

# C++

## Básico ao Avançado

Tipo assim. . .

Heitor Rodrigues Savegnago

UFABC Rocket Design

2017.3

**1** Tipos de dados

**2** Variáveis

**3** Exibindo

**4** Hora de brincar

# Tipos

- Todo programa precisa armazenar dados, até o código mínimo utiliza uma
- quantidade de memória.
- Podem ser armazenados dados numéricos, de texto, estados lógicos.
- Os tipos aqui apresentados são denominados *tipos primitivos*.

# Tipos

- Estados lógicos são armazenados nas memórias tipo `bool`
- Caracteres de texto são armazenados nas memórias tipo `char`
- Valores numéricos apresentam dois tipos de armazenamento:
  - Números inteiros nos tipo `int`
  - Números flutuantes nos tipo `float` e `double`

Há ainda um tipo especial, que não armazena dados, o tipo `void`, seu será explicado daqui algumas aulas.

# Modificadores

- Os modificadores de faixa são palavras-chave que alteram os valores registráveis por um tipo.
- Os modificadores `signed` e `unsigned` alteram a assinatura da faixa.
- Os modificadores `short` e `long` alteram o comprimento da faixa.

# Modificadores

**Tabela:** Relação de faixa e tamanhos de memória para tipos primitivos com modificadores de faixa

código	tamanho (B)	valor mínimo	valor máximo
<code>bool</code>	1	0	1
<code>signed char</code>	1	-127	126
<code>char</code>	1	-127	126
<code>unsigned char</code>	1	0	255
<code>signed short int</code>	2	-32768	32767
<code>short int</code>	2	-32768	32767
<code>unsigned short int</code>	2	0	64535
<code>signed int</code>	4	-2147483648	2147483647
<code>int</code>	4	-2147483648	2147483647
<code>unsigned int</code>	4	0	4294967295
<code>signed long int</code>	8	-9223372036854775808	9223372036854775807
<code>long int</code>	8	-9223372036854775808	9223372036854775807
<code>unsigned long int</code>	8	0	18446744073709551616
<code>float</code>	4	$1.2 \cdot 10^{-38}$	$3.4 \cdot 10^{+38}$
<code>double</code>	8	$1.73 \cdot 10^{-308}$	$1.7 \cdot 10^{+308}$
<code>long double</code>	16	$3.4 \cdot 10^{-4932}$	$3.4 \cdot 10^{+4932}$

# Variáveis

De nada adianta *existir* um tipo de armazenamento de dados se não soubermos usá-lo.

Uma variável é declarada colocando a lista de modificadores, o tipo e o nome da variável.

```
<modificadores> <tipo> <nome>;
```

Quando uma variável é declarada, ela pode vir com lixo de memória, para evitar isso, declaramos a variável com um valor de inicialização, seguindo a sintaxe:

```
<modificadores> <tipo> <nome>(<valor>;
```

# Variáveis

```
1 bool falso(0);           //Com número
2 bool verdadeiro(true);   //Com palavra-chave
3 char igual(0x3D);        //Sinal de igual ASCII
4 char letraA('A');        //Aspas simples
5
6 int contador(1), acumulador(0); //Várias variáveis do mesmo
   tipo
7 unsigned int positivo(523);    //Inteiro sem sinal
8 short doisBytes(93);          //Modificador de
   comprimento
9 long grande(32416189349L);     //Número grande
10 double cargaFundamental(1.6e-19); //Notação científica
11 float pi(3.14159265358979323846264338327950288419f); //
   Flutuante preciso
```

Note os detalhes!



# *printf*

- O `printf` é uma das opções para exibição na tela.
- Pular linha? Caracter especial!  
<[en.cppreference.com/w/cpp/language/escape](http://en.cppreference.com/w/cpp/language/escape)>
- Exibir variáveis? Sequência especial!  
<[en.cppreference.com/w/cpp/io/c/fprintf](http://en.cppreference.com/w/cpp/io/c/fprintf)>

```
printf(<string>, <...>);
```

# printf

O código:

```
printf("cinco + sete = %i\n", 12);  
printf("%i + %i = %s\n", 5, 7, "doze");  
float pi(3.1415);  
printf("Pi vale %f\n", pi);
```

Gera a saída:

```
cinco + sete = 12  
5 + 7 = doze  
Pi vale 3.141500
```

# Vamos testar!