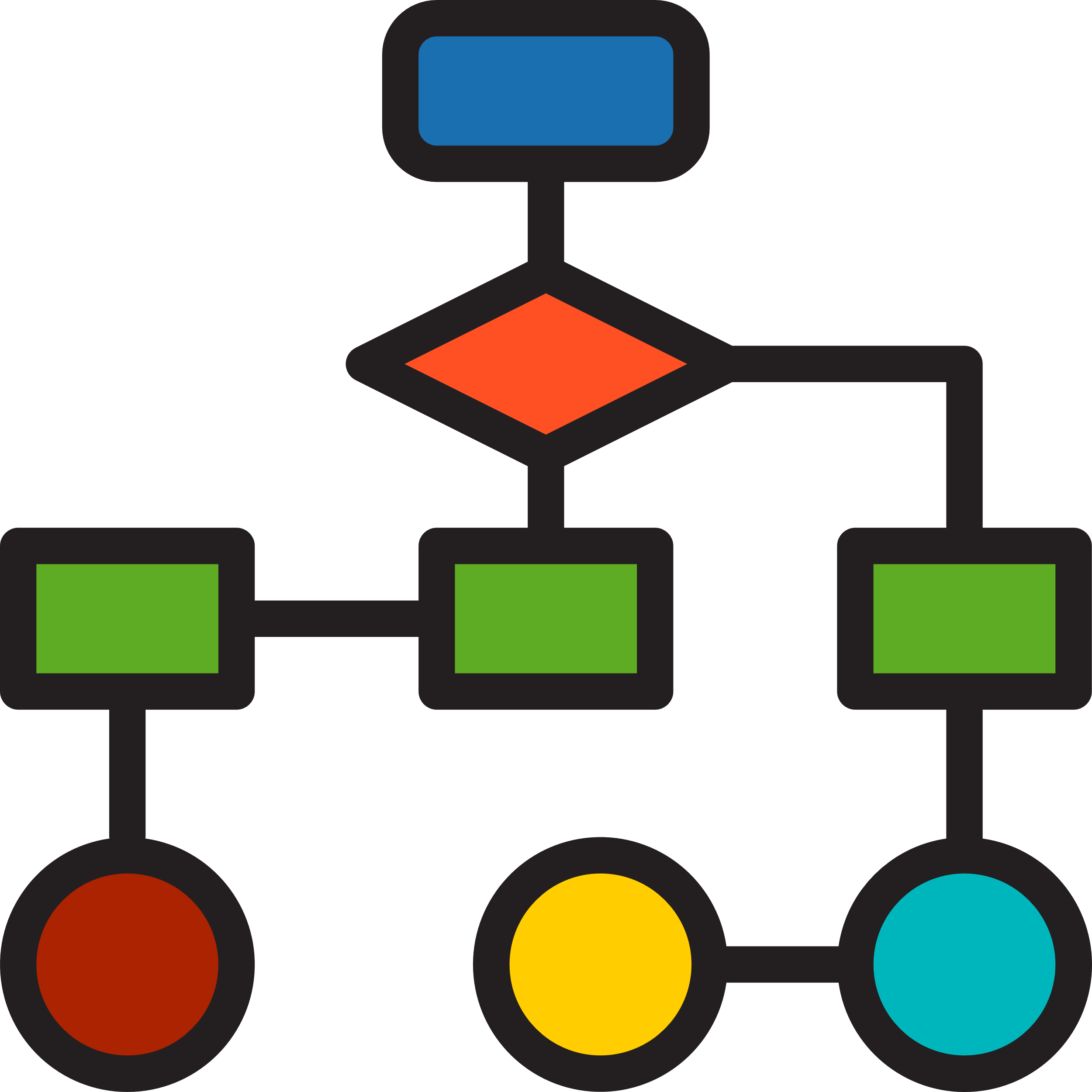
ssssss

**ESTRUTURAS DE CONTROLE**

A estrutura de seleção pertence as chamadas estruturas de controle de fluxo em um algoritmo. As principais estruturas de controle são:

* Estruturas de sequenciação;
* Estruturas de seleção;
* Estruturas de repetição.

***ESTRUTURAS******DE SEQUENCIAÇÃO***

A estrutura de sequencial é a mais simples de todas e pode ser dita como uma estrutura de ações que deverão ser executadas de maneira linear de cima para baixo e da esquerda para direita [1]. Além disso muitas linguagens adotam as identações para estabelecer o fluxo de um algoritmo principalmente quando existem estruturas de seleção e repetição, por exemplo.

|  |
| --- |
| **>>>** # Início -> Aqui é uma linha de comentário em Python  **>>>** H = 50  **>>>** J = 25  **>>>** X = 35  **>>>** MEDIA = (H + J + X) / 3  **>>>** **print**(MEDIA)  36.666666666666664 |

**Exercício 1.1:** Escreve um algoritmo sequencial que determine a média final de um aluno universitário que fez 5 provas com os seguintes valores: [1,0; 3,0; 4,5; 10,0; 9,5]. Os valores devem ser introduzidos no algoritmo de forma escrita no corpo do algoritmo.

**Exercício 1.2:** Escreve um algoritmo sequencial que forneça as quantidades de insumos para execução de um assentamento de piso quando o usuário informa a quantidade total do serviço de assentamento. Resolver o exemplo para execução de 155 m2 de piso.

Composição Unitário de Custo com os consumos unitários:

* Piso porcelanato = 1,10 m2/ m2
* Rejunte = 0,24 kg/ m2
* Argamassa colante = 8,62 kg/ m2
* Azulejista = 0,95 hr/ m2
* Ajudante de azulejista = 0,34 hr/ m2

Para solução do exercício deve-se lembrar que consumo total de um insumo é a quantidade do serviço vezes o consumo unitário.

***ESTRUTURAS DE SELEÇÃO***

A estrutura de seleção permite a escolha de um bloco de ações a serem executadas quando determinadas condições, representadas por expressões lógicas ou relacionais são ou não satisfeitas [1].

Estas estruturas de repetição podem apresentar seleção simples ou seleção compostas (também chamadas de encadeadas). As estruturas são exemplificadas abaixo:

|  |
| --- |
| **>>>** **if** (VAR\_1 > 50):  **>>>** A = B +80 # Executa comando se verdadeiro |

**Seleção simples**

|  |
| --- |
| **>>>** **if** (VAR\_1 > 50):  **>>>** A = B +80 # Executa comando se verdadeiro  **>>>** **else**:  **>>>** A = 35 # Executa comando se falso |

**Seleção composta**

|  |
| --- |
| **>>>** # Executa o comando por agrupamento de seleções  **>>>** **if** (VAR\_1 > 50):  **>>> if** (VAR\_2 > 50):  **>>>** A = B +80  **>>>** **elif** (VAR\_1 > 50) **and** (VAR\_1 < 170):  **>>>** A = 35 |

**Seleção composta encadeada**

**Exercício 1.3:** Crie um algoritmo que receba a idade de um jogador de futebol e classifique-o da seguinte forma:

* 5 a 8 anos: Infantil
* 9 a 13 anos: Juvenil A
* 14 a 17 anos: Juvenil B
* Maiores de 18 anos: Adulto

**Exercício 1.4** [1]**:** Crie um algoritmo para que dada as coordenadas (X, Y) de 3 pontos o programa deverá devolver *output* as respostas para as seguintes perguntas:

* É um triângulo?
* Se triângulo, qual o tipo de triângulo?
* Se triângulo, qual a área do triângulo?

***ESTRUTURAS DE REPETIÇÃO***

As estruturas de repetição são alternativas de código para criar instruções repetidas. Tal fato evita que o programador tenha que repetir blocos de códigos aumentando consideravelmente o tamanho do algoritmo [2].

Nas linguagens de programação essas estruturas normalmente são representadas pela instrução *for* e *while*. A instrução *for* é empregada quando se conhece o número de repetições necessárias para solução do problema, por exemplo a leitura e somatório de 20 notas de alunos de um curso. Já a instrução *while* é empregada quando não se tem conhecimento do número de repetições necessárias e então faz-se uso de uma chave para interromper a repetição.

|  |
| --- |
| **>>>** A = 0  **>>>** N = 50  **>>>** **for** I **in** **range(**N**)**:  **>>>** A = A +80 # Executa a instrução de atribuição por 50 vezes |

**Instrução *for***

|  |
| --- |
| **>>>** ERRO = 50  **>>>** **while** ERRO **>** 1:  **>>>** ERRO = ERRO / 4 # Executa a instrução até que ERRO seja > 1,00 |

**Instrução *while***

**Exercício 1.5** [2]**:** Crie um algoritmo que permita digitar o nome de uma pessoa, seu salário bruto e então imprima na tela a alíquota de imposta de renda. Crie a possibilidade em um laço tipo *for* que o usuário faça esse mesmo processo para 10 pessoas automaticamente.

|  |  |
| --- | --- |
| **Salário** | **Alíquota** |
| Salário menor que R$ 600,00 | isento |
| Salário R$ 600,00 e R$ 1.500,00 | 10% do salário bruto |
| Salário R$ 1.500,00 | 15% do salário bruto |

**Exercício 1.6** [2]**:** Dados um país A, com 5.000.000 de habitantes e uma taxa de natalidade de 3% ao ano, e um país B, com 7.000.000 de habitantes e uma taxa de natalidade de 2% ao ano, calcular e imprimir o tempo necessário para que a população do país A ultrapasse a população do país B. Utilize a instrução *while* para construção desse algoritmo.

**REFERÊNCIAS**

[1] Forbellone ALV, Eberspächer HF. Lógica de programação: a construção de algoritmos e estruturas de dados. São Paulo: Pearson Prentice Hall; 2007.

[2] Lopes A, Garcia G. Introdução à programação: 500 algoritmos resolvidos. Rio de Janeiro (RJ): Campus; 2002.