C# jegyzet

# C# keletkezése, története

A C# nyelvet a Microsoft fejlesztette ki, és először 2000-ben mutatták be. A nyelv kidolgozása Anders Hejlsberg nevéhez köthető, aki korábban a Turbo Pascal és a Delphi nyelvek fejlesztésében is részt vett. A C# a .NET keretrendszer része, és eredetileg a Java nyelvre adott válaszként született, bár mára önálló, erősen fejlett nyelvvé vált.

# C# és VFP

A Visual FoxPro (VFP) adatbázis-központú programozási nyelv és fejlesztői környezet, amelyet főleg üzleti alkalmazások fejlesztésére használtak. A Visual FoxPro és a C# nagyon eltérő paradigmákon alapul.

|  |  |  |
| --- | --- | --- |
| **Tulajdonság** | **C#** | **Visual FoxPro** |
| **Paradigma** | Objektumorientált (OOP) és típusbiztos nyelv | Objektumorientált, de dinamikus és laza típuskezelés |
| **Használat** | Modern alkalmazások fejlesztése (*web, mobil, asztali, játékok, cloud, AI, stb.*) | Adatbázis-központú alkalmazások fejlesztése |
| **Típusosság** | Erősen típusos (*statikus*) | Gyengén típusos (*dinamikus adattípusokkal*) |

### Objektum-orientált paradigmára való áttérés

|  |  |  |
| --- | --- | --- |
| **Tulajdonság** | **C#** | **Visual FoxPro** |
| **OOP támogatás** | Teljes OOP (*osztályok, öröklődés, polimorfizmus, interfészek, absztrakt osztályok*) | OOP-t támogat, de korlátozott |
| **Metódusok túlterhelése** | Igen | Nem |
| **Interfészek és absztrakt osztályok** | Igen | Nem |

### **Adatbázis-kezelés különbségei**

A C#-ban nincs beépített adatbázis-kezelés, de a .NET keretrendszer széleskörű támogatást nyújt különböző adatbázisokhoz (*pl. SQL Server, MySQL, SQLite*) az **ADO.NET**, **Entity Framework** vagy **Dapper** segítségével.

|  |  |  |
| --- | --- | --- |
| **Tulajdonság** | **C#** | **Visual FoxPro** |
| **Adatbázis-motor** | Külső adatbázisokkal (*SQL Server, MySQL, PostgreSQL, SQLite, stb.*) működik | Beépített adatbázis (DBF fájlok), SQL támogatás korlátozott |
| **Kapcsolat adatbázishoz** | ADO.NET, Entity Framework, Dapper | Beépített SQL Pass-Through, Cursor, DBF táblák |
| **Többfelhasználós támogatás** | SQL szerverekkel skálázható | Többfelhasználós, de fájlalapú DBF szerkezet miatt sérülékeny |

### ORM (*Object-Relational Mapping*) használata

Az ORM automatikusan leképezi az adatbázis tábláit **C# osztályokra**, így nem kell SQL lekérdezéseket írnod minden egyes adatkezelési művelethez.

**Két fő ORM lehetőség C#-ban:**

| **ORM** | **Leírás** |
| --- | --- |
| **Entity Framework (EF) Core** | A Microsoft hivatalos ORM-je, erősen integrált a .NET ökoszisztémába. |
| **Dapper** | "Micro ORM", amely kevesebb absztrakciót tartalmaz és gyorsabb, mint az EF Core. |

**ORM előnyei**

* **Kevesebb kód** → Automatikusan kezeli az adatbázis műveleteket (DbContext, DbSet).
* **Egyszerűbb karbantartás** → Nem kell kézzel írni minden SQL lekérdezést.
* **Típusbiztos** → Az adatok automatikusan leképeződnek C# osztályokra.
* **Automatikus migrációk** → Az adatbázis szerkezete programból kezelhető.
* **Kapcsolatok támogatása** → Pl. One-to-Many, Many-to-Many kapcsolatok kezelése könnyebb.

**ORM hátrányai**

* **Lassabb lehet nagy lekérdezések esetén**, mert túl sok absztrakciót tartalmaz.
* **Nem mindig hatékony SQL-t generál**, ami teljesítményveszteséget okozhat.
* **Tanulási görbe** → Az ORM-ek működését és az optimalizálásukat meg kell tanulni.

**Sebesség-összehasonlítás ORM és ADO.NET között**

|  |  |  |  |
| --- | --- | --- | --- |
| **Művelet** | **Entity Framework Core** | **Dapper** | **ADO.NET** |
| **Egyszerű SELECT (1 rekord)** | 🔴 Lassúbb | 🟡 Közepesen gyors | 🟢 Leggyorsabb |
| **Több rekord lekérdezése (10,000+ rekord)** | 🔴 Lassabb | 🟢 Gyors | 🟢 Gyors |
| **Adat beszúrása (INSERT)** | 🟡 Közepes | 🟢 Gyors | 🟢 Gyors |
| **Összetett SQL lekérdezés (JOIN, GROUP BY, stb.)** | 🔴 Lassabb | 🟡 Közepes | 🟢 Gyors |
| **Tanulási görbe** | 🟢 Könnyű | 🟡 Közepes | 🔴 Nehéz |

**Mikor érdemes ORM-et használni?**

* Ha **gyors fejlesztést** szeretnél, és nem akarsz SQL-lekérdezésekkel foglalkozni.
* Ha **kis vagy közepes méretű adatbázist** kezelsz.
* Ha **több fejlesztő dolgozik a projekten**, mert az ORM strukturáltabb kódot biztosít.

### Felhasználói felület (UI)

|  |  |  |
| --- | --- | --- |
| **Tulajdonság** | **C#** | **Visual FoxPro** |
| **UI keretrendszer** | WinForms, WPF, Blazor, MAUI, ASP.NET, Unity | Beépített UI komponensek |
| **Webes támogatás** | Igen (ASP.NET, Blazor) | Nem (csak asztali alkalmazások) |
| **Korszerű UI támogatás** | Modern UI/UX lehetőségek | Elavult felület, korlátozott lehetőségek |

# **Elemi adattípusok**

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
|  | ***Típus*** | ***Típusnév másképpen (egyenértékű)*** | ***Helyigény (bit)*** | ***Példa*** |
| **Logikai változó** | **bool** | Boolean | 8 (!) | true, false |
| **Egész szám** (***\*****: előjel nélküli*) | **sbyte, byte\*** | SByte, Byte | 8 | - |
| **short, ushort\*** | Int16, UInt16 | 16 | - |
| **int, uint\*** | Int32, UInt32 | 32 | 12, 12u |
| **long, ulong\*** | Int64, UInt64 | 64 | 12l, 12ul |
| **Lebegőpontos szám** | **float** | Single | 32 | 6.5f |
| **double** | Double | 64 | 6.5 |
| **decimal** | Decimal | 128 | 12m |
| **Karakter** | **char** | Char | 8 | ’c’ |
| **Szöveg** | **string** | String | változó | ”szöveg” |

1. **Összetett adatszerkezetek** – Például tömbök, listák (List<T>), szótárak (Dictionary<K,V>), halmazok (HashSet<T>), valamint saját osztályok és struktúrák.
2. **Vezérlési szerkezetek** – Feltételes elágazások (if, switch), ciklusok (for, while, foreach), kivételkezelés (try-catch).
3. **Metódusok és függvények** – Hogyan szervezd a kódot újrafelhasználható egységekké, paraméterátadás, visszatérési értékek.
4. **OOP (Objektumorientált programozás)** – Osztályok, öröklődés, interfészek, absztrakt osztályok, polimorfizmus.
5. **Fájlkezelés és adatbázis kapcsolat** – Hogyan olvashatsz és írhatsz fájlokat (StreamReader, StreamWriter), illetve használhatsz adatbázisokat (pl. SQL kapcsolat Entity Framework segítségével).
6. **Aszinkron programozás** – async és await, párhuzamos végrehajtás (Task, Thread).