Lógica da Computação

jamescript

```
2     const result: integer = a + b / 2
3     return result
4   }
5
6   const x:integer = 2
7   const y:integer = 10
8
```

Motivação

Construir uma linguagem fácil de aprendizado e com sintaxe que seja intuitiva aos olhos humanos. A ideia principal é ter clareza do que é cada coisa ao olhar o código

Explorar elementos de outras linguagens como Javascript, principalmente, mas com a exigência de tipagens.

Características da Linguagem

Simplicidade

- Sintaxe intuitive (pelo menos bem mais que Java).
- Uso de palavras-chave fáceis de entender.
- Estrutura clara de blocos de Código.
- Declarações simples de tipos (variáveis e retornos de funções).

Extensibilidade

- Permite que os programadores definam suas próprias funções e variáveis.
- Definir funções em formatos diferentes.

Estrutura da linguagem

- **Variáveis**: Uma variável pode conter um valor numérico ou uma sequência de caracteres. Podem ser definidas pelo programador ou automaticamente.
- **Condicionais**: Permite que o programa faça diferentes ações dependendo do valor de uma expressão.
- **Loops**: Permite que o programa execute o mesmo bloco de código várias vezes, de acordo com uma condição.
- **Funções**: Permite que o programa organize seu código em pequenas unidades reutilizáveis, que aceitam parâmetros e retornam valores.

Exemplos de código

```
def factorial(integer a) : integer {
  const factor : integer = 1
  const index : integer = 1

  while (index ⩽ a) {
    factor = factor * index
    index = index + 1
  }

  return factor
}

const x : integer = factorial(5)

stdout("Fatorial de", 5, "é:", x)
// Fatorial de 5 é: 120
```

- Funções e operações matemáticas simples.
- Blocos “while” e “function” bem delimitados por chaves.
- Print de múltiplos valores

Exemplos de código

- Aceita strings
- Operações entre strings

  (comparações, concatenação)

```
1  const x : integer
2  const y : integer
3
4  const a : string = "James"
5  const b : string = " Bond"
6
7  x = a == b
8  y = a =/= b
9
10 // Concatena, compara se igual, compara se diferente
11 stdout(a . b, x, y)
12 // James Bond 0 1
```

Exemplos de código

```
1  // Retorna o máximo de 2 elementos
2  def max(integer m, integer n) : integer {
3    const max : integer
4
5    if (m ⩾ n) {
6      max = m
7    } else {
8      max = n
9    }
10
11   return max
12 }
13
14 // Retorna o mínimo de 2 elementos
15 const min = (integer m, integer n) : integer ⇒ {
16   const min : integer
17
18   if (m ⩽ n) {
19     min = m
20   } else {
21     min = n
22   }
23
24   return min
25 }
26
27 const x : integer = 10
28 const y : integer = 20
29 const z : integer = 30
30
31 stdout("Máximo entre y e z:", max(y, z))
32 stdout("Mínimo entre x e y:", min(x, y))
```

- Funções podem ser criadas com “def” com estrutura de “arrow functions” como em Javascript.
- If/Else bem delimitados por chaves