

Министерство науки и высшего образования Российской Федерации
Федеральное государственное бюджетное образовательное учреждение
высшего образования «Тульский государственный университет»

КАФЕДРА ИНФОРМАЦИОННОЙ БЕЗОПАСНОСТИ

РАБОТА С ФАЙЛАМИ В VBA. ФОРМА РЕГИСТРАЦИИ

отчет о лабораторной работе №13

по дисциплине
ТЕХНОЛОГИИ И МЕТОДЫ ПРОГРАММИРОВАНИЯ

Выполнила: студент гр. 230711

Павлова В.С.

Проверил: ассистент каф. ИБ

Курбаков М.Ю.

Тула, 2023 г.

ЦЕЛЬ РАБОТЫ

Цель: создать форму регистрации, которая записывала бы данные в файл в зашифрованном виде.

ЗАДАНИЕ НА РАБОТУ

В данной работе требуется создать проект с пользовательской формой, в которой будет реализована регистрация пользователя и шифрование его данных.

ТЕКСТ ПРОГРАММЫ

Для описания форм в качестве альтернативы VBA был использован язык С# и платформа WinForms. Содержимое кодового файла, описывающего разработанный проект (регистрация пользователя и сохранение его зашифрованных данных в файл), представлено в листинге 1.

Листинг 1. Описание формы SignUpForm.cs

```
public partial class SignUpForm : Form
{
    List<UserData> AccountsManager
;
    public SignUpForm()
    {
        InitializeComponent();
        CheckUsers();
    }

    private void CheckUsers()
    {
        string filePath = "D:\\WORK\\2 КУРС\\ТМП ЛР 2 курс 4
CEMECTP\\Forms\\FormLogin\\LoginForm\\userInfo.json";

        var userDataJson = File.ReadAllText(filePath);
        AccountsManager =
JsonConvert.DeserializeObject<List<UserData>>(userDataJson);
    }

    private void buttonLogin_Click(object sender, EventArgs e)
    {
        foreach (var item in AccountsManager)
        {
            if (textBoxLogin.Text == XORDecode(item.Login, "mysecretkey")
&& textBoxPassword.Text == XORDecode(item.Password,
"mysecretkey"))
            {
                MessageBox.Show("Авторизация прошла успешно!",
                    "Success",
                    MessageBoxButtons.OK,
                    MessageBoxIcon.Information);
                return;
            }
        }
        MessageBox.Show(
            "Ошибка! Неверный логин или пароль.",
            "Error",
            MessageBoxButtons.OK,
            MessageBoxIcon.Error);
        textBoxLogin.Text = string.Empty;
        textBoxPassword.Text = string.Empty;
    }

    private void SaveNewUser()
    {
        string filePath = "D:\\WORK\\2 КУРС\\ТМП ЛР 2 курс 4
CEMECTP\\Forms\\FormLogin\\LoginForm\\userInfo.json";
        string fileContent = JsonConvert.SerializeObject(AccountsManager);
        File.WriteAllText(filePath, fileContent);
    }
}
```

Листинг 1. Описание формы SignUpForm.cs (продолжение)

```
private void buttonSignUp_Click(object sender, EventArgs e)
{
    string login = XOREncode(textBoxLogin.Text, "mysecretkey");
    string password = XOREncode(textBoxPassword.Text, "mysecretkey");

    if (!AccountsManager.Any(u => u.Login == login))
    {
        var newUser = new UserData(login, password);
        AccountsManager.Add(newUser);
        SaveNewUser();
        CheckUsers();

        MessageBox.Show("Регистрация прошла успешно!",
            "Success",
            MessageBoxButtons.OK,
            MessageBoxIcon.Information);
    }
    else
    {
        MessageBox.Show("Ошибка! Такой пользователь уже зарегистрирован!",
            "Error",
            MessageBoxButtons.OK,
            MessageBoxIcon.Error);
    }
}

// Шифрование на основе XOR
public static string XOREncode(string text, string key)
{
    byte[] textBytes = Encoding.UTF8.GetBytes(text);
    byte[] keyBytes = Encoding.UTF8.GetBytes(key);
    byte[] encodedBytes = new byte[textBytes.Length];

    for (int i = 0; i < textBytes.Length; i++)
    {
        encodedBytes[i] = (byte)(textBytes[i] ^ keyBytes[i %
keyBytes.Length]);
    }

    return Convert.ToBase64String(encodedBytes);
}

public static string XORDecode(string encodedText, string key)
{
    byte[] encodedBytes = Convert.FromBase64String(encodedText);
    byte[] keyBytes = Encoding.UTF8.GetBytes(key);
    byte[] decodedBytes = new byte[encodedBytes.Length];

    for (int i = 0; i < encodedBytes.Length; i++)
    {
        decodedBytes[i] = (byte)(encodedBytes[i] ^ keyBytes[i %
keyBytes.Length]);
    }

    return Encoding.UTF8.GetString(decodedBytes);
}

private void buttonSignUp_Click(object sender, EventArgs e)
{
    string login = textBoxLogin.Text;
```

Листинг 1. Описание формы SignUpForm.cs (продолжение)

```
string password = textBoxPassword.Text;
if (!AccountsManager.Any(u => u.Login == login))
{
    var newUser = new UserData(login, password);
    AccountsManager.Add(newUser);
    SaveNewUser();
    CheckUsers();
    MessageBox.Show("Регистрация прошла успешно!",
        "Success",
        MessageBoxButtons.OK,
        MessageBoxIcon.Information);
}
else
{
    MessageBox.Show("Ошибка! Такой пользователь уже зарегистрирован!",
        "Error",
        MessageBoxButtons.OK,
        MessageBoxIcon.Error);
}
}
```

ДЕМОНСТРАЦИОННЫЙ ПРИМЕР

При запуске приложения отображается форма регистрации и входа с полями для ввода пароля и логина (рисунок 1):

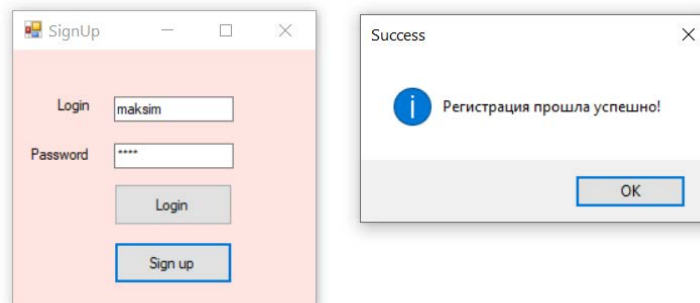
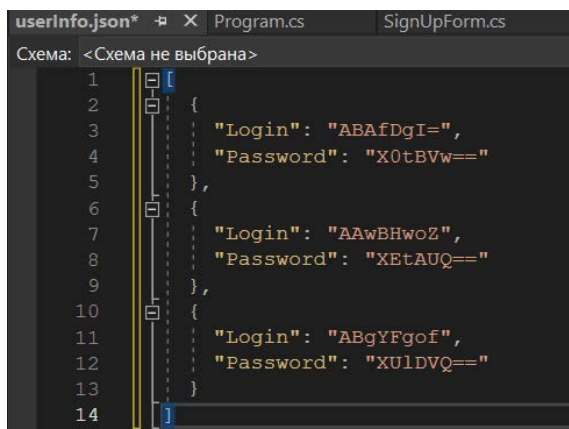


Рисунок 1 – Общий вид приложения

После ввода и нажатия на «Sign up» происходит регистрация: шифрование и сохранение введённых данных. Содержимое полученного после регистрации json-файла данных с паролем и логином показано на рисунке 2.

A screenshot of a code editor window with three tabs: 'userInfo.json*', 'Program.cs', and 'SignUpForm.cs'. The 'userInfo.json' tab is active, showing a JSON array of three encrypted user records. The records are separated by commas and each is enclosed in curly braces. The first record has 'Login': 'ABAfDgI=' and 'Password': 'X0tBVw=='. The second record has 'Login': 'AAwBHwoZ,' and 'Password': 'XEtAUQ=='. The third record has 'Login': 'ABgYFgof,' and 'Password': 'XU1DVQ=='. The editor has a dark theme and a vertical line of numbers on the left side of the code area, ranging from 1 to 14.

```
1  {  
2  {  
3    "Login": "ABAfDgI=",  
4    "Password": "X0tBVw=="  
5  },  
6  {  
7    "Login": "AAwBHwoZ",  
8    "Password": "XEtAUQ=="  
9  },  
10 {  
11   "Login": "ABgYFgof",  
12   "Password": "XU1DVQ=="  
13 }  
14 }
```

Рисунок 2 – Сохранённые зашифрованные данные

Шифрование данных происходит на основе XOR-шифрования: для каждого байта сообщения мы берем соответствующий байт из ключа и выполняем операцию XOR (исключающее «ИЛИ») между ними. Результатом будет новый байт, который мы записываем в зашифрованное сообщение. Для расшифровки мы снова берем ключ и выполняем операцию XOR между каждым байтом зашифрованного сообщения и соответствующим байтом ключа. Результатом будет исходный символ сообщения.

ВЫВОД

В ходе выполнения данной работы на примере интерфейса программирования приложений Windows Forms я научилась создавать форму регистрации и обрабатывать шифрованием пользовательскую информацию.