← Voltar ao curso

CURSO BÁSICO

Capítulo 4 - Variáveis, constantes e tipos de dados

Agora começamos a guardar informações dentro do programa para que a ESP32 possa usá-las nas decisões da ferrovia.

Introdução	Um programa precisa guardar informações: número de um pino, posição de um desvio, estado de uma rota, quantidade de passagens ou um valor de velocidade. Neste capítulo veremos como representar essas informações de forma simples.
Objetivo	Compreender o que são variáveis, constantes e alguns tipos básicos de dados usados em programas para ESP32.
Meta de aprendizagem	Ao final, você deverá reconhecer quando um valor pode mudar, quando deve permanecer fixo e escolher um tipo de dado simples para representar informações da ferrovia.

1. O programa também precisa de memória

No capítulo anterior vimos que a ESP32 executa instruções. Para tomar decisões, porém, ela também precisa guardar valores durante a execução do programa.

Pense em situações da ferrovia: um desvio pode estar em uma posição ou outra; uma rota pode estar liberada ou bloqueada; um contador pode registrar quantas vezes uma composição passou por determinado ponto.

Essas informações podem ser representadas no programa por variáveis e constantes.

2. O que é uma variável?

Uma variável é um espaço usado pelo programa para guardar um valor que pode mudar. Ela recebe um nome para que possamos utilizá-la depois.

> **Exemplo ferroviário:** uma variável chamada `trainCount` pode guardar a quantidade de passagens registradas. Seu valor pode começar em 0 e depois mudar para 1, 2, 3 e assim por diante.

```
int trainCount = 0;
```

Nessa linha:

- `int` informa o tipo de dado;
- `trainCount` é o nome escolhido para a variável;
- `= 0` atribui o valor inicial;
- `;` encerra a instrução.

3. O que é uma constante?

Uma constante guarda um valor que não deverá ser alterado durante a execução do programa. Isso é útil para informações fixas.

Por exemplo, se futuramente decidirmos que determinado sensor está ligado a um pino específico da ESP32, esse número poderá ser representado por uma constante.

```
const int sensorS1Pin = 25;
```

A palavra `const` informa que esse valor deve permanecer fixo. Se o programa tentar alterá-lo depois, a compilação indicará um erro.

4. Por que usar nomes claros?

Um programa pode funcionar usando nomes muito curtos, mas fica mais difícil de entender. Por isso, neste curso usaremos nomes em inglês, porém claros e relacionados à função de cada informação.

Nome	O que representa
`sensorS1Pin`	Pino associado ao sensor S1.
`routeReady`	Indica se uma rota está pronta para uso.
`trainCount`	Quantidade de passagens registradas.
`speedLimit`	Um valor de limite de velocidade usado pelo programa.

5. Tipos de dados que usaremos primeiro

O tipo de dado informa ao programa que espécie de valor será guardada. Não precisamos estudar todos os tipos do C++ agora. Quatro exemplos já são suficientes para começar.

Tipo	Guarda	Exemplo na ferrovia
`int`	Números inteiros.	Número de um pino ou contador de passagens.
`bool`	Apenas dois estados: `true` ou `false`.	Rota liberada ou não liberada.
`float`	Números que podem ter casas decimais.	Um valor numérico de velocidade ou outro ajuste que precise de decimal.
`char`	Um único caractere.	Uma letra usada como identificação simples.

6. O tipo `bool`: verdadeiro ou falso

O tipo `bool` será muito útil na automação porque diversas situações possuem apenas dois estados lógicos.

```
bool routeReady = false;
```

Nesse exemplo, a variável `routeReady` começa com o valor `false`, isto é, falso. Mais adiante, depois de verificações de segurança, o programa poderá mudar seu valor para `true`.

Essa ideia será importante quando começarmos a trabalhar com sensores, desvios e rotas.

7. Uma variável pode receber um novo valor

Depois de criada, uma variável pode mudar de valor. Não é necessário repetir o tipo na hora da alteração.

```
int trainCount = 0;

trainCount = 1;
```

A primeira linha cria a variável. A segunda apenas altera o valor já guardado nela.

8. Um pequeno programa para observar a estrutura

O exemplo abaixo reúne variáveis e constantes sem acionar nenhum componente. Ele serve apenas para praticar a escrita e a compilação.

```
const int sensorS1Pin = 25;
int trainCount = 0;
bool routeReady = false;
float speedLimit = 35.0;

void setup() {
  trainCount = 1;
  routeReady = true;
}
```

```
void loop() {
  // Nos próximos capítulos colocaremos aqui
  // as ações que serão repetidas pela ESP32.
}
```

Observe que os valores foram declarados antes de `setup()`. Assim, poderão ser usados pelas duas partes principais do programa quando isso for necessário.

9. Não precisamos decorar tudo

Neste ponto, o mais importante é compreender a lógica: antes de guardar uma informação, damos um nome a ela e informamos que tipo de valor será armazenado.

Com a prática, declarações como `int`, `bool` e `const int` passarão a ser reconhecidas naturalmente.

EXERCÍCIO PRÁTICO

Meta: criar e compilar um programa simples contendo variáveis e constantes relacionadas à ferrovia.

1. Abra a Arduino IDE.
2. Crie um novo programa.
3. Digite o programa apresentado no item 8.
4. Salve como `basico-04-variaveis`.
5. Clique em Verificar/Compilar.
6. Altere `trainCount` de 0 para 2 e compile novamente.
7. Altere `routeReady` de `false` para `true` e compile novamente.
8. Depois, tente alterar `sensorS1Pin` dentro de `setup()`. Observe o erro indicado pela compilação e desfaça essa alteração.

Resultado esperado: reconhecer a diferença entre um valor variável e um valor constante e conseguir compilar corretamente o programa depois de restaurar a constante.

Resumo do capítulo

- Variáveis guardam valores que podem mudar durante a execução.
- Constantes representam valores que devem permanecer fixos.
- `int` é usado para números inteiros.
- `bool` representa os estados `true` e `false`.
- `float` pode representar valores com casas decimais.
- Nomes claros tornam o programa mais fácil de entender e manter.

Próximo capítulo: Entradas e saídas digitais.

← Voltar ao curso