

# Mutirão C

Introdução C

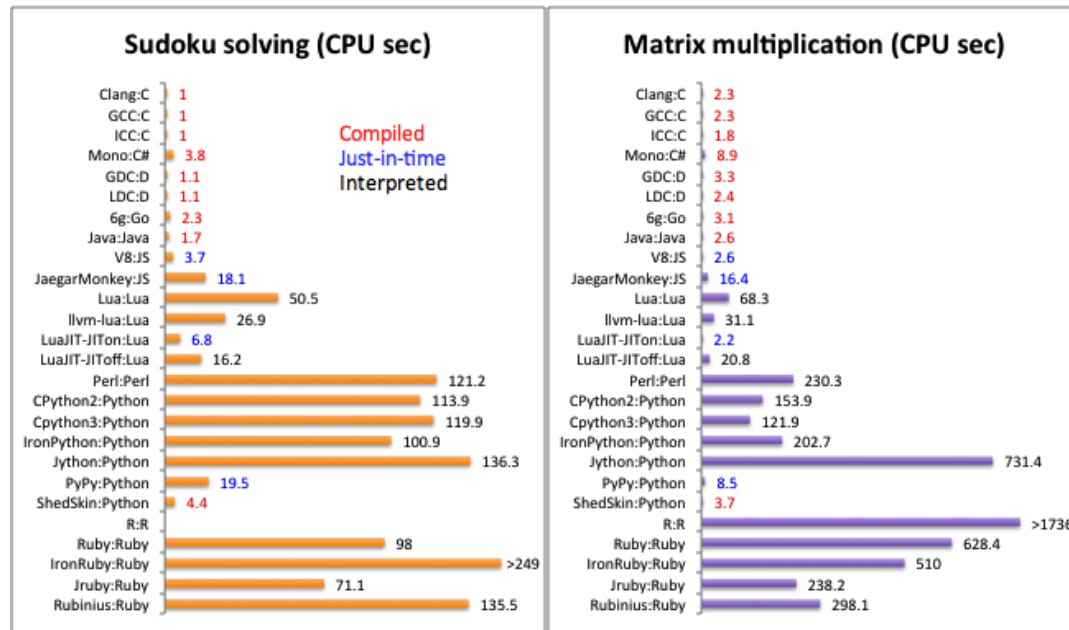
Aula 1

9-8-2017

# Por que C ?

- Possui ótima performance
- Portátil
- É bastante utilizado pela comunidade
- Linguagem de "baixo nível"
- Boas ferramentas gratuitas

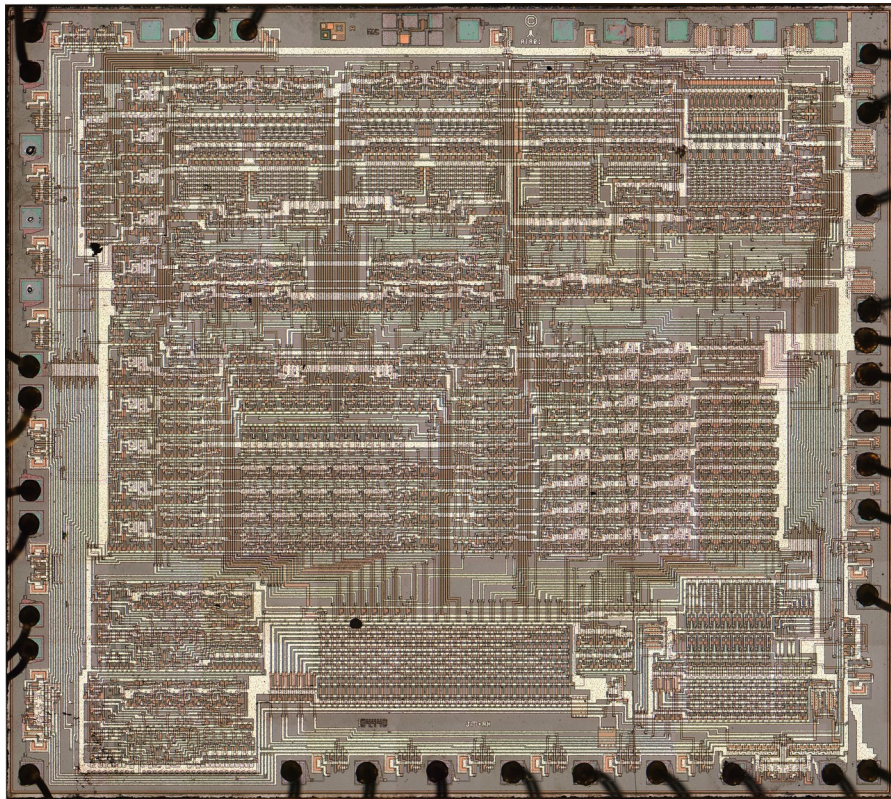
# Possui ótima performance



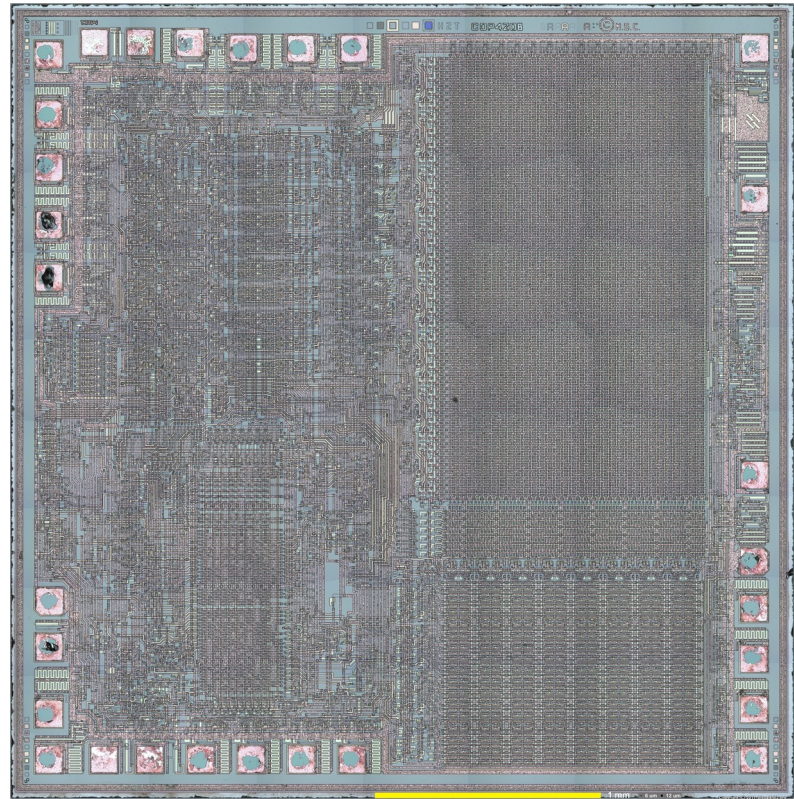
[ref] <https://attractivechaos.github.io/plb/>



# Portável



Atari e 8080



Portável









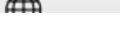
Linux™



FreeBSD®























# É bastante utilizado (2017-a)

Language Rank	Types	Jobs Ranking
1. C		100.0
C is used to write software where speed and flexibility is important, such as in embedded systems or high-performance computing.		
2. Java		98.4
3. Python		96.9
A scripting language that is often used by software developers to add programmability to their applications, such as engineering-analysis tools or animation software.		
4. C++		93.0
5. JavaScript		88.8
6. C#		86.4
7. PHP		81.8
8. Ruby		79.5
9. HTML		78.9

[ref] <http://spectrum.ieee.org/static/interactive-the-top-programming-languages-2016>

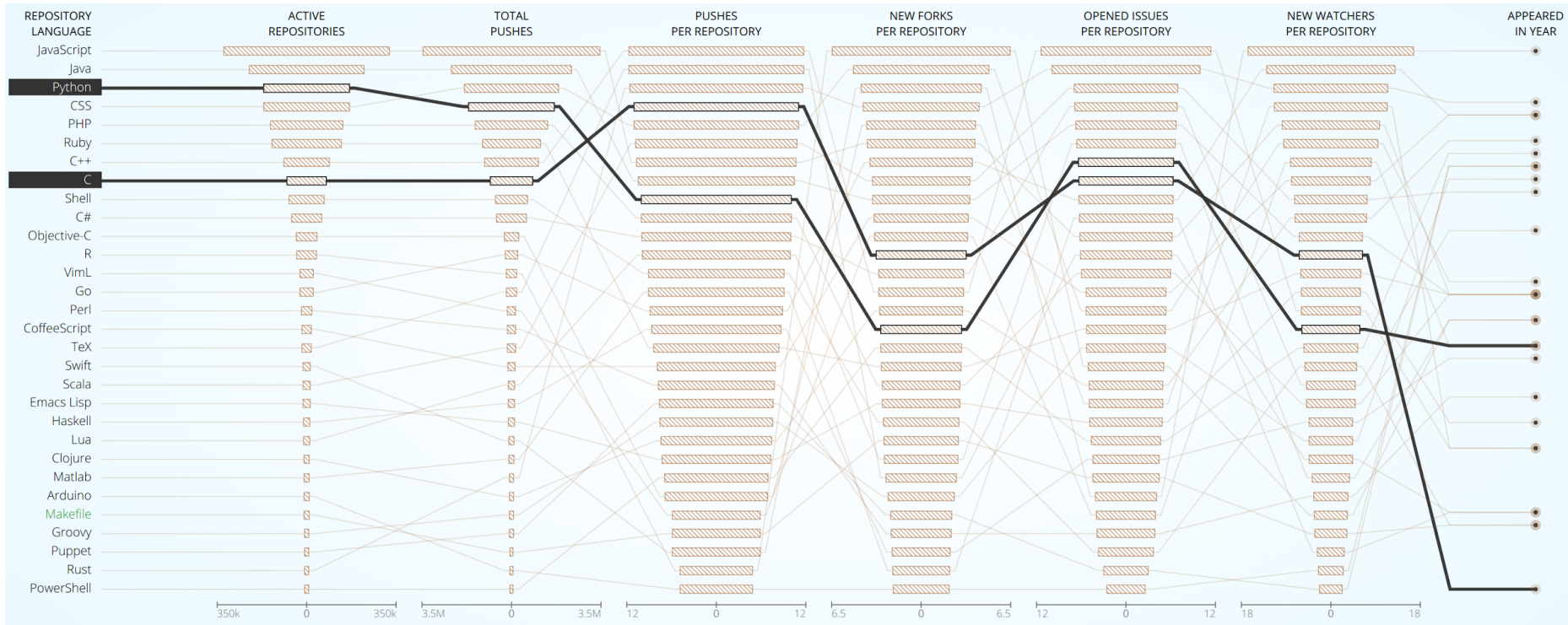
# Ops, as coisas mudam muito rápido (2017-b)

Language Rank	Types	Spectrum Ranking
1. Python	 	100.0
2. C	  	100.0
3. Java	  	99.4
4. C++	  	96.9
5. C#	  	88.6
6. R		88.1
7. JavaScript	 	85.3
8. PHP		81.1
9. Go	 	75.7

[ref] <http://spectrum.ieee.org/static/interactive-the-top-programming-languages-2016>



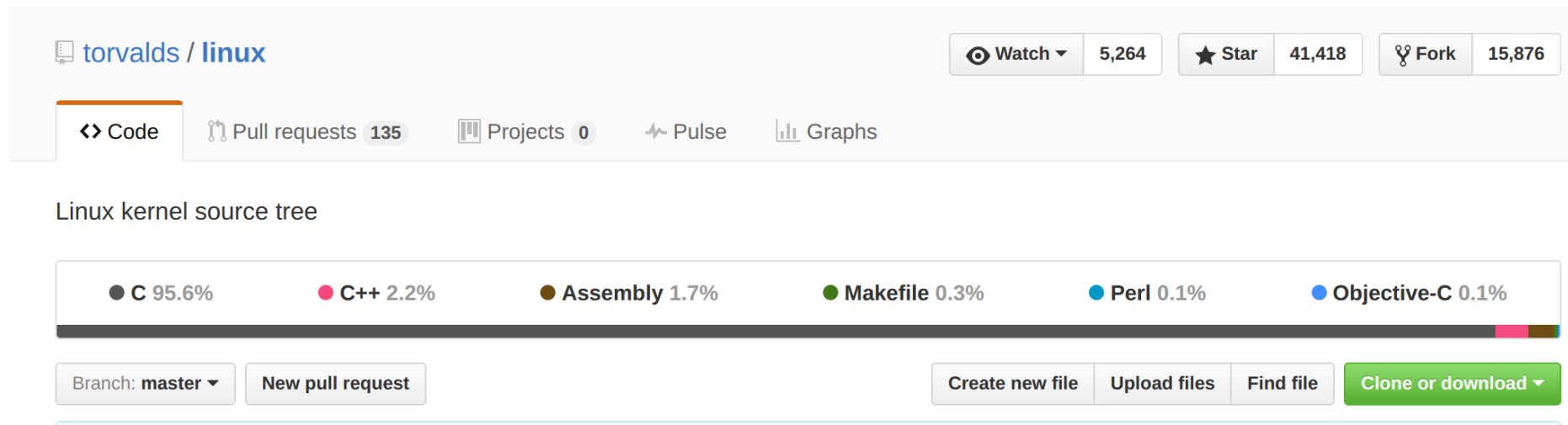
# Github info



[ref] <http://github.info/>

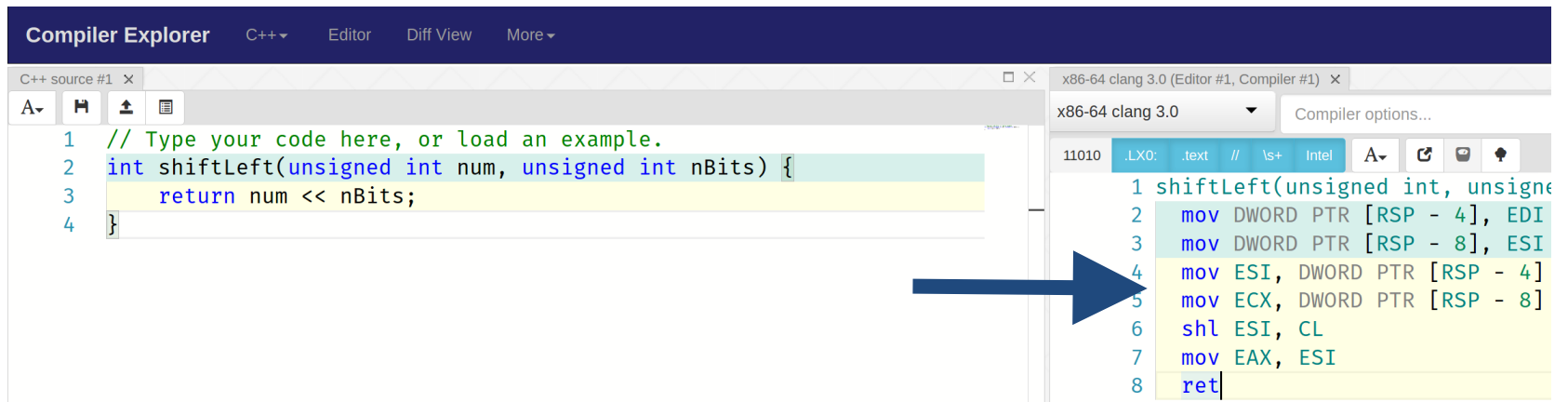


# Linux Kernel



[ref] <https://github.com/torvalds/linux>

# Baixo nível



The image shows a screenshot of the Compiler Explorer web interface. The left pane displays C++ source code for a function named `shiftLeft`. The right pane shows the corresponding assembly code generated by x86-64 clang 3.0. A blue arrow points from the C++ code to the assembly code.

```
1 // Type your code here, or load an example.
2 int shiftLeft(unsigned int num, unsigned int nBits) {
3     return num << nBits;
4 }
```

Assembly code (x86-64 clang 3.0):

```
11010 .LX0: .text // \s+ Intel
1 shiftLeft(unsigned int, unsigne
2     mov DWORD PTR [RSP - 4], EDI
3     mov DWORD PTR [RSP - 8], ESI
4     mov ESI, DWORD PTR [RSP - 4]
5     mov ECX, DWORD PTR [RSP - 8]
6     shl ESI, CL
7     mov EAX, ESI
8     ret
```

# Programação Aulas

Aula 1 - Introdução C

Aula 2 - Funções, vetores, matrizes e strings

Aula 3 - Ponteiros

Aula 4 – Exercícios DP

Aula 5 – Exercícios CE

Aula 6 – Exercícios HS

Aula 7 - Diagnóstico

**Mãos na massa !**

# Infraestrutura

- Linux ou/
  - VmWare / VirtualBox com Linux
- gcc (`sudo apt-get install build-essential`)
- github
  - ( <https://github.com/Insper/mutiraoC> )



# Como um código em C parece ?

```
#include <stdio.h>

void main(){

    printf("Hello !! \n");

}
```

mutiraoC/Aula1/hello.c

- ← Bibliotecas a serem utilizadas
- ← Função principal
- ← Imprime na tela um texto.

# GCC

```
$ gcc hello.c -o hello
```

↑            ↑            ↑            ↑  
compilador   código fonte   output   executável

1. *# Gera o Assembler*

2. **gcc** hello.c -o hello

3.

4. *# Executa o programa*

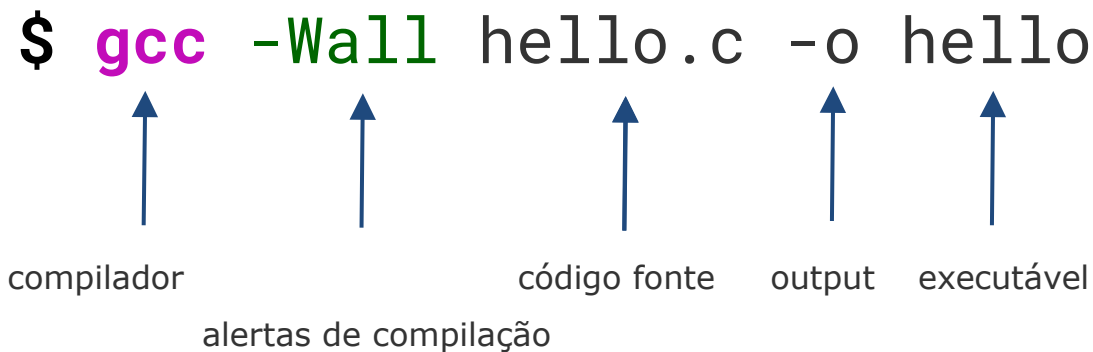
5. ./hello

→ hello

→ Hello!!

# GCC alertas de compilação

\$ gcc -Wall hello.c -o hello



The diagram illustrates the components of the GCC command line. Blue arrows point from the following labels to the corresponding tokens in the command: 'compilador' points to 'gcc', 'alertas de compilação' points to '-Wall', 'código fonte' points to 'hello.c', 'output' points to '-o', and 'executável' points to 'hello'.

compilador      alertas de compilação      código fonte      output      executável

```
hello.c:3:6: warning: return type of 'main' is not  
'int' [-Wmain]  
void main({  
  ^
```

# GCC alertas de compilação

```
hello.c:3:6: warning: return type of 'main' is not  
'int' [-Wmain]  
void main(){  
    ^
```

Por que desse erro ?

# Syntax



# Syntax $\sim$ Java

- Aritimética :
  - \* +, -, \*, /, %
  - \* ++, --, \*=, ...
- Relacional: <, >, <=, >=, ==, !=
- Lógico: &&, ||, !, ? :
- Bit: &, |, ^, ~, <<, >>
- if( ){ } else { }
- while( ){ }
- do { } while( )
- for(i=0; i<100; i++){ }
- switch( ) { case 0: ... }
- break, continue, return

# Exercício 1

(8 min)

O código **mdc.c** deveria calcular o maior divisor comum (MDC) entre dois números, porém o mesmo apresenta problemas.  
Corrija o código e valide o programa.

# Tipos

# Tipos inteiros

Type	Storage size	Value range
char	1 byte	-128 to 127 or 0 to 255
unsigned char	1 byte	0 to 255
signed char	1 byte	-128 to 127
int	2 or 4 bytes	-32,768 to 32,767 or -2,147,483,648 to 2,147,483,647
unsigned int	2 or 4 bytes	0 to 65,535 or 0 to 4,294,967,295
short	2 bytes	-32,768 to 32,767
unsigned short	2 bytes	0 to 65,535
long	4 bytes	-2,147,483,648 to 2,147,483,647
unsigned long	4 bytes	0 to 4,294,967,295

# Tipos inteiros - **Varia**m !!!

PIC 18F microcontrolador

**TABLE 2-1: INTEGER DATA TYPE SIZES AND LIMITS**

Type	Size	Minimum	Maximum
<code>char<sup>(1,2)</sup></code>	8 bits	-128	127
<code>signed char</code>	8 bits	-128	127
<code>unsigned char</code>	8 bits	0	255
<code>int</code>	16 bits	-32,768	32,767
<code>unsigned int</code>	16 bits	0	65,535
<code>short</code>	16 bits	-32,768	32,767
<code>unsigned short</code>	16 bits	0	65,535
<code>short long</code>	24 bits	-8,388,608	8,388,607
<code>unsigned short long</code>	24 bits	0	16,777,215
<code>long</code>	32 bits	-2,147,483,648	2,147,483,647
<code>unsigned long</code>	32 bits	0	4,294,967,295

**Note 1:** A plain *char* is signed by default.

**2:** A plain *char* may be unsigned by default via the `-k` command-line option.



# Tipos inteiros - x64

x86\_64 GNU/Linux

Var	Bytes
char	1
unsigned char	1
signed char	1
int	4
unsigned int	4
short	2
unsigned short	2
long	8
unsigned long	8

← /exemplos/Aula1/tamanhoTipos.c

# Tipos floating-point

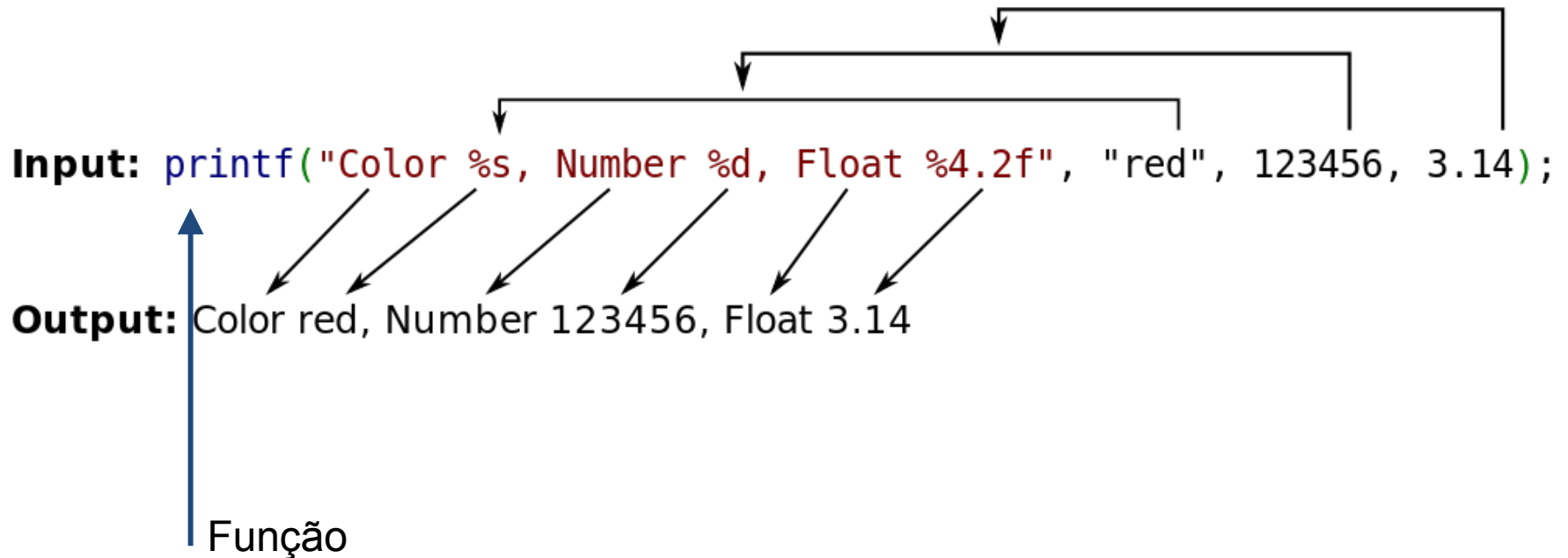
Type	Storage size	Value range	Precision
float	4 byte	1.2E-38 to 3.4E+38	6 decimal places
double	8 byte	2.3E-308 to 1.7E+308	15 decimal places
long double	10 byte	3.4E-4932 to 1.1E+4932	19 decimal places

## x86\_64 GNU/Linux

Var	Bytes	
long	4	← /exemplos/Aula1/tamanhoTipos..c
double	8	
long double	16	

**printf / scanf**

# printf - exibição de dados




[https://en.wikipedia.org/wiki/Printf\\_format\\_string](https://en.wikipedia.org/wiki/Printf_format_string)

# Printf

```
#include <stdio.h>

void main(){
    unsigned char contador;
    for(contador=0; contador < 10; contador++){
        printf("Val %d \n", contador);
    }
}
```

mutiraoC/Aula1/**printf.c**



```
Val 0
Val 1
Val 2
Val 3
Val 4
Val 5
Val 6
Val 7
Val 8
Val 9
```



## Exercício 2

(8 min)

Altere o código **printf.c** para exibir na tela um contador porém agora classificando se o número é ímpar ou par, exemplo :

Val 1 (ímpar)

Val 2 (par)

Val 3 (ímpar)

Val 4 (par)

(interrompa quando chegar em 1024)

# scanf - leitura de dados

/exemplos/Aula1/4-scanf.c

```
#include <stdio.h>

int main(void)
{
    int n;
    scanf("%d", &n);
    printf("\n0 valor inserido foi : %d \n", n);
    return 0;
}
```

mutiraoC/Aula1/**scanf.c**

./4-scanf  
12

0 valor inserido foi : 12

## Exercício 3

(8 min)

Faça um programa (em C) que calcule o juros compostos de um produto, o programa deve receber como parâmetros :

**PV**: Valor Presente

**i** : Taxa de Juros

**n** : Número de Períodos

e retornar o :

**FV**: Valor Futuro

Utilize a equação:

$$FV = PV \times ( 1 + i ) ^ n$$

[ref] <http://hcinvestimentos.com/2009/06/21/juros-compostos/?hvid=2BYUMA>

## Exercício 4 (para casa ?!)

Escreva um programa que receba do usuário 3 itens (nome, preço e quantidade) e formate o resultado similar a tabela:

Produto	Preço Unitário	Quantidade	Preço Total
Banana	R\$ 2.50	2	R\$ 5.00
Uva	R\$ 6.50	6	R\$ 39.00
Pessegue	R\$ 10.22	10	R\$ 102.20
		Sub-Total	R\$ 146.20
	Imposto (5%)		R\$ 7.31
		Total	R\$ 153.51

# Bibliografias

<http://www.slideshare.net/amraldo/introduction-to-c-programming-7898353>

[ftp://ftp.ufv.br/dma/tutoriais/c%2B%2B/introd\\_ling\\_c.pdf](ftp://ftp.ufv.br/dma/tutoriais/c%2B%2B/introd_ling_c.pdf)