C# proqramlaşdırma dilində data type — proqram daxilində saxlanılan verilənlərin növünü və ölçüsünü təyin edir. Hər bir data type yaddaşda müəyyən ölçüdə yer tutur və yalnız uyğun verilən növünü qəbul edir.

C#-da data type-lar əsasən 3 böyük kateqoriyaya bölünür:

1. Value Type-lar

Value type-lar verilənlərin öz dəyərini birbaşa yaddaşda saxlayır. Bu tiplər stack yaddaş sahəsində yerləşir.

Əsas Value Type-lar:

* int, double, float, bool, char, long, byte, short
* struct
* enum

Note: Value type-lar struct uzerinde qurulub.

Stack yaddaş çox sürətli işləyir və value type-lar kiçik ölçülü olduqları üçün burada saxlanmaq daha effektivdir.

1. Reference Type-lar

Reference type-lar dəyişənin öz dəyərini deyil, yaddaşda yerləşdiyi ünvanı saxlayır. Bu tiplər heap yaddaş sahəsində yerləşir. Əgər iki dəyişən eyni obyektə istinad edirsə, hər ikisi həmin obyektə birbaşa təsir edə bilər.

Əsas Reference Type-lar:

* string
* class
* object

Note: C#-da object tipi butun data type-larin base classidir, cunki .Netde butun tipler System. Objectden toreyir.

Reference type-lar çox zaman daha böyük və mürəkkəb strukturlar olduğu üçün heap kimi dinamik yaddaşda saxlanılır. Heap daha çevikdir, ancaq idarə olunması üçün əlavə zaman tələb edir (GC - Garbage Collector vasitəsilə).

1. Pointer Type-lar

C#-da pointer-lər xüsusi unsafe bloklarda istifadə olunur və birbaşa yaddaş ünvanları ilə işləməyə imkan verir. Adətən aşağı səviyyəli sistem proqramlaşdırmasında istifadə olunur.

2.

**1. C#-da Data Type-lar, harda saxlanılır və səbəbləri**

C#-da data type-lar iki əsas kateqoriyaya bölünür: **Value type** və **Reference type**.

* **Value type-lar** — int, double, bool, char, struct, enum və s. kimi tiplərdir. Bunlar **Stack** yaddaşında saxlanılır və dəyərin **kopyası ilə işləyir**. Bu, onların daha sürətli işləməsinə səbəb olur.
* **Reference type-lar** — string, class, object, interface, delegate, array və s. kimi tiplərdir. Bunlar **Heap** yaddaşında saxlanılır və dəyişən yalnız obyektin **referansını** saxlayır. Məsələn, iki dəyişən eyni obyektə baxa bilər.

Yəni:

* int a = 5; int b = a; → b ayrı 5 saxlayır
* Person p1 = new Person(); Person p2 = p1; → hər ikisi eyni obyektə baxır

**2. CLR nədir, nələri təmin edir və neçə hissəyə bölünür?**

**CLR (Common Language Runtime)** — .NET-in əsas işləmə mühərrikidir. Yazılmış C# kodu əvvəlcə **IL (Intermediate Language)** koduna çevrilir, sonra CLR tərəfindən **runtime zamanı icra olunur**.

CLR-in əsas funksiyaları:

* **JIT Compilation** – IL kodunu maşın koduna çevirir
* **Memory Management** – Garbage Collector istifadə olunmayan obyektləri silir
* **Exception Handling** – səhvləri idarə edir (try-catch)
* **Type Safety** – tip uyğunsuzluğunun qarşısını alır
* **Security** – təhlükəsizlik yoxlamaları aparır

CLR əsasən bu hissələrə bölünür:

1. **Class Loader**
2. **JIT Compiler**
3. **Garbage Collector**
4. **Security Engine**
5. **Exception Manager**

**Compiler** (məsələn, csc.exe) isə C# kodunu **IL koduna** çevirir. CLR isə bu kodu **icra edir**.

**3. Misc operatorlar və istifadə məqsədləri**

C#-da bəzi operatorlar xüsusi məqsədlərlə istifadə olunur:

* **sizeof** – yalnız unsafe blokda işləyir, value type-ların yaddaş ölçüsünü qaytarır.  
  Məs: sizeof(int) → 4
* **typeof** – bir tipin Type obyektini verir.  
  Məs: typeof(string) → System.Type obyektidir
* **is** – obyektin verilmiş tipə uyğun olub olmadığını yoxlayır, true/false qaytarır.  
  Məs: x is string
* **as** – tip çevirməyə çalışır, əgər mümkün deyilsə null verir.  
  Məs: obj as Person

is yoxlayır, as çevirməyə çalışır. is true/false verir, as isə ya obyekt, ya null.

**4. switch ilə switch pattern arasında fərqlər**

Ənənəvi switch yalnız sabitlərlə (case 1, case 2 və s.) müqayisə aparır. Bu struktur sadə, amma məhduddur.

**Pattern-based switch** isə daha müasir və güclü yanaşmadır. Burada **şərt ifadələri**, **tip yoxlamaları**, when açar sözü və s. istifadə oluna bilər.

Məsələn:

csharp

CopyEdit

switch (x)

{

case int n when n > 0: Console.WriteLine("Positive"); break;

case < 0: Console.WriteLine("Negative"); break;

case 0: Console.WriteLine("Zero"); break;

}

Bu, switch pattern-dir və çox çevikdir.

Əsas fərqlər:

* Ənənəvi switch yalnız == müqayisəsi edir
* Pattern switch isə şərtlərlə, tiplə və when ilə daha ağıllı işləyir