Aula 01 - O processo de programação

Prof. Me. Claudiney R. Tinoco

profclaudineytinoco@gmail.com

Faculdade de Computação (FACOM)
Bacharelado em Ciência da Computação (BCC)
Bacharelado em Sistemas de Informação (BSI)

Programação Procedimental (PP)
GBC014 - GSI002

Introdução

Objetivo da Computação

Auxiliar os seres humanos em trabalhos repetitivos e manuais, diminuindo esforços e economizando tempo.

Introdução

Objetivo da Computação

Auxiliar os seres humanos em trabalhos repetitivos e manuais, diminuindo esforços e economizando tempo.

- O computador é capaz de auxiliar em "qualquer coisa" que lhe seja solicitada, mas
  - Não tem iniciativa;
  - Não é independente;
  - Não é criativo nem inteligente.

Introdução

Objetivo da Computação

Auxiliar os seres humanos em trabalhos repetitivos e manuais, diminuindo esforços e economizando tempo.

- O computador é capaz de auxiliar em "qualquer coisa" que lhe seja solicitada, mas
  - Não tem iniciativa;
  - Não é independente;
  - Não é criativo nem inteligente.
- É necessário que o computador receba suas instruções nos mínimos detalhes, para que tenha condições de realizar suas tarefas.

Finalidade de um computador

- O computador deve receber, manipular e armazenar dados (todas essas operações são realizadas por meio de programas);

Finalidade de um computador

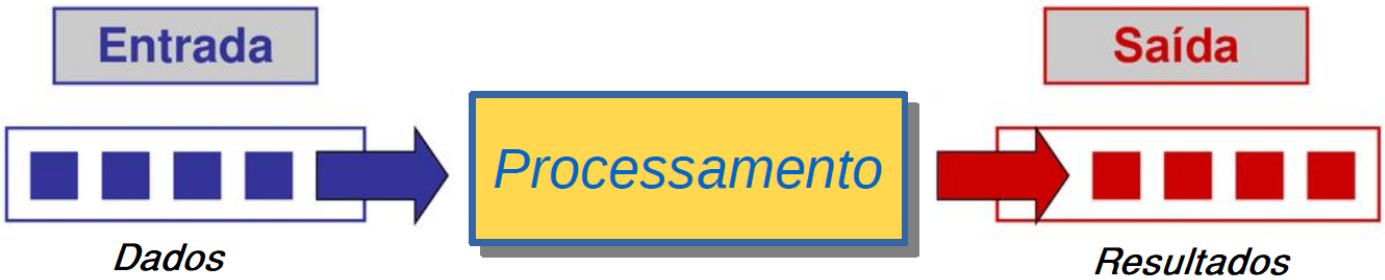
- O computador deve receber, manipular e armazenar dados (todas essas operações são realizadas por meio de programas);
- Quando construímos um software para realizar determinado processamento de dados, devemos escrever um programa ou vários programas interligados;

Finalidade de um computador

- O computador deve receber, manipular e armazenar dados (todas essas operações são realizadas por meio de programas);
- Quando construímos um software para realizar determinado processamento de dados, devemos escrever um programa ou vários programas interligados;
- Para que o computador consiga ler o programa e entender o que fazer, este programa deve ser escrito em uma linguagem que o computador entenda $\rightarrow$ LINGUAGEM DE PROGRAMAÇÃO.

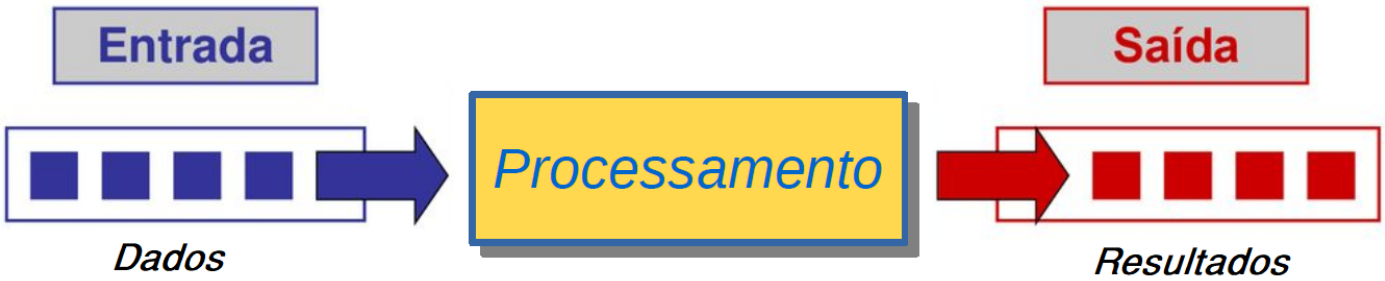
Funcionamento geral de um programa

- **Dados** $\rightarrow$ informações vindas de usuários ou de outras máquinas;



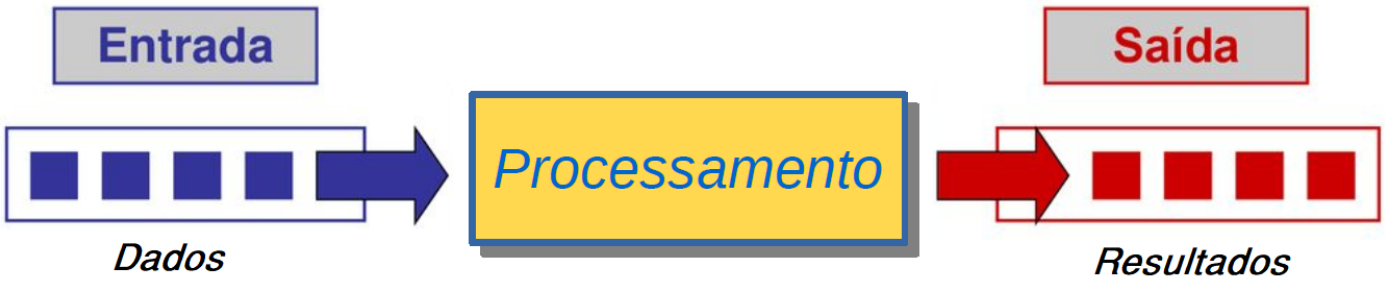
Funcionamento geral de um programa

- **Dados** → informações vindas de usuários ou de outras máquinas;
- **Processamento** → transformação dos dados, de acordo com os desejos do usuário ou de outra máquina;



Funcionamento geral de um programa

- **Dados** → informações vindas de usuários ou de outras máquinas;
- **Processamento** → transformação dos dados, de acordo com os desejos do usuário ou de outra máquina;
- **Resultados** → aquilo que vem do processamento, e que servirá aos propósitos do usuário.



Etapas de desenvolvimento de um programa

- **Análise:**
  - estuda-se o enunciado do problema para definir os dados de entrada, processamento e dados de saída.

Etapas de desenvolvimento de um programa

- **Análise:**
  - estuda-se o enunciado do problema para definir os dados de entrada, processamento e dados de saída.
- **Algoritmo:**
  - utiliza-se ferramentas do tipo descrição narrativa, fluxograma ou português estruturado para descrever COMO resolver o problema identificado.

Etapas de desenvolvimento de um programa

- **Análise:**
  - estuda-se o enunciado do problema para definir os dados de entrada, processamento e dados de saída.
- **Algoritmo:**
  - utiliza-se ferramentas do tipo descrição narrativa, fluxograma ou português estruturado para descrever COMO resolver o problema identificado.
- **Codificação:**
  - transforma-se o algoritmo em códigos na linguagem de programação escolhida.

Análise

Definição:

Especificar de forma clara e precisa os dados de entrada e os dados de saída (resultados) do problema.

Metodologia:

- A especificação dos dados de entrada e saída deve responder às seguintes questões:
  - Quais são os dados de entrada?
  - Quais são os seus valores válidos e inválidos?
  - Quais valores serão produzidos?
  - Qual o formato dos resultados?

Algoritmo

Definição:

É uma sequência de instruções finita e ordenada de forma lógica para a resolução de uma determinada tarefa ou problema.

Importante!

Para a grande maioria dos problemas, é possível haver mais de um algoritmo de resolução.

Os algoritmos não são exclusivos para os computadores, podem ser aplicados a qualquer problema cuja solução possa ser decomposta em um grupo de instruções.

Exemplos de algoritmos:

- Instruções para se utilizar um aparelho eletrodoméstico;
- Uma receita para preparo de algum prato;
- Guia de preenchimento para declaração do imposto de renda;
- Fazer um sanduíche;
- Trocar uma lâmpada;
- Sacar dinheiro em um banco 24 horas.

Vamos a um exemplo...

Ex-lamp01.: Como seria um algoritmo para trocar uma lâmpada?

Vamos a um exemplo...

Ex-lamp01.: Como seria um algoritmo para trocar uma lâmpada?

```
1 Pegar uma escada;
2 Posicionar a escada embaixo da lâmpada;
3 Buscar uma lâmpada nova;
4 Subir na escada;
5 Retirar a lâmpada velha;
6 Colocar a lâmpada nova.
7
```

- Processamento sequencial
  - Uma instrução é executada depois da outra

Vamos a um exemplo...

Ex-lamp01.: Como seria um algoritmo para trocar uma lâmpada?

```
1 Pegar uma escada;
2 Posicionar a escada embaixo da lâmpada;
3 Buscar uma lâmpada nova;
4 Subir na escada;
5 Retirar a lâmpada velha;
6 Colocar a lâmpada nova.
7
```

- Processamento sequencial
  - Uma instrução é executada depois da outra

E se a lâmpada velha não estivesse queimada?
Como seria o novo algoritmo?

Ex-lamp02.: Algoritmo caso a lâmpada velha não estivesse queimada...

```
1 Pegar uma escada;
2 Posicionar a escada embaixo da lâmpada;
3 Buscar uma lâmpada nova;
4 Acionar o interruptor;
5 SE a lâmpada velha não acender, então:
6     subir na escada;
7     retirar a lâmpada queimada;
8     colocar a lâmpada nova;
9
```

- Decisão para uma determinada condição
  - Estrutura condicional

Ex-lamp02.: Algoritmo caso a lâmpada velha não estivesse queimada...

```
1 Pegar uma escada;
2 Posicionar a escada embaixo da lâmpada;
3 Buscar uma lâmpada nova;
4 Acionar o interruptor;
5 SE a lâmpada velha não acender, então:
6     subir na escada;
7     retirar a lâmpada queimada;
8     colocar a lâmpada nova;
9
```

- Decisão para uma determinada condição
  - Estrutura condicional

Como melhorar o algoritmo acima?

Ex-lamp03.: Algoritmo anterior melhorado...

```
1 Acionar o interruptor;
2 SE a lâmpada não acender, então:
3     pegar uma escada;
4     posicionar a escada embaixo de uma lâmpada;
5     buscar uma lâmpada nova;
6     subir na escada;
7     retirar a lâmpada queimada;
8     colocar a lâmpada nova;
9
```

Ex-lamp03.: Algoritmo anterior melhorado...

```
1 Acionar o interruptor;
2 SE a lâmpada não acender, então:
3     pegar uma escada;
4     posicionar a escada embaixo de uma lâmpada;
5     buscar uma lâmpada nova;
6     subir na escada;
7     retirar a lâmpada queimada;
8     colocar a lâmpada nova;
9
```

E se a lâmpada nova **não** *funcionar?*

Ex-lamp04.: Possível solução caso a lâmpada nova não funcione

```
1  Acionar o interruptor;
2  SE a lâmpada não acender, então:
3      pegar uma escada;
4      posicionar a escada embaixo de uma lâmpada;
5      buscar uma lâmpada nova;
6      subir na escada;
7      retirar a lâmpada queimada;
8      colocar a lâmpada nova;
9      SE a lâmpada não acender, então:
10         retirar a lâmpada queimada;
11         colocar outra lâmpada nova;
12         SE a lâmpada não acender, então:
13             retirar a lâmpada queimada;
14             colocar outra lâmpada nova;
15                         .
16                         .    Até quando???
17                         .
18
```

Ex-lamp05.: Solução melhorada para caso a lâmpada nova não funcione

```
1  Acionar o interruptor;
2  SE a lâmpada não acender, então:
3      pegar uma escada;
4      posicionar a escada embaixo de uma lâmpada;
5      buscar uma lâmpada nova;
6      subir na escada;
7      retirar a lâmpada queimada;
8      colocar a lâmpada nova;
9      ENQUANTO a lâmpada não acender, faça:
10         retirar a lâmpada queimada;
11         colocar outra lâmpada nova;
12
```

Ex-lamp05.: Solução melhorada para caso a lâmpada nova não funcione

```
1  Acionar o interruptor;
2  SE a lâmpada não acender, então:
3      pegar uma escada;
4      posicionar a escada embaixo de uma lâmpada;
5      buscar uma lâmpada nova;
6      subir na escada;
7      retirar a lâmpada queimada;
8      colocar a lâmpada nova;
9      ENQUANTO a lâmpada não acender, faça:
10         retirar a lâmpada queimada;
11         colocar outra lâmpada nova;
12
```

- Evitar repetição de um determinado trecho de código
  - Estrutura de repetição (determinada ou indeterminada)

Algoritmos

Características dos Algoritmos

Todo algoritmo deve apresentar algumas características:

1. ter fim;
2. não dar margem à dupla interpretação (não ambíguo);
3. capacidade de receber dado(s) de entrada do mundo exterior;
4. poder gerar informações de saída para o mundo externo ao do ambiente do algoritmo;
5. ser efetivo (todas as etapas especificadas no algoritmo devem ser alcançáveis em um tempo finito)

Algoritmos

Representação dos Algoritmos

Existem três formas gerais para representar algoritmos:

1. Descritiva Narrativa;
2. Fluxograma;
3. Pseudocódigo (Linguagem Estruturada ou Portugol).

Algoritmos → Representação dos Algoritmos

Descrição Narrativa

Definição:

Dado um problema, consiste em escrever em linguagem natural, os passos a serem seguidos para sua resolução (receita de bolo).

- Vantagem
  - Não é necessário aprender novos conceitos, pois a língua natural já é bem conhecida.
- Desvantagem
  - A língua natural abre espaço para várias interpretações, dificultando a transcrição desse algoritmo para um linguagem de programação.

Exemplo

Bolo de Chocolate

- Aqueça o forno a 180 C
- Unte uma forma redonda
- Numa taça
  - Bata
    - 75g de manteiga
    - 250g de açúcar
  - até ficar cremoso
  - Junte
    - 4 ovos, um a um
    - 100g de chocolate derretido
  - Adicione aos poucos 250g de farinha peneirada
- Deite a massa na forma
- Leve ao forno durante 40 minutos

Exercícios

1. Faça a descrição narrativa de um algoritmo que receba três notas e seus respectivos pesos, calcule e mostre a média entre essas notas.

2. Faça a descrição narrativa de um algoritmo que receba o salário de um funcionário, calcule e mostre o novo salário, sabendo-se que este sofreu um aumento de 25%

Algoritmos → Representação dos Algoritmos

Fluxogramas

Definição:

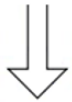
Representação gráfica de algoritmos, onde formas geométricas diferentes implicam ações (instruções, comandos) distintas.

- Vantagem
  - Entendimento de símbolos gráficos é mais fácil que entendimento de textos.
- Desvantagens
  - Necessário aprender a simbologia;
  - Em alguns casos, o algoritmo resultante não apresenta muitos detalhes, dificultando sua transcrição para um programa;
  - Complica-se à medida que o algoritmo cresce

Algoritmos → Representação dos Algoritmos → Fluxogramas

Principais formas geométricas

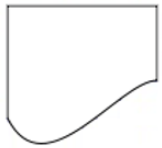
Início e Fim do algoritmo



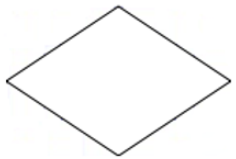
Sentido do fluxo de dados. Conecta símbolos ou blocos existentes

Processos em geral (cálculos ou atribuições de valores)

Entrada de dados

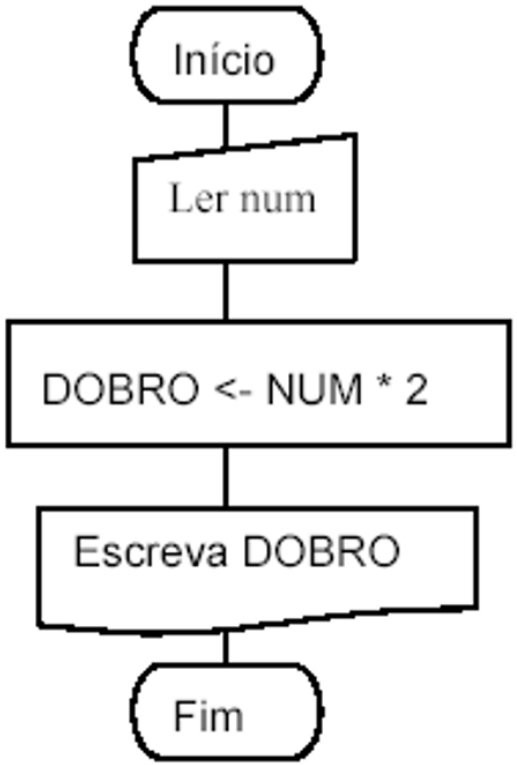


Saída de dados



Tomada de decisão, indicando a possibilidade de desvios

Algoritmos → Representação dos Algoritmos → Fluxogramas

EXEMPLO	EXPLICAÇÃO
	Início do algoritmo
	Entrada do número
	Cálculo do dobro do número
	Apresentação do resultado
	Fim do algoritmo

Exercícios

1. Faça o fluxograma de um algoritmo que receba três notas e seus respectivos pesos, calcule e mostre a média entre essas notas.

2. Faça o fluxograma de um algoritmo que receba o salário de um funcionário, calcule e mostre o novo salário, sabendo-se que este sofreu um aumento de 25%

Algoritmos → Representação dos Algoritmos

Pseudocódigo

Definição:

Pseudocódigo é uma forma genérica de escrever um algoritmo por meio de regras predefinidas, utilizando uma linguagem simples (nativa a quem o escreve) sem necessidade de conhecer a sintaxe de nenhuma linguagem de programação.

Assemelha-se bastante à forma em que os programas são escritos → **bastante aceito**.

Algoritmos → Representação dos Algoritmos

Pseudocódigo

- Vantagens
  - Forma de representação de algoritmos rica em detalhes;
  - A passagem para o código em linguagem de programação é quase imediata.
- Desvantagem
  - Exige o aprendizado das regras do pseudocódigo.

Estrutura básica de um pseudocódigo

```
algoritmo "Nome_do_Algoritmo"
var
    declaração de variáveis
inicio
    Comando-1
    Comando-2
    :
    Comando-N
fimalgoritmo
```

Algoritmos → Representação dos Algoritmos → Pseudocódigo

Exemplo 1

Faça um algoritmo para mostrar o resultado da **multiplicação de dois números**.

Exemplo 1

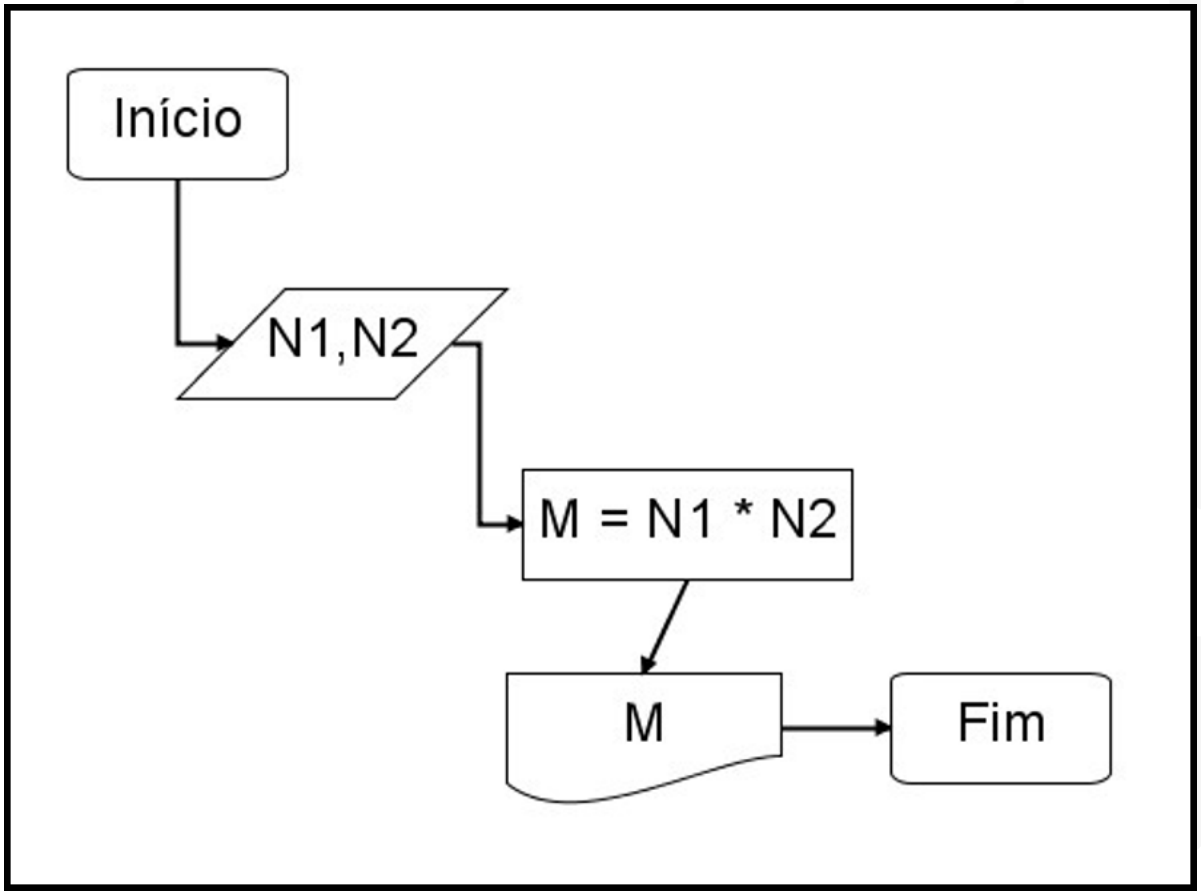
Algoritmo em descrição narrativa:

1. Passo1: Receber os dois números que serão multiplicados
2. Passo2: Multiplicar os números
3. Passo3: Mostrar o resultado obtido na multiplicação

Algoritmos → Representação dos Algoritmos → Pseudocódigo

Exemplo 1

Algoritmo em fluxograma:



Exemplo 1

Algoritmo em fluxograma:

```
algoritmo "Multiplicação"

var
        n1,n2,m: inteiro
inicio
    escreva("Digite dois números:")
    leia(n1)
    leia(n2)
    m ← n1 * n2
    escreva("Multiplicação = ", m)
fimalgoritmo
```

Algoritmos → Representação dos Algoritmos → Pseudocódigo

Exemplo 2

Faça um algoritmo para mostrar o resultado da **divisão de dois números**.

Exemplo 2

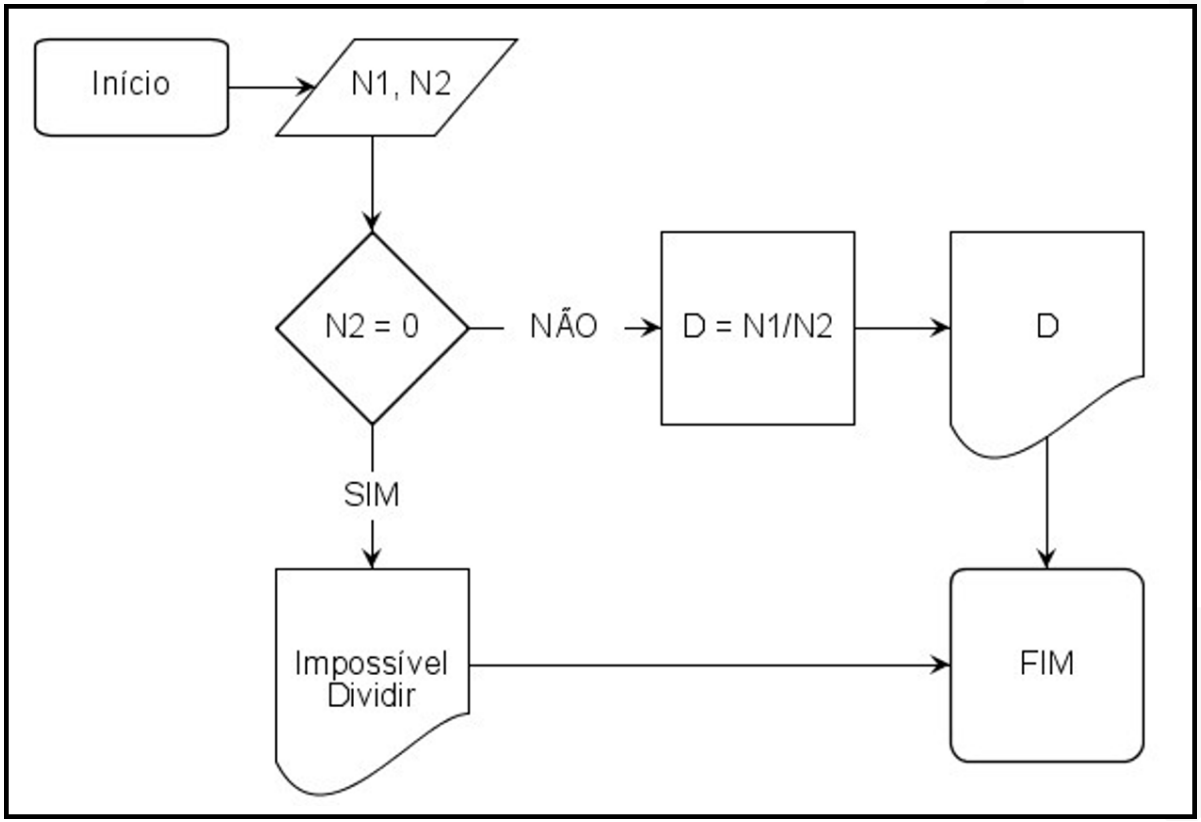
Algoritmo em descrição narrativa:

1. Passo1: Receber os dois números que serão divididos
2. Passo2: Se o segundo número for igual a zero, não poderá haver divisão, pois não existe divisão por zero;
3. Passo3: Dividir os números
4. Passo4: Mostrar o resultado obtido na divisão

Algoritmos → Representação dos Algoritmos → Pseudocódigo

Exemplo 2

Algoritmo em fluxograma:



Algoritmos → Representação dos Algoritmos → Pseudocódigo

Exemplo 2

Algoritmo em fluxograma:

```
algoritmo "Divisão"

var
        n1, n2, D: real
inicio
        escreva( "Digite dois números:")
        leia(n1)
        leia(n2)
        se n2 = 0 então
           escreva("Impossível dividir.")
        senao
              D ← n1/n2
              escreva("Divisão = ",D)
        fimse
  fimalgoritmo
```

Exercícios

1. Faça o pseudocódigo de um algoritmo que receba três notas e seus respectivos pesos, calcule e mostre a média entre essas notas.

2. Faça o pseudocódigo de um algoritmo que receba o salário de um funcionário, calcule e mostre o novo salário, sabendo-se que este sofreu um aumento de 25%

Algoritmos

Teste de Mesa

Após desenvolver um algoritmo é preciso testá-lo. Uma maneira de se fazer isso 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

Algoritmos

Teste de Mesa

- Criar uma tabela de modo que
  - Cada coluna representa uma valor;
  - As linhas correspondem as alterações naquela variável (de cima para baixo).

valor	N	soma

Algoritmos → Teste de Mesa

Exemplo

Imprimir a média dos números positivos digitados. Parar quando um valor negativo ou zero por 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

Codificação

Definição

A etapa de codificação traduz a representação do projeto detalhado (algoritmo) em termos de uma linguagem de programação.

O resultado da codificação de um algoritmo é um **programa de computador**.

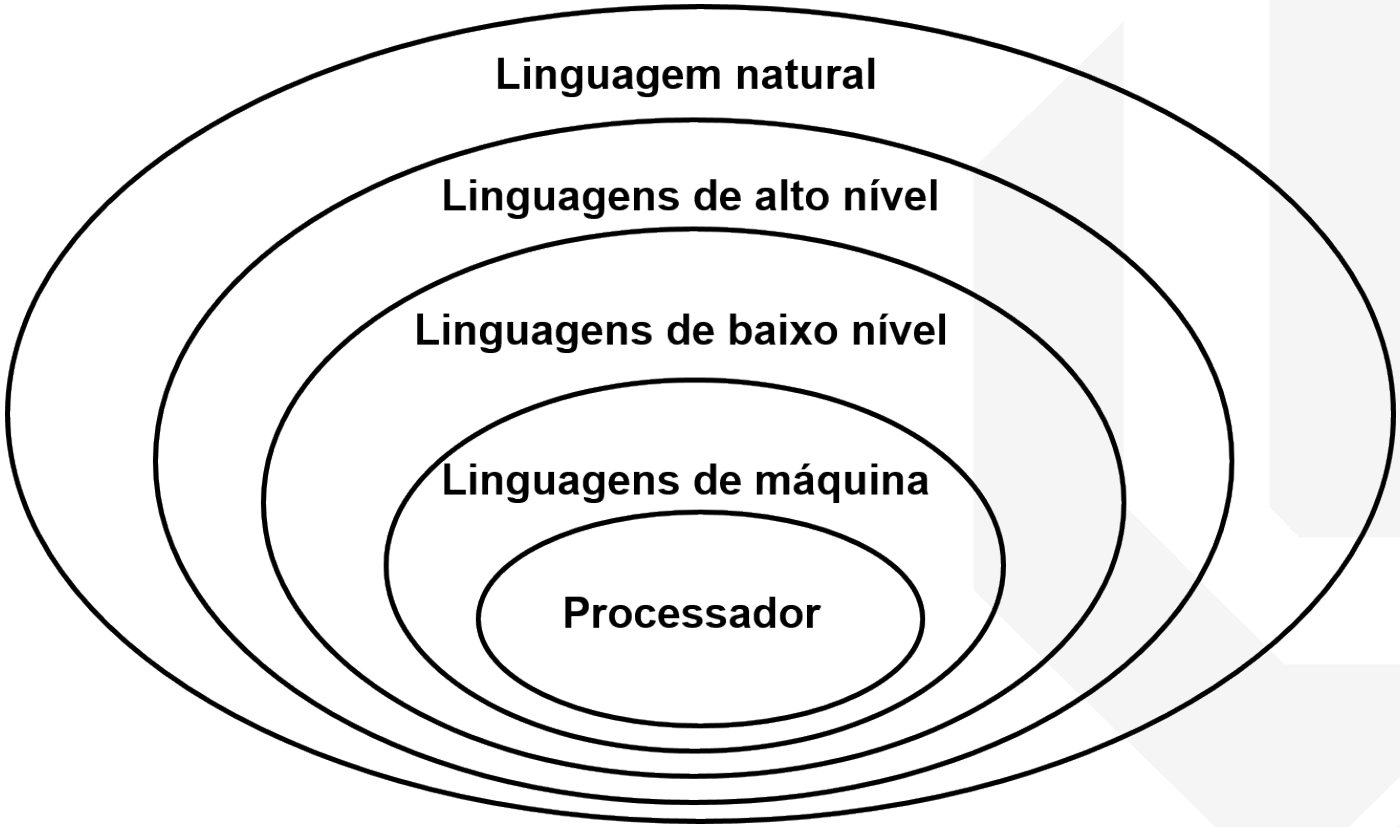
Nesta disciplina, a linguagem C será utilizada nos exemplos e atividades práticas desenvolvidas.

Programa

Definição:

Um programa é procedimento que indica ao computador, passo a passo, como resolver um determinado problema; mais especificamente, uma sequência inequívoca e ordenada de instruções computacionais necessárias para alcançar tal solução.

Níveis de abstração das linguagens



Linguagem natural

A linguagem natural é a nossa língua em si, i.e., a forma como falamos, escrevemos e nos comunicamos.

Na linguagem natural — tanto escrita, quanto falada — é repleta de regras e ambiguidades, que dependem inclusive do idioma.

Linguagem de alto nível

- A linguagem de alto nível está muito mais próxima do programador do que da máquina, se assemelhando mais com a linguagem humana.
- Entretanto, estas linguagens correspondem a um nível elevado de abstração
  1. `a = (b + c + d + e)`

Linguagem de baixo nível

- A sintaxe está mais distante da linguagem natural;
- Estruturalmente semelhantes às instruções do processador;

- Adição $a = (b + c + d + e)$ em baixo nível:

```
1 add a, 0      #soma (a+0) e armazena em a
2 add a, b      #soma (a+b) e armazena em a
3 add a, c      #soma (b+c) e armazena em a
4 add a, d      #soma (b+c+d) e armazena em a
5 add a, e      #soma (b+c+d+e) e armazena em a
```

L. Baixo Nível vs. L. Alto Nível

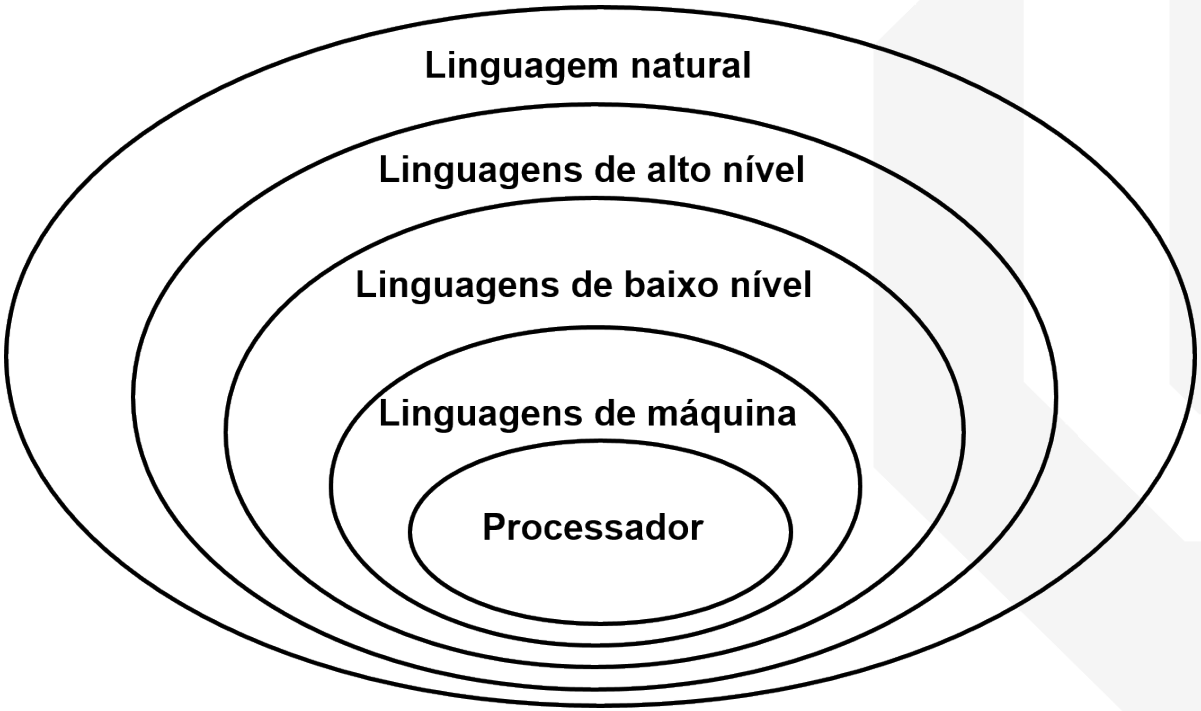
✓ Linguagens de Baixo Nível
- Prós
  1. tempo de processamento mais rápido;
  2. melhor aproveitamento arquitetura do computador.
- Contras
  1. maior tempo para compreender e dominar a sintaxe;
  2. necessário conhecer detalhes do hardware;
  3. menor produtividade e maior engessamento.

✓ Linguagens de Alto Nível
- Prós
  1. facilidade de aprendizagem;
  2. maior produtividade;
  3. manutenção simplificada, o que reduz custos.
- Contras
  1. ocupa mais memória;
  2. desempenho prejudicado, pois exige mais processamento.

Linguagem de máquina

- Para que a máquina entenda um programa, é necessário que suas instruções estejam codificadas na forma binária, i.e., na forma de 0s e 1s.
- "add" corresponde ao código binário: 000000100000

Se o computador entende apenas linguagem de máquina, i.e., 0s e 1s, como acontece a passagem de linguagens de alto e baixo nível para a linguagem de máquina?



Tradutores

Definição:

Um tradutor, em programação é um termo genérico para se referir a qualquer programa que converta código de nível superior em outro código de alto ou baixo nível.

- Basicamente, temos três tipos de tradutores: compiladores, interpretadores e montadores/assemblers.
- O processo de tradução de uma linguagem de alto nível para uma linguagem de baixo nível é feito por compiladores ou interpretadores.
- O processo de tradução de uma linguagem de montagem para uma linguagem de máquina é feito por montadores/assemblers.

O processo de tradução

Programa em linguagem de alto nível (em C)

```
swap(int v[], int k) {
    int temp;
    temp = v[k];
    v[k] = v[k+1];
    v[k+1] = temp;
}
```

compilador

Programa em linguagem de montagem (Assembly)

```
swap:
        muli $2, $5, 4
        add $2, $4, $2
        lw $15, 0($2)
        lw $16, 4($2)
        sw $16, 0($2)
        sw $15, 4($2)
        jr $31
```

montador

Programa em linguagem de máquina

```
00000000101000010000000000011000
00000000000011000000110000100001
10001100011000100000000000000000
10001100111100100000000000000100
10101100111100100000000000000000
10101100011000100000000000000100
00000011111000000000000000001000
```

Compiladores

Definição:

Um compilador é um programa de computador que traduz um programa descrito em uma linguagem de alto nível para um programa equivalente em linguagem de montagem.

- Exemplos de linguagens compiladas:
  - C/C++
  - Pascal
  - Fortran
  - Cobol
  - Delphi

Os passos da compilação

1. recebe primeira instrução
2. confere se está correta
3. converte para linguagem de montagem
4. passa para próxima instrução

Obs.: - O programa é executado somente depois de ter todo o código convertido para a linguagem de máquina;

- Progs. compilados são mais rápidos, mas, gastam mais memória;

- Se o mesmo programa for executado novamente, não haverá necessidade de repetir o processo de tradução, pois todas as conversões são armazenadas.

Interpretadores

Definição:

Um interpretador é um programa que lê um código-fonte de uma linguagem de programação interpretada e o converte em código executável, i.e., a execução é realizada passo-a-passo.

- Exemplos de linguagens interpretadas:
  - Java
  - Python
  - PHP
  - Ruby
  - C#

Os passos da interpretação

1. recebe primeira instrução
2. confere se está correta
3. converte para linguagem de máquina
4. **executa a instrução**
5. passa para próxima instrução

Obs.: - Quando uma instrução passa pelo ciclo de interpretação, a instrução anterior é perdida;

- Apenas uma instrução fica na memória a cada instante;

- Se o programa for executado novamente, todo o processo será refeito, pois, as conversões para a linguagem de máquina não são armazenadas.

Montadores

Definição:

Um montador traduz um programa escrito em linguagem Assembly para linguagem de máquina.

- São os tipos de tradutores mais simples;
- Convertem os mnemônicos em instruções binárias;
- Pode ser descrito efetivamente um compilador para a linguagem Assembly, mas também pode ser usado interativamente como um interpretador.

Referências

✓ Básica

- **DAMAS, Luís. "Linguagem C". Grupo Gen-LTC, 2016.**
- **MIZRAHI, Victorine V. "Treinamento em linguagem C", 2a. ed., São Paulo, Pearson, 2008.**

✓ Extra

- BACKES, André. *"Programação Descomplicada Linguagem C"*. Projeto de extensão que disponibiliza vídeo-aulas de C e Estruturas de Dados. Disponível em: `https://www.youtube.com/user/progdescomplicada`. Acessado em: 25/04/2022.

✓ Baseado nos materiais dos seguintes professores:

- Prof. André Backes (UFU)
- Prof. Guilherme Tavares de Assis (UFOP)
- Prof. Jean Roberto Ponciano (UFU)
- Prof. Rachel Reis (UFV)
- Prof. Renato Pimentel (UFU)

Dúvidas?

Prof. Me. Claudiney R. Tinoco
profclaudineytinoco@gmail.com

Faculdade de Computação (FACOM)
Universidade Federal de Uberlândia (UFU)