**Avaliação e Comparação das Estruturas de Controle, Unidades de Programa, Comandos, Gerenciamento de Memória e Desempenho em Linguagens de Programação: Uma Análise entre Imperativa, Orientada a Objetos, Lógica e Funcional**

**Introdução**

Nesse trabalho foi feito a avaliação e comparação de diferentes tipos de paradigmas de programação, que foram: Imperativa, Orientada a Objetos, Lógica e Funcional. As linguagens utilizadas foram: C, Java, Prolog e Scala. Neste trabalho também foi feita a implementação de códigos que resolvem problemas de forma mais eficiente em cada um dos paradigmas listados.

**Análise das áreas em cada uma das linguagens selecionadas**

**Estruturas de Controle**

**Estruturas de controle em C**, oferece estruturas condicionais, incluindo entre elas (for, while, do-while), e também estruturas de repetição que incluem (if, else, switch). Essas estruturas citadas são utilizadas para controlar o fluxo de execução do programa**. Em Java**, é oferecido estruturas bem semelhantes das existentes em C, incluindo os loops (for, while) e condicionais (if, else, switch), elas são também utilizadas da mesma forma. Já **estruturas de controle em** **Prolog** é utilizado regras que verifica a correspondência de padrões e unificações para controlar o fluxo que o programa segue, ele não possui loops nem condicionais. O fluxo é exclusivamente conduzido pelas regras definidas. Como em C e Java, **estruturas de controle em** **Scala** também possui as estruturas de controle comuns, mas ele incentiva o uso de funções de ordem superior para fazer o controle do seu fluxo.

**Unidades de Programa**

As unidades de programa em **C** são organizadas em funções, que são blocos de códigos que são reutilizáveis e que podem ser chamadas em várias partes do programa. Já em **Java** as unidades de programa são definidas como classes e métodos. As classes é onde fica a base da programação orientada a objetos, já os métodos são as funções que são definidas para essas classes. Na linguagem **Prolog**, suas unidades de programa são conhecidas como predicados, seus programas são compostos pela estrutura de fatos e regras, que fazem a definição de relações logicas. Em **Scala**, são funções e classes, ele suporta a programação orientada a objetos e funcional, dessa forma permitindo a definição de funções de primeira classe.

**Comandos**

Em C é oferecido uma grande opção de comandos, entre eles a manipulação de strings, manipulação de entradas/saídas e gerenciamento de exceções, através de construções como try, catch (em C++). Já **Java** oferece recursos bem mais avançados de manipulação de strings, de entradas e saídas, utilizando para isso classes como Scanner e System.out. Ele também faz o gerenciamento de erros/exceções com try, catch, finally para fazer o tratamento dos erros. Os comandos em **Prolog** são utilizados as correspondências de padrões para se executar as operações, como manipulação de listas e árvores. Suas entradas e saídas são tratadas com predicados de consultas e respostas. Em **Scala** é oferecido recursos para fazer a manipulação de strings, entradas e saídas, gerenciamentos de exceções e incluindo expressões lambda e funções de alta ordem.

**Gerenciamento de memória**

Em **C**, o gerenciamento de memória é feito pelo programador, sendo necessário você alocar e desalocar de forma explica a memória, fazendo o uso de funções como malloc, free, ou operadores em C++ como new e delete. Dessa forma sendo possível o controle direto da memória, mas isso também pode levar a vazamentos caso não seja tratado de forma correta. Na linguagem **Java**, o gerenciamento de memória é gerenciado de forma automática, permitindo que o garbage collector cuide automaticamente da alocação e desalocação de memória, dessa forma reduzindo os riscos de vazamentos de memória, seu ponto negativo é que pode introduzir uma sobrecarga no sistema. Já em **Prolog** é interno ao interpretador ou complicador, dessa forma, ele lida com a alocação e desalocação de memória automaticamente, isso não sendo uma preocupação ao programador de forma explicita. Em **Scala**, como ele é executado em uma JVM, ele possui o mesmo gerenciamento de memória do Java.

**Desempenho**

Foi realizado o teste de desempenho de todos os paradigmas com algoritmos parecidos, os testes foram: Teste de cálculos intensivos e teste de manipulação de grandes conjuntos de dados. A seguir falaremos sobre os resultados obtidos em cada um dos paradigmas.

**Teste de Cálculos intensivos**

Nesse teste foi utilizado um algoritmo que realizava a soma de números, esses números sendo de 1 a 1.000.000, tivemos os seguintes resultado

Linguagem Imperativa **C**: Nesse teste obtivemos um bom resultado, tendo seu tempo de execução de 0.001 segundos, dessa forma sendo perceptível sua boa capacidade de realizar esses cálculos.

Linguagem Orientada a Objetos (**Java**): Em java tivemos um resultado de 0.002 segundos, sendo bastante eficiente, mas com resultado inferior ao C

Linguagem Logica (**Prolog**): Em Prolog obtivemos um resultado de 1.170 segundos, demonstrando ter uma eficiência baixa comparado aos outros.

Linguagem Funcional (**Scala**): Na linguagem Scala tivemos um resultado de 0.026 segundos, sendo o terceiro colocado no teste de desempenho

**Teste de manipulação de grandes conjuntos de dados**

Nesse teste foi realizado um algoritmo que cria uma lista de valores aleatórios com o tamanho de 1.000.000 e tendo valores gerados de 0 a 4999. A lista e os valores são criados, após isso é verificado qual o maior valor presente na lista. Seus resultados foram:

Linguagem Imperativa (**C**): No teste em C foi obtido o resultado de 0.01 segundos, sendo bem eficiente, mas em comparação com o teste anterior teve um desempenho mais baixo.

Linguagem Orientada a Objetos (**Java**): Com o teste realizado em Java tivemos o resultado de 0.028 segundos, tendo um resultado comparativo parecido com o de C, foi eficiente, mas com um tempo de execução maior que o outro teste.

Linguagem Logica (**Prolog**): Em Prolog tivemos um resultado bem parecido com o teste anterior, atingindo o tempo de 1.878 segundos, sendo novamente o pior entre os paradigmas testados.

Linguagem Funcional (**Scala**): Nessa linguagem, o resultado novamente foi o classificado em terceiro lugar, tendo como tempo de execução 0.464 segundos.

**Resultado dos Testes**

Com os testes foi possível ver que em questão de tempo, a linguagem mais eficiente foi a Linguagem Imperativa (C), porem um ponto a se destacar, é que a sua implementação não foi a mais fácil a ser realizada no segundo teste, pois foi necessário fazer a manipulação de memória, enquanto nas outras linguagens não foi necessário, conforme essas analises, podemos concluir que a linguagem é altamente eficiente em cálculos e grandes conjuntos de dados porem de implementação mais complexa pelo tratamento da memória. Na Linguagem Orientada a Objetos, nossa segunda colocada me ambos os testes, teve um código de fácil implementação, de forma que podemos dizer que Java é uma linguagem que possui um bom desempenho em muitos cenários, como exemplo aplicação de sistemas de grande escala, porém ela possui um desenvolvimento inferior a C em cenários que se utiliza cálculos e manipulação de muitos dados. Em Prolog, uma linguagem logica, podemos afirmar de forma concisa que ela é extremamente eficaz para problemas lógicos, mas para problemas de cálculos e manipulação de dados ela não é uma linguagem adequada. Já na linguagem funcional Scala, temos um desempenho um pouco parecido com Java, pois ela é executada também em uma JVM, porém a programação funcional tem um desempenho melhor em algoritmos concisos e eficientes, o que não foi exatamente o que testamos, fazendo ela ser inferior ao Java.

Com esses resultados podemos afirmar que para os exemplos testados, C teve um melhor desempenho, o que torna ela melhor nesses cenários. Porém as outras linguagens têm também aplicações excelentes em outros contextos, como já foi citado anteriormente. Diante disso não é possível dizer qual a linguagem mais eficiente em geral, e sim dizer qual é mais eficiente em diferentes exemplos de algoritmos