FACULDADE ÚNICA DE IPATINGA

LUANA MAIARA VIEIRA GOMES
LUCAS PAULO DE OLIVEIRA PINTO
MATHEUS PIRES SILVA
RICARDO ALVES PAULA

COMPILADORES: LINGUAGEM DOOW PARTE 01

Ipatinga - MG
2020

LUANA MAIARA VIEIRA GOMES
LUCAS PAULO DE OLIVEIRA PINTO
MATHEUS PIRES SILVA
RICARDO ALVES PAULA

COMPILADORES: LINGUAGEM DOOW PARTE 01

Parte 01 do trabalho da construção de uma nova linguagem de programação, apresentado à Disciplina de Compiladores da Faculdade de Única de Ipatinga.

Responsável da Disciplina: Prof. Filipe Fernandes

Ipatinga - MG
2020

Linguagem DOOW

A linguagem **DOOW** foi baseada na linguagem C em conjunto com JavaScript. Essa nova linguagem possui as seguintes características:

- Permitir Declaração de Bibliotecas;
- Permitir Declaração de Variáveis;
- Permitir Declaração de Constantes;
- Permitir Declaração de Funções completas (permitindo parâmetros e permitindo retorno de valores, podendo não os ter);
- Definição dos Operadores de Atribuição, Lógico, Relacionais e Aritméticos;
- Definição de expressões das mais variadas;
- Atribuição;
- Entrada de Dados via teclado, somente;
- Saída de Dados para o monitor, somente;
- Comandos de Seleção (Condicional);
- Comandos de Repetição;

Palavras reservadas da linguagem DOOW

- ifTern
- inArray
- indexOf
- converteVirgulaPonto
- isNullEmptyUndefined
- convertePontoVirgula
- try...catch
- addLibs
- replace
- async
- await
- promise
- console.log
- then
- forEach
- map
- length
- reduce

- return
- default
- enum
- for
- if
- break
- case
- const
- while
- continue
- switch
- void
- else

Manual de utilização, instruções e exemplos da Linguagem DOOW

- **ifTern:** É uma função que realiza um If ternário sem a utilização de “?” e “:” como nas outras linguagens.

```
ifTern(a,b,c){
  let result;
  a ? result = b : result = c;
  return result;
}
```

- **inArray:** Essa função verifica se possui um valor dentro de determinado array, onde retorna false caso não encontrar e true caso encontre.

```
inArray(lista, item): boolean {
  return lista.indexOf(item) !== -1 ? true : false;
}
```

- **indexOf:** É retornada a posição da primeira ocorrência do elemento na string. Caso o elemento não seja encontrado, é retornada a posição -1.

```
if (num.indexOf('.') !== -1) {
  return num.replace('.', ',');
}
```

- **converteVirgulaPonto**: converte as vírgulas de um valor em ponto

```
converteVirgulaPonto(n) {
  const num = n + '';
  if (num.indexOf(',') !== -1) {
    return num.replace(',', '.');
  }
  return num;
}
```

- **isNullEmptyUndefined:** Essa função verifica se determinado valor é vazio, nulo ou indefinido.

```
isNullUndefinedEmpty(e: any) {
  if (e === null || e === undefined || e === '') {
    return true;
  }
  return false;
}
```

- **convertePontoVirgula**: converte os pontos de um valor em virgula

```
convertePontoVirgula(n) {
  if (!n) return '';
  const num = n + '';
  if (num.indexOf('.') !== -1) {
    return num.replace('.', ',');
  }
  return num;
}
```

- **try...catch:** Executa determinado bloco e caso der erro chama uma exceção que ficará declarada dentro do catch. O catch é caso der algo errado na execução do try.

```
try {
    // bloco de código
} catch {
    //bloco de código (tratamento do erro)
}
```

- **addLibs**: importa bibliotecas (semelhante ao using do C# e include do C++)

```
addLibs{ BibliotecaXYZ } from '../libs/BibliotecaXYZ';
```

- **replace:** recebe dois parâmetros e substitui um pelo outro.

```
num.replace('.', ',');
```

- **async:** ao ser declarada transforma em uma função ela se transforma em uma funcao assincrona e permite a utilização da palavra await.

```
const nomeDaFuncao = async () => {
    //bloco de código
    const variavel = await promessa; // espera o retorno da promisse
}
```

- **await:** é utilizado para esperar por uma Promise. Ele pode ser usado apenas dentro de uma async function.

```
const nomeDaFuncao = async () => {
    //bloco de código
    const variavel = await promessa; // espera o retorno da promisse
}
```

- **promise:** é um objeto usado para processamento assíncrono. Um Promise (de "promessa") representa um valor que pode estar disponível agora, no futuro ou nunca.

```
new Promise((resolve, reject) => {
    console.log('inicio');

    resolve();
})
.then(() => {
    throw new Error('caso algo falhe');

    console.log('faça isso');
})
.catch(() => {
    console.log('se tem falha faça isso');
})
```

- **console.log**: imprime um texto no **console** (terminal ou console do navegador)

```
console.log("Teste");
```

- **then:** O método then() retorna uma Promise. Possui dois argumentos, ambos são "call back functions", sendo uma para o sucesso e outra para o fracasso da promessa.

```
new Promise((resolve, reject) => {
    console.log('inicio');

    resolve();
})
.then(() => {
    throw new Error('caso algo falhe');

    console.log('faça isso');
})
.catch(() => {
    console.log('se tem falha faça isso');
})
```

- **forEach:** executa o callback fornecido uma vez para cada elemento da ordem com um valor atribuido. Ele não é invocado para propriedades de índices que foram deletados ou que não foram inicializados (por ex. em arrays esparsos).

```
const array1 = ['a', 'b', 'c'];

array1.forEach(element => console.log(element));

// expected output: "a"
// expected output: "b"
// expected output: "c"
```

- **map:** O método map() invoca a função callback passada por argumento para cada elemento do Array e devolve um novo Array como resultado.

```
var numbers = [1, 4, 9];
var doubles = numbers.map(function (num) {
  return num * 2;
});
// doubles é agora [2, 8, 18]. numbers ainda é [1, 4, 9]
```

- **length:** Comando usado para retornar o tamanho de uma variável.

```
if(a.length == 2){
  return true;
}
```

- **reduce**: O método reduce() executa uma função reducer (provida por você) para cada membro do array, resultando num único valor de retorno.

```
const array1 = [1, 2, 3, 4];
const reducer = (accumulator, currentValue) => accumulator + currentValue;

// 1 + 2 + 3 + 4
console.log(array1.reduce(reducer));
// saída: 10
```

- **return:** Retorna um valor dentro de uma função ou força o abandono da mesma.

```
if(a){
  return true;
}
```

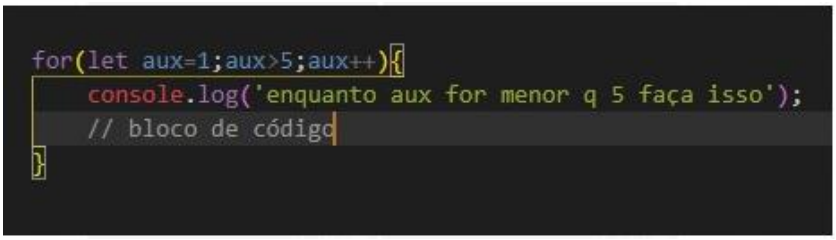
- **default:** É utilizado dentro de switch…case para tratar valores não definidos anteriormente nas opções case.

```
switch(a>2) {
    case a==1:
        console.log('faça isso');
        break;
    case a==2:
        console.log('faça isso');
        break;
    default:
        console.log('caso não seja igual a nenhuma das opções anteriores faça isso');
}
```

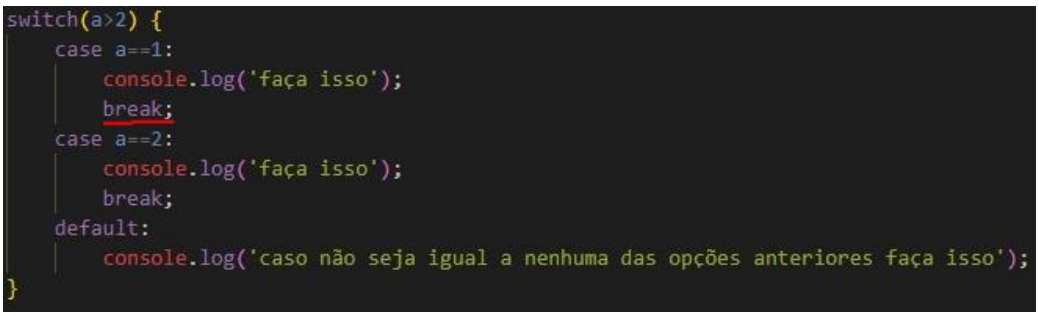
- **enum:** Tipo de dados definido pelo programador que permite a definição de constantes

```
enum Direcao {
  Cima,
  Baixo,
  Esquerda,
  Direita
}
```

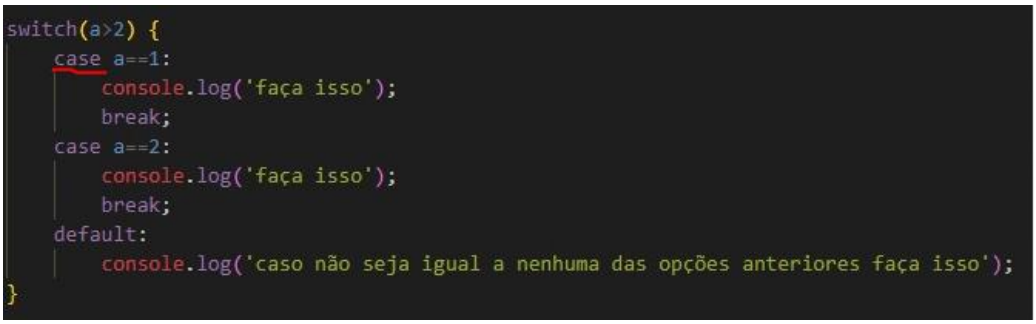
- **for:** Estrutura de repetição que utiliza condições e contador.



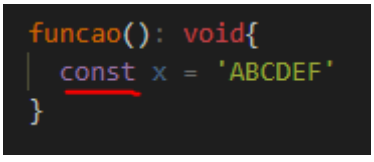
- **break:** comando para forçar a saída imediata dos comandos switch, for, while, e , do…while.



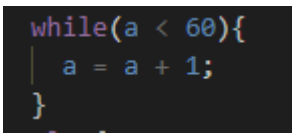
- **case:** Utilizado dentro do comando switch para selecionar uma constante.



- **const:** Impede que uma variável seja modificada, criando assim uma constante.



- **while:** Estrutura de repetição que executa enquanto uma condição é verdadeira.



- **continue:** Força a interrupção dos loops for , while , ou do…while fazendo com que passem para a próxima iteração.

```
funcao(a): void{
  for (let i = 0; i < 10; i++) {
    if (i === 3) {
      continue;
    }
    a = a + i;
  }

  console.log(a);
  // expected output: "012456789"
}
```

- **switch:** Comando de seleção usando em conjunto com o comando case, permite escolher entre várias opções.

```
switch(a>2) {
    case a==1:
        console.log('faça isso');
        break;
    case a==2:
        console.log('faça isso');
        break;
    default:
        console.log('caso não seja igual a nenhuma das opções anteriores faça isso');
}
```

- **void:** Comando que indica que a função não retorna nada ou que não tem parâmetros de entrada.

```
funcao(): void{
  //corpo da função
}
```

- **if:** Comando condicional que altera o fluxo do programa de acordo com uma condição que pode ser verdadeira ou falsa.

```
if(a){
  return true;
}
```

- **else:** Indica um bloco de comandos a ser executado quando a condição do comando if for falsa.

```
if(a.length == 2){
  return true;
}
else{
  return false;
}
```

Regras para construção

- **Declarações de bibliotecas**: Declarações de bibliotecas só são permitidas no topo do código.
  Exemplo: **$using biblioteca.math;**

- **Variáveis:** Não podem ser declaradas começando com número, nem com "_" e também não podem possuir o mesmo nome de nenhuma palavra reservada.
  Exemplo: **var result;**

- **Declaração de Função:** Toda função para ser declarada será a sequencia do nome da função seguido de um abre e fecha parênteses e abre fecha chaves. As funções podem ter parâmetros opcionais, parâmetros obrigatórios, ou não ter parâmetros.
  Exemplo 1: **função () { };** (Função sem parâmetros)
  Exemplo 2: **função (a, b) { };** (Função com 2 parâmetros obrigatórios)
  Exemplo 3: **função (a?, b) { };** (Função com 1 parâmetro opcional e outro obrigatório, respectivamente)

- **Tipos de variáveis:** Todas as variáveis não possuem tipo, podendo armazenar qualquer valor (booleano, inteiro, float, etc).
  Exemplo: **var variavel**;

- **Atribuições:** Todas as atribuições serão realizadas utilizando o sinal de igual (=);
  Exemplo: **var x = 5;**

- **Comparação:** As comparações serão feitas utilizando 3 sinais de igual (===). A comparação do tipo OU, usará o operador "||" e a do tipo E usará o "&&".
  Exemplo: **if (x === 2 || (x > 5 && x < 10)) { }**

- **Operadores lógicos**: A multiplicação usará o operador = "*", a divisão a "/", a subtração o "-", a adição o "+" e o resto de uma divisão a "%".

Exemplo: **x = ((90 / 2) +(9 * 2)) % 2;**

- **Comandos de repetição:** Serão utilizados para laço de repetição as palavras reservadas “for”, “while”, “map” e “foreach”. (Exemplos no Manual do item acima).

- **Comentários de linhas:** Os comentários iniciam e terminam com o caracter “#”.
  Exemplo: **#comentarios#**

- **Constante:** As constantes deverão ser declaradas começando com o caracter “$” seguindo da palavra reservada “const” e depois o nome da constante.
  Exemplo: **$const tam;**

- **Final das linhas:** As linhas deverão ser terminadas com ponto e vírgula ao final de cada instrução.
  Exemplo: **var resultado = 4;**

- **Await:** A palavra reservada await poderá ser utilizada apenas dentro de um método **async**. (Exemplo no Manual do item acima).

REFERÊNCIAS

- BELTRAME, Refael; Introdução à linguagem C. **UFSM**. Disponível em: < http://coral.ufsm.br/beltrame/arquivos/disciplinas/graduacao_estrutura_de_dados/Estrutura-de-Dados_Aula-03_Introducao-Linguagem-C.pdf>. Acesso em: 24 abr. 2020;

- MOZILA, Developer; JavaScript. **MDN web docs**. Disponível em: < https://developer.mozilla.org/en-US/docs/Web/JavaScript>. Acesso em: 24 abr. 2020;

- FILHO, Cláudio; Identação, espaço em branco e ponto e vírgula em c++. **Excript**. Disponível em: < http://excript.com/cpp/indentacao-espaco-em-branco-ponto-e-virgula-cpp.html>. Acesso em: 25 abr. 2020;

- BAJAJ, Twinkl; Node.js | forEach() function. **GeeksforGeeks**. Disponível em: < https://www.geeksforgeeks.org/node-js-foreach-function/>. Acesso em: 26 abr. 2020;