Introdução à Programação

ano lectivo de 2006 / 2007

Sumario

- Introdução
- História da computação
- Funcionamento do computador
- Linguagens de Programação
- Desenvolvimento de algoritmos
  - Algoritmos não computacionais
  - Algoritmos computacionais
- Exercícios

O que é um computador?

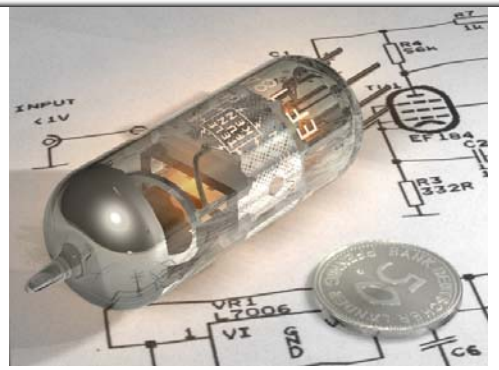
- Dicionário – Porto Editora
  - 1
    - que ou aquele que faz cômputos;
    - calculador;
    - calculista.
  - 2
    - aparelho electrónico que processa dados em função de um conjunto de instruções previamente fornecidas.

Computador

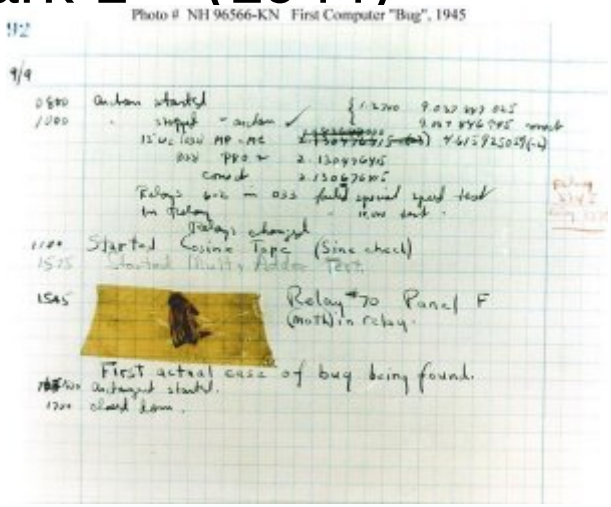
As cinco gerações

1ª Geração - válvulas

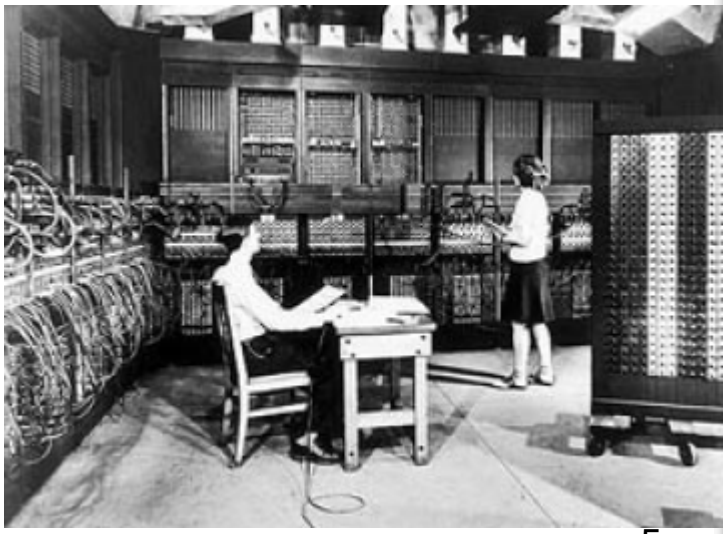
Mark 1 – (1944)



válvula

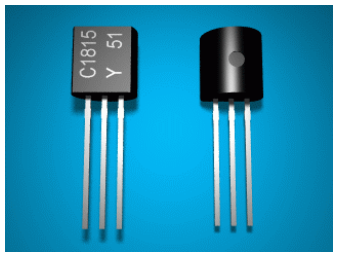


O primeiro Bug



ENIAC – (1945)

2ª geração - Transistor

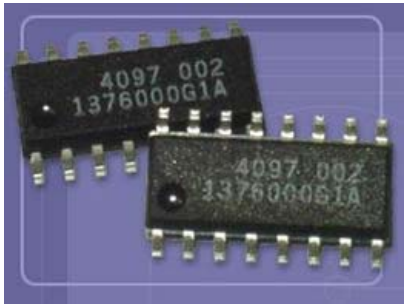


- Vantagens em relação às válvulas
  - Menor energia consumida
    - Menor aquecimento
  - Maior velocidade de processamento

UNIVAC (1956)

3ª geração - Circuitos integrados

- Componentes miniaturizados
  - Transistores
  - Resistores
  - Diodos
- Chips
  - Conjunto de componentes
- Circuitos integrados
  - Conjunto de chips
- Sistema Operativo - MS-DOS
- Graficos EGA – 16 cores



PC-XT (1981)

4ª geração - VLSI

- Integração de circuitos em larga escala
- Slots ISA de 16 bits
- Slots PCI
- Placas VGA e SVGA

PC-AT (1985)

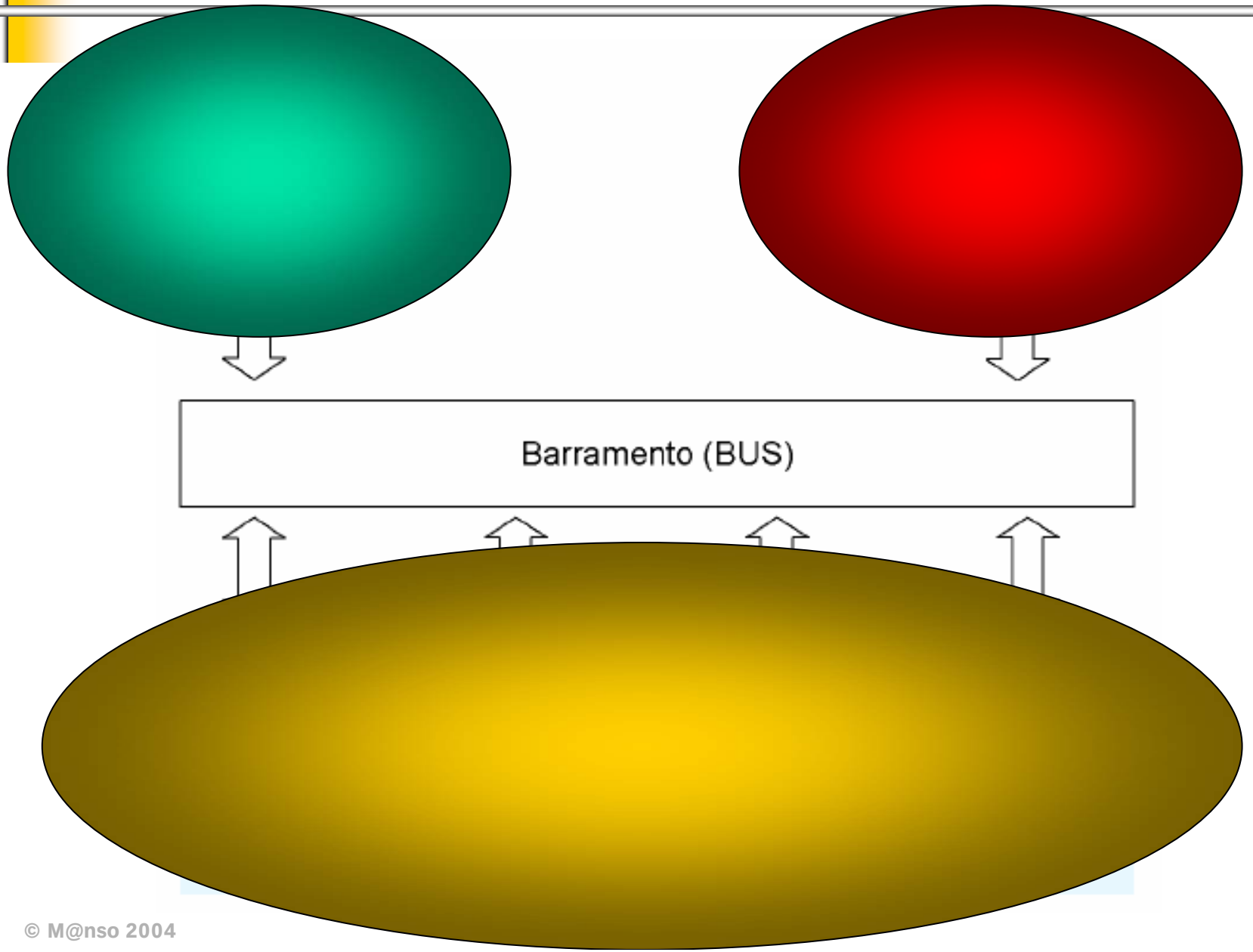
5ª geração - ULSI

- Integração em muito larga escala
- Processamento paralelo
- Slots AGP
- USB
- SATA

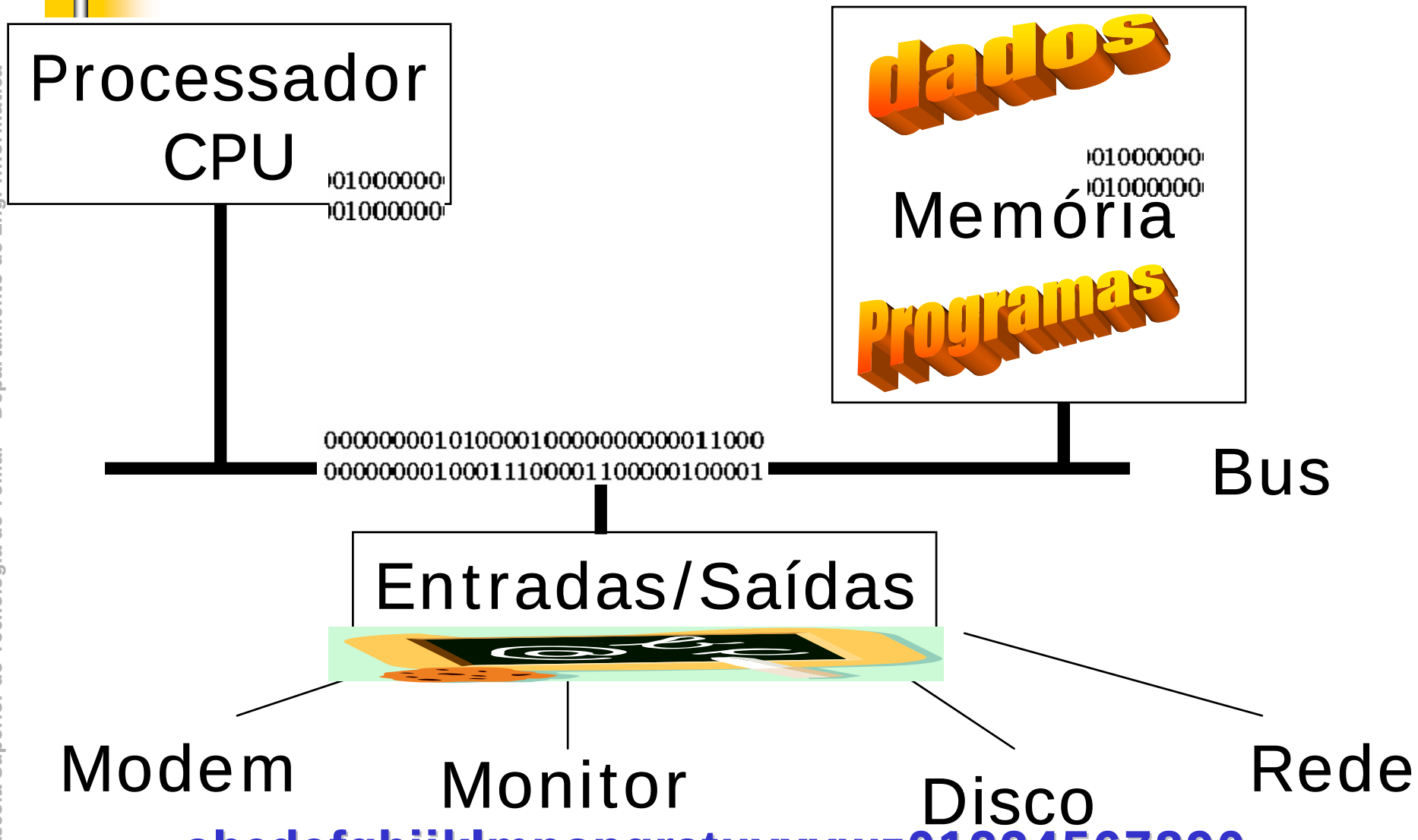
Computadores na actualidade

Funcionamento do computador digital

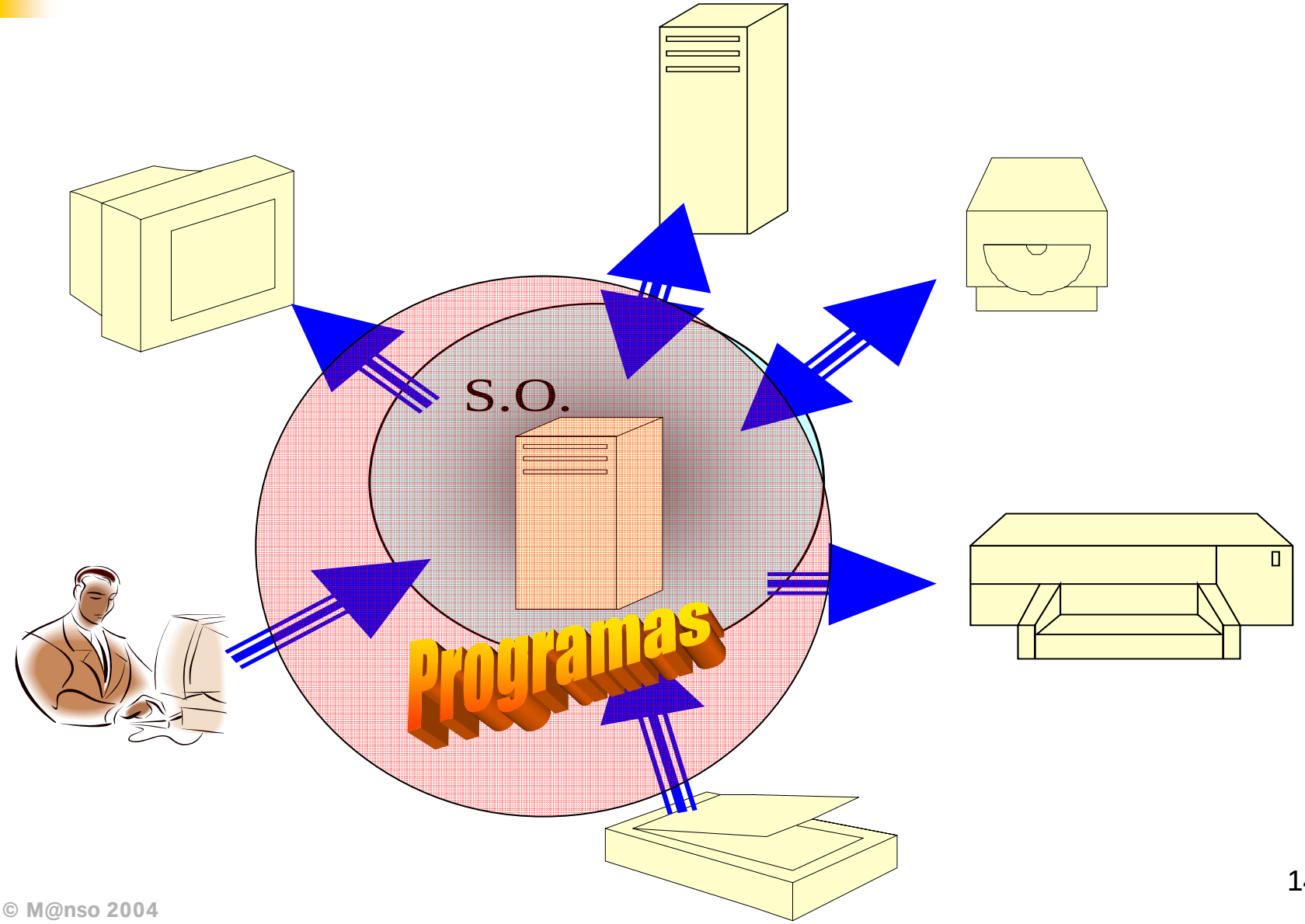
Modelo de Von Neuman



Arquitectura básica do Computador



Software / Hardware



Software

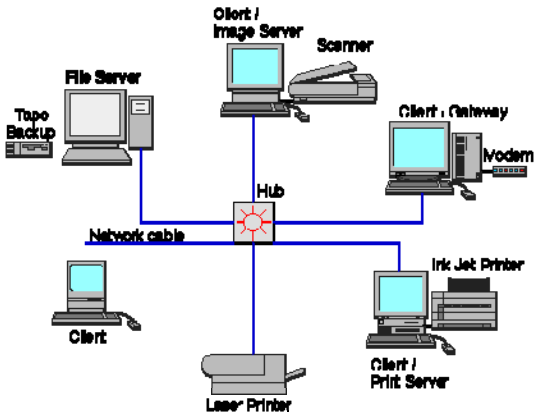
- Aplicativos
  - Realizam tarefas específicas
    - Processamento de texto
      - Microsoft Word
    - Desenho
      - Corel Draw
      - Autocad
    - Jogos
    - Etc.
- Sistema operativo
  - Serve de interface entre a máquina e os programa de aplicação

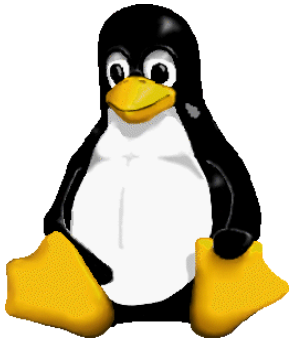
Tipos de sistemas informáticos

- Quanto ao tamanho e capacidade
  - Grande porte
    - Supercomputadores
    - Mainframes
  - Médio porte
    - Workstations
    - MiniComputadores
  - Pequeno porte
    - Microcomputadores
    - Ultra-Microcomputadores

Tipos de sistemas informáticos

- Quanto ao número de utilizadores e de tarefas
  - Mono utilizador
    - Monotarefa
      - Ex. PC com MS-DOS
    - Multitarefa
      - Ex. PC com Windows
  - Multi utilizador
    - Multi-posto
      - Servidor UNIX
      - Terminais
    - Redes de computadores
      - Internet

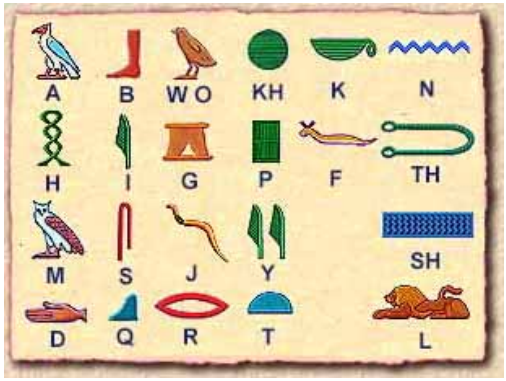




Linguagens de Programação

A linguagem que as máquinas entendem

Linguagens e alfabetos



Hello World

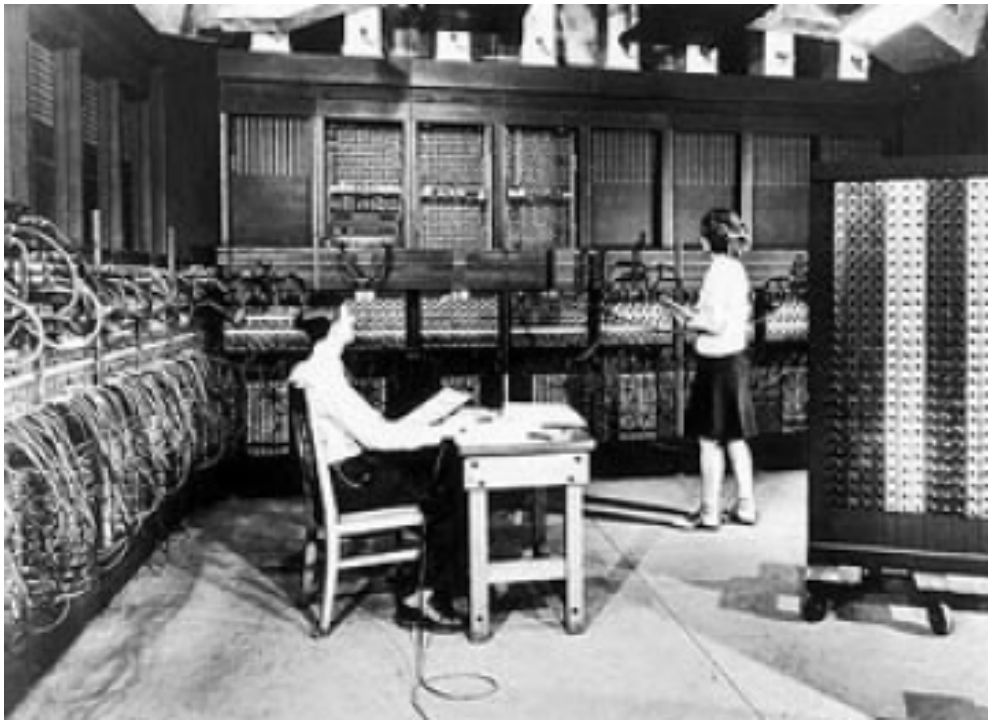
Olá Mundo

Bonjour Monde

Halo welt

1ª geração – Linguagem máquina

- 1ª geração – Linguagem máquina
  - Conjunto de dígitos binários do "*instruction set*" do processador
    - Os programas correm apenas no computador para o qual foram projectados.



```
0100 1010100
0100 1010110
0110 1001100
0101 1010101
1100 1001100
```

2ª geração – Assembler

- 2ª geração – Assembler
  - Mneumónicas do "*instruction set*" do processador
    - Assemblador – Programa que traduz o código assembly para linguagem máquina
  - Os Programas funcionam apenas num tipo processador
    - Mov –> 00001100
    - int -> 10001101
  - Desenvolvimento de programas muito difícil e demorado

```
dosseg
.model small
.stack 100h

.data
hello_message db 'Hello, World!','$'

.code
main  proc
      mov    ax,@data
      mov    ds,ax

      mov    ah,9
      mov    dx,offset hello_message
      int    21h

      mov    ax,4C00h
      int    21h
main  endp
end   main
```

Assembly para o IBM-PC

2ª geração – Assembler

```
printf:   pea      text
          move.w   #9,-(sp)
          trap     #1
          addq.l   #6,sp
          bra      printf

text:     dc.b     "Hello, World !",0
```

Assembly 680x0 on an Atari computer

```
reset
    LDX #$00
cycle
    LDA hworld,X
    BEQ reset
    STX cache
    JSR $FFD2
    LDX cache
    INX
    JMP cycle
hworld
.text "Hello, World!"
.byte 13,0
cache
.byte 0
```

Assembly on an Commodore 64

2ª geração – Assembler

- Desvantagens
  - Pequeno número de instruções
    - Programas longos
    - Pouco legíveis
    - Difíceis de modificar
  - Utiliza directamente os recursos da máquina
    - Os programas não são portáteis entre computadores
- Vantagens
  - Código optimizado
    - Velocidade de processamento elevado
  - Controlo total do hardware

3ª geração – Linguagens de alto nível

- 3ª geração – Linguagens de alto nível
  - Uma instrução pode corresponder a um grande número de instruções em assembly
    - Intruções em linguagem natural
      - write, read, print, . . .
      - Ler, escrever, repetir
  - Linguagens de propósito geral
    - Cálculo matemática
    - Gestão de documentos
    - Controlo
  - Exemplos
    - Basic
    - Pascal
    - C
    - Cobol
    - Fortran

```
10 print"Hello World!"
20 goto 10
```

Basic

3ª geração – Linguagens de alto nível

```c
#include <stdio.h>
main()
{
  for(;;)
    printf ("Hello World!\n");
}
```

C

```
program Hello_World;
Begin
  repeat
    writeln('Hello World!')
  until 1=2;
End.
```

Pascal

```
      PROGRAM HELLO
      DO 10, I=1,10
      PRINT *,'Hello World'
   10 CONTINUE
      STOP
      END
```

Fortran

```
100200 MAIN-LOGIC SECTION.
100300 BEGIN.
100400   DISPLAY " " LINE 1 POSITION 1 ERASE EOS.
100500   DISPLAY "HELLO, WORLD." LINE 15 POSITION 10.
100600   STOP RUN.
100700 MAIN-LOGIC-EXIT.
100800     EXIT.
```

Cobol

4ª Geração de Linguagens de aplicação

- 4ª geração – Linguagens de alto nível com aplicações a áreas concretas
  - Funções muito específicas
    - Gestão de bases de dados
    - Elaboração de relatórios
    - Geração de ecrãs
  - Exemplos
    - DBASE
    - SQL
    - CLIPPER

```
SET ECHO OFF
CLEAR
DO WHILE 1=1
     @1,1 SAY "Hello, World!"
ENDDO
```

DBASE

```
CREATE TABLE HELLO (HELLO CHAR(12))
UPDATE HELLO
       SET HELLO = 'HELLO WORLD!'
SELECT * FROM HELLO
```

SQL

5ª geração – Linguagens de muito alto nível

5ª geração – Linguagens de muito alto nível

- Programação declarativa
  - Declaração dos problemas
  - Métodos específicos de resolução dos problemas
- Linguagens de Inteligência Artificial
  - Prolog

```
hello :-
printstring("HELLO WORLD!!!!").

printstring([]).
printstring([H|T]) :- put(H), printstring(T).
```

Programação imperativa versus declarativa

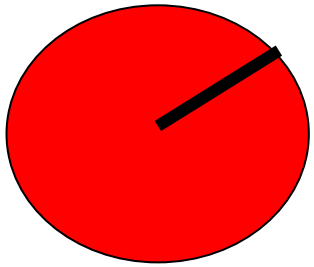
- Programação imperativa
  - "Qual é o procedimento que resolve o problema?"
    - Descrição pormenorizada de como um problema deve ser resolvido
      - Algoritmo
    - O computador segue os passos descritos no programa

- Programação declarativa
  - "Qual é o problema?"
    - O programador declara o conhecimento necessário para a resolução do problema
    - A linguagem possui métodos próprios para a resolução dos problemas.
      - inferência

Exemplo em linguagem imperativa

```
# include <stdio.h>
int main(int argc, char *argv[]) {
  double raio;
  printf(" raio do círculo :");
  scanf(%f",&raio);
  if( raio < 0 )
    printf (" ERRO – Raio negativo)
  else {
      printf("Perimetro do circulo:";
      printf("%f", 3.14* raio);
   }
}
```

Cálculo do Perímetro do círculo



Programação imperativa
Linguagem C

C:\Calculo.exe
Raio do circulo: 2
Perimetro do circulo: 6.28

Exemplos imperativa versus declarativa

```
Mae(iria, ines).
Mae(iria, antonio).
Mae(ines, tiago).

Irmao(X,Y) :- Mae(Z,X)
              Mae(Z,Y)

Avo(X,Y):- Mae(X,Z),
           Mae(Z,Y).
```

Relações familiares

Programação declarativa (prolog)

```
?: Avo( X,tiago)
   X= Iria.
?: Irmao( X,Y)
   X = ines
   Y = antonio
```

Resolução de problemas

Introdução à programação imperativa

Algoritmo

- Origem da palavra
  - al-Khwarizmi - Matemático árabe
    - Algoritmo
    - Algarismo

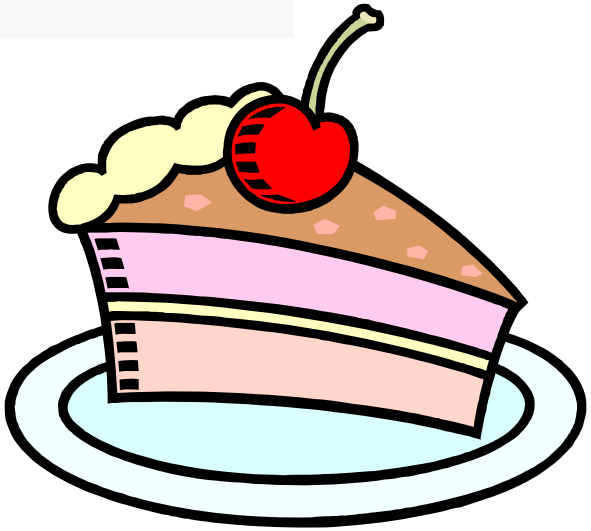
- Definição
  - É uma sequência finita de passos ou instruções, ordenadas de forma lógica, que levam a execução de uma tarefa ou solução de um problema.

Exemplo de um Algoritmo

- 250g de farinha
- 150g de margarina
- 5 ovos
- 2 colheres de fermento
- 200 gramas de acucar

1. Misturar os ingredintes
2. cozinhar o bolo.

Receita



Abordagem Top Down

Receita:

1 - Misturar os ingredintes
    1.1 – juntar a margarina e a farinha e bater até obter um creme
    1.2 – Juntar os ovos e mexer
    1.3 – juntar o fermento
2 –Cozinhar o bolo
    2.1 Aquecer o forno a 180ºc
    2.2 Cozer o bolo durante 45 min

- **Refinamento:**
  - **Obter creme**
  - **Juntar ovos**
  - **Ligar e regular o forno**
  - **Desligar o forno**

Pode um computador fazer um bolo ?

Algoritmos

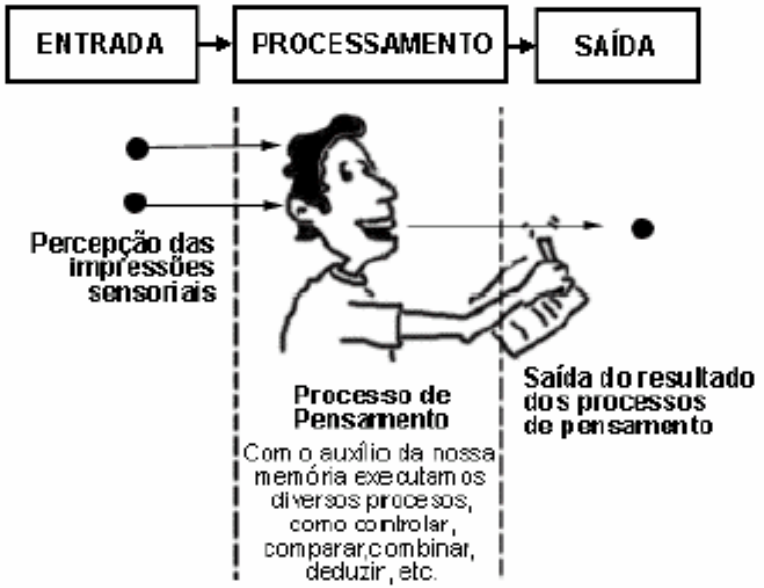
- Algoritmo não computational
  - Exemplos
    - Receita
    - Manual de instruções
  - Depende da perícia do utilizador!

- Algoritmo computational
  - Manipular informação
    - Receber dados
    - Guardar dados
    - Devolver informação
  - Executar instruções
    - Fazer operações aritméticas
    - Fazer operações lógicas
    - Escolha entre várias instruções.
    - Repetir um conjunto de instruções

Um algoritmo computacional é uma sequencia de passo tão bem definida que até um computador o é capaz de a executar

Componentes de um algoritmo

- Problema
  - Conjunto das possíveis entradas
  - Conjunto das saídas
  - Conjunto de operações válidas
- Solução Algorítmica
  - Conjunto ordenado de operações válidas que transformam o conjunto de entradas na saída desejada



Exemplo

- Como se constroem algoritmos?
- Problema
  - Trocar uma lâmpada fundida
- Algoritmo 1
  - Retirar a lâmpada fundida
  - Colocar a lâmpada boa

A formulação de um problema é frequentemente mais essencial do que a sua solução, a qual pode ser meramente uma questão de habilidade matemática ou experimental

Einstein

Exemplo

- Trocar uma lâmpada fundida

- Entrada
  - Lâmpada fundida
  - Lâmpada nova
  - Escada
- Saída
  - Lâmpada nova a funcionar
- Operações válidas
  - Retirar a lâmpada
  - Colocar a lâmpada
  - Subir a escada
  - Descer a escada

- Algoritmo 2
  - Subir a escada
  - Retirar a lâmpada fundida
  - Colocar a lâmpada boa
  - Descer a escada

Exemplo

- Trocar uma lâmpada fundida

- Entrada
  - Lâmpada fundida
  - Lâmpada nova
  - Escada
  - Caixote de reciclagem
- Saída
  - Lâmpada nova a funcionar
  - Lâmpada fundida na reciclagem
- Operações válidas
  - Retirar a lâmpada
  - Colocar a lâmpada
  - Subir a escada
  - Descer a escada
  - Deitar a lâmpada na reciclagem

- Algoritmo 2
  - Subir a escada
  - Retirar a lâmpada fundida
  - Colocar a lâmpada boa
  - Descer a escada
  - Colocar a lâmpada fundida na reciclagem

Exemplo

- Trocar uma lâmpada fundida

- Entrada
  - Lâmpada nova
  - Escada
  - Caixote de reciclagem
- Saída
  - Lâmpada nova a funcionar
- Operações válidas
  - Retirar a lâmpada
  - Colocar a lâmpada
  - Subir a escada
  - Descer a escada
  - Deslocar a escada
  - Deitar a lâmpada na reciclagem

- Algoritmo 3
  - Colocar a escada debaixo da lâmpada
  - Colocar a lâmpada boa no bolso
  - Subir a escada
  - Retirar a lâmpada fundida
  - Colocar a lâmpada boa
  - Descer a escada
  - Colocar a lâmpada fundida na reciclagem
  - Arrumar a escada

É um bom algoritmo ?

Características dos bons algoritmos

- Interagir com o utilizador
  - Realiza uma tarefa útil ao utilizador
  - Resolve o problema
- Ser finito
  - Termina sempre e com o resultado previsto
- Ser correctamente definido
  - Instruções claras que o utilizador consegue compreender e seguir
- Ser eficaz
  - Resolve sempre o problema mesmo nas situações mais problemáticas
- Ser eficiente
  - Utiliza o mínimo de recursos possível

É um bom algoritmo ?

- Interagir com o utilizador
- Ser finito
- Ser correctamente definido
- Ser eficaz
- Ser eficiente

- Algoritmo 3
  - Colocar a escada debaixo da lâmpada
  - Colocar a lâmpada boa no bolso
  - Subir a escada
  - Retirar a lâmpada fundida
  - Colocar a lâmpada boa
  - Descer a escada
  - Colocar a lâmpada fundida na reciclagem
  - Arrumar a escada

Algoritmo ou algoritmos ?

1. Inicio
2. Subir a escada
3. Fim

1. Inicio
2. Suba um degrau
3. Suba um degrau
4. Suba um degrau
5. Suba um degrau
6. Suba um degrau
7. Suba um degrau
8. . . . .
9. Fim

Algoritmo ou algoritmos ?

Instruções válidas:

- **Subir um degrau**
- **Verificar se está no topo**

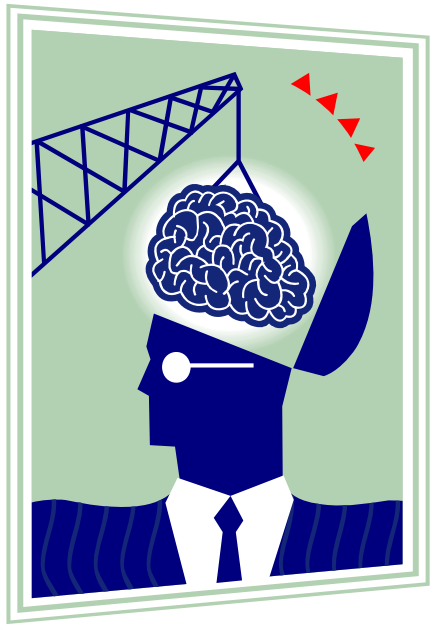
1. **Inicio**
2. **Suba um degrau**
3. **Se não chegou ao topo vá para 2**
4. **Fim**

1. **Inicio**
2. **Repita**
3. **Suba um degrau**
4. **Até chegar ao topo**
5. **Fim**

1. **Inicio**
2. **Enquanto não chegar ao topo**
3. **Suba um degrau**
4. **Fim**

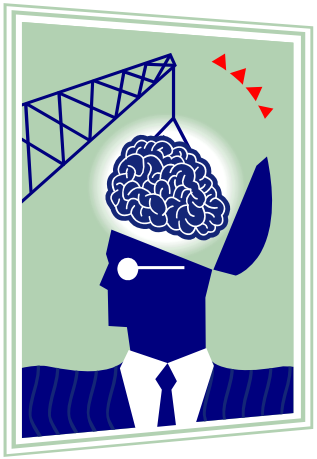
Conclusão

- O algoritmo não é a solução de um problema
  - É uma forma de chegar á solução
- Não se aprende
  - A copiar algoritmos
  - Ler algoritmos prontos
  - A decorar algoritmos
- Aprende-se
  - Construindo algoritmos
  - Testando algoritmos



Conclusão - Construir Algoritmos

- Qual é o problema.
  - O que pretendemos do algoritmo
- Definir quais são os dados que entram
  - Qual a situação inicial
  - O que é necessário para resolver o problema
- Definir quais são os dados que saem
  - Qual a situação final
  - Que resultados devem ser apresentados
- Definir o Algoritmo
  - Definir quais as instruções disponíveis/necessárias
  - Organizar as instruções de forma a resolver o problema
    - transformar as entradas na saída
- Testar o algoritmo
  - Verificar se resolve o problema
  - Verificar se resolve todos os casos
- Optimizar o algoritmos
    - Verificar se não utiliza recursos supérfluos



Exercícios

Reunião de negócios

- Imagine que uma pessoa decidia ir de táxi a uma reunião de negócios. Monte uma sequência de acções para que ela chegue ao prédio onde vai ocorrer a reunião.

a)Entrar no prédio da reunião;
b)Sair do táxi;
c)Acenar para que o táxi pare;
d)Perguntar o preço da corrida;
e)Informar o destino ao motorista;
f)Esperar o táxi;
g)Pagar ao taxista;
h)Entrar no táxi.

- Esperar o táxi
- Acenar para que o táxi pare
- Entrar no táxi
- Informar o destino ao motorista
- Perguntar o preço da corrida
- Pagar ao taxista;
- Sair do táxi;
- Entrar no prédio da reunião

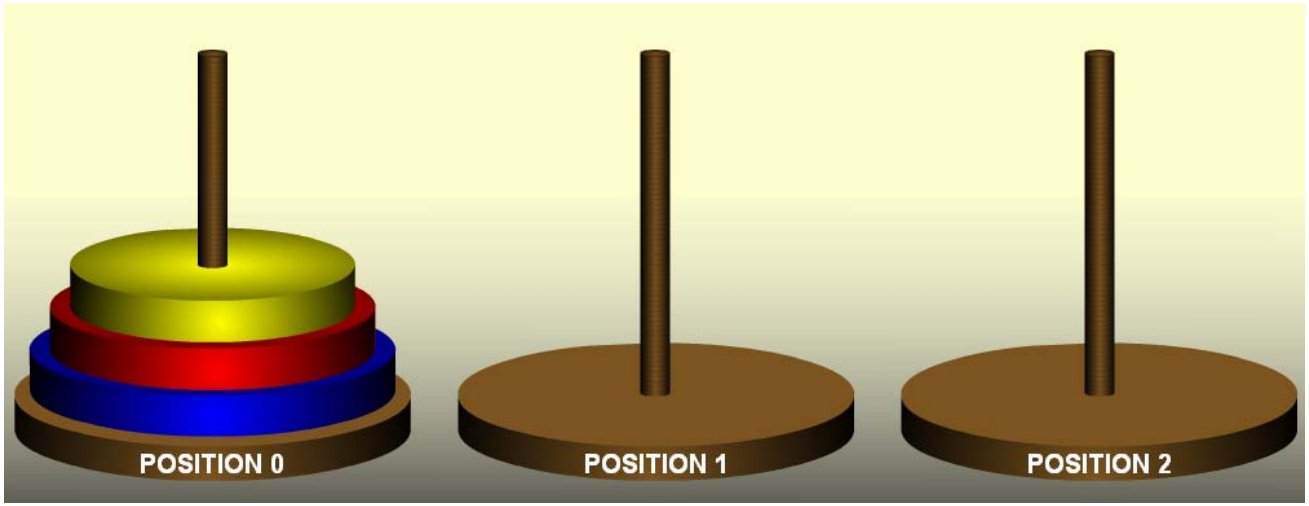
Algoritmos

- Escreva um algoritmo para trocar o pneu de um carro
- Escreva um algoritmo par meter gasolina num posto de abastecimento Self-service.
- Escreva um algoritmo para telefonar de uma cabine pública.

Algoritmos

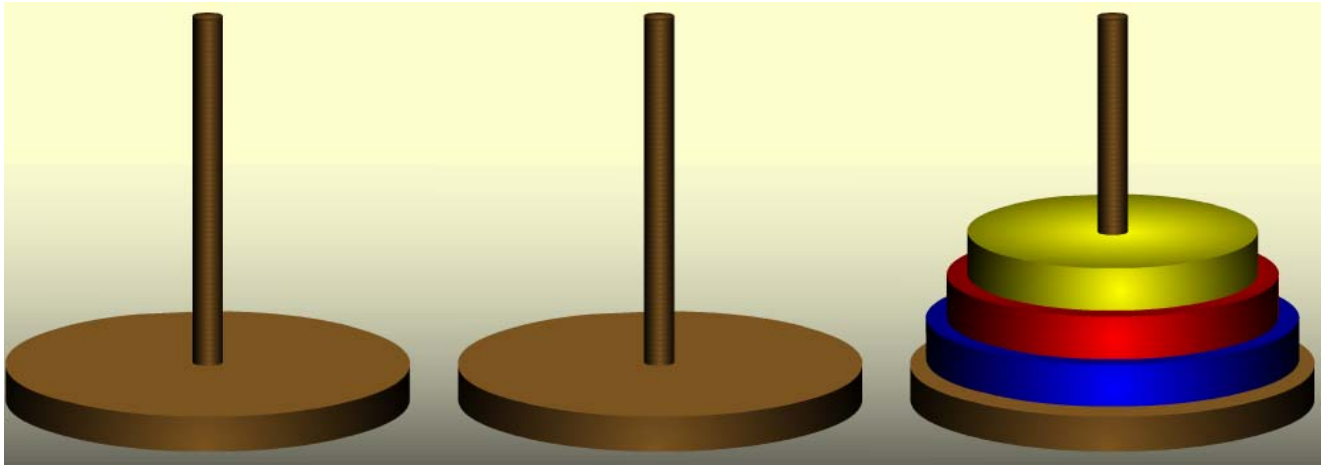
- Um barqueiro que possui uma lancha que leva 2 pessoas faz a travessia de um rio. Escreva um algoritmo que resolva o problema da travessia de 3 pessoas.
- O mesmo problema com 10 pessoas
- O mesmo problema com n pessoas
- A lancha leva 4 pessoas

Torres de Hanoi

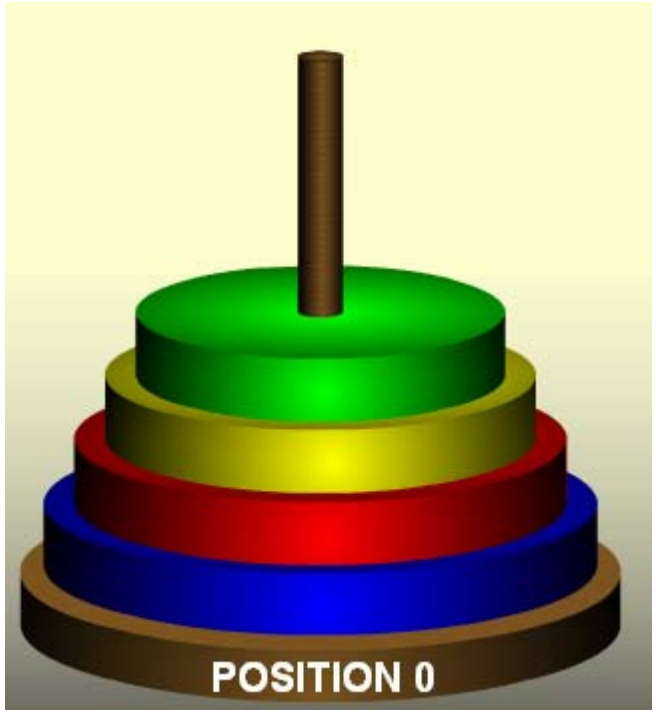


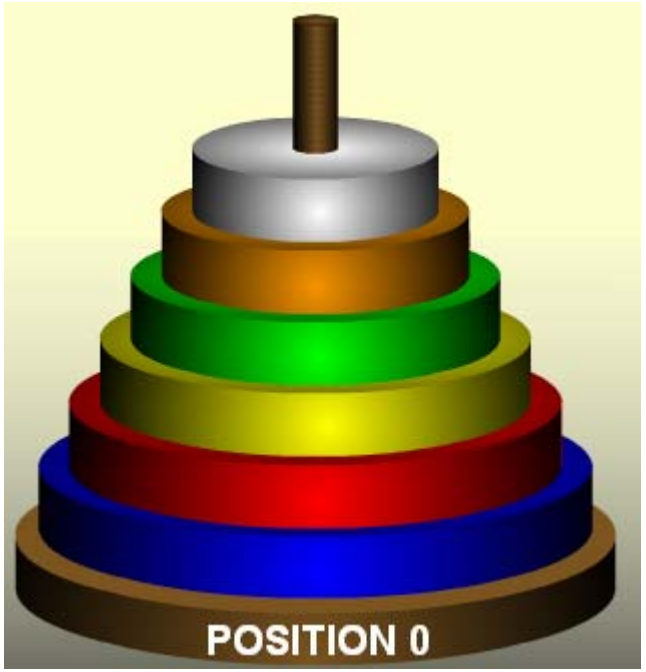
Mover Cor para Destino

- mover amarelo para 2



Torres de Hanoi





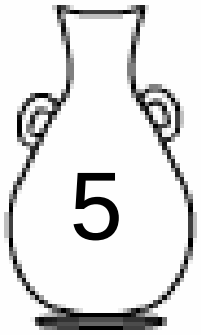
O homem o lobo o carneiro e as alfaces

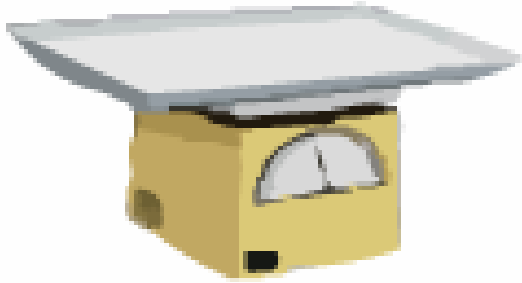
- Um homem, um lobo, um carneiro e um saco de alfaces encontram-se do mesmo lado de um rio e querem atravessá-lo. No mesmo lado do rio há uma canoa que devido ao seu estado de degradação apenas consegue transportar dois itens de cada vez. Como fazer a travessia, tendo em conta que o lobo não pode ficar sozinho com o carneiro e o carneiro não pode ficar sozinho com o saco das alfaces .




Problema das Jarras

- Existem duas jarras, uma de 3 litros e uma de 5 litros, ambas vazias e nenhuma delas tem qualquer marcação de medida. Existe uma fonte que pode ser usada para encher ou despejar as jarras com água.
- Como se colocam exactamente 4 litros de água na balança?





Missionários e Canibais

- Três missionários e três canibais encontram-se do mesmo lado de um rio e querem atravessá-lo. No mesmo lado do rio há uma canoa que devido ao seu estado de degradação apenas consegue transportar duas pessoas de cada vez. Os missionários acham que os canibais os atacam se ficarem em maior número e por isso querem elaborar um plano de travessia em que nunca os canibais ficam em maior número nas margens do rio ou no barco.
- O mesmo para N missionários e N canibais

O jogo do rio

- Encontram-se na margem A, a mãe e duas filhas, Pai e dois filhos e um polícia e um ladrão, que pretendem atravessar para a margem B. Para realizar a travessia existe uma jangada cuja lotação é de duas pessoas. Durante a travessia devem ser respeitadas as regras seguintes:
  - A jangada apenas pode ser conduzida pelo Pai, pela Mãe ou pelo Polícia.
  - A Mãe nunca pode ficar na mesma margem na presença de um ou mais filhos sem a presença do Pai.
  - O Pai nunca pode ficar na mesma margem na presença de uma ou mais filhas sem a presença do Mãe.
  - O ladrão deve estar sempre acompanhado pelo polícia quando existem na mesma margem outros elementos.
- Construa o algoritmo da travessia