VÝCHOVA PROGRAMÁTORA PRO PRÁCI V DATABANKOVÉM PROSTŘEDÍ

RNDr. Zbyněk Štěpánek

1. Úvodem zamyšlení o programování jako vědě a programátorovi jako člověku

S rychlým rozvojem výpočetní techniky je nutně spjat neméně rychlý vývoj v oblasti programování. Programátorem se člověk stává v okamžiku, kdy skládá posloupnosti příkazů pro nějaký procesor, a to ne posloupnosti náhodné, ale takové, které formulují jistý algoritmus. Zavádění počítačů do běžného života působí, že programováním se zabývá stále větší okruh lidí. Programování se ostatně dostalo do učebních osnov většiny středních a vysokých škol. Znalost programování se tak podobně jako znalost matematiky stává obecným jevem. Tím, že v budoucnu bude programovat značně široký okruh lidí, nezanikne programování jako profese. V dalším se budeme zabývat programováním z hlediska programátora - odborníka. Jako každý odborník, musí i programátor svůj obor ovládnout a nadále se v něm vzdělávat, musí být veden a vychováván k tomu, aby dosahoval kvalit znalce řemesla - a v případě programování i umění. S rozvojem programování se mění nároky na programátora, mění se požadavky na jeho znalosti, mění se jeho postavení v procesu tvorby programu, mění se jeho vztahy k okolí. Po období počátků programování, charakterizovaném tím, že programování mělo jisté prvky svobodného řemesla, nastalo období zavádění řádu, předpisů a omezení s cílem vytvářet efektivní, přesné a přehledné programy. Období programátorské volnosti vystřídalo období pořádku, pravidel a norem pro programátorskou činnost. Výchova programátora se nesoustředila pouze na popis programovacích prostředků, ale položila důraz na efektivní používání těchto prostředků. Byla to výchova jistěm úspěšná. Všeobecné rozšíření několika vyšších programovacích jazyků, rozsáhlé využívání struktrálního, modulárního a parametrického programování může budit dojem, že programátora nečekají v současnosti ani v budoucnu žádné převratné změny, že se bude pouze zdokonalovat v navyklých postupech. V dalším ukážeme, že tomu tak být nemusí. Již například uplatnění parametrických programů znamená odsunutí programátora z některých dříve držených pozic. Perspektiva

programátora však není v tom, že místo formulace algoritmů bude pouze zapisovat vstupní parametry obecných parametrických programů. Tím by se dostal do role robota. Úkolem programátora bude i nadále především tvorba programů ve vyšších programovacích jazycích. Tyto programy budou stále častěji součástí vyšších informačních systémů. Programátor tak musí znát stále více souvislostí, musí vědět "pro koho" programuje a jaký má tato skutečnost dopad na použité programovací techniky. Jako příklad může sloužit přechod programátora k databankovému systému, do databankového prostředí. Z vnějšího pohledu přitom k podstatným změnám ve způsobu programování nedochází. Nové jsou ale souvislosti a vztahy v programátorském okolí, nové je postavení uživatele a podobně.

Výchovou programátora rozumíme dlouhodobý výukový proces, během kterého si pracovník osvojuje používání jistých programovacích prostředků a technik a tak je schopen vytvářet programy, které na jedné straně splňují funkční a další požadavky na ně kladené, na druhé straně vykazují jistý osobitý způsob řešení algoritmu s pomocí prostředků programovacího jazyka. Výchovu programátora rozlišujeme na výchovu

- jazykovou
- problémovou
- systémovou
- matematicko-filozofickou
- psychologickou.

Od programování je neoddělitelné zvládnutí programovacího jazyka. Je třeba si uvědomit, že i když programátor ovládá více programovacích jazyků, jeden z nich preferuje a v něm "myslí". Obvykle to bývá první zvládnutý jazyk. Programátor tak neuvědoměle přenáší do své další programátorské činnosti přednosti, ale i nedostatky jazyka, přes který se vžil do role programátora. Na znalost programovacího jazyka musí bezprostředně navazovat schopnost tohoto jazyka účinně využívat. Problémová výchova zahrnuje širokou škálu úloh od optimalizace zápisů různých algoritmů přes programovací techniky až k metodám ověřování správnosti a korektnosti programu. Zatímco popis programovacího jazyka je proveden vždy v přesných, téměř matematických formulacích (jinak ani nelze syntaxi jazyka

popsat), zapomíná se na obdobnou přesnost při propagování nových programovacích technik mezi programátorskou veřejností. Základní pojmy bývají definovány obsáhlým slovním popisem bez přísného induktivního nebo deduktivního popisu a s řadou nepřesností. Výsledkem jsou diskuse programátorů, zda daný program je nebo není strukturovaný a kde vlastně začíná banka dat. Teorie programování se při výchově programátorů opomíjí z obavy, že začínající programátoři budou přílišným teoretizováním odrazováni od náklonosti k programování a zkušení programátoři budou obtěžováni zbytečnostmi. Tím vzniká nebezpečí, že dojde k nepochopení záměrů autorů teoretických statí o programování ze strany větší - té praktické - části programátorů. Programátoři se často spokojí s nepřesným výkladem základních entit a s neúplným zvládnutím nových metod. Žádné nepříjemné důsledky přitom nepociťují. V databankovém prostředí mohou být nepříjemné důsledky této skutečnosti závažnější.

Programátor pracuje většinou na programech pro konkrétní počítač. Musí tedy znát některé parametry fyzického počítače, musí být zasvěcen do potřebných stránek operačního systému, musí vědět, zda pracuje pro některou část databankového systému. Systémová výchova programátora zahrnuje operační systém, systém řízení báze dat, případně další nadřazené informační systémy, pod kterými programátor pracuje. Matematická výchova neznamená v žádném případě požadovat od programátorů podrobné znalosti matematických výpočetních metod (pokud je ovšem pro konkrétní programy nepotřebují). Spíše jde o stránku matematicko-filozofickou a o "matematický" přístup k programování.

Společný postup matematika a programátora spočívá
- v možnosti definovat vlastní vědu axiomaticky
- v přesném definování pojmů
- ve vytváření stále obecnějších a širších teoretických základů
- v zaměření teorie na praktické aplikace
- ve zpětném promítání zkušeností a potřeb do praxe do teoretické roviny.

Vedle obsahové a metodické stránky jsou matematika a programování podobné svým celkovým zaměřením, svým "smyslem". Programátor

pohlíží na realitu (kterou má naprogramovat) a na programové prostředky (kterými má programovat) podobně jako matematik na přírodu, která je mu realitou a na matematický aparát, který mu je jazykem k popisu a zachycení přírodních procesů. Každý programátor je tak vlastně matematikem, a to jak v galileovském smyslu, když si uvědomí, že programování je právě ta jediná činnost, která umožňuje přenést realitu do datového modelu v počítači, tak ve smyslu archimédovském a vyslovit parafrázi: Dejte mi několik instrukcí a já naprogramuji, jak pohnout Zemí.

Jednotlivé druhy výchovy programátora nejsou samozřejmě od sebe odděleny, ale neustále se prolínají. Ucelený názor na výchovu programátora nebyl dosud vytvořen. O některé aspekty této výchovy v databankovém prostředí se v dalším pokusíme.

2. Pokračování úvah v databankovém prostředí

Databankový systém (DBS) je vytvořen z několika základních komponent: systému řízení báze dat (SŘBD), báze dat, uživatelského propojení, výstupního procesoru a metainformačního systému pro daný SŘBD (METIS). Dále k DBS patří uživatelé - koncoví, intermediální, parametričtí, uživatelé - programátoři apod. Mezi jednotlivými složkami DBS a mezi DBS a jednotlivými uživateli existují vzájemné vztahy. Říkáme, že programátor pracuje v databankovém prostředí, jestliže tvoří programy, které se stávají součástí některé komponenty DBS nebo počítají s uživatelem tohoto systému. SŘBD je programový produkt, který je dodáván výrobcem v konečné podobě. Programování přímo pro SŘBD není tedy obvyklý. Jediný častý případ vzniku databankového prostředí je aplikace určitého SŘBD v aplikační oblasti (výrobním podniku, školství apod.). Těžiště programátorské práce je pak v tvorbě aplikačních programů pro práci s bází dat, pro komunikaci uživatele se SŘBD (dialogy), pro výstupní sestavy, případně pro práci s adresářem dat.

Programátor je v databankovém prostředí ve větší závislosti na nadřazených systémech. Je-li SŘBD aplikován nevhodně, není-li využito specifických předností banky dat a jedná-li se pouze o náhradu klasických souborů soubory databankovými, pracuje SŘBD neefektivně, i když jsou aplikační programy přehledné, jednoduché a úsporné.

Na druhé straně může programátor značně ovlivnit efektivnost celého DBS. Dobrý program provede aktualizaci báze dat během několika desítek minut, špatný program provede tutéž aktualizaci za několik hodin.

Práce na aplikaci SŘBD a tím i výchova programátora pro databankové prostředí je rozdělena do dvou fází. V první fázi - přípravné - se provádí analýza požadavků uživatelů a většina projekčních prací (návrhy struktury báze dat). Je to období přípravy programátora na vlastní práci pro DBS. Druhá fáze aplikace SŘBD začíná jeho instalací pod daným operačním systémem a po implementaci návrhu struktury báze dat pokračuje programováním a laděním aplikačních programů, které jsou těžištěm programátorské práce pro banku dat.

Přechod do databankového prostředí znamená pro programátora nezbytně rozšíření dosavadních znalostí, pochopení nových pojmů a vztahů i nový vztah k okolí. Používání vyššího programovacího jazyka se přechodem k bance dat zpravidla nemění. Pro implementovaný SŘBD se vybere (z nabízených možností) ten hostitelský jazyk, jehož používání ve výpočetním středisku převažuje. Dochází samozřejmě k rozšíření jazyka o příkazy spojené s databázovými funkcemi. Praktické zkušenosti ukazují, že zvládnutí tohoto rozšíření jazyka není spojeno s obtížemi. V databankovém prostředí je třeba výchovu programátora soustředit na několik rozhodujících momentů, které dále ovlivňují jeho vztah k práci i k okolí. Nové nároky databankového prostředí se týkají

- rozšíření hostitelského jazyka
- jazyka pro popis dat a jazyka pro manipulaci s daty
- struktury báze xdat nebo její části
- funkcí SŘBD (obslužné programy)
- slovníku a adresáře dat
- dalších komponent DBS (uživatelské propojení apod.)
- aplikační oblasti, pro kterou je DBS budován
- zaměření a účelu databankové technologie
- vztahu banky dat k "nedatabankovému" okolí.

Programátor musí zvládnout jazyk pro popis dat a jazyk pro manipulaci s daty. Nově je v databankovém systému řešen problém

nezávislosti programů a dat. Použití slovníku a adresáře dat umožňuje provádět popis dat důsledně mimo aplikační programy. Programátor však potřebuje znát struktura báze dat. Zpravidle mu není zpřístupněna celá, ale jen její část (tzv. subschemový pohled). V DBS se ochrana a utajení dat uplatňuje nejen k uživateli,ale i k programátorovi. Některé omezení - např. v použití databázových funkcí v programech - nejsou ani aplikačnímu programátorovi známa, protože jsou zavedena přímo na úrovni SŘBD. Programátor tato omezení nemusí pociťovat nepříjemně, je-li jeho pozornost zaměřena k hlavnímu úkolu: zvládnutí problematiky báze dat. Pro aplikační programátory je důležité

- přesné pochopení nových pojmů, které odrážejí nové entity (databázové věty, sety atd.).
- používání příkazů pro databázové operace (vlastně pohyb v bázi dat)
- ošetření chybových stavů báze dat.

Báze dat jako souhrn databázových souborů se od klasických souborů liší zásadně ve dvou ohledech. Jednak má báze dat čtyři rovnocenné složky - data (databázové věty) a vztahy mezi daty (sety). Dále je třeba neustále ošetřovat zásahy do integrity dat. Programátor musí za každý příkaz pro databázovou operaci bezprostředně zařadit ošetření možných výjimečných nebo chybových stavů. Zvládnutí programovacích prostředků pro práci s bází dat však ještě nezaručuje, že výsledné aplikační programy budou dostatečně efektivní. Programátor musí být vychován k tomu, aby náročnějších prostředků SŘBD používal úměrně a s ohledem na optimalizaci pohybu v bázi dat. Některé vymoženosti, například kontrola setříděnosti a duplicit v setříděných setech, prodlužují při neuváženém používání neúměrně dobu provádění databázové operace. Často se vyskytují případy, kdy programátor musí tutéž operaci naprogramovat několikrát - v závislosti na počtu výskytů vět nebo setů. Určitá cesta v bázi dat,která je v jednom případě optimální,může být velmi neefektivní,změní-li se počet výskytů některých vět nebo setů. Programátorovi nestačí proto znát způsoby ukládání dat do báze dat a zapojování dat do setů, ale musí vycházet z logického a fyzického návrhu struktury báze dat, kde jsou dány počty databázových vět a setů a jejich charakteristiky.

Nezávislost programů a dat se tak dostává do poněkud ironické roviny. Na jedné straně máme data v adresáři dat, na druhé straně nepředvídaná změna v počtu vět nebo setů databázového souboru vyvolává potřebu úprav v procedurální části aplikačních programů. Záležitost ještě komplikuje problém zpráv o chybových stavech báze dat nebo nenormálním ukončení programů. Z hlediska snadné orientace je pro programátory vhodné rozčlenění chybových zpráv do tří skupin:

- zprávy operačního systému o chybách mimo SŘBD
- zprávy SŘBD o chybách mimo SŘBD
- zprávy SŘBD o chybách uvnitř databankového systému.

Pro programátory, kteří začínají pracovat v databankovém prostředí je zpočátku neobvyklé, že chyby s příčinami mimo aplikační program (např. chyby JCL) jsou zachyceny SŘBD a projevují se jiným způsobem, než tomu bylo v "nedatabankovém" prostředí.

Specifická je problematika výchovy programátora pro práci v dalších komponentech DBS. Pro uživatelské propojení je například rozhodující vztah programátor - uživatel. Nejvíce práce čeká programátora při zajištění přímého přístupu uživatele do báze dat přes terminál. Nad aplikací SŘBD, která s tímto přímým přístupem neuvažuje, vždy visí otazník. Pro nedostatek prostoru se však těmito okolnostmi nebudeme zabývat.

3. Závěrem zamyšlení nad účelem a smyslem programování

Databankové prostředí nevyžaduje od programátora pouze matematickou přesnost, obsáhlé znalosti nebo schopnost hledat optimální varianty. To vše by mohlo být nedostačující. Na prvním místě musí být zaujetí pro banku dat jako takovou, přesvědčení o tom, že idea banky dat je správná. Dějiny lidstva a lidské práce (a tedy i dějiny programování) jsou uskutečňování idejí _ a který programátor by nechtěl vejít do dějin ?

Výchova programátora je vlastně jen konkretizace obecných ideálů člověka na oblast programování. Od starověku existuje kalokagathía - ideál souladu duševních a fyzických sil člověka.

Od středověku máme ideál člověka univerzálního - homo universalis a novověk přidal ideál gentlemana, který zákony vkusu a estetiky povyšuje na úroveň zákonů mravních. A co jiného požadujeme od programátora, než aby byl vyrovnanou osobností a tvořil stejně "vyrovnané" programy, aby se nebál univerzálnějšího rozhledu a nesetrvával u navyklých programátorských způsobů, aby výsledné programy byly nejen funkčně správné, ale i přehledné, srozumitelné a tím i estetické. Skutečnost však mívá do ideálu hodně daleko. Skutečnost ale ideál potřebuje. Skutečné a ne ideální je také databankové prostředí, ze kterého byly čerpány podněty pro tento příspěvek. Jedná se o oborovou úlohu "Aplikace SŘBD IDMS v podmínkách stavebního podniku", která je řešena ve výpočetním středisku Dopravních staveb Olomouc.