Kapitola 1

Vývojová prostředí a překladače

V této sekci se budeme zabývat tím, jak získat, instalovat a používat vybrané překladače. Vzhledem k tomu, že předpokládáme, že naším čtenářem je průměrný student, zaměříme se na nástroje volně dostupné. Prostředí, které je placené, by mělo mít vlastní zákaznickou podporu. Svou pozornost zaměříme především na volně (zdarma) dostupné překladače společnosti Microsoft a na UNIXové prostředí MONO.

1.1 Microsoft Visual Studio .NET

Tomuto prostředí budeme nadále říkat pro stručnost jen Studio. Jedná se o komplikované prostředí vyvinuté (jak ostatně název napovídá) společností Microsoft. Toto prostředí nabízí několik programovacích jazyků. Zejména jsou zde překladače jazyků C++, Visual Basic, C# a implementace Microsoft SQL. Pro některé z nás překvapivě existuje od tohoto Studia volně dostupná verze. Kromě placených verzí Standard, Professional a Team System existuje (již od verze z roku 2005) verze "Express Edition", která je oproti plným verzím ořezaná, ovšem pro nás stále ještě plně použitelná. Soudobou verzí je ta z roku 2008 (tedy Visual Studio 2008), jiná (starší) zřejmě již ani není distribuována. Více se můžeme dočíst na stránkách společnosti Microsoft: `http://www.microsoft.com/cze/msdn/produkty/vstudio/default.mspx`. V současné době je možné získat zdarma i verzi Professional, ovšem pouze jako 90denní demoverzi, což nepovažujeme pro účely studia za ideální (a proto svou pozornost v textu nadále omezíme jen na Express Edition).

Jelikož předmětem našeho zájmu bude jazyk C#, kde by to bylo na újmu srozumitelnosti, omezíme se z celého Studia pouze na části zabývající se ja-

zykem C# (Express Edition je totiž oproti ostatním verzím rozdělena do modulů). Co platí pro ostatní moduly, si alespoň základním způsobem zkušený uživatel dovede domyslet. Je vhodné si každopádně uvědomit, že celé Studio sestává z mnoha modulů, které se nás budou různým způsobem pokoušet zmást. Například k běhu Studia je zapotřebí Microsoft Visual Studio .NET Framework, který je ovšem součástí instalace C# a proto mu nemusíme věnovat zvláštní pozornost (jen podotkněme, že pokusit se instalovat C# bez něj - tedy Framework "vyclicknout", až budeme dotázáni, zda jej chceme, není šťastné). Jako u celého jazyka se omezíme na stručný popis zvaný "kuchařka". Pokud by někdo toužil po detailech, jsou mu k dispozici v mnoha manuálech, které jsou ovšem v průměrném případě zbytečně podrobné.

1.1.1 Instalace a úvodní předvedení

Studio lze stáhnout na stránkách společnosti Microsoft, konkrétně na adrese `http://www.microsoft.com/express/download/default.aspx`. Tam jsme již v sekci *Download*. Na této stránce si buďto vybereme příslušné moduly k on-line instalaci (tedy v našem případě C#), anebo si můžeme stáhnout ISO-image DVD pro off-line instalaci. My se zaměříme na on-line instalaci, off-line instalace poběží, předpokládáme, přiměřeným způsobem podobně (samozřejmě až poté, co image vypálíme na DVD). Pro on-line instalaci je důležité stáhnout správný instalátor, konkrétně `vcssetup.exe`. Pokud se soubor, který jste stáhli jmenuje jinak, a to kupříkladu `vbsetup.exe` nebo `vcsetup.exe`, gratulujeme vám k volbě jiného programovacího jazyka, než C#. :-)

Instalace

Abychom vůbec mohli pomýšlet na úspěšnou instalaci, je potřeba splnit *instalační požadavky*, které jsou *softwarové*: Windows XP, 2003 nebo Vista a dále hardwarové.

Minimální požadavky na hardware jsou: Procesor taktovaný na aspoň 1,6 GHz, 192 MB paměti RAM, rozlišení displaye aspoň 1024 x 768 a disk točící se aspoň 5400 krát za minutu.

Doporučený hardware má být ještě silnější, a to: Procesor taktovaný aspoň na 2,2 GHz, aspoň 384 MB paměti RAM, display s rozlišením 1280 x 1024 a disk, který umí aspoň 7200 otáček za minutu. Já jsem pracoval na 1,6GHz stroji s 1024 MB paměti, maximálním rozlišením displaye 1024 x 768 a diskem s 5400 otáčkami za minutu. Při instalaci stroj silně lapal po (chladném) vzduchu a instalace trvala kolem hodiny. Po instalaci jsem zpozoroval zábor

místa na disku kolem 0,5 GB. Samotné prostředí pak běží bez zásadních obtíží (použitelně rychle). Instalace samotná je sice prakticky plně automatizovaná, jediná interakce vyžadovantá ode mě bylo rebootování stroje, což však bylo (na pokyn instalátoru) potřeba udělat hned dvakrát, nelze tudíž počítat, že instalace proběhne, zatímco budeme pryč.

Instalace sama o sobě není nijak komplikovaná. Instalátor se vyptá, kam chceme instalovat, a doporučí nám, co chceme instalovat. Kromě C# nám doporučí ještě *Microsoft SQL*, což je databázové prostředí, které se dle mého názoru může časem hodit, proto jsem proti jeho instalaci neprotestoval. Dále se mi instalátor pokusil podsunout webový browser *Silverlight* od Microsoftu, který lze bez vážných následků odmítnout. Stejně tak lze odmítnout *RSS* (obzvláště pokud chcete být často odpojeni od sítě), jelikož se má jednat o on-line rady od společnosti Microsoft.

Registrace

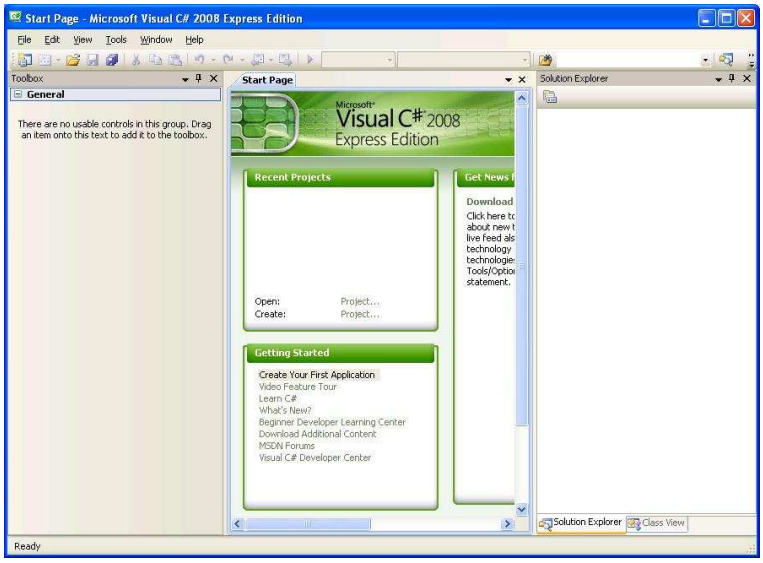
Nyní nastává čas prostředí spustit a začít používat. Najděte v menu (*Start* → *Programy*) položku `Microsoft Visual C# Express Edition`. Po chvíli startování se objeví okno s *IDE* (integrated desktop enviroment – tedy integrované prostředí). Od této chvíle vás začíná pronásledovat poznámka, že prostředí můžete sice používat zdarma, ale do třiceti dnů je nutné se registrovat, nebo fungovat přestane.

Registrace je podobně automatizovaná jako instalace, tedy stačí kliknout na odkaz v okně, které se objeví vybereme-li *Help* → *Register product* (otevře se nám webový prohlížeč a dovede nás na stránky společnosti Microsoft). Tam je třeba vyplnit požadované osobní údaje (případně – pokud jste tak ještě neučinili, se registrovat i tam) a po chvíli vyplňování získáme registrační číslo, které zadáme do registračního okénka, které při troše štěstí ještě máme otevřené někde pod hromadou oken webového prohlížeče.

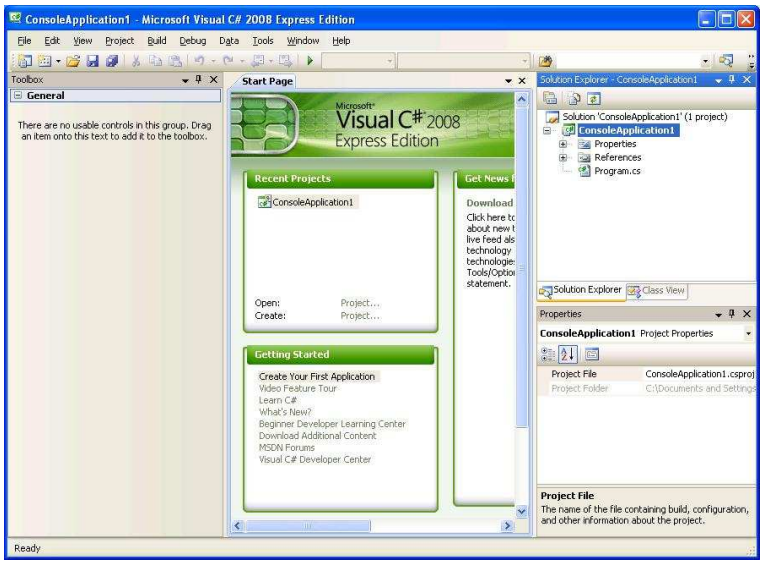
Na této proceduře je jistým kamenem úrazu právní nejistota, jelikož poskytujete své osobní údaje společnosti Microsoft, která sídlí v Redmondu a tudíž podléhá dozoru amerických úřadů, které jsou sice přísné, ale řídí se americkou legislativou, tedy například nepochodíte se zákonem 101/2000 Sb. "Na ochranu osobních údajů". Abychom tak mohli říci, co se společnost Microsoft ještě smí odvážit s našimi údaji udělat, a co ne, museli bychom nastudovat příslušné právní normy Spojených států amerických.

První seznámení

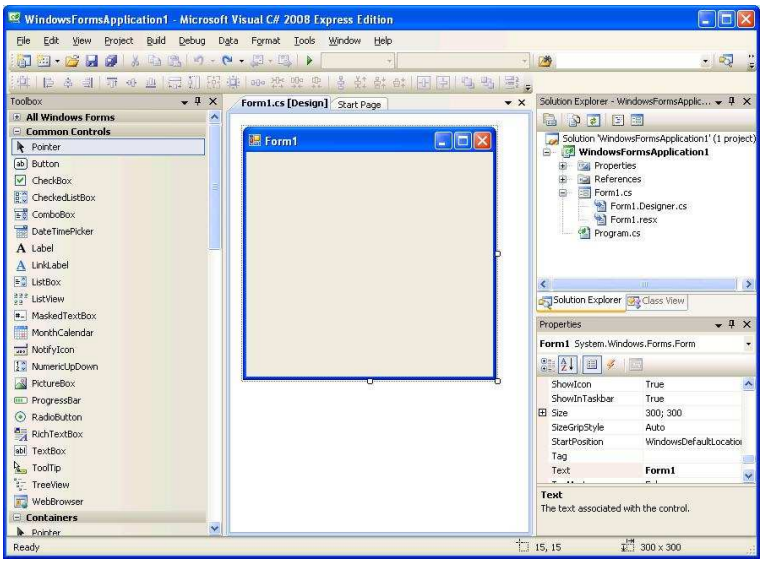
Po administrativních záležitostech již není vyhnutí a je třeba přistoupit k práci se samotným IDE. Bez újmy na obecnosti začněme tvorbou nového



Obrázek 1.1: Demonstrace Visual Studia ihned po startu.



Obrázek 1.2: Projekt **ConsoleApplication1** je otevřen, jak vidíme v pravé třetině v *Solution Exploreru*, ale okno editoru zdrojáků se neotevřelo. Toho dosáhneme kupř. dvojkliknutim na text `Program.cs` v *Solution Exploreru*. Porůznu se stává, že Visual Studio neudělá přesně to, co očekáváme, a proto je nutné si hned zkraje osvojit určité návyky a zejména si povšimnout, jak které okno vypadá a kde o něj můžeme v případě potreby požádat, pokud nás o něj Studio "ošidí".

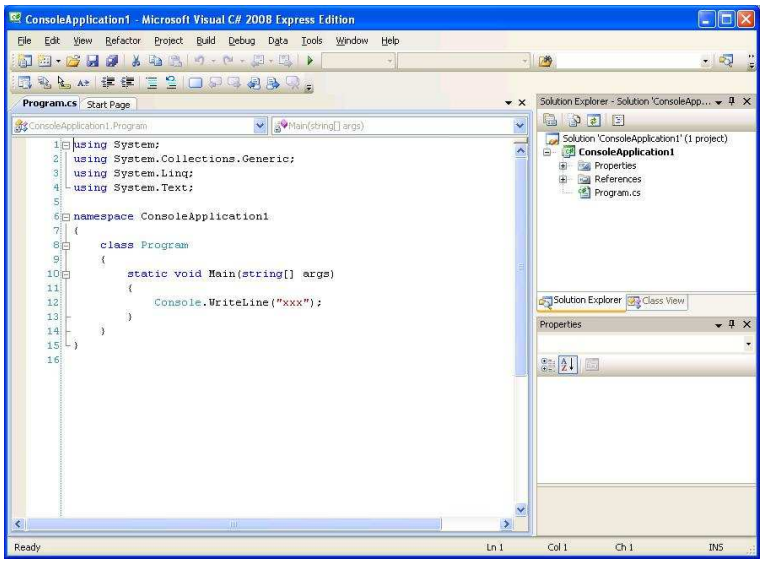


Obrázek 1.3: Demonstrace *Visual Studia* po založení *Windows Forms Application*. V levé části okna je *Toolbox*, v prostřední formulář a v pravé Solution Explorer. V horní části právé třetiny vidíme strukturu projektu, v dolní části pak atributy jednotlivých objektů na formuláři, případně samotného formuláře.

projektu, tedy vyberme v menu *File* → *New Project*. Projekt pojmenujme dle vlastního uvážení (implicitně se nám nabízí `WindowsFormProject1` či `ConsoleApplication1` a podobně) a současně musíme vybrat typ projektu. Na typu projektu velmi záleží, jelikož tím stanovíme, jak bude naše dílo vypadat. Nás budou zajímat především dva typy projektů: *Windows Forms Application* (první zleva) a *Console Application* (druhý zprava). Větší část výuky absolvujeme s tím druhým typem (tedy *Console Application*), v této chvíli (a jedné lekci) pro nás však bude mnohem zajímavější *Windows Forms Application*. Vyberme tedy typ *Windows Forms Application*.

IDE Visual Studia nám umožňuje provádět různé úkony související s programováním, a tedy je poměrně komplikované. Povšimněte si, že po potvrzení typu aplikace se obsah okna úplně změnil – můžeme přímo říci, že se zkomplikoval. K základní orientaci nám může pomoci uvědomit si, že během práce můžeme být konfrontováni s několika desítkami oken různých typů, a proto jsou tato okna roztříděna do alespoň čtyř kategorií (podle toho, která okna má smysl zobrazovat současně). Vidíme tedy, že okno je rozdělené na tři sloupce.

V levém sloupci vidíme panel *Toolbox*, v pravém sloupci je panel *Solution Explorer* a v prostřední třetině vidíme okno s něčím, co nám správně při-



Obrázek 1.4: Takto vypadá Visual Studio po založení Console Application.

pomíná vzhled okna naší první aplikace hlásící se jako `Form1.cs[Design]`. Vybereme-li v levém panelu (*Toolbox*) pomocí myši položku *Button*, najedeme myší nad okno `Form1.cs[Design]` (dále zvané formulář), konkrétně nad okénko `Form1`, zmáčkneme levé tlačítko myši a se stisknutým tlačítkem myší mírně zatáhneme a tlačítko myši uvolníme, uvidíme, že jsme na okno aplikace nakreslili tlačítko. Vidíme tedy, že levou třetinu (*Toolbox*) má smysl zobrazovat s obsahem prostřední třetiny (tedy formulářem). Co právě chceme dělat, je řízeno z pravé třetiny (`Solution Exploreru`). V tom vidíme stromovou strukturu celého našeho projektu. V dolní polovině pravé třetiny vidíme okno *Properties* určující vlastnosti jednotlivých prvků na formuláři, v horní části pravé třetiny je samotný `Solution Explorer`, který nám umožňuje celou práci řídit. Výběrem jednotlivých položek řekneme, co chceme právě dělat. V současné době pracujeme s `Form1.cs`. Pokud vybereme (dvojkliknutím myší) kupříkladu `Form1.Designer.cs`, levá a prostřední třetina se překreslí a my uvidíme zdrojový kód v C#, který můžeme modifikovat. Vybereme-li `Form1.cs`, objeví se původní obsah, tedy v levé třetině *toolbox* plný grafických prvků a v prostřední třetině obrázek formuláře. Všimněme si, že v prostřední třetině máme nyní dvě záložky: `Form1.Designer.cs` (zdrojový kód) a `Form1.cs[Design]` ("malování"). Používání záložek (též zvaných *taby*) je pro Visual Studio velmi typické. Přestože je ovládání tzv. "intuitivní", často se stává, že se obrazovka nepřekreslí (případně tomu svou netrpělivostí nějak napomůžeme) a přestože to, co jsme chtěli, je otevřené, příslušný tab je schovaný za něčím, o co teď naprosto nestojíme a správný soubor si musíme

vybrat. Typicky se toto stane hned při dalším spuštění Visual Studia. Pokud tento svůj pokus nyní uložíme (Ctrl+Shift+S) a při dalším spuštění se k němu chceme vrátit, můžeme si jej vybrat hned na startovní obrazovce mezi *Recent projects*. Pokud na něj clickneme myší jednou, projekt se otevře a obrazovka by se měla překreslit. Občas (za ne zcela jasných okolností) se však stane, že se obrazovka nepřekreslí a člověk stále kouká na *Start Page* (viz obrázek 1.2. Projekt přitom je otevřený a jen je potřeba dostat správný tab do prostřední třetiny. Problémy se zobrazováním jednotlivých panelů nám většinou pomůže vyřešit menu *View*. Taktéž je možné přenastavit si vzhled Studia například tak, aby se panel *Toolbox* skrýval kdykoliv z něj odjedeme myší. Proto se není možno spoléhat na přesné chování a umístění jednotlivých okének a je tudíž třeba se naučit, jak se kterému oknu říká a jak se k němu dostat.

Jako další (a poslední) pokus v této sekci můžeme vytvořit novou *Console Application* (vybereme v menu *File* → *New Project...*, pojmenujeme a vybereme typ *Console Application*). Povšimněme si, že najednou okno vypadá úplně jinak. Místo levé a prostřední třetiny máme okno přes dvě třetiny obrazovky, nevidíme žádné "malování", ale něco, co nám připomíná IDE Pascalu, známé z minulého semestru, mnohem spíše. Jde o důsledek toho, co jsme si už řekli: Prostředí nás může ohromit desítkami oken různých typů, jenže ne vždy mají všechny typy oken dobrý smysl. Chceme-li vytvořit konsolovou aplikaci, budeme tvořit program, který má číst data ze vstupu a vypisovat na výstup podobně, jako to dělal kupříkladu Pascal. Proto je "klikací rozhraní" triviální (nemá smysl je definovat) a proto nám je překladač ani nenabízí. Právě konsolová aplikace bude typ aplikace, kterému se budeme věnovat větší část semestru, jelikož předmětem výuky nebude estetická stránka věci (jaký má být poměr velikosti okna a velikosti buttonu), ale technická, tedy jak donutit program ze zadaných dat vyrobit nějaký výstup. Jeho estetické zpracování nás zpravidla nebude zajímat.

Na závěr této sekce konstatujme, že C# je jazyk poskytující množství funkcionalit, ať už se jedná o možnost grafického návrhu vzhledu aplikace, nebo o to, že je to jazyk objektový a velmi nový, založený na všem, co bylo k dispozici v době jeho vzniku. Na jedné straně se jedná o jazyk velmi pohodlně použitelný, na straně druhé je potřeba mu poměrně detailně rozumět, abychom všech jeho schopností dovedli využít. Jelikož nás budou primárně zajímat algoritmy, pokusíme se omezit na co nejmenší podmnožinu jazyka, což bude náročné, jelikož časem zjistíte, že jazyk žije vlastním životem zcela odlišným od toho, jak se vám jej pokusíme stručně přiblížit. V současné chvíli však není možné vysvětlit vám cosi, co bychom mohli nazvat "vnitřní logikou" jazyka. Buďte proto, prosím, při studiu trpěliví. Několikařádková chybová hlášení zpočátku nedávají příliš dobrý smysl, hlášení v Pascalu o pár slovech byla v tomto ohledu mnohem srozumitelnější. Zřejmě však není možné chybu

popsat snáze. Mimo jiné právě s ohledem na rozsáhlost jazyka překladač sice zjistí, že jste udělali něco špatně, ale nemůže snadno říci, co jste asi chtěli udělat a poradit vám. Občas se dokonce stane, že člověk omylem vyrobí text, který je sice platným zdrojovým kódem, ale znamená něco úplně jiného, než si představoval. Pak se typicky stává, že chybová hlášení naprosto nedávají smysl, a je potřeba je zinterpretovat tak, že jste asi udělali chybu, většinou se i shoduje přibližné místo chyby, ale místo nápovědy překladače (o tom, co je předmětem chyby) je třeba použít vlastní úvahu a zkusit text přepsat jinak.

1.1.2 Sdílení projektů mezi různými verzemi Studia

Velmi snadno se nám může stát, že budeme chtít pracovat na jednom projektu na více místech současně, tedy část práce budeme chtít udělat doma, část ve škole, případně se může stát, že předvedení práce má proběhnout na jiném než domácím stroji a na problém bude zaděláno ve chvíli, kdy na těchto místech budou různé verze Studia. Lze si vynutit automatickou konverzi pro novější Studio. Horší to však je, pokud chceme projekt vytvořený v novějším vývojovém prostředí otevřít pomocí nějaké starší verze. Zřejmě nejčastěji se stane, že tyto dvě verze budou Studio 2008 a 2005. Proto si popíšeme, co v takovém případě dělat. Samozřejmě změny ve verzi 2008 byly komplexní, tedy musíme dát již při vývoji pozor, abychom nepoužívali to, co není ve staré verzi implementováno (což je v tuto chvíli těžké popsat, na druhé straně my bychom měli používat část jazyka implementovanou v obou verzích).

Klíčové je umístit přenášený projekt (v cílovém počítači) na místo (disku), kde jej lze spustit. V opačném případě můžeme sice všechno dělat správně, ale buďto nás upozorní hroutící se `vshost.exe`, nebo jakýkoliv pokus o spuštění vyústí v ArgumentException. V takovém případě je projekt potřeba přestěhovat jinam, a to nejlépe na místo, kde Studio samo vytváří své projekty.

Máme-li projekt vytvořený ve Studiu 2008 úspěšně okopírovaný do vhodného adresáře na stroji osazeném Studiem 2005, je potřeba oeditovat následující soubory libovolným textovým editorem podporující formát holého textu (kupř. notepadem). Předpokládejme, že projekt se jmenuje `nazev_projektu`:

- Soubor solution (`nazev_projektu.sln`).
  V tom je zapotřebí změnit první dva řádky takto:

  ```
  Microsoft Visual Studio Solution File, Format Version 9.00
  # Visual Studio 2005
  ```

  původně tyto řádky zněly asi takto:

```
Microsoft Visual Studio Solution File, Format Version 10.00
# Visual C# Express 2008
```

První řádek byl volný, což je lepší ponechat (i když při testu na Studiu 2008 jsem nepozoroval problémy po vymazání tohoto prázdného řádku).

- Soubor projektu (`nazev_projektu\nazev_projektu.csproj`).
  - V tomto souboru změníme hodnotu atributu `ToolsVersion` (který najdeme nejspíše na druhém řádku) ze 3.5 na 2.0 takto:

    ```
    <Project ToolsVersion="2.0" DefaultTargets="Build"
    xmlns="http://schemas.microsoft.com/developer/msbuild/2003">
    ```

  - Kousek níže pak musíme změnit verzi produktu z 9.0.30729 (nebo podobné) na 8.0.5072, tedy řádek z tvaru

    ```
    <ProductVersion>9.0.30729</ProductVersion>
    ```

    přepíšeme na

    ```
    <ProductVersion>8.0.50727</ProductVersion>
    ```

  - Mnohem dále (až poblíže konce souboru) změníme `MSBuildToolsPath` na `MSBuildBinPath` a `Microsoft.CSharp.targets` na `Microsoft.CSharp.Targets` (tedy zvětšíme počáteční písmeno ve slově `targets`). Příslušný řádek má tudíž vypadat asi takto:

    ```
    <Import Project="$(MSBuildBinPath)\Microsoft.CSharp.Targets" />
    ```

  - Potom ještě můžeme vyházet Includy, které nejsou ve Studiu 2005 definovány, zejména:

    ```
    <Reference Include="System.Core">
      <RequiredTargetFramework>3.5</RequiredTargetFramework>
    </Reference>
    <Reference Include="System.Xml.Linq">
      <RequiredTargetFramework>3.5</RequiredTargetFramework>
    </Reference>
    <Reference Include="System.Data.DataSetExtensions">
      <RequiredTargetFramework>3.5</RequiredTargetFramework>
    </Reference>

    <Reference Include="System.Deployment" />
    <Reference Include="System.Drawing" />
    <Reference Include="System.Windows.Forms" />
    ```

Tento krok ovšem můžeme přeskočit (a provést později v IDE).

Nyní již otevřeme takto ošetřenou solution ve Visual Studiu 2005 (`File` → `Open` → `Project/Solution`). Pokud jsme tak neučinili, je třeba zbavit se odkazů ve Studiu 2005 nedefinovaných. Zkusíme proto solution zkompilovat, abychom zjistili zbývající chyby. Některé z nich budou odkazy na neexistující součásti systému (pokud jsme přeskočili poslední krok editování projektového souboru, nebo při jeho provádění něco opomenuli). Po dvojclicknutí na jednotlivé chyby uvidíme místo, kde příslušné chyby nastaly. Uvidíme chyby dvou typů:

1. Chyba odkazuje k Solution Exploreru. V tomto případě vidíme zpravidla ikonu zničeného souboru (odkazující k neexistujícímu souboru). Tuto chybu opravíme smazáním odkazů na příslušné (neexistující) soubory stiskem klávesy `delete`).

2. Chyba nastala ve zdrojovém textu. V tomto případě jde zpravidla o řádku "`using System.Linq;`" a řešením chyby je tuto řádku smazat (samozřejmě za předpokladu, že jsme nic ze `System.Linq` nepoužili v programu).

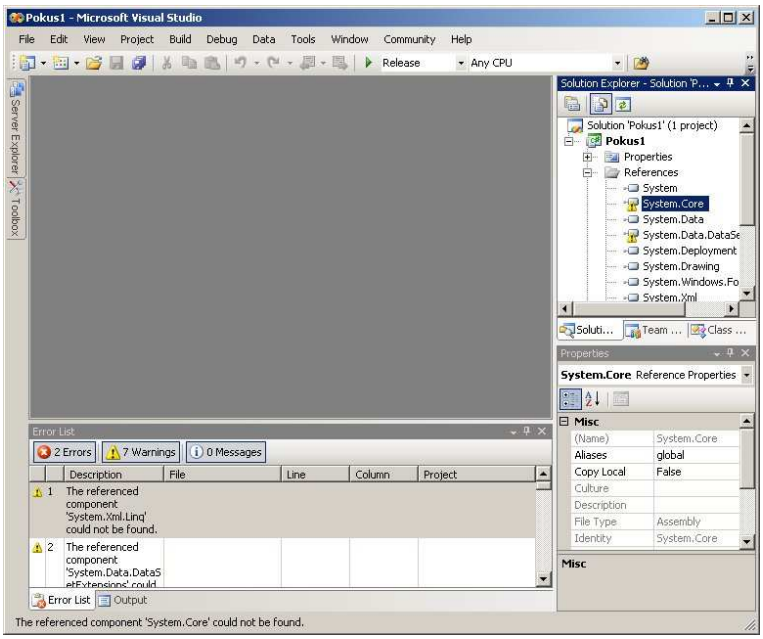
Po provedení tohoto postupu bychom měli mít projekt kompilovatelný a spustitelný Studiem 2005.

1.1.3 Užitečné konfigurace

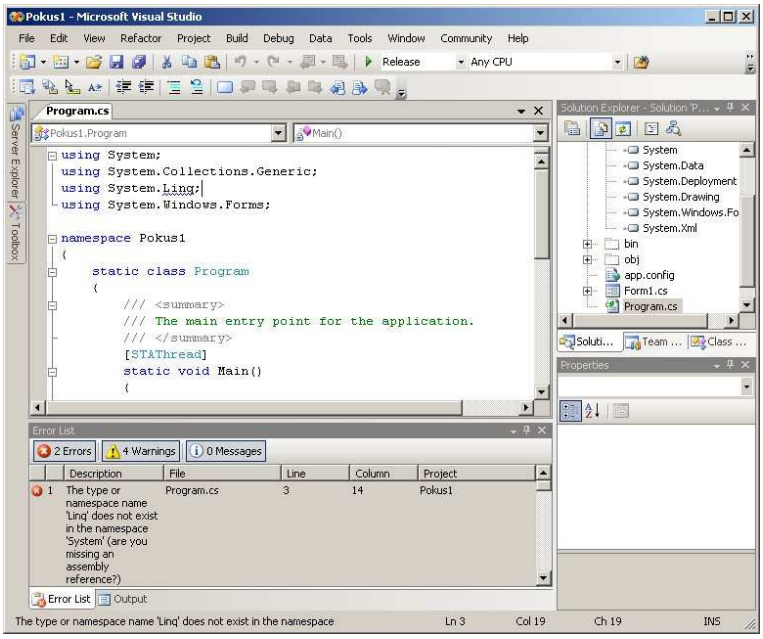
Obsah projektu (Soubory a jejich významy)

Studio ukládá jednotlivé projekty implicitně do adresáře *Dokumenty*, tedy zpravidla
`Plocha` → `Dokumenty` → `Visual Studio 2008`.

V projektech se nacházejí tyto soubory a adresáře (`Dokumenty\Visual Studio 2008\Projects\Pokus1` – uvědomte si, že `Pokus1` je název projektu):
`Pokus1.suo` (Binární soubor, který lze smazat a Studio vygeneruje nový)
`Pokus1.sln` (Solution-file popisuje zejména polohu souboru s projektem)
`Pokus1` (adresář obsahující následující):
+ `bin` (adresář obsahující binární soubory včetně spustitelného programu, jeho obsah můžeme smazat, kompilátor jej i s obsahem po další kompilaci vytvoří znovu)
+ `obj` (podobně jako `bin`)
+ `Properties` (adresář obsahující zejména AssemblyInfo)
+ `Pokus1.csproj` (soubor projektu – popisuje, co do něj patří)



Obrázek 1.5: Příklad nedefinovaného odkazu. V Solution Exploreru vidíme několik ikon poničených souborů, které smažeme.



Obrázek 1.6: Příklad chyby ve zdrojovém textu. Používáme `System.Linq`, což není ve Studiu 2005 definováno. Příslušný řádek smažeme.

\+ a plno dalších souborů. Ty s koncovkou `.cs` jsou zpravidla zdrojové kódy C#, ostatní jsou zpravidla soubory obsahující nějaké XML-popisy různých konfigurací. Ovlivňují například obsahy adresářů `bin` a `obj`, tedy i obsah podadresářů `Debug` a `Release`...

Adresáře `bin` a `obj` lze bez obav smazat, překladač jejich obsah vygeneruje v případě potřeby ze zdrojových kódů znovu.

Co z projektu přenášet?

Problém: Chci cvičícímu poslat zápočtovou úlohu, nechci mu zavalit mailovou schránku a nechci si nechat pro sebe něco důležitého.

Nyní je rozhodující, zda je aplikace consolová, nebo okenní a v kolika souborech máme zdrojové texty.

Pokud jde o consolovou aplikaci v jednom souboru, stačí obclicknout zdrojový text a přibalit k mailu (cvičící si založí vlastní projekt, do kterého zaslaný zdroják nějakým způsobem okopíruje). Pokud má zdrojový text consolové aplikace více zdrojových souborů, je třeba zaslat obsah všech souborů. V takovém případě je vhodné obsah všech souborů "zchmoustat" do jednoho. Jen je třeba dát pozor například na neidempotentní úkony (jejichž provedení jednou a podruhé není to samé) a na úkony, kdy není totéž, když dva dělají totéž (tedy například pokud si definujete ve vlastním jmenném prostoru metodu Output a v jednom souboru vyvoláte `using` na tento svůj jmenný prostor, v jiném souboru přidáte "`using System;`" a v obou použijete metodu Output, aniž byste specifikovali, ze kterého jmenného prostoru má být. Tím se vyhnete případným problémům s přenášením projektu mezi různými verzemi Studia.

Pokud se jedná o Windows Application, je situace komplikovanější. Pak zřejmě nezbývá, než celý projekt zabalit a poslat (lze jej ovšem očesat o adresáře `bin` a `obj`).

Jak zapnout číslování řádků?

Číslo řádku při vypnutém číslování zjistíme dole na liště `(Ln 1 Col 1 Ch 1 INS)`.

K zapnutí čísel řádků vybereme v menu: *Tools* → *Options*. V okně vybereme v levé části položku *Text Editor* → *C#*. V dolní části okna zaklikněme *Show all settings* a v pravé části okna se objeví checkbox *Line numbers* (který číslování řádků ovládá). Důležité je nezapomenout zakliknout *Show all settings*, jinak se tato volba vůbec neobjeví!

Průšvih! Smazal jsem si z *Toolboxu* omylem prvek *Button*. Nepomáhá ani když Studio restartuju! Co s tím?

Prvky z *Toolboxu* se dají ostranit (pravé tlačítko myši a v menu vybereme *delete*). Prvek zmizí a nepomůže ani když Studio restartujeme. Zatímco

ve škole si můžeme "na hulváta" přesednout k jinému počítači, doma pomůže opět pravé tlačítko myši (nad toolboxem) a tentokrát vybereme *Reset Toolbox* a máme toolbox jako nový.

Obecně je užitečné problémy řešit za pomoci webových vyhledávačů. Kromě našeho návodu k prvnímu problému (zapnutí číslování řádků) můžeme postupovat takto: Stručně si problém popíšeme v Angličtině, vybereme klíčová slova z popisu a zadáme do vyhledávače. Tedy například zadáme *"line numbering C#"* do *Googlu* a třetí odkaz, co vypadne, je návod pro *Studio 2005*. Funguje jen částečně (tehdy nebylo potřeba zaklikávat *Show all settings*). Proto zkusíme dotaz upřesnit na *"line numbering C# 2008"* a tento trik se dozvíme hned v prvním odkazu [údaje jsou platné ke dni 16. 10. 2008, od té doby se struktura stránek změnila a čísla odkazů na Googlu mohou být jiná].

1.2 MONO

Jedná se o open-source projekt mající za cíl implementaci .NET Frameworku pro různé platformy (projekt si říká cross-platformní, tedy neměl by být omezen jen na jeden konkrétní operační systém na jednom konkrétním typu počítačů). My se omezíme na instalaci pro Linux. Instalovat můžeme buďto přímo ze zdrojových textů, anebo ve vybraných distribucích Linuxu přímo binární balíčky.

1.2.1 Instalace ze zdrojových textů

Instalace ze zdrojových textů probíhá tak, že si stáhneme zdrojáky (kupř. z adresy `http://ftp.novell.com/pub/mono/sources-stable/`). Stáhneme soubor `mono-2.0.1.tar.bz2`, který obsahuje překladač a interpret[1]. Rozbalíme pomocí
`tar xjf mono-2.0.1.tar.bz2`, vstoupíme do adresáře MONO: `cd mono-2.0.1`
a zkompilujeme standardním způsobem, tedy sekvencí `./configure`, `make` a `make install`. Takto můžeme postupovat, pokud máme správcovská práva (jsme uživatel root). Pokud jsme obyčejný uživatel, buďto nepříjemnou práci s instalací necháme na správci, nebo si instalujeme vlastní MONO v domovském (nebo jakémkoliv jiném) adresáři, do kterého máme práva zapisovat. Cílový adresář musíme oznámit skriptu `configure` pomocí parametru `--prefix` do kterého přiřadíme cílový adresář, tedy kompilace bude vypadat například:

```
./configure --prefix=/home/user/me-krasne-mono
```

[1] Součástí názvu je i verze, jejíž číslo se bude časem zřejmě zvyšovat!

```
make
mkdir /home/user/me-krasne-mono
make install
```

Projekt je dobře dokumentován, tedy při kompilaci a instalaci se lze řídit přibalenou dokumentací. *Požadavky na instalovaný software* (ktery bude pouziván při kompilaci a běhu) lze nalézt v souboru `README` a jde (ve verzi 2.0.1) o *GTK 2.4* a `pkg-config`, volitelně pak (pokud nechcete přijít o vybrané funkcionality) `libgdiplus` a `libzlib`. Závislosti se mohou mezi verzemi změnit, věnujte proto, prosim, pozornost souboru `README` (to plati při instalovani jakéhokoliv software sířeného pod GNU GPL).

1.2.2 Instalování binárních balíčků

V některých distribucích můžete využít již předpřipravené binární package. Jde kupříkladu o distribuce *Debian, Ubuntu* nebo *Fedora*. Instalování je v tomto případě až nečekaně pohodlné. V mém případě probíhalo na distribuci Debian (konkrétně Etch). Stačí jen spustit package manager (na Debianu `dselect`) a po chvíli hledání se objevil balík *mono-gmcs* s mnoha závislostmi. Instalace proběhne standardním způsobem a po pár minutách máte překladač C# na Linuxu. Zatím jsme popsali výhody. Nevýhodou je, že stabilní verze Debianu patrně dosud neobsahuje *MonoDevelop*, což je IDE podobné tomu, se kterým člověk přijde do styku ve *Visual Studiu*. Toto je k dispozici až v testing verzi Debiana (zvané Lenny). Další nevýhodou instalace binárních balíčků je, že se balíčky jmenují na různých distribucich různě a může být zapotřebí určité tvořivosti k jejich nalezení. Krom toho balíčky může instalovat pouze správce (tedy uživatel *root*). A konečně u binárního balíčku nemáte kontrolu nad tím, jaký kód vlastně spouštíte. Může se tak stát, že si stáhnete balíček z pochybného zdroje i s nežádoucí funkcionalitou spočívajíci v lepším případě v rozesílání spamu, v horším pak ve slídění po vašem stroji.

1.2.3 Úvodní předvedení

Nyní (po instalování balíku `mono-2.0.1.tar.bz2` případně jeho ekvivalentu) máte instalované prostředí *MONO*. Pro začátek svou pozornost omezíme na programy `gmcs` a `mono`. Ten první je překladač jazyka C# a druhý interpret. Pokud chceme vytvořit tradiční program *Hello World*, připravíme si zdrojový text v jazyku C# do nějakého souboru, kupříkladu `hello.cs`. Tento zdrojový soubor můžeme vyrobit v jakémkoliv textovem editoru podporujicím formát holého textu (kupř. `vi`, `emacs` nebo `pico`, osobně používám ten první, ten třetí člověku rád přeformátovává vstup způsobem vhodným pro

```
using System;
class Hello
{
        static void Main()
        {
                Console.WriteLine ("Hello, World!");
        }
}
```

Obrázek 1.7: *Hello world* – tento text si připravíme do souboru `hello.cs`.

přirozený, nikoliv programovací jazyk). Program zkompilujeme pomocí překladače `gmcs`, tedy napíšeme:
`gmcs hello.cs`

Pokud jsme ve zdrojovém textu udělali chyby, objeví se chybová hlášení. Pokud ne, překladač nic neřekne a objeví se soubor `hello.exe`, což je binární (spustitelný) kód. Ke spuštění musíme zavolat interpret `mono`, tedy napíšeme: `mono hello.exe` a interpret spustí příslušný program.

Rejstřík

Console Application, 5–7

gmcs, 14

IDE, 3

mono, 14

Properties, 6

Solution Explorer, 5

Tab, 6
Toolbox, 5

Windows Forms Application, 5