N_ALG PROG_A1 - Texto de apoio

Site: EAD Mackenzie
Tema: ALGORITMOS E PROGRAMAÇÃO I {TURMA 01D} 2023/1
Livro: N_ALG PROG_A1 - Texto de apoio

Impresso por: CAIO FRESSATTI PINHEIRO .
Data: quinta, 2 fev 2023, 14:09

Índice

1. FUNDAMENTOS DE ALGORITMOS E PROGRAMAÇÃO

Algoritmos e Programação

O termo **algoritmo** pode ser visto desde o **século IX**. Foi nesta época que um cientista, astrônomo e matemático persa usou pela primeira vez o termo para indicar **regras** de operações aritméticas utilizando algarismos indoarábicos.

No século XII, Adelardo de Bath traduziu o termo para o latim **Algorithmi**. De lá para cá, o termo evoluiu bastante e passou a incluir todos os **procedimentos definidos para resolver problemas** ou **realizar tarefas**.

A formalização da noção de algoritmo ocorreu em 1936 com os trabalhos de **Alan Turing** e **Alonzo Church**, que desenvolveram, independentemente, os modelos de **Máquinas de Turing** e **Cálculo Lambda**.

Do ponto de vista computacional, um **algoritmo** pode ser visto como um **conjunto de regras e procedimentos lógicos perfeitamente definidos que levam à solução de um problema em um número finito de passos** .

Donald Knuth, um dos pesquisadores mais respeitados em algoritmos, indica uma lista de cinco propriedades que são **requisitos para algoritmos**:

- **Finitude**: um algoritmo deve sempre terminar após um número finito de etapas (ou passos).
- **Definição**: cada passo de um algoritmo deve ser definido com precisão. As ações a serem executadas deverão ser especificadas rigorosamente e sem ambiguidades.
- **Entrada**: valores que são dados ao algoritmo antes que ele inicie.
- **Saída**: os valores resultantes das ações do algoritmo a partir de uma determinada entrada.
- **Eficácia**: todas as operações a serem realizadas pelo algoritmo devem ser suficientemente básicas para poderem, em princípio, ser feitas com precisão e em um período de tempo finito por um homem usando papel e lápis.

Figura 1 – Elementos de um programa

Fonte: E laborado pela autora.

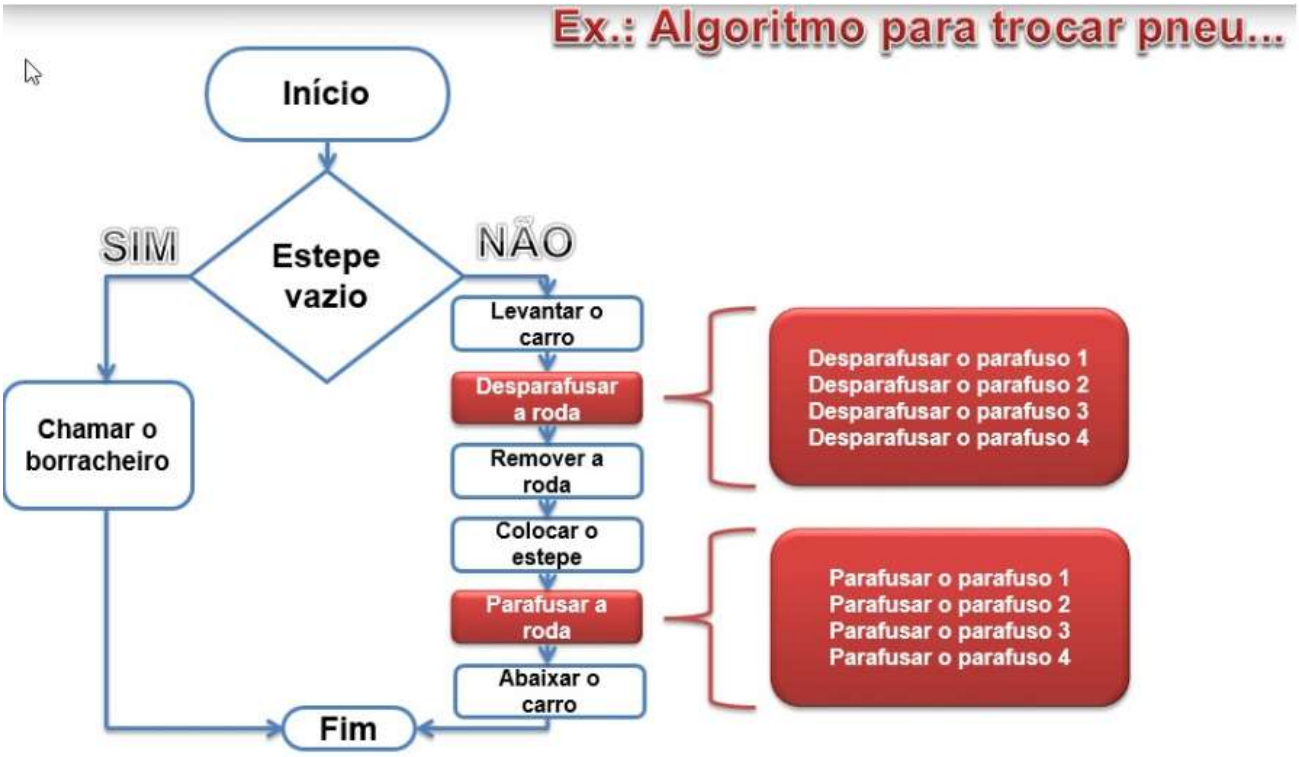
Embora os requisitos de Knuth sejam intuitivos, falta-lhes **rigor formal**. A formalidade pode ser conseguida com o uso de **lógica**. Assim, exigiremos que um algoritmo seja **uma sequência lógica de passos com começo, meio e fim**.

Comumente, es sa lógica é conhecida como **lógica de programação**, e isso ocupará grande parte de nossa disciplina.

De um ponto de vista mais geral, podemos dizer que **Lógica** é uma parte da **Filosofia** que **trata das formas do pensamento em geral (dedução, indução, hipótese, inferência, dentre outros) e das operações intelectuais que visam à determinação do que é verdadeiro ou falso**. Dentre outras coisas, a Lógica pode produzir algoritmos para es sas formas e operações.

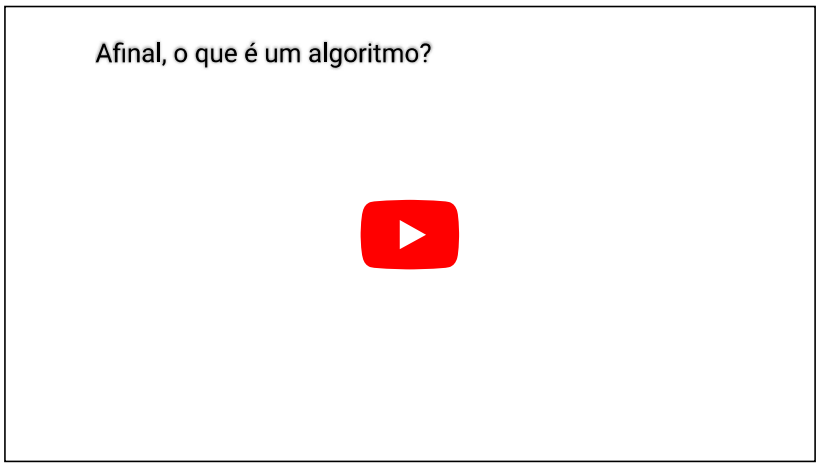
ATENÇÃO: A fim de resolver um problema computacionalmente, duas coisas são necessárias: uma representação que capta todos os aspectos relevantes do problema, e um algoritmo que resolve o problema pelo uso da representação.

Figura 2 – Algoritmo para troca de pneu



Fonte: http://mecatronizar.blogspot.com/2016/08/fluxogramas.html

Para você entender um pouco mais sobre o que estamos falando, assista ao vídeo “Afinal, o que é um algoritmo?”.



2. FORMAS DE REPRESENTAÇÃO DA RESOLUÇÃO DE UM PROBLEMA

A) Fluxograma

Para fazer um fluxograma, sugerimos o uso do **RAPTOR**.

RAPTOR é um programa gratuito desenvolvido para o sistema operacional Windows, incluindo o Windows XP. Projetados para ajudar os alunos a visualizar seus algoritmos, o *RAPTOR – Flowchart Interpreter* fornece suporte visual e ajuda os novos programadores a rastrear a execução por meio do uso de fluxogramas. Esses fluxogramas são provados para ajudar o aluno a criar melhores algoritmos do que usando linguagem tradicional ou fluxogramas escritos sozinhos. Para novos alunos ou para aqueles que ainda estão desenvolvendo os conceitos básicos de programação, o RAPTOR é uma das ferramentas mais úteis disponíveis.

Para a instalação do Raptor, use o link a seguir: https://raptorflowchart.softonic.com.br/

Sobre fluxograma, saiba mais em: https://www.citisystems.com.br/fluxograma/

B) Pseudocódigo

A linguagem que o *VisuAlg* interpreta é bem simples: é uma versão portuguesa dos pseudocódigos largamente utilizados nos livros de introdução à programação, conhecida como "*Portugol*". O *Visualg* possui uma sintaxe muito simples e "liberal", para que o usuário se preocupe mais com a lógica da resolução dos problemas e não com as palavras-chave, pontos e vírgulas etc. No entanto, possui alguma formalidade, para criar um sentido de disciplina na elaboração do "código-fonte".

Para instalação do Visualg, acesse o site: http://www.apoioinformatica.inf.br/produtos/visualg

C) Linguagem de Programação: Python

A linguagem Python foi escolhida por ser uma linguagem muito versátil, usada não só no desenvolvimento Web, mas também em muitos outros tipos de aplicações. Embora simples, é uma linguagem poderosa, podendo ser usada para administrar sistemas e desenvolver grandes projetos. É uma linguagem clara e objetiva, pois vai direto ao ponto, sem rodeios.

Na página oficial do Python no Brasil, disponível em: https://python.org.br/empresas/, é possível conhecer algumas empresas que utilizam a linguagem. Então o leitor deste material estará aprendendo a programar em uma linguagem que poderá utilizar na prática e não ficará apenas na teoria.

Curiosidade: O nome Python é uma homenagem ao grupo humorístico inglês Monty Python, criador e intérprete da série cômica Monty Python's Flying Circus (sucesso na década de 1970).

Apesar de sua sintaxe simples e clara, Python oferece muitos recursos disponíveis também em linguagens mais complexas como Java e C++, como: programação orientada a objetos, recursos avançados de manipulação de textos, listas e outras estruturas de dados, possibilidade de executar o mesmo programa sem modificações em várias plataformas de hardware e sistemas operacionais.

Python é um software livre, ou seja, é gratuito, graças ao trabalho da Python Foundation (http://www.python.org/) e de inúmeros colaboradores. Python pode ser utilizada em praticamente qualquer arquitetura de computadores ou sistema operacional, como Linux, Microsoft Windows ou Mac OS X.

Um programa em Python recebe a extensão .py ao ser gravado no IDLE (*Integrated Development Environment*) do Python, que oferece as ferramentas necessárias para a edição, tradução, execução e depuração de seus programas.

3. CARACTERÍSTICAS DO PYTHON

Veja agora as razões que tornam a linguagem Python uma grande linguagem de programação:

- Python tem código aberto e pode ser usada e distribuída gratuitamente. Ainda, conta com uma grande comunidade de desenvolvimento que sempre está aperfeiçoando a linguagem.
- Python suporta programação procedural, funcional e orientada a objetos.
- Python prima pela legibilidade, é fácil de entender, mesmo que você não a conheça.
- Python gera um código fonte menor que outras linguagens como Java e C. Assim, a produtividade do programador aumenta bastante.
- Python é portátil. Isso significa que pode ser executado sem modificações em diversas plataformas.
- Python oferece uma grande variedade de bibliotecas que incorporam importantes funcionalidades, como o desenvolvimento de jogos.
- Python pode se comunicar com outras aplicações e linguagens como C e C++.

Saiba mais: https://en.wikipedia.org/wiki/Python_(programming_language)

4. VARIÁVEIS E TIPOS DE DADOS

Muitas vezes, em nossos programas, precisamos reservar espaço na memória para guardar valores que o programa acessará e modificará. A este espaço de memória, devidamente rotulado por um nome (identificador), chamamos **variável**.

Quando o espaço de memória possui um valor que não será modificado durante o programa, o chamamos de **constante**.

Quando queremos guardar um dado na memória, devemos classificá-lo quanto ao seu tipo. Pense nos conjuntos matemáticos, tais como inteiros, reais etc.

Variáveis e Identificadores

- Uma **variável** tem um nome (identificador) que está associado a um espaço em memória que armazena um valor. Uma variável pode receber diferentes valores durante a execução de um programa, por isso o nome "variável".

Exemplo:

```
>>> num = 1
>>> num
1
>>> num = 2
>>> num
2
```

num 1 num 2

```
>>> mensagem = "Alo, mundo!"
>>> mensagem
'Alo, mundo!'
```

Valores são atribuídos às variáveis usando o operador de atribuição (=).

Um **nome** ou **identificador** de uma variável é formado por uma sequência de um ou mais caracteres. Regras:

- Pode conter apenas letras, dígitos e *underscores* (sublinhas).
- Pode começar por uma letra ou *underscore*.
- Não é permitido o uso de outros caracteres especiais (por exemplo, espaço).
- Não é permitido o uso de palavras reservadas da linguagem a ser utilizada.
- O identificador deve ser conciso, porém descritivo (idade é melhor que i, *tamanho_nome* é melhor que *tamanho_do_nome_da_pessoa*).

Python: A versão 3 da linguagem Python permite a utilização de acentos em nome de variáveis, pois, por padrão, os programas são interpretados utilizando-se um conjunto de caracteres chamado UTF-8 (https://pt.wikipedia.org/wiki/UTF-8), capaz de representar praticamente todas as letras dos alfabetos conhecidos.

Palavras reservadas em Python:

and	as	assert	break	class	continue	def
del	elif	else	except	finally	for	from
global	if	import	in	is	lambda	nonlocal
not	or	pass	raise	return	try	while
with	yield	false	none	true		

5. TIPOS DE DADOS

Saber qual é o tipo de dado mais adequado para ser armazenado em uma variável é muito importante para garantir a resolução de um problema. Python tem vários tipos de dados, mas os mais comuns são números inteiros, números de ponto flutuante, *strings* e lógico.

Há um conjunto predefinido de tipos de dados chamados de tipos embutidos. Vejamos alguns deles:

Números

Um dado numérico é composto por uma sequência de dígitos (0 a 9), um sinal opcional (+ ou –) e um possível ponto decimal (usa-se o ponto e não a vírgula para separar a parte inteira da fracionária). São classificados como **inteiro** ou de **ponto flutuante** (real).

Exemplos:

Corretos				**Incorretos**	
Inteiro	**Ponto flutuante**				
5	5.	5.0	0.125	5,03	
2500	2500.	2500.0	2500.125	2,500	2,500.125
+2500	+2500.	+2500.0	+2500.125	+2,500	+2,500.125
-2500	-2500.	-2500.0	-2500.125	-2,500	-2,500.125

Limite de variação e precisão na representação de números

Para números **inteiros**, Python utiliza um sistema de precisão **ilimitada**. O número de **ponto flutuante**, por sua vez, tem variação e precisão **limitada** – usa o formato padrão de dupla precisão – fornecendo uma variação de 10-308 a 10308 com 16 a 17 dígitos de precisão.

Para indicar uma variação tão grande de valores, números de ponto flutuante podem ser representados em notação científica:

9.0045602e+5 (9.0045602 x 10^5, 8 dígitos de precisão)

1.006249505236801e8 (1.006249505236801 x 10^8, 16 dígitos de precisão)

4.239e-16 (4.239 x 10^{-16}, 4 dígitos de precisão)

6. STRINGS

Uma **string** representa uma sequência de caracteres (letras, dígitos, símbolos especiais). Em Python, as *strings* podem ser delimitadas por um par de aspas simples (') ou duplas (").

Exemplos:

"olá"

'Universidade Presbiteriana Mackenzie'

'Rua da Consolação, 930'

7. REFERÊNCIAS

DIERBACH, C. *Introduction to Computer Science Using Python*: A Computational Problem Solving Focus. New York: Wiley, 2012.

MENEZES, N. N. C. *Introdução à Programação com Python*: algoritmos e lógica de programação para iniciantes. São Paulo: Novatec, 2014.