Лабораторна робота № 5

Обробка одновимірних масивів і матриць

Мета роботи –набуття практичних навичок щодо обробки одновимірних масивів та освоєння методики опрацювання таблиць.

Дана лабораторна робота сприяє напрацюванню таких **компетентностей** відповідно до Національної рамки кваліфікацій:

знання:

призначення, оголошення й визначення масиву;

способів доступу до елементів масиву;

способів ініціалізації елементів масиву;

типових алгоритмів перетворення масивів;

алгоритмів сортування елементів масиву;

уміння:

складати програми, дані в яких надані у вигляді відповідних масивів;

виконувати налагодження та покрокове тестування типових програм перетворення масивів у середовищі системи Visual C# .NET;

розробляти програми формування багаторядкових табличних документів;

комунікації:

обґрунтування рекомендацій команді учасників проекту щодо доцільності застосування поданих даних у вигляді відповідних масивів;

робота в команді окремими частинами складного коду, який містить обробку масивів;

автономність і відповідальність:

прийняття рішення щодо доцільності застосування відповідних алгоритмів перетворення масивів;

самостійне обґрунтування можливих варіантів обробки таблиць.

Основні положення

Масив – це набір елементів одного і того ж типу, об'єднаних загальним ім'ям.

Масиви в C# можна використовувати за аналогією з тим, як вони використовуються в інших мовах програмування. Однак C# - масиви мають суттєві відмінності: вони відносяться до посилальних типів даних, більш того ─ реалізовані як об'єкти.

Виділення пам'яті під елементи відбувається на етапі ініціалізації масиву. А за звільненням пам'яті стежить система збору сміття ─ невикористовувані масиви автоматично утилізуються даною системою.

Одновимірний масив ─ це фіксована кількість елементів одного і того ж типу, об'єднаних загальним ім'ям, де кожен елемент має свій номер.

Нумерація елементів масиву в C# починається з нуля, тобто, якщо масив складається з 10 елементів, то його елементи матимуть такі номери: 0, 1, 2, 3, 4, 5, 6, 7, 8, 9.

Одновимірний масив у C# реалізується як об'єкт, тому його створення є двоступеневим процесом. Спочатку оголошується посилальна змінна на масив, потім виділяється пам'ять під необхідну кількість елементів базового типу, і посилальної змінної присвоюється адреса нульового елемента в масиві. Базовий тип визначає тип даних кожного елемента масиву. Кількість елементів, які будуть зберігатися в масиві, визначається розмір масиву.

Загальний синтаксис для оголошення й створення масиву наведений в наступному синтаксичному блоці.

Оголошення масиву й присвоювання йому значення:

Оголошення_масиву ::=

<Базовий_тип> [ ] <Ідентифікатор масиву>;

Створення_масиву ::= new <Базовий_тип> [<Довжина_масиву>];

Присвоювання_посилання_на_масив ::= <Ідентифікатор_масиву> =

new <Базовий_тип> [<Довжина_масиву>];

Присвоювання_посилання_на_масив ::= <Оголошення_масиву>

new <Базовий_тип> [<Довжина_масиву>];

<Базовий_тип> в оголошенні повинен бути ідентичним <Базовий_тип> в операторі породження об'єкта.

<Довжина_масиву> повинна бути позитивною й належати типу, що неявно перетворюється до int. Це може бути літерал, константа або змінна.

Квадратні дужки [ ] в даному випадку є частиною синтаксису; вони не вказують на необов'язковість заключних у них елементів синтаксису.

Коли масив оголошений і присвоєне посилання на його об'єкт, можна звертатися і до його окремих елементів.

У загальному випадку процес оголошення змінної типу масив, і виділення необхідного обсягу пам'яті може бути розділене. Крім того, на етапі оголошення масиву можна провести його ініціалізацію. Тому для оголошення одновимірного масиву може використовуватися одна з таких форм запису:

Форма 1.

базовий_тіп [ ] ім’я__масива;

Наприклад:

int [ ] a;

Описано посилання на одновимірний масив, яке в подальшому може бути використане:

для адресації на вже існуючий масив;

передачі масиву в метод як параметр відстроченого виділення пам'яті під елементи масиву.

Форма 2.

базовий_тіп [ ] ім’я__масива = new базовий_тип [розмір];

Наприклад:

int [ ] a = new int [10];

Оголошено одновимірний масив заданого типу і виділена пам'ять під одновимірний масив зазначеного розміру. Адреса даної області пам'яті записана в посилальну змінну. Елементи масиву дорівнюють нулю.

Треба зазначити, що в C # елементам масиву присвоюються початкові значення за замовчуванням в залежності від базового типу.

Для арифметичних типів ─ нулі, для посилальних типів ─ null, для символів ─ пробіл.

Форма 3.

базовий_тип [ ] ім’я__масива = {список ініціалізації};

Наприклад:

int [ ] a = {0, 1, 2, 3};

Виділена пам'ять під одновимірний масив, розмірність якого відповідає кількості елементів у списку ініціалізації. Адреса цієї області

пам'яті записана в посилальну змінну. Значення елементів масиву відповідає списку ініціалізації.

Звернення до елементів масиву відбувається за допомогою індексу, для цього потрібно вказати ім'я масиву та в квадратних дужках його номер. Наприклад, a [0], b [10], c [i].

Оскільки масив є набором елементів, об'єднаних загальним ім'ям, то обробка масиву зазвичай проводиться в циклі.

Слід розглянути кілька простих прикладів роботи з одновимірними масивами.

Приклад 1. Програма виводить на екран монітору заданий масив у вигляді рядка

```
using System;
   class Program
   {
      static void Main(string[] args)
      {
                  int[ ] myArray = { 0, 1, 2, 3, 4, 5, 6, 7, 8, 9 };
         int i;
         for (i = 0; i < 10; ++i)
            Console.Write(" {0}",myArray[i]);
         Console.WriteLine();
      }
   }
```

Завдання. Змініть програму так, щоб числа виводилися в стовпчик.

Приклад 2.

```
static void Main ()
{
  int [ ] myArray = new int [10];
  int i;
  for (i = 0; i <10; i ++)
       myArray [i] = i * i;
  for (i = 0; i <10; i ++)
       Console.WriteLine (myArray [i]);
}
```

Завдання. Змініть програму так, щоб оброблявся масив із n чисел.

Під час ініціалізації масиву немає необхідності використовувати операцію new, все ж масив можна задати такий спосіб:

```
int [ ] myArray = new int [ ] {99, 10, 100, 18, 78, 23, 163, 9, 87, 49};
```

Незважаючи на надмірність, дана форма ініціалізації масиву може виявитися корисною в тому випадку, коли вже існуючої посиланню на одновимірний масив присвоюється посилання на новий масив. наприклад:

```
static void Main ()
{
  int [ ] myArray = {0, 1, 2, 3, 4, 5};
  int i;
  for (i = 0; i <10; i ++)
       Console.Write ("" + myArray [i]);
  Console.WriteLine ("\ nНовий масив:");
  myArray = new int [ ] {99, 10, 100, 18, 78, 23, 163, 9, 87, 49}; // 1
  for (i = 0; i <10; i ++)
      Console.Write ("" + myArray [i]);
}
```

Слід зазначити, що спочатку змінна myArray посилалася на 6-ти елементний масив. У рядку 1 змінної myArray була привласнена посилання на новий 10-елементний масив, у результаті чого вихідний масив надалі не застосовується, оскільки на нього тепер не посилається жоден об'єкт. Тому він автоматично буде видалений складальником сміття.

Приклад 3. Обчислення функції y = a*x^2 + sin(x)

```
using System;
class Class1
{
static void Main()
        {
        const double a = 10.5;
        double [ ] mas = new double[7];
//      double [ ] mas = {-1, -0.93, -0.49, 0, 1.13, 0.96, 1.75};
//      double [ ] mas = new double[7]  {-1, -0.93, -0.49, 0, 1.13, 0.96,
1.75};
        double y;
        // Введення масиву
        Console.WriteLine("Введіть значення елементів масиву");
        for ( int i=0; i < mas.Length; i++)
```

```
                {
                        Console.Write("mas[{0}] = ",i);
                        mas[i] = Convert.ToDouble(Console.ReadLine());
                }
        // Контрольне виведення масиву
        Console.WriteLine("Вихідний масив:");
        for ( int i=0; i < mas.Length; i++)
                {
                        Console.Write("{0} ",mas[i]);
                }
        Console.WriteLine(); // переведення рядка
        //  Обчислення функції
        for ( int i=0; i < mas.Length; i++)
                {
                y = a*mas [ i ] * mas [ i ] - Math.Sin ( mas [ i ] );
                Console.WriteLine("При  значенні  x={0},  y={1:N4}",mas[i],
y);
                }
        }
}
```

Завдання. Змінити програму для обчислення такої функції: y = a + 1/x + 1/x^2 + 1/x^3 +1/x^4 + … +1/x^n, де параметр n задається з клавіатури.

Приклад 4. Пузиркове сортування.

```
using System;
class Class1
{
        static void Main()
        {
        const double a = 10.5;
        double [ ] mas = new double[7];
        // double [ ] mas = {-1, -0.93, -0.49, 0, 1.13, 0.96, 1.75};
        // double [ ] mas = new double[7] {-1, -0.93, -0.49, 0, 1.13, 0.96,
1.75};
        double y;
        // Введення масиву
        Console.WriteLine("Введіть значення елементів масиву");
```

```
            for ( int i=0; i < mas.Length; i++)
                {
                    Console.Write("mas[{0}] = ",i);
                    mas[i] = Convert.ToDouble(Console.ReadLine());
                }
            // Контрольне виведення масиву
            Console.WriteLine("Вихідний масив:");
            for ( int i=0; i < mas.Length; i++)
                {
                    Console.Write("{0}  ",mas[i]);
                }
            Console.WriteLine(); // переведення рядка
            //  Обчислення функції
            double t;
                for ( int i=0; i < mas.Length-1; i++)
                    for ( int j=0; j < mas.Length-1; j++)
                if(mas[j] < mas[j+1])      // варіант if(mas[j] > mas[j+1])
                    {
                    t=mas[j];
                    mas[j] = mas[j+1];
                    mas[j+1]=t;
                    }
            // Контрольне виведення масиву
                Console.WriteLine("Масив після сортування:");
                for ( int i=0; i < mas.Length; i++)
                {
                    Console.Write("{0}  ",mas[i]);
                }
                Console.WriteLine(); // переведення рядка
            }
        }
```

Завдання. Змінити програму, щоб була можливість введення з клавіатури розмірності масиву.

Перебір елементів масиву за допомогою оператора foreach.

Крім циклу for, для проходження по всьому масиву можна скористатися оператором foreach.

Наприклад, для виведення значення кожного елемента масиву childbirths, оголошеного й визначеного як

```
uint [ ] childbirths =  {1340, 3240, 1003, 4987, 3877};
```

може використовуватися оператор foreach:

```
foreach (uint  temp  in  childbirths )
        {
          Console.WriteLine(temp);
        }
```

який дає виведення: 1340 3240 1003 4987 3877

Оператор foreach складається із заголовка й тіла (рис. 40). Тіло циклу може бути одиночним або складеним оператором. Перші два слова в круглих дужках заголовка є, відповідно, типом і ідентифікатором. Разом вони оголошують ітераційну змінну оператора foreach. У даному випадку вона називається temp (від слова temporary – тимчасовий).

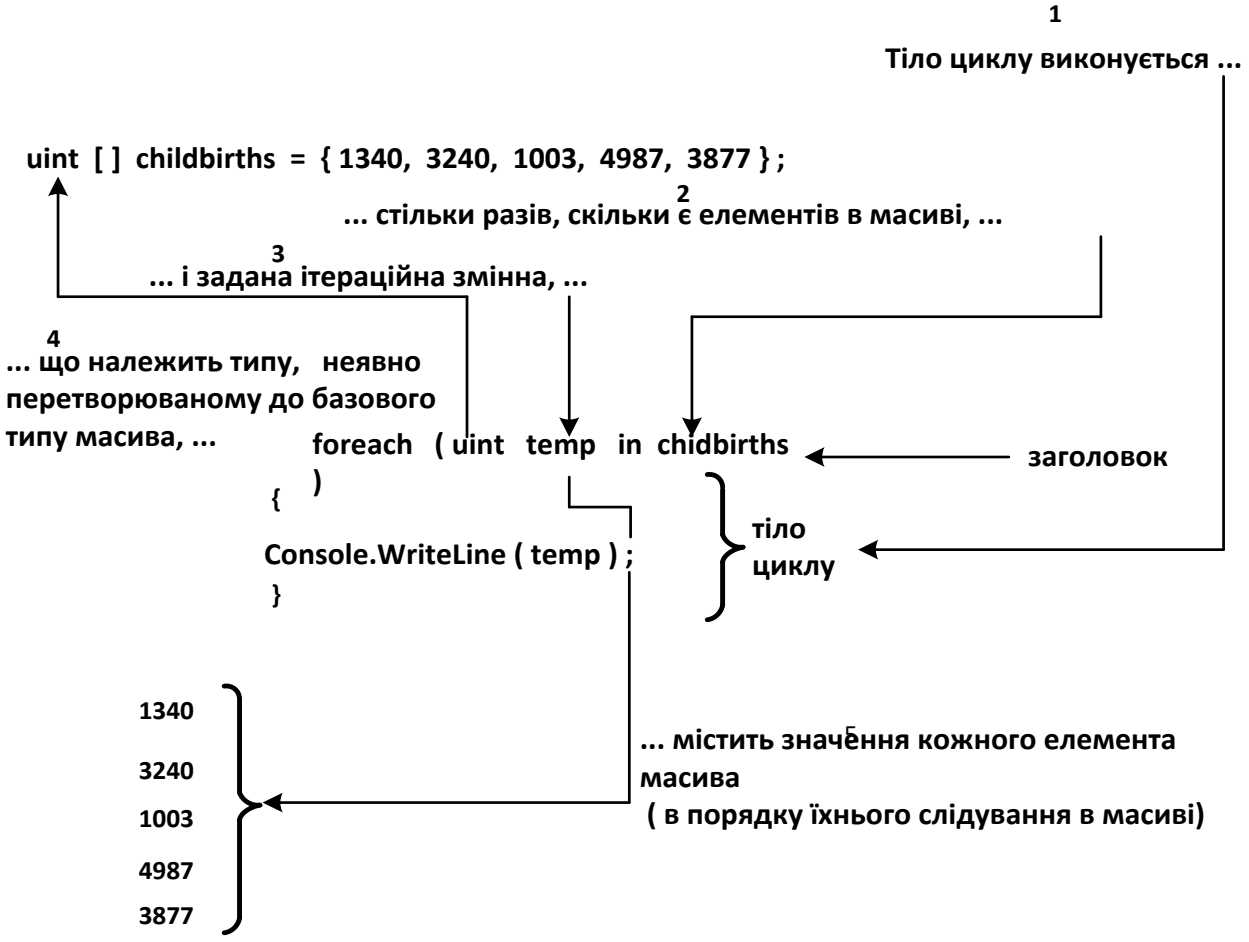


Рис. 40. **Оператор foreach**

Праворуч від ітераційної змінної знаходиться ключове слово in, за яким іде масив (у даному випадку childbirths). Важливо, щоб тип ітераційної змінної міг бути неявно перетворений в базовий тип масиву.

Тіло циклу виконується один раз для кожного елемента. Ітераційну змінну можна використовувати й у тілі циклу. Під час першого проходу (виконання) вона дорівнює першому елементу масиву й т.д.

Синтаксичний блок оператора foreach:

Оператор foreach ::=

foreach (<Тип> <Ідентифікатор_ітераційної_змінної>
in <Ідентифікатор_масиву>)
[< Оператор > | <Складений_оператор>]

Виконуючи оператор foreach, C# автоматично визначає кількість ітерацій. При цьому кожний елемент масиву присвоюється ітераційній змінній без необхідності явної індексації.

Лічильник, умова й оновлення циклу (необхідні в стандартному циклі for), в даному випадку непотрібні, що забезпечує простоту та ясність коду.

Багатовимірні масиви.

Оголошення й визначення двовимірного масиву

Мова C# дозволяє визначати двовимірні масиви, де для ідентифікації елемента потрібні два індекси (фактично в C# можна визначати масиви будь-якої розмірності).

На рис. 41 наведено приклад оголошення й визначення двовимірного масиву.

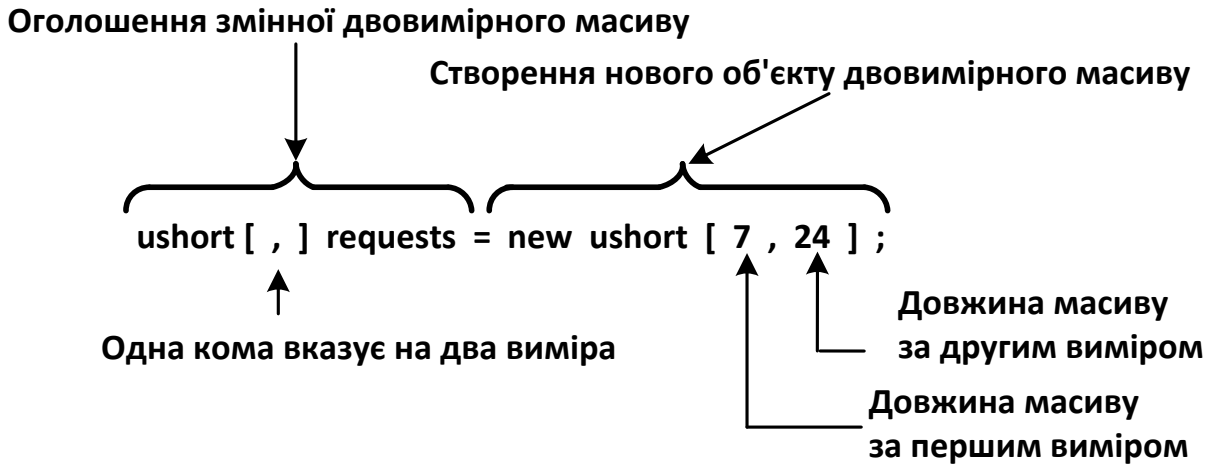
Оголошення змінної масиву, створення нового об'єкта і присвоювання змінній посилання на об'єкт (перший рядок на рис. 41) можна (як і у випадку одновимірного масиву) розбити на два оператори. Результат показано в нижній частині рис. 41.

Індекс першого виміру змінюється від 0 до 6, а другого – від 0 до 23.

Доступ до елементів двовимірного масиву.

Після оголошення змінної масиву і присвоювання їй посилання на двовимірний об'єкт можна звертатися до його окремих елементів.

За винятком того, що для звертання до елементів двовимірного масиву потрібен додатковий індекс, вони використовуються аналогічно елементам одномірного масиву. Отже, будь-який з них можна використовувати так само, як і окрему змінну базового типу.



Попередній оператор, як у випадку одновимірних масивів, можна розбити на два рядки:

```
ushort [ , ]  requests ;
requsts  =  new  ushort  [  7 , 24 ] ;
```

Рис. 41. **Приклад оголошення й визначення двовимірного масиву**

Наприклад:

requests[0,0] = (ushort) 89;

Тут застосовується операція приведення до типу (ushort), оскільки він є базовим типом requests. Вона необхідно, тому що літерал 89 належить до типу int, що не може бути неявно перетворений в ushort.

Подання двовимірного масиву як масиву масивів.

Два виміри масиву requests можна розглядати як масив масивів. Якщо звернутися до першого виміру requests (щонаводить, наприклад, дні тижня), сім днів можна вважати одновимірним масивом (рис. 42). Якщо тепер включити в розгляд години, то кожний елемент "день" можна вважати складеним з одновимірного масиву "години".

Приклад 5. Обробка матриці зарплати.

```
using System;
class Class1
{
        static void Main()
        {
        int kol_rab;          // кількість робітників у бригаді
```

```
int kol_brigad;     // кількість бригад
char vibor;         // для організації діалогу
do
{
//      Введення матриці із клавіатури
        Console.WriteLine("Введіть кількість робітників в
бригаді...");
        kol_rab=Convert.ToInt32(Console.ReadLine());
        Console.WriteLine("Введіть кількість бригад...");
        kol_brigad=Convert.ToInt32(Console.ReadLine());
```

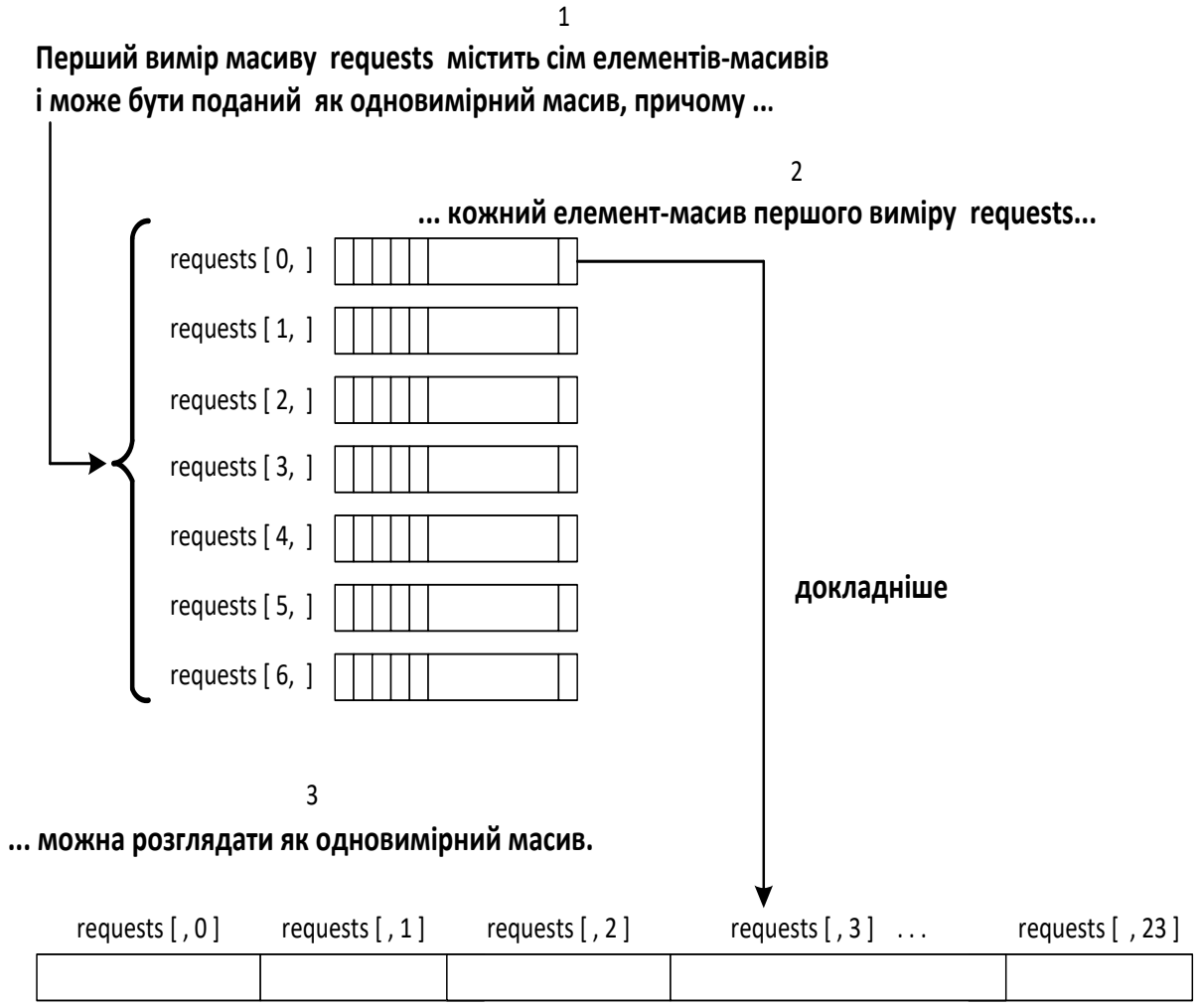


Рис. 41. **Двовимірний масив можна подати як два одновимірних масиви**

```
double  [,] Matr = new double[kol_rab,kol_brigad];
```

```
                for( int i=0; i < kol_rab;  i++ )
                {
                        for (int j=0; j < kol_brigad;  j++)
                        {
                        Console.WriteLine("Введіть зарплату {0} робітника
з {1} бригади", i+1,j+1);
                        Console.WriteLine("Matr[{0},{1}]=?",i,j);
                        Matr[i,j]=Convert.ToDouble(Console.ReadLine());
                        }
                }
            // Контрольне виведення матриці
                Console.WriteLine("Матриця зарплат:");
                for( int i=0; i < kol_rab;  i++ )
                {
                        for (int j=0; j < kol_brigad;  j++)
                        {
                                Console.Write("{0}  ",Matr[i,j]);
                        }
                Console.WriteLine();  // переведення рядка
                }
                // Обробка матриці
                Console.WriteLine("Розрахунок максимальної зарплати
по бригадах:\n");
                Console.WriteLine("Номер бригади     Номер робітника
Зарплата");
                int  n_rab = 0,       // номер робітника
                n_brigad = 0;         // номер бригади
                double max_zarplata = 0;
                for( int j=0; j < kol_brigad;  j++ )
                {
                        for (int i=0; i < kol_rab ;  i++)
                        {
                        if(Matr[i,j]>max_zarplata)
                        {
                                max_zarplata = Matr[i,j];
                                n_rab = i;
                                n_brigad = j;
```

```
                    }
                    }
                Console.WriteLine("{0,7}{1,15}{2,15}",
n_brigad+1,n_rab+1,max_zarplata );
                max_zarplata = 0;  // підготовка до наступної ітерації
                }
            // Діалог для продовження роботи
            Console.WriteLine(" Будете продовжувати роботу? Так - <Y>,
Ні - <N> " );
            vibor=Convert.ToChar(Console.ReadLine());
            } while( (vibor == 'y') || (vibor == 'Y') ); // закінчення циклу do /
        while
            Console.WriteLine("Роботу завершено!" );
            }
        }
```

Масив як об'єкт.

Масиви в C# реалізовані як об'єкти. Якщо говорити більш точно, то вони реалізовані на основі базового класу Array, визначеного в просторі імен System. Даний клас містить різні властивості і методи. Наприклад, властивість Length дозволяє визначати кількість елементів у масиві. Інші властивості і методи класу Array наведені в таблиці 6:

Таблиця 6

Властивості і методи класу Array

Елементи	Види	Опис
1	2	3
Length	Властивість	Кількість елементів масиву (за всіма вимірюванням)
BinarySearch	Статичний метод	Двійковий пошук у відсортованому масиві
Clear	Статичний метод	Присвоєння елементам масиву значень за замовчуванням
Copy	Статичний метод	Копіювання заданого діапазону елементів одного масиву в інший
CopyTo	Екземпляр методу	Копіювання всіх елементів поточного одновимірного масиву в інший масив
GetValue	Екземпляр методу	Отримання значення елемента масиву

Закінчення табл. 6.

1	2	3
IndexOf	Статичний метод	Пошук першого входження елемента в одновимірний масив
LastIndexOf	Статичний метод	Пошук останнього входження елемента в одновимірний масив
Reverse	Статичний метод	Зміна порядку проходження елементів на зворотний
SetValue	Екземпляр методу	Установка значення елемента масиву
Sort	Статичний метод	Упорядкування елементів одновимірного масиву

Виклик статичних методів відбувається через звернення до імені класу, наприклад, Array.Sort (myArray). В даному випадку відбудеться звернення до статичного методу Sort класу Array і передача даному методу як параметр об'єкт myArray ─ екземпляр класу Array.

Звернення до властивості або виклик екземпляра методу робиться через звернення до примірника класу, наприклад, myArray.Length або myArray.GetValue (i).

Приклад 6.

```
class Program
{
  static void Main ()
  {
    try
    {
      int [ ] MyArray;
      Console.Write ("Введіть розмірність масиву:");
      int n = int.Parse (Console.ReadLine ());
      MyArray = new int [n];
      for (int i = 0; i <MyArray.Length; ++ i)
      {
        Console.Write ("a [{0}] =", i);
        MyArray [i] = int.Parse (Console.ReadLine ());
      }
      PrintArray ("Вихідний масив:", MyArray);
      Array.Sort (MyArray);
      PrintArray ("Масив відсортований за зростанням", MyArray);
      Array.Reverse (MyArray);
```

```
            PrintArray ("Масив відсортований за спаданням", MyArray);
          }
          catch (FormatException)
          {
            Console.WriteLine ("Неправильний формат вводу даних");
          }
          catch (OverflowException)
          {
            Console.WriteLine ("Переповнення");
          }
          catch (OutOfMemoryException)
          {
            Console.WriteLine ("Недостатньо пам'яті для створення
нового об'єкта");
          }
        }
        static void PrintArray (string a, int [ ] mas)
        {
          Console.WriteLine (a);
          for (int i = 0; i <mas.Length; i ++) Console.Write ("{0}", mas [i]);
          Console.WriteLine ();
        }
      }
      }
```

Завдання. Додати в програму метод InputArray, який призначено для введення з клавіатури елементів масиву. Продемонструвати роботу даного методу.

Порядок виконання лабораторної роботи

Загальна частина.

1. Ознайомитися з прикладами програм, які були наведені в розділі «Основні положення» даної лабораторної роботи. Звернути увагу на додаткові завдання, які наведені після лістингів програм.

2. Проекспериментувати з програмами: змінити вихідні дані;

дослідити, як впливають синтаксичні помилки на результат компіляції програми. Які при цьому виникають помилки компіляції?

Індивідуальна частина.

1. Написати програму, яка здійснює обробку одновимірного масиву, що складається з n дійсних елементів. Варіанти алгоритмів обробки взяти з додаткового файлу з індивідуальними завданнями.

2. Написати програму, яка здійснює обробку квадратної матриці з n на n дійсних елементів. Варіанти алгоритмів обробки взяти з додаткового файла з індивідуальними завданнями.

3. Для пунктів 1 та 2 завдання підготувати контрольні приклади початкових масивів та результати їх обробки згідно з індивідуальним варіантом.

Для кожного з пунктів розробити графічну схему (блок-схему) відповідного алгоритму.

4. Відповідно до алгоритму набрати і відкомпілювати тексти програми, усуваючи у разі необхідності помилки.

6. Дослідити роботу програми, аналізуючи виконання контрольних чисельних прикладів.

Зміст звіту

1. Титульний лист.

2. Цілі лабораторного заняття і вказівка, які навички та вміння передбачається отримати в результаті його виконання.

3. Постановка задачі індивідуального завдання.

4. Для кожного з алгоритмів надати:

словесний опис його виконання, який супроводжується чисельним прикладом;

графічну схему (блок-схему) відповідного алгоритму.

5. Тексти налагоджених програм із результатом виконання всіх контрольних чисельних прикладів індивідуального завдання.

6. Висновки.

Контрольні запитання

1. У чому полягають особливості призначення, оголошення й визначення масиву?

2. Як відтворюється доступ до окремих елементів масиву?

3. Наведіть приклади варіантів ініціалізації масиву.

4. Опишіть загальну схему перебору елементів масиву за допомогою оператора foreach.

5. Наведіть приклади алгоритмів пошуку заданих елементів масиву.

6. Наведіть приклади алгоритмів перетворення масиву.

7. У чому суть алгоритму сортування елементів масиву методом «Пузирку»?

8. Як реалізується доступ до елементів двовимірного масиву?

9. У чому суть подання двовимірного масиву як масиву масивів?

10. Наведіть приклади обробки матриць.