Bootcamp Java 16,17,18,19/09/22

Primeiros passos para começar a programa

Objetivo Geral:

Capacidade de entender o que significa pensar computacionalmente.

Aprender a resolver problemas de maneira mais eficaz

Percurso

**Aula 1:Pensamento computacional**

Objetivo: apresentar os conceitos que caracterizam o pensamento computacional, permitindo ao DEV entender o que isso significa

Percurso:

E introdução ao pensamento computacional

Habilidades complementares

Pilares: decomposição

Pilar de padrões

Abstração e algoritmos

O que é pensamento computacional

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

Na verdade, ele é um processo sistemático e eficiente para formular e resolver problemas. Ela deve ser resolvível pela máquina e pelo ser humano.

Ela na verdade é uma habilidade generalista, que pode ser utilizada em todo âmbito da vida.

Como exemplo: matemática, leitura e escrita.

Ele é baseado em 4 pilares

* Decomposição
* Reconhecimento de padrões
* Abstração
* Design de algoritmos

A decomposição basicamente é pegar o problema e o dividir em problemas menores.

Como exemplo do dia a dia: receita de bolo. Temos a massa, cobertura e recheio

Na hora de montar há uma ordem, porém ele é dividido em problemas menores.

Reconhecimento de padrões é identificar coisas que se repetem, as similaridades e diferenças entre os problemas.

Abstração

Extrapolar o conceito de um problema para uma forma generalista.

Criar parâmetros para a informação e generalizar. Como exemplo fórmula da circunferência já que ela se aplica a qualquer situação

Design de algoritmos

Automatizar os processos, definindo o passo a passo da solução desse problema.

Exemplo: criar um software que faz uma soma

Decompor a soma em receber 2 valores, reconhecer o padrão de que podemos somar qualquer valor. Abstrair a soma a qualquer valor e não somente aos pedidos e por fim criar o programa.

Processo contínuo

* Definir uma solução
* Testar a solução
* Aperfeiçoar a solução encontrada

Uma imagem contendo Seta

Descrição gerada automaticamente

Ele usa os dois lados, tanto as habilidades humanas com a resolução de problemas e a expressão da solução e a parte do computador que é a execução da solução e avaliação (refinamento)

Competências Ganhas

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

Habilidades complementares

Raciocínio lógico, aperfeiçoamento.

Raciocínio lógico forma de pensamento estruturado que permite encontrar a conclusão ou determinar a resolução de um problema.

Habilidade que necessita de treinamento, não somente saber ela.

Indução: vêm de um fenômeno observado, criando leis e teorias.

Dedução: previsão e explicação com base em leis e teorias.

Abdução: a partir de uma conclusão se tira uma premissa. Estabelecendo algo a partir de outra coisa. Podendo ela ser verdadeira ou não.

Aperfeiçoamento

Encontrar uma solução eficiente, otimizar processos (melhor uso dos recursos)

Simplificar linhas de códigos, funções bem definidas (melhorar códigos e algoritmos)

Decomposição

Dado um grande problema, devemos o dividir em problemas menores.

Estratégia análise: processo de quebrar e determinar partes menores e gerenciáveis, através de uma análise.

Estratégia síntese: combinar elementos recompondo o problema original, ou seja, desmembrar ele e juntar novamente para ter um sentido.

Ordem de execução de tarefas

Sequencial: quando há dependência precisamos resolver um problema para depois outro. Como exemplo quando uma função precisa de outra para funcionar

Paralelo: fazer mais de uma sem ordem específica.

Não basta aplicar, temos que desenvolver essa decomposição por nós mesmos.

Como decompor?

Identificar e coletar os dados, agregar os dados e por fim entregar a funcionalidade.

Exemplo: cozinhar

* Identificar os ingredientes
* Determinar as etapas (sequencial ou paralelo)
* Executar cada etapa
* Agregar os ingredientes para finalizar (recompor com coerência)

Exemplo: criar um app

* Finalidade
* Interface
* Funcionalidades
* Pré-requisitos

Padrões

Modelo base, estrutura invariante e repetição

Por que determinar padrões: generalizar com objetivo de obter resolução para problemas diferentes.

Podemos separar os padrões em categorias. Com o grau de similaridade, coisas que se parecem, são semelhantes.

A máquina só pode analisar padrões representando atributos, ensinando um conceito associado ao objeto

Armazenar os dados e por fim dar as regras de decisão.

Os padrões vêm da extração de características e classificação dos dados, criando métodos e aplicando os reconhecimentos dos padrões.

Abstração

Abstrair: observar, um ou mais elementos, avaliando características e propriedades em separado

Abstração: processo intelectual de isolamento de um objeto da realidade

Generalizar: tornar geral, mais amplo, estender.

A generalização, na lógica, é a operação 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

Ex: estudantes

Características:

Nome, matrícula, endereço...

Verificar os pontos essenciais, e deixar os detalhes que não serão usados de lado.

Algoritmos

Processamento dos dados: o computador recebe, manipular e armazenar os dados.

Os programas têm instruções que dão ao sistema, e os algoritmos são um processo de resolução de problemas passo a passo.

O que precisa ser feito e qual a ordem de execução das instruções

Ele deve ser entendido pelo humano e pela máquina

Desenvolvimento de programa

* Análise: estudo e definição dos dados de entrada e saída
* Algoritimo: descreve o problema por meio de ferramentas narrativas, fluxograma ou pseudocódigo
* Codificação: o uso de linguagem de programação para codificar o algoritimo

São instruções executadas passo a passo para concluir uma tarefa

Como construir um algoritimo

* Compreender o problema
* Definir os dados de entrada e saída
* O processamento dos dados
* Os dados de saída
* Utilizar um método de construção
* E por fim fazer o teste e diagnósticos.

Narrativa: sem novos conceitos, usando a linguagem natural. Narrar o passo a passo, podendo ser ambígua

Fluxograma: estrutura gráfica com símbolos pré-definidos que demonstram o tipo de ação aparecendo na tela.

Portugol: parecido com uma linguagem de programação, e usa a linguagem nativa para programar

Ex fluxograma:

Diagrama

Descrição gerada automaticamente

Estudo de caso

Soma de nº entre 1 e 200

Em vez de somar uma a um sendo menos eficiente podemos começar no último e ir decrementando tipo 200,199,198. E somar com o primeiro crescendo 1,2,3.

Ex:

200+1 = 201

199+2= 201

198+3= 201

Assim é feita a decomposição, e o padrão da soma.

Verificar quantas vezes se repete.

São 2 números a cada soma, total 200. 200/2 = 100 vezes ele se repete

Resultado 201\*100 20.100

Trazendo para código

X,y: são os valores

Y+x = resultado parcial

(y-1) + (x+1) = resultado parcial

y/2 = total

total\*resultado parcial = resultado

Soma de n

Aula 2

Introdução a lógica de programação

O que é logica?

Problema: desvio de percurso para atingir o que eu quero.

Logica: formas de pensamento em geral, e das operações intelectuais que visam a determinação do que é verdade ou não

Forma como desencadeiam acontecimentos, organizar de forma coesa, formar raciocínio a partir de uma ordenação

Resumindo: organização e planejamento das instruções em um algoritimo, a fim de viabilizar a implantação de programa.

Seres humanos podem prever comportamentos, computadores não. Eles precisam de instruções detalhadas para funcionar.

Técnicas de lógica de programação

Técnica linear

* Modelo tradicional
* Não tem vínculo, estrutura hierárquica, programação de computadores
* Muito relacionada a matemática
* Execução sequenciada de elementos, recursos limitados e uma única dimensão.

Técnica estruturada

* Organização, disposição e ordem dos elementos essenciais que compõem um corpo.
* Não é linear, podemos fazer escolhas.

Técnica modular

* Partes independentes, controlado por um conjunto de regras.
* Cada módulo possui sua própria regra.
* Simplificar o algoritmo, decompor o problema e verificar por módulo. (verificar um módulo em vez do código todo)

Aula 3

Fundamentos de Algoritmos

Objetivo: conceitos básicos como variáveis, tipos de dados, instruções, condições, entre outros temas relacionados.

Tipologia e variáveis

A função do computador: processar as informações dadas a eles, em dados e instruções.

Os dados são tratados e processados

Tipos de dados:

Numéricos: inteiros e reais

Caractere: tudo aquilo que não é representado como número, letras, caracteres especiais.

Lógico: verdadeiro-1, falso-0. Que chamamos de booleano.

Variável

Tipo de estrutura mutável, que varia, não é constante. Que pode receber qualquer valor

Somente sabemos o tipo do dado, não seu valor inserido.

Ela fica restrita ao seu tipo.

A variável vai identificar o determinado conteúdo.

Regras para nome

* Atribuição de um ou mais caracteres
* Primeira letra- não número
* Sem espaço em branco
* Vedado (não pode ter palavras reservadas)
* Com sentido
* Pode ter caracteres e números

Papéis de uma variável

Ação: modifica o estado do programa

Controle: vigiada, controla determinado algoritmo do programa.

Instruções primitivas

Operadores: fazem os cálculos matemáticos

Dentro desses cálculos usamos variáveis e constantes.

Tabela

Descrição gerada automaticamente

Instrução: faz uma determinada ação com um dado, que faz o tratamento deles.

As instruções primitivas mudam de acordo com a linguagem, no caso isso é a sintaxe

Os dados entram no computador, são processado e são retornados de alguma maneira

Estruturas condicionais e operadores

Condição: estado de uma pessoa ou coisa

Condicional: expressão que faz uma coisa caso a condição é satisfeita

Estrutura simples: condição e operação

Estrutura composta: condição, operação e exceção (não for satisfeita)

Estrutura encadeada: condição e operação, condição e operação e exceção

Operadores relacionais

Tabela

Descrição gerada automaticamente

Estrutura condicional

Condicional simples: condição e operação

Condicional composta: condição e operação caso verdadeiro e uma exceção caso falso.

Condicional encadeado: condição e operação caso uma condição seja verdade, caso não testa outra condição e se ela for verdade faz outra operação e caso as duas sejam falsas é feita a exceção.

Operadores lógicos: and, or e not.

Quando utilizar

* Verficação de v ou f
* Substituições

Operador E

Retorna verdadeiro se as duas condições sejam atendidas

Operador Ou

Retorna verdadeiro caso uma das condições seja verdadeira

Se nenhuma for ele retorna falso

Operador not

Inversão do resultado lógico se ela for verdadeira retorna falso e vice-versa.

Estruturas de Repetição.

Podem ser chamadas de laços, controles de fluxo, malhas de repetição, repetição ou loop.

É preciso de uma condição de parada

* Número de repetições
* Ou uma condição ser feita

Vantagens

Redução de linhas

Compreensão facilidade e redução de erros.

Enquanto

Enquanto uma condição for cumprida repita as ações.

Repita

Teste lógico é no final, é feita a instrução e depois verifica se ela cumpre a condição

Para

Teste feito no início, e número de repetições definidas.

Vetores e matrizes.

Vetor: variável dimensionada com tamanho pré-fixado.

Ela é unidimensional, ou visto como uma matriz unidimensional.

Matrizes são como uma junção de vetores, várias linhas e colunas.

Ambas necessitam de um índice para encontrar seus valores.

Funções

Blocos de instruções que realizam tarefas específicas, São blocos de instruções, identificados por nomes e parâmetros

Vantagens

Modularização do programa

Código mais claro e conciso

Reutilização de código.

Definição, nome, invocação e variável local

O nome que a função vai ter

O que ela vai invocar de fora como parâmetros

E as variáveis locais dela. Que são destruídas ao final da função

Instruções de entrada e saída.

Entrada

Quais os tipos de dados que irão entrar e como eles serão mostrados.

Inserção e recebimento de dados.

Saída

Impressão dos dados por meio de ação de alguma interface.

Existem dois tipos de saída

Programada: pode ou não ser condicional, e aguarda o dispositivo.

Interrupção: definida por periféricos

Casos

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

Aula 4

Linguagens de Programação

Objetivo: apresentar os paradigmas de programação, o conceito. As linguagens são baseadas em um ou mais paradigmas.

Introdução

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

As evoluções tecnologias evoluem primeiro no hardware e depois no software.

Primeira linguagem de programação

* 1949
* Assembly
* Primeira linguagem
* Linguagem de montagem, máquina.

nos anos 50 temos as primeiras linguagens

como cobol, Fortran e lisp

entre 60 e 70 temos os paradigmas como C, prolog e machine learning

década de 90 linguagens de alto nível.

Problemas computacionais: eles são recorrentes, que na verdade é um objeto de discussão que possui instruções passo a passo que são mais facilmente resolvíveis em um ambiente computacional.

Problema de otimização, busca e decisão

Problemas de decisão: caráter lógico, sim ou não. Ideia de pertencimento.

Problema decidível

Problemas de busca

Relacionamento binário.

X está em A como exemplo

Como um computador entende o programa.

Código fonte: gerado pela linguagem de alto nível, ele é traduzido ou interpretado

Um programa é um amontoado de palavras caso o computador não entenda.

Processo de tradução, ligado a compilação.

Compilador: analisa o código e o muda para linguagem de baixo nível. Que recebe o nome de programa objeto.

Tradução

* Geração do programa objeto
* Execução do programa objeto

Interpretação

* Programa fonte executado diretamente.

Interpretação

Diagrama

Descrição gerada automaticamente

Características de um programa

Diretrizes

* Legibilidade
* Redigibilidade
* Confiabilidade
* Custo

Legibilidade

* facilidade de leitura
* compreensão
* ortográfica
* definição adequada das estruturas.

Redigibilidade

* pode conflitar com a legibilidade
* ortogonalidade
* simplicidade da escrita
* suporte a abstração
* reuso do código
* expressividade

Confiabilidade

* verificação de tipos
* trata exceções
* uso de ponteiros
* compatibilidade entre compiladores.
* Fazer o que foi programado para fazer

Custo

* Treinamento
* Codificação
* Compilação
* Execução
* Infraestrutura

Características

Atualizações

Uso para IA

Disponibilidade de ferramentas

Comunidade ativa

Adoção pelo mercado

Análise de código

Análise léxica

* Particionar
* Classificar
* Eliminar

Leitura, caractere por caractere. Classificando e eliminando o que não é necessário.

Análise sintática

* Relacionado a forma
* Como ele é escrito, se ele está correto.

Análise semântica

* Relacionado ao significado, estudo do significado
* É a lógica do programa.

Usar os operadores corretos

Paradigmas de programação

O que é?

Forma de resolução de problema com diretrizes e limitações específicas de cada paradigma utilizando a linguagem de programação

Exemplos

* Orientada a objeto
* Procedural
* Funcional
* Estruturado
* Computação distribuída e lógico

Paradigma estruturado

* Instruções executadas em sequências
* Decisão teste lógico
* Iteração laços

Utilização

* Problemas simples e diretos
* Aprender programação

Orientação a objeto

* Baseado na utilização de objetos e suas interações
* Um objeto é descrito por características específicas comportamentos e estados
* O que tenho, sou capaz de fazer como faço.

Ex:

Uma imagem contendo Diagrama

Descrição gerada automaticamente

O que eu tenho: atributo

Sou capaz de fazer: métodos

Como faço: estados

Classe

* Alocação em memória
* Operações associadas

Pilares da orientação a objeto

* Herança: classe filha herda comportamento e estados gerais da mãe, mas também possui suas próprias características
* Encapsulamento:
* Polimorfismo

Aula 5

Primeiro contato com a programação

Parte prática

Considerações finais