

# Bài 5.7: Biểu thức lambda

---

- ✓ Mục đích và cú pháp
- ✓ Biểu thức lambda vs delegate
- ✓ Kiểu trả về của biểu thức lambda
- ✓ Ví dụ minh họa
- ✓ Bài tập thực hành

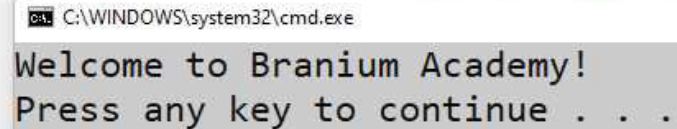
# Mục đích và cú pháp

- ✓ Biểu thức lambda là cấu trúc được sử dụng để tạo các hàm vô danh.
- ✓ Biểu thức lambda sử dụng toán tử khai báo lambda dạng mũi tên => để phân tách phần danh sách tham số và phần thân của nó.
- ✓ Biểu thức lambda có thể chia thành 2 loại:
  - ✓ Expression lambda: biểu thức lambda thường chứa các câu lệnh đơn ngắn gọn trong thân nó.
  - ✓ Statement lambda: câu lệnh lambda chứa các câu lệnh bên trong phần thân của nó.
- ✓ Biểu thức lambda: **(params) => expression;**
- ✓ Câu lệnh lambda: **(params) => { statements; }**
- ✓ Trong đó:
  - ✓ Params là nơi chứa các tham số mang dữ liệu đầu vào cho biểu thức hoạt động.
  - ✓ Expression là một biểu thức cần thực hiện.
  - ✓ Statements là các câu lệnh cần thực hiện.

# Ví dụ biểu thức lambda

- ✓ Sau đây là ví dụ về một biểu thức lambda không tham số:

```
// định nghĩa ra một biểu thức lambda và gán nó cho biến action
var action = () => Console.WriteLine("Welcome to Branium Academy!");
action(); // gọi biểu thức lambda như gọi phương thức
// kết quả:
```



```
C:\WINDOWS\system32\cmd.exe
Welcome to Branium Academy!
Press any key to continue . . .
```

```
var findMax = (int a, int b, int c) =>
{
    int max = a;
    if (max < b)
    {
        max = b;
    }
    if (max < c)
    {
        max = c;
    }
    return max;
};
int maxValue = findMax(5, 6, 9);
Console.WriteLine($"Max value in (5, 6, 9) is {maxValue}");
// kết quả: Min value in (5, 6, 9) is 9
```

# Ví dụ biểu thức lambda

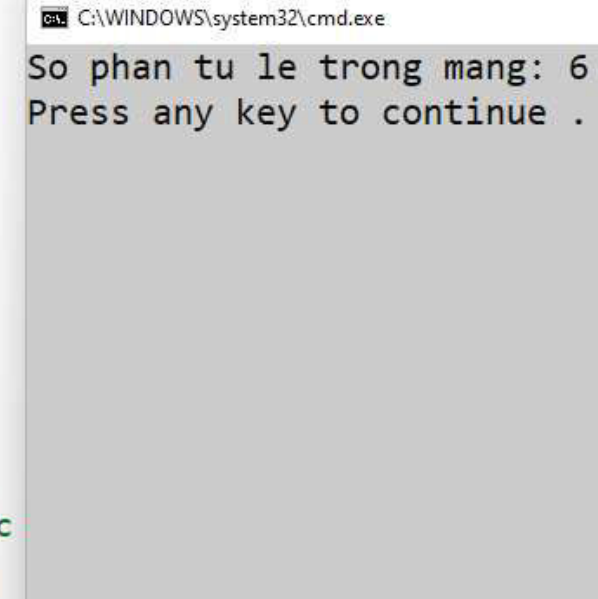
✓ Tìm max trong 3 số bằng biểu thức lambda:

```
var findMax = (int a, int b, int c) =>
{
    int max = a;
    if (max < b)
    {
        max = b;
    }
    if (max < c)
    {
        max = c;
    }
    return max;
};
int maxValue = findMax(5, 6, 9);
Console.WriteLine($"Max value in (5, 6, 9) is {maxValue}");
// kết quả: Max value in (5, 6, 9) is 9
```

# Ví dụ biểu thức lambda

✓ Sau đây là ví dụ về biểu thức lambda nhận vào một mảng và đếm số lượng phần tử lẻ:

```
int[] numbers = { 1, 2, 3, 4, 5, 6, 7, 8, 9, 10, 11, 12 };  
// định nghĩa câu lệnh lambda  
var countOddNumber = (int[] arr) =>  
{  
    int counter = 0;  
    foreach (var item in arr)  
    {  
        if (item % 2 != 0)  
        {  
            counter++;  
        }  
    }  
    return counter;  
};  
var result = countOddNumber(numbers); // gọi biểu tới thức  
Console.WriteLine("Số phần tử lẻ trong mảng: " + result);
```



```
C:\WINDOWS\system32\cmd.exe  
Số phần tử lẻ trong mảng: 6  
Press any key to continue .
```

# Biểu thức lambda vs delegate

- ✓ Delegate là một tham chiếu trỏ đến phương thức. Có 3 loại delegate với mục đích sử dụng khác nhau:
  - ✓ Action: loại delegate này thường sử dụng để tham chiếu tới các phương thức có kiểu trả về là void.
  - ✓ Func: loại delegate này thường sử dụng để tham chiếu tới các phương thức để chỉ rõ kiểu của các tham số cũng như kiểu trả về của phương thức.
  - ✓ Predicate: loại delegate này thường dùng để tham chiếu tới các phương thức luôn trả về kiểu bool và có nhận các tham số đầu vào.
- ✓ Bất kì biểu thức lambda nào cũng có thể chuyển đổi sang delegate.
- ✓ Kiểu delegate được xác định qua kiểu của các tham số và kiểu trả về của biểu thức lambda.
- ✓ Nếu biểu thức lambda không trả về gì cả. Ta có thể chuyển nó thành kiểu Action.
- ✓ Nếu biểu thức lambda có trả về, có nhận vào các tham số thì ta có thể chuyển nó sang kiểu Func delegate.

# Cú pháp chuyển đổi

- ✓ Cú pháp chung: **delegateType**<**parameterTypes**> **delegateVariable** = **lambdaExpression**;
- ✓ Trong đó:
  - ✓ delegateType có thể là Action, Func, Predicate.
  - ✓ parameterTypes là danh sách các kiểu tương ứng của tham số. Một delegate có thể nhận 0, 1 hoặc nhiều tham số. Tham số cuối với Func luôn là kiểu trả về của biểu thức lambda.
  - ✓ delegateVariable là tên biến delegate.
  - ✓ lambdaExpression là biểu thức lambda.

# Chuyển đổi biểu thức lambda sang kiểu delegate

```
// sử dụng Action delegate
Action action = () => Console.WriteLine("Welcome to Branium Academy!");
action(); // gọi delegate
// kết quả: Welcome to Branium Academy!

int[] numbers = { 1, 2, 3, 4, 5, 6, 7, 8, 9, 10, 11, 12 };
// định nghĩa câu lệnh lambda
Func<int[], int> countOddNumber = (int[] arr) =>
{
    int counter = 0;
    foreach (var item in arr)
    {
        if (item % 2 != 0)
        {
            counter++;
        }
    }
    return counter;
};
var result = countOddNumber(numbers); // gọi tới delegate
Console.WriteLine("Số phần tử lẻ trong mảng: " + result);
// kết quả: Số phần tử lẻ trong mảng: 6
```



# Chuyển biểu thức lambda sang kiểu delegate

```
Func<int, int, int, int> findMax = (int a, int b, int c) =>
{
    int max = a;
    if (max < b)
    {
        max = b;
    }
    if (max < c)
    {
        max = c;
    }
    return max;
};

int maxValue = findMax(5, 6, 9);
Console.WriteLine($"Max value in (5, 6, 9) is {maxValue}");
// kết quả: Max value in (5, 6, 9) is 9

// tìm số đầu tiên chia hết cho k != 0 trong mảng
Func<int[], int, int> firstDivisibleByK = (int[] arr, int k) =>
{
    foreach (var item in arr)
    {
        if (item % k == 0)
        {
            return item;
        }
    }
    return -1;
};

int k = 6;
Console.WriteLine(firstDivisibleByK(numbers, k)); // 6
```

# Kiểu trả về của biểu thức lambda

- ✓ Thông thường kiểu trả về của biểu thức lambda bị bỏ qua và tự suy luận dựa trên kết quả trả về.
- ✓ Từ C# 10, ta có thể chỉ định tường minh kiểu trả về của biểu thức lambda bằng cách đặt kiểu trả về ngay trước () chứa các tham số.

```
var findMax = int (int a, int b, int c) =>
{
    int max = a;
    if (max < b)
    {
        max = b;
    }
    if (max < c)
    {
        max = c;
    }
    return max;
};
int maxValue = findMax(5, 6, 9);
Console.WriteLine($"Max value in (5, 6, 9) is {maxValue}");
// kết quả: Min value in (5, 6, 9) is 9
```



# Nội dung tiếp theo

## Kiểu dữ liệu struct