**08/06/2022**

**Módulo I – Princípios de Desenvolvimento de Software**

**Curso – Introdução à programação e pensamento computacional**

Instrutora: Juliana Mascarenhas

**Pensamento computacional**

Apresentação

Apresentação da instrutora

Objetivo geral

Percurso

Aula 1 Pensamento computacional

Aula 2 Introdução à lógica da programação

Aula 3 Fundamentos de algoritmos

Aula 4 Linguagens de programação

Aula 5 Primeiro contato com a programação

Introdução

Aula 1 – pensamento computacional

O que é pensamento computacional:

* processo de pensamento envolvido na expressão de soluções em passos computacionais ou algoritmos
* Sistemático e eficiente: formulação e resolução de problemas
* Humanos e máquinas devem ser capazes de resolver
* Habilidade generalista: matemática, leitura, escrita

Quatro pilares do pensamento computacional

* Decomposição: dividir um problema complexo em subproblemas
* Reconhecimento de padrões
* Abstração: do particular para o genérico, modelagem do fenômeno
* Design de algoritmos: automatização da resolução de problemas, passo a passo da solução
  + Input -> operador -> output
  + Processo contínuo: definir solução, testar, aperfeiçoar / análise, refinamento e teste
  + Círculo virtuoso
  + Habilidades humanas x recursos computacionais:
    1. Abstração: resolução de problemas
    2. Automatização: expressão de soluções
    3. Análise: execução de solução e avaliação (refinamento)
  + Competências:
    1. Pensamento sistemático
    2. Colaboração dentro da equipe
    3. Criatividade e design
    4. Facilitador

Habilidades complementares

Raciocínio lógico:

* Forma de pensamento estruturado que permite determinar a resolução de um problema.
* Habilidade em treinamento
* Classificação:
  + Indução: a partir de um fenômeno observado gerando leis e teorias
  + Dedução: a partir de leis e teorias, se deduz previsões e explicações
  + Abdução : a partir de uma conclusão, se supõe uma premissa, muito utilizado em processos investigativos e diagnósticos
* Raciocínio lógico
  + Sintética
    - Abdução
    - Indução
  + Analítica
    - Dedução
* Aperfeiçoamento
  + Melhoramento
  + Aprimoramento
  + Refinamento
  + Ato de aperfeiçoar
    - Melhor uso de recursos
      * Encontrar solução eficiente
      * Otimizar processos
    - Melhores códigos e algoritmos
      * Simplificar linhas de código
      * Funções bem definidas
  + Determinar pontos de melhoria e refinamento

Pilares: Decomposição

Identificar o problema e dividi-lo em problemas menores, mais fáceis de resolver e de gerenciar

Estratégia

* Análise:
  + Quebrar o problema em problemas maiores
  + Estudar/explorar
  + Realizar exame detalhado
  + Decompor em elementos constituintes
* Síntese:
  + Combinar os elementos recompondo o problema original
  + Consiste em reunir elementos distintos em um único elemento
  + Processo de reconstrução
  + Fundir elementos de maneira coerente
* Ordem de execução
  + Sequencial: existe dependência e devem ser executadas em fila
  + Paralela: tarefas podem ser executadas concomitantemente e posteriormente agregadas

Decomposição

* Habilidade generalista em treinamento
* Existem maneiras distintas de resolver um problema
  + Custo x benefício
* Como decompor:
  + Identificar etapas ou coletar os dados
  + Agregar os dados
  + Funcionalidade
* Etapas:
  + Identificar as atividades
  + Determinar etapas, em paralelo ou sequencial
  + Executar cada etapa
  + Agregar as atividades, recompondo com coerência
* Exemplo: movimentos de um avatar
  + Ações
    - Padrão
      * Virar
        + Esquerda
        + Direita
      * Mover
        + Cima
        + Baixo
    - Movimento
      * Andar
      * Correr
        + Mover a perna
        + Impulsionar
      * Sentar
      * Levantar

Pilares: Padrões

Reconhecimento de padrões:

* Modelo base (referências)
* Estrutura invariante
* Repetição

Detecção de similaridades e diferenças

* Armazenamento de fotos: primeiro compressão e posterior armazenamento
* Reconhecimento de padrões é intuitivo na vida cotidiana

Determinando padrões:

* Generalizar para obter resolução de problemas diferentes
* Grau de similaridade
* Grupos conhecidos x objeto desconhecido
* Reconhecimento de padrão é feito por comparação

Padrões – como simular o comportamento humano

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

Abordagem:

* Extração de características
* Classificação de dados
* Diferentes métodos de reconhecimento de padrão

Aplicações:

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

Pilares: Abstração

Conceitos:

* Generalização: tornar mais amplo, extenso, geral – permite definir uma classe geral
* Abstração: extrapolar um objeto da realidade

Como classificar os dados

* Características: identificação
* Pontos essenciais: ignorar características que não são relevantes
* Generalizar x detalhar: definir que características ignorar ou detalhar

Exemplos de abstrações:

* Árvores
* Listas
* Grafos
* Máquinas de estados finitos
* Linguagens de programação
* Arquiteturas de TI

Pilares: Algoritmos

O computador não opera sozinho: precisa de instruções detalhadas do que e de como processar.

Instruções – o entendimento deve ser possível por humanos e máquinas:

* O que precisa ser feito
* Qual a ordem de execução

Desenvolvimento de programas

* Análise: estudo e definição de entradas e saídas
* Algoritmo: descreve o problema por meio de ferramentas
* Codificação: utiliza linguagem de programação

Algoritmos

* Sequência de passos com objetivo definido
* Execução de tarefas específicas
* Conjunto de operações que resultam em uma sucessão finita de ações

Como construir um algoritmo

* Compreensão do problema
* Definição de dados de entrada
* Definir processamento
* Definir dados de saída
* Utilizar método de construção
* Teste e diagnóstico

Construção algoritmo

* Narrativa: linguagem natural
* Fluxograma: utilização de símbolos pré-definidos
* Pseudocódigo: portugol

Como aprimorar essa habilidade

* Buscando explicações para cada decisão em seu exercícios de pensamento computacional

Estudo de caso: Perdido

Caso: existe uma pessoa perdida na floresta

Objetivo: como maximizar a sua sobrevivência

Método: decomposição, reconhecimento de padrões e abstração

Detalhamento:

* Água:
  + Chuva
  + Nascente
    - Fogo
* Comida:
  + Coletar
  + Caçar
    - Fogo
* Abrigo
  + Localização
    - Mapa
  + Proteção
    - Fogo
    - Lança
  + Quente e seco
    - Fogo

Estudo de caso: Soma de um intervalo

Caso: soma de um número de 1 a 200

Objetivo: Fazer o cálculo de modo simples, utilizando um padrão (função)

Método: decomposição, reconhecimento de padrões e abstração

A definição do padrão cria uma função matemática

Estudo de caso: Adivinhe um número

Caso: Adivinhe um número

Objetivo: Determinar o número escolhido por uma pessoa dentro de um intervalo, a partir de perguntas com respostas do tipo sim ou não. Não existe limite de perguntas,

Método: decomposição, reconhecimento de padrões e abstração

A definição do padrão de perguntas busca limites superior e inferior, como por exemplo “o número é maior do que 50?”

**Introdução à lógica de programação**

O que é lógica

Conceito: forma de raciocínio, ordenada, que segue convenções, de forma coesa.

Definição: organização e planejamento de instruções, assertivas, em um algoritmo, a fim de viabilizar a implantação de um programa.

Pressupõem a previsão de comportamentos.

Lógica ensina a pensar.

Técnicas de lógica de programação

Técnicas de lógica:

* Técnica linear:
  + Modelo tradicional
  + Execução sequenciada
  + Unidimensional
  + Recursos limitados
* Técnica estruturada (hierarquia)
  + Escrita
  + Entendimento
  + Validação
  + Manutenção
* Técnica modular
  + Partes independentes, controladas por um conjunto de regras
  + Modelo:
    - Dados de entrada
    - Processo de transformação
    - Dados de saída
  + Objetivos (metas):
    - Simplificação
    - Decomposição de problemas
    - Verificação por módulo

**09/06/2022**

**Módulo I – Princípios de Desenvolvimento de Software (continuação)**

**Curso – Introdução à programação e pensamento computacional (continuação)**

**Fundamentos de algoritmos**

Tipologias e variáveis

Informação

* Dados
* Instruções

Tipos de variáveis

* Inteiros
* Reais
* Caracteres
* Lógico (boleano): verdadeiro (1) e falso (0)

Variáveis:

* Definições: estrutura mutável; possui variações; incerto; instável, inconstante
* Pode assumir qualquer um dos valores de um determinado conjunto, restrito ao tipo definido para ela.
* Regras (boas práticas)
  + Nomenclatura
  + Primeira letra não numérica
  + Sem espaços em branco
  + É vetado a utilização de palavras reservadas da linguagem em uso
* Papeis
  + Ação: modificação de estado
  + Controle: vigiada, de controle
* Variável => do tipo constante, seu conteúdo é inalterado.

Instruções Primitivas

Instruções determinam as ações a serem aplicadas sobre os dados, por exemplo, cálculos matemáticos

Tipos de operadores:

* Unário
* Binário

Prioridades de execução dos operadores

Definição formal: linguagem de palavras-chaves de uma determinada programação

Linguagens de programação:

* Notação: sintaxes distintas, particular, por linguagem
* Operadores

Conceitos: entrada -> processamento -> saída

Instruções executadas sequencialmente

Estruturas condicionais e operadores

Na execução das instruções de processamento contém:

* Estrutura condicional: se condição satisfeita executar uma ação se não satisfeita, gera exceção
  + Simples
  + Composta
  + Encadeada
* Operadores relacionais:
  + Igual a
  + Diferente de
  + Maior que
  + Menor que
  + Maior ou igual a
  + Menor ou igual a
* Operadores lógicos: (para respostas simples, do tipo verdadeiro e falso
  + And
  + Or
  + Not

Estruturas de repetição

Quando um trecho de programa precisa ser executado mais de uma vez

Laços = controle de fluxo = malhas de repetição = repetição = loop

Vantagens:

* Redução de linhas de código
* Compreensão facilitada
* Redução de erros

Tipos de estrutura de repetição

* Enquanto <condição> faça [...]
* Repita [...] até <condição>
* Para <range> de <condição1> até (condição2) faça [...]

Os tipos de estrutura podem ser mesclados e/ou utilizadas uma dentro da outra

Vetores e matrizes

Vetor: lista de valores, em uma sequência linear, matriz unidimensional

Matriz: lista de lista de valores, em uma sequência linear, matriz multidimensional

Características:

* Coleção de variáveis
* Contíguas em memória
* Indexadas

Operações:

* Definição
* Atribuição
* Leitura

O que são funções

Funções computacionais são “derivadas” de funções matemáticas

Outros nomes:

* Subalgoritmo
* Função
* Bloco
* Método
* Sub-rotina

Representação: F : A -> B

Definição: são blocos de instruções que realizam tarefas específicas

Vantagens:

* Decomposição dos algoritmos
* Modularização do programa
* Código mais claro e conciso
* Reutilização de instruções

Características:

* Possui assinatura: nome, parâmetros de entrada e parâmetros de saída (resultado)
* Definição
* Nome
* Invocação
* Variáveis locais

Instruções de entrada e saída

Computação: dados de entrada -> processamento -> resultado

Definições:

* Entrada de dados: recebimento de dados via dispositivos de entrada (teclado, tela, outros)
* Saída de dados: impressão da informação para dispositivos de saída (tela, relatório, outros)

Tipos de saída:

* Saída programada
* Saída por interrupção

Saídas esperadas:

* Saída bem sucedida
* Erros de sintaxe
* Erros de programação
* Problemas com a interface

**Linguagens de programação**

Introdução à linguagens de programação

Objetivo: paradigmas existentes, conceito e uma breve história

Introdução:

* História da computação, alimenta novos paradigmas
* Toda evolução tecnológica inicia com hardware e continua no software
* Linha do tempo
  + Ábaco
  + Instrumentos computacionais
  + Ada Lovelace (1942) – álgebra boleana
  + Alan Turing – matemático, cálculo para decriptografia, “máquinas pensam?”
  + Claude Shannon – álgebra boleana
  + Máquina de tabular – censo americano -> IBM
* Primeiros computadores
  + Válvulas e relés
  + Harvard Mark 1 - IBM – separação de memória; dicisão, algoritmo
  + Eniac (2ª guerra mundial) – programável por fio
  + Colossus – programação por fio
  + EDVAC – programa armazenado
  + Computadores pessoais
  + Intel
  + Apple
  + IBM
* Sistemas operacionais
* Linguagens de programação
  + Assembly
    - Primeira linguagem
    - Linguagem (de montagem) de máquina
  + Lisp
  + Cobol
  + Fortran
  + Linguagens paradigmas: C, Prolog e Machine Learning
  + Linguagens de alto nível: C#, Python, Ruby, etc
  + Linguagens de novos conceitos, multiparadigmas
* Problemas computacionais: resolvível em ambiente computacional
  + Problemas de decisão
  + Problemas de busca
  + Problemas de otimização
* Definição: método padronizado composto por um conjunto de regras sintáticas e semânticas de implementação de um código fonte.

Como um computador entende o programa

Linguagem - > Código fonte -> tradução e interpretação

Sintaxe e semântica

Tradução: o compilador traduz a linguagem de programação (programa fonte) em linguagem de máquina (programa objeto)

Interpretação: não gera programa objeto, é mais lento que a tradução, mas fornece mais flexibilidade

Transpilação (tradução + interpretação): o compilador traduz a linguagem de programação em outra linguagem de alto nível (mais baixa) e então executada

Exemplos: Python é uma linguagem interpretada, JavaScript é uma linguagem de transpilação

Características de um programa

Boas práticas

Características importantes:

* Legibilidade
  + Facilidade de leitura
  + Compreensão
  + Ortogonalidade: coerência de código
  + Definição adequada das estruturas
* Redigibilidade
  + Conflita com a legibilidade
  + Ortogonalidade
  + Simplicidade de escrita – pode não ser fácil de ler
  + Suporte à abstração
  + Reuso de código
  + Expressividade
* Confiabilidade
  + Verificação de tipos
  + Tratamento de exceção
  + Uso de ponteiros
  + Compatibilidade entre compiladores
* Custo (análise de impacto)
  + Treinamento
  + Codificação
  + Compilação
  + Execução
  + Infraestrutura
* Outras características
  + Atualizações
  + Uso para IA (inteligência artificial)
  + Disponibilidade de ferramentas
  + Comunidade ativa
  + Adoção pelo mercado

Análises de código

Compiladores

* Análise léxica (scanner): léxicos / tokens
  + Particionar
  + Classificar:
  + Eliminar
* Análise sintática:
  + Forma, componente do sistema linguístico
  + Gramática específica de uma linguagem de programação
* Análise semântica
  + Estudo do significado, relação entre significantes
  + Erros de semântica produzem resultados diferentes do esperado

Paradigmas de programação

O que é um paradigma: forma de resolução de problemas com diretrizes e limitações específicas de cada paradigma de linguagem

Classificação:

* Orientação a objeto:
* Procedural: sucesso
* Funcional: fundamentado em instruções/funções
* Estruturado: blocos aninhados
* Computação distribuída: funções independentes que permitem distribuição
* Lógica

Paradigma Estrutural

* As instruções são executas em sequência
* Conceitos: sequencia, decisão, iteração (funções, laços, condições)
* A ênfase é em sequência
* Utilização:
  + Problemas simples e diretos
  + Ensino/aprendizagem de programação

Paradigma Orientação à Objeto

Baseado no mundo real, em objetos e seus interações

O que é um objeto e qual o comportamento de um objeto real

Objetos é descrito por características (atributos) específicas, comportamentos (métodos) e estados (como faz)

Classificação:

* Modelo (classe)
* Atributos
* Métodos
* Estados

Do ponto de vista da programação

* Classe
  + Alocação em memória
  + Operações associadas
* Pilares de orientação à objeto
  + Herança
    - Classe filha herda características de uma classe mãe
    - Especializações: comportamentos e estados específicos
  + Encapsulamento
  + Polimorfismo
  + Abstração

**10/06/2022**

**Módulo I – Princípios de Desenvolvimento de Software (continuação)**

**Curso – Introdução à programação e pensamento computacional (continuação)**

**Primeiro contato com a programação**

Algoritmos em portugol

Portugol Webstudio:

* Ferramenta para pseudo-código
* IDE de código aberto para programar em Portugol direto de seu navegador
* <https://portugol-webstudio.cubos.io>
* Cubos: “Somos um hub de conhecimento e inovação, criamos nossas próprias empresas digitais e apoiamos empresas na tomada de decisão, desenvolvimento e resolução de desafios técnicos complexos.”
* “A Cubos Academy oferece cursos de tecnologia para todos os perfis, do iniciante ao avançado, em diversas áreas”

Vantagens do uso do portugol: permite abstrair da sintaxe de uma linguagem de programação e focar na lógica de programação, para resolução de problemas

Considerações finais

Referências bibliográficas: Computational Thinking

Materiais de apoio

Download da apresentação utilizada no curso: “Primeiros passo atualizado.pdf”

Certifique seu conhecimento

Quiz para responder em 12 minutos: 100% aproveitamento