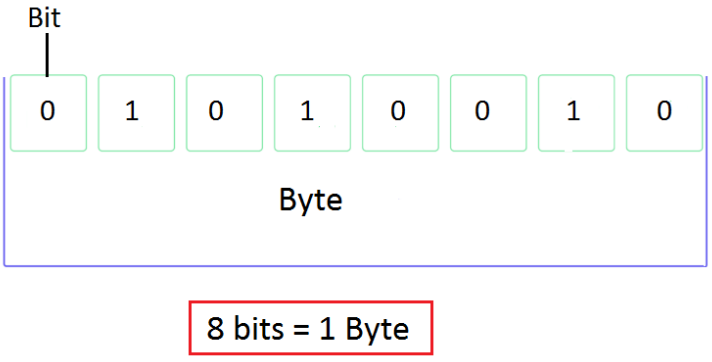
Lógica de Programação

Módulo 02

Variáveis, Constantes e Operadores

Tipos de Dados

Você já deve saber que tudo que o computador compreende é representado através de Zeros e Uns – os famosos *bits*. Como não é possível representar todos os caracteres através apenas de bits, foram criados os *bytes* (conjuntos de oito *bits*).



8 bits = 1 Byte

O que acontece quando você pressiona a tecla "A" no seu teclado? Este caractere é 'convertido' para a sequência de *bits* 01000001, que representa o caractere "A". Ou seja, 01000001 é o byte que representa o caractere "A". Dessa maneira, todos os dados que são armazenados pelo computador podem ser representados por zeros e uns.

Quando um programador vai desenvolver um sistema ele precisa armazenar uma série de dados para que o programa consiga realizar as tarefas necessárias. Vamos recordar um algoritmo que vimos no módulo anterior. Vimos um algoritmo que lê o nome e as 4 notas bimestrais de um aluno. Em seguida o algoritmo calcula e escreve a média obtida.

```
ALGORITMO MEDIA_FINAL;
VAR
   NOTA1, NOTA2, NOTA3, NOTA4, MEDIA: REAL;
   NOME: CARACTERE;
INICIO
   LEIA (NOME);
   LEIA (NOTA1, NOTA2, NOTA3, NOTA4);
   MEDIA := (NOTA1 + NOTA2 + NOTA3 + NOTA4) / 4;
```

```
    ESCREVA (NOME, MEDIA);
FIM.
```

Observe que o algoritmo vai armazenar os dados das notas e a média num formato que foi chamado de **REAL**. Já o nome será armazenado num formato que foi chamado de **CARACTERE**. REAL e CARACTERE neste caso são **tipos de dados**. O tipo de um dado define o conjunto de valores ao qual o dado pertence, bem como o conjunto de todas as operações que podem atuar sobre qualquer valor daquele conjunto de valores.

Quais são os possíveis **tipos de dados** que podemos usar? Observe o esquema abaixo:

- INTEIRO: este é o tipo de dado para valores numéricos negativos ou positivos, *sem casas decimais*. Serve, Por exemplo, para armazenar uma idade.
- REAL: este é o tipo de dado para valores numéricos negativos ou positivos, *com casas decimais*. Poderíamos armazenar o peso, por exemplo.
- LÓGICO: Este tipo de dado pode assumir apenas dois valores VERDADEIRO ou FALSO. Também é conhecido como booleano. Reserva apenas um bit na memória, onde, normalmente, o valor 1 representa VERDADEIRO e o valor 0 representa FALSO. Poderia armazenar um dado referente a algo que está ligado ou desligado.
- LITERAL (TEXTO): Tipo de dado que armazena uma sequência de caracteres que podem ser letras, dígitos e símbolos especiais. São representados nos algoritmos pelo delimitador aspas (“) no seu início e término.

Normalmente as linguagens de programação chamam esses tipos de dados que citamos aqui de **tipos primitivos**. É importante frisar que muitas vezes são criados subtipos ou até mesmo outros tipos de dados. Você deve aprender os tipos de dados de cada linguagem quando for trabalhar com ela.

Variáveis

Vamos retomar novamente nosso exemplo para compreender o que são variáveis.

```
ALGORITMO MEDIA_FINAL;
VAR
   NOTA1, NOTA2, NOTA3, NOTA4, MEDIA: REAL;
   NOME: CARACTERE;
INICIO
   LEIA (NOME);
   LEIA (NOTA1, NOTA2, NOTA3, NOTA4);
   MEDIA := (NOTA1 + NOTA2 + NOTA3 + NOTA4) / 4;
   ESCREVA (NOME, MEDIA);
FIM.
```

Observe que antes do bloco **INICIO** nós criamos um bloco que começa por **VAR**. Esse **VAR** indica que ali vamos colocar nossas variáveis. Em algumas linguagens é necessário declarar as variáveis dessa forma. Em outras você pode declarar variáveis em qualquer lugar do código. Mas o que é uma variável?

Uma variável é um espaço reservado na memória do computador para armazenar um tipo de dado determinado. Elas devem receber nomes para poderem ser referenciadas e modificadas quando necessário. Um programa deve conter declarações que especificam de que tipo são as variáveis que ele utilizará e, às vezes, um valor inicial.

No nosso algoritmo **MEDIA_FINAL** temos SEIS variáveis. Cinco são do tipo de dado REAL e uma é do tipo de dado CARACTERE.

Veja mais algumas características das variáveis:

- É uma entidade destinada a guardar uma informação.
- Chama-se variável, pois o valor contido nela varia com o tempo, ou seja, não é um valor fixo.
- O conteúdo de uma variável pode ser alterado, consultado ou apagado quantas vezes forem necessárias no algoritmo.
- Ao alterar o conteúdo de uma variável, a informação anterior é perdida. Ou seja, a variável armazena sempre a última informação recebida.
- Em geral, uma variável possui três atributos: **nome, tipo de dado** e **valor** armazenado por ela.

Nome	Deve começar com uma letra e não deve conter nenhum caractere especial, exceto o underline/underscore (_).
Tipo de dado	Conforme visto anteriormente: inteiro, real, literal (caractere), lógico, etc.
Valor	De acordo com o tipo de dado definido.

Por exemplo, podemos criar uma variável chamada "peso" para armazenar o peso de uma pessoa. Você pode imaginar uma variável como uma gaveta etiquetada. Chamamos este espaço alocado na memória de variável porque o valor armazenado neste espaço de memória pode ser alterado ao longo do tempo, ou seja, o valor ali alocado é **variável** ao longo do tempo.

Regras para Criação de Variáveis

Existem algumas regras que devem ser seguidas no momento de fornecer um nome para a variável. Vejamos:

- Devem ser iniciadas sempre por uma letra.
- Não devem conter caracteres especiais.
- Não devem conter espaços em branco.
- Não devem conter hífen entre os nomes (utilize underline).

Além dessas regras, muitas linguagens trazem um guia de estilo para a criação de variáveis. Por exemplo, no Java uma variável deve seguir o padrão camelCase com a

primeira letra em minúscula. Já no Delphi é recomendado que também se siga o padrão CamelCase, mas com a primeira letra em maiúscula. Dessa forma, se fôssemos criar uma variável para armazenar a altura da porta do carro, o nome dela ficaria assim nessas linguagens:

Delphi	AlturaPortaCarro
Java	alturaPortaCarro

E como você vai saber qual estilo usar em cada linguagem? Você deve se preocupar com isso quando for usar a linguagem em questão. Não faz sentido querer saber neste momento os estilos para as linguagens existentes. Uma vez que você vá trabalhar com uma linguagem, estude seus tipos de dados e seu estilo de codificação e mantenha-se alinhado com o estilo.

Atribuição de Valores

Esse é outro tópico muito importante. Quais são mesmo os três atributos das variáveis? Vamos recordar.

Nome	Deve começar com uma letra e não deve conter nenhum caractere especial, exceto o underline/underscore (_).
Tipo de dado	Conforme visto anteriormente: inteiro, real, literal (caractere), lógico, etc.
Valor	De acordo com o tipo de dado definido.

Agora retornando ao nosso exemplo da média final:

```
ALGORITMO MEDIA_FINAL;
VAR
   NOTA1, NOTA2, NOTA3, NOTA4, MEDIA: REAL;
   NOME: CARACTERE;
INICIO
   LEIA (NOME);
   LEIA (NOTA1, NOTA2, NOTA3, NOTA4);
   MEDIA := (NOTA1 + NOTA2 + NOTA3 + NOTA4) / 4;
   ESCREVA (NOME, MEDIA);
FIM.
```

Veja que temos uma instrução para ler o nome LEIA(NOME). Nesse momento, o valor que o usuário digitar no teclado será atribuído à variável NOME. Logo depois temos o LEIA(NOTA1, NOTA2, NOTA3, NOTA4). O que vai acontecer aqui é que o computador vai solicitar para o usuário digitar quatro números e vai atribuir cada número à sua respectiva variável. Agora observe a instrução a seguir:

```
MEDIA := (NOTA1 + NOTA2 + NOTA3 + NOTA4) / 4;
```

O que está acontecendo na linha acima? Podemos observar que um cálculo está sendo realizado. Ele está somando as quatro médias e está dividindo por quatro. Dá pra perceber que ele está armazenando o resultado desse cálculo na variável MEDIA. É aí que entra o símbolo ":=" (dois pontos e igual juntos). Esse é um operador de atribuição. Ele serve para atribuir o que está à sua direita para a variável que está à sua esquerda.

É bom frisar que este não é um operador normalmente utilizado em algoritmos. Esse é o operador de atribuição do Delphi. Normalmente, quando criamos algoritmos em pseudocódigo utilizamos o símbolo ← (seta para esquerda) como operador de atribuição.

E qual o operador de atribuição do Java, do C#, do JavaScript etc? Você vai aprender sobre o operador de atribuição quando você for utilizar cada uma das linguagens na prática.

---

int x = 10; int y = 10; int z = x+y; printf("%d",z);

Qual é o nome do filme?

O código dá 20.

Constantes

Existe um outro algoritmo que apresentamos no módulo anterior que lê o raio de uma circunferência e calcula sua área.

```
ALGORITMO AREA_CIRCUNFERENCIA;
CONST
   PI = 3.1416;
VAR
   RAIO, AREA: REAL;
INICIO
   LER (RAIO); // PROCESSAMENTO
   AREA := PI * SQR(RAIO); // ENTRADA
   ESCREVA "AREA=" + AREA); // SAÍDA
FIM.
```

Note que nesse algoritmo temos um outro bloco que começa com **CONST**. Esse bloco indica que vamos colocar ali algumas constantes. Constantes? Sim, isso mesmo. E o que é uma constante? O nome é bem sugestivo não é mesmo? Se a variável varia, a constante se mantém … constante! Uau!

É isso aí. A **constante** é um espaço reservado na memória para armazenar um valor que não muda com o tempo. No nosso algoritmo AREA_CINCUNFERENCIA temos a contante PI cujo valor é 3.1416. Nós usamos essa constante para realizar o cálculo da área. A constante PI nunca terá seu valor alterado. Assim sendo, tenha bem em mente os seguintes conceitos relacionados a constantes:

- Em programação, uma constante armazena um valor fixo, que **NÃO** mudará com o tempo de execução do programa. Ou seja, o valor será definido uma única vez e jamais será alterado durante a execução da aplicação.
- Uma constante deve ser utilizada quando uma informação **NÃO** tem qualquer possibilidade de alteração, ou variação, no decorrer da execução do algoritmo.

Entrada e Saída de Dados

Você observou que estamos usando os termos LEIA e ESCREVA nos nossos algoritmos. Normalmente um programa solicita dados do usuário e exibe dados para o mesmo. Essas são operações de entrada (quando solicitamos os dados) e de saída (quando exibimos dados para o usuário).

- A instrução LEIA é utilizada quando se deseja obter informações do usuário por meio do teclado, ou seja, é um **comando de entrada de dados.**

- A instrução ESCREVA é utilizada para mostrar informações na tela do computador, ou seja, é um **comando de saída de dados**.

Comentários

Note que no algoritmo que calcula a área da circunferência existem algumas palavras que aparecem depois de duas barras (//).

```
LER (RAIO); // PROCESSAMENTO
AREA := PI * SQR(RAIO); // ENTRADA
ESCREVA "AREA=" + AREA); // SAÍDA
```

Tudo que aparece depois de duas barras é considerado como um comentário. A inserção de comentários no decorrer do algoritmo facilita a leitura deste por outros programadores. Os comentários também servem para auxiliar o programador a relembrar o próprio código depois de um tempo sem utilizá-lo. Existem porém um conceito de que, se precisou comentar é porque o código não está bem escrito!

Operadores

Nós falamos anteriormente sobre o operador de atribuição, onde pegamos um valor qualquer e atribuímos para uma variável ou constante. Mas ele não é o único operador que existe. Vamos descobrir agora quais outros operadores podemos utilizar nos nossos algoritmos.

Antes de continuarmos, porém, vamos definir o que é um operador. Um operador serve para relacionar valores de tal maneira que resulte em outro valor. Quando você resolve uma expressão matemática, você está usando operadores. Quais são eles? São os operadores de somar, subtrair, multiplicar, dividir etc. Na programação nós temos esses mesmo operadores e temos ainda outros. Vamos tratar aqui de quatro tipos de operadores:

1. Operadores Aritméticos.
2. Operadores Relacionais.
3. Operadores Lógicos.
4. Operadores de Atribuição.

Operadores Aritméticos

Em programação, esses operadores são tão simples quanto os que você aprendeu na escola. Apenas existem alguns que você pode não conhecer. É o conjunto de símbolos que representa as operações básicas da matemática como: somar, subtrair, multiplicar, dividir etc. Esses operadores somente poderão ser utilizados entre variáveis com os tipos de dados numéricos.

Operador	Operação
+	Adição
-	Subtração
*	Multiplicação
/	Divisão
% | MOD	Módulo (resto da divisão)

Esses operadores obedecem às regras matemáticas comuns:

- As expressões dentro dos parênteses são sempre resolvidas antes das expressões fora dos parênteses.
- Quando existe um parêntese dentro de outro, a solução sempre inicia do parêntese mais interno para o mais externo (de dentro para fora).
- Quando duas ou mais expressões tiverem a mesma prioridade, a solução é sempre iniciada da expressão mais à esquerda para a mais à direita.

Quais operadores aritméticos estamos utilizando no algoritmo MEDIA_FINAL?

```
ALGORITMO MEDIA_FINAL;
VAR
   NOTA1, NOTA2, NOTA3, NOTA4, MEDIA: REAL;
   NOME: CARACTERE;
INICIO
   LEIA (NOME);
   LEIA (NOTA1, NOTA2, NOTA3, NOTA4);
   MEDIA := (NOTA1 + NOTA2 + NOTA3 + NOTA4) / 4;
   ESCREVA (NOME, MEDIA);
FIM.
```

Operadores Relacionais ou de Comparação

Operadores relacionais são utilizados para comparar valores (de qualquer tipo), o resultado de uma expressão relacional é um valor booleano (VERDADEIRO ou FALSO).

Operador	Operação
<	Menor que
>	Maior que
<=	Menor ou igual a
>=	Maior ou igual a
==	Igual
!=	Diferente

Vamos alterar o algoritmo MEDIA_FINAL para utilizarmos alguns operadores relacionais. Observe.

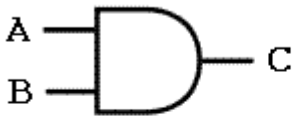
```
ALGORITMO MEDIA_FINAL;
VAR
   NOTA1, NOTA2, NOTA3, NOTA4, MEDIA: REAL;
   NOME: CARACTERE;
INICIO
   LEIA (NOME);
   LEIA (NOTA1, NOTA2, NOTA3, NOTA4);
   MEDIA := (NOTA1 + NOTA2 + NOTA3 + NOTA4) / 4;
   SE (MEDIA> 8)
      ESCREVA (“PARABENS, VOCE FOI APROVADO”);
   SENAO SE (MEDIA >=6 E MEDIA < 8)
      ESCREVA (“VOCE ESTA DE RECUPERACAO”);
   SENAO SE (MEDIA < 6)
      ESCREVA (“VOCE FOI REPROVADO”);
   FIM SE
FIM.
```

Operadores Lógicos

Os operadores lógicos são usados para combinar duas ou mais expressões e, a partir daí, tomar uma decisão. Talvez você já tenha estudado sobre portas lógicas, que são responsáveis pelo surgimento de diversas novas tecnologias, como os circuitos lógicos e os sistemas digitais. Foi através delas que muitos dos aparelhos eletrônicos de nosso cotidiano puderam ser criados. Portas lógicas são dispositivos que trabalham com um ou mais sinais de entrada e os transformam em um único sinal de saída. São elas: AND, NOT, NAND, OR, NOR, XOR, XNOR.

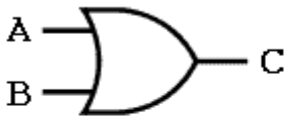
Para o nosso estudo vamos nos concentrar nas portas lógicas AND, NOT e OR. Ou seja, vamos pegar dois valores de entrada e observar o que acontece na saída para cada opção desejada. Veja a imagem abaixo.

AND



Inputs		Output
A	B	C
0	0	0
0	1	0
1	0	0
1	1	1

OR



Inputs		Output
A	B	C
0	0	0
0	1	1
1	0	1
1	1	1

NOT

Input	Output
A	C
0	1
1	0

O valor ZERO (0) representa o FALSE e o valor UM (1) representa o TRUE. Assim sendo, numa operação AND nós só teremos um retorno verdadeiro caso as duas entradas sejam verdadeiras.

Observe o seguinte trecho de código:

```
SE (MEDIA >=6 E MEDIA < 8)
   ESCREVA ("VOCE ESTA DE RECUPERACAO");
```

O aluno só estará de recuperação se sua média for maior ou igual a 6 **E** menor que 8 **AO MESMO TEMPO**. Vamos criar um teste de mesa para ver o que acontece.

Média	Maior ou igual a 6	Menor que 8	Resultado
4	FALSE [0]	TRUE [1]	FALSE [0]
9	TRUE [1]	FALSE [0]	FALSE [0]
7	TRUE [1]	TRUE [1]	TRUE [1]

Fica claro no teste acima que o resultado só será verdadeiro se as duas instruções forem verdadeiras, como deve ocorrer com o operador lógico AND.

Já o operador OR é diferente. O resultado só será falso caso as duas entradas sejam falsas. Se pelo menos uma delas for verdadeira, então o resultado será verdadeiro. Vamos alterar um pouco o nosso código para observar o que acontece.

```
SE (MEDIA >=6 OU MEDIA < 8)
   ESCREVA ("VOCE ESTA DE RECUPERACAO");
```

Estamos imaginando agora que o aluno ficará de recuperação caso sua média seja maior/igual a seis **OU** menor que oito. Dessa vez não precisamos ter as duas situações ocorrendo ao mesmo tempo, basta que uma delas seja verdadeira. Vamos ao teste de mesa.

Média	Maior ou igual a 6	Menor que 8	Resultado
4	FALSE [0]	TRUE [1]	TRUE [1]
9	TRUE [1]	FALSE [0]	TRUE [1]
7	TRUE [1]	TRUE [1]	TRUE [1]

Veja que, com essa alteração, esse trecho do programa perde o sentido, pois ele sempre retorna verdadeiro. O programa vai rodar normalmente, mas o resultado não é o esperado: independente da média, o aluno sempre estará de recuperação. Temos aí o que chamamos de **Erro de Lógica**. Outros diriam que, neste caso, o problema está entre a cadeira e o teclado.

Finalmente temos o NOT, que nada mais é que a negação da expressão fornecida. Tenha muito cuidado ao usar o NOT no seu código, pois ele pode ficar confuso muito rapidamente. Quer ver?

```
SE (NAO(MEDIA >=6 E MEDIA < 8))
   ESCREVA ("VOCE ESTA DE RECUPERACAO");
```

Quando colocamos ali o NAO na frente do parêntese onde está o teste relacionado à média, significa que estamos negando o que está acontecendo dentro do parêntese. Ou seja, o algoritmo só vai escrever que o aluno está de recuperação se houver a negação de:

1. A média é maior ou igual a 6 **E**
2. A média é menor que 8

Sendo assim, ele vai fazer o exato oposto do que gostaríamos que ele fizesse no início. Todo e qualquer número que não esteja no intervalo fará com que o aluno fique de recuperação. Veja o teste de mesa.

Média	Maior ou igual a 6	Menor que 8	Resultado	(nao) Resultado
4	FALSE [0]	TRUE [1]	FALSE [0]	TRUE [1]
9	TRUE [1]	FALSE [0]	FALSE [0]	TRUE [1]
7	TRUE [1]	TRUE [1]	TRUE [1]	FALSE [0]

Achou confuso? Sim, é confuso. Por isso mesmo que esse operador deve ser usado com muito cuidado, apenas se necessário e se o código ficar legível (claro).

```
/* Querido programador:
Quando eu escrevi este código,
apenas Deus e eu sabíamos como
ele funcionava.
Agora, apenas Deus sabe!

Portanto, se você estiver
tentando melhorar esta rotina porque
está falhando e "com certeza está",
por favor, aumente este contador
como um aviso para a próxima
pessoa

total_horas_gastas_aqui: 254
*/
```

Operadores de Atribuição

Já falamos algumas vezes sobre eles né. Veja quais as características dos operadores de atribuição.

- Têm como função retornar um valor atribuído de acordo com a operação indicada.
- A operação é feita entre os dois operandos, sendo atribuído o resultado ao primeiro.

Nós vimos anteriormente o operador de atribuição simples. Mas existem outros. Observe.

Operadores de Atribuição		
Atribuição simples **:= ou = ou ←**	**Atribuição com subtração** **-=**	**Atribuição com divisão** **/=**
Atribuição com adição **+=**	**Atribuição com multiplicação** ***=**	**Atribuição com módulo** **%=**