{ arrays}

10.1 CONCEITOS INICIAIS

Um array é um conjunto ordenado de valores.

Quando tem uma só dimensão (que é o que vamos ver nesse capítulo) podemos chamar um array de vetor.

Podemos misturar em um array números e strings ums vez que para o PERL todos são igualmente variáveis escalares.

O nome de um array é sempre precedido de **@**.

Criamos um array assim:

```
@conj = (25, 14, 37);
```

Para acessar um elemento do array usamos o sinal **$** antes do nome do array e a indexação é feita a partir de ZERO. Assim:

```
$conj[1]     seria 14.
```

Podemos ter um novo array constituido de elementos de outro array:

```
@arr = @conj[0,2];  teria os valores 25 e 37.
```

Se assinalarmos um array para uma variável essa vai conter a quantidade de elementos do array.Em:

```
$qtd = @arr;   teríamos 2 na variável $qtd.
```

O índice do último elemento de um array é designado assim, por exemplo:

```
$#conj  teria 2 (lembre-se que a origem é ZERO!).
```

10.2 ACRESCENTANDO E TIRANDO

Para acrescentar um elemento no fim de um array temos o operador **push( )**.

Ele tem dois argumentos: o primeiro é o nome do array e o segundo é o valor a ser acrescentado.Por exemplo:

```
push(@conj,88);
```

Para tirar um elemento no fim de um array temos um operador chamado **pop( )** que funciona como uma função com retorno pois tira o elemento e o retorna. Assim, podemos fazer:

```
$ult = pop(@conj);
```

Em $ult teríamos o último elemento de @conj.

Para acrescentar um elemento no início de um array temos o operador **unshift( )** e para tirar temos: **shift( )**.

10.3 CLASSIFICANDO

Podemos classificar ascendentemente um array com strings usando **sort( )** que retorna um novo array classificado mas não altera o original. Vamos ver mais a frente como classificamos um array com números.

Existe um operador **reverse( )** que retorna os elementos na ordem inversa, o que pode ser usado após o sort(), para uma classificação descendente.

10.4 OPERADOR CONSTRUTOR DE ARRAY

Na criação de um array, se colocarmos dois valores separados por **dois pontos** (não dois-pontos) teremos no array elementos que começarão pelo valor da esquerda e terão incremento de **uma unidade** até chegar ao valor mais próximo do elemento da direita. Assim:

```
@cj1 = (1 .. 4)  será:  (1, 2, 3, 4)
@cj2 = (3.4 .. 6.3) será:  (3.4,  4.4,  5.4)
```

10.5 ARRAY ASSOCIATIVO

Em PERL temos um outro tipo de array que é muito usado nas operações com CGI. Chama-se **array associativo**.

Em outras linguagens (Java, por exemplo) temos coisa semelhante e chamamos de **hashtable**. Enfim, o que temos é uma tabela com duas colunas sendo que podemos achar um valor da segunda coluna se dermos a **chave** ou melhor dizendo, o valor equivalente da primeira coluna.Qualquer coisa assim:

```
001 Bola
002 Boneca
003 Trem
```

Para criarmos um array associativo existem muitas maneiras. Vamos mostrar uma que talvez não seja a mais simples mas que vai nos ajudar a compreender certas coisas no futuro (tenha confiança...)

Para criarmos um array associativo chamado **campos** com os dados aí de cima escreveríamos no programa PERL:

```
($chave,$valor)=("001","Bola");
$campos{$chave}=$valor;
```

```
($chave,$valor)=("002","Boneca");
$campos{$chave}=$valor;
($chave,$valor)=("003","Trem");
$campos{$chave}=$valor;
```

Repare (isto é importante) que às vezes temos parênteses e às vezes temos **chaves** - { }.

Se quizermos "achar" o valor "Boneca" e imprimí-lo bastaria usar:

```
print $campos{"002"};
```

Seria impresso: **Boneca**.

A maneira "clássica" de criar esse array associativo seria:

```
%campos=("001","Bola","002","Boneca","003","Trem");
```

Um array associativo importante no CGI é o **%ENV** que tem como uma de suas chaves **QUERY_STRING**.

Correspondendo a esta chave temos como valor, tudo que é enviado depois da **?** numa chamada a um programa PERL usando CGI.