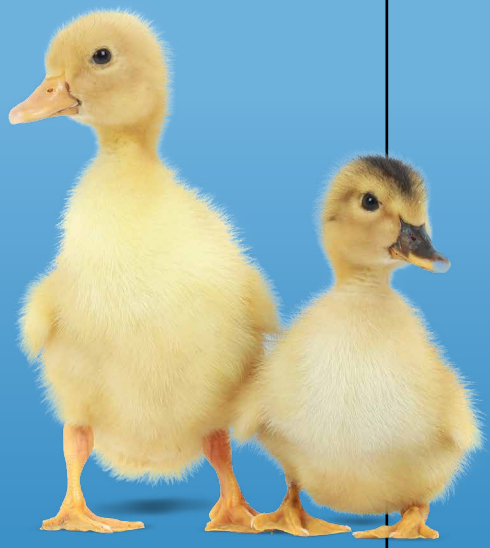
EDICE
myslíme v ...

knihovna programátora

- Přehledná a praktická učebnice C# pro začátečníky i uživatele ostatních programovacích jazyků
- Základní programovací konstrukce a potřebné pojmy (program, operační systém, programovací jazyk, objektově orientované programování a další)
- Programovací jazyk C# od základů k pokročilým konstrukcím
- Úvod do nejdůležitějších knihoven, .NET Core, verze .NET 5 a do používání vývojových nástrojů



Programování v C#

od základů k profesionálnímu použití

MIROSLAV VIRIUS

soubory ke stažení na WWW.GRADA.CZ

GRADA

EDICE
myslíme v ...

knihovna programátora

Programování v C#

od základů k profesionálnímu použití

Miroslav Virius

GRADA
Publishing

Miroslav Virius

Programování v C#

Od základů k profesionálnímu použití

Vydala Grada Publishing, a.s.
U Průhonu 22, Praha 7
obchod@grada.cz, www.grada.cz
tel.: +420 234 264 401
jako svou 7866. publikaci

Odborná korektura Jan Hájek, NETWORG
Odpovědný redaktor Petr Somogyi
Sazba Petr Somogyi
Počet stran 424
První vydání, Praha 2021
Vytiskla Tiskárna v Ráji, s.r.o., Pardubice

ISBN 978-80-271-4004-6 (ePub)
ISBN 978-80-271-4003-9 (pdf)
ISBN 978-80-271-1216-6 (print)

Obsah

Předmluva

Kniha, kterou jste právě otevřeli, vás seznámí se základy programování v jazyce C#. Je určena jak čtenářům, kteří ještě vůbec neprogramovali, tak i čtenářům, kteří již v nějakém jazyce programovali a chtějí se s tímto programovacím jazykem seznámit. Přitom předpokládám, že jste sice začátečníky v programování, ale umíte s počítačem zacházet uživatelsky, tedy že umíte spustit program, zkopírovat nebo smazat soubor atd. V průběhu osmnácti kapitol této knihy se vás pokusím dovést na úroveň lehce pokročilého programátora.

Programovací jazyk C#

Programovací jazyk C# uveřejnila firma Microsoft v roce 2002. I když byl původně určen pouze k programování pro Windows, v současné době se používá k vytváření aplikací i pro řadu dalších platforem, jako je Linux, macOS, Android a další a lze v něm vytvářet cloudové aplikace.

Co v této knize najdete

Jediným způsobem, jak se naučit nějaký programovací jazyk, je psát v něm programy, a proto už ve druhé kapitole začneme programovat. Cílem druhé, třetí a čtvrté kapitoly je uvést na scénu vybrané konstrukce jazyka C# a umožnit vám zkoušet si vše, o čem si v následujících kapitolách povíme, na vlastních programech – jednoduchých a možná nešikovných, ale fungujících.

Než se ovšem pustíme do programování, musíme si ujasnit některé pojmy a sjednotit terminologii. Proto se v první kapitole seznámíme s pojmy, které budeme používat. Povíme si, co je třeba vědět o počítači, o programovacích jazycích, algoritmech, objektově orientovaném programování a některých dalších věcech, bez nichž se při programování neobejdeme.

Ve druhé kapitole napíšeme svůj první program, ukážeme si, jak ho přeložit – neboť jazyk, v němž ho píšeme, je jiný než jazyk, v němž „myslí" počítač – a jak ho spustit. Seznámíme se s integrovaným vývojovým prostředím Visual Studio 2019, které budeme v této knize používat.

Ve třetí kapitole napíšeme několik verzí jednoduchých programů, které přečtou číslo, vypočtou z něj jistou hodnotu a výsledek vypíší. To nám umožní seznámit se alespoň povrchně s některými základními programovacími konstrukcemi, s rozdělením programu do několika souborů, s pojmem projektu apod. Napíšeme také jednoduchou vlastní programovou knihovnu pro čtení dat z klávesnice. Se znalostmi z této kapitoly byste měli být schopni zkoušet si vše, co se v dalších kapitolách naučíte, na vlastních programech.

Ve čtvrté kapitole se seznámíme s kolekcemi – nástroji, které umožňují uchovávat v programu libovolné množství dat –, seznámíme se velmi povrchně s generickými třídami a ukážeme si, jak ladit program – tedy jak zjistit, proč dělá něco jiného, než co jsme chtěli naprogramovat.

Od páté kapitoly začíná výklad „naostro". Po seznámení se základními pojmy postupně poznáme vestavěné datové typy, příkazy jazyka C#, s výrazy a operátory, které máme k dispozici, naučíme se používat tzv. pole. Pak přijdou na řadu kapitoly věnované programátorem definovaným negenerickým objektovým typům, rozhraním, genericitě, delegátům a lambda-výrazům (to vše jsou zajímavé a užitečné konstrukce, které jazyk C# nabízí) a tzv. výjimkám (nástrojům pro ošetřování chyb za běhu). Samostatná kapitola je věnována tzv. nulovatelným typům. Nelze samozřejmě pominout nástroj označovaný zkratkou LINQ, který umožňuje získávat vybraná data z různých zdrojů – mohou to být už zmíněné kolekce, soubory a mnohé další.

Pak přijdou na řadu kapitoly věnované použití vybraných programových knihoven. Začneme nástroji pro práci se znakovými řetězci, tedy s textem. Pak se podíváme na vstupní a výstupní operace, na vytváření grafického uživatelského rozhraní pomocí knihovny Windows Forms a pomocí knihovny Windows Presentation Foundation a na nástroje pro paralelní počítání.

Na co se nedostalo

Víc se mi bohužel do této knihy nevešlo. Nedostalo se na tvorbu databázových aplikací, na tvorbu webových aplikací, na tvorbu mobilních aplikací, na práci s datovými typy, které nebyly v době překladu k dispozici (tzv. reflexi), na použití nespravovaného kódu a dynamických knihoven napsaných v jiných programovacích jazycích, na práci s místním nastavením (kulturou) a na mnoho dalších zajímavých a užitečných témat.

Nástroje

Výklad v této knize je založen na jazyce C# verze 8 publikované v roce 2019. Pro psaní, překlad, sestavení a ladění programů používám vývojové prostředí Visual Studio 2019 Community Edition, které lze zdarma získat na stránkách firmy Microsoft.

Není to ovšem jediná možnost: Dobrou alternativou je nástroj Visual Studio Code, který je také zdarma a který lze instalovat pod operačními systémy Windows 7, 8 a 10, Debian, Ubuntu, Red Hat, Fedora, Suse a macOS 10.10 a pozdějších. Jeho ovládání se poněkud liší od ovládání Visual Studia; to ale nic nemění na platnosti výkladu o jazyce C# a jeho knihovnách, které v této knize najdete.

Terminologie

V celé knize používám důsledně českou terminologii. Jsem přesvědčen, že použití vhodných českých názvů výrazně usnadní pochopení, oč jde. Anglické termíny samozřejmě uvádím alespoň při prvním výskytu také.

Příklady

Výklad v této knize doprovází velké množství příkladů. Jejich zdrojové texty si můžete stáhnout z webových stránek nakladatelství Grada Publishing (vyhledejte na *www.grada.cz* stránku této knihy), nebo z mých osobních stránek, jejichž adresu najdete dále. V příkladech důsledně používám české identifikátory, a to včetně háčků a čárek. Vím, že profesionální programátoři se tomu vyhýbají a že v mezinárodních týmech jsou anglické identifikátory samozřejmostí. Mám ale dlouholetou zkušenost, že při výkladu určeném začátečníkům mohou české identifikátory výrazně usnadnit orientaci v ukázkách zdrojového kódu – a to je můj hlavní cíl.

Na závěr

I přes veškerou péči, kterou jsem této knize věnoval, se do ní mohly vloudit chyby. Jestliže při studiu nějakou najdete, pošlete mi prosím zprávu na níže uvedenou adresu; bude-li to možné, uveřejním na svých webových stránkách opravu.

Miroslav Virius, Praha, 21. srpna 2020
miroslav.virius@fjfi.cvut.cz
http://people.fjfi.cvut.cz/virius

1 Než opravdu začneme

Než začneme programovat, měli bychom se seznámit s pojmy, které budeme později potřebovat. Povíme si krátce něco o nejen o počítačích a programovacích jazycích, ale také o algoritmech a objektově orientovaném programování. Mnohé z toho jistě znáte; přesto vám doporučuji tuto kapitolu alespoň zběžně přečíst, abychom si sjednotili terminologii.

1.1 Počítač

Kdesi – myslím, že šlo o návod, jak vyplnit daňové přiznání – jsem četl, že počítač je „stroj na zpracování informací". To je samozřejmě pravda, ale člověk, který se chce naučit programovat, o něm musí vědět přece jen víc.

S trochou zjednodušení můžeme říci, že běžný počítač se skládá ze čtyř základních částí:

- Součást, která opravdu „počítá", tedy zpracovává informace, a která také řídí činnost všech ostatních částí počítače, se nazývá *procesor*.
- *Operační paměť* slouží k přechodnému ukládání dat (informací), která počítač zpracovává, a programu, tedy příkazů, které určují, co má dělat. Používá se pro ni také anglická zkratka *RAM* (*Random Access Memory*, doslova „paměť s náhodným přístupem"). Vše, co je v této paměti, se při vypnutí počítače ztratí, „zapomene".
- *Vstupní a výstupní* zařízení slouží počítači k výměně informací s okolím. Nejběžnější vstupní zařízení jsou klávesnice, myš, skener, zařízení na čtení CD/DVD atd. Nejobvyklejší výstupní zařízení osobních počítačů je obrazovka monitoru nebo tiskárna. Pro vstupní a výstupní zařízení se používá zkratka V/V nebo I/O (z anglického input/output).
- Trvalá paměť slouží – jak název napovídá – k trvalému ukládání dat a programů; data v ní se při vypnutí počítače neztrácejí. Má zpravidla mnohonásobně větší kapacitu než operační paměť, ale práce s ní je také mnohonásobně pomalejší než práce s operační pamětí (přibližně tisíckrát). Jako vnější paměť se používají převážně magnetické disky (tzv. pevný disk) nebo polovodičové disky (obvykle označovaný zkratkou SSD z anglického *Solid State Drive*). Vedle trvalé paměti napevno zabudované v počítači se často používá i externí trvalá paměť, kterou k počítači připojujeme pouze v případě potřeby, pro účely zálohování dat atd.
- Součást, jež propojuje všechny ostatní části počítače, se nazývá *sběrnice* (anglicky *bus*).

1.2 Operační paměť

Bity

Operační paměť je tvořena elektronickými obvody, které mohou mít dva dobře rozlišitelné stavy – např. vypnuto nebo zapnuto. Jeden z těchto stavů obvykle odpovídá číslici 0, druhý číslici 1. Jakékoli údaje do paměti proto můžeme zapisovat jen pomocí nul a jedniček, v tzv.

dvojkové soustavě. Každé místo, na které můžeme zapsat jednu číslici 0 nebo 1, označujeme jako *bit*. Operační paměť je tedy dlouhá řada bitů.

Bajty

S jednotlivými bity však pracujeme jen výjimečně, protož je to nepohodlné, a to jak pro člověka, tak pro počítač. Bity v operační paměti se v dnešních počítačích sdružují do skupin velikosti 8 bitů a tyto skupiny se nazývají *bajty* (anglicky *byte*, tj. slabika).

Adresa

Aby mohl počítač s pamětí snadno pracovat, jsou jednotlivé bajty, které paměť tvoří, očíslovány. Počáteční bajt má číslo 0, následující má číslo 1 atd. Toto pořadové číslo se nazývá *adresa* bajtu. (Na některých počítačích, včetně PC, je záležitost s adresami ve skutečnosti trochu složitější, ale to můžeme v souvislosti s jazykem C# ponechat stranou. Pro pochopení dalšího výkladu stačí, co jsme si řekli.)

1.3 Datové typy a proměnné

Není těžké zjistit, že nejmenší číslo, které může jeden bajt obsahovat, se skládá z osmi nul a představuje ve dvojkové i v desítkové soustavě nulu. Největší takové číslo se bude skládat z osmi jedniček a v desítkové soustavě vyjadřuje 255. To je pro téměř jakékoli počítání málo; v běžných programech se obvykle objevují daleko větší čísla. Proto se pro ukládání dat zpravidla používají různě velké skupiny za sebou následujících bajtů.

Ani to ovšem nestačí. Vezmeme-li např. dva za sebou následující bajty, můžeme do nich uložit celá čísla v rozmezí od 0 do 65 535 (od 0 do 1111 1111 1111 1111 ve dvojkové soustavě). Pokud nám ani to nevystačí, můžeme vzít skupinu 4 nebo třeba 8 bajtů. Jenže ani to neřeší problém, co dělat, když budeme potřebovat záporná čísla, reálná čísla, znaky nebo třeba logické hodnoty (jež vyjadřují, že nějaké tvrzení platí nebo neplatí).

Musíme tedy najít nějaký způsob, který nám umožní reprezentovat data různých „druhů" v paměti počítače. Jinými slovy, musíme najít způsob, jakým určité hodnotě jednoznačně přiřadíme skupinu bitů, která bude tuto hodnotu představovat – jak tuto hodnotu v počítači *zakódovat*.

Datový typ

Můžeme se např. dohodnout, že bajt s hodnotou 65 bude představovat znak `A`. Táž skupina bitů může za jiných okolností také představovat celé číslo, které má v desítkové soustavě hodnotu 65. Bajt se stejnou hodnotou ale může být i součástí většího celku s naprosto jiným významem.

Odtud je vidět, že počítači nestačí znát adresu bajtu nebo bajtů, s nimiž pracuje. Vedle adresy musí vědět, jak velký úsek – kolik bajtů – má vzít a jak má jeho obsah interpretovat. Jinými slovy, musí znát *datový typ* hodnoty, která je tam uložena – musí vědět, zda jde o celé číslo, znak, logickou hodnotu atd.

Datový typ také určuje operace, které lze s danou hodnotou provádět. Celá čísla lze například sčítat a odečítat, znaky lze spojovat do *řetězců*, tedy do souvislého textu.

Proměnná

Až dosud jsme se hodnotami, uloženými v paměti, zabývali z hlediska počítače; nyní se na ně podívejme z hlediska programátora. Hodnotu, s níž budeme chtít ve svém programu pracovat, potřebujeme uložit do paměti: Musíme si vyhradit místo a říci, jakého typu budou údaje, které bude obsahovat. Takovéto místo pro ukládání hodnoty budeme nazývat *proměnná*.

Počítač bude s proměnnou zacházet pomocí její adresy (pořadového čísla jejího počátečního bajtu). Pro programátora by ale práce s adresou byla nepohodlná, a proto ji pojmenujeme, dáme jí *identifikátor*. Tomu se v programování říká *deklarace proměnné*.

1.4 Programy a programovací jazyky

Budeme-li od počítače chtít, aby zpracovával data, která mu předložíme, musíme mu také říci, co s nimi má vlastně dělat – musíme mu dát *program*. Program, podobně jako data, uložíme do operační paměti.

Procesor umí s daty provádět různé jednoduché operace. Umí např. sečíst, odečíst, vynásobit nebo vydělit dvě čísla, zadáme-li mu jejich adresy, umí vzít znak a zobrazit ho na monitoru atd. Protože však do jeho paměti nelze uložit nic jiného než čísla, musí být tyto příkazy vyjádřeny – zakódovány – také čísly. Číselné vyjádření instrukcí (příkazů) pro procesor se nazývá *strojový kód* (v programátorštině *stroják*) a je to jediná věc, které procesor rozumí.

Aby nebyl život příliš jednoduchý, používají různé procesory různé strojové kódy, takže programy ve strojovém kódu nejsou obvykle přenositelné mezi počítači s různými procesory.

Vyšší programovací jazyky

Je asi jasné, že programování ve strojovém kódu je velice namáhavé a nepřehledné. Také to téměř nikdo nedělá; místo toho se používají tzv. vyšší programovací jazyky, jako je Ada, Basic, C, C++, … a také C#.

Program ve vyšším programovacím jazyce je textový soubor, který obsahuje popis řešené úlohy vyjádřený pomocí vybraných anglických slov a pomocí výrazů zapsaných podobně jako v matematice. Napsat program ve vyšším programovacím jazyce je samozřejmě daleko snazší než napsat odpovídající program ve strojovém kódu. Ovšem nic není zadarmo: Program ve vyšším programovacím jazyce nelze na počítači přímo spustit, neboť počítač mu nerozumí. Takovýto program se proto musí buď *přeložit* do strojového kódu, nebo *interpretovat*. V obou případech k tomu potřebujeme další program, který to za nás udělá.

Překlad

Textový soubor, který obsahuje zápis programu ve vyšším programovacím jazyce, se zpravidla nazývá *zdrojový kód* nebo *zdrojový program*, v programátorské hantýrce *„zdroják"*. K *překladu* do strojového kódu (hovoříme také o *kompilaci*), slouží program zvaný *překladač* neboli *kompilátor*. Často s ním spolupracuje ještě *sestavovací program* neboli *linker*, který dokáže spojit několik nezávisle přeložených částí programu do jednoho celku. Linker také připojí knihovny – části programu, které už někdo naprogramoval předem a které můžeme už jen používat. Dnes je však sestavovací program zpravidla součástí překladače.

Překladem (a následným sestavením) programu vznikne soubor obsahující strojový kód, který již lze na cílovém počítači spustit. Mezi typické překládané programovací jazyky patří např. C, C++ nebo Pascal.

Interpretace

Místo překladu však můžeme v některých případech použít speciální program, který bude číst zdrojový text a interpretovat ho, tj. provádět příkazy, které v něm najde. Typickým interpretovaným jazykem je klasický Basic.[1] Interpretované programy obvykle běží podstatně pomaleji

1 To se netýká současných verzí Visual Basicu; s programem v tomto jazyce se zachází stejně jako s programem v C#, jak o tom budeme hovořit dále.

než překládané programy. Vedle toho musíme na cílový počítač spolu s naším programem instalovat také interpretační program.

Jazyk C#

C# je tak trochu zvláštní případ. Zdrojový kód programu napsaného v tomto jazyce se přeloží, ovšem nikoli do strojového kódu počítače, ale do univerzálního pomocného jazyka označovaného *Common Intermediate Language* (*CIL*, dříve označovaný *Microsoft Intermediate Language*, *MSIL*, nebo jen IL). Ten se pomocí dalšího překladače převede do strojového kódu cílového počítače – ale zpravidla až v okamžiku, kdy program spustíme.

To znamená, že na cílovém počítači musí být překladač, který překládá z IL do strojového kódu. Tento počítač se nazývá JIT (z anglického *Just In Time*, neboť překládá právě v době, kdy je to potřeba). Věc je ale ještě složitější: Na cílovém počítači musí instalováno prostředí .NET, které kromě překladačů JIT obsahuje i další součásti potřebné pro běh programů vytvořených v jazyce C#.

Na závěr ještě doplníme, že překladač libovolného programovacího jazyka (i C#) zároveň kontroluje syntaktickou správnost programu – tedy zda je program napsán podle jistých formálních pravidel, která zaručují, že mu počítač porozumí. (Syntaktická správnost programu bohužel nezaručuje věcnou správnost programu, tj. nezaručuje, že program bude dělat to, co si přejeme.)

1.5 Operační systém

Počítač bez programového vybavení by nám nebyl mnoho platný. Proto se s ním zpravidla dodává alespoň jeden základní program, který se rozeběhne automaticky hned po spuštění počítače a běží po celou dobu jeho provozu. Tento program se nazývá *operační systém* a „oživuje" počítač, tj. přijímá pokyny uživatele, stará se o jejich provedení a informuje uživatele o výsledcích. Pokyny mohou být vyjádřeny příkazem zapsaným v příkazové řádce, kliknutím myší na ikoně nebo jiným způsobem.

Operační systém má ovšem ještě řadu dalších úloh, z nichž pro nás nejdůležitější je, že poskytuje služby dalším programům. Stará se o jejich spouštění, o přidělování paměti, poskytuje nástroje pro práci se soubory atd. Když např. chceme v programu otevřít soubor, program předá odpovídající požadavek operačnímu systému a ten se postará o vše potřebné.

Mezi nejznámější operační systémy na osobních počítačích patří různé verze Windows a Linuxu, macOS, IOS a Android. Poznamenejme, že pro všechny tyto systémy lze v jazyce C# vytvářet programy, to znamená, že je pro ně k dispozici prostředí .NET.

1.6 Program a algoritmus

Jazyk C#, podobně jako ostatní programovací jazyky, slouží k zápisu programu. Už víme, že program je nějaký soubor instrukcí (příkazů), které počítači říkají, co má dělat.

Každý program představuje návod k řešení nějakého problému. Při programování si ale musíme uvědomit, že počítač neumí najít způsob, jak problém vyřešit: To musíme udělat my. Když už známe cestu, jak najít řešení, můžeme to naučit také počítač (musíme mu napsat program). Počítač umí jen to, co ho my nebo někdo jiný naučí, na co má program.