ساختار برنامه فرترن:

برنامه فرترن دارای ساختار زیر است:

```
PROGRAM program-name
 IMPLICIT NONE
 [specification part]
 [execution part]
 [subprogram part]
END [ PROGRAM  program-name ]
```

```
PROGRAM program-name
 IMPLICIT NONE
 [specification part]
 [execution part]
 [subprogram part]
END [PROGRAM  program-name]
```

نکات مهم:

- مندرجات درون کروشه ها [ ] اختیاری–انتخابی می باشند.
- در فرترن حروف کوچک و بزرگ فرقی نمی کند.
- قرار دادن END در انتهای برنامه الزامی است.
- برای نوشتن توضیحات در برنامه از علامت ! استفاده کنید.
- اگر اسپل یک دستور درست نوشته شود، کلمه آبی می شود، مابقی نوشته ها سیاه بوده و توضیحات بعد از علامت !، سبز رنگ است

جملات توضیحی (Program Comments)

- یک جمله توضیحی در فرترن ۹۰با علامت تعجب شروع شده (!) و کلیه عبارات نوشته شده تا پایان خط به منزله جمله توضیحی در نظر گرفته می شود و مفسر آنها را نادیده می گیرد.

- معمولاً در ابتدای برنامه، در معرفی متغیرهای برنامه و همچنین نحوه عملکرد زیرروال یا توابع از جملات توضیحی استفاده می گردد.

```
1 PROGRAM roots
2 !*************************************************************
3 !  Purpose:
4 !    This program solves for the roots of a quadratic equation of the
5 !    form a*x**2 + b*x + c = 0.  It calculates the answers regardless
6 !    of the type of roots that the equation possesses.
7 !*************************************************************
```

تدوام خطوط (Continuation Lines)

- در بسیار موارد دستورات فرترن به ویژه هنگام محاسبه برخی روابط ریاضی طولانی به هیچ عنوان در یک خط جا نمی گیرد. در چنین موردی برنامه نویس ادامه دستور خود را به خط یا خطوط بعدی موکول می کند. برا یاین کار از علامت & در انتهای جمله استفاده می شود. مفسر فرترن با مشاهده این نشان ادامه دستور را در اولیه خط غیر خالی دنبال می کند.

- توجه: قرار دادن علامت & در پایان جمله توضیحی به معنی ادامه آن در خطوط بعدی نمی باشد و با خطای تفسیر مواجه می شوید.

```
28
29 ! Calculate discriminant
30 delta = b**2 - &
31                4. * a * c
32
```

الفبای فرترن (Fortran Alphabets)

فرترن تنها از کاراکترهای زیر استفاده می کند:

- حروف(Letters):

```
A B C D E F G H I J K L M N O P Q R S T U V W X Y Z
 a b c d e f g h i j k l m n o p q r s t u v w x y z
```

- ارقام(Digits):

```
0 1 2 3 4 5 6 7 8 9
```

- کاراکترهای ویژه(Special Characters):

```
space ' " ( ) * + - / : = _ ! & $ ; < > % ? , .
```

ثابت ها در فرترن (Constants)

ثابت در فرترن ۹۰ داده ای است که مقدار آن قبل از آنکه برنامه اجرا شود مشخص شده و مقدارش در حین اجرای برنامه تغییر نمی کند.

انواع مختلف ثابت ها شامل:

- صحیح (INTEGER)
- حقیقی (REAL)
- اعداد مختلط (COMPLEX)
- منطقی (LOGICAL)
- رشته ای (STRING)

ثابت صحیح (Integer Constants)

ثابت صحیح: هر عددی است که دارای ممیز نیست.

- **ساده ترین نحوه نمایش آنها به صورت اعداد صحیح بی/با علامت می باشد.**

1520 0 -9999 +453

- **قرار دادن نماد مثبت برای اعداد صحیح مثبت اختیاری است**

ثابت حقیقی (Real Constants)

ثابت حقیقی: به منظور نمایش اعداد حقیقی به کار می رود و به دو صورت نشان داده می شود:

- فرمت معمولی اعداد اعشاری(decimal form):

در فرم معمولی عدد می بایست دارای نقطه اعشار باشد که با علامت "." نشان داده می شود

مثال:

0.07	25.	25.0	.0	-0.624

- فرمت نمایی(decimal and exponential):

فرم نمایی به صورت یک عدد با نقطه اعشار، حرف E و توان عدد به صورت پشت هم است.

مثال: -0.0015➔-1.5E-4 412.0 ➔ 4.12E+2

نکته: به منظور افزایش دقت به جای E از D می توان استفاده نمود.

مثال: -0.0015➔-1.5D-4 412.0 ➔ 4.12D+2

ثابت منطقی (logical Constants)

یک ثابت منطقی می تواند یکی از مقادیر .TRUE یا .FALSE را دارا باشد.

تذکر: دقت کنید که نقطه های اطراف .TRUE یا .FALSE ضروری می باشد.

- ثابت های منطقی به ندرت مورد استفاده قرار می گیرند.

ثابت کاراکتری - رشته کاراکتری
(Character Constants-Character String)

نوع داده کاراکتری شامل رشته هایی از کاراکترهای حرفی- عددی است.

ثابت کاراکتری: یک رشته از کاراکترهای محصور شده در تک گیومه (‘) یا گیومه دوگانه (“) است.

- تعداد کمینه کاراکترها در یک رشته، یکی است.

مثال: ‘()()’ “#$%^” ‘John Dow’ “abc”

ثابت کاراکتری - رشته کاراکتری
(Character Constants-Character String)

طول رشته: تعداد کاراکترهای محصور شده بین تک گیومه یا گیومه دوگانه است.

مثال:	‘( )( )’	“#$%^”	‘John Dow’	“abc”
طول رشته:	۶	۴	۸	۳

- رشته کاراکتری دربردارنده یک تک گیومه (گیومه دوگانه) می تواند توسط گیومه دوگانه (تک گیومه) محصور گردد.
- مثال: “ ‘This is a test!’ “ ‘ “This is a test!” ’

معرف ها (Identifiers)

معرف ها: نام هایی هستند که برای مشخص کردن اسم برنامه، ثابت ها، متغیر ها و دیگر موارد در یک برنامه به کار می روند.

- معرف ها در فرترن ۹۰شامل ۱تا ۳۱ نویسه حرفی-عددی می باشد و حتما باید با یک حرف آغاز گردد.
- منظور از نویسه حرفی-عددی شامل ۲۶حرف زبان انگلیسی(A,B,C,...)، ارقام (1,2,3,...)و علامت زیرخط(_) است.
- مثال:

معرف های ناصحیح	معرف های صحیح
1X	X1
Gamma ray	GammaRay
GammaRay 1	GammaRay
Gamma-ray	Gamma_ray

معرف ها (Identifiers)

- معرف ها در فرترن ۹۰ نسبت به حروف کوچک و بزرگ حساس نیستند.
  مثال: کلمات name، Name، NAME و nAME همگی در فرترن ۹۰ یکسان هستند.

- به منظور زیباتر شدن برنامه به شدت توصیه می شود از قرار داد نوشتن کلمات کلیدی با حروف بزرگ و استفاده از حروف کوچک برای نوشتن معرف ها استفاده شود.

- در فرترن ۹۰هیچ اسم رزرو شده ای برای زبان برنامه نویسی در نظر گرفته نشده است. با این حال، به علت احتمال خطای اجرای برنامه ناشی از شباهت اسامی و دستورات، توصیه می شود از این امر اجتناب گردد.
  مثال: از کلمات PROGRAM، END و... می توان به عنوان معرف استفاده کرد. اما به هیچ عنوان توصیه نمی شود

اعلان یا معرفی (Declarations)

- متغیر ها: مکانی در حافظه است که مقدار آن در طول اجرای برنامه تغییر می کند.
- به منظور اعلان(معرفی) متغیرها به صورت زیر عمل می شود:

نام متغیر (list) :: نوع متغیر(type-specifier)

- نوع متغیر یکی از کلمات INTEGER، REAL، COMPLEX، LOGICAL و یا CHARACTER است.
- نام متغیر نیز از قانون نام معرف پیروی می کند.

مثال:

```
INTEGER   :: i
REAL      :: velocity
CHARACTER :: name
```

ویژگی پارامتر(The PARAMETER Attribute)

- پارامترها کمیت هایی هستند که مقدارشان ثابت بوده و در طول اجرای برنامه تغییر نمی کند.
- به منظور اعلان(معرفی) پارامتر به صورت زیر عمل می شود:

```
type, PARAMETER :: name = value
```

- نوع (type) یکی از کلمات INTEGER، REAL، COMPLEX، LOGICAL و یا CHARACTER است.
- نام پارامتر نیز از قانون نام معرف پیروی می کند.

مثال:

```
REAL, PARAMETER :: m_e = 9.11E-31   ! electron mass (kg)

REAL, PARAMETER :: h   = 6.63E-34   ! Planck constant (j/s)
```

مقدار دهی اولیه متغیرها (Variable Initialization)

سه روش قابل استفاده برای مقدار دهی اولیه متغیرهای موجود در برنامه فرترن وجود دارد:

۱) مقدار دهی در دستور اعلان نوع (Initialization)

- مقدار اولیه متغیرها را می توان هنگام دستور اعلان نوع و قبل از اجرای برنامه مشخص نمود:

```
type :: var1= value, [var2= value , ...]
```

مثال:

```
INTEGER  ::  loop = 10

REAL     ::  time = 0.0 , distanse = 512.0
```

مقدار دهی اولیه متغیرها (Variable Initialization)

۲) مقدار دهی توسط عبارت جایگزینی (Assignment)

- یک عبارت جایگزینی، مقدار عبارت سمت راست تساوی را به متغیر موجود در سمت چپ آن تخصیص می دهد.

```
type :: var1
var1= value
```

مثال:

```
INTEGER :: loop
loop = 10

REAL    :: time
time = 0.0
```

مقدار دهی اولیه متغیرها (Variable Initialization)

۳) مقدار دهی توسط دستور ورودی (Input)

- دستور **READ** می تواند برای مقدار دهی اولیه متغیرها با مقادیر وارد شده توسط کاربر، مورد استفاده قرار گیرد.

مثال:

```
REAL :: time

WRITE(*,*)"Please enter a time:"

READ(*,*) time
```

عملگرها در فرترن (Operators in Fortran)

عملگرها در فرترن به ۴ دسته تقسیم می شوند:

۱) عملگرهای حسابی (Arithmetic Operators)

۲) عملگرهای ربطی (Relational Operators)

۳) عملگرهای منطقی (Logical Operators)

۴) عملگرهای کاراکتری (character Operators) (خارج از موضوع کلاس است)

عملگرها در فرترن (عملگرهای حسابی)

عملگر	معنی
**	توان
*	ضرب
/	تقسیم
+	جمع
-	تفریق

اولویت عمگرهای حسابی:

I. پارانتز (از داخلی ترین پارانتز) (1+2)*3 = 3*3 = 9

II. توان (از راست به چپ) 2**2**3 = 2**8 = 256

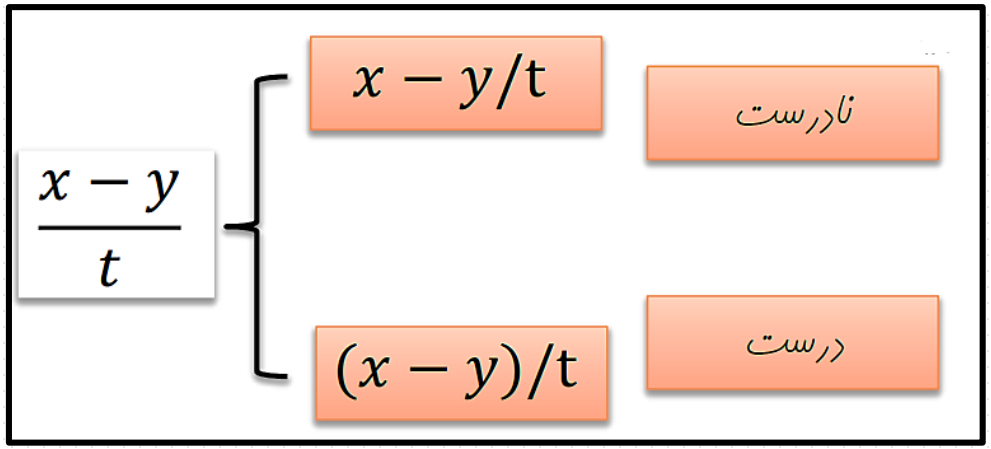
III. ضرب و تقسیم (از چپ به راست) 10/5*2 = 2*2 = 4

IV. جمع و تفریق (از چپ به راست) 10-8+2 = 2+2 =4

عملگرها در فرترن (عملگرهای حسابی)

نکات مهم:

- هیچ دو عملگری نمی توانند در کنار هم قرار گیرند. بنابراین عبارت a*-b غیر مجاز است. عبارت مذکور در فرترن می بایست به صورت a*(-b) نوشته شود.
- در فرترن ضرب ضمنی غیر مجاز است. بنابراین عبارت x(y+z) غیر مجاز است. عبارت مذکور در فرترن می بایست به صورت x*(y+z) نوشته شود.
- فریب عملگر تقسیم را نخورید.



عملگرها در فرترن (عملگرهای حسابی- حساب صحیح)

- **حساب صحیح:** حسابی است که فقط شامل داده های صحیح است. حساب صحیح همواره یک نتیجه صحیح تولید می کند. این قاعده هنگامی که از تقسیم صحیح استفاده می کنیم اهمیت خاصی می یابد.

مثال:

10 / 3	=	3
19 / 4	=	4
4 / 5	=	0
- 8 / 3	=	-2
3 * 10 / 3	=	10
10 / 3 * 3	=	9

عملگرها در فرترن (عملگرهای حسابی- حساب حقیقی)

- **حساب حقیقی:** حسابی است که فقط شامل داده های حقیقی است. حساب حقیقی همواره یک نتیجه حقیقی تولید می کند، که اساسا جواب چیزی است که انتظار داریم.

مثال:

$$\frac{6.0}{4.0} = 1.50$$

$$\frac{5.}{4.} = 1.25$$

$$\frac{3.0}{4.} = 0.75$$

عملگرها در فرترن (عملگرهای حسابی- حساب مختلط)

- به طور کلی، اعمال حسابی فقط بین اعداد از نوع مشابه (یا یکسان) با معنی است. به عنوان مثال جمع یک عدد صحیح و یک عدد حقیقی غیر مجاز است. زیرا اعداد صحیح و اعداد حقیقی به شکل های کاملا متفاوتی در رایانه ذخیره می شوند.
- عبارت حاوی هم اعداد حقیقی و هم اعداد صحیح، عبارت مختلط نامیده می شود و حساب دربردارنده اعداد حقیقی و صحیح، حساب مختلط نام دارد.
- در صورت وجود یک عمل بین یک عدد حقیقی و یک عدد صحیح، رایانه عدد صحیح را به عدد حقیقی تبدیل نموده و حساب حقیقی را بر روی اعداد مورد استفاده قرار می دهد.

مثال:

$$1+\frac{1}{4}=1$$

$$1.+\frac{1}{4}=1$$

$$1.+\frac{1.}{4}=1.25$$

عملگرها در فرترن (عملگرهای حسابی- حساب مختلط و به توان رساندن)

- به طور کلی فرترن برای به توان رساندن (اعشاری) اعداد از رابطه زیر استفاده می کند:

$$result = y^x \rightarrow result = e^{xlny}$$

عملگرها در فرترن (عملگرهای ربطی)

- عملگرهای ربطی، عملگرهایی هستند با دو عملوند(operands) عددی یا کاراکتری که یک نتیجه منطقی به دست می دهند. شکل کلی آن به صورت a_1 op a_2 است، که a_1 و a_2 ثابت ها یا رشته های کاراکتری بوده و op یکی از عملگرهای زیر می باشد:

عمل		
مفهوم	شیوه نگارش قدیم	شیوه نگارش جدید
مساوی است با	.EQ.	==
مساوی نیست با	.NE.	/=
بزرگتر است از	.GT.	>
بزرگتر یا مساوی است با	.GE.	>=
کوچکتر است از	.LT.	<
کوچکتر یا مساوی است با	.LE.	<=

عملگرها در فرترن (عملگرهای ربطی)

- اگر رابطه بین a_1 و a_2 که توسط عملگر نشان داده شده است، درست باشد آنگاه نتیجه یک مقدار .TRUE باز می گرداند. در غیر این صورت، نتیجه .FALSE باز می گرداند.

مثال:

عمل	نتیجه
3 < 4	.TRUE.
3 <= 4	.TRUE.
3 == 4	.FALSE.
3 > 4	.FALSE.
4 <= 4	.TRUE.
‘A’ < ‘B’	.TRUE.

- عبارت منطقی آخر .TRUE. است، چون کاراکترها با ترتیب الفبایی برآورد می گردند.

عملگرها در فرترن (عملگرهای منطقی)

- عملگرهای منطقی، عملگرهایی هستند که با یک یا دو عملوند (operands) یک نتیجه منطقی به دست می دهند. شکل کلی عملگر منطقی به صورت زیر است:

(عملگرهای دوتایی)	L_1 .op. L_2
(عملگرهای تکتایی)	.op. L_1

- که L_1 و L_2 عبارت منطقی، متغیر یا ثابت بوده و .op. یکی از عملگرهای منطقی جدول زیر است.

تعریف	عملگر
نتیجه .TRUE. است اگر هم L_1 و هم L_2 صحیح(.TRUE.) باشد	L_1 .AND. L_2
نتیجه .TRUE. است اگر یکی یا هر دو L_1 و L_2 صحیح(.TRUE.) باشد	L_1 .OR. L_2
نتیجه .TRUE. است اگر L_1 و L_2 مشابه باشند(هر دو صحیح(.TRUE.) یا هر دو غلط(FALSE))	L_1 .EQV. L_2
نتیجه .TRUE. است اگر یکی از L_1 و L_2 صحیح(.TRUE.) و دیگری غلط(FALSE) باشد	L_1 .NEQV. L_2
نتیجه .TRUE. است اگر L_1 غلط(FALSE) باشد و غلط(FALSE) است اگر L_1 صحیح(.TRUE.) باشد	.NOT. L_1

عملگرها در فرترن (اولویت عملگرها- جدول کلی)

نوع	عملگر						شرکت پذیری
حسابی	**						راست به چپ
	*			/			چپ به راست
	+			-			چپ به راست
ربطی	<	<=	>	>=	==	/=	ندارند
منطقی	.NOT.						راست به چپ
	.AND.						چپ به راست
	.OR.						چپ به راست
	.EQV.			NEQV.			چپ به راست

عملگرها در فرترن (عملگرهای منطقی)

مثال:

عبارت منطقی	نتیجه
.NOT. (3 < 4)	.FALSE.
(3 < 4) .OR. (6 /= 5)	.TRUE.
(3 < 4) .AND. (6 /= 5)	.TRUE.
(6 /= 5) .NEQV. (4 > 5)	.TRUE.
(3 < 4) .OR. (6 /= 5) .AND. (4 > 5)	.TRUE.
.NOT.((3 < 4) .EQV. (6 /= 5))	.FALSE.

عبارت جایگزینی (Assignment Statement)

- منظور از عبارت جایگزینی محاسبه مقدار یک عبارت عددی و جایگزین کردن آن به یک متغیر می باشد. در حالت کلی داریم:

```
variable_name = expression
```

- عبارت جایگزینی، مقدار عبارت سمت راست تساوی را محاسبه و مقدار آن را به متغیر نامگذاری شده موجود در سمت چپ علامت تساوی اختصاص می دهد.

- دقت کنید که علامت تساوی به معنای برابری، به معنای معمول کلمه نیست.

مثال:

```
x = x + 1
```

عبارت جایگزینی (Assignment Statement)

variable_name = expression

نکات مهم:

- **اگر نوع** variable_name **و** expression **یکسان باشد، نتیجه محاسبه در** variable_name **ذخیره می شود.**
- **اگر نوع** variable_name **و** expression **یکسان نباشد، نتیجه** expression **ابتدا به نوع** variable_name **تبدیل شده سپس ذخیره می گردد.**

مثال:

```
INTEGER :: Total, Amount
INTEGER :: Unit = 5

Amount = 100.99
Total = Unit * Amount
```

```
REAL, PARAMETER :: PI = 3.1415926
REAL :: Area
INTEGER :: Radius

Radius = 5
Area = (Radius ** 2) * PI
```

توابع ذاتی فرترن (Fortran Intrinsic Functions)

- فرترن برای بسیاری از توابع و اعمال ریاضی، توابع متناظری را فراهم کرده است که توابع ذاتی نام دارد.

- در هنگام استفاده از توابع ذاتی اطلاعات زیر نیاز است:
  - نام و معنی هر تابع (به عنوان مثال SQRT( ) جهت محاسبه جذر اعداد به کار می رود)
  - تعداد آرگومان های هر تابع ذاتی (مقادیر ورودی به تابع تحت عنوان آرگومان شناخته شده و آنها بلافاصله پس از نام تابع در پرانتزها ظاهر می گردند.)
  - نوع و حدوده آرگومان (به عنوان مثال آرگومان تابع SQRT( ) می بایست نامنفی باشد)
  - نوع مقداری که تابع بازمی گرداند.

توابع ذاتی فرترن (Fortran Intrinsic Functions)

Function	*Meaning*	*Arg. Type*	*Return Type*
`ABS(x)`	absolute value of `x`	`INTEGER`	`INTEGER`
		`REAL`	`REAL`
`SQRT(x)`	square root of `x`	`REAL`	`REAL`
`SIN(x)`	sine of `x` radian	`REAL`	`REAL`
`COS(x)`	cosine of `x` radian	`REAL`	`REAL`
`TAN(x)`	tangent of `x` radian	`REAL`	`REAL`
`ASIN(x)`	arc sine of `x`	`REAL`	`REAL`
`ACOS(x)`	arc cosine of `x`	`REAL`	`REAL`
`ATAN(x)`	arc tangent of `x`	`REAL`	`REAL`
`EXP(x)`	exponential e^x	`REAL`	`REAL`
`LOG(x)`	natural logarithm of `x`	`REAL`	`REAL`

`LOG10(x)` is the common logarithm of `x`!

توابع ذاتی فرترن (Fortran Intrinsic Functions)

Function	*Meaning*	*Arg. Type*	*Return Type*
`INT(x)`	truncate to integer part `x`	`REAL`	`INTEGER`
`NINT(x)`	round nearest integer to `x`	`REAL`	`INTEGER`
`FLOOR(x)`	greatest integer less than or equal to `x`	`REAL`	`INTEGER`
`FRACTION(x)`	the fractional part of `x`	`REAL`	`REAL`
`REAL(x)`	convert `x` to `REAL`	`INTEGER`	`REAL`

مثال:

```
INT(-3.5) → -3
NINT(3.5) → 4
NINT(-3.4) → -3
FLOOR(3.6) → 3
FLOOR(-3.5) → -4
FRACTION(12.3) → 0.3
REAL(-10) → -10.0
```

توابع ذاتی فرترن (Fortran Intrinsic Functions)

Function	*Meaning*	*Arg. Type*	*Return Type*
`MAX(x1, x2, ..., xn)`	maximum of `x1`, `x2`, ... `xn`	`INTEGER`	`INTEGER`
		`REAL`	`REAL`
`MIN(x1, x2, ..., xn)`	minimum of `x1`, `x2`, ... `xn`	`INTEGER`	`INTEGER`
		`REAL`	`REAL`
`MOD(x,y)`	remainder `x - INT(x/y)*y`	`INTEGER`	`INTEGER`
		`REAL`	`REAL`

قاعده نوع ضمنی(IMPLICIT NONE)

- در نسخه های پیشین فرترن، چیزی به نام قاعده نوع ضمنی وجود داشت. بدین معنی که اگر نام متغیری با حروف I تا N (I ,J, K, L, M, N)آغاز می شد به صورت خودکار، نوع عدد صحیح برای آن در نظر گرفته می شد و اسامی با مابقی حروف به صورت خودکار نوع حقیقی در نظر گرفته می شدند.

- به دلیل بروز احتمالی خطا در تعریف نام متغیر در قاعده نوع ضمنی، در برنامه های فرترن عموماً به صورت پیش فرض این قابلیت غیر فعال می گردد. مکان نوشتن دستور آن بعد از نام برنامه و پیش از معرفی متغیرها است.

- توصیه اکید می شود که همواره این گزینه در برنامه فرترن شما غیر فعال باشد و با توجه به خصوصیات هر متغیر نوع آن آگاهانه تعریف و استفاده گردد.

ورودی (Input)

- فرترن ۹۰ از دستور READ(*,*) به منظور مقدار دهی متغیرها، از طریق صفحه کلید، استفاده می کند.

READ(*,*) :: v_1, v_2, v_3, …, v_n
READ(*,*)

- شکل دوم اشاره شده در بالا مفهوم خاصی دارد که در اسلایدهای بعدی به آن اشاره می شود.

مثال:

```
INTEGER                  :: Age
REAL                     :: Amount, Rate
CHARACTER(LEN=10)        :: Name

READ(*,*)   Name, Age, Rate, Amount
```

ورودی (Input)

- دستور READ(*,*) به طور پیش فرض داده های ورودی را از صفحه کلید دریافت می کند.
- اگر دستور READ(*,*) دارای n متغیر باشد، می بایست n ثابت (داده) ورودی به آن اختصاص یابد. هر داده ورودی می بایست دارای نوع یکسانی با متغیر تعریف شده متناظر باشد.
- داده های ورودی را می توان توسط فاصله (با کلید space) و همچنین ویرگول از هم جدا کرد.

مثال:

```
CHARACTER(LEN=5)  :: Name
REAL              :: height, length
INTEGER           :: count, MaxLength

READ(*,*) Name, height, count, length, MaxLength
```

```
Input: "Smith" 100.0 25 123.579 10000
```

ورودی (Input)

- هرگاه مفسر فرترن با دستور جدید READ(*,*) روبرو شود، آنگاه کلیه مقادیر در خط کنونی در حال خواند را رها نموده و در یک خط جدید به دنبال داده می گردد.
- مراقب دستورات READ(*,*) که در چند خط متوالی نوشته شده است باشید!!

مثال:

```
INTEGER :: I,J,K,L,M,N

READ(*,*) I, J
READ(*,*) K, L, M
READ(*,*) N
```

Input:

```
100 200
300 400 500
600
```

```
INTEGER :: I,J,K,L,M,N

READ(*,*) I, J, K
READ(*,*) L, M, N
```

```
100  200  300
500  600  700
900
```

ignored!

```
400
800
```

READ(*,*) always starts with a new line

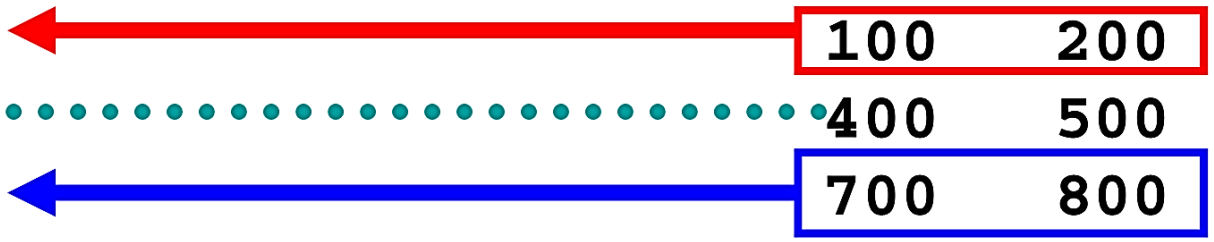
ورودی (Input)

- دستور READ(*,*) همواره در خط جدید مفسر فرترن شروع به اجرا می شود. یک دستور READ(*,*) بدون متغیر بدان معناست که ورودی های آن خط نادیده گرفته می شوند.

مثال:

```
INTEGER :: P, Q, R, S

READ(*,*) P, Q
READ(*,*)
READ(*,*) R, S
```



خروجی (Output)

- فرترن ۹۰ از دستور WRITE(*,*) به منظور نمایش اطلاعات بر روی صفحه نمایش استفاده می کند و به دو صورت زیر نوشته می شود:

```
WRITE(*,*) :: exp1, exp2, exp3, …, expn
WRITE(*,*)
```

- دستور WRITE(*,*) نتایج هر یک از عبارت های $exp_1, exp_2, \ldots, exp_n$ را ارزیابی و در صفحه نمایش چاپ می کند.

- هر دستور WRITE(*,*) نتایج عبارت ها را در یک خط جدید می نویسد.

- دستور WRITE(*,*) که فاقد عبارت های $exp_1, exp_2, \ldots, exp_n$ باشد منجر به رها کردن یک خط در صفحه خروجی می گردد.