

**МІНІСТЕРСТВО ВНУТРІШНІХ СПРАВ УКРАЇНИ  
ХАРКІВСЬКИЙ НАЦІОНАЛЬНИЙ УНІВЕРСИТЕТ  
ВНУТРІШНІХ СПРАВ**

**Кафедра інформаційних технологій факультету № 4**

**ТЕКСТ ЛЕКЦІЇ**

з навчальної дисципліни «Алгоритмізація та програмування»  
обов'язкових компонент  
освітньої програми першого рівня вищої освіти  
125 «Кібербезпека»  
(«Поліцейська діяльність у кіберсфері», «Безпека інформаційних та  
комунікаційних систем»)

**за темою – «Розробка програм обробки масивів»**

**Харків 2019**

## **ЗАТВЕРДЖЕНО**

Науково-методичною радою  
Харківського національного  
університету внутрішніх справ  
Протокол від 24.01.19 № 1

## **СХВАЛЕНО**

Вченою радою факультету № 4  
Протокол від 16.01.19 № 1

## **ПОГОДЖЕНО**

Секцією Науково-методичної ради  
ХНУВС з технічних дисциплін

Протокол від 17.01.19 № 1

Розглянуто на засіданні кафедри інформаційних технологій (протокол від 15.01.19 № 1)

### **Розробники:**

1. Зав. кафедри, к.т.н., доцент Струков В.М.

### **Рецензенти:**

1. д.т.н., професор Зацеркляний М.М.

## План лекції

1. Одновимірні масиви.
2. Двовимірні масиви.

## Література:

### Основна:

1. Э. Троелсен. Язык программирования C# 5.0 и платформа .NET 4.5. , 6-е изд. : Пер. с англ. — М. : ООО "И.Д. Вильямс", 2013. — 1312 с. : ил. — Парал. тит. англ.
2. Конспект лекцій.

### Додаткова:

3. Программирование на языке C#: учеб. пособие / Л.В. Соловей, Н.Н. Мирошниченко, Н.Г. Пономарёв. – Х. : НТУ «ХПИ», 2016. – 356 с.

## Текст лекції

### 1. Одновимірні масиви.

Масив - це найбільш поширена структура даних для будь-якої алгоритмічної мови високого рівня. У загальному випадку масив може розташовуватися на будь-якому носії інформації. Але традиційно в програмуванні під масивом, якщо це не обумовлено інакше, мають на увазі набір даних, розташованих в оперативній пам'яті.

Масивом будемо далі називати набір однотипних змінних, що мають однакове ім'я і пронумеровані за порядком. Звернення до елементу масиву здійснюється шляхом зазначення імені масиву і його номера (індексу) у квадратних дужках - `a[1]`. Для розміщення масиву в оперативній пам'яті виділяється безперервна ділянка. Її загальний обсяг залежить від типу і кількості елементів масиву. Нумерація елементів масиву в C# починається з нуля. Тому останній елемент масиву буде мати номер на одиницю менший кількості елементів масиву.

Кількість елементів називається розміром масиву.

Допустимі варіанти оголошення одновимірного масиву:

```
<тип> []<ім'я>; // 1
<тип> []<ім'я>= new <тип> [<розмір>]; // 2
<тип> []<ім'я>= {<список_ініціалізаторов>;} // 3
<тип> []<ім'я>= new <тип> [] {<список_ініціалізаторов>;} // 4
<тип> []<ім'я>= new <тип> [<розмір>] {<список_ініціалізаторов>;} // 5
```

<тип> - оголошує конкретний тип елементів масиву; квадратні дужки вказують на те, що оголошується одновимірний масив;

<розмір> - визначає число елементів масиву.

Масив є динамічним типом даних. Тому при його оголошенні у форматі //1 в оперативній пам'яті виділяється комірка, в якій зберігатиметься адреса початку безпосередньо масиву (екземпляра), але оперативна пам'ять для елементів масиву не виділяється. Для створення екземпляра масиву (виділення його елементів оперативної пам'яті і присвоєння їм стартових значень) необхідно виконати відповідну процедуру – *new*, як це зроблено у зразку // 2. Тут же вказана і потрібна кількість елементів масиву - <розмір>. Після створення екземпляра масиву його можна використовувати – записувати в нього конкретні значення і зчитувати значення елементів. При оголошенні масиву можна виконувати його ініціалізацію – задавати конкретні значення елементів (варіанти // 3, // 4, // 5).

Приклади оголошення одновимірних масивів:

```
int[] nums1; //1
int[] nums2 = new int[4]; //2
int[] nums3 = new int[4] { -1, 2, -3, 5 }; //3
int[] nums4 = new int[] { 1, 2, 3, 5 }; //4
int[] nums5 = new[] { 1, 2, 3, 5 }; //5
int[] nums6 = { 1, 2, 3, 5 }; //6
```

У випадку //1 оголошується масив *nums1*, але екземпляр масиву не створюється, тобто цей масив ще використовувати неможна.

У випадку //2 оголошується масив і створюється його екземпляр – масив із 4 елементів цілого типу і в них записуються значення по замовчуванню, для цілого типу – це 0.

Оголошення //3, //4, //5 і //6 еквівалентні – оголошується масив *nums2*, створюється його екземпляр із 4 елементів цілого типу, який ініціалізується значеннями, які перераховані у фігурних дужках.

Якщо при оголошенні не заданий розмір (випадок //4), кількість елементів обчислюється за кількістю ініціалізованих значень.

Операцію *new* можна опускати, вона буде виконана за замовчуванням (випадок //6).

Приклад. Введення елементів масиву з клавіатури.

```
int [] B;
B = new int [30];
int n;
Console.WriteLine(□Введіть кількість елементів□); // 3
n = Convert.ToInt32(Console.ReadLine()); // 4
B = new int[n];
for (int i = 0; i < n; i++)
    B[i] = Convert.ToInt32(Console.ReadLine());
```

Приклад. Виведення елементів масиву на екран.

```
int[] A = new int[10];
int i;
// Формування елементів масиву
```

```

for (i = 0; i <= 9; i = i + 1)
    A[i] = i;
// Виведення елементів масиву на екран
Console.WriteLine("Масив A: ");
Console.WriteLine();
for (i = 0; i <= 9; i = i + 1)
    Console.WriteLine(" A[{0}]= {1} ", i, A[i]);
Console.ReadLine();

```

Всі масиви в C # побудовані на основі базового класу *Array*, який містить властивості та методи, корисні при складанні програм і які можна використовувати для обробки масивів. Нижче в таблиці наведено деякі з них.

Елемент	Вид	Опис
Length	властивість	Кількість елементів масиву (за всіма розмірностями)
Copy	метод	Копіювання заданого діапазону елементів одного масиву в інший
CopyTo	метод	Копіювання всіх елементів поточного одновимірного масиву в інший одновимірний масив
IndexOf	метод	Пошук першого входження елемента в одновимірний масив
LastIndex	метод	Пошук останнього входження елемента в одновимірний масив
Reverse	метод	Зміна порядку елементів на зворотний
Sort	метод	Сортування елементів одновимірного масиву

Для заповнення елементів масиву часто використовуються генератори псевдовипадкових чисел. Псевдовипадкові числа вибираються з рівною ймовірністю з кінцевого набору чисел. Обрані числа не є строго випадковими, так як для їх вибірки використовується детермінований математичний алгоритм, але вони досить випадкові для практичного застосування.

В нижченаведеній таблиці наведено список методів класу *Random*, використовуваних для генерації різних типів випадкових чисел.

Таблиця. Список методів класу *Random*.

Метод	Призначення
Next()	Повертає невід'ємне випадкове ціле 32-бітове число зі знаком в діапазоні 0..Int32.MaxValue (Int32.MaxValue = 2 147 483 647)
Next(MaxVal)	Повертає невід'ємне випадкове число, що не перевищує максимально допустиме значення, передане у вигляді аргументу, яке повинно знаходитися в діапазоні 0..Int32.MaxValue.
Next(MinVal, MaxVal)	Повертає випадкове число n в зазначеному діапазоні $\text{MinVal} \leq n < \text{MaxVal}$ . Значення аргументів повинні перебувати в діапазоні 0..Int32.MaxValue. Другий аргумент повинен бути більше або дорівнювати першому.

Приклад. Заповнення масиву випадковими числами.

```
int[] A = new int[10];  
int i;  
Random rnd = new Random();  
    // Формування елементів масиву  
    for (i = 0; i <= 9; i++)  
        A[i] = Convert.ToInt32(rnd.Next(1, 10));  
Console.ReadLine();
```

## 2. Типові програмні рішення обробки одновимірних масивів.

Приклад 2.1. Визначення суми і добутку елементів одновимірного масиву.

```
const int n = 10;  
int i;  
int[] A = new int[n] { 1,-4,-7,8,-10,12,-14,-17,20,21 };  
long sum = 0; // сума елементів  
long dob = 1; // добуток елементів  
for (i=0; i<=n-1; i++)  
{  
    sum = sum + A[i];  
    dob = dob * A[i];  
}  
Console.WriteLine("Сума елементів = " + sum);  
Console.WriteLine("Добуток = " + dob);  
Console.ReadLine();
```

Приклад 2.2. Визначення максимального і мінімального елементів одновимірного масиву.

```
const int n = 10;  
int i;  
int[] A = new int[n] { 1,-4,-7,8,-10,12,-14,-17,20,21 };  
long max = A[0];  
long min = A[0];  
for (i=0; i<=n-1; i++)  
{  
    if (A[i]>max) max=A[i];  
    if (A[i]<min) min=A[i];  
}  
Console.WriteLine("Максимальний елемент = " + max);  
Console.WriteLine("Мінімальний елемент = " + min);  
Console.ReadLine();
```

Приклад 2.3. Визначення номеру максимального елемента одновимірного

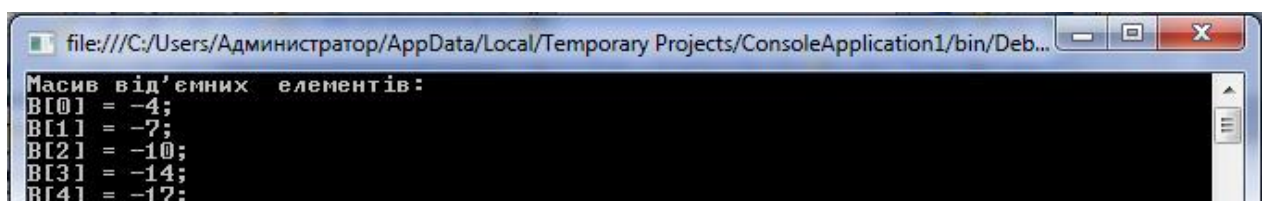
масиву.

```
const int n = 10;
int i;
int[] A = new int[n] { 1,-4,-7,8,-10,12,-14,-17,20,21 };
long max = A[0];
long imax = 0;
for (i=1; i<=n-1; i++)
{
    if (A[i]>max)
    {
        max=A[i];
        imax=i;
    }
}
Console.WriteLine("Максимальний елемент = " + max);
Console.WriteLine("Номер максимального елемента = " + imax);
Console.ReadLine();
```

Приклад 2.4. Переписати всі від'ємні елементи одновимірного масиву в інший одновимірний масив.

```
const int n = 10;
int i;
int k=0;
int[] A = new int[n] { 1,-4,-7,8,-10,12,-14,-17,20,21 };
int[] B = new int[n];
for (i=0; i<n-1; i++)
{
    if (A[i]<0)
    {
        B[k]=A[i];
        k = k + 1;
    }
}
Console.WriteLine("Масив від'ємних елементів: ");
for (i=0; i<=k-1; i++)
    Console.WriteLine("B[{0}] = {1}; ",i,B[i]);
Console.ReadLine();
```

Результат роботи програми матиме наступний вигляд:



```
file:///C:/Users/Администратор/AppData/Local/Temporary Projects/ConsoleApplication1/bin/Deb...
Масив від'ємних елементів:
B[0] = -4;
B[1] = -7;
B[2] = -10;
B[3] = -14;
B[4] = -17;
```

Приклад 2.5. Відсортувати елементи одновимірного масиву за зростанням.

```
// введення чисел
int[] nums = new int[5];
Console.WriteLine("Введіть п'ять чисел");
for (int i = 0; i < nums.Length; i++)
{
    Console.Write("{0}-е число: ", i + 1);
    nums[i] = Int32.Parse(Console.ReadLine());
}

// сортування масиву
int temp;
for (int i = 0; i < nums.Length-1; i++)
{
    for (int j = i + 1; j < nums.Length; j++)
    {
        if (nums[i] > nums[j])
        {
            temp = nums[i];
            nums[i] = nums[j];
            nums[j] = temp;
        }
    }
}

// виведення відсортованого масиву
Console.WriteLine("Виведення відсортованого масиву: ");
for (int i = 0; i < nums.Length; i++)
{
    Console.WriteLine(nums[i]);
}
Console.ReadLine();
```

В цьому прикладі використаний бульбашковий алгоритм сортування.

### **3. Багатовимірні масиви.**

Окрім одновимірних масивів можна використовувати багатовимірні масиви – двовимірні, трьохвимірні і т.д. Найбільш поширеними є двовимірні масиви. Вони представляють матрицю із рядків і стовпчиків. Рядки і стовпчики пронумеровані за порядком починаючи з 0. Звернення до будь-якого елементу матриці здійснюється вказівкою імені масиву та номеру рядка і номеру



стовпчика, на перехресті яких цей елемент розташований.

Нижче наведені всі можливі варіанти оголошення двовимірних масивів:

```
int[,] nums1; // 1
int[,] nums2 = new int[2, 3]; // 2
int[,] nums3 = new int[2, 3] { { 0, 1, 2 }, { 3, 4, 5 } }; // 3
int[,] nums4 = new int[, ] { { 0, 1, 2 }, { 3, 4, 5 }, { 5, 4, 5 } }; // 4
int[,] nums5 = new [, ] { { 0, 1, 2 }, { 3, 4, 5 } }; // 5
int[,] nums6 = { { 0, 1, 2 }, { 3, 4, 5 } }; // 6
```

У випадку // 1 оголошений масив *nums1*, але оперативна пам'ять для нього не виділена. Тому для подальшого використання цього масиву у програмі необхідно виконати ще одну процедуру – створення екземпляру масиву за допомогою оператора *new*.

У випадку // 2 оголошений масив *nums2*, для нього виділена оперативна пам'ять, всім елементам матриці за замовчуванням присвоєне значення 0.

Випадки // 3 - // 6 еквівалентні: оголошений відповідний двовимірний масив (*nums3-nums6*), для кожного виділена оперативна пам'ять, всім елементам матриці присвоєні значення, що перелічені у фігурних дужках.

Приклади звернення до елементів матриць:

```
nums2[0,2]=5; // 1
nums2[i,j]= nums2[i,j-1]; // 2
```

У випадку // 1 елементу матриці, який розташований на перехресті першого рядка (номер 0) і третього стовпчика (номер 2), присвоєне ціле число 5.

У випадку // 2 елементу матриці, який розташований на перехресті рядка з номером *i* і стовпчика з номером *j*, присвоєне значення елемента матриці, який стоїть в тому ж рядку, що і перший, але в стовпчику, номер якого на одиницю менше попереднього. Зауважимо, що в даному випадку значення *i* та *j* повинні бути визначеними до звернення.

Від багатовимірних масивів треба відрізнити масив масивів або так званий "зубчастий масив":

```
int[][] nums = new int[3][];
nums[0] = new int[2] { 1, 2 };
nums[1] = new int[3] { 1, 2, 3 };
nums[2] = new int[5] { 1, 2, 3, 4, 5 };
```

Тут дві групи квадратних дужок вказують, що це масив масивів, тобто такий масив, елементами якого, в свою чергу, є інші масиви. В даному випадку у нас масив *nums* містить три масиви. Причому розмірність кожного з цих масивів може бути різною.

В якості елементів масивів можна використовувати і багатовимірні масиви:

```
int[,] [,] nums = new int[3][,]
{
```

```

new int[,] { {1,2}, {3,4} },
new int[,] { {1,2}, {3,6} },
new int[,] { {1,2}, {3,5}, {8, 13} }
};

```

В таких випадках відповідним чином виконується звернення до елементів масивів:

```
nums[0][0, 1] = 9;
```

В цьому прикладі в першу матрицю (номер 0) в комірку, що розташована на перехресті першого рядка і другого стовпчика ( [0,1]), записується число 9.

#### 4. Перебір елементів масивів. Цикл *foreach*.

Цикл *foreach* призначений для перебору елементів в контейнерах, в тому числі в масивах. Слід зазначити, що цей оператор насправді більшою мірою призначений для перебору елементів таких об'єктів, які не мають безпосередньої нумерації (списки, колекції та т.і.), тобто у випадках, коли неможливо застосувати безпосередньо розглянуті раніше оператори циклу. Формат запису оператора *foreach*:

```

foreach (<тип> <имя_змінної> in <контейнер>)
{
    // оператори тіла циклу
}

```

Оператор *foreach* циклічно перебирає елементи масиву по черзі від початку і до кінця.

**Особливість.** Змінна циклу в операторі *foreach* служить тільки для читання. Це означає, що присвоювати цій змінній явним чином в тілі циклу будь-яке значення заборонено.

Приклад.

```

const int n = 5;
int[] A = new int[n] {2,5,3,1,4};
foreach (int k in A) Console.WriteLine(" {0}, ",k);
Console.ReadLine();

```

#### 5. Типові програмні рішення обробки двовимірних масивів.

Приклад 1. Сума від'ємних елементів матриці.

```

int[,] m = {{ -3, 8, 1},
            { -4, 0, 10} };
int s = 0;
for (int i = 0; i < 2; i++)
{
    for (int j = 0; j < 3; j++)
        if (m[i,j] < 0) s = s + m[i,j];
}

```

```
Console.WriteLine("Сума={0}", s);
Console.ReadLine();
```

Приклад 2. Максимальний елемент матриці.

```
int[,] m = {{ -3, 8, 1},
            { -4, 0, 10} };
int max = m[0, 0];
for (i = 0; i < 2; i++)
{
    for (j = 0; j < 3; j++)
        if (m[i, j] > max) max = m[i, j];
}
Console.WriteLine("Max={0}", max);
Console.ReadLine();
```

Приклад 3. Мінімальний елемент кожного рядка матриці.

```
int[,] m = {{ -3, 8, 1},
            { -4, 0, 10} };
int i,j,min;
for (i = 0; i < 2; i++)
{
    min = m[i, 0];
    for (j = 0; j < 3; j++)
        if (m[i, j] < min) min = m[i, j];
    Console.WriteLine("Рядок {0} Min={1}",i,min);
}
Console.ReadLine();
```

Приклад 4. Координати мінімального елемента матриці.

```
int[,] m = {{ -3, 8, 1},
            { -4, 0, 10} };
int i,j,min=m[0,0], imin, jmin;
for (i = 0; i < 2; i++)
{
    for (j = 0; j < 3; j++)
        if (m[i, j] < min)
        {
            min = m[i, j];
            imin = i;
            jmin = j;
        }
}
Console.WriteLine(" imin= {0} jmin ={1}", imin, jmin);
```

```
Console.ReadLine();
```