Ing. Bohumil Ševčík
ČKD Praha op., Výpočetní centrum

VZDĚLÁVÁNÍ PROGRAMÁTORŮ SE ZAMĚŘENÍM NA ROZHODOVACÍ TABULKY

Profese programátor spracování dat je jednou z nových profesí, které vznikly až v poslední době s nástupem počítačů. A právě proto, že jde o profesi novou, je zde mnoho problémů, které nejsou zcela vyjasněny a je třeba je řešit, diskutovat o nich a hledat návrhy na jejich definitivní řešení.

V naší zemi se datuje nástup počítačů do sféry spracování dat do první poloviny šedesátých let, tedy do doby zhruba před 15 lety. Tehdy se počítače dostaly do rukou lidí nejrůznějších profesí jako matematiků, ekonomů, pracovníků strojně početních stanic, řídících pracovníků z výroby a pod. O nikom z nich nebylo možno říci, že jsou programátoři, a přece vytvářeli, a mnohdy velice úspěšně, programy a celé programové systémy. Tajemství jejich úspěchu tkvělo v tom, že šlo většinou o lidi mimořádně schopné a nadané, kteří se uměli mnohdy autodidaktickým způsobem vžít do nové profese a s neúnavnou pílí a houževnatostí v ní pracovat. Jejich příklad strhával ostatní a hned v počátcích začali vychovávat nové následovníky.

V tomto okamžiku se začal rodit další nový problém - výchova a výuka programátorů. Je zřejmé, že k počítačům, i když jde o velice složitou, náročnou a drahou techniku, nelze neustále přibírat lidi jen výborné a nejlepší. Takových lidí nebude nikdy dostatek v kterémkoli odvětví. Je třeba zapojit a pracovat s lidmi, kteří nejsou už nejlepší, avšak

je třeba je do nové profese uvést, naučit je potřebným znalostem a pracovním metodám, aby mohli pracovat produktivně na stále rostoucích úkolech zpracování dat v nejrůznějších podnicích a správních úřadech.

Je možno namítnout, že toto je úlohou školství. Avšak je naše školství schopno vychovat dostatečný počet odborníků ? Není a patrně dlouho nebude, neboť potřeby narůstají rychleji než možnosti. A není tomu jinak ani v jiných vyspělých státech. Vždycky bude třeba, aby si uživatelé počítačů vychovávali část programátorů sami. A je nutno říci, že někdy jsou takto vychovaní programátoři lepší než programátoři vychovaní ve školách. Je to patrně tím, že škola nevychovává pouze úzké specialisty pro obor programátor zpracování dat, ale absolventům dává širší profil (matematické výpočty, metody operační analýzy, statistika, operování počítačů) a nemalou úlohu zde bude hrát pravděpodobně ta okolnost, že v době, kdy se budoucí student rozhoduje pro tento obor, neví se ještě o jeho duševních vlastnostech tolik, aby bylo možno jej včas přeřadit na jiný studijní obor.

Je zde tedy problém - výchova programátorů pro obor zpracování dat přímo u uživatelů počítačů. A je to problém závažný, protože se jedná o celostátně velké množství pracovníků, kteří jsou každoročně do nové profese zaškolováni. Na příklad v našem Výpočetním centru ČKD PRAHA, kde pracuje v oddělení programování pro zpracování dat zhruba do 25 lidí, vyškolujeme každoročně 4-5 nových programátorů a pro celou VHJ ČKD PRAHA se každoročně pořádá kurs základů programování v trvání asi 5-6 týdnů, který navštěvuje okolo 25-30 posluchačů.

Problém by byl jednodušší, kdyby se nástup nových uchazečů o tuto profesi mohl dít k jednomu termínu v roce, např. k 1. srpnu nebo k 1. září. Avšak není tomu zdaleka tak. Sotva skončí kurs, hlásí se další uchazeči. Je tedy nutno zaškolovat individuálně a při tom kvalitně, aby programátor mohl po několika měsících aktivně programovat a aby vykazoval uspokojivou produktivitu práce.

Náplň profese

Při výuce a nácviku programátorů si musíme klást otázku - co má programátor nezbytně umět a vědět, aby byl programátorem ? Předpokládáme, že se jedná o nácvik uchazeče středoškoláka. Toto programátorské minimum je představováno těmito disciplinami:

a) základní vědomosti o počítači - jak počítač pracuje, způsob ukládání dat, činnost periferních zařízení, paměťová média, rychlost jednotlivých částí počítače, zobrazování dat aj.;
b) obecné řešení počítačových úloh bez ohledu na konkrétní programovací jazyk, simulace činnosti počítače;
c) programovací jazyk na základní úrovni;
d) operační systém a další nezbytný software v nejnutnější míře.

I když se všeobecně soudí, že toto je opravdové minimum, stává se často, že nácvik nového programátora se soustředí především na bod c), t.j. výuku programovacího jazyka. Jakmile nastávající programátor zvládne tento bod, je zařazen do pracovního procesu, jsa odsouzen do role kódovače. O tom co programuje nebo o metodě, jakou je daný problém řešen, nemá čas přemýšlet a ani nemůže, poněvadž se o žádných metodách řešení nestýkal, často mu není zkontrolovat výsledky, kterých se dopočítal. Časem, když hlouběji pronikl do tajů nového povolání, se pokouší o samostatné řešení slovně formulovaných problémů, které zapsal funkční mnohem výše postavený analytik. Úroveň těchto formulací bývá různá, o jednoznačnosti a přesnosti popisu problémů se dá hovořit jen zřídka.

Úroveň řešení ? Nízká, téměř výhradně provádění formou vývojového diagramu; odpovídá znalostem a zkušenostem programátora.

Metoda řešení ? Dělej jak umíš.

Racionálnost řešení ? Zpravidla nízká jak co do nákladů na odladění programu, tak i do nákladů na zpracování při rutinním používání.

Hlavní měřítko ? Program "chodí" a je včas hotov.

Čitelnost programu, předpoklady pro provádění změn ? Nízké nebo velmi nízké, protože se s nimi vůbec nepočítalo.

Důvod tohoto neutěšeného stavu ? Nedostatečná výchova programátora. Jak můžeme předpokládat, že by výsledky mohly být lepší, když jsme jej vlo nenaučili ? A guruové, kteří by to nějak "vytušili" se nerodí. I samotný Dijkstra, otec myšlenky efektivního programování s minimem příkazů GOTO, se ke své metodě propracovával řadu let. V jednom rozhovoru pro časopis Datamation uvádí zajímavé srovnání.

Ještě v dobách svých začátků řešil určitý problém a řešení tohoto problému mu trvalo dva měsíce. Později, asi po 15 letech, když sepisoval skriptum pro své studenty, si vzpomněl, že by se tento problém mohl popsat a řešil jej znovu. Vyřešil jej za 20 minut a byl překvapen sebou samým. Zkoumal příčiny tohoto úspěchu a došel k závěru, že po 15 letech programování má mnohem více zkušeností a znalostí o metodě řešení různých problémů.

Opravdu, zkušenosti nám září, a my jim děkujeme za pravdu, že pro efektivní programování a řešení úloh zpracování dat nestačí pouze zdravý selský rozum, znalost kreslení vývojových diagramů a znalost programovacího jazyka. Největší nedostatky jsou způsobeny právě tím, že se při výchově programátorů tak málo zabýváme metodikou řešení úloh.

Máme-li hledat vhodné srovnání pro tuto neutěšenou situaci v jiném oboru, napadá mě myšlenka, jak by asi vypadala silniční doprava, kdyby se výchova řidičů omezila pouze na zvládnutí nezbytné jízdy, pravidla silničního provozu by se nevydala a předpisy pravidel by se považovaly jakožto omezitost "intuice" a inteligence samotných řidičů.

Po celou dobu, co počítače slouží v oblasti zpracování dat sbírá celý svět zkušenosti a snaží se je zakotvit do nějakých pravidel. Tak se zrodilo již známé programování "bez GO-TO", modulární programování, strukturované programování, normované programování, rozhodovací tabulky, ale i různá pravidla přenesená z vědy o systémech, jako zásada minimalizace vazeb, rozkladu složitého systému na subsystémy aj.

Výchova programátorů

Jak tedy dosáhnout uspokojivého stavu ?

V předchozích odstavcích je ukázána rozpornost současného stavu ve výchově programátorů, kdy na jedné straně věda a teoretická pracoviště dosahují pozoruhodných výsledků a přinášejí nové metody a postupy do praktického programování a na druhé straně vidíme značně zaostávající "masové" programování. Odstranění tohoto rozporu možno dosáhnout jen systematickou výchovou programátorů.

Cílem tohoto výchovného procesu by mělo být efektivní zaškolení průměrného jedince středoškoláka na takovou úroveň vědomostí, aby dovedl samostatně řešit jednoduché úlohy nebo části složitých úloh, a realizovat je do podoby hotového programu. Při tom by provedení programu mělo odpovídat všem požadavkům standardizace a metodiky programování včetně náležité dokumentace programu.

Postup výchovy by tedy měl odpovídat vytčenému cíli. Osnovy kursů a zaškolovací plány by měly vedle výuky programovacím jazykům obsahovat také disciplinu zvanou např. metodika programování. Co by bylo náplní tohoto předmětu ? Bylo by to především teoretické řešení příkladů z oblasti zpracování hromadných dat. Programátor by měl získat především základy práce se soubory a měl by umět řešit typické úlohy ve zpracování dat. Přitom by měl nabýt takové schopnosti abstrakce, že by dovedl rozpoznávat jednotlivé typizované části programů. Snad nejlépe by bylo vychovávat programátory tak, aby se naučili většinu programů převést na formu normovaného programování, aby dovedli všechny činnosti daného programu přiřadit do určitého typového modulu a uvnitř tohoto modulu teprve řešit jednotlivé detaily.

Není to žádná novinka, ale již i u nás existují pracovní kolektivy, ve kterých je úroveň práce na takovém stupni, že vnější struktura programů, t.j. řešení hlavní sítě programů, je prováděna určitým unifikovaným nebo normovaným způsobem.

Dobrá metoda, při které se programátoři – začínající naučí pracovat se soubory je výklad a procvičení zcela elementární úlohy, např. sečíst určitou položku všech vět souboru a výsledek vytisknout. Jakmile je příklad pochopen a vyřešen, následuje tentýž příklad s malou změnou – vytisknout každou větu. Po té další změna – tisk hlaviček na každou stránku. Další změna – číslování stránek. Dále – výběr vět podle určitého kritéria. V dalším přidávání úkolů následuje výklad konstrukce klíčových polí třídicího souboru a úkol utvořit součet při změně jednoho klíče, dvou, tří a více klíčů. Další přirozený pokračující příklad spočívá v přidávání dalšího souboru na vstup. Samostatným problémem je aktualizace souborů a zvlášť je nutno procvičovat rovněž programy konverze a kontroly vstupních dat. Spojí-li se tato výuka současně s konkrétním použitím části programu pro hlavní síť, pak si programátoři hned od počátku vštěpují do hlavy celou řadu návyků a postupů, jichž se pak mohou ve své další práci držet.

Důležitou součástí výchovy programátorů je jejich další průběžné vzdělávání. V každém kolektivu by se mělo alespoň jednou za 2 – 4 týdny konat odborné školení, na němž by se přednesla určitá téma v rozsahu asi 2 hodin. Obsah tohoto školení by měl být vybrán pro diskusi.

Rovněž se ukazuje užitečné organizovat kursy programování pro pokročilé, na nichž by se vyložily složitější programovací techniky, výklad efektivnějších partií softwaru atd.

Velkou roli při výuce hraje i osobnost vedoucího programátora a jeho osobní postoj k tvůrčí práci. On může svým příkladem nebo doporučením trvale působit na zkvalitňování práce podřízených programátorů a tím učinit z výchovného procesu proces trvalý.

Rozhodovací tabulky (RT)

Jsou bezesporu uznávaným pracovním nástrojem při definici počítačových úloh i při jejich programování. Obecně je zdůrazňována jejich přednost před klasickým vývojovým diagramem, tam, kde jde o logicky poměrně složité rozhodování. V těchto případech jsou rozhodovací tabulky mnohem stručnější, čitelnější, umožňují se snáze kontrolovat co do správnosti, snadno se do nich promítají změny a nakonec při vhodných předpokladech jsou snadnější na programování.

Přednost rozhodovací tabulky před verbálním popisem v případě složitějšího rozhodování je nesporná. Málokdo je takový mistr slova, aby dovedl zcela jednoznačně popsat vyčerpávajícím způsobem logicky složitější rozhodovací problém, a když to svede, je popis mnohem hůře čitelný než rozhodovací tabulka.

Rozhodovací tabulky mají přednost rovněž v tom, že provádění změn v programu, popsaném pomocí rozhodovacích tabulek je mnohem snadnější.

Přestože metoda rozhodovacích tabulek je všeobecně uznávaná a ceněná, jsou přednosti rozhodovacích tabulek v programátorské praxi nedoceněny a málo využívány. Na schůzce předních programátorů z oboru rozhodovací tabulky pro přípravu semináře Novinky 79 bylo konstatováno, že prvotní příčinou, proč se rozhodovací tabulky v praxi nedostatečně využívají je jejich <u>neznalost</u> nebo <u>povrchní znalost</u>. To je závažné obvinění v době, kdy v češtině i slovenštině vyšla již celá řada publikací o rozhodovacích tabulkách, kdy jsou každému programátorovi dostupné podrobné učebnice o rozhodovacích tabulkách v mateřském jazyce.

Jak danou situaci řešit? Především apelovat na ty programátory, ale i analytiky, kteří rozhodovací tabulky znají a umějí efektivně používat, aby je používali všude, kde je to vhodné. Zde je nutno zdůraznit výchovnou úlohu analytiků a starších programátorů. Je užitečné, že <u>je-li řešený problém formulován pomocí RT, je i jeho řešení v programu provedeno pomocí RT.</u>

Dalo by se považovat za samozřejmé, že lidé z řad programátorů a analytiků, kteří mají nejvyšší kvalifikační zařazení, znají a aktivně používají RT v každodenní praxi. Avšak průzkumem v našem Výpočetním centru ČKD PRAHA i na jiných pracovištích bylo prokázáno, že tomu tak není. Že mnozí z těchto pracovníků metodu RT sice znají, ale dosud nepoužívají nebo že ji dokonce považují za hradu ve své práci. To jen potvrzuje oprávněnost konstatování, že v oboru RT panuje obecně dosti velká nevědomost.

Náprava této nedobré situace spočívá tedy především v odstranění hlavní příčiny t.j. nevědomosti nebo povrchní znalosti. Na všech pracovištích by se nad touto situací měli zamyslet a udělat opatření v těchto směrech:

- přimět vedoucí pracovních skupin k dalším používání RT;
- vypracovat vzorník řešených úloh, v němž by byly úlohy popsány verbálně a byly řešeny úlohy jednak klasickou metodou vývojových diagramů, jednak metodou RT;
- do kursů programátorů zařadit každopádně téma RT;
- kursy programování dělit jednak pro začátečníky, jednak pro pokročilé a v obou kursech vyžadovat aktivní používání RT.

Ujdeme-li se zjištění, že jednou formulovaný problém pomocí RT je dále řešen již výhradně metodou RT, jakýkoli autor, měli by to analytičtí, ale i ne vedoucích programátorech, jak oni sami, mají aktivně používat RT ve své praxi. Z vlastní zkušenosti mohu říci, že snadno jsem problém popsaný analytikem pomocí verbálně převedl do RT. Jednak jsem tím přesvědčil analytika, že RT je výstižnější, jednak jsem tím vyvolal následnou vnější programátorovu, že daný problém se věcí samozřejmostí řeší metodou RT. Další vzdělávací proces se tedy odehrává u jedince, který je cílem jako první aktivně použít RT při formulaci problému.

Z á v ě r

Největším nepřítelem pokroku je pohodlí a uspokojen s daným stavem věcí. Chceme-li s oblasti programování udělat opravdu profesi hodnou přívlastku pokroková, je nutno nebát se nejrůznějších překážek, prosazovat vše nové a pokrokové a přesvědčovat lidi o správnosti nových myšlenek. Jen tak se nám může podařit vychovat novou generaci programátorů, pro které bude samozřejmostí používání všech pokrokových metod, včetně RT. Jednou přijde doba, kdy bude řadovým programátorům divné, proč by se neměly RT používat. A my chceme, aby tato doba přišla co nejdříve.