Министерство науки и высшего образования Российской Федерации

Федеральное государственное автономное образовательное учреждение высшего образования

«Пермский национальный исследовательский политехнический университет»

Электротехнический факультет

Кафедра «Информационные технологии и автоматизированные системы»

Направление 09.03.04 – «Программная инженерия»

Профиль: «Разработка программно-информационных систем»

Дисциплина: «Защита информации»

Семестр 5

ОТЧЕТ

по лабораторной работе №1

Тема: «Шифры перестановки и замены»

Вариант 15

|  |  |  |
| --- | --- | --- |
|  | Выполнил: Студент группы РИС-23-2б  Д.Е. Колосов | |
|  |  | подпись, дата |
|  | Проверил: Ст. преподаватель кафедры ИТАС  В. Г. Шереметьев | |
|  |  | подпись, дата |

Пермь, 2025

**Цель работы**

Получить практические навыки по применению шифров перестановки и шифров простой замены.

**Задание**

Реализовать шифрование текстового сообщения, используя систему шифрования Виженера (15 вариант).

**ТЕОРЕТИЧЕСКИЕ СВЕДЕНИЯ**

Система Виженера подобна такой системе шифрования Цезаря, у которой ключ подстановки меняется от буквы к букве. Этот шифр многоалфавитной замены можно описать таблицей шифрования, называемой таблицей (квадратом) Виженера. Ниже показаны таблицы Виженера для русского и английского алфавитов соответственно.

Таблица Виженера используется для шифрования и дешифрования. Таблица имеет два входа:

* верхнюю строку подчеркнутых символов, используемую для считывания очередной буквы исходного открытого текста;
* крайний левый столбец ключа.

Последовательность ключей обычно получают из числовых значений букв ключевого слова.

При шифровании исходного сообщения его выписывают в строку, а под ним записывают ключевое слово (или фразу). Если ключ оказался короче сообщения, то его циклически повторяют. В процессе шифрования находят в верхней строке таблицы очередную букву исходного текста и в левом столбце очередное значение ключа. Очередная буква шифртекста находится на пересечении столбца, определяемого шифруемой буквой, и строки, определяемой числовым значением ключа.

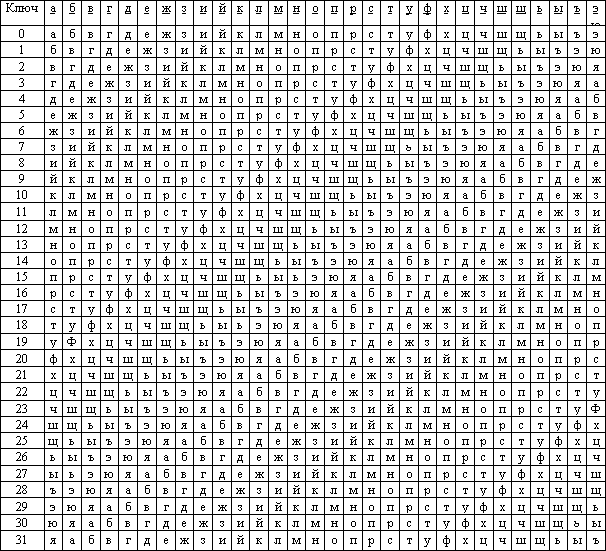


Таблица Виженера для русского алфавита

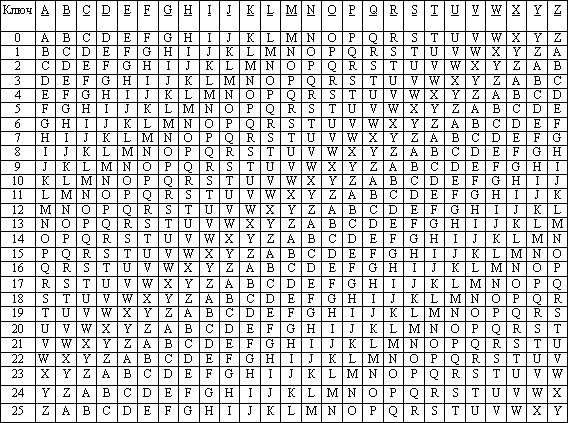


Таблица Виженера для английского алфавита

Рассмотрим пример получения шифртекста с помощью таблицы Виженера. Пусть выбрано ключевое слово АМБРОЗИЯ. Необходимо зашифровать сообщение ПРИЛЕТАЮ СЕДЬМОГО.

Выпишем исходное сообщение в строку и запишем под ним ключевое слово с повторением. В третью строку будем выписывать буквы шифртекста, определяемые из таблицы Виженера.

|  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- |
| Сообщение | П | Р | И | Л | Е | Т | А | Ю |  | С | Е | Д | Ь | М | О | Г | О |
| Ключ | А | М | Б | Р | О | З | И | Я |  | А | М | Б | Р | О | З | И | Я |
| Шифртекст | П | Ъ | Й | Ы | У | Щ | И | Э |  | С | С | Е | К | Ь | Х | Л | Н |

**ХОД РАБОТЫ**

Первым делом стоит обратить внимание, что при программной реализации нет необходимости хранить полностью таблицу шифрования Виженера, достаточно одной строки – полного алфавит в отсортированном порядке (АБВГДЕЖЗИЙКЛМНОПРСТУФХЦЧШЩЪЫЬЭЮЯ). При этом шифрование будет реализовываться с помощью математических операций над номерами позиций букв исходного и ключевого слов. Заметим, что в таблице Виженера каждая новая строка является тем же исходным алфавитом, но смещенным влево на количество символов равное номеру строки. Это позволяет при программной реализации не хранить всю таблицу шифрования, а ограничится формулой вычисления номера зашифрованной буквы в алфавите: (номер\_буквы\_исходного\_слова + номер\_букквы\_ключа) % (длинна\_алфавита), где % - нахождение остатка от деления, необходимо чтобы получившийся номер не выходил за пределы алфавита.

Из данной формулы следует, что при использовании в качестве ключа слова любой длинны состоящей только из букв а – зашифрованное слово останется равным исходному, то есть никакого шифрования не произойдёт.

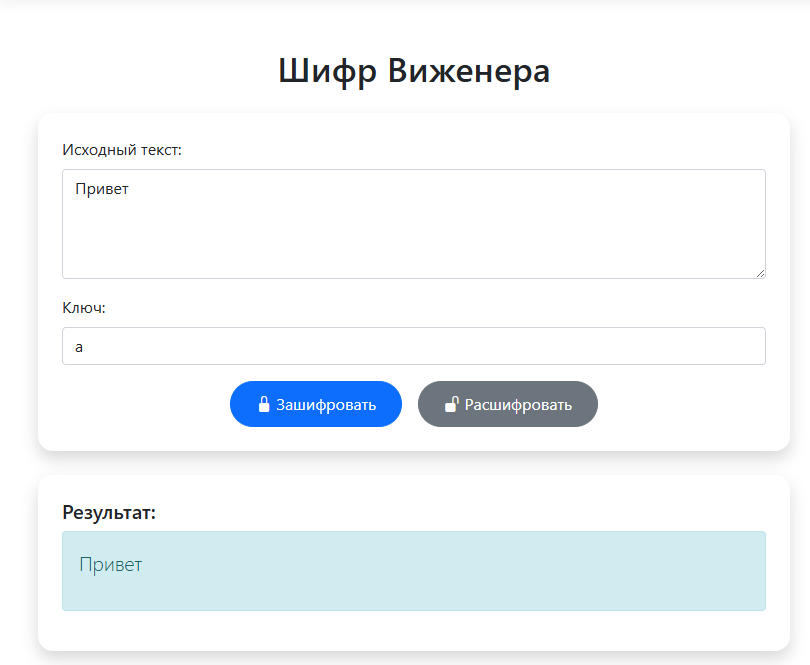


Рисунок 1 – шифрование без шифрования

Если брать букву б, то все буквы исходного слова сдвинуться по алфавиту в право.

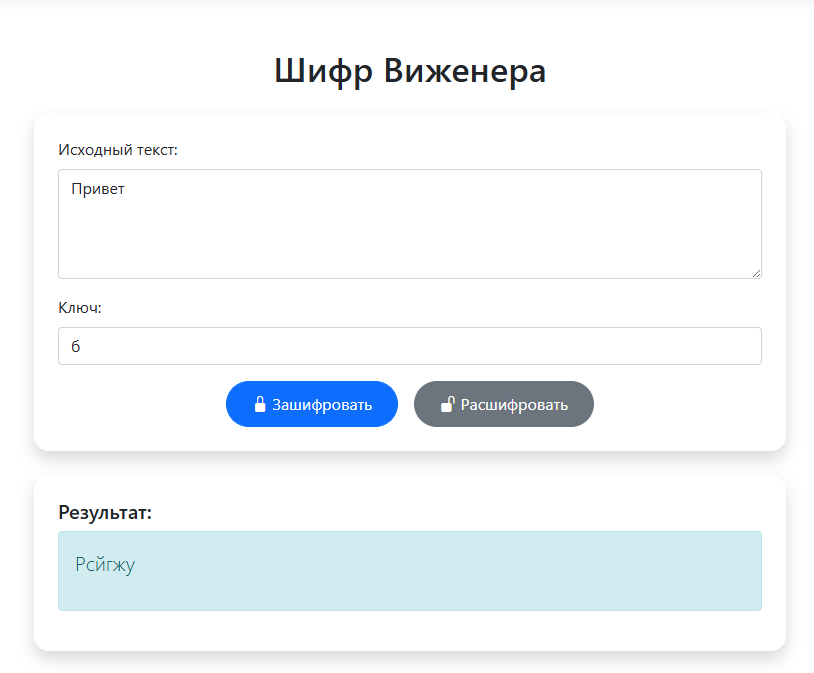


Рисунок 2 – шифрование сдвигом на 1 позицию вправо

Используя комбинацию букв б и в можно сдавить буквы исходного слова на 1 и 2 позиции вправо соответственно.

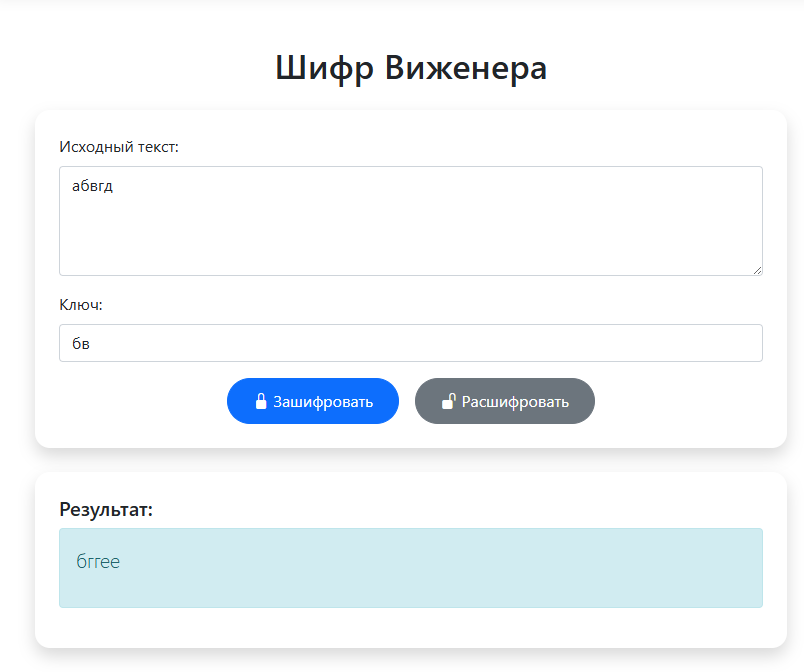


Рисунок 3 – шифрование с использованием более сложного ключа

Проведем шифрование и дешифрование текста и ключа из примера. Получаем схожие результаты, разница которых обусловлена разным порядком указания твёрдого и мягкого знаков – в русском алфавите (который использовался для программы) твердый знак идет первым, в таблице из примера первым идет мягкий знак.

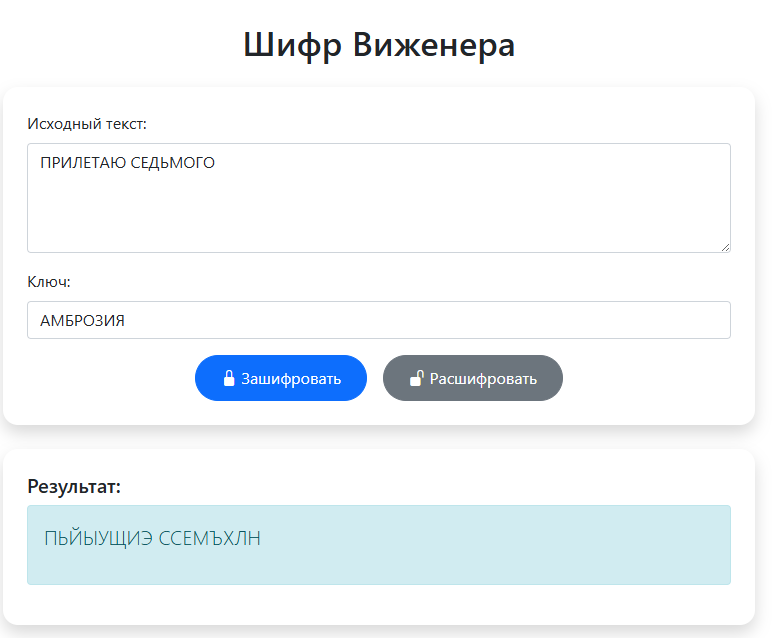


Рисунок 4 – шифрование текста и ключа из примера

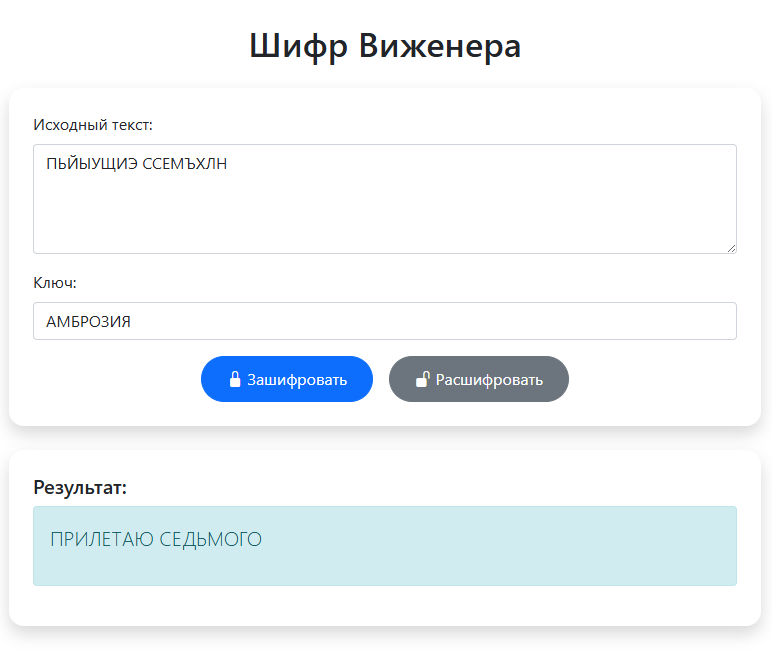


Рисунок 5 – дешифрование текста и ключа из примера

Таким образом в результате лабораторной работы были освоены практические навыки по применению шифров перестановки и шифров простой замены.

**ПРИЛОЖЕНИЕ А**

**Листинг программы**

Index.xshml

@page

@model lab1.Pages.IndexModel

@{

ViewData["Title"] = "Шифр Виженера";

}

<div class="container mt-5">

<h1 class="text-center mb-4">Шифр Виженера</h1>

<form method="post" class="card p-4 shadow">

<div class="mb-3">

<label for="InputText" class="form-label">Исходный текст:</label>

<textarea class="form-control" id="InputText" name="InputText" rows="4"

placeholder="Введите текст на русском языке">@Model.InputText</textarea>

</div>

<div class="mb-3">

<label for="Key" class="form-label">Ключ:</label>

<input type="text" class="form-control" id="Key" name="Key"

placeholder="Введите ключевое слово" value="@Model.Key">

</div>

<div class="d-grid gap-2 d-md-flex justify-content-md-center">

<button type="submit" asp-page-handler="Encrypt" class="btn btn-primary me-md-2">

<i class="bi bi-lock-fill"></i> Зашифровать

</button>

<button type="submit" asp-page-handler="Decrypt" class="btn btn-secondary">

<i class="bi bi-unlock-fill"></i> Расшифровать

</button>

</div>

</form>

@if (!string.IsNullOrEmpty(Model.Result))

{

<div class="card mt-4 p-4 shadow">

<h5 class="card-title">Результат:</h5>

<div class="alert alert-info">

<p class="lead">@Model.Result</p>

</div>

</div>

}

</div>

<style>

.container {

max-width: 800px;

}

.card {

border: none;

border-radius: 15px;

}

.btn {

border-radius: 25px;

padding: 10px 25px;

}

.alert-info {

background-color: #d1ecf1;

border-color: #bee5eb;

color: #0c5460;

}

</style>

Index.cshtml.cs

using Microsoft.AspNetCore.Mvc;

using Microsoft.AspNetCore.Mvc.RazorPages;

using System.Text;

using System.Collections.Generic;

namespace lab1.Pages

{

public class IndexModel : PageModel

{

[BindProperty]

public string InputText { get; set; } = "";

[BindProperty]

public string Key { get; set; } = "";

[BindProperty]

public string Result { get; set; } = "";

private List<bool> upperKey = new List<bool>();

private List<int> spaceAfterIndex = new List<int>();

private const string RussianAlphabet = "АБВГДЕЖЗИЙКЛМНОПРСТУФХЦЧШЩЪЫЬЭЮЯ";

public void OnGet()

{

}

public IActionResult OnPostEncrypt()

{

if (string.IsNullOrEmpty(InputText) || string.IsNullOrEmpty(Key))

{

Result = "Введите текст и ключ!";

return Page();

}

Result = Encrypt(InputText/\*.ToUpper()\*/, Key.ToUpper());

return Page();

}

public IActionResult OnPostDecrypt()

{

if (string.IsNullOrEmpty(InputText) || string.IsNullOrEmpty(Key))

{

Result = "Введите текст и ключ!";

return Page();

}

Result = Decrypt(InputText/\*.ToUpper()\*/, Key.ToUpper());

return Page();

}

private string Encrypt(string text, string key)

{

var result = new StringBuilder();

var preparedText = PrepareText(text);

var preparedKey = PrepareKey(key, preparedText.Length);

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

{

var textChar = preparedText[i];

var keyChar = preparedKey[i];

if (RussianAlphabet.Contains(textChar) && RussianAlphabet.Contains(keyChar))

{

int textIndex = RussianAlphabet.IndexOf(textChar);

int keyIndex = RussianAlphabet.IndexOf(keyChar);

int encryptedIndex = (textIndex + keyIndex) % RussianAlphabet.Length;

if (upperKey[i])

result.Append(RussianAlphabet[encryptedIndex]);

else

result.Append(char.ToLower(RussianAlphabet[encryptedIndex]));

}

else

{

result.Append(textChar);

}

}

string res = result.ToString();

foreach (int i in spaceAfterIndex)

{

res = res.Insert(i, " ");

}

spaceAfterIndex.Clear();

return res;

}

private string Decrypt(string text, string key)

{

var result = new StringBuilder();

var preparedText = PrepareText(text);

var preparedKey = PrepareKey(key, preparedText.Length);

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

{

var textChar = preparedText[i];

var keyChar = preparedKey[i];

if (RussianAlphabet.Contains(textChar) && RussianAlphabet.Contains(keyChar))

{

int textIndex = RussianAlphabet.IndexOf(textChar);

int keyIndex = RussianAlphabet.IndexOf(keyChar);

int decryptedIndex = (textIndex - keyIndex + RussianAlphabet.Length) % RussianAlphabet.Length;

result.Append(RussianAlphabet[decryptedIndex]);

}

else

{

result.Append(textChar);

}

}

string res = result.ToString();

foreach (int i in spaceAfterIndex)

{

res = res.Insert(i, " ");

}

spaceAfterIndex.Clear();

return res;

}

private string PrepareText(string text)

{

var result = new StringBuilder();

foreach (char c in text)

{

if (RussianAlphabet.Contains(char.ToUpper(c)))

{

result.Append(char.ToUpper(c));

if (char.ToUpper(c) == c)

upperKey.Add(true);

else

upperKey.Add(false);

}

else

if (c == ' ')

{

spaceAfterIndex.Add(text.IndexOf(c));

text.Remove(text.IndexOf(c), 1);

}

}

return result.ToString();

}

private string PrepareKey(string key, int length)

{

var result = new StringBuilder();

for (int i = 0; i < length; i++)

{

result.Append(key[i % key.Length]);

}

return result.ToString();

}

}

}