### **RELATÓRIO: SIMULAÇÃO COMPUTACIONAL DE UM SISTEMA DE VOTAÇÃO DE COMITÊ**

#### **1. Introdução**

O presente relatório descreve uma **simulação computacional de um sistema de votação**, projetado para um comitê de uma multinacional [Rascunho Ads]. A lógica por trás deste sistema é baseada nos princípios da **Lógica Computacional**, que envolve a tradução de regras de decisão complexas para uma linguagem que o computador possa processar [LIVRO\_UNICO logica.pdf, 20]. Especificamente, o código reflete conceitos abordados na **Unidade 1, Seção 2** do material de "Lógica Computacional", onde a **Álgebra Booleana** e os estados binários "ligado" (1) e "desligado" (0) são fundamentais para a construção de algoritmos e o desenvolvimento de sistemas [LIVRO\_UNICO logica.pdf, 47, 59].

#### **2. Objetivo**

O objetivo principal do código é **simular um sistema de votação** para um comitê, com o propósito de determinar se um projeto é aprovado ou não [Rascunho Ads]. A aprovação do projeto, que resulta no acendimento de uma "luz", depende de duas condições lógicas específicas [Rascunho Ads]:

* O Diretor Executivo (representado por 'A') deve votar a favor [Rascunho Ads].
* O projeto deve obter maioria de votos, ou seja, pelo menos dois votos a favor [Rascunho Ads].

O sistema deve retornar um valor binário: **1 para projeto aprovado** (luz acesa) ou **0 para projeto não aprovado** (luz apagada) [Rascunho Ads].

#### **3. Material e Métodos**

A metodologia empregada para a simulação é a implementação de um **código em Python** que traduz as regras de aprovação em lógica computacional [Rascunho Ads]. Os materiais e métodos incluem:

* **Ferramentas de Suporte:** A biblioteca itertools é importada no início do código, especificamente a função itertools.product, que é essencial para **gerar todas as combinações possíveis** de votos. Isso permite a construção de uma **Tabela Verdade** abrangente, fundamental para analisar todos os cenários da votação [Rascunho Ads].
* **Definição da Função de Votação:** Uma função nomeada simular\_votacao é definida, aceitando três parâmetros de entrada: A, B e C. Estes representam os votos dos três membros do comitê. Os votos são representados por valores binários: 1 para "a favor" e 0 para "contra" [Rascunho Ads].
* **Implementação das Condições de Aprovação:**
  + **Condição do Diretor:** condicao\_diretor = (A == 1). Esta linha verifica se o voto do Diretor Executivo (A) é 1. Se for, a condição é avaliada como True [Rascunho Ads].
  + **Condição de Maioria:** condicao\_maioria = (A + B + C >= 2). Esta linha soma os votos (A, B, C) para verificar se o total é igual ou superior a 2. Se for, a condição é avaliada como True [Rascunho Ads]. Por exemplo, se A=1, B=1, C=0, a soma é 2 e a condição de maioria é True [Rascunho Ads].
* **Lógica de Decisão:** O operador lógico fundamental **and (e)** é utilizado na estrutura condicional if condicao\_diretor and condicao\_maioria:. Este operador garante que o projeto só será aprovado **se ambas as condições (condicao\_diretor E condicao\_maioria) forem verdadeiras** [Rascunho Ads]. Se ambas forem True, a função retorna **1** (projeto aprovado e luz acesa). Caso contrário, a função retorna **0** (projeto não aprovado e luz apagada) [Rascunho Ads].

#### **4. Resultados e Discussão**

O código demonstra a aplicação prática da **Álgebra Booleana**, onde os valores 0 e 1 representam "falso" e "verdadeiro", respectivamente. A lógica implementada resulta em [Rascunho Ads]:

* **Aprovação:** O projeto é aprovado (retorna 1) **somente quando o Diretor Executivo vota a favor (A == 1) E a soma total dos votos é de pelo menos dois (A + B + C >= 2)**. Neste cenário, a "luz se acende" [Rascunho Ads].
* **Não Aprovação:** Em qualquer outro cenário onde uma ou ambas as condições não são satisfeitas, o projeto **não é aprovado** (retorna 0), e a "luz não se acende" [Rascunho Ads].

Este comportamento é uma manifestação direta da **tabela verdade do operador lógico AND (conjunção)**, que só produz um resultado verdadeiro se todas as suas entradas forem verdadeiras [LIVRO\_UNICO logica.pdf, 230-231]. A simulação permite visualizar como um sistema digital (como um circuito elétrico virtual, onde 1 é "ligado" e 0 é "desligado") pode ser projetado para tomar decisões com base em regras lógicas bem definidas, assim como visto nos conceitos de **Lógica Clássica** e **Período Booleano** [LIVRO\_UNICO logica.pdf, 47, 59-60].

#### **5. Conclusão**

O "Rascunho Ads" fornece um exemplo claro e funcional de como os **princípios da lógica** são aplicados na construção de sistemas computacionais. Através da implementação das condições de voto e do uso de operadores lógicos como o and, o código efetivamente **traduz um problema do mundo real em um algoritmo** que pode ser sistematicamente avaliado [LIVRO\_UNICO logica.pdf, 16-17, 20]. A utilização de valores binários e a dependência de todas as condições para uma saída positiva (projeto aprovado) são características intrínsecas da **Álgebra Booleana**, reforçando a sua relevância na base do desenvolvimento de softwares e na compreensão do raciocínio computacional [LIVRO\_UNICO logica.pdf, 29, 59].

#### **6. Referências Bibliográficas**

SCHEFFER, Vanessa Cadan; VIEIRA, Gilberto; LIMA, Thiago Pinheiro Felix da Silva e. **Lógica Computacional**. Londrina: Editora e Distribuidora Educacional S.A., 2020.

TANAKA, Neide Mariko et al. **Alternativa ao uso de animais para disciplina de técnica cirúrgica**. Londrina: UNOPAR, 2015.

*Nota: As fontes "Rascunho Ads" e "LMC-11-Fundamentos\_2024-08-21.pdf" são documentos fornecidos diretamente na nossa conversa e não publicações formais. Por essa razão, foram citadas no texto conforme sua identificação original, mas não incluídas na seção de Referências Bibliográficas, que segue o padrão ABNT para publicações formais.*