**1. Fundamentos de Hardware**

• **Conceito de Hardware**: Refere-se a todos os **componentes físicos e tangíveis** de um dispositivo eletrônico, como computadores, celulares, tablets, smart TVs e eletrodomésticos inteligentes. É o "corpo" de um sistema computacional e onde todas as operações ocorrem.

• **Evolução e Lei de Moore**: Os componentes de hardware evoluem rapidamente, tornando-se mais potentes, compactos e acessíveis, em grande parte devido à **miniaturização dos circuitos eletrônicos**. A **Lei de Moore** descreve que o número de transistores em um chip dobra em 18 a 24 meses.

• **Componentes Básicos**:

    ◦ **CPU (Unidade Central de Processamento)**: O "cérebro" do computador, responsável por executar instruções e processar dados.

    ◦ **Registradores**: Pequenas áreas de memória na CPU para dados temporários, garantindo acesso quase instantâneo.

    ◦ **Memória Cache**: Memória intermediária veloz entre CPU e RAM, armazena dados frequentes para acelerar o processamento.

    ◦ **Memória RAM (Random Access Memory)**: Memória principal, temporária e volátil, que armazena dados em uso. Essencial para a execução simultânea de programas.

    ◦ **Memória Persistente (Armazenamento Permanente)**: Mantém dados mesmo após o desligamento. Exemplos incluem HD (Hard Disk), SSD (Solid State Drive), Pen drive e Cartões SD.

    ◦ **Microcontroladores**: Pequenos computadores em um único chip para tarefas específicas, presentes em dispositivos embarcados e IoT.

    ◦ **Interdependência**: Todos os componentes de hardware são interdependentes; o funcionamento eficiente de um sistema depende da **integração harmoniosa** entre eles.

• **Unidades de Armazenamento de Dados**:

    ◦ A base é o **bit (0 ou 1)**, a menor unidade de informação.

    ◦ Um **byte** é formado por 8 bits.

    ◦ Unidades maiores facilitam o entendimento: **Kilobyte (KB)**, **Megabyte (MB)**, **Gigabyte (GB)**, **Terabyte (TB)**, **Petabyte (PB)**, **Exabyte (EB)**, **Zettabyte (ZB)** e **Yottabyte (YB)**.

• **Lógica Binária**:

    ◦ Computadores operam com base em sinais elétricos com **apenas dois estados (ligado/desligado)**, representados por **0s e 1s (bits)**.

    ◦ Essa simplicidade reduz erros e facilita a construção de circuitos. Qualquer tipo de informação (números, letras, imagens) é convertida para uma sequência de bits.

**2. Fundamentos de Software**

• **Conceito de Software**: É o **conjunto de instruções, programas e dados** que faz qualquer dispositivo eletrônico funcionar. Se o hardware é o corpo, o software é a "mente" que diz ao hardware o que fazer. Sem software, o hardware seria inútil.

• **Tipos de Software**:

    ◦ **Software de Código Aberto**: O código-fonte é liberado ao público, permitindo estudo, modificação e distribuição. Exemplos: Linux, Firefox.

    ◦ **Software de Código Fechado**: O código-fonte não é divulgado; apenas o desenvolvedor original pode modificá-lo. Exemplos: Windows, Microsoft Office.

• **Software vs. Programa**: Um **programa** é um conjunto de instruções para uma tarefa específica. Um **software** completo pode ser composto por vários programas, bibliotecas e arquivos auxiliares que trabalham juntos para uma solução mais ampla (ex: Microsoft Office).

• **Sistema Operacional (SO)**:

    ◦ **Papel**: É o **elo entre o usuário e o hardware**, gerenciando recursos e fornecendo uma interface para interação. Sem ele, o equipamento não seria prático.

    ◦ **Funções Principais**: Gerenciamento de processos, gerenciamento de memória, gerenciamento de arquivos, gerenciamento de dispositivos (via "drivers"), e segurança/controle de acesso.

    ◦ **Tipos**: Windows, Linux, MacOS, Android, iOS, entre outros para diversos dispositivos.

• **Firmware e Middleware**:

    ◦ **Firmware**: Software especial gravado em chips, controlando **funções fundamentais de hardware** e responsável pelo processo de **inicialização (boot)** do dispositivo. É uma ponte entre hardware e software principal.

    ◦ **Middleware**: Software intermediário que atua como **ponte entre diferentes sistemas e aplicações** ou entre software e hardware. Ele integra e gerencia a comunicação em ambientes complexos (ex: servidores web, IoT).

• **Diferença Software e Hardware (Recapitulação)**:

    ◦ **Hardware é físico e tangível** (processador, teclado).

    ◦ **Software é lógico e intangível** (programas, instruções).

    ◦ Ambos são **interdependentes e essenciais** para o funcionamento de qualquer sistema computacional.

**3. Linguagens de Programação**

• **Linguagens Compiladas vs. Interpretadas**:

    ◦ **Linguagem Compilada**: O código-fonte é **traduzido completamente para o "idioma" do computador (código executável) *antes* da execução** por um compilador (ex: C, C++, Go, Rust). Vantagens: geralmente **superior em desempenho** e maior controle. Desvantagens: menor flexibilidade, dependência da plataforma.

    ◦ **Linguagem Interpretada**: O código-fonte é lido e **executado linha por linha *no momento da execução*** por um interpretador (ex: Python, JavaScript, PHP, Ruby). Vantagens: **maior facilidade para testar e modificar** durante o desenvolvimento, mesmo código em diferentes plataformas. Desvantagens: geralmente **inferior em desempenho**, dependência do interpretador, maior exposição do código-fonte.

• **Níveis de Linguagem de Programação**: Classificadas pela proximidade com o hardware:

    ◦ **Linguagens de Baixo Nível**: Muito próximas do hardware (ex: linguagem de máquina, Assembly). Trabalham diretamente com 0s e 1s ou operações específicas do processador. Oferecem **máximo controle sobre o hardware** e eficiência, mas são mais complexas para programar.

    ◦ **Linguagens de Alto Nível**: Mais próximas da lógica humana (ex: Python, Java, C#, JavaScript). Permitem comandos mais intuitivos, simplificando o desenvolvimento e tornando-o mais rápido e menos propenso a erros.

• **Paradigmas e Ambientes de Programação**:

    ◦ **Paradigmas de Programação**: São os **modelos ou estilos que orientam a estrutura e o desenho dos programas**. Ajuda o programador a escolher a melhor abordagem para resolver um problema. Exemplos incluem:

        ▪ **Paradigma Procedural (ou Estruturado)**: Sequência de instruções ou procedimentos em blocos claros (funções, rotinas).

        ▪ **Paradigma Orientado a Objetos**: Organiza o programa em "objetos" que representam entidades do mundo real, com características (atributos) e comportamentos (métodos).

        ▪ **Paradigma Imperativo**: Foca nas instruções que mudam o estado do programa, detalhando o passo a passo.

        ▪ **Paradigma Funcional**: Organiza o programa em funções matemáticas puras, sem efeitos colaterais.

    ◦ **IDEs (Ambientes de Desenvolvimento Integrado)**: **Ferramentas que reúnem em uma única interface** editor de texto, ferramentas de depuração (para corrigir erros), gerenciamento de arquivos e outros recursos inteligentes. Tornam o processo de programação mais rápido, produtivo e menos sujeito a erros (ex: Visual Studio Code, Eclipse, PyCharm)

Exercícios:

Interface gráfica do usuário, Texto, Aplicativo, Email

O conteúdo gerado por IA pode estar incorreto.

Interface gráfica do usuário, Texto, Aplicativo, Email

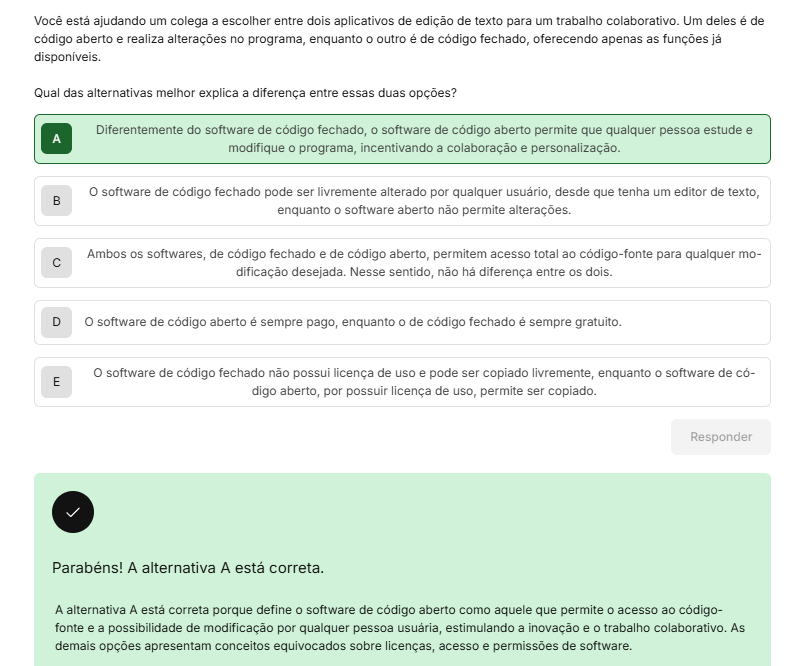
O conteúdo gerado por IA pode estar incorreto.

Interface gráfica do usuário, Texto, Aplicativo, Email

O conteúdo gerado por IA pode estar incorreto.

Interface gráfica do usuário, Texto, Aplicativo, Email

O conteúdo gerado por IA pode estar incorreto.



Interface gráfica do usuário, Texto, Aplicativo, Email

O conteúdo gerado por IA pode estar incorreto.

Interface gráfica do usuário, Texto, Aplicativo, Email

O conteúdo gerado por IA pode estar incorreto.

Interface gráfica do usuário, Texto, Aplicativo, Email

O conteúdo gerado por IA pode estar incorreto.

Interface gráfica do usuário, Texto, Aplicativo, Email

O conteúdo gerado por IA pode estar incorreto.

Interface gráfica do usuário, Texto, Aplicativo, Email

O conteúdo gerado por IA pode estar incorreto.

Interface gráfica do usuário, Texto, Aplicativo, Email

O conteúdo gerado por IA pode estar incorreto.

Interface gráfica do usuário, Texto, Aplicativo, Email

O conteúdo gerado por IA pode estar incorreto.

Interface gráfica do usuário, Texto, Aplicativo, Email

O conteúdo gerado por IA pode estar incorreto.

Interface gráfica do usuário, Aplicativo

O conteúdo gerado por IA pode estar incorreto.

Interface gráfica do usuário, Aplicativo

O conteúdo gerado por IA pode estar incorreto.

Interface gráfica do usuário, Texto, Aplicativo

O conteúdo gerado por IA pode estar incorreto.

Interface gráfica do usuário, Texto, Aplicativo

O conteúdo gerado por IA pode estar incorreto.