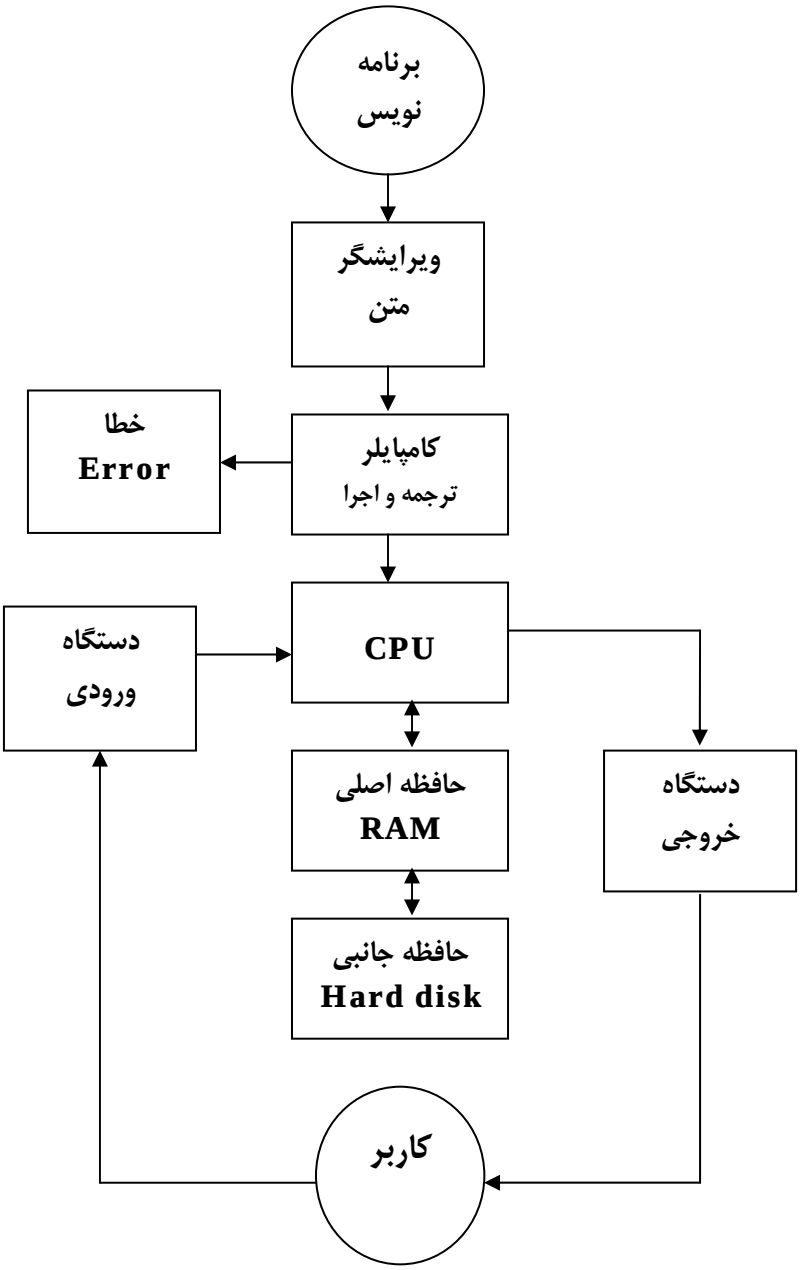
## خلاصه جزوهٔ درس

# برنامه سازی کامپیوتر

# ( پاسکال مقدماتی)

تهیه کننده : گنجی




***نمودار ساده شده‌ای از تعامل برنامه نویس، کامپیوتر و کاربر استفاده کننده از برنامه***

**مقدمه:**

**دستگاه‌های ورودی : صفحه کلید ، اسکنر ، موس، مودم**

**دستگاه‌های خروجی : مانیتور ، پرینتر ، مودم**

**کامپایلر : برنامهٔ سیستمی که دستورات سطح بالا برنامه را به دستورات ساده و قابل اجرا برای ماشین (CPU) تبدیل می‌کند و دستورات و توابع نوشته شده توسط برنامه نویس را به زبان ماشین تبدیل**

**می کند و جهت اجرا آن را تبدیل به دستورات قابل اجرا برای ماشین می کند. کامپایلر جزئی از هر زبان برنامه نویسی است.**

**واحد پردازش مرکزی یا همان ( CPU) به صورت منحصر به فرد از هرنوع و تولید هر شرکت یک سری دستورات مخصوص به خود را دارد و در اصطلاح یک سری کلمات سطح پایین را می تواند شناسایی کند . کامپایلر نیز دستورات و توابع نوشته‌شده توسط ما (که دستورات سطح بالا گفته می‌شوند) را به همان دستورات قابل فهم و اجرای هر نوع واحد پردازش مرکزی تبدیل می کند. اصولا کار کامپایلر تبدیل دستورات سطح بالا به صفر یا یک می باشد و این کار را کامپایل کردن می‌گویند . وظیفه برنامه نویس تجزیه و تحلیل هر کار خواسته شده (که ممکن است نسبتا پیچیده باشد) به دستورات سطح بالا ولی ساده‌ای است که یک کامپایلر می‌فهمد. مجموعه این دستورات برنامه را تشکیل می‌دهند. هر برنامه یک بار نوشته می‌شود، اما ممکن است هزاران یا میلیونها بار توسط کاربرهای بسیار و در مکانها و زمانهای مختلف استفاده و اجرا شود.**

## آشنایی با اصطلاحات برنامه نویسی

سنگ بناهای تشکیل هر برنامه عبارتند از:

ـ **کلمات کلیدی key word :** کلمات اصلی هر زبان برنامه نویسی

ـ **شناسه‌ها identifier :** نام هایی که در برنامه برای نام گذاری استفاده می شود.

ـ **عملها ، عملگر ها ، عملوند ها :** operand / operator / opration محاسباتی و منطقی

با یک مثال تشریح می‌شود:

| عملوند | عملگر | عملوند | | نتیجه عمل |
|---|---|---|---|---|
| 5 | + | 3 | → | 8 |

عمل

ـ **عبارت ها:**

**محاسباتی**
**منطقی**

(عبارت به مجموعه ای از چند عمل موثر بر هم گفته می‌شود که منجر به یک نتیجه نهایی می شود)

ـ **توضیحات ( comment) :** قسمتی در متن برنامه که فقط برای راهنمایی برنامه نویس نوشته می شود و ترجمه یا اجرا نمی شود.

ـ **علامت ها ( symbol) :** مهمترین علامتهای پاسکال عبارتند از ( ) [ ] { } ( * *) ^ ' : ; . ,

ـ **برچسب ها ( label ) :** نام یا عددی است که به محل دلخواهی از برنامه داده می شود .

ـ **ثابت ها ( constant) :** مقادیری که در برنامه قرار نیست تغییر کند یا تغییر برایشان معنی ندارد.

### انواع داده های ساده در پاسکال :

۱) **عدد صحیح Integer**

۲) **عدد حقیقی Real**

۳) **کاراکتر Char :** کاراکتر کوچکترین واحد اطلاعات نوشتاری است. حروف الفبای کوچک و بزرگ، ارقام ۰ تا ۹، علامتهای نوشتاری و تعدادی علامت متفرقه که در مجموع ۲۵۶ نشانه متفاوت را تشکیل میدهند تمام کاراکترهای

استاندارد هستند. هر کدام از این کاراکترها با عددی بین ۰ و ۲۵۵ متناظرند که به عدد استاندارد مختصه به آن کاراکتر کد اسکی آن کاراکتر می‌گویند.

**٤) رشته String:** مجموعه ای از چند کاراکتر در کنار هم

**۵) بولی Boolean :** نوع داده منطقی که برای ارزیابی صحت یا عدم صحت گزاره‌ها بکار می‌رود و فقط یکی از دو حالت زیر را می‌تواند دارا باشد:

**درست True**
**نادرست False**

**نحوهٔ نمایش ثابتها در پاسکال:**

**۱) اعداد صحیح ثابت :** ارقام ۰ تا ۹ و احتمالا علامت ( + ) یا ( − ) در سمت چپ. فاصلهٔ خالی بین اعداد و یا به کاربردن کاراکتر های دیگر مجاز نیست. مثال: 4235 و 23- و 2000+ نمونه‌هایی از ثوابت integer درست هستند.

**۲) اعداد حقیقی ثابت:**

الف) معمولی : ارقام ۰ تا ۹ و علامت ( - یا + ) در سمت چپ و یک نقطه (.) که حتما باید بین دو رقم محصور باشد، مثل:

+0.00452 و -5.0 و 236.3

ب) تواندار: یک عدد توان دار دارای یک پایه در سمت چپ و یک نما در سمت راست است که با یک حرف e یا E از هم جداشده اند. پایه مانند اعداد حقیقی معمولی و نما مانند اعداد صحیح نمایش داده می‌شود. فضای خالی در نمایش اعداد ممنوع است. مثالهایی از چند عدد تواندار درست:

+6.0E8 و 23.502e-13 و -0.24E+23

**۳) کاراکتر های ثابت :** برای نشان دادن یک کاراکتر از cotation در دوطرف آن استفاده می کنیم مثل:

'6' و '*' و '+' و ' M '

اگر از cotation استفاده نشود و مثلاً فقط 6 را بنویسیم مفهوم عدد 6 را دارد و می توان از آن در محاسبات ریاضی استفاده کرد ولی '6' معادل یک کاراکتر و دارای کد اسکی است. حتی فضای خالی هم مانند یک کاراکتر عمل می کند:

' ' ( به فضای خالی بین کوتیشن ها توجه شود )

**۳) رشته‌های ثابت :** مجموعه ای از کاراکتر ها در کنار هم که با cotation محصور شده اند مثل:

' my naim is ali ' و '32' و 'ab'

نکته : اگر در یک رشته بیش از دو cotation استفاده کنیم خطا خواهیم داشت: 'don ' t care'

که در این صورت باید به شکل مقابل نوشته شود : 'don''t care'

**متغیر :**

متغیر نماینده محلی از حافظه است که قابلیت آن را دارد که اطلاعات دلخواه و از هر نوع دلخواه تعریف شده در پاسکال ( مثل عدد صحیح، حقیقی، کاراکتر و ... ) در آن ذخیره شود. در واقع هر متغیر مانند یک ظرف عمل می‌کند که نوع داده خاصی را میتواند در خود نگه دارد. برنامه نویس به دلخواه و بنا بر نیاز خود مختار است متغیرهایی برای برنامه خود در نظر بگیرد. هر متغیر با پارامترهایی شناخته می‌شود که عبارتند از:

- نام متغیر
- نوع دادهای که می‌تواند در خود نگه دارد
- مقدار، یا اطلاعات ذخیره در آن

این پارامترها را برنامه نویس بعنوان خالق متغیر تعیین می‌کند. نامگذاری هر متغیر باید از قوانین نامگذاری که بعداً ذکر خواهدشد پیروی کند.

بطور خلاصه:

- هر متغیر ظرفی برای ذخیره کردن اطلاعات است که نماینده محل نا معلومی از حافظه هم است
- هر متغیر باید دارای نام مختص به خود باشد
- باید نوع داده ای که در آن می خواهیم ذخیره کنیم از قبل معین شده باشد
- مقدار مناسب و همنوع با نوع متغیر در آن قابل ذخیره است .

**معرفی متغیر ها:**

ذکر شد که در ابتدای هر برنامه باید متغیر هایی را که می خواهیم از آن استفاده کنیم معرفی کنیم. معرفی متغیرها مانند ثبت نام از آنها نزد کامپایلر برای شناسایی و رزرو جای آنها در حافظه است.

**روش معرفی یک متغیر :**

**Var (ها)نام متغیر : نوع متغیر**

که Var یک کلمه ی کلیدی پاسکال است . مثال:

```
var   ali :  string ;
var   hamid, final:  real;
var   key :  char;
```

هرگاه دو یا چند متغیر از یک نوع داشته باشیم از یک دستور Var واحد استفاده می‌کنیم:

```
Var    x,y,z : real ;
         a: integer ;
```

معرفی متغیرها قبل از شروع دستورات اصلی برنامه باید توسط برنامه نویس انجام بگیرد. دقت کنید بین هر دو دستور مستقل در پاسکال باید علامت سمیکالن(;) قرار گیرد.

**مقدار دهی به متغیر(انتساب):**

**دستور انتساب:** برای مقداردهی به هر متغیر دلخواه به کار می‌رود و طرز استفاده از آن به این شکل است:

**نام متغیر := عبارت دلخواه**

نوع داده‌ای حاصل عبارت با نوع متغیر باید یکی باشد. در این صورت ابتدا حاصل عبارت مورد نظر توسط کـامپیوتر محاسـبه و سپس نتیجه آن در متغیر (ظرف) نامبرده شده قرار می‌گیرد.

**ساختار کلی یک برنامه پاسکال:**

**Var** ..... معرفی متغیر ها =>
.....
**Begin** ..... بلوک اصلی برنامه
.....
**End.**

**مثال:**

```
Var   x , y : integer ;
      z : real ;
      c: char ;
Begin
      x:= 15 ;
      y:= 5*6 div x-2 +3 ;
      z:= x/y ;
      c: = 'Q'
End.
```

سمت چپ دستور انتساب حتما باید نام ( فقط) یک متغیر باشد و اگر یک عبارت یا یک ثابت باشد خطاسـت بـه عنـوان مثـال کامپایلر در حالت زیر خطا (error) می دهد چون در سمت چپ باید نام یک متغییر که از حروف الفبا است قرار داشته باشد.

```
x + y : =  5  ;
3 * x  : = 10 ;
81 : =  x  ;
```

همچنین اگر متغیری در برنامه نام برده شود بی آنکه در ابتدای برنامه معرفی شده باشد کامپایلر اعلام خطا (Error ) خواهد نمود.

**قواعد نامگذاری :**

هر نام درست و معتبر در پاسکال تشکیل شده است از تعدادی حروف الفبا و ارقام 0 تا 9 که سمت چپ آن یکی از حروف الفبا باشد .

Ali  x2y  p5  taghi  <= مثالهایی از نام های مجاز

x*y  a b  2x  1  <= مثالهایی از نام های غیر مجاز

↑
**فاصله**

**عملگر های پاسکال :**

به ترتیب اولویت محاسبه عبارتند از:

| | |
|---|---|
| ردیف یک (بالاترین اولویت) | **Not** |
| ردیف دو | **And** * / Div Mod |
| ردیف سه | **Or** + - |
| ردیف چهار (پایینترین اولویت) | < <= > = = < > |

**توضیحات لازم:**

ـ عملگرهای + و - همان کار جمع و تفریق ریاضی را انجام می‌دهند.

ـ عملگر * کار عمل ضرب را انجام می‌دهد. سه عملگر ذکرشده در صورت صحیح (integer ) بودن هردو عملوندشان نتیجه صحیح و الا نتیجه حقیقی می‌دهند.

ـ عملگر / تقسیم حقیقی را انجام می‌دهد و نتیجه تقسیم عددی حقیقی خواهد بود.

ـ عملگر Div خارج قسمت تقسیم صحیح و عملگر Mod باقیمانده تقسیم صحیح را محاسبه میکنند و نتیجه صحیح می‌دهند.

ـ عملگرهای Not و And و Or اعمال « **نه**» و « **و** » و « **یا** » ی منطقی را انجام می‌دهند و نتیجه آنها همواره مقداری بولی است (True یا False). جدول زیر عملکرد این عملگرها را نشان می‌دهد.

ـ در هر ردیف اولویت عملگرهای آن ردیف یکسان است.

| P | Q | P And Q | P Or Q | Not P |
|---|---|---|---|---|
| False | False | False | False | True |
| False | True | False | True | True |
| True | False | False | True | False |
| True | True | True | True | False |

ـ عملگرهای ردیف چهار عملگرهای مقایسه‌ای نام دارند و دو عملوندشان را با هم مقایسه کرده و نتیجه بولی میدهند. توجه کنید = معنی تساوی، <> معنی عدم تساوی، =< معنی کوچکتر یا مساوی و => معنی بزرگتر یا مساوی می‌دهد.

**نکته:** تمام عملگرهای ذکرشده روی دو عملوند عمل می‌کنند بجز Not که تک عملوندی است.

**نکته:** پرانتز () اگر چه عملگر نمی باشد، در بالاترین اولویت قرار دارد. در عبارات همیشه داخل پرانتز زودتر محاسبه می‌شود.

**بطور خلاصه:** عملگر های منطقی نتیجه بولی می‌دهند ( false , true ) که عبارتند از: or , and , not و عملگرهای مقایسه‌ای.

عملگر های محاسباتی نتیجه عددی ( صحیح یا حقیقی ) می‌دهند که عبارتند از: * / mod div + -

| عملگر | کاربرد | مثال |
|---|---|---|
| * | ضرب صحیح و حقیقی | 3 * 7= 21 |
| / | تقسیم فقط صحیح و حقیقی | 10/4= 2.5 |
| + | جمع صحیح و حقیقی | 6 + 4 =10 |
| - | تفریق صحیح و حقیقی | 2 – 7 = -5 |
| Div | تقسیم فقط صحیح ( خارج قسمت تقسیم ) | 15 div 7 = 2 |
| Mod | باقی مانده فقط صحیح ( باقی مانده تقسیم ) | 15 mod 7 = 1 |

مثال:

5 div 6 = 0
5 mod 6 = 0
6 <> 4 = true
10 > 8 = true
2 = 3 >>>> false
true or true => true
true or false => true
false or false => false

true and true => true
true and false => false
false and false => false

not true => false
not false => true

نکته: عملگر های مقایسه ای در هنگام ترکیب با Or یا Not یا And باید در پرانتز قرار بگیرند، مثلا عبارت زیر بی معنی تلقی می‌شود و خطاست:

5 > 1 Or 6 > 13

زیرا اولویت Or از مقایسه بیشتر است. درست آن بشکل زیر است:

( 5 >1 ) Or ( 6 > 13 )

**عبارتها :**

بر دو نوعند:

۱) **محاسباتی:** در این عبارتها فقط عملگرهای محاسباتی به کار رفته است و نتیجه آنها یک عدد خواهد بود.

۲) **منطقی:** دست کم یک عملگر منطقی یا مقایسه‌ای در این نوع عبارات دیده میشود. این عبارات همواره نتیجه بولی می‌دهند.

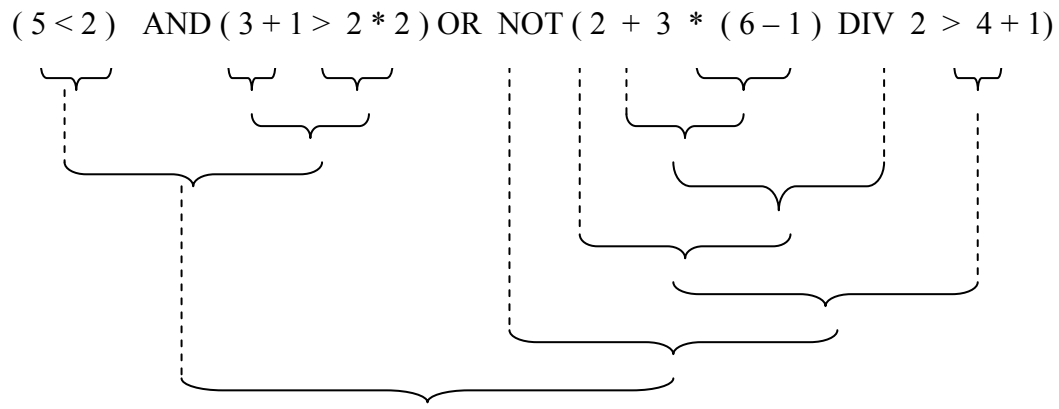
برای محاسبه نتیجه یک عبارت اینکه کدام عمل زودتر انجام شود نتیجه عبارت را تعیین می‌کند. مـثلا در ریاضـی دو عبـارت زیر که در هردو فقط یک عمل جمع و یک ضرب بکاررفته کاملا نتیجه مختلف می‌دهند:

$Y = 5 + 2x \neq 7x$

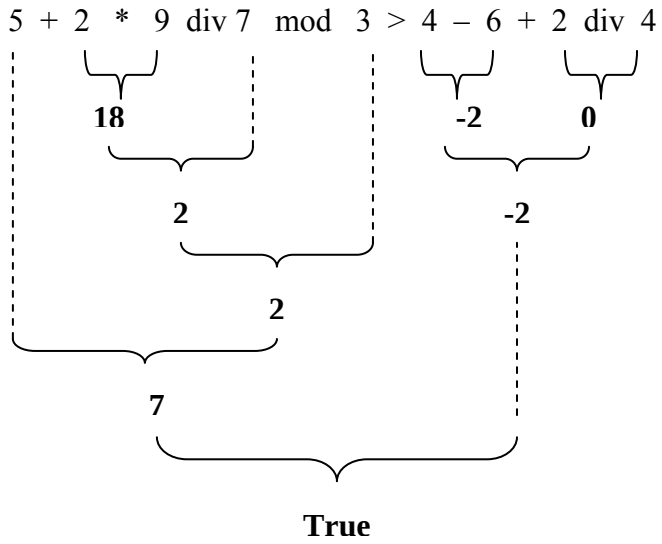
$Y = (5 + 2)\,x = 7x$

برای وقتی که پرانتز اولویت عملگرها را تعیین نکرده باشد باید مطابق جدول عملگرها و اولویتهایشان عملها را یکی یکی در نوبت مقتضی انجام داد. توجه کنید هر بین دو عملی که هم اولویتند آنکه در عبارت زودتر ظاهر شده مقدم است.

**مثال و حل آن: ترتیب اجرای عملگرها را در عبارت زیر مشخص کنید.**




**مثال: حاصل عبارات زیر را بدست آورید : (حل)**




نتیجه کل بولی است ، چون یک عملگر منطقی در این عبارت وجود دارد .

**نکته:**

– در یک عبارت اولویت با داخلی ترین پرانتز است.

– عبارت داخل یک پرانتز مانند یک عبارت مستقل محاسبه و ارزیابی می شود .

**دستورات ورودی / خروجی :**

**دستور ورودی:** دستوری است که از طریق دستگاه ورودی مثل صفحه کلید اطلاعات نوشتاری کاربر را به کامپیوتر منتقل می کند.

**دستور خروجی:** دستوری است که از طریق دستگاه خروجی مثل مونیتور اطلاعات نوشتاری را از کامپیوتر ا به کاربر منتقل می کند.

**دستور خروجی Write :**

write یک دستور خروجی برای چاپ عبارات نوشتاری برنامه نویس بر روی مانیتور است.

**نحوه استفاده از دستور Write :**

**Write** ( لیستی از مقادیر قابل چاپ )

**ـ لیست:** تعدای مولفه که به وسیله ی علامت کاماا از هم جدا شده باشند را لیست می نامیم.

**ـ مقادیر قابل چاپ:** متغیرها، ثابتها و حاصل عبارات مقادیر قابل چاپ محسوب می‌شوند.

مثلاً:

**Write** ( ' hello' , 5 * 2 div 3 , 'a' )

**رشته ثابت** **عبارتی که نتیجه اش ۳ خواهد بود** **کاراکترثابت**

این نتیجه را به ما می دهد:

**hello3a**

در لیست مثال بالا سه مولفه وجود دارد که به ترتیب روی مونیتور چاپ خواهند شدند.

به عنوان مثال دیگر یک دستور خروجی مانند :

```
x: = 5 * 2 ;
Write ( x ,'  ', x – 1 )
```

این نتیجه را به ما می دهد :

**9  10**

فاصله ای که بین 10 و 9 چاپ می شود اثر مؤلفه دوم تابع Write است.

**دستور ورودی Read :**

اطلاعات را از صفحه کلید می گیرد و به متغیر های خواسته شده می دهد.

**Read ( لیستی از نام متغیر ها )**

مثلا برای عبارت **Read ( x , y , z)** می‌توان گفت که:

ـ این دستور به کامپایلر می گوید سه عدد را از صفحه کلید بگیرد و به ترتیب در متغیرهای x , y , z قرار بدهد ( در این صورت تا وقتی که کمتر از سه عدد به صفحه کلید داده شده کامپیوتر اجرای برنامه را ادامه نمی‌دهد ).

بین پرانتزهای دستور Read فقط مجاز هستیم نام متغیر قرار دهیم نه چیز دیگر، بنا بر این تمام دستورات زیر خطا هستند :

Read ( 2 ) چون 2 یک ثابت صحیح است و نام یک متغیر نیست

Read ( x+y )
Read ( x*3) } چون read فقط نام متغیر را می پذیرد و نه یک عبارت را

Read('x') چون 'x' یک ثابت کاراکتری است و نام یک متغیر نیست

**مثال) برنامه ای بنویسید که عددی حقیقی را از کاربر بگیرد و مجذور آن را چاپ کند :؟**

```
 Var  x,y : Real ;
Begin
  Read ( x ) ;
  Y : = x*x ;
  Write ( y )
End.
```

در این جا کامپایلر منتظر می ماند تا مقدار x را وارد کنیم و enter را بزنیم مثلاً

چون در دستور قید نشده که جواب به چه صورت اجرا شود خود دستگاه به صورت علمی و استاندارد جواب را به ما می دهد یعنی چاپ می‌کند:

8.10000E+1

**مثال) برنامه ای بنویسید که دو عدد صحیح از کاربر بگیرد و میانگین آنها را به شکل یک عدد حقیقی نمایش دهد :؟**

**Real** **integer**

**حل:**

```
Var  x , y : integer ;
     Z :  real ;
Begin
  Read ( x , y )  ;
  Z : = ( x+y) / 2   ;
  Write ( Z )
End .
```

```
Var  x , y integer ;
     Z : Real   ;  { منظور میانگین است  Z  }
Begin
  Write ( 'Enter two numbers' ) ;
  Read ( x , y )   ;
  Z : = ( x+y)  / 2  ;
  Write ( ' The average is :' ,  Z  )
End .
```

در سمت چپ حل مثال و در سمت راست حل کاملتر آن مشاهده می‌شود که قسمتهای سیاه خطوطی اختیاری هستند که می‌توان برای راحتی بیشتر کاربر و فقط به منظور راهنمایی او به برنامه اضافه کرد. به رشته‌های اضافی که با این دستورات خروجی اختیاری می‌خواهیم چاپ شوند ***پیام*** میگوییم.

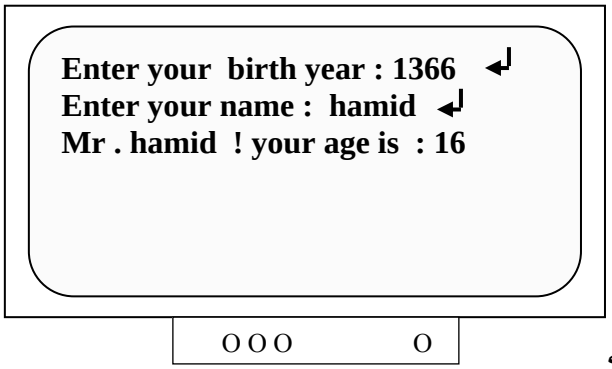
**نتیجه روی مانیتور :**

```
Enter two numbers :    5    6 ↵
The average is :       5.5 E+0000
```

**توضیحات(کامنتها) :** عباراتی هستند که بین (* *) یا {} قرار می گیرند و کامپایلر کاری به آنها ندارد و فقط برای خوانایی برنامه برای برنامه نویس استفاده دارد.

**مثال : برنامه ای بنویسید که سال تولد و نام کاربر را بگیرد و با پیام مناسبی سن او را چاپ کند :؟**

```
Var   brith , age : integer ;
      Name : string  ;
Begin
      Write ( ' enter your brith year: ' ) ;
      Read( birth ) ;
      age := 1383 – birth ;
      write ( ' enter your name ' ) ;
      Read ( name ) ;
      Write ( 'Mr.' ,  name , ' ! your age is :  ' , age)
End.
```




**خروجی اجرای برنامه**

همانگونه که در مثالهای قبل هم دیده شد در هنگام چاپ اعداد حقیقی، کامپیوتر روش نمایش علمی(Scientific) را به کار می‌برد که برای ما خیلی جالب نیست. اگر بخواهیم مقادیر حقیقی به شکل دلخواهمان روی مونیتور چاپ شود باید دستور خروجی Write را قالب بندی کنیم.

**قالب بندی(فرمت) خروجی در دستور Write :**

**Write ( عبارت دلخواهی که قرار است چاپ شود : طول فیلد : دقت )**

**ـ پارامتر های قابل تنظیم برای چاپ یک عبارت :**

۱) **طول فیلد :** ( تعداد کاراکتر های اختصاص داده شده به عبارت روی مونیتور )

۲) **دقت :** ( تعدا د ارقام پشت ممیز )

به عنوان مثال در زیر دو نوع استفاده از دستور Write را برای چاپ عدد پی مشاهده می کنید:

```
P: = 3.1415926535;
Write (P);
```

3.1415926535E+000

اگر بخواهیم این عدد را محدود تر کنیم:

**Write** ( P : 7 : 2 )

3.14

طول فیلد

دقت ( یعنی در این جا تا دو رقم اعشار چاپ می شود )

البته نحوهْ چاپ عدد موردنظر را با انعطاف بیشتری هم می توانیم تعیین کنیم. مثلاً

| مثال | نتیجه ظاهرشده روی مونیتور |
|---|---|
| **Write ( P : 4 : 2 );** | **3.14** |
| **Write ( P : 6 : 3 );** | **3.141** |
| **Write ( P : 1 : 0 );** | **3** |
| **Write ( P : 2 : 2 );** | **3.14** |

در حالت اخیر که تا دورقم اعشار خواسته شده است حداقل فیلد لازم برای نمایش عدد ۴ تـا اسـت ولـی در ایـن جـا برنامـه نویس فیلد را ۲ قید کرده است. در مواردی که طول فیلد و دقت با هم سازگار نباشند کامپیوتر ارجحیت را به دقت می‌دهـد و سعی می‌کند دقت مورد نظر را حتی به بهای نقض فیلد رعایت کند. یعنی کامپیوتر ملزم به رعایت دقت خواسته شـده اسـت و حتی الامکان عدد را در فیلد خواسته شده ( از راست) می نویسد اما اگر عدد با دقتی که خواسته شده در فیلـد خواسـته شـده نگنجد دیگر طول فیلد را رعایت نخواهد کرد .

نکته: اگر نخواهیم طول فیلد خاصی را برای نمایش عدد تعیین کنیم می توانیم طول فیلد را صفر در نظر بگیریم تـا کـامپیوتر طبق روش خود عمل کند . مشخص کردن دقت فقط مخصوص داده‌های نوع Real مـی باشـد و سـایر انـواع داده را فقـط فیلدشان را می توان تنظیم کرد.

**دستور خروجی WriteLn :**

مانند همان دستور Write است با این تفاوت که بعد از چاپ عبارت خواسته شده کرسر خروجی را به سطر جدید منتقل می کند تا احیانا دستور Write بعدی برنامه اطلاعات خودش را در سطر بعدی چاپ کند. از این بـه بعـد مـا بـه جـای دسـتور Write از دستور Writeln استفاده می کنیم مگر آنکه بخواهیم عمداً داده‌های هر دو دستور را چسبیده بـه هـم چـاپ کنیم. مثال:

```
Writeln('Hello');
Writeln('Friend');
```

```
Hello
Friend
_
```

**بلاک ( Block ) :**

به مجموعه ای از دستورات دلخواه و مرتبط به هم که به صورت متوالی در برنامه ذکر می شوند و هدف خاصی را دنبال می کنند و بین Begin و End محصور شده اند گفته می‌شوند.

```
Begin
.
.        } دستورات دلخواه
.
End
```

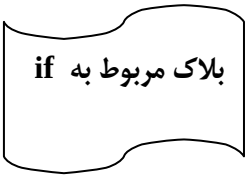
در هر برنامه می توان بنا بر نیاز به صورت نا محدودی بلاک در نظر گرفت. بلاک ها می توانند تو در تو یا ازهم جـدا باشـند. پاسکال با یک بلاک همانند یک دستور مرکب عمل می کند. باید همیشه (**فقط**) در پایان بلاک اصلی برنامه نقطه بگـذاریم. *بقیه بلاکها با نقطه پایان نمی‌یابند*

**ساختارهای تصمیم گیری**

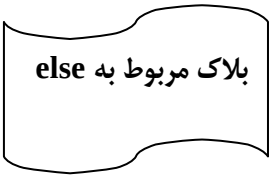
**ساختار های شرطی :**

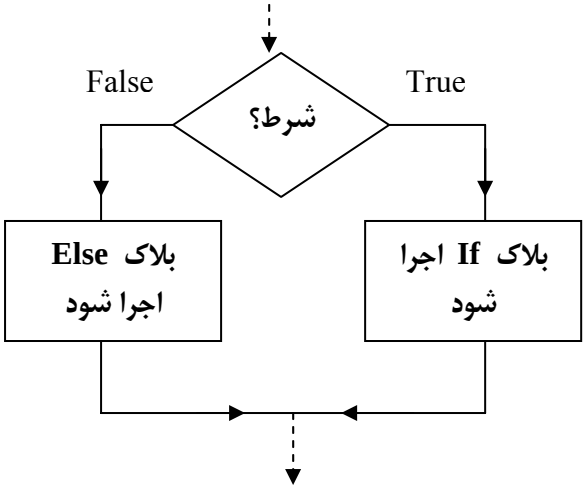
**ـ دستور شرطی If - Then - Else ( اگر ـ آنگاه ـ والا ):**

**If شرط Then**




**Else**




شرط می‌تواند هر عبارت شرطی دلخواه باشد که نتیجه TRUE یا FALSE داشته باشد.

**مثال: برنامه ای بنویسید که عددی حقیقی را از کاربر بگیرد و قدر مطلق آن را بنویسد :؟**

حل: عدد مورد نظر را در ظرف x و قدر مطلق آن را بعد از محاسبه در ظرف a می‌گذاریم.

```
Var  a, x: Real;
Begin
Writeln ( 'enter a number') ;
Read ( x ) ;
If  x > = 0  Then        ──►  شرط خواسته شده
Begin
   a : = x        ──►  ( if   block )
End
Else
Begin
    a : = -x      ──►  ( else  block )
End;
Writeln( ' absolute of x  is' ,  a :10  : 3 )
End .
```

نکته : همیشه و همیشه فقط یکی از دو بلاک if و یا else اجرا می شود.

نکته : if را بدون else می توان استفاده کرد ولی else را بدون if و بلاک آن نمی توان به کاربرد .

نکته: اگر یک بلاک تنها شامل یک دستور باشد می‌توان Begin و End برای آن استفاده نکرد ( اختیاری است ).

**مثال : برنامه ای بنویسید که درایه‌های یک ماتریس ۲ در۲ را به عنوان ورودی بگیرد ، در صورتی که معکوس پذیر باشد ماتریس معکوس را محاسبه کند و نمایش دهد و اگر معکوس پذیر نباشد این امر را گوشزد کند .**

نکته : شرط معکوس پذیر بودن ماتریس آن است که دترمینان آن صفر نشود.

برا ی یک ماتریس ۲x۲ داریم :

$$X = \begin{pmatrix} a & b \\ c & d \end{pmatrix}, \ det = |X| = ad-bc \ \Rightarrow \ X^{-1} = \begin{pmatrix} \mathbf{d/det} & \mathbf{-b/det} \\ \mathbf{-c/det} & \mathbf{a/det} \end{pmatrix}$$

**حل:** متغیرهای مورد نیاز:

a , b , c ,d درایه‌های ماتریس اصلی را نشان می‌دهند

ai , bi , ci, di درایه‌های ماتریس معکوس را نشان می‌دهند

det دترمینان ماتریس را نشان می‌دهد

```
Var   a , b , c , d,  det , ai , bi , ci , di: real
Begin
Writeln ( ' enter elements of the matrix :' ) ;
Read ( a , b , c , d ) ;
Det  :  = a * d – b * c ;
  If det <> 0.0 then
   Begin
     ai  : =   d / det  ;
     bi  : =  -b / det  ;
     ci  : =  -c / det  ;
     di  : =   a / det  ;
     writeln ( ai : 8 : 3: , bi: 8: 3 );
     writeln ( ci : 8 : 3  , di: 8 : 3)
   end
else
  Begin
    writeln (  ' inverse of matrix not exists ' )
  End
End .
```

**نکته:**

- در هربرنامه دستورات هم طراز مانند if و else باید زیر هم و همچنین Begin و end مربوط به یک بلاک بهتر است زیر هم نوشته شود .
- قبل از دستور else نباید (;) قرار بگیرد.
- بلاک else در برنامه فوق تک دستوری است و می‌توان Begin و End آن را حذف کرد.

**مثال : برنامه ای بنویسید که با توجه به نمودار زیر از روی سنی که کاربر وارد میکند مشخص کند در چه مرحله ای از زندگی قرار دارد .**

**کودک < ۱۲**
**نوجوان ۱۸ ~ ۱۲**
**جوان ۳۸ ~ ۱۸**
**میانسال ۵۵ ~ ۳۸**
**پیر > ۵۵**

**حل:** (محدوده هر بلاک مشخص شده است)

```
Var   age  :  integer ;

Begin
        Read ( age);
        If  age < 12  then
                Writeln( 'koodak')
        Else
                If  age < 18 then
                        Writeln ( ' nojavan')
                Else
                        If   age < 38 then
                                Writeln ( ' javan' )
                        Else
                                If  age < 55  then
                                        Writeln (' miansal')
                                else
                                        Writeln ( 'pir')
        End.
```

در برنامه فوق If های دوم و سوم و چهارم خود دستوراتی از بلاک Else مربوط به دستور های شرطی قبل بودند. این امر هیچ مشکلی ایجاد نمی‌کند زیرا در هر بلاک هر دستور معتبر دلخواه می‌توان به کار برد، ازجمله دستور مرکب دیگری که خود هم ممکن است دارای شاخ و برگهای متعدد باشد. در مثال ذکر شده ساختارهای شرطی به شکل تو در تو و زنجیرهای به کار رفته اند تا حالتهای متعدد یک متغیر را بررسی و در مورد آن تصمیم مناسب را بگیرند.

**نکته: هر دستور شرطی If-Then-Else تا پایان بلاک Else خود فقط یک دستور مستقل محسوب می‌شود.** با این اوصاف بلاک تمام If و Else های برنامه فوق تک دستوری هستند.

**نکته: کلمات Begin و End دستور مستقل محسوب نمی‌شوند.**

**نکته: هر دو دستور مستقل متوالی در پاسکال باید با علامت سمی‌کالن(;) از هم جدا شوند. (فاصله عمودی ملاک جداسازی دو دستور نیست و اصولا تفاوتی با فاصله افقی ندارد)**

**نکته: آخرین دستور هر بلاک نیازی به سمی‌کالن ندارد زیرا دستوری بعد از آن نیست که لازم به جداسازی آن ده باشد.**

**نکته: If و Else آن دو دستور مستقل محسوب نمی‌شوند و جداسازی آنها با سمی‌کالن خطا می‌باشد، یعنی سمی‌کالن قبل از Else خطاست.**

**نکته : در پاسکال جذر $x$ با $Sqrt(x)$ بیان می‌شود:**

$$\sqrt{x} \equiv Sqrt(x)$$

**مثال : برنامه ای بنویسید که با گرفتن ضرایب $a$ , $b$, $c$ از معادلهٔ درجه دوم در مورد تعداد و مقدار ریشه‌های آن اطلاعات جامعی چاپ کند.**

**حل:**

```
Var   a , b , c , delta , x1 , x2 : real ;
   Begin
        Read ( a , b , c )
        Delta : = b*b – 4*a*c ;
        If delta >0 then
        Begin
            x1 : = ( -b + sqrt (delta)) / ( 2*a);
            x2 : = ( -b – sqrt (delta)) / ( 2*a);
           writeln ('do rishe darim ' , x1 , x2 : 6 : 2 )
        end
        else
            if delta = 0 then
            Begin
                x1 : = -b / ( 2*a);
                writeln('yek rishe darim ' , x1 : 6 : 2)
                end
             Else
                 Writeln ( ' rishe nadarim')
end .
```

**ساختار شرطی Case – of –Else :**

مثالی را به یاد بیاورید که در آن خواسته شده بود با گرفتن سن یک شخص مشخص شود در چه مرحله‌ای از زندگی قرار دارد. برای برآوردن هدف فوق مجبور بودیم از ساختارهای If-Else تو در تو استفاده کنیم که شرط هر کدام وظیفه بررسی یک حالت خاص از بین تعداد محدودی حالت ممکن را برعهده می‌گرفت و هر ساختار در بلاک Else ساختار قبلی باید نوشته میشد. این زنجیره پیچیدگی مساله را زیاد می‌کرد و درک آن را مشکل می‌نمود. از سوی دیگر مسائل زیادی نظیر مساله مذکور در عمل وجود دارد که به طریقی مشابه می‌بایست حل شوند. برای حل مسائلی اینچنین ساختار شرطی ویژه‌ای تدارک دیده شده است که کار را بسیار آسان و قابل فهم می‌کند. این ساختار مقداری را برای مطابقت با چند حالت محدود بررسی می‌کند و با در هر حالت که تطبیق مورد نظر وجود داشته باشد دستورات مقتضی برای آن حالت خاص را اجرا کرده و پایان می‌یابد.

**روش استفاده:**

**Case مقدار مورد نظر Of**

**<u>لیست مقادیر حالت اول</u>:**

**بلاک دستورات مقتضی برای حالت اول ;**

**<u>لیست مقادیر حالت دوم</u>:**

**بلاک دستورات مقتضی برای حالت دوم ;**

**<u>لیست مقادیر حالت سوم</u>:**

**بلاک دستورات مقتضی برای حالت سوم ;**

**...**
**...**
**...**
**...**

**Else**

**بلاک دستورات مربوط به عدم تطبیق تمام حالات**

**End**

که Case و Of کلمات کلیدی هستند. وجود Else و بلاک آن اختیاری است.
لیست مقادیر هر حالت متشکل از یک یا چند مقدار غیر حقیقی (جدا شده با علامت کاما از هم) است یا یک زیر بازه از یک نوع ترتیبی است. انواع ترتیبی ساده عبارتند از: integer, char, Boolean .

**زیربازه SubRange: به یک زیرمجموعه محدود از مجموعه‌ای کاملتر از مقادیر ترتیبی متوالی می‌گویند که عضو ابتدا و انتهایی آن مشخص و با دو نقطه افقی از هم جدا شده باشند.** مثل **10 .. 99** که زیربازه‌ای ( از اعداد صحیح ) متشکل از اعداد دورقمی‌صحیح است یا **'A' .. 'Z'** که زیربازه حاوی حروف الفبای بزرگ انگلیسی است.

**مثال: برنامه‌ای بنویسید که کاراکتری را از صفحه کلید دریافت کند و در صورتی که آن کاراکتر یکی از حروف صدادار بزرگ انگلیسی باشد این امر را گوشزد کند.**

**حل:**

```
Var    c: Char;
Begin
        Read(c);
        Case c Of
                'A', 'E', 'I', 'O', 'U', 'Y':
                        Writeln(' It is a Voiced letter ');
                Else
                        Writeln(' It is not a voiced letter' )
        End
End.
```

حالت مورد نظر با یک لیست مشخص شده

**مثال: برنامه‌ای بنویسید که شماره یکی از ماههای شمسی را بگیرد و تعداد روزهای آن را چاپ کند و در صورت نادرست بودن آن شماره ، پیام مناسبی صادر نماید.**

**حل:**

```
Var    month : Integer;
Begin
        Writeln(' Enter month number ');
        Read(month);
        Case month Of
                1 .. 6 :
                        Writeln(' Has 31 Days ');
                7 .. 11:
                        Writeln(' Has 30 Days ');
                12:
                        Writeln(' Has 29 Days ');
                Else
                        Writeln(' Invalid month number ')
        End
End.
```

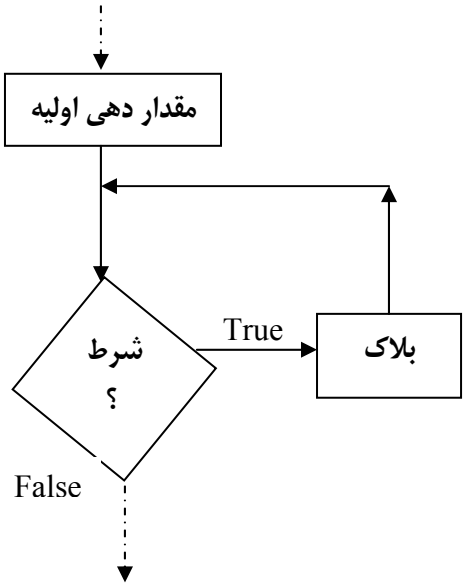
شش ماه اول سال با یک زیربازه مشخص شده

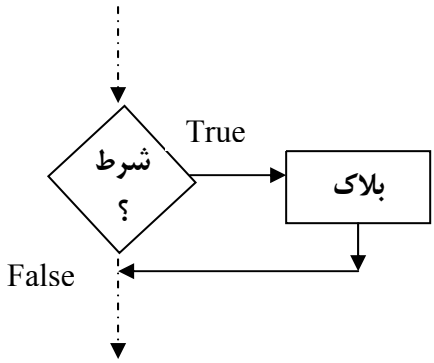
پنج ماه بعدی سال با یک زیربازه مشخص شده

ماه اسفند با یک عدد مشخص شده

ساختار هاي تكرار (حلقه‌ها )

ساختار **While** : **(تا زمانی که شرط درست است دستورات بلاک را تکرار کن )**




**فلوچارت یک ساختار شرطی ( if بدون else) برای مقایسه با ..... ساختار تکرار دستور ( while )**

**طرز استفاده :**

**While شرط Do**

**; بلاک**

**اجزا ی هر حلقه درست:**

۱) متغیر حلقه ۲) مقدار دهی اولیه به متغیر حلقه ( قبل از شروع حلقه ) ۳) شرط حلقه

۴) بلاک حلقه ۵) بهنگام سازی ( تغییر متغییر حلقه ) در بلاک

برای تغییر یک متغیر ( از جمله در بهنگام سازی حلقه) فقط از دو دستور ( انتساب ) و ( ) read میتوان استفاده کرد. دستور انتساب مقدار دهی به متغیر توسط برنامه نویس را ممکن می سازد و دستور read مقدار دهی توسط کاربر را فراهم می سازد.

**مثال: برنامه ای بنویسید که اعداد صحیح از ۱ تا ۱۰۰۰ را چاپ کند .**

حل:

متغیر حلقه

```
Var   I : integer ;
 Begin
I : = 1 ;
 While   I <=1000   do
 Begin
      Writeln ( I );
      I : = I + 1
 End
End .
```

مقداردهی اولیه به متغیر حلقه

شرط حلقه

بلاک حلقه

**بلاک حلقه**

بهنگام سازی حلقه

**مثال : برنامه ای بنویسید که عددی را از کاربر بگیرد و تمام اعداد طبیعی کوچکتر از آن را به شکل نزولی ( شمارش معکوس ) چاپ کند .**

```
Var   I , x : integer ;
Begin
      Writeln ( ' enter a number ' );
      Read ( x);
      I : = x – 1;
While    I > 0   do
Begin
      Writeln ( I );
      I : = I – 1
End

End .
```

true

false

**کامپایلر**

| | |
|---|---|
| 0 | 0 |
| 4 | 5 |
| 3 | |
| 2 | |
| 1 | |
| 0 | |

```
Enter a number :
4
3
2
1
```

(فلشها مسیر اجرای خطوط برنامه را مشخص می‌کند)

**Trace کردن : چک کردن و اجرای دستور به دستور برنامه توسط برنامه نویس برای یافتن اشکال یـا اطمینان از صحت برنامه**

***مثال : برنامه ای بنویسید که تمام اعداد دورقمی فرد را چاپ کند .***
***مثال : برنامه ای بنویسید که عدد صحیحی را بگیرد و تمام مقسوم علیه‌های آن را چاپ کند .***
***مثال : برنامه ای بنویسید که ۵۰ عدد را از کاربر بگیرد و میانگینشان را چاپ کند .***

**مثال: برنامه ای بنویسید که یک عدد طبیعی را از کاربر بگیرد و مشخص کند اول است یا نه ؟**

```
Var  x , I , flag : integer ;
Begin
        Writeln ( ' enter a number ') ;
        Read (x) ;
        Flag : = 0 ;
        I : = 2 ;
        While I < x do
        Begin
                If ( x mod i) = 0 then
                        Flag : = 1 ;
                I : = I + 1
        End ;
        If  flag = 1 then
                Writeln ( ' aval nist');
     Else
                Writeln ( ' aval ast ')
End .
```

**توجه: هر گاه بلاکی Begin و end نداشته باشد یک دستور بیشتر ندارد.**

**حل مثال قبل به روشی دیگر :**

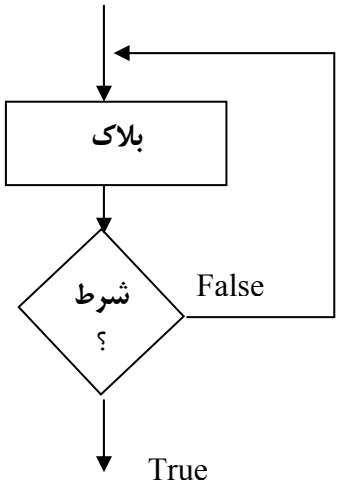
(در این روش از یکی از تعاریف اعداد اول استفاده شده که بیان میکند:
عددی اول است که تعداد مقسوم علیه‌هایش ۲ باشد )

**عدد مورد نظر** **متغییر حلقه** **تعداد مقسوم علیهx**

```
Var    x    ,    I    ,    n    : integer ;
```

```
Begin
        Read(x);
        I : = 1 ;
        n : = 0 ;
        While  I < = x  do
        Begin
                If ( x mod I ) = 0  then
                        n : = n + 1;
                 I : = I +1
        End;
        If  n <> 2 then
                Writeln ( ' aval nist' )
        Else
                Writeln ( 'aval ast')
End .
```

**ساختار تکرار Repeat – Until :**




**طرز استفاده :**

**Repeat**
**بلاک**
**Until شرط ;**

**(آنقدر تکرارکن بلاک را ... تا اینکه شرط True شود )**

این ساختار همانند while یک ساختار تکرار همه منظور است اما دو تفاوت اساسی با while دارد : ۱) این حلقه تا زمان false بودن شرط تکرار می شود و به محض true شدن شرط پایان می یابد ( بر خلاف while)

۲) بلاک دستورات این حلقه قبل از شرط واقع شده است یعنی ابتدا احتماً یک بار بلاک اجرا می شود و بدون آنکه شرط بررسی شده باشد یعنی تعداد دفعات تکرار بلاک حداقل یکبار است.

**نکته: ساختار repeat – until نیاز به Begin و End ندارد.**

**مثال : برنامه ای بنویسید که معدل ۱۰۰ دانشجو را بگیرد و تعداد دانشجویان مشروطی را چاپ کند.**

**حل:** avg معدل هر دانشجو و m تعداد دانشجویان مشروطی را ذخیره خواهد کرد.

```
Var   I , m : integer ;
      avg  : real  ;
 Begin
      m : =  0 ;
      I :  = 0 ;
      Repeat
            Read ( avg );
            If  avg <12  then
                  M : = M + 1;
            I : = I +1
      Until I =100 ;
      Writeln ( ' mashrooti ha  =' ,  m )
End.
```

**مثال : فرض کنید می خواهید برای یک فروشنده برنامه ای بنویسید که با آن بتواند در پایان وقت کاری در آمد روزانه اش را محاسبه کند و نیز میانگین قیمت کالا ها ی فروخته شده اش را بدست آورد . نکته اینجاست که او تا پایان تعطیل شدن کارش نمی داند چند مشتری خواهد داشت . به عنوان قرار داد فرض کنید وارد کردن قیمت صفر برای یک کالا به معنای در خواست اتمام برنامه و مشاهده خروجی کاربر باشد.**

**حل:** avg میانگین قیمت، price قیمت کالای فعلی، sum مجموع قیمتهای کالاهای فروخته شده و n تعداد کالاهای فروخته شده را نشان می‌دهد.

```
Var    avg, price, sum: Real;
       n: Real;
Begin
       Repeat
              Read(price);
              sum:=sum+price;
              n:=n+1
       Until  price=0;
       avg:=sum/(n-1);
       Writeln('Daramad = ' , sum);
       Writeln('Miangin = ' , avg);
End.
```

**مثال : برنامه ای بنویسید که ابتدا تعداد دروس یک دانش آموز را بپرسد و با گرفتن تعداد دروس , نمره درس های مختلفش را هم بگیرد و در نهایت معدلش را چاپ کند .**

**مثال : برنامه ای بنویسید که عدد صحیحی را بگیرد و معادل مبنای دو آن را چاپ کند .**

**مثال : برنامه ای بنویسید که با استفاده از یک حلقه مناسب حاصل سری:** $S = 1 - \frac{1}{2} + \frac{1}{3} - \frac{1}{4} + \frac{1}{5} - \cdots$ **را تا ٤ رقم اعشار بدست آورد .**

**ساختار تکرار For :**

یکی دیگر از ساختار های پاسکال ساختار for است . در این ساختار یک متغییر صحیح به ازای مقادیری از یک مقدار اولیه تا یک مقدار نهایی یک واحد، یک واحد اضافه یا کم می شود و به ازای هر مقدار یک بار بلاک اجرا میشود . ممکن است مقدار نهایی بیشتر از مقدار اولیه باشد در این صورت درهنگام نوشتن کلمه کلیدی to استفاده میشود اما در حالت عکس کلمه کلیدی downto استفاده می شود . در حقیقت برنامه نویس با انتخاب downto یا to افزایشی یا کاهشی بودن بهنگام سازی را تعیین می کند . همان گونه که مشاهده می شود مقدار دهی اولیه بلافاصله بعد ار کلمه for انجام می شود و شرط اولیه for دیگر به شکل یک عبارت منطقی دلخواه نیست بلکه شرط اتمام حلقه تجاوز متغییر حلقه از مقدار نهایی است که در خود ساختار مستتر است یعنی ساختار اولیه for یک ساختار فشرده منظم است اما دیگر مانند ساختار های قبل همه منظوره نیست.

به دلیل آنکه به هنگام سازی تکلیفش کاملا مشخص است برنامه نویس هیچ دستور اضافی برای به هنگام سازی حلقه در بلاک نمی نویسد و به هنگام سازی اتوماتیک توسط کامپایلر انجام میشود .

**طرز استفاده:**

**For مقدار دهی اولیه To/Downto مقدار نهایی Do**
**; بلاک**

**(به ازای ..... از ..... تا ..... هر با انجام بده ..... دستورات بلاک را )**

**مثال : برنامه ای بنویسید که تمام اعداد دورقمی را ازکوچک به بزرگ چاپ کند.**

```
Var  I : integer ;
Begin
For I : = 10  to 99  do
    Writeln ( I )
End .
```

چون بلاک for اینجا تک دستوری است در آن از Begin و end استفاده نمی شود

نکته : for یک ساختار همه منظوره نیست اما هر جا که قابل استفاده باشد کار برنامه نویس را ساده می کند.

**مثال : برنامه ای بنویسید که عدد صحیحی را از کاربر بگیرد و تمام اعداد طبیعی کوچکتر از آن را از بزرگ به کوچک چاپ کند :**

```
Var   I , x : integer ;
Begin
    Writeln ( ' enter x ' ) ;
```

```
    Read ( x ) ;
    For I : = x – 1  down to  1 do
        Writeln ( I )
End .
```

**مثال : برنامه ای بنویسید که عدد صحیحی را بگیرد و با شمردن تعدادمقسوم علیه‌هایش اول بودن یا نبودن آن را تعیین کند :**

**حل:**

```
Var  x , I , n : integer
Begin
    Writeln ( ' enter  x ' );
    Read ( x )
    For I:=1  to  x  Do
        If  x  Mod  I = 0 then
            n:=n+1;

    if  n = 2 then
        Writeln ( ' aval ast ' )
    Else
        Writeln ( 'aval nist ' )
End .
```

*x عدد مورد نظر است، n تعداد مقسوم علیه‌های x و I شمارنده حلقه*

*به ازای تمام اعداد از یک تا x هربار مااحل زیر را انجام بده:*
*اگر x بر بخشپذیر است یعنی باقیمانده x بر I صفر است آنگاه*
*یک واحد به شمارنده مقسوم علیه‌های x اضافه کن (یعنی بشمار)*

*اگر تعداد مقسوم علیه‌های x که در n ذخیره شده ۲ تاست*
*چاپ کن x اول است*
*والا چاپ کن x اول نیست*

**مثال : برنامه ای بنویسید که عدد صحیحی را بگیرد و فاکتوریل آن را چاپ کند .**

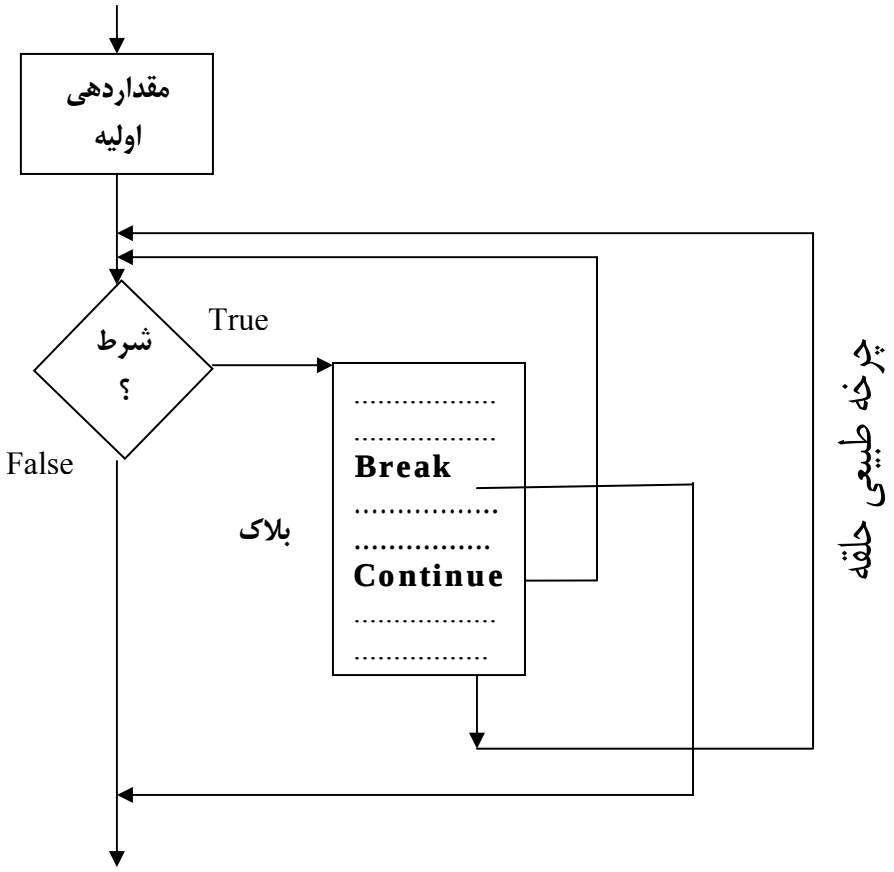
```
Var  I , x , fact : integer ;
Begin
    Writeln ( ' enter  x ' ) ;
    Read ( x );
    Fact:= 1 ;
    For  I : =1 to  x  do
        Fact : = fact * I ;
    Writeln ( ' factorial of  ' , x , '  is  ' , fact )
End .
```

**کلمات کلیدی continue و break :**

**Break : پایان زود رس حلقه => پرش به انتهای حلقه**

**Continue : بررسی زود هنگام شرط و شروع زود رس چرخه بعدی حلقه ( پرش به شرط حلقه)**

این دو کلمه کلیدی روند طبیعی تکرار حلقه را دستکاری می‌کنند و بعنوان ابزارهای مفیدی هستند که برنامه نویس را در جهت هر چه بیشتر و راحتتر و با انعطاف بیشتر در اختیار گرفتن کنترل ساختارهای تصمیم گیری و تکرار یاری می‌کنند.
دستور continue به کامپایلر دستور می دهد تا چرخه بعدی حلقه را از سر گیرد یعنی سراغ شرط رود و در نتیجه از دستورات بعد از continue در آن چرخه صرفنظر کند. دستور break هر زمان اجرا شود در واقع به کامپایلر دستور می دهد تا حلقه ای را که break در آن است را خاتمه دهد حتی اگر شرط در آن تکرار حلقه را تایید کرده باشد .




به دلیل آنکه break و continue نباید در تمام چرخه‌ها فعال باشد لازم است که در این دستورات به شکل مشروط استفاده شوند یعنی در بلاک یک if یا else واقع شده باشند و همچنین به دلیل کاربردشان که فقط برای حلقه‌ها است خارج از بلاک حلقه نمی توان از آنها استفاده کرد. رعایت کردن این دو نکته برای استفاده از این دستورات ضروری است.

**مثال : فرض کنید دانشگاه بخواهد به تمام دانشجویان مرد که متاهل باشند و بیش از دو فرزند داشته باشند و معدلشان بالای ۱۶ است تخفیف بدهد ( تعداددانشجویان مشخص نیست ). برنامه ای بنویسید که مشخصات دانشجویان یعنی نام و معدل و جنسیت و وضعیت تاهل و و تعداد فرزندان را دریافت نموده و اگر دانشجو واجد شرایط تخفیف است این امر را با پیغام مناسبی گوشزد کند و تا زمانی که کاربر کلمه finish را به جای نام دانشجو وارد نکند این کار را ادامه بدهد و در نهایت تعداد دانشجویان واجد تخفیف را نیز گوشزد کند.**

**حل:**

متغیرهای لازم:

| متغیر | نام متغیر | نوع متغیر |
|---|---|---|
| نام | name | string |
| معدل | avg | real |
| جنسیت | jens | char ( 'm' مرد ، 'z' زن ) |
| وضعیت تاهل | taahol | char ( 'y' متاهل، 'n' مجرد ) |
| تعداد فرزندان | nfrz | integer |
| تعداد واجدین تخفیف | nvjd | integer |

```
Var    name : string ;
       Avg  : real ;
        Taahol ,jens : char ;
       nfrz , nvjd : integer ;
Begin
      nvjd : = 0 ;
      repeat
         writeln ( ' enter the name ' ) ;
         readln ( name );
       if name = ' finish '  then
              break  ;
       writeln ( ' enter jensiat ( z or m ?)');
       readln ( jens );
        if jens = ' z ' then
              continue  ;
       Writeln ( ' enter average ' ) ;
       Readln ( avg );
       If  avg < 16 then
              Continue ;
       Writeln ( ' taahol , yes or no ? ' );
       Readln ( taahol );
       If taahol = ' n '   then
                     Continue ;
       Writeln ( ' chand farzand ? ' ) ;
       Readln ( nfrz );
       If  nfrz <= 2 then
                     Continue ;
       Writeln ( ' takhfif migiri ');
       nvjd : = nvjd +1
```

```
  Until  false ;   { تا همیشه }
  Writeln ( nvjd , ' nafar  takhfif  gereftand ')
End .
```

**نکته : اگر بلاک یک if شامل دستوراتی مانند continue و break باشد بلاک else آن if نیازی به محصور شدن و ذکر کلمه کلیدی else ندارد.**

**مثال: خطاهای برنامه زیر را بیابید :**

```
1)      Var   3pt , ali  : integer
2)                min ; avg : real ;
3)                x , y , z : char  ;
4)      begin
5)                writeln ( enter a number  ) ;
6)      read (x , avj *2 );
7)      if  x <> avg then
8)      break ;
9)      else
10)     x + y := 2 ;
11)     writeln ( '1+1=3 ')
12)     while true   do  y : = '  a  ' ;
13)     z = r ;
14)     ali : = 8/2 ;
15)     end ;
```

**حل : خطاها (ERRORS) عبارتند از: (شماره خط هر خطا ذکر شده)**

۱) نام 3pt نام معتبری نیست ـ آخر خط ; مورد نیاز

۲) به جای ; باید , گذاشت

۳) ـ

۴) ـ

۵) کوتیشن جا افتاده است

۶) عبارت نمی تواند بین پرانتز های read استفاده شود

۷) نوع x و avg متفاوت است و قابل مقایسه با هم نیستند

۸) break فقط در حلقه مجاز است ، ضمن اینکه قبل از else سمی کالون جایز نیست

۹) ـ

۱۰) x + y نام متغیر نیست و نمی‌تواند سمت چپ دستور انتساب واقع شود

۱۱) ; می خواهد

۱۲) ـ

۱۳) r تعریف نشده است

۱۴) عدم تطابق نوع در دستور انتساب ( چون حاصل 8/2 یک عدد real خواهد بود در integer قرار نخواهد گرفت.)

۱۵) به جای ; نقطه مورد نیاز است .