Caderno de anotações e resumos

Programação vs codificação

Antes de tudo, é preciso entender uma pequena diferença entre estes dois conceitos.
Quando falamos de programação, estamos falando do ato de programar ou planejar as ações que serão executadas. Imagine a necessidade de fixar um quadro na parede, você já tem a furadeira e os parafusos, se apenas fizer dois furos e pendurar o quadro, quais as chances de ficar torto? Grande, não?

É preciso se programar, pegar uma régua, medir a distância entre os furos, a altura do quadro, pegar a máquina, fazer os furos, colocar os parafusos e só então pendurar o quadro.

Essa descrição dos passos para executar uma tarefa feita de forma simplificada, como quem fala para outra pessoa é o que podemos chamar de programar. “Estabelecer um plano para fazer algo”

Codificar, por sua vez, é a ação de transformar em códigos aquilo que foi programado.
Existem muitas linguagens de programação, e cada uma delas tem sua forma de descrever os passos a executar, encare isso como idiomas, inglês, português e chines, todos com origens diferentes mas exercem a mesma função.

Logo quando abordamos temas como lógica de programação e algorítmos, falamos de programação. Por outro lado, estruturas de repetição, variáveis, constantes, tipos de dados, operadores e condicionais, fazem parte da codificação.

Vamos abordar esses temas a seguir! ;)

Lógica de programação

Lógica de programação é o conjunto de regras e técnicas que os programadores utilizam para projetar e desenvolver programas de computador

É a habilidade de pensar de forma lógica e estruturada, decompondo um problema complexo em etapas mais simples

Estas ações decompostas em etapas mais simples, são o que é chamado de algorítmos

Algoritmo

Algoritmo é um conjunto de regras e procedimentos lógicos perfeitamente definidos que levam à solução de um problema em um número finito de etapas

É importante salientar que as etapas precisam ser finitas, visto que há um problema/objetivo definido, o algoritmo precisa ter etapas até que atinja o objetivo, nem mais e nem menos.

Por fim, dizer “previamente definidos”, deixa claro que este é um processo que precede, vem antes, de uma ação, a ação de codificar! Por isso é de suma importância que o algoritmo definido seja o mais detalhado possível, evitando assim falhas indesejadas na etapa de codificação.

Caderno de anotações e resumos

Conceitos de codificação

Dados

O primeiro conceito importante a ser abordado são os as informações que um programa tem de manipular, ou seja os dados manipulados. Estes dados podem ter diversas classes ou naturezas, sendo algumas bastante utilizadas, que vou descrever a seguir.

Numéricos

Dados numéricos podem ser definidos como Inteiros (int) ou como Ponto Flutuante (float), os quais podem assumir tipos de números diferentes, sendo inteiros, capazes de receber apenas números que não possuem casas/partes decimais ou seja: 1, 50, -5, -245. Os dados numéricos de ordem ponto flutuante por sua vez, são capazes de lidar com números com parte decimal, como por exemplo: 3.14, -2.5, 312.73, 0.04501802.

Caracteres

Caracteres (char) são capazes de conter caracteres, porém é possível armazenar apenas um por vez, como por exemplo: considerando o dado “char” registrado o caracter ‘F’, temos **char = F**. Mas será atribuído o valor ‘&’ em char. E ao invés de resultar em **char = ‘F&’** (errado), o valor de char será reescrito para &, obtendo o resultado **char = ‘&’** (correto).

Cadeias de Caracteres

Quando for necessário manipular um dado com múltiplos caracteres, usa-se a variável cadeias de caracteres que é comumente conhecida por string. Armazenando múltiplos caracteres, podem ser uma frase ordenada, assim como uma sequência não regular. Exemplos: "Olá, mundo!", "#FFFFFF", "Eu F&1!".

Booleanos

Esta variável, bool, tem a característica particular, ela é capaz de armazenar um estado, verdadeiro ou falso. Sendo possível definir o estado como caixas de marcação, slides de confirmação, a fim de aferir o estado de um dado específico

O estado é “selecionado”, logo, temos bool = true

O estado é “desselecionado”, logo, temos bool = false

Estruturas de repetição

Quando há a necessidade executar uma atividade de forma consecutiva, até que uma condição seja cumprida, podemos utilizar uma estrutura de repetição, podendo ser chamado de laço ou loop (mais utilizado). Esta estrutura pode ser definida como um bloco de instruções/código a ser executado indefinidamente até que a condição “master” seja cumprida.

Caderno de anotações e resumos

De forma prática, podemos observar uma contagem de cronómetro, de 0 a 59 para cada minuto completo, utilizando uma estrutura de repetição define-se o valor inicial “x” igual a zero, daremos a indicação de que a cada segundo incrementar um valor à “x”.
Se não colocarmos uma condição esse número crescerá para sempre, porém, como estamos contando segundos diremos para ir até 59.
Aplicando uma estrutura de repetição neste ponto, diremos “sempre que o valor de “x” for maior que 59, “x” deve ser igualado a 0 até que haja comando para parar”
Dessa forma, sem a necessidade que escreva o código com várias linhas repetidamente, com a estrutura de repetição apenas um comando é necessário.

Variáveis e constantes

Como os próprios nomes sugerem, os dados manipulados em uma codificação podem ser constantes, como a aceleração da gravidade da terra g= 9.8 m/s², que pode ser manipulado e nunca será alterado, ou seja constante!

As variáveis, por sua vez, mudam, ou podem ser alteradas ao longo do tempo. Podemos considerar situações onde os dados são manipulados e quando nós alteramos seu valor.
Dados que alteram-se com o tempo podem ser observados na quantidade de bateria do celular, há um monitoramento da bateria que coleta a quantidade disponível e exibe em porcentagem, conforme a bateria for acabando, a porcentagem na tela (um dado que varia mas não é manipulado de forma externa). Um exemplar de dado variável é o saldo de uma conta bancária, o valor apresentado varia? Sim! Mas depende de movimentação externa para tal, ou seja, toda movimentação efetuada seja um levantamento ou um depósito (entrada ou saída), o valor será atualizado, sendo assim uma variável que são manipuladas

Tipos de dados

Agora falando sobre toda informação a ser manipulada ao longo de qualquer codificação, de forma mais específica, os dados manipulados. Os dados são conjuntos de valores que uma variável pode armazenar, estando divididos em duas categorias, básicos e complexos, também conhecidos como primitivos e estruturados respectivamente.

Os dados básicos geralmente estão associados a uma característica específica do dado ali alocado, podemos observar esta particularidade observando alguns exemplos de tipos de dados primitivos como:

- Inteiro (int): Representa números inteiros, positivos ou negativos, sem casas decimais
- Ponto Flutuante (float/double): Representa números com casas decimais
- Caractere (char): Representa um único caractere, como uma letra, número ou símbolo
- Booleano (bool): Representa um valor lógico, podendo ser verdadeiro (True) ou falso (False).
- String: Representa uma sequência de caracteres (ex: "Olá, mundo!", "123")

Os dados chamados complexos, por sua vez, possuem uma condição de armazenamento. Vamos observar alguns exemplos:

- Arrays/Vetores: Coleções ordenadas de elementos do mesmo tipo.
- Listas: Semelhantes a arrays, mas podem ser mais flexíveis em termos de tamanho e operações.

Caderno de anotações e resumos

Registros/Estruturas: Agrupam diferentes tipos de dados relacionados em uma única unidade.
Pilhas (Stacks): Estruturas de dados que seguem o princípio LIFO (Last In, First Out).
Filas (Queues): Estruturas de dados que seguem o princípio FIFO (First In, First Out).
Árvores: Estruturas hierárquicas usadas para representar relações entre dados.
Grafos: Representam relações entre elementos de forma mais geral, não necessariamente hierárquica

Operadores e condicionais

Por fim, os operadores e as condicionais são elementos fundamentais para criar lógica e tomar decisões dentro de um programa, estas decisões podem ser tomadas por meio cálculos e comparações, quando utiliza-se operadores, estes operadores podem ser:

Aritméticos: Realizam operações matemáticas como adição (+), subtração (-), multiplicação (*), divisão (/), e módulo (%).

Comparação: Compararam valores, retornando um booleano (verdadeiro ou falso). Exemplos incluem igual a (==), diferente de (!=), maior que (>), menor que (<), maior ou igual a (>=), e menor ou igual a (<=).

Lógicos: Combinam expressões booleanas usando AND (&&), OR (||), e NOT (!).

Atribuição: Atribuem valores a variáveis, como (=), +=, -=.

As condicionais, por sua vez, avaliam condições e de acordo com a "resposta" desta avaliação, executa o bloco de código correspondente ao resultado. Algumas delas são:

if: Executa um bloco de código se uma condição for verdadeira.
else: Executa um bloco de código se a condição do if for falsa.
elif (ou else if): Permite verificar múltiplas condições em sequência.
switch (ou case): Permite escolher entre vários blocos de código com base no valor de uma variável

Com estas operações, é possível gerar uma estrutura de decisões complexas e capazes de cumprir tarefas exigentes.