P Ř E Ž I J E P R O G R A M Á T O R R O K 2 0 0 0 ?

Ing. Richard Bětr

Příspěvek se zabývá současným stavem a perspektivami programování; poukazuje na existující i předpokládané problémy a snaží se ukázat náměty k jejich řešení. Jádrem příspěvku je pokus o střízlivé a věcné hodnocení vývojových trendů v oblasti programování.

1. Úvod

"Tatínku, co je to programátor ? "
"To je ten velký knoflík u pračky, synu."

Počítače a jejich vybavení se prudce rozvíjejí a tempo vývoje nevypadá na to, že by se chtělo zmírnit. Obrovská podpora, kterou náš stát v posledních létech poskytuje mikroelektronice už vede k pozitivním výsledkům. Používání počítačů se rozšiřuje, počítače se zabydlely i tam, kde z nich dříve měli jen legraci. Zároveň se však tiše a zatím nepříliš nápadně rozrůstají nepříjemné a pro budoucnost i nebezpečné jevy.

Jako příklad si uveďme naprosté nepochopení počítačové problematiky sdělovacími prostředky /existují ovšem čestné výjimky a zasloužily by si zvláštní oslavný článek/. Počítače jsou široké veřejnosti prezentovány jako žertovná zařízení, způsobující trapné situace nebo zneužívaná šarlatány k pochybným účelům. Jindy zase je příslušný redaktor uveden počítačem v nekritický úžas a jme se barvitě popisovat vizi úspory stovek lidí, náhrada celé administrativy včetně ředitele "elektronkovým mozkem" atd. a přitom přisuzuje počítačům vlastnosti a schopnosti, které nikdy neměly a ani nechtějí mít. Vrcholem pak jsou situace, kdy na př. filmař vyrábí celovečerní dílo, založené na myšlence výběru partnerů počítačem; odmítne skutečný počítač jako nefotogenický a nechá si zkonstruovat ničemu nepodobné monstrum, obsluhované příslušně šíleným vědcem nevábného

zevnějšku. Jiné veledílo předvádí scénu, kdy nůžní milenci rukopisem vyplní děrné štítky, které pak vhazují do snímače v podobě poštovní schránky; nebo je dramatická zapletka založena na rozkošné scéně, ve které si geniální programátor náhle vzpomene, že jeho počítač vlastně zvládne multiprogramování a tím vyřeší tíživou situaci podniku.

Tyto skutečnosti nelze přecházet mávnutím ruky; zkušený profesionál ví dobře, s jakými problémy se potkává při zavádění nových systémů u takových uživatelů, kteří byli výše zmíněnými uměleckými efekty na výpočetní techniku "připraveni".

Avšak i sami programátoři jsou zmítáni neklidem a vášněmi. Sotva se programátorská profese jakž takž stabilizovala, již na ni útočí nové jazyky, racionalizační metody, grafika, dialogy, databanky a jiné podobné různé. Programátor slyší zvěsti, že pátou generaci už nebude nutno vůbec programovat a přitom ho obléhají desetiletí uličníci, kteří se dožadují pomoci při ladění svých programů.

Domnívám se, že otázky tohoto typu mají být na tak slavném semináři jako je PROGRAMOVÁNÍ PIC 99 probírány a diskutovány, neboť se dotýkají samé podstaty profese programátora. Proto byl napsán tento příspěvek.

2. Žhavá současnost

2.1 Problém "-WARE":

Pod pojmem "programátor" rozumíme každý něco jiného. Někdo trvá na striktním "programátor = kódovač", někdo zahrnuje pod tento pojem i analytiky, někdo žádá po programátorech i tvorbu organizačního zabezpečení a tak různě. Příjemné vzrušení vnesli do problému jazykově zatížení vědci, kterým se zalíbily pojmy "hardware", "software". Bystře odvodili "orgware" /organizační zabezpečení/, "lifeware" /kádrové zabezpečení/, "investware" /investice/ a další libozvučné názvy /na počouchlé náměty typu mašinkware, bundesware, seware, ahasware reagovali dosti nerudně/. Pak se rozštěpili na dva tábory: jeden hlásal, že programátor má na starosti všechen -ware, druhý pak přisuzoval každý -ware jinému okruhu pracovníků a tvořil funkce i jejich názvy. Vzniklé zmatky vedly k depresivnímu stavu: sémantika

pojmu "programátor" je nejasné. Autor soudí, že už sám tento fakt má v praxi neblahý vliv a doporučuje vyvinout úsilí o sjednocení významové náplně jak na odborné, tak i všeobecné úrovni.

Tato poznámka budiž považována za námět k diskusi veřejné i kuloární.

2.2 Továrny na programy:

Programování bývá považováno za umění, řemeslo nebo i seriovou výrobu. Je jisté, že některé programy jsou uměleckým skvostem, jiné perfektním řemeslným dílem a některé továrním výrobkem v nejlepším slova smyslu. Známe však také programy dekadentní, kýčovité a neivistické /někdy i surrealistické/, programy zfušované a programy zmetkové.

Tuto problematiku posuzujeme ze tří hledisek:

a/ Účel programu a jeho určení je hlavním vodítkem. Umělecky zpracovaný /pojednaný/ program evidence základních prostředků je nesmysl a měl by být trestný.

b/ Nutno přesně rozlišovat úroveň algoritmu a úroveň programu. Geniální, invencí nabitý algoritmus může být realizován řemeslně důkladným programem bez uměleckých programátorských opiček.

c/ Za kvalitní tovární výrobek není třeba se stydět. Seriový výrobek je levný, spolehlivý, snadno se k němu seženou náhradní díly; drobnými vkusnými doplňky ho můžeme i vylepšit /a pak už nevadí, že je třeba pomalejší nebo méně honosný než drahý zdlouhavě individuálně vyrobený unikát/.

Myšlenka továren na programy ve světě prorazila, je uznávána a využívána. U nás je často chápána jako ponižující a Programátorů /s velkým P/ nehodná. Je to chyba a společenská i ekonomická ztráta. Kdy už konečně u nás vznikne "software-house" /a jak se to bude česky jmenovat?/, který bude hmotně zainteresován na kvalitě, co nejširší použitelnosti /a tím i prodejnosti/ vyrobených programů?

2.3 Chiméry a utopie:

Ve svých příspěvcích na seminářích r. 1976, 1977 a 1982 jsem satiricky tepal zchytralá individua, která se přiživují

na rozvoji výpočetní techniky a vyznačují se tím, že

- v ústních i písemných projevech používají několika odposlechnutých odborně znějících pojmů, jimiž oslňují laiky a děsí odborníky /"portabilita software musí zajistit, aby indexsekvenční organizace v databankovém prostředí neomezovala interaktivní odezvu subsystému, zvláště pak s ohledem na realizaci alokační fáze"/
- vyžadují binární názory /ano - ne, dobře - špatně, pokrokové - zaostalé, .../ i řešení
- oplývají směly mi perspektivami
- vyhýbají se pořádné práci.

Tito "pracovníci" neblaze ovlivňují programátorskou činnost tím, že propagují různé neuskutečnitelné chiméry a utopie /a často vyžadují jejich realizaci/. Je tragedií programátora, podlehne-li ve slabé chvíli některé z falešných idejí a pokusí se ji realizovat; neúspěch bývá výrazný a je vždy připsán programátorovi.

Za účelem varování upozorním nyní na některá často se vyskytující nebezpečí:

a/ Automatizovaný systém řízení: původní užitečná a chytrá myšlenka ztroskotává na
- přebujelém metodickém členění /ASŘ, ASŘP, ASŘTP, ASŘSČ, OASŘ, .../ a jeho přenášením do systémových řešení
- snaze budovat obrovský systém najednou jako jeden celek /takže řešení trvá desítky let, náklady vzrůstají nade všechny meze, technické a organizační nároky přesahují možnosti společnosti XX. století atd./
- mylném názoru, že automatizovaný systém řízení musí sám řídit /že tedy na př. ředitel se bude dovídat od počítače, co má v jaké situaci dělat/
- myšlence, že "všechno musí být v počítači" i když se přesně neví proč

a podobně. Chiméra jednorázové realizace obrovských systémů bují v naší profesi jako nebezpečný plevel. Přitom řada pracovišť už dokázala, že tichou a nenápadnou prací, postupným řešením úloh a jejich propojováním lze dosáhnout cenných a společensky užitečných výsledků.

b/ Zajímavý je nápad, že "uživatel musí být pro výpočetní techniku převychován". Tento požadavek slýcháme i čteme každou chvíli a přitom

- všeobecná výchova občanů je na dosti špatné úrovni /viz úvod tohoto příspěvku/
- převychování konkrétního nastávajícího uživatele bývá prováděno dosti drsně /"koupili jsme si počítač, tak se na něm musí něco zpracovávat"/.

Autor prohlašuje /a je ochoten doložit příklady z praxe/, že nejlepší výchovou uživatele je vypracování takového systému, který je uživateli užitečný a příliš ho neobtěžuje /ideální je systém "uživatelsky přítulný" - viz Programování '84/. Jestliže uživatel zjistí, že mu počítač opravdu pomáhá, začne výpočetní techniku fandit a máme vyhráno. Při zaškolování uživatele je vždy nutno věnovat pozornost nejen vysvětlování "co musíme dělat", ale také "proč to musíme dělat". Je zajímavým poznatkem, že u špatných systémů není někdy ani autor systému schopen dát uspokojivou odpověď na druhou otázku.

c/ Chiméra typových řešení: Bylo by jistě ideální, kdyby pro typové úlohy /mzdy, MTZ, základní prostředky, .../ existovala typová řešení; postupující unifikace počítačových systémů by dokonce umožňovala tvorbu typových programů. Na některé úskalí "typovosti" jsem upozornil ve svém příspěvku v Havířově r. 1976; tvrzení, která jsem tenkrát uvedl, zůstávají v platnosti. Navíc se však ukázalo, že

- dobrý typový projekt může vypracovat pouze nezávislá organizace, která takovýto projekt bude zpracovávat jako "tržní zboží"; zřejmě je těžké řešit účelový projekt agendy pro vlastní potřebu a současně jej zobecňovat na úroveň typového řešení;
- zaměření na typová řešení celých agend není nejsprávnější; praxe ukazuje, že je účelné vynaložit úsilí na zpracování typových modulů, ze kterých pak lze různé úlohy skládat; tento poznatek je ve světě realizován, softwarové firmy dodávají velké množství typových modulů a programátoři je v nejširší míře využívají pro tvorbu účelových řešení; lze si jen přát, aby se tento způsob práce rozšířil i u nás /a v této souvislosti opět vyvstává otázka

"programového domu" nebo aspoň "software-cottage", zmíněná v bodu 2.2 tohoto příspěvku/; blíže se k této myšlence vrátím v bodu 3.2.

3. Reálné a nereálné perspektivy

3.1 Quo vadis computer science ?

Jak již bylo řečeno, prudký vývoj působí na řadu programátorů depresivně. Není to tak dávno, kdy jsme do 4 kilobytů museli nacpat program i data. Teď máme k dispozici megabyty; "klasický" programátor tím bývá kupodivu zaskočen. Kládež naopak bezuzdně řádí a na strojích pod 1 MB odmítá pracovat.

Assembler umí dnes jen systémový programátor. Na agendového programátora se však valí nové jazyky. Dodává se PASCAL, ADA, APL, APLGOL. Máme v nich psát agendy? Autor - vědom si všech důsledků - zaujímá zde dosti konzervativní stanovisko: s novými výmysly si můžeme hrát, zkoumat je a semtam si je vyzkoušet. Agendy však píšeme v COBOLu nebo FORTRANu. Proč ?

a/ Kdesi jsem slyšel výrok: "PASCAL je jazyk, který je velmi vhodný pro psaní kompilátoru jazyka PASCAL".

b/ V jazyku COBOL nebo FORTRAN napíšeme program pro jakoukoliv úlohu, která se může vyskytnout; proč bychom tedy používali jiný jazyk ?

c/ Nové jazyky skýtají řadu nových /dříve zcela nevídaných/ možností. Abych těchto vlastností dobře využil, musím být na výši jednak teoreticky /množiny, abstraktní struktury atd./ a jednak i co do znalosti fines jazyka. Jestliže nabízených /špičkových/ možností nevyužiji, degeneruje celý program a je méně efektivní než v "klasickém" jazyku. Průměrný šéf průměrně uvažuje: zvládnou vůbec má průměrná děvčata nabízené nadprůměrné možnosti? A za jakou cenu? Jaký bude efekt? Napíšeme to tedy v COBOLu, i když to nebude tak elegantní.

Beru na sebe obvinění ze zpátečnictví, ale tvrdím: žádný nový jazyk zatím nepřinesl takové výhody, aby je bylo nutno z hlediska průmyslové výroby programů brát vážně.

Pro práce umělecké a někdy i řemeslné můžeme ovšem nové jazyky s výhodou využít.

Musíme však
- zvolit jazyk pro danou úlohu vhodný /to předpokládá pochopení podstaty jazyka a zjištění jeho výhod/
- zvolený jazyk bezpečně ovládat v celém rozsahu
- efektivně využívat jeho výhod a rozdílů proti jiným jazykům
- neustále dbát o dobrou čitelnost a srozumitelnost programu.

U větších programátorských pracovišť je navíc nutno zajistit, aby s novým jazykem byla seznámena celá skupina programátorů s ohledem na vzájemnou zastupitelnost pracovníků při opravách a úpravách programů.

Trochu jiná je problematika t.zv. "racionalizace tvorby programů". V posledních létech se vyskytly řady metod, jak zaručeně správně napsat zaručeně správný program.
K tomu lze podotknout:
- Obecně vzato je racionalizace programování možná; na to přišli již staří Univaci a vymysleli normované programování, které se dodnes používá /pro určitý typ úloh!/.
- Nutno rozlišovat metody, které jsou zatím jen teoretické od prakticky využitelných postupů. Příklady: teorie ověřování správnosti programů je bohatá, pestrá a zajímavá, pro praktické použití se zatím příliš nehodí; teorie překladačů pokročila tak daleko, že z ní lze odvodit prakticky použitelné poučky - jestliže dříve byl kompilátor záležitostí několika člověkoroků, dnes může být zadán studentovi jako ročníková práce.
- Racionalizační metody jsou zaměřeny účelově /ač jejich autoři většinou tvrdí, že jsou zcela obecné/. Klíčovým problémem bývá vyhmátnout u každé metody její optimální oblast použití /normované programování nepoužijeme pro výpočet parametrů družice/.
- Jádro metody bývá obvykle velmi chytré, je však zpravidla obaleno různými parádami; pro praktickou aplikaci je důležité pochopit smysl a základní přínos metody.

Vždy je dobré si uvědomit, že neexistuje /a dlouho ještě existovat nebude/ metoda, která by sama napsala program, zatímco programátor by se oddával duchovním či světským požitkům.

3.2 Pomůcky:

Programátorské práci hodně pomáhají různé drobnosti, které se také rychle rozvíjejí. U nás bývají často podceňovány /někdy i odmítány/; nemáme také výrobce, který by se tvorbou a rozvojem pomůcek zabýval.

Začíná to třeba programátorskou brašnou. Ve světě jsou k dostání aktovky, do kterých lze snadno uložit blok, formuláře a testovací výpisy, dále nezbytné fixy, propisky a případně i šablonky. Do aktovky uložíme také programátorskou kalkulačku. To je specializovaný výrobek, který umí pracovat v desítkové, šestnáctkové, dvojkové a někdy i osmičkové číselné soustavě, provádět převody mezi soustavami a zvládne i výpočetní operace ve zvolené soustavě. Výpočetní logika je doplněna různými funkcemi, které programátor ve své praxi potřebuje.

Řeknete maličkost, snobárna, frajeřina; dovoluji si však tvrdit, že velice užitečná i prospěšná.

Pomůcky mohou být i rozsáhlejší. Na semináři v r. 1984 jsme slyšeli nadšeného Ing. Rusína /lit./1//, který si vyzkoušel interaktivní systém pro tvorbu a ladění programů. Mohu potvrdit, že programátorské práce s takovouto podporou nabývá zcela nových kvalit. Paradoxem je, že systémy tohoto typu jsou k dispozici na malých personálních počítačích, kdežto u velkých strojů je bolestně postrádáme. Vůbec je možno konstatovat, že programové vybavení osobních a domácích počítačů je "programátorsky přítulné", zatímco velké ponuré skříně se s námi moc nevybavují.

Velmi smutná je situace v oboru programových prefabrikátů a polotovarů. U nás je řadíme spíše k pomůckám, ve světě jsou však nezbytnou integrální součástí programátorské práce. Mám na mysli užitečné podprogramky /obecně pojaté a důkladně propracované/, ze kterých lze sestavovat i velké programové celky. Jde na př. o počítačovou grafiku, o dialogy a ovládání obrazovek, práce se strukturovanými soubory dat a podobně. Viděl jsem rozsáhlé programové systémy pro vědecký výzkum, které na př. dokázaly matematicky analyzovat pohyby zvířete, snímané televizní kamerou; výsledky byly v barevné grafice předváděny na displeji. Takový systém vypracoval jediný programátor v neuvěřitelně krátkém čase; jeho práce spočívala ve skládání hotových modulů, které vybral

z katalogu a nechal zakoupit v "prodejně software". Nám dnes chybí jak výroba takových modulů, tak i dobře organizovaný trh. Přitom různá velmi chytrá řešení na různých pracovištích existují, ale potřebovala by

- dotáhnout zobecnění
- vybavit perfektní dokumentací
- katalogizovat a být nabízena odpovídající formou.

V tomto směru byla kdysi na př. velice užitečná kniha /2/, ze které jsem často vybíral a skládal programové prvky a řešil tak rychle a efektivně různé specializované úlohy.

Můžete namítnout, že lze celkem snadno sehnat různé knihovny typu "matematika","statistika" a pod. Mám však na mysli trochu jiný styl, který snad vyplyne z odst. 3.3.

3.3 Šok ve vývoji kalkulaček:

O fenoménu kalkulaček by bylo možno napsat několik více či méně čtivých článků. Pokusme se však nyní o programátorský pohled:

První programovatelné kalkulačky srazily lidstvo k assembleru. Výrobci to věděli a snažili se ulehčit trudný život uživatelů přidáváním různých opiček a tvorbou "typových modulů", které bylo možno lacino či draho dokoupit ve formě magnetických štítků, výměnných kostiček ROM a podobně. Náhle však došlo ke zlomu: kabelkové a někdy i kapesní hračky se naučily BASIC. Tvrdím, že od tohoto okamžiku ztratily "klasické" programovatelné kalkulačky /HP 67, TI 58, ... / existenční oprávnění. Zavedením vyššího jazyka do kapesních prostředků získáváme totiž filozoficky, výchovně a mnohdy i technicky zcela kompatibilní řadu "od kalkulačky až po největší počítač". To je významný krok kupředu a mám takový dojem, že nebyl ještě plně doceněn.

Nedávno jsem však dostal do ruky popis jednoho z nejnovějších modelů kabelkového počítače; o neuvěřitelné technické stránce se nemíním šířit - zajímalo mě programové vybavení. Zůstal jsem šokován: v rozsáhlé paměti ROM je velký operační systém se spoustou prvků, ale žádný programovací jazyk! /Samozřejmě: dokoupíme-li si floppy diskovou jednotku - též kabelkovou na monočlánky - máme k dispozici všechno od BASICu až po PASCAL/. V základ-

ním operačním systému jsou šikovně složeny různé užitečné programové moduly, obalené dialogovým rámcem /dialogy zásadně typu "nabídka" - viz můj příspěvek na Programování '84/. Celé je to uděláno tak chytře, že "normální" uživatel nepotřebuje vůbec nic programovat. Bližším studiem jsem si ověřil, že celý tento systém je pouze promyšlenou a velmi rafinovanou kombinací modulů, běžně dodávaných špičkovými programátorskými firmami /a jsme zase u odst. 3.2/.

Tento příklad ilustruje některé moderní myšlenky v oblasti programování a ve svých důsledcích se netýká jen malých personálních počítačů. Uvedené principy již pronikají do větších, uživatelsky /i "agendově"/ orientovaných systémů.

3.4 SCI-FI aneb šestá generace:

Neustále čteme o tom, jak v tajných a tajemných vývojových pracovištích vzniká již nyní pátá a šestá generace počítačů, které budou přinejmenším myslet a především se budou samy programovat.

Zanikne tedy povolání programátora? Osobně si myslím, že to nebude tak zlé. Dnes čilá dívka napíše

```
01  PRACOVNIK.
  02 PRIJMENI PIC X/24/.
  02 JMENO PIC X/12/.
  02 TITUL PIC X/6/.
```

a je to. Před počítač šesté generace předstoupí mladý vědec /pro jeho profesi zatím není vymyšlen název/ a rozvine se dialog:

"Vypočti mzdy pro pracovníky podniku!"
"Charakterizuj pojem pracovníci."
"Pracovníci je množina prvků, jejímž opakovaným elementem je pracovník."
"Čím je množina pracovníci dána?"
"Seznamem pracovníků, který ti bude předložen."
"Čím je charakterizován element pracovník?"
"Má jméno, příjmení, titul, datum narození, adresu, číslo."
"Co je to mzda?"
.... mám pokračovat? A to jsme ještě s myslícím strojem nezačali probírat, jak bude vypadat výplatní páska!

4. Programování jako zálibu

4.1 Vznik programátora:

V historických dobách se programátory stávali pracovníci různých profesí, které věc zajímala a kteří našli zálibu v pocitu "vlády nad strojem". Programovat se učili sami, jen zřídka byli vysíláni na firemní školení /tam se naučili některým jazykům, ale na př. jak udělat agendu jim stejně nikdo nevysvětlil/. Na taje a záludy řemesla přicházeli sami nebo je odkoukávali od zběhlejších kolegů.

Dnes se programování učí na různých školách a měli bychom tedy mít dostatek kvalifikovaných programátorů. Vynořily se však problémy, na př.:

- Programování se vykládá stejně jako jiné předměty - většinou suše a nezáživně a hlavně za účelem zkoušení.
- Nebyl dořešen problém jazyka, s kterým by měla mládež začínat. Zkusilo se vyvinout nové jazyky pro potřeby výuky /"PETRA"/, ale jednak jich vzniklo hned několik a jednak je neumí žádný počítač a žáci si své programy nemohou vyzkoušet.
- Málokde se přednáší praktický přístup k programování / i analýze/, spíše je snaha prezentovat některé chiméry /viz odst. 2.3/ jako reálně využitelné metody.

Obecně není stav tak děsivý, jak jsem teď naznačil; za to děkujeme především některým zaníceným pedagogům a také vzrůstajícímu zájmu mládeže. V žádném případě však nedostáváme ze škol hotové programátory, plně připravené na praktickou práci /to platí ostatně i v jiných oborech a profesích/. Musíme tedy v každém případě programátory doškolovat a vést je k určitým pracovním návykům. Jistá potíž vzniká z prudkého rozvoje oboru; máme totiž současně problémy s pracovníky té původní samorostlé generace, kteří se často odmítají něco nového učit a kazí mládež svými dnes již zastaralými přístupy. Zvládnout tuto nelehkou situaci je úkolem vedoucích pracovníků. Zmiňuji se o tomto problému proto, že bývá v praxi často přehlížen a opomíjen. Úroveň vedoucího pracovníka poznáme snadno, prohlédneme-li si projekty a programy od různých jemu podřízených programátorů. Na první pohled uvidíme, jsou-li psány v určitém jednotném stylu; detailním zkoumáním můžeme odhalit, zda programátoři používají chybné nebo neracionální

prvky a praktiky.

Osobně doporučuji /ve světě se to tak dělá/ pověřit čas od času skupinu špičkových programátorů průzkumem úrovně programů na některém náhodně vybraném pracovišti. Osvědčilo se také zadat jeden a tentýž projekt dvěma různým pracovištím a výsledky srovnat ze všech hledisek.

Já vím, že na to není čas ani peníze. Kdybychom však byli schopni vyvodit z uvedených průzkumů nějaká opatření /morální i hmotná/ směrem k šéfům zkoumaných pracovišť, vynaložený čas i náklady by se nám brzy vrátily!

4.2 Děti programují:

Již několikrát jsme na semináři Programování mluvili o šoku, který utrpěli odborníci, když pustili děti k počítači. Tento šok byl o to drsnější, že nás zasáhl ze dvou směrů:

- velké množství dospělých /včetně řídících pracovníků/ trpí jakýmsi panickým /od slova "panika", nikoliv od "panic"/ odporem k počítačům, který je nutno lámat a odstraňovat přesvědčováním i mocí; děti se však na počítače přímo sápou;
- to, co programátoři /zvláště služebně starší generace/ považovali za tajemnou, složitou a náročnou vědu, berou děti jako zábavu a hru.

Jeden špičkový pracovník našeho oboru mi kdysi vyprávěl, jak jeho patnáctiletý syn napsal ve FORTRANu program na řešení Rubikovy kostky. Na jedné straně byl samozřejmě na čilého potomka hrdý, na druhé straně si patrně nostalgicky uvědomoval, jak by to dopadlo, kdyby takový progrem zadal některému ze "zkušených" programátorů.

Zjišťuje se, že děti k programování přistupují spontánně a projevují trochu jiný způsob a styl myšlení než "klasický" programátor. Přitom - a to je velmi zajímavé - nepovažují děti programování za technickou disciplínu. Dívenka s výraznými sklony k humanitním oborům může programovat lépe než racionálně technicky založený mládenec.

Co z toho plyne pro nás?

Především to, že můžeme očekávat na našich pracovištích nástup nové, zcela jinak myslící a uvažující generace; bude záležet

na nás, jak se s tím vypořádáme a co z nového přístupu využijeme ke zvelebení naší profese. Ze všech dětí ovšem nebudou programátoři; vyrůstá nám tedy i nová generace uživatelů, která nebude před počítači v hrůze prchat, ale bude je vyžadovat a nás programátory /pozor!/ stavět před nové, velmi obtížné úkoly !

4.3 6 miliard programátorů:

Na jedné straně je nám předkládána vize, že nové počítače nebudou programátory potřebovat. Na druhé straně nám houfy dětí bystře programují. Bude tedy v příštím tisíciletí každý programátorem nebo nebudou programátoři vůbec?

Je obtížné být futurologem; to si lze dovolit až v hodně pokročilém věku, aby se člověk nedožil srdečného smíchu potomků nad vážně a vědecky míněnými prognózami.

Pokusme se však alespoň odhadnout nejbližší budoucnost, abychom se na ni mohli připravit:

- Počítače zapadnou do života společnosti a budou považovány za samozřejmou součást civilizačního komfortu /jako třeba televize, umělé hmoty, aplikační družice/; to povede k podstatné změně postoje uživatelů k výpočetní technice všeobecně.
- Současná vlna programování ve stylu hobby ponáhlu opadne a vytvoří se stabilizované společenství programátorů-amatérů; zároveň se ustálí poměr mezi amatérem a profesionálem tak, jako se to stalo v řadě jiných oborů /radiotechnika, astronomie, fotografování, .../.
- Změna přístupu k programování povede k tomu, že jednoúčelové a jednorázové programy si budou vypracovávat specialisté jiných oborů sami.
- Všechny tři dosud jmenované trendy směřují k podstatnému zvýšení nároků na profesionální programátory. Uživatelé, amatéři i specialisté budou vyžadovat perfektní programovou podporu svých potřeb a cílů, zpracovanou na vysoké úrovni, která snese i nebývale fundovanou kritiku konzumentů.

5. Cesta profesionálů

5.1 Skupiny řešitelů a jejich vývoj:

Autor je teď nucen mluvit o "řešitelích", neboť pojem "programátor" nám pro další úvahy nepostačí.

Popsaný vývoj způsobí v řešitelské oblasti na příklad:

- zvýšení závažnosti t.zv. "systémových inženýrů" /někde se jim říká jinak/, tedy pracovníků, kteří jsou schopni koncepčně navrhovat velké a složité systémy /kde výpočetní technika bude jen jedním z prvků/ s respektováním hlavní zásady systémového inženýrství - "aby se na něco nezapomnělo" a s ohledem na potřeby a přání lidí i strojů v systému i jeho okolí;
- přesun pozornosti analytiků a programátorů na vytváření obecné, jednoduché a účinné programové podpory pro práci jiných skupin pracovníků;
- vytváření nového druhu programátorů, kteří z hotové nabídky programového zabezpečení budou schopni budovat účelové datové i programové systémy; do této skupiny se časem zařadí i dnešní "správci dat".

5.2 Vývoj versus profese:

Vyvstává nyní otázka, jak se má zachovat profesionální výrobce programů, když kolem něj pádí bouřlivý vývoj. Analogicky se stavem v jiných oborech a s učením /z ruštiny!/ všeho, co jsme dosud zvážili, se zdají nejlepšími tyto rady:

- Zachovat klid a soudnost.
- Sledovat vývoj v oboru, kriticky hodnotit, užitečné prvky s rozmyslem aplikovat.
- Sledovat stav a úroveň programování mimo profesionální základnu
  - dobré nápady přebírat, dotáhnout a využívat
  - vytvářet programovou podporu, usnadňující neprofesionálům práci.
- Kdorovat vývoj uživatelské sféry, předvídat jeho trendy a potřeby; nejen plnit konkrétní uživatelské požadavky, ale nabízet uživateli i řešení takových úloh, na které ještě sám nepřipadl.

Souhrnně: udržovat se na úrovni profesionála v tom nejlepším slova smyslu.

5.3 Závěr:

Naznačili jsme některé trudné i potěšitelné skutečnosti a výhledy v oboru "programátorství". Pokusili jsme se vyvodit nějaké více či méně užitečné závěry a dokonce i rady. Uvědomujeme si, že zítra může být všechno jinak /přečtěte si v 10-15 let starých sci-fi partie o počítačích a neubráníte se úsměvu/.

Autor nicméně zdvořile žádá, aby na semináři "Programování '84" byl tento příspěvek podroben diskusi a shovívavému zhodnocení.

6. Literatura

Vzhledem k množství vzájemně si odporujících publikací na dané téma uvádím pouze literaturu, ze které příspěvek přímo či nepřímo čerpá:

/1/ Rusín,Z: Kompilační, testovací a diagnostické prostředky v interaktivním prostředí. Sborník "Programování '84", DT ČSVTS Ostrava,1984.

/2/ Olehla, Tišer: Praktické použití Fortranu. NADAS, 1976.

/3/ Rose,K: Počítače, řízení a společnost. Svoboda,Praha 1977.

/4/ Bébr,R: Příspěvky ve sbornících semináře Havířov a Ostrava, DT ČSVTS Ostrava 1976,1977,1980,1981,1982,1984.