Gabriel Henrique Rudey

# Linguagem Blank

Guia de Referência

# Introdução

A linguagem Blank é uma linguagem procedural que pode lembrar C ou Javascript. Construída com influências de Ruby, C, PHP e Javascript, tem aspectos únicos de cada uma.

Os Scripts de instrução com a linguagem Blank devem ser arquivos de texto sem codificação especial com extensões ".blk". Não há convenção de nomes dos arquivos para os scripts. Exemplo: "helloworld.blk".

O interpretador foi construído de forma versátil, então você não precisa se preocupar com a questão de quantidade de espaços entre as instruções, mas deve tomar cuidado para não concatenar instruções e fazer com que algum token não seja identificado.

# Instruções

Os delimitadores de instruções na Blank são os "fim de linha" (\n), portanto, não é necessário inserir um ";" no final de instruções, muito menos início de blocos com chaves "{", deste modo, Blank se caracteriza por "1 comando por linha", pois não é possível concatenar comandos, por exemplo, fazer várias declarações de variáveis e adicionar um loop:

```
var a var b var c loop(a < 10)
```

## Declaração

Para fazer uma declaração de variável na Blank, é utilizado o token "var", e separado do token, o nome da variável:

```
var a
var teste
var abacate
var qualquerCoisa
```

Lembre-se de não usar uma palavra reservada, pois esta não será considerada um nome válido e o interpretador irá identificar como função, podendo retornar um erro em seu código.

## Atribuição

Depois de criada sua variável, você poderá adicionar um valor á ela com o comando de atribuição, realizado pelo token "=":

```
var teste
teste = 10
teste = 12.3
teste = "minha string"
```

Blank não é fortemente tipada, então você pode adicionar e trocar o valor e tipo das variáveis como quiser. As atribuições válidas são:

```
Inteiros: 10, 20, 999
Números com Pontos Flutuante: 10.2, 1415.24
Strings: “texto”, “a”, “palavras e mais palavras”
```

Também é possível declarar a variável e diretamente incluir um valor para ela, fazer uma variável assumir o valor de outra ou então uma variável ser resultado de uma expressão, os seguintes exemplos são válidos:

```
var a = 10
var b
b = a // b tem o valor 10
var c = 10 + 10 // valor 20
var verdadeiro = 10 < 20 // valor 1.0
var falso = 20 < 10 // valor 0.0
```

## Comentários

A Blank também suporta comentários, ou seja, textos que serão ignorados em sua execução. Os comentários são identificados por duas barras: “//” seguido do texto do comentário e só funcionam para a linha em questão.

**Atenção:** não há um fim de bloco para o comentário, tudo o que for acrescentado após as duas barras será ignorado.

```
var teste // Declara uma variável
teste = 5 // Atribui 5 á variável teste
// var a = 10  esta linha nada faz
```

## Tela

É possível mostrar valores na tela pelo comando “show”, assim como no comando “var”, os atributos vem depois do token. Todo comando “show” automaticamente pula uma linha após o texto. Por exemplo:

```
var meuNumero = 10
show meuNumero // Mostra 10 na tela
show 12 // mostra 12 na tela
show “oi” // mostra oi na tela
show “oi, meuNumero é: ” . meuNumero // Concatenação
```

O resultado na tela será:

```
10
12
oi
oi, meuNumero é: 10
```

É possível concatenar inúmeros valores, e o token para concatenar informações como se fossem Strings é o " . " (neste caso o nosso operador de concatenação **precisa estar entre espaços**) entretanto, não é possível fazer concatenação pulando linhas:

```
show "oi " . "10 " . " 20 " . " 30"
      . " 40 " // Errado
```

## Leitura da Entrada Padrão

Para se ler da entrada padrão se utiliza o comando "ask", ao ser invocado, o programa esperará por uma entrada do usuário e por uma confirmação com o "enter". O comando ask pode ser aplicado em qualquer situação que envolver valores:

```
var a
show "Digite um valor para a variável 'a':"
a = ask
show "O valor de a é: " . a
show "Digite outro valor: "
a = 10 + ask
show "O valor digitado + 10 é: " . a
```

## Operações

Grande parte das operações conhecidas são identificadas pela Blank, desde matemáticas até operações lógicas.

As operações lógicas também geram um número como resultado, que futuramente significam um valor verdadeiro ou falso. O número 0 ou 0.0 é sempre considerado um resultado falso. Qualquer outro número utilizado que não seja 0 (zero) é considerado como resultado verdadeiro. As operações lógicas geram o número 1.0 para resultados verdadeiros e 0.0 para resultados falsos.

As operações matemáticas aceitas são:

| Operação | Token | Exemplo | Resultado |
|---|---|---|---|
| **Soma** | **+** | 10 + 20 | 30 |
| **Subtração** | **-** | 20 – 10 | 10 |
| **Multiplicação** | ***** | 2 * 2 | 4 |
| **Divisão** | **/** | 10 / 2 | 5 |
| **Mod (resto)** | **%** | 10 % 5 | 2 |

As operações lógicas aceitas são:

| Operação | Token | Exemplo | Resultado |
|---|---|---|---|
| **Maior** | **>** | 10 > 20<br>20 > 10 | 0.0<br>1.0 |
| **Menor** | **<** | 1 < 100<br>100 < 50 | 1.0<br>0.0 |
| **Maior ou Igual** | **>=** | 2 >= 2<br>1 >= 2 | 1.0<br>0.0 |
| **Menor ou Igual** | **<=** | 5 <= 5<br>6 <= 5 | 1.0<br>0.0 |
| **Igual** | **==** | 10 == 10<br>100 == 200 | 1.0<br>0.0 |
| **Diferente** | **!=** | 10 != 10<br>100 != 200 | 0.0<br>1.0 |
| **E (and)** | **&&** | 1.0 && 0.0<br>1.0 && 1.0 | 0.0<br>1.0 |
| **Ou (or)** | **\|\|** | 1.0 \|\| 0.0<br>0.0 \|\|01.0 | 1.0<br>0.0 |

É possível aninhar mais de uma operação, porém lembre-se da precedência! Funciona exatamente como as regras matemáticas, primeiramente *, / e % na ordem que aparecerem, depois + e -, e por último todas as operações lógicas na ordem em que aparecem.

Blank até o momento não suporta a sobreposição de precedência com parênteses.

```
var a = 1 + 2 + 3 * 4 // resultado 10
var b = 10 % 2 == 0 // resultado 1.0
```

## Controladores de Fluxo

Os controladores de fluxo funcionam de maneira muito próxima: precisam de uma expressão com um único resultado para que possam fazer o seu trabalho.

### Controlador if

O controlador “if” faz uma comparação da expressão dada, e caso verdadeira, executa o bloco seguinte, caso contrário, ignora o bloco de instruções até encontrar o token “else” ou o token que finaliza seu bloco “endif”.

```
if (10 < 20)
    show “10 é menor que 20!”
else
    // nunca entrará neste bloco
endif
```

```
if (20 == 10)
	// nunca entrará neste bloco
else
	show “20 não é igual á 10!”
endif

if (10 != 10)
	// Não faz nada
endif
```

É possível aninhar comandos “if” dentro de outro comando “if” e não há um número limitado para isso.

## Controlador loop

O controlador loop funciona com o mesmo princípio das operações que a instrução if funciona: é necessário um único valor indicando um resultado que será processado e será a decisão pela execução do código ou não.

```
var a = 0
// Será repetido até a variável a assumir valor 10
loop (a < 10)
	show “Vou ser mostrado 10 vezes!”
	a = a + 1
endloop

loop (20 == 10)
	// nunca entrará neste bloco
endloop

a = 10

loop (a > 0)
	show a
	a = a - 1
endloop

show “Fim!”
```

Este exemplo mostrará na tela:

```
Vou ser mostrado 10 vezes!
Vou ser mostrado 10 vezes!
Vou ser mostrado 10 vezes!
Vou ser mostrado 10 vezes!
Vou ser mostrado 10 vezes!
Vou ser mostrado 10 vezes!
Vou ser mostrado 10 vezes!
Vou ser mostrado 10 vezes!
Vou ser mostrado 10 vezes!
Vou ser mostrado 10 vezes!
10
9
8
7
6
5
4
3
2
1
Fim!
```