Proměnné a konstanty

Úvod

Pojem `proměnná` (`variable`) už známe z matematiky (především z nauky o funkcích). V programování se budeme s proměnnými setkávat neustále. Co to tedy proměnná vlastně je

Vymezení pojmu

Proměnné nám umožňují uchovávat libovolnou hodnotu pod námi zvoleným názvem. Zavedení proměnné a následné přiřazení její hodnoty pak matematicky vyjádříme například takto: `x = 1`.

> V jazyce C# budeme proměnné vytvářet obdobným způsobem. Než si ale ukážeme konkrétní postup, budeme muset pochopit, co jsou to datové typy a typový systém. To je předmětem další kapitoly.

Kdybychom chtěli osvětlit pojem z hlediska hardwaru, mohli bychom říci, že proměnná je pro nás označením nebo pojmenováním určité části paměti počítače. Z hlediska našeho oboru (informatiky a programování) je proměnná jazykový konstrukt, který nám umožní ukládat data všeho druhu.

Pravidla pro práci s proměnnými

Program nemůže obsahovat více proměnných stejného názvu. Daná problematika je ve skutečnosti ještě o něco složitější a existují případy, kdy je i toto možné. Pro zachování jednoduchosti článku ale zamlčím určitá fakta a zatím si postačíme s tímto neúplným pravidlem, než ho v budoucích kapitolách uvedeme na pravou míru.

Pro volbu názvů proměnných existuje několik jasných pravidel a několik poněkud méně jasných, ale stále důležitých doporučení. Název proměnné může obsahovat písmena, číslice a některé speciální znaky. Začínat ale musí vždy písmenem nebo povoleným speciálním znakem – například podtržítko (`_`) nebo v některých případech i zavináč (`@`). Jazyk C# je velmi benevolentní a dovoluje například i použití písmen z ruské abecedy nebo jiné cizojazyčné znaky. **Že to dovoluje ale neznamená, že byste měli znaky z jiných abeced používat.** Pro naše účely bude nejlepší si pamatovat, že háčky, čárky ani jiné exotické znaky do zdrojového souboru nepatří. Jedinou výjimkou z tohoto pravidla jsou obsahy proměnných typu string, ve kterých jsou české znaky přípustné (a žádoucí). Tedy string `p = "Zadejte desetinné číslo:"` je v pořádku, ale deklarace `string příjmení = "Novák"` v pořádku není. Přestože kompilátor C# nebude protestovat proti názvu proměnné "příjmení" (s

háčkem a čárkami), považuje se mezi programátory použití takového názvu za nepřijatelné a trestuhodné.

Pokud z nějakého důvodu i tak chceme použít klíčové slovo jako název proměnné, prefixujeme daný název znakem zavináče (`@`). Například tedy `@int` nebo `@out`. Pokud to není nezbytně nutné, nedělejte to a najděte lepší název!

Konvence pro práci s proměnnými

Pro jistotu znovu uvedu, že konvence jsou doporučení, která dodržujeme (nebo se o to aspoň snažíme), ale při jejich porušení neohrozíme funkčnost programu. K proměnným se váže velké množství konvencí a my si nyní představíme některé z nich.

Mezi první doporučení patří, že názvy proměnných píšeme s malým počátečním písmenem a vždy bez diakritiky nebo jiných speciálních znaků. Například tedy `jmeno`, `vyska` nebo `prumer`. Pokud se název proměnné skládá z více slov, používáme tzv. `camel case` (česky velbloudí notace). To znamená, že jednotlivá slova neoddělujeme mezerou, ale pouze velkým počátečním písmenem nového slova. Příkladem budiž `pocetStudentu`, `prumernaMzda` nebo `vazenyAritmetickyPrumer`.

Dále se snažíme, aby názvy proměnných byly „sebepopisné“. Tedy například budeme-li popisovat nějakou osobu, její jméno budeme mít v proměnné `jmeno`, věk v proměnné `vek` a datum narození v proměnné `datumNarozeni`. Výjimku z tohoto pravidla tvoří především celočíselné indexační proměnné, pro které se typicky volí jednopísmenné názvy jako `i`, `j`, `k` apod. O takovýchto proměnných se více dozvíme v budoucích kapitolách. Nikdy však nezacházejme v sebepopisnosti do extrémů. Délka jména proměnné by neměla být neúnosná.

> Ačkoliv to nemusí být na první pohled zřejmé, dodržování konvencí má svůj smysl a je naprosto klíčové pro psaní přehledného a do budoucna rozšiřitelného kódu.

Konstanty

Pro `konstanty` platí stejná pravidla a doporučení jako pro proměnné. Jak ale jejich název napovídá, narozdíl od proměnných jsou neměnné.

Příklad

```
const int minDelkaHesla = 8;
```

Hodnota konstanty musí být známá v okamžiku překladu (kompilace) programu. Například

Chybná ukázka použití konstanty

```
int i = 5;
const int minDelkaHesla = 4 * i;
```

vyhodí chybu, protože hodnota na pravé straně přiřazení není v okamžiku kompilace známá.

> Přesný význam tohoto kódu si osvětlíme v příští kapitole.

Kompilátor kontroluje, zda se kdekoliv v programu nevyskytuje `minDelkaHesla` na levé straně od operátoru přiřazení (`=`). Na rozdíl od jiných jazyků se konstanty obvykle nepíší velkými písmeny (v tomto případě `MINDELKAHESLA`), ale dodržuje se obvyklá "velbloudí" konvence.

Literály

Hodnoty, které definujeme ve zdrojovém kódu se nazývají literály. Zde je několik příkladů:

Ukázka literálů

```
bool jeDoma = false;  // typ bool může mít hodnoty true nebo false
char velkeA = 'A';  // jednoduché uvozovky, není totéž, co string velkeA = "A"!!
char velkeA = '\u0065';  // totéž co předchozí deklarace, tentokrát pomocí kódu znaku v
Unicode
char tabulator = '\t';
char novyRadek = '\n';
byte b = 64;
int i = 16;
int k = 0x10;  // celočíselný literál v šestnáctkovém(hexadecimálním) tvaru
long velkeCislo = 68000L;
int velkeCislo = 68000L;  // vyhodí chybu, přestože hodnota je v rozsahu povoleném pro int,
znak L vynutí větší alokovanou paměť
double polomer = 2.5;  // když je to bez písmene, považuje se hodnota 2.5 za double
decimal polomer = 2.5m;  // když chceme zvýšenou přesnost, používáme typ decimal. Deklarace
decimal polomer = 2.5 vyhodí chybu - viz implicitní konverze v další kapitole
float polomer = 2.5f;  // i v tomto případě float polomer = 2.5 vyhodí chybu - viz implicitní
konverze v další kapitole
double vzdalenostMesice = 384.4e6; // == 384.4 * 106
double deformace = 1.3e-3; // == 1.3 * 10^-3
string cesta = "C:\\Windows\\Notepad.exe";
string cesta = @"C:\Windows\Notepad.exe";
string nazor = "Nejlepší skladbou Jimmiho Hedrixe je \"Hey Joe\"";
```

Složené závorky

Složené závorky definují blok kódu a zároveň obor platnosti proměnných:

Blok kódu (chyba)

```
{
  int i = 1;
  int i = 2;
}
```

V tomto případě kompilátor vyhodí chybu, protože v jednom bloku nemohou být deklarované dvě proměnné stejného jména.

Blok kódu (v pořádku)

```
{ // blok A
  int i = 1;
}
{ // blok B
  int i = 2;
}
```

Tento kód je v pořádku. První proměnná i má rozsah platnosti v bloku A a když program opustí blok A, proměnná i, která je v něm definovaná, zanikne. Další proměnná i má rozsah platnosti v bloku B a není tedy v konfliktu s první proměnnou i z bloku A.

Závorky nejsou jen oddělovače a matoucí smetí!

Shrnutí

Proměnné jsou cesta, jak dočasně uchovávat data všeho druhu. V další kapitole si ukážeme, jak proměnnou založit a napíšeme tedy i náš první kód.

Revision #5
Created 2025-05-19 10:19:22 UTC by Magdalena Dobešová
Updated 2025-05-19 13:45:30 UTC by Sklenář Maxim