Programação Estruturada

Marcus Vinicius Aquino Costa
Otávio Fernandes Santos e Silva
Reilson Batista da Fonseca
João Ricardo Santana

Sumário Teórico

- 3: O que é Programação Estruturada?
- 4: Características Principais:
- 5: Como surgiu a Programação Estruturada? I
- 6: Como surgiu a Programação Estruturada? II
- 7: Como surgiu a Programação Estruturada? III
- 8: Como surgiu a Programação Estruturada? IV
- 9: Como surgiu a Programação Estruturada? V
- 10: Como surgiu a Programação Estruturada? VI
- 11: Por que a Programação Estruturada é Importante?
- 12: Principais Motivos da Importância I
- 13: Principais Motivos da Importância II
- 14: Analogia
- 15: Diagrama de Blocos (Fluxograma Básico)
- 16: Aplicações Reais da Programação Estruturada I
- 17: Aplicações Reais da Programação Estruturada II
- 18: Resumo Teórico – Programação Estruturada I
- 19: Resumo Teórico – Programação Estruturada II
- 20: Resumo Teórico – Programação Estruturada III
- 21: Resumo Teórico – Programação Estruturada IV
- 22: Perguntas
- 23: Referências

O que é Programação Estruturada?

Programação Estruturada é um paradigma de programação que organiza o código em blocos bem definidos e controlados, com o objetivo de torná-lo mais legível, modular e fácil de manter. Esse paradigma surgiu como uma alternativa à programação não estruturada (com uso excessivo de comandos como goto), que gerava códigos confusos e difíceis de entender — o famoso "código espaguete".

Características Principais:

Controle de Fluxo Estruturado:

Usa apenas três estruturas básicas:

- Sequência: execução linha por linha;
- Decisão: if, else, switch;
- Repetição: for, while, do-while.

Modularização:

O programa é dividido em funções ou procedimentos, cada um responsável por uma tarefa específica.

Uso de Variáveis Locais e Globais com Cuidado

Incentiva o uso de variáveis dentro de funções (escopo local), para evitar conflitos e facilitar a manutenção.

Facilidade de Leitura e Depuração:

O código é mais fácil de entender, testar e modificar.

Como surgiu a Programação Estruturada? I

A Programação Estruturada surgiu como resposta à crescente complexidade dos programas de computador nas décadas de 1950 e 1960. Naquela época, a maioria dos códigos era escrita de forma linear, com uso frequente do comando GOTO, que fazia o programa "pular" de uma parte para outra do código, sem uma ordem clara. Isso levava a programas difíceis de entender, testar e manter — o chamado “código espaguete”.

O Problema:

- Programas cresciam e ficavam incontroláveis.
- O código era confuso e difícil de depurar.
- Qualquer alteração podia gerar novos erros.

Como surgiu a Programação Estruturada? II

A solução: Programação Estruturada

Nos anos 60, o cientista da computação Edsger W. Dijkstra propôs uma nova forma de programar:

- Em vez de usar saltos (GOTO), usar blocos estruturados de controle de fluxo: sequência, decisão, repetição.
- Escrever códigos claros, organizados e previsíveis.

Em 1968, Dijkstra publicou o artigo “Go To Statement Considered Harmful”, que foi um marco para esse movimento.

Esse artigo defendia que programas devem ser escritos com estruturas de controle bem definidas, sem depender de desvios incontroláveis no fluxo.

Impacto:

- O conceito de modularização (dividir o programa em funções) começou a ser amplamente adotado.
- Surgiram linguagens de programação com suporte direto a esse paradigma, como Pascal e C.
- Tornou-se a base do ensino de programação em escolas e universidades.

Como surgiu a Programação Estruturada? III

1970 – Teorema de Böhm-Jacopini

Contexto: A comprovação de que qualquer algoritmo pode ser construído apenas com três estruturas básicas (sequência, decisão, repetição) foi o grande impulso teórico.
Esse teorema deu base à criação de linguagens que impediam ou desencorajavam o uso de GOTO.
Passou a orientar manuais, livros didáticos e currículos universitários**.**

1970-1980 – Linguagens estruturadas ganham força

Pascal (1970)

Desenvolvedor: Niklaus Wirth
Propósito: Ensino de boas práticas de programação.
Características marcantes:
Tipagem forte e estática
Estruturas de controle claras
Modularização por meio de procedures e functions
Uso: Muito adotado em universidades até os anos 1990.

C (1972)

- Desenvolvedor: Dennis Ritchie (em Bell Labs)
- Importância:
- Linguagem estruturada de baixo nível
- Ideal para escrever sistemas operacionais e drivers
- Base para o sistema UNIX
- Influenciou várias outras linguagens (C++, Java, C#)
- Controle de fluxo completo:
- if, else, switch, for, while, do-while

Modula-2 e Ada

- Evoluções do Pascal voltadas para aplicações maiores e críticas (Ada, por exemplo, foi usada pelo Departamento de Defesa dos EUA).
- Ada (1980) fortaleceu a modularização e a programação estruturada com contratos formais e verificação estática.

Como surgiu a Programação Estruturada? IV

Anos 1980 – Consolidação da Programação Estruturada

Contexto:

- Após a década de 1970 consolidar a teoria e lançar linguagens como C e Pascal, os anos 80 focaram na prática, ensino e aplicações comerciais.

Principais linguagens:

- Pascal: dominante no ensino (Turbo Pascal)
- C: cada vez mais usado na indústria (principalmente sistemas)
- Ada: linguagem estruturada desenvolvida pelo Departamento de Defesa dos EUA para aplicações críticas

Educação:

- Pascal virou linguagem-padrão em escolas e universidades
- Prática de ensino centrada em estruturas: sequência, decisão, repetição
- Introdução ao uso de fluxogramas, pseudocódigos, e estruturas condicionais e de laço

Indústria:

- **Introdução das metodologias:**
  - **Análise estruturada (Yourdon, DeMarco)**
  - **Projeto estruturado**
- **Uso crescente de ferramentas CASE e diagramação (DFD, ER)**

Anos 1990 – Transição para Orientação a Objetos, sem abandonar a estrutura

Contexto:

- Surge e se populariza a Programação Orientada a Objetos (POO)
- Mesmo assim, a base estruturada permanece intacta, agora dentro de métodos e classes

Linguagens importantes:

- C++ (1985): extensão de C com suporte a objetos, mas mantendo estruturas como if, for, while
- Java (1995): 100% OO, mas com lógica estruturada dentro dos métodos
- Delphi: evolução do Pascal com POO

Educação:

- Ensino começa a migrar de Pascal para C/C++
- Livros didáticos combinam lógica estruturada com introdução à OO
- Fluxogramas e pseudocódigo ainda baseiam a formação em lógica

Indústria:

- Desenvolvimento de sistemas mais complexos usando:
- Arquitetura em camadas
- Métodos como RUP (Rational Unified Process)

Como surgiu a Programação Estruturada? V

Anos 2000 – Linguagens modernas e popularização da web

Contexto:

- A Web exige novos estilos de programação (cliente/servidor, eventos), mas a lógica estruturada continua nas bases dos scripts, backends e APIs.

Linguagens populares:

- Python: altamente legível, estrutura de código clara e organizada
- C#: inspirado em C++/Java, forte estrutura de controle e OO
- PHP, JavaScript: dominam o desenvolvimento web com base em estruturas tradicionais
- VB.NET: evolução estruturada + OO do Visual Basic

Educação:

- Ensino de lógica de programação baseado em pseudocódigo, com tradução para Python ou C++
- Linguagens como Python ganham espaço pelo foco em clareza e estrutura

Indústria:

- Desenvolvimento ágil e web com base em frameworks como:
  - .NET (C#), Spring (Java), Django (Python)
- Programas ainda fortemente estruturados em métodos e funções bem definidos

2010–Hoje – Paradigmas híbridos e programação visual

Contexto:

- Linguagens modernas misturam vários paradigmas (OO, funcional, orientado a eventos), mas todas herdam a lógica estruturada.
- Programação visual e ensino para crianças reforçam conceitos estruturados com blocos visuais.

Linguagens e ferramentas:

- Kotlin, Swift: modernas, estruturadas, OO + funcional
- Python: padrão de ensino de lógica estruturada em escolas e universidades
- Scratch, Blockly: blocos que representam if, repeat, while de forma visual
- TypeScript: estrutura lógica mais rigorosa sobre JavaScript

Educação atual:

- Introdução à lógica com programação visual (Scratch, Code.org)
- Transição para Python ou Java
- Currículos de ensino técnico e superior continuam com algoritmos estruturados antes da POO

Indústria:

- A estrutura ainda é a base do raciocínio algorítmico, mesmo em projetos com IA, machine learning ou APIs:
  - Scripts com controle sequencial
  - Condições e loops ainda essenciais em todas as áreas

Como surgiu a Programação Estruturada? VI

Linha do tempo simplificada:

1950-1960 — Problema
Programas caóticos com uso excessivo de GOTO (código espaguete).
Dificuldade de manutenção e entendimento.
1968 — Solução proposta
Edsger W. Dijkstra publica o artigo:
“Go To Statement Considered Harmful”
Defende o uso de estruturas: sequência, decisão e repetição.
1970 — Teorema de Böhm-Jacopini
Comprova que qualquer algoritmo pode ser construído apenas com:

- Sequência
- Decisão
- Repetição

Anos 1970 — Linguagens estruturadas surgem
Pascal (1970) — Ensino de boas práticas.
C (1972) — Uso industrial e desenvolvimento de sistemas.

Anos 1980 — Consolidação
Uso massivo no ensino e na indústria.
Ferramentas de modelagem (DFD, ER).
Linguagens: Pascal, C, Ada.
Anos 1990 — Transição
Chegada da Programação Orientada a Objetos (POO).
Códigos OO mantêm base estruturada.
Linguagens: C++, Java, Delphi.
Anos 2000 — Web e modernização
Estruturas aplicadas no desenvolvimento web.
Linguagens: Python, C#, PHP, JavaScript.
Ensino começa a focar em Python pela clareza.
2010 — Hoje — Paradigmas híbridos
Integração de estruturado + OO + funcional.
Ensino começa com programação visual (Scratch, Blockly).
Linguagens modernas: Python, Kotlin, Swift, TypeScript.
Estruturas de controle ainda são base em qualquer programação.

Por que a Programação Estruturada é Importante?

A Programação Estruturada é fundamental porque estabelece boas práticas de organização e clareza no código, facilitando tanto o desenvolvimento quanto a manutenção de programas. Ela é considerada a base da lógica de programação moderna, sendo essencial para quem está aprendendo a programar ou desenvolvendo sistemas de pequeno a médio porte.

Principais Motivos da Importância I

Facilita a leitura e compreensão do código

- O código segue um fluxo lógico e previsível (início → meio → fim).
- Quem não escreveu o programa consegue entender com facilidade.

Facilita a manutenção e atualização

- Programas estruturados são mais fáceis de alterar sem gerar novos erros.
- Dividir o código em funções/modularização permite alterar partes específicas sem impactar o todo.

Melhora o processo de depuração (debug)

- Como cada bloco tem uma função clara, erros são mais fáceis de identificar e corrigir.

Principais Motivos da Importância II

Estimula boas práticas de lógica

- Ajuda a desenvolver o pensamento computacional de forma lógica e organizada.
- Forma a base para outros paradigmas, como a Programação Orientada a Objetos.

Facilita o reaproveitamento de código

- Funções e blocos podem ser reutilizados em outros projetos com pouca ou nenhuma modificação.

Melhora o trabalho em equipe

- Em projetos colaborativos, a organização estrutural permite que múltiplas pessoas contribuam sem se perder no código.

Analogia

Montar um móvel com manual (IKEA, por exemplo)

- O manual mostra um passo a passo com instruções claras e sequenciais.
- Cada passo depende do anterior: você não pula etapas.
- Se algo der errado, você pode voltar ao ponto específico e corrigir.

// Relação com programação estruturada:

Assim como montar um móvel, escrever um código estruturado exige sequência lógica, organização e execução passo a passo.

https://img.freepik.com/fotos-premium/retrato-de-homem-montando-moveis-faca-voce-mesmo-montagem-de-moveis-em-casa_230311-47844.jpg

Receita de bolo

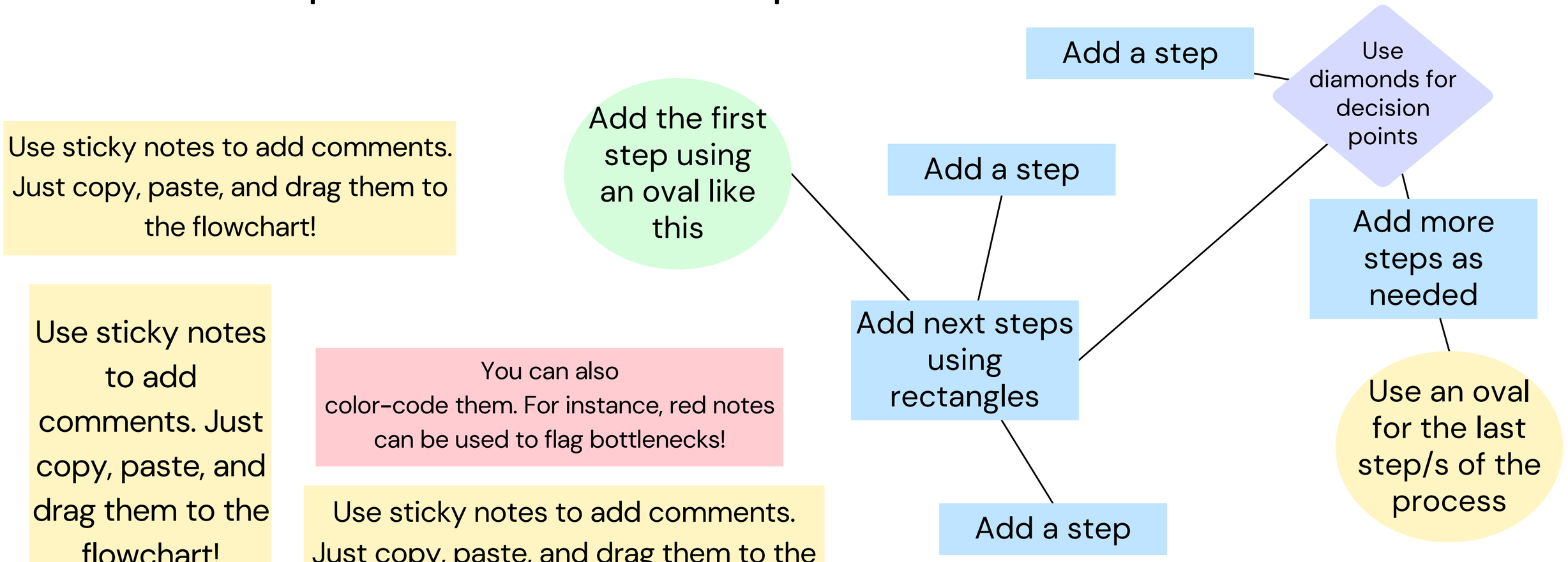
- Ingredientes (variáveis) → são preparados.
- Passos (funções) → são executados na ordem correta.
- Condicional: “Se o forno estiver quente, asse por 30 minutos.”
- Repetição: “Mexa até a massa ficar homogênea.”

// A receita usa estruturas como sequência, decisão e repetição, assim como um programa estruturado.

https://guiadacozinha.com.br/wp-content/uploads/2020/01/pessoa-fazendo-bolo.jpg

Diagrama de Blocos (Fluxograma Básico)

Um fluxograma é uma forma gráfica de representar um algoritmo estruturado. Aqui está um modelo simples:



Fluxograma ilustrativo com modelo pronto do Canva

Aplicações Reais da Programação Estruturada I

Microcontroladores Simples (Arduino, sensores)

Dispositivos embarcados como sensores de temperatura, luz, ou sistemas de irrigação automática usam:

- Decisões condicionais (ex: se a temperatura passar de 30°C → ligar ventilador).
- Laços de repetição para monitoramento contínuo.

https://th.bing.com/th/id/R.0b2e12313863a1d039baaaf0e97515ba?rik=1N4cJiu3j4GzcA&riu=http%3a%2f%2fmaxdesign.com.br%2fletras-em-mdf%2fwp-content%2fuploads%2f2015%2f12%2farduino-microcontroladores.png&ehk=jYygUWx3S6RAgqgxTWeCGobYuGChKooDmxCUJgcjR8c%3d&risl=1&pid=ImgRaw&r=0

Scripts Automatizados (Batch, Shell, Python básico)

Programas que:

- Copiam arquivos automaticamente.
- Fazem backup diário.
- Renomeiam documentos.
- Automatizam tarefas repetitivas com laços (for, while) e decisões (if).

https://gaea.com.br/wp-content/uploads/2020/05/original-f58009d8f2b6286d21bd1a4f97e43b01.jpeg

Aplicações Reais da Programação Estruturada II

Softwares de Cadastro (biblioteca, escola, mercado)

Exemplo: um sistema que cadastra alunos.

- Recebe nome, idade e nota.
- Calcula a média.
- Exibe se o aluno está aprovado ou reprovado.

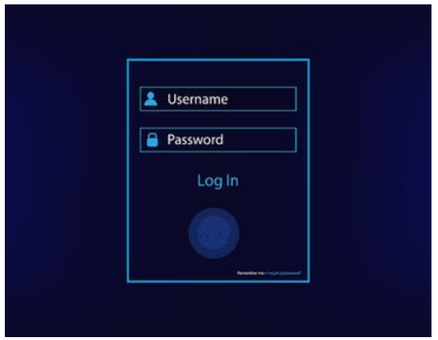
Jogos Simples

- Um jogo de adivinhação, jogo da velha ou roleta digital.
- A lógica é toda baseada em estruturas sequenciais, condicionais e repetitivas.

Menu de Senhas ou Autenticação

- Um sistema que verifica se o usuário digitou a senha certa.
- Se sim, permite o acesso; se não, exibe uma mensagem de erro — exemplo direto de uso de if e else.

https://www.escolagames.com.br/diversao/jogo-da-velha



https://pt.vecteezy.com/arte-vetorial/15639278-menu-de-formulario-de-nome-de-usuario-e-senha-ou-login-de-identificacao-de-impressao-digital

Resumo Teórico – Programação Estruturada 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.

Resumo Teórico – Programação Estruturada 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.

Resumo Teórico – Programação Estruturada 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.

Resumo Teórico – Programação Estruturada 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?

Referências

- 1: https://blog.casadodesenvolvedor.com.br/logica-de-programacao/
- 2: https://www.dca.fee.unicamp.br/cursos/EA876/apostila/HTML/node7.html
- 3: Algorithmics: The Spirit of Computing. Addison-Wesley. 2004
- 4:http://www.newtoncbraga.com.br/index.php/microcontroladores/142-texas-instruments/8217-microcontrolador-msp430-parte-iii-mic094

Sumário Prático

- 25: Sistema de Cadastro de Produtos com Estoque Mínimo em Python I
- 26: Sistema de Cadastro de Produtos com Estoque Mínimo em Python II
- 27, 28: Sistema de Cadastro de Produtos com Estoque Mínimo em Python III
- 29: Sistema de Cadastro de Produtos com Estoque Mínimo em Python IV
- 30: Sistema de Cadastro de Produtos com Estoque Mínimo em Python V
- 31: Sistema de Cadastro de Produtos com Estoque Mínimo em Python VI
- 32: Sistema de Cadastro de Produtos com Estoque Mínimo em Python VII
- 33: Sistema de Cadastro de Produtos com Estoque Mínimo em Python VIII
- 34: Referências

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 I

```
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()**.

Sistema de Cadastro de Produtos com Estoque Mínimo em Python III

Tupla com categorias II

```
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.

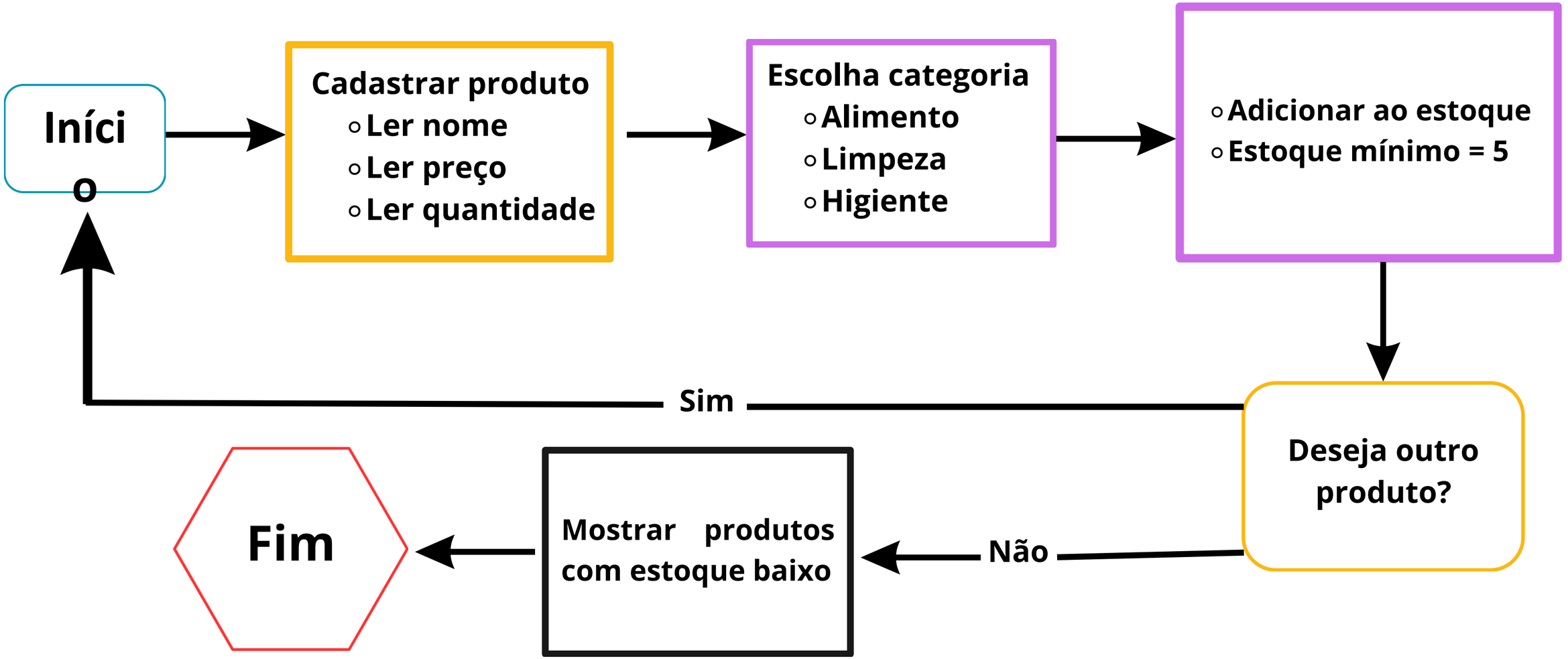
Sistema de Cadastro de Produtos com Estoque Mínimo em Python VII

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']}")
        else:
            print("Nenhum")

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/node7.html
- 3: Algorithmics: The Spirit of Computing. Addison-Wesley. 2004.
- 5: https://www.newtoncbraga.com.br/index.php/microcontroladores/142-texas-instruments/8217-microcontrolador-msp430-parte-iii-mic094
- 6: https://docs.python.org/3/library/functions.html#input
- 7: https://docs.python.org/3/library/functions.html#int
- 8: https://docs.python.org/3/tutorial/datastructures.html#tuples-and-sequences