Lista 1 — Explicações (C++)

Curso: Análise e Desenvolvimento de Sistemas (ADS)

Disciplina: Algoritmos

Aluno: [Nadson Santos Nascimento]

Data: 09/10/2025

Este documento apresenta, de forma objetiva e prática, as explicações das soluções implementadas para a Lista 1. As soluções foram desenvolvidas em C++, com foco em validação de entrada, clareza de lógica e exemplos de execução. O repositório deverá conter os códigos-fonte (.cpp) separados por questão e este arquivo .docx com as explicações.

# Questão 1 — Vogal ou Consoante (q1\_vogal\_consoante.cpp)

## Objetivo

Ler uma letra e informar se é vogal ou consoante.

## Solução proposta

* Valido se o caractere digitado é uma letra (std::isalpha).
* Converto para minúscula (std::tolower) e verifico se pertence ao conjunto {a, e, i, o, u}.
* Caso não seja letra, informo que a entrada é inválida.

## Exemplos de execução

Digite uma letra: a  
Vogal

Digite uma letra: b  
Consoante

Digite uma letra: 3  
Entrada invalida: nao e uma letra.

## Como compilar e executar

g++ -std=c++17 q1\_vogal\_consoante.cpp -o q1  
./q1

# Questão 2 — Nota entre 0 e 10 com validação (q2\_validacao\_nota.cpp)

## Objetivo

Solicitar uma nota entre 0 e 10; enquanto for inválida, continuar pedindo.

## Solução proposta

* Uso de laço while(true) para continuar solicitando até a entrada ser válida.
* Trato entradas não numéricas limpando o estado de std::cin (clear + ignore).
* Verifico faixa fechada [0, 10].

## Exemplos de execução

Digite uma nota (0 a 10): 12  
Valor invalido! A nota deve estar entre 0 e 10.

Digite uma nota (0 a 10): abc  
Valor invalido! Digite um numero.

Digite uma nota (0 a 10): 8.5  
Nota valida: 8.5

## Como compilar e executar

g++ -std=c++17 q2\_validacao\_nota.cpp -o q2  
./q2

# Questão 3 — Crescimento populacional fixo (q3\_populacao.cpp)

## Objetivo

Partindo de A = 80.000 (3% a.a.) e B = 200.000 (1,5% a.a.), calcular quantos anos são necessários para A alcançar ou ultrapassar B.

## Solução proposta

* Inicializo A, B e as taxas anuais (0,03 e 0,015).
* Itero ano a ano multiplicando A e B por (1 + taxa).
* Conto os anos até A >= B e apresento resultado.

## Exemplo de execução (saída típica)

Anos necessarios para A alcançar/ultrapassar B: 63  
Populacao A aproximada: 212254  
Populacao B aproximada: 211681

## Como compilar e executar

g++ -std=c++17 q3\_populacao.cpp -o q3  
./q3

# Questão 4 — Versão parametrizada com validação e repetição (q4\_populacao\_parametrizada.cpp)

## Objetivo

Permitir que o usuário informe populações iniciais e taxas (em %) para A e B, validando as entradas e possibilitando repetir o cálculo.

## Solução proposta

* Funções de leitura com validação: valores positivos (populações) e não negativos (taxas).
* Casos especiais: se A >= B, anos = 0; se taxaA <= taxaB e A < B, A nunca alcança B.
* Laço de repetição controlado por (S/N).

## Exemplo de execução (resumo)

Informe a populacao inicial do pais A: 50000  
Informe a populacao inicial do pais B: 120000  
Informe a taxa de crescimento de A (em %): 3  
Informe a taxa de crescimento de B (em %): 1.5  
Anos necessarios: 63  
Deseja repetir (S/N)? N

## Como compilar e executar

g++ -std=c++17 q4\_populacao\_parametrizada.cpp -o q4  
./q4

# Questão 5 — Posto de combustíveis (q5\_posto\_combustiveis.cpp)

## Objetivo

Calcular o valor a pagar por litros de Álcool (A) ou Gasolina (G) conforme a tabela de descontos por litro. Preços: Álcool = R$ 1,90; Gasolina = R$ 2,50.

## Solução proposta

* Leio a quantidade de litros (> 0) e o tipo de combustível (A/G) com validação.
* Defino o preço por tipo e a taxa de desconto: Álcool (<=20: 3%; >20: 5%) | Gasolina (<=20: 4%; >20: 6%).
* Cálculo: total = litros × preço × (1 − desconto). Exibo com 2 casas decimais.

## Exemplos de execução

Litros: 25 | Tipo: A  
Preco base: 1,90 | Desconto: 5%  
Valor a pagar: R$ 45,13

Litros: 18 | Tipo: G  
Preco base: 2,50 | Desconto: 4%  
Valor a pagar: R$ 43,20

## Como compilar e executar

g++ -std=c++17 q5\_posto\_combustiveis.cpp -o q5  
./q5

# Observações finais

Todos os programas foram escritos pensando em entradas reais de usuário e em mensagens claras de erro. As validações evitam loop infinito e tratam dados fora do formato esperado. No repositório, recomendo colocar este .docx em ~/listas/1\_bimestre/lista1 e referenciar no README.md conforme solicitado.