Vyšší odborná škola obalové techniky
a střední škola, Štětí

Digitální učební materiály

Programování - Programování C#

Ivan Pomykacz

esf evropský sociální fond v ČR
EVROPSKÁ UNIE
MINISTERSTVO ŠKOLSTVÍ, MLÁDEŽE A TĚLOVÝCHOVY
OP Vzdělávání pro konkurenceschopnost
INVESTICE DO ROZVOJE VZDĚLÁVÁNÍ

Licence

Disclaimer

Tento PDF dokument byl strojově vygenerován z HTML stránek

http://dumy.odbornaskola.cz/pomykacz/.

Je tedy možné, že sazba textu může obsahovat chyby. Jde převážně o vizuální a typografické chyby, které mohou narušit plynulou čitelnost textu. V některých případech může být ohrožena i funkčnost některých komponent (jako vnitřní odkazy).

Vzhledem k tomu, že vypracované materiály nebyly nikdy určeny pro výstupní formát PDF, autor se zříkává jakékoli odpovědnosti za nalezené chyby. Nesnažte se proto v této souvislosti autora kontaktovat.

Programové vybavení

Textový procesor

Obsah

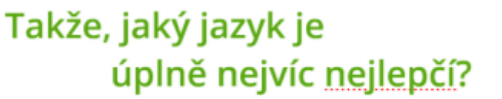
Zdrojový kód

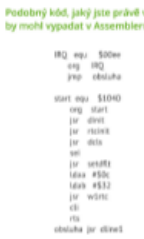
Název školy	Vyšší odborná škola obalové techniky a Střední škola, Štětí, příspěvková organizace		
Adresa školky	Kostelní 134, 411 08 Štětí		
IČ	46773509		
Název operačního programu	OP Vzdělávání pro konkurenceschopnost		
Registrační číslo	CZ.1.07/1.5.00/34.1006		
Označení vzdělávacího materiálu	VY_32_INOVACE_23_PRG_441		
Název tématické oblasti (sady)	Programování		
Název materiálu	Zdrojový kód		
Anotace	Prezentace uvádí posluchače do problematiky programovacích jazyků z pohledu psaní a čitelnosti zdrojového kódu. Vysvětluje, co je to zdrojový kód, a co se s ním v počítači děje.		
Autor	Ivan Pomykacz	**Jazyk**	čeština
Očekávaný výstup	Získá povědomí o různorodosti programovacích jazyků (vyšší, nižší) a čitelnosti zdrojového kódu.		
Klíčová slova	zdrojový kód,programovací jazyk		
Druh výukového zdroje	Výklad	**Věková skupina žáků**	15+
Typ interakce	aktivita	**Ročník**	1.
Speciální vzdělávací potřeby	žádné		
Zhotoveno, (datum/období)	04.09.2013	**Celková velikost**	

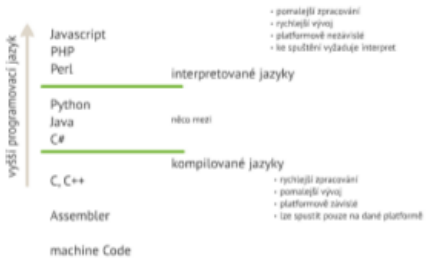
Obsah

Prezentace

online verze: http://prezi.com/
offline verze: zdrojový-kód.pdf







Vývojové prostředí

Název školy	Vyšší odborná škola obalové techniky a Střední škola, Štětí, příspěvková organizace		
Adresa školky	Kostelní 134, 411 08 Štětí		
IČ	46773509		
Název operačního programu	OP Vzdělávání pro konkurenceschopnost		
Registrační číslo	CZ.1.07/1.5.00/34.1006		
Označení vzdělávacího materiálu	VY_32_INOVACE_23_PRG_442		
Název tématické oblasti (sady)	Programování		
Název materiálu	Vývojové prostředí		
Anotace	Představení vývojových prostředí pro programovací jazyk C#. Dostupnost prostředí dle licence a ceny. Popis instalace pro MS Windows a Linux.		
Autor	Ivan Pomykacz	**Jazyk**	čeština
Očekávaný výstup	Nainstaluje si na počítač vlastní instanci vývojového prostředí. Založí nový projekt a spustí předpřipravenou šablonu kódu.		
Klíčová slova	IDE, Monodevelop, SharpDevelop, instalace		
Druh výukového zdroje	Výklad	**Věková skupina žáků**	15+
Typ interakce	aktivita	**Ročník**	1.
Speciální vzdělávací potřeby	žádné		
Zhotoveno, (datum/období)	09.09.2013	**Celková velikost**	

Obsah

Shrnutí

Pro vývoj aplikací v programovacím jazyce C# existuje několik vývojových prostředí.

- Monodevelop (Windows, Linux, Mac OS)
- SharpDevelop (Windows)
- ReSharper (Windows)
- Visual Studio (Windows)

Poslední dva ve výčtu jsou pod zpoplatněnou licencí.

Integrované vývojové prostředí (neboli IDE - Integrated Development Environment) je tzv. vše v jednom.

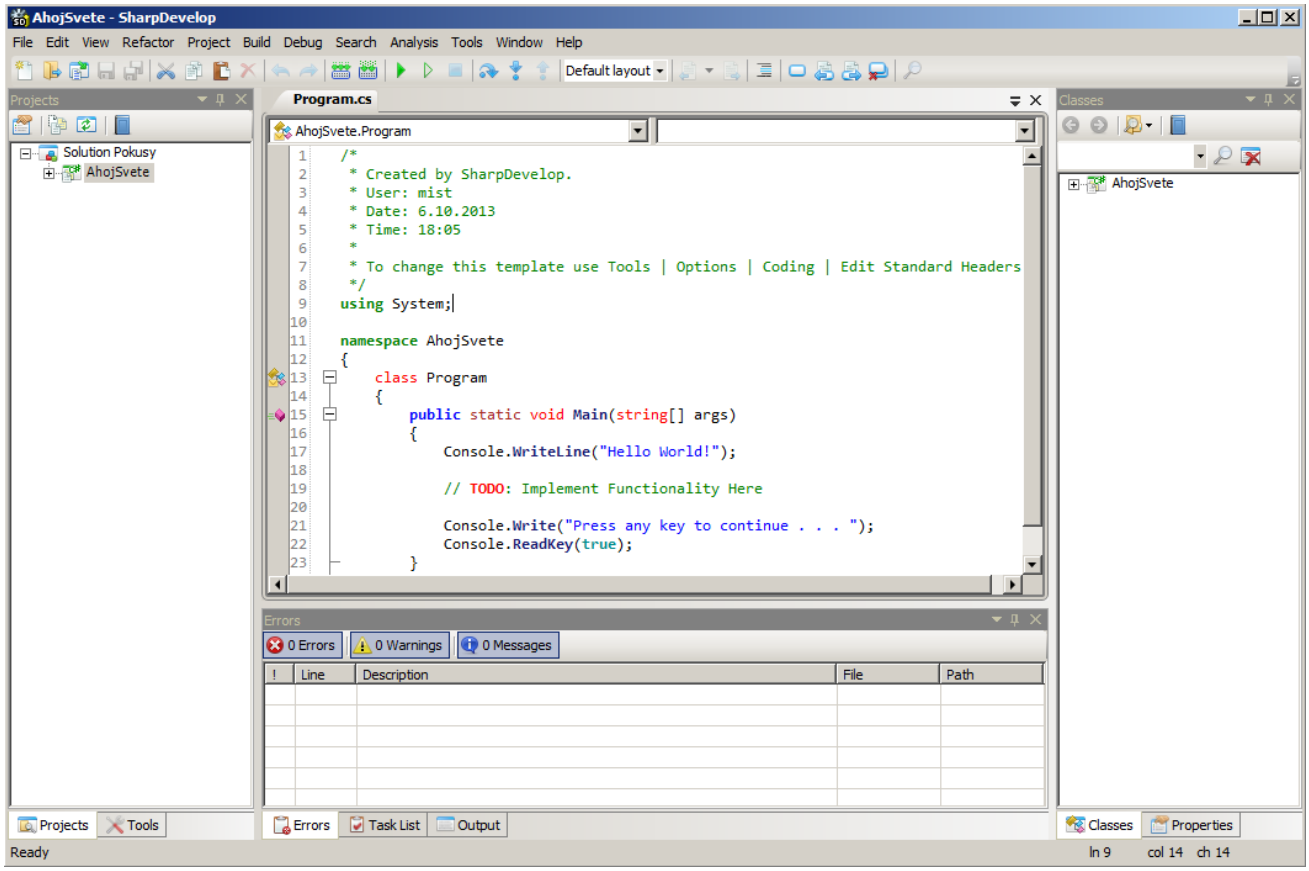
- Umožňuje vývoj aplikací - psaní zdrojového kódu, a navíc jeho psaní ulehčuje různými pomůckami.
- Zvýrazňuje syntaxi psaného kódu.
- Provádí kompilaci a následné spuštění vyvíjené aplikace (a samozřejmě debugging).
- Obsahuje referenční příručku k danému jazyku.
- Podporuje různé systémy správy verzí (GIT, Subversion, Mercurial, TFS).
- Provádí deployment aplikací.
- Obsahuje spoustu cool udělátek ...

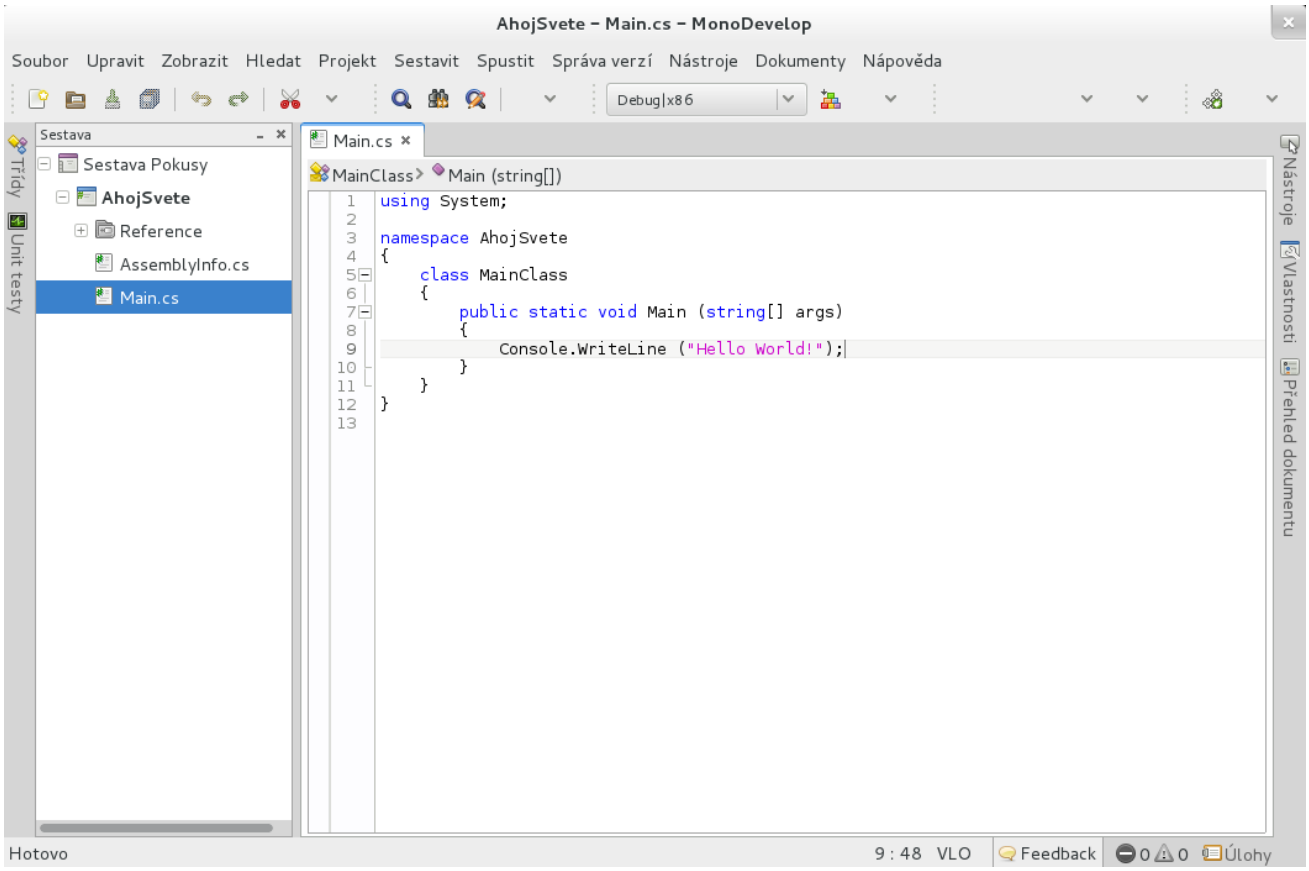
Jaké si vybrat?

Pro začátečníka bych doporučil SharpDevelop a nebo Monodevelop. Zbylé dvě jsou pro začátečníka možná příliš komplexní, i když mnohem profesionálnější. Navíc jak ReSharper, tak Visual Studio mají zpoplatněnou licenci - je třeba propadnout programování, nikoli jeho vývojovému prostředí ...

Instalace

V textu dále se budeme zaobírat pouze open source produkty: SharpDevelop pro MS Windows a Monodevelop pro Linux.

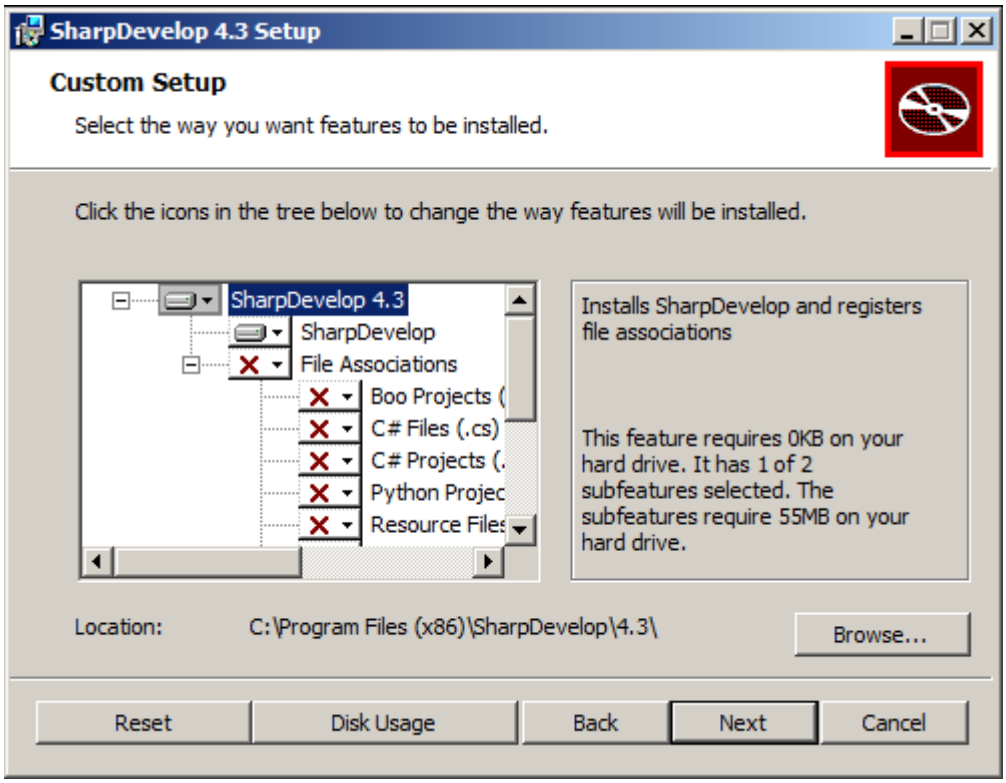




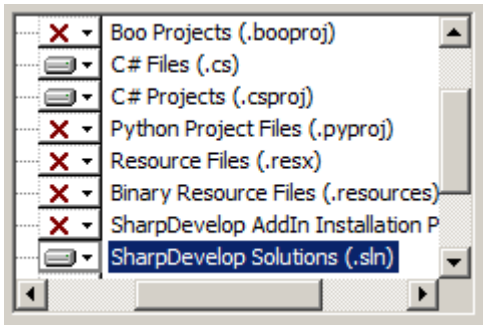
MS Windows

Odkaz ke stažení najdete na http://www.icsharpcode.net/opensource/sd/. Patrně bude nutné ještě doinstalovat .NET Framework.

Při instalaci se vás instalátor otáže, zda-li instalovat - asociovat (File Associations) koncovky souborů s programem SharpDevelop.



Pokud s tím nemáte problém, povolte alespoň tyto (nebo všechny).



Linux

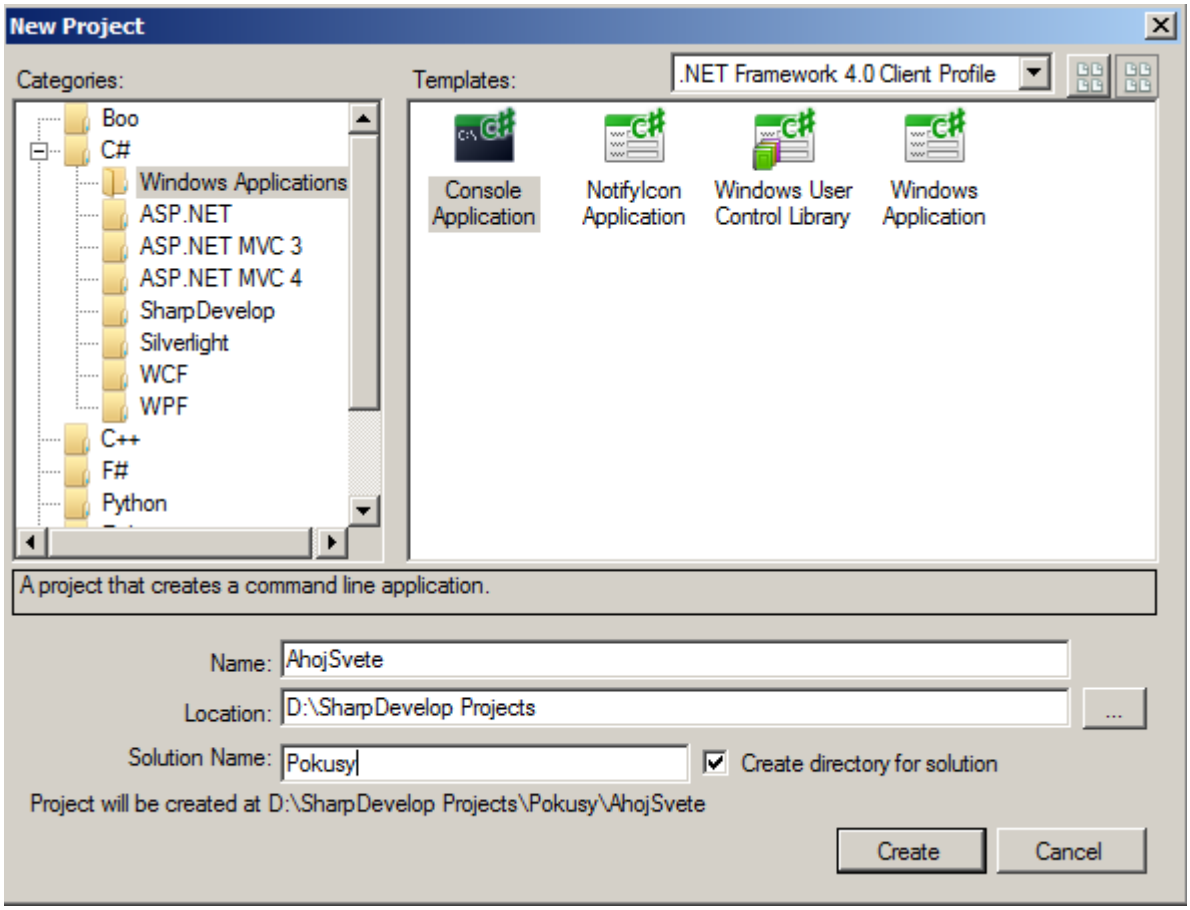
Pro Debian a odvozené distribuce (Ubuntu, Mint, Elementary OS, ...) použijte příkaz
`apt-get install monodevelop`

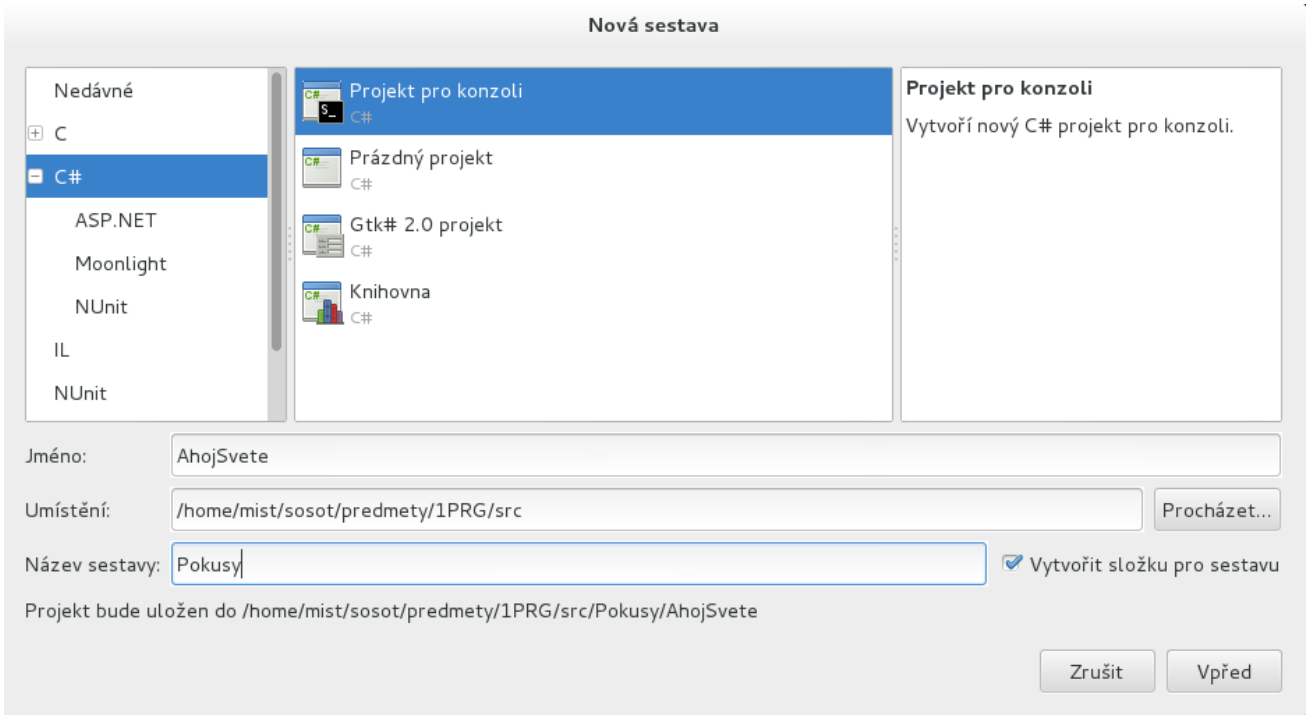
V distribuci Fedora (19) použijte příkaz
`yum install monodevelop`

Ahoj světe!

V následujícím textu budou uvedeny vždy dva screenshoty obrazovek (první pro SharpDevelop (anglicky), druhý pro MonoDevelop(česky))

Pro ověření funkčnosti spusťte nainstalované IDE a založte novou sestavu (solution);





Kliknutím na "Vytvořit" (Create) založíte novou sestavu (Pokusy) a v ní projekt (AhojSvete). Současně se do výchozího souboru vložil ze šablony zdrojový kód, který nedělá nic jiného, než že při spuštění vypíše na obrazovku (do konzole) slova "Hello World!"

Proveďte tedy spuštění kódu klávesou F5. Po chvilce kompilace na vás vyskočí konzole (terminál), kde budou ona notoricky známá slova: "Ahoj Světe!".

IDE

Název školy	Vyšší odborná škola obalové techniky a Střední škola, Štětí, příspěvková organizace		
Adresa školky	Kostelní 134, 411 08 Štětí		
IČ	46773509		
Název operačního programu	OP Vzdělávání pro konkurenceschopnost		
Registrační číslo	CZ.1.07/1.5.00/34.1006		
Označení vzdělávacího materiálu	VY_32_INOVACE_23_PRG_443		
Název tématické oblasti (sady)	Programování		
Název materiálu	IDE		
Anotace	Prezentace workflow pro psaní aplikací ve vývojovém prostředí Monodevelop (Linux) a Sharpdevelop (MS Windows). Založení projektu (solution). Sestavení aplikace. Spuštění aplikace.		
Autor	Ivan Pomykacz	**Jazyk**	čeština
Očekávaný výstup	Založí nový solution nebo projekt v solution. Provede sestavení (build) projektu/solution. Spustí sestavenou aplikaci.		
Klíčová slova	IDE, Monodevelop, SharpDevelop, projekt, sestava, sestavení, spuštění		
Druh výukového zdroje	Výklad	**Věková skupina žáků**	15+
Typ interakce	aktivita	**Ročník**	1.
Speciální vzdělávací potřeby	žádné		
Zhotoveno, (datum/období)	11.09.2013	**Celková velikost**	

Monodevelop

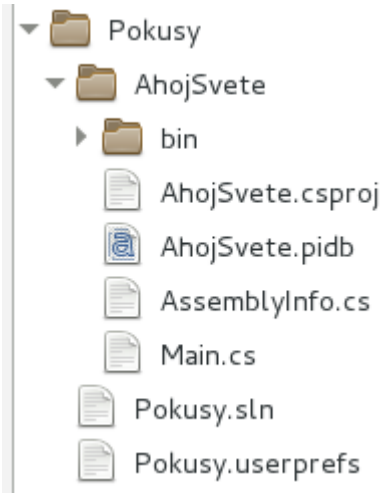
Obsah

Shrnutí

Jak organizovat své projekty? Co je to sestavení (build) projektu? Jak se spouští náš program?

Solution / Sestava

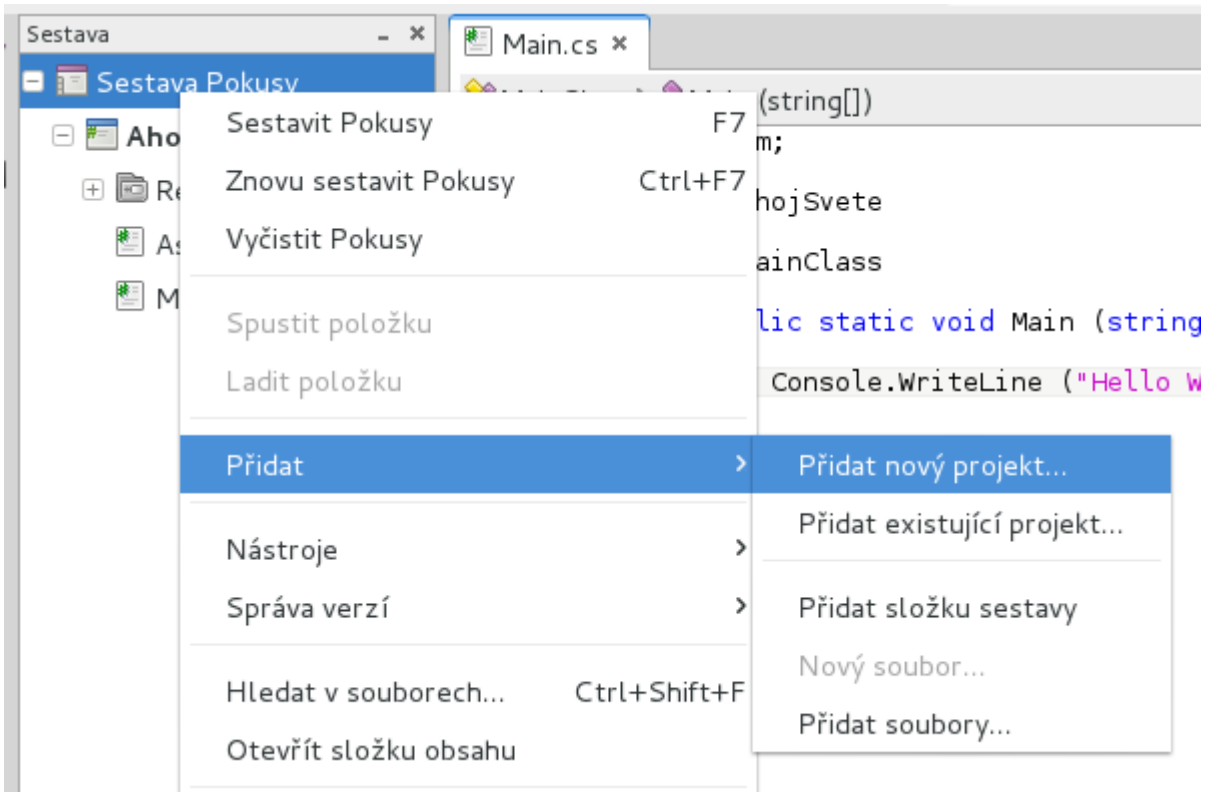
Při zakládání nového projektu se automaticky vytváří i tzv. "Sestava". To má mj. za následek následující adresářovou strukturu nového programu.



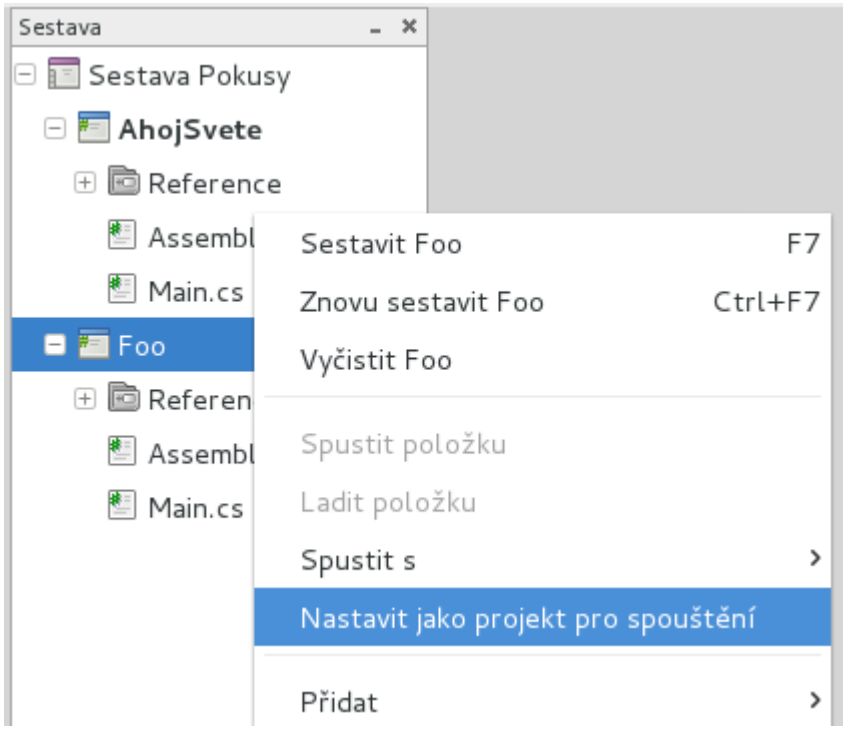
V podstatě jde o to, že sestava (zde "Pokusy") může obsahovat více projektů (zde pouze jeden projekt "AhojSvete"). Zejména takových projektů, které spolu nějak souvisí.

Současně může být otevřená právě jedna sestava a všechny projekty v této sestavě. Samozřejmě v rámci jedné instance vývojového prostředí.

Založme další projekt tím, že klikneme pravým tlačítkem na název sestavy a z kontextového menu vybereme "Přidat nový projekt". Název projektu zvolme "Foo".

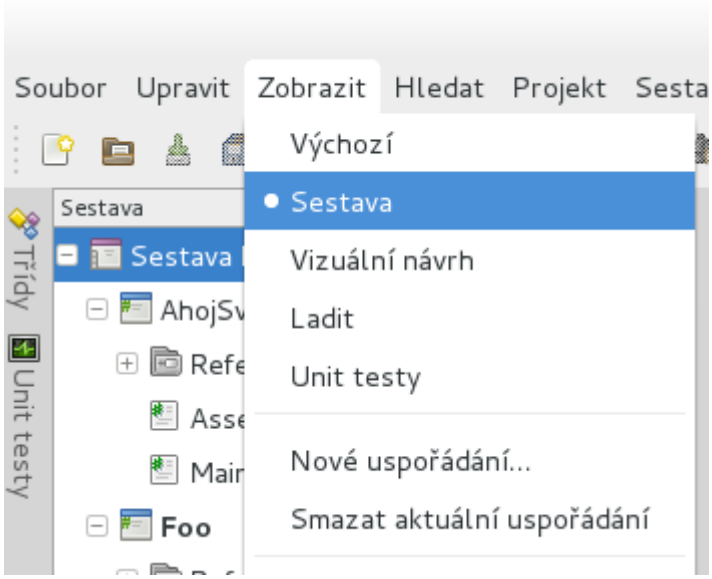


Přidaný projekt se ovšem nenastaví jako výchozí, tj. takový, který se při stisku klávesové zkratky pro spuštění spustí (případně sestaví). To lze změnit nastavením výchozího projektu pro spouštění.



Pohledy pracovního prostředí

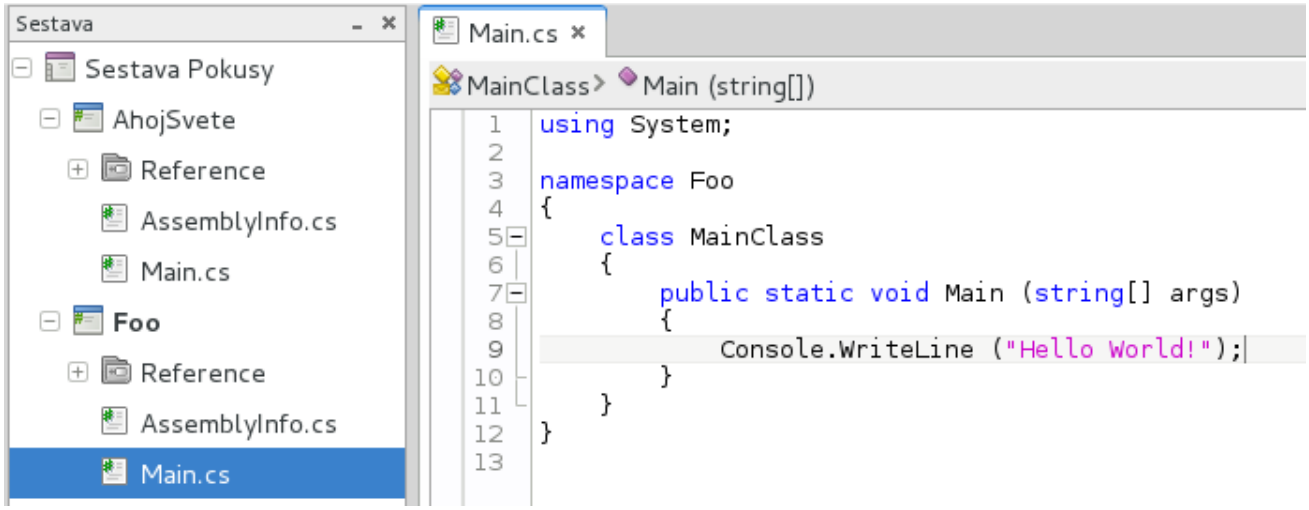
Monodevelop se přepíná mezi různými pohledy podle toho, jakou činnost zrovna vykonáváte. Pokud píšete/upravujete zdrojový kód a potřebujete často otevírat soubory v projektech, pak se vám bude hodit zobrazení "Sestava", dostupné v hlavní nabídkové liště.



Projekt

Výchozím souborem nového projektu je soubor `Main.cs`. Je to soubor, který obsahuje výchozí bod našeho programu - ten musí někde začínat.

Každý projekt jej již obsahuje. Průvodce založením nového projektu automaticky vygeneruje takovýto soubor a vloží do něj připravenou šablonu "Hello World!".



Sestavení

Jde o proces, ve kterém jsou schované činnosti jako parsování zdrojového kódu a jeho následná kompilace.

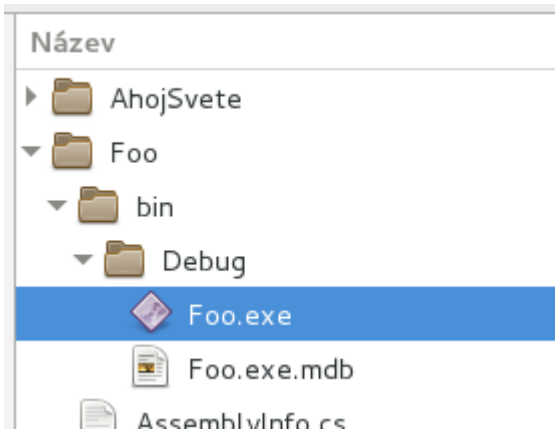
- Pro sestavení celé sestavy (solution) použijte klávesu `F8`.
- Pro sestavení výchozího projektu použijte klávesu `F7`.

Parsování je proces, který odhalí případné syntaktické chyby. Na ty vás ale většinou už upozorní samotné IDE již při psaní kódu.

Kompilátor pak provádí důkladnou analýzu kódu a hlídá si, mj. zda-li jsou deklarované všechny potřebné proměnné, zda-li mají přiřazenou hodnotu a zda-li se někde neděje chybné přiřazení

datových typů.

Výsledkem celého "sestavení" je pak binární soubor s koncovkou `.exe`.



Sestavení je vhodné průběžně provádět. Umožňuje totiž odhalit případné chyby ještě na začátku. Pokud razíte techniku napsat co nejvíc kódu a pak se modlit, jestli to bude fungovat, pak tento krok můžete přeskočit.

Spuštění

Spuštění spočívá v tom, že dojde ke skutečnému spuštění vaší aplikace (tj. zkompilovaného kódu). Jsou v podstatě dva režimy spouštění kódu - klasické spuštění a nebo v režimu ladění (debugging).

Spustit

Klávesová zkratka `Ctrl+F5`. Váš program se nejprve sestaví (nebyl-li ještě aktuálně sestaven) a potom se spustí.

Ladit

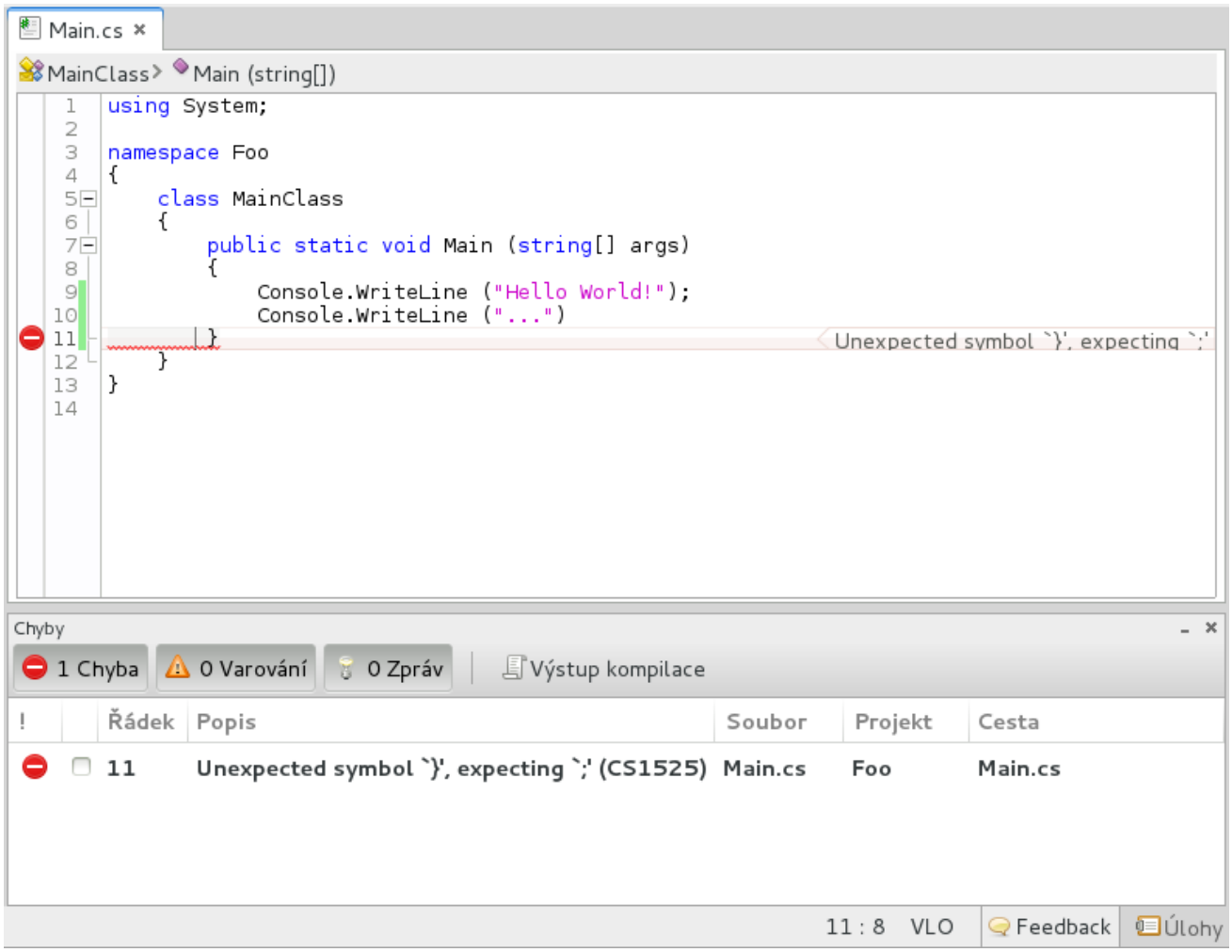
Klávesová zkratka `F5`. Váš program se nejprve sestaví (nebyl-li ještě aktuálně sestaven) a potom se spustí v režimu ladění. To mj. znamená, že pokud máte nastaveny tzv. break pointy, tak na těchto break pointech dojde k pozastavení programu (ať je v jakékoli fázi).

Je to něco jako pauza a vy se během té pauzy můžete podívat na stav programu na určené pozici ve zdrojovém kódu. Akorát nyní můžete např. nahlížet do hodnot v proměnných a nebo sledovat tok programu.

Ladění je pokročilý nástroj, který uvítáte u komplexnějších programů s tisíci řádky zdrojového kódu. I když někdy stačí ladit i obyčejnou rekurzi napsanou na deseti řádcích.

Chyby při sestavení

Pokud se při sestavování projektu odhalí chyby, budete na ně patřičně upozorněni. S odstraněním chyby si už ale budete muset poradit sami.



U výpisu chyb najdete číslo řádku, popis chyby a soubor, kde byla chyba nalezena.

Ne vždy ovšem bude nalezená chyba jednoznačná. Např. na obrázku výše byl zjevně zapomenut středník, ovšem na řádku č. 10 (nikoli 11).

SharpDevelop

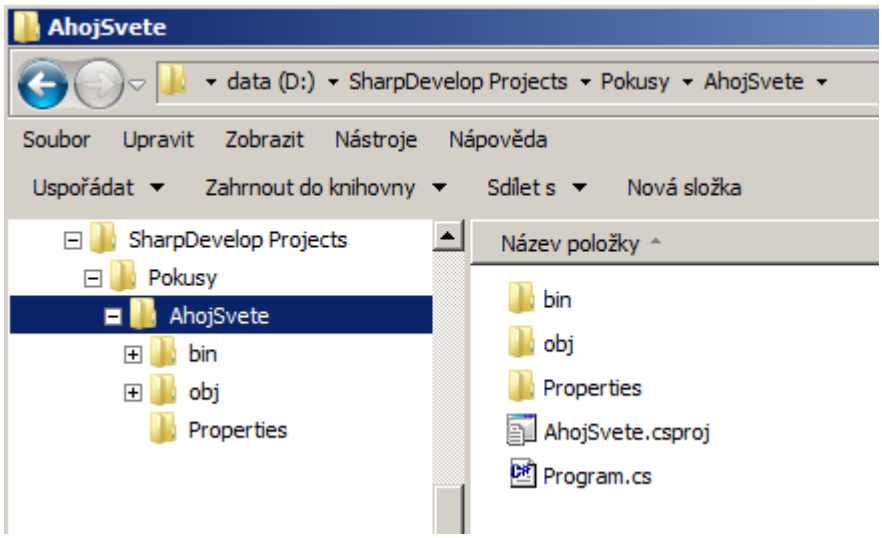
Obsah

Shrnutí

Jak organizovat své projekty? Co je to sestavení (build) projektu? Jak se spouští náš program?

Solution / Sestava

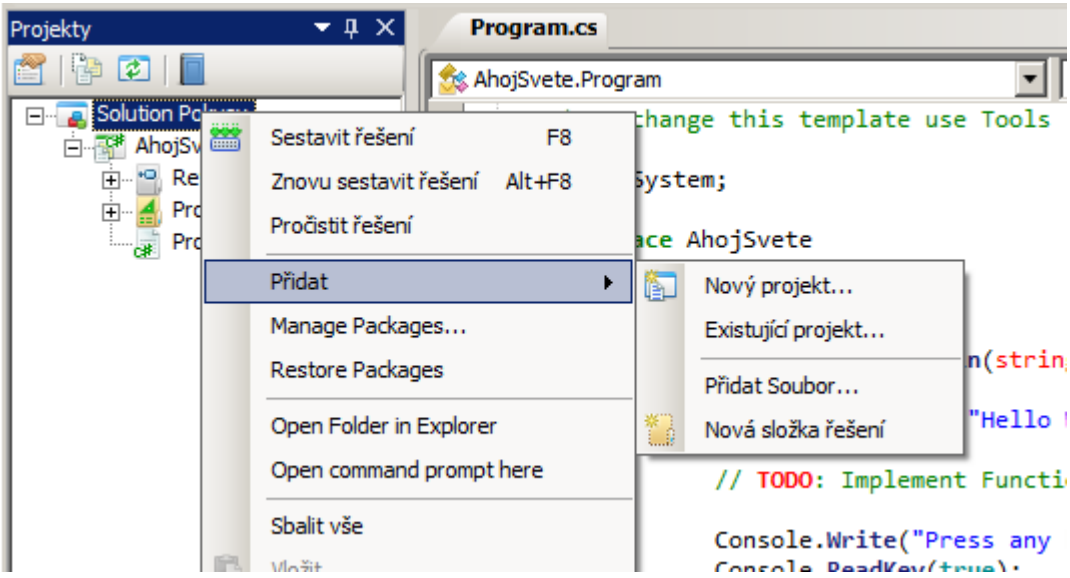
Při zakládání nového projektu se automaticky vytváří i tzv. "Sestava". To má mj. za následek následující adresářovou strukturu nového programu.



V podstatě jde o to, že sestava (zde "Pokusy") může obsahovat více projektů (zde pouze jeden projekt "AhojSvete"). Zejména takových projektů, které spolu nějak souvisí.

Současně může být otevřená právě jedna sestava a všechny projekty v této sestavě. Samozřejmě v rámci jedné instance vývojového prostředí.

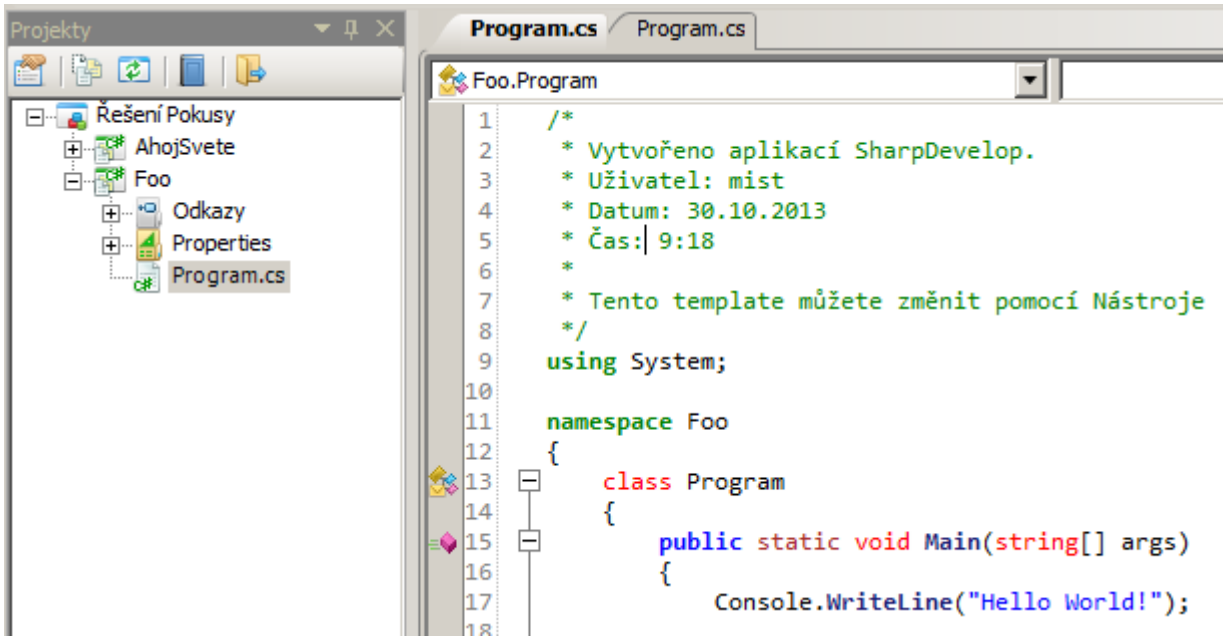
Založme další projekt tím, že klikneme pravým tlačítkem na název sestavy a z kontextového menu vybereme "Přidat -> Nový projekt". Název projektu zvolme "Foo".



Projekt

Výchozím souborem nového projektu je soubor `Program.cs`. Je to soubor, který obsahuje výchozí bod našeho programu - ten musí někde začínat.

Každý projekt jej již obsahuje. Průvodce založením nového projektu automaticky vygeneruje takovýto soubor a vloží do něj připravenou šablonu "Hello World!".



Sestavení

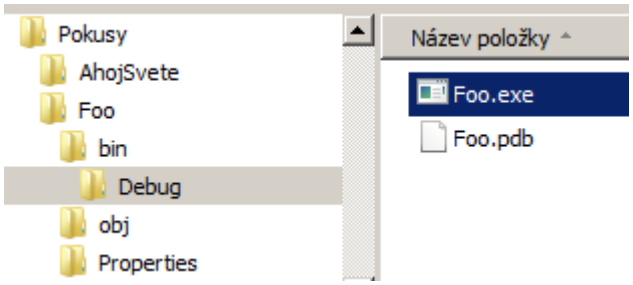
Jde o proces, ve kterém jsou schované činnosti jako parsování zdrojového kódu a jeho následná kompilace.

- Pro sestavení celého řešení (solution) použijte klávesu `F8`.
- Pro sestavení výchozího projektu použijte klávesu `F9`.

Parsování je proces, který odhalí případné syntaktické chyby. Na ty vás ale většinou už upozorní samotné IDE již při psaní kódu.

Kompilátor pak provádí důkladnou analýzu kódu a hlídá si, mj. zda-li jsou deklarované všechny potřebné proměnné, zda-li mají přiřazenou hodnotu a zda-li se někde neděje chybné přiřazení datových typů.

Výsledkem celého "sestavení" je pak binární soubor s koncovkou `.exe`.



Sestavení je vhodné průběžně provádět. Umožňuje totiž odhalit případné chyby ještě na začátku. Pokud razíte techniku napsat co nejvíc kódu a pak se modlit, jestli to bude fungovat, pak tento krok můžete přeskočit.

Spuštění

Spuštění spočívá v tom, že dojde ke skutečnému spuštění vaší aplikace (tj. zkompilovaného kódu). Jsou v podstatě dva režimy spouštění kódu - klasické spuštění a nebo v režimu ladění (debugging).

Spustit

Klávesová zkratka `Ctrl+F5`. Váš program se nejprve sestaví (nebyl-li ještě aktuálně sestaven) a potom se spustí.

Ladit

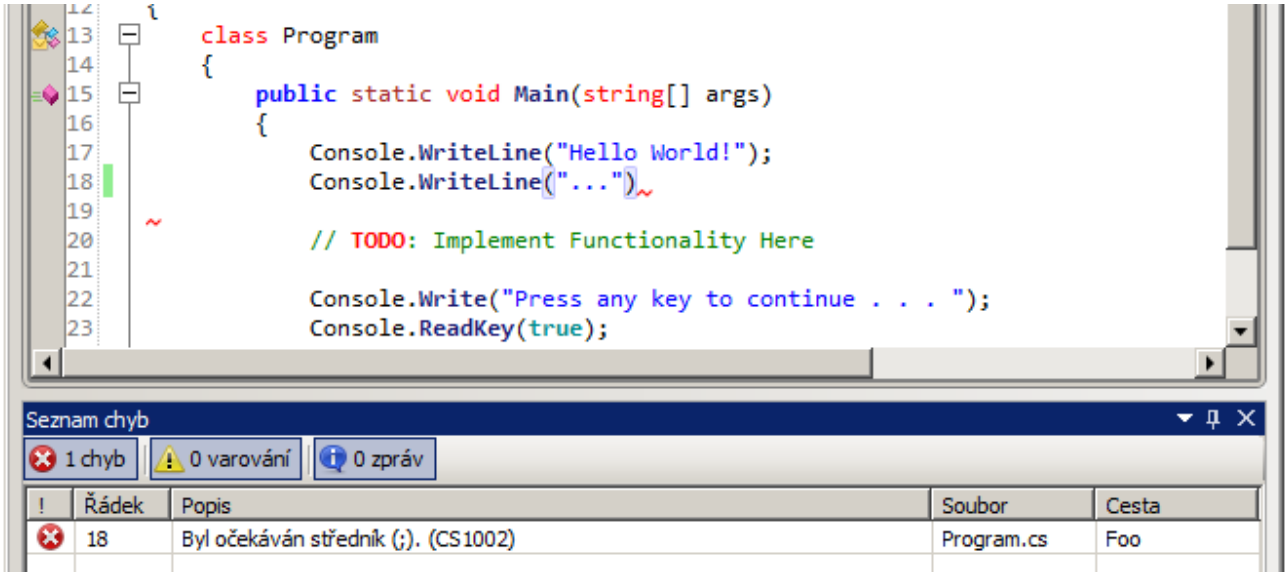
Klávesová zkratka `F5`. Váš program se nejprve sestaví (nebyl-li ještě aktuálně sestaven) a potom se spustí v režimu ladění. To mj. znamená, že pokud máte nastaveny tzv. break pointy, tak na těchto break pointech dojde k pozastavení programu (ať je v jakékoli fázi).

Je to něco jako pauza a vy se během té pauzy můžete podívat na stav programu na určené pozici ve zdrojovém kódu. Akorát nyní můžete např. nahlížet do hodnot v proměnných a nebo sledovat tok programu.

Ladění je pokročilý nástroj, který uvítáte u komplexnějších programů s tisíci řádky zdrojového kódu. I když někdy stačí ladit i obyčejnou rekurzi napsanou na deseti řádcích.

Chyby při sestavení

Pokud se při sestavování projektu odhalí chyby, budete na ně patřičně upozorněni. S odstraněním chyby si už ale budete muset poradit sami.



U výpisu chyb najdete číslo řádku, popis chyby a soubor, kde byla chyba nalezena.

Ne vždy ovšem bude nalezená chyba jednoznačná jako na obrázku výše.

Referenční příručka

Název školy	Vyšší odborná škola obalové techniky a Střední škola, Štětí, příspěvková organizace		
Adresa školky	Kostelní 134, 411 08 Štětí		
IČ	46773509		
Název operačního programu	OP Vzdělávání pro konkurenceschopnost		
Registrační číslo	CZ.1.07/1.5.00/34.1006		
Označení vzdělávacího materiálu	VY_32_INOVACE_23_PRG_444		
Název tématické oblasti (sady)	Programování		
Název materiálu	Referenční příručka		
Anotace	Materiál shrnuje probrané metody (objekty) formou referenční příručky. Je zde kromě popisu uveden i způsob použití uvedených metod.		
Autor	Ivan Pomykacz	**Jazyk**	čeština
Očekávaný výstup	Pracuje s referenční příručkou. Vyhledává informace v oficiální příručce jazyka.		
Klíčová slova	reference jazyka, metody, příklady použití		
Druh výukového zdroje	Výklad	**Věková skupina žáků**	15+
Typ interakce	aktivita	**Ročník**	1.
Speciální vzdělávací potřeby	žádné		
Zhotoveno, (datum/období)	13.09.2013	**Celková velikost**	

Obsah

Shrnutí

Přehled použitých metod ve cvičeních z Programování. Metody jsou psány abecedně podle názvu a následně podle jmenného prostoru (namespace). U každé metody je krátká ukázka jejího použití.

Pokud jsou v ukázkách použity tři tečky (...), znamená to, že kód na sebe nenavazuje a zpravidla se obě části kódu píší na jiných místech, než za sebou. Nicméně je nezbytné, aby byly uvedeny obě části "někde" v kódu.

Toto je pouze chabá náhražka referenční příručky jazyka C#. Rozhodně je doporučeno používat odkazovanou referenční příručku.

Přehled

Metoda	Objekt	Namespace	Použití

`System.IO.File.AppendAllText()`

AppendAllText()	File	*System.IO*	
Přidá nový obsah na konec souboru.			
	return	`string`	

`System.Math.Ceiling()`

Ceiling()	Math	*System*	
Provede zaokrouhlení desetinného čísla směrem nahoru.			
	return	`double`	

System.IO.File.Exists()

Exists()	File	*System.IO*	
Testuje existenci souboru.			
	return	`bool`	

System.Math.Log()

Log()	Math	*System*	
Spočítá přirozený logaritmus nebo logaritmus daného základu.			
	return	`double`	

System.DateTime.Now

Now	DateTime	*System*	
Vrací aktuální čas (včetně data)			
	return	`DateTime`	

double.Parse()

Parse()	double	*builtin*	
Parsuje vstupní argument, kterým je string, a pokud to lze, převede hodnotu na číselný datový typ double.			
	return	`double`	

```
double cislo;
double = double.Parse("3,14");

string text = "2,87";
cislo = double.Parse(text);
```

int.Parse()

Parse()	int	*builtin*	
Parsuje vstupní argument, kterým je string, a pokud to lze, převede hodnotu na číselný datový typ int.			
	return	`int`	

```
int cislo;
cislo = int.Parse("42");

string text = "158";
cislo = int.Parse(text);
```

System.IO.File.ReadAllLines()

ReadAllLines()	File	*System.IO*	
Program načte všechny řádky z textového souboru a uloží do pole řetězců.			
	return	`string[]`	

System.IO.File.ReadAllText()

ReadAllText()	File	*System.IO*	
Načte obsah souboru do paměti.			
	return	`string`	

System.Console.ReadLine()

ReadLine()	Console	*System*	
Umožní zadat uživateli text z klávesnice a vrátit jej jako string. Program se pozastaví - čeká na stisk klávesy Enter.			
	return	`string`	

```
using System;
//...
string jmeno;
Console.Write("Zadej jméno: ");
jmeno = Console.ReadLine();
```

string.Substring()

Substring()	string	*builtin*	
Metoda vrací část řetězce.			
	return	`string`	

string.ToLower()

ToLower()	string	*builtin*	
Převede všechna velká písmena v řetězci na malá.			
	return	`string`	

System.DateTime.Today

Today	DateTime	*System*	
Vrací aktuální datum (nikoli čas)			
	return	`DateTime`	

string.Trim()

Trim()	string	*builtin*	
Zbaví okraje řetězece bílých znaků (mezery, řádkové zlomy, tabulátory, ...)			
	return	`string`	

System.Console.Write()

Write()	Console	*System*	
Vypíše parametr metody na obrazovku (konzoli) bez řádkového zlomu.			
	return	`void`	

```
using System;
//...
Console.Write("Ahoj světe!");
string jmeno = "John";
Console.Write("Hello " + jmeno + "!");
```

System.IO.File.WriteAllText()

WriteAllText()	File	*System.IO*	
Uloží - přepíše soubor novým obsahem.			
	return	`string`	

`System.Console.WriteLine()`

WriteLine()	Console	*System*	
Vypíše parametr metody na obrazovku (konzoli) s řádkovým zlomem.			
	return	`void`	

```
using System;
//...
Console.WriteLine("Ahoj světe!");
string jmeno = "John";
Console.WriteLine("Hello " + jmeno + "!");
```

Priorita operátorů

Název školy	Vyšší odborná škola obalové techniky a Střední škola, Štětí, příspěvková organizace		
Adresa školky	Kostelní 134, 411 08 Štětí		
IČ	46773509		
Název operačního programu	OP Vzdělávání pro konkurenceschopnost		
Registrační číslo	CZ.1.07/1.5.00/34.1006		
Označení vzdělávacího materiálu	VY_32_INOVACE_23_PRG_445		
Název tématické oblasti (sady)	Programování		
Název materiálu	Priorita operátorů		
Anotace	Text seznamuje se základními (probíranými) operátory v programovacím jazyce. Vysvětluje pojmy výraz, operand, operátor. Objasňuje typy operátorů a příklady využití. Velký důraz je kladen na prioritu operátorů.		
Autor	Ivan Pomykacz	**Jazyk**	čeština
Očekávaný výstup	Pojmenuje operátor a operand ve výrazu. Sestavuje výrazy s operátory a operandy. Rozumí prioritě – způsobu vyhodnocování operátorů.		
Klíčová slova	operátor, operand, priorita, unární operátor, binární operátor, ternární operátor		
Druh výukového zdroje	Výklad	**Věková skupina žáků**	15+
Typ interakce	aktivita	**Ročník**	1.
Speciální vzdělávací potřeby	žádné		
Zhotoveno, (datum/období)	21.10.2013	**Celková velikost**	

Obsah

Shrnutí

Výraz

Výraz je tvořen sekvencí operátorů a operandy.

Ukažme si to na příkladu z matematiky, tam se to operátory a operandy jen hemží. Např. výraz `x = 1 + 2`.

Kolik je zde operátorů a operandů?

- operátory: `+`, `=`
- operandy: `1`, `2` a `x`

Ano, patrně jste si odvodili, že operátor je ten kdo s něčím operuje (pracuje). Operand je ten flákač, který se sebou nechá dělat, co se vám zlíbí.

Význam operátorů

Jakmile si uvědomíme významy jednotlivých operátorů, budeme výrazům lépe rozumět.

Operátor `+` provádí operaci součet, tj. dokáže sečíst dva operandy, které jsou po jeho levé a pravé straně. Ano, na tom totiž záleží. V našem programovacím jazyce není možné napsat např. `x = + 1 2` .

Operátor `=` provádí přiřazení hodnoty zprava doleva. Tzn. výsledek operace la pravé straně uloží do operandu na levé straně.

Priorita

Zcela automaticky chápeme výraz `x = 1 + 2`. Když se vás někdo zeptá kolik je *x*, okamžitě odpovíte, že 3.

Pokud se máme bavit o nějaké prioritě operátorů, tak určitě víte, že pokud si chceme ujasnit, co se má provést (vypočítat) dřív (nejprve), tak to dáme do závorek.

Kdyby jste dostali za úkol "ozávorkovat" náš výraz `x = 1 + 2`, tak vás nejspíš napadne jediná možnost: `x = (1 + 2)`. A to zároveň vystihne i prioritu, v jaké se provedou jednotlivé operace. Nejprve se provede operátor `+`, protože je v závorce.

Teprve potom se provede `=`. Ono to funguje ale i bez závorek. To proto, že `+` má vyšší prioritu, než `=`.

Představte si situaci, kdyby tomu tak nebylo. Kdyby `=` mělo vyšší prioritu, než `+`. Pak bychom museli náš výraz ozávorkovat takto: `(x = 1) + 2`. To už vypadá trochu méně smysluplněji, že?

Proto mají operátory svou prioritu, abychom věděli, kde s vyhodnocováním výrazu začít. Tedy my ne, ale programovací jazyk. Nám se vyplatí znát pravidla, abychom mohli výrazy vhodně poskládat.

Platí, že čím nižší priorita, tím později operátor provede svou akci.

Typy operátorů

V programovacím jazyce se můžeme setkat se třemi typy operátorů:

- **unární** - mají pouze jeden operand. Např. `x++`.
- **binární** - mají klasicky dva operandy, jako `x + y`.
- **ternární** - je pouze jeden `?:`, a ano má tři operandy: `c ? a : b`

Vyhodnocování výrazů

Pořadí, v jakém se vyhodnocují jednotlivé operátory je dáno prioritou. To ale nestačí.

Mějme výraz `x + y * z`. Ze základní školy víme, že násobení má přednost před sčítáním. To platí samozřejmě i zde. Kdybychom měli aplikovat závorky na náš výraz, tak by to dopadlo takto: `x + (y * z)`.

Ale co když bude náš výraz takovýto: `x + y + z`?

V jakém pořadí se vyhodnotí teď? A není to jedno? Přeci `x + (y + z)`, je to samé jako `(x + y) + z`. Ano i ne.

Počítač je striktně deterministický stroj. Pro náhodu zde není místo. (Generování náhodný čísel je proto trochu oříšek, ale to je zase jiná pohádka) Vyhodnocování výrazů se musí řídit přesnými pravidly.

Proto se u operátorů ještě definuje tzv. asociativnost zleva nebo zprava.

Kromě operátoru `=`, jsou všechny binární operátory asociativní zleva. To znamená, že se vyhodnocují zleva doprava. Tudíž výraz uvedený výše se vyhodnotí jako: `(x + y) + z`.

Priority operátorů

Tabulka obsahuje výčet některých operátorů. Priorita je od shora (nejvyšší) dolů (nejnižší) a zleva (vyšší) doprava (nižší).

priorita	kategorie	operátory
nejvyšší	Primární	`x++`, `x--`
	Unární	`++x`, `--x`
	Multiplikativní	`*`, `/`
	Aditivní	`+`, `-`
	Relační	`<`, `>`, `<=`, `>=`
	Porovnávací	`==`, `!=`

nejnižší	Přiřazovací	`=`, `*=`, `/=`, `+=`, `-=`

Proměnné

Název školy	Vyšší odborná škola obalové techniky a Střední škola, Štětí, příspěvková organizace		
Adresa školky	Kostelní 134, 411 08 Štětí		
IČ	46773509		
Název operačního programu	OP Vzdělávání pro konkurenceschopnost		
Registrační číslo	CZ.1.07/1.5.00/34.1006		
Označení vzdělávacího materiálu	VY_32_INOVACE_23_PRG_446		
Název tématické oblasti (sady)	Programování		
Název materiálu	Proměnné		
Anotace	Text poprvé seznamuje s pojmem proměnná. Uvádí způsoby zápisu (deklarace). Představuje základní datové typy a uvádí jejich význam. Předkládá cvičení pro deklaraci proměnných dle ukládaných hodnot.		
Autor	Ivan Pomykacz	**Jazyk**	čeština
Očekávaný výstup	Chápe, jak funguje proměnná v počítači. Deklaruje proměnnou. Volí vhodný datový typ dle ukládaných dat. Přiřadí do proměnné hodnotu.		
Klíčová slova	proměnná, datový typ, deklarace, přiřazení hodnoty		
Druh výukového zdroje	Výklad	**Věková skupina žáků**	15+
Typ interakce	aktivita	**Ročník**	1.
Speciální vzdělávací potřeby	žádné		
Zhotoveno, (datum/období)	16.09.2013	**Celková velikost**	

Obsah

Shrunutí

Jak se deklarují proměnné? Jak se do proměnných přiřazují hodnoty?

Proměnná

Jedním ze základních konceptů v programování jsou proměnné. Proměná je slovo (identifikátor), který dokáže udržet jednu hodnotu. Například, programujeme počítačovou hru a potřebujeme, aby si program pamatoval, kolik hráči zbývá many. Vytvoříme proto proměnnou "mana" a uložíme do ní hodnotu 42;

Jinými slovy, proměnná je *pojmenované* místo v operační paměti, ke kterému můžeme přistupovat (číst z něj), a nebo do něj zapisovat. Proč pojmenované? No protože jinak bychom si museli pamatovat adresy konkrétního místa v paměťovém prostoru.

Zápis proměnné

Obyčejně používáme pro zápis názvu proměnné malá písmena anglické abecedy.

Skládá-li se název proměnné z více slov, pak můžete případné mezery nahradit podtržítky, a nebo každé další slovo začít psát velkým písmenem.

```
moje_promenna
mojePromenna
moje promenna
```

Deklarace proměnné

Abychom mohli proměnné používat, musíme je nejprve deklarovat, tj. oznámit dopředu kompilátoru, že budeme potřebovat místo v paměti pro uložení nějakého typu dat.

Deklarace tedy spočívá v tom, že uvedeme typ proměnné (int, double, string, ...) a následně název proměnné.

Je asi zřejmé, že názvy se nesmí opakovat. Kompilátor vás na tento jev případně upozorní fatální chybou.

Datové typy

Jak už bylo zmíněno výše, kompilátor potřebuje dopředu vědět, kolik má vyhradit místa v paměti pro spuštění našeho programu. Proto se u každé proměnné definuje tzv. datový typ, tj. zda-li jde o číslo nebo text, případně úplně jiný objekt.

Uveďme si nejprve základní datové typy:

- **int** (Integer) .. pro ukládání *celých* čísel. Tj. taková čísla, která neobsahují desetinnou čárku (resp. tečku). Rozsah je omezen od -2 147 483 648 do 2 147 483 647.
- **double** .. pro ukládání *desetinných* čísel a samozřejmě i čísel celých.Double je tzv. dvojitá přesnost na až 16 desetinných míst.
- **string** .. pro ukládání textových řetězců. Řetězcem může být téměř jakýkoli text, tzn. písmena, číslice a jiné znaky (mezera, čárka, střední, pomlčka, ...).
- **bool** (Boolean) .. pro ukládání *pravdivostních* hodnot: *True* a *False* (pravda, nepravda).

Ukázka kódu

```
using System;

namespace Promenne01
{
    class MainClass
    {
        public static void Main (string[] args)
        {
            Console.WriteLine("Program: Proměnné a datové typy 1.0\n");

            // deklarace proměnné spočívá v zápisu datového typu a následně
názvu proměnné
            // dále už se pracuje jen s názvem proměnné
            int    celeCislo;
            // názvy proměnných se mohou skládat jen z některých znaků, např.
nesmí obsahovat mezeru
            int    cele_cislo;
            double desetCislo;
            bool   pravdaNepravda;
            string text;

            // pro zápis hodnot do proměnných se používá operátor "="
            celeCislo  = 155;
            cele_cislo = 42;
            // u desetinných čísel píšeme desetinnou tečku
            desetCislo = 3.14;
            // pravdivostní hodnota může být buď true a nebo false
            pravdaNepravda = true;

            // jakýkoli text se musí uzavřít do uvozovek
            text = "bflmpsvz";

            // výpis proměnných na obrazovku
            Console.WriteLine(celeCislo);
            // výpis lze rozšířit o další řetězec a tyto pak spojit pomocí
```

```
"+"
            Console.WriteLine("Celé číslo: " + cele_cislo);
            Console.WriteLine("Desetinné číslo: " + desetCislo);
            Console.WriteLine("Boolean (pravda): " + pravdaNepravda);
            pravdaNepravda = false;
            Console.WriteLine("Boolean (nepravda): " + pravdaNepravda);
            Console.WriteLine("Text: " + text);

            Console.Write("Press any key to continue . . . ");
            Console.ReadKey(true);
        }
    }
}
```

Příklady k procvičení

```
using System;

namespace Cviceni
{
    class MainClass
    {
        public static void Main (string[] args)
        {
            Console.WriteLine ("Program: Procvičování v1.0\n");

            // Deklarujte proměnnou: pes, a uložte do ní hodnotu: Azor.


            // Deklarujte proměnnou: cena, a uložte do ní hodnotu: 4.50.


            // Deklarujte proměnnou: den, a uložte do ní hodnotu: 24.



            // Na řádku č. 24 změňte hodnotu proměnné pocet na 64.
            int pocet;
            pocet = 42;

            Console.WriteLine("Počet: " + pocet);
            // Do proměnné soucet uložte operaci součtu čísel 1+2 (nikoli
výsledek). Jakmile tak učiníte, můžete zrušit // na řádku 29
            int soucet;

            //Console.WriteLine("Součet: " + soucet);


        }
```

```
    }
}
```

Výstup

Název školy	Vyšší odborná škola obalové techniky a Střední škola, Štětí, příspěvková organizace		
Adresa školky	Kostelní 134, 411 08 Štětí		
IČ	46773509		
Název operačního programu	OP Vzdělávání pro konkurenceschopnost		
Registrační číslo	CZ.1.07/1.5.00/34.1006		
Označení vzdělávacího materiálu	VY_32_INOVACE_23_PRG_447		
Název tématické oblasti (sady)	Programování		
Název materiálu	Výstup		
Anotace	Text představuje metody pro výstup textově orientovaných informací na monitor (do terminálu). Zároveň ukazuje možnosti výpisu hodnot uložených v proměnných.		
Autor	Ivan Pomykacz	**Jazyk**	čeština
Očekávaný výstup	Vypíše text nebo hodnotu proměnné na obrazovku (do terminálu). Spojuje do výstupu hodnoty z různých zdrojů pomocí operátoru.		
Klíčová slova	výstup		
Druh výukového zdroje	Výklad	**Věková skupina žáků**	15+
Typ interakce	aktivita	**Ročník**	1.
Speciální vzdělávací potřeby	žádné		
Zhotoveno, (datum/období)	19.09.2013	**Celková velikost**	

Obsah

Shrnutí

Pod výstupem programu si můžeme představit cokoli, co opustilo `{..}` a je to vidět někde jinde než ve zdrojovém kódu. Např. zobrazení textu "Ahoj světe!" na monitoru. Jako výstupní zařízení byl v tomto případě zvolen monitor, resp. grafická karta. A ano, ve finále jde o okno terminálu, kde se text zobrazí.

Dalším výstupním zařízením by mohla být tiskárna. Ale zrovna tak by to mohl být soubor na pevném disku.

Console

Zaměřme se na první popisovaný výstup, tj. do konzole (terminálu, příkazové řádky, shellu atp.). V C# je dostupný objekt `Console`, který obsahuje metody `Write()` a `WriteLine()`. Tyto se liší pouze tím, že druhá jmenovaná provede po vypsání textu řádkový zlom (vloží stisk kl. Enter).

Vyzkoušejte chování

```
Console.Write("Ahoj");
Console.Write("Světe");
Console.Write("!");
```

a následně

```
Console.WriteLine("Ahoj");
Console.WriteLine("Světe");
Console.WriteLine("!");
```

Argumenty

Prozatím si řekněme, že jak metoda `Write()`, tak `WriteLine()` přebírají vstupní argument typu `string`.

Tzn., že můžeme metodě předat buď řetězec `"Ahoj svete!"` (včetně uvozovek), jak jste mohli vidět výše. Nebo proměnnou typu `string`.

```
string text;
text = "Ahoj světe!";
Console.WriteLine(text);
```

Spojování řetězců

Určitě se nám stane, že budeme chtít na výstup zkombinovat text a hodnoty uložené v různých proměnných. Jako např. v následujícím příkladu:

```
string ahoj;
int pocet;

ahoj = "Ahoj světe!"
pocet = 4;

Console.WriteLine(ahoj + " Už po " + pocet + ".");
```

Všimněte si, že v příkladu se míchá dokonce i typ `int`. Jak to, že to funguje? Proměnná `pocet` je objekt typu `int`, který obsahuje mimo jiné metodu `ToString()`, která se použije, když je potřeba.

Pokud se kombinuje více proměnných s textem, je argument metody `WriteLine()` poněkud nepřehledný. Zejména proto, jak se neustále spojují jednotlivé části pomocí operátoru `+` a potom ty uvozovky.

Jiný způsob

Pro výpis proměnných existuje proto ještě další způsob, který může být přehlednější. Vizte ukázku:

```
string ahoj;
int pocet;

ahoj = "Ahoj světe!"
pocet = 4;

Console.WriteLine("{0} Už po {1}.", ahoj, pocet);
```

Do složených závorek se uvede pořadí proměnné, které jsou umístěny za úvodním řetězcem, jako další argumenty funkce. Pozor, čísluje se od nuly.

Proměnné - cvičení

Název školy	Vyšší odborná škola obalové techniky a Střední škola, Štětí, příspěvková organizace		
Adresa školky	Kostelní 134, 411 08 Štětí		
IČ	46773509		
Název operačního programu	OP Vzdělávání pro konkurenceschopnost		
Registrační číslo	CZ.1.07/1.5.00/34.1006		
Označení vzdělávacího materiálu	VY_32_INOVACE_23_PRG_448		
Název tématické oblasti (sady)	Programování		
Název materiálu	Proměnné - cvičení		
Anotace	Soubor úloh pro procvičování práce s proměnnými – deklarace, přiřazení hodnoty a výstup.		
Autor	Ivan Pomykacz	**Jazyk**	čeština
Očekávaný výstup	Deklaruje proměnné. Volí vhodné datové typy. Provádí jednoduché matematické operace s číselnými proměnnými. Vypisuje informace na obrazovku. Chápe sekvenčnost vykonávání kódu.		
Klíčová slova	proměnná, cvičení, deklarace, přiřazení hodnoty, datové typy		
Druh výukového zdroje	Pracovní list	**Věková skupina žáků**	15+
Typ interakce	aktivita	**Ročník**	1.
Speciální vzdělávací potřeby	žádné		
Zhotoveno, (datum/období)	20.09.2013	**Celková velikost**	

Cvičení I

Obsah

Příklad

```
using System;

namespace Promenne02
{
    class MainClass
    {
        public static void Main (string[] args)
        {
            /*
             * Cvičení I
             *
             * Do proměnných "x" a "y" vložte lib. celá čísla.
             * Do proměnné "z" uložte součet "x" a "y".
             * Vypište proměnnou "z"
             *
             */

            // 1. deklarece prom.
            int x;
            int y;
            int z;

            // 2. přiřazení hodn.
            x = 12;
            y = 30;

            // 3. operace
            z = x + y;

            // 4. výpis (výstup)
            Console.WriteLine("Výsledek = " + z);

            /*
             * Cvičení II
             *
             * Do proměnné "a" celočíselného typu uložte lib. číslo.
             * Do proměnné "b" desetinného typu uložte lib. desetinné číslo.
             * Do proměnné "c" uložte součet proměnných a+b
             * Vypište proměnnou "c"
             */

            int a = 3;
```

```
            double b = 3.14;
            // int c; nelze použít, protože výsledný datový typ musí být
stejný nebo vyšší než ten, který může objevit na výstupu operace
            // je to stejné jako s množinami čísel, desetinná čísla mohou
obsahovat i celá čísla, ale u množiny celých čísel nemohou být čísla
desetinná
            double c;

            // sečíst a+b a uložit do "c"
            // vypsat "c"

            c = a + b;

            Console.WriteLine("Výsledek = " + c);


            Console.Write("Press any key to continue . . . ");
            Console.ReadKey(true);
        }
    }
}
```

Cvičení II - Restaurace v1.0

Obsah

Příklad

```
using System;

namespace Restaurace
{
    class MainClass
    {
        public static void Main (string[] args)
        {
            /**
             * Vytvořte program, který si bude pamatovat název a cenu jídla.
             * Přičemž cena jídla je bez DPH.
             * Dále bude mít program uloženou hodnotu DPH.
             *
             * Program bude počítat cenu jídla včetně DPH.
             *
             * Nakonec zahrňte do výpočtu spropitné 10% z ceny jídla (včetně
DPH).
             *
```

```
             * Vytvořte podrobný výstup kalkulace.
             */

            Console.WriteLine("Program: Restaurace\n");

            string nazev; // název jídla
            double jidlo; // cena jídla v €
            double dph;
            double cena;
            double tip = 0.10;
            double cena_s_dph;

            nazev = "Palačinky";
            jidlo = 4.50;
            dph   = 0.20; // 20%

            // cena jídla + dph
            cena_s_dph = jidlo * dph + jidlo;
            // cena jídla + tip
            cena = cena_s_dph * tip + cena_s_dph;

            Console.WriteLine("Jídlo       : " + nazev);
            Console.WriteLine("Cena bez DPH: " + jidlo);
            Console.WriteLine("DPH         : " + (dph*100) + "%");
            Console.WriteLine("Spropitné   : " + (cena_s_dph*tip) + " euro");
            Console.WriteLine("Celkem      : " + cena + " euro");
            Console.WriteLine("\n");

            Console.Write("Press any key to continue . . . ");
            Console.ReadKey(true);
        }
    }
}
```

Test I

Obsah

Zadání

```
using System;

namespace Test
{
    class MainClass
    {
```

```
        public static void Main (string[] args)
        {
            Console.WriteLine ("Program: Download v1.0\n");

            /**
             * Vytvořte program pro výpočet doby stahování dat.
             *
             * Do proměnné uložte objem přenášených dat v MB (např. 1024 MB)
             * Do jiné proměnné uložte rychlost stahování v Mbit/s (např. 10
Mbit/s)
             *
             * Hlavní zádrhel programu bude nejspíše v tom,
             * že velikost se uvádí v bytech a rychlost bitech za vteřinu.
             *
             * Aby bylo možné spočítat dobu stahování je potřeba:
             *   - převést MB na Mbit (1B = 8bit, takže 1024*8)
             *   - provést podíl velikosti dat a rychlosti stahování (zde
8192/10)
             *
             * Vypište výsledek na monitor.
             * Přepočítejte čas ve vteřinách na minuty.
             */


            // deklarace
            double objem;
            double objemBit;
            double rychlost;
            double cas;

            // přiřazení hodnot
            objem = 1024; // velikost v MB
            rychlost = 10; // rychlost v Mbit/s

            // operace
            objemBit = objem * 8;
            cas = objemBit / rychlost;

            // výstup
            Console.WriteLine("Objem dat        : " + objem + " MB");
            Console.WriteLine("Rychlost přenosu : " + rychlost + " Mbit/s");
            Console.WriteLine("Převod MB->Mbit  : " + objemBit + " Mbit");
            Console.WriteLine("-----------------------------");
            Console.WriteLine("Doba stahování   : " + cas + " s");
            Console.WriteLine("                   " + (cas/60) + " m");
        }
    }
}

using System;
```

```
namespace TestB
{
    class MainClass
    {
        public static void Main (string[] args)
        {
            Console.WriteLine ("Program: Upload v1.0\n");

            /**
             * Vytvořte program pro výpočet objemu odesílaných dat.
             *
             * Do proměnné uložte čas v hodinách (např. 1h)
             * Do jiné proměnné uložte rychlost odesílání v Mbit/s (např. 10
Mbit/s)
             *
             * Spočítejte objem přenesených dat za daný čas a při dané
rychlosti.
             *   - převeďte čas na vteřiny (protože rychlost je v Mbit za
vteřinu)
             *   - proveďte vynásobení rychlosti a času (zde 10 * 3600)
             *
             * Vypište výsledek na monitor.
             * Přepočítejte výsledek z Mbit na MB, platí že 1B = 8b, takže
36000/8.
             */


            // deklarace
            double objem;
            double objemBit;
            double rychlost;
            double casH;
            double cas;

            // přiřazení hodnot
            casH = 1; // čas v hodinách
            rychlost = 10; // rychlost v Mbit/s

            // operace
            cas = casH * 60 * 60;
            objemBit = cas * rychlost;
            objem = objemBit / 8;

            // výstup
            Console.WriteLine("Doba nahrávání   : " + casH + " h");
            Console.WriteLine("Převod h->s      : " + cas + " s");
            Console.WriteLine("Rychlost přenosu : " + rychlost + " Mbit/s");
            Console.WriteLine("-----------------------------");
            Console.WriteLine("Objem dat        : " + objemBit + " Mbit");
            Console.WriteLine("                   " + objem + " MB");
```

```
            }
        }
}
```

Vstup

Název školy	Vyšší odborná škola obalové techniky a Střední škola, Štětí, příspěvková organizace		
Adresa školky	Kostelní 134, 411 08 Štětí		
IČ	46773509		
Název operačního programu	OP Vzdělávání pro konkurenceschopnost		
Registrační číslo	CZ.1.07/1.5.00/34.1006		
Označení vzdělávacího materiálu	VY_32_INOVACE_23_PRG_449		
Název tématické oblasti (sady)	Programování		
Název materiálu	Vstup		
Anotace	Text představuje možnost zadávání dat do běžící aplikace pomocí klávesnice. Zavádí pojem návratová hodnota. Rozebírá problematiku při ukládání textových dat a čistě číselných dat z klávesnice do proměnných.		
Autor	Ivan Pomykacz	**Jazyk**	čeština
Očekávaný výstup	Načte stisknuté klávesy a uloží do proměnné. Jde-li o číslo, provede parsování hodnoty, tak aby mohl být použitý vhodný datový typ.		
Klíčová slova	vstup z klávesnice, parsování řetězce, návratová hodnota		
Druh výukového zdroje	Výklad	**Věková skupina žáků**	15+
Typ interakce	aktivita	**Ročník**	1.
Speciální vzdělávací potřeby	žádné		
Zhotoveno, (datum/období)	23.09.2013	**Celková velikost**	

Obsah

Shrnutí

Pod vstupem u programu si můžeme představit cokoli, co se dostalo do našeho programu námi definovaným způsobem. Např. uživatel mohl napsat něco na klávesnici. Stisknuté klávesy se pak předaly do našeho programu a ten na jejich základě provedl nějakou akci. (Klávesnice je typickým vstupním zařízením)

Dalším vstupním zařízením by mohla být myš, nebo čtečka čárových (nebo QR) kódů, nebo i scanner. Jiným vstupem by mohl být soubor, ze kterého bude program načítat data. Dalším pak síťová komunikace, kde program přijímá data po síti (ať už z Internetu nebo lokální sítě).

Console

Zaměřme se na první popisovaný vstup, tj. vstup z klávesnice. V C# je dostupný objekt `Console`, který obsahuje metodu `ReadLine()`.

Metoda `ReadLine()` pozastaví běh programu a čeká, dokud uživatel stiskne klávesu `Enter`.

```
string jmeno;
Console.Write("Napiš tvé jméno: ");

jmeno = Console.ReadLine();

Console.WriteLine("Ahoj " + jmeno + "!");
Console.WriteLine("Bohužel neumím skloňovat ...");
```

Všimněte si, jak je v ukázce metoda `Console.Write()` a `Console.WriteLine()`. Při použití `Write()` nedojde k odřádkování a uživatel pak zadává své jméno ihned za dvojtečku (v řetězci metody `Write()`).

Návratová hodnota

Co je kritické k pochopení u tohoto příkladu je to, že metoda `ReadLine()`, vůbec jako první námi použitá metoda, vrací nějakou hodnotu. V tomto případě jde o text, který napsal uživatel na klávesnici.

Aniž bychom cokoli vysvětlili, automaticky je v příkladu použitá proměnná typu `string`. Automaticky proto, že cokoli co napíšeme na klávesnici je vlastně text, a to i v případě, že napíšeme nějaké číslo. Prostě vše, co napíšete na klávesnici je text, a tedy typ `string`.

Metody mohou, ale nemusí mít návratovou hodnotu. Např. metoda `WriteLine()` nemá žádnou návratovou hodnotu (přesněji řečeno má, nazývá se `void`).

To, jestli a jakou má metoda návratovou hodnotu se lze dočíst v referenční příručce jazyka (hledejte

klíčové slovo *return*.

Vstup a číslo

Z toho, co bylo právě popsáno **není možné použít následující** kód:

```
double cislo;
Console.Write("Zadej číslo: ");
cislo = Console.ReadLine();
```

Kompilátor okamžitě zahlásí chybu, že není možné přetypovat proměnnou `string` na typ `int`.

Jenomže mi potřebujeme pracovat s čísly, a chceme čísla zadávat z klávesnice.

Proto existují metody, které dokáží analyzovat textový řetězec, a pokud v něm rozpoznají číslo, tak provedou jakousi konverzi na požadovaný datový typ (int nebo double). Vizte následující ukázku:

```
int cislo;
string text;

Console.Write("Zadej celé číslo: ");
text = Console.ReadLine();

cislo = int.Parse(text);
```

Nejprve jsme do proměnné `text` uložili řetězec, který pravděpodobně bude obsahovat číslo. Následně jsme použili metodu `int.Parse()`, která má vstupní argument typu `string` a vrací typ `int`.

Celý příklad lze ještě zjednodušit o to, že nebudeme zbytečně deklarovat proměnnou typu `string`.

```
int cislo;

Console.Write("Zadej celé číslo: ");
cislo = int.Parse(Console.ReadLine());
```

Obdobně bychom mohli parsovat čísla na typ `double`. V příkladu výše stačí nahradit int za double.

Pozor! Metoda `Parse()` může též tzv. "vyhodit (throw)" výjimku (*exception*). To se může stát např. tehdy, zadáte-li takový řetězec, který neobsahuje číslo (nebo číslo jinak zkomolené). Prozatím nechme tento stav neošetřený, a vrátíme se k němu později.

Vstup - cvičení

Název školy	Vyšší odborná škola obalové techniky a Střední škola, Štětí, příspěvková organizace		
Adresa školky	Kostelní 134, 411 08 Štětí		
IČ	46773509		
Název operačního programu	OP Vzdělávání pro konkurenceschopnost		
Registrační číslo	CZ.1.07/1.5.00/34.1006		
Označení vzdělávacího materiálu	VY_32_INOVACE_23_PRG_450		
Název tématické oblasti (sady)	Programování		
Název materiálu	Vstup - cvičení		
Anotace	Soubor úloh pro procvičování práce s proměnnými a vstupem z klávesnice.		
Autor	Ivan Pomykacz	**Jazyk**	čeština
Očekávaný výstup	Tvoří jednoduché, jednoúčelové aplikace, které dokáží reagovat na vstup z klávesnice a ovlivňovat tak výsledek výpočtu.		
Klíčová slova	vstup z klávesnice, parsování řetězce, návratová hodnota, výstup		
Druh výukového zdroje	Pracovní list	**Věková skupina žáků**	15+
Typ interakce	aktivita	**Ročník**	1.
Speciální vzdělávací potřeby	žádné		
Zhotoveno, (datum/období)	25.09.2013	**Celková velikost**	

Cvičení III - Restaurace v2.0

Obsah

Příklad

```
using System;

namespace Restaurace2
{
    class MainClass
    {
        public static void Main (string[] args)
        {
            Console.WriteLine("Program: Restaurace v2.0");

            /**
             * Vytvořte program, který
             * - vyzve uživatele k zadání názvu jídla
             * - vyzve uživatele k zadání ceny jídla bez DPH
             * - vyzve k zadání DPH
             *
             * - následně spočítá cenu jídla včetně DPH
             * - zobrazí podrobný výpis
             *   Jídlo        : Palačinky
             *   Cena bez DPH : 4.50
             *   DPH          : 21%
             *   ...
             */

            // 1. deklarace prom.
            string nazev;
            double cena; // cena jídla bez dph
            double dph; // dph
            double celkem; // celková cena

            // 2. vstup
            Console.Write("Zadej název jídla: ");
            nazev = Console.ReadLine();

            Console.Write("Zadej cenu jídla: ");
            cena = double.Parse(Console.ReadLine());

            Console.Write("Zadej DPH (v %): ");
            dph = double.Parse(Console.ReadLine());
            dph = dph / 100.0;
            // 3. operace
            celkem = cena * dph + cena;
```

```
            // 4. výpis
            Console.WriteLine("Jídlo        : " + nazev);
            Console.WriteLine("DPH          : " + (dph * 100) + "%");
            Console.WriteLine("Celkem       : " + celkem);

            Console.Write("Press any key to continue . . . ");
            Console.ReadKey(true);
        }
    }
}
```

Cvičení IV - Kalkulačka v1.0

Obsah

Příklad

```
using System;

namespace Kalkulacka1
{
    class MainClass
    {
        public static void Main (string[] args)
        {
            Console.WriteLine("Program: Kalkulačka v1.0\n");
            // "\n" je sekvence pro klávesu Enter - provede odřádkování

            /**
             * Nechejte uživatele po sobě zadat dvě celá čísla.
             *   Console.ReadLine()
             * Následně tato čísla sečtěte.
             *
             * Výsledek vypište na obrazovku.
             *   Console.WriteLine("...")
             *
             * Rozšiřte program o: *, /, -
             */

            // 1. deklarace prom.
            int x;
            int y;
            int vysK;
            int vysP;
```

```
            int vysM;
            int vysD;

            // 2. přiřazení hodnot
            Console.Write("Zadej číslo: ");
            x = int.Parse(Console.ReadLine());

            Console.Write("Zadej další číslo: ");
            y = int.Parse(Console.ReadLine());

            // 3. operace
            vysP = x + y;
            vysM = x - y;
            vysK = x * y;
            vysD = x / y;

            // 4. výstup
            Console.WriteLine(x + " + " + y + " = " + vysP);
            Console.WriteLine(x + " - " + y + " = " + vysM);
            Console.WriteLine(x + " * " + y + " = " + vysK);
            Console.WriteLine(x + " / " + y + " = " + vysD);

            Console.WriteLine("\n");

            Console.Write("Press any key to continue . . . ");
            Console.ReadKey(true);
        }
    }
}
```

Test II

Obsah

Zadání

Přepínač

Název školy	Vyšší odborná škola obalové techniky a Střední škola, Štětí, příspěvková organizace		
Adresa školky	Kostelní 134, 411 08 Štětí		
IČ	46773509		
Název operačního programu	OP Vzdělávání pro konkurenceschopnost		
Registrační číslo	CZ.1.07/1.5.00/34.1006		
Označení vzdělávacího materiálu	VY_32_INOVACE_23_PRG_451		
Název tématické oblasti (sady)	Programování		
Název materiálu	Přepínač		
Anotace	Představení řídící konstrukce, která dokáže ovlivňovat, které bloky kódu se vykonají, a které ne. Na základě vstupního výrazu se provede požadovaná akce.		
Autor	Ivan Pomykacz	**Jazyk**	čeština
Očekávaný výstup	Píše komplexnější programové konstrukce. Chápe sekvenčnost vykonávání kódu a možnosti některé části přeskočit pomocí řídící konstrukce na základě vstupního výrazu.		
Klíčová slova	řídící konstrukce, přepínač		
Druh výukového zdroje	Pracovní list	**Věková skupina žáků**	15+
Typ interakce	aktivita	**Ročník**	1.
Speciální vzdělávací potřeby	žádné		
Zhotoveno, (datum/období)	30.09.2013	**Celková velikost**	

Switch

Obsah

Shrnutí

Řídící konstrukce umožňují usměrňovat tok kódu, tj. vykonávání některých příkazů a jiné třeba přeskočit (vynechat). Jsou to právě tyto konstrukce, díky kterým můžeme tvořit různé algoritmy a výsledné programy pak skutečně něco "dělají".

switch

Konstrukt přepínače (switch) je na syntaxi poněkud těžkopádnější. Ale protože ještě neznáme jiné konstrukce, tak není s čím porovnávat.

Přepínač uvozujeme klíčovým slovem **switch()** společně s kulatými závorkami, do kterých se zapisuje *condition* podmínka, neboli výraz, který se bude vyhodnocovat.

Následuje tělo switche, které je uzavřeno do složených závorek { }.

Všimněte si, že zatím žádný středník.

```
switch (1) {

}
```

Do těla switche se vkládají tzv. **case** (případ). Switch pak na základě podmínky rozhodne, který case (případ) vykoná. Přičemž u každého case je uvedená hodnota (v příkladu níže to jsou čísla 1 a 2), pro kterou existuje případ.

Každý case je ukončen příkazem **break;**.

```
switch (1) {
case 1:
    Console.WriteLine("Volba 1");
    break;
case 2:
    Console.WriteLine("Volba 1");
    break;
default:
    Console.WriteLine("Neznámá volba");
    break;
}
```

Je-li tedy výraz ve switchi (v kulatých závorkách) roven hodnotě 1, pak by program při spuštění

vypsal text "Volba 1"

Právě uvedený příklad tak trochu pozbývá smysl. V podmínce switche je totiž hodnota 1 a vždy tam bude, tudíž se na ostatní případy nikdy nedostane.

Poslední blok v těle konstruktu switch je blok **default**. Je-li uveden (není totiž povinný), pak se provede vždy, když žádný case neodpovídá zadané podmínce. Pokud byste změnili hodnotu v podmínce switche z 1 na 3, pak by se provedl právě blok default.

Příklad

Následující ukázka ukazuje poněkud smysluplnější využití switche. Uživatel je nejprve vyzván k zadání písmene "a" nebo "b" a podle této volby pak switch rozhodne, který blok kódu provede.

```
using System;

namespace Prepinac
{
    class MainClass
    {
        public static void Main (string[] args)
        {
            Console.WriteLine("Program: Přepínač v1.0\n");

            string volba;
            Console.Write("Zvol buď \"a\" nebo \"b\": ");
            volba = Console.ReadLine();

            switch (volba) {
            case "a":
                Console.WriteLine("Áčko");
                break;

            case "b":
                Console.WriteLine("Béčko");
                break;

            default:
                 Console.WriteLine("Neznám");
                 break;
            }

            Console.Write("Press any key to continue . . . ");
            Console.ReadKey(true);
        }
    }
}
```

Cvičení V - Kalkulačka v2.0

Obsah

Příklad

```
using System;

namespace Kalkulacka2
{
    class MainClass
    {
        public static void Main (string[] args)
        {
            Console.WriteLine("Program: Kalkulačka v2.0\n");

            /**
             * Vytvořte program, který vyzve uživatele k zadání
             * dvou čísel a početní operace (+,-,*,/)
             *
             * Následně program vyhodnotí pomocí switche jakou
             * operaci má vykonat a provede odpovídající výpočet.
             *
             * Program zobrazí výsledek výpočtu na monitor.
             */

            double x;
            double y;
            double vys;
            string operace;

            Console.Write("Zadej operaci (+,-,*,/): ");
            operace = Console.ReadLine();

            Console.Write("Zadej číslo: ");
            x = double.Parse(Console.ReadLine());

            Console.Write("Zadej druhé číslo: ");
            y = double.Parse(Console.ReadLine());

            switch (operace) {
            case "+":
                vys = x + y;
                Console.WriteLine(x + " + " + y + " = " + vys);
                break;
            case "-":
                vys = x - y;
                Console.WriteLine(x + " - " + y + " = " + vys);
```

```
                break;
            case "*":
                vys = x * y;
                Console.WriteLine(x + " * " + y + " = " + vys);
                break;
            case "/":
                vys = x / y;
                Console.WriteLine(x + " / " + y + " = " + vys);
                break;
            default:
                Console.WriteLine("Neznámý operátor!!!");
                break;
            }

            Console.Write("Press any key to continue . . . ");
            Console.ReadKey(true);
        }
    }
}
```

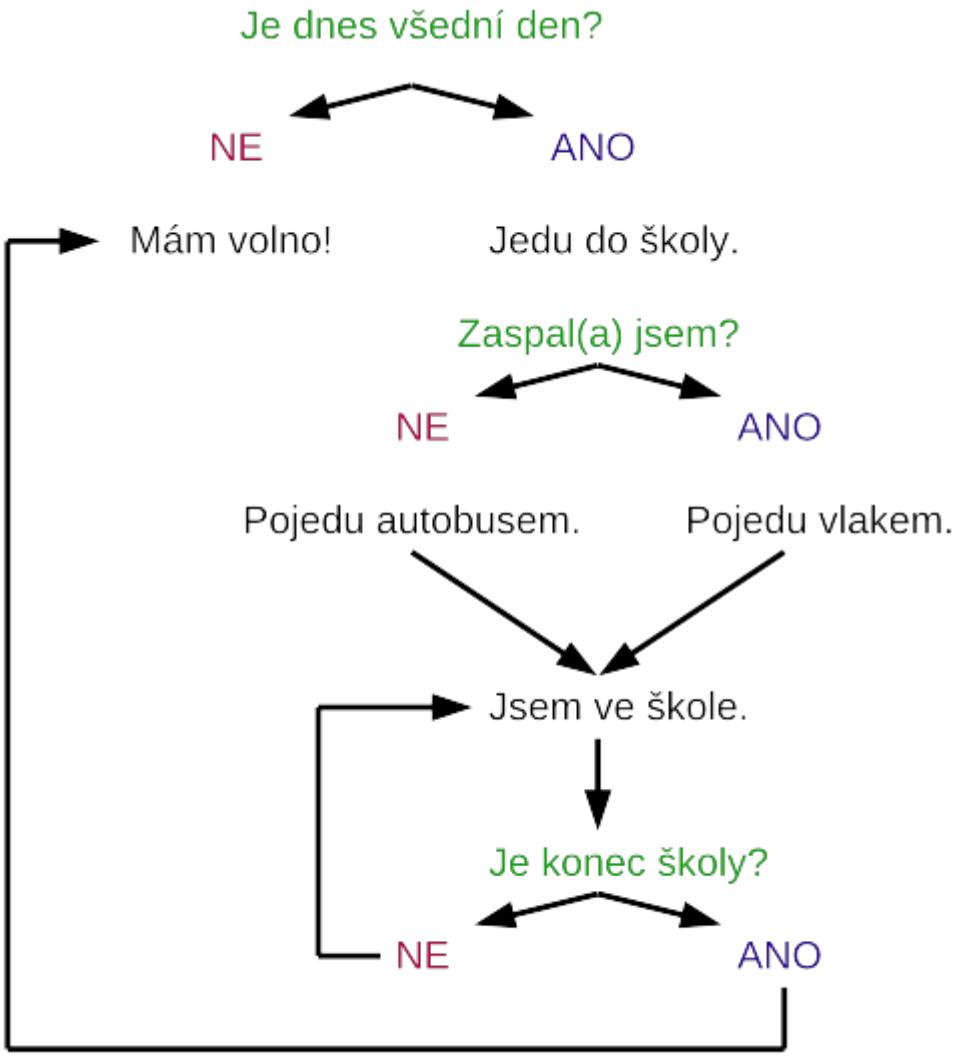
Co když

Název školy	Vyšší odborná škola obalové techniky a Střední škola, Štětí, příspěvková organizace		
Adresa školky	Kostelní 134, 411 08 Štětí		
IČ	46773509		
Název operačního programu	OP Vzdělávání pro konkurenceschopnost		
Registrační číslo	CZ.1.07/1.5.00/34.1006		
Označení vzdělávacího materiálu	VY_32_INOVACE_23_PRG_452		
Název tématické oblasti (sady)	Programování		
Název materiálu	Co když		
Anotace	Text uvádí řídící konstrukci if-else. Na pseudo vývojovém diagramu demonstruje větvení kódu. Zavádí operátory porovnání, které bývají součástí výrazů v podmínkách. Předkládá cvičení na pravdivostní hodnoty true a false. Nakonec předkládá příklady zápisu řídící konstrukce v základní podobě a následně rozšířené o bloky else-if-else.		
Autor	Ivan Pomykacz	**Jazyk**	čeština
Očekávaný výstup	Chápe význam hodnot true a false při vyhodnocování výrazů. Sestavuje jednoduché podmínky pro ovlivnění způsobu vykonávání kódu. Ovlivňuje, který kód se dle vyhodnocení výrazu vykoná a který ne.		
Klíčová slova	podmínka, výraz, větvení kódu		
Druh výukového zdroje	Výklad	**Věková skupina žáků**	15+
Typ interakce	aktivita	**Ročník**	1.
Speciální vzdělávací potřeby	žádné		
Zhotoveno, (datum/období)	04.10.2013	**Celková velikost**	

Obsah

Shrnutí

Řídící konstrukce umožňují usměrňovat tok kódu, tj. vykonávání některých příkazů a jiné třeba přeskočit (vynechat). Jsou to právě tyto konstrukce, díky kterým můžeme tvořit různé algoritmy a výsledné programy pak skutečně něco "dělají".



Operátory porovnání

Používají se v podmínkách při testování nějakého stavu, např. je-li něco rovno něčemu jinému a nebo je-li něčeho více, atp.

Patrně je znáte už z matematiky, ale přeci jen pro úplnost toho, jak vypadají v jazyke C#.

- `==` .. rovná se
- `!=` .. je různý od (nerovná se)
- `>` .. je větší
- `>=` .. je větší nebo rovno
- `<` .. je menší
- `<=` .. je menší nebo rovno

Pravda nepravda

- Pokuste se přiřadit hodnotu *True* nebo *False* na základě následujících výrazů:
```
bool bool1;
bool bool2;
bool bool3;
bool bool4;
bool bool5;

// 17 < 18/100
bool1 =

// 100 == 33 * 3 + 1
bool2 =

// 19 <= Math.Pow(2,4)
bool3 =

// -22 >= -18
bool4 =

// 99 != 98 + 1
bool5 =

Console.WriteLine(bool1);
Console.WriteLine(bool2);
Console.WriteLine(bool3);
Console.WriteLine(bool4);
Console.WriteLine(bool5);
```

- Pokuste se zjistit hodnotu *True* nebo *False* na základě poněkud složitějších výrazů:
```
bool bool1;
bool bool2;
bool bool3;
bool bool4;
bool bool5;

// 20 + -10 * 2 > 10 / 3 / 2
bool1 =

// Math.Pow(10 + 17, 2) == Math.Pow(3,6)
bool2 =

// Math.Pow(1,Math.Pow(2,3)) <= -(-(-1))
```

```
bool3 =

// 40 / 20 * 4 >= Math.Pow(-4,2)
bool4 =

// Math.Pow(100,0.5) != 6 + 4
bool5 =

Console.WriteLine(bool1);
Console.WriteLine(bool2);
Console.WriteLine(bool3);
Console.WriteLine(bool4);
Console.WriteLine(bool5);
```

Příklad

Cvičení výše bylo na pochopení stavů *Pravda*, *Nepravda*, a toho, že není nic mezi tím. Pokračujme praktičtějším příkladem.

Od uživatele si necháme z klávesnice zadat číslo. A následně rozhodneme, zda-li toto číslo je kladné nebo záporné, případně rovno 0.

```
int x;

Console.Write("Zadej celé číslo: ");
x = int.Parse(Console.ReadLine());

if (x > 0) {
    Console.WriteLine("Číslo je kladné");
}

if (x < 0) {
    Console.WriteLine("Číslo je záporné");
}

if (x == 0) {
    Console.WriteLine("Číslo je nula");
}
```

Konstrukce if

Konstrukt *if* začíná právě slovem `if ()` společně s kulatými závorkami, do kterých se zapisuje

výraz, jenž se bude vyhodnocovat.

```
if (vyraz) {
    // příkazy, které se mají provést, je-li výsledek výrazu true (pravda)
}
```

Ihned za kulatými závorkami následují složené závorky `{` a `}`, které ohraničují příkazy, které se vykonají pouze tehdy, je-li výraz v kulatých závorkách vyhodnocen jak `true` (a je to tedy typ *bool*).

Výrazem tedy může být téměř cokoli, co vrátí typ bool (tedy true, false):

- proměnná typu bool;
- metoda, která vrací typ bool;
- výraz, jehož součástí je alespoň jeden z operátorů porovnání (viz výše);
- jeden z booleanovských operátorů (and - `&&`, or `||`);

Konstrukce if-else

Konstrukt lze ještě rozšířit o blok `else`. Stojí v podstatě v opozici proti tvrzení `if`.

```
if (vyraz) {
    Console.Write("Výsledek výrazu je true");
}
else {
    Console.Write("Výsledek výrazu je false");
}
```

Blok `else` může být jen jeden a vždy na konci. Proč? No, buď je výraz pravdivý, a pak se vykonají příkazy uvnitř bloku `if`, a nebo pravdivý není, nic mezi tím není.

Muž a žena

Mějme program, který vyhodnocuje zadané údaje, konkrétně, zda-li uživatel vyplnil správně informaci o tom, zda-li je "muž" nebo "žena".

Pokud souhlasíte, tak existují pouze tyto dva stavy: *je-li uživatel muž, nemůže být žena* a obráceně, *je-li uživatel žena, nemůže být muž*.

Potom, mějme proměnnou `volba`, do které budeme ukládat uživatelovu odpověď, tedy to, co napíše na klávesnici. A podle výše domluveného, platí-li výraz `volba == "muž"`, pak jde o muže. Jelikož další jediná možná odpověď je "žena", můžeme kód zapsat např. takto:

```
string volba;

Console.Write("Jsi muž nebo žena? Odpověz \"muž\" nebo \"žena\": ");
volba = Console.ReadLine();

if (volba == "muž") {
    Console.WriteLine("Eviduji muže.");
}
else {
```

```
    Console.WriteLine("Eviduji ženu.");
}
```

Všimněte si, že testujeme pouze to, zda-li proměnná obsahuje řetězec "muž". Pokud výraz není pravda, tzn. proměnná obsahuje jiný řetězec, uživatel bude automaticky vyhodnocen jako "žena". Vyzkoušíte-li výše uvedený kód, tak zjistíte, že pokud například napíšete do odpovědi třeba jen "Muž" s velkým počátečním písmenem, dojde ke konstatování "Eviduji ženu".

Jak z toho ven? Mohli bychom blok `else` vynechat a napsat další `if` pro detekci řetězce `volba == "žena"`.

```
if (volba == "muž") {
    Console.WriteLine("Eviduji muže.");
}

if (volba == "žena") {
    Console.WriteLine("Eviduji ženu.");
}
```

A tím jsme ošetřili chybné vyhodnocení muže a ženy.

Ale co se teď stane se všemi chybnými odpověďmi? Tedy s takovými odpověďmi, které obsahují překlepy nebo schválně nesmyslná slova?

Program by měl být schopen tyto vstupy rozpoznat a adekvátně na ně zareagovat, aby se uživatel dozvěděl, zda-li to, co zadal, je OK.

V takových případech se hodí provést tzv. zřetězení bloků `if` do jednoho mega bloku.

```
string volba;

Console.Write("Jsi muž nebo žena? Odpověz \"muž\" nebo \"žena\": ");
volba = Console.ReadLine();

if (volba == "muž") {
    Console.WriteLine("Eviduji muže.");
}
else if (volba == "žena") {
    Console.WriteLine("Eviduji ženu.");
}
else {
    Console.WriteLine("Chyba zadání.");
}
```

Všimněte si, že uvedená konstrukce obsahuje dvakrát slovo `else`, ale poprvé na `else` navazuje `if`.

Jak se to pak celé chová?

- **Jestliže [if]** `volba == "muž"`, vypiš "Eviduji muže",
- **jinak jestliže [else if]** `volba == "žena"`, vypiš "Eviduji ženu",
- **jinak [else]** vypiš "Chyba zadání".

Co když - cvičení

Název školy	Vyšší odborná škola obalové techniky a Střední škola, Štětí, příspěvková organizace		
Adresa školky	Kostelní 134, 411 08 Štětí		
IČ	46773509		
Název operačního programu	OP Vzdělávání pro konkurenceschopnost		
Registrační číslo	CZ.1.07/1.5.00/34.1006		
Označení vzdělávacího materiálu	VY_32_INOVACE_23_PRG_453		
Název tématické oblasti (sady)	Programování		
Název materiálu	Co když - cvičení		
Anotace	Soubor úloh pro procvičování práce s řídícími konstrukcemi.		
Autor	Ivan Pomykacz	**Jazyk**	čeština
Očekávaný výstup	Tvoří výrazy. Tvoří jednoduché, jednoúčelové aplikace, které dokáží podmíněně vykonávat konkrétní bloky kódu dle vyhodnoceného výrazu. Navrhuje a realizuje vlastní větvení kódu.		
Klíčová slova	podmínka, výraz, větvení kódu		
Druh výukového zdroje	Pracovní list	**Věková skupina žáků**	15+
Typ interakce	aktivita	**Ročník**	1.
Speciální vzdělávací potřeby	žádné		
Zhotoveno, (datum/období)	07.10.2013	**Celková velikost**	

Cvičení VI - Kalkulačka v3.0

Obsah

Příklad

```
using System;

namespace Kalkulacka3
{
    class MainClass
    {
        public static void Main (string[] args)
        {
            Console.WriteLine("Program: Kalkulačka v3.0\n");

            /**
             * Vytvořte program podobný Kalkulačce v2.0, ale
             * na místo konstruktu switch použijte if-else
             *
             * V programu bude navíc ošetřeno dělení nulou.
             *
             */

            double x;
            double y;
            double vysledek;
            string op;

            Console.Write("Zadej číslo: ");
            x = double.Parse(Console.ReadLine());

            Console.Write("Zadej číslo: ");
            y = double.Parse(Console.ReadLine());

            Console.Write("Zadej operaci (*,/,+,-): ");
            op = Console.ReadLine();

            if (op == "+") {
                vysledek = x + y;
                Console.WriteLine(x + " " + op + " " + y + " = " + vysledek);
                //Console.WriteLine("{0} {1} {2} = {3}", x, op, y, vysledek);
            }
            else if (op == "-") {
                vysledek = x - y;
                Console.WriteLine(x + " " + op + " " + y + " = " + vysledek);
            }
            else if (op == "*") {
```

```
                vysledek = x * y;
                Console.WriteLine(x + " " + op + " " + y + " = " + vysledek);
            }
            else if (op == "/") {
                if (y == 0) {
                    Console.WriteLine("Zadané číslo nemůže být rovno 0.");
                }
                else {
                    vysledek = x / y;
                    Console.WriteLine(x + " " + op + " " + y + " = " +
vysledek);
                }
            }
            else {
                Console.WriteLine("Neznámý operátor!");
            }

            Console.Write("Press any key to continue . . . ");
            Console.ReadKey(true);
        }
    }
}
```

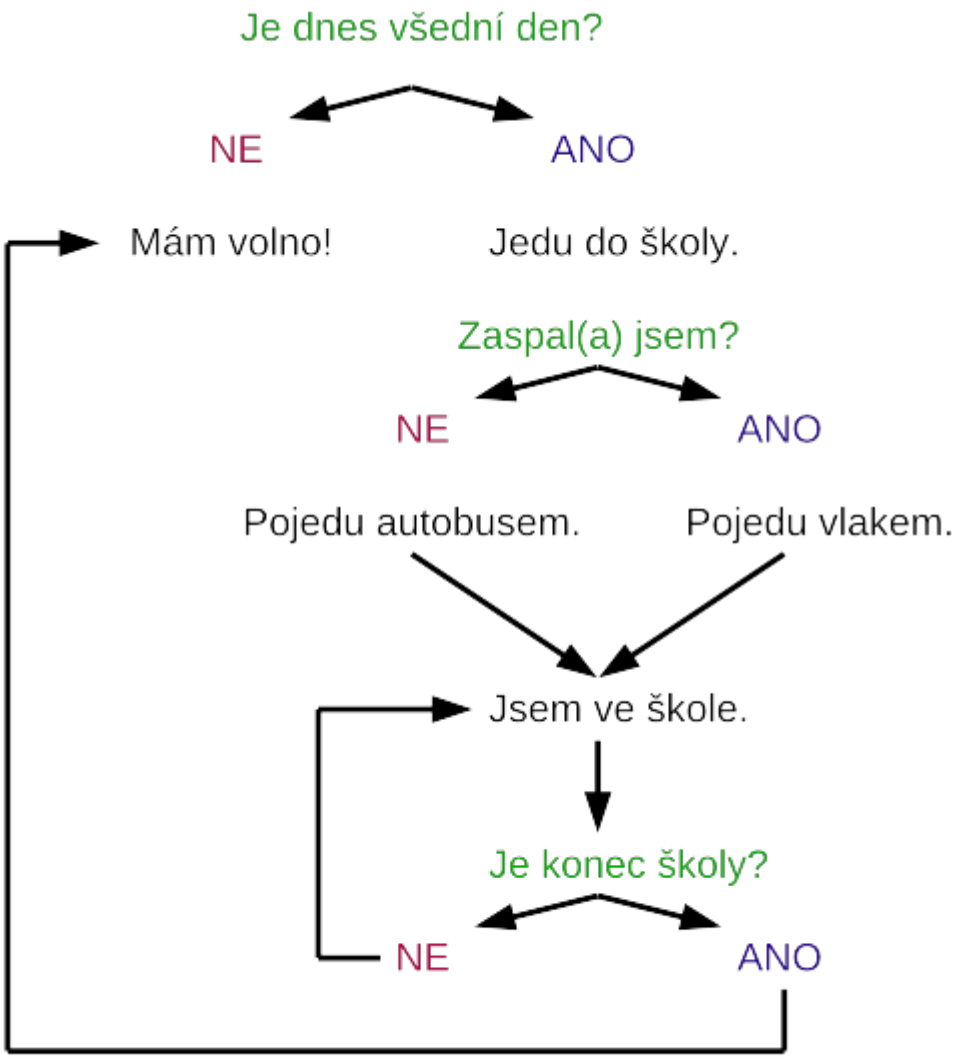
Cvičení VII - Diagram 1

Obsah

Příklad

Proveďte implementaci diagramu na obrázku níže.

Použijte dosavadní znalosti konstrukce `if-else`.



Řešení

```
Console.WriteLine ("Program: Do školy v1.0");

Console.WriteLine("Na otázky odpovídej \"ano\" nebo \"ne\".");

string volba;

Console.Write("Je dnes všední den? ");
volba = Console.ReadLine();
if (volba == "ano") {
    Console.WriteLine("Jedu do školy.");
    Console.Write("Zaspal(a) jsem? ");
    volba = Console.ReadLine();
    if (volba == "ano") {
        Console.WriteLine ("Pojedu vlakem.");
    }
    else if (volba == "ne") {
        Console.WriteLine ("Pojedu autobusem.");
    }
```

```
    else {
        Console.WriteLine ("Neumím rozhodnout.");
    }

    Console.WriteLine ("Jsem ve škole.");
    Console.Write("Je konec školy? ");
    volba = Console.ReadLine();
    if (volba == "ano") {
        Console.WriteLine("Mám volno!");
    }
    else if (volba == "ne") {
        Console.WriteLine("Jsem stále vše škole!");
    }
    else {
        Console.WriteLine ("Nevím, co se děje.");
    }
}
else if (volba == "ne") {
    Console.WriteLine("Mám volno!");
}
else {
    Console.WriteLine("Neplatná volba!");
}
```

Cvičení VIII - Diagram 2

Obsah

Pomocné metody

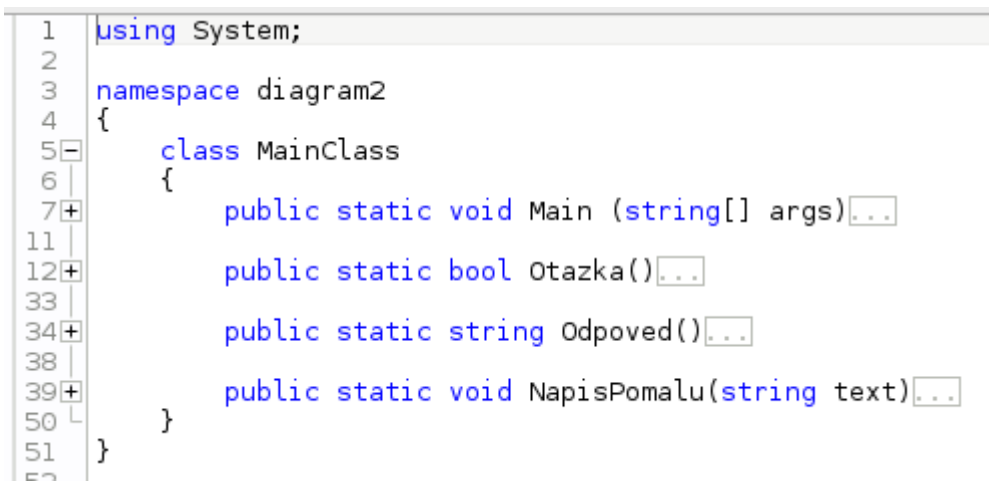
Následující kód obsahuje tři pomocné metody.

Celý kód zkopírujte a vložte do svého projektu takovým způsobem, aby nově vložený kód byl na stejné úrovni, jako je metoda `Main`.

```
public static bool Otazka() {
    string klav;
    bool odpoved;

    while (true) {
        klav = Odpoved();
        if (klav == "ano") {
            odpoved = true;
```

```
            break;
        }
        else if (klav == "ne") {
            odpoved = false;
            break;
        }
        else {
            NapisPomalu("Nerozumím, zadej buď \"ano\" nebo \"ne\".");
        }
    }

    return odpoved;
}

public static void Pauza(int doba = 200) {
    Thread.Sleep(doba);
}

public static string Odpoved() {
    Console.Write(": ");
    return Console.ReadLine();
}

public static void NapisPomalu(string text) {
    int rychlost = 100;

    foreach (char znak in text.ToCharArray()) {
        if (rychlost > 50) {
            Thread.Sleep(rychlost);
        }
        Console.Write(znak);
    }
    Console.Write("\n");
}
```

Pokud použijete sbalovací tlačítka vlevo od kódu `-`, na všechny metody, měli byste vidět podobný přehled, jako na obrázku. (Na obrázku už je kód i zarovnaný)



```
1  using System;
2
3  namespace diagram2
4  {
5      class MainClass
6      {
7          public static void Main (string[] args)...
11
12         public static bool Otazka()...
33
34         public static string Odpoved()...
38
39         public static void NapisPomalu(string text)...
50     }
51 }
```

Všimněte si hlavně mezi jakými závorkami je kód vložený. Až doposud jsme totiž vkládali kód jen do metody `Main`.

Using

Nakonec ještě dopište do horní části kódu, hned za `using System;` nový řádek: `using System.Threading;`.

```
using System;
using System.Threading;
```

Otestujte klávesou F8, zda je vše OK.

Příklad

Vytvořte si vlastní diagram s otázkami a odpověďmi. Následně proveďte jeho implementaci s využitím metod `NapisPomalu` a `Otazka`.

Řešení

```
NapisPomalu("Program: Pomalu v1.0");

bool odp;

NapisPomalu("Píše se ti rychle?");
odp = Otazka();
if (odp == true) {
    NapisPomalu("Tak OK, není co řešit");
}
else {
    NapisPomalu("A chceš zrychlit?");
    odp = Otazka();
    if (odp == true) {
        NapisPomalu("Co psaní všemi deseti?");
        odp = Otazka();
        if (odp == true) {
            NapisPomalu("Na netu je celkem dost kurzů i zdarma.");
        }
        else {
            NapisPomalu("Je to moc těžký?");
            odp = Otazka();
            if (odp == true) {
                NapisPomalu("Jo, souhlasím.");
            }
            else {
                NapisPomalu("No, pokud si o nějaký prst přišel, tak mě to
mrzí.");
            }
        }
```

```
    }
    else {
        NapisPomalu("Hmm, tak nic.");
    }
}
```

Test III

Obsah

Test

Vytvořte program "Test", který uživateli předloží několik (min 4) testových otázek. Jde o formu klasikého (někdy i zatracovaného) testu, kde testovaný subjekt má na výběr ze tří (nebo více) variant.

Pro test platí následující pravidla:

- U každé otázky je správná pouze jedna odpověď.
- U každé otázky existuje správná odpověď.
- Za správnou odpověď je možné získat právě jeden bod.
- Není možné se vracet k již vyplněným odpovědím.
- Není možné otázky přeskakovat.
- Test není možné přerušit.

Ukázka testu

Stáhněte si a spusťte ukázku testu (zip verze), abyste věděli, jak by měl program vypadat.

Vyhodnocení testu

Na konci testu se zobrazí celkový počet získaných bodů. Následně se vypíše hodnotící text (ten si můžete vymyslet).

Hodnotící text se váže k počtu bodů:

1. Nikdo není dokonalý ... Ale tohle je kacířství!
2. Mezi nebem a zemí jsou věci, o kterých zatím nevíš.
3. Neutrální ... Nezvažuješ snad temnou stranu síly - nevědomost?
4. Správně, není dobré vynášet před sebe hned všechny karty.

5. Evidentně jsi IT GURU (bow).

Náplň testu

Můžete si vymyslet svůj vlastní test (při dodržení zadání). Nebo můžete použít následující texty:

Co je to CSS?
a) Computer Science Senatorium
b) Common Sense Stubbornness
c) Cascading Style Sheets

Co je to Fedora?
a) Nový engine pro Crysis V (pouze pro PC)
b) Operační systém
c) Zásilková služba (v USA)

Co je to Gnu?
a) Goniometrická funkce (Gunus)
b) Hnutí za globální svobodu
c) Pakůň

001+010+011+100+101+110+111 = ?
a) 28
b) 32
c) 34

Bonus

Můžete použít metody `NapisPomalu()` a `Pauza()`, které jsou uvedeny ve *Cvičení VIII - Diagram 2*.

Odevzdání

Odevzdejte pouze soubor se zdrojovým kódem.

Mini příklady I

Obsah

Cvičení

Úloha 1

Program vyzve uživatele k zadání dvou čísel. Následně program čísla porovná a vyhodnotí, zda-li první číslo je větší, menší nebo rovno druhému z čísel.

Úloha 2

Program vyzve uživatele k zadání čísla od 1 do 5. Zadá-li uživatel číslo mimo zadaný rozsah, program vypíše chybovou hlášku.

Úloha 3

Vytvořte program na výpočet objemu kvádru (`V = a*b*c`). Zajistěte aby program upozornil uživatele na chybně zadanou hodnotu. Chybně zadaná hodnota je:

- pokud je číslo nula;
- je-li číslo záporné.

Nastane nějaká z chyb, program neprovede výpočet.

Úloha 4

Vytvořte program, který bude počítat přibližný čas potřebný k cestování z místa A do místa B.

Program nejprve vyzve uživatele k zadání vzdálenosti mezi místem A a B (km). Následně se dotáže (nechá uživatele vybrat z možností) na způsob dopravy. Možnosti dopravy a jejich průměrné rychlosti budou:

- pěšky (4 km/h);
- autem (70 km/h);
- letadlem (600 km/h);
- teleportem (1 079 252 848 km/h).

Program poté spočítá potřebný čas (v hodinách) k dosažení cíle a vypíše tuto informaci na obrazovku. Pokud by bylo číslo příliš malé, převede se čas na vteřiny.

V programu ošetřete stavy, kde by uživatel mohl zadat nesmyslnou hodnotu (např. záporná vzdálenost). Pokud takový stav nastane program uživatele varuje a skončí nebo jej nechá vložit údaj znovu, záleží na vás. (Vzorec na výpočet potřebného času je `t = s / v`)

Úloha 5

Obyčejný film je promítán o snímkové frekvenci 25 snímků za vteřinu. Pokud film trvá jednu minutu, obsahuje 60*25 snímků (`1min = 60s => 60*25 = 1500`).

Velikost jednoho snímku u full HD rozlišení je 5.9MB (pokud snímek není komprimován).

Nechť program vyzve uživatele k zadání *názvu filmu* a následně jeho *délky v minutách* (např. 2 hodinový film bude mít přibližně 120min).

Program pak spočítá velikost takového filmu (vynásobením velikosti snímku a délkou filmu ve vteřinách a ještě krát počet snímků). Program by měl hlídat takové chybové stavy, kde délka by neměla být záporná.

Následně program nabídne možnost uložení filmu na:

- DVD (8.55 GB);
- Blu-ray (50 GB);
- X-ray (1000 GB).

Bude potřeba převést velikost filmu (která je nyní v MB, na GB, tj. vydělit 1024). Poté vydělením velikost filmu a velikostí média zjistit odpovídající počet médií.

Úloha 6

Program vyzve uživatele k zadání textového řetězce (např jména). Následně program zjistí délku textu (vizte ukázku níže) a rozhodne, zda-li nebyla překročena maximální povolená délka. Maximální povolená délka řetězce je 10 znaků.

```
string s = "Lorem ipsum";
int delka;
// do prom. delka se uloží délka řetězce s
delka = s.Length;
```

Odevzdání

Odevzdejte pouze soubory `.cs`, které vhodně pojmenujete.

Mini příklady II

Obsah

Úloha 1 - Barevná hloubka

Program spočítá na základě počtu barev, kolika bitová barevná hloubka tyto barvy pojme. Např. uživatel zadá, že chce barevnou hloubku pro 16 barev. Program výpočtem zjistí, že tomu odpovídá 4bitová barevná hloubka.

Pro výpočet můžete použít funkci `Math.Log()`, která přebírá dva parametry, číslo a základ logaritmu. Např. u příkladu se 16 barvami bude funkce vypadat `Math.Log(16, 2)`. Číslo 2 je základem proto, že bit může nabývat dvou hodnot.

Program pak bude navíc výsledek zaokrouhlovat, aby nevycházela čísla jako 3.9bit. K tomu můžete použít funkci `Math.Ceiling()`, která přebírá jeden vstupní argument (číslo) a vrací upravenou hodnotu..

Program ohlídá takové stavy, jako třeba záporné hodnoty.

Testovací hodnoty:

- -5
- 15
- 16
- 65000
- 15000000

demo

Úloha 2 - Pixmapa

Program spočítá velikost pixelové mapy (bitmapy) na základě vstupních údajů, které mu předloží uživatel. Program bude potřebovat informaci o tom,

- jaký je rozměr mapy (šířku a výšku),
- jakou má použít barevnou hloubku (kolik bitů).

Výsledek se zobrazí v jednotkách Byte. Pokud ovšem bude číslo příliš veliké, přepočítá se na kB, případně MB. Příliš veliké znamená, že bude 1024 krát větší než je nejbližší násobek.

Např. pixmapa o rozměrech 1024x768 při hloubce 1bit bude zabírat: 98 304 Byte (číslo bylo děleno 8 - bit na Byte).
Tzn., že výsledek je větší jak 1024 Byte, ale menší jak 1024 KiloByte (což by bylo 1 048 576 Byte), proto se číslo převede na kB, tedy 96kB.

Testovací hodnoty (Š x V x H):

- 1024x768x1
- 1920x1080x24

demo

A zároveň

Název školy	Vyšší odborná škola obalové techniky a Střední škola, Štětí, příspěvková organizace		
Adresa školky	Kostelní 134, 411 08 Štětí		
IČ	46773509		
Název operačního programu	OP Vzdělávání pro konkurenceschopnost		
Registrační číslo	CZ.1.07/1.5.00/34.1006		
Označení vzdělávacího materiálu	VY_32_INOVACE_23_PRG_454		
Název tématické oblasti (sady)	Programování		
Název materiálu	A zároveň		
Anotace	Text představuje logický operátor součinu. Na příkladu demonstruje jeho nasazení, výsledkem čehož je přehlednější a čitelnější kód. Jsou ukázána dvě řešení úlohy – pomocí dosavadních znalostí, jejichž výsledkem je neefektivní a nepřehledný kód. Druhé řešení používá logický součin.		
Autor	Ivan Pomykacz	**Jazyk**	čeština
Očekávaný výstup	Přečte výraz poskládaný z logických součinů.		
Klíčová slova	logický součin		
Druh výukového zdroje	Výklad	**Věková skupina žáků**	15+
Typ interakce	aktivita	**Ročník**	1.
Speciální vzdělávací potřeby	žádné		
Zhotoveno, (datum/období)	11.10.2013	**Celková velikost**	

Obsah

Shrnutí

Podmínky je možné spojovat (nikoli řetězit jako v případě `if-else-if`) pomocí logických operátorů AND a OR. To ve výsledku usnadňuje i zpřehledňuje zápis kódu.

Dilema

Žena povídá svému muži, který se živí programováním: „Běž koupit do obchodu 6 rohlíků, a když budou mít vejce, kup jich 20“.

Zřejmě každý správný programátor z výše uvedeného vyvodí, že když budou mít v obchodě alespoň jedno vejce, měl by koupit 20 rohlíků.

Implementace problému

Proveďme deklaraci proměnných, které budou simulovat zboží na krámě. Zrovna tyto proměnné nastavme na kladnou hodnotu, která udává množství daného zboží.

```
int rohliky = 16;
int vejce = 21;

Console.WriteLine(" zboží   |   množství ");
Console.WriteLine("----------------------");
Console.WriteLine("rohlíky  |     {0} ks", rohliky);
Console.WriteLine("vejce    |     {0} ks", vejce);
```

Nyní napíšeme podmínku, která podpoří programátora a pomůže mu se rozhodnout, čeho má kolik koupit.

```
if (vejce > 0) {
    Console.WriteLine("Kup 20 rohlíků.");
    rohliky-= 20;
}
else {
    Console.WriteLine("Kup 6 rohlíků.");
    rohliky-= 6;
}

Console.WriteLine("Zůstatek rohlíků: {0}", rohliky);
```

Podmínka hovoří jasně. Budou-li mít alespoň jedno vejce, kupuji 20 rohlíků. V opačném případě jich beru jen 6.

Na konci je vypsáno, kolik rohlíků zbylo. Patrně jste si všimli zápisu `rohliky -= 20`. Jde o zkrácený zápis `rohliky = rohliky - 20`.

Vylepšení

Co když ale nebudou mít skladem všech 20 rohlíků? Nebo dokonce ani těch 6?

To jsou stavy, které mohou nastat a jen by zbytečně našemu programátorovi popletly hlavu. Proto si pomocí podmínek pohlídáme, že na krámě mají dostatečné množství rohlíků.

```
if (vejce > 0) {
    if (rohliky >= 20) {
        Console.WriteLine("Kup 20 rohlíků.");
        rohliky-= 20;
    }
    else {
        Console.WriteLine("Nemají dost rohlíků.");
    }
}
else {
    if (rohliky >= 6) {
        Console.WriteLine("Kup 6 rohlíků.");
        rohliky-= 20;
    }
    else {
        Console.WriteLine("Nemají dost rohlíků.");
    }
}
```

Patrně jste si všimli, že jsme "opět" museli přizvat stádo if-ů, abychom mohli problém dostatečně ošetřit. Navíc se nám v kódu opakuje ta samá hláška, totiž že není dostatek rohlíků.

A zároveň - &&

Pokud máme více podmínek, které musí být splněny zároveň (současně), jako u výše probíraného příkladu, totiž že:

- mají-li vejce ... a
- je dost rohlíků

případně

- vejce nemají ... a
- je dost rohlíků

Pak můžeme tyto podmínky spojit do jediného výrazu pomocí tzv. logického součinu AND - `&&`.

```
if (vejce > 0 && rohliky >= 20) {
    Console.WriteLine("Kup 20 rohlíků.");
    rohliky-= 20;
}
```

```
else if (vejce == 0 && rohliky >= 6) {
    Console.WriteLine("Kup 6 rohlíků.");
    rohliky-= 6;
}
else {
    Console.WriteLine("Nemají dost rohlíků.");
}
```

Výsledek je, že pokud mají alespoň jedno vejce `vejce > 0`, **a zároveň** `&&` mají dostatek rohlíků `rohliku >= 20`, pak je podmínka splněna, a je tedy možné koupit 20 rohlíků.

Nebo jestliže `else if`, pokud nemají vejce `vejce == 0`, **a zároveň** mají dostatek rohlíků `rohliku >= 6`, pak je podmínka splněna, a je tedy možné koupit 6 rohlíků.

A nebo

Název školy	Vyšší odborná škola obalové techniky a Střední škola, Štětí, příspěvková organizace		
Adresa školky	Kostelní 134, 411 08 Štětí		
IČ	46773509		
Název operačního programu	OP Vzdělávání pro konkurenceschopnost		
Registrační číslo	CZ.1.07/1.5.00/34.1006		
Označení vzdělávacího materiálu	VY_32_INOVACE_23_PRG_455		
Název tématické oblasti (sady)	Programování		
Název materiálu	A nebo		
Anotace	Text představuje logický operátor součtu. Na příkladu demonstruje jeho nasazení, výsledkem čehož je přehlednější a čitelnější kód. Jsou ukázána dvě řešení úlohy – pomocí dosavadních znalostí, jejichž výsledkem je neefektivní a nepřehledný kód. Druhé řešení používá logický součet.		
Autor	Ivan Pomykacz	**Jazyk**	čeština
Očekávaný výstup	Přečte výraz poskládaný z logických součtů. Chápe významu logický součet – nebo.		
Klíčová slova	logický součet		
Druh výukového zdroje	Výklad	**Věková skupina žáků**	15+
Typ interakce	aktivita	**Ročník**	1.
Speciální vzdělávací potřeby	žádné		
Zhotoveno, (datum/období)	18.10.2013	**Celková velikost**	

Obsah

Shrnutí

Podmínky je možné spojovat (nikoli řetězit jako v případě `if-else-if`) pomocí logických operátorů AND a OR. To ve výsledku usnadňuje i zpřehledňuje zápis kódu.

V praxi je možné se setkat ještě dalším typem spojení podmínek. Totiž chceme-li, aby **alespoň jedna** z podmínek byla splněna (nikoli všechny současně).

Dilema

Ocitli jsme se na klinice v městské nemocnici. Stojíme v chodbě, kde na jejím konci jsou dvoje dveře. Jedny dveře vedou doprava a druhé doleva.

Kterými dveřmi se vydáte?

Implementace problému

Rozepišme program, který alespoň trošičku bude simulovat *old school* textovou hru.

```
Console.WriteLine("Nacházíš se v nemocnici. Stojíš v chodbě na jejímž konci
jsou dvoje dveře.");
Console.WriteLine("Chceš se vydat dveřmi vlevo nebo vpravo? ");

string volba = Console.ReadLine();
```

Necháme uživatele aby zapsal svou odpověď z klávesnice `Console.ReadLine()`. A nyní se rozhodneme, co se bude dít dál.

```
if (volba == "vlevo") {
    Console.WriteLine("Vstupuješ dveřmi nalevo do obrovské haly.");
}
else if (volba == "vpravo") {
    Console.WriteLine("Dveřmi vpravo sestupuješ po táhlém schodišti někam
dolů do sklepení.");
}
else {
    Console.WriteLine("Neznámá volba ..");
}
```

Vylepšení

Co když bychom chtěli uživateli ulehčit psaní? To aby nemusel pracně vypisovat `"vlevo"`, resp.

`"vpravo"`. Mohli bychom použít třeba anglická počáteční písmena `"l"` -> left a `"r"` -> right.

Chceme ale zachovat i původní chování - uživatel může uvést celou odpověď, ale zároveň může zadat jako odpověď jen jedno písmenko.

```
if (volba == "vlevo") {
    Console.WriteLine("Vstupuješ dveřmi nalevo do obrovské haly.");
}
else if (volba == "l") {
    Console.WriteLine("Vstupuješ dveřmi nalevo do obrovské haly.");
}
else if (volba == "vpravo") {
    Console.WriteLine("Dveřmi vpravo sestupuješ po táhlém schodišti někam
dolů do sklepení.");
}
else if (volba == "r") {
    Console.WriteLine("Dveřmi vpravo sestupuješ po táhlém schodišti někam
dolů do sklepení.");
}
else {
    Console.WriteLine("Neznámá volba ..");
}
```

Ano, nezbude nám nic jiného, než připsat další if-else do stáda. To by ale tak nevadilo jako spíš to, že musíme některé části kódu celé zopakovat. V programování však opakování kódu vede k záhubě programátora. Toto řešení je naprosto nepřijatelné - a to jak záhuba, tak opakování se.

A nebo - `||`

Pokud máme více podmínek, u kterých nám stačí alespoň u jedné aby výsledek byl `true`, jako u výše probíraného příkladu, totiž že:

- uživatel napsal "vlevo" ... a nebo
- uživatel napsal "l"

případně

- uživatel napsal "vpravo" ... a nebo
- uživatel napsal "r"

Pak můžeme tyto podmínky spojit do jediného výrazu pomocí tzv. logického součtu OR - `||`

```
if (volba == "vlevo" || volba == "l") {
    Console.WriteLine("Vstupuješ dveřmi nalevo do obrovské haly.");
}
else if (volba == "vpravo" || volba == "r") {
    Console.WriteLine("Dveřmi vpravo sestupuješ po táhlém schodišti někam
dolů do sklepení.");
}
else {
    Console.WriteLine("Neznámá volba ..");
}
```

Výsledek je, že pokud řetězec `volba == "vpravo"` **nebo** `||` řetězec `volba == "r"`, pak je podmínka splněna.

Nebo jestliže `else if`, pokud řetězec `volba == "vlevo"` **nebo** `||` řetězec `volba == "l"`, pak je podmínka splněna.

Logické operátory

Název školy	Vyšší odborná škola obalové techniky a Střední škola, Štětí, příspěvková organizace		
Adresa školky	Kostelní 134, 411 08 Štětí		
IČ	46773509		
Název operačního programu	OP Vzdělávání pro konkurenceschopnost		
Registrační číslo	CZ.1.07/1.5.00/34.1006		
Označení vzdělávacího materiálu	VY_32_INOVACE_23_PRG_456		
Název tématické oblasti (sady)	Programování		
Název materiálu	Logické operátory		
Anotace	Text shrnuje poznatky o logických operátorech. Přidává operátor logické negace. Dále předkládá cvičení na procvičení nasazení logických operátorů.		
Autor	Ivan Pomykacz	**Jazyk**	čeština
Očekávaný výstup	Rozhodne o použití logického operátoru. Používá logické operátory v kódu.		
Klíčová slova	logický součet, logický součin, logická negace, výraz, operátor		
Druh výukového zdroje	Pracovní list	**Věková skupina žáků**	15+
Typ interakce	aktivita	**Ročník**	1.
Speciální vzdělávací potřeby	žádné		
Zhotoveno, (datum/období)	21.10.2013	**Celková velikost**	

And, Or, Not

Obsah

Shrnutí

Logické operátory umožňují tvořit komplexnější podmínky, které pak dokážou vlastně jednodušeji popsat daný problém. O čemž jste se mohli přesvědčit v A zároveň a A nebo.

Pravdivostní tabulky

V souvislosti s logickými operátory se uvádějí tzv. pravdivostní tabulky, ve kterých vystupují logický součin, součet a logická negace.

V tabulkách níže se objevuje `x`, za které lze dosadit libovolný výraz, jehož výsledkem je buď `true` nebo `false`. V tabulce jsou pak uvedeny všechny kombinace, které by mohly nastat včetně výsledku `y`.

Logický součin `&&`

Provedeme-li se všemi `x` logický součin, pak výsledkem je `y`.

AND - `&&`		
x_1	x_2	y
`false`	`false`	`false`
`false`	`true`	`false`
`true`	`false`	`false`
`true`	`true`	`true`

U logického součinu tedy platí, že **právě všechna** `x` musí být `true`, aby i výsledek byl `true`.

V příkladu jsou pouze $x_1..x_2$, nicméně tento počet není omezen.

Logický součet `||`

Provedeme-li se všemi `x` logický součet, pak výsledkem je `y`.

OR - `||`		
x_1	x_2	y

`false`	`false`	`false`
`false`	`true`	`true`
`true`	`false`	`true`
`true`	`true`	`true`

U logického součtu platí, že stačí aby **alespoň jedno** `x` bylo `true`, a pak i výsledek je `true`.

Logická negace `!`

Logická negace tzv. neguje (převrátí) pravdivostní hodnotu na její opak. Tedy z `true` udělá `false`, a obráceně z `false` bude `true`.

NOT - `!`	
x	y
`false`	`true`
`true`	`false`

Oproti logickému součtu nebo součinu má negace právě jeden argument (max jedno `x`). Pokud ovšem použijeme závorky, můžeme negovat výsledek takové závorky, např. `!(true || false)`. Výsledkem bude vždy `false`, protože závorka vždy vrátí `true`.

Cvičení

Napište program, který bude simulovat "logickou kalkulačku".

Nechť uživatel zadá po sobě dvě čísla 0 nebo 1.

Program pak vyhodnotí logický součin, součet a negaci těchto čísel.

Řešení

```
Console.WriteLine ("Program: and-or-not v1.0");

string a;
string b;

Console.Write("Zadej log. 1 nebo 0: ");
a = Console.ReadLine();
Console.Write("Zadej ještě jednu log. 1 nebo 0: ");
b = Console.ReadLine();

Console.WriteLine("Logický součin - &&");
if (a == "1" && b == "1") {
    Console.WriteLine("{0} && {1} = 1", a, b);
}
else {
    Console.WriteLine("{0} && {1} = 0", a, b);
}
```

```
Console.WriteLine("Logický součet - ||");
if (a == "0" && b == "0") {
    Console.WriteLine("{0} || {1} = 0", a, b);
}
else {
    Console.WriteLine("{0} || {1} = 1", a, b);
}

Console.WriteLine("Logická negace - &&");
if (a == "1") {
    Console.WriteLine("{0} ! 0", a);
}
else {
    Console.WriteLine("{0} ! 1", a);
}
if (b == "1") {
    Console.WriteLine("{0} ! 0", b);
}
else {
    Console.WriteLine("{0} ! 1", b);
}
```

Cvičení IX

Obsah

Zadání

Vytvořte aplikaci, která bude vyžadovat přihlášení uživatele. *Náš program bude značně omezený, ale jen do té doby, než se seznámíme s dalšími metodami a technikami programování.*

Postup

- Program vyzve uživatele k zadání uživatelského jména a hesla. Použijte metodu `Console.ReadLine()` pro získání uvedených dat.
- Proveďte ověření totožnosti uživatele. V naší aplikaci bude veden pouze jeden uživatelský účet. Pro úspěšné přihlášení je potřeba aby přihlašovací jméno a heslo byly stejné jako ty, co jsou uložené v programu. Můžete použít logický součin k vyřešení autentizace.
- Vypište informaci o úspěchu či neúspěchu přihlášení.Pozor! Z bezpečnostního hlediska byste neměli uživatele informovat o tom, zda-li zadal chybné jméno nebo heslo. Útočník zpravidla nezná ani jméno ani heslo. A pokud mu napovíte tím, že uhádl heslo chybovou hláškou "Špatně zadané

heslo", je na půl cesty k tomu, aby získal i heslo.

Verze 1.1

Schovejme heslo před zraky náhodných čumilů.

Pomocí vlastností `Console.ForegroundColor` a `Console.ForegroundColor` můžeme dočasně dosadit barvu pozadí do barvy popření a tím vlastně zneviditelnit psaný text.

Deklarujme novou proměnnou `barva` typu `ConsoleColor`, do které uložíme aktuální barvu popření (písma).

```
ConsoleColor barva;
barva = Console.ForegroundColor;
```

Následně můžeme změnit barvu popření na barvu pozadí.

```
Console.ForegroundColor = Console.BackgroundColor;
```

Po zadání hesla to ovšem musíme vrátit zpět :)

```
Console.ForegroundColor = barva;
```

Řešení

```
Console.WriteLine("Program: Login v1.1\n");

string secretName = "root";
string secretPass = "toor";

string loginName;
string loginPass;

Console.WriteLine("Přihlášení");
Console.Write("login: ");
loginName = Console.ReadLine();

// příprava na změnu barvy
ConsoleColor barva;
barva = Console.ForegroundColor;

Console.Write("heslo: ");
// záměna barvy
Console.ForegroundColor = Console.BackgroundColor;
// načtení hesla
loginPass = Console.ReadLine();
// barva zpět
Console.ForegroundColor = barva;
```

```
// autentizace
if (secretName == loginName && secretPass == loginPass) {
    Console.WriteLine("\nPřihlášení proběhlo úspěšně");
}
else {
    Console.WriteLine("\nChybně zadané jméno nebo heslo!");
}
```

Mini příklady III

Obsah

Úloha 1 - Hádankář

Program nabídne uživateli hádanku.
Pokud uživatel odpoví správně, položí program další hádanku. Celkem budou v programu 3 hádanky.

Pokud uživatel zodpoví všechny hádanky správně, program vypíše informaci o úspěšném řešení.
Pokud uživatel odpoví špatně na alespoň jednu hádanku, tak program okamžitě skončí.

Program by měl tolerovat, zda-li bude odpověď napsána různě velkými písmeny. Případně by mu nemělo vadit, zda-li před nebo za slovem bude mezera.

demo

Úloha 2 - Mysli si číslo

Program vyzve uživatele, aby si myslel číslo od 1-5. Následně se program bude ptát uživatele na otázky, pomocí kterých zjistí, jaké si uživatel myslel číslo. Otázky budou typu: Je tvé číslo větší nebo menší jak ...

demo

Znaky a řetězce

Název školy	Vyšší odborná škola obalové techniky a Střední škola, Štětí, příspěvková organizace		
Adresa školky	Kostelní 134, 411 08 Štětí		
IČ	46773509		
Název operačního programu	OP Vzdělávání pro konkurenceschopnost		
Registrační číslo	CZ.1.07/1.5.00/34.1006		
Označení vzdělávacího materiálu	VY_32_INOVACE_23_PRG_457		
Název tématické oblasti (sady)	Programování		
Název materiálu	Znaky a řetězce		
Anotace	Text seznamuje zevrubněji s textovými datovými typy string a char. Vysvětluje použití typu char. Představuje metody, které lze provádět nad řetězci (např. převést písmena na malá). Ukazuje možnosti zřetězení metod.		
Autor	Ivan Pomykacz	**Jazyk**	čeština
Očekávaný výstup	Používá typ char. Používá metody objektu string. Rozumí principu zřetězení volání metod.		
Klíčová slova	string, char, metody objektu, zřetězení metod		
Druh výukového zdroje	Výklad	**Věková skupina žáků**	15+
Typ interakce	aktivita	**Ročník**	1.
Speciální vzdělávací potřeby	žádné		
Zhotoveno, (datum/období)	08.11.2013	**Celková velikost**	

Obsah

Shrnutí

Již víme, že existuje typ `string`, který může obsahovat text. Textem rozumíme písmena abecedy, číslice a jiné znaky, jako např. interpunkce (tečka, čárka, otazník, vykřičník), ale i např. symboly z matematiky (závorky, aritmetické operátory), a tak podobně.

Řetězec může být prázdný, např.

```
string text = "";
```

Může obsahovat jeden znak

```
string text = "a";
```

ale i více znaků (proto řetězec)

```
string text = "Lorem ipsum dolor sit amet.";
```

Přičemž bereme v potaz, že **na velikosti písmen záleží**. Pro kompilátor jsou malé `p` a velké `P` natolik odlišné znaky, jako jako byste míchali hrušky s banány.

Char

Datový typ `char` je specifický ve dvou ohledech (oproti typu `string`). Může obsahovat pouze jeden znak. Hodnota se zapisuje do jednoduchých uvozovek, tedy:

```
char znak = 'a';
```

K čemu `char`? Předně typ `string` je poskládán právě z objektů typu `char`. Tudíž, proměnná `string slovo = "lorem";` se vlastně skládá (vnitřně) z pěti *charů*.

Tyto *chary* můžeme ze *stringu* vytáhnout třeba takto:

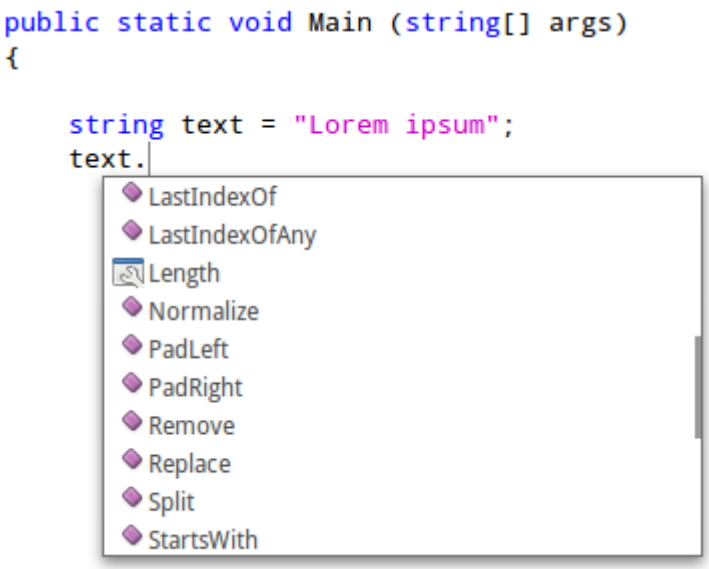
```
string slovo = "lorem";
char prvni;
char posledni;

prvni = slovo[0];
posledni = slovo[4];

Console.WriteLine("První znak: {0}, poslední znak {1}.", prvni, posledni);
```

Tato znalost nám pak pomůže v pokročilejší práci s typem `string`. Se samotným *charem* budeme pracovat málokdy, ale přeci, někdy na něj narazíme.

String

Nyní víme, jak se věci mají. String nám vlastně zjednodušuje práci s řetězci, které jsou poskládány z jednotlivých *charů*.

Jaké operace můžeme s řetězci provádět? Pokaždé když napíšete proměnnou typu `string` a připojíte tečku, IDE vám nabídne metody, které se vážou k danému objektu (zde `string`).



Kompletní přehled metod je samozřejmě v oficiální referenci jazyka.

Navíc, ne všechno jsou metody. Na obrázku je schválně zachycena vlastnost `Length`. Vlastnosti na rozdíl od metod neprovádí žádnou akci. Jsou to vlastně proměnné, které mají přímou vazbu k objektu.

Metody objektu `string`

Ukažme si to na příkladu. Do proměnné `answer` budeme ukládat odpověď, kterou uživatel napíše z klávesnice. Následně otestujeme, zda-li zadal "ano", resp. "ne".

```
string answer;
Console.Write("Je již rozhodnuto? ");

answer = Console.ReadLine();

if (answer == "ano") {
    Console.WriteLine("Výborně, můžeme pokračovat");
}
else if (answer == "ne") {
    Console.WriteLine("A kde to vázne?");
}
else {
```

```
    Console.WriteLine("Chyba! Odpověď nebyla rozpoznána.");
}
```

Určitě vás napadlo, co když uživatel zadá např. "**A**no" s velkým počátečním písmenem. To je ale přece úplně jiná hodnota a program tak vstup vyhodnotí jako chybu.

Ano, mohli bychom uživateli napovědět:

```
Console.Write("Je již rozhodnuto? Napiš pouze \"ano\" nebo \"ne\" a je třeba
psát pouze malými písmeny. ");
```

Ale mnohem lepší, tedy alespoň pro uživatele, bude, když se pokusíme jeho vstup přiměřeně opravit.

Vše na malá

Objekt `string` obsahuje mj. metodu `ToLower()`. Ta převádí text na malá písmena (tam kde to má smysl, čísla samozřejmě zůstávají pořád stejně veliká).

Funguje to tak, že po zavolání metody nad objektem string se vrátí upravená hodnota tohoto objektu. Tedy:

```
string text = "Lorem Ipsum";
string upraveno;
upraveno = text.ToLower();
```

Do proměnné `upraveno` se uloží modifikovaný text `lorem ipsum`. Původní obsah proměnné `text` zůstává nezměněn. Všimněte si, že návratová hodnota metody `ToLower()` je opět `string`. Co jiného by to také mohlo být, že?

Není vždy potřeba ukládat změny do proměnné, jelikož pokaždé se pracuje se stringem, stačí v podmínce zapsat:

```
if (answer.ToLower() == "ano") {
    Console.WriteLine("Výborně, můžeme pokračovat");
}
else if (answer.ToLower() == "ne") {
    Console.WriteLine("A kde to vázne?");
}
else {
    Console.WriteLine("Chyba! Odpověď nebyla rozpoznána.");
}
```

Zřetězení metod

Někdy potřebujeme provést na objektem více akcí. Např. chceme text převést na malá písmena a odstranit ze začátku a z konce řetězce bílé znaky (např. mezery, konce řádků, tabulátory).

Metodu `ToLower()` už známe. Ta, co nám umožní pročistit řetězec se nazývá `Trim()`.

S metodou `Trim()` to bude vypadat obdobně.

```
string text = "Lorem Ipsum";
```

```
string upraveno;
upraveno = text.ToLower();
upraveno = upraveno.Trim();
```

Celé to lze ovšem zapsat i kratším způsobem, tzv. zřetězením:

```
string text = "Lorem Ipsum";
string upraveno;
upraveno = text.ToLower().Trim();
```

Kvůli přehlednosti a i efektivnosti kódu by pak naše podmínka mohla vypadat:

```
string answer;
Console.Write("Je již rozhodnuto? ");

answer = Console.ReadLine().ToLower().Trim();

if (answer == "ano") {
    Console.WriteLine("Výborně, můžeme pokračovat");
}
else if (answer == "ne") {
    Console.WriteLine("A kde to vázne?");
}
else {
    Console.WriteLine("Chyba! Odpověď nebyla rozpoznána.");
}
```

Změna je pouze v úpravě řádku u volání metody `ReadLine()`.

Práce s textem

Název školy	Vyšší odborná škola obalové techniky a Střední škola, Štětí, příspěvková organizace		
Adresa školky	Kostelní 134, 411 08 Štětí		
IČ	46773509		
Název operačního programu	OP Vzdělávání pro konkurenceschopnost		
Registrační číslo	CZ.1.07/1.5.00/34.1006		
Označení vzdělávacího materiálu	VY_32_INOVACE_23_PRG_458		
Název tématické oblasti (sady)	Programování		
Název materiálu	Práce s textem		
Anotace	Komplexnější cvičení na program shrnující všechny dosavadní znalosti. Prezentuje klasický vývojový diagram, na němž demonstruje rozfázování programu na dílčí části. Dále popisuje způsob implementace programu.		
Autor	Ivan Pomykacz	**Jazyk**	čeština
Očekávaný výstup	Provádí základní analýzu programu. Čte vývojový diagram. Implementuje postupně program dle vývojového diagramu.		
Klíčová slova	implementace, vývojový diagram, analýza, metody string, logické operátory		
Druh výukového zdroje	Pracovní list	**Věková skupina žáků**	15+
Typ interakce	aktivita	**Ročník**	1.
Speciální vzdělávací potřeby	žádné		
Zhotoveno, (datum/období)	13.11.2013	**Celková velikost**	

Obsah

Pig Latin translator

Pig Latin je jazyková hra pro anglický jazyk. Používá se pro pobavení i pro zachování "soukromí" mezi mluvčími. Princip hry spočívá v úpravě slov přidáním atypických přípon nebo přesunutím první souhlásky na konec původního slova.

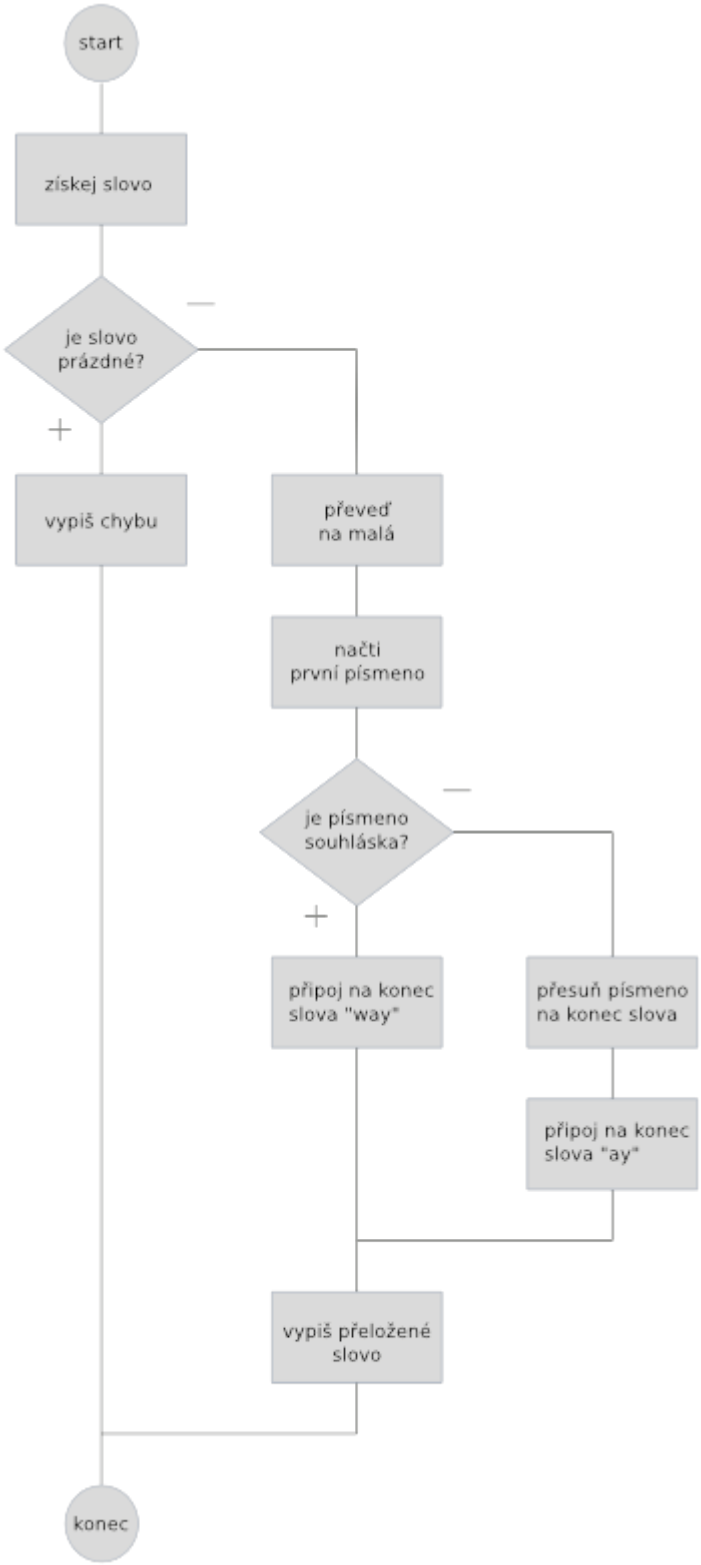
Naším úkolem bude vytvořit překladač z angličtiny do Pig Latin.

Pravidla

- Začíná-li slovo samohláskou, připojíme na konec slova `way`.
- Začíná-li slovo souhláskou, pak první písmeno přesuneme na konec slova a přidáme `ay`.

Implementace

U takto rozsáhlejšího programu je potřeba si jednotlivé části rozfázovat a realizovat postupně.



Vstup

Začněme kódem, který umožní uživateli napsat slovo, které pak budeme překládat.

```
string slovo;

slovo = Console.ReadLine();
```

Následně provedeme test, zda-li zadané slovo není prázdný řetězec.

```
if (slovo != "") {
    Console.WriteLine(slovo);
    //...
}
else {
    Console.WriteLine("Chyba! Prázdný řetězec.");
}
```

Úpravy

Nyní potřebujeme znaky ve `slovo` převést na malá písmena a oříznout o případné bílé znaky. Vytvořme proto novou proměnnou, do které změny uložíme.

```
upravene_slovo = slovo.Trim().ToLower();
```

Počáteční písmeno

Pak budeme potřebovat z upraveného slova vytáhnout počáteční písmeno, abychom mohli zjistit, zda-li je to samohláska.

Bude nezbytné deklarovat novou proměnnou `char prvni;`.

```
prvni = upravene_slovo[0];
```

Detekce samohlásky

Pro detekci samohlásky použijeme podmínku, která zjistí, zda-li první písmeno slova je jedno ze samohlásek (a,e,i,o,u).

A protože nám stačí, aby ze všech možností (a,e,i,o,u) vyhověla alespoň jedna, použijeme logický součet `||`.

```
if (prvni == 'a' || prvni == 'e' || prvni == 'i' || prvni == 'o' || prvni ==
'u') {
    // jde o samohlásku
}
else {
    // je to souhláska
}
```

Pozor! Porovnáváme `char`, proto musíme použít jednoduché uvozovky.

Překlad

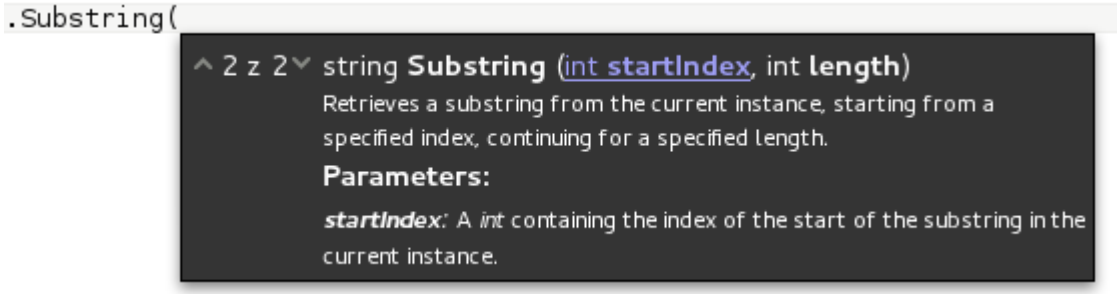
Pokud počáteční písmeno není samohláskou, pak stačí na konec slova připojit text "way".

```
nove_slovo = upravene_slovo + "way";
```

Ano, v kódu je použitá nová proměnná `nove_slovo`, která ještě nebyla deklarována.

V opačném případě, je-li první písmeno slova souhláska, musíme ji přesunout na konec slova.

Je možné k tomu využít metodu `Substring()` objektu `string`. `Substring()` totiž dokáže z řetězce vykopírovat určitou jeho podmnožinu. Podíváme-li se na parametry metody, pak zjistíme, že má jeden povinný parametr a druhý volitelný.



Povinný parametr je počáteční index, tedy číslo, od kolikátého znaku se má začít kopírovat nový řetězec (počítáno od nuly).

Volitelný parametr je délka, tj. kolik znaků se má kopírovat. Nevyplníme-li tento parametr, použije se zbývající délka původního řetězce.

```
nove_slovo = upravene_slovo.Substring(1) + prvni + "ay";
```

Metody

Název školy	Vyšší odborná škola obalové techniky a Střední škola, Štětí, příspěvková organizace		
Adresa školky	Kostelní 134, 411 08 Štětí		
IČ	46773509		
Název operačního programu	OP Vzdělávání pro konkurenceschopnost		
Registrační číslo	CZ.1.07/1.5.00/34.1006		
Označení vzdělávacího materiálu	VY_32_INOVACE_23_PRG_459		
Název tématické oblasti (sady)	Programování		
Název materiálu	Metody		
Anotace	Text seznamuje s pojmem metoda. Ukazuje na příkladu použití metod, jejich argumentů a návratových hodnot.		
Autor	Ivan Pomykacz	**Jazyk**	čeština
Očekávaný výstup	Chápe významu použití metod. Rozumí, co je vstupní argument metody. Chápe, co je návratová hodnota metody.		
Klíčová slova	metody, vstupní a výstupní argumenty, návratová hodnota		
Druh výukového zdroje	Výklad	**Věková skupina žáků**	15+
Typ interakce	aktivita	**Ročník**	1.
Speciální vzdělávací potřeby	žádné		
Zhotoveno, (datum/období)	20.11.2013	**Celková velikost**	

Obsah

Definice

Metoda v objektově orientovaném programování (nebo funkce v klasickém, procedurálním programování) je tzv. znovu-použitelný blok kódu, který vykonává specifickou úlohu v programu. Proč psát oddělené bloky kódu, když to funguje i bez nich?

- Pokud se v kódu objeví chyba, je mnohem rychlejší její nalezení a opravení, pokud je program (zdrojový kód) dobře organizován. To že mají různé funkce různou úlohu pomáhá v organizaci programu.
- To že přidělíme specifické úlohy rozdílným funkcím má za následek méně nadbytečného kódu. Rovněž vzniká tzv. znovu použitelný kód. Funkce můžeme opakovaně spouštět (volat) a nebo je dokonce využít i v jiném programu.
- V objektově orientovaném programování (a C# je objektově orientovaný jazyk) se s metodami dá provádět mnoho zajímavých věcí.
- D.R.Y. - znamená **D**ont **R**epeat **Y**ourself (Neopakuj se). Při používání funkcí (metod) se tomuto dá lehce předejít.
- Rekurze je velmi mocný nástroj. Umožňuje spustit opakovaně kód, který spouští sám sebe. Existuje několik programovacích problémů, které nelze bez rekurze vyřešit. Díky funkcím, můžeme provádět rekurzi.

Příklad

Mějme metody `WriteLine()` a `ReadLine()`. Obě jsou svázány s objektem `Console`, proto je používáme tak jak doposavad.

```
Console.WriteLine();
Console.ReadLine();
```

Používali jsme i metodu `Parse()` z objektu `int`.

Tuto metodu jsme kombinovali společně s metodou `Console.ReadLine()`, díky čemuž jsme z řetězce `string` udělali číslo `int`.

```
int.Parse(Console.ReadLine());
```

Platí, že nejprve se výraz vyhodnocuje od těch nejvnořenějších metod (zde `Console.ReadLine()`) směrem vzhůru (zde `int.Parse()`).

Tzn. jako první se provede `Console.ReadLine()`, která čeká na stisk klávesy Enter a ihned poté předá získaná data z klávesnice metodě `int.Parse()`.

Vstupní argumenty (parametry)

Metoda může a nemusí mít vstupní argumenty.

Jelikož metoda je oddělený (samostatný) blok kódu, byl definován způsob, jak těmto metodám předat nějaká vstupní data, se kterými by mohly pracovat.

Např. metoda `Console.WriteLine()`. Metoda vypíše do konzole předaný argument a vloží řádkový zlom (jako byste napsali Enter). Přičemž řádkový zlom vloží vždy, i když nepředáte metodě žádný argument.

Argumenty jsou konkrétního datového typu. Není tedy možné libovolné metodě předat proměnnou typu `int` nebo `string`, pokud to nepodporuje.

`Console.WriteLine()` je tak trochu výjimka. Je jí totiž celkem jedno, co jí předáte.

```
string jmeno = "Lorem Ipsum";
int cislo = 12345;
Console.WriteLine(jmeno);
Console.WriteLine(cislo);
```

Jak je vidět výše, předali jsme metodě nejprve `string` a potom `int` a kompilátor žádnou chybu nezahlásil.

O tom, jaké vstupní argumenty metoda podporuje, se lze dočíst v referenční příručce jazyka (knihovny).

Návratové hodnoty

Každá metoda má definovaný tzv. návratový typ. Jde vlastně o datový typ (např. `int`, `string`). Proč mají metody návratové hodnoty?

Účelem metody je něco dělat, a pokud to má smysl, tak vrátit nějaký výsledek.

Pokud se bavíme o metodě `Console.ReadLine()`, tak tato metoda čeká na vstup z klávesnice. Je-li stisknuta klávesa Enter, čtení vstupu se ukončí, a to, co bylo napsáno, se předá na výstupu metody.

```
string jmeno;
jmeno = Console.ReadLine();
```

Metoda `Console.ReadLine()` vrací vždy datový typ `string`. Proto, když chceme uložit data z klávesnice do proměnné, musí být tato proměnná datového typu `string`.

O tom, jaké datové typy metoda vrací, se lze dočíst opět v referenční příručce jazyka (knihovny).

Prázdnota

Některé metody nevrací nic, resp. vrací "nic". Např. metoda `Console.WriteLine()` nic nevrací. Přesto musí něco vracet. Programátoři to vymysleli šibalsky a přidali návrátový typ `void` (prázdnota).

Pokud tedy existuje metoda, která nemá nic vracet, pak musí vrátit typ `void`.

Proměnnou typu `void` se vám však deklarovat nepodaří.

Metody a objekty

C# je objektově orientovaný jazyk. Proto nemůže metoda existovat jen tak, bez svého objektu.

Patrně jste si všimli, že některé metody se používají trochu jinak, než bylo popsáno v předchozích kapitolách.

Např. metoda `string.ToLower()`. Tato metoda nemá žádný vstupní argument, a přesto vrací hodnotu typu `string`.

```
string jmeno = "Lorem Ipsum";
string mala = jmeno.ToLower();
```

Je tomu tak proto, že metoda `ToLower()` je svázána se svým objektem (typu `string`). Pokud zavoláme tuto metodu, pak operaci převodu na malá písmena provede nad objektem, ze kterého byla volána (zde `jmeno`). A nepotřebuje tak vstupní argument.

Deklarace metody

Název školy	Vyšší odborná škola obalové techniky a Střední škola, Štětí, příspěvková organizace		
Adresa školky	Kostelní 134, 411 08 Štětí		
IČ	46773509		
Název operačního programu	OP Vzdělávání pro konkurenceschopnost		
Registrační číslo	CZ.1.07/1.5.00/34.1006		
Označení vzdělávacího materiálu	VY_32_INOVACE_23_PRG_460		
Název tématické oblasti (sady)	Programování		
Název materiálu	Deklarace metody		
Anotace	Prezentace deklarace metody bez/s vstupními argumenty a bez/s návratovou hodnotou. Volání metody.		
Autor	Ivan Pomykacz	**Jazyk**	čeština
Očekávaný výstup	Popíše metodu dle její deklarace. Deklaruje metodu. Volá metodu v kódu.		
Klíčová slova	deklarace metody		
Druh výukového zdroje	Výklad	**Věková skupina žáků**	15+
Typ interakce	aktivita	**Ročník**	1.
Speciální vzdělávací potřeby	žádné		
Zhotoveno, (datum/období)	20.11.2013	**Celková velikost**	

Obsah

Příklad

V kódu níže je deklarována jen jedna metoda. Jmenuje se `Main()`.

```
using System;

namespace Deklarace
{
    class MainClass
    {
        public static void Main (string[] args)
        {
            Console.WriteLine ("Hello World!");
        }
    }
}
```

Můžeme o ní říci, že:

- Je veřejná (public).
- Je statická (static).
- Vrací `void`, tedy nevrací nic.
- Má jeden argument typu `string[]` a názvem `args`.
- Je součástí třídy (class) `MainClass`, a ta je součástí jmenného prostoru (namespace) `Deklarace`. Vše (namespace, class a metoda) má své složené závorky `{}`.

Vzhledem k tomu, že mi se ještě nebudeme zabývat plně objektovým programováním, budeme některá vyhrazená slova (static a public) tiše přecházet.

Navíc všechny metody budeme psát do jediné třídy, zde `MainClass`.

Nová metoda

Proveďme deklaraci nové metody s názvem `Info`. Metoda zobrazí název našeho programu a základní informace o něm. Nebude žádný vstupní argument a nebude nic vracet.

```
public static void Info()
{

}
```

Klíčová slova *public* a *static* budeme "tiše" psát před každou nově deklarovanou metodou, dokud si neřekneme, co to znamená.

Dávejte dobrý pozor, do kterého místa v kódu metodu vpisujete.

```
using System;

namespace Deklarace
{
    class MainClass
    {
        public static void Main (string[] args)
        {
            Console.WriteLine ("Hello World!");
        }

        public static void Info()
        {

        }
    }
}
```

Implementace

Implementace spočívá v tom, že napíšeme kód, který provede požadovanou akci.

V našem případě jde o to vypsat informace o programu.

```
public static void Info()
{
    Console.WriteLine("Program: Deklarace v1.0");
    Console.WriteLine("Zdrojový kód programu demonstruje deklaraci nové
metody a její volání.");
    Console.WriteLine("-
-
----------------------------------------------------------------------------
-");
}
```

Volání metody

Volání metody spočívá v tom, že někam do kódu napíšeme název metody `Info()`, čímž při vyhodnocování kódu dojde k jejímu zavolání = spuštění.

V našem případě se tedy vrátíme do metody `Main()` a dopíšeme volání metody `Info()`.

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

```
    Info();
    Console.WriteLine ("Hello World!");
}
```

Opakované volání

Jak jsme již zmiňovali, metodu lze volat opakovaně. U metody `Info()` to nemá zřejmě moc význam, ale ...

```
public static void Main (string[] args)
{
    Info();
    Info();
    Console.WriteLine ("Hello World!");
    Info();
    Info();
}
```

Metody - cvičení

Název školy	Vyšší odborná škola obalové techniky a Střední škola, Štětí, příspěvková organizace		
Adresa školky	Kostelní 134, 411 08 Štětí		
IČ	46773509		
Název operačního programu	OP Vzdělávání pro konkurenceschopnost		
Registrační číslo	CZ.1.07/1.5.00/34.1006		
Označení vzdělávacího materiálu			
Název tématické oblasti (sady)	Programování		
Název materiálu	Metody - cvičení		
Anotace			
Autor	Ivan Pomykacz	**Jazyk**	čeština
Očekávaný výstup			
Klíčová slova			
Druh výukového zdroje	Pracovní list	**Věková skupina žáků**	15+
Typ interakce	aktivita	**Ročník**	1.
Speciální vzdělávací potřeby	žádné		
Zhotoveno, (datum/období)	2013	**Celková velikost**	

Cvičení X - Deklarace

Obsah

Název metody

Napište metodu, která do konzole vypíše text "Nutno implementovat".

Název metody bude `NazevMetody()`.
Metoda nemá vstupní argumenty.
Metoda vrací `void`.

Zavolejte metodu `NazevMetody()` 3x z hlavní metody `Main()`.

Řešení

```
public static void Main(string[] args)
{
    NazevMetody();
    NazevMetody();
    NazevMetody();

    Console.ReadKey(false);
}

public static void NazevMetody() {
    Console.WriteLine("Nutno implementovat!");
}
```

Obsah čtverce

Napište metodu, která spočítá obsah čtverce dle zadané délky strany.

Název metody bude `ObsahCtverce()`.
Metoda má jeden vstupní parametr `double a`.
Metoda vrací `double`.

Zavolejte metodu `ObsahCtverce()` z hlavní metody `Main()`. Jelikož metoda nic nevypisuje, zkombinujte ji s metodou `Console.WriteLine()`.

Řešení

```
public static void Main(string[] args)
{
    Console.WriteLine(ObsahCtverce(3));

    Console.ReadKey(false);
}

public static double ObsahCtverce(double a) {
    double obsah;
    obsah = a*a;

    return obsah;
}
```

Zkrácená verze

```
public static void Main(string[] args)
{
    Console.WriteLine(ObsahCtverce(3));

    Console.ReadKey(false);
}

public static double ObsahCtverce(double a) {
    return a*a;
}
```

Zadej číslo

Napište metodu, která vyzve uživatele k zadání čísla a toto číslo vrátí jako svou návratovou hodnotu.

Název metody bude `ZadejCislo()`.
Metoda nemá vstupní parametr.
Metoda vrací `double`.

Zavolejte metodu `ZadejCislo()` z hlavní metody `Main()`. Jelikož metoda nic nevypisuje, zkombinujte ji s metodou `Console.WriteLine()`.

Řešení

```
public static void Main(string[] args)
{
    Console.WriteLine(ZadejCislo());

    Console.ReadKey(false);
}

public static double ZadejCislo() {
    Console.Write("Zadej číslo: ");
```

```
    double cislo = double.Parse(Console.ReadLine());

    return cislo;
}
```

Vylepšení

Nechť metoda `ZadejCislo` má vstupní parametr, kterým bude text, jenž bude obsahovat hlášku, která uživatele vyzve k zadání čísla. Smyslem je, aby šla tato hláška nastavit dle konkrétní situace. Přeci jen text: "Zadej číslo", je dost obecný.

```
public static void Main(string[] args)
{
    Console.WriteLine(ZadejCislo("Zadej libovolné číslo: "));

    Console.ReadKey(false);
}

public static double ZadejCislo(string vyzva) {
    Console.Write(vyzva);
    double cislo = double.Parse(Console.ReadLine());

    return cislo;
}
```

Zkombinování

Nyní můžeme použít metody `ZadejCislo()` a `ObsahCtverce()` a jejich kombinací rozšířit funkčnost programu.

Metoda `ZadejCislo()` totiž vrací hodnotu typu `double`, a tu lze použít jako vstupní parametr pro metodu `ObsahCtverce()`.
Výsledek pak můžeme poslat do metody `Console.WriteLine()`.

```
public static void Main(string[] args)
{
    Console.WriteLine("Obsah čtverce: " + ObsahCtverce(ZadejCislo("Zadej
stranu a: ")) + " j^2");

    Console.ReadKey(false);
}

public static double ZadejCislo(string vyzva) {
    Console.Write(vyzva);
    double cislo = double.Parse(Console.ReadLine());

    return cislo;
}

public static double ObsahCtverce(double a) {
    double obsah;
```

```
    obsah = a*a;

    return obsah;
}
```

Cvičení XI - Objemy

Obsah

Zadání

Napište program pro výpočet objemu těles: krychle, kvádru a koule.

Program vyzve uživatele k výběru tělesa, pro který se bude následně objem počítat. Pro každé těleso bude existovat metoda, jež bude počítat jeho objem. Názvy metod jsou libovolné, např:

- `ObjemKvadr()`
- `ObjemKrychle()`
- `ObjemKoule()`

Metody budou mít různý počet vstupních argumentů (zejména pro výpočet objemu kvádru). Více vstupních argumentů se odděluje čárkou. Každá metoda bude vracet datový typ `double`.

V metodě `Main()` pak napište kód, který vyzve uživatele k volbě jednoho ze tří těles. Následně se provede výpočet a výpis výsledku.

Můžete využít metodu `ZadejCislo()` z předchozího cvičení.

Stáhněte si a spusťte demoverzi pro lepší představu.

Řešení

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

    Console.WriteLine("Program: Objemy v1.0\n");

    Console.WriteLine("Zvol objekt pro výpočet objemu:");
    Console.WriteLine("a) Krychle, b) Kvádr, c) Koule");
    Console.Write (":");
    string volba = Console.ReadLine();
    if (volba == "a") {
        double strana;
        strana = ZadejCislo("Zadej stranu a: ");
        double objem;
        objem = Krychle(strana);
```

```
        Console.WriteLine("Objem krychle = " + objem + "j^3");
    }
    else if (volba == "b") {
        double stranaA = ZadejCislo("Zadej stranu a: ");
        double stranaB = ZadejCislo("Zadej stranu b: ");
        double stranaC = ZadejCislo("Zadej stranu c: ");

        double objem = Kvadr(stranaA, stranaB, stranaC);
        Console.WriteLine("Objem kvádru = " + objem + "j^3");
    }
    else if (volba == "c") {
        double objem = Koule(ZadejCislo("Zadej poloměr r: "));
        Console.WriteLine("Objem koule = " + objem + "j^3");
    }
    else {
        Console.WriteLine("Neznámá volba");
    }


    Console.Write("Press any key to continue . . . ");
    Console.ReadKey(true);

}

public static double ObjemKrychle(double a) {
    double objem;
    objem = a*a*a;

    return objem;
}

public static double ObjemKvadr(double a, double b, double c) {
    double objem;
    objem = a*b*c;

    return objem;
}

public static double ObjemKoule(double r) {
    double objem;
    objem = 4*Math.PI*r*r*r/3;

    return objem;
}

public static double ZadejCislo(string vyzva) {
    double cislo;
    Console.Write(vyzva);
    cislo = int.Parse(Console.ReadLine());

    return cislo;
```

```
}
```

Cvičení XII - Login 2

Obsah

Zadání

Napište program, který bude simulovat přihlašování uživatele. Půjde o velice podobný kód, který jsme již psali, ovšem tentokrát jej rozčleníme do metod.

- Napište metodu `Prihlaseni()`, která bude implementovat přihlašování uživatele.
  Jako vstupní parametry metody budou přihlašovací jméno a heslo. Metoda sama tedy tyto údaje získávat nebude. Metoda pouze staticky porovnává vstupní jméno a heslo se `secretName`, resp. `secretPass` (vizte první verzi programu Login). Metoda vrací datový typ `bool`.
  Hodnotu `true` v případě úspěšné autentizace, `false` v opačném případě.
- Napište metody `ZadejJmeno()` a `ZadejHeslo()` pro získání uživatelského jména a hesla z klávesnice. Metody budou vracet datový typ `string`.

Kombinací výše popsaného by měl jít napsat následující kód do metody `Main()`:

```
public static void Main(string[] args)
{
    string jmeno = ZadejJmeno();
    string heslo = ZadejHeslo();

    if (Prihlaseni(jmeno, heslo) == true) {
        Console.WriteLine("Přihlášení bylo úspěšné.");
    }
    else {
        Console.WriteLine("Přihlášení selhalo.");
    }

    Console.ReadKey(false);
}
```

Řešení

```
public static string ZadejJmeno() {
        Console.Write("Přihlašovací jméno: ");
        string jmeno = Console.ReadLine();

        return jmeno;
}
```

```
public static string ZadejHeslo() {
        Console.Write("Přihlašovací heslo: ");

        ConsoleColor barva;
        barva = Console.ForegroundColor;

        Console.ForegroundColor = Console.BackgroundColor;
        string logPass = Console.ReadLine(); //změna
        Console.ForegroundColor = barva;

        return logPass; //dopsáno
}

public static bool Prihlaseni(string jmeno, string heslo) {
        string secretName = "admin";
        string secretPass = "12345";

        if (jmeno == secretName && heslo == secretPass) {
                return true;
        }
        else {
                return false;
        }
}
```

Test V

Obsah

Slovníky

Název školy	Vyšší odborná škola obalové techniky a Střední škola, Štětí, příspěvková organizace		
Adresa školky	Kostelní 134, 411 08 Štětí		
IČ	46773509		
Název operačního programu	OP Vzdělávání pro konkurenceschopnost		
Registrační číslo	CZ.1.07/1.5.00/34.1006		
Označení vzdělávacího materiálu			
Název tématické oblasti (sady)	Programování		
Název materiálu	Slovníky		
Anotace			
Autor	Ivan Pomykacz	**Jazyk**	čeština
Očekávaný výstup			
Klíčová slova			
Druh výukového zdroje		**Věková skupina žáků**	15+
Typ interakce	aktivita	**Ročník**	
Speciální vzdělávací potřeby	žádné		
Zhotoveno, (datum/období)		**Celková velikost**	

Unhandled exception Template /srv/www/dumy-data/pomykacz/ww-
-root/../../shell/res/tpl/ProgramovaniC/slovniky.html does not exists

Cykly

Název školy	Vyšší odborná škola obalové techniky a Střední škola, Štětí, příspěvková organizace		
Adresa školky	Kostelní 134, 411 08 Štětí		
IČ	46773509		
Název operačního programu	OP Vzdělávání pro konkurenceschopnost		
Registrační číslo	CZ.1.07/1.5.00/34.1006		
Označení vzdělávacího materiálu			
Název tématické oblasti (sady)	Programování		
Název materiálu	Cykly		
Anotace			
Autor	Ivan Pomykacz	**Jazyk**	čeština
Očekávaný výstup			
Klíčová slova			
Druh výukového zdroje		**Věková skupina žáků**	15+
Typ interakce	aktivita	**Ročník**	
Speciální vzdělávací potřeby	žádné		
Zhotoveno, (datum/období)		**Celková velikost**	

Obsah

Cyklus `while`

Cyklus `for`

Příklady

průměr vypište sudá/lichá čísla v intervalu 1..100 vypsat text po písmenkách (zpomaleně) zadej ano, ne ... nepustit, dokud nezadá správně hodiny / stopky / odpočet mysli si číslo hádej číslo

Zadání

Práce se soubory

Název školy	Vyšší odborná škola obalové techniky a Střední škola, Štětí, příspěvková organizace		
Adresa školky	Kostelní 134, 411 08 Štětí		
IČ	46773509		
Název operačního programu	OP Vzdělávání pro konkurenceschopnost		
Registrační číslo	CZ.1.07/1.5.00/34.1006		
Označení vzdělávacího materiálu			
Název tématické oblasti (sady)	Programování		
Název materiálu	Práce se soubory		
Anotace			
Autor		**Jazyk**	
Očekávaný výstup			
Klíčová slova			
Druh výukového zdroje		**Věková skupina žáků**	
Typ interakce		**Ročník**	
Speciální vzdělávací potřeby	žádné		
Zhotoveno, (datum/období)		**Celková velikost**	

Unhandled exception Template /srv/www/dumy-data/pomykacz/ww-
-root/../../shell/res/tpl/ProgramovaniC/cviceni-xv.html does not exists

Datum a čas

Název školy	Vyšší odborná škola obalové techniky a Střední škola, Štětí, příspěvková organizace		
Adresa školky	Kostelní 134, 411 08 Štětí		
IČ	46773509		
Název operačního programu	OP Vzdělávání pro konkurenceschopnost		
Registrační číslo	CZ.1.07/1.5.00/34.1006		
Označení vzdělávacího materiálu			
Název tématické oblasti (sady)	Programování		
Název materiálu	Datum a čas		
Anotace			
Autor		**Jazyk**	
Očekávaný výstup			
Klíčová slova			
Druh výukového zdroje		**Věková skupina žáků**	
Typ interakce		**Ročník**	
Speciální vzdělávací potřeby	žádné		
Zhotoveno, (datum/období)		**Celková velikost**	

Obsah

Statická třída `DateTime`

Pro práci s datem a časem lze použít třídu `DateTime`. Jde o statickou třídu, ze které lze tvořit i instance. Proto se můžete setkat s operátorem `new`, obdobně jako u třídy `Random`.

Dnes

Dnešní datum můžeme získat z vlastnosti `Today` třídy `DateTime`.

```
Console.WriteLine(DateTime.Today);
```

Pokud bychom chtěli uložit dnešní datum do proměnné, musí být typu `DateTime`, vizte příklad:

```
DateTime dnes = DateTime.Today;
Console.WriteLine(dnes);
```

Nyní

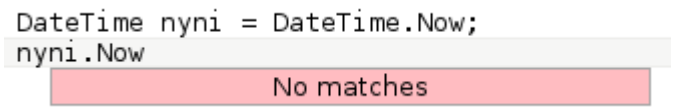
Naprosto stejně se chová i vlastnost `Now`, s tím rozdílem, že součástí data je i čas, který odpovídá času, kdy byla vlastnost použita.

```
DateTime nyni = DateTime.Now;
Console.WriteLine(nyni);
```

Objekt `DateTime`

Jaký že je rozdíl mezi statickou třídou a objektem? Máme-li *objekt* typu DateTime, tak je to vlastně již otisk statické třídy s konkrétním časem a datem uložený někde v paměti. Jakmile si tedy necháme vrátit čas vlastností `DateTime.Now`, získáme objekt, který obsahoval aktuální čas v době jeho vzniku.

Všimněte si, že objekt typu `DateTime` již nemá vlastnost `Now`.



A objekt jsme vytvořili pouze tím, že jsme do proměnné uložili aktuální čas.

Metody objektu `DateTime`

Objekt `DateTime` obsahuje metody pro manipulaci s datem a časem:

- `DateTime.AddHours()`
- `DateTime.AddDays()`
- `DateTime.AddMonths()`
- `DateTime.AddYears()`
- ...

Při zavolání metody se vrátí nový objekt `DateTime`.

Následující příklad uloží aktuální čas do proměnné `nyni`. Potom do proměnné `zahodinu` uloží nový čas, kde bude o hodinu více.

```
DateTime nyni = DateTime.Now;
DateTime zahodinu = nyni.AddHours(1);
Console.WriteLine(nyni);
Console.WriteLine(zahodinu);
```

Vlastnosti

Vlastnost objektu je vlastně "pouze" hodnota, kterou lze číst, případně měnit. Vlastnosti objektu musí být určitého datového typu (stejně jako proměnné). Některé vlastnosti objektu `DateTime`:

- `DateTime.AddDay`
- `DateTime.AddMonth`
- `DateTime.AddYear`
- `DateTime.DayOfYear`
- ...

Vlastní metody

Vytvořte metody `Zitra()` a `Vcera()`, které vrátí zítřejší, resp. včerejší datum.

Metody nebudou mít vstupní parametr (není potřeba). Návratová hodnota bude `DateTime`.

Jakmile metody vytvoříte, měl by fungovat následující kód v metodě `Main()`.

```
Console.WriteLine("Včera: " + Vcera());
Console.WriteLine("Dnes : " + DateTime.Now);
Console.WriteLine("Zítra: " + Zitra());
```

Náhodné číslo

Název školy	Vyšší odborná škola obalové techniky a Střední škola, Štětí, příspěvková organizace		
Adresa školky	Kostelní 134, 411 08 Štětí		
IČ	46773509		
Název operačního programu	OP Vzdělávání pro konkurenceschopnost		
Registrační číslo	CZ.1.07/1.5.00/34.1006		
Označení vzdělávacího materiálu			
Název tématické oblasti (sady)	Programování		
Název materiálu	Náhodné číslo		
Anotace			
Autor		**Jazyk**	
Očekávaný výstup			
Klíčová slova			
Druh výukového zdroje		**Věková skupina žáků**	
Typ interakce		**Ročník**	
Speciální vzdělávací potřeby	žádné		
Zhotoveno, (datum/období)		**Celková velikost**	

Obsah

Generování náhodných čísel

Chceme-li vygenerovat náhodné číslo, můžeme využít třídu `Random`.

Provedeme deklaraci proměnné typu `Random`, do které vložíme novou instanci třídy `Random`.

```
Random nahoda = new Random();
```

Generování náhodných čísel v počítači, který je striktně deterministický stroj, není až tak triviální. Proto je kód generování čísel umístěn ve své speciální třídě `Random`, která navíc není statická (jako např. třída `File`).

Není statická zejména proto, že zdrojem entropie je mimo jiné i systémový čas. Pokud bychom vytvářeli hodně instancí třídy `Random` bezprostředně po sobě, nemusela by generovaná čísla být až tak náhodná.

Náhodné číslo

Pro vygenerování náhodného čísla v rozsahu `int` stačí zavolat metodu `Random.Next()`, která vrací `int`.

```
Random nahoda = new Random();
int cislo = nahoda.Next();
```

Náhodné číslo z rozsahu

Pro vygenerování čísla z nějakého intervalu využijeme druhou podobu metody `Random.Next()`, kde stanovíme pomocí dvou vstupních parametrů spodní a horní hranici pro generování čísel (v příkladu 1 až 8).

```
Random nahoda = new Random();
int cislo = nahoda.Next(1, 8);
```

Barvy

Název školy	Vyšší odborná škola obalové techniky a Střední škola, Štětí, příspěvková organizace		
Adresa školky	Kostelní 134, 411 08 Štětí		
IČ	46773509		
Název operačního programu	OP Vzdělávání pro konkurenceschopnost		
Registrační číslo	CZ.1.07/1.5.00/34.1006		
Označení vzdělávacího materiálu			
Název tématické oblasti (sady)	Programování		
Název materiálu	Barvy		
Anotace			
Autor		**Jazyk**	
Očekávaný výstup			
Klíčová slova			
Druh výukového zdroje		**Věková skupina žáků**	
Typ interakce		**Ročník**	
Speciální vzdělávací potřeby	žádné		
Zhotoveno, (datum/období)		**Celková velikost**	

Unhandled exception Template /srv/www/dumy-data/pomykacz/ww-
-root/../../shell/res/tpl/ProgramovaniC/barvy.html does not exists

Program

Název školy	Vyšší odborná škola obalové techniky a Střední škola, Štětí, příspěvková organizace		
Adresa školky	Kostelní 134, 411 08 Štětí		
IČ	46773509		
Název operačního programu	OP Vzdělávání pro konkurenceschopnost		
Registrační číslo	CZ.1.07/1.5.00/34.1006		
Označení vzdělávacího materiálu			
Název tématické oblasti (sady)	Programování		
Název materiálu	Program		
Anotace			
Autor		**Jazyk**	
Očekávaný výstup			
Klíčová slova			
Druh výukového zdroje		**Věková skupina žáků**	
Typ interakce		**Ročník**	
Speciální vzdělávací potřeby	žádné		
Zhotoveno, (datum/období)		**Celková velikost**	

Obsah

Vlastnosti programů

- **Zvýraznění** - Program umí barevně zvýraznit nějakou část textu. Např. chybovou hlášku.
- **Záporné číslo** - Detekce záporného čísla v případech, kde nemá co dělat. Nejčastěji se objevuje při uživatelském vstupu. Příklad: "Zadej délku strany čtverce".
- **Uživatelská volba** - Uživatel si může zvolit, např. z voleb a) b) c). Program dle toho pak vykoná požadovanou akci. Příklad: "Chceš spočítat a) Obsah čtverce b) Obsah obdélníka c) Obsah trojúhelníka".
- **Smyčka programu** - Program běží v "nekonečné smyčce", dokud jej uživatel neukončí. Ukončením se nemyslí přerušení programu zavřením jeho okna (křížkem). Příklad: Program provede požadovanou akci (např. spočítá obsah čtverce). Po vypsání výsledku se program zeptá uživatele zda-li chce nový výpočet nebo skončit.
- **Opakovaná výzva** - Uživatel volí z nabídky voleb. Program jej nepustí dál, dokud nezvolí jednu z uvedených voleb. Nebo uživatel zadává číslo. Na číslo jsou kladeny nějaké požadavky, např. musí být kladné nebo z nějakého rozsahu.
- **Práce se soubory** - Program umí načítat obsah souborů nebo do souborů ukládat textová data. Program se dokonce může zeptat uživatele na název souboru, který má otevřít nebo do kterého se mají uložit data. Pro zjednodušení uvažujme soubory umístěné ve stejném adresáři jako je program.
- **...** -
- -

Caesarova šifra

Název školy	Vyšší odborná škola obalové techniky a Střední škola, Štětí, příspěvková organizace		
Adresa školky	Kostelní 134, 411 08 Štětí		
IČ	46773509		
Název operačního programu	OP Vzdělávání pro konkurenceschopnost		
Registrační číslo	CZ.1.07/1.5.00/34.1006		
Označení vzdělávacího materiálu			
Název tématické oblasti (sady)	Programování		
Název materiálu	Caesarova šifra		
Anotace			
Autor		**Jazyk**	
Očekávaný výstup			
Klíčová slova			
Druh výukového zdroje		**Věková skupina žáků**	
Typ interakce		**Ročník**	
Speciální vzdělávací potřeby	žádné		
Zhotoveno, (datum/období)		**Celková velikost**	

Obsah

```
class Program
{
        public static string abeceda =
"aábcčdďeéěfghiíjklmnňoópqrřsštťuúůvwxyýzž ,.:?!";
        public static int klic = 3;

        public static void Main(string[] args)
        {

                string sifra = Sifruj("ahoj světe");
                Console.WriteLine(sifra);

                string text = DeSifruj("cjqm");
                Console.WriteLine(text);

                Console.ReadKey(true);
        }

        public static string Sifruj(string text) {
                string sifra = "";

                int pozice;
                for (int i = 0; i < text.Length; i++) {
                        pozice = abeceda.IndexOf(text[i]);
```

```
                        if (pozice == -1) {
                                Console.WriteLine("Znak " + text[i] + " není
v abecedě!");
                        }
                        pozice = pozice + klic;
                        if (pozice >= abeceda.Length) {
                                pozice = pozice - abeceda.Length;
                        }
                        sifra = sifra + abeceda[pozice];
                }

                return sifra;
        }

        public static string DeSifruj(string sifra) {
                string text = "";

                int pozice;
                for (int i = 0; i < sifra.Length; i++) {
                        pozice = abeceda.IndexOf(sifra[i]);
                        if (pozice == -1) {
                                Console.WriteLine("Znak " + sifra[i] + " není
v abecedě!");
                        }
                        pozice = pozice - klic;
                        if (pozice < 0) {
                                pozice = pozice + abeceda.Length;
                        }
                        text = text + abeceda[pozice];
                }

                return text;
        }
}
```