Introdução a algoritmos

Rafael Beserra Gomes

UFRN

Material compilado em 13 de dezembro de 2017.

Conceitos iniciais de algoritmos

Objetivo

Sequência ordenada e não ambígua de passos que levam à solução de um dado problema [TREMBLAY, 1979].

Conceitos iniciais de algoritmos

Exemplos:

- Como ir da sala até o Natal Shopping?
- Tarefas do studio.code.org

- Quais são os passos possíveis no algoritmo?
- Existe um único algoritmo para finalizar cada etapa?

Algoritmos computacionais

Algoritmos computacionais

Algoritmos computacionais

Algoritmo computacional

Um algoritmo que pode ser traduzido em uma sequência de instruções que pode ser executado em um computador.

Algoritmos computacionais

- Os algoritmos serão expressos de acordo com o que o computador pode executar
- Vamos conhecer um pouco mais sobre o computador!

Sistemas computacionais

Introdução

- Adquire dados
- Armazena dados
- Processa dados
- Exibe dados

Introdução

Adquire dados ← Obtemos informações do mundo através de **dispositivos de entrada**.

Armazena dados

Processa dados

Exibe dados

Introdução

Adquire dados

Armazena dados

Processa dados

Exibe dados

Obtemos informações do mundo
através de **dispositivos de entrada**:
- Teclado (texto)
- Câmera (imagens)
- Scanner (imagens)
- Leitor de digitais (imagens)
- Microfone (som)
- Mouse (coordenada, botões)
- Eletrocardiógrafo (atividade elétrica no coração)
- Touch Screen

Introdução

Adquire dados

Armazena dados ⟵ Através das memórias é possível armazenar dados.

Processa dados

Exibe dados

Introdução

Adquire dados

Armazena dados

Processa dados

Exibe dados

Através das memórias é possível armazenar dados:

Memória volátil
Os dados são perdidos quando a energia cessa
- Mem. RAM, mem. cache, registradores (memória principal)

Memória não volátil
- HD, Disquete, CD, DVD, Pen-Driver (memória secundária)

Introdução

Adquire dados

Armazena dados

Processa dados

Exibe dados

O processamento dos dados é usualmente realizado por um ou mais processadores (CPU).
O processador contém registradores e efetuam nestes operações básicas como:
adição, subtração, multiplicação.

Introdução

Adquire dados

Armazena dados

Processa dados

Exibe dados

O computador utiliza **dispositivos de saída** para que esses dados sejam compreensíveis para o ser humano.

Introdução

Adquire dados

Armazena dados

Processa dados

Exibe dados

O computador utiliza **dispositivos de saída** para que esses dados sejam compreensíveis para o ser humano:
- Impressora
- Monitor
- Alto-falante

Sistema operacional

Sistema Operacional

O sistema operacional (SO) é um conjunto de softwares que controla os recursos do computador e oferece serviços básicos para qualquer aplicativo. Entre os sistemas operacionais mais conhecidos estão:

- Windows
- Linux (diversas distribuições como Ubuntu)
- Android (baseado em Linux)
- Mac OS

Sistema operacional

Programas

Programas (software) são sequência de instruções que podem ser executadas em um processador.

Exemplos:

Processador de texto	Microsoft Word, OpenOffice Writer, Latex
Editor de imagens	Adobe Photoshop, Gimp
Editor de vídeos	Adobe After Effects, Sony Vegas
Científicos	Matlab, Geogebra
Jogos	Stunts, Sim City, Super Mario, Street Fighter
Animação	3D Studio, Maya, Blender, Adobe Flash
Navegadores	Internet Explorer, Mozilla Firefox, Opera, Google Chrome
Programas mais simples	cat, calculadora, grep, head, ipconfig (ifconfig)

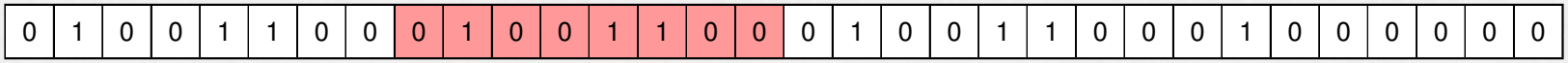
Você é capaz de responder com que tipo de informações cada programa desse trabalha?

Memória

Como essas informações são representadas em um computador?

Memória

Os dados na memória são armazenados em sequências de bits



- 8 bits = 1 byte
- 1024 bytes = 1 kilobyte (kB)
- 1024 kilobytes = 1 megabyte (MB)
- 1024 megabytes = 1 gigabyte (GB)

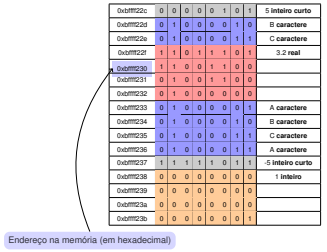
Memória

Exemplo de dados na memória:

0xbffff22c	0	0	0	0	0	1	0	1	5 **inteiro curto**
0xbffff22d	0	1	0	0	0	0	1	0	B **caractere**
0xbffff22e	0	1	0	0	0	0	1	1	C **caractere**
0xbffff22f	1	1	0	1	1	1	0	1	3.2 **real**
0xbffff230	1	1	0	0	1	1	0	0	
0xbffff231	0	1	0	0	1	1	0	0	
0xbffff232	0	1	0	0	0	0	0	0	
0xbffff233	0	1	0	0	0	0	0	1	A **caractere**
0xbffff234	0	1	0	0	0	0	1	0	B **caractere**
0xbffff235	0	1	0	0	0	0	1	1	C **caractere**
0xbffff236	0	1	0	0	0	0	1	1	A **caractere**
0xbffff237	1	1	1	1	1	0	1	1	-5 **inteiro curto**
0xbffff238	0	0	0	0	0	0	0	0	1 **inteiro**
0xbffff239	0	0	0	0	0	0	0	0	
0xbffff23a	0	0	0	0	0	0	0	0	
0xbffff23b	0	0	0	0	0	0	0	1	

Endereço na memória (em hexadecimal)

Exemplo de dados na memória:



1. A contagem dos números em hexadecimal é: 0, 1, 2, 3, 4, 5, 6, 7, 8, 9, A, B, C, D, E, F. O uso da base 16 é conveniente, pois reduz a quantidade de dígitos para representar a mesma informação em binário e a conversão de e para binário é simplificada. Os números em hexadecimal são geralmente precedidos por 0x. Por exemplo, para converter A3B para binário basta concatenar a conversão individual de cada dígito ($A_{16} = 1100_2$, $3_{16} = 0011_2$ e $B = 1101_2$, portanto $A3B_{16} = 110000111101_2$).
2. Nesse slide, assim como no restante da disciplina, as informações estão representadas em big-endian. Nessa representação, a ordem de armazenamento dos bytes é do mais significativo para o menos significativo, apesar de little-endian ser o mais usual (byte menos significativo primeiro).

Memória

Um tipo de dado corresponde a uma sequência de bits de tamanho fixo com uma interpretação específica.

número inteiro = 4 bytes	utiliza complemento de 2 para englobar números negativos
número real = 4 bytes	padrão IEEE 754
um caractere ASCII = 1 byte	por exemplo: A = 65 (01000001)
um caractere Unicode = 4 bytes	cerca de 107 mil caracteres

Um tipo de dado corresponde a uma sequência de bits de tamanho fixo com uma interpretação específica.

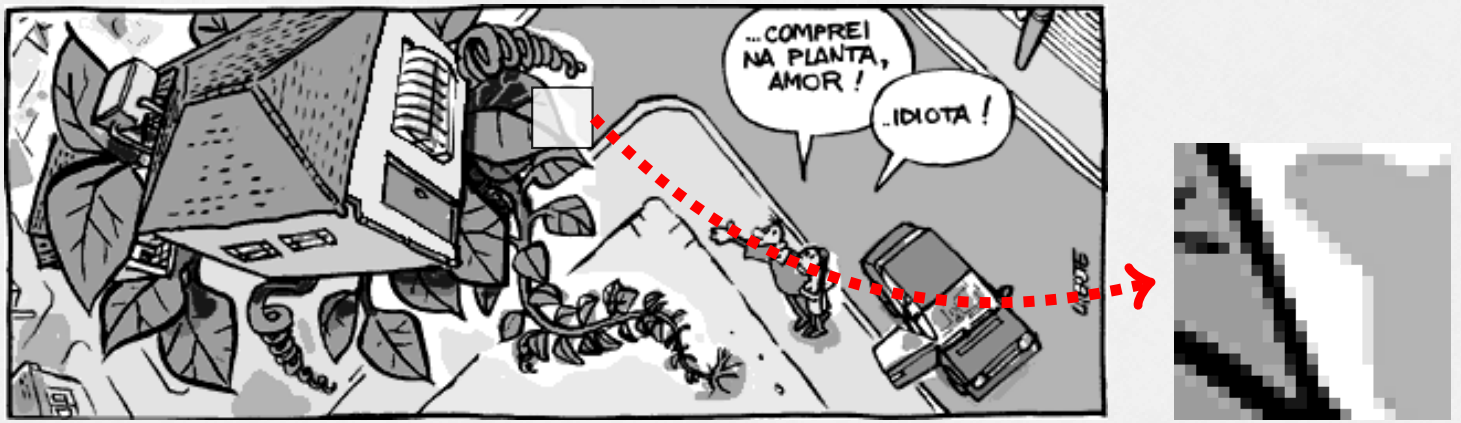
- número inteiro = 4 bytes — utiliza complemento de 2 para englobar números negativos
- número real = 4 bytes — padrão IEEE 754
- um caractere ASCII = 1 byte — por exemplo: A = 65 (01000001)
- um caractere Unicode = 4 bytes — cerca de 107 mil caracteres

1. No Linux você pode conferir a tabela ASCII digitando **man ascii** (q para sair do manual). Sempre que houver alguma dúvida sobre algum comando do Linux, você pode tentar consultar o manual. Por exemplo, para ver o manual do ls: **man ls**

Memória

Imagem

Uma imagem digital é composta por pixels.

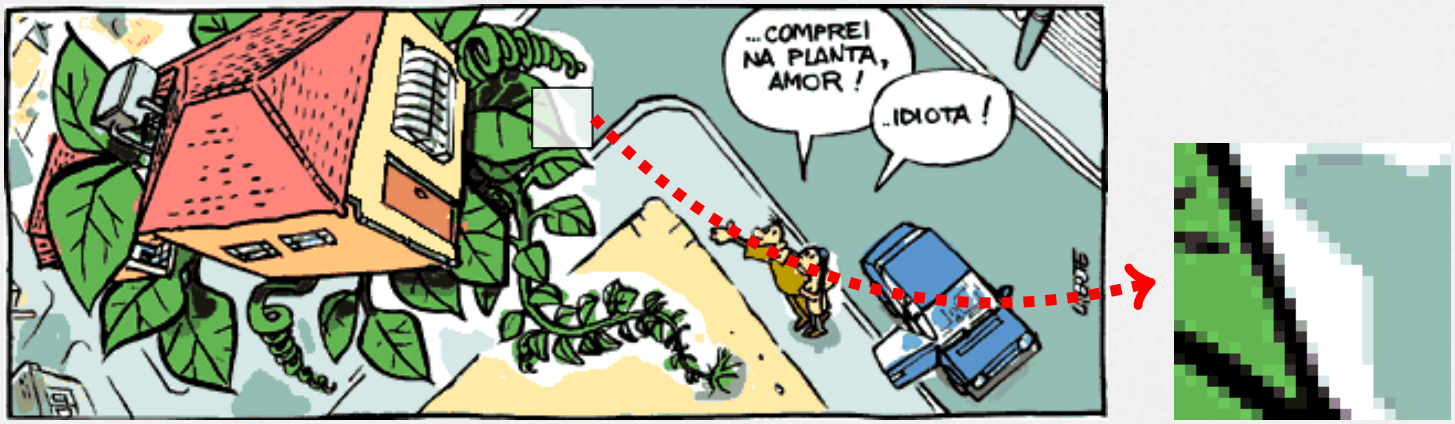


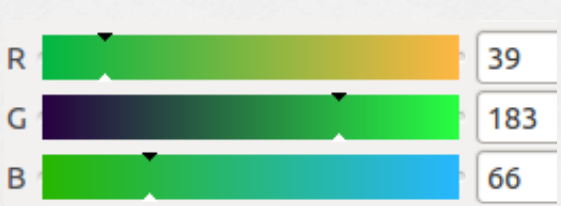
Preto (valor 0) Cinza (valor 127) Branco (valor 255)

Memória

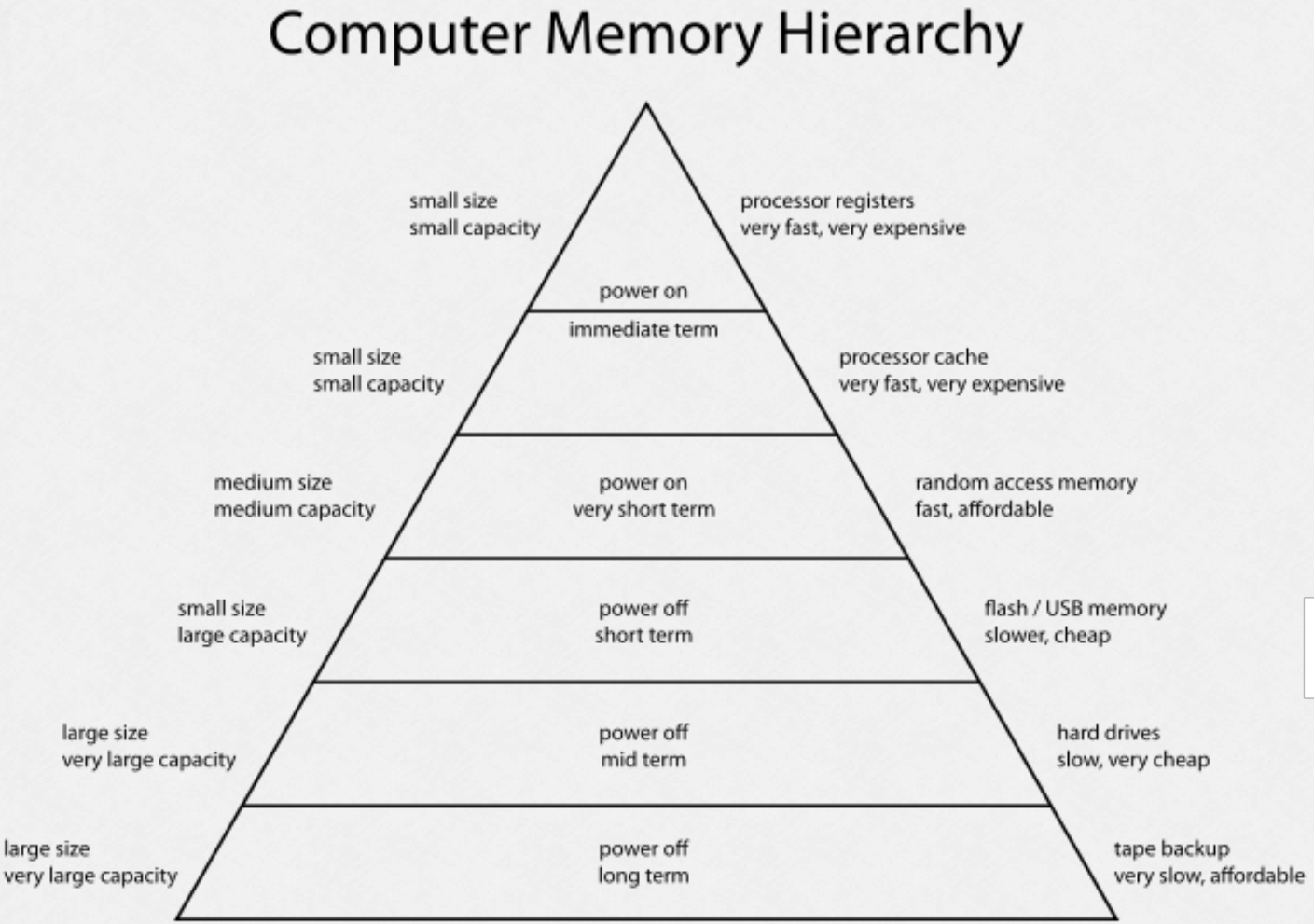
Imagem colorida

Em uma imagem colorida, cada pixel pode ser descrito como a composição de 3 canais: R (red) , G (green), B (blue)

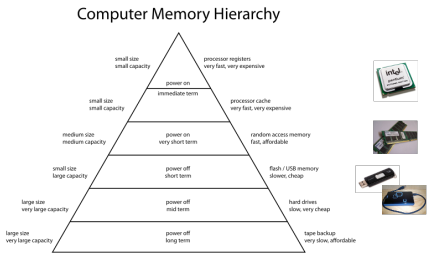




Memória


Computer Memory Hierarchy
small size
small capacity
processor registers
very fast, very expensive
power on
immediate term
small size
small capacity
processor cache
very fast, very expensive
medium size
medium capacity
power on
very short term
random access memory
fast, affordable
small size
large capacity
power off
short term
flash / USB memory
slower, cheap
large size
very large capacity
power off
mid term
hard drives
slow, very cheap
large size
very large capacity
power off
long term
tape backup
very slow, affordable


intel
pentium



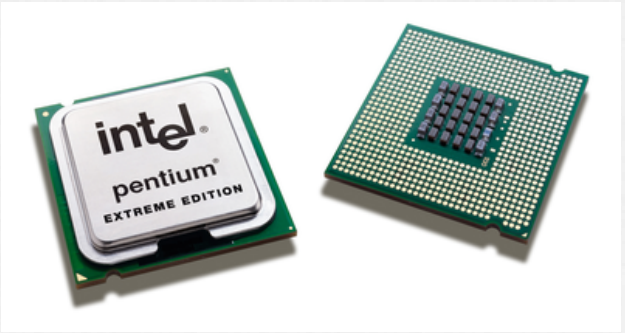
1. Note que quanto mais rápida a memória, mais cara. Por esse motivo, as memórias mais rápidas também possuem menor capacidade de armazenamento.

CPU

A CPU (unidade central de processamento) é um dos responsáveis pelo processamento dos dados em um computador.

A cada ciclo de clock uma instrução é executada na CPU.

O conjunto de instruções disponível depende do modelo de processador.

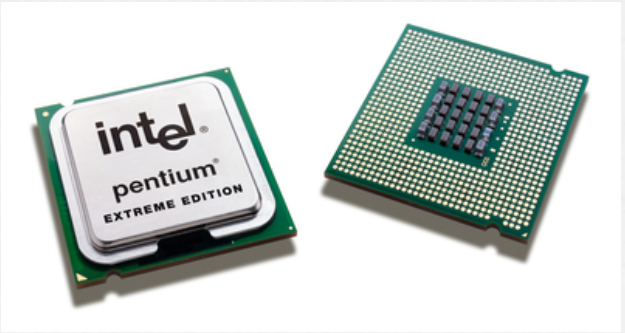


CPU

A CPU (unidade central de processamento) é um dos responsáveis pelo processamento dos dados em um computador.

A cada ciclo de clock uma instrução é executada na CPU.

O conjunto de instruções disponível depende do modelo de processador.



CPU

A CPU (unidade central de processamento) é um dos responsáveis pelo processamento dos dados em um computador.

A cada ciclo de clock uma instrução é executada na CPU.

O conjunto de instruções disponível depende do modelo de processador.

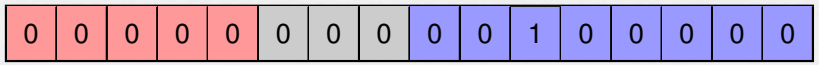
Instruções básicas como:
- somar valores de dois endereços
- multiplicar dois números
- obter um valor da memória

CPU

Quais são as instruções disponíveis para criar algoritmos no computador?

Linguagem/código de máquina

- Específica para cada arquitetura de computador
- Um exemplo fictício (processador de 16 bits):



  - 5 bits: instrução
  - 3 bits: registrador
  - 8 bits: valor
- Por exemplo: 10110 é o código para colocar um valor em um determinado registrador. Os 3 seguintes bits especificam qual registrador. Os demais 8 bits representam o valor a ser armazenado no registrador.

CPU

- Quer saber como funciona o seu processador e o conjunto de instruções? Leia o manual!
  https://software.intel.com/en-us/articles/intel-sdm
- Programar em linguagem de máquina é um trabalho árduo
- A **linguagem assembly** utiliza uma linguagem mais fácil para escrever programas
- Um *assembler* transforma um código em *assembly* para código de máquina

- Quer saber como funciona o seu processador e o conjunto de instruções? Leia o manual! https://software.intel.com/en-us/articles/intel-sdm
- Programar em linguagem de máquina é um trabalho árduo
- A **linguagem assembly** utiliza uma linguagem mais fácil para escrever programas
- Um *assembler* transforma um código em *assembly* para código de máquina

1. Não vamos programar em linguagem de máquina nessa disciplina.
2. A memória principal armazena tanto dados do programa como os próprios programas
3. Geralmente o HD faz o papel de memória secundária. Geralmente os programas estão armazenados nessa memória, mas quando solicitamos sua execução, as instruções são transferidas para a memória principal (memória RAM).

Linguagens de Programação

Conceitos

As linguagens de programação facilitam a programação de computadores.

Conceitos

C++ Pascal Portugol Java C Python Ruby Basic Matlab Lua JavaScript
Perl PHP Lisp C# Fortran Scheme Haskell

Conceitos


C++
Pascal
Portugol
Java
C
Python
Ruby
Basic
Matlab
Lua
JavaScript
Perl
PHP
Lisp
C#
Fortran
Scheme
Haskell
C é uma linguagem compilada

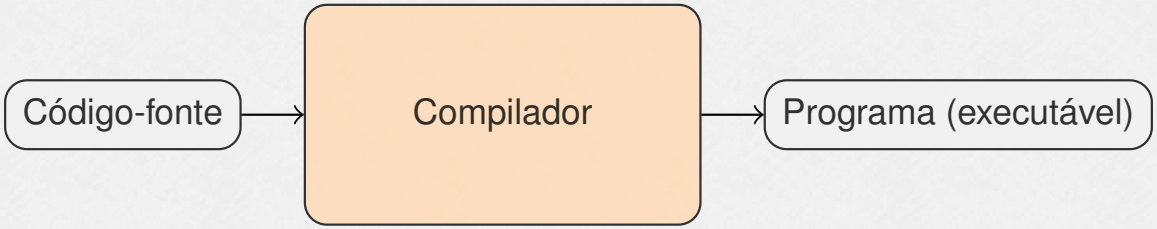
1. Cada linguagem de programação possui suas vantagens e desvantagens e, portanto, a linguagem mais indicada depende do contexto da aplicação.
2. A linguagem C apesar de bastante antiga é ainda uma das mais populares e é excelente para aprender programação.
3. Acesse esse artigo da IEEE para conhecer mais sobre a popularidade de algumas linguagens de programação.

Sintaxe e semântica

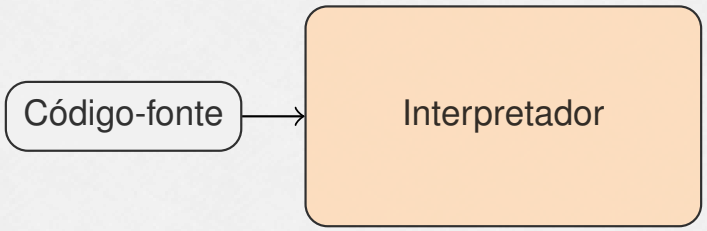
- As linguagens de programação possuem regras de sintaxe e semântica!
- Um algoritmo escrito nessa linguagem de programação só será aceito se estiver completamente de acordo com as regras gramaticais

Linguagens compiladas e interpretadas

Nas **linguagens compiladas**, programas específicos chamados compiladores convertem um código escrito em uma linguagem (código-fonte) para uma sequência de instruções do processador.



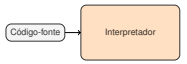
Nas **linguagens interpretadas**, um programa interpretador interpreta um código escrito em uma linguagem e executa a sequência de instruções correspondentes.



Introdução a algoritmos
Linguagens de Programação
Linguagens compiladas e interpretadas

Nas **linguagens compiladas**, programas específicos chamados compiladores convertem um código escrito em uma linguagem (código-fonte) para uma sequência de instruções do processador.

Nas **linguagens interpretadas**, um programa interpretador interpreta um código escrito em uma linguagem e executa a sequência de instruções correspondentes.



1. As linguagens interpretadas tendem a ser mais lentas pois várias instruções são executadas com a finalidade de interpretar o código-fonte durante a execução do programa.
2. As linguagens interpretadas têm a grande vantagem de facilitar a portabilidade entre sistemas diferentes.

Linguagens compiladas e interpretadas

- Faça o seguinte teste para a linguagem C:
  - baixe o código zeroanove.c disponível na página
  - **Visualize o código-fonte:** cat zeroanove.c
  - **Compile o código-fonte:** gcc zeroanove.c -o executavelZeroANove
  - **Execute o código-fonte:** ./executavelZeroANove
  - **Compile parcialmente**[1]**:** gcc zeroanove.c -S
  - **Visualize o código-fonte assembly:** cat zeroanove.s
  - **Visualize o programa em código de máquina:** xxd -b executavelZeroANove | more

[1] a compilação é interrompida na etapa assembler

- Faça o seguinte teste para a linguagem C:
  - baixe o código zeroanove.c disponível na página
  - **Visualize o código-fonte:** cat zeroanove.c
  - **Compile o código-fonte:** gcc zeroanove.c -o executavelZeroANove
  - **Execute o código-fonte:** ./executavelZeroANove
  - **Compile parcialmente**[1]**:** gcc zeroanove.c -S
  - **Visualize o código-fonte assembly:** cat zeroanove.s
  - **Visualize o programa em código de máquina:** xxd -b executavelZeroANove | more

[1] a compilação é interrompida na etapa assembler

1. O comando cat exibe na tela o conteúdo do(s) arquivo(s) especificado(s)
2. O comando xxd exibe na tela o conteúdo do arquivo em hexadecimal ou binário. O | more significa que o conteúdo não será exibido de uma vez caso ocupe mais espaço do que o disponível na tela.

Linguagens compiladas e interpretadas

- Faça o seguinte teste para a linguagem Python:
  - baixe o código zeroanove.py disponível na página
  - **Visualize o código-fonte:** cat zeroanove.py
  - **Interprete o código-fonte:** python zeroanove.py

Linguagens compiladas e interpretadas

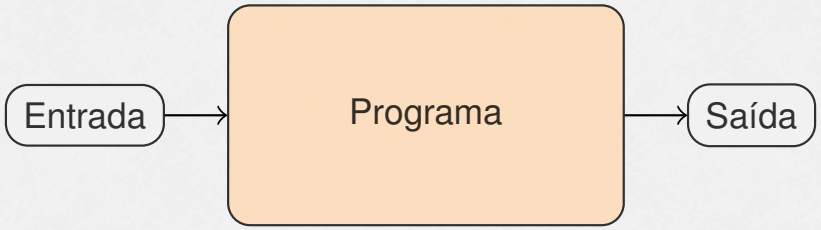
Neste curso utilizaremos como dispositivo de entrada um teclado e como dispositivo de saída o monitor através de um terminal do sistema em modo texto.

Dispositivo de entrada: Teclado Scanner Câmera Mouse

Dispositivo de saída: Monitor (terminal) Impressora Alto-falante

Entrada e saída de um programa

Cada programa a ser desenvolvido nesse curso terá uma **entrada** fornecido pelo usuário através do teclado e uma **saída** visualizável no monitor.



Exemplo:

Um programa que calcula a média de um aluno dadas as notas das 3 unidades:
Entrada: nota 1, nota 2 e nota 3 (números reais)
Saída: média parcial (número real)

Representação dos algoritmos

Formas para representar um algoritmo:

- Narrativa (linguagem natural)
- Fluxograma
- Pseudo-código

Narrativa

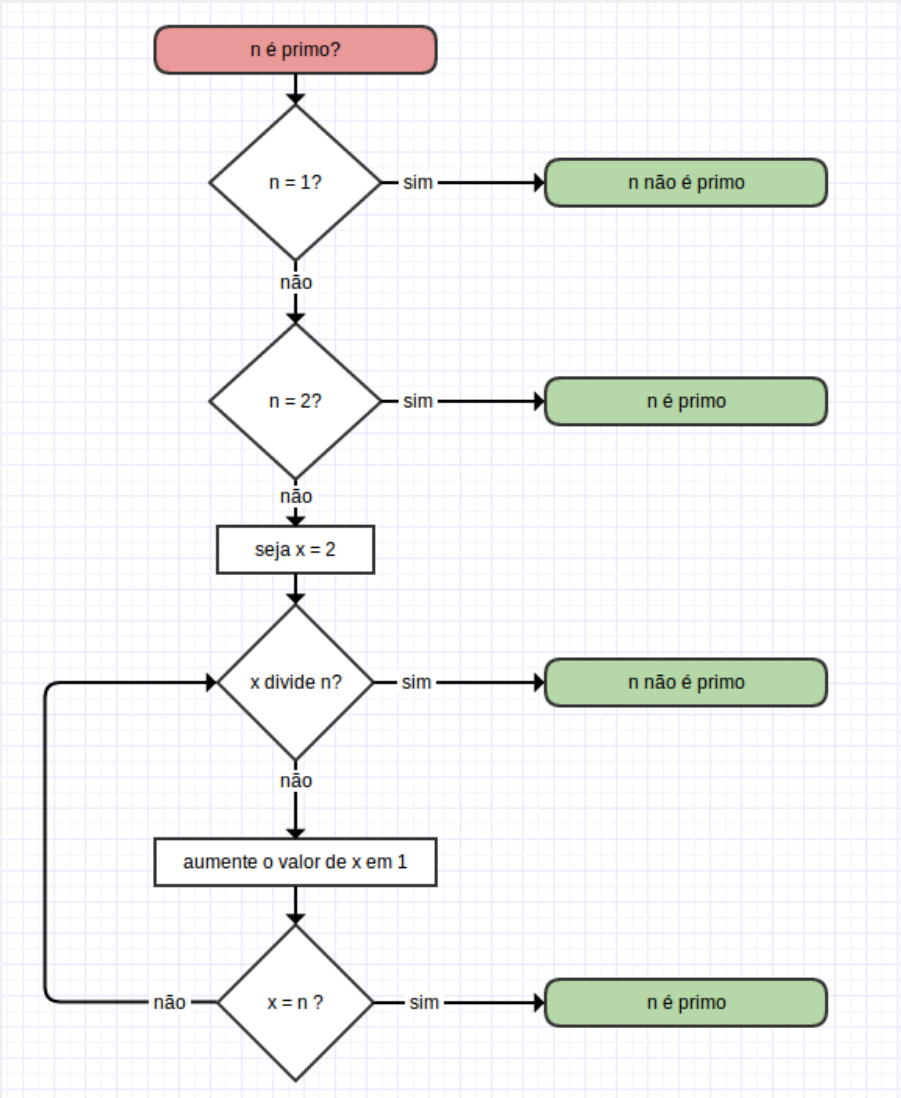
Narrativa:

- Descrição textual em linguagem natural
- Útil para ideias iniciais (lembre-se de que o objetivo é ter um algoritmo escrito em uma linguagem de programação)
- Problema: determine se um número inteiro n maior que 0 é primo
- Solução 1: o número n é primo se ele é diferente de 1 e divisível somente por 1 e por ele próprio
- Solução 2: desconsiderando os casos triviais n = 1 e n = 2, verifique se há algum número entre 2 e n-1 que divide n. Se não houver, então é primo!

Fluxograma

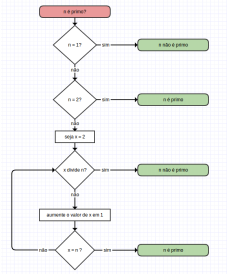
Fluxograma

- Representação gráfica muito boa para visualizar o fluxo lógico



Fluxograma

- Representação gráfica muito boa para visualizar o fluxo lógico



1. A representação em fluxograma não é muito comum, pois ocupa muito espaço e é difícil produzí-la.

Pseudo-código

Pseudo-código

- Representação mais próxima de uma linguagem de programação
- Flexibilidade em relação às regras gramaticais da linguagem e uso de linguagem natural

```
se n for 1 então conclua que n não é primo
se n for 2 então conclua que n é primo
caso contrário para cada um dos números de 2 até n-1 faça o seguinte:
        - verifique se esse número divide o número n
        - se dividir então conclua que n não é primo
conclua que n é primo
```