**Základy programování a algoritmizace**

**Proměnné**

**Erik Král**



**2020**

**Informace o autorech:**

Ing. et Ing. Erik Král, Ph.D.

Univerzita Tomáše Bati ve Zlíně

Fakulta aplikované informatiky

Nad Stráněmi 4511

760 05 Zlín

ekral@utb.cz

OBSAH

[OBSAH 3](#_Toc60865304)

[1 Úvod 4](#_Toc60865305)

[1.1 Proměnná 4](#_Toc60865306)

[1.2 Nejjednodušší program 5](#_Toc60865307)

[1.3 Identifikátor proměnné 7](#_Toc60865308)

[1.4 Struktura programu 8](#_Toc60865309)

[1.5 Proměnná z hlediska paměti 9](#_Toc60865310)

[1.6 Základní typy proměnných 13](#_Toc60865311)

[1.7 Velikost typů v paměti 14](#_Toc60865312)

[1.8 Rozsah platnosti proměnné 15](#_Toc60865313)

[Seznam použité literatury 18](#_Toc60865314)

[Seznam obrázků 19](#_Toc60865315)

# Úvod

V tomto materiálu se seznámíme s proměnnou, rozsahem její platnosti, základními typy proměnných a s pojmy výraz, příkaz a blok příkazů.

## Proměnná

Jen málo smysluplných programů může fungovat bez ukládání hodnot do paměti RAM. Těmto hodnotám v paměti říkáme proměnné.

**Proměnná** je pojmenovaná hodnota v paměti interpretovaná podle konkrétního datového typu.

**Hodnota** je množina bitů interpretovaná podle konkrétního datového typu.

**Datový typ** objektu vymezuje operace, které lze s tímto objektem provádět a množinu hodnot, kterých může objekt nabývat.

Například příkaz:

int x = 5;

rezervuje místo v paměti pro celé číslo, uloží tam hodnotu 5 jako množinu bitů a této hodnotě v paměti potom říkáme *x*. Typ int potom určuje, jaké operace (sčítání, odčítaní atd.) s proměnnou *x* může program provádět.

Hodnotu proměnné získáme použitím jejího jména, například následující příklad vypíše na konzoli hodnotu proměnné x:

System.Console.WriteLine(x);

Hodnotu proměnné změníme přiřazením hodnoty pomocí operátoru přiřazení, například následující příklad přiřadí proměnné x hodnotu 1:

x = 1;

## Nejjednodušší program

Pojďme se teď podívat na komplexnější příklad a rozebrat si jeho jednotlivé části:

using System;

namespace Property

{

    class Program

    {

        static void Main(string[] args)

        {

            int x = 5;

            Console.WriteLine(x);

        }

    }

}

První príkaz:

using System;

slouží k tomu, aby nebylo nutné zadávat plně kvalifikované jmeno včetně názvu jmenného prostoru (namespace).

Místo tohoto delšího zápisu:

System.Console.WriteLine(x);

pak můžeme použí kratší zápis:

Console.WriteLine(x);

Druhý příkaz definuje jmenný prostor. Jmenný prostor používáme proto aby nedocházelo ke konfliktu názvů mezi jednotlivými kódy, typicky v knihovnách. Pokud by někdo nazval třídu stejným názvem, tak plně kvalifikované jméno třídy obsahuje i název jmenného prostoru.

namespace Property

{

}

Další příkaz definuje třídu, třída je uživatelský typ, který může obsahovat členské prvky jako fieldy, property a metody. Třídy probereme podrobně později.

class Program

{

}

Třída *Program* obsahuje jednu statickou metodu. Metoda představuje blok kódu, který můžeme v program opakovaně spouštět. Metoda můžeme mít argumenty. Metody probere později. Metoda *Main* může být v jazyce C# jen jednou a je to první metoda, která se pouští v programu.

static void Main(string[] args)

{

}

Metoda Main potom obsahuje dva příkazy. První příkaz nadefinuje proměnnou *x* typu *int* a přiřadí ji hodnotu *5*.

int x = 5;

V paměti RAM se tedy rezervuje místo pro celé číslo a na toto místo se uloží hodnota 5 zakódovaná v jedničkách a nulách.

A druhý příkaz vypíše hodnotu této proměnné *x* na terminál. Díky použití using System; můžeme použít zkrácený zápis a napsat:

Console.WriteLine(x);

Teprve až od C# 9 a .NET 5 je možné použít Top Level Statements [1] a není nutné mít v programu metodu *Main* a zápis celého předcházející program může být mnohem jednodušší:

using System;

int x = 5;

Console.WriteLine(x);

## Identifikátor proměnné

U identifikátoru proměnné, tedy jejího názvu, se rozlišuje mezi velkými a malými písmeny. Tedy int x; a int X; jsou jiné proměnné.

U názvů proměnných, i když je to možné, nepoužíváme diakritiku, především kvůli možné spolupráci s vývojáři, kteří by potom mohli být problém s pochopením našeho textu. Například následující název proměnné je neplatný:

int poloměr;

Název proměnné nesmí obsahovat mezery:

int polomer kruhu;

Název proměnné dále nesmí začínat číslicí:

int 1polomer;

Název proměnné nesmí být totožný s klíčovým slovem nebo spojením klíčových slov jazyka C, například:

int return;

Název proměnné se nesmí shodovat s názvem jiné proměnné, která byla ve zdrojovém kódu jazyka C již dříve definovaná ve stejném rozsahu platnosti.

Co se týče jmenných konvencí, tak vycházíme z dokumentace jazyka C# a z pravidel týmu nebo projektu na kterém pracujeme. Zatím nám stačí pravidlo, že názvy lokálních proměnných začínají malým písmenem a využívají Lower camel case notation. Napřílad:

int polomerKruhu;

## Struktura programu

Na obrázku 1 jsou popsány základní stavební bloky programu. Konkrétně se program může skládat z výrazů, příkazů a bloku příkazů.

**Příkaz** představuje spustitelnou část programu, pokud se provede jeden příkaz, tak se vyhodnotí všechny výrazy v něm. Příkaz musí být v jazyce C# ukončený středníkem. V některých jazycích, například v jazyce Python se příkaz středníkem neukončuje, ale v jazyce C# a dalších jazycích vyházejících ze syntaxe jazyka C představuje vynecházní středníku na konci příkazu jednu z nejčastějších chyb začátečníků.

Příkaz se může skládat z **výrazů**, ty se vyhodnotí teprve až se provede příkaz, ve kterém jsou použité.

A nakonec máme **blok příkazů**, tedy více příkazů v jednom. V bloku příkazů se jednotlivé příkazy provádějí sekvenčně shora dolů. Blok příkazů je v jazyce C# označený složenými závorkami. V některých jazycích, například v jazyce Python je blok definovaný odsazením textu. V jazyce C# na odsazení textu nezáleží a používá se jen pro přehlednější kód.



Obrázek Příkazy a výrazy

## Proměnná z hlediska paměti

Již víme, že proměnná je pojmenovaná hodnota v paměti interpretovaná podle konkrétního datového typu. Projděme si teď jednoduchý příklad z hlediska paměti RAM.

Následující program definuje nejprve proměnnou x, přířadí ji hodnotu, pak hodnotu změní. Poté definuje druhou proměnou s názvem *y* a přířadí ji hodnotu proměnné *y*.

int x = 0;

x = 3;

int y = 0;

y = x;

Nyní si postupně projdeme jednotlivé příkazy a zároveň si je zobrazíme z hlediska paměti RAM.

**První příkaz** definuje proměnnou *x* typu celého čísla a přiřadí ji hodnotu *0* pomocí operátoru přířazení. Pokud proměnné přiřadíme hodnotu hned při definici, tak říkáme, že jsme tuto proměnnou inicializovali.

Na místo v paměti, kterému říkáme *x*, například na adresu *1000*, se uloží hodnota *0*. Protože typ *int*, tedy celé číslo, zabere v paměti 4 bajty, tak proměnná x zabírá v paměti přesně tolik bajtů. Adresa v příkladu je jen demonstrativní, v reálném programu budou adresy jiné.

int x = 0;

RAM

0

x

adresa 1000

**Druhý příkaz** poté změní pomocí operátoru přiřazení hodnotu proměnné *x* na *3*:

x = 3;

RAM

3

x

adresa 1000

**Třetí příkaz** definuje novou proměnou s názvem *y* a přiřadí jí hodnotu *0*. V paměti se tedy rezervuje nové místo pro typ *int*, tedy celé číslo.

int y = 0;

RAM

3

x

adresa 1000

0

y

adresa 1004

A konečně **poslední příkaz** přiřadí proměnné *y* hodnotu proměnné *x*. Prakticky to znamená, že program zkopíruje hodnotu paměti, kterému říkáme *x* a nahradí jím hodnotu v paměti které říkáme *y*.

y = x;

RAM

3

x

adresa 1000

3

y

adresa 1004

## Základní typy proměnných

V této kapitole si projdeme nejběžnější datové typy v jazyce C#.

**Nejpoužívanější datové typy**

Jeden z nejpoužívanějších datových tyůp, se kterým jsme se už v předchozích příkladech seznámili, je typ *int*, který reprezentuje celé číslo se znaménkem a má dostatečný rozsah pro většinu programů.

int celeCislo = 0;

Druhý častý datový typ reprezentuje desetinné číslo se znaménkem, má dvojitou přesnost dostatečnou pro většinu běžných programů.

double desetinne = 0.1;

Další typ představuje booleanovskou proměnnou a nabývá dvou stavů *true*, nebo *false*. Používá se například u podmíněných příkazů a cyklů.

bool logicka = false;

**Další často používané numerické typy**

Typ byte představuje jeden byte bez znaménka, má hodnoty *0* až *255* a je vhodný například pro čtení ze sériového portu.

byte jedenByte = 1;

*Float* představuje desetinné číslo se znaménkem a nižší přesností než *double* a zabere méně místa než *double*. Používá se například v počítačové grafice, protože výpočet s nižší přesností je rychlejší. U numerické konstany typu *float* je nutné použít sufix *f*.

float position = 1.2f;

Typ decimal není běžný typ, ale má své použítí při výpočet s penězi a financemi. Je přesnější, než *double* ale má menší rozsah [2]. Na rozdíl od typu *double* nebo *float* mohou být desetinná čísla jako například *0,1* reprezentovány v typu *decimal* přesně. Taková čísla představují často v typu *double* nebo *float* čísla s nekonečným desetinným rozvojem a mohou způsobovat chybu ze zaokrohlování [3]. Numerická konstanta typu decimal má sufix *m*.

decimal money = 0.1m;

## Velikost typů v paměti

**Velikost typů v paměti**

Kolik ale zabírají tyto typy v paměti? Typy *int* a *float* zabírají 4 bajty, typ *double* 8 bajtů, typ *decimal* 16 bajtů, a nakonec typy *bool* a *byte* jen jeden bajt.

Následující program využívá klíčové slovo *sizeof*, které vrací velikost typu v bajtech a vypisuje jej na konzoli:

using System;

namespace CviceniZap

{

    class Program

    {

        static void Main(string[] args)

        {

            Console.WriteLine(sizeof(bool));

            Console.WriteLine(sizeof(byte));

            Console.WriteLine(sizeof(int));

            Console.WriteLine(sizeof(float));

            Console.WriteLine(sizeof(double));

            Console.WriteLine(sizeof(decimal));

        }

    }

}

## Rozsah platnosti proměnné

Lokální proměnná existuje od místa, kde je nadefinovaná do konce bloku ohraničeného složenými závorkami. Proměnná je přístupná i ve všech vnořených blocích.

V následujícím kódy tedy proměnná *x* existuje od své definice po konec bloku ohraničený složenými závorkami, ve kterém byla nadefinována.

using System;

namespace CviceniZap

{

    class Program

    {

        static void Main(string[] args)

        {

            int x = 1;

            Console.WriteLine(x);

        }

    }

}

Proměnnou, která nebyla ještě nadefinována nemůžeme používat. Například následující pokus vypsat ještě nenadefinovanou proměnnou *x* by představoval chybu.

using System;

namespace CviceniZap

{

    class Program

    {

        static void Main(string[] args)

        {

            Console.WriteLine(x);

            int x = 1;

        }

    }

}

Proměnná existuje vždy jen v bloku, ve kterém byla nadefinována, například v následujícím příkladu existuje proměnná *x* jen do konce podmíněného příkazu. Pokus o výpis hodnoty proměnné *x* mimo podmíněný příkaz *if* je opět neplatný. Vlastní podmíněný příkaz probereme až později.

using System;

namespace CviceniZap

{

    class Program

    {

        static void Main(string[] args)

        {

            if(true)

            {

                int x = 1;

                Console.WriteLine(x);

            }

            Console.WriteLine(x);

        }

    }

}

Proměná je přístupná i v bloku, který je vnořený do bloku ve kterém byla definována. Například v následujícím příkladu je proměnná *x* nadefinovaná již před podmíněným příkazem a můžeme ji tedy vypsat jak uvnitř bloku podmíněného příkazu, tak i za ním.

using System;

namespace CviceniZap

{

    class Program

    {

        static void Main(string[] args)

        {

            int x = 1;

            if(true)

            {

                Console.WriteLine(x);

            }

            Console.WriteLine(x);

        }

    }

}

Seznam použité literatury

1. What's new in C# 9.0 - C# Guide | Microsoft Docs. [online]. Copyright © Microsoft 2020 [cit. 22.12.2020]. Dostupné z: <https://docs.microsoft.com/en-us/dotnet/csharp/whats-new/csharp-9#top-level-statements>
2. Floating-point numeric types - C# reference | Microsoft Docs. [online]. Copyright © Microsoft 2020 [cit. 01.01.2021]. Dostupné z: <https://docs.microsoft.com/en-us/dotnet/csharp/language-reference/builtin-types/floating-point-numeric-types>
3. Types - C# language specification | Microsoft Docs. [online]. Copyright © Microsoft 2020 [cit. 01.01.2021]. Dostupné z: https://docs.microsoft.com/en-us/dotnet/csharp/language-reference/language-specification/types#the-decimal-type

Seznam obrázků

[Obrázek 1 Příkazy a výrazy 7](#_Toc60221099)