



LẬP TRÌNH C# 1

BÀI 3: MẢNG DỮ LIỆU

- ⊙ Mạng một chiều
- ⊙ Mạng nhiều chiều



Phần I: Mảng 1 chiều

 Mảng 1 chiều

 Các thao tác mảng 1 chiều

 ArrayList

Phần II: Mảng nhiều chiều

 Mảng Rectangular Array

 Mảng Jagged Array



- ❑ Mảng là cấu trúc lưu trữ nhiều phần tử có cùng kiểu dữ liệu

0	1	2	3	4	5	6	7	8	← Indices
5	7	9	1	45	1	9	9	2	← Elements

- ❑ Để truy xuất các phần tử cần biết chỉ số (index).
- ❑ Trong C# chỉ số phần tử là các số nguyên không âm và bắt đầu từ 0 1 2 3...
- ❑ Các phần tử trong mảng dùng chung một tên.

❑ Các thao tác mảng

- ❖ Khai báo
- ❖ Truy xuất (đọc/ghi) phần tử
- ❖ Lấy số phần tử
- ❖ Duyệt mảng
- ❖ Tìm kiếm phần tử
- ❖ Chèn phần tử
- ❖ Sắp xếp các phần tử mảng

❑ Cú Pháp

```
// Cách 1
```

```
datatype[] arrayName; // Khai báo tên mảng và kiểu dữ liệu  
arrayName = new datatype[length]; // Tạo mảng
```

```
// Cách 2
```

```
datatype[] arrayName = new datatype[length];
```

❑ Khai báo không khởi tạo

- ❖ `int[] intArray = new int[6];` // mảng số nguyên chưa biết số phần tử
- ❖ `string[] c = new string[5];` // mảng chứa 5 chuỗi

❑ Khai báo có khởi tạo

- ❖ `double[] d1 = new double[]{2, 3, 4, 5, 6};` // mảng số thực, 5 phần tử, đã được khởi tạo
- ❖ `double[] d2 = {2, 3, 4, 5, 6};` // mảng số thực, 5 phần tử, đã được khởi tạo

❑ Sử dụng từ khóa Var

❖ Khởi tạo mảng có kiểu tường minh

```
var a = new int[] {2, 4, 5, 8, 10};
```

❖ Khởi tạo mảng kiểu ngầm định

- Kiểu dữ liệu của mảng được hiểu ngầm định dựa vào giá trị các phần tử

```
var a = new[] {1, 10, 100, 1000}; // int[]
```

```
var b = new[] {1, 1.5, 2, 2.5}; // double[]
```

```
var c = new[] {"hello", "Fpoly", "Phan Viet The"}; // String[]
```

❑ Sử dụng từ khóa Var

❖ Khởi tạo mảng kiểu ngầm định

- Giá trị của các phần tử phải có cùng khả năng chuyển kiểu được
- Ví dụ: khai báo sau không thể chuyển kiểu ngầm định

```
var d = new[] {1, "Fpoly", 2, "laptrinhcsharp"}; // giá trị các phần tử không cùng kiểu dữ liệu.
```

```
var e = new[] {3,2, null, 1}; // int là kiểu trị, không được.
```

```
var f = new[] {3,2, true, 1}; // true không thể chuyển kiểu.
```


❑ Sử dụng chỉ số (**index**) để phân biệt các phần tử.

Chỉ số mảng tính từ 0.

❖ `Int[] a = {4, 3, 5, 7};`

❖ `a[2] = a[1] * 4; // 3*4=12`

❖ Sau phép gán này mảng là {4, 3, 12, 7};

❑ Sử dụng thuộc tính **length** để lấy số phần tử của mảng

```
static void Main(string[] args)
{
    int[] IntArray = { 3, 9, 10 };
    //Chiều dài của mảng là 3
    Console.WriteLine(IntArray.Length);
}
```

- ❑ 2 vòng lặp thường được sử dụng để duyệt mảng là for và foreach.

```
int[] a = {4, 3, 5, 9};  
for(int i=0; i<a.Length; i++){  
    Console.WriteLine(a[i]);  
}
```

← **for(;;)**

foreach →

```
int[] a = {4, 3, 5, 9};  
foreach (int x in a){  
    Console.WriteLine (x);  
}
```

❑ Ví dụ sau tính tổng các số chẵn của mảng.

- ❖ Lấy từng phần tử từ mảng với foreach
- ❖ Nếu là số chẵn thì cộng vào tổng

```
static void Main(string[] args)
{
    int tong = 0;
    int[] a = { 4, 3, 5, 9, 65, 23, 42 };
    foreach (int x in a)
    {
        if (x % 2 == 0)
        {
            tong = tong + x;
        }
    }
    Console.WriteLine(tong);
}
```



DEMO

Nhập mảng số nguyên

+ Tính và xuất trung bình cộng

+ Xuất lập phương các phần tử



- ❑ C# hỗ trợ nhiều phương thức và thuộc tính giúp lập trình nhanh hơn

```
static void Main(string[] args)
{
    string[] names = {"Jane", "Frank", "Alice", "Tom" };

    Array.Sort(names);

    foreach(string el in names)
    {
        Console.Write(el + " ");
    }

    Console.Write('\n');

    Array.Reverse(names);

    foreach(string el in names)
    {
        Console.Write(el + " ");
    }

    Console.Write('\n');
}
```

- ❑ C# hỗ trợ nhiều phương thức và thuộc tính giúp lập trình nhanh hơn

Phương thức	Mô tả
SetValue(value, position)	Thiết lập giá trị cho phần tử dựa vào vị trí (position)
GetValue(position)	Lấy giá trị của phần tử
IndexOf(Array, element)	Tìm kiếm một phần tử (element)
GetLength()	Cho biết số lượng phần tử trong mảng
Reverse(Array)	Đảo ngược thứ tự các phần tử
Sort(Array)	Sắp xếp các phần tử của mảng

- ❑ Arrays.sort(mảng) không thể thực hiện
 - ❖ Sắp xếp giảm
 - ❖ Các kiểu không so sánh được
- ❑ Giải pháp: tự xây dựng thuật toán sắp xếp

```
int a[] = {8,2,6,2,9,1,5};  
for(int i=0; i<a.length-1; i++){  
    for(int j=i+1; j<a.length; j++){  
        if(a[i] > a[j]){  
            int temp = a[i];  
            a[i] = a[j];  
            a[j] = temp;  
        }  
    }  
}
```

Nếu thay đổi toán tử so sánh thành $<$ thì thuật toán trở thành sắp xếp tăng dần.



DEMO

Nhập mảng 5 SV và xuất tăng
dần theo alphabet





LẬP TRÌNH C# 1

BÀI 3: MẢNG DỮ LIỆU (P2)

- ❑ Là một Collections giúp lưu trữ và quản lý một danh sách các đối tượng theo kiểu mảng (truy cập các phần tử bên trong thông qua chỉ số index)
- ❑ Cho phép thêm hoặc xóa các phần tử một cách linh hoạt và có thể tự điều chỉnh kích cỡ một cách tự động.
- ❑ Để sử dụng các Collections trong **.NET** ta cần thêm thư viện System.Collections
- ❑ Sử dụng từ khóa new để tạo ArrayList

- ❑ Dùng phương thức "add" thêm phần tử vào mảng và truy xuất giá trị phần tử thông qua index

Adding elements
to the array list

```
static void Main(string[] args)
{
    ArrayList a1 = new ArrayList();
    a1.Add(1);
    a1.Add("Example");
    a1.Add(true);
    Console.WriteLine(a1[0]);
    Console.WriteLine(a1[1]);
    Console.WriteLine(a1[2]);
    Console.ReadKey();
}
```

1

Defining an array list

2

3

Displaying the elements
of the array list

❑ Một số phương thức thông dụng trong ArrayList:

TÊN PHƯƠNG THỨC	Ý NGHĨA
Add(object Value)	Thêm đối tượng Value vào cuối ArrayList .
AddRange(ICollection ListObject)	Thêm danh sách phần tử ListObject vào cuối ArrayList .
BinarySearch(object Value)	<p>Tìm kiếm đối tượng Value trong ArrayList theo thuật toán tìm kiếm nhị phân.</p> <p>Nếu tìm thấy sẽ trả về vị trí của phần tử ngược lại trả về giá trị âm.</p> <p>Lưu ý là ArrayList phải được sắp xếp trước khi sử dụng hàm.</p>

❑ Một số phương thức thông dụng trong ArrayList:

Clear()	Xoá tất cả các phần tử trong ArrayList .
Clone()	Tạo 1 bản sao từ ArrayList hiện tại.
Contains(object Value)	Kiểm tra đối tượng Value có tồn tại trong ArrayList hay không.
GetRange(int StartIndex, int EndIndex)	Trả về 1 ArrayList bao gồm các phần tử từ vị trí StartIndex đến EndIndex trong ArrayList ban đầu.
IndexOf(object Value)	Trả về vị trí đầu tiên xuất hiện đối tượng Value trong ArrayList . Nếu không tìm thấy sẽ trả về -1.

❑ Ví dụ dùng Count, Contains, RemoveAt:

```
static void Main(string[] args)
{
    ArrayList a1 = new ArrayList();

    a1.Add(1);
    a1.Add("Example");
    a1.Add(true);

    Console.WriteLine(a1.Count);

    Console.WriteLine(a1.Contains(2));

    Console.WriteLine(a1[1]);
    a1.RemoveAt(1);
    Console.WriteLine(a1[1]);
}
```

Removing an element and showing that the element has been removed

3

Count of items in the Array list

1

Checking to see if the Array List contains the element

2



DEMO

1. Sử dụng arraylist tạo mảng số nguyên và thực hiện đếm có bao nhiêu số chẵn, xóa các Số chẵn trong arraylist.



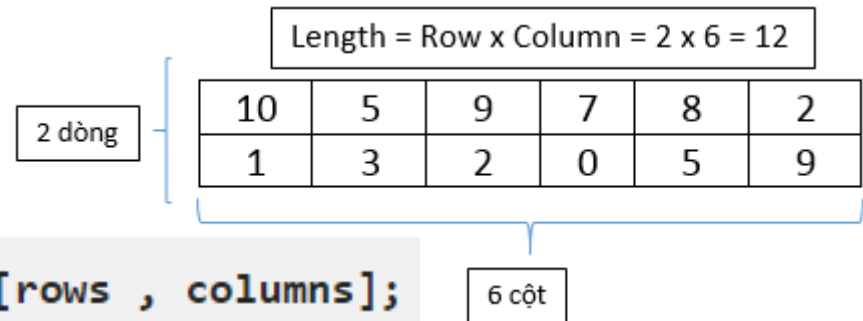
- ❑ Mảng đa chiều cho phép chúng ta lưu trữ dữ liệu trên nhiều dòng
- ❑ Kích thước của mảng được xác định dựa vào số dòng và số cột

	Column 0	Column 1	Column 2	Column 3
Row 0	<code>a[0][0]</code>	<code>a[0][1]</code>	<code>a[0][2]</code>	<code>a[0][3]</code>
Row 1	<code>a[1][0]</code>	<code>a[1][1]</code>	<code>a[1][2]</code>	<code>a[1][3]</code>
Row 2	<code>a[2][0]</code>	<code>a[2][1]</code>	<code>a[2][2]</code>	<code>a[2][3]</code>

- ❑ Có 2 loại cơ bản là : Rectangular Array và Jagged Array

❑ Rectangular Array: đặc trưng là mảng 2 chiều có có m dòng và n cột

❑ Cú pháp:



```
datatype[,] arrayName = new datatype[rows , columns];
```

```
namespace TwoDimensions
{
    class Program
    {
        static void Main(string[] args)
        {
            int[,] twodim = new int[,] { {1, 2, 3}, {1, 2, 3} };

            int d1 = twodim.GetLength(0);
            int d2 = twodim.GetLength(1);

            for (int i=0; i<d1; i++)
            {
                for (int j=0; j<d2; j++)
                {
                    Console.WriteLine(twodim[i, j]);
                }
            }
        }
    }
}
```

- ❑ Jagged Array: tương tự Rectangular Array ngoại trừ số cột trên mỗi dòng có thể khác nhau

10	5	9	7	8	2
1	3	2	0	5	9
3	1	2			

- ❑ Cú pháp:

```
data_type[][] name_of_array = new data_type[rows][]
```

❑ Jagged Array:

Output:

```
Row(0): 1 2 3 4
Row(1): 11 34 67
Row(2): 89 23
Row(3): 0 45 78 53 99
```

```
// Main Method
public static void Main()
{

    // Declare the Jagged Array of four elements:
    int[][] jagged_arr = new int[4][];

    // Initialize the elements
    jagged_arr[0] = new int[] {1, 2, 3, 4};
    jagged_arr[1] = new int[] {11, 34, 67};
    jagged_arr[2] = new int[] {89, 23};
    jagged_arr[3] = new int[] {0, 45, 78, 53, 99};

    // Display the array elements:
    for (int n = 0; n < jagged_arr.Length; n++) {

        // Print the row number
        System.Console.Write("Row({0}): ", n);

        for (int k = 0; k < jagged_arr[n].Length; k++) {

            // Print the elements in the row
            System.Console.Write("{0} ", jagged_arr[n][k]);
        }
        System.Console.WriteLine();
    }
}
```




DEMO

Demo sử dụng mảng nhiều Jagged Array
Lưu danh sách các ngành học: Cntt, kinh tế,
QTKD. Mỗi ngành có số lượng và tên môn học
Khác nhau (số lượng, tên tùy ý)



Tổng kết bài học

Phần I: Mảng 1 chiều

 Mảng 1 chiều

 Các thao tác mảng 1 chiều

 ArrayList

Phần II: Mảng nhiều chiều

 Mảng Rectangular Array

 Mảng Jagged Array





KẾT THÚC