Функционално програмиране

Какво е функционално програмиране (ФП)?

Основни принципи:

- **Функциите са „чисти“ (pure functions):** **резултатът зависи само от входните параметри, без странични ефекти (напр. промяна на глобални променливи).**

- **Неизменяемост (immutability):** **данните не се променят, вместо това се създават нови.**

- **Функциите са „граждани от първи клас“:** **могат да се подават като аргументи, да се връщат като резултат, да се съхраняват в структури от данни.**

- **Декларативност:** **акцент върху какво искаме да получим, не върху как се изпълнява.**

Разлика с императивното програмиране

Характеристика	Императивно	Функционално
Модел	„Какво да правим стъпка по стъпка“	„Какво представлява резултатът“
Данни	Изменяеми (mutable)	Неизменяеми (immutable)
Функции	Често имат странични ефекти	Предпочитат се чисти функции
Контрол	Цикли (for, while)	Рекурсия, map, filter, reduce
Състояние	Постоянно се променя	Минимално или без промени

Къде се използва функционално програмиране?

- **Надеждност и предвидимост са важни – тъй като чистите функции са по-лесни за тестване и дебъгване.**

  Пример: банкови системи, блокчейн, телеком софтуер.

- **Работа с големи количества данни – map/reduce трансформации в big data екосистеми.**

- **Успоредна и конкурентна обработка – поради липса на странични ефекти и изменяемо състояние.**

  Пример: многопроцесорни изчисления, AI системи.

- **Езикови среди, които я насърчават – Haskell, Erlang, Elixir, Clojure, F#, но и все повече в Python, JavaScript.**

Разлики

```
using System;
using System.Collections.Generic;

class Program
{
    static void Main()
    {
        var numbers = new List<int> { 1, 2, 3, 4, 5, 6 };
        int sum = 0;

        foreach (var n in numbers)
        {
            if (n % 2 == 0)
            {
                sum += n; // променяме състоянието
            }
        }

        Console.WriteLine($"Сума (императивно): {sum}");
    }
}
```

```
using System;
using System.Linq;

class Program
{
    static void Main()
    {
        var numbers = new[] { 1, 2, 3, 4, 5, 6 };

        int sum = numbers
            .Where(n => n % 2 == 0)  // филтриране (filter)
            .Sum();                  // агрегиране (reduce)

        Console.WriteLine($"Сума (функционално): {sum}");
    }
}
```