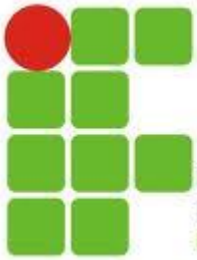


INSTITUTO FEDERAL DE
EDUCAÇÃO, CIÊNCIA E TECNOLOGIA
RIO GRANDE DO NORTE

AULA 13

PROCEDIMENTOS

Disciplina: Algoritmos e POO

Professora: Alba Lopes

alba.lopes@ifrn.edu.br

http://docente.ifrn.edu.br/albalopes

Procedimentos

- Em Visualg, os procedimentos diferem das funções apenas por não retornarem valor nenhum
- A sintaxe utilizada na criação de procedimentos é:

```
procedimento <nome do procedimento> (<parâmetros>)
var
       <declaração das variáveis locais>
inicio
       <lista de comandos>
fimprocedimento
```

PROCEDIMENTOS

- **Exemplo 1**: Crie um procedimento que receba um valor como parâmetro e escreva o dobro desse número.

```
algoritmo "Procedimento"
var
   numero: inteiro

   procedimento Dobro(valor: inteiro)
   var
      total: inteiro
   inicio
      total <- valor * 2
      escreva("O dobro é: ", total)
   fimprocedimento

inicio
   escreva("Digite um número: ")
   leia(numero)
   Dobro(numero)
fimalgoritmo
```

Não possui tipo de retorno e não retorna nada. Apenas executa o que está na seção de comandos

Procedimentos

- **Exemplo 1**: Crie um procedimento que receba um valor como parâmetro e escreva o dobro desse número.

```
algoritmo "Procedimento"
var
   numero: inteiro

   procedimento Dobro(valor: inteiro)
   var
      total: inteiro
   inicio
      total <- valor * 2
      escreva("O dobro é: ", total)
   fimprocedimento

inicio
   escreva("Digite um número: ")
   leia(numero)
   Dobro(numero)
fimalgoritmo
```

É feita apenas a chamada do procedimento, sem precisar atribuir a nenhuma variável.

Procedimentos

- **Exemplo 2**: Crie um procedimento que receba um número como parâmetro e escreva a tabuada desse número.

```
algoritmo "ProcedimentoTabuada"
var
   numero: inteiro

   procedimento Tabuada(valor: inteiro)
   var
      i: inteiro
   inicio
      escreval("*** Tabuada de ", valor, " ***")
      para i de 1 ate 10 faca
         escreval(valor, " *", i, " = ", valor*i)
      fimpara
   fimprocedimento

inicio
   escreva("Digite um número: ")
   leia(numero)
   Tabuada(numero)
fimalgoritmo
```

Procedimentos

- **Exemplo 3**: Crie um algoritmo que utilize o procedimento criado anteriormente, e escreva a tabuada dos números de 1 a 9:

...

```
    fimprocedimento

inicio
    para numero de 1 ate 9 faca
        Tabuada(numero)
    fimpara
fimalgoritmo
```

Procedimentos

- **Exemplo 4**: Crie um procedimento que, dado um número N por parâmetro, desenhe o seguinte padrão na tela:
  - Por exemplo, para N = 5

```
* * * * *
* * * * *
* * * * *
* * * * *
* * * * *
```

Procedimentos

- **Exemplo 4**:

```
algoritmo "ProcedimentoDesenharPadrao"
var
   numero: inteiro

   procedimento DesenharPadrao(n: inteiro)
   var
      i, j: inteiro
   inicio

      para i de 1 ate n faca
         para j de 1 ate n faca
            escreva ("* ")
         fimpara
         escreval
      fimpara

   fimprocedimento

inicio
   escreva("Digite um valor: ")
   leia(numero)
   DesenharPadrao(numero)
fimalgoritmo
```

Procedimentos

- **Exemplo 5**: Crie um procedimento que, dado um número N por parâmetro, desenhe o seguinte padrão na tela:
  - Por exemplo, para N = 5

```
1 * * * *
* 2     *
*   3   *
*     4 *
* * * * 5
```

PROCEDIMENTOS

- **Exemplo 5**:

```
algoritmo "ProcedimentoDesenharPadrao"
var
   numero: inteiro
procedimento DesenharPadrao(n: inteiro)
var
   i, j: inteiro
inicio

   para i de 1 ate n faca
      para j de 1 ate n faca
         se (i = j) entao
            escreva(j)
         senao
            se (i = 1) ou (i = n) entao
               escreva(" *")
            senao
               se (j = 1) ou (j = n) entao
                  escreva(" *")
               senao
                  escreva ("  ")
               fimse
            fimse
         fimse
      fimpara
      escreval
   fimpara

fimprocedimento
```

```
inicio
   escreva("Digite um valor: ")
   leia(numero)
   DesenharPadrao(numero)
fimalgoritmo
```

Exercícios Procedimentos

1. Escreva um procedimento que receba 3 valores reais X, Y e Z e que verifique se esses valores podem ser os comprimentos dos lados de um triângulo e, neste caso, escrever qual o tipo de triângulo esses valores formam.
2. Escreva um procedimento que receba um número inteiro positivo por parâmetro e escreva os divisores desse número.

   Crie um procedimento que, dado um número N por parâmetro, desenhe os seguintes padrões na tela:

3.
```
*
*   *
*       *
*           *
*   *   *   *   *
```

4.
```
1   2   3   4   5
2                   4
3                   3
4                   2
5   4   3   2   1
```

Passagem de Parâmetros

- **Parâmetros** são canais por onde os dados são transferidos pelo algoritmo chamador a um subalgoritmo.
  - **Parâmetros Formais** são os nomes simbólicos usados na definição dos parâmetros de um subalgoritmo.

```
procedimento soma(a, b: inteiro)
var
   calculo: inteiro
```

Parâmetros Formais

  - **Parâmetros Reais** são aqueles que substituem os parâmetros formais quando da chamada de um subalgoritmo.

```
inicio
   x <- 10
   y <- 2
   soma(x, y)
fimalgoritmo
```

Parâmetros Reais

Mecanismos de Passagem de Parâmetros

- A substituição dos parâmetros formais pelos parâmetros reais no ato da invocação de um subalgoritmo é denominada **de passagem de parâmetros**
- Pode se dar por dois mecanismos distintos:
  - Passagem por valor (ou por cópia)
  - Passagem por referência

Passagem por Valor

- Na passagem por valor, é criada uma cópia dos parâmetros reais
- As modificações efetuadas no parâmetro formal não afetam o parâmetro real, pois trabalha-se apenas com uma cópia.

Passagem por Valor

```
algoritmo "PassagemPorValor"
var
   x, y: inteiro

   procedimento soma(a, b: inteiro)
   var
      calculo: inteiro
   inicio
      calculo <- a + b
      escreval("O valor da soma é ", calculo)
      a <- 3
      b <- 4
   fimprocedimento

inicio
   x <- 1
   y <- 2
   soma(x, y)
   escreval("Os valores de x e y são: ", x, y)
fimalgoritmo
```

RESULTADO

```
O valor da soma é  3
Os valores de x e y são:  1 2

*** Fim da execução.
```

Mesmo alterando os valores de **a** e **b**, os valores de **x** e **y** continuam os mesmos após a chamada do subalgoritmo

Passagem por Referência

- O espaço de memória ocupado pelos parâmetros reais é compartilhado pelos parâmetros formais correspondentes
- As modificações efetuadas nos parâmetros formais também afetarão os parâmetros reais

- Na linguagem do Visualg, utiliza-se a palavra **var** antes do nome do parâmetro na declaração da função para informar que a passagem será por **referência**

Passagem por Referência

```
algoritmo "PassagemPorReferencia"
var
   x, y: inteiro

   procedimento soma(var a, b: inteiro)
   var
      calculo: inteiro
   inicio
      calculo <- a + b
      escreval("O valor da soma é ", calculo)
      a <- 3
      b <- 4
   fimprocedimento

inicio
   x <- 1
   y <- 2
   soma(x, y)
   escreval("Os valores de x e y são: ", x, y)
fimalgoritmo
```

A palavra **var** antes dos parâmetros indica passagem por referência

Passagem por Referência

```
algoritmo "PassagemPorReferencia"
var
   x, y: inteiro

   procedimento soma(var a, b: inteiro)
   var
      calculo: inteiro
   inicio
      calculo <- a + b
      escreval("O valor da soma é ", calculo)
      a <- 3
      b <- 4
   fimprocedimento

inicio
   x <- 1
   y <- 2
   soma(x, y)
   escreval("Os valores de x e y são: ", x, y)
fimalgoritmo
```

RESULTADO

```
O valor da soma é  3
Os valores de x e y são:  3 4

*** Fim da execução.
```

Agora, ao alterar os valores de **a** e **b**, os valores de **x** e **y** são também alterados

Passagem por Referência

- **Exemplo 1**: Crie um procedimento que receba dois valores inteiros por parâmetro e realize a troca desses valores.

```
algoritmo "TrocarValores"
var
   x, y: inteiro

   procedimento troca(var a, b: inteiro)
   var
      auxiliar: inteiro
   inicio
      auxiliar <- a
      a <- b
      b <- auxiliar
   fimprocedimento

inicio
   x <- 5
   y <- 8
   escreval("Os valores de x e y ANTES da troca são: ", x, y)
   troca(x, y)
   escreval("Os valores de x e y DEPOIS da troca são: ", x, y)
fimalgoritmo
```

Passagem por Referência

- Um mesmo subalgoritmo pode ter parâmetros que são passados por valor e outros que são passados por referência:

Os parâmetros **a** e **b** são passados por valor e o parâmetro **valorMedia** é passado por referência

```
algoritmo "ValorEReferencia"
var
   n1, n2, resultado: real

procedimento media(a, b: real; var valorMedia: real)
var
   soma: real
inicio
   soma <- a + b
   valorMedia <- soma/2
fimprocedimento

inicio
   n1 <- 5
   n2 <- 2
   media(n1, n2, resultado)
   escreva("O valor da média é: ", resultado)
fimalgoritmo
```

Exercícios

1. Crie um procedimento que receba dois valores por referência e ordene-os em ordem crescente. Crie um algoritmo principal para chamar o procedimento e exibir os valores após a ordenação.

2. Crie um procedimento para resolver uma equação de segundo grau. O procedimento deve receber 5 parâmetros: os coeficientes **a**, **b** e **c** da equação (por valor), e **raiz1** e **raiz2** (por referência). Crie um algoritmo principal que leia os coeficientes da equação e chame o procedimento. Em seguida, mostre o resultado das raízes.

3. Crie um procedimento que receba uma frase por parâmetro e remova todos os caracteres de espaços da frase. Ex: a frase “O livro está em cima da mesa” deverá ficar como: “Olivroestáemcimadamesa”.

Variáveis Locais e Globais

- As variáveis locais são visíveis apenas dentro dos subalgoritmo que as criou

- Já as variáveis globais são visíveis tanto no algoritmo principal como nos subalgoritmos
  - Por esse motivo, é importante que as funções/procedimentos sejam declarados na seção **var** após a declaração das variáveis globais

- É importante não criar variáveis locais e globais com o mesmo nome para evitar ambiguidades.

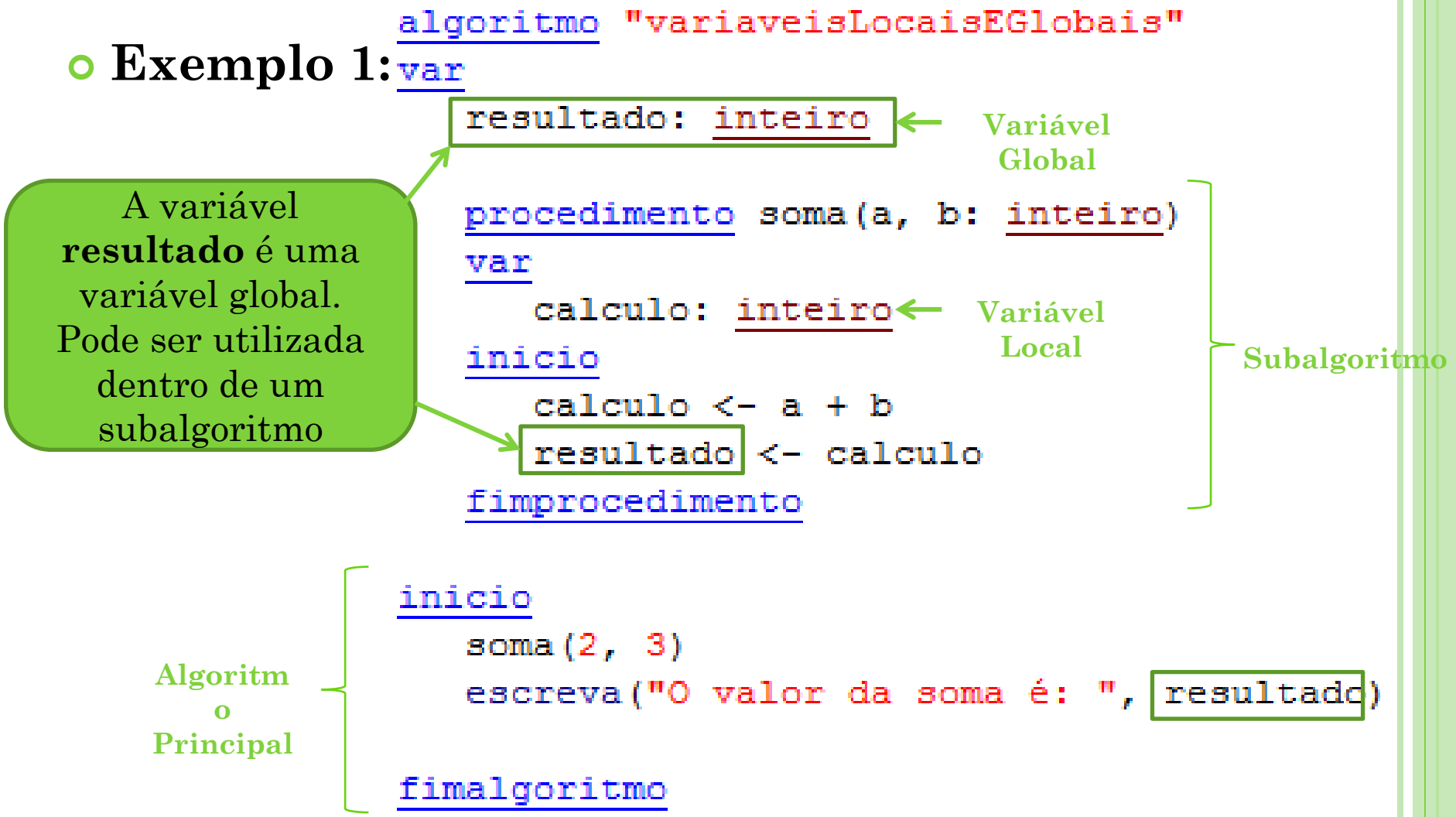
Variáveis Locais e Globais

- **Exemplo 1:**

```
algoritmo "variaveisLocaisEGlobais"
var
   resultado: inteiro

   procedimento soma(a, b: inteiro)
   var
      calculo: inteiro
   inicio
      calculo <- a + b
      resultado <- calculo
   fimprocedimento

inicio
   soma(2, 3)
   escreva("O valor da soma é: ", resultado)

fimalgoritmo
```

Variáveis Locais e Globais

- **Exemplo 1:**



Variáveis Locais e Globais

- **Exemplo 1:**
  - Já a utilização de variáveis locais dentro do algoritmo principal não é permitida. A execução de um algoritmo como o mostrado abaixo causará um **erro**:

```
algoritmo "variaveisLocaisEGlobais"
var
   resultado: inteiro

   procedimento soma(a, b: inteiro)
   var
      calculo: inteiro
   inicio
      calculo <- a + b
      resultado <- calculo
   fimprocedimento

inicio
   soma(2, 3)
   escreva("O valor da soma é: ", calculo)

fimalgoritmo
```

A variável **calculo** é uma variável local do procedimento **soma** e **NÃO** pode ser utilizada dentro do algoritmo principal

Variáveis Locais e Globais

- O Visualg ainda não permite a passagem de Vetores e Matrizes como parâmetros de subalgoritmos
- Para criar subalgoritmos que precisam de vetores e matrizes, utilize variáveis globais

- **Exemplo 1:** Crie um procedimento para preencher um vetor de posições

Variáveis Locais e Globais

- **Exemplo 1:**

```
algoritmo "vetorEmSubalgoritmos"
var
   numeros: vetor [1..10] de inteiro

   procedimento preencherVetor
   var
      i: inteiro
   inicio
      para i de 1 ate 10 faca
         escreva("Digite o valor para a posição ", i, " do vetor: ")
         leia(numeros[i])
      fimpara
   fimprocedimento

inicio
   preencherVetor
fimalgoritmo
```

Variáveis Locais e Globais

- **Exemplo 2:** Crie um procedimento para exibir o conteúdo de um vetor

```
algoritmo "AlgImprimirVetor"
var
   numeros: vetor [1..10] de inteiro
   cont: inteiro
   procedimento ImprimirVetor
   var
      i: inteiro
   inicio
      para i de 1 ate 10 faca
         escreva(numeros[i])
      fimpara
   fimprocedimento

inicio
   para cont de 1 ate 10 faca
      numeros[cont] <- cont
   fimpara
   ImprimirVetor
fimalgoritmo
```

Variáveis Locais e Globais

- **Exemplo 3:** Crie um procedimento para imprimir o conteúdo de um vetor de 10 posições. Esse procedimento deve receber um parâmetro do tipo caractere que indica se o conteúdo deve ser exibido na ordem correta ou na ordem inversa (“C” para correta e “I” para inversa)

Variáveis Locais e Globais

- **Exemplo 3:**

```
algoritmo "AlgImprimirVetor"
var
   numeros: vetor [1..10] de inteiro
   cont: inteiro

   procedimento ImprimirVetor(ordem: caractere)
   var
      i: inteiro
   inicio

      se (ordem = "C") entao
         para i de 1 ate 10 faca
            escreva(numeros[i])
         fimpara
      senao
         se (ordem = "I") entao
            para i de 10 ate 1 passo -1 faca
               escreva(numeros[i])
            fimpara
         fimse
      fimse

   fimprocedimento

inicio
   para cont de 1 ate 10 faca
      numeros[cont] <- cont
   fimpara
   ImprimirVetor ("I")
   ImprimirVetor ("C")
fimalgoritmo
```

Variáveis Locais e Globais

- **Exemplo 4:** Crie uma função para inserir um elemento em um vetor.

  Crie um variável global do tipo vetor de inteiro de 10 posições e uma variável global para indicar quantas posições preenchidas o vetor possui.
  A função de inserção deve receber por parâmetro o valor a ser inserido no vetor. Se o vetor já estiver cheio, a função deve retornar um valor lógico (**falso**) para informar que não foi possível inserir um valor no vetor. Caso contrário, deve retornar verdadeiro.

Variáveis Locais e Globais

- **Exemplo 4:.**

```
algoritmo "FuncaoInserirValorVetor"
var
   numeros: vetor [1..10] de inteiro
   qtd, i: inteiro
   teste: logico

   funcao inserir(valor: inteiro): logico
   var
      //Nenhuma variável local necessária
   inicio
      se (qtd >= 10) entao
         retorne falso
      senao
         qtd <- qtd + 1
         numeros[qtd] <- valor
         retorne verdadeiro
      fimse
   fimfuncao

inicio
   qtd <- 0
   para i de 1 ate 20 faca
      teste <- inserir(i)
      se (teste = falso) entao
         escreval("Impossível inserir valor ", i , ". Vetor cheio!")
      fimse
   fimpara
fimalgoritmo
```

Variáveis Locais e Globais

- **Exemplo 5:** Ao algoritmo anterior, acrescente o procedimento para imprimir o conteúdo do vetor. Note que a função deve mostrar apenas as posições preenchidas do vetor. Mesmo sendo um vetor de 10 elementos, se apenas 1 tiver sido inserido, a função deve mostrar apenas esse elemento.elemento em um vetor.

Variáveis Locais e Globais

- **Exemplo 5:**

```
procedimento imprimir
var
   p: inteiro
inicio
   para p de 1 ate qtd faca
      escreva(numeros[p])
   fimpara
   escreval
fimprocedimento
```

Variáveis Locais e Globais

- **Exemplo 6:** Ao algoritmo anterior, acrescente a função para remover um elemento em um vetor.

  Crie um variável global do tipo vetor de inteiro de 10 posições e uma variável global para indicar quantas posições preenchidas o vetor possui.
  A função de remoção deve receber por parâmetro a posição do elemento a ser removido. Se a posição informada não possui nenhum valor, retornar o valor falso. Caso a remoção seja bem sucedida, retornar verdadeiro.

  Obs: *Note que, ao remover um valor do vetor, todos os valores após o valor removido devem ser deslocados em uma posição*

Variáveis Locais e Globais

- **Exemplo 6:**

```
funcao remover(posicao: inteiro): logico
var
   p: inteiro
inicio
   se (posicao > qtd) entao
      retorne falso
   senao
      para p de posicao ate qtd-1 faca
         numeros[p] <- numeros[p+1]
      fimpara
      qtd <- qtd - 1
      retorne verdadeiro
   fimse
fimfuncao
```

Variáveis Locais e Globais

- **Exemplo 7:** Crie um algoritmo principal para testar as funções/procedimentos criadas

```
inicio
   qtd <- 0
   para i de 1 ate 20 faca
      teste <- inserir(i)
      se (teste = falso) entao
         escreval("Impossivel inserir valor ", i, ". Vetor cheio!")
      fimse
   fimpara

   imprimir

   teste <- remover(10)
   se (teste = falso) entao
       escreval("Impossível remover valor na posição ", i , "!")
   fimse

   imprimir

   teste <- remover(5)
   se (teste = falso) entao
       escreval("Impossível remover valor na posição ", i , "!")
   fimse

   imprimir

fimalgoritmo
```

Exercícios

1. Crie uma função que receba um valor por parâmetro um nome (caractere) e verifique se esse valor se encontra em um vetor (também de caractere, declarado globalmente). Retorne verdadeiro caso o valor se encontre e falso, caso contrário. Crie um algoritmo principal para testar a função criada.

2. Crie uma função que receba por parâmetro um número inteiro e verifique quantas vezes esse valor se encontra em um vetor (declarado globalmente). Retorne a quantidade de vezes que o valor se encontra no vetor. Crie um algoritmo principal para testar a função criada.

3. Crie um procedimento que receba por parâmetro um número inteiro e remova todas as ocorrências desse valor de um vetor declarado globalmente.

Exercícios

4. Escreva um algoritmo que apresente um menu com quatro opções:
   - 1 - Inserir
   - 2 - Remover
   - 3 - Imprimir
   - 4 - Substituir
   - 5 - Sair

   Considere um vetor de inteiro de 20 posições declarado globalmente. Quando for escolhida a:
   - Opção 1: um subalgoritmo **Insere** deve ser chamado para inserir o valor na próxima posição livre do vetor
   - Opção 2: chamar o subalgoritmo **Remover** e eliminar o elemento na posição passada como parâmetro
   - Opção 3: chamar o subalgoritmo **Imprimir** para escrever na tela os valores que se encontram no algoritmo
   - Opção 4: chamar o subalgoritmo **Substituir**, que recebe por parâmetro um valor e uma posição. O subalgoritmo deve substituir o valor existente na posição informada pelo valor passado por parâmetro. Se ainda não houver elemento na posição informada, deve-se inserir o valor na próxima posição livre.
   - Opção 5: O programa deverá ser encerrado.

Referências

- NAPRO – Núcleo de Apoio Aprendizagem de Programação. Disponível em: http://www.guanabara.info/logica/Apostilas/VisuAlg_Ref.pdf
- http://www.inf.pucrs.br/~pinho/LaproI/Exercicios/SeqDecisao/lista1.htm
- http://www.inf.pucrs.br/flash/lapro/listafunc.html