

فهرست مطالب

پنج

آغاز سخن

1 فصل یکم: برنامه نویسی با Fortran 90/95

- 1 ..... ۱-۱- پیشگفتار
- 1 ..... ۲-۱- متغیرها
- ۳ ..... ۳-۱- عملیات محاسباتی و تقدم آنها
- ۴ ..... ۴-۱- ساختار برنامه های فرترن
- ۵ ..... ۵-۱- ورودی داده ها
- ۶ ..... ۶-۱- خروجی برنامه
- ۸ ..... ۷-۱- دستورهای تکرار عملیات
- ۱۰ ..... ۸-۱- دستورهای شرطی
- ۱۱ ..... ۹-۱- دستور پرش (Go to)
- ۱۲ ..... ۱۰-۱- آرایه ها (متغیرهای چند بعدی)
- ۱۵ ..... ۱۱-۱- مقدار دهی به متغیرها و آرایه ها
- ۱۶ ..... ۱۲-۱- زیر برنامه (Subroutine)
- ۱۹ ..... ۱۳-۱- پرونده ها
- ۲۲ ..... ۱۴-۱- توابع کتابخانه ای
- ۲۴ ..... ۱۵-۱- ساخت تابع (Function)
- ۲۵ ..... ۱۶-۱- نکاتی در برنامه نویسی
- ۲۶ ..... ۱۷-۱- ترجمه و اجزای برنامه ها

22 فصل دوم: نصب نرم افزار

- 22 ..... ۱-۲- پیشگفتار
- 22 ..... ۲-۲- تواناییها و ویژگیهای نرم افزار
- 28 ..... ۳-۲- نیازمندیهای سیستم

ظفیری، حمید، ۱۳۵۴-

کتاب آموزشی Visual Fortran [ ویژوال فورترن ]:

Fortran PowerStation 4.0 & Compaq Visual Fortran 6.5

[فورترن پاوراستیشن ...] نویسنده: حمید صفاری؛ ویراستار سبد

سعید موسوی ندوشنی. تهران: حمید صفاری، ۱۳۸۳.

پنج، ۱۴۸ ص. مصور.

ISBN 964-06-4577-X ۱۸۰۰۰ ریال

فهرستنويسي بر اساس اطلاعات فپا.

۱. فورترن (زبان برنامه نويسی کامپیوتر). الف. عنوان.

۰۰۵/۱۳۳ ف۹ QA۷۶/۷۳

۷

۵۷۵ کتابخانه ملی ایران

نام کتاب: کتاب آموزشی Visual Fortran

مؤلف: مهندس حمید صفاری

ناشر: مؤلف

لیتوگرافی: رجاء نقشیه

چاپ و صحافی: آرین

نوبت چاپ: ۱۳۸۳ (اول)

شمارگان: ۳۰۰ نسخه

قیمت: ۱۸۰۰۰ ریال

شابک: ۹۶۴-۰-۶۴۵۷۷-X

ISBN: 964-06-4577-X

مرکز پخش: انتشارات سیما دانش تلفن: ۰۵-۱۱۴۶۶۹۶

حق چاپ محفوظ است.

## فصل چهارم : برنامه نویسی با امکانات پنجره و منو

۶۰	..... ۴-۱- پیشگفتار .....
۶۵	..... ۴-۲- کاربرد منوهای پیش فرض .....
۶۷	..... ۴-۳- شرح منوهای پیش فرض .....
۶۸	..... ۴-۴- مدیریت زیر پنجره ها .....
۶۹	..... ۴-۴-۱- ساخت زیر پنجره ها .....
۷۰	..... ۴-۴-۲- آشکار کردن نشانگر موس .....
۷۱	..... ۴-۴-۳- تنظیم رنگ پس زمینه زیر پنجره ها .....
۷۱	..... ۴-۴-۴- تنظیم رنگ متن .....
۷۲	..... ۴-۴-۵- کنترل اندازه و موقعیت پنجره ها .....
۷۳	..... ۴-۴-۶- مرکزیت دادن به یک پنجره .....
۷۴	..... ۴-۴-۷- حذف و یا باقی نگهداشتن پنجره ها .....
۷۵	..... ۴-۵- ویرایش منوها .....
۷۶	..... ۴-۵-۱- حذف یک عنوان منو یا آیتم منو .....
۷۷	..... ۴-۵-۲- درج یک منو با آیتم منو .....
۷۸	..... ۴-۵-۳- الحاق یک منو با آیتم منو .....
۷۹	..... ۴-۵-۴- تغییر خصوصیات آیتم های منو در حین برنامه .....
۷۹	..... ۴-۶- ساخت منوها از ابتدا .....
۸۰	..... ۴-۷- تعریف یک جمعیه توضیحی .....
۸۰	..... ۴-۸- تعریض آیکنهای پنجره اصلی و زیر پنجره ها .....
۸۲	..... ۴-۹- چگونگی استفاده از موس .....

## فصل پنجم : کاربرد پنجره های مکالمه

۸۵	..... ۵-۱- پیشگفتار .....
۸۵	..... ۵-۲- ساخت ظاهر پنجره مکالمه .....
۸۶	..... ۵-۱-۲-۵- درج ابزار .....
۸۷	..... ۵-۲-۲-۵- تنظیم خصوصیات کنترل .....
۸۸	..... ۵-۳-۲-۵- ذخیره پنجره مکالمه .....

۲۸	..... ۴-۲- نصب نرم افزار .....
۲۹	..... ۴-۱-۴-۲- نصب روی دیسک سخت .....
۳۰	..... ۴-۲-۴-۲- نصب روی شبکه .....
۳۰	..... ۴-۳-۴-۲- اجرا از روی CD .....
۳۱	..... ۴-۵- اجرای نرم افزار .....
۳۱	..... ۴-۶- آماده سازی محیط اولیه .....
۳۲	..... ۴-۷-۲- سفارشی نمودن منوهای نرم افزار .....
۳۴	..... ۴-۸-۲- معروفی منوها .....
۳۸	..... ۴-۹-۲- دسترسی به راهنمای نرم افزار .....
	مرور مثالان دایرکتوری

## فصل سوم: مفاهیم پروژه

۴۰	..... ۳-۱- پیشگفتار .....
۴۰	..... ۳-۲- تعریف فضای کار پروژه ای (Project Workspace) .....
۴۶	..... ۳-۳- ساخت و اجرا یک فایل تنها .....
۴۸	..... ۳-۴- ایجاد فضای کار پروژه ای .....
۵۰	..... ۳-۵- انواع پروژه .....
۵۱	..... ۳-۶-۲- مدیریت پروژه .....
۵۱	..... ۳-۶-۳- ۱- درج و حذف فایلها در پروژه .....
۵۲	..... ۳-۶-۳- ۲- ترجمه و اشکال یابی پروژه .....
۵۳	..... ۳-۶-۳- ۳- ساخت و اجرا پروژه .....
۵۳	..... ۴-۶-۳- اشکال زدایی در منطق برنامه (Debug) .....
۵۵	..... ۷-۳- زیر پروژه .....
۵۸	..... ۸-۳- ایجاد یا حذف پیکربندی در یک پروژه .....
۶۰	..... ۹-۳- کاربرد پروژه های بیرونی .....
۶۱	..... ۱۰-۳- نمونه کاربردی .....

## آغاز سخن

فرترن نخستین زبان علمی می باشد که برای کامپیوتر طراحی شده است. این زبان در برنامه نویسی موضوعات علمی و مهندسی کاربرد فراوان دارد. با وجود گذشت بیش از ۳۰ سال از اولین نسخه های این زبان برنامه نویسی، به سبب ارتقاء قابلیت های این نرم افزار توسعه طراحتان آن، همتواره کاملترین زبان علمی و مهندسی به شمار رفته است.

پیشرفت علمی زبان فرترن در دهه های گذشته را می توان از Fortran 77 به Fortran 90 نام برد. نرم افزارهای حمایت کننده این زبان نیز از سیستم عامل Dos به Windows3.1 ارتقاء یافته اند. از جمله نمی توان به ۱.۰ Fortran PowerStation اشاره نمود. اما با وجود چنین پیشرفت هایی، هنوز راحتی کاربرد آنها برویژه در برنامه های گرافیکی میسر نگردید. با به میان آمدن نرم افزار F.P.S.4 (Fortran PowerStation 4.0) توانیهای فرترن پیشرفت چشمگیری یافت. این نسخه سازگار با سیستم عامل های Windows95,98,NT و نگارش های بالاتر می باشد. همچنین شامل تابعهای پیشرفته گرافیکی، کتابخانه های گستره ریاضی و ابزار برنامه نویسی Visual است و به کمک آن می توان پروژه های بزرگی را ایجاد و مدیریت نمود. در این نسخه، امکان برنامه نویسی با زبان های مختلف، استفاده از ابزار Visual و عملگرهای پیشرفته ویندوز نیز وجود دارد. خاطر نشان می سازد، شرکت مايكروسافت این نرم افزار را به عنوان بهترین گزینه برای مهندسان و دانشمندان پیشنهاد نموده است. زیرا افزون بر قابلیت های عنوان شده، از دقت محاسباتی بالا و سرعت اجرای زیادی برخوردار است؛ بطوریکه در سایر نرم افزارهای علمی مانند MATLAB نیز، برای افزایش سرعت اجرای برنامه (حتی تا ۲۵ برابر) برخی از اجزای آن (مانند حلقه ها) را به زبان فرترن می نویسند و از آن به عنوان M file استفاده می نمایند.

نرم افزار F.P.S.4 دارای محیط یکسانی با سایر محصولات شرکت مايكروسافت مانند Microsoft Visual C++ و Microsoft Visual Test می باشد. همچنین توانایی مرتبط نمودن پرونده های این نرم افزارها را به محیط خود دارد. دیگر محصولات مايكروسافت مانند Visual Basic و Visual Basic آسانی می توانند با F.P.S.4 آمیخته شوند و برنامه های کاربردی پیشرفته ای را به وجود آورند.

شایان توجه است که چندی پیش محصول F.P.S.4 و امتیاز آن توسعه شرکت Microsoft از Compaq خریداری شد و با نام Visual Fortran Compaq انتشار یافت. نرم افزار Compaq Visual Fortran در واقع همان F.P.S.4 و با ساختار و قابلیت های مشابه آن می باشد. البته ظاهر نرم افزار تغییرات اندکی داشته است. در این نسخه دسترسی به مثالها از راهنمای نرم افزار قدری دشوار

## فصل ششم : عملیات گرافیکی

۶-۱- پیشگفتار .....	۸۸
۶-۲- ساختار برنامه های گرافیکی .....	۹۰
۶-۳- انواع سیستم مختصات .....	۹۱
۶-۴- تابعهای ترسیمی .....	۹۱
۶-۵- تابعهای نوشتاری .....	۹۲
۶-۶- استفاده از برنامه های گرافیکی آماده فرترن .....	۹۲

## فصل هفتم : کاربرد توابع ریاضی و آماری

۷-۱- پیشگفتار .....	۱۲۷
۷-۲- تابعهای خودساخته .....	۱۲۷
۷-۳- کاربرد تابعهای ریاضی و آماری F.P.S.4 .....	۱۳۰
۷-۴- تابعهای ریاضی F.P.S.4 .....	۱۳۱
۷-۵- تابعهای آماری F.P.S.4 .....	۱۳۵
۷-۶- توابع مخصوص در کتابخانه ریاضی .....	۱۳۸

## پیوست ۱: یک برنامه ساده Visual

## پیوست ۲: برنامه نویسی مختلف با Visual Basic و Fortran

دقت شود دستور Real همان Real\*4 (با دقت معمولی) است که معمولاً چون پیش فرض است صحیح و در غیر این صورت حقیقی (اعشاری) به شمار می روند. با رعایت این اصل دیگر نیازی به نوشتندگی شود.

نکته: اگر بخواهیم کلیه متغیرهای حقیقی استفاده شده در برنامه را دقت مضاعف تعریف کنیم باید در ابتدای برنامه دستور زیر را بنویسیم:

Implicit Double precision (A-H, O-Z)

در این صورت دیگر نیازی به استفاده از دستورهای Real با Real8 نخواهد بود. از آنجا که دقت معمول فرترن مناسب است جز در موارد حساس نیازی به تعریف دقت مضاعف نمی باشد. توجه شود پیش فرض چاپ خروجی برنامه ۶ رقم اعشار است. اگر متغیرها با دقت مضاعف تعریف گردند با ۱۶ رقم اعشار چاپ خواهند شد که خروجی را شلوغ می نمایند. (البته می توان با دستور Format که بعداً توضیح داده خواهد شد هر متغیر را با تعداد رقم دلخواه چاپ نمود. ولی این کار مستلزم صرف وقت و دقت بیشتر در برنامه نویسی است).

### ۱-۳- عملیات محاسباتی و تقدم آنها

در فرترن عملیات جمع، تفریق، ضرب و تقسیم را به سادگی می توان با نمادهای +، -، \* / انجام داد. فقط باید دقت نمود که عملیات توان با نماد " " انجام می گیرد.

تقدیم عملیات در فرترن نیز به ترتیب زیر است:

۱- محاسبه عبارت داخل پرانتزها و توابع کتابخانه ای

۲- توان

۳- ضرب و تقسیم

۴- جمع و تفریق

در زیر برای درک بیشتر تقدیم ها نتیجه چند عبارت نوشته شده است:

$$(10+6)/2=8$$

$$10+6/2=13$$

$$10+6/2^{**}2=11.5$$

نمایش  
۱۱.۵

بطور پیش فرض متغیرهایی که حرف اول نام آنها یکی از حروف I,J,K,L,M,N باشد بعنوان متغیر صحیح و در غیر این صورت حقیقی (اعشاری) به شمار می روند. با رعایت این اصل دیگر نیازی به تعریف متغیرهای صحیح و حقیقی در ابتدای برنامه نیست، مگر آنکه بخواهیم یک متغیری که حرف اول آن یکی از این حروف است را اعداداً متغیر حقیقی (برخلاف پیش فرض) در نظر بگیریم. در این صورت باید در ابتدای برنامه آن متغیر را حقیقی تعریف کنیم (مانند M1 در برنامه بالا). یا بطور عکس یک متغیر حقیقی (بر اساس پیش فرض) را صحیح جلوه دهیم. دقت شود متغیرهای رشته ای که شامل بک کلمه یا جمله می شوند حتی باید در ابتدای برنامه از نوع Char تعریف گردد.

طول متغیرهای رشته ای (متنی) باید بر حسب کاراکتر در جلوی دستور char بعد از علامت \* درج شود. بنابراین متغیری مانند Name (در برنامه بالا) حداقل تا ۱۰ کاراکتر تعریف شده است و چنانچه جمله ای طولانی به آن انتساب یابد فقط تا کاراکتر دهم آن در متغیر ذخیره خواهد شد. چنانچه ظرفیت متغیرهای رشته ای با یکدیگر تفاوت داشته باشد می توان به روش زیر آنها را تعریف نمود:

Char name\*10,txt1\*10,txt2\*15

انتساب یک عبارت به متغیرهای برنامه با از طریق خواندن (Read) و یا با علامت = " انجام می گیرد. چنانچه بخواهیم با این علامت به یک متغیر متنی عبارتی را انتساب بدهیم، باید عبارت را داخل گیوه بنویسیم. مثال:

Char\*10 name

name='Ali'

متغیرهای منطقی (مانند var1) نیز فقط می توانند دو ارزش درست (True) یا نادرست (False) را به خود اختصاص دهند. روش این انتساب در حین برنامه با علامت تساوی مانند زیر انجام می گیرد:

Var1=.True.

### ۰ تعیین دقت متغیرها

بطور پیش فرض کلیه متغیرهای حقیقی مورد استفاده در برنامه با دقت ۶ رقم اعشار در محاسبات استفاده می شوند. برای افزایش دقت محاسبات تا ۱۶ رقم اعشار باید نام متغیرها را در مقابل دستور

(منیز)

Real\*8 a,b,c

قرار داد:

Real\*8

## ۱-۴- ساختار برنامه های فرترن

برنامه های فرترن معمولاً به ترتیب دارای یک نام دلخواه برای برنامه در جلوی دستور program در اولین سطر برنامه، دستورات غیر اجرایی مانند معرفی متغیرها و ثابت ها، دستورات اجزایی مانند ورودی، خروجی و محاسبات می باشند. روند اجرای برنامه نیز با stop متوقف و با end پایان می پذیرد.

program نام برنامه

Real ... معرفی متغیرها (در صورت لزوم)  
Integer ...

Read(\*,\*)a,b,c دستورات اجرایی مانند  
d=b\*\*2+c\*c دستورات اجرایی ورودی و خروجی محاسبات  
Write (\*,\*)d  
stop توقف برنامه  
End پایان برنامه

نکته: دستورات را می توان در یک سطر و پشت سرهم با درج علامت " ; " بین آنها نیز نوشت.  
از نظر سبک نوشنی برنامه دونوع ساختار، یکی قدیمی (فایلهایی با پسوند For) و دیگری جدید (با پسوند F90) برای برنامه های فرترن وجود دارد:

## بروندهای با پسوند For

برنامه های قدیمی نوشته شده فرترن با این پسوند می باشند. چنانچه از این پسوند استفاده شود، رعایت موارد زیر الزامی است:

۱- دستورات باید از ستون هفتم به بعد نوشته شوند. در نرم افزار Fortran PowerStation 4.0 برای راهنمایی کاربر، ستون ششم برونهای با پسوند For به

رنگ سیزد می آید و برنامه نویس باید از ستون هفتم به بعد دستورات را بنویسد.

۲- شماره گذاری برای دستورات (در صورت نیاز) از ستون یکم تا پنجم می باشد.

۳- هر سطر مشکل از ۷۲ ستون است. در نوشتن دستورات طولانی (بیشتر از ۷۲ کاراکتر) برای ادامه خط باید در ستون ششم سطر بعد یک علامت دلخواه گذاشت و ادامه دستور را از ستون هفتم به بعد نوشت.

۴- برای نوشنی جملات توضیحی باید در ابتدای آن سطر (ستون یکم) حرف C را تاب نمود.

## بروندهای با پسوند F90

بروندهای جدید در فرترن بهتر است با این پسوند نوشته شوند. در این برونهای ها موارد زیر را باید مورد توجه قرار داد:

۱- نوشنی دستورات از ستون ۱ تا حداقل ستون ۲۵۵ هر سطر. در صورت داشتن شماره دستور، رعایت حداقل یک فاصله بین شماره و دستور الزامی است.

۲- در صورت ادامه دستور از یک سطر باید در انتهای این سطر علامت & درج نمود و سپس ادامه دستور را در سطر بعدی نوشت.

۳- برای نوشنی جملات توضیحی باید علامت " ! " در ابتدای هر جایی از متن (با رعایت یک فاصله از آخرین حرف) درج نمود.

## ۱-۵- ورودی داده ها

در برنامه های کوچک (یا به هنگام آزمودن صحت اجرای برنامه ها) می توان به متغیرها در داخل برنامه مقدار داد. مثلا:

```
X=2
Y=3.3
D=(x**2+y**2)**0.5
Write (*,*) D
```

چاپ کن D را

```
Stop
END
```

اما با اجرای برنامه ها در چندین بار یا به ازای مقادیر مختلف برای داده های ورودی، نیاز به خواندن آنها از صفحه کلید یا یک فایل می شود. بدین منظور از دستور Read استفاده می شود. بعنوان مثال:

```
Read
Real (*,*) x,y
```

شکل (\*,\*) Read در حقیقت دستور خواندن از صفحه کلید و با فرمت آزاد می باشد. با این دستور می توان مقادیر مختلفی را ( فقط با وجود یک فاصله یا علامت ویرگول بین آنها ) خواند.

چنانچه خروجی برنامه در چندین سطر باشد از آنجا که متغیرها در میدان مربوطه از راست به چپ چیده می شوند نتایج بصورت زیبا و منظم چاپ خواهد شد و داده ها به آسانی از یکدیگر قابل تشخیص خواهند بود.

توجه: برای چاپ خروجی های با دقت مضاعف استفاده از دستور Format: توصیه می شود. زیرا هر یک از خروجی ها با این دقت دارای ۱۶ رقم اعشار می باشند که تشخیص نتایج و تفکیک آنها را دشوار می نماید.

چنانچه خروجیها حداکثر میدان اخصوص یافته به خود را پر کنند برای تشخیص آنها نیاز به یک فضای خالی می باشد. گاهی اوقات نیز نیاز به یک متن کوچک لایه لای خروجی می باشد. این سبک ها و سایر جزئیات دستور Format در زیر شرح داده می شوند:

خروجی (شماره فرمت و...) write  
 { Format(nx, nIm, nFw.d, nAw, nH, nx, /, )}

در این دستور # (ضریب عملگرها) تعداد میدان می باشد. برای نمونه ۵I2 نشانگر دو میدان عدد صحیح با طول ۵ کاراکتر می باشد. عملگرهای مهم نیز عبارتند از:

X: جای خالی

m: خروجی عدد صحیح با طول میدان

Fw.d: خروجی عدد اعشاری با طول میدان w و مقدار اعشار d

Ew.d: خروجی عدد توانی با طول میدان w و مقدار اعشار d

Aw: خروجی رشته ای با طول میدان w

nH کاراکتر که پس از حرف H نوشته می شوند عینا در خروجی و در محل مربوطه چاپ می گردند.

nX قلم ویراشگر را # ستون جلو می برد.

\*: هرچه که داخل گیومه قرار داده شود عینا چاپ می شود.

/: قلم ویراشگر را به سطر بعد می برد. ۳) توان همراه با سمت مثبت می باشد.

: سبب می شود تا در دستور Read بعدی داده های ورودی از ادامه آخرین سطر چاپ شده (نه سطر

جديد) خوانده شوند. مثلا دو دستور زیر باعث می شوند که نشانگر برنامه هنگام اجرا برای خواندن

مقادیر L1 و L2 جلوی عبارت Enter L1,L2: ظاهر شود نه سر سطر بعد.

Write(\*, '(A, \'))' Enter L1,L2:  
 Read(\*, \*)L1,L2

#### ۶-۴- خروجی برنامه

پس از پردازش اطلاعات در داخل برنامه نوبت به چاپ نتایج می شود. این نتایج را می توان روی صفحه نمایش کامپیوتر چاپ نمود یا آنکه در یک فایل بصورت متنی ذخیره کرد. برای چاپ اطلاعات روی صفحه نمایش از دستور Write استفاده می شود. بین متغیر هایی که قرار است چاپ شوند باید علامت ":" قرار داد:

Write (\*,\*) a, b

تکنی: برای آنکه عبارتی عینا چاپ شود باید آن را داخل گیومه قرار داد. مثلا:

write (\*,\*) 'Result', a

Result=5.000000

می توان نتایج برنامه را با ترتیب خاصی مانند مشخص بودن میدان قرار گرفتن اعداد، تعداد اعشار آنها و... چاپ نمود. برای این کار از دستور Format استفاده می شود.

#### ۶-۵- دستور Format

دستور (\*,\*) Write خروجی را مطابق با نوع و دقت تعیین شده برای متغیرها (اعم از ساده یا مضاعف) و با فاصله کافی از یکدیگر چاپ می کند. در واقع علامت ":" دوم داخل دستور write نشانگر فرمت آزاد (یا بدون رعایت بک ترتیب خاص) می باشد. در صورتیکه هدف چاپ خروجی با یک ترتیب خاص باشد باید بجای علامت . یک عدد گذاشت و در سطر دیگری ابتدا شماره و سپس دستور Format را برای آن نوشت (عنوان نمونه :

write (\*,12)M,N,a,b

12 Format(2I3, 2F12.4)

دستور Format بالا سبب می شود تا دو متغیر صحیح اول در یک میدان سه کاراکتری (2I3) و دو متغیر حقیقی بعدی در یک میدان ۱۲ کاراکتری ولی هر یک با ۴ رقم اعشار چاپ شوند: عنوان مثال چنانچه مقادیر M, N, a, b به ترتیب برابر ۳۰۵۱۶، ۱۸۲۵۹۴۶، ۰/۳۰۶۹ و ۳.۵۶۹ باشد چاپ آنها بصورت زیر خواهد بود:

3 51 18.2594 3.569

منظور از علامت : جای خالی می باشد. دقت شود علامم معیز، مثبت و منفی نیز یک کاراکتر محسوب می شوند).

```
Write (*, *) 'sum = ', sum
END
```

در برنامه بالا ۸ عدد بطور پایه خوانده می شوند و در پایان مجموع آنها چاپ می شود.

تکته ۱: چنانچه مقدار افزایش شمارنده منفی باشد حلقه بطور معکوس (کاهشی) اجرا خواهد گردید.

تکته ۲: می توان مقابل دستور Do از درج شماره خط پایان حلقه خودداری نمود. به شرط آنکه در پایان

حلقه عبارت End do را نوشت. مثلا:

```
Do I=1,8
  چند دستور اجرایی
End do
```

#### • دستور while

می توان بجای آنکه یکسری دستور به تعداد مشخصی تکرار شوند آنها را تازمانی که یک شرط

برقرار باشد تکرار نمود. به عنوان مثال:

```
DO while (Delta < 0.1)
  ...
END DO
```

در این چرخه تازمانی که متغیر Delta بزرگتر یا مساوی ۰/۱ شود عملیات تکرار می یابد.

تکته: با دستور Exit می توان از درون یک حلقه خارج شد:

```
DO while (Delta >= 0.1)
  ...
IF (n > 100) Exit
END DO
```

توجه: در فرترن ۹۰ سبک ساده تری نیز وجود دارد که حلقه بی پایان نام دارد. این روش دقیقاً مانند حلقه

است با این تفاوت که نیازی به تعریف شرط در مقابل Do نیست و تنها عامل برای خارج

شدن از حلقه همان دستور Exit می باشد. به عنوان مثال:

```
DO
  ...
IF (s==0) Exit
END DO
```

شایان توجه است که در فرترن می توان حلقه های تودرتو ایجاد نمود. ولی تداخل (قطع کردن حلقه ها)

سبب ایجاد خطأ خواهد شد.

تکته ۱: بجای نوشن دستور فرمت می توان عملگرهای آن را در خود دستور چاپ و در داخل پرانتز

درونو گیومه به شیوه زیر نوشت:

#### • خروجی ها ('عملگرها')

تکته ۲: عملگرهای مختلف دستور Format باید با ویرگول از یکدیگر جدا نوشه شوند.

تکته ۳: برای جلوگیری از نوشن میدانهای اندازه می توان آنها را داخل پرانتز و با ضرب ب تکرار مربوطه قرار داد. مثلا:

14 Format(I4,5x,2(F12.4, 2x,I10))

توجه: دقیقاً مشابه دستور write به هنگام استفاده از دستور Read نیز می توان داده ها را با

سبک (Format) مشخص خواند. البته توصیه می شود برای جلوگیری از اشتباه در خواندن داده ها و

حسابی آنها از فرمت آزاد (Read(\*,\*)) استفاده گردد.

#### • دستور Read و Write داخلی

از این دستورات برای تبدیل کاراکتر به عدد یا بالعکس بسته به نیاز استفاده می شود. شرح این

فرمانها با مثال در بخش ۶-۵ کتاب آورده شده است.

#### ۱-۲- دستورهای تکرار عملیات

##### • دستور DO

برای تکرار یک یا چند دستور پایه ای از فرمان DO با ساختاری مشابه مثال زیر استفاده می شود:

##### شماره دستور پایان

##### متغیر شمارنده حلقه

Do 10 i=L1,L2,m

:

10 Continue

L1 و L2 به ترتیب حد پایین و بالای شمارش و m نیز میزان افزایش شمارنده است. چنانچه این مقدار

برابر ۱ (پیش فرض) باشد نیازی به نوشن آن نیست. در ادامه یک مثال برای کاربرد دستور Do می آید:

Sum=0

Do 10 i=1,8

Write (\*, \*) 'Inter variable'

Read (\*, \*) variable

Sum=sum+variable

:

10 continue

## آغاز سخن

فرترن نخستین زبان علمی می باشد که برای کامپیوتر طراحی شده است. این زبان در برنامه نویسی موضوعات علمی و مهندسی کاربرد فراوان دارد. با وجود گذشت بیش از ۳۰ سال از اولین نسخه های این زبان برنامه نویسی، به سبب ارتقاء قابلیت های این نرم افزار توسعه طراحان آن، همتواره کاملترین زبان علمی و مهندسی به شمار رفته است.

پیشرفت علمی زبان فرترن در دهه های گذشته را می توان از Fortran 77 به Fortran 90 نام برد. نرم افزارهای حمایت کننده این زبان نیز از سیستم عامل Dos به Windows3.1 ارتقاء یافته اند. از جمله نمی توان به ۱.۰ Fortran PowerStation اشاره نمود. اما با وجود چنین پیشرفت هایی، هنوز راحتی کاربرد آنها برویژه در برنامه های گرافیکی میسر نگردید. با به میان آمدن نرم افزار F.P.S.4 (Fortran PowerStation 4.0) با سیستم عامل های Windows95,98,NT و نگارش های بالاتر می باشد. همچنین شامل تابعهای پیشرفته گرافیکی، کتابخانه های گستره ریاضی و ابزار برنامه نویسی Visual است و به کمک آن می توان پروژه های بزرگی را ایجاد و مدیریت نمود. در این نسخه، امکان برنامه نویسی با زبان های مختلف، استفاده از ابزار Visual و عملگرهای پیشرفته ویندوز نیز وجود دارد. خاطر نشان می سازد، شرکت مایکروسافت این نرم افزار را به عنوان بهترین گزینه برای مهندسان و دانشمندان پیشنهاد نموده است. زیرا افزون بر قابلیت های عنوان شده، از دقت محاسباتی بالا و سرعت اجرای زیادی برخوردار است؛ بطوریکه در سایر نرم افزارهای علمی مانند MATLAB نیز، برای افزایش سرعت اجرای برنامه (حتی تا ۲۵ برابر) برخی از اجزای آن (مانند حلقه ها) را به زبان فرترن می نویسند و از آن به عنوان M file استفاده می نمایند.

نرم افزار F.P.S.4 دارای محیط یکسانی با سایر محصولات شرکت مایکروسافت پروونده های این نرم افزارها را به محیط خود دارد. دیگر محصولات مایکروسافت مانند Microsoft Visual C++ و Microsoft Visual Test Microsoft Visual Basic و Microsoft Excel با سانسی می توانند با F.P.S.4 آمیخته شوند و برنامه های کاربردی پیشرفته ای را به وجود آورند.

شایان توجه است که چندی پیش محصول F.P.S.4 و امتیاز آن توسعه شرکت Microsoft از Compaq خریداری شد و با نام Compq Visual Fortran انتشار یافت. نرم افزار Compq Visual Fortran در واقع همان F.P.S.4 و با ساختار و قابلیت های مشابه آن می باشد. البته ظاهر نرم افزار تغییرات اندکی داشته است. در این نسخه دسترسی به مثالها از راهنمای نرم افزار قدری دشوار

۶-۴- تابعهای دیالوگ.	نوشن برنامه فراخواننده پنجه ره مقالمه
۶-۵- شاخص های کنترل.	۶-۵- کنترل ها.

۸۸	
۹۰	
۹۱	
۹۱	

## فصل ششم : عملیات گرافیکی

۱۰۹	
۱۰۹	-۱- پیشگفتار
۱۰۹	-۲- ساختار برنامه های گرافیکی
۱۱۰	-۳- انواع سیستم مختصات
۱۱۲	-۴- تابعهای ترسیمی
۱۱۶	-۵- تابعهای نوشاري
۱۲۱	-۶- استفاده از برنامه های گرافیکی آماده فرترن

## فصل هفتم : کاربرد توابع ریاضی و آماری

۱۲۷	
۱۲۷	-۱- پیشگفتار
۱۲۷	-۲- تابعهای خودساخته
۱۳۰	-۳- کاربرد تابعهای ریاضی و آماری F.P.S.4
۱۳۱	-۴- تابعهای ریاضی F.P.S.4
۱۳۵	-۵- تابعهای آماری F.P.S.4
۱۳۸	-۶- تابع مخصوص در کتابخانه ریاضی

## پیوست ۱: یک برنامه ساده Visual

## پیوست ۲: برنامه نویسی مختلف با Visual Basic و Fortran

۱۴۱	
۱۴۵	

چنانچه با برقرار بودن یک شرط نیاز به اجرای بیش از یک دستور باشد می توان از روش زیر بهره جست:  
 IF (عبارت منطقی) Then  
 دستورات اجرایی  
 End if

می توان به شیوه زیر دستور IF را طوری نوشت که در صورت برقرار بودن شرط اول دستورات دیگری را اجرا کند.

IF (عبارت منطقی) Then  
 دستورات اجرایی اول  
 Else  
 دستور اجرایی دوم  
 END IF

IF ( $a \geq 1$ ) Then  
 $S = a^{**2} - b$   
 Else  
 $S = 0$   
 END IF

مثال:

## ۱-۹- دستور پرش ( Go to )

دستور Go to برای فرستادن کنترل خط به یک سطر مشخص از برنامه (که شامل دستور اجرایی باشد) به کار می رود:

شماره سطر Go to  
 دستور اجرایی شماره سطر

IF ( $a > b$ ) Go to 10  
 write (\*, \*) 'a <= b'  
 Go to 20  
 10 write (\*, \*) 'a < b'  
 20 a = ...

مثال:

## ۱-۸- دستورهای شرطی

به کمک دستورهای شرطی می توان اجرای بخشی از برنامه را متوط به برقرار بودن شرطی انجام داد.  
 ساختار این دستور به صورت زیر است:

## فرمان اجرایی ( یک عبارت منطقی )

عبارت منطقی می تواند یک شرط تساوی یا عدم آن باشد. در زبان فرترن از علائم زیر برای کاربرد شرطها استفاده می شود:

بزرگتر	>	یا	GT.
کوچکتر	<	یا	LT.
نامساوی	==	یا	EQ.

عنوان مثال :

If ( $a < b$ ) z = -1If ( $a == b$ ) z = 0

یعنی اگر متغیر a کوچکتر از b باشد z را برابر -1 قرار داده

یعنی اگر متغیر a با b مساوی باشد z را برابر صفر قرار ده

توجه: همچنانکه در بخش متغیرها نیز عنوان شد، متغیرهای منطقی در ابتدای برنامه باید از نوع Logical تعریف شوند و در حین برنامه فقط ارزش درست (True). و یا ارزش نادرست (False). می تواند به آنها نسبت داده شود.

یعنی اگر متغیر منطقی ارزش True داشته باشد... if یک دستور اجرایی (متغیر منطقی) باشد...  
 یا

یعنی اگر متغیر ارزش نادرست (False) داشته باشد... if یک دستور اجرایی (متغیر منطقی) باشد...  
 می توان چند شرط را با یکدیگر به کمک عاملهای زیر ترکیب منطقی نمود:

AND. معادل "و" به معنی برقرار بودن دو شرط

Or. معادل "یا" به معنی برقرار بودن حداقل یکی از شرط ها

Not. به معنی عدم برقراری شرطی

مثال :

If ( $i < 3$  .and.  $k == 3$ ) s = 0.0

If (Status) a = 1

If (.Not.Status) a = -1

## (۱۰-۱) آرایه ها (متغیرهای چند بعدی)

تاکنون به هر متغیر فقط یک داده اختصاص می‌یافتد. اما متغیرهایی هستند که می‌توانند شامل مقادیر زیادی باشند. به اینگونه متغیرها آرایه گفته می‌شود. در واقع آرایه‌ها مانند ماتریس‌های یک سطحی یا چند سطحی هستند که با توجه به اندیس سطح و ستون مربوطه می‌توانند داده به خود اختصاص دهند.

برای تعریف یک ماتریس سطوحی با یک آرایه یک بعدی یا یک ماتریس چند بعدی با آرایه چند بعدی باید نام آنها را مقابل دستور Dimension قرار داد. سپس باید در طی برنامه به اجزای آرایه مقدار نسبت داد. مقدار دهی می‌تواند بصورت ساده (با علامت =)، با دستور Data و یا از طریق خواندن از درون یک پرونده انجام گیرد. در زیر چگونگی تعریف دو ماتریس A و D بصورت آرایه شرح داده می‌شود:

$$D = \{1 \ 3 \ 5 \ 7\} \quad A = \begin{bmatrix} 5 & 8 & 9 \\ 2 & 4 & 11 \end{bmatrix}$$

Dimension D(4),A(3,2)

D(1)=1

D(2)=3

A(1,1)=5

A(1,2)=8

A(2,3)=11

S=D(3)\*A(2,3)

Write(\*,\*) 'S=' , S

End

آرایه‌ها که در حقیقت متغیرهای چند بعدی می‌باشند بطور پیش فرض بر حسب حرف اول نام خود (مطابق گذشته) جفتی یا صحیح ارزیابی می‌شوند. مگر آنکه بطور مؤکد و در مقابل دستور Real یا... فوار گیرند. همچنین چنانچه آرایه‌ای با دستورهای Integer, real,... تعریف شود نباید قبل از دستور Dimension تعریف شده باشد به عبارت دیگر برای تعریف آرایه فقط باید از یکی از دو روش موجود (قرار دادن جلوی دستور Dimension یا Real و ...) استفاده نمود.

باید توجه نمود بعد مربوط به آرایه‌ها برای اختصاص حافظه لازم به آنها باید در ابتدای برنامه تعریف شود. این بعد چنانچه بطور قطعی مشخص نباشد بهتر است مقداری بیشتر از حد مورد نیاز برای آرایه تعریف گردد تا در صورت نیاز به تعریف درایه‌ای جدید، مشکلی ایجاد نشود. البته تعریف بعد طولانی

(در صورت عدم نیاز) برای آرایه‌ها نه تنها حافظه بیشتری اشغال می‌نماید بلکه در صورت نیاز به اشکال زدایی (Debug) مرور آنها را مشکل تر می‌کند.

چگونگی انتساب داده به اجزای آرایه توسط دستور Data و یا خواندن از پرونده‌ها بعداً شرح داده خواهد شد.

اختصار: در F.P.S.4 چنانچه آرایه‌ای با بعد n تعریف شود و در طی برنامه به بیش از حد تعریف شده برای آن متغیری انتساب داده شود: باعث اشغال و جایگزینی در مسیرهای بعدی حافظه شده و پاسخ برنامه را نادرست می‌سازد. مثال:

Dimension A(5),B(5)

A(1)=...

B(1)=...

⋮

B(5)=...

Do i=1,7

A(i)=1

End do

End

در حلقة برنامه بالا انتساب (6) A به بعد با توجه به اینکه آرایه A پنج عضو بیشتر نمی‌پذیرد سبب اشغال

سایر مسیرهای حافظه (جایگزینی روی خانه‌های ماتریس B) خواهد گردید. متأسفانه F.P.S.4 این ایراد را نمی‌تواند تشخیص دهد و به کاربر اخطار نمی‌دهد. لذا هنگام کار باید به این مسئله دقت فراوان داشت.

• کاربرد آرایه‌ها بدون تعریف ابعاد آنها در ابتدای برنامه.

خیلی از اوقات در شروع برنامه بعد یک آرایه مشخص نیست و در طول برنامه با وارد کردن ابعاد توسط کاربر مشخص می‌شود. در این صورت یک کار معمول و ساده آن است که بعد آرایه در ابتدای برنامه قدری بیشتر تعریف شود تا با اجرایی مختلف (و اختصاص محدوده ای بیشتر به آن) به اشکال برخورد. اما تعریف ابعاد آرایه‌ها بطور دقیق، باعث کاهش حجم اشغال شده حافظه می‌شود. برای تعریف ابعاد دقیق آرایه‌ها در جین برنامه ذو روش وجود دارد:

روش یکم: استفاده از دستور Allocate

در این روش باید آرایه‌های با ابعاد نامشخص را بصورت [allocatable] در ابتدای برنامه تعریف نمود.

بهای هر بعد نامشخص نیز باید علامت ":" قرار داد. در مثال زیر ماتریس یک بعدی D و دو بعدی A

بدون بعد تعریف شده و پس از خواندن بعدهای آنها از صفحه تعیین حدود گردیده‌اند.

Real D[allocatable](::, ::, ::)

برای انجام عملیات بر بخشی از یک آرایه نیز می توان از برش بندی استفاده نمود. مثلا برای نوشتن برای درایه های سوم تا هفتم ماتریس سطری  $D$  با درایه های یکم تا چهارم ماتریس  $F$  می توان نوشت:

$$D(3:7)=F(1:4)$$

در ماتریس های دو یا چند بعدی نیز دقیقا می توان به همین روش بر بخشهايی از ماتریس عملیات انجام داد:

$$A(1:2,3:7)=B(3:4,2:6)$$

برای انتخاب شرطی بخشهايی از آرایه نیز می توان از دستور Where مانند مثال زیر بهره جست:

$$\text{Where } (A < 0) A = 0$$

دستور بالا سبب می شود درایه هایی از آرایه  $A$  که منفی هستند برابر صفر فرض شوند. دستور Where را دقیقا می توان با ساختارهای گوناگون مشابه فرمان If به کار برد. باید افزود عملیات ریاضی پیچیده بر آرایه ها را می توان به کمک برخی تابعهای داخلی و یا کتابخانه های ریاضی فرترن انجام داد.

#### ۱۱-۱- مقدار دهی به متغیرها و آرایه ها

##### • مقدار دهی به کمک دستور Data

در برنامه های کوچک به متغیرها می توان با دستور تساوی (=) و یا از طریق خواندن از صفحه کلید و یا یک فایل مقدار داد. ولی اگر تعداد متغیرها زیاد باشد یا برای آرایه ها می توان با کمک دستور Data عملیات مقدار دهی را خلاصه نمود. این دستور یک فرمان غیراجراگری است و باید در ابتدای برنامه نوشته شود. ساختار این دستورات بصورت زیر است:

مقدار متغیر  $a1, a2, \dots$ , مقدار متغیر دوم، مقدار متغیر اول / متغیر  $a1, a2, \dots$ , متغیر دوم، متغیر یکم Data

مثال:

Data a,b,c,d/1,2,3,4/

دستور بالا به متغیرهای  $a$  تا  $d$  به ترتیب مقدار ۱ تا ۴ را نسبت می دهد.

تکته ۱: اگر چند متغیر هم نوع دارای ارزش یکسانی باشند می توان بجای تکرار مقدار آنها (جز مقدادر منطقی) از یک ضرب استفاده نمود. به عنوان نمونه اگر مقدار متغیرهای  $a, b, c$  برابر  $1/5$  و مقدار متغیر  $d$  برابر ۴ باشد می توان نوشت:

$$\text{Data a,b,c,d}/3*1.5,6/$$

```
Read(*,*),Ld
Read(*,*),L1a,L2a
Allocate (D(Ld))
Allocate (A(L1a,L2a))
```

\* روش دوم: تعریف آرایه درون یک زیر برنامه و قرار دادن ابعاد آن بعنوان ورودی زیر برنامه (با متغیرهای مشترک)

در روش یکم آرایه ها فقط یکبار تعیین ابعاد می شوند. برای حالاتی که آرایه در بخش حلقه ای از برنامه قرار گرفته باشد و چندین بار تعیین ابعاد شود چاره ای جز استفاده از شیوه دوم نیست.. به این روش پس از مطرح نمودن زیر برنامه ها پرداخته خواهد شد.

#### • کاربرد حلقه هضمی برای آرایه ها

با توجه به اینکه آرایه ها می توانند از تعداد زیادی سطر و یا ستون تشکیل شده باشند، لذا برای آسانی کار با آنها می توان از روشی زیر که بصورت حلقه هضمی عملیات خواندن یا نوشتن را انجام می دهند استفاده نمود:

درایه های ماتریس  $D$  بطور سطری خوانده می شوند  
درایه های ماتریس  $A$  ستون به ستون خوانده شده و بطور سطری چاپ می شوند (سرول)  
تکته: به هنگام خواندن آرایه های دو یا چند بعدی، بطور پیش فرض عملیات خواندن ستون به ستون انجام می گیرد. به عبارت دیگر برنامه داده هایی که توسط کاربر در یک سطر بطور پیاپی وارد شده اند را در درایه های ستون اول ماتریس (یا ستونهای بعدی) جای می دهد. همین روش دقیقاً مبنای چاپ آنها نیز می باشد. برای آنکه خواندن یک آرایه چند بعدی (مثلاً ماتریس  $A$  معرفی شده در قبل) متناظر با ماتریس اختصاص داده شده به آن صورت پذیرد می توان از روش ساده زیر استفاده نمود:

```
Do 10 i=1,2
10 Read(*,*),(A(i,j),j=1,3)
```

عملیات چاپ بطور متناظر نیز دقیقاً مشابه دستور بالا و با جایگزینی دستور Write بجای Read انجام می گیرد.

#### • عملیات بر آرایه ها

بطور کاملاً مشابه با متغیرهای ساده می توان آرایه ها را با یکدیگر جمع، تفریق، ضرب و... نمود. فقط باید توجه داشت که آرایه ها از نظر بعد با یکدیگر سازگار باشند. مثال:

$$\begin{aligned} A &= B + C \\ D &= A * B \end{aligned}$$

لکته ۲: اگر متغیرها از نوع کاراکتر باشند ارزش انتساب داده شده به آنها چه از طریق دستور Data و چه با کمک دستور انتساب (=) حتما باید درون گیوه (۱) قرار گیرد.  
برای مقدار دهنده به آرایه ها نوشت نام آنها (بدون ذکر بعد) در مقابل دستور Data بسته می کند.  
بعنوان نمونه آرایه D و A را می توان به روش زیر نیز مقدار دهنده نمود:

Real D(4),A(2,3)

Data D/1,3,5,7/      D={1 3 5 7}

Data (A(i,j),j=1,3),i=1,2)/5,8,9,2,4,11/      A= 
$$\begin{bmatrix} 5 & 8 & 9 \\ 2 & 4 & 11 \end{bmatrix}$$

Data (A(i,j),j=1,3),i=1,2)/5,8,9,2,4,11/      A= 
$$\begin{bmatrix} 7 & 8 & 3 & 4 \\ 5 & 6 & 7 & 4 \\ 3 & 4 & 5 & 6 \\ 2 & 3 & 4 & 5 \end{bmatrix}$$

مقدار دهنده به کمک دستور Parameter.

تاکنون فقط به متغیرها پرداخته شد. مقدار مناسب به آنها همانطور که از نام آنها بر می آید در طی برنامه قابل تغییر است. اما اگر بخواهیم از نامهایی استفاده کنیم که مقدار نسبت داده شده به آنها در طی برنامه قابل تغییر نباشد، باید آنها را مقابل دستور Parameter (در ابتدای برنامه) نوشت. بعنوان مثال:

Parameter (PI=3.141592, E=2.1e5)

کاربرد دستور Parameter فقط جهت اطمینان از عدم تغییر مقدار مربوطه در طی برنامه استفاده می شود و گزنه اجباری در استفاده از دستور نیست و می توان از همان متغیرهای معمولی برای انتساب داده به آنها استفاده نمود.

لکته: اگر متغیرها غیر عددی (مثلا character) باشند باید قبل از دستور Parameter نوع آنها را تعریف نمود.

#### ۱۲-۱- زیر برنامه (Subroutine)

هنگامی که دستورات یک برنامه طولانی شود رفع اشکال و در کمکرد بخشای مختلف آن دشوار خواهد بود. معمولاً مراحل مختلف برنامه های طولانی را در بخشای کوچکتری به نام زیر برنامه (Subroutine) می نویسند. در این صورت برنامه اصلی فقط از چند خط تشکیل می شود که زیر برنامه ها را با دستور Call فراخوانی می کنند. همانطور که هر بخش محاسباتی شامل چند داده اولیه و سپس یک یا چند نتیجه است، در استفاده از زیر برنامه های متغیرهای ورودی و خروجی در داخل پرانتز و مقابل نام زیر برنامه گذاشته می شود. یک زیر برنامه در واقع یک برنامه کامل مستقل است که پس از اتمام برنامه اصلی قرار می گیرد. در ادامه ساختار یک برنامه شامل Subroutine نوشته می شود.

! Main Program

Real ...

Call (متغیرهای ورودی و خروجی) (نام زیر برنامه یکم)

Call (متغیرهای ورودی و خروجی) (نام زیر برنامه دوم)

End

Subroutine (متغیرهای ورودی و خروجی) (نام زیر برنامه یکم)

Real ...

Return

End

Subroutine (متغیرهای ورودی و خروجی) (نام زیر برنامه دوم)

Real ...

Return

End

در زیر یک مثال کوچک برای توضیح زیر برنامه آورده شده است:

```
read(*,*) a,b,c
call F(a,b,c,Result)
write(*,*) Result
end
```

```
Subroutine F(x1,x2,x3,x4)
c1=(x1+x2)**2
c2=(x1+x3)**2
c3=(x2+x3)**2
x4=(c1+c2+c3)**2
return
end
```

در برنامه بالا متغیرهای a,b,c,Result به ترتیب به متغیرهای X1,X2,X3,X4 زیر برنامه نسبت

داده می شوند. در واقع فقط یکی بودن ترتیب و نوع متغیرهای زیر برنامه در دستور Call و Subroutine مهم است و تفاوت نام اهمیتی ندارد. از مزایای این ویژگی در اختیار قرار دادن Subroutine ها به سایر کاربران بدون اعمال هیچگونه تغییر در برنامه اصلی آنها و فقط با رعایت ساختار داده های ورودی زیر برنامه می باشد.

خطای نشان می گردد در ارتباط با استفاده از Subroutine متغیرهای زیر برنامه کاملاً مجزا از متغیرهای برنامه اصلی هستند، حتی اگر نام آنها یکسان باشد. چنانچه خواسته شود بعضی از متغیرها در برنامه اصلی با زیر برنامه دارای ارزش یکسان باشند باید با دستور Common هم در برنامه اصلی و هم در زیر برنامه آنها را مشترک نمود. این دستور پیش از دستورات اجرایی نوشته می شود.

Common  
در ادامه ساختار یک برنامه شامل

- روش دوم کاربرد آرایه ها بدون تعریف ابعاد آنها در ابتدای برنامه آرایه ها را می توان درون یک زیر برنامه قرار داد و ابعاد آنها را بعنوان ورودی زیر برنامه (یا متغیرهای مشترک) در نظر گرفت. مثال:

```
Write(*,!(A,))' Enter L1,L2:'
```

```
read(*,*)L1,L2
```

```
Call Sub1(L1,L2,S)
```

```
:
```

```
Write(*,*)S='s
```

```
end
```

```
subroutine Sub1(L1,L2,S)
```

```
real A(L1,L2)
```

```
:
```

```
S='s
```

```
Return
```

```
End
```

در پایان این بخش باید افزود که چنانچه زیر برنامه ها زیاد باشد یا آنکه بخواهیم دیگران بدون هیچ تغییری در برنامه خود از آنها استفاده نمایند می توان آنها را درون یک فایل فرترنی دیگر وجود آنها از برنامه اصلی نوشت. کاربر یا هر شخص دیگر می تواند فقط با افزودن این فایل به پروژه خود و فراخوانی نام زیر برنامه ها (با رعایت ساختار ورودی و خروجی) از آنها استفاده نماید. این کار دقیقاً مانند نوشت زیر برنامه ها در پایان برنامه اصلی می باشد.

### ۱۳- پرونده ها

چنانچه اطلاعات ورودی برنامه یا نتایج خروجی آن زیاد باشد می توان برای آسانی کار خواندن اطلاعات یا چاپ خروجی از پرونده (File) استفاده نمود. اطلاعات را می توان به کمک نرم افزار برنامه نویسی در یک فایل جدید ریخت و سپس فایل را ذخیره نمود. راه دیگر برای ساخت فایل اطلاعات استفاده از محیط ویندوز است. بدین منظور باید روی صفحه ویندوز کلیک راست نمود و گزینه New و گزینه <sup>۱</sup> Text Document را انتخاب کرد. با این کار یک فایل متنی خالی ایجاد خواهد شد. پس دوبار کلیک روی این فایل نرم افزار Notepad اجرا خواهد شد و می توان اطلاعات را در آن محیط وارد کرد و با نام دلخواه ذخیره نمود. پرونده لازم برای خواندن اطلاعات حتی می تواند خروجی یک نرم افزار دیگر باشد، ولی در هر صورت باید با فرمت متنی باشد. اکثر فایلها با این فرمت دارای پسوند .txt می باشند. البته چنانچه مانند پرونده ای از نوع متنی باشد حتی نیازی به داشتن پسوند .txt برای آن نیست.

```
Real A(3,3)
```

```
Common A,x,y
```

```
:
```

```
Call f(c1,c2,s)
```

```
:
```

```
End
```

```
Subroutine f(c1,c2,s)
```

```
Real A(3,3)
```

```
Common A,x,y
```

```
:
```

```
Return
```

```
End
```

دقت شود تعداد، تیپ و دقت متغیرها یا آرایه هایی که با Common بین برنامه اصلی و زیر برنامه مشترک تعریف می گردند باید در برنامه اصلی و زیر برنامه یکی باشد، اما یکی بودن نام آنها الزامی نیست. همچنین آرایه ها را می توان بعنوان متغیرهای Subroutine (البته بدون درج حدود آنها) مورد استفاده قرار داد. آرایه های مشترک باید در ابتدای برنامه اصلی و زیر برنامه ها مطابق معazel تعریف شوند و سپس بدون درج حدود در دستور Common قرار گیرند. یا اینکه فقط یکبار (با درج حدود) در دستور Common تعریف شوند. نکته قابل توجه دیگر آنست که نمی توان یک متغیر را هم در ورودی داد و هم با دستور Common مشترک تعریف نمود.

چنانچه چندین زیر برنامه وجود داشته باشد، متغیرهای مشترک بین برنامه با هر زیر برنامه را می توان با نوشت یک نام بلوک داخل علامت / / دسته بندی نمود.

نام دلخواه برای بلوک  
Common / blok1 /

نام متغیرهای مشترک بین برنامه اصلی با زیر برنامه ۱ /

Common / blok2 /

در صورت استفاده از این روش باید برای هر زیر برنامه دستور Common مربوط به آن و برای برنامه اصلی همه دستورهای Common تعریف شده در زیر برنامه ها نوشته شود. در برنامه های طولانی یا با تعداد زیاد متغیر و زیر برنامه بهتر است برای پرهیز از تکراری تعریف، کلمه دستورهای Common داخل یک فایل نوشته شود و در ابتدای برنامه اصلی و زیر برنامه ها به کمک دستور زیر به آنها اضافه گردد.

نام و مسیر فایل محتوی دستورهای Common

تکته ۳: معمولا برای راهنمایی کاربر در فایل داده ها سطرهایی نوشته می شود. برای آنکه این سطرهای توسط برنامه اصلی خوانده نشوند و برنامه از آنها رد شود کافی است برای هر سطر توضیحی بکار دستور `(*، شماره پرونده) read (بدون ورودی)` نوشته شود.

تکته ۴: همانطور که در برنامه بالا نیز نوشته شده است فایل خروجی را می توان به سادگی و با همان روش معمولی ساخت. با هر بار دستور `(*، شماره پرونده) write` می توان نتایج و یا سطر خالی در فایل خروجی ایجاد نمود.

تکته ۵: شماره پرونده در واقع یک عدد موقت برای تفکیک پرونده ها از یکدیگر است. تکته ۶: دقت در ترار دادن صحیح شماره پرونده برای فرمانهای `read` و `write` بسیار مهم است. خیلی از کاربران، گاه در حین برنامه به اشتباه بجای خواندن اطلاعات از پرونده با دستور `(*، شماره پرونده)`

`Read [ از دستور (*,*) Read ] که برای ورود داده ها از صفحه کلید است استفاده می کنند. این کار باعث می شود بجای خواندن داده ها از پرونده، نشانگر موس روی صفحه نمایش مشکی رنگ ظاهر شود و منتظر وارد کردن مقدار از صفحه کلید گردد. این اشتباه هنگام چاپ خروجی ها نیز می تواند رخ دهد.`

از این رو بررسی برنامه برای اطمینان از عملکرد صحیح آن نیاز است.

تکته ۷: چنانچه نوع (اعم از صحیح، حقیقی یا رشته ای) یا تعداد متغیرها در فایل ورودی با تعداد متغیرهایی که توسط فرمان `read` خوانده می شوند کوچکترین تفاوتی داشته باشد، باعث ایجاد اشکال کلی یا جزئی در عملکرد یا صحت برنامه خواهد شد. مثلا چنانچه هدف خواندن یک داده حقیقی باشد و به اشتباه یک داده رشته ای وارد شده باشد برنامه با ایراد کلی مواجه خواهد شد. با آنکه داده ورودی بصورت حقیقی (با اعشار) باشد و به اشتباه به یک متغیر صحیح نسبت داده شود به دلیل حذف اعشار نتایج برنامه صحیح نخواهد بود. در صورت بروز خطأ در اجرای برنامه باید آن را با اجرای سطر به سطر اشکال زدایی کرد تا ورودی های هر دستور `Read` از پرونده با مقادیر نسبت داده شده به متغیرها در برنامه مقایسه شود و محل ایراد مشخص گردد.

#### • بستن پرونده (Close)

معمولا نیازی به بستن پرونده ها نمی باشد و با اتمام برنامه بطور خودکار آنها بسته می شوند. در صورت لزوم (برای کاهش حجم اشغال شده حافظه) پرونده را می توان با دستور `(شماره پرونده) Close`

بست. مثال:

`Close(4)`

پرونده شماره ۴ را می بندد

برای فراخوانی اطلاعات از یک پرونده از پیش موجود باید دستور اجرایی زیر را نوشت:

`(نام پرونده از پیش موجود = File، شماره پرونده)`

`Read (*, شماره پرونده)`

برای چاپ خروجی روی یک پرونده نیز می توان از دستورات زیر استفاده نمود:

`(نام دلخواه برای فایل نتایج = File، شماره پرونده)`

`Write (خروجی ها = *, شماره پرونده)`

برای مثال فرض شود که فایل متغیر زیر قبل از `Data.txt` ساخته شده است. این فایل شامل سطرهای زیر است:

`Data.txt`

`Data for Example 1`

5  
2 9 7  
4

اینک برای استفاده از اطلاعات فایل `Data.txt` باید برنامه ای بصورت زیر نوشته شود.

`Open (1, File = 'Data.txt')`

`(برنامه دنبال فایل جدید Output.txt) می گردد و چون از قبل موجود نیست (ساخته شود آن را ایجاد می کند.)`

`Read (1, *)`

این سطر باعث می گردد که یک خط در عمل خوانده نشود.

`Read (1, *) N`

این سطر باعث می شود که مقدار  $N$  به  $\text{N}$  تعلق گیرد.

`Read (1, *) a, b, c`

این سطر باعث می شود که به  $a, b, c$  به ترتیب اعداد ۴، ۹ و ۷ تعلق گیرد.

`D = a*b*c`

`Write (2, *) 'D= ', D`

`END`

تکته ۱: با هر بار استفاده از دستور `(*، شماره پرونده) read` یک سطر از فایل اطلاعات خوانده می شود.

در واقع با فرمت آزاد هر بار خواندن اطلاعات از سطر بعد انجام می گیرد.

تکته ۲: در فایل اطلاعات برای تفکیک داده ها وجود حداقل یک فضای خالی بین آنها ضروری است.

INT(a)	عدد حقیقی a را با حذف قسمت اعشاری آن به عدد صحیح تبدیل می کند.
REAL(N)	عدد صحیح N را به نوع حقیقی تبدیل می کند. با این کار عدد N امکان دریافت اعشار می یابد.
SIN(a),COS(a),TAN(a),COTAN(a)	به ترتیب سینوس، کسینوس، تانزان و کاتانزان عدد a را محاسبه می نمایند.
ASIN(a),ACOS(a),ATAN(a)	به ترتیب معکوس سینوس، کسینوس و تانزان عدد a را محاسبه می نمایند.
SINH(a),COSH(a),TANH(a)	به ترتیب سینوس، کسینوس و تانزان هیپربولیک عدد a را محاسبه می نمایند.
EXP(a)	هم ارز است با $e^a$
Log(a)	هم ارز است با Ln(a)
Log10(a)	هم ارز است با Log(a)
SQRT (a)	هم ارز است با $\sqrt{a}$
	توضیح: برای محاسبه ریشه n ام عدد a باید نوشت: $a^{**}(1/n)$
MAX(a,b,...)	حاصل این تابع بیشینه اعداد ورودی آن خواهد شد.
MIN(a,b,...)	حاصل این تابع کمینه اعداد ورودی آن خواهد شد.
MOD(a,b)	با قیمانده تقسیم صحیح a بر b را محاسبه می کند.
CALL RANDOM(a)	به عدد a یک مقدار تصادفی بین ۰ تا ۱ نسبت می دهد.
CALL GETTIM(Ih,Im,Is,I100th)	زمان را با دقت صدم ثانیه دریافت می کند. پس از اجرای این دستور متغیرهای Ih,Im,Is,I100th به I100th,Is,Im,Is خواهد شد.
	ترتیب برابر ساعت بین (۱ تا ۲۴)، دقیقه، ثانیه و صدم ثانیه ( بصورت اعداد صحیح ) می باشد.
CALL SETTIM (Ih,Im,Is,I100th)	چنانچه متغیرهای صحیح Ih,Im,Is,I100th دارای مقدار باشند ساعت جدید را ثبت می کند.
GETDAT (Iyr,Imon,Iday)	پس از اجرای این دستور سال، ماه و روز در متغیرهای Iyr,Imon,Iday بخواهد شد.
SETDAT (Iyr,Imon,Iday)	مقادیر سال، ماه و روز جدید روی دستگاه ثبت خواهد شد.
	توجه: واحد اندازه گیری زاویه در تابعهای ریاضی رادیان می باشد.
	شایان توجه است در فرترن تابعهای های ریاضی پیشرفته و فراوانی وجود دارند که برای استفاده باید با دسترسی دادن برنامه به کتابخانه های ریاضی و به کمک دستور Call آنها را فراخوان نمود. در فصل هفتم کتاب بطور مفصل به این کار پرداخته خواهد شد.

## ۰ از سرگیری خواندن اطلاعات در یک پرونده

چنانچه پرونده ای در حال استفاده باشد و خواسته شود که کنترل خط به ابتدای آن فرستاده شود تا خواندن اطلاعات دوباره از ابتدای آن انجام پذیرد از دستور زیر استفاده می شود:

## (شماره پرونده) Rewind

اگر هدف برگرداندن کنترل خط به یک یا چند سطر قبل در فایل داده ها باشد می توان دستور زیر را به تعداد لازم اجرا کرد:

## (شماره پرونده) Backspace

در پایان این بخش باید افزود، پرونده ها کمک شایانی به برنامه نویسی منی کنند و برای اطلاعات ورودی و یا خروجی با حجم زیاد چاره ای جز استفاده از آنها نیست. فقط باید به هنگام برنامه نویسی و یا ساخت فایل داده ها دقت کافی اعمال شود تا عمل انتساب داده از پرونده به متغیر متناظر با آن در برنامه به درستی انجام گیرد.

## ۱۴-۱- توابع کتابخانه ای

بنظور ساده سازی برنامه نویسی تابعهای زیادی توسط زبان فرترن تعریف شده اند که کاربر فقط با نوشتن نام آنها می تواند از آنها استفاده نماید. بعنوان مثال در برنامه زیر از تابع LOG برای محاسبه لگاریتم طبیعی بهره گرفته شده است:

```
a = 2.718282
write(*,*) 'Log(a) = ',LOG(a)
End
```

نتیجه: Log(a) = 1.000000

در جدول (۱-۲) برخی از تابعهای مهم و ورودی آنها آورده شده است.

## جدول (۱-۲)- توابع کتابخانه ای مهم در فرترن

ABS(a)	قدر مطلق عدد a را محاسبه می نماید.
AINT(a)	قسمت اعشاری عدد a را حذف می نماید ولی نتیجه همچنان یک عدد حقیقی است.
ANINT(a)	عدد حقیقی a را گرد می نماید.
NINT(a)	عدد حقیقی a را گرد می نماید و نتیجه یک عدد صحیح(با حذف اعشار) خواهد شد.

## ۱-۱۶- تکاتی در برنامه نویسی

## • جمع افزایشی

برای جمع افزایشی چندین داده می‌توان به روش مشابه مثال زیر اقدام نمود:

```
Sum=0
Do 10 i=1,n
10 Sum=Sum+A(i)
```

## • ضرب متواالی

برای ضرب پیاپی چندین داده در یکدیگر از روش مشابه مثال زیر استفاده می‌شود:

```
P=1
Do 10 i=1,n
10 P=P*A(i)
```

## • اتصال داده های کاراکتری

مانظور که پیشتر گفته شد انتساب عبارت به متغیرهای کاراکتری یا از طریق دستور Read و یا با علامت "=" صورت می‌پذیرد. برای اتصال دو یا چند متغیر (یا داده) کاراکتری به یکدیگر می‌توان از عملگر "//" استفاده نمود.

مثال ۱: اتصال دو کلمه

```
Character name1*10, name2*10, text*20
name1='Ali'
name2='Rezaee'
text=name1//name2
Write(*,*),text
```

خروجی: Ali Rezaee

از آنجا که در برنامه بالا متغیرهای کاراکتری name1 و name2 هر کدام دارای میدان ۱۰ تایی هستند، لذا هنگام اتصال آنها مانده اشغال نشده میدان name1 باعث ایجاد یک فاصله بین دو ارزش خواهد شد: اگر بخواهیم مانده اشغال نشده میدان یک متغیر کاراکتری را حذف نسائیم بطوریکه فقط محدود به کاراکرهای اشغال شده باشد، باید از دستور Trim استفاده کنیم. در برنامه پیش اگر متغیر Text به شیوه زیر تعریف شود خروجی بطور مناسب و با رعایت فقط یک فاصله بین دو کلمه خواهد بود:

```
Text=trim(name1)//' '//trim(name2)
```

خروجی: Ali Rezaee

مثال ۲: خواندن یک فایل و ساخت فایل خروجی با افزودن پسوندی بر نام آن

```
Character*15 filein,fileout
Write(*,'(A,/)') Inter File name:'
Read(*,*),filein
open(1,file=filein)
fileout=Trim(filein)//'.out'
open(2,file=fileout)
end
```

## ۱-۱۵- ساخت تابع (Function)

برای انجام محاسبات یا عملیاتی که در فرترن یا کتابخانه های ریاضی و آماری آن تابعی وجود ندارد می‌توان به دو روش زیر تابع مربوطه را تعریف کرد و مورد استفاده قرار داد.

روش یکم: تعریف تابع یک جمله‌ای

آسانترین روش تعریف تابع، نوشتن رابطه مربوطه پس از دستورات غیراجرا (مانند Data Integer و ...) و پیش از دستورات اجرایی (مانند Read و ...) می‌باشد. در کاربرد تابع یک جمله‌ای بهتر آن است که نام تابع با نام تابعهای داخلی فرترن یکی نباشد. به مثال زیر توجه نمایید:

```
real a,b,c
f(x,y,z)=(x+y+z)**2
read(*,*),a,b,c
write(*,*),f(a,b,c)
end
```

تکته: مانظور که در مثال بالا مشاهده می‌گردد، متغیرهای بکار گرفته شده برای تابع فقط از نظر ترتیب فرار گیری مهم هستند و تغیر نام آنها در کاربرد مشکلی ایجاد نمی‌نماید. یعنی با کاربرد تابع بصورت f(a,b,c) متغیر c=z , b=y , a=x قلمداد می‌شود.

## روش دوم: تعریف تابع به کمک دستور Function

چنانچه تابع تعریف شده توسط کاربر طولانی باشد، بهتر است بصورت یک Function تعریف شود. برای این کار تابع با دستور Function پس از برنامه اصلی نوشته می‌شود. فراخوانی تابع در متن برنامه اصلی فقط با نوشتن آن صورت می‌پذیرد. مثال:

```
read(*,*),a,b,c
write(*,*),F(a,b,c)
end
```

```
Function F(x,y,z)
c1=(x+y)**2
c2=(x+z)**2
c3=(y+z)**2
F=(c1+c2+c3)**2
End
```

تکته: نام تابع درون زیر برنامه Function باید بدون درج متغیرهایش استفاده شود. یعنی کاربرد F(x,y,z) بجای F سبب ایجاد خطأ خواهد گردید.