

C# Tham Khảo Nhanh

Kiểu dữ liệu, LINQ, async/await, collection, OOP

Cơ Bản

Hello World

```
Console.WriteLine("Hello, World!"); // top-level (C# 10+)
// Có thể: class Program { static void Main() { ... } }
```

Build & Chạy

```
dotnet new console -n MyApp # tạo dự án
dotnet run # biên dịch và chạy
dotnet build # chỉ biên dịch
```

Biến & Hằng

```
int x = 42;
var name = "Alice"; // suy luận kiểu
const double Pi = 3.14159;
readonly int maxRetries = 3; // đặt một lần, trong constructor
```

Kiểu Dữ Liệu

Value Types

int	Số nguyên có dấu 32-bit
long	Số nguyên có dấu 64-bit
float	Dấu phẩy động 32-bit (hậu tố f)
double	Dấu phẩy động 64-bit
decimal	Độ chính xác cao 128-bit (hậu tố m)
bool	true / false
char	Ký tự Unicode 16-bit

Reference Types

string	Văn bản UTF-16 bất biến
object	Kiểu cơ sở cho tất cả kiểu dữ liệu
dynamic	Bỏ qua kiểm tra kiểu lúc biên dịch
int[]	Mảng số nguyên
List<T>	Danh sách generic (System.Collections.Generic)

Nullable & Tuples

```
int? age = null; // value type nullable
string? name = null; // reference nullable (C# 8+)
var point = (X: 1, Y: 2); // named tuple
Console.WriteLine(point.X);
```

Tính Năng Chuỗi

```
string name = "World";
string msg = $"Hello, {name}!"; // nội suy
string path = @"C:\Users\file.txt"; // verbatim
string raw = """raw "string" here"""; // raw (C# 11+)
```

Luồng Điều Khiển

If / Else

```
if (x > 0) Console.WriteLine("dương");
else if (x == 0) Console.WriteLine("không");
else Console.WriteLine("âm");
```

Switch & Pattern Matching

```
string label = x switch {
    > 0 => "dương", 0 => "không", _ => "âm"
};
if (obj is string s && s.Length > 0) { } // khớp pattern
```

Vòng Lặp

```
for (int i = 0; i < 10; i++) { }
foreach (var item in collection) { }
while (condition) { }
do { } while (condition);
```

Classes

Định Nghĩa Class

```
public class Person {
    public string Name { get; set; }
    public int Age { get; init; } // init-only (C# 9+)
    public Person(string name, int age) { Name = name; Age = age; }
}
```

Records (C# 9+)

```
public record Point(double X, double Y);
var p1 = new Point(1, 2);
var p2 = p1 with { X = 3 }; // sao chép không phá hủy
// tự động: Equals, GetHashCode, ToString, deconstruct
```

Kế Thừa

```
public abstract class Shape { public abstract double Area(); }
public class Circle(double r) : Shape {
    public override double Area() => Math.PI * r * r;
}
```

Access Modifiers

public	Truy cập từ bất kỳ đâu
private	Chỉ cùng class (mặc định cho member)
protected	Cùng class và class dẫn xuất
internal	Chỉ cùng assembly (mặc định cho class)
protected internal	Cùng assembly hoặc class dẫn xuất

Interfaces

Định Nghĩa Interface

```
public interface IShape {
    double Area();
    double Perimeter() => 0; // impl mặc định (C# 8+)
}
public class Rect(double w, double h) : IShape { public double
Area() => w * h; }
```

Interface Thường Dùng

IEnumerable<T>	Hỗ trợ iteration (foreach, LINQ)
IDisposable	Dọn dẹp tài định (using statement)
IComparable<T>	Sắp xếp tự nhiên
IComparable<T>	Số sánh bằng nhau theo giá trị
ICloneable	Nhân bản object

LINQ

Cú Pháp Method

```
var result = numbers
    .Where(n => n > 3)
    .OrderBy(n => n)
    .Select(n => n * 2)
    .ToList();
```

Cú Pháp Query

```
var result = from n in numbers
              where n > 3
              orderby n
              select n * 2;
```

Các Method LINQ Thường Dùng

.Where(pred)	Lọc phần tử
.Select(func)	Chiếu / biến đổi phần tử
.OrderBy(key)	Sắp xếp tăng dần
.GroupBy(key)	Nhóm phần tử theo key
.First() / .FirstOrDefault()	Phần tử đầu tiên (hoặc mặc định)
.Any(pred)	true nếu có phần tử nào khớp
.Count()	Số lượng phần tử
.Sum() / .Average()	Tổng hợp giá trị số
.Distinct()	Bỏ trùng lặp
.SelectMany(func)	Làm phẳng collection lồng nhau

Async/Await

Async Method

```
public async Task<string> FetchAsync(string url) {
    using var client = new HttpClient();
    return await client.GetStringAsync(url);
}
```

Task Combinators

```
var results = await Task.WhenAll(task1, task2, task3);
var first = await Task.WhenAny(task1, task2);
```

Các Pattern Async

Task	Trả về async void (không có kết quả)
Task<T>	Trả về async với kết quả kiểu T
ValueTask<T>	Task nhẹ cho các đường dẫn đồng bộ nhanh
await foreach	Iteration async trên IAsyncEnumerable<T>
CancellationToken	Hủy bỏ hợp tác cho thao tác async

Collections

Collection Thường Dùng

List<T>	Mảng động, truy cập index nhanh
Dictionary<K, V>	Hash map, tra cứu O(1) theo key
HashSet<T>	Phần tử duy nhất, tra cứu O(1)
Queue<T>	Collection FIFO
Stack<T>	Collection LIFO
LinkedList<T>	Danh sách liên kết đôi
SortedDictionary<K, V>	Sắp xếp theo key (dựa trên cây)

Sử Dụng Dictionary

```
var dict = new Dictionary<string, int> {
    ["Alice"] = 90, ["Bob"] = 85
};
dict.TryGetValue("Alice", out int score);
foreach (var (key, val) in dict) { }
```

Immutable Collections

```
using System.Collections.Immutable;
var list = ImmutableList.Create(1, 2, 3);
var newList = list.Add(4); // trả về list mới
```

Properties

Cú Pháp Property

```
public string Name { get; set; }
public int Age { get; private set; }
public string Email { get; init; } // init-only
public string Display => $"{Name} ({Age})"; // tính toán
```

C# Tham Khảo Nhanh

Indexers

```
public double this[int row, int col] {  
    get => data[row, col];  
    set => data[row, col] = value;  
}
```

Các Pattern Property

<code>{ get; set; }</code>	Auto-property đọc-ghi
<code>{ get; }</code>	Chỉ đọc (chỉ đặt trong constructor)
<code>{ get; init; }</code>	Chỉ đọc sau khởi tạo (C# 9+)
<code>{ get; private set; }</code>	Đọc công khai, ghi riêng tư
<code>=> expression</code>	Property tính toán (expression-bodied)

Exceptions

Try / Catch / Finally

```
try { int result = int.Parse(input); }  
catch (FormatException ex) { Console.Error.WriteLine(ex.Message); }  
catch (Exception ex) when (ex is not OutOfMemoryException) { }  
finally { /* luôn thực thi */ }
```

Using Statement

```
using var file = File.OpenRead("data.txt");  
// file.Dispose() gọi tự động khi kết thúc scope  
// tương đương try/finally với Dispose()
```

Exception Thường Gặp

ArgumentNullException	Tham số null được truyền vào method
ArgumentOutOfRangeException	Tham số ngoài phạm vi hợp lệ
InvalidOperationException	Thao tác không hợp lệ với trạng thái hiện tại
NullReferenceException	Hủy tham chiếu đến object null
KeyNotFoundException	Không tìm thấy key trong dictionary
NotImplementedException	Method chưa được triển khai

Custom Exception

```
public class AppException : Exception {  
    public int Code { get; }  
    public AppException(string msg, int code)  
        : base(msg) { Code = code; }  
}
```