Міністерство освіти і науки України
Криворізький національний університет
Факультет інформаційних технологій
Кафедра комп’ютерних систем та мереж

Методичні вказівки

до виконання лабораторних робіт
з дисципліни «Об’єктно-орієнтоване програмування»
для студентів спеціальності
123 «Комп’ютерна інженерія»
усіх форм навчання
Частина 2

Кривий Ріг
2021

Укладачі: Музика І. О., канд. техн. наук, доцент
Кузнєцов Д. І., канд. техн. наук, доцент
Рецензент: Тиханський М. П., канд. техн. наук, доцент

Дані методичні вказівки містять завдання та теоретичні відомості для виконання лабораторних робіт з дисципліни «Об’єктно-орієнтоване програмування» за спеціальністю 123 «Комп’ютерна інженерія». Висвітлені питання проектування програмних додатків з візуальним інтерфейсом з використанням технологій Windows Forms, Windows Presentation Foundation, ADO.NET, Sockets, LINQ мовою C#.

Розглянуто
на засіданні кафедри
комп’ютерних систем та мереж

Протокол № 1
від 27.08.2021 р.

Схвалено
на вченій раді факультету
інформаційних технологій

Протокол № 1
від 30.08.2021 р.

ВСТУП

Об’єктно-орієнтоване програмування (ООП) – одна з парадигм програмування, яка розглядає програму як множину «об’єктів», що взаємодіють між собою. У ній використано декілька технологій, зокрема успадкування, модульність, поліморфізм та інкапсуляцію. Незважаючи на те, що ця парадигма з’явилась ще в 1960-х роках, вона не мала широкого застосування до 1990-х. Сьогодні багато мов програмування (зокрема, C#, C++, Java, D, Python, PHP, Ruby, Objective-C, Object Pascal та ін.) підтримують ООП.

Дисципліна «Об’єктно-орієнтоване програмування» – одна із фундаментальних дисциплін професійної підготовки студентів спеціальності 123 «Комп’ютерна інженерія». Передумовою успішного вивчення даної дисципліни є володіння навичками програмування та алгоритмізації, які отримані під час вивчення дисциплін першого року навчання: «Основи інформаційних технологій», «Вища математика» та «Програмування».

Метою даної дисципліни є оволодіння мовою програмування C#, а також методами об’єктно-орієнтованого проектування та розробки складних програм. Як результат навчання за курсом «Об’єктно-орієнтоване програмування» студенти повинні: знати основи побудови класів, способи обробки виключних ситуацій, організовувати перевантаження операцій, працювати з інтерфейсами, виконувати читання та запис текстових і бінарних файлів, вміти будувати додатки типу Console, Windows Forms, Windows Presentation Foundation на базі .NET Framework з використанням середовища Microsoft Visual Studio. Згідно навчального плану вивчення даної дисципліни передбачено протягом двох семестрів.

Запропонований матеріал буде корисним як студентам, що вивчають програмування за спеціальностями галузі знань 12 «Інформаційні технології», так і професійним програмістам, а також усім тим, хто має інтерес до даної області знань.

ЛАБОРАТОРНА РОБОТА № 1

Тема: основи технології Windows Forms.

Мета: розробити програми типу Windows Forms Application на мові C#.

Теоретичні відомості

Windows Forms – технологія розробки додатків з графічним інтерфейсом (рисунок 1.1). Дана технологія спрощує доступ до елементів інтерфейсу Microsoft Windows за рахунок створення обгортки для існуючого Win32 API у керованому коді.

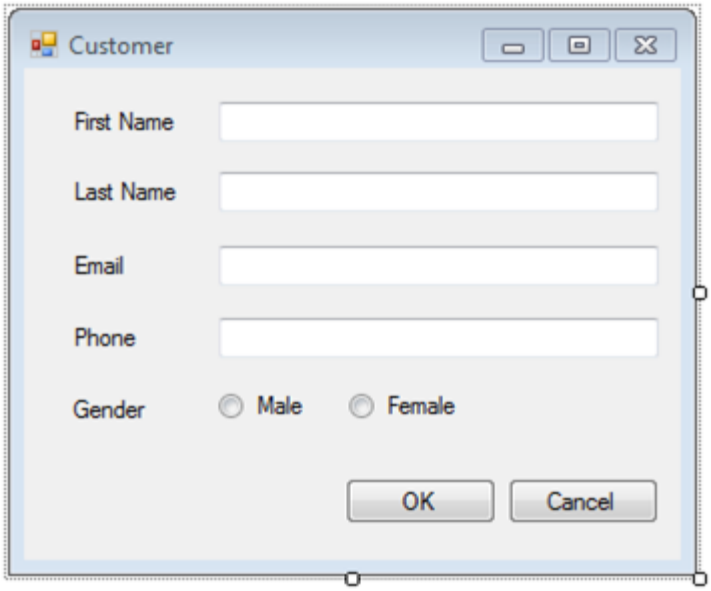


Рисунок 1.1 – Приклад додатку Windows Forms

З одного боку, Windows Forms розглядається як заміна більш старої і складної бібліотеці MFC, спочатку написаної на мові C++. З іншого боку, Windows Forms не пропонує парадигму, порівнянну з MVC (Model-View-Controller). Для виправлення цієї ситуації і реалізації даної функціональності в Windows Forms існують сторонні бібліотеки. Однією з найбільш використовуваних подібних бібліотек є User Interface Process Application Block, випущена спеціальною групою Microsoft, що займається прикладами і рекомендаціями, для

безкоштовного скачування. Ця бібліотека також містить вихідний код і навчальні приклади для прискорення навчання/

Всередині .NET Framework, Windows Forms реалізується в рамках простору імен System.Windows.Forms.

Розглянемо створення додатку Windows Forms. Для прикладу створимо пусту форму і відобразимо її на екрані. При розробці цього прикладу ми поки що не будем використовувати Visual Studio. Наберемо в тестовому редакторі таких код програми.

```
using System;
using System.Windows.Forms;
namespace NotepadForms
{
  public class MyForm : System.Windows.Forms.Form
  {
    public MyForm()
    {
    }
    [STAThread]
    static void Main()
    {
      Application.Run(new MyForm());
    }
  }
}
```

Якщо цей код скомпілювати і запустити, то отримаємо маленьку пустую форму без заголовка. Ніяких реальних корисних функцій, проте це – Windows Forms.

У наведеному коде заслуговують уваги дві речі. Перша – той факт, що при створенні класу `MyForm` використано спадкування. Наступний рядок об'являє `MyForm` як нащадку `System.Windows.Forms.Form`:

```
public class MyForm : System.Windows.Forms.Form
```

Клас `Form` – один із головних класів у просторі імен `System.Windows.Forms`. Наступний фрагмент коду варто розглянути більш детально:

```
[STAThread]
static void Main()
{
  Application.Run(new MyForm());
}
```

Main – точка входу за умовчанням у будь-який клієнтський додаток на C#. Як правило, у більш крупних проектах метод Main() не буде знаходитися у класі форми, а скоріш за все в класі, який відповідає за процес запуска. У даному випадку необхідно встановити ім'я такого запускаючого класу у діалоговому вікні властивостей проекту. Зверніть увагу на атрибут [STAThread]. Він встановлює модель багатопоточності COM в STA (однопоточний апартамент). Модель багатопоточності STA потрібна для взаємодії з COM і встановлюється за умовчанням в кожному проекті Windows Forms.

Метод Application.Run() відповідає за запуск стандартного циклу повідомлень додатку. Application.Run() має три перевантаження.

Перше з них не приймає параметрів; друге приймає у якості параметра об'єкт ApplicationContext. У нашому прикладі об'єкт MyForm стає головною формою додатку. Це означає, що коли форма закривається, то додаток завершується. Використовуючи клас ApplicationContext, можна у більшій мірі контролювати завершення головного циклу повідомлень і вихід з додатку.

Клас Application містить у собі дуже корисну функціональність. Він надає групу статичних методів і властивостей для керування процесом запуска і зупинки додатку, а також забезпечує доступ до повідомлень Windows, які оброблює додаток.

А тепер як буде виглядати цей додаток, якщо його згенерувати в Visual Studio? Перше, що слід відзначити – буде створено два файли. Причина в тому, що Visual Studio використовує можливість часткових (partial) класів і виділяє весь код, згенерований візуальним дизайнером, в окремий файл. Якщо використовується ім'я за замовчуванням – Form1, то ці два файли будуть називатися Form1.cs і Form1.Designer.cs. Якщо тільки у вас не включена опція Show All Files

(Показати всі файли) в меню Project (Проект), то ви не побачите в провіднику Solution Explorer файлу `Form1.Designer.cs`. Нижче показаний код цих двох файлів, згенерованих Visual Studio. Спочатку – `Form1.cs`:

```
using System;
using System.Collections.Generic;
using System.ComponentModel;
using System.Data;
using System.Drawing;
using System.Linq;
using System.Text;
using System.Windows.Forms;

namespace VisualStudioForm
{
  public partial class Form1 : Form
  {
    public Form1()
    {
      InitializeComponent();
    }
  }
}
```

Тут ми бачимо тільки оператори `using` і простий конструктор. А тепер – код `Form1.Designer.cs`:

```
namespace VisualStudioForm
{
  partial class Form1
  {
    /// <summary>
    /// Required designer variable.
    /// </summary>
    private System.ComponentModel.IContainer components = null;
    /// <summary>
    /// Clean up any resources being used.
    /// </summary>
```

```
    /// <param name="disposing"> true if managed resources
should be disposed; otherwise, false. </param>
    protected override void Dispose(bool disposing)
    {
      if (disposing && (components != null))
      {
        components.Dispose();
      }
      base.Dispose(disposing);
    }
#region Windows Form Designer generated code
    /// <summary>
    /// Required method for Designer support - do not modify
    /// the contents of this method with the code editor.
    /// </summary>
    private void InitializeComponent()
    {
      this.components = new System.ComponentModel.Container();
      this.AutoScaleMode =
        System.Windows.Forms.AutoScaleMode.Font;
      this.Text = "Form1";
    }
#endregion
  }
}
```

Файл, що згенерував дизайнером форм, рідко піддається ручному редагування. Єдиним винятком може бути випадок, коли необхідна спеціальна обробка в методі `Dispose()`.

Якщо поглянути на цей код прикладу додатки в цілому, то ми побачимо, що він набагато довший, ніж простий приклад командного рядка. Тут перед початком класу присутні кілька операторів `using`, і більшість з них в даному прикладі не потрібні. Однак їх присутність нічим не заважає. Клас `Form1` успадковується від `System.Windows.Forms.Form`, як і в попередньому, введеному в Notepad прикладі, але в цій точці починаються розбіжності. По-перше, в файлі `Form1.Designer` з'являється рядок:

```
    private System.ComponentModel.IContainer components = null;
```

У даному прикладі цей рядок коду нічого не робить. Але, додаючи компонент в форму, ви можете також додати його в об’єкт `components`, який представляє собою контейнер. Причина додавання цього контейнера – в необхідності правильної обробки знищення форми. Клас форми підтримує інтерфейс `IDisposable`, тому що він реалізований у класі `Component`. Коли компонент додається в контейнер, то цей контейнер повинен подбати про коректне знищення свого вмісту при закритті форми. Це можна побачити в методі `Dispose` нашого прикладу:

```
protected override void Dispose(bool disposing)
{
  if (disposing && (components != null))
  {
    components.Dispose();
  }
  base.Dispose(disposing);
}
```

Тут ми бачимо, що коли викликається метод `Dispose` форми, то також викликається метод `Dispose` об’єкта `components`, оскільки він містить у собі інші компоненти, які також повинні бути коректно видалені.

Конструктор класу `Form1`, що знаходиться у файле `Form1.cs`, виглядає так:

```
public Form1()
{
  InitializeComponent();
}
```

Зверніть увагу на виклик `InitializeComponent()`.

Метод `InitializeComponent()` знаходиться у файлі `Form1.Designer.cs` і виконує те, що слідує з його найменування – ініціалізує всі елементи керування, які можуть бути добавлені на форму. Він також ініціалізує властивості форми.

У нашому прикладі метод `InitializeComponent()` виглядає так:

```
private void InitializeComponent()
```

```
    {
      this.components = new System.ComponentModel.Container();
      this.AutoScaleMode =
System.Windows.Forms.AutoScaleMode.Font;
      this.Text = "Form1";
    }
```

Як бачите, тут присутній лише базовий код ініціалізації. Цей метод пов'язаний з візуальним дизайнером Visual Studio. Коли в форму вносяться зміни в дизайнері, вони відображаються на `InitializeComponent()`. Якщо ви вносите будь-які зміни в `InitializeComponent()`, то наступного разу після того, як щось буде змінено в дизайнері, ці ручні зміни будуть втрачені. `InitializeComponent()` повторно генерується після кожної зміни дизайну форми. Якщо виникає необхідність додати деякий додатковий код для форми або елементів управління і компонентів форми, це повинно бути зроблено після виклику `InitializeComponent()`. Цей метод також відповідає за створення екземплярів елементів управління, тому будь-який виклик, що посилається на них, виконаний до `InitializeComponent()`, завершиться появою виключення нульового посилання.

Ієрархія класів

Важливість розуміння ієрархії стає очевидною в процесі проектування і конструювання призначених для користувача елементів управління. Якщо такий елемент управління успадкований від конкретного бібліотечного елемента управління, наприклад, коли створюється текстове поле з деякими додатковими методами і властивостями, то має сенс успадкувати його від звичайного текстового поля і потім перевизначити, і додати необхідні методи. Однак якщо доводиться створювати елемент управління, який не відповідає жодному з існуючих в .NET Framework, то його доведеться успадкувати від одного з базових класів: `Control` або `ScrollableControl`, якщо потрібні можливості прокрутки, або

ContainerControl, якщо він повинен служити контейнером для інших елементів управління.

Стандартні елементи керування і компоненти

Button

Клас Button представляє просту командну кнопку і успадковується від ButtonBase (рисунок 1.2). Найчастіше необхідно написати код обробки події Click. Наступний фрагмент коду реалізує обробник події Click. Коли виконується клік на кнопці, з'являється вікно повідомлення, яке відображає ім'я кнопки.

```
private void btnTest_Click(object sender,
  System.EventArgs e)
{
  MessageBox.Show("Выполнен щелчок на " +
    ((Button)sender).Name + ".");
}
```

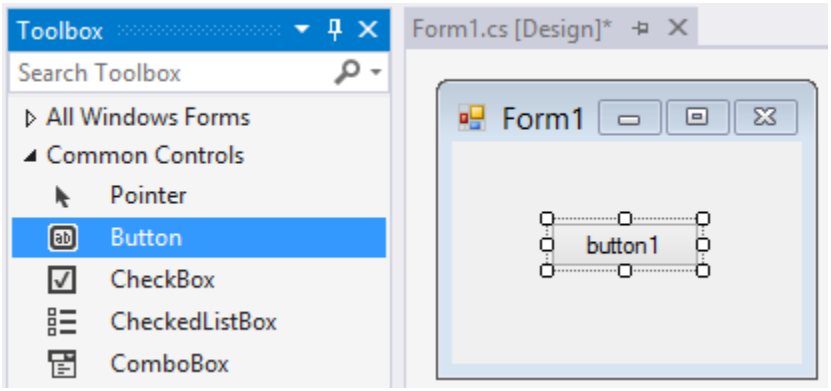


Рисунок 1.2 – Компонент Button на формі

За допомогою методу PerformClick можна емулювати подію Click кнопки без необхідності дійсного виконання клацання користувачем. Метод NotifyDefault приймає в параметрі значення булевского типу і повідомляє кнопці, щоб вона відобразила себе як кнопку за замовчуванням. Зазвичай кнопка за замовчуванням на формі має злегка потовщену рамку.

Щоб ідентифікувати кнопку як кнопку за замовчуванням, потрібно встановити властивість AcceptButton форми рівним посиланням на цю

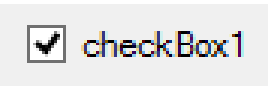
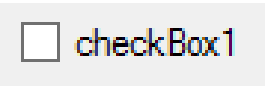
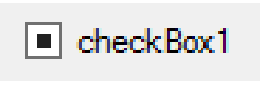
кнопку. Після цього, якщо користувач натисне клавішу <Enter>, згенерує подія `Click` цієї кнопки за замовчуванням.

Кнопки можуть містити на своїй поверхні як текст, так і графічне зображення. Зображення доступні для кнопок через об'єкт `ImageList` або властивість `Image`. Об'єкт `ImageList` є саме те, що можна припустити по його назві: список зображень, який управляється компонентом, поміщеним на форму.

Як `Text`, так і `Image` мають властивість `Align`, призначене для вирівнювання тексту або зображення на поверхні кнопки. Властивість `Align` приймає значення типу перерахування `ContentAlignment`. Текст або зображення можуть бути вирівняні в комбінації з лівого або правого кордоні кнопки або по верхній або нижній межі.

Checkbox

Елемент керування `CheckBox` (прапорець) також успадкований від `ButtonBase` і використовується для прийняття команди користувача з двома або трьома станами. Якщо властивість `ThreeState` установлено в true, то властивість `CheckState` елемента `CheckBox` може приймати одну з наступних трьох значень:

	`Checked`	Елемент CheckBox відмічений
	`Unchecked`	Елемент Checkbox не відмічений
	`Indeterminate`	У цьому стані елемент CheckBox не доступний

Стан `Indeterminate` може бути встановлено тільки програмно, а не користувачем. Це зручно, якщо ви хочете повідомити користувачеві, що опція не була встановлена. Для отримання поточного стану у вигляді булевского значення можна звернутися до властивості `Checked`.

Події `CheckedChanged` і `CheckStateChanged` виникають, коли змінюється властивість `CheckState` або `Checked`. Перехоплення цих подій може стати в нагоді для установки інших значень на основі

нового стану CheckBox. У класі форми frmControls подія CheckedChanged для декількох елементів CheckBox обробляється наступним методом:

```
private void checkBoxChanged(object sender, EventArgs e)
{
  CheckBox checkBox = (CheckBox)sender;
  MessageBox.Show("НоВое значение " + checkBox.Name +
    " равно " +
    checkBox.Checked.ToString());
}
```

При зміні стану кожного з цих елементів відображається вікно повідомлення з ім'ям елемента CheckBox і його новим станом.

RadioButton, ComboBox, ListBox, CheckedListBox, Label, TextBox, RichTextBox и MaskedTextBox, Panel та ін.

Детальніше можна ознайомитися у книзі: C# и платформа .NET 4 для профессионалов / К. Нейгел, Б. Ивьен, Дж. Глини, К. Уотсон; пер. с англ. – М.: Вильямс, 2011. – С. 1257-1293.

Аудиторне завдання

Створити додаток Windows Forms. Обчислити значення виразу, передбачивши обробку помилок та видачу відповідних повідомлень користувачу. Параметри виразу a, b, N користувач вводить у вікні програми.

$$\sum_{i=1}^{N} \frac{a + i^2 - \sqrt{b}}{\sin(i\pi) + \sqrt{\pi}}$$

Розв'язання

На формі необхідно налаштувати такі візуальні компоненти (рисунок 1.3).

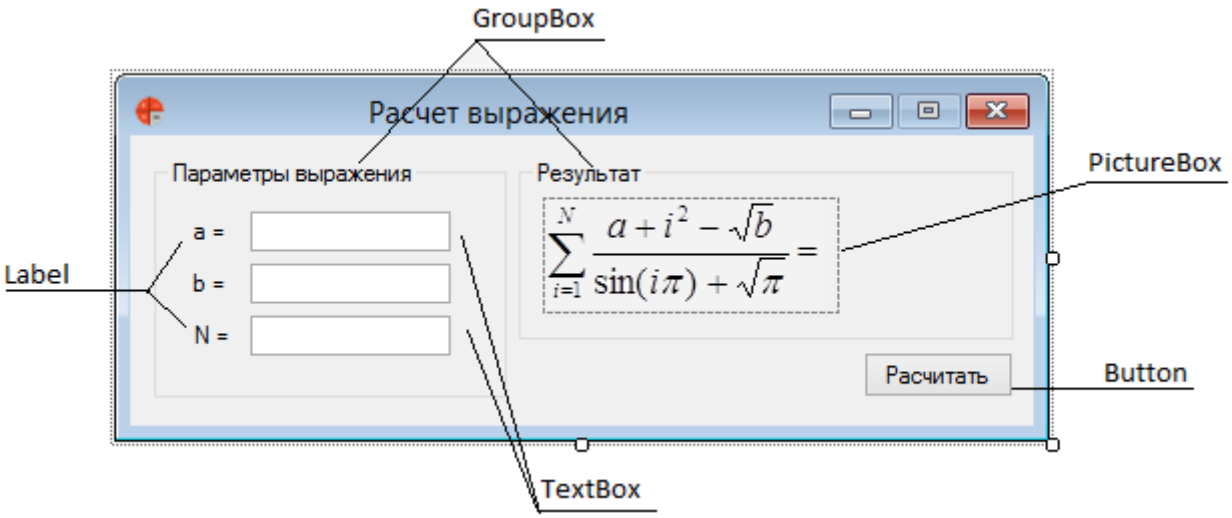


Рисунок 1.3 – Створення форми у редакторі Form Design

Для форми доцільно налаштувати такі властивості:

Властивість	Значення	Пояснення
`Name`	`FormMain`	назва форми
`FromBorederStyle`	`FixedSingle`	заборонити можливість зміни розміру вікна
`Icon`	Імпортоване зображення	іконка у верхньому лівому кутку вікна
`MaximizeBox`	`False`	заборона розгортати вікно на весь екран
`StartPosition`	`CenterScreen`	запускати вікно посередині екрану
`Text`	Расчет выражения	назва, що відображається у заголовку вікна

Нижче подано частину програмного коду форми.

```
public partial class FormMain : Form
    {
        public FormMain()
        {
            InitializeComponent();
        }
        // Обработка ввода только дефствительных чисел
```

```
      private void textBoxA_KeyPress(object sender,
KeyPressEventArgs e)
        {
            TextBox textBox = (TextBox)sender;
            if (char.IsControl(e.KeyChar))
                return;
            if (e.KeyChar.Equals('-') && textBox.Text.Length == 0)
                return;
            if (e.KeyChar.Equals('.') &&
!textBox.Text.Contains('.'))
                return;
            if (char.IsDigit(e.KeyChar))
                return;
            e.Handled = true;
        }
        private void buttonCalc_Click(object sender, EventArgs e)
        {
            try
            {
                double a = double.Parse(textBoxA.Text),
                       b = double.Parse(textBoxB.Text),
                       N = double.Parse(textBoxN.Text),
                       S = 0;
                for (int i = 0; i < N; i++)
                {
                    S += (a + i * i - Math.Sqrt(b)) /
                         (Math.Sin(i * Math.PI) +
Math.Sqrt(Math.PI));
                }
                labelRes.Text = String.Format("{0:f3}", S);
            }
            catch (FormatException)
            {
                MessageBox.Show("Проверьте параметры", "Ошибка");
            }
            catch (DivideByZeroException)
            {
                MessageBox.Show("Деление на ноль :(", "Ошибка");
            }
        }
```

Якщо користувач не вводить ніякі дані, то виникає помилка (рисунок 1.4)

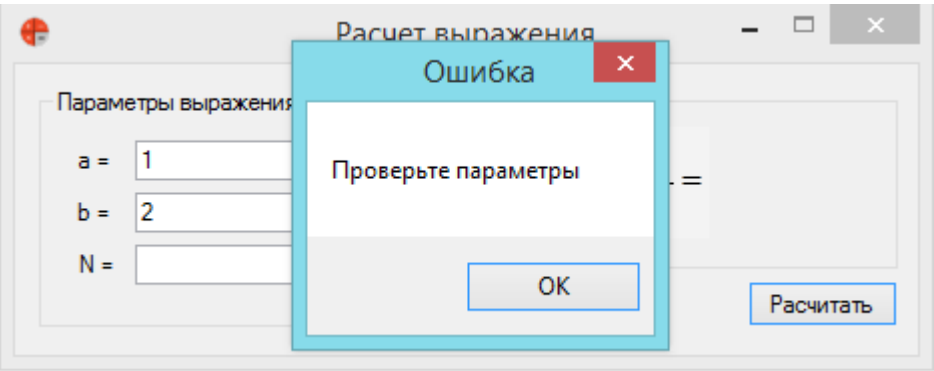


Рисунок 1.4 – Помилка роботи додатку

Зауваження: для вивчення деталей налаштування візуальних компонентів рекомендується скористатися даним проектом, який розміщено у архіві *Lab1.zip*.

Завдання до лабораторної роботи

Відповідно до варіанта та обраного рівня складності виконати наступні завдання.

Рівень завдань		
Початковий	Базовий	Високий
Відкомпілювати та протестувати аудиторне завдання	Завдання № 1	Завдання № 2

Завдання № 1

Створити додаток Windows Forms. Обчислити значення виразу, передбачивши обробку помилок та видачу відповідних повідомлень користувачу. Параметри виразу *x*, *y*, *z* задаються користувачем.

№ варіанта	Завдання
1	$\sum_{i=1}^{zy} \frac{x^z}{x^2 + y - iz}$

№ варіанта	Завдання
16	$\sum_{i=1}^{x-1} \frac{z \sin x}{i - y^2}$

№ варіанта	Завдання
2	$\sum_{i=1}^{x}(i-xy)\Big/\sum_{i=1}^{y}(i-xz)$
3	$\sum_{i=1}^{x-y}\frac{z\sin x}{i-y}+\sum_{i=1}^{x+z}\sqrt{\frac{1}{xy-z}}$
4	$\sum_{i=1}^{y+2}\frac{zx}{i-y^2}$
5	$\sum_{i=0}^{x+y+z}\frac{tg(y)\cdot ctg(x)}{i-2x}$
6	$\sum_{i=1}^{z}\frac{x^y+2}{x^i+y-z}$
7	$\sum_{i=1}^{z}\prod_{j=1}^{10}\frac{x-i+\cos j}{\sin i+j+z}$
8	$\sum_{i=1}^{x-1}\frac{z+\sin x}{i-y^2}+\sum_{i=1}^{4}\sqrt{\frac{1}{x+yz}}$
9	$\sum_{i=1}^{x}\sum_{j=2}^{y}\frac{x+y+2z}{i+z}$
10	$\sum_{i=0}^{xyz}\frac{i+x+y+z+2}{i-z}$
11	$\sum_{i=0}^{yx+z}\frac{\cos y\sin x}{i^2-y^2}$
12	$\prod_{i=x}^{x+y+z}\sqrt{\frac{i+10}{x-y+i}}$
13	$\sum_{i=1}^{x}\prod_{j=1}^{y}\frac{i-j}{i+j+z}$
14	$\sum_{i=1}^{x}\sum_{j=z}^{y}\frac{z+x}{i+yz}$
15	$\prod_{i=1}^{x+y}\sqrt{\frac{2i-zx}{x^3-i+i^2}}$

№ варіанта	Завдання
17	$\prod_{i=1}^{y}\sqrt{\frac{i+z}{x-y+i^2}}$
18	$\sum_{i=0}^{y-x}\frac{\cos y\sin x}{i^2-y^2}$
19	$\prod_{i=1}^{x+y+z}\sqrt{\frac{i+10}{x-y+i}}$
20	$\sum_{i=1}^{x}\prod_{j=1}^{y}\frac{i-j}{i+j+z}$
21	$\sum_{i=1}^{x}\sum_{j=2}^{y}\frac{z+x}{i+yz}$
22	$\prod_{i=1}^{x+y}\sqrt{\frac{2+i-z}{x^3-i+i^2}}$
23	$\sum_{i=1}^{z}\prod_{j=1}^{10}\frac{x-i+\cos j}{\sin i+j+z}$
24	$\sum_{i=4}^{z-x+y}\frac{z+x^i}{x+i-y}$
25	$\sum_{i=1}^{zy}\frac{x^y}{x^i+y-z}$
26	$\sum_{i=1}^{x}(i+x)\Big/\sum_{i=1}^{y}(i+xz)$
27	$\sum_{i=1}^{z}\prod_{j=1}^{i}\frac{x-i}{i+j+z}$
28	$\sum_{i=0}^{xyz}\frac{\cos y\sin x}{i-z}$
29	$\sum_{i=1}^{x}\prod_{j=1}^{i}\frac{xyz}{i+j+z}$
30	$\sum_{i=1}^{x-1}\frac{z\sin x}{i-y^2}+\sum_{i=1}^{8}\sqrt{\frac{1}{xyz}}$

Завдання № 2

Розробити програми типу Windows Forms відповідно до варіанта. Забезпечити перевірку на коректність введених даних, організувати зручний користувацький інтерфейс.

№ варіанта	Завдання
1	Розробити електронний калькулятор для виконання операцій +, –, *, / над дійсними числами. Реалізувати можливість розрахунку математичних виразів з операціями в дужках, наприклад, шляхом введення користувачем виразів вигляду 5*((1+2,5)-0,26)
2	Розробити програму «Телефонний довідник», яка повинна містити інформацію про прізвище, ім’я абонента, його номер телефону, адресу проживання. Забезпечити можливість збереження та завантаження файлу даних з жорсткого диску. Програма-довідник має дозволяти вносити нові дані до бази та редагувати існуючу інформацію
3	Розробити електронний калькулятор для виконання операцій *, /, % над цілими числами. Реалізувати можливість розрахунку математичних виразів з операціями в дужках, наприклад, шляхом введення користувачем виразів вигляду 20%(1+10/(3-1))
4	Розробити програму «Органайзер», яка відображає поточний час, дату, обсяг вільної оперативної пам’яті. Програма повинна згортатися у трей і при наведенні миші відображати підказкою всю інформацію. При натисненні комбінації Ctrl+Alt+F8 повинен запускатися стандартний калькулятор Windows
5	Розробити програму «Довідник студента», яка повинна містити інформацію про навчальні дисципліни, вид контролю, кількість лекцій та лабораторних занять. Забезпечити можливість збереження та завантаження

№ варіанта	Завдання
	файлу даних з жорсткого диску. Програма має дозволяти вносити нові дані до бази та редагувати існуючу інформацію
6	Розробити електронний калькулятор для виконання операцій +, –, *, / над цілими числами у шістнадцятковій системі числення. Реалізувати можливість розрахунку математичних виразів з операціями в дужках, наприклад, шляхом введення користувачем виразів вигляду F6*(F01C*(A+2)-1B)
7	Розробити програму для приховування номерів телефону. Користувач обирає деякий текстовий файл, якщо програма знаходить у ньому номер телефону у форматі +380XX-XXX-XX-XX, або XXX-XX-XX, або XX-XX-XX, то програма замінює три випадкові цифри номеру на зірочки * та зберігає файл під новим ім’ям
8	Розробити електронний калькулятор для виконання тригонометричних операцій *sin*, *cos*, *tg*, *ctg* над дійсними числами. Реалізувати можливість розрахунку математичних виразів з операціями в дужках, наприклад, шляхом введення користувачем виразів вигляду *cos*(*sin*(*tg*(0.5))
9	Розробити електронний калькулятор для виконання операцій +, –, *ln*, *lg* над дійсними числами. Реалізувати можливість розрахунку математичних виразів з операціями в дужках, наприклад, шляхом введення користувачем виразів вигляду *ln*(1.25 + *lg*(6-2.5))
10	Розробити електронний калькулятор для виконання операцій +, –, *, / над цілими числами у двійковій системі числення. Реалізувати можливість розрахунку математичних виразів з операціями в дужках, наприклад, шляхом введення користувачем виразів вигляду 100010*(110*(1+1001)-1010)

Контрольні питання

1. Перелічіть найбільш широко вживані візуальні компоненти Windows Froms?
2. Які властивості має виключення `Control`?
3. Як змінити координати візуального компонента у режимі виконання програми?
4. Які невізуальні елементи можна використати для створення мультимедійних додатків?
5. Пояснити принцип зворотної польської нотації.

ЛАБОРАТОРНА РОБОТА № 2

Тема: основи роботи з графікою засобами Windows Forms.

Мета: розробити програми типу Windows Forms Application на мові C# для роботи з графікою.

Теоретичні відомості

Перед початком виконання лабораторної роботи необхідно ознайомитися з теоретичними матеріалами з книги: C# и платформа .NET 4 для профессионалов / К. Нейгел, Б. Ивьен, Дж. Глини, К. Уотсон; пер. с англ. – М.: Вильямс, 2011. – Глава 48 (на диску).

Аудиторне завдання

Розробити програму типу Windows Forms Application та програмно намалювати червоним кольором зображення напівпровідникового діода.

Розв’язання

Зважаючи на необхідність звільнення пам’яті при використанні класу Pen, програмний код може мати наступний вигляд:

```
public partial class FormMain : Form
{
    Pen pen;

    public FormMain()
    {
        InitializeComponent();
        pen = new Pen(Color.Red, 3);
    }
   private void FormMain_Paint(object sender, PaintEventArgs e)
    {
        e.Graphics.DrawLine(pen, 50, 50, 80, 50);
```

```
            e.Graphics.DrawPolygon(pen, new Point[] {
                new Point(80, 30),
                new Point(80, 70),
                new Point(120, 50) });
            e.Graphics.DrawLine(pen, 120, 30, 120, 70);
            e.Graphics.DrawLine(pen, 120, 50, 150, 50);
        }
        private void FormMain_FormClosing(object sender,
                                          FormClosingEventArgs e)
        {
            pen.Dispose();
        }
}
```

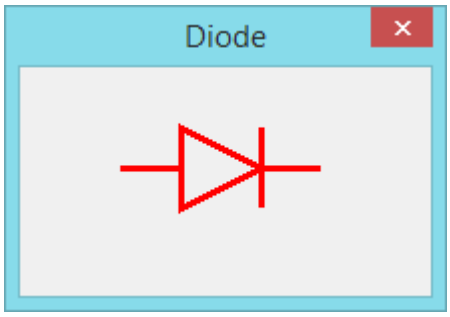


Рисунок 2.1 – Результат роботи програми

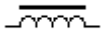
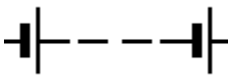
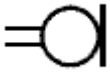
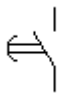
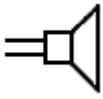
Завдання до лабораторної роботи

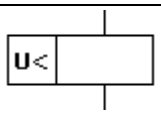
Відповідно до варіанта та обраного рівня складності виконати наступні завдання.

Рівень завдань		
Початковий	Базовий	Високий
Відкомпілювати та протестувати аудиторне завдання	Завдання № 1	Завдання № 2

Завдання № 1

Розробити програму типу Windows Forms Application та програмно намалювати зазначеним пером відповідне електричне умовне позначення елемента.

№ варіанта	Завдання		
	Елемент	Опис	Колір
1		транзистор n-p-n структури	синій
2		світлодіод	чорний
3		транзистор p-n-p структури	червони
4		трансформатор	білий
5		дросель	жовтий
6		конденсатор електролітичний	фіолетовий
7		батарея акумуляторна	рожевий
8		лампа індикаторна	коричневий
9		телефон	сірий
10		контакт	синій
11		мікрофон	чорний
12		контакт	червони
13		діод Шоткі	білий
14		гучномовець	жовтий

№ варіанта	Завдання		
	Елемент	Опис	Колір
15		контакт термореле	фіолетовий
16		реле	рожевий
17		зумер	коричневий
18		заземлення	сірий
19		розрядник	білий
20		амперметр	жовтий

Завдання № 2

Розробити графічний редактор, схожий за своїми функціональними можливостями на Windows Paint. Забезпечити наступні функції:

– малювання ліній, кіл, прямокутників;

– виведення поточних координат курсора;

– вибір кольору пера;

– збереження зображення у файл та завантаження файлу з жорсткого диску.

Додатково доповнити функціями відповідно до варіанта.

№ варіанта	Завдання
1	Забезпечити можливість виділення та переміщення фрагменту зображення
2	Забезпечити можливість заливки

№ варіанта	Завдання
3	Забезпечити можливість роботи з файлами форматів PNG, BMP, JPEG
4	Додати можливість малювання тексту
5	Створити пункт контекстного меню «Інвертувати колір» та реалізувати дану функцію
6	Додати можливість зміни яскравості зображення за допомогою колеса миші
7	Додати можливість малювання сітки із заданим кроком
8	Забезпечити можливість малювання шестикутника, інтерактивно змінюючи його радіус за допомогою миші
9	Додати до графічного редактора можливість використовувати буфер обміну
10	Забезпечити можливість малювання рівностороннього трикутника, інтерактивно змінюючи його розмір за допомогою миші

Контрольні питання

1. Які можливості надає клас Graphics?
2. Коротко охарактеризуйте класи Pen та Brush.
3. Якими властивостями потрібно керувати для зміни типу згладжування зображень?
4. Як програмно працювати з буфером обміну?
5. Як зробити градієнту заливку деякої фігури?

ЛАБОРАТОРНА РОБОТА № 3

Тема: програмування мережевих додатків на мові C#.

Мета: розробити клієнтський та серверний додаток для передачі даних із застосуванням протоколів TCP та UDP.

Теоретичні відомості

Мережні сокети та порти

Мережевий сокет (network socket) – це кінцева точка двухсторонньої комунікаційної взаємодії комп'ютерів в мережі.

Інтерфейсний сокет (socket API) – програмний інтерфейс, який надає операційна система, та дозволяє програмним додаткам керувати та використовувати мережні сокети.

Адреса сокету (socket address) – поєднання IP-адреси та номера порту. Базуючись на цій адресі інтернет-сокети доставляють відповідному програмному процесу чи потоку пакети даних, що надходять з мережі.

Слід розрізняти клієнтські і серверні сокети. Клієнтські сокети можна порівняти з кінцевими апаратами телефонної мережі, а серверні – з комутаторами. Клієнтський додаток, наприклад браузер, використовує тільки клієнтські сокети, а серверний, наприклад веб-сервер, якому браузер посилає запити – як клієнтські, так і серверні сокети.

Інтерфейс сокетів вперше з'явився в BSD Unix. Програмний інтерфейс сокетів описаний в стандарті POSIX.1 і в тій чи іншій мірі підтримується всіма сучасними операційними системами. Кожен процес може створити прослуховуючий сокет або серверний сокет і прив'язати його до якого-небудь порту операційної системи (рисунок 3.1). Процес, що прослуховує канал, зазвичай знаходиться в циклі очікування, тобто прокидається при появі нового з'єднання. При цьому зберігається можливість перевірити наявність з'єднань на даний момент, встановити тайм-аут для операції.

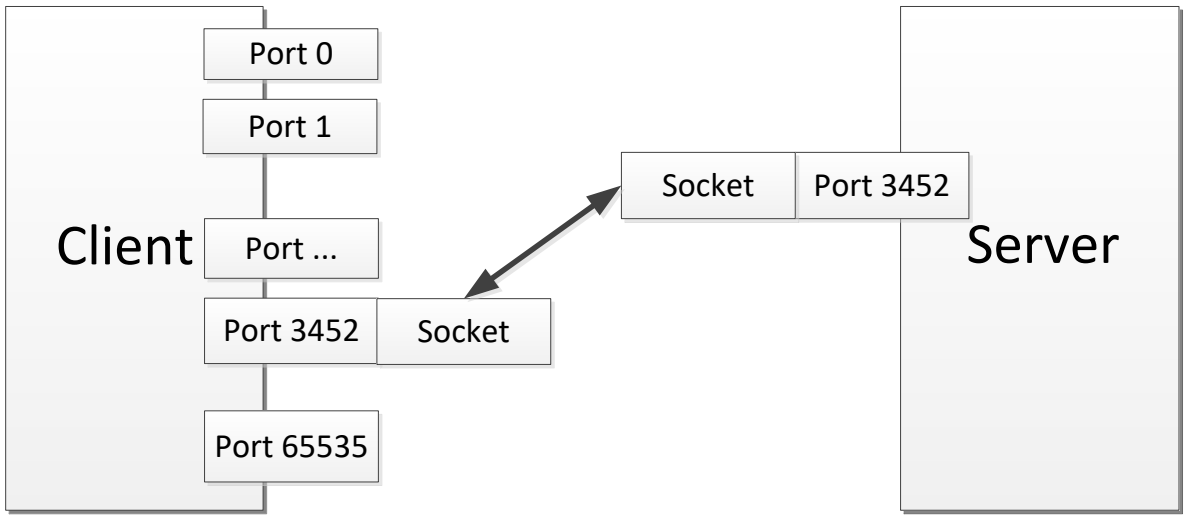


Рисунок 3.1 – Загальна схема роботи сокетів

Кожен сокет має свою адресу. Операційні системи сімейства UNIX можуть підтримувати багато типів адрес, але обов'язковими є INET-адреса і UNIX-адреса. Якщо прив'язати сокет за UNIX-адресою, то буде створений спеціальний файл або файл сокету по заданому шляху, через який зможуть «спілкуватися» будь-які локальні процеси шляхом читання/запису з нього. Сокети типу INET доступні з мережі і вимагають виділення номера порту. Зазвичай клієнт явно під'єднується до слухача, після чого будь-яке читання або запис через його файловий дескриптор буде передаватися між ним і сервером. Загальновідомі сокети представляють собою зручний механізм апріорного прив'язування адреси сокету до якого завгодно стандартного сервісу. Наприклад, процес-сервер для програми Telnet жорстко зв'язаний з конкретним сокетом (рисунок 3.2). Адреса сокету може бути зарезервована для доступу до процедури перегляду, яка могла б вказувати сокет, крізь який можна було б отримати новоутворені послуги.

У протоколах TCP/UDP порт – це системний ресурс, який має номер та виділяється програмі для зв'язку з додатками, що виконуються на інших мережних хостах (таблиця 3.1). Порт може бути зайнятий тільки однією програмою і в цей час не може

використовуватися іншою. Всі програми для зв'язку між собою за допомогою мережі використовують порти (до 65536 портів).

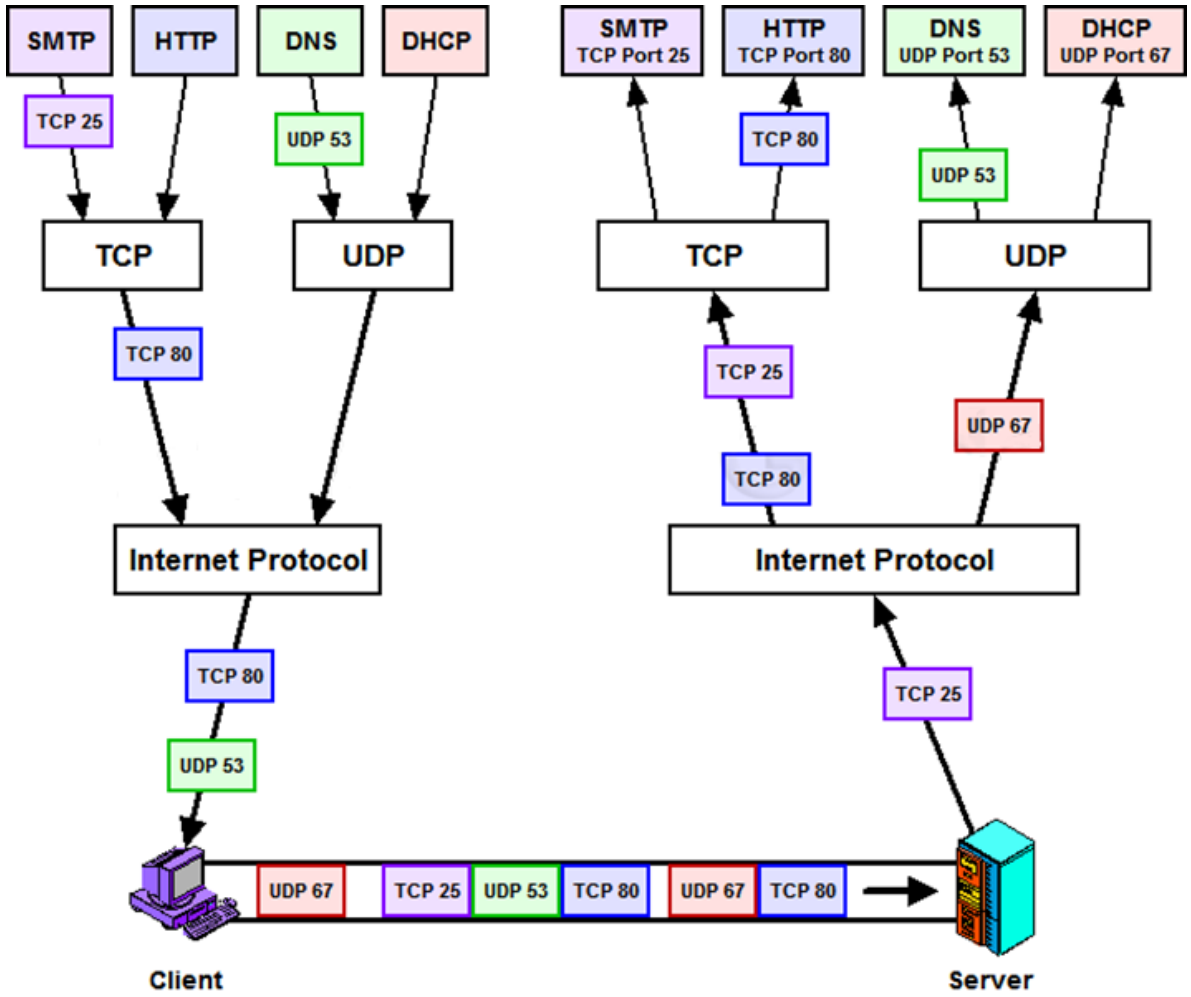


Рисунок 3.2 – Пояснення схеми роботи портів

Таблиця 3.1 – Деякі загальновідомі порти

Порт	TCP	UDP	Опис
0	√	√	Зарезервовано
20	√		Протокол FTP (дані)
21	√		Протокол FTP (команди)
22	√	√	Протокол SSH
23	√	√	Протокол Telnet
53		√	Протокол DNS
80	√	√	Протокол HTTP
443	√		Протокол HTTPS
6969	√		Клієнт BitTorrent
33434	√	√	Службова утиліта Traceroute

Стек протоколів TCP/IP. Протоколи TCP та UDP

TCP/IP – це абревіатура терміну Transmission Control Protocol / Internet Protocol (Протокол керування передачею / міжмережевий протокол). Фактично TCP/IP не один протокол, а декілька (таблиця 3.2). Саме тому ви часто чуєте, як його називають набором, або комплектом протоколів, серед яких TCP і IP – два основні. Фактично TCP/IP представляє цей базовий набір протоколів, відповідальний за розбивання вихідного повідомлення на пакети (TCP), доставку пакетів на вузол адресата (IP) і збирання (відновлення) вихідного повідомлення з пакетів (TCP).

Таблиця 3.2 – Співвідношення моделі OSI та стеку протоколів TCP/IP

<table>
<tr><th>OSI Model</th><th colspan="2">TCP/IP Model</th></tr>
<tr><td>Application</td><td rowspan="3">Application</td><td rowspan="3">Telnet, SMTP, POP3, FTP, HTTP, SNMP, DNS, SSH, …</td></tr>
<tr><td>Presentation</td></tr>
<tr><td>Session</td></tr>
<tr><td>Transport</td><td>Transport</td><td>TCP, UDP</td></tr>
<tr><td>Network</td><td>Internet</td><td>IP, ICMP, ARP, DHCP</td></tr>
<tr><td>Data Link</td><td rowspan="2">Network Access</td><td rowspan="2">Ethernet, ADSL, …</td></tr>
<tr><td>Physical</td></tr>
</table>

Прикладний рівень

Протоколи прикладного рівня TCP/IP визначають процедури стосовно організації взаємодії прикладних процесів (програм) різних мережних комп’ютерів і форми подання інформації за такої взаємодії. По ознакам взаємодії прикладних процесів виділяють два типи прикладного програмного забезпечення: програма-клієнт та програма-сервер. Протоколи прикладного рівня зорієнтовано на конкретні прикладні завдання. Серед традиційних послуг, котрі забезпечують протоколи прикладного рівня із сімейства TCP/IP, сьогодні найпопулярнішими є електронна пошта – протоколи SMTP та POP3,

передавання файлів – FTP та TFTP, емуляції віддаленого терміналу – TELNET тощо.

В інтернеті активно використовуються послуги, які базуються на технології WWW, яка грунтується на протоколі передавання гіпертексту HTTP. Сьогодні є популярні послуги пакетної IP-телефонії на базі стандартів IETF, яку стосуються спеціальних протоколів прикладного, транспортного і мережного рівнів, наприклад сигналізації SIP, передавання в режимі реального часу RTP та RTCP, резервування ресурсів RSVP, рекомендацій ITU H.323 тощо.

Транспортний рівень

Протоколи транспортного рівня TCP/IP надають транспортні послуги прикладним процесам. Основними протоколами транспортного рівня TCP/IP є протокол керування передаванням TCP (Transmission Control Protocol) і протокол користувальницьких дейтаграм UDP (User Datagram Protocol). Транспортні послуги цих протоколів суттєво відрізняються. Протокол UDP доставляє датаграми без установлення з'єднання. При цьому він не гарантує їхнього доставляння. Протокол TCP забезпечує надійне доставляння байтових потоків (сегментів) із попереднім встановленням транспортного дуплексного з'єднання (віртуального каналу) між модулями TCP мережних комп'ютерів. Для розв'язання транспортних завдань протоколи TCP та UDP в перебігу передавання даних формують і додають до даних свої заголовки обсягом 20 байт та 8 байт відповідно.

Кожен прикладний процес взаємодіє з модулем транспортного рівня TCP або UDP через окремий порт, що дозволяє при взаємодії систем однозначно ідентифікувати прикладні процеси. Ці порти нумеруються починаючи з нуля. При передаванні запиту прикладної програми клієнта до прикладної програми сервера транспортний модуль, формуючи датаграму чи сегмент, вказує номери портів програмних модулів прикладних протоколів сервера й клієнта. З цією метою в заголовку пакета протоколу транспортного рівня виділено два поля – «порт одержувача» і «порт відправника», обсягом по 2 байти.

Номери портів TCP та UDP до прикладних протоколів сервера стандартизовано IETF. Для цього надано номери в діапазоні від 1 до 1023. Наприклад, програмний модуль TCP сервера взаємодіє з модулем протоколу HTTP через порт з номером 80. Взаємодія модуля TCP чи UDP клієнта з будь-яким модулем прикладного протоколу відбувається через порт, якому надається вільний номер, за значенням більший ніж 1023.

Мережний рівень

Протоколи мережного рівня TCP/IP забезпечують взаємодію поміж мережами різної архітектури тощо. Основним протоколом мережного рівня технології TCP/IP є міжмережний протокол IP та його допоміжні протоколи: адресний протокол ARP; реверсний адресний протокол RARP (Reverse ARP); протокол діагностичних повідомлень ICMP (Internet Control Message Protocol), який надсилає повідомлення вузлам мережі про помилки на маршруті, які виникають при передаванні пакетів тощо.

Головне завдання міжмережного протоколу IP – це маршрутизація пакетів даних поміж різнотипними комп'ютерними мережами. Для розв'язання цього завдання протокол IP підтримує IP-адресацію мереж та вузлів, використовує таблицю маршрутизації пакетів, виконує, за необхідності, фрагментацію та дефрагментацію цих пакетів.

Функціонування мережного рівня також забезпечує низка протоколів динамічної маршрутизації RIP, OSPF, які динамічно формують маршрути таблиці маршрутизації за алгоритмами вектора VDA (Vector Distance Algorithm) і стану зв'язку LSA (Link State Algorithm) відповідно, протоколів політики зовнішньої маршрутизації EGP (Exterior Gateway Protocol), BGP (Border Gateway Protocol) тощо.

Засоби мережевого рівня забезпечують доставку даних між пристроями в складових мережі, а саме комп'ютерами, маршрутизаторами і т.д. Однак не слід забувати, що на одному вузлі може функціонувати паралельно декілька програм, яким потрібен доступ до мережі. Отже, дані всередині комп'ютерної системи повинні

розподілятися між програмами. Тому, при передачі даних по мережі недостатньо просто адресувати конкретний вузол. Необхідно також ідентифікувати програму-одержувача, що неможливо здійснити засобами мережевого рівня.

Іншою серйозною проблемою протоколів мережевого рівня є відсутність засобів, що дозволяють передавати великі масиви даних. Коли вихідні дані перевищують максимально допустимий розмір пакета мережного рівня, то ці дані повинні бути розбиті на порції, кожна з яких передається в мережу окремим пакетом. Проте кожен пакет мережевого рівня передається по мережі як єдиний, незалежний від інших блок даних. У разі якщо будь-які пакети "загубилися", то модуль мережевого протоколу на приймаючій стороні не зможе виявити втрату, і, отже – виявити порушення цілісності загального масиву даних. Тому кошти транспортного рівня забезпечують відсутність втрат інформації. Такий режим передачі даних отримав назву гарантованої доставки.

Таким чином, засоби транспортного рівня представляють собою функціональну надбудову над мережним рівнем і вирішують дві основні задачі:

1) забезпечення доставки даних між конкретними програмами, що функціонують, в загальному випадку, на різних вузлах мережі;

2) забезпечення гарантованої доставки масивів даних довільного розміру.

В даний час в Інтернет використовуються два транспортних протоколу – UDP, що забезпечує негарантовану доставку даних між програмами, і TCP, що забезпечує гарантовану доставку з встановленням віртуального з’єднання.

Для ідентифікації програм протоколи транспортного рівня в мережі Інтернет (TCP і UDP), використовують унікальні числові значення, так звані порти. Номери портів призначаються програмами відповідно до її функціонального призначення на основі певних стандартів. Для кожного протоколу існують стандартні списки відповідності номерів портів і програм. Так, наприклад, програмне забезпечення WWW, що працює через транспортний протокол TCP,

використовує TCP-порт 80, модулі протоколу FTP – TCP-порт 21, а служба DNS взаємодіє з транспортними протоколами TCP і UDP через TCP-порт 53 і UDP-порт 53 відповідно.

Таким чином, протокол мережевого рівня IP і транспортні протоколи TCP і UDP реалізують дворівневу схему адресації: номери TCP-і UDP-портів дозволяють однозначно ідентифікувати програму в рамках вузла, а сам вузол однозначно визначається IP-адресою. Отже, комбінація IP-адреси і номера порту дозволяє однозначно ідентифікувати програму в мережі Інтернет. Така комбінована адреса називається сокетом (socket).

Протокол UDP (User Datagram Protocol) – протокол транспортного рівня, що входить в стек протоколів TCP/IP, що забезпечує негарантовану доставку даних без встановлення віртуального з'єднання (рисунок 3.3).

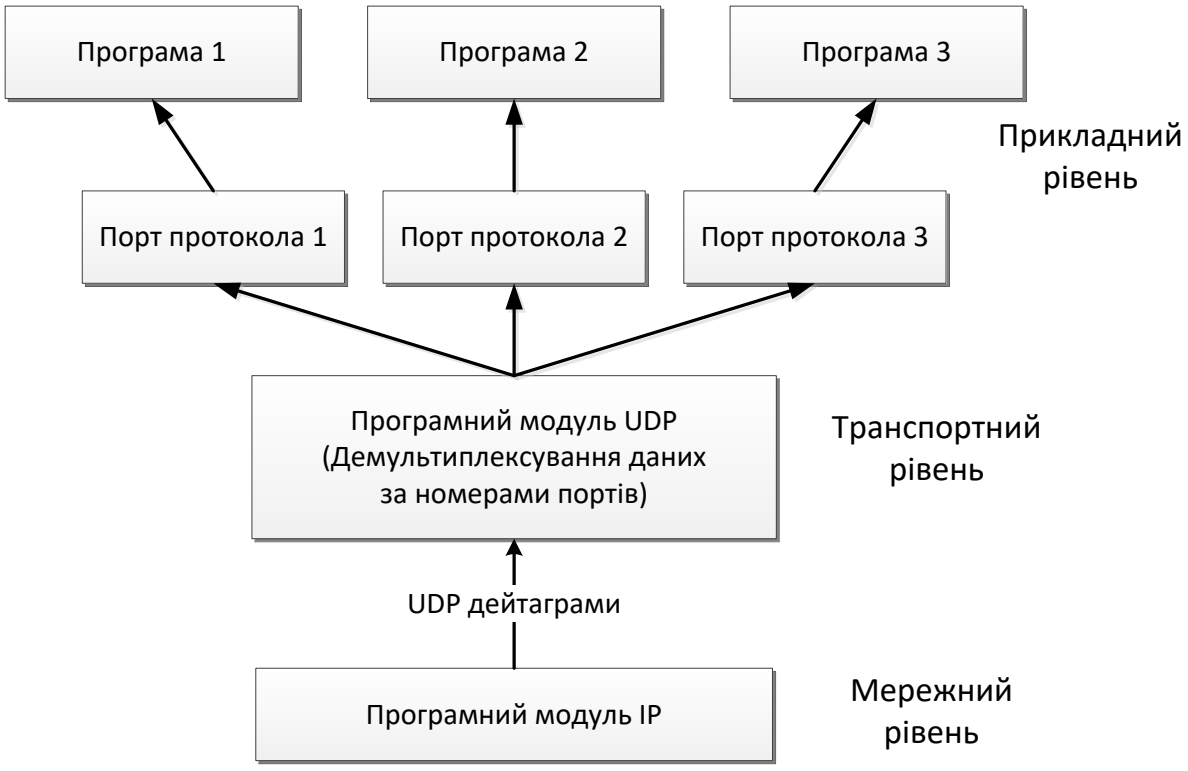


Рисунок 3.3 – Робота додатку за UDP протоколом

Оскільки на протокол не покладається завдань щодо забезпечення гарантованої доставки, а лише потрібно забезпечувати зв'язок між

різними програмами, то структура заголовка дейтаграми UDP, таку назву має пакет протоколу, виглядає досить просто – вона включає в себе всього чотири поля. Перші два поля містять номери UDP-портів програми-відправника та програми-одержувача. Два інших поля в структурі заголовка дейтаграми призначені для управління обробкою – це загальна довжина дейтаграми і контрольна сума заголовка.

Протокол TCP (Transmission Control Protocol) є транспортним протоколом стека протоколів TCP/IP, що забезпечує гарантовану доставку даних з встановленням віртуального з'єднання (рисунок 3.4).

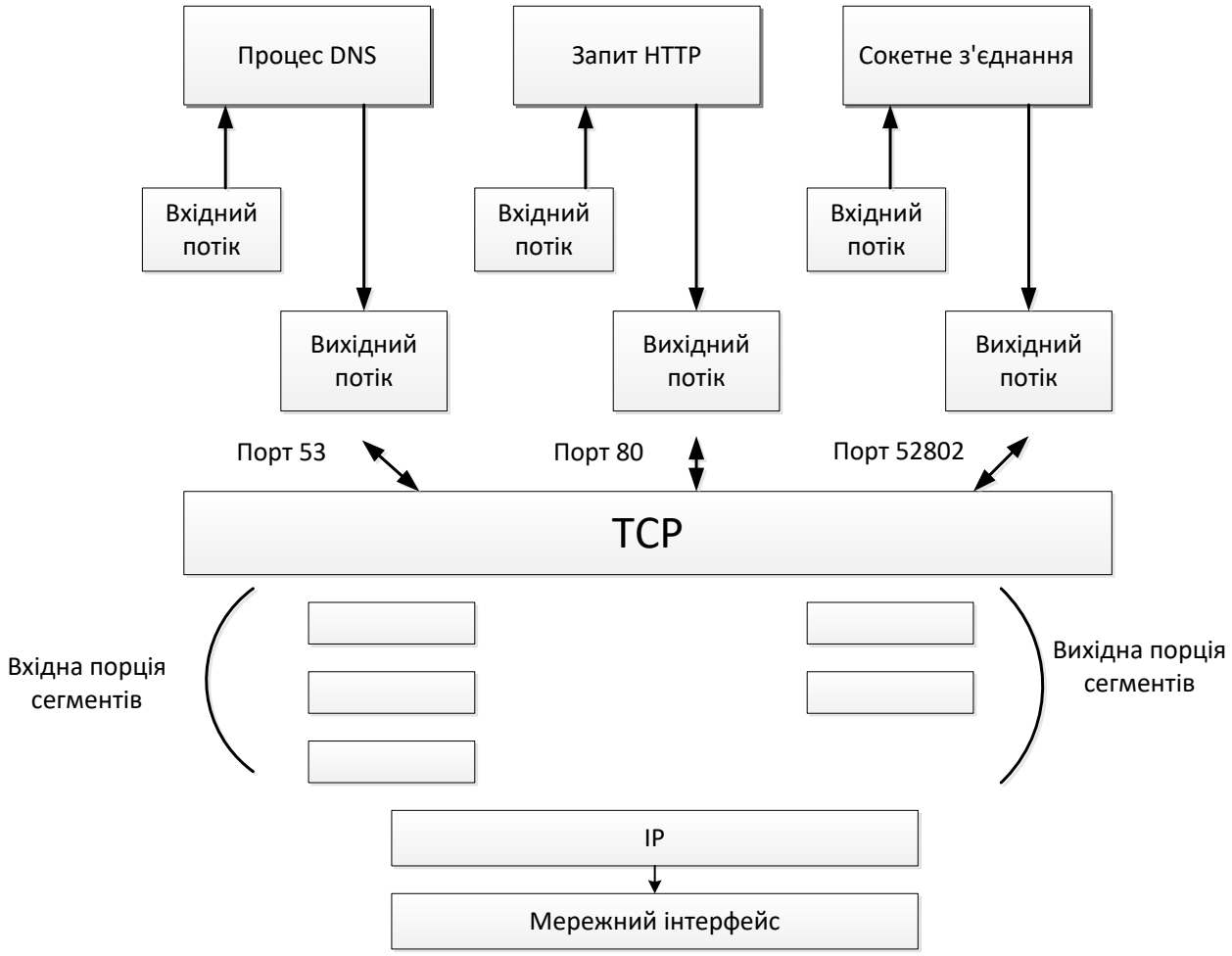


Рисунок 3.4 – Робота додатку за TCP протоколом

Протокол надає програмам, що використовують його, можливість передачі безперервного потоку даних. Дані, що підлягають відправці в мережу, розбиваються на порції, кожна з яких забезпечується

службовою інформацією, тобто формуються пакети даних. У термінології TCP пакет називається сегментом.

Відповідно до функціонального призначення протоколу структура TCP-сегмента передбачає наявність наступних інформаційних полів:

1) номер порту-відправника і номер порту-одержувача – номери портів, що ідентифікують програми, між якими здійснюється взаємодія;

2) поля, призначені для забезпечення гарантованої доставки: розмір вікна, номер послідовності і номер підтвердження;

3) керуючі прапори – спеціальні бітові поля, що управляють протоколом.

Для забезпечення гарантованої доставки протокол TCP використовує механізм відправки підтвердження. З метою зниження завантаження мережі протокол TCP допускає посилку одного підтвердження відразу для декількох отриманих сегментів. Обсяг даних, які можуть бути передані в мережу відправником до отримання підтвердження, визначається спеціальним параметром протоколу TCP – розміром вікна. Розмір вікна узгоджується при встановленні з'єднання між відправником та одержувачем і може автоматично змінюватися програмними модулями протоколу TCP в залежності від стану каналу зв'язку. Якщо в процесі передачі даних втрати відбуваються досить часто, то розмір вікна зменшується, і навпаки – вікно може мати великий розмір, якщо висока надійність каналу даних.

Для того, щоб дані могли бути правильно зібрані одержувачем в потрібному порядку, в заголовку TCP-сегмента присутня інформація, яка визначає положення вкладених даних в загальному потоці. Відправляючи підтвердження, одержувач вказує положення даних, які він очікує отримати в наступному сегменті, тим самим побічно повідомляючи відправнику, який фрагмент загального потоку був успішно прийнятий. Відповідні поля заголовка TCP-сегмента отримали назву номер послідовності і номер підтвердження.

Таблиця 3.3 – Порівняння протоколів TCP та UDP

TCP	UDP
Надійний протокол	Ненадійний протокол
Встановлення віртуального з’єднання	Без встановлення віртуального з’єднання
Повторна передача сегментів та керування потоком	Без повторної передачі повідомлень
Використовується впорядкування сегментів	Випадковий порядок отримання сегментів
Для контролю слугують сегменти підтвердження	Підтвердження передачі відсутнє

Перед початком передачі потоку даних абоненти повинні узгодити параметри передачі: розмір вікна та початкові номери послідовностей, щодо яких буде відраховуватися положення переданих в сегментах даних усередині загального потоку. Очевидно, що таке узгодження передбачає обмін спеціальними сегментами і виділення ресурсів, зокрема, блоків пам’яті, необхідних для прийому та обробки даних і підтверджень. Відповідна послідовність дій називається встановленням віртуального з’єднання.

Розглянемо схему створення TCP-з’єднання (рисунок 3.5).

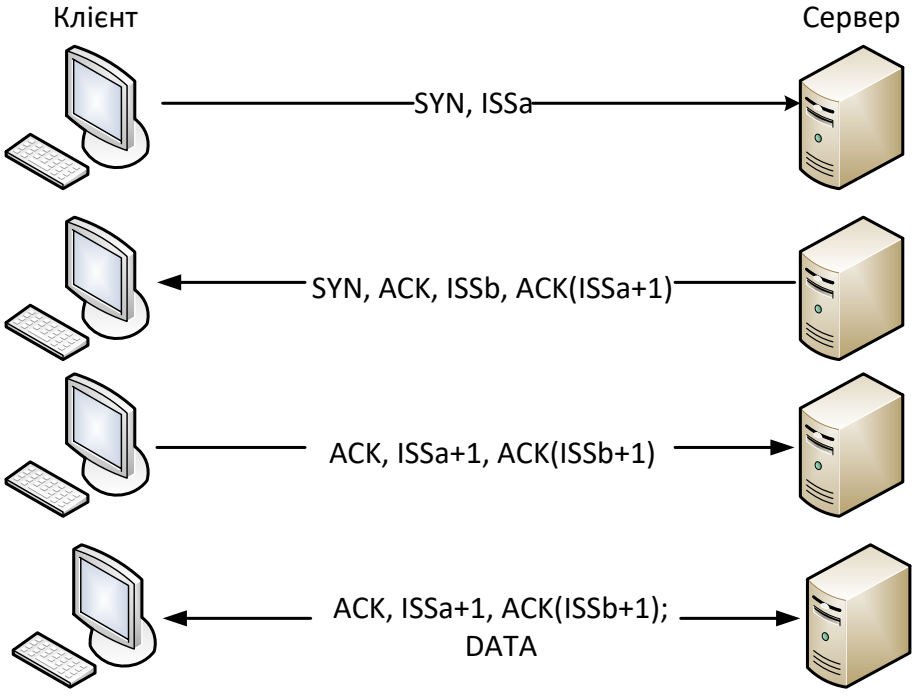


Рисунок 3.5 – Схема створення сокетного TCP з’єднання

Припустимо, що клієнту A необхідно створити TCP-з'єднання з сервером B. Тоді A посилає на B наступне повідомлення: SYN, ISSa. Це означає, що в повідомленні, яке передає A, встановлений біт SYN (synchronize sequence number), а у поле Sequence Number встановлено початкове 32-бітне значення ISSa (Initial Sequence Number). Далі B відповідає: SYN, ACK, ISSb, ACK(ISSa+1).

У відповідь на отриманий від A запит B відповідає повідомленням, в якому встановлений біт SYN та встановлений біт ACK; у поле Sequence Number сервером B встановлюється своє початкове значення лічильника – ISSb. Поле Acknowledgment Number містить значення ISSa, отримане у першому пакеті від клієнта A та збільшене на одиницю. A, завершуючи рукостискання (handshake), надсилає: ACK, ISSa+1, ACK(ISSb+1).

У цьому пакеті встановлений біт ACK. Поле Sequence Number містить ISSa + 1. Поле Acknowledgment Number містить значення ISSb + 1. Відсиланням цього пакету на B закінчується трьохступеневий handshake та TCP-з'єднання між A та B вважається встановленим. Тепер клієнт може посилати пакети з даними на B по щойно створеному віртуальному TCP-каналу: ACK, ISSa+1, ACK(ISSb+1); DATA.

Щоб ідентифікувати окремі потоки даних, які підтримує протокол TCP, останній визначає ідентифікатори портів. Оскільки ідентифікатори портів обираються кожною програмою протокола TCP незалежно, то вони не будуть унікальними. Щоб забеспечити унікальність адрес для кожної програми протокола TCP, треба об'єднати ідентифікуючу цю програму Internet адресу та ідентифікатор порта. В результаті отримуємо сокет, який буде унікальний у всіх локальних мережах, об'єднаних у єдине ціле.

З'єднання повністю визначається парою сокетів на своїх кінцях. Локальний сокет може брати участь у багатьох з'єднаннях з різноманітними чужими сокетами. З'єднання можна використовувати для передачі даних у обох напрямках, іншими словами, воно є повністю дуплексним.

Протокол TCP може довільним чином зв’язувати порти з процесами. Проте при будь-якій реалізації протоколу необхідно дотримуватися деяких основних концепцій. Мають бути присутні загальновідомі сокети, які протокол TCP асоціює виключно з відповідними їм процесами.

З’єднання задається командою OPEN, виконаною з локального порту та має аргументом чужий сокет. У відповідь на такий запит програма протокола TCP надає ім’я локального з’єднання. За цим ім’ям користувач адресується до даного з’єднання при наступних викликах. Існує певна структура даних, що має назву блок управління передачей або Transmission Control Block (TCB), призначена для збереження описаної вище інформації. Можна реалізувати протокол таким чином, щоб локальне ім’я для з’єднання було б вказівником на структуру TCB останнього. Запит OPEN вказує також, чи здійснюється з’єднання активним чином, чи проходить пасивне очікування з’єднання ззовні.

Запит на пасивне відкриття з’єднання означає, що процес чекає отримання ззовні запитів на з’єднання, замість того, щоб намагатися самому встановити його. Часто процес, що зробив запит на пасивне відкриття, буде приймати запити на з’єднання від будь-якого іншого процесу. У цьому випадку чужий сокет вказується як такий, що складається повністю з нулей, що означає невизначеність. Невизначені чужі сокети можуть бути присутні лише в командах пасивного відкриття. Сервісний процес, що бажає обслужити інші, невідомі йому процеси, міг би здійснити запит на пасивне відкриття з вказанням невизначеного сокета. У цьому випадку з’єднання може бути встановлене з будь-яким процесом, що запросив з’єднання з цим локальним сокетом. Така процедура буде корисною, якщо відомо, що обраний локальний сокет асоційований з певним сервісом.

Мережні засоби платформи .NET Framework

У таблиці 3.4 коротко подано перелік класів бібліотеки .NET, з використанням яких можна побудувати мережний додаток.

Таблиця 3.4. – Основні класи для роботи з сокетами в .NET

Клас .NET	Опис
`Socket`	Забезпечує базову функціональність сокета для додатка
`TcpClient`	Побудований на класі Socket, щоб забезпечити TCP обслуговування на більш високому рівні. TcpClient надає декілька методів для відправки та отримання даних мережею
`TcpListener`	Побудований на низькорівневому класі Socket. Його основне призначення – серверні додатки. Він очікує вхідні запити на з'єднання від клієнтів та повідомляє додатки про будь-які з'єднання
`UdpClient`	Призначений для реалізації UDP обслуговування
`SocketException`	Цей виняток генерується, коли у сокеті виникає помилка

Базовим класом при подубові мережних додатків є клас System.Net.Sockets.Socket, деякі властивості та методи якого представлено в таблиці 3.5.

Таблиця 3.5 – Короткий опис класу `System.Net.Sockets.Socket`

Властивості та методи	Короткий опис та призначення
`AddressFamily`	Сімейство адрес сокета (значення із перерахунку Socket.AddressFamily)
`Available`	Об'єм даних, які можна зчитати
`Blocking`	Чи знаходиться сокет в блокуючому режимі
`Connected`	Чи з'єднаний сокет з віддаленим хостом
`LocalEndPoint`	Локальна кінцева точка сокета
`ProtocolType`	Тип протокола сокета
`RemoteEndPoint`	Віддалена кінцева точка сокета
`SocketType`	Тип сокета
`Accept()`	Створює новий сокет для обробки вхідного запиту на зєднання

Властивості та методи	Короткий опис та призначення
`Bind()`	Звязує сокет з локальною кінцевою точкою для очікування вхідних запитів на зєднання
`Close()`	Закриває сокет
`Connect()`	Встановлює з’єднання з віддаленим хостом
`Listen()`	Переводить сокет в режим прослуховування
`Receive()`	Отримує дані від приєднаного сокета
`Poll()`	Визначає статус сокета
`Send()`	Відправляє дані з’єднаному сокету
`ShutDown()`	Забороняє операції відправки та отримання даних на сокеті

Аудиторні завдання

TCP сервер, побудований з використанням класу `Socket`

```
static void Main(string[] args)
{
    //Sockets.NET Server

    IPEndPoint ipEndPoint =
        new IPEndPoint(IPAddress.Parse("127.0.0.1"),
8000);
    Socket serverSock = new
Socket(AddressFamily.InterNetwork,
                        SocketType.Stream,
ProtocolType.Tcp);
    serverSock.Bind(ipEndPoint);
    Console.WriteLine("Ожидание входящего TCP
подключения...");
    serverSock.Listen(10);

    Socket clientSock = serverSock.Accept();
```

```
    IPEndPoint remote =
(IPEndPoint)clientSock.RemoteEndPoint;
    Console.WriteLine("Соединение установлено по
адресу {0}.
                Входящее сообщение:",
remote.Address.ToString());
    byte[] bufRecieve = new byte[8];
    clientSock.Receive(bufRecieve);

Console.WriteLine(Encoding.UTF8.GetString(bufRecieve))
;
    Console.WriteLine("Для завершения нажмите любую
клавишу...");
    Console.ReadKey();
    serverSock.Close();
    clientSock.Close();
}
```

TCP клієнт, побудований з використанням класу `Socket`

```
static void Main(string[] args)
{
    //NET.Socket Client
    Socket cliSock = new
Socket(AddressFamily.InterNetwork,
                    SocketType.Stream,
ProtocolType.Tcp);
    IPAddress ipServ = IPAddress.Parse("127.0.0.1");
    IPEndPoint ipEndP = new IPEndPoint(ipServ, 8000);
    try
    {
        cliSock.Connect(ipEndP);
    }
    catch (SocketException ex)
    {
```

```
        Console.WriteLine(ex);
    }
    byte[] pack = Encoding.UTF8.GetBytes("Hello");
    cliSock.Send(pack);
    Console.WriteLine("Передача завершена. Для
продолжения нажмите любую клавишу...");
    Console.ReadKey();
    cliSock.Close();
  }
```

TCP сервер, побудований з використанням класу `TcpListener`

```
static void Main(string[] args)
{
    //TCPListener .NET
    TcpListener tcpServer =
       new    TcpListener(IPAddress.Parse("127.0.0.1"),
8000);
    tcpServer.Start();
    Console.WriteLine("Ожидание подключения...");
    TcpClient tcpClient = tcpServer.AcceptTcpClient();
    Console.WriteLine("Подключение установлено,
                       входное сообщение:");
    byte[] buffer = new byte[8];
    NetworkStream streamTcp = tcpClient.GetStream();
    streamTcp.Read(buffer, 0, buffer.Length);

Console.WriteLine(Encoding.UTF8.GetString(buffer));
    streamTcp.Close();
    tcpClient.Close();
    tcpServer.Stop();
    Console.WriteLine("Нажмите    любую    клавишу    для
завершения...");
    Console.ReadKey();
}
```

TCP клієнт, побудований з використанням класу `TcpListener`

```
static void Main(string[] args)
{
    //TCPClient .NET
    TcpClient clientTcp = new TcpClient();
    clientTcp.Connect(IPAddress.Parse("127.0.0.1"),
8000);
    NetworkStream streamTcp = clientTcp.GetStream();
    byte[] buffer = new byte[8];
    buffer = Encoding.UTF8.GetBytes("Hello");
    streamTcp.Write(buffer, 0, buffer.Length);
    streamTcp.Close();
    clientTcp.Close();
    Console.WriteLine("Передача выполнена.
                        Нажмите любую клавишу для
завершения...");
    Console.ReadKey();
}
```

UDP сервер

```
static void Main(string[] args)
{
    byte[] buffer = new byte[8];
    IPEndPoint ipEndpoint = new
IPEndPoint(IPAddress.Any, 8000);
    Socket serverSock = new
Socket(AddressFamily.InterNetwork,
                          SocketType.Dgram,
ProtocolType.Udp);
    serverSock.Bind(ipEndpoint);
    Console.WriteLine("Ожидание подключения UDP
протокол...");
    serverSock.Receive(buffer);
    Console.WriteLine("Передача данных UDP,
сообщение:");
```

```
Console.WriteLine(Encoding.UTF8.GetString(buffer));
    serverSock.Close();
    Console.WriteLine("Для завершения нажмите любую
клавишу...");
    Console.ReadKey();
}
```

UDP клієнт

```
static void Main(string[] args)
{
    //UDP Server based on Sockets.NET
    byte[] buffer = new byte[8];
    IPEndPoint ipEndpoint = new
IPEndPoint(IPAddress.Any, 8000);
    Socket serverSock = new
Socket(AddressFamily.InterNetwork,
                        SocketType.Dgram,
ProtocolType.Udp);
    serverSock.Bind(ipEndpoint);
    Console.WriteLine("Ожидание подключения UDP
протокол...");
    serverSock.Receive(buffer);
    Console.WriteLine("Передача данных UDP,
сообщение:");

Console.WriteLine(Encoding.UTF8.GetString(buffer));
    serverSock.Close();
    Console.WriteLine("Для завершения нажмите любую
клавишу...");
    Console.ReadKey();
}
```

Завдання до лабораторної роботи

Відповідно до варіанта та обраного рівня складності виконати наступні завдання.

Рівень завдань		
Початковий	Базовий	Високий
Відкомпілювати та протестувати аудиторні завдання	Завдання №1	Завдання №2
	Завдання №2	Завдання №3

Завдання № 1

Розробити клієнтський та серверний додаток шляхом внесення незначних змін в програми аудиторного завдання.

№ варіанта	Завдання
1	Ввести своє ім’я та вік з консолі, відправити їх з клієнта на сервер.
2	Відправити рядок тексту з сервера на клієнт, на клієнті вивести довжину рядка в символах.
3	Відправити масив із 10 цілих чисел з клієнта на сервер.
4	Відправити масив із 5 чисел типу `double` з сервера на клієнт.
5	Відправити два числа з клієнта на сервер та знайти їх добуток.
6	Відправити три числа з сервера на клієнт та знайти їх середнє арифметичне значення.
7	Відправити з сервера на клієнт повідомлення та вивести на клієнті його у зворотному порядку.
8	Відправити своє прізвище, ім’я, по батькові від клієнта до сервера та вивести на консоль.
9	Знайти максимальний елемент масиву, відправити масив та це значення з серверу на клієнт.
10	Відправити свою IP-адресу на сервер та знайти суму октетів на ньому.

№ варіанта	Завдання
11	Ввести своє ім’я та вік з консолі, відправити їх з сервера на клієнт.
12	Відправити з сервера на клієнт повідомлення та вивести на клієнті його прописними літерами.
13	Відправити масив із 8 беззнакових цілих чисел з клієнта на сервер.
14	Відправити масив із 6 чисел типу `byte` з клієнта на сервер.
15	Ввести своє прізвище та групу з консолі, відправити їх з клієнта на сервер.
16	Відправити рядок текста з клієнта на сервер, на сервері вивести довжину рядка в символах.
17	Відправити масив із 5 символів з клієнта на сервер.
18	Відправити масив із 10 чисел типу `double` з клієнта на сервер.
19	Відправити два числа з клієнта на сервер та знайти остачу від їх частки.
20	Відправити три числа з сервера на клієнт та знайти їх середнє геометричне значення.
21	Відправити з сервера на клієнт повідомлення та вивести на клієнті його у без пробілів.
22	Відправити своє прізвище ім’я, по батькові від клієнта до сервера та вивести на консоль прізвище та ініціали.
23	Відправити масив з серверу на клієнт та знайти мінімальний елемент масиву.
24	Відправити свою IP-адресу на сервер та знайти на ньому побітову операцію АБО октетів.
25	Ввести своє прізвище та факультет з консолі, відправити їх з сервера на клієнт.
26	Відправити три числа з клієнта на сервер та знайти суму тільки додатніх.
27	Відправити два рядки текста з сервера на клієнт, на клієнті вивести довжину рядків в символах.

№ варіанта	Завдання
28	Відправити масив із 10 символів з клієнта на сервер, на сервері змінити їх регістр на протилежний.
29	Відправити масив із 5 чисел типу `float` з клієнта на сервер.
30	Відправити два числа з клієнта на сервер та знайти їх різницю.

Завдання № 2

Розробити клієнтський та серверний додаток, обґрунтувавши обраний протокол передачі даних TCP або UDP.

№ варіанта	Завдання
1	Відправити з клієнта на сервер своє імя, на сервері додати до імені своє прізвище та відправити результат на клієнт.
2	Відправити на сервер два числа, знайти їх суму та повернути результат клієнту.
3	Відправити на клієнт два числа, знайти їх добуток та повернути результат серверу.
4	Відправити на сервер масив, знайти його максимальний елемент та повернути результат клієнту.
5	Відправити на сервер число, сервер перевіряє, чи є воно більшим за введене на ньому значення, якщо так, то повертає на клієнт своє число, інакше повертає клієнту його ж число.
6	Відправити на сервер число, сервер перевіряє, чи є це число паліндромом. Якщо так, то відправляє на клієнт «Yes», якщо ні, то повертає «No».
7	Відправити з сервера на клієнт масив чисел, відсортувати їх за зростанням та повернути на сервер.
8	Відправити на сервер повідомлення, знайти у ньому кількість літер «а», та відправити на клієнт результат.

№ варіанта	Завдання
9	Відправити на клієнт повідомлення, знайти його довжину і відправити відповідну кількість нулів на сервер.
10	Відправити повідомлення на сервер, перевірити чи є в ньому цифри, якщо так, то відправити повідомлення «Yes» на клієнт, інакше «No»
11	Відправити з клієнта на сервер своє прізвище, на сервері додати до прізвища своє ім’я та відправити результат на клієнт.
12	Відправити на сервер три числа, знайти їх суму та повернути результат клієнту.
13	Відправити на клієнт три числа, знайти добуток від’ємних чисел та повернути результат серверу.
14	Відправити на сервер масив, знайти його мінімальний та максимальний елементи та повернути результати клієнту
15	Клієнт відправляє чотири числа на сервер, сервер додає до чисел їх середнє значення та повертає клієнту.
16	Відправити на сервер число, сервер перевіряє, чи є це число простим. Якщо так, то відправляє на клієнт «Yes», якщо ні, то повертає «No»
17	Відправити з сервера на клієнт масив цілих чисел, який вводить користувач, відсортувати їх за спаданням та повернути на сервер.
18	Відправити на сервер повідомлення, знайти у ньому кількість пробілів та відправити на клієнт результат.
19	Відправити на клієнт повідомлення, знайти його довжину і відправити відповідну кількість символів «X» на сервер.
20	Відправити повідомлення на сервер, перевірити чи є в ньому пробіли, якщо так, то відправити повідомлення «Yes» на клієнт, інакше «No».
21	Відправити на сервер число, розкласти його на прості множники, результат повернути клієнту у вигляді масиву.

№ варіанта	Завдання
22	Відправити на сервер число, сервер перевіряє, чи є це число квадратом іншого цілого числа. Якщо так, то відправляє на клієнт «Yes», якщо ні, то повертає «No».
23	Відправити з сервера на клієнт масив чисел, замінити від'ємні значення нулем та повернути на сервер.
24	Відправити на сервер повідомлення, знайти у ньому кількість голосних літер та відправити на клієнт результат.
25	Відправити на клієнт повідомлення, знайти його довжину і відправити відповідну кількість випадкових чисел на сервер.
26	Відправити повідомлення на сервер, перевірити чи містить воно знаки пунктуації, якщо так, то відправити повідомлення «Yes» на клієнт, інакше «No»
27	Відправити з клієнта на сервер рядок тексту, на сервері вилучити всі прописні літери, результат повернути клієнту.
28	Відправити на сервер три числа, знайти їх середнє геометричне та повернути результат клієнту.
29	Відправити на клієнт два повідомлення, якщо вони однакові, то серверу надіслати рядок тексту «Failed».
30	Відправити на сервер масив, знайти добуток його елементів та повернути результати клієнту

Завдання № 3

Розробити клієнтський та серверний додаток на основі протоколу TCP. З'єднання між клієнтом та сервером підтримувати до того часу, поки сервер не отримає від клієнта повідомлення «Exit». Передбачити обробку помилок введення даних користувачем. Функції клієнта та сервера подано нижче.

№ варіанта	Завдання
1	Відправляти повідомлення на сервер, видаляти у ньому усі голосні літери та повертати отримане повідомлення на клієнт.
2	Розробити просту програму-тестування. Сервер має список питань, на які можна дати відповіді «Yes» чи «No». Після підключення він по черзі задає питання та підраховує кількість вірних відповідей. Після закінчення опиту пересилає на клієнт кількість вірних відповідей.
3	Створити програму віддаленого віртуального піаніно. На клієнті пишеться мелодія у вигляді нот-символів, потім вона відправляється на сервер, де мелодія програється. Коли сервер став вільним, він повідомляє клієнт повідомленням «Playing done». Для генерації звуків використати метод `Console.Beep()`.
4	Створити програму-гру «Вгадай вік», на сервері задається число, клієнт відправляє варіант на сервер, сервер обробляє результат та відсилає «Equal», «More», «Less».
5	Організувати віддалений стек за принципом Last Input First Output (LIFO). Відправляти повідомлення з клієнта на сервер у вигляді «push *value*», де *value* – дійсне число. Команда «pop» вилучає з сервера останнє передане число. Команди «add» та «sub» додають та віднімають відповідно два числа, що знаходяться на вершині стеку на сервері і потім сервер повертає результат клієнту.
6	Розробити клієнт, який намагається виконати DDoS-атаку на сервер, посилаючи неперервний потік повідомлень «gettime». У випадку, коли навантаження на сервер не перевищує три повідомлення за секунду, у відповідь на запит «gettime» сервер повертає клієнту поточний локальний час, у зворотньому випадку – сервер не відповідає. Розробити клієнт так, щоб можна було промоделювати різну частоту генерування повідомлень.

№ варіанта	Завдання
7	Створити програму-чат між клієнтом та сервером.
8	Створити клієнт для перегляду файлової структури серверу. На клієнті реалізувати наступні операції: «ls» – показати перелік файлів поточного каталогу; «pwd» – показати поточний каталог; «cd..» – піднятися в дереві каталогів у батьківську директорію.
9	Відправляти з клієнта на сервер команди у форматі «getrnd *n*», де *n* – кількість випадкових чисел, які сервер повертає клієнту у вигляді масиву.
10	Відправляти числа з клієнта на сервер, сервер додає їх до масиву. Якщо надсилається повідомлення «show», то на клієнт посилається масив чисел та виводиться на консоль.
11	На клієнті реалізувати можливість отримання даних від сервера про поточну дату та час. У консолі клієнта запит здійснюється за допомогою повідомлень: «showdate, showtime».
12	Створити аналог команди ping. Луна-сервер повинен повторювати клієнту, відправлені ним повідомлення, розмір повідомлення та час відповіді сервера.
13	Відправляти на сервер повідомлення з 5 чисел та команду, що з ними робити. «SORTUP» – сортувати за зростанням, «SORTDOWN» – сортувати за спаданням. Результат повернути клієнту.
14	Відправляти на сервер повідомлення з двома числами та команду, що з ними робити. «SUM» – знайти суму, «DEVIDE» – поділити, «MUL» – помножити. Результат повернути клієнту.
15	Відправляти масив чисел з клієнта на сервер, сервер сортує їх за зростанням та відправляє результат на клієнт.
16	Відправляти масив чисел з клієнта на сервер, сервер сортує їх за спаданням та відправляє результат на клієнт.

№ варіанта	Завдання
17	Відправляти на сервер повідомлення, сервер випадковим чином переставляє в повідомленні букви та відправляє результат клієнту.
18	Відправляти на сервер два восьми розрядних двійкових числа записаних з використанням символів «0» та «1». Сервер виконує побітові операції AND, OR, XOR між прийнятими числами та відправляє результат клієнту.
19	Відправляти на сервер повідомлення, у відповіді від сервера вказати чи є в повідомленні парні цифри, якщо так, то передати їх загальну кількість, якщо ні, то відповідне повідомлення.
20	Відправляти декілька рядків тексту на сервер, сервер випадковим чином змінює порядок рядків та повертає клієнту.
21	Створити програму-чат між клієнтом та сервером, заборонивши при цьому передавати числа та прописні літери.
22	Відправляти двомірну матрицю на сервер, знаходити суму її елементів та відправляти результат на клієнт. Розмірність матриці та елементи повинні задаватися користувачем.
23	Відправляти повідомлення з клієнта на сервер, сервер обробляє їх, та повертає відповідь у вигляді довжини повідомлення.
24	Відправляти на сервер повідомлення, у відповіді від сервера вказати чи є в ньому цифри, якщо так, то передати їх загальну кількість, якщо ні, то відповідне повідомлення.
25	Створити аналог команди ping. Луна-сервер повинен повторювати клієнту, відправлені ним повідомлення у зворотньому порядку.
26	Відправляти на сервер повідомлення з 8 чисел та команду, що з ними робити. «VAR» – розрахувати дисперсію,

№ варіанта	Завдання
	«MEAN» – розрахувати математичне очікування. Результат повернути клієнту.
27	Створити елемент скриптової мови. Клієнт відправляє команди серверу: «*var* = *value*» – задати змінній *var* значення *value* (наприклад, «a = 5»); «*var*» – вивести на консоль значення змінної *var*. Всі змінні повинні зберігатися на сервері.
28	Відправляти масив чисел з клієнта на сервер, сервер сортує тільки від'ємні значення за зростанням та відправляє результат на клієнт.
29	Розробити просту програму-тестування. Сервер має список питань, на які можна дати відповіді «Yes» чи «No». Після підключення він по черзі задає питання та підраховує кількість помилок. Після закінчення опиту пересилає на клієнт результати тестування.
30	Створити програму-чат між клієнтом та сервером, перевіряючи при підключенні пароль, встановлений на сервері.

Контрольні питання

1. Що таке мережний сокет?
2. Які основні принципи мережної взаємодії додатків із застосуванням портів?
3. Коротко охарактеризуйте стек протоколів TCP/IP.
4. Перерахуйте номери портів таких протоколів та служб: HTTP, DNS, FTP, Telnet, ICQ, Skype.
5. Як співвідноситься модель відкритої взаємодії OSI та стек протоколів TCP/IP?
6. Чим відрізняються протоколи транспортного рівня TCP та UDP?
7. Коротко опишіть схему створення сокетного TCP з'єднання.

8. Які засоби платформи .NET Framework призначені для передачі даних між вузлами комп’ютерної мережі?

9. Запропонуйте підхід до створення багатоклієнтського сервера на основі протокола TCP.

10. Яким чином можна реалізувати луна-сервер та визначити час передачі даних між вузлами?

ЛАБОРАТОРНА РОБОТА № 4

Тема: технологія доступу до баз даних засобами ADO.NET.

Мета: розробити клієнтський додаток типу WinForms із застосуванням технології ADO.NET для оброблення інформації із бази даних під управлінням Microsoft SQL Server, розробити запити на мові SQL.

Теоретичні відомості

ADO.NET (ActiveX Data Objects .NET) – це набір бібліотек, що поставляється з Microsoft .NET Framework і призначений для взаємодії з різними сховищами даних з .NET-додатків. Бібліотеки ADO.NET включають класи для приєднання до джерела даних, виконання запитів і обробки їхніх результатів. Крім того, ADO.NET можна використовувати в якості надійного, ієрархічно організованого, відокремленого кешу даних для автономної роботи з даними. ADO.NET була розроблена компанією Microsoft, для вирішення проблем, які виникали при роботі з ADO та попередніми технологіями, такими як: Data Access Objects (DAO), Remote Data Objects (RDO).

ADO.NET має багато переваг порівняно з іншими технологіями доступу до даних, а саме: підтримка XML, простота модифікації та програмування, висока продуктивність.

ADO також підтримує XML, але не буде так само ефективно обробляти XML-дані, як це робить ADO.NET, оскільки ADO.NET створювався з врахуванням XML.

Протягом терміну служби системи в неї можна вносити незначні зміни, однак спроби провести архітектурні зміни трапляються рідко, через виняткову складність завдання. На жаль, при природному розвитку подій такі зміни іноді виявляються необхідними.

Компоненти даних ADO.NET в Visual Studio інкапсулюють функціональні можливості доступу до даних різними способами, що допомагає розробляти програмні продукти значно швидше і з меншою кількістю помилок.

Для непідключених додатків набори даних ADO.NET дають виграш у продуктивності у порівнянні з непідключеними наборами записів ADO. Передача непідключеного набору записів між рівнями за допомогою COM-упаковки може призвести до великої витрати обчислювальних ресурсів, тому що значення в наборі записів перетворюються до типів даних, відомих COM. У ADO.NET таке перетворення типів даних не потрібно.

Три моделі ADO.NET

Бібліотеки ADO.NET можна застосовувати трьома концептуально різними способами: в підключеному режимі, в автономному режимі та за допомогою технології Entity Framework. При використанні підключеного рівня (connected layer) кодова база ьезпосередньо підключається до відповідного сховища даних та відключається від нього. При такому способі використання ADO.NET взаємодія зі сховищем даних звичайно проводиться за допомогою об'єктів підключення, об'єктів команд та об'єктів читання даних.

Автономний рівень (disconnected layer) дозволяє працювати з набором об'єктів DataTable, які містяться в DataSet), котрий на стороні клієнта надає копію зовнішніх даних. При отриманні DataSet зі допомогою відповідного об'єкта адаптера даних підключення відкривається та закривається автоматично. Такий підхід дозволяє швидко звільняти підключення для інших викликів та підвищує масштабованість системи.

Отримавши об'єкт DataSet, викликаючий код може розглядати та обробляти дані без витрат на мережевий трафік. А якщо потрібно занести зміни до сховища даних, то адаптер даних (рахом з набором операторів SQL) використовується для оновлення даних – при цьому підключення відкривається заново для проведення оновення в базі даних, а потім одразу ж хакривається.

Після випуску .NET 3.5 SP1 в ADO.NET з'явилася підримка нового API, який називається Entity Framework (EF). Технологія EF показує, що багато низькорівневих деталей роботи з базами даних (наприклад,

складні SQL-запити) приховані від програміста та обровлюються за нього при генерації відповідного LINQ-запита (LINQ Entities).

Постачальник даних ADO.NET

Постачальник даних .NET – це набір класів, призначених для взаємодії зі сховищем даних певного типу. .NET Framework включає два постачальника – SQL Client.NET Data Provider і OLE DB.NET Data Provider. Постачальник OLE DB.NET Data Provider дозволяє взаємодіяти з різними сховищами даних за допомогою постачальника OLE DB. Постачальник SQL Client.NET Data Provider розрахований виключно на взаємодію з БД SQL Server. Кожен постачальник даних. NET реалізує однакові базові класи – Connection, Command, DataProvider, Parameter і Transaction, конкретне ім'я яких залежить від постачальника (таблиця 6.1). Так, у постачальника SQL Client.NET Data Provider є об’єкт SqlConnection, а у постачальника OLE DB. NET Data Provider – об’єкт OleDbConnection.

Таблиця 4.1 – Основні об’єкти постачальників даних ADO.NET

№ з/п	Постачальник даних	Простір імен	Компоновочний блок
1	OLE DB	System.Data.OleDb	System.Data.dll
2	Microsoft SQL Server	System.Data.SqlClient	System.Data.dll
3	Microsoft SQL Server Mobile	System.Data.SqlServerCe	System.Data.SqlServerCe.dll
4	ODBC	System.Data.Odbc	System.Data.dll
5	Oracle	System.Data.OracleClient	System.Data.OracleClient.dll

У таблиці 4.2 наведено деякі загальні основні об’єкти, їх базові класи (визначені у просторі імен System.Data.Common) і основні інтерфейси (визначені у просторі імен System.Data), які вони реалізують.

Таблиця 4.2 – Основні об’єкти постачальників даних ADO.NET

№ з/п	Тип об’єкта	Базовий клас	Відповідні інтерфейси	Призначення
1	Connection	DbConnection	IDbConnection	Дозволяє підключатися до сховища даних та відключатися від нього. Крім того об’єкти підключення забезпечують доступ до відповідних об’єктв транзакцій
2	Command	DbCommand	IDbCommand	Представляє SQL-запит чи збережену процедуру. Крім того об’єкти команд надають доступ до об’єктів читання даних конкретного постачальника даних
3	DataReader	DbDataReader	IDataReader, IDataRecord	Надає доступ до даних тільки для читання у прямому напрямку за допомогою курсора на строні сервера
4	DataAdapter	DbDataAdapter	IDataAdapter, IDbDataAdapter	Пересилає набори даних із сховища даних до виклакаючого процесу і в зворотньому напрямі. Адаптери дани містять підключення та набір із чотирьох втутрішніх об’єктів команд для вибірки, вставки, зміни та видалення інформації з бази даних
5	Parameter	DbParameter	IDataParameter, IDbDataParameter	Представляє іменований параметр у

№ з/п	Тип об’єкта	Базовий клас	Відповідні інтерфейси	Призначення
				параметризованому запиті
6	Transaction	DbTransaction	IDbTransaction	Інкапсулює транзакцію у базі даних

Простори імен ADO.NET

Крім просторів імен, які визначають типи конкретних постачальників даних, у бібілотеках базових класів .NET містяться додаткові простори імен, орієнтовані на ADO.NET. Деякі з них перелічен в таблиці 4.3.

Таблиця 4.3 – Додаткові простори імен для роботи з ADO.NET

№ з/п	Простір імен	Призначення
1	Microsoft.SqlServer.Server	Містить типи для роботи служби інтеграції CLR та SQL Server
2	System.Data	Визанчає основні типи ADO.NET, які використовуються всіми постачальниками даних, у тому числі загальні інтерфейси та різні типи, що забезпечуюють автономний рівень (наприклад, DataSet, DataTable та ін.)
3	System.Data.Common	Містить типи для загального використання всіми постачальниками ADO.NET, у тому числі і загальні абстрактні базові класи
4	System.Data.Sql	Містять типи, які дозволяють визначити екземпляри Microsoft SQL Server, які встановлені у локальній мережі
5	System.Data.Sql.Types	Містить типи даних, які використовує Microsoft SQL Server

Аудиторне завдання

Створити власну базу даних типу Microsoft SQL Server. Додати до бази даних одну таблицю «Ноутбуки». Заповнити таблицю за допомогою інструментів Visual Studio, додавши до неї декілька записів. Програмно вивести зміст таблиці на головне вікно додатку типу Windows Forms за допомогою компоненту DataGridView. Додатково створити конструктор запитів для одного з полів таблиці, для вибірки записів за цим критерієм (наприклад, відібрати всі ноутбуки з обсягом оперативної пам’яті не менше 4 ГБ.)

Розв’язання

Спочатку потрібно створити проект типу Windows Form, який дозволяє проектувати графічний інтерфейс користувача з використанням елементів керування Windows (рисунок 4.1).

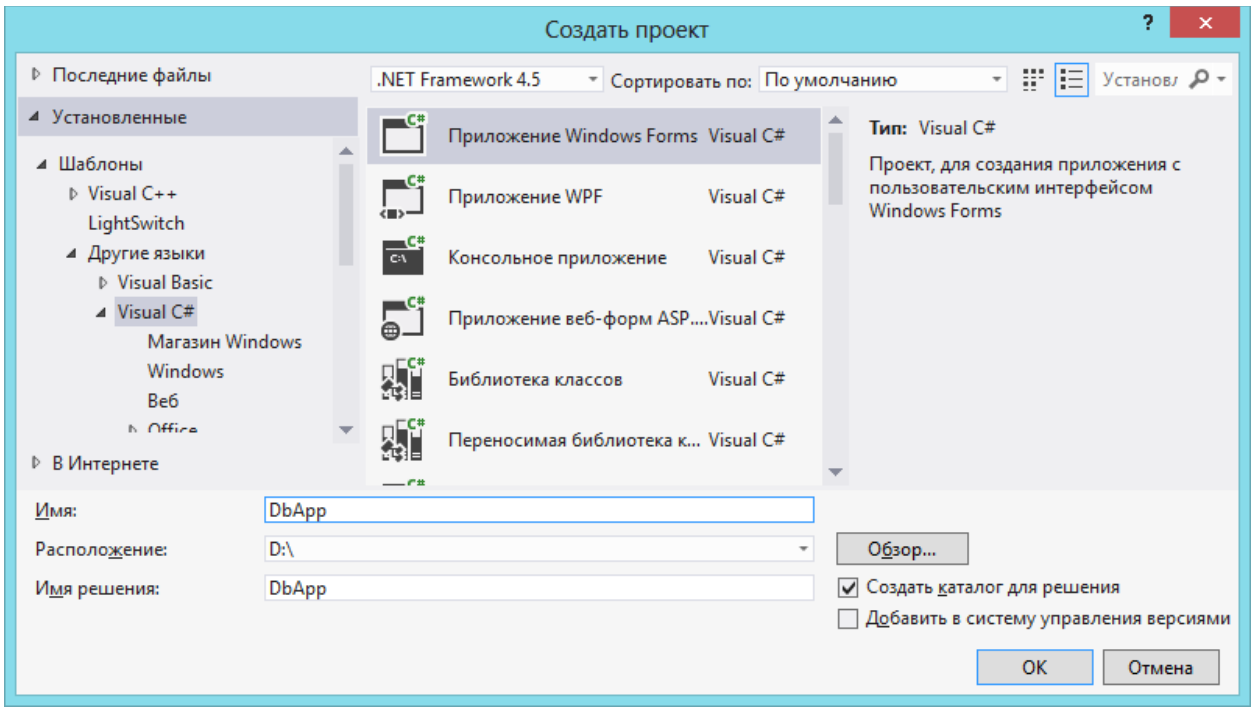


Рисунок 4.1 – Діалог створення нового проекту WinForms

До проекту необхідно додати новий елемент (локальну базу даних).

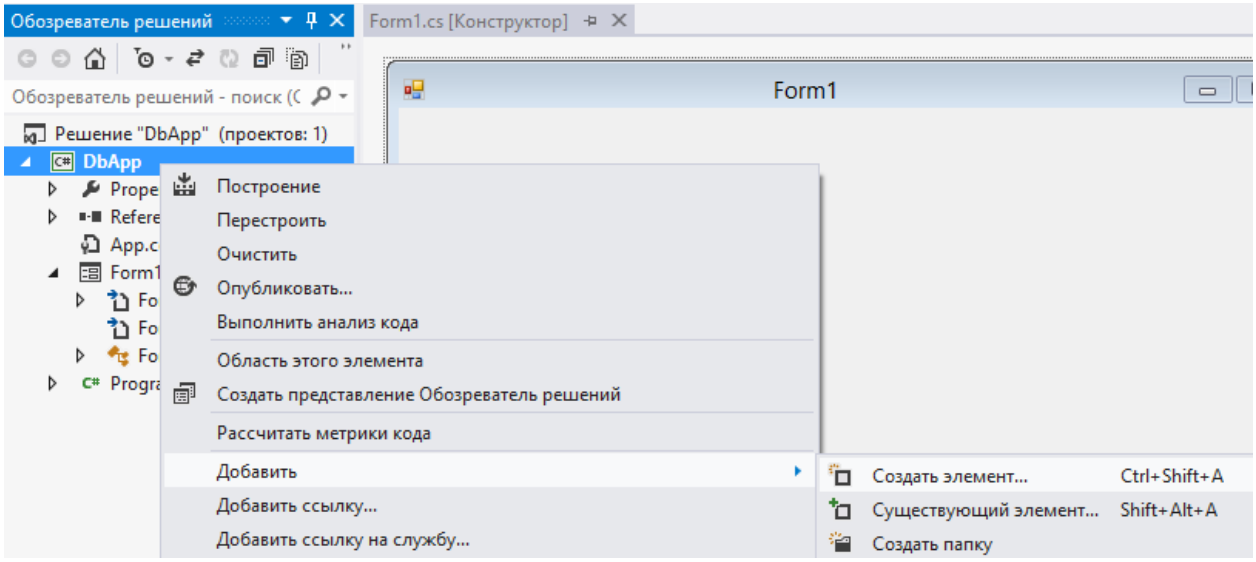


Рисунок 4.2 – Додавання до проекту нового елемента

У якості сховища даних потрібно обрати локальну базу даних на основі служби типу Service-based Database (рисунок 4.3).

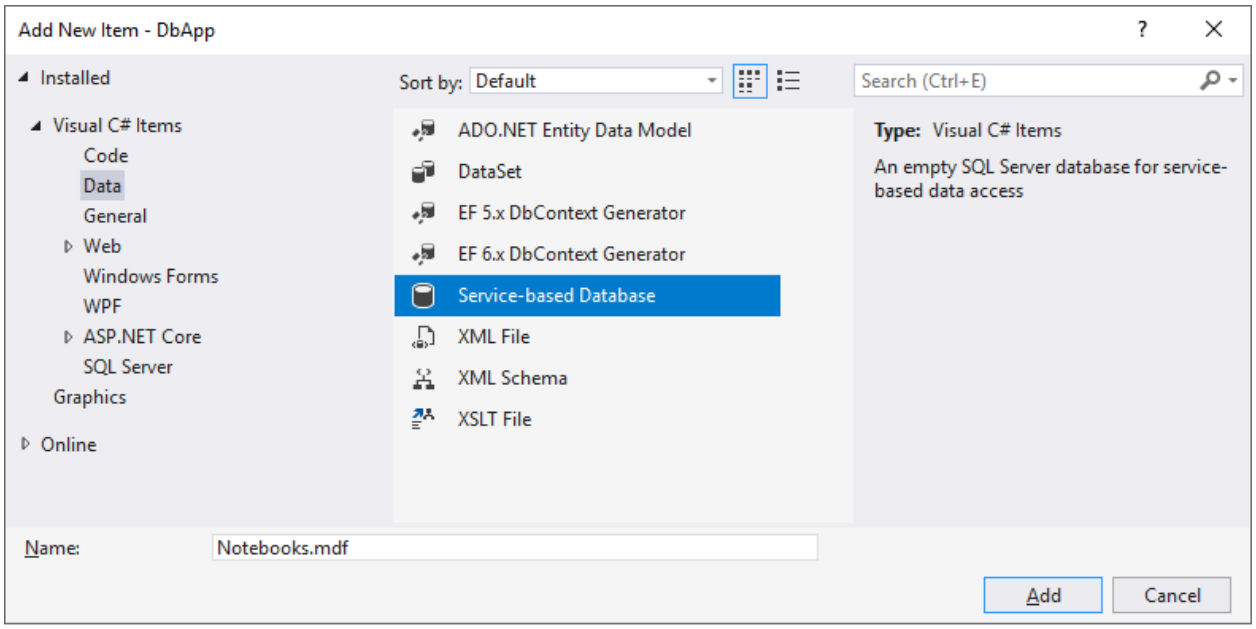


Рисунок 4.2 – Створення локальної бази даних Notebooks

Для роботи зі створеною базою даних необхідно відкрити переглядач серверів за допомогою відповідного пункту меню «Вид» (рисунок 4.3).

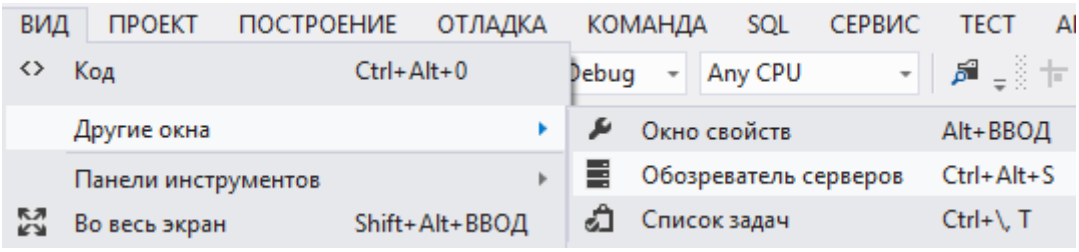


Рисунок 4.3 – Відображення вікна «Обозреватель серверов»

Після підключення до бази даних потрібно додати таблицю за допомогою контекстного меню вкладки «Таблиці» (рисунок 4.4).

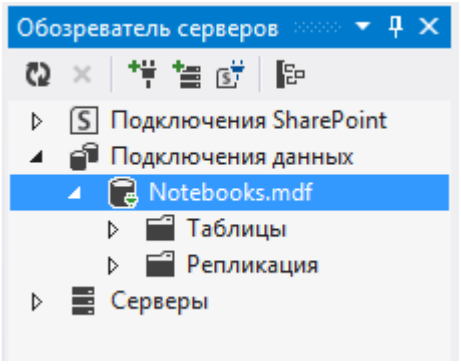


Рисунок 4.4 – Перегляд структури бази даних Notebooks

Для опису ноутбуків застосовано первинний ключ ID. Таблиця бази даних містить поля щодо моделі ноутбука, встановленого процесора, обсягу оперативної пам’яті та жорсткого диску, а також ціну на дану модель (рисунок 4.5).

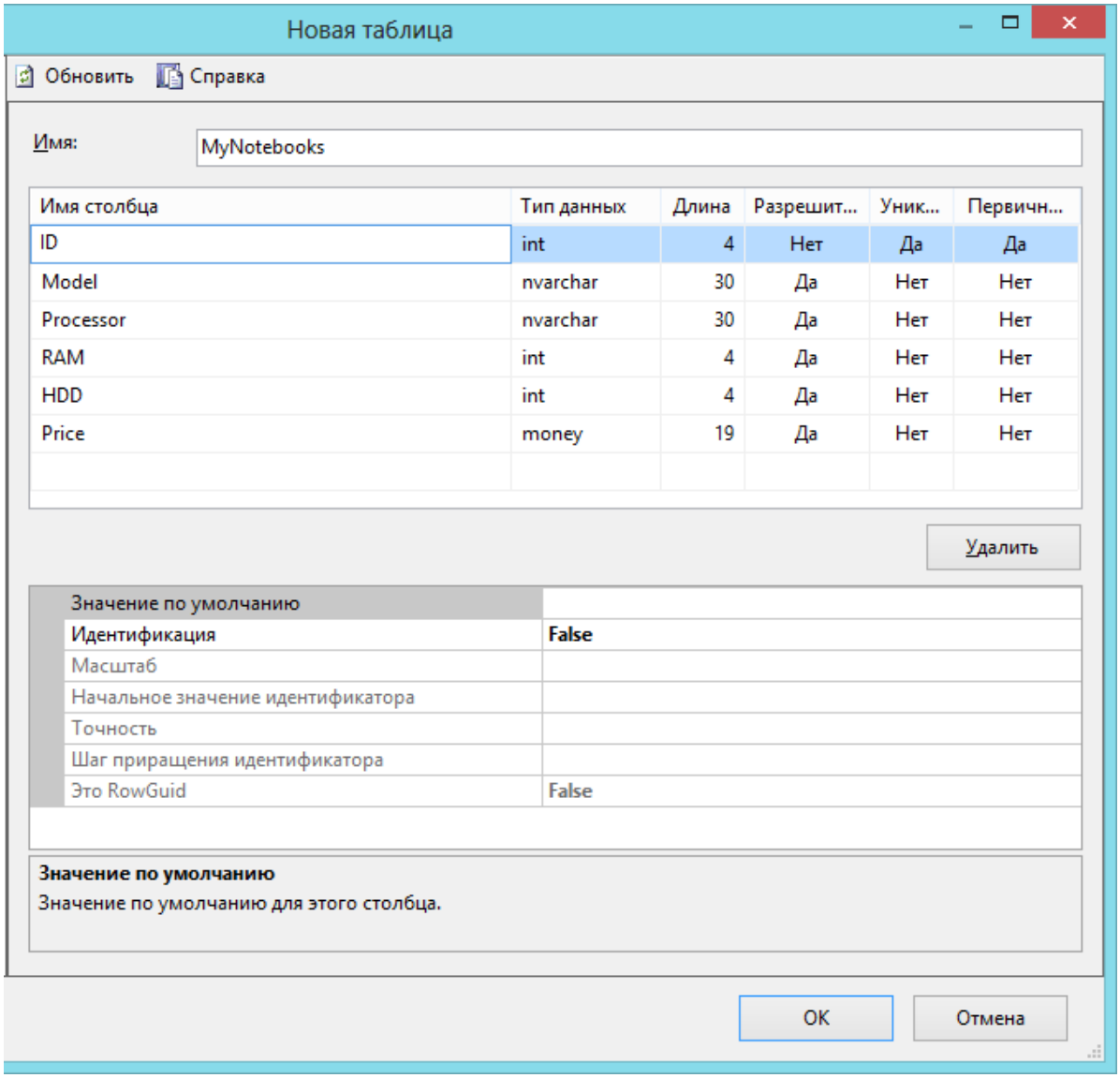


Рисунок 4.5 – Створення структури таблиці, що містить інформацію про ноутбуки

Після створення таблиці її потрібно наповнити даними (рисунки 4.6, 4.7).

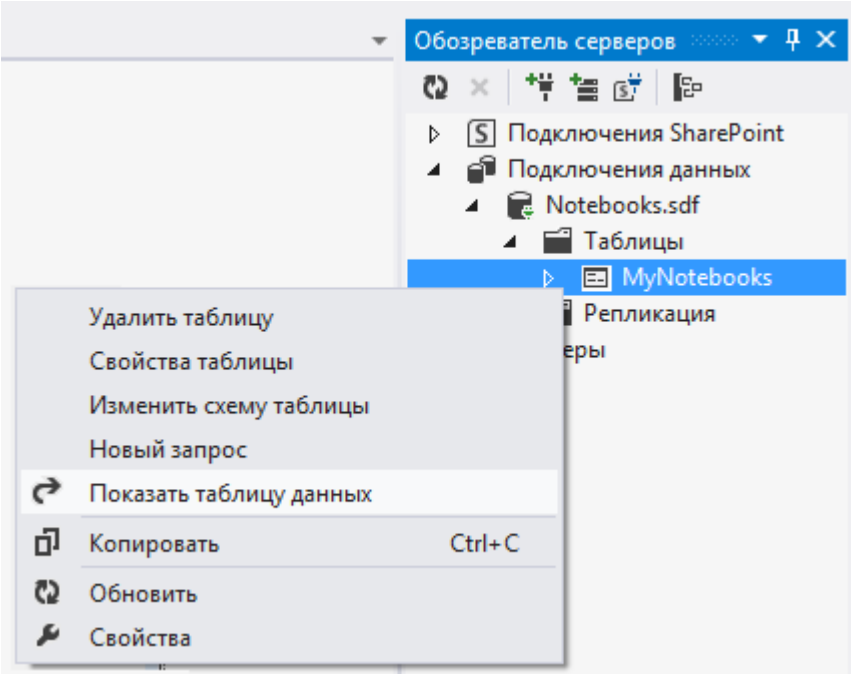


Рисунок 4.6 – Відображення таблиці для введення даних

dbo.MyNotebooks [Data]

Max Rows: 1000

Id	Model	Processor	RAM	HDD	Price
1	HP 655 (B6N22EA)	AMD Dual-Core E2-1800	4	500	3170.0000
2	Lenovo IdeaPad B570e	Intel Pentium B950	4	500	3300.0000
3	Lenovo IdeaPad G580GH	Intel Core i3-2328M	4	1000	4200.0000
4	Asus X301A (X301A-RX157D)	Intel Core i3-3110M	4	500	4350.0000
5	Samsung 350V5	Intel Core i5-3210M	6	750	7300.0000
NULL	NULL	NULL	NULL	NULL	NULL

Рисунок 4.7 – Додавання даних до таблиці

Для створення графічного інтерфейсу користувача застосовані об’єкти: DatagridView, Label, MaskedTextBox, Button (рисунок 4.8).

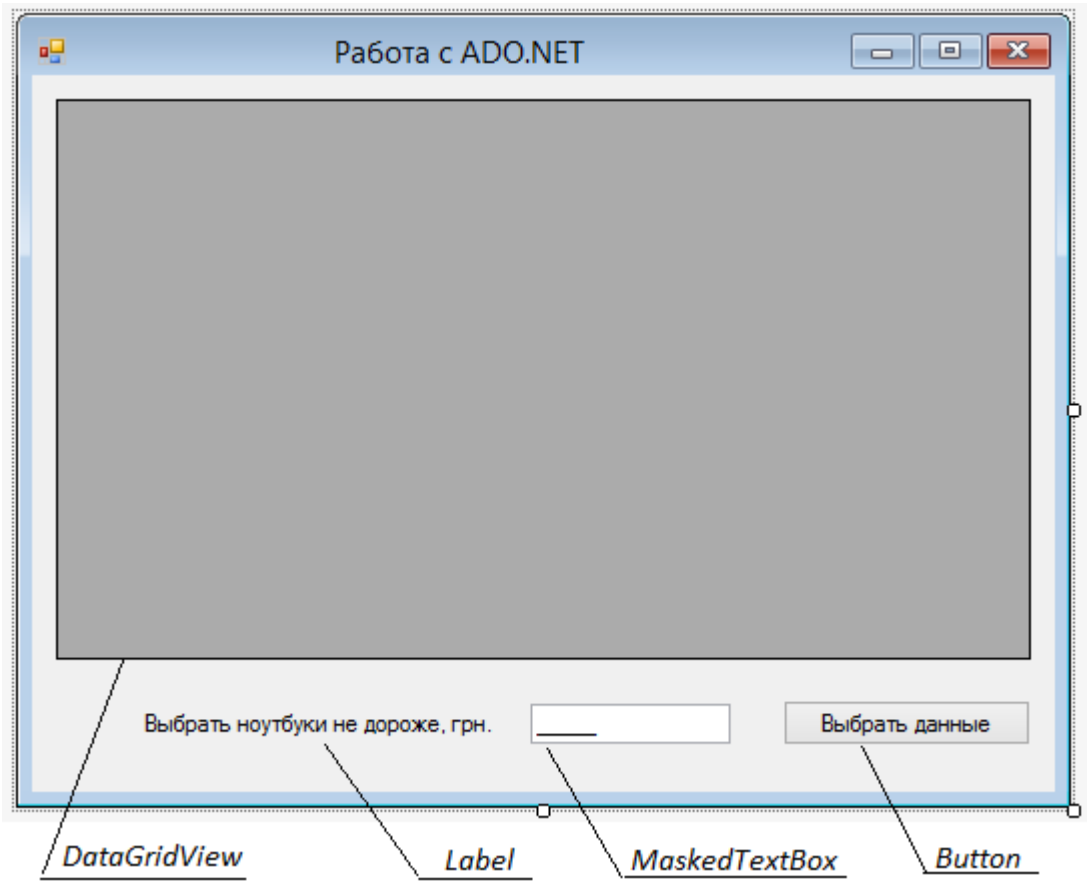


Рисунок 4.8 – Додавання елементів у конструкторі форм

Після коригування імен візуальних компонентів за допомогою вікна «Властивості» та налаштування компоненту MaskedTextBox, програмний код може мати такий вигляд:

```
using System;
using System.Data;
using System.Data.SqlClient;
using System.Windows.Forms;

namespace DbApp
{
  public partial class FormMain : Form
  {
    public FormMain()
    {
      InitializeComponent();
    }
```

```
    private void buttonSelect_Click(object sender, EventArgs e)
    {
      string connectionStr = @"Data
Source=(LocalDB)\MSSQLLocalDB;
AttachDbFilename=d:\DbApp\DbApp\Notebooks.mdf; Integrated
Security=True";
      SqlConnection con = new SqlConnection(connectionStr);
      string sqlStr = "Select * from MyNotebooks where Price <=
" + maskedTextBox.Text; // SQL-запит
      SqlDataAdapter dataAdapter = new SqlDataAdapter(sqlStr,
con);
      DataSet queryResult = new DataSet();
      dataAdapter.Fill(queryResult);
      dataGridView.DataSource = queryResult.Tables[0];
      con.Close();
    }
  }
}
```

Слід звернути увагу, у змінній connectionStr міститься шлях до файлу бази даних. У цьому прикладі вважається, що проект знаходить у корені диска D:\.

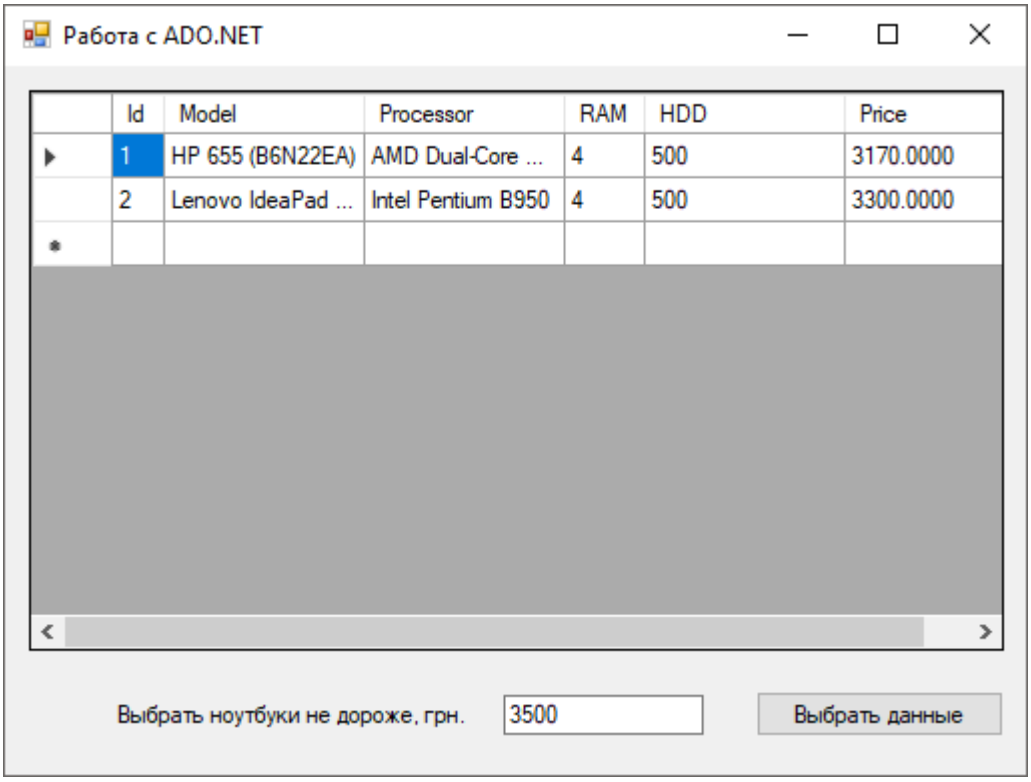


Рисунок 4.9 – Тестування додатку

Після компіляції та запуску програми можна впевнитися, що додаток працює з локальною базою даних, виконуючи відповідний SQL-запит для вибору ноутбуків, які не дорожчі за вказану ціну (рисунок 4.9).

Зауваження: з метою спрощення пояснення роботи з ADO.NET шлях до бази даних задавався жорстким посиланням в коді. Звичайно у випадку переміщення бази даних, програма перестане працювати. Найбільш доцільно у цьому випадку або використати поточний локальний каталог, з якого запускається додаток, наприклад, за допомогою `System.IO.Directory.GetCurrentDirectory()`, або застосувати діалогове вікно відкриття файлу бази даних `OpenFileDialog`.

Завдання до лабораторної роботи

Відповідно до варіанта та обраного рівня складності виконати наступні завдання.

Рівень завдань		
Початковий	Базовий	Високий
Повторити та протестувати аудиторні завдання	Завдання № 1	Завдання № 2
	Завдання № 2	Завдання № 3

Завдання № 1

Створити власну базу даних типу Microsoft SQL Server. Додати до бази даних одну таблицю, яка містила б не менше чотирьох інформаційних полів. Заповнити таблицю за допомогою інструментів Visual Studio, додавши не менше п’яти записів. Програмно вивести зміст таблиці на головне вікно додатку типу Windows Forms за допомогою компоненту DataGridView. Таблицю індивідуальних варіантів подано нижче.

Завдання № 2

Виконати завдання 1. Додатково створити два запити типу Select, результати яких повинні виводитись до DataGridView. Таблицю індивідуальних варіантів подано нижче.

Завдання № 3

Виконати завдання 1. Додатково створити конструктор запитів для одного з полів таблиці, для вибірки записів за цим критерієм (наприклад, відібрати всіх студентів у яких поле «вік» > 21, або «бал» < 4 тощо). Таблицю індивідуальних варіантів подано нижче.

№ варіанта	Завдання
1	Таблиця «Група КСМ»
2	Таблиця «Результати екзамену»
3	Таблиця «Склад кафедри»
4	Таблиця «Факультети університету»
5	Таблиця «Предмети курсу»
6	Таблиця «Міста України»
7	Таблиця «Розклад занять»
8	Таблиця «Математична довідка»
9	Таблиця «Звідка погоди»
10	Таблиця «Моє місто – основні дані»
11	Таблиця «Комп’ютери»
12	Таблиця «Кадри»
13	Таблиця «Товари»
14	Таблиця «Лікарські препарати»
15	Таблиця «Автомобілі»
16	Таблиця «Пацієнти»
17	Таблиця «Абітурієнти»
18	Таблиця «Закони»
19	Таблиця «Фільми»

№ варіанта	Завдання
20	Таблиця «Музика»
21	Таблиця «Мікропроцесори»
22	Таблиця «Монітори»
23	Таблиця «Планшети»
24	Таблиця «Мобільні телефони»
25	Таблиця «Мережеві кабелі»
26	Таблиця «Журнали»
27	Таблиця «Книги»
28	Таблиця «Розклад»
29	Таблиця «Операційні системи»
30	Таблиця «Мови програмування»

Контрольні питання

1. Коротко охарактеризуйте технологію ADO.NET.
2. З якими базами даних може працювати ADO.NET?
3. Які основні моделі роботи зі сховищами даних передбачено у ADO.NET?
4. Для чого використовуються провайдери даних?
5. Призначення та функціональність класів SqlConnection, SqlDataAdapter, DataSet.
6. Які найбільш уживані візуальні елементи керування WinForms потрібні для побудови додатку для роботи з базами даних?
7. Синтаксис SQL-оператора Select.
8. Як вибрати дані з декількох таблиць бази даних?
9. Як видалити рядки з таблиці, які відповідають певній умові?
10. Які нормальні форми баз даних Вам відомі?

ЛАБОРАТОРНА РОБОТА № 5

Тема: програмування додатків на базі технології Windows Presentation Foundation (WPF).

Мета: розробити програми типу WPF на мові C#, ознайомитися з мовою розмітки XAML.

Теоретичні відомості

Windows Presentation Foundation (WPF) – це система для побудови клієнтських додатків Windows з візуально привабливими можливостями. На наступному рисунку показано приклад одного із таких додатків Contoso Healthcare Sample Application.

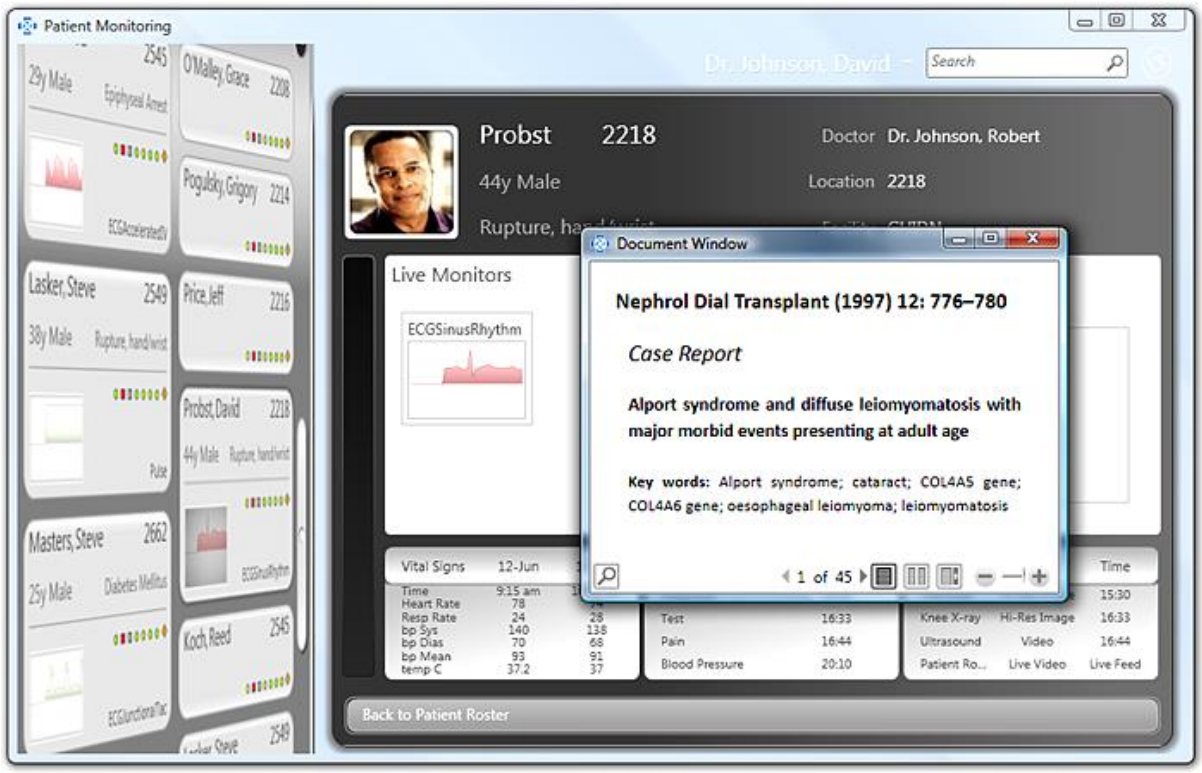


Рисунок 5.1 – Приклад додатка WPF

В основі WPF лежить векторна система малювання, яка не залежить від розширення і створена з урахуванням можливостей сучасного графічного обладнання.

Детальніше з технологією WPF можна ознайомитися у книзі: Троелсен Э. Язык программирования C# 6.0 и платформа .NET 4.6 / Э. Троелсен, Ф. Джепикс; пер. с англ. – [7-е изд.]. – М.: Вильямс, 2016. – С. 997-1119.

Аудиторне завдання

Створити текстовий редактор з використанням WPF. Забезпечити можливість завантаження та зберігання інформації у файл. Створити головне меню, додати панель інструментів, організувати можливість форматування тексту (жирний, похилий та підкреслений шрифт).

Розв’язання

Після створення додатку із шаблону WPF на головній формі потрібно видалити менеджер розміщення `Grid` та додати `DockPanel`. `StatusBar` необхідно стикувати з нижньою стороною `DockPanel`, а `Menu`, `ToolBarTray` – до верхньої сторони.

До проекту необхідно додати окрему папку з малюнками (*.ico), які будуть розміщуватися на пунктах головного меню та на кнопках панелі інструментів. При цьому у властивостях малюнків необхідно задати режим вбудовування у *.exe файл у вигляді ресурсу.

Розмітка головної форми з всіма компонентами представлена нижче:

```
<Window

xmlns="http://schemas.microsoft.com/winfx/2006/xaml/present
ation"
  xmlns:x="http://schemas.microsoft.com/winfx/2006/xaml"
  xmlns:Properties="clr-namespace:Lab6.Properties"
  x:Class="Lab6.MainWindow"
  Title="KSN Text Editor v1.0" Height="337.649"
Width="522.728"
  Icon="images/Icon.ico"
WindowStartupLocation="CenterScreen">
  <Window.CommandBindings>
```

```
    <CommandBinding Command="ApplicationCommands.New"

Executed="NewExecuted"></CommandBinding>
    <CommandBinding Command="ApplicationCommands.Open"

Executed="OpenExecuted"></CommandBinding>

    <CommandBinding Command="ApplicationCommands.Save"

Executed="SaveExecuted"></CommandBinding>
  </Window.CommandBindings>
  <DockPanel LastChildFill="False">
    <StatusBar DockPanel.Dock="Bottom" >
      <TextBlock x:Name="rowIndex" Text="Row: 0"/>
      <Separator/>
      <TextBlock x:Name="colIndex" Text="Column: 0"/>
    </StatusBar>
    <Menu VerticalAlignment="Top" DockPanel.Dock="Top">
      <MenuItem Header="File">
         <MenuItem Header="New"
Command="ApplicationCommands.New">
            <MenuItem.Icon>
               <Image Source="images/new.ico"/>
              </MenuItem.Icon>
           </MenuItem>
       <MenuItem Header="Open"
Command="ApplicationCommands.Open">
           <MenuItem.Icon>
              <Image Source="Images/open.ico"></Image>
           </MenuItem.Icon>
       </MenuItem>
    <MenuItem Header="Save"
Command="ApplicationCommands.Save">
      <MenuItem.Icon>
        <Image Source="Images/save.ico" />
      </MenuItem.Icon>
      </MenuItem>
        <Separator/>
      <MenuItem Header="Exit"/>
```

```
      </MenuItem>
        <MenuItem Header="Edit">
      <MenuItem Header="Cut"
Command="ApplicationCommands.Cut">
        <MenuItem.Icon>
          <Image Source="images/cut.ico"></Image>
        </MenuItem.Icon>
      </MenuItem>
      <MenuItem Header="Copy"
Command="ApplicationCommands.Copy">
        <MenuItem.Icon>
          <Image Source="images/copy.ico"></Image>
        </MenuItem.Icon>
      </MenuItem>
      <MenuItem Header="Paste"
Command="ApplicationCommands.Paste">
        <MenuItem.Icon>
          <Image Source="images/paste.ico"></Image>
        </MenuItem.Icon>
       </MenuItem>
     </MenuItem>
     <MenuItem Header="Help" >
        <MenuItem Header="About" Click="AboutClick"/>
        </MenuItem>
      </Menu>

  <ToolBarTray VerticalAlignment="Top"
DockPanel.Dock="Top">
    <ToolBar VerticalAlignment="Top" DockPanel.Dock="Top">
      <Button Width="25" Margin="4,2" Content="B"
FontWeight="Bold"
        Command="EditingCommands.ToggleBold"/>
      <Button Width="25" Margin="4,2" Content="I"
FontStyle="Italic"
        Command="EditingCommands.ToggleItalic"/>
      <Button Width="25" Margin="4,2"
        Command="EditingCommands.ToggleUnderline">
      <TextBlock TextDecorations="Underline" Text="U"/>
      </Button>
```

```
    </ToolBar>
  </ToolBarTray>
  <RichTextBox x:Name="textBox" VerticalAlignment="Stretch"
    SelectionChanged="TextBox_SelectionChanged"
FontSize="18"
    FontFamily="Times New Roman" >
    <FlowDocument></FlowDocument>
  </RichTextBox>
</DockPanel>
</Window>
```

Обробки подій представлено нижче:

```
using System;
using System.Collections.Generic;
using System.Linq;
using System.Text;
using System.Threading.Tasks;
using System.Windows;
using System.Windows.Controls;
using System.Windows.Data;
using System.Windows.Documents;
using System.Windows.Input;
using System.Windows.Media;
using System.Windows.Media.Imaging;
using System.Windows.Navigation;
using System.Windows.Shapes;
using System.IO;
using Microsoft.Win32;

namespace Lab6
{
/// <summary>
/// Interaction logic for MainWindow.xaml
/// </summary>
public partial class MainWindow : Window
{
public MainWindow()
{
    InitializeComponent();
```

```
}
private void NewExecuted(object sender,
ExecutedRoutedEventArgs e)
{
    textBox.Document.Blocks.Clear();
}

private void OpenExecuted(object sender,
ExecutedRoutedEventArgs e)
{
    OpenFileDialog dlg = new OpenFileDialog();
    dlg.DefaultExt = "Text documents (.rtf)|*.rtf";
    Nullable<bool> res = dlg.ShowDialog();
    if (res == true)
    {
        TextRange textRange =
          new TextRange(textBox.Document.ContentStart,
                        textBox.Document.ContentEnd);
        using (FileStream fileStream = new
FileStream(dlg.FileName,

FileMode.Open))
        {
            textRange.Load(fileStream, DataFormats.Rtf);
        }
    }
}

private void SaveExecuted(object sender,
ExecutedRoutedEventArgs e)
{
    SaveFileDialog dlg = new SaveFileDialog();
    dlg.AddExtension = true;
    dlg.DefaultExt = ".rtf";
    dlg.Filter = "Text documents (.rtf)|*.rtf";
    Nullable<bool> res = dlg.ShowDialog();
    if (res == true)
    {
        TextRange textRange =
```

```
            new TextRange(textBox.Document.ContentStart,
                          textBox.Document.ContentEnd);
        using (FileStream fileStream = new
FileStream(dlg.FileName,

FileMode.OpenOrCreate))
        {
            textRange.Save(fileStream, DataFormats.Rtf);
        }
    }
}

private void TextBox_SelectionChanged(object sender,
RoutedEventArgs e)
{
    TextPointer tp1 =
textBox.Selection.Start.GetLineStartPosition(0);
    TextPointer tp2 = textBox.Selection.Start;
    int column = tp1.GetOffsetToPosition(tp2);
    int someBigNumber = int.MaxValue;
    int lineMoved;

    textBox.Selection.Start.GetLineStartPosition(-
someBigNumber,
                                                 out
lineMoved);
    int currentLineNumber = 1 - lineMoved;
    rowIndex.Text = "Row: " + currentLineNumber.ToString();
    colIndex.Text = "Column: " + column.ToString();
}

private void AboutClick(object sender, RoutedEventArgs e)
{
    MessageBox.Show("KSN Text Editor v 1.0", "About");
}}}
```

Для детального ознайомлення з проектом програми слід скористатися архівом `Lab6.zip`.

На рисунку 5.1 представлено скриншот роботи програми.

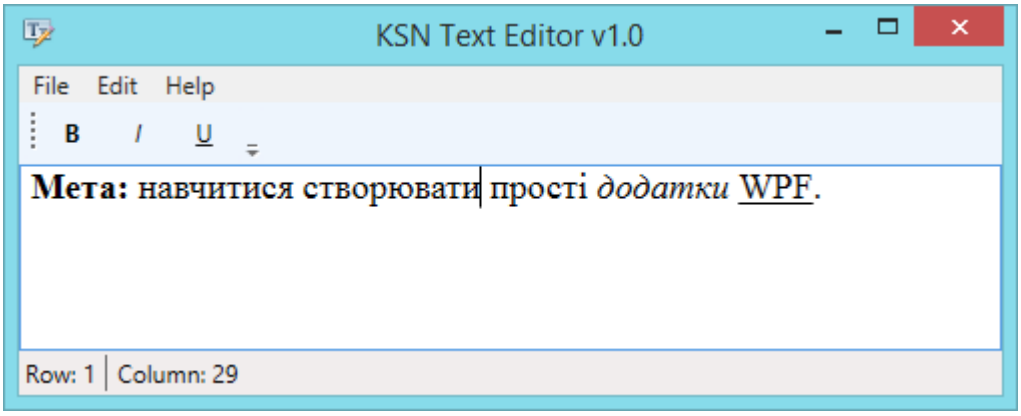


Рисунок 5.1 – Приклад роботи програми

Завдання до лабораторної роботи

Відповідно до варіанта та обраного рівня складності виконати наступні завдання.

Рівень завдань		
Початковий	Базовий	Високий
Відкомпілювати та протестувати аудиторне завдання	Завдання № 1	Завдання № 2

Завдання № 1

Розробити програму типу Windows Presentation Foundation, додати необхідні візуальні компоненти для введення початкових значень задачі та виведення результатів.

№ варіанта	Завдання
1	Користувач заповнює одновимірний масив із 10 цілих чисел. Програма визначає суму додатних чисел і середнє арифметичне всіх чисел масиву.
2	Користувач заповнює одновимірний масив із 20 дійсних чисел. Програма знаходить суму від'ємних чисел, відсікає дробову частку чисел і визначає, скільки серед них парних.

№ варіанта	Завдання
3	Користувач заповнює масив із 20 цілих чисел випадковими числами. Програма знаходить максимальне і мінімальне число.
4	Знайти три найбільших елемента одновимірного масиву із 15-ти елементів.
5	Користувач заповнює одновимірний масив із 20-ти елементів випадковими числами. Програма визначає кількість таких елементів в масиве, для яких попередній елемент менше, а наступний більше даного елемента.
6	Дано одновимірний масив із 16-ти символів. Вивести елементи масиву на екран так, що всі цифри були виведені зеленим кольором, а останні символи – червоним.
7	Дано одновимірний масив із дійсних чисел. Ввести число c і знайти значення виразу $c + \frac{c}{a_1} + \frac{c}{a_2} + \cdots + \frac{c}{a_n}$, де a – елементи масиву.
8	Одновимірний масив із 10-ти цілих чисел заповнити з клавіатури, визначити суму тих чисел, котрі більші 5 і є парними.
9	Заповнити одновимірний масив 15-ма символами. У масиві із символів всі цифри замінити на «*». Вивести елементи зміненого масиву.
10	Скласти програму підрахунку загальної кількості цифр і знаків «+», «–», «*» у рядку s, введеного з клавіатури.
11	Скласти програму підрахунку кількості цифр у заданому рядку тексту
12	Написати програму, яка у рядку довільної довжини знаходить символ «а» і видаляє за ним 5 символів.
13	Користувач заповнює масив із 12-ти цілих чисел випадковими числами. Програма знаходить максимальний елемент масиву і виводить його на екран зеленим кольором.

№ варіанта	Завдання
14	Відредагувати задане речення, видаляючи з нього всі слова, які цілком складені із не більше, ніж двох букв. Приклад: AKKA KNOBIKAISE → KNOBIKAISE.
15	Перевірити, чи вірно, що у заданому реченні будь-яке несиметричне слово має парну довжину.
16	Відредагувати задане речення, видаляючи із нього слова, які зустрічаються в реченні задане число раз.
17	Надрукувати задане речення таким чином, щоб кожне його слово повністю знаходилось в одному і тому ж рядку роздруківки (тобто уникнути переносів слів та їх розривання).
18	В заданому реченні вказати слово, в якому частка голосних (A, E, I, O, U) максимальна.
19	Перелічити всі слова заданого речення, які складаються із тих же букв, що і перше слово речення.
20	Переставить і роздрукувати слова заданого речення згідно із збільшенням частки приголосних (B, C, D, F, G, H, K, L, M, N, P, Q, R, S, T, V, W, X, Z) у цих словах.
21	Для кожного символу заданого тексту вказати, скільки разів воно зустрічається у тексті. Повідомлення про однин символ повинно друкуватися не більше одного разу.
22	Замінити закінчення ING кожного слова, що зустрічається у заданому реченні, на ED.
23	Програма вводить два рядки довільної довжини і формує масив із номерів позицій, в яких символи рядків не співпадають.
24	Програма вводить два рядки. Якщо довжина другого рядка більше 50 символів, то програма виводить на екран символи першого рядка з непарними номерами. Якщо довжина першого рядка менше 15 символів, то програма виводить символи другого рядка з парними номерами.

№ варіанта	Завдання
25	Підрахувати кількість цифр у записі числа *n* в системі числення з основою 2.
26	Програма вводить два рядки. Вивести на екран, скільки букв «П» у довгому рядку, і скільки букв «R» у короткому рядку. Результат вивести на екран.
27	Програма вводить два рядки. Вивести на екран конкатенацію рядків, а також вивести кожний рядок на екран, якщо в ньому міститься буквосполучення «окр».
28	Програма вводить 10 рядків, виводить на екран сполучення 1-го і 10-го рядка, 2-го і 9-го, і т. д.
29	Програма вводить два рядки довільної довжини і формує масив із номерів позицій, в яких символи рядків не співпадають.
30	Програма вводить два рядки з клавіатури, виводить на екран конкатенацію 2-х рядків і довжину об’єднаного рядка

Завдання № 2

Розробити текстовий редактор на базі технології WPF. Забезпечити наступні функції:

- створення нового файлу, збереженні та завантаження файлу;
- виведення поточної позиції курсора у документі;
- вікно «Про програму» з ім’ям розробника програми.

Додатково доповнити функціями відповідно до варіанта.

№ варіанта	Завдання
1	Забезпечити можливість вибору розміру та кольору шрифту
2	Додати пункт «Зберегти як», при цьому програма повинна видавати повідомлення при створенні нового документа, якщо не збережено поточний.

№ варіанта	Завдання
3	Додати можливість вставки у текст поточної дати при натисненні комбінації Ctrl+Alt+D
4	Забезпечити можливість роботи програми з файлами форматів *.*txt* та *.*rtf* (завантаження, збереження, повідомлення про втрату форматування)
5	Додати у рядок `StatusBar` випадаючий список, яким можна змінювати розмір шрифту виділеного фрагмента тексту
6	Організувати пошук у тексті заданого слова, створити відповідний пункт меню
7	Додати можливість задавати формат виділеного фрагменту тексту (надкреслений, підкреслений, закреслений)
8	Додати кнопку на панелі інструментів, яка дозволяє виконати інвертування кожного слова у тексті, наприклад: *this is example → siht si elpmaxe*
9	Створити вікно «Про програму», на якому необхідно відобразити таку інформацію: логотип додатку, ім’я та прізвище автора, рік створення додатку, загальний обсяг оперативної пам’яті, яку споживає додаток
10	Додати пункт меню, який дозволяє замінити всі числа в тексті на ім’я поточного облікового запису користувача Windows.

Контрольні запитання

1. Коротко охарактеризуйте технологію WPF?
2. Які візуальні компоненти для введення даних та компоновки їх розміщення передбачені у додатках WPF?
3. Яка структура розмітки XAML?
4. Опишіть кроки компіляції проекту WPF?
5. Які класи передбачено для створення 3D-додатків з анімацією?

ЛАБОРАТОРНА РОБОТА № 6

Тема: основи анімації візуальних компонентів та створення мультимедійних додатків на базі WPF.

Мета: розробити програми типу WPF на мові C# для перегляду відео та аудіо файлів, виконати анімацію властивостей візуальних компонентів.

Теоретичні відомості

Перед початком виконання лабораторної роботи необхідно ознайомитися з теоретичним матеріалом з книги: Троелсен Э. Язык программирования C# 6.0 и платформа .NET 4.6 / Э. Троелсен, Ф. Джепикс; пер. с англ. – [7-е изд.]. – М.: Вильямс, 2016. – С. 1120-1245.

Аудиторні завдання

Завдання № 1

Створити анімаційну панель інструментів для запуску стандартних додатків Windows: калькулятора, командного рядку та графічного редактора Paint.

Розв’язання

Макет головної форми досить простий. Він містить лише одну панель інструментів, яка розміщена по центру форми. Анімація виконується за допомогою тригерів мови XAML.

Рисунок 6.1 – Ефект анімації кнопок при наведені курсора миші

Код розмітки XAML наведено нижче (з метою компактності лише для однієї кнопки):

```
<Window x:Class="Animation.MainWindow"

xmlns="http://schemas.microsoft.com/winfx/2006/xaml/presentatio
n"
        xmlns:x="http://schemas.microsoft.com/winfx/2006/xaml"
        Title="Animation Demo" Height="154" Width="365"
        WindowStartupLocation="CenterScreen">
  <DockPanel Background="#FFEEF5FD">
    <ToolBarTray VerticalAlignment="Center"
HorizontalAlignment="Center">
      <ToolBar Background="#FFEEF5FD">
        <Button Name="btnCalc" Height="48"
Click="btnCalcClick">
          <Image  Source="images/calc1.ico"></Image>
          <Button.Triggers>
            <EventTrigger RoutedEvent="Button.MouseEnter">
              <BeginStoryboard>
                <Storyboard>
                  <DoubleAnimation
                    Storyboard.TargetName="btnCalc"
                    Storyboard.TargetProperty="Height"
                    To="80" Duration="0:0:0.25"
                    />
                </Storyboard>
              </BeginStoryboard>
            </EventTrigger>
            <EventTrigger RoutedEvent="Button.MouseLeave">
              <BeginStoryboard>
```

```
                <Storyboard>
                  <DoubleAnimation
                    Storyboard.TargetName="btnCalc"
                    Storyboard.TargetProperty="Height"
                    To="48" Duration="0:0:0.1"
                    />
                </Storyboard>
              </BeginStoryboard>
            </EventTrigger>
          </Button.Triggers>
        </Button>
      </ToolBar>
    </ToolBarTray>
  </DockPanel>
</Window>
```

Програмна частина містить декілька обробників вигляду:

```
private void btnCalcClick(object sender, RoutedEventArgs e)
{
    System.Diagnostics.Process.Start("calc");
}
```

Завдання № 2

Створити програвач аудіо-відео файлів з використанням WPF. Додати елементи завантаження файлу, запуску, паузи та зупинки. Програвач повинен відкривати файли найбільш поширених форматів: **.mp3, *.wav, *.mp4, *avi*.

Розв’язання

Для роботи з аудіо-відео контентом у WPF слугує компонент `MediaElement`. Інтерфейс програми розділено на 3 вертикальні частини. Верхня частина містить панель інструментів, середня слугує для відображення відео, а нижня – містить декілька компонентів. У нижній частині вікна програми показується час, який пройшов від початку

програвання файлу, елемент `Slider` дозволяє змінити поточну позицію (кадр), а елемент `ProgressBar` слугує для відображення гучності, яку можна змінити за допомогою колеса миші (рисунок 6.2).

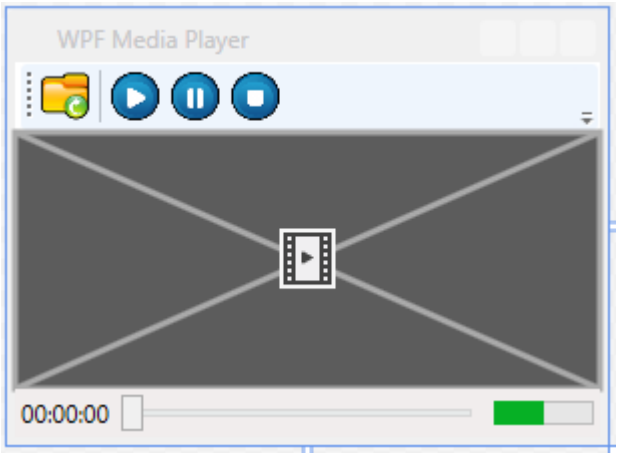


Рисунок 6.2 – Розміщення компонентів на формі

Зауважимо, що у даному додатку використовуються WPF-команди замість простих обробників натиснення кнопок. Це дозволяє у досить простий спосіб повторно використовувати функціональні можливості, наприклад, у випадку додавання контекстного меню. Команди також дозволяють спростити перевірку доступності програми у залежності від стану плеєра.

Розмітка головної форми з всіма компонентами представлена нижче:

```
<Window x:Class="Lab7.MainWindow"
xmlns="http://schemas.microsoft.com/winfx/2006/xaml/presentation"
  xmlns:x="http://schemas.microsoft.com/winfx/2006/xaml"
  Title="WPF Media Player" Height="300" Width="300"
  MinWidth="300" SizeToContent="WidthAndHeight"
  WindowStartupLocation="CenterScreen">
  <Window.CommandBindings>
    <CommandBinding Command="ApplicationCommands.Open"
      CanExecute="Open_CanExecute" Executed="Open_Executed" />
    <CommandBinding Command="MediaCommands.Play"
      CanExecute="Play_CanExecute" Executed="Play_Executed" />
    <CommandBinding Command="MediaCommands.Pause"
      CanExecute="Pause_CanExecute" Executed="Pause_Executed"
/>
    <CommandBinding Command="MediaCommands.Stop"
```

```
      CanExecute="Stop_CanExecute" Executed="Stop_Executed" />
  </Window.CommandBindings>
  <Grid MouseWheel="Grid_MouseWheel">
    <Grid.RowDefinitions>
      <RowDefinition Height="Auto" />
      <RowDefinition Height="*" />
      <RowDefinition Height="Auto" />
    </Grid.RowDefinitions>
    <ToolBar>
      <Button Command="ApplicationCommands.Open">
        <Image Width="24" Height="24" Source="Images/Open.ico"
/>
      </Button>
      <Separator />
      <Button Command="MediaCommands.Play">
        <Image Width="24" Height="24" Source="Images/Play.ico"
/>
      </Button>
      <Button Command="MediaCommands.Pause">
        <Image Width="24" Height="24" Source="Images/Pause.ico"
/>
      </Button>
      <Button Command="MediaCommands.Stop">
        <Image Width="24" Height="24" Source="Images/Stop.ico"
/>
      </Button>
    </ToolBar>
    <MediaElement Name="mePlayer" Grid.Row="1"
LoadedBehavior="Manual"
      Stretch="None" />
    <StatusBar Grid.Row="2">
      <StatusBar.ItemsPanel>
        <ItemsPanelTemplate>
          <Grid>
            <Grid.ColumnDefinitions>
              <ColumnDefinition Width="Auto" />
              <ColumnDefinition Width="*" />
              <ColumnDefinition Width="Auto" />
            </Grid.ColumnDefinitions>
```

```
          </Grid>
        </ItemsPanelTemplate>
      </StatusBar.ItemsPanel>
      <StatusBarItem>
        <TextBlock
Name="lblProgressStatus">00:00:00</TextBlock>
      </StatusBarItem>
      <StatusBarItem Grid.Column="1"
HorizontalContentAlignment="Stretch">
        <Slider Name="sliProgress"
          Thumb.DragStarted="sliProgress_DragStarted"
          Thumb.DragCompleted="sliProgress_DragCompleted"
          ValueChanged="sliProgress_ValueChanged" />
        </StatusBarItem>
      <StatusBarItem Grid.Column="2">
        <ProgressBar Name="pbVolume" Width="50" Height="12"
Maximum="1"
          Value="{Binding ElementName=mePlayer, Path=Volume}"
/>
      </StatusBarItem>
    </StatusBar>
  </Grid>
</Window>
```

Обробники подій представлено нижче:

```
using System;
using System.Windows;
using System.Windows.Input;
using Microsoft.Win32;
using System.Windows.Threading;
using System.Windows.Controls.Primitives;

namespace Lab7
{
  public partial class MainWindow : Window
  {
    private bool mediaPlayerIsPlaying = false;
    private bool userIsDraggingSlider = false;
    public MainWindow()
```

```
    {
      InitializeComponent();
      DispatcherTimer timer = new DispatcherTimer();
      timer.Interval = TimeSpan.FromSeconds(1);
      timer.Tick += timer_Tick;
      timer.Start();
    }
    private void timer_Tick(object sender, EventArgs e)
    {
      if ((mePlayer.Source != null) &&
        (mePlayer.NaturalDuration.HasTimeSpan) &&
(!userIsDraggingSlider))
      {
        sliProgress.Minimum = 0;
        sliProgress.Maximum =
          mePlayer.NaturalDuration.TimeSpan.TotalSeconds;
        sliProgress.Value = mePlayer.Position.TotalSeconds;
      }
    }

    private void Open_CanExecute(object sender,
                                 CanExecuteRoutedEventArgs e)
    {
      e.CanExecute = true;
    }

    private void Open_Executed(object sender,
                               ExecutedRoutedEventArgs e)
    {
      OpenFileDialog openFileDialog = new OpenFileDialog();
      openFileDialog.Filter = "Media files
        (*.mp3;*.mpg;*.mpeg;*.mp4)|
        *.mp3;*.mpg;*.mpeg;*.mp4|All files (*.*)|*.*";
      if (openFileDialog.ShowDialog() == true)
      {
        mePlayer.Source = new Uri(openFileDialog.FileName);
        mePlayer.Play();
      }
    }
```

```
    private void Play_CanExecute(object sender,
                                 CanExecuteRoutedEventArgs e)
    {
      e.CanExecute = (mePlayer != null) && (mePlayer.Source !=
null);
    }
    private void Play_Executed(object sender,
                               ExecutedRoutedEventArgs e)
    {
      mePlayer.Play();
      mediaPlayerIsPlaying = true;
    }
   private void Pause_CanExecute(object sender,
                                 CanExecuteRoutedEventArgs e)
    {
      e.CanExecute = mediaPlayerIsPlaying;
    }
    private void Pause_Executed(object sender,
                                ExecutedRoutedEventArgs e)
    {
      mePlayer.Pause();
    }
    private void Stop_CanExecute(object sender,
                                 CanExecuteRoutedEventArgs e)
    {
      e.CanExecute = mediaPlayerIsPlaying;
    }
    private void Stop_Executed(object sender,
                               ExecutedRoutedEventArgs e)
    {
      mePlayer.Stop();
      mediaPlayerIsPlaying = false;
    }
    private void sliProgress_DragStarted(object sender,
                                         DragStartedEventArgs e)
    {
      userIsDraggingSlider = true;
    }
```

```
    private void sliProgress_DragCompleted(object sender,
                                    DragCompletedEventArgs e)
    {
      userIsDraggingSlider = false;
      mePlayer.Position =
        TimeSpan.FromSeconds(sliProgress.Value);
    }
    private void sliProgress_ValueChanged(object sender,
RoutedPropertyChangedEventArgs<double> e)
    {
      lblProgressStatus.Text =
        TimeSpan.FromSeconds(sliProgress.Value)
            .ToString(@"hh\:mm\:ss");
    }
    private void Grid_MouseWheel(object sender,
MouseWheelEventArgs e)
    {
      mePlayer.Volume += (e.Delta > 0) ? 0.1 : -0.1;
    }
  }
}
```

Для детального ознайомлення з проектом програми слід скористатися архівом `Lab7.zip`. На рисунку 6.3 представлено скріншот роботи програми.

Рисунок 6.3 – Приклад роботи відео програвача

Завдання до лабораторної роботи

Відповідно до варіанта та обраного рівня складності виконати наступні завдання.

Рівень завдань		
Початковий	Базовий	Високий
Відкомпілювати та протестувати аудиторні завдання	Завдання № 1	Завдання № 2

Завдання № 1

Розробити програму типу Windows Presentation Foundation, додати необхідні компоненти для введення початкових значень задачі та виведення результатів. Створити для двох будь-яких візуальних компонентів довільний анімаційний ефект.

№ варіанта	Завдання
1	Програма вводить два рядки. Якщо довжина другого рядка більше 50 символів, то програма виводить на екран символи першого рядка з непарними номерами. Якщо довжина першого рядка менше 15 символів, то програма виводить символи другого рядка з парними номерами.
2	Підрахувати кількість цифр у записі числа n в системі числення з основою 2.
3	Програма вводить два рядки. Вивести на екран, скільки букв «П» у довгому рядку, і скільки букв «R» у короткому рядку. Результат вивести на екран.
4	Програма вводить два рядки. Вивести на екран конкатенацію рядків, а також вивести кожний рядок на екран, якщо в ньому міститься буквосполучення «окр».
5	Програма вводить 10 рядків, виводить на екран сполучення 1-го і 10-го рядка, 2-го і 9-го, і т. д.

№ варіанта	Завдання
6	Програма вводить два рядки довільної довжини і формує масив із номерів позицій, в яких символи рядків не співпадають.
7	Програма вводить два рядки з клавіатури, виводить на екран конкатенацію 2-х рядків і довжину об'єднаного рядка
8	Користувач заповнює одновимірний масив із 10 цілих чисел. Програма визначає суму додатних чисел і середнє арифметичне всіх чисел масиву.
9	Користувач заповнює одновимірний масив із 20 дійсних чисел. Програма знаходить суму від'ємних чисел, відсікає дробову частку чисел і визначає, скільки серед них парних.
10	Користувач заповнює масив із 20 цілих чисел випадковими числами. Програма знаходить максимальне і мінімальне число.
11	Знайти три найбільших елемента одновимірного масиву із 15-ти елементів.
12	Користувач заповнює одновимірний масив із 20-ти елементів випадковими числами. Програма визначає кількість таких елементів в масиве, для яких попередній елемент менше, а наступний більше даного елемента.
13	Дано одновимірний масив із 16-ти символів. Вивести елементи масиву на екран так, що всі цифри були виведені зеленим кольором, а останні символи – червоним.
14	Дано одновимірний масив із дійсних чисел. Ввести число c і знайти значення виразу $c + \frac{c}{a_1} + \frac{c}{a_2} + \cdots + \frac{c}{a_n}$, де a – елементи масиву.
15	Одновимірний масив із 10-ти цілих чисел заповнити з клавіатури, визначити суму тих чисел, котрі більші 5 і є парними.

№ варіанта	Завдання
16	Заповнити одновимірний масив 15-ма символами. У масиві із символів всі цифри замінити на «*». Вивести елементи зміненого масиву.
17	Скласти програму підрахунку загальної кількості цифр і знаків «+», «–», «*» у рядку s, введеного з клавіатури.
18	Скласти програму підрахунку кількості цифр у заданому рядку тексту
19	Написати програму, яка у рядку довільної довжини знаходить символ «a» і видаляє за ним 5 символів.
20	Користувач заповнює масив із 12-ти цілих чисел випадковими числами. Програма знаходить максимальний елемент масиву і виводить його на екран зеленим кольором.
21	Відредагувати задане речення, видаляючи з нього всі слова, які цілком складені із не більше, ніж двох букв. Приклад: AKKA KNOBIKAISE → KNOBIKAISE.
22	Перевірити, чи вірно, що у заданому реченні будь-яке несиметричне слово має парну довжину.
23	Відредагувати задане речення, видаляючи із нього слова, які зустрічаються в реченні задане число раз.
24	Надрукувати задане речення таким чином, щоб кожне його слово повністю знаходилось в одному і тому ж рядку роздруківки (тобто уникнути переносів слів та їх розривання).
25	В заданому реченні вказати слово, в якому частка голосних (A, E, I, O, U) максимальна.
26	Перелічити всі слова заданого речення, які складаються із тих же букв, що і перше слово речення.
27	Переставить і роздрукувати слова заданого речення згідно із збільшенням частки приголосних (B, C, D, F, G, H, K, L, M, N, P, Q, R, S, T, V, W, X, Z) у цих словах.

№ варіанта	Завдання
28	Для кожного символу заданого тексту вказати, скільки разів воно зустрічається у тексті. Повідомлення про однин символ повинно друкуватися не більше одного разу.
29	Замінити закінчення ING кожного слова, що зустрічається у заданому реченні, на ED.
30	Програма вводить два рядки довільної довжини і формує масив із номерів позицій, в яких символи рядків не співпадають.

Завдання № 2

На базі аудиторного завдання удосконалити аудіо-відео програвач. Розробити анімацію елементів графічного інтерфейсу відповідно до варіанта.

№ варіанта	Завдання
1	Виконати анімацію всіх кнопок панелі інструментів. При наведенні курсора на кнопку вона декілька разів змінює свій розмір, а після прибирання курсора стає поступово напівпрозорою
2	Після завантаження відео воно повинно з’являтися поступово змінюючи прозорість та розмір
3	Додати наверх зображення відео програвача напівпрозоре відображення поточної гучності, яке з’являється при зміні гучності, а потім поступово зникає
4	При натисненні на кнопку «Пауза» зображення повинно поступово ставати напівпрозорим
5	Забезпечити можливість згортання відео програвача у трей, при цьому зупиняючи програвання файлу. Згортання вікна повинно відбуватися анімовано шляхом поступового зменшення його розміру та прозорості

№ варіанта	Завдання
6	Додати контекстне меню, за допомогою якого можна керувати програванням відео. У випадку зупинки програвання зображення повинно поступово згортатися на формі у нижній правий куток вікна
7	Зробити можливість вибору декількох відео-файлів. При цьому перемикання між різними відео повинно проводитися шляхом поступової зміни прозорості між ними
8	Створити анімаційний ефект: після розгортання вікна на весь екран зображення повинно поступово заповнити всю необхідну область
9	Зробити можливість хования панелі інструментів шляхом вибору відповідного пункту контекстного меню. Приховування панелі інструментів повинно бути анімаційним
10	При подвійному клацанні мишкою на зображенні воно повинно поступово зникати, змінюючи прозорість. При повторному клацанні зображення повинно з’являтися шляхом розширення із середини

Контрольні питання

1. Які засоби анімації передбачено технологією WPF?
2. Які властивості класів можна анімувати у додатках WPF?
3. Перелічити елементи мови розмітки XAML, які використовуються для створення анімації?
4. Які класи бібліотеки .NET Framework дозволяють працювати з аудіо-відео контентом?
5. Що таке стилі у додатках WPF?

ЛАБОРАТОРНА РОБОТА № 7

Тема: програмування мовою C# з використанням технології Language Integrated Query (LINQ).

Мета: розробити додатки типу Windows Forms та виконати оброблення даних за допомогою технології LINQ.

Теоретичні відомості

Перед початком виконання лабораторної роботи необхідно ознайомитися з теоретичним матеріалом із книги: Троелсен Э. Язык программирования C# 6.0 и платформа .NET 4.6 / Э. Троелсен, Ф. Джепикс; пер. с англ. – [7-е изд.]. – М.: Вильямс, 2016. – С. 431-460.

Аудиторне завдання

Дано масив із 10 цілих чисел. Застосувавши синтаксис LINQ інвертувати масив у зворотному порядку; обчислити середнє, максимальне та мінімальне значення; сформувати масив з непарних елементів; створити список із елементів початкового масиву, які більші ніж 10.

Розв’язання

Можливий варіант вирішення задачі представлено нижче:

```
using System;
using System.Collections.Generic;
using System.Linq;
namespace Lab8
{
  class Program
  {
    static void Print(int[] m)
    {
```

```
  foreach (int i in m)
  {
    Console.Write("{0}; ", i);
  }
  Console.WriteLine();
}

static void Main(string[] args)
{
  int[] array = { 1, 3, 6, 7, -10, 15, 20, 2, 9, 23 };

  Console.Write("Array = ");
  Print(array);

  Console.Write("Reverse array = ");
  Print(array.Reverse().ToArray());

  Console.Write("Avg = ");
  Console.WriteLine(array.Average());

  Console.Write("Max = ");
  Console.WriteLine(array.Max());

  Console.Write("Min = ");
  Console.WriteLine(array.Min());

  int[] arrayOdd = (from val in array where val % 2 != 0
                    select val).ToArray();
  Console.Write("Odd array = ");
  Print(arrayOdd);
  Console.Write("List above 10 = ");
  List<int> list = (from val in array where val > 10
                    select val).ToList();
  foreach (int i in list)
  {
    Console.Write("[{0}] ", i);
  }
  Console.ReadKey();
}
```

```
    }
}
```

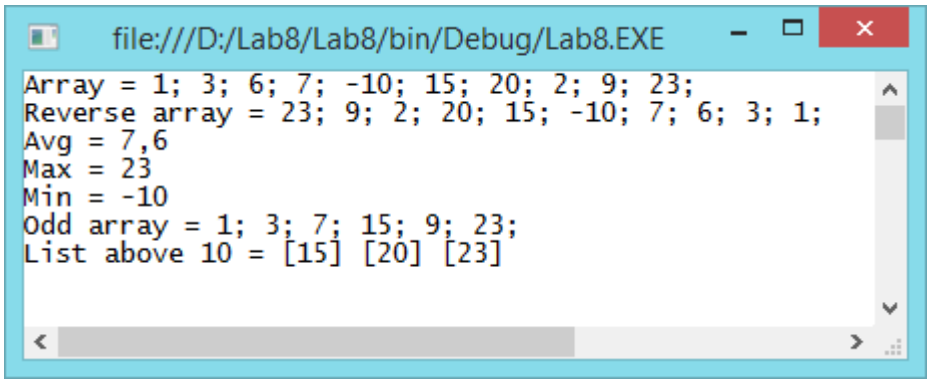


Рисунок 7.1 – Результат роботи програми

Завдання до лабораторної роботи

Відповідно до варіанта та обраного рівня складності виконати наступні завдання.

Рівень завдань		
Початковий	Базовий	Високий
Відкомпілювати та протестувати аудиторне завдання	Завдання № 1	Завдання № 2

Завдання № 1

Розробити програму типу Windows Forms або Windows Presentation Foundation, додати необхідні компоненти для введення початкових значень задачі та виведення результатів. Оброблення даних виконати за допомогою LINQ.

№ варіанта	Завдання
1	Користувач вводить назви міст. Вивести їх відсортувавши за алфавітом; вивести тільки ті назви, які починаються з голосної літери; вивести назви, які містять більше 10 букв

№ варіанта	Завдання
2	Користувач вводить вісім дійсних чисел. Вивести їх за спаданням; вивести тільки числа, які менші 40; знайти мінімальний елемент початкового масиву
3	Користувач вводить речення. Вивести тільки ті речення, які містять парну кількість слів; вивести речення, які не містять ком; вивести речення, які починаються з приголосної літери
4	Програма генерую 25 випадкових цілих чисел. Необхідно вивести числа, які є паліндромами; вивести числа, які не містять у своєму складі цифр 0 та 9
5	Програма генерує 20 кольорів із деякої множини можливих значень. Вивести назви кольорів, назви яких відрізняються не більше, ніж на 3 символи
6	Дано список назв зірок. Вивести тільки ті назви, які мають парну кількість голосних літер; вивести назви зірок, які довші 8 символів
7	Користувач вводить пари ціли чисел (координати точок у двовимірному просторі). Вивести тільки ті точки, які знаходяться на відстані більше 20 від початку координат
8	Користувач вводить довільні слова (не менше 10 слів). Підрахувати кількість однакових слів; визначити кількість, слів які починаються з літер «А», «Б», «П», «Р»
9	Користувач вводить назви країн. Вивести їх відсортувавши за алфавітом; вивести тільки ті назви, які починаються з приголосної літери; вивести назви, які містять менше 10 букв
10	Користувач вводить 12 дійсних чисел. Вивести їх за зростанням; вивести тільки числа, які більші 19; знайти мінімальний елемент початкового масиву
11	Користувач вводить речення. Вивести тільки ті речення, які містять непарну кількість слів; вивести речення, які містять

№ варіанта	Завдання
	не менше двох ком; вивести речення, які починаються з голосної літери
12	Програма генерую 18 випадкових цілих чисел. Необхідно вивести числа, які не є паліндромами; вивести числа, які не містять у своєму складі цифр 2 та 4
13	Програма генерує 12 кольорів із деякої множини можливих значень. Вивести назви кольорів, назви яких відрізняються не більше, ніж на 4 символи
14	Дано список назв річок. Вивести тільки ті назви, які мають парну кількість приголосних літер; вивести назви річок, які довші 10 символів
15	Користувач вводить пари ціли чисел (координати точок у двовимірному просторі). Вивести тільки ті точки, які знаходяться на відстані менше 40 від початку координат
16	Користувач вводить довільні слова (не менше 8 слів). Підрахувати кількість однакових слів; визначити кількість, слів які починаються з літер «В», «Н», «У», «О»
17	Дано три списки, які містять прізвища студентів. Обчислити кількість студентів, які зустрічаються тільки у одному списку, двох списках і всіх трьох списках
18	Користувач пише речення. Програма визначає кількість слів, які мають довжину більше 5 символів. Вивести всі слова, відсортувавши їх за алфавітом; вивести слова, відсортувавши їх за довжиною
19	Користувач вводить назви країн. Вивести їх відсортувавши за алфавітом; вивести тільки ті назви, які починаються з приголосної літери; вивести назви, які містять менше 8 букв
20	Користувач вводить 16 дійсних чисел. Вивести їх за зростанням; вивести тільки числа, які більші 25; знайти середнє арифметичне значення

Завдання № 2

Розробити програму типу Windows Forms, додати необхідні компоненти для введення початкових значень задачі та виведення результатів. Оброблення даних виконати із застосуванням синтаксису LINQ.

№ варіанта	Завдання
1	Дано файл, у якому міститься інформація про результати сесії студентів: прізвище та ім’я студента, оцінки з математики, фізики та програмування, за контрактом чи на бюджетній основі навчається студент. Вивести всіх бюджетників, які повинні отримувати стипендію (середній бал не менше 4). Передбачити можливість виведення студентів відсортувавши їх за середнім балом, прізвищем
2	Дано три файли, в кожному з яких містяться назви міст світу. Вивести всі міста, які є в кожному із файлів. Вивести назви всіх міст з трьох файлів без повторювань, відсортувавши їх за алфавітом. Вивести ті міста, які є в першому файлі, але немає у двох останніх
3	Дано файл, у якому міститься інформація про автомобілі на стоянці: номер автомобіля, його колір, марка, ПІБ власника, час прибуття на автостоянку. За кожну годину перебування на автостоянці нараховуються плата, яку вводить користувач у вікні програми. Вивести тих власників, заборгованість яких перевищує 100 грн. при цьому відсортувати за спаданням боргу. Вивести автомобілі згрупувавши їх за маркою
4	Користувач вводить у вікні програми декілька слів через крапку з комою. Сформувати файл з речень, які містять всі можливі комбінації введених слів. Вивести дані речення у файл спочатку за алфавітом, потім за зростанням довжини речення

№ варіанта	Завдання
5	Користувач вводить назви річок, їх довжину, площу водостоку. Вивести у перший файл річки за зростанням їх довжини. Вивести у другий файл тільки 3 річки з найбільшим водостоком.
6	Дано файл, який містить інформацію про книги у бібліотеці: назву книги, рік видання, дату видачі читачеві, ПІБ читача. Вивести всіх читачів-боржників, відсортувавши за датою видачі. Вивести всі книги, які є у бібліотеці за винятком книг, які видано читачам. Вивести 4 найбільш нові книги, які видано за останній тиждень
7	Дано файл, який містить інформацію про ноутбуки у магазині: модель ноутбука, обсяг оперативної пам’яті, обсяг жорсткого диску, вартість. Вивести ноутбуки, які містять не менше 4 Гб оперативної пам’яті та не дорожчі 5000 грн. Вивести перелік ноутбуків, відсортувавши їх за ціною, та за обсягом оперативної пам’яті
8	Користувач вводить інформацію про аудіо-диски: назва виконавця, кількість треків, рік випуску, вартість. Вивести аудіо-диски, які вийшли за останній рік та не дорожчі 40 грн. Відсортувати всі диски за кількістю треків
9	Дано 2 файли, які містять назви операційних систем та дати їх релізу. Необхідно визначити кількість операційних систем, які містяться у тільки у другому файлі. Вивести за алфавітом всі операційні системи без повторювань. Вивести у файл назви всіх операційних систем, які вийшли за останні 6 місяців
10	Користувач вводить інформацію про відеокарти. Вивести всі відеокарти, які надійшли в продаж за останній рік відсортувавши їх за датою виходу. Вивести всі відеокарти, які дорожчі за 1500 грн. та мають більше 1 Гб пам’яті

Контрольні питання

1. Коротко охарактеризуйте технологію LINQ?
2. Що таке методи розширення?
3. Наведіть приклади анонімних типів.
4. Що таке лямда-вираз і як він застосовується в C#?
5. Перерахуйте методи, які додаються до класу Array завдяки LINQ?

СПИСОК РЕКОМЕНДОВАНОЇ ЛІТЕРАТУРИ

Основна література

1. Троелсен Э., Джепикс Ф. Язык программирования C# 7, платформы .NET и .NET Core : 8-е изд., пер. с англ. СПб. : Диалектика, 2018. 1328 с.
2. Скит Джон. C# для профессионалов. Тонкости программирования. М. : Вильямс, 2019. 608 с.
3. Тузовский А. Ф. Объектно-ориентированное программирование. М. : Юрайт, 2018. 206 с.
4. Албахари Джозеф, Албахари Бен. C# 7.0. Справочник. Полное описание языка 7-е изд. М. : Вильямс, 2018. 1220 с.
5. Рихтер Д. CLR via C#. Программирование на платформе Microsoft .NET Framework 4.5 на языке C# : 4-е изд. СПб. : Питер, 2018. 896 с.
6. Васильев А. Программирование на C# для начинающих. Основные сведения. М. : Эксмо, 2018. 592 с.
7. Тепляков С. Паттерны проектирования на платформе .NET. СПб. : Питер, 2015. 320 с.

Додаткова література

8. Troelsen Andrew, Japikse Phil. Pro C# 8 with .NET Core 3. Foundational Principles and Practices in Programming : 9th ed. USA : Apress, 2020. 1262 p.
9. Jon Skeet. C# in Depth : 4th ed. USA : Manning Publications, 2019. 528 p.
10. John Sharp. Microsoft Visual C# Step by Step. USA : Microsoft Press, 2018. 832 p.
11. Nagel Christian. Professional C# 7 and .NET Core 2.0. Birmingham : Wrox, 2018. 1440 p.

12. Solis Daniel, Schrotenboer Cal. Illustrated C# 7. The C# Language Presented Clearly, Concisely, and Visually : 5th ed. USA : Apress, 2018. 817 p.

13. Perkins Benjamin, Jacob Vibe Hammer, Jon D. Reid. Beginning C# 7 Programming with Visual Studio 2017. Birmingham : Wrox, 2018. 912 p.

14. Sarcar Vaskaran. Design Patterns in C#. A Hands-on Guide with Real-World Examples. USA : Apress, 2018. 465 p.

15. MacDonald M. Pro .NET 2.0 Windows Forms and Custom Controls in C#. USA : Apress, 2006. 1081 p.

16. Албахари, Джозеф, Албахари, Бен. C# 8.0. Карманный справочник. СПб. : Диалектика, 2020. 240 с.

17. Прайс М. Дж. C# 7 и .NET Core. Кросс-платформенная разработка для профессионалов : 3-е изд. СПб. : Питер, 2018. 640 с.

18. Вагнер Билл. Эффективное программирование на C#. 50 способов улучшения кода. М. : Вильямс, 2017. 224 с.

19. Бойко Б. І., Омельчук Л. Л., Русіна Н. Г. Об’єктно-орієнтоване програмування. Лабораторний практикум : навч. посіб. К. : Київський національний університет ім. Тараса Шевченка, 2016. 90 с.

20. Гаско Рик. Объектно-ориентированное программирование. М. : Солон-Пресс, 2016. 298 с.

21. Фленов М. Е. Библия C# : 3-е изд., перераб. и доп. СПб. : БХВ-Петербург, 2016. 544 с.

22. Бублик В. В. Об’єктно-орієнтоване програмування : підручник К. : ІТ-книга, 2015. 624 с.

23. Петцольд Ч. Программирование с использованием Microsoft Windows Forms. Мастер-класс. М. : Русская Редакция; СПб. : Питер, 2006. 432 с.

ДОДАТОК А
Правила оформлення звіту

1. Перед виконанням лабораторної роботи необхідно уважно вивчити теоретичну частину до неї.

2. Створити документ Microsoft Word.

2.1. Встановити параметри сторінки: відступ зліва – 2 см; відступ справа – 1,5 см; відступ знизу – 2 см; відступ зверху – 1,5 см.

2.2. Розмір шрифту – 14, міжрядковий інтервал – одинарний.

2.3. Нумерація звіту повинна бути наскрізною, починаючи з титульного аркуша. Номер сторінки зазначають посередині верхньої частини аркуша над текстом. На титульному аркушу номер сторінки не ставлять, але рахують.

2.4. У звіті можуть бути наведені переліки, перед якими ставлять двокрапку. Перед кожною позицією переліку ставлять риску (–) або рядкову літеру з дужкою. Для подальшої деталізації переліків використовують арабські цифри з дужкою, наприклад:

а) . . . . . .

б) . . . . . .

1) . . . . . .

2) . . . . .

в) . . . . . .

2.5. У тексті звіту не дозволяється: вживати звороти розмовної мови; вживати застарілі та жаргонні терміни і вислови; вживати скорочені слова, крім встановлених стандартами скорочень.

2.6. Якщо в тексті наводиться ряд числових значень в однакових одиницях, то позначення одиниці виміру зазначають тільки після останнього числового значення, наприклад: 1, 2, 3 м; або від 5 до 10 мм. Одиниці вимірювання від числових величин відокремлюють нерозривним пробілом (Ctrl+Shift+Space).

2.7. Числові значення величин треба відокремлювати від десяткової частини комою, наприклад: 7,5; 8,75; 10,00. У необхідних випадках треба використовувати математичне округлення, наприклад: вірно...розмір файлу складає 20 МБ; невірно … розмір файлу складає 20,036 МБ.

2.8. Послідовність розміщення матеріалу у звіті повинна бути наступною:

− титульний аркуш (додаток А);

− зміст;

− лабораторні роботи:

- номер лабораторної роботи (стиль *Заголовок1)*;
- тема та мета роботи;
- анотовані теоретичні відомості, які застосовуються при розв'язанні індивідуальних завдань;
- умова завдання та його розв'язок;
- матеріали самостійної роботи студента згідно виданого завдання (матеріали повинні бути оброблені та систематизовані, не допускається прямого копіювання з джерел інформації);
- висновок до лабораторної роботи;
- список використаних джерел.

− загальний висновок до лабораторного курсу.

2.9. Оформлення ілюстрацій.

2.9.1. Усі ілюстрації у звіті у вигляді креслень, ескізів, схем, графіків, діаграм, фотографій та ін. називаються рисунками.

2.9.2. Рисунки можуть бути виконані олівцем, пастою, тушшю, фломастером та розташовані на окремих аркушах або безпосередньо в тексті записки, якщо рисунки невеликі.

2.9.3. Рисунки нумеруються в межах кожної лабораторної роботи двома цифрами – номером роботи і порядковим номером рисунку в роботі, розділеними крапкою.

2.9.4. Кожний рисунок повинен мати найменування. Слово «Рисунок», його номер та найменування розміщують під рисунком та записують таким чином:

Рисунок 1.3 – Функціональна схема пристрою

2.9.5. Після номеру ставиться тире (–), а після найменування крапка не ставиться.

2.9.6. На усі рисунки повинні бути посилання у тексті роботи, наприклад: ... наведено на рисунку 2.6.

2.9.7. Графіки повинні мати координатні осі та координатну сітку. На координатних осях необхідно наносити числові значення змінних

величин; найменування фізичної величини, яка пишеться текстом паралельно відповідній осі, та через кому позначають одиницю виміру фізичної величини. Напис розміщують поза полем графіка, у кінці напису крапка не ставиться.

2.10. Оформлення таблиць.

2.10.1. Таблиці нумерують у межах кожної роботи арабськими цифрами, розділеними крапкою, та розташовують над таблицею ліворуч. Кожна таблиця повинна мати назву, яку пишуть над таблицею. Перед назвою таблиці пишуть слово «Таблиця» і її номер, який складається з номера розділу і порядкового номера таблиці в межах розділу. Номер таблиці від назви виділяють тире, наприклад:

Таблиця 4.1 – Технічні характеристики модему ADE-4400

2.10.2. Якщо висота таблиці перевищує одну сторінку, її продовження переноситься на наступну сторінку. При цьому лінію, що обмежує першу частину таблиці знизу, не проводять, а над продовженням таблиці на наступній сторінці пишуть «Продовження таблиці 4.1». При переносі таблиці допускається її головку замінювати номерами граф, відповідно до їх номерів у першій частині таблиці.

2.10.3. На всі таблиці повинні бути посилання у тексті записки, наприклад: ... наведено в таблиці 4.1.

2.11. Оформлення формул.

2.11.1. Формули і математичні рівняння подаються у тексті окремим рядком і розташовуються на його середині. Переносити формулу на наступний рядок дозволяється тільки по знаках операцій, який повторюють на початку наступного рядка.

2.11.2. Формули нумерують у межах розділу арабськими цифрами. Номер складається з номера розділу та порядкового номера формули, розділених крапкою. Номер формули записують у круглих дужках на рівні праворуч формули. Посилання на формули у тексті записки дають у дужках, наприклад: ... у формулі (2.1).

2.11.3. Пояснення символів і числових коефіцієнтів, які входять у формулу, необхідно подавати безпосередньо під формулою. Пояснення кожного символу треба давати з нового рядка, причому перший рядок пояснення повинен починатися зі слова «де» без двокрапки після нього.

2.12. Оформлення списку використаних джерел.

2.12.1. Посилання на джерело наводиться у вигляді порядкового номера джерела, взятого в квадратні дужки. Якщо необхідно посилатися одночасно на декілька джерел, їх номери зазначають через кому чи тире, наприклад: [12]; [1,4,7]; [5-9]; [2 с. 4]; [3 таблиця 2.1].

2.12.2. Перелік літературних джерел розміщують у порядку їх згадування у роботі (найзручніший спосіб) або в алфавітному порядку.

2.12.3. Бібліографічний опис джерела в переліку має відповідати вимогам ДСТУ ГОСТ 7.1:2006 «Система стандартів з інформації, бібліотечної та видавничої справи. Бібліографічний запис. Бібліографічний опис. Загальні вимоги та правила складання». Бібліографічний опис дається мовою оригіналу. Прізвища авторів від їх ініціалів відокремлюють нерозривним пробілом (Ctrl+Shift+Space).

2.12.4. Дозволяється також у якості джерел інформації використовувати ресурси глобальної мережі інтернет, проте тільки офіційні сайти виробників обладнання, інтернет-магазинів для складання кошторису тощо. Заборонено включати до переліку використаних джерел такі сайти як www.google.com, www.yandex.ru, www.yahoo.com та ін., які є загальними пошуковими сервісами, а також www.wikipedia.org, www.ukrreferat.com, www.referat.ru та ін., де інформація може мати неперевірений характер і додаватися на сайт неспеціалістами.

3. До захисту поточної лабораторної роботи допускаються студенти, які надали звіт в електронному варіанті і захистили попередні лабораторні роботи.

4. Кожен студент захищає лабораторну роботу індивідуально.

5. На захисті викладач може задати будь-яке питання, що стосується теоретичної частини, і будь-яке завдання іншої бригади або підсумкове.

6. На останньому занятті викладачу подається роздрукований звіт з лабораторних робіт та його електронний варіант.

7. До екзамену допускаються студенти, які захистили всі лабораторні роботи.

ЗМІСТ

М е т о д и ч н і в к а з і в к и
до виконання лабораторних робіт
з дисципліни «Об’єктно-орієнтоване програмування»
для студентів спеціальності
123 «Комп’ютерна інженерія»
усіх форм навчання.
Частина 2

УКЛАДАЧІ: Музика Іван Олегович
Кузнєцов Денис Іванович

Реєстраційний № ___

Підписано до друку _____ 2021 р.
Формат _____А5_____
Обсяг _____111_____ стор.
Тираж _____ прим.

Видавничий центр
Криворізького національного університету,
вул. Віталія Матусевича, 11, м. Кривий Ріг