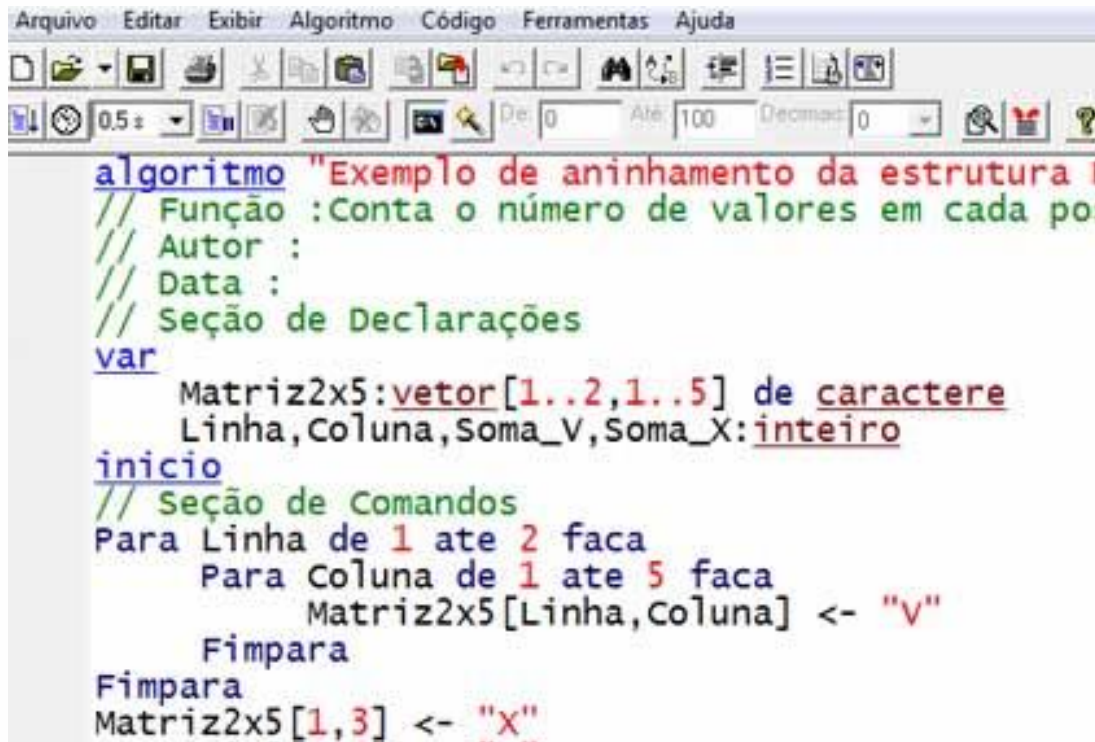


Lógica de programação



```
Arquivo Editar Exibir Algoritmo Código Ferramentas Ajuda
// Função :Conta o número de valores em cada po
// Autor :
// Data :
// Seção de Declarações
var
    Matriz2x5:vetor[1..2,1..5] de caractere
    Linha,Coluna,Soma_V,Soma_X:inteiro
início
// Seção de Comandos
Para Linha de 1 ate 2 faca
    Para Coluna de 1 ate 5 faca
        Matriz2x5[Linha,Coluna] <- "V"
    Fimpara
Fimpara
Matriz2x5[1,3] <- "X"
```

Lógica de programação

Algoritmo

Aqui você digita o nome do seu algoritmo.

```
{ algoritmo "NOME_DO_ALGORITMO"
```

Aqui você define os nomes das variáveis e seus tipos.

```
var
```

```
nome_da_variável : tipo_da_variável
```

Aqui você escreve as instruções, comandos e operações de seu algoritmo propriamente dito.

```
inicio
```

```
<instrução#1>
```

```
<instrução#2>
```

```
...
```

```
<instrução#n>
```

```
fimalgoritmo
```

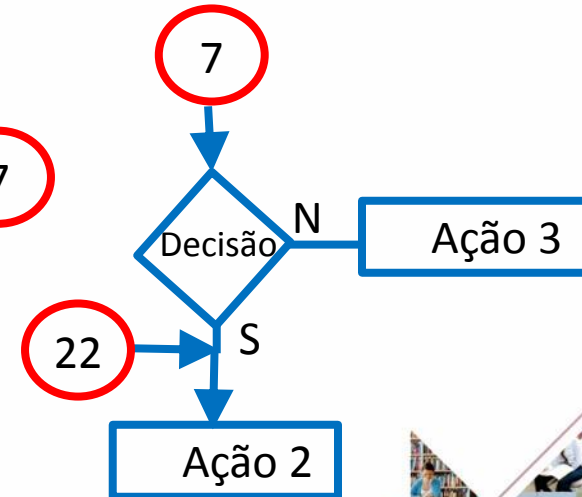
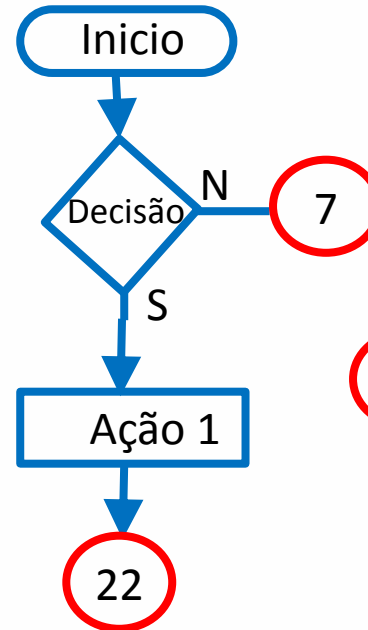
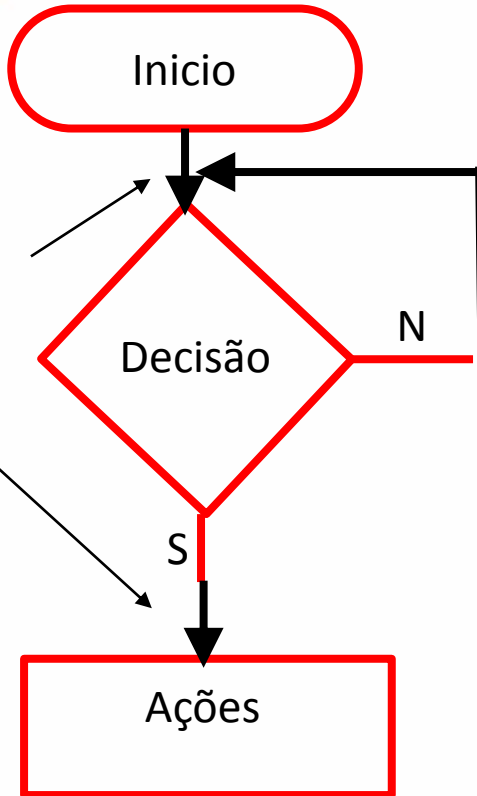


Fluxograma (reduzido)



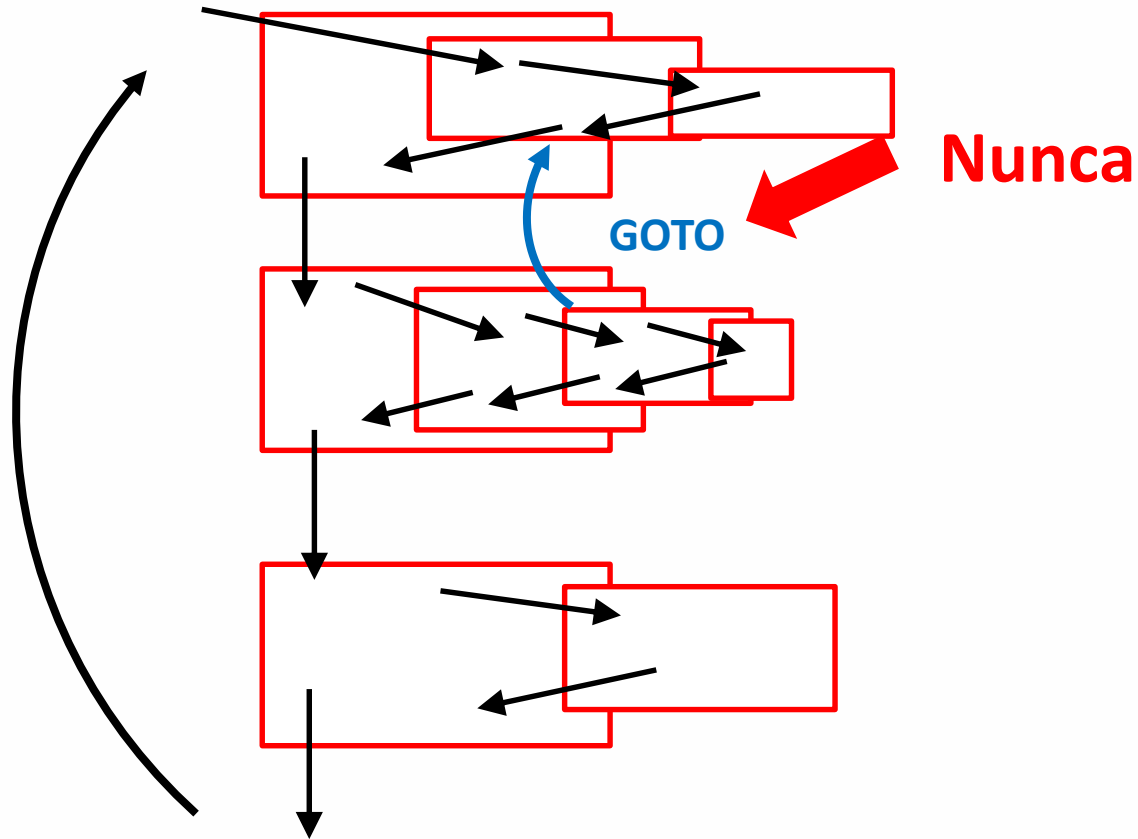
N Conector onde N é o
numero do conector

Indicadores
de Fluxo





Linguagens estruturadas





Visual G



- Tabela de tipos Visualg:

| Tipo | Descrição |
|----------------------|------------------------------------------------------------------------------------------------------------------------------------|
| Inteiro | Representa valores inteiros. Exemplos: 10, 5, -5, -10 |
| Real ou Numérico | Representa valores reais (com ponto separador da parte decimal). Exemplos: 10, 15.5, -14.67 |
| Literal ou Caractere | Representa texto (seqüência ou cadeia de caracteres) entre aspas duplas. Exemplo "Esta é uma cadeia de caracteres", "B", "1234" |
| Lógico | Representa valores lógicos (VERDADEIRO ou FALSO) |



Visual G

Entrada e saída

- **Entrada:** obtenção de dados provenientes do meio externo
 - Comando: **leia**
 - Exemplos:
 - **leia (X)**
 - **leia (A, NOTA)**

Visual G

Entrada e saída

- **Saída:** entrega dos resultados ao meio externo
- Comandos: **escreva** ou **escreval**
- Exemplos:
 - **escreva** (X)
 - **escreva** (B, MEDIA, 2+2)
 - **escreval** (“cliente cadastrado com sucesso”)



Visual G

Operador de atribuição



- Visualg: “:=” ou “←”
- Notação:
x1 ← 23;
temp ← x1;
nome ← “Carlos da Silva”;



Visual G

Operador Aritmetico

- **Dados de entrada:** tipo numérico (inteiro ou real)
- **Resultado:** tipo numérico (inteiro ou real)
- Exemplos:
 - $x_2 \leftarrow 2 + 3;$
 - $x_3 \leftarrow 3 / 2;$
 - $\text{alfa} \leftarrow 1 \text{ div } 5;$
 - $\text{resto} \leftarrow 10 \% 3;$
 - $\text{resto} \leftarrow 1 \% 4;$
 - $\text{delta} \leftarrow 5 * 5 - 4 * 1 * 4;$

Visual G

Operador Aritmetico

| Operação | Operador |
|------------------|------------------------------|
| Adição | + |
| Subtração | - |
| Multiplicação | * |
| Divisão | / |
| Divisão Inteira | \ ou div |
| Exponenciação | ^ ou Exp (<base>,<expoente>) |
| Resto da Divisão | % |

Visual G

Operador Relacional

- **Dados de entrada:** tipo numérico (int ou float) ou caracteres
- **Resultado:** tipo lógico

| Operador | Operação |
|------------|------------------------|
| $a < b$ | a é menor que b |
| $a \leq b$ | a é menor ou igual a b |
| $a > b$ | a é maior que b |
| $a \geq b$ | a é maior ou igual a b |
| $a = b$ | a é igual a b |
| $a \neq b$ | a não é igual a b |

Visual G

Operador Lógico

- **Dados de entrada:** tipo lógico
- **Resultado:** tipo lógico
- E (AND), OU (OR), NAO (NOT)

| Operação | Resultado |
|---------------|--------------------------------------------------------------------------------------------------------------------------|
| a E b | VERDADEIRO se ambas as partes (a e b) forem verdadeiras |
| a OU b | VERDADEIRO se apenas uma das partes (a ou b) é verdadeira. |
| NAO a | Nega uma afirmação, invertendo o seu valor lógico: se a for VERDADEIRO retorna FALSO, se a for FALSO retorna VERDADEIRO. |

Visual G

Operador Lógico

| a | b | a E b | a OU b | NAO a | NAO b |
|---|---|-------|--------|-------|-------|
| V | V | V | V | F | F |
| V | F | F | V | F | V |
| F | V | F | V | V | F |
| F | F | F | F | V | V |



Visual G

Estrutura condicional Simples

- Utilizada quando precisamos testar uma certa condição antes de executar uma ação

```
se <condição> entao  
    <bloco de ações>  
fimse
```



Visual G

Estrutura condicional Simples

```
Algoritmo "Media"
var
    // declaração de variáveis:
    N1, N2, NF, media : real
inicio
    // início do programa
    leia(N1,N2,NF)

    media ← (N1 + N2 + NF) / 3.0

    se (media ≥ 7.0) entao
        escreva("Aluno aprovado")
    fimse

fimalgoritmo
```



Visual G

Estrutura condicional Composta

- Utilizada em situações em que duas alternativas dependem da mesma condição, uma da condição verdadeira (então) e a outra da condição falsa (senão).

```
se <condição> entao  
    <bloco de ações1>  
senão  
    <bloco de ações2>  
fimse
```



Visual G

Estrutura condicional Composta

```
Algoritmo "Media2"  
var  
    // declaração de variáveis:  
    N1, N2, NF, media : real  
inicio  
    // início do programa:  
    leia(N1,N2,NF)  
  
    media  $\leftarrow$  (N1 + N2 + NF) / 3.0  
  
    se (media  $\geq$  5.0) entao  
        escreva("Aluno aprovado")  
    senão  
        escreva("Aluno reprovado")  
    fimse  
  
fimalgoritmo
```

Visual G

Estrutura condicional Composta

```
Algoritmo Maior3Numeros
var
  N1, N2, N3 : real
inicio
  leia (N1,N2,N3)

  se N1 ≥ N2 e N1 ≥ N3 entao
    escreva(N1, "é o maior")
  senao
    se N2 ≥ N1 e N2 ≥ N3 entao
      escrever(N2, "é o maior")
    senao
      escrever(N3, "é o maior")
    fimse
  fimse
fimse
fimalgoritmo
```



Visual G

Estruturas de repetição



- Permitem que uma sequência de comandos seja executada repetidamente, até que determinada condição de interrupção seja satisfeita

- ✓ Conhecidas como Loop onde cada repetição é uma interação
 - Com verificação no início
 - Com verificação no fim
 - Por contagem





Visual G

Estruturas de repetição

- Com verificação no início

```
enquanto <condição> faça  
    <ação 1>  
    <ação 2>  
    ...  
    <ação n>  
fimenquanto
```





Visual G

Estruturas de repetição

- Com verificação no início

```
soma ← 0
i ← 1
enquanto (i ≤ N) faca
    soma ← soma + i
    i ← i + 1
fimenquanto
escreval("a soma é:", soma)
```





Visual G

Estruturas de repetição

- Com verificação no Fim

```
repita  
    <ação 1>  
    <ação 2>  
    ...  
    <ação n>  
ate <condição>
```





Visual G

Estruturas de repetição

- Com verificação no Fim

```
soma ← 0
i ← 1
repita
    soma ← soma + i
    i ← i + 1
ate (i > N)
escreval("a soma é:", soma)
```





Visual G

Estruturas de repetição

- Repetição por contagem

```
para  variavel de inicio ate  
fim passo <incremento> faca  
    <ação 1>  
    <ação 2>  
    ...  
    <ação n>  
fimpara
```





Visual G

Estruturas de repetição



- Repetição por contagem

```
para variavel de inicio ate  
fim passo <incremento> faca  
    <ação 1>  
    <ação 2>  
    ...  
    <ação n>  
fimpara
```

- início: indica a variável de controle do laço (contador) e seu valor inicial.
- fim: define o valor final da variável de controle
- incremento: define como a variável de controle se altera a cada repetição





Visual G

Estruturas de repetição

- Repetição por contagem

```
soma ← 0
para i de 1 ate N passo 1
faca
    soma ← soma + i
fimpara
escreval("a soma é:", soma)
```



Var Nome : vetor [1..2] de Caracter nota:vetor [1..8] de real

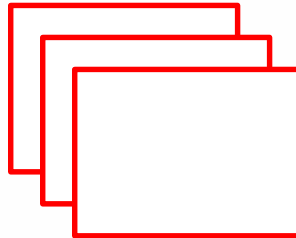
Linguagem C

- **Array, Vetores e Matrizes**

Um vetor ou array é uma estrutura de dados que armazena uma coleção de elementos, identificado por um índice.

Uma matriz é um vetor com 2 ou mais direções (matriz bidimensional ou tridimensional ou não desenhável)

| | | |
|---|---|---|
| A | B | C |
| D | E | F |
| G | H | I |



Exemplo:

Var

Nome : vetor [1..2] de Caracter

nota: vetor [1..8] de real

Nome_Matriz : vetor [1..2,1..5] de Caracter