**Primeiros Passos Para Começar a Programar** – Juliana Mascarenhas

**Modulo: Introdução à lógica de programação**

**Aula 1: O que é lógica em programação**

**- Logica é uma estrutura que visa solucionar problemas.**

**- 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 normal.**

- Logica: Definição formal: Parte da filosófica que trata das formas do pensamento em geral (dedução, indução, hipótese, inferência etc) e das operações intelectuais que visam a determinação do que é verdadeiro ou não.

**- Logica é a maneira de pensar**

**- Logica pode ser interpretada como uma organização coesa de pensamento afim de que se chegue a um objetivo; ordenação que segue convenções; forma de raciocínio; forma como desencadeiam acontecimentos.**

**- Logica em computação: Organização e planejamento das instruções assertivas em um algoritmo, a fim de viabilizar a implantação de um programa.**

**- Logica é uma sequência de instruções para a resolução de um problema.**

- O Processo de pensamento sequenciado é logica, seres humanos podem prever comportamentos de forma automática, sem nem racionalizar isso, os computadores não, estes precisam de ordem.

- A programação de um computador nos ensina a como pensar, a pensar de uma forma eficiente, deixando o seu pensamento objetivo.

**Aula 02: Técnicas de lógica de programação**

**- Logica é um processo de pensamento que nos permite determina as instruções e seguir essas instruções passo a passo para atingir o objetivo.**

**Técnica Linear:** **Modelo Tradicional; Não tem vínculo c/ Estrutura hierárquica ou programação de computadores.**

Linear: Homogêneo, unidirecional

**- Execução sequenciada de eventos. Ordenação de elementos por uma única propriedade.**

- Recursos limitados; Única dimensão

Ex.: Dormi, acorda, faz o café, toma o café. = Sequencia de eventos uma propriedade (vou acordar e tomar café

**Técnica Estruturada:** **Organização, disposição e ordem dos elementos essências que compõem um corpo (Concreto ou abstrato)**

**- Objetivo: Facilitar a Escrita; Entendimento; Validação; Manutenção para os Programas**

- Processamento de dados.

**Técnica Modular: Controlada por um conjunto de regras, partes independentes.**

**- Modelo padrão: Dados de entrada – Processo de Transformação – Dados de saída**

**- Metas: Simplificar; Decompor o problema; Verificação do módulo**

**Modulo: Algoritmos**

**Aula 01: O que é um algoritmo?**

- O computador é eficiente, rápido, e tem energia, mas não opera sozinho.

**- O computador precisa de instruções detalhadas**

**- O computador tem como objetivo o processamento de dados, o computador recebe, manipula e armazena dados.**

**- Algoritmo é um processo de resolução de problemas step by step (passo a passo) utilizando instruções**

- Instruções: O que precisa ser feito? Qual a ordem da execução?

- Desenvolvimento do Programa:

Analise: Estudo e definição dos dados de entrada e saída

**Algoritmo: Descreve o problema por meio de ferramentas narrativas, fluxograma ou pseudocódigo**

**Codificação: O algoritmo é codificado de acordo com a linguagem de programação escolhida**

- Algoritmos é uma sequencia de passos com objetivo definido; Execução de tarefas especificas; Conjunto de operações que resultam em uma sucessão finita de ações.

**- Algoritmos executar passo a passo para concluir a tarefa.**

**Aula 02: Construção de Algoritmos**

**- Como construir um algoritmo?**

**Compreensão do problema – Pontos mais importantes**

**Definição dados de entrada – Dados fornecidos e cenário**

**Definir processamento – Cálculos e Restrições**

**Definir dados de saída – Após processamento**

Utilizar um Método de Construção e Teste e Diagnostico – Construção e refinamento do algoritmo.

**- Construção de algoritmos:**

**Narrativa: Utilização da linguagem natural (Ponto positivo sem conceitos novos; ponto negativo diversas interpretações)**

**Fluxograma: Utilização de símbolos pré-definidos (Ponto positivo simples entendimento; necessário conhecimento prévio da estrutura e símbolos)**

**Pseudocódigo: Portugol (regras definidas, passos a serem seguidos)**

**Modulo: Fundamentos de Algoritmos**

**Aula 01: Tipologia e variáveis**

Variáveis:

**Qual a função do computador? Informação – Dados e Instruções**

**Dados: Numéricos; Caracteres, logico**

**Numéricos são os inteiros e reais**

**Caracteres: Símbolos, Números, Letras**

**Logico: Booleano – Verdadeiro 1 ou Falso 0 (Português estruturado: .Verdadeiro .V ou .S)**

Variável: pode assumir qualquer um dos valores de um determinado conjunto de valores (a-b=d; a + b = c) precisa ser previamente identificadas.

**Papeis de uma variável: ação (Modificação de estado); Controle**

**Aula 02: Instruções Primitivas**

**- Instruções primitivas: Operadores Binário e Unário**

**Aula 03: Estruturas condicionais e operadores**

**- Condição: um estado, uma pessoa**

**- Condicional: expressa uma condição ou suposição; contem ou implica uma suposição ou hipótese.**

- Estrutura Condicional:

Simples: Condição – Operação

Composta: Condição – Exceção e Operação

Encadeada: Condição – Operação – Condição – Operação – Exceção

Operadores Relacionais:

|  |  |
| --- | --- |
| Símbolo | Significado |
| = | Igual a |
| <> | Diferente de |
| > | Maior que |
| < | Menor que |
| >= | Maior ou Igual a |
| < | Menor ou igual a |

**- Operadores Lógicos: Relacionados a lógica booliana**

Quando utilizar? Verificação de V ou F; Substituição: encadeamento de condições

AND (E) – Operador Logico (Interseção: Ex.: O que está em A e B)

OR (OU) – (União)

NOT (Não) – Inversão do resultado logico; operador de negação

**Aula 04: Estruturas de Repetição**

**- Estrutura de repetição: Trecho de um programa: Controle de fluxo; Malhas de repetição; Loop; Repetição.**

- Condições de Parada para a estrutura de repetição: Números de repetições pré-fixada; Condição a ser satisfeita.

**- Redução de linhas; Compreensão facilitada; Redução de erro**

- Na codificação você faz testes utiliza a repetição para verificar e reduzir erros para entregar o código.

**Aula 05: Vetores e Matrizes**

**- Vetores: Um vetor é caracterizado por uma variável dimensionada com tamanho pré-fixado.**

- Matriz: é uma tabela organizada em linhas e colunas no formato m x n, onde m representa o numero de linhas (horizontal) e n o número de colunas (vertical)

**- Matriz é uma coleção de variáveis; contiguas em memória; índices**

**Aula 06: O que são funções?**

- Função: Bloco; Sub-rotina; Subalgoritimo;

**- Função: As funções são blocos de instruções que realizam tarefas específicas**

**- Função vem da decomposição do algoritmo**

- Função: Modularização do programa; Código mais claro e conciso; Reutilização de instruções

**- Funções: São blocos de instruções (Código) identificados por nomes e parâmetros**

**Aula 07: Instruções de Entrada/Saída**

- Dados – Processamento – Resultado

**- Entrada: Consiste na inserção e 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 desde simples arquivos binários até complexas querys de banco de dados.**

Saída programada: Condicional (Aguarda o dispositivo) e Incondicional

Saída por interrupção: Definida pelos periféricos

Casos de Saída: Bem sucedida; Erros de sintaxe ou outro; Erros de programação; Problemas com interface

**Modulo: Linguagem da Programação**

**Aula 01: Introdução a linguagens de programação**

**- 1949 surgiu Assembly (linguagem de baixo nível), linguagem de montagem, linguagem de maquina.**

- Década 50 surgiram Fortran; Cobol; Lisp

- Década 60 e 70 surgiram C; Prolog; ML

**- Linguagem de alto nível década de 90 Java; Python; Ruby; C#, Javascript**

- 2000 linguagem multiparadigmas Go; Rust; Dart; TypeScript

- Problemas computacionais: objetivo de discussão (problema) que possui instruções passo a passo (algoritmo) que são mais facilmente resolvíveis em ambiente computacional.

- Problemas de decisão: Caráter logico – Sim ou Não; Pertencimento – é possível de ser executado, decidivel.

- Problema de busca: Relacionamento binário

- Problema de otimização: Objetivo Maximizar ou Minimizar uma função

**- O que é uma linguagem de programação: Método padronizado composto por um conjunto de regras sintáticas e semânticas de implementação de um código fonte (conjunto de palavras com regras)**

**- Cada linguagem de programação tem suas regras.**

**Aula 02: Como um computador entende um programa?**

**- Código fonte pode ser traduzido ou interpretado.**

**- A maioria das linguagens de programação são de alto nível, o computador não consegue entender o comando de forma direta.**

- Processo de código de tradução: Um programa é um amontado de palavras se não for possível que o computador entenda. (Execução mais rápida)

Ex: Python (Linguagem de Alto nível) – Executa analise do programa (Compilador) – Assembly (Linguagem de máquina)

São 2 níveis para a tradução: 1 Geração do programa objeto; 2 Execução do programa objeto

- Processo de Interpretação: Programa Fonte executado diretamente (Programas menores; Maior flexibilidade)

**Aula 03: Características de um programa**

**- Desenvolvimento de programas seguem algumas diretrizes são elas: Legibilidade; Redigibilidade; Confiabilidade; Custo**

**- Legibilidade: Estado que é legível**

Facilidade de leitura; Compreensão; Ortogonalidade (coerência nas instruções); Definição adequadas das estruturas.

**- Redigibilidade: Facilidade de escrita de código**

Ortogonalidade; Simplicidade da escrita; Suporte a abstração; Reuso do código; Expressividade

**- Confiabilidade: Faz o que foi programado para fazer**

Verificação de tipos; trata exceções; uso de ponteiros; compatibilidade entre compiladores

**- Custo: Analise de impacto**

Treinamento; Codificação; compilação; execução; infraestrutura

**Outras Características: Atualizações; Uso para IA; Disponibilidade de ferramentas; Comunidade ativa; Adoção pelo mercado**

**Aula 04: Analise de Código**

**- Analise léxica: Particionar – Classificar – Eliminar**

Particionar vai verificar: Elementos (identificadores, palavras, reservadas) denominados tokens

Classificar vai eliminar elementos (caracteres de espaços)

**- Analise sintática: Sintaxe (forma): Componente do sistema linguístico que interligam os constituintes da sentença, atribuindo-lhe uma estrutura**.

Padrão-gramatica: Depende da linguagem de programação utilizada

**- Analise semântica: Semântica (significado) É o estudo do significado. Incide sobre a relação entre significantes, como: palavras, frases, sinais e símbolos**

Erro de semântica: o programa não faz o que é esperado

**Aula 05: Paradigmas de programação**

**- O que é paradigma? Forma de resolução de problemas com diretrizes e limitações especificas de cada paradigma utilizando linguagem de programação.**

- Paradigma: Orientação à objeto (Abstração que trata o programa com base em classe e objetos); Procedural (Chamadas sucessivas e procedimentos separados); Funcional (Instruções são baseadas em funções); Estruturado (Alinha o código em blocos); Computação distribuída (Funções executadas de forma independente); Logico

**- Paradigma estruturado: Sequencia; Decisão (Teste logico); Interação (Funções, laços)**

**Utilização: Problemas simples e diretos; Aprender programação**

**- Paradigma Orientação a objeto: paradigma de programação baseado na utilização de objetos e suas interações; análogo ao mundo real.**

Um objeto neste caso é descrito por características especificas, comportamentos e estado

**Critérios para classificar: O que tenho (atributos)/Sou capaz de fazer (métodos)/ Como faço (estados)**

Pilares de Orientação a Objeto: Herança; Encapsulamento; Polimorfismo; Abstração

**- Diferença entre Paradigma orientação a Objeto e Paradigma Estruturado é que o POO faz reuso do código e o PE é solução para problemas específicos e diretos.**