Avaliação de critérios de linguagens de programação na realização de inversão de matrizes: C++ e R

Thaissa Vitória Guimarães Daldegan de Sousa[1]

Resumo

O presente artigo tem objetivo de esclarecer os critérios e características de cada um deles no projeto do desenvolvimento de software, é usado um problema com uma grande quantidade de dados matriciais dos quais deseja se obter a inversa de matrizes de ordem 12. Desenvolvidos em C++ e R, os algoritmos são analisados de acordo com cada critério e é comentada sobre a escolha de linguagem para o desenvolvimento desse e de outros projetos.

Palavras-chave: R, CPP, C++, inversa, matriz.

Abstract

This article aims to qualify the requirements and characteristics of each one of them developed in the software development project, a problem with a large amount of matrix data is used, from which you want to obtain an inverse of matrices of order 12. Developed in C++ and R , the organizers are commented according to each criterion and it is about the choice of language for the development of this and other projects.

Keywords: R, CPP, C++, inverse, matrix.

1. Introdução

Devido ao avanço das tecnologias de informação e dos problemas computacionais que emergem delas, a aplicação de softwares para resolução eficiente e precisa desses problemas se tornou algo indispensável.

A realização de operações matriciais, muito utilizadas em aplicações bancárias e comerciais, está entre as áreas em que os softwares são aplicados. Pela demanda de resultados rápidos e confiáveis, no momento do planejamento desse tipo de projeto de software, a escolha de uma linguagem de programação (LP) deve passar pela análise de diversos critérios.

No presente trabalho, tomou-se como exemplo a realização de uma operação de inversão de matrizes com uma grande quantidade de entrada de dados. Foram

[1] Graduando em Engenharia de Computação pelo CEFET-MG. E-mail: thaissacefetmg@gmail.com

escolhidas para o desenvolvimento do problema as linguagens R e C++, as quais passaram por uma avaliação de critérios que serão discutidos nos resultados.

O objetivo da pesquisa é compreender sobre a avaliação de fundamentos das linguagens, em especial das escolhidas (R e C++) e definir os pontos fortes e fracos da aplicação das duas na resolução de problemas matriciais.

2. Fundamentação teórica

Como dito anteriormente, no momento do desenvolvimento do projeto do software, devem ser avaliados critérios importantes, são eles: a legibilidade, a facilidade de escrita, a confiabilidade e o custo. Esses critérios são afetados por diversas características, das quais iremos detalhar posteriormente.

Segundo SEBESTA (2011), é praticamente impossível fazer dois cientistas da computação concordarem com o valor de certas características de linguagens em relação às outras, mas ambos concordariam que a avaliação delas é importante.

- **Legibilidade**
  - **Simplicidade**

    A simplicidade influencia principalmente na legibilidade do programa, seus três problemas potenciais são a multiplicidade de recursos para uma mesma operação, o número de construções básicas que a linguagem apresenta e a sobrecarga de operadores, onde um operador pode ter mais de um significado.

  - **Ortogonalidade**

    Ortogonalidade é a capacidade que as construções primitivas de uma linguagem têm de, se combinadas a um conjunto de formas, gerar estruturas de dados e controles da linguagem. É fortemente ligada a simplicidade, quanto mais ortogonal, mais fácil será o aprendizado, leitura e entendimento da linguagem.

  - **Tipos de dados**

    Essa característica se refere a presença de tipos de variáveis adequados para solução de um problema. Afeta também na legibilidade, pois, se não há um tipo adequado para solução de determinado problema, as alternativas que o

desenvolvedor de software pode ter, arriscam-se a serem confusas para outros programadores.

- **Projeto de sintaxe**

O formato dos identificadores de funções e variáveis, a forma de palavras especiais (exemplo: for,while e class) e a apresentação de sentenças de maneira que seja mais fácil inferir o significado delas são considerações do projeto de sintaxe, que é outra característica que interfere na legibilidade de um programa.

- **Facilidade de Escrita**

Além de também ser afetadas pela simplicidade e ortogonalidade, a facilidade de escrita também é interferida por outros fatores, como o suporte à abstração e a expressividade.

- **Suporte para abstração**

Abstração é a possibilidade de definir estruturas ou operações complicadas de forma a permitir que muitos detalhes sejam ignorados (SEBESTA,2011). Podem ser de processos, como uso de subprogramas em algoritmos repetitivos e de dados, como o CPP oferece para aplicação da estrutura de dados árvore.

- **Expressividade**

Conjunto de artifícios de uma linguagem que permitem uma escrita mais fácil e didática, interferindo na facilidade de escrita e na legibilidade.

- **Confiabilidade**

Diretamente afetada pelos dois critérios anteriores, a confiabilidade diz respeito ao nível de segurança de resultados e métodos de uma linguagem. Outras características que são analisadas quando se está avaliando a confiabilidade é a cerificação de tipos, o tratamento de exceções e os apelidos restritos.

- **Verificação de tipos**

Execução de testes para detectar erros de tipos de funções e variáveis de um programa. Podem ser em tempo de execução ou compilação, mas o segundo é mais desejável por apresentar um custo menor.

- **Tratamento de exceções**

Segundo SEBESTA (2011), é a habilidade de um programa de interceptar erros em tempo de execução, fazer as correções necessárias e continuar.

- **Apelidos restritos**

O uso de nomes diferentes para acessar uma mesma célula de memória pode ser perigoso para a confiabilidade do programa, pois depende da memória do desenvolvedor para que, na hora de alterar ou excluir qualquer um deles, lembrar dos outros.

- **Custo**

O custo é um critério extremamente amplo, pois pode se referir ao financeiro, como por exemplo o de treinar programadores, temporal, como o de escrever os programas e executar, o de hardware necessário para compilação do programa, entre outros. Normalmente, todos esses pontos de vistas de análise de custo estão ligados entre si ou entre os outros critérios de avaliação de linguagem, o que torna a análise se custos extensa e complexa.

3. Metodologia

- **Interfaces, compilação e execução**

Para o C++ a IDE utilizada foi o Visual Studio Code, que apresenta uma interface organizada e que conta com diversas extensões que podem ajudar no momento da escrita e organização do código. Com a ajuda de um arquivo Makefile (que automatiza o processo de construção de aplicações), permite que o programa seja executado pelo terminal da IDE usando apenas o comando “make r”.

Na linguagem R, foi necessária a instalação do software R e da interface RStudio, própria para programação em R. Depois de escrito o código, bastou clicar no botão “run”, presente na parte superior direita de onde o código é escrito para que ele fosse executado.

- **Bibliotecas utilizadas**

  Em R, foram utilizadas as bibliotecas gratuitas *readxl*, para leitura do arquivo xlsx e *dplyr*, para manipulação dos dados. Antes de utilizá-las, foi necessário instalá-las no próprio terminal do RStudio com o compando *install.packages("nome_da_biblioteca").* Depois bastou incluí-las no código com o *library(nome_da_biblioteca).*

  Em C++ foram utilizadas diversas bibliotecas gratuitas:

  - iostream, para executar as entradas e saídas do programa;
  - string, para que fosse possível usar o tipo string;
  - fstream, para trabalhar com o arquivo, que teve que ser convertido para csv pela falta de abstração da linguagem CPP para arquivos xlsx;
  - sstream, para fornecer classes de stream de strings;
  - iomanip, para ajudar na coleta de dados importantes;
  - vector, para que fosse possível utilizar o tipo vector;
  - math.h, para facilitar operações matemáticas ;
  - chrono, para coletar o tempo de execução.

- **Lógica**

  Em R, em um único arquivo, a função *sys.time()* é utilizada para coletar o tempo final e inicial da execução do programa. Ele inicia lendo a planilha por meio da função *read_excel* e salvando em uma variável, a primeira coluna e a primeira linha da tabela são retiradas e utilizando a função *solve*, se calcula e imprime a inversa da matriz. Depois, usa-se as funções *slice* e *add_row* para tirar a primeira linha e adicionar a próxima, respectivamente, e em loop, vai calculando-se e mostrando na tela as inversas dessas matrizes, até que o arquivo acabe. Depois de tudo, é mostrado o tempo de execução do programa.

  Em C++ foram criados três arquivos, o principal (*main.cpp*) e os que definem e realizam as operações (*funções.cpp* e *funções.hpp*). São combinados dois *vectors* para se criar uma estrutura bidimensional que será a matriz e a matriz identidade dela (necessária para o cálculo da inversa). A função *leArquivo*

realiza todos os procedimentos com a ajuda dos subprogramas *matrizInversa* (que gera a inversa por meio do método de Gauss-Jordan) e *print* (que imprime as matrizes).

O arquivo csv é aberto e tokenizado e convertido para double, linha por linha, enquanto é salvo no vector bidimensional (matriz). Um contador de linhas é incrementado e quando chega no marco da primeira matriz pronta, ou seja, quando atinge o valor 12, calcula a inversa da primeira e continua-se o loop, porém a partir desse ponto, retirando a primeira linha da matriz e adicionando a próxima do arquivo na matriz.

A função *matrizInversa* retorna um inteiro que se refere a quantidade de matrizes que apresenta um NaN (Not a Number) como resultado da inversão, ou seja, a quantidade de matrizes que não podem ser invertidas. Esse número é mostrado no final da execução, juntamente com o tempo coletado a partir da função *steady_clock::now* da biblioteca *chrono*.

4. Resultados

A linguagem R é uma linguagem multiparadigma, interpretada e fracamente tipada, que apresenta diversos recursos estáticos para manipulação da planilha submetida para o problema. O desafio inicial da linguagem foi aprender sobre ela, pois, por não ser uma linguagem não muito difundida na área de aprendizado acadêmico, foi necessário um treinamento do zero para o desenvolvimento do aplicativo.

Na questão dos critérios de avaliação de linguagem podemos avaliá-la da seguinte maneira:

- Legibilidade: a linguagem é de simples compreensão, por oferecer os tipos de dados necessários para realização do problema, sua legibilidade não foi comprometida.

- Facilidade de Escrita: não foi necessário uso de subprogramas e as estruturas básicas da linguagem ofereceram um ambiente de programação didático e simples para o desenvolvimento.

- Confiabilidade: o programa verifica os tipos e as exceções, porém não consegue corrigir os problemas e continuar a execução. Quando uma matriz não é invertível, o programa para de executar e informa o erro.

- Custo: o custo de tempo de execução foi em torno de 46 segundos. Outros custos relevantes foram o de tempo para o aprendizado da linguagem e instalação dos programas necessários.

Já a linguagem C++ é uma linguagem também multiparadigma (podendo suportar linguagens imperativas, orientada a objetos e genérica), compilada e fortemente tipada. O desafio da linguagem foi a criação de abstrações para a manipulação de arquivos, já que as bibliotecas gratuitas disponíveis não apresentavam as soluções para esse tipo de problema.

Na questão dos critérios de avaliação de linguagem podemos avaliá-la da seguinte maneira:

- Legibilidade: a linguagem é de difícil compreensão, a grande variedade de estruturas básicas disponíveis pode comprometer a legibilidade do programa, os tipos de variáveis também necessitaram de combinações para atender as questões solicitadas. Porém houve um esforço da minha parte para nomear as funções de forma que a funcionalidade de cada uma fosse clara. A compreensão do código é difícil, principalmente para os programadores não habituados com a linguagem.

- Facilidade de Escrita: abusou-se do uso de subprogramas, mas a linguagem apresentou um bom ambiente para abstração de dados e funções, visto que eu já tinha conhecimento avançado da linguagem. Porém, é reconhecível que para quem está iniciando na programação, realizar essa tarefa em C++ seria extremamente difícil.

- Confiabilidade: o programa verifica os tipos e as exceções, consegue corrigir alguns erros e outros informa precisamente para o desenvolvedor onde o erro está sendo cometido. Quando uma matriz não é invertível, o programa imprime a inversa como um dado NaN (Not a Number).

- Custo: o custo de tempo de execução foi em torno de 12 segundos. Outros custos relevantes foram o de tempo para o desenvolvimento do algoritmo e

para correção de erros gerais relacionados a tipagem de variáveis (erros que não aconteciam no R).

5. Conclusão

Particularmente, preferi o desenvolvimento na linguagem C++, por ser uma linguagem fortemente tipada, por eu já estar habituada com ela e por apresentar melhor onde os erros estavam, permitindo uma correção mais precisa. Além de que, ela apresentou melhor tempo de execução e não encerrou o programa quando a matriz não era invertível.

Já com uma visão imparcial, em vista dos aspectos supracitados e dos critérios analisados, podemos perceber que as duas linguagens apresentam pontos negativos e positivos. Essa questão aparece em quase todos os projetos de desenvolvimento de software. Cabe ao programador, ou a empresa, escolher qual das linguagens é melhor para a seção do problema que deseja resolver de acordo com os recursos disponíveis.

6. Referências

SEBESTA, R. W. **Conceitos de Linguagens de Programação**. 9. ed. Porto Alegre: Bookman, 2011. 795p.