ДОДАТОК А.
Код використаних алгоритмів та програм

Код генерування зображення першого типу:

```
private static uint GetBytePart(uint i, int byteIndex)
{
  return ((i >> (8 * byteIndex)) % 256 + 256) % 256;
}

public static Color GetColor(uint i)
{
  float r = GetBytePart(i, 0) / 255f;
  float g = GetBytePart(i, 1) / 255f;
  float b = GetBytePart(i, 2) / 255f;
  return new Color(r, g, b);
}
```

Код генерування зображення другого типу:

```
var max = 0;
for (var i = 0; i < ints.Length; i += 2)
{
  var x = GetBytePart(ints[i], ByteIndex);
  var y = GetBytePart(ints[i + 1], ByteIndex);
  var value = coords[x, y];
  value++;
  max = Mathf.Max(value, max);
  coords[x, y] = value;
}
```

Код лінійного конгруентного генератору

```
#define RAND_MAX 32767

static unsigned long int next = 1;
```

```
int rand(void)
{
  next = next * 1103515245 + 12345;
  return (unsigned int)(next/65536) % (RAND_MAX + 1);
}

void srand(unsigned int seed)
{
  next = seed;
}
```

Код генератора на основі Вихору Мерсена

```
ulong x;
 if (mti >= NN)
 {
  // generate NN words at one time
  for (var i = 0; i < NN - MM; i++)
  {
   x = (mt[i] & UM) | (mt[i + 1] & LM);
   mt[i] = mt[i + MM]
         ^ (x >> 1) ^ MAG01[(int) (x & 0x1L)];
  }
  for (var i = NN - MM; i < NN - 1; i++)
  {
   x = (mt[i] & UM) | (mt[i + 1] & LM);
   mt[i] = mt[i + (MM - NN)]
         ^ (x >> 1) ^ MAG01[(int) (x & 0x1L)];
```

```
 }
 x = (mt[NN - 1] & UM) | (mt[0] & LM);
 mt[NN - 1] = mt[MM - 1]
            ^ (x >> 1) ^ MAG01[(int) (x & 0x1L)];
 mti = 0;
}
x = mt[mti++];
x ^= (x >> 29) & 0x5555555555555555L;
x ^= (x << 17) & 0x71d67fffeda60000L;
x ^= (x << 37) & 0xfff7eee000000000L;
x ^= x >> 43;
return x;
```

Код генератору XorShift

```
namespace xorShift
{
   class Program
   {
      static void Main(string[] args)
      {
         Random rd = new Random();
         long x = (long)((r.NextDouble() * 0.5d) * long.MaxValue);
         Random rd1 = new Random();
         long y = (long)((r.NextDouble() * 0.5d) * long.MaxValue);
         Random rd2 = new Random();
         long z = (long)((r.NextDouble() * 0.5d) * long.MaxValue);
         Random rd3 = new Random();
         long w = (long)((r.NextDouble() * 0.5d) * long.MaxValue);
```

```
        public static uint x , y , z , w ;
        public static uint xorShift()
        {
            uint t = x ^ (x << 11);
            x = y; y = z; z = w;
            return w = w ^ (w >> 19) ^ t ^ (t >> 6);
        }
    }
  }
}
```

Приклад використання вбудованої хеш-функції MD5 у мові C# як генератора псевдовипадкових чисел.

```
var hash = new Hash(0);
var rn0 = hash.GetHash(0);
var rn1 = hash.GetHash(1);
var rn2 = hash.GetHash(12);
var rn3 = hash.GetHash(13, 5);

var rn4 = Hash.GetHash(0, 0);
var rn5 = Hash.GetHash(0, 1);
var rn6 = Hash.GetHash(0, 12);
var rn7 = Hash.GetHash(0, 13, 5);
```

ДОДАТОК Б.
Вихідний код розробленого прототипу

Класс, що описує з’єднання з базою даних:

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

namespace PassGenerator
{
   internal class DB
   {
      MySqlConnection connection = new MySqlConnection("server=localhost;port=3306;username=;password=;database=");

      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;
        }
    }
}
```

Класс, що описує користувача:

```
using System;
using System.IO;
using System.Collections.Generic;
using System.Linq;
using System.Text;
using System.Threading.Tasks;

namespace DiplomaPassGenerator
{
    internal class User
    {
        public static string username;
        private static string hashPassword;
        public static string uFile;
        public static byte[] key;
```

```
public User()
{
    User.username = null;
    User.hashPassword = null;
    User.uFile = null;
    User.key = null;
}

public User(string username, string hashPassword, string uFile, byte[] key)
{
    User.username = username;
    User.hashPassword = hashPassword;
    User.uFile = uFile;
    User.key = key;
}

public bool generateFile()
{
    uFile = uFile + ".txt";

    try
    {
        File.Create(uFile).Close();
        return true;
    }
    catch
    {
        return false;
```

```
            }
        }
    }
}
```

Класс, що містить алгоритм генератора:

```
using Standart.Hash.xxHash;
using System;
using System.Collections.Generic;
using System.Linq;
using System.Text;
using System.Threading.Tasks;
using System.Security.Cryptography;

namespace DiplomaPassGenerator
{

    class Generator
    {
        private string seed;
        private int counter;

        public Generator()
        {
            DateTime date = DateTime.Now;
            this.seed = date.ToString("dd/MM/yyyy HH:mm:ss");
            counter = 0;
```

```
}
public int Next()
{
    MD5 md5 = new MD5CryptoServiceProvider();
    byte[] hash = Encoding.ASCII.GetBytes(seed + (++counter));
    return BitConverter.ToInt32(md5.ComputeHash(hash),0);
}

public int Next(int top)
{
    MD5 md5 = new MD5CryptoServiceProvider();
    byte[] hash = Encoding.ASCII.GetBytes(seed + (++counter));
    byte[] hashenc = md5.ComputeHash(hash);
    int res = (BitConverter.ToInt32(hashenc, 0) % top);
    if (res < 0)
    {
        return res * -1;
    }
    else
    {
        return res;
    }

}

public int Next(int button, int top)
{
    MD5 md5 = new MD5CryptoServiceProvider();
```

```
            byte[] hash = Encoding.ASCII.GetBytes(seed + (++counter));
            byte[] hashenc = md5.ComputeHash(hash);
            return (button + (BitConverter.ToInt32(hashenc, 0) % top) % (top - button
+ 1));
        }
    }
}
```

Клас, що описує алгоритм шифрування файлу:

```
using System;
using System.Collections.Generic;
using System.Linq;
using System.Text;
using System.Threading.Tasks;
using System.IO;
using System.Security.Cryptography;

namespace DiplomaPassGenerator
{
    internal class FileEncrypt
    {
        public void EncryptFile(string path, byte[] key)
        {
            string tmpPath = Path.GetTempFileName();
            using (FileStream fsSrc = File.OpenRead(path))
            using (AesManaged aes = new AesManaged() { Key = key })
            using (FileStream fsDst = File.Create(tmpPath))
            {
```

```
            fsDst.Write(aes.IV,0,aes.IV.Length);
            using (CryptoStream cs = new CryptoStream(fsDst, aes.CreateEncryptor(), CryptoStreamMode.Write, true))
            {
                fsSrc.CopyTo(cs);
            }
        }
        File.Delete(path);
        File.Move(tmpPath, path);
    }

    public void DecryptFile(string path, byte[] key)
    {
        string tmpPath = Path.GetTempFileName();
        using (FileStream fsSrc = File.OpenRead(path))
        {
            byte[] iv = new byte[16];
            fsSrc.Read(iv, 0, iv.Length);
            using (AesManaged aes = new AesManaged() { Key = key, IV = iv })
            using (CryptoStream cs = new CryptoStream(fsSrc, aes.CreateDecryptor(), CryptoStreamMode.Read, true))
            using (FileStream fsDst = File.Create(tmpPath))
            {
                cs.CopyTo(fsDst);
            }
        }
        File.Delete(path);
        File.Move(tmpPath, path);
    }
```

```
    }
}
```

Клас, що описує логіку під час реєстрації користувача, створення файлу, його заповнення та шифрування

```
using MySql.Data.MySqlClient;
using PassGenerator;
using System;
using System.Collections.Generic;
using System.ComponentModel;
using System.Data;
using System.Drawing;
using System.IO;
using System.Linq;
using System.Text;
using System.Threading.Tasks;
using System.Windows.Forms;

namespace DiplomaPassGenerator
{
    public partial class frmRegister : Form
    {
        public frmRegister()
        {
            InitializeComponent();
        }

        private void registerBtn_Click(object sender, EventArgs e)
```

```
        {

            if (isUserExists())
                return;

            if (txtUsername.Text == "" && txtPassword.Text == "" && txtComPassword.Text == "")
            {
                MessageBox.Show("Заповніть усі поля форми", "Реєстрація невдала", MessageBoxButtons.OK, MessageBoxIcon.Error);
            }
            else if (txtPassword.Text == txtComPassword.Text)
            {
                DB db = new DB();
                MySqlCommand command = new MySqlCommand("INSERT INTO `users` (`login`, `pass`, `kee`) VALUES (@login, @pass, @key)", db.getConnection());

                md5 md = new md5();
                string password = md.hashPassword(txtPassword.Text);
                byte[] key = Enumerable.Range(0, 32).Select(x => (byte)x).ToArray();

                command.Parameters.Add("@login", MySqlDbType.VarChar).Value = txtUsername.Text;
                command.Parameters.Add("@pass", MySqlDbType.VarChar).Value = password;
                command.Parameters.Add("@key", MySqlDbType.VarChar).Value = Encoding.ASCII.GetString(key);
                User user = new User(txtUsername.Text, txtPassword.Text, txtUsername.Text, key);
```

```
                FileEncrypt fileEncrypt = new FileEncrypt();




                db.openConnection();

                if(user.generateFile() && command.ExecuteNonQuery() == 1)
                {
                    fileFill();
                    fileEncrypt.EncryptFile(User.uFile, User.key);

                    MessageBox.Show("Аккаунт зареєстровано", "Успіх",
MessageBoxButtons.OK, MessageBoxIcon.Information);

                    txtUsername.Text = "";
                    txtPassword.Text = "";
                    txtComPassword.Text = "";
                }
                else
                {
                    MessageBox.Show("Помилка при реєстрації", "Реєстрація
невдала", MessageBoxButtons.OK, MessageBoxIcon.Error);
                }

                db.closeConnection();
            }
            else
            {
```

```
            MessageBox.Show("Паролі не співпадають", "Реєстрація невдала", MessageBoxButtons.OK, MessageBoxIcon.Error);
            txtPassword.Text = "";
            txtComPassword.Text = "";
            txtPassword.Focus();
        }
    }

    public bool isUserExists()
    {
        DB db = new DB();
        DataTable table = new DataTable();
        MySqlDataAdapter adapter = new MySqlDataAdapter();
        MySqlCommand command = new MySqlCommand("SELECT * FROM `users` WHERE `login` = @uL", db.getConnection());
        command.Parameters.Add("@uL", MySqlDbType.VarChar).Value = txtUsername;

        adapter.SelectCommand = command;
        adapter.Fill(table);

        if (table.Rows.Count > 0)
        {
            MessageBox.Show("Такий користувач вже існує", "Реєстрація невдала", MessageBoxButtons.OK, MessageBoxIcon.Error);
            return true;
        }
        else
        {
            return false;
```

```
        }
    }

    private void checkbxShowPas_CheckedChanged(object sender,
EventArgs e)
    {
        if (checkbxShowPas.Checked)
        {
            txtPassword.PasswordChar = '\0';
            txtComPassword.PasswordChar = '\0';
        }
        else
        {
            txtPassword.PasswordChar = '*';
            txtComPassword.PasswordChar = '*';
        }
    }

    private void button1_Click(object sender, EventArgs e)
    {
        txtUsername.Text = "";
        txtPassword.Text = "";
        txtComPassword.Text = "";
        txtUsername.Focus();
    }

    private void label6_Click(object sender, EventArgs e)
    {
        new frmLogin().Show();
```

```
            this.Hide();
        }

        private void fileFill()
        {
            Generator gen = new Generator();
            const string chars = "0123456789abcdefghijklmnopqrstuvwxyzABCDEFGHIJKLMNOPQRSTUVWXYZ";
            using (FileStream fstream = new FileStream(User.uFile, FileMode.Append))
            {
                using (StreamWriter stream = new StreamWriter(fstream))
                {
                    for (int i = 0; i < 1000000; i++)
                    {
                        int index = gen.Next(chars.Length);
                        stream.Write(chars[index]);
                    }
                }
            }
        }

        private void frmRegister_FormClosing(object sender, FormClosingEventArgs e)
        {
            DialogResult dialog = MessageBox.Show("Вы действительно хотите выйти из программы?", "Завершение программы", MessageBoxButtons.YesNo, MessageBoxIcon.Warning);
            if (dialog == DialogResult.Yes)
```

```
            {
                Environment.Exit(0);
                e.Cancel = false;
            }
            else
            {
                e.Cancel = true;
            }
        }
    }
}
```

Класс, що описує алгоритм під час реєстрації:

```
using MySql.Data.MySqlClient;
using PassGenerator;
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 DiplomaPassGenerator
{
```

```
public partial class frmLogin : Form
{
   public frmLogin()
   {
      InitializeComponent();
   }

   private void registerBtn_Click(object sender, EventArgs e)
   {


      DB db = new DB();
      DataTable table = new DataTable();
      MySqlDataAdapter adapter = new MySqlDataAdapter();
      MySqlCommand command = new MySqlCommand("SELECT * FROM `users` WHERE `login` = @uL AND `pass` = @uP", db.getConnection());

      md5 md = new md5();
      string password = md.hashPassword(txtPassword.Text);

      command.Parameters.Add("@uL", MySqlDbType.VarChar).Value = txtUsername.Text;
      command.Parameters.Add("@uP", MySqlDbType.VarChar).Value = password;
      User.username = txtUsername.Text;
      User.uFile = txtUsername.Text + ".txt";


      adapter.SelectCommand = command;
```

```
            adapter.Fill(table);

            if (table.Rows.Count > 0)
            {
                FileEncrypt fileEncrypt = new FileEncrypt();
                string key = table.Rows[0].Field<string>("kee");
                User.key = Encoding.ASCII.GetBytes(key);
                fileEncrypt.DecryptFile(User.uFile, User.key);
                new MainWindow().Show();
                this.Hide();
            }
            else
            {
                MessageBox.Show("Невірний пароль або ім'я користувача, будь ласка спробуйте знову", "Авторизація невдала", MessageBoxButtons.OK, MessageBoxIcon.Error);
                txtUsername.Text = "";
                txtPassword.Text = "";
                txtUsername.Focus();
            }
        }

        private void checkbxShowPas_CheckedChanged(object sender, EventArgs e)
        {
            if (checkbxShowPas.Checked)
            {
                txtPassword.PasswordChar = '\0';
            }
```

```
            else
            {
                txtPassword.PasswordChar = '*';
            }
        }

        private void button1_Click(object sender, EventArgs e)
        {
            txtUsername.Text = "";
            txtPassword.Text = "";
            txtUsername.Focus();
        }

        private void label6_Click(object sender, EventArgs e)
        {
            new frmRegister().Show();
            this.Hide();
        }

        private void frmLogin_FormClosing(object sender,
FormClosingEventArgs e)
        {
            DialogResult dialog = MessageBox.Show("Вы дійсно хочете вийти
з програми?", "Завершення програми", MessageBoxButtons.YesNo,
MessageBoxIcon.Warning);
            if (dialog == DialogResult.Yes)
            {
                Environment.Exit(0);
                e.Cancel = false;
```

```
            }
            else
            {
                e.Cancel = true;
            }
        }
    }
}
```

Класс, що описує роботу основного вікна

```
using System;
using System.Collections.Generic;
using System.ComponentModel;
using System.Data;
using System.Drawing;
using System.IO;
using System.Linq;
using System.Text;
using System.Threading.Tasks;
using System.Windows.Forms;
using Standart.Hash.xxHash;

namespace DiplomaPassGenerator
{

    public partial class MainWindow : Form
    {
```

```
        public MainWindow()
        {
            InitializeComponent();

            ToolStripMenuItem copyMenuItem = new ToolStripMenuItem("Копіювати");

            contextMenuStrip1.Items.AddRange(new[] { copyMenuItem });

            listPasswords.ContextMenuStrip = contextMenuStrip1;

            copyMenuItem.Click += copyMenuItem_Click;
        }

        void copyMenuItem_Click(object sender, EventArgs e)
        {
            Clipboard.SetText((string)listPasswords.SelectedItem);
        }

        private void проПрограмуToolStripMenuItem_Click(object sender, EventArgs e)
        {
            new AboutBox1().Show();

        }

        private void clearBtn_Click(object sender, EventArgs e)
        {
            firstElementField.Text = "";
```

```
            lastElementField.Text = "";
            listPasswords.Items.Clear();
        }


        static Generator gen = new Generator();


        private void generatePassBtn_Click(object sender, EventArgs e)
        {
            int start = Int32.Parse(firstElementField.Text);
            int end = Int32.Parse(lastElementField.Text);
            using (FileStream fstream = new FileStream(User.uFile, FileMode.Open))
            {
                byte[] output = new byte[end-start];
                fstream.Seek(start, SeekOrigin.Begin);
                fstream.Read(output, 0, output.Length);
                string textFromFile = Encoding.Default.GetString(output);
                listPasswords.Items.Insert(0, textFromFile);
            }
        }


        private void MainWindow_FormClosing(object sender, FormClosingEventArgs e)
        {
            DialogResult dialog = MessageBox.Show("Вы действительно хотите выйти из программы?", "Завершение программы", MessageBoxButtons.YesNo, MessageBoxIcon.Warning);
            if (dialog == DialogResult.Yes)
            {
```

```
                FileEncrypt fileEncrypt = new FileEncrypt();
                fileEncrypt.EncryptFile(User.uFile, User.key);
                Environment.Exit(0);
                e.Cancel = false;
            }
            else
            {
                e.Cancel = true;
            }
        }
    }
}
```

ДОДАТОК В.
Комплект графічних матеріалів

Прототипування генератора псевдовипадкових чисел для систем парольного захисту

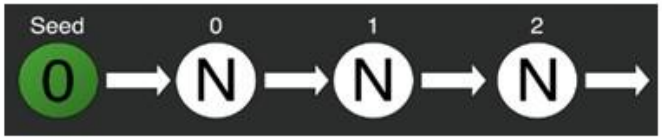
Пароль – умовне слово або комбінація символів, що складається з букв, цифр та інших символів, призначений для підтвердження особи або повноважень. Паролі використовують для захисту інформації від несанкціонованого доступу. У більшості обчислювальних систем комбінація "ім'я користувача - пароль" використовується для посвідчення користувача.

Для полегшення роботи користувачів з ними було створено системи зберігання та генерації паролів. Основою системи генерації паролів є алгоритми генерування псевдовипадкових чисел або алгоритми генерування випадкових чисел за заданими користувачем параметрами. Системи зберігання паролів зазвичай використовують бази даних у яких зберігаються хеші паролів або вони шифруються за допомогою інших алгоритмів.

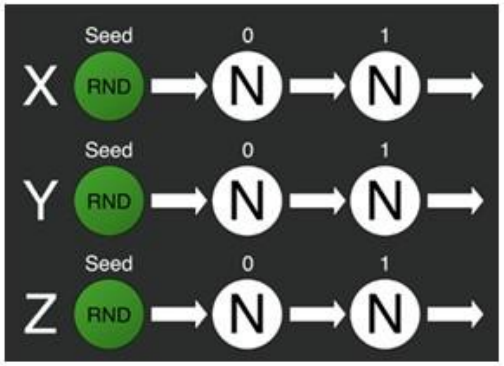
У даній роботі розглянуто використання таких систем та алгоритмів, які вони використовують, їх недоліки та переваги, проведено порівняння різних алгоритмів.

На основі проведених досліджень було розроблено функціонуючий прототип з генератором псевдовипадкових чисел на основі хеш – функції, що пропонує нову концепцію для програм – генераторів паролів

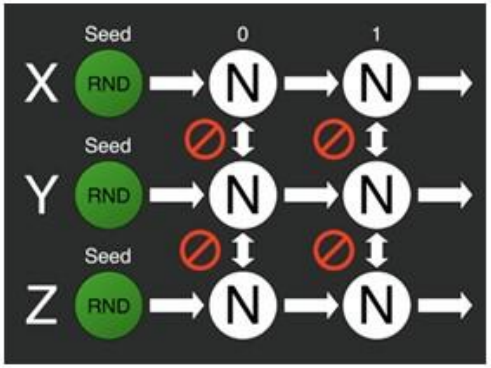
Принцип роботи генератора псевдовипадкових чисел



а)



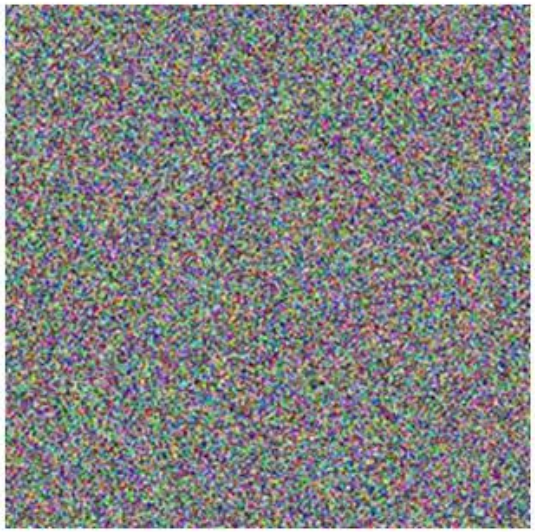
б)

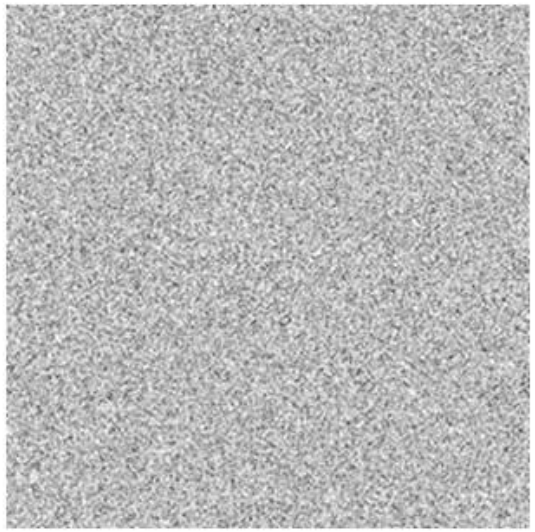


в)

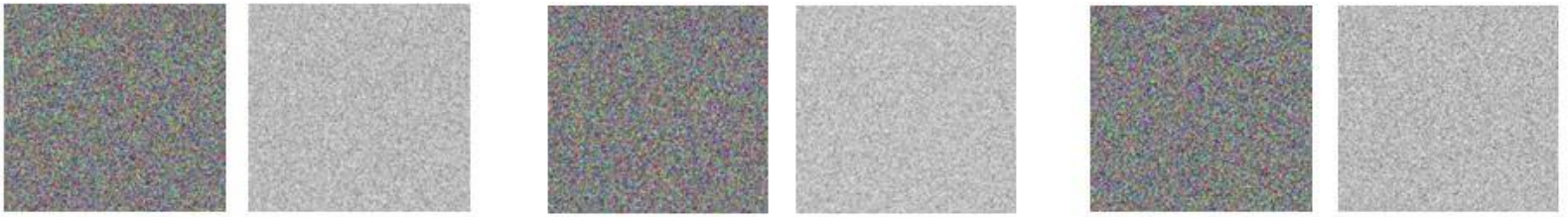
а) Схема ініціалізації генератора
б) Схема ініціалізаці та розділення генератора
в) Проблема розділення генератору

Алгоритм оцінки якості генерації алгоритмів





Результати оцінки алгоритмів генераторів псевдовипадкових чисел

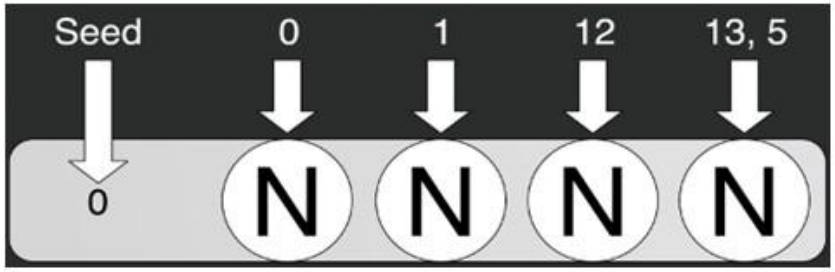


1.1 1.2 2.1 2.2 3.1 3.2

	$0 \dots n$	$0\ seed\ 0 \dots n$
ЛКГ	10	28
Вихор Мерсенна	11	1870
XorShift	7	26

5

Хеш-функції у якості генератору випадкових послідовностей



а)

```
var hash = new Hash(0);
var rn0 = hash.GetHash(0);
var rn1 = hash.GetHash(1);
var rn2 = hash.GetHash(12);
var rn3 = hash.GetHash(13, 5);

var rn4 = Hash.GetHash(0, 0);
var rn5 = Hash.GetHash(0, 1);
var rn6 = Hash.GetHash(0, 12);
var rn7 = Hash.GetHash(0, 13, 5);
```

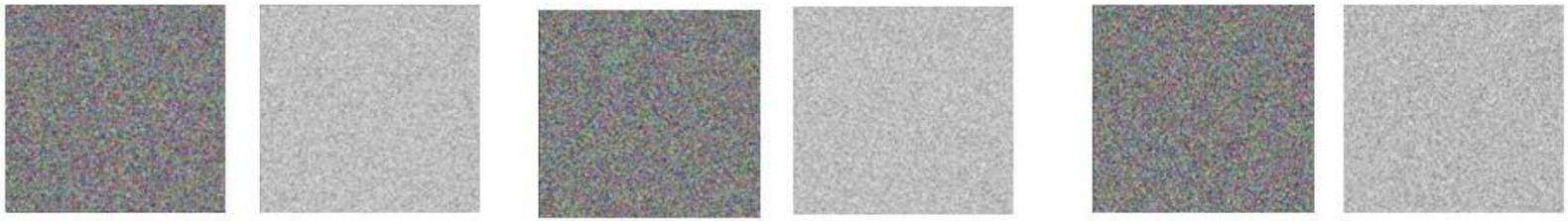
б)

```
class HashRandom
{
    private int seed;
    private int counter;
    public HashRandom(int seed)
{
    this.seed = seed;
    }
    public uint Next()
    {
    return Hash.GetHash(seed, counter++);
    }
}
```

в)

а) Схематичне зображення доступу до елементів у хеш-функції
б) Приклад використання хеш функції у якості генератора
в) Приклад найпростішого класу генератора на основі хеш-функції

Результати оцінки алгоритмів генераторів псевдовипадкових чисел на основі хеш функції



1.1 1.2 2.1 2.2 3.1 3.2

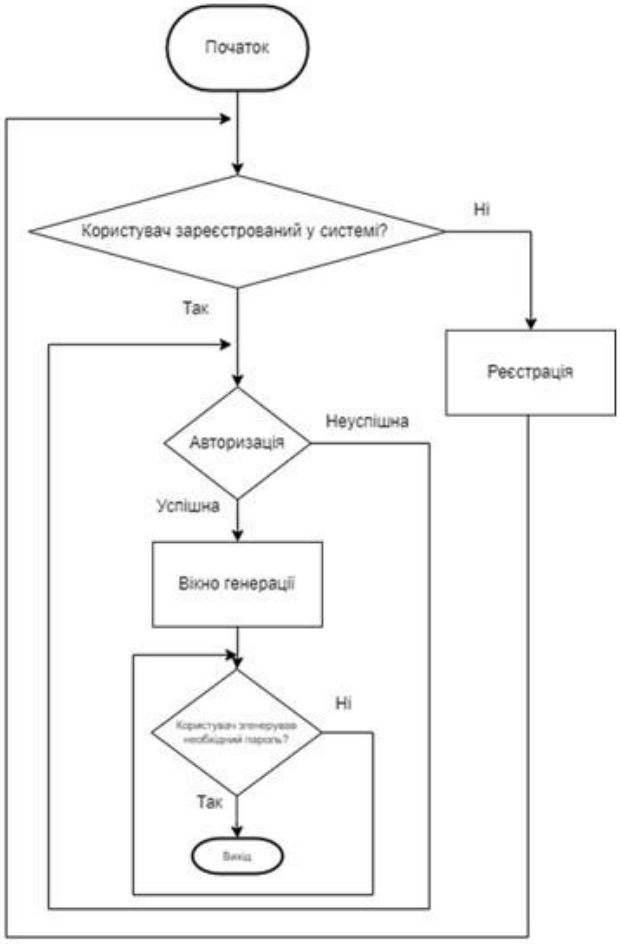
	$0 \dots n$
MD5	202
MurmurHash	9
xxHash	8

7

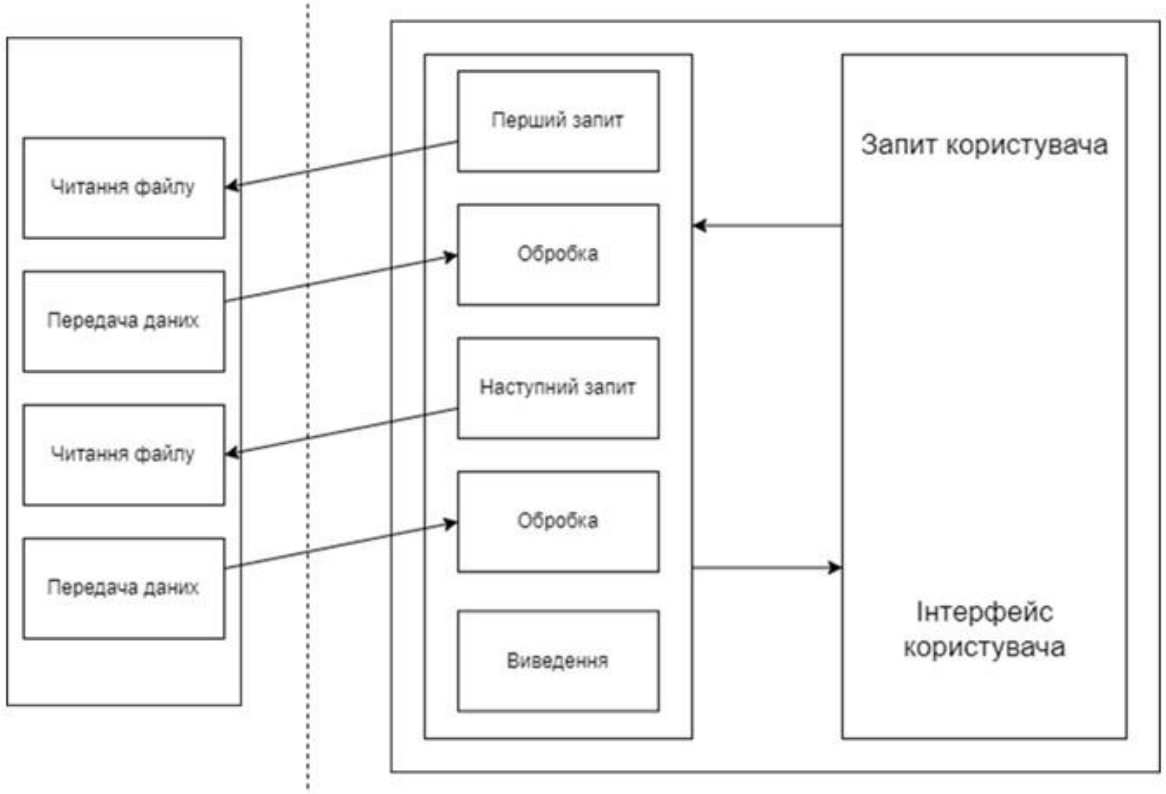
Вимоги та опис розроблюваного прототипу.

До програми висуваються наступні вимоги:

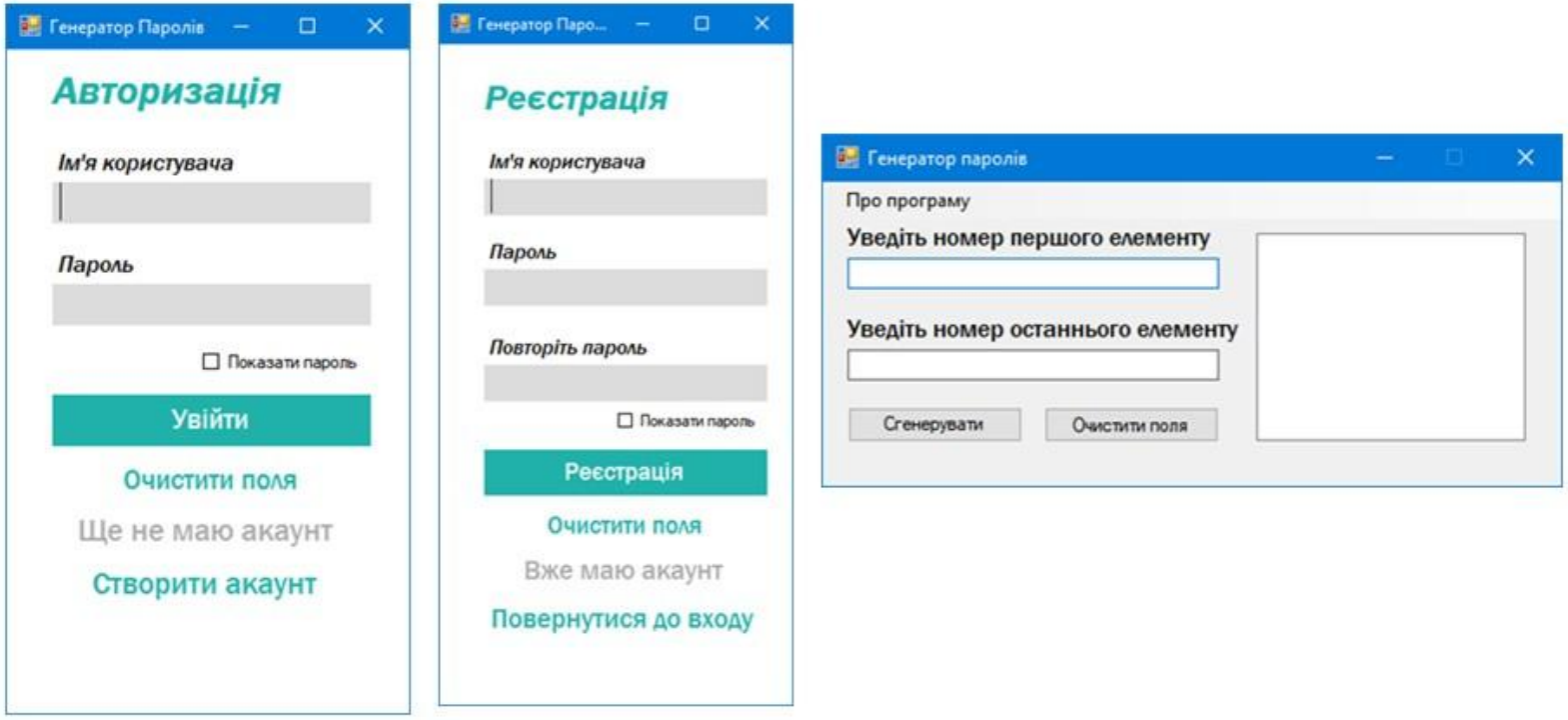
- систему аутентифікації користувача;
- система генерації послідовності та її зберігання;
- система шифрування згенерованої послідовності;
- система видачі паролю за вказаним діапазоном.



Архітектура розробленого прототипу



Зовнішній вигляд розробленого прототипу

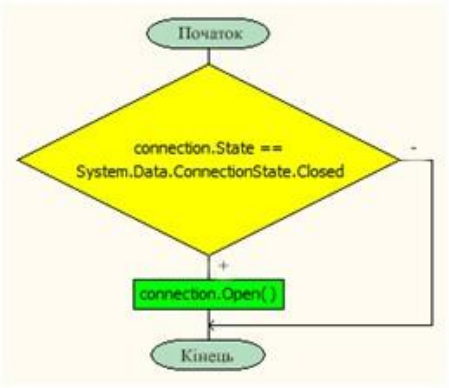


Зміст файлу до та після розшифрування

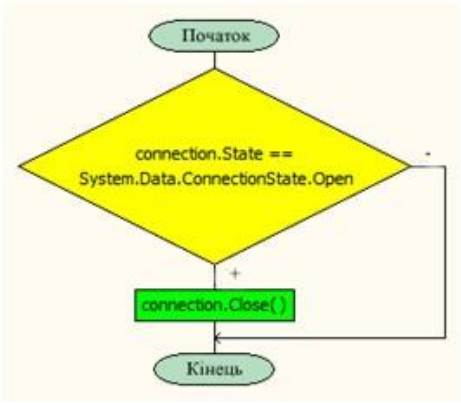
Блок – схеми функціональних частин

Клас DB:

Функція public void openConnection()

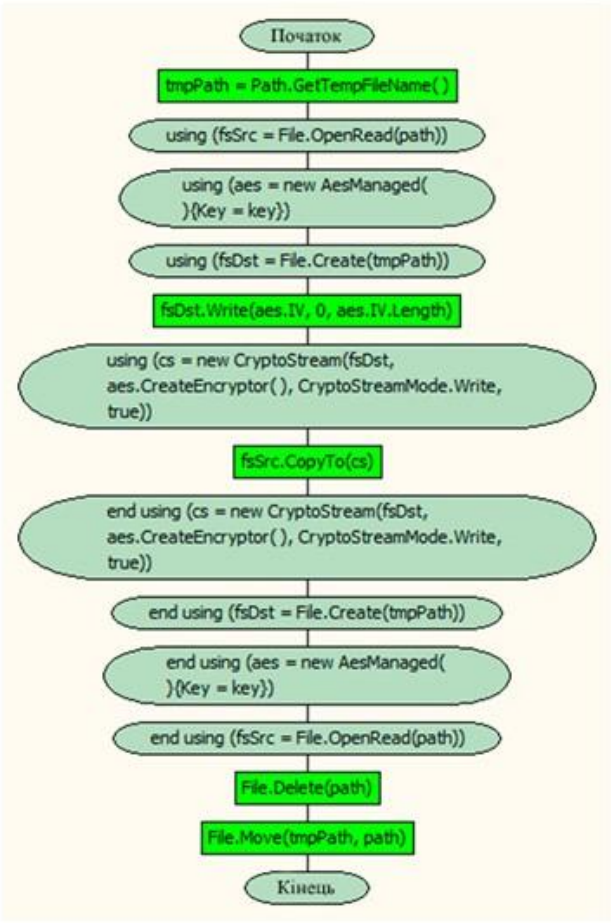


Функція public void closeConnection()



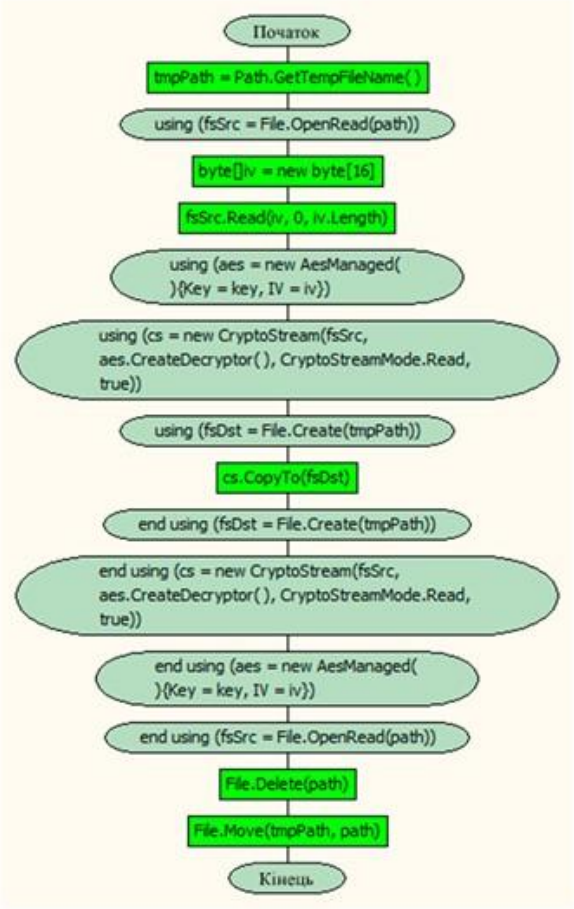
Класс FileEncrypt

Функція public void EncryptFile(string path, byte[] key)

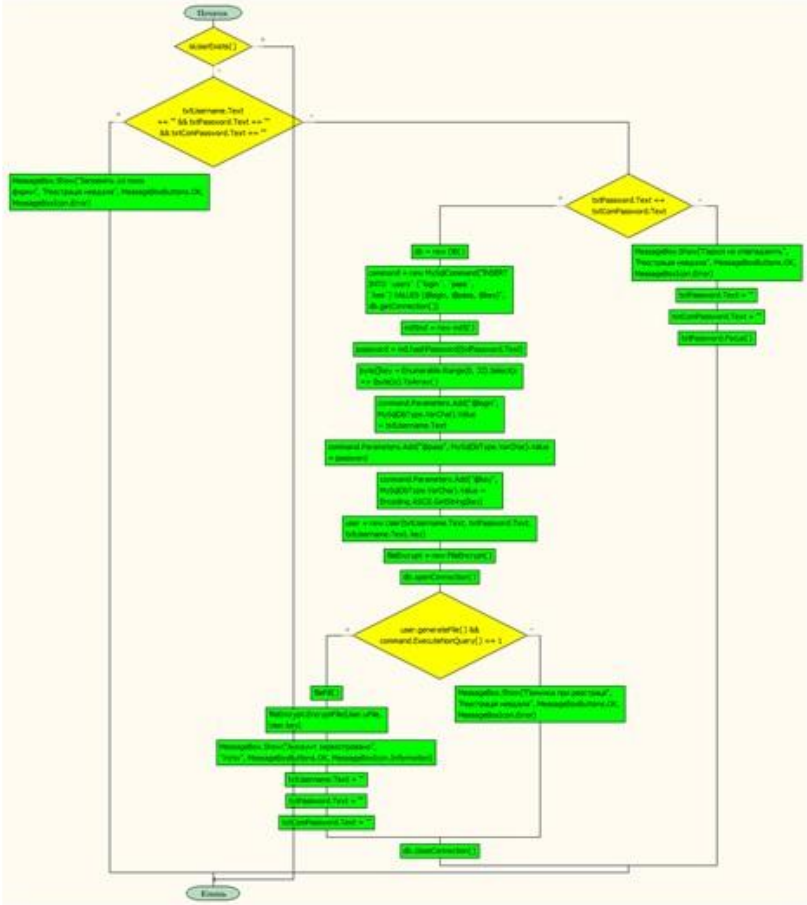


Блок – схеми функціональних частин ч.2

Функція public void DecryptFile(string path, byte[] key)

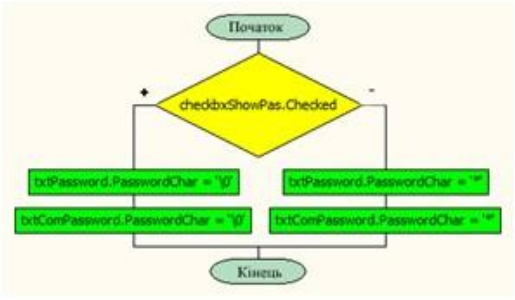


Класс frmRegister, що описує процес реєстрації.
Функція private void registerBtn_Click(object sender, EventArgs e)

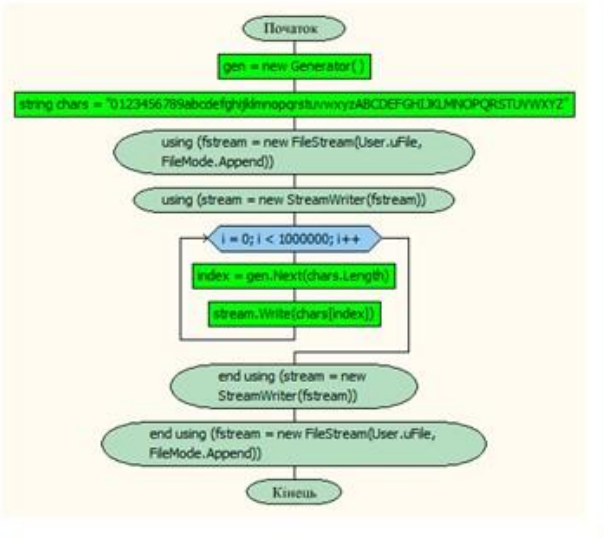


Блок – схеми функціональних частин ч.3

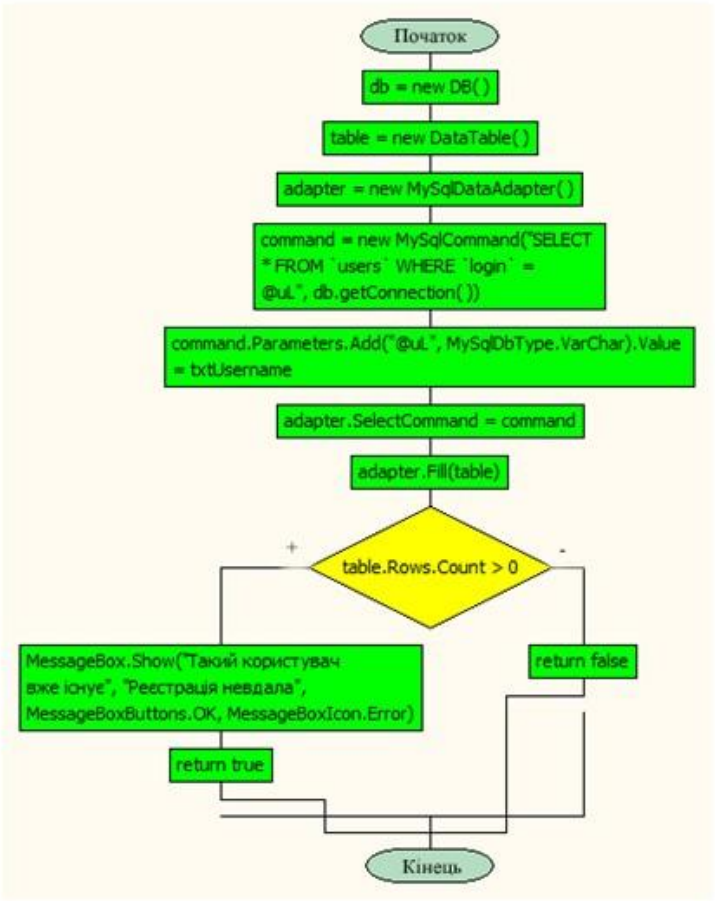
Функція checkbxShowPas_CheckedChanged(object sender, EventArgs e)



Функція private void fileFill()

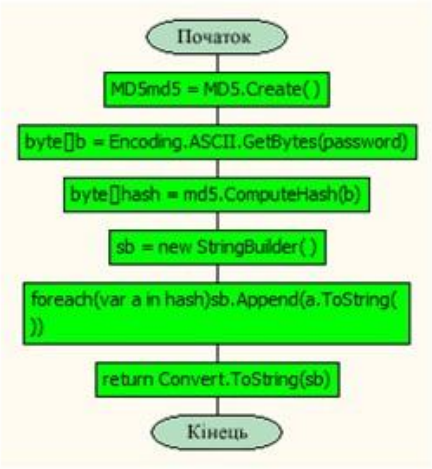


Функція isUserExists()



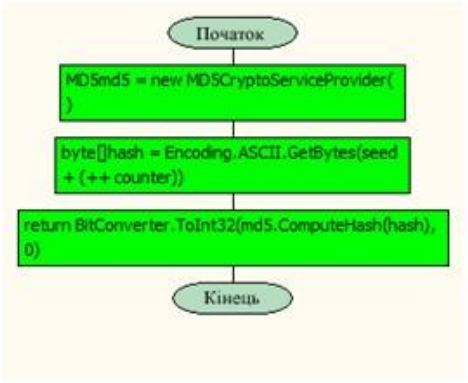
Блок – схеми функціональних частин ч.4

Клас md5, що описує хеш-функцію
Функція public string hashPassword(string password)

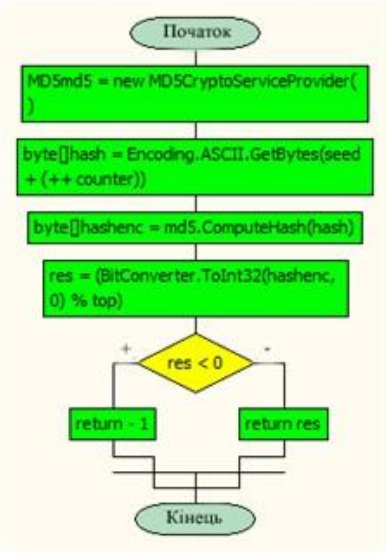


Класс Generator що описує генератор псевдовипадкових чисел
Конструктор класу public Generator()

Функція public int Next()

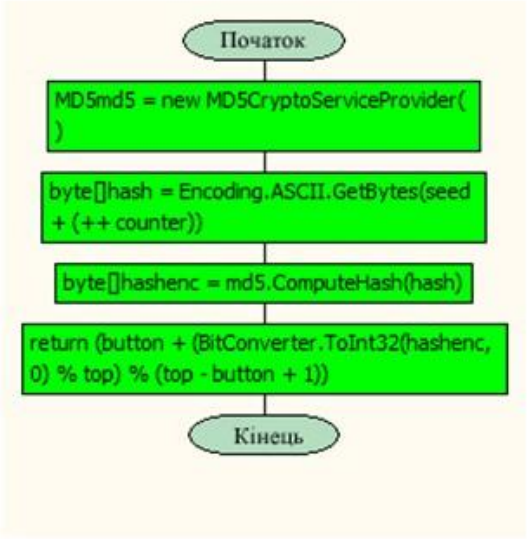


Функція public int Next(int top)

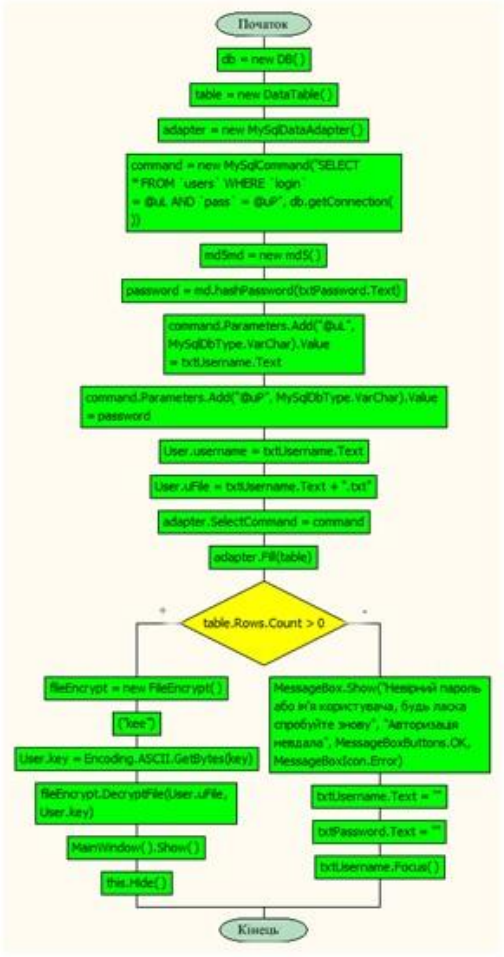


Блок – схеми функціональних частин ч.5

Функція public int Next(int button, int top)



Класс frmLogin, що описує логіку при реєстрації
Функція private void registerBtn_Click(object sender, EventArgs e)



Переваги та недоліки розробленого прототипу

У розробленому прототипі можливо виділити наступні переваги:

- масштабованість - можливість розширювати систему і збільшувати її продуктивність, за рахунок додавання нових модулів;
- ремонтопридатність - зміна одного модуля не вимагає зміни інших модулів;
- можливість тестування - модуль можна від'єднати від всіх інших і протестувати / полагодити;
- супровід - розбиту на модулі програму легше розуміти і супроводжувати.

Недоліками розробленого прототипу є наступне:

- при використанні послідовності маленького розміру зменшуєтьс кількість варіацій доступних паролів і збільшується можливіст злому;
- оскільки відкритий ключ зберігається у базі даних то при злом зловмисник зможе розшифрувати файл з послідовністю;
- при пошкодженні або втраті файлу користувач втрачає уск послідовність, значить, й усі паролі.

Виходячи з вище зазначеного пропонуються наступні модифікації:

- збільшення ентропії алгоритму шляхом додавання у зерно значень іншого генератора;
- модифікування алгоритму шифрування файлу з послідовністю для зменшення ризиків дешифрування;
- створення системи резервного копіювання та доопрацювання системи роботи з файлом.

Висновки

Під час написання випускної кваліфікаційної роботи було розглянуто основні алгоритми генерації псевдовипадкових чисел, розглянуто як їх використовують при генерації паролів.

Також було розглянуто використання ряду існуючих хеш-функцій у якості генераторів псевдовипадкових чисел та використано даний алгоритм на практиці у розробленому прототипі, що використовує нову концепцію для програм – генераторів паролів. В результаті дослідження були вирішені такі задачі:

1) проведено огляд сучасних алгоритмів генерації псевдовипадкових чисел;

2) досліджено можливості використання хеш-функцій у якості генераторів псевдовипадкових чисел;

3) запропоновано алгоритм використання хеш-функції у якості генератора псевдовипадкоих чисел;

4) розроблено прототипу, що використовує алгоритм генерації на основі хеш-функції з новою концепцією зберігання паролів;

Номер документу	Позначення	Найменування	Прим.
		Текстові документи	
1		Пояснювальна записка	93 с.
		Графічний матеріал	
2		Прототипування генератора	А4 ел.ф.
		псевдовипадкових чисел для систем	
		парольного захисту	
3		Принцип роботи генератора	А4 ел.ф.
		псевдовипадкових чисел	
4		Алгоритм оцінки якості генерації	А4 ел.ф.
		алгоритмів	
5		Результати оцінки алгоритмів	А4 ел.ф.
		генераторів псевдовипадкових чисел	
6		Хеш-функції у якості генератору	А4 ел.ф.
		випадкових послідовностей	
7		Результати оцінки алгоритмів	А4 ел.ф.
		генераторів псевдовипадкових	
		чисел на основі хеш-функції	
8		Вимоги та опис розроблюваного	А4 ел.ф.
		прототипу	
9		Архітектура розробленого прототипу	А4 ел.ф.
10		Зовнішній вигляд розробленого	А4 ел.ф.
		прототипу	
11		Зміст файлу до та після	А4 ел.ф.
		розшифрування	
12-16		Блок-схеми функціональних частин	5хА4 ел.ф.
17		Переваги та недоліки розробленого	А4 ел.ф.
		прототипу	
18		Висновки	А4 ел.ф.

Из	Лист	№ докум.	Підпис	Дата				
Розроб.		Коновченко			*ПРОТОТИПУВАННЯ ГЕНЕРАТОРА ПСЕВДОВИПАДКОВИХ ЧИСЕЛ ДЛЯ СИСТЕМ ПАРОЛЬНОГО ЗАХИСТУ*	Літера	Лист	Листів
Перев.		Горелов				Д		
						ХНУРЕ Кафедра КРіСТЗІ		
Н.контр.		Милютченко						
Затв.		Антіпов			*Відомість кваліфікаційної роботи магістра*			