Metody i rekurencja

Czas: 2 × 45 min | praca indywidualna | Wykład 1.09

Cel: Zbudujesz statyczną klasę `Matematyka` z metodami obliczeniowymi w wersjach iteracyjnych i rekurencyjnych oraz użyjesz parametrów `out` i `ref`.

Po zajęciach potrafisz:

- definiuje metody statyczne z parametrami i wartością zwracaną
- zapisuje algorytm rekurencyjnie i iteracyjnie
- stosuje parametry `out` i `ref`
- dokumentuje metody komentarzem `///`

Sprzęt i materiały

- komputer z Windows 11 i dostępem do internetu
- pendrive lub konto w repozytorium do zapisania projektu

Oprogramowanie

- **Visual Studio Community 2026 (obciążenie „Programowanie aplikacji klasycznych platformy .NET")** — darmowe dla uczniów i nauczycieli; w pracowni zainstalowane visualstudio.microsoft.com/pl/downloads
- **.NET 10 SDK (LTS)** — instalowany razem z Visual Studio; sprawdzenie: `dotnet --version` dotnet.microsoft.com/pl-pl/download/dotnet/10.0

Bezpieczeństwo

- Zapisuj projekt w swoim folderze (np. `C:\Users\<login>\source\repos`), a na koniec zajęć skopiuj go na pendrive lub wyślij do repozytorium.
- Nie uruchamiaj programów i pakietów z nieznanych źródeł; pakiety NuGet instaluj tylko te, które wskazuje zadanie.

Przebieg zadania

1 Klasa Matematyka

Utwórz projekt `Biblioteka`. Metody umieść w statycznej klasie pod instrukcjami najwyższego poziomu.

```
Console.WriteLine($"NWD(84, 36) = {Matematyka.Nwd(84, 36)}");

static class Matematyka
{
    /// <summary>Największy wspólny dzielnik (algorytm Euklidesa).</summary>
    public static int Nwd(int a, int b)
    {
        while (b != 0) (a, b) = (b, a % b);
        return Math.Abs(a);
    }
}
```

1. Utwórz `static class Matematyka` na końcu pliku `Program.cs` (lub w osobnym pliku `Matematyka.cs`).
2. Napisz `Nwd(int a, int b)` algorytmem Euklidesa z resztą z dzielenia i `Nww(a, b) = a / Nwd(a, b) * b`.
3. Napisz `CzyPierwsza(int n)` (dla n < 2 zwróć false).
4. Nad każdą metodą dodaj komentarz `///` z opisem (Visual Studio wstawi szablon po wpisaniu trzech ukośników).

2 Iteracja i rekurencja

Każdą z metod napisz w dwóch wersjach.

1. `Silnia(int n)` — pętlą, `SilniaRek(int n)` — rekurencyjnie; wynik typu `long`.
2. `Potega(long a, int n)` i `PotegaRek(long a, int n)`.
3. Porównaj wyniki obu wersji dla kilku danych.
4. Oblicz 20! i 21!. Co się stało? Sprawdź wynik dla `checked`.
5. Wywołaj `SilniaRek(100000)` i zapisz, jaki błąd się pojawił.

3 Parametry out i ref

Metody zwracające więcej niż jedną wartość.

1. Napisz `MinMax(int[] t, out int min, out int max)`.
2. Napisz `Podwoj(ref int x)` i pokaż wartość zmiennej przed i po wywołaniu.
3. Napisz drugą wersję `MinMax`, która zwraca krotkę `(int Min, int Max)`, i porównaj wygodę użycia.

4 Testy ręczne

Sprawdź metody na przygotowanych danych.

1. Wypełnij tabelę testów w sprawozdaniu (wyniki oczekiwane oblicz wcześniej na kartce).
2. Popraw metody, jeśli wyniki się nie zgadzają.
3. Zatwierdź pracę w Git.

W sprawozdaniu umieść

- ☐ kod klasy Matematyka z komentarzami
- ☐ tabela testów
- ☐ wyjaśnienie przepełnienia przy 21!
- ☐ opis błędu przy głębokiej rekurencji

Pytania kontrolne

1. Co oznacza słowo `static` przy klasie i metodzie?
2. Czym różni się parametr `out` od `ref`?
3. Co to jest warunek zakończenia rekurencji i co się dzieje, gdy go brakuje?
4. Kiedy rekurencja jest gorszym wyborem niż pętla?

Kryteria oceny

Kryterium	Punkty
NWD, NWW, CzyPierwsza	5
Silnia i potęga w dwóch wersjach	5
MinMax z out, Podwoj z ref, wersja z krotką	4
Komentarze /// i tabela testów	3
Pytania	3
Razem	**20**

Sprawozdanie przygotuj na wzorze dołączonym do zadania (plik .docx o tym samym numerze).