1. Variáveis

Como a memória do Arduino é pequena (2KB na versão UNO), é interessante escolher o melhor tipo de variável conforme o tipo de dado. A relação mostra os tipos mais comuns.

- **Char** ➔ utilizam 1 byte, e são utilizados para armazenar caracteres.
- **Byte** ➔ podem armazenar um número entre 0 e 255.
- **Int** ➔ ocupam 2 bytes (16 bits) e tanto armazenam um número entre -32.768 e 32,767.
- **Long** ➔ ocupam 32 bits (4 bytes), de -2.147.483.648 até 2.147.483.647.
- **Float** ➔ números decimais que ocupam 32 bits (4 bytes).
- **Double** ➔ também armazena números decimais, mas possuem 8-bytes (64 bit).

Mais informações em:
https://www.arduino.cc/en/Reference/VariableDeclaration

A atribuição é simples, indique o tipo da variável seu nome e o conteúdo a ser atribuído, como nos exemplos:

```
int idade = 15;
byte numero = 189;
float nota = 10.5;
char resposta = 's';
```

As atribuições podem ser realizadas mediante operações, por exemplo:

```
int x = 10;
int y = 20;
int resp = x + y;
```

As variáveis do tipo objeto ***String***, servem para armazenar cadeias de texto.

```
String textoum = "Oi, meu texto";
```

Podemos realizar a conversão de char para String.

```
String textodois =  String("a");
```

Conversão de texto fixo para objeto String.

```
String textotres =  String("Esse texto como objeto");
```

Conversão de número para texto.

```
String textoquatro =  String(100);
```

1.1. Visibilidade

As variáveis podem ser visíveis de forma ***GLOBAL*** ou ***LOCAL***.

Exemplo 01 - Global

```
int contador = 10;
// inicialização com o valor 10

void setup()
{
        Serial.begin(9600);
         Serial.println(contador);
        // teremos o valor 10 sendo impresso
        Serial.println("Dados - Setup");

}
void loop()
{
        contador = contador + 1;
        Serial.println(contador);
        // teremos os valores 11, 12, 13, 14, etc
        delay(1000);
}
```

Define-se como variável global quando pode ser acessada em qualquer parte do programa, neste caso pelas funções ***setup()*** e ***loop().***

Exemplo 02 - Local

```
void setup()
{
        int contador = 10;
        Serial.begin(9600);
        Serial.println(contador);
        // teremos o valor 10 sendo impresso
        Serial.println("Dados - Setup");

}
 void loop()
{
        contador = contador + 1;
        Serial.println(contador);
        // teremos agora um erro, a variável contador não existe neste contexto
        delay(1000);
}
```

Corrigindo o código teremos:

```
void setup()
{
   int contador = 10;
        Serial.begin(9600);
        Serial.println(contador);
        // teremos o valor 10 sendo impresso
        Serial.println("Dados - Setup");

}
void loop()
{
        int contador = 0;
        contador = contador + 1;
        Serial.println(contador);
        // teremos os valores 1, 1, 1, 1, 1, etc
        delay(1000);
}
```

ERRO!!! Os valores da função loop(), serão sempre “1”, pois a inicialização ocorre a cada interação da função ***loop()***, então como resolver?

Exemplo 03

```
int contador = 0;

void setup()
{
          int contador = 10;
         Serial.begin(9600);
          Serial.println(contador);
         // teremos o valor 10 sendo impresso
         Serial.println("Dados - Setup");

}

void loop()
{
         contador = contador + 1;
         Serial.println(contador);
         // teremos os valores 1, 2, 3, 4, 5, etc
         delay(1000);
}
```

As variáveis da função loop(), para que ***não*** sejam inicializadas a cada interação devem ser declaradas no início do código.

2. Operadores

2.1. Atribuição

(=)

2.2. Aritméticos

(+) adição
(-) subtração
(*) multiplicação|
(/) divisão
(%) resto da divisão
(++) incremento
(--) decremento

2.3. Relacionais

(>) Maior que
(>=) Maior ou igual que
(<) Menor
(<=) menor ou igaul que
(==) igual a
(!=) diferente

2.4. Lógicos

(&&) and (e)
(||) or (ou)
(!) not (não)

2.5. Lógicos Bit a Bit

(&) and (e)
(|) or (ou)
(^) xor (ou exclusivo)
(~) not (não)
(>>) deslocamento à direita
(<<) deslocamento à esquerda

2.6. Associação de Operadores (forma reduzida)

(x = x + y) = (x += y)
(x = x - y) = (x -= y)
(x = x * y) = (x *= y)
(x = x / y) = (x /= y)
(x = x % y) = (x %= y)
(x = x & y) = (x &= y)
(x = x | y) = (x |= y)
(x = x ^ y) = (x ^= y)
(x = x << y) = (x <<= y)
(x = x >> y) = (x >>= y)