Didaktika programování a morálka vztahu mezi analytiky-programátory a okruhem uživatelů

Ing. Stanislav Hrabě, CSc., Energoprojekt Praha

Ve vyspělých státech se silným růstem vybavení počítači je zaznamenáván rostoucí nedostatek počtu programátorů, a to dokonce při silném růstu jejich přípravy na školách mnoha specializací. Je vysvětlitelné vlivem ekonomie velkosériové až masové produkce elektronických elementů i systémů, že přes všechny obtíže vývoje technologie i pestrosti sortimentu potřebných materiálů je posléze technické vybavení v dostatku. Pak se úzkým průřezem vývoje využití stává lidská schopnost tuto techniku náležitě ovládat. Nejde pouze o programátory, ale jde vůbec o lidi způsobilé mimo programování formulovat a připravit problematiku i prostředí k využívání možností kybernetiky. Ať se jakkoli usnadní programování i ovládání operačních systémů, ať jakkoli naroste nabídka speciálních strojů a programových knihoven pro konstrukci a projektování nebo pro využití ve výzkumu, formulace a příprava problému zůstává vždy člověku. Nadto využití nejrůznějších nabídek t. zv. automatizace projektování a příbuzných činností je rovnocenné svými nároky na kvalifikaci stejně jako vývoj analytiků programátorů. Zůstaňme proto "při zemi" a zamyslemme se nad tímto aktuálním problémem. Růst schopností analytiků-programátorů v kvantitě i v kvalitě je nyní rozhodující.

Uvažme jako vyhraněnou situaci pracoviště s novým strojem, s novým operačním systémem a případně s novým jazykem. Novost zde platí v lecčemž i pro již zkušené pracovníky s dřívější praxí, v programovacím jazyce pak zejména pro programátory s praxí v jiném jazyce a ovšem hlavně pro začátečníky, bez jejichž značné početnosti se rychlý vývoj neobejde. Tempo vyškolení v programování při náležitém ovládnutí operačního systému musí být zvýšeno proti dosud běžnému. To vyžaduje cílevědomý růst školitelů programování při uplatňování účinných didaktických postupů. Věci by se měly chopit organizace vědeckotechnické společnosti v rámci domů techniky, skupiny organizací VTS vhodně dohodnutých mezi podniky, jakož i příslušné útvary spolu s VTS pro svůj podnik či VHJ. Nároky, které naznačím

o změně vyučovacích metod, budou totiž sotva splnitelné v týdenních nebo čtrnáctidenních kurzech Kancelářských Strojů nebo jiných orgánů reprezentujících dodavatele. Tyto akce obvykle představují předávání více méně nedokonalých manuálů s více méně nedokonalým komentářem.

Budou-li si organizátoři i programátoři s pedagogickým zájmem klást vědomě požadavek vyvinutí účinných učebních a studijních postupů, mohou značně narůstat obecné zkušenosti tohoto druhu a dojde se k žádoucím výsledkům. Některé rysy účinných didaktických postupů mohou zde být naznačeny a posloužit jako využitelné podněty. Zdá se mi vhodné podle nemalé srovnávací zkušenosti seřadit znaky dobrých didaktických postupů takto:

Je nutno vyvíjet sbírku vhodných příkladů, z nichž lze sestavovat posloupnost k výkladu a k výcviku stoupající náročnosti a obtížnosti. Programování lze vyučovat pouze na nějakém předmětu zpracování na počítači, nikoli samo o sobě. Instrukce a příkazy musí být ovládnuty v působivých kombinacích při tvorbě "taktických" a "strategických" obratů programu k docílení účelných algoritmů řešení úlohy. Úlohy musí být natolik obsažné, aby předvedení těchto obratů umožňovaly. Nutno přiznat, že výběr úloh pro školení se tvoří lépe v oblasti matematických a vědeckotechnických výpočtů než pro zpracování hromadných dat. Příčina? Je v podstatě v možnosti nabídnout ukázky v rámci vyučovací hodiny či dvouhodiny, při čemž pro zpracování hromadných dat jsou reálné příklady tohoto rozměru obtížněji konstruovatelné, nemají-li být násilným zjednodušením naivně primitivní. Podstatným požadavkem je ukázat účinek kombinací příkazů, ale úloha atraktivní svou reálností je rovněž pro psychologii výuky velmi významná. Další princip v posloupnosti výkladu a úloh nazvu "postup ve spirále". To znamená začít od malého okruhu instrukcí a příkazů, se kterými lze předvést již zajímavou akci na počítači. Pak tento okruh vykládaných příkazů o nevelké přírustky zvětšovat, aby se opět ukázalo další využití buď v úplných programech, nebo v názorných programových paséžích. To splňuje známou zákonitost, že nejsilnější je paměť aktivní a neúmyslná, pojmy a pravidla takto poznávaná se zapamatují snadno a trvale. Na třetím místě uvedu dostatečné střídání teoretického výkladu s akcemi na počítači. To ovšem závisí na konkrétních hmotných a organizačních podmínkách, ve kterých se musí vyučovat. Čím delší musí být teore-

tická pasáž, než se dostaneme k počítači, tím kvalitnější by měla být její názornost. A nyní velmi důležitý princip, jehož prosazení míří proti značně rozšířenému zlozvyku mechanického probírání manuálů! Ve všech partiích výuky programování musí být vykládána vždy pohromadě sémantika i syntaxe, nesmí být prováděno jejich nesmyslné odtrhování dle uspořádání manuálů nebo popisů jazyka. Zejména se nemá dopouštět předcházení syntaxe ve velkých pasážích bez sémantiky. Adept programování se musí dovídat, co udělá stroj, když je v programu něco napsáno, nebo co má napsat, chce-li donutit stroj, aby určitý výkon udělal. Jak se programy a příkazy píší gramaticky, pravopisně, syntakticky správně, to se musí vykládat současně při studiu funkčního významu instrukcí a příkazů. Velký a úplný přehled syntaxe může vstřebávat pouze ten, komu je funkce příkazů již jasná, jinak vnímá posluchač rozsáhlé pasáže syntaxe jako pravidla beze smyslu a nemůže nic správně pochopit a tím i zapamatovat.

Domnívám se dále, že málokdo si uvědomuje význam toho, že růst zdatnosti programátorů závisí dosti na tom, jak nabývají představy o fyzikálně technické podstatě realizace programovaných výkonů počítače. Nejde tu o žádné důkladné speciální studium hardware, jde tu jen o představivost zaměřenou na významné vztahy toho, co se programuje a jak to probíhá v počítači. Sem patří např. pochopení přenosů mezi operační pamětí a magnetickými nosiči, poznání významu programování bufferů, rozlišování režimů přímého přístupu k datovým souborům a pod. Sotva lze poznat v praxi zdatnou skupinu programátorů, jejíž členové by se živě z osobního zájmu nezajímali o různé technické podrobnosti počítačů, i kdyby její někteří členové měli původní průpravu např. z filozofické fakulty z oblasti logiky. K doplnění toho je též známo, že naopak absolventi technických specializací, dokonce zejména slaboproudu nebo elektroniky, stanou-li se později programátory, vyvíjejí se velmi rychle a na dobrou úroveň, a to i ve zpracování hromadných dat. Proto pokládejme včasné komentáře o činnosti hardware za dobrou část výuky programování.

Tolik zde k obecným principům účinné výuky. Nyní něco k obsahovému uspořádání osnov školení, aby bylo vyvinuto pochopení algoritmizace úloh pro počítače.

Ve studiu kteréhokoli jazyka je třeba ukazovat obecné principy

algoritmů programovaných v počítačích, které se prosazují přes rozmanitost vyjadřování v různých programovacích jazycích. To začíná především zásadou plnění příkazů strojem v pořadí jejich zápisu, z čehož jsou prováděny výjimky podle zvláštní skupiny příkazů řídících. Je třeba ukázat mohutný vzrůst rozmanitosti vlastností algoritmů, kterého dosahujeme tím, že disponujeme příkazy skoků a větvení, příkazy testovacími. Spěch na některé aplikace může vést k tomu, že brzy zařadíme speciální příkazy cyklů. Ve prospěch hlubšího pochopení je však dobré, když ve studiu předchází výcvik po poznání větvení se skoky "vpřed", vytvoření cyklů rozepsaných pomocí skoků "vzad" a potřebných "if". V této fázi mají být zvládnuty i cykly vložené, cykly určené počtem oběhů s mutnými čítači a podmíněnými skoky, jakož i stavba cyklů podle toho, zda cyklová řada příkazů probíhá aspoň jednou, možnost neproběhnutí ani jednou, jakož i větvení uvnitř cyklové řady příkazů. Mají se též ovládnout příčiny chyb "věčného" "zacyklování". Potom se mnohem lépe a hlouběji pochopí podstata speciálních příkazů cyklů v kterémkoli jazyce. Je třeba přesně rozumnět, jak v příkazech cyklu jsou skryty podmíněné skoky a přesná funkce podmínek se má umět rozepsat. V přiměřených dávkách s příkazy výkonnými se má studovat pestrá paleta datových útvarů, jejichž možnost deklarací v soudobých symbolických jazycích stoupá. Indexované proměnné číselné i nečíselné, pole, vektory, matice, věty-záznamy neboli struktury, a posléze i datové soubory. Zde platí opět při vyučování neodtrhovat deklarace dat od výkonných příkazů, které ukazují jejich použití. Typickým příkladem je souvislost cyklů s indexy jako cyklovou proměnou a deklarace polí a masivů. Není žádná škoda ve studiu, neučí-li se posluchač všechny deklarace všech datových útvarů najednou, z počátku lze udělat bez nároku na důkladnou znalost pouze předběžný přehled. Velmi výrazný přechod k partiím náročnějším tvoří ovládnutí podstaty vztahů - program volající a volané podprogramy, volání podprogramu a tělo podprogramu, podstata možností podprogramů - skok s "pamatováním" adresy návratu, způsoby přenosu dat mezi programem volajícím a podprogramem, obecné podprogramy a funkční podprogramy.

Při výuce příkazů vstupu a výstupu je nutné včas zařazovat tyto možnosti k řešení úloh, avšak postupně jednodušší formy postačující a ve dvou třech etapách ovládnout vstup a výstup v plné rozmanitosti. Pro vstahy sloves "read" a "write" a příkazů typu "format"

platí opět požadavek vyučování případů "format" a slovesa vstupu či výstupu pohromadě, a nikoli absolvování velkých a často vzdálených pasáží manuálů v celku, bez jasného vyjádření toho, jak každý "format" je k tomu, aby byl využit některým "read" nebo "write", a jak mají tyto páry správně spolu korespondovat.

V této stati nelze mnohé další požadavky racionální výuky rozepisovat, chci jen sdůraznit, že při cílevědomém úsilí bude mnoho účinného ve vyučování nalezeno a vyzkoušeno. K těmto otázkám stojí ještě za zmínku něco o způsobu zapracování programátorů začátečníků. Nejednou se stává, že programátor určený svou průpravou k tomu, aby se stal posléze samostatným programátorem a dokonce analytikem programátorem, je ze začátku a dosti dlouho zaměstnáván tím, že má dělat doplňky a změny k programům dřívějších řešitelů. Nic proti tomu, aby také touto prací získával kus své praxe. Ale zůstávat při tom je velmi špatné. Skutečná výchova budoucího samostatného programátora mu musí poskytovat problém, třeba zatím snadnější, ale zcela samostatně řešený od počátku jeho formulace až po úplné úspěšné odladění. Jen tak má tato výchova účinek. Že vedoucí pracoviště musí přitom uznávat zákonitost, že začátečník dělá programy zatím např. třikrát až pětkrát pomaleji, než je bude dělat po dvouleté nebo tříleté praxi, to už patří k prozíravosti vedení. Nechce-li někdo obětovat tyto zvýšené náklady na zapracování, dočká se zvýšení programátorské kapacity za velmi prodlouženou dobu!

Uzavírám první část z názvu statě: nekvalita výuky a výchovy programátorů zpomaluje růst jejich počtu i kvality, zhoršuje ovšem pak i výběr vhodných adeptů, což má být také součástí dobré organizace výuky, a vede k nekvalitnímu startu jistého počtu pracovišť s moderní výpočetní technikou. Takový nekvalitní start pak spoluutváří určité nepříznivé poměry mezi programátorským týmem a okruhem uživatelů. Příčin je jistě více, ale zaměřuji se právě na tuto velmi významnou. Programátorská práce je jistě duševně náročná a i nervově značně namáhavá. Je proto pochopitelné, že hrozba přetěžování programátorské kapacity uživatelskými požadavky se snáší velmi špatně. Vyjasnění výběru i množství úloh, ke kterým se programátorský tým má zavazovat, je věc velmi odpovědná. Jak se to však má objektivně řešit, když máme co činit s nevyspělostí uživatelů, ale zároveň s malou kvalitou programátorů. K nyní časté nevyspělosti uživatelů, kteří mají na svou neznalost počítačů nárok, se ještě vyjádřím. V na-

vazování kontaktu a dorozumění mezi programátory a uživateli mají vedoucí odpovědnost ti, kteří z profese se vyznají v počítačích. Na nich je přirozená povinnost zjednat nápravu v nedorozuměních a v nedostatku kontaktu. Ti musí posléze vychovávat uživatele ke schopnosti spolupracovat s programátorem.

Z toho je tedy jasno, že výběr a výchova programátorských týmů má povinnosti značné společenské odpovědnosti. Musí posléze směřovat k vytváření zdravé otiščitelnosti programátorů prokazovat vysoké možnosti počítačů. Není-li toto plněno, vytváří se nebezpečí opaku, nebezpečí utváření pohodlné "bariéry" kolem programátorské práce, nebezpečí zneužívání svého druhu "monopolu", který tvoří programátorská profese v jistém okruhu své působnosti.

Sledujme nejprve linii příznivého vývoje, čím se má vyznačovat působení zdatných a vynalézavých programátorů na uživatelský okruh. Programátoři musí se svými odbornými znalostmi nabývat zkušenosti ve styku s laickým zadavatelem - budoucím uživatelem. Od profesionálů musí být zadavatel poučován v praxi formulace úloh. Z tohoto praktického styku se musí zadavatel dovídat, co vyžaduje jasnost, jednoznačnost a bezrozpornost úmluvy o zadání úlohy mezi uživatelem a analytikem programátorem. Analytik programátor musí umět uplatňovat sám ze své iniciativy dostatečnou nebo spíše zvýšenou obecnost vlastností a použitelnosti programů, a to ve větší míře, než je to schopen laický zadavatel. Proti všem tvrzením zadavatele ve stylu "Potřebujeme jenom případy 1, 2, ... až 4 nebo 5" musí programátor uplatňovat podezření, že nastanou ještě případy jiné, které si zadavatel ze své praktické činnosti ani neuvědomuje. Programátor je povinen ze své profese objevit případné obecnější řešení, které by mohlo zahrnout i mnoho případů jednotlivě nepředvídatelných a včas je uživateli nabídnout a vysvětlit. Uživatel má být poučen programátorem o důležité zákonitosti ekonomie programování: program úzce specializovaný na několik málo přesně stanovených modifikací až stojí např. 100 jednotek pracnosti a nákladnosti. Bude-li sjednán program vysoce zobecněný, který zahrnuje širší třídu řešení, bude sice jeho pracnost a nákladnost např. 130 či 140 jednotek, zato však bude vyhovovat i pro případy dnes nepředvídané. Avšak "běda!" tvrdí-li se dnes, že "Potřebujeme zaručeně jen případ první až pátý", aby za čtvrt roku po zavedení došlo k požadavku verze šesté, pak

dodělávky a předělávky spolu s původní pracností a nákladem dosáhnou 180 nebo 200, jestli nepřesáhnou hodně dvěstě procent. Analytici programátoři jsou povinni provádět mezi zadavateli a budoucími uživateli "osvětu" mimo jiné o tom, jaké je riziko neodpovědných požadavků a změn programů při dodatečných nápadech uživatelů. Zdroj takových nepříjemností může být také zcela v nedostatečném zájmu vedoucích o problematiku využití počítače. Je jistě nemálo situací v našem průmyslu, kde vedoucí pracovníci v období příprav požadavků na počítač vše přenechávají výkonným podřízeným pracovníkům. Tito mohou spolu s programátory přípravu řešit svědomitě a s rozhledem, jasně, jednoznačně a bezrozporně. Program nebo soustava programů je hotova a uváděna do praxe. A tu vidí takoví vedoucí pracovníci první sestavy vycházející v dané problematice z počítače. A tu se teprve začnou v jejich hlavách rojit nápady na to a ono, že "to by se mělo vynechat" a "místo toho přidělat tohle" a tak a ta sestava že má vypadat trochu jinak. Nedomníváme se, že toto nebezpečí odstraňují tlustospisy na téma "Projekt systému ... atd". Těmto tlustospisům rozměrů 200 až 300 stran věnují vedoucí pracovníci ještě méně pozornosti než návrhům podstatně kratším a přehlednějším. A nadto některé z takových vznešených "Projektů informačních systémů" jsou pro skutečné provedení věci na počítači málo směrodatné, a to zvláště tenkrát, jestliže je zpracovali pracovníci nebo dokonce týmy osob nemajících skutečné znalosti o programování a využití počítačů.

Programátorský tým musí v celém svém prostředí vyvinout náležité poučení laiků o tom, že v programech jsou případné změny snadné s velikými přírustky výhod pro použití programu, ale že více hrozí změny neúnosné nákladnosti blížící se náročnosti novému vypracování programu. Dopouštět takové požadavky dokonce při tom, jde-li o změny malicherného významu pro uživatele, to znamená neodpovědné mrhání programovací kapacitou.

Dosud zde v této charakteristice jsem naznačil, jaká je problematika vztahu mezi programátorským týmem a okruhem uživatelů za předpokladu, že programátoři jsou odborně na výši a iniciativně se uplatňují ve spolupráci s uživately. Jak to však dopadá při převládnutí slabé úrovně nevalně připraveného programátorského kolektivu? V takovém kolektivu velmi předčasně vznikne pocit přetížitelnosti uživatelskými požadavky, a to bez ohledu na to, zda se uživatelé

chovají ve svém kontaktu s programátory odpovědně či méně odpovědně. Odborně chabý programátor není schopen požadavky uživatele správně posuzovat a tím méně je schopen uživatele ovlivňovat ve smyslu lepšího pochopení možností počítačů a ke zdokonalení účasti při formulaci úloh pro počítač. Příliš lákavá je představa "monopolního" postavení programátorské specializace v jistém okruhu její působnosti a tak se začne vyvíjet stereotypní odmítnutí požadavků nebeoprávněnějších bez ohledu na objektivní možnosti jejich realizace na počítači. Odborně slabý programátor začne odpovídat stereotypně "To nejde!" a uživatelský okruh je mu "pod úroveň", než aby se namáhal vysvětlovat, proč co nejde, nebo že něco na počítači "prozatím nejde" a vůbec vysvětlit uživatelům, co na počítačích a jak lze a nelze! V okolí takové "bariery" kolem programátorské práce, založené na mentalitě slabého týmu, vyjádřené představou "Bez nás to nikdo nenaprogramuje!" se začíná vytvářet pověst, jak ty počítače jsou k ničemu a jak všechno možné se nedá udělat, protože je to na počítači! V životě však nebývá často, že něco je jen "bílé" a něco je jen "černé", a tak bývají nahodile utvářené programátorské kolektivy složeny jak z lidí méně zdatných a špatně připravených, tak z lidí schopnějších a odborně iniciativních a vynalézavých. Kdyby vedení organizace bylo na výši k řízení rozvoje využití výpočetní techniky, ovlivnilo by kritalizaci ve prospěch rozvoje větších schopností kolektivu a posléze k dobrému uspokojování potřeb organizace v rostoucím využívání velkých počítačových investic. Když však v té nahodilé rovnováze odborně zdatných, iniciativních a vynalézavých a méně schopných a pohodlnějších dojde k opakovanému mrhání programátorskou prací vlivem nedbalosti, neznalosti a neodpovědnosti zadavatelů, je provokován v celém kolektivu "sebeobranný reflex", ti zdatnější jsou ve svém postoji uvnitř kolektivu poraženi a musejí se přidat k převládající mentalitě, a také navenek dávat proti požadavkům uživatelů především najevo, že "TO nejde!". Tak se mohou na nějakou dobu ustálit týmy programátorů zatížené zlozvykem "vodit uživatele za nos" při každé příležitosti projednávání plánu programátorských prací.

Tím jsem naznačil souvislost otázek kvality přípravy programátorů a toho, co jsem označil v názvu statě za morálku vztahu mezi programátory a uživately. Je jasné, jak zmatné je se společenského hlediska překonat špatné případy této morálky. Tvrdím, že slabá úroveň

vztahu mezi programátory a uživately je především způsobena slabinami ve vývoji programátorského kolektivu jako prvotní příčinou. Neznalosti a nedbalosti na straně uživatelů mohou vést pak jen ke zhoršování tohoto vztahu, protože "monopolistické" choutky slabých programátorů jsou více podněcovány jako sebeobranný reflex. Zvýšením kvality v přípravě programátorů, ve zdokonalování metodiky studia a vyučování bude dán základ k tomu, aby i odpovídající znalosti na straně uživatelů je přivedly k odpovědnějšímu přístupu ve formulaci požadavků.

Je nutno dospět posléze k takové spolupráci, ve které se rozvíjí upřímnost ve vyjadřování o možnostech počítačů se strany programátorů, v jejich vysvětleních o reálných obtížích ve splnitelnosti některých požadavků. Stereotypní odpověď "To nejde!" musí zmizet z těchto kontaktů. Programátor si musí být vědom toho, že kdykoli je nucen dát k něčemu negativní vyjádření, že spíše musí sdělit, proč něco prozatím nelze na počítači provádět, respektive za jakých podmínek co jde a co proč nejde! Kolektiv analytiků programátorů musí umět vychovávat a poučit uživatele,aby těmto vysvětlením rozuměli. Někdy jsou rozšířeny představy, jako by ve vztahu k počítačům se lidé dělili na ty, kdož jsou v tom odborníky a aktivně počítače ovládají a na ty, kdož jsou laiky. Nevěnuje se pozornost závažným mezistupňům znalostí počítačů, jaké přísluší okruhu zadavatelů jako budoucích uživatelů a dále jaké přísluší pracovníkům vedoucím. Uživatele lze školit rovněž ke znalostem potřebným právě z uživatelského hlediska. Možná někdo bude pokládat za přehnané, řeknu-li, že uživatel se může naučit spočítat potřebné rozsahy diskových datových souborů, nutných v problematice, která se pro jeho agendu má programovat, aniž se musí školit jako programátor. Toto i mnohé jiné může dosáhnout kultivace styku a spolupráce mezi programátorem a uživatelem. Začne-li takto stoupat úroveň porozumnění uživatelů pro výpočetní techniku, kterou mají využívat, pak ovšem se nedaří ani ledabylému zlozvyku na straně programátorů "To nejde!" a manýře "vodit uživatele za nos" musí být odzvoněno.

Domnívám se, že z toho, co bylo uvedeno, může být celý problém souvislosti zdokonalení didaktiky programování, výběru a přípravy programátorů a kvality spolupráce mezi programátory a okruhem uživatelů náležitě pochopen.

V celku problematiky jde o velké hodnoty v investicích do počítačů i o značné náklady lidské práce programátorů i jejich uživatelského okolí. Proto je naléhavé nedostatky kvality naznačeného druhu energicky odstraňovat.

Tato náprava se ovšem nemůže stát bez náležité účasti vedoucích pracovníků. Také pro ně platí, že mají mít znalosti nikoli programátorské, ale přesi jen značné znalosti o výpočetní technice a o předpokladech jejího využití. To se vztahuje na ředitele, náměstky ředitelů i na mnohé vedoucí odborů v různých úsecích organisací průmyslových i neprůmyslových. Jinak nemohou tito vedoucí ze svých pozic rozvoj využití výpočetní techniky ovlivňovat příznivě. A je v tomto ohledu situace příznivá? Někdo by snad řekl, že ředitelé, náměstci ředitelů a jiní vedoucí pracovníci mají již dnes takové ponětí o výpočetní technice, že nároky na vedení v tomto směru splňují. Pokládám to rozhodně za omyl. Totiž to, že ředitelé a ostatní vedoucí pracovníci mají o využití počítačů již náležité ponětí, je iluze, kterou v nich vytváří četba denního tisku, jakož hojnost poslechu o tom z rozhlasu a televize. Teprve kdyby byly pořádány kvalitní týdenní kurzy pro ředitele a ostatní vedoucí s exkurzemi s dobrým výkladem a předvedením zpracování úloh na počítačích, začala by se iluze o znalostech v tomto směru vytrácet a snad by se mnozí z nich začali na svůj vliv na předpoklady zvyšování využití výpočetní techniky dívat pozorněji.