**PRIMEIROS PASSOS PARA COMEÇAR A PROGRAMAR:**

**AULA 01 - HISTÓRIA DA COMPUTAÇÃO:**

**1. APRESENTAÇÃO DO CURSO:**

**2. INTRODUÇÃO:**

**3. BERÇO DA COMPUTAÇÃO:**

-Alan Turing (1912 a 1954): pai da computação. Operacionalização do pensamento intuitivo humano por passo a passo e definiu o conceito de algoritmo. Computo para resolver problemas. Decifrou códigos de guerra em 1940. Codificador de voz. Inteligência Artificial em 1950. Codificação. Linguagem decimal para binária.

-Claude Shannon: Percebeu a aplicabilidade da álgebra booleana em computadores digitais, que são a base dos computadores atuais. Dissertou sobre teoria da informação.

- Von Newmann: primeira tentativa de computador.

**4. INSTRUMENTOS COMPUTACIONAIS:**

-Blaise Pascal: primeira máquina de operações simples.

- Charles Xavier Thomas: primeira máquina confiável.

\*Essas máquinas computacionais tinham componentes para serem considerados computadores, mas não eram programáveis, funcionavam pela inserção de números.

-Charles Babbage: protótipo de madeira e latão em 1822 e descreveu seu trabalho em 1837. Conceito de software, com diferentes funções e conceitos mutáveis.

-Ada Augusta Lovelace: primeira programadora. Instruções para cálculo analítico. Patrona da computação.

\*computador analítico de Babbage

-Herman Hollerith: máquinas de tabulação. Usada no censo americano de 1880, que reduziu o tempo de censo consideravelmente. Fitas e cartões perfuráveis. Criou a CTR que mais tarde se tornou a IBM.

**5. PRIMEIROS COMPUTADORES:**

-Konrad Zuze: Z1 com 100 válvulas; Z2 relés; Z3 relés mais núcleos de ferrites, destruído em um bombardeio na guerra.

\*IBM – automatic sequence controller calculator ou Harvard mark i: modificação de instruções dinâmicas e com unidade de decisão algorítmica. Com memórias separadas para instruções e dados.

\*Segunda guerra mundial.

\*eniac – formato em “u” com memórias gigantescas e 18 mil válvulas. Ele foi o primeiro programável.

\*edvac – primeiro com programa armazenado, 1954.

\*colossos – programado por fio, 1943.

**6. PRIMEIROS COMPUTADORES PESSOAIS:**

1975 – popular eletronics.

\*Criação da linguagem basic.

1976 – apple i

1977 – apple ii

1981 – imb-pc (open sorce, código aberto)

**7. CONSIDERAÇÕES SOBRE CIÊNCIA DA COMPUTAÇÃO:**

\*Historicamente: alto custo e programas de processamento de dados.

\*Surgimento da engenharia de software

\*Ciência da computação como matéria curricular.

\*Motivos para estudar a história: descobrir analogias, determinar paralelismos, padrões e reconhecimento de tendências que permitem prever algum dado futuro.

**AULA 02 - INTRODUÇÃO À LÓGICA DE PROGRAMAÇÃO:**

**1. O QUE É LÓGICA DE PROGRAMAÇÃO:**

\*Conceito de problema: é uma questão que foge a uma determinada regra, ou melhor, é um desvio de percurso, o qual impede de atingir um objetivo com eficiência e eficácia.

\*Conceito de lógica: parte da filosofia que trata das formas de pensamento em geral (dedução, indução, hipótese, inferência, etc.) e das operações intelectuais que visam à determinação do que é verdadeiro ou não.

-ordenação que segue convenções; organização coesa; forma como desencadeiam acontecimentos; forma de raciocínio.

\*Conceito de lógica aplicado a computação: organização e planejamento de instruções assertivas em um algoritmo, a fim de viabilizar a implantação de um programa.

**2. TÉCNICAS DE LÓGICA DE PROGRAMAÇÃO:**

\*Técnica Linear: modelo tradicional, em vínculo de estrutura hierárquica. Execução sequencial, recursos limitados, única dimensão.

\*Técnica estruturada: organização, disposição e ordem dos elementos essenciais que compõem um corpo (concreto ou abstrato). Estrutura hierárquica.

-Objetivo dessa técnica: facilitar a escrita, entendimento, validação e manutenção de programas. Muito usado no processamento de dados.

\*Técnica modular: decomposição de um problema em partes independentes para a sua solução. São controladas por um conjunto de regras.

-Modelo padrão: divisão geral em três partes: dados de entrada – processo de transformação – dados de saída.

-Metas dessa técnica: simplificação, decomposição do problema e verificação do módulo (independente).

**AULA 03 - ALGORITMOS:**

**1. ALGORITMO:**

\*Algoritmo: é rápido e eficaz, mas não opera sozinho e precisa de instruções detalhadas. É usado para processamento de dados por meio de programas com instruções. A linguagem precisa ser entendida por máquinas e humanos.

-É um processo de resolução de problemas passo a passo usando instruções. O que precisa ser feito e a ordem de execução. Sequência de passos com objetivo definido, execução de tarefas específicas e conjunto de operações que resultam em uma sucessão finita de ações.

**2. CONSTRUÇÃO DE ALGORITMOS:**

\*Como construir um algoritmo?

-compreensão do problema

-definição dados de entrada

-definir processamento

-definir dados de saída

-utilizar um método de construção

-teste e diagnóstico

-Ferramentas: narrativa (português), fluxograma (uso de símbolos predefinidos) e pseudocódigo (portugol).

**AULA 04 - FUNDAMENTOS DE ALGORITMOS:**

**1. TIPOLOGIA E VARIÁVEIS:**

\*Tipos de dados: numéricos (inteiros e reais); caractere (“”); lógico (booleano – verdadeiro ou falso)

\*variável (pode assumir qualquer um dos valores de um determinado conjunto de valores) e precisam ser previamente identificadas com sua tipologia

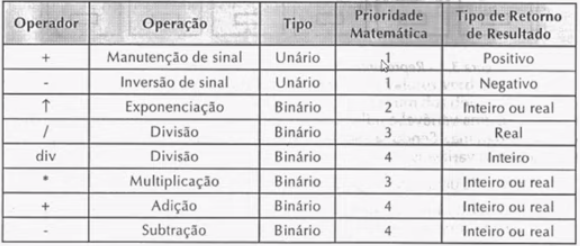
-regras: atribuição de um ou mais caracteres; primeira letra não pode ser número; sem espaços em branco; não pode usar palavras reservadas, podem usar caracteres e números.

-papéis: ação (modificação de estado) e controle (vigiada)

\*constante: é fixa, estável.

**2. INSTRUÇÕES PRIMITIVAS:**

\*operadores: binário e unário.



\*Instruções: é uma linguagem de palavras-chave (vocabulário) de uma determinada programação que tem por finalidade comandar um computador que irá tratar os dados.

-Cada linguagem de programação tem sua notação diferente. (janela, window, ventana).

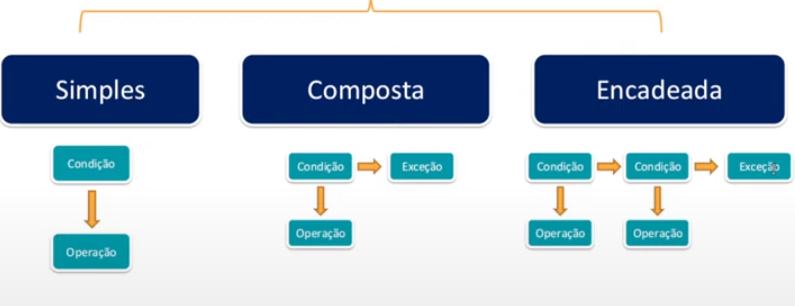
\*Outros conceitos:

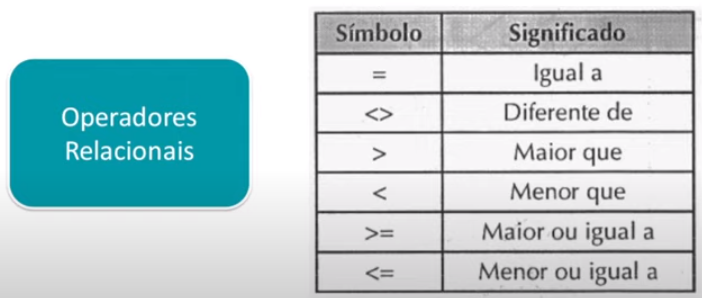
-entrada, processamento e saída.

**3. ESTRUTURAS CONDICIONAIS E OPERADORES:**

\*condição é o estado de uma pessoa ou coisa.

\*condicional é a verificação de uma condição, suposição ou hipótese.





\*operadores lógicos: and, or e not. usados para verificar v ou f, substituição e encadeamento de condições.

-and: todas as operações devem ser satisfeitas. (interseção)

-or: uma operação deve ser satisfeita. só dá falso se ambas forem falsas. (união)

-not: negação da condição, inversão do resultado lógico. ex.: a-b = not b

**4. ESTRUTURAS DE REPETIÇÃO:**

A estrutura de repetição é usada para executar um mesmo trecho de programa inúmeras vezes ou até um determinado número de vezes programado.

\*condições de parada: ou por número de repetições pré definidas ou uma condição a ser satisfeita.

\*benefícios: redução de linhas, compreensão facilitada e redução de erros.

-enquanto: início do teste lógico. repetições indefinidas

-repita: final do teste lógico, repetições indefinidas.

-para: início do teste lógico, repetições definidas.

\*É possível mesclar estruturas de repetição.

**5. VETORES E MATRIZES:**

\*vetores: um vetor é caracterizado por uma variável dimensionada com tamanho pré-fixado (matriz unidirecional)

\*matrizes: é uma tabela organizada em linhas e colunas no formato m x n; em que m representa o número de linhas horizontais e n o número de colunas na vertical. é também uma coleção de variáveis, contíguas em memória e indexadas.

-para as operações é necessário a definição do vetor, atribuição de valor por posição e leitura (escrever) por posição. ex.: matriz\_alunos [6][6].

**6. O QUE SÃO FUNÇÕES:**

\*Funções: as funções, ou subrotinas, são blocos de instruções que realizam tarefas específicas. É uma decomposição do algoritmo. Leva a uma modularização do problema.

-Definição formal de função: são blocos de instruções (código), identificados por nomes e parâmetros. A assinatura de uma função é seu nome e parâmetro.

-componentes da função: definição – nome – invocação – variável local (são destruídas ao encerrar a função).

-A função funciona pegando um conjunto de dados e gerando um resultado.

**7. INSTRUÇÕES DE ENTRADA E SAÍDA:**

-dados-processamento-resultado

\*entrada: inserção ou recebimento de dados do mundo real por meio de ação de alguma interface, seja teclado, mouse, arquivos, entre outros.

\*saída: consiste na impressão dos dados do mundo abstrato, digital por meio de ação de alguma interface. Os formatos podem variar de simples arquivos binários até complexas querys de banco de dados.

-dois tipos de saida: programada (condicional, que aguarda o dispositivo estar preparado para receber a informação ou incondicional, independente do estado do dispositivo) ou por interrupção (definida pelos periféricos)

-casos de entrada/saída: bem sucedida, com erro de sintaxe, com erros de programação, com problemas de interface.

**AULA 05 - LINGUAGEM DE PROGRAMAÇÃO:**

**1. REFERÊNCIAS BIBLIOGRÁFICAS:**

**2. INTRODUÇÃO À LINGUAGEM DE PROGRAMAÇÃO:**

\*problema computacional: objeto de discussão que possui instruções passo a passo que são mais facilmente resolvíveis em ambiente computacional.

\*problemas de decisão: caráter lógico, sim ou não é um problema decidível. Ex.: dado um número inteiro positivo, determine de n é primo.

\*problema de busca: relacionamento binário.Ex.: x está em a?

\*problemas de otimização: maximizar ou minimizar uma função.

\*O que é uma linguagem de programação? É um método padronizado composto por um conjunto de regras sintáticas e semânticas de implementação de um código fonte (conjunto de palavras com regras).

**3. COMO O COMPUTADOR ENTENDE O PROGRAMA:**

\*código fonte: pode ser traduzido ou interpretado.

\* tradução: geração do programa objeto e execução do programa objeto. (execução mais rápida)

-linguagem de alto nível (programa fonte) - compilador - assembly (programa objeto) - linguagem de máquina.

\*interpretação: o programa fonte é executado diretamente. (programas menores e mais flexibilidade).

**4. CARACTERÍSTICAS DE UM PROGRAMA:**

\*diretrizes do desenvolvimento de programas: legibilidade, redigibilidade, confiabilidade e custo.

-legibilidade: é um estado que é legível: facilidade de leitura, compreensão, ortogonalidade (coerência das instruções) e definição adequada das estruturas.

-redigibilidade: é a facilidade de escrita do código: pode ser confundido com a legibilidade, ortogonalidade, simplicidade de escrita, suporte à abstração, reuso de código, expressividade (operador ++, uso do for).

-confiabilidade: faz o que foi programado para fazer. possui verificação de tipos, trata de exceções, uso de ponteiros, tem compatibilidade entre compiladores.

-custo: análise de impacto. possui treinamento, codificação, compilação, execução e infra-estrutura.

\*Outras características:: atualizações, uso para IA, disponibilidade de ferramentas, comunidade ativa e adoção pelo mercado.

**5. ANÁLISES DE CÓDIGO:**

\*análises: léxica - sintática - semântica

-léxica: análise do conjunto de palavras dentro de determinado idioma. particionamento do código fonte em tokens, que são classificados (identificadores, palavras reservadas, números, strings) e são usados na etapa posterior que é a eliminação daquilo que não interessa para a execução do programa (espaços em branco, comentários, etc).

-sintática: é componente do sistema linguístico que interligam os constituintes da sentença, atribuindo-lhe uma estrutura (forma)

-semântica: é o estudo do significado. incide sobre a relação entre os significantes, como: palavras, frases, sinais e símbolos. um erro de semântica ocorre quando não faz o que é esperado. (conteúdo).

**6. PARADIGMAS DE PROGRAMAÇÃO:**

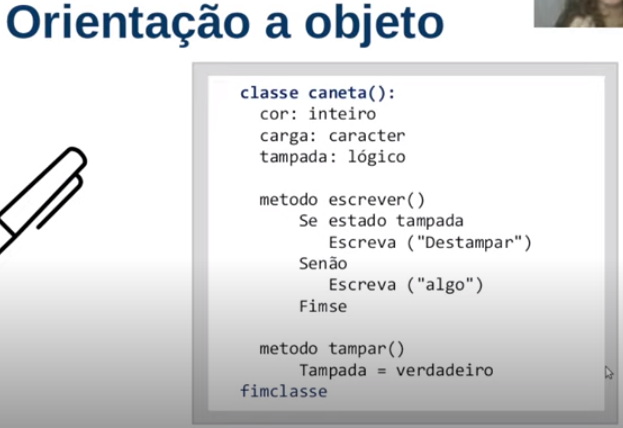
\*paradigma: é a forma de resolução de problemas com diretrizes e limitações específicas de cada paradigma utilizando linguagem de programação.

\*Classificação dos paradigmas: orientação à objeto (abstração que trata o programa com base em classes e objetos), procedural (chamadas sucessivas e procedimentos separados), funcional (instruções baseadas em funções), estruturado (estrutura de blocos alinhados), computação distribuída (possibilita funções executadas de forma independente), lógico.

\*paradigma estruturado (pe): ênfase em sequência. sequência: de instruções, decisão: testes lógicos, iteração: funções, laços, condições..

\*paradigma orientado a objeto (poo): tenta aproximar o cenário da programação ao mundo real, usando objetos e a interação entre eles.

-objeto é descrito por características específicas, comportamentos e estado. Ex.: caneta. As características representam o que tenho (atributo - modelo, cor), o que sou capaz de fazer (método - escrever, desenhar) e o como faço (estado - tampada, destampada, em uso).



\*classes: pode-se ter classes que recebem herança da outra mais geral (atributos e métodos). Classes mãe e filha.

\*comparativo poo x pe: poo com reuso de código, pe para problemas específicos e diretos.