

Artigo

[Emily Scoparo](#) · Set. 2, 2022 5min de leitura

UTILIZANDO ALGUMAS FUNÇÕES DO COS

Desde o início dos meus estudos, as funções do ObjectScript sempre me chamaram muita atenção, pois facilitam diversas atividades do dia-a-dia, como a manipulação de strings.

Separei as funções que acho mais úteis e vou explicar um pouco delas aqui:

Obs: nos exemplos utilizei rotinas, mas as funções podem ser usadas em métodos, procedimentos, entre outros.

1. \$FIND:

- Localiza uma substring dentro de uma string por valor;
- Retorna um inteiro correspondente à posição do primeiro caractere após a substring procurada;
- Se não localizar a substring dentro da string, retorna 0;
- Sintaxe: \$FIND(string, substring, posicaoinicial)
 - string: a string de pesquisa, a substring será procurada dentro dela;
 - substring: o que você deseja procurar dentro da string;
 - posicaoinicial: (opcional) posição que você deseja iniciar a pesquisa;
- Exemplos:

```
1 FIND
2     Set findPosic = $FIND("Amora", "Amor")
3     Write "Posição: " _findPosic
4
```

```
USER>d FIND^TestesFUNCOES
Posição: 5
```

- A substring "Amor" será procurada dentro da string "Amora";
- O valor "5" corresponde à posição do último "a" de "Amora", pois é o primeiro caractere após a substring procurada;
- Nesse caso, a função iniciou a pesquisa do primeiro caractere, pois não adicionamos o último argumento. Se definirmos o valor de posicaoinicial, a função inicia a pesquisa a partir da posição indicada. Exemplo:

```
1 FIND
2     Set findPosic = $FIND("Amora", "Amor", 2)
3     Write "Posição: " _findPosic
```

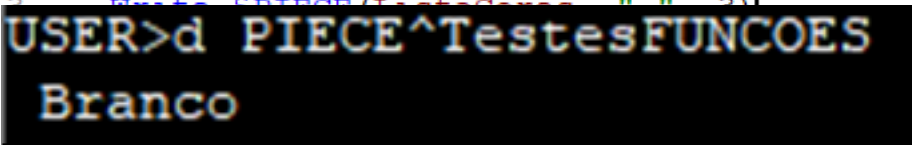
```
USER>d FIND^TestesFUNCOES
Posição: 0
```

- Quando definimos o argumento posicaoinicial como 2 indicamos que ele deve começar a busca a partir do segundo caractere da string, no nosso exemplo o "m";
- Sendo assim, a função está realizando a busca na string "mora", que não contém a substring "Amor", por isso retorna o valor 0;

2. \$PIECE:

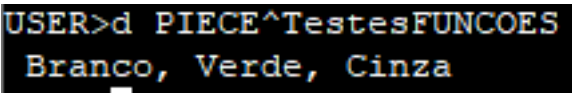
- Pode ser utilizada de 2 formas:
 - Retorna uma substring de uma string utilizando um delimitador, ou seja, se tenho uma lista separada por vírgulas, posso utilizar o \$PIECE para retornar algum elemento dessa lista definindo seu delimitador como “ , ” ;
 - Sintaxe: \$PIECE(string,delimitador,de,até)
 - string: string onde a função vai procurar a substring;
 - delimitador: delimitador utilizado para separar os elementos da string;
 - de – até : tem 2 formas: se utilizado sem definir o até, retorna o elemento correspondente ao inteiro especificado em de; se até for especificado, porém, retorna desde o elemento indicado em de até o indicado em até;
 - Exemplos:

```
1 $PIECE
2 Set ListaCores = "Azul, Amarelo, Branco, Verde, Cinza, Preto"
3 Write $PIECE(ListaCores, ",", 2)
```



Retorna o 2º elemento da string ListaCores;

```
1 $PIECE
2 Set ListaCores = "Azul, Amarelo, Branco, Verde, Cinza, Preto"
3 Write $PIECE(ListaCores, ",", 3, 5)
```



- Retornou os elementos 3, 4 e 5 da string ListaCores;

- Substitui uma substring dentro da string utilizando um delimitador, ou seja, se tenho uma lista separada por vírgulas, posso utilizar o \$PIECE para substituir algum elemento dessa lista definindo seu delimitador como “ , ” ;
 - Sintaxe: \$PIECE(string,delimitador,de,até) = valor
 - string: string onde a função vai procurar a substring;
 - delimitador: delimitador utilizado para separar os elementos da string;
 - de – até : tem 2 formas: se utilizado sem definir o até, substitui o elemento correspondente ao inteiro especificado em de; se até for especificado, porém, substitui desde o elemento indicado em de até o indicado em até;
 - valor: o valor que deve ser inserido na posição especificada na função, substituindo o valor antigo;
 - Exemplos:

```

1 PIECE
2   Set ListaCores = "Azul, Amarelo, Branco, Verde, Cinza, Preto"
3   Write !, "- Lista de cores: " _ListaCores
4   Set $PIECE(ListaCores, ",", 3) = " Rosa"
5   Write !, "- Lista de cores att: " _ListaCores, !
USER>d PIECE^TestesFUNCOES

```

```

- Lista de cores: Azul, Amarelo, Branco, Verde, Cinza, Preto
- Lista de cores att: Azul, Amarelo, Rosa, Verde, Cinza, Preto

```

A posição 3 da ListaCores foi substituída pela substring " Rosa";

```

1 PIECE
2   Set ListaCores = "Azul, Amarelo, Branco, Verde, Cinza, Preto"
3   Write !, "- Lista de cores: " _ListaCores
4   Set $PIECE(ListaCores, ",", 3, 5) = " Rosa, Ciano"
5   Write !, "- Lista de cores att: " _ListaCores, !
~

```

```

USER>d PIECE^TestesFUNCOES

```

```

- Lista de cores: Azul, Amarelo, Branco, Verde, Cinza, Preto
- Lista de cores att: Azul, Amarelo, Rosa, Ciano, Preto

```

- Os elementos 3, 4 e 5 foram substituídos pela substring " Rosa, Ciano";
- A função ignora a quantidade de elementos, então podemos substituir um ou mais elementos com outro ou outros n elementos;

3. \$REPLACE:

- Fornecemos uma string origem, onde a função vai realizar a pesquisa;
- Depois, adicionamos a substring que deve ser pesquisada dentro da string origem (pois é ela que será substituída);
- Depois devemos passar outra substring que ficará no lugar da substring pesquisada na string origem;
- **Sintaxe:** \$REPLACE(string,substringPesquisa,substringSubstituta,Inicio,Count,Case)
 - string: string origem (onde vai ser realizada a pesquisa);
 - substringPesquisa: a substring que deve ser pesquisada dentro da string, pois é ela quem vai ser substituída;
 - substringSubstituta: substring que deve ser colocada no lugar da substringPesquisa;
 - Inicio: (opcional) posição onde a pesquisa deve ser iniciada dentro da string;
 - Count: (opcional) número de substituições que devem ser feitas na string;
 - Case: (opcional) booleano indicando se a pesquisa deve diferenciar maiúsculas e minúsculas, sendo:
 - 0 (padrão): diferencia maiúsculas e minúsculas;
 - 1: não diferencia maiúsculas e minúsculas;
- Exemplos:

```

USER>d REPLACE^TestesFUNCOES

String: A cor é verde
Nova string: A cor é marrom
USER>d REPLACE^TestesFUNCOES

String: A cor é verde
Nova string: A cor é marrom

```

4. \$TRANSLATE:

- Retorna uma substring que é a substituição de vários caracteres por outros correspondentes;
- Devemos fornecer uma string origem e 2 substrings: uma que representa os caracteres a serem substituídos e uma que, sequencialmente, representa os caracteres que devem ser colocados no lugar dos substituídos;
- Sintaxe: `$TRANSLATE(string, ident, assoc);`
 - *string*: string origem, onde vai ser realizado a pesquisa dos *ident*;
 - *ident*: substring de caracteres que devem ser pesquisados e substituídos dentro de *string*;
 - *assoc*: caracteres correspondentes que devem ser colocados no lugar dos *ident*;
- Exemplos:

```

1 TRANSLATE
2   Set DataHoraNasc = "19-11-2001, às 13h45"
3   Write !, "Data e hora do nascimento: " _ DataHoraNasc
4   Write !, "Data e hora do nascimento att: " _ $TRANSLATE(DataHoraNasc, "-h", "/:")
USER>d TRANSLATE^TestesFUNCOES

Data e hora do nascimento: 19-11-2001, às 13h45
Data e hora do nascimento att: 19/11/2001, às 13:45
- Substituiu o caractere - pelo caractere / e o h pelo .,

```

5. \$EXTRACT:

- Pode ser utilizada de 2 formas:
 - Retorna uma substring de uma string por posição:
 - Sintaxe: `$EXTRACT(string, de, até)`
 - Exemplos:

```

1 EXTRACT
2   Set Data = "19/11/2001"
3   Write !, "Data: " _ Data
4   Write !, "Ano: " _ $EXTRACT(Data, 7, 10)
-

```

```
USER>d EXTRACT^TestesFUNCOES
```

```
Data: 19/11/2001
```

```
Ano: 2001
```

- Extraia os caracteres de 1 a 10 de Data;

- Substitui parte de uma string (definida por posição) por ua substring:

- Sintaxe: Set \$EXTRACT(string, de, até) = valor
- Exemplos:

```
1 EXTRACT
2   Set Data = "19/11/2001"
3   Write !, "Data: " _ Data
4   Set $EXTRACT(Data, 1, 2) = "20"
5   Write !, "Data att: " _ Data
USER>d EXTRACT^TestesFUNCOES
```

```
Data: 19/11/2001
```

```
Data att: 20/11/2001
```

- OBSERVAÇÕES:

- Essas são só algumas na vasta lista de funções úteis do ObjectScript, por isso vou deixar abaixo o link da lista de funções do COS: ->

<https://docs.intersystems.com/iris20211/csp/docbook/Doc.View.cls?KEY=RCO...>

URL de origem: <https://pt.community.intersystems.com/post/utilizando-algumas-fun%C3%A7%C3%B5es-do-cos>