

Algoritmos: definição e representação

Bruno A. N. Travençolo – FACOM-UFU

Algoritmos

- ▶ Objetivo do computador é realizar tarefas que envolvam processamento de informações.
- ▶ Livra os seres humanos de esforços repetitivos, tediosos e sujeito a erros.
- ▶ Obtenção de resultados confiáveis em tempos hábil, mesmo sendo uma imensa quantidade de dados.
- ▶ Computador não tem senso próprio
 - ▶ Deve receber instruções explícitas (algoritmos)



Algoritmos

- Algoritmo é uma seqüência finita e bem definida de passos que, quando executados, realizam uma tarefa específica ou resolvem um problema.
- ▶ **Ex:** Receitas de culinária, manual de instruções, coreografia, etc.
- **Propriedades do algoritmo:**
 - Composto por ações simples e bem definidas (não pode haver ambigüidade, ou seja, cada instrução representa uma ação que deve ser entendida e realizada).
 - Seqüência ordenada de ações
 - Conjunto finito de passos
- ▶ **Pergunta:** Como saber se já temos detalhes suficientes para o algoritmo ser entendido e realizado?



R: Depende da relação de instruções reconhecidas pelo **AGENTE EXECUTOR** do algoritmo.

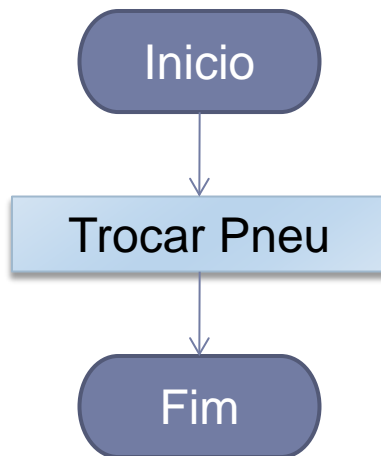
Ex: receita de bolo \Rightarrow **Ser Humano**

algoritmo computacional \Rightarrow **Computador**



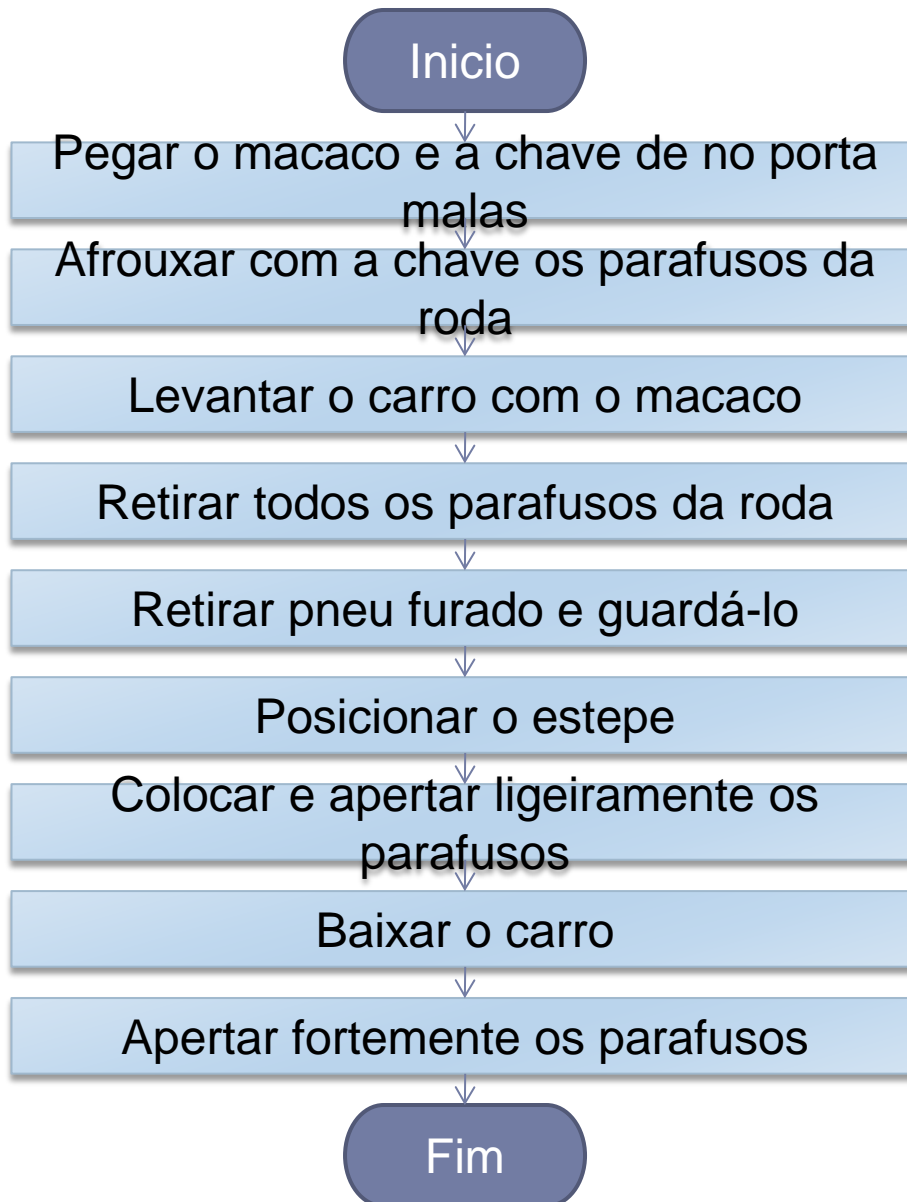
Exemplo:

- ▶ Algoritmo para trocar o pneu de um carro



O nível de detalhe da instrução “Trocar Pneu” é suficiente para vc?

Exemplo



Estrutura
sequencial

Algoritmo

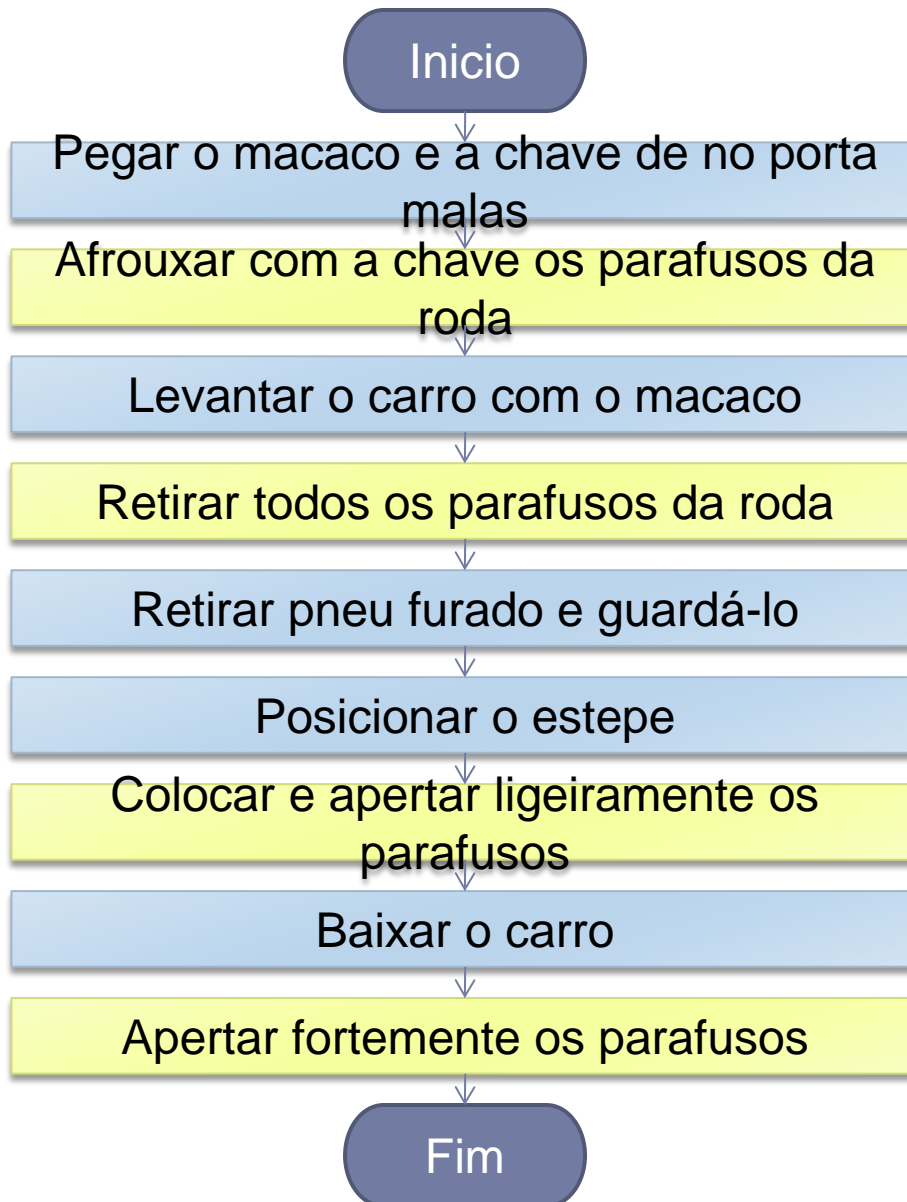
Início

Pegar o macaco e a chave de no porta malas
Afrouxar com a chave os parafusos da roda
Levantar o carro com o macaco
Retirar todos os parafusos da roda
Retirar pneu furado e guardá-lo
Posicionar o estepe
Colocar e apertar ligeiramente os parafusos
Baixar o carro
Apertar fortemente os parafusos

Fim

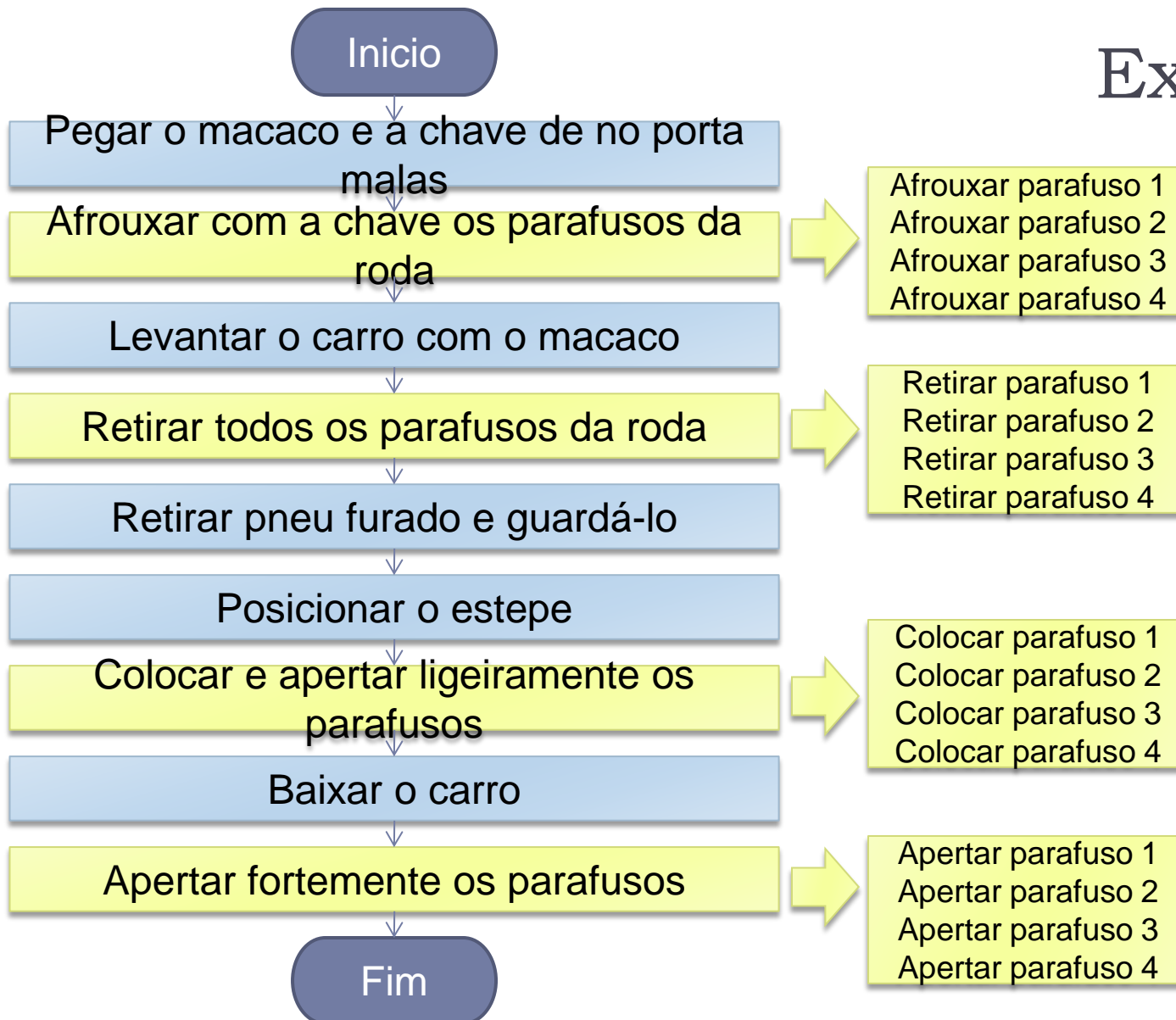


Exemplo

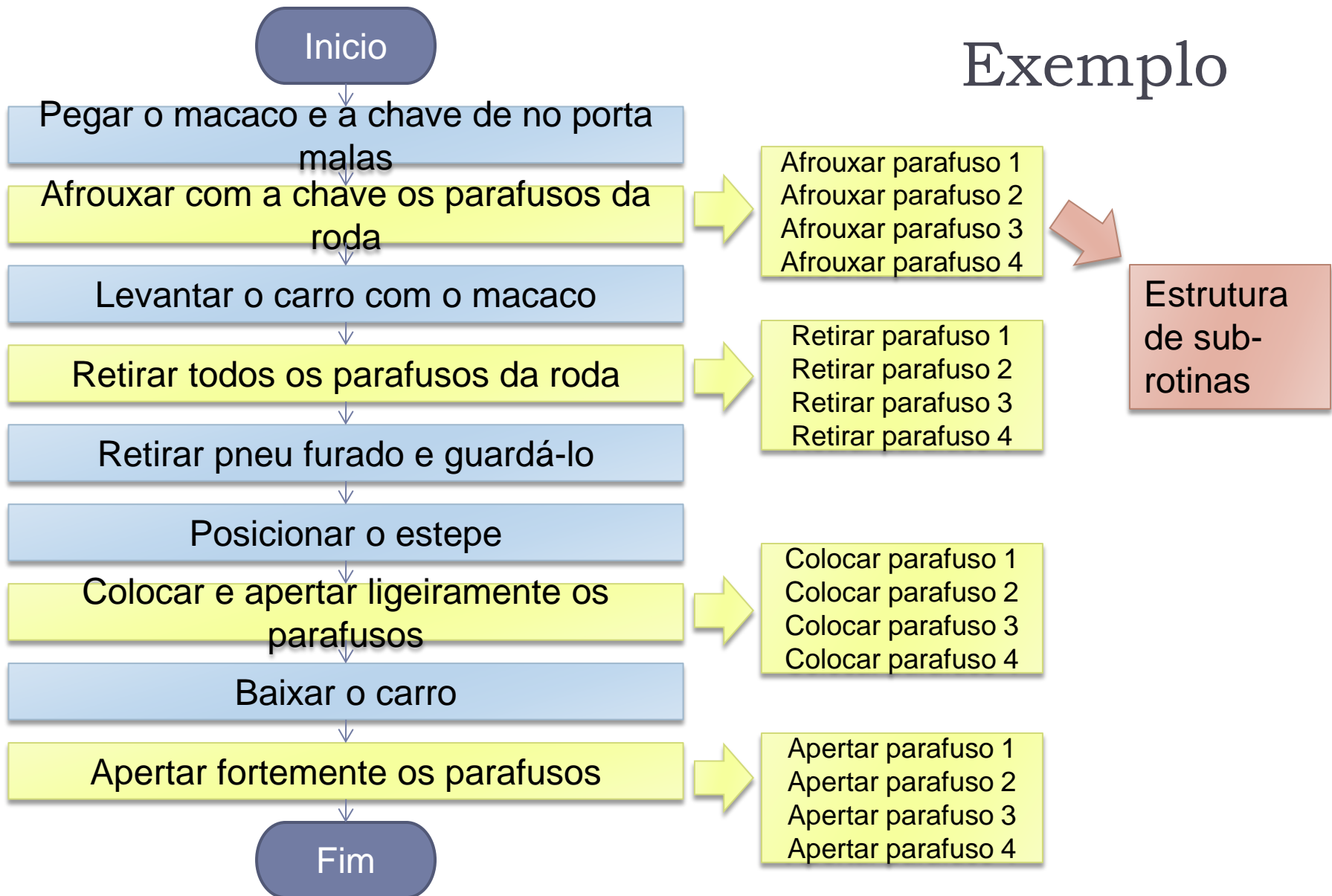


► Podemos detalhar algumas etapas, ou seja, um passo pode ser refinado em passos menores

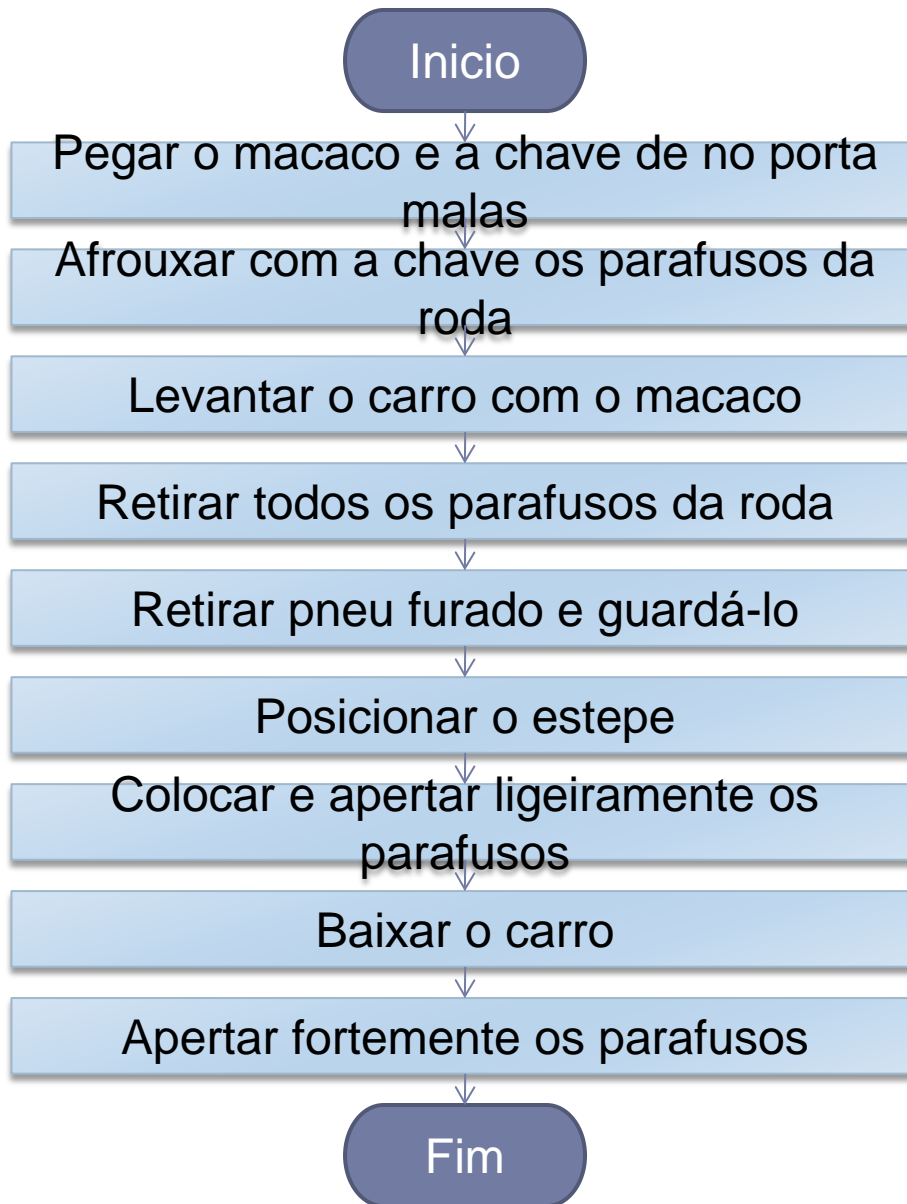
Exemplo



Exemplo



Exemplo

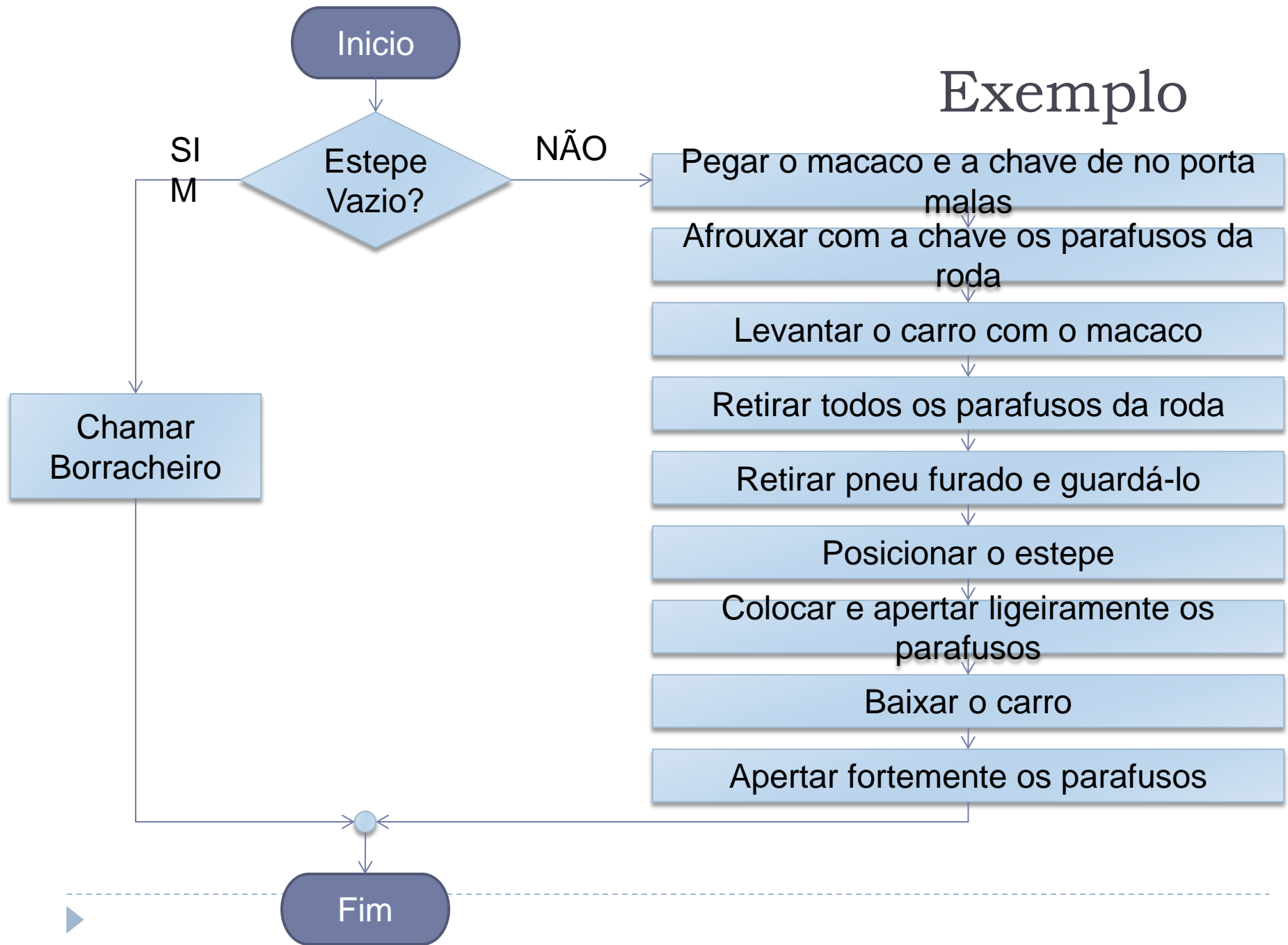


Problemas

- ▶ E se não for possível realizar todos os passos?
- ▶ E se algum imprevisto ocorrer?
- ▶ E se existir mais de uma alternativa



Exemplo



Algoritmo

Início

SE (estepe está vazio)

Chamar borracheiro

SENÃO

Pegar o macaco e a chave de no porta malas

Afrouxar com a chave os parafusos da roda

Levantar o carro com o macaco

Retirar todos os parafusos da roda

Retirar pneu furado e guardá-lo

Posicionar o estepe

Colocar e apertar ligeiramente os parafusos

Baixar o carro

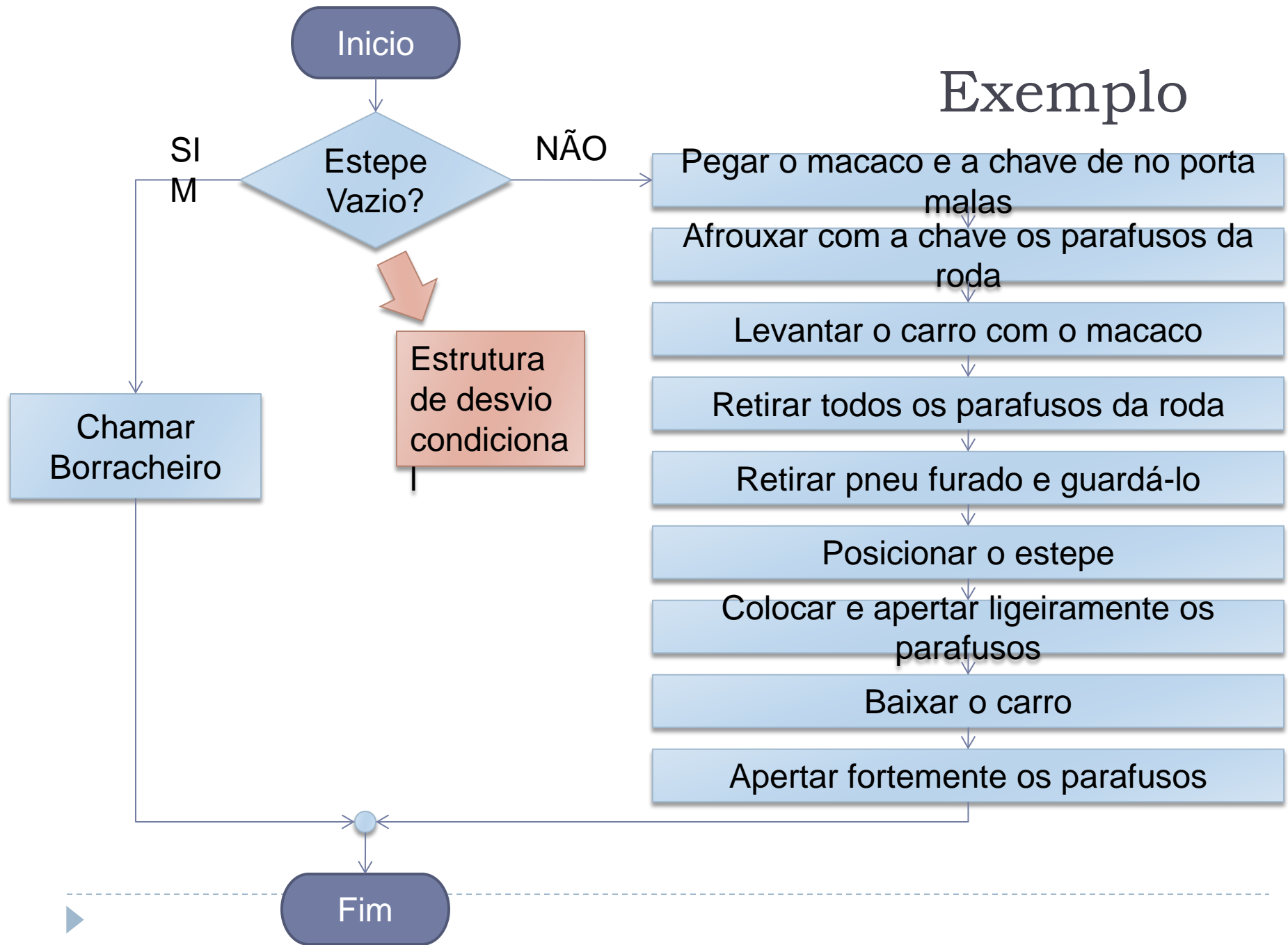
Apertar fortemente os parafusos

FIM SE

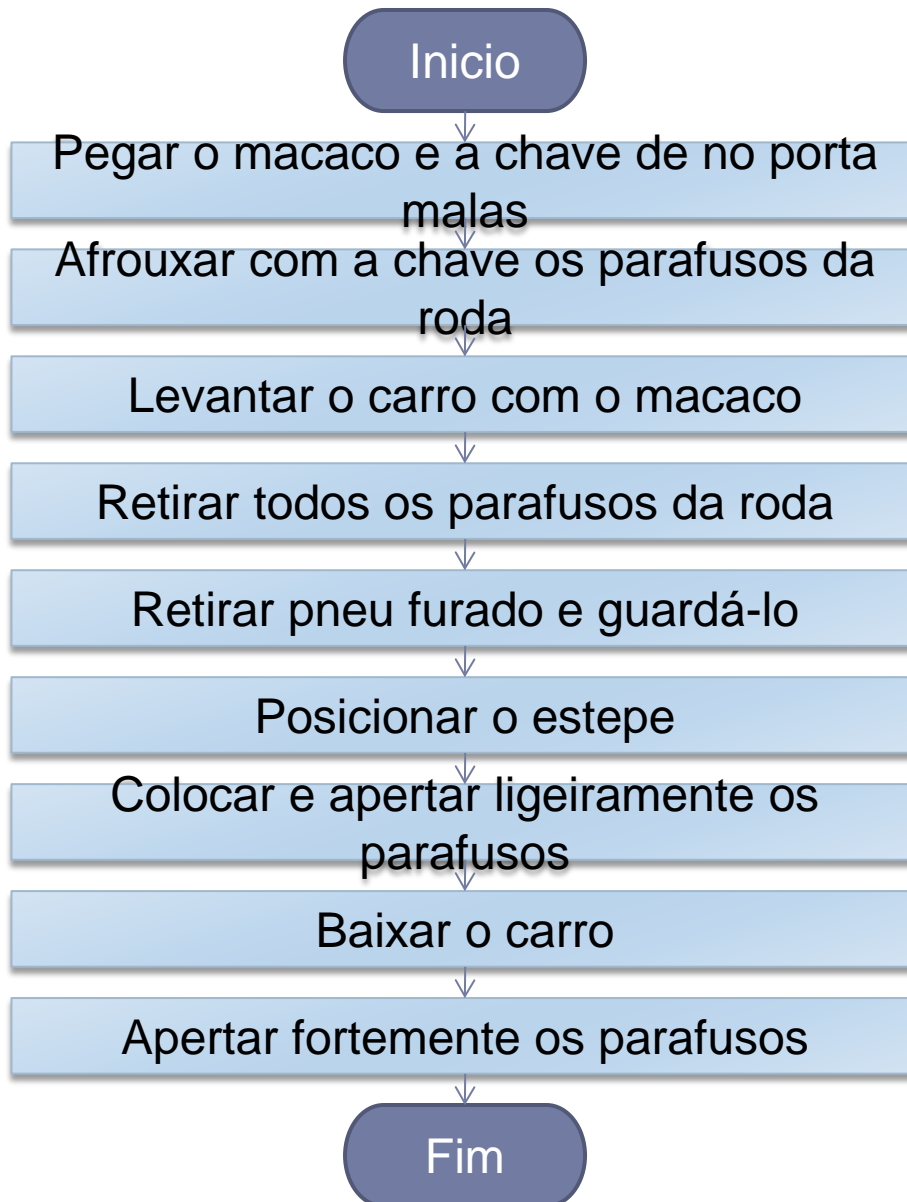
Fim



Exemplo



Exemplo



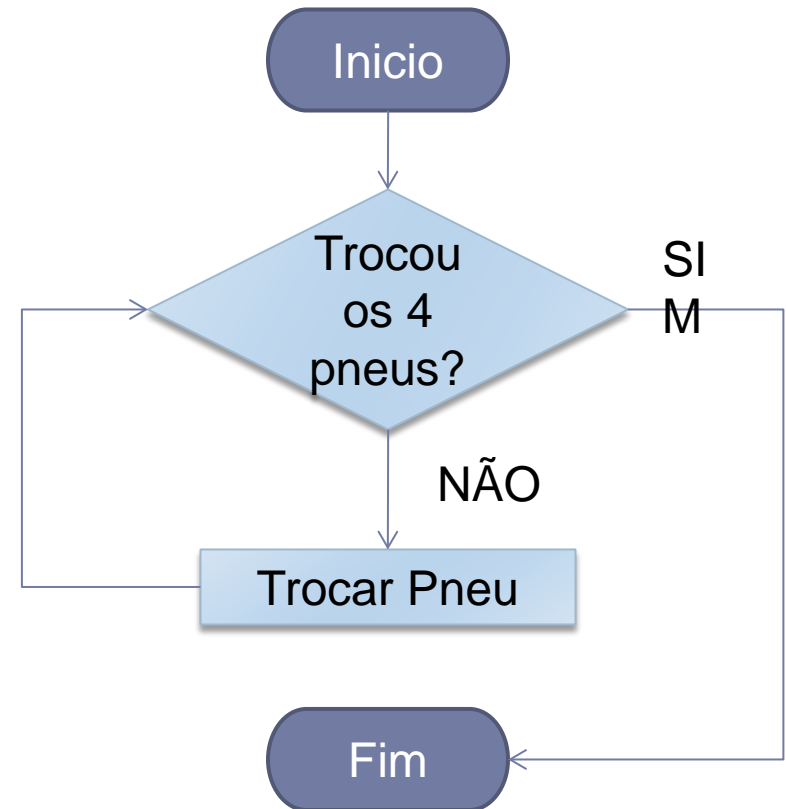
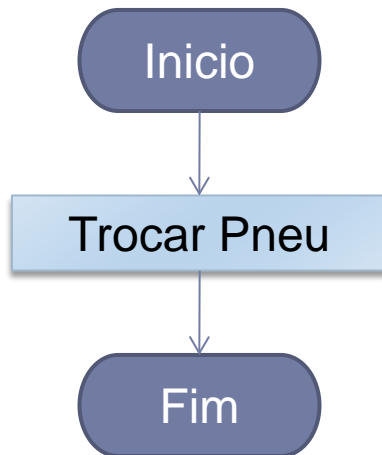
Outras possibilidades

- Em um pit stop devemos trocar os quatro pneus... Como fazer isso?

Exemplo

Outras possibilidades

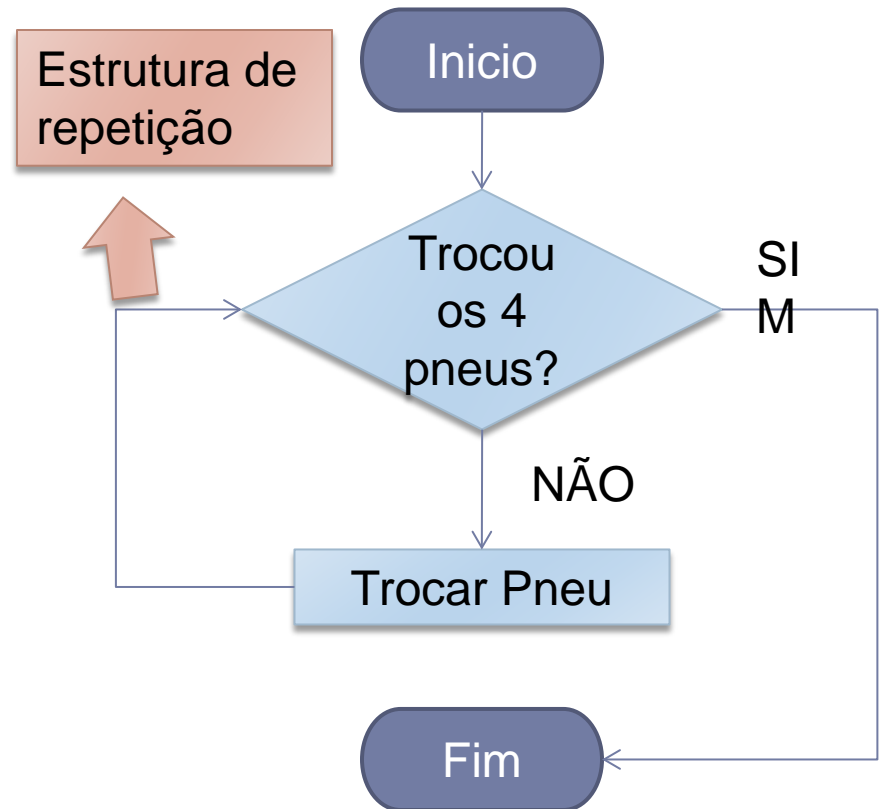
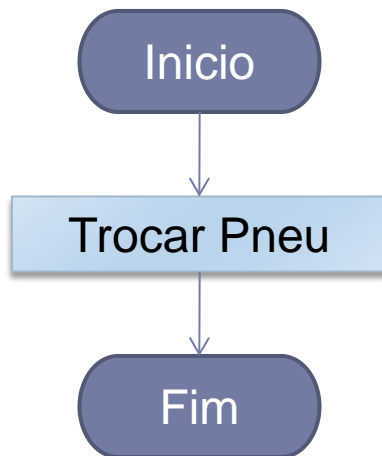
- Em um pit stop devemos trocar os quatro pneus... Como fazer isso?



Exemplo

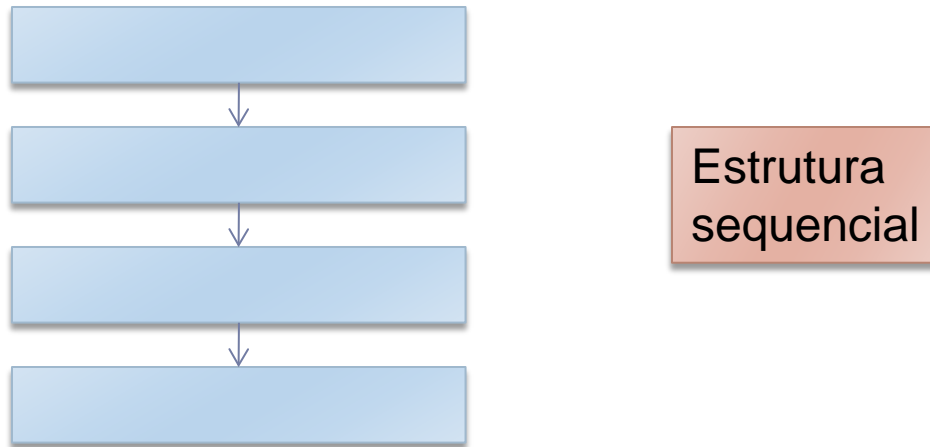
Outras possibilidades

- Em um pit stop devemos trocar os quatro pneus... Como fazer isso?



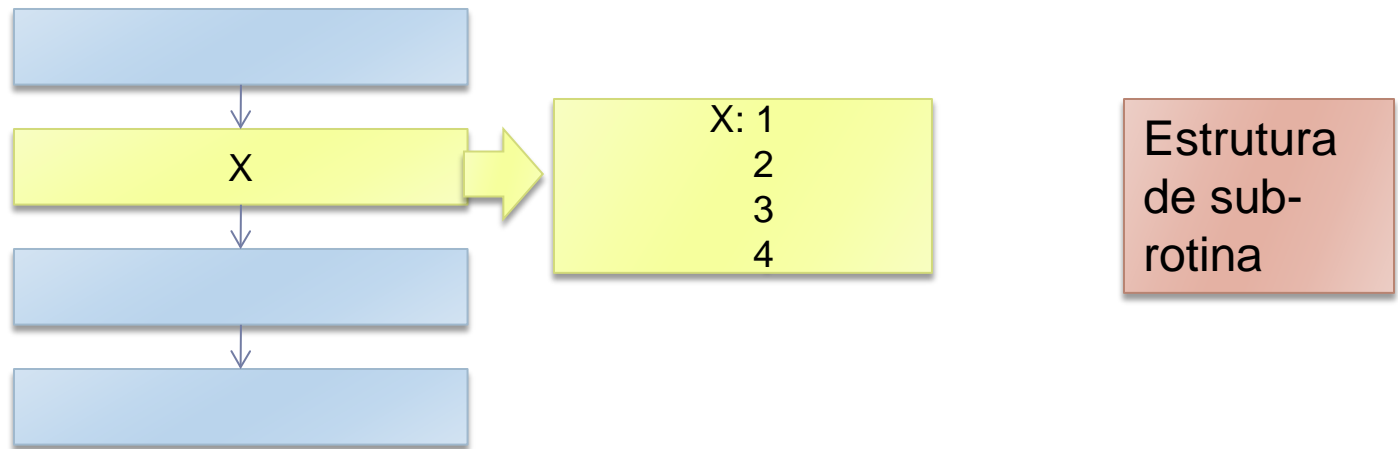
Estrutura dos Algoritmos

- ▶ Em uma estrutura sequencial, os passos são tomados em uma sequência pré-definida



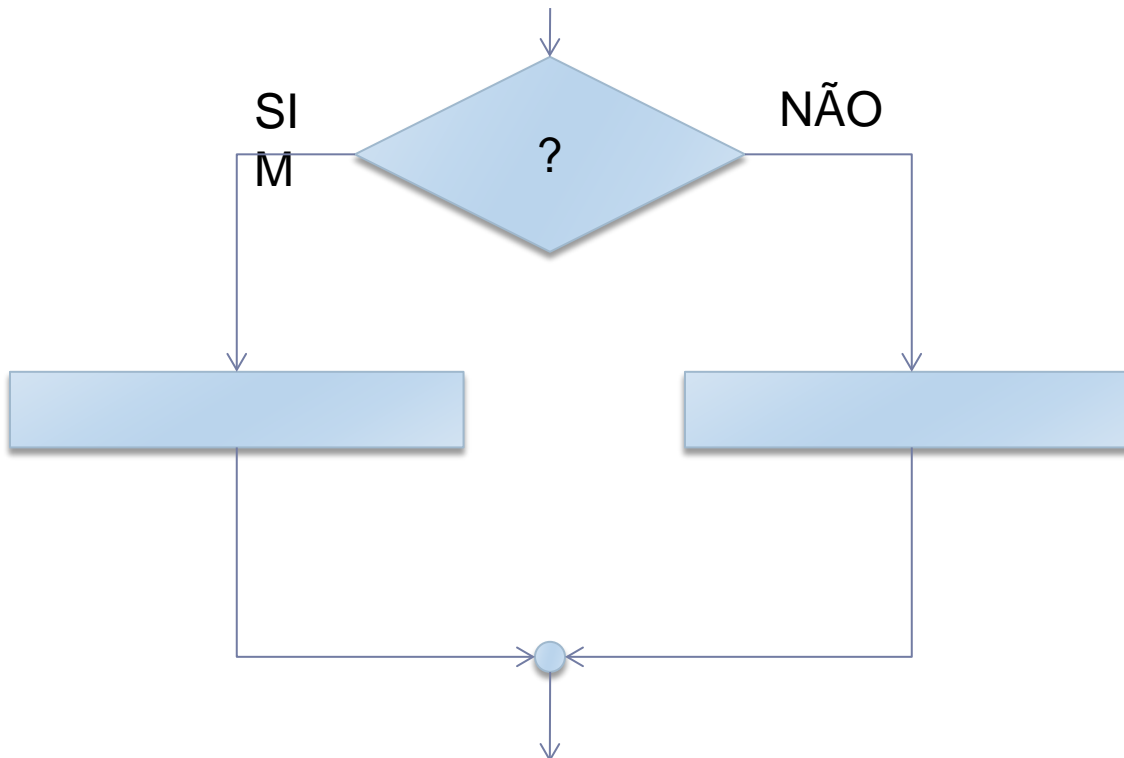
Estrutura dos Algoritmos

- ▶ Em uma estrutura de sub-rotina, a execução é desviada para uma seqüência de comandos que executam uma tarefa, voltando ao fluxo normal.



Estrutura dos Algoritmos

- ▶ Uma estrutura condicional permite a escolha do grupo de ações a ser executado quando determinada condição é ou não satisfeita

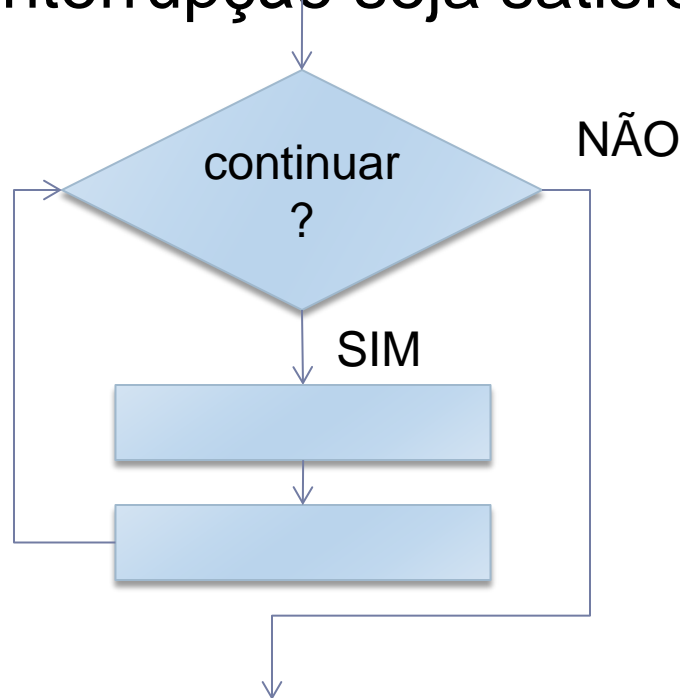


Estrutura
condiciona



Estrutura dos Algoritmos

- ▶ Uma estrutura de repetição permite que uma seqüência de comandos seja executada repetidamente até que uma determinada condição de interrupção seja satisfeita.



Estrutura de
repetição

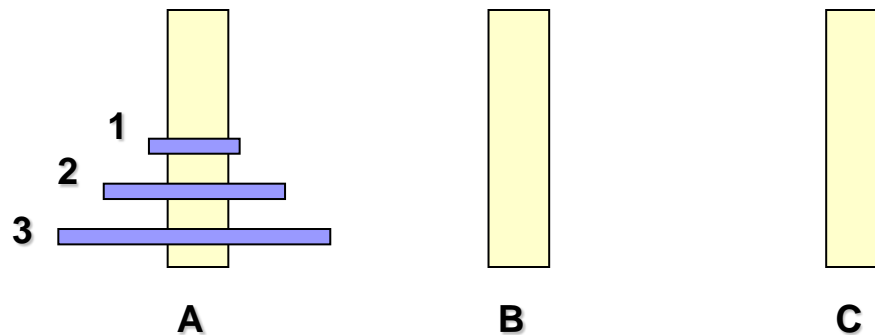
Exercícios

- ▶ Escreva um algoritmo para a troca de uma lâmpada queimada
- ▶ Construa um algoritmo para solução do problema da Torres de Hanói



Construindo um Algoritmo (Problema das Torres de Hanói):

Regra: Mover os discos da haste A para a haste C sem que o disco maior fique sobre o disco menor.



Solução:

Início

Move o disco 1 para a haste C

Move o disco 2 para a haste B

Move o disco 1 para a haste B

Move o disco 3 para a haste C

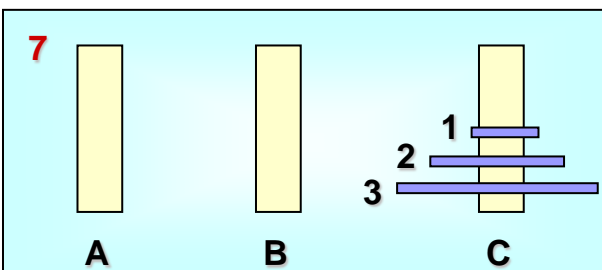
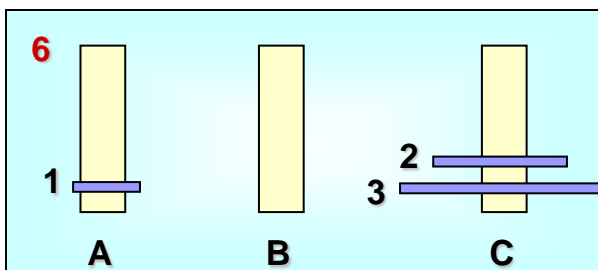
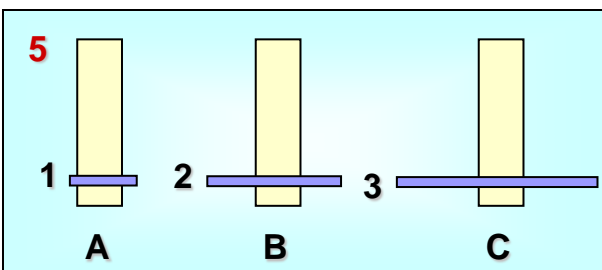
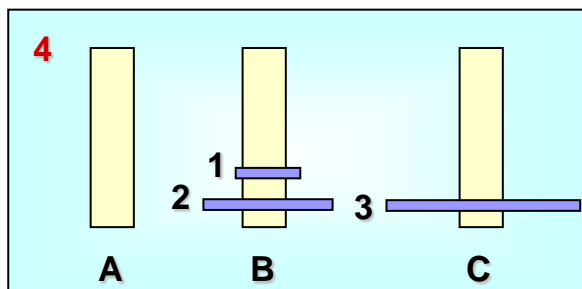
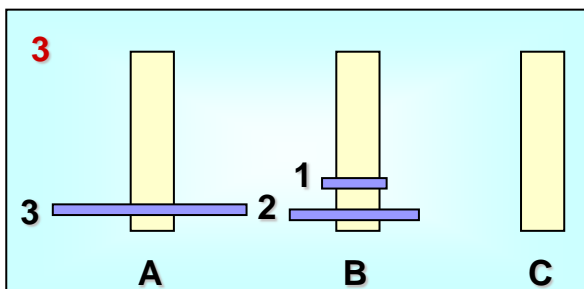
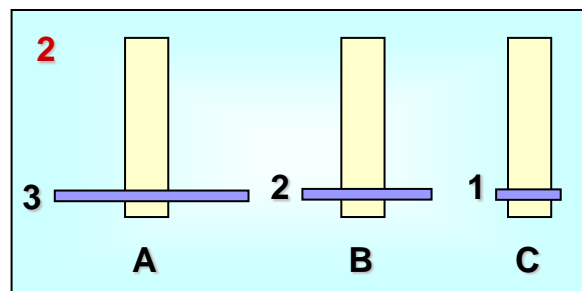
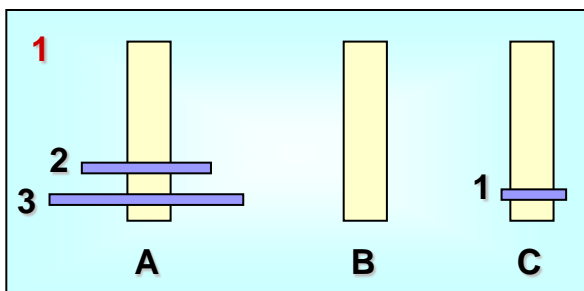
Move o disco 1 para a haste A

Move o disco 2 para a haste C

Move o disco 1 para a haste C

Fim





Exemplo: alg. em
C

Exercícios de Lógica:

1. Temos 3 recipientes de tamanhos distintos (8, 5 e 3 litros), sendo que o recipiente de 8 litros está totalmente cheio. Considerando que os recipientes não sejam graduados, deseja-se colocar 4 litros em dois recipientes.
2. Um comerciante está transportando um lobo, um coelho e 500 kg de cenouras. Durante a viagem, ele se depara com um rio e um pequeno barco, no qual só é possível transportar um elemento por vez. Descreva quais serão as ações tomadas pelo comerciante para atravessar o rio, de modo que ele nunca deixe o lobo e o coelho ou o coelho e as cenouras sozinhos em uma das margens.



- Solução problema 1

8 0 0

3 5 0

3 2 3

6 2 0

6 0 2

1 5 2

1 4 3

4 4 0

- Solução problema 2

Leva o coelho

Deixa coelho

Volta vazio

Leva lobo

Deixa lobo

Volta com coelho

Deixa coelho

Leva cenoura

Deixa cenoura

Volta vazio

Leva coelho



Mais exercícios

- ▶ Exercícios de lógica em:
- ▶ <http://www.plastelina.net/>



Formas de Representação de Algoritmos

A descrição de um algoritmo de **forma clara e fácil de ser seguida** ajuda no seu desenvolvimento, depuração (localização e correção de erros) e futura migração para uma **linguagem de programação**.

Para facilitar este trabalho, são utilizadas ferramentas específicas de representação da lógica de programação (*seqüência de ações a serem realizadas*).



Linguagens de Programação

- ▶ Linguagem de programação é um conjunto de termos (vocabulário) e de regras (sintaxe) que permitem a formulação de instruções a um computador
- ▶ Linguagem de máquina: nível mais baixo, elementar. Utiliza-se apenas 0 e 1

0001000010001110001011000111001

101010010100101001011111

01010101001010101



Linguagens de Programação

- ▶ Linguagem simbólica (assembly): utiliza códigos mnemônicos (abreviações para as instruções ao invés de números). Esses códigos são traduzidos para instruções de máquina executáveis pelo computador

- ▶ Exemplo

Instruction (AT&T syntax)

.begin

.org 2048

.equ 3000

ld length,%

be done

addcc %r1,-4,%r1

addcc %r1,%r2,%r4

ld %r4,%r5

ba loop

addcc %r3,%r5,%r3

jmp1 %r15+4,%r0

20

a_start

.org a_start

Linguagem Assembly p/ ling. de Máquina

Endereço	Rótulo	Instruction (AT&T syntax)	Código objeto (linguagem de máquina)
		.begin	
		.org 2048	
	a_start	.equ 3000	
2048		ld length, %	
2064		be done	00000010 10000000 00000000 00000110
2068		addcc %r1, -4, %r1	10000010 10000000 01111111 11111100
2072		addcc %r1, %r2, %r4	10001000 10000000 01000000 00000010
2076		ld %r4, %r5	11001010 00000001 00000000 00000000
2080		ba loop	00010000 10111111 11111111 11111011
2084		addcc %r3, %r5, %r3	10000110 10000000 11000000 00000101
2088	done:	jmp l %r15+4, %r0	10000001 11000011 11100000 00000100
2092	length:	20	00000000 00000000 00000000 00010100
2096	address:	a_start	00000000 00000000 00001011 10111000
		.org a_start	
3000	a:		

Linguagens de Programação

- ▶ Linguagens de alto nível: Uma linguagem de programação procedural, orientada a problemas, mais próxima a linguagem humana e longe do código de máquina.

- ▶ Ex: C; C++; Java

- ▶ Programa em C

```
#include<stdio.h>
void torre(int n, char source, char dest, char aux);

main()
{
    int n;
    char source = 'A';
    char dest = 'C';
    char aux = 'B';
    printf("entre com o numero de discos: ");
    scanf("%d",&n);
    torre(n, source, dest, aux);
    getchar();
    getchar();
}

void torre(int n, char source, char dest, char aux)
{
    static int step = 0;

    printf("torre(%d, %c, %c, %c)\n", n, source, dest, aux);
    if (n==1)
        printf("\t\t\tstep %3d: mova de %c para %c\n", ++step, source, dest);
    else
```

C para Assembly

```
#include<stdio.h>
void torre(int n, char source, char dest,char aux);

main()
{
    int n;
    char source = 'A';
    char dest = 'C';
    char aux = 'B';
    printf("entre com o numero de discos: ");
    scanf("%d",&n);
    torre(n, source,dest,aux);
    getchar();
    getchar();
}

void torre(int n, char source, char dest,char aux)
{
    static int step = 0;

    printf("torre(%d, %c, %c, %c)\n",n, source, dest,aux);
    if (n==1)
        printf("\t\t\tstep %3d: mova de %c para %c\n",++step, source, dest);
    else
        torre(n-1, dest, source, aux);
    torre(n-1, source, dest, aux);
    step++;
}
```

```
00401190 50          push eax
00401191 68C7204000 push $004020c7
00401196 E8AB020000 call $00401446
0040119B 83C408      add esp,$08
Unit1.c.12: torre(n, source,dest,aux);
0040119E 8A55F9      mov dl,[ebp-$07]
004011A1 52          push edx
004011A2 8A4DFA      mov cl,[ebp-$06]
004011A5 51          push ecx
004011A6 8A45FB      mov al,[ebp-$05]
004011A9 50          push eax
004011AA FF75FC      push dword ptr [ebp-$04]
004011AD E848000000 call torre(int,signed char,signed char,signed char)
004011B2 83C410      add esp,$10
Unit1.c.13: getchar();
004011B5 8B151C514000 mov edx,[$0040511c]
004011BB FF4A08      dec dword ptr [edx+$08]
004011BE 780A        js $004011ca
004011C0 8B0D1C514000 mov ecx,[$0040511c]
004011C6 FF01        inc dword ptr [ecx]
004011C8 EB0C        jmp $004011d6
004011CA FF351C514000 push dword ptr [$0040511c]
004011D0 E82F020000 call $00401404
004011D5 59          pop ecx
```

Compilador

- Conjunto de programas que transforma uma linguagem de alto nível em linguagem de máquina executável por computador

```
#include<stdio.h>
void torre(int n, char source, char dest, char aux);
```

```
main()
{
    int n;
    char source = 'A';
    char dest = 'C';
    char aux = 'B';
    printf("entre com o numero de discos: ");
    scanf("%d",&n);
    torre(n, source, dest, aux);
    getchar();
    getchar();
}
```

```
void torre(int n, char source, char dest, char aux)
{
    static int step = 0;

    printf("torre(%d, %c, %c, %c)\n", n, source, dest, aux);
    if (n==1)
        printf("\t\t\tstep %3d: mova de %c para %c\n", ++step, source, dest);
    else
```

→

```
00010000100011100010111001
101010010100101001011111
01010101001010101
```

(voltando...)

Formas de Representação de Algoritmos

A descrição de um algoritmo de **forma clara e fácil de ser seguida** ajuda no seu desenvolvimento, depuração (localização e correção de erros) e futura migração para uma **linguagem de programação**.

Para facilitar este trabalho, são utilizadas ferramentas específicas de representação da lógica de programação (*seqüência de ações a serem realizadas*).



•Descrição Narrativa

Especificação verbal dos passos em linguagem natural.

Desvantagens:

A linguagem natural é imprecisa (possibilita ambigüidades).

Proporciona maior trabalho na codificação.

Sugere-se sua utilização apenas para **comentar** algoritmos e/ou programas, esclarecendo ou realçando pontos específicos.



•Fluxograma

Uso de ilustrações gráficas para representar as instruções.

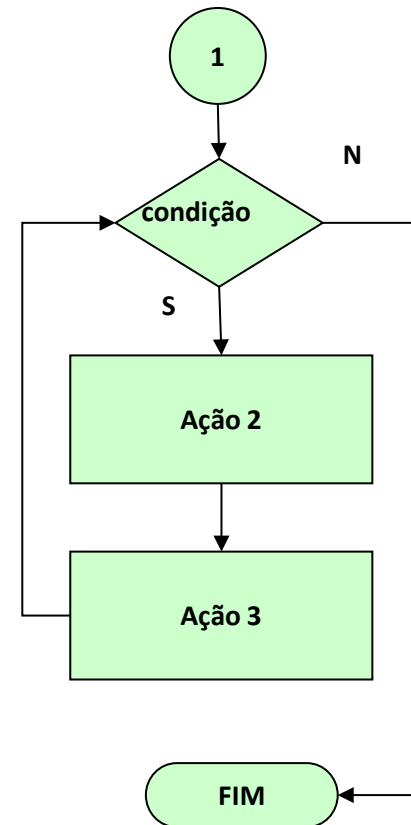
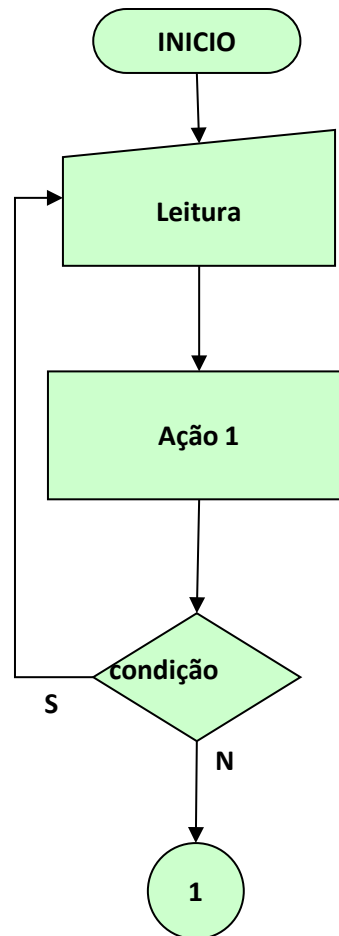
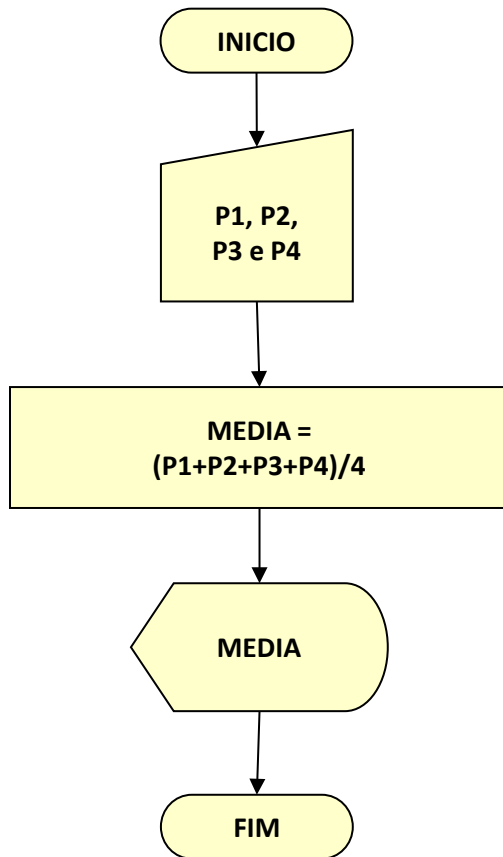
Apresenta a lógica de um algoritmo, enfatizando passos individuais (objetos gráficos) e o fluxo de execução (setas)

Desvantagens:

Fluxogramas detalhados podem obscurecer a estrutura do programa.

Permite transferências arbitrárias de controle





•Pseudolinguagem

Linguagem especial para desenvolvimento de algoritmos, que **utiliza expressões pré-definidas** para representar ações e fluxos de controle.

Funciona como uma linguagem simplificada de programação, logo, **facilita a codificação** futura.

É uma descrição textual, estruturada e regida por regras; que descrevem os passos executados no algoritmo.

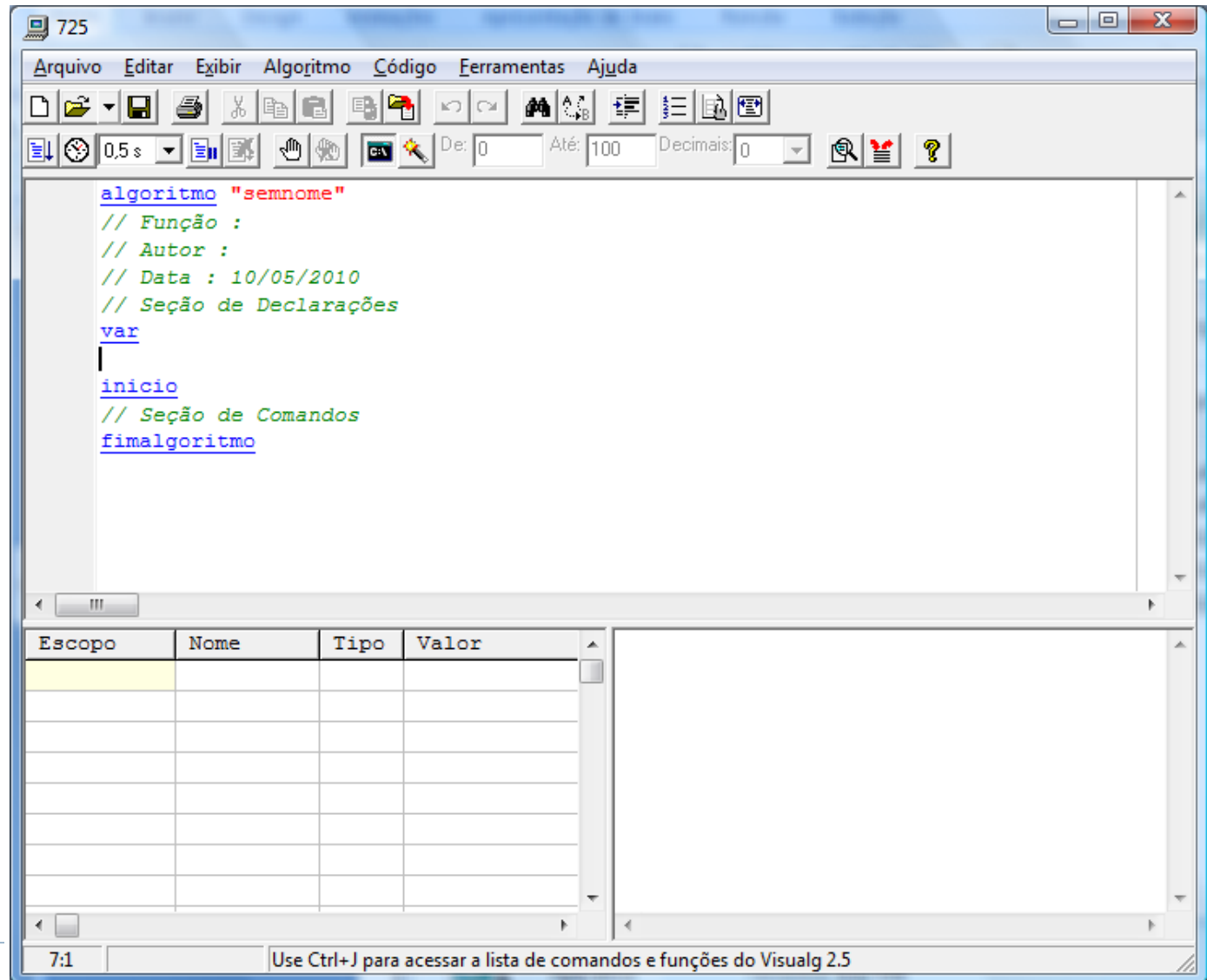


Possui características similares às linguagens de programação:

- Utiliza palavras-chaves (ex: escreva, se-então, etc.);
- Indentação (alinhamento dos blocos de comandos);
- Possui um comando por linha;



VisuAlg

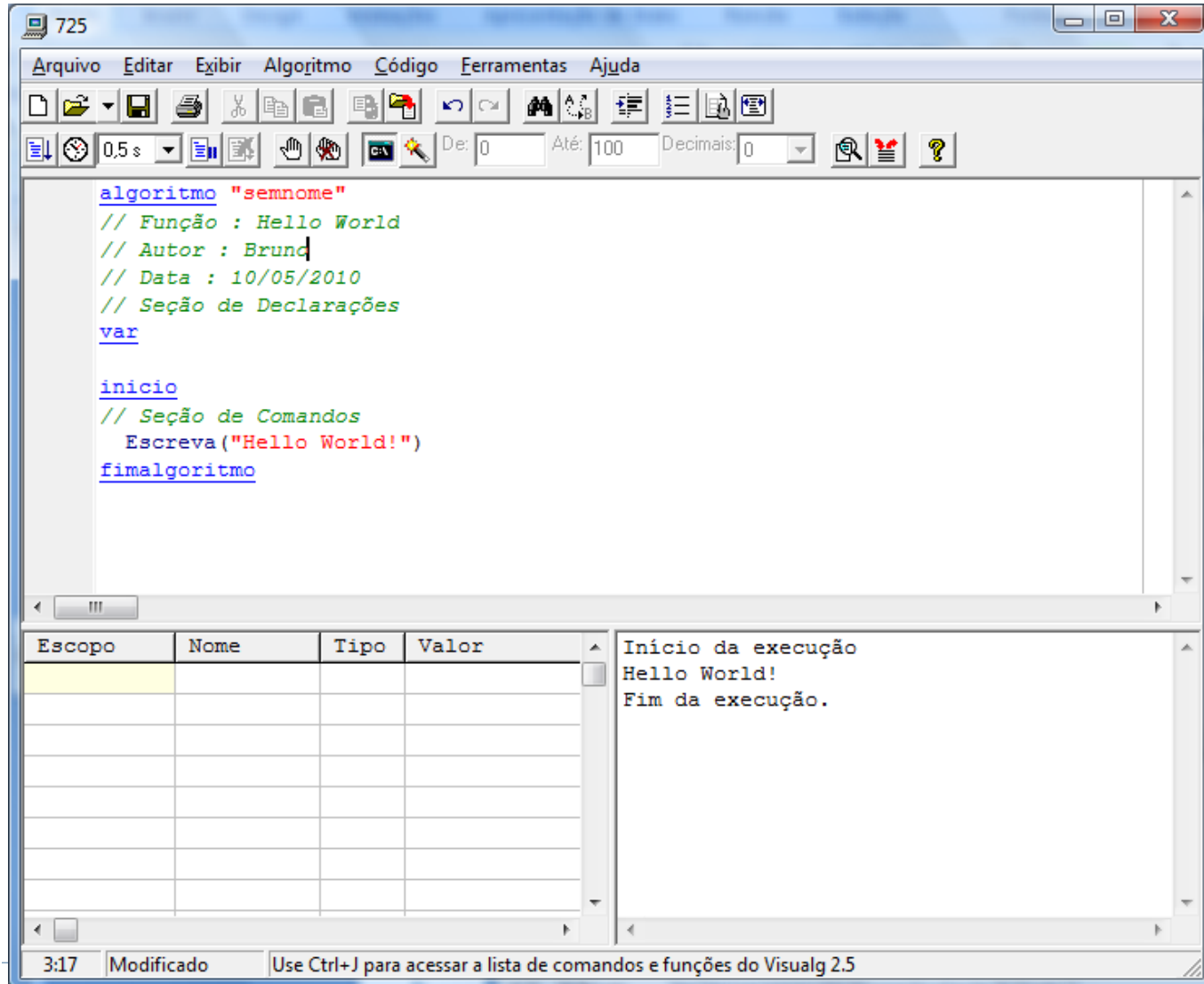


Formato básico do pseudo-código da linguagem do VisualAlg

```
algoritmo "semnome"  
  // Função :  
  // Autor :  
  // Data :  
  // Seção de Declarações  
inicio  
  // Seção de Comandos  
finalgoritmo
```



Hello World em VisuALG



Comandos de Entrada

- ▶ É o comando que faz com que o sistema receba os **dados de entrada** do programa e os armazena em **variáveis**
- ▶ Os dados de entrada são fornecidos ao sistema por meio de um dispositivo de entrada, como um teclado
- ▶ Na execução de um comando de Entrada, o processamento é interrompido até que sejam fornecidos, por uma unidade de entrada, valores para os dados



Comando de Saída

- ▶ É o comando pelo qual o sistema fornece, numa **unidade de saída**, resultados do processamento e mensagens
- ▶ Resultados do processamento
 - ▶ Conteúdo de **variáveis**
 - ▶ Valores de constantes
 - ▶ Resultados de operações aritméticas e lógicas
- ▶ Mensagens: são utilizadas para que o programa dê informações a respeito de sua execução
 - ▶ Conteúdo de variáveis
 - ▶ Constantes do tipo String (sequencia de caracteres)
 - ▶ Mensagens informativas



Alguns comandos da pseudo-linguagem do VisualAlg

- ▶ **Leia (x)** – le um valor do teclado e atribui à variável x
- ▶ **Escreva (“texto”, lista de variáveis)** – escreve o valor das variáveis que foram especificadas no comando
- ▶ **Escreval(“texto”, lista de variáveis)** - escreve o valor das variáveis que foram especificadas no comando e *pula uma linha*



Comandos de Entrada X Saída

- ▶ **Leia(x)** é um comando de **ENTRADA**
- ▶ **Escreva("")** é um comando de **SAÍDA**



Tipos de Dados

- ▶ O que a seguinte informação que está na memória RAM representa?

1100001

- ▶ O número 97?
 - ▶ A letra “a”? (vide tabela ASCII)
- ▶ É necessário definir durante a programação



Referências

- ▶ Slides adaptados da Web - WIKI ICMC
<http://wiki.icmc.usp.br/index.php/SSC-501> (Prof. Fernando Osório) e aulas da Prof. Denise Guliato –FACOM-UFU

