F O R T R A N 7 7 A P O K U S O J E H O O M E Z E N O U I M P L E M E N T A C I

Ing. Jan Kučera, ÚFM ČSAV Brno

Abstrakt: V příspěvku se popisují hlavní odchylky jazyka Fortran 77 od "klasického" Fortranu. Je zavedena podmnožina Fortranu 77 nazvaná Fortran 40.5 obsahující ty konstrukce Fortranu 77, které mohou být pomocí preprocesoru převedeny do "klasického" Fortranu, resp. jeho implementace na počítačích IBM/JSEP. Jsou nastíněny dvě možné varianty implementace tohoto preprocesoru. Závěrem jsou uvedany některé praktické zkušenosti s používáním Fortranu 77 pro vědeckotechnické výpočty ilustrující účelnost přechodu na tuto vyšší formu jazyka Fortran.

1. Úvod - prehistorie Fortranu 77

Fortran je jedním z nejstarších vyšších programovacích jazyků; jeho první implementace pochází z roku 1956 od firmy IBM. Pro svou značnou (a v té době neobvyklou) strojovou nezávislost a snadnost používání se Fortran brzy rozšířil na řadu jiných počítačů a stal se v oblasti vědeckotechnických výpočtů bezkonkurenčně nejrozšířenějším jazykem. V důsledku tohoto rychlého rozšíření vznikla řada jeho dialektů, které byly sjednoceny americkou státní normou ASA X3.9-1966. K této normě byla později publikována oficiální upřesnění a vyjasnění sporných míst. Fortran podle této normy budeme nadále nazývat "klasickým" Fortranem; často se též označuje jako "Fortran IV".

V průběhu dalšího vývoje se ukázaly jisté nedostatky výše zmíněné normy a zbytečná omezení. Proto prakticky všechny implementace zaváděly jistá rozšíření vůči normě; tím však opět docházelo k zavádění odlišných dialektů, což si vynutilo jejich opětné sjednocení. To bylo realizováno vydáním nové normy ANSI X3.9-1978 [1]. Jazyk odpovídající této normě se všeobecně (i když neoficiálně) nazývá Fortran 77. Protože norma [1] je u nás dosti obtížně dostupná, doporučujeme zájemcům o podrobnější seznámení s Fortranem 77 knihu [2].

2. Hlavní rozdíly mezi Fortranem 77 a klasickým Fortranem

Podle návaznosti na klasický Fortran či jeho běžné dialekty lze vytypovat 4 druhy rozdílů:

- rozšíření existující již u dialektů klasického Fortranu;
- principiálně nová rozšíření;
- rozšíření existující již u dialektů klasického Fortranu, avšak provedená jiným způsobem;
- omezení vůči klasickému Fortranu.

Mezi rozšíření, která jsou do Fortranu 77 převzata z dialektů klasického Fortranu lze zahrnout:

- příkazy ENTRY, RETURN n, PROGRAM, IMPLICIT, PAUSE 'text' a STOP 'text';
- textové konstanty a formáty zapsané ve tvaru 'text';
- širší kombinace typů v aritmetických výrazech;
- volnější syntaxe indexových výrazů;
- pojmenované podprogramy BLOCK DATA (a jejich napojení uvedením jejich názvu v příkazu EXTERNAL);
- parametry podprogramu typu návěští;
- zavedení generických jmen standardních funkcí.

V pozměněné formě jsou do normy zahrnuta další rozšíření:

- Překódování textových dat podle formátu (místo příkazů DECODE a ENCODE se však používá zvláštních tvarů příkazů READ a WRITE);
- Příkazy vstupu a výstupu pro soubory s přímým přístupem;
- Příkazy vstupu a výstupu podle volného formátu.

Poznamenejme, že některá rozšíření obvyklá v různých dialektech klasického Fortranu však do Fortranu 77 zahrnuta nejsou, např. příkazy NAMELIST a DEBUG nebo specifikační příkazy typu REAL*8, LOGICAL*1 apod.

Nejdůležitější změny profilující jazyk Fortran 77 však spočívají v principiálně nových příkazech nebo dílčích konstrukcích:

Rozšířený příkaz cyklu. Ačkoli příkaz DO vypadá na první pohled stejně jako v klasickém Fortranu, jsou jeho možnosti podstatně rozšířeny: Řídicí proměnná (stejně jako meze cyklu a krok) může být nejen typu INTEGER, ale též REAL nebo DOUBLE PRECISION; navíc nemusí být téhož typu (potom se převedou na ten typ, jako má řídicí proměnná.) Meze cyklu a krok mohou být zadány jako aritme-

tické výrazy a nemusí být kladné (krok ovšem nesmí být nulový.) Počet průchodů cyklem se spočítá na základě hodnot mezí a kroku v okamžiku provádění příkazu DO výrazem INT((horní mez - dolní mez + krok) / krok). Je-li hodnota tohoto výrazu nulová nebo záporná, naprovede se tělo cyklu ani jednou.

Blokový příkaz IF zavádí moderní řídicí strukturu tvaru:

```
IF(podmínka)THEN
   libovolný počet fortranských příkazů
ELSE IF(podmínka)THEN
   libovolný počet fortranských příkazů
ELSE
   libovolný počet fortranských příkazů
END IF
```

Části ELSE IF a ELSE mohou být vynechány, část ELSE IF se může libovolněkrát opakovat. Důležité je, že mezi vloženými fortranskými příkazy mohou být i další příkazy IF včetně blokových.

Práce s texty. Kromě datových typů INTEGER, REAL, COMPLEX, DOUBLE PRECISION a LOGICAL je zaveden nový datový typ CHARACTER. Různé možnosti deklarace proměnných a polí tohoto typu ilustrují následující příklady:

```
CHARACTER NAZEV*8,CPOLE(8),PAR*(*),TEXT(20)*(DELKA)
CHARACTER*800 VETA,KOPIE
```

Uvedené příkazy deklarují proměnnou NAZEV o délce 8 znaků, pole CPOLE skládající se z 8 prvků po jednom znaku, proměnnou PAR (která je formálním parametrem podprogramu a její délka je dána délkou odpovídajícího skutečného parametru), pole TEXT skládající se z 20 prvků (délka každého z nich je dána hodnotou pojmenované konstanty DELKA - viz dále) a 2 proměnné VETA a KOPIE o shodné délce 800 znaků.

Stejně jako u ostatních datových typů lze tvořit textové výrazy složené z proměnných, prvků polí, konstant a funkčních hodnot typu CHARACTER. Nad těmito základními textovými prvky lze provádět operace sřetězení (operátorem "//") a výběr podřetězu ve tvaru TEXT(od znaku : do znaku).

Příklad:

```
CHARACTER ZPRAVA*24,KOD*7,CHYBY(10)*20
```

```
DATA CHYBY(3)/'CHYBI ZNAMENKO'/
KOD='WARNING'
ZPRAVA='CHYBA UROVNE '//KOD(:1)//':'//CHYBY(3)(7:14)
```

Proměnná ZPRAVA má nyní hodnotu 'CHYBA UROVNE W: ZNAMENKO'

Pojmenované konstanty. Příkazem PARAMETER(jméno=konstanta,...) lze definovat pojmenované konstanty. Ty lze potom používat na většině míst, kde lze použít obyčejné konstanty; navíc lze na takovém místě obvykle zapsat výraz skládající se z obyčejných a pojmenovaných konstant. Hlavní oblastí použití je definice rozměrů polí a matematických konstant.

Deklarace polí. Provádí se stejně jako v klasickém Fortranu v příkazech DIMENSION, COMMON nebo v příkazu typu. Navíc poskytuje Fortran 77 možnost zadat dolní mez pole různou od jednotky - např. INTEGER POCTY(0:50). U polí, která jsou formálními parametry podprogramů je možno horní mez posledního rozměru nahradit hvězdičkou, např. DIMENSION VEKTOR(*),MATICE(N,*). V tomto případě musí VEKTOR a MATICE být formální parametry a N musí být buď formální parametr nebo veličina v COMMON-bloku.

Vstupy a výstupy obsahují proti klasickému Fortranu množství drobných i zásadních rozšíření. Rozsah i celkové zamíření tohoto příspěvku nám neumožňuje jejich vyčerpávající probrání; uvedeme pouze heslovitě změny v příkazech existujících již v klasickém Fortranu a nově zavedené příkazy.

Příkazy READ a WRITE mají následující rozšíření:

- V závorce za slovem READ či WRITE lze kromě identifikace jednotky a formátu uvést další parametry: END (návěští skoku při dočtení souboru do konce), ERR (návěští skoku při chybě), IOSTAT (indikace úspěšnosti provedení příkazu), REC (pořadové číslo záznamu při nesekvenčním zpracování).
- Místo čísla jednotky lze uvést celočíselný výraz, hvězdičku (označuje standardní jednotku) nebo název položky typu CHARACTER (pak se jedná o překódování textu analogické příkazům DECODE/ENCODE z některých dialektů Fortranu).
- Na místě formátu lze kromě návěští či názvu pole (pouze typu CHARACTER) uvést též hvězdičku (volný formát), celočíselnou

proměnnou (pokud jí příkazem ASSIGN bylo přiřazeno návěští příkazu FORMAT) nebo textový výraz zadávající přímo formát.
- V seznamu vystupujících položek u příkazu WRITE mohou být zapsány i výrazy.

Příkazy ENDFILE, BACKSPACE a REWIND mohou mít podobně jako READ a WRITE parametry ERR a IOSTAT.

Příkaz FORMAT nebo jiným způsobem zadaný formát umožňuje navíc tisknout u celých čísel levostranné nuly, tisknout i kladná znaménka, zadávat absolutní či relativní pozice ve větě (čísla sloupců) včetně možnosti návratu zpět v rámci věty, používat formátovou specifikaci A bez následujícího čísla (počet znaků je dán délkou položky v seznamu) aj.

Příkazy OPEN a CLOSE slouží vytváření a rušení souborů, jejich navazování na fortranská čísla jednotek a definici souborů s přímým přístupem. Umožňují tak na úrovni Fortranu provádět akce, které u klasického Fortranu musí být prováděny na úrovni jazyka pro řízení úloh. Tyto akce lze navíc vykonávat dynamicky v průběhu programu.

Příkaz INQUIRE umožňuje zjišťovat nejrůznější údaje o souboru nebo o fortranské jednotce, např. zda je dané jednotce přiřazen nějaký soubor a který, zda je určitý soubor možno číst bezformátově, zda existuje soubor určitého jména apod.

Další rozšíření: Ve Fortranu 77 je zavedena řada dalších drobných rozšíření jako nové standardní funkce (zejména pro práci s tarty), nové logické operátory, příkaz SAVE řídící zapomínání proměnných a polí při východu z podprogramu aj.

Přes rozsáhlá rozšíření není Fortran 77 nadmnožinou klasického Fortranu a některé konstrukce platné v klasickém Fortranu nelze ve Fortranu 77 použít. Většina těchto omezení je zcela nepodstatná, závažná omezení jsou však:
- zákaz rozšířeného rozsahu cyklu (zpětný skok do těla cyklu);
- podstatné omezení v ukládání textových informací do položek jiného typu než CHARACTER;
- zákaz Hollerithovských konstant tvaru nH... (Hollerithovské formáty používat lze, ale jen pro výstup.)

3. Návrh jazyka Fortran 40.5

Ačkoli většina předních světových výrobců počítačů považuje za samozřejmé dodávat kompilátor Fortranu 77, na počítačích JSEP, které jsou u nás nejrozšířenější, není Fortran 77 implementován. Proto jsme se pokusili navrhnout alespoň vhodnou podmnožinu Fortranu 77, která by splňovala dva protichůdné požadavky:

- zahrnovala co nejvíce důležitých rozšíření;
- umožňovala přitom automatický převod programů z této podmnožiny do klasického Fortranu nebo alespoň do Fortranu G/H firmy IBM.

Je zřejmé, že řada prvků Fortranu 77, zejména v oblasti vstupů a výstupů, nemůže být do klasického Fortranu převedena. Navržená podmnožina proto zůstane někde uprostřed mezi "Fortranem 4" a Fortranem 77; z toho důvodu pro ni byl zvolen název Fortran 40.5: (4+77)/2=40,5.

Na základě pečlivého rozboru byly do Fortranu 40.5 zařazeny následující prvky Fortranu 77:

a) všechny konstrukce, které společně s Fortranem 77 jsou zahrnuty i ve Fortranu G a H (jedná se o kombinace různých typů v aritmetických výrazech, mnohorozměrná pole, obecnější tvar indexových výrazů, příkazy IMPLICIT, ENTRY, RETURN n, STOP 'text', PAUSE 'text', pojmenovaný BLOCK DATA a jeho citace v EXTERNAL a některé drobnosti ve formátech - vynechávání oddělovačů a popisovače Tn);
b) blokový příkaz IF;
c) rozšířený příkaz cyklu;
d) hvězdička na místě posledního rozměru pole - formálního parametru;
e) proměnné, pole, výrazy a standardní funkce typu CHARACTER, avšak ne v plné šíři.

4. Možnosti implementace Fortranu 40.5

Jak bylo uvedeno v předchozí kapitole, je jazyk 40.5 navržen tak, aby bylo možno vytvořit preprocesor, překládající programy z Fortranu 40.4 do Fortranu G/H. Volba způsobu implementace preprocesoru musí vycházet ze způsobu, jak budou konstrukce

Fortranu 40.5 převáděny do Fortranu G/H. Proberme proto v hrubých rysech možné mapování jednotlivých konstrukcí do Fortranu G/H podle bodů v kapitole 3:

ad a) Tyto konstrukce lze samozřejmě ponechat beze změny, Fortran G nebo H je schopen je přeložit.

ad b) Blokové IF lze mapovat přímo do klasického Fortranu:

```
a IF(b)THEN                          a IF(.NOT.(b))GOTO u
                                       c
  c                                    GOTO z
  ELSE IF(d)THEN          ⟹         u IF(.NOT.(d)GOTO v
                                       e
  e                                    GOTO z
  .                                  v .
  .                                    .
  .                                    .
  ELSE                                 .
  f                                  y f
  END IF                             z CONTINUE
```

(Písmena ze začátku abecedy representují části původního textu, písmena z konce abecedy jsou návěští generovaná preprocesorem.)

ad c) Podobně lze mapovat obecný příkaz DO:

```
a DO b c=d,e,f                       a c=d
                                       w=(e-d+f)/f
                                       x=f nebo x=1
                            ⟹       y IF(e.LE.0)GOTO z
  g                                    g
h i                                  h i
                                       c=c+x
                                       w=w-1
                                       GOTO y
                                     z CONTINUE
```

(Písmena ze začátku abecedy representují části původního textu, písmena z konce abecedy návěští nebo celočíselné proměnné generované preprocesorem.)

ad d) Hvězdička na místě posledního rozměru pole může být ve Fortranu G a H nahrazena jedničkou.

ad e) Implementace práce s texty není dosud plně dořešena. Je zřejmé, že proměnným typu CHARACTER budou odpovídat pole typu LOGICAL*1 a polím typu CHARACTER pole typu LOGICAL*1 s počtem rozměrů o 1 vyšším. První rozměr bude mít rozsah daný délkou elementární položky, tj. číslem uvedeným

na znaky CHARACTERm. Tato délka bude navíc zapsána v celočíselné proměnné genrované preprocesorem s názvem jednoznačně odvozeným od názvu textové proměnné či pole. Vlastní operace s texty budou překládány do volání pomocných podprogramů.

Z výše uvedeného způsobu mapování konstrukcí Fortranu 40.5 do Fortranu G/H vyplývá, že nejvýhodnějším implementačním nástrojem pro tvorbu preprocesoru je univerzální makroprocesor vyšší úrovně, například u nás dosti rozšířený makroprocesor ML/1. Převod konstrukcí rozebíraných výše pod body b), c) a d) je možno realizovat přímo relativně jednoduchými makry ML/1 (zejména za předpokladu, že na vstupu do makroprocesoru se ve vstupním textu sloučí pokračovací řádky do 1 věty se základním řádkem a na výstupu se dlouhé věty opět rozdělí). Převod konstrukcí obsahujících práci s texty bude obtížnější, avšak i zde bude ML/1 vhodným nástrojem, zejména pokud se zařídí, že při vstupu textu do makroprocesoru se speciálním znakem označí příkazy operující s texty; to lze poměrně jednoduchými prostředky zajistit.

Alternativní způsob implementace spočívá v technice zvané v anglosaské literatuře "bootstrapping". Při tomto způsobu implementace by preprocesor byl napsán plně ve Fortranu 40.5. Ten je pro tento účel sice mnohem méně vhodný než ML/1, bohaté možnosti manipulace s texty však i tuto variantu činí realizovatelnou v přijatelně krátké době. Protože Fortran 40.5 je čistou podmnožinou Fortranu 77, může tato verse preprocesoru (A) být přeložena na jakémkoli počítači s překladačem Fortranu 77. Jestliže se takto přeloženému preprocesoru předloží jako vstupní data opět zdrojový text preprocesoru napsaný ve Fortranu 40.5, (A), vznikne na jeho výstupu preprocesor (B) zapsaný ve Fortranu G/H. Ten již může být přeložen na jakémkoli počítači JSEP pod operačním systémem DOS, OS nebo DOS 3/4 do cílového modulu (C). Relativní složitost preprocesoru zapsaného ve Fortranu 40.5 přitom poskytuje ideální testovací prostředek: Jestliže se preprocesoru (C) předloží jako vstupní data text (A), musí výsledkem být text totožný s (B).

4. Závěr - zkušenosti s používáním Fortranu 77

Autor a jeho kolegové na pracovišti mají možnost pracovat ve Fortranu 77 asi 1,5 roku. Získané zkušenosti lze zobecnit do následujících závěrů, které (jak autor doufá) mohou posloužit jako zdůvodnění výhodnosti přechodu na používání Fortranu 77 nebo 40.5:

- Za nejvýznamnější pokrok lze pokládat zavedení dat typu CHARACTER a zejména prostředků pro manipulaci s těmito daty.
- Podpora strukturovaného programování (blokové IF a zdokonalené DO) sice nerozšiřuje významové možnosti jazyka, umožňuje však psát (po jistém přivyknutí) srozumitelnější a snad i méně chybové programy. Je však nezbytné doprovázet používání těchto prostředků vhodnou grafickou úpravou (odsazováním podřízených příkazů).
- Z nových možností vstupů a výstupů se nejvíce cení možnost ošetřit programem chybný formát dat, výrazy ve výstupních seznamech, zápis formátů přímo v příkazech READ a WRITE a možnost vytvářet, přiřazovat a rušit soubory přímo z Fortranu v průběhu výpočtu.
- Některé rysy Fortranu 77 se uplatní zejména při interaktivních aplikacích.
- Velmi žádoucí je plná kompatibilita přeložených modulů s moduly přeloženými klasickým Fortranem (možnost vzájemného vyvolávání). V tomto směru má implementace Fortranu 77 dostupná autorovi citelné mezery.
- Bylo by mimořádně účelné, aby se do dlouhé řady výrobců počítačů vybavených překladačem Fortranu 77 zařadili i výrobci počítačů JSEP či jejich operačních systémů.

5. Literatura

[1] American National Standard Programming Language Fortran - X3.9-1978. American National Standard Institute, Inc., 1978.

[2] H.Katzan Jr.: Fortran 77. Van Nostrand, New York, 1978. (Ruský překlad: Mir, Moskva, 1982).