Lógica de programação

É a área da programação em que estudamos as estrututras básicas que compõe qualquer código de programação. Para escrever esses códigos, utilizaremos algoritmos. Algoritmos são uma sequencia de passos finitos que quando bem organizados e estruturados resolvem a solução de um problema.

Algoritmo atravessarRua

Olhar para direita

Olha para a esquerda

Se estiver vindo carro, não atravessar

Se não estiver vindo carro, atravessar

Fimalgoritmo

Um Algoritmo pode ser escrito de diversas maneiras diferentes:

Algoritmo atravessarRua

Olhar para esquerda

Olha para a direta

Se não estiver vindo carro, atravessar

Se estiver vindo carro, não atravessar

Fimalgoritmo

Variaveis

Variaveis são espaços reservados na memoria do computador criados para se guardar determinado tipo de dado. Na programação esses dados são diferenciados pelos seus TIPO PRIMITIVOS.

Inteiros – Vai armazenar um valor inteiro: ex: ,5, 10 , 520, 985

Real(float) – Vai armazenar um valor decimal. 1.5 , 20.89 , 15.87, 4.8 **obs: Não se deve separar as casas decimais por virgula, apenas ponto.**

Caractere – Vai armazenar letras, palavras, textos (String). Ex: “Victor”, “F”, “Logica de programação é legal”. Obs: Todo texto na programação deve ser colocado entre aspas duplas, não duas aspas simples.

Lógico – Vai armazenar variáveis do tipo lógico(boolenos). Verdadeiro ou falso

Criação de Variáveis

Regras na hora de criar variáveis. O Javascript é uma linguagem Sensitive case, ou seja, diferencia letras minúsculas de maiúsculas.

1. Uma variável SEMPRE deve começar com uma letra, nunca com números. A partir do segundo caractere pode colocar números. Ex: 1numero, numero1
2. Não pode utilizar caracteres especiais ou acentuação, exceto o underline \_: Ex: ação, acao
3. Não pode conter espaços em branco. Caso queira criar variáveis com duas palavras, utilize ou \_ ou o padrão camelCase. Ex: primeiroNome.
4. Não pode ser uma palavra reservada. Uma palavra que faça parte da sintaxe da linguagem.

Operadores Aritmeticos

A =5

B =2

|  |  |  |  |
| --- | --- | --- | --- |
| Adicao | + | A + B | 7 |
| Subtracao | - | A - B | 3 |
| Multiplicacao | *\** | A \* B | 10 |
| Divisao | / | A / B | 2.5 |
| Exponenciaçao | \*\* | A ^ B | 25 |
| Módulo(Resto) | % | A % B | 1 |

Ordem de Precedencia

|  |  |
| --- | --- |
| Parentesis | ( ) |
| Exponenciacao | \*\* |
| Multiplicacao/Divisao | \*/ |
| Adicao/Subtracao | +- |

Estrutura condicional If..else(Se...senao)

A estrutura condicional diz respeito a ter uma ação especifica para cada situaçao que possa ocorrer. Sintaxe:

Estrutura condicional simples

If (condição for verdadeira){

// Execute esse bloco de codigo

}else{

//Execute este outro bloco de codigo

}

Estrutura condicional composta

If(situação for verdadeira){

//Execute este bloco

} else if(situação for verdadeira){

//Execute este bloco

}else{

//Execute este bloco

}

Operadores Relacionais

Os operadores relacionais serão utilizados para criar as condições, pois um operador relacional SEMPRE ira retornar um valor lógico.

|  |  |  |
| --- | --- | --- |
| MAIOR QUE | > | 5 > 2 |
| MENOR QUE | < | 5 < 2 |
| MAIOR OU IGUAL | >= | 5 >= 2 |
| MENOR OU IGUAL | <= | 5 <= 2 |
| IGUAL | = | 5 = 2 |
| DIFERENTE | != | 5 != 2 |

Operadores Lógicos

Existem situações em que precisamos de testar duas ou mais condições, para fazer essa junção utilizamos os operadores lógicos. São eles: E, Ou e Não.

Operador E

|  |  |  |
| --- | --- | --- |
| **P** | **Q** | **P && Q** |
| V | V | V |
| V | F | F |
| F | V | F |
| F | F | F |

Operador OU

|  |  |  |
| --- | --- | --- |
| **P** | **Q** | **P || Q** |
| V | V | V |
| V | F | V |
| F | V | V |
| F | F | F |

|  |  |
| --- | --- |
| ! V | Falso |
| ! F | Verdadeiro |

Estrutura condicional Switch..case(Escolha..caso)

A estrutura switch..case vai ter uma variável como parâmetro e dependendo do resultado dessa variável, ele vai executar uma ação especifica.

Sintaxe:

Switch(variavel){

Case <Resultado seja esse valor>

//Execute este bloco de código

break

Case <Resultado seja esse valor>

//Execute este bloco de código

break

Default:

//Execute este bloco de código

}

Laços de repetição

Também conhecido como loops, os laços de repetição são estrutura utilizadas em momentos em que se deseja repetir uma ação determinado numero de vezes, podendo ou não ter um limite para a quantidade de repetições. Por exemplo, pode ser solicitado que se escreva o nome de 4 pessoas, então existe um limite. Mas também poderão haver situações que não teremos limite, como por exemplo se eu falar que enquanto a casa estiver bagunçada, não sairei, visto que so depende desta pessoa para esse loop acabar. Ela pode fica 30minutos, 1hora, 5horas e etc. Perceba que não tem um limite. Para que se estipule esse limite, utilizamos uma variável contadora.

Laço de repetição While(enquanto)

Enquanto uma situação for verdadeira, ele vai executar um bloco de código.

Sintaxe:

While(condição for verdadeira){

//Execute este bloco de codigo

}

Variável contadora

Uma variável que vai criar uma contagem para estabelecer um limite para o laço de repetição. Geralmente colocamos o nome dessa variável contadora com a letra c.

Var c = 1; // criando a variável c recebendo o numero 1

|  |  |  |
| --- | --- | --- |
| Lógica | Javascript | Resultado |
| C = c + 1 | C++ | 2 |
| C = c - 1 | c-- | 0 |
| C = c + 5 | C +=5 | 6 |
| C = c \* 2 | C\*=2 | 2 |
| C = c / 2 | c/=1 | 1 |

Estrutura de repetição do..while(Faça..enquanto)

Faça um bloco de código enquanto uma condição for verdadeira.

Sintaxe:

Do{

//Execute este bloco

}while(enquanto uma situação for verdadeira)

Qual a diferença entre as estruturas while e do..while

A diferença é que na estrutura while o teste lógico é feito no inicio e na estrutura do..while o teste logico é feito no final, sendo assim, na segunda estrutura o código sera executado pelo menos uma vez antes de verificar a condição, na while ele só vai executar o código se a condição já for verdadeira.

Estrutura de repetição for(para)

Para o meu contador começando de algum valor, ele vai acontecer enquanto estiver atendendo a uma condição e vai andar sobre um passo definido.

Sintaxe:

For(let c = 1; c <= 10; c++){

Console.log(c)

}

Arrays

Um array é uma lista de variáveis que são refererenciadas por um nome e posições. Primeiramente para entedermos o que são arrays, temos que entender o que são as variáveis comuns.

Var numero = 5

Var nome = “Miguel”

Var numero = [1,5,8,9]

Var nome = [“Victor”, “Amanda”, “Miguel”]

Var nome = [1, “Victor”, true]

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
|  |  |  |  | Nome = “Miguel” |
|  |  | Numero = 5 |  |  |
|  |  |  |  |  |
| Numero[0] = 1 | Numero[1]= 5 | Numero[2]= 8 | Numero[3]= 9 |  |
| Nome[0] = “Victor” | Nome[1] = “Amanda” | Nome[2] = “Miguel” |  |  |

Funções

Uma função é utilizada quando se deseja atribuir uma tarefa alguma coisa. As funções na programação são utilizadas em situações que faz-se necessário a repetição. Uma função pode ou não ter parâmetros. Uma função pode ou não ter um retorno.

Exemplo de função sem parâmetro:

Function fazerArroz(){

Pegue 300g de arroz, 400ml de agua e 4g de sal

Coloque tudo numa panela e deixe ferver a fogo alto

Depois que a agua secar, acrescente mais agua ate que cubra o arroz

Depois que secar o arroz estará pronto

}

fazerArroz()

fazerArroz()

fazerArroz()

Exemplo de função com parâmetro

Function irAoMercado(dinheiro, produto){

Vai ate o mercado

Vai na sessão do produto que quer

Pegue o produto e leve ao caixa

Pague esse produto com o dinheiro

Volte para casa

}

irAoMercado(20 reais, arroz)

irAoMercado(17 reais, caixa de ovos)

irAoMercado(5 reais, biscoito)

Outro exemplo de função com parâmetro:

Function fazerArroz(arroz,agua,sal){

Pegue o arroz, a agua e o de sal

Coloque tudo numa panela e deixe ferver a fogo alto

Depois que a agua secar, acrescente mais agua ate que cubra o arroz

Depois que secar o arroz estará pronto

}

fazerArroz(500g, 600ml, 5g)

fazerArroz(300g, 500ml, 4g)

Funçao com retorno:

Function soma(numero1, Numero2){

Return numero1 + numero2

}

Soma(10,10)

Soma(5,4)

Soma(10,30)