MODERNÍ VÝUKA PROGRAMOVÁNÍ

Radek Vystavěl
Katedra informatiky
Vysoká škola manažerské informatiky a ekonomiky
Vltavská 14, 150 00 Praha 5
vystavel@vsmie.cz

ABSTRAKT:
V příspěvku jsou formulovány otázky autorem považované za klíčové při vytváření kursu programování. Jako odpovědi na tyto otázky je představen autorův úvodní kurs programování na Vysoké škole manažerské informatiky a ekonomiky.

KLÍČOVÁ SLOVA:
moderní výuka, výuka programování, učebnice programování

Cílem tohoto příspěvku je podnítit diskusi k výuce programování na vysokých a středních školách s důrazem na výuku studentů bez předchozích programátorských zkušeností.
V příspěvku rozebírám otázky, jež pro koncepci výuky programování považuji za nejdůležitější Současně zde uvádím, jak jsem na tyto otázky odpověděl ve svém dvousemestrálním úvodním kursu programování, který vyučuji na Vysoké škole manažerské informatiky a ekonomiky.

CÍLE VÝUKY PROGRAMOVÁNÍ

V počátcích výuky informatiky se informatika de facto redukovala na programování. Učit se něco o počítačích znamenalo učit se programovat. S rozvojem osobních počítačů a jejich aplikačního programového vybavení byla tato koncepce opuštěna. Převážil uživatelský přístup s tím, že výuka programování se zaměřila na informaticky orientované studijní obory. Stejně je tomu i na naší škole, která kromě studia informatiky nabízí ještě studium ekonomie. Programovat se učí pouze informatikové.

Nutno však poznamenat, že informatikové nejsou homogenní skupinou. Vývoj softwaru bude mít jako hlavní náplň práce pouze část z nich. Ostatní se učí programovat jednak proto, aby tak lépe porozuměli fungování počítačů, jednak proto, že je dobré umět si něco naprogramovat, jelikož veškerý software potřebný pro činnost jakékoli firmy se obvykle koupit nedá a přinejmenším menší věci je potřeba si samostatně vytvořit.

Tato skutečnost byla jedním z důvodů, proč jsem se ve svém kursu nepřiklonil k výukovému modelu „objects first“ [1], který navrhuje seznamovat začátečníky nejprve s koncepcí objektů a tříd na základě argumentu, že pak v budoucnu budou studenti lépe zvládat návrh velkých programů. Za problém modelu „objects first“ považuji vyšší míru abstrakce a delší čas potřebný k tomu, aby se studenti dostali k zajímavým aplikacím. Myslím, že je důležité, aby studenti hned od počátku viděli, jaké zajímavé výsledky programování může přinášet, aby co nejdříve zažili radost z úspěšného vytvoření zajímavého programu. Rovněž považuji za důležité, aby vše, čím ve svých začátcích studenti procházejí, bylo co možná nejkonkrétnější.

AKTUÁLNOST KURSŮ

Obecnou vlastností školství, a to nejen českého, je jeho obrovská setrvačnost. Výuka v řadě předmětů odráží stav poznání někdy před 50 či 100 lety. Nové zajímavé věci se do učebnic dostávají jen obtížně a stejně tak obtížně se naopak vyřazují věci, jejichž význam už průběhem času poklesl. Jakýmsi způsobem se v potřebě naučit velké množství studentů ztrácí prvek tvořivosti učitelů a autorů. Důsledky tohoto stavu na smysluplnost výuky a následně na motivaci studentů jsou nasnadě.

V informatice a programování není zmíněné riziko zastarání tak velké, jednak samozřejmě díky relativně krátké historii informatiky, ale také jednak díky převážně technickému charakteru tohoto odvětví. Bezprostřední aplikace v praxi zde přeci jen působí větším inovačním tlakem, než jaký je přítomen v disciplínách zaměřených akademičtěji. Navzdory zmíněným faktorům se ale zaostávání výuky za současným stavem odvětví projevuje i v programování. Rád bych nyní poukázal na některá problémová místa.

Prvním a asi nejviditelnějším místem je volba programovacího jazyka a odpovídajícího vývojového prostředí. Velké množství úvodních kursů programování na českých školách (včetně těch renomovaných) stále probíhá v Pascalu na legendárním, ale DOSovském vývojovém prostředí Turbo/Borland/Free Pascal. Samozřejmě, základní stavební prvky programu jako jsou proměnné, cykly, rozvětvení či podprogramy jsou víceméně stejné ve všech programovacích jazycích. Také se jistě dá souhlasit s názorem, že jakmile se jednou člověk naučí dobře programovat v jednom jazyce, snadno přejde na jiný. Na druhou stranu ale musím ze zkušeností z naší školy říct, že studenti to takto nevnímají. Když jsem u nás dříve učil Pascal, nesli těžce, že se mají učit něco, co se dnes používá již minimálně. Podobně archaicky působilo prostředí Turbo Pascalu. Z toho důvodu jsem ve svých kursech postupně přešel na programovací jazyk C# a vývojové prostředí *Microsoft Visual C# 2005/2008 Express Edition* dostupné zdarma z webových stránek producenta.

Druhým místem je otázka uživatelského rozhraní programů, které se studenti učí vytvářet. Přestože již celou řadu let má většina aplikačního programového vybavení uživatelské rozhraní grafické, probíhá úvodní výuka většinou na aplikacích s rozhraním textovým, čímž opět narážíme na otázku archaičnosti. Argumentem, který se ve prospěch textových rozhraní obvykle uvádí, bývá údajná jednoduchost ve srovnání s okénkovými programy. Myslím však, že v dnešní době tento argument již neplatí. S moderními vývojovými nástroji je tvorba pěkně vypadajícího okénkového programu stejně jednoduchá nebo obtížná jako tvorba konzolové aplikace pro „černou obrazovku“. Pro ilustraci, uvedu příklad výpisu jedné výstupní hodnoty uživateli:

```
Console.WriteLine("Výsledkem je {0}", výsledek); // textově
poleVýsledek.Text = výsledek.ToString();         // graficky
```

Celý rozdíl mezi oběma druhy uživatelského rozhraní se redukuje na jedno cvičení, ve kterém se studenti naučí ovládat nástroj pro návrh grafického uživatelského rozhraní. V mém kursu tedy již se začátečníky vytváříme okénkové aplikace.

Třetí otázkou, o které se chci zmínit, je seznamování se s objekty a objektovým programováním. Přestože z důvodů uvedených výše neinklinuji k přístupu „objects first“, je jistě objektové programování natolik důležité, aby se s ním studenti pozvolna seznamovali. Objektový jazyk C# a programy s grafickým uživatelským rozhraním k tomu poskytují pěknou příležitost. Nejprve studenti začnou objekty (převážně objekty grafické a objekty

uživatelského rozhraní) používat a až postupem času své vlastní objekty vytvářet. V mém kursu je tím časem zhruba polovina druhého semestru.

Čtvrtou otázkou je aplikační zaměření vytvářených programů. Klasický přístup je založen především na číselných a řetězcových aplikacích. Nejsem ale určitě sám, kdo by měl studenty, k jejímž koníčkům čísla a matematika nepatří. Samozřejmě, dříve bývaly jiné druhy aplikací problematické z důvodu obtížnosti, dnes však tento důvod pominul. Moderní vývojové platformy obsahují mocné a i pro začátečníka snadno použitelné knihovny tříd, které za pomocí jednoho, dvou příkazů umožňují manipulovat s obrázky, zvuky, tabulkami, animacemi atd. Uvádím příklad zobrazení rastrového obrázku v jazyce C# na platformě .NET:

```
Image obrázek = Image.FromFile("panáček.png");
kreslicíPlocha.DrawImage(obrázek, x, y);
```

PRAKTICKÉ ZAMĚŘENÍ

Vysokoškolská i středoškolská výuka často trpí přemírou teorie. Studenti jsou seznamováni s obrovským množstvím poznatků, především však formou výkladu vyučujícího nebo učebnice. Praktickému seznamování s problematikou toho kterého předmětu bývá věnována nepoměrně menší pozornost, ať už je důvodem větší náročnost přípravy nebo cokoli jiného. Následkem toho studenti často umí vcelku spolehlivě odříkat přednesenou látku, máloco z ní však dovedou prakticky použít.

Z důvodů již diskutovaných se tato situace týká výuky programování v podstatně menší míře, než je tomu u „akademických“ předmětů. Přesto nepovažuji za zbytečné uvést, že praktické zaměření kursu programování je něco, na co by vyučující neměl ani na chvíli zapomenout. V programování je dvojnásob důležité umět vyřešit konkrétní problém, umět naprogramovat konkrétní věc. Ve svých kursech aplikuji tuto myšlenku ve dvou rovinách.

V první řadě se na přednáškách snažím veškeré nové věci ukazovat ne na umělých, ale na aplikačně zajímavých příkladech. Když vysvětluji novou věc, dejme tomu větvení programu, tak se neomezím na výklad, jak se zapisuje `if`, případně na nakreslení vývojového diagramu. Naopak se vždy pokouším najít několik problémů zajímavých jako aplikace, ty postupně rozebírám a rozbory zakončuji řešeními, jež využívají příslušnou novou věc, např. zmíněné větvení. Studenti tak mohou vidět nejen, že „existuje nějaký `if`“, ale především, k čemu je dobrý, kde se dá použít a jakým způsobem.

Ve druhé rovině přednášky doplňuji velkým množstvím úloh k procvičení. Úlohy jsou navrženy tak, aby k jejich řešení vždy postačovaly postupy a techniky vysvětlené od začátku roku až po danou přednášku. Úlohy slouží jednak pro práci na cvičeních, jednak pro studium doma. Pokud jde o cvičení, tak se studenty společně vyřešíme jednu nebo dvě úlohy a dále již pracují samostatně, každý svým tempem. Já pak jen sleduji, jak jim to jde, případně poradím a odpovídám na dotazy.

Důraz na schopnost konkrétní věc naprogramovat na závěr kladu i při zkoušení. Po každém semestru studenti na zkoušce dostanou tři úlohy, které mají u počítače naprogramovat. Jen tak se pozná, jestli student opravdu něco umí nebo ne. Pozná se to mnohem spolehlivěji, než u lecjaké ústní zkoušky, kde povídaví studenti umějí z nulových znalostí „uvařit“ aspoň trojku.

STUDIJNÍ MATERIÁLY

Pro kvalitní vysokoškolskou výuku (středoškolskou ze zkušenosti posoudit nemohu) jakéhokoli předmětu považuji za nezbytné, aby studenti měli k dispozici vhodné studijní materiály. Něco, k čemu se mohou doma obrátit a v klidu studovat. Případy, kdy jsou studenti odkázáni pouze na zápisky z přednášek, rozhodně nejsou následováníhodné. Vysokoškolské studium se vyznačuje větší volností, tak proč by měli být studenti nuceni chodit na každou přednášku a na ní při snaze věci porozumět ještě vše rychle zapisovat? A to nemluvím o studentech dálkového studia, kteří mají výuku ještě podstatně stručnější.

Při hledání vhodné literatury pro svůj kurs jsem ovšem narazil na několik zádrhelů. Největším z nich byla opravdová dostupnost začátečníkům. Přestože mnohé knihy dostupnost začátečníkům deklarují, při bližším prozkoumání jsem zjistil, že tempo výkladu je dosti vysoké a pro porozumění je předchozí programátorská zkušenost potřebná.

Navíc jsem chtěl, aby literatura alespoň částečně vyhovovala výše diskutovaným kritériím aktuálnosti a praktického zaměření, což se také ukázalo určitým způsobem problematické. Řada knih na trhu je vyloženě zaměřena na některý programovací jazyk, na všechny jeho detaily, podstatné i méně podstatné. Nové věci právě ukazuje tak trochu zmíněným stylem „toto je `if` a konec“.

Zkrátka přestože počítačové a speciálně programátorské literatury bývá v knihkupectví spousta, nakonec člověk zjistí, že jí je málo! A ještě méně je původní české produkce. Ono se ostatně není co divit. Připravit solidní knihu je strašná spousta práce a autorské honoráře na českém trhu nejsou nijak závratné, bohužel.

Za dané situace jsem tedy otázku studijních materiálů řešil napsáním svých vlastních. Ty mají v současnosti podobu učebnice [2] a sbírky úloh [3] *Moderní programování* dostupných na celostátním trhu. Bližší informace o těchto knihách včetně ukázkových kapitol najdete na webových stránkách `www.moderniProgramovani.cz`

Zmíněné knihy používám v prvním semestru svého kursu a v nejbližší době připravuji jejich pokračování pro středně pokročilé čtenáře. Samostatný svazek sbírky úloh má zdůraznit důležitost studentovy vlastní snahy něco naprogramovat.

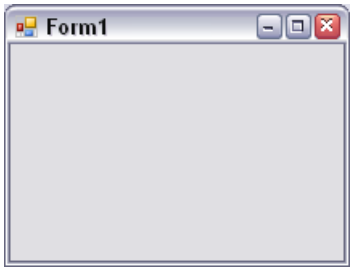
Úlohy ve sbírce jsou členěny do pěti kategorií: jednoduché, aplikační, technické a rozšiřující úlohy a kontrolní otázky. Jednoduchými úlohami by měl čtenář (student) práci nad danou kapitolou začít. Aplikační úlohy bývají již o něco složitější a jsou charakteristické silným aplikačním aspektem (programuje se něco zajímavého). Technické úlohy slouží k prozkoumání nějakého detailu bez ohledu na aplikační aspekt úlohy. Rozšiřující úlohy tvoří jakýsi „nadstandard“ pro studenty, kteří jsou oproti zbytku studijní skupiny napřed a nechtějí se nudit. A konečně kontrolní otázky nevedou na sestavení programu, spíše nějakým způsobem testují čtenářovo porozumění.

Ke všem úlohám jsou k dispozici nápovědy, po nichž může čtenář sáhnout, když si s úlohou hned neví rady. K většině úloh jsou také k dispozici hotová řešení pro ty, kteří si se zadáním neporadí ani s pomocí nápovědy. I studiem hotových řešení se může čtenář leccos naučit.

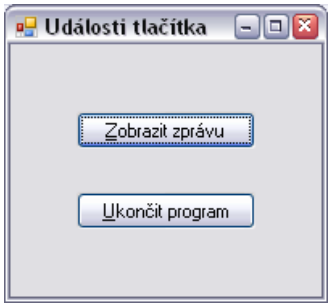
UKÁZKY PROGRAMŮ

V této závěrečné části příspěvku ukazuji některé programy z kursu. Časově všechny spadají do prvního semestru.

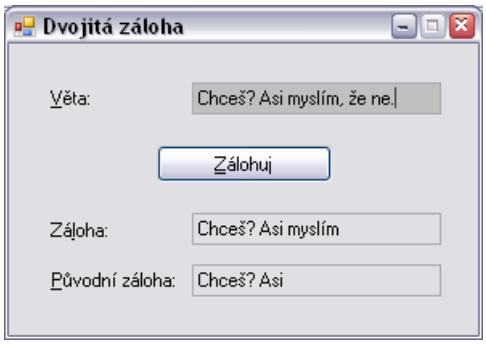
Úvodní program, kterým kurs začíná. Studenti de facto spustí připravenou funkční šablonu okénkové aplikace.



Program, který již něco dělá. U tlačítek je obsluhován jednak jejich stisk, jednak zastavení myši.

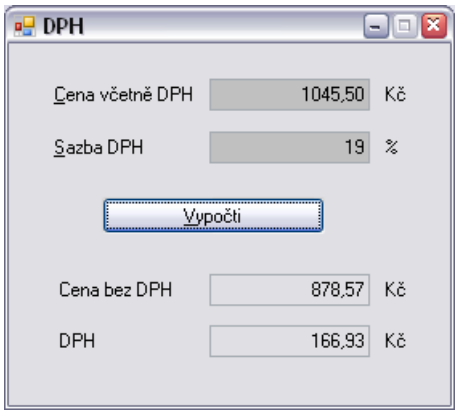


Program, na kterém ukazuji sekvenční zpracování. V tomto případě záleží na pořadí vykonávání příkazů.

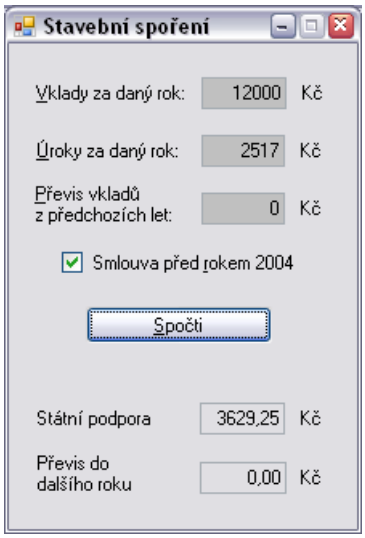


Program s pohybujícím se panáčkem využívá členské proměnné (souřadnice panáčka), vykreslování rastrového obrázku i data vestavěná do .exe souboru (obrázek).

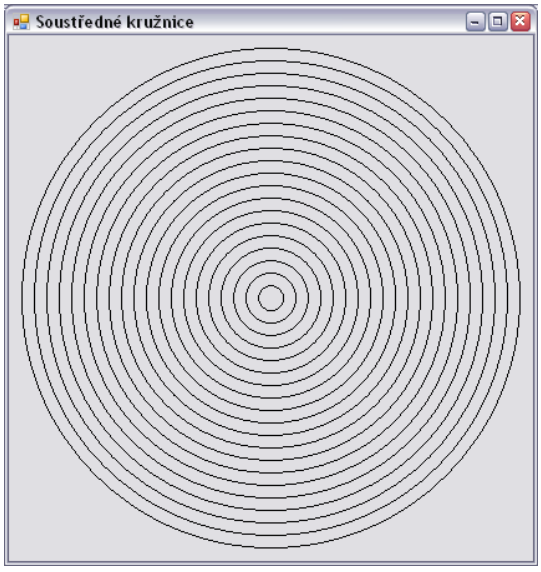
Úplně bez výpočtů to nejde. Když už tu ale jsou, tak praktické. Minikalkulačka daně z přidané hodnoty.



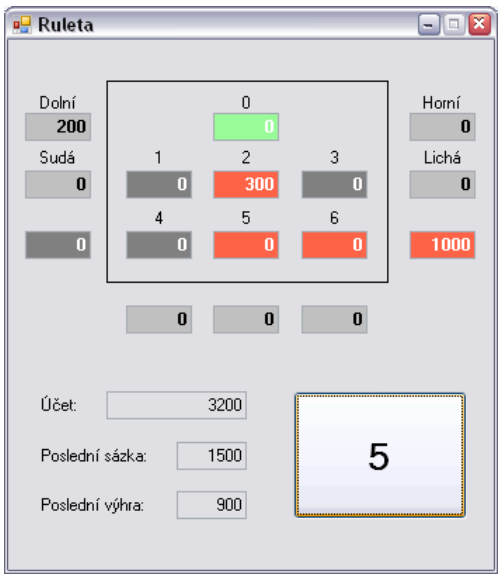
Anebo stavební spoření. Krásná aplikace na četná opakovaná rozvětvení. Ukázka, že nejprve je třeba porozumět dané problematice, potom navrhovat program a až nakonec zapisovat kód



Proč problematiku cyklu neilustrovat na otázce kreslení pravidelného obrazce?



Jedna ze závěrečných aplikací – zjednodušená verze sázkové hry.



LITERATURA

[1] The Joint Task Force on Computing Curricula: *Computing Curricula 2001*, Final Report. [on-line] `http://www.sigcse.org/cc2001/`

[2] VYSTAVĚL, R. *Moderní programování – učebnice pro začátečníky*. moderníProgramování s.r.o., 2007. ISBN 978-80-903951-0-7.

[3] VYSTAVĚL, R. *Moderní programování – sbírka úloh k učebnici pro začátečníky*. moderníProgramování s.r.o., 2007. ISBN 978-80-903951-0-7.

[4] HOLT, J. *Proč děti neprospívají*. Volary: Stehlík, 2003. ISBN 80-902707-6-X.