Bài 4.1: Tìm hiểu về phương thức

- ✓ Khái niệm và mục đích sử dụng
- ✓ Cú pháp và quy ước
- ✓ Kiểu trả về của phương thức
- ✓ Ví dụ minh họa
- ✓ Bài tập thực hành

Khái niệm và mục đích sử dụng

- ✓Phương thức là một tập hợp các câu lệnh cùng thực hiện trọn vẹn 1 chức năng.
- ✓Phương thức thường liên quan đến một chức năng, hành động cụ thể nào đó.
- ✓Phương thức giúp ta dễ dàng quản lý mã nguồn, tái sử dụng và bảo trì, mở rộng code.
- ✓Ta đã sử dụng rất nhiều các phương thức từ đầu khóa học tới giờ. Ví dụ:
  - ✓ Phương thức Console.ReadLine() để đọc dữ liệu vào từ bàn phím.
  - ✓ Phương thức Main() để chạy chương trình.
  - ✓ Phương thức Parse() của từng kiểu dữ liệu để chuyển đổi string sang giá trị của kiểu đó.
  - ✓ Phương thức Split() của một đối tượng String để tách các từ.
  - ✓ Phương thức Write(), WriteLine() để hiển thị dữ liệu ra màn hình.
  - ✓ Các phương thức khác của lớp Math, String, Console…
- ✓Nội dung chương này ta sẽ mở rộng hiểu biết về phương thức để sử dụng hiệu quả cũng như tự tạo ra các phương thức để giải quyết vấn đề gặp phải.

Cú pháp và quy ước

✓Cú pháp tổng quát định nghĩa phương thức:

```
<access modifier> <static> <return type> MethodName(params) {
        // statements
}
```

✓Trong đó:

- ✓ Phần <access modifier> quản lý mức độ khả dụng của phương thức với code bên ngoài lớp. Chúng được điều khiển bởi các keyword như public, private, protected, internal… Chi tiết ta sẽ tìm hiểu ở phần hướng đối tượng. Bây giờ ta bỏ qua phần này.
- ✓ Phần <static> có thể có hoặc không. Bây giờ mặc định ta bắt đầu phương thức với keyword static. Keyword này cho phép ta gọi phương thức mà không cần tạo đối tượng.
- ✓ Phần <return type> là bắt buộc. Nó chỉ ra kiểu trả về của phương thức. Một phương thức bắt buộc phải có kiểu trả về như int, float, string,… hoặc void nếu không trả về gì cả.
- ✓ MethodName là tên phương thức. Tên phương thức thường là một động từ, cụm động từ hoặc chức năng mà nó đảm nhiệm. Tên phương thức bắt đầu với chữ cái hoa và nếu có nhiều từ thì viết liền và viết hoa chữ cái đầu từ.

Cú pháp và quy ước

- ✓Cú pháp tổng quát định nghĩa phương thức:

```
<access modifier> <static> <return type> MethodName(params) {
        // statements
}
```

- ✓Trong đó:
  - ✓ Đi sau tên phương thức là cặp ngoặc tròn.
  - ✓ Bên trong cặp ngoặc tròn là các tham số(params) cung cấp dữ liệu cần thiết để phương thức vận hành. Một phương thức có thể có 0, 1 hoặc nhiều tham số. Các tham số phân tách nhau bởi dấu phẩy.
  - ✓ Phần thân của phương thức bao bởi cặp ngoặc {} và chứa các câu lệnh cần thực hiện bên trong.
  - ✓ Một phương thức phải nằm bên trong một lớp nào đó.
- ✓Khi đã có phương thức, ta sẽ gọi phương thức qua tên và cung cấp các đối số cần thiết.
- ✓Biến trong ngoặc tròn của định nghĩa phương thức gọi là tham số(parameters).
- ✓Giá trị truyền vào phương thức trong lời gọi phương thức gọi là đối số(arguments).

Cú pháp và quy ước

- ✓Khi gọi phương thức bắt buộc phải truyền đủ và đúng kiểu các đối số vào các tham số tương ứng.
- ✓Tên các đối số và tham số không bắt buộc phải trùng nhau nhưng giá trị phải cùng kiểu tương ứng.
- ✓Ví dụ:

```
1 reference
static bool IsPerfectNumber(int number)
{
    int sum = 0;
    // tính tổng ước
    for (int k = 1; k < number; k++)
    {
        if (number % k == 0)
        {
            sum += k;
        }
    }
    // nếu sum == number, return true, ngược lại return false
    return sum == number;
}
```

```
// kiểm tra và đưa ra kết luận n có hoàn hảo không
if (IsPerfectNumber(n))
{
    Console.WriteLine($"Test {i}: YES");
}
else
{
    Console.WriteLine($"Test {i}: NO");
}
```

Ví dụ khác

```
    int a = 1;
    int b = 30;
    int k = 6;
    ListedDivisibleByK(a, b, k);
}

// Liệt kê các số chia hết cho k trong đoạn [a, b]
1 reference
static void ListedDivisibleByK(int a, int b, int k)
{
    for (int i = a; i <= b; i++)
    {
        if(i % k == 0)
        {
            Console.Write(i + " ");
        }
    }
    Console.WriteLine();
}
```

Lời gọi phương thức

Đối số

Tham số

```
C:\WINDOWS\system32\cmd.exe
6 12 18 24 30
Press any key to continue . . .
```

Kiểu trả về của phương thức

- ✓Nếu một phương thức sau khi thực hiện không cần thông báo kết quả cho nơi gọi nó.
- ✓Hoặc nếu kết quả thực hiện của một phương thức không được sử dụng làm đầu vào cho các chức năng nào khác ta để kiểu trả về của phương thức đó là void.
- ✓Trong phương thức kiểu void có thể có keyword return; ở cuối hoặc không.

```
// Liệt kê các số chia hết cho k trong đoạn [a, b]
1 reference
static void ListedDivisibleByK(int a, int b, int k)
{
    for (int i = a; i <= b; i++)
    {
        if(i % k == 0)
        {
            Console.Write(i + " ");
        }
    }
    Console.WriteLine();
    return; // có hoặc không đều ok
}
```

Kiểu trả về của phương thức

- ✓Ngược lại, nếu kết quả của một phương thức là quan trọng, sẽ được tái sử dụng hoặc là đầu vào của chức năng nào đó về sau, phương thức đó nên có kiểu trả về khác void.
- ✓Với mọi phương thức có kiểu trả về khác void, ta phải có return expression; ở cuối hoặc trong mọi trường hợp có thể xảy ra của phương thức.
- ✓Giá trị trả về trong expression phải cùng kiểu với kiểu trả về đã chỉ định của phương thức.
- ✓Ta có thể sử dụng trực tiếp giá trị trả về này trong lời gọi phương thức hoặc gán vào biến cục bộ cho thuận tiện về sau.

Ví dụ

✓Kiểm tra xem n có phải số nguyên tố không:

```
// Phương thức kiểm tra n có phải số nguyên tố không
0 references
static bool IsPrimeNumber(int n)
{
    if (n < 2) // mọi số nguyên < 2 không phải số nguyên tố
    {
        return false;
    }
    int bound = (int)Math.Sqrt(n); // lấy phần nguyên căn bậc hai của n
    for (int i = 2; i <= bound; i++)
    {
        if(n % i == 0) // nếu tồn tại ước trong đoạn [2, bound]
        {
            return false; // n không phải số nguyên tố
        }
    }
    return true; // lúc này n không có ước nào khác 1 và n nên là số nguyên tố
}
```

Ví dụ

✓Lời gọi và kết quả:

```
int n = 21;
int p = 7;
Console.WriteLine($"{n} is prime number? {IsPrimeNumber(n)}");
Console.WriteLine($"50 is prime number? {IsPrimeNumber(50)}");
bool result = IsPrimeNumber(p); // lưu lại kết quả
Console.WriteLine($"{p} is prime number? {result}");
// kết quả:
// 21 is prime number ? False
// 50 is prime number ? False
// 7 is prime number ? True
```

Nội dung tiếp theo

Tìm hiểu về các tham số