

Referensi Cepat C#

Tipe, LINQ, async/await, koleksi, esensial OOP

Dasar

Hello World

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

Build & Jalankan

```
dotnet new console -n MyApp # create project
dotnet run # compile and run
dotnet build # compile only
```

Variabel & Konstanta

```
int x = 42;
var name = "Alice"; // type inference
const double Pi = 3.14159;
readonly int maxRetries = 3; // set once, in ctor
```

Tipe

Value Type

int	Integer 32-bit bertanda
long	Integer 64-bit bertanda
float	Floating point 32-bit (suffix f)
double	Floating point 64-bit
decimal	Presisi tinggi 128-bit (suffix m)
bool	true / false
char	Karakter Unicode 16-bit

Reference Type

string	Teks UTF-16 yang immutable
object	Tipe dasar semua tipe
dynamic	Lewati pemeriksaan tipe compile-time
int[]	Array integer
List<T>	Daftar generik (System.Collections.Generic)

Nullable & Tuple

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

Fitur String

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

Control Flow

If / Else

```
if (x > 0) Console.WriteLine("positive");
else if (x == 0) Console.WriteLine("zero");
else Console.WriteLine("negative");
```

Switch & Pattern Matching

```
string label = x switch {
    > 0 => "positive", 0 => "zero", _ => "negative"
};
if (obj is string s && s.Length > 0) { } // pattern match
```

Loop

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

Class

Definisi 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; }
}
```

Record (C# 9+)

```
public record Point(double X, double Y);
var p1 = new Point(1, 2);
var p2 = p1 with { X = 3 }; // non-destructive copy
// auto: Equals, GetHashCode, ToString, deconstruct
```

Inheritance

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

Access Modifier

public	Dapat diakses dari mana saja
private	Hanya class yang sama (default untuk member)
protected	Class yang sama dan class turunan
internal	Hanya assembly yang sama (default untuk class)
protected internal	Assembly yang sama atau class turunan

Interface

Definisi Interface

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

Interface Umum

IEnumerable<T>	Dukungan iterasi (foreach, LINQ)
IDisposable	Pembersihan deterministik (pernyataan using)
IComparable<T>	Pengurutan alami untuk sorting
IEquatable<T>	Perbandingan kesetaraan nilai
ICloneable	Kloning objek

LINQ

Sintaks Method

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

Sintaks Query

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

Method LINQ Umum

.Where(pred)	Filter elemen
.Select(func)	Proyeksi / transformasi elemen
.OrderBy(key)	Urutkan ascending
.GroupBy(key)	Kelompokkan elemen berdasarkan key
.First() / .FirstOrDefault()	Elemen pertama (atau default)
.Any(pred)	true jika ada elemen yang cocok
.Count()	Jumlah elemen
.Sum() / .Average()	Agregasi nilai numerik
.Distinct()	Hapus duplikat
.SelectMany(func)	Ratakan koleksi bersarang

Async/Await

Method Async

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

Kombinator Task

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

Pola Async

Task	Return async void (tanpa hasil)
Task<T>	Return async dengan hasil bertipe T
ValueTask<T>	Task ringan untuk jalur sync-fast
await foreach	Iterasi async atas IAsyncEnumerable<T>
CancellationToken	Pembatalan kooperatif untuk operasi async

Koleksi

Koleksi Umum

List<T>	Array dinamis, akses indeks cepat
Dictionary<K, V>	Hash map, lookup O(1) berdasarkan key
HashSet<T>	Elemen unik, lookup O(1)
Queue<T>	Koleksi FIFO
Stack<T>	Koleksi LIFO
LinkedList<T>	Linked list dua arah
SortedDictionary<K, V>	Diurutkan berdasarkan key (berbasis tree)

Penggunaan Dictionary

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

Koleksi Immutable

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

Property

Sintaks Property

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

Referensi Cepat C#

Indexer

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

Pola Property

{ get; set; }	Auto-property read-write
{ get; }	Read-only (hanya bisa di-set dalam constructor)
{ get; init; }	Read-only setelah inialisasi (C# 9+)
{ get; private set; }	Bisa dibaca publik, ditulis privat
=> expression	Property expression-bodied (computed)

Exception

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 { /* always executes */ }
```

Pernyataan Using

```
using var file = File.OpenRead("data.txt");
// file.Dispose() called automatically at scope end
// equivalent to try/finally with Dispose()
```

Exception Umum

ArgumentNullException	Argumen null dikirim ke method
ArgumentOutOfRangeException	Argumen di luar rentang valid
InvalidOperationException	Operasi tidak valid untuk kondisi saat ini
NullReferenceException	Dereference objek null
KeyNotFoundException	Key tidak ditemukan dalam dictionary
NotImplementedException	Method belum diimplementasikan

Exception Kustom

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