Programovací jazyky

Úvod

`Programovací jazyk` je prostředek pro zápis `příkazů`, jež mohou být poté provedeny na počítači. Souhrn takovýchto příkazů pak nazýváme `programem`. Programovací jazyky můžeme dělit do rodin podobně, jako to děláme například s jazyky lidskými. Můžeme je dělit podle jejich účelu, generace nebo na jazyky s vysokou či nízkou úrovní abstrakce. Cílem této kapitoly bude seznámit se základními vlastnostmi programovacích jazyků, vývojovým prostředím a poté již napíšeme i náš první kód.

Účel jazyka

K dnešnímu dni existuje opravdu velké množství programovacích jazyků. Kromě jazyka `C#`, kterému se budeme věnovat my můžeme vyjmenovat například JavaScript, Kotlin, Lisp nebo Swift. Nyní se možná sami sebe ptáte: „Na co jich potřebujeme takové množství?“ Na počátku vzniku nového jazyka nejčastěji stojí potřeba řešit nějaký problém. V případě jazyka JavaScript to byla například nutnost přidání logiky do jinak statických (= ne příliš interaktivních) webových stránek a jazyk Kotlin vznikl jako moderní alternativa k Javě. Ne vždy ale tato potřeba byla technického rázu. Například C# byl odpovědí Microsoftu na neustálé licenční spory s firmou Sun (později odkoupena firmou Oracle), která Microsoftu nedovolovala editovat zdrojové kódy Javy. Z tohoto důvodu bychom dnes nalezli velké podobnosti mezi oběma jazyky.

Úroveň jazyka

Jazyky můžeme dělit na `nízkoúrovňové` a `vysokoúrovňové`. Nízkoúrovňové jazyky obvykle operují na nejnižší softwarové úrovni a disponují velkou rychlostí. Daní za tuto rychlost je pak člověkem obtížná čitelnost, která prakticky znemožňuje psaní složitějšího softwaru, a tudíž klade i velké nároky na programátora. Vysokoúrovňové jazyky (někdy se též můžete setkat s pojmem jazyky s vysokou mírou `abstrakce`) jsou pak pravým opakem předešlé skupiny. Takovéto jazyky odstiňují programátora od rutinních úkolů jako je například alokace paměti nebo její další správa, což z nich dělá skvělé kandidáty na psaní rozsáhlejších aplikací, kde není na první místě výkon. Tyto pokročilé nástroje se totiž vždy negativně podepisují na celkovém výkonu jazyka.

> Nic nebývá jen černé nebo bílé a našli bychom jazyky, které se pohybují na pomezí obou výše zmíněných skupin.

>

> Pojem abstrakce v informatice označuje způsob práce s daty nebo technickými prostředky počítače, jakým by s nimi pracoval člověk. Příkladem budiž například proměnná, která je pro člověka abstraktním pojmem a každý si pod ní představíme například libovolnou hodnotu. Počítač však pojem proměnná nezná a pracuje pouze s jedničkami a nulami. Míru abstrakce tedy vždy zajišťuje daný programovací jazyk.

Interpretace a kompilace

Jazyk jako takový je pouze sbírka pravidel (specifikace), které musíme dodržovat. Pokud tak neučiníme, náš zápis není v daném jazyce validní (správný). O samotný překlad našeho kódu (například do spustitelné aplikace) se stará nejčastěji `interpreter` nebo `kompilátor`.

Kompilátor

Začneme tou jednodušší variantou. Kompilátor je program, který analyzuje náš kód, odhalí chyby a v případě, že žádné nenajde zkompiluje naší aplikaci. V případě jazyka C# výsledkem kompilace nejčastěji bývá spustitelný soubor (například `.exe`). V tento moment už kompilátoru není zapotřebí a jeho úloha zde končí. Výsledný soubor .exe je v tuto chvíli plně přenositelný například mezi zařízeními operujícími pod Windows a to i mezi těmi, které kompilátor nainstalovaný nemají.

> V případě jazyka C# je náš zdrojový kód přeložen do CIL. Jedná se o nejnižší člověkem čitelnou implementaci našeho programu. CIL se postará o správný běh našeho programu na specifickém hardwaru počítače, který se pochopitelně může lišit stroj od stroje.

Interpreter

Interpreter je na rozdíl od kompilátoru přítomen po celou dobu běhu programu. Bez interpreteru daného jazyka si v něm napsaný program nespustíme. Překlad takového programu nejčastěji probíhá příkaz po příkazu, kdy interpreter obdrží příkaz daného jazyka a poté jej rovnou přeloží na spustitelnou instrukci.

> Ve skutečnosti je vše ještě trochu složitější a našli bychom případy, kdy je využívána například i kombinace obojího. Za účelem stručnosti článku se tímto tématem nebudeme hlouběji zabývat. Kdo by měl však zájem si může jistě spoustu informací dohledat sám, neboť o této problematice byl napsán nespočet odborných publikací a článků.

Syntaxe, sémantika a konvence

Pojmem `syntaxe` rozumíme souhrn pravidel pro psaní výrazů v programovacím jazyce. Syntaxe jazyka C# nám například přikazuje, že veškeré příkazy ukončujeme znakem středníku (`;`) nebo že

názvy proměnných nesmí začínat číslovkou. `Sémantika` je pak slovník daného jazyka. Jako příklad si ukážeme funkci `Console.Write()`, jež slouží pro výpis textového řetězce (tzn. nějaké věty nebo slova) do konzole. V jazyce JavaScript bychom stejný úkol provedli příkazem `console.log()` a v Ruby za pomocí `puts`. `Konvence` je souhrn pravidel, jež bychom měli dodržovat, ale jejich porušení neohrozí běh a funkčnost programu. Příkladem budiž třeba doporučení, že náš zdrojový kód by neměl obsahovat speciální znaky jako je třeba `á`, `ü` nebo `□`.

Jazyk C# a platforma .NET

Když už známe některé druhy jazyků, jejich účel a vlastnosti, pojďme se nyní i blíže podívat na samotný C#.

Jazyk C# (vysluvujeme [sí šarp], někdy též psáno jako C Sharp) je kompilovaný, staticky typovaný vysokoúrovňový jazyk. Spadá do rozsáhlé platformy jménem `.NET` (vyslovujeme [dot net]), kterou si lze představit jako sbírku softwarových řešení, která zastřešuje například mobilní telefony, web od serverové části po klientskou, desktopové aplikace a kromě C Sharpu bychom zde našli i několik dalších programovacích jazyků.

Dříve platilo, že byl tento jazyk (včetně platformy .NET) úzce spjat s operačním systémem Windows. Dnešní .NET framework umožňuje psaní mobilních aplikací například pro operační systémy iOS, WatchOS, iPadOS, Android nebo Android Wear (Xamarin framework), psaní serverových aplikací (ASP.NET), vývoj webového frontendu (Blazor), vývoj aplikací na platformě Linux (MonoDevelop) a našli bychom toho ještě určitě víc.

K samotnému vývoji v jazyce C# nám postačí vždy nějaká konkrétní implementace frameworku .NET. Například pokud chceme vyvíjet pouze na platformě Windows, postačí nám původní verze jménem `.NET Framework`, která je vyvíjena a průběžně aktualizována již od roku 2002. Pokud nás zajímá multiplatformní vývoj nebo vývoj na webu, sáhneme po `.NET Core`. Pokud jsou našim cílem mobilní zařízení nebo vývoj her v herním enginu Unity, použijeme `MonoDevelop`. V případě, že nás zajímají univerzální aplikace pro Windows 10, framework `UWP` (Universal Windows Platform) nám poskytne potřebné nástroje.

Každá tato odnož frameworku (.NET Framework, Core, Mono a UWP) navíc vždy obsahuje kompilátor, dodatečné pomocné frameworky (např. Windows Forms nebo Xamarin) a všechny další nástroje potřebné k vývoji.

> Pokud to pro vás bylo příliš mnoho informací najednou, nemusíte si dělat starosti. Microsoft nabízí `vývojové prostředí`, které udělá všechnu náročnou práci za vás a je tedy velice snadné hned začít s psaním aplikací.

Vývojové prostředí

`Vývojové prostředí` též někdy označované jako `IDE` (integrated development environment – integrované vývojové prostředí) je chytrý editor kódu, který nám pomůže při psaní našich programů a pokud ho budeme používat efektivně, výrazně urychlí i čas vývoje. Microsoft pro vývojáře dodává hned několik variant svého vlastního řešení. Pro Windows je to `Visual Studio` (v době psaní článku Visual Studio 2019), pro Mac OS `Visual Studio for Mac` a na Linuxu můžeme využít například `MonoDevelop`. Pro všechny tyto 3 platformy je dostupné také vývojové prostředí `Visual Studio Code`, které se nejlépe hodí pro webový vývoj.

Každé toto vývojové prostředí na dané platformě nabízí trochu jiné možnosti, neboť operuje pod jinou odnoží frameworku .NET. V tomto materiálu se budeme primárně zabývat pouze konzolovými aplikacemi (tzn. aplikacemi jež ovládáme přes příkazovou řádku) a pro tyto účely je vyhovující jakékoliv výše zmíněné řešení.

> Jak již bylo avizováno výše, prostředí Visual Studio Code je vhodné spíše pro webový vývoj a jeho konfigurace pro kompilaci konzolových aplikací by mohla být pro začátečníka náročná. Doporučujeme tedy používat nejlépe standardní Visual Studio pro Windows nebo Mac.

Instalace vývojového prostředí

Nyní si ukážeme, jak probíhá instalace vývojového prostředí `Visual Studio Community 2019`. Odkaz na stažení tohoto prostředí můžete velmi snadno nalézt na oficiálních stránkách microsoftu. Po spuštění instalačního souboru se vám otevře průvodce instalací:IMG_5e39724798307.png*Úvodní obrazovka instalátoru.*

IMG_5e397251b09cf.png*Instalátor nabízí velké množství sad, které můžeme využít. Nás bude zajímat pouze "Vývoj desktopových aplikací pomocí .NET".*

IMG_5e3972608d86b.png*Tuto volbu zaškrtněte.*

U programování platí stejně jako ve válce - těžko na cvičišti, lehko na bojišti. Je moc pěkné, že si Microsoft dal tu práci a přeložil Visual Studio do češtiny, ale programátorů tím prokázal medvědí službu - když budou chtít najít pomoc na StackOverflow nebo jinde, dostanou instrukce v angličtině, které budou potom v českém prostředí horko-těžko aplikovat.

Je lepší si hned od začátku zvykat na tvrdší dosed a instalovat anglickou verzi namísto české. Hodně dobře je to vidět u pokročilejších funkcí VS, jako je práce s gitem - tam jsou překlady do češtiny naprosto kontraproduktivní.

Pro potřeby těchto skript si bohatě vystačíme pouze s balíčkem Vývoj desktopových aplikací pomocí .NET. Pokud jste spokojeni s výběrem balíčků, dokončete instalaci.

Založení nového projektu

Po instalaci a spuštění vývojového prostředí nyní zkusíme založit i náš první projekt. Z menu vybíráme možnost `Vytvořit nový projekt` a poté `Konzolová aplikace (.NET framework)`. Po potvrzení by se nám měl otevřít textový editor s kostrou naší aplikace.

> Instalace vývojového prostředí a následné založení nového projektu na systému MacOS probíhá velmi podobně. Na macu ovšem zakládáme konzolový projekt na platformě `.NET Core` (jak je vidět na posledním obrázku níže), neboť odnož platformy .NET jménem `.NET Framework` zaštiťuje pouze systém Windows. Tato problematika byla podrobněji rozepsána v předchozí sekci.

IMG_5e397327dbec3.png

IMG_5e39733d251dc.png*Poté vybíráme "Konzolová aplikace (.NET Framework)"*

IMG_5e39732e48280.png*Zadáme podrobnosti a klikáme na tlačítko "Vytvořit".*

IMG_5e39733570196.png*Takto by měl vypadat prázdný projekt.*

IMG_5e397344863c1.png*Uživatelé systému MacOS vybírají konzolovou aplikaci pod záložkou ".NET Core".*

Struktura našeho programu by nyní měla vypadat nějak takto:

```
using System;

namespace MojeAplikace
{
    class Program
    {
        public static void Main(string[] args)
        {

        }
    }
}
```

Prozatím si ukážeme pouze nutné základy a vše ostatní probereme v příslušných kapitolách tohoto materiálu. V jazyce C# má každý znak svůj smysl a význam. Jak si můžeme povšimnout, ačkoliv jsme založili úplně nový projekt, vývojové prostředí nám již předgenerovalo určitou aplikační strukturu. Složené závorky tvoří tzv. blok kódu. Tento blok začíná otevřenou složenou závorkou (`{`) a končí uzavřenou složenou závorkou (`}`) na stejné úrovni:

```
using System;

namespace MojeAplikace
{ // Začátek bloku "namespace"
    class Program
    { // Začátek bloku "class"
        public static void Main(string[] args)
        { // Začátek bloku Main

        } // Konec bloku Main
    } // Konec bloku "class"
} // Konec bloku "namespace"
```

Na ukázce výše jsme použili tzv. komentář. Cokoliv za dvěma lomítky (//) na stejném řádku bude kompilátorem ignorováno. Setkat se můžeme i s víceřádkovým komentářem:

```
using System;

namespace MojeAplikace
{
    class Program
    {
        public static void Main(string[] args)
        {
            /*
                Komentář přes více
                řádků!
            */

            // Komentář na jeden řádek
        }
    }
}
```

Prozatím si zvykneme na to, že veškerý kód budeme vždy psát do nejhlouběji zanořených složených závorek, než si postupem času odtajníme význam všech nám neznámých jazykových konstruktů. Pro kontrolu, že jsme vše nainstalovali správně si nyní zkusíme spustit tento fragment kódu poklepáním na ikonku zelené šipky v horním toolbaru. Pokud vše proběhlo bez chyb, měli bychom nyní vidět konzoli s nápisem `Ahoj světe!`. Po poklepání na libovolnou klávesu by se konzole měla zavřít.

```
using System;

namespace MojeAplikace
{
    class Program
    {
        public static void Main(string[] args)
        {
            /*
                Všechny budoucí ukázky budou počítat s tím, že kód bude vkládán
                do nejhlouběji zanořených složených závorek takto:
            */

            Console.WriteLine("Ahoj světě!");
            Console.ReadKey();
        }
    }
}
```

Shrnutí

V tomto článku jsme si představili některé základní vlastnosti programovacích jazyků, nainstalovali jsme si vývojové prostředí a napsali náš první program. V dalších sekcích se podíváme na základní jazykové kontrukce a objasníme si pojem proměnná.

Revision #4
Created 2025-05-19 09:33:02 UTC by Sklenář Maxim
Updated 2025-05-19 09:57:02 UTC by Sklenář Maxim