Universidade Federal do Ceará

Sistemas e Mídias Digitais
Lista 2 – Exercícios de Programação I
Prof. George Gomes

1. Objetivo: Da Lógica à Interatividade. O foco agora é dar ao seu código a capacidade de tomar decisões. Esta lista introduz as estruturas condicionais (if/else), que permitem que seus programas analisem os valores guardados em variáveis (como mouseX, mouseY, ou variáveis que você mesmo cria) e respondam a diferentes situações. O objetivo é criar *sketches* que não são apenas visuais, mas também reativos e dinâmicos.

2. O Poder das Condições Compostas. Raramente uma decisão depende de apenas um fator. Por isso, exploraremos os operadores lógicos E (&&), OU (||) e NÃO (!). Eles são as ferramentas para construir regras complexas, como "verificar se a variável *mouseX* está dentro de um limite E a variável *mouseY* está dentro de outro". Dominar esses operadores é fundamental para criar interações precisas.

3. Variáveis como Memória do Programa. Usaremos variáveis para dar memória ao nosso programa. Uma variável pode guardar um contador de cliques, a velocidade de um objeto, ou o estado atual de um sistema (ex: verde, amarelo, vermelho). Em especial, as variáveis de estado (booleanas) são cruciais para controlar comportamentos complexos, como um interruptor (ligado/desligado), permitindo que o programa "lembre" de uma escolha do usuário entre um quadro e outro.

4. Depuração e Lógica: Encontrando o Erro. Programar também é investigar. Alguns exercícios desta lista, como o 9, apresentam códigos com falhas lógicas propositais. O objetivo é treinar sua capacidade de analisar uma expressão condicional, entender por que ela não funciona como esperado e corrigi-la. Essa é uma das habilidades mais importantes no desenvolvimento de software.

5. Compartilhe! Desenvolva cada item em um sketch distinto no Processing. Compartilhe as soluções no Discord para que possamos discutir diferentes abordagens para os mesmos problemas lógicos

Exercício 1 – Quadrado Central

Crie um sketch em que um quadrado de 80 × 80 px aparece exatamente no centro da janela, independentemente dos valores passados a *size()*. Use *width* e *height* para calcular a posição. Objetivo: Refinar o uso das variáveis de sistema *width/height* e consolidar o conceito de coordenadas relativas.

Exercício 2 – Grade 3 × 3

Em uma janela quadrada, desenhe nove círculos idênticos formando uma grade 3 × 3 perfeitamente uniforme. O diâmetro de cada círculo deve ser 1/3 do tamanho da tela. Calcule automaticamente as posições para reforçar raciocínio de proporção espacial. Objetivo: Planejar posições relativas e praticar o mapeamento da tela em nove regiões equivalentes.

Exercício 3 – Alvo Concêntrico Retangular

Construa três círculos concêntricos coloridos. O maior deve ocupar 100 % da largura mínima entre *width* e *height*; o segundo, 66 % desse valor; o terceiro, 33 %. Escolha cores distintas para facilitar a percepção das camadas. Objetivo: Aplicar proporções e teste do mínimo de *width* e *height* para evitar distorção.

Exercício 4 – Faixas Diagonais com *quad*()

Usando exclusivamente a primitiva *quad*(), crie quatro paralelogramos inclinados aproximadamente 45 ° que recubram a área da tela como listras. É proibido empregar rotate(); todas as coordenadas devem ser calculadas a partir da primitiva geométrica *quad()*. Pode usar uma janela quadrada para facilitar. Objetivo: Explorar coordenadas absolutas e relações entre vértices de um paralelogramo.

Exercício 5 – Depuração de Tipos e Divisão Inteira

Analise o código abaixo:

```
void setup() {
  size(300, 300);
  noStroke();
  int cx = 150.5;          // (A)
  int cy = 150;
  int diametro = 50 / 100; // (B)
  fill(0);
  ellipse(cx, cy, diametro, diametro);
}
```

1. (A) O programa executa? Se não, indique o erro e corrija.
2. (B) Caso execute, explique por que o círculo não aparece na tela.
3. Ajuste o código para que o círculo preto de diâmetro 50px seja exibido corretamente no centro.

Objetivo: Reconhecer erros de tipagem, compreender divisão inteira versus ponto-flutuante e consolidar correções mínimas.

Exercício 6 – Detector de Quadrante

Implemente um sistema que imprime no console “NE”, “NO”, “SE” ou “SO” conforme o ponteiro do mouse esteja no quadrante respectivo da janela (dividida ao meio vertical e horizontalmente). Utilize apenas uma cadeia de *if/else if*.

Objetivo: Praticar condicionais compostas e raciocínio espacial.

Exercício 7 – Corredor Vertical

Crie uma faixa horizontal de 80 px de altura centrada verticalmente na janela (40 px acima e 40 px abaixo da linha y = height / 2).

Exiba a palavra “DENTRO” se o ponteiro do mouse estiver dentro dessa faixa; caso contrário, exiba “FORA”.

Condição sugerida: *if (mouseY >= height/2 - 40 && mouseY <= height/2 + 40)*

Objetivo: Reforçar o raciocínio sobre limites superiores e inferiores em um único eixo, usando operadores relacionais e lógicos.

Exercício 8 – Semáforo de Áreas

Pinte o fundo da janela conforme a posição do mouse:

- Verde: dentro do quadrado centralizado na tela;
- Amarelo quando o ponteiro estiver na faixa de 100 px ao redor desse quadrado;
- Vermelho em qualquer outra região.

Objetivo: Dominar múltiplas faixas condicionais utilizando *if / else if*.

Exercício 9 – Equivalência Booleana

O código a seguir tenta detectar se o ponteiro do mouse está dentro do retângulo cujos cantos opostos são (100, 100) e (200, 200). Contudo, a condição não funciona corretamente:

```
if (mouseX > 100 && mouseY > 100 || mouseX < 200 && mouseY < 200) {
  println("DENTRO");
} else {
  println("FORA");
}
```

- Explique por que a expressão permite que mensagens erradas sejam exibidas.
- Reescreva a condição lógica de forma concisa, garantindo que DENTRO apareça apenas quando o cursor estiver efetivamente dentro do retângulo.

Objetivo: Compreender a os operadores && e || e formular corretamente condições compostas.

Exercício 10 – Máquina de Estado simples

Coloque quatro retângulos fixos lado a lado. As teclas 1–4 devem exibir apenas um retângulo por vez. A tecla 0 restaura os quatro. Cada retângulo tem uma variável booleana de visível (verdadeiro ou falso).

Desafio extra: Modifique a lógica para que cada retângulo possa ser alternado independentemente, permitindo combinações simultâneas:

- Pressionar 1, 2, 3 ou 4 deve alternar (ligar/desligar) apenas o retângulo correspondente, sem alterar os demais.
- Pressionar 0 continua habilitando os quatro retângulos de uma só vez.

Objetivo: Praticar o controle de visibilidade com variáveis booleanas independentes, reforçando o uso de operadores lógicos e entradas de teclado sem recorrer a estruturas de dados compostas.

Exercício 11 - Interruptor Lógico (Toggle)

Desenhe um único retângulo que funcionará como um interruptor. Crie uma variável booleana global (ex: boolean ligado = false;). A lógica deve ser a seguinte:

- Se o usuário clicar com o mouse dentro do retângulo, o estado da variável ligado deve ser invertido. A forma mais concisa de fazer isso é com a expressão: *ligado = !ligado;*
- A cor do retângulo deve refletir o estado desta variável: uma cor clara quando ligado for *true* e uma cor escura quando for false.

Objetivo: Introduzir o operador de negação **!** como uma ferramenta eficaz para alternar estados, um padrão fundamental em programação interativa.

Exercício 12 – Zonas Excludentes

Divida a largura da tela em três faixas verticais de tamanhos idênticos. O fundo da tela deve reagir à posição do mouse da seguinte forma:

- Se o mouse estiver na faixa da esquerda OU na faixa da direita, o fundo da tela deve ficar branco.
- Se o mouse estiver na faixa central, o fundo deve permanecer preto.
- Não desenhe as linhas divisórias, a mudança de cor deve ser o único feedback visual.

Objetivo: Praticar o uso do operador || para agrupar condições espaciais não contíguas, ativando uma mesma resposta para diferentes áreas da tela.

Exercício 13 – Movimento com teclado e limites de tela

Controle um quadrado com WASD e setas. A velocidade é constante e o objeto não pode sair da janela (use verificações de borda). Pressionar a tecla espaço para centralizar o objeto.

Objetivo: Praticar eventos de teclado, atualização incremental de posição e limites de tela (0..width, 0..height).

Exercício 14 – Contador Módulo 12

Mostre na tela um contador que incrementa +1 a cada clique e retorna a 0 após atingir 11. Utilize o operador % para garantir o ciclo.

Objetivo: Introduzir aritmética modular e variáveis globais.

Exercício 15 – Corrida Horizontal

Bolinha parte da borda esquerda e move-se autonomamente até alcançar a borda direita. Ao tocar, exiba "Vencedor!". Pressione a barra de espaço para reiniciar a posição.

Objetivo: Loop de jogo simples, reinicialização por evento de teclado.

Exercício 16 – Bolinha Acelerada

Crie um programa que exiba um círculo quicando nas bordas da tela. Utiliza duas variáveis para controlar a velocidade do círculo no eixo x (speedX) e no eixo y (speedY). Quando o círculo colidir com qualquer borda: (a) inverta a velocidade correspondente; (b) multiplique-a por 1.2.

Objetivo: Aplicar reflexão de velocidade e aceleração multiplicativa.

Exercício 17 – Semáforo Finite-State (Verde → Amarelo → Vermelho)

Construa um semáforo vertical com três círculos:

- 0 = verde aceso;
- 1 = amarelo aceso;
- 2 = vermelho aceso.

Controles de teclado:

- Tecla UP → se estado < 2, incremente estado em 1 (verde → amarelo → vermelho).
- Tecla UP → se estado == 2, estado = 0 (vermelho → verde).

Implemente a troca apenas com operadores condicionais.

Objetivo: Modelar máquina de estados simples controlada por teclado.

Exercício 18 – Botão Hover & Clique

Desenhe um botão retangular. Quando o mouse estiver sobre ele, clareie a cor (efeito *hover*). Ao clicar dentro, altere o *background* para uma cor aleatória. Fora do botão, o fundo deve permanecer neutro.

Objetivo: Detectar região do mouse, estados booleanos e *feedback* visual.

Exercício 19 – Teste de Reação

A ideia é calcular quanto tempo o usuário demora para clicar quando a cor da tela muda. Após um tempo aleatório entre 1 e 4 s, o fundo troca de cinza para verde. Calcule o tempo (ms) entre a troca e o clique do usuário e mostre no console.

Objetivo: Manipular *millis*() e medir intervalos de tempo.

Exercício 20 – Jogo "Clique no Quadrado"

Um quadrado azul surge em posição aleatória. O jogador dispõe de 2000 ms para clicar nele. A cada acerto: pontuação incrementa e o tempo limite é reduzido em 10 %. Ao clique errado ou ao exceder o tempo, exiba total de acertos e duração da partida.

Objetivo: Integrar aleatoriedade, temporização decrescente, controle de estados e pontuação.