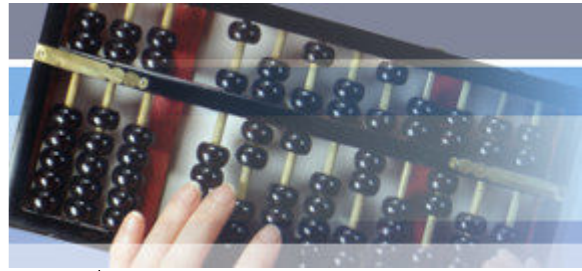


Curso de C

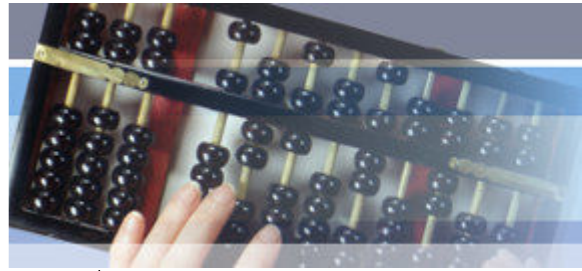
Introdução



Introdução

Objetivos:

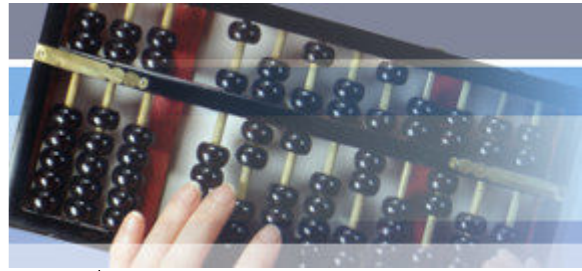
- Compreender:
 - Como funciona o processo de desenvolvimento de um programa
 - Como transformar um algoritmo em um programa executável
 - A diferença entre diversas linguagens de programação e níveis de abstração



Introdução

Roteiro:

- Recordando Algoritmos
- Linguagem de Programação
- O computador
- Instruções de Máquina
- Níveis de Abstração
- Compilação



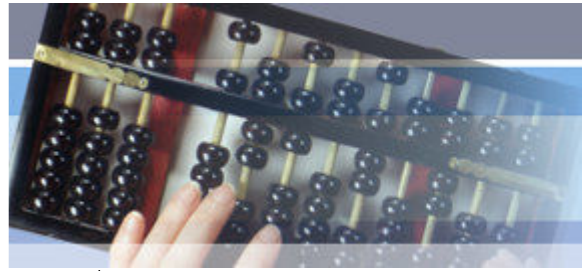
Algoritmos

Recordando:

Algoritmo: conjunto finito de instruções

- Usualmente, começa com a primeira instrução
- Execução seqüencial, uma instrução de cada vez, com possibilidade de saltos para outras instruções
- Instruções individuais suficientemente elementares, ou primitivas
- Sempre deve alcançar uma instrução PARE, para terminar a execução do algoritmo.

Utiliza dados (**entrada**) e gera um resultado (**saída**)



Algoritmos

Exemplo: Calcular o máximo divisor comum

Algoritmo em Português

Entrada

1. Leia dois números.
2. Divida o primeiro pelo segundo e guarde o resto.
3. Se o resto for diferente de 0 (zero),
então salta para passo 5
4. Escreva o segundo número e PARE.
5. Substitua o primeiro número pelo segundo.
6. Substitua o segundo número pelo resto da divisão.
7. Retorne ao passo 2.

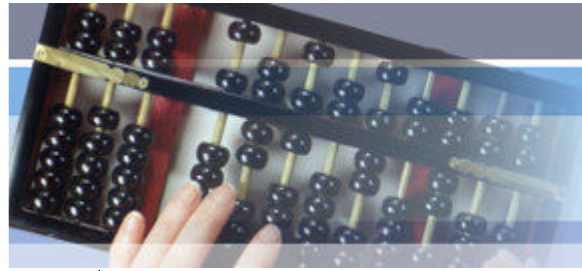
Condição

Saída

Salto

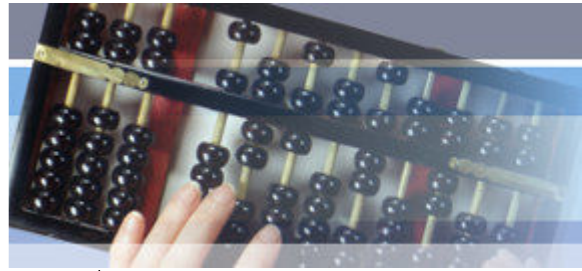
Início

Fim



Algoritmos

- **Cada instrução precisa ser:**
 - Não ambígua
 - Uma única operação bem definida
 - Tecnicamente viável
- **Representações possíveis:**
 - Diagramas
 - Modelos matemáticos
 - Linguagens de programação



Algoritmos

Exemplo: Calcular o máximo divisor comum

Algoritmo em Português

1. Leia dois números.
2. Divida o primeiro pelo segundo e guarde o resto.
3. Se o resto for diferente de 0 (zero),
então salta para passo 5
4. Escreva o segundo número e PARE.
5. Substitua o primeiro número pelo segundo.
6. Substitua o segundo número pelo resto da divisão.
7. Retorne ao passo 2.

Potencialmente
ambíguo

Não é uma única
instrução bem
definida

Linguagem de Programação

Opções de representar algoritmos:

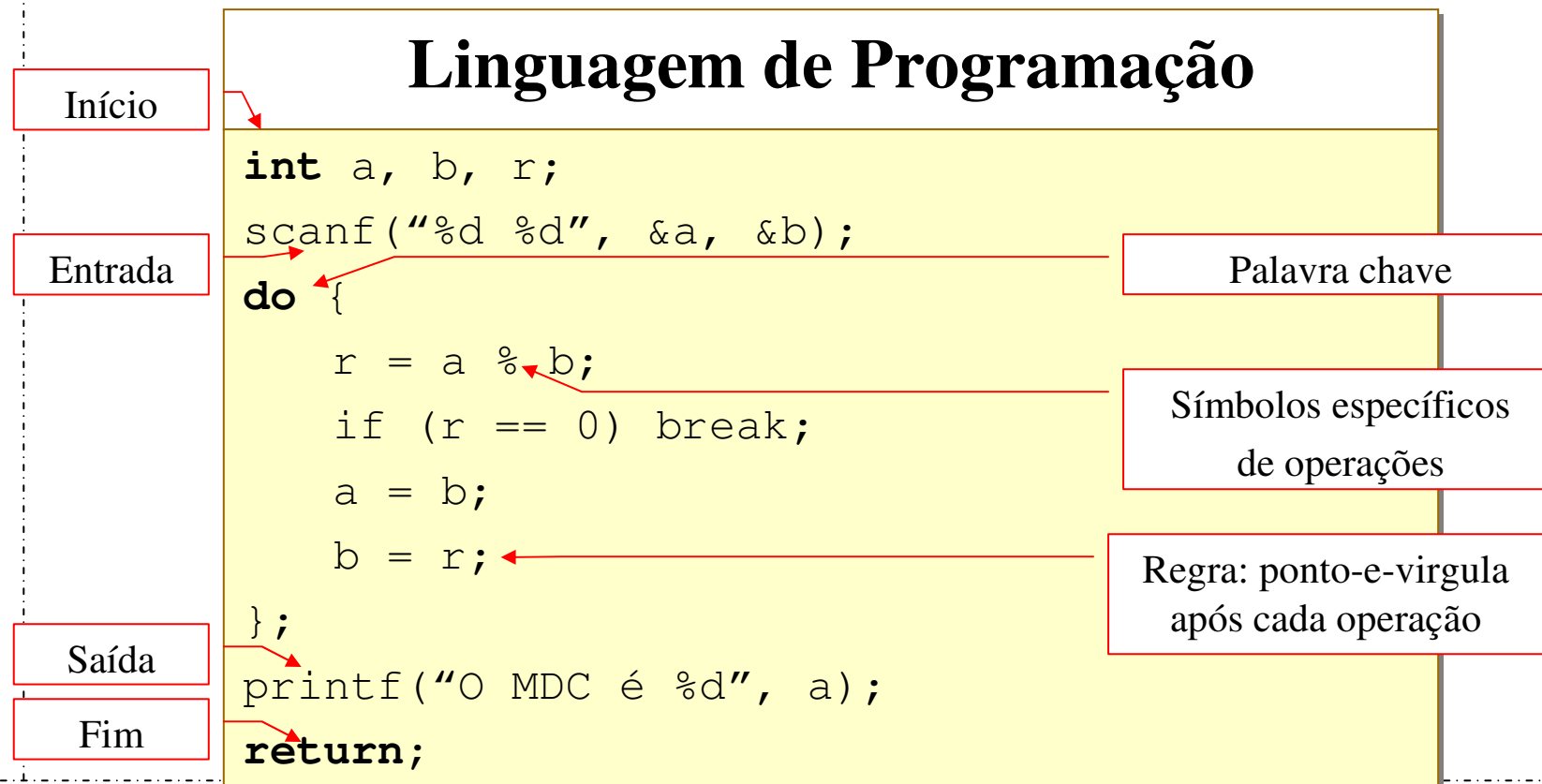
- Diagramas
- Uma linguagem específica para escrever algoritmos
- Linguagens de programação

Linguagem de programação

- **Conceito:**
 - A linguagem de programação é um veículo para se escrever algoritmos.
- **Características:**
 - Vocabulário restrito
 - Regras de sintaxe
 - Recursos automáticos de verificação da sintaxe da linguagem

Exemplo

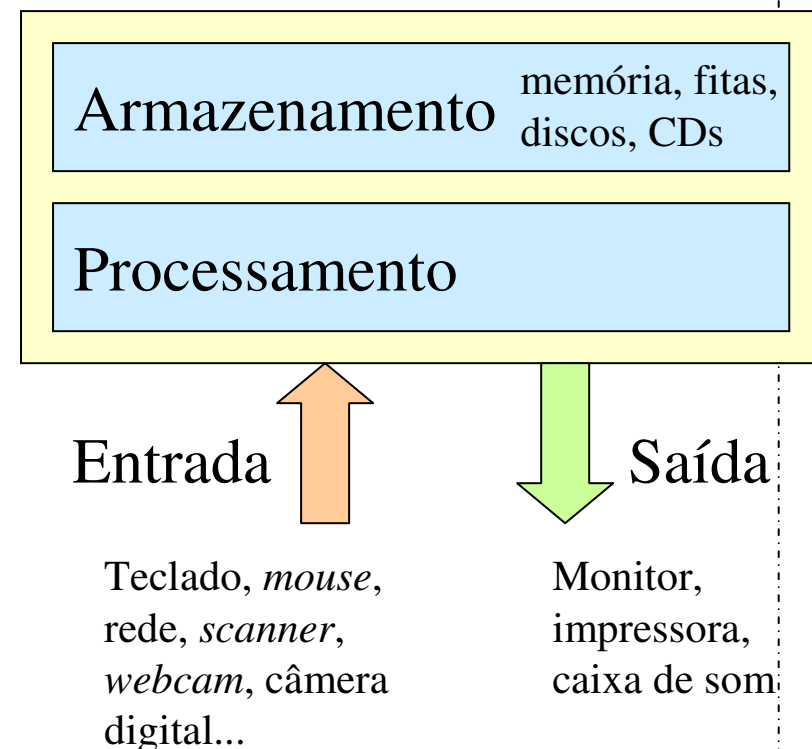
Exemplo: Calcular o máximo divisor comum



O Computador

Conceito:

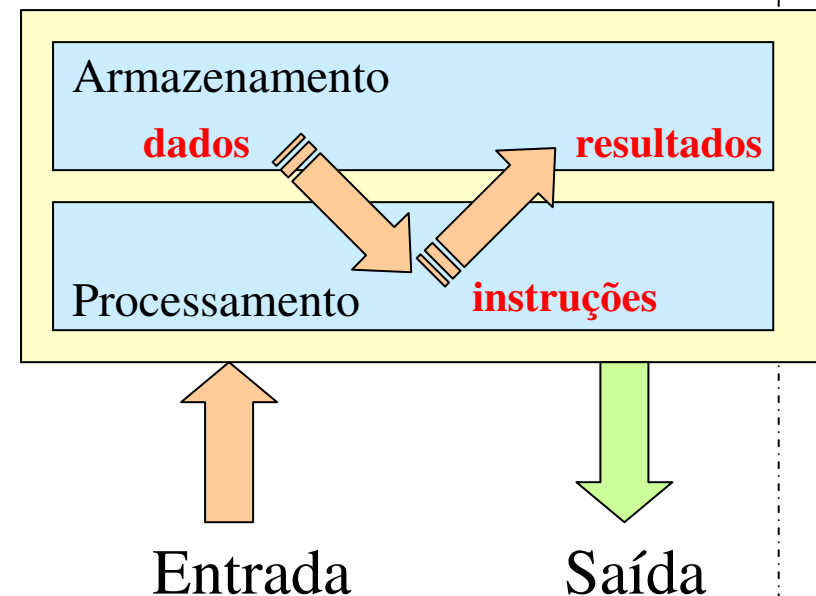
- Realiza **processamento** sobre dados **armazenados** no computador.
- Executa **operações matemáticas** e **lógicas** sobre dados.
- Recebe dados do meio externo (**entrada**)
- Apresenta os resultados para o meio externo (**saída**)



O Computador

Instruções de Máquina:

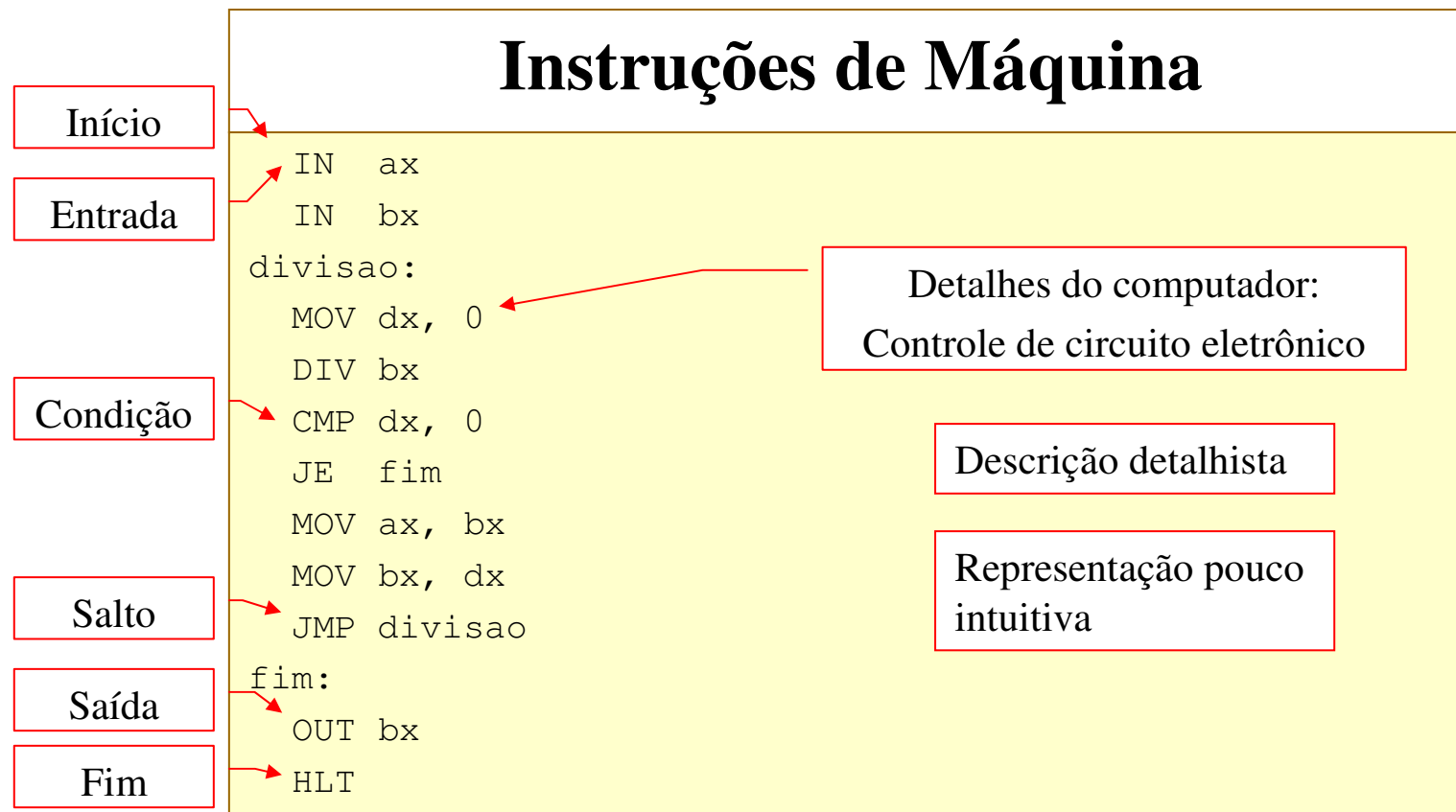
- Representam primitivas operações sobre dados.
- Em formato digital próprio (código de máquina)



O computador é uma máquina **rápida** e **eficiente**
para **simular algoritmos**!

O Computador

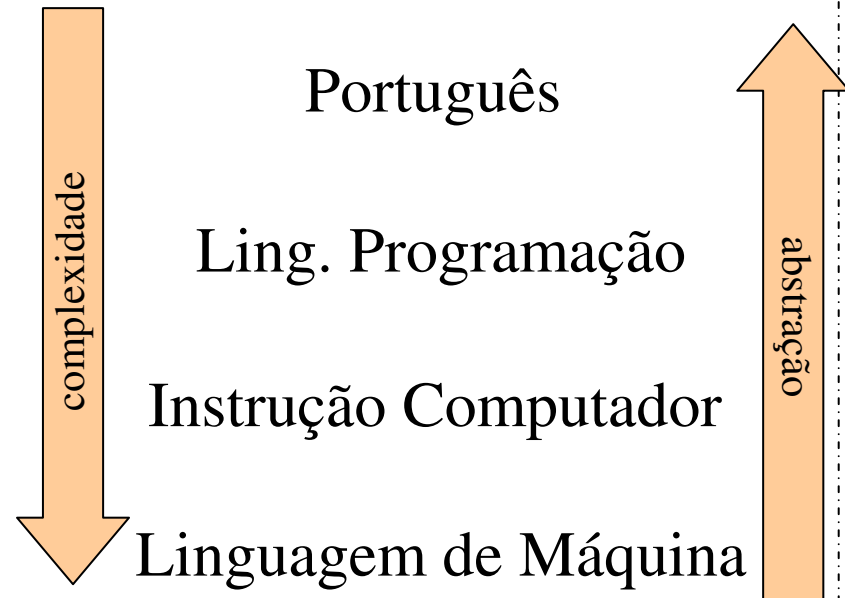
Exemplo: Calcular o máximo divisor comum



Níveis de Abstração

Dois casos extremos:

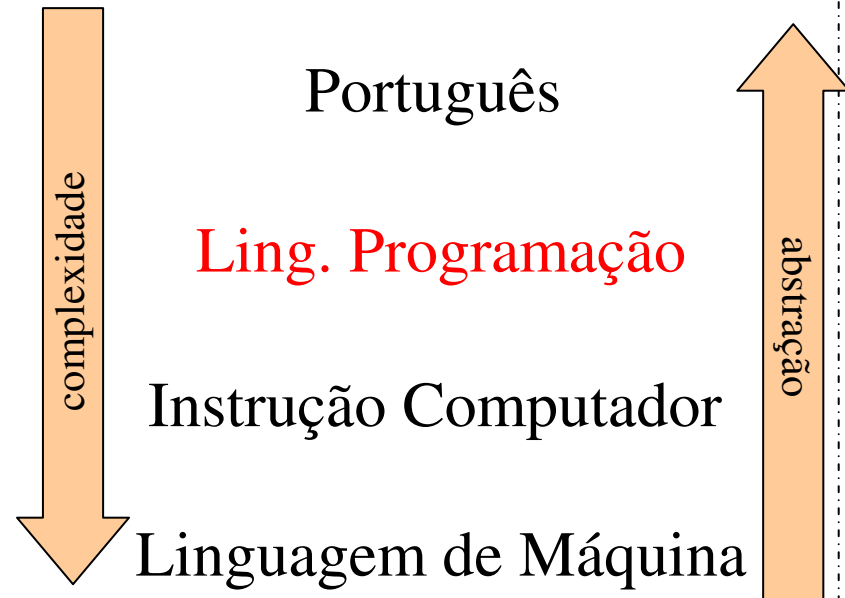
- Português:
 - Fácil, intuitivo
 - Computador não entende
 - Ambíguo, mal definido
- Linguagem de Máquina:
 - Complexo e trabalhoso
 - Única forma aceita pelo computador
 - Preciso, bem definido
 - Envolve detalhes específicos do computador, irrelevantes para o algoritmo



Níveis de Abstração

Objetivos:

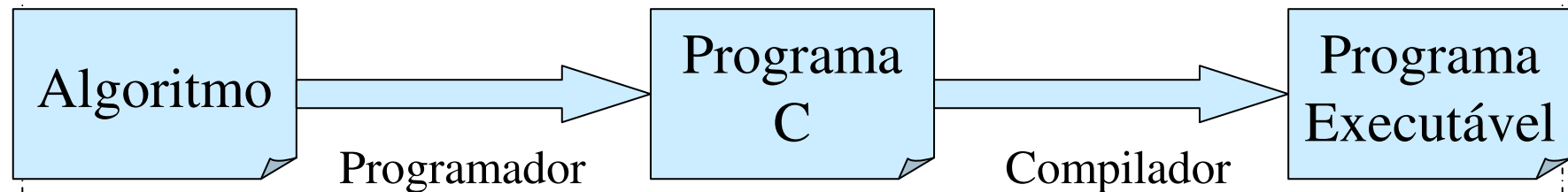
- Descrição precisa do algoritmo
- Independente do computador
- Nível de complexidade intermediário!



➡ **Linguagem de programação**

Construção do Programa

- Passo 1: Elaborar um algoritmo
- Passo 2: Reescrever o algoritmo em C
- Passo 3: Acionar o compilador

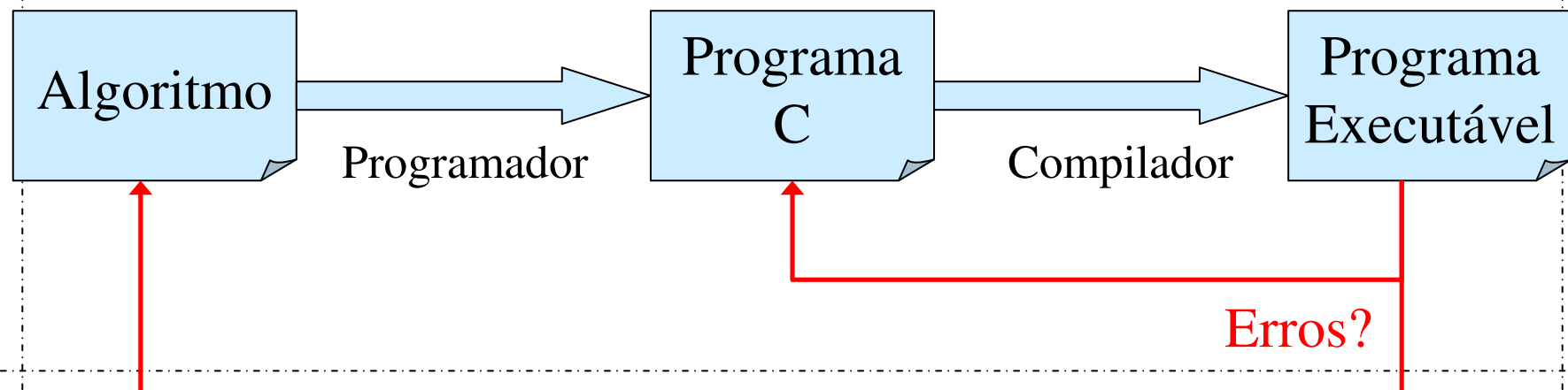


Construção do Programa

- Passo 4: Testar o programa

Erros? Verificar programa em C

Continuam erros? Verificar o algoritmo!



Curso de C

Primeiro Programa

Primeiro Programa

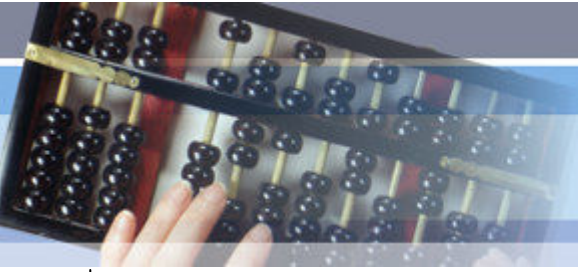
Objetivos:

- Compreender:
 - O modelo genérico usado para escrever programas na linguagem C

Primeiro Programa

Roteiro:

- O primeiro programa em C
- Estrutura do código fonte
 - Comentários
 - Diretivas de compilador
 - Procedimento principal
- Estilo do código fonte



Exemplo

O Programa “Bom Dia”:

```
// PrimeirosPassos.c: Nosso primeiro programa em C

#include <stdio.h> // mais um comentario
#include <stdlib.h>

int main(int argc, char* argv[]) {
    printf("O primeiro programa lhe deseja um bom dia!");
    return 0;
}
```

Estrutura do Código Fonte

Comentários

Diretivas de
compilador

Procedimento
principal

Instrução

Instrução

Pontuação

```
// PrimeirosPassos.c
```

```
#include <stdio.h>
```

```
#include <stdlib.h>
```

```
int main(int argc, char* argv[]) {
```

```
    printf("bom dia!");
```

```
    return 0;
```

```
}
```


Estrutura do Código Fonte

Comentários:

- Texto ignorado pelo compilador
- Documentação útil para descrever trechos do algoritmo
- Possível em qualquer posição do código fonte
- Duas formas para comentários:
 - Uma linha: `// Comentário ...`
 - Várias linhas: `/* Comentário...
mais comentários ... */`

```
// PrimeirosPassos.c: Nosso primeiro programa em C
```

```
/* PrimeirosPassos.c: Nosso primeiro programa em C */
```

```
/* PrimeirosPassos.c:  
Nosso primeiro programa em C */
```



Estrutura do Código Fonte

Diretivas de Compilador:

- Informam outros arquivos que devem ser consultados antes de compilar
- Definem parâmetros utilizados pelo compilador
- Colocadas no início do código fonte

```
#include <stdio.h>  
#include <stdlib.h>  
#include <math.h>
```



Estrutura do Código Fonte

Procedimento principal:

- Seqüência de instruções
- Pontuação: ponto-e-vírgula termina instruções
- Chaves agrupam instruções relacionadas

```
int main(int argc, char* argv[]) {  
    printf("bom dia!");  
    return 0;  
}
```

Estrutura do Código Fonte

Procedimento principal:

- Siga sempre o seguinte esqueleto:

```
int main(int argc, char* argv[]) {  
    ...  
    Algoritmo  
    ...  
    return 0;  
}
```

Obrigatório

Obrigatório



Estrutura do Código Fonte

Práticas interessantes:


- Linhas em branco são ignoradas

```
int main(int argc, char* argv[]) {  
    printf("bom dia!");  
    return 0;  
}
```

(recomendado)

(permitido)

```
int main(int argc, char* argv[]) {  
  
    printf("bom dia!");  
  
    return 0;  
}
```



Estrutura do Código Fonte

Práticas interessantes:

- Espaços e tabulações são ignoradas

```
int main(int argc, char* argv[]) {  
    printf("bom dia!");  
    return 0;  
}
```

(recomendado)

(permitido,
mais confuso)

```
int main(int argc, char* argv[]) {  
printf("bom dia!");    return 0;  
}
```



Estrutura do Código Fonte

Práticas interessantes:

- De preferência, uma instrução por linha

```
int main(int argc, char* argv[]) {  
    printf("bom dia!");  
    return 0;  
}
```

(recomendado)

(permitido,
mais confuso)

```
int main(int argc, char* argv[]) {  
    printf("bom dia!"); return 0;  
}
```