Национальный технический университет Украины

«киевский политехнический институт»

Факультет информатики и вычислительной техники

Кафедра автоматики и управления в технических системах

Лабораторная работа № 1

по дисциплине «Сучасні технології програмування - 2»

Тема: Графічний редактор

Шаблоны «одиночка», «итератор», «прокси», «состояние», «стратегия»

|  |
| --- |
|  |

Выполнили:

студенты группы ІА-31

Стеценко Максим

Проверил:

Амонс О. А.

Киев 2015

**Задание.**

**1. Ознакомиться с краткими теоретическими сведениями. 2. Реализовать часть функционала рабочей программы в виде классов и их взаимодействий для достижения конкретных функциональных возможностей. 3. Применение одного из рассматриваемых шаблонов при реализации программы.**

**ТЕОРЕТИЧНИЙ МАТЕРІАЛ**

1. **Шифр Цезаря**

Також відомий як шифр здвигу, код Цезаря або здвиг Цезаря – один із найпростіших і найбільш відомих методів шифрування.

Шифр Цезаря - це вигляд шифру підстановки, моно-алфавітний шифр заміщення, в якому кожен символ у відкритому тексті замінюється буквою що знаходиться на деяке постійне число позицій лівіше або правіше за нього в алфавіті. Наприклад, в шифрі із зрушенням 3, А була б замінена на Г, Би стане Д, і так далі.

Шифр названий на честь римського імператора Гая Юлія Цезаря, що використав його для секретного листування зі своїми генералами.

* 1. Шифрування

Крок шифрування, виконуваний шифром Цезаря, часто включається як частина складніших схем, таких як шифр Віженера, і все ще має сучасне застосування в системі RОТ13. Як і всі моно-алфавітні шифри, шифр Цезаря легко зламується і не має практично жодного вживання на практиці.

Якщо зіставити кожному символу алфавіту його порядковий номер (нумеруючи з 0), то шифрування і дешифровку можна виразити формулами модульної арифметики:

𝑦 = (𝑥 + 𝑘 ) 𝑚𝑜𝑑 𝑛

𝑥 = (𝑦 − 𝑘)𝑚𝑜𝑑 𝑛,

де х – символ відкритого тексту, у – символ шифрованого тексту, 𝑛 - потужність алфавіту, 𝑘 – ключ.

З точки зору математики шифр Цезаря є окремим випадком аффінного шифру.

* 1. Дешифрування

Для шифру Цезаря є простий метод розшифровки – метод смужок. Із зашифрованого повідомлення вибирається деяка частина з 𝑛 символів. Заготовлює 𝑛 смужок з вертикально виписаним алфавітом. Смужки прикладаються один до одного на позиціях частини повідомлення, що дешифрується, і позиціюють переміщенням так, щоб у визначеному рядку виявилося слово, що дешифрувалося, з 𝑛 букв, після цього проводиться переміщення по рядках отриманої таблиці до здобуття єдиного осмисленого поєднання, яке служить розшифровкою даного слова. Одночасно знаходимо величину зрушення.

* 1. Алгоритм шифрування

1) Буквам алфавіту присвоюються порядкові номери. Визначуваний порядковий номер шифруючої букви. 2) Вибираємо крок зсуву 𝑘. 3) Обчислюємо порядковий номер у за допомогою рівняння шифрування

𝑦 = (𝑥 + 𝑘 ) 𝑚𝑜𝑑 𝑛

Дешифрування в зворотному порядку.

1. **ШИФР ТРИТЕМІУСА**

Це вдосконалений шифр Цезаря, тобто шифр зсуву. По алгоритму шифрування, кожна буква зсуву зміщується на букву, що відстає від даних на деякий крок. Тут крок зсуву робиться змінним, тобто залежним від якихнебудь додаткових чинників. Наприклад, можна задати закон зсуву у вигляді лінійної функції (рівняння шифрування) позиції шифруючої букви. Сама функція повинна гарантувати цілочисельне значення. Пряма функція шифрування повинна мати зворотну функцію шифрування, теж цілочисельну.

* 1. Криптосистема шифру Тритеміуса

Рівнянням шифрування називається співвідношення, що описує процес утворення зашифрованих даних з відкритих даних в результаті перетворень, заданих алгоритмом криптографічного перетворення. Рівняння шифрування для шифру Тритеміуса має наступний вигляд:

𝐿 = (𝑚 + 𝑘)𝑚𝑜𝑑 𝑁,

де 𝐿 - номер зашифрованої букви в алфавіті, 𝑚 - номер позиції букв шифрую чого тексту в алфавіті, 𝑘 - крок зміщення, 𝑁 - число букв алфавіту.

Шифр Тритеміуса ще називають шифром з обмеженим ключем, так як в якості ключа використовується випадковий текст, але обмеженого об’єму.

* 1. Алгоритм шифрування

1) Визначаємо порядковий номер шифруючої букви в тексті. 2) Визначаємо код букви в алфавіті. 3) Обчислюємо зміщення 𝑘. 4) Знаходимо код зашифрованої букви, користуючись наступним рівнянням:

𝐿 = (𝑚 + 𝑘)𝑚𝑜𝑑 𝑁,

де 𝐿 - номер зашифрованої букви в алфавіті, 𝑚 - номер позиції букв шифрую чого тексту в алфавіті, 𝑘 - крок зміщення, 𝑁 - число букв алфавіту.

5) За кодом 𝐿 відновлюємо наступну букву криптограми. 6) Повторюємо пункти 1..5 до закінчення тексту шифрограми.

* 1. Алгоритм дешифрування

1) Визначаємо порядковий номер дешифруючої букви в тексті. 2) Визначаємо код букви в алфавіті. 3) Обчислюємо зміщення 𝑘. 4) Знаходимо код розшифрованої букви, користуючись наступним рівнянням:

𝑚 = (𝐿 − 𝑘)𝑚𝑜𝑑 𝑁,

де 𝐿 - номер зашифрованої букви в алфавіті, 𝑚 - номер позиції букв шифрую чого тексту в алфавіті, 𝑘 - крок зміщення, 𝑁 - число букв алфавіту.

5) За кодом 𝑚 відновлюємо наступну букву криптограми. 6) Повторюємо пункти 1..5 до закінчення тексту шифрограми.

**Хід роботи**

1. Шифр Цезаря

class Caesar:Cyphers.Cypher

{

int key;

Alphabets.Alphabet alphabet;

public Caesar(int key, Alphabets.Alphabet alphabet):base()

{

this.key = key;

this.alphabet = alphabet;

}

public override string Decrypt(string text)

{

StringBuilder result = new StringBuilder();

if (key > alphabet.Quantity)

key = key % alphabet.Quantity;

foreach (var letter in text)

{

if (alphabet.Letters.Contains(letter))

if (alphabet.Letters.IndexOf(letter) - key < 0)

result.Append(alphabet.Letters[alphabet.Letters.IndexOf(letter) - key + alphabet.Quantity]);

else

result.Append(alphabet.Letters[alphabet.Letters.IndexOf(letter) - key]);

else

result.Append(letter);

}

return result.ToString();

}

public override string Encrypt(string text)

{

StringBuilder result = new StringBuilder();

if (key > alphabet.Quantity)

key = key % alphabet.Quantity;

foreach (var letter in text)

{

if (alphabet.Letters.Contains(letter))

if (alphabet.Letters.IndexOf(letter) + key > alphabet.Quantity - 1)

result.Append(alphabet.Letters[alphabet.Letters.IndexOf(letter) + key - alphabet.Quantity]);

else

result.Append(alphabet.Letters[alphabet.Letters.IndexOf(letter) + key]);

else

result.Append(letter);

}

return result.ToString();

}

}

1. Шифр Тритеміуса

class Tritemius: Cypher

{

public Alphabets.Alphabet Alphabet { get; set; }

public int A { get; set; }

public int B { get; set; }

public int C { get; set; }

public String Key { get; set; }

private bool stringKey { get; set; }

public Tritemius(Alphabets.Alphabet \_alphabet, int \_a, int \_b, int \_c)

{

this.Alphabet = \_alphabet;

this.A = \_a;

this.B = \_b;

this.C = \_c;

this.stringKey = false;

}

public Tritemius(Alphabets.Alphabet \_alphabet, String \_key)

{

this.Alphabet = \_alphabet;

this.Key = \_key;

this.stringKey = true;

}

public override string Encrypt(String \_text)

{

String cryptedText = "";

int indexInText = 1;

foreach (char a in \_text)

{

cryptedText += Alphabet.Letters[NewSymbolIndex(a, indexInText)];

Alphabet.Letters.IndexOf(a);

indexInText++;

}

return cryptedText;

}

public override string Decrypt(String \_cryptedText)

{

String decryptedText = "";

int indexInText = 1;

foreach (char a in \_cryptedText)

{

decryptedText += Alphabet.Letters[OldSymbolIndex(a, indexInText)];

Alphabet.Letters.IndexOf(a);

indexInText++;

}

return decryptedText;

}

private int NewSymbolIndex(char symbol, int indexInText)

{

int result;

if (!stringKey)

{

result = (Alphabet.Letters.IndexOf(symbol) + ((int)Math.Pow(indexInText, 2) \* A + (indexInText \* B) + C)) % Alphabet.Letters.Length;

}

else

{

result = (Alphabet.Letters.IndexOf(symbol) + (Alphabet.Letters.IndexOf(Key[(indexInText - 1) % Key.Length]) + 1)) % Alphabet.Letters.Length;

}

return result;

}

private int OldSymbolIndex(char symbol, int indexInText)

{

int result;

if (!stringKey)

{

result = (Alphabet.Letters.IndexOf(symbol) - ((int)Math.Pow(indexInText, 2) \* A + (indexInText \* B) + C)) % Alphabet.Letters.Length;

}

else

{

result = (Alphabet.Letters.IndexOf(symbol) - (Alphabet.Letters.IndexOf(Key[(indexInText - 1) % Key.Length]) + 1)) % Alphabet.Letters.Length;

}

if (result >= 0)

{

return result;

}

else

{

return Alphabet.Letters.Length - Math.Abs(result);

}

}

}

1. Гамування

public class XOREncryption: Cypher

{

string gammaKey;

Alphabets.Alphabet alphabet;

public XOREncryption(string gammaKey, Alphabets.Alphabet alphabet)

{

this.alphabet = alphabet;

this.gammaKey = gammaKey;

}

public override string Encrypt(string text)

{

int keyLenght = gammaKey.Length;

int counter = 0;

StringBuilder gamma = new StringBuilder();

StringBuilder result = new StringBuilder();

foreach (var letter in text)

{

if (counter == keyLenght)

counter = 0;

gamma.Append(gammaKey[counter]);

counter++;

}

for (int i = 0; i < text.Length; i++)

{

char letter = text[i];

char gammaLetter = gamma[i];

if (alphabet.Letters.Contains(letter))

{

int indexResult = (alphabet.Letters.IndexOf(gammaLetter) + alphabet.Letters.IndexOf(letter)) % alphabet.Quantity;

result.Append(alphabet.Letters[indexResult]);

}

else

result.Append(letter);

}

return result.ToString();

}

public override string Decrypt(string text)

{

int keyLenght = gammaKey.Length;

int counter = 0;

StringBuilder gamma = new StringBuilder();

StringBuilder result = new StringBuilder();

foreach (var letter in text)

{

if (counter == keyLenght)

counter = 0;

try

{

gamma.Append(gammaKey[counter]);

}

catch (IndexOutOfRangeException)

{

return null;

}

counter++;

}

for (int i = 0; i < text.Length; i++)

{

char letter = text[i];

char gammaLetter = gamma[i];

if (alphabet.Letters.Contains(letter))

{

int indexResult = (alphabet.Letters.IndexOf(letter) - alphabet.Letters.IndexOf(gammaLetter) + alphabet.Quantity) % alphabet.Quantity;

result.Append(alphabet.Letters[indexResult]);

}

else

result.Append(letter);

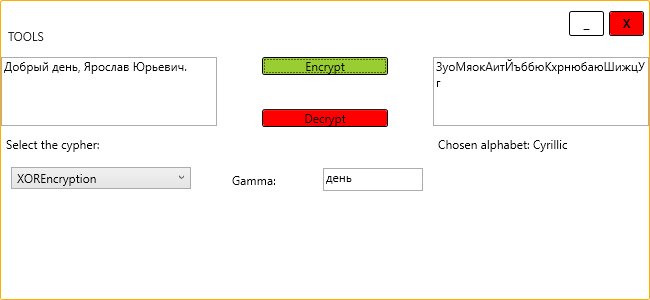
}

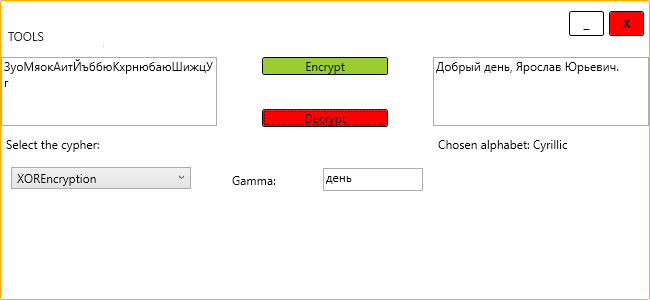
return result.ToString();

}

}

1. Програмне забезпечення для використання шифрів





**Висновки**

В даній лабораторній роботі були розглянуті наступні шифри: Цезаря, Трітеміуса, Гамування. Було зроблено висновки, що шифр Цезаря простий в реалізації, але його легко взломати. Шифр Трітеміуса більш складний для побудови, але більш надійний. Гамування – це найнадійніший із всіх розглянутих шифрів. З цього можна зробити висновки, що поліалфавітні шифри більш надійні за моноалфавітні.