


| | | |
|---|---|--|
|  | <p>Centro Universitário Tiradentes Campus Imbiribeira Curso: Ciências da Computação; ID: 833; Disciplina: Laboratório de Programação; ID: F106011; Dia: Segundas-Feiras; Horário: 07:50 às 11:50; Professor: Vinícius Costa Amador</p> <p>Discente: _____, Data: __/__/__</p> | <p>Oficina: Tema da Oficina</p> |
|---|---|--|

2ª Up

Sumário

| | |
|--|-----------|
| Programação Estruturada – Teoria I..... | 2 |
| O que é? | 2 |
| Programação Estruturada – Teoria II..... | 3 |
| Como surgiu? | 3 |
| Programação Estruturada – Teoria III..... | 4 |
| Por que é importante? | 4 |
| Comparações Didáticas..... | 4 |
| Programação Estruturada – Teoria IV..... | 5 |
| Estruturas Básicas | 5 |
| Aplicações Reais | 5 |
| Perguntas | 6 |
| Sobre Conceitos..... | 6 |
| Sobre Aplicações | 6 |
| Sobre Analogias | 6 |
| Sistema de Cadastro de Produtos com Estoque Mínimo em Python I..... | 7 |
| Função para cadastrar produtos | 7 |
| Sistema de Cadastro de Produtos com Estoque Mínimo em Python II | 8 |
| Variáveis dentro da função | 8 |
| Sistema de Cadastro de Produtos com Estoque Mínimo em Python III | 9 |
| Tupla com categorias | 9, 10, 11 |
| Sistema de Cadastro de Produtos com Estoque Mínimo em Python IV | 12 |
| Função para mostrar produtos com estoque abaixo do mínimo | 12 |
| Sistema de Cadastro de Produtos com Estoque Mínimo em Python V | 13 |
| Lista e loop principal com while True e break..... | 13 |
| Sistema de Cadastro de Produtos com Estoque Mínimo em Python VI | 14 |
| Mostrar produtos com estoque baixo | 14 |
| Sistema de Cadastro de Produtos com Estoque Mínimo em Python VII | 15 |
| Código completo | 15 |
| Sistema de Cadastro de Produtos com Estoque Mínimo em Python VIII | 16 |
| Fluxograma..... | 16 |
| Referências | 17 |

Programação Estruturada – Teoria I

O que é?

A Programação Estruturada é um paradigma de programação baseado no uso de estruturas de controle bem definidas:

- Sequência
- Decisão (condicionais)
- Repetição (laços de repetição)

Ela prioriza a organização lógica do código, evitando o uso de comandos como GOTO, que tornam o código confuso e difícil de manter.

Programação Estruturada – Teoria II

Como surgiu?

A Programação Estruturada surgiu como uma resposta aos problemas de desenvolvimento de softwares desorganizados, comuns nas décadas de 1950 e 1960, quando era normal usar comandos de desvio como o GOTO, que geravam códigos difíceis de entender, chamados de "código espaguete".

O conceito principal da Programação Estruturada é que qualquer programa pode ser desenvolvido utilizando apenas três estruturas fundamentais:

- Sequência: execução linear das instruções.
- Decisão (condição): escolha entre dois ou mais caminhos (ex.: if, else).
- Repetição (laços): execução de blocos de código várias vezes (ex.: while, for).

A partir dessa ideia, os programas passaram a ser mais organizados, fáceis de ler, entender, modificar e manter. O uso de funções e procedimentos também se popularizou, promovendo a modularização, ou seja, a divisão do código em partes menores e mais gerenciáveis.

Esse paradigma foi tão importante que influenciou diretamente a criação de linguagens como Pascal, C e Ada, que foram largamente utilizadas no ensino e na indústria. Além disso, o conceito de estruturação foi a base para o desenvolvimento da Programação Orientada a Objetos (POO), que surgiu posteriormente, mas manteve as mesmas estruturas dentro dos métodos.

Atualmente, mesmo com a evolução para paradigmas como POO, funcional e programação orientada a eventos, a base da lógica estruturada continua presente em praticamente todas as linguagens modernas.

Programação Estruturada – Teoria III

Por que é importante?

- Clareza e organização do código.
- Facilita leitura, manutenção e testes.
- Estimula o pensamento lógico e sequencial.
- Base para paradigmas modernos (ex: programação orientada a objetos).
- Permite reaproveitamento de código com funções/módulos.

Comparações Didáticas

- Montar um móvel com manual: seguir uma sequência lógica, um passo depende do outro.
- Receita de bolo: entrada (ingredientes), processo (mistura), decisão (assar ou não), repetição (mexer até ficar homogêneo).
- Semáforo de trânsito: fluxo previsível com decisões e repetições.

Programação Estruturada – Teoria IV

Estruturas Básicas

1. Sequência: comandos executados em ordem.
2. Decisão (if, else): tomada de decisões com base em condições.
3. Repetição (for, while): execução de comandos enquanto uma condição for verdadeira.
4. Modularização: divisão do programa em funções para facilitar a reutilização e manutenção.

Aplicações Reais

- Calculadoras básicas (soma, subtração, etc.).
- Microcontroladores simples (Arduino, sensores).
- Scripts automatizados (backup, renomear arquivos).
- Softwares de cadastro (escolas, bibliotecas).
- Jogos simples (adivinhação, roleta).
- Menus de autenticação (login com senha).

Perguntas

Sobre Conceitos

- O que torna um código estruturado diferente de um bagunçado?
- Por que evitar o uso do GOTO é considerado uma boa prática?
- Qual a função da modularização em um programa estruturado?
- Como a programação estruturada ajuda na hora de corrigir erros?

Sobre Aplicações

- Vocês conseguem pensar em algum aplicativo simples ou ferramenta que deve ter sido feita com lógica estruturada?
- Se fossem montar um programa de calculadora, quais partes usariam sequência, decisão e repetição?

Sobre Analogias

- Se a programação estruturada fosse uma receita de bolo, o que seria o if? E o while?
- Por que montar um móvel com manual pode ser comparado com escrever um código estruturado?

Sistema de Cadastro de Produtos com Estoque Mínimo em Python I

Função para cadastrar produtos

def cadastrar_produto():

- Aqui declaramos uma função chamada **cadastrar_produto**.
- Uma função é um bloco de código que executa uma tarefa específica, que podemos chamar várias vezes para reaproveitar o código.
- **def** significa definição da função.
- Tudo que estiver indentado depois de **def** pertence a essa função.

Sistema de Cadastro de Produtos com Estoque Mínimo em Python II

Variáveis dentro da função

nome = input("Digite o nome do produto: ")

- **nome** é uma variável do tipo **string** que armazena o nome do produto.
- **input()** captura texto digitado pelo usuário.

preco = float(input("Digite o preço do produto: "))

- **preco** é uma variável do tipo **float (número decimal)** para guardar o preço.
- Convertendo o texto digitado (**input()**) para número decimal com **float()**.

quantidade = int(input("Digite a quantidade em estoque: "))

- **quantidade** é uma variável do tipo **inteiro (int)**, para guardar quantas unidades do produto estão no estoque.
- Convertendo o texto digitado para número inteiro com **int()**.

estoque_minimo = 5

- **estoque_minimo** é uma variável do tipo inteiro, fixa com valor 5.
- Ela define o limite mínimo que o estoque deve ter para o produto

Sistema de Cadastro de Produtos com Estoque Mínimo em Python III

Tupla com categorias

```
categoria_opcoes = ("Alimento", "Limpeza", "Higiene")
```

- Aqui criamos uma **tupla** chamada **categoria_opcoes**.
- **Tupla** é uma estrutura imutável que armazena vários valores. Neste caso, três categorias possíveis.

```
print("Categorias disponíveis:")
```

```
for i, categoria in enumerate(categoria_opcoes, 1):
```

```
    print(f"{i} - {categoria}")
```

- Mostramos na tela as opções de categoria, usando **enumerate()** para numerar de 1 a 3.

```
escolha = int(input("Escolha o número da categoria: "))
```

- Aqui o usuário digita o número da categoria que deseja.
- Convertido para inteiro com **int()**.

if escolha == 1:

categoria = categoria_opcoes[0]

elif escolha == 2:

categoria = categoria_opcoes[1]

elif escolha == 3:

categoria = categoria_opcoes[2]

else:

categoria = "Outros"

- Usamos **if**, **elif** e **else** para verificar qual número foi escolhido:
- **if escolha == 1** → categoria será "**Alimento**"
- **elif** verifica se foi 2 ou 3
- **else** cobre qualquer número inválido, e define a categoria como "Outros"

```
return {  
  
    "nome": nome,  
  
    "preco": preco,  
  
    "quantidade": quantidade,  
  
    "estoque_minimo": estoque_minimo,  
  
    "categoria": categoria  
  
}
```

- O comando **return** devolve um dicionário com as informações do produto.
- Dicionário é uma estrutura com pares chave-valor.

Sistema de Cadastro de Produtos com Estoque Mínimo em Python IV

Função para mostrar produtos com estoque abaixo do mínimo

def mostrar_produtos_estoque_baixo(produtos):

- Outra função, chamada **mostrar_produtos_estoque_baixo**.
- Ela recebe um argumento chamado **produtos** (que esperamos que seja uma lista de dicionários).

print("\nProdutos com estoque abaixo do mínimo:")

- O **print()** imprime um texto na tela.

for produto in produtos:

- Aqui temos um laço de repetição **for**.
- Ele percorre cada item da lista **produtos** e armazena temporariamente na variável **produto**.
- Cada produto é um dicionário com as informações que cadastramos.

if produto["quantidade"] < produto["estoque_minimo"]:

- Usamos uma estrutura de decisão **if** para verificar se a quantidade em estoque é menor que o estoque mínimo.

print(f"{produto['nome']} - Quantidade: {produto['quantidade']}")

- Se a condição for verdadeira, mostramos o nome e a quantidade do produto na tela.
- **f""** é uma **string** formatada que permite inserir variáveis diretamente no texto.

Sistema de Cadastro de Produtos com Estoque Mínimo em Python V

Lista e loop principal com while True e break

produtos = []

- Criamos uma **lista** chamada produtos, inicialmente vazia.
- Vamos guardar nela os dicionários retornados pela função **cadastrar_produto()**.

while True:

- Um laço **while** com condição sempre verdadeira.
- Executa para sempre, até que usamos **break**.

produto = cadastrar_produto()

- Chamamos a função **cadastrar_produto()**.
- O valor retornado (dicionário com os dados) é guardado na variável **produto**.

resposta = input("Deseja cadastrar outro produto? (s/n): ").lower()

- Perguntamos se o usuário quer continuar cadastrando.
- **.lower()** transforma a resposta em minúsculas.

if resposta != 's':

break

- Se a resposta for diferente de 's', usamos **break** para sair do **while**.

Sistema de Cadastro de Produtos com Estoque Mínimo em Python VI

Mostrar produtos com estoque baixo

mostrar_produtos_estoque_baixo(produtos)

- Após sair do laço, chamamos a função **mostrar_produtos_estoque_baixo()** para exibir os produtos com estoque abaixo do mínimo.

Sistema de Cadastro de Produtos com Estoque Mínimo em Python VII

Código completo

```
def cadastrar_produto():
    nome = input("Digite o nome do produto: ")
    preco = float(input("Digite o preço do produto: "))
    quantidade = int(input("Digite a quantidade em estoque: "))
    estoque_minimo = 5

    categoria_opcoes = ("Alimento", "Limpeza", "Higiene")
    print("Categorias disponíveis:")
    for i, categoria in enumerate(categoria_opcoes, 1):
        print(f"{i} - {categoria}")

    escolha = int(input("Escolha o número da categoria: "))
    if escolha == 1:
        categoria = categoria_opcoes[0]
    elif escolha == 2:
        categoria = categoria_opcoes[1]
    elif escolha == 3:
        categoria = categoria_opcoes[2]
    else:
        categoria = "Outros"

    return {
        "nome": nome,
        "preco": preco,
        "quantidade": quantidade,
        "estoque_minimo": estoque_minimo,
        "categoria": categoria
    }

def mostrar_produtos_estoque_baixo(produtos):
    print("\nProdutos com estoque abaixo do mínimo:")
    for produto in produtos:
        if produto["quantidade"] < produto["estoque_minimo"]:
            print(f"{produto['nome']} - Quantidade: {produto['quantidade']}")

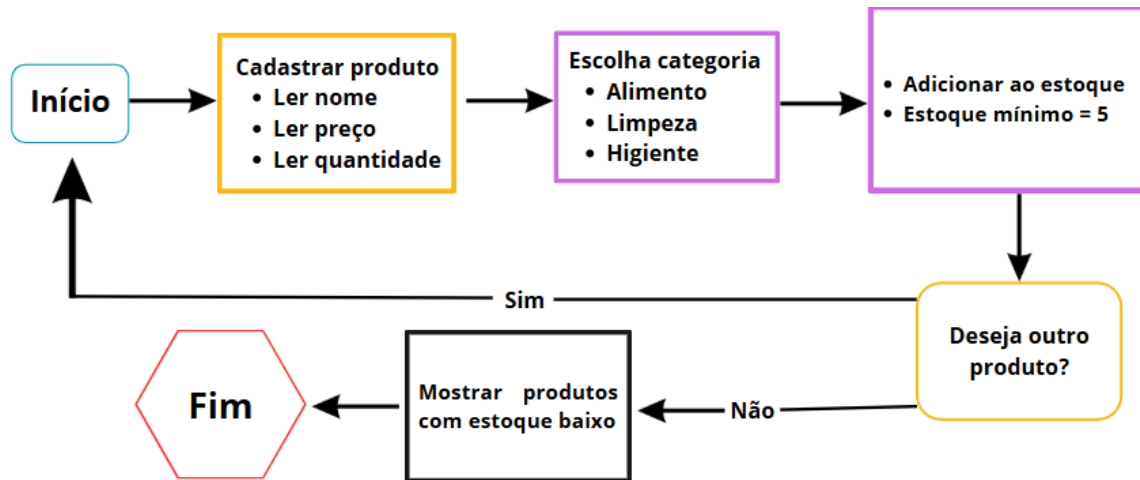
produtos = []

while True:
    produto = cadastrar_produto()
    produtos.append(produto)
    resposta = input("Deseja cadastrar outro produto? (s/n): ").lower()
    if resposta != 's':
        break

mostrar_produtos_estoque_baixo(produtos)
```

Sistema de Cadastro de Produtos com Estoque Mínimo em Python VIII

Fluxograma



Referências

- 1: <https://blog.casadodesenvolvedor.com.br/logica-de-programacao/>
- 2: <https://www.dca.fee.unicamp.br/cursos/EA876/apostila/HTML/no-de7.html>
- 3: Algorithmics: The Spirit of Computing. Addison-Wesley. 2004.
- 4: <https://www.newtoncbraga.com.br/index.php/microcontroladores/142-texas-instruments/8217-microcontrolador-msp430-parte-iii-mic094>
- 5: <https://docs.python.org/3/library/functions.html#input>
- 6: <https://docs.python.org/3/library/functions.html#int>
- 7: <https://docs.python.org/3/tutorial/datastructures.html#tuples-and-sequences>