

# Algoritmos e Programação

---

**Prof. Dr. Josenalde Barbosa de Oliveira**

josenalde@eaj.ufrn.br

Aulas: 2T345, 4T345 – 6 CRDS – 90h

Atendimento presencial: 3 e 4: e-mail, discord

# Segundo dados de março 2018\*

---

<https://www.impacta.com.br/blog/2019/03/25/conheca-as-7-linguagens-de-programacao-mais-requisitadas-no-mercado/>

## Por que aprender uma linguagem de programação?

De acordo com uma pesquisa realizada pelo [Escritório de Estatísticas de Trabalho dos EUA](#), os empregos na área de desenvolvimento de software tendem a crescer aproximadamente 8% nos próximos 7 anos e conseguir um bom emprego no ramo é o objetivo de grande parte dos que buscam aprender uma linguagem de programação.

Além disso, a área conta com ótimos salários e benefícios que diversas áreas não têm — como [flexibilidade de horário e local](#), por exemplo. Tudo isso tem tornado a área bastante atrativa, mas o mercado ainda conta com diversas vagas para as principais plataformas e linguagens.

<https://www.impacta.com.br/blog/2019/03/25/conheca-as-7-linguagens-de-programacao-mais-requisitadas-no-mercado/>

1. Javascript
2. Java
3. Python
4. C#
5. PHP
6. C++
7. C
8. Typescript
9. Ruby
10. Swift

Já esta outra lista destaca linguagens prioritariamente para desenvolvimento web e mobile:

Java (web), Python (web), Ruby (web), PHP (web).

Javascript (web,mobile), C# (desktop,mobile), C (hardware)

\*<https://www.devmedia.com.br/top-10-linguagens-de-programacao-mais-usadas-no-mercado/39635>

# Porém surge nova ênfase no C++

---

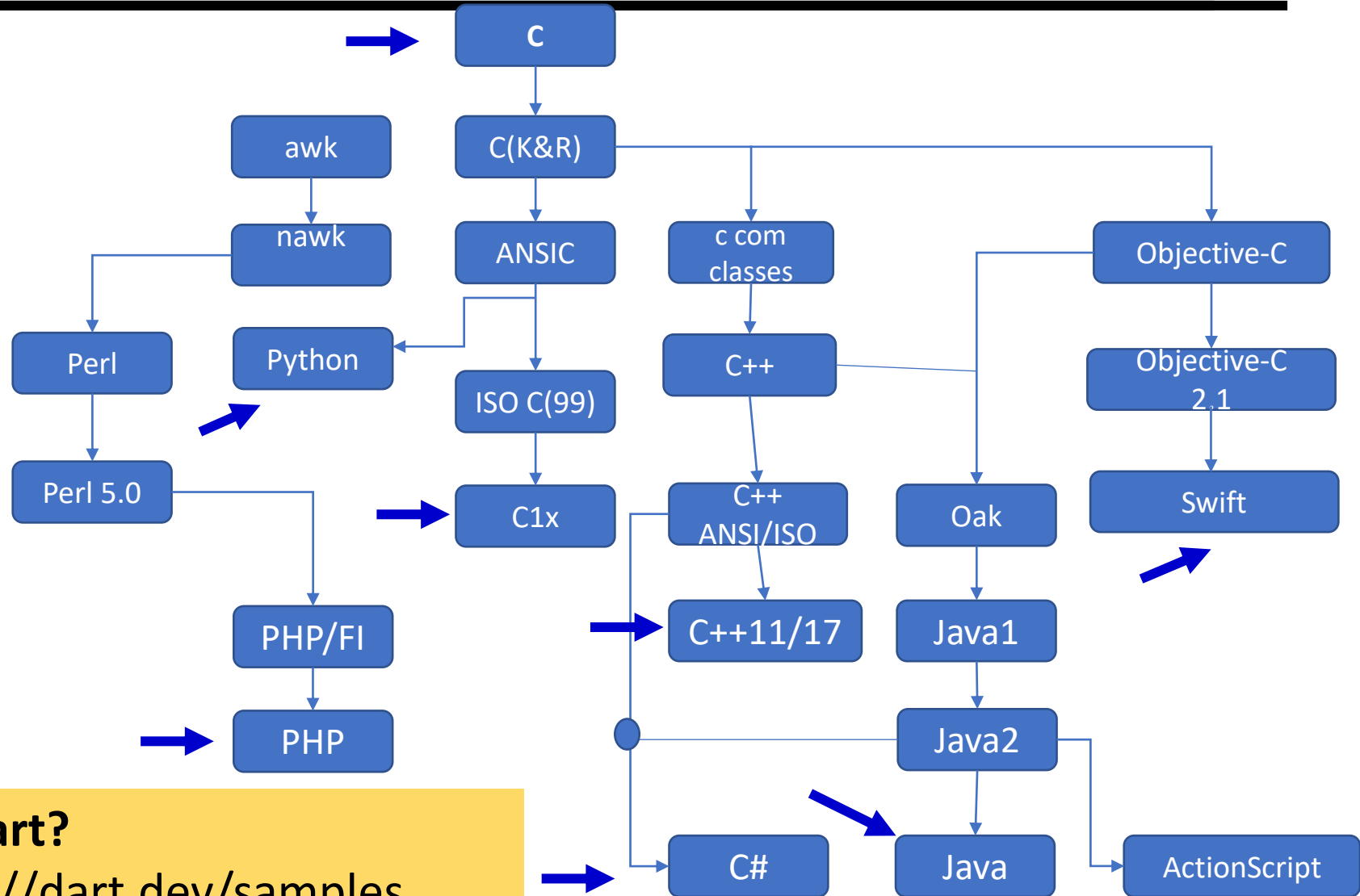
## Bem-vindo C++ de volta para o modernoC++

10/01/2020 • 17 minutos para ler •   

Desde sua criação, C++ o tornou-se uma das linguagens de programação mais amplamente usadas do mundo. Os programas bem escritos que a utilizam são rápidos e eficientes. A linguagem é mais flexível do que outras linguagens: ela pode funcionar nos níveis mais altos de abstração e no nível do silício. C++ fornece bibliotecas padrão altamente otimizadas. Ele permite o acesso a recursos de hardware de baixo nível, para maximizar a velocidade e minimizar os requisitos de memória. Usando C++, você pode criar uma ampla variedade de aplicativos. Jogos, drivers de dispositivo e software científico de alto desempenho. Programas inseridos. Aplicativos cliente do Windows. Até mesmo bibliotecas e compiladores para outras linguagens de programação C++ são escritos.

<https://docs.microsoft.com/pt-br/cpp/cpp/welcome-back-to-cpp-modern-cpp?view=vs-2019>

# C é referência para várias outras



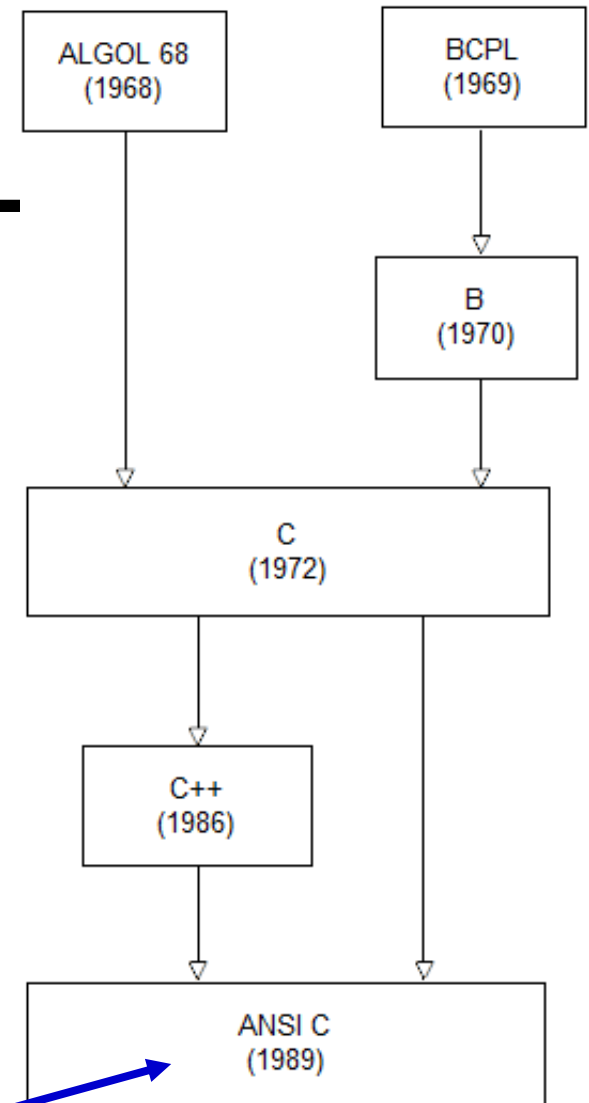
# E a Dart?

<https://dart.dev/samples>  
<https://flutter.dev/>

# Evolução C

---

- ➡ Desenvolvida em 1972, nos Laboratório Bell: Dennis Ritchie e Ken Thompson;
- ➡ -Utilizada para programação de qualquer tipo de sistema (sistemas operacionais, planilhas eletrônicas, processadores de texto, gerenciadores de bancos de dados, processadores gráficos, sistemas de transmissão de dados e telefonia, aparelhos de medicina, engenharia, física etc.)
- ➡ Exemplo: UNIX
- ➡ Poderosa, portátil, flexível, padronizada, execução rápida



**Revisão do ANSI C: C99, C11 e atualmente C18**

# Evolução C++ (1983...)

- Compatível com o C-style, porém agrega recursos mais otimizados, robustos
- Incorpora conceitos de Orientação a Objetos (OO)
- ISO/IEC 14882:1998 (rev 2003, 2014: **c17**)

## Junior Software Engineer

Availability Location

**Coimbra (PT), Lisboa (PT), Porto (PT), Tomar (PT), Vila Real (PT), Viseu (PT)**

Job Code

**354**

✓ **Apply Now**

Working at Critical Software is more than just a job: here, you can really make a difference.

We're on a mission to make the world a better and safer place, just like real superheroes. We build rock-solid software for leading industries' most critical applications, so every line of code we write and every idea we think of can affect the lives of millions.

Now, we are looking for exceptional **Junior Software Engineers** to embrace this mission.

Are you finishing your academic degree? Or do you have less than two years' worth of professional experience in software development? Then it's time to join our team of superheroes and start your professional career at Critical Software!

### KEY RESPONSIBILITIES

- To become part of a software development team in a real-life challenging project
- To participate in all project development cycle phases, with a primary focus on the implementation phase
- To participate in training sessions in order to build strong technical skills

### WHAT WE ARE LOOKING FOR:

- Recent graduates in computer engineering, electrical and computer engineering, telecommunications engineering, biomedical engineering, aerospace, mathematics, physics and other related backgrounds, or software engineers with up to two years' worth of professional experience
- Knowledge in at least one of the following: C, C++, .Net, Java, Javascript, Databases
- Good at solving complex problems

# Ativa no mercado...

---

## ESSENTIAL QUALIFICATIONS, SKILLS AND EXPERIENCE

## Embedded Software Engineer

- ▶ Academic background in Engineering or similar (Software, Electronics, Mechanical, Physics, Mathematics)
- ▶ At least 2 years of experience in software development
- ▶ Knowledge of Ada, C or C++
- ▶ Knowledge of system development using model-based development techniques and tools (for example, SCADE and MATLAB)
- ▶ Full-lifecycle software development experience, from initial requirements elicitation to design, coding, testing, documentation, implementation, integration and training
- ▶ Passionate and ambitious who loves to work as part of a team
- ▶ Good oral and written communication skills in English
- ▶ Availability to travel, mainly within Europe (United Kingdom, Germany, France).



# Ativa no mercado...

---

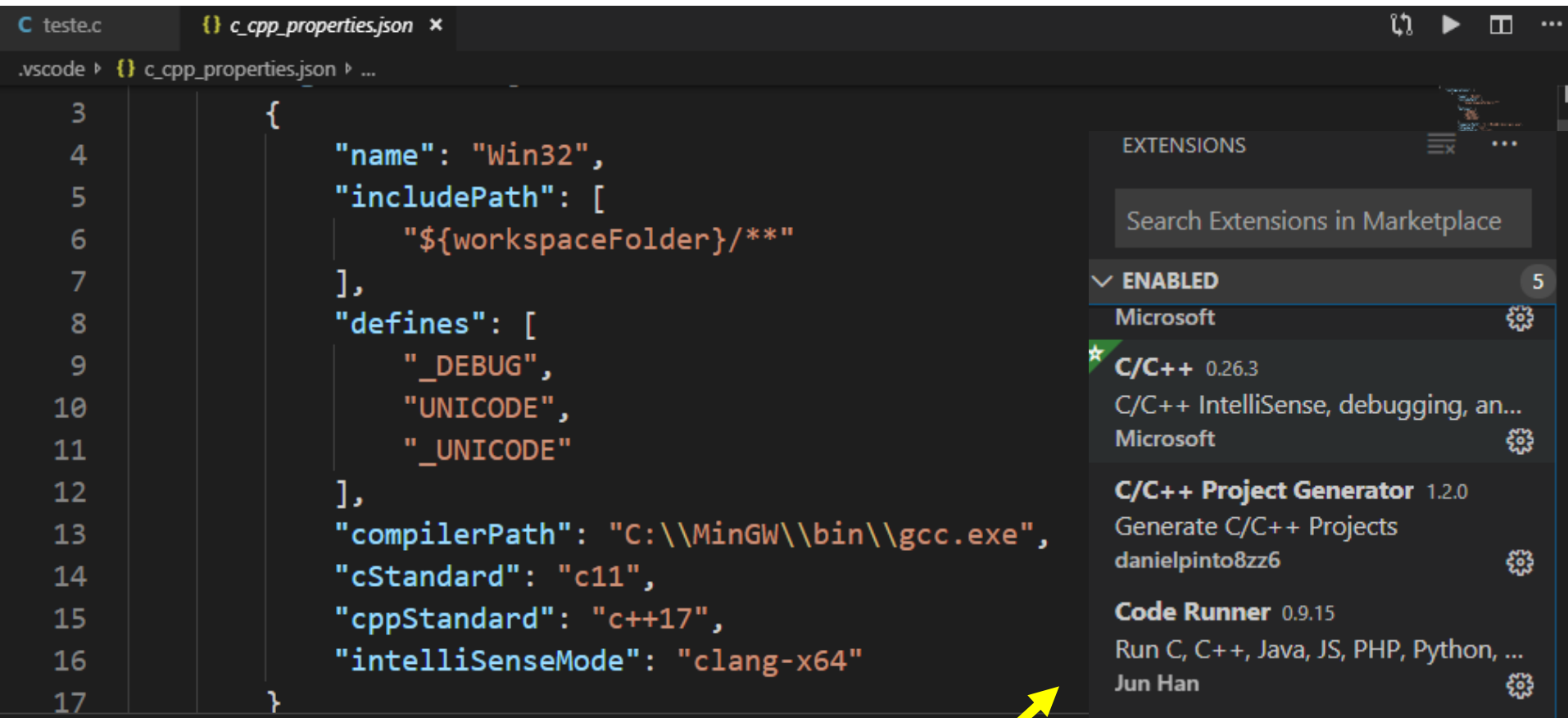
```
main.cpp ✕  
1 #include "def_principais.h"  
2 #define LED PB0  
3  
4 int main(int argc, char *argv[]) {  
5     DDRB = 0xff;  
6  
7     while(1){  
8         set_bit(PORTB, LED);  
9         _delay_ms(1000);  
10        clr_bit(PORTB, LED);  
11        _delay_ms(1000);  
12    }  
13  
14    return 0;  
15 }  
16
```

**Sistemas Embarcados**  
ARM, Arduino, ESP, DSP...





# No VS Code (C/C++ Edit Configurations)



The image shows the Visual Studio Code interface. The main editor displays the `c_cpp_properties.json` file with the following configuration:

```
3 {
4     "name": "Win32",
5     "includePath": [
6         "${workspaceFolder}/**"
7     ],
8     "defines": [
9         "_DEBUG",
10        "UNICODE",
11        "_UNICODE"
12    ],
13    "compilerPath": "C:\\\\MinGW\\\\bin\\\\gcc.exe",
14    "cStandard": "c11",
15    "cppStandard": "c++17",
16    "intelliSenseMode": "clang-x64"
17 }
```

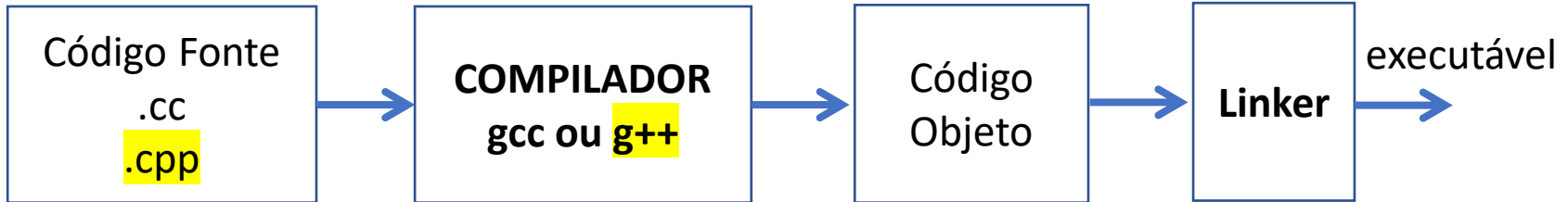
On the right side, the **EXTENSIONS** view is open, showing a list of installed extensions. A yellow arrow points from the word **EXTENSÕES** at the bottom to the **C/C++** extension entry in the list.

EXTENSIONS	
Search Extensions in Marketplace	
▼ <b>ENABLED</b>	5
Microsoft	⚙️
★ <b>C/C++</b> 0.26.3	
C/C++ IntelliSense, debugging, an...	
Microsoft	⚙️
<b>C/C++ Project Generator</b> 1.2.0	
Generate C/C++ Projects	
danielpinto8zz6	⚙️
<b>Code Runner</b> 0.9.15	
Run C, C++, Java, JS, PHP, Python, ...	
Jun Han	⚙️

EXTENSÕES

# Compilação

---



O compilador aciona o linkeditor (ligador) que agrega as funções em linguagem de máquina que foram utilizadas no programa e estão em arquivos de bibliotecas. (libraries (.lib), header files - .h). Ao final deste processo temos um arquivo EXECUTÁVEL (.exe, por exemplo) [Linux: com permissão x]

Existem vários ambientes de desenvolvimento e compiladores gratuitos disponíveis, como o CodeBlocks (gcc,g++). Também podem ser usados o Microsoft Visual Studio Code, Eclipse, ambientes online (repl.it etc).

Compilar: `g++ <fonte.c> -o [nome_exec]`

Executar terminal: `./<nome_exec>`

# Estrutura básica em C++

---

- Um programa em C++ consiste em uma ou várias funções, cuja forma geral é:

```
tipoRetornoFunc nomeFunc(listaParametros)
{
    declaração de variáveis;           /* nenhuma outra função pode ter o
    instrução_2;                        nome main, pois é palavra
    .....                             reservada do C++ */
    instrução_n;
    [return variável ou valor];
}
```

```
int main() {
    return 0; // ou exit(0)
}
```

Portanto, todo projeto C++ deve ter uma função **main**. Ela marca o ponto de partida do programa, que termina quando for encerrada a execução desta função. Se um programa for constituído de uma única função, esta será **main()**. **No caso do retorno é um inteiro (int), que também pode ser usado na função exit(). O número 0 indica ao SO que houve SUCESSO na execução. É OPCIONAL**

# Arquivos de cabeçalho, espaço de nomes...

inclui definições e implementações de funções e constantes que são adicionadas ao código fonte. São incluídos com o símbolo # e palavra-chave include, e o compilador interpreta como diretiva de pré-compilação, ou seja, antes de iniciar a compilação em si, processa esta diretiva **#include**, adicionando o conteúdo dos respectivos arquivos de cabeçalho (em C, .h)

```
#include <stdio.h>
#include <stdlib.h>
#include <iostream>
using namespace std;
int main(){
    printf("ola mundo");
    setlocale("LC_ALL", Portuguese);
    printf("olá mundo");
    cout >> "olá mundo c++";
}
```



Preparando o ambiente....

# Arquivos de cabeçalho, espaço de nomes...

---

NAMESPACE: agrupa bibliotecas de natureza similares Em C++ TODAS as bibliotecas de uso geral estão no namespace STANDARD (padrão), abreviado **std**.

Pode ser entendido como um **diretório** que concentra itens similares

Exemplos básicos:

```
using namespace std;
#include <iostream>
void main() {
    int x;
    cin >> x;
    cout << x;
}
```

```
using std::cin;
using std::cout;
#include <iostream>
//OU...
std::cin >> x;
```

# Caracteres especiais para formatação...

Códigos especiais	Significado
\n	Nova linha
\t	Tabulação
\b	Retrocesso (impressora)
\f	Salto de página de formulário
\a	Beep – toque do alto falante
\r	CR – retorno do cursor
\\	\ - barra invertida
\0	Zero
\'	Aspas simples (apóstrofo)
\"	Aspas dupla
\xdd	Representação hexadecimal
\odd	Representação octal

Tipo	Bits	Bytes	Escala
char	8	1	-128 a 127
int	32	4	-2.147.483.648 a 2.147.483.647
short	16	2	-32.765 a 32.767
long	32	4	-2.147.483.648 a 2.147.483.647
unsigned char	8	1	0 a 255
unsigned	32	4	0 a 4.294.967.295 (int)
unsigned long	32	4	0 a 4.294.967.295
unsigned short	16	2	0 a 65.535
float	32	4	$3,4 \times 10^{-38}$ a $3,4 \times 10^{38}$
double	64	8	$1,7 \times 10^{-308}$ a $1,7 \times 10^{308}$
long double	*	10	$3,4 \times 10^{-4932}$ a $3,4 \times 10^{4932}$
void	0	0	Nenhum valor

O tipo **int** tem o tamanho da palavra da máquina!

A depender da máquina **long double** ocupa 80, 96 ou 128 bits!

O tipo **char** pode ser usado para números inteiros na faixa -128 a 127 ou o **unsigned char** para números inteiros na faixa 0-255, ou seja, naturais nesta faixa limitada.

# No caso de números FLOAT ou DOUBLE

---

```
using namespace std;
```

```
cout.precision(2);
```

```
cout << std::fixed;
```

```
float x = 10.345;
```

```
cout << x;
```

//o fixed indica que a precisão é na parte fracionária. Sem o fixed, assume o número total de dígitos, parte inteira + parte fracionária