

# Algoritmos e Programação (23-1)

## Introdução

Profs.: Andressa e Marina



# Noções Iniciais

- Programar
  - Organizar uma coleção de instruções **sequenciais** que executam uma tarefa específica no **computador**
  - Esta coleção é chamada de **programa**
- Programa
  - Escrito em uma **linguagem** de computador escolhida pelo programador: C, Python, Java, Haskell, entre outras
- Linguagens
  - São descritas em alto nível para que sejam entendidas por humanos (na maioria das vezes em inglês)



# Algoritmos

- Algoritmo é um termo mais genérico para definir programação
  - A diferença que um algoritmo não precisa ser criado utilizando uma linguagem de programação
  - Assim, um algoritmo, dependendo da forma que foi escrito, pode ser executado por uma pessoa

# Algoritmos

- Exemplo: algoritmo para fritar um ovo
  - 1 - Pegue uma frigideira coloque sobre a boca de um fogão
  - 2 - Coloque óleo na frigideira
  - 3 - Acenda a boca
  - 4 - Pegue um ovo
  - 5 - Quebre-o
  - 6 - Coloque sal sobre o ovo
  - 7 - Espere fritar





# Algoritmos

- Exemplo: algoritmo para fritar um ovo
  - 1- Pegue uma frigideira coloque sobre a boca de um fogão
  - 2- Coloque óleo na frigideira e acenda a boca onde se encontra a frigideira
  - 3- Pegue um ovo
  - 4- Quebre-o de tal forma que o conteúdo caia dentro da frigideira (jogue as cascas fora)
  - 5- Pegue um pouco de sal e jogue sobre o ovo
  - 6- Espere o ovo fritar conforme o seu gosto
  - 7- Tire da frigideira e coloque em um prato
- Perceba que a ordem das instruções é importante. Não podemos trocar, por exemplo, o passo 4 com o 3

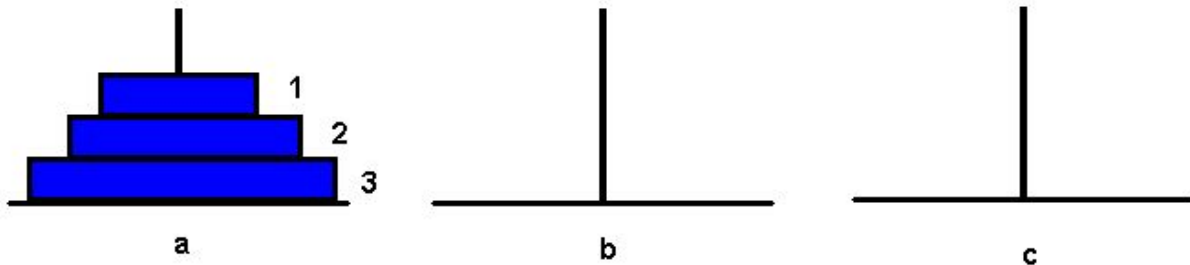


# Algoritmos

- Exercício 1:
  - Um homem precisa atravessar um rio com um barco que possui capacidade apenas para carregar ele mesmo e mais uma de suas três cargas: um lobo, um bode e um maço de alfafa. O que o homem deve fazer para conseguir atravessar o rio sem perder nenhuma das cargas?
  - Escreva um algoritmo que garanta uma travessia segura, considerando que o lobo não pode ficar sozinho com o bode e que o bode não pode ficar sozinho com a alfafa.

# Algoritmos

- Exercício 2:
  - Elabore um algoritmo que mova três discos de uma Torre de Hanói, que consiste em três hastes (a-b-c), uma das quais serve de suporte para três discos de tamanhos diferentes (1-2-3), os menores sobre os maiores. Pode-se mover um disco de cada vez para qualquer haste, contanto que nunca seja colocado um disco maior sobre um menor. O objetivo é transferir os três discos para outra haste.





# Programas

- Os primeiros programas eram implementados em linguagem de máquina
- Era muito difícil programar (décadas de 1940 e 1950)
- Por exemplo, um programa que fizesse os seguintes passos

```
Informe a sua idade: 20
Você pode votar
```



# Programas

- Em linguagem de baixo nível (máquina) ficaria assim:

Informe a sua idade: 20  
Você pode votar

Início

```
.LC0:  
  .string "Informe a sua idade: "  
.LC1:  
  .string "%d"  
.LC2:  
  .string "Voce pode votar"  
.LC3:  
  .string "Voce nao pode votar"  
.text  
.globl main  
.type main,@function
```

```
main:  
.LFB0:  
  .cfi_startproc  
  pushq %rbp  
  .cfi_def_cfa_offset 16  
  .cfi_offset 6, -16  
  movq %rsp, %rbp  
  .cfi_def_cfa_register 6  
  subq $16, %rsp  
  movq %fs:40, %rax  
  movq %rax, -8(%rbp)  
  xorl %eax, %eax
```

```
  leaq .LC0(%rip), %rdi  
  movl $0, %eax  
  call printf@PLT  
  leaq -12(%rbp), %rax  
  movq %rax, %rsi  
  leaq .LC1(%rip), %rdi  
  movl $0, %eax  
  call __isoc99_scanf@PLT  
  movl -12(%rbp), %eax  
  cmpl $16, %eax  
  jle .L2  
  leaq .LC2(%rip), %rdi
```

```
  movl $0, %eax  
  call printf@PLT  
  jmp .L3  
.L2:  
  leaq .LC3(%rip), %rdi  
  movl $0, %eax  
  call printf@PLT  
.L3:  
  movl $0, %eax  
  movq -8(%rbp), %rdx  
  xorq %fs:40, %rdx  
  je .L5
```

```
  call __stack_chk_fail@PLT  
.L5:  
  leave  
  .cfi_def_cfa 7, 8  
  ret  
.cfi_endproc  
.LFE0:  
  .size main,.-main  
  .ident "GCC: (Ubuntu 7.4.0-1ubuntu1~18.04.1) "  
  .section .note.GNU-stack,"",@progbits
```

Fim



# Programas

- Utilizando uma linguagem de alto nível:

Informe a sua idade: 20  
Você pode votar

//C

```
#include <stdio.h>
int main(void)
{
    int id;
    printf("Informe a sua idade: ");
    scanf("%d",&id);
    if (id>=16)
        printf("Você pode votar\n");
    else
        printf("Você não pode votar\n");
    return 0;
}
```

#python

```
id=input('Informe a sua idade: ')
id=int(id)
if id>=16:
    print('Você pode votar')
else:
    print('Você não pode votar')
```

{Pascal}

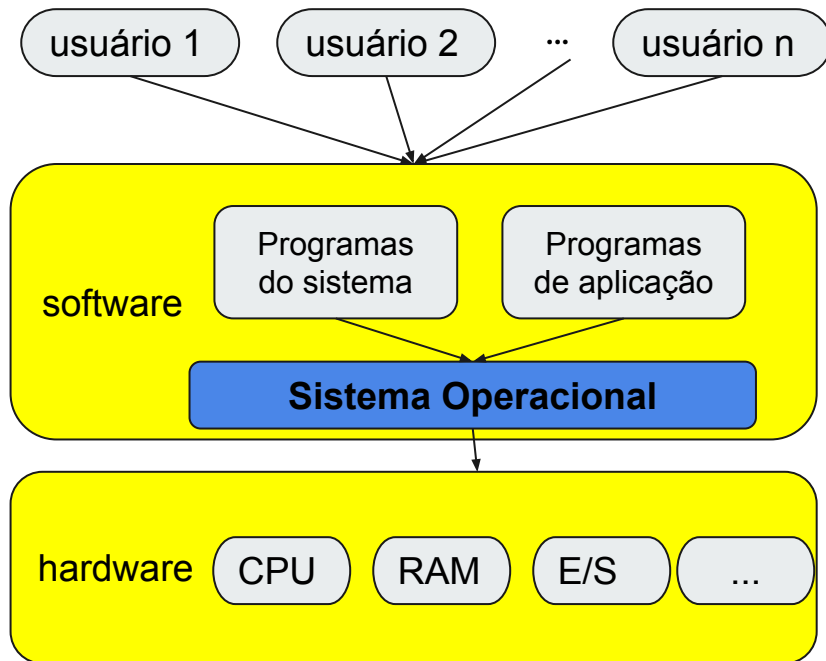
```
program vota;
var
    id : integer;
begin
    write('Informe a sua idade: ');
    readln(id);
    if id>=16
    then
        writeln('Você pode votar')
    else
        writeln('Você não pode votar')
end.
```



# Executando Programas

- Como um programa escrito em linguagem de alto nível é **entendível** pelo computador?
  - Existe **um programa mestre** (sistema) que torna os computadores mais acessíveis pelos humanos
  - Faz as transformações de nossas requisições de entrada (teclado, voz, mouse, ...) para o computador
  - Transforma as respostas do computador para um formato entendível por nós (monitor, impressora, som, vídeo, ...)
- Sistema operacional (SO)
  - Gerenciar os recursos de hardware e software do computador, e oferece serviços comuns para os programas serem executados

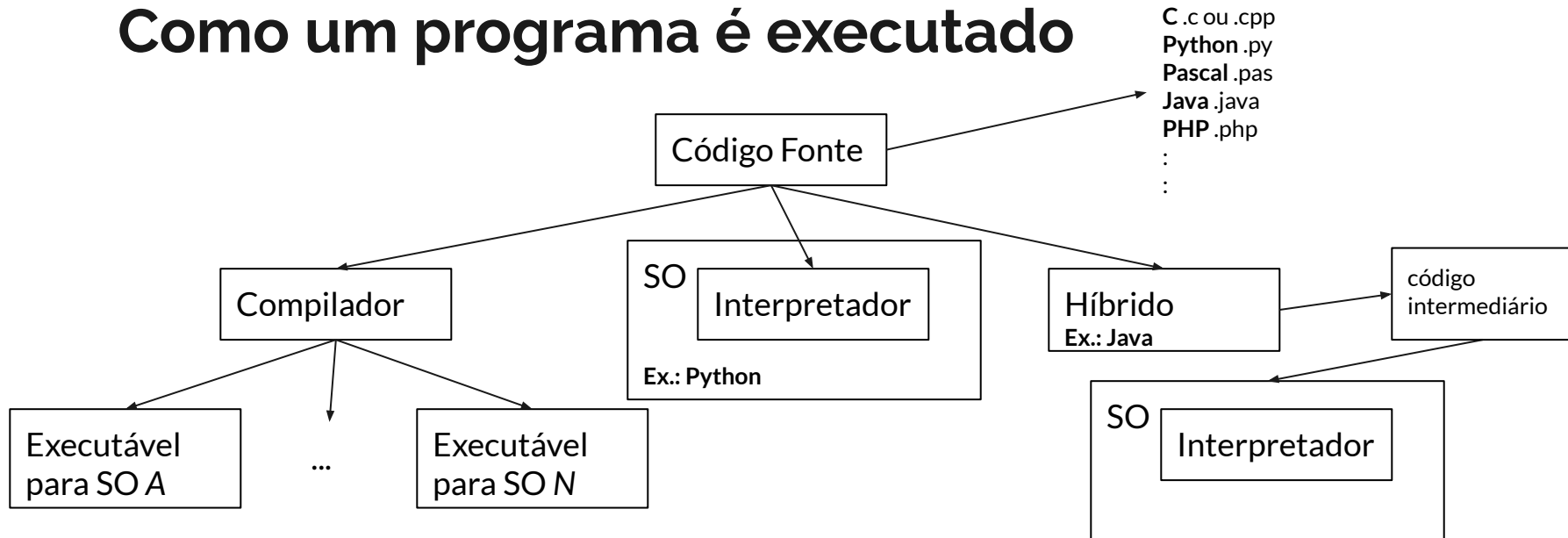
# Sistema Operacional (SO)



# Sistema Operacional (SO)

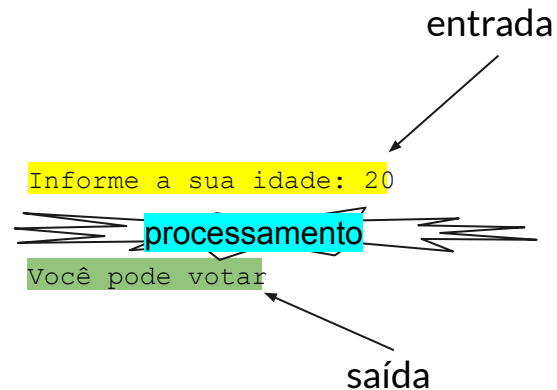


# Como um programa é executado



# Algoritmos

- Um algoritmo/programa é formado por 3 partes:
  - Entrada
  - Processamento
  - Saída



```
#python
id=input('Informe a sua idade: ')  entrada
id=int(id)                         processamento
if id>=16:
    print('Você pode votar')
else:
    print('Você não pode votar')
```

} saída



# Algoritmos

- Entrada
  - Informações (dados) que o programa precisa para resolver o problema
  - No nosso exemplo: **a idade do usuário**
- Processamento
  - Processar os dados de entrada para obter a saída
  - No nosso exemplo: verificar se a **idade informada é maior ou igual a 16 ou não** (idade para votar)
- Saída
  - Apresentar o resultado do processamento. Este resultado deve ser condizente com o problema que o programa está resolvendo
  - No nosso exemplo: imprimir uma mensagem informando se alguém com a idade digitada pode ou não votar.





# Entrada

- Os dados informados pelo usuário durante a entrada devem ser armazenados em uma região da memória principal do computador (chamada RAM)
  - Variável
  - O programador deve criar as variáveis dando nomes as mesmas:
    - O nome deve começar com uma letra. Não pode ter espaços, caracteres especiais (por exemplo: #, @, &, (, ), !, entre outros)
    - Nomes válidos: idade, tipo2, val\_hora, nomeCliente, minhaHora, salario, ...
    - Inválidos: nome cliente, #idade, salário, tempo(dia), eu&voce, ...
- As entradas geralmente são feitas pelo **teclado, mouse, telas do tipo touch screen, leitores de códigos de barras**, entre outros.



# Processamento

- Os dados de entrada, representados pelas variáveis são utilizados para produzir a saída
  - Expressões matemáticas
  - Expressões lógicas
  - Comandos condicionais
  - Comandos de iteração (laços)
  - :



# Saída

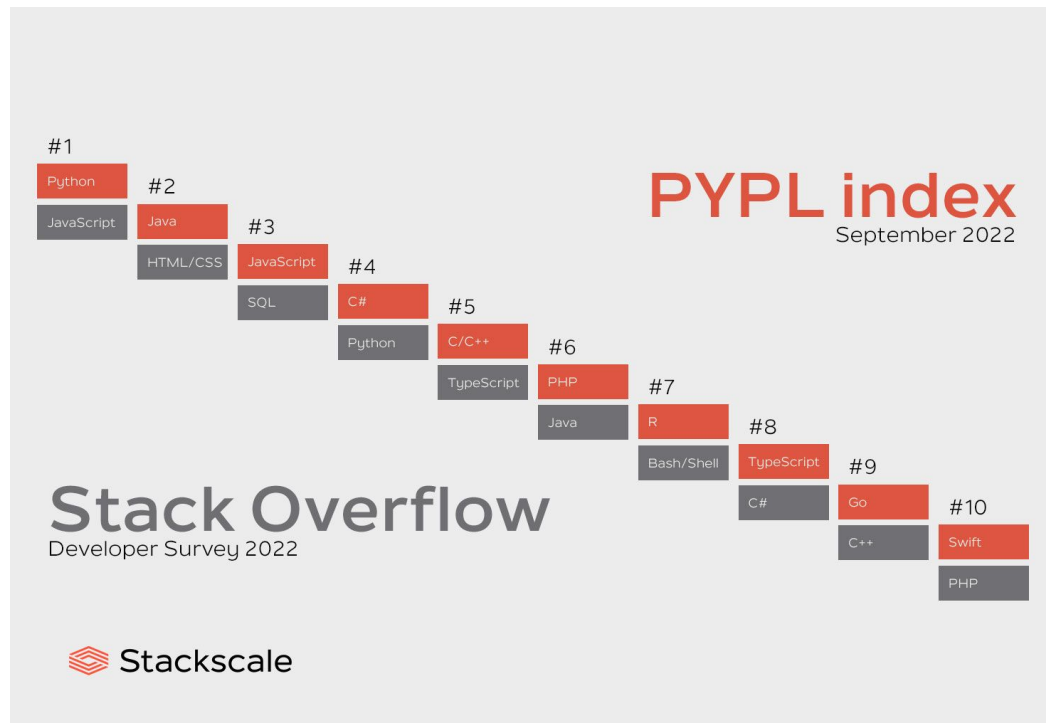
- Os resultados obtidos pelo processamento (que podem ser armazenados em variáveis) são apresentados para o usuário do aplicativo
  - Geralmente na tela do computador/celular através de comandos específicos (**print**)
  - Na impressora
  - Pelo alto falante (som)
  - Podem ser armazenadas diretamente no disco



# Exemplo

- Suponha um algoritmo para dado o ano do nascimento do usuário e outro ano qualquer, calcular a idade do usuário no ano qualquer
  - **Entrada:** devemos solicitar ao usuário que informe o ano de nascimento dele e também o ano para calcular a idade
    - Vamos armazenar o ano de nascimento na variável `anonasc` e o ano para calcular a idade de `anoatual`
  - **Processamento:** agora temos que encontrar a fórmula matemática para calcular a idade, pensando um pouco, sabemos que se subtrairmos o ano para calcular a idade (`anoatual`) pelo ano de nascimento (`anonasc`), encontraremos a idade do usuário. O resultado do cálculo deve ser armazenado em uma variável para ser utilizado na saída (`idade`)
    - `idade = anoatual - anonasc`
  - **Saída:** agora basta apresentarmos o resultado do cálculo
    - Imprimir o conteúdo da variável `idade`

# Algoritmos - Ranking linguagens



<https://pypl.github.io/PYPL.html>