01) O que é Fluxo de Controle em Dart?

Fluxo de controle em Dart se refere à ordem em que as instruções são executadas em um programa. É a maneira de controlar o fluxo de execução do programa com base em determinadas condições.

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

As três estruturas básicas de Fluxo de Controle em Dart são: instruções condicionais (if-else, switch), loops (for, while, do-while) e tratamento de exceções (try-catch-finally).

03) O que é uma instrução if em Dart?

A instrução if em Dart é uma instrução condicional que executa um bloco de código somente se a condição especificada for verdadeira.

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

A instrução if-else em Dart é uma instrução condicional que executa um bloco de código se a condição especificada for verdadeira e outro bloco de código se a condição for falsa.

05) O que é uma instrução switch em Dart?

A instrução switch em Dart é uma estrutura condicional que executa um bloco de código diferente para cada valor possível de uma expressão.

06) Como usar a instrução switch em Dart?

A instrução switch em Dart é usada da seguinte maneira:

```
switch (expressao) {
  case valor1:
    // bloco de código para valor1
    break;
  case valor2:
    // bloco de código para valor2
    break;
  default:
    // bloco de código padrão
}
```

07) O que é uma instrução for em Dart?

A instrução for em Dart é uma estrutura de repetição que executa um bloco de código um número fixo de vezes, com base em uma condição especificada.

08) Como usar a instrução for em Dart?

A instrução for em Dart é usada da seguinte maneira:

```
for (inicializacao; condicao; incremento) {
  // bloco de codigo
}
```

09) O que é uma instrução while em Dart?

A instrução while em Dart é uma estrutura de repetição que executa um bloco de código enquanto uma condição especificada for verdadeira.

10) Como usar a instrução while em Dart?

A instrução while em Dart é usada da seguinte maneira:

```
while (condicao) {
  // bloco de codigo
}
```

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

A instrução do-while em Dart é uma estrutura de repetição que executa um bloco de código pelo menos uma vez e continua a executar o bloco enquanto uma condição especificada for verdadeira.

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

A instrução do-while em Dart é usada da seguinte maneira:

```
do {
  // bloco de codigo
} while (condicao);
```

13) O que é uma instrução break em Dart?

A instrução break em Dart é usada para encerrar um loop ou switch antes que a condição normal de saída seja atingida.

14) Como usar a instrução break em Dart?

A instrução break em Dart é usada da seguinte maneira:

```
while (condicao) {
  if (algumaCondicao) {
    break; // encerra o loop
  }
  // bloco de codigo
}
```

15) O que é uma instrução continue em Dart?

A instrução continue em Dart é usada para pular o restante do bloco de código atual e continuar a execução no próximo loop.

16) Como usar a instrução continue em Dart?

A instrução continue em Dart é usada da seguinte maneira:

```
while (condicao) {
  if (algumaCondicao) {
    continue; // pula o resto do loop atual e vai para o próximo loop
  }
  // bloco de codigo
}
```

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

A instrução try-catch em Dart é usada para lidar com exceções ou erros que podem ocorrer durante a execução do programa.

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

A instrução try-catch em Dart é usada da seguinte maneira:

```
try {
  // código que pode gerar uma exceção
} catch (e) {
  // código para tratar a exceção
}
```

19) O que é uma instrução finally em Dart?

A instrução finally em Dart é usada em conjunto com a instrução try-catch, para executar um bloco de código independentemente de ter ocorrido ou não uma exceção. O bloco finally é executado sempre, mesmo que uma exceção seja lançada e capturada pelo bloco

catch.

20) Como usar a instrução finally em Dart?

Para usar a instrução finally em Dart, basta adicioná-la após o bloco catch em uma instrução try-catch. Exemplo:

```
try {
  // código que pode gerar uma exceção
} catch (e) {
  // código para tratar a exceção
} finally {
  // código que será executado sempre
}
```