Portuscript

Lógica da Computação - Cicero Tiago

1 Motivação

```
funcao abs_diff(a, b) {
  se (a > b) {
    retorne a - b;
  } senao {
    retorne b - a;
  }
}

imprima(abs_diff(10, 5));
```

A motivação da linguagem criada pode ser definida como a necessidade de facilitar o aprendizado e o ensino de lógica de programação e algoritmos sem necessariamente precisar aprender uma linguagem “comercial” logo de início.

Apesar de ser em português, a linguagem é baseada em javascript.

2 Características e Curiosidades

```
funcao abs_diff(a, b) {
  se (a > b) {
    retorne a - b;
  } senao {
    retorne b - a;
  }
}

imprima(abs_diff(10, 5));
```

Portuscript possui as operações fundamentais de javascript:

- Operações Aritméticas
- Operações Lógicas
- Input (próximos passos)
- Output
- Comparações
- Loops
- Condicionais
- Funções

2.1 Operações Lógicas

Suporta operações lógicas fundamentais

- *and*
- *or*
- *not (próximos passos)*

```
seja a = verdadeiro;
seja b = falso;

imprima(a && b);
imprima(a || b);
```

2.2 Operações Aritméticas

```
funcao fib(n) {
  se ((n == 0) || (n == 1)) {
    retorne 1;
  }
  senao {
    retorne fib(n - 1) + fib(n - 2);
  }
}

imprima(fib(30));
```

Operações aritméticas fundamentais:

- Soma
- Subtração
- Divisão
- Multiplicação

2.3 Input/Output

Output pode ser feito por meio da função built-in de saída de dados. Futuramente, deve ser implementada a entrada.

- imprima
- leia (próximos passos)

```
funcao fib(n) {
  se ((n == 0) || (n == 1)) {
    retorne 1;
  }
  senao {
    retorne fib(n - 1) + fib(n - 2);
  }
}

imprima(fib(30));
```

2.4 Comparações

```
constante a = 5;
constante b = 10;

imprima(a > b);
imprima(a == b);
imprima(a < b);
```

Comparações fundamentais:

- Maior;
- Maior ou igual; (próximo passos)
- Menor;
- Menor ou igual; (próximo passos)
- Igual;
- Diferente; (próximo passos)

2.5 Loops

Aceita o loop ‘enquanto’.

O exemplo ao lado gera os seguintes valores no console:

```
x: 1
x: 2
x: 3
x: 4
x: 5
x: 6
x: 7
x: 8
x: 9
```

```
constante limite = 10;
seja x = 1;

enquanto (x < limite) {
  imprima("x:", x);
  x = x + 1;
}
```

2.6 Condicionais

Aceita a execução condicional conforme o exemplo ao lado.

```
constante limite = 10;
seja x = 1;

enquanto (x < limite) {
  se (limite - x < ) {
    imprima("está perto: ", limite - x);
  }
  imprima("x:", x);
  x = x + 1;
}
```

2.7 Funções

```
funcao fib(n) {
  se ((n == 0) || (n == 1)) {
    retorne 1;
  }

  retorne fib(n - 1) + fib(n - 2);
}

constante limite = 10;
seja x = 1;

enquanto (x < limite) {
  imprima(fib(x));
  x = x + 1;
}
```

A linguagem também permite a declaração e a chamada de funções, inclusive com recursividade.

Próximos passos

Para os próximos passos, existem alguns pontos que já estão previstos para serem desenvolvidos:

- escrever mais testes para os nodes automáticos;
- implementar o tipo undefined;
- escopo de bloco (não só em funções);
- ignorar comentários;
- operador lógico “not”;
- input do usuário;
- documentação;