МІНІСТЕРСТВО ОСВІТИ І НАУКИ УКРАЇНИ

КИЇВСЬКИЙ НАЦІОНАЛЬНИЙ УНІВЕРСИТЕТ БУДІВНИЦТВА І АРХІТЕКТУРИ
ФАКУЛЬТЕТ ІНФОРМАЦІЙНИХ ТЕХНОЛОГІЙ
КАФЕДРА ІНФОРМАЦІЙНИХ ТЕХНОЛОГІЙ

М.І. Цюцюра
Т.А. Гончаренко

МЕТОДИЧНІ ВКАЗІВКИ ДЛЯ ВИКОНАННЯ ЛАБОРАТОРНИХ РОБІТ

з курсу «Об'єктно-орієнтоване програмування»

для студентів напрямів підготовки
«Інформаційні управляючі технології управління»
«Інформаційні технології проектування»
«Безпека інформаційних комп'ютерних систем»
«Комп'ютерні системи та мережі»
«Професійна освіта. Комп'ютерні технології»

Київ 2018

Методичні вказівки до лабораторних робіт з дисципліни “Об’єктно-орієнтоване програмування” (Частина 1) для студентів напрямків підготовки та всіх форм навчання / М. І Цюцюра, Т.А. Гончаренко. – Київ: КНУБА, 2018. – 36 с.

Автори: М. І. Цюцюра, к.т.н., доцент
Т. А. Гончаренко, старший викладач

Відповідальний за випуск: С. В. Цюцюра, д.т.н., професор

Рецензент: Є. В. Бородовка, д.т.н., професор

Затверджено
на засіданні каф.
Інформаційних технологій

Протокол №__11__

від “23” листопада 2017 р.

ЗМІСТ

ВСТУП

Метою даного курсу є вивчення теоретичних основ та практичних аспектів об'єктно-орієнтованого програмування. Дисципліна “Об’єктно-орієнтоване програмування” спрямована на отримання студентом базових знань та практичних навичок з основ сучасної технології створення складних програмних продуктів на базі ідей і принципів об'єктно-орієнтованого методу. Такі знання призначені для використання у розробках програмного забезпечення інформаційних технологій в управлінні, у проектуванні, з урахуванням сучасних вимог у відношенні до надійності, якості інтерфейсу та ефективності програмних продуктів, які створюються. Отримані знання та практичні навички мають служити базою для опанування у подальшому нових майбутніх систем програмування, які базуються на ідеях візуального програмування, CASE-технологіях, баз даних, штучного інтелект та інше.

В якості інструментальної мови програмування для виконання лабораторних робот рекомендовано використовувати мову програмування C#. Головною вимогою для використання компілятора є підтримка стандарту *ISO/IEC 14882 “Standard for the C# Programming Language”*.

ЛАБОРАТОРНА РОБОТА №1
ПРОГРАМУВАННЯ ОБЧИСЛЮВАЛЬНОГО ПРОЦЕСУ

Мета роботи

Вивчити середовище розробки Visual Studio.Net та основи програмування C#, розробити проект C# Windows Forms.

Порядок виконання лабораторної роботи

1. Застосувавши конспект лекцій та додаткову літературу, вивчити основи середовища Visual Studio.Net, створити проект C# Windows Forms з відповідним інтерфейсом для реалізації програми.
2. Виконати завдання відповідно до номеру варіанта та вимогам до виконання, застосувавши мову програмування C#. Для створення програми слід використовувати лінійні, умовні оператори та оператори циклу.
3. Оформити звіт.
4. Зробити висновки.
5. Відповісти на контрольні запитання.

Завдання на лабораторну роботу:

Для програмування циклічного обчислювального процесу створити проект C# Windows Forms, в якому розробити відповідний інтерфейс для введення-виведення даних, використовуючи наступні компоненти: **Label, Textbox, Button, DataGridView та Chart.** Передбачити обробку помилок користувача та виняткових ситуацій при виконанні програми.

Вимоги до програми:

- створити проект C# Windows Forms, розробити відповідний інтерфейс;
- використати компоненти Label та Textbox для введення даних;
- використати компонент Button – кнопку виконання програми;
- використати компоненти DataGridView та Chart для виведення результатів;
- застосувати оператори if, while або for для обчислювального процесу;
- передбачити перевірку виняткових ситуацій та вивести відповідне повідомлення про помилку введення, обробки та виведення даних (наприклад, заборонити введення літер, виключити ділення на нуль тощо).

Приклад виконання роботи

Завдання:

Створити додаток для табулювання і виведення на екран значення функції та побудувати її графік.

$$y = f(x) = \begin{cases} f_1(x), \text{если} & x \le 0 \\ f_2(x), \text{если} & 0 < x \le a \\ f_3(x), \text{если} & x > a \end{cases}$$

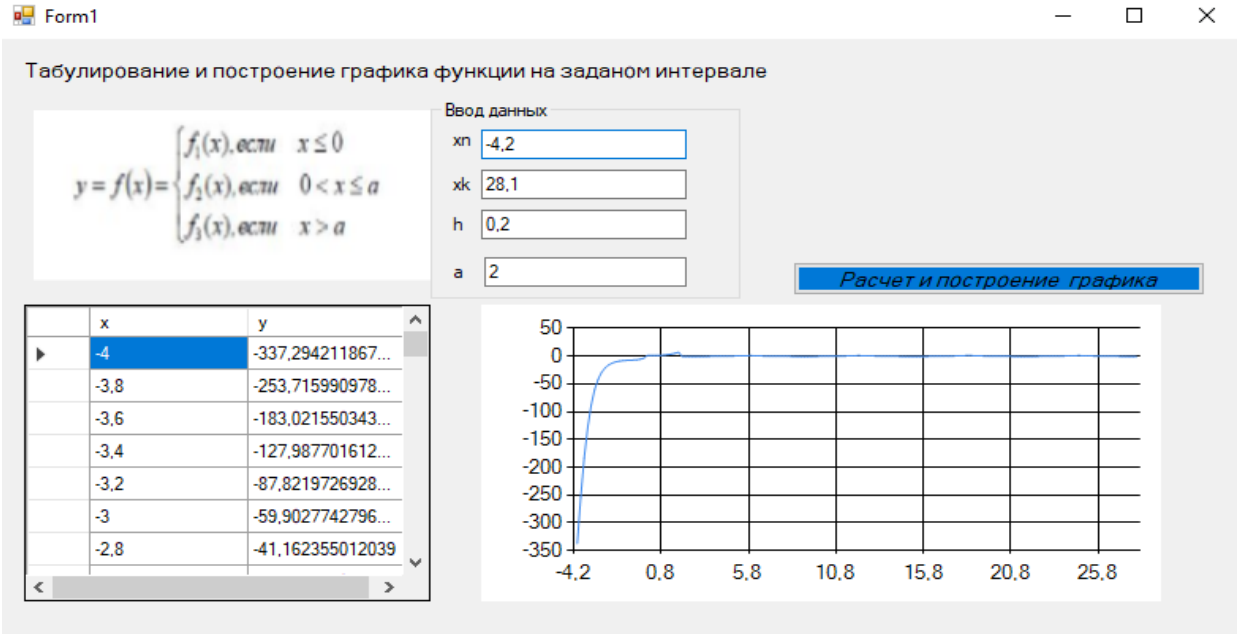
Вариант задания	Функции			Начальное значение xn	Конечное значение xk	Шаг (xh)
	$f_1(x)$	$f_2(x)$	$f_3(x)$			
1	2	3	4	5	6	7
2	$\lvert 5\rvert^{\sin(x)+2}x+\sin(x)$	$x^3+(\lvert x\rvert+1)^{0,1x}$	$\frac{\sin^5(x+2)}{\sqrt[4]{\sin^2 x+\cos^4 x}}$	–4,2	28,1	0,1

Код програми з коментаріями:

```
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 ООП_1
{
   public partial class Form1 : Form
   {
      public Form1()
      {
         InitializeComponent();
      }

      private void button1_Click(object sender, EventArgs e)
      {
         try
         {
            double x, xn, xk, h, a, y;
// Зчитування вхідних даних з елементів textBox
            xn = Convert.ToDouble(textBox1.Text);
            xk = Convert.ToDouble(textBox2.Text);
            h = Convert.ToDouble(textBox3.Text);
            a = Convert.ToDouble(textBox4.Text);
            x = xn;
// Підготовка (очистка) компонентів dataGridView та Chart для виведення даних
            dataGridView1.Rows.Clear();
            chart1.Series[0].Points.Clear();
// Організація оператору циклу
            while (x <= xk)
            {
// Організація умовного оператору
               if (x <= 0)
               {
```

```
                y = (Math.Pow(Math.Abs(5), (Math.Sin(x) + 2))) * x + Math.Sin(x);

            }
            else if (x > 0 && x <= a)
            {
                y = Math.Pow(x, 3) + Math.Pow(Math.Abs(x) + 1, 0.1 * x);
            }
            else y = (Math.Pow(Math.Sin(x + 2), 3)) / (Math.Pow(Math.Pow(Math.Sin(x), 2) +
Math.Pow(Math.Cos(x), 4), 1 / 4));
// Виведення результатів в компоненти dataGridView та Chart
            dataGridView1.Rows.Add(x.ToString(), y.ToString());
            chart1.Series[0].Points.AddXY(x, y);
            x = x + h;
        }
    }
// Обробка виняткових ситуацій
    catch
    {
        MessageBox.Show("Ошибка ввода");
    }
  }

 }
}
```

Результат роботи програми:



Контрольні запитання

1. Дайте загальне визначення класу.Наведіть загальну структуру класу.
2. Що таке інтерфейс класу та його реалізація?
3. У чому полягає роль конструкторів та деструкторів у класі?
4. Які способи можна застосувати для ініціалізації об’єкта класу? Як для цього застосовуються конструктори?

ЛАБОРАТОРНА РОБОТА № 2
ПРОСТІ КЛАСИ ТА ОБ'ЄКТИ. ІНКАПСУЛЯЦІЯ.

Мета роботи

Навчитись створювати класи та екземпляри класів – об'єкти та використовувати їх при розробці програм.

Порядок виконання лабораторної роботи

1. Застосувавши конспект лекцій та додаткову літературу, вивчити основні принципи створювання класів та об'єктів.
2. Виконати завдання відповідно до номеру варіанта, застосувавши мову програмування C#. При виконанні програм слід використовувати конструктори та деструктори.
3. Оформити звіт.
4. Зробити висновки.
5. Відповісти на контрольні запитання.

Завдання на лабораторну роботу:

Створити проект, в якому описати клас для вирішення задачі згідно варіанту. Розроблений клас має містити наступні елементи – закриті та відкриті поля класу, конструктори без параметрів та з параметрами, методи та властивості. Створити екземпляри класу (об'єкти) для кожного типу конструктора.

Вимоги до програми:

- створити проект C# Windows Forms, використовуючи елементи, розробити відповідний інтерфейс для різних варіантів ініціалізації даних;
- при створені класу передбачити різні типи конструкторів:
  1. з параметрами - введення даних для обробки з клавіатури,
  2. з параметрами - введення меж діапазону для генерування випадкових значень;
  3. без параметрів;
- передбачити створення об'єктів одного класу з різними конструкторами;
- обробити виняткові ситуації для некоректного введення даних та вивести відповідне повідомлення про помилку.

Приклад виконання роботи

Завдання:

Визначити клас для правильної чотирикутної піраміди, який містить інформацію про довжину основи та висоту піраміди. Створити метод обчислення її об'єму та площі повної поверхні.

Код програми:

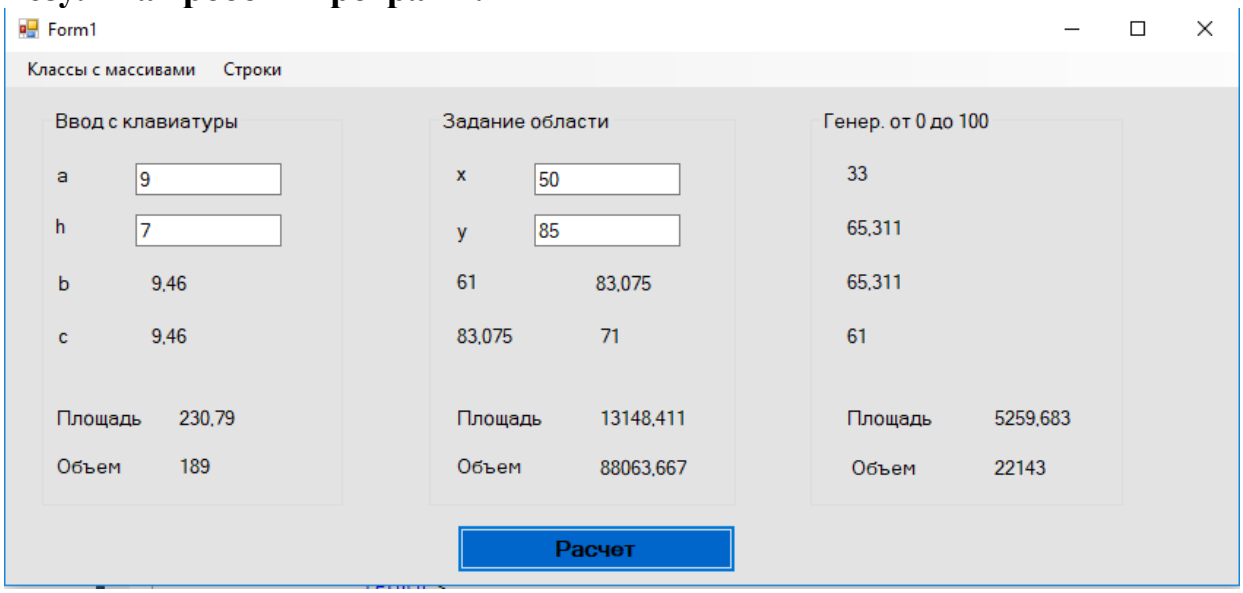
```
using System;
using System.Collections.Generic;
using System.ComponentModel;
using System.Data;
using System.Drawing;
using System.Linq;
```

```
using System.Text;
using System.Threading.Tasks;
using System.Windows.Forms;

namespace ООП__3
{
   public partial class Form1 : Form
   {
      class Piramida
      {
         double a;
         double b;
         double c;
         double h;
         double area;
         double volume;
         // Конструктор з параметрами вхідних значень з клавіатури
         public Piramida(double a1, double h1)
         {
            a=a1;
            h=h1;
         }
         //Конструктор з параметрами меж діапазону
         public Piramida(int x,int y)
         {
            Random ran = new Random();
            a = ran.Next(x, y);
            h = ran.Next(x, y);
         }
         // Конструктор без параметрів, діапазон - від 1 до 100.
         public Piramida()
         {
            Random ran = new Random();
            a = ran.Next(1, 100);
            h = ran.Next(1, 100);
         }
         // Властивість для обчислення повної поверхні
         public double Area
         {
            get
            {
            b = Math.Sqrt((Math.Pow(((a1 * Math.Sqrt(2.0)) / 2.0), 2) + (h1 * h1)));
            c = Math.Sqrt((Math.Pow(((a1 * Math.Sqrt(2.0)) / 2.0), 2) + (h1 * h1))); ;
            area = 4.0 * (Math.Sqrt(((a + b + c) / 2.0) * (((a + b + c) / 2.0) - a) * (((a + b + c) / 2.0) -
b) * (((a + b + c) / 2.0) - c))) + Math.Pow(a, 2.0);
               return area;
            }
         }
         // Властивість для обчислення об’єму
         public double Volume
         {
            get
            {
```

```
            volume = (1.0/3.0)*h*Math.Pow(a,2.0);
            return volume;
          }
        }
// Властивості для передачі в інтерфейс полів класу

        public double A
        {
          get { return a;}
        }
         public double B
        {
          get { return b;}
        }
         public double C
        {
          get { return c;}
        }
         public double H
        {
          get { return h;}
        }

      }  // закриття класу Piramida
      public Form1() // ініціалізація Form1() – запис створюється автоматично
      {
        InitializeComponent();
      }

      private void button1_Click_1(object sender, EventArgs e) // запис створюється автоматично
      {
        Piramida l1, l2, l3; // Об'ява екземплярів класу
        try
        {
          int  x, y;
          double a, h;
          // зчитування вхідних даних з Form1 та ініціалізація змінних
          a = Convert.ToInt16(textBox1.Text);
          h = Convert.ToInt16(textBox4.Text);
          x = Convert.ToInt16(textBox5.Text);
          y = Convert.ToInt16(textBox6.Text);
          // створення екземплярів класу з різними конструкторами
          l1 = new Piramida(a,h);
          l2 = new Piramida(x,y);
          l3 = new Piramida();
         // виведення результатів обробки даних у відповідні мітки Form1
          label27.Text = (Math.Round (l1.B, 3)).ToString();
          label28.Text = (Math.Round(l1.C, 3)).ToString();
          label7.Text = Convert.ToString(Math.Round(l1.Area,3));
          label8.Text = Convert.ToString(Math.Round(l1.Volume,3));
          label17.Text = Convert.ToString(Math.Round(l2.Area,3));
          label18.Text = Convert.ToString(Math.Round(l2.Volume,3));
          label25.Text = Convert.ToString(Math.Round(l3.Area,3));
          label26.Text = Convert.ToString(Math.Round(l3.Volume,3));
```

```
            label11.Text = l2.A.ToString();
            label12.Text = (Math.Round(l2.B, 3)).ToString();
            label13.Text = (Math.Round(l2.C, 3)).ToString();
            label14.Text = l2.H.ToString();
            label19.Text = (Math.Round(l3.A, 3)).ToString();
            label20.Text = (Math.Round(l3.B, 3)).ToString();
            label21.Text = (Math.Round(l3.C, 3)).ToString();
            label22.Text = l3.H.ToString();
        }
        catch
        {
            MessageBox.Show("Помилка введення даних");
        }
    }
```

Результат роботи програми:



Контрольні запитання

1. Що означає об’єкт класу? Уява об’єкта у пам’яті.
2. Чи можливо перевантажувати функції-члени класу?
3. В чому полягає особливість статичних елементів класу?
4. Які області видимості використовують для створення класів?
5. Коли використовують дружні функції?
6. Чи можна присвоювати значення одного об’єкта іншому.
7. Яки специфікатори видимості для членів класу ви знаєте?
8. Чи можливо створювати масиви об’єктів?
9. Для чого використовується покажчик this?
10. Чи можливо передавати об’єкти як аргументи функціям?
11. Чи можливо щоб функція повертала значення типа клас?

ЛАБОРАТОРНА РОБОТА № 3
МАСИВИ ТА ІНДЕКСАТОРИ

Мета роботи

Формування вмінь і навиків програмування алгоритмів обробки одновимірних та двомірних масивів та закріплення вмінь і навиків використання функцій вводу-виводу.

Порядок виконання лабораторної роботи

1. Застосувавши конспект лекцій та додаткову літературу, вивчити основні принципи обробки одновимірних та двомірних масивів.
2. Виконати завдання відповідно до номера варіанта застосувавши мову програмування C#.
3. Оформити звіт.
4. Зробити висновки.
5. Відповісти на контрольні запитання.

Завдання на лабораторну роботу:

Створити окремі класи для обробки одновимірного та двомірного масивів. В класах при кожному доступі перевіряти знаходження елементу закритого масиву та його індексу в заданому діапазоні. Реалізувати методи обробки масивів відповідно до варіанту та передбачити виведення результату у відповідну форму інтерфейсу основної програми.

Вимоги до програми:

- обробку даних для кожного типу масиву організувати в окремих формах, на які здійснюється перехід з головного меню проекту, створеного у лабораторній роботі №2, використовуючи компонент **MenuStrip**;
- у кожній формі передбачити введення значення кількості елементів масиву та меж діапазону випадкових чисел для його ініціалізації;
- передбачити для двовимірного масиву динамічне формування кількості і ширині стовпчиків у компоненті **DataGridView**;
- для обробки елементів масиву використати оператори циклу;
- використати індексатори з ключовими словами – **get, return, set, value;**
- передбачити виняткові ситуації при некоректному введенні та обробці даних та вивести відповідне повідомлення про помилку.
-

Приклад виконання роботи

Завдання № 1 для одновимірного масиву:

Створити масив з випадкових цілочисельних елементів и визначити серед них елемент з мінімальним значенням.

Код програми:

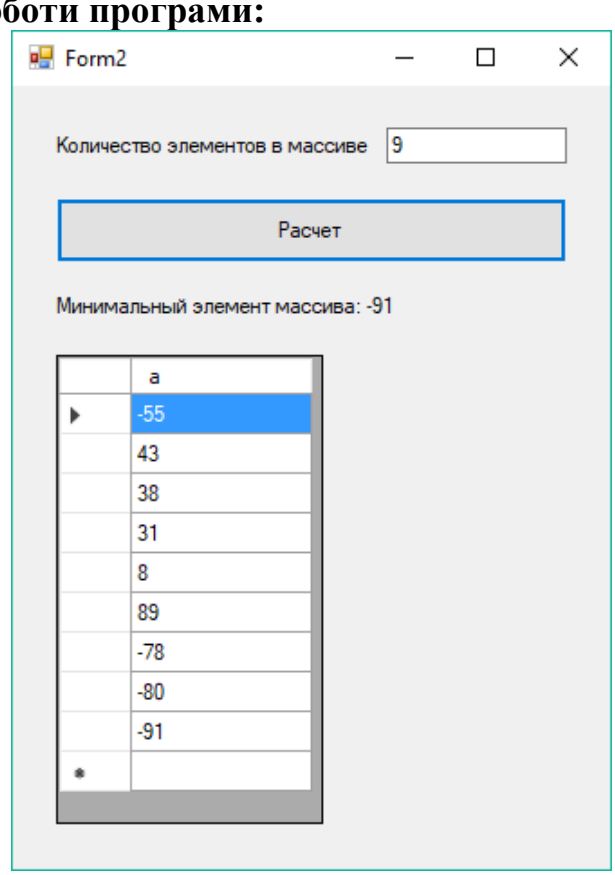
Програмний код класу Arrays:

```
class Arrays
  {
    public bool error = false;
    int[] a; // закритий масив класу
```

```
        int length; // закрите поле класу
        public Arrays (int size) // Конструктор з параметром – кількості елементів масиву
        {
            a = new int[size];
            length = size;
        }
        public int Length // Властивість повернення довжини масиву
        {
            get
            {
                return length;
            }
        }

        public int Min // Пошук мінімального елементу
        {
            get
            {
                int min = a[0];
                for (int i = 0; i < length; i++)
                {
                    if (a[i] < min)
                    {
                        min = a[i];
                    }
                }
                return min;
            }
        }
        public int this[int i] // Індексатор
        {
            get
            {
                if (i >= 0 && i < length) return a[i];
                else { error = true; return 0; }
            }
            set
            {
                if (i >= 0 && i < length && value >= -100 && value <= 100) a[i] = value;
                else error = true;
            }
        }
    }
// Використання цього класу  в Form2.cs:
public partial class Form2 : Form
    {
        public Form2()
        {
            InitializeComponent();
        }

        private void button1_Click(object sender, EventArgs e) //
        {
```

```
        int n;
        dataGridView1.Rows.Clear();
        Random ran = new Random();
        try
        {
            n = Convert.ToInt32(textBox1.Text);
            Arrays a = new Arrays(n); // створення об’єкту класу  Arrays
            // a.Lenght – вбудована функція визначення довжини (кількості елементів) масиву
            for (int i = 0; i < a.Lenght; i++)
            {
                a[i] = ran.Next(-100, 100);
                dataGridView1.Rows.Add(a[i].ToString());
            }
            if (a.error) MessageBox.Show("Помилка обробки даних");
            label2.Text = "Минимальный элемент массива: " + a.Min.ToString();
        }
        catch
        {
            MessageBox.Show("Помилка введення даних");
        }
    }
}
```

Результат роботи програми:



Завдання № 2 для двовимірного масиву:

В таблиці, яка зберігає дані про кількість сировини в кожному з N-цехів, який складається з М-участків, визначити номер того цеху, в якому зберігається найменше сировини.

Код програми:

Створення класу двовимірного масиву:

```
class TwoArray
  {
    public bool error = false;
    int[,] a; // закритий масив класу
    int lengthi; //
    int lenghtj;
    public TwoArray(int size1, int size2) // конструктор з параметрами
    {
      a = new int[size1, size2];
      lengthi = size1;
      lenghtj = size2;
    }
    public int Lenghti
    {
      get
      {
        return lengthi;
      }
    }
    public int Lenghtj
    {
      get
      {
        return lenghtj;
      }
    }
    public int this[int i, int j]  // індексатор двовимірного масиву
    {
      get
      {   // перевірка знаходження індексу масиву в певному діапазоні
        if ((i >= 0 && i < lengthi) && (j >= 0 && j < lenghtj)) return a[i, j];
        else { error = true; return 0; }
      }
      set
      {  // перевірка знаходження елементу масиву та його індексу в заданих межах
        if (i >= 0 && i < lengthi && j >= 0 && j < lenghtj && value >= 0 && value <= 1000)
        a[i, j] = value;
        else error = true;
      }
    }
    public int[] Min
    {
      get
      {
        int s = 0;
        int[] summ = new int[lengthi]; // створення одновимірного масиву
        for (int i = 0; i < lengthi; i++)
        {
          for (int j = 0; j < lenghtj; j++)
          {
            s += a[i, j];
```

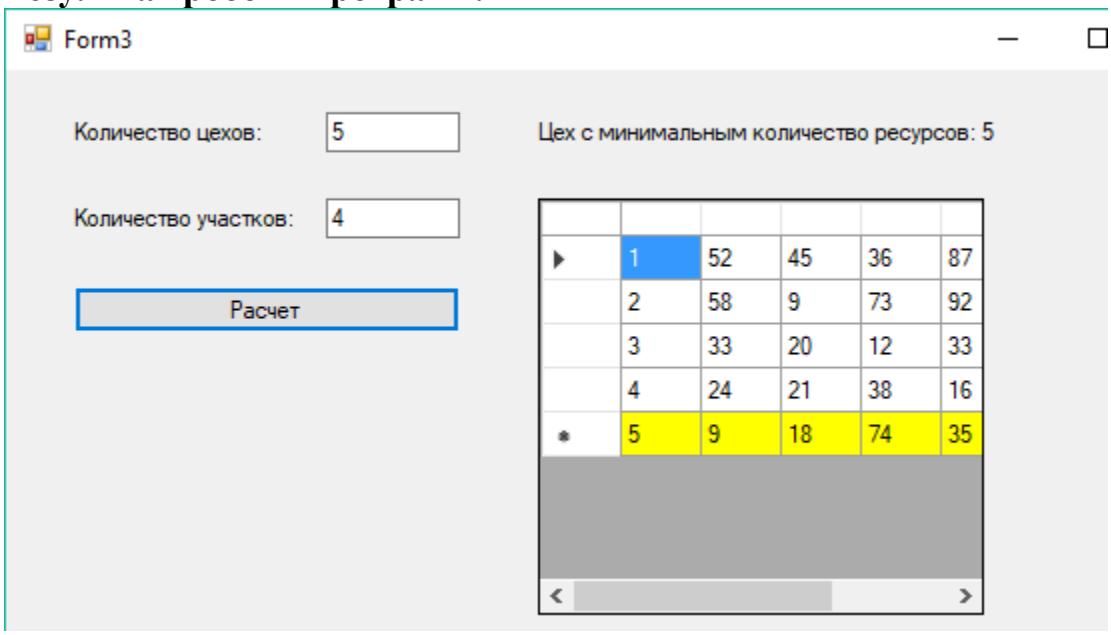
```
            }
            summ[i] = s;
            s = 0;
        }
        int min = summ[0];
        int indexMin = 0;
        int[] massiv = new int[summ.Length + 1];
        for (int i = 0; i < summ.Length; i++)
        {
            if (summ[i] < min)
            {
                indexMin = i;
            }
        }
        massiv[0] = indexMin;
        for (int i = 0; i < summ.Length; i++)
        {
            massiv[i+1] = summ[i];
        }
        return massiv;
    }
  }
}
```

Використання цього класу в Form3.cs:

```
public partial class Form3 : Form
  {
    public Form3()
    {
      InitializeComponent();
    }
    private void button1_Click(object sender, EventArgs e)
    {
      Random ran = new Random();
      int n, m;
      try
      {
        n = Convert.ToInt32(textBox1.Text);
        m = Convert.ToInt32(textBox2.Text);
        DataGridViewTextBoxColumn dgvAge;
        for (int i = 0; i < m; i++)
        {
          dgvAge = new DataGridViewTextBoxColumn();
          dgvAge.Width = 40;
          dataGridView1.Columns.Add(dgvAge);
        }
        dataGridView1.Rows.Clear();
        dataGridView1.ColumnCount = m + 1 ;
        dataGridView1.RowCount = n;
        TwoArray a = new TwoArray(n,m+1);
        for (int i = 0; i < n; i++)
        {
          a[i, 0] = i+1;
          dataGridView1.Rows[i].Cells[0].Value = a[i, 0].ToString();
```

```
                for (int j = 1; j <= m; j++)
                {
                    a[i, j] = ran.Next(0, 100);
                    dataGridView1.Rows[i].Cells[j].Value = a[i,j].ToString();
                }
            }
            if (a.error)
            {
                MessageBox.Show("Ошибка ввода данныхууу");
            }
            int zeh = a.Min[0] + 1;
            label3.Text = "Цех с минимальным количеством ресурсов: " + zeh.ToString() + "";
            int[] massiv = a.Min;
            dataGridView1.Rows[a.Min[0]].DefaultCellStyle.BackColor = Color.Yellow;
        }
        catch
        {
            MessageBox.Show("Ошибка ввода данных");
        }
    }
}
```

Результат роботи програми:



Контрольні запитання

1. Чим відрізняється динамічний масив даних від статичного?
2. Яку структуру має об’єкт динамічного масиву, у чому полягають його особливості?
3. Поясніть особливості виконання операції присвоєння для об’єктів динамічного масиву.
4. Як виконується редагування значень елементів масивів.
5. Застосування індексаторів в класі масивів. Розкрийте алгоритм передачі об’єкта класу як параметра функції.
6. Який механізм повернення значень типа клас?
7. Чи можливо об’єкти класу передавати за посиланням?

ЛАБОРАТОРНА РОБОТА № 4
ОБРОБКА ТЕКСТОВИХ РЯДКІВ

Мета роботи

Формування вмінь і навиків програмування алгоритмів обробки текстових рядків та масивів символів.

Порядок виконання лабораторної роботи

1. Застосувавши конспект лекцій та додаткову літературу, вивчити основні методи та принципи обробки текстових рядків.
2. Виконати завдання відповідно до номера варіанта застосувавши мову програмування C#.
3. Оформити звіт.
4. Зробити висновки.
5. Відповісти на контрольні запитання.

Завдання на лабораторну роботу:

Створити клас зі строковим полем та з двома конструкторами - з параметром та без параметрів, з двома методами - з використанням стандартних функцій **StringBuilder** та без них (за допомогою масиву) для вирішення завдання відповідно до варіанту. При виникненні помилок необхідно обробляти некоректні виключення.

Вимоги до програми:

- доповнити проект, створений у лабораторної роботи № 3;
- значення текстового рядка для обробки ввести з клавіатури та як константу визначити в конструкторі класу;
- для обробки рядків використати оператори циклу;
- результати виконання програми вивести в одному діалоговому вікні, відобразивши в ньому як вхідний так і вихідний текстові рядки одночасно для двох конструкторів двома методами

Приклад виконання роботи

Завдання:

1. Необхідно до попередньої лабораторної роботи № 3 в головне меню: додати пункт Рядки -> Робота з рядками

2. Створити клас Tstring, що містить строкове поле, два конструктора - з параметром та без, і два методи для видалення з рядка всіх символів * і заміни символу «к» на символ «а».

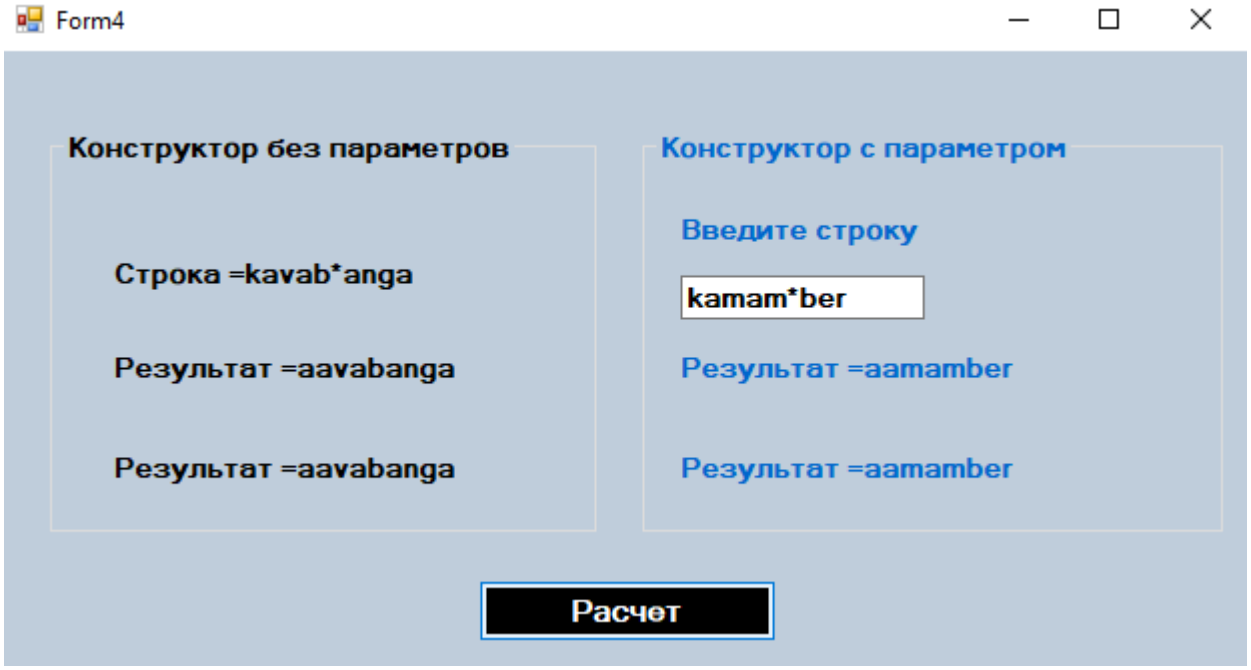
Код програми з коментаріями:

```
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 ООП__3
{
   public partial class Form4 : Form
   {
      // створення класу
      class Tstring
      {
         StringBuilder s;
         public Tstring()  // Конструктор без параметрів
         {
            s = new StringBuilder();
            s = s.Append("kavab*anga");
         }
         public Tstring(string ss) // Конструктор з параметром
         {
            s = new StringBuilder();
            s = s.Append(ss);
         }
         //Метод заміни без стандартної функції
         public StringBuilder rep1()
         {
            StringBuilder s1;
            s1 = new StringBuilder();
            int j = 0;
            for (int i = 0; i < s.Length; i++)
            {
               if (s[i] == 'k') // Знаходження символу 'k' в рядку (масиві) символів
               {
                  s1 = s1.Append('a'); // Заміна символу 'k' на символ 'a'
               }
               else
                  {
                     s1 = s1.Append(s[i]);
                  }
            }
            j = 0; // змінна для підрахунку кількості символу '*' у рядку
           // Організація циклу для пошуку символу '*' в рядку
            for (int i = 0; i < s.Length-j; i++)
            {
               if (s1[i] == '*') // Знаходження символу '*' в рядку (масиві) символів

               {
                  s1[i] = s1[i+1]; //Переміщення на позицію символу '*' наступного символу
                  j++; // підрахунок  кількості символу '*' у рядку
               }
               else s1[i] = s1[i+j];
            }
            s1.Remove(s1.Length - j, j); // Зменшення довжини рядка на кількість
                                         //   переміщених символів
            return s1;
         }
```

```
        //Метод заміни із стандартною функцією
        public StringBuilder rep2()
        {
            StringBuilder s1 = new StringBuilder();
            s1 = s1.Append(s);
            s1.Replace("k", "a");
            for (int i = 0; i < s1.Length; i++)
            {
                if (s1[i] == '*')
                    s1.Remove(i , 1);
            }
            return s1;
        }
        public StringBuilder S
        {
            get
            {
                return s;
            }
        }
    }

    public Form4()
    {
        InitializeComponent();
    }
    private void button1_Click(object sender, EventArgs e)
    {
        Tstring s1, s2;
        s1 = new Tstring();  // Створення об’єкту № 1без параметрів
        label1.Text = "Строка =" + s1.S;
        label2.Text = "Результат =" + s1.rep1(); // Виведення результату для об’єкту №1
        label3.Text = "Результат =" + s1.rep2();
        s2 = new Tstring(textBox1.Text); // Створення об’єкту № 2з параметрами
        label5.Text = "Результат =" + s2.rep1();// Виведення результату для об’єкту №2
        label6.Text = "Результат =" + s2.rep2();

    }
}
}
```

Результат роботи програми:



Контрольні запитання

1. Які типи даних використовуються для зберігання текстових рядків і символів в C#?
2. Як записується операція поєднання текстових рядків? Чим ця операція відрізняється від операції додавання чисел?
3. Як записується мовою C# функція для визначення довжини рядка? Скільки разів доцільно використовувати цю функцію в програмі і чому?
4. Що таке порожній рядок? Як він записується в програмах? Яка його довжина?
5. Як виконати арифметичну операцію з числом, записаним у вигляді рядка символів? Як перевести число в текстовий рядок?

ЛАБОРАТОРНА РОБОТА №5
ПЕРЕВАНТАЖЕННЯ ОПЕРАЦІЙ

Мета роботи

Навчитись використовувати перевантаження математичних та логічних, унарних та бінарних операцій при розробці класів.

Порядок виконання лабораторної роботи

1. Застосувавши конспект лекцій та додаткову літературу, вивчити основні принципи перевантаження.
2. Виконати завдання відповідно до номера варіанта застосувавши мову програмування C#.
3. Оформити звіт.
4. Зробити висновки.
5. Відповісти на контрольні запитання.

Завдання на лабораторну роботу:

Розроблений клас має містити наступні елементи – закриті та відкриті поля класу, конструктори (один з них має передавати масив в якості параметра) та перевантажні математичні або логічні операції, згідно варіанту завдання. В програмі має бути перевірка всіх розроблених елементів класу.

Вимоги до програми:

- доповнити проект, створений у лабораторної роботи № 3;
- описати в класі за допомогою ключового слова **operator** перевантаження операцій як відкритий статичний метод;
- параметри в клас мають передаватися по значенню;
- перевантажні операції не повинні змінювати значення оператора, що передається;
- перевантажна операція, яка повертає величину типу **class,** повинна створювати новий об’єкт цього класу, виконати з ним певні дії та передати його як результат;
- обробити виняткові ситуації для некоректного введення даних та вивести відповідне повідомлення про помилку.

Приклад виконання роботи

Завдання:

Визначити клас для роботи с одновимірним масивом. Клас має реалізовувати можливість визначення добутку елементів масиву на заданий скаляр. В класі реалізувати перевантаження операції – добуток.

Код програми з коментарями:

```
class Arrays
    {
        public bool error = false;
        int[] a;
        int length;
        public Arrays(int size) // створення конструктора
        {
            a = new int[size];
```

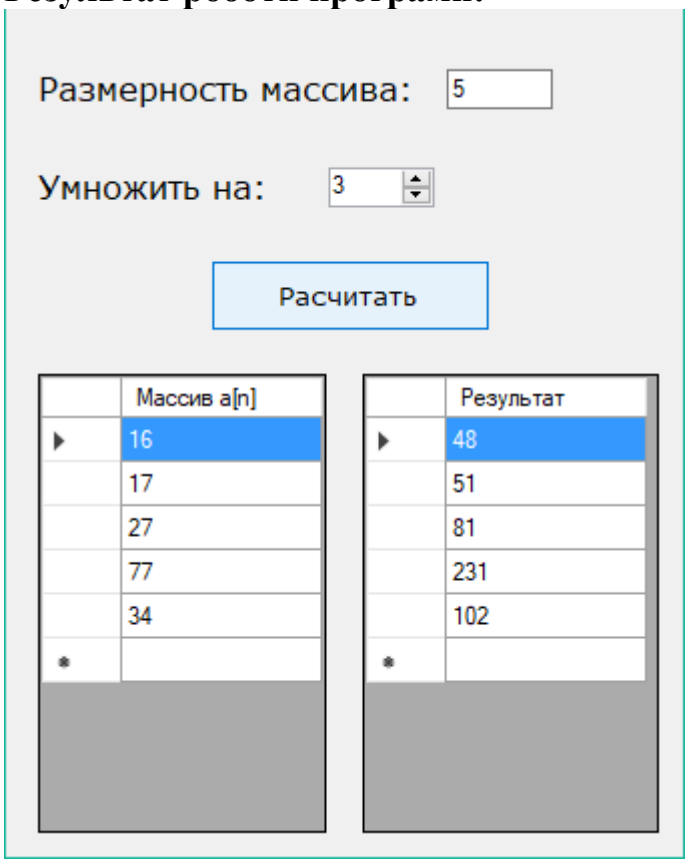
```
            length = size;
        }
        public int Length
        {
            get
            {
                return length;
            }
        }
        public static Arrays generateRandomArray(Arrays array)
        {
            Random rand = new Random();
            for (int i = 0; i < array.Length; i++)
            {
                array[i] = rand.Next(0, 100);
            }
            return array;
        }
        public static Arrays operator *(Arrays arrays, int mul)
        {
            Arrays resArray = new Arrays(arrays.Length);
            for (int i = 0; i < resArray.length; i++)
            {
                resArray[i] = arrays[i] * mul;
            }
            return resArray;
        }
        public int this[int i] //Індексатор
        {
            get
            {
                if (i >= 0 && i < length) return a[i];
                else { error = true; return 0; }
            }
            set
            {
                if (i >= 0 && i < length && value >= -100 && value <= 10000) a[i] = value;
                else error = true;
            }
        }
    }
```

Використання розробленого класу Arrays в Form1.cs:

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

        private void button1_Click(object sender, EventArgs e)
        {
            int n = 15;
            dataGridView1.Rows.Clear();
            dataGridView2.Rows.Clear();
            Random ran = new Random();
            try
            {
                n = Convert.ToInt32(textBox2.Text);
                //mul = Convert.ToInt32(textBox1.Text);
                int mul = (int)numericUpDown1.Value;
                Arrays a = new Arrays(n);
                Arrays resArray = new Arrays(n);
```

```
                a = Arrays.generateRandomArray(a);

                resArray = a * mul; // перевантаження операції добутку

                for (int i = 0; i < a.Length; i++)
                {
                    dataGridView1.Rows.Add(a[i].ToString());
                    dataGridView2.Rows.Add(resArray[i].ToString());
                }
            }
            catch
            {
                MessageBox.Show("Ошибка ввода данных");
            }
        }
    }
```

Результат роботи програми:



Контрольні запитання

1. Для чого використовують перевантаження операцій?
2. В чому різниця між перевантаженням операцій як членів класу і дружніх функцій ?
3. Які оператори не можна перевантажувати як дружні функції?
4. Які оператори не можна перевантажувати як члени класу?
5. В чому полягає особливість перевантаження операторів введення/виведення “<<”, “>>”?
6. Назвіть особливості перевантаження унарних та бінарних операторів.
7. Чи можливо потік передавати до функції за значенням?
8. Чи можливо змінювати пріоритет операції?

ЛАБОРАТОРНА РОБОТА № 6
СПАДКУВАННЯ

Мета роботи

Навчитись використовувати спадкування при розробці об’єктно-орієнтованих програм.

Порядок виконання лабораторної роботи

1. Застосувавши конспект лекцій та додаткову літературу, вивчити основні принципи спадкування ООП.
2. Виконати завдання відповідно до номера варіанта застосувавши мову програмування C#.
3. Оформити звіт.
4. Зробити висновки.
5. Відповісти на контрольні запитання.

Завдання на лабораторну роботу:

Розробити програму згідно варіанту з одним базовим (батьківським) класом і похідним (нащадком) класом.

Вимоги для виконання:

- всі поля повинні бути закритими.
- базовий клас повинен містити конструктори з параметрами, методи доступу до закритих полів, виведення полів та розроблений вказаний в таблиці метод.
- похідний клас повинен містити доповнення та зміни, реалізовувати виведення нових полів нащадка, при цьому назви методів збігаються з назвами методів базового класу.
- скласти тестуючи програму з видачею результатів.
- створити об'єкти базового і похідного класів.
- у програмі повинна виконуватися перевірка всіх розроблених елементів класу.

Теоретичні відомості

Клас в C # може мати довільну кількість похідних класів - нащадків і тільки один базовий клас - предка. При описі класу ім'я його предка записується в заголовку класу після двокрапки:

<специфікатор доступу> **class** <назва похідного класу> **:** <специфікатор доступу> <назва базового класу>

{

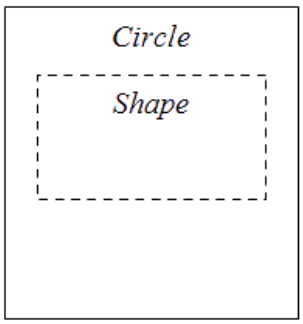
// тіло класу

}

Якщо ім'я предка не вказано, тоді предком вважається базовий клас всієї ієрархії System.Object.

Відношення успадкування між класами може бути представлено у графічному вигляді наступним чином:

Популярною та ефективною реалізацією поняття похідних класів є представлення об'єкта похідного класу у вигляді об'єкта базового класу та інформації, що відноситься тільки до похідного класу.



Похідний клас може сам, у свою чергу, служити базовим класом.

Якщо члени базового класу не були перевизначені в похідних класах, тоді вони позначаються і трактуються так само, як і члени похідного класу. В такому випадку кажуть, що члени базового класу **успадковуються** похідним класом. Операція вирішення області **видимості ::** може вживатися для явного посилання на член базового класу. Це забезпечує доступ до імені, яке може бути перевизначеним в похідних класах.

```
class Base
{
public: int a, b;
}
class Derived : public Base
{
public: int b, c;
}
Derived d;    // Створення обєкту d класу Derived
d.a      = 1;  // Ініціалізація a, успадкованого з класу Base
d.Base.b = 2;        // Ініціалізація b з класу Base
d.b      = 3;  // Ініціалізація b, оголошеного в класі Derived
d.c      = 4;
```

Клас називається *безпосереднім базовим класом,* якщо він згадується при оголошенні похідного класу, і *непрямим базовим класом,* якщо він є базовим класом для одного з базових класів оголошуємо класу.

Специфікатори доступу для базових класів

Специфікатор доступу може приймати значення:

- *public* - у цьому випадку публічні члени базового класу стають публічними членами похідного класу, а захищені члени базового класу стають захищеними членами похідного класу;
- *protected* - у цьому випадку публічні і захищені члени базового класу стають захищеними членами похідного класу;

- *private* - у цьому випадку публічні і захищені члени базового класу стають приватними членами похідного класу.

Приватні члени класу недоступні у похідних класах, якщо тільки він не оголошений як дружній. У випадку коли специфікатор доступу не вказано (за замовчанням) використовується специфікатор доступу *private.*

Розглянемо приклад використання різних специфікаторів доступу для членів класу:

```
class Base
{
private     int a;
protected int b;
public      int c;
}
class Derived1 : public Base
{...  // a - недоступний
      // b - захищений член класу Derived1
      // c - публічний член класу Derived1
}
class Derived2: protected Base
{...  // a недоступний
      // b i c - захищені члени класу Derived2
}
class Derived3: private Base
{...  // a недоступний
      // b i c - приватні члени класу Derived3
}
```

Виклик конструкторів базового класу

Конструктори не успадковуються, тому похідний клас повинен мати власні конструктори. Порядок виклику конструкторів визначається такими правилами:

1. Якщо в конструкторі похідного класу явний виклик конструктора базового класу відсутня, автоматично викликається конструктор базового класу без параметрів.
2. Для ієрархії, що складається з декількох рівнів, конструктори базових класів викликаються, починаючи з самого верхнього рівня. Після цього виконуються конструктори тих елементів класу, які є об'єктами, в порядку їх оголошення в класі, а потім виповнюється конструктор класу. Таким чином, кожен конструктор ініціалізує свою частину об'єкта.
3. Якщо конструктор базового класу вимагає вказівки параметрів, він повинен бути явним чином викликаний в конструкторі похідного класа в списку ініціалізації. Виклик виконується за допомогою ключового слова **base.**

Формат розширеного оголошення такий:

```
назва_похідного_класа(список_параметрів):base (список_аргументів)
{
// тіло конструктора
}
```

Тут за допомогою елемента (список_аргументів) задаються аргументи, необхідні конструктору в базовому класі. За допомогою ключового слова base можна викликати будь-який конструктор, визначений в базовому класі. Реально ж виконається той конструктор, параметри якого будуть відповідати переданим при виклику аргументів. При відсутності ключового слова base автоматично викликається конструктор базового класу, який діє за замовчуванням.

Успадкування та приховування імен

Новим членам (полям, методам і властивостям) похідного класу можна давати імена, що збігаються з іменами членів базового класу. В цьому випадку перед членом похідного класу необхідно поставити ключове слово **new.** При цьому, хоча відповідні члени базового класу успадковуються, вони стають прихованими в похідному класі. Коли ім'я члена в похідному класі приховує член з таким же ім'ям в базовому класі, для звернення до останнього застосовується посилання base, яка завжди вказує на базовий клас похідного класу.

Формат її записи такий:

base. <член базового класу>

Тут в якості елемента <член базового класу> можна вказувати або метод, або змінну екземпляра класу.

Приклад виконання роботи

Завдання:

Створити базовий клас, в якому визначаються двовимірні геометричні фігури та похідний клас для трикутника, в якому визначаються його тип та площа.

Код програми з коментаріями:

```
using System;
{

class TwoF // клас TwoF
{
        public double w; // відкритий член. ;
        double h; // закритий член.
        public TwoF() // конструктор за замовчуванням,
        {
                w = h = 0.0;
        }
        public TwoF(double w1, double h1) //конструктор з  параметрами
        {
                w = w1;
                h = h1;
        }
        public TwoF(double x) // конструктор з  параметром, в якому висота дорівнює ширині
        {
                w = h = x;
        }
        public double height // Свойство height.
        {
                get { return h; }
```

```
                set { h = value; }
        }
        public void showD()
        {
                Console.WriteLine("Ширина та висота = "+width+" та "+height);
        }
}

class Treug:TwoF //Класс Treug, похідний від класу TwoF.
        {
                string style; // закритий член
// Конструктор за замовчуванням, які автоматично викликає конструктор
за замовчуванням класу TwoF.
        public Treug()
        {
                style = "null";
        }
// конструктор з 3 аргументами
        public Treug(string s,double w,double h): base(w,h)
        {
                style = s;
        }
        public Treug(double x):base(x) // створюємо рівнобічний трикутник
        {
                style = " рівнобічний ";
        }
        public double Pl() // метод (як властивість) підрахунку площі трикутника
        {
                return w * height / 2;
        }
        public void showStyle() // визначаємо тип трикутника,
        {
                Console.WriteLine("Трикутник " + style);
        }
}
class Class4
{
public static void Main()
{
        Treug tl = new Treug(); // створюємо об’єкт класу Treug () без параметрів
// створюємо об’єкт класу Treug () з трьома параметрами:
        Treug t2 = new Treug("прямокутний", 8.0, 12.0);
// створюємо об’єкт класу Treug () з одним параметром
        Treug t3 = new Treug(4.0); tl = t2;
        Console.WriteLine("Інформація про tl: " ) ;
        tl.showStyle();
        tl.showD();
        Console.WriteLine ("Площа= "+tl.Pl());
        Console.WriteLine("Інформация про t2: " );
        t2.showStyle();
        t2.showD();
        Console.WriteLine("Площа =  "+t2.Pl());
        Console.WriteLine("Інформація про t3: " );
```

```
t3.showStyle();
t3.showD ();
Console.WriteLine("Площа =  "+t3.Pl());
}
}
```

Контрольні запитання

1. Чи відрізняється уява об’єкту похідного класу у пам’яті комп’ютера від атрибутів доступу?
2. На що впливають атрибути доступу?
3. Що означає множинне спадкування?
4. Перерахуйте основні правила спадкування.
5. Коли використовуються простір імен?
6. Що містить простір імен *std*?
7. Як підключити простір імен?
8. Чи можливо підключити декілька просторів імен?
9. Особливості організації простора імен C#.
10. Як зробити компоненти доступними у похідному класі, але закритими від зовнішнього доступу?

СПИСОК ІНФОРМАЦІЙНИХ ДЖЕРЕЛ

1. Шилдт Г. C#: полное руководство.-М.:ООО “Вильямс”, 2011 .-1056с.
2. Культин Н.Б. Microsoft Visual C# в задачах и примерах. - СПб.: БХВ-Петербург, 2009. – 320 с.
3. Майо Дж. Самоучитель Microsoft Visual Studio 2010 - СПб.: БХВ-Петербург, 2011. – 464 с.
4. Стиллмен Э., Грин Дж. Изучаем C# - СПб.: Питер, 2014. – 816с.
5. Албахари Джозеф, Албахари Бен C# . Справочник.-М.:ООО “Вильямс”, 2014 .-1008с
6. Інформаційний ресурс - https://www.litmir.me/bd/?b=250895
7. Інформаційний ресурс -http://mycsharp.ru
8. Інформаційний ресурс https://professorweb.ru/my/csharp/charp_theory/