

**МІНІСТЕРСТВО ОСВІТИ І НАУКИ УКРАЇНИ**

**Національний аерокосмічний університет ім. М. Є. Жуковського  
«Харківський авіаційний інститут»**

**С. М. Андрєєв, В. А. Жилін, А. С. Нечаусов**

**АЛГОРИТМІЧНІ ОСНОВИ ГЕОМАТИКИ І СИСТЕМОЛОГІЇ**

**Навчальний посібник до практичних занять**

**Харків «ХАІ» 2020**

УДК 911: 004+004.65 (075.8)  
А65

Рецензенти: д-р техн. наук, проф. Р. Е. Пащенко,  
канд. техн. наук, доц. Б. М. Іващук

**Андрєєв, С. М.**

А65 Алгоритмічні основи геоматики і системології [Текст] : навч. посіб.  
до практ. занять / С. М. Андрєєв, В. А. Жилін, А. С. Нечаусов. –  
Харків : Нац. аерокосм. ун-т ім. М. Є. Жуковського «Харків. авіац.  
ін-т», 2020. – 88 с.

ISBN 978-966-662-766-0

Викладено практичні рекомендації щодо розроблення програмних кодів і наведено приклади виконання обчислювальних алгоритмів мовою C# у середовищі розроблення Microsoft Visual Studio.

Для студентів, що вивчають дисципліни «Алгоритмічні основи геоматики і системології», «Космічна метеорологія», «Транспортно-навігаційні ГІС» за спеціальностями 193 «Геодезія і землеустрій» (освітня програма «Геоінформаційні системи і технології») і 103 «Науки про Землю» (освітня програма «Космічний моніторинг Землі»).

Іл. 24. Бібліогр.: 10 назв

**УДК 911: 004+004.65 (075.8)**

© Андрєєв С. М., Жилін В. А., Нечаусов А. С., 2020

© Національний аерокосмічний  
університет ім. М. Є. Жуковського  
«Харківський авіаційний інститут», 2020

ISBN 978-966-662-766-0

## ЗМІСТ

Вступ .....	4
Практичне заняття № 1. Реалізація лінійних обчислювальних процесів мовою програмування C# у середовищі розроблення Microsoft Visual Studio .....	5
Практичне заняття № 2. Реалізація обчислювальних процесів із розгалуженням мовою програмування C# у середовищі розроблення Microsoft Visual Studio .....	15
Практичне заняття № 3. Реалізація циклічних обчислювальних процесів мовою програмування C# у середовищі розроблення Microsoft Visual Studio .....	21
Практичне заняття № 4. Реалізація обчислювальних процесів оброблення одновимірних масивів мовою програмування C# у середовищі розроблення Microsoft Visual Studio .....	27
Практичне заняття № 5. Реалізація обчислювальних процесів оброблення двовимірних масивів мовою програмування C# у середовищі розроблення Microsoft Visual Studio .....	33
Практичне заняття № 6. Розроблення найпростіших класів мовою програмування C# у середовищі розроблення Microsoft Visual Studio ..	37
Практичне заняття № 7. Реалізація обчислювальних процесів оброблення рядків мовою програмування C# у середовищі розроблення Microsoft Visual Studio .....	41
Практичне заняття № 8. Розроблення класів з операціями мовою програмування C# у середовищі розроблення Microsoft Visual Studio ..	44
Практичне заняття № 9. Розроблення класів з наслідуванням мовою програмування C# у середовищі розроблення Microsoft Visual Studio .....	51
Практичне заняття № 10. Розроблення структур мовою програмування C# у середовищі розроблення Microsoft Visual Studio ..	60
Практичне заняття № 11. Розроблення мовою C# у середовищі Microsoft Visual Studio програм з використанням стандартних параметризованих колекцій для зберігання екземплярів класів .....	65
Практичне заняття № 12. Створення Windows-застосунків мовою C# у середовищі розроблення Microsoft Visual Studio .....	75
Рекомендації з програмування .....	84
Бібліографічний список .....	87

## ВСТУП

Нагальні потреби відновлення і розвитку усіх сфер економіки України, мета покращання добробуту громадян нашої держави та побудови конструктивних відносин зі світовою спільнотою передбачають необхідність підвищення якості підготовки вітчизняних фахівців з геодезії, землеустрою та систем дистанційного зондування Землі (ДЗЗ) для забезпечення високої якості життя, запобігання виникненню техногенних і природних надзвичайних ситуацій, а також збереження навколишнього середовища для майбутніх поколінь.

Практичний досвід вирішення будь-яких економічних і суспільних задач доводить превентивне значення застосування комп'ютеризованих засобів оброблення інформації про явища та об'єкти суспільного буття для вирішення переважної більшості невідкладних задач на регіональному та державному рівнях.

Найважливішими напрямками удосконалення комплексів ДЗЗ, зважаючи на світові статистичні показники, є роботизація, а звідси — діджиталізація засобів отримання, оброблення, перетворення, візуалізації, збереження та передачі геоданих, тобто створення та удосконалення геоінформаційних систем спостереження, контролю і регуляції умов безпечного та якісного життя громадян держави і людства взагалі.

Шляхами досягнення означеної мети є удосконалення комп'ютерної підготовки фахівців з геоінформаційних технологій та космічного моніторингу Землі, які мають досконало володіти теоретичними знаннями і практичними навичками із застосування цифрових систем оброблення геоданих.

Саме цьому присвячено цей посібник, що сприятиме оволодінню студентами компетенціями створення, налагодження і компіляції комп'ютерних програм, які реалізують обчислювальні алгоритми математичного забезпечення геоінформаційних систем.

## Практичне заняття № 1

# РЕАЛІЗАЦІЯ ЛІНІЙНИХ ОБЧИСЛЮВАЛЬНИХ ПРОЦЕСІВ МОВОЮ ПРОГРАМУВАННЯ C# У СЕРЕДОВИЩІ РОЗРОБЛЕННЯ MICROSOFT VISUAL STUDIO

**Мета заняття:** отримання навичок написання, налагодження і компіляції програм, що реалізують лінійні обчислювальні алгоритми мовою C# у середовищі розроблення Microsoft Visual Studio.

### Апаратне, програмне та методичне забезпечення

1. Засоби обчислювальної техніки (ПК).
2. Інсталяційний пакет програмного забезпечення Microsoft Visual Studio.
3. Пакет індивідуальних завдань.

### Порядок виконання

1. Вивчення загальних теоретичних положень з Microsoft Visual Studio [1, с. 7 - 38].
2. Інсталяція пакета Microsoft Visual Studio на ПК.
3. Вивчення основних положень про платформу .NET [2, с. 8 - 68].
4. Розроблення МОВОЮ C# програм лінійних обчислювальних процесів згідно з варіантами індивідуальних завдань [2, с. 370 - 371].
5. Підготовка контрольного звіту за результатами виконання практичного заняття.

### Зміст звіту

1. Титульний лист.
2. Індивідуальний варіант завдання.
3. Програмні коди реалізації МОВОЮ C# у середовищі розроблення Microsoft Visual Studio обчислювальних процесів згідно з варіантами індивідуальних завдань.
4. Висновки (про те, що виконано на практичному занятті та які навички отримано).

Звіт має бути виконаний у текстовому процесорі Microsoft Word. На перевірку слід надати папку, назва якої (великими буквами) має містити шифр групи, прізвище виконавця і номер практичного заняття (наприклад, 412ст-ІВАНЕНКО-ПЗ1). У цій папці слід розмістити:

- файл звіту (412ст-Іваненко-ПЗ1.docx або 412ст-Іваненко-ПЗ1.doc);
- папки та файли рішення MS Visual Studio (наприклад, папку "ConsoleApplication3" і файл "ConsoleApplication3.sln", де цифри у назві папки та назві файла мають збігатися з номером варіанта

індивідуального завдання). Якщо створюються декілька рішень MS Visual Studio, то нумерувати відповідні папки та файли слід так: "ConsoleApplication3-1", "ConsoleApplication3-1.sln", "ConsoleApplication3-2", "ConsoleApplication3-2.sln" і т. д., де перша цифра (тут 3) у назві відповідає номеру варіанта, а друга цифра — номеру рішення MS Visual Studio у межах варіанта індивідуального завдання.

При роботі з Microsoft Visual Studio 2010 папку та файл рішення (наприклад, папку "ConsoleApplication1" і файл "ConsoleApplication1.sln") треба скопіювати з папки "ConsoleApplication1", яка за замовчуванням знаходиться так: C:\ Рабочий стол \ Компьютер \ Документы \ VisualStudio \ Projects \ ConsoleApplication1.

### **Технічно це виконується так.**

1. На Робочому столі двічі клікнути лівою клавшею миші (ЛКМ) по іконці "Комп'ютер" (рис. 1).



Рис. 1. Відкриття вікна "Комп'ютер"

2. У вікні "Комп'ютер" обрати меню "Документи" (рис. 2).

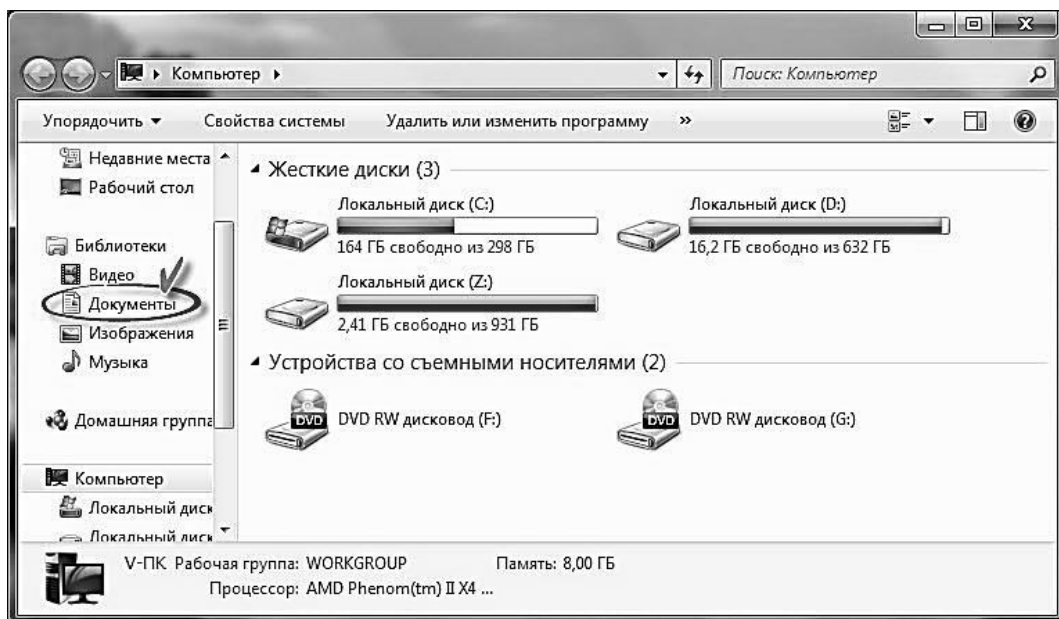


Рис. 2. Вибір меню "Документи"

3. У вікні "Документи" відкрити папку "Visual Studio 2010" (рис. 3).

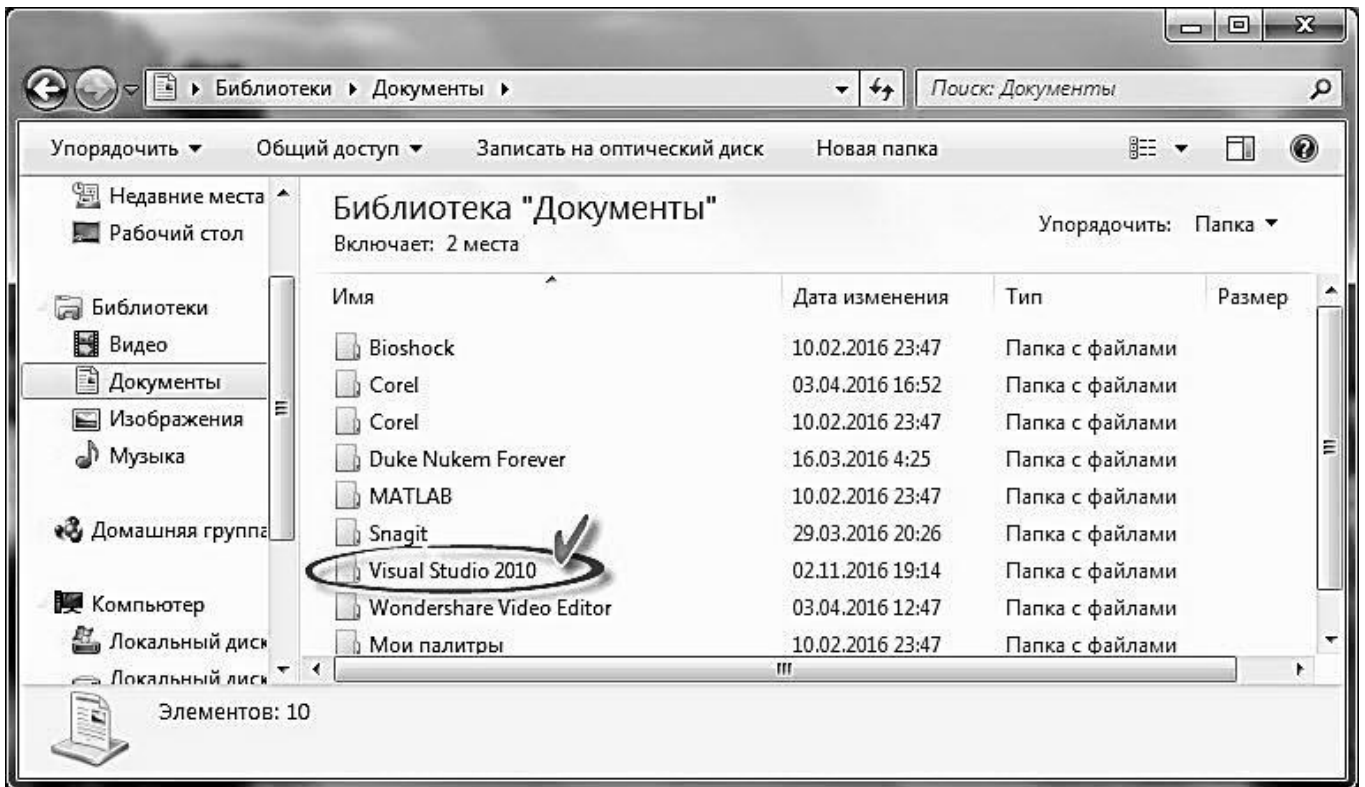


Рис. 3. Відкривання папки "Visual Studio 2010"

4. У папці "Visual Studio 2010" відкрити папку "Projects" (рис. 4).

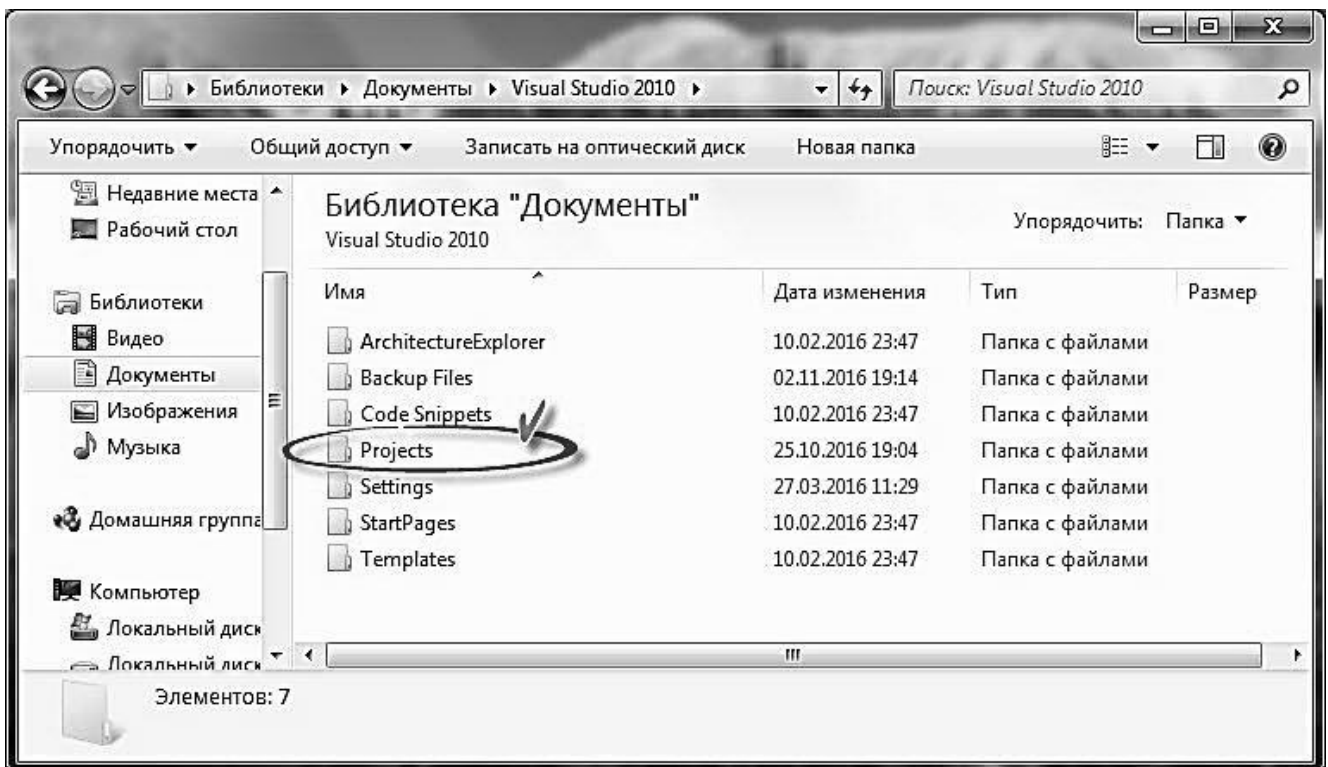


Рис. 4. Відкривання папки "Projects"

5. У папці "Projects" знайти та скопіювати папку "ConsoleApplication1" (рис. 5) у папку 412ст-ІВАНЕНКО-ПЗ1, яку слід здати старості групи. Зрозуміло, що у цій же папці 412ст-ІВАНЕНКО-ПЗ1 треба розмістити й файл звіту (412ст-Іваненко-ПЗ1.docx або 412ст-Іваненко-ПЗ1.doc).

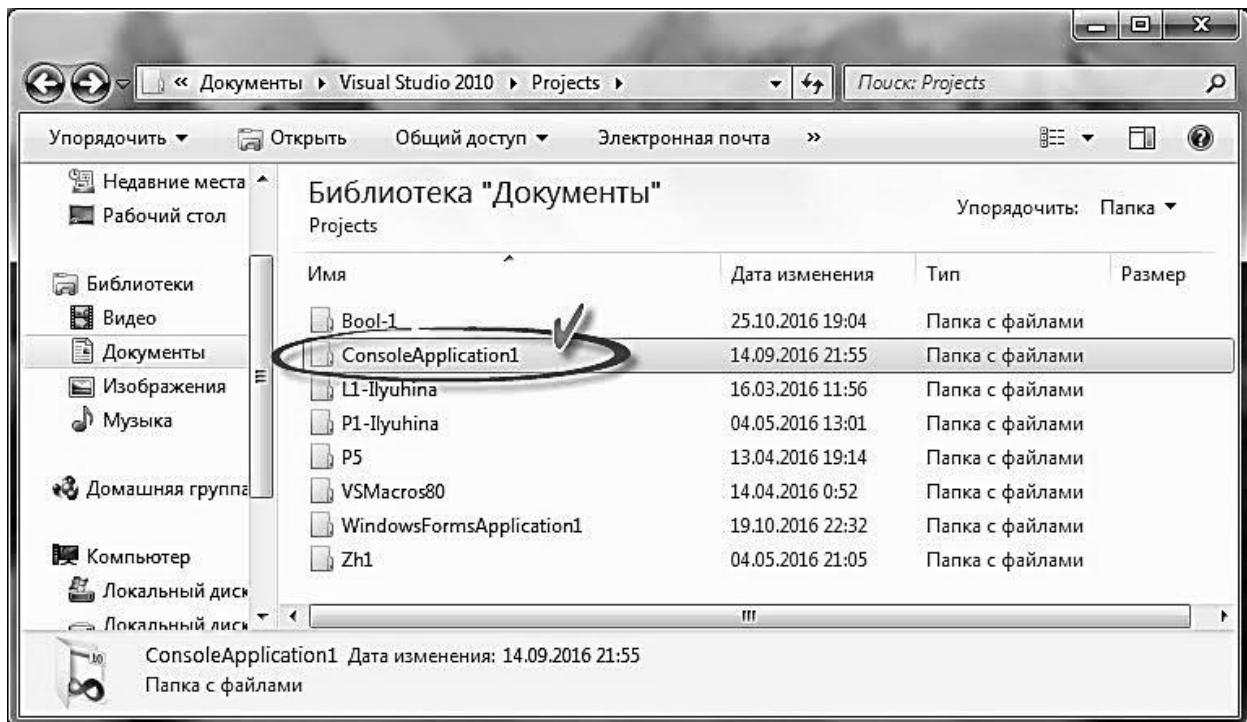


Рис. 5. Копіювання папки "ConsoleApplication1"

Таким чином, у папці зі звітом про виконану роботу 412ст-ІВАНЕНКО-ПЗ1 розмістяться усі файли, які необхідні для перевірки викладачем (рис. 6).

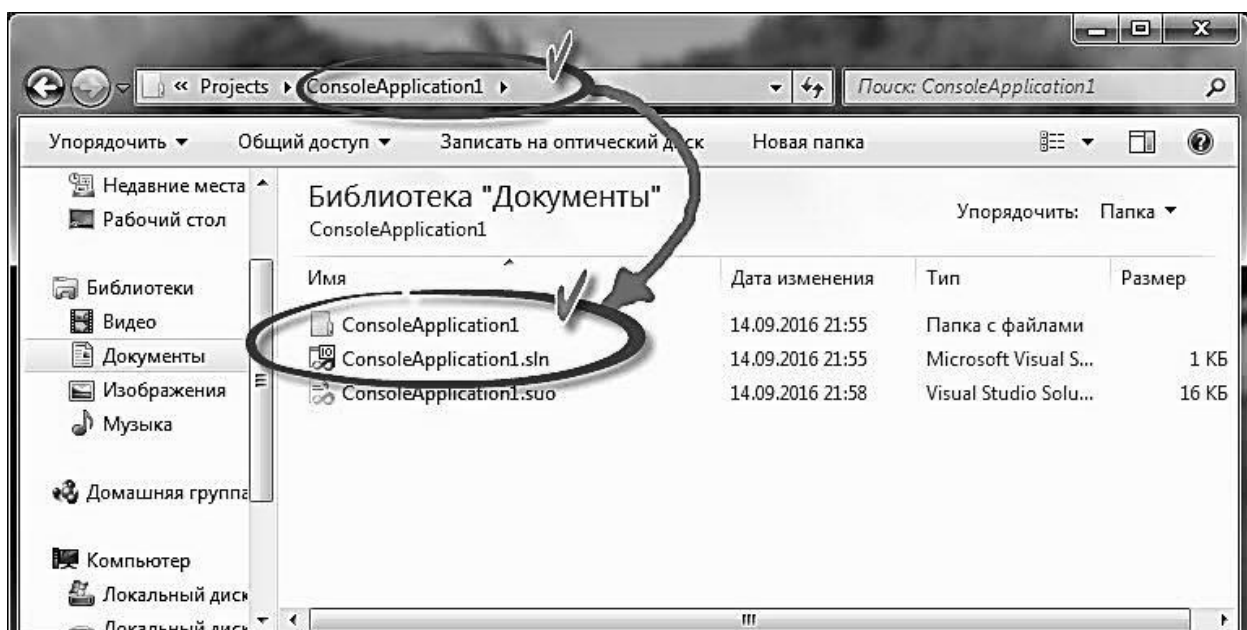


Рис. 6. Вміст папки "ConsoleApplication1"



Для версій Microsoft Visual Studio, що є більш сучаснішими, ніж Microsoft Visual Studio 2010 (наприклад, для Microsoft Visual Studio 2019), файли, що містять ваше рішення, наприклад "ConsoleApp1", слід шукати таким шляхом: **C:\Users\<Ім'я комп'ютера>\source\repos\**

**Технічно це виконується так.**

1. Натисніть кнопку "Пуск" (тут V — ім'я комп'ютера) (рис. 7).

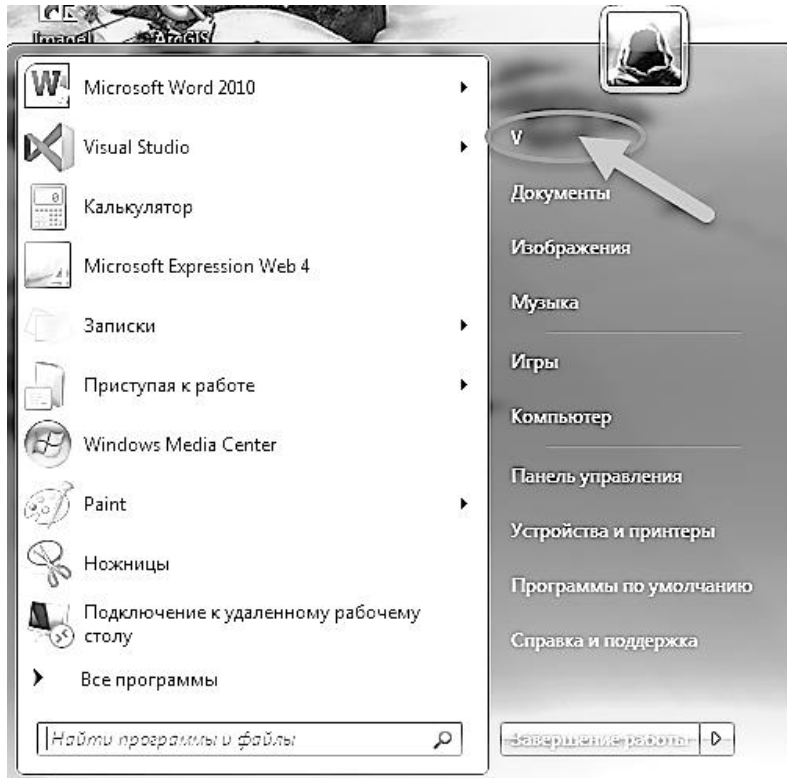


Рис. 7. Відкриття персональної папки користувача

2. У папці користувача відкрийте папку "source" (рис. 8).

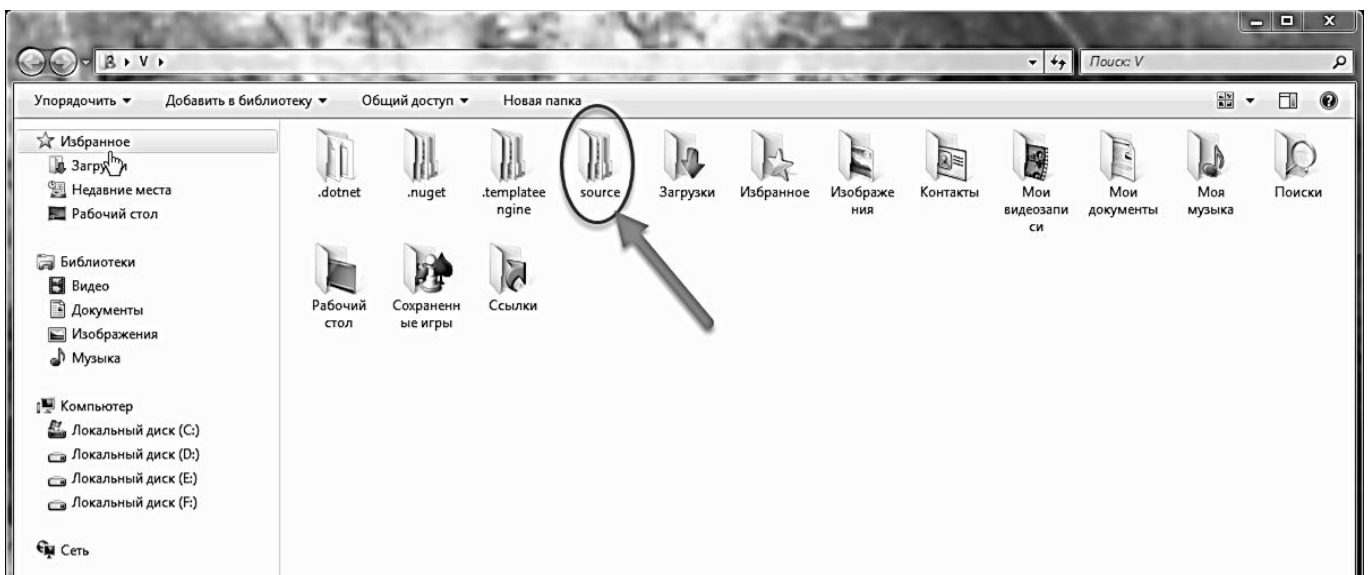


Рис. 8. Відкриття папки "source"

3. У папці "source" відкриті папку "repos" (рис. 9).

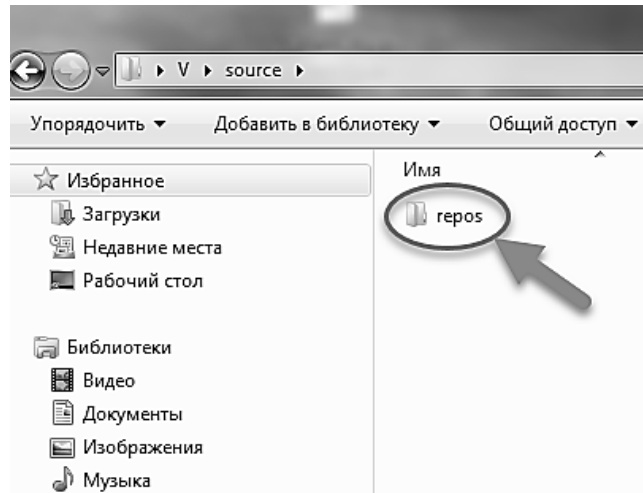


Рис. 9. Відкриття папки "repos"

4. У папці "repos" відкриті папку "ConsoleApp1" (рис. 10).

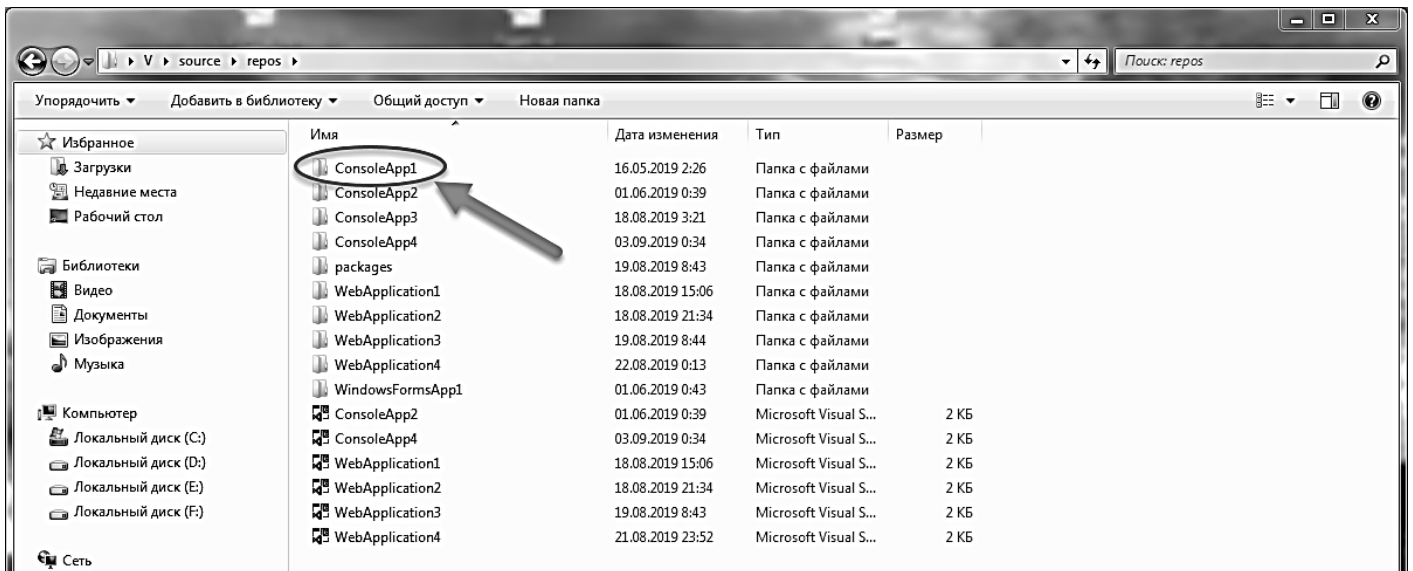


Рис. 10. Відкриття папки "ConsoleApp1"

Саме цю папку (вона може мати оригінальну назву розробника) з проектом (рис. 11) слід здавати викладачеві на перевірку.

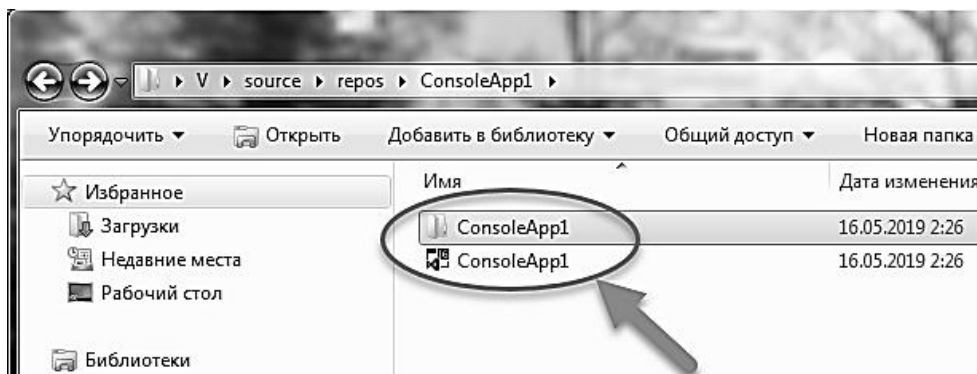


Рис. 11. Вміст папки "ConsoleApp1" з папки "repos"

З рис. 11 видно, що папка "ConsoleApp1" містить папку з аналогічною назвою "ConsoleApp1" (її вміст показано на рис. 12), а також файл рішення ConsoleApp1.sln.

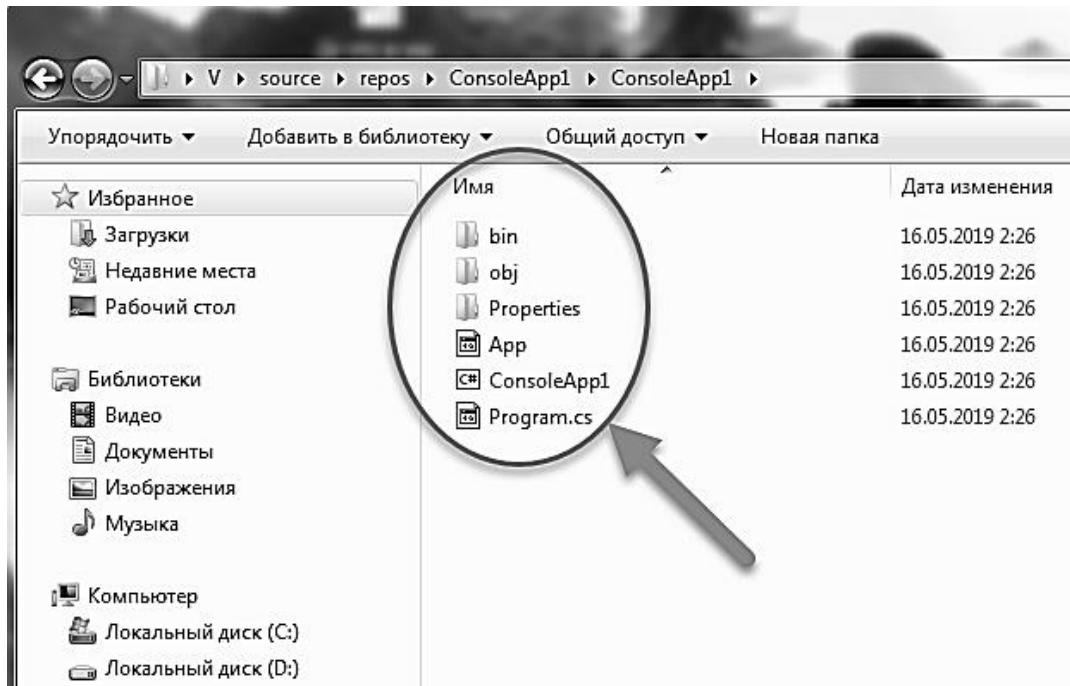


Рис. 12. Вміст папки "ConsoleApp1" з папки "repos\ConsoleApp1"

Звіт має бути виконаний у текстовому процесорі Microsoft Word (шрифтом **Times New Roman**, 14 пт, Вирівнювання: **по ширине**, Міжрядковий інтервал: **1,15**).

Папку зі звітом з практичного заняття слід здати в електронному вигляді старості академічної групи.

Староста групи перевіряє правильність найменування папок звітів студентів, поміщує ці папки в папку під назвою 412ст-ПЗ1 і здає цю папку (в електронному вигляді) на перевірку викладачеві на початку наступного практичного заняття.

### Приклад виконання

**Завдання.** Написати мовою C# у середовищі розроблення Microsoft Visual Studio програму для розрахунку за двома формулами:

$$Z = \frac{\sin \alpha + \cos(2\beta - \alpha)}{\cos \alpha - \sin(2\beta - \alpha)} ;$$

$$Z = \frac{1 + \sin 2\beta}{\cos 2\beta} .$$

## Програмний код

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

namespace _412_IVANENKO_PZ1
{
    class Programm
    {
        static void Main(string[] args)
        {
            Console.WriteLine("Оберіть номер формули з 13 варіанта (1 або 2)"); //1
            string zadacha = (Console.ReadLine()); //2
            if (zadacha == "1"){ //3
                Console.WriteLine("Уведіть a"); //4
                double a = (Convert.ToDouble(Console.ReadLine())); //5
                Console.WriteLine("Уведіть b"); //6
                double b = (Convert.ToDouble(Console.ReadLine())); //7
                if ((Math.Cos(a) - Math.Sin(2 * b - a) == (0))) //8
                {
                    Console.WriteLine("Знаменник не може дорівнювати нулю."); //9
                    Console.ReadKey(); //10
                }
                else //11
                {
                    Console.WriteLine("Відповідь:" + (((Math.Sin(a) + Math.Cos(2 * b - a)) /
(Math.Cos(a) - Math.Sin(2 * b - a)))); //12
                    Console.ReadKey(); //13
                }
            }

            else //14
            {
                if (zadacha == "2") //15
                {
```

```

    Console.WriteLine("Уведіть b"); //16
    double b = (Convert.ToDouble(Console.ReadLine())); //17
    if ((Math.Cos(2 * b) == (0)) //18
    {
        Console.WriteLine("Знаменник не може дорівнювати нулю."); //19
        Console.ReadKey(); //20
    }
    else { //21
        Console.WriteLine("Відповідь:" + ((1 + Math.Sin(2 * b)) / (Math.Cos(2 *
b)))); //22
        Console.ReadKey(); //23
    }
    else //24
    {
        Console.WriteLine("Неправильний номер формули."); //25
        Console.ReadLine(); //26
    }
}
}
}
}
}

```

### Коментарі до програмного коду

- 1 - Програма виводить на екран повідомлення «Оберіть номер формули з 13 варіанта (1 або 2)».
- 2 - У змінну `zadacha` типу `string` зчитується введене в консоль значення.
- 3 - Ініціюється перевірка значення змінної `zadacha` та у випадку, коли ця змінна дорівнює 1, виконуються команди 4 - 9 (розв'язується формула № 1).
- 4 - Виводиться на екран повідомлення «Уведіть а».
- 5 - Зчитується введене значення, переводиться у тип `double` та присвоюється змінній `a`.
- 6 - Виводиться на екран повідомлення «Уведіть b».
- 7 - Зчитується введене значення, переводиться у тип `double` та присвоюється змінній `b`.
- 8 - Перевіряється, чи дорівнює знаменник формули № 1 нулю, якщо так — виконуються команди 9 - 10.
- 9 - Виводиться на екран повідомлення «Знаменник не може дорівнювати нулю».

- 10 - Очікується натискання на будь-яку клавішу.
- 11 - Якщо знаменник формули № 1 не дорівнює нулю, то виконуються команди 12 - 13.
- 12 - Розраховується і виводиться на екран відповідь на формулу № 1.
- 13 - Очікується натискання на будь-яку клавішу.
- 14 - Якщо змінна *zadacha* не дорівнює 1, то виконується команда 15.
- 15 - Якщо змінна *zadacha* дорівнює 2, то виконуються команди 16 -18 (виконується завдання 2).
- 16 - Виводиться на екран повідомлення «Уведіть *b*».
- 17 - Зчитується введене значення, переводиться у тип *double* та присвоюється змінній *b*.
- 18 - Якщо знаменник формули № 2 дорівнює нулю, виконуються команди 19 - 20.
- 19 - Виводиться на екран повідомлення «Знаменник не може дорівнювати нулю».
- 20 - Очікується натискання на будь-яку клавішу.
- 21 - Якщо знаменник формули № 2 не дорівнює нулю, виконуються команди 22 - 23.
- 22 - Розраховується та виводиться на екран відповідь на формулу № 2.
- 23 - Очікується натискання на будь-яку клавішу.
- 24 - Якщо введений номер формули (змінна *zadacha*) не дорівнює 1 або 2, виконуються команди 25 - 26.
- 25 - Виводиться на екран повідомлення «Неправильний номер формули».
- 26 - Очікується натискання на будь-яку клавішу.

## **Висновки**

При виконанні практичного завдання було проведено роботу зі встановлення Microsoft Visual Studio на ПК, відбулося ознайомлення з інтерфейсом даного середовища розроблення, а також з процесом написання лінійних програм, дослідження використання класу *Math*, конструкції *if-else* тощо. Ці навички були використані на практиці під час написання та компіляції програми, яка знаходить результат однієї з двох тригонометричних формул на вибір користувача

## **Контрольні запитання**

1. Структурний склад середовища Visual Studio.NET.
2. Склад мови програмування C#.
3. Типи даних мови програмування C#.
4. Змінні та іменовані константи мови програмування C#.
5. Операції та вирази мови програмування C#.
6. Особливості реалізації лінійних програм мовою програмування C#.

## Практичне заняття № 2

### РЕАЛІЗАЦІЯ ОБЧИСЛЮВАЛЬНИХ ПРОЦЕСІВ ІЗ РОЗГАЛУЖЕННЯМ МОВОЮ ПРОГРАМУВАННЯ C# У СЕРЕДОВИЩІ РОЗРОБЛЕННЯ MICROSOFT VISUAL STUDIO

**Мета заняття:** отримання навичок написання, налагодження і компіляції програм, що реалізують обчислювальні алгоритми із розгалуженням мовою C# у середовищі розроблення Microsoft Visual Studio.

#### Апаратне, програмне та методичне забезпечення

1. Засоби обчислювальної техніки (ПК).
2. Інсталяційний пакет програмного забезпечення Microsoft Visual Studio.
3. Пакет індивідуальних завдань.

#### Порядок виконання

1. Повторення загальних теоретичних положень з Microsoft Visual Studio [1, с. 7 - 38].
2. Повторення основних положень про платформу .NET [2, с. 8 - 68].
3. Вивчення основних теоретичних положень із синтаксису операторів розгалуження [2, с. 69 - 75].
4. Розроблення МОВОЮ C# програм обчислювальних процесів із розгалуженням згідно з варіантами індивідуальних завдань [2, с. 371 - 379].
5. Підготовка контрольного звіту за результатами виконання практичного заняття.

#### Зміст звіту

1. Титульний лист.
2. Індивідуальний варіант завдання.
3. Програмні коди реалізації МОВОЮ C# у середовищі розроблення Microsoft Visual Studio обчислювальних процесів згідно з індивідуальним варіантом завдання.
4. Висновки (про те, що виконано на практичному занятті та які навички отримано).

Звіт має бути виконаний у текстовому процесорі **Microsoft Word** (шрифтом **Times New Roman, 14 пт**, Вирівнювання: **по ширине**, Міжрядковий інтервал: **1,15**). На перевірку слід надати папку, назва якої (великими буквами) містить шифр групи, прізвище виконавця і номер практичного заняття (наприклад, 412ст-ІВАНЕНКО-П32). У цій папці слід розмістити:

- файл звіту (412ст-Іваненко-П31.docx або 412ст-Іваненко-П32.doc);

- папки та файли рішення MS Visual Studio (наприклад, папку "ConsoleApplication3" і файл "ConsoleApplication3.sln", де цифри у назві папки та назві файла мають збігатися з номером варіанта індивідуального завдання). Якщо створюються декілька рішень MS Visual Studio, то нумерувати відповідні папки та файли слід так: "ConsoleApplication3-1" і "ConsoleApplication3-1.sln", "ConsoleApplication3-2" і файл "ConsoleApplication3-2.sln" і т. д., де перша цифра (тут 3) у назві відповідає номеру варіанта, а друга цифра — номеру рішення MS Visual Studio у межах варіанта індивідуального завдання.

Папку зі звітом з практичного заняття слід здати в електронному вигляді старості академічної групи. Староста групи перевіряє правильність найменування папок звітів студентів, поміщує ці папки в папку під назвою 412ст-П32 і здає цю папку (в електронному вигляді) на перевірку викладачеві на початку наступного практичного заняття.

### Приклад виконання

**Завдання 1.** Написати МОВОЮ С# у середовищі розроблення Microsoft Visual Studio програму, що за введеним значенням аргумента обчислює значення функції, яку задано у вигляді графіка (рис. 13). Параметр  $R$  задається як константа у програмному коді.

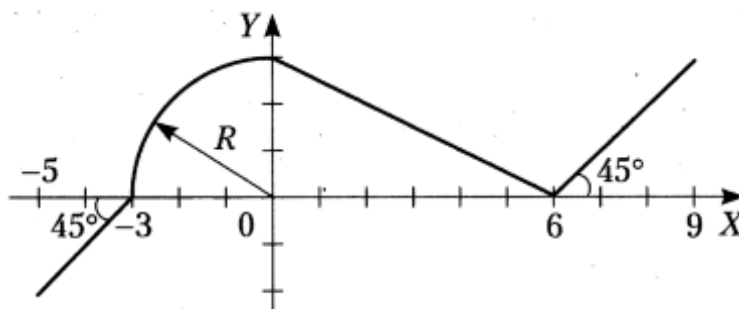


Рис. 13. Задана функція у вигляді графіка

### Програмний код

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

namespace _412_IVANENKO_PZ2_1
{
    class Program
```



```

{
static void Main(string[] args)
{ Console.WriteLine("Уведіть x");//1
  double x = Convert.ToDouble(Console.ReadLine());//2
  const int R = 3;//3
  Double y = 9999;//4
  if (x <= -3 && x > -5)//5
  {
    y = x + 3;//6
  }
  if (x > -3 && x < 0)//7
  {
    y = Math.Sqrt(R * R - x * x);//8
  }
  if (x >= 0 && x < 6)//9
  {
    y = (3 - x / 2);//10
  }
  if (x >= 6 && x < 9)//11
  {
    y = (x - 6);//12
  }
  if (x < -5 | x > 9)//13
  {
    Console.WriteLine("Неприпустиме значення x");//14
    Console.ReadKey();//15
  }
  else
  {
    Console.WriteLine("y =" + y);//16
    Console.ReadKey();//17
  }
}
}
}

```

## Коментарі до програмного коду

- 1 - На екран виводиться "Уведіть x";
- 2 - Зчитується введене в консоль значення та присвоюється змінній x.
- 3 - Створюється константа R, яка є значенням радіуса.
- 4 - Створюється змінна типу int.
- 5 - Якщо  $x \leq -3$ , виконується команда 6.
- 6 - Змінній y присвоюється значення  $x + 3$ .
- 7 - Якщо  $x > -3$  та  $x < 0$ , виконується команда 8.
- 8 - Змінній y присвоюється значення  $\sqrt{R^2 - x^2}$ .
- 9 - Якщо  $x \geq 0$  та  $x < 6$ , виконується команда 9.
- 10 - Змінній y присвоюється значення  $3 - x / 2$ .
- 11 - Якщо  $x \geq 6$ , виконується команда 12.
- 12 - Змінній y присвоюється значення  $x - 6$ .
- 13 - Якщо  $x < -5$  або  $x > 9$ , виконуються команди 14-15. В іншому випадку виконуються команди 16-17.
- 14 - На екран виводиться «Неприпустиме значення x».
- 15 - Очікується натиснення будь-якої клавіші.
- 16 - На екран виводиться результат обчислення.
- 17 - Очікується натиснення будь-якої клавіші.

**Завдання 2.** Написати мовою C# в середовищі розроблення Microsoft Visual Studio програму, яка визначає, чи попадає точка із заданими координатами в область, зафарбовану на рисунку сірим кольором (рис. 14). Результат роботи програми вивести у вигляді текстового повідомлення.

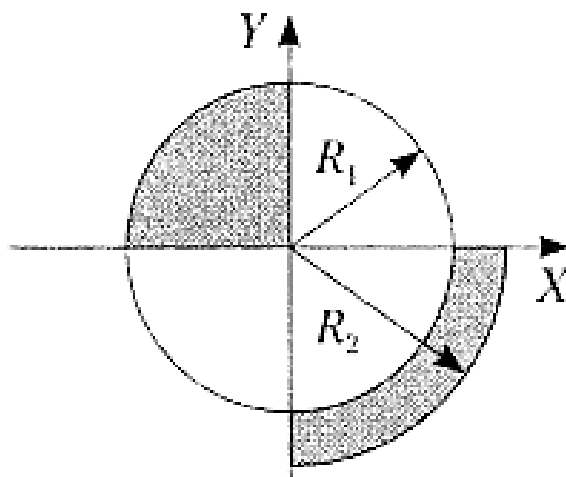


Рис. 14. Задані координатні області

## Програмний код

```
using System;//1
using System.Collections.Generic;//2
using System.Linq;//3
using System.Text;//4
using System.Threading.Tasks;//5

namespace _412_IVANENKO_PZ2_2
{
    class Program
    {
        static void Main(string[] args)
        {
            Console.WriteLine("Уведіть x");//6
            double x = Convert.ToDouble(Console.ReadLine());
            Console.WriteLine("Уведіть y");
            double y = Convert.ToDouble(Console.ReadLine());
            Console.WriteLine("Уведіть R1");
            double R1 = Convert.ToDouble(Console.ReadLine());
            Console.WriteLine("Уведіть R2");
            double R2 = Convert.ToDouble(Console.ReadLine());//7
            if (((x * x + y * y) < (R1 * R1)) && x <= 0 && y >= 0) | (((x * x + y * y <=
R2 * R2) && (x * x + y * y >= R1 * R1)) && (x >= 0 && y <= 0))//8
            {
                Console.WriteLine("Задана точка знаходиться у заштрихованій
області");//9
                Console.ReadKey();//10
            }
            else
            {
                Console.WriteLine("Задана точка не знаходиться у заштрихованій
області");//11
                Console.ReadKey();//12
            }
        }
    }
}
```

## Коментарі до програмного коду

- 1 - Надається можливість не прописувати в подальших командах System.
- 2 - Надається можливість не прописувати в подальших командах System.Collections.Generic.
- 3 - Надається можливість не прописувати в подальших командах System.Linq (LINQ — це функція .NET 3.5 для вбудованої підтримки з мовою від C # 3.0 і Visual Basic 2008).
- 4 - Надається можливість не прописувати в подальших командах System.Text (застосовується система кодування символів).
- 5 - Завдання класу (і вся остання бібліотека TPL) визначається у просторі імен System.Threading.Tasks.
- 6 - 7 - Набір команд, що забезпечує введення вхідних даних x, y, R1, R2.
- 8 - Команда, яка за допомогою логічних операторів і рівняння кола перевіряє, чи потрапляє точка з заданими координатами в заштриховану область.
- 9 - Якщо точка належить заштрихованій області, виводиться на екран «Задана точка знаходиться у заштрихованій області».
- 10 - Очікується натиснення будь-якої клавіші.
- 11 - Якщо точка не належить заштрихованій області, виводиться на екран «Задана точка не знаходиться у заштрихованій області».
- 12 - Очікується натиснення будь-якої клавіші.

## Висновки

У результаті виконання практичного заняття отримано навички написання, налагодження та компіляції програм, що реалізують обчислювальні алгоритми із розгалуженням МОВОЮ С# у середовищі розроблення Microsoft Visual Studio.

## Контрольні запитання

1. Наведіть приклади виразів МОВОЮ програмування С#.
2. Призначення пустих операторів мови програмування С#.
3. Призначення операторів розгалуження мови програмування С#.
4. Умовний оператор розгалуження if: формат і структурні схеми.
5. Оператор вибору switch: призначення, формат і структурні схеми.

## Практичне заняття № 3

### РЕАЛІЗАЦІЯ ЦИКЛІЧНИХ ОБЧИСЛЮВАЛЬНИХ ПРОЦЕСІВ МОВОЮ ПРОГРАМУВАННЯ C# У СЕРЕДОВИЩІ РОЗРОБЛЕННЯ MICROSOFT VISUAL STUDIO

**Мета заняття:** отримання навичок написання, налагодження і компіляції програм, що реалізують циклічні обчислювальні алгоритми мовою C# у середовищі розроблення Microsoft Visual Studio.

#### Апаратне, програмне та методичне забезпечення

1. Засоби обчислювальної техніки (ПК).
2. Інсталяційний пакет програмного забезпечення Microsoft Visual Studio.
3. Пакет індивідуальних завдань.

#### Порядок виконання

1. Повторення загальних теоретичних положень з Microsoft Visual Studio [1, с. 7 - 38].
2. Повторення основних положень про платформу .NET [2, с. 8 - 68].
3. Повторення основних теоретичних положень із синтаксису операторів розгалуження [2, с. 69 - 75].
4. Вивчення основних теоретичних положень із синтаксису операторів циклу [2, с. 75 - 88].
5. Розроблення МОВОЮ C# програм циклічних обчислювальних процесів згідно з варіантами індивідуальних завдань [2, с. 379] (завдання 1 - Обчислення і побудова таблиці значень функцій; завдання 2 - Визначення приналежності точок із заданої множини заданій площині).
6. Підготовка контрольного звіту за результатами виконання практичного заняття.

#### Зміст звіту

1. Титульний лист.
2. Індивідуальний варіант завдання.
3. Програмні коди реалізації МОВОЮ C# у середовищі розроблення Microsoft Visual Studio обчислювальних процесів згідно з індивідуальним варіантом завдання.
4. Висновки (про те, що виконано на практичному занятті та які навички отримано).

Звіт має бути виконаний у текстовому процесорі **Microsoft Word** (шрифтом **Times New Roman, 14 пт**, Вирівнювання: **по ширине**, Міжрядковий інтервал: **1,15**). На перевірку слід надати папку, назва якої (великими буквами) містить шифр групи, прізвище виконавця і номер

практичного заняття (наприклад, 412ст-ІВАНЕНКО-ПЗЗ). У цій папці слід розмістити:

- файл звіту (412ст-Іваненко-ПЗЗ.docx або 412ст-Іваненко-ПЗЗ.doc);
- папки та файли рішення MS Visual Studio (наприклад, папку "ConsoleApplication3" і файл "ConsoleApplication3.sln", де цифри у назві папки та назві файла мають збігатися з номером варіанта індивідуального завдання). Якщо створюється декілька рішень MS Visual Studio, то нумерувати відповідні папки та файли слід так: "ConsoleApplication3-1" і "ConsoleApplication3-1.sln", "ConsoleApplication3-2" і файл "ConsoleApplication3-2.sln" і т. д., де перша цифра (тут 3) у назві відповідає номеру варіанта, а друга цифра — номеру рішення MS Visual Studio у межах варіанта індивідуального завдання.

Папку зі звітом з практичного заняття слід здати в електронному вигляді старості академічної групи. Староста групи перевіряє правильність найменування папок звітів студентів, поміщує ці папки в папку під назвою 412ст-ПЗЗ і здає цю папку (в електронному вигляді) на перевірку викладачеві на початку наступного практичного заняття.

### Приклад виконання

**Завдання 1.** Обчислити та вивести на екран у вигляді таблиці значення функції, яку задано у вигляді графіка (рис. 15). З цією метою написати МОВОЮ С# у середовищі розроблення Microsoft Visual Studio програму, що реалізує циклічний обчислювальний процес на інтервалі від  $x_{\text{поч}}$  до  $x_{\text{кінц}}$  з кроком  $dx$ . Інтервал і крок задати таким чином, щоб перевірити усі гілки графіка. Параметр  $R$  задається як константа у програмному коді. Таблицю "оздобити шапкою".

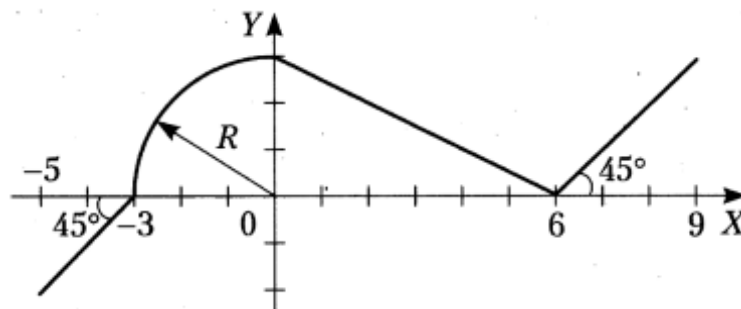


Рис. 15. Задана функція у вигляді графіка

### Програмний код

```
using System;  
using System.Collections.Generic;  
using System.Linq;
```

```

using System.Text;
using System.Threading.Tasks;

namespace _412_IVANENKO_PZ3_1
{
    class Program
    {
        static void Main(string[] args)
        {
            Console.WriteLine(" _____");//1
            Console.WriteLine("|x   y |");
            Double x = -5;//2
            const int R = 3;//3
            Double y = 9999;
            for (int i = 0; i < 15;i = i+1)//4
            {
                if (x <= -3 && x >= -5)//5
                {
                    y = Math.Round((x + 3),2);//6
                }
                if (x > -3 && x < 0)//7
                {
                    y = Math.Round(Math.Sqrt(R * R - x * x),2);//8
                }
                if (x >= 0 && x < 6)//9
                {
                    y = Math.Round((3 - (x / 2)),2);//10
                }
                if (x >= 6 && x < 9)//11
                {
                    y = Math.Round((x - 6),2);//12
                }
                if (x < -5 | x > 9)//13
                {
                    Console.WriteLine("Неприпустиме значення x");//14
                }
                else
                {
                    Console.WriteLine("|" + x + "   " + y + "|");//16
                }
                x = x + 1;//17
            }
            Console.ReadKey();//18
        }
    }
}

```

```
}  
}  
}
```

### Коментарі до програмного коду

- 1 - Виводиться на екран «шапка» таблиці.
- 2 - Створюються змінні для програми.
- 3 - Створюється константа, що містить значення радіуса.
- 4 - Розпочинається цикл типу for, який буде виконано 15 разів.
- 5 - Якщо  $x \leq -3$ , виконується команда 6.
- 6 - Змінній  $y$  присвоюється значення  $x + 3$ .
- 7 - Якщо  $x > -3$  і  $x < 0$ , виконується команда 8.
- 8 - Змінній  $y$  присвоюється значення  $\sqrt{R^2 - x^2}$ .
- 9 - Якщо  $x \geq 0$  і  $x < 6$ , виконується команда 9.
- 10 - Змінній  $y$  присвоюється значення  $3 - x / 2$ .
- 11 - Якщо  $x \geq 6$ , виконується команда 12.
- 12 - Змінній  $y$  присвоюється значення  $x - 6$ .
- 13 - Якщо  $x < -5$  або  $x > 9$ , виконуються команди 14 - 15. В іншому випадку виконуються команди 16 - 17.
- 14 - На екран виводиться «Неприпустиме значення  $x$ ».
- 15 - Очікується натиснення будь-якої клавіші.
- 16 - Збільшується на одиницю змінна  $x$  перед початком наступного циклу.
- 17 - На екран виводиться результат обчислення.
- 18 - Очікується натиснення будь-якої клавіші.

**Завдання 2.** Написати мовою C# у середовищі розроблення Microsoft Visual Studio програму, що визначає, чи попадає точка із заданими координатами в область, зафарбовану на рисунку сірим кольором (рис. 16), для десяти пар координат  $(x, y)$ . Результат роботи програми виводити у вигляді текстового повідомлення на кожній ітерації циклу.

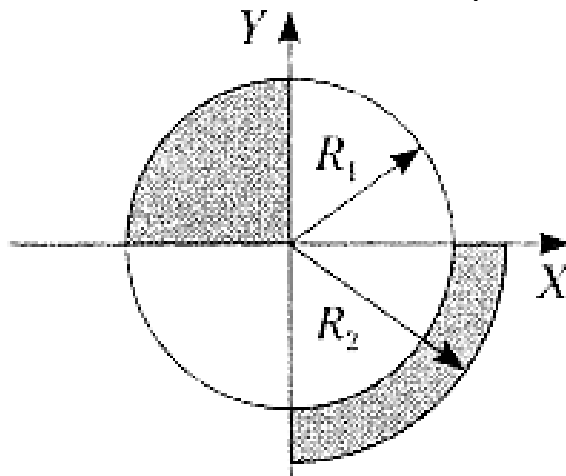


Рис. 16. Задані координатні області



## Програмний код

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

namespace _412_IVANENKO_PZ3_2
{
    class Program
    {
        static void Main(string[] args)
        { for (int i = 0; i < 10; i++)//1
            {
                Console.WriteLine("Уведіть x");//2
                double x = Convert.ToDouble(Console.ReadLine());
                Console.WriteLine("Уведіть y");
                double y = Convert.ToDouble(Console.ReadLine());
                Console.WriteLine("Уведіть R1");
                double R1 = Convert.ToDouble(Console.ReadLine());
                Console.WriteLine("Уведіть R2");
                double R2 = Convert.ToDouble(Console.ReadLine());//3
                if (((x * x + y * y) < (R1 * R1)) && x <= 0 && y >= 0) | (((x * x + y * y <=
R2 * R2) && (x * x + y * y >= R1 * R1)) && (x >= 0 && y <= 0))//4
                {
                    Console.WriteLine("Задана точка знаходиться у заштрихованій області
");//5
                }
                else
                {
                    Console.WriteLine("Задана точка не знаходиться у заштрихованій
області ");//6
                }
                if (i == 10)//7
                {
                    Console.ReadKey();//8
                }
            }
        }
    }
}
```

## Коментарі до програмного коду

- 1 - Створюється цикл for, що має 10 ітерацій.
- 2 - 3 - Набір команд, що реалізує введення з клавіатури вхідних даних x, y, R1, R2.
- 4 - Команда, яка з використанням логічних операторів і рівняння кола перевіряє, чи потрапляє точка із заданими координатами в заштриховану область.
- 5 - Якщо точка належить заштрихованій області, виводиться на екран «Задана точка знаходиться у заштрихованій області».
- 6 - Якщо точка не належить заштрихованій області, виводиться на екран «Задана точка не знаходиться у заштрихованій області».
- 7 - 8 - Якщо лічильник і дорівнює 10, то очікується натиснення будь-якої клавіші для завершення програми.

## Висновки

У результаті виконання практичного заняття отримано навички написання, налагодження і компіляції програм, що реалізують циклічні обчислювальні алгоритми МОВОЮ програмування С# у середовищі розроблення Microsoft Visual Studio.

## Контрольні запитання

1. Призначення та види операторів циклу мови програмування С#.
2. Дайте визначення основних понять циклу: тіло циклу, ітерація, початкові установки, передумова, постумова, параметр циклу, лічильник циклу, ітеративний цикл.
3. Цикл з передумовою while: призначення та формат.
4. Цикл з постумовою do: призначення та формат.
5. Цикл з параметром for: призначення та формат.
6. Цикл перебору foreach.

## Практичне заняття № 4

### РЕАЛІЗАЦІЯ ОБЧИСЛЮВАЛЬНИХ ПРОЦЕСІВ ОБРОБЛЕННЯ ОДНОВИМІРНИХ МАСИВІВ МОВОЮ ПРОГРАМУВАННЯ C# У СЕРЕДОВИЩІ РОЗРОБЛЕННЯ MICROSOFT VISUAL STUDIO

**Мета заняття:** отримання навичок написання, налагодження і компіляції програм, що реалізують обчислювальні алгоритми оброблення одновимірних масивів мовою C# у середовищі розроблення Microsoft Visual Studio.

#### Апаратне, програмне та методичне забезпечення

1. Засоби обчислювальної техніки (ПК).
2. Інсталяційний пакет програмного забезпечення Microsoft Visual Studio.
3. Пакет індивідуальних завдань.

#### Порядок виконання

1. Вивчення основних теоретичних положень [2, с. 89 - 151].
2. Розроблення МОВОЮ C# програм обчислювальних процесів оброблення одновимірних масивів згідно з варіантами індивідуальних завдань [2, с. 385 - 389].
3. Підготовка контрольного звіту за результатами виконання практичного заняття.

Примітка. У кожному індивідуальному завданні мається на увазі, що програма повинна забезпечувати оброблення будь-якої кількості елементів масиву, що вводяться з клавіатури, не за збільшенням або зменшенням, а в довільному порядку.

#### Зміст звіту

1. Титульний лист.
2. Індивідуальний варіант завдання.
3. Програмні коди реалізації МОВОЮ C# у середовищі розроблення Microsoft Visual Studio обчислювальних процесів згідно з індивідуальним варіантом завдання.
4. Висновки (про те, що виконано на практичному занятті та які навички отримано).

Звіт має бути виконаний у текстовому процесорі **Microsoft Word** (шрифтом **Times New Roman**, **14** пт, Вирівнювання: **по ширине**, Міжрядковий інтервал: **1,15**). На перевірку слід надати папку, назва якої (великими буквами) містить шифр групи, прізвище виконавця і номер практичного заняття (наприклад, 412ст-ІВАНЕНКО-П34). У цій папці слід розмістити:

- файл звіту (412ст-Іваненко-ПЗ4.docx або 412ст-Іваненко-ПЗ4.doc);
- папки та файли рішення MS Visual Studio (наприклад, папку "ConsoleApplication3" і файл "ConsoleApplication3.sln", де цифри у назві папки та назві файла мають збігатися з номером варіанта індивідуального завдання). Якщо створюється декілька рішень MS Visual Studio, то нумерувати відповідні папки та файли слід так: "ConsoleApplication3-1" і "ConsoleApplication3-1.sln", "ConsoleApplication3-2" і файл "ConsoleApplication3-2.sln" і т. д., де перша цифра (тут 3) у назві відповідає номеру варіанта, а друга цифра — номеру рішення MS Visual Studio у межах варіанта індивідуального завдання.

Папку зі звітом з практичного заняття слід здати в електронному вигляді старості академічної групи. Староста групи перевіряє правильність найменування папок звітів студентів, поміщує ці папки в папку під назвою 412ст-ПЗ4 і здає цю папку (в електронному вигляді) на перевірку викладачеві на початку наступного практичного заняття.

### Приклад виконання

**Завдання.** В одновимірному масиві, що складається з  $n$  цілочислових елементів, вчислити:

- 1) добуток елементів масиву з парними номерами;
- 2) суму елементів масиву, розташованих між першим та останнім нульовим елементами.

### Програмний код

```
using System;
using System.Collections.Generic;
using System.Linq;
using System.Text;

namespace ConsoleApplication1
{
    class Program
    {
        static void Main(string[] args)
        {
            Console.WriteLine("Уведіть кількість елементів n");

            int n = int.Parse(Console.ReadLine());
            int[] mas = new int[n];
            Console.WriteLine("- Заповнити масив вручну" + "- Заповнити масив випадковими числами від - 5 до 10" + "Оберіть потрібну дію: ");
```

```

string s = Console.ReadLine();

switch(s)
{
    case "1":
        Console.Clear();
        for //Розглядається 1-й випадок: масив заповнюється вручну
        {
            Console.Write("Mas[" + i + "]:");
            mas[i] = int.Parse(Console.ReadLine()); // Уведення елементів масиву,
їх оброблення і аналіз (Parse)
        }
        break; //Вихід з циклу з використанням оператора break. Використати
цей оператор є сенс після задоволення певної умови, інакше цикл завершиться на
першій ітерації.
        case "2":
            Random rnd = new Random(); //Ініціалізація нового екземпляра Random
класу за допомогою початкового значення
            for // Розглядається 2-й випадок: масив заповнюється випадковими
величинами від - 20 до 40, Length - властивість (загальна кількість елементів у
масиві)
            {
                mas[i] = rnd.Next(- 20, 40);
            }
            break;
        default: //Вирази значення за умовчанням: створює значення типу за
умовчанням.
            Console.WriteLine("Потребувалося ввести 1 або 2.");
            break; //Вихід з циклу з використанням оператора break.
        }

        Console.Clear(); //Видаляє з буфера консолі та її вікна інформацію, що
відображується.
        Console.WriteLine("Початковий масив: ");

        for // Length - властивість (загальна кількість елементів у масиві)
        {
            Console.Write(mas[i] + " ");
        }

        Int64 umn = 1;

        //тут потрібно застосувати i+=2, щоб обирати тільки парні елементи

```

```
for //i=2 початкове положення, індекс першого елемента, що  
переглядається  
{  
    //i<mas.Length - умова виходу з циклу  
    //i+2 крок руху по циклу, який рухається не по кожному елементу, а  
через один, тобто вибирає тільки парні номери.
```

```
    umn = umn * mas[i]; //добуток парних елементів  
}
```

```
int firstIndexNul = 0;  
int lastIndexNul = 0;  
int summ = 0;
```

```
//Йдемо по масиву з початку в кінець і, як тільки знаходиться нуль,  
зберігаємо індекс відповідного елемента і завершуємо цикл
```

```
for(int i = 0; i < mas.Length; i++)  
{  
    if(mas[i] == 0)  
    {  
        firstIndexNul = i;  
        break;  
    }  
}
```

```
//тут йдемо по масиву з кінця в початок і, як тільки знаходиться нуль,  
зберігаємо індекс відповідного елемента і завершуємо цикл
```

```
for(int i = mas.Length - 1; i >= 0; i--)  
{  
    if(mas[i] == 0)  
    {  
        lastIndexNul = i;  
        break;  
    }  
}
```

```
//після того, як виконано збереження першого і останнього індексів  
нульових елементів, потрібно просто пройти по масиву від першого індексу до  
останнього і порахувати суму елементів
```

```
for(int i = firstIndexNul; i < lastIndexNul; i++)  
{  
    summ += mas[i];  
}
```

```

    Console.WriteLine("Добуток елементів масиву з парними номерами: " +
    umn);
    Console.WriteLine("Сума елементів, розташованих між першим і останнім
    нульовим: " + summ);
    Console.WriteLine("Перетворений масив: ");

    for(int i = 1; i < mas.Length; i++)
    {
        if(mas[i] >= 0)
        {
            Console.Write(mas[i] + " ");
        }
    }

    for(int i = 0; i < mas.Length; i++)
    {
        if(mas[i] < 0)
        {
            Console.Write(mas[i] + " ");
        }
    }

    Console.ReadKey();
}
}
}

```

```

1. Console.Write("- Заповнити масив вручну" + "- Заповнити масив випадковими
числами від - 5 до 10" + "Оберіть потрібну дію: ");
string s = Console.ReadLine();
switch(s)

```

### Примітки

З користувачем програми ведеться діалог, що надає можливість вибрати спосіб заповнення масиву:

1-й спосіб — ручне заповнення масиву користувачем;

2-й спосіб — випадкове заповнення масиву.

Залежно від вибраного користувачем випадку масив або заповнюється випадковими числами, або введенням елементів з клавіатури користувачем.

Частковим, але важливим випадком вибору з декількох варіантів є ситуація, при якій вибір варіанта визначається значеннями певного виразу. Відповідний оператор switch успадковано С# з невеликими змінами в

синтаксисі від C++. Отже, спочатку обчислюється значення switch-виразу. Потім воно по черзі в порядку дотримання case порівнюється на збіг з константними виразами. Як тільки досягнуто збіг, виконується відповідна послідовність операторів case-гілки. Оскільки останній оператор цієї послідовності є оператором переходу (найчастіше це оператор break), то зазвичай він завершує виконання оператора switch. Використання операторів переходу — не завжди вдала ідея. Таким оператором може бути оператор goto, що передає управління іншій case-гілці, яка, своєю чергою, може передати управління ще будь-куди з отриманням втрати добре структурованої послідовності операторів.

При цьому семантика ускладнюється ще й тим, що case-гілка може бути порожньою послідовністю операторів. Тоді у разі збігу константного вираження цієї гілки зі switch-значенням виразу виконуватиметься перша не порожня послідовність чергової case-гілки. Якщо значення switch-виразу не збігаються з жодним константним виразом, то виконується послідовність операторів гілки default. Якщо ж такої гілки немає, то оператор switch еквівалентний порожньому операторові.

## **Висновки**

У результаті виконання практичного заняття отримано навички написання, налагодження і компіляції програм оброблення одновимірних масивів МОВОЮ C# у середовищі розроблення Microsoft Visual Studio.

## **Контрольні запитання**

1. Оператор передачі управління goto: призначення та формат.
2. Оператор передачі управління break: призначення та формат.
3. Оператор передачі управління continue: призначення та формат.
4. Оператор передачі управління return: призначення та формат.
5. Базові конструкції структурного програмування.



## Практичне заняття № 5

### РЕАЛІЗАЦІЯ ОБЧИСЛЮВАЛЬНИХ ПРОЦЕСІВ ОБРОБЛЕННЯ ДВОВИМІРНИХ МАСИВІВ МОВОЮ ПРОГРАМУВАННЯ C# У СЕРЕДОВИЩІ РОЗРОБЛЕННЯ MICROSOFT VISUAL STUDIO

**Мета заняття:** отримання навичок написання, налагодження і компіляції програм, що реалізують обчислювальні алгоритми оброблення двовимірних масивів мовою C# у середовищі розроблення Microsoft Visual Studio.

#### Апаратне, програмне та методичне забезпечення

1. Засоби обчислювальної техніки (ПК).
2. Інсталяційний пакет програмного забезпечення Microsoft Visual Studio.
3. Пакет індивідуальних завдань.

#### Порядок виконання

1. Вивчення основних теоретичних положень [2, с. 89 - 151].
2. Повторення основних теоретичних понять про оброблення елементів масивів і рядків з використанням мови програмування C# [2, с. 126 - 151].
3. Розроблення мовою C# програм обчислювальних процесів оброблення двовимірних масивів згідно з варіантами індивідуальних завдань [2, с. 389 - 393].
4. Підготовка контрольного звіту за результатами виконання практичного заняття.

Примітка. У кожному індивідуальному завданні мається на увазі, що програма повинна забезпечувати оброблення будь-якої кількості елементів масиву, що вводяться з клавіатури, не за збільшенням або убуванням, а в довільному порядку.

#### Зміст звіту

1. Титульний лист.
2. Індивідуальний варіант завдання.
3. Програмні коди реалізації мовою C# у середовищі розроблення Microsoft Visual Studio обчислювальних процесів згідно з індивідуальним варіантом завдання.
4. Висновки (про те, що виконано на практичному занятті та які навички отримано).

Звіт має бути виконаний у текстовому процесорі **Microsoft Word** (шрифтом **Times New Roman, 14 пт**, Вирівнювання: **по ширине**, Міжрядковий інтервал: **1,15**). На перевірку слід надати папку, назва якої

(великими буквами) містить шифр групи, прізвище виконавця і номер практичного заняття (наприклад, 412ст-ІВАНЕНКО-П35). У цій папці слід розмістити:

- файл звіту (412ст-Іваненко-П35.docx або 412ст-Іваненко-П35.doc);
- папки та файли рішення MS Visual Studio (наприклад, папку "ConsoleApplication3" і файл "ConsoleApplication3.sln", де цифри у назві папки та назві файла мають збігатися з номером варіанта індивідуального завдання). Якщо створюється декілька рішень MS Visual Studio, то нумерувати відповідні папки та файли слід так: "ConsoleApplication3-1" і "ConsoleApplication3-1.sln", "ConsoleApplication3-2" і файл "ConsoleApplication3-2.sln" і т. д., де перша цифра (тут 3) у назві відповідає номеру варіанта, а друга цифра — номеру рішення MS Visual Studio у межах варіанта індивідуального завдання.

Папку зі звітом з практичного заняття слід здати в електронному вигляді старості академічної групи. Староста групи перевіряє правильність найменування папок звітів студентів, поміщує ці папки в папку під назвою 412ст-П35 і здає цю папку (в електронному вигляді) на перевірку викладачеві на початку наступного практичного заняття.

### Приклад виконання

**Завдання.** Виконати циклічний зсув елементів прямокутної матриці на  $n$  елементів вправо або вниз (залежно від уведеного режиму),  $n$  може бути більше кількості елементів у рядку або стовпці.

### Програмний код

```
using System;
using System.Collections.Generic;
using System.Text;

namespace П35
{
    class Program
    {
        static void Main(string[] args)
        {
            const int z = 4, x = 5; //Оголошення констант
            int[,] a = new int[z, x] { //Створення масиву
                {1, 2, 3, 4, 5},
                {6, 7, 8, 9, 10},
                {11, 12, 13, 14, 15},
            };
        }
    }
}
```

```

        {16,17,18,19, 20}
    };
    Console.WriteLine("Вихідний масив:"); // Виведення
повідомлення
    for (int i = 0; i < z; ++i) // Виведення вихідного
масива
    {
        for (int j = 0; j < x; ++j)
            Console.Write("\t" + a[i, j]);
        Console.WriteLine();
    }
    Console.Write("Уведіть n: "); // Виведення
повідомлення
    string buf1 = Console.ReadLine(); // Зчитування n
    int n = int.Parse(buf1); // Переведення n в int
    Console.Write("Оберіть режим зсуву ( 1 - праворуч, 2
- униз ): "); // Виведення повідомлення
    string buf2 = Console.ReadLine(); // Зчитування buf2
    int sd = int.Parse(buf2); // Переведення в int buf2
    int inew = 0, jnew = 0; // Створення змінних
    int g = 0;
    if (sd == 1) // Зсув праворуч
    {
        Console.WriteLine("Зсув праворуч:"); // Виведення
повідомлення
        int nnew = x - n % x; // Створення змінної номера
НОВОГО СТОВПЦЯ
        for (int i = 0; i < z; ++i) // Цикл переміщення
СТОВПЦІВ
        {
            inew = i;
            for (int j = 0; j < x; ++j)
            {
                jnew = (j + nnew) % x;
                Console.Write("\t" + a[inew, jnew]);
                // Виведення одиничного елемента матриці
            }
            Console.WriteLine(); // Виведення порожнього
рядка
        }
    }
    else if (sd == 2) // Переміщення униз
    {

```

```

        Console.WriteLine("Зсув униз:");//Виведення
повідомлення
        int nnew = z - n % z;//Створення змінної номера
нового рядка
        for (int i = 0; i < z; ++i)//Цикл переміщення
рядків
        {
            for (int j = 0; j < x; ++j)
            {
                jnew = j;
                inew = (i + nnew) % z;
                Console.Write("\t" + a[inew, jnew]);
//Виведення одиничного елемента матриці
            }
            Console.WriteLine();//Виведення порожнього
рядка
        }
    }
    else Console.WriteLine("Помилка!");//Виведення
повідомлення "Помилка!"
    Console.ReadKey();//Зчитування будь-якої натиснутої
клавiші
}
}
}

```

## Висновки

У результаті виконання практичного заняття отримано навички написання, налагодження і компіляції програм, що реалізують обчислювальні алгоритми оброблення двовимірних масивів МОВОЮ С# у середовищі розроблення Microsoft Visual Studio.

## Контрольні запитання

1. Оператор передачі управління goto: призначення та формат.
2. Оператор передачі управління break: призначення та формат.
3. Оператор передачі управління continue: призначення та формат.
4. Оператор передачі управління return: призначення та формат.
5. Базові конструкції структурного програмування.
6. Відмінні особливості оброблення елементів двовимірних масивів порівняно з обробленням одновимірних масивів.
7. Базові алгоритми, що необхідно застосовувати для оброблення елементів двовимірних масивів.

## Практичне заняття № 6

### РОЗРОБЛЕННЯ НАЙПРОСТІШИХ КЛАСІВ МОВОЮ ПРОГРАМУВАННЯ C# У СЕРЕДОВИЩІ РОЗРОБЛЕННЯ MICROSOFT VISUAL STUDIO

**Мета заняття:** отримання навичок написання, налагодження і компіляції програм, що реалізують класи мовою C# в середовищі розроблення Microsoft Visual Studio.

#### Апаратне, програмне та методичне забезпечення

1. Засоби обчислювальної техніки (ПК).
2. Інсталяційний пакет програмного забезпечення Microsoft Visual Studio.
3. Пакет індивідуальних завдань.

#### Порядок виконання

1. Повторення загальних теоретичних положень з Microsoft Visual Studio [1, с. 7 - 38].
2. Повторення основних положень про платформу .NET [2, с. 8 - 68].
3. Повторення основних теоретичних положень із синтаксису операторів розгалуження [2, с. 69 - 75].
4. Повторення основних теоретичних положень із синтаксису операторів циклу [2, с. 75 - 88].
5. Вивчення основних теоретичних положень з оброблення виняткових ситуацій [2, с. 89 - 99].
6. Вивчення основних теоретичних понять про класи мови програмування C# [2, с. 100 - 125].
7. Розроблення мовою C# програм, що реалізують класи згідно з варіантами індивідуальних завдань [2, с. 381 - 385].
8. Підготовка контрольного звіту за результатами виконання практичного заняття.

#### Зміст звіту

1. Титульний лист.
2. Індивідуальний варіант завдання.
3. Програмні коди реалізації мовою C# у середовищі розроблення Microsoft Visual Studio обчислювальних процесів згідно з індивідуальним варіантом завдання.
4. Висновки (про те, що виконано на практичному занятті та які навички отримано).

Звіт має бути виконаний у текстовому процесорі **Microsoft Word** (шрифтом **Times New Roman**, **14** пт, Вирівнювання: **по ширине**, Міжрядковий інтервал: **1,15**). На перевірку слід надати папку, назва якої

(великими буквами) містить шифр групи, прізвище виконавця і номер практичного заняття (наприклад, 412ст-ІВАНЕНКО-ПЗ6). У цій папці слід розмістити:

- файл звіту (412ст-Іваненко-ПЗ6.docx або 412ст-Іваненко-ПЗ6.doc);
- папки та файли рішення MS Visual Studio (наприклад, папку "ConsoleApplication3" і файл "ConsoleApplication3.sln", де цифри у назві папки та назві файла мають збігатися з номером варіанта індивідуального завдання). Якщо створюється декілька рішень MS Visual Studio, то нумерувати відповідні папки та файли слід так: "ConsoleApplication3-1" і "ConsoleApplication3-1.sln", "ConsoleApplication3-2" і файл "ConsoleApplication3-2.sln" і т. д., де перша цифра (тут 3) у назві відповідає номеру варіанта, а друга цифра — номеру рішення MS Visual Studio у межах варіанта індивідуального завдання.

Папку зі звітом з практичного заняття слід здати в електронному вигляді старості академічної групи. Староста групи перевіряє правильність найменування папок звітів студентів, поміщує ці папки в папку під назвою 412ст-ПЗ5 і здає цю папку (в електронному вигляді) на перевірку викладачеві на початку наступного практичного заняття.

### Приклад виконання

**Завдання.** Описати найпростіший клас і написати програму, що реалізує створення на підставі введених даних інформаційної довідки автомобіля.

### Програмний код

```
using System;

namespace HelloWorld
{
    class Auto //1
    {
        public string Marka;//2
        public string model;//3
        public int age;//4
        public string sostoyanie ;//5
        public string price;//6
    }
    class Program//7
    {
        static void Main(string[] args)//8
```

```

{
    Auto Lada = new Auto();//9
    Console.WriteLine("Уведіть марку авто");//10
    Lada.Marka = Console.ReadLine();//11
    Console.WriteLine("Уведіть модель авто");//12
    Lada.model = Console.ReadLine();//13
    Console.WriteLine("Уведіть рік випуску авто");//14
    Lada.age = (2019 -
    Convert.ToInt32(Console.ReadLine()));//15
    Console.WriteLine("Уведіть дані про стан авто");//16
    Lada.sostoyanie = Console.ReadLine();//17
}
Console.WriteLine("Уведіть вартість авто");//18
Lada.price = Console.ReadLine();//19
Console.WriteLine("Натисніть будь-яку клавішу для
вивода інформації про авто");//20
Console.ReadKey(); //21
Console.WriteLine("\b \b");//22
Console.WriteLine("Марка: " + Lada.Marka);//23
Console.WriteLine("Модель: " + Lada.model);
Console.WriteLine("Вік: " + Lada.age);
Console.WriteLine("Стан: " + Lada.sostoyanie);
Console.WriteLine("Вартість: " + Lada.price);
Console.ReadKey();
}
}
}

```

### Коментарі до програмного коду

- 1 - Створення класу "Auto".
- 2 - Створення поля "Marka" у класі "Auto" .
- 3 - Створення поля "model" у класі "Auto" .
- 4 - Створення поля "age" у класі "Auto" .
- 5 - Створення поля "sostoyanie" у класі "Auto".
- 6 - Створення поля "price" у класі "Auto".
- 7 - Вхід у клас "Program".
- 8 - Точка входу в програму.
- 9 - Створення об'єкта "Lada".
- 10 - Виведення на екран "Уведіть марку авто".

- 11 - Присвоювання об'єкту значення "Lada.Marka", уведеного з клавіатури.
- 12 - Виведення на екран "Уведіть модель авто".
- 13 - Присвоювання об'єкту "Lada.model" значення, уведеного з клавіатури.
- 14 - Виведення на екран "Уведіть рік випуску авто".
- 15 - Присвоювання об'єкту "Lada.model" значення, уведеного з клавіатури.
- 16 - Виведення на екран "Уведіть стан авто".
- 17 - Присвоювання об'єкту "Lada.sostoyanie" значення, уведеного з клавіатури.
- 18 - Виведення на екран "Уведіть вартість авто".
- 19 - Присвоювання об'єкту "Lada.price" значення, уведеного з клавіатури.
- 20 - Виведення на екран "Натисніть будь-яку клавішу для виведення інформації про авто".
- 21 - Зчитування значення з будь-якої клавіші, натиснутої на клавіатурі.
- 22 - Стирання з консолі значення введеної з будь-якої клавіші клавіатури.
- 23 - Виведення результатів.

### **Висновки**

У результаті виконання практичного заняття отримано навички написання, налагодження і компіляції програм, що реалізують класи мовою C# у середовищі розроблення Microsoft Visual Studio.

### **Контрольні запитання**

1. Узагальнене поняття класу. Поняття й призначення екземплярів (об'єктів), даних, функцій та подій класу.
2. Синтаксична форма опису класу мовою C#.
3. Специфікатори класу: призначення, приклади.
4. Призначення та відмінності методів класу (статичних методів) і методів екземплярів (методів).
5. Елементи класу та їх призначення.
6. Структурна схема складу класу.



## Практичне заняття № 7

### РЕАЛІЗАЦІЯ ОБЧИСЛЮВАЛЬНИХ ПРОЦЕСІВ ОБРОБЛЕННЯ РЯДКІВ МОВОЮ ПРОГРАМУВАННЯ C# У СЕРЕДОВИЩІ РОЗРОБЛЕННЯ MICROSOFT VISUAL STUDIO

**Мета заняття:** отримання навичок написання, налагодження і компіляції програм, що реалізують алгоритми оброблення рядків мовою C# у середовищі розроблення Microsoft Visual Studio.

#### Апаратне, програмне та методичне забезпечення

1. Засоби обчислювальної техніки (ПК).
2. Інсталяційний пакет програмного забезпечення Microsoft Visual Studio.
3. Пакет індивідуальних завдань.

#### Порядок виконання

1. Вивчення основних теоретичних положень [2, с. 126 - 151].
2. Розроблення мовою C# програм обчислювальних процесів оброблення рядків згідно з варіантами індивідуальних завдань [2, с. 393 - 395].
3. Підготовка контрольного звіту за результатами виконання практичного заняття.

#### Зміст звіту

1. Титульний лист.
2. Індивідуальний варіант завдання.
3. Програмні коди реалізації мовою C# у середовищі розроблення Microsoft Visual Studio обчислювальних процесів згідно з індивідуальним варіантом завдання.
4. Висновки (про те, що виконано на практичному занятті та які навички отримано).

Звіт має бути виконаний у текстовому процесорі **Microsoft Word** (шрифтом **Times New Roman**, **14** пт, Вирівнювання: **по ширине**, Міжрядковий інтервал: **1,15**). На перевірку слід надати папку, назва якої (великими буквами) містить шифр групи, прізвище виконавця і номер практичного заняття (наприклад, 412ст-ІВАНЕНКО-П37). У цій папці слід розмістити:

- файл звіту (412ст-Іваненко-П37.docx або 412ст-Іваненко-П37.doc);
- папки та файли рішення MS Visual Studio (наприклад, папку "ConsoleApplication3" і файл "ConsoleApplication3.sln", де цифри у назві папки та назві файла мають збігатися з номером варіанта індивідуального завдання). Якщо створюється декілька рішень MS

Visual Studio, то нумерувати відповідні папки та файли слід так: "ConsoleApplication3-1" і "ConsoleApplication3-1.sln", "ConsoleApplication3-2" і файл "ConsoleApplication3-2.sln" і т. д., де перша цифра (тут 3) у назві відповідає номеру варіанта, а друга цифра — номеру рішення MS Visual Studio у межах варіанта індивідуального завдання.

Папку зі звітом з практичного заняття слід здати в електронному вигляді старості академічної групи. Староста групи перевіряє правильність найменування папок звітів студентів, поміщує ці папки в папку під назвою 412ст-П37 і здає цю папку (в електронному вигляді) на перевірку викладачеві на початку наступного практичного заняття.

### Приклад виконання

**Завдання.** Написати програму, яка зчитує текст з файла і виводить на екран тільки ті рядки, що містять двозначні числа.

### Програмний код

```
using System;
using System.Collections.Generic;
using System.Linq;
using System.Text.RegularExpressions;

namespace ConsoleApplication1
{
    class Program
    {
        static void Main(string[] args)
        {
            System.IO.FileInfo f = new System.IO.FileInfo("1.txt");

            string str = string.Empty;
            string read = string.Empty;

            if(f.Exists)
                using (System.IO.StreamReader sr = new System.IO.StreamReader
                    (f.FullName, Encoding.Default))
                {
                    while((str = sr.ReadLine()) != null)
                    {
                        read += str;
                    }
                    string[] split = read.Split(' ');
                    foreach(string s in split)
```

```

        {
            if(System.Text.RegularExpressions.Regex.IsMatch(s, @"\d{2}$"))
            {
                Console.WriteLine(s + " ");
            }
        }
    }
    Console.ReadLine();
}
}
}

```

Примітки. 1. Для того щоб працювати з регулярними виразами, необхідно на початку програми підключити простір імен `using System.Text.RegularExpressions`.

2. Розроблена програма забезпечує зчитування і подальше виведення необхідних рядків з файла, який має бути розташованим у теці `ConsoleApplication`.

## Висновки

У результаті виконання практичного заняття отримано навички написання, налагодження і компіляції програм для оброблення рядків мовою C# у середовищі розроблення Microsoft Visual Studio.

## Контрольні запитання

1. Суттєвість та особливості створення масивів у C#.
2. Одновимірні масиви: варіанти та приклади описання.
3. Прямокутні масиви: варіанти та приклади описання.
4. Ступінчасті масиви: варіанти та приклади описання.
5. Призначення та елементи класу `System.Array`.
6. Оператор `foreach`: призначення, формат, особливості роботи з масивами об'єктів.
7. Призначення типу `char`. Основні методи класу `System.Char`.
8. Призначення та особливості застосування масивів символів.
9. Способи створення рядків типу `string`. Операції, визначені для рядків типу `string`.
10. Призначення та елементи класу `System.String`.
11. Особливості форматування рядків. Метод `Format`.
12. Призначення та способи створення рядків типу `StringBuilder`.
13. Призначення та елементи класу `System.Text.StringBuilder`.
14. Призначення та елементи класу `System.Random`. Конструкції створення екземплярів класу `Random`.

## Практичне заняття № 8

### РОЗРОБЛЕННЯ КЛАСІВ З ОПЕРАЦІЯМИ МОВОЮ ПРОГРАМУВАННЯ C# У СЕРЕДОВИЩІ РОЗРОБЛЕННЯ MICROSOFT VISUAL STUDIO

**Мета заняття:** отримання навичок написання, налагодження і компіляції програм розроблення класів із заданими операціями мовою C# у середовищі розроблення Microsoft Visual Studio.

#### Апаратне, програмне та методичне забезпечення

1. Засоби обчислювальної техніки (ПК).
2. Інсталяційний пакет програмного забезпечення Microsoft Visual Studio.
3. Пакет індивідуальних завдань.

#### Порядок виконання

1. Вивчення основних теоретичних положень [2, с. 152 - 171].
2. Розроблення мовою C# програм розроблення класів із заданими операціями згідно з варіантами індивідуальних завдань [2, с. 395 - 400].
3. Підготовка контрольного звіту за результатами виконання практичного заняття.

#### Зміст звіту

1. Титульний лист.
2. Індивідуальний варіант завдання.
3. Програмні коди реалізації мовою C# у середовищі розроблення Microsoft Visual Studio обчислювальних процесів згідно з індивідуальним варіантом завдання.
4. Висновки (про те, що виконано на практичному занятті та які навички отримано).

Звіт має бути виконаний у текстовому процесорі **Microsoft Word** (шрифтом **Times New Roman, 14 пт**, Вирівнювання: **по ширине**, Міжрядковий інтервал: **1,15**). На перевірку слід надати папку, назва якої (великими буквами) містить шифр групи, прізвище виконавця і номер практичного заняття (наприклад, 412ст-ІВАНЕНКО-П38). У цій папці слід розмістити:

- файл звіту (412ст-Іваненко-П38.docx або 412ст-Іваненко-П38.doc);
- папки та файли рішення MS Visual Studio (наприклад, папку "ConsoleApplication3" і файл "ConsoleApplication3.sln", де цифри у назві папки та назві файла мають збігатися з номером варіанта індивідуального завдання). Якщо створюється декілька рішень MS Visual Studio, то нумерувати відповідні папки та файли слід так:

"ConsoleApplication3-1" і "ConsoleApplication3-1.sln", "ConsoleApplication3-2" і файл "ConsoleApplication3-2.sln" і т. д., де перша цифра (тут 3) у назві відповідає номеру варіанта, а друга цифра — номеру рішення MS Visual Studio у межах варіанта індивідуального завдання.

Папку зі звітом з практичного заняття слід здати в електронному вигляді старості академічної групи. Староста групи перевіряє правильність найменування папок звітів студентів, поміщує ці папки в папку під назвою 412ст-П38 та здає цю папку (в електронному вигляді) на перевірку викладачеві на початку наступного практичного заняття.

### Приклад виконання

**Завдання.** Описати клас для роботи з одновимірним масивом рядків фіксованої довжини.

Забезпечити такі можливості:

- 1) завдання довільних меж індексів при створенні об'єкта;
- 2) звернення до окремого рядка масиву за індексом з контролем виходу за межі масиву;
- 3) виконання операцій поелементного зчеплення двох масивів з утворенням нового масиву;
- 4) виконання злиття двох масивів з виключенням елементів, що повторюються;
- 5) виведення на екран елемента масиву за заданим індексом, а також усього масиву.

Написати програму, що демонструє усі розроблені елементи класу.

### Програмний код

```
using System;
using System.Collections.Generic;
using System.Linq;
using System.Text;

namespace ConsoleApplication1
{
    // створюємо клас-контейнер для стрингів (реалізовується ICloneable),
    //щоб не псувати початкові масиви під час об'єднання
    public class MyArray: ICloneable
    {
        private int firstIndex, lastIndex;
        string[] arr; //оголошуємо внутрішній службовий масив
        public MyArray // конструктор, що викликається при створенні об'єкта
        {
            if(firstIndex >= lastIndex)
```

```

        throw new ArgumentException("Перший індекс має бути меншим за
останній");
        this.firstIndex = firstIndex; // ініціалізували межі нашого контейнера
        this.lastIndex = lastIndex;
        arr = new string[lastIndex - firstIndex + 1]; // ініціалізували внутрішній масив
    }
    public int FirstIndex // властивість, що повертає індекс першого елемента
нашого контейнера
    {
        get { return firstIndex; }
    }
    public int LastElemPos // властивість, що повертає індекс останнього елемента
нашого контейнера
    {
        get { return arr.Length + firstIndex; }
    }
    public string this[int pos] // для того щоб звертатися до нашого контейнера
синтаксисом MyArr[i], створюємо індексатор
    {
        get
        {
            if // перевіряємо межі масиву, якщо вишли за межі - генеруємо
ВИКЛЮЧЕННЯ
                throw new ArgumentOutOfRangeException("Вийшли за межі масиву :(
");
            else
                return arr[pos - firstIndex];
        }
        set
        {
            if // перевіряємо межі масиву, якщо вишли за межі - генеруємо
ВИКЛЮЧЕННЯ
                throw new ArgumentOutOfRangeException("Вийшли за межі масиву
:(");
            else
                arr[pos - firstIndex] = value;
        }
    }
    public object Clone() // реалізовується клонування
    {
        MyArray copy = new MyArray(this.firstIndex, this.lastIndex);
        for(int i = 0; i < copy.arr.Length; i++)
        {
            copy.arr[i] = this.arr[i];
        }
    }

```

```

    }
    return copy;
}
public static MyArray Concat //об'єднуємо масиви, Concat - конкатенація
рядків
{
    //клонуємо вхідні масиви, щоб не спотворювати дані
    MyArray temp1(MyArray) arr1.Clone();
    MyArray temp2(MyArray) arr2.Clone();

    int temp = temp1.arr.Length;
    Array.Resize(ref temp1.arr, temp1.arr.Length + temp2.arr.Length);
    temp1.lastIndex += temp2.arr.Length;
    temp2.arr.CopyTo(temp1.arr, temp);
    return temp1;
}

public static MyArray Merge(MyArray pArr1, MyArray pArr2)
{
    MyArray arr1(MyArray) pArr1.Clone();
    MyArray arr2(MyArray) pArr2.Clone();

    int sizeForAdd = arr2.arr.Length;
    for(int i = 0; i < arr1.arr.Length; i++)
    {
        for(int j = 0; j < arr2.arr.Length; j++)
        {
            if // якщо є збіги, елементу, що повторюється, присвоюється null
            {
                arr2.arr[j] = null;
                sizeForAdd--; // декрементация зменшує елемент на одиницю, тобто
розмір об'єданого масиву зменшується на ту кількість елементів, які збіглись
            }
        }
    }

    int temp = arr1.arr.Length;
    Array.Resize(ref arr1.arr, arr1.arr.Length + sizeForAdd);

    for(int i = 0; i < arr2.arr.Length; i++)
    {
        if(arr2.arr[i] != null)
        {
            arr1.arr[temp] = arr2.arr[i];

```

```

        temp++;
    }
}
arr1.lastIndex += sizeForAdd;// a+=b: a= a+b
return arr1;
}
}

```

```
class Program
```

```

{
    static void Main(string[] args)
    {

        MyArray arr = new MyArray(- 5, 3);

        MyArray arr2 = new MyArray(4, 7);

        for(int i = arr.FirstIndex; i < arr.LastElemPos; i++)
        {
            arr[i] = "рядок масиву 1: " + i;
        }

        for(int i = arr2.FirstIndex; i < arr2.LastElemPos; i++)
        {
            arr2[i] = "рядок масиву 2: " + i;
        }

        Console.WriteLine("Усі елементи першого масиву");
        for(int i = arr.FirstIndex; i < arr.LastElemPos; i++)
        {
            Console.WriteLine(arr[i]);
        }

        Console.WriteLine();
        Console.WriteLine("Усі елементи другого масиву");
        for(int i = arr2.FirstIndex; i < arr2.LastElemPos; i++)
        {
            Console.WriteLine(arr2[i]);
        }

        Console.WriteLine();
        Console.WriteLine("Елемент з індексом 1 першого масиву - {0}" + " ",
arr[1]);
        Console.WriteLine();
    }
}

```



```

MyArray concatArr = MyArray.Concat(arr, arr2);
Console.WriteLine("Об'єднаний масив");

for(int i = concatArr.FirstIndex; i < concatArr.LastElemPos; i++)
{
    Console.WriteLine(concatArr[i]);
}
Console.WriteLine();

MyArray MergeArr = MyArray.Merge(arr, arr2); // поелементне зчеплення

Console.ReadLine();

}
}
}

```

Примітки. 1. Для того щоб скористатися масивом у програмі, потрібно виконати двоетапну процедуру, оскільки масиви реалізовані у вигляді об'єктів. По-перше, треба створити екземпляр масиву, використовуючи оператор `new`.

2. Індексатори дозволяють індексувати об'єкти і використати їх як масиви. Фактично індексатори дозволяють нам створювати спеціальні сховища об'єктів або колекції.

3. `if(firstIndex >= lastIndex) throw new ArgumentException("перший індекс має бути меншим за останній");`

`this.firstIndex = firstIndex; // ініціалізували межі створеного контейнера`

`this.lastIndex = lastIndex;`

`arr = new string[lastIndex - firstIndex + 1]; // ініціалізували внутрішній масив.`

4. Якщо перший індекс більше другого індексу або дорівнює йому, то розглядаємо виключення, яке викликається, коли один з аргументів, наданих методу, недійсний.

5. Властивість у C# — це член класу, який надає зручний механізм доступу до поля класу (читання поля і запис). Властивість є чимось середнім між полем і методом класу. При використанні властивості ми звертаємося до нього, як до поля класу, але насправді компілятор перетворює це звернення на виклик відповідного неявного методу. Такий метод називається асесор (*accessor*). Існує два таких методи: `get` (для отримання даних) і `set` (для запису даних). Оголошення простої властивості має таку структуру:

```

    [модифікатор доступу] [тип] [ім'я_властивості]
{
    get

```

```
{  
  // тіло асесора для читання з поля  
}
```

За допомогою `get { return firstIndex; }` можна контролювати читання.

Стандартне визначення властивості містить блоки `get` і `set`. У блоці `get` повертають значення поля, а у блоці `set` встановлюють. Параметр `value` є передаваним значенням.

6. `public static MyArray Merge(MyArray pArr1, MyArray pArr2)`

Сортування злиття ґрунтується на парадигмі «розділяй і володарюй». Його найгірший час роботи має нижчий порядок зростання, ніж сортування вставки. Оскільки маємо справу з субзадачами, то визначаємо кожний підпроцес як сортування підматриці  $A [p \dots r]$ . Спочатку  $p = 1$  і  $r = n$ , але ці значення змінюються у міру того, як ми рекурсивно перевіряємо підпроцеси.

```
7. if(arr1.arr[i] == arr2.arr[j])  
arr2.arr[j] = null;  
sizeForAdd--;
```

Якщо при поелементному злитті елементів двох масивів знаходяться збіги, то елементам, що повторюються, програма присвоює значення 0 і опускає його. Унаслідок цього об'єднаний масив зменшується на ту кількість елементів, які збіглися.

## Висновки

У результаті виконання практичного заняття отримано навички написання, налагодження та компіляції програм розроблення класів із заданими операціями мовою C# у середовищі розроблення Microsoft Visual Studio.

## Контрольні запитання

1. Поняття про класи C#: визначення, елементи класу, описання класу.
2. Специфікатори класу.
3. Екземпляри (об'єкти) класу, статичні елементи (дані) і дані екземпляра.
4. Призначення методів класу (статичних методів) і методів екземпляра (або просто методів).
5. Визначення усіх елементів класу: константи, поля, методи, властивості, конструктори, деструктори, індексатори, операції, події, типи.

## Практичне заняття № 9

### РОЗРОБЛЕННЯ КЛАСІВ З НАСЛІДУВАННЯМ МОВОЮ ПРОГРАМУВАННЯ C# У СЕРЕДОВИЩІ РОЗРОБЛЕННЯ MICROSOFT VISUAL STUDIO

**Мета заняття:** отримання навичок написання, налагодження і компіляції програм розроблення класів з наслідуванням мовою C# у середовищі розроблення Microsoft Visual Studio.

#### Апаратне, програмне та методичне забезпечення

1. Засоби обчислювальної техніки (ПК).
2. Інсталяційний пакет програмного забезпечення Microsoft Visual Studio.
3. Пакет індивідуальних завдань.

#### Порядок виконання

1. Вивчення основних теоретичних положень [2, с. 172 - 187].
2. Розроблення мовою C# програм розроблення класів з наслідуванням згідно з варіантами індивідуальних завдань [2, с. 400 - 405].
3. Підготовка контрольного звіту за результатами виконання практичного заняття.

#### Зміст звіту

1. Титульний лист.
2. Індивідуальний варіант завдання.
3. Програмні коди реалізації мовою C# у середовищі розроблення Microsoft Visual Studio обчислювальних процесів згідно з індивідуальним варіантом завдання.
4. Висновки (про те, що виконано на практичному занятті та які навички отримано).

Звіт має бути виконаний у текстовому процесорі **Microsoft Word** (шрифтом **Times New Roman**, **14** пт, Вирівнювання: **по ширине**, Міжрядковий інтервал: **1,15**). На перевірку слід надати папку, назва якої (великими буквами) містить шифр групи, прізвище виконавця і номер практичного заняття (наприклад, 412ст-ІВАНЕНКО-П39). У цій папці слід розмістити:

- файл звіту (412ст-Іваненко-П39.docx або 412ст-Іваненко-П39.doc);
- папки та файли рішення MS Visual Studio (наприклад, папку "ConsoleApplication3" і файл "ConsoleApplication3.sln", де цифри у назві папки та назві файла мають збігатися з номером варіанта індивідуального завдання). Якщо створюється декілька рішень MS

Visual Studio, то нумерувати відповідні папки та файли слід так: "ConsoleApplication3-1" і "ConsoleApplication3-1.sln", "ConsoleApplication3-2" і файл "ConsoleApplication3-2.sln" і т. д., де перша цифра (тут 3) у назві відповідає номеру варіанта, а друга цифра — номеру рішення MS Visual Studio у межах варіанта індивідуального завдання.

Папку зі звітом з практичного заняття слід здати в електронному вигляді старості академічної групи. Староста групи перевіряє правильність найменування папок звітів студентів, поміщує ці папки в папку під назвою 412ст-ПЗ9 і здає цю папку (в електронному вигляді) на перевірку викладачеві на початку наступного практичного заняття.

### Приклад виконання

**Завдання.** Створити абстрактний клас Vehicle (транспортний засіб). На його основі реалізувати класи Plane (літак), Car (машина) і Ship (кораблик). Класи повинні забезпечувати можливість задавати і отримувати координати і параметри засобів пересування (ціну, швидкість, рік випуску тощо) за допомогою властивостей.

Для літака мають бути визначені висота, для літака і корабля — кількість пасажирів, для корабля — порт прописки. Динамічні характеристики описати за допомогою методів.

### Програмний код

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

namespace ConsoleApplication2
{
    class Program
    {
        abstract class Vehicle
        {
            // виключення для класів спадкоємців
            protected Exception OutOfMaxBorder = new Exception("Виключення: перевищена максимальна межа");
            protected Exception NonBellowZero = new Exception("Виключення: введене значення не може бути негативним!");

            protected int price, maxspeed, year;
```

```

// властивість "Вартість"
public int Price
{
    get { return price; }
    set { if(value > 0) price = value; }
}
// властивість "Максимальна швидкість"
public int Maxspeed
{
    get { return maxspeed; }
    set { if(value > 0) maxspeed = value; }
}
// властивість "Рік випуску"
public int Year
{
    get { return year; }
    set { if(value <= DateTime.Today.Year) year = value; }
}

//конструктор класу

public Vehicle(int price, int maxspeed, int year)
{
    this.Price = price;
    this.Maxspeed = maxspeed;
    this.Year = year;
}

    class Ship: Vehicle
{
    //оригінальне поле для байків
    public int MaxPassengers { get; set; }

    //конструктор + конструктор базового класу
    public Ship(int prise, int maxspeed, int year, int MaxPassengers) : base(prise,
maxspeed, year)
    {
        this.MaxPassengers = MaxPassengers;
    }
}

    class Car: Vehicle
{
    //оригінальне поле для авто

```

```

public int Power { get; set; }

//конструктор + успадкований конструктор
public Car(int prise, int maxspeed, int year, int Power)
    : base(prise, maxspeed, year)
{
    this.Power = Power;
}
}

    class Plane: Vehicle
{
public int MaxCapacity { get; set; }

public Plane(int prise, int maxspeed, int year, int MaxCapacity)
    : base(prise, maxspeed, year)
{
    this.MaxCapacity = MaxCapacity;
}
}

    class Garrage
{
public List<Plane> planes = new List<Plane>();
public List<Car> cars = new List<Car>();
public List<Ship> shipes = new List<Ship>();

public void AddPlane()
{
    int prise = 0, maxspeed = 0, year = 0, MaxCapacity = 0;
    Console.WriteLine("Уведіть ціну:");
    try
    {
        prise = Convert.ToInt32(Console.ReadLine());
    }
    catch { }
    Console.WriteLine("Уведіть максимальну швидкість");
    try
    {
        maxspeed = Convert.ToInt32(Console.ReadLine());
    }
    catch { }
    Console.WriteLine("Уведіть рік:");
    try
    {

```

```

        year = Convert.ToInt32(Console.ReadLine());
    }
    catch { }
    Console.WriteLine("Уведіть вантажопідйомність");
    try
    {
        MaxCapacity = Convert.ToInt32(Console.ReadLine());
    }
    catch { }
    planes.Add(new Plane(prise, maxspeed, year, MaxCapacity));
}

```

```

public void AddCar()
{
    int prise = 0, maxspeed = 0, year = 0, Power = 0;
    Console.WriteLine("Уведіть ціну :");
    try
    {
        prise = Convert.ToInt32(Console.ReadLine());
    }
    catch { }
    Console.WriteLine("Уведіть максимальну швидкість");
    try
    {
        maxspeed = Convert.ToInt32(Console.ReadLine());
    }
    catch { }
    Console.WriteLine("Уведіть рік:");
    try
    {
        year = Convert.ToInt32(Console.ReadLine());
    }
    catch { }
    Console.WriteLine("Уведіть потужність:");
    try
    {
        Power = Convert.ToInt32(Console.ReadLine());
    }
    catch { }
    cars.Add(new Car(prise, maxspeed, year, Power));
}

```

```

public void AddShip()
{

```

```

int prise = 0, maxspeed = 0, year = 0, MaxPassengers = 0;
Console.WriteLine("Уведіть ціну :");
try
{
    prise = Convert.ToInt32(Console.ReadLine());
}
catch { }
Console.WriteLine("Уведіть максимальну швидкість");
try
{
    maxspeed = Convert.ToInt32(Console.ReadLine());
}
catch { }
Console.WriteLine("Уведіть рік.");
try
{
    year = Convert.ToInt32(Console.ReadLine());
}
catch { }
Console.WriteLine("Уведіть кількість пасажирів.");
try
{
    MaxPassengers = Convert.ToInt32(Console.ReadLine());
}
catch { }
shipes.Add(new Ship(prise, maxspeed, year, MaxPassengers));
}

public void GetAllPlane()
{
    foreach(Plane plane in planes)
    {
        Console.WriteLine("Ціна: {0}, Максимальна швидкість: {1}, Рік: {2},
Максимальна вантажопідйомність: {3};"plane.Price, plane.Maxspeed, plane.Year,
plane.MaxCapacity);
    }
}

public void GetAllCar()
{
    foreach(Car car in cars)
    {
        Console.WriteLine("Ціна: {0}, Максимальна швидкість: {1}, Рік: {2},
Потужність: {3};"car.Price, car.Maxspeed, car.Year, car.Power);
    }
}

```



```

    }
}

public void GetAllShip()
{
    foreach(Ship ship in shipes)
    {
        Console.WriteLine("Ціна: {0}, Максимальна швидкість: {1}, Рік: {2},
Максимум пасажирів: {3}";ship.Price, ship.Maxspeed, ship.Year,
ship.MaxPassengers);
    }
}
class Program
{

    static void Main(string[] args)
    {
        Garrage garrage = new Garrage();

        while(true)
        {
            int i=0;
            Console.WriteLine(@"Виберіть потрібну Вам дію:
1. Додати до списку кораблик;
2. Додати до списку машину;
3. Додати до списку літак;
4. Вивести усі кораблики зі списку;
5. Вивести усі машини зі списку;
6. Вивести усі літаки зі списку;
7. Вихід");
            try
            {
                i=Convert.ToInt32( Console.ReadLine());
            }
            catch{}
            switch(i)
            {
                case 1: garrage.AddShip(); break;
                case 2: garrage.AddCar(); break;
                case 3: garrage.AddPlane(); break;
                case 4: Console.WriteLine("Кораблі:"); garrage.GetAllShip(); break;
                case 5: Console.WriteLine("Машини:"); garrage.GetAllCar(); break;
                case 6: Console.WriteLine("Літаки:"); garrage.GetAllPlane(); break;
                case 7: return;
            }
        }
    }
}

```

```
        default: Console.WriteLine(""); break;
    }
}
```

Примітки. 1. Exception — базовий клас для усіх виключень.

2. Protected — ключове слово, модифікатор доступу члена. Захищений член базового класу доступний в похідному класі, тільки якщо доступ відбувається через тип похідного класу.

3. Стандартне визначення властивості містить блоки get і set. У блоці get повертаємо значення поля, а у блоці set встановлюємо. Параметр value є передаваним значенням. Блоки set і get не обов'язково одночасно мають бути присутніми у властивості. Якщо властивість визначають тільки блок get, то така властивість доступна тільки для читання — можна отримати її значення, але не встановити. І, навпаки, якщо властивість має тільки блок set, тоді ця властивість доступна тільки для запису — можна тільки встановити значення, але не можна отримати.

4. Date — оскільки date повертає поточну дату без поточного часу, властивість Today підходить для використання у застосунках, що працюють тільки з датами.

5. This — ключове слово належить до поточного екземпляра класу, а також використовується як модифікатор першого параметра методу розширення.

6. Оператор try - catch складається з блока try, за яким йде одне або декілька пропозицій catch, в яких визначаються обробники для різних виключень.

7. Оператор foreach використовується для ітерації колекції з метою отримання необхідної інформації.

8. While (true) — це цикл, який працюватиме доти, доки застосунок не завершиться, або якщо в коді є команда break. Це звичайний спосіб зберегти потік.

9. Switch — це оператор вибору, який вибирає один *розділ перемикача* для виконання зі списку кандидатів на основі відповідності шаблону з *вираженням відповідності*.

10. Літерал default може використовуватися для виразів значення за умовчанням, якщо компілятор може вивести тип вираження. Літерал default створює те ж значення, що й еквівалент default(T), де T є виведеним типом. Це може зробити код компактнішим за рахунок скорочення надмірності у разі неодноразового оголошення типу.

## **Висновки**

У результаті виконання практичного заняття отримано навички написання, налагодження та компіляції програм розроблення класів з наслідуванням мовою C# у середовищі розроблення Microsoft Visual Studio.

## **Контрольні запитання**

1. Присвоювання та порівняння об'єктів з урахуванням особливостей хіпу та стека.
2. Дані класу: поля і константи.
3. Специфікатори полів і констант класу.
4. Методи класу: визначення, синтаксис, основні елементи описання.
5. Визов методу. Сигнатура методу.
6. Дії, що виконуються при визові методу.
7. Способи передачі параметрів методу.

## Практичне заняття № 10

### РОЗРОБЛЕННЯ СТРУКТУР МОВОЮ ПРОГРАМУВАННЯ C# У СЕРЕДОВИЩІ РОЗРОБЛЕННЯ MICROSOFT VISUAL STUDIO

**Мета заняття:** отримання навичок написання, налагодження і компіляції програм, що реалізують структури мовою C# у середовищі розроблення Microsoft Visual Studio.

#### Апаратне, програмне та методичне забезпечення

1. Засоби обчислювальної техніки (ПК).
2. Інсталяційний пакет програмного забезпечення Microsoft Visual Studio.
3. Пакет індивідуальних завдань.

#### Порядок виконання

1. Вивчення основних теоретичних положень [2, с. 188 - 219].
2. Розроблення мовою C# програм, що реалізують структури, згідно з варіантами індивідуальних завдань [2, с. 405 - 411].
3. Підготовка контрольного звіту за результатами виконання практичного заняття.

#### Зміст звіту

1. Титульний лист.
2. Індивідуальний варіант завдання.
3. Програмні коди реалізації МОВОЮ C# у середовищі розроблення Microsoft Visual Studio обчислювальних процесів згідно з індивідуальним варіантом завдання.
4. Висновки (про те, що виконано на практичному занятті та які навички отримано).

Звіт має бути виконаний у текстовому процесорі **Microsoft Word** (шрифтом **Times New Roman, 14 пт**, Вирівнювання: **по ширине**, Міжрядковий інтервал: **1,15**). На перевірку слід надати папку, назва якої (великими буквами) містить шифр групи, прізвище виконавця і номер практичного заняття (наприклад, 412ст-ІВАНЕНКО-ПЗ10). У цій папці слід розмістити:

- файл звіту (412ст-Іваненко-ПЗ10.docx або 412ст-Іваненко-ПЗ10.doc);
- папки та файли рішення MS Visual Studio (наприклад, папку "ConsoleApplication3" і файл "ConsoleApplication3.sln", де цифри у назві папки та назві файла мають співпадати з номером варіанта індивідуального завдання). Якщо створюється декілька рішень MS Visual Studio, то нумерувати відповідні папки та файли слід так: "ConsoleApplication3-1" і "ConsoleApplication3-1.sln",

"ConsoleApplication3-2" і файл "ConsoleApplication3-2.sln" і т. д., де перша цифра (тут 3) у назві відповідає номеру варіанта, а друга цифра — номеру рішення MS Visual Studio у межах варіанта індивідуального завдання.

Папку із звітом з практичного заняття слід здати в електронному вигляді старості академічної групи. Староста групи перевіряє правильність найменування папок звітів студентів, поміщує ці папки в папку під назвою 412ст-ПЗ10 і здає цю папку (в електронному вигляді) на перевірку викладачеві на початку наступного практичного заняття.

### Приклад виконання

**Завдання.** Описати структуру з іменем Student, що містить такі поля:

- 1) прізвище та ініціали;
- 2) номер групи;
- 3) успішність (масив з п'яти елементів).

Написати програму, що виконує такі дії:

1) уведення з клавіатури даних у масив, що складається з десяти структур типу Student (записи мають бути впорядковані за збільшенням середнього бала);

2) виведення на екран прізвищ і номерів груп для усіх студентів, що мають оцінки "добре" та "відмінно" (якщо таких студентів немає, вивести відповідне повідомлення).

### Програмний код

```
using System;
using System.Collections.Generic;
using System.Linq;
using System.Text;

namespace ConsoleApplication1
{
    class Program
    {
        static void Main(string[] args)
        {
            Student[] students = new Student[2];

            for(int i = 0; i < students.Length; i++)
            {
                Console.WriteLine("Уведіть ім'я та ініціали {0}-го студента: ", i + 1);
                string name = Console.ReadLine();
                Console.WriteLine("Уведіть номер групи {0}-го студента: ", i + 1);
                int number = int.Parse(Console.ReadLine());
            }
        }
    }
}
```

```

1);
    Console.WriteLine("Уведіть через кому 5 оцінок {0}-го студента: ", i +
string[] marks = Console.ReadLine().Split(',');

int[] progress = new int[5];
for(int a = 0; a < 5; a++)
    progress[a] = int.Parse(marks[a].ToString());

students[i] = new Student(name, nomber, progress);
}

var stud = from i in students orderby i.MediumBall() select i; //Сортування

Console.WriteLine("Упорядкований за середнім балом масив студентів (від
меншого): ");

foreach(Student student in stud)
    Console.WriteLine("" + student.ToString());

int count = 0;
foreach(Student student in students)
    if(student.MediumBall() > 4)
        count++;

if(count == 0)
    Console.WriteLine("Нема студентів із середнім балом більше 4!");
else
    Console.WriteLine("Список студентів із середнім балом більше 4: ");

foreach(Student student in students)
    if(student.MediumBall() > 4)
        Console.WriteLine("" + student.ToString());

Console.ReadKey();
}
}

struct Student
{
    public string Name;
    public int GroupNumber;
    int[] progress;

    public Student(string Name, int GroupNumber, int[] marks)

```

```

{
    this.Name = Name;
    this.GroupNumber = GroupNumber;
    progress = marks;
}

public double MediumBall()
{
    double MedBall = 0;

    foreach(int i in progress)
        MedBall += i;
    MedBall /= progress.Length;

    return MedBall;
}

public override string ToString()
{
    return string.Format("ПІБ: {0} Номер групи: {1}"Name, GroupNumber);
}
}
}

```

Примітки. 1. У класі, в якому не оголошено жодного конструктора, існує неявний конструктор за умовчанням, який викликається при створенні об'єкта за допомогою оператора new.

Оголошення конструктора має таку структуру:

```

public [ім'я_класу] ([аргументи])
{
    // тіло конструктора
}

```

2. Для розділення рядків на слова доцільно використовувати функцію Split.

3. Метод Parse() як параметр приймає рядок і повертає об'єкт поточного типу.

4. Для сортування набору даних за збільшенням використовується оператор orderby, який приймає критерій сортування. У цьому випадку критерієм виступає саме число.

5. Count — показує, скільки буде скопійовано букв у масиві рядків.

6. Foreach (int i in progress) — цикл, що перебирає елементи.

## **Висновки**

У результаті виконання практичного заняття отримано навички написання, налагодження і компіляції програм розроблення структур мовою C# у середовищі розроблення Microsoft Visual Studio.

## **Контрольні запитання**

1. Типи параметрів функції, що викликає, та функції, що викликається.
2. Параметри-значення методів класу.
3. Параметри-посилання методів класу.
4. Вихідні параметри методів.
5. Ключове слово `this`.
6. Конструктори: призначення, властивості, типи конструкторів.
7. Властивості: визначення, синтаксис, код доступу.



## Практичне заняття № 11

### РОЗРОБЛЕННЯ МОВОЮ C# У СЕРЕДОВИЩІ MICROSOFT VISUAL STUDIO ПРОГРАМ З ВИКОРИСТАННЯМ СТАНДАРТНИХ ПАРАМЕТРИЗОВАНИХ КОЛЕКЦІЙ ДЛЯ ЗБЕРІГАННЯ ЕКЗЕМПЛЯРІВ КЛАСІВ

**Мета заняття:** отримання навичок написання, налагодження і компіляції в середовищі розроблення Microsoft Visual Studio програм з використанням стандартних параметризованих колекцій для зберігання екземплярів класів.

#### Апаратне, програмне та методичне забезпечення

1. Засоби обчислювальної техніки (ПК).
2. Інсталяційний пакет програмного забезпечення Microsoft Visual Studio.
3. Пакет індивідуальних завдань.

#### Порядок виконання

1. Вивчення основних теоретичних положень [2, с. 188 - 219; 291 - 310].
2. Розроблення МОВОЮ C# програм з використанням стандартних параметризованих колекцій для зберігання екземплярів класів згідно з варіантами індивідуальних завдань [2, с. 411].
3. Підготовка контрольного звіту за результатами виконання практичного заняття.

#### Зміст звіту

1. Титульний лист.
2. Індивідуальний варіант завдання.
3. Програмні коди реалізації МОВОЮ C# у середовищі розроблення Microsoft Visual Studio обчислювальних процесів згідно з індивідуальним варіантом завдання.
4. Висновки (про те, що виконано на практичному занятті та які навички отримано).

Звіт має бути виконаний у текстовому процесорі **Microsoft Word** (шрифтом **Times New Roman**, **14** пт, Вирівнювання: **по ширине**, Міжрядковий інтервал: **1,15**). На перевірку слід надати папку, назва якої (великими буквами) містить шифр групи, прізвище виконавця і номер практичного заняття (наприклад, 412ст-ІВАНЕНКО-ПЗ11). В цій папці слід розмістити:

- файл звіту (412ст-Іваненко-ПЗ11.docx або 412ст-Іваненко-ПЗ11.doc);

- папки та файли рішення MS Visual Studio (наприклад, папку "ConsoleApplication3" і файл "ConsoleApplication3.sln", де цифри у назві папки та назві файла мають співпадати з номером варіанта індивідуального завдання). Якщо створюється декілька рішень MS Visual Studio, то нумерувати відповідні папки та файли слід так: "ConsoleApplication3-1" і "ConsoleApplication3-1.sln", "ConsoleApplication3-2" і файл "ConsoleApplication3-2.sln" і т. д., де перша цифра (тут 3) у назві відповідає номеру варіанта, а друга цифра — номеру рішення MS Visual Studio у межах варіанта індивідуального завдання.

Папку зі звітом з практичного заняття слід здати в електронному вигляді старості академічної групи. Староста групи перевіряє правильність найменування папок звітів студентів, поміщує ці папки в папку під назвою 412ст-ПЗ11 і здає цю папку (в електронному вигляді) на перевірку викладачеві на початку наступного практичного заняття.

### Приклад виконання

**Завдання:** виконати завдання практичного заняття № 9, використовуючи для зберігання екземплярів розроблених класів стандартні параметризовані колекції. В усіх класах реалізувати інтерфейс IComparable та перевантажити операції відношення для реалізації значущої семантики порівняння об'єктів у будь-якому полі на власний розсуд.

### Програмний код

```
using System;
using System.Collections.Generic;
using System.Linq;
using System.Text;

namespace linecollection
{
    class Line: IComparable<Line> //1
    {

        private string line; //2
        private int word;

        public string Lines { set { this.line = value; } get { return line; } } //3
        public string Word { set { this.word = Convert.ToInt32(value); } get { return
word.ToString(); } }

        public Line() //4
```

```

{
    this.line = "Практичне заняття № 11";
    this.word = line.Length * 2;
}

    public Line    //5
{
    this.line = ch.ToString();
}

public Line    //6
{
    this.line = line;
    this.word = line.Length * 2;
}

public int GetLength()    //7
{
    return line.Length;
}

public void Clear()    //8
{
    line = "";
}

public int CompareTo    //9
{
    return this.line.CompareTo(other.Lines);
}

public static bool operator //10
{
    return FirstLine.CompareTo == 0;
}

public static bool operator !=(Line FirstLine, Line SecondLine)
{
    return FirstLine.CompareTo(SecondLine) != 0;
}

public static bool operator <(Line FirstLine, Line SecondLine)
{
    return FirstLine.CompareTo(SecondLine) != 1;
}

```

```

public static bool operator >(Line FirstLine, Line SecondLine)
{
    return FirstLine.CompareTo(SecondLine) != - 1;
}

public static bool operator <=(Line FirstLine, Line SecondLine)
{
    return FirstLine.CompareTo(SecondLine) <= 0;
}

public static bool operator >=(Line FirstLine, Line SecondLine)
{
    return FirstLine.CompareTo(SecondLine) >= 0;
}
}
}

```

### Коментарі до програмного коду

1. Оголошення класу і використання для зберігання його екземплярів інтерфейсу IComparable.
2. Оголошення полів для зберігання символів рядка і його довжини у байтах.
3. Getter&Setter використовується для доступу до прихованих полів класів (у цьому випадку — до приватних полів).
4. Конструктор без параметрів. Значення рядка встановлене розробником.
5. Конструктор, що приймає як параметр символ і встановлює значення рядка відповідно до цього символу.
6. Конструктор, що приймає рядок як параметр і встановлює його довжину.
7. Метод, що повертає довжину рядка.
8. Метод, що очищає рядок.
9. Реалізація інтерфейсу IComparable передбачає використання методу CompareTo (порівняння елементів).

### Реалізація інтерфейсу IComparable

```

using System;
using System.Collections.Generic;
using System.Linq;
using System.Text
namespace linecollection
{

```

```
class DecimalLine: Line, IComparable<DecimalLine> //11
```

```
{
```

Оголошення класу-спадкоємця з подальшим використанням згаданого вище інтерфейсу.

```
public DecimalLine //12
```

```
{
```

```
try
```

```
{
```

```
line = Convert.ToInt32(line).ToString();
```

```
}
```

```
catch(Exception )
```

```
{
```

```
line = "0";
```

Оброблення виключень (якщо ввести текст на цьому етапі, рядок набуде нульового значення)

```
}
```

```
Word(line.Length * 2).ToString();
```

```
Lines = line;
```

```
}
```

```
public string Sub //13
```

```
{
```

```
return Convert.ToInt32(this.Lines) -
```

```
Convert.ToInt32(decimalLine.Lines)).ToString(); ;
```

Метод, що обчислює різницю рядків

```
}
```

```
public bool Less(DecimalLine decimalLine)
```

```
{
```

```
return Convert.ToInt32(this.Lines) > Convert.ToInt32(decimalLine.Lines); //14
```

```
}
```

```
public bool More(DecimalLine decimalLine)
```

```
{
```

```
return Convert.ToInt32(this.Lines) < Convert.ToInt32(decimalLine.Lines); //15
```

```
}
```

```
public int CompareTo //16
```

```
{
```

```
if(Convert.ToInt32(this.Lines) > Convert.ToInt32(other.Lines))
```

```
return 1;
```

```
if(Convert.ToInt32(this.Lines) < Convert.ToInt32(other.Lines))
```

```
return - 1;
```

```
else
```

```
return 0;
```

```

    }

    public static bool operator <(DecimalLine FirstLine, DecimalLine SecondLine)
//17
    {
        return Convert.ToInt32(FirstLine.Lines) < Convert.ToInt32(SecondLine.Lines);
    }

    public static bool operator >(DecimalLine FirstLine, DecimalLine SecondLine)
    {
        return Convert.ToInt32(FirstLine.Lines) > Convert.ToInt32(SecondLine.Lines);
    }

    public static bool operator <=(DecimalLine FirstLine, DecimalLine SecondLine)
    {
        return Convert.ToInt32(FirstLine.Lines)
<=Convert.ToInt32(SecondLine.Lines);
    }

    public static bool operator >=(DecimalLine FirstLine, DecimalLine SecondLine)
    {
        return Convert.ToInt32(FirstLine.Lines) >=
Convert.ToInt32(SecondLine.Lines);
    }
}
}

```

## Реалізація методу CompareTo при реалізації інтерфейсу IComparable

Метод порівнює два рядки і повертає значення 1, 0 або -1 залежно від результату порівняння:

```

using System;
using System.Collections.Generic;
using System.Linq;
using System.Text;

namespace linecollection
{
    class Program
    {
        static void Main(string[] args)
        {
            Console.Write("(1) - Compare operators(2) - Sortet(0) - Exit: "); //18
            switch(Console.ReadLine())

```

```

{
    case "1" :
        Console.WriteLine();
        Compare();
        Console.WriteLine("");
        Main(null);
        break;
    case "2" :
        Console.WriteLine();
        Sorter();
        Console.WriteLine("");
        Main(null);
        break;
    case "0" :
        break;
    default:
        Console.WriteLine("Pls. Chose 1,2 or 3.");
        Main(null);
        break;
}
}

```

```

public static void Compare() //19

```

```

{
    Console.Write("Enter first operand: ");
    string s1 = Console.ReadLine();
    Console.Write("Enter second operand: ");
    string s2 = Console.ReadLine();

    DecimalLine l1 = new DecimalLine(s1);
    DecimalLine l2 = new DecimalLine(s2);

    Console.WriteLine("" + l1.Lines + " == " + l2.Lines + " : " + (l1 == l2) +
        "\n      First operand type: " + l1 +
        "\n      Second operand type: " + l2 + "");
    Console.WriteLine(l1.Lines + " != " + l2.Lines + " : " + (l1 != l2) +
        "\n      First operand type: " + l1 +
        "\n      Second operand type: " + l2 + "");
    Console.WriteLine(l1.Lines + " > " + l2.Lines + " : " + (l1 > l2) +
        "\n      First operand type: " + l1 +
        "\n      Second operand type: " + l2 + "");
    Console.WriteLine(l1.Lines + " < " + l2.Lines + " : " + (l1 < l2) +
        "\n      First operand type: " + l1 +
        "\n      Second operand type: " + l2 + "");
}

```

```

Console.WriteLine(l1.Lines + ">= " + l2.Lines + " : " + (l1 >= l2) +
    "\n          First operand type: " + l1 +
    "\n          Second operand type: " + l2 + "");
Console.WriteLine(l1.Lines + "<= " + l2.Lines + " : " + (l1 <= l2) +
    "\n          First operand type: " + l1 +
    "\n          Second operand type: " + l2 + "");

```

```

Line l3;
if(s11 != "")
{
    l3 = new Line(s11);
}
else
{
    l3 = new Line();
}

```

```

Line l4;
if(s12 != "")
{
    l4 = new Line(s12);
}
else
{
    l4 = new Line();
}

```

```

Console.WriteLine("_____
_____"); //20

```

```

Console.WriteLine(l3.Lines + " == " + l4.Lines + " : " + (l3 == l4) +
    "\n          First operand type: " + l3 +
    "\n          Second operand type: " + l4 + "");
Console.WriteLine(l3.Lines + " != " + l4.Lines + " : " + (l3 != l4) +
    "\n          First operand type: " + l3 +
    "\n          Second operand type: " + l4 + "");
Console.WriteLine(l3.Lines + " > " + l4.Lines + " : " + (l3 > l4) +
    "\n          First operand type: " + l3 +
    "\n          Second operand type: " + l4 + "");
Console.WriteLine(l3.Lines + " < " + l4.Lines + " : " + (l3 < l4) +
    "\n          First operand type: " + l3 +
    "\n          Second operand type: " + l4 + "");
Console.WriteLine(l3.Lines + " >= " + l4.Lines + " : " + (l3 >= l4) +

```



```

        "\n          First operand type: " + 13 +
        "\n          Second operand type: " + 14 + """);
    Console.WriteLine(13.Lines + "<= " + 14.Lines + " : " + (13 <= 14) +
        "\n          First operand type: " + 13 +
        "\n          Second operand type: " + 14 + """);
}

```

```

public static void Sorter()
{

```

```

    Collection col = new Collection(); //21

```

```

    col.Add(new DecimalLine("30"));
    col.Add(new DecimalLine("+30"));
    col.Add(new Line("SomeString"));
    col.Add(new DecimalLine("40"));
    col.Add(new DecimalLine("+140"));
    col.Add(new DecimalLine("qwerty"));
    col.Add(new Line("-85"));
    col.Add(new Line("0"));
    col.Add(new DecimalLine("1"));

```

```

    Console.WriteLine("List of Line : "); //22

```

```

    for(int i = 0; i < col.lines.Count; i++)
    {

```

```

        Console.Write("  ");

```

```

        Console.WriteLine("Value: " + col.lines[i].Lines + " -----> L/S :(" +
col.lines[i].GetLength() + "/" + col.lines[i].Word + ")");

```

```

    }

```

```

    col.lines.Sort();

```

```

    Console.WriteLine(""); //23

```

```

    Console.WriteLine("Sorted List of Line : ");

```

```

    for(int i = 0; i < col.lines.Count; i++)
    {

```

```

        Console.Write("  ");

```

```

        Console.WriteLine("Value: " + col.lines[i].Lines + " -----> L/S :(" +
col.lines[i].GetLength() + "/" + col.lines[i].Word + ")");

```

```

    }

```

```

    //-----

```

```

Console.WriteLine("_____");

Console.WriteLine("List of DecimalLine : "); //24
for(int i = 0; i < col.dlines.Count; i++)
{
    Console.Write(" ");
    Console.WriteLine("Value: " + col.dlines[i].Lines + " -----> L/S :(" +
col.dlines[i].GetLength() + "/" + col.dlines[i].Word + ")");
}

col.dlines.Sort();

Console.WriteLine("");

Console.WriteLine("Sorted List of DecimalLine : "); //25
for(int i = 0; i < col.dlines.Count; i++)
{
    Console.Write(" ");
    Console.WriteLine("Value: " + col.dlines[i].Lines + " -----> L/S :(" +
col.dlines[i].GetLength() + "/" + col.dlines[i].Word + ")");
}
}
}
}
}

```

## Висновки

У результаті виконання практичного заняття отримано навички написання, налагодження та компіляції в середовищі розроблення Microsoft Visual Studio програм з використанням стандартних параметризованих колекцій для зберігання екземплярів класів.

## Контрольні запитання

1. Суттєвість перевантаження методів класів.
2. Рекурсивні методи класів.
3. Методи зі змінною кількістю аргументів.
4. Призначення та особливості методу Main.
5. Індексатори як різновиди властивостей.
6. Операції класу. Правила описів операцій класу.

## Практичне заняття № 12

### СТВОРЕННЯ WINDOWS-ЗАСТОСУНКІВ МОВОЮ ПРОГРАМУВАННЯ C# У СЕРЕДОВИЩІ РОЗРОБЛЕННЯ MICROSOFT VISUAL STUDIO

**Мета заняття:** отримання навичок написання, налагодження і компіляції програм, що реалізують Windows-застосунки мовою C# у середовищі розроблення Microsoft Visual Studio.

#### Апаратне, програмне та методичне забезпечення

1. Засоби обчислювальної техніки (ПК).
2. Інсталяційний пакет програмного забезпечення Microsoft Visual Studio.
3. Пакет індивідуальних завдань.

#### Порядок виконання

1. Вивчення основних теоретичних положень з програмування під Windows [2, с. 311 - 343].
2. Вивчення основних теоретичних положень з розроблення Windows-застосунків [3, с. 13 - 138].
3. Ознайомлення з матеріалами короткого довідника з Windows Forms [3, с. 329 - 344] для подальшого використання при розробленні Windows- застосунків.
4. Розроблення мовою C# Windows-застосунків згідно з варіантами індивідуальних завдань [2, с. 412 - 419].
5. Підготовка контрольного звіту за результатами виконання практичного заняття.

#### Зміст звіту

1. Титульний лист.
2. Індивідуальний варіант завдання.
3. Програмні коди реалізації мовою C# у середовищі розроблення Microsoft Visual Studio обчислювальних процесів згідно з індивідуальним варіантом завдання.
4. Висновки (про те, що виконано на практичному занятті та які навички отримано).

Звіт має бути виконаний у текстовому процесорі **Microsoft Word** (шрифтом **Times New Roman, 14 пт**, Вирівнювання: **по ширине**, Міжрядковий інтервал: **1,15**). На перевірку слід надати папку, назва якої (великими буквами) містить шифр групи, прізвище виконавця і номер практичного заняття (наприклад, 412ст-ІВАНЕНКО-ПЗ12). У цій папці слід розмістити:

- файл звіту (412ст-Іваненко-ПЗ12.docx або 412ст-Іваненко-ПЗ12.doc);

- папки та файли рішення MS Visual Studio (наприклад, папку "WindowsFormsApplication3" і файл "WindowsFormsApplication3.sln", де цифри у назві папки та назві файла мають збігатися з номером варіанта індивідуального завдання). Якщо створюється декілька рішень MS Visual Studio, то нумерувати відповідні папки та файли слід так: "WindowsFormsApplication3-1" і "WindowsFormsApplication3-1.sln", "WindowsFormsApplication3-2" і файл "WindowsFormsApplication3-2.sln" і т. д., де перша цифра (тут 3) у назві відповідає номеру варіанта, а друга цифра — номеру рішення MS Visual Studio у межах варіанта індивідуального завдання.

При роботі з Microsoft Visual Studio 2010 папку та файл рішення (тобто, наприклад, папку "WindowsFormsApplication1" і файл "WindowsFormsApplication1.sln") треба скопіювати з папки "WindowsFormsApplication1", яка за замовчуванням знаходиться так: C:\ Рабочий стол \ Комп'ютер \ Документи \ Visual Studio2010 \ Projects \ WindowsFormsApplication1.

#### **Технічно це виконується так.**

1. На Робочому столі двічі клікнути лівою клавішею миші по іконці "Комп'ютер" (рис. 17).



Рис. 17. Відкриття вікна "Комп'ютер"

2. У вікні "Комп'ютер" обрати меню "Документи" (рис. 18).

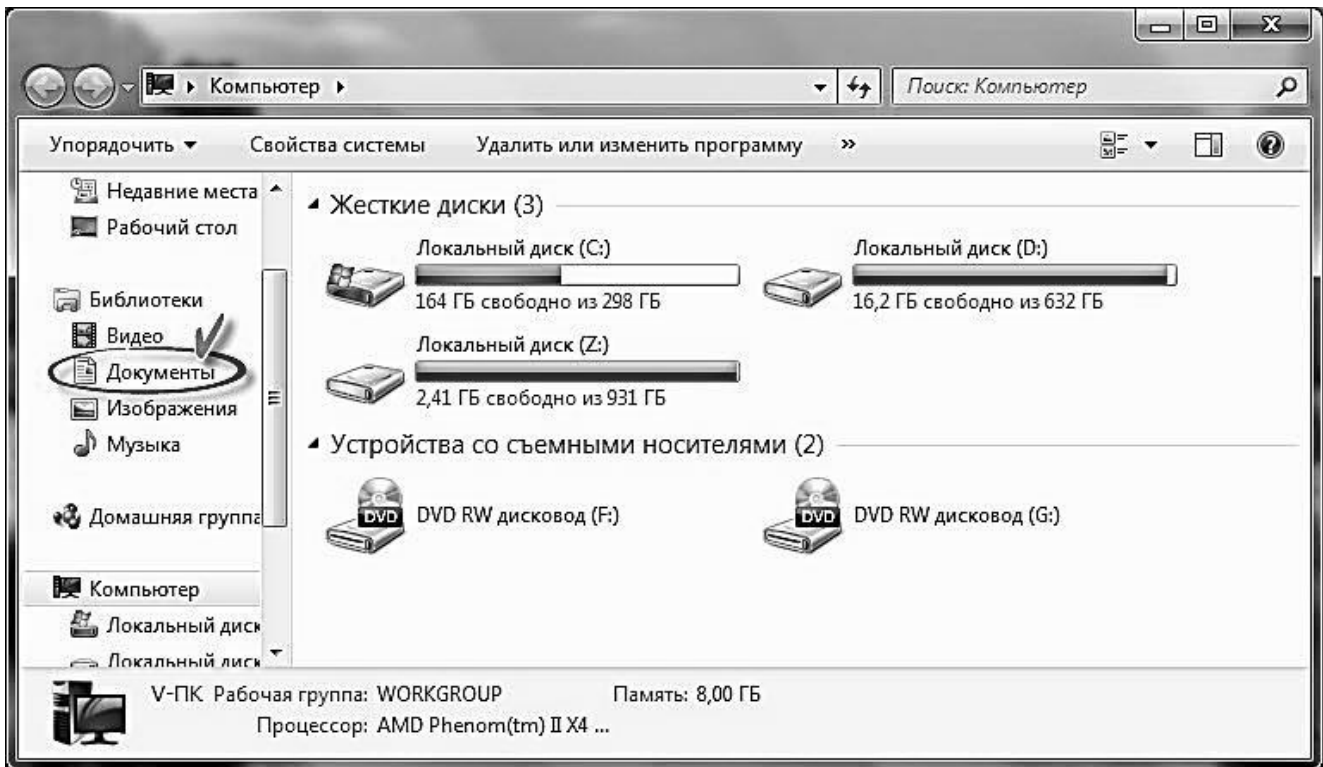


Рис. 18. Вибір меню "Документи"

3. У вікні "Документи" відкрити папку "Visual Studio 2010" (рис. 19).

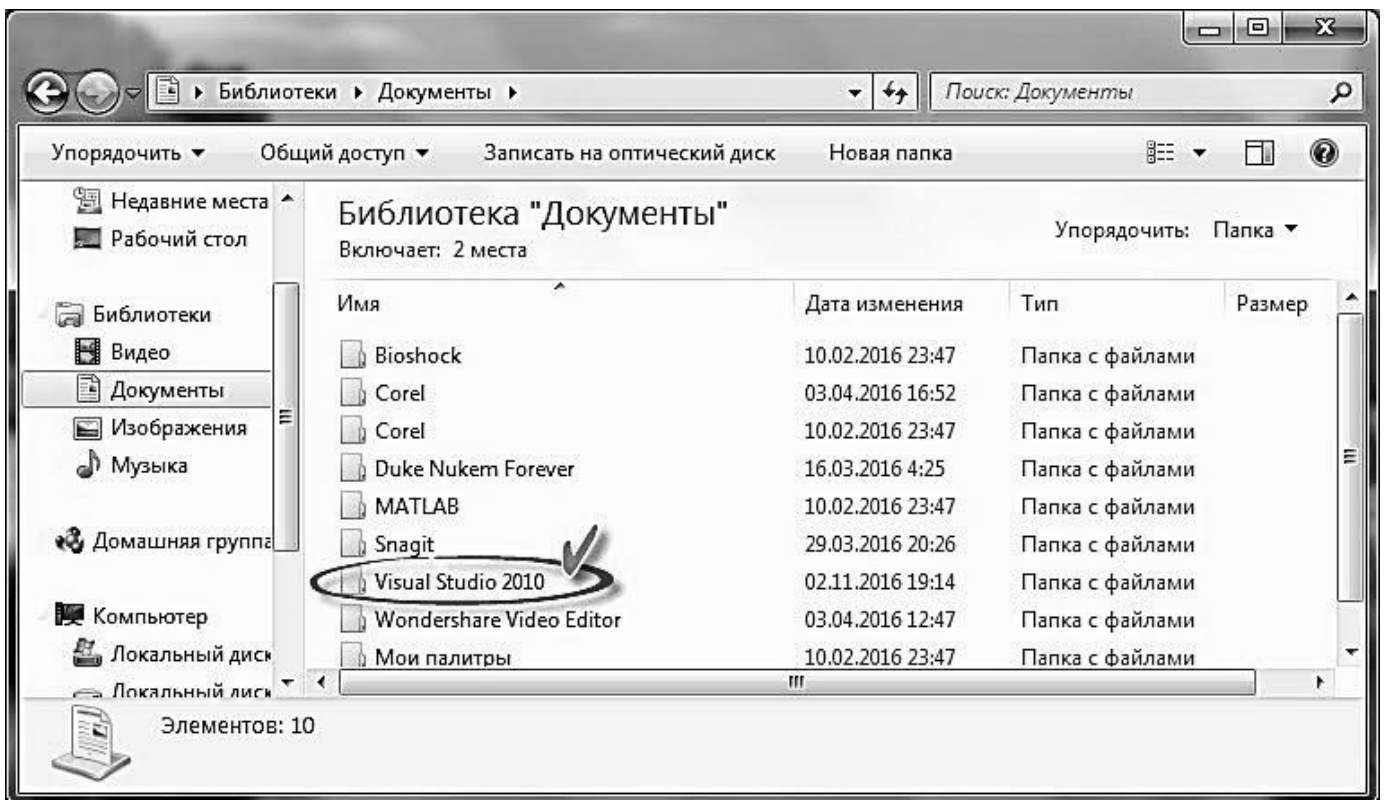


Рис. 19. Відкривання папки "Visual Studio 2010"

4. У папці "Visual Studio 2010" відкрити папку "Projects" (рис. 20).

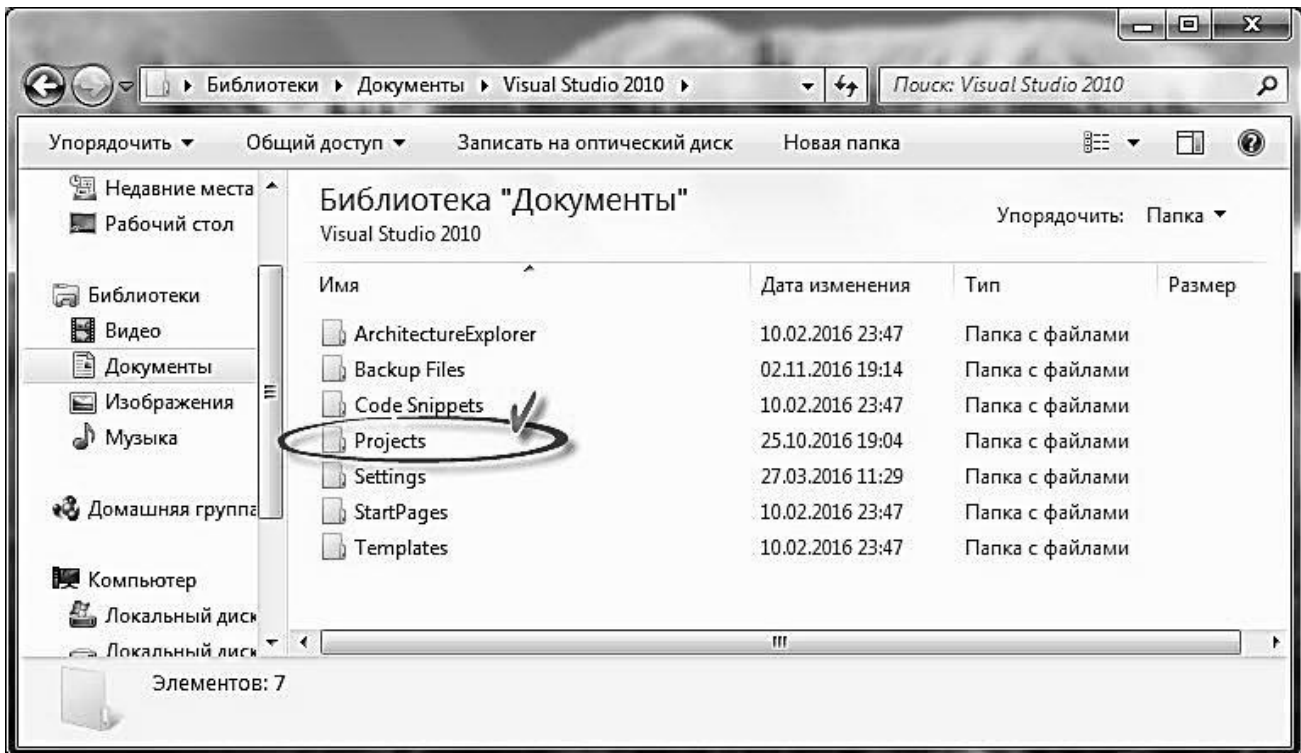


Рис. 20. Відкривання папки "Projects"

5. У папці "Projects" знайти та скопіювати папку "WindowsFormsApplication1" (рис. 21) у папку 412ст-ІВАНЕНКО-П312, яку слід здати старості групи. Зрозуміло, що у цій же папці 412ст-ІВАНЕНКО-П312 треба розмістити й файл звіту (412ст-Іваненко-П312.docx).

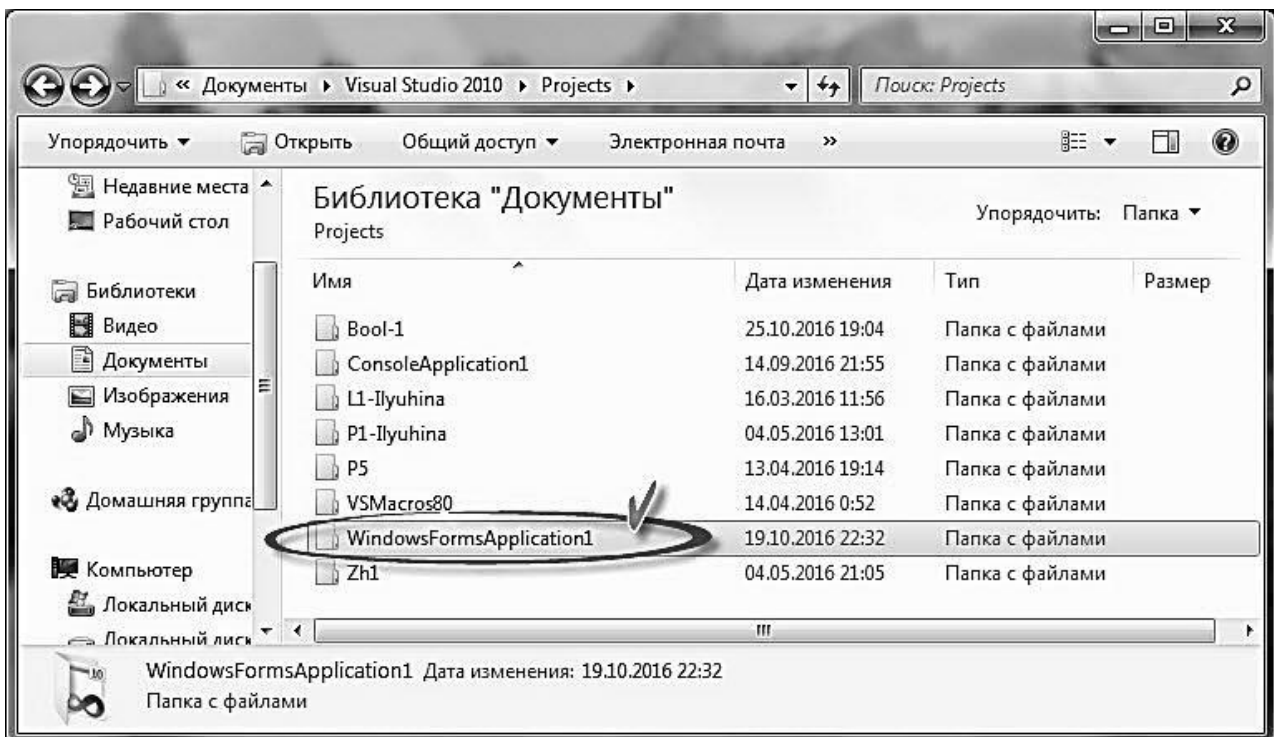


Рис. 21. Копіювання папки "ConsoleApplication1"

Таким чином, у папці зі звітом про виконану роботу 412ст-ІВАНЕНКО-ПЗ12 розмістяться усі файли, які необхідні для перевірки викладачем (рис. 22).

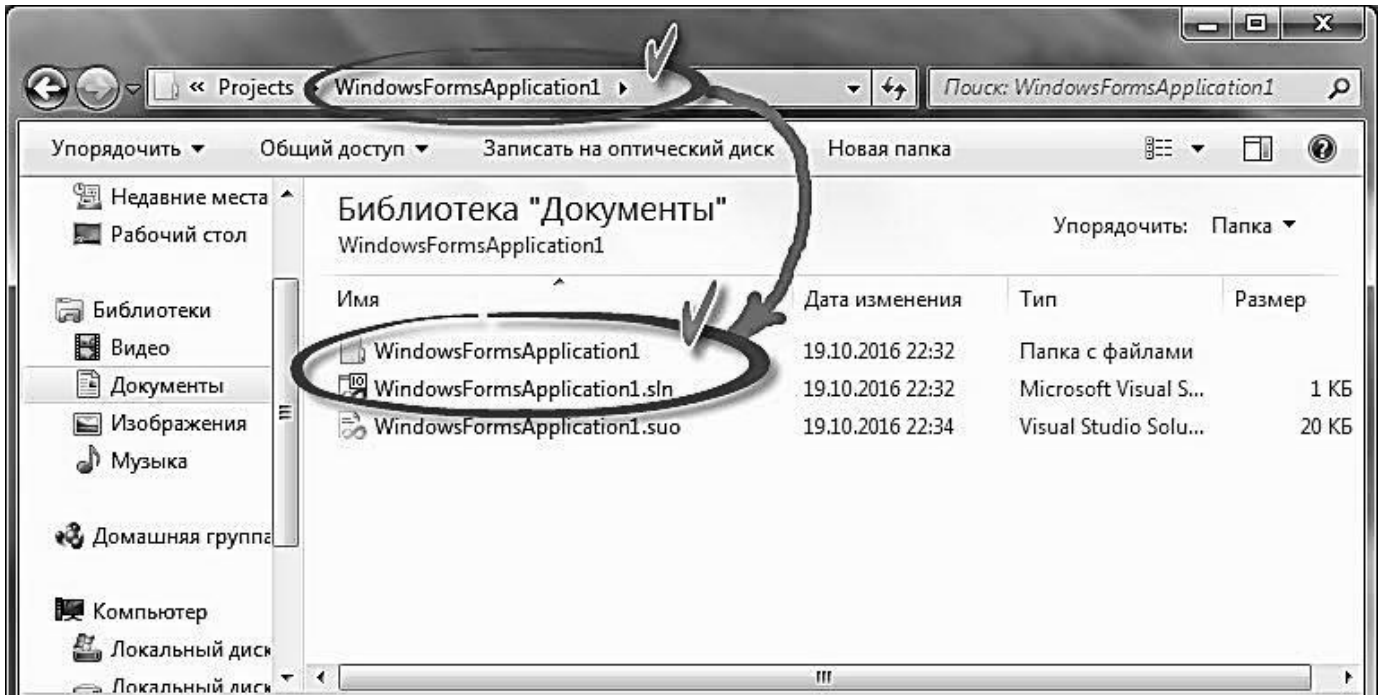


Рис. 22. Вміст папки "ConsoleApplication1"

Папку зі звітом з практичного заняття слід здати в електронному вигляді старості академічної групи. Староста групи перевіряє правильність найменування папок звітів студентів, поміщує ці папки в папку під назвою 412ст-ПЗ12 і здає цю папку (в електронному вигляді) на перевірку викладачеві на початку наступного практичного заняття.

### Приклад виконання

**Завдання.** Створити меню з командами Input, Work, Exit. При виборі команди Exit додаток завершує роботу. При виборі команди Input відкривається діалогове вікно, що містить:

- 1) три поля введення типу TextBox з мітками Radius, Height, Density;
- 2) групу з двох прапорців (Volume, Mass) типу CheckBox;
- 3) кнопку типу Button.

Забезпечити можливість:

- 1) уведення радіуса, висоти і щільності конуса;
- 2) вибору режиму за допомогою прапорців, тобто підрахунок об'єму і/або маси конуса.

При виборі команди Work відкривається вікно повідомлень з результатами.

## Програмний код

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

namespace WindowsFormsApplication1
{
    public partial class Form1: Form//Конструкція, яка дозволяє визначити один клас
    в декількох файлах
    {
        private double _radius;//Модифікатор доступу. Закритий доступ є рівнем
    доступу з мінімальними правами.
        private double _height;
        private double _density;
        private double _volume;
        private double _mass;
        private bool _calculateVolume;
        private bool _calculateMass;

        public Form1()
        {
            InitializeComponent();//Ініціалізація (створення) компонентів

            _radius = 0;
            _height = 0;
            _density = 0;
            _volume = 0;
            _mass = 0;
            _calculateVolume = false;
            _calculateMass = false;
        }

        private void exitToolStripMenuItem_Click// Sender- об'єкт, який згенерував
    подію exitToolStripMenuItem
        {
            this.Close();// При натисненні на Exit програма закривається
        }

        private void inputToolStripMenuItem_Click_1(object sender, EventArgs e)
```



```

{
    Dialog inputDialog = new Dialog();// При натисненні на input відкривається
діалогове вікно

    inputDialog.ShowDialog();//При натисненні на input відкривається
діалогове вікно, де користувач заповнює поля

    _radius = inputDialog.Radius;
    _height = inputDialog.Height;
    _density = inputDialog.Density;
    _calculateVolume = inputDialog.Volume;
    _calculateMass = inputDialog.Mass;
}

private void workToolStripMenuItem_Click(object sender, EventArgs e)
{
    if(_radius == 0 || _height == 0 || _density == 0(_calculateVolume ||
_calculateMass))
    {
        MessageBox.Show("Please enter input data in the menu item Input", "No
data entered", MessageBoxButtons.OK, MessageBoxIcon.Warning);
        return;// Якщо користувач не заповнив усі поля і не вибрав команду,
з'явиться діалогове вікно із повідомленням про помилку
    }

    CalculateVolume();// Розрахунок об'єму

    CalculateMass();// Розрахунок маси

    DialogResult dialogResult = new DialogResult();//Нове діалогове вікно, з
порахованими результатами

    dialogResult.Volume = _calculateVolume ? _volume: 0;
    dialogResult.Mass = _calculateMass ? _mass: 0;

    dialogResult.ShowDialog();
}

private void CalculateMass()
{
    _mass = _volume * _density;//Формула для обчислення маси
}

private void CalculateVolume()

```

```
{  
    _volume / 3; //Формула для обчислення об'єму  
}
```

`private void Form1_Load`//Цю подію можна використати для виконання завдань (виділення ресурсів, використовуваних формою)

```
{  
  
}
```

`private void toolStripMenuItem1_Click`//Вибір команди "Показати отримані результати"

```
{  
  
}  
}
```

Вигляд вікон інтерфейсу розробленого Windows-застосунку показано на рис. 23, 24.

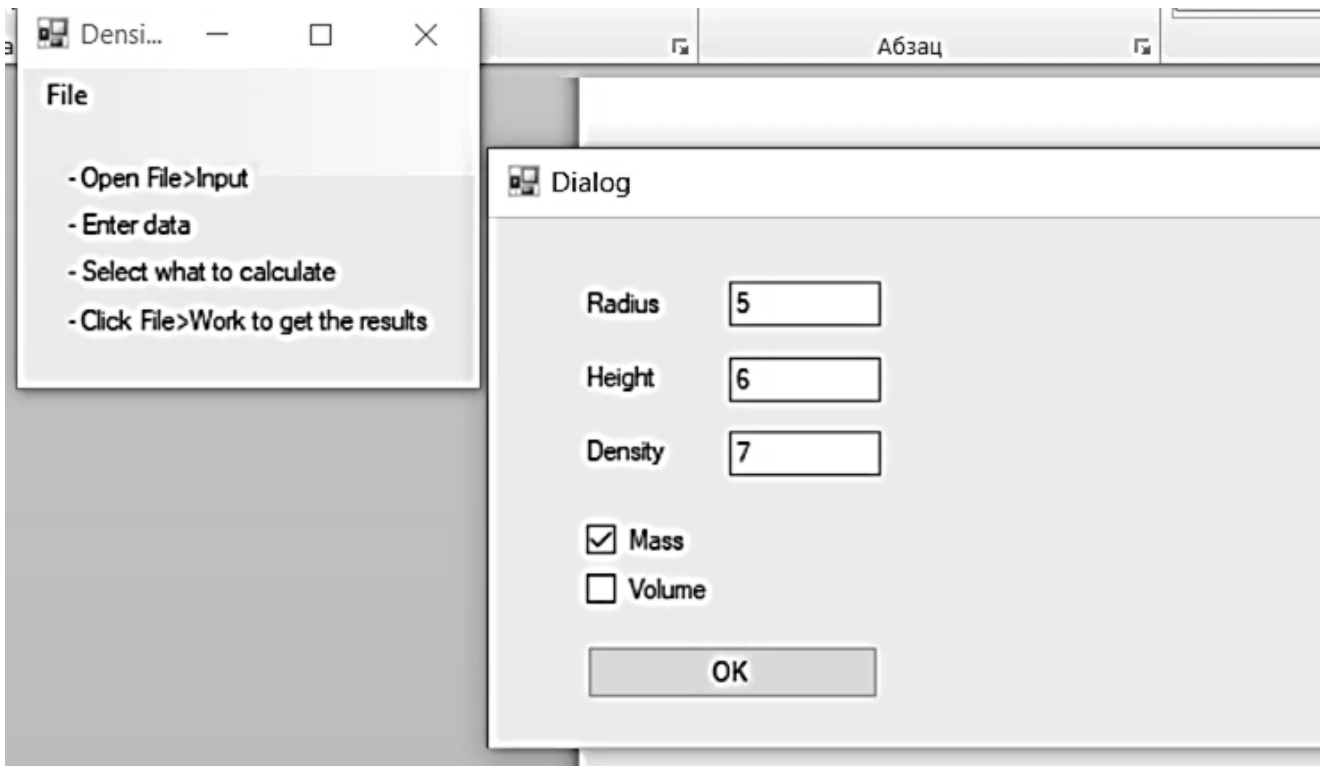


Рис. 23. Вікно "Dialog" розробленого Windows-застосунку

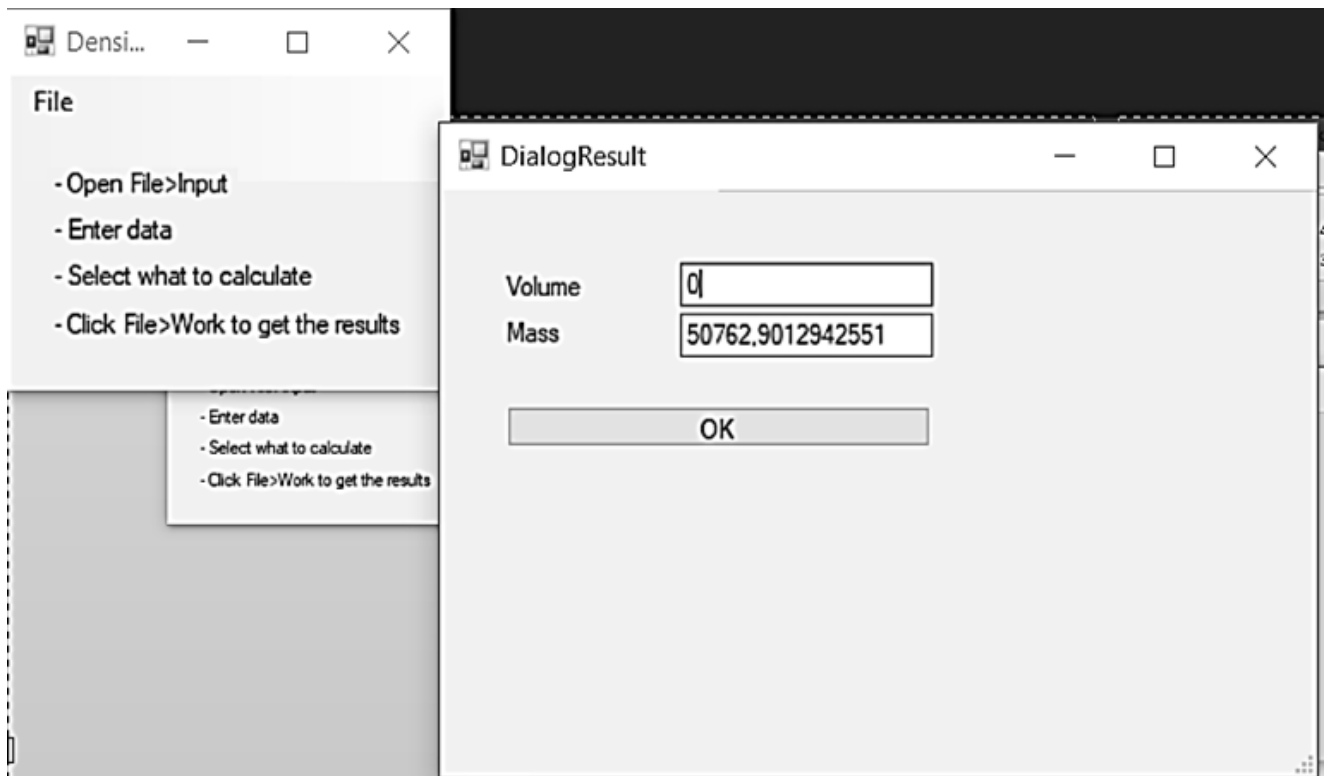


Рис. 24. Вікно "DialogResult" розробленого Windows-застосунку

## Висновки

У результаті виконання практичного заняття отримано навички написання, налагодження і компіляції програм, що реалізують Windows-застосунки МОВОЮ С# у середовищі розроблення Microsoft Visual Studio.

## Контрольні запитання

1. Суттєвість оброблення виняткових ситуацій, що виникають у процесі виконання програми.
2. Оператор try: призначення та формат.
3. Оператор throw: призначення та формат.
4. Клас Exception: властивості для отримання інформації про виняткові ситуації.
5. Оператори checked і unchecked: призначення та формат.

## РЕКОМЕНДАЦІЇ З ПРОГРАМУВАННЯ

Для того щоб вільно писати і читати програми, необхідно розуміти, з яких елементів мови програмування вони складаються. Це допомагає при пошуці помилок, зверненні до довідкової системи, а також під час вивчення нових версій мови програмування. Більш того, вивчення будь-якої нової мови програмування рекомендується починати саме з вивчення підтримуваних нею лексем, тобто мінімальних одиниць мови програмування, що мають самостійне значення.

Поняття типу даних лежить в основі більшості мовних засобів. При вивченні будь-якого типу необхідно розглядати дві речі: внутрішнє подання типу (а відповідно й множину можливих значень величин цього типу), а також те, що можна робити з цими величинами. Множина типів даних, що реалізована у мові програмування, є однією з найважливіших характеристик мови. Вибір найбільш підходящого типу для подання даних — одна з необхідних умов для створення ефективних програм.

Нові мови та засоби програмування з'являються безперервно, тому програміст вимушений вчитися все життя. Дуже важливо одразу навчитися вчитись швидко та ефективно. Для цього треба підходити до засвоєння кожної мови програмування системно: виділити складові частини, зрозуміти їх організацію та взаємозв'язок, подібності та відмінності від засобів, вивчених раніше, тобто за короткий час розкласти все в мозку "по полицках" так, щоб нові знання гармонійно доповнили наявні. Тільки в цьому випадку ними буде легко та приємно користуватися.

Програміст-професіонал має вміти:

- грамотно поставити задачу;
- обрати відповідні мовні засоби;
- обрати найбільш підходящі для подання даних структури;
- розробити ефективний алгоритм;
- написати та документувати надійну програму, що легко модифікується;
- забезпечити вичерпне тестування розробленої програми.

Крім того, все це необхідно виконувати у заздалегідь визначені терміни. Все зазначене стосується програмування як на C#, так і на інших мовах.

Пристаючи до написання програми, треба чітко визначити, що є її вихідними даними і що треба отримати в результаті. Тип змінних потрібно обирати з урахуванням діапазону та необхідної точності подання даних.

Змінним треба давати такі імена, що відображують їх значення. Правильно обрані імена змінних можуть зробити програму деякою мірою самодокументованою. Невдалі імена, навпаки, є джерелом проблем.

В іменах слід уникати скорочень. Вони роблять програму менш зрозумілою, до того ж часто легко забути, як саме було скорочене те чи

інше слово. Загальна тенденція така: чим більше область дії змінної, тим довше у неї ім'я. Перед таким ім'ям можна поставити префікс типу (одну чи декілька букв, за якими можна визначити тип змінної). Навпаки, для змінних, все "життя" яких триває протягом декількох рядків програмного коду, краще обійтися однобуквеними іменами, наприклад `b` або `k`.

Імена змінних логічного типу, використовувані як прапори, мають бути такими, щоб за ними можна було судити про те, що містять значення `true` та `false`. Наприклад, ознаку "порожньо" краще описати не як `bool flag`, а як `bool empty`.

До речі, в *C#* прийнято називати класи, методи та змінні відповідно до нотації *Паскаля*, а локальні змінні — відповідно до нотації *Camel*.

Стосовно мов програмування під нотацією прийнято розуміти угоду про правила створення імен.

У нотації *Паскаля* кожне слово, що становить ідентифікатор, починається з великої літери, наприклад, `MaxLength`, `MyFuzzyShooshpanchik`.

*Угорська нотація* (її запропонував угорець за національністю, співробітник компанії Microsoft) відрізняється від попередньої наявністю префікса, що відповідає типу величини, наприклад, `iMaxLength`, `lpfnMyFuzzyShooshpanchik`.

Згідно з нотацією *Camel* з великої букви починається кожне слово, що становить ідентифікатор, крім першого, наприклад, `maxLength`, `myFuzzyShooshpanchik`. При цьому абрис імені може нагадувати верблюда, звідки й виникла назва цієї нотації.

Існує ще одна традиція — розділяти слова, що становлять ім'я, знаками підкреслювання: `max_length`, `my_fuzzy_shooshpanchik`, при цьому усі складові частини починаються з малої літери.

Змінні бажано ініціювати при їх об'явленні, а об'являти слід якомога ближче до місця їх безпосереднього використання. З іншого боку, зручно усі об'явлення локальних змінних методу розміщувати на початку блока так, щоб їх було неважко знайти. При невеликих розмірах методів обидві ці рекомендації доволі легко поєднати.

Бажано уникати в програмі використання чисел у явному вигляді. Константи мають бути з осмисленими іменами, заданими за допомогою ключового слова `const`. Символічне ім'я робить програму більш зрозумілою. Крім того, у разі необхідності змінити значення константи потрібно буде змінити програму тільки в одному місці. Звичайно, ця порада не відноситься до констант 0 і 1.

Уведення з клавіатури слід випереджати запрошенням, а значення, що виводяться, — поясненнями. Для контролю одразу ж після уведення слід виводити вихідні дані на дисплей (принаймні, в процесі налагодження програми).

Перед запуском програми слід підготувати тестові приклади, що містять вихідні дані та очікувані результати. Окремо слід перевірити реакцію програми на неправильні вихідні дані.

Під час запису виразів слід звертати увагу на пріоритет операцій. Якщо в одному вираженні є сусідами кілька операцій однакового пріоритету, операції присвоювання та умовна операція виконуються справа наліво, решта — зліва направо. Для змінювання порядку виконання операцій слід використовувати круглі дужки.

Текст програми треба ретельно форматувати так, щоб його було зручно читати. Пробіли слід ставити після знаків пунктуації, знаки операцій — відокремлювати пробілами. Не варто писати багато операторів в одному рядку. Для поділу закінчених логічних фрагментів програми слід використовувати порожні рядки та коментарі.

Отже, створення програми треба починати з визначення її вихідних даних і результатів. При цьому слід замислюватись не тільки щодо змісту кожної величини, але й про те, яку множину значень вона може приймати. Відповідно до цього обираються типи змінних.

На наступному кроці доцільно записати, що саме і як повинна робити програма природною мовою (наприклад, українською). При цьому варто застосовувати узагальнені блок-схеми або псевдокоди. Це допоможе детально продумати алгоритм, знайти на самій ранній стадії деякі помилки, збудувати програму у вигляді логічної послідовності блоків, а також забезпечити коментарі до програми.

Під час кодування, тобто в процесі написання тексту програми, слід пам'ятати про принципи структурного програмування: програма має складатись з чіткої послідовності блоків — базових конструкцій, кожна з яких має один вхід та один вихід. Конструкції можуть вкладатися, але не перетинатися.

Головна мета, до якої треба прагнути, — отримати зрозумілу програму якомога простішої структури. Врешті-решт, усі технології програмування спрямовані на досягнення саме цієї мети, адже тільки так можна добитися надійності програми та легкості її модифікації.

## БІБЛІОГРАФІЧНИЙ СПИСОК

1. Mayo, Joe. Tutorial Microsoft Visual Studio [Електронний ресурс] / Joe Mayo. – Режим доступу або URb: <http://padabum.com/d.php?id=125024>. – 30.07.2020.
2. C # Programming in a high level language. Textbook for high schools [Електронний ресурс]. – Режим доступу або URb: <https://diamail.com.ua/bookout/6180.html>. – 30.07.2020.
3. Культин, Н. Б. Основы программирования в Microsoft Visual C# / Н. Б. Культин. – СПб. : БХВ-Петербург, 2011. – 368 с.
4. Мамонова, Т. Е. Информатика. Общая информатика. Основы языка C++ : учеб. пособие / Т. Е. Мамонова. – Томск : Изд-во Томск. политехн. ун.-ту, 2011. – 206 с.
5. Флейшман, Б. С. Основы системологии / Б. С. Флейшман. – М. : Радио и связь, 1982. – 368 с.
6. Гагарина, Л. Г. Алгоритмы и структуры данных: учеб. пособие / Л. Г. Гагарина, В. Д. Колдаев. – М. : Финансы и статистика; ИНФРА-М, 2009. – 304 с.
7. Культин, Н. Б. Microsoft Visual C# в задачах и примерах / Н. Б. Культин. – СПб. : БХВ-Петербург, 2009. - 320 с.
8. Рендольф, Ник. Visual Studio 2010 для профессионалов : пер. с англ. / Ник Рендольф и др. – М. : ООО "И. Д. Вильямс", 2011. - 1184 с.
9. Кристиан, Н. C# 5.0 и платформа .NET 4.5 для профессионалов : пер. с англ. / Н. Кристиан. – М. : ООО "И. Д. Вильямс", 2014. – 1440 с.
10. Албахари, Дж. C# 6.0. Справочник. Полное описание языка : пер. с англ. / Дж. Албахари. – 6-е изд. – СПб. : БХВ-Петербург, 2016. – 1040 с.

Навчальне видання

**Андрєєв Сергій Михайлович**

**Жилін Володимир Анатолійович**

**Нечаусов Артем Сергійович**

## **АЛГОРИТМІЧНІ ОСНОВИ ГЕОМАТИКИ І СИСТЕМОЛОГІЇ**

Редактор С. П. Гевло

Зв. план, 2020

Підписано до друку 19.10.2020

Формат 60×84 1/16. Папір офс. № 2. Офс. друк

Ум. друк. арк. 4,9. Обл.-вид. арк. 5,5. Наклад 75 пр.

Замовлення 265. Ціна вільна

---

Видавець і виготовлювач

Національний аерокосмічний університет ім. М. Є. Жуковського

«Харківський авіаційний інститут»

61070, Харків-70, вул. Чкалова, 17

<http://www.khai.edu>

Видавничий центр «ХАІ»

61070, Харків-70, вул. Чкалова, 17

[izdat@khai.edu](mailto:izdat@khai.edu)

Свідоцтво про внесення суб'єкта видавничої справи  
до Державного реєстру видавців, виготовлювачів і розповсюджувачів  
видавничої продукції сер. ДК № 391 від 30.03.2001