Podstawy C#

Cz.1

CENEO.pl

Wersje języka

C# 1.0

- Klasy
- Struktury
- Interfejsy
- Eventy
- Property
- Delegaty
- Atrybuty

C# 2.0

- Typy generyczne
- „yield”
- Typy częściowe
- Metody anonimowe
- Typy Nullable

C# 3.0

- Auto-property
- Wyrażenia lambda
- Proste inicjowanie obiektów
- Typy anonimowe
- LINQ (Language Integrated Query)

C# 4.0

- „dynamic”
- „ref”
- Parametry opcjonalne

C# 5.0

- „await” i „async”
- Atrybut „Caller Info”

C# 6.0

- Operator „nameof”
- Dyrektywa „using static”
- Filtrowanie wyjątków
- Inicjowanie propert
- Interpolacja Stringa
- Inicjowanie słowników i podobnych

C# 7.0

- Zmienne „out”
- Tuple i dekonstrukcja
- Pattern matching – „is”
- Funkcje lokalne (wewnątrz metod)

Konwencje

Jeden wiersz

- Instrukcja
- Deklaracja
- Pusty wiersz między definicjami metod i propert

Nazewnictwo

- Wielką literą
  - Klasy i interfejsy
  - Metody
  - Zmienne statyczne
- Małą literą
  - Zmienne (dodatkowo prywatne zaczynać od '_')
  - Parametry
- CamelCase i język angielski

Komentarze

- Najlepiej nie używać ich wcale, kod powinien komentować sam siebie, ale jeśli już trzeba, to:
  - W osobnych liniach
  - Pełnymi zdaniami
  - Po znakach „//” zostawiamy przerwę

Typy

- Wartościowe
  - Całkowite
    - **byte**, *sbyte*, **short**, *ushort*, **int**, *uint*, **long**, *ulong*
  - char
  - bool
  - Zmiennoprzecinkowe
    - float, double, decimal
- Referencyjne
  - object
  - String
  - dynamic

Operatory

- x.y
- x?y
- f(x)
- [x]
- x++ / x--
- ++x / --x
- new
- typeof
- sizeof

- await
- (T)x
- !x
- -x

- x * y
- x / y
- x % y
- x + y
- x - y
- x < y / x <= y
- x > y / x >= y
- is
- as

- x == y
- x != y
- x ?? y
- t ? x : y
- x = y
- x += y / x -= y
- x *= y / x /= y
- x %= y
- =>

Operatory

Instrukcje warunkowe

```
if (x == y)
{
    // Do something.
}
else
{
    // Do something else.
}
```

```
switch (x)
{
    case 1:
        // Do something.
        break;
    case 2:
        // Do something else.
        break;
}
```

- x == y
- t ? x : y

Instrukcje warunkowe

Pętle

```
for (int i = 0; i < x; i++)
{
    // Do something.
}
```

```
foreach (var element in collection)
{
    // Do something.
}
```

```
while (condition)
{
    // Do something.
}
```

```
do
{
    // Do something.
} while (condition);
```


Pętle

- Pętle można zagnieżdżać
- Kontrolowanie pętli
  - break
  - continue

String - operacje

- „String”, a „string”
  - string jest aliasem String

- Składanie ciągów
  - String.Concat()
  - x + y / x += y

- Parsowanie ciągów
  - int.Parse(”6”)

- Przekształcanie w ciąg
  - x.ToString()

String - operacje

Obsługa wyjątków

- blok try…catch…finally
  - wyjątek łapany jest w „catch”
  - „finally” wykona się jedynie po złapaniu wyjątku. W przypadku niezłapania wyjątku „finally” nie wykona się.

DEMO