# Министерство образования и науки Российской Федерации

## Федеральное государственное бюджетное образовательное учреждение высшего образования

## «Новосибирский государственный технический университет»

NSTU_Logo_blue

## Кафедра теоретической и прикладной информатики

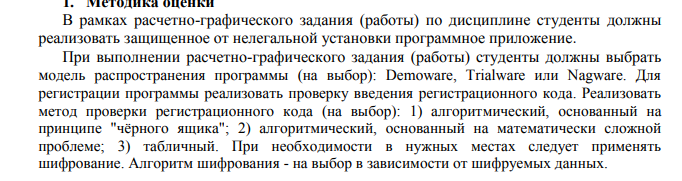
### Расчетно-графическое задание по дисциплине «Информационная безопасность»

|  |  |  |
| --- | --- | --- |
| сигма градиент синий1 | Факультет: | ПМИ |
| Группа: | ПМИ-71 |
| Студенты: | Антонов С.С |
| Вариант: | 1 |
| Преподаватель: | Авдеенко Т.В. |

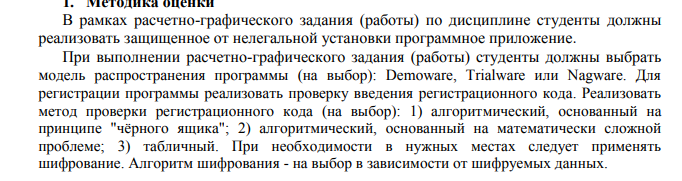
Новосибирск

2021

1. **Цель работы**



1. **Задание**



1. Калькулятор. Секретная информация: хранящаяся в файле история вычислений, которую в открытом виде можно посмотреть только с помощью калькулятора.

1. **Обоснование выбора модель распространения программы**

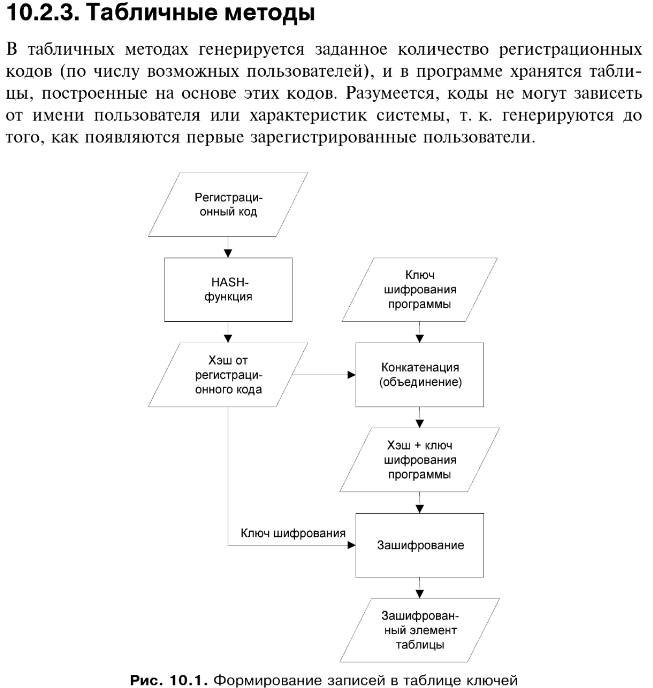
В рамках разработки органайзера мною была выбрана модель распространения Demoware.

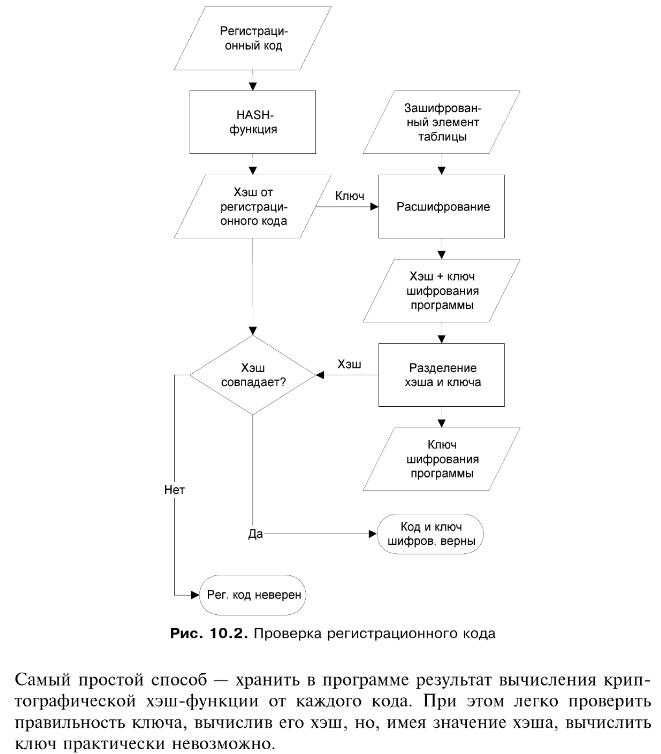
**Demoware** — тип коммерческой лицензии программного обеспечения, который служит для демонстрации возможностей программы, распространяется, обычно, бесплатно и имеет ограниченный функционал, в сравнении с основной версией. Это секвестированные версии платных программ, которые можно скачать для того, чтобы попробовать программу в действии.

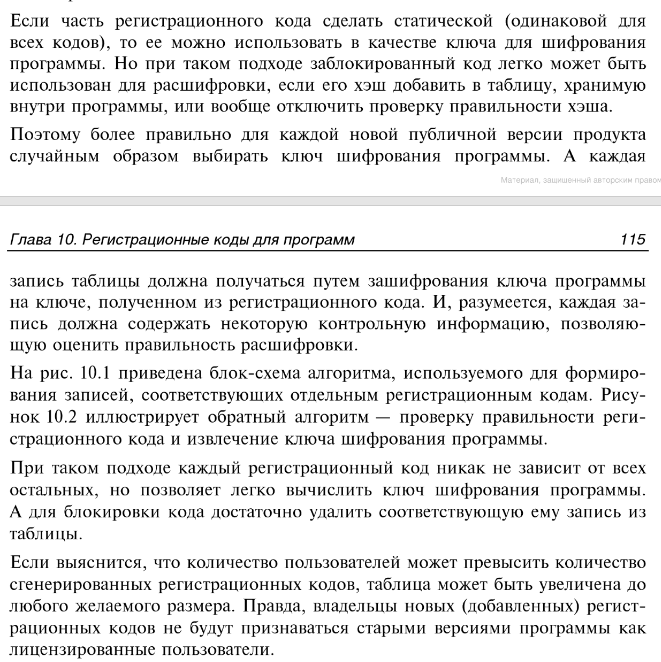
Данная модель была выбрана, потому что она служит демонстрацией возможностей программы с ограничениями при этом не доставая пользователя сообщениями о том что ему надо приобрести лицензию.

1. **Обоснование выбора метода проверки регистрационного кода**

В рамках разработки органайзера мною была выбрана метод проверки регистрационного кода: табличный.

****

****

****

Данная метод была выбрана, потому что только табличный метод один из весьма не сложных методов, а также дает возможность полностью блокировать скомпрометированные регистрационные коды, но не позволяет привязывать к имени пользователя.

1. **Обоснование применения шифрования.**

В разработанном мной приложении шифрование применяется для хеширования пароля пользователя, хеширования и шифрования регистрационного кода, для шифрования секретной информации(описания события).

1. Пароль, для безопасности хешируется с применением алгоритма SHA-512, а затем сравнивается с хэшем пароля из базы данных. В случае несоответствия хешей пароля, приложение запрещает доступ к основным функциям.
2. Для реализации табличного метода.
3. Для реализации задания, а так же для защиты секретной информации от доступа посторонних.
4. **Код программы.**

**Calc.cs**

using System;

using System.Collections.Generic;

using System.ComponentModel;

using System.Data;

using System.Drawing;

using System.Linq;

using System.Text;

using System.Threading.Tasks;

using System.Windows.Forms;

namespace CaLtoSetgey

{

public partial class Calc : Form

{

float a, b;

int count;

bool znak = true;

bool flag = false;

public Calc()

{

InitializeComponent();

var vers = Work\_sess.Access.ToString();

if (vers == "l\_0")

{

timer1.Tick += timer1\_Tick;

timer1.Enabled = true;

timer1.Start();

}

}

void func()

{

timer1.Enabled = false;

timer1.Stop();

}

private void button17\_Click(object sender, EventArgs e)

{

if (flag == true)

{

textBox1.Clear();

label1.Text = "";

flag = false;

}

textBox1.Text = textBox1.Text + 0;

}

private void button13\_Click(object sender, EventArgs e)

{

if (flag == true)

{

textBox1.Clear();

label1.Text = "";

flag = false;

}

textBox1.Text = textBox1.Text + 1;

}

private void button14\_Click(object sender, EventArgs e)

{

if (flag == true)

{

textBox1.Clear();

label1.Text = "";

flag = false;

}

textBox1.Text = textBox1.Text + 2;

}

private void button15\_Click(object sender, EventArgs e)

{

if (flag == true)

{

textBox1.Clear();

label1.Text = "";

flag = false;

}

textBox1.Text = textBox1.Text + 3;

}

private void button9\_Click(object sender, EventArgs e)

{

if (flag == true)

{

textBox1.Clear();

label1.Text = "";

flag = false;

}

textBox1.Text = textBox1.Text + 4;

}

private void button10\_Click(object sender, EventArgs e)

{

if (flag == true)

{

textBox1.Clear();

label1.Text = "";

flag = false;

}

textBox1.Text = textBox1.Text + 5;

}

private void button11\_Click(object sender, EventArgs e)

{

if (flag == true)

{

textBox1.Clear();

label1.Text = "";

flag = false;

}

textBox1.Text = textBox1.Text + 6;

}

private void button5\_Click(object sender, EventArgs e)

{

if (flag == true)

{

textBox1.Clear();

label1.Text = "";

flag = false;

}

textBox1.Text = textBox1.Text + 7;

}

private void button6\_Click(object sender, EventArgs e)

{

if (flag == true)

{

textBox1.Clear();

label1.Text = "";

flag = false;

}

textBox1.Text = textBox1.Text + 8;

}

private void button7\_Click(object sender, EventArgs e)

{

if (flag == true)

{

textBox1.Clear();

label1.Text = "";

flag = false;

}

textBox1.Text = textBox1.Text + 9;

}

private void button4\_Click(object sender, EventArgs e)

{

if(label1.Text != "")

{

b = a + float.Parse(textBox1.Text);

a = b;

label1.Text = b.ToString();

}

else

a = float.Parse(textBox1.Text);

textBox1.Clear();

count = 1;

label1.Text = a.ToString() + "+";

znak = true;

}

private void button8\_Click(object sender, EventArgs e)

{

if (label1.Text != "")

{

b = a + float.Parse(textBox1.Text);

a = b;

label1.Text = b.ToString();

}

else

a = float.Parse(textBox1.Text);

textBox1.Clear();

count = 2;

label1.Text = a.ToString() + "-";

znak = true;

}

private void button12\_Click(object sender, EventArgs e)

{

if (label1.Text != "")

{

b = a + float.Parse(textBox1.Text);

a = b;

label1.Text = b.ToString();

}

else

a = float.Parse(textBox1.Text);

textBox1.Clear();

count = 3;

label1.Text = a.ToString() + "\*";

znak = true;

}

private void button16\_Click(object sender, EventArgs e)

{

if (label1.Text != "")

{

b = a + float.Parse(textBox1.Text);

a = b;

label1.Text = b.ToString();

}

else

a = float.Parse(textBox1.Text);

textBox1.Clear();

count = 4;

label1.Text = a.ToString() + "/";

znak = true;

}

private void button19\_Click(object sender, EventArgs e)

{

if (textBox1.Text != "" )

{

label1.Text += textBox1.Text + " =";

calculate();

flag = true;

}

}

private void button18\_Click(object sender, EventArgs e)

{

textBox1.Text = textBox1.Text + ",";

}

private void button1\_Click(object sender, EventArgs e)

{

textBox1.Text = "";

label1.Text = "";

}

private void button2\_Click(object sender, EventArgs e)

{

int lenght = textBox1.Text.Length - 1;

string text = textBox1.Text;

textBox1.Clear();

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

{

textBox1.Text = textBox1.Text + text[i];

}

}

private void button3\_Click(object sender, EventArgs e)

{

if (znak == true)

{

textBox1.Text = "-" + textBox1.Text;

znak = false;

}

else if (znak == false)

{

textBox1.Text = textBox1.Text.Replace("-", "");

znak = true;

}

}

private void timer1\_Tick(object sender, EventArgs e)

{

MessageBox.Show("Поддержите автора, купите лицензию");

}

private void лицензияToolStripMenuItem\_Click(object sender, EventArgs e)

{

Cod examp = new Cod(new MyDelegate(func));

examp.Show();

this.Hide();

}

private void историяСообщенийToolStripMenuItem\_Click(object sender, EventArgs e)

{

History examp = new History();

examp.Show();

this.Hide();

}

private void выходToolStripMenuItem\_Click(object sender, EventArgs e)

{

Auth examp = new Auth();

examp.Show();

this.Hide();

}

private void calculate()

{

switch (count)

{

case 1:

b = a + float.Parse(textBox1.Text);

string mes = label1.Text + b.ToString();

DB\_operation.WriteTask(Work\_sess.login,DateTime.Now, mes);

textBox1.Text = b.ToString();

break;

case 2:

b = a - float.Parse(textBox1.Text);

mes = label1.Text + b.ToString();

DB\_operation.WriteTask(Work\_sess.login, DateTime.Now, mes);

textBox1.Text = b.ToString();

break;

case 3:

b = a \* float.Parse(textBox1.Text);

mes = label1.Text + b.ToString();

DB\_operation.WriteTask(Work\_sess.login, DateTime.Now, mes);

textBox1.Text = b.ToString();

break;

case 4:

b = a / float.Parse(textBox1.Text);

mes = label1.Text + b.ToString();

DB\_operation.WriteTask(Work\_sess.login, DateTime.Now, mes);

textBox1.Text = b.ToString();

break;

default:

break;

}

}

}

}

**Auth.cs**

using System;

using System.Collections.Generic;

using System.ComponentModel;

using System.Data;

using System.Drawing;

using System.Linq;

using System.Text;

using System.Threading.Tasks;

using System.Windows.Forms;

namespace CaLtoSetgey

{

public partial class Auth : Form

{

public Auth()

{

InitializeComponent();

}

string Log = "";

public string demo(int i)

{

string s = "";

if (i == 0)

s = "демо версия";

if (i == 1)

s = "полная версия";

return s;

}

private void button1\_Click(object sender, EventArgs e)

{

Log = textBox1.Text;

if (Log == "")

MessageBox.Show("Введите логин.");

else

{

if (textBox2.Text == "")

MessageBox.Show("Введите пароль.");

else

{

if (DB\_operation.Authentication(Log, textBox2.Text))

{

Calc examp = new Calc();

examp.Show();

Hide();

}

else

{

MessageBox.Show("Доступ запрещён.\nНе верный логин или пароль.");

}

}

}

}

private void button2\_Click(object sender, EventArgs e)

{

Regist examp = new Regist();

examp.Show();

this.Hide();

}

}

}

**Cod.cs**

using System;

using System.Collections.Generic;

using System.ComponentModel;

using System.Data;

using System.Drawing;

using System.Linq;

using System.Text;

using System.Threading.Tasks;

using System.Windows.Forms;

using System.Net.Mail;

using System.Net;

namespace CaLtoSetgey

{

public partial class Cod : Form

{

private MyDelegate d;

public Cod(MyDelegate sender)

{

InitializeComponent();

d = sender;

}

private void Cod\_Load(object sender, EventArgs e)

{

var vers = Work\_sess.Access.ToString();

if (vers == "l\_0")

{

}

else

{

label1.Text = "Лицензия уже активировано";

label2.Visible = false;

label3.Visible = false;

textBox1.Visible = false;

textBox2.Visible = false;

button1.Visible = false;

button2.Visible = false;

}

}

private void button1\_Click(object sender, EventArgs e)

{

var msg = DB\_operation.SendCod(Work\_sess.login);

SmtpClient Smtp = new SmtpClient("smtp.yandex.ru", 25);

Smtp.EnableSsl = true;

Smtp.Credentials = new NetworkCredential("zeny35214@yandex.ru", "R@s55adR))t");

MailMessage Message = new MailMessage();

Message.From = new MailAddress("zeny35214@yandex.ru");

Message.To.Add(new MailAddress(textBox1.Text));

Message.Subject = "Код активации";

Message.Body = msg;

try

{

Smtp.Send(Message);

}

catch (Exception ex)

{

MessageBox.Show(ex.Message);

}

}

private void button2\_Click(object sender, EventArgs e)

{

if (DB\_operation.checkCod(Work\_sess.login, textBox2.Text))

{

Work\_sess.Access = (Param.demo\_level)1;

DialogResult result = MessageBox.Show("Успешно", "Активация прошла успешно !", MessageBoxButtons.OK, MessageBoxIcon.Information, MessageBoxDefaultButton.Button1, MessageBoxOptions.DefaultDesktopOnly);

if (result == DialogResult.OK)

{

d();

Calc examp = new Calc();

examp.Show();

this.Close();

}

}

else

MessageBox.Show("Код активации не верен.");

}

}

}

**Crypto.cs**

using System;

using System.Collections.Generic;

using System.Linq;

using System.Text;

using System.Threading.Tasks;

using System.IO;

using System.Security.Cryptography;

using System.Collections;

namespace CaLtoSetgey

{

class Crypto

{

public byte[] SHA\_512(byte[] data) // генератор хэша на основе SHA512

{

byte[] res = new byte[1];

SHA512 SHA = new SHA512Managed();

res = SHA.ComputeHash(data);

return res;

}

public byte[] SHA\_256(byte[] data) // генератор хэша на основе SHA512

{

byte[] res = new byte[1];

SHA256 SHA = new SHA256Managed();

res = SHA.ComputeHash(data);

return res;

}

}

}

**Db.cs**

using System;

using System.Collections.Generic;

using System.Linq;

using System.Text;

using System.Threading.Tasks;

using MySql.Data.MySqlClient;

namespace CaLtoSetgey

{

class DB

{

MySqlConnection connection = new MySqlConnection("server=localhost; port=3306; username=root; password=root; database=calc");

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;

}

}

}

**DB\_operations.cs**

using System;

using System.Collections.Generic;

using System.Linq;

using System.Text;

using System.Threading.Tasks;

using System.IO;

using System.Data.SqlClient;

using System.Security.Cryptography;

using System.ComponentModel;

using System.Data;

using MySql.Data.MySqlClient;

namespace CaLtoSetgey

{

class DB\_operation

{

static public bool check(string login, string pass)

{

DataTable table = new DataTable();

MySqlDataAdapter adapter = new MySqlDataAdapter();

DB db = new DB();

MySqlCommand cmd = new MySqlCommand("Select \* from `users` where `Login` = N'" + login + "' or `Password` = N'" + pass + "'", db.getconnection());

var result = true;

adapter.SelectCommand = cmd;

adapter.Fill(table);

if (table.Rows.Count > 0)

{

result = false;

}

return result;

}

static public void Regist(string login, string pass)

{

Crypto CR = new Crypto();

var hash = CR.SHA\_512(Encoding.Default.GetBytes(pass));

var hash2 = CR.SHA\_256(Encoding.Default.GetBytes(pass));

string s\_hash = null;

string s\_hash2 = null;

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

s\_hash += hash[i].ToString("x");

for (int i = 0; i < hash2.Length; i++)

s\_hash2 += hash2[i].ToString("x");

DB db = new DB();

MySqlCommand cmd = new MySqlCommand("insert into `users`(`Login`, `Password`, `Version`, `Sha512`, `Sha256`) values (N'" + login + "', N'" + pass + "', " + 0 + ", N'" + s\_hash + "', N'" + s\_hash2 + "' )", db.getconnection());

db.openconnection();

cmd.ExecuteNonQuery();

db.closeconnection();

}

static public bool Authentication(string login, string password) // функция аутентификации пользователя

{

DB db = new DB();

Crypto CR = new Crypto(); // экземпляр класса для шифрования

var hash = CR.SHA\_512(Encoding.Default.GetBytes(password)); // хеш введенного пароля, должен быть равен тому, что в бд

MySqlCommand cmd = new MySqlCommand("select `Sha512`, `Version` from `users` where `Login` = N'" + login + "'", db.getconnection());

DataTable table = new DataTable();

MySqlDataAdapter adapter = new MySqlDataAdapter();

adapter.SelectCommand = cmd;

adapter.Fill(table);

if (table.Rows.Count == 0)

return false;

DataRow row = table.Rows[0];

var trueHash = row.Field<string>("Sha512"); // берем строку из первого столбца результата

var version = row.Field<int>("Version"); // подключение больше не нужно

string s\_hash = "";

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

s\_hash += hash[i].ToString("x");

trueHash = trueHash.Trim(' ');

for (int i = 0; i < trueHash.Length; i++) // сравниваем пароли

{

if (trueHash[i] != s\_hash[i]) return false;

}

Work\_sess.Access = (Param.demo\_level)version; // задаём уровень доступа

Work\_sess.login = login; // задаём логин

return true;

}

static public void WriteTask(string login, DateTime time, string mes)

{

Sym\_alg SM = new Sym\_alg();

byte[] k = SM.key\_gen();

byte[] iv = SM.key\_gen();

byte[] b\_mess = Encoding.Default.GetBytes(mes);

byte[] encypt = SM.AES\_Encrypt(b\_mess, k, iv);

DB db = new DB();

MySqlCommand cmd = new MySqlCommand("INSERT INTO `history`(`login`, `Date`, `Mes`, `Key`, `IV`) VALUES(N'"+ login +"', N'"+time.ToString()+"',@msg, @key, @iv)", db.getconnection());

cmd.Parameters.Add("@msg", MySqlDbType.VarChar).Value = Encoding.Default.GetString(encypt);

cmd.Parameters.Add("@key", MySqlDbType.VarChar).Value = Encoding.Default.GetString(k);

cmd.Parameters.Add("@iv", MySqlDbType.VarChar).Value = Encoding.Default.GetString(iv);

db.openconnection();

cmd.ExecuteNonQuery();

db.closeconnection();

}

static public string SendCod(string login)

{

string res = null;

int id = 0;

DB db = new DB();

MySqlCommand cmd = new MySqlCommand("SELECT \* FROM `regcod` AS `r1` JOIN(SELECT(RAND() \* (SELECT MAX(`id`) - MIN(`id`) FROM `regcod` where `act` = 0) + (SELECT MIN(`id`) FROM `regcod` where `act` = 0)) AS `id`) AS `r2` WHERE `act` = 0 and `r1`.`id` >= `r2`.`id` ORDER BY `r1`.`id` ASC LIMIT 1", db.getconnection());

DataTable table = new DataTable();

MySqlDataAdapter adapter = new MySqlDataAdapter();

adapter.SelectCommand = cmd;

adapter.Fill(table);

id = table.Rows[0].Field<int>("Id");

res = table.Rows[0].Field<string>("Registcod");

byte[] Key = Encoding.Default.GetBytes(res);

Crypto hash = new Crypto();

byte[] hash\_arr = new byte[1];

hash\_arr = hash.SHA\_256(Key);

string hash\_arr1 = Encoding.Default.GetString(hash\_arr);

MemoryStream mStream = new MemoryStream();

Aes AES = Aes.Create(); // объект класса AES

AES.Mode = CipherMode.CFB; // включаепм режим CFB, чтобы установить длину шифротекста в 64 бита

byte[] key = AES.Key;

byte[] IV = AES.IV;

byte[] haskey = new byte[hash\_arr.Length + key.Length];

Buffer.BlockCopy(hash\_arr, 0, haskey, 0, hash\_arr.Length);

Buffer.BlockCopy(key, 0, haskey, hash\_arr.Length, key.Length);

CryptoStream cStream = new CryptoStream(mStream,

AES.CreateEncryptor(hash\_arr, IV),

CryptoStreamMode.Write); // создаём поток для шифрования

byte[] toEncrypt = haskey;

cStream.Write(toEncrypt, 0, toEncrypt.Length); // помещаем данные в поток шифрования

cStream.FlushFinalBlock();

byte[] ret = mStream.ToArray(); // извлекаем шифротекст

cStream.Close(); // закрываем потоки

mStream.Close();

cmd = new MySqlCommand("update `regcod` set `Secretcod` = @msg, `Key` = @key, `IV` = @iv, `Hash` = @hash where `Id` = " + id + "", db.getconnection());

cmd.Parameters.Add("@msg", MySqlDbType.VarChar).Value = Encoding.Default.GetString(ret);

cmd.Parameters.Add("@key", MySqlDbType.VarChar).Value = Encoding.Default.GetString(key);

cmd.Parameters.Add("@iv", MySqlDbType.VarChar).Value = Encoding.Default.GetString(IV);

cmd.Parameters.Add("@hash", MySqlDbType.VarChar).Value = Encoding.Default.GetString(hash\_arr);

db.openconnection();

cmd.ExecuteNonQuery();

db.closeconnection();

return res;

}

static public bool checkCod(string login, string cod)

{

DB db = new DB();

MySqlCommand cmd = new MySqlCommand("select count(\*) from `regcod` where `Act` = " + 0 + "", db.getconnection());

db.openconnection();

object a = cmd.ExecuteScalar();

db.closeconnection();

byte[,] data = new byte[Convert.ToInt32(a), 4];

cmd = new MySqlCommand("select `Id`, `Secretcod`, `IV`, `Key`, `Hash` from regcod where `Act` = " + 0 + "", db.getconnection());

DataTable table = new DataTable();

MySqlDataAdapter adapter = new MySqlDataAdapter();

adapter.SelectCommand = cmd;

adapter.Fill(table);

byte[] Key = Encoding.Default.GetBytes(cod);

Crypto hash = new Crypto();

byte[] hash\_arr = new byte[1];

hash\_arr = hash.SHA\_256(Key);

for (int i = 0; i < table.Rows.Count; i++)

{

DataRow row = table.Rows[i];

byte[] key = Encoding.Default.GetBytes(row.Field<string>("Key"));

byte[] iv = Encoding.Default.GetBytes(row.Field<string>("IV"));

byte[] secretcod = Encoding.Default.GetBytes(row.Field<string>("Secretcod"));

byte[] tryehash = Encoding.Default.GetBytes(row.Field<string>("Hash"));

MemoryStream mStream = new MemoryStream(); // создаём поток памяти

Aes AES = Aes.Create(); // объект класса 3DES

AES.Mode = CipherMode.CFB; // включаепм режим CFB, чтобы установить длину шифротекста в 64 бита

CryptoStream cStream = new CryptoStream(mStream,

AES.CreateDecryptor(tryehash, iv),

CryptoStreamMode.Write); // создаём поток для дешифрования

byte[] toEncrypt = secretcod;

cStream.Write(toEncrypt, 0, toEncrypt.Length); // помещаем данные в поток дешифрования

cStream.FlushFinalBlock();

byte[] ret = mStream.ToArray(); // извлекаем текст

cStream.Close(); // закрываем потоки

mStream.Close();

int lenght = ret.Length - key.Length;

int flag = 0;

for (int k = 0; k < lenght; k++)

{

if (ret[k] != hash\_arr[k])

{

flag = 0;

break;

}

flag = 1;

}

if (flag == 1)

{

cmd = new MySqlCommand("update `regcod` set `Act` = 1 where `Id` = " + table.Rows[i].Field<int>("id") + "", db.getconnection());

db.openconnection();

cmd.ExecuteNonQuery();

db.closeconnection();

cmd = new MySqlCommand("update `users` set `Version` = 1 where `Login` = " + login + "", db.getconnection());

db.openconnection();

cmd.ExecuteNonQuery();

db.closeconnection();

return true;

}

}

return false;

}

static public DataTable Edit\_Table(string login)

{

Sym\_alg SM = new Sym\_alg();

DB db = new DB();

MySqlCommand cmd = new MySqlCommand("SELECT \* FROM `history` where `login` = N'" + login + "'", db.getconnection());

DataTable table = new DataTable();

MySqlDataAdapter adapter = new MySqlDataAdapter();

adapter.SelectCommand = cmd;

adapter.Fill(table);

for (int i = 0; i < table.Rows.Count; i++)

{

byte[] key = Encoding.Default.GetBytes(table.Rows[i].Field<string>("key"));

byte[] iv = Encoding.Default.GetBytes(table.Rows[i].Field<string>("iv"));

byte[] mes = Encoding.Default.GetBytes(table.Rows[i].Field<string>("mes"));

byte[] decypt = SM.AES\_Decrypt(mes, key, iv);

string dec\_res1 = Encoding.Default.GetString(decypt, 0, decypt.Length);

table.Rows[i]["mes"] = dec\_res1;

}

return table;

}

}

}

**History.cs**

using System;

using System.Collections.Generic;

using System.ComponentModel;

using System.Data;

using System.Drawing;

using System.Linq;

using System.Text;

using System.Threading.Tasks;

using System.Windows.Forms;

namespace CaLtoSetgey

{

public partial class History : Form

{

public History()

{

InitializeComponent();

}

private void History\_FormClosing(object sender, FormClosingEventArgs e)

{

dataGridView1.Rows.Clear();

Calc examp = new Calc();

examp.Show();

}

private void History\_Load(object sender, EventArgs e)

{

DataTable table = DB\_operation.Edit\_Table(Work\_sess.login);

for (int i = 0; i < table.Rows.Count; i++)

{

dataGridView1.Rows.Add(table.Rows[i].Field<int>("id"), table.Rows[i].Field<string>("date"), table.Rows[i].Field<string>("mes"));

}

}

}

}

**Param.cs**

using System;

using System.Collections.Generic;

using System.Linq;

using System.Text;

using System.Threading.Tasks;

namespace CaLtoSetgey

{

public static class Param

{

public enum demo\_level { l\_0, l\_1}; // тип данных для уровней допуска

}

}

**Program.cs**

using System;

using System.Collections.Generic;

using System.Linq;

using System.Threading.Tasks;

using System.Windows.Forms;

namespace CaLtoSetgey

{

static class Program

{

/// <summary>

/// Главная точка входа для приложения.

/// </summary>

[STAThread]

static void Main()

{

Application.EnableVisualStyles();

Application.SetCompatibleTextRenderingDefault(false);

Application.Run(new Auth());

}

}

public delegate void MyDelegate();

}

**Regist.cs**

using System;

using System.Collections.Generic;

using System.ComponentModel;

using System.Data;

using System.Drawing;

using System.Linq;

using System.Text;

using System.Threading.Tasks;

using System.Windows.Forms;

namespace CaLtoSetgey

{

public partial class Regist : Form

{

public Regist()

{

InitializeComponent();

}

private void выходToolStripMenuItem\_Click(object sender, EventArgs e)

{

Auth examp = new Auth();

examp.Show();

Close();

}

private void button1\_Click(object sender, EventArgs e)

{

if (textBox1.Text == "")

MessageBox.Show("Введите логин.");

else

{

if (textBox2.Text == "")

MessageBox.Show("Введите пароль.");

else

{

if (DB\_operation.check(textBox1.Text, textBox2.Text) == true)

{

DB\_operation.Regist(textBox1.Text, textBox2.Text);

MessageBox.Show("Регистрация успешно завершена.");

Auth examp = new Auth();

examp.Show();

Close();

}

else

MessageBox.Show("Такой логин или пароль уже существует.");

}

}

}

}

}

**Sum\_alg.cs**

using System;

using System.Collections.Generic;

using System.Linq;

using System.Text;

using System.Threading.Tasks;

using System.IO;

using System.Security.Cryptography;

using System.Collections;

using System.Threading;

namespace CaLtoSetgey

{

class Sym\_alg

{

public byte[] key\_gen() // генератор 128-битного ключа

{

byte[] key = new byte[256]; // основной ключ

byte[] b1 = new byte[128]; // 64-битная часть ключа

byte[] b2 = new byte[128]; // 64-битная часть ключа

byte[] xor = new byte[128]; // 64-битная блок случайных бит для равномерного преобразования ключа

byte[] bit = new byte[1]; // случайный бит для выбора совмещения двух частей ключа (1 - !b1 + b2, 0 - b2 + !b1)

Thread.Sleep(20); // тормозим генерацию ключа, чтобы новый ключ был случайным

var rand = new Random((int)DateTime.Now.Ticks & 0x0000FFFF);

rand.NextBytes(b1); // создаём случайную половину ключа

rand.NextBytes(b2); // создаём случайную половину ключа

rand.NextBytes(xor); // создаём случайную последовательность для оперции xor с половинами ключа

ulong k1 = BitConverter.ToUInt64(b1, 0); // переводим в ulong формат

ulong k2 = BitConverter.ToUInt64(b2, 0); // переводим в ulong формат

ulong xr = BitConverter.ToUInt64(xor, 0); // переводим в ulong формат

k1 ^= xr; // приводим к равномерному виду

xr = BitConverter.ToUInt64(xor, 0);

k2 ^= xr; // приводим к равномерному виду

rand.NextBytes(bit); // создаём случайное число (0 или 1), чтобы выбрать, какую половину ключа поставить первой и какую инвертировать

if (bit[0] == 1)

{

b1 = BitConverter.GetBytes(~k1); // обрачаем число

b2 = BitConverter.GetBytes(k2);

key = b1.Concat(b2).ToArray(); // соединяем половины 128-битного ключа

}

else

{

b1 = BitConverter.GetBytes(k1);

b2 = BitConverter.GetBytes(~k2); // обрачаем число

key = b2.Concat(b1).ToArray(); // соединяем половины 128-битного ключа

}

return key;

}

public byte[] AES\_Encrypt(byte[] Data, byte[] Key, byte[] IV) // AES шифратор

{

MemoryStream mStream = new MemoryStream(); // создаём поток памяти

Aes AES = Aes.Create(); // объект класса AES

AES.Mode = CipherMode.CFB; // включаепм режим CFB, чтобы установить длину шифротекста в 64 бита

CryptoStream cStream = new CryptoStream(mStream,

AES.CreateEncryptor(Key, IV),

CryptoStreamMode.Write); // создаём поток для шифрования

byte[] toEncrypt = Data;

cStream.Write(toEncrypt, 0, toEncrypt.Length); // помещаем данные в поток шифрования

cStream.FlushFinalBlock();

byte[] ret = mStream.ToArray(); // извлекаем шифротекст

cStream.Close(); // закрываем потоки

mStream.Close();

return ret;

}

public byte[] AES\_Encrypt1(byte[] Data, byte[] Key, byte[] IV) // AES шифратор

{

MemoryStream mStream = new MemoryStream(); // создаём поток памяти

Aes AES = Aes.Create(); // объект класса AES

AES.Mode = CipherMode.CFB; // включаепм режим CFB, чтобы установить длину шифротекста в 64 бита

byte[] key = AES.Key;

byte[] haskey = new byte[Key.Length + key.Length];

Buffer.BlockCopy(Key, 0, haskey, 0, Key.Length);

Buffer.BlockCopy(key, 0, haskey, Key.Length, key.Length);

CryptoStream cStream = new CryptoStream(mStream,

AES.CreateEncryptor(Key, IV),

CryptoStreamMode.Write); // создаём поток для шифрования

byte[] toEncrypt = haskey;

cStream.Write(toEncrypt, 0, toEncrypt.Length); // помещаем данные в поток шифрования

cStream.FlushFinalBlock();

byte[] ret = mStream.ToArray(); // извлекаем шифротекст

cStream.Close(); // закрываем потоки

mStream.Close();

return ret;

}

public byte[] AES\_Decrypt(byte[] Data, byte[] Key, byte[] IV) // AES дешифратор

{

MemoryStream mStream = new MemoryStream(); // создаём поток памяти

Aes AES = Aes.Create(); // объект класса 3DES

AES.Mode = CipherMode.CFB; // включаепм режим CFB, чтобы установить длину шифротекста в 64 бита

CryptoStream cStream = new CryptoStream(mStream,

AES.CreateDecryptor(Key, IV),

CryptoStreamMode.Write); // создаём поток для дешифрования

byte[] toEncrypt = Data;

cStream.Write(toEncrypt, 0, toEncrypt.Length); // помещаем данные в поток дешифрования

cStream.FlushFinalBlock();

byte[] ret = mStream.ToArray(); // извлекаем текст

cStream.Close(); // закрываем потоки

mStream.Close();

return ret;

}

}

}

**Work\_sess.cs**

using System;

using System.Collections.Generic;

using System.Linq;

using System.Text;

using System.Threading.Tasks;

namespace CaLtoSetgey

{

public static class Work\_sess

{

public static string login; //логин

public static string name; //имя

public static string s\_name; //фамилия

public static string pos; //должность

public static Param.demo\_level Access; //уровень доступа

}

}

1. **Тестирование:**

