**Paradigma Estruturado**

O paradigma estruturado organiza o programa em blocos de código que seguem um fluxo lógico definido por estruturas de controle como laços (for, while) e condicionais (if, else). O objetivo é tornar o código mais claro e compreensível. Neste contexto, **sem o uso de funções**, todo o código é escrito diretamente na sequência, sem modularização.

**Características principais:**

O programa é composto por uma sequência de instruções executadas linearmente.

Uso extensivo de laços e condicionais para controle de fluxo.

A lógica é implementada diretamente, sem encapsulamento.

**Paradigma Funcional**

O paradigma funcional trata a computação como a avaliação de funções matemáticas, sem alterar estados ou usar variáveis mutáveis. Ele se baseia em funções puras e evita efeitos colaterais.

**Características principais:**

Uso intensivo de funções puras (mesmo input, mesmo output).

Evita mutabilidade e estados globais.

**Vantagens e Desvantagens**

**Paradigma Estruturado**  
**Vantagens:**

Mais simples para problemas menores, já que tudo é implementado linearmente.

Fácil de entender para iniciantes, pois o fluxo do código é direto.

Reduz a necessidade de abstração, já que não há subdivisão em funções.

**Desvantagens:**

Torna-se desorganizado e difícil de manter à medida que o programa cresce.

Dificulta o reaproveitamento de código.

Reduz a legibilidade ao misturar lógica e controle de fluxo.

**Paradigma Funcional**  
**Vantagens:**

Código mais conciso e declarativo.

Redução de efeitos colaterais, melhorando a confiabilidade.

Facilidade de paralelização devido à imutabilidade.

**Desvantagens:**

Curva de aprendizado mais acentuada para iniciantes.

Dificuldade em aplicar em problemas com lógica sequencial complexa.

Algumas linguagens funcionais têm menos suporte de ferramentas e bibliotecas.

Soma dos quadrados em Kawa:

C:\Users\leont\Documents\Faculdade Ciências da Computação\ParadigmasComputacao\kawa-3.1.1\lib>java -jar kawa.jar

#|kawa:1|# (define (soma-dos-quadrados lista) (apply + (map (lambda (x) (\* x x)) lista)))

#|kawa:2|# (display (soma-dos-quadrados '(1 2 3 4)))

30

Filtrar pares em Kawa:

C:\Users\leont\Documents\Faculdade Ciências da Computação\ParadigmasComputacao\kawa-3.1.1\lib>java -jar kawa.jar

#|kawa:1|# (define (filter pred lista)

(if (null? lista)

'()

(if (pred (car lista))

(cons (car lista) (filter pred (cdr lista)))

(filter pred (cdr lista)))))

#|.....2|# #|.....3|# #|.....4|# #|.....5|# #|.....6|# #|kawa:7|#

#|kawa:8|# (define (filtro-pares lista) (filter (lambda (x) (= (modulo x 2) 0)) lista))

#|kawa:9|# (display (filtro-pares '(1 2 3 4 5 6)))

(2 4 6)

**Comparação de Código**

**Qual paradigma resultou em um código mais conciso e fácil de entender?**

O **paradigma funcional** resultou em um código mais conciso, especialmente no caso do filtro de números pares. Isso se deve ao uso de abstrações como a função filter e funções lambda, que encapsulam a lógica de forma declarativa.

O código no paradigma **estruturado** tende a ser mais detalhado e explícito, com iterações e condicionais separadas, o que pode torná-lo mais longo.

Assim, em termos de clareza, considero o estruturado mais fácil de entender por usar uma linguagem mais conhecida (Python), além de uma sintaxe familiar entre as principais linguagens de programação. Porém em termos de concisão, a funcional usando kawa ganhou pelo uso de funções de ordem superior.

**Qual paradigma foi mais fácil de implementar para você? Justifique.**

O paradigma **estruturado** foi mais fácil de implementar, pois segue uma lógica sequencial clara e não depende de abstrações ou de bibliotecas adicionais, ou implementações extras como a função filter no funcional.

O paradigma funcional pode ser mais complexo inicialmente, especialmente se o ambiente (como Kawa) não suporta diretamente funções comuns ou exige configurações extras.

Além disso Kawa não é muito conhecido e não utilizei IDE, fora a utilização de muitos parenteses que gera bastante confusão podendo resultar em erros de sintaxe.

**Como o paradigma estruturado aborda a solução dessas tarefas? Compare com a abordagem funcional.**

No paradigma estruturado, o problema é resolvido de maneira sequencial e explícita, utilizando laços (for) e condicionais (if). Cada etapa do processamento é claramente definida no código.

No paradigma funcional, o foco está em descrever *o* que deve ser feito, em vez de como. Por exemplo, ao usar filter e lambda, descrevemos diretamente a condição de filtragem, sem nos preocupar com o controle explícito de laços.

**Comparação:**

O paradigma estruturado oferece maior controle sobre o fluxo de execução, o que pode ser vantajoso para tarefas específicas ou complexas.

O paradigma funcional é mais expressivo para operações sobre coleções de dados, como mapeamentos e filtragens.

**Se fosse necessário alterar a funcionalidade para trabalhar com números negativos, qual código seria mais fácil de modificar? Por quê?**

Considerando alguém com igual domínio sobre ambas as linguagens, o paradigma funcional seria mais fácil de modificar. Alterar a lógica para considerar números negativos geralmente envolve apenas ajustar o predicado na função lambda, sem precisar mexer no fluxo geral do programa. No paradigma estruturado, seria necessário ajustar as condições dentro do laço explícito, o que pode levar a uma maior probabilidade de erros, especialmente se o código não estiver bem organizado.

O paradigma funcional é mais adequado para modificações rápidas e locais, enquanto o estruturado exige maior cuidado para evitar problemas ao alterar o fluxo do programa.