String

As variáveis também podem ser declaradas como String

- Em C uma String é na realidade um Array do tipo **char**
- **Char** é um tipo **int** no sentido que é um número, mas um número entre 0 e 127 que representa um caractere.
- Isso significa que é possível fazer aritmética em caracteres, de acordo com a tabela ASCII

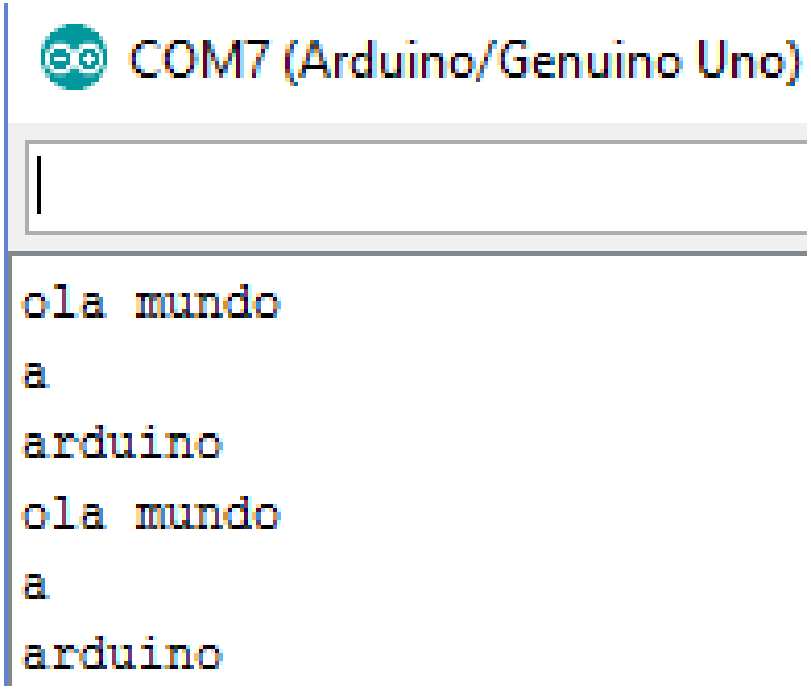
char → 0 a 127

Códigos mais comuns →

Caractere	Código ASCII (em decimal)
a–z	97–122
A–Z	65–90
0–9	48–57
espaço	32

String e Caractere - Regras

- Char – É utilizado entre ‘ ‘
- String – Utilizado entre “ “



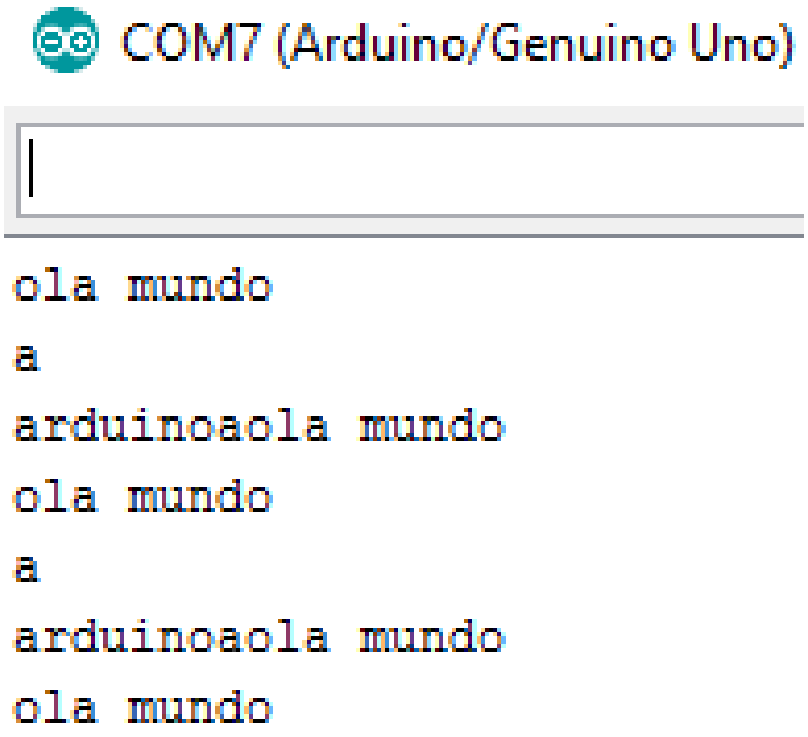
Exemplo 9

Escreva na porta serial uma string “olá mundo”, um caracter ‘a’ e uma cadeia de caractere ‘arduino’ a cada 1 segundo.

```
char message[] = "ola mundo";
char caracter = 'a';
char cadeiadecaracter[7] = {'a','r','d','u','i','n','o'};

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

void loop()
{
  Serial.println(message);
  delay(1000);
  Serial.println(caracter);
  delay(1000);
  Serial.println(cadeiadecaracter);
  delay(1000);
}
```



Exemplo 9 - Solução

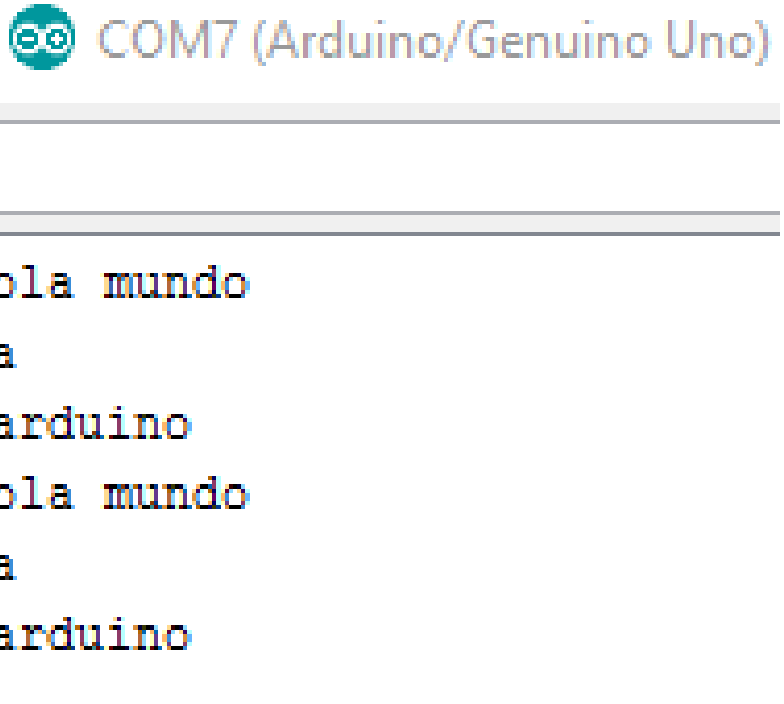
= char caracter = 97;

Tabela ASCII

```
char message[] = "ola mundo";
char caracter = 'a';
char cadeiadecaracter[8] = {'a','r','d','u','i','n','o', '\0'};

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

void loop()
{
  Serial.println(message);
  delay(1000);
  Serial.println(caracter);
  delay(1000);
  Serial.println(cadeiadecaracter);
  delay(1000);
}
```



Exemplo 10

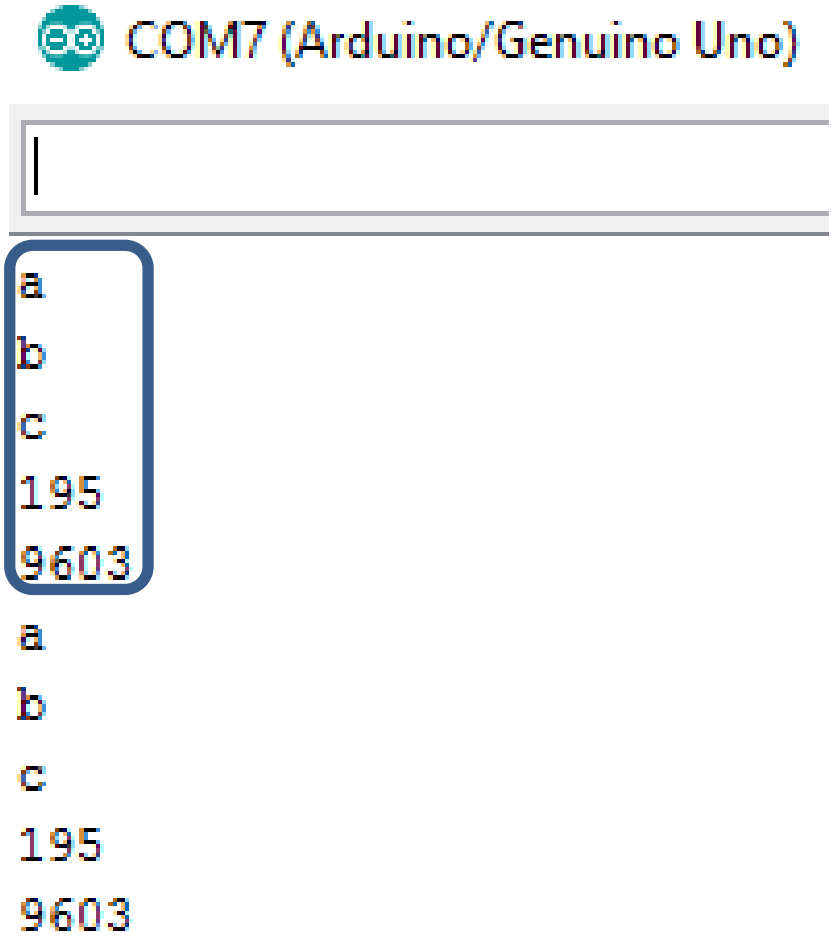
Realize operações matemáticas com caracteres.

```
char caracter = 'a';
char caracter2 = 'b';
char caracter3 = 99;

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

void loop()
{

  Serial.println(caracter);
  delay(1000);
  Serial.println(caracter2);
  delay(1000);
  Serial.println(caracter3);
  delay(1000);
  int c = caracter + caracter2;
  int d = caracter * caracter3;
  Serial.println(c);
  delay(1000);
  Serial.println(d);
  delay(1000);
}
```



Código Morse

Nesta parte, será utilizado o Código Morse como exemplo para construir um programa mais complexos do que foi visto sobre **arrays** e **strings.**

A	.-	N	-.	0	-----
B	-...	O	---	1	.----
C	-.-.	P	.--.	2	..---
D	..	Q	--.-	3	...--
E	.	R	.-.	4	....-
F	..-.	S	...	5	.....
G	-.	T	-	6	-....
H	....	U	..-	7	--...
I	..	V	...-	8	---..
J	.---	W	.--	9	----.
K	-.-	X	-..-		
L	.-..	Y	-.--		
M	--	Z	--..		

Letras em código Morse

Funcionamento do Código Morse

-  - Significa um tempo de espera de X segundos
-  - Significa um tempo de espera de 3X segundos

Exemplo 11

Construa um tradutor de código Morse Complexo. Para isto:
Elabore um sistema através de LED´s que transmita uma mensagem digitada através do controle de tempo, em deixar o LED Acesso e Apagado durante um tempo configurado, de acordo com a regra do código MORSE e que utilize toda a tabela de pontos, traços e números da tabela do código Morse.

Siga estes procedimentos:

 - O Led deve acender e esperar 200 ms e em seguida apagar.

 - O Led deve acender e esperar 600 ms e em seguida apagar.

→ Depois que o Led se apagar, ele deve esperar o tempo de 1 ponto.
→ Esperar um tempo de 3 pontos entre as letras
→ Esperar um tempo de 4 pontos entre as palavras.

Expressar os dados da tabela

- Tabela para LETRAS

```
char* letters[] = {
  ".-", "-...", "-.-.", "-..", ".", // A-I
  "..-.", "--.", "....", "..",
  ".---", "-.-", ".-..", "--", "-.", // J-R
  "---", ".--.", "--.-", ".-.",
  "...", "-", "..-", "...-", ".--", // S-Z
  "-..-", "-.--", "--.."
};
```

- Tabela para Números

```
char* numbers[] = {
   "-----", ".----", "..---", "...--", "....-",
"....." , "-....", "--...", "---..", "----."};
```

Exemplo 11 – PARTE 1 - Programação por Intenção

Elaborar um algoritmo sobre o que você quer fazer ou qual é a sua intenção.

Exemplo de um programa de intensão:

1. Se Houver um caractere para ser lido na estrada USB;
2. Se for uma letra, transmita-o pelo LED usando o array de letras;
3. Se for um número, transmita-o pelo LED usando o array de números;
4. Se for um espaço, espere quatro vezes a duração de um ponto;

Caractere	Código ASCII (em decimal)
a–z	97–122
A–Z	65–90
0–9	48–57
espaço	32

Resolucão

```
void loop()
{
   char ch;
   if (Serial.available() > 0) 1
   {
     ch = Serial.read();
     if (ch >= 'a' && ch <= 'z') 2
     {
        flashSequence(letters[ch - 'a']);
     }
     else if (ch >= 'A' && ch <= 'Z') 2.1
     {
        flashSequence(letters[ch - 'A']);
     }
     else if (ch >= '0' && ch <= '9') 3
     {
        flashSequence(numbers[ch - '0']);
     }
     else if (ch == ' ') 4
     {
      delay(dotDelay * 4); // intervalo de tempo entre palavras
     }
   }
```

97 < Ch < 122
(Letra Minúscula)

Se o que veio da USB for “a” então:

- Transmita: 

Qual é a String com a sequência de pontos e traços que é necessário enviar para a função flashSequence?

O que significa então: `flashSequence(letters[ch - 'a']);`

- É a posição da string no Array letters[]

Ex.: a letrada digitada foi “a”. Assim:
a – a = 97 – 97 = 0

```
char* letters[] = {
  ".-", "-...", "-.-.", "-..", ".", // A-I
```

0 1 2

Irá obter o elemento 0 que neste caso é: .-

Exemplo 11 – PARTE 2 – Escrever a função flashSequence

Por enquanto a função **flashSequence** recebeu uma string contendo uma sequência de pontos e traços. Desta forma ela deverá acende e apagar o LED com os intervalos de tempo necessário.

Vamos escrever esta função da seguindo os seguintes passos:

1. Para cada elemento da string de pontos e traços (tal como )
    - Faça o LED piscar conforme seja um ponto ou um traço;
    - Deve ser lido toda a sequência de string até encontrar o marcador final, \0;
    - É necessário fazer a contagem, que inicia em 0 e é incrementada sempre que cada ponto ou traço for processado.

Resolução: Função flashSequence

```
void flashSequence(char* sequence)
{
   int i = 0;
   while (sequence[i] != '\0')
   {
      flashDotOrDash(sequence[i]);
      i++;
   }
   delay(dotDelay * 3); // intervalo de tempo entre letras
}
```

Será a função que acenderá e pagará o LED.

Aqui manda para a função flashDotOrDash o ponto ou traço que acabou de chegar..

Função flashDotOrDash – Passos para escreve-la

Está função é que realmente executa o trabalho de ligar e desligar o LED.

Tudo que esta função precisa fazer é LIGAR o LED e se:

- O argumento for um ponto, deverá esperar o intervalo de tempo de um ponto
- O argumento for um traço, deverá esperar três vezes o intervalo de tempo de um ponto.

->Em seguida, a função deve apagar o LED.
->Finalmente, ela precisa aguardar o tempo de um ponto para fazer a separação entre os pontos e traços.

Função flashDotOrDash

```
void flashDotOrDash(char dotOrDash)
{
   digitalWrite(ledPin, HIGH);
   if (dotOrDash == '.') // pontoOuTraço
   {
   delay(dotDelay); // espera o tempo de um ponto (dot)
   }
   else // deve ser um traço - (dash)
   {
      delay(dotDelay * 3); // espera o tempo de um traço
   }
   digitalWrite(ledPin, LOW);
   delay(dotDelay); // intervalo de tempo entre os flashes
}
```

Exemplo 11 – Completo – setup

```
// sketch 5-05
int dotDelay = 200;
int ledPin = 13;
char* letters[] = {
   ".-", "-...", "-.-.", "-..", ".", "..-.", "--.", "....", "..", // A-I
   ".---", "-.-", ".-..", "--", "-.", "---", ".--.", "--.-", ".-.", // J-R
   "...", "-", "..-", "...-", ".--", "-..-", "-.--", "--.." // S-Z
};
char* numbers[] = {"-----", ".----", "..---", "...--", "....-",
   ".....", "-....", "--...", "---..", "----."};
void setup()
{
   pinMode(ledPin, OUTPUT);
   Serial.begin(9600);
}
```

Exemplo 11 – Completo - loop

```
void loop()
{

  char ch;
  if (Serial.available() > 0)
  {

    ch = Serial.read();
    if (ch >= 'a' && ch <= 'z')
    {
      flashSequence(letters[ch - 'a']);
    }
    else if (ch >= 'A' && ch <= 'Z')
    {
      flashSequence(letters[ch - 'A']);
    }
    else if (ch >= '0' && ch <= '9')
    {
      flashSequence(numbers[ch - '0']);
    }
    else if (ch == ' ')
    {
     delay(dotDelay * 4); // intervalo de tempo entre palavras
    }
  }
}
```

Exemplo 11 – Completo – função 1

```
void flashSequence(char* sequence)
{
   int i = 0;
   while (sequence[i] != '\0')
   {
      flashDotOrDash(sequence[i]);
      i++;
   }
   delay(dotDelay * 3); // intervalo de tempo entre letras
}
```

Exemplo 11 – Completo – função 2

```
void flashDotOrDash(char dotOrDash)
{
   digitalWrite(ledPin, HIGH);
   if (dotOrDash == '.')
   {
      delay(dotDelay); // espera o tempo de um ponto (dot)
   }
   else // deve ser um traço - (dash)
   {
      delay(dotDelay * 3); // espera o tempo de um traço
   }
   digitalWrite(ledPin, LOW);
   delay(dotDelay); // intervalo de tempo entre os flashes
}
```

Referência

Monk, Simon; Programação com Arduino, começando com sketches. Porto Alegre: Bookman, 2013.