**Linguagem de Programação Python**

**Capítulo 1 – Tópico 3: Pseudocódigo**

**Pseudocódigo ou Português Estruturado**

Pseudocódigo é uma técnica natural, comum e originário a quem o codifica por meio de uma linguagem natural, no nosso caso, a língua portuguesa. Permite a compreensão por parte de qualquer pessoa e NÃO exige que nenhum padrão de sintaxe seja seguido na linguagem da programação.

O pseudocódigo usa termos da língua portuguesa encadeados e organizados de forma parecida com as diferentes linguagens de programação. Seu objetivo é simplificar o processo de programação.

Para entender um algoritmo escrito em pseudocódigo, você não precisa ter o conhecimento prévio de nenhuma linguagem de programação específica. Nesse tipo de algoritmo, as estruturas lógicas são escritas em uma linguagem próxima ao português e na sequência em que devem ser executadas

Texto

Descrição gerada automaticamente com confiança média

Ao construir um projeto de programa (algoritmo) em pseudocódigo, você representará o encadeamento lógico necessário para resolver um problema da forma mais próxima que existe de uma linguagem de programação. A opção pelo pseudocódigo na etapa de estruturação do algoritmo proporciona um ganho de tempo na etapa de desenvolvimento do código.

Agora, vejamos como utilizar o português estruturado na prática! Para isso, vamos voltar ao algoritmo de conversão de real em dólar.

Nesse caso, primeiro relembraremos o algoritmo usando a descrição narrativa e, em seguida, veremos como fica o algoritmo usando o pseudocódigo.

Descrição Narrativa Pseudocódigo

Texto

Descrição gerada automaticamente com confiança média

Tal como você pôde perceber, o pseudocódigo é uma técnica que organiza os algoritmos em blocos formados por comandos escritos.

Para isso, são usados, por exemplo, os comandos Leia e Escreva para inserção (captura) e exibição (apresentação) de dados, respectivamente.

Além dos comandos, o pseudocódigo permite a declaração de variáveis e expressões aritméticas para realizar contas.

Texto

Descrição gerada automaticamente

Você notou como o pseudocódigo é uma forma bastante detalhada de escrever sequências lógicas de instruções e processamentos? Nesse sentido, a principal característica desse método é ser muito semelhante ao jeito como as linguagens de programação são codificadas. E foi justamente o fato de ser muito parecido com os códigos de programação que fez com que surgisse o nome pseudocódigo, ou seja, “falso” (fictício) código.

É importante que você saiba que o pseudocódigo tem grande aceitação entre os programadores. Mas por que será que isso ocorre? O principal motivo está no fato de o pseudocódigo possibilitar uma tradução praticamente direta para qualquer linguagem de programação específica. Justamente por isso, iremos apresentar a você vários exemplos de algoritmos em pseudocódigo daqui em diante.

Interface gráfica do usuário, Texto

Descrição gerada automaticamente

Todo algoritmo começa com a definição de um nome de identificação para ele, na primeira linha: <nome\_do\_algoritmo>

Na linha de baixo, deve ser feita a declaração de variáveis, cuja palavra reservada é Var. O termo Var é uma redução da palavra "variável". Nesse sentido, o termo Var é um marcador, assim como as palavras INÍCIO, FIM, SE, SENÃO, ENTÃO, FAÇA CASO etc. Um marcador ou uma palavra reservada são termos interpretados de forma diferenciada pelos programas. Por isso, não devem ter outros usos, como nomear variáveis.

**Tipo de Dado**

Programar computadores requer a definição e o uso de espaços em memória para receber informações inseridas pelos usuários. Os espaços em memória recebem informações que podem variar no decorrer de um processamento. Por isso, é muito importante aprender a criar e definir o tipo, bem como manipular variáveis e constantes.

Interface gráfica do usuário, Texto

Descrição gerada automaticamente

Ao criar um espaço em memória, com conteúdo variável ou constante, precisamos definir e registrar o tipo de dado que será inserido e manipulado.

* *Literais*

*Texto

Descrição gerada automaticamente*

* *Numéricos*

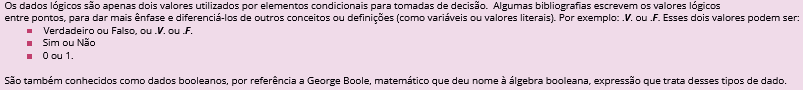
* *

*Texto

Descrição gerada automaticamenteTexto

Descrição gerada automaticamente*

* *Lógicos*

**

**Uso e Funções de Constantes e Variáveis**

Chamamos de constante todo valor fixo que não será alterado durante a execução de um programa.

Ícone

Descrição gerada automaticamente

*Vejamos um exemplo de uso de CONSTANTE:*

**

Observe que, na fórmula para calcular o IMC (Índice de Massa Corporal), o número dois é um valor constante, pois não muda. Desse modo, o número dois é diferente dos demais elementos (IMC, peso e altura), que vão variar em cada caso. Em relação ao tipo do dado, o número dois deve ser classificado como uma constante do tipo numérica inteira.

*Agora, vamos entender melhor as VARIÁVEIS.*

Uma variável representa um endereço da memória RAM que armazena, temporariamente, valores e informações. Esses endereços recebem um nome e um conteúdo. Cada vez que mencionamos o nome da variável, seu conteúdo é manipulado.

Vamos retomar o exemplo da fórmula para calcular o IMC (Índice de Massa Corporal)? Observe a seguir:

Diagrama

Descrição gerada automaticamente

É importante sabermos que as variáveis podem armazenar todos os tipos de dados. Além disso, para criar variáveis ou constantes, precisamos definir um nome simbólico, uma identificação a partir da qual elas serão chamadas, o tipo de dado que elas receberão e um valor inicial.

A criação de variáveis ocorre no momento de codificação do programa. Desse modo, precisamos atribuir um nome simbólico a determinado endereço da memória RAM. No decorrer do programa, esse nome vai ser utilizado para manipular a informação contida no endereço da memória relacionado à variável.

Para nomear uma variável, precisamos seguir algumas regras.

Os nomes das variáveis:

* Devem sempre começar por uma letra.
* Não devem ter espaços em branco.
* Não devem ter caracteres especiais – no máximo, o símbolo sublinhado.

Por exemplo:

* Nome\_cliente
* Num\_funcionario
* Datanascimento

Além disso, precisamos saber que as variáveis devem ser declaradas sempre no início dos algoritmos e dos programas.

Os valores armazenados nas variáveis permanecem:

* Até que o computador seja desligado pelo usuário ou por falta de energia, já que são armazenados na memória RAM.
* Por término do programa ou da rotina onde foram criados.
* Até que seja atribuído um novo valor para a mesma variável.

*Vejamos um exemplo:*

Se escrevermos A = 5, o valor cinco será o conteúdo da variável A.

Se, mais adiante, escrevermos A = 20, o valor vinte passará a ser o novo conteúdo da variável A.

Nesse caso, o valor anterior 5 (cinco) será eliminado (substituído).

**Teste de Mesa**

Uma vez que escolhemos a forma de representar o algoritmo e concluímos sua estrutura, como saber se o encadeamento lógico está correto?

Ao desenvolver um algoritmo, devemos submeter a rotina elaborada a um processo de verificação. O objetivo da verificação é simular o funcionamento e a eficiência do código ao resolver o problema proposto. Em outras palavras, a ideia é testar a sequência lógica para ver se o encadeamento das instruções e dos processamentos está correto ou para ver se há necessidade de ajustes. No jargão da Informática, esse tipo de teste é chamado de teste de mesa. Na prática, esse termo significa seguir as instruções dos algoritmos, de maneira precisa, para verificar se o procedimento utilizado está correto ou não.

Para aplicar a técnica denominada teste de mesa, precisaremos de papel, caneta e paciência, para ler e realizar cada instrução encadeada como se fôssemos um computador.

Podemos aplicar o teste de mesa a qualquer tipo de algoritmo, seja uma descrição narrativa, um fluxograma ou um pseudocódigo.

Para demonstrar o teste de mesa, partiremos do algoritmo Quatro\_operacoes\_basicas, representado no pseudocódigo a seguir. Esse algoritmo foi construído com o objetivo de solicitar ao usuário a inserção de dois números quaisquer para a realização das quatro operações matemáticas básicas. Para testar algoritmos como esse, que conta com a inserção de dados, processamentos e exibição de resultados, o primeiro passo é criar uma tabela, colocando, em cada coluna, o nome das variáveis declaradas.

Texto

Descrição gerada automaticamenteTabela

Descrição gerada automaticamente

Observe que tabela prevê a realização de três testes. No entanto, esse número pode variar de acordo com o contexto e a complexidade do algoritmo. Depois de montada a tabela, é preciso ler atentamente cada instrução a fim de verificar o encadeamento das ações, a clareza das instruções e a escrita dos processamentos.

Lembre-se de que a proposta do teste de mesa é simular a realização da rotina do algoritmo por meio do preenchimento da tabela.

1º TESTE

Considere os valores 30 e 60 como os dois números (NUM\_A e NUM\_B, respectivamente) solicitados pelo algoritmo. A partir desses dois valores, siga o encadeamento de ações proposto nas linhas do pseudocódigo.

Quando o algoritmo solicitar a ação “Escreva”, preencha o campo correspondente na tabela de caneta azul. Para marcar as ações Leia, coloque o valor lido entre parênteses de caneta vermelha. Essas regras simples são importantes para registrar a verificação de todas as ações previstas no algoritmo.

2º TESTE

Repita o procedimento realizado no primeiro teste, considerando os valores -20 e 40 como NUM\_A e NUM\_B, respectivamente.

3º TESTE

Repita o procedimento, considerando, agora, os valores 20 e -40.

Calendário

Descrição gerada automaticamente

**Utilização de Pseudocódigos**

Como você deve ter percebido, a criação de algoritmos em pseudocódigo é relativamente simples, não é mesmo?

O pseudocódigo é um método que não usa estruturas complexas e rígidas – como as sintaxes de linguagens de programação –, nem exige ambientes especiais para edição. Em outras palavras, o pseudocódigo pode ser encarado com uma versão similar e simplificada dos códigos de programação, com o benefício de poder ser escrito em qualquer editor de texto. Isso significa que podemos escrever um algoritmo em pseudocódigo usando o Bloco de Notas, por exemplo. Essa característica faz do pseudocódigo uma ferramenta útil e simples para elaboração de projetos de programas.

Uma das principais vantagens desse método é que ele pode ser entendido por “não programadores”. Com isso, os desenvolvedores podem apresentar suas ideias e validar sequências instrucionais com profissionais que não são da área de programação.

*Vamos estipular algumas regras de padronização para a construção de um pseudocódigo*

1. Todo algoritmo em pseudocódigo deve ser iniciado com Algoritmo: NomeDoAlgoritmo.

2. O início e fim do programa são limitados pelos marcadores Início e Fim.

3. As variáveis são declaradas no início do algoritmo, abaixo do marcador Var, da seguinte forma: NomeDaVariável: Tipo da variável.

4. Os nomes das variáveis NÃO podem:

* Iniciar por número (erro: 1nome)
* Ter espaço (erro: nome completo)
* Ter caracteres especiais (´,`, ~,ç ,- e outros).

5. As palavras-reservadas devem ser evitadas: Início, Fim, Var, Se e Senão.

6. Os nomes das variáveis são case sensitive. Dessa forma, ao manipularmos variáveis, devemos usar o mesmo nome declarado no início, considerando o uso de letras maiúsculas e minúsculas.

7. O comando Leia deve ser usado para receber (capturar) dados do usuário, fase do processamento conhecida como “Entrada de Dados”.

8. O comando Escreva deve ser usado para exibir (apresentar, mostrar) dados ao usuário, fase do processamento conhecida como “Saída de Dados”.

9. Os textos a serem exibidos na tela ou que devam ser inseridos como caractere são colocados entre "aspas“ (representação universal de um valor literal ou string – “Sistemas”).

10. Os comentários sobre o código podem ser inseridos {entre chaves} (incomum por confundir com agrupamentos) ou inseridos utilizando // (mais comum) no início da linha de instruções. O comentário não altera a execução do código. Contudo, ele é de fundamental importância para documentar e tornar inteligíveis as escolhas de programação realizadas no código para outros programadores.

*As regras são muito importantes, já que todos os elementos do pseudocódigo – e dos códigos – são fundamentais para o funcionamento correto do programa. Desse modo, entender o rigor da padronização e aplicá-la é essencial a quem deseja atuar com programação de computadores.*

**Operadores: Tipos e Usos**

Podemos afirmar que programar computadores é lidar com informações e instruções para manipular dados e chegar a resultados. Nesse sentido, os programas podem realizar cálculos simples e complexos, comparar dados inseridos pelos usuários e dar diferentes encaminhamentos a esses dados, dependendo do processamento.

O que o computador fará depende do uso dos chamados operadores.

Tela de celular com texto preto sobre fundo branco

Descrição gerada automaticamente

*Existem três tipos diferentes de operadores: aritméticos, relacionais e lógicos.*

Tabela

Descrição gerada automaticamente

Tabela

Descrição gerada automaticamente



Tabela

Descrição gerada automaticamente



**Atribuição de Valores às Variáveis**

Ao longo da rotina estabelecida em um algoritmo ou programa, as variáveis devem receber valores de acordo com o tipo de dado que foi declarado inicialmente. Nesse sentido, se declararmos uma variável como dado literal (caractere), não poderemos fazer cálculos com ela.

Interface gráfica do usuário, Texto, Aplicativo

Descrição gerada automaticamente

No exemplo, associamos o valor digitado pelo usuário às variáveis.

Perceba que podemos utilizar o sinal de igual “=”, que não é considerado uma igualdade na programação, mas sim um sinal de atribuição, para associar valores, já que esse operador tem a função de associar (atribuir) um valor a um identificador, um operando, isto é, uma variável.

*Para cada variável criada no desenvolvimento de um algoritmo, precisamos pensar em três aspectos: nome, tipo e conteúdo.*

*Tela de celular com texto preto sobre fundo branco

Descrição gerada automaticamente com confiança média*

*Tela de celular com texto preto sobre fundo branco

Descrição gerada automaticamente com confiança média*

*Interface gráfica do usuário, Texto

Descrição gerada automaticamente*

As três definições devem ter coerência entre si para que o programa funcione corretamente e para que seu trabalho como desenvolvedor seja mais fácil.

Cabe lembrar que as etapas de estruturação do algoritmo e de teste de mesa existem para que ajustes de definição de variáveis (nome e tipo), de processamento e de encadeamentos lógicos possam ser realizados.

O planejamento de uma solução computadorizada e o teste em situação real são fundamentais para que o algoritmo funcione como um verdadeiro projeto do programa. Tais etapas servem para apurar o processo, sendo fundamentais para que a codificação de programas aconteça sem perda de tempo ou retrabalhos.