

«به نام خدا»



گروه دینامیک سیالات محاسباتی
دانشگاه صنعتی اصفهان

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

امیررضا هاشمی

کلیه حقوق این کتاب الکترونیک متعلق به گروه دینامیک سیالات محاسباتی دانشگاه صنعتی اصفهان می‌باشد
هر گونه چاپ، تکثیر و استفاده از آن با ذکر منبع بلا مانع است.

<http://CFD.iut.ac.ir>

آرایه‌ها:

آرایه‌ها در فرترن آرایشی از متغیرهاست که همانند تعریف متغیرها انواع صحیح ساده، اعشاری و کاراکتری می‌باشد. یکی از مهمترین تفاوت‌های فرترن ۹۰ با فرترن‌های قبل از آن امکان اختصاص حافظه دینامیک است که در فرترن‌های قبلی امکان تعریف آرایه با طول متغیر وجود نداشت اما در نسخه ۹۰ 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:

INTEGER, DIMENSION (dl, du) :: ARRAY اعداد صحیح ساده

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 در نظر می‌گیرد.

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

REAL, DIMENSION (N) :: VECTOR اعداد اعشاری

REAL :: ARRAY (N) :: VECTOR

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

CHARACTER (LEN = 10), DIMENSION (-5,5) :: NAMES آرایه‌های کاراکتری

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 سه بعدی است که یک بعد آن از ۵ تا ۲۳ اندیس گذاری، بعد دوم از -۶ تا صفر و بعد سوم از ۱ تا ۸ اندیس گذاری از نوع اعشاری تعریف شده است.

عملیات بر روی آرایه‌ها:

$A(5, 7) = 2.5, B(-1, 0, 3) = 7.3$	مقداردهی به آرایه‌ها
$A(:, 5) = 2.3$	این دستور به تمام عضوهای با بعد دوم ۵ مقدار ۲,۳ را نسبت میدهد
$NAME(10)(1, 5)$	کاراکتر دهم عضو کاراکتری (۱ و ۵) را نشان می‌دهد

برخی عملیات جبری:

$ARRAY = ARRAY + 1.5$	این دستور به تمام اعضای آرایه مقدار ۱,۵ را اضافه می کند
$C = A + B$	در صورت تطابق پذیری A, B مجموع این دو را در C قرار می دهد
$ARRAY / N$	همه اعضای آرایه را بر عدد N تقسیم می نماید
$C = A * B$	در صورت تطابق پذیری A, B ضرب این دو را در C قرار می دهد

برش با گام:

$ARRAY = ARRAY (dl : du : stride)$
این دستور آرایه با اندیس ۵۰ تا ۱۰۰ را با گام ۲ برش می دهد.
 $ARRAY = ARRAY (50 : 100 : 2)$

برعکس نمودن آرایش متغیرها:

$REVERSE = REVERSE (dl : du : -1)$
این دستور اندیس منتسب به درایه های آرایه مورد نظر را برعکس می نماید.
 $REVERSE = REVERSE (50 : 100 : -1)$

مقدار صفر:

در فزترین آرایه ای که در آن dl بزرگتر از du باشد به عنوان اندازه صفر معرفی می گردد. 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.

آرایه ها با طول متفاوت در برنامه:

برای این دسته از آرایه ها طول ابتدا و انتها تعریف نمی شود 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 می شود. برای تعریف این نوع آرایه ها از شکل:

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

این دستور در ابتدای برنامه می آید و آرایه سه بعدی با طول متغیر را تعریف می کند عبارت $ALLOCATABLE$ نشان دهنده قابل انطباق بودن این آرایه است. برای رفع تخصیص این قابلیت از آرایه تعریف شده از دستور:

$DEALLOCATABLE (ARRAY) (II)$

استفاده می شود.

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

خواندن و نوشتن در فرترن:

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

فرمت ها:

فرمت ها مدل هایی قابل دسترس برای نمایش خروجی ها و یا نحوه ذخیره اطلاعات یک برنامه می باشد. به کمک این مشخصه ها می توان نحوه خروج داده ها را کنترل یا محدود نمود.

فرمت I : این فرمت طول میدان اعداد صحیح را محدود می کند. 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 می دهد و ذکر آن اختیاری است.
برای مثال:
$$-73 \xrightarrow{-16.3} bbb - 73$$

فرمت F : طول میدان اعشاری را محدود می کند. 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 می دهد.
برای مثال:
$$73.445 \xrightarrow{F14.6} bbbbbb 73.445000$$

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 می شود را نشان می دهد.
برای مثال:
$$5.3 \times 10^{234} \xrightarrow{E13.4E3} bbb.5300E + 235$$

فرمت A : برای محدود کردن عبارت های کاراکتری مورد استفاده قرار می گیرد فرم کلی آن AW که W طول میدان عبارت کاراکتری را تعیین می کند.

`WEIGHT 'A10' → bbbbWIEGHT`

برای مثال:

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 (*FORMATS*) N است که N تعداد تکرار این نوع فرمت‌هاست.

`5(F14.6,E12.5E2)`

برای مثال

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

استفاده از فرمت‌ها در خروجی:

`PRINT '(FORMATS)',VAR`

نحوه نمایش

`PRINT '(I7,F7.3,E10.5,A3)',I,R1,R2,CHAR`

برای مثال:

دیگر دستور عبارت‌های 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.

`PRINT (I7,5X,F7.3/TR2,E10.5,T2,A3),I,R1,R2,CHAR`

برای مثال

این دستور ابتدا I را با دامنه ۷ 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 E چاپ 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.

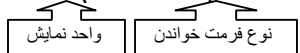
`READ (UNIT=U, FMT=F) VAR1,VAR2,...`

فرم کلی نمایش ورودی



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

فرم کلی نمایش خروجی



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 قبلاً وجود داشته یا قرار است جایگزینی صورت گیرد یا غیره که حالات مختلف آن به قرار زیر است:

`OLD` : فایل قبلاً روی سیستم وجود داشته است.

`NEW` : فایل قبلاً روی سیستم وجود نداشته است.

`REPLACE` : فایل قبلاً 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`

مهمترین گزینه‌ها در `CLOSE` :

IOSTSA=ios : یک متغیر صحیح پیش فرض است که اگر دستور به درستی اجرا شود صفر خواهد بود و در غیر این صورت 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 می شود.

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

'KEEP' : فایل را در حافظه نگه می دارد.

'DELETE' : فایل را از حافظه پاک می کند.

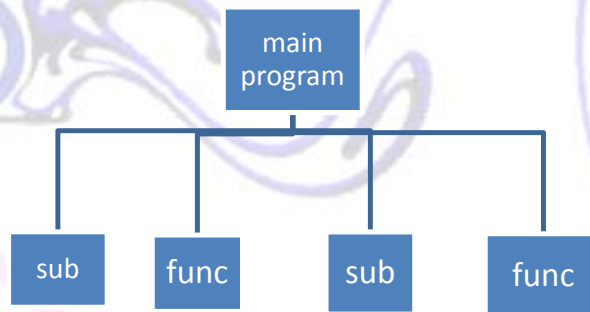
فایلی که **STATUS='SCRATCH'** باز شده است نمی توان **STATUS='KEEP'** ذخیره کرد.

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

تعداد گزینه ها در دستورهای **OPEN , CLOSE** بسیار زیاد است که می توان به **help** فرتن جهت استفاده از دیگر گزینه ها مراجعه نمود.

تابع ها و زیرشاخه ها:

function و **subroutine** از جمله زیر برنامه های داخلی (درونی) زبان فرتن اند. تفاوت مهم بین **function** ها و **subroutine** ها در این است که **function** تنها یک متغیر را برمی گرداند ولی **subroutine** ها می توانند چند متغیر را برگردانند.



یک برنامه می تواند تعداد زیادی **function** و **subroutine** داشته باشد.

function : فرم کلی تعریف **function**

FUNCTION (نام نتیجه) **RESULT** (آرگومان) نام

دستورات فرتن

تعداد نتیجه = نام نتیجه

END FUNCTION نام

FUNCTION SOLVER(INPUT) RESULT(OUTPUT)

برای مثال

دستورات فرترن, REAL::INPUT,OUTPUT

(OUTPUT=INPUT)

END FUNCTION SOLVER

می توان RESULT را حذف نمود و جواب را به وسیله ی خود آرگومان دریافت نمود.

FUNCTION SOLVER(INPUT)

برای مثال

دستورات فرترن

(INPUT=INPUT+S)

END FUNCTION SOLVER

SUBROUTINE : تعریف این دسته از زیر برنامه ها به شکل:

SUBROUTINE (آرگومان های خروجی, آرگومان های ورودی) نام

دستورات فرترن

RETURN → به عنوان کنترل کننده در برنامه می آید و می توان آن را حذف کرد.

END SUBROUTINE نام

نحوه قرارگیری function در برنامه

Program Main

دستورات فرترن

Contains

Functions(تابع ها)

END program Main

نحوه فراخوانی subroutine در برنامه اصلی :

هر subroutine با دستور زیر که در برنامه اصلی می آید می تواند با برنامه اصلی link گردد.

Call subroutine (آرگومان های ورودی و خروجی) نام

فرم کلی

Call subroutine(A,B,C)

مثال:

تعریف متغیرهای ورودی و خروجی در subroutine :

Type,INTENT(RANGE):: نام متغیرها

فرم کلی متغیرها یا آرایه های ورودی و خروجی

مثلا برای SUBROUTINE SOLVER(A,B,C) :

REAL,INTENT(IN)::A متغیرهای ورودی
 REAL,INTENT(OUT)::B,C متغیرهای خروجی
 REAL,INTENT(INOUT)::D متغیرهای هم ورودی و هم خروجی
 REAL,INTENT(OUT),DIMENSION(12,20,50)::B,C آرایه‌های خروجی
 یا REAL,INTENT(OUT)::B(12,20,50),C(12,20,50)

FUNCTION های پر کاربرد :

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

زیربرنامه‌های خارجی !:

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 MODULE، COMMON، PARAM 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 فرتن ۹۰

```
Include 'COMMON.f90'
```

در فرتن ۷۷

```
Include 'COMMON.for'
```

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

تعریف متغیرها

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

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 فرتن است که برای رفع نواقص COMMON توسط سازندگان فرتن 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:

نام مدول MODULE

تعریف متغیرهایی که قرار است به اشتراک گذاشته شود.

نام مدول END MODULE

Module variables

برای مثال:

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

Use variable,only::A,Array

برای مثال:

خاصیتی جالب برای مدول‌ها:

مدول‌ها علاوه بر آنکه 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 (متغیر ورودی و خروجی) نام تابع

(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:

Real(8),public::A,B

متغیرهای عمومی

Integer,private::A,B

متغیرهای خصوصی