Programování

Operátory, práce s datumem a časem, adresace

Operátor is

- Operátor kontrolující shodu datového typu
- Vhodný pro účely testování a pro přetypování
  - Porovnání datového typu s očekávaným
  - Porovnávat lze i se vzorem
- Vhodnější například pro odhalování **null** hodnot
- Null hodnotou chápeme nedefinovanou prázdnou hodnotu, která nemá konkrétní datový typ

```
class Program
{
    Počet odkazů: 0
    static void Main(string[] args)
    {
        Object o = new Person();

        if ( o is Person)
        {
            Console.WriteLine("Tohle je osoba");
        }
    }
}

Počet odkazů: 2
class Person
{

}
```

Funkce sizeof()

- Jednoduchá základní funkce pro určení velikosti konkrétního datového typu
- Vrací velikost paměti potřebnou pro uložení proměnné s určeným datovým typem
- Vhodná pro využití v místech, kde pracujeme s omezenou pamětí

```
Console.WriteLine(sizeof(int));
```

Podmíněný operátor ?:

- Občas se označuje jako ternární operátor
- Slouží pro vyhodnocení logického výrazu a vrátí jeden z možných výrazů podle toho, zda je výraz vyhodnocen jako ***true*** nebo ***false***
- Na podmíněný operátor lze pohlížet jako na expresní použití if-else bloku
- **Podmínka ? Důsledek : Alternativa**

```
int x=20;
string res;
if (x > 0)
{
    res = "kladné";
}
else
{
    res = "záporné";
}

string result =  x > 0 ? "kladné" : "záporné";
```

Kdy lze vynechat složené závorky?

- Pro přehlednost a optimalizaci si můžeme dovolit v kódu vynechávat závorky ohraničující blok kódu
- V případě, že se tělo kódu nachází na jednom řádku, můžeme závorky vynechat
- Nejčastěji můžeme na tyto případy narazit u ***if-else*** podmínek nebo u ***for*** cyklu

```
if (x == 0)
    Console.WriteLine("Nula");
else
    Console.WriteLine("Jiné číslo");
```

Ukazatele (pointers)

- Pro nalezení a použití proměnné z paměti využíváme její jméno
- C# stejně jako další jazyky však umožňují získat i konkrétní fyzickou adresu dané proměnné
- Pomocí unárních operátorů lze získat adresu proměnné a hodnotu proměnné na kterou adresa ukazuje
- **&** nám umožní získat adresu proměnné
- ***** nám umožní získat hodnotu na dané adrese
- Práce s ukazateli vyžaduje nezabezpečený kontext, klíčové slovo ***unsafe***

```
Počet odkazů: 0
unsafe class A
{
    int promenna = 5;
    int* ukazetel = &promenna;
}
```

Formátování řetězce

Interpolovaný řetězec

- Interpolovaný řetězec přímo obsahuje proměnné a výrazy
- Jednodušší na čtení
- V případě potřeby použití speciálních znaků je píšeme dvakrát, abychom je odlišili od zbytku řetězce

Kompozitní řetězec

- Řetězec a seznam objektu se používají jako argumenty metod
- Náročnější na čtení
- Důležité je pořadí seznamu objektů

interpolovaný

```
string user = "abc";
string password = "psw" ;
Console.WriteLine($"Uživatel: {user}, heslo: {password}");
```

kompozitní

```
string user = "abc";
string password = "psw" ;
Console.WriteLine("Uživatel: {0}, heslo: {1}",user,password);
```

Práce s datumem a časem

- V některých aplikacích se setkáváme s potřebou evidovat nebo zpracovávat datum a čas
- C# pro tyto účely používá struktury **DateTime** a **TimeSpan**
- **DateTime** představuje okamžitý čas v požadovaném formátu
- **TimeSpan** představuje časový interval mezi dvěma časy

```
DateTime time = new DateTime(2021,12,24,18,59,55);
DateTime now = DateTime.Now;
DateTime today = DateTime.Today;

TimeSpan interval = time-now;
Console.WriteLine(interval.Days);
```