{desvios e loops}

11.1 IF/ELSE

Para executar algo condicionalmente o PERL tem a expressão:

```
if (verdadeiro)
{
 faz-tal-coisa;
 faz-tal-coisa;
}
```

Para uma expressão ser verdadeira NÃO pode retornar ZERO ou VAZIO (Undef). Qualquer outro valor é válido como **true**. Zero ou Undef equivalem a:**false**. O valor equivalente a **true** normalmente é: 1.

Podemos juntar duas ou mais condições usando os símbolos:

```
&&      and lógico
||      or lógico
```

Para um desvio alternativo temos:

```
if (verdadeiro)
{
 faz-tal-coisa1;
 faz-tal-coisa2;
}
else
{
 faz-tal-coisa3;
 faz-tal-coisa4;
 }
```

Podemos ter várias opções:

```
    if (verdadeiro1)
    {
     faz-tal-coisa1;
     faz-tal-coisa2;
    }
    elseif (verdadeiro2)
    {
     faz-tal-coisa3;
     faz-tal-coisa4;
    }
    else
    {
     faz-tal-coisa5;
  }
```

11.2 UNLESS

Para fazer determinadas coisas exceto se uma opção for verdadeira temos:

```
unless (verdadeiro)
{
  faz-tal-coisa;
}
```

11.3 LOOPS

Para repetir determinadas coisas enquanto um teste for verdadeiro:

```
while(verdadeiro)
{
  faz-tal-coisa;
}
```

Para o loop ser feito até que um teste retorne verdadeiro:

```
until(verdadeiro)
{
  faz-tal-coisa;
}
```

Um caso muito comum de loop é quando temos uma variável que vai sendo incrementada e testamos se ela chegou a um limite. O PERL tem a expressão:

```
for(assinalamento-inicial; teste; incremento)
{
  faz-tal-coisa;
}
```

Veja um exemplo simples, para listar números de 0 até 5:

```
for($i=0;$i<=5;$i=$i+1)
{
 print "$i \n";
}
```

Podemos usar: $i++ no lugar de: $i = $i + 1 , para alegria do pessoal do C++.

11.4 FOREACH

Uma outra construção interessante é com foreach. Repetimos várias linhas de código, substituindo uma determinada variável ($x, por exemplo) pelos valores de uma lista. Assim:

```
foreach $x (1,2,3,4)
{
  $y = 4*$x;
  print  "$y \n";
}
```

listaria os números: 4,8,12,16.

No lugar da lista poderíamos usar um array.