**Introdução a Programação e Pensamento Computacional**

**Pensamento computacional?**

Processo de pensamento envolvido na expressão de soluções em passos computacionais ou algoritmos que podem ser implementados no computador.

* Sistemático e eficiente tanto na formulação como na resolução de problemas
* Tem que ser utilizado de uma maneira que humanos & máquinas sejam capazes de resolver
* Não é uma disciplina acadêmica, sim, uma habilidade generalista. (Matemática, leitura e escrita)

**Baseado em 4 pilares:**

1. Decomposição: dividir um problema complexo em subproblemas.
2. Reconhecimento de padrões: Identificar padrões ou tendência.
3. Abstração: Extrapolar o conceito do problema para uma forma generalista
4. Design de algoritmos: Automatizar, definir passo a passo a solução do problema.

**Competências**

* Pensamento sistemático
* Colaboração dentro da equipe
* Criatividade e design
* Facilitador

**Habilidades complementares**

Raciocínio logico: é uma forma de pensamento estruturado, ou raciocínio, que permite encontrar a conclusão ou determinar a solução de um problema.

**Classificação**

Indução: geralmente vem apartir de um fenômeno observado, apartir dele conseguimos extrapolar e determinar, leis e teorias relacionados aquele fenomeno. Está relacionado a ciências experimentais, imperismo.

Dedução: Trabalho inverso da indução, onde a parte de leis e teorias você deduz previsões e explicações para aquele fenômeno. Utilizado em ciencias exatas.

Abdução: A partir de uma conclusão você tira a premissa, apartir de algo observado você supõe outra coisa. Exemplo: a grama está molhada, logo deve ter chovido. Entao você estabelece uma premissa de que choveu e a grama molhou, nesse caso a premissa não necessariamente será verdadeira. Esse tipo de técnica é muito utilizado no processo investigativo, diagnósticos.

**Aperfeiçoamento**

Melhoramento, Ato de aperfeiçoar, Aprimoramento, Refinamento. Todos sinônimos, mas com o mesmo objetivo. A partir de uma solução determinar pontos de melhora e refinamento.

Diagrama

Descrição gerada automaticamente

**Pilares: Decomposição**

Primeiro passo da resolução de problemas dentro do conceito de pensamento computacional.

“Dado um problema complexo, devemos quebrá-lo em problemas menores. Portando, problemas mais fáceis e gerenciáveis.”

**Padrões**

Reconhecimento de Padrões

* Modelo base
* Estrutura invariante
* Repetição

Similaridades e diferenças entre os contexto e objetos.

Por que determinar padrões?

Generalizar, com objetivo de obter resolução para problemas diferentes. Através de classes e categorias, onde a categoria vai depender do domínio e tipo de mídia.

Pelo ser humano

* Grau de similaridade
* Grupos conhecidos x objeto desconhecido

Como um computador reconhece padrões?

Comparação

* Representação de atributos
* Aprendizado – conceito associado ao objeto
* Armazenar dados
* Regras de decisão

A detecção de padrão ela vem da ideia de você extrair características, a fim de classificar os seus dados, é esse tipo de abordagem que a gente utiliza no reconhecimento de padrões. E você consegue utilizar diferentes métodos de reconhecimentos padrões e aplicar a diferentes contextos. Aplicações distintas utilizando métodos de reconhecimento de padrões.

Exemplos de aplicações:

* Classificação de dados
* Reconhecimento de imagem
* Reconhecimento de fala
* Análise de cenas
* Classificação de documentos

Algumas áreas que utilizam: Machine Learning, Redes Neurais, Ciência de dados, Inteligência Artificial.

**Abstração**

Generalização / Abstração

ABSTRAIR ABSTRAÇÃO Generalizar

Processo intelectual de isolamento de um objeto da realidade

Observar, um ou mais elementos, avaliando características e propriedades em separado

Torna-se geral, mais amplo, extensão.

Generalização, na lógica, é a operação intelectual que consiste em reunir numa classe geral, um conjunto de seres ou fenômenos similares.

Como classificar os dados?

* Características
* Pontos essenciais
* Generalizar x detalhar

Representação de dados

Características, identificar os pontos essenciais e descartar os detalhes.

Conceitos baseados em abstrações

A abstração é usada em diversas áreas de conhecimento, mas dentro da computação podemos numeras algumas questões.

* Merge sort: algoritmo de pesquisa e ordenação
* Clusteting: algoritmo para tornar o processo mais rápido, se algum cair outros assume.
* Busca binária

Estruturas / tipos abstratos de dados

* Árvore
* Lista
* Grafos

São estruturas que a gente cria e abstrai de tal maneira que elas não são palpáveis, conseguimos programas e cria-las. Mas, elas são abstrações de algo específico.

Máquinas de estado infinito, também é uma abstração. E a própria linguagem de programação de algo concreto do mundo real para algo abstrato.

**Algoritmos**

O computador tem / é

* Energia

PRECISA DE INSTRUÇÕES DETALHADAS

* Trabalhador

NÃO OPERA SOZINHO

* Eficiência
* Rapidez

Processamento de dados

O computador recebe, manipula e armazena dados.

Os programas por sua vez, são constituídos pelas instruções e elas possui o passo a passo do que o computador precisa realizar.

Processo de resolução de problemas “step by setp” Utilizando instruções.

O que precisa ser feito?

Qual a ordem de execução?

Desenvolvimento do programa

* Análise

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.

Sequência de passos com objetivos definidos

Execução de tarefas específicas

Conjunto de operações que resultam em uma sessão finita de ações

Exemplos:

* Preparar um sanduíche
* Trocar uma lâmpada
* Trajeto ao trabalho
* Fazer uma receita de bolo

Como construir um algoritmo?

* Compreensão do problema: quais são os pontos mais importante relacionados ao seu contexto
* Definição dados de entrada: dados fornecidos e cenário
* Definir o processamento: cálculo e restrições
* Definir data de saída: após o processamento
* Utilizar o método de construção: construção e refinamento do algoritmo
* Teste de diagnóstico

Construção de algoritmos

Narrativa: É a mais simples, não atribui com conceitos novos. Ela utiliza a linguagem natural, contudo ela pode ser ambígua e fornecer diversas interpretações dependendo de como foi escrito.

Fluxograma: ele é uma estrutura gráfica, onde você tem símbolos predefinidos, que definem qual é o tipo de ação que está sendo executada ali, se é uma operação ou print na tela, se é uma variável que está sendo alocada.

Pseudocódigo: Portugol, parte de regras bem definidas o que você tem passa passos a ser executado. Ele está muito próximo a codificação, mas não é uma linguagem de programação efetivamente, é só algo meio termo para que você possa se acostumar ao mundo da programação.

Algoritmo: É uma sequência de instruções passo a passo para que você possa resolver algum tipo de tarefa e consegui o objetivo final.

**Estudo de caso conceitual: perdido**

Como resolver o problema utilizando o pensamento computacional?

* Identificar mecanismos
* Recursos comuns
* Detalhes mais importantes

**Estudo de caso aplicado: soma de um intervalo**

Ex: soma de n° entre 1 e 200

200 + 1

199 + 2

198 + 3

197 + 4 ...

200 + 1 = 201

199 + 2 = 201

198 + 3 = 201

197 + 4 = 201 ...

O valor 200 vai se repetir quantas vezes?

200/2 = 100

Resultado 201 x 100 = 20.100

Expressar em variáveis

Ex: soma de n° entre x e y

Interface gráfica do usuário, Aplicativo

Descrição gerada automaticamente

Interface gráfica do usuário, Texto, Aplicativo, chat ou mensagem de texto

Descrição gerada automaticamente

Algoritmo

Passo 1 - Recebe os valores (x e y)

passo 2 – Resolva:

y/2 = total

Passo 3 – Resolva:

y+x = resultado\_parcial

Passo 4 – Ache o total

Final = total x resultado\_parcial

Passo 5 – Imprima o resultado

**Estudo de caso aplicado: Adivinhe um número**

Ex: adivinhe o número

O problema consiste em determinar o número escolhido por uma pessoa dentro de um intervalo.

**Introdução à lógica de programação**

O que é lógica?

A lógica vem para solucionar a resolução de algum problema, e o problema é uma questão, uma proposta duvidosa, um objeto de discussão e pode ter aí numerosas soluções.

Definição formal

“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”.

Definição formal de lógica

“Parte da filosofia 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.”

Lógica

Seria uma organização coesa, uma forma de raciocínio, onde nós conseguiríamos ordenar a moderação, que segue as contrações e teríamos uma forma de desencadear acontecimentos. Então ela oferece uma ordenação, um sentido para o que é verdadeiro ou não, uma ordem para um determinado contexto/situação.

“Organização e planejamento das instruções, assertivas em um algoritmo, a fim de viabilizar a implantação de um programa.”

**Técnicas de lógica de programação**

Técnica Linear

* Modelo tradicional
* Não tem vínculo
* Possui estrutura hierárquica
* É utilizada na Programação de computadores, podemos entender essa técnica como modelo de desenvolvimento e resolução de problemas.
* É uma técnica associada a matemática pela sua característica linear.

Podemos entender que: ela é a execução sequenciada de uma série de operações, a ordenação de elementos por uma única propriedade, por exemplo, onde nós temos recursos limitados em uma única dimensão.

Técnica Estruturada

Organização, disposição e ordem dos elementos essenciais que compõem um corpo (concreto ou abstrato).

Dado o processamento de dados, temos como objetivo a técnica estruturada à parte da escrita de programas, entendimento, validação e manutenção(facilitar).

Técnica Modular

Onde definimos parte independentes que são controladas por um conjunto de regras e cada modulo tem seu conjunto de regras específicos.

Esse é o modelo padrão de uma técnica modular:

Texto

Descrição gerada automaticamente

Conseguimos com isso: a simplificação do algoritmo, resolução do problema, podemos decompor o problema em problemas menores, e assim fazer a verificação por módulo. Ao invés de sair verificando todo o código, de repente se mexer em uma parte do código vai alterar outra e acrescentar erros de inconsistências, podemos concentrar as alterações dentro de um modulo específico, aquele que estou querendo modificar.

**Fundamentos de algoritmos**

Tipologia e variáveis

Qual a função do computador?

Processar as informações que nós passamos para ele, e as informações são compostas por 2 tipos de conteúdo, a primeira são os dados, são dados puros, objeto de manipulação e de processamento de um computador. E a instruções são as normativas e diretivas utilizadas pelo computador para executar determinadas ações que irão executar aqueles dados.

Dados são tratados e processados.

Tipos existentes

* Numéricos: Os números inteiros são todos números positivos ou negativos que não possuem casa decimais.

0, 1, 5, 60, 800... -58, 50, 49, 32, -10, -5.

Os Números reais são todos que compõe o nosso expecto numérico positivo e negativo com casa decimais.

5.95, 9.54, -8.8, -0.555... 0, 1, 5, 50, 60, 800, -58, -50, -49, 32, -10 -5

* Caracteres: Tudo aquilo que não representamos como número, até o próprio número pode ser um caractere. As letras têm uma representação especifica com aspas duplas, o caractere tem um limite de tamanho.
* Lógicos: Booleano, o lógico, dentro da lógica booleana temos apenas verdadeiro ou falso.

O que é uma variável?

Uma variável é um tipo de estrutura mutável, que pode variar dentro do seu valor e inconstante, podendo receber mais de um valor e modificar o conteúdo. Ela irá receber um valor, não sabe qual, mas, sabe o tipo daquele dado.

A variável ela pode assumir qualquer um dos valores de um determinado conjunto de valores, contudo ela esta restrita ao seu tipo, se é numérica recebe numérica, se é string recebe string.

Nome da variável, boas práticas:

* Atribuição de um ou mais caracteres
* Primeira letra - não número
* Sem espaços em brancos
* Vedado

Utilização de palavras reservadas

* Caracteres e números

Papéis de uma variável

* Ação, quando ela modifica o estado de algoritmo ou programa.
* Controle, quando é vigiada e utilizada para controle de alguma estrutura ou então dentro de alguma equação.

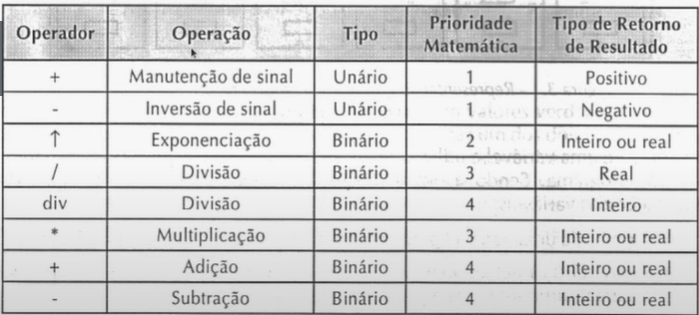
Definição de constante

Tudo aquilo que é fixo ou estável. A ideia da constante é ser inalterável diferente da variável que pode sim variar.

**Instruções primitivas**

As instruções elas vão determinar as ações que nós iremos executar em cima dos nossos dados, geralmente cálculo matemático, para isso utilizamos os operadores. E dentro do cálculo utilizamos como informação/input as variáveis e constantes.

Os operadores podem ser tanto binários como unário:



Definição formal de instrução

Instruções são linguagens de **palavras chaves**(vocabulário) de uma determinada de programação que tem por finalidade comandar um computador que irá **tratar os dados.**

Entrada, processamento e saída de dados

Esses dados são importados de algum lugar, arquivo ou diretório. Levados AO ambiente computacional para serem processados e imprimidos de alguma forma, em tela ou dispositivo de saída.

**Estrutura condicionais e operadores**

Dado o estado de uma pessoa ou coisa existe uma condição para aquilo acontecer, o condicional expressa uma condição ou uma suposição, ele contém ou implica uma suposição ou hipótese.

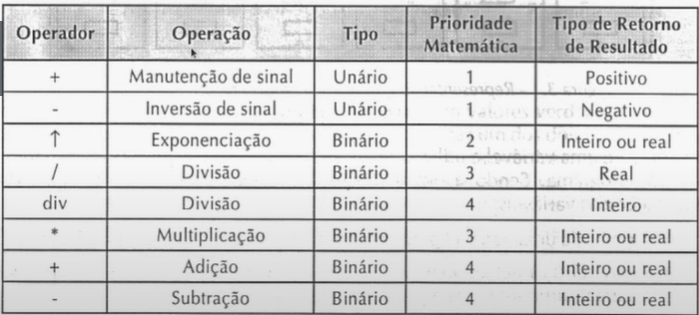
Qual a ideia de uma estrutura condicional?

Dado uma condição que foi satisfeita eu executo uma determinada operação, essa condição sendo satisfeita a operação é executada. Caso sendo uma inverdade e a condição não sendo satisfeita acarreta uma exceção.

Estruturas

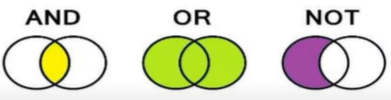
* Simples: apenas verificamos se uma condição foi satisfeita.
* Composta: se a condição não for satisfeita, eu jogo uma exceção.
* Encadeada: si, e, si, não, uma sucessão de estruturas condicionais

Operadores Relacionais



Todos esses operadores são utilizados em diferentes estruturas condicionais, posso está verificar uma determinada ação e se ela for satisfeita ( a condição for verdadeira) ele executa à ação.

Operadores lógicos



Quando utilizar?

* Quando precisa de uma resposta simplificada, de V ou F
* Substituição

encadeamento de condições

AND (e) - Verifica as entradas que devem ser satisfeitas, ele sempre é falso se for um ou outro, e verdadeiro quando os 2 são satisfeitos.

Interseção

Ex: Curso de inglês

Se (gramática.e. conversacao) então

escreva" aprovado”

senão

escreva” reprovado”

OR (ou) – Se ambas as condições forem verdadeiras encaminho verdadeiro, preciso que apenas uma seja verdadeira. Se ambas forem falsas, encaminho falsos.

NOT - operador de negação onde a inversão do resultado lógico, é o resultado dessa operação. Se eu tenho uma condição que é verdadeira ela se torna falso, e a falsa se torna verdadeira.

**Estrutura de repetição**

Uma estrutura de repetição irá executar um trecho de um programa a partir de certos parâmetros estabelecidos dentro desta estrutura de repetição.

Condições de parada

* Número de repetições pré-fixada
* condição a ser satisfeita

Qual a vantagem de você utilizar uma estrutura de repetição?

* Redução de linhas
* Compreensão facilitada
* Redução de erro

Tem mais de um tipo de estrutura de repetição?

Enquanto... faça

Repita... até

Para... de... até... faça

**Vetores e matrizes**

Definição formal

“Um vetor é caracterizado por uma variável dimensionada com tamanho pré-fixado.”

Ou seja, o vetor é uma variável que possui uma sequência, um tamanho pré-fixado que irá receber N valores. Ele também pode ser encarado como um contêiner, ou também visto como matriz unidimensional.

Definição formal

“Matriz é uma tabela organizada em linhas e colunas no form ato m x n, onde m representa o número de linhas (horizontal) e n o agora o dormiu número de colunas (vertical)”.

As matrizes são uma coleção de variáveis, uma condição de vetores. Elas são contiguas em memória, ou seja, elas estão armazenadas juntamente dentro da memória do computador. Temos o índice, para podermos estar pesquisando e puxando informações dentro dessas matrizes.

**O que são funções?**

As funções são conhecidas como: Subprograma, Subalgoritmo, Sub-rotina, Método (muito dentro do conceito de programação orientada a objeto), Bloco. Todos são sinônimos e querem trazer a mesma ideia de função.

Uma função vai implicar que um conjunto de elemento A, esteja conectado a um conjunto de elemento B. Então a função a partir de um argumento que ela recebe te retorna um valor.

Definição formal

As funções, ou sub-rotinas são blocos de instruções que realizam tarefas específicas. Modularização do problema.

Função? Não é mais fácil fazer tudo junto?

* Modularização do programa
* Código mais Claro e conciso
* Reutilização de instruções

Definição formal

“São blocos de instruções (código), identificados por **nomes e parâmetros**”. Assinatura da função.

Atrelados a uma função

* Definição, objetivo que ela possui.
* Nome
* Invocação
* Variável local, são utilizadas apenas dentro da função e destruídas no momento que a função retorna o resultado para o problema principal.

**Instruções de entrada e saída**

Dado que eu tenho um conjunto de dados, o objetivo do meu computador é armazenar, processar e tratar esses dados. Através de um determinado algoritmo, uma determinada função, dentro do contexto que eu desejo ele exprime os resultados.

Definição formal Entrada

consiste na inserção e recebimento de dados do mundo real por meio de ação de alguma interface, seja teclado, mouse, arquivos, entre outros.

Definição formal 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.

Existem 2 tipos de saída dentro de um programa (algoritmo)

* Saída programada, ela pode ser condicional ou incondicional. Condicional: Aguarda o dispositivo para poder acionar a saída e disparar a condição de saída para imprimir os valores.
* Saída por interrupção, geralmente definida pelos periféricos. Aconteceu alguma coisa dentro do ambiente computacional pelos periféricos que interrompeu a execução e imprimiu uma saída.

Para toda saída existem alguns casos:

* Bem-sucedida
* Erro de sintaxe ou outro
* Erros de programação
* Problemas com a interface

**Introdução a linguagens de programação**

Introdução

* Compreender as dificuldades enfrentadas;
* Fundamentos da computação;
* O processo de pensamento;

Então a gente percebe pela história que a base / base da tecnologia, é baseada em pesquisas anteriores, baseados em lições aprendidas e geram novos paradigmas ao decorrer do tempo. Algo interesse de se pontuar é que toda evolução tecnológica se inicia pelo hardware e depois software.

Assembly

Em 1949 surgiu a primeira linguagem, Assembly (classificação correta para o Assembly é linguagem de máquina). É uma linguagem de montagem/máquina.

Nos anos 50 surgiu o Cobol, Fortran, Lisp, etc.

Entre 60 e 70 surgiram as linguagens de programação, os paradigmas, como Linguagem C, Prolog e ML.

Na década de 90 começaram a surgir as linguagens com mais alto nível como, Python, C#, Ruby, Java e Javascript.

Em 2000 novos conceitos foram surgindo como Swift, TypeScript etc.

Problemas computacionais

Objetos de discussão que possui instruções passo a passo que são mais facilmente resolvíveis em ambiente computacional.

Problemas de decisão

São de caráter lógico, onde devemos responder se sim ou não.

Ex: “Dado um número n inteiro positivo, determine se n é primo”.

Problemas de busca

* Relacionamento binário
* Objetivo
* Semelhante ao nome
* x está em A?
* Recorrente em teorias de grafos
* Ex: clique

Problemas 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.**

Código fonte, pode ser traduzido ou interpretado.

**Como um computador entende o programa?**

Um programa é um amontado de palavras senão for possível que o computador entenda.

Existe o processo de tradução que esta ligado a compilação.

Dado a minha linguagem de alto nível, ela vai ser enviada para o compilador que vai executar a análise do programa, e esse compilador vai traduzir o meu programa de alto nível em um código de baixo nível, de linguagem de máquina, como por exemplo o Assembly.

Código de baixo nível: Programa Objeto

Programa Fonte, é o programa que vai ser enviado ao compilador para que o computador possa entendê-lo

Tradução: (1) Geração do programa objeto

(2) Execução do programa objeto

Execução mais rápida e gera programas menores.

C++ e Java

Interpretação: Programa fonte executado diretamente

Maior flexibilidade, mais fácil de se programar só que é mais lenta.

JS, Ruby e Python

**Características de um programa**

Desenvolvimento de programas, diretrizes:

Legibilidade

* Facilidade leitura
* Compreensão
* Ortogonalidade
* Definição adequada das estruturas

Coerência nas instruções, que o código seja coerente, a partir da legibilidade a gente define quais são as estruturas mais adequadas para aquele contexto. Então legibilidade, estado que é legível

Redigibilidade

* Pode conflitar com a legibilidade
* Ortogonalidade
* Simplicidade de escrita
* Suporte a abstração
* Reuso do código
* Expressividade

Redigibilidade, questão da facilidade de escrita de código. Utilizar técnicas que deixam o código específico, como operador ++ ou for.

Confiabilidade, faz o que foi programado para fazer.

* Possui:
* Verificação de tipos
* Trata exceções
* Uso de ponteiros
* Compatibilidade entre compiladores

Custo, análise de impacto.

* Treinamento
* Codificação
* Compilação
* Execução
* Infraestrutura

Características

Outras características:

* Atualizações
* Uso para IA
* Disponibilidade de ferramentas
* Comunidade ativa
* Adoção pelo mercado

**Análise de código**

Análise léxica

* Particionar:

Ao particionar ela vai identificar os elementos denominados tokens. E vai agrupá-los.

Elementos: Identificadores, palavras reservadas, números, strings. Todo conteúdo que é relevante para codificação do nosso programa.

E ela vai eliminar elementos: Caracteres de espaços em brancos, comentários...

* Classificar
* Eliminar

Análise sintática

Sintaxe: “Componente do sistema linguístico que interligam os constituintes da sentença, atribuindo-lhe uma estrutura”. Análise sintática define a corretude do programa.

Padrão – gramática

* Depende da linguagem de programação utilizada.

Análise semântica

Significado

Semântica: “É o estudo do significado. Incide sobre a relação entre significantes, como: palavras, frases, sinais e símbolos”. É a lógica do programa.

Uma imagem contendo Texto

Descrição gerada automaticamenteErro de semântica:

* Não faz o que é esperado

Uma imagem contendo Diagrama

Descrição gerada automaticamenteO correto seria:

**Paradigmas de programação**

O que é um paradigma?

Definição formal: “Forma de resolução de problemas com diretrizes e limitações especificas de cada paradigma utilizando linguagem de programação.”

Classificação

* Orientação a objetos (Python, lua, Java, C++)
* Procedural: chamadas sucessivas e procedimentos separados (Fortran, Lua)
* Funcional: instruções são baseadas em funções (JS, Python, Lua, Java)
* Estruturado: estrutura de blocos alinhados (C.C++, JS, Java)
* Computação distribuída: funções executadas de forma independente
* Lógico (Java)

Paradigma Estruturado

Conceitos:

* Sequência, ênfase em sequência
* Decisão, relacionado a um teste lógico.
* Iteração, relacionado a funções, laços e condições.

Utilização:

* Problemas simples e diretos
* Aprender programação

POO ( programação orientada a objetos) ainda não é compreendida por muitos

Função em C:

Texto

Descrição gerada automaticamente com confiança baixa

Orientação a objetos

Paradigma de programação baseado na utilização de objetos e suas interações. Análogo ao mundo real

O que é um objeto?

“Um objeto é descrito por características (o que eu tenho) especificas, comportamentos (o que sou capaz de fazer) e estado (como faço)”.

Ponto de vista da programação

Classe POO

* Alocação em memória
* Operações associadas

Estruturada

* Alocação em memória

Variável

* Operações associadas

Pilares de Orientação a Objeto

* Herança: uma classe filha herda as características de uma classe mãe.
* Encapsulamento
* Polimorfismo
* Abstração

POO: Reuso de código

PE: Problemas específicos e diretos