Об’єктно-орієнтоване програмування

Шпортько О. В.

Лабораторний практикум
для студентів денної та заочної форм навчання

Освітньої програми	Інженерія програмного забезпечення Інтернету речей
Рівня вищої освіти	першого (бакалаврського)
За спеціальністю	121 Інженерія програмного забезпечення
Галузі знань	12 Інформаційні технології

Рівне 2023

Міністерство освіти і науки України

ПВНЗ «Міжнародний економіко-гуманітарний університет
імені академіка Степана Дем'янчука»

Факультет кібернетики

Кафедра інформаційних систем та обчислювальних методів

Об'єктно-орієнтоване програмування

Лабораторний практикум
для студентів денної та заочної форм навчання

Освітньої програми	Інженерія програмного забезпечення Інтернету речей
Рівня вищої освіти	першого (бакалаврського)
За спеціальністю	121 Інженерія програмного забезпечення
Галузі знань	12 Інформаційні технології

Рівне 2023

УДК 004.422/004.424
ББК 32.973-01
Ш 84

Друкується за рішенням Навчально-методичної комісії ПВНЗ "Міжнародний економіко-гуманітарний університет імені академіка Степана Дем’янчука" (протокол № 10 від 20.06.2023 р.).

Об’єктно-орієнтоване програмування. Лабораторний практикум для студентів денної та заочної форм навчання освітньої програми «Інженерія програмного забезпечення Інтернету речей» першого (бакалаврського) рівня вищої освіти за спеціальністю 121 Інженерія програмного забезпечення галузі знань 12 Інформаційні технології / уклад. О. В. Шпортько. Рівне: ПВНЗ "МЕГУ ім. акад. С. Дем’янчука", 2023. 116 с.

Укладач: О. В. Шпортько, кандидат технічних наук, доцент, доцент кафедри інформаційних систем та обчислювальних методів ПВНЗ "Міжнародний економіко-гуманітарний університет імені академіка Степана Дем’янчука".

Рецензенти: А. Я. Бомба, доктор технічних наук, професор, професор кафедри комп’ютерних наук та прикладної математики Національного університету водного господарства та природокористування;

П. С. Янчук, кандидат фізико-математичних наук, доцент, професор кафедри інформаційних систем та обчислювальних методів ПВНЗ "Міжнародний економіко-гуманітарний університет імені академіка Степана Дем’янчука".

Відповідальний за випуск Ю. Г. Лотюк, кандидат педагогічних наук, доцент, завідувач кафедри інформаційних систем та обчислювальних методів ПВНЗ "Міжнародний економіко-гуманітарний університет імені академіка Степана Дем’янчука".

Лабораторний практикум розроблений у відповідності з практичною частиною робочої програми дисципліни «Об’єктно-орієнтоване програмування» для студентів денної та заочної форм навчання освітньої програми «Інженерія програмного забезпечення Інтернету речей» першого (бакалаврського) рівня вищої освіти за спеціальністю 121 Інженерія програмного забезпечення. Містять впорядкований набір завдань і вказівок до виконання лабораторних робіт та самостійного опрацювання матеріалу дисципліни. Основна увага приділяється вивченню базових принципів ООП та формуванню вмінь і навиків програмування їх практичного застосування засобами мови програмування C#. Призначений для студентів, викладачів та всіх, хто прагне оволодіти підходами ООП для ефективної обробки даних.

Затверджені кафедрою інформаційних систем та обчислювальних методів ПВНЗ "МЕГУ імені академіка Степана Дем’янчука" (протокол № 11 від 13.06.2023 р.).

Схвалені навчально-методичною комісією факультету кібернетики ПВНЗ "МЕГУ імені академіка Степана Дем’янчука" (протокол № 11 від 13.06.2023 р.).

ББК 32.973-01

Зміст

Передмова

На сьогоднішній день важко уявити собі діяльність сучасного суспільства без комп'ютерної техніки, локальних та глобальних мереж і відповідного програмного забезпечення. З комп'ютерами ми стикаємося всюди: у банку та відділі бухгалтерії, в пенсійному фонді та податковій інспекції, у паспортному столі та на залізничному вокзалі, в ЖЕКу та Укртелекомі, в редакціях газет та друкарнях, в рекламних агенціях та конструкторських бюро. Високі темпи накопичення інформації, вимоги мобільності, оперативності поширення та надійності зберігання даних, швидкості та достовірності їх обробки спонукають розвиток апаратних засобів, системних та прикладних оболонок і пакетів. На сьогоднішній день 97 % населення світу користується мобільними телефонами, а кількість комп'ютерів у світі перевищує 1361 мільйонів і продовжує зростати. При цьому приблизно один раз в півтора року подвоюються показники основних технічних характеристик апаратних засобів, один раз в два-три роки змінюються покоління програмного забезпечення і один раз на п'ять-сім років змінюється база стандартів, інтерфейсів та протоколів. Разом з тим, темп кількісного зростання інформаційних систем значно перевищує темп підготовки спеціалістів, здатних ефективно працювати з ними.

В цих умовах кожен випускник ВНЗ з галузі інформаційних технологій, що має використовувати комп'ютерну техніку в своїй професійній діяльності для розробки, експлуатації та супроводження інформаційних систем, повинен не лише володіти елементарними навиками використання програмного забезпечення (на сьогодні це повинна вміти кожна освічена людина), а й розуміти внутрішню організацію системного та прикладного забезпечення, розробляти алгоритми розв'язку задач, кваліфіковано та продуктивно працювати з сучасними оболонками мов програмування, вміти відлагоджувати створені програми та переконуватися в їх дієздатності, розробляти сучасний інтерфейс для створеного програмного коду та здійснювати його адаптацію до потреб користувача, що неможливо без розуміння принципів ООП. Формуванню саме таких фахівців має сприяти вся дисципліна та даний лабораторний практикум зокрема. Після опанування матеріалу практикуму студент повинен вміти швидко та якісно розробляти візуальні додатки для розв'язування прикладних завдань з використанням засобів сучасних операційних систем та оболонок мов програмування.

Даний лабораторний практикум містить впорядкований набір завдань і вказівок до виконання лабораторних робіт та самостійного опрацювання матеріалу дисципліни. Виконання завдань кожної лабораторної роботи вимагає використання знань, умінь та навиків, здобутих під час виконання попередніх робіт, що, з одного боку, дозволяє автоматично повторювати вивчений матеріал, а з іншого – сприяє системності його викладу.

Перед завданнями кожної лабораторної роботи з метою актуалізації знань, отриманих під час лекційних занять, наведено відповідні контрольні запитання.

Всі лабораторні роботи виконуються студентами поваріантно (номер варіанту студента співпадає з його порядковим номером у списку підгрупи). Відлагодження програм та перевірка різних тестових випадків виносяться на самостійне опрацювання. За результатами виконання кожної лабораторної роботи студент оформляє письмовий звіт, що складається з наступних розділів:

- теми та мети лабораторної роботи;
- відповідей на контрольні запитання лабораторної роботи;
- формулювання, математичного розв'язку (при необхідності), тексту програми та результатів тестування (запропонованих у завданні та розроблених самостійно вхідних даних і відповідних результатів виконання програми) кожного завдання лабораторної роботи;
- висновків стосовно знань, умінь та навичок, здобутих при виконанні лабораторної роботи.

Лабораторна робота № 1

Тема. Ознайомлення з візуальним середовищем розробки додатків MS Visual Studio. Створення найпростіших додатків-форм.

Мета. Формування вмінь і навиків роботи у середовищі візуального програмування. Закріплення вмінь і навиків використання функцій вводу-виводу. Застосування вмінь і навиків програмування алгоритмів лінійної структури.

Контрольні запитання

1. У чому полягає принцип швидкої розробки застосувань?
2. На які частини умовно розділяють вікно середовища розробки Visual Studio?
3. Що таке проект в Visual Studio?
4. Як створити проект з користувацькими вікнами (формами)?
5. Які основні файли проекту?
6. Як створити і переглянути процедури обробки подій форми і окремого елемента керування?
7. В якому файлі міститься головна програма у віконному проекті?

Хід роботи

1. Завантажте середовище візуального програмування **Visual Studio**.
2. Створіть у власній папці вкладену папку **OOPLR1**.
3. Використовуючи пункт меню **Файл – Создать – Проект – Шаблоны – Visual C# - Приложение Windows Forms**, створіть додаток з однією формою, зберігши його файли у створеній папці. Проект також назвіть **OOPLR1**.
4. Використовуючи можливості пункту головного меню **Вид**, виведіть на екран панелі **Обозреватель решений** та **Панель элементов** і вікно властивостей, як на рис. 1.

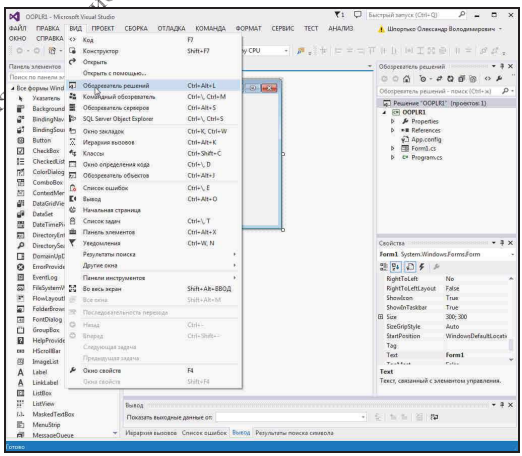


Рис. 1. Відображення оглядача рішень, панелі елементів та вікна властивостей

5. В панелі **Обозреватель решений** перегляньте структуру створеного проекту. Встановіть призначення кожного файла створеної форми. Як з цього оглядача, використовуючи контекстне меню, вивести форму у візуальному режимі, а як переглянути її код?
6. Виділіть у візуальному редакторі форму, перейдіть у вікно її властивостей та віднайдіть перелік властивостей зовнішнього вигляду (рис. 2). Серед цих властивостей віднайдіть властивість **Text** і введіть у неї текст *Вас вітає студент групи <шифр групи> <прізвище, ім'я по батькові>*, вказавши свої дані. Завантажте додаток на виконання (кнопка 

або пункт головного меню **Отладка – Начать отладку**). Що задає властивість **Text**?

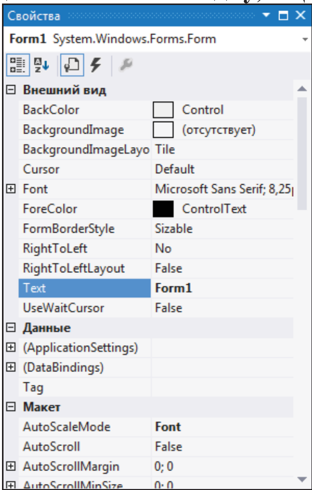


Рис. 2. Перелік властивостей форми

7. Поверніться до програмування додатку (закрийте завантажену програму-додаток, а не Visual Studio). Використовуючи панель елементів **Все формы Windows Forms**, виберіть елемент керування мітку (**Label**) та натягніть її прямокутну область у верхній частині форми. Введіть у її властивості **Text** текст вашого улюбленого прислів'я (рис. 3). Задайте для мітки нестандартний колір тексту та розмір шрифта (у складеній властивості **Font** вікна властивостей). Відмініть автоматичне встановлення розміру за розмірами тексту і встановіть вирівнювання тексту надпису по центру як по вертикалі, так і по горизонталі (властивість **TextAlign**).

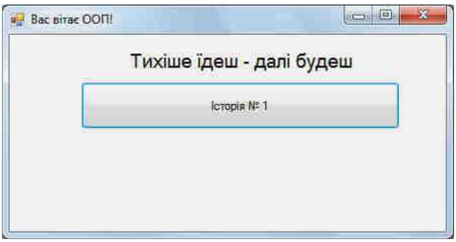


Рис. 3. Вигляд форми в режимі конструктора

8. Натягніть у формі нижче надпису кнопку **Button** (див. рис. 3). Задайте для неї надпис **Історія № 1**. Для відображення при натисненні кнопки діалогового вікна з текстом у вікні властивостей перейдіть на вкладку подій кнопки (рис. 4), віднайдіть подію **Click** та двічі клацніть мишкою у полі біля неї. Яка при цьому створилася процедура обробки події? Як інтерпретується її назва?

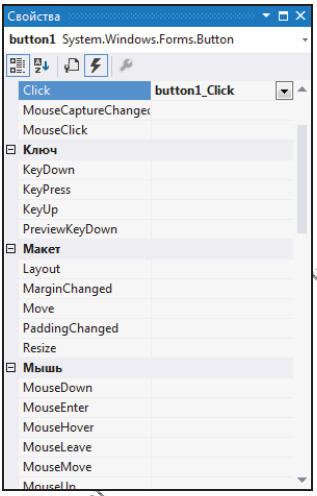


Рис. 4. Перелік подій кнопки

9. В процедурі обробки події натиснення кнопки введіть команду *MessageBox.Show("Розпочинаємо вивчення основ ООП");*.
10. Завантажте програму на виконання та переконайтеся в її дієздатності.
11. Самостійно створіть нижче ще 4 кнопки для відображення ваших різних історій з різними заголовками, кнопками та зображеннями.
12. У нижній частині форми створіть шосту кнопку з надписом *Завершити роботу програми*. Для забезпечення її функціонування в процедурі обробки події **Click** введіть команду *close()*;.
13. Встановіть розмір виконуваного файла програми. Для цього в оглядачі рішень виділіть назву проекту, оберіть в її контекстному меню пункт **Открыть папку в проводнике**, перейдіть в провіднику з відкритої папки у вкладену папку **\Bin\Debug** та визначте розмір додатку з назвою вашого проекту.

Лабораторна робота № 2

Тема. Розробка елементарних класів. Використання різних видів конструкторів.

Мета. Формування вмінь і навиків проектування елементарних класів та використання різних видів конструкторів. Закріплення вмінь і навиків використання функцій вводу-виводу. Застосування вмінь і навиків програмування алгоритмів лінійної, розгалуженої та циклічної структури.

Контрольні запитання

1. Який синтаксис опису класів в C#?
2. Який початковий клас C# в ієрархії наслідування? Які його метод використовуються найчастіше?
3. Які області видимості використовуються для класів, а які – для членів класів?
4. Для чого використовуються конструктори та деструктори? Які види конструкторів використовуються в C#?
5. Як створити об'єкт класу?

Завдання.

1. Створіть додаток-форму Windows. Опишіть в модулі форми *під класом форми* базовий клас згідно варіанту, передбачивши в ньому не менше трьох полів, двох конструкторів різних видів, двох методів для виконання обчислень. Створіть також у цьому класі метод *Info* для виводу інформації.

Варіанти:

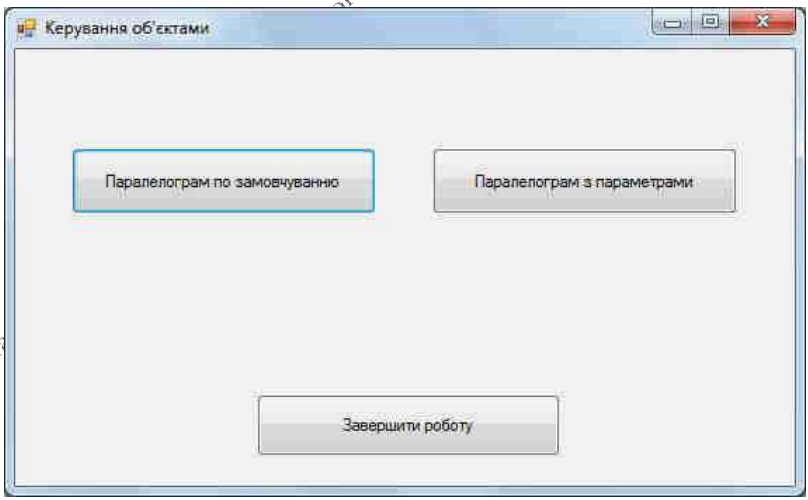
№ варіанту	Базовий клас
1.	Літак
2.	Риба
3.	Банківська картка
4.	Телефон
5.	Лист
6.	Автомобіль
7.	Верхній одяг
8.	Продукт
9.	Взуття
10.	Цукерки
11.	Тварина
12.	Книга
13.	Періодичне видання
14.	Миючий засіб
15.	Годинник

Наприклад, опис базового класу *Parallelogram* може виглядати так:

```
class Parallelogram
{   double a, b, alfa;

    public Parallelogram()
    { }

    public Parallelogram(double a, double b, double alfa)
    { this.a = a; this.b = b; this.alfa = alfa; }

    public double area()
    { return a * b * Math.Sin(alfa / 180 * Math.PI); }

    public double perimeter()
    { return 2 * (a + b); }

    public void Info()
    { MessageBox.Show("Дані паралелограма:\nдві сторони по " + a.ToString() + " та дві по " + b.ToString
                      " од.;\nплоща: " + area().ToString() + " кв. од.;\nпериметр: " + perimeter().ToS
                      " од.;\nдва кути по " + alfa.ToString() + " і два кути по " + (180 - alfa).ToStr
                      "Інформація", MessageBoxButtons.OK, MessageBoxIcon.Information);
    }
}
```

2. Створіть у формі дві кнопки для створення об'єктів описаного класу, викликаючи, відповідно, конструктор по замовчуванню і конструктор з параметрами та метод *Info* створених об'єктів. В нижній частині форми створіть кнопку для завершення роботи та забезпечте її функціонування.

Наприклад, для створення об'єктів класу паралелограма доцільно створити таку головну кнопкову форму:



Тоді процедура обробки натиснення першої кнопки може виглядати так:

```
private void button1_Click(object sender, EventArgs e)
 {Parallelogram p1 = new Parallelogram();
```

```
   p1.Info();
  }     .
```

А другої кнопки, відповідно, так:

```
private void button2_Click(object sender, EventArgs e)
 {double a, b, alfa;
  a=Convert.ToDouble(Interaction.InputBox("Введіть першу сторону паралелограма",
                     "Введення","4"));
  b=Convert.ToDouble(Interaction.InputBox("Введіть другу сторону паралелограма",
                     "Введення", "5"));
  alfa=Convert.ToDouble(Interaction.InputBox("Введіть гострий кут між сторонами "+
                        "паралелограма в градусах", "Введення", "30"));
  Parallelogram p2 = new Parallelogram(a, b, alfa);
  p2.Info();
 }
```

Лабораторна робота № 3

Тема. Реалізація наслідування елементарних класів.

Мета. Формування вмінь і навиків проектування елементарних класів з використанням наслідування. Закріплення вмінь і навиків використання функцій вводу-виводу. Застосування вмінь і навиків програмування алгоритмів лінійної, розгалуженої та циклічної структури.

Контрольні запитання

1. Для чого використовується наслідування класів?
2. Як реалізується наслідування класів? Від скількох класів можливе наслідування в C#?
3. Як з методу класу-нащадка викликати метод батька?
4. Як з методу класу-батька викликати метод нащадка?
5. У чому різниця між статичними і нестатичними класами?

Завдання.

1. **Створіть додаток-форму Windows. Опишіть чи скопіюйте з попередньої лабораторної в модуль форми *під класом форми* базовий клас згідно варіанту, передбачивши в ньому не менше трьох полів, двох конструкторів різних видів, двох методів для виконання обчислень. Передбачте в цьому класі змінну *count* для підрахунку кількості відповідних створених об'єктів та *nomer* – для зберігання номера поточного об'єкта. Створіть також у цьому класі метод *Info* для виводу інформації про об'єкт. *Забезпечте виклик цього методу з усіх конструкторів*. Реалізуйте деструктор класу для виведення номера знищуваного об'єкта та зменшення їх загальної кількості.**
2. **Опишіть *під базовим класом* два породжених класи від базового класу згідно варіанту і створіть у кожному з них не менше двох конструкторів різних видів. Доповніть їх додатковими полями та методами. Створіть також у цих класах методи *Info* для виводу інформації, але *не викликайте його з конструкторів*.**

Варіанти:

№ варіанту	Базовий клас	Породжений клас № 1	Породжений клас № 2
1.	Літак	Військовий літак	Цивільний літак
2.	Риба	Океанічна риба	Прісноводна риба
3.	Банківська картка	Картка для виплат	Картка універсальна
4.	Телефон	Стаціонарний телефон	Мобільний телефон
5.	Лист	Поштовий лист	Електронний лист
6.	Автомобіль	Вантажний автомобіль	Легковий автомобіль
7.	Верхній одяг	Куртка	Пальто
8.	Продукт	Овоч	Фрукт
9.	Взуття	Чоботи	Туфлі
10.	Цукерки	Шоколадні цукерки	Карамельні цукерки
11.	Тварина	Свійська тварина	Хижа тварина
12.	Книга	Паперова книга	Електронна книга
13.	Періодичне видання	Газета	Журнал
14.	Миючий засіб	Пральний порошок	Мило
15.	Годинник	Механічний годинник	Електронний годинник

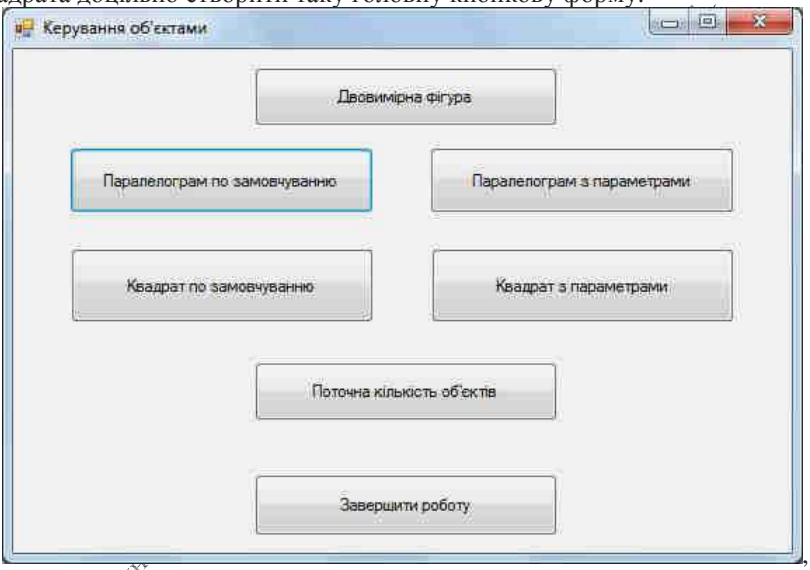
Наприклад, опис базового класу *Figure2D* та породжених від нього класів *Parallelogram* та *Square* може виглядати так:

```
class Figure2D
 {public static int count=0;
  int nomer;

  public Figure2D()
   {Info();
    count++;
    nomer = count;
   }

  ~Figure2D()
   {MessageBox.Show("Знищується фігура № " + nomer.ToString(), "Увага!",
                    MessageBoxButtons.OK, MessageBoxIcon.Exclamation);
    count--;
   }

  public double area()
   {return 0; }
  public double perimeter()
   {return 0; }
  public void Info()
   {MessageBox.Show("Дані абстрактної двовимірної фігури", "Інформація",
                    MessageBoxButtons.OK, MessageBoxIcon.Information);
   }
}
```

```
class Parallelogram : Figure2D
 {double a, b, alfa;

  public Parallelogram()
   { }

  public Parallelogram(double sa, double sb, double salfa): base()
   {a = sa; b = sb; alfa = salfa; }

  public double area()
   {return a * b * Math.Sin(alfa / 180 * Math.PI); }

  public double perimeter()
   {return 2 * (a + b); }

  public void Info()
   {MessageBox.Show("Дані паралелограма:\nдві сторони по " + a.ToString() + " та дві по " + b.ToString(
                    " од.;\nплоща: " + area().ToString() + " кв. од.;\nпериметр: " + perimeter().ToStri
                    " од.;\nдва кути по " + alfa.ToString() + " і два кути по " + (180 - alfa).ToString
                    " градусів", "Інформація", MessageBoxButtons.OK, MessageBoxIcon.Information);
   }
 }
class Square : Figure2D
 {double a;

  public Square()
   { }

  public Square(double sa): base()
   {a = sa; }

  public double area()
   {return a * a; }

  public double perimeter()
   {return 4 * a; }

  public void Info()
   {MessageBox.Show("Дані квадрата:\nчотири сторони по " + a.ToString() + " од.;\nплоща: " + Math.Sqrt(
                    " од.;\nплоща " + area().ToString() + " кв. од.;\nпериметр " + perimeter().ToString
                    " од.;\nчотири кути по 90 градусів", "Інформація", MessageBoxButtons.OK, MessageBox
   }
 }
```

3. Розробіть у формі кнопку для генерації об'єкта базового класу, по дві кнопки для створення об´єктів кожного з породжених описаних класів (викликаючи, відповідно, конструктор по замовчуванню і конструктор з параметрами та метод *Info* створених об'єктів). В нижній частині форми створіть кнопку для виведення поточної кількості створених об'єктів та кнопку для завершення роботи і забезпечте її функціонування. Завантажте додаток на виконання та переконайтеся у дієздатності кнопок. Чому спочатку виводиться інформація про базовий об'єкт по замовчуванню, а вже потім – інформація про створений об'єкт? Як забезпечити вивід інформації лише про створюваний об'єкт?

Наприклад, для керування об'єктами класів двовимірної фігури, паралелограма та квадрата доцільно створити таку головну кнопкову форму:



Тоді процедура обробки події натиснення, наприклад, кнопки *Паралелограм з параметрами* буде такою:

```
private void button7_Click(object sender, EventArgs e)
 {double a, b, alfa;
  a=Convert.ToDouble(Interaction.InputBox("Введіть першу сторону паралелограма",
                     "Введення","4"));
  b=Convert.ToDouble(Interaction.InputBox("Введіть другу сторону паралелограма",
                     "Введення", "5"));
  alfa=Convert.ToDouble(Interaction.InputBox("Введіть гострий кут між сторонами "+
                        "паралелограма в градусах", "Введення", "30"));
  Parallelogram p2 = new Parallelogram(a, b, alfa);
  p2.Info();
 }
```

Процедура ж обробки події натиснення кнопки *Поточна кількість об'єктів* буде така:

```
private void button7_Click(object sender, EventArgs e)
```

```
{MessageBox.Show("Поточна кількість об'єктів: " + Figure2D.count.ToString(),
                 "Увага!", MessageBoxButtons.OK, MessageBoxIcon.Exclamation);
}
```

4. **Перед описом методу *Info* в базовому класі встановіть модифікатор *virtual*, а в породжених – *override*. Що змінилося при створенні об'єктів за допомогою кнопок? Чому? Звідки взялися значення полів перед викликом кожного методу *Info*?**

Лабораторна робота № 4

Тема. Створення об’єктів класів. Використання абстрактних класів для обробки споріднених об’єктів.

Мета. Формування вмінь і навиків створення об’єктів та абстрактних класів засобами C#. Закріплення вмінь і навиків наслідування класів, використання об'єктів, підпрограм, елементів керування. Застосування вмінь і навиків програмування алгоритмів лінійної, розгалуженої та циклічної структури.

Контрольні запитання.

1. Як створити об’єкт класу? Як виконати однакову операцію над всіма об’єктами-елементами керування форми?
2. Для чого призначені абстрактні класи?
3. Як описуються абстрактні методи та класи? Чи може абстрактний метод міститися в неабстрактному класі? Чому?
4. Яке зв’язування по замовчуванню використовується для абстрактних методів? Якому модифікатору це відповідає?
5. З яким модифікатором мають перевизначатися абстрактні методи в породжених класах? Чи обов’язково це робити?
6. Чим абстрактні методи відрізняються від віртуальних?
7. Який механізм виконання абстрактних методів?

Завдання.

1. **Створіть у формі попередньої лабораторної роботи нову кнопку, яка буде генерувати аналогічну кнопку та виведіть її у формі випадковим чином.** Процедура обробки натиснення цієї кнопки може бути така:

```
private void button8_Click(object sender, EventArgs e)
{   Button a = new Button();
    Random rnd = new Random();
    a.Width=200;
    a.Height=50;
    a.Left=rnd.Next(500);
    a.Top=rnd.Next(400);
    a.Text = ((Button)sender).Text;
    a.Parent = this;
    a.Show();
    a.Click += new System.EventHandler(this.button8_Click);
}
```

2. **Створіть у формі ще одну кнопку, яка буде переміщувати вліво всі її елементи керування.** Процедура обробки натиснення цієї кнопки може виглядати так:

```
private void button9_Click(object sender, EventArgs e)
 {foreach (Control elem in this.Controls)
   elem.Left += 60;
 }
```

3. **Відкрийте розроблений раніше додаток-форму з описом базового та породжених класів згідно варіанту. *Під класом форми* перетворіть у ньому базовий клас в абстрактний з абстрактними та віртуальними методами, *не знищуючи попередньо розроблені класи*. Забезпечте дієздатність породжених класів та модифікованої форми.**

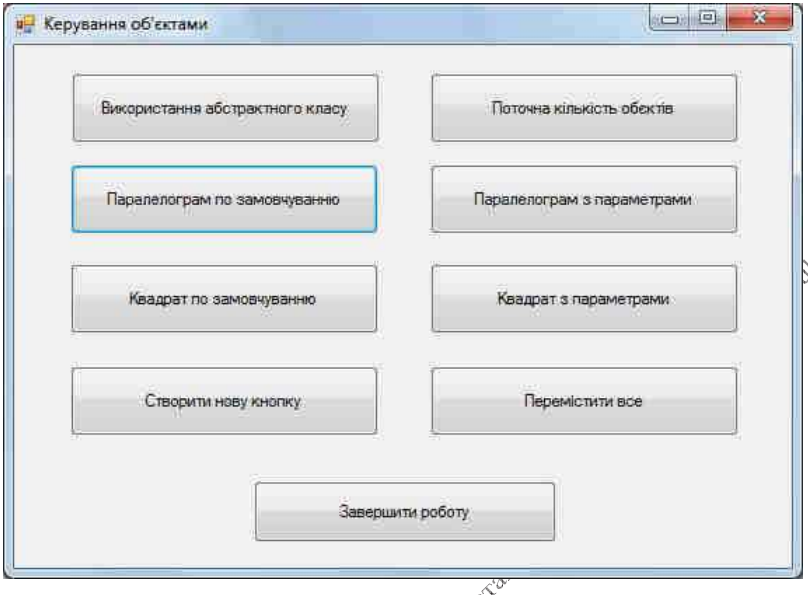
Наприклад, для геометричних фігур базовий абстрактний клас двовимірної фігури може бути таким:

```
abstract class Figure2D
 { public Figure2D()
    { Info(); }
   public abstract double area();
   public abstract double perimeter();
   public virtual void Info()
    { MessageBox.Show("Дані абстрактної двовимірної фігури", "Інформація",
                      MessageBoxButtons.OK, MessageBoxIcon.Information);
    }

 }
```

Тоді в породжених класах перед всіма перевизначеними методами потрібно вказати модифікатор `override`.

4. **Змініть надпис кнопки використання батьківського класу на *Використання абстрактного класу*. В процедурі обробки події натиснення цієї кнопки знищіть попередній код та забезпечте:**
   - 4.1. **Створення двох об'єктів різних породжених класів;**
   - 4.2. **Створення вказівки на об'єкт батьківського класу;**
   - 4.3. **Почергове занесення у вказівку на об'єкт батьківського класу вказівок на об'єкти породжених класів та вивід інформації про них.**

Наприклад, у формі для керування об'єктами паралелограма та квадрата доцільно так доопрацювати головну кнопкову форму:



Тоді процедура обробки натиснення кнопки використання абстрактного класу може бути, наприклад, такою:

```
private void button3_Click(object sender, EventArgs e)
{Parallelogram p1 = new Parallelogram(5, 7, 60);
 Square d1 = new Square(8);
 Figure2D f;
 f = p1;
 f.Info();
 f = d1;
 f.Info();
}
```

Чому для вказівки об'єкта батьківського класу виводиться інформація про нащадків? Яке зв'язування методів тут використовується?

5. **Самостійно створіть у формі ще дві кнопки. Перша з них має виконувати довільні інші дії над всіма елементами керування (але не переміщувати назад), а друга – по іншому використовувати абстрактні класи.**
6. **Додатково 2 бали до рейтингу. Самостійно створіть форму для демонстрації функціонування двох об'єктів (по одному з кожного з породжених класів). Створіть у цій формі також ще одну кнопку для випадкового вибору одного з цих об'єктів і виводу інформації про нього.**

Лабораторна робота № 5

Тема. Розробка та використання властивостей і індексаторів.

Мета. Формування вмінь і навиків створення та використання властивостей і індексаторів об'єктів. Закріплення вмінь і навиків використання класів, об'єктів, підпрограм, функцій вводу-виводу та перетворення типів. Застосування вмінь і навиків програмування алгоритмів лінійної, розгалуженої та циклічної структури.

Контрольні запитання.

1. Чим поля класу відрізняються від його методів?
2. Що таке властивості об'єкта? Чим властивості об'єкта відрізняються від полів? Чому для властивості зазначають тип даних?
3. Чому доступ до даних об'єкта рекомендується виконувати через його властивості?
4. Як заборонити зчитування/запис властивості об'єкта?
5. Що таке індексатори? Що спільного між властивостями та індексаторами?

Завдання.

1. **Відкрийте розроблений раніше додаток з описом базового та породжених класів згідно варіанту. Відмініть виклик методу інформування з конструктора базового класу. Доповніть один з розроблених породжених класів властивостями для коректного зчитування/запису значень його полів.**
2. **Створіть ще один породжений клас від класу з властивостями. Використайте ці властивості в породжених класах для виводу інформації про об'єкти та виконання обчислень.**
3. **Модифікуйте форму, доповнивши її кнопками для *супроводження життєвого циклу (виконання дій)* над *двома* об'єктами нового класу.**

Варіанти:

№ варіанту	Базовий клас	Породжений клас № 1	Породжений клас № 2
1.	Літак	Військовий літак	Цивільний літак
2.	Риба	Океанічна риба	Прісноводна риба
3.	Банківська картка	Картка для виплат	Картка універсальна
4.	Телефон	Стаціонарний телефон	Мобільний телефон
5.	Лист	Поштовий лист	Електронний лист
6.	Автомобіль	Вантажний автомобіль	Легковий автомобіль
7.	Верхній одяг	Куртка	Пальто
8.	Продукт	Овоч	Фрукт
9.	Взуття	Чоботи	Туфлі
10.	Цукерки	Шоколадні цукерки	Карамельні цукерки
11.	Тварина	Свійська тварина	Хижа тварина
12.	Книга	Паперова книга	Електронна книга
13.	Періодичне видання	Газета	Журнал
14.	Миючий засіб	Пральний порошок	Мило
15.	Годинник	Механічний годинник	Електронний годинник

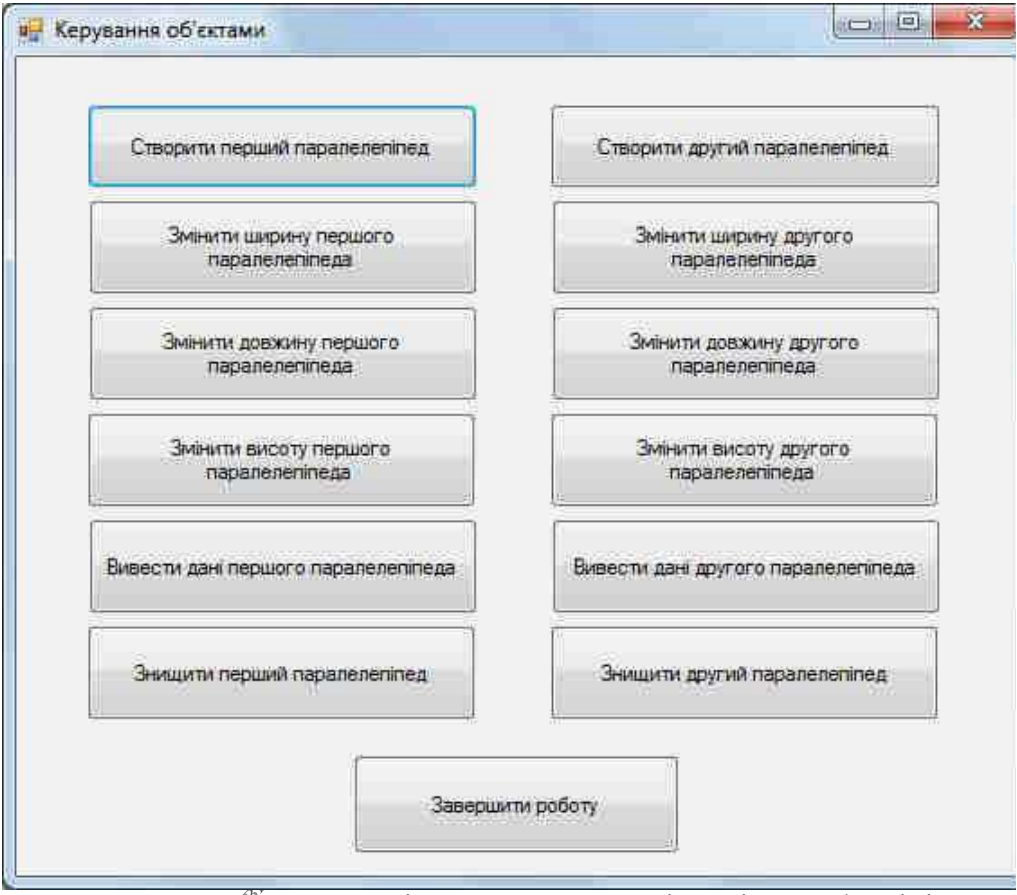
Наприклад, опис класу з властивостями модифікованого паралелограма *Parallelogram* та породженого від нього класу **прямокутного** паралелепіпеда *Parallelepiped* може виглядати так:

```
class Parallelogram:Figure2D
{
    private double a, b, alfa;

    public double A
    {get { return a; }
     set { if (value >= 0)
            a = value;}}

    public double B
    {get { return b; }
     set { if (value >= 0)
            b = value;}}

    public double Alfa
    {get { return alfa; }
     set { if (value >= 0)
            alfa = value;}}
```

```
    public Parallelogram(double a, double b, double alfa)
    {
        A = a; B = b; Alfa = alfa;
    }

    public override double area()
    {
        return a * b * Math.Sin(alfa / 180 * Math.PI);
    }

    public override double perimeter()
    {
        return 2 * (a + b);
    }

    public override void Info()
    {
        MessageBox.Show("Дані паралелограма:\nдві сторони по "+a.ToString()+
                        " та дві по "+b.ToString()+" од.;\nплоща: " + area().ToS
                        " кв. од.;\nпериметр: " + perimeter().ToString() +
                        " од.;\nдва кути по " + alfa.ToString() + " і два кути п
                        (180 - alfa).ToString() + " градусів","Інформація",
                        MessageBoxButtons.OK,MessageBoxIcon.Information);
    }
}
```

```
class Parallelepiped : Parallelogram
{
    private double h;

    public double H
    {get { return h; }
     set { if (value >= 0)
            h = value;}}

    public Parallelepiped(double a,double b,double h)
        : base(a, b, 90)
    { H = h;
    }

    new public double area()
    { return 2*(A * B + A * H + B * H);
    }

    public double volume()
    { return A * B * H;
    }

    public override void Info()
    { MessageBox.Show("Дані паралелепіпеда:\nсторони по " + A.ToString() + ", " +
                      B.ToString()+", "+H.ToString()+" од.;\nплоща: " + area().To
                      " кв. од.\nоб'єм: " + volume().ToString() + " куб. од.", "I
                      MessageBoxButtons.OK, MessageBoxIcon.Information);
    }
}
```

А головна кнопкова форма набуде такого вигляду:



Для керування двома паралелепіпедами створимо дві вказівки в формі і допоміжну процедуру:

```
public partial class Form1 : Form
{
    Parallelepiped p1 = null, p2 = null;

    public Form1()
    {
        InitializeComponent();
    }

    static bool inputDouble(ref double x, string Povidom)
    {
        string S;
        S = x.ToString();
    Povtor:
        S = Interaction.InputBox(Povidom, "Введення", S);
        try
        {
            x = Convert.ToDouble(S);
        }
        catch (System.FormatException)
        {
            if (MessageBox.Show("Ви ввели не число\n\nБажаєте повторити?",
                "Увага", MessageBoxButtons.YesNo, MessageBoxIcon.Warning) == DialogResult.Ye
                goto Povtor;
            else
                return false;
        }
        return true;
    }
...
}
```

Нагадаємо, що, наприклад, процедура обробки події натиснення кнопки *Створити перший паралелепіпед* буде приблизно такою:

```
private void button1_Click(object sender, EventArgs e)
 {   double a=4, b=5, h=8;
     if (!inputDouble(ref a, "Введіть довжину першого паралелепіпеда"))
      return;
     if (!inputDouble(ref b, "Введіть ширину першого паралелепіпеда"))
         return;
     if (!inputDouble(ref h, "Введіть висоту першого паралелепіпеда"))
         return;
     p1 = new Parallelepiped(a, b, h);
 }        .
```

А процедура обробки події натиснення кнопки *Змінити висоту першого паралелепіпеда* може бути такою:

```
private void button9_Click(object sender, EventArgs e)
 { if (p1==null)
    {MessageBox.Show("Перший паралелепіпед ще не створено!", "Увага!",
                     MessageBoxButtons.OK, MessageBoxIcon.Warning);
     return;
    }
```

```
        double h=p1.H;
        if (inputDouble(ref h, "Змініть висоту першого паралелепіпеда"))
            p1.H = h;
    }
```

4. Створіть новий клас чи використайте існуючий і опишіть у ньому індексатор. Продемонструйте його використання.

Наприклад, для розрахунку номера вкладеного прямокутного паралелепіпеда, об'єм якого не перевищує задане число, доцільно створити такий клас і його індексатор:

```
class EmbeddedParallelepiped : Parallelepiped
 {public EmbeddedParallelepiped(double a, double b, double h) : base(a, b, h)
   { }

  public Parallelepiped this[int i]
     {get {if (i <= 0) return new Parallelepiped(A, B, H);
           else return new Parallelepiped(A / Math.Pow(2, i), B / Math.Pow(2, i),
                                          H / Math.Pow(2, i)); }
     }
 }
```

Тоді використання об'єкту цього класу при натисненні відповідної кнопки (справа знизу на зразку) буде таким:

```
  private void button4_Click(object sender, EventArgs e)
    {var vp=new EmbeddedParallelepiped(5,7,6);
     int i=1;
     double VMax = 17;
     if (!inputDouble(ref VMax, "Введіть обмеження об'єму")) return;
     while (vp[i].volume() > VMax) i++;
     MessageBox.Show("Вкладений паралелепіпед № " + i.ToString() + " має об'єм " +
      vp[i].volume() + " куб. од.", "Інформація", MessageBoxButtons.OK,
      MessageBoxIcon.Information);
    }
```

Лабораторна робота № 6

Тема. Перевантаження методів. Перевантаження унарних та бінарних операторів.

Мета. Формування вмінь і навиків перевантаження операцій над об'єктами. Закріплення вмінь і навиків використання класів, об'єктів, підпрограм, функцій вводу-виводу та перетворення типів. Застосування вмінь і навиків програмування алгоритмів лінійної, розгалуженої та циклічної структури.

Контрольні запитання.

1. Навіщо виконують перевантаження операцій?
2. Чому перевантаження операцій завжди статичне?
3. Які операції можливо перевантажити в C#?
4. Як при порівнянні об'єктів забезпечити перевірку рівності їх полів, а не вказівок?
5. Для яких операцій застосовується перевантаження ++?

Завдання.

1. **Відкрийте розроблений раніше додаток-форму з описом базового та породжених класів згідно варіанту. Перевантажте в одному з ваших класів хоча б один унарний оператор.**
2. **Перевантажте в одному з ваших класів оператори перевірки об'єктів на рівність та нерівність.**
3. **Розробіть у формі кнопки для демонстрації дій перевантажених операцій та використайте метод *Info()* для відображення стану об'єктів до і після використання цих операцій.**

Наприклад, перевантаження операцій рівності, нерівності та інкрементації для класу прямокутного паралелепіпеда *Parallelepiped* може виглядати так:

```
class Parallelepiped : Parallelogram
{
    public double minSide()
    {return Math.Min(Math.Min(A, B), H);
    }

    public double maxSide()
    {double max = A;
     if (B > max) max = B;
     if (H > max) max = H;
     return max;
    }

    new public double perimeter()
    {return 4*(A+B+H);
    }

    public static bool operator ==(Parallelepiped p1, Parallelepiped p2)
    {return p1.minSide() == p2.minSide() &&
            p1.maxSide() == p2.maxSide() &&
            p1.perimeter() == p2.perimeter();
    }

    public static bool operator !=(Parallelepiped p1, Parallelepiped p2)
    {return !(p1 == p2);
    }

    public static Parallelepiped operator ++(Parallelepiped p)
    {return new Parallelepiped(p.A + 1, p.B + 1, p.H + 1);
    }
}
```

Тоді процедура обробки події натиснення кнопки *Інкрементації паралелепіпеда* буде така:

```
private void button8_Click(object sender, EventArgs e)
 {var p = new Parallelepiped(5, 4, 8); p.Info();
  p++; p.Info();
  ++p; p.Info();
 }
```

А процедура обробки події натиснення кнопки *Порівняння паралелепіпедів* буде, наприклад, такою:

```
private void button7_Click(object sender, EventArgs e)
 {   var p1 = new Parallelepiped(4, 5, 8);
     p1.Info();
     var p2 = new Parallelepiped(5, 4, 10);
     p2.Info();
     if (p1 == p2) MessageBox.Show("Паралелепіпеди рівні");
     else MessageBox.Show("Паралелепіпеди не рівні");
 }
```

```
    class Parallelepiped : Parallelogram
    {…
        public double minSide()
        {return Math.Min(Math.Min(A, B), H);
        }

        public double maxSide()
        {double max = A;
         if (B > max) max = B;
         if (H > max) max = H;
         return max;
        }

        new public double perimeter()
        {return 4*(A+B+H);
        }

        public static bool operator ==(Parallelepiped p1,
Parallelepiped p2)
        {return p1.minSide() == p2.minSide() &&
                p1.maxSide() == p2.maxSide() &&
                p1.perimeter() == p2.perimeter();
        }

        public static bool operator !=(Parallelepiped p1,
Parallelepiped p2)
        {return !(p1 == p2);
        }

        public static Parallelepiped operator ++(Parallelepiped p)
        {return new Parallelepiped(p.A + 1, p.B + 1, p.H + 1);
        }
    }
```

4. Створіть у власному класі два методи *Info()*. Перший – для виводу інформації про об'єкт по замовчуванню, другий – для виводу інформації у вікні з вказаним заголовком. Реалізуйте перший з цих перевантажених методів за допомогою другого. Використайте ці методи для виводу інформації про різні об'єкти.

Лабораторна робота № 7

Тема. Використання інкапсуляції для організації взаємодії вкладених об'єктів.

Мета. Формування вмінь і навиків програмування алгоритмів обробки подій елементів керування у формах. Закріплення вмінь і навиків створення форм та елементів керування. Застосування вмінь і навиків програмування алгоритмів лінійної, розгалуженої та циклічної структури.

Завдання.

1. **Створіть форму-калькулятор, подібну до зразка, наведеного нижче, з власним оригінальним дизайном та забезпечте її функціональність. Обов'язково відобразіть у формі вміст комірки пам'яті. У нижній частині форми виведіть інформацію про розробника.** Для цього:

1.1. Створіть новий додаток-форму. Дайте формі ім'я **FormCalc** (властивість **(Neme)**), введіть заголовок, пов'язаний з вашим прізвищем (властивість **Text**) на забезпечте для неї вивід по центру екрана (властивість **StartPosition**);

1.2. Натягніть у формі поля, надпис та кнопки, керуючись зразком, але на власний розсуд;

1.3. Дайте полю для відображення введення і результатів ім'я **textTablo**, а полю вмісту пам'яті – **textMemory**. Забороніть до них доступ (властивість **Enabled**) та задайте вирівнювання по правому краю (властивість **TextAlign**);

1.4. При роботі з пам'яттю в калькуляторі має виконувати попередня операція, а після відображення результатів наступне число має вводитися спочатку, тому створемо **у формі** дві локальні змінні, які будуть відповідати за ці стани:

```
public partial class FormCalc : Form
{
    bool resultTablo = true;
    string operation="";
```

1.5. Оскільки кнопки з цифрами мають додавати відповідні цифри на табло, то для зменшення розміру коду та кількості помилок створимо загальну процедуру додавання цифри на табло:

```
void plusTablo(char symbol)
{   if (resultTablo)
        textTablo.Text = "";
    if (textTablo.Text == "0")
         textTablo.Text = "";
    textTablo.Text += symbol.ToString();
    resultTablo = false;
}
```

тоді для введення, наприклад, цифри 9 в процедурі обробки натиснення відповідної кнопки достатньо вказати

```
private void button4_Click(object sender, EventArgs e)
 {plusTablo('9');
 }
```

Самостійно забезпечте функціональність решти кнопок з цифрами.

1.6. Кома між цілою і дробовою частиною може вводитися лише тоді, коли її ще немає. Тому процедура обробки натиснення цієї кнопки може бути така:

```
private void buttonKoma_Click(object sender, EventArgs e)
{   if (resultTablo)
        textTablo.Text = "0";
    bool available = false;
    int i, len;
    len = textTablo.Text.Length;
    for (i = 0; i < len; i++)
        if (textTablo.Text[i] == ',')
        {
            available = true;
            break;
        }
    if (!available)
        textTablo.Text += ",";
    resultTablo = false;
}
```

1.7. **BackSpace** має витирати останній символ, а це можна реалізувати вирізкою підрядка з початку табло:

```
private void button5_Click(object sender, EventArgs e)
{   if (resultTablo)
        textTablo.Text = "0";
    else
        textTablo.Text = textTablo.Text.Substring(0, textTablo.Text.Leng
    if (textTablo.Text == "")
        textTablo.Text = "0";
    resultTablo = false;
}
```

1.8. Створимо тепер процедуру для виконання обчислень безпосередньо над табло, при цьому саму дію передамо параметром підпрограми:

```
private void runOperationTablo(string opr)
{   double tablo;
    if (opr == "")
        return;
    try
    {tablo = Convert.ToDouble(textTablo.Text);
    }
    catch (System.FormatException)
    {   MessageBox.Show("Операцію виконати неможливо", "Увага", MessageBoxButtons.OK, Messag
        textTablo.Text = "0";
        return;
    }
    switch (opr)
    {   case "Sqrt":
            if (tablo < 0)
            {MessageBox.Show("Операція неможлива: корінь з від'ємного числа", "Увага", Messa
             return;
            }
            tablo = Math.Sqrt(tablo);
            break;
        case "%":
            tablo *= 0.01;
            break;
        case "1/x":
            if (tablo == 0)
            {MessageBox.Show("Операція неможлива: ділення на нуль", "Увага", MessageBoxButto
             return;
            }
            tablo = 1 / tablo;
            break;
        case "+/-":
            tablo *= -1;
            break;
    }
    textTablo.Text = tablo.ToString();
    resultTablo = true;
}
```

Тоді для виконання обчисль, пов'язаних лише з табло, достатньо викликати цю процедуру з назвою відповідної дії. Наприклад, процедура обробки натиснення кореня може бутии такою:

```
private void button7_Click(object sender, EventArgs e)
 {runOperationTablo("Sqrt");
 }
```

1.9. І, нарешті, створимо процедуру для виконання обчислень над табло та пам'яттю. При цьому сама дія не передається в процедуру, а виконується попередня збережена операція:

```
private void runOperationMemory()
{   double tablo, memory;
    if (operation == "")
        return;
    try
    {   tablo = Convert.ToDouble(textTablo.Text);
        memory = Convert.ToDouble(textMemory.Text);
    }
    catch (System.FormatException)
    {   MessageBox.Show("Операцію виконати неможливо", "Увага", MessageBoxButtons.OK, Messa
        textTablo.Text = textMemory.Text = "0";
        return;
    }
    switch (operation)
    {       case "+":
            tablo += memory;
            memory = tablo;
            break;
        case "-":
            tablo = memory - tablo;
            memory = tablo;
            break;
        case "*":
            tablo *= memory;
            memory = tablo;
            break;
        case "/":
            if (tablo == 0)
            {   MessageBox.Show("Операція неможлива: ділення на нуль", "Увага", MessageBox
                return;
            }
            tablo = memory / tablo;
            memory = tablo;
            break;
        case "M+":
            memory += tablo;
            break;
```

```
            case "M-":
                memory -= tablo;
                break;
            case "MC":
                memory = 0;
                break;
            case "MR":
                tablo = memory;
                break;
            case "MT":
                memory = tablo;
                break;
            case "C":
                tablo = 0;
                memory = 0;
                break;
        }
        operation = "";
        textTablo.Text = tablo.ToString();
        textMemory.Text = memory.ToString();
        resultTablo = true;
    }
```

Тоді при натисненні, наприклад, кнопки додавання потрібно спочатку **виконати попередню операцію**, потім **результат зчитати з пам'яті в табло**, а **+** запам'ятати для наступного виконання:

```
private void buttonPlus_Click(object sender, EventArgs e)
{   runOperationMemory();
    operation = "MT";
    runOperationMemory();
    operation = "+";
}
```

1.10. Функціональність решти кнопок забезпечте замостійно

2. **Доповніть форму-калькулятор двома функціональними кнопками, відмінними від кнопок одногрупників, для виконання двох корисних функцій (наприклад, піднесення до степеня) та забезпечте їх дієздатність.**

Лабораторна робота № 8

Тема. Використання змінних методів та об'єктів класів. Статичні змінні, методи та класи.

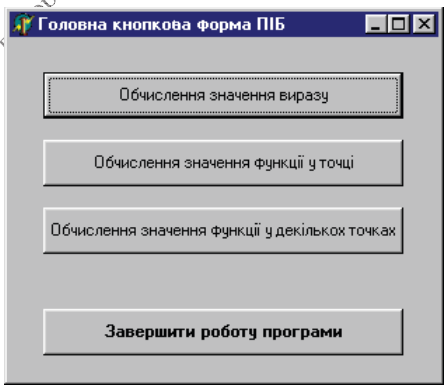
Мета. Формування вмінь і навиків створення та використання методів класів, форм і їх компонентів. Закріплення вмінь і навиків використання функцій вводу-виводу. Застосування вмінь і навиків програмування алгоритмів лінійної, розгалуженої та циклічної структури.

Контрольні запитання

1. Як створити додаток з інтерактивною формою?
2. З чого складається і де відображається будова проекту з візуальними формами? Як перейти до тексту програми візуальної форми?
3. У файлах з якими назвами та розширеннями зберігаються описи класів та параметри зовнішнього вигляду форм?
4. Для чого використовується і як активізувати оглядач рішень?
5. Як створити візуальні компоненти у формах? Де зберігаються параметри їх зовнішнього вигляду?
6. Як описати або створити автоматично, використати та знищити процедури обробки подій форми та її компонентів?
7. Де описуються методи класів? Як ці методи викликаються з процедур обробки подій?

Завдання

1. Створити програму з формою за зразком. В процедурі обробки події натиснення четвертої кнопки для забезпечення завершення роботи програми вказати *Close();*. В класі форми описати методи для коректного вводу дійсних та цілих чисел.



Опис методу для коректного вводу дійсних чисел може бути, наприклад, таким:

```
public partial class Form1 : Form
{
    public Form1()...

    static bool InputDouble(ref double x, string povidom)
     {  string s;
        s = x.ToString();
    Povtor:
        s = Interaction.InputBox(povidom, "Введення", s);
        try
         {x = Convert.ToDouble(s);
         }
        catch (System.FormatException)
         {if (MessageBox.Show("Ви ввели не число.\n\nБажаєте повторити?", "Увага",
                              MessageBoxButtons.YesNo, MessageBoxIcon.Warning) == DialogResult.Yes
           goto Povtor;
          else
           return false;
         }
        return true;
     }
    ...
}
```

2. Скласти програму, що викликається при натисненні першої кнопки з головної кнопкової форми, для обчислення значень виразу.

Вимоги до програми:

- числові значення параметрів обчислень ввести з клавіатури, константи з умов завдання використати як значення по замовчуванню;
- початкові дані та результати обчислень вивести в одному діалоговому вікні.

Варіанти.

1. $\dfrac{zy+x}{2z-x}$ при x=0.137, y=1.27, z=4.7561;
2. $\dfrac{x(2y+3z)}{y-xz}$ при x=0.0399, y=4.83, z=0.072;
3. $\dfrac{7x+8y}{x-4zy}$ при x=1.576, y=1.786, z=1.1236.
4. $\dfrac{z+y}{2z-x}$ при x=0.137, y=1.27, z=4.7561;
5. $\dfrac{x(y+z)}{y-yz}$ при x=0.0399, y=4.83, z=0.072;

6. $\frac{x+y}{4x-zy}$ при x=1.576, y=1.786, z=1.1236;

7. $\frac{xy-4z}{x+4y}$ при x=12.743, y=0.654, z=0.0208;

8. $\frac{z-y}{14x+yz}$ при x=3.49, y=0.456, z=0.0059;

9. $\frac{xz+3y}{(y-z)x}$ при x=0.0976, y=2.371, z=1.1587;

10. $\frac{(x-y)}{y+zx}$ при x=82.356, y=34.42, z=7.0046;

11. $\frac{y+z}{4y+2x}$ при x=0.11578, y=4.675, z=4.654;

12. $\frac{x-y}{xy+z}$ при x=3.7156, y=3.034, z=0.756;

13. $\frac{x+4y}{xy-z(x-y)}$ при x=7.654, y=0.876, z=0.0987;

14. $\frac{y+z}{(z-x)y}$ при x=0.036, y=3.987, z=4.654;

15. $\frac{2(x+y)}{z+y(x+y)}$ при x=0.327, y=0.0098, z=4.675.

Наприклад, процедура обробки події натиснення першої кнопки для варіанту № 9 може бути такою:

```
private void button1_Click(object sender, EventArgs e)
{   Double x, y, z, res;
    x = 0.0976;
    y = 2.371;
    z = 1.1587;
    if (!InputDouble(ref x, "Введіть значення x"))
        return;
    if (!InputDouble(ref y, "Введіть значення y"))
        return;
    if (!InputDouble(ref z, "Введіть значення z"))
        return;
    res = (x * z+3*y) / ((y-z)* x);
    MessageBox.Show("При x=" + x.ToString() + "\n          y=" + y.ToString() +"\n
                    z.ToString() + "\nобчислено значення виразу: " + res.ToString(
                    "Результат", MessageBoxButtons.OK, MessageBoxIcon.Information)
}
```

3. Скласти програму, що викликається при натисненні другої кнопки з головної кнопкової форми, для обчислення значення функції з використанням оператора розгалуження.

Вимоги до програм:

- числові значення параметра обчислень ввести з клавіатури;
- для обчислення значень залежної змінної використати оператор розгалуження;
- початкові дані та результати обчислень вивести в діалоговому вікні;
- забезпечити коректність функціонування програми на всіх вітках розгалуження.

Варіанти.

1. $$Y=\begin{cases}\sqrt{sin^3(x-1)} & -1{,}2\le x\le 1{,}2\\ e^{-x} & x\ge 6\\ 3{,}5\,ln|x| & 1{,}2>|x|>6\\ 1{,}5x & x\le -6\end{cases};$$

2. $$Y=\begin{cases}e^{-2{,}5x^3} & x<0\\ \dfrac{x-1}{x-\sin^2 x} & 0\le x<5\\ 2x & x>5\end{cases};$$

3. $Y=\begin{cases} x^e - e^{-x} & |x|<2 \\ ln\, x^2 & x\le -2; \\ sin^2 x & x\ge 2 \end{cases}$

4. $Y=\begin{cases} cos^3 x^2 - \sin x & x\ge 0 \\ x^2 - 0{,}83 & 0<x<3. \\ \dfrac{1{,}4+x}{ln\, x} & x\ge 3 \end{cases}$

5. $Y=\begin{cases} \sin^2 x & 0\le x<1 \\ \sqrt{\ln x - \sin x} & 1\le x\le 3 \\ x\ln^3 x & x>3, x<0 \end{cases}$;

6. $Y=\begin{cases} -arctg\left((x+\pi)/x^2\right) & 0<x\le 1.2 \\ \ln\sqrt{x^3} & 1.2<x<9 \\ e^{-x} & x\ge 9, x\le 0 \end{cases}$;

7. $Y=\begin{cases} 2x\sqrt[3]{x^2+5} & 1<x\le 12 \\ arctgx & 0<x\le 1 \\ e^{x+3} & x\le 0, x>12 \end{cases}$;

8. $Y=\begin{cases} \ln\left|\dfrac{\pi}{15}-x\right| & 0\le x<\dfrac{1}{4} \\ \left(x^2+2{,}04\right)^{3{,}14} & \dfrac{1}{4}\le x<1 \\ \arccos\dfrac{x}{4} & x\ge 1, x<0 \end{cases}$;

9. $Y=\begin{cases} 0 & x\le -1 \\ ctg\,\dfrac{x-1}{e} & -1<x\le 0; \\ ln\, x & x>0 \end{cases}$

10. $Y = \begin{cases} e^{-|x|} & x \ge 1 \\ ln\sqrt{1-x^2} & |x| < 1 \\ arctgx & x \le -1 \end{cases}$;

11. $Y = \begin{cases} 2^{x-1} + 3{,}5 & \pi \le x < 8{,}6 \\ \sqrt{|\pi - 3x|} & -8{,}6 < x < \pi \\ 2{,}7 & |x| \ge 8{,}6 \end{cases}$;

12. $Y = \begin{cases} 1{,}3^{\,2+x} * x^2 & |x| \ge 5 \\ ln^{x-1} & -1 \le x \le 1 \\ cos^{|x-1|} & 1 < |x| < 5 \end{cases}$;

13. $Y = \begin{cases} sin\left(-x^2+1\right) & x \le -3 \\ 2x + ln^2\ 4{,}4 & x \ge 0 \\ -e^{\frac{1}{x}} & -3 < x < 0 \end{cases}$;

14. $Y = \begin{cases} \sqrt[3]{ln\,x + ln\,x^2} & x > 1 \\ e^{-x} + 1 & 0 < x \le 1 \\ cos^3\,x & x \le 0 \end{cases}$;

15. $Y = \begin{cases} ln(\,x + 13{,}2\,) & 4 < |x| < 10 \\ sin\,e^x - 2 & |x| \le 4 \\ 1{,}5x & x < -4,\ x \ge 10 \end{cases}$.

Наприклад, процедура обробки події натиснення другої кнопки для варіанту № 2 може бути такою:

```
private void button2_Click(object sender, EventArgs e)
{Double x=7.4, y=0;
 if (!InputDouble(ref x, "Введіть значення x"))
     return;
 if (x < 0)
  y = Math.Exp(-2.5*Math.Pow(x,3));
 else
  if (x < 5)
     y = (x-1)/(x-Math.Pow(Math.Sin(x), 2));
  else
     y = 2*x;
 MessageBox.Show("При x=" + x.ToString() + "\nобчислено значення виразу: " +
                 "Результат",MessageBoxButtons.OK, MessageBoxIcon.Informatio
}
```

4. Скласти програму, що викликається при натисненні третьої кнопки з головної кнопкової форми, для обчислення значень функції, починаючи з заданої точки у вказаній кількості точок.

Вимоги до програми:

- числові значення параметрів обчислень ввести з клавіатури, константи з умов завдання використати як значення по замовчуванню;
- значення аргументів та відповідні їм значення результатів вивести в діалоговому вікні у вигляді таблиці.

Варіанти.

1. $Y = \dfrac{2x+8}{|\cos 2.5x|+1}$ при $0.1 \le x, \quad \Delta x = 0.3, \quad n = 6;$
2. $Y = \dfrac{x - \ln 2x}{2x+1}$ при $0.6 \le x, \quad \Delta x = 2.5, \quad n = 5;$
3. $Y = \dfrac{x + \tan 2x}{3x}$ при $0.8 \le x, \quad \Delta x = 0.2, \quad n = 6;$
4. $Y = \dfrac{(x+7)^2}{\sqrt{x^2+1}}$ при $6.5 \le x, \quad \Delta x = 0.3, \quad n = 9.$
5. $Y = \dfrac{\sin^2 x}{(x+1)^2}$ при $0.5 \le x, \quad \Delta x = 0.1, \quad n = 9;$
6. $Y = \dfrac{e^{2x} - 8}{x+3}$ при $1.5 \le x, \quad \Delta x = 0.3, \quad n = 6;$
7. $Y = \dfrac{x + \cos 2x}{3x}$ при $1.2 \le x, \quad \Delta x = 0.2, \quad n = 7;$
8. $Y = \dfrac{x + \sin 2xe^{2x} - 8}{x+2}$ при $0.6 \le x, \quad \Delta x = 1.5, \quad n = 6;$
9. $Y = \dfrac{\cos^3 x^2}{1.5x+2}$ при $0.8 \le x, \quad \Delta x = 0.3, \quad n = 5;$
10. $Y = \dfrac{x + \sin 2x}{x^2 - 3}$ при $3.1 \le x, \quad \Delta x = 0.8, \quad n = 6;$

11. $Y = \dfrac{x^2 + 2}{3\cos\sqrt{x} + 1}$ при
$0.3 \le x, \quad \Delta x = 0.8, \quad n = 7;$

12. $Y = \dfrac{x^3 - 3}{3\ln x}$ при
$2.5 \le x, \quad \Delta x = 1.5, \quad n = 5;$

13. $Y = \dfrac{(3x+2)^2}{\sin x + 3}$ при
$0.2 \le x, \quad \Delta x = 0.7, \quad n = 6;$

14. $Y = \dfrac{2\sin^3(x+2)}{x^2 + 1}$ при
$0.7 \le x, \quad \Delta x = 0.1, \quad n = 5;$

15. $Y = \dfrac{1.9x - \ln 3x}{3x + 1}$ при
$0.8 \le x, \quad \Delta x = 1.2, \quad n = 6;$

Наприклад, процедура обробки події натиснення третьої кнопки для варіанту № 14 може бути такою:

```
private void button3_Click(object sender, EventArgs e)
{ Double x, y, dx=0.1, xp=0.7;
  int i, n=5;
  if(!InputDouble(ref xp, "Введіть початок проміжку табулювання"))
    return;
  if(!InputDouble(ref dx, "Введіть крок"))
    return;
  if(!InputInt(ref n, "Введіть кількість точок"))
    return;
  string res = "Протабульована функція:\n   x\t  y";
  for (i = 0; i < n; i++)
   {x = xp + i * dx;
    y = 2*Math.Pow(Math.Sin(x+2),3)/(Math.Pow(x,2) + 1);
    res+= "\n" + x.ToString() + "\t" + y.ToString();
   }
  MessageBox.Show(res, "Результати табулювання",
                  MessageBoxButtons.OK, MessageBoxIcon.Information);
}
```

5. *Забезпечте встановлення попередніх параметрів обчислень при повторному натисненні трьох перших кнопок (додатково 2 бали до рейтингу).

Лабораторна робота № 9

Тема. Створення та використання вкладених об'єктів – елементів керування Windows-форм.

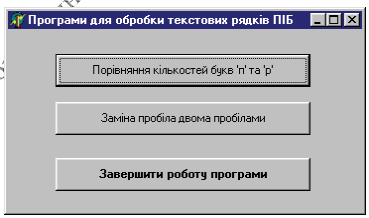
Мета. Формування вмінь і навиків програмування алгоритмів обробки текстових рядків. Закріплення вмінь і навиків використання функцій вводу-виводу. Застосування вмінь і навиків програмування алгоритмів лінійної, розгалуженої та циклічної структури.

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?

Завдання.

1. **Створити програму з формою за зразком (надписи на кнопках мають відповідати наступним завданням). Використовуючи процедуру обробки події натиснення третьої кнопки, забезпечте при її натисненні завершення роботи програми.**



2. **Скласти програму, що викликається при натиснення першої кнопки з головної кнопкової форми.**

Вимоги до програми:

- значення текстового рядка для обробки ввести з клавіатури;
- для обробки рядків використати оператори циклу;
- результати виконання алгоритму вивести в одному діалоговому вікні, відобразивши в ньому як вхідний так і вихідний текстові рядки.

Варіанти:

1. У текстовому рядку підрахувати кількість букв 'о';
2. Переписати текстовий рядок навпаки;

3. Подвоїти у текстовому рядку букву 'к';
4. Встановити, чи у текстовому рядку букв 'п' більше ніж букв 'р';
5. Визначити позиції входження у текстовий рядок букви 'С';
6. Визначити кількість входжень у текстовий рядок буквосполучення "нн";
7. Видалити з текстового рядка букву 'к';
8. Вставити у текстовий рядок після кожної букви знак питання;
9. У текстовому рядку підрахувати кількість букв 'м';
10. Подвоїти у текстовому рядку букву 'л';
11. Встановити, чи у текстовому рядку букв 'ч' більше ніж букв 'ш';
12. Визначити позиції входження у текстовий рядок букви 'Ф';
13. Визначити кількість входжень у текстовий рядок буквосполучення "кк";
14. Видалити з текстового рядка букву 'д';
15. Вставити у текстовий рядок після кожної букви знак оклику.

3. Скласти програму, що викликається при натисненні другої кнопки з головної кнопкової форми.

Вимоги до програми:

- значення текстового рядка для обробки ввести з клавіатури;
- для обробки рядків використати оператори циклу;
- результати виконання алгоритму вивести в одному діалоговому вікні, відобразивши в ньому як вхідний так і вихідний текстові рядки.

Варіанти:

1. Замінити в текстовому рядку кожну крапку трьома крапками;
2. Перетворити заданий текстовий рядок з n символів, видаливши кожен символ * і повторивши кожен символ, відмінний від *;
3. Видалити з текстового рядка всі буквосполучення *про*;
4. В текстовому рядку замінити пробіли двома пробілами;
5. В текстовому рядку підрахувати кількість букв з вашого імені.
6. Замінити в текстовому рядку кожну кому двома знаками питання;
7. Перетворити заданий текстовий рядок, повторивши кожен символ, відмінний від '?';
8. Видалити з текстового рядка всі буквосполучення *не*;
9. В текстовому рядку замінити лапки двома комами;
10. В текстовому рядку видалити зайві пробіли між словами, залишивши по одному;
11. Переписати даний текстовий рядок так, щоб порядок символів у ньому став оберненим, самі символи подвоїлися і перед кожним з'явилася крапка;
12. З заданому текстовому рядку, у подвоїти всі букви *л*, *н*, *с*, *о*;
13. Видалити з текстового рядка всі коми і замінити знаком + всі цифри *3*;
14. Перетворити задану послідовність символів, замінивши всі буквосполучення *оце* буквосполученням *це*;
15. Видалити групи пробілів, якими починається і якими закінчується текстовий рядок.

Лабораторна робота № 10

Тема. Обробка даних вкладених об'єктів – елементів керування Windows-форм.

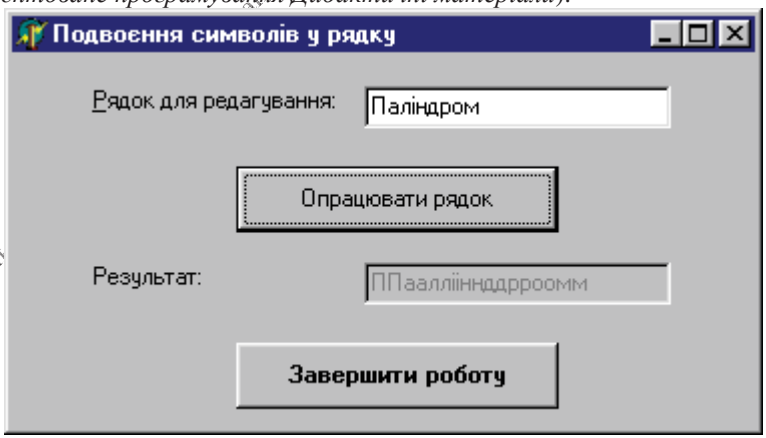
Мета. Формування вмінь і навиків програмування алгоритмів обробки текстових рядків у формах. Закріплення вмінь і навиків використання функцій обробки рядків. Застосування вмінь і навиків програмування алгоритмів лінійної, розгалуженої та циклічної структури.

Контрольні запитання.

1. Які елементи керування найчастіше використовуються у формах для введення даних від користувача?
2. Як і навіщо змінюють назви елементів керування?
3. Як встановити значення по замовчуванню для поля?
4. Як заборонити редагування даних у полі? Навіщо це роблять?
5. Що таке компоненти? Це об'єкт чи клас? Де вони описуються у модулі форми?

Завдання.

1. **Створити форму за зразком з полями редагування для введення початкового рядка і відображення результату, кнопкою для виконання завдання згідно варіанту та кнопкою для завершення роботи програми (максимальна оцінка – 2 бали). В процесі виконання завдання керуйтеся послідовністю дій, вказаних на лекції. При потребі використайте першу половину дидактичних матеріалів до лабораторної роботи** (завантажуються на сайті www.comp.ucoz.net з сторінки *Об'єктно-орієнтоване програмування\Дидактичні матеріали*).



Вимоги до програми:

- значення текстового рядка для обробки ввести з форми;
- для обробки рядків використати оператори циклу;

- результати виконання алгоритму вивести в іншому рядку тієї самої форми.

Варіанти:

1. У текстовому рядку підрахувати кількість букв 'о';
2. Переписати текстовий рядок навпаки;
3. Подвоїти у текстовому рядку букву 'к';
4. Встановити, чи у текстовому рядку букв 'п' більше ніж букв 'р';
5. Визначити позиції входження у текстовий рядок букви 'С';
6. Визначити кількість входжень у текстовий рядок буквосполучення "нн";
7. Видалити з текстового рядка букву 'к';
8. Вставити у текстовий рядок після кожної букви знак питання;
9. У текстовому рядку підрахувати кількість букв 'м';
10. Подвоїти у текстовому рядку букву 'л';
11. Встановити, чи у текстовому рядку букв 'ч' більше ніж букв 'ш';
12. Визначити позиції входження у текстовий рядок букви 'Ф';
13. Визначити кількість входжень у текстовий рядок буквосполучення "кк";
14. Видалити з текстового рядка букву 'д';
15. Вставити у текстовий рядок після кожної букви знак оклику.

2. Створити форму, дотримуючись вимог попередньої задачі, для виконання завдання згідно варіанту (максимальна оцінка – 2 бали).

Варіанти:

1. Замінити в текстовому рядку кожну крапку трьома крапками;
2. Перетворити заданий текстовий рядок з n символів, видаливши кожен символ * і повторивши кожен символ, відмінний від *;
3. Видалити з текстового рядка всі буквосполучення *про*;
4. В текстовому рядку замінити пробіли двома пробілами;
5. В текстовому рядку підрахувати кількість букв з вашого імені.
6. Замінити в текстовому рядку кожну кому двома знаками питання;
7. Перетворити заданий текстовий рядок, повторивши кожен символ, відмінний від '?';
8. Видалити з текстового рядка всі буквосполучення *не*;
9. В текстовому рядку замінити лапки двома комами;
10. В текстовому рядку видалити зайві пробіли між словами, залишивши по одному;
11. Переписати даний текстовий рядок так, щоб порядок символів у ньому став оберненим, самі символи подвоїлися і перед кожним з'явилася крапка;
12. З заданому текстовому рядку, у подвоїти всі букви *л*, *н*, *с*, *о*;
13. Видалити з текстового рядка всі коми і замінити знаком + всі цифри *3*;
14. Перетворити задану послідовність символів, замінивши всі буквосполучення *оце* буквосполученням *це*;
15. Видалити групи пробілів, якими починається і якими закінчується текстовий рядок.

3. Створити форму, дотримуючись вимог першої задачі, для виконання завдання стосовно обробки слів (групи символів, відокремлених одним або кількома розділовими знаками, що не містять розділових знаків всередині себе, будемо називати *словами*). Для розв'язання завдання опишіть у модулі форми після декларації *implementation* функцію *RozdilZnak*. Максимальна оцінка – 2 бали. Розробка замість форми по варіанту лише форми для поділу на слова – 1 бал.

Варіанти:

1. Підрахувати загальну кількість слів;
2. Знайти кількість слів, що починаються на букву *С*;
3. Знайти кількість слів, в яких перший і останній символи співпадають;
4. Знайти яке-небудь слово, що починається з букви *а*, або вказати на його відсутність;
5. У словах з закінченням *ий* замінити його закінченням *енький*;
6. Знайти і вивести всі слова, що починаються складом *не*;
7. Визначити слово даного рядка з максимальною часткою голосних;
8. Визначити, скільки разів у тексті зустрічається введене слово;
9. Перетворити даний рядок, замінюючи всі слова *оце* словом *це*;
10. Визначити, який процент слів у тексті починається з букви *к*;
11. Визначити, який процент слів у тексті містить подвоєну приголосну;
12. Визначити, з якої букви починається найбільше слів у рядку;
13. В заданому текстовому рядку всюди замінити слово А1 словом А2 (довжини слів не співпадають);
14. Студенти шифрують свої записки, записуючи всі слова навпаки. Скласти алгоритм і програму шифрування і розшифровування тексту;
15. Вказати мінімальну кількість перших букв, за якими можна розрізняти слова рядка.

Лабораторна робота № 11

Тема. Обробка подій вкладених об'єктів – елементів керування Windows-форм. Розробка власних методів у формах. Організація коректного введення даних у формах.

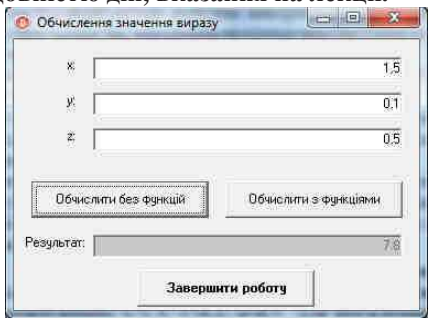
Мета. Формування вмінь і навиків створення та використання процедур та функцій користувача. Закріплення вмінь і навиків використання функцій вводу-виводу та обробки рядків. Застосування вмінь і навиків програмування алгоритмів лінійної, розгалуженої та циклічної структури.

Контрольні запитання.

1. Коли та з якою метою доцільно використовувати підпрограми?
2. Чи прискорює час виконання програми використання підпрограм? Чи зменшує об'єм коду програми та час її розробки таке використання?
3. Які типи підпрограм використовуються в C#? Чим відрізняються синтаксиси їх опису? Як викликаються підпрограми кожного типу?
4. Де описуються підпрограми користувача? Як визначити їх область видимості?
5. Від чого залежить результат виконання підпрограми? Параметри яких типів даних можуть використовуватися підпрограмою?
6. Чим формальні параметри підпрограми відрізняються від фактичних?
7. Як повернути результат виконання підпрограми-функції в основну програму?
8. Які виклики функцій називаються рекурсивними? Як їх здійснити? Які вимоги при цьому мають виконуватися для уникнення зациклень?

Завдання.

1. **Скласти програму обчислення значення виразу згідно варіанту при значеннях параметрів по замовчуванню x=1.5; y=0.1; z=0.5. Для візуалізації обчислень створіть форму за зразком з полями редагування для введення і відображення результату, кнопками для виконання завдання згідно варіанту та кнопкою для завершення роботи програми (максимальна оцінка – 3 бали). В процесі виконання завдання керуйтеся послідовністю дій, вказаних на лекції.**



Вимоги до програми:

- значення змінних ввести з форми;
- для розв'язання завдання при натисненні другої кнопки та для перевірки введення числа створити та використати підпрограми;
- результат виконання програм вивести в рядок виводу.

Варіанти:

1. $S = \dfrac{\sqrt{4^2 + x^2}}{\sqrt{x^2 + y^2}} + \dfrac{\sqrt{65^2 + y^2}}{\sqrt{6{,}4^2 + z^2}}$;

2. $S = \dfrac{87}{\lg\left|1 + x\tan y\right|} + \dfrac{\lg\left|1 + 2\tan y\right|}{34} + \dfrac{7{,}54}{\lg\left|1 + \tan 1.3\right|}$;

3. $S = \dfrac{78}{\lg\left|1.3 - x\tan z\right|} + \dfrac{\lg\left|x - 1.9\tan y\right|}{4{,}75} + \dfrac{5{,}34}{\lg\left|zx - 3\tan 1.2\right|}$;

4. $S = \dfrac{65}{\cos(xz - 1.7)} + \dfrac{\cos\left(3.6 - x^2\right)}{65{,}87} + \dfrac{6{,}74}{\cos(x - 0.3z)}$;

5. $S = \dfrac{\sqrt[3]{2 + \cos x^2}}{254} + \dfrac{9{,}56}{\sqrt[3]{3 + \cos y^2}} + \dfrac{\sqrt[3]{2 + \cos(xy)^2}}{7{,}4}$;

6. $S = \dfrac{\lg\left|1 + \cos^2 x^2\right|}{83} + \dfrac{56{,}9}{\lg\left|1 + \cos^2 0.8\right|} + \dfrac{\lg\left|1 + \cos^2 4.9\right|}{7{,}94}$;

7. $S = \dfrac{\lg\left|\sin x + y\right|}{87} + \dfrac{234}{\lg\left|\sin xy + 3\right|} + \dfrac{\lg\left|\sin 3.1 + x^2 y\right|}{z}$;

8. $S = \dfrac{\sqrt[3]{2 + \cos x^2}}{84} + \dfrac{5{,}87x}{\sqrt[5]{3 + \cos y^2}} + \dfrac{\sqrt[3]{2 + \cos(xy)^2}}{9}$;

9. $S = \dfrac{\left|x + y^2 \tan z\right|}{756} + \dfrac{8{,}45}{\left|x - 1 + \tan y\right|} + \dfrac{\left|x^2 + \tan 0.5\right|}{xy}$;

10. $S = \dfrac{\ln\left|\cos^2 x^2 + 1\right|}{67} - \dfrac{8{,}45}{\ln\left|\cos^2 x + 1\right|} + \dfrac{\ln\left|\cos^2 0.81 + 1\right|}{34x}$;

11. $S = \dfrac{\ln\left|x - y\right|}{76} - \dfrac{3zx}{\ln\left|1.3 - xy\right|} + \dfrac{\ln\left|y - 1.3z\right|}{98}$;

12. $S = \dfrac{\sin^2(x - y)}{78} + \dfrac{xy}{\sin^2(1.3 - xy)} + \dfrac{\sin^2(1.3x - 0.6)}{5z}$;

13. $S = \dfrac{45}{\sqrt[3]{\cos x^2 - y}} + \dfrac{\sqrt[5]{\cos y^2 - x}}{65x} - \dfrac{76zy}{\sqrt{\cos z^2}}$;

14. $S = \dfrac{xz}{\cos^2(0.7 - x)} - \dfrac{\cos^3(0.7 - xy)}{59x} + \dfrac{5xyz}{\cos^2(0.7 - 1.4y)}$;

15. $S = \dfrac{\sqrt[3]{\cos x^2 + 2}}{78} + \dfrac{87xz}{\sqrt[5]{\cos y^2 + 3}} + \dfrac{\sqrt[5]{\cos(yz)^2 + x}}{xy - z}$.

Наприклад, для розв'язання завдання першого варіанту необхідно:

1.1. Створити форму за наведеним зразком в режимі конструктора, послідовно задати полям вводу імена *PoleX*, *PoleY*, *PoleZ*, а полю-результату *PoleRes*;

1.2. Для полів вводу у властивості *Text* задати числові значення по замовчуванню;

1.3. Використовуючи властивість цих полів *TextAlign*, забезпечити вирівнювання введених чисел по правому краю;

1.4. Забезпечити коректне введення чисел в поля вводу, а саме:

1.4.1. Забезпечити можливість введення в ці поля лише чисел, коми та знака мінуса. Для цього потрібно створити процедуру обробки їх подій *KeyPress*, подібну до такої:

```
private void Pole_KeyPress(object sender, KeyPressEventArgs e)
{  if (e.KeyChar != 8 && e.KeyChar != ',' && e.KeyChar != '-' && (e.KeyChar < 48 || e.Ke
    {e.Handled = true;
     return;
    }
   int i;
   if (e.KeyChar=='-')
    {if(((TextBox)sender).SelectionStart>0)
      {e.Handled = true;
       return;
      }
     if (((TextBox)sender).Text.Length > 0)
      {bool minus=false;
       for (i = 0; i < ((TextBox)sender).Text.Length;i++)
        if (((TextBox)sender).Text[i]=='-')
         {minus = true;
          break;
         }
       if (minus) //мінус вже є
        {e.Handled = true;
         return;
        }}}
```

```
    if (e.KeyChar==',')
     {if (((TextBox)sender).Text.Length > 0)
       {bool koma=false;
        for (i = 0; i < ((TextBox)sender).Text.Length;i++)
         if (((TextBox)sender).Text[i]==',')
          {koma = true;
           break;
          }
        if (koma) //кома вже є
         {e.Handled = true;
          return;
         }}}
}
```

1.4.2. Встановити значення цих полів рівними нулю, коли з них видаляються всі символи. Для цього треба створити процедуру обробки їх подій *TextChanged*, наприклад, таку:

```
private void Pole_TextChanged(object sender, EventArgs e)
{if (((TextBox)sender).Text.Length==0)
  {((TextBox)sender).Text="0";
   ((TextBox)sender).SelectAll();
  }
}
```

1.4.3. Перевірити можливість переведення введеного рядка в число. Для цього доцільно створити **загальну** процедуру переведення тексту поля вводу в число та використати її в процедурі обробки події *Validating*:

```
static bool InputDoublePole(out double zminna, TextBox Pole, string povidom)
{ zminna = 0;
  try //відключити контроль помилок
   {zminna=Convert.ToDouble(Pole.Text);
   }
  catch(System.FormatException)
   {MessageBox.Show(povidom, "Увага", MessageBoxButtons.OK, MessageBoxIcon.E
    Pole.Focus();
    return false;
   }
  return true;
 }

private void Pole_Validating(object sender, CancelEventArgs e)
{double a = 0;
 if (!InputDoublePole(out a,(TextBox)sender,"Ви ввели не дійсне число"))
  {e.Cancel = true;
   return;
  }
}
```

1.5. Створити процедуру обробки натиснення кнопки *Обчислити без функції*, яка у випадку коректного введення чисел в поля вводу

обчислює значення виразу безпосередньо. Для цього прикладу вона може бути, наприклад, така:

```
private void button1_Click(object sender, EventArgs e)
 {Double x=0, y=0, z=0, res;
  if (!InputDoublePole(out x, PoleX, "Ви ввели не число в поле змінної x"))
   return; //якщо в PoleX не введено число, то фокус переходить в це поле
  if (!InputDoublePole(out y, PoleY, "Ви ввели не число в поле змінної y"))
   return;
  if (!InputDoublePole(out z, PoleZ, "Ви ввели не число в поле змінної z"))
   return;
  res=Math.Sqrt(4*4+x*x)/Math.Sqrt(x*x+y*y)+Math.Sqrt(65*65+y*y)/Math.Sqrt(6.4
  PoleRes.Text=res.ToString();
 }
```

1.6. Враховуючи те, що обчислення кореня з суми квадратів двох чисел у цьому прикладі виконується чотири рази, в класі цієї форми доцільно створити статичну функцію, яка буде обчислювати корінь з суми квадратів двох чисел і викликати цю функцію чотири рази в процедурі обробки натиснення кнопки *Обчислити з функцією*, наприклад, так:

```
static double Korin(double C1, double C2)
 {return Math.Sqrt(C1*C1+C2*C2);
 }

private void button2_Click(object sender, EventArgs e)
 {Double x=0, y=0, z=0, res;
  if (!InputDoublePole(out x, PoleX, "Ви ввели не число в поле змінної x"))
   return; //якщо в PoleX не введено число, то фокус переходить в це поле
  if (!InputDoublePole(out y, PoleY, "Ви ввели не число в поле змінної y"))
   return;
  if (!InputDoublePole(out z, PoleZ, "Ви ввели не число в поле змінної z"))
   return;
  res = Korin(4, x) / Korin(x, y) + Korin(65, y) / Korin(6.4, z);
  PoleRes.Text=res.ToString();
 }
```

1.7. Забезпечити початкове обчислення результату при завантаженні форми (подія *Load*) за допомогою виклику наведеної вище процедури обробки натиснення кнопки:

```
private void Form1_Load(object sender, EventArgs e)
 {button2_Click(this, new EventArgs());
 }
```

2. Опишіть у модулі форми лабораторної роботи № 7 функції для коректного введення цілих і дійсних чисел (максимальна оцінка – 2 бали). Використайте ці функції для забезпечення коректного введення значень всіх числових змінних.

Лабораторна робота № 12

Тема. Реалізація наслідування класів-форм. Програмування віртуальних методів класів форм.

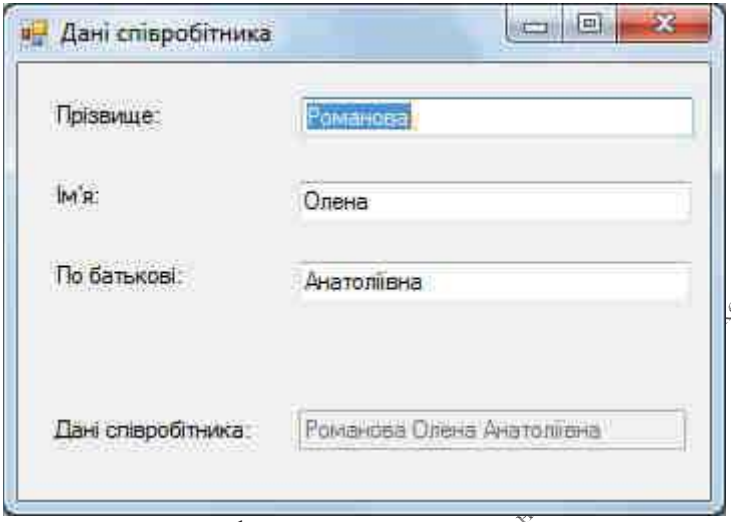
Мета. Формування вмінь і навиків програмування віртуальних методів класів. Закріплення вмінь і навиків програмування наслідування об'єктів, використання модулів, класів, об'єктів, файлів, підпрограм, функцій вводу-виводу та обробки рядків. Застосування вмінь і навиків програмування алгоритмів лінійної, розгалуженої та циклічної структури.

Контрольні запитання.

1. Як реалізувати наслідування класів?
2. Як змінити клас і назву для об'єкта-форми? Як змінити батьківський клас для класу?
3. Як перейти до тексту програми з візуальної форми? У файлах з якими розширеннями зберігаються описи класів та параметри зовнішнього вигляду форм?
4. Як описати, створити та використати візуальні компоненти батьківського класу?
5. Як звернутися до аналогічного методу батьківського класу?
6. Як забезпечити виклик з батьківського класу методу нащадка?

Завдання.

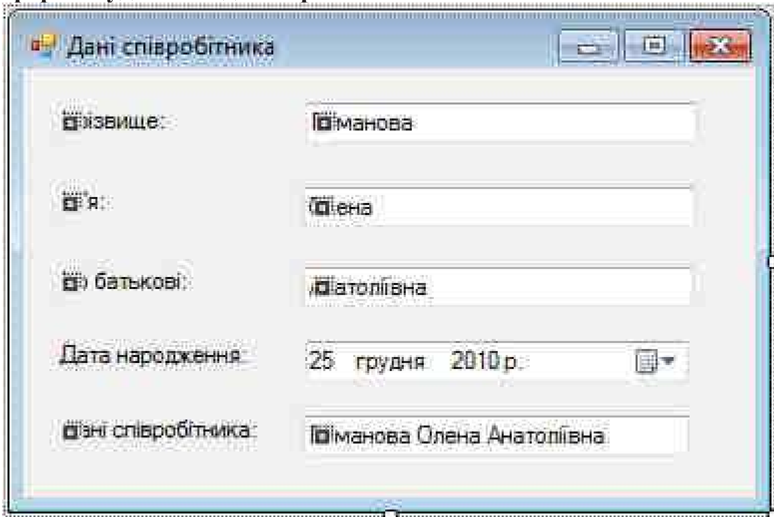
1. **Створіть новий проект, а в ньому – форму за зразком, наведеним нижче. Перейменуйте клас та файл форми на *FormBase*. Перейменуйте перших три поля відповідно на *PrizvFO, NameFO та FNameFO*, *а четверте – на Rez.* В перших три поля введіть ваші дані по замовчуванню. Забороніть доступ до четвертого поля в процесі експлуатації. Опишіть у класі форми публічну процедуру *formRez()* для автоматичного формування значення четвертого поля з перших трьох. Забезпечте її виклик як при редагуванні кожного з трьох перших полів (подія *TextChanged*), так і при завантаженні форми (подія *Load*). Забезпечте загальнодоступність поля результату *Rez* з класу форми та породжених класів. Для цього або в режимі конструктора виділіть це поле та встановіть значення його властивості *Modifiers* рівним *public* чи *protected* або перейдіть у файл опису візуальних компонентів і забезпечте в ньому загальнодоступність поля результату (замінивши модифікатор доступу *private* на *public* чи *protected*). Переконайтеся в дієздатності розробленої форми.**



При цьому текст модуля форми може виглядати, наприклад, так:

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

    virtual protected void formRez()
     {string s;
      s=PrizvFO.Text;
      if (NameFO.Text!="")
       {if (s!="")
          s+=" ";
        s+=NameFO.Text;
       }
      if (FNameFO.Text!="")
       {if (s!="")
          s+=' ';
        s+=FNameFO.Text;
       }
      Rez.Text = s;
     }

    private void Prizv_TextChanged(object sender, EventArgs e)
     {formRez();
     }

    private void Name_TextChanged(object sender, EventArgs e)
     {formRez();
     }

    private void FName_TextChanged(object sender, EventArgs e)
     {formRez();
     }

    private void FormBase_Load(object sender, EventArgs e)
    {formRez();
    }
}
```

2. Додайте до вашого проекту ще одну форму Windows, в якій крім прізвища, імені та по батькові буде вводитися ще й дата народження, а додатково обчислюватися ще й кількість років. В режимі конструктора ця форма буде виглядати приблизно так:



Для цього:

2.1. В оглядачі рішень виділіть файл проекту та оберіть в його контекстному меню пункт *Добавить – Форма Windows*, після чого у вікні створення нового елемента вкажіть *Форма Windows Form*;

2.2. Перейменуйте клас та файл цієї форми на *FormPorodgeno.* Забезпечте відображення цієї форми при завантаженні додатку замість завантаженням об'єкта форми *FormBase*. Для цього у файлі *Program.cs* замініть команду `Application.Run(new FormBase());` на `Application.Run(new FormPorodgeno());`

2.3. **З метою зменшення кількості повторюваних описів забезпечте породження класу цієї форми від *FormBase*.** Після цього в режимі конструктора форми мають автоматично з'явитися всі елементи керування базової форми з символом закріплення. Чому поле результату переміщувати і редагувати можливо, а вхідні поля – ні;

2.4. Для введення дати народження додайте у форму *FormPorodgeno* елемент керування класу *DateTimePicker*. Дайте йому назву *ДатаНародження*, та вкажіть вашу дату народження як значення по замовчуванню у властивості *Value*;

2.5. Перевизначте у класі *FormPorodgeno* публічну процедуру *formRez()* для додатково підрахунку кількості прожитих років та викличте її і при зміні поля *ДатаНародження* (подія *ValueChanged*), наприклад, так:

```
public partial class FormPorodgeno : FormBase
{
    public FormPorodgeno() : base()
    {
        InitializeComponent();
    }

    protected void formRez()
    {
        base.formRez();
        string s = Rez.Text;
        if (s != "")
         s+=" ";
        s+= " ("+((int)((double)(DateTime.Now-ДатаНародження.Value).Days/36
            +" років"+ ')';
        Rez.Text = s;
    }

   private void ДатаНародження_ValueChanged(object sender, EventArgs e)
    {formRez();
    }
}
```

Що означає зарезервоване слово *base*?

2.6. Завантажте додаток на виконання. Як відрізняється значення поля результату при редагуванні трьох перших полів і поля дати народження? Чому?

2.7. **Для відображення повної інформації про співробітника і при редагуванні перших трьох полів забезпечте виклик оновленої процедури *formRez()* в межах нащадка *FormPorodgeno* і з батьківського класу *FormBase*. Для цього в модулі батьківського класу *FormBase* перед описом процедури *formRez()* вкажіть зарезервоване слово *virtual* (якщо ви цього не зробили раніше), а в модулі класу *FormPorodgeno* перед описом цієї ж процедури – зарезервоване слово *override*.** Що змінилося у функціонуванні об'єкта форми *FormPorodgeno*? Яке зв'язування методів при цьому реалізується?

3. Конкурс на 5 балів до рейтингу за найшвидше розв'язання завдання: забезпечте одним додатковим рядком коду відкриття при завантаженні програми не лише екземпляра форми *FormPorodgeno*, а й екземпляра форми *FormBase*.

Лабораторна робота № 13

Тема. Програмування перевантажуваних методів у формах. Використання текстових файлів та діалогових вікон загального призначення.

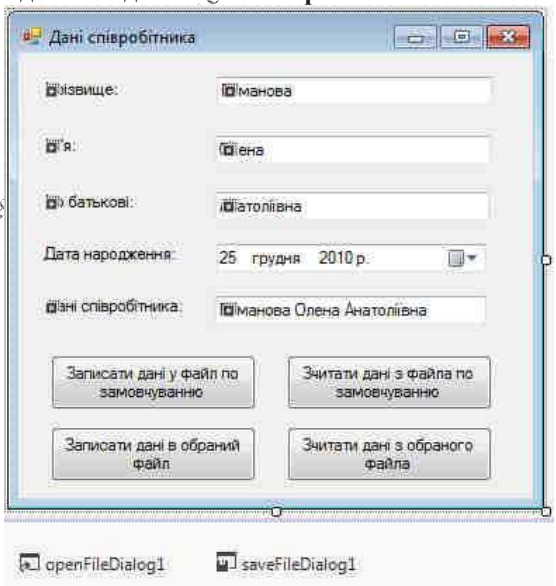
Мета. Формування вмінь і навиків програмування перезавантажуваних методів, зчитування і запису з текстових файлів. Закріплення вмінь і навиків використання модулів, класів, об'єктів, файлів, підпрограм, функцій вводу-виводу та обробки рядків. Застосування вмінь і навиків програмування алгоритмів лінійної, розгалуженої та циклічної структури.

Контрольні запитання.

1. Які методи називаються перезавантажуваними? Коли вони використовуються?
2. Як описати перезавантажувані методи?
3. Як створити, використати і знищити об'єкт стандартного діалогового вікна?
4. В яких режимах можна опрацьовувати текстові файли? Чому ці файли відносяться до файлів послідовного доступу?
5. Коли використовують текстові файли?
6. У чому переваги використання текстових файлів?

Завдання.

1. Відкрийте проект *FormPorodgeno*, розроблений в попередній лабораторній роботі. Доповніть його форму *FormPorodgeno* чотирма кнопками згідно наведеного нижче зразка.



2. **Створіть у цій формі дві перезавантажувані процедури *writeFile*: перша – для запису прізвища, імені, по батькові і дати народження у файл по замовчуванню, друга – для запису цих даних у файл з вказаною назвою.** Для цього в модулі форми відкрийте спочатку простір імен *System.IO*. А самі процедури можуть виглядати, наприклад, так:

```
private void WriteFile(string filename)
{string[] masStr = new string[4];
 masStr[0] = PrizvFO.Text;
 masStr[1] = NameFO.Text;
 masStr[2] = FNameFO.Text;
 masStr[3] = ДатаНародження.Value.ToString();
 File.WriteAllLines(filename, masStr);
 MessageBox.Show("Дані збережено", "Інформація");
}

private void WriteFile()
{WriteFile("defaul.txt");
}
```

3. **Забезпечте виклик першого варіанту цієї процедури при натисненні верхньої лівої кнопки. В процедурі обробки події натиснення нижньої лівої кнопки для введення назви файла використайте об'єкт стандартного діалогового вікна та передайте цю назву в процедуру *writeFile*, як це показано нижче.**

```
private void button2_Click(object sender, EventArgs e)
{saveFileDialog1.DefaultExt = "txt";
 saveFileDialog1.AddExtension = true;
 saveFileDialog1.Title = "Оберіть файл для збереження параметрів";
 if (saveFileDialog1.ShowDialog() == DialogResult.Cancel)
  return;
 WriteFile(saveFileDialog1.FileName);
}
```

4. **Створіть дві перевантажувані процедури для зчитування даних з файлів по замовчуванню і з введеною назвою.** Ці процедури можуть виглядати, наприклад, так:

```
private void ReadFile(string filename)
{if (!File.Exists(filename))
  {MessageBox.Show("Файл не існує! Дані не зчитано", "Увага",
                   MessageBoxButtons.OK, MessageBoxIcon.Exclamation);
   return;
  }
 string[] masStr;
 masStr=File.ReadAllLines(filename);
 if (masStr.Length!=4)
  {MessageBox.Show("Файл пошкоджено! Дані не зчитано", "Увага",
                   MessageBoxButtons.OK, MessageBoxIcon.Exclamation);
   return;
  }
 PrizvFO.Text=masStr[0];
 NameFO.Text=masStr[1];
 FNameFO.Text=masStr[2];
 try
 {ДатаНародження.Value = Convert.ToDateTime(masStr[3]);
 }
 catch
 {MessageBox.Show("Дату не зчитано. Встановіть коректну дату", "Увага",
                  MessageBoxButtons.OK, MessageBoxIcon.Exclamation);
  ДатаНародження.Focus();
  return;
 }
 MessageBox.Show("Дані зчитано успішно","Інформація");
}

private void ReadFile()
{ReadFile("defaul.txt");
}
```

5. **Використовуючи процедури попереднього завдання, забезпечте функціональність двох кнопок справа.**

Лабораторна робота № 14

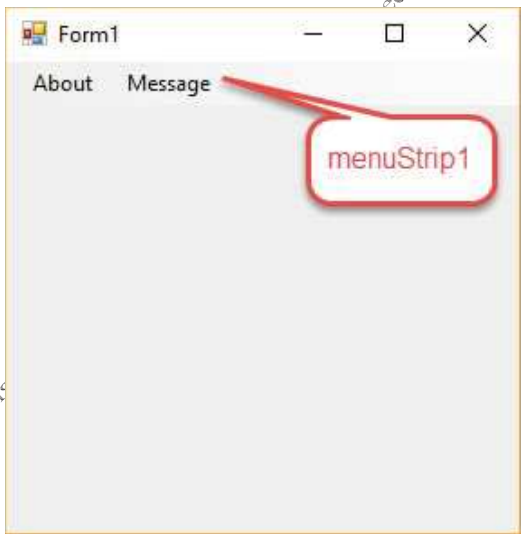
Тема. Створення додатків з головним меню. Використання конструкторів форм.

Мета. Формування вмінь і навиків створення додатків з головним меню та використання конструкторів форм. Закріплення вмінь і навиків використання функцій вводу-виводу. Застосування вмінь і навиків програмування алгоритмів лінійної, розгалуженої та циклічної структури.

Співавторлабораторної: Крайчук Сергій Олександрович

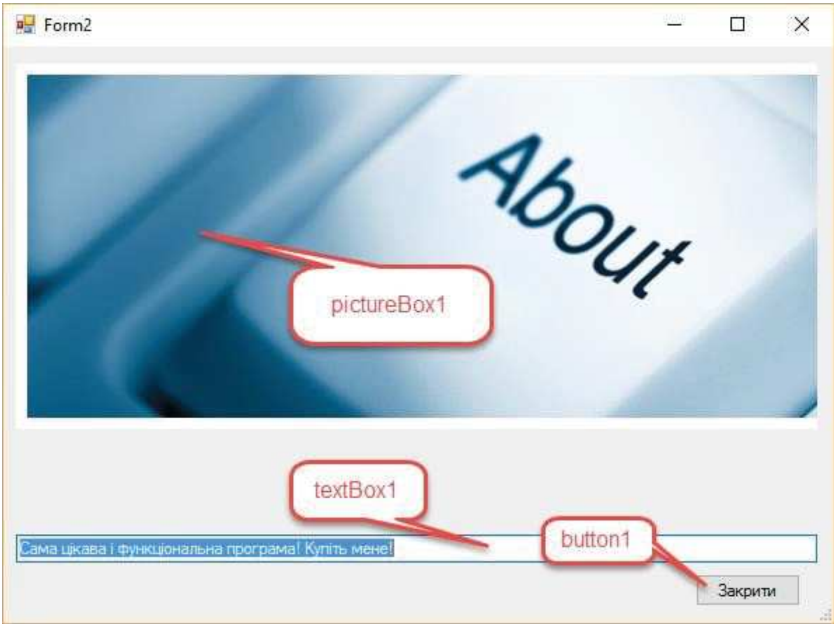
Метою даної лабораторної роботи буде організувати роботу з «універсальними» вікнами. У нас буде два вікна, одне – головне, а інше буде, в залежності від обраного пункту меню, відображати різні дані. Для прикладу, ми візьмемо обробку пункту про програму (*About*) та формування повідомлення (*Message*). Пункти будуть відображати різну інформацію, але за допомогою однієї форми. Отже, приступимо.

Перше вікно буде головним і воно буде мати такий вигляд:



Єдиний компонент, який ми будемо тут використовувати, – це рядок меню.

Одразу ж наведемо і вигляд другої форми:



У цій формі нам вже потрібні область для відображення зображення та якийсь компонент, що буде відображати текст (для простоти візьмемо елемент *textBox*).

Тепер нам потрібно описати базові функції нашої «універсальної» форми. Оскільки нам потрібно, щоб наша форма з самого початку була створена з потрібним написом та малюнком – перекриємо її конструктор:

```
public Form2(string fn, string text, bool
linked)
//конструктор форми містить три аргументи: назва
файла зображення,
//текст для виводу та тип розміщення файла
зображення – пов'язаний
//(міститься у папці \Bin\Debug біля .exe-файла
рішення) чи
//вбудований (міститься в ресурсах проекту, тобто в
самому .exe-
//файлі)
 {InitializeComponent(); //ініціалізація
візуальних компонентів
```

```
  //форми. Без неї редагування візуальних
компонентів недоступне
  textBox1.Text = text; //присвоюємо текст полю
вводу
  if (linked)
   pictureBox1.Load(fn); //завантажуємо зображення з
файла
  else
   {System.Resources.ResourceManager rm =
    Properties.Resources.ResourceManager;
    //завантажуємо зображення з ресурсів
    pictureBox1.Image=(Image)rm.GetObject(fn);
   }}
```

Функціонування кнопки *Закрити* забезпечте самостійно.

Отже, допоміжна форма готова. Тепер нам потрібно її створювати в залежності від обраного пункту меню головної форми з відповідними параметрами. Наприклад, процедура обробки події вибору пункту *About* може бути така:

```
private void aboutToolStripMenuItem_Click(object
sender,EventArgs e)
 {Form2 f2 = new Form2("about.jpg",
   "Сама цікава і функціональна програма! Купіть
мене!", true);
  //викликаємо форму, передаємо назву файла
зображення, текст для
  //підпису та тип зв'язку – пов'язаний
  f2.ShowDialog(); //модально виводимо форму на
екран
 }
```

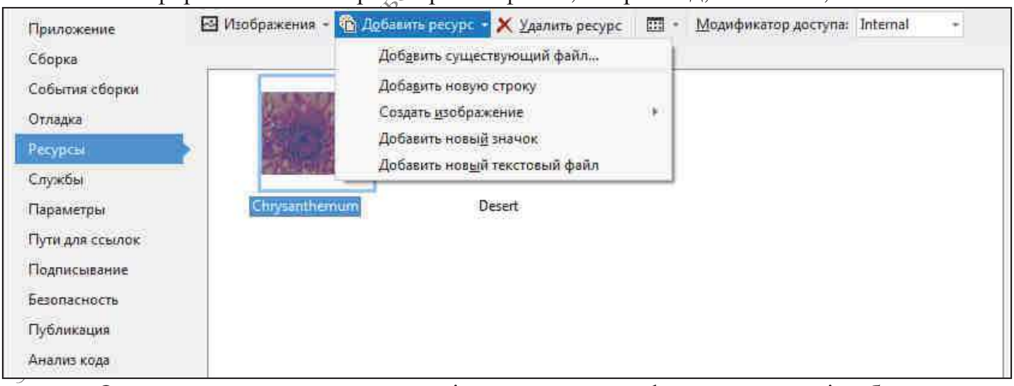
І код елемента меню «Message»:

```
private void messageToolStripMenuItem_Click(object
sender,

EventArgs e)
 {Form2 f2 =new Form2("Desert","Все для Вас і
безкоштовно!", false);
```

```
   //викликаємо форму, передаємо назву ресурсу, текст
для підпису
   //та тип зв’язку – вбудований з ресурсів проекту
   f2.ShowDialog();
  }
```

Для забезпечення функціонування розробленого рішення залишилося тільки забезпечити доступність використовуваних зображень. Для цього:

- файли **пов’язаних** зображень (третій параметри при створенні *Form2* рівний *true*) мають знаходитися у вкладеній папці *Bin**Debug* біля *.exe*–файла рішення. Назви таких файлів передаються в конструктор форми з розширенням, наприклад, `about.jpg`;
- назви **вбудованих** зображень (третій параметри при створенні *Form2* рівний *false*) повинні міститися в ресурсах проекту. Для опрацювання цих ресурсів необхідно відкрити вікно властивостей проекту (виділити проект з піктограмою *C#* у вікні оглядача рішень і обрати в його контекстному меню чи в головному меню *Проект* пункт *Свойства*) і перейти у ньому на вкладку *Ресурсы*. Щоб додати до ресурсів файл зображення, необхідно на цій вкладці у списку кнопки *Добавить ресурс* обрати пункт *Добавить существующий файл* (як на рис. нижче), відмітити потрібний файл та натиснути кнопку *Открыть*. Назви збережених ресурсів відображаються на даній вкладці та передаються в конструктор форми *Form2* без розширення файла, наприклад, `Desert`;



Ось таким чином можна динамічно створювати форми в процесі роботи. Для успішної здачі лабораторної роботи вам потрібно:

- додати в головне меню пункт *Фотоальбом*, а в нього – підпункти з назвами ваших фотографій (не менше трьох), при виборі яких має відображатися відповідна фотографія та коментар до неї;
- додати в кінець головного меню пункт *Вихід* та забезпечити його функціонування.

Лабораторна робота № 15

Тема. Одночасне використання кількох об'єктів декількох споріднених класів в об'єктно-орієнтованому програмуванні за допомогою вказівок.

Мета. Формування вмінь і навиків використання форм та класів раніше розроблених проектів. Закріплення вмінь і навиків використання функцій стандартних бібліотек, власних підпрограм, класів, об'єктів, функцій вводу-виводу та обробки рядків. Застосування вмінь і навиків програмування алгоритмів лінійної, розгалуженої та циклічної структури.

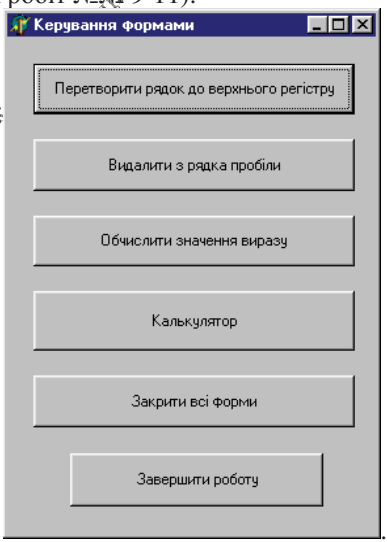
Контрольні запитання.

1. Що зберігається в пам'яті для класу і для об'єкта?
2. Як створити і знищити об'єкт класу?
3. Як перенести в новий проект розроблену раніше форму?
4. Як перевірити існування об'єкта класу форми?
5. Що необхідно зробити після створення об´єкта класу, породженого від форми, для забезпечення можливості використання його користувачем?

Завдання.

1. Створіть форму з кнопками для керування раніше розробленими формами. Для цього:

1.1. У новому рішенні *LR13MenuForm* розробіть зовнішній вигляд форми за зразком (надписи на кнопках мають відповідати завданням лабораторних робіт №№ 9-11):



Закрийте це рішення;

1.2. Для уникнення конфліктів у назвах форм почергово відкрийте рішення лабораторних робіт №№ 9-11 та задайте **в панелі оглядача рішень унікальні назви файлам форм** і погодьтеся з перейменуванням пов´язаних даних. З цією метою в панелі оглядача рішень відмітьте файл форми (з розширенням *.cs*), перейменуйте його з *Form1* на іншу згідно з її призначенням (наприклад, *Form1.cs* на *FormCalc.cs*, *FormStr.cs* чи *FormMath.cs*) і погодьтеся з автоматичним перейменуванням пов´язаних посилань. Як при цьому змінилася властивість *Имя файла* у вікні властивостей? Чи змінилася властивість *Name* для кожної форми? Якщо властивість *Name* для якоїсь форми не змінилася (це відбувається тоді, коли простір імен не є простором імен проекту по замовчуванню), то змініть її згідно імені файла безпосередньо у вікні властивостей в режимі конструктора або у вікні коду форми, обираючи в контекстному меню її назви пункт *Выполнить рефакторинг – Переименовать*. Чи змінилися назви просторів імен у вікнах коду форм?

1.3. Відкрийте одну з перейменованих форм в режимі конструктора і змініть її властивість *Name*. Чи змінилися при цьому назва файла, назва простору імен, назви пов'язаних даних, назва форми у вікні коду? Відновіть значення властивості *Name* згідно назви файла. Закрийте ці рішення;

1.4. Додайте перейменовані модулі форм лабораторних робіт №№ 9-11 (як з розширенням *.cs*, так і пов'язані файли) до нового рішення кнопкової форми. Для цього завантажте рішення кнопкової форми *LR13Menu*, в панелі оглядача рішень виділіть проект *LR13Menu* (з позначкою *C#*) та почергово в його контекстному меню оберіть пункт *Добавить – Существующий элемент* і додайте три перейменовані форми попередніх лабораторних робіт з розширенням *.cs* (наприклад, *FormCalc.cs*). Які ще файли додалися при цьому автоматично?

1.5. У вікні програми кнопкової форми відкрийте простори імен скопійованих форм, дописуючи у розділі описів їх назви (наприклад, *using LR11Calc;*);

1.6. На початку опису класу кнопкової форми *FormCalc* додайте вказівки на класи скопійованих форм, вказавши у наступних рядках після рядка-заголовка класу кнопкової форми клас кожної скопійованої форми та назву відповідної вказівки. Занесіть у ці вказівки початкове значення *null*. При цьому початок коду кнопкової форми має бути подібним до такого:

```
public partial class Form1 : Form
 {FormCalc f1=null;
  …
  public Form1()
   {InitializeComponent();
   }
  …
```

1.7. В процедурі обробки події натиснення кнопки виклику калькулятора забезпечте створення, а при існуванні – фокусування відповідної форми. При цьому код обробки цієї події має бути приблизно таким:

```
private void button4_Click(object sender, EventArgs e)
 {if (f1==null)
   {f1 = new FormCalc();
    f1.Show();
   }
  else
   if (f1.IsDisposed)
    {f1 = new FormCalc();
     f1.Show();
    }
   else
    {if (f1.WindowState == FormWindowState.Minimized)
      f1.WindowState = FormWindowState.Normal;
     else
      if (!f1.Focused)
       f1.Focus();
    }
 }
```

1.8. Аналогічно забезпечте завантаження відповідних форм класів інших підключених модулів для інших (крім двох останніх) кнопок головної кнопкової форми;

1.9. Для закриття відкритих форм створіть у кнопковій формі таку загальну процедуру:

```
private void closeForm()
 {if (f1!=null)
   if (!f1.IsDisposed)
    {f1.Close();
     f1=null;
    }
   else
    f1=null;
  … // інші форми знищуються аналогічно
 }
```

1.10. Забезпечте закриття відкритих форм при натисненні передостанньої і останньої кнопки та закритті кнопкової форми користувачем.

Лабораторна робота № 16

Тема. Одночасне використання багатьох об'єктів декількох споріднених класів в об'єктно-орієнтованому програмуванні за допомогою масивів. Поліморфні виклики методів об'єктів.

Мета. Формування вмінь і навиків створення та використання багатьох об'єктів декількох споріднених класів. Закріплення вмінь і навиків використання модулів, власних підпрограм, класів, об'єктів, функцій вводу-виводу та обробки рядків. Застосування вмінь і навиків програмування алгоритмів лінійної, розгалуженої та циклічної структури.

Контрольні запитання.

1. У чому полягає суть властивості поліморфізму?
2. Як створити і вивести на екран форму заданого класу?
3. Як організувати доступ до багатьох об'єктів за допомогою масиву? Який тип даних його елементів? Чому?
4. Чи може одна змінна використовуватися для керування багатьма об'єктами? Чому?
5. Які способи знищення форм ви знаєте?
6. Коли і як знищуються всі виведені форми?

Завдання.

1. Вдосконалте головну кнопкову форму, розроблену на попередній лабораторній роботі, забезпечивши керування багатьма об'єктами декількох споріднених класів. Для цього:

1.1. На початку опису класу головної кнопкової форми оголосіть константу максимальної кількості створених форм-об'єктів та масив вказівок на об'єкти. При створенні головної кнопкової форми занесіть у всі елементи цього масиву значення *null*. Наприклад, так:

```
public partial class Form1 : Form
 {const int maxForm=18;
  Form[] masForm=new Form[maxForm];

  public Form1()
   {InitializeComponent();
    for (int i = 0; i < maxForm; i++)
     masForm[i] = null;
   }
```

1.2. Реалізуйте загальну процедуру для створення об’єктів обраного класу, наприклад, таку:

```
private void createForm(int index)
 {int i = 0;
  while (i<maxForm)
   if (masForm[i]==null)
    break;
   else
    if (masForm[i].IsDisposed)
     break;
    else
     i++;
  if (i==maxForm)
   {MessageBox.Show("Більше форм створити неможливо");
    return;
   }
  switch (index)
   {case 1:masForm[i]= new FormCalc();
           break;
    case 2:masForm[i] = new FormString();
           break;
    … // інші об'єкти створюються аналогічно
   }
  masForm[i].Show();
 }
```

Забезпечте її виклики з різними значеннями параметрів для різних кнопок головної кнопкової форми. Наприклад, в процедурі обробки натиснення кнопки завантаження калькулятора потрібно вказати *createForm(1)*;

1.3. Модифікуйте процедуру для знищення всіх відкритих створених раніше форм-об'єктів, наприклад, так:

```
private void button2_Click(object sender, EventArgs e)
 {for (int i = 0; i < maxForm;i++ )
   if (masForm[i]!= null)
    if (!masForm[i].IsDisposed)
     {masForm[i].Close();
      masForm[i] = null;
     }
 }
```

1.4. Створіть додаткові кнопки і відповідні процедури для переміщення і зміни розміру всіх створених форм-об'єктів;

1.5. Реалізуйте власні процедури додаткових групових операцій над формами-об'єктами.

Лабораторна робота № 17

Тема. Одночасне використання необмеженої кількості об'єктів декількох споріднених класів в об'єктно-орієнтованому програмуванні за допомогою списків.

Мета. Формування вмінь і навиків створення та використання необмеженої кількості об'єктів декількох споріднених класів за допомогою списків. Закріплення вмінь і навиків використання модулів, власних підпрограм, класів, об'єктів, функцій вводу-виводу та обробки рядків. Застосування вмінь і навиків програмування алгоритмів лінійної, розгалуженої та циклічної структури.

Контрольні запитання.

1. Чим списки відрізняються від масивів?
2. Як створити список? Як додати у нього окремі елементи?
3. Як можливо звернутися до елементів списку? Які цикли для цього використовуються?
4. Які способи зміни елементів списку використовуються в C#?
5. Як очищується список? Що при цьому відбувається з елементами, які він містить і з об´єктами, на які вказують такі елементи?
6. Як знищити всі форми, на які вказують елементи списку?

Завдання.

1. Вдоскональте головну кнопкову форму, розроблену на попередній лабораторній роботі, забезпечивши керування необмеженою кількістю об´єктів декількох споріднених класів. Для цього:

1.6. На початку опису класу головної кнопкової форми оголосіть список форм без внесення в нього об´єктів, наприклад, так:

```
public partial class Form1 : Form
 {List<Form> listForm=new List<Form>();
```

1.7. Модифікуйте загальну процедуру для створення об’єктів обраного класу, наприклад, так:

```
public partial class Form1 : Form
 {List<Form> listForm=new List<Form>();

  private void createForm(int index)
   {Form f=null;
    switch (index)
     {case 1:f = new FormCalc();
             break;
      case 2:f = new FormString();
             break;
     }
    listForm.Add(f);
    f.Show();
   }
```

.

Забезпечте її виклики з різними значеннями параметрів для різних кнопок головної кнопкової форми. Наприклад, в процедурі обробки натиснення кнопки завантаження калькулятора потрібно вказати *createForm(1)*;

1.8. Модифікуйте процедуру для знищення всіх відкритих створених раніше форм-об'єктів, наприклад, так:

```
private void button2_Click(object sender, EventArgs e)
 {foreach (Form f in listForm)
   if (!f.IsDisposed)
    f.Close();
  listForm.Clear();
 }
```

;

1.9. Модифікуйте додаткові кнопки і відповідні процедури для переміщення і зміни розміру всіх створених форм-об'єктів, використовуючи два типи циклів, наприклад, так:

```
private void button6_Click(object sender, EventArgs e)
 {foreach (Form f in listForm)
   if (!f.IsDisposed)
    f.SetDesktopLocation(f.Location.X+10, f.Location.Y);
 }

private void button7_Click(object sender, EventArgs e)
 {int i, count = listForm.Count;
  for (i=0; i<count;i++)
   if (!listForm[i].IsDisposed)
    listForm[i].Size =
     new Size(listForm[i].Size.Width-10,listForm[i].Size.Height-10);
 }
```

;

1.10. Реалізуйте власні процедури додаткових групових операцій над формами-об'єктами.

Лабораторна робота № 18

Тема. Використання інтерфейсів та делегатів.

Мета. Формування вмінь і навиків створення інтерфейсів та делегатів засобами C#. Закріплення вмінь і навиків наслідування звичайних та абстрактних класів, використання об'єктів, підпрограм, елементів керування. Застосування вмінь і навиків програмування алгоритмів лінійної, розгалуженої та циклічної структури.

Контрольні запитання.

1. Що спільного між абстрактними класами та інтерфейсами? Чим вони відрізняються?
2. Як описуються інтерфейси?
3. Які специфікатори доступу і де можливо вказувати для інтерфейсів? Чому для членів інтерфейсів не вказуються специфікатори доступу?
4. Від скількох інтерфейсів може наслідуватися клас? Де при цьому вказується базовий клас, якщо від наявний?
5. Як використовується наслідування різних класів від одного інтерфейсу?
6. Які об'єкти називаються делегатами? Як використовуються делегати?

Завдання.

1. **Вдоскональте комплекс форм, розроблених на попередній лабораторній роботі, забезпечивши зберігання та використання тем лабораторних робіт, на яких вони були створені. Для цього:**
   1.1. Створіть нову папку для рішення цієї лабораторної роботи. Скопіюйте в неї рішення попередньої лабораторної роботи. Перейменуйте скопійоване рішення згідно теми цієї лабораторної роботи;
   1.2. Відкрийте скопійоване рішення. Змініть назву класу форми головного меню на *FormMenu*. У файлі коду головного меню під описом класу форми опишіть інтерфейс для зберігання тем лабораторних робіт всіма пов'язаними формами**.** Цей код може бути, наприклад, таким:

```
namespace LR19ListInterface
{
    public partial class FormMenu : Form...

    interface ILabWork
    {
        string GetTemaLabWork();
    }

}
```

Який рівень доступу описаного інтерфейсу?

1.3. Забезпечте підтримку розробленого інтерфейсу всіма пов'язаними формами. Для цього, по-перше, вкажіть назву цього інтерфейсу серед батьківських об'єктів цих форм і, по-друге, доповніть їх публічними методами *GetTemaLabWork()*. Тоді, наприклад, початок коду форми калькулятора може бути таким:

```
public partial class FormCalc : Form, ILabWork
 {  bool resultTablo = true;
    string operation="";

    public FormCalc()
    { InitializeComponent(); }

    public string GetTemaLabWork()
    { return "Організація взаємодії елементів керування у формах"; }

    ...
```

1.4. Доповніть головну кнопкову форму кнопкою *Підрахувати кількості форм по темах*. Підтримка спільного інтерфейсу всіма пов'язаними формами дає можливість виконувати явне перетворення до нього всіх форм зі списку, створених користувачем. Процедура її натиснення може бути, наприклад, такою:

```
private void button8_Click(object sender, EventArgs e)
{   List<ILabWork> ListAnaliz = new List<ILabWork>();
    foreach (Form f in listForm)
        ListAnaliz.Add((ILabWork)f);
    string res = "",tema;
    int index,count;
    while (ListAnaliz.Count > 0)
    {   tema = ListAnaliz[0].GetTemaLabWork();
        index = 0; count = 0;
        while (index < ListAnaliz.Count)
            if (ListAnaliz[index].GetTemaLabWork() == tema)
            {   count++;
                ListAnaliz.RemoveAt(index);
            }
            else
                index++;
        res += tema + " - " + count.ToString() + " форм\n";
    }
    if (res == "")
        res = "Форми відсутні";
    MessageBox.Show(res, "Статистика форм",
        MessageBoxButtons.OK, MessageBoxIcon.Information);
}
```

2. **Створіть в головній кнопковій формі делегат для порівняння двох форм за назвою теми. Доповніть головну кнопкову форму кнопкою *Відсортувати за темою і розмістити каскадом*.**

Додатковий код може бути, наприклад, таким:

```
public int SortByTema(Form f1, Form f2)
{   if (f1 == null || f2 == null)
        throw new ArgumentException("Окремі аргументи не є формами");
    return String.Compare((f1 as ILabWork).GetTemaLabWork(),
                          (f2 as ILabWork).GetTemaLabWork());
}

private void button9_Click(object sender, EventArgs e)
{   listForm.Sort(SortByTema);
    button5_Click(this, new EventArgs());
}
```

3. **Доповніть головну кнопкову форму кнопкою *Відсортувати за заголовком форми і розмістити каскадом* та самостійно забезпечте її функціонування.**

Лабораторна робота № 19

Тема. Вбудовані інтерфейси. Ітератори. Використання інтерфейсів та делегатів для порівняння та сортування об'єктів.

Мета. Формування вмінь і навиків використання інтерфейсів для порівняння та сортування об'єктів. Закріплення вмінь і навиків наслідування звичайних та абстрактних класів засобами C#, використання об'єктів, підпрограм, елементів керування. Застосування вмінь і навиків програмування алгоритмів лінійної, розгалуженої та циклічної структури.

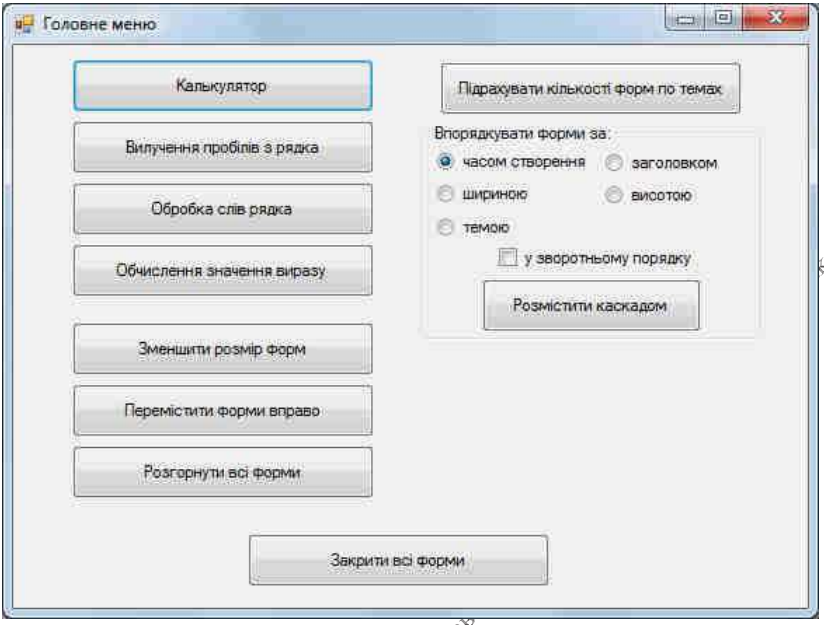
Контрольні запитання.

1. Які дві основні ролі класів? Які дві основні ролі вкладених класів?
2. Як організувати доступ до членів вкладеного класу? До яких членів класу можна звертатися без створення об'єкта класу?
3. Які основні перевантаження методу *Sort()* реалізовані для списків в C#? В чому між ними принципова різниця?
4. Який інтерфейс має підтримувати клас для можливості порівняння його об'єктів між собою по замовчуванню? Який метод при цьому потрібно додатково реалізувати?
5. Як реалізувати сортування об'єктів класу за різними критеріями з використанням вкладених класів? Який інтерфейс має підтримувати вкладений клас для можливості порівняння об'єктів основного класу між собою?

Завдання.

1. **Вдоскональте зовнішній вигляд головної кнопкової форми комплексу форм, розроблених на попередній лабораторній роботі, забезпечивши можливість впорядкування форм за різними критеріями під час їх розміщення каскадом.** Для цього доповніть головну кнопкову форму групою взаємозалежних перемикачів для сортування форм. З цією метою виберіть на панелі елементів відокремлену групу з надписом *GroupBox* та відмітьте у формі прямокутну область для розміщення відповідного елемента керування. Після цього виберіть на панелі елементів перемикач *RadioButton* та натягніть у створеній групі декілька перемикачів для сортування зображених об'єктів за різними критеріями. Серед критеріїв сортування має обов'язково бути за *часом створення*. Встановіть властивість *Checked* цього перемикача в значення *true*, а всіх інших – в значення *false*. Для чого встановлювати цю властивість? Встановіть назви перемикачам англійською мовою згідно їх призначення. Натягніть у створеній групі також прапорець *checkBox* та дайте йому назву *sortDesc* для сортування форм за спаданням.

Після цього вигляд головної кнопкової форми може бути, наприклад, таким:



2. Для забезпечення можливості сортування форм за темою лабораторної роботи скористайтеся розробленим делегатом з попередньої лабораторної роботи та врахуйте, що при сортуванні за спаданням достатньо змінити на протилежний порядок форм (реверсувати), впорядкованих за зростанням. Тоді процедура обробки натиснення кнопки для впорядкування форм каскадом та делегат набуде, наприклад, такого вигляду:

```
public int SortByTema(Form f1, Form f2)
{   if (f1 == null || f2 == null)
        throw new ArgumentException("Окремі аргументи не є формами");
    return String.Compare((f1 as ILabWork).GetTemaLabWork(),
                          (f2 as ILabWork).GetTemaLabWork());
}

private void button5_Click(object sender, EventArgs e)
{   //сортування списку форм
    if (sortTema.Checked)
        listForm.Sort(SortByTema);
    if (sortDesk.Checked)
        listForm.Reverse();
    //вивід форм каскадом
    int left = 200, top = 200;
    for (int i = 0; i < listForm.Count; i++)
        if (!listForm[i].IsDisposed)
        {   listForm[i].Left = left;
            left += 20;
            listForm[i].Top = top;
            top += 15;
            listForm[i].Focus();
        }
}
```

3. **З метою забезпечення можливості сортування форм за часом створення по замовчуванню:**
   3.1. **Реалізуйте збереження моменту часу створення форми в одній з її властивостей, яка не використовується (наприклад, *Tag*) в процедурі створення форм:**


```
private void createForm(int index)
{   Form f = null;
    switch (index)
    {case 1: f = new FormCalc();
             break;
             ...
    }
    f.Tag = DateTime.Now.Ticks.ToString();
    listForm.Add(f);
    f.Show();
}
```

3.2. **Для забезпечення можливості порівняння створюваних форм між собою по замовчуванню реалізуйте підтримку ними інтерфейсу *IComparable*. З цією метою створіть новий клас, породжений від класу *Form* з підтримкою цього інтерфейсу та реалізуйте в ньому метод порівняння поточного об'єкта з зовнішнім за значенням, збереженим у попередньому підпункті:**

```
public class MForm : Form, IComparable
{
    int IComparable.CompareTo(object c)
    {   long t1 = Convert.ToInt64(this.Tag);
        long t2 = Convert.ToInt64(((Form)c).Tag);
        if (t1 < t2)
            return -1;
        if (t1 > t2)
            return 1;
        return 0;
    }
}
```

3.3. **В імпортованих формах *всіх* попередніх лабораторних робіт (*FormCalc, FormString* та інших) змініть батьківський клас на клас попереднього підпункту, забезпечивши цим самим підтримку інтерфейсу *IComparable*, наприклад, так:**

```
public partial class FormCalc : MForm, ILabWork
    ...
```

3.4. **Доповніть процедуру обробки натиснення кнопки для впорядкування форм каскадом обробкою перемикача з надписом за *часом створення*, викликаючи при цьому стандартний метод *Sort()* списку форм:**

```
private void button5_Click(object sender, EventArgs e)
{   //сортування списку форм
    if (sortTime.Checked)
        listForm.Sort();
    if (sortTema.Checked)
        listForm.Sort(SortByTema);
    if (sortDesk.Checked)
        listForm.Reverse();
    //вивід форм каскадом
    int left = 200, top = 200;
    for (int i = 0; i < listForm.Count; i++)
        if (!listForm[i].IsDisposed)
        {   listForm[i].Left = left;
            left += 20;
            listForm[i].Top = top;
            top += 15;
            listForm[i].Focus();
        }
}
```


4. З метою сортування форм за висотою:

4.1. **Опишіть *у класі головної кнопкової форми* вкладений клас з підтримкою інтерфейсу *IComparer*. Реалізуйте у цьому класі цілочисельний метод *Compare(Form f1, Form f2)* для підтримки інтерфейсу *IComparer*;**

4.2. **Доповніть процедуру обробки натиснення кнопки для впорядкування форм каскадом обробкою перемикача з надписом за *висотою*, викликаючи при цьому метод сортування списку форм з новим екземпляром вкладеного класу:**


```
public class SortByHeight : IComparer<Form>
{  public int Compare(Form f1, Form f2)
    {
        if (f1 == null || f2 == null)
            return 0;
        if (f1.Height < f2.Height)
            return -1;
        if (f1.Height > f2.Height)
            return 1;
        return 0;
    }
}
```

```
private void button5_Click(object sender, EventArgs e)
{   //сортування списку форм
    if (sortTime.Checked)
        listForm.Sort();
    if (sortTema.Checked)
        listForm.Sort(SortByTema);
    if (sortHeight.Checked)
        listForm.Sort(new SortByHeight());
    if (sortDesk.Checked)
        listForm.Reverse();
    //вивід форм каскадом
    int left = 200, top = 200;
    for (int i = 0; i < listForm.Count; i++)
        if (!listForm[i].IsDisposed)
        {   listForm[i].Left = left;
            left += 20;
            listForm[i].Top = top;
            top += 15;
            listForm[i].Focus();
        }
}
```

5. Самостійно забезпечте сортування форм за шириною та за заголовком (властивість *Text*) під час впорядкування каскадом.

Лабораторна робота № 20

Тема. Створення візуальних таблиць. Зберігання даних візуальних таблиць у файлах.

Мета. Формування вмінь і навиків створення та використання візуальних таблиць. Закріплення вмінь і навиків використання класів, об'єктів, підпрограм, функцій вводу-виводу та обробки рядків. Застосування вмінь і навиків програмування алгоритмів лінійної, розгалуженої та циклічної структури.

Контрольні запитання.

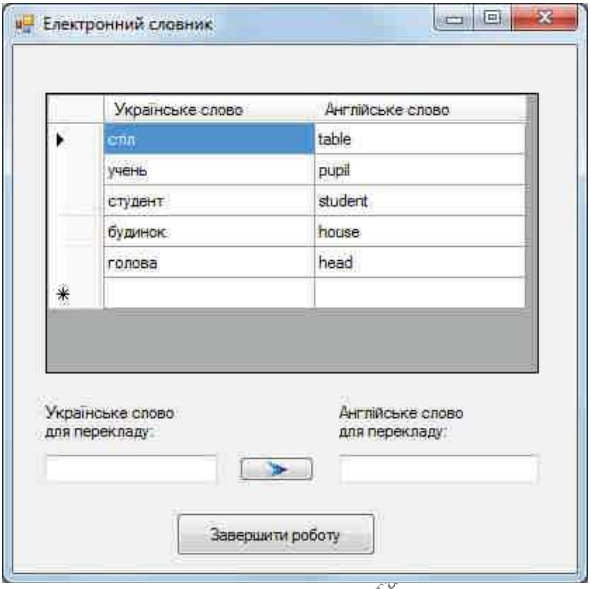
1. Як створити візуальну таблицю?
2. Як вказати кількість стовпців візуальної таблиці? Як змінити характеристики стовпців?
3. Як змінити кількість рядків візуальної таблиці?
4. Як звертаються до даних візуальної таблиці з модуля форми? Як індексуються елементи цього масиву?
5. Для яких подій форми необхідно задати процедури їх обробки, щоб забезпечити зберігання та зчитування даних візуальної таблиці з файлу?

Завдання.

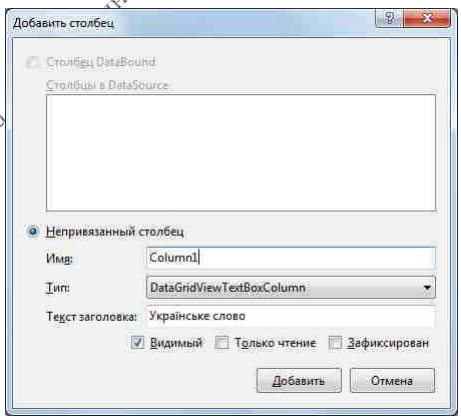
1. Створіть новий проект, а в ньому – візуальну таблицю для зберігання записів двомовного словника. Забезпечте для словника можливості редагування вмісту, доповнення та знищення його записів і виконання пошуку термінів.

Вимоги до програми:

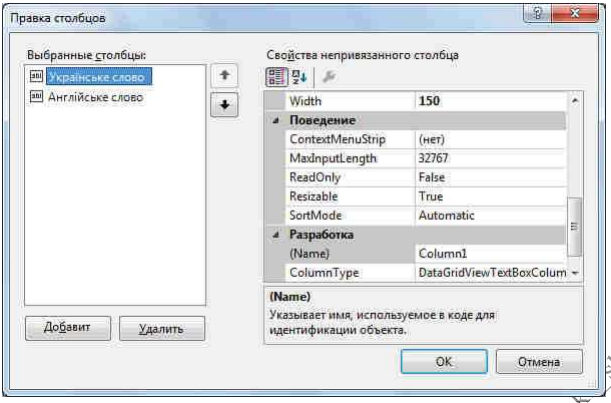
- записи словника редагувати безпосередньо у візуальній таблиці типу *DataGridView*;
- новий запис у словник вставляти в кінці таблиці;
- вилучення проводити для поточного запису;
- для введення терміну і відображення результатів його пошуку у словнику для кожної з мов створити два редагованих поля та відповідні кнопки між ними, щоб форма словника набула, наприклад, такого вигляду:



Для цього розмістіть в режимі конструктора форми у верхній її частині елемент керування типу *DataGridView* та дайте йому ім'я *DG1*. Решту елементів керування створіть самостійно. З метою відображення у таблиці *DG1* двох стовпців оберіть в її контекстному меню *Правка столбцов* та напишіть у ньому кнопку *Добавить*. Не змінюючи ім'я стовпця, вкажіть його заголовок *Українське стово*, як на рисунку нижче, та натисніть кнопку *Добавить*.



Самостійно створіть другий стовбець *Англійське слово*, щоб вікно редагування стовпців було, як на рисунку нижче, та натисніть *OK*.



Для збереження/зчитування даних словника розробіть такі приватні процедури:

```
private void WriteFile()
{   if (MessageBox.Show("Зберегти зміни в словнику?", "Увага!",MessageBoxBu
                        MessageBoxIcon.Exclamation) != DialogResult.Yes)
     return;
    string[] masStr = new string[DG1.RowCount*2];
    int i = 0;
    for (int j = 0; j < DG1.RowCount; j++)
    {masStr[i++] = (string)DG1[0,j].Value;
     masStr[i++] = (string)DG1[1,j].Value;
    }
    File.WriteAllLines("dictionary.txt", masStr); }

private void ReadFile()
{   string[] masStr;
    if (!File.Exists("dictionary.txt"))
        return;
    masStr=File.ReadAllLines("dictionary.txt");
    DG1.RowCount = masStr.Length / 2;
    int i = 0;
    for (int j = 0; j < DG1.RowCount; j++)
    {DG1[0, j].Value=masStr[i++];
     DG1[1, j].Value=masStr[i++];
    } }
```

Забезпечте виклик цих процедур з процедур обробки подій форми *FormClosed* та *Load*, наприклад, так:

```
private void Form1_Load(object sender, EventArgs e)
{ReadFile();
}

private void Form1_FormClosed(object sender, FormClosedEventArgs e)
{WriteFile();
}
```

А процедура обробки натиснення кнопки перекладу з української мови на англійську може бути такою:

```
private void button1_Click(object sender, EventArgs e)
{  if (textUA.Text == "")
    {MessageBox.Show("Введіть українське слово для перекладу!", "Увага!"
                     MessageBoxButtons.OK, MessageBoxIcon.Exclamation);
     textUA.Focus();
     return;
    }
    for (int j = 0; j < DG1.RowCount; j++)
     if (String.Compare((string)DG1[0, j].Value, textUA.Text,true) == 0)
      {textEN.Text = (string)DG1[1, j].Value;
       return;
      }
    textEN.Text = "Таке слово у словнику відсутнє"; }
```

2. Створіть новий проект з візуальною таблицею для зберігання даних списків по варіанту. Забезпечте для таблиці можливості редагування вмісту, доповнення та знищення записів і виконання відповідного пошуку.

Вимоги до програми:

- дані редагувати безпосередньо у візуальній таблиці;
- новий запис у таблицю вставляти в кінці таблиці;
- вилучення проводити для поточного запису;
- для введення параметрів пошуку створити **редаговані** поля.

Варіанти:

1. Список студентів містить їх прізвища, імена та по батькові. Знайти дані студентів з вказаним прізвищем;
2. Список студентів містить їх прізвища, імена та по батькові. Знайти дані студентів з вказаним ім'ям;
3. Список студентів містить їх прізвища, імена, по батькові та абревіатури груп. Знайти дані студентів вказаної групи;
4. Заданий список, що містить перелік прізвищ студентів і назви вулиць, на яких вони проживають. Знайти дані студентів з вказаної вулиці;

5. Список студентів містить їх прізвища, імена та по батькові. Знайти дані студентів з вказаним значення по батькові;
6. Список містить прізвища студентів і назви улюблених предметів кожного з них. Знайти дані студентів, для яких улюбленим є вказаний предмет;
7. Список містить прізвища студентів та суми отриманих балів кожним з них. Знайти дані студентів з вказаною сумою балів;
8. Список містить перелік футбольних команд та кількість очок, набраних у чемпіонаті кожною з них. Знайти команди, які набрали вказану кількість очок;
9. Список співробітників містить їх прізвища, імена, по батькові і номери їх домашніх телефонів. Знайти телефони співробітника з вказаним прізвищем;
10. Заданий список, що містить перелік прізвищ студентів і назви вулиць, на яких вони проживають. Знайти дані студентів з вказаним прізвищем;
11. Скласти програму-словник термінів з інформатики. Знайти пояснення вказаного терміна;
12. Список містить відомості про автомобілі: їх марки, номери і прізвища власників. Знайти дані автомобілів вказаної марки;
13. Список містить відомості про автомобілі: їх марки, номери і прізвища власників. Знайти всі автомобілі за вказаним прізвищем власника;
14. Список містяться відомості про кожну книгу: прізвище автора, рік та назву видання, Знайти назви книг, виданих після вказаного року;
15. Список містить дані про кожну партію експортованих товарів: назву, країну, що імпортує товар та об'єм партії, що поставляється в штуках. Знайти товари, які імпортує вказана країна.

Лабораторна робота № 21

Тема. Використання структур та діалогових вікон для обробки записів візуальних таблиць.

Мета. Формування вмінь і навиків створення та використання структур та діалогових вікон для обробки записів візуальних таблиць. Закріплення вмінь і навиків використання модулів, класів, об'єктів, візуальних компонентів, функцій вводу-виводу та обробки рядків. Застосування вмінь і навиків програмування алгоритмів лінійної, розгалуженої та циклічної структури.

Контрольні запитання.

1. Для чого і як створюються структури?
2. Чому для обробки записів таблиць доцільно використовувати діалогові вікна? Чим діалогові вікна відрізняються від звичайних та спливаючих?
3. Як доцільно передавати початкові значення елементів керування у діалогові вікна? Чому?
4. Як зчитати дані з діалогового вікна? Коли це відбувається?
5. Коли та з якою метою доцільно використовувати файли в програмуванні?
6. Які типи файлів використовуються в C#? У чому переваги та недоліки кожного типу файла стосовно форматів записів та швидкості обробки?

Завдання.

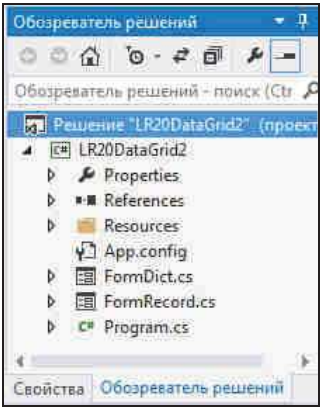
1. Забезпечте редагування даних словника попередньої лабораторної роботи лише за допомогою діалогових вікон. Для цього забороніть редагування даних відповідної візуальної таблиці безпосередньо та доповніть її форму кнопками для введення нових, редагування існуючих та видалення записів.

Вимоги до програми:

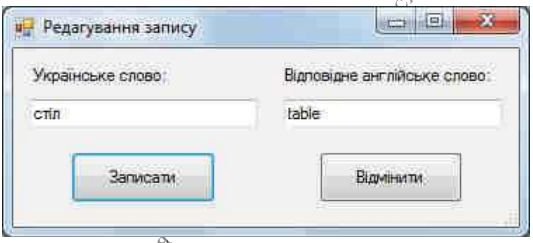
- для опрацювання даних одного запису таблиці словника описати та використати відповідну структуру;
- для редагування та створення нових записів таблиці створити окрему форму;
- значення комірок візуальної таблиці заносити/зчитувати з запису користувача з використанням оператора присвоєння.

Хід роботи:

1. Для доповнення/редагування даних таблиці двомовного словника доповніть її проект новою формою. Змініть ім'я її файла та класу на *FormRecord.* Перейменуйте також форму і клас двомовного словника на *FormDict*. Тоді вікно оглядача рішень набуде такого вигляду:



2. Самостійно розробіть в режимі конструктора вигляд форми *FormRecord*, аналогічний наведеному нижче. Перейменуйте перше поле введення на *wordUA*, а друге – на *wordEN*, першу кнопку – на *buttonSave*, а другу – на *buttonClose*;



3. Забезпечте неможливість введення пробілів у поля вводу. Для цього створіть таку процедуру обробки події *KeyPress* для першого поля вводу:

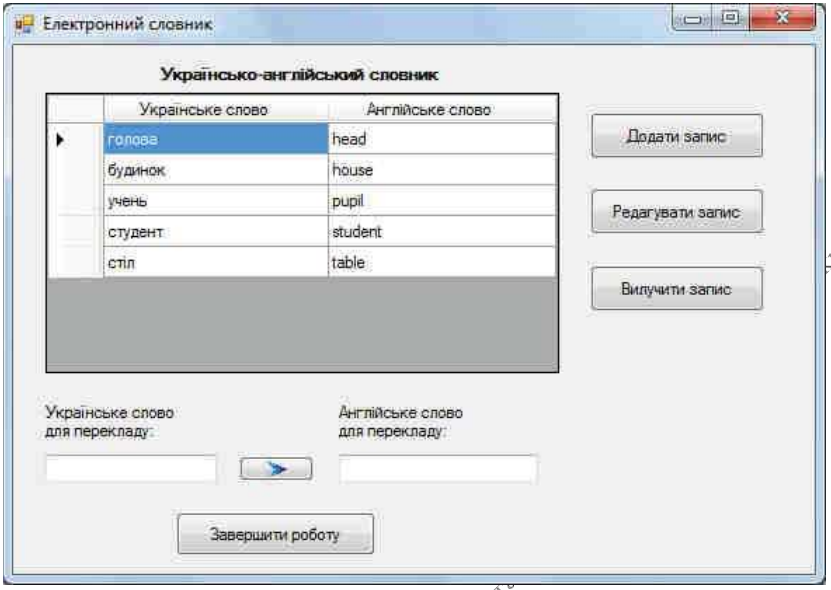
```
private void wordUA_KeyPress(object sender, KeyPressEventArgs e)
{ if (e.KeyChar==32)
    e.Handled=true;
}
```

Що в ній означає код символа *32*? Оберіть цю ж процедуру обробки для події *KeyPress* другого поля вводу;

4. У файлі коду форми *FormRecord* створіть публічну структуру *recordDict* для зберігання запису таблиці словника. Доповніть конструктор форми *FormRecord* параметром типу структури *recordDict* для передачі початкових значень та забезпечте їх запис у поля редагування. В класі цієї форми створіть змінну типу цієї структури з метою повернення введених значень та забезпечте присвоєння значень її полям при натисненні кнопки *buttonSave*. В процедурах обробки натиснення кнопок встановіть відповідні значення для властивості *DialogResult*. Тоді код форми буде приблизно таким:

```
public struct recordDict
{
    public string ukrWord, engWord;
}

public partial class FormRecord : Form
{
    public recordDict resultRecord;

    public FormRecord(recordDict start)
    { InitializeComponent();
      wordUA.Text=start.ukrWord;
      wordEN.Text=start.engWord;
    }

    private void buttonSave_Click(object sender, EventArgs e)
    { resultRecord.ukrWord = wordUA.Text;
      resultRecord.engWord = wordEN.Text;
      this.DialogResult = DialogResult.OK;
      Close();
    }

    private void buttonClose_Click(object sender, EventArgs e)
    { this.DialogResult = DialogResult.Cancel;
      Close();
    }

    private void wordUA_KeyPress(object sender, KeyPressEventArgs e)
    { if (e.KeyChar==32)
       e.Handled=true;
    }
}
```

5. Відкрийте форму *FormDict* в режимі конструктора та забороніть безпосереднє редагування даних візуальної таблиці. Для цього у вікні властивостей таблиці *DG1* забороніть доповнення і вилучення її записів, встановивши значення властивостей *AllowUserToAddRows* і *AllowUserToDeleteRows* рівними *false*, та забезпечте лише читання даних, задавши значення властивості *ReadOnly* рівним *true*;
6. Доповніть цю форму трьома кнопками для додавання, редагування і вилучення записів словника, розмістивши їх біля таблиці, як на рисунку нижче:



В процедурах обробки натиснення цих кнопок забезпечте корегування даних таблиці словника з використанням допоміжної форми, наприклад, так:

```
private void buttonAdd_Click(object sender, EventArgs e)
{recordDict R = new recordDict();
 FormRecord F = new FormRecord(R);
 if (F.ShowDialog() != DialogResult.OK)
  return;
 DG1.RowCount++;
 DG1[0, DG1.RowCount - 1].Value = F.resultRecord.ukrWord;
 DG1[1, DG1.RowCount - 1].Value = F.resultRecord.engWord;
 DG1.CurrentCell = DG1[0, DG1.RowCount - 1];
}
```

```
private void buttonEdit_Click(object sender, EventArgs e)
{if (DG1.CurrentRow == null)
  {MessageBox.Show("Не обрано запис для редагування!", "Увага!",
                   MessageBoxButtons.OK, MessageBoxIcon.Exclamation);
    return;
  }
 recordDict R;
 int row = DG1.CurrentRow.Index;
 R.ukrWord = (string)DG1[0,row].Value;
 R.engWord = (string)DG1[1, row].Value;
 FormRecord F = new FormRecord(R);
 if (F.ShowDialog() != DialogResult.OK)
    return;
 DG1[0, row].Value = F.resultRecord.ukrWord;
 DG1[1, row].Value = F.resultRecord.engWord;
}
private void buttonDelete_Click(object sender, EventArgs e)
{if (DG1.CurrentRow != null)
  DG1.Rows.Remove(DG1.CurrentRow);
 else
  MessageBox.Show("Не обрано запис для вилучення!", "Увага!",
                  MessageBoxButtons.OK, MessageBoxIcon.Exclamation);
}
```

2. Створіть новий проект з візуальною таблицею для зберігання даних списків по варіанту. Забезпечте для таблиці можливості редагування вмісту, доповнення та знищення записів з окремого діалогового вікна, зберігання/зчитування даних з текстового файла і виконання завдання по варіанту.

Вимоги до програми:

- для зберігання полів, що характеризують один об'єкт описати та використати структуру;
- дані таблиці зберігати в текстовому файлі;
- значення полів записів опрацьовувати з використанням оператора присвоєння, забезпечивши відповідну перевірку коректності даних;
- введення параметрів та відображення результатів обробки виконайте в елементах керування під візуальною таблицею.

Варіанти:

1. Список учнів містить їх прізвища, імена та по батькові. Вияснити, чи є в даному не впорядкованому за алфавітом списку учнів задане прізвище. Якщо є, то вказати порядкові номери таких прізвищ, відповідні імена та по батькові;

2. Список учнів містить їх прізвища, імена та по батькові. Вияснити, чи є в даному не впорядкованому за алфавітом списку учнів однофамільці. При виявленні однофамільців надрукувати їх дані;
3. Список учнів містить їх прізвища, імена, по батькові та абревіатури класів. Визначити, в яких класах зустрічається прізвище *Поліщук* (списки класів не впорядковані за алфавітом);
4. Заданий список, що містить перелік прізвищ учнів і назви вулиць, на яких вони проживають. Визначити, на якій з вулиць пропиває найбільша кількість учнів;
5. Список учнів містить їх прізвища, імена та по батькові. Визначити, яке ім'я у класі зустрічається найчастіше;
6. Список містить прізвища учнів і назви улюблених предметів кожного з них. Визначити, який з предметів вказаний найбільшу кількість разів і для скількох учнів він є улюбленим;
7. Список містить прізвища учнів та суми отриманих балів кожним з них. Впорядкувати список за спаданням суми отриманих балів;
8. Список містить перелік футбольних команд та кількість очок, набраних у чемпіонаті кожною з них. Вказати назву команди-чемпіона. При наявності кількох команд з максимальною кількістю набраних очок надрукувати назви всіх таких команд;
9. Список співробітників містить їх прізвища, імена, по батькові і номери їх домашніх телефонів. Знайти телефон співробітника за прізвищем або вказати на відсутність даних у списку. Якщо для співробітника зазначено декілька телефонів, то вивести їх всі;
10. Заданий список, що містить перелік прізвищ учнів і назви вулиць, на яких вони проживають. Вказати, чи є серед учнів такі, що проживають по вулиці *Соборній*;
11. Заданий список, що містить перелік прізвищ учнів і назви вулиць, на яких вони проживають. Вказати кількість і прізвища учнів, що проживають по вулиці *Дубенський*.
12. Список містить відомості про автомобілі: їх марки, номери і прізвища власників. Вказати кількість автомобілів даної марки;
13. Список містить відомості про автомобілі: їх марки, номери і прізвища власників. Вказати прізвища власників і номери автомобілів даної марки;
14. Список містяться відомості про кожну книгу: прізвище автора, рік та назву видання, Знайти назви книг даного автора;
15. Список містить дані про кожну партію експортованих товарів: назву, країну, що імпортує товар та об'єм партії, що поставляється в штуках. Вказати загальний об'єм експорту даного товару.

Лабораторна робота № 22

Тема. Узагальнення класів. Класи-колекції. Використання полів зі списками, списків, груп перемикачів і прапорців для фільтрування та сортування записів візуальних таблиць..

Мета. Формування вмінь і навиків створення та використання форм і їх компонентів. Закріплення вмінь і навиків використання властивостей та методів об'єктів. Застосування вмінь і навиків програмування алгоритмів лінійної, розгалуженої та циклічної структури.

Контрольні запитання

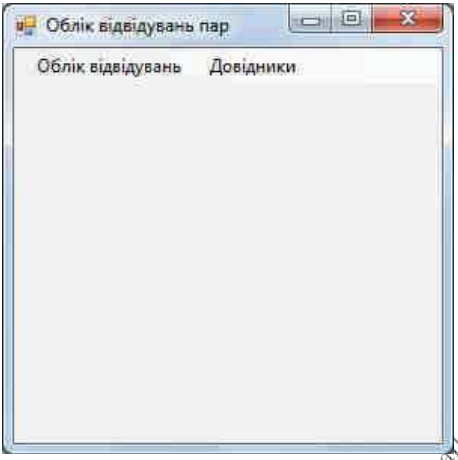
1. Як в проект Windows Form додати ще одну форму? Як змінити ім'я форми?
2. Як створити головну форму з горизонтальним меню? Як забезпечити вивід інших модальних форм при виборі пунктів горизонтального меню?
3. Які змінні описуються в методі, а які – в класі?
4. Які варіанти вибору елементів використовуються в списках?
5. Як створити взаємопов'язані перемикачі в формах? Як встановити вибір одного з цих перемикачів по замовчуванню?

Завдання

Створіть власний додаток з головною формою, яка має містити горизонтальне меню для виклику форм оперативної та умовно-постійної інформації в модальному режимі. Форми з даними мають демонструвати можливості використання полів зі списками, списків, груп перемикачів і прапорців для фільтрування та сортування записів візуальних таблиць. Якщо розробити власний додаток не вдається, то виконайте наступні завдання для створення додатку за зразком:

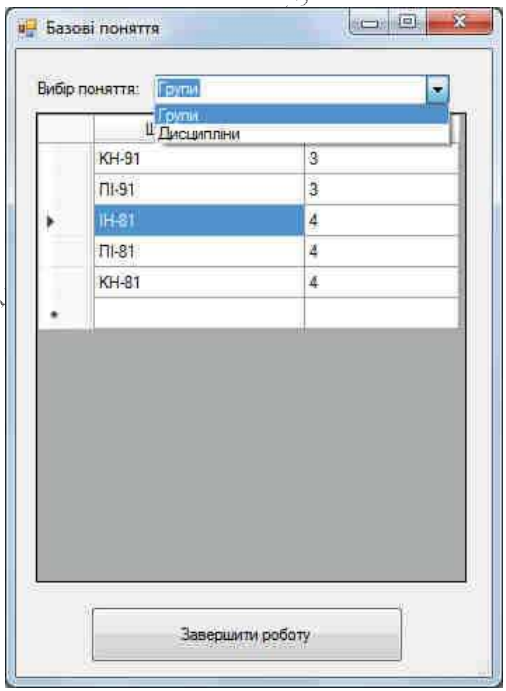
1. Для організації фільтрування та сортування записів візуальних таблиць створіть спочатку проект Windows Form та дайте йому назву, яка б містила ваше прізвище латинськими літерами та номер лабораторної без пробілів.

2. Створіть головну форму *Menu.cs* з горизонтальним меню за таким зразком:



3. Ознайомтеся з підрозділом по розробці полів зі списками (*ComboBox*) https://metanit.com/sharp/windowsforms/4.7.php навчального курсу по створенню Windows Forms на сайті https://metanit.com/.

4. Створіть форму *Terminy.cs* для введення базових понять (груп та предметів) в систему за таким зразком:



Забезпечте автоматичне збереження базових понять при закритті форми і їх завантаження – при відкритті. Код цієї форми може бути таким:

```
public partial class Terminy : Form
{
    string[,] dataGrupa = new string[0, 2];
    string[,] dataPredmet = new string[0, 2];
    int typTermin = 0;

    public Terminy()
    {
        InitializeComponent();
        CBVubir.SelectedIndex = 0;
    }

  private void writeDG1(int typ)
   {typTermin=typ;
    switch (typTermin)
    {
        case 1: DG1.RowCount = dataGrupa.GetLength(0) + 1;
            for (int i = 0; i < dataGrupa.GetLength(0); i++)
            {
                DG1[0, i].Value = dataGrupa[i, 0];
                DG1[1, i].Value = dataGrupa[i, 1];
            }
            break;
        case 2: DG1.RowCount = dataPredmet.GetLength(0) + 1;
            for (int i = 0; i < dataPredmet.GetLength(0); i++)
            {
                DG1[0, i].Value = dataPredmet[i, 0];
                DG1[1, i].Value = dataPredmet[i, 1];
            }
            break;
    }
   }

    private void readDG1()
    {switch(typTermin)
      {
        case 1: dataGrupa = new string[DG1.RowCount - 1, 2];
                for (int j = 0; j < DG1.RowCount - 1; j++)
                   {dataGrupa[j, 0] = "" + DG1[0, j].Value;
                    dataGrupa[j, 1] = "" + DG1[1, j].Value;
                   }
                break;
        case 2: dataPredmet = new string[DG1.RowCount - 1, 2];
```

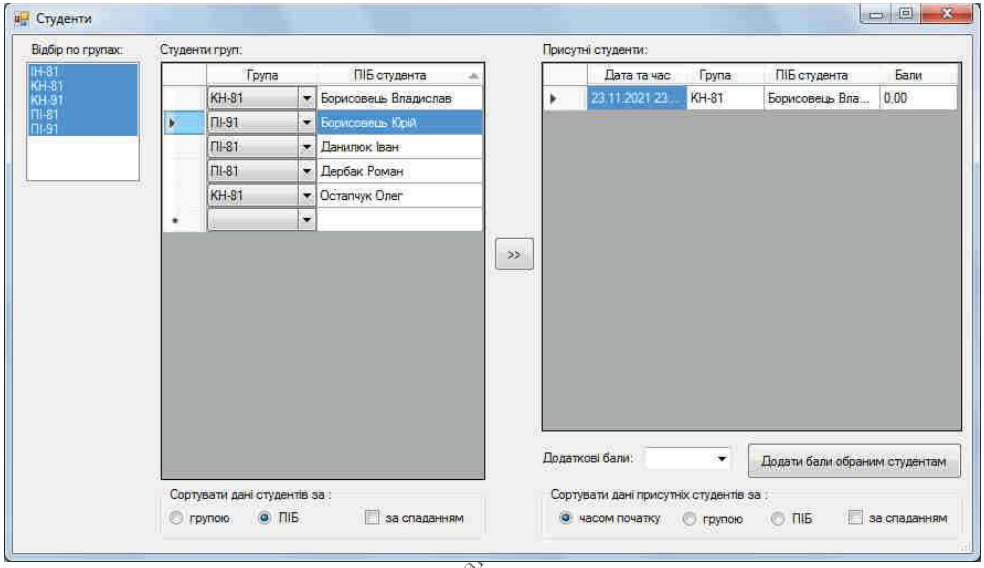
```
                    for (int j = 0; j < DG1.RowCount - 1; j++)
                       {dataPredmet[j, 0] = "" + DG1[0, j].Value;
                        dataPredmet[j, 1] = "" + DG1[1, j].Value;
                       }
                    break;
                }
            }

            private void writeFile()
            {
                readDG1();
                if (dataGrupa.Length == 0 && dataPredmet.Length == 0)
                {
                    File.Delete("Terminy.txt");
                    return;
                }
                string[] masStr = new string[(dataGrupa.GetLength(0) +
dataPredmet.GetLength(0))*3];
                int indexStr = 0;
                for (int j = 0; j < dataGrupa.GetLength(0); j++)
                {
                    masStr[indexStr++] = "1";
                    for (int i = 0; i < 2; i++)
                        masStr[indexStr++] = ""+dataGrupa[j, i];
                }
                for (int j = 0; j < dataPredmet.GetLength(0); j++)
                {
                    masStr[indexStr++] = "2";
                    for (int i = 0; i < 2; i++)
                        masStr[indexStr++] = ""+dataPredmet[j, i];
                }
                File.WriteAllLines("Terminy.txt", masStr);
            }

            private void readFile()
            {
                if (!File.Exists("Terminy.txt"))
                {
                    dataGrupa = new string[0, 2];
                    dataPredmet = new string[0, 2];
                    return;
                }
                string[] masStr = File.ReadAllLines("Terminy.txt");
                int rowCount = masStr.Length / 3;
```

```
            int countGrupa = 0;
            for (int j = 0; j < masStr.Length; j+=3)
                if (masStr[j] == "1")
                    countGrupa++;
                else
                    break;
            int countPredmet = rowCount - countGrupa;
            dataGrupa = new string[countGrupa, 2];
            int indexStr = 0;
            for (int j = 0; j < countGrupa; j++)
            {
                indexStr++;
                for (int i = 0; i < 2; i++)
                    dataGrupa[j, i] = masStr[indexStr++];
            }
            dataPredmet = new string[countPredmet, 2];
            for (int j = 0; j < countPredmet; j++)
            {
                indexStr++;
                for (int i = 0; i < 2; i++)
                    dataPredmet[j, i] = masStr[indexStr++];
            }
        }

        private void button1_Click(object sender, EventArgs e)
        {
            Close();
        }

        private void Form1_FormClosing(object sender,
FormClosingEventArgs e)
        {
            writeFile();
        }

        private void Form1_Load(object sender, EventArgs e)
        {
            readFile();
            if (CBVubir.SelectedIndex == 0)
                writeDG1(1);
            else
                writeDG1(2);
        }
```

```
        private void CBVubir_SelectedIndexChanged(object sender,
EventArgs e)
        {
            if (CBVubir.SelectedIndex == 0)
            {
                DG1.Columns[0].HeaderText="Шифр групи";
                DG1.Columns[1].HeaderText = "Курс";
                if (typTermin != 1)
                {
                    readDG1();
                    writeDG1(1);
                }
            }
            else
            {
                DG1.Columns[0].HeaderText = "Назва дисципліни";
                DG1.Columns[1].HeaderText = "К-ть годин";
                if (typTermin != 2)
                {
                    readDG1();
                    writeDG1(2);
                }
             }
        }
    }
```

Забезпечте виклик цієї форми в модальному режимі з головної форми при виборі пункту меню *Довідники – Базові поняття.*

5. Створіть форму *Students.cs* за зразком, наведеним нижче. Забезпечте виклик цієї форми в модальному режимі з головної форми при виборі пункту меню *Облік відвідувань*.



Код цієї форми може бути таким:

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

    private void writeFile()
    {
        string[] masStr = new string[(DG1.RowCount-1) * 2];
        int indexStr = 0;
        for (int j = 0; j < DG1.RowCount-1; j++)
            for (int i = 0; i <= 1; i++)
                masStr[indexStr++] = ("" + DG1[i,
j].Value).ToString();
        File.WriteAllLines("student.txt", masStr);
    }

    private void readFile()
    {
        if (!File.Exists("student.txt"))
```

```
                return;
            string[] masStr = File.ReadAllLines("student.txt");
            DG1.RowCount = masStr.Length / 2+1;
            int indexStr = 0;
            for (int j = 0; j < DG1.RowCount-1; j++)
                for (int i = 0; i <= 1; i++)
                {
                    if (i == 0)
                        if (Group.Items.IndexOf(masStr[indexStr])
< 0)
                            masStr[indexStr] = "";
                    DG1[i, j].Value = masStr[indexStr++];
                }

        }

        private void Students_Load(object sender, EventArgs e)
        {
            if (File.Exists("Terminy.txt"))
            {string[] masStr = File.ReadAllLines("Terminy.txt");
             for (int j = 0; j < masStr.Length; j += 3)
                if (masStr[j] == "1")
                    Group.Items.Add(masStr[j+1]);
                else
                    break;
            }
            if (Group.Items.Count==0)
            {
                MessageBox.Show("Спочатку введіть групи в
довіднику базових понять",
                                "Увага!", MessageBoxButtons.OK,
MessageBoxIcon.Exclamation);
                Close();
                return;
            }
            Group.Sorted = true;
            foreach(string grupa in Group.Items)
               ((DataGridViewComboBoxColumn)
DG1.Columns[0]).Items.Add(grupa);
            readFile();
            for (int i = 0; i < Group.Items.Count; i++)
                Group.SetSelected(i, true);
        }
```

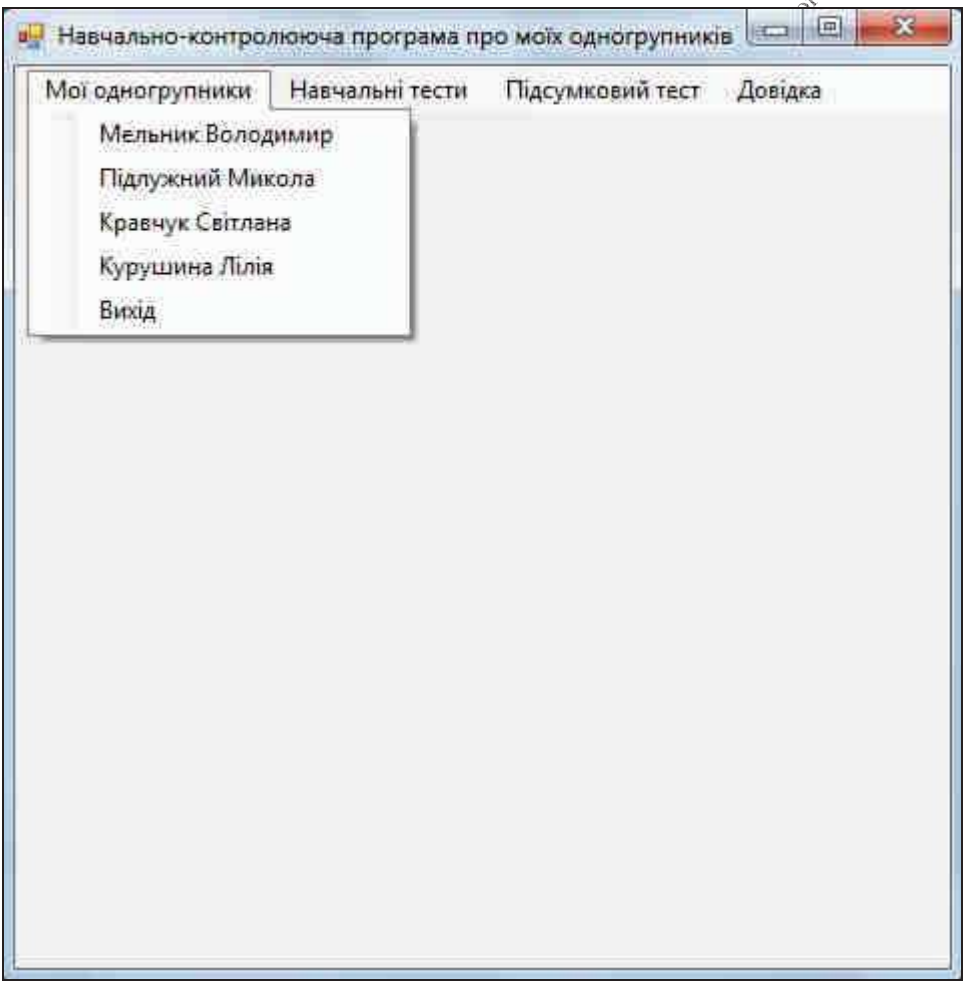
```
        private void Students_FormClosing(object sender,
FormClosingEventArgs e)
        {
            writeFile();
        }

        private void Group_SelectedIndexChanged(object sender,
EventArgs e)
        {
            for (int j = 0; j < DG1.RowCount-1; j++)
                if (Group.SelectedItems.IndexOf("" + DG1[0,
j].Value) < 0)
                    DG1.Rows[j].Visible = false;
                else
                    DG1.Rows[j].Visible = true;
        }

        private void button1_Click(object sender, EventArgs e)
        {for (int j=0; j<DG1.SelectedRows.Count; j++)
          {
            DG2.RowCount++;
            DG2[0, DG2.RowCount - 1].Value = DateTime.Now;
            DG2[1, DG2.RowCount - 1].Value =
DG1.SelectedRows[j].Cells[0].Value;
            DG2[2, DG2.RowCount - 1].Value =
DG1.SelectedRows[j].Cells[1].Value;
            DG2[3, DG2.RowCount - 1].Value = "0,00";
          }
        }

        private void button2_Click(object sender, EventArgs e)
        {
            if (""+plusBal.Text).Trim()=="")
            {
                MessageBox.Show("Оберіть додаткові бали",
"Увага!",
                                MessageBoxButtons.OK,
MessageBoxIcon.Exclamation);
                plusBal.Focus();
                return;
            }
            if (DG2.SelectedRows.Count==0)
            {
                MessageBox.Show("Виберіть студентів, яким
додаються бали", "Увага!",
```

```
                                    MessageBoxButtons.OK,
MessageBoxIcon.Exclamation);
                DG2.Focus();
                return;
            }
            for (int j = 0; j < DG2.SelectedRows.Count; j++)
                DG2.SelectedRows[j].Cells[3].Value =

(Convert.ToDouble(DG2.SelectedRows[j].Cells[3].Value) +

Convert.ToDouble(plusBal.Text)).ToString();

        }
    }
```

6. Доповніть цю форму групами взаємозалежних перемикачів для сортування даних студентів. З цією метою для кожної групи виберіть на панелі елементів відокремлену групу з надписом *GroupBox* та відмітьте у формі прямокутну область для розміщення відповідного елемента керування. Після цього виберіть на панелі елементів перемикач *RadioButton* та натягніть у створеній групі декілька перемикачів для сортування зображених об'єктів за різними критеріями. Встановіть властивість *Checked* одного з перемикачів кожної групи у значення *true*, а всіх інших – в значення *false*. Для чого встановлювати цю властивість? Натягніть у кожній створеній групі для сортування за спаданням також прапорці *checkBox* та дайте їм назви *checkBoxDesc1* та *checkBoxDesc2* відповідно. Самостійно забезпечте функціонування цих елементів керування.

Завдання для розробки індивідуального навчально-дослідного проекту

1. Оберіть собі тему для розробки навчально-контролюючої програми. Створіть для її розробки новий проект. Стартову форму цього проекту перейменуйте на *FormMenu.cs*. Створіть у цій формі головне меню, передбачивши у ньому можливість вибору не менше трьох форм з теоретичними відомостями, не менше трьох навчальних тестів, підсумкового тесту та довідки (з відомостями про програму і про розробника). Наприклад, для програми про одногрупників це меню може виглядати так:



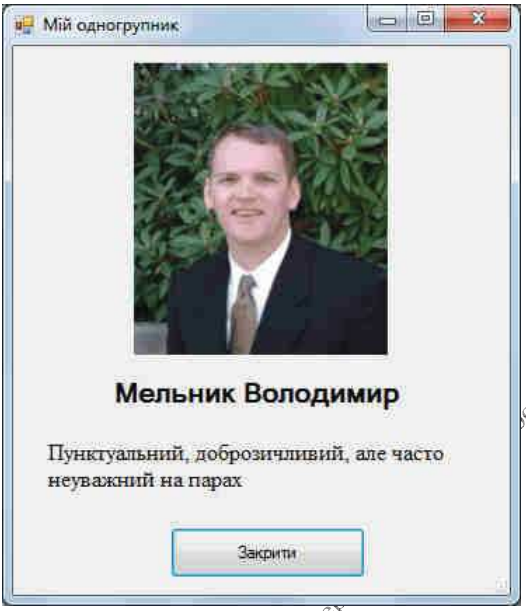
2. Вставте в ресурси проекту не менше трьох зображень для різних теоретичних відомостей.
3. Створіть у цьому проекті нову форму або імпортуйте сюди з проекту лабораторної роботи № 15 *Form2.cs* для відображення теоретичних відомостей. Перейменуйте її на *FormEducation.cs*. Забезпечте у цій формі відображення зображення і не менше двох текстових полів з описом теоретичних відомостей та різними характеристиками (назва зображення та тексти полів мають передаватися параметрами у форму). Забезпечте виклик цієї форми не менше ніж з трьох пунктів теоретичних відомостей головного меню з відповідними параметрами. Наприклад, в програмі про одногрупників конструктор цієї форми може виглядати так:

```
public partial class FormEducation : Form
 {public FormEducation(string Spivrob, string msg, string nameImg)
    {  InitializeComponent();
       this.Spivrob.Text = Spivrob;
       Message.Text = msg;
       System.Resources.ResourceManager rm =
            Properties.Resources.ResourceManager;
       Picture.Image = (Image)rm.GetObject(nameImg);
     }
  ...
 }        .
```

Код для створення об'єкта класу цієї форми при виборі першого підпункту першого пункту меню може бути таким:

```
private void тема1ToolStripMenuItem_Click(object sender, EventArgs e)
 {(new FormEducation("Мельник Володимир",
    "Пунктуальний, доброзичливий, але часто неуважний на парах",
"Foto1")).ShowDialog(); }
```

Тоді при виконанні програми вибір цього підпункту призведе до відображення такої форми:



Функціонування інших підпунктів першого пункту головного меню для створення аналогічних вікон з даними інших одногрупників забезпечується аналогічно.

4. Забезпечте відображення вікна кожних теоретичних відомостей лише один раз, як в лабораторній роботі № 16: якщо таке вікно не створено – його потрібно створити, якщо створено і згорнути – треба розгорнути, якщо не активне – сфокусувати. Для цього створіть у формі головного меню вказівки на кожну з форм теоретичних відомостей і модифікуйте процедуру виклику кожної з цих форм, наприклад, так:

```
public partial class FormMenu : Form
 {      Form TheorF1 = null, TheorF2 = null, TheorF3 = null;
        public FormMenu()
        {
            InitializeComponent();
        }

        private void тема1ToolStripMenuItem_Click(object sender,
EventArgs e)
        {  if (TheorF1 == null || TheorF1.IsDisposed)
            { TheorF1 = new FormEducation("Мельник Володимир",
                "Пунктуальний, доброзичливий, але часто неуважний
на парах", "Foto1");
              TheorF1.Show(); }
```

```
            else
             if (TheorF1.WindowState == FormWindowState.Minimized)
                 TheorF1.WindowState = FormWindowState.Normal;
             else
              if (!TheorF1.Focused)
               TheorF1.Focus(); }
 ... }
```

Конструювання та організація взаємодії форм навчальних тестів

5. З метою забезпечення в подальшому можливості отримання кількості набраних балів з кожного вікна тестування опишіть під кодом будь-якої форми чи в окремому файлі відповідний інтерфейс, наприклад, так:

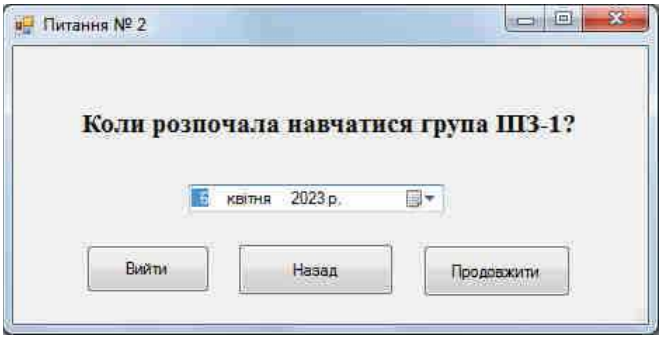
```
interface IQuationTest
 {double balQuestion(); }
```

6. Для підрахунку з різних форм загальної кількості пройдених питань, кількості набраних балів та максимально можливих балів, а також для відображення результатів кожного тесту створіть на початку статичного класу *Program* у файлі *Program.cs* відповідні статичні змінні та метод, наприклад, так:

```
internal static class Program
   {   public static int countQuestionTest1;
       public static double balTest1, maxBalTest1;

       public static void resTest1()
       {MessageBox.Show("За результатами першого тесту ви
відповіли на " +
         countQuestionTest1 + " питань і набрали\n" +
         balTest1.ToString() + " балів з " + maxBalTest1 + "
можливих", "Результати",
         MessageBoxButtons.OK, MessageBoxIcon.Information); }
     ... }
```

7. Створіть форми з різними елементами керування для окремих питань кожного навчального тесту. Наприклад, такі:

7.1. Форма для вибору дати для відповіді на питання може виглядати так:



Функція для підрахунку балів **описується в середині класу цієї форми**, яка аналізує рік, місяць та день обраної дати, може виглядати так:

```
public partial class Question3_2 : Form, IQuationTest
 { public Question3_2()
    {InitializeComponent(); }

   public double balQuestion()
    {double bal = 0;
     if (dateStart.Value.Year==2021 && dateStart.Value.Month == 9
&&
         dateStart.Value.Day == 1)
      {MessageBox.Show("Ви дали правильну відповідь та набрали 1
бал", "Питання № 2",
                       MessageBoxButtons.OK,
MessageBoxIcon.Information);
       bal = 1; }
      else
       {MessageBox.Show("Ви помилилися!\nГрупа розпочала навчання
1.09.2021\n\n"+
                       "Набрано " + bal.ToString() + " балів з
одного", "Питання # 2",
                       MessageBoxButtons.OK,
MessageBoxIcon.Information); }
      Program.countQuestionTest1++;
      Program.balTest1 += bal;
      Program.maxBalTest1 += 1;
      return bal; }
```

Кнопка *Вийти* має завершувати тестування, тобто виводити правильну відповідь на поточне питання, закривати поточну форму і виводити загальні результати за навчальний тест. Тому процедура обробки її натиснення може бути такою:

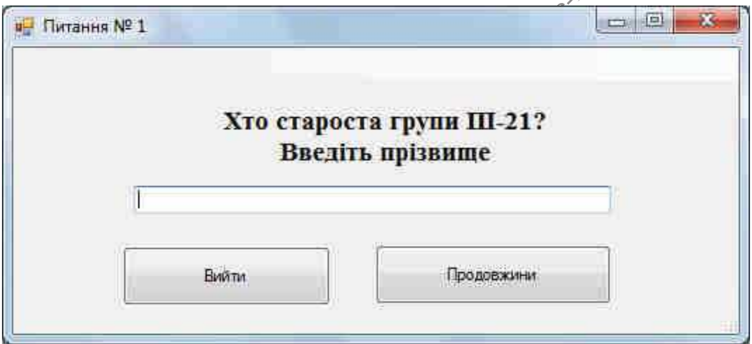
```
private void button4_Click(object sender, EventArgs e)
 {balQuestion();
  Close();
  Program.resTest1(); }
```

Кнопка *Продовжити* має виводити правильну відповідь на поточне питання, закривати поточну форму та створювати і виводити форму наступного питання:

```
private void button1_Click(object sender, EventArgs e)
 {balQuestion();
  Close();
  new Question3_3().Show(); }
```

Обробка натиснення кнопки *Назад* виконується аналогічно.

7.2. Форма для введення відповіді від користувача у текстовому полі може бути такою:



Тоді функція для підрахунку балів на це питання може виглядати так (тут *Starosta* – ім'я візуального об'єкта класу *TextBox*):

```
public double balQuestion()
 {double bal = 0;
  if (Starosta.Text.ToUpper()=="ТЕНЬКОВ")
   {MessageBox.Show("Ви дали правильну відповідь та набрали 1 
бал", "Питання № 2",
                          MessageBoxButtons.OK, 
MessageBoxIcon.Information);
     bal = 1; }
  else
   {MessageBox.Show("Ви помилилися!\nСтароста групи - 
Теньков\n\nНабрано " +
                          bal.ToString() + " балів з одного", 
"Питання № 2",
```

```
                        MessageBoxButtons.OK,
MessageBoxIcon.Information); }
      Program.countQuestionTest1++;
      Program.balTest1 += bal;
      Program.maxBalTest1 += 1;
      return bal; }
```

Кнопки переходу в цій та наступних формах пункту реалізуються, як в попередній формі.

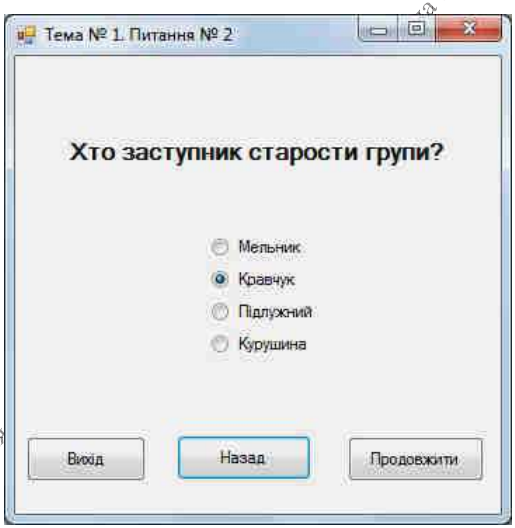
7.3. Форма з можливістю вибору декількох варіантів реалізується за допомогою прапорців класу *CheckBox* і може виглядати так:

Нехай правильна відповідь на це питання реалізується встановлення першого і другого прапорців. Тоді за встановлення кожного з цих прапорців і за зняття третього і четвертого прапорців будемо нараховувати 0.25 бала, щоб з одного боку за це питання можна було набрати максимум 1 бал, а з іншого при повністю неправильній відповіді – не отримати від'ємних балів. Функція для підрахунку балів на це питання може бути такою:

```
    public double balQuestion()
     {double bal = 0;
      if (check1.Checked && check2.Checked &&
          !check3.Checked && !check4.Checked)
       {MessageBox.Show("Ви дали правильну відповідь та набрали 1
бал", "Питання # 1",
                        MessageBoxButtons.OK,
MessageBoxIcon.Information);
        bal = 1; }
      else
       {if (check1.Checked) bal += 0.25;
```

```
        if (check2.Checked) bal += 0.25;
        if (!check3.Checked) bal += 0.25;
        if (!check4.Checked) bal += 0.25;
        MessageBox.Show("Ви помилилися!\nНа понад 60% пар
спізнюються Мельник "+
                  "та Кравчук!\n\nНабрано "+bal.ToString()+ "
балів з одного",
                  "Питання # 1", MessageBoxButtons.OK,
MessageBoxIcon.Information); }
      Program.countQuestionTest1++;
      Program.balTest1 += bal;
      Program.maxBalTest1 += 1; ;
      return bal; }
```

7.4. Форма з забезпеченням вибору одного варіанту з декількох заданих за допомогою перемикачів реалізується за допомогою об'єктів класу *RadioButton* і може бути такою:

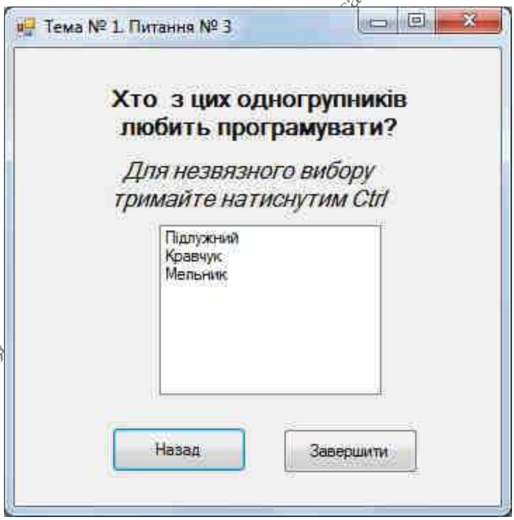


Правильна відповідь на таке питання реалізується вибором відповідного перемикача. Якщо задати третьому перемикачу з цієї форми ім'я *zastupnyk*, то функцію для підрахунку балів за це питання можна реалізувати так:

```
    public double balQuestion()
     {double bal = 0;
      if (zastupnyk.Checked)
       {MessageBox.Show("Ви дали правильну відповідь та набрали 1
бал", "Питання # 2",
```

```
                        MessageBoxButtons.OK,
MessageBoxIcon.Information);
        bal = 1; }
      else
       {MessageBox.Show("Ви помилилися!\nЗаступник старости групи
- Підлужна!" +
                 "\n\nНабрано " + bal.ToString() + " балів з
одного", "Питання # 2",
                MessageBoxButtons.OK, MessageBoxIcon.Information);
}
      Program.countQuestionTest1++;
      Program.balTest1 += bal;
      Program.maxBalTest1 += 1;
      return bal; }
```

7.5. І, нарешті, форма з можливістю незв'язного вибору декількох позицій у списку реалізується за допомогою об'єкта класу *ListBox*, в якому тексти позицій задаються в колекції *Items*, а сама можливість незв'язного вибору забезпечується встановлення у властивості *SelectionMode* значення *MultiExtended*. Ця форма може виглядати так:



Якщо правильна відповідь на питання цієї форми полягає у виборі першого та третього елементів списку, то функція для підрахунку набраних балів може бути такою:

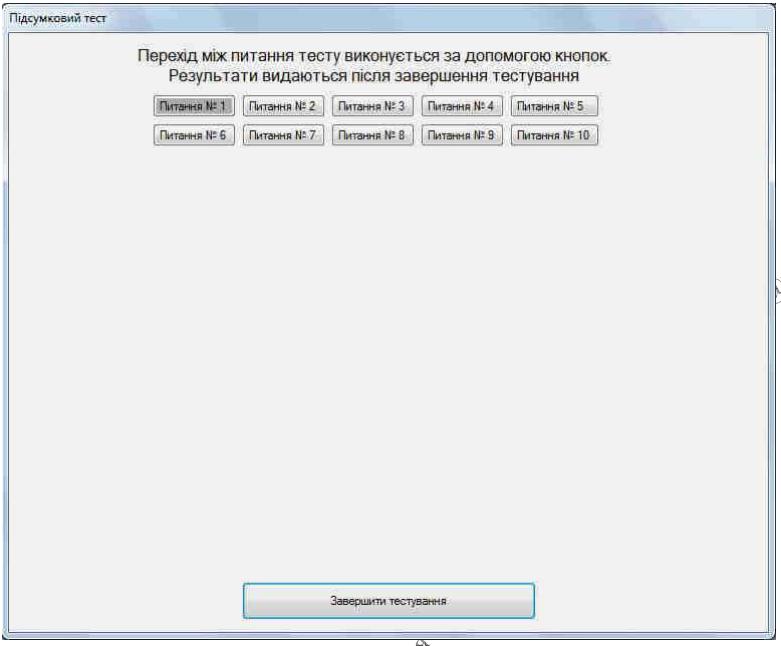
```
public double balQuestion()
{double bal = 0;
 if (listStudent.SelectedIndices.Count==2  &&
     listStudent.SelectedIndices[0]==0 &&
```

```
        listStudent.SelectedIndices[1]==2)
      {MessageBox.Show("Ви дали правильну відповідь та набрали 1
бал", "Питання № 3",
                        MessageBoxButtons.OK,
MessageBoxIcon.Information);
       bal = 1; }
     else
      {if (listStudent.GetSelected(0)) bal += 0.33;
       if (!listStudent.GetSelected(1)) bal += 0.33;
       if (listStudent.GetSelected(2)) bal += 0.34;
       MessageBox.Show("Ви помилилися!\nПрограмувати люблять
Підлужний та Мельник!" +
                       "\n\nНабрано " + bal.ToString() + " балів з
одного", "Питання № 3",
                       MessageBoxButtons.OK,
MessageBoxIcon.Information); }
     Program.countQuestionTest1++;
     Program.balTest1 += bal;
     Program.maxBalTest1 += 1;
     return bal; }
```

В цій функції *listStudent* – це ім'я об'єкта списку, властивість *SelectedIndices* – масив індексів обраних елементів, а *GetSelected* – це метод, що вказує на виділення елемента з вказаним індексом, хоча зрозуміло, що підрахувати набрані бали можна й за допомогою інших властивостей та методів списку.

Розробка форми та вкладених форм підсумкового тесту

8. Створіть форму з не менше, ніж десятьма перемикачами класу *RadioButton* у верхній чи лівій її частині для вибору питань підсумкового тесту. Для відображення цих перемикачів у вигляді пов'язаних кнопок встановіть для них у властивості *Apperance* значення *Button*. Задайте підписи цим кнопкам з зазначенням номера питання, а для властивості *Checked* значення *false* (тобто, по замовчуванню кнопки не натиснуті). В нижній частині форми створіть кнопку для завершення підсумкового тестування. Самостійно приховайте для цієї форми кнопку закриття. Виглядати ця форма може так:



9. Для зберігання вибраних відповідей користувача під час переходів між різними питаннями створіть в конструкторі форми підсумкового тесту не менше десяти форм окремих питань (по одній для кожної кнопки вибору) та збережіть вказівки на них в масиві вказівок на форми. Ці форми будуть приховуватися/відображатися при переходах між кнопками вибору, а закриватися – лише після виводу загальних результатів тестування. Для створення цих вкладених форм для окремих питань і зберігання вказівок на них в масиві реалізуйте також процедуру *createForm*, яка за індексом буде створювати відповідну форму, приховувати наявні в ній кнопки і межу для створення ефекту належності формі підсумкового тестування. Це можна реалізувати, наприклад, так:

```
public partial class FinalTest : Form
{   const int countQuestion = 10; // кількість питань
підсумкового тестування
    Form[] masForm = new Form[countQuestion];

    public FinalTest()
    {   InitializeComponent();
        for (int i = 0; i < countQuestion; i++)
            createForm(i); }

    private void createForm(int index)
```

```
        {   Form f=null;
            switch (index)
            { case 0:f = new Quation1_3(); break;
              case 1:f = new Quation1_1(); break;
              case 2:f = new Quation1_2(); break;
              ...
            }
            if (f != null)
            { foreach (Control c in f.Controls)
                    if (c is Button)
                        c.Visible = false;
              f.FormBorderStyle=FormBorderStyle.None;
            }
            masForm[index] = f; }
    ... }
```

10. Для відображення при натисненні обраної кнопки вибору відповідного їй питання встановіть для кожної кнопки відповідний їй послідовний унікальний індекс, починаючи від нуля, у властивості *Tag*. Створіть процедуру обробки натиснення довільної кнопки вибору і забезпечте виклик цієї процедури всіма іншими кнопками вибору. Ця процедура має приховувати всі виведені пов'язані форми питання і відображати лише форму, відповідну властивості *Tag* обраної кнопки вибору. Виглядати вона може, наприклад, так:

```
    private void radioButton1_Click(object sender, EventArgs e)
     {// змінюємо колір фону кнопки, вказуючи на те, що питання
переглядалося
      ((RadioButton)sender).BackColor =
System.Drawing.SystemColors.AppWorkspace;
      //приховуємо всі виведені форми питань
      for (int i = 0; i < countQuestion; i++)
       if (masForm[i] != null && !masForm[i].IsDisposed &&
masForm[i].Visible)
         masForm[i].Visible = false;
      //відображаємо лише форму, відповідну натиснутій кнопці
      int index = Convert.ToInt32(((RadioButton)sender).Tag);
      Form f = masForm[index];
      //якщо форма питання закрита – то створюємо її знову
      if (f==null || f.IsDisposed)
       {createForm(index);
        f = masForm[index]; }
      if (f == null) return;
      f.Show(); }
```

11. Використовуючи інтерфейс *IQuationTest*, реалізуйте обробку події натиснення кнопки завершення підсумкового тестування, забезпечивши в ній підрахунок набраних балів. Після цього в процедурі виведіть результати тестування, закрийте саму форму тестування та пов'язані форми питань. Реалізувати це можна, наприклад, так:

```
private void button1_Click(object sender, EventArgs e)
 {int countQuestionTest = 0;
  double sumBalTest = 0;
  for (int i = 0; i < countQuestion; i++)
   if (masForm[i] != null && !masForm[i].IsDisposed &&
       ((RadioButton)(Controls["RadioButton" + (i +
1).ToString()])).BackColor ==
        System.Drawing.SystemColors.AppWorkspace)
    {countQuestionTest++;
     sumBalTest += ((IQuationTest)masForm[i]).balQuestion(); }
  MessageBox.Show("За результатами підсумкового тесту ви відповіли
на " +
                  countQuestionTest + " питань і набрали " +
                  sumBalTest.ToString() + " балів", "Результати",
                  MessageBoxButtons.OK,
MessageBoxIcon.Information);
  Close();
  for (int i = 0; i < countQuestion; i++)
   if (masForm[i] != null && !masForm[i].IsDisposed)
    masForm[i].Close(); }
```

12. Самостійно створіть форми для підпунктів *Про розробника* та *Про програму* пункту головного меню *Довідка* розробленої навчально-контролюючої програми.

Рекомендована література

1. Брила А. Ю., Антосяк П. П., Глебена М. І. та ін. Основи об'єктно-орієнтованого програмування у C#. Методичні вказівки до лабораторних робіт для студентів І-го курсу математичного факультету спеціальності "Прикладна математика". Ужгород, 2014. 73 с.
2. Кравець П. О. Об'єктно-орієнтоване програмування. Львів: Видавництво Львівської політехніки, 2012. 624 с.
3. Ковалюк Т. В. Алгоритмізація та програмування: Підручник. Львів: Магнолія-2006, 2019. 400 с.
4. Ришковець Ю. В., Висоцька В. А. Алгоритмізація та програмування: Навчальний посібник. Львів: Новий світ 2000, 2020. Ч. 1 – 337 с., Ч. 2 – 314 с.
5. Троелсен Ендрю, Джепікс Філіп. Мова програмування C# 7.0 та платформи .NET і .NET Core. 8-е вид. Київ: Діалектика, 2020. 656 с. Рос. мовою.
6. Кормен Т. Х. Лейзерсон Ч., Ріверс Р., Штайн К. Алгоритми: побудова й аналіз. Київ: Діалектика, 2020. 1324 с.

Навчальне видання

Об’єктно-орієнтоване програмування. Лабораторний практикум для студентів денної та заочної форм навчання освітньої програми «Інженерія програмного забезпечення Інтернету речей» першого (бакалаврського) рівня вищої освіти за спеціальністю 121 Інженерія програмного забезпечення галузі знань 12 Інформаційні технології

Укладач
Олександр Володимирович Шпортько

Відповідальний за випуск доц. Ю. Г. Лотюк
Комп'ютерна верстка О. В. Шпортька

Підписано до друку 21.06.2023.
Формат 60x84 1/16. Папір офсетний № 1.
Умовн. друк. арк. 5,1. Наклад 50 примірників.

Віддруковано засобами оперативної поліграфії
ПВНЗ "Міжнародний економіко-гуманітарний університет
імені академіка Степана Дем’янчука".
м. Рівне, вул. С. Дем’янчука, 4.