Autoria: Anna Kelly S S Sousa
Maio/2024

LÓGICA DE PROGRAMAÇÃO PARA INICIANTES: DOMINE A ARTE DE PENSAR COMO UM COMPUTADOR

ANNA SOUSA

INTRODUÇÃO

Bem-vindos a esta jornada pelo mundo fascinante da lógica de programação. Através deste e-book tenho a honra de guiá-los passo a passo pelos conceitos básicos da lógica, a linguagem fundamental para construir programas poderosos e eficientes. Nesta aventura, desvendaremos os mistérios da lógica de forma clara e acessível, utilizando analogias criativas e exemplos práticos que facilitarão o aprendizado. Imaginem-se como magos aprendendo a conjurar feitiços mágicos para controlar máquinas e realizar tarefas complexas.

Ao longo desta jornada, exploraremos os blocos de construção da lógica, como variáveis, operadores, estruturas de controle e laços de repetição. Cada um desses elementos é uma ferramenta poderosa que nos permitirá criar programas que resolvam problemas, automatizem tarefas e até mesmo criem jogos divertidos.

Lembrem-se, a lógica de programação é a base para se tornar um programador de sucesso. Com dedicação e persistência, vocês poderão alcançar seus objetivos e criar projetos cada vez mais desafiadores e complexos.

Embarque nesta jornada épica e domine a arte de pensar como um computador!

01

O QUE É LÓGICA DE PROGRAMAÇÃO E POR QUE ELA É IMPORTANTE?

Autoria: Anna Kelly S S Sousa
Maio/2024

Imagine um chef de cozinha experiente que para preparar um prato delicioso, ele precisa seguir uma receita passo a passo, combinando os ingredientes certos na ordem correta para que o resultado seja perfeito. Ou quando você está montando um lego e segue as instruções de montagem, elas te guiam para que você consiga construir o brinquedo corretamente. Se você seguir as instruções com atenção, o Lego ficará perfeito! Mas se você trocar as peças ou pular um passo, o resultado pode ser um desastre. A lógica de programação funciona de forma parecida: ela define as regras, o passo a passo que o computador precisa seguir para realizar uma tarefa específica.
Em termos simples, a lógica de programação é o conjunto de ferramentas e técnicas que você utiliza para estruturar o raciocínio de um computador. É como um manual de instruções detalhado que guia a máquina a realizar uma determinada ação. Ao dominar a lógica de programação, você poderá criar programas usando uma linguagem de programação.

Mas, você deve tá se perguntando: Por que a Lógica de Programação é importante?

Autoria: Anna Kelly S S Sousa
Maio/2024

A lógica de programação é essencial para diversas áreas do conhecimento, pois permite a criação de softwares, aplicativos, websites e até mesmo sistemas de inteligência artificial. Ela é a base fundamental para:

Automatizar tarefas: Imagine um robô que realiza tarefas repetitivas em uma fábrica, como montar peças ou embalar produtos. A lógica de programação é o que permite que o robô execute essas tarefas com precisão e eficiência.
Solucionar problemas: Imagine um GPS que calcula a rota mais curta entre duas cidades. A lógica de programação é o que permite que o programa analise diferentes caminhos e encontre a opção mais rápida e eficiente.
Controlar dispositivos: Imagine um sistema de automação residencial que controla luzes, eletrodomésticos e outros dispositivos. A lógica de programação é o que permite que o sistema interprete seus comandos e execute as ações desejadas.
Simular realidades complexas: Imagine um simulador de voo que reproduz com precisão o comportamento de um avião em diferentes situações. A lógica de programação é o que permite que o simulador leve em conta fatores como física, aerodinâmica e condições climáticas para criar uma experiência realista.

Autoria: Anna Kelly S. da S. Sousa

02

ENTENDENDO ALGORITMOS

Nesta jornada pelo mundo da lógica de programação, você aprenderá a arte de pensar como um computador. Para isto, veja-se como um mago desbravando os segredos da lógica, aprendendo a criar receitas mágicas (algoritmos) que transformam suas ideias em programas poderosos e eficientes.

Afinal, o que são Algoritmos?

Em resumo, algoritmos são como instruções detalhadas que guiam um computador passo a passo para realizar uma tarefa ou resolver um problema. São como receitas mágicas que definem a ordem e a forma como as ações devem ser executadas.

E por que os Algoritmos são importantes?

Dominar os algoritmos é fundamental para entender como funciona a programação, pois eles permitem:

- Criar programas eficientes: Algoritmos bem estruturados otimizam o tempo e os recursos do computador, tornando seus programas mais rápidos e eficientes.
- Resolver problemas complexos: Algoritmos poderosos podem lidar com desafios complexos, dividindo-os em etapas menores e solucionáveis.
- Comunicar ideias com clareza: Algoritmos bem documentados facilitam a compreensão e a colaboração entre programadores.

Desvendando os Passos da Criação de um Algoritmo:

- Definição do Problema: O primeiro passo é entender claramente o problema que você deseja resolver. Imagine um mago que precisa encontrar um ingrediente específico em seu grimório. Qual é o ingrediente? Onde ele está escondido? Quais informações você tem para encontrá-lo?

- Análise do Problema: Divida o problema em etapas menores e mais fáceis de gerenciar. Imagine o mago dividindo a busca pelo ingrediente em etapas: consultar o índice do grimório, procurar na seção correta, verificar cada página, etc.

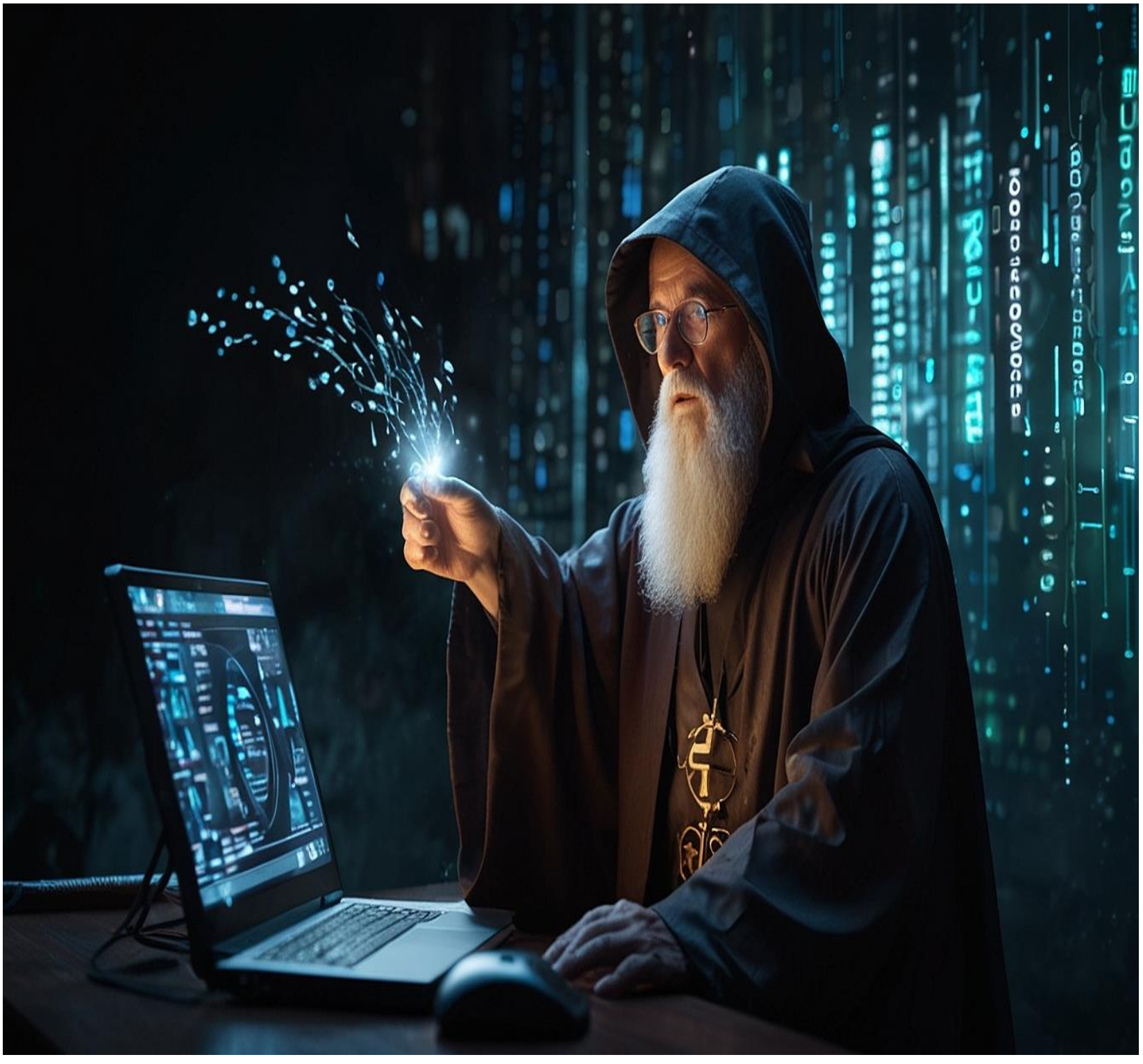
- Desenvolvimento do Algoritmo: Escreva o algoritmo em uma linguagem de programação clara e compreensível. Imagine o mago escrevendo as instruções passo a passo para buscar o ingrediente, utilizando comandos de busca e lógica.

- Teste e Refinamento: Teste o algoritmo com diferentes cenários para garantir que ele funcione corretamente. Imagine o mago testando o algoritmo com diferentes ingredientes e grimórios para verificar sua eficiência.

Autoria: Anna Kelly S. da S. Sousa

Autoria: Anna Kelly S S Sousa
Maio/2024

- Documentação: Documente o algoritmo para facilitar sua compreensão e reutilização no futuro. Imagine o mago escrevendo um manual detalhado sobre o algoritmo, explicando seu funcionamento e como usá-lo.



03

DESVENDANDO OS BLOCOS DE CONSTRUÇÃO DA LÓGICA

Autoria: Anna Kelly S S Sousa
Maio/2024

Neste capítulo dessa jornada épica pelo mundo da lógica de programação, exploraremos os blocos de construção que dão vida aos nossos programas. Imagine-se como um alquimista desbravando um laboratório mágico, descobrindo os segredos de ingredientes poderosos que transformarão suas ideias em realidade.
Agora vamos falar sobre elementos fundamentais da lógica de programação, como:

1. **Variáveis**: Baús Mágicos de Tesouros

Imagine caixas mágicas escondidas em um calabouço encantado. Cada caixa contém um item valioso, como uma espada, um mapa do tesouro, entre outros. As variáveis funcionam da mesma forma, armazenando informações importantes para o seu programa. São pequenos espaços ocupados na memória do seu computador guardando alguma informação. Elas podem conter números, textos, até mesmo outros programas!
Exemplo: Imagine um programa que calcula a média de três notas. As variáveis podem armazenar as notas dos alunos, como nota1, nota2 e nota3.

2. Operadores: Feitiços Mágicos para Manipular Informações

Imagine feitiços mágicos que realizam cálculos, comparações e operações lógicas. Os operadores são como esses feitiços, permitindo que você manipule as informações armazenadas nas variáveis. São eles:

Autoria: Anna Kelly S S Sousa
Maio/2024

Operadores aritméticos: Realizam operações matemáticas básicas como adição (+), subtração (-), multiplicação (*) e divisão (/). Pensem em usar um feitiço para calcular a soma de três números.
Exemplo: valor1 = 5; valor2 = 6.5;   valor3 = 8.5
Soma = valor1 + valor2 + valor3 = 20

Operadores de comparação: Comparar valores para tomar decisões. Imagine usar um feitiço para verificar se um número é maior, menor ou igual a outro. Usamos o sinais de menor que(<), maior que (>), igual(=), não igual(!=).
OBS.: no caso do “não igual” usamos para indicar que na comparação os resultados foram diferentes. E ele é representado pelo ponto de exclamação (!) seguindo do sinal de igualdade (=).
Exemplo: valor1 = 5 valor2 = 6.5 → valor1 < valor2 (5 é menor que 6.5)

Operadores lógicos: Combinar informações para criar condições complexas. Imagine usar um feitiço para verificar se duas condições são verdadeiras ao mesmo tempo. Para isso, temos os seguintes operadores:

- **E (&&):** A União Mágica

O operador E (&&) funciona como a união de dois encantos para um feitiço. Ele só é bem-sucedido se ambas as condições forem verdadeiras. É como se você precisasse recitar duas palavras mágicas para que o feitiço funcione:

Exemplo: Imagine que você quer criar um portal mágico que só se abre se a pessoa for bruxa e estiver usando um chapéu de mago. Para isso, você usaria o feitiço **E**:

Se (pessoa é bruxa) E (pessoa usa chapéu de mago)
  Então: abrir portal

- **OU (||):** A Disjunção Mágica

O operador **OU** (||) funciona como a disjunção de dois encantos para um feitiço. Ele é bem-sucedido se pelo menos uma das condições for verdadeira. É como se você tivesse duas palavras mágicas que podem abrir o portal:
Exemplo: Voltando ao portal mágico, você também pode usar o feitiço OU para que ele se abra se a pessoa for bruxa **ou** estiver usando um chapéu de mago:

Se (pessoa é bruxa) **OU** (pessoa usa chapéu de mago)
  Então: abrir portal

- **NÃO (!):** A Negação Mágica

O operador **NÃO** (!) funciona como a negação de um encantamento. Ele inverte o resultado do feitiço original. É como se você tivesse uma palavra mágica que cancela o efeito de outra:
Exemplo: Imagine que você quer criar um portal que se abre para todos, exceto os trolls. Para isso, você usaria o feitiço NÃO:

Se **NÃO** (pessoa é troll)
  Então: abrir portal

Autoria: Anna Kelly S S Sousa
Maio/2024

Combinando Feitiços: Magia Avançada

Os operadores lógicos podem ser combinados para criar feitiços ainda mais poderosos! Imagine um portal que só se abre se a pessoa for bruxa e estiver usando um chapéu de mago, mas não for um troll:

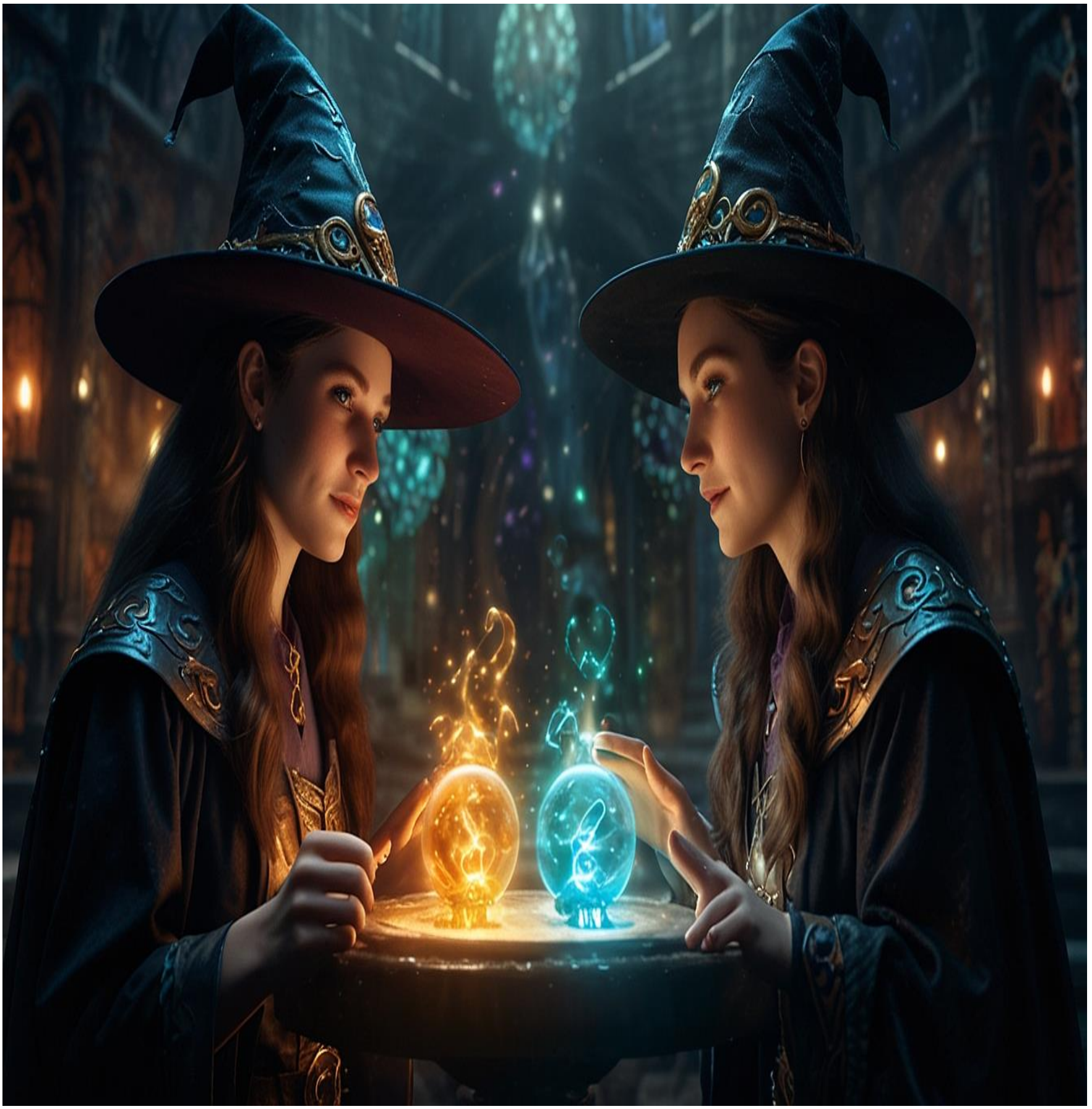
Se (pessoa é bruxa) E (pessoa usa chapéu de mago) E NÃO (pessoa é troll)
  Então: abrir portal
Com os operadores lógicos, você pode criar condições complexas para controlar a magia dos seus programas, assim como um verdadeiro mago experiente!

Lembre-se:

O operador **E (&&)** exige que ambas as condições sejam verdadeiras.
O operador **OU (||)** permite que pelo menos uma das condições seja verdadeira.
O operador **NÃO (!)** inverte o resultado do feitiço original.
Você pode combinar os feitiços para criar condições complexas.
Com prática e criatividade, você se tornará um mestre dos operadores lógicos e poderá criar programas mágicos incríveis!

Autoria: Anna Kelly S. da S. Sousa

Autoria: Anna Kelly S S Sousa
Maio/2024



Autoria: Anna Kelly S. da S. Sousa

04

ESTRUTURAS DE CONTROLE: MAPAS MÁGICOS PARA GUIAR O PROGRAMA

Bem-vindos de volta, bravos aventureiros da lógica de programação! Nesta jornada épica pelo mundo da programação, exploraremos as estruturas de controle, elementos essenciais para guiar nossos passos pelos caminhos corretos. Vejam-se como exploradores desbravando um labirinto mágico, munidos de um mapa que revela o caminho ideal a cada passo.

1. Estruturas de Decisão (if/else): Encruzilhadas e Escolhas Estratégicas

Imagine um cavaleiro em uma encruzilhada, deparando-se com duas opções: seguir pelo caminho da esquerda ou pela direita. A estrutura if permite que o programa, assim como o cavaleiro, tome uma decisão com base em uma condição.

Exemplo: Imagine que o cavaleiro precisa atravessar uma ponte, mas ela só suporta um peso máximo. A estrutura ***if*** (se) pode ser usada para verificar se o cavaleiro e sua armadura ultrapassam esse limite, caso não ultrapasse o peso máximo (condição verdadeira) nosso cavaleiro irá atravessa a ponte:

Se (peso cavaleiro + peso armadura <= limite peso ponte)
Então: Mostrar mensagem: "Ponte segura! Atravesse com cuidado!

1.1 - O Feitiço else: Explorando Novos Caminhos

Mas imagine que o peso do nosso cavaleiro e sua armadura ultrapassaram o peso máximo suportado pela ponte, ele não irá poder atravessar a ponte e terá de procurar uma solução alternativa. É aí que entra o feitiço ***else***, permitindo que o programa explore um caminho diferente caso a condição do **if** seja falsa:

Exemplo:
Se (peso cavaleiro + peso armadura <= limite peso ponte)
Então: Mostrar mensagem: "Ponte segura! Atravesse com cuidado!
Senão: Mostrar mensagem: "Ponte insegura! Retorne e encontre outro caminho, ou nade até o outro lado."

2. Estruturas de Repetição (for e while): Magias Repetidas e Feitiços Incansáveis

Agora vamos imaginar um poderoso mago recitando um encantamento mágico para abrir um portal. Para que o portal se abra completamente, ele precisa repetir o encantamento várias vezes. As estruturas de repetição permitem que o programa execute um conjunto de instruções várias vezes, sem precisar escrevê-las repetidamente, como se o mago estivesse repetindo seu feitiço. É aí que entram os feitiços mágicos da programação: ***for*** e ***while***.

2.1 Feitiço For: Repetição com Precisão

O feitiço for funciona como um encantamento com um número definido de repetições. Imagine que o mago precisa lançar o feitiço de proteção 5 vezes. O feitiço for pode ser usado da seguinte forma:

Para *i* de 1 até 5
Lançar feitiço de proteção
Mostrar mensagem: "Feitiço de proteção lançado com sucesso!"
Próximo *i*

Como funciona:
O feitiço define uma variável chamada *i* que começa com o valor 1.
A cada repetição, o valor de *i* aumenta em 1.
O feitiço repete as instruções dentro do “**Para”** até que o valor de *i* seja maior que 5.
A mensagem "Feitiço de proteção lançado com sucesso!" é exibida a cada repetição.

2.2 Feitiço While: Repetição Indefinida

O feitiço *while* funciona como um encantamento que se repete enquanto uma condição for verdadeira. Imagine que o mago precisa lançar o feitiço de proteção até que os aldeões estejam seguros. O feitiço *while* pode ser usado da seguinte forma:


```
Enquanto aldeões em perigo
    Lançar feitiço de proteção
  Mostrar mensagem: "Protegendo os aldeões!"
Próximo
```

Como funciona:
O feitiço verifica a condição aldeões em perigo.
Se a condição for verdadeira, o feitiço repete as instruções dentro do enquanto.
A mensagem "Protegendo os aldeões!" é exibida a cada repetição.
O feitiço repete até que a condição aldeões_em_perigo se torne falsa.

Autoria: Anna Kelly S. da S. Sousa

Entenda:

O feitiço ***for (PARA)*** é ideal para quando você sabe quantas vezes precisa repetir o encantamento (como no exemplo que eram 5 vezes). Na programação usamos o ***for*** para loop (repetição) fixos, quando sabemos quantas vezes ele irá ser executado até o momento de acabar.

O feitiço ***while (ENQUANTO)*** é ideal para quando você precisa repetir o encantamento até que uma condição seja satisfeita (como no exemplo que o feitiço de proteção ficou se repetindo enquanto tinha os aldeões em perigo, só parou quando todos estavam a salvos). Na programação usamos o ***while*** para loop (repetição) condicional, quando não sabemos ao certo quantas vezes a condição será executada até que seja falsa.

Com os feitiços ***for*** e ***while*** você poderá criar programas que repetem tarefas automaticamente, como um verdadeiro mago da programação!

3.Dominando as Estrutura de Controle Avançada

Agora vamos conhecer a estrutura de controle avançada, o lendário ***switch.***
No mundo da programação, o switch é como um mapa mágico que nos guia por caminhos diferentes, dependendo da informação que temos em mãos. Ele nos permite fazer escolhas complexas de forma organizada e eficiente, levando-nos a diferentes destinos em nosso código.

Autoria: Anna Kelly S. da S. Sousa

Autoria: Anna Kelly S S Sousa
Maio/2024

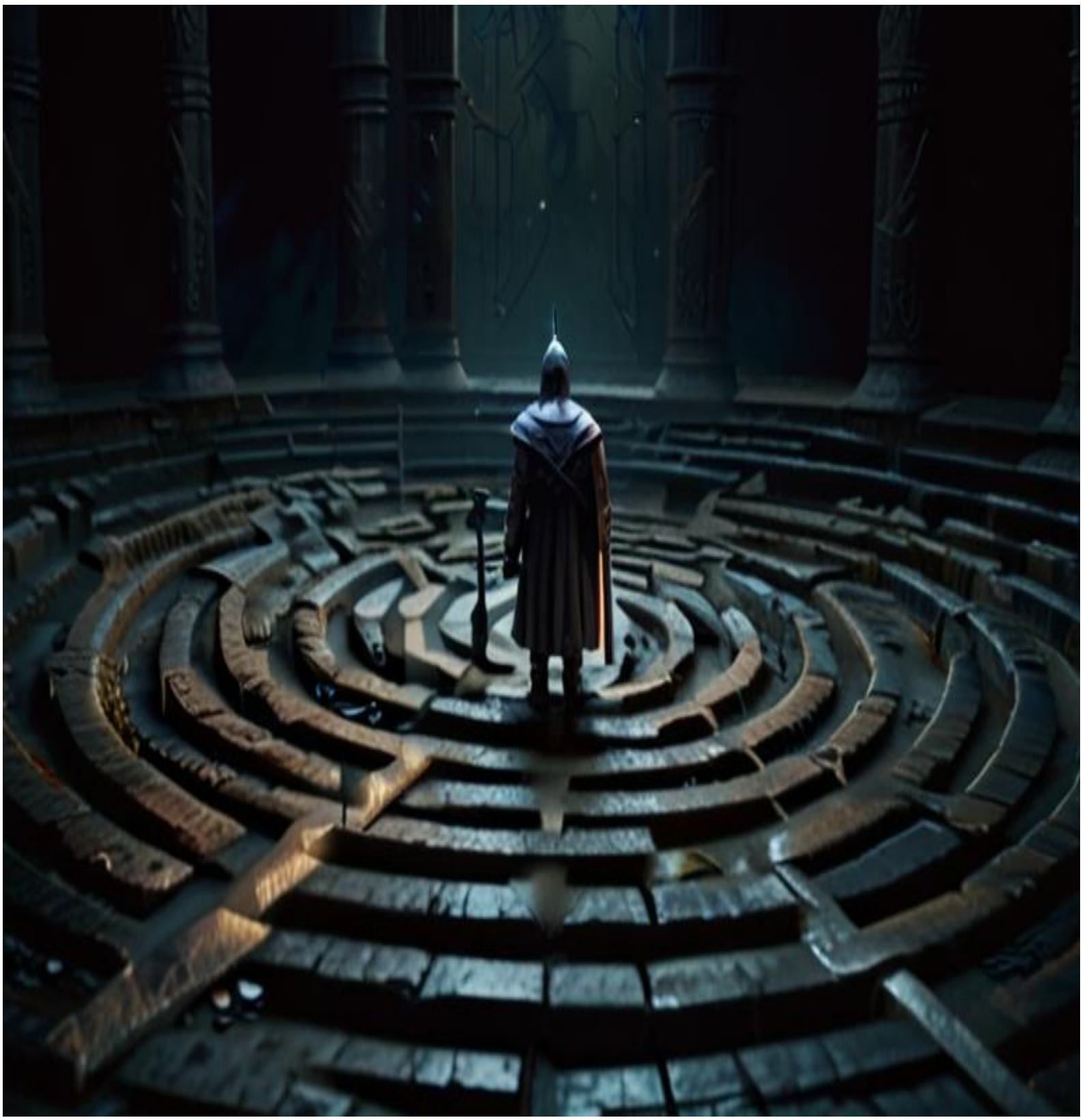
Vamos ilustrar isso com um exemplo épico:
Digamos que você, bravo herói, está em busca de um tesouro perdido. Você entra em uma caverna misteriosa e se depara com três portas mágicas, cada uma levando a uma sala diferente cheia de perigos e desafios. Você consulta o mapa (***o switch***) para decidir qual porta explorar:

```
porta = 2  (Você escolhe a segunda porta)

switch (porta)
  case 1:
    entrar_na_sala_do_dragao()
    break
  case 2:
    entrar_na_sala_do_labirinto()
    break
 case 3:
    entrar_na_sala_do_tesouro()
    break
default:
  voltar_para_casa()
```


Neste código épico, você, herói destemido, decide para onde ir com base no valor da variável porta. Se o valor corresponder a uma das opções listadas (um case), você realiza uma ação específica (neste exemplo, porta 2 você entrará na sala do labirinto). E se nenhum número corresponder às portas disponíveis, você decidirá voltar para casa e planejar sua próxima aventura.

Portanto, o ***switch*** é como um mapa de escolhas em sua jornada épica, guiando-os por diferentes caminhos com base em informações específicas. Que suas decisões sejam sempre sábias e corajosas em sua busca pela glória!



05

FUNÇÕES: A REUTILIZAÇÃO MÁGICA QUE SIMPLIFICA E ORGANIZA OS CÓDIGOS

Aventureiros da lógica de programação, preparem-se para desvendar os mistérios das funções na lógica de programação, ferramentas poderosas que transformam seus programas em obras-primas de eficiência e organização. Imagine um mago experiente que conjura feitiços complexos com um simples gesto. As funções são como os feitiços desses magos, permitindo que você execute tarefas repetitivas de forma modular e reutilizável.

O que são Funções?

Funções são como feitiços mágicos que agrupam instruções relacionadas em um único bloco. Imagine um feitiço para preparar uma poção mágica: ele reúne todas as etapas necessárias, desde a coleta dos ingredientes até a mistura final, em um único comando.

Benefícios das Funções:

- Organização: As funções deixam o código mais organizado e fácil de entender, como um livro com capítulos separados para cada assunto.
- Reutilização: Você pode usar o mesmo feitiço (função) várias vezes, sem precisar repetir todas as instruções, como usar um livro de receitas para fazer diferentes pratos.
- Manutenção: Se precisar modificar algo, você só precisa mudar o feitiço (função) em um único lugar, e a mudança será aplicada automaticamente em todas as vezes que ele for usado.
- Clareza: As funções dão nomes significativos aos blocos de código, tornando-os mais intuitivos e compreensíveis.

Autoria: Anna Kelly S S Sousa
Maio/2024

Vamos imaginar que estamos em uma aventura mágica em busca de um tesouro escondido em um labirinto encantado. Para isso, precisamos de uma função que nos ajude a decifrar os enigmas do labirinto e encontrar o caminho certo.

```
Função: Decifrar Enigma

função decifrar enigma(enigma):
    se enigma for igual a "Água Sagrada":
        retorne "Vire à esquerda"
    senão se enigma for igual a "Chama Eterna":
        retorne "Vire à direita"
    senão:
        retorne "Continue em frente“
```

Neste pseudocódigo:

- função indica que estamos definindo uma função.

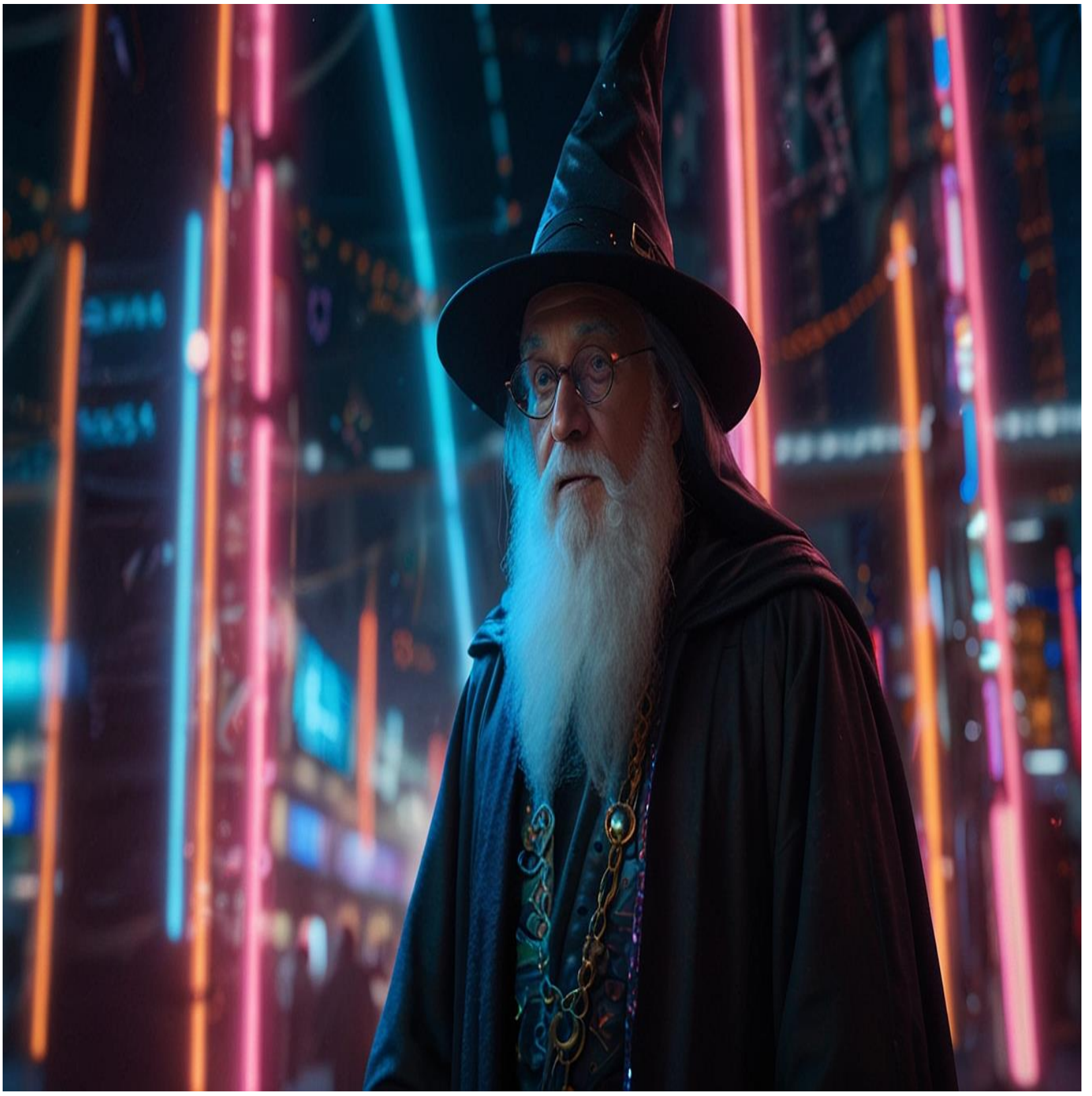
decifrar enigma é o nome da função.
(enigma) é o parâmetro que a função recebe, representando o enigma que encontramos.

- Dentro da função, usamos condicionais para decifrar o enigma:
- Se o enigma for "Água Sagrada", a função sugere virar à esquerda.
- Se o enigma for "Chama Eterna", a função sugere virar à direita.
- Caso contrário, sugere continuar em frente.

retorne é usado para retornar a sugestão de direção com base no enigma recebido.
Agora, sempre que nos depararmos com um enigma no labirinto, podemos invocar nossa função decifrar enigma para nos guiar na direção certa! 🔑 🔮

Autoria: Anna Kelly S. da S. Sousa

Agora, sempre que nos depararmos com um enigma no labirinto, podemos invocar nossa função decifrar enigma para nos guiar na direção certa! 🔑 🔮



06

TESTE SEUS CONHECIMENTOS E APRIMORE SUAS HABILIDADES: UM QUIZ ÉPICO PARA INICIANTES!

Bem-vindos bravos aprendizes da lógica de programação! Chegamos a um momento crucial: o teste de seus conhecimentos! Preparem-se para um quiz épico que irá desafiar seus aprendizados e te ajudar a aprimorar suas habilidades.

Instruções:
Para cada questão, leia atentamente as opções e escolha a alternativa que melhor responde ao problema. Marque apenas uma resposta por questão.

1. Imagine um programa que calcula a média de três notas. Qual variável armazena a média?

(a) media
(b) nota1
(c) nota2
(d) nota3

2. Qual operador é utilizado para comparar se um número é maior que outro?

(a) +
(b) -
(c) *
(d) >

3. Qual estrutura de controle permite que o programa repita um conjunto de instruções um número específico de vezes?

(a) if
(b) else
(c) for
(d) while

4. Qual feitiço mágico permite que o programa explore um caminho alternativo caso a condição do if seja falsa?

(a) abracadabra
(b) alakazam
(c) else
(d) Potion

5. Imagine um programa que calcula a média de todas as notas de um aluno. Qual estrutura de controle você usaria para repetir a operação de cálculo para cada nota?

(a) if
(b) else
(c) for
(d) while

6. Qual estrutura de controle permite que o programa execute um conjunto de instruções enquanto uma condição específica for verdadeira?

(a) if
(b) else
(c) for
(d) while

7. Qual é a função de um switch?

(a) Repetir um conjunto de instruções um número específico de vezes.
(b) Executar diferentes conjuntos de instruções com base em uma condição.
(c) Armazenar um valor que pode ser usado em diferentes partes do programa.
(d) Realizar um cálculo matemático complexo.

8. Qual é a principal diferença entre um for e um while?

(a) O for repete um conjunto de instruções um número específico de vezes, enquanto o while repete enquanto uma condição for verdadeira.
(b) O for é usado para comparar valores, enquanto o while é usado para realizar cálculos matemáticos.
(c) O for é mais eficiente para grandes conjuntos de dados, enquanto o while é mais adequado para situações em que a condição não é conhecida previamente.
(d) O for pode ser usado dentro de um while, mas o while não pode ser usado dentro de um for.

9. Qual das seguintes opções não é uma característica de uma função?

(a) Recebe um conjunto de instruções como parâmetro.
(b) Pode retornar um valor.
(c) Pode ser reutilizada em diferentes partes do programa.
(d) É obrigatória em todos os programas.

Lembre-se: este quiz é apenas um guia para avaliar seu conhecimento. O mais importante é continuar praticando, aprendendo e se divertindo com a lógica de programação!

Respostas:
(a) media
(d) >
(c) for
(c) else
(c) for
(d) while
(b) Executar diferentes conjuntos de instruções com base em uma condição.
(a) O for repete um conjunto de instruções um número específico de vezes, enquanto o while repete enquanto uma condição for verdadeira.
(d) É obrigatória em todos os programas.

Análise de Desempenho:

8 a 9 pontos: Parabéns, você está dominando as estruturas de controle e funções! Continue praticando e explorando novos conceitos para se tornar um programador de sucesso.
6 a 7 pontos: Você está bem encaminhado, mas ainda há alguns pontos que precisam ser reforçados. Revise os conceitos que você teve mais dificuldade e continue praticando exercícios.
4 a 5 pontos: É hora de focar nos estudos! Refaça o quiz, revise os conceitos de estruturas de controle e funções e busque recursos adicionais para te ajudar.
3 pontos ou menos: Não desanime! Aprender lógica de programação requer tempo e dedicação. Comece revisando os conceitos básicos e busque ajuda de um professor ou mentor experiente.

Recursos Incríveis para sua Jornada Contínua:
Explore uma lista épica de recursos para continuar sua jornada na lógica de programação, incluindo:

- Tutoriais online: Guias passo a passo para aprender conceitos específicos da lógica de programação.
- Livros introdutórios: Materiais completos e aprofundados sobre lógica de programação para iniciantes.
- Cursos online: Aulas interativas com professores experientes para te guiar no aprendizado.
- Comunidades online: Espaços para tirar dúvidas, trocar experiências e se conectar com outros programadores iniciantes.

CONCLUSÃO

Parabéns, aventureiro! Você completou sua jornada épica pelo mundo da lógica de programação. Agora você possui as ferramentas e conhecimentos básicos para desvendar os segredos da programação e criar seus próprios programas incríveis. Continue explorando, aprendendo e praticando, e logo você estará pronto para construir projetos cada vez mais desafiadores e complexos. Com o conhecimento e dedicação, você estará pronto para escolher uma linguagem de programação, criar programas incríveis e realizar seus sonhos!

Te desejo muito sorte nessa expedição épica!!!