C# 12

Ca parte a lansarii .NET 8.0 din 14 noiembrie 2023, Microsoft a dezvaluit si noile caracteristici C# 12, cea mai recenta versiune a celui mai popular limbaj de programare .NET. Astfel, C# 12 aduce o productivitate mai bună pentru dezvoltatori, cu o sintaxă simplificată și o execuție mai rapidă.

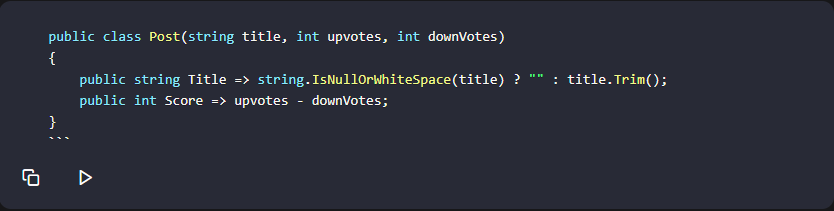
C# 12 introduce *expresii de colecție*, constructori primari pentru toate clasele și structurile, array-uri inline, interceptori, sintaxa pentru orice tip de alias și parametrii impliciti pentru expresiile lambda, respectiv parametri de tip *ref readonly*.

1. *Constructori primari*

Acum pot fi creati **constructori primari** în orice **clasa** și **struct**. Constructorii primari nu mai sunt restricționați la tipuri **record**. Parametrii constructorului primar sunt în domeniul de aplicare pentru întregul corp al clasei.

Adăugarea unui constructor primar la o clasa împiedică compilatorul să declare un constructor implicit fără parametri. Într-un struct, constructorul implicit fără parametri inițializează toate câmpurile, inclusiv parametrii constructorului primar la modelul de 0 biți.

Compilatorul generează proprietăți publice pentru parametrii constructorului primar numai în tipuri record, fie clase sau structuri de tip record. Clasele și structurile **non**-**record** ar putea să nu dorească întotdeauna acest comportament pentru parametrii constructorului primar.



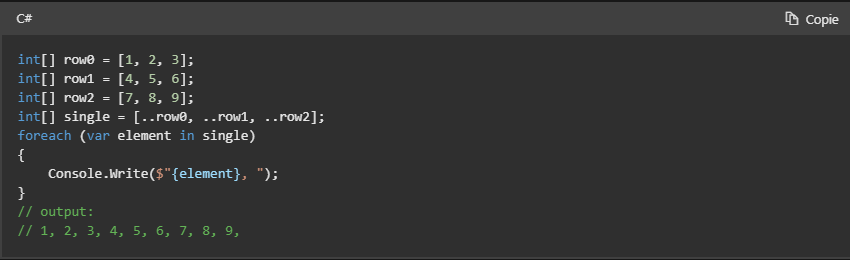
1. *Expresii de colecție*

Expresiile de colecție introduc o nouă sintaxă concisă pentru a crea valori comune de colecție. Integrarea altor colecții în aceste valori este posibilă folosind un operator de împrăștiere **…**.

Pot fi create mai multe tipuri asemănătoare colecțiilor fără a necesita suport extern **BCL**. Aceste tipuri sunt:

* Tipuri de matrice, cum ar fi **int[]**.
* **System.Span<T>** și **System.ReadOnlySpan<T>** .
* Tipuri care acceptă inițializatorii de colecții, cum ar fi **System.Collections.Generic.List<T>**.





Se pot folosi expresii de colecție oriunde este nevoie de o colecție de elemente. Ele pot specifica valoarea inițială pentru o colecție sau pot fi transmise ca argumente metodelor care preiau tipuri de colecție.

1. *Parametri* ***ref readonly***

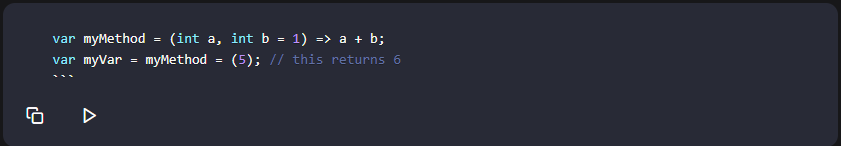
C# a adăugat parametrii **in** ca o modalitate de a transmite referințe numai în citire. Parametrii **in** permit atât variabile, cât și valori și pot fi utilizați fără nicio adnotare pe argumente.

Adăugarea de parametri **ref readonly** permite mai multă claritate pentru **API**-urile care ar putea folosi parametri **ref** sau **in**:

* API-urile create înainte de ina fi introduse ar putea folosi ref, chiar dacă argumentul nu este modificat. Aceste API-uri pot fi actualizate cu **ref readonly**. Nu va fi o schimbare importantă pentru apelanți, așa cum ar fi dacă parametrul **ref** ar fi schimbat în **in**. Un exemplu este **System.Runtime.InteropServices.Marshal.QueryInterface**.
* API-uri care preiau un **in** parametru, dar necesită in mod logic o variabilă. O expresie de valoare nu funcționează. Un exemplu este **System.ReadOnlySpan<T>.ReadOnlySpan<T>(T)** .
* API-uri care folosesc **ref** deoarece necesită o variabilă, dar nu modifică acea variabilă. Un exemplu este **System.Runtime.CompilerServices.Unsafe.IsNullRef**.

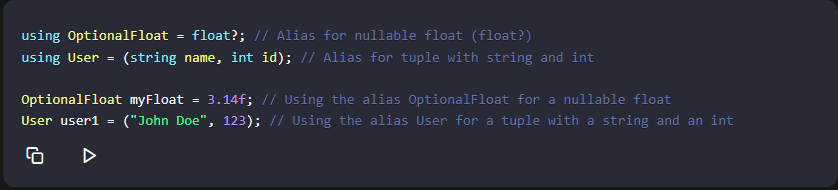
1. *Parametrii lambda impliciti*

In C# 12, se pot defini valori implicite pentru parametrii expresiilor lambda. Sintaxa și regulile sunt aceleași cu adăugarea valorilor implicite pentru argumente la orice metodă sau funcție locală.



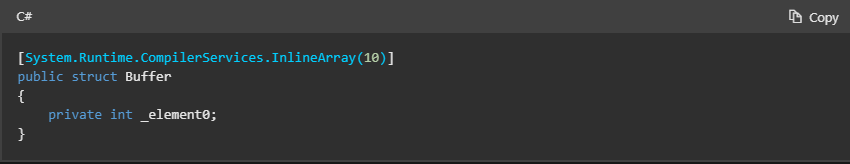
1. *Alias pentru orice tip*

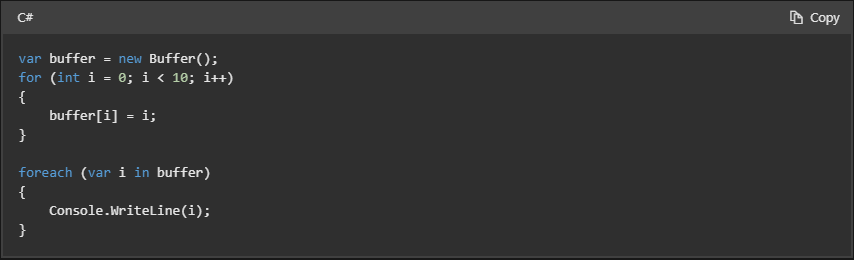
Se poate utiliza **using** pentru a alia orice tip, nu doar tipurile numite. Aceasta înseamnă că puteți crea aliasuri semantice pentru **tuple**, **tablouri**, **pointeri** sau alte tipuri nesigure.



1. *Array inline*

Vectorii **inline** sunt folosiți de echipa **runtime** și de alți autori de librarii pentru a îmbunătăți performanța aplicațiilor. Tablourile **inline** permit unui developer să creeze un vector de dimensiune fixă într-un tip **struct**. O structură cu un buffer inline ar trebui să ofere caracteristici de performanță similare cu un buffer **nesigur** de dimensiune fixă. Probabil vectori **inline** nu vor fi declarați de programator, dar acestia ii vor utiliza în mod transparent atunci când sunt expusi ca obiecte **System.Span<T>** sau **System.ReadOnlySpan<T>** din API-urile de rulare.

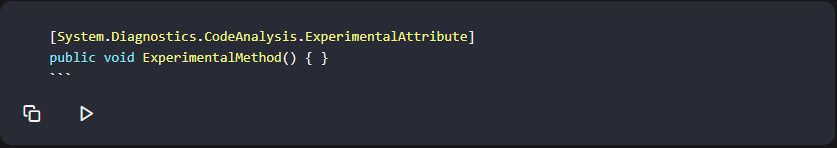




Compilatorul poate profita de informațiile cunoscute despre un vector **inline**, aceasta constituind o diferenta fata de vectorii normali.

1. *Atribute experimentale*

Tipurile, metodele sau ansamblurile pot fi marcate cu **System.Diagnostics.CodeAnalysis.ExperimentalAttribute** pentru a indica o **caracteristică experimentală**. Compilatorul emite un avertisment dacă accesați o metodă sau un tip adnotat cu **ExperimentalAttribute** . Toate tipurile incluse într-un ansamblu marcat cu atributul **Experimental** sunt experimentale.



1. *Interceptori*

**Interceptorii** sunt o caracteristică **experimentală**, disponibilă în modul de previzualizare cu C# 12. Caracteristica poate fi supusă modificărilor sau eliminării într-o versiune viitoare. Prin urmare, **nu** este recomandat pentru aplicații de producție sau lansate.

Pentru a utiliza interceptori, proiectul utilizatorului trebuie să specifice proprietatea **<InterceptorsPreviewNamespaces>**. Aceasta este o listă de spații de nume cărora li se permite să conțină interceptori.

Un interceptor este o metodă care poate substitui declarativ un apel la o metodă interceptabilă cu un apel către ea însăși în timpul compilării. Această înlocuire are loc prin faptul că interceptorul declară locațiile sursă ale apelurilor pe care le interceptează. Interceptoarele oferă o facilitate limitată de a schimba semantica codului existent prin adăugarea de cod nou la o compilare, de exemplu într-un generator de sursă.

