Estrutura e funcionamento de um compilador

Prof. Me. Hélio Esperidião

linguagens de alto nível

são linguagens com sintaxe mais fácil de ser compreendida pelo homem e independentes do hardware (máquina).

Um programa escrito em uma linguagem de alto nível é denominado código-fonte.

A tradução da linguagem de alto nível para a linguagem de máquina, normalmente, é feita em mais de um passo e o programa que faz essa tradução é denominado compilador

tipos

Os tipos existentes em uma linguagem serão os indicativos para o programador implementar as estruturas para armazenar dados e executar cálculos.

Há tipos simples e outros mais complexos, como listas, arvores, funções e classes

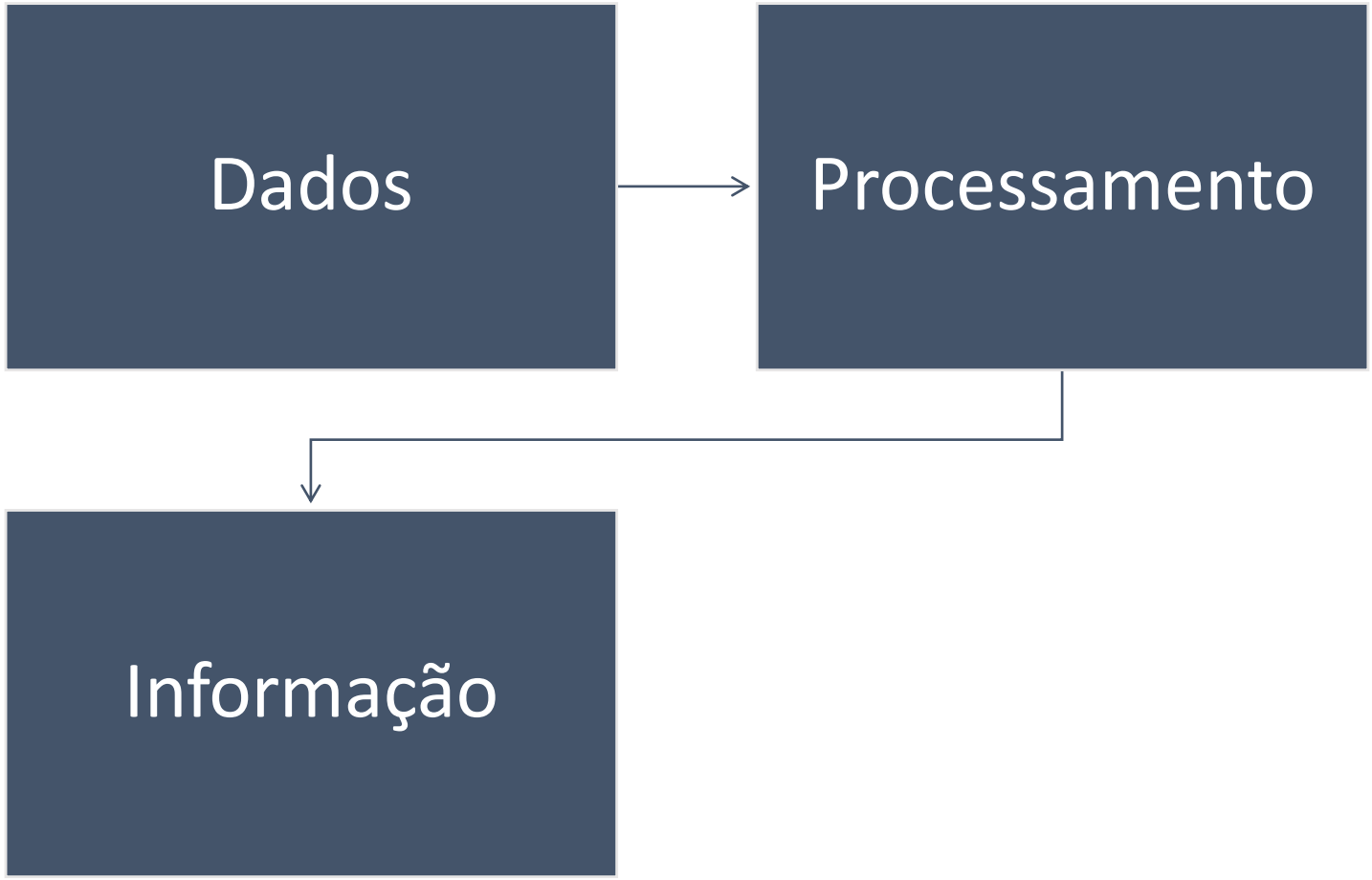
O QUE É UM DADO?

- Dado pode ser definido como a matéria-prima originalmente obtida de uma ou mais fontes (etapa de coleta).

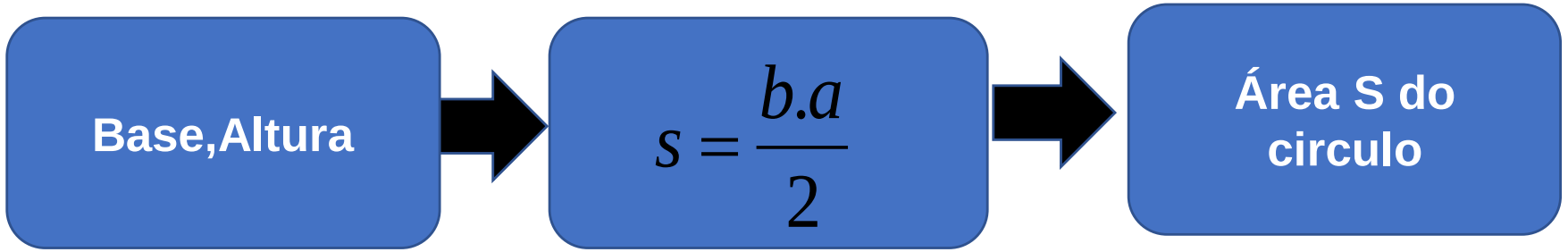
o que é a informação

- A Informação é o resultado do processamento.
- Isto é, o dado processado ou "acabado”.

Obtendo a informação



Exemplo de Processamento

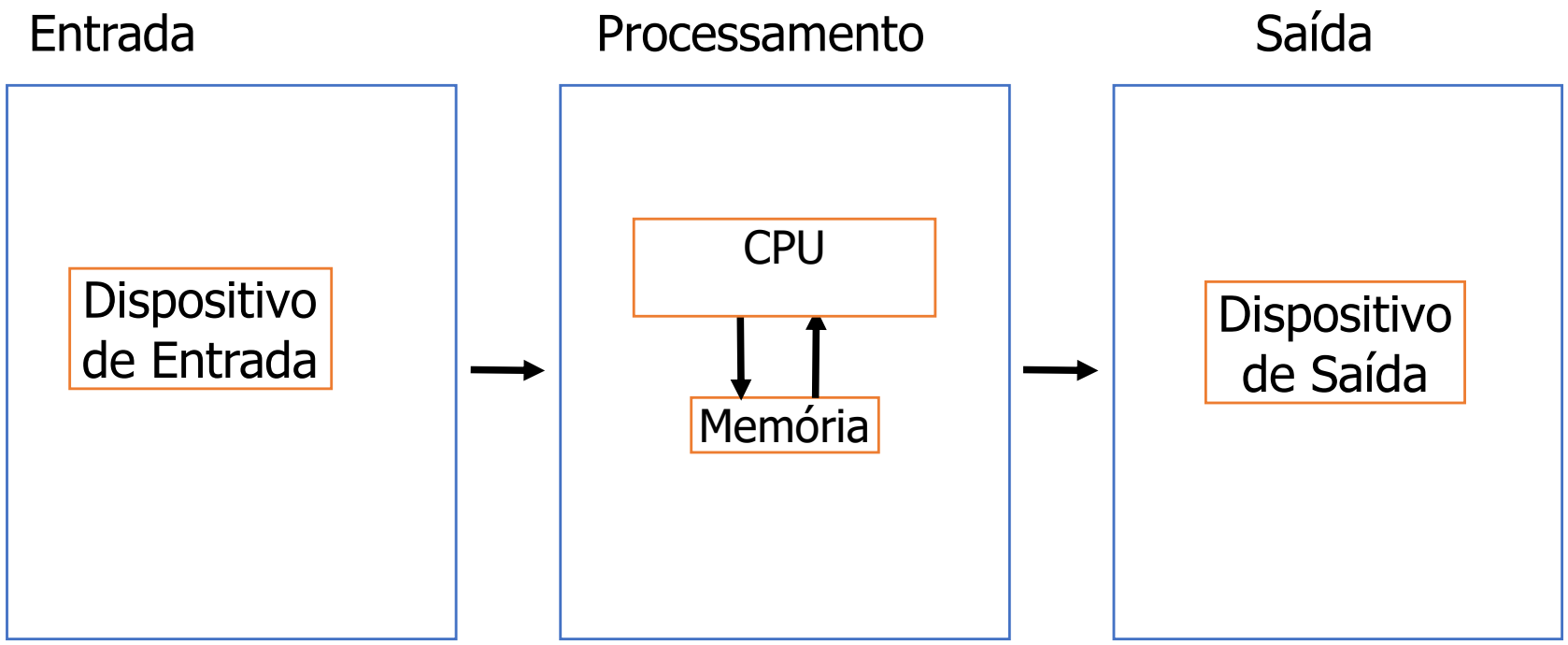


Modelo Matemático

Implementação

(Padrão de bits/rotinas)

Processamento de Dados:
O esquema.



Definindo Abstração

Abstração

Quando a matéria-prima usada num processo é abstrata, isto é, apresenta-se sob a forma de valores, quantidades ou símbolos, então falamos em processamento de dados.

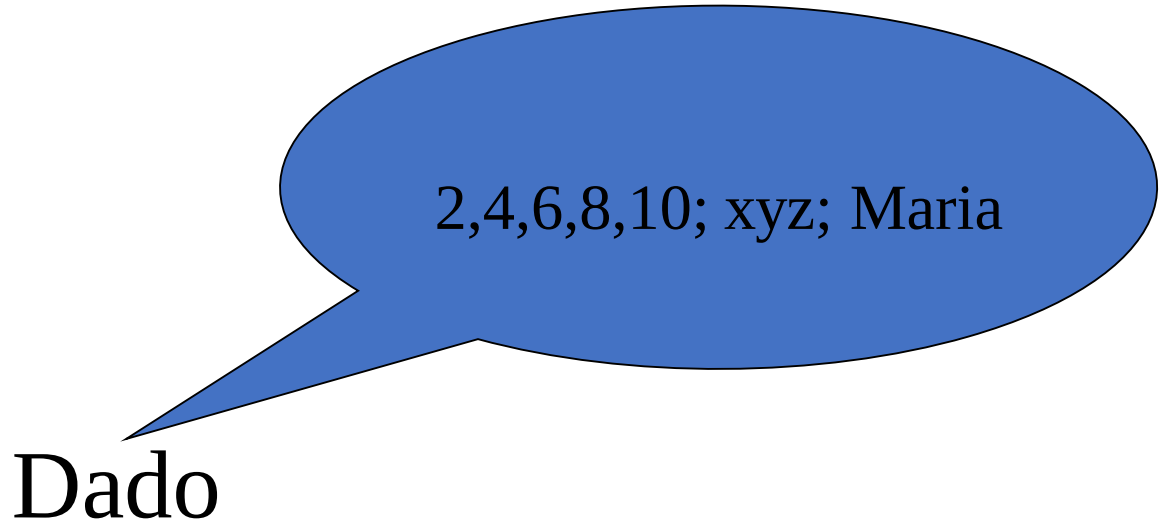
Quando o processamento é realizado por um computador, entrada refere-se aos dados colhidos do mundo real externo ao computador, e processo refere-se a uma série finita de operações que são realizadas a partir destes dados, a fim de transformá-los em alguma informação desejada (saída).

Importante

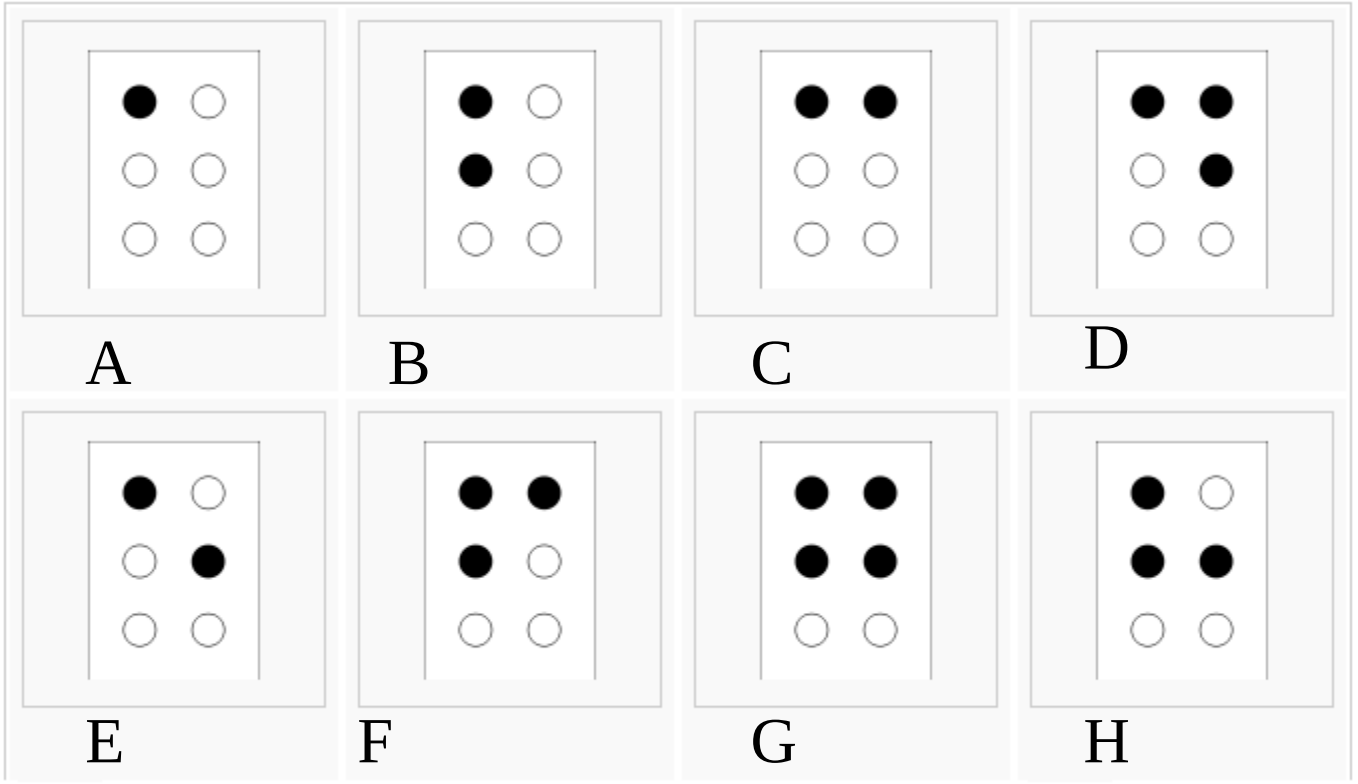
- Nem todo tipo de dado abstrato pode ser implementado em toda sua generalidade.
- Observe o conjunto Z
- Z = {...,-3,-2,-1,0,1,2,3,...}
- O conjunto Z deve ser finito.

Dado

É um conjunto de letras, números ou dígitos que colocado isoladamente, não agrega nenhum conhecimento, não contem significado claro.



Exemplo de dado (Informação)



Alfabeto Braille de seis dígitos

Dado => Informação

-.-. --- -.. .. --. --- / -- --- .-. ... .

C O D I G O (espaço) M O R S E

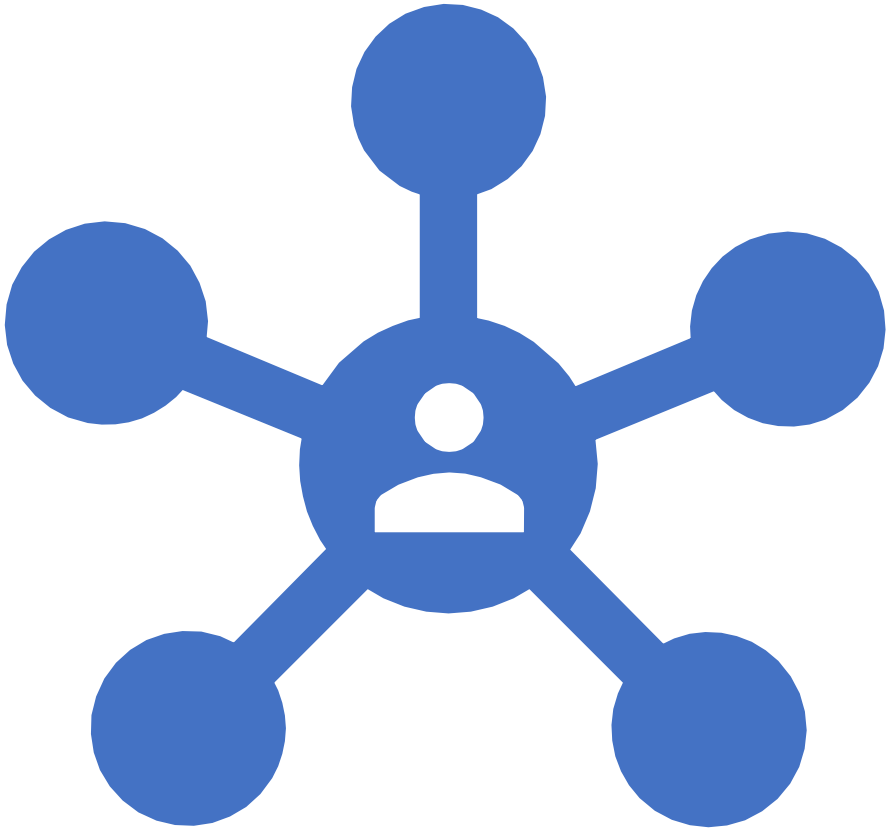
Informação

O conceito de informação vem ser o dado trabalhado ou tratado agregado com sentido natural e lógico para quem usa a informação. Define-se como algo útil.

2,4,6,8,10 – São Múltiplos de dois.

x,y,z - São coordenadas cartesianas.

Maria - Nome de uma pessoa.



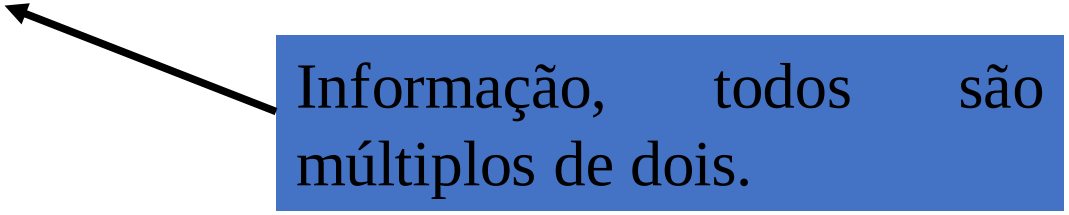
Conhecimento

Quando a informação é “trabalhada” por pessoas e pelos recursos computacionais, possibilitando geração de cenários, simulações e oportunidades, pode ser chamada de conhecimento.

Exemplo de Conhecimento Problema

Desenvolver uma função matemática para gerar apenas múltiplos de dois.

N={2,4,6,8,10...} <= Dado



Conhecimento : N = 2x

Conceito (Dado, Informação e Conhecimento)

Interpretação de símbolos

Simbologias (SI)

Proibido Fumar

Alta Tensão

Deficiente Físico

Enviar dados para Impressora

Radiação no local

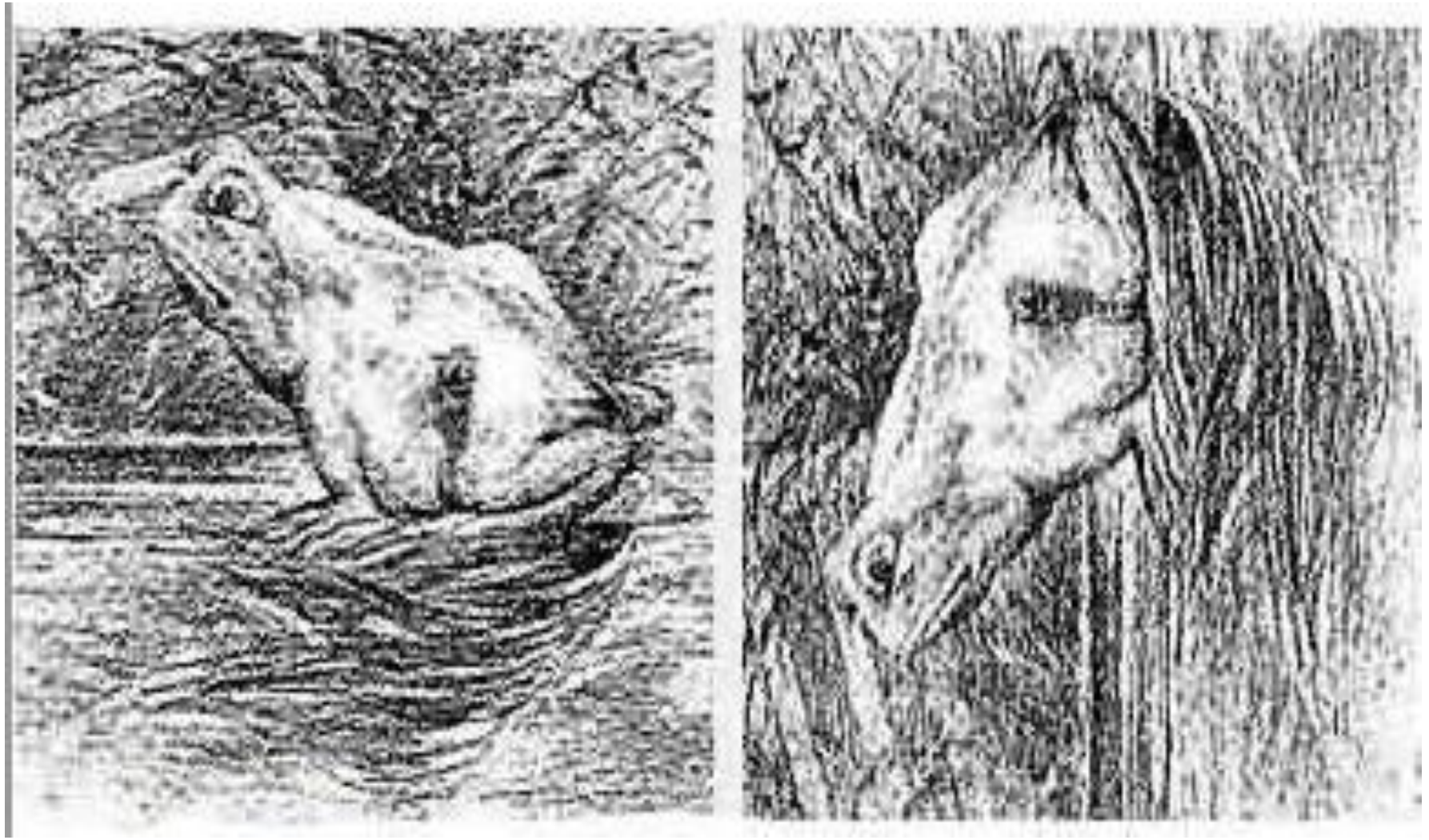
Mulher e Homem

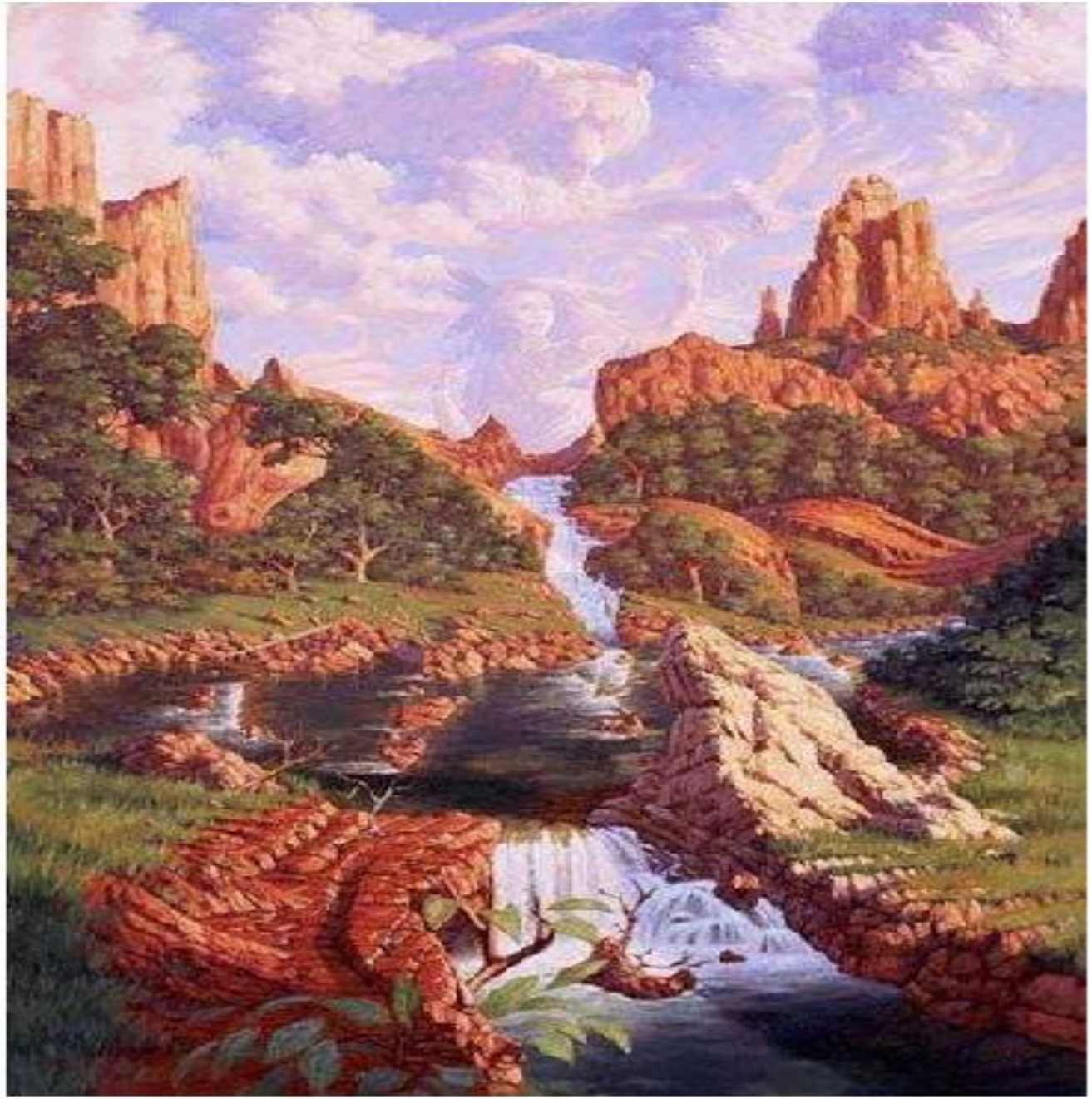
Laser no local

Proibido Estacionar

Conhecimento Visual

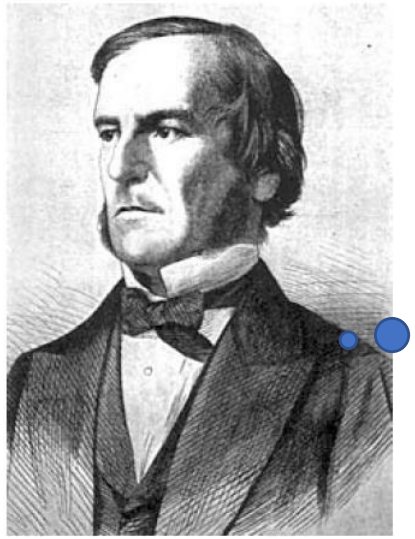
O Olho e o cérebro usam os sentidos para determinar formas implícitas ou codificadas nos objetos e formar conhecidos. Abstraindo dados da imagem e criando informações.



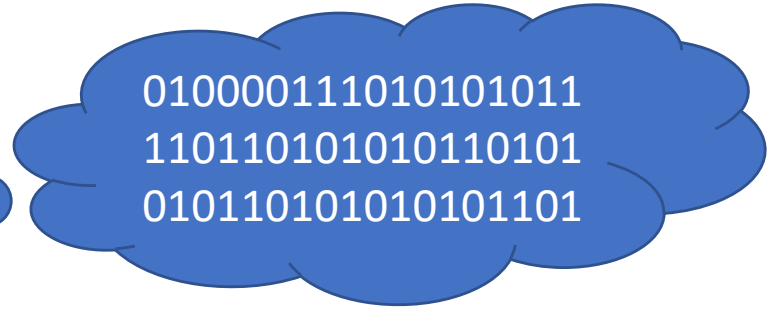


REPRESENTAÇÃO DE DADOS

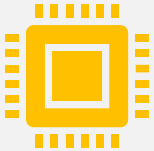
- O matemático inglês George Boole (1815-1864) publicou em 1854 os princípios da lógica booleana.
- Segundo Boole tudo poderia ser representado utilizando apenas os números 0 e 1.



George Boole



Bit



Simplificação de “dígito binário”(BInary digiT em inglês)

É a menor unidade de informação que pode ser armazenada ou transmitida.

Um bit pode assumir somente 2 valores, por exemplo: 0 ou 1, verdadeiro ou falso.

Byte

- ✓ Um byte nada tem de especial, é apenas um número binário de oito algarismos

0 1 0 1 0 1 1 1

Bytes

- ✓ 1 Byte é representado por uma cadeia de 8 bits

1 byte = 8 bits

1024 bytes = 1 K byte

1.048.576 bytes = 1 Mega byte

Noção de tamanho

Bit		2^0	0 ou 1
Byte		2^3	8 bits
Kilo	1 Kbyte	2^{10}	1024 Bytes
Mega	1 Mbyte	2^{20}	1 024 kB
Giga	1 Gbyte	2^{30}	1 024 MB
Tera	1 Tbyte	2^{40}	1 024 GB
peta	1 Pbyte	2^{50}	1 024 TB
Exa	1 Ebyte	2^{60}	1 024 PB
Zetta	1 Zbyte	2^{70}	1 024 EB
Yotta	1 Ybyte	2^{80}	1 024 ZB

Tipos de dados

Tipo	descrição	Bits	
byte	Inteiro sem sinal	8	0 a 255
sbyte	inteiro com sinal com sinal	8	-128 a 127
int	inteiro com sinal com sinal	32	-2,147,483,648 to 2,147,483,647
uint	Inteiro sem sinal	32	0 a 4294967295
short	inteiro com sinal com sinal	16	-32.768 a 32.767
long	inteiro com sinal com sinal	64	-9223372036854775508 to 9223372036854775507
ulong	Inteiro sem sinal	64	0 a 18446744073709551615

Importância da escolha correta do tipo de dados

- Economia de memória.
- Economia de processador.
- Economia de Disco.
- Qual o resultado da economia?

Conceito de Estrutura de Dados

Uma estrutura de dados é um modo de armazenar os dados no computador para que os dados sejam usados com eficiência.

Normalmente devem ser escolhidas cuidadosamente;

Uma estrutura de dados bem desenvolvida permite que uma variedade de operações críticas sejam implementadas por uma linguagem de programação com os tipos de dados e referências e as operações advindas dos mesmos.

A **semântica** em um programa é o efeito que aquele comando tem sobre os valores envolvidos. Por exemplo:

```
a = b ;  // para as variáveis  a   e   b   a semântica deve analisar se os tipos
         // das variáveis a   e   b   são compatíveis com as regras da LP
calcule ( n1, n2 ) ;  /*  é uma chamada para a  função calcule
                          a semântica, aqui, deve analisar se método calcule
                          aceita dois parâmetros, se o tipo dos parâmetros
                          e o tipo do valor retornado são compatíveis.
                       */
```

semântica

A semântica em um programa é o efeito que aquele comando tem sobre os valores envolvidos.

Exemplificando

Regras de nomes em C:

Variável: começa com _ e letra, os símbolos seguintes podem ser _, letra ou número.

a, **a1** e **endereco_residencial** são exemplos de variáveis válidas.

1a e endereco residencial são exemplos não válidos, pois o primeiro começa com número e o segundo tem espaço.

A linguagem **C** é *case-sensitive*, isto é, a variável saldo é diferente da variável **SALDO**.

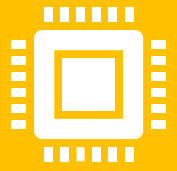
Já o **Visual Basic** (VB) não é *case-sensitive*.

Em programa escrito em VB a variável **saldo** e **SALDO**, são as mesmas entidades (iguais).

Paradigma imperativo

as linguagens imperativas têm as variáveis e os programas armazenados juntos na memória, além dos comandos e das atribuições, que são usados para cálculos, entradas e saídas.

A estrutura e os recursos da linguagem permitem uma transcrição quase direta da solução algorítmica.



Exemplos de linguagens de programação imperativas: FORTRAN, COBOL, C, Basic, ALGOL, Pascal, PHP e Java.

Paradigma orientado a objeto

a base para entender o paradigma orientado a objeto está na abstração dos dados e tipos.

Pensar em uma solução orientada a objeto é modelar o mundo como peças (objetos) de forma abstrata.

Exemplo de linguagens POO: Smalltalk, C++, C#, Java e Python

Paradigma funcional

- A primeira linguagem funcional, a LISP, surgiu em 1958. Hoje, a demanda das linguagens funcionais está em franco crescimento em virtude das aplicações na área de inteligência artificial, pois facilita a programação para sistemas, baseando-se em regras e processamento de linguagem natural.

- A linguagem ELIXIR é uma linguagem funcional nova, que em apenas cinco anos de existência obteve a aceitação extraordinária no mercado.

- A HASKELL é totalmente funcional. Ela surgiu de 1990, sendo recomendada para quem quer começar a desenvolver usando esse paradigma

Paradigma lógico

a chave para você raciocinar de acordo com o paradigma lógico é desenvolver a solução declarando qual resultado o programa deve alcançar, em vez de como o programa deve alcançar tal resultado.

A linguagem Prolog é o melhor exemplo para esse paradigm

Java multiparadigmas

Exemplificando

Vejamos um exemplo em Java, que é uma linguagem multiparadigmas, para percorrer uma lista de dados.

No exemplo, da linha 7 à 9, é usada uma instrução imperativa. Na linha 11 utilizamos o paradigma funcional, e nas linhas 13 a 16 a orientação a objeto com a abstração de dados.

```
01 import java.util.*;
02 public class Exemplo01 {
03     public static void main(String[] args){
04     //declaração dos dados ( lista )
05      List<Integer> dados = Arrays.asList(0,1,2,3,4,5,6,7,8,9,10);
06      //imperativa
07      for( int num : dados ) {
08          System.out.println(num);
09      }
10     // funcional
11       dados.stream().forEach((num) -> {System.out.println(num);});
12     //orientação a objeto
13       for (Iterator<Integer> it = dados.iterator(); it.hasNext();) {
14                                        int num = it.next();
15                                        System.out.println(num);
16        } //fim for(...)
17    }  // fim main()
18  } // fim class Exemplo01
```

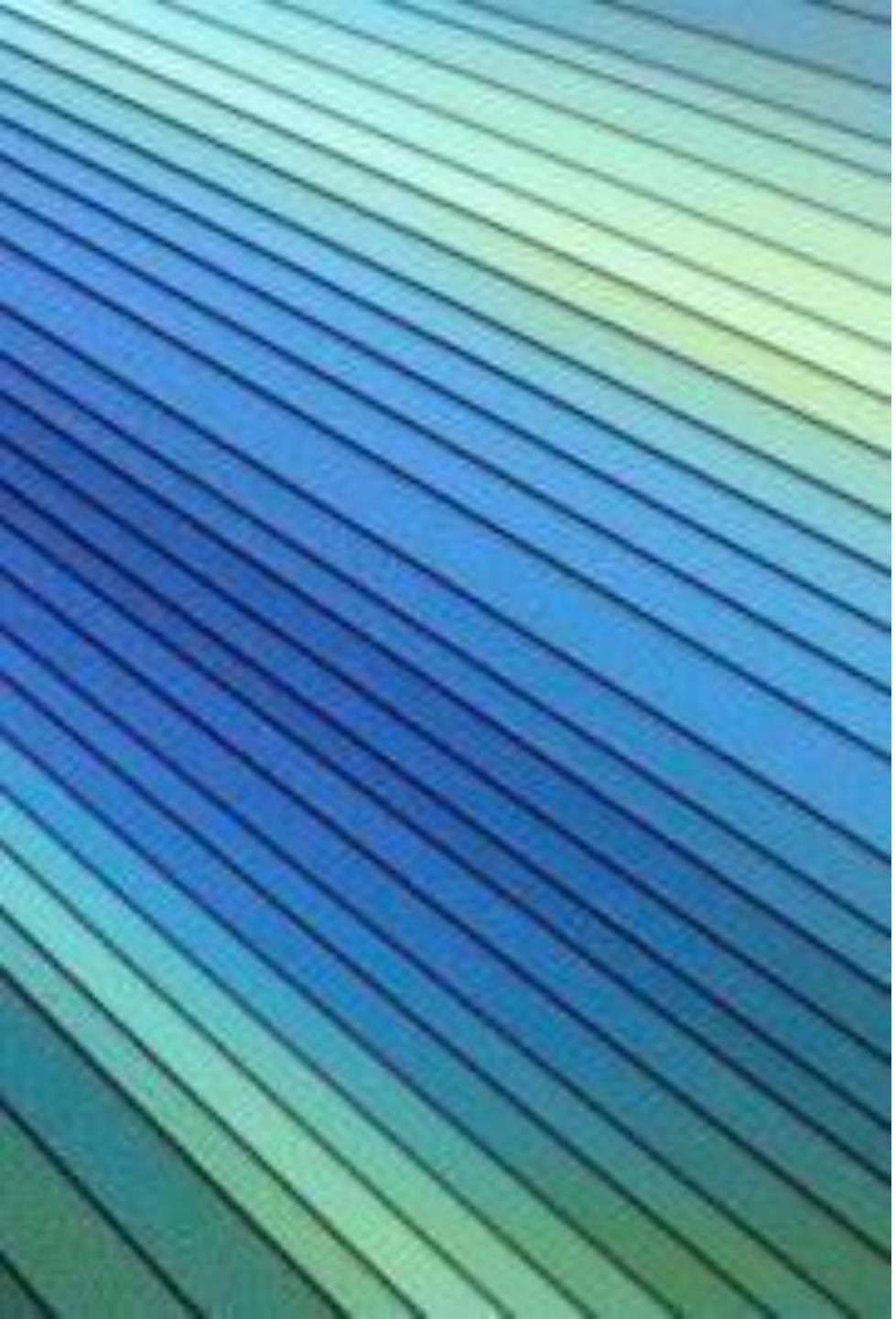
Evolução das linguagens de programação.

- A evolução das linguagens de programação está associada aos tipos de tradutores, que, segundo Simão & Prince (2001), podem ser classificados em:

Montadores

programas que traduzem um código fonte escrito em linguagem básica (assembly) em código de máquina, também denominado Assembler.

Essa tradução normalmente é feita em um ou dois passos



Compiladores

traduzem o código fonte, programa escrito em uma linguagem de alto nível, em código alvo.

Após o processo de compilação, o código alvo não precisa mais passar pelo processo de compilação para ser novamente executado.

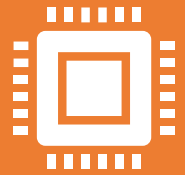
É um programa complexo e requer várias fases

Interpretadores

fazem o mesmo processo do compilador, mas cada comando (linha de programa) analisado executa todas as fases de tradução até alcançar o código alvo e já o executa.

As linguagens interpretadas exigem que os programas sejam submetidos ao interpretador todas as vezes que necessitarem ser executadas

Compiladores híbridos



seguem todos os passos de um compilador, mas não geram o código executável, e sim um código intermediário que será executado por uma máquina virtual.

A grande vantagem desse tipo de compilador é a portabilidade

A fase da análise

está diretamente associada à verificação de se o programa foi escrito corretamente, isto é, de acordo com as regras da linguagem.

A análise está subdividida em 3 etapas

análise Léxica

em que se verifica se os nomes das entidades estão corretos.

Hélio

Helio

Elio

Élio?

Sintática

analisa-se se os comandos estão corretos.

Chama a verificação da frase.

Aqui, não basta escrever as palavras corretamente, importando também a ordem em que elas aparecem

Semântica

- nesse ponto, verifica-se o contexto.
- No caso das linguagens de programação, o compilador deverá analisar se os valores envolvidos nos comandos estão compatíveis.

- Int x= “helio”;

fase de síntese

Acontece depois da fase de análise.

A fase de síntese visa gerar o código alvo a partir dos resultados obtidos pelas fases intermediárias, que são:

Fase 1: Gerenciamento da tabela de símbolos

- as diversas etapas do compilador alimentam e consultam essa tabela para coletar informações (nomes, tipos, atributos).

Fase 2: Geração do código intermediário:

Exemplo: r = a + b * raiz(16)

```
s1 = raiz(16)
s2 = b * s1
s3 = a + s2
r = s3
```

- a conversão para o código alvo é feita em etapas, e esta é uma conversão intermediária, em que as instruções serão representadas em não mais que três endereços:

Fase 4 :Otimização do código:

- realiza transformações no código com o objetivo de melhorar o tempo de execução, o consumo de memória ou o tamanho do código

Fase final: Geração do código

- que gera o código alvo

Referências:

- Compiladores / Regina Fedozzi. – Londrina : Editora e Distribuidora Educacional S.A., 2018.