«به نام خدا»

گروه دینامیک سیالات محاسباتی

دانشگاه صنعتی اصفهان

فرترن

بخش اول(دستورهای کلیدی)

امیررضا هاشمی

http://CFD.iut.ac.ir

مقدمه

زبان برنامه‌نویسی فرترن در سال ۱۹۴۸ در دانشگاه منچستر نوشته شده است. کلمه *fortran* مخفف عبارت *formula_transformation* (ترجمه یا تبدیل فرمول) و این امکان را در اختیار کاربران قرار می‌داد تا فرمول‌ها را در برنامه‌ها به صورت کد کامپیوتری درآورند. فرترن از ابتدا به عنوان زبان علمی و مهندسی شناخته شده بود نام فرترن معرف همین موضوع می‌باشد.

فرترن‌ها معمولاً براساس سال تولید آن نام‌گذاری می‌گردند مثلاً فرترن ۶۶ مربوط به سال۱۹۶۶ و فرترن ۷۷ مربوط به سال ۱۹۷۷ و فرترن۹۰ مربوط به دهه ۸۰ است که در ابتدا با عنوان فرترن $8x$ و نسخه تکمیلی آن در حوالی سال ۱۹۹۰ در اختیار برنامه‌نویسان قرار گرفت. بعد از آن نیز فرترن ۹۵ در سال ۱۹۹۵ به بازار آمد که در میان این نسخ فرترن ۷۷ به دلیل پیشرفتهای نسبت به نسخه قبل (فرترن۶۶) مورد توجه بسیاری قرار گرفت اما با وجود پیشرفتهای صورت گرفته نقص‌هایی از جمله عدم امکان تخصیص "ذخیره دینامیک" و همچنین وجود برخی دستورهای نسخ‌شده در برنامه‌نویسی باعث شد تا سازندگان با ارائه نسخه ۹۰ علاوه بر رفع این مشکلات ارائه شد سیستم‌های ذخیره اطلاعات در آن بهبود یافت.

نام‌های مجاز:

نام متغیرها و پارامترها در فرترن از کاراکترهای حرفی، عددی و همچنین *underline* "_" می‌توان استفاده کرد.

ملاحظات:

۱. طول نام‌ها در فرترن تا ۳۱ کاراکتر و اولین کاراکتر آن باید کاراکتر حرفی باشد.

۲. در فرترن حروف بزرگ و کوچک تفاوتی ندارد. برای مثال: $SeLEct = sEleCT$

۳. برخی از کلمات به عنوان دستورهای کلیدی است و نباید به عنوان نام متغیرها و پارامترها استفاده شود. برای مثال $abs, sqrt, if, do, \ldots$

مثال‌هایی از نام‌های مجاز	مثال‌هایی از نام‌های غیرمجاز
iut_university	*iut.university*
input	*iut university*
Y_1234	*123_Y*
SOLver	*V/U*

متغیرها:

نوع متغیر	تعریف در فرترن۷۷	تعریف در فرترن۹۰
اعداد صحیح ساده	*INTEGER A, I, BETA*	*INTEGER :: A, I, BETA*
اعداد اعشاری با دامنه کم	*REAL FLOAT, SUM*	*REAL :: FLOAT, SUM*
اعداد اعشاری با دامنه بلند(*Double precision*)	*REAL (8) FLOAT, SUM*	*REAL (8):: FLOAT, SUM*

تعریف اعداد اعشاری با دامنه موردنظر	-	*REAL(KIND = 5) :: FLOAT*
تعریف کاراکتر(که طول کاراکتر با ضرب اندازه طول در نام کاراکتر تعیین می‌شود.)	*CHARACTER CHAR * 5*	*CHARACTER :: CHAR * 5*
تعریف عبارات منطقی	*LOGICAL RIGHT*	*LOGICAL :: RIGHT*

دستور *IMPLICIT NONE*:

این دستور در ابتدای هر برنامه یا زیر شاخه نوشته می‌شود و تعیین نوع ضمنی را توسط کامپایلر غیرفعال می‌کند. بدین ترتیب باید نام همه متغیرهای یک برنامه را تعریف نمود.

این کار از به وجود آمدن برخی اشتباهات در حین کدنویسی مانند غلط املایی و... جلوگیری می‌کند.

شروع و پایان یک برنامه:

برنامه‌ها در فرترن به صورت *Source File* فایل‌های برآمده از منبع می‌باشد یعنی برای نوشتن برنامه تنها می‌توان به فرمت تعیین شده کامپایلر برنامه را نوشت، در فرترن ۷۷ به قبل این شکل برنامه نویسی حتی در نوشتار ثابت (*Fixed format*) بود به طوریکه در نوشتن یک برنامه می‌بایست ۶ فضای اول هر خط را برای نوشتن شماره دستور توسط کاراکتر هفتم جدا می‌شد اما در فرترن ۹۰ این قید هم به دلیل محدودیت تایپ حذف شد و شکل برنامه در آن به صورت *Free format* ارائه شده است.

PROGRAM نام برنامه(برای مثال *SOLVER*)

دستورات فرترن

END
END PROGRAM
END PROGRAM نام برنامه(برای مثال *SOLVER*)

ملاحظات:

۱. طول هر خط نوشتن برنامه در فرترن ۱۲۰ تا ۸۰ کاراکتر است.
۲. ترجیح بر این است که نوع نوشتن به گونه‌ای باشد که خطها بسیار بلند نباشد، نهایت طول خط دستورات به گونه‌ای باشد که در یک صفحه *A4* بتوان کد را پرینت نمود.

کاراکترهای خاص:

- !: برای قرار دادن توضیحات در میان برنامه از این کاراکتر استفاده می‌شود و خط متناظر با خود را به رنگ سبز در می‌آید و این خط در حین اجرای برنامه کامپایل نمی‌گردد.
- &: برای دستوراتی که طول دستورات از حد مجاز بیشتر باشد در ابتدای خط دوم برای ایجاد اتصال بین خطوط دستور مورد استفاده قرار می‌گیرد.
- ;: با این کاراکتر می‌توان در یک خط چند دستور را تایپ نمود و با ; از هم جدا نمود معمولاً دستورها در انتهای هر خط پایان می‌یابد و استفاده از این کاراکتر توصیه نمی‌شود.

عملیات جبری:

اولویت بیشتر	به توان رساندن(نمایی)	**
↑	ضرب	*
	تقسیم	/
	جمع	+
اولویت کمتر	تفریق	–

مثال: $4**5/2**3*8=\frac{4^5}{2^3\times 8}=16$

اگر نوع عبارت و متغیر یا متغیر و متغیر در جایگزینی‌ها یکسان نباشد برای رفع مشکل از فرمت زیر استفاده می‌شود:

برای مثال:

$A=TYPE(B)$
$A=INTEGER$
$B=REAL$
$A=INTEGER(B+A)$

برخی توابع درونی در فرترن:

قدر مطلق	$ABS(X)$
جزء صحیح	$INT(X)$
نزدیکترین به عدد صحیح	$NINT(X)$
نزدیکترین به عدد صحیح کوچکتر از X نباشد	$CEILING(X)$
نزدیکترین به عدد صحیح بزرگتر از X نباشد	$FLOOR(X)$
تبدیل به اعشاری	$REAL(X)$
باقیمانده تقسیم	$MOD(X,Y), MODULE(X,Y)$
ماکزیمم و مینیمم	$MAX(X,Y), MIN(X,Y)$

عملیات ریاضی:

ریشه دوم	$\sqrt{x}$	$sqrt(x)$
نما رساندن	$\exp(x)$	$\exp(x)$
لگاریتم طبیعی	$\ln x$	$\log x$
لگاریتم معمولی	$\log_{10} x$	$\log 10x$
سینوس زاویه	$\sin x$	$\sin(x)$
کسینوس زاویه	$\cos x$	$\cos(x)$
تانژانت زاویه	$tan\, x$	$tan(x)$
سینوس هذلولوی	$\sinh x$	$\sinh(x)$
کسینوس هذلولوی	$\cosh x$	$\cosh(x)$
تانژانت هذلولوی	$\tanh x$	$\tanh(x)$
سینوس معکوس	$\sin^{-1} x$	$\sin^{-1}(x)$
کسینوس معکوس	$\cos^{-1} x$	$\cos^{-1}(x)$
تانژانت معکوس	$tan^{-1} x$	$tan^{-1}(x)$
عدد مختلط بر حسب مولفه‌های اعشاری و موهومی	$x+iy$	$cmplx(x)$
مزدوج مختلط	$x-iy$	$conjg(x)$

حلقه‌های تکرار:

دستور DO با شمارنده در فرترن ۹۰:

نام حلقه + توضیحات: $DO \quad I = 0,10,2$

شروع حلقه — پایان حلقه — گام حلقه

دستورات فرترن

$END \quad DO$ نام حلقه

برای مثال: $LOOP1: \quad DO \quad I = 0,10,2$

دستورات فرترن

$END\ \ DO\ \ LOOP1$

شکل کلی حلقه

$DO\ \ VARTIANLE = START, END, PITCH$

تعداد دفعاتی که حلقه اجرا می‌شود: $\left[\frac{(END - START)}{PITCH}\right]$

حلقه‌های بی‌نهایت:

DO
دستورات
$EXIT$
دستورات
$END\ \ DO$

$DO\ \ I = 0,100$
دستورات
$IF(Q <= 0)\ \ EXIT$
دستورات
$END\ \ DO$

DO
دستورات
$CIRCLE$
دستورات
$END\ \ DO$

$DO\ \ I = 0,100$
دستورات
$IF(Q <= 0)\ \ CIRCLE$
دستورات
$END\ \ DO$

دستور حلقه در فرترن۷۷:

$DO\ \ LABEL\ \ WHILE(\log ic\ \ statement)$

دستورات فرترن۷۷

$LABEL\ \ CONTINUE$

مثال:

$DO\ \ 100\ \ \ \ I = 0,10$

دستورات فرترن۷۷

100 $CONTINUE$

عبارات منطقی:

همانطور که گفته شد عبارت‌های منطقی به شکل زیر تعریف می‌گردند که تعریف این عبارت‌ها در حلقه‌های شرطی برای اعمال قیود موردنیاز است به این ترتیب:

تعریف عبارت منطقی	*LOGICAL :: RIGHT ,WRONG*
درست	*.TRUE.*
غلط	*.FALSE.*
برای مثال:	*LOGICAL , PARAMETER :: T = .TRUE.*

عملگرهای منطقی:

و	*.AND.*
یا	*.OR.*
مساوی با	*.EQV .*
نامساوی با	*.NEQV .*

عملگرهای قیاسی:

	فرترن۹۰	**فرترن۷۷**
کوچکتر از	<	*.lt.* less than
کوچکتر یا مساوی با	<=	*.le.* less equal
بزرگتر از	>	*.gt.* greater than
بزرگتر یا مساوی با	>=	*.ge.* greater equal
مساوی با	==	*.eq.* equal
مخالف با	~=	*.ne.* non equal

دستورات شرطی:(منطقی)

دستور IF:

فرمت کلی:

$IF(\log ic \;\; statement)$ دستورات فرترن

برای مثال $IF(A > B) \quad A = A + B$

دستور IF **بلوکی:** $label : if(\log ic \;\; statement)\ then$

دستورات فرترن

$end \;\; if \;\; label$

برای مثال $label : if(10.lt.A.ge.20)\ then$

$save = A$

$end \;\; if \;\; label$

دستور IF **با دو بلوک:** $label : if(\log ic \;\; statement)\ then$

دستورات فرترن

$else$

دستورات فرترن

$end \;\; if \;\; label$

برای مثال $label : if(A.gt.B)\ then$

$C = A / B$

$else$

$C = B / A$

$end \;\; if \;\; label$

دستور IF **با بلوک‌های تودرتو:** $label : if(first \;\; \log ic \;\; statement)\ then$

دستورات فرترن

else if (second logic statement) *then*

دستورات فرترن

else if (third logic statement) *then*

دستورات فرترن

else if (nth logic statement) *then*

دستورات فرترن

else

دستورات فرترن

end if label

برای مثال

```
label : if (1800 <= year < 1900) then
CENT = 19th
else if (1900 <= year < 2000) then
CENT = 20th
else if (2000 <= year) then
CENT = 21th
end if label
```

دستور شرطی *CASE*:

label :*select case*(logic statement)

بلوک اول

case(first logic state)

دستورات فرترن

بلوک دوم

case(second logic state)

دستورات فرترن

بلوک n ام

$case(nth\ \log ic\ state)$

دستورات فرترن

بلوک آخر

$case\ defualt$

دستورات فرترن

$end\ label\ case$

برای مثال:

$century : select\ case(year)$
$case(1800:1900)$
$cent = 19th$
$case(1900:2000)$
$cent = 20th$
$case(2000:)$
$cent = 21th$
$end\ case\ century$

«به نام خدا»

گروه دینامیک سیالات محاسباتی
دانشگاه صنعتی اصفهان

فرترن
بخش دوم(آرایه‌ها ،ورودی و خروجی‌ها،توابع درونی و خارجی)

امیررضا هاشمی

آرایه‌ها:

آرایه‌ها در فرترن آرایشی از متغیرهاست که همانند تعریف متغیرها انواع صحیح ساده، اعشاری و کاراکتری می‌باشد. یکی از مهمترین تفاوت‌های فرترن۹۰ با فرترن‌های قبل از آن امکان اختصاص حافظه دینامیک است که در فرترن‌های قبلی امکان تعریف آرایه با طول متغیر وجود نداشت اما در نسخه۹۰ امکان تعریف آرایه با طول متغیر فراهم شده است.

آرایه‌ها از لحاظ ابعاد می‌تواند تا ۷ بعد داشته باشد اما به دلیل نیاز با حافظه بسیار زیاد برای آرایه‌های بیش از ۳ بعد، عملاً آرایه‌ها تا ۳ بعد مورد استفاده قرار می‌گیرد و به تعداد ابعاد یک آرایه رتبه($RANK$) گفته می‌شود.

آرایه‌ها به صورت‌های زیر تعریف می‌شوند:

اعداد صحیح ساده

$$INTEGER, DIMENSION\ (dl, du) :: ARRAY$$
$$INTEGER :: ARRAY\ (dl, du)$$

که dl, du به ترتیب اندیس بالا و اندیس پایین آرایه را مشخص می‌نماید، این اندیس‌ها می‌تواند هر عبارت صحیحی که اعداد صحیح یا اعشاری مثبت یا منفی تولید کند باشد اما در فرترن تنها $INT\ (dl)$ و $INT\ (du)$ را به عنوان اندیس‌های بالا و پایین درنظر می‌گیرد.

نمایش بالا یک آرایه یک بعدی را نشان می‌دهد که طول آرایه آن از dl تا du به صورت $INTEGER$ تعریف گردیده است.

اعداد اعشاری

$$REAL, DIMENSION\ (N) :: VECTOR$$
$$REAL :: ARRAY\ (N) :: VECTOR$$

آرایه بالا یک آرایه یک‌بعدی است که اندیس پایین آن ۱ است و اندیس بالای آن عدد N می‌باشد.

آرایه‌های کاراکتری

$$CHRACTER\ (LEN\ = 10), DIMENSION\ (-5, 5) :: NAMES$$

این دستور آرایه $NAMES$را با ۱۱ عضو کاراکتری نشان می‌دهد که طول هر عبارت کاراکتری در نهایت ۱۰ کاراکتر باشد.

به طور کلی نمایش آرایه به شکل:

$$TYPE, DIMENSION\ (WEST\ : EAST, SOUTH\ : NORTH, BOTTOM\ : TOP) :: ARRAY$$

نشان داده می‌شود. برای مثال:

$$REAL, DIMENSION\ (5 : 23, -6 : 0, 8) :: VOL$$

این یک تعریف آرایه سه‌بعدی است که یک بعد آن از ۵ تا ۲۳ اندیس‌گذاری، بعد دوم از ۶- تا صفر و بعد سوم از ۱ تا ۸ اندیس‌گذاریِ از نوع اعشاری تعریف شده است.

عملیات بر روی آرایه‌ها:

مقداردهی به آرایه‌ها	$A(5,7) = 2.5, B(-1,0,3) = 7.3$
این دستور به تمام عضوهای با بعد دوم۵ مقدار۲,۳ را نسبت میدهد	$A(:,5) = 2.3$
کاراکتر دهم عضو کاراکتری(۵و۱) را نشان می‌دهد	$NAME\ (10)(1,5)$

برخی عملیات جبری:

این دستور به تمام اعضای آرایه مقدار ۱٫۵ را اضافه می‌کند	$ARRAY = ARRAY + 1.5$
در صورت تطابق‌پذیری A,B مجموع این دو را در C قرار می‌دهد	$C = A + B$
همه اعضای آرایه را بر عدد N تقسیم می‌نماید	$ARRAY / N$
در صورت تطابق‌پذیری A,B ضرب این دو را در C قرار می‌دهد	$C = A * B$

برش با گام:

این دستور آرایه با اندیس ۵۰ تا ۱۰۰ را با گام ۲ برش می‌دهد.

$$ARRAY = ARRAY\,(dl:du:stride)$$
$$ARRAY = ARRAY\,(50:100:2)$$

برعکس نمودن آرایش متغیرها:

این دستور اندیس منتسب به درایه‌های آرایه موردنظر را برعکس می‌نماید.

$$REVERSE = REVERSE\,(dl:du:-1)$$
$$REVERSE = REVERSE\,(50:100:-1)$$

مقدار صفر:

در فرترن آرایه‌ای که درآن dl بزرگتر از du باشد به عنوان اندازه صفر معرفی می‌گردد. مثلاً برای:

$$REAL, DIMENSION\,(10) :: A$$

$A(11:10)$ یا $A(1:11)$ دارای اندازه صفر است اما $A(-1:5)$ چون در محدوده تعریف شده نیست غیرمجاز است.

آرایه‌ها با طول متفاوت در برنامه:

برای این دسته از آرایه‌ها طول ابتدا و انتها تعریف نمی‌شود و اندازه آرایه به نوع برنامه و تعداد متغیرهای اعمالی از طریق برنامه بستگی دارد، در برنامه هر تعداد برای طول آرایه در نظر گرفته شود همان اندازه به عنوان طول آرایه نسبت داده می‌شود. برای تعریف این نوع آرایه‌ها از شکل:

$$REAL, DIMENSION\,(:,:,:), ALLOCATABLE :: ARRAY \quad (I)$$

این دستور در ابتدای برنامه می‌آید و آرایه سه‌بعدی با طول متغیر را تعریف می‌کند عبارت $ALLOCATABLE$ نشان دهنده قابل انطباق بودن این آرایه است. برای رفع تخصیص این قابلیت از آرایه تعریف شده از دستور:

$$DEALLOCATABLE\,(ARRAY\,) \quad (\Pi)$$

استفاده می‌شود.

پس هر قسمتی از برنامه که دستور I چاپ گردد. آرایه متناظر با آن با طول متغیر تبدیل می‌شود و با دستور Π این امکان برداشته می‌شود. در صورتی که آرایه تعریف شده در ادامه برنامه رها شود نیازی به ذکر دستور Π برای رفع تخصیص نیست.

خواندن و نوشتن در فرترن:

چاپ کردن یک متغیر یا پارامتر	*PRINT *,VAR,A,I*
چاپ کردن یک متغیر یا پارامتر شکل اجرا شده:	*READ*,INPUT ,SELECT* (I) 450 360 *ENTER* (Π) 450 *ENTER* 360 *ENTER*
چاپ کردن یک متن	*PRINT *,"This is an example text "* *PRINT *,'This is an "example"text '*
ترکیب نمایش چاپ یک متن و متغیر	*PRINT *,'The value of variable is :',VAR*
نمایش طول یک کاراکتر(این دستور تمام فضاهای خالی را حذف می‌کند)	*PRINT *,LEN ("SELECT ")*
نمایش طول یک کاراکتر(این دستور فضاهای خالی را حذف نمی‌کند)	*PRINT *,TRIM ("SELECT ")*

فرمت‌ها:

فرمت‌ها مدل‌هایی قابل دسترس برای نمایش خروجی‌ها و یا نحوه ذخیره اطلاعات یک برنامه می‌باشد. به کمک این مشخصه‌ها می‌توان نحوه خروج داده‌ها را کنترل یا محدود نمود.

فرمت I: این فرمت طول میدان اعداد صحیح را محدود می‌کند. شکل کلی آن با شکل $IW.m$ است که W (یک عدد صحیح غیر صفر) طول میدان عدد صحیح و m تعداد رقم چاپ را نمایش می‌دهد و ذکر آن اختیاری است.

برای مثال: $-73 \xrightarrow{I6.3} bbb-73$

$-73 \xrightarrow{I6} -73$

فرمت F: طول میدان اعشاری را محدود می‌کند. فرم کلی آن $FW.d$ که W (یک عدد صحیح غیر صفر شامل قسمت قبل و بعد از اعشار+ممیز اعشاری) طول میدان عدد اعشاری و d تعداد ارقام بعد از اعشار را نشان می‌دهد.

برای مثال: $73.445 \xrightarrow{F14.6} bbbbb73.445000$

$73.445 \xrightarrow{F8.3} 73.445$

فرمت E: این فرمت کلیت بهتری نسبت به فرمت F دارد و برای محدود کردن اعداد اعشاری است فرمت کلی آن $EW.dEe$ است که W (یک عدد صحیح غیر صفر شامل قسمت قبل و بعد از اعشار+ممیز اعشاری) طول میدان عدد اعشاری و d تعداد رقم بعد از نقطه اعشار و e رقمی که در نما ظاهر می‌شود را نشان می‌دهد.

برای مثال: $5.3\times10^{234} \xrightarrow{E13.4E3} bbb.5300E+235$

فرمت A: برای محدود کردن عبارت‌های کاراکتری مورداستفاده قرار می‌گیرد فرم کلی آن AW که W طول میدان عبارت کاراکتری را تعیین می‌کند.

برای مثال: $'WEIGHT' \xrightarrow{A10} bbbbWIEGHT$

تکرارها در مشخصه‌های فرمت:

اگر بخواهیم بیش از یک عبارت را با یک نوع فرمت محدود کنیم باید فرمت موردنظر را تکرار کنیم. فرمت کلی تکرار $N\,(FORMATS)$است که N تعداد تکرار این نوع فرمت‌هاست.

برای مثال $5(F14.6,E12.5E2)$

این عبارت یعنی فرمت‌های F,E متناظر برای متغیرها ۵ بار تکرار می‌شوند.

استفاده از فرمت‌ها در خروجی:

نحوه نمایش $PRINT\ '(FORMATS)',VAR$

برای مثال: $PRINT\ '(I7,F7.3,E10.5,A3)',I,R1,R2,CHAR$

دیگر دستورعبارت‌های مورد استفاده در چاپ:

فضاهای خالی: برای قرار دادن فضاهای خالی از فرم کلی nX که n تعداد فضای خالی را معین می‌کند.

ابتدای سطر: برای شروع شدن از خط جدید از علامت / استفاده می‌شود.

انتقال موقعیت چاپ: برای انتقال موقعیت چاپ به چپ و راست به ترتیب از TRn,Tn استفاده می‌شود که n تعداد فواصل(TAB) می‌باشد.

برای مثال $PRINT\ (I7,5X,F7.3/,TR2,E10.5,T2,A3),I,R1,R2,CHAR$

این دستور ابتدا I را با دامنه۷ چاپ می‌کند سپس ۵ جای خالی در نظر می‌گیرد و $R1$ را با فرمت F چاپ می‌نماید سپس به خط جدید می‌رود و بعد از ۲ تب به سمت راست $R2$ را با فرمت Eچاپ می‌کند سپس با ۲ تب به چپ برمی‌گردد و $CHAR$را با فرمت $A3$ چاپ می‌کند.

خروجی/ورودی به واحد دلخواه:

در اکثر سیستم‌ها، صفحه کلید شماره واحد ۵ و مانیتور شماره واحد ۶ را داراست. با استفاده از این شماره‌ها می‌توان معین نمود که ورودی یا خروجی بر روی چه واحدی ظاهر شود. برای نمایش روی واحد خاص دستور ورودی همچنان READ ولی دستور خروجی به WRITE تبدیل می‌شود.

فرم کلی نمایش ورودی

READ (UNIT=U , FMT=F) VAR1,VAR2,...

(واحد نمایش — نوع فرمت خواندن)

فرم کلی نمایش خروجی

WRITE (UNIT=U , FMT=F) VAR1,VAR2,...

(واحد نمایش — نوع فرمت خواندن)

اگر U=5 باشد ورودی از صفحه کلید صورت می‌گیرد.

و اگر U=6 باش خروجی از طریق صفحه مانیتور نمایش داده می‌شود.

در صورتی که ورودی و خروجی مهم نباشند کاراکتر * استفاده می‌شود، و در صورتی که فرمت ورودی و خروجی مشخص نباشد نیز از کاراکتر * استفاده می‌شود. با این توضیح می‌توان گفت که:

READ(*,*) = READ*,PRINT* = WRITE(*,*)

برای مثال :

READ(5,*) VAR1,VAR2

WRITE(6,*) VAR1,VAR2

هرچه ورودی و خروجی برنامه از محدودیت فرمت کمتری برخوردار باشد احتمال وقوع خطاهای format نیز کاهش می‌یابد. برای جلوگیری از به وجود آمدن خطا شکل FMT را می‌توان حذف نمود. برای مثال:

WRITE(6) VAR , READ(5) VAR

فایل‌ها:

نحوه ساخت file : به وسیله دستور OPEN می‌توان یک فایل ایجاد نمود. فرمت کلی:

OPEN(UNIT=N , لیست گزینه‌ها)

OPEN(UNIT=N,FILE='Name.Suffix',STATUS='STAT'

عبارت UNIT = این عبارت به عنوان عددی است که ما به این فایل نسبت داده‌ایم. وجود عبارت UNIT اختیاری است.

=FILE : در روبروی این عبارت نام و پسوندی که فایل با آن ساخته می‌شود را می‌نویسیم.

=STATUS : وضعیت فایل را معین می‌کند که آیا این فایل ساخته شده جدید است یا قبلا وجود داشته یا قرار است جایگزینی صورت گیرد یا غیره که حالات مختلف آن به قرار زیر است:

OLD : فایل قبلا روی سیستم وجود داشته است.

NEW : فایل قبلا روی سیستم وجود نداشته است.

REPLACE : فایل قبلا روی سیستم وجود نداشته است و قرار است جایگزین آن شود و یا اینکه فایل وجود نداشته و قرار است ایجاد شود.

SCRATCH : وضعیت فایل وابسته به وضعیت فایل روی سیستم است به همین دلیل وقتی 'STAT='SCRATCH عبارت =FILE باید حذف گردد چون نام و پسوند فایل روی سیستم معین است.

UNKNOWN : بستگی به وضعیت سیستم دارد.

دستور CLOSE :

به طور معمول همه‌ی فایل‌ها در پایان برنامه بسته می‌شوند اما اگر بخواهیم باز شدن فایل را در مکانی از برنامه ببندیم. از دستور CLOSE استفاده می‌شود.

فرم کلی CLOSE(UNIT=شماره واحد , لیست گزینه‌ها)

مهمترین گزینه‌ها در CLOSE :

IOSTSA=ios : یک متغیر صحیح پیش‌فرض است که اگر دستور به درستی اجرا شود صفر خواهد بود و در غیر این‌صورت یک مقدار مثبت خواهد داشت.

ERR=error-label : برچسب یا شماره دستوری است که در صورت وقوع یک خطا در طی اجرا دستور کنترل به آن منتقل می‌شود.

STATUS= : وضعیت فایل را همانند دستور OPEN معین می‌کند که به قرار زیر است.

'KEEP' : فایل را در حافظه نگه می‌دارد.

'DELETE' : فایل را از حافظه پاک می‌کند.

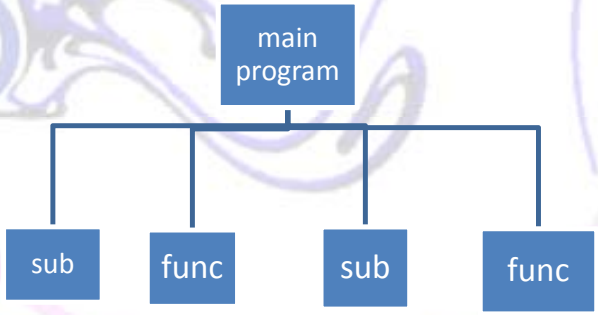
فایلی که STATUS='SCRATCH' باز شده است نمی‌توان STATUS='KEEP' ذخیره کرد.

مثال
CLOSE(16,IOSTAS=SUM , STATUS='DELETE')

تعداد گزینه‌ها در دستورهای OPEN , CLOSE بسیار زیاد است که می‌توان به help فرترن جهت استفاده از دیگر گزینه‌ها مراجعه نمود.

تابع‌ها و زیرشاخه‌ها:

function و subroutine از جمله زیر برنامه‌های داخلی (درونی) زبان فرترن اند. تفاوت مهم بین functionها و subroutineها در این است که function تنها یک متغیر را برمی‌گرداند ولی subroutine ها می‌توانند چند متغیر را برگردانند.



یک برنامه می‌تواند تعداد زیادی function و subroutine داشته باشد.

function : فرم کلی تعریف function

FUNCTION نام(آرگومان) RESULT(نام نتیجه)
دستورات فرترن
تعدادنتیجه = نام نتیجه
END FUNCTION نام

برای مثال

```
FUNCTION SOLVER(INPUT) RESULT(OUTPUT)
 دستورات فرترن, REAL::INPUT,OUTPUT
   (OUTPUT=INPUT)
END FUNCTION SOLVER
```

می‌توان RESULT را حذف نمود و جواب را به وسیله‌ی خود آرگومان دریافت نمود.

برای مثال

```
FUNCTION SOLVER(INPUT)
 دستورات فرترن
(INPUT=INPUT+S)
END FUNCTION SOLVER
```

SUBROUTINE : تعریف این دسته از زیر برنامه‌ها به شکل:

```
SUBROUTINE نام(آرگومان‌های خروجی,آرگومان‌های ورودی)

دستورات فرترن

RETURN → به عنوان کنترل کننده در برنامه می‌آید و می‌توان آن‌را حذف کرد.
END SUBROUTINE نام
```

نحوه قرارگیری function در برنامه

```
Program Main

دستورات فرترن

Contains
Functions(تابع‌ها)
END program Main
```

نحوه فراخوانی subroutine در برنامه اصلی :

هر subroutine با دستور زیر که در برنامه اصلی می‌آید می‌تواند با برنامه اصلی link گردد.

فرم کلی

```
Call subroutine نام(آرگومان‌های ورودی و خروجی)
```

مثال:

```
Call subroutine(A,B,C)
```

تعریف متغیرهای ورودی و خروجی در subroutine :

فرم کلی متغیرها یا آرایه‌های ورودی و خروجی

```
Type,INTENT(RANGE):: نام متغیرها
```

مثلا برای SUBROUTINE SOLVER(A,B,C) :

متغیرهای ورودی REAL,INTENT(IN)::A

متغیرهای خروجی REAL,INTENT(OUT)::B,C

متغیرهای هم ورودی و هم خروجی REAL,INTENT(INOUT)::D

آرایه‌های خروجی REAL,INTENT(OUT),DIMENSION(12,20,50)::B,C

یا REAL,INTENT(OUT)::B(12,20,50),C(12,20,50)

FUNCTIONهای پرکاربرد :

C=MATMUL(A,B)	جمع دو آرایه A و B
C=DOT-PRODUT(A,B)	ضرب نقطه‌ای بردارهای تطابق پذیر A و B
C=TRANSPOSE(A)	تشکیل ترانهاده یک آرایه دوبعدی
C=MAXVAL(A,B)	تعداد حداکثر آرایه‌های AوBرا برمی‌گرداند.
C=MINVAL(A,B)	تعداد حداقل آرایه‌های AوBرا برمی‌گرداند.
C=MAXLOC(A)	مکان عضو حداکثر را در آرایه بر می‌گرداند.
C=MINLOC(A)	مکان عضو حداقل را در آرایه بر می‌گرداند.
C=SUM(A)	مجموع همه‌ی اعضا آرایه را بر می‌گرداند.
C=LBOUND(A,1)	اولین محدوده‌ی پایین را باز می‌گرداند.
C=UBOUND(A,1)	اولین محدوده‌ی بالا را باز می‌گرداند.

زیربرنامه‌های خارجی[1]:

زیربرنامه‌های خارجی توابعی درونی هستند که تنها لیستی از تعریف متغیرها و پارامترها را با ویژگی‌های مشخصی ممکن می‌سازد و شامل دستورات اجرایی نیستند. این توابع به صورت ضمیمه در برنامه اصلی و یا زیر برنامه‌های درونی فراخوانده می‌شود.

مهمترین این زیربرنامه‌های خارجی PARAM، COMMON، MODULE می‌باشد که به تشریح آن می‌پردازیم:

PARAM:

برای تعریف پارامترهایی که معمولاً مقدار آن‌ها در کل برنامه اصلی و زیرشاخه‌های درونی یکسان و ثابت در نظر گرفته می‌شود و با دستور

Include ‘param.inc’

که در ابتدای هر برنامه یا زیربرنامه درونی فراخوانده می‌شود و به قسمت External Dependencies در برنامه اجرایی اضافه می‌گردد.

PARAM یک فایل NOTEPAD با پسوند INC است (برای مثال PARAM.INC) که در محل برنامه ساخته می‌شود.

[1] External links

COMMON :

این زیربرنامه برای جلوگیری از تعریف متغیرهای مشترک در ابتدای هر برنامه اصلی و زیرشاخه‌ها استفاده می‌شود و ضمن برنامه اصلی و در ابتدای آن‌ها تعریف می‌شود که شکل دستور آن

در فرترن ۹۰ Include 'COMMON.f90'

در فرترن ۷۷ Include 'COMMON.for'

می‌باشد و شکل تعریف متغیرها و به اشتراک گذاشتن آن‌ها به صورت زیر است:

تعریف متغیرها

Common/نام متغیرهایی که قرار است به اشتراک گذاشته شود/نام گذاری گروهی برای متغیرهای به اشتراک گذاشته شده

برای مثال:

```
Real(8),dimension(Nx,Ny)::Array
Real(8)::A,B
Integer:I
Common/var/ array,A,B,I
```

MODULE:

جدیدترین لینک خارجی موجود در فرترن است که برای رفع نواقص COMMON توسط سازندگان فرترن به آن اضافه شده است که بدلیل آنکه تمامی متغیرها به اشتراک گذاشته شده در COMMON ها در کل برنامه اصلی و زیرشاخه‌هایی که COMMON را ضمیمه نموده‌اند تعریف می‌گردد، عملاً امکان محدود کردن متغیرها در زیرشاخه‌ای خاص وجود ندارد. به همین دلیل تابع خارجی مدول ارائه شده تا به برنامه‌نویس این امکان را بدهد که متغیرهای تعریف شده را بین هر زیرشاخه و برنامه اصلی موردنظر و خاص به اشتراک بگذارد و به‌گونه‌ای اشتراک پاره‌ای به وجود آورد.

چگونگی تعریف یک مدول ساده به شکل زیر است:

MODULE نام مدول

تعریف متغیرهایی که قرار است به اشتراک گذاشته شود.

END MODULE نام مدول

برای مثال:

```
Module variables
```

Real(8)::A,VAR
Integer::I
Real(8),dimension(Nx,Ny)::Array
End module variables

نحوه فراخوانی مدول در برنامه اصلی و زیرشاخه‌ها در ابتدای برنامه به شکل زیر صورت می‌گیرد:

Use نام مدول,only:: نام متغیرهایی که در این برنامه اصلی یا زیرشاخه موردنظر است

برای مثال: Use variable,only::A,Array

خاصیتی جالب برای مدول‌ها:

مدول‌ها علاوه برآنکه امکان به اشتراک گذاشتن متغیرها را فراهم می‌کنند امکان به اشتراک گذاشتن یک یا چند تابع(Function) را برای یک برنامه یا زیرشاخه ممکن می‌سازد. فرم کلی آن به شکل زیر است:

Module نام مدول

تعریف متغیرهایی که قرار است به اشتراک گذاشته شود

Contain

Function نام تابع (متغیر ورودی و خروجی) (می‌توان چندین تابع در این قسمت به اشتراک گذاشت)

دستورات

End function نام تابع

End module نام مدول

در اینجا اگر برخی متغیرهایی داشته باشیم که تنها در خود مدول کاربرد داشته باشد مثلاً در Functionها باید آن متغیر را به صورت خصوصی و بقیه متغیرها را به صورت عمومی تعریف نمود شکل کلی تعریف متغیرهای خصوصی و عمومی در مدول‌ها برای مثال به شکل زیر است:

متغیرهای عمومی Real(8),public::A,B

متغیرهای خصوصی Integer,private::A,B