

## Comparação

Estrutura Para	Condição Não tem $v \leq v_f$	Quantidade de Execuções $((v_f - v_i) \text{ div } p) + 1$	Condição de Existência
Enquanto Repita	Início Fim falsa	0 ou muitas condição mínimo 1 condição	verdadeira

```
cont:=cont+1;
acum:=acum+valor;
```

```
acum::=1 //para acumular multiplicação
acum:=0 // para acumular soma
```

## Algoritmo Exemplo1;

### Const

TAM=50;

### Var

a:inteiro  
b:real;  
c:logico;  
d:caracter;  
e:string;

### Início

```
//comentario
imprimir("ola");
ler(a);
```

### Fim.

### INICIO e FIM;

quando tiver mais de um comando.

#### // estrutura de decisao

```
se <condicao>entao <comando>; fimse;
Se <condicao> entao <comando>;
Senao <comando>; FimSe;
```

#### //usada para executar um comando entre e varios do tipo INTEIRO OU CARACTER.

```
Escolha (<expressao>)
<opcao_1>: <comando>;
<opcao_2>: <comando>;
<opcao_N>: <comando>;
Senao
<comandos>;
FimEscolha;
```

#### //repeticao com variavel de controle

```
Para <var> de <vi> ate <vf> passo <p> faca
<comando1>;
<comando2>;
<comandoN>;
```

FimPara; PASSO <P> // OPCIONAL

#### repticao com controle de fluxo no inicio

```
Enquanto <condicao> faca
<comando1>;
<comando2>;
FimEnquanto;
```

#### //repticao com controle de fluxo no final

```
Repita
<Comando1>;
<Comando2>;
até <condicao>;
```

```
//
if (<expressao logica>){ <comando>; }
else { <comando>; };
```

```
//
switch (<variavel>)
{
case 1:<bloco de instruções>;break;
case '+':case '-':<bloco de instruções>;break;
default:<bloco de instruções>
}
```

```
for (VC=0;VC<50;VC++) // VC= Variavel contadora
{ <INSTRUCAO>; }
while ( VC < 50) // VC= Variavel contadora
{ <INSTRUCAO>; }
```

```
do {
}while ( VC < 50); // VC= Variavel contadora
```

POW(BASE, EXPOENTE) // potencia em C  
SQRT(Numero) // raiz em C  
cont++; //contador em C

<blocos de instrução> = conjunto de <comandos>  
{=INICIO  
}= FIM; condicao= expressao logica??

Par= x mod y = 0  
Impar= x mod y <> 0

```
Se (cont = 1) Então
Início
maiorAltura := altura;
menorAltura := altura;
Fim;
Senão
Início
Se (altura > maiorAltura) Então
maiorAltura := altura;
FimSe;
Se (altura < menorAltura) Então
menorAltura := altura;
FimSe;
Fim;
FimSe;
```

dividendo |divisor  
resto |quociente

### divisao INTEIRO E REAL :

- não é possível dividir por 0;  
- Resumindo, verifique se o valor do dividendo não é igual a zero e, se for dividir um inteiro por um real ou vice-versa, faça este resultado ser armazenado em uma variável real.

1) a divisão: 16 : 5 = 3 com resto 1  
Dividendo 16 : divisor 5 = quociente 3 + resto 1  
2) a multiplicação: 8 x 5 = 40  
Multiplicando 8 x multiplicador 5 = produto 40  
3) a soma: 8 + 9 = 17  
parcelas ou termos da adição: 8 e 9 ; soma 17  
4) a subtração: 15 - 10 = 5  
Minuendo 15 - subtraendo 10 = diferença 5

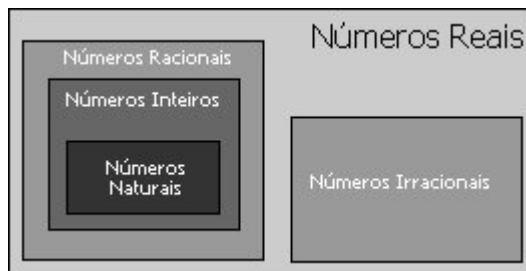
### Tipos primitivos:

Inteiro  
real  
logico(boleano v ou F)  
Caracter  
\*\*\*String

```
#include <stdio.h>
#include <stdlib.h>
main()
{
//%d %f %c %s
int a; float b; char c,d[50];
printf("frase");
scanf("%d",a);
system("pause");
}
```

**Operadores Lógicos**  
[]Três operadores básicos  
[]Negação (não) [ ! ]  
[]Conjunção (e) [ && ]  
[]Disjunção (ou) [ || ]

exp(base,exp) // potencia  
exp(base,(1/3))//raiz cubica  
Mod(resto) 10 mod 3 =1  
div(quociente) 10 div5=2



### Estrutura de um Algoritmo

1. Nome do algoritmo;
2. Declaração das constantes e variáveis globais;
3. Criação dos procedimentos e das funções;
4. Indicador do início do algoritmo;
5. Código principal;
6. Indicação do final do algoritmo.

O algoritmo pode ser representado de duas formas: textual e grafica

## Expressões Aritméticas

- Denominamos Expressão Aritmética
- Operadores são aritméticos
- Operandos
- Constantes numéricas (inteiro ou real)
- Variáveis do tipo numérico

==

## Operadores Aritméticos

- Conjunto de símbolos que representam as operações básicas da matemática:
- Adição (+):  $2+3$ ,  $X+Y$
- Subtração (-):  $4-2$ ,  $N-M$
- Multiplicação (\*):  $3*4$ ,  $A*B$
- Divisão (/):  $5/8$ ,  $C/P$
- Ainda temos a radiciação e a potenciação
- Potenciação [pot(x,y)]:  $\text{pot}(2,3)$
- Radiciação [rad(x)]:  $\text{rad}(9)$
- E, por último, resto e quociente da divisão inteira
- Resto da divisão (mod):  $9 \bmod 4$  é 1
- Quociente da divisão (div):  $9 \text{ div } 4$  é 2

==

## Prioridades

- Na resolução de expressões aritméticas, as operações guardam uma hierarquia entre si
- Prioridades Operadores
- 1a. parênteses mais internos
- 2a. pot rad
- 3a. \* / div mod
- 4a. + -

==

## Expressões Lógicas

- Denominamos expressão lógica
- Operadores são lógicos ou relacionais
- Operandos
- Relações lógicas
- Variáveis do tipo lógico
- Constantes lógicas

==

## Operadores Relacionais

- Utilizados para realizar comparações entre valores do mesmo tipo primitivo
- Igual a (=):  $3=3$ ,  $X=Y$
- Maior que (>):  $5>4$ ,  $X>Y$
- Menor que (<):  $3<6$ ,  $X<Y$
- Maior ou igual a (>=):  $5>=3$ ,  $X>=Y$
- Menor ou igual a (<=):  $3<=5$ ,  $X<=Y$
- Diferente de (<>):  $8<>9$ ,  $X<>Y$
- Resultado SEMPRE será um valor LÓGICO
- Exemplos:
- $2 * 4 = 24 / 3$   $8 = 8$  V
- $15 \bmod 4 < 19 \bmod 6$
- $3 * 5 \text{ div } 4 <= \text{pot}(3,2) / 0,5$
- $2 + 8 \bmod 7 >= 3 * 6 - 15$

## Potencia

- pot(base,expoente)
- exp(base,expoente)//visualg

## RAIZ

- rad(numero) //raiz quadrada
- exp(numero,(1/3)) // raiz cubica

## 1-ESTRUTURAS DE SELECAO:

- SE..ENTAO
- ESCOLHA

## 2- ESTRUTURAS DE REPETICAO:

- PARA (repeticao definida)
- ENQUANTO ( controle no inicio)  
executa 0 vez se for falso a condicao.
- REPITA(controle no fim)  
executa 1 vez mesmo que nao seja verdadeira a condicao

int(x/y) //div

mod (% em c)

==

## Operadores Lógicos

- Três operadores básicos
- Negação (não)
- Conjunção (e)
- Disjunção (ou)

==

## Tabelas Verdade

- Conjunto de todas as possibilidades combinatórias entre

- os valores de variáveis lógicas
- Operação de Negação (não A)
- Operação de Conjunção (A e B)
- Operação de Disjunção (A ou B)

==

## Prioridades

### Prioridades Operadores

- 1a. não
- 2a. e
- 3a. ou

### Prioridades Operadores

- 1a. parênteses mais internos
- 2a. operadores aritméticos
- 3a. operadores relacionais
- 4a. operadores lógicos

===

## Comando de Atribuição ( <- ou := )

- Permite fornecer um valor a uma variável
- O valor deve ser compatível com o tipo da variável
- Sintaxe

<identificador> := <expressão> ;

expressão

expressão aritmética

expressão lógica

- É obrigatória a existência de uma variável no lado esquerdo

Enquanto que no lado direito teremos o valor a ser atribuído a essa variável.

Esse valor poderá ser ob do de outra variável, de uma constante ou uma expressão que resulte um valor de um po compa vel com a variável do lado esquerdo da atribuição.

```
i:inteiro
d:logico // flag bandeira

d:=FALSO

leia(i)

se (i=2012) entao
d:=VERDADEIRO
fimse

se(d=VERDADEIRO)entao
escreva("PARABENS!")
senao
escreva("VOCE ERROU ")
fimse
```