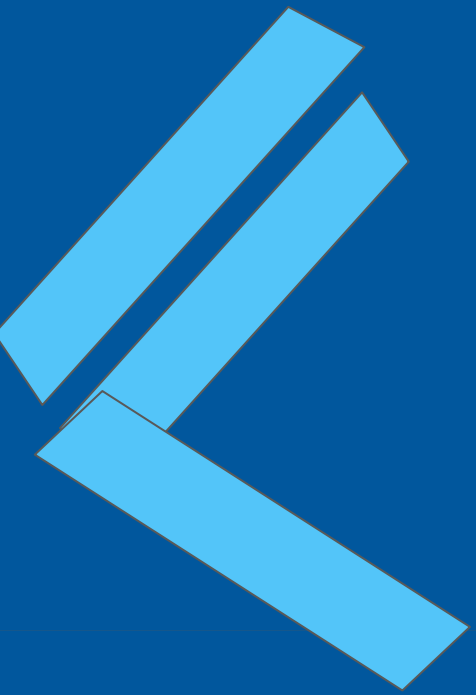


Desenvolvimento Mobile 1
Aula 03

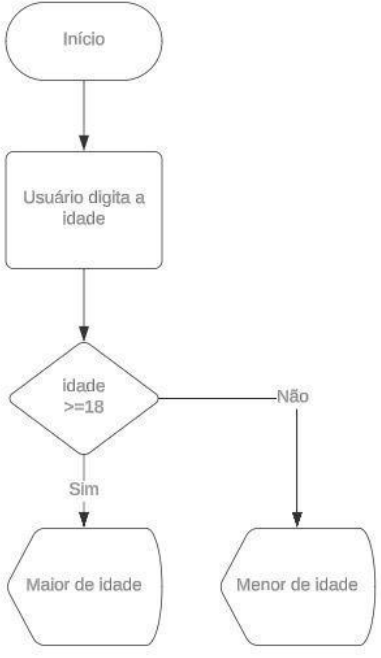
Prof. Me Daniel Vieira

SENAI

Agenda

1- Estrutura condicional
2-Estrutura de repetição
3 - Funções
4- Exercícios

Estrutura condicional




```
1. // Declaração da variável idade com o valor 18
2. int idade = 18;
3.
4. // Estrutura condicional para verificar se a idade é maior ou igual a 18 e exibe a
mensagem no terminal.
5. //Se a condição for falsa vai para o else e exibe a mensagem menor de idade.
6. if (idade >= 18) {
7.   print('Adulto');
8. } else {
9.   print('Menor de idade');
10. }
11.
```

Estrutura condicional switch case

Outra estrutura condicional é a switch case que realiza a comparação de uma variável com diferentes valores e para no ponto em que a condição é atingida

- switch / case

O switch-case realiza uma comparação direta de uma variável com valores constantes ou pré-definidos. Ele para no momento em que uma condição é atingida e executa o código correspondente.

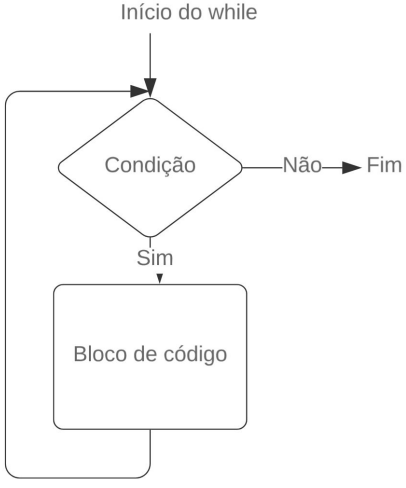
Características principais

- Ideal para situações em que uma variável precisa ser comparada com diversos valores.
- É mais organizado e legível do que usar várias condições if-else.
- Contém um caso padrão (default) que será executado quando nenhuma das condições anteriores for atendida.

Estrutura condicional switch case

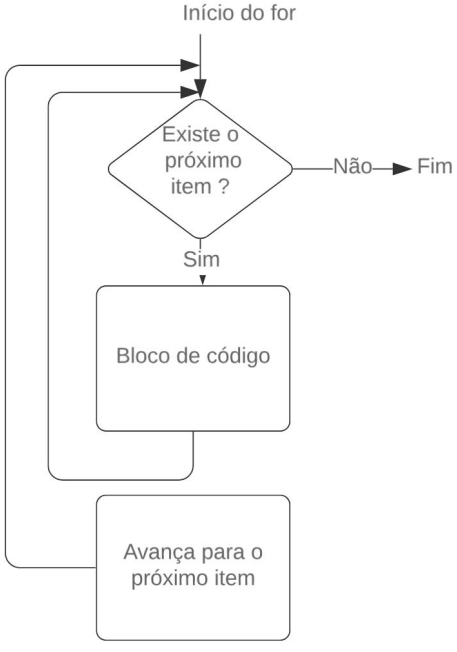
```
1. // Declaração da variável cor com o vermelho
2. String cor = 'vermelho';
3.
4. /*Estrutura condicional que compara se a variável cor com diferentes valores e exibe
mensagem no terminal.
5. switch (variável) {
6.    case valor1:
7.      // Código a ser executado quando variável == valor1
8.      break; // Finaliza a execução do switch
9.    case valor2:
10.     // Código a ser executado quando variável == valor2
11.     break;
12.   default:
13.     // Código a ser executado quando nenhum caso é atendido
14. }
15. O switch avalia a expressão (cor no exemplo) e compara seu valor com os casos definidos
(case).
16. Quando um case é satisfeito, o bloco associado é executado.
17. O break é usado para evitar que outros casos sejam executados após um case ser
satisfeito.
18. O default é executado quando nenhum dos casos é atendido.
19. */
20. switch (cor) {
21.
22.   case 'vermelho':
23.     print('Pare!');
24.     break;
25.   case 'verde':
26.     print('Siga!');
27.     break;
28.   default:
29.     print('Atenção!');
30. }
31.
32.
```

Estrutura de repetição - while



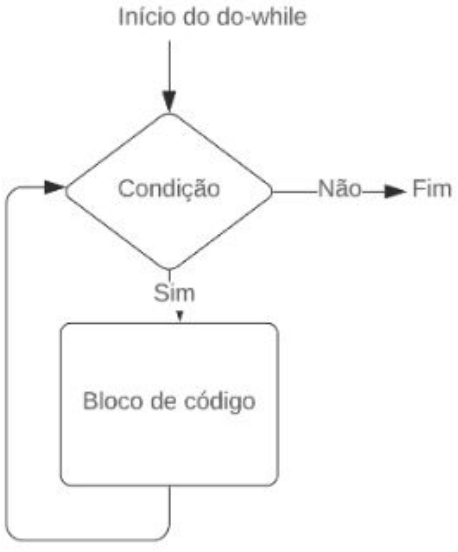

```
1. //
2. Estrutura while(condição){
3. bloco de código
4. altera condição
5. }
6.
7. while
8. int contador = 0;
9. while (contador < 5) {
10.   print(contador);
11.   contador++;
12. }
13.
```

Estrutura de repetição - for




```dart
/* Estrutura for
for(inicialização da variável, condição, incremento)

Estrutura for
for(inicialização da variável, condição, incremento)
*/

Código for
for (int i = 0; i < 5; i++) {

  print(i); // 0, 1, 2, 3, 4
}

```

Estrutura de repetição - do while




```
1. /*Estrutura do
2. {
3. bloco de código
4. altera condição
5. }while(condição){
6.
7. }
8. */
9. Código do-while
10. int contador = 0;
11. do {
12.   print(contador);
13.   contador++;
14. } while (contador < 5);
15.
```

Funções

No contexto da programação, uma função é um bloco de código projetado para realizar uma tarefa específica. Ela encapsula uma lógica que pode ser reutilizada em diferentes partes do programa, tornando o código mais organizado, modular e manutenível. As funções desempenham um papel fundamental no desenvolvimento de aplicativos móveis, especialmente em frameworks como Flutter, onde funções são amplamente usadas na manipulação de dados e na construção de interfaces.

Características das Funções

- Reutilização: Uma vez definida, uma função pode ser chamada várias vezes, evitando duplicação de código.
- Modularidade: Funções ajudam a dividir o programa em partes menores e mais gerenciáveis.
- Legibilidade: Nomear funções de forma descritiva torna o código mais fácil de entender.
- Manutenção: Alterações em uma função afetam todas as suas chamadas, facilitando atualizações e correções.

Funções

```
/* tipoRetorno:
 Especifica o tipo do valor retornado pela função
 (por exemplo, int, String, void). Se a função não retorna nada, usamos void.
nomeDaFuncao:

Nome único que identifica a função.
parâmetros: Valores que a função recebe para realizar sua tarefa (opcional).

return: Usado para retornar um valor da função (opcional).
*/
tipoRetorno nomeDaFuncao(parâmetros)
 {
 // Corpo da função return valor;
 // Opcional
}

```

Funções

```dart
/*Função sem retorno

Usadas para executar uma tarefa sem retornar um valor.
*/


void saudar(String nome)
{
 print("Olá, $nome! Bem-vindo ao Flutter.");
 }

 void main() {
saudar("Daniel");
 }

/*
Função com retorno
Usadas quando precisamos de um valor como resultado da função.
*/


int somar(int a, int b) {
return a + b;
 }
void main() {
 int resultado = somar(5, 10);
print("A soma é: $resultado"); // Saída: A soma é: 15
}

```

Funções

```
/*
Função com passagem de parâmetros opcionais
Dart permite declarar parâmetros opcionais, que podem ter valores padrão.
*/
void exibirMensagem(String mensagem, [String remetente = "Anônimo"]) {
  print("Mensagem de $remetente: $messagem");
}

void main() {
  exibirMensagem("Bem-vindo ao curso!"); // Remetente: Anônimo
  exibirMensagem("Parabéns pelo progresso!", "Professor");
}

```

Funções

```
/*

Função com passagem de parâmetros nomeados

Os parâmetros podem ser nomeados, tornando o código mais legível.
*/

void criarUsuario({required String nome, int idade = 18}) {
  print("Usuário: $nome, Idade: $idade");
}

void main() {
  criarUsuario(nome: "Ana");
  criarUsuario(nome: "Carlos", idade: 25);
}


/*

Função anônima
Funções sem nome são úteis quando precisamos de lógica temporária.

*/
(){
print(“Desenvolvimento Mobile”)
}

```

Funções

```
77. /*
78. Função assíncrona
79. Usadas para operações que levam tempo para serem concluídas, como chamadas de
APIs.
80. */
81.
82. Future<void> carregarDados() async {
83.   print("Carregando...");
84.   await Future.delayed(Duration(seconds: 2));
85.   print("Dados carregados!");
86. }
87.
88. void main() async {
89.   await carregarDados();
90. }
91.
92.
```

Benefícios de se utilizar funções

Usando Funções em Aplicações Móveis

As funções são frequentemente usadas para:

- Manipular dados exibidos na interface do usuário.
- Realizar cálculos complexos.
- Chamar APIs para buscar ou enviar dados.
- Modularizar a construção de widgets no Flutter.

Os principais benefícios de se utilizar funções no desenvolvimento de código são:

- Benefícios de Funções Bem Estruturadas
- Reutilização de Código: Evita duplicação e simplifica alterações.
- Manutenção Facilitada: Alterações em uma função afetam todas as suas chamadas.
- Melhor Desempenho: Funções assíncronas otimizam operações demoradas.
- Colaboração: Funções bem nomeadas e documentadas facilitam o trabalho em equipe.

Exercícios

1. Crie uma função que receba as informações de um usuário digitado pelo teclado: Nome, Curso, Idade.

2. Crie uma função que calcule a área de um triângulo a partir de dados digitados pelo usuário. A = (b* h)/2 e retorne esse valor.

3. Crie uma função que calcule o salário líquido do usuário a partir dos valores digitados pelo teclado considerando um desconto de 10% de impostos e bonificação de 20% em cima do salário.

4. Crie um programa que converta valores de reais (R$) para outras moedas de acordo com a escolha do usuário: euro (EUR), dólar (USD), franco suíço (CHF).
Considere o valor do Euro 7,00
Dólar 6,56
Franco Suiço 4,35

Exercícios

5. Crie um programa que receba a nota de dois alunos, calcule sua média e informe se o aluno está aprovado ou reprovado conforme a média.
Se média for maior ou igual a 7: Aprovado.
Maior ou igual a 4 e menor do que 7: Exame.
Menor do que 4: Reprovado.

6. Crie um programa que receba a idade de duas pessoas e print na tela qual é a pessoa mais velha.

7. Crie um programa que receba o valor médio de três modelos de carro e indique qual é mais caro e o mais barato.

Exercícios

8. Um posto de combustíveis oferece **descontos variados** com base no tipo de combustível adquirido e na quantidade comprada. O desconto é aplicado diretamente no valor total, e as condições específicas para cada tipo de combustível devem ser consideradas.

Escreva um programa que:

1. Solicite ao usuário:
- A **quantidade de litros** comprada.
- O **tipo de combustível**, sendo:
    - **E** para etanol,
    - **D** para diesel,
    - **G** para gasolina.

1. Calcule:
- O **valor do desconto** utilizando a fórmula:
  desconto = preço do litro × quantidade de litros × percentual de desconto.
- O **valor total a ser pago** utilizando a fórmula:
  valor total = (preço do litro × quantidade de litros) - desconto.

1. Exiba o valor a ser pago pelo cliente.

Observações:

- O programa deve tratar corretamente os diferentes tipos de combustíveis.
- A fórmula do desconto depende do preço do litro e do percentual aplicável para cada combustível.

Exercícios

Tabela 21: Valor dos combustíveis e desconto

Tipo de combustível	Valor	Desconto
Etanol	R$ 1,70	Compra >= 15L Desconto de 4% por litro Compra < 15L Desconto de 3% por litro
Diesel	R$ 2,00	Compra >= 15L Desconto de 5% por litro Compra < 15L por litro Desconto de 3% por litro
Gasolina	R$ 4,50	Compra >= 20L Desconto de 3% por litro Compra < 20L sem desconto

Exercícios

9. Escreva um programa que calcule o preço a pagar pelo fornecimento de energia elétrica.
   - Pergunte para o usuário a quantidade de kWh consumida e o tipo de instalação: Residência (R), Indústrias (I), Comércio (C).
   - Calcule o preço da energia com base na tabela a seguir.
   - O preço a pagar pelo fornecimento da energia elétrica deve ser calculado preço unitário do KWh * quantidade de KWh inserido pelo usuário

Tipo	Faixa(KWh)	Preço (R$)
Residencial	Até 500	0,50
	Acima de 500	0,70
Comercial	Até 1000	0,65
	Acima de 1000	0,60
Industrial	Até 5000	0,55
	Acima de 5000	0,50

Exercícios

- 10. Crie um programa que receba 4 valores numéricos digitados pelo usuário. Em seguida, permita que o usuário escolha a operação que deseja realizar entre as seguintes opções: soma (+), subtração (-), multiplicação (*) e divisão (/).

A entrega dos exercícios deverá ser via Link do repositório GitHub com os códigos adicionado ao TEAMS

Obrigado!

Prof. Me Daniel Vieira

Email: danielvieira2006@gmail.com

Linkedin: Daniel Vieira

Instagram: Prof daniel.vieira95