



**МИНОБРНАУКИ РОССИИ**  
**федеральное государственное автономное образовательное учреждение**  
**высшего образования**  
**«Московский государственный технологический университет «СТАНКИН»**  
**(ФГАОУ ВО «МГТУ «СТАНКИН»)**

---

Институт цифровых  
интеллектуальных систем

Кафедра  
компьютерных систем управления

Дисциплина «Информационные системы»

**Отчет по семинару №4**

**Выполнил:**  
**студент гр. АДБ-22-06:**

\_\_\_\_\_

(дата)

\_\_\_\_\_

(подпись)

**Ахмадуллин А.Э.**

**Проверил:**  
**Ассистент**

\_\_\_\_\_

(дата)

\_\_\_\_\_

(подпись)

**Абросимов М.А.**

**Москва 2025 г.**

## Оглавление

Цель работы: .....	3
Решение: .....	3
Результат выполнения программы: .....	12

## Цель работы:

Создать клиента, который отправляет фамилию студента в хешированном виде на сервер, сервер по хэшу ищет в базе данных информацию о студенте, записывает ее через «;» и отправляет клиенту в зашифрованном виде. Клиент расшифровывает данное сообщение.

## Решение:

Для начала создадим две библиотеки. Одна отвечает за работу с БД, другая за шифрование и расшифрование данных.

*Библиотека для работы с БД:*

```
using MySqlConnection;
using System;
using System.Collections.Generic;
using System.Data;
using System.Linq;
using System.Text;
using System.Threading.Tasks;

namespace BD_lib
{
    public class BD_lib
    {
        private DataTable Table = new DataTable();
        private MySqlDataAdapter adapter = new MySqlDataAdapter();

        public string Table_source;
        public string connectionString;
        MySqlConnection connection;
        MySqlCommand command;
        public void UploadPassword(string Password, string Database_source, string
USER_ID)
        {
            string connectionString = $"Server=localhost; Database =
{Database_source}; " +
            $"User ID = {USER_ID}; Password = {Password}";
            connection = new MySqlConnection(connectionString);

            public void OpenConnection()
            {
                if (connection.State == System.Data.ConnectionState.Closed)
connection.Open();
            }
            public void CloseConnection()
            {
                if (connection.State == System.Data.ConnectionState.Open)
connection.Close();
            }
            public MySqlConnection GetConnection()
            {
                return connection;
            }
            public void CreateSQLquery(string query)
```

```

    {
        command = new MySqlCommand(query, GetConnection());
    }

    public void FillTable(string query)
    {
        //
        OpenConnection();
        CreateSQLquery(query);
        adapter.SelectCommand = command;
        adapter.Fill(Table);
        CloseConnection();
    }

    public string GetAllDone()
    {
        if (Table.Rows.Count > 0)
        {
            DataRow dr = Table.Rows[0];
            return string.Join(";", dr[1], dr[2]);
        }
        return "ERROR";
    }
}
}

```

*Библиотека, отвечающая за шифрование и расшифрование данных:*

```

using System;
using System.Collections.Generic;
using System.Linq;
using System.Numerics;
using System.Runtime.Intrinsics;
using System.Text;
using System.Threading.Tasks;
using System.Security.Cryptography;
namespace RSA_lib
{
    public class RSA_lib
    {

        public static string GetMd5Hash(string input)
        {
            using (MD5 md5 = MD5.Create())
            {
                // Преобразуем строку в байтовый массив (UTF8 кодировка)
                byte[] inputBytes = Encoding.UTF8.GetBytes(input);

                // Вычисляем хеш
                byte[] hashBytes = md5.ComputeHash(inputBytes);

                // Конвертируем байтовый массив в строку в hex-формате
                StringBuilder sb = new StringBuilder();
                for (int i = 0; i < hashBytes.Length; i++)
                {
                    sb.Append(hashBytes[i].ToString("x2")); // "x2" - формат двух
hex-цифр
                }

                return sb.ToString();
            }
        }
    }
}

```

```

    }
}

//RSA  \//

//////////////////////////////// ГЕНЕРАЦИЯ КЛЮЧЕЙ //////////////////////////////////
// этап 1 - поиск двух простых чисел
// - - - - Поиск Простого числа - - - - -
public bool GetSimpleNum(int num)
{
    if (num == 0)
        return false;
    if ((num == 2) || (num == 3))
        return true;
    if (num % 2 == 0)
        return false;
    for (int i = 2; i < (int)(Math.Sqrt(num) + 1); i++)
    {
        if (num % i == 0)
            return false;
    }
    return true;
}

public int CreateRandom()
{
    Random rnd = new Random();
    int answer = rnd.Next(0, 602); //простые до 601
    while (GetSimpleNum(answer) == false)
    {
        answer = CreateRandom();
    }
    return answer;
}
// - - - - -

// этап 2 - поиск их произведения (модуля) (N)
public int FindModule(int p, int q)
{
    return p * q;
}

// этап 3 - вычисление функции Эйлера  $\phi(N)$ 
public int EulerFunc(int p, int q)
{
    return (p - 1) * (q - 1);
}

public int NOD(int a, int b)
{
    while (b != 0)
    {
        int temp = b;
        b = a % b;
        a = temp;
    }
    return a;
}

// этап 4 - выбор открытой экспоненты e
public int FindPublicExponent(int fi)
{
    int exponent = 0;
    while (true)

```

```

    {
        exponent = CreateRandom();
        if (exponent > 1)
            if (exponent < fi)
                if (NOD(exponent, fi) == 1)
                    return exponent;
    }
}

// этап 5 - вычисление секретной экспоненты (d)

public int EuckildAlgotithm(int fi, int e)
{
    int a = fi, b = e;
    int x = 0, y = 1;
    int last_x = 1, last_y = 0;

    while (b != 0)
    {
        int quotient = a / b;

        // Обновление a и b
        int temp = b;
        b = a % b;
        a = temp;

        // Обновление x и y
        temp = x;
        x = last_x - quotient * x;
        last_x = temp;

        temp = y;
        y = last_y - quotient * y;
        last_y = temp;
    }

    if (last_y < 0)
    {
        last_y += fi;
    }

    return last_y;
}

public List<int> CreateKeys()
{
    List<int> keys = new List<int>();
    int p = CreateRandom();
    int q = CreateRandom();

    int n = FindModule(p, q);
    int Fn = EulerFunc(p, q);
    int Expon = FindPublicExponent(Fn);
    int SecExpon = EuckildAlgotithm(Fn, Expon);
    keys.Add(Expon);
    keys.Add(SecExpon);
    keys.Add(n);

    return keys;
}

```

////////////////////////////////////

```

// ////////////////////////////////// ФУНКЦИЯ ДЛЯ РАБОТЫ С БОЛЬШИМИ ЧИСЛАМИ
////////////////////////////////////
// Быстрое возведение в степень по модулю (без переполнения, если mod^2 <
2^64)
public ulong pow_mod(ulong bbase, ulong exp, ulong mod)
{
    ulong result = 1;

    bbase %= mod; // На случай, если C > N
    while (exp > 0)
    {
        if (exp % 2 == 1)
        {
            result = (result * bbase) % mod;
        }
        bbase = (bbase * bbase) % mod;
        exp /= 2;
    }
    return result;
}

////////////////////////////////////

//////////////////////////////////// ФУНКЦИИ ДЛЯ ШИФРОВАНИЯ
////////////////////////////////////
public List<int> RSAencrypted(List<int> wordasc, List<int> openKey)
{
    List<int> answer = new List<int>();
    for (int sindex = 0; sindex < wordasc.Count; sindex++) // Исправлено:
Capacity -> Count
    {
        if (wordasc[sindex] == -1)
        {
            answer.Add(-1);
        }
        if (wordasc[sindex] == -2)
        {
            answer.Add(-2);
        }
        else
        {
            // Явное преобразование int в ulong
            ulong result = pow_mod(
                (ulong)wordasc[sindex],
                (ulong)openKey[0],
                (ulong)openKey[1]);

            // Обратное преобразование ulong в int с проверкой переполнения
            answer.Add((int)result);
        }
    }
    return answer;
}

public List<int> RSAdecrypted(List<int> wordasc, List<int> privateKey) //
функция расшифровки символов с закрытым ключом
{
    //cout << "\n";
    List<int> answer = new List<int>();
    for (int sindex = 0; sindex < wordasc.Count; sindex++)
    {
        if (wordasc[sindex] == -1) { answer.Add(-1); }
        if (wordasc[sindex] == -2) { answer.Add(-2); }
        else if (wordasc[sindex] == 0) { answer.Add(privateKey[1]); }
    }
}

```

```

        else
        {
            ulong result = pow_mod(
                (ulong)wordasc[sindex],
                (ulong)privateKey[0],
                (ulong)privateKey[1]);

            // Обратное преобразование ulong в int с проверкой переполнения
            answer.Add((int)result);
        }
    }
    return answer;
}

////////////////////////////////////
//////////////////////////////////// ФУНКЦИИ ДЛЯ РАБОТЫ С КОНСОЛЬЮ И ВЕКТОРАМИ
////////////////////////////////////

public class RussianAlphabetConverter
{
    private static readonly Dictionary<char, int> russianAlphabet = new
Dictionary<char, int>
    {
        {';', -2}, {'#', 0}, {' ', -1},
        {'A', 1}, {'a', 1},
        {'Б', 2}, {'б', 2},
        {'В', 3}, {'в', 3},
        {'Г', 4}, {'г', 4},
        {'Д', 5}, {'д', 5},
        {'Е', 6}, {'е', 6},
        {'Ё', 7}, {'ё', 7},
        {'Ж', 8}, {'ж', 8},
        {'З', 9}, {'з', 9},
        {'И', 10}, {'и', 10},
        {'Й', 11}, {'й', 11},
        {'К', 12}, {'к', 12},
        {'Л', 13}, {'л', 13},
        {'М', 14}, {'м', 14},
        {'Н', 15}, {'н', 15},
        {'О', 16}, {'о', 16},
        {'П', 17}, {'п', 17},
        {'Р', 18}, {'р', 18},
        {'С', 19}, {'с', 19},
        {'Т', 20}, {'т', 20},
        {'У', 21}, {'у', 21},
        {'Ф', 22}, {'ф', 22},
        {'Х', 23}, {'х', 23},
        {'Ц', 24}, {'ц', 24},
        {'Ч', 25}, {'ч', 25},
        {'Ш', 26}, {'ш', 26},
        {'Щ', 27}, {'щ', 27},
        {'Ъ', 28}, {'ъ', 28},
        {'Ы', 29}, {'ы', 29},
        {'Ь', 30}, {'ь', 30},
        {'Э', 31}, {'э', 31},
        {'Ю', 32}, {'ю', 32},
        {'Я', 33}, {'я', 33}
    };

    private static Dictionary<int, char> russianAlphabetReversed = new
Dictionary<int, char>

```



```

{
    {0, '#'},
    {1, 'А'},
    {2, 'Б'},
    {3, 'В'},
    {4, 'Г'},
    {5, 'Д'},
    {6, 'Е'},
    {7, 'Ё'},
    {8, 'Ж'},
    {9, 'З'},
    {10, 'И'},
    {11, 'Й'},
    {12, 'К'},
    {13, 'Л'},
    {14, 'М'},
    {15, 'Н'},
    {16, 'О'},
    {17, 'П'},
    {18, 'Р'},
    {19, 'С'},
    {20, 'Т'},
    {21, 'У'},
    {22, 'Ф'},
    {23, 'Х'},
    {24, 'Ц'},
    {25, 'Ч'},
    {26, 'Ш'},
    {27, 'Щ'},
    {28, 'Ъ'},
    {29, 'Ы'},
    {30, 'Ь'},
    {31, 'Э'},
    {32, 'Ю'},
    {33, 'Я'}
};

public List<int> GetWordNumber(string word)
{
    List<int> answer = new List<int>();
    foreach (char c in word)
    {
        if (c == ' ')
        {
            answer.Add(-1);
        }
        if (c == ';')
        {
            answer.Add(-2);
        }

        else if (russianAlphabet.TryGetValue(c, out int number))
        {
            answer.Add(number);
        }
    }
    return answer;
}

public string GetOtvvet(List<int> wordNumbers)
{
    StringBuilder answer = new StringBuilder();
    foreach (int number in wordNumbers)
    {
        if (number == -1)

```



```

        OTVET += otvet[i].ToString();
        OTVET += ':';
    }
    return OTVET;
}
return "ERROR";
});

```

app.Run();

*Код Клиента:*

```

internal class Program
{
    public static async Task Main()
    {
        string message = "";
        HttpClient httpClient = new HttpClient();

        Console.WriteLine("Введите фамилию студента: ");
        RSA_lib.RSA_lib lib = new RSA_lib.RSA_lib();
        RSA_lib.RSA_lib.RussianAlphabetConverter alp = new
RSA_lib.RSA_lib.RussianAlphabetConverter();

        string id = Console.ReadLine();

        List<int> list = new List<int>();
        List<int> PrivateKey = new List<int>();
        list = lib.CreateKeys();
        int OpenKey1 = list[0];
        int OpenKey2 = list[2];

        PrivateKey.Add(list[1]);
        PrivateKey.Add(list[2]);

        string md5hash = RSA_lib.RSA_lib.GetMd5Hash(id);

        message = md5hash + ":" + OpenKey1.ToString() + ":" + OpenKey2.ToString();
        string[] parsing = message.Split(':');

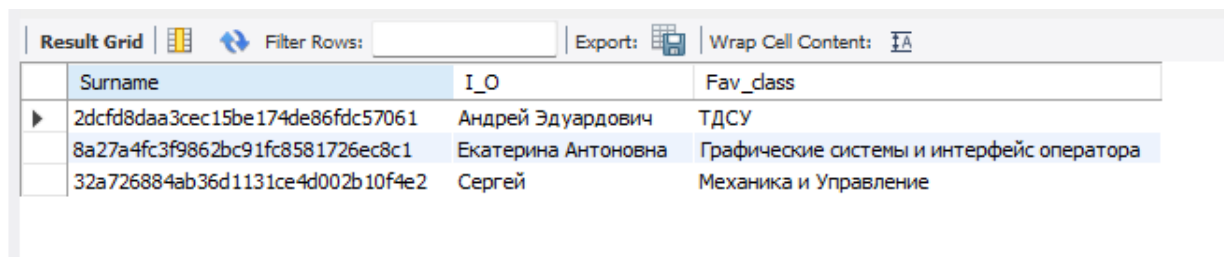
        Console.WriteLine("Информация о студенте, его любимый предмет:");
        StringContent content = new StringContent(message);
        using var response = await
httpClient.PostAsync("https://localhost:7243/ADB", content);
        string responseText = await response.Content.ReadAsStringAsync();

        string[] newparsing = responseText.Split(":");
        List<int> code = new List<int>();
        for (int i = 0; i < newparsing.Length-1; i++)
        {
            code.Add(int.Parse(newparsing[i]));
        }
        string Otvet = alp.GetOtvvet(lib.RSAdecrypted(code, PrivateKey));
        Console.WriteLine(Otvvet);

    }
}

```

*Используемая бд:*

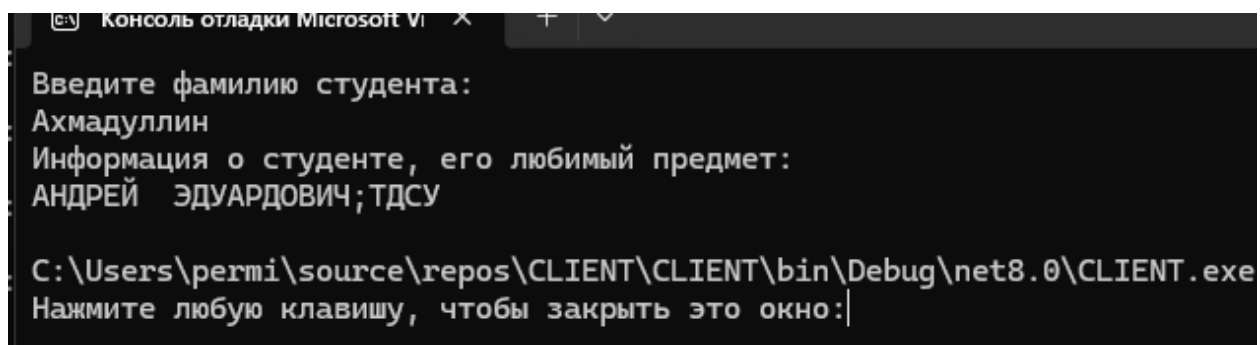


The screenshot shows a 'Result Grid' window with a table of student data. The table has three columns: 'Surname', 'I\_O', and 'Fav\_class'. There are three rows of data. The first row has a blue header, the second row is highlighted, and the third row is the current selection.

Surname	I_O	Fav_class
2dcfd8daa3cec15be174de86fdc57061	Андрей Эдуардович	ТДСУ
8a27a4fc3f9862bc91fc8581726ec8c1	Екатерина Антоновна	Графические системы и интерфейс оператора
32a726884ab36d1131ce4d002b10f4e2	Сергей	Механика и Управление

Рис.1 База данных

**Результат выполнения программы:**



The screenshot shows a Windows command prompt window titled 'Консоль отладки Microsoft Vi'. It displays the execution of a program that prompts for a student's surname and favorite subject. The user has entered 'Ахмадуллин' and 'АНДРЕЙ ЭДУАРДОВИЧ;ТДСУ'. The program path is shown as 'C:\Users\permi\source\repos\CLIENT\CLIENT\bin\Debug\net8.0\CLIENT.exe'.

```
Введите фамилию студента:  
Ахмадуллин  
Информация о студенте, его любимый предмет:  
АНДРЕЙ ЭДУАРДОВИЧ;ТДСУ  
  
C:\Users\permi\source\repos\CLIENT\CLIENT\bin\Debug\net8.0\CLIENT.exe  
Нажмите любую клавишу, чтобы закрыть это окно:
```

Рис.2 Результат выполнения программы