# **Масиви**

**Масив** – іменована структура даних, фіксованого розміру, яка дозволяє зберігати, послідовність однотипних елементів, до яких можна звертатися за допомогою індексу.

С# підтримує наступні типи масивів:

* одновимірні;
* ступінчасті;
* багатовимірні.

## **Одновимірні масиви (Single-Dimensional arrays)**

**Одновимірний**, або лінійний масив – це набір елементів фіксованої довжини та наперед заданого типу, доступ до яких здійснюється з використанням одного індексу.

Кожен зі створених одновимірних масивів має вигляд:

| ***1*** | ***2*** | ***3*** | ***4*** | ***5*** | ***6*** | ***7*** | ***8*** | ***9*** | ***10*** |
| --- | --- | --- | --- | --- | --- | --- | --- | --- | --- |

Синтаксис оголошення масиву, подібний до створення змінної, тільки після типу вказують квадратні дужки:

тип[] ім'я\_масиву;

Приклад оголошення цілочисельного масиву:

int[] array;

В мові C#, самі масиви мають тип даних **посилання**, це означає, що після оголошення змінної типу масив, виділяється пам’ять, тільки для зберігання посилання на екземпляр масиву, при цьому значення змінної рівне null. Для створення екземпляру масиву, а отже виділення пам’яті для зберігання елементів, необхідно використовувати оператор **new**.

Для створення масиву використовується наступний синтаксис:

тип[] ім'я\_масиву = new тип[довжина\_масиву];

Присвоїмо, попередньо створений, змінній масив з десяти елементів:

array = new int[10];

При цьому всі елементи масиву будуть мати тип **int**, і значення за замовчуванням для цього типу даних - 0.

Масиви, подібно до змінних, підтримують ініціалізацію, для цього використовуються масиви-літерали. В C#, можна ініціалізувати масив декількома варіантами, які для компілятора є рівноцінними:

int[] arr1 = new int[5] { 0, 2, 4, 6, 8 };

int[] arr2 = new int[] { 0, 2, 4, 6, 8 };

int[] arr3 = new[] { 0, 2, 4, 6, 8 };

var arr4 = new[] { 0, 2, 4, 6, 8 };

int[] arr5 = { 0, 2, 4, 6, 8 };

*В перших трьох варіантах, можна використовувати* ***var*** *замість* ***int[]***.

Для доступу до елементів масиву, використовуються **індекси**, при цьому початковий індекс рівний **нулю**, відповідно кінцевий індекс на одиницю менший за довжину.

var a = new[] { 10, 20, 30 };

int t = a[2]; //t = 30, оскільки елемент з індексом 2 має значення 30

a[0] = 15; // 15, 20, 30

a[1] = 25; // 15, 25, 30

Довжину одновимірного масиву, можна отримати з поля **Length**.

**.Length** - повертає кількість елементів у масиві. Наприклад, створимо масив arr і вкажемо для нього 3 елементи. При виклику функції буде видано число 3: arr.Length;

Зазвичай для роботи з масивами, використовуються цикли. Розглянемо приклад заповнення та виводу даних масиву:

byte[] bytes = new byte[7];

//заповнення масиву

for (byte i = 0; i < bytes.Length; i++)

{

bytes[i] = i;

}

//друк значень елементів в консоль

//під час кожної ітерації циклу foreach, змінна b приймає наступне значення з масиву

foreach (var b in bytes)

{

Console.WriteLine(b);

}

Як було описано вище, масив в C#, має **тип посилання**(*reference type*), не плутати з типом елементів. Це означає, що у змінній з назвою масиву, зберігається посилання екземпляр, а не самі дані. Це потрібно враховувати при роботі з масивами.

Розглянемо ситуацію, коли потрібно скопіювати дані одного цілочисельного масиву в інший. У випадку зі змінними примітивних типів даних, ми могли записати копіювання наступним чином:

var k = 9;

var p = k; // k = 9; p = 9;

k = 7; // k = 7; p = 9;

p = 2; // k = 7; p = 2;

Якщо застосувати таку саму логіку роботи до масивів, отримаємо:

var a = new[] { 1, 2, 3 };

var b = a; //a == {1, 2, 3}; b == {1, 2, 3}

a[0] = 9; //a == {9, 2, 3}; b == {9, 2, 3}

b[2] = 8; //a == {9, 2, 8}; b == {9, 2, 8}

У другому рядку коду, ми присвоюємо змінній **b** посилання на ті самі дані, на які вказує змінна **a**, тобто фактично у нас є тільки один набір елементів, а доступні вони через дві змінні.

Для створення фізичної копії масиву, можна використати наступний синтаксис:

int[] A = new[] { 10, 20, 30 };

int[] B = new int[A.Length];

for (int i = 0; i < B.Length; i++)

{

B[i] = A[i];

}

Крім цього клас **Array** дозволяє нам зробити копію масиву, викликом методу **Copy**:

int[] a = { 11, 12, 13, 14 };

int[] b = new int[a.Length];

Array.Copy(a, b, a.Length);

## **Ступінчастий масив (Масив масивів, Jagged Arrays)**

Масив масивів - це масив, елементи якого є масивами і можуть бути різних розмірів. Масив із масивом іноді називають "масивом масивів".

Тип даних масиву може бути будь-яким, тому ми можемо створити масив масивів, різної розмірності, саме така структура має назву – **ступінчастий масив**.

Для створення ступінчастого масиву використовується такий синтаксис:

byte[][] arr = new byte[3][];

arr[0] = new byte[4] { 1, 2, 3, 4 }; //створюємо 1-й підмасив

arr[1] = new byte[5] { 1, 2, 3, 4, 5 }; //створюємо 2-й підмасив

arr[2] = new byte[8] { 1, 2, 3, 4, 5, 6, 7, 8 }; //створюємо 3-й підмасив

Він має вигляд:

| ***1*** | ***2*** | ***3*** | ***4*** |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- | --- |
| ***1*** | ***2*** | ***3*** | ***4*** | ***5*** |  |  |  |
| ***1*** | ***2*** | ***3*** | ***4*** | ***5*** | ***6*** | ***7*** | ***8*** |

Для звертання до елементу, використовуються два індекси:

arr[1][2] = 0;

Console.WriteLine(arr[0][3]);

Використовуючи вкладені цикли, можна здійснювати обхід всіх елементів ступінчастого масиву:

char[][] symbolTable = new char[][]

{

new char[]{ 'a','b', 'c'},

new char[]{ 'f','g', 'h', 'j','k'},

new char[]{ 'w','x', 'y', 'z'}

};

foreach (var symbolRow in symbolTable)

{

foreach (var symbol in symbolRow)

{

Console.Write("{0}\t", symbol);

}

Console.WriteLine();

}

## **Багатовимірний масив(Multidimensional arrays)**

Для багатовимірних масивів характерним є **ранг**(*rank*), або кількість вимірів. Двовимірний масив – це таблиця, тривимірний – куб…

string[,] table = new string[4, 5]; //двовимірний масив

string[,,] cube = new string[3, 3, 3]; //тривимірний масив

Створення двовимірного масиву можна записати наступним чином(всі варіанти рівнозначні):

int[,] table1 = new int[3, 3] { { 1, 2, 3 }, { 4, 5, 6 }, { 7, 8, 9 } };

int[,] table2 = new int[,] { { 1, 2, 3 }, { 4, 5, 6 }, { 7, 8, 9 } };

int[,] table3 = new [,] { { 1, 2, 3 }, { 4, 5, 6 }, { 7, 8, 9 } };

int[,] table4 = { { 1, 2, 3 }, { 4, 5, 6 }, { 7, 8, 9 } };

*В перших трьох варіантах, замість* ***int[,]*** *можна використовувати* ***var***.

Кожен зі створених двовимірних масивів має вигляд:

| ***1*** | ***2*** | ***3*** |
| --- | --- | --- |
| ***4*** | ***5*** | ***6*** |
| ***7*** | ***8*** | ***9*** |

Доступ до елементів здійснюється через два індекси:

int[,] tbl = new int[2, 3];

tbl[0, 0] = 10;

tbl[1, 1] = 20;

Масиви можуть мати і більшу кількість вимірів. Оголошення тривимірного масиву могло б виглядати так:

int[,,] nums = new int[2, 3, 4];

Відповідно можуть бути і чотиривимірні масиви, і масиви з великою кількістю вимірів. Але на практиці зазвичай використовують одновимірні та двовимірні масиви.

Цикл **foreach** може обходити всі елементи двовимірного масиву, а використання інших циклів трохи ускладнюється через те, що поле **Length** містить в собі загальну довжину масиву.

Для отримання кількості рядків і стовпців, використовують метод ***GetUpperBound***, який повертає значення останнього індексу, для переданого в якості аргументу виміру. Оскільки як результат ми отримуємо індекс, то для отримання довжини, значення потрібно збільшити на одиницю:

* Кількість рядків – **GetUpperBound(0) + 1**;
* Кількість стовпців – **GetUpperBound(1) + 1**.

Розглянемо приклад заповнення та виводу двовимірного масиву:

var table = new int[3, 4];

//кількість рядків

var rowCount = table.GetUpperBound(0) + 1;

//кількість стовпців

var columnCount = table.GetUpperBound(1) + 1;

//заповнення таблиці

for (int i = 0; i < rowCount; i++)

for (int j = 0; j < columnCount; j++)

table[i, j] = i + j;

//вивід елементів на екран

Console.WriteLine("Результат роботи циклу foreach");

foreach (var element in table)

{

Console.Write("{0}\t", element);

}

Console.WriteLine("Вивід у вигляді таблиці");

var r = 0;

while (r < rowCount)

{

var c = 0;

while (c < columnCount)

{

Console.Write("{0}\t", table[r, c]);

c++;

}

r++;

Console.WriteLine();

}

**Основні поняття масивів**

До основних характеристик масивів належать:

* **Розмірність або ранг** – кількість індексів за допомогою яких звертаються до елемента(одновимірні, двовимірні, тривимірні…);
* **Розмір або довжина масиву** – загальна кількість елементів в масиві;
* **Розмірність кожного виміру** – довжина окремо взятого виміру масиву(для двовимірного - кількість рядків і стовпців).