Programovanie v jazyku C#

Úvod do jazyka C#, základy OOP

prednáška 1

Ing. Ján Magyar, PhD.

ak. rok. 2024/2025 ZS

C#

1. v roku 2002, primárne staticky typovaný OO jazyk
2. v roku 2005, podpora generického programovania
3. v roku 2007, LINQ, anonymné typy a lambda výrazy
4. v roku 2010, dynamické typy, nepovinné parametre
5. v roku 2012, zjednodušenie asynchrónnych operácií
6. v roku 2015, statické importy, read-only properties
7. v roku 2017, funkcionálne programovanie, n-tice, lokálne metódy
8. v roku 2019, nullable typy, switch, defaultné metódy rozhraní
9. v roku 2020, záznamy a top-level príkazy
10. v roku 2021, zjednodušenie syntaxe
11. v roku 2022, rozšírenie generického programovania, lepšia práca so skupinami
12. v roku 2023, primárne konštruktory, rozšírené možnosti parametrov

.NET Framework

softvérový rámec na podporu vývoja aplikácií

súčasťou Windows

dve základné časti

- Common Language Runtime
- Base Class Library

dnes v podstate iba pre Windows

viac-menej legacy

verzia od tretích strán: Mono, Xamarin, Unity

.NET Core

cielene viacplatformové riešenie

CoreCLR namiesto CLR, CoreFX namiesto BCL

častejšie aktualizácie

od verzie 5 pod menom .NET

Kompilácia a spustenie kódu

1. kompilácia
2. statická kontrola
3. transformácia do Common Intermediate Language
4. konvertovanie do strojového kódu (CLR v rámci .NET)
5. vytvorenie balíkov, resp. assembly
6. kompilátor vráti výsledok spusteného kódu

Syntax C#

príkazy a bloky

každý príkaz musí byť uzavretý `;`

bloky sa uvádzajú pomocou `{ }`

bloky sa používajú pri definícii menných priestorov, tried, vetiev, cyklov, atď.

C# berie do úvahy veľkosť písmen

Premenné

majú typ a hodnotu

```
string s1 = new string("Hello, World!");
string s2 = "Hello, World!";
var s3 = "Hello, World!";
string s4 = new("Hello, World!");
```

Argumenty

pri každom spustení programu sa vygeneruje premenná `args`

pre získanie argumentov z príkazového riadku

argumenty zadáme napríklad pri spustení pomocou:

```
dotnet run -- arg1 arg2 arg3
```

pri použití `Main` funkcie treba ju zadefinovať ako parameter:

```
static void Main(string[] args)
```

Dostupnosť premenných

členská premenná - v rámci triedy, kým je trieda dostupná

lokálna premenná - do konca bloku alebo metódy

lokálna premenná v iterácii je dostupná v rámci iterácie

v prípade nejednoznačností je možné použiť dot notation

Konštanty

premenné, hodnotu ktorých nedokážeme meniť

```
const int n = 9;
```

kompilátor nahradí všetky výskyty konštanty jej hodnotou

používa sa iba v prípade, že hodnota sa nemení ani v novších verziách programu

hodnota konštanty musí byť:

- zadaná pri deklarácii konštanty
- vypočítateľná počas kompilácie
- statická

Nullovateľné typy

najmä pri mapovaní z/do databáz

hodnotové typy:

- `int? i1 = null;`
- flag `i1.HasValue` a hodnota `i1.Value`

referenčné typy:

- kompilačné chyby namiesto run-time chýb
- nastavenie pre projekt: `Nullable` - `enable`
- `string? s1 = null;`

Základné typy

`short` (16-bitové)

`ushort` (16-bitové bez znamienka)

`int` (32-bitové)

`uint` (32-bitové bez znamienka)

`long` (64-bitové)

`ulong` (64-bitové bez znamienka)

`BigInteger` (podľa pamäte)

`Half` (16-bitové)

`float` (32-bitové)

`double` (64-bitové)

Ďalšie základné typy

`bool`

`char` (apostrof)

`string` (úvodzovky)

`object`

Práca s reťazcami

`string` je nemeniteľný typ

pre dynamické vytváranie reťazcov je potrebné použiť `StringBuilder`

interpolácia:

```
int a = 2, b = 3;
string s = $"{a} times {b} is {a * b}";
```

verbatim stringy:

```
string s = @"a tab: \t"
```

Vetvenie

```
if (podmienka) {
   telo1;
} else if (podmienka) {
   telo2;
} else {
   telo3;
}
```

Switch

```
switch (num)
{
    case 1:
        telo1;
        break;
    case 2:
    case 3:
        telo2aj3;
        break;
    default:
        telo;
        break;
}
```

Iterácie

```
for (int i = 0; i < 10; i++)
{
    telo;
}
```

```
foreach (int num in numbers)
{
    telo;
}
```

```
while (podmienka)
{
    telo;
}
```

```
do
{
    telo;
} while (podmienka);
```

Direktívy

```
#define DEBUG

// do something here
#if DEBUG
Console.WriteLine("something");
#endif

#undef DEBUG
```

Direktívy

```
#if DEBUG && RELEASE
#error "DEBUG and RELEASE cannot be true
simultaneously"
# endif


#warning "Remove before handing in"
Console.WriteLine("I hate this course");
```

Paradigmy programovania

sekvenčné programovanie

štrukturované programovanie

modulárne programovanie

objektovo orientované programovanie

otázky?