

# Tecnologia da Informática

Licenciatura em  
Engenharia  
Informática  
Universidade de  
Coimbra  
2023/2024

# Conteúdo

- Variáveis: definição e tipos
- Comunicações série
- Operadores aritméticos, relacionais e lógicos

# Introdução às variáveis com o programa *Blink*

# Variáveis

As variáveis são um **conceito poderoso**:

- poupam trabalho
- evitar repetições
- ajudam a GENERALIZAR os algoritmos.

Exemplo: Programa blink da aula anterior:

```
void setup()
{
    pinMode(13, OUTPUT);
}
void loop(){
    digitalWrite(13, HIGH);
    delay(1000); // Wait for 1000 millisecond(s)
    digitalWrite(13, LOW);
    delay(1000); // Wait for 1000 millisecond(s)
}
```

Que sucede se mudarmos o LED do pino 13 para o 12 ?

# Variáveis

Se mudarmos o LED de pino, temos de atualizar o programa em três locais distintos:

```
void setup()
{
    pinMode(12, OUTPUT);
}
void loop(){
    digitalWrite(12, HIGH);
    delay(1000); // Wait for 1000 millisecond(s)
    digitalWrite(12, LOW);
    delay(1000); // Wait for 1000 millisecond(s)
}
```

De prático, isto tem pouco...e se o programa fosse maior ?

# Variáveis

Vamos simplificar:

- **declarar uma variável** para o pino do LED
- **usar a variável** do pino do LED nas instruções

A variável **LED** é usada como argumento nas funções.

Para alterar o pino físico do **LED**, só temos de atualizar a inicialização da variável com o número/valor correto.

Definimos e inicializamos a variável – denominada **LED** que armazena o valor do pino que queremos utilizar (12)

```
int LED=12;
void setup()
{
    pinMode(LED, OUTPUT);
}
void loop(){
    digitalWrite(LED, HIGH);
    delay(1000); // Wait for 1000 millisecond(s)
    digitalWrite(LED, LOW);
    delay(1000); // Wait for 1000 millisecond(s)
}
```

Acabámos de **Generalizar** a solução

# Variáveis

As variáveis são estruturas que guardam valores em memória.

- Criam-se por meio de DECLARAÇÕES.

Exemplo:

```
int x; //declara uma variável com o nome “x” do tipo inteiro (int)
```

- Quando se declara uma variável, o Arduino reserva espaço na memória para ela.
- Cada variável tem um TIPO de dados associado (quando usada em funções, deve ser compatível com a sua SINTAXE).

# Variáveis

Uma variável é uma estrutura que guarda um valor em memória.

- Para associar um valor a uma variável, utilizamos o operador de atribuição. Por exemplo:

```
int a; //Declaração da variável do tipo inteiro com o nome 'a'
void loop(){
    a=0; //definimos o valor zero na variável
    /* Restantes instruções */
}
```

- Podemos também efectuar a atribuição inicial/inicialização aquando da declaração da variável :

```
int a=0; //Declaração e inicialização
/* Restantes instruções */
```



# Variáveis

Ainda podemos melhorar isto...

E que tal guardar o intervalo do *delay()* numa variável ?

```
int LED=12;
int delay_interval=1000; //Declara e inicializa Variável com tempo de espera
void setup()
{
    pinMode(LED, OUTPUT);
}
void loop(){
    digitalWrite(LED, HIGH);
    delay(delay_interval); // Espera pelo tempo definido
    digitalWrite(LED, LOW);
    delay(delay_interval); // Espera pelo tempo definido
}
```

Se quisermos alterar a cadência do piscar,  
basta mudar o valor da variável ***delay\_interval***

# Variáveis

E se quisesse alterar o programa, de modo a que o tempo em que o LED está desligado fosse maior em 200ms que o tempo com o LED ligado ?

# Variáveis

E se quisesse alterar o programa, de modo a que o tempo em que o LED está desligado fosse maior em 200ms que o tempo com o LED ligado ?

```
int LED=12;
int delay_interval=1000; //Declara e inicializa Variável com tempo de espera
void setup()
{
    pinMode(LED, OUTPUT);
}
void loop(){
    digitalWrite(LED, HIGH);
    delay(delay_interval); // Espera pelo tempo definido
    digitalWrite(LED, LOW);
    delay(delay_interval+200); // Espera pelo tempo definido + 200 ms
}
```

# Comunicação Série no ARDUINO

A função *Serial.println()*

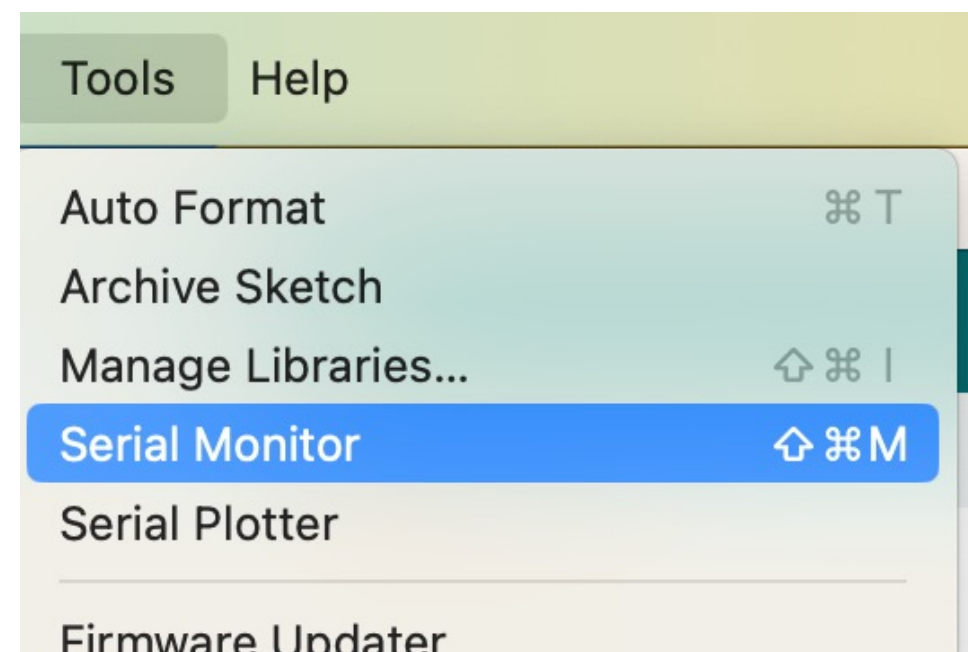
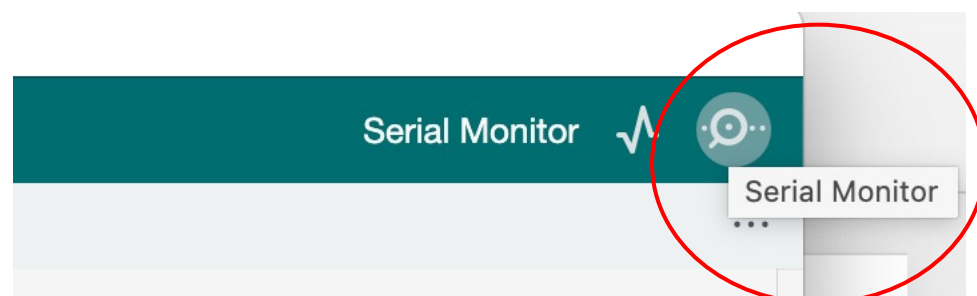
# A função *Serial.println()*

A função *Serial.println()*:

- permite o envio de dados via porta série
- permite acompanhar a execução de um programa (via Serial Monitor)

```
void setup()
{
    Serial.begin(9600); //Define velocidade de transmissão da informação
                        // Número de bits a ser trocado por unidade de tempo
}
void loop(){
    Serial.println("Hello World");
}
```

Ativar o Monitor Série.  
Opção **Tools** -> **Serial Monitor**



# *Serial.println()* e Monitor Série

The screenshot displays the Arduino IDE 2.0.0-rc9 interface. The main editor window shows a sketch named 'sketch\_jul25b.ino' with the following code:

```
1 void setup() {  
2   // put your setup code here, to run once:  
3   Serial.begin(9600);  
4 }  
5  
6 void loop() {  
7   // put your main code here, to run repeatedly:  
8   Serial.println("Hello World!");  
9   delay(1000);  
10 }  
11
```

Below the editor, the 'Serial Monitor' tab is active. It shows the output of the sketch, which is 'Hello World!' repeated four times. The monitor settings are set to 'Both NL & CR' and '9600 baud'. The status bar at the bottom indicates 'Ln 11, Col 1 UTF-8' and 'Arduino Uno on COM13'.

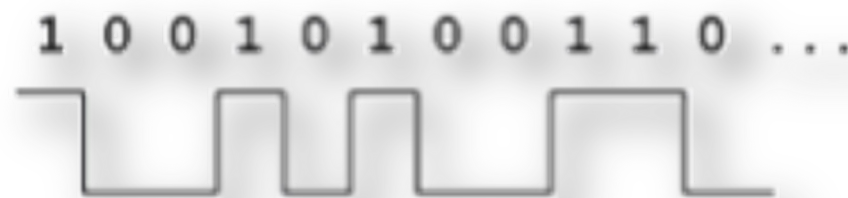
A função **Serial.println()**:

- permite o envio de dados via porta série
- **permite acompanhar a execução de um programa (via Serial Monitor)**

# Comunicação Série

É um método para transferir dados entre dois dispositivos, neste caso entre o PC e o Arduino.

- Dados circulam entre o PC e o Arduino através do cabo USB
- Dados são transmitidos sequencialmente sob a forma de bits (i.e., como zeros '0' e uns '1')





# Comunicação Série



## No Arduino:

- o pino de I/O digital 0 para recepção (RX)
- o pino de I/O digital 1 à transmissão (TX).

Existem LEDs associados a estes pinos :

- RX
- TX

Idealmente, deverá evitar usar estes pinos nos seus circuitos e programas.



# Comunicação Série

Comunicação série ocorre:

- no *upload* de programas (programas enviados do computador para o Arduino)
- nas comunicações entre o programa e o utilizador (via Serial Monitor)

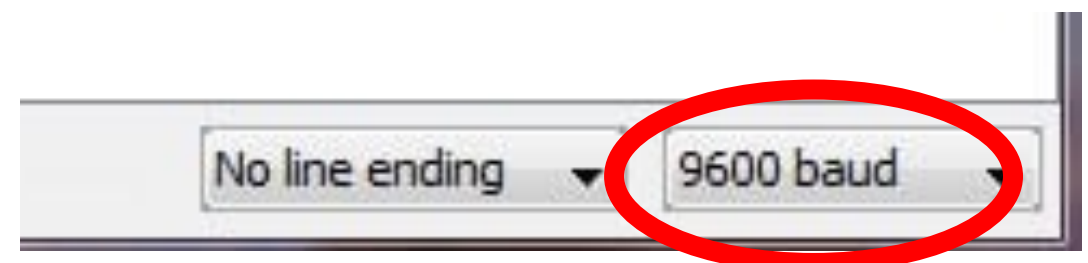
Aspetos importantes na comunicação série:

- No programa deve-se definir a velocidade do número de bits por segundo (bps) através função **Serial.begin()**.  
O Serial Monitor deve ser configurado com a mesma velocidade

Programa no ARDUINO

```
void setup()  
{  
  Serial.begin(9600);  
}
```

Monitor Série



Variáveis

Tipos e Operações

# Variáveis - tipos

//Integer – contém números inteiros (positivos ou negativos)

```
int myNumber = 30000;
```

```
int myAge = -48;
```

// Floats – números com casas decimais

```
float myFloat = 9.54;
```

// Booleans – usados para operações lógicas

// Podem ter valor True (verdadeiro) ou False (falso)

// **(Internamente, true é = 1 e false é = 0)**

```
bool mySwitch = true;
```

```
bool mySwitch2 = false;
```

//Char – caracteres

```
char z = 'd';
```

// Strings

// Uma variável deste tipo contém texto

// Não podem ser aplicadas operações matemáticas

```
String myName = "bananablast9000";
```

```
String mySentence = "o super-herói";
```

# Variáveis - tipos

As variáveis podem ser de vários tipos...

O tipo define a característica dos dados que a variável pode conter.

Deve haver coerência de tipo na manipulação de variáveis. Ou seja, não devemos somar um `float` com um `int` e armazenar numa variável `int` (por causa da redução de precisão).

# Variáveis e operações

As variáveis podem estar envolvidas em operações aritméticas, lógicas, etc...

Exemplo de adição:

```
int a=0;  
a=a+1; // Incrementa um valor à variável 'a'
```

Operações aritméticas e operadores:

+	Adição
-	Subtração
/	Divisão
*	Multiplicação
%	Resto da divisão inteira

# Programa com variáveis

O que faz o seguinte programa ?

```
int a=0;

void setup()
{
  Serial.begin(9600);
}

void loop()
{
  a=a+1;
  Serial.println(a);
}
```

# Programa com variáveis

O que faz o seguinte programa ?

```
int a=0;

void setup()
{
  Serial.begin(9600);
}

void loop()
{
  a=a+1;
  Serial.println(a);
}
```



Serial Monitor

1546  
1547  
1548  
1549  
1550  
1551  
1552  
1553  
1554  
1555  
1556  
----

# Variáveis e operações relacionais

As variáveis podem estar envolvidas em operações aritméticas, lógicas, etc...

## Operadores de comparação (relacionais):

<	Menor que
>	Maior que
<=	Menor ou igual que
>=	Maior ou igual que
==	Igual a
!=	Diferente de...

O resultado destas operações é sempre um **bool/boolean**  
(True ou False)



# Variáveis e operações lógicas

As variáveis podem estar envolvidas em operações aritméticas, lógicas, etc...

**Operadores lógicos** (lembrando a lógica booleana):

Operador	Operação	Descrição
	OR ("OU lógico")	Devolve <i>True</i> <b>SE UM</b> dos elementos for <i>True</i>
&&	AND ("E lógico")	Devolve <i>True</i> <b>SE AMBOS</b> os elementos forem <i>True</i>
!	NOT ("Negação/inversão lógica")	<i>True</i> Passa a <i>False</i> <i>False</i> passa a <i>True</i>

O resultado destas operações é sempre um **bool/boolean**  
(True ou False)

# Programa 2 com variáveis

O que faz o seguinte programa ?

```
int a=0;

void setup()
{
  Serial.begin(9600);
}

void loop()
{
  a=a+1;
  Serial.println(a);
  Serial.println(a>100);
}
```

# Programa 2 com variáveis


O que faz o seguinte programa ?

```
int a=0;

void setup()
{
  Serial.begin(9600);
}

void loop()
{
  a=a+1;
  Serial.println(a);
  Serial.println(a>100);
}
```

1	97
0	0
2	98
0	0
3	99
0	0
4	100
0	0
5	101
0	1
6	102
0	1
7	103
	1
	104
	1
	105
	1
1	
490	
1	
491	
1	
492	
1	
493	
1	
494	
1	
495	

 Serial Monitor

# Programa 3 com variáveis

O que faz o seguinte programa ?

```
int a=0;

void setup()
{
    Serial.begin(9600);
}

void loop()
{
    a=a+1;
    Serial.println(a);
    Serial.println((a>100) && (a%2==0));
}
```

# Programa 3 com variáveis

O que faz o seguinte programa ?

```
int a=0;

void setup()
{
  Serial.begin(9600);
}

void loop()
{
  a=a+1;
  Serial.println(a);
  Serial.println((a>100) && (a%2==0));
}
```



# Exemplos de bool/boolean

O que faz o seguinte programa ?

```
1  bool A, B, C;
2  int a=12,b=12,c=13;
3
4  void setup(){
5      Serial.begin(9600);
6      Serial.println(a);
7
8      A=true;
9      B=false;
10
11     Serial.println(!A);
12
13     C=A && B;
14     Serial.println(C);
15
16     C=A || B;
17     Serial.println(C);
18
19     C=A && !B;
20     Serial.println(C);
21
22     C=(a==b);
23     Serial.println(C);
24
25     C=(a>c);
26     Serial.println(C);
27 }
28
29 void loop(){ }
```

# Exemplos de bool/boolean

O que faz o seguinte programa ?

```
1  bool A, B, C;
2  int a=12,b=12,c=13;
3
4  void setup(){
5      Serial.begin(9600);
6      A=true;
7      B=false;
8
9      Serial.println(!A);
10
11     C=A && B;
12     Serial.println(C);
13
14     C=A || B;
15     Serial.println(C);
16
17     C=A && !B;
18     Serial.println(C);
19
20     C=(a==b);
21     Serial.println(C);
22
23     C=(a>c);
24     Serial.println(C);
25 }
26
27 void loop(){ }
```

Linha 9 !A=!true= !1 → 0

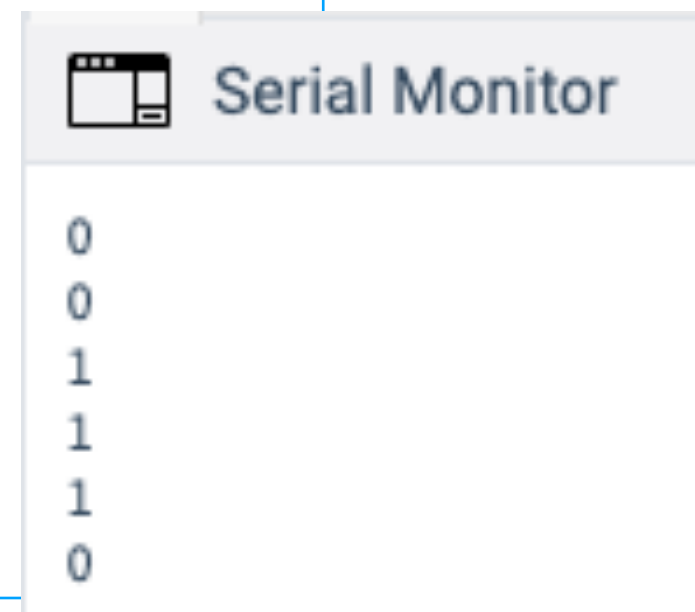
Linha 12 C=A && B → true && false → 0

Linha 15 C=A||B → true || false → 1

Linha 18 C=A && !B → true && !false → 1

Linha 18 C=(a == b) → 12 == 12 → 1

Linha 24 C=(a>c) → 12 > 13 → 0



# Também temos instruções condicionais...

(dependem de uma condição avaliada como *True* ou *False*: um *boolean*)

O que faz o seguinte programa ?

```
int a=0;

void setup()
{
  Serial.begin(9600);
}

void loop()
{
  a=a+1;
  Serial.println(a);
  if ((a>100) && (a%2==0))
    Serial.println("par, maior que 100");
}
```

 Serial Monitor

```
97
98
99
100
101
102
par, maior que 100
103
104
par, maior que 100
105
106
par, maior que 100
107
108
par, maior que 100
109
110
par, maior que 100
111
112
par, maior que 100
113
114
par, maior que 100
115
```

A sintaxe é:

```
if <condição> <instrução ou bloco de instruções>;
[else <instrução ou bloco de instruções>;]
```



# Variáveis e âmbito

As variáveis têm um **âmbito** !

1. **LOCAL**: Se uma variável for criada dentro de um bloco (entre “{” e “}”) ela só existe dentro desse bloco.
2. **GLOBAL**: Uma variável criada no princípio do código (antes do setup()), fora dos blocos de código, visto que o seu âmbito abrange toda a extensão do programa. Pode ser usada em todos os blocos.

**NOTA:** um bloco de instruções está sempre delimitado entre chavetas (‘{’ e ‘}’).

# Exemplo do âmbito das variáveis

O que acontece com o seguinte programa ?

```
int a=0;

void setup()
{
    int local=100;
    Serial.begin(9600);
}

void loop()
{
    a=a+1;
    Serial.println(a);
    Serial.println(local);
}
```

# Exemplo do âmbito das variáveis

O que acontece com o seguinte programa ?

```
int a=0;

void setup()
{
  int local=100;
  Serial.begin(9600);
}

void loop()
{
  a=a+1;
  Serial.println(a);
  Serial.println(local);
}
```

A variável '*local*' é **LOCAL** só existe dentro do bloco do setup().

A variável '*a*' é **GLOBAL**.

Sorry, it seems like your code has some errors.

```
In function 'void loop()':
14:18: error: 'local' was not declared in this scope
exit status 1
```

# Exemplo do âmbito das variáveis

Consegue antever o que irá acontecer com os seguintes programas?

```
int a=0;

void setup() {
  Serial.begin(9600);
}

void loop() {
  a=a+1;
  Serial.println(a);
}
```

```
void setup() {
  Serial.begin(9600);
}

void loop() {
  int a=0;
  a=a+1;
  Serial.println(a);
}
```

# Exemplo do âmbito das variáveis

Consegue antever o que irá acontecer com os seguintes programas?

Serial monitor

```
1  
2  
3  
4  
5  
6
```

```
int a=0;  
  
void setup() {  
    Serial.begin(9600);  
}  
  
void loop() {  
    a=a+1;  
    Serial.println(a);  
}
```

```
void setup() {  
    Serial.begin(9600);  
}  
  
void loop() {  
    int a=0;  
    a=a+1;  
    Serial.println(a);  
}
```

Serial monitor

```
1  
1  
1  
1  
1  
1  
1
```

No segundo caso, a variável '*a*' é **LOCAL** e só existe dentro do bloco do *loop()*. É efémera, na medida em que será destruída e recriada a cada execução do bloco,

# Variáveis e Conversão entre tipos de dados

# Exemplo de conversão entre **int** e **float**

O que faz o seguinte programa ?

```
int a=10,b=5,divide=0,resto,soma,c=10,f=3;
float d=15.33, e;

void setup(){
    Serial.begin(9600);

    a=a+1;
    Serial.println(a);

    divide=a/b;
    Serial.println(divide);

    resto=a%b;
    Serial.println(resto);

    e=d+a;
    Serial.println(e);

    soma=a+d;
    Serial.println(soma);
}

void loop(){ }
```

# Exemplo de conversão entre **int** e **float**

O que faz o seguinte programa ?

```
1  int a=10,b=5,divide=0,resto,soma,c=10,f=3;
2  float d=15.33, e;
3
4  void setup(){
5      Serial.begin(9600);
6
7      a=a+1;
8      Serial.println(a);
9
10     divide=a/b;
11     Serial.println(divide);
12
13     resto=a%b;
14     Serial.println(resto);
15
16     e=d+a;
17     Serial.println(e);
18
19     soma=a+d;
20     Serial.println(soma);
21 }
22
23 void loop(){ }
```

Linha 8 **int**  $a=a+1 \rightarrow 11$

Linha 11 **int**  $divide=a/b = 11/5 \rightarrow 2$

Linha 14 **int**  $resto=a\%b = 11\%5 \rightarrow 1$

Linha 17 **float**  $e=d+a = 15.33+11 \rightarrow 26.33$

Linha 20 **int**  $soma=a+d = 15.33+11 \rightarrow 26$



Serial Monitor

```
11
2
1
26.33
26
```



# Exemplo de conversão entre **int** e **float**

O que faz o seguinte programa ?

```
1 int a=10,b=5,divide=0,resto,soma,c=10,f=3;
2 float d=15.33, e;
3 void setup(){
4     Serial.begin(9600);
5     a=a+1;
6     Serial.println(a);
7     divide=a/b;
8     Serial.println(divide);
9 }
```

Linha 8 **int** divide=a/b = 11/5 → 2

## Porquê é que o resultado foi 2 e não 2.2 ?

Em C. quando os dois operandos numa divisão são inteiros, é efetuada uma divisão inteira. De facto, **a** e **b** são inteiros, razão pela qual o resultado da divisão será truncado (isto é, a parte fracional será descartada).

Em **Python 2** o comportamento do operador “/” era semelhante ao C (para números inteiros positivos), mas em **Python 3** tal já não sucede.

```
Python 2.7.16 (default,
[GCC 4.2.1 Compatible Ap
on darwin
Type "help", "copyright"
[>>> print (11/5)
2
>>> █
```

```
Python 3.7.4 (default,
[Clang 4.0.1 (tags/RELE
Type "help", "copyright"
[>>> print (11/5)
2.2
[>>> _
```

# Exemplo de Strings e **char**

O que faz o seguinte programa ?

```
1 String myName="bananablast9000";
2 String mySentence = " o super-heroi"; //NOTEM AS ASPAS !!!
3 char a='A';                          //NOTEM AS PLICAS !!
4
5 void setup(){
6     Serial.begin(9600);
7     Serial.println(a);
8     Serial.println("O nome:"+myName);
9     Serial.println(myName+mySentence); //Exemplo de concatenação
10 }
11
12
13 void loop(){ }
```

# Exemplo de Strings e **char**

O que faz o seguinte programa ?

```
1 String myName="bananablast9000";
2 String mySentence = " o super-heroi"; //NOTEM AS ASPAS !!!
3 char a='A';                          //NOTEM AS PLICAS !!
4
5 void setup(){
6     Serial.begin(9600);
7     Serial.println(a);
8     Serial.println("O nome:"+myName);
9     Serial.println(myName+mySentence); //Exemplo de concatenação
10 }
11
12
13 void loop(){ }
```



Serial Monitor

```
A
O nome:bananablast9000
bananablast9000 o super-heroi
```

# Exercícios TPC

- Exercício 1.1
- Exercício 1.2

# Para estudar

Existem outros tipos de variáveis, a consultar:

1. [byte](#)
2. [double](#)
3. [long](#)
4. [short](#)
5. [size\\_t](#)
6. [string](#)
7. [unsigned char](#)
8. [unsigned int](#)
9. [unsigned long](#)
10. [word](#)

Para cada tipo, procure também descobrir o espaço (em bits) utilizado em memória.