ASPEKTY PROGRAMOVÁNÍ VĚDECKOTECHNICKÝCH VÝPOČTŮ VE FORTRANU 77 A PERSPEKTIVY JAZYKA FORTRAN

RNDr.Jiří Hřebíček CSc., RNDr.Ivan Kopeček CSc., Ing.Jan Kučera
ÚFM ČSAV Brno

Abstrakt
V příspěvku jsou diskutovány otázky neportability mezi jazykem Fortran podle normy z r. 1966 (dále jen Fortran 66) a Fortran 77, které ztěžují využití programového vybavení vytvořeného ve Fortranu při novém vývoji programů pro VTV ve Fortranu 77. Dále jsou vyzdvihnuty nové prvky jazyka fortran 77, které přinášejí zefektivnění programování VTV. V závěru je diskutován další vývoj jazyka Fortran.

1. Úvod

Mezi programovacími jazyky určenými pro programování vědeckotechnických výpočtů (VTV) se stal nejpoužívanějším jazyk Fortran, v němž je vytvořena naprostá většina programů pro VTV. V sedmdesátých letech, kdy došlo k mohutnému rozvoji výpočetní techniky, přestal Fortran 66 splňovat požadavky kladené na moderní programovací jazyk. Proto byla provedena revize tohoto programovacího jazyka a byl navržen nový kvalitativně vyšší programovací jazyk Fortran 77. Fortran 77 je zejména vhodný pro technické, ekonomické a vědeckotechnické výpočty a poskytuje rovněž rozsáhlé možnosti pro práci s texty. Některá rozšíření Fortranu 77 byla již diskutována na semináři PROGRAMOVÁNÍ '85 v [5], proto je zde nebudeme uvádět.

Kompilátor jazyka Fortran 77 je součástí programového vybavení počítačů všech předních světových výrobků a je již používán i na mnoha počítačích vyrobených v zemích RVHP. Jedná se o počítače řady SMEP, dále ADT 4500. V poslední době byl implementován kompilátor Fortranu 77 na počítačích řady JSEP2, a to na počítačích EC 1045. Tím se Fortran 77 stává běžným programovacím jazykem i v ČSSR a jazyk Fortran 66 by bylo možno při vývoji programů VTV již přestat používat.

V dalším poukážeme na některé obtíže, které vznikají při kompilaci dříve vyvinutých programů VTV ve Fortranu 66 kompilátorem Fortranu 77.

2. Rozpory mezi fortranem 77 a předchozí normou

Nová definice jazyka Fortran byla vytvořena tak, aby byly minimalizovány rozpory mezi novou normou a její předchozí verzí. Přes rozsáhlá rozšíření není Fortran 77 nadmnožinou Fortranu 66 a některé konstrukce platné ve Fortranu 66 nelze ve Fortranu 77 použít. Změny jazyka Fortran 77 zavedené novou normou ANSI X3.9-1978, [1], způsobují konflikty s předchozí normou jazyka Fortran ANSI X3.9-1966 [2] pouze tehdy, kdy to bylo nutné k opravě chyb v předchozím standardu Fortranu 66 nebo k podstatnému rozšíření možností stávající jazykové konstrukce. Tyto změny však způsobí, že nelze vždy dříve vytvořené programové vybavení přeložit kompilátorem jazyka Fortran 77 bez syntaktických chyb. Většina těchto změn ve Fortranu 77 je zcela nepodstatná a je možné (pokud se na ně výjimečně narazí) je snadno obejít, závažnější jsou však:

- zákaz přenosu řízení dovnitř těla cyklu DO z vnějšku těla cyklu. Přístup k příkazům těla cyklu DO je pouze prostřednictvím provedení příkazu DO. V normě [2] byl povolen přenos řízení zvenku dovnitř těla cyklu za předpokladu splnění jistých podmínek. To dovolovalo ve Fortranu 66 konstrukci známou jako "rozšířené tělo cyklu".
- není dovoleno používat Hollerithovské konstanty a Hollerithovská data. Předchozí norma [2] připouštěla použití Hollerithovských konstant v příkazech DATA a CALL, uložení textů v aritmetických proměnných a polích a připouštěla odkazy na aritmetická pole na místě specifikace formátu. Poznamenejme však, že formátový popisovač H není Hollerithovská konstanta a její použití pro výstup dat je ve Fortranu 77 přípustné stejně jako "apostrofový" formátový popisovač. Naopak nová norma zakazuje čtení do formátové konverze H v popisu FORMAT.
- hodnota žádného indexového výrazu v indexu pole nesmí přesáhnout odpovídající horní hranici příslušného pole deklarovanou v programové jednotce. Např.

```
REAL A(10,5)
X=A(11,1)
```

Odkaz na prvek A(11,1) pole A není ve výše uvedeném příkaze přípustný. Norma [2] však připouštěla, aby indexový výraz přesáhl deklarovanou horní hranici indexu, pokud nebyla překročena maximální celková hodnota indexu v poli.

- odkaz na prvek pole musí mít vždy tolik indexových výrazů, kolik jich je v programové jednotce deklarováno pro jméno pole. To se týká i příkazu EQUIVALENCE. Např.

```
REAL A(2,3,5),B(10,10)
EQUIVALENCE (A(13),B(1,1))
```

Odkaz A(13) je nepřípustný. V normě [2] se však dovolovalo, aby pole deklarované jako dvou nebo trojdimensionální bylo v příkazu EQUIVALENCE uvedeno s jednodimensionálním indexem.

- jméno standardní funkce, které je použito jako skutečný parametr, musí být zapsáno v příkazu INTRINSIC. Toto jméno nesmí být zapsáno v příkazu EXTERNAL. Poznamenejme, že třída standardních funkcí ve Fortranu 77 je nadmnožinou třídy vnitřních funkcí i základních vnějších funkcí Fortranu 66.

- jméno standardní funkce nelze zapsat v příkazu specifikace typu, který by definoval jiný (než předpokládaný) datový typ. V předchozí normě [2] takový zápis postačoval k tomu, aby jméno standardní funkce označovalo uživatelský objekt. Tím se jména standardních funkcí mohou dostat do rozporu se jmény v uživatelských podprogramech. Jedná se o následující jména standardních funkcí:
ACOS, ANINT, ASIN, CHAR, COSH, DACOS, DASIN, DCOSH, DDIM, DINT, DNINT, DPROD, DSINH, DTAN, DTANH, ICHAR, IDNINT, INDEX, LEN, LGE, LGT, LLE, LLT, LOG, LOG10, MAX, MIN, NINT, SINH a TANH.

- ve vstupních a výstupních (V/V) příkazech jsou následující omezení:
do sekvenčního souboru nesmí být po zapsání záznamu "konec souboru" zapsán žádný další záznam. Stará norma to výslovně nezakazovala, i když tomuto případu nedávala jasnou interpretaci. Sekvenční soubor nesmí obsahovat současně formátové a neformátové záznamy, což bylo v normě [2] dovoleno.

jsou zakázány záporné hodnoty identifikátorů V/V jednotek. Ve V/V seznamech je zakázáno používat jednoduchých V/V seznamů uzavřených do závorek. To znamená, že závorky uzavírající více než jeden prvek V/V seznamu musí označovat cyklický seznam.

Další podrobnosti o rozporech mezi Fortranem 66 a Fortranem 77 lze nalézt v [6] a [7].

3. Vlastnosti normy Fortranu 77 znesnadňující portabilitu

I když účelem normy jazyka Fortran 77 bylo dosáhnout co největší portability vytvářených fortranských programů, jsou v její definici určitá místa, která portabilitu programů znesnadňují. To je nutné mít na zřeteli při vytváření velkých programových systémů VTV. Jedná se o následující zdroje neportability:

- podprogramy psané v jiných jazycích než ve Fortranu 77 a komunikující s fortranskými programovými jednotkami nemusí být portabilní.
- vzhledem k tomu, že porovnávací posloupnost (uspořádání znaků abecedy) nebyla v normě specifikována v celém rozsahu, nemusí být výsledek porovnání znakových výrazů stejný ve všech kompilátorech. Avšak pro porovnání znakových entit je možno používat standardní funkce LGE, LGT, LLE a LLT, které ve všech kompilátorech porovnávají na základech kódů ASCII.
- znaková data, formátová ediční konverze H, apostrofová konverze a komentářové řádky mohou obsahovat znaky, které jsou akceptovány některými kompilátory, avšak nepřípustné na jiných kompilátorech.
- nejsou definovány žádné konvence pro tvar jména souboru. Jméno souboru přípustné v jednom kompilátoru a operačním systému, nemusí být přípustné na jiném nebo může mít jinou interpretaci.
- čísla V/V zařízení a vlastnosti V/V jednotek se mohou u různých procesorů odlišovat.
- není a priori zaručeno, že programová jednotka přeložena Fortranem 66 může bez problému volat podprogram přeložený

Fortranem 77 a naopak. To může znesnadnit používání již vytvořených programových systémů VTV.

Na druhé straně však poznamenejme, že problémy s portabilitou u Fortranu 66 byly mnohem větší.

4. Komplikace kontra výhody programování ve Fortranu 77

Pokud předchozí kapitoly našeho příspěvku vzbudily dojem, že přechod na nový programovací jazyk Fortran 77 přinese programátorovi VTV spoustu potíží a nejistot, chtěli bychom zde (a to se skálopevným přesvědčením podloženým vlastní dlouholetou praxí v používání Fortranu 77) položit důraz na optimistickou stránku věci. Ta spočívá v řadě významných argumentů, mluvících ve prospěch Fortranu 77.

Nejdříve je třeba zdůraznit, že problémy, které byly diskutovány v předchozích kapitolách, budou ve skutečnosti se vší pravděpodobností spíše ojedinělé a snadno překonatelné. Přinejmenším jsou to zkušenosti autorů příspěvku i řady programátorů, kteří ve Fortranu 77 již programují. Dalším důležitým faktorem je, že přechod od Fortranu 66 do Fortranu 77 je velmi snadný a dalo by se říci, že pro programátora VTV i velmi příjemný. Programátorovi se poskytuje doposud nebývalý komfort a přitom jeho myšlení není znásilňována žádnými radikálně novými abstraktními konstrukcemi. Přesvědčené zastánce strukturovaného programování jistě potěší, že Fortran 77 obsahuje konstrukci IF-THEN-ELSE, a praktické ocení manipulaci s texty, atd.

Konečně snad také odstraníme schizofrenii teoretického "Fortran je mrtvý jazyk" s praktickým zjištěním "ve Fortranu se programuje 90 % VTV".

Nejdůležitější argument si nechám nakonec; je to efektivita a přehlednost jazyka, které budeme demonstrovat na následujícím příkladu.

Příklad:
Na intervalu OMEZ, HMEZ tabelujme funkci definovanou vztahem

$$f(x) = c_1 * x + c_2 * x^2 \begin{cases} x \geq 0,\ c_1 = 1,\ c_2 = 2 \\ x < 0,\ c_1 = 3{,}14159,\ c_2 = 3 \end{cases}$$

s krokem H.

Nejprve uvedeme příklad naprogramovaný ve Fortranu 66:

```
      WRITE(6,999)
  999 FORMAT(33HZADEJ DOLNI MEZ, HORNI MEZ A KROK)
      READ(5,998)DMEZ,HMEZ,H
  998 FORMAT(3F5.0)
      X=DMEZ
      WRITE(6,996)
  996 FORMAT(8HTABELACE//)
    1 F=FUN1(X)
      WRITE(6,997)X,F
  997 FORMAT(4H X= ,G15.5,4X,6HF(X)= ,G15.5)
      X=X+H
      IF(X.GT.HMEZ)STOP
      GOTO 1
      END
      REAL FUNCTION FUN1(X)
      IF(X.LT.0.)GOTO 1
      C1=1.
      C2=2.
      GOTO 2
    1 C1=3.14159
      C2=3.
    2 FUN1=C1*X+C2*X**2
      RETURN
      END
```

A nyní uvedeme týž příklad naprogramovaný ve Fortranu 77:

```
      PROGRAM TF77
      PRINT'('' ZADEJ DOLNI MEZ, HORNI MEZ A KROK'')'
      READ *,DMEZ,HMEZ,H
      PRINT'(A//2(A,G15.5))','TABELACE',(' X= ',X,
     *'     F(X)= ',FUN77(X),X=DMEZ,HMEZ,H)
      END
      REAL FUNCTION FUN77(X)
      IF(X.LT.0.)THEN
          C1=3.14159
          C2=3.
        ELSE
          C1=1.0
          C2=2.
      ENDIF
      FUN77=C1*X+C2*X**2
      RETURN
      END
```

Další příklady, ve kterých bude demonstrována efektivita Fortranu 77 budou uvedeny při prezentaci příspěvku.

5. Budoucnost Fortranu

Přestože norma jazyka Fortran z roku 1978 [1], známá jako Fortran 77, podstatně rozšířila možnosti tohoto programovacího jazyka ve srovnání s klasickou normou Fortranem 66 [2] a přispěla k jeho sjednocení, nebylo jí řečeno zřejmě poslední slovo. Připravuje se další revize Fortranu, která má vyústit v novou

normu, jejíž přijetí lze očekávat v roce 1988. Jazyk podle nově navrhované normy se prozatímně označuje jako Fortran 8x. Ačkoli Fortran 8x v sobě zahrnuje Fortran 77 téměř bez výjimky jako svou podmnožinu, jedná se v podstatě o nový jazyk. Řada prvků klasického Fortranu (i Fortranu 77) je do Fortranu 8x zahrnuta pouze jako archaismy v zájmu historické návaznosti a neměly by být používány v nových programech. (V další revizi jazyka koncem 90.let se počítá s jejich vypuštěním.) Patří sem i dnes velmi běžné konstrukce profilující současnou verzi jazyka.

Fortran tím však neztrácí prostředky odpovídající potlačovaným konstrukcím. Naopak - místo nich se zavádějí nové konstrukce, které jsou mohutnější a podstatně lépe odpovídají moderním vývojovým tendencím v oblasti programovacích jazyků. Fortran je dále rozšířen o mnoho zcela nových prvků, které z něj tvoří skutečně moderní programovací jazyk schopný v oblasti VTV úspěšně konkurovat i tak slavným jazykům jako je Ada. Přestavba jazyka je natolik zásadní, že programy psané důsledně v duchu Fortranu 8x nebudou na první pohled příliš připomínat klasický fortran.

V následujících kapitolách probereme nejdříve nedoporučovatelné archaické prvky a potom postupně hlavní plánovaná rozšíření. Nová norma je teprve ve stádiu přípravy a jednotlivé publikace o jejím návrhu popisují proto jeho různé varianty ([3],[4]). V konečném znění normy lze proto očekávat jisté odchylky oproti tomu, co bude řečeno v následujícím textu.

5.1 Archaismy

Mezi konstrukce ponechané ve Fortranu pouze dočasně budou zřejmě patřit následující prvky Fortranu 66 a Fortranu 77:

- Rozdělení příkazů do sloupců odvozené z formátu děrných štítků (včetně dosavadního kódování pokračovacích a komentářových řádků).
- Popisy EQUIVALENCE, COMMON, BLOCK DATA, DIMENSION, DOUBLE PRECISION, ENTRY.
- Z výkonných příkazů: aritmetické IF, počítané a přiřazené GOTO, ASSIGN, PAUSE, RETURN n, DO s neceločíselnou řídicí proměnnou.
- Jednopříkazové funkce.
- Specifická (tj.nikoli generická) jména standardních funkcí.

- Pseudodynamická pole popsaná jako REAL X(*) a předávání úseku pole do podprogramu citováním prvku pole na místě jména pole.
- Klauzule END= a ERR= v příkazech vstupu a výstupu.
- Ediční popisovače H, X a D.

5.2 Tvar zápisu programu

Program se zapíše jako posloupnost řádků libovolné délky. Žádné sloupce nemají vyhrazený význam. Komentáře lze psát na samostatný řádek nebo na konec libovolného významného řádku; v obou případech začínají vykřičníkem. Je-li účelné rozdělit příkaz na více řádků, označí se to znakem "&" na konci předcházejícího řádku. Naproti tomu lze na jeden řádek zapsat několik příkazů oddělených od sebe středníkem.

Názvy (identifikátory) mohou mít délku až 31 znaků a mohou obsahovat i znaky "_" (podtržení). Abeceda jazyka obsahuje všechny znaky kódu ASCII (tedy i malá písmena). Mezera má význam oddělovače a nesmí se proto vyskytovat uvnitř názvů, netextových konstant a základních slov jazyka (s výjimkou slov jako GO TO nebo END FILE).

Část textu programu lze oddělit do separátního souboru a vložit jej na požadované místo příkazem USING (analogickým příkazu INCLUDE u kompilátorů Fortranu 77 na počítačích IBM a VAX nebo konstrukci *** READ v implementaci ICL).

5.3 Data

Oblast datových typů a deklarací zaznamenává snad největší změny oproti Fortranu 77. Mezi ně patří:
- Datový typ BIT (a samozřejmě operace nad daty tohoto typu a odpovídající ediční popisovače pro formátování).
- Rozšířené možnosti práce s veličinami typu CHARACTER: texty nulové délky, přiřazování překrývajících se podřetězců.
- Dává se přednost deklaraci všech atributů téže veličiny v jediném příkazu (podobně jako v PL/1), přičemž více veličin se stejnými atributy lze deklarovat současně:

```
REAL, ARRAY(50), SAVE, INITIAL(50*1.0) :: A,B,C
```

Tento zápis je mnohem úspornější a přehlednější než jeho ekvivalent ve Fortranu 77:

```
REAL A(50),B(50),C(50)
SAVE A,B,C
DATA A,B,C/150*1.0/
```

- Lze explicitně řídit interní zobrazení proměnných, např.:

```
REAL, PRECISION(10)=12, EXP_RANGE=100, ARRAY(10)::X
```

  vynutí takové interní zobrazení, při němž je zaručeno alespoň 12 platných (dekadických) číslic a je možno pracovat s čísly v rozsahu alespoň od 1E-100 do 1E100.
- Příkazem IMPLICIT NONE lze zakázat defaultování typů a vynutit explicitní deklarace typů veličin.
- Uživatel má možnost zavádět privátní datové typy (jako struktury složené z položek základních typů), definovat operace nad nimi a potom s nimi pracovat, jako kdyby šlo o standardní typy.
- Příkazem MODULE lze odděleně popsat data, interní procedury (viz dále) a formáty. Tyto popisy lze potom importovat do libovolných programových jednotek příkazem USE (s případným přejmenováním). V nejjednodušším případě lze tuto konstrukci použít jako dokonalejší náhradu popisů COMMON a BLOCK DATA, koncepce modulů má však mnohem rozsáhlejší aplikační možnosti.
- Další rozsáhlá rozšíření práce s daty představuje nové řešení polí popsané dále v kapitole 5.6.

5.4 Nové řídicí struktury

Třebaže ve Fortranu 77 byla zavedena alespoň nejdůležitější moderní řídicí struktura IF - THEN - ELSE, zůstával Fortran 4 poté v tomto směru za většinou jiných programovacích jazyků. Fortran 8x zavádí další důležitá rozšíření v tomto směru:

- Je možno používat nový tvar zápisu cyklu [části zapsané v hranatých závorkách je možno vynechat]:

```
[název] DO [řídicí parametr]
  blok příkazů
REPEAT [název]
```

  "řídicí parametr" může být vynechán (nekonečný cyklus ukončený příkazem skoku), může zadávat pouze počet opakování nebo může mít tvar podobný klasickému příkazu DO, tj. proměnná=od,do,krok) se všemi čtyřmi komponentami typu INTEGER. Klasický příkaz DO zůstává rovněž dostupný s tím, že varianta s neceločíselnou řídicí proměnnou se nadále nedoporučuje.
- Je zavedena řídicí struktura SELECT CASE umožňující větvení po-

hodnoty celočíselného, logického nebo znakového výrazu.

5.5 Podprogramy

V příkazu CALL lze zadávat parametry pomocí klíčových slov a vynechávat některé skutečné parametry:

```
SUBROUTINE PODPR(A,B,C)
...
CALL PODPR(C=1.0,A=X)
```

První parametr má hodnotu X, druhý není zadán, třetí má hodnotu 1.0. V příkazu SUBROUTINE nelze sice vynechaným parametrům zadat standardní "default" hodnoty, je však možno testovat standardní funkcí PRESENT, zda byl určitý parametr při vyvolání zadán.

Uvnitř programové jednotky lze definovat vnitřní podprogramy nebo funkce lokální v této programové jednotce (zařazením této definice do modulů popsaných výše lze zajistit širší oblast jejich použitelnosti.) Lze jimi nahradit nedoporučované jednopříkazové funkce, jsou však určeny rovněž pro definování operací nad privátními datovými typy. Vnitřní funkcí lze definovat tak, že její volání má tvar běžného volání funkce nebo tak, že má funkci operátoru. Může být definováno několik funkcí téhož jména s různými typy parametrů (generické funkce) a podle typu parametrů kompilátor vybere tu, která je v daném kontextu na místě.

Příkazem RECURSIVE SUBROUTINE, resp. RECURSIVE FUNCTION je možno definovat rekursivní procedury.

5.6 Práce s poli

Manipulace s poli je (vedle zavedení modulů) jedním ze dvou nejzávažnějších rozšíření. Dosavadní verze Fortranu dovolovaly pracovat s poli v podstatě pouze po jednotlivých prvcích. To mělo za následek jednak zdlouhavé kódování řady elementárních operací nad poli, jednak to nedávalo kompilátorům dostatečné možnosti generování optimálního kódu pro daný počítač, zejména u počítačů s maticovými a vektorovými procesory.

Fortran 8x dává možnost pohlížet na pole integrálně a podporuje tak tvorbu efektivně pracujících kompilátorů i na počí-

tačích se speciálními prostředky pro operace nad poli. Zavádí rovněž řadu dalších novinek dále usnadňujících práci s poli:

- Jsou definovány operace nad poli, do nichž mohou vstupovat pole téhož tvaru a skaláry. Operace nad poli se provádějí nad všemi stejnolehlými prvky (postupně nebo paralelně), skaláry vstupují shodně do operací nad všemi prvky. Můžeme proto psát např.

```
REAL, ARRAY(5,20)::X,Y
REAL, ARRAY(-2:2,20)::Z
Z=4.0*Y*SQRT(X)
```

  Pokud by bylo nutno zabránit odmocňování záporných prvků, lze psát:

```
WHERE(X.GE.0.0)
  Z=4.0*SQRT(X)
ELSEWHERE
  Z=0.0
END WHERE
```

- Z daného pole lze vytvářet výseky (podpole). Například je-li pole Z deklarováno stejně jako v předchozím příkladě, znamená zápis Z(0,:) vektor tvořený jeho třetím řádkem (tento vektor není v paměti uložen na sousedních adresách). Tato nespojitost může jít ještě dále, zápis Z(2:-2:-2,1:19:2) vybírá v obráceném pořadí každý druhý prvek každého druhého sloupce matice Z. S takto vytvořeným podpolem lze pracovat stejně jako s obyčejným polem.
- V podprogramu není nutno uvádět meze indexů a postačí formální deklarace jako REAL A(:,:,:). Pro zjištění mezí v jednotlivých rozměrech jsou k dispozici standardní funkce.
- Příkazem IDENTIFY lze dynamicky deklarovat ještě složitější definovaná podpole. Například diagonálu matice lze předefinovat jako vektor.
- Pole je možno alokovat dynamicky, pracovní pole pro podprogram tedy již nemusí být deklarována v nadřazené programové jednotce a předávána podprogramu jako parametry.
- Je možno mapovat vícerozměrné pole jako vektor a naopak, i provádět složitější dynamické změny tvaru pole.
- Je k dispozici řada dalších standardních funkcí pro operace

nad poli, například pro sečtení nebo pronásobení prvků pole, nalezení minima a maxima, spojování polí, posouvání jejich prvků aj. Akce pomocí těchto funkcí lze provádět buď nad všemi prvky pole nebo jen nad prvky vybranými podle zadané masky.

- Existují samozřejmě i konstanty typu pole, takže je možno provést například následující přiřazení:

```
INTEGER, ARRAY(0:4)::POLE
POLE=POLE+[1,0,5,18,-5]
```

5.7 Různé další možnosti

- Existuje řada tzv. "konstantních" funkcí, poskytujících implementačně závislé konstanty, např. přesnost, s níž je zobrazena určitá proměnná nebo momentální reálný čas.
- V příkaze OPEN lze předem nastavit soubor do určité pozice a zadat, jaké operace lze se souborem provádět (např. jen čtení).
- Jestliže se v příkaze READ, WRITE nebo PRINT místo formátu uvedou dvě hvězdičky, označuje to vstup či výstup řízený seznamem. Např. vstupní data pro příkaz

```
READ(2,**)X,N,Z
```

mohou mít tvar:

```
N=5,Z=0.0,X=-1E6
```

6. Závěr

Z příspěvku vyplývá, že práce programátora VTV není lehká. Sotva získá (často podloudně) překladač jazyka Fortran 77, naučí se jej používat a zvykne si na něj, již se na něho valí nový Fortran 8x. Ten má být podle autorů normy "programovacím jazykem budoucnosti". Naštěstí jsou tu vedoucí VS, náměstci, ředitelé, koncepční pracovníci, a ti programátorovi VTV zabrzdí příliv přílišného pokroku. A tak se ptáme - přežijeme s Fortranem 77 rok 2000 ? A tuto otázku nechávám do diskuse.

7. Literatura

[1] American National Standard Programming Language Fortran - X3.9-1978. ANSI 1978.

[2] American National Standard Programming Language Fortran - X3.9-1966. ANSI 1966.

[3] M.Metcalf: Fortran Optimization. Academic Press, London, 1982. (Ruský překlad Mir, Moskva, 1985.)

[4] M.Metcalf: Has fortran a Future? In: Software Engineering, Methods And Tools in Computational Physics, JČSMF, 1985.

[5] J.Kučera: Fortran 77 a pokus o jeho omezenou implementaci. PROGRAMOVÁNÍ '85, DT ČSVTS Ostrava 1985, str. 196.

[6] P.Hanzálek: Fortran 77, přehled jazyka. VZ 611/733, ÚVT ČSAV Brno, 1985.

[7] P.Hanzálek, J.Hřebíček, I.Kopeček, J.Kučera, P.Polcar: Programovací jazyk Fortran 77 a vědeckotechnické výpočty. Academia (v přípravě k publikaci).