برنامه نویسی به زبان

پرل

مرجع کامل

تالیف
مهندس هادی کیامرثی

تمام مثال های موجود در این کتاب با کامپیوتر تست شده اند تا از هر گونه خطا مبرا باشند با این حال ممکن است باز هم خطاهایی در آن وجود داشته باشد از کلیه خوانندگان این کتاب ، اساتید و دانشجویان محترم خواهشمندم برای مطلع کردن مولف از این خطا ها لطفا با ایمیل آدرس زیر تماس بگیرند

hadikiamarsi@gmail.com

فصل هشتم

در این فصل مطالب زیر را خواهید آموخت

حلقه بی نهایت

حلقه while

حلقه until

حلقه for

حلقه foreach

حلقه do...while

حلقه تو در تو

دستور next

دستور last

دستور continue

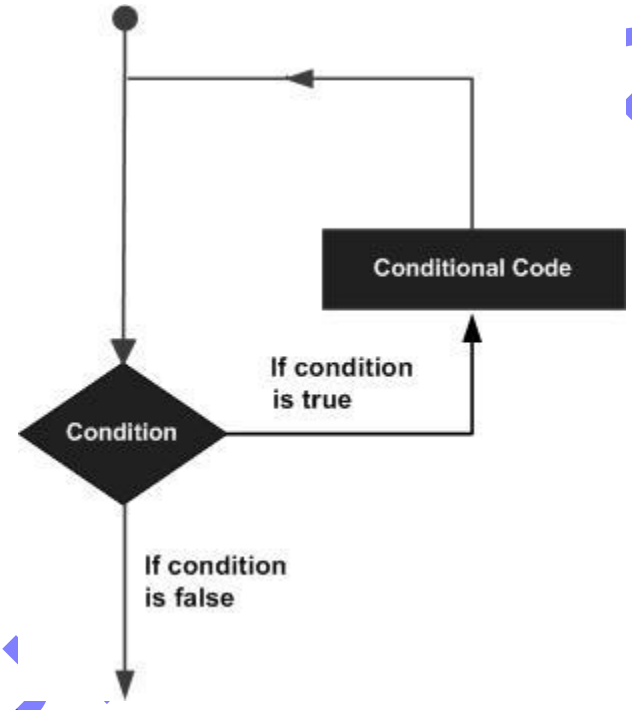
دستور redo

دستور goto

در برنامه نویسی خیلی وقت ها در وضعیتی قرار می گیرید که نیاز دارید یک دسته دستور را چندین و چند بار اجرا نمایید در چنین موقعیت هایی از حلقه ها استفاده می گردد . حلقه ها این توانایی را دارند که هر چند بار که برنامه نویس بخواهد یک تعداد دستور را اجرا نمایند .

زبان برنامه نویسی پرل از چندین دستور ایجاد کننده حلقه پشتیبانی می کند که بنا به نیازتان می توانید از آنها استفاده نمایید . همه دستورات حلقه به تفصیل در این فصل توضیح داده خواهند شد .

در زیر نمودار حلقه ها آورده شده است .



به یاد داشته باشید حلقه ها در درون خودشان دارای دستورات شرطی و کنترلی می باشند بنابراین شما می توانید از آنها بجای دستورات شرطی نیز استفاده نمایید به عبارتی شما می توانید برنامه ای بنویسید که هیچ دستور شرطی نداشته باشد و فقط از دستورات حلقه در آن استفاده شده باشد .

حلقه بی نهایت

برای بعضی کاربردها (طراحی سیستم عامل ، برنامه نویسی سوکت ، درایور نویسی) نیاز می شود که دستور حلقه تا بی نهایت اجرا گردد برای پیاده سازی چنین حلقه ای در زبان برنامه نویسی پرل مانند مثال زیر عمل می نماییم .

```
#!/usr/local/bin/perl

for( ; ; ) {
   printf "This loop will run forever.\n";
}
```

در مثال بالا جمله This loop will run forever بی نهایت بار در صفحه چاپ می گردد .
لازم به ذکر است برای متوقف کردن یک حلقه بی نهایت می توانید از کلید های Ctrl + C استفاده نمایید .

حلقه while

این دستور سریع ترین دستور برای پیاده سازی حلقه در زبان برنامه نویسی پرل می باشد اگر نیاز دارید برنامه تان با سرعت زیادی اجرا گردد از این حلقه استفاده نمایید .

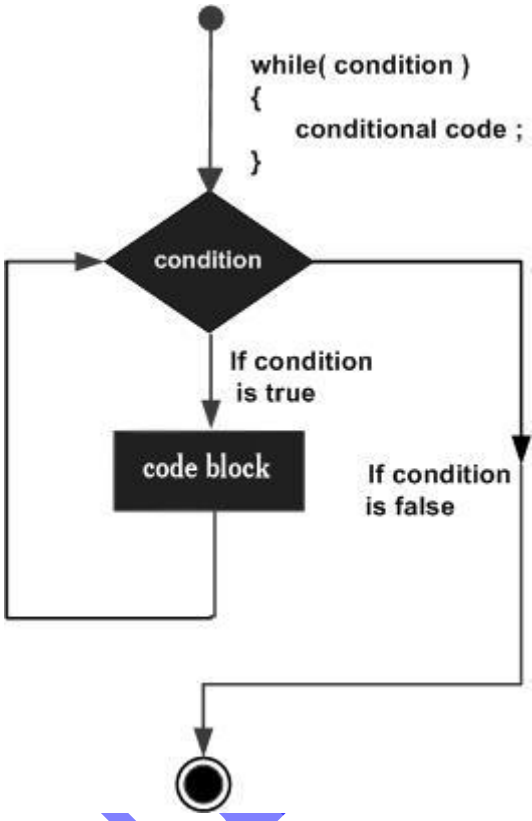
دستور نحوی

نحوه پیاده سازی این دستور در زبان برنامه نویسی پرل در زیر آورده شده است

```
while(condition) {
   statement(s);
}
```

در این دستور تا زمانی که عبارت شرطی condition بر قرار باشد دستورات statement(s) اجرا می گردند

نمودار راهنما



مثال

```
#!/usr/local/bin/perl

$a = 10;

# while loop execution
while( $a < 20 ) {
   printf "Value of a: $a\n";
   $a = $a + 1;
}
```

اجرای کد بالا نتیجه زیر را ظاهر خواهد نمود

```
Value of a: 10
Value of a: 11
Value of a: 12
Value of a: 13
Value of a: 14
Value of a: 15
Value of a: 16
Value of a: 17
Value of a: 18
Value of a: 19
```

در مثال بالا تا زمانی که مقدار متغیر $a کوچکتر از عدد 20 هست حلقه دستور چاپ عبارت Value of a و افزایش مقدار متغیر $a را انجام می دهد

حلقه until

این نوع حلقه مانند حلقه while می باشد با این تفاوت که در حلقه while حلقه تا زمانی که شرطش نقض بشه اجرا می شود ولی در حلقه until از ابتدا شرط اشتباه است و حلقه تا زمانی که شرط درست بشود اجرا می گردد .

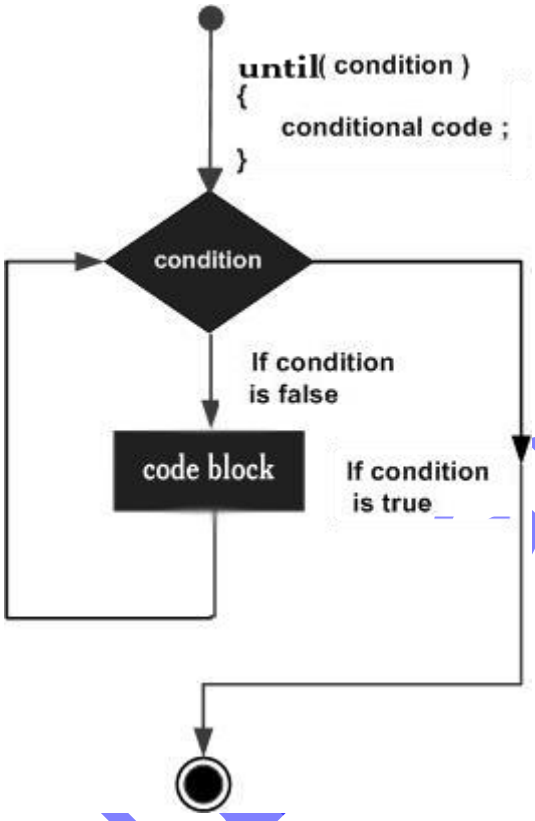
دستور نحوی

نحوه پیاده سازی این دستور در زبان برنامه نویسی پرل در زیر آورده شده است

```
until(condition) {
   statement(s);
}
```

در دستور نحوی بالا تا زمانی که عبارت شرطی condition بر قرار نباشد دسته دستورات statement(s) اجرا می گردد ولی همین که عبارت شرطی condition بر قرار شد حلقه متوقف می گردد

نمودار راهنما



مثال

```
#!/usr/local/bin/perl

$a = 5;

# until loop execution
until( $a > 10 ) {
   printf "Value of a: $a\n";
   $a = $a + 1;
}
```

اجرای کد بالا نتیجه زیر را ظاهر خواهد نمود

```
Value of a: 5
Value of a: 6
Value of a: 7
Value of a: 8
Value of a: 9
Value of a: 10
```

در مثال بالا تا زمانی که مقدار متغیر $a از عدد 10 کوچکتر باشد دستور چاپ عبارت Value of a: انجام می پذیرد .

for حلقه

این دستور حلقه یکی از پرکاربردترین دستورات حلقه می باشد چرا که در حین تعریف حلقه می توان گام افزایش یا کاهش حلقه را تعیین کرد و ضمنا این حلقه در بسیاری از زبان های برنامه نویسی دیگر مانند C ، PHP و .. وجود دارد و برنامه نویسان آشنایی بیشتری با این دستور ایجاد حلقه دارند .

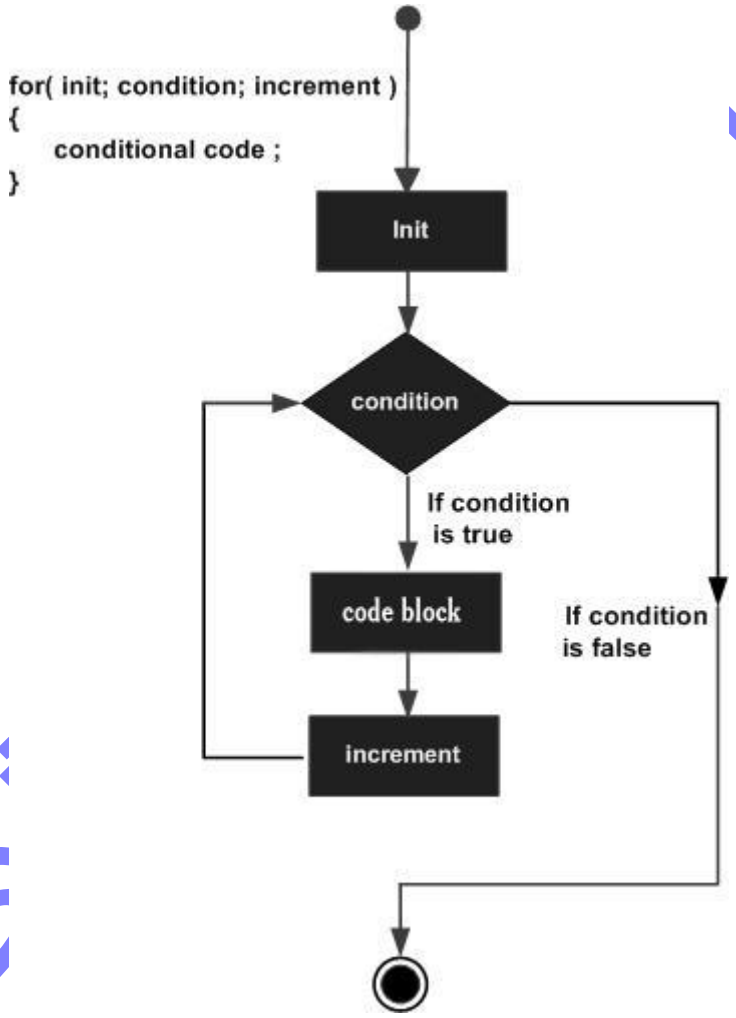
دستور نحوی

نحوه پیاده سازی این دستور در زبان برنامه نویسی پرل در زیر آورده شده است

```
for ( init; condition; increment ) {
   statement(s);
}
```

در دستور نحوی بالا عبارت init نام متغیر کنترل کننده گام حلقه می باشد و عبارت condition شرط حلقه می باشد و عبارت increment میزان گام افزایش یا کاهش حلقه می باشد . لازم به ذکر است عبارت شرطی موجود در حلقه for مانند حلقه while باید بر قرار باشد تا حلقه اجرا گردد و به محظ نقض شرط حلقه متوقف می گردد .

نمودار راهنما



مثال

```
#!/usr/local/bin/perl

# for loop execution
for( $a = 10; $a < 20; $a = $a + 1 ) {
   print "value of a: $a\n";
}
```

اجرای کد بالا نتیجه زیر را ظاهر خواهد نمود

```
value of a: 10
value of a: 11
value of a: 12
value of a: 13
value of a: 14
value of a: 15
value of a: 16
value of a: 17
value of a: 18
value of a: 19
```

حلقه foreach

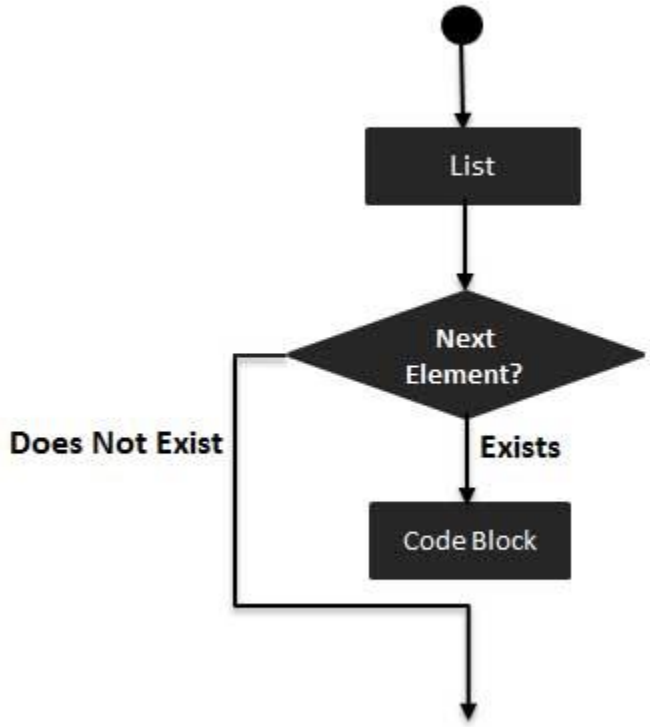
این دستور مشابه دستور for می باشد به این تفاوت که بیشتر برای آرایه ها ، لیست ها ، هش ها بکار می رود

دستور نحوی

نحوه پیاده سازی این دستور در زبان برنامه نویسی پرل در زیر آورده شده است

```
foreach var (list) {
...
}
```

نمودار راهنما



مثال

```
#!/usr/local/bin/perl

@list = (2, 20, 30, 40, 50);

# foreach loop execution
foreach $a (@list) {
   print "value of a: $a\n";
}
```

اجرای کد بالا نتیجه زیر را ظاهر خواهد نمود

```
value of a: 2
value of a: 20
value of a: 30
value of a: 40
value of a: 50
```

حلقه do...while

در تمام دستورات حلقه هایی که در بالا توضیح داده شد شرط در ابتدای حلقه بررسی می شد ولی در دستور حلقه **do...while** شرط در انتهای حلقه بررسی می گردد به این معنا که حلقه شما حداقل حتما یکبار اجرا می گردد

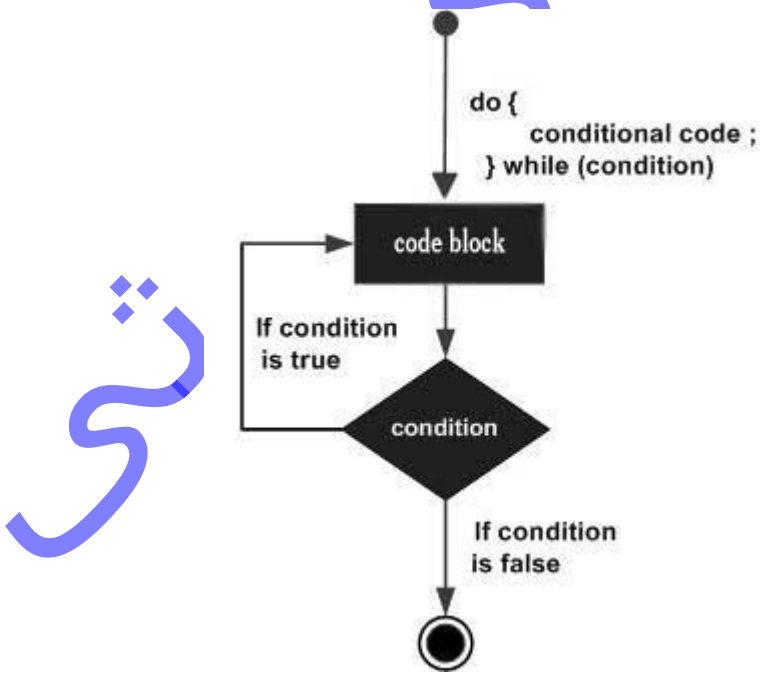
دستور نحوی

نحوه پیاده سازی این دستور در زبان برنامه نویسی پرل در زیر آورده شده است

```
do {
   statement(s);
}while( condition );
```

در دستور نحوی بالا ابتدا دستورات statement(s) اجرا می گردند سپس عبارت شرطی condition بررسی می گردد لازم به ذکر است در این حلقه نیز مانند حلقه while عبارت شرطی condition باید برقرار باشد تا اجرای حلقه تداوم بیابد و در صورت نقض شرط حلقه متوقف می گردد

نمودار راهنما



مثال

```
#!/usr/local/bin/perl

$a = 10;

# do...while loop execution
do{
   printf "Value of a: $a\n";
   $a = $a + 1;
}while( $a < 20 );
```

اجرای کد بالا نتیجه زیر را ظاهر خواهد نمود

```
Value of a: 10
Value of a: 11
Value of a: 12
Value of a: 13
Value of a: 14
Value of a: 15
Value of a: 16
Value of a: 17
Value of a: 18
Value of a: 19
```

حلقه تو در تو

در برنامه نویسی مواقعی پیش می آید که برنامه نویس نیاز دارد از چند حلقه برنامه نویسی به صورت تو در تو استفاده نماید . این یک دستور حلقه جدید نیست بلکه از همان دستورات توضیح داده شده در بالا استفاده می گردد .

دستور نحوی

نحوه پیاده سازی حلقه تو در تو با دستور for در زیر آورده شده است

```
for ( init; condition; increment ) {
   for ( init; condition; increment ) {
      statement(s);
   }
   statement(s);
}
```

نحوه پیاده سازی حلقه تو در تو با دستور while در زیر آورده شده است

```
while(condition) {
   while(condition) {
      statement(s);
   }
   statement(s);
}
```

نحوه پیاده سازی حلقه تو در تو با دستور do..while در زیر آورده شده است

```
do{
   statement(s);
   do{
      statement(s);
   }while( condition );

}while( condition );
```

نحوه پیاده سازی حلقه تو در تو با دستور until در زیر آورده شده است

```
until(condition) {
   until(condition) {
      statement(s);
   }
   statement(s);
}
```

نحوه پیاده سازی حلقه تو در تو با دستور foreach در زیر آورده شده است

```
foreach $a (@listA) {
   foreach $b (@listB) {
      statement(s);
   }
   statement(s);
}
```

مثال

```
#/usr/local/bin/perl

$a = 0;
$b = 0;

# outer while loop
while($a < 3) {
   $b = 0;
   # inner while loop
   while( $b < 3 ) {
      print "value of a = $a, b = $b\n";
      $b = $b + 1;
   }
   $a = $a + 1;
   print "Value of a = $a\n\n";
}
```

اجرای کد بالا نتیجه زیر را ظاهر خواهد نمود

```
value of a = 0, b = 0
value of a = 0, b = 1
value of a = 0, b = 2
Value of a = 1

value of a = 1, b = 0
value of a = 1, b = 1
value of a = 1, b = 2
Value of a = 2

value of a = 2, b = 0
value of a = 2, b = 1
value of a = 2, b = 2
Value of a = 3
```

دستور next

با این دستور نیز می توانید یک حلقه بسازید . دستور next یک منتقل کننده روند اجرای برنامه می باشد براحتی با این دستور می توانید روند اجرای برنامه را به هر نقطه ای از برنامه که بخواهید منتقل نمایید .

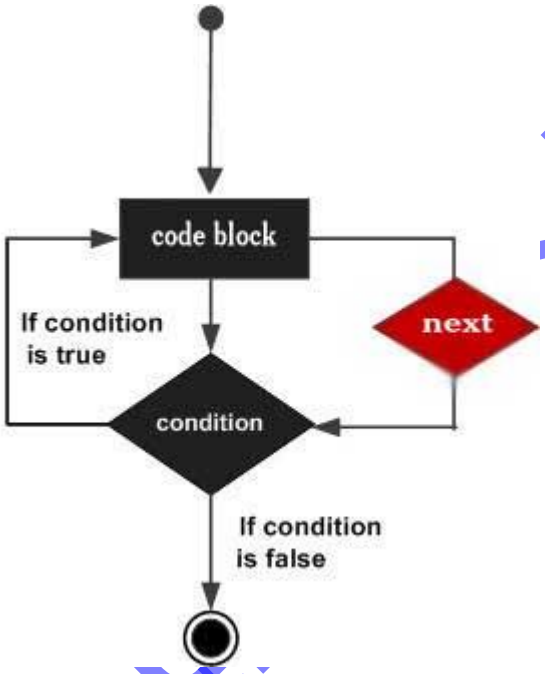
دستور نحوی

نحوه پیاده سازی این دستور در زبان برنامه نویسی پرل در زیر آورده شده است

```
next [ LABEL ];
```

در دستور بالا دستور next روند اجرای برنامه را به نقطه ای که در [LABEL] مشخص شده است منتقل می نماید . اگر در جلوی دستور next نقطه ای برای انتقال روند برنامه مشخص نشده باشد دستور next روند اجرای برنامه را به آولین حلقه منتقل می نماید .

نمودار راهنما



مثال

```
#!/usr/local/bin/perl

$a = 10;
while( $a < 20 ) {
   if( $a == 15) {
      # skip the iteration.
      $a = $a + 1;
      next;
   }
   print "value of a: $a\n";
   $a = $a + 1;
}
```

اجرای کد بالا نتیجه زیر را ظاهر خواهد نمود

```
value of a: 10
value of a: 11
value of a: 12
value of a: 13
value of a: 14
value of a: 16
value of a: 17
value of a: 18
value of a: 19
```

به مثال زیر توجه نمایید

```
#!/usr/local/bin/perl

$a = 0;
OUTER: while( $a < 4 ) {
   $b = 0;
   print "value of a: $a\n";
   INNER:while ( $b < 4) {
      if( $a == 2) {
         $a = $a + 1;
         # jump to outer loop
         next OUTER;
      }
      $b = $b + 1;
      print "Value of b : $b\n";
   }
   print "\n";
   $a = $a + 1;
}
```

اجرای کد بالا نتیجه زیر را ظاهر خواهد نمود

```
value of a : 0
Value of b : 1
Value of b : 2
Value of b : 3
Value of b : 4

value of a : 1
Value of b : 1
Value of b : 2
Value of b : 3
Value of b : 4

value of a : 2
value of a : 3
Value of b : 1
Value of b : 2
Value of b : 3
Value of b : 4
```

دستور last

این دستور روند اجرای یک حلقه را متوقف می نماید . این دستور درون هر نوع دستور حلقه بکار رود روند اجرای حلقه مورد نظر را متوقف می نماید . شما همچنین می توانید برای این دستور مشخص نمایید پس از متوقف کردن حلقه روند اجرای برنامه را به کجا منتقل نماید .

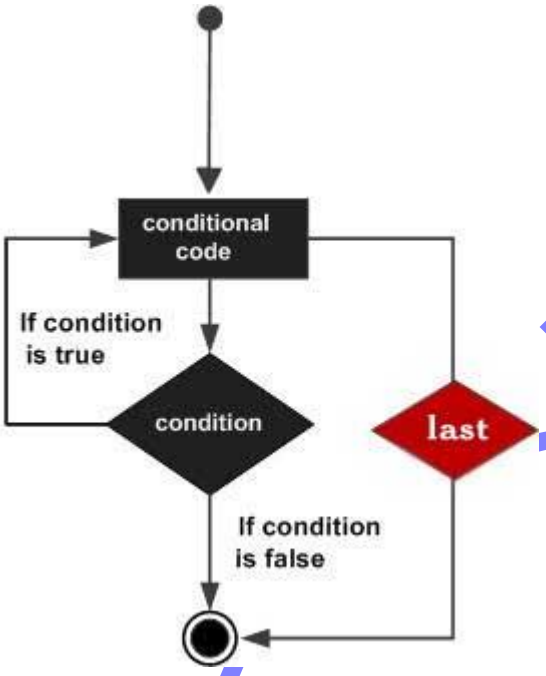
دستور نحوی

نحوه پیاده سازی این دستور در زبان برنامه نویسی پرل در زیر آورده شده است

```
last [LABEL];
```

در دستور بالا پس از متوقف شدن حلقه روند اجرای برنامه به نقطه [LABEL] منتقل می شود

نمودار راهنما



مثال اول

```
#!/usr/local/bin/perl

$a = 10;
while( $a < 20 ) {
   if( $a == 15) {
      # terminate the loop.
      $a = $a + 1;
      last;
   }
   print "value of a: $a\n";
   $a = $a + 1;
}
```

اجرای کد بالا نتیجه زیر را ظاهر خواهد نمود

```
value of a: 10
value of a: 11
value of a: 12
value of a: 13
value of a: 14
```

مثال دوم

به مثال زیر توجه نمایید

```
#!/usr/local/bin/perl

$a = 0;
OUTER: while( $a < 4 ) {
   $b = 0;
   print "value of a: $a\n";
   INNER:while ( $b < 4) {
      if( $a == 2) {
         # terminate outer loop
         last OUTER;
      }
      $b = $b + 1;
      print "Value of b : $b\n";
   }
   print "\n";
   $a = $a + 1;
}
```

اجرای کد بالا نتیجه زیر را ظاهر خواهد نمود

```
value of a : 0
Value of b : 1
Value of b : 2
Value of b : 3
Value of b : 4

value of a : 1
Value of b : 1
Value of b : 2
Value of b : 3
Value of b : 4

value of a : 2
```

دستور continue

این دستور چند کاربرد مختلف دارد ولی یکی از مهمترین کاربرد آن خنثی کردن دستور last می باشد به عبارتی اگر روند اجرای یک حلقه بوسیله دستور last متوقف شده باشد با این دستور می توان اجرای آن را از سر گرفت . ولی مهمترین کاربرد این دستور در حلقه ها می باشد زمانی که روند اجرای یک حلقه بوسیله نقض شرط آن متوقف شده باشد .

دستور نحوی

نحوه پیاده سازی دستور continue در دستور while در زیر آورده شده است

```
while(condition) {
   statement(s);
} continue {
   statement(s);
}
```

نحوه پیاده سازی دستور continue در دستور foreach در زیر آورده شده است

```
foreach $a (@listA) {
   statement(s);
} continue {
   statement(s);
}
```

نحوه پیاده سازی دستور continue در زیر آورده شده است

```
continue {
   statement(s);
}
```

مثال

در مثال زیر حلقه **for** با استفاده از دستور **while** پیاده سازی شده است

```
#/usr/local/bin/perl

$a = 0;
while($a < 3) {
   print "Value of a = $a\n";
} continue {
   $a = $a + 1;
}
```

اجرای کد بالا نتیجه زیر را ظاهر خواهد نمود

```
Value of a = 0
Value of a = 1
Value of a = 2
```

در مثال زیر کاربرد دستور **continue** را در حلقه **foreach** خواهید دید

```
#/usr/local/bin/perl
```

```
@list = (1, 2, 3, 4, 5);
foreach $a (@list) {
   print "Value of a = $a\n";
} continue {
   last if $a == 4;
}
```

اجرای کد بالا نتیجه زیر را ظاهر خواهد نمود

```
Value of a = 1
Value of a = 2
Value of a = 3
Value of a = 4
```

دستور redo

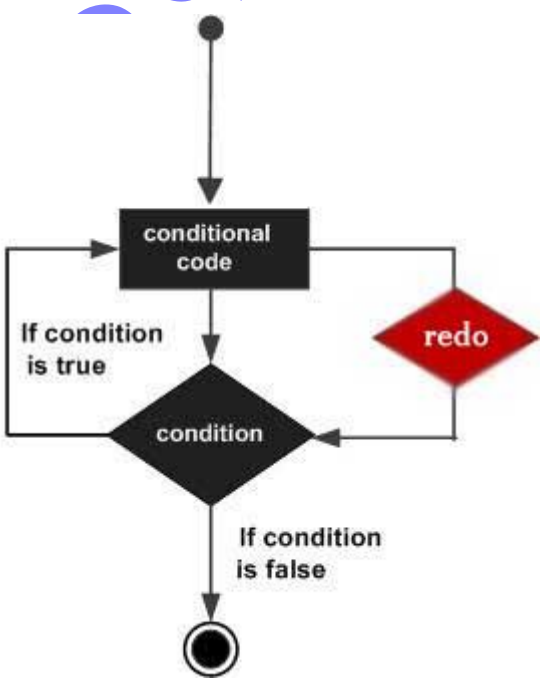
این دستور روند اجرای یک حلقه را به ابتدای آن منتقل می نماید

دستور نحوی

نحوه پیاده سازی این دستور در زبان برنامه نویسی پرل در زیر آورده شده است

```
redo [LABEL]
```

نمودار راهنما



مثال

در مثال زیر کاربرد دستور **redo** در زیر آورده شده است

```
#/usr/local/bin/perl

$a = 0;
while($a < 10) {
   if( $a == 5 ) {
      $a = $a + 1;
      redo;
   }
   print "Value of a = $a\n";
} continue {
   $a = $a + 1;
}
```

اجرای کد بالا نتیجه زیر را ظاهر خواهد نمود

```
Value of a = 0
Value of a = 1
Value of a = 2
Value of a = 3
Value of a = 4
Value of a = 6
Value of a = 7
Value of a = 8
Value of a = 9
```

دستور goto

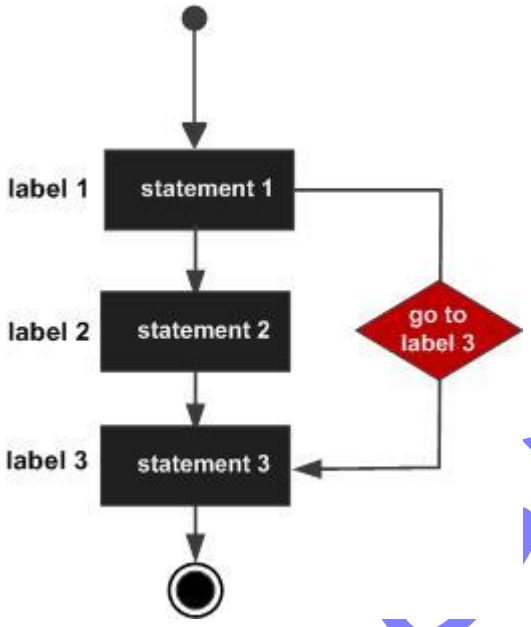
این دستور روند اجرای برنامه را به هر نقطه از برنامه که برنامه نویس بخواهد منتقل می نماید این دستور به عنوان آرگومان می تواند یک اسم (**LABEL**) یا یک عبارت دستوری (**EXPR**) و یا نام یک تابع (**&NAME**) را بپذیرد .

دستور نحوی

نحوه پیاده سازی این دستور در زبان برنامه نویسی پرل در زیر آورده شده است

```
goto LABEL

or

goto EXPR

or

goto &NAME
```

نمودار راهنما



مثال

در مثال زیر کاربرد دستور **goto** نشان داده شده است

```
#/usr/local/bin/perl

$a = 10;

LOOP:do {
   if( $a == 15) {
      # skip the iteration.
      $a = $a + 1;
      # use goto LABEL form
      goto LOOP;
   }
   print "Value of a = $a\n";
   $a = $a + 1;
} while( $a < 20 );
```

اجرای کد بالا نتیجه زیر را ظاهر خواهد نمود

```
Value of a = 10
Value of a = 11
Value of a = 12
Value of a = 13
Value of a = 14
Value of a = 16
Value of a = 17
Value of a = 18
Value of a = 19
```

در مثال زیر کاربرد دستور goto به همراه یک عبارت دستوری به عنوان آرگومان نشان داده شده است

```
#/usr/local/bin/perl

$a = 10;
$str1 = "LO";
$str2 = "OP";

LOOP:do {
   if( $a == 15) {
      # skip the iteration.
      $a = $a + 1;
      # use goto EXPR form
      goto $str1.$str2;
   }
   print "Value of a = $a\n";
   $a = $a + 1;
} while( $a < 20 );
```

اجرای کد بالا نتیجه زیر را ظاهر خواهد نمود

```
Value of a = 10
Value of a = 11
Value of a = 12
Value of a = 13
Value of a = 14
Value of a = 16
Value of a = 17
Value of a = 18
Value of a = 19
```