🌱Curso Linguagem C++ – Nível 2: Codemancer

✨ **"O mago dos comandos — conjura lógica com estruturas poderosas."**

🧙‍♂️ Aula 1 — Condicionais

🎯 Objetivo

Usar `if`, `else if`, `else` e o operador ternário para controlar o fluxo do programa.

🧠 Teoria (5 min)

- `if (condicao) { ... }` executa o bloco se a condição for verdadeira.
- Encadeie com `else if` e finalize com `else` para cobrir casos restantes.
- Operador ternário: `cond ? valor_se_verdadeiro : valor_se_falso;` — ótimo para decisões curtas.
- Curto-circuito (em `&&` e `||`) evita avaliações desnecessárias.

💻 Exemplos

```
#include <iostream>
using namespace std;

int main() {
    int idade;
    cout << "Idade: ";
    cin >> idade;

    if (idade < 0) {
        cout << "Idade invalida.\n";
    } else if (idade < 12) {
        cout << "Crianca\n";
    } else if (idade < 18) {
        cout << "Adolescente\n";
    } else {
        cout << "Adulto\n";
    }

    // Ternário para rótulo rápido
    string status = (idade >= 18) ? "maior" : "menor";
    cout << "Voce e " << status << " de idade.\n";
}
```

🧩 Exercícios

1. Par/Ímpar: ler um inteiro e dizer se é par ou ímpar.

2.IMC simplificado: ler peso e altura, calcular IMC e classificar (abaixo/normal/sobrepeso/obesidade) com `if/else if`.

3.Maior de dois: ler dois inteiros e imprimir o maior (use o ternário).

4.Validação de nota: ler nota de 0 a 10; se fora do intervalo, avisar “Nota inválida”.

🔥 Desafio

Validador de senha: ler uma string e validar:

•mínimo 8 caracteres

•contém pelo menos um dígito

•contém pelo menos uma letra
Imprimir mensagens específicas do que faltou.

🏠 Katas

•Escrever 5 expressões booleanas diferentes e testar (ex.: `a%3==0 && a>10`).

🧙‍♀️ Aula 2 — `switch/case` e validação

🎯 Objetivo

Aplicar `switch` para escolhas múltiplas e reforçar validações de entrada.

🧠 Teoria (5 min)

• `switch (expressao_integral)` distribui para `case` específicos e opcional `default`.

• Sempre terminar cada case com `break` (salvo quando desejar “fallthrough” deliberado).

• Use `default` para casos não mapeados.

💻 Exemplo

```
#include <iostream>
using namespace std;

int main() {
    int opcao;
    cout << "[1] Somar\n[2] Subtrair\n[3] Multiplicar\n[4] Dividir\nEscolha: ";
    if (!(cin >> opcao)) {
        cout << "Entrada invalida.\n";
        return 0;
    }
```

```
    int a, b;
    cout << "Digite dois inteiros: ";
    cin >> a >> b;

    switch (opcao) {
        case 1: cout << "Resultado: " << (a + b) << "\n"; break;
        case 2: cout << "Resultado: " << (a - b) << "\n"; break;
        case 3: cout << "Resultado: " << (a * b) << "\n"; break;
        case 4:
            if (b == 0) cout << "Divisao por zero!\n";
            else cout << "Resultado: " << (a / b) << "\n";
            break;
        default: cout << "Opcao invalida.\n";
    }
}
```

🧩 Exercícios

1. Menu de conversão: 1) km → m, 2) m → km, 3) °C → °F, 4) °F → °C.
2. Dia da semana: ler 1–7 e imprimir o nome. Usar `default` para inválido.
3. Notas em conceito: 0–10 → A/B/C/D/E (defina sua regra de faixas) com `switch` usando divisão inteira (ex.: `nota/2`).

🔥 Desafio

Mini-roteador de comandos: ler um caractere de ação (`'N','S','L','O'`) e mover um ponto `(x,y)` no plano. `N` soma em `y`, `S` subtrai, etc. Exibir nova posição.

🏠 Katas

- Reescreva um `if/else` longo em `switch` e compare clareza.

🧙‍♀️ Aula 3 — Laços (for, while, do…while)

🎯 Objetivo

Repetir tarefas com segurança e eficiência; dominar `break` e `continue`.

🧠 Teoria (5 min)

- `for (ini; cond; passo)` ideal para contagens.
- `while (cond)` para repetições condicionais abertas.
- `do { ... } while (cond);` garante uma execução mínima.
- `break` encerra o laço; `continue` pula para a próxima iteração.

💻 Exemplos

```
#include <iostream>
using namespace std;

int main() {
    // Soma de 1..n (for)
    int n; cout << "n: "; cin >> n;
    int soma = 0;
    for (int i = 1; i <= n; ++i) soma += i;
    cout << "Soma = " << soma << "\n";

    // Leitura até valor válido (while)
    int x;
    cout << "Digite um inteiro positivo: ";
    while (cin >> x && x <= 0) {
        cout << "Tente de novo: ";
    }

    // Menu que repete (do...while)
    int op;
    do {
        cout << "[1] Ola  [0] Sair: ";
        cin >> op;
        if (op == 1) cout << "Ola!\n";
    } while (op != 0);
}
```

🧩 Exercícios

1. Tabuada: imprimir a tabuada de 1 a 10 (laço duplo).
2. Fatorial: ler `n` e calcular `n!` (tratando `0! = 1`).
3. Contagem de dígitos: ler um inteiro e contar quantos dígitos possui (trabalhar com número positivo).
4. Números primos (básico): ler `n` e dizer se é primo (teste até `n/2` ou `sqrt(n)` se quiser otimizar).

🔥 Desafio

Gerador de séries: ler `n` e imprimir os `n` primeiros termos de Fibonacci.

🏠 Katas

- Implementar somatório de pares 1..n usando `continue`.

🧙‍♂️ Aula 4 — Funções (parâmetros, retorno, overload)

🎯 Objetivo

Quebrar problemas em funções reutilizáveis; praticar sobreposição de funções.

🧠 Teoria (5 min)

- Funções reduzem repetição e clarificam intenção.
- Assinatura: tipo de retorno + nome + parâmetros.
- Overload: várias funções com mesmo nome e assinaturas diferentes.

💻 Exemplos

```
#include <iostream>
#include <cmath>
using namespace std;

// Overloads de 'area'
double area(double lado) {             // quadrado
    return lado * lado;
}
double area(double base, double altura) { // retangulo
    return base * altura;
}
double area(double raio, bool circulo) {  // circulo se 'circulo' for true
    return (circulo ? M_PI * raio * raio : raio); // truquezinho didático
}

// Utilitárias
int soma(int a, int b) { return a + b; }
long soma(long a, long b) { return a + b; } // overload por tipo

int main() {
    cout << "Area quadrado(5) = " << area(5.0) << "\n";
    cout << "Area retangulo(3,4) = " << area(3.0, 4.0) << "\n";
    cout << "Area circulo(2) = " << area(2.0, true) << "\n";
    cout << "soma int = " << soma(2,3) << ", soma long = " << soma(2L, 3L) << "\n";
}
```

🧩 Exercícios

1. Máximo de três: função `max3(a,b,c)` que retorna o maior.
2. Potência inteira: `pot(base, expoente)` (expoente ≥ 0) sem usar `pow`.
3. Contagem em string: função que conta vogais em uma `std::string`.
4. Validador: função `bool isPrime(int n)` e reutilize na Aula 3.

🔥 Desafio

Mini-calculadora modular: criar funções `somar`, `subtrair`, `multiplicar`, `dividir` e uma função `executar(int op, int a, int b)` que roteia a operação.

🏠 Katas

- Reescreva 3 exercícios da Aula 3 em funções puras (sem `cout` dentro da função, apenas retorno).

👩‍🔬 Aula 5 — Escopo, `static`, `.h` e organização

🎯 Objetivo

Entender escopo de variáveis, tempo de vida, `static` local, e separar código em .h / .cpp.

🧠 Teoria (5 min)

- Escopo de bloco: variáveis existem dentro de `{}`.
- Sombras (shadowing): variáveis internas escondem externas com mesmo nome.
- `static` local: mantém valor entre chamadas.
- Organização:
  - Arquivo de cabeçalho (`.h`): declarações.
  - Arquivo fonte (`.cpp`): definições.
  - `#include "arquivo.h"` para incluir seu cabeçalho.

💻 Exemplo — contador com `static`

```
#include <iostream>
using namespace std;

void tocar() {
    static int vezes = 0; // preserva valor entre chamadas
    ++vezes;
    cout << "Tocou " << vezes << " vez(es)\n";
}

int main() {
    tocar(); tocar(); tocar();
}
```

💻 Exemplo — organização em múltiplos arquivos

math_utils.h

```
#ifndef MATH_UTILS_H
#define MATH_UTILS_H

int max3(int a, int b, int c);
bool isPrime(int n);

#endif
```

math_utils.cpp

```
#include "math_utils.h"
#include <cmath>

int max3(int a, int b, int c) {
    int m = (a > b ? a : b);
    return (m > c ? m : c);
}

bool isPrime(int n) {
    if (n < 2) return false;
    for (int i = 2; i * i <= n; ++i)
        if (n % i == 0) return false;
    return true;
}
```

main.cpp

```
#include <iostream>
#include "math_utils.h"
using namespace std;

int main() {
    cout << "max3(7, 2, 9) = " << max3(7,2,9) << "\n";
    cout << "isPrime(29) = " << (isPrime(29) ? "true" : "false") << "\n";
}
```

Compilação (Dev-C++ ou g++):
`g++ main.cpp math_utils.cpp -o app`

🧩 Exercícios

1.Reorganize sua mini-calculadora (Aula 4) em `.h/.cpp` com funções separadas.

2.Crie uma função `contChamadas()` com `static` que retorna quantas vezes já foi chamada, e demonstre no `main`.

3.Implemente `int clamp(int x, int min, int max)` em um `.h/.cpp` e use em outro arquivo.

🔥 Desafio

Mini-projeto “Boletim” modularizado:

•`aluno.h/.cpp` com funções: `media3(double a,b,c)`, `conceito(media)`, `aprovado(media)`.

•`main.cpp` lê 3 notas, imprime média, conceito e situação.

🏠 Katas

•Refatore um exercício antigo criando um módulo utilitário (`utils.h/.cpp`) para entrada validada.

🎮 Projetinhos Sugeridos (Codemancer)

1. Jogo “Adivinhe o Número”
Gera número aleatório 1–100; usuário tenta; dica “maior/menor”; conta tentativas.

2. Caixa Eletrônico Simplificado
Menu com saldo, depósito, saque (validando saldo); usa funções e loop principal.

3. Conversor de Bases
Decimal ↔ Binário/Hex; modularizar em `conversor.h/.cpp`.

4. Gerenciador de Tarefas no Console
Adicionar/listar/remover itens; armazenar em `std::vector<std::string>`.

Assinado por:
Zahroniel Syrran & Kael’Aran