Operadores

Lógica de Programação I

Operadores e Precedência

Assim como temos os operadores matemáticos da aritmética, álgebra, trigonometría, cálculo etc. O Java define símbolos especiais para realizar operações com nossas variáveis também. Esses operadores podem tomar um, dois ou três termos e agir sobre eles.

- **Aritméticos**: soma (+), subtração (-), multiplicação (*) e divisão (/);
- **Lógicos**: negação (!), E (&&), OU (||) // duplo pipe
- **Relacionais**: maior que (>), menor que (<), igual (==), diferente (≠)
- **Ternário**: ? :

Operadores

Existem algumas variações desses operadores que encurtam as expressões que escrevemos. Por exemplo, ao invés de:

```
int contador = 0;

contador = contador + 1;
```

Podemos utilizar o operador unário ++ na forma **pós-fixada**, nesse cenário, o valor atual do contador é informado, e em seguira, seu valor é adicionado 1.

```
contador++
```

ada

Operadores

A forma pré-fixada, primeiro é realizado o acréscimo e, depois, o novo valor é retornado

```
++contador
```

Operadores

Um operador aritmético especial, e que não foi mencionado, é o *mod* %. Este operador informa o "resto da divisão", por exemplo:

```
int resultado = 3 % 2; // resultado = 1
```

O operador *mod* é, frequentemente, usado para verificar se um número é par ou ímpar.

Precedência

Precedência

A precedência refere-se à ordem em que operações são avaliadas em uma expressão. Em lógica de programação e em muitas linguagens de programação, existem regras claras sobre a ordem em que operadores são aplicados, garantindo que as operações sejam realizadas de maneira consistente.

Precedência

1. **Parênteses:** Operações dentro de parênteses são sempre avaliadas primeiro. Se houver parênteses aninhados, os mais internos serão avaliados primeiro.

2. **Multiplicação, Divisão e Módulo:** Estas operações têm uma precedência mais alta do que adições e subtrações. Assim, elas são avaliadas antes das operações de adição e subtração.

3. **Adição e Subtração:** Se não houver parênteses, as operações de adição e subtração são avaliadas depois das operações de multiplicação, divisão e módulo.

4. **Operadores Relacionais:** São usados para comparar valores e, geralmente, têm uma precedência mais baixa do que operadores aritméticos, mas mais alta do que operadores lógicos.

5. **Operadores Lógicos:** Como **AND, OR, NOT**. Eles têm a precedência mais baixa na maioria das linguagens de programação, mas é sempre uma boa prática usar parênteses para tornar as intenções claras, especialmente em expressões complexas.

Precedência

Resumo

Regras Gerais de Precedência:

1. **Parênteses:** Têm a precedência mais alta; as operações dentro dos parênteses são sempre avaliadas primeiro.

2. **Operações Aritméticas (Multiplicação, Divisão, Módulo):** Têm precedência sobre adições e subtrações.

3. **Adições e Subtrações:** São avaliadas depois das operações aritméticas.

4. **Operadores Relacionais e Lógicos:** Têm uma precedência específica para comparações e operações booleanas.

Exercícios

Exercício 1: Controle de Acesso Múltiplo

Para que um usuário tenha acesso a um sistema, ele deve ser ativo, ter nível de permissão acima de 7 e não ter pendências financeiras.

Qual das alternativas representa corretamente essa condição?

a) *usuarioAtivo* && nivelPermissao > 7 && !pendenciasFinanceiras
b) *usuarioAtivo* || nivelPermissao > 7 && !pendenciasFinanceiras
c) *usuarioAtivo* && nivelPermissao > 7 || !pendenciasFinanceiras
d) *usuarioAtivo* && nivelPermissao >= 7 && !pendenciasFinanceiras

Exercício 2: Cálculo de Desconto Complexo

Um cliente recebe um desconto de 20% se comprar a partir 20 itens e seu valor total exceder a partir de 1000. Qual das alternativas representa corretamente essa condição?

Qual das alternativas representa corretamente essa condição?

a) *numItens* > 20 && *valorTotal* > 1000
b) *numItens* >= 20 && *valorTotal* >= 1000
c) *numItens* > 20 || *valorTotal* > 1000
d) *numItens* >= 20 || *valorTotal* >= 1000

Exercício 3: Critérios de Promoção

Para um produto ser elegível para promoção, ele deve estar disponível em estoque e ter sido lançado nos últimos 6 meses.

Qual das alternativas representa corretamente essa condição?

a) `emEstoque && lancamentoRecente`
b) `!emEstoque || !lancamentoRecente`
c) `emEstoque || lancamentoRecente`
d) `emEstoque && !lancamentoRecente`

Exercício 4: Combinação de Requisitos

Um cliente recebe frete grátis se ele for um membro premium ou se sua compra exceder 500 e o endereço estiver dentro do país.

Qual das alternativas representa corretamente essa condição?

a) `isMembroPremium || valorCompra > 500 || enderecoNacional`
b) `isMembroPremium && valorCompra > 500 || enderecoNacional`
c) `isMembroPremium || valorCompra > 500 && enderecoNacional`
d) `isMembroPremium && valorCompra > 500 && enderecoNacional`

Exercício 5: Avaliação de Acesso Especial

Para um usuário ter acesso especial a recursos premium, ele deve ter uma assinatura ativa e pelo menos 50 pontos de reputação.

Qual das alternativas representa corretamente essa condição?

a) `assinaturaAtiva && pontoReputacao >= 50`
b) `assinaturaAtiva || pontoReputacao >= 50`
c) `assinaturaAtiva || pontoReputacao > 50`
d) `assinaturaAtiva && pontoReputacao > 50`

Exercício 6:Verificar o Maior de Três Números

Escreva um programa que aceite três números inteiros e determine o maior entre eles **usando apenas o operador ternário.**

Exercício 7: Determinar a Categoria de um Atleta com Base na Idade

Suponha que você queira determinar a categoria de um atleta com base em sua idade:

- Se a idade for inferior a 18, a categoria é "Juvenil".
- Se a idade estiver entre 18 e 40 (inclusive), a categoria é "Adulto".
- Se a idade for superior a 40, a categoria é "Master".

Usar apenas o operador ternário.

Exercício 8: Verificar se um Ano é Bissexto

Um ano é considerado bissexto se:

- É divisível por 4, mas não por 100, ou
- É divisível por 400.

Escreva um programa que determine se um ano é bissexto ou não, **usando apenas o operador ternário.**

Obrigada