

# LESSON 05

## I. VÒNG LẶP **WHILE** TRONG C#

### ❖ Định nghĩa

Là một trong những cú pháp vòng lặp đã đề cập trong Lesson04 và cũng rất quan trọng trong lập trình C#.

#### ➤ Cú pháp:

```
while (<Điều kiện lặp>)  
{  
    // khối lệnh lặp lại  
}
```

- **Điều kiện lặp** là một biểu thức logic bắt buộc phải có với kết quả trả về bắt buộc là **true** hoặc **false**.
- Từ khóa **while** biểu thị đây là một vòng lặp **while**. Các câu lệnh trong khối lệnh sẽ được lặp lại đến khi không còn thỏa mãn **điều kiện lặp** sẽ kết thúc vòng lặp **while**

#### ➤ Tiến trình:

- Đầu tiên trình biên dịch sẽ đi vào dòng **while (<Điều kiện lặp>)**. Kiểm tra điều kiện lặp có thỏa mãn hay không. Nếu kết quả là **true** thì sẽ đi vào bên trong thực hiện khối code. Khi gặp ký tự **}** sẽ quay lên kiểm tra điều kiện lặp và tiếp tục thực hiện khối code. Quá trình chỉ kết thúc khi điều kiện lặp là **false**.
- Điều kiện lặp luôn bằng **true** thì vòng lặp **while** sẽ trở thành vòng lặp vô tận
- Điều kiện lặp luôn bằng **false** thì vòng lặp sẽ không được thực thi

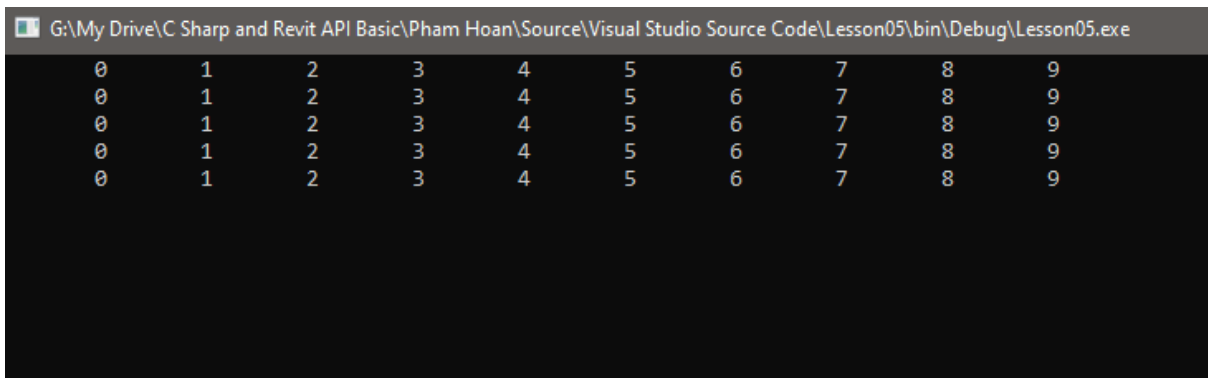
### ➤ Ví dụ: In ra một ma trận số

```
0 references
static void Main(string[] args)
{
    int countLoop = 0;
    int countLoopTime = 0;

    int valueNum = 10;
    int loopTime = 5;

    // Vẽ từ trên xuống LoopTime lần
    while (countLoopTime < loopTime)
    {
        countLoop = 0;
        // vẽ từ trái qua valueNum lần
        while (countLoop < valueNum)
        {
            Console.Write("{0,8}", countLoop);
            countLoop++;
        }
        // Mỗi khi hoàn thành một vòng lặp nhỏ thì lại xuống dòng chuẩn vị vẽ lần tiếp theo
        Console.WriteLine();
        countLoopTime++;
    }
    Console.ReadKey();
}
```

Khi chạy chương trình trên ta sẽ nhận được:




```
G:\My Drive\C Sharp and Revit API Basic\Pham Hoan\Source\Visual Studio Source Code\Lesson05\bin\Debug\Lesson05.exe
0      1      2      3      4      5      6      7      8      9
0      1      2      3      4      5      6      7      8      9
0      1      2      3      4      5      6      7      8      9
0      1      2      3      4      5      6      7      8      9
0      1      2      3      4      5      6      7      8      9
```

- Chúng ta cũng có thể in ra ma trận số với giá trị ngẫu nhiên bằng cách sử dụng lớp [Random](#).

- **Ví dụ:** Chúng ta cũng có thể vẽ một hình chữ nhật NxM (Ngoài chữ @, trong dấu .) với vòng lặp

**while.** (Yêu cầu học viên làm bài này)



## II. VÒNG LẶP **DO WHILE** TRONG C#

- **Cú pháp:**

```
do
{
    // khối lệnh lặp lại
}
while (<Điều kiện lặp>);
```

- **Điều kiện lặp** là một biểu thức logic bắt buộc phải có với kết quả trả về bắt buộc là **true** hoặc **false**.
- Từ khóa **do while** biểu thị đây là một vòng lặp **do while**. Các câu lệnh trong khối lệnh sẽ được lặp lại đến khi không còn thỏa mãn **điều kiện lặp** sẽ kết thúc vòng lặp **do while**.

### ➤ Tiến trình:

- Đầu tiên trình biên dịch sẽ đi vào dòng **do** và thực hiện khối lệnh bên trong. Sau đó khi gặp ký tự **}** sẽ kiểm tra điều kiện lặp có thỏa mãn hay không. Nếu kết quả là **true** thì sẽ quay lại ký tự **{** thực hiện khối code. Quá trình chỉ kết thúc khi điều kiện lặp là **false**.

- Điều kiện lặp luôn bằng **true** thì vòng lặp **while** sẽ trở thành vòng lặp vô tận.
- Điều kiện lặp luôn bằng **false** thì vòng lặp sẽ không được thực thi

### ➤ Lưu ý:

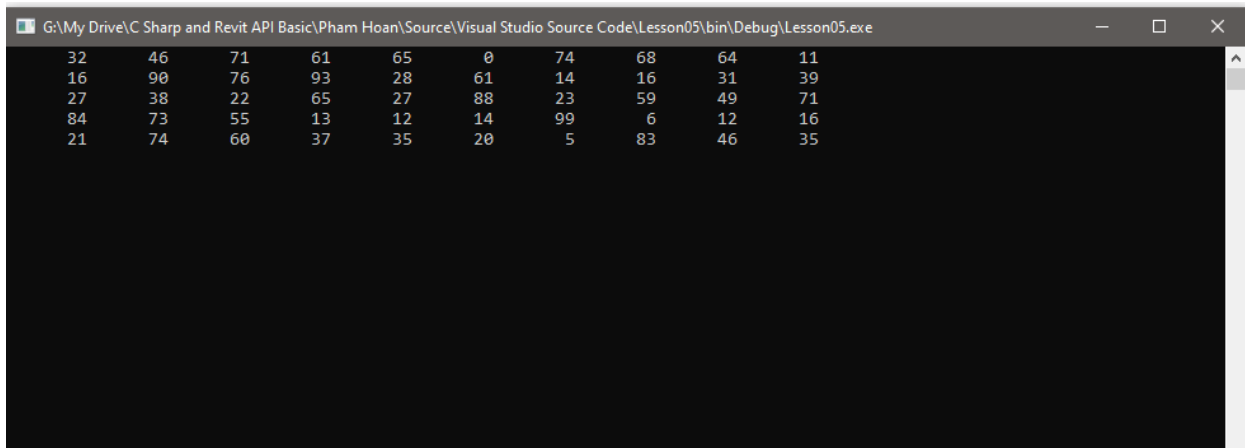
- vòng lặp **do while** sẽ thực hiện câu lệnh trong khối code xong rồi mới kiểm tra điều kiện lặp. Cuối vòng lặp **do while** có dấu **;** ở cuối.

### ➤ Ví dụ 01:

```
0 references
static void Main(string[] arg)
{
    int countLoop = 0;
    int countLoopTime = 0;
    int valueNum = 10;
    int loopTime = 5;
    int minRandomValue = 0;
    int maxRandomValue = 100;

    Random rand = new Random();
    // vẽ từ trên xuống LoopTime lần
    do
    {
        countLoop = 0;
        // vẽ từ trái qua valueNum lần
        do
        {
            int showValue = rand.Next(minRandomValue, maxRandomValue);
            Console.Write("{0,8}", showValue);
            countLoop++;
        }
        while (countLoop < valueNum);
        Console.WriteLine();
        countLoopTime++;
    }
    while (countLoopTime < loopTime);
    Console.ReadKey();
}
```

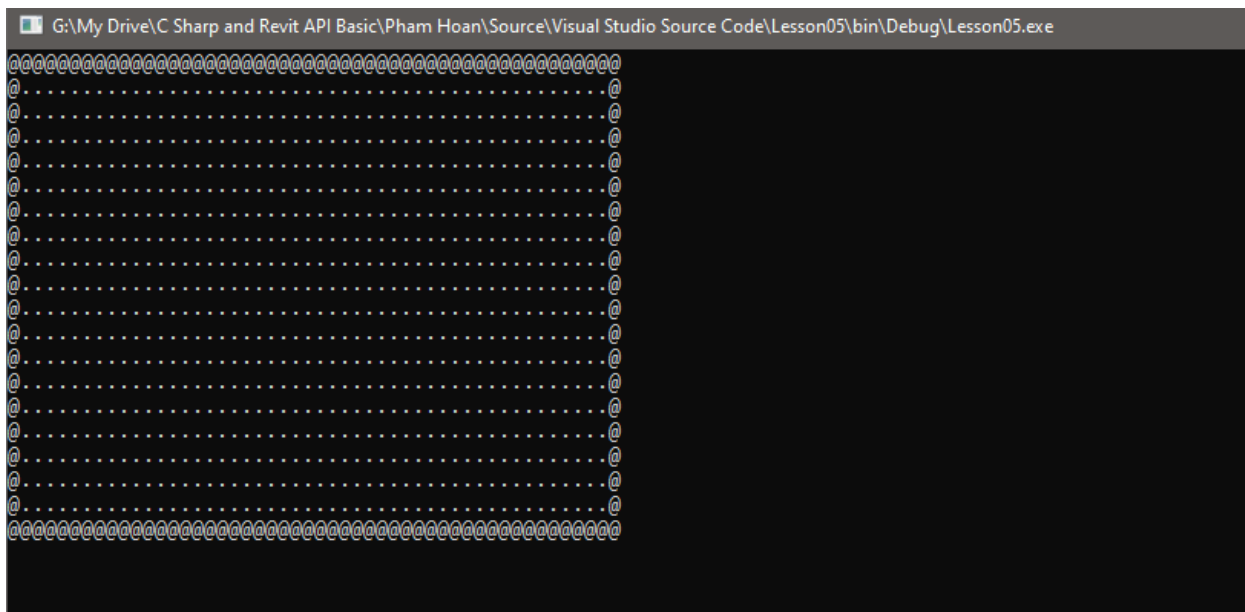
Khi chạy chương trình trên ta sẽ nhận được:



```
G:\My Drive\C Sharp and Revit API Basic\Pham Hoan\Source\Visual Studio Source Code\Lesson05\bin\Debug\Lesson05.exe
32 46 71 61 65 0 74 68 64 11
16 90 76 93 28 61 14 16 31 39
27 38 22 65 27 88 23 59 49 71
84 73 55 13 12 14 99 6 12 16
21 74 60 37 35 20 5 83 46 35
```

- **Ví dụ 02:** Chúng ta cũng có thể vẽ một hình chữ nhật NxM (Ngoài chữ @, trong dấu .) với vòng lặp

**while.** (Yêu cầu học viên làm bài này)



```
G:\My Drive\C Sharp and Revit API Basic\Pham Hoan\Source\Visual Studio Source Code\Lesson05\bin\Debug\Lesson05.exe
@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@
@. . . . . @
@. . . . . @
@. . . . . @
@. . . . . @
@. . . . . @
@. . . . . @
@. . . . . @
@. . . . . @
@. . . . . @
@. . . . . @
@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@
```

## BÀI TẬP

1. Ví dụ nếu bạn nhập số hàng là 5 thì vẽ tam giác sao có dạng:

```
*
**
***
****
*****
```

2. Kiểm tra số hoàn hảo: (VD: 6 có các ước số ngoại trừ chính nó là 1, 2, 3 và có tổng các ước là  $1+2+3=6 \rightarrow 6$  là số hoàn hảo)

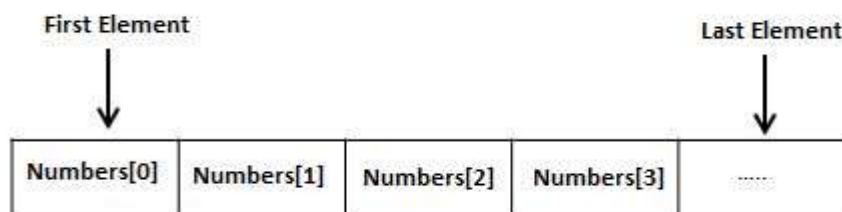
### III. MẢNG 1 CHIỀU TRONG C#

#### ❖ Khái niệm

- Tập hợp các đối tượng có cùng kiểu dữ liệu.
- Mỗi đối tượng trong mảng được gọi là một phần tử
- Các phần tử phân biệt với nhau bằng chỉ số phần tử. Trong C# chỉ số phần tử là các số nguyên không âm và bắt đầu từ 0 1 2 3...

#### ❖ Đặc điểm

- Các phần tử trong mảng dùng chung một tên và được truy xuất thông qua chỉ số phần tử.
- Một mảng cần có giới hạn số phần tử mà mảng có thể chứa.
- Phải cấp phát vùng nhớ mới có thể sử dụng mảng
- Vị trí ô nhớ của các phần tử trong mảng được cấp phát liên kế nhau



#### ➤ Cú pháp:

**<kiểu dữ liệu> [] <tên mảng>;**

- <kiểu dữ liệu> là kiểu dữ liệu của các phần tử trong mảng.
- Cặp dấu [] là ký hiệu cho khai báo mảng 1 chiều.
- <tên mảng> là tên của mảng, cách đặt tên mảng cũng như cách đặt tên biến

Để sử dụng được mảng ta phải khởi tạo giá trị hoặc cấp phát vùng nhớ cho mảng.

Cấp phát vùng nhớ:

- Được thực hiện thông qua toán tử new

- Lưu ý là khi cấp phát vùng nhớ cho mảng 1 chiều ta cần chỉ ra số phần tử tối đa của mảng.

```
0 references
static void Main7()
{
    /*
     * Khai báo mảng 1 chiều kiểu string và có tên là TTDesign
     * Sau đó thực hiện cấp phát vùng nhớ với số phần tử tối đa của mảng là 3.
     */
    string[] TTDesign = new string[3];
}
```

- Sau khi mảng được cấp phát vùng nhớ thì các phần tử trong mảng sẽ mang giá trị mặc định:

- Đối với số nguyên là 0
- Đối với số thực là 0.0
- Đối với kiểu ký tự là ' ' (ký tự rỗng)
- Đối với kiểu tham chiếu là **null**

- Chúng ta có thể khởi tạo giá trị khác mà chúng ta mong muốn ngay khi cấp phát vùng nhớ bằng cú pháp sau:

```
<KDL>[] <tên mảng> = new <KDL>[] { <giá trị 1>, ..., <giá trị n> };
```

- Các giá trị khởi tạo nằm trong cặp dấu ngoặc nhọn { } và cách nhau bởi dấu phẩy.
- Chúng ta không cần cung cấp số phần tử tối đa mà trình biên dịch sẽ tự đếm xem bạn đã khởi tạo bao nhiêu giá trị và xem nó như số phần tử tối đa. Vì thế dù việc khai báo số phần tử tối đa không lỗi nhưng trong trường hợp này nó không có ý nghĩa lắm!

#### ➤ Khởi tạo giá trị:

```
<kiểu dữ liệu>[] <tên mảng> = { <giá trị 1>, ..., <giá trị n> };
```

**Ví dụ:**

```
int[] IntArray = { 3, 9, 10 };
```

➤ **Tóm lại ta có 3 cách khai báo và khởi tạo sau:**

```
//CÁCH 1: Khai báo và cấp phát vùng nhớ
string[] Array = new string[3];

//CÁCH 2: Khai báo, cấp phát và khởi tạo giá trị cho mảng
string[] TTD = new string[] { "TT Design", "This is our company!" };

//CÁCH 3: Khởi tạo giá trị cho mảng
int[] IntArray = { 3, 9, 10 };
```

➤ **Sử dụng mảng:**

Các phần tử của mảng được truy xuất thông qua chỉ số phần tử và cặp dấu []. Có thể xem các phần tử của mảng như là các biến đơn và thao tác như thao tác với biến bình thường:

- Thêm phần tử trong mảng hoặc Sửa 1 phần tử trong mảng
- Không hỗ trợ xóa 1 phần tử trong mảng do chiều dài mảng là cố định ngay từ lúc khởi tạo và được lưu trữ trong bộ nhớ và không thể thay đổi được.
- Một số thuộc tính và phương thức đặc trưng của mảng 1 chiều:

Member	Nội dung
Length	Thuộc tính cho biết số lượng phần tử trong mảng
Rank	Thuộc tính cho biết số chiều mảng
Clone()	Copy (nhân bản) đối tượng mảng
GetValue(index)	Lấy giá trị phần tử trong mảng
Min()	Trả về giá trị nhỏ nhất trong mảng
Max()	Trả về giá trị lớn nhất trong mảng
Sum()	Trả về giá trị tổng cộng các phần tử

➤ **Cách duyệt mảng 1 chiều:**

- Để truy xuất đến các phần tử của mảng cần thông qua chỉ số phần tử.
- Mà chỉ số phần tử là các số nguyên tăng dần.
- Từ đó ta có thể tận dụng vòng lặp để tăng giá trị 1 biến lên rồi xem biến đó như là chỉ số phần tử của mảng.



➤ **Ví dụ:**

```
0 references
static void Main(string[] arg)
{
    C.OutputEncoding = Encoding.Unicode;

    int[] bangGtr = { 3, 7, 0, 5, 9, 6, 4 };

    for (int i = 0; i < 7; i++)
    {
        C.WriteLine($"Giá trị thứ {i} trong mảng là: {bangGtr[i]}");
    }

    C.ReadKey();
}
```

## **BÀI TẬP**

1. Viết chương nhập vào số lượng trong mảng và số phần tử trong mảng, Sau đó tính tổng trong mảng.
2. Xóa 1 phần tử trong mảng và in ra.
3. Tương tự như bài 1 trên nhưng tìm ra những số duy nhất trong mảng;
4. Tương tự như bài 1 nhưng sắp xếp mảng theo thứ tự tăng dần và giảm dần.

