

Introdução

IF61A/ IF71A - Computação 1

Prof. Leonelo Almeida

Universidade Tecnológica Federal do Paraná

Agenda de hoje

- Organização da disciplina
- Introdução à computação
- Algoritmos

Organização da disciplina

- Materiais e avisos estarão disponíveis em:

<http://dainf.ct.utfpr.edu.br/~leonelo>

Um pouco de história

- Ábaco
 - Babilônia 2400 a.c.



Um pouco de história

- Primeira máquina calculadora
 - Wilhelm Schickard (1592-1635)
- Máquina calculadora
 - Blaise Pascal (1623-1662)
 - Contabilidade:
 - apenas somas e subtrações



Um pouco de história

- Tear de Jacquard (1801)
 - Programável com cartões perfurados
 - Mesma máquina produzia estampas diferentes



Um pouco de história

- Charles Babbage (1837)
- Conceito de “tecer números”
- Máquina analítica: dispositivo projetado para ser programável
 - Possuía as características de um computador moderno

Um pouco de história

- Charles Babbage (1837)
- Conceito de “tecer números”
- Máquina analítica: dispositivo projetado para ser programável
 - Possuía as características de um computador moderno

Mas sua construção não chegou a ser finalizada

Um pouco de história

- Ada Lovelace (1815 - 1852)
- Primeiros programas de computador
- Parceria com Babbage
- Inventou conceitos importantes como:
 - Sub-rotinas
 - Sequências de instruções
 - Laços (loops)
 - Saltos condicionais

Um pouco de história

- Máquina de classificação de Hollerith (1880-1890)
 - Contagem do censo americano de 1890
 - Dados coletados em cartões perfurados
 - Censo de 1880 levou 7 anos
 - O de 1890 levou 2,5 anos

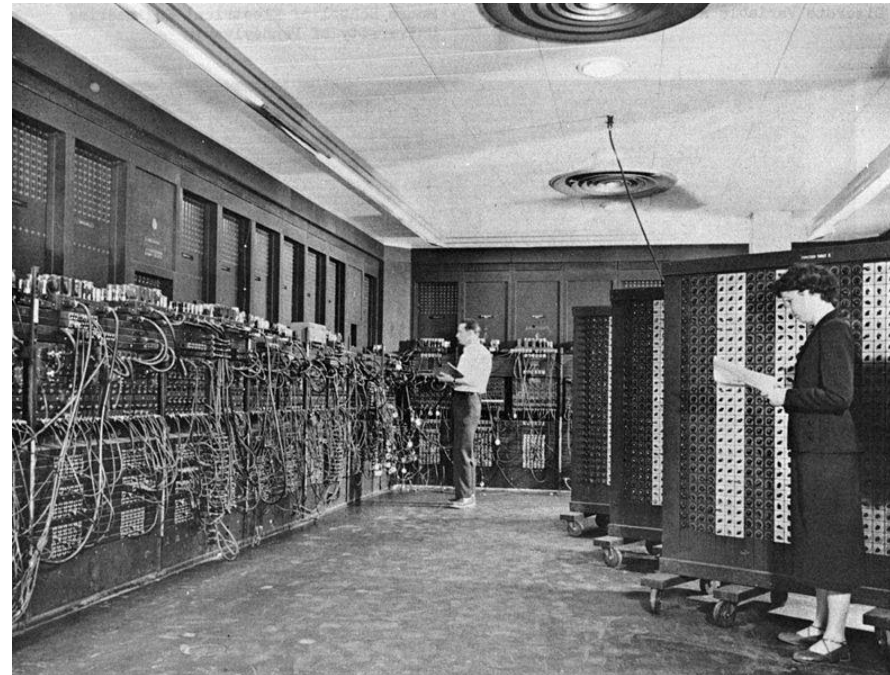


A computação e as guerras

- Segunda guerra mundial
 - Harvard e Marinha americana: Mark I
 - Mark I: pesava 5 toneladas, 17m de comprimento, 2.5 de altura
 - Capacidade: multiplicar dois números de dez dígitos em três segundos

- Exército americano: ENIAC

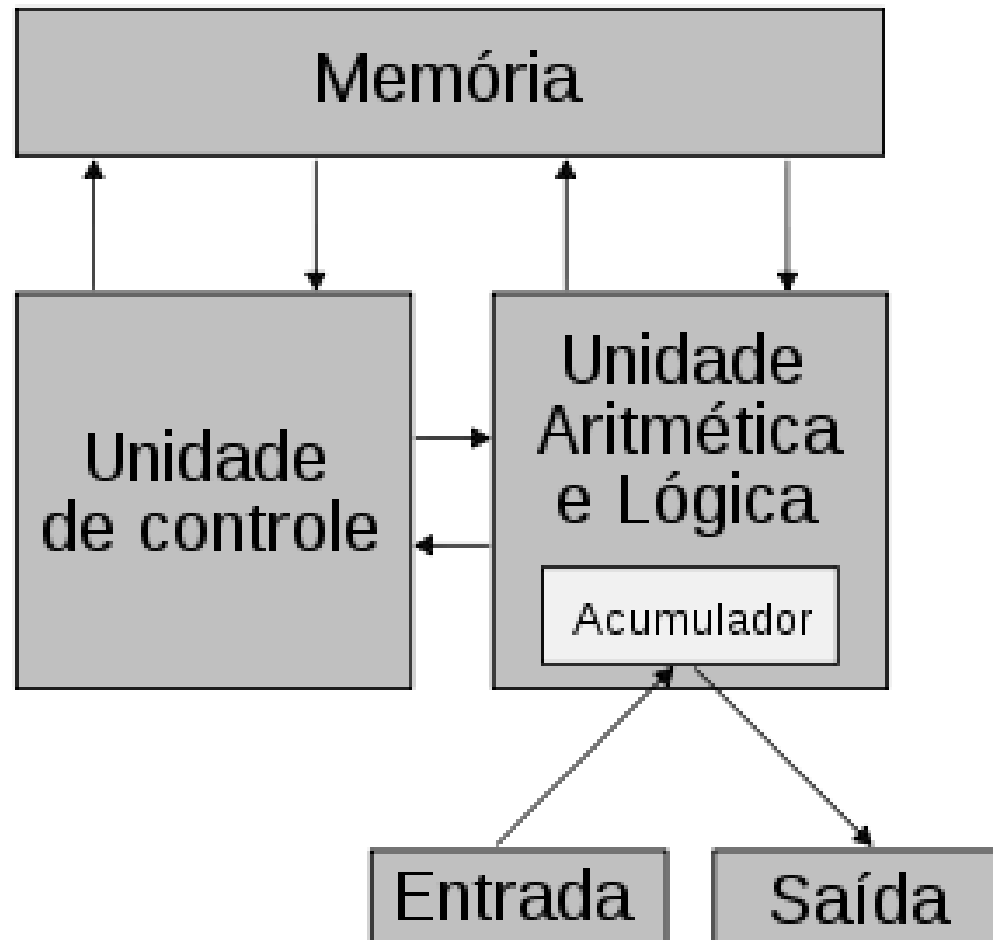
- Pesava 30 toneladas
- **Uma avaria a cada 6 horas**
- Consumia muita energia: baixava luz nas cidades vizinhas
- Capacidade: 500 multiplicações por segundo



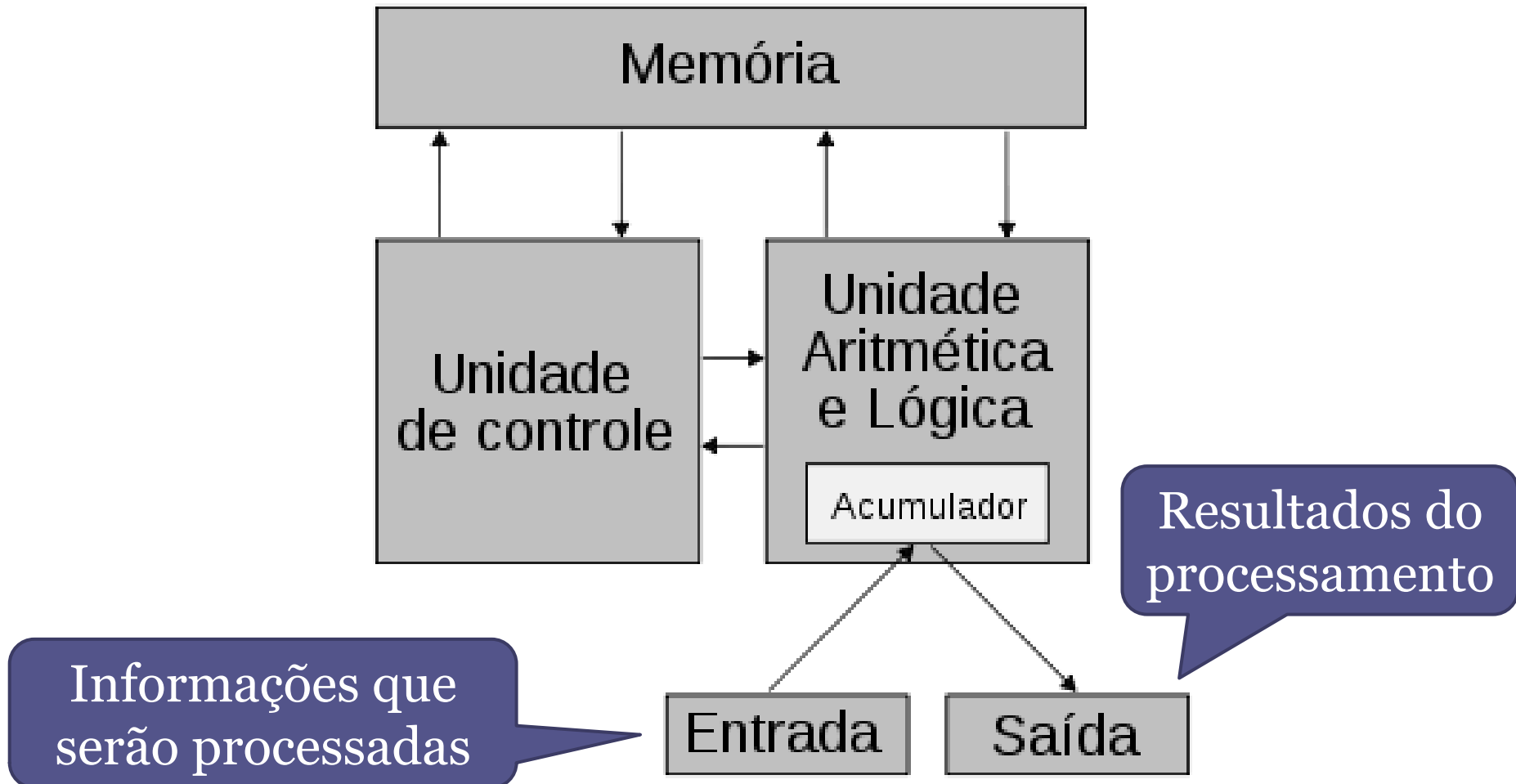
Ciência da Computação

- Alan Turing (1912-1954)
 - Máquina de Turing: calcular qualquer número e função de acordo com um conjunto de instruções
 - Formalização de **algoritmos**
- Von Neumann (1903-1957)
 - Projeto lógico de um computador
 - Armazenamento em memória
 - Comportamento passo-a-passo determinístico, ou seja, dadas as mesmas entradas e um mesmo programa tem-se as mesmas saídas sempre

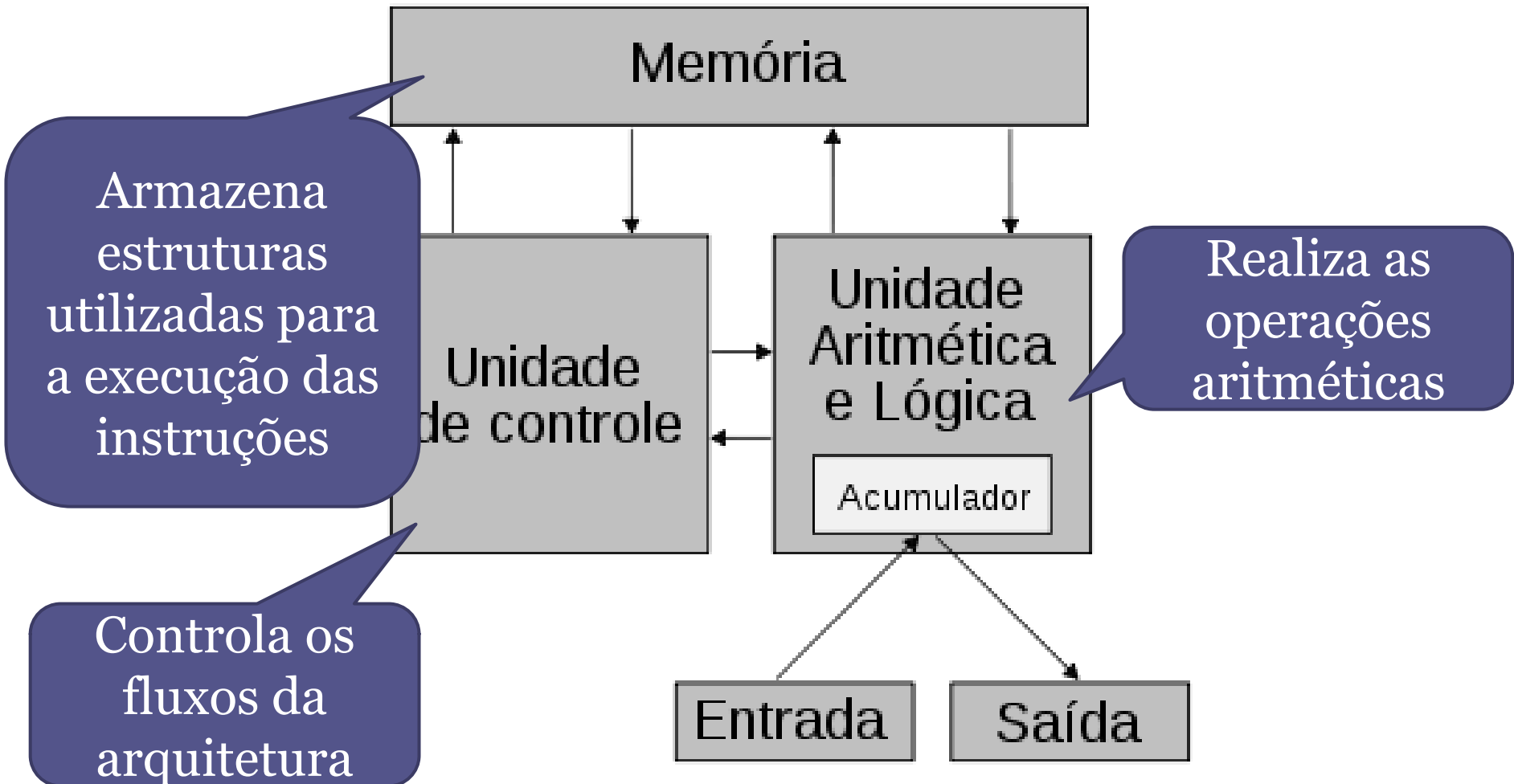
Projeto Lógico de Von Neumann



Projeto Lógico de Von Neumann



Projeto Lógico de Von Neumann



Até os anos 70

- Computação restrita a *mainframes*
- Grandes corporações

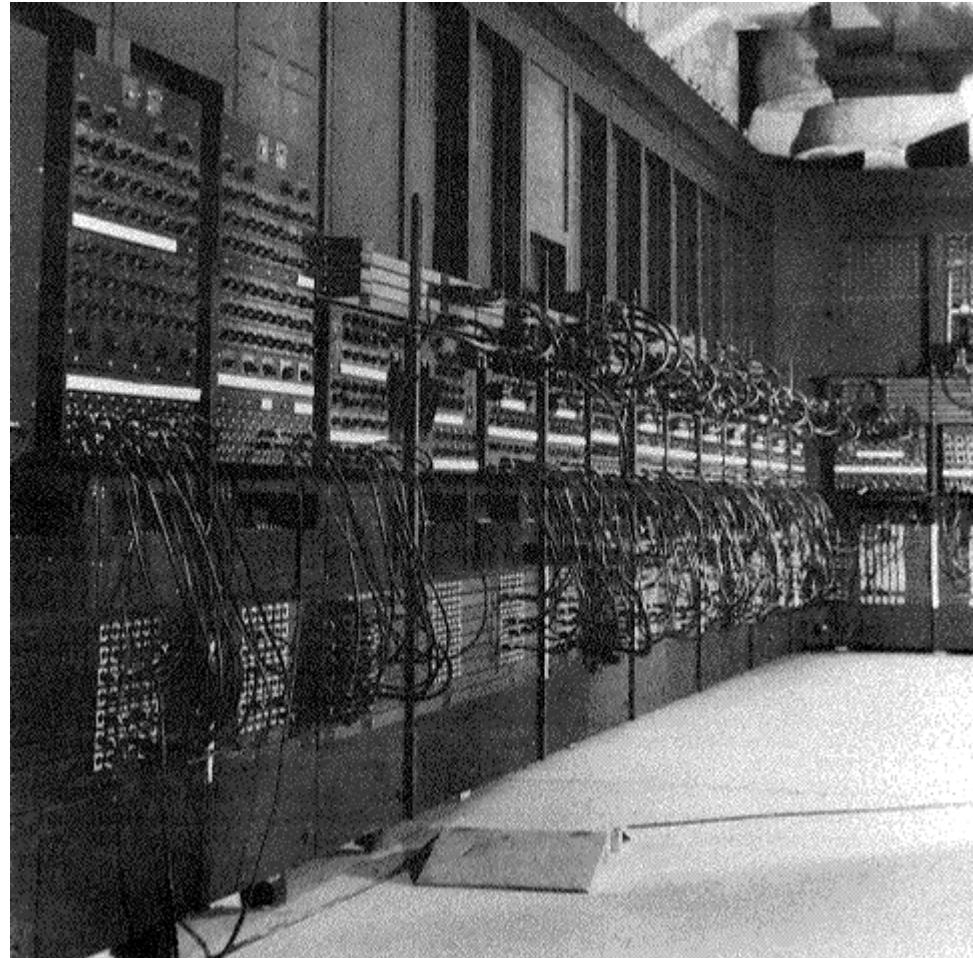


Figura: ENIAC 2

Computadores pessoais



- Altair 8800 (vendido com um kit de montar)
- Bill Gates e Paul Allen “criaram” a linguagem Basic para o Altair
 - Criação da Microsoft
- Steve Jobs e Steve Wozniak: criaram a Apple
- Mais sobre isso no filme:
 - “Piratas do Vale do Silício” (1999)



Hoje

- Fazendas de servidores
- Hologramas 4D
- Interfaces Cérebro-Máquina
- Computação Quântica



Hoje

- Fazendas de servidores
- **Hologramas 4D**
- Interfaces Cérebro-Máquina
- Computação Quântica



Hoje

- Fazendas de servidores
- Hologramas 4D
- Interfaces Cérebro-Máquina
- Computação Quântica



Hoje

- Fazendas de servidores
- Hologramas 4D
- Interfaces Cérebro-Máquina
- Computação Quântica

Princípios da Física Quântica

Ainda não disponível de maneira completa

Qbits x bits da arquitetura tradicional

- Bits: assume 1 de 2 valores: 0 ou 1
- Q-bits: assume um terceiro valor, o 01
- Aplicações:
 - Inteligência artificial
 - Problemas matemáticos complexos

Neste curso

- Aprenderemos como resolver problemas que podem ser programados no computador
- Aprenderemos como traduzir as soluções de problemas para programas de computador

PROGRAMAS são **MEIOS**
para resolver
PROBLEMAS que são o **FIM**

Para que aprender a programar?

- Computação está em praticamente todas as atividades profissionais
- Colabora para a:
 - Automatização de processos
 - Simulação de projetos
 - Realização de cálculos em tempo reduzido e livres de erros humanos
- Mesmo que não pretendam se tornar programadores, é necessário para especificar programas para eles

Conteúdo

- Resolução de problemas
- Conceitos básicos de programação
 - Algoritmos
 - Programas
 - Variáveis e coleções de variáveis
 - Estruturas de repetição e de decisão
 - Funções
 - Manipulação de arquivos
- Utilizaremos a linguagem de programação C

Fora do escopo deste curso

- Uso de aplicativos como
 - Windows,
 - Linux,
 - Office,
 - Firefox,
 - etc

Como aprender a programar computadores?

Não se aprende a tocar violão lendo livros,
mas sim **praticando**.

A mesma lógica se aplica à programação.

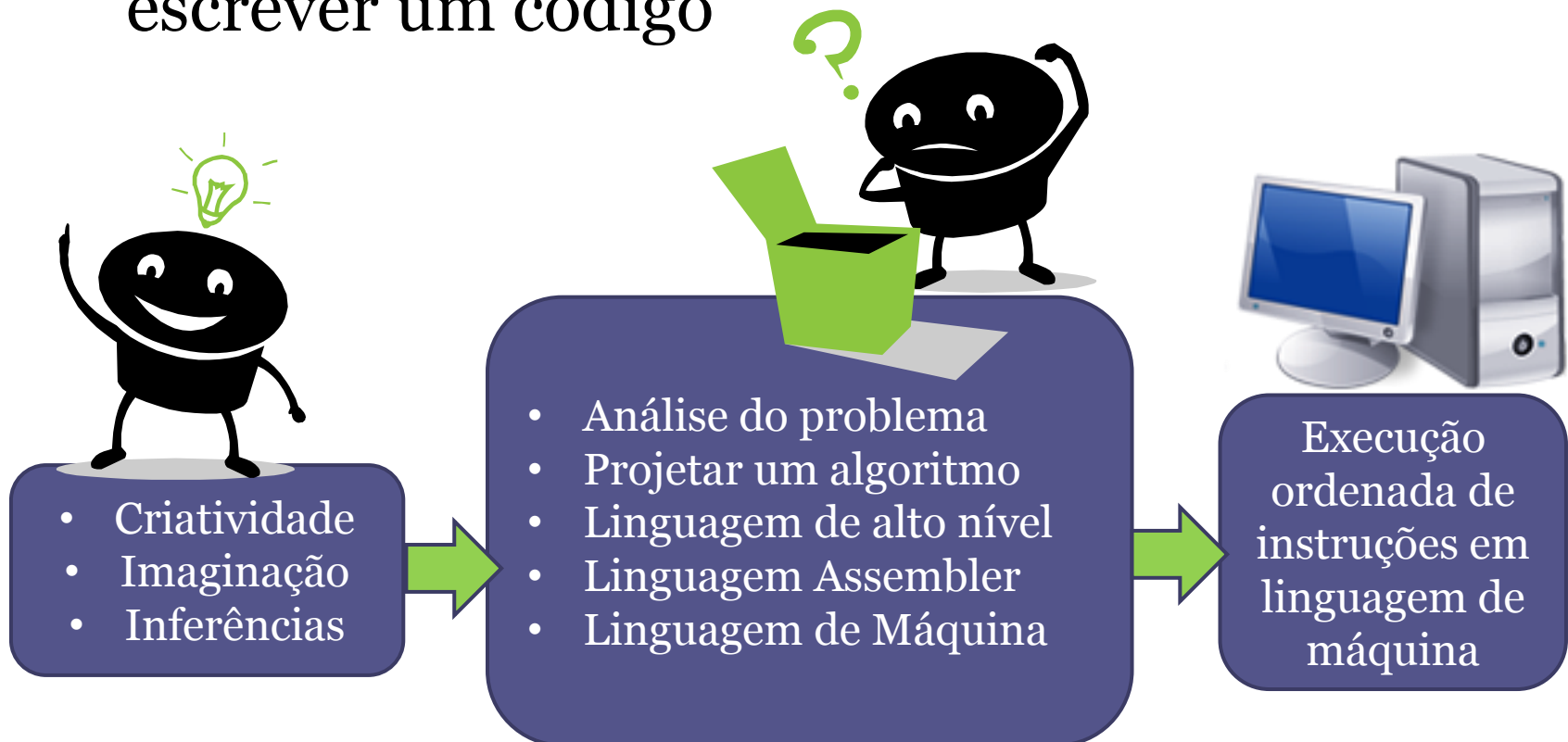
- Entender os conceitos apresentados. **Não deixe nada para trás!**
- **Fazer** as listas de exercícios
- Usar outros materiais como livros e Web
- Tirar dúvidas nos horários de PA ou com monitores do DAINF

O que é programar?

- Vai além de sentar em frente do computador e escrever um código

O que é programar?

- Vai além de sentar em frente do computador e escrever um código



Exemplo de resolução de problemas

- Calcular o dobro de um dado número.

Exemplo de resolução de problemas

- Calcular o dobro de um dado número.
- Passos:
 - Pedir para que o usuário informe o número
 - Multiplicar o número informado por 2
 - Informar ao usuário o resultado da multiplicação



Foco no problema
Sem formalismos

Fácil compreensão

Exemplo de resolução de problemas

- Calcular o dobro de um dado número

Algum formalismo
Fácil compreensão
Não depende do
computador

- Algoritmo:

Início

```
inteiro: numero, dobro;  
imprima("Informe o número:");  
leia(numero);  
dobro <- numero * 2;  
imprima("O dobro é: ", dobro);
```

Fim

Exemplo de resolução de problemas

- Calcular o dobro de um dado número.

- Programa em C:

```
#include <stdio.h>
int main( ) {
    int numero, dobro;
    printf("Informe o número:");
    scanf("%d", &numero);
    dobro = numero * 2;
    printf("O dobro é: %d", dobro);
    return 0;
}
```

Linguagem de alto nível

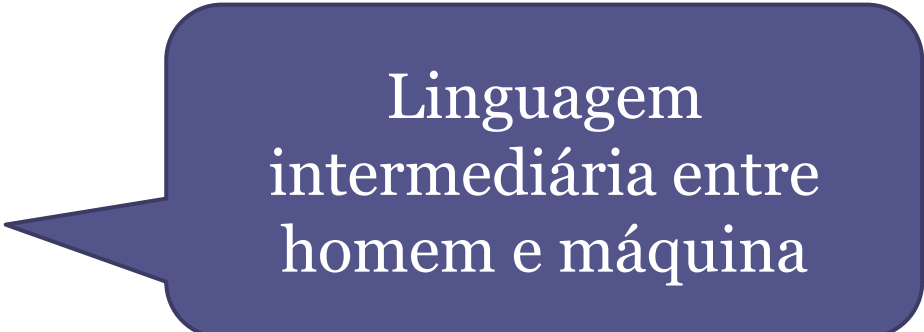
Instruções seguem a
sintaxe da linguagem

Exemplo de resolução de problemas

- Calcular o dobro de um dado número.

- Linguagem Assembler:

```
INICIO: IN A, (TECLADO)
LD (NUMERO), A
LD A, 2
MUL A, (NUMERO)
LD (DOBRO), A
LD A, (DOBRO)
OUT A, (NUMERO)
JMP INICIO
```



Linguagem
intermediária entre
homem e máquina

Exemplo de resolução de problemas

- Calcular o dobro de um dado número.
- Linguagem de Máquina:

03

00

30

02

03

01

31

...

Pronto para ser usado
pelo computador

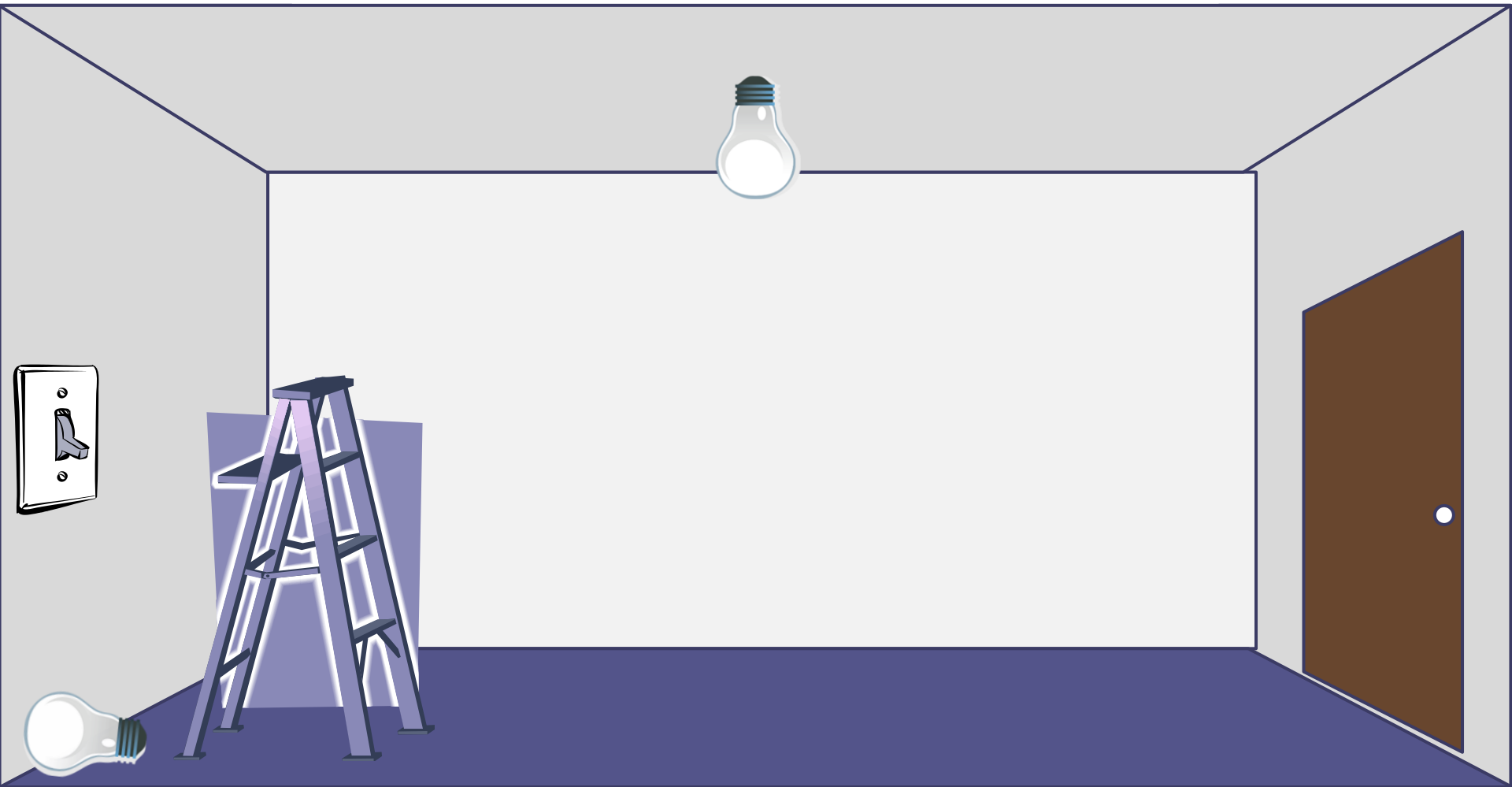
Não é legível por nós

Estruturas de dados

- São as informações necessárias para a resolução do problema
- Podem ser:
 - Informadas como **ENTRADAS** para o problema
 - Serem as **SAÍDAS** geradas como resultados da solução
 - Usadas para armazenar resultados **intermediários** à solução dos problemas

Exemplo

- Qual seria a sequência de passos para trocar a lâmpada dessa sala?

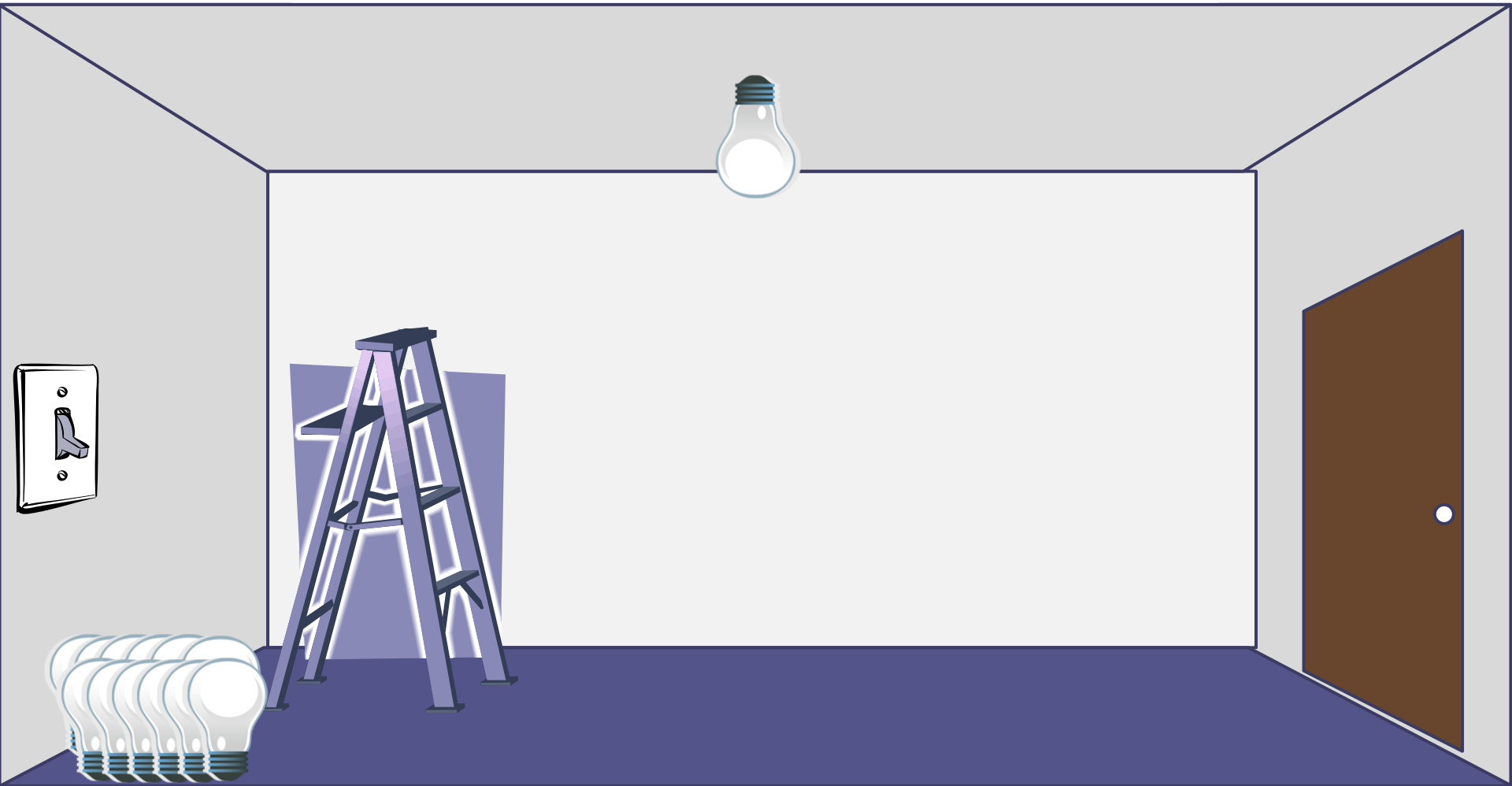


Exemplo

- Qual seria a sequência de passos para trocar a lâmpada dessa sala?
 1. Abra a porta
 2. Entre na sala
 3. Posicione a escada sob a lâmpada
 4. Suba a escada
 5. Retire a lâmpada queimada
 6. Desça a escada
 7. Troque pela lâmpada nova
 8. Suba a escada
 9. Coloque a lâmpada nova
 10. Fim

Exemplo 2

- Agora suponha que as lâmpadas novas também pudessem estar queimadas ...

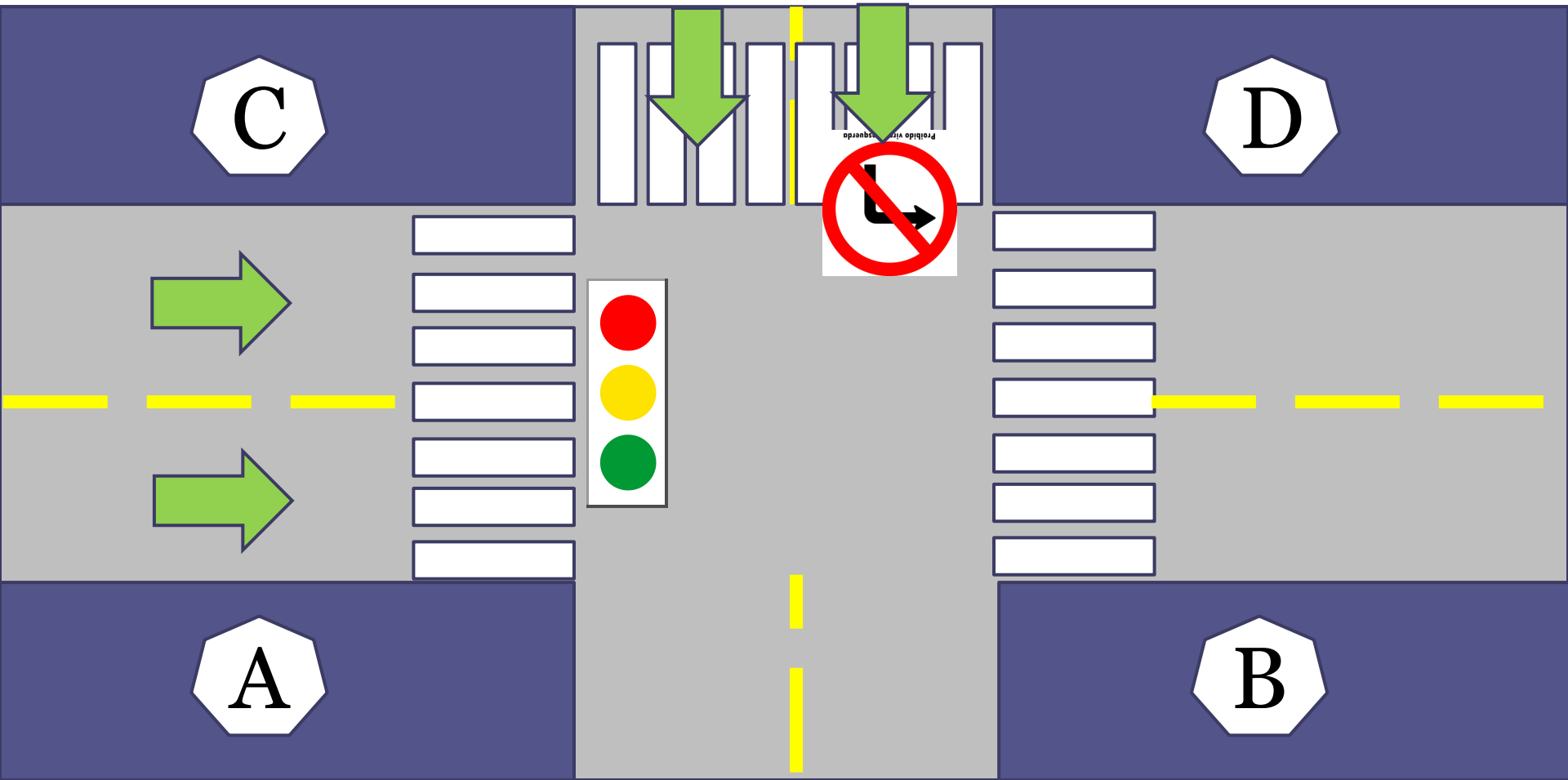


Exemplo 2

- Agora suponha que as lâmpadas novas também pudessem estar queimadas ...
 1. Abra a porta
 2. Entre na sala
 3. Posicione a escada sob a lâmpada
 4. Suba a escada
 5. Retire a lâmpada queimada
 6. Desça a escada
 7. Troque pela lâmpada nova
 8. Suba a escada
 9. Coloque a lâmpada nova
 10. Teste o interruptor
 11. Enquanto a lâmpada estiver queimada volte para 4
 12. Fim
- 

Exemplo 3

Especifique os passos para que uma pessoa que está no local (a) possa ir, em segurança, até o local (b). Para tal, observe o sentido do tráfego na encruzilhada, a faixa, o semáforo dos carros e a placa de trânsito.



Algoritmos

“Um algoritmo é um conjunto de regras para se obter uma saída específica a partir de uma entrada específica. Cada passo deve ser precisamente definido de maneira que possa ser traduzido para uma linguagem de computador e executado por uma máquina.”

Tradução livre de Donald Knuth

Algoritmos - Entendendo o conceito ...

- Conjunto de regras
- Uma saída específica
- Uma entrada específica
- Traduzidos para uma linguagem de computador
- Executado por uma máquina

- Precisamente definidas
- Descritas em passos sequenciais
- Livres de ambiguidade

O policial alcançou o rapaz correndo.

A senhora viu a moldura da janela.

Algoritmos - Entendendo o conceito ...

- Conjunto de regras
- **Uma saída específica**
- Uma entrada específica
- Traduzidos para uma linguagem de computador
- Executado por uma máquina



A solução do problema

Algoritmos - Entendendo o conceito ...

- Conjunto de regras
- Uma saída específica
- **Uma entrada específica**
- Traduzidos para uma linguagem de computador
- Executado por uma máquina

As informações
necessárias para resolver
o problema

Algoritmos - Entendendo o que é

- Conjunto de regras
- Uma saída específica
- Uma entrada específica
- Traduzidos para uma linguagem de computador
- Executado por uma máquina

Exemplo:

Calcular o dobro de um número inteiro informado pelo usuário.

Contraexemplo:

Calcular o dobro de 10.

O objetivo é construir algoritmos que resolvam de maneira genérica um problema.

Etapas para construção de um algoritmo

1. Ler o enunciado do problema com **MUITA** atenção
2. Identificar as entradas de dados
3. Identificar as saídas de dados
4. Determinar o que será feito para transformar as entradas (etapa 1) em saídas (etapa 2)
 1. Verificar se é necessário gerar valores internos/intermediários e a necessidade de criação de variáveis
 2. Identificar todas as transformações necessárias
5. Construir o algoritmo
6. Executar e testar o algoritmo

Etapas para construção de um algoritmo - Exemplo

Problema: Crie um algoritmo para ordenar quaisquer três números inteiros informados pelo usuário

- **Entradas:** três números inteiros.
- **Saídas:** os mesmos três números inteiros da entrada, mas ordenados.
- **O que será feito:** comparação entre os números informados para saber a ordem. As possibilidades de saída correspondem a uma permutação entre os números ($n!$).
- **Valores internos?** Não são necessários.

Etapas para construção de um algoritmo

- **Construção – Estrutura básica**

Nome: fazerAlgumaCoisa

Início

[declaração de variáveis]

/* No formato:

Tipo de dado: nomeVariável1, nomeVariável2, ...*/

[sequência de passos]

/*Podem ser operações de entrada e saída,
atribuições, blocos condicionais e de repetição*/

Fim

Etapas para construção de um algoritmo - Exemplo

- **Construção**

Nome: ordenar3Inteiros

Início

*/*valores informados pelo usuário*/*

inteiro: num1, num2, num3;

imprima(“Informe o primeiro valor: ”);

leia(num1);

...

Fim

Testes de mesa

- Servem para testar o funcionamento de um algoritmo sem a necessidade de escrever um código executável em linguagem de programação
- Permitem analisar passo-a-passo o algoritmo e:
 - as alterações em variáveis
 - avaliação de expressões lógicas
 - estruturas condicionais
 - estruturas de repetição

Testes de mesa - como realizar?

- Numere sequencialmente as linhas de seu algoritmo

nome: calculaAreaRetangulo

inicio

real: ladoA, ladoB, area;

1. imprima(“Informe o primeiro lado (em centímetros):”);
 2. leia(ladoA);
 3. imprima(“Informe o segundo lado (em centímetros):”);
 4. leia(ladoB);
 5. $area \leftarrow ladoA * ladoB$;
 6. imprima(“A área do retângulo é: ”, area, “centímetros”);
- fim

Testes de mesa - como realizar?

- Agora crie uma tabela linhas de código x variáveis

nome: calculaAreaRetangulo

inicio

real: ladoA, ladoB, area;

1. imprima("Informe o primeiro lado: ");

2. leia(ladoA);

3. imprima("Informe o segundo lado: ");

4. leia(ladoB);

5. area <- ladoA*ladoB;

6. imprima("A área do retângulo é: ", area, " centímetros ");

fim

Linha	ladoA	ladoB	area
1			
2			
3			
4			
5			
6			

Testes de mesa - como realizar?

- Simule a execução de cada passo com entradas e certifique se a saída é a esperada

nome: calculaAreaRetangulo
inicio

real: ladoA, ladoB, area;

1. imprima("Informe o primeiro lado: ");

2. leia(ladoA);

3. imprima("Informe o segundo lado: ");

4. leia(ladoB);

5. $area \leftarrow ladoA * ladoB$;

6. imprima("A área do retângulo é: ", area, "centímetros");

fim

Linha	ladoA	ladoB	area
1	?	?	?
2	5	?	?
3	5	?	?
4	5	1.5	?
5	5	1.5	7.5
6	5	1.5	7.5

Testes de mesa

- Podem se tornar muito longos tanto em linhas como em colunas
- Recomenda-se, para algoritmos maiores, avaliar trechos do algoritmo
 - Você pode selecionar um bloco de código significativo (por ex.: um condicional, uma estrutura de repetição, um bloco com um resultado parcial)

Algoritmos com qualidade

1. Devem ser legíveis por outras pessoas (e não somente por quem o criou)
 1. Use recuo (indentação) de blocos de código
 2. Use espaços em branco para organizar o código
 3. Escolha nomes representativos para variáveis
 4. Escreva um comando por linha
 5. Use parênteses em expressões

Algoritmos com qualidade - Exemplo

```
...  
se akj_a>7 então  
  leia(vlr12);  
  am<-10;  
  enquanto vlr12<>-1 faça  
    akj<-vlr12;akj<-akj+1-1;  
    se akj>10 então  
      imprima (“erro nota:”,am, vlr12);  
    senão  
      kjg(akj);  
  fim se;  
  leia(vlr12)  
fim enquanto;  
imprima(“ok”);  
fim se;  
...
```

Algoritmos com qualidade - Exemplo

```
...
se akj_a > 7 então
leia(vlr12);
am <- -10;
enquanto vlr12 <> -1 faça
akj <- -vlr12; akj <- -akj + 1 - 1;
se akj > 10 então
imprima (“erro nota:”, am, vlr12);
senão
kjpg(akj);
fim se;
leia(vlr12)
fim enquanto;
imprima(“ok”);
fim se;
...
```

```
...
se notaAnterior > 7 então
    leia(novaNota);
    notaMaxima <- 10;

enquanto novaNota <> -1 faça
    se novaNota > notaMaxima então
        imprima (“Erro na nota:”, novaNota,
            “A nota não pode ser maior que”,
            notaMaxima);

    senão
        armazenaNota(novaNota);
    fim se;
    leia(novaNota);
fim enquanto;

imprima(“Programa concluído.”);
fim se;
...
```

Algoritmos com qualidade

2. Devem ser comentados/documentados (comentários devem ser relevantes)
 1. O propósito do algoritmo
 2. Como utilizá-lo
 3. Autor do algoritmo
 4. Data de criação e histórico das atualizações
 5. Qual a finalidade das variáveis mais importantes
 6. Explicar atribuições e estruturas condicionais/repetição/de dados que sejam relevantes e/ou complexas

Algoritmos com qualidade - Exemplo

Problema: Crie um algoritmo para ordenar quaisquer três números inteiros informados pelo usuário

Nome: ordenar3nros

/*

Propósito: ordenação de três números inteiros

Utilização: o usuário deve informar três números inteiros para que o programa os retorne ordenadamente

Autor: Leonelo

Data de criação: 02/01/20213

Variáveis: numA, numB, numC - armazenam os três número inteiros informados pelo usuário

*/

Inicio

...

Expressões

- Envolvem:
 - Variáveis
 - Operadores
 - Constantes
- Exemplo:

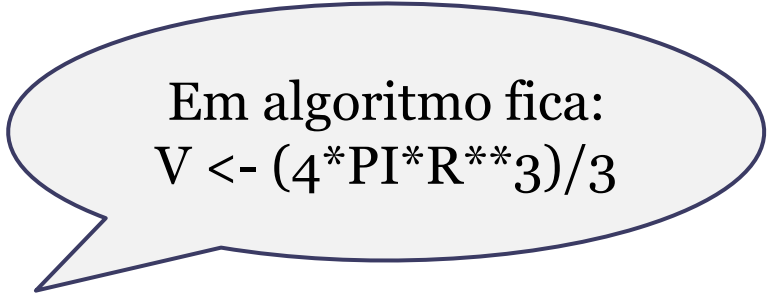
$$V = \frac{4\pi R^3}{3}$$

Expressões

- Envolvem:
 - Variáveis
 - Operadores
 - Constantes

- Exemplo:

$$V = \frac{4\pi R^3}{3}$$



Em algoritmo fica:
`V <- (4*PI*R**3)/3`

Expressões

- Envolvem:

- Variáveis

V, PI e R

- Operadores

*, ** e /

- Constantes

4 e 3

- Exemplo:

$$V = \frac{4\pi R^3}{3}$$

O operador <- é de atribuição. A variável à esquerda do <- recebe o resultado da expressão à direita.

Em algoritmo fica:
 $V <- (4 * PI * R ** 3) / 3$

Variáveis

- São nomes que referenciam locais da memória do computador que podem armazenar um valor de um tipo de dado
- Exemplo:
 - `inteiro valor <- 10;`
 - Onde:
 - **Inteiro**: é o tipo de dado que pode ser armazenado na variável
 - **Valor**: é o nome (ou identificador) da variável
 - **10**: é a constante inteira que está armazenada na variável

Arquitetura de um computador



RAM

- **Primária**
- Volátil
- Requer alimentação de energia contínua

- **Secundária**
- Persistente
- Não requer energia para manter os dados



HARD DISK

Pen-drives
CDs, DVDs



ENTRADA



SAÍDA



Arquitetura de um computador



Declaração de variáveis

- Toda variável deve ser declarada
- Tipos básicos de variáveis:
 - **Inteiro:** 0, 5064, 98
 - **Real:** 1.45, 0.99998
 - **Caractere:** “casa”, “bits”
 - **Lógico:** assume VERDADEIRO ou FALSO
- Exemplos:
 - Inteiro: idade, numFilhos;
 - Real: peso;
 - Caractere: nome, sobrenome;
 - Lógico: temPassaporte, ehMotorista;

Atividades

- Supondo que as variáveis NOTALUNO, NOMEALUNO, NUMMAT, GEN sejam utilizadas para armazenar a nota, nome, o número de matrícula e o gênero de um aluno, declare-as corretamente e atribua valores a elas.
- Encontre as falhas das declarações de variáveis e as corrija:
 - inteiro: ENDEREÇO, CIMENTO, NFIHOS;
 - caractere: IDADE, VOLUME, X ;
 - real: PREÇO, PESO, TOTALFALTAS;
 - lógico: LAMPADA;

Comandos e Operadores

- Comando de atribuição: <-
- Formato:
 - nomeVariavel <- expressão;
- Exemplo:
 - Idade <- 15;
 - Altura <- 1.85;

Comandos e Operadores

- Funções
 - Várias categorias: matemáticas, impressão, leitura, etc
- Exemplos:
 - Raiz(x) : raiz quadrada de x
 - Exp(x, y) : exponencial de x elevado a y
 - Abs(x) : Módulo (valor absoluto) da variável x
 - Int(x) : Parte inteira da variável x

Operadores aritméticos

- `**` ou `^` : exponencial
- `*` : multiplicação
- `/` : divisão
- `+` : soma
- `-` : subtração
- `Res` ou `%` : resto da divisão

Operadores relacionais

- $=$: igual
- $<>$ ou $\neq$: diferente
- $<$: menor
- $\leq$ ou $\leq$: igual ou igual
- $>$: maior
- $\geq$ ou $\geq$: maior ou igual

Operadores lógicos

- e : conjunção
- ou : disjunção
- não : negação
- x-ou : disjunção exclusiva

Operadores lógicos

- Tabela verdade: conjunto das combinações entre os valores de duas ou mais variáveis lógicas

A	B	A <u>e</u> B
F	F	F
F	V	F
V	F	F
V	V	V

A	B	A <u>ou</u> B
F	F	F
F	V	V
V	F	V
V	V	V

Operadores lógicos

- Tabela verdade: conjunto das combinações entre os valores de duas ou mais variáveis lógicas

A	B	$A \text{ e } B$
F	F	F
F	V	F
V	F	F
V	V	V

A	B	$A \text{ x-ou } B$
F	F	F
F	V	V
V	F	V
V	V	F

A	B	$A \text{ ou } B$
F	F	F
F	V	V
V	F	V
V	V	V

A	$\text{não } A$
F	V
V	F

Exemplos

$$(2 < 5) \text{ e } ((15/3) = 5)$$

$$(\text{n\u00e3o VERDADEIRO}) \text{ ou } ((3^{**}2)/3) < (15-(35 \text{ res } 7))$$

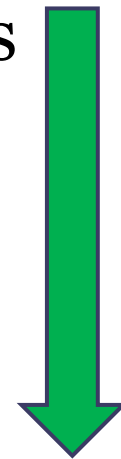
Exemplos

$$\begin{array}{c} (2 < 5) \text{ e } ((15/3) = 5) \\ (2 < 5) \text{ e } (5 = 5) \\ V \text{ e } V \\ V \end{array}$$

$$\begin{array}{c} (\text{n\~ao VERDADEIRO}) \text{ ou } ((3^{**}2)/3) < (15-(35 \text{ res } 7)) \\ F \text{ ou } (9/3) < (15 - 0) \\ F \text{ ou } 3 < 15 \\ F \text{ ou } V \\ V \end{array}$$

Ordem de prioridade

- Para resolver expressões é necessário seguir a ordem de prioridades entre funções e operadores
- Parênteses mais internos
- Funções matemáticas
- Operadores aritméticos
- Operadores relacionais
- Operadores lógicos

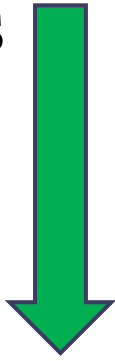


Maior prioridade

Ordem de prioridade

- Operadores aritméticos

- $**$ ou $^$
- $*$ ou $/$ ou Res ou $\%$
- $+$ ou $-$



Maior prioridade

- Operadores lógicos

- não
- e ou ou ou x-ou



Maior prioridade

Atividades

inteiro: A, B, C ;
real: D;
caractere: TIPO; TEMP;
lógico: L

- Suponha as declarações e atribuições de valores em um algoritmo qualquer.
- Quais os resultados das expressões abaixo ?

A ← 5;
B ← 10;
C ← -8;
D ← 1.5;
TIPO ← “noite”;
TEMP ← “fria”;
L ← falso;

- a) $2 * (A \text{ res } 3) - C;$
- b) $\text{int}(\text{raiz}(2 * \text{abs}(C)) / 4);$
- c) $(\text{int}(A/B) + \text{abs}(C)) ** 3;$
- d) $3 + (\text{raiz}((C+16) * ((3 \text{ res } (D+0.5)) * 2)))$;
- e) $\text{int}((A+B)/A) * \text{int}(\text{abs}(C)/D) - \text{int}(D*3);$
- f) $B = A * C \text{ e } L \text{ ou } \text{verdadeiro};$
- g) $\text{“dia”} = \text{TIPO} \text{ ou } \text{“fria”} \neq \text{TEMP};$
- h) $L \text{ e } \text{int}(B/A) \geq C \text{ ou } \text{não } A = C;$
- i) $\text{raiz}(7 ** 2) = 14 / (B/A) \text{ e } (B - 3 \leq C + 0.5);$
- j) $\text{não } L \text{ ou } \text{verdadeiro} \text{ e } \text{abs}(C) \geq \text{int}(A/A);$
- k) $\text{abs}(B) = \text{int}(10 ** 2 / 10);$

Comandos de entrada e saída

- Comando de entrada:
 - `leia(nomeVariaveis);`
- Comando de saída:
 - `imprima(texto, nomeVariaveis);`

Exemplo

Nome: somaNros

início

```
/* lê duas variáveis e imprime a soma delas */
```

```
real: A, B, SOMA;
```

```
imprima("Entre com os valores de A e B");
```

```
leia(A, B);
```

```
imprima("O Valor da primeira variável é", A);
```

```
imprima("O Valor da segunda variável é", B);
```

```
SOMA <- A + B;
```

```
imprima("O Valor da soma A+B é", SOMA);
```

fim

Atividades

- Quais serão as saídas do algoritmo a seguir, supondo a primeira entrada igual a 10 e a segunda igual a 4 ?

início

inteiro : X, Y;

real : Z;

leia (X);

imprima(X, “ao cubo = “, X**3);

leia(Y);

imprima(X+Y);

Z←X/Y;

imprima (Z);

Z←int(Z);

imprima(Z);

Z←Z+1;

X←((Y+Z) res 2);

imprima(X);

fim

Atividades

- Elabore um algoritmo para o cálculo e impressão da média de quatro notas de um aluno (as notas são fornecidas pelo usuário) .
- Elabore um algoritmo para o cálculo e impressão do volume de uma esfera de raio R , onde R é fornecido pelo usuário.
- Elabore um algoritmo para o cálculo do total de latas de tinta necessárias para se pintar um muro, considerando-se as seguintes situações:
 - O muro tem 15 m de largura e 3 m de altura e 1 lata de tinta é suficiente para se pintar 4 m² do muro.
- Agora assuma que o tamanho do muro (largura e altura) é fornecido pelo usuário. Nesta segunda situação 1 lata de tinta é suficiente para se pintar R m² do muro, onde o valor de R (rendimento) também é fornecido pelo usuário
 - OBS. Considere que total de latas pode ser um valor quebrado (isto é um valor real ou de ponto flutuante)