Educação Profissional Paulista

Técnico em Desenvolvimento de Sistemas

Secretaria da Educação

SÃO PAULO
GOVERNO DO ESTADO

Estruturas de repetição

Conceituação e laço de repetição: *while*

Aula 4

Código da aula: [SIS]ANO1C1B1S5A4

Exposição

Objetivo da aula

Compreender as aplicações do laço *while* em diferentes linguagens de programação.

Competências da Unidade (técnicas e socioemocionais)

- Desenvolver sistemas computacionais utilizando ambiente de desenvolvimento;
- Resolver problemas computacionais com estratégias ágeis.

Recursos didáticos

- Recurso audiovisual para a exibição de vídeos e imagens;
- Caderno para anotações.

Duração da aula

50 minutos.

Exposição

Integração de laços de repetição com estruturas de decisão

A habilidade de integrar laços de repetição com estruturas de decisão é fundamental para a criação de programas complexos e reativos. Isso permite que os programadores escrevam códigos que podem se adaptar a diferentes condições e eventos enquanto operam dentro de ciclos de repetição. A compreensão dessa relação é particularmente vital para resolver problemas que requerem lógica condicional dentro de repetições para decisões dinâmicas.

- Entendimento das estruturas de decisão.
- Uso conjunto com laço *while*.
- Prevenção e manejo de laços infinitos com decisões.

Exposição

Definição e relação entre laços e decisões

Importante

Laços e decisões são dois conceitos fundamentais na programação, que, quando combinados, permitem a criação de programas dinâmicos e interativos.

Laços, ou *loops*, são usados para repetir um conjunto de instruções até que uma condição específica seja atendida.

Estruturas de decisão direcionam o fluxo de execução com base no resultado de uma condição booleana.

Exposição

Definição e relação entre laços e decisões

© Getty Images

Laços de repetição como o *while* e o *for* dependem fortemente de avaliações condicionais para determinar a continuidade da repetição.

No laço while, a condição é verificada antes de cada iteração, e o loop continua até que a condição seja avaliada como falsa.

Exposição

Definição e relação entre laços e decisões

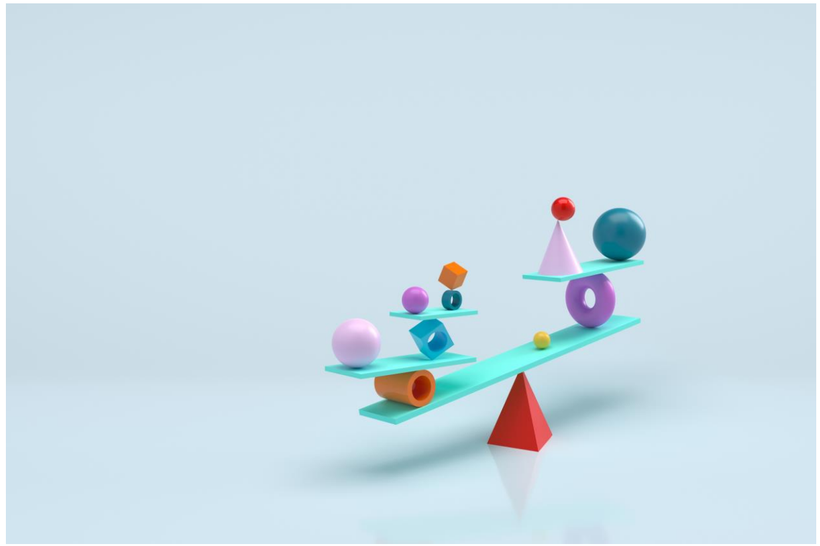
© Getty Images

Estruturas de decisão, como *if*, *else* e *elif*, são usadas para executar blocos de código diferentes, dependendo se uma condição for verdadeira ou falsa. Eles são o mecanismo pelo qual um programa pode tomar decisões e executar diferentes ações em diferentes circunstâncias.

A relação entre laços e decisões é essencial para tarefas como processar entradas até que uma condição válida seja encontrada ou sair de laços baseados em um estado específico alcançado dentro do *loop*. Por exemplo, você pode ter um laço *while* que lê entradas do usuário até que seja fornecido um valor válido, com um *if* interno para verificar a validade e, potencialmente, *break*, para sair do laço.

Exposição

Comparativo entre *while* e outras estruturas de repetição em contexto de decisão



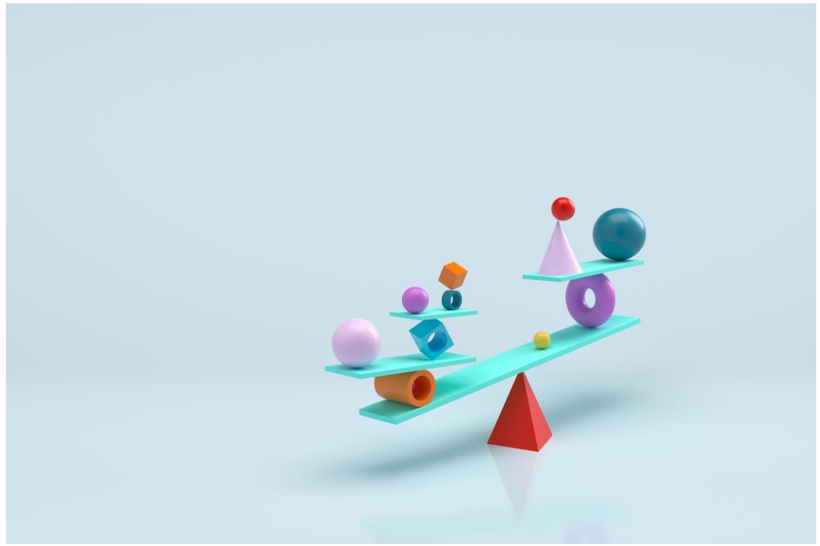
© Getty Images

O laço *while* é tipicamente utilizado quando não se sabe antecipadamente quantas vezes o bloco de código precisa ser repetido, sendo ideal para situações com condições de terminação variáveis. Isso contrasta com o laço *for*, que é mais adequado quando você pode determinar o número de repetições antes de o *loop* começar.

O laço *do-while*, disponível em algumas linguagens como C e C++, é uma variação que garante que o corpo do *loop* seja executado pelo menos uma vez antes de se verificar a condição, o que é útil quando a verificação deve ocorrer após a execução do bloco de código.

Exposição

Comparativo entre *while* e outras estruturas de repetição em contexto de decisão



© Getty Images

A escolha entre essas estruturas de repetição geralmente se resume à natureza da condição de controle e ao contexto em que estão sendo usadas.

Enquanto o *for* é excelente para iterar sobre coleções ou quando o número de iterações é conhecido (como processar todos os elementos em um *array*), o *while* é preferível em cenários em que as iterações podem terminar com base em eventos externos ou condições que mudam dinamicamente durante a execução do laço.

Exposição

Boas práticas na combinação de laços e decisões

Ao combinar laços e estruturas de decisão, as seguintes práticas são recomendadas para manter a clareza e a eficiência do código:

- **Minimize a lógica dentro de laços:** mantenha o corpo do laço tão simples quanto possível. Se a lógica de decisão se torna complexa, considere mover parte dela para uma função separada.

- **Use *break* e continue com cuidado:** embora úteis para controlar o fluxo de execução, o uso excessivo dessas instruções pode tornar o código difícil de se seguir e se manter.

- **Evite modificar o contador ou a condição do laço externamente:** isso pode levar a bugs sutis e difíceis de depurar. Qualquer variável que controle o laço deve ser alterada de maneira previsível e transparente.

Construção de um fluxo de decisão e repetição

Objetivo: proporcionar a aplicação prática dos conceitos de estruturas de decisão e repetição, resultando na criação de um fluxo lógico que envolva ambas as estruturas para solucionar um problema específico.

Motivação: seu *squad* está planejando o desenvolvimento de uma nova aplicação e recebeu a demanda de criar um fluxo de relacionamento entre as estruturas condicionais e de repetição, de acordo com o cenário de um sistema de pizzaria.

Materiais necessários:

- Cadernos ou folhas de papel para cada grupo;
- Canetas.

Formato de entrega: ao término do desenvolvimento, o grupo deverá entregar um fluxograma ilustrativo sobre o processo.

Construção de um fluxo de decisão e repetição

O cenário escolhido é o sistema de atendimento de uma pizzaria online que precisa lidar com uma série de decisões e repetições conforme se segue:

- O sistema recebe pedidos de clientes;
- Para cada pedido, o sistema verifica se todos os itens estão disponíveis;
- Se algum item estiver indisponível, o cliente é informado e deve decidir se quer escolher um substituto ou cancelar o item faltante;
- O sistema repete o processo até que o pedido esteja completo ou o cliente decida não substituir ou cancelar os itens indisponíveis;
- Finalmente, o sistema confirma o pedido e procede com o pagamento.

Vamos fazer uma **atividade**

Construção de um fluxo de decisão e repetição

Tempo estimado: 15 min

1. Use papel e caneta ou uma ferramenta de desenho on-line para criar o fluxograma.
2. Indique onde as estruturas de decisão (*if/else*) atuam (por exemplo, verificação da disponibilidade, decisão do cliente para substituição ou cancelamento).
3. Indique onde os laços de repetição (*while*) são necessários (por exemplo, processamento de pedidos, atualizações do cliente).
4. Discuta em grupo e determine a lógica por trás de cada decisão e como o *loop* deve proceder. Envie a atividade pelo AVA, conforme orientação do professor.

O que nós **aprendemos hoje?**

Hoje desenvolvemos:

1. O relacionamento existente entre fluxos de decisão e fluxos de repetição de códigos.

2. A identificação em um comparativo entre o fluxo *while* e as estruturas de decisão adjacentes, criando um relacionamento entre os dados.

3. A prática do processo de construção de um fluxo de repetição a partir de um cenário real.

Saiba mais

Para aprofundar-se ainda mais em Python e no contexto de estruturas de repetição, recomendamos o conteúdo a seguir:

“Curso Python #014 - Estrutura de repetição while”, do canal CURSO EM VÍDEO. Disponível em: https://www.youtube.com/watch?v=LH6OIn2lBaI. Acesso em: 26 jan. 2024.

Referências da aula

CURSO EM VÍDEO. **Curso Python #014 - Estrutura de repetição while**. Disponível em: https://www.youtube.com/watch?v=LH6OIn2lBaI. Acesso em: 26 jan. 2024.

LOPES, E. Loops e estruturas de repetição no Python. **Python Academy**, 8 jul. 2021. Disponível em: https://pythonacademy.com.br/blog/estruturas-de-repeticao. Acesso em: 26 jan. 2024.

Identidade visual: imagens © Getty Images

Educação Profissional Paulista

Técnico em Desenvolvimento de Sistemas

Secretaria da Educação

SÃO PAULO
GOVERNO DO ESTADO