**Масиви**

# *Поняття масиву*

Як відомо, будь-яку задачу людина, як і машина, розв’язує виконуючи кроки деякого алгоритму. Алгоритмом називають наперед задану скінчену послідов­ність чітких правил або команд (кроків), виконання яких призводить до одержання розв'язку задачі. Програма ж є одним із способів запису алгоритму.

У своїй роботі програміст виконує маніпуляції з величинами. З точки зору програмування, величини – це дані, що обробляються програмами. Носіями даних у програмах є константи, змінні(значення яких зберігається в оперативній пам'яті) та файли(які зберігаються на носіях інформації).

Константи– це величини, значення яких у процесі виконання програми не змінюється. Змінні– це величини, значення яких у процесі виконання програми можуть змінюватися. Імена констант та змінних, як і інших програм­них об'єктів, записують у формі ідентифікаторів (послідовності символів латинського алфавіти, цифр та символу нижнього підкреслювання “\_”, яка не може починатися з цифри). Кожна змінна і константа належать до визначеного типу. Тип даних **–** це сукупність властивостей певного набору даних, від яких залежать діапазон значень, якого можуть набувати ці дані, а також сукупність операцій, які можна виконувати над цими даними.

Усі типи даних розділяють на дві групи: скалярні (прості) та структуровані (складені). До скалярних відносять типи даних (цілі, дійсні, символьний, рядковий, логічний), які дозволяють опрацьовувати окремі числа, символи, рядки і т.д. Але у більшості задач доводиться розглядати сукупності величин одного типу (вектори, матриці, таблиці, числові послідовності і т.д.). Звичайно, можна було би для кожної величини ввести змінну. Але уявіть собі, що буде у випадку, коли кількість величин рівна 100. Тому для збереження сукупності елементів одного типу у мовах програмування визначається так званий структурований тип даних – *масив*.

Можна дати кілька означень масиву.

*Масив –*  це структура даних, що являє собою однорідну (за типом), фіксовану (за розміром і конфігурацією) сукупність нумерованих елементів.

*Масивом* називається скінчена послідовність елементів одного типу, які різняться порядковим номером.

*Масив* – формальне об’єднання декількох однотипних об’єктів (чисел, символів, рядків і т.д.), які розглядаються як єдине ціле.

Масиви відносяться до структур з прямим або довільнимдоступом. Щоб визначити окремий елемент масиву потрібно вказати його *індекс*. В одновимірному масиві (коли для ідентифікації елемента необхідно вказати значення одного індекса) індекс – це порядковий номер елементу масиву (нумерація починається з нуля). Якщо для ідентифікації елемента необхідно вказати декілька індексів, то такий масив називають багатовимірним. Кількість індексів при цьому називають *розмірністю,* кількість дозволених значень кожного індексу –  *діапазоном*, а сукупність розмірності та діапазону – *формою масиву.*

Класифікація масивів може здійснюватися за багатьма ознаками:

1. можливістю зміни значень елементів масиву (масиви-константи та масиви-змінні);
2. кількістю індексів, які необхідно вказати для того, щоб ідентифікувати елемент у масиві (масиви одновимірні, двовимірні, тривимірні і т.д.);

Базовим типом для всіх масивів є тип System.Array.

**Одновимірні масиви**

*Опис одновимірного масиву*

Одновимірний масив часто асоціюють із вектором. Загальний вигляд опису типу одновимірного масиву:

**<тип елем.>[ ]** <ім’я типу масиву>;

Ім’я типу масиву задається у формі ідентифікатора.

**Приклад**.

**byte [ ] b; //** Опис одновимірного масиву **b** типу **byte**

**double [ ] d; //** Опис одновимірного масиву **d** типу **double**

char [ ] r; // Опис одновимірного масиу r типуchar

*Виділення пам’яті*

Виділення пам’яті здійснюється з використанням оператора new

**<тип елем.>** [ ] <ім’я масиву> **= new <тип елем.> [<кількість елементів>];**

**Приклад**

byte [ ] **b = new byte [20]; //**Виділення пам’яті для 20 елементів типу **byte**

double [ ] **d = new double[50]; //**Виділення пам’яті для 50 елементів типу **double**

char [ ] r=new char[15]; **//**Виділення пам’яті для 15 елементів типу char

Зауважимо, що кількість елементів може бути не тільки константою чи літералом, а й значення деякої змінної

Приклад.

int n; // n - кількість елементів

Console.Write("n=");

n = int.Parse(Console.ReadLine()); // Введення кількості елементів з клавіатури

int [ ] a = new int[n]; // Виділення пам’яті для n елементів масиву a

*Звертання до елементів масиву*

Для ідентифікації елемента масиву необхідно вказати ім’я масиву та індекс елемента, який записується у квадратних дужках після імені масиву

<ім’я масиву> [ <індекс> ]

**Приклад.**

b [ 0 ] = 2;

d [ 4 ] = 2.4;

r [ 1 ] = ‘М’ ;

Наведемо декілька важливих зауважень:

1. елементи масиву можуть бути довільного, але тільки одного типу;
2. значення індексу не повинно виходити за межі описаного діапазону;
3. оскільки процедури введення/виведення розраховані на введення та виведення значень простих типів, то масив потрібно вводити/виводити поелементно, тобто кожен елемент окремо.

*Початкова ініціалізація. Ініціалізатори масивів*

При описанні масивів можуть використовуватися інціалізатори масивів (конструкції, що дозволяють здійснювати початкову ініціалізацію значень елементів масиву). При цьому розмірність масиву та кількість вказуваних констнт повинні співпадати.

**Приклад.**

byte [ ] b = new byte [ 4 ] {11, 25, 23, 14};

double [ ] d=new double [ 3 ] { 2.3, 2.5, 8 };

При ініціалізації перед ініціалізатором може бути опущений тип масиву, оскільки він повністю ввизначається типом поля або змінної. Так, наприклад, ініціалізатор

byte [ ] b = {11, 25, 23, 14};

є скороченою формою запису ініціалізатора

byte [ ] b = new byte [ ] {11, 25, 23, 14};

і є еквівалентним наступному фрагменту програми

byte [ ] b = new byte [ 4 ];

b[0]=11;

b[1]=25;

b[2]=23

b[3]=14;

*Представлення одновимірного масиву в пам’яті ЕОМ*

Для збереження масивів в пам’яті ЕОМ використовується послідовне представлення. Тобто для збереження масиву наперед виділяється ділянка оперативної пам’яті величиною

<**кількість елементів**> \* <**кількість байтів для збереження одного елемента**>

і елементи розміщуються послідовно один за одним.

**Приклад.**

byte [ ] b = new byte [ 5 ];

Змінна b в пам’яті ЕОМ буде розміщена наступним чином

|  |  |  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- | --- | --- |
|  |  | 150 |  |  |  |  |  | Адреси |
| 150 |  |  |  |  |  |  |  | Комірки |
| b | … | b[0] | b[1] | b[2] | b[3] | b[4] | … | **Імена комірок** |
|  |  | Область масиву b | | | | |  |  |

byte [ ] c;

c=b;

c[0]=89; b[0]=89

|  |  |  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- | --- | --- |
|  |  | 150 |  |  |  |  |  | Адреси |
| 150 | 150 | 89 |  |  |  |  |  | Комірки |
| b | c | b[0], c[0] | b[1],  c[1] | b[2]  c[2] | b[3] | b[4] | … | **Імена комірок** |
|  |  | Область масиву b | | | | |  |  |

*Введення елементів масиву*

Масив відноситься до структурованих типів даних, тому введення і виведення масивів повинно здійснюватися поелементно. Як правило, введення елементів масиву здійснюється у циклі. Загальний вигляд циклу введення елементів масиву у консольних програмах може бути таким:

for(<параметр>= 0; <параметр> < <кільк.елем.>; <параметр>++)

{

Console.Write (“<ім’я мас.>[ {0} ]= ”,<параметр>);

<ім’я мас.>[ <параметр> ] =<тип елементів>.Parse(Console.ReadLine());

}

**Приклад.**

**for (int i = 0; i < n; i++)**

**{**

**Console.Write("a[{0}]=",i);**

**a[i] = double.Parse(Console.ReadLine());**

**}**

*Виведення елементів масиву*

Як і введення, виведення елементів масиву також здійснюється поелементно. Загальний вигляд циклу виведення елементів масиву може виглядати наступним чином:

for(<параметр>= 0; <параметр> < <кільк.елем.>; <параметр>++)

{

Console.WriteLine(“<ім’я мас.>[ {0} ]={1} ”,<параметр>,<ім’я мас.>[ <параметр>]);

}

**Приклад.**

**for (int i = 0; i < n; i++)**

**{**

**Console.WriteLine("a[{0}]={1}",i, a[i]);**

**}**

*Приклади задач*

Наведемо декілька прикладів задач, розв’язаних з використанням масивів.

**Приклад.** Дано , знайти .

***Розв’язання***

Для збереження векторів використаємо однойменні одновимірні масиви , і . Кількість елементів  будемо вводити з клавіатури.

static void Main(string[] args)

{

int n;

Console.Write("n="); //Введення кількості елементів n

n = int.Parse(Console.ReadLine());

double[] a = new double[n]; //Виділення пам’яті для масиву a

for (int i = 0; i < n; i++) //Введення елементів масиву a

{

Console.Write("a[{0}]=",i);

a[i] = double.Parse(Console.ReadLine());

}

double[] b = new double[n]; //Виділення пам’яті для масиву b

for (int i = 0; i < n; i++) //Введення елементів масиву b

{

Console.Write("b[{0}]=", i);

b[i] = double.Parse(Console.ReadLine());

}

double[] c = new double[n]; //Виділення пам’яті для масиву c

for (int i = 0; i < n; i++) //Знаходження елементів масиву c

{

c[i] = a[i] + b[i];

}

for (int i = 0; i < n; i++) //Виведення елементів масиву с

{

Console.WriteLine("c[{0}]={1}", i,c[i]);

}

Console.ReadLine();

}

Результати роботи програми.

n=3

a[0]=1

a[1]=2

a[2]=3

b[0]=4

b[1]=5

b[2]=6

c[0]=5

c[1]=7

c[2]=9

**Приклад**. Дано одновимірний масив з  елементів. Знайти суму елементів масиву.

static void Main(string[] args)

{

int n;

Console.Write("n="); //Введення кількості елементів n

n = int.Parse(Console.ReadLine());

double[] a = new double[n]; //Виділення пам’яті для масиву a

for (int i = 0; i < n; i++) //Введення елементів масиву a

{

Console.Write("a[{0}]=",i);

a[i] = double.Parse(Console.ReadLine());

}

double Sum = 0;

for (int i = 0; i < n; i++) //Знаходження суми елементів масиву a

{

Sum+= a[i];

}

//Виведення суми на екран

Console.WriteLine("Sum={0}",Sum);

Console.ReadLine();

}

*Функції для роботи з одновимірними масивами*

Як було зазначено раніше, базовим типом для всіх масивів є тип System.Array, який містить ряд функцій, які дозволяють здійснювати деякі маніпуляції над масивами.

Наприклад,

<ім’я масиву>.Sum() – дозволяє знайти суму елементів масиву,

<ім’я масиву>.Max() – дозволяє знайти максимальний елмент,

<ім’я масиву>.Min() – дозволяє знайти мінімальний елемент,

<ім’я масиву>.Average() – дозволяє знайти середнє значення.

*Копіювання масивів*

Оскільки масив є типом посилання, то для коректного копіювання масивів використовують спеціальний метод CopyTo

<масив 1>.CopyTo (<масив 2>, <поч. індекс>) – дозволяє скопіювати елементи <масиву 1> у <масив 2> починаючи з вказаного початкового індекса.

Приклад.

a.CopyTo(b,1);

В результаті елементи масиву a буде скопійовано у масив b починаючи з 1-го елемента (тобто буде виконано послідовність операцій присвоєння b[1]=a[0], b[2]=a[1], … ).

# Багатовимірні масиви

Багатовимірними масивами є масиви, в яких для ідентифікації елемента необхідно вказати більше одного індексу. Найбільш використовуваними є двовимірні масиви, тому їх буде детально розглянуто. З багатовимірними масивами маніпуляції виконуються аналогічно.

## Прямокутні масиви

Розглянемо прямокутні двовимірні масиви (у кожному рядку однакова кількість елементів). Оскільки двовимірні масиви часто асоціюються з двовимірними таблицями і матрицями, то прийнято вважати, що перший індекс позначає номер рядка, а другий – номер стовпця.

*Опис двовимірного масиву*

<тип елементів> [ , ] <ім’я масиву>;

**Приклад.**

int [ , ] A;

double [ , ] d;

*Виділення пам’яті*

Як і при роботі з одновимірними масивами, виділення пам’яті здійснюється з використанням оператора new

<тип елементів> [ , ] <ім’я масиву>=new <тип елементів>[кільк. рядків, кількі. стовпців];

Приклад.

int [ , ] A= new int[ 30,20];

double [ , ] d = new double [5 , 3];

Змінну A можна відобразити у вигляді матриці



Як і при роботі з одновимірними масивами, кількість рядків і кількість стовпців можуть бути змінними величинами

int n; //Введення кількості рядків 

Console.Write("n=");

n = int.Parse(Console.ReadLine());

int m; //Введення кількості стовпців 

Console.Write("m=");

m = int.Parse(Console.ReadLine());

int[,] A = new int[n, m]; //Виділення пам’яті

*Звертання до елементів масиву*

<ім’я масиву> [ <індекс1> , <індекс2> ]

або, що те ж саме,

<ім’я масиву> [ <номер рядка > , <номер стовпця > ]

Приклад.

А[2,7]=21;

d[1,5]=34;

*Ініціалізатори масивів*

Як і при роботі з одновимірними масивами при використанні двовимірних маисів початкова ініціалізація може бути виконана з використанням ініціалізаторів (у фігурних дужках поступово вказують елементи кожного з рядків, як одновимірних масивів).

Прилад.

int [ , ] b = new int [2,3] {{0,1,2}, {2,3,4}};

*Введення*

Введення двовимірних масивів здійснюється поелементно, як правило, з використанням двох циклів (вкладених). Наведемо приклад такого введення

int n; //Введення кількості рядків 

Console.Write("n=");

n = int.Parse(Console.ReadLine());

int m; //Введення кількості стовпців 

Console.Write("m=");

m = int.Parse(Console.ReadLine());

int[,] A = new int[n, m]; //Виділення пам’яті

for (int i = 0; i < n; i++) // Введення масиву з клавіатури

{

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

{

Console.Write("A[{0},{1}]=", i, j);

A[i, j] = int.Parse(Console.ReadLine());

}

}

*Виведення елементів масиву*

Виведення елементів також здійснюється поелементно. Наведемо приклад такого виведення

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

{

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

{

Console.WriteLine("A[{0},{1}]={2}", i, j,A[i, j]);

}

}

*Приклади задач*

**Приклад.** Дано матрицю . Необхідно помножити кожен елемент матриці на її найбільший елемент.

*Розв’язання*

static void Main(string[] args)

{

int n;

Console.Write("n="); //Введення кількості рядків 

n = int.Parse(Console.ReadLine());

int m;

Console.Write("m="); //Введення кількості стовпців 

m = int.Parse(Console.ReadLine());

int[,] A = new int[n, m]; //Виділення пам’яті для масиву 

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

{

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

{

Console.Write("A[{0},{1}]=", i, j);

A[i, j] = int.Parse(Console.ReadLine());

}

}

//Знаходження максимального 

int max = A[0, 0];

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

{

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

{

if (max < A[i, j])

{

max = A[i, j];

}

}

}

//Множення елементів на масиву на максимальний

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

{

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

{

A[i, j] \*= max;

}

}

//Виведення елементів масиву на екран

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

{

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

{

Console.WriteLine("A[{0},{1}]={2}", i, j, A[i, j]);

}

}

Console.ReadLine();

}

**Результати роботи програми**

n=2

m=3

A[0,0]=1

A[0,1]=2

A[0,2]=3

A[1,0]=4

A[1,1]=5

A[1,2]=6

A[0,0]=6

A[0,1]=12

A[0,2]=18

A[1,0]=24

A[1,1]=30

A[1,2]=36

***Багатовимірні масиви з різною кількістю елементів у рядку***

Раніше було зазначено, що елементи масиву можуть буди довільного типу, в тому числі і масивами. У випадку, якщо маємо одновимірний масив, елементами якого є також одновимірні масиви то маємо двовимірний масив. При цьому кількість елементів у кожному з масивів може бути різною.

*Опис*

<тип елементів> [ ] [ ] <ім’я масиву>;

**Приклад.**

**double [ ] [ ] d;**

*Виділення пам’яті*

Виділення пам’яті для таких масивів відбувається у два етапи: на першому етапі виділяється пам’ять для масиву посилань на рядки, на другому – виділяється пам’ять під безпосередньо елементи рядків)

Приклад 1.

**double [ ] [ ] d = new double[3] []; //Виділення пам’яті для посилань на рядки**

d[0]=new double [3]; **//Виділення пам’яті для елементів рядків**

d[1]=new double [7];

d[2]=new double [5];

Як бачимо, кількість елементів у кожному з рядків різна.

Приклад 2.

**double [ ] [ ] d = new double[50] []; //Виділення пам’яті для посилань на рядки**

**for (int i=0; i<50; i++)**

**d[ i ] = new** double**[ i+1 ];**

*Звертання до елементів*

<ім’я масиву> [ <індекс1> ][ <індекс2> ]

або, що те ж саме,

<ім’я масиву> [ <номер рядка > ] [ <номер стовпця > ]

Приклад.

d[1][2]=34;

Зауважимо, що використання ініціалізаторів можливе тільки для прямокутних масивів.

**Приклад.**  Знайти найстаршого учня у кожному з класів (кількість учнів у кожному із класів може бути різною.

Розв’язання

Для збереження віку учнів використаємо двовимірний масив  з різною кількістю елементів у рядку.

using System;

using System.Collections.Generic;

using System.Linq;

using System.Text;

namespace ConsoleApplication1

{

class Program

{

static void Main(string[] args)

{

int[][] a;

int n;

Console.Write("Number of classes = ");

n = int.Parse(Console.ReadLine());

a = new int[n][];

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

{

int m;

Console.Write("Number of pupils in class {0} = ", i);

m = int.Parse(Console.ReadLine());

a[i] = new int[m];

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

{

Console.Write("Age: class {0} pupil {1} =", i,j);

a[i][j] = int.Parse(Console.ReadLine());

}

}

Console.WriteLine("Age of seniors: ");

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

{

for (int j = 0; j < a[i].Length; j++)

{

Console.Write("-- "+a[i][j]);

}

Console.WriteLine("--== max A[{0}]={1} ",i,a[i].Max());

}

Console.ReadLine();

}

}

}