Prova de Fluxo de Controle em Dart

01) O que é Fluxo de Controle em Dart?

Fluxo de controle é uma técnica de programação que permite controlar a ordem em que as instruções são executadas em um programa de computador. No contexto dos pipelines de dados, o fluxo de controle garante o processamento ordenado de um conjunto de tarefas.

02) Quais são as três estruturas básicas de Fluxo de Controle em Dart?

As três estruturas básicas de fluxo de controle em Dart são as estruturas condicionais e de repetição. [As estruturas condicionais permitem executar um trecho de código caso uma expressão lógica seja verdadeira ou ignorá-lo se ela for falsa](https://www.devmedia.com.br/linguagem-dart-controle-de-fluxo/40758). [As estruturas de repetição permitem executar um bloco de código várias vezes](https://www.treinaweb.com.br/blog/estruturas-condicionais-e-de-repeticao-no-dart). As três estruturas básicas são:

if/else

switch/case

while/for

03) O que é uma instrução if em Dart?

A instrução `if` em Dart é usada para executar um bloco de código se uma condição for verdadeira. A sintaxe básica é a seguinte:

```

if (condição) {

// código a ser executado se a condição for verdadeira

}

```

Você também pode usar a instrução `if-else` para executar um bloco de código se a condição for verdadeira e outro bloco de código se a condição for falsa. A sintaxe básica é a seguinte:

```

if (condição) {

// código a ser executado se a condição for verdadeira

} else {

// código a ser executado se a condição for falsa

}

```

04) O que é uma instrução if-else em Dart?

A instrução if-else é uma estrutura condicional que permite executar um trecho de código caso uma expressão lógica seja verdadeira ou ignorá-lo se ela for falsa. Em Dart, a sintaxe da instrução if-else é a seguinte:

```

if (expressão lógica) {

// código a ser executado se a expressão for verdadeira

} else {

// código a ser executado se a expressão for falsa

}

```

Por exemplo, o seguinte código pede para o usuário digitar um número inteiro e verifica se ele é maior que 10:

```

import 'dart:io';

void main() {

stdout.write("Digite um número inteiro: ");

int valor = int.parse(stdin.readLineSync());

if (valor > 10) {

print("O valor é maior que 10.");

} else {

print("O valor não é maior que 10.");

}

}

```

05) O que é uma instrução switch em Dart?

A instrução switch é uma estrutura condicional que permite executar um trecho de código diferente dependendo do valor de uma variável. Em Dart, a sintaxe da instrução switch é a seguinte:

```

switch (variável) {

case valor1:

// código a ser executado se a variável for igual a valor1

break;

case valor2:

// código a ser executado se a variável for igual a valor2

break;

default:

// código a ser executado se a variável não for igual a nenhum dos valores anteriores

}

```

Por exemplo, o seguinte código pede para o usuário digitar um número inteiro e verifica se ele é maior que 10 usando uma instrução switch:

```

import 'dart:io';

void main() {

stdout.write("Digite um número inteiro: ");

int valor = int.parse(stdin.readLineSync());

switch (valor) {

case 1:

print("O valor é igual a 1.");

break;

case 2:

print("O valor é igual a 2.");

break;

default:

print("O valor não é igual a 1 nem a 2.");

}

}

```

06) Como usar a instrução switch em Dart?

Para usar a instrução switch em Dart, você precisa seguir a seguinte sintaxe:

switch (variável) {

case valor1:

// código a ser executado se a variável for igual a valor1

break;

case valor2:

// código a ser executado se a variável for igual a valor2

break;

default:

// código a ser executado se a variável não for igual a nenhum dos valores anteriores

}

Aqui está um exemplo de como usar a instrução switch em Dart:

void main() {

var opcao = 'A';

switch (opcao) {

case 'A':

print('Opção A selecionada.');

break;

case 'B':

print('Opção B selecionada.');

break;

default:

print('Opção inválida.');

}

}

Neste exemplo, o programa verifica o valor da variável opcao e executa o código correspondente ao caso correspondente.

07) O que é uma instrução for em Dart?

A instrução `for` em Dart é usada para executar um bloco de código várias vezes. A sintaxe básica é a seguinte:

```

for (inicializador; condição; iterador) {

// código a ser executado

}

```

O `inicializador` é executado uma vez antes do loop começar e é usado para inicializar uma variável de controle. A `condição` é verificada antes de cada iteração do loop e o loop é interrompido se a condição for falsa. O `iterador` é executado após cada iteração do loop e é usado para atualizar a variável de controle.

Você também pode usar a instrução `for-in` para iterar sobre uma coleção de objetos. A sintaxe básica é a seguinte:

```

for (variável in coleção) {

// código a ser executado

}

```

08) Como usar a instrução for em Dart?

A instrução for é uma estrutura de repetição que permite executar um trecho de código várias vezes. Em Dart, a sintaxe da instrução for é a seguinte:

```

for (inicialização; condição; incremento) {

// código a ser executado

}

```

Aqui está um exemplo de como usar a instrução for em Dart:

```

void main() {

for (var i = 0; i < 5; i++) {

print('O valor de i é $i');

}

}

```

Neste exemplo, o programa executa o código dentro do bloco de instruções cinco vezes, incrementando o valor da variável `i` em cada iteração.

09) O que é uma instrução while em Dart?

A instrução `while` em Dart é usada para executar um bloco de código enquanto uma condição for verdadeira. A sintaxe básica é a seguinte:

```

while (condição) {

// código a ser executado

}

```

O bloco de código é executado enquanto a condição for verdadeira. A condição é verificada antes de cada iteração do loop e o loop é interrompido quando a condição se torna falsa.

Você também pode usar a instrução `do-while` para executar um bloco de código pelo menos uma vez e depois repeti-lo enquanto uma condição for verdadeira. A sintaxe básica é a seguinte:

```

do {

// código a ser executado

} while (condição);

```

O bloco de código é executado pelo menos uma vez e depois repetido enquanto a condição for verdadeira.

10) Como usar a instrução while em Dart?

A instrução while é uma estrutura de repetição que permite executar um trecho de código várias vezes enquanto uma condição for verdadeira. Em Dart, a sintaxe da instrução while é a seguinte:

```

while (condição) {

// código a ser executado

}

```

Aqui está um exemplo de como usar a instrução while em Dart:

```

void main() {

var i = 0;

while (i < 5) {

print('O valor de i é $i');

i++;

}

}

```

Neste exemplo, o programa executa o código dentro do bloco de instruções enquanto a variável `i` for menor que 5.

11) O que é uma instrução do-while em Dart?

A instrução `do-while` em Dart é usada para executar um bloco de código pelo menos uma vez e depois repeti-lo enquanto uma condição for verdadeira. A sintaxe básica é a seguinte:

```

do {

// código a ser executado

} while (condição);

```

O bloco de código é executado pelo menos uma vez e depois repetido enquanto a condição for verdadeira. A diferença entre o laço `do-while` e o laço `while` é que o primeiro testa a condição de continuidade no final, enquanto o segundo testa a condição de continuidade no início.

12) Como usar a instrução do-while em Dart?

A instrução do-while é uma estrutura de repetição que permite executar um trecho de código várias vezes enquanto uma condição for verdadeira. Em Dart, a sintaxe da instrução do-while é a seguinte:

```

do {

// código a ser executado

} while (condição);

```

Aqui está um exemplo de como usar a instrução do-while em Dart:

```

void main() {

var i = 0;

do {

print('O valor de i é $i');

i++;

} while (i < 5);

}

```

Neste exemplo, o programa executa o código dentro do bloco de instruções pelo menos uma vez e depois verifica se a condição é verdadeira. Se a condição for verdadeira, o programa executa o bloco de instruções novamente.

13) O que é uma instrução break em Dart?

A instrução `break` em Dart é usada para tirar o controle de uma construção, como um loop ou uma instrução `switch`. Usando `break` em um loop faz com que o programa saia do loop. A sintaxe básica é a seguinte:

```

while (condição) {

// código a ser executado

if (condição) {

break;

}

}

```

O código dentro do loop é executado enquanto a condição for verdadeira. Se a condição dentro do loop for verdadeira, o programa sairá do loop usando a instrução `break`.

14) Como usar a instrução break em Dart?

A instrução break é usada para interromper a execução de um loop ou switch statement em Dart. Quando o programa encontra uma instrução break dentro de um loop ou switch statement, ele sai do loop ou switch statement imediatamente.

Aqui está um exemplo de como usar a instrução break em Dart:

```

void main() {

for (var i = 0; i < 10; i++) {

if (i == 5) {

break;

}

print(i);

}

}

```

Neste exemplo, o programa executa o loop for e imprime os valores de i até que i seja igual a 5. Quando i é igual a 5, o programa encontra a instrução break e sai do loop.

15) O que é uma instrução continue em Dart?

A instrução `continue` em Dart é usada para pular a iteração atual do loop e continuar com a próxima iteração. A sintaxe básica é a seguinte:

```

while (condição) {

// código a ser executado

if (condição) {

continue;

}

// código a ser executado

}

```

O código dentro do loop é executado enquanto a condição for verdadeira. Se a condição dentro do loop for verdadeira, o programa pulará a iteração atual e continuará com a próxima iteração usando a instrução `continue`.

16) Como usar a instrução continue em Dart?

A instrução continue é usada para pular a execução de uma iteração de um loop em Dart. Quando o programa encontra uma instrução continue dentro de um loop, ele pula a execução da iteração atual e passa para a próxima iteração.

Aqui está um exemplo de como usar a instrução continue em Dart:

```

void main() {

for (var i = 0; i < 10; i++) {

if (i == 5) {

continue;

}

print(i);

}

}

```

Neste exemplo, o programa executa o loop for e imprime os valores de i até que i seja igual a 5. Quando i é igual a 5, o programa encontra a instrução continue e pula a execução da iteração atual. Em seguida, ele passa para a próxima iteração e continua imprimindo os valores de i.

17) O que é uma instrução try-catch em Dart?

A instrução `try-catch` em Dart é usada para lidar com exceções em tempo de execução. O bloco `try` contém o código que pode lançar uma exceção e o bloco `catch` contém o código que lida com a exceção lançada.

A sintaxe básica é a seguinte:

```

try {

// código que pode lançar uma exceção

} catch (e) {

// código que lida com a exceção

}

```

Se uma exceção for lançada dentro do bloco `try`, o programa saltará imediatamente para o bloco `catch`, onde o código de tratamento de exceção será executado.

18) Como usar a instrução try-catch em Dart?

A instrução try-catch é usada para lidar com exceções em Dart. Quando você envolve um bloco de código com uma instrução try-catch, você pode capturar e lidar com exceções que podem ocorrer durante a execução do bloco de código.

Aqui está um exemplo de como usar a instrução try-catch em Dart:

```

void main() {

try {

// Código que pode lançar uma exceção

} catch (e) {

// Código que lida com a exceção

}

}

```

Neste exemplo, o programa executa o bloco de código dentro da instrução try. Se ocorrer uma exceção durante a execução do bloco de código, o programa pula para a instrução catch e executa o código dentro dela.

19) O que é uma instrução finally em Dart?

A instrução `finally` é usada em conjunto com a instrução `try-catch`. O bloco `finally` contém o código que é executado independentemente de ocorrer ou não uma exceção.

A sintaxe básica é a seguinte:

```

try {

// código que pode lançar uma exceção

} catch (e) {

// código que lida com a exceção

} finally {

// código que é executado independentemente de ocorrer ou não uma exceção

}

```

O bloco `finally` é opcional e pode ser omitido se não for necessário.

20) Como usar a instrução finally em Dart?

A instrução try-catch é usada para lidar com exceções em Dart. Quando você envolve um bloco de código com uma instrução try-catch, você pode capturar e lidar com exceções que podem ocorrer durante a execução do bloco de código.

Aqui está um exemplo de como usar a instrução try-catch em Dart:

```

void main() {

try {

// Código que pode lançar uma exceção

} catch (e) {

// Código que lida com a exceção

}

}

```

Neste exemplo, o programa executa o bloco de código dentro da instrução try. Se ocorrer uma exceção durante a execução do bloco de código, o programa pula para a instrução catch e executa o código dentro dela.