EAD UNISANTA

LÓGICA DE PROGRAMAÇÃO

Me. Claudio F. de Carvalho

GUIA DA
DISCIPLINA

Objetivo geral

O módulo de Lógica de Programação tem como principal objetivo fornecer os subsídios necessários para uma correta estruturação, resolução e validação de variados problemas computacionais, despertando o interesse pelo desenvolvimento de soluções lógicas eficazes.

Introdução

Caro(a) estudante, espero que, através do módulo de Lógica de Programação, que você neste momento inicia, eu seja capaz de lhe despertar para a importância do estudo dos diversos assuntos que permeiam o estudo dos algoritmos e da lógica de programação, que considero de imprescindível importância para a sua formação acadêmica e profissional.

Ao final deste módulo, e dos diversos temas e tópicos que estamos prestes a iniciar a seguir, espero que você seja capaz de, de forma consistente e padronizada, produzir soluções computacionais conceituais que poderão ser eventualmente implementadas em qualquer linguagem de programação, caso você pretenda trilhar o caminho da área de desenvolvimento.

Espero de antemão que você goste do “mergulho” que está prestes a fazer neste novo mundo.

Prontos e prontas? Vamos lá!

Prof. Alexandre Sobrino.

1. CONCEITOS BÁSICOS

1.1. Variáveis

Na matemática, uma variável é normalmente identificada por um símbolo, ou uma letra, tal como a, b, c, x, y ou z, entre outras possibilidades, correspondendo a um número a ser descoberto.

Se refletirmos sobre esta questão, agora analisando-a do ponto de vista computacional, uma variável é representada por um valor (não necessariamente numérico, como veremos a seguir) ou expressão, que precisa ser armazenado na memória do computador para posterior ou imediata utilização.

Como você já deve imaginar, a exemplo do que também acontece na matemática, uma variável possui um nome, como x, y ou z, além de outros mais sofisticados, devendo também ser de um determinado tipo, conceito que será descrito a seguir.

Exemplos:

x=3

y=2.5

nomeProfessor=”Alexandre Sobrino”

1.2. Tipos de dados

Variáveis, como já dissemos, podem armazenar diferentes tipos de informações. Ao criarmos uma variável, devemos inicialmente refletir sobre o tipo de informação que nela será armazenado.

1.2.1. Numérico

Uma variável cujo tipo de dado seja numérico pode armazenar valores inteiros ou reais (por vezes conhecidos como fracionários), quer sejam estes positivos ou negativos.

Exemplos:

z=5

peso=70

y=-1

altura=1.80
notaProva=9.5

1.2.2. Alfanumérico

Uma variável cujo tipo de dado seja alfanumérico pode armazenar e manipular números, letras ou símbolos, sendo facilmente reconhecidas pelo uso das aspas como delimitadores do conteúdo nela armazenada, conforme veremos abaixo.

Por vezes, este tipo de dados é conhecido por outros nomes igualmente válidos, corretos e amplamente utilizados, como string ou caractere.

Exemplos:
nomeProfessor=”Alexandre Sobrino”
nomeDisciplina=”Lógica de Programação”
pais=”Brasil”

1.2.3. Lógico

Este tipo de dados possui apenas dois valores possíveis, que são *true* (verdadeiro) ou *false* (falso).

Por vezes, este tipo de dados é também conhecido como *booleano*, em reconhecimento ao trabalho do matemático britânico George Boole, de importância inegável tanto para o desenvolvimento da computação quanto da álgebra.

Exemplos:
estudaLogica=true
estudaQuimica=false

1.3. Exercícios de fixação

Importante

1.3.1. Analise os seguintes valores abaixo disponíveis e insira nas respectivas lacunas os tipos de dados identificados de acordo com a legenda que se segue: [A] para alfanumérico/caractere, [NRP] para numérico real positivo, [NRN] para numérico real negativo, [NIP] para numérico inteiro positivo, [NIN] para numérico inteiro negativo e [L] para lógico.

a) [ ] 5.0
b) [ ] “Louis Pasteur”
c) [ ] -8.75
d) [ ] 10
e) [ ] false
f) [ ] “7.5”
g) [ ] “true”
h) [ ] -3
i) [ ] “8 x 5”
j) [ ] “-100”

1.3.2. Considerando o que o estudamos sobre nomes de variáveis, atribua [V] para os nomes de variáveis que podem ser considerados válidos e [F] para aqueles que não são.

a) [ ] nota2
b) [ ] preço unitário
c) [ ] mediaFinal
d) [ ] VALOR
e) [ ] taxa de cambio
f) [ ] fatorCalculo

2. OPERADORES (PARTE 1)

2.1. Introdução

Operadores representam elementos essenciais na realização de cálculos e comparações, e são simplesmente imprescindíveis para o trabalho que realizaremos ao longo deste nosso módulo.

2.2. Tipos de operadores

2.2.1. Operadores aritméticos

Como o próprio nome faz supor, os operadores ditos aritméticos são os que permitem a realização de cálculos diversos.

São eles:

+	Adição
-	Subtração
*	Multiplicação
/	Divisão
^	Potenciação
%	Resto de divisão

Exemplos:

2+7 resultará em 9
3-1 resultará em 2
7.5*2 resultará em 15
15/3 resultará em 5
(4-2)/4 resultará em 0.5
2^4 resultará em 16
10%3 resultará em 1

2.2.2. *Operadores relacionais*

Possivelmente também já vistos em algum momento de sua vida estudantil, os operadores relacionais são elementos simplesmente fundamentais no processo de tomada de decisão computacional. Expressões que os envolvam resultam em *true* (verdadeiro) ou *false* (falso).

São eles:

>	Maior que
<	Menor que
>=	Maior ou igual a
<=	Menor ou igual a
=	Igual a
<>	Diferente de

Exemplos:

1>2 resultará em *false (falso)*

5<10 resultará em *true (verdadeiro)*

8>=6 resultará em *true (verdadeiro)*

5<=3 resultará em *false (falso)*

4=5 resultará em *false (falso)*

7<>8 resultará em *true (verdadeiro)*

2.3. Exercícios de fixação

Importante

2.3.1. Considerando as seguintes variáveis e seus valores (X=2, Y=4, Z=5, e W=7) determine os resultados aritméticos provenientes das seguintes expressões:

a) [ ] X+W
b) [ ] X+Y/Z
c) [ ] W/Y
d) [ ] X^Y
e) [ ] (X+Y)/Z
f) [ ] (Y+Z)-X
g) [ ] Y%X
h) [ ] (X+W)*Z
i) [ ] Z%Y

2.3.2. Considerando, nas situações cabíveis, as seguintes variáveis e seus valores (A=1, B=2, C=3 e D=2), determine os resultados das expressões a seguir:

a) [ ] 5>8
b) [ ] 10<=20
c) [ ] 15>5
d) [ ] 3<>10
e) [ ] A=C
f) [ ] B=D
g) [ ] A<=C
h) [ ] B<>D
i) [ ] C>=B

3. OPERADORES (PARTE 2)

3.1. Introdução

Neste capítulo, estendemos o nosso trabalho acerca de operadores, de imprescindível importância para os assuntos que serão estudados nas etapas posteriores do nosso curso.

3.2. Tipos de operadores

3.2.1. Operadores lógicos

Trabalhando em conjunto com os operadores relacionais, estudados em nosso capítulo anterior, os chamados operadores lógicos permitem o desenvolvimento das chamadas expressões lógicas. Tal como estudados no tópico sobre operadores relacionais, expressões envolvendo estes operadores igualmente resultarão em *true* (verdadeiro) ou *false* (falso).

3.2.1.1. Operador .e.

Proveniente do inglês AND, o operador lógico .e. apresentará o resultado lógico *true* (verdadeiro) quando todas as expressões analisadas, não importando quantas sejam, forem verdadeiras.

Entendendo

O resultado de uma expressão envolvendo o operador .e. (AND) será verdadeiro somente se todas as demais expressões envolvidas na análise forem verdadeiras.

Exemplo:

5>6	.e.	6<8	resultará em *false* (falso)
(*false*)		(*true*)	

3.2.1.2. Operador .ou.

Do inglês OR, o operador lógico .ou. apresentará o resultado lógico *true* (verdadeiro) quando pelo menos uma das expressões analisadas, novamente não importando quantas sejam, forem verdadeiras.

Entendendo

O resultado de uma expressão envolvendo o operador .ou. (OR) será verdadeiro se pelo menos uma das expressões envolvidas na análise for verdadeira.

Exemplo:

10>=7 (*true*) .ou. 7=5 (*false*) resultará em *true* (verdadeiro)

3.2.1.3. Operador .não.

Também conhecido como operador lógico de negação, do inglês NOT, o operador .não. inverte (ou nega) o valor lógico de uma expressão a ele associada. Logo, se uma determinada expressão resulta em *true*, a submissão da mesma a este operador resultará em *false*. Do mesmo modo, se uma determinada expressão resulta em *false*, submetido a este operador, resultará em *true*.

Exemplo:

.não. (5>6) (*false*) resulta em *true* (verdadeiro)

3.3. Exercícios de fixação

Importante

3.3.1. Considerando as seguintes variáveis e seus valores (X=5, Y=5 e Z=8) para as situações cabíveis, determine os resultados das expressões a seguir:

a) [ ] (X-Y)>=(Z*Y)

b) [ ] (30<>50) .ou. (50>60)

c) [ ] (X>=Y) .e. (X<=Z)

d) [ ] .não.(.não.(X<>Y))

e) [ ] .não.((X>Y) .e. (X<>Z))

f) [ ] (Z>=Y) .ou. (Y>X)

g) [ ] .não. (X<>Z)

h) [ ] ((X>=Z) .ou. (Y>Z) .ou. (Y=Z)) .e. .não. (X<=Y)

i) [ ] ((Y>=X) .ou. (Z>Y)) .ou. (Z<>X) .ou. (X>Y)

4. ALGORITMOS

4.1. Introdução

Podemos dizer que um algoritmo corresponde a uma sequência ordenada, mas finita, de passos capaz de resolver um problema computacional.

Entendendo

Se você pretende atuar na área de desenvolvimento de software, esforce-se para aprender, primeiramente, como definir o comportamento da sua solução do ponto de vista lógico, deixando a codificação apenas para a etapa posterior.

Como estudamos, algoritmos podem ser produzidos fazendo uso de diferentes estratégias, que podem ser tanto gráficas como textuais.

4.2. Pseudocódigo

Neste módulo, optamos por fazer uso de uma estratégia textual conhecida como pseudocódigo – algo como “falso código”.

Abaixo, temos um exemplo de um pseudocódigo que calcula e apresenta na tela a média aritmética obtida por um(a) determinado(a) estudante.

Entendendo

Apresentarei, na próxima página deste guia, um algoritmo completo destinado a resolver o problema acima. Após acompanhar a aula contendo a minha explicação, leia-o com a devida atenção, inclusive mais de uma vez se considerar necessário, de modo a se familiarizar com os nomes das instruções que utilizaremos na escrita de nossas soluções com o auxílio de pseudocódigo. Você os identificará visualmente com relativa facilidade, já que, neste guia, serão destacados em negrito. Para facilitar ainda mais o seu entendimento, incluí neste algoritmo-exemplo, à direita de cada instrução apresentada, uma legenda explicativa.

```
algoritmo calculaMedia                          ◂ nome do algoritmo

variáveis                                       ◂ área de declaração de variáveis
          NOTA1, NOTA2, MEDIA: real

início                                          ◂ início do algoritmo

escreva "Digite a primeira nota:"               ◂ exibição de mensagem em tela
leia NOTA1                                      ◂ entrada de dados
escreva "Digite a segunda nota:"                ◂ exibição de mensagem em tela
leia NOTA2                                      ◂ entrada de dados
MEDIA=(NOTA1+NOTA2)/2                           ◂ processamento ou cálculo
escreva "A media é:", MEDIA                     ◂ exibição de mensagem em tela

fim                                             ◂ final do algoritmo
```

4.3. Teste de mesa

Com o intuito de observarmos se nosso raciocínio computacional está realmente correto, podemos fazer uso de uma ferramenta bastante eficiente, popularmente conhecida como teste de mesa.

Na tabela a seguir, podemos examinar o resultado proveniente da execução do algoritmo-exemplo anteriormente apresentado, permitindo-nos aferir que nosso "falso código" cumpre realmente o objetivo proposto.

NOTA1	NOTA2	MEDIA	Tela
8.5	6.5	7.5	Digite a primeira nota
			Digite a segunda nota
			A média é 7.5

4.4. Exercícios de fixação

Importante

4.4.1. Elabore um algoritmo onde, a partir de um determinado valor em centímetros fornecido pelo teclado, seja exibido na tela o valor equivalente em metros.

4.4.2. A partir do fornecimento do preço de um determinado produto em dólar, e do valor da cotação do dólar do dia, crie um algoritmo capaz de apresentar o valor do respectivo produto em reais.

4.4.3. Admitindo que a primeira nota digitada pelo usuário tenha peso 1 e a segunda nota tenha peso 2, produza um algoritmo capaz de calcular a média ponderada de um(a) determinado(a) estudante.

4.4.4. Considerando que um vendedor receberá uma comissão fixa de 3% sobre o valor de cada produto vendido, elabore um algoritmo onde, a partir do fornecimento do valor do produto vendido, seja apresentado na tela o valor da devida comissão.

4.4.5. A partir da digitação do salário atual de um determinado colaborador e do respectivo porcentual de reajuste que está recebendo de seu empregador, escreva um algoritmo capaz de apresentar na tela qual será o seu próximo salário.

5. DESVIOS CONDICIONAIS SIMPLES

5.1. Introdução

Fato é que um computador é capaz de tomar decisões através da verificação de uma ou mais condições definidas pelo desenvolvedor ou desenvolvedora. A partir desta verificação, é capaz de, por exemplo, realizar um cálculo, apresentar uma mensagem na tela ou, até mesmo, simplesmente nada fazer, razão pela qual estudaremos, a partir de agora, os chamados desvios condicionais.

5.2. Condições

Do ponto de vista técnico, uma condição é representada por uma expressão relacional, assunto já abordado no capítulo 2 de nosso módulo.

Tal como vimos na ocasião, uma expressão relacional resultará invariavelmente em *true* (verdadeiro) ou *false* (falso).

Para recapitular, reexaminemos algumas situações.

Exemplos:

5>2 resultará em *true (verdadeiro)*

10<=5 resultará em *false (falso)*

5.3. Tipos de desvios condicionais

5.3.1. Desvios condicionais simples

Podemos dizer que estamos diante de um desvio condicional simples quando nosso algoritmo efetua o processamento de instruções exclusivamente para o caso da condição a ser analisada resultar em *true* (verdadeiro).

O pseudocódigo a seguir, e que faz uso da instrução de pseudocódigo `se`, demonstra a implementação de uma situação desta natureza.

Entendendo

Uma vez que já as demais seções existentes neste pseudocódigo já foram devidamente apresentadas em nossa lição anterior, observe com mais atenção o trecho de pseudocódigo onde surge a instrução `se`, que utilizaremos para a realização de desvios condicionais.

Perceba também que a instrução `escreva`, apresentada com maior recuo de modo a sinalizar visualmente seu vínculo com a instrução `se`, somente será executada caso o primeiro valor digitado pelo usuário efetivamente for igual ou maior do que o segundo. Cabe destacar que, ainda que não obrigatório, o uso deste recuo é desejável.

Por fim, observe que a instrução `fim_se` deve ser utilizada como delimitadora deste trecho de pseudocódigo.

```
algoritmo exemploDesvioSimples

variáveis
          A, B: inteiro

início

leia A,B
se (A>=B) então
     escreva "A é maior ou igual a B"
fim_se

fim
```

Entendendo

Tenha em mente que, caso a condição analisada retorne o valor lógico *false* (falso), nosso algoritmo nada fará. Como vimos, essa é uma característica que nos permite identificar rapidamente este tipo de desvio condicional.

5.4. Exercícios de fixação

Importante

5.4.1. Crie um algoritmo que receba um número via teclado que supostamente deverá ser positivo, apresentando uma mensagem de confirmação na tela exclusivamente neste caso.

5.4.2. Adapte o exercício anterior de modo que agora sejam recebidos dois números supostamente positivos, igualmente apresentando na tela uma mensagem de confirmação desta condição.

5.4.3. Adapte a solução anterior de modo que os números sejam exibidos na tela apenas se o primeiro for positivo, e o segundo, negativo.

5.4.4. Crie um algoritmo que receba pelo teclado uma nota de 0 a 10, apresentando-a na tela apenas se for válida.

5.4.5. Elabore um algoritmo que receba via teclado dois nomes próprios, apresentando uma confirmação na tela se pelo menos um deles for exatamente Alexandre.

6. DESVIOS CONDICIONAIS COMPOSTOS

6.1. Introdução

Como estudamos em nossa lição anterior, os desvios condicionais permitem que um computador seja efetivamente capaz de tomar decisões. Vimos também que uma decisão computacional, invariavelmente, leva em consideração a verificação de uma ou mais condições.

6.2. Tipos de desvios condicionais

6.2.1. Desvios condicionais compostos

Anteriormente, descobrimos que, quando estamos diante de um desvio condicional simples, nossa solução computacional conceitual realiza o processamento de instruções apenas quando a condição a ser analisada produz o resultado lógico *true* (verdadeiro).

Mas o que deveríamos fazer caso quiséssemos que realizasse algum tipo de processamento também quando a análise da condição resultasse no valor lógico *false* (falso)? Veremos, a seguir, uma adaptação do pseudocódigo referente ao algoritmo apresentado em nossa lição anterior e que reflete justamente essa possibilidade, configurando-se no chamado desvio condicional composto.

```
algoritmo exemploDesvioComposto

variáveis
          A, B: inteiro

início

leia A, B
se (A>=B) então
     escreva "A é maior ou igual a B."
senão
     escreva "A não é maior ou igual a B."
fim_se

fim
```

Entendendo

Observe que, em um desvio condicional composto, devemos utilizar a instrução de pseudocódigo `senão` de modo que uma ou mais instruções sejam executadas também se a condição analisada resultar no valor lógico *false* (falso).

6.3. Estratégias adicionais com desvios condicionais

6.3.1. Desvios condicionais encadeados

Em determinadas situações, podemos optar, ou mesmo precisar, que condições sejam analisadas, e que decisões sejam tomadas, de forma encadeada.

Na prática, isso significa que poderíamos implementar desvios condicionais, representados pela instrução de pseudocódigo `se` e objeto de nosso estudo atual, dentro de outros desvios condicionais.

Entendendo

O pseudocódigo a seguir demonstra uma situação típica envolvendo desvios condicionais encadeados.

No exemplo apresentado na próxima página, podemos observar que a digitação de um segundo valor será solicitada ao usuário apenas se o primeiro número for positivo.

Por fim, observe que, se os dois valores digitados forem positivos, esses serão devidamente somados, armazenados na variável SOMA e, apenas depois disso, apresentados na tela.

Cabe destacar que nenhuma mensagem de alerta será apresentada ao usuário caso um dos valores digitados não seja positivo – situação que você poderá opcionalmente implementar, se assim o desejar.

```
algoritmo exemploDesvioEncadeado

variáveis
          A, B, SOMA: inteiro

início

leia A
se (A>0) então
     leia B
     se (B>0) então
          SOMA=A+B
          escreva "Soma dos valores:", SOMA
     fim_se
fim_se

fim
```

6.3.2. Seleção de casos

Através da seleção de casos podemos tornar o processo de tomada de decisão de nossas soluções muito mais organizado e legível, impedindo o sequenciamento e a utilização excessiva de desvios condicionais.

Observe, a seguir, o fragmento de um pseudocódigo destinado a apresentar, por extenso, um determinado número digitado pelo usuário.

```
(...)
leia N

se (N=1) então
     escreva "Um"
fim_se

se (N=2) então
     escreva "Dois"
fim_se

se (N=3) então
     escreva "Três"
fim_se

se (N=4) então
     escreva "Quatro"
fim_se
(...)
```

Perceba, por fim, através do exemplo abaixo, como a seleção de casos é capaz de proporcionar um pseudocódigo não apenas mais legível, mas também mais organizado e, consequentemente, de maior qualidade.

Entendendo

A seleção de casos deve ser implementada através da instrução de pseudocódigo caso, devidamente delimitada pela instrução fim_caso. Observe ainda que caso nenhuma das condições verificadas através da instrução de pseudocódigo seja resulte em verdadeiro, podemos apresentar uma mensagem de alerta ao usuário com o auxílio da instrução senão.

```
Algoritmo selecaoDeCasos

variáveis
          N: inteiro

início

leia N

caso N
     seja 1 faça
          escreva "Um"
     seja 2 faça
          escreva "Dois"
     seja 3 faça
          escreva "Três"
     seja 4 faça
          escreva "Quatro"
senão
     escreva "Número for a da faixa permitida"
fim_caso

fim
```

6.4. Exercícios de fixação

Importante

6.4.1. Elabore um algoritmo capaz de apresentar na tela o maior de dois números inteiros digitados.

6.4.2. Adapte o algoritmo anterior de modo a garantir que os dois números digitados sejam efetivamente diferentes entre si.

6.4.3. Aperfeiçoe a solução do exercício 4.4.3 de modo que, se a média do(a) estudante for pelo menos 7 (sete), seja apresentada uma mensagem indicando sua aprovação. Caso esta pontuação não tenha sido atingida, apresente na tela a mensagem "Estudante de exame".

6.4.4. Escreva um algoritmo capaz de exibir na tela o maior de três números inteiros distintos fornecidos via teclado.

6.4.5. Escreva um algoritmo que, após receber pelo teclado uma média válida de um(a) estudante, atribua-lhe e apresente na tela o respectivo conceito de acordo com a tabela abaixo. Tenha em mente que o usuário, respeitando o sistema de notas da instituição, somente digitará valores de médias que evoluam de 0.5 em 0.5 (como 4.5, 5, 5.5, 6, etc.), o que justifica os valores presentes na tabela abaixo.

Média	Conceito
9,0 ou mais	A
7,0 a 8,5	B
6,0 a 6,5	C
4.0 a 5,5	D
3.5 ou menos	E

6.4.6. Caso não o tenha feito desta forma, adapte o núcleo de decisões do problema anterior de modo a resolvê-lo através de seleção de casos. Caso assim o já tenha solucionado, faça-o agora através de estruturas de decisão encadeadas.

7. ESTRUTURAS DE CONTROLE DE REPETIÇÃO (PARTE 1)

7.1. Introdução

A programação com laços (ou *loops*) permite que um algoritmo – e, naturalmente, também um futuro programa de computador - possa repetir, de forma controlada e organizada, uma ou mais instruções definidas pelo desenvolvedor ou desenvolvedora. Uma vez codificados, tais trechos de um programa de computador podem efetivamente ser repetidos quantas vezes forem necessários.

Neste nosso módulo de Lógica de Programação estudaremos algumas das principais e mais conhecidas estratégias envolvendo as chamadas estruturas de controle de repetição.

7.2. Estruturas de controle de repetição incondicionais

7.2.1. Estrutura de controle de repetição `para`/`fim_para`

A estrutura de controle de repetição `para`, representada por uma instrução de pseudocódigo de mesmo nome, delimitada por uma instrução `fim_para` e considerada uma estrutura de laço incondicional, envolve a utilização de um contador, que possui a responsabilidade de controlar o número de repetições ao qual as instruções serão submetidas.

Faz também parte desta estratégia a indicação do passo deste contador, ou seja, do incremento - ou decremento - ao qual a variável responsável por tal contagem será submetida.

O pseudocódigo a seguir, implementado com o auxílio da estrutura de controle de repetição `para/fim_para`, demonstra como uma mesma instrução de pseudocódigo poderia ser repetida uma determinada quantidade de vezes de modo a apresentar na tela os números de 1 a 5.

```
algoritmo apresentaNumeros1a5

variáveis
          A: inteiro

início

para A de 1 até 5 passo 1
     escreva A
fim_para

fim
```

Entendendo

Para uma melhor fixação do funcionamento deste tipo de estratégia de programação, sugiro, como aliás sempre o faço, que não deixe de realizar o respectivo teste de mesa deste e de outros algoritmos.

Ainda que entenda que o teste de mesa seja um recurso simplesmente vital para a compreensão do funcionamento de qualquer algoritmo, especialmente neste momento em que você está treinando o seu raciocínio computacional e conhecendo o funcionamento das instruções e estratégias disponíveis, essa ferramenta torna-se especialmente mais valiosa nos seus primeiros contatos com estruturas de controle de repetição.

7.3. Exercícios de fixação

Importante

7.3.1. Escreva um algoritmo que, utilizando a estrutura de controle de repetição para/fim_para, exiba na tela os números existentes entre 150 e 250, inclusive estes.

7.3.2. Desenvolva um algoritmo capaz de exibir na tela, em ordem decrescente, os números existentes entre 200 e 1, inclusive estes. Dica do professor Alexandre: lembre-se da importância da instrução responsável pelo controle do passo para a implementação deste algoritmo.

7.3.3. Elabore um algoritmo capaz de exibir na tela os números ímpares existentes entre 1000 e 2000.

7.3.4. Crie um algoritmo que apresente na tela todos os múltiplos de 5 existentes entre 5 e 50, inclusive estes.

7.3.5. Utilize a estrutura de controle de repetição para/fim_para de modo a exibir na tela todos os números pares existentes entre 400 e 800, inclusive estes.

7.3.6. Escreva um algoritmo capaz de apresentar na tela a somatória de todos os números existentes entre 1 e 10, inclusive estes.

8. ESTRUTURAS DE CONTROLE DE REPETIÇÃO (PARTE 2)

8.1. Introdução

Em nossa última lição, apresentamos a importância das estruturas de controle de repetição, que permitem que uma ou mais instruções sejam repetidas de forma organizada e, é claro, controlada.

Complementando o tema, ainda que existam outras, nesta disciplina do curso estudaremos outras duas estruturas de suma importância, conhecidas como **faça/enquanto e enquanto/faça.**

8.2. Estruturas de controle de repetição condicionais

8.2.1. Estrutura de controle de repetição `enquanto/faça`

A estrutura de controle de repetição `enquanto/faça`, um dos objetos de estudo desta lição, diferentemente da estrutura `para/fim_para`, apresentada no capítulo anterior, é considerada um tipo de laço condicional, o que significa que, na prática, conta com a explícita verificação de uma condição pré-determinada de modo que as instruções existentes dentro da estrutura sejam executadas.

No momento em que a condição for verificada, tantas quantas vezes resultar no valor lógico *true* (verdadeiro), e esta é uma informação extremamente relevante para o entendimento do funcionamento desta estrutura, as instruções presentes no laço seguirão sendo repetidas.

Entendendo

A estrutura de controle de repetição enquanto/faça seguirá repetindo o conjunto de instruções determinadas em seu interior enquanto a condição determinada, no momento da verificação, resultar no valor lógico *true* (verdadeiro). No momento em que a condição analisada resultar no valor lógico *false* (falso), o laço será imediatamente interrompido.

Mais uma vez, estamos diante de um pseudocódigo capaz de, agora com o auxílio da estrutura de controle de repetição `enquanto/faça`, apresentar na tela os números existentes entre 1 e 5. Observe que, agora por meio desta estrutura, estamos estabelecendo, de forma explícita, o comportamento da variável envolvida no controle de repetição (vide instruções envolvendo a variável I), além de determinar, como já mencionado, a condição que, enquanto for verdadeira, promoverá a repetição do conjunto de instruções desejado em nosso algoritmo:

```
algoritmo apresentaNumeros1a5

variáveis
          I: inteiro

início

I=1

enquanto (I<=5) faça
     escreva I
     I=I+1
fim_enquanto

fim
```

8.2.2. Estrutura de controle de repetição `faça/enquanto`

Igualmente considerado um tipo de laço condicional, a estrutura de controle de repetição `faça/enquanto`, assim como `enquanto/faça`, anteriormente estudada, também está associada à verificação de uma condição que precisará resultar no valor lógico *true* (verdadeiro) para que as instruções existentes dentro da estrutura sejam executadas.

Entendendo

A exemplo da estrutura de controle de repetição enquanto/faça, faça/enquanto também seguirá repetindo o conjunto de instruções estabelecidas enquanto a condição determinada, no momento da verificação, resultar no valor lógico *true* (verdadeiro). O laço, portanto, novamente será interrompido quando a condição verificada gerar o valor lógico *false* (falso).

Ainda que discreta, há, no entanto, uma diferença entre as duas estruturas de controle de repetição neste capítulo de nosso curso: no caso da estrutura `faça/enquanto`, diferentemente da estrutura `enquanto/faça`, é conveniente observar que a verificação da condição envolvida no controle da repetição será realizada apenas após o término do primeiro ciclo de execução das instruções estabelecidas. Isso significa que, independentemente do atendimento à condição determinada, o bloco de instruções interno à estrutura será executado pelo menos uma vez.

Finalizamos este capítulo com o pseudocódigo que, agora utilizando a estrutura de controle de repetição `faça/enquanto`, também apresenta na tela os números existentes entre 1 e 5.

```
algoritmo apresentaNumeros1a5

variáveis
          I: inteiro

início

I=1

faça
     escreva I
     I=I+1
enquanto (I<=5)

fim
```

8.3. Exercícios de fixação

Importante

8.3.1. Elabore um algoritmo que exiba na tela os números existentes entre 150 e 250 (inclusive estes), mas agora fazendo uso da estrutura de controle de repetição `faça/enquanto`.

8.3.2. Utilize a estrutura de controle de repetição `enquanto/faça` de modo a exibir na tela, em ordem decrescente, os números existentes entre 200 e 1, inclusive estes.

8.3.3. Adapte o algoritmo correspondente do capítulo anterior de modo que por meio da estrutura de controle de repetição `faça/enquanto` sejam exibidos na tela os números ímpares existentes entre 1001 e 1999, inclusive estes.

8.3.4. Fazendo agora uso da estrutura `enquanto/faça`, apresente na tela todos os múltiplos de 5 existentes entre 5 e 50, inclusive estes.

8.3.5. Nesta oportunidade fazendo uso da estrutura de controle de repetição `faça/enquanto`, apresente na tela todos os números pares existentes entre 400 e 800, inclusive estes.

8.3.6. Escreva, agora por meio da estrutura de controle de repetição `enquanto/faça`, um algoritmo capaz de apresentar na tela a somatória de todos os números existentes entre 1 e 10, inclusive estes.

8.3.7. Escreva um algoritmo capaz de apresentar na tela os 20 primeiros termos daquela que é conhecida como a "Sequência de Fibonacci" (1, 2, 3, 5, 8, 13, 21, 34, 55...). Utilize, a seu critério, a estrutura de controle de repetição `enquanto/faça` ou `faça/enquanto` para a solução dessa questão.

9. VARIÁVEIS INDEXADAS

9.1. Introdução

Uma variável simples, como já estudamos, são aquelas capazes de armazenar tão somente um único valor de cada vez. Ainda que seja possível, caso fosse necessário armazenarmos várias dezenas, centenas ou mesmo milhares de valores diferentes, poderíamos, sim, criar um igual número de variáveis simples. Isso, no entanto, dificilmente seria algo plausível, já que temos à nossa disposição as assim chamadas variáveis indexadas.

Também conhecida pelo nome de vetor ou *array*, uma variável indexada, por vezes também chamada de variável composta, permite que, sob um mesmo nome, possamos agrupar séries de valores capazes de armazenar dados de um mesmo tipo, que podem ser diretamente acessados com o auxílio de uma variável a qual caberá o papel de índice.

A estrutura abaixo introduz a representação de uma variável indexada chamada MEDIA, que armazena as notas de 10 estudantes.

`MEDIA[]`

[0]	[1]	[2]	[3]	[4]	[5]	[6]	[7]	[8]	[9]
6.5	7.5	7.0	9.0	6.5	8.0	7.0	8.0	7.0	3.5

O pseudocódigo presente na página seguinte demonstra como duas estruturas de controle de repetição `para/fim_para`, dispostas em sequência, são utilizadas de modo que um vetor, denominado `MEDIA[]` e com 10 posições, seja em um primeiro momento populado pelo usuário, valor a valor, com dados recebidos através do teclado (instrução `leia`).

É importante destacar que cada uma das médias digitadas está sendo armazenada em uma posição diferente da estrutura da variável definida para a tarefa (a já citada `MEDIA[]`), e que o valor armazenado em cada uma das posições poderá ser acessado com o auxílio da variável `I`, que cumprirá, como já dissemos anteriormente, o papel de

índice. Torna-se ainda importante frisarmos que, ainda que faça clara referência à palavra índice, poderia ter qualquer outro nome, não sendo obrigatório chamar-se `I`.

Por fim, o pseudocódigo é encerrado com uma segunda estrutura de controle de repetição `para/fim_para`, a qual cabe a missão de apresentar na tela, um a um, os valores armazenados no vetor (instrução `escreva`).

```
algoritmo  exemploVariaveisIndexadas

variáveis
      I: inteiro
      MEDIAS: vetor [0..9] de real

início

para I de 0 até 9 passo 1 faça
     leia MEDIAS[I]
fim_para

para I de 0 até 9 passo 1 faça
     escreva MEDIAS[I]
fim_para

fim
```

9.2. Exercícios de fixação

Importante

9.2.1. Elabore um algoritmo capaz de receber e armazenar, em uma variável indexada e utilizando a estrutura de controle de repetição `para/fim_para`, o dobro de 20 números inteiros quaisquer digitados pelo usuário, apresentando-os ao término das entradas.

9.2.2. Crie um algoritmo que receba via teclado e armazene, com o auxílio de uma variável indexada, 30 números inteiros positivos digitados pelo usuário. Concluídas as entradas, sua solução deverá apresentá-los em ordem inversa - isto é, do último para o primeiro número fornecido. Importante: você deverá utilizar, obrigatoriamente, a estrutura de controle de repetição `faça/enquanto` para atender a pelo menos uma das necessidades apresentadas neste enunciado.

9.2.3. Adapte o algoritmo anteriormente desenvolvido de modo a armazenar, com o auxílio de uma variável indexada e da estrutura de controle de repetição

`enquanto/faça`, os primeiros 20 termos da chamada "Sequência de Fibonacci". Ao término, sua solução deverá apresentar na tela, utilizando qualquer estrutura de controle de repetição à sua escolha, além dos termos propriamente ditos, a somatória total dos 20 termos.

9.2.4. Utilizando a estrutura de controle de repetição `faça/enquanto` em algum aspecto da sua solução, elabore um algoritmo que receba via teclado e armazene as médias (válidas) de 30 estudantes. Concluídas as entradas, e considerando que a média a ser obtida em um determinado curso é 6, sua solução deverá apresentar na tela a quantidade total de estudantes aprovados(as) e de exame.

9.2.5. Adapte o algoritmo anterior de modo que, após todas as entradas, e a partir da digitação de uma determinada nota a ser buscada, seja exibido na tela o total de estudantes que obtiveram especificamente a nota digitada. Caso não haja nenhum caso, exiba na tela um aviso ao usuário.

10. RESPOSTAS

1.3.1 a) NRP

b) A

c) NRN

d) NIP

e) L

f) A

g) A

h) NIN

i) A

j) A

1.3.2 a) V

b) F

c) V

d) V

e) F

f) V

2.3.1. a) 9

b) 2.8

c) 1.75

d) 16

e) 1.2

f) 7

g) 0

h) 45

i) 1

2.3.2. a) F

b) V

c) V

d) V

e) F

f) V

g) V

h) F

i) V

3.3.1. a) F

b) V

c) V

d) F

e) V

f) V

g) F

h) F

i) V

4.4.1.

```
algoritmo converteCentParaMetro
variáveis
        VC: real
início
escreva "Digite o valor em centímetros:"
leia VC
escreva "Valor em metros:", VC/100
fim
```

4.4.2. **algoritmo** converteDolarParaReal

variáveis

VP,VD,PR: **real**

início

escreva "Digite o valor do produto (em dólares):"

leia VP

escreva "Digite o valor do dólar:"

leia VD

PR=VP*VD

escreva "O preço a pagar em reais é: ", PR

fim

4.4.3. **algoritmo** calculaMediaPonderada

variáveis

NOTA1, NOTA2, MEDIA: **real**

início

escreva "Digite a primeira nota:"

leia NOTA1

escreva "Digite a segunda nota:"

leia NOTA2

MEDIA=(NOTA1+NOTA2*2)/3

escreva "A média ponderada é:", MEDIA

fim

4.4.4. **algoritmo** calculaComissao

variáveis

CV, VPV, CF: **real**

início

CF=0.03

escreva "Digite o valor em reais do produto vendido:"

leia VPV

CV=VPV*CF

escreva "Comissão do vendedor: R$ ", CV

fim

4.4.5. **algoritmo** reajustaSalario
variáveis
SC, PR, SN: **real**
início
CF=0.03
escreva "Digite o valor do atual salário do colaborador:"
leia SC
escreva "Digite o porcentual de reajuste:"
leia PR
SN=SC+(SC*PR/100)
escreva "O novo salário do colaborador é:", SN
fim

5.4.1. **algoritmo** exibeSePositivo
variáveis
N: **real**
início
escreva "Digite um número positivo:"
leia N
se (N>0) **então**
escreva "Confirmação: o número digitado é mesmo positivo."
fim_se
fim

5.4.2. **algoritmo** exibeSePositivos
variáveis
N1, N2: **real**
início
escreva "Digite um primeiro número positivo:"
leia N1
escreva "Digite um segundo número positivo:"
leia N2
se (N1>0) **.e.** (N2>0) **então**
escreva "Confirmação: os números digitados são positivos."
fim_se
fim

5.4.3. **algoritmo** exibeSePositivoENegativo
variáveis
N1, N2: **real**
início
escreva "Digite um primeiro número positivo:"
leia N1
escreva "Digite um segundo número negativo:"
leia N2
se (N1>0) **.e.** (N2<0) **então**
escreva "Confirmação: o primeiro número digitado é positivo, e o segundo, negativo."
escreva "Positivo:", N1
escreva "Negativo:", N2
fim_se
fim

5.4.4. **algoritmo** exibeNotaValida

```
variáveis
        N: real
início
escreva "Digite uma nota válida (0 a 10):"
leia N
se (N>=0) .e. (N<=10) então
    escreva "Confirmação: a nota fornecida (",N,") é válida."
fim_se
fim
```

5.4.5. **algoritmo** confirmaNomeProfessor

```
variáveis
        N1, N2: alfanumérico
início
escreva "Digite o primeiro nome:"
leia N1
escreva "Digite o segundo nome:"
leia N2
se (N1="Alexandre") .ou. (N2="Alexandre") então
    escreva "Confirmação: um dos nomes digitados é Alexandre."
fim_se
fim
```

6.4.1. **algoritmo** maiorDeDoisNumeros
variáveis
N1, N2: **inteiro**
início
escreva "Digite o primeiro número:"
leia N1
escreva "Digite o segundo número:"
leia N2
se (N1>N2) **então**
escreva "Maior número:", N1
senão
escreva "Maior número:", N2
fim_se
fim

6.4.2. **algoritmo** maiorDeDoisNumeros2
variáveis
N1, N2: **inteiro**
início
escreva "Digite o primeiro número:"
leia N1
escreva "Digite o segundo número:"
leia N2
se (N1=N2) **então**
escreva "Os números fornecidos precisam ser diferentes. Algoritmo encerrado."
senão
se (N1>N2) **então**
escreva "Maior número:", N1
senão
escreva "Maior número:", N2
fim_se
fim_se
fim

6.4.3. **algoritmo** calculaMediaPonderada2
variáveis
NOTA1, NOTA2, MEDIA: **real**
início
escreva "Digite a primeira nota:"
leia NOTA1
escreva "Digite a segunda nota:"
leia NOTA2
MEDIA=(NOTA1+NOTA2*2)/3
escreva "A média ponderada é:", MEDIA
se (MEDIA>=7) **então**
escreva "Estudante aprovado(a)."
senão
escreva "Estudante de exame."
fim_se
fim

6.4.4. **algoritmo** maiorDeTresNumeros

```
variáveis
        N1, N2, N3: inteiro
início
escreva "Digite o primeiro número:"
leia N1
escreva "Digite o segundo número:"
leia N2
escreva "Digite o terceiro número:"
leia N3

se (N1=N2) .ou. (N1=N3) .ou. (N2=N3) então
        escreva "Os números fornecidos precisam ser diferentes. Algoritmo encerrado."
senão
        se (N1>N2) .e. (N1>N3) então
                escreva "Maior número:", N1
        senão
                se (N2>N3) então
                        escreva "Maior número:", N2
                senão
                        escreva "Maior número:", N3
                fim_se
        fim_se
fim_se
fim
```

6.4.5. **algoritmo** mediaEConceito
variáveis
M: **real**
C: **alfanumérico**
início
escreva "Digite a média:"
leia M
se (M>=0) **.e.** (M<=10) **então**
se (M>=9) **então**
C="A"
senão
se (M>=7) **então**
C="B"
senão
se (M>=6) **então**
C="C"
senão
se (M>=4) **então**
C="D"
senão
C="E"
fim_se
fim_se
fim_se
fim_se
escreva "Conceito: ", C
senão
escreva "A média fornecida é inválida."
fim_se
fim

6.4.6. **algoritmo** mediaEConceito2

```
variáveis
        M: real
        C: alfanumérico
início
escreva "Digite a média:"
leia M
se (M>=0) .e. (M<=10) então
        caso N
                seja 8.5, 9, 9.5, 10 faça
                        C="A"
                seja 7, 7.5, 8 faça
                        C="B"
                seja 6, 6.5 faça
                        C="C"
                seja 4, 4.5, 5, 5.5 faça
                        C="D"
                senão
                        C="E"
        fim_caso
        escreva "Conceito:", C
senão
        escreva "A média fornecida é inválida."
fim_se
fim
```

7.3.1.
```
algoritmo exibe150a250
variáveis
        N: inteiro
início
para N de 150 até 250 passo 1 faça
        escreva N
fim_para
fim
```

7.3.2.
```
algoritmo exibe200a1
variáveis
        N: inteiro
início
para N de 200 até 1 passo -1 faça
        escreva N
fim_para
fim
```

7.3.3.
```
algoritmo exibe1001a2000
variáveis
        N: inteiro
início
para N de 1001 até 2000 passo 2 faça
        escreva N
fim_para
fim
```

7.3.4.
```
algoritmo multiplosDe5
variáveis
        N: inteiro
início
para N de 5 até 50 passo 5 faça
        escreva N
fim_para
fim
```

7.3.5.

```
algoritmo paresDe400a800
variáveis
        N: inteiro
início
para N de 400 até 800 passo 2 faça
        escreva N
fim_para
fim
```

7.3.6.

```
algoritmo somatoria1a10
variáveis
        N, S: inteiro
início
S=0
para N de 1 até 10 passo 1 faça
        S=S+N
fim_para
escreva "Soma dos números:", S
fim
```

8.3.1.

```
algoritmo exibe150a250
variáveis
        N: inteiro
início
N=150
faça
        escreva N
        N=N+1
enquanto (N<=250)
fim
```

8.3.2. **algoritmo** apresenta200a1

```
variáveis
        N: inteiro
início
N=200
enquanto (N>=1) faça
        escreva N
        N=N-1
fim_enquanto
fim
```

8.3.3. **algoritmo** exibe1001a2000

```
variáveis
        N: inteiro
início
N=1001
faça
        escreva N
        N=N+2
enquanto (N<=2000)
fim
```

8.3.4. **algoritmo** multiplosDe5

```
variáveis
        N: inteiro
início
N=5
enquanto (N<=50) faça
        escreva N
        N=N+5
fim_enquanto
fim
```

8.3.5. **algoritmo** exibePares400a800
variáveis
N: **inteiro**
início
N=398
faça
N=N+2
escreva N
enquanto (N<800)
fim

8.3.6. **algoritmo** somatoria1Ate10
variáveis
N, S: **inteiro**
início
N=0
S=0
enquanto (N<10) **faça**
N=N+1
S=S+N
fim_enquanto
escreva "Soma dos números:", S
fim

8.3.7. **algoritmo** 20TermosFibonacci
variáveis
TA, TB, TF, N: **inteiro**
início
TA=0
TB=1
N=0
enquanto (N<20) **faça**
TF=TA+TB
escreva TF
TA=TB
TB=TF
N=N+1
fim_enquanto
fim

9.2.1. **algoritmo** vetorNumerosDobrados
variáveis
VETDOBRO: **vetor**[0..19] **de inteiro**
NUMERO, I: **inteiro**
início
para I **de** 0 **até** 19 **passo** 1 **faça**
leia NUMERO
VETDOBRO[I]=NUMERO*2
fim_para
para I **de** 0 **até** 19 **passo** 1 **faça**
escreva VETDOBRO[I]
fim_para
fim

9.2.2.

```
algoritmo vetor30NumerosPositivos
variáveis
        VETNUMEROS: vetor[0..29] de inteiro
        NUMERO, I: inteiro
início
I=0
enquanto (I<=29) faça
        faça
                leia NUMERO
        enquanto (NUMERO<=0)
        VETNUMEROS[I]=NUMERO
        I=I+1
fim_enquanto

para I de 29 até 0 passo -1 faça
        escreva VETNUMEROS[I]
fim_para
fim
```

9.2.3. **algoritmo** vetorFibonacci

variáveis

VETFIBO: **vetor**[0..19] **de inteiro**

I, TA, TB, TF, ST: **inteiro**

início

```
TA=0
TB=1
ST=0

enquanto (I<20) faça
        TF=TA+TB
        ST=ST+TF
        VETFIBO[I]=TF
        TA=TB
        TB=TF
        I=i+1
fim_enquanto

para I de 0 até 19 passo 1 faça
        escreva VETFIBO[I]
fim_para

escreva "Somatória dos termos:", ST
fim
```

9.2.4. **algoritmo** vetorAprovadosReprovados
variáveis
VETMEDIAS: **vetor**[0..29] **de real**
MEDIA: **real**
I, TA, TE: **inteiro**
início
TA=0
TE=0
I=0
faça
faça
leia MEDIA
enquanto (MEDIA<0) **.ou.** (MEDIA>10)

VETMEDIAS[I]=MEDIA

se (VETMEDIAS[I]>=6) **então**
TA=TA+1
senão
TE=TE+1
fim_se

I=I+1
enquanto (I<=29)

escreva "Total de alunos aprovados:", TA
escreva "Total de alunos de exame:", TE
fim

9.2.5. **algoritmo** vetorAprovadosReprovados2.0

```
variáveis
        VETMEDIAS: vetor[0..29] de real
        MEDIA: real
        I, TA, TE, TN: inteiro
início
TA=0
TE=0
TN=0
I=0

faça
        faça
                leia MEDIA
        enquanto (MEDIA<0) .ou. (MEDIA>10)

        VETMEDIAS[I]=MEDIA

        se (VETMEDIAS[I]>=6) então
                TA=TA+1
        senão
                TE=TE+1
        fim_se

        I=I+1
enquanto (I<=29)

faça
        escreva "Quer descobrir quantos(as) tiraram determinada nota? Digite-a agora:”
        leia MEDIA
enquanto (MEDIA<0) .ou. (MEDIA>10)
```

```
para I de 0 até 29 passo 1 faça
        se (VETMEDIAS[I]=MEDIA) então
                TN=TN+1
        fim_se
fim_para

se (TN=0) então
        escreva "Esta nota não foi tirada por nenhum(a) estudante."
senão
        escreva "Total de estudantes que tiraram a nota informada:", TN
fim_se


escreva "Total de alunos aprovados:", TA
escreva "Total de alunos de exame:", TE
fim
```

Referências

ALVES, William Pereira. ***Linguagem e lógica de Programação***. 1 ed. São Paulo: Érica, 2014.

ASCENCIO, Ana Fernanda; CAMPOS, Edilene Veneruchi. ***Fundamentos da programação de computadores***. 3 ed. São Paulo: Pearson Education do Brasil, 2012.

FARRELL, Joyce. **Lógica e design de programação**. São Paulo: Cengage Learning, 2010.

MANZANO, José Augusto e OLIVEIRA, Jayr Figueiredo de. ***Algoritmos: lógica para desenvolvimento de programação de computadores***. 26 ed. São Paulo: Érica, 2012.

PIVA, Dilermando et al. ***Algoritmos e programação de computadores***. Rio de Janeiro: Elsevier, 2012.

VENANCIO, Claudio. ***Desenvolvimento de algoritmos: uma nova abordagem***. 2. ed. São Paulo: Érica, 2000.

Concluindo

Caro(a) estudante, espero que, neste módulo do seu curso, eu tenha conseguido suscitar o seu interesse pelo fascinante mundo de possibilidades proporcionado pelo estudo da Lógica de Programação e dos algoritmos.

Lembre-se sempre: antes de efetivamente colocar energia na solução de um problema, quer este seja computacional ou não, permita-se parar por um instante para apenas refletir sobre seus diferentes aspectos, colocando a sua energia na solução apenas em um momento posterior.

Espero sinceramente que você tenha gostado dessa nossa incursão por esse maravilhoso universo!

Dúvidas? Você poderá me encontrar em sobrino@unisanta.br.

Saúde, paz e sucesso!

Prof. Alexandre Sobrino.