МІНІСТЕРСТВО ОСВІТИ І НАУКИ УКРАЇНИ

**НАЦІОНАЛЬНИЙ УНІВЕРСИТЕТ "ЛЬВІВСЬКА ПОЛІТЕХНІКА"**

Інститут **КНІТ**

Кафедра **ПЗ**



ЗВІТ

До лабораторної роботи №1

**З дисципліни:** *“Безпека програм та даних”*

**На тему:** *“Створення генератора псевдовипадкових чисел”*

**Лектор:**

доцент каф. ПЗ

Сенів М. М.

**Виконав:**

ст. гр. ПЗ-43 Лесневич Є. Є.

**Прийняв:**

ст. викладач каф. ПЗ

Угриновський Б. В.

« » 2024 р.

∑= \_ .

Львів – 2024

**Тема роботи**: створення генератора псевдовипадкових чисел.

**Мета роботи**: ознайомитись з джерелами та застосуванням випадкових чисел, алгоритмами генерування псевдовипадкових чисел та навчитись створювати програмні генератори псевдовипадкових чисел для використання в системах захисту інформації.

**Теоретичні відомості**

Сучасна інформатика широко використовує випадкові числа в різних програмах – від методу Монте-Карло до криптографії. Ряд алгоритмів захисту мережі, заснованих на засобах криптографії, передбачає використання випадкових чисел. Ці застосування висувають дві вимоги до послідовності випадкових чисел: випадковість і непередбачуваність.

Джерелами дійсно випадкових чисел потенційно можуть бути фізичні генератори шумів, такі як імпульсні детектори іонізуючого випромінювання, газорозрядні лампи, конденсатори з втратами струму тощо. Однак такі пристрої можуть знайти доволі обмежене застосування в додатках для захисту інформації. Туту існують проблеми як з випадковістю, так і з точністю отриманих таким методом чисел, не кажучи вже про проблеми підключення такого роду пристроїв до кожної системи в мережі.

Тому криптографічні додатки зазвичай використовують алгоритмічні методи генерування випадкових чисел. Відповідні алгоритми є детермінованими і тому породжують послідовності чисел, які не є статистично випадковими. Однак, якщо алгоритм є достатньо хорошим, породжувані ним послідовності чисел витримують багато тестів на випадковість. Такі числа часто називають псевдовипадковими.

Генератор псевдовипадкових чисел – алгоритм, що генерує послідовність чисел, елементи якої незалежні один від одного і підлягають заданому розподілу.

Найбільш популярним алгоритмом для генерування псевдовипадкових чисел є алгоритм, запропонований Лемером, який називається методом лінійного порівняння. Цей алгоритм має чотири наступних параметри.

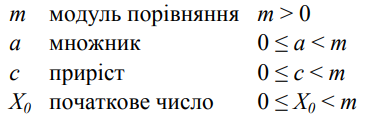
****

Рис. 1 Параметри алгоритму метолу лінійного порівняння

Послідовність псевдовипадкових чисел {X0} отримують за допомогою ітерацій наступного співвідношення:

****

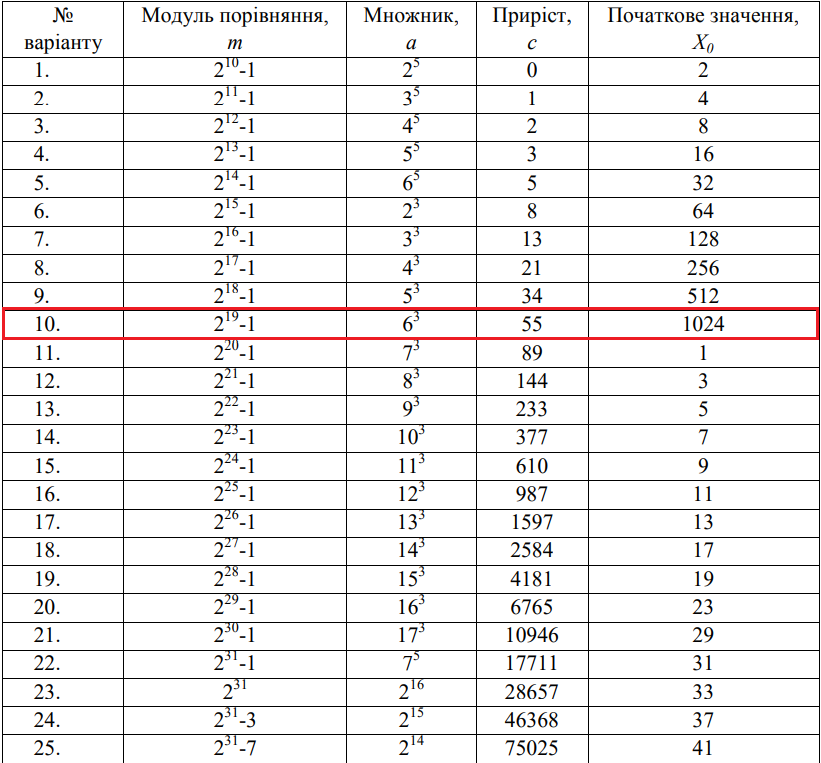
Рис. 2 Співвідношення послідовності псевдовипадкових чисел

**Завдання до виконання роботи**

Згідно до варіанту, наведеного в таблиці, створити програмну реалізацію генератора псевдовипадкових чисел за алгоритмом лінійного порівняння. Програма повинна генерувати послідовність із заданої при вводі кількості псевдовипадкових чисел, результати повинні як виводитись на екран, так і зберігатись у файл. Перевірити період функції генерації, зробити висновок про адекватність вибору параметрів алгоритму. У звіті навести протокол роботи програми, значення періоду функції генерації та зробити висновок про придатність цього генератора для задач криптографії.

**Індивідуальне завдання**

**Варіант: 10**

**.**

**Код програми**

Назва файлу: MainWindow.xaml

<Window x:Class="Lab08\_Wpf.MainWindow"

xmlns="http://schemas.microsoft.com/winfx/2006/xaml/presentation"

xmlns:x="http://schemas.microsoft.com/winfx/2006/xaml"

xmlns:d="http://schemas.microsoft.com/expression/blend/2008"

xmlns:mc="http://schemas.openxmlformats.org/markup-compatibility/2006"

xmlns:local="clr-namespace:Lab08\_Wpf"

mc:Ignorable="d"

Title="Text Animation" Height="450" Width="800" Icon="/color-icon.png">

<Window.Resources>

<ResourceDictionary>

<ResourceDictionary.MergedDictionaries>

<ResourceDictionary Source="/Styles/MainWindowStyle.xaml"></ResourceDictionary>

<ResourceDictionary Source="/Styles/TableTheme.xaml"></ResourceDictionary>

</ResourceDictionary.MergedDictionaries>

</ResourceDictionary>

</Window.Resources>

<Grid>

<Grid.RowDefinitions>

<RowDefinition Height="Auto"/>

<RowDefinition Height="Auto"/>

<RowDefinition Height="\*"/>

</Grid.RowDefinitions>

<!-- Panel with radio buttons centered at the top -->

<StackPanel Grid.Row="0" Orientation="Vertical" HorizontalAlignment="Center" Margin="0,20">

<TextBox Name="TextBox" Style="{DynamicResource FunTextBoxStyle}"/>

<StackPanel Grid.Row="0" Orientation="Vertical" HorizontalAlignment="Center" Margin="0,5">

<CheckBox Name="CheckBoxColor" Style="{DynamicResource CheckboxStyle}"

Content="Color"

HorizontalAlignment="Left"/>

<CheckBox Name="CheckBoxMove" Style="{DynamicResource CheckboxStyle}"

Content="Move"

HorizontalAlignment="Left"/>

<CheckBox Name="CheckBoxSize" Style="{DynamicResource CheckboxStyle}"

Content="Size"

HorizontalAlignment="Left"/>

</StackPanel>

<StackPanel Grid.Row="0" Orientation="Horizontal" HorizontalAlignment="Center" Margin="0,5">

<Button Style="{DynamicResource ButtonStyle}" Click="Button\_Click\_X" Margin="5">X</Button>

<Button Style="{DynamicResource ButtonStyle}" Click="Button\_Click\_Y" Margin="5">Y</Button>

<Button Style="{DynamicResource ButtonStyle}" Click="Button\_Click\_Sin" Margin="5">Sin</Button>

</StackPanel>

</StackPanel>

<!-- Canvas for text animation -->

<Canvas x:Name="canvasX" Grid.Row="2" Background="Transparent"

VerticalAlignment="Center">

<TextBlock x:Name="movingTextX" Visibility="Hidden" Text="Animated Text" FontSize="24" FontWeight="Bold" Foreground="Blue"

Canvas.Left="0" Canvas.Top="0"

VerticalAlignment="Center"/>

</Canvas>

<Canvas x:Name="canvasY" Grid.Row="2" Background="Transparent"

HorizontalAlignment="Center">

<TextBlock x:Name="movingTextY" Visibility="Hidden" Text="Animated Text" FontSize="24" FontWeight="Bold" Foreground="Blue"

Canvas.Left="0" Canvas.Top="0"

HorizontalAlignment="Center"/>

</Canvas>

<Canvas x:Name="canvasSin" Grid.Row="2" Background="Transparent">

<TextBlock x:Name="movingTextSin" Visibility="Hidden" Text="Animated Text" FontSize="24" FontWeight="Bold" Foreground="Blue"

Canvas.Left="0" Canvas.Top="0"/>

</Canvas>

</Grid>

</Window>

Назва файлу: MainWindow.xaml.cs

using System.Windows;

using System.Windows.Controls;

using System.Windows.Media;

using System.Windows.Media.Animation;

using System.Windows.Media.Effects;

namespace Lab08\_Wpf

{

/// <summary>

/// Interaction logic for MainWindow.xaml

/// </summary>

public partial class MainWindow : Window

{

public MainWindow()

{

InitializeComponent();

Loaded += MainWindow\_Loaded;

}

private void MainWindow\_Loaded(object sender, RoutedEventArgs e)

{

}

private void StartTextAnimationSin(

Canvas canvas,

TextBlock movingText,

string value,

bool isColorAnim,

bool isMoveAnim,

bool isSizeAnim)

{

if (isMoveAnim)

{

// Text animation

DoubleAnimation animationX = new DoubleAnimation

{

From = 0,

To = canvas.ActualWidth - movingText.ActualWidth,

Duration = TimeSpan.FromSeconds(4),

RepeatBehavior = RepeatBehavior.Forever

};

animationX.AutoReverse = true;

DoubleAnimation animationY = new DoubleAnimation

{

From = 0,

To = canvas.ActualHeight - movingText.ActualHeight,

Duration = TimeSpan.FromSeconds(2),

RepeatBehavior = RepeatBehavior.Forever

};

animationY.AutoReverse = true;

animationY.EasingFunction = new SineEase();

movingText.BeginAnimation(Canvas.LeftProperty, animationX);

movingText.BeginAnimation(Canvas.TopProperty, animationY);

}

else

{

movingText.BeginAnimation(Canvas.LeftProperty, null);

movingText.BeginAnimation(Canvas.TopProperty, null);

}

if (isColorAnim)

{

// Rainbow color animation

ColorAnimationUsingKeyFrames colorAnimation = new ColorAnimationUsingKeyFrames();

colorAnimation.Duration = TimeSpan.FromSeconds(4);

colorAnimation.RepeatBehavior = RepeatBehavior.Forever;

Color[] rainbowColors = { Colors.Red, Colors.Orange, Colors.Yellow, Colors.Green, Colors.Blue, Colors.Indigo, Colors.Violet };

foreach (Color color in rainbowColors)

{

colorAnimation.KeyFrames.Add(new DiscreteColorKeyFrame(color, KeyTime.Paced));

}

movingText.Foreground = new SolidColorBrush(Colors.Red); // Set initial color

movingText.Foreground.BeginAnimation(SolidColorBrush.ColorProperty, colorAnimation);

// Glow effect animation

DropShadowEffect glowEffect = new DropShadowEffect

{

Color = Colors.Black,

BlurRadius = 0,

ShadowDepth = 1,

Opacity = 1

};

movingText.Effect = glowEffect;

DoubleAnimation glowAnimation = new DoubleAnimation

{

From = 10,

To = 50,

Duration = TimeSpan.FromSeconds(1),

AutoReverse = true,

RepeatBehavior = RepeatBehavior.Forever

};

glowEffect.BeginAnimation(DropShadowEffect.BlurRadiusProperty, glowAnimation);

}

else

{

movingText.Effect = null;

movingText.Foreground = new SolidColorBrush(Colors.Red); // Set initial color

movingText.Foreground.BeginAnimation(SolidColorBrush.ColorProperty, null);

}

if (isSizeAnim)

{

// Size animation from top to bottom

DoubleAnimation animationSize = new DoubleAnimation

{

From = 12,

To = 30,

Duration = TimeSpan.FromSeconds(2),

RepeatBehavior = RepeatBehavior.Forever

};

animationSize.AutoReverse = true;

movingText.BeginAnimation(FontSizeProperty, animationSize);

}

else

{

movingText.BeginAnimation(FontSizeProperty, null);

}

movingText.Text = value;

movingText.Visibility = Visibility.Visible;

}

private void StartTextAnimationX(

Canvas canvas,

TextBlock movingText,

string value,

bool isColorAnim,

bool isMoveAnim,

bool isSizeAnim)

{

if (isMoveAnim)

{

// Text animation from left to right

DoubleAnimation animationX = new DoubleAnimation

{

From = 0,

To = canvas.ActualWidth - movingText.ActualWidth,

Duration = TimeSpan.FromSeconds(4),

RepeatBehavior = RepeatBehavior.Forever

};

animationX.AutoReverse = true;

movingText.BeginAnimation(Canvas.LeftProperty, animationX);

movingText.BeginAnimation(Canvas.TopProperty, null);

}

else

{

movingText.BeginAnimation(Canvas.LeftProperty, null);

movingText.BeginAnimation(Canvas.TopProperty, null);

}

if (isColorAnim)

{

// Rainbow color animation

ColorAnimationUsingKeyFrames colorAnimation = new ColorAnimationUsingKeyFrames();

colorAnimation.Duration = TimeSpan.FromSeconds(4);

colorAnimation.RepeatBehavior = RepeatBehavior.Forever;

Color[] rainbowColors = { Colors.Red, Colors.Orange, Colors.Yellow, Colors.Green, Colors.Blue, Colors.Indigo, Colors.Violet };

foreach (Color color in rainbowColors)

{

colorAnimation.KeyFrames.Add(new DiscreteColorKeyFrame(color, KeyTime.Paced));

}

movingText.Foreground = new SolidColorBrush(Colors.Red); // Set initial color

movingText.Foreground.BeginAnimation(SolidColorBrush.ColorProperty, colorAnimation);

// Glow effect animation

DropShadowEffect glowEffect = new DropShadowEffect

{

Color = Colors.Black,

BlurRadius = 0,

ShadowDepth = 1,

Opacity = 1

};

movingText.Effect = glowEffect;

DoubleAnimation glowAnimation = new DoubleAnimation

{

From = 10,

To = 50,

Duration = TimeSpan.FromSeconds(1),

AutoReverse = true,

RepeatBehavior = RepeatBehavior.Forever

};

glowEffect.BeginAnimation(DropShadowEffect.BlurRadiusProperty, glowAnimation);

}

else

{

movingText.Effect = null;

movingText.Foreground = new SolidColorBrush(Colors.Red); // Set initial color

movingText.Foreground.BeginAnimation(SolidColorBrush.ColorProperty, null);

}

if (isSizeAnim)

{

// Size animation from top to bottom

DoubleAnimation animationSize = new DoubleAnimation

{

From = 12,

To = 30,

Duration = TimeSpan.FromSeconds(2),

RepeatBehavior = RepeatBehavior.Forever

};

animationSize.AutoReverse = true;

movingText.BeginAnimation(FontSizeProperty, animationSize);

}

else

{

movingText.BeginAnimation(FontSizeProperty, null);

}

movingText.Text = value;

movingText.VerticalAlignment = VerticalAlignment.Center;

movingText.Visibility = Visibility.Visible;

}

private void StartTextAnimationY(

Canvas canvas,

TextBlock movingText,

string value,

bool isColorAnim,

bool isMoveAnim,

bool isSizeAnim)

{

if (isMoveAnim)

{

// Text animation from top to bottom

DoubleAnimation animationY = new DoubleAnimation

{

From = 0,

To = canvas.ActualHeight - movingText.ActualHeight,

Duration = TimeSpan.FromSeconds(4),

RepeatBehavior = RepeatBehavior.Forever

};

animationY.AutoReverse = true;

movingText.BeginAnimation(Canvas.TopProperty, animationY);

movingText.BeginAnimation(Canvas.LeftProperty, null);

}

else

{

movingText.BeginAnimation(Canvas.LeftProperty, null);

movingText.BeginAnimation(Canvas.TopProperty, null);

}

if (isColorAnim)

{

// Rainbow color animation

ColorAnimationUsingKeyFrames colorAnimation = new ColorAnimationUsingKeyFrames();

colorAnimation.Duration = TimeSpan.FromSeconds(4);

colorAnimation.RepeatBehavior = RepeatBehavior.Forever;

Color[] rainbowColors = { Colors.Red, Colors.Orange, Colors.Yellow, Colors.Green, Colors.Blue, Colors.Indigo, Colors.Violet };

foreach (Color color in rainbowColors)

{

colorAnimation.KeyFrames.Add(new DiscreteColorKeyFrame(color, KeyTime.Paced));

}

movingText.Foreground = new SolidColorBrush(Colors.Red); // Set initial color

movingText.Foreground.BeginAnimation(SolidColorBrush.ColorProperty, colorAnimation);

// Glow effect animation

DropShadowEffect glowEffect = new DropShadowEffect

{

Color = Colors.Black,

BlurRadius = 0,

ShadowDepth = 1,

Opacity = 1

};

movingText.Effect = glowEffect;

DoubleAnimation glowAnimation = new DoubleAnimation

{

From = 10,

To = 50,

Duration = TimeSpan.FromSeconds(1),

AutoReverse = true,

RepeatBehavior = RepeatBehavior.Forever

};

glowEffect.BeginAnimation(DropShadowEffect.BlurRadiusProperty, glowAnimation);

}

else

{

movingText.Effect = null;

movingText.Foreground = new SolidColorBrush(Colors.Red); // Set initial color

movingText.Foreground.BeginAnimation(SolidColorBrush.ColorProperty, null);

}

if (isSizeAnim)

{

// Size animation from top to bottom

DoubleAnimation animationSize = new DoubleAnimation

{

From = 12,

To = 30,

Duration = TimeSpan.FromSeconds(2),

RepeatBehavior = RepeatBehavior.Forever

};

animationSize.AutoReverse = true;

movingText.BeginAnimation(FontSizeProperty, animationSize);

}

else

{

movingText.BeginAnimation(FontSizeProperty, null);

}

movingText.Text = value;

movingText.Visibility = Visibility.Visible;

}

private void Button\_Click\_Sin(object sender, RoutedEventArgs e)

{

movingTextX.Visibility = Visibility.Hidden;

movingTextY.Visibility = Visibility.Hidden;

Button but = sender as Button;

if (but is not null)

{

string textValue = TextBox.Text;

StartTextAnimationSin(

canvasSin,

movingTextSin,

textValue,

CheckBoxColor.IsChecked != null && CheckBoxColor.IsChecked.Value,

CheckBoxMove.IsChecked != null && CheckBoxMove.IsChecked.Value,

CheckBoxSize.IsChecked != null && CheckBoxSize.IsChecked.Value);

}

}

private void Button\_Click\_X(object sender, RoutedEventArgs e)

{

movingTextY.Visibility = Visibility.Hidden;

movingTextSin.Visibility = Visibility.Hidden;

Button but = sender as Button;

if (but is not null)

{

string textValue = TextBox.Text;

StartTextAnimationX(

canvasX,

movingTextX,

textValue,

CheckBoxColor.IsChecked != null && CheckBoxColor.IsChecked.Value,

CheckBoxMove.IsChecked != null && CheckBoxMove.IsChecked.Value,

CheckBoxSize.IsChecked != null && CheckBoxSize.IsChecked.Value);

}

}

private void Button\_Click\_Y(object sender, RoutedEventArgs e)

{

movingTextX.Visibility = Visibility.Hidden;

movingTextSin.Visibility = Visibility.Hidden;

Button but = sender as Button;

if (but is not null)

{

string textValue = TextBox.Text;

StartTextAnimationY(

canvasY,

movingTextY,

textValue,

CheckBoxColor.IsChecked != null && CheckBoxColor.IsChecked.Value,

CheckBoxMove.IsChecked != null && CheckBoxMove.IsChecked.Value,

CheckBoxSize.IsChecked != null && CheckBoxSize.IsChecked.Value);

}

}

}

}

**Результати роботи**

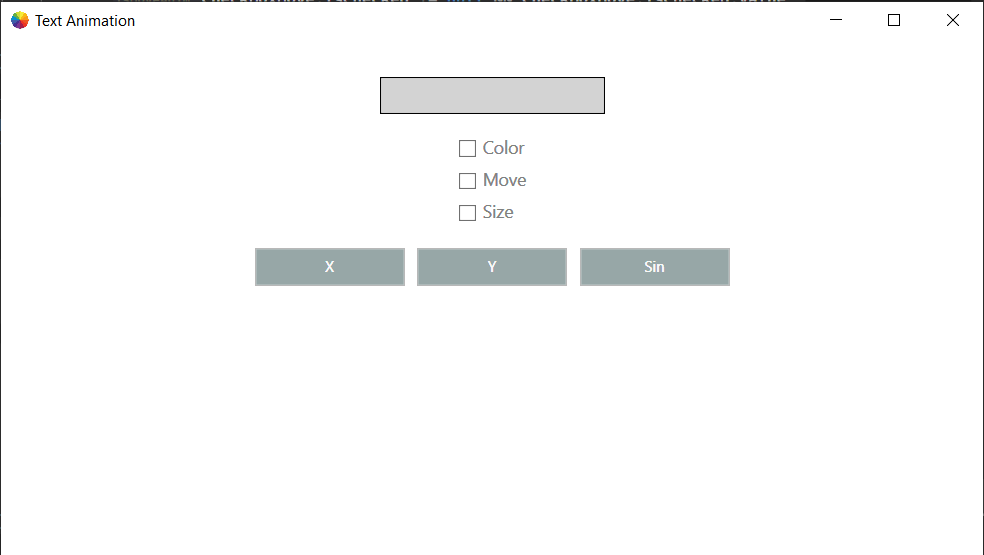


Рис. 3 Головне вікно програми

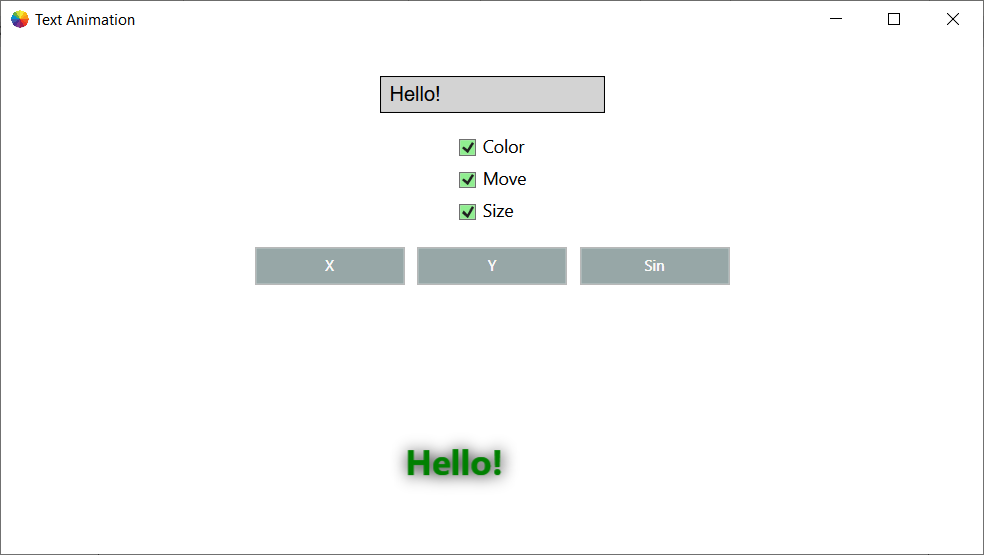


Рис. 4 Вигляд створеної анімації

**Відповіді на контрольні питання**

1. Означте поняття XML з погляду гіпермедіа. У яких іще сферах використовують XML?

XML, або "eXtensible Markup Language", є мовою розмітки, яка використовується для представлення і структурування даних у текстовому форматі. З погляду гіпермедіа, XML часто використовується для створення та обміну структурованими даними, які включають у себе гіперпосилання на інші ресурси, такі як зображення, відео, аудіо, а також текстовий контент. Використовують у веб розробці, базах даних, конфігураційних файлах, документації та інтеграції додатків.

9. Назвіть ключові особливості WPF.

Ключові особливості WPF:

* Розмітка за допомогою XAML
* Розділення дизайну та логіки
* Векторна графіка
* Стилі та шаблони
* Анімація та ефекти
* Геометрія та трансформації
* Широкий вибір елементів управління
* Підтримка стилів, тем та ресурсів
* Вбудована підтримка мультимедіа
* Гнучка система подій та зв'язку даних

12. Перелічіть варіанти надання WPF-вікну нестандартної форми.

* Закруглені кути та прозорість
* Скасування рамки вікна
* Створення власної форми використовуючи шаблони
* Використання бібліотек сторонніх розробників
* Користування прозорими картинками для фону
* Використання бібліотеки WindowsAPICodePack

**Висновки**

Отже, під час виконання даної лабораторної роботи я отримав практичні навички роботи з мультимедійними даними та анімованими об’єктами звикористанням засобів WPF.