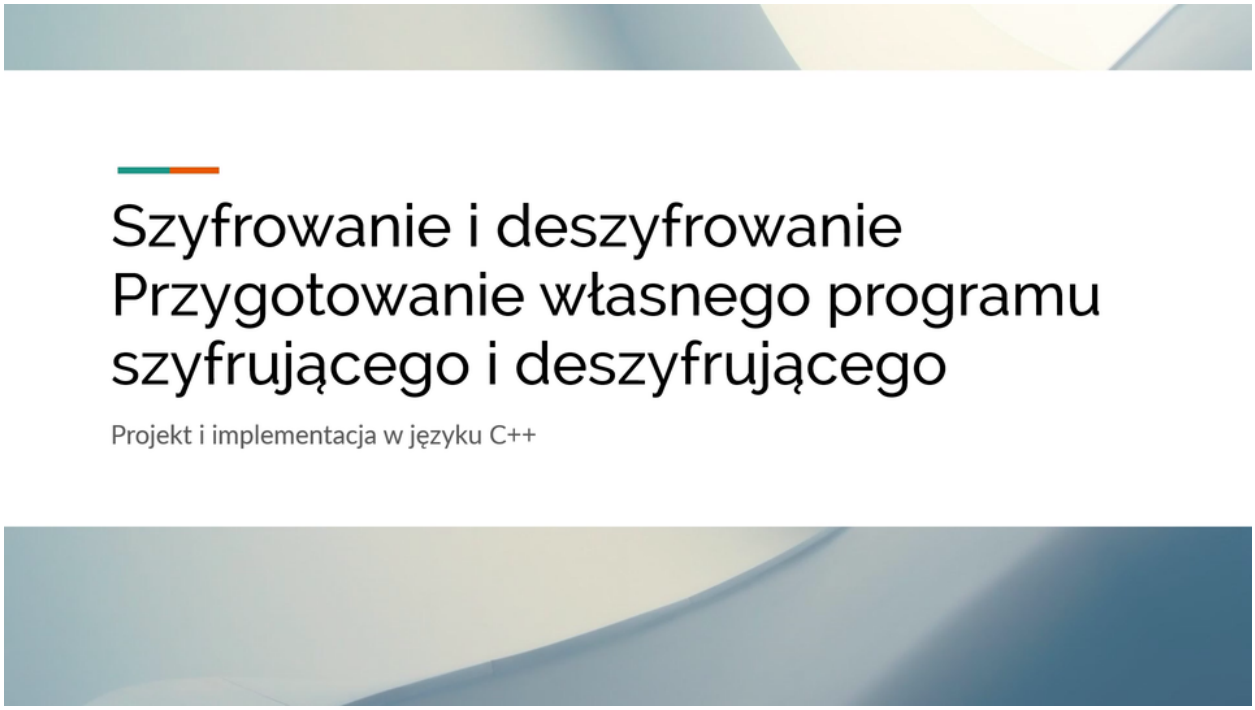
Problem 1

Specyfikacja problemu:

Przygotuj projekt aplikacji wykorzystującej szyfr Vigenère'a. Program powinien uruchamiać się, prezentując menu główne z trzema opcjami: szyfrowania, deszyfrowania i zakończenia pracy aplikacji. W przypadku wyboru opcji szyfrowania, bądź deszyfrowania program powinien pobrać treść wiadomości lub szyfrogramu oraz klucz. Wynik działania programu powinien zostać zapisany do pliku tekstowego.

Polecenie 1

Porównaj swoje rozwiązanie z tym zaprezentowanym w filmie.



Film dostępny pod adresem /preview/resource/RzlL703ckZZN0

Film nawiązujący do treści materiału: Przygotowanie programu szyfrującego i deszyfrującego.

Kod programu zaprezentowanego w filmie:

Plik o rozmiarze 3.57 KB w języku polskim

Przeczytaj

Projekt – co zamierzamy napisać?

Pierwszym krokiem, jaki musimy wykonać, jest ustalenie, do czego ma służyć nasz program oraz jak ma wyglądać jego interfejs. W tym wypadku stworzymy program, który najpierw pozwoli nam wybrać metodę, z której będziemy korzystać podczas naszych działań, a następnie zdecydować, czy chcemy szyfrować, czy też deszyfrować tekst.

Szyfr Cezara

Najważniejsze informacje dotyczące szyfrowania oraz deszyfrowania wiadomości za pomocą szyfru Cezara oraz jego implementacji w języku C++ znajdziesz w następujących e-materiałach:

- Szyfr Cezara,
- Uogólniony szyfr Cezara,
- Implementacja szyfru Cezara w języku C++,
- Implementacja uogólnionego szyfru Cezara w języku C++.

Szyfr Vigenère'a

Najważniejsze informacje dotyczące szyfrowania oraz deszyfrowania wiadomości za pomocą szyfru Vigenère'a oraz jego implementacji w języku C++ znajdziesz w następujących e-materiałach:

- Szyfr polialfabetyczny,
- Implementacja szyfru polialfabetycznego w języku C++.

Projekt – kod źródłowy

Aby kod źródłowy był jak najbardziej przejrzysty i czytelny, napiszemy go, dzieląc poszczególne etapy działania programu na osobne funkcje.

Krok 1

Na początku umieszczamy dobrze znany nam szkielet każdego programu pisanego w języku C++:

```
1 #include <iostream>
```

```
2
3 using namespace std;
4
5 int main()
6 {
7
8 }
```

Krok 2

Kolejnym krokiem będzie zapisanie nieskończonej pętli, by możliwe było wielokrotne szyfrowanie i deszyfrowanie przy pomocy naszego programu bez jego ponownego uruchamiania.

```
1  #include <iostream>
2
3  using namespace std;
4
5  int main()
6  {
7
8      while (true)
9      {
10
11     }
12
13 }
```

Krok 3

Następnie zadeklarujemy zmienne, w których będziemy przechowywali informację o wybranej opcji w naszym menu. Nasz program będzie składał się z dwóch menu (jedno z możliwością rozbudowy o dodatkowe metody szyfrowania), zatem do przechowywania wyboru, jakiego dokona użytkownik, zadeklarujemy zmienną typu `int`. Natomiast druga zmienna może mieć tylko dwie wartości, ponieważ możemy albo szyfrować, albo deszyfrować, więc będzie to zmienna typu logicznego `bool`.

```
1 #include <iostream>
2
3 using namespace std;
```

```
4
5  int main()
6  {
7      int MetodaSzyfrowania;
8      bool CzySzyfrujemy;
9
10     while (true)
11     {
12
13     }
14
15 }
```

Krok 4

Stwórzmy teraz funkcję odpowiedzialną za wyświetlenie menu oraz wybór metody szyfrowania/deszyfrowania.

```
1  int StworzMenu()
2  {
3      int menu;
4      do {
5          cout << "Wybierz metode szyfrowania: " << endl << endl;
6          cout << "1. Szyfr Cezara" << endl;
7          cout << "2. Szyfr Vigenere'a" << endl;
8          cout << "3. Wyjscie" << endl;
9          cin >> menu;
10
11         if (menu == 3)
12             exit(0);
13
14         system("cls");
15     } while (menu != 1 && menu != 2);
16
17     return menu;
18 }
```

- Linia pierwsza naszej funkcji – nagłówek. W omawianym wypadku jest to funkcja typu `int`, ponieważ będziemy zwracali numer opcji wybranej przez użytkownika.

- Następnie deklarujemy zmienną `menu`, w której będzie przechowywane to, co wybrał użytkownik.
- Tworzymy pętlę `do..while()` z warunkiem `menu != 1 && menu != 2`, by pętla wykonywała się, dopóki nie zostanie wybrana odpowiednia opcja z menu.
- Wypisujemy możliwe opcje w menu oraz odpowiadające im liczby, jakie ma wprowadzić użytkownik.
- Wczytujemy wybór użytkownika do zmiennej `menu`.
- Ponieważ trzecia opcja w naszym menu to wyjście, musimy sprawdzić, czy użytkownik jej nie wybrał.
- Jeśli tak, zamykamy program poleceniem `exit(0)`
- W przeciwnym wypadku używamy polecenia `system("cls")`, by wyczyścić ekran konsoli z poprzednio wyświetlonego menu. Polecenie `system` pochodzi z biblioteki `Windows.h`, którą musimy dołączyć poleceniem:

```
#include <Windows.h>
```

- Po wyjściu z pętli mamy pewność, że liczba, którą podał użytkownik, jest przypisana do jednej z opcji w menu, zatem możemy ją zwrócić w miejsce wywołania.

Krok 5

Stwórzmy funkcję odpowiedzialną za wyświetlanie menu oraz wybór, czy użytkownik chce szyfrować, czy deszyfrować wiadomość.

```
bool SzyfrDeszyfr()
{
    int szyfr;
    do {
        cout << "1. Szyfrowanie" << endl;
        cout << "2. Deszyfrowanie" << endl;
        cin >> szyfr;

        system("cls");
    } while (szyfr != 1 && szyfr != 2);

    if (szyfr == 1)
        return true;
    else
        return false;
}
```

- Linia pierwsza funkcji - nagłówek. Nasza funkcja jest typu `bool`, ponieważ będziemy zwracali to, co wybrał użytkownik, a ma on do wyboru tylko dwie opcje: szyfrowanie bądź deszyfrowanie.
- Następnie deklarujemy zmienną `szyfr`, w której będzie przechowywane to, co wybrał użytkownik.
- Tworzymy pętlę `do..while()` z warunkiem `menu != 1 && menu != 2`, by pętla wykonywała się, dopóki nie zostanie wybrana odpowiednia opcja z menu.
- Wypisujemy możliwe opcje w menu oraz odpowiadające im liczby, jakie ma wprowadzić użytkownik.
- Wczytujemy wybór użytkownika do zmiennej `szyfr`.
- Używamy polecenia `system("cls")`, by wyczyścić ekran konsoli z poprzednio wyświetlonego menu.
- Po wyjściu z pętli sprawdzamy, czy użytkownik wybrał opcję szyfrowania. Jeśli tak, to zwracamy w miejsce wywołania naszej funkcji `prawda`, w przeciwnym przypadku zwracamy `fałsz`.

Krok 6

W funkcji `main` wywołujemy funkcje odpowiedzialne za wyświetlanie obu menu.

```
#include <iostream>

using namespace std;

int main()
{
    int MetodaSzyfrowania;
    bool CzySzyfrujemy;

    while (true)
    {
        MetodaSzyfrowania = StworzMenu();
        CzySzyfrujemy = SzyfrDeszyfr();
    }

}
```

Ponieważ funkcje zwracają wybory wskazane przez użytkownika, przypisujemy je do odpowiednio zadeklarowanych zmiennych `MetodaSzyfrowania` oraz `CzySzyfrujemy`.

Krok 7

Deklarujemy ciąg znaków typu `string` – zapiszemy w nim wiadomość, którą będziemy szyfrować bądź deszyfrować.

```
#include <iostream>

using namespace std;

int main()
{
    int MetodaSzyfrowania;
    bool CzySzyfrujemy;

    while (true)
    {
        MetodaSzyfrowania = StworzMenu();
        CzySzyfrujemy = SzyfrDeszyfr();

        string wiadomosc;

    }

}
```

Krok 8

Wyświetlamy odpowiedni komunikat w zależności od tego, czy użytkownik wybrał opcję szyfrowania, czy deszyfrowania.

```
#include <iostream>

using namespace std;

int main()
{
    int MetodaSzyfrowania;
    bool CzySzyfrujemy;

    while (true)
    {
```

```
        MetodaSzyfrowania = StworzMenu();
        CzySzyfrujemy = SzyfrDeszyfr();

        string wiadomosc;

        cout << "Podaj wiadomosc do ";

        if (CzySzyfrujemy == true)
            cout << "zaszyfrowania: ";
        else
            cout << "odszyfrowania: ";
    }
}
```

Krok 9

Wczytujemy wiadomość, na której będziemy działać.

```
#include <iostream>

using namespace std;

int main()
{
    int MetodaSzyfrowania;
    bool CzySzyfrujemy;

    while (true)
    {
        MetodaSzyfrowania = StworzMenu();
        CzySzyfrujemy = SzyfrDeszyfr();

        string wiadomosc;

        cout << "Podaj wiadomosc do ";

        if (CzySzyfrujemy == true)
            cout << "zaszyfrowania: ";
        else
            cout << "odszyfrowania: ";

```

```
        cin >> wiadomosc;
    }

}
```

Krok 10

Wyświetlamy prośbę o podanie klucza szyfrującego.

```
#include <iostream>

using namespace std;

int main()
{
    int MetodaSzyfrowania;
    bool CzySzyfrujemy;

    while (true)
    {
        MetodaSzyfrowania = StworzMenu();
        CzySzyfrujemy = SzyfrDeszyfr();

        string wiadomosc;

        cout << "Podaj wiadomosc do ";

        if (CzySzyfrujemy == true)
            cout << "zaszyfrowania: ";
        else
            cout << "odszyfrowania: ";

        cin >> wiadomosc;

        cout << "Podaj klucz szyfrujacy: ";
    }
}
```

Krok 11

Wykorzystujemy przełącznik `switch` i sprawdzamy za jego pomocą, którą metodę szyfrowania wybrał użytkownik. W naszym przypadku wystarczyłaby instrukcja `if`, jednak wykorzystujemy przełącznik `switch`, by ułatwić przyszłą rozbudowę programu.

```
#include <iostream>

using namespace std;

int main()
{
    int MetodaSzyfrowania;
    bool CzySzyfrujemy;

    while (true)
    {
        MetodaSzyfrowania = StworzMenu();
        CzySzyfrujemy = SzyfrDeszyfr();

        string wiadomosc;

        cout << "Podaj wiadomosc do ";

        if (CzySzyfrujemy == true)
            cout << "zaszyfrowania: ";
        else
            cout << "odszyfrowania: ";

        cin >> wiadomosc;

        cout << "Podaj klucz szyfrujacy: ";

        switch (MetodaSzyfrowania)
        {
        case 1:
        case 2:
        }
    }
}
```

Krok 12

Zapisujemy odpowiednie instrukcje dla przypadku pierwszego.

```
#include <iostream>

using namespace std;

int main()
{
    int MetodaSzyfrowania;
    bool CzySzyfrujemy;

    while (true)
    {
        MetodaSzyfrowania = StworzMenu();
        CzySzyfrujemy = SzyfrDeszyfr();

        string wiadomosc;

        cout << "Podaj wiadomosc do ";

        if (CzySzyfrujemy == true)
            cout << "zaszyfrowania: ";
        else
            cout << "odszyfrowania: ";

        cin >> wiadomosc;

        cout << "Podaj klucz szyfrujacy: ";

        switch (MetodaSzyfrowania)
        {
        case 1:
            int KluczCezar;
            cin >> KluczCezar;
            if (CzySzyfrujemy == true)
                cout << "Zaszyfrowana wiadomosc: " << SzyfrCezar(
            else
                cout << "Odszyfrowana wiadomosc: " << DeszyfrCeza
            break;
        case 2:
        }
```

```
    }
}
```

- Pierwsza z instrukcji to deklaracja odpowiedniej zmiennej do przechowania klucza szyfrującego. Jest to zmienna typu `int`, ponieważ w szyfrze Cezara możemy przesuwać tylko o wartości w postaci liczb całkowitych.
- Wczytujemy klucz szyfrujący do naszej zmiennej.
- Sprawdzamy, czy użytkownik wybrał opcję szyfrowania, czy deszyfrowania. Jeśli zmienna `CzySzyfrujemy` ma wartość `prawda`, to wykonujemy szyfrowanie podanej wiadomości, wykorzystując wczytany przed chwilą klucz szyfrowania.
- Jeśli użytkownik wybrał opcję szyfrowania, wyświetlamy odpowiedni komunikat wraz z zaszyfrowaną wiadomością, którą uzyskujemy z funkcji `SzyfrCezar()`. Funkcja ta jest dokładnie tą samą, która została opisana we wcześniejszych e-materiałach.
- Jeśli użytkownik wybrał opcję deszyfrowania, wyświetlamy odpowiedni komunikat wraz z odszyfrowaną wiadomością, którą uzyskujemy z funkcji `DeszyfrCezar()`. Funkcja ta jest dokładnie tą samą, która została opisana we wcześniejszych e-materiałach.
- Wykorzystujemy instrukcję `break`, by nasz program po wykonaniu instrukcji dla przypadku pierwszego nie zaczął wykonywać instrukcji dla przypadku drugiego.

Krok 13

Zapisujemy odpowiednie instrukcje dla przypadku drugiego.

```
#include <iostream>

using namespace std;

int main()
{
    int MetodaSzyfrowania;
    bool CzySzyfrujemy;

    while (true)
    {
        MetodaSzyfrowania = StworzMenu();
        CzySzyfrujemy = SzyfrDeszyfr();

        string wiadomosc;

        cout << "Podaj wiadomosc do ";
```

```
18
19          if (CzySzyfrujemy == true)
20              cout << "zaszyfrowania: ";
21          else
22              cout << "odszyfrowania: ";
23
24          cin >> wiadomosc;
25
26          cout << "Podaj klucz szyfrujacy: ";
27
28          switch (MetodaSzyfrowania)
29          {
30          case 1:
31              int KluczCezar;
32              cin >> KluczCezar;
33              if (CzySzyfrujemy == true)
34                  cout << "Zaszyfrowana wiadomosc: " << SzyfrCezar(
35              else
36                  cout << "Odszyfrowana wiadomosc: " << DeszyfrCeza
37              break;
38          case 2:
39              char tabela[26][26];
40              wypelnij(tabela);
41              string KluczVigenere;
42              cin >> KluczVigenere;
43              if (CzySzyfrujemy == true)
44                  cout << "Zaszyfrowana wiadomosc: " << SzyfrVigene
45              else
46                  cout << "Odszyfrowana wiadomosc: " << DeszyfrVige
47              break;
48          }
49      }
50  }
```

- Deklarujemy tablicę 26 na 26 znaków potrzebną do szyfrowania Vigenère'a.
- Wypełniamy zadeklarowaną przed chwilą tablicę funkcją `wypelnij`.
- Deklarujemy łańcuch znaków `KluczVigenere`, w którym będziemy przechowywać klucz szyfrujący.
- Sprawdzamy, czy użytkownik wybrał opcję szyfrowania, czy deszyfrowania. Jeśli zmienna `CzySzyfrujemy` ma wartość `prawda`, to wykonujemy szyfrowanie podanej wiadomości, wykorzystując wczytany przed chwilą klucz szyfrowania.

- Jeśli użytkownik wybrał opcję szyfrowania, wyświetlamy odpowiedni komunikat wraz z zaszyfrowaną wiadomością, którą uzyskujemy z funkcji `SzyfrVigenere()`. Funkcja ta jest dokładnie tą samą, która została opisana we wcześniejszych e-materiałach.
- Jeśli użytkownik wybrał opcję deszyfrowania, wyświetlamy odpowiedni komunikat, wraz z odszyfrowaną wiadomością, którą uzyskujemy z funkcji `DeszyfrVigenere()`. Funkcja ta jest dokładnie tą samą, która została opisana we wcześniejszych e-materiałach.
- Wykorzystujemy instrukcję `break`, by nasz program po wykonaniu instrukcji dla przypadku pierwszego nie zaczął wykonywać instrukcji dla przypadku drugiego.

Krok 14

Po wyświetleniu wiadomości czekamy na dowolny znak od użytkownika przy użyciu polecenia `system("pause")`, a następnie czyścimy ekran konsoli, korzystając z polecenia `system("cls")`.

```
#include <iostream>

using namespace std;

int main()
{
    int MetodaSzyfrowania;
    bool CzySzyfrujemy;

    while (true)
    {
        MetodaSzyfrowania = StworzMenu();
        CzySzyfrujemy = SzyfrDeszyfr();

        string wiadomosc;

        cout << "Podaj wiadomosc do ";

        if (CzySzyfrujemy == true)
            cout << "zaszyfrowania: ";
        else
            cout << "odszyfrowania: ";

        cin >> wiadomosc;

```

```
26          cout << "Podaj klucz szyfrujacy: ";
27
28          switch (MetodaSzyfrowania)
29          {
30          case 1:
31              int KluczCezar;
32              cin >> KluczCezar;
33              if (CzySzyfrujemy == true)
34                  cout << "Zaszyfrowana wiadomosc: " << SzyfrCezar(
35              else
36                  cout << "Odszyfrowana wiadomosc: " << DeszyfrCeza
37              break;
38          case 2:
39              char tabela[26][26];
40              wypelnij(tabela);
41              string KluczVigenere;
42              cin >> KluczVigenere;
43              if (CzySzyfrujemy == true)
44                  cout << "Zaszyfrowana wiadomosc: " << SzyfrVigene
45              else
46                  cout << "Odszyfrowana wiadomosc: " << DeszyfrVige
47              break;
48          }
49          system("pause");
50          system("cls");
51      }
52  }
```

Słownik

instrukcja `exit()`

instrukcja, która wymusza zamknięcie programu z kodem umieszczonym w nawiasach okrągłych; kod 0 oznacza EXIT_SUCCESS

instrukcja `system("cls")`

instrukcja, która czyści okno konsoli z wyświetlonych dotychczas komunikatów

instrukcja `system("pause")`

instrukcja, która zatrzymuje działanie programu i czeka do momentu naciśnięcia dowolnego klawisza

Sprawdź się

Pokaż ćwiczenia: 

Ćwiczenie 1

Napisz program, który wypisze zaszyfrowaną wiadomość za pomocą jednej z dostępnych w menu metod szyfrowania.

Specyfikacja:

Dane:

- `wiadomosc` - ciąg znaków
- `kluczV` - ciąg znaków

Wynik:

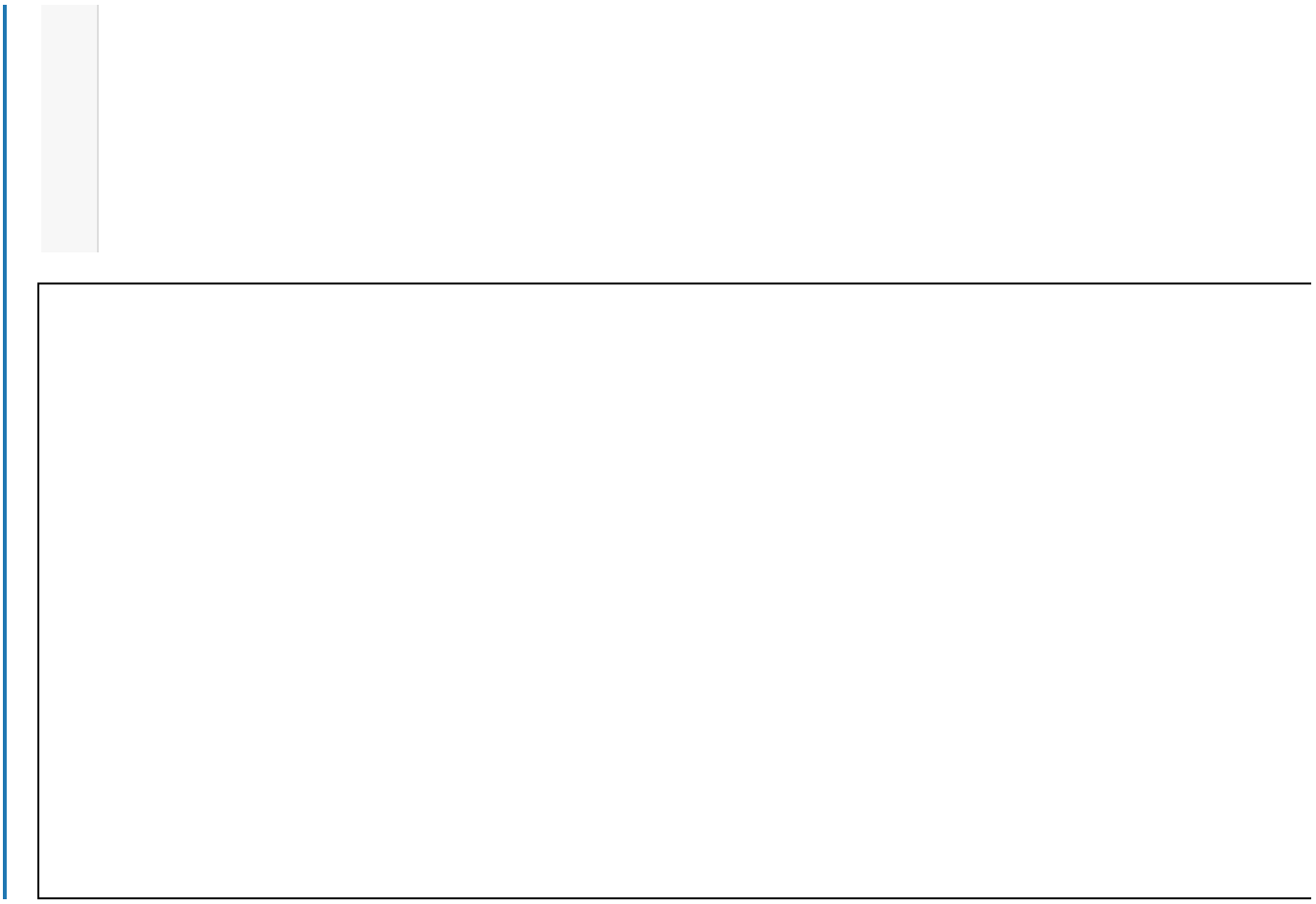
- x - ciąg znaków

Twoje zadania

1. Wybór metody szyfrowania ma się odbyć poprzez ustawienie wartości w kodzie programu. Wyświetlona ma zostać tylko poprawna odpowiedź. Załóż, że użytkownik wybierze szyfr Vigenère'a.

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

using namespace std;

string szyfrCezar(string, int, int);

void wypelnij(char[26][26]);
string szyfrVigenere(string, string, char[26][26]);

int main()
{
    string wiadomosc = "kotmaale";
    string kluczV = "pies";
```

```

```



Ćwiczenie 2

Napisz program deszyfrujący podaną wiadomość, za pomocą szyfru Cezara.

Specyfikacja:

Dane:

- `zaszyfrowanaWiadomosc` - ciąg znaków
- `kluczC` - liczba całkowita

Wynik:

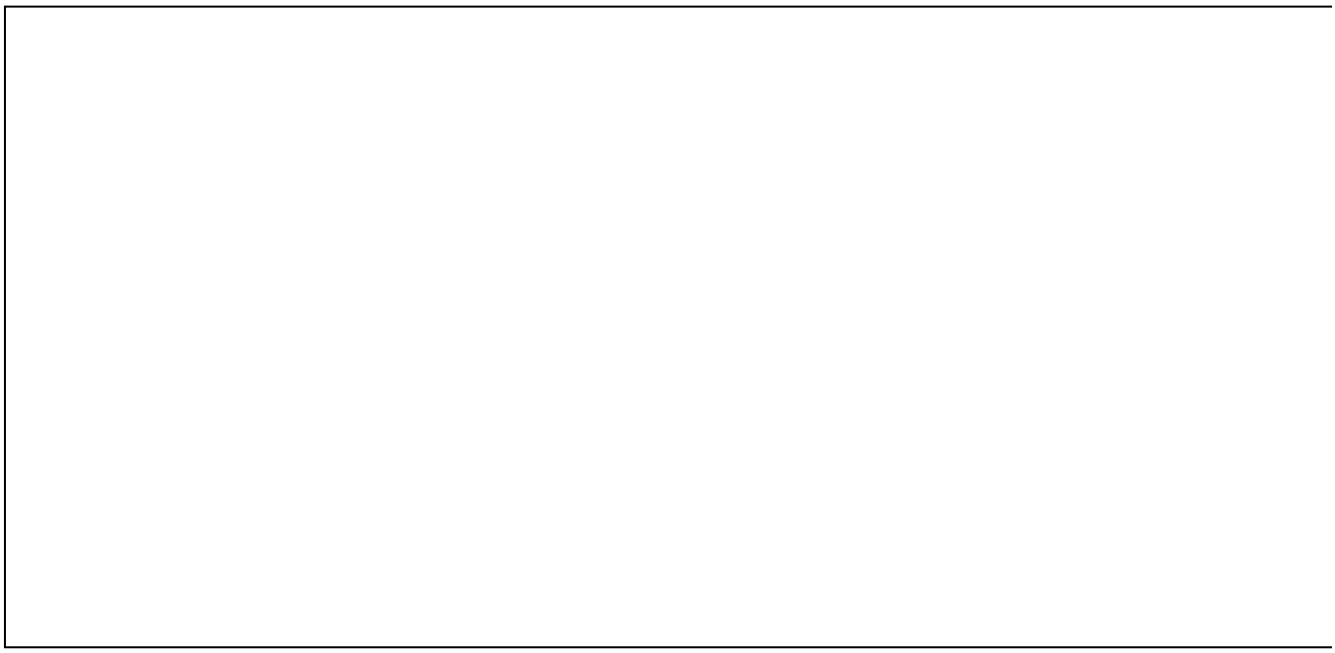
- `wiadomoscOdszyfrowana` - ciąg znaków

Twoje zadania

1. Program powinien wyświetlić odszyfrowaną wiadomość zapisaną za pomocą szyfru Cezara.

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

using namespace std;

int main()
{
    string zaszyfrowanaWiadomosc = "BihoaxfjwrnSnbcLrntjfn";
    int kluczC = 9;
}
```

```
1
```



Ćwiczenie 3

Napisz program, który zakoduje tekst za pomocą szyfru Cezara, a następnie wykorzysta ten tekst, by zaszyfrować wiadomość szyfrem Vigenère"a.

Specyfikacja:

Dane:

- `tekst` - ciąg znaków
- `WiadomoscV` - ciąg znaków
- `kluczC` - liczba całkowita

Wynik:

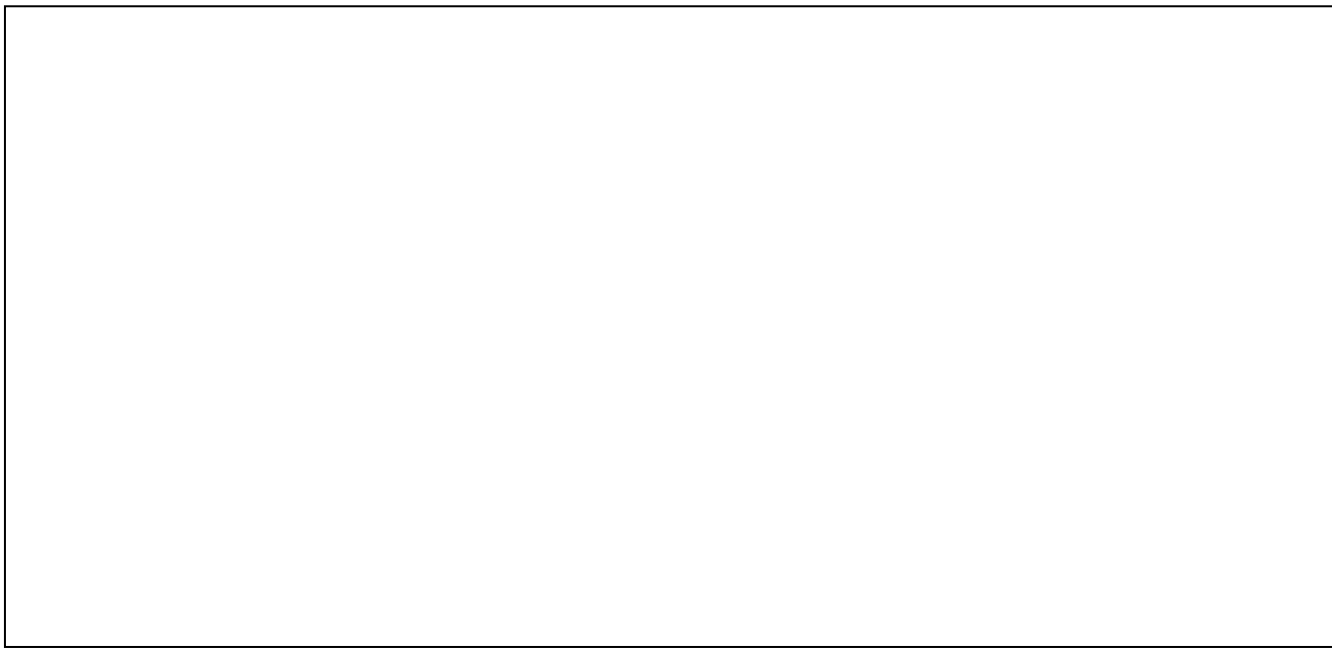
- `WiadomoscZaszyfrowana` - ciąg znaków

Twoje zadania

1. Program powinien zakodować tekst za pomocą szyfru Cezara, a następnie to, co otrzyma, wykorzystać jako klucz, by zaszyfrować wiadomość.

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

using namespace std;

string SzyfrCezar(string, int, int);

void wypelnij(char[26][26]);
string SzyfrVigenere(string, string, char[26][26]);

int main()
{
    string tekst = "szyfrcezara";
    string WiadomoscZaszyfrowana;
```

```
1
```



Dla nauczyciela

Autor: Maurycy Gast
Przedmiot: Informatyka

Temat: Szyfrowanie i deszyfrowanie 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.

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 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;

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):

- Opracujesz program, który będzie realizował operacje szyfrowania i deszyfrowania.
- Powtórzysz poznane metody szyfrowania i deszyfrowania tekstu.
- Wykorzystasz zdobytą wiedzę na temat algorytmów szyfrujących do zaimplementowania praktycznego programu.

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: „Szyfrowanie i deszyfrowanie w języku C++". Nauczyciel prosi uczniów o zapoznanie się z treściami w sekcji „Przeczytaj".

Faza wstępna:

1. Nauczyciel inicjuje rozmowę wprowadzającą w temat lekcji. Przedstawia cele zajęć oraz 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.** Uczniowie przystępują do cichego czytania tekstu zawartego w sekcji „Przeczytaj", jeśli nauczyciel – na podstawie raportu na platformie – uważa, że przygotowanie uczniów jest wystarczające, może pominąć tę czynność.
2. **Praca z multimedium.** Nauczyciel wyświetla zawartość sekcji „Film samouczek". Uczniowie wspólnie zapoznają się z treścią zawartego w niej multimedium. Następnie wykonują program przedstawiony na filmie.
3. **Ćwiczenie umiejętności.** Uczniowie wykonują indywidualnie ćwiczenia nr 1-2 z sekcji „Sprawdź się". Po wykonaniu każdego z nich następuje omówienie rozwiązania przez nauczyciela.

Faza podsumowująca:

1. Wybrany uczeń podsumowuje zajęcia, zwracając uwagę na nabyte umiejętności, omawia ewentualne problemy podczas rozwiązania ćwiczeń z programowania w języku C++.

Praca domowa:

1. Uczniowie wykonują ćwiczenie nr 3 z sekcji „Sprawdź się".
2. Uczniowie dodają komentarze do kodu zapisanego w ramach Polecenia 1 z sekcji „Film samouczek".

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 „Film samouczek” można wykorzystać jako podsumowanie i utrwalenie wiedzy uczniów.