**Lógica de programação**

É disciplina que ensina como um computador entende um código. A esses códigos damos o nome de algoritmos. Um algoritmo é uma sequencia de passos que qunado bem organizados e estruturados resolvem um problema.

Exemplo

algoritmoAtravessarRua

Olha para a direita

Olhar para a esquerda

Se estiver vindo carro, não atravessar

Se não estiver vindo carro, atravessar.

Um algoritmo pode ser escrito de diversas formas, ou seja, um mesmo problema pode ter varias soluções diferentes.

**Variaveis**

Variaves são espaços resevados na memoria do computador utilizados para armazenar dados. A esses dados damos o nome de TIPO PRIMITIVO. Existem 4 tipos principais:

Inteiro – Armazena números do tipo inteiro. Ex: 5 , 4 ,10 ,150 ,10000

Real – Armazena um numero decimal (float). Ex: 4.5 , 10.58, 150.987. obs: SEMPRE SEPARAR NUMEROS DECIMAIS POR UM PONTO, JAMAIS POR VIRGULA.

Lógico – É um valor booleano. Verdadeiro ou falso.

Caracteres – Vai armazenar dados em texto, palavra, frase (String). Ex: “Victor”, “A vida é bela”

**Regras para criar variáveis**

1 – Uma variável NUNCA deve começar com um numero, SEMPRE com uma letra. A partir do segundo caractere pode ser colocado um numero. Ex: n1, n2

2 – Uma variável não pode ter caracteres especiais. EX: dinheiro$. Exceto underline(\_). Porém underline não é utilizado, então caso preciso de variáveis com nomes compostos utilize o padrão camelCase. Bascicamente a primeira letra da primeira palavra em minúsculo e a primeira letras das palavas subsequentes em maiúsculo. Ex: primeiroNome

3 – Variaveis não podem ter espaços. Ex: primeiro nome

4 – Variaveis não podem ser uma palavra reservada, ou seja, uma palavra que faça parte da sintaxe da linguagem.

**Operadores Aritméticos**

A<- 5

B<- 3

|  |  |  |
| --- | --- | --- |
| Soma | + | A + B = 8 |
| Subtração | - | A – B = 2 |
| Divisao | / | A / B = 1.6 |
| Multiplicacao | \* | A \* B = 15 |
| Exponenciação | ^ | A ^ B = 125 |
| Módulo(Resto) | % | A % B= 2 |

Ordem de precedência

|  |  |
| --- | --- |
| Parentesis | () |
| Exponenciação | ^ |
| Multiplicação/divisao | \*/ |
| Soma/subtraçao | +- |

**Estruturas condicionais**

Estruturas condicionais são utilizadas em momentos em que uma situação pode erar resultados diferentes. Um exemplo disso é o exemplo do inicio do documento onde você so vai atravessa a rua se não estiver vindo carro, caso contrario você nao vai atravessar. Perceba que tem uma situação para cada codição diferente. Na programação temos duas estruturas condicionais. If..else(Se..entao) e a switch..case(escolha...caso).

**Estrutura condicional If..else(Se..entao)**

Se uma situação for verdadeira, execute um bloco de código. Caso não seja verdadeira execute outro bloco de código.

SE(condição for verdadeira) então

//Execute este bloco de código

Senão

//Execute este bloco de código

Fimse

Para criar condições utilizaremos dos OPERADORES RELACIONAIS. Estes operadores SEMPRE irão retornar um valor lógico.

|  |  |  |
| --- | --- | --- |
| Maior | > | A > B |
| Menor | < | A < B |
| Maior ou igual | >= | A >= B |
| Menor ou igual | <= | A <= B |
| Igual | = | A = B |
| Diferente | <> | A <> B |

**Operadores Lógicos**

Os operadores lógicos serão utilizados em situação em que temos mais de uma condição logica. São eles: E, OU e NÃO.

**Operador E:**

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

**Operador OU:**

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

**Operador Não:**

|  |  |
| --- | --- |
| Não V | **F** |
| Não F | **V** |

**Estrutura condicional escolha caso**

Escolha uma variável e dependendo do valor dela, vamos ter um caso especifico para cada valor. A diferença entre as duas estruturas condicionais é que esta não pode ser utilizada para casos em que haja um valor lógico. Não pode utilizar esta estrutura caso queira verificar a condição de um aluno se esta aprovado ou nao

Sintaxe:

Escolha(variavel)

Caso <alguma condição>:

//execute este bloco de código

Caso <outra condição>:

//execute este bloco de código

Caso <outra condição>:

//execute este bloco de código

Outrocaso

// exceute este bloco de código

Fimescolha

**Estruturas de repetição**

A estruturas de repetição são utilizadas em momentos que queremos repetir uma ação podendo ou não ter limite.

**Estrutura de repetição enquanto**

Enquanto uma situação for verdadeira, execute um bloco de código. A estrutura enquanto pode ou não ter um limite. Um exemplo de limite seria você querer pegar 4 notas de um aluno, esse laço vai ser repetir 4 vezes.

Exemplo sem limite de repetição.

Enquanto(não arrumar o quarto) faca

//ficar de castigo

Fimenquanto

Saiu do laço

Para estipular o limite do laço de repetição, utiliza-se uma VARIÁVEL CONTADORA.

C<- 1

c = c + 1 // c<- 2

c = c + 1 // c<- 3

**Estrutura de repetição repita..ate**

Repita um bloco de código, ate que uma situação seja verdadeira. A diferença entre as duas estruturas é que nesta o teste lógico é feito no final, e na enquanto é feito no incio. Esta estrutura também pode ou não ter um limite e esse limite é imposto pela variável contadora.

Repita

//bloco de código

Ate(situação seja verdadeira)

Exemplo

Faca

ficarDeCastigo()

ate(arrumar o quarto)

**Estrutura de repetição Para**

Estrutura Para é a mais utilizada, pois na própria sintaxe já tem tudo que é necessário para rodar o loop. A diferença dela para as outras é que ela so deve ser utilizada quando existe um limite. Caso tenha uma situação que não tenha limite, aí pdeve ser utilizada uma das outras duas.

Sintaxe:

Para c de <inicio> ate <final> passo <passo> faca

//loop vai repetir 10x

fimpara

Arrays(listas)

Arrays são uma sequencia de variaves que são referenciadas por um nome e uma posição. Quando criamos uma variável simples, na memoria do computador ele reserva um espaço aleatório para guardar esse valor. Qyuando criamos um array, ele cria uma lista de variáveis que terá uma posição e ela terá um tamanho pre estabelecido. O array SEMPRE ira começar da posição 0. Por motivos de estudo, começaremos da posição 1.

Nome: caractere

Numero: inteiro

Numeros: vetor[1..5] de inteiro

Numero<- 5

Nome<- “Victor”

Numeros[1]<- 10

Numeros[2]<- 100

Numeros[3]<- 5

Numeros[4]<- 90

Numeros[5]<- 6

Para i de 1 ate 5 passo 1

Escreva(“Numero: ”)

Leia(números[i])

fimpara

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
|  |  |  | Nome: “Victor” |  |
|  | Numero: 5 |  |  |  |
|  |  |  |  |  |
| Numeros[1]:10 | Numeros[2]:100 | Numeros[3]:5 | Numeros[4]:90 | Numeros[5]:6 |