Linguagem de programação Krauts

Manual do Programador

Krauts é uma linguagem de programação utilizando notação polonesa e uma sintaxe similar a de linguagens de montagem (Assembly).

Krauts é interpretada linha por linha; assim, o final de cada comando é reconhecido pelo caractere fim-de-linha, e não há como ter mais de um comando na mesma linha.

Os comandos da linguagem Krauts são **case-insensitive**, ou seja, não diferenciam letras maiúsculas e minúsculas. Nomes de variáveis e textos, contudo, são **case-sensitive**.

Por que notação polonesa?

A maioria de nós está acostumado com a notação infixa, que segue o modelo **operando1 operador operando2**, ou A + B, por exemplo. Isto implica trabalhar com parênteses e prioridade de operadores para evitar expressões ambíguas.

A notação polonesa, também conhecida como notação prefixa, segue o modelo **operador operando1 operando2**, ou + A B, assim evitando a necessidade de parênteses ou prioridade de operadores para resolver expressões.

Exemplo

Expressão em notação infixa: **`var = (5 - 6) × 7`**

Expressão em notação prefixa: **`= var * - 5 6 7`**

Expressão em Krauts: **`ATR var MUL SUB 5 6 7`**

Comentários

Comentários em Krauts são indicados pelo caractere "til" (**~**). Exemplo:

```
prt add 10 5 ~ Este é um comentário de linha
~ O valor impresso pela linha acima foi 15
```

Sintaxe da linguagem Krauts

Segue um resumo dos comandos da linguagem e suas respectivas sintaxes.

Comando	Utiliza-se no início da linha?	Resumo	Sintaxe
new	Sim	Declara uma nova variável do tipo double.	**NEW a** **new b**
atr	Sim	Atribui à variável **dest** o resultado da expressão **orig**.	**ATR a ADD 2 2** **atr b a**
prt	Sim	Imprime na saída padrão o resultado de uma expressão aritmética ou booleana, de acordo com o primeiro argumento passado. Caso o primeiro argumento tenha como prefixo o caractere aspas duplas (**“**), imprime todo o resto da linha como uma mensagem na tela.	**prt “Ola Mundo!** **prt MOD 10 3** **prt eq a b** **prt gt 10 5 ~True**
if	Sim	Controle de fluxo: se a expressão for verdadeira, executa o código das linhas que antecedem o comando **fi**.	**if <expr...>**
fi	Sim	Encerra o escopo do último comando **if**.	**fi**
while	Sim	Laço de repetição: executa o código das linhas que antecedem o comando **done** enquanto a expressão for verdadeira.	**while <expr...>**
done	Sim	Encerra o escopo do último comando **while**.	**done**
add	Não	Retorna a soma do resultado de duas expressões.	**ADD i 10** **add a b**
sub	Não	Retorna a subtração do resultado de duas expressões.	**sub op1 op2**
mul	Não	Retorna o produto do resultado de duas expressões.	**mul op1 op2**
div	Não	Retorna o quociente da divisão do resultado de duas expressões. Programa encerra em erro caso **op2** seja equivalente a zero.	**div op1 op2**
mod	Não	Retorna o resto da divisão do chão do resultado de duas expressões. Programa encerra em erro caso **op2** seja equivalente a zero.	**mod op1 op2**
eq	Não	Retorna 1 se **op1** é igual a **op2**, 0 caso contrário	**eq op1 op2** **if eq a b**
lt	Não	Retorna 1 se **op1** é menor que **op2**, 0 caso contrário	**lt op1 op2** **lt 10 8 ~ falso** **lt 20 30 ~ verd.**
gt	Não	Retorna 1 se **op1** é maior que **op2**, 0 caso contrário	**gt op1 op2**
leq	Não	Retorna 1 se **op1** é no máximo **op2**, 0 caso	**leq op1 op2**

		contrário	
geq	Não	Retorna 1 se **op1** é no mínimo **op2**, 0 caso contrário	**geq op1 op2**
neq	Não	Retorna 1 se **op1** é diferente de **op2**, 0 caso contrário	**neq op1 op2**
not	Não	Verifica o resultado da expressão e retorna-o, negado.	**not <expr...>**
and	Não	Retorna 1 se ambos **op1** e **op2** são diferentes de zero	**and op1 op2**
or	Não	Retorna 1 se pelo menos **op1** ou **op2** for diferente de zero	**or op1 op2**
xor	Não	Retorna 1 se **op1** for igual a zero e **op2** diferente de zero	**xor op1 op2**
nand	Não	Retorna 1 se pelo menos **op1** ou **op2** for igual a zero	**nand op1 op2**
nor	Não	Retorna 1 se ambos **op1** e **op2** forem iguais a zero	**nor op1 op2**
xnor	Não	Retorna 1 se **op1** e **op2** forem ambos iguais ou diferentes de zero	**xnor op1 op2**