Editora Solução
a solução para o seu concurso!

IBGE

CENSO AGRO

ANALISTA CENSITÁRIO
DESENVOLVIMENTO DE TECNOLOGIA DA INFORMAÇÃO

- Língua Portuguesa
- Raciocínio Lógico Quantitativo
- Conhecimentos Específicos

INCLUI QUESTÕES GABARITADAS

DE ACORDO COM O EDITAL Nº 2/2026

BÔNUS
ÁREA DO CONCURSEIRO

- **Português:** Ortografia, Fonologia, Acentuação Gráfica, Concordância, Regência, Crase e Pontuação.
- **Informática:** Computação na Nuvem, Armazenamento em Nuvem, Internet, Conceitos, Protocolos e Segurança da informação.

41 ANOS
A SOLUÇÃO PARA O SEU CONCURSO

AVISO IMPORTANTE:

Este é um Material de Demonstração

Este arquivo é apenas uma amostra do conteúdo completo da Apostila.

Aqui você encontrará algumas páginas selecionadas para que possa conhecer a qualidade, estrutura e metodologia do nosso material. No entanto, esta não é a apostila completa.

POR QUE INVESTIR NA APOSTILA COMPLETA?

- Conteúdo totalmente alinhado ao edital
- Teoria clara, objetiva e sempre atualizada
- Questões gabaritadas
- Diferentes práticas que otimizam seus estudos

Ter o material certo em mãos transforma sua preparação e aproxima você da APROVAÇÃO.

Garanta agora o acesso completo e aumente suas chances de aprovação:
https://www.editorasolucao.com.br/

Editora Solução
a solução para o seu concurso!

IBGE AGRO

CENSO AGROPECUÁRIO, FLORESTAL E AQUÍCOLA - FUNDAÇÃO INSTITUTO BRASILEIRO DE GEOGRAFIA E ESTATÍSTICA

Analista Censitário - Desenvolvimento de Tecnologia da Informação

EDITAL Nº 2/2026

CÓD: SL-093JH-26
7901183000579

Língua Portuguesa

Raciocínio Lógico Quantitativo

Conhecimentos Específicos
Analista Censitário - Desenvolvimento de Tecnologia da Informação

ÍNDICE

ÍNDICE

LÍNGUA PORTUGUESA

COMPREENSÃO E INTERPRETAÇÃO DE TEXTO; ESTRUTURA E SEQUÊNCIA LÓGICA DE FRASES E PARÁGRAFOS

Compreender um texto nada mais é do que analisar e decodificar o que de fato está escrito, seja das frases ou de ideias presentes. Além disso, interpretar um texto, está ligado às conclusões que se pode chegar ao conectar as ideias do texto com a realidade.

A compreensão básica do texto permite o entendimento de todo e qualquer texto ou discurso, com base na ideia transmitida pelo conteúdo. Ademais, compreender relações semânticas é uma competência imprescindível no mercado de trabalho e nos estudos.

A interpretação de texto envolve explorar várias facetas, desde a compreensão básica do que está escrito até as análises mais profundas sobre significados, intenções e contextos culturais. No entanto, quando não se sabe interpretar corretamente um texto pode-se criar vários problemas, afetando não só o desenvolvimento profissional, mas também o desenvolvimento pessoal.

▸ Busca de sentidos

Para a busca de sentidos do texto, pode-se extrair os tópicos frasais presentes em cada parágrafo. Isso auxiliará na compreensão do conteúdo exposto, uma vez que é ali que se estabelecem as relações hierárquicas do pensamento defendido, seja retomando ideias já citadas ou apresentando novos conceitos.

Por fim, concentre-se nas ideias que realmente foram explicitadas pelo autor. Textos argumentativos não costumam conceder espaço para divagações ou hipóteses, supostamente contidas nas entrelinhas. Deve-se atentar às ideias do autor, o que não implica em ficar preso à superfície do texto, mas é fundamental que não se criem suposições vagas e inespecíficas.

▸ Importância da interpretação

A prática da leitura, seja por prazer, para estudar ou para se informar, aprimora o vocabulário e dinamiza o raciocínio e a interpretação. Ademais, a leitura, além de favorecer o aprendizado de conteúdos específicos, aprimora a escrita.

Uma interpretação de texto assertiva depende de inúmeros fatores. Muitas vezes, apressados, descuidamo-nos dos detalhes presentes em um texto, achamos que apenas uma leitura já se faz suficiente. Interpretar exige paciência e, por isso, sempre releia o texto, pois a segunda leitura pode apresentar aspectos surpreendentes que não foram observados previamente.

Para auxiliar na busca de sentidos do texto, pode-se também retirar dele os tópicos frasais presentes em cada parágrafo, isso certamente auxiliará na apreensão do conteúdo exposto. Lembre-se de que os parágrafos não estão organizados, pelo menos em um bom texto, de maneira aleatória, se estão no lugar que estão, é porque ali se fazem necessários, estabelecendo uma relação hierárquica do pensamento defendido; retomando ideias já citadas ou apresentando novos conceitos.

Concentre-se nas ideias que de fato foram explicitadas pelo autor: os textos argumentativos não costumam conceder espaço para divagações ou hipóteses, supostamente contidas nas entrelinhas. Devemos nos ater às ideias do autor, isso não quer dizer que você precise ficar preso na superfície do texto, mas é fundamental que não criemos, à revelia do autor, suposições vagas e inespecíficas.

Ler com atenção é um exercício que deve ser praticado à exaustão, assim como uma técnica, que fará de nós leitores proficientes.

▸ Diferença entre compreensão e interpretação

A compreensão de um texto envolve realizar uma análise objetiva do seu conteúdo para verificar o que está explicitamente escrito nele. Por outro lado, a interpretação vai além, relacionando as ideias do texto com a realidade. Nesse processo, o leitor extrai conclusões subjetivas a partir da leitura.

▸ Gêneros Discursivos

▪ **Romance:** descrição longa de ações e sentimentos de personagens fictícios, podendo ser de comparação com a realidade ou totalmente irreal. A diferença principal entre um romance e uma novela é a extensão do texto, ou seja, o romance é mais longo. No romance nós temos uma história central e várias histórias secundárias.

▪ **Conto:** obra de ficção onde é criado seres e locais totalmente imaginário. Com linguagem linear e curta, envolve poucas personagens, que geralmente se movimentam em torno de uma única ação, dada em um só espaço, eixo temático e conflito. Suas ações encaminham-se diretamente para um desfecho.

▪ **Novela:** muito parecida com o conto e o romance, diferenciado por sua extensão. Ela fica entre o conto e o romance, e tem a história principal, mas também tem várias histórias secundárias. O tempo na novela é baseada no calendário. O tempo e local são definidos pelas histórias dos personagens. A história (enredo) tem um ritmo mais acelerado do que a do romance por ter um texto mais curto.

▪ **Crônica:** texto que narra o cotidiano das pessoas, situações que nós mesmos já vivemos e normalmente é utilizado a ironia para mostrar um outro lado da mesma história. Na crônica o tempo não é relevante e quando é citado, geralmente são pequenos intervalos como horas ou mesmo minutos.

▪ **Poesia:** apresenta um trabalho voltado para o estudo da linguagem, fazendo-o de maneira particular, refletindo o momento, a vida dos homens através de figuras que possibilitam a criação de imagens.

▪ **Editorial:** texto dissertativo argumentativo onde expressa a opinião do editor através de argumentos e fatos sobre um assunto que está sendo muito comentado (polêmico). Sua intenção é convencer o leitor a concordar com ele.

▪ **Entrevista:** texto expositivo e é marcado pela conversa de um entrevistador e um entrevistado para a obtenção de informações. Tem como principal característica transmitir a opinião de pessoas de destaque sobre algum assunto de interesse.

▪ **Cantiga de roda:** gênero empírico, que na escola se materializa em uma concretude da realidade. A cantiga de roda permite as crianças terem mais sentido em relação a leitura e escrita, ajudando os professores a identificar o nível de alfabetização delas.

▪ **Receita:** texto instrucional e injuntivo que tem como objetivo de informar, aconselhar, ou seja, recomendam dando uma certa liberdade para quem recebe a informação.

SIGNIFICAÇÃO DAS PALAVRAS: SINÔNIMOS, ANTÔNIMOS, HOMÔNIMOS E PARÔNIMOS

As palavras podem ter diversos sentidos em uma comunicação. E isso também é estudado pela Gramática Normativa: quem cuida dessa parte é a Semântica, que se preocupa, justamente, com os significados das palavras.

Veremos, então, cada um dos conteúdos que compõem este estudo.

▸ Antônimo e Sinônimo

O **antônimo** é a palavra que possui sentido oposto. Por exemplo, “felicidade” é o antônimo de “tristeza”, porque o significado de uma é o oposto da outra. Da mesma forma ocorre com “quente” que é antônimo de “frio”.

Já os **sinônimos** são palavras que têm sentido aproximado e que podem, inclusive, substituir umas às outras. O uso de sinônimos é muito importante para evitar a repetição desnecessária de palavras nos textos. Observe os exemplos a seguir:

Felicidade é sinônimo de alegria/contentamento; e
Rápido é sinônimo de veloz.

▸ Hipônimos e Hiperônimos

Estes conceitos são simples de entender: o **hipônimo** designa uma palavra de sentido mais específico, enquanto que o **hiperônimo** designa uma palavra de sentido mais genérico.

Ex:

Cachorro e gato são hipônimos, pois têm sentido específico.

Já “animais domésticos” é uma expressão hiperônima, pois indica um sentido mais genérico de animais.

Outros exemplos ajudam a visualizar melhor a relação entre termos específicos e gerais:

rosa, orquídea e lírio são hipônimos de flor;
flor é hiperônimo de cada uma dessas espécies;
veículo, por sua vez, é hiperônimo de carro, moto, ônibus e bicicleta.

Essa relação hierárquica é sempre de “termo geral” → “termos específicos”.

Atenção: não confunda hiperônimo com substantivo coletivo. Hiperônimos estão no ramo dos sentidos das palavras.

▸ Conotação e Denotação

Observe as frases:
Amo pepino na salada.
Tenho um “pepino” para resolver.
As duas frases têm uma palavra em comum: pepino.

Mas na primeira frase, pepino está no sentido **denotativo**, ou seja, a palavra está sendo usada no sentido próprio, comum, dicionarizado.

Já na segunda frase, a mesma palavra está no sentido **conotativo**, pois ela está sendo usada no sentido figurado e depende do contexto para ser entendida.

Convém destacar que denotação corresponde ao significado básico, dicionarizado, enquanto conotação corresponde ao

RACIOCÍNIO LÓGICO QUANTITATIVO

AVALIAÇÃO DA HABILIDADE DO CANDIDATO EM ENTENDER A ESTRUTURA LÓGICA DE RELAÇÕES ENTRE PESSOAS, LUGARES, COISAS E/OU EVENTOS, DEDUZIR NOVAS INFORMAÇÕES E AVALIAR AS CONDIÇÕES USADAS PARA ESTABELECER A ESTRUTURA DESSAS RELAÇÕES

Estruturas lógicas

Antes de tudo, é essencial compreender o conceito de proposições. Uma proposição é definida como uma sentença declarativa à qual podemos atribuir um único valor lógico: verdadeiro ou falso, nunca ambos. Em outras palavras, trata-se de uma sentença que pode ser considerada fechada.

Existem diferentes tipos de proposições, sendo as principais:

▪ **Sentenças abertas:** são sentenças para as quais não é possível atribuir um valor lógico verdadeiro ou falso, e, portanto, não são consideradas frases lógicas.

Exemplos incluem:

Frases interrogativas: "Quando será a prova?", "Estudou ontem?", "Fez sol ontem?".

Frases exclamativas: "Gol!", "Que maravilhoso!".

Frases imperativas: "Estude e leia com atenção.", "Desligue a televisão.".

Frases sem sentido lógico (expressões vagas, paradoxais, ambíguas, etc.): "Esta frase é falsa." (expressão paradoxal), "O cachorro do meu vizinho morreu." (expressão ambígua), "2 + 5 + 1".

▪ **Sentença fechada:** Uma sentença lógica é aquela que admite um ÚNICO valor lógico, seja ele verdadeiro ou falso.

Proposições simples e compostas

Proposições simples, também conhecidas como atômicas, são aquelas que NÃO contêm nenhuma outra proposição como parte integrante de si mesma. Elas são designadas pelas letras latinas minúsculas p, q, r, s..., sendo chamadas de letras proposicionais.

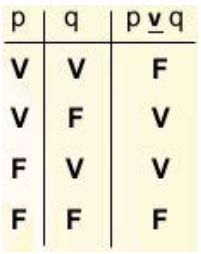
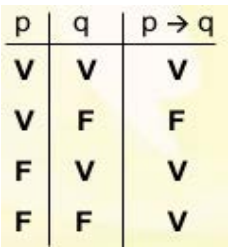
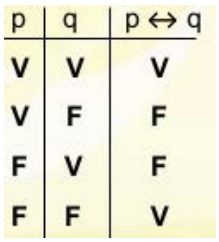
Por outro lado, proposições compostas, também conhecidas como moleculares ou estruturas lógicas, são formadas pela combinação de duas ou mais proposições simples. Elas são designadas pelas letras latinas maiúsculas P, Q, R, S..., também chamadas de letras proposicionais.

É importante ressaltar que TODAS as proposições compostas são formadas por duas ou mais proposições simples.

Proposições Compostas – Conectivos

As proposições compostas são constituídas por proposições simples conectadas por conectivos, os quais determinam seu valor lógico. Isso pode ser observado na tabela a seguir:

Operação	Conectivo	Estrutura Lógica	Tabela verdade
Negação	~	**Não** p	p | ~p V | F F | V
Conjunção	^	p **e** q	p | q | p ∧ q V | V | V V | F | F F | V | F F | F | F

Disjunção Inclusiva	v	p **ou** q	
Disjunção Exclusiva	$\underline{v}$	**Ou** p **ou** q	
Condicional	→	**Se** p **então** q	
Bicondicional	↔	p **se e somente se** q	

p	q	p v q
V	V	V
V	F	V
F	V	V
F	F	F

p	q	p $\underline{v}$ q
V	V	F
V	F	V
F	V	V
F	F	F

p	q	p → q
V	V	V
V	F	F
F	V	V
F	F	V

p	q	p ↔ q
V	V	V
V	F	F
F	V	F
F	F	V

Em resumo, a tabela verdade das proposições simplifica a resolução de várias questões.

P	Q	P ∧ Q	P ∨ Q	P $\underline{v}$ Q	P → Q	P ↔ Q
V	V	V	V	F	V	V
V	F	F	V	V	F	F
F	V	F	V	V	V	F
F	F	F	F	F	V	V

As sequências podem ser compostas por números, letras, pessoas, figuras e assim por diante. Há várias maneiras de estabelecer uma sequência, mas o importante é que haja pelo menos três elementos que caracterizem a lógica de sua formação. No entanto, algumas séries exigem mais elementos para definir sua lógica. Ter um bom conhecimento em Progressões Aritméticas (PA) e Progressões Geométricas (PG) torna a dedução das sequências simples e sem complicações. É crucial estar atento a vários detalhes oferecidos por elas, como nos exemplos abaixo:

CONHECIMENTOS ESPECÍFICOS

CONCEITO DE COMPILAÇÃO E LIGAÇÃO DE PROGRAMAS

Um programa normalmente começa como um conjunto de instruções escritas por uma pessoa em uma linguagem de programação, como C, C++, Java, Pascal ou outras linguagens de alto nível. Esse texto inicial recebe o nome de código-fonte. Embora seja compreensível para programadores, ele não pode ser executado diretamente pelo processador, pois a unidade central de processamento trabalha com instruções em baixo nível, representadas internamente por sequências binárias.

A transformação do código-fonte em uma forma executável é, portanto, um processo de tradução. Essa tradução permite que comandos mais próximos da linguagem humana sejam convertidos em instruções que o computador consiga interpretar e executar. No caso de linguagens compiladas, essa conversão ocorre antes da execução do programa, por meio de ferramentas específicas, especialmente o compilador e o ligador.

▸ Linguagens de alto nível, linguagem de montagem e código de máquina

Diferenças entre os níveis de representação de um programa

As linguagens de alto nível foram criadas para facilitar o desenvolvimento de programas, permitindo que o programador escreva comandos mais abstratos e organizados. Em vez de lidar diretamente com endereços de memória, registradores e instruções específicas do processador, o programador utiliza estruturas como variáveis, funções, comandos condicionais, laços de repetição e tipos de dados.

A linguagem de montagem, também chamada de assembly, está em um nível mais próximo do hardware. Ela utiliza instruções simbólicas que correspondem de maneira mais direta às operações realizadas pelo processador. Já o código de máquina é a forma final, composta por instruções binárias que a CPU consegue executar diretamente.

Para compreender essa diferença, é útil observar a progressão entre as formas de representação de um programa:

- Código-fonte em alto nível: é escrito de maneira mais compreensível ao programador, com comandos estruturados e maior abstração.
- Linguagem de montagem: representa instruções próximas do processador, usando símbolos em vez de sequências binárias puras.
- Código de máquina: corresponde às instruções finais, codificadas em formato binário, executáveis diretamente pelo hardware.

▸ O papel do compilador no desenvolvimento de software

Tradução, verificação e preparação do código

O compilador é o programa responsável por analisar o código-fonte e transformá-lo em uma representação de baixo nível, geralmente chamada de código objeto. Durante esse processo, ele não apenas traduz comandos, mas também verifica se o programa foi escrito de acordo com as regras da linguagem. Assim, erros de sintaxe, incompatibilidades de tipos e construções inválidas podem ser identificados antes que o programa seja executado.

Além da tradução, o compilador pode realizar otimizações. Isso significa reorganizar ou ajustar partes do código para que o programa final seja mais eficiente, consuma menos memória ou execute mais rapidamente. Essas otimizações, porém, devem preservar o comportamento lógico definido pelo programador.

▸ Visão geral do fluxo: código-fonte, compilação, ligação e execução

A construção progressiva de um programa executável

O caminho entre escrever um programa e executá-lo envolve etapas organizadas. Primeiro, o programador cria o código-fonte. Depois, o compilador processa esse código e gera arquivos intermediários ou arquivos objeto. Esses arquivos ainda não correspondem necessariamente a um programa completo, pois podem depender de outras partes do projeto ou de bibliotecas externas.

Nesse ponto, entra o processo de ligação. O ligador reúne os arquivos objeto, resolve dependências entre funções e variáveis, associa bibliotecas necessárias e produz o arquivo executável. Somente depois disso o sistema operacional pode carregar o programa na memória e iniciar sua execução.

Assim, compilação e ligação são etapas complementares. A compilação traduz cada parte do programa para uma forma próxima da máquina, enquanto a ligação combina essas partes em uma unidade executável coerente. Compreender essa separação é essencial para interpretar erros, organizar projetos e entender como programas são efetivamente construídos.

Processo de compilação

Compreensão geral da compilação

A compilação como etapa de análise e transformação

A compilação é o processo responsável por transformar o código-fonte escrito pelo programador em uma forma mais próxima da linguagem da máquina. Essa transformação não ocorre de maneira simples ou imediata. Antes de gerar o código objeto, o compilador precisa verificar se o programa está corretamente escrito, se as instruções seguem as regras da linguagem, se os tipos de dados são compatíveis e se as estruturas usadas fazem sentido dentro do contexto do programa.

Dessa forma, a compilação pode ser compreendida como uma sequência de etapas organizadas. Cada etapa observa o código sob um ponto de vista diferente. Algumas fases se concentram na forma do texto escrito, outras verificam o significado das instruções, e outras produzem representações intermediárias ou finais. Essa divisão torna o processo mais seguro, pois permite detectar erros antes da execução e preparar o programa para uma conversão eficiente.

Principais etapas do processo de compilação

Análise léxica, sintática e semântica

A primeira grande parte da compilação consiste em analisar o código-fonte. Nessa fase, o compilador examina o texto escrito pelo programador e procura compreender sua estrutura. A análise léxica identifica os elementos básicos do código, como palavras reservadas, nomes de variáveis, operadores, números e símbolos. A análise sintática verifica se esses elementos estão organizados conforme a gramática da linguagem. Já a análise semântica avalia se as instruções fazem sentido, considerando tipos de dados, declarações e uso correto de variáveis e funções.

Para tornar essa sequência mais didática, a tabela a seguir resume a função de cada análise dentro da compilação:

Etapa de análise	O que verifica	Exemplo de problema identificado
Análise léxica	Identifica os componentes básicos do código	Símbolo inválido ou palavra malformada
Análise sintática	Verifica a organização gramatical das instruções	Falta de ponto e vírgula, parêntese ou chave
Análise semântica	Avalia o sentido das instruções	Uso de variável não declarada ou tipo incompatível

Geração e otimização de código

Do código intermediário ao código objeto

Após as análises iniciais, o compilador pode gerar uma representação intermediária do programa. Essa representação não é necessariamente o código de máquina final, mas uma forma interna usada para facilitar novas transformações. O código intermediário permite que o compilador trabalhe de maneira mais organizada, separando a lógica do programa dos detalhes específicos do processador ou do sistema.

Em seguida, podem ocorrer otimizações. O objetivo da otimização é melhorar o programa gerado sem alterar seu comportamento. O compilador pode eliminar instruções desnecessárias, simplificar expressões, reorganizar trechos de código e escolher formas mais eficientes de executar determinadas operações. Essas alterações são feitas de modo controlado, respeitando a lógica definida pelo programador.

Depois dessas etapas, o compilador gera o código objeto. Esse código já está em baixo nível, mas normalmente ainda não é um programa executável completo. Ele pode conter referências a funções, variáveis ou bibliotecas que serão resolvidas apenas na fase de ligação. Por isso, a compilação prepara partes do programa, mas nem sempre entrega o arquivo final que será executado diretamente pelo sistema operacional.

Erros de compilação

Como interpretar falhas detectadas pelo compilador

Os erros de compilação surgem quando o compilador encontra problemas que impedem a tradução correta do código-fonte. Esses erros geralmente indicam falhas na escrita do programa, como comandos incompletos, uso incorreto de tipos, variáveis não declaradas ou estruturas mal formadas. Diferentemente dos erros de execução, que aparecem quando o programa já está rodando, os erros de compilação são detectados antes da criação do programa final.

A leitura das mensagens de erro exige atenção. Muitas vezes, o compilador informa a linha aproximada do problema, o tipo de erro encontrado e uma breve descrição da causa. No entanto, o erro real pode estar em uma linha anterior, especialmente quando faltam símbolos de fechamento, como parênteses, chaves ou ponto e vírgula. Por isso, interpretar mensagens de compilação envolve analisar o trecho indicado e também o contexto imediatamente anterior.

Algumas categorias comuns de erro ajudam a compreender melhor o diagnóstico apresentado pelo compilador:

- Erro léxico: ocorre quando o compilador encontra caracteres ou símbolos que não pertencem à linguagem utilizada.
- Erro sintático: ocorre quando os elementos do código estão escritos em uma ordem inválida ou incompleta.
- Erro semântico: ocorre quando a instrução é estruturalmente válida, mas apresenta sentido incorreto dentro do programa.
- Erro de tipo: ocorre quando uma operação envolve dados incompatíveis, como atribuir texto a uma variável numérica sem conversão adequada.

Síntese didática da compilação

A função da compilação no ciclo de construção de programas

Editora Solução
a solução para o seu concurso!

GOSTOU DESSE MATERIAL?

Então não pare por aqui: a versão **COMPLETA** vai te deixar ainda mais perto da sua aprovação e da tão sonhada estabilidade. Aproveite o **DESCONTO EXCLUSIVO** que liberamos para Você!

EU QUERO DESCONTO!