

## Область видимості (контекст) змінних

Кожна змінна доступна в рамках певного контексту або області видимості. Поза цим контекстом змінна вже не існує. Існують різні контексти:

- Контекст класу. Змінні, визначені на рівні класу, доступні в будь-якому методі цього класу
- Контекст методу. Змінні, визначені на рівні методу, є локальними і доступні тільки в рамках даного методу. В інших методах вони недоступні
- Контекст блоку коду. Змінні, визначені на рівні блоку коду, також є локальними і доступні тільки в рамках даного блоку. Поза свого блоку коду вони не доступні.

Наприклад, нехай клас Program визначений таким чином:

```
1  class Program // начало контекста класса
2  {
3      static int a = 9; // переменная уровня класса
4
5      static void Main(string[] args) // начало контекста метода Main
6      {
7          int b = a - 1; // переменная уровня метода
8
9          { // начало контекста блока кода
10
11              int c = b - 1; // переменная уровня блока кода
12
13          } // конец контекста блока кода, переменная c уничтожается
14
15          //так нельзя, переменная c определена в блоке кода
16          //Console.WriteLine(c);
17
18          //так нельзя, переменная d определена в другом методе
19          //Console.WriteLine(d);
20
21          Console.Read();
22
23      } // конец контекста метода Main, переменная b уничтожается
24
25      void Display() // начало контекста метода Display
26      {
27          // переменная a определена в контексте класса, поэтому доступна
28          int d = a + 1;
29
30      } // конец контекста метода Display, переменная d уничтожается
31
32  } // конец контекста класса, переменная a уничтожается
```

Тут визначено чотири змінних: a, b, c, d. Кожна з них існує в своєму контексті. Змінна **a** існує в контексті всього класу Program і доступна в будь-

якому місці і блоці коду в методах Main і Display.

Змінна **b** існує тільки в рамках методу Main. Також як і змінна **d** існує в рамках методу Display. У методі Main ми не можемо звернутися до змінної **d**, так як вона в іншому контексті.

Змінна **c** існує тільки в блоці коду, межами якої є що відкриваються і закриваються фігурні дужки. Поза його межами змінна **c** не існує і до неї не можна звернутися.

Нерідко кордону різних контекстів можна асоціювати з відкриваються і закриваються фігурними дужками, як в даному випадку, які задають межі блоку коду, методу, класу.

## Рекурсивні функції

Рекурсивна функція являє таку конструкцію, при якій функція викликає саму себе.

Візьмемо, наприклад, обчислення факторіала, яке використовує формулу  $n! = 1 * 2 * \dots * n$ . Наприклад, факторіал числа 5 дорівнює  $120 = 1 * 2 * 3 * 4 * 5$ .

Визначимо метод для знаходження факторіала:

```
1 static int Factorial(int x)
2 {
3     if (x == 0)
4     {
5         return 1;
6     }
7     else
8     {
9         return x * Factorial(x - 1);
10    }
11 }
```

При створенні рекурсивної функції в ній обов'язково повинен бути певний базовий варіант, який використовує оператор **return** і поміщається на початку функції. У випадку з факторіалом це **if (x == 0) return 1;**.

Іншим поширеним показовим прикладом рекурсивної функції служить функція, що обчислює числа Фібоначчі.  $n$ -й член послідовності Фібоначчі визначається за формулою:  $f(n) = f(n-1) + f(n-2)$ , причому  $f(0) = 0$ , а  $f(1) = 1$ . Тобто послідовність Фібоначчі буде виглядати так 0 (0-й член), 1 (1-й член), 1, 2, 3, 5, 8, 13, 21, 34, 55, 89, 144, .... Для визначення чисел цієї послідовності визначимо наступний метод:

```

1 static int Fibonacci(int n)
2 {
3     if (n == 0 || n == 1)
4     {
5         return n;
6     }
7     else
8     {
9         return Fibonacci(n - 1) + Fibonacci(n - 2);
10    }
11 }

```

Це найпростіші приклад рекурсивних функцій, які покликані дати розуміння роботи рекурсії.

Приклад. Описати рекурсивну функцію, яка за заданим дійсним значенням **x** та цілим значенням **n** обчислює значення величини  $x^n$  у відповідності до формули:

$$x^n = \begin{cases} 1, & \text{якщо } n = 0, \\ \frac{1}{x^{|n|}}, & \text{якщо } n < 0, \\ x \cdot x^{n-1}, & \text{якщо } n > 0. \end{cases}$$

```

static double f(double x, int n)
{
    if (n == 0)
        return 1;
    else
        if (n < 0)
            return 1 / f(x, Math.Abs(n));
        else
            return x * f(x, n - 1);
}

```