Poprvé s Fortranem

Fortran je programovací jazyk vhodný pro řešení numerických problémů (Formula Translator). Je nejstarším vyšším (dnes i nižším) programovacím jazykem (1957), jeho podobu udává několik norem: klasický Fortran 77, moderní 95, objektový 2003. Podobně jako pro C/C++, které je jeho konkurentem v oboru, existuje kolem desítky aktivních výrobců překladačů. Fortran i C jsou běžně k dispozici ve výpočetních centrech a velké knihovny numerických metod (NAG, IMSL, MKL, ACML, LAPACK) i paralelizační knihovny (OpenMP, MPI) mají rozhraní pro oba jazyky. K volně dostupným překladačům Fortranu patří gfortran (s vývojovým prostředím NetBeans) a drobný g95, k předním výrobcům komerčních systémů se řadí Intel (volný pro Linux) a Portland Group (umožňující programovat GPU).

Základ

f90/f95/f03, f/for	přípony zdrojových souborů s volným, resp. pevným (klasickým) formátováním
!, &, ;	pro řádkový komentář, pokračovací řádek a oddělování příkazů na jednom řádku
klíčová slova, jména	na velikosti písmen nezáleží

Standardní datové typy a literály

Standardními datovými typy se rozumějí typy celočíselné, typy reálné neboli s plovoucí desetinnou tečkou, typy komplexní, typy logické a typ znakový. Efektivita výpočtů je závislá na podpoře v hardwaru, která se zpětně promítá do reprezentace číselných typů a tím i do přesnosti a rozsahu hodnost, které lze daným typem zachytit.

INTEGER(kind)	podtyp 1, 2, 4 (default), 8, rozsah pro podtyp 4: -2**31:+2**31-1 literály: 0, +1, -1_4, typ vždy znaménkový
REAL(kind)	podtyp 4 (default, „single precision"), 8 („double precision"), zřídka 16 („quad precision") pro 4/8 B: přesnost 24/53 bitů, 7/15 desítkových číslic, rozsah max. ~3e38/1e308 literály: 0., -0., .1, +.1_8, 1e3, -1.e-3_8, 1d0 (postaru totéž co 1._8)
COMPLEX(kind)	podtypy jako REAL literály: (0.,1.), (0._8,1._8)
LOGICAL(kind)	podtypy jako INTEGER literály .FALSE., .TRUE., .TRUE._1
CHARACTER(len)	vždy statická délka řetězců (default 1), podtyp 1 (default), 2 (Unicode) literály 'A', 'AB', "abc"
typové konverze	výrazy s INTEGER operandy se vyčíslují INTEGER, výrazy s alespoň jedním REAL operandem v REAL (tj. single precision), výrazy s alespoň jedním REAL(8) v REAL(8), výrazy s alespoň jedním COMPLEX v COMPLEX př. 1/3 je 0, 1./3 je .3333333 (jen 7 platných míst!), 1._8/3 je .33... (15 platných míst)
IMPLICIT NONE	zrušení implicitní jmenné konvence, vynucení explicitních popisů jmen (konvence: nepopsaná jména začínající na I-N jsou INTEGER, ostatní REAL)

Struktura programu

Fortranský program se skládá z programových jednotek: právě jednoho hlavního programu (PROGRAM) a libovolného počtu procedur (podprogramy SUBROUTINE a funkce FUNCTION). Jednotky mohou být řazeny sekvenčně a nezávisle na sobě, lépe však je procedury sdružovat do modulů (MODULE) a ty k hlavnímu programu připojovat (USE). Jednotky lze vnořovat do jiných jednotek (CONTAINS). Každá jednotka končí popisem END, každá samostatná (nevnořená) jednotka by měla rušit implicitní jmennou konvenci (IMPLICIT NONE).

Př. modul s modulovými daty a vnořenými modulovými procedurami

```
MODULE mName
IMPLICIT NONE
specifikace modulových dat, datových typů, rozhraní procedur, definice operátorů aj.
CONTAINS ; modulové procedury
END MODULE
```

Př. hlavní program a samostatné procedury s vnořenými procedurami

```
PROGRAM name   n.   SUBROUTINE name(args)   n.   FUNCTION name(args)
USE mName
IMPLICIT NONE
specifikace lokálních dat, argumentů procedur aj.; příkazy
CONTAINS ; vnořené procedury
END PROGRAM    n.   END SUBROUTINE          n.   END FUNCTION
```

Příkazy

Dostupné příkazy pokrývají standardní výbavu imperativního programovacího jazyka: přiřazovací příkaz, podmíněný příkaz a přepínač, cyklus indexovaný a s podmínkou, skoky v cyklech a volání podprogramu s návratovým příkazem; k tomu příkazy vstupu a výstupu a příkazy pro dynamickou alokaci polí.

standardní V/V		PRINT *,expressions; WRITE (*,*) expressions; READ *,variables; READ (*,*) variables
přiřazení		x=expression (pro skaláry, pole a struktury), x=>target (pro ukazatele)
větvení:	podmínka	IF (condition) THEN; …; ELSEIF (condition) THEN; …; ELSE; …; END IF
	přepínač	SELECT CASE (expression); CASE (list); …; CASE DEFAULT; …; END SELECT
	podle masky	WHERE (mask); …; ELSEWHERE (mask); …; ELSEWHERE; …; END WHERE
cykly:	indexovaný	DO i=imin,imax,istride; …; ENDDO
	s podmínkou	DO WHILE (condition); …; ENDDO
	nekonečný	DO; …; ENDDO
	nesekvenční	FORALL (i=imin:imax:istride, j=…, scalar-mask); …; END FORALL
pojmenované konstrukce		name: DO; …; ENDDO name, též IF, SELECT CASE, WHERE a FORALL
skoky		v DO cyklech CYCLE [name] a EXIT [name], obecně GOTO
řádkové varianty		IF (condition) command; WHERE (mask) command; FORALL (…) command
volání podprogramu		CALL subroutine(arguments)
návrat z procedury		RETURN
konec, pozastavení		STOP string; PAUSE string (zastaralé)
příkazy vstupu a výstupu		READ, PRINT, WRITE, OPEN, CLOSE aj.
dynamické proměnné		ALLOCATE (arrays, pointers); DEALLOCATE (arrays, pointers); NULLIFY (pointers)
vložení zdrojového kódu		INCLUDE 'filename'

Specifikace

Popisy specifikují datový typ proměnných a symbolických konstant a uvádějí další atributy v případě statických a dynamických polí, ukazatelů a formálních parametrů procedur. Proměnné mohou, konstanty musí být svými popisy inicializovány. Lze definovat odvozené typy pro struktury a objekty. Realizace globálních proměnných pomocí společných bloků je zastaralá, je nahrazena modulovými proměnnými.

před specifikacemi	připojení modulů USE, zrušení implicitní jmenné konvence IMPLICIT NONE
specifikace typu	type(kind),attributes :: name=value, ...
kde	type je číselný nebo logický standardní typ, znakový typ CHARACTER(len) uvádí délku, attributes jsou PARAMETER pro symbolické konstanty, DIMENSION(bounds) pro pole, ALLOCATABLE pro alokovatelná pole, POINTER a TARGET pro ukazatele a cíle, INTENT(in,out) a OPTIONAL pro popis parametrů procedur, value je inicializační výraz (skalární výraz, konstruktor pole, konstruktor struktury, funkce)
popis odvozeného typu	TYPE tname; specifications of components; END TYPE
popis struktury	TYPE(tname),attributes :: name=value, ...
zastaralé popisy	společné bloky COMMON, překrývání v paměti EQUIVALENCE, uchování hodnoty SAVE ad.

Výrazy a operátory

Výrazy zahrnují běžné aritmetické, relační a logické operátory, dále operátor pro umocňování a řetězení znakových výrazů, Fortran umožňuje definovat i vlastní operátory. Výrazy jsou vyčíslovány v pořadí priorit operací, v případě rovnosti zleva doprava. Silnou stránkou Fortranu je zobecnění výrazů pro použití s poli na místech operandů.

operátory	aritmetické	umocnění ** (nejvyšší priorita), multiplikativní *, / (nižší), adiční +, – (nejnižší)
	znakové	řetězení //, podřetězec př. CH(2:2), CH(:5), CH(2:), CH(:)
	relační	==, /=, <, <=, >, >= (starší ekvivalenty .EQ., .NE., .LT., .LE., .GT., .GE.)
	logické	.NOT. (nejvyšší priorita), .AND. (nižší), .OR. (nižší), .EQV., .NEQV. (nejnižší)
	definované	lze definovat (vazbou na příslušné funkce) pro operandy standardních i odvozených typů
operátory podle priority		aritmetické a znakové (nejvyšší), relační (nižší), logické (nejnižší) v případě rovnosti priorit zleva doprava (umocnění zprava doleva)
operandy		nejen skaláry, i pole nebo sekce polí (prvkové výrazy) a struktury
celočíselné dělení		operátor / s integer argumenty, funkce MOD pro zbytek po dělení
bitové operace		pomocí vnitřních funkcí NOT, IAND, IOR ad.

Vnitřní procedury

Pole

Vnější procedury

Procedury (podprogramy a funkce) člení program do menších úseků, které plní čitelný cíl, mohou obsahovat z vnějšku nedostupná lokální data a mohou být samostatně překládány. Je definováno jejich rozhraní pomocí formálních argumentů, kterým jsou při volání procedury přiřazeny (odkazem nebo hodnotou) skutečné argumenty; procedury mohou též sdílet s okolím (globální) modulová data. Fortran neprovádí typové konverze argumentů, o to potřebnější je dbát o explicitní zpřístupnění rozhraní procedur volajícím jednotkám. Procedury mohou být rekurzivní, přetížené, čisté a prvkové.

Volání:	volání funkce (výraz): name(actual_arguments), př. sqrt(2.), sqrt(2._8)	
	volání podprogramu (příkaz): CALL name(actual_arguments), př. call random_number(x)	
Specifikace:	funkce:	FUNCTION name(dummy_arguments) s volitelným RESULT (resultname) a specifikace typu pro name, resp. resultname
	podprogram:	SUBROUTINE name(dummy_arguments)

Přiřazení mezi skutečnými a formálními argumenty je poziční nebo pomocí formálních jmen, př. CMPLX(x[,y][,kind]): print *,cmplx(1.,0.,8),cmplx(x=1.,kind=8).

Předávání argumentů: skutečné argumenty jsou předávány do procedury přednostně odkazem (předáním adresy), nelze-li, pak hodnotou (kopírováním hodnoty do lokálního paměťového místa). Odkazem lze předat jen veličinu s adresou, tj. proměnnou, prvek pole, položku struktury. Konverze typu a podtypu se neprovádějí v žádném případě, takže např. sqrt(1) nelze ani přeložit a volání f(1) pro function f(x); real f,x; f=sqrt(x); end; vrátí nesprávnou hodnotu.

Rozhraní implicitní a explicitní: shoda formálních a skutečných argumentů při volání samostatných jednotek se nekontroluje (překladač ji odvozuje implicitně). Vhodnější a často nezbytné je poskytnutí explicitního rozhraní volající jednotce, a to pomocí INTERFACE bloků nebo vnořením procedury do připojeného modulu nebo volající jednotky.

Pole předpokládaného tvaru: formální pole lze specifikovat bez uvedených mezí, které jsou při volání převzaty od skutečných argumentů a zpřístupněny funkcemi SHAPE, SIZE, LBOUND a UBOUND; př. REAL a(:),b(0:,0:).

Sdílení dat: sdílení globálních dat lze dosáhnout připojením modulu s takovými daty ve volající jednotce i volané proceduře. Vnořené proceduře jsou dostupná data jejího hostitele. Lokální data procedury jsou vně procedury nedostupná.

Rekurzivní procedury: procedury odvolávající se na sebe, buď přímo nebo nepřímo (prostřednictvím jiné procedury). Popisují se klíčovým slovem RECURSIVE, např. recursive function name(dummies) result (resultname). Rekurzivní funkce se volají pomocí name a hodnotu vracejí pomocí resultname.

Přetížené, čisté a prvkové podprogramy: přetížení je zavedení generického jména pro specifické procedury rozlišené různým počtem, typem a podtypem argumentů, obvykle pomocí INTERFACE bloku pro modulové procedury. Čisté procedury nemají vedlejší efekty (nemění hodnoty argumentů, nevypisují apod.). Prvkové procedury se definují pro skalární argumenty a volány mohou být s poli jako skutečnými argumenty.

Vstup a výstup

Příkazy pro standardní V/V:	READ fmt,list ; READ (*,fmt,attribs) list; PRINT fmt,list ; WRITE (*,fmt,attribs) list	
formátování V/V:	atribut FMT=* nebo řetězec s formátovou specifikací	
odřádkování:	atribut ADVANCE='YES' | 'NO' (default 'YES')	
ošetření V/V chyb:	atribut IOSTAT=ierr (ierr=0 bez chyby, <0 end-of-file, >0 jiná chyba)	
Formátové soubory:	OPEN (id,FILE='filename') ; READ (id,form) ; WRITE (id,form), CLOSE (id)	
Formátová specifikace:		
datové deskriptory	pro INTEGER	Iw, Iw.m, a BOZ ekvivalenty pro 2, 8, 16kovou soustavu
	pro REAL a COMPLEX	Fw.d, Ew.d, Ew.dEe, ENw.d, ESw.d
	pro LOGICAL	Lw
	pro CHARACTER	A, Aw
	obecně	Gw.d, Gw.dEe
	jsou opakovatelné	př. nIw
	Iw, Fw.d připouštějí nulové w pro výstup v minimální potřebné šířce	
řídicí deskriptory	nX pro mezery, / nebo n/ pro odřádkování, nestd. $ pro ADVANCE='NO', aj.	
vnořování	n(…), př. (2147483647(2147483647E10.3))	

Různé

měření času	call CPU_TIME(t1); call CPU_TIME(t2); print *,t2-t1 ! pro čas v sekundách
příkazový řádek	if (COMMAND_ARGUMENT_COUNT()>0) call GET_COMMAND_ARGUMENT(number,value)
proměnné prostředí	call GET_ENVIRONMENT_VARIABLE(name,value,length)
volání vnějšího příkazu	call SYSTEM('cmd')

Další možnosti

bloky rozhraní, definované operátory a přiřazení, ukazatele, OOP, interoperabilita s C

Překladače

gfortran, g95	GNU překladač, člen početné rodiny překladačů; g95 samostatná odnož
ifort	Intel překladač, v balíčku s C/C++, MKL, OpenMP a MPI
pgfortran	Portland Group překladač, v balíčku s C/C++, ACML, OpenMP a MPI, programování GPU
netbeans	grafické integrované vývojové prostředí (IDE) s překladači GNU

Vybrané volby překladačů:

	gfortran	g95	ifort	pgfortran
nápověda:	man gfortran gcc	/usr/share/g95/*.pdf	-help, man ifort	-help, man
výstupní soubor:	-o outfile (default a.out)	-o outfile	-o outfile	-o outfile
jen překlad:	-c	-c	-c	-c
min. optimalizace:	-O0 (default)	-O0 (default)	-O0	-O0, -O1 (default)
optimalizace:	-O2	-O2	-O2 (default)	-fast
max. optimalizace:	-O3 -march=native	-O3	-O3 -xH, -fast	-fast -O4
OpenMP:	-fopenmp	není	-openmp	-mp
kontrola chyb za běhu:			-C, -check all	
kontrola mezí polí:	-fbounds-check	-fbounds-check	-CB, -check bounds	-C -Mbounds
kontrola neinic. prom.:	-Wuninitialized	-Wunset-vars	-CU, -check uninit	není

Knihovny numerických metod

NAG, IMSL	velké knihovny numerických metod, IMSL často šířena s překladači
MKL, ACML	Intel Math Kernel Library, AMD Core Math Library, knihovny šířené s překladači (LA, FFT aj.)
LAPACK, BLAS	Linear Algebra Package, Basic LA Subprograms, volně šiřitelné, součást všech předchozích
připojení MKL k ifort:	ifort -mkl
připojení LAPACKu ke gfortranu:	gfortran -llapack

Odkazy

Metcalf M., Reid J., Cohen M., Modern Fortran Explained, Oxford Science, 2011
Press W. H., Teukolsky S. A., Vetterling W. T., Flannery B. P., Numerical Recipes: The Art of Scientific Computing, Third Edition, Cambridge University Press, 2007, www.nrbook.com/a/bookfpdf.php
Fortran 2003 Committee Draft, j3-fortran.org/doc/2003_Committee_Draft/04-007.pdf
Dokumentace k překladačům
Intel: software.intel.com/sites/products/documentation/hpc/compilerpro/en-us/fortran/lin/compiler_f/index.htm
GNU: gcc.gnu.org/onlinedocs

L. H., 15. 11. 2011