

**Akademia Techniczno-Humanistyczna**

**w Bielsku-Białej**

Wydział Budowy Maszyn i Informatyki

**Sprawozdanie z przedmiotu:**

**Bezpieczeństwo technologii informatycznych**

**(ćwiczenia laboratoryjne)**

**Ćwiczenie numer:**

**1**

**Temat ćwiczenia: Szyfr Cezara**

**Data: 02.10.2021**

**Wykonał: Piotr Olearczyk (nr alb. 49691)**

1. **Wykonanie programu**

**Program został napisany w języku C#**

* 1. **Metoda szyfrowania**

Metoda szyfrująca zwraca zaszyfrowany tekst i przyjmuje 2 parametry:

1. Tekst do zaszyfrowania.
2. Klucz – liczba całkowita o jaką ma przesunąć się wartość przesunięcia szyfrującego.

public static string Encrypt(string text, int key)

{

var encrypted = string.Empty;

foreach (var character in text)

{

if (char.IsUpper(character))

{

var characterIndex = character - 'A';

var characterShifted = (characterIndex + key) % 26 + 'A';

encrypted += (char)(characterShifted);

}

else if (char.IsLower(character))

{

var characterIndex = character - 'a';

var characterShifted = (characterIndex + key) % 26 + 'a';

encrypted += (char)(characterShifted);

}

else

{

throw new ArgumentException("Text cannot be different than alphabet");

}

}

return encrypted;

}

* 1. **Metoda deszyfrująca**

Metoda deszyfrująca zwraca odszyfrowany tekst i przyjmuje 2 parametry:

1. Tekst do odszyfrowania.
2. Klucz – liczba całkowita o jaką ma przesunąć się wartość przesunięcia deszyfrującego.

public static string Decrypt(string text, int key)

{

var decrypted = string.Empty;

key %= 26;

foreach (var character in text)

{

if (char.IsUpper(character))

{

var characterIndex = character - 'A';

var characterOrgPos = (characterIndex - key) % 26 + 'A';

decrypted += (char)(characterOrgPos);

}

else if (char.IsLower(character))

{

var characterIndex = character - 'a';

var characterOrgPos = (characterIndex - key) % 26 + 'a';

decrypted += (char)(characterOrgPos);

}

else

{

throw new ArgumentException("Text cannot be different than alphabet");

}

}

return decrypted;

}

* 1. **Struktura główna programu – wywołanie szyfrowania**

Program przyjmuje 3 argumenty:

* + 1. Słowa „encrypt” lub „decrypt” informują program o wykonaniu szyfrowania lub deszyfrowania tekstu.
    2. Następnym argumentem jest liczba przesunięcia
    3. Ostatnim argumentem jest tekst do szyfrowania lub odszyfrowania.

public class Program

{

public static void Main(string[] args)

{

if (args.Length != 3)

{

Console.WriteLine("Program for encryption/decryption must have 3 parameters:\n" +

"1) 'encrypt' or 'decrypt' keyword\n" +

"2) Integer key\n" +

"3) Text to encrypt");

}

var encryptionMethod = args[0];

if(args.Length != 3)

return;

var keyExist = int.TryParse(args[1], out int key);

if (!keyExist)

{

Console.WriteLine("Cannot parse second parameter - must be integer");

return;

}

if (encryptionMethod == "encrypt")

{

var encryptedTxt = CaesarCipher.Encrypt(args[2], key);

Console.WriteLine(encryptedTxt);

}

else if (encryptionMethod == "decrypt")

{

var decryptedTxt = CaesarCipher.Decrypt(args[2], key);

Console.WriteLine(decryptedTxt);

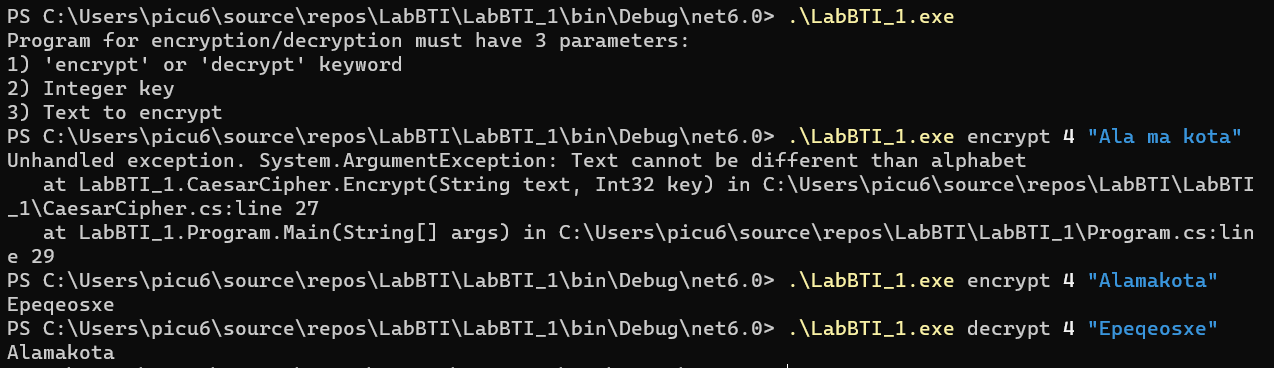
}

}

}

1. **Testy**

Testy programu w konsoli PowerShell

****

* + 1. Jak można zobaczyć na powyższym zdjęciu ekranu celem szyfrowania był tekst „Ala ma kota”
    2. Ponieważ napisany przeze mnie program nie uwzględnia spacji – wywołałem szyfrowanie tekstu „Alamakota” o przesunięciu 4.
    3. Otrzymałem zaszyfrowany tekst „Epeqeosxe”
    4. Następnie odszyfrowałem otrzymany tekst i dostałem z powrotem „Alamakota”

1. **Wnioski**
   1. Szyfrowanie szyfrem Cezara jest prostym sposobem – w przypadku szyfrowania zbyt prostym – odpowiednim programem można techniką brute-force znaleźć klucz szyfrujący do odkodowania każdej z zaszyfrowanych wiadomości.
   2. Jak widzimy te 2 metody szyfrujące dzieli tylko i wyłącznie znak przy ustawianiu indeksu litery – świadczy to o prostocie techniki szyfrującej.