



UNIVERSIDADE
FEDERAL DO CEARÁ



CK0179 - Programação Computacional para Engenharia : Noções de Lógica e a Linguagem C

Prof. Maurício Moreira Neto

Objetivos

- Apresentar os conceitos de elementos de lógica de programação e sua aplicação no cotidiano
- Definir algoritmo
- Relacionar lógica a algoritmos: lógica de programação
- Demonstrar o uso de algoritmos em situações do dia-a-dia
- Saber utilizar as diversas representações de algoritmos

O que é lógica?

Noções de lógica

- “Lógica” é originária do grego *logos* = **Linguagem Racional**
- Ciência das formas do pensamento, raciocínio
- A lógica é o ramo da filosofia que cuida das regras do bem pensar, ou do pensar correto, sendo, portanto, um instrumento do pensar
- Segundo o ***Michaelis***:
 - "Análise das formas e leis do pensamento, não se preocupando com a sua produção mas, somente, sua forma. Ou seja, a maneira que forma um pensamento é organizado e apresentado, possibilitando a uma conclusão"

Noções de lógica

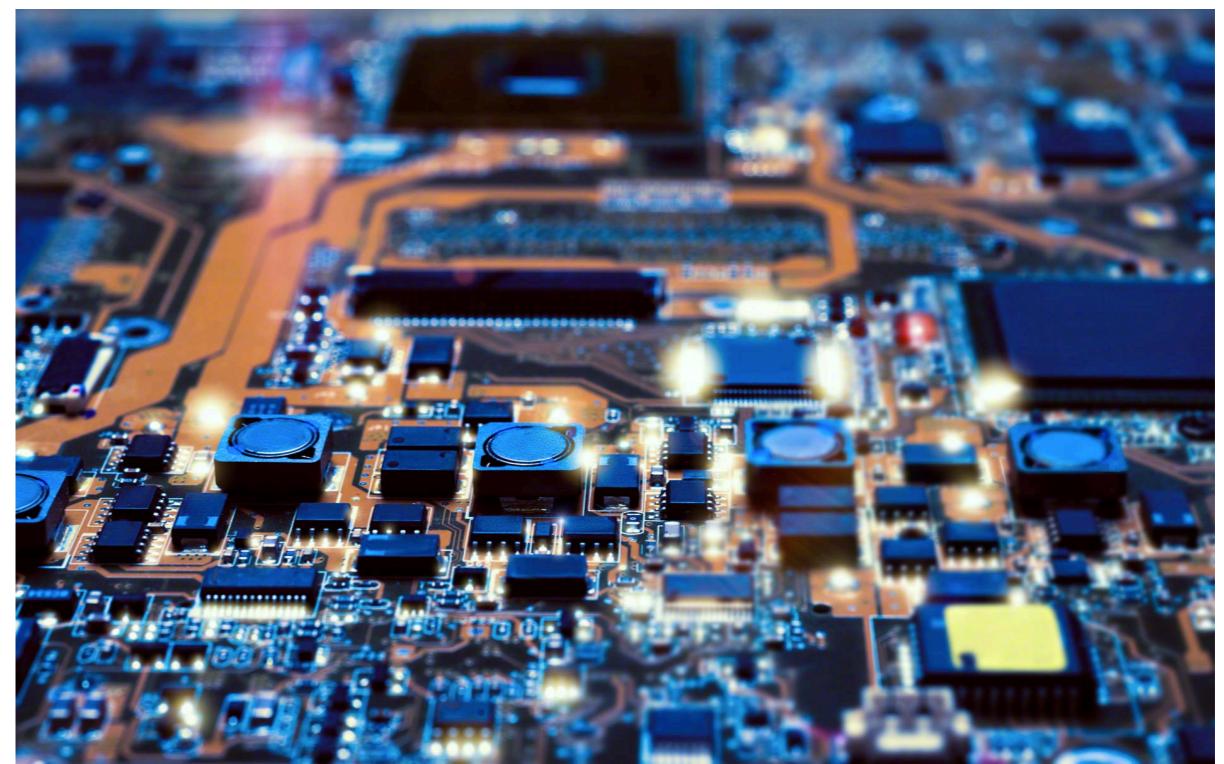
- **Silogismo:** a partir de duas **premissas**, podemos chegar a uma **conclusão**
 1. Todo mamífero é um animal
 2. Todo cavalo é mamífero
 3. Portanto, todo cavalo é um animal.
- Lógica no dia-a-dia
 - Sempre que quisermos colocar ordem no pensamento, estamos usando a lógica

Noções de lógica

- O homem usa o raciocínio lógico para realizar suas atividades
- Sempre é estabelecido uma sequencia adequada para realizar uma determinada tarefa
- Ex: Mudança marcha de um carro
 1. Coloca-se a marcha no ponto morto
 2. Passa-se a marcha na posição correta correspondente
 3. Após determinado valor de velocidade, muda-se a marcha novamente

Noções de lógica

- A lógica é aplicada a diversas ciências
- A lógica na computação é usada em todas as áreas desde o **hardware até o software**



Lógica para programação

- Computadores são máquinas e, por si sós, não podem ser inteligentes
 - É necessário que alguém as projete
- Um computador pode realizar um cálculo 10 milhões de vezes mais rápido que o cérebro humano
- Os programas de computadores são criados usando-se os conceitos de lógica

Lógica para programação

- A lógica é usada na programação para desenvolver soluções logicamente válidas e coerentes para os problemas a serem resolvidos
- Raciocínio é algo abstrato, independente de linguagem ou idioma
- Algoritmos são utilizados para manter a lógica independente de uma linguagem de programação
- **O que é um algoritmo?**

Algoritmo

- É uma sequência finita de passos (ou instruções) que visam atingir um objetivo bem definido
- A lógica é utilizada para ordenar os passos
- A sequência deve ser:
 - Finita
 - Não ambígua
- **Algoritmos estão continuamente presentes em nosso dia-a-dia**
 - receitas culinárias
 - instruções para realizar alguma tarefa
 - dirigir um determinado veículo
 - ...

Exemplo de algoritmos

- ...trocar uma lâmpada
- ...fazer um sanduíche
- ...fazer um omelete
- ...instalar um DVD Player
- ...tirar o carro da garagem
- ...sacar dinheiro do caixa eletrônico

Exemplo de Algoritmos

- Sacar dinheiro em um caixa eletrônico
 - 2. Ir até um caixa eletrônico
 - 3. Inserir cartão do caixa eletrônico
 - 4. Retirar cartão do caixa eletrônico
 - 5. Escolher opção de saque
 - 6. Digitar valor a ser sacado
 - 7. Digitar código de letras
 - 8. Inserir cartão do caixa eletrônico
 - 9. Retirar cartão do caixa eletrônico
 - 10. Se o saldo for maior ou igual à quantia desejada e a quantia desejada for menor ou igual ao limite diário disponível, sacar; senão, mostrar mensagem de indisponibilidade de saque.

Algoritmo

- Um algoritmo é um procedimento computacional definido composto de 3 partes
 - **Entrada de dados**
 - São os dados do algoritmo informados pelo usuário
 - **Processamento de dados**
 - São os procedimentos utilizados para chegar ao resultado
 - É responsável pela obtenção dos dados de saída com base nos dados de entrada
 - **Saída de dados**
 - São os dados já processados, apresentados ao usuário

Método p/ construção de algoritmos

- 1. Compreender completamente o problema**
2. Definir os dados de entrada, as condições iniciais, que dados serão fornecidos e quais objetos fazem parte do problema
- 3. Definir os dados de saída**
4. Definir o processamento, cálculos a serem efetuados, restrições. Transformação dos dados de entrada em dados de saída. Identificar as responsabilidades
- 5. Construir o algoritmo utilizando técnicas descritas a seguir**
6. Testar o algoritmo usando simulações

Tipos de representação de algoritmos

- Descrição Narrativa
- Fluxograma
- Pseudocódigo ou Portugol

Descrição narrativa

- Consiste em analisar o enunciado do problema e escrever os passos a serem seguidos para a sua resolução utilizando **uma linguagem natural** (português, por exemplo)
- **Vantagem**
 - A linguagem já é conhecida
- **Desvantagem**
 - Ambiguidade, múltiplas interpretações, imprecisão nas instruções
- **Exemplo:** o programador disse ao amigo que o seu programa era o melhor

Exemplos de algoritmos: Descrição narrativa

- Algoritmos para mostrar o resultado da multiplicação de dois números

Passo 1: Receber os dois números que serão multiplicados.

Passo 2: Multiplicar os números.

Passo 3: Mostrar o resultado obtido na multiplicação.

Exemplos de algoritmos:

Descrição narrativa

- Faça um algoritmo para mostrar o resultado da divisão de dois números
 - Descrição narrativa

Passo 1: Receber os dois números que serão divididos.

Passo 2: Se o segundo número for igual a zero, não será possível ser feita a divisão. Caso contrário, dividir os números e mostrar o resultado da divisão

Fluxograma

- Consiste em analisar o enunciado do problema e escrever os passos a serem seguidos para a sua resolução utilizando **símbolos gráficos predefinidos**



Entrada de dados



Linhas de fluxo: indica sequência das etapas e a direção do fluxo

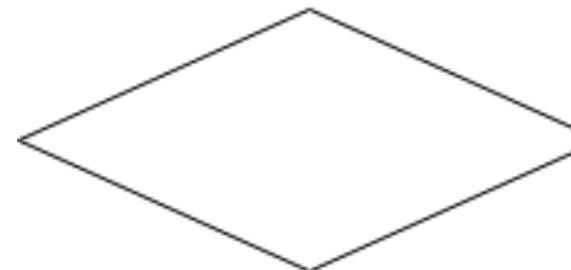


Saída de dados

Fluxograma



Início/Fim: Marca o início ou fim de um programa



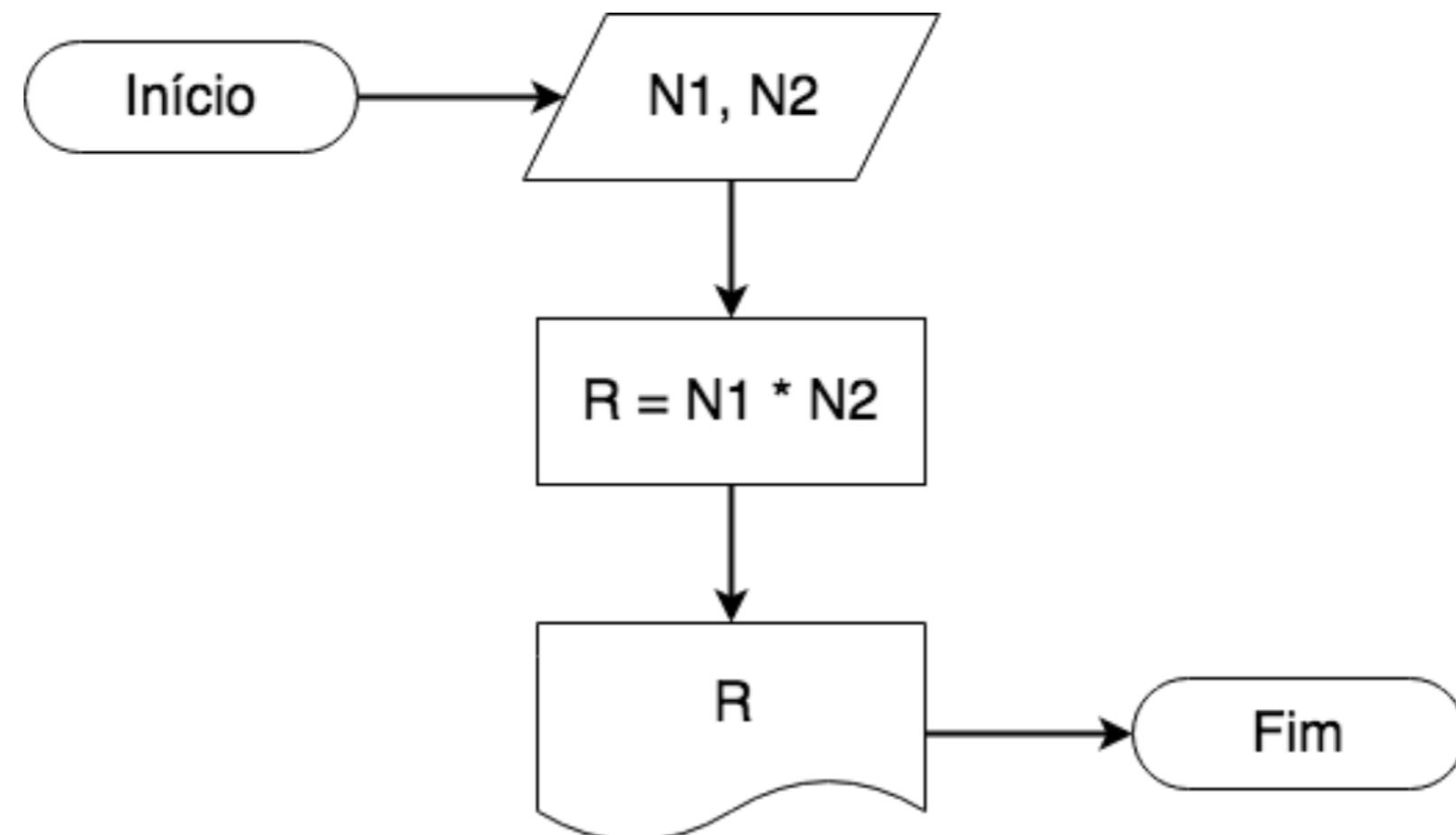
Decisão: indica desvios na sequência lógica de execução do programa



Processamento: qualquer operação com alteração no conteúdo de uma variável

Exemplos de fluxogramas

- Faça um algoritmo para mostrar o resultado da multiplicação de dois números
 - Fluxograma

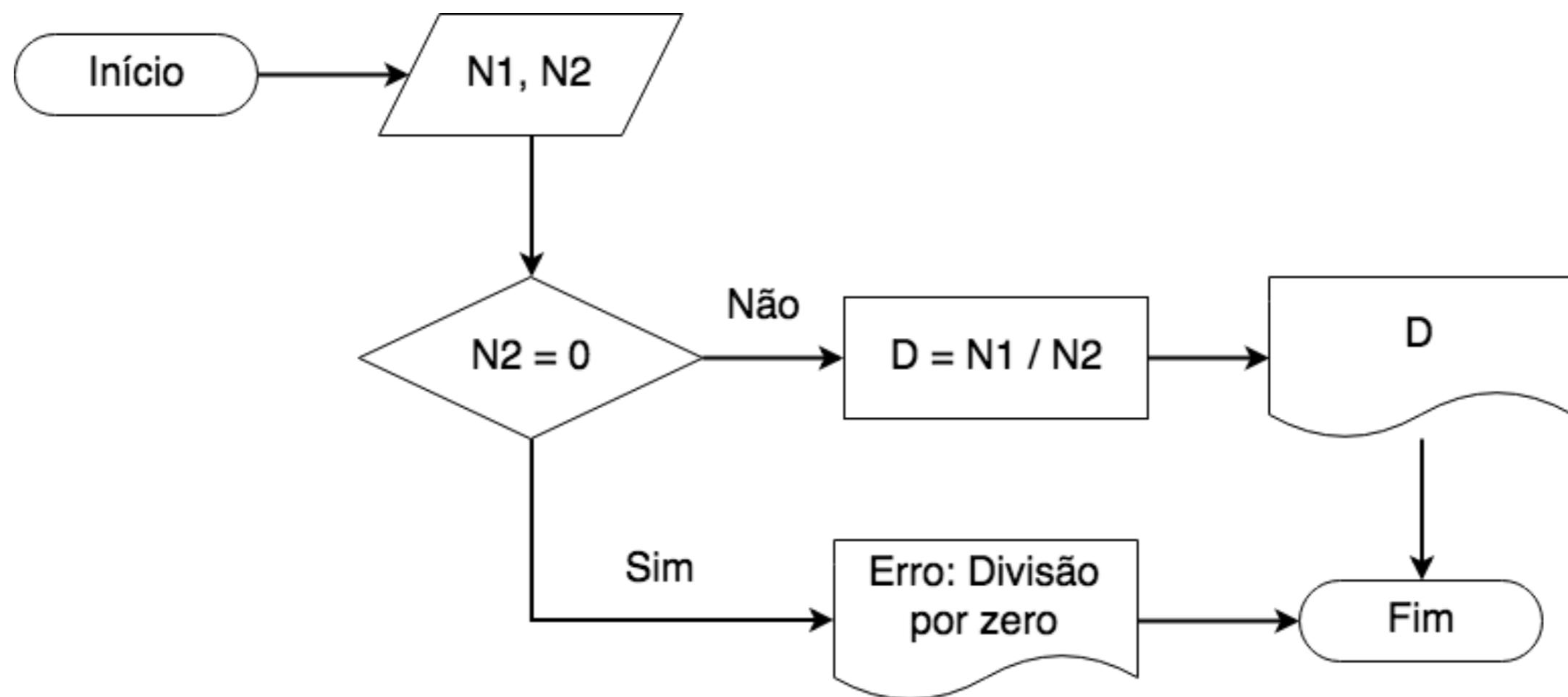


Exemplos de fluxograma

- Faça um algoritmo para mostrar o resultado da divisão de dois números
 - Fluxograma

Exemplos de fluxograma

- Faça um algoritmo para mostrar o resultado da divisão de dois números
 - Fluxograma



Fluxograma

- **Vantagem**
 - Entendimento de elementos gráficos é mais simples que o entendimento de textos
- **Desvantagem**
 - É necessário aprender a simbologia dos fluxogramas
 - Fluxograma pode ser muito conciso, dificultando sua transição para um programa

Pseudocódigo ou Portugol

- Consiste em analisar o enunciado do problema e escrever os passos a serem seguidos para a sua resolução por meio de **regras bem definidas**
- **Vantagem**
 - A transição do algoritmo para qualquer linguagem é quase imediata, bastando conhecer as palavras reservadas dessa linguagem que serão utilizadas
- **Desvantagem**
 - É preciso aprender as regras do pseudocódigo

Exemplos de algoritmos

- Faça um algoritmo para mostrar o resultado da multiplicação de dois números
 - Pseudocódigo

```
algoritmo
declare N1, N2, M número
escreva("Digite dois números")
leia(N1)
leia(N2)
M = N1 * N2
escreva("Multiplicação = ", M)
fim algoritmo
```

Exemplos de algoritmos

- Faça um algoritmo para mostrar o resultado da divisão de dois números
 - Pseudocódigo

Exemplos de algoritmos

- Faça um algoritmo para mostrar o resultado da divisão de dois números
 - Pseudocódigo

```
algoritmo
declare N1, N2, D número
escreva("Digite dois números")
leia(N1)
leia(N2)
se N2 = 0
    então escreva("Divisão por zero")
senão
    início
        D = N1 / N2
        escreva("Divisão = ", D)
    fim
fim algoritmo
```

Exercícios

- Faça um algoritmo para calcular a média aritmética entre duas notas de um aluno e mostrar sua situação, que pode ser aprovado ou reprovado, utilizando:
 - Descrição narrativa, fluxograma e pseudocódigo
- Faça um algoritmo para dizer se um número é par ou ímpar. Dica: % indica o resto da divisão.:
 - Descrição narrativa, fluxograma e pseudocódigo
- Faça um algoritmo para converter uma temperatura dada em Celsius para Fahrenheit, utilizando:
 - Descrição narrativa, fluxograma e pseudocódigo

Tipos de processamento

- Ao elaborar um algoritmo, devemos ter em mente qual o tipo de processamento será executado
- **Basicamente, existem 3 tipos de processamento:**
 - Processamento seqüencial
 - Processamento condicional
 - Processamento com repetição
 - Repetição determinada
 - Repetição Indeterminada

Tipos de processamento

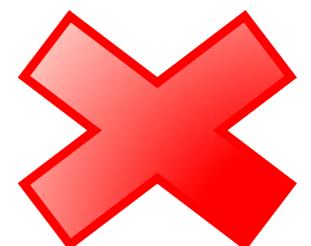
- Processamento sequencial
 - Instruções são executadas uma após a outra
 - Não existe desvio na sequencia das instruções
 - Cada instrução é executada uma única vez
- Exemplo:
 - Imprimir a media aritmética de duas notas

```
algoritmo media
    declare nota1, nota2, media : numérico
    Leia (nota1)
    Leia (nota2)
    media = (nota1 + nota2)/2
    Imprima (media)
```

Tipos de processamento

- Processamento sequencial
 - A ordem das instruções é importante!

```
algoritmo media
declare nota1, nota2, media : numérico
Leia (nota1)
Leia (nota2)
Imprima (media)
media = (nota1 + nota2)/2
```



```
algoritmo media
declare nota1, nota2, media : numérico
Leia (nota1)
Leia (nota2)
media = (nota1 + nota2)/2
Imprima (media)
```



Tipos de processamento

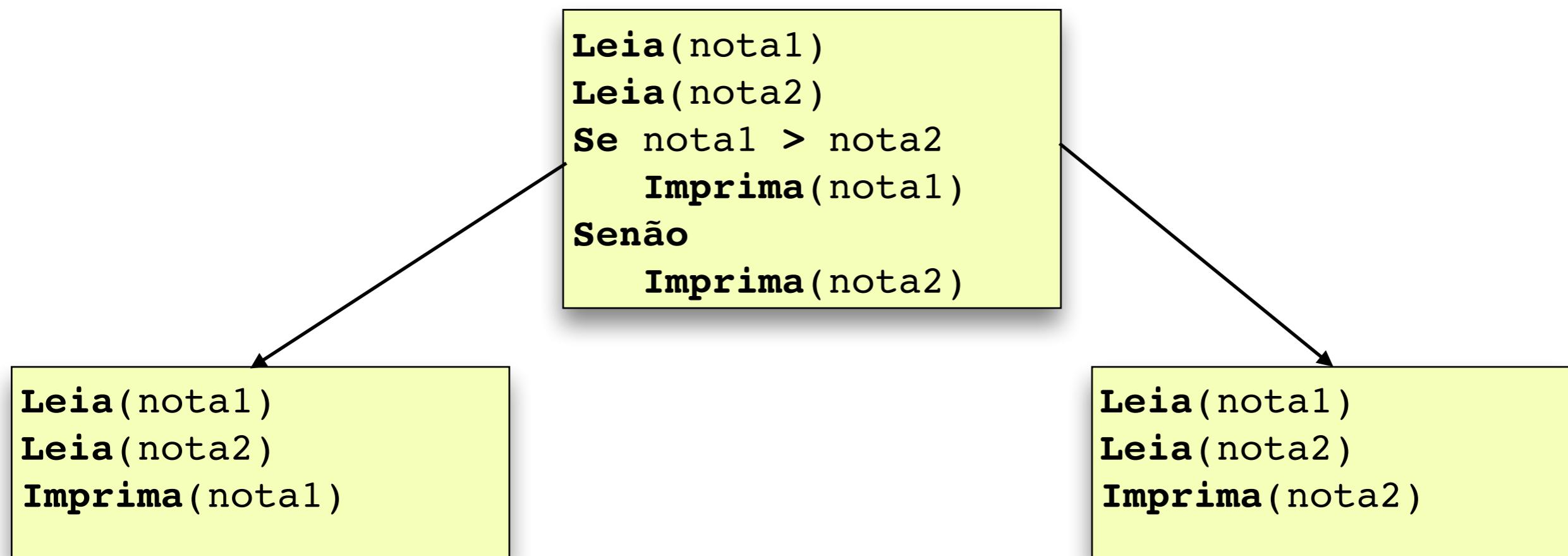
- **Processamento condicional**

- Um conjunto de instruções (pode ser apenas uma) que pode ou não ser executado
- Depende de uma condição
- Se a condição testada for verdadeira, o conjunto de instruções é executado



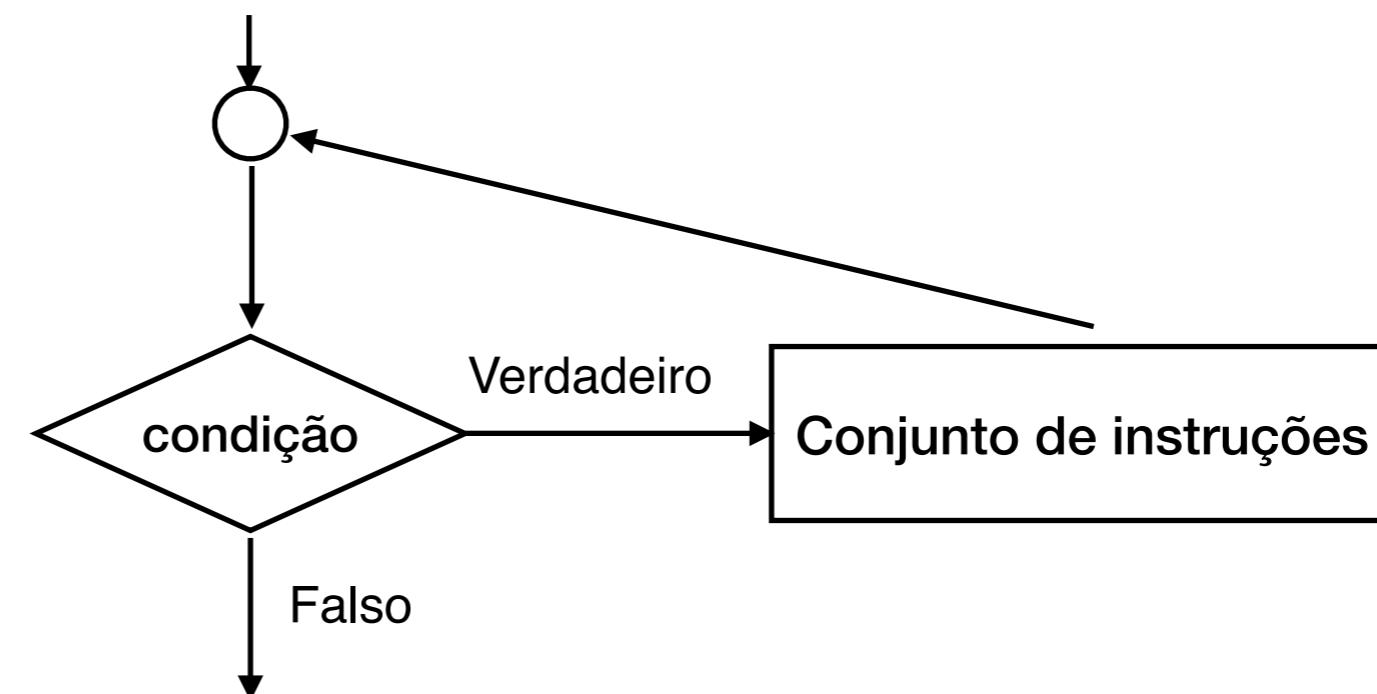
Tipos de processamento

- **Processamento condicional**
 - As instruções executadas dependem da situação
- Exemplo:
 - Imprimir a maior dentre duas notas lidas



Tipos de processamento

- **Processamento com repetição**
 - Um conjunto de instruções (pode ser apenas uma) é executado um número definido ou indefinido de vezes
 - Pode ser determinado por uma condição de parada
 - O conjunto de instruções é executado enquanto a condição for verdadeiro
 - O teste da condição é realizado antes de qualquer operação



Tipos de processamento

- **Processamento com repetição**
 - Também chamado de laços condicionais
 - Repetem um conjunto de comandos em seu interior
- Exemplo:
 - Imprimir a soma dos números inteiros de 1 a N
 - $\text{Soma} = 1 + 2 + 3 + \dots + N$
 - Necessário identificar o que deve ser repetido no algoritmo

$$\text{Soma} = 1 + 2 + 3 + \dots + N$$

Tipos de processamento

- **Processamento com repetição**
 - Ex: Imprimir a soma dos números inteiros de 1 a N
 - Soma = 1 + 2 + 3 + ... + N
 - Identificar: valor inicial (nro = 1), valor final (N), onde o resultado será armazenado (soma), quando parar (nro <= N), variável (contador) que controla o número de repetições (nro), ...

```
Leia(N)
soma = 0
nro = 1
Enquanto nro <= N
    soma = soma + nro
    nro = nro + 1
Imprima(soma)
```

Teste de mesa

- Após desenvolver um algoritmo é preciso testá-lo
- Uma maneira de fazer o teste é usando o **teste de mesa**
 - Basicamente, esse teste consiste em seguir as instruções do algoritmo de maneira precisa para verificar se o procedimento utilizado está correto ou não
 - Tentar utilizar um caso onde se conhece o resultado esperado
 - Permite reconstituir o passo a passo do algoritmo



Teste de mesa

- Criar uma tabela de modo que:
 - cada coluna represente uma variável
 - As linhas correspondem as alterações naquela variável (de cima para baixo)

valor	N	soma

Teste de mesa

- **Exemplo:** Imprimir a média dos valores dos números positivos digitados. Parar quando um valor negativo ou zero for digitado
 - Valores digitados: 4, 2, 3 e -1
 - Média é 3

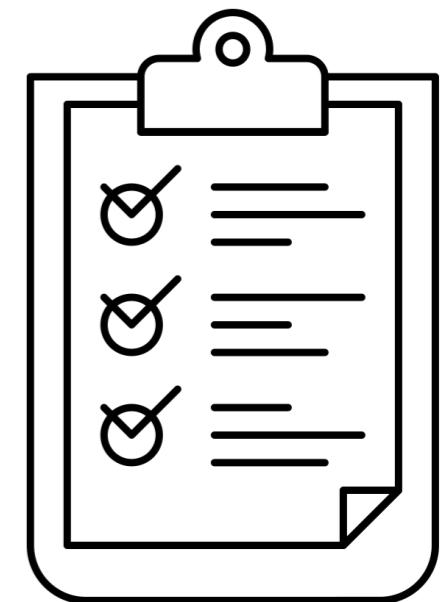
```

soma = 0
N = 0
Leia(valor)
Enquanto valor > 0
    soma = soma + valor
    N = N + 1
    Leia(valor)
Imprima(soma/N)
  
```

valor	N	soma
4	0	0
2	1	4
3	2	6
-1	3	9

Metodologia de programação

- A resolução de um problema começa com a definição dos dados e tarefas básicas
- Esta definição inicial é feita em nível bem alto e geral
- Não há preocupação com os detalhes (refinamentos)



Metodologia de programação

- Refinamentos Sucessivos (Top-Down)
 - Consiste em pegar um grande problema, de difícil solução, e dividi-lo em problemas menores que devem ser mais facilmente resolvidos
 - Decompor uma ou várias tarefas em sub-tarefas mais detalhadas
 - É um processo iterativo, isto é, sub-tarefas podem ser decompostas em sub-tarefas ainda mais detalhadas

Algoritmo - Trocar um pneu furado

1. Levantar o carro parcialmente
2. Retirar o pneu furado
3. Instalar o novo pneu
4. Abaixar o carro

Algoritmo - Trocar um pneu furado

1. **Retirar o estepe**
2. Levantar o carro parcialmente
3. Retirar o pneu furado
4. Instalar o novo pneu
5. **Apertar bem as porcas**
6. Abaixar o carro

Refinamentos sucessivos

Algoritmo – Trocar um pneu furado

- 1. Pegar as ferramentas no porta-malas**
- 2. Retirar o estepe**
- 3. Instalar o macaco**
- 4. Levantar o carro parcialmente**
- 5. Afrouxar os parafusos do pneu furado**
- 6. Retirar o pneu furado**
- 7. Instalar o novo pneu**
- 8. Apertar bem as porcas**
- 9. Abaixar o carro**
- 10. Guardar o pneu furado e as ferramentas**

Refinamentos sucessivos

- O algoritmo proposto pode ainda ser refinado de várias outras formas
 - O que fazer se o macaco não estiver no porta-malas?
 - O que fazer se o estepe também estiver vazio?
 - Deve-se sempre puxar o freio de mão antes de executar estas operações
 - Limpar as mãos
 - Consertar o pneu furado
 - ...

Linguagem de Programação

Linguagem de programação

- É uma linguagem artificial utilizada para expressar sequências de ações ou comandos que devem ser executados pela máquina (computador)
- Existem várias linguagens de programação
 - Python
 - Java
 - C, C++
 - Assembly
- **A linguagem que iremos estudar:**

C

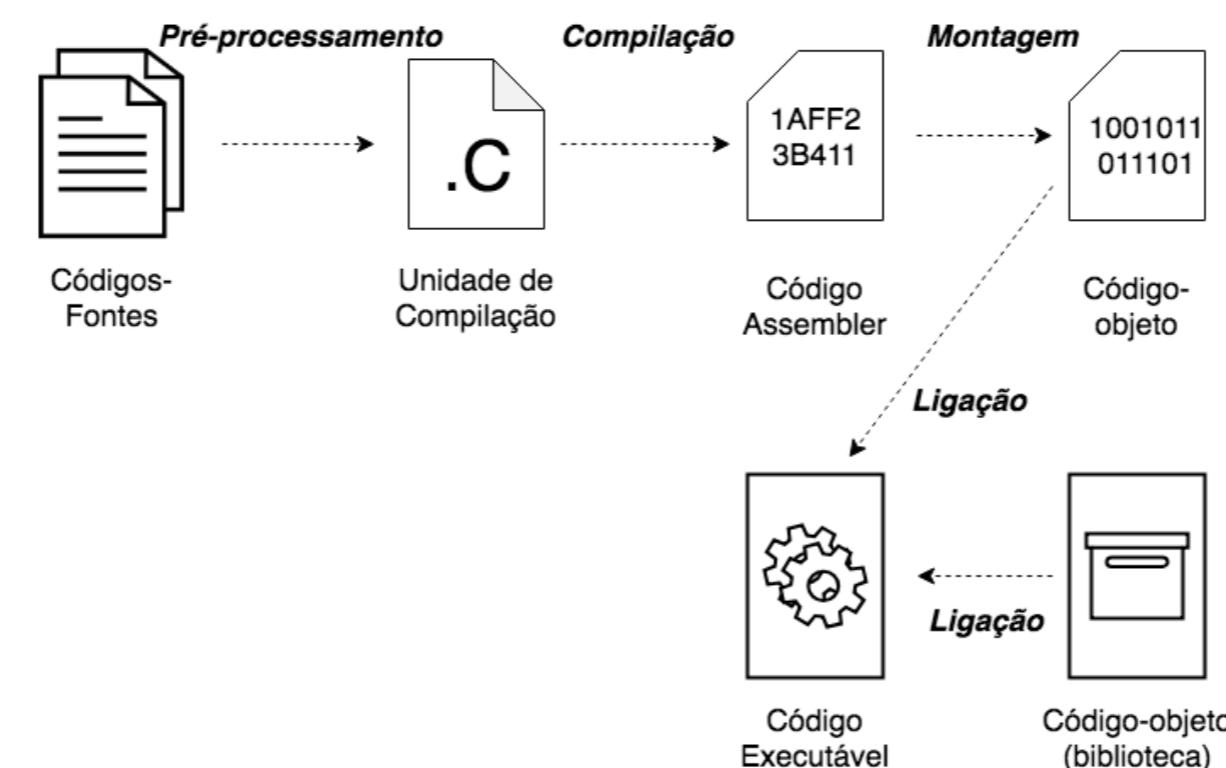
O que é a linguagem C

- A linguagem C foi desenvolvida nos anos 70 por Dennis Ritchie em um computador DEC PDP-11, usando o sistema operacional Unix.
- O C++ foi concebido após a detecção de algumas limitações da linguagem C. Por isso, o estudioso Bjarne Stroustrup acrescentou ao C novas funcionalidades e em 1983 o C++ foi criado.



A linguagem C

- Os programas C são textos contendo os comandos e declarações que devem ser traduzidos para a linguagem de máquina da arquitetura-alvo
- Compilação
 - Transformação do código-fonte → código executável



A linguagem C

- **Código-fonte:** é o código escrito em uma linguagem de programação (pode ter diversos códigos armazenados em arquivos).
- **Código-objeto:** é código gerado na linguagem de máquina da arquitetura-alvo. Não pode ser diretamente executado no processador.
- **Código executável:** é o código gerado na linguagem de máquina, com todas as referências resolvidas, podendo ser executado diretamente pelo processador.

A linguagem C

- Os arquivos que contêm os programas possuem extensões padronizadas:
 - **.c** - **Os programas-fontes**, contendo o código a ser compilado, são armazenados em arquivos com extensão .c
 - **.h** - **As declarações dos programas-fontes** que podem ser usadas por outras unidades de compilação são armazenadas em arquivos com extensão .h (conhecidos como *arquivos-cabeçalhos*)
 - **.s** - **Os programas assembler**, gerados na segunda etapa de compilação, são armazenados em arquivos com extensão .s (são removidos após o término da compilação)
 - **.o** - **Os programas-objetos**, gerados na terceira etapa da compilação, são armazenados em arquivos com extensão .o

A linguagem C

- Os **arquivos-cabeçalhos** são códigos-fontes contendo as declarações de variáveis e funções que permitem ao compilador verificar a correção das referências feitas a esses elementos
- **Bibliotecas** são arquivos especiais que contêm o código-objeto de funções
- Por meio dessa funcionalidade, é possível criar um código **modularizado**

A linguagem C

- **Biblioteca-padrão** - Especifica um conjunto de funções que devem estar disponíveis em bibliotecas para serem incorporadas aos programas dos usuários
 - Ex: função de entrada e saída de dados (*stdio.h*)
- **Arquivos-cabeçalhos do sistema** - Contêm as declarações das variáveis e funções cujos códigos-objetos estão armazenados na biblioteca-padrão
- **Arquivos-cabeçalhos do usuário** - O programador pode desenvolver arquivos-cabeçalhos próprios contendo declarações de funções e variáveis

A linguagem C

- **Inclusão de arquivos-cabeçalhos** - Os arquivos-cabeçalhos são incluídos com a diretiva de pré-processamento **#include**
 - Ex: `#include <stdio.h> ; #include <math.h> ;
include <stdin.h>`
- A referência entre chaves `<>` = arquivos-cabeçalhos do sistema
- A referência entre aspas duplas `" "` = arquivos-cabeçalhos do usuário

A linguagem C

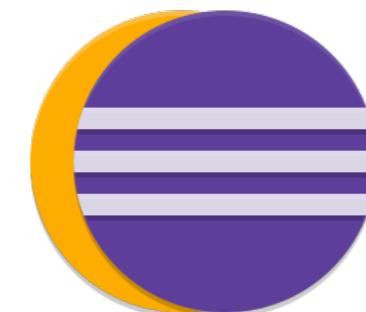
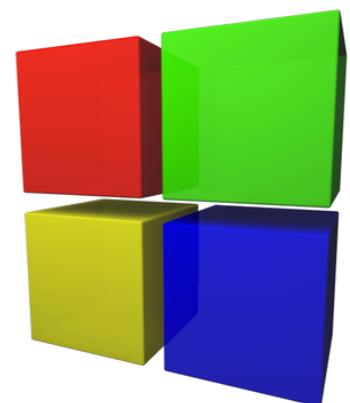
- O compilador que será usado é o gcc (GNU Compiler Collection)
 - **É simples! Basta relacionar os arquivos-fontes com o que se quer compilar**
- **gcc prog.c** - Compila o programa que está no arquivo `prog.c` e gera um executável (`a.out`)
- **gcc -o prog prog.c** - Compila o programa que está no arquivo `prog.c` e gera um arquivo executável `prog`
- **gcc prog.c -o prog_exem** - Compilar o programa cujo código está distribuído nos arquivos `prog.c` e gera um arquivo executável `prog_exem`

A linguagem C

- Para facilitar a programação utiliza-se, geralmente, um Ambiente Integrado de Desenvolvimento ou *Integrated Development Environment* - IDE
- Mas qual usar??!



NetBeans



A linguagem C

- Vamos fazer nosso primeiro programa em C!!

```
#include <stdio.h>

int main(void){
    printf("Olá Mundo!");
    return 0;
}
```

Obrigado!

maumneto@gmail.com

Referências

- André Luiz Villar Forbellone, Henri Frederico Eberspächer, Lógica de programação (terceira edição), Pearson, 2005, ISBN 9788576050247.
- Ulysses de Oliveira, Programando em C - Volume I - Fundamentos, editora Ciência Moderna, 2008, ISBN 9788573936599.
- **Slides baseados no material do site "Linguagem C Descomplicado"**
 - <https://programacaodescomplicada.wordpress.com/complementar/>