TRABALHO DE CONCEITOS DE LINGUAGENS DE PROGRAMAÇÃO

(Implementação da Eliminação de Gauss)

Cleber Farias Berndsen Junior
Gustavo Peres Fernandes
Hyhickle Ryozo Umetsubo Gonçalves
Igor Yuji Ishihara Sakuma

PELOTAS
2023

Compare o código das linguagens, diferenciando os tipos de dados, acesso às variáveis, organização de memória, chamadas de função e comandos de controle de fluxo:

- Tipos de dados:
C: Usa arrays para representar matrizes e vetores, usa ponteiros para alocar memória dinamicamente e realiza conversões explícitas de tipo.
Rust: Usa vetores para representar matrizes e vetores, usa referências para acessar dados e realiza conversões explícitas de tipo.
Go: Usa slices para representar matrizes e vetores, usa o operador "&" para obter um ponteiro para um valor e realiza conversões implícitas de tipo quando possível.

- Acesso às variáveis:
C: Usa o operador de índice [] para acessar elementos de arrays, usa o operador * para acessar o valor apontado por um ponteiro.
Rust: Usa o operador de índice [] para acessar elementos de vetores e usa o operador & para criar referências a valores.
Go: Usa o operador de índice [] para acessar elementos de slices e usa o operador & para obter um ponteiro para um valor.

- Organização de memória:
C: Usa ponteiros para alocar e desalocar memória dinamicamente e para acessar elementos de arrays e structs.
Rust: Usa o sistema de propriedade de memória para gerenciar a alocação e desalocação de memória e usa referências e ponteiros para acessar dados.
Go: Usa a coleta de lixo para gerenciar a alocação e desalocação de memória e usa slices para representar fatias de memória.

- Chamadas de função:
C: Usa protótipos de função para declarar funções e usa o operador de chamada () para invocar funções.
Rust: Usa declarações de função para declarar funções e usa o operador de chamada () para invocar funções.
Go: Usa declarações de função para declarar funções e usa o operador de chamada () para invocar funções.

- Comandos de controle de fluxo:
C: Usa comandos if/else, for e while para controlar o fluxo do programa.
Rust: Usa comandos if/else, for e while para controlar o fluxo do programa, bem como macros para gerar código condicional.
Go: Usa comandos if/else, for e switch para controlar o fluxo do programa, bem como o operador := para declarar e atribuir valores a variáveis.

Compare o código das linguagens com algumas métricas:

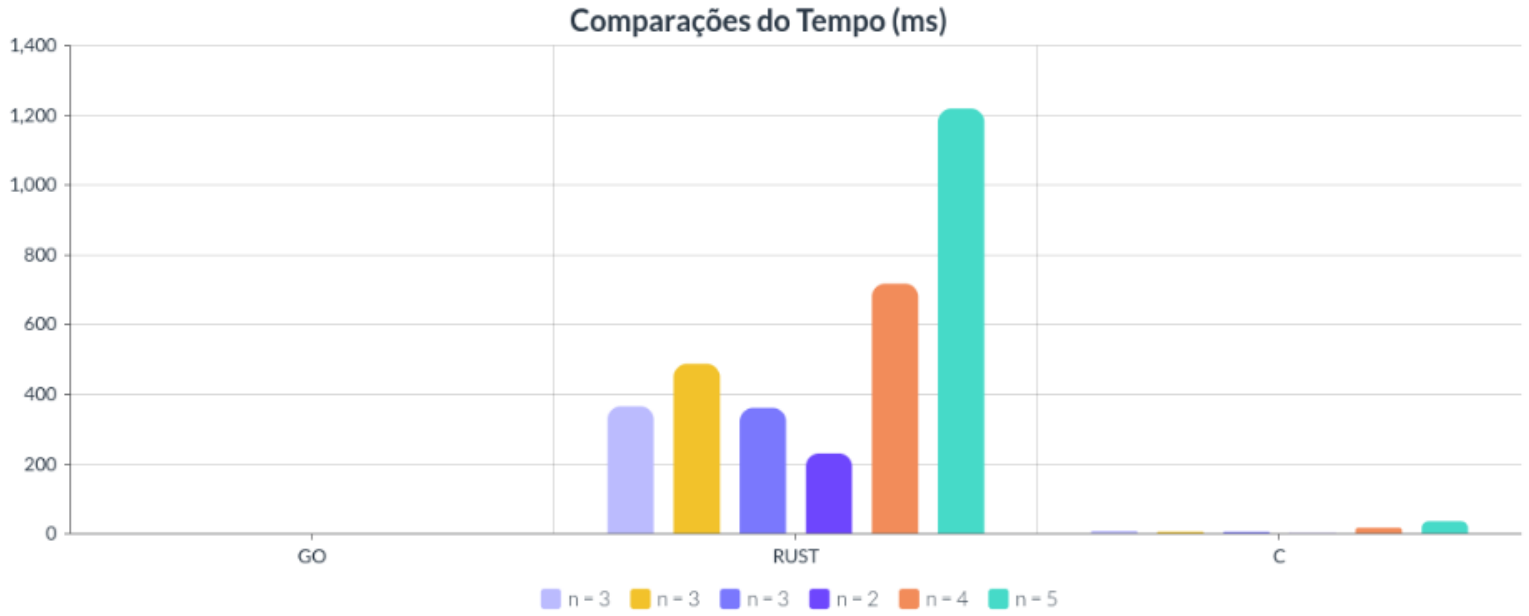
- Número de linhas (SLOC):
C: 123
Rust: 112
Go: 88

- Número de comandos:
C: 99
Rust: 100
Go: 66

- Número de bibliotecas utilizadas:
C: 5
Rust: 3
Go: 3

- Curva de Evolução:
C é uma linguagem mais antiga e estabelecida, que continua sendo relevante para certas aplicações.
Rust é uma linguagem mais moderna e em constante evolução e é mais conhecida por sua segurança e confiabilidade
Go é uma linguagem mais moderna e em constante evolução e é mais conhecida por sua facilidade de uso e desempenho

Compare o desempenho das linguagens, oferecendo uma comparação do tempo de execução nas diferentes linguagens, considerando diferentes tamanhos para a matriz trabalhada (apresenta na forma de tabela ou gráfico, não deixando de apresentar características da máquina utilizada - CPU, memória...):



Podemos simplificar a análise dos resultados, dada a grande discrepância, por meio de uma tabela:

1		GO	RUST	C
2	**n = 3**	0.01 ms	366.4 ms	9.68 ms
3	**n = 3**	0.01 ms	488.3 ms	7.50 ms
4	**n = 3**	0.009 ms	361.9 ms	7.26 ms
5	**n = 2**	0.005 ms	231.8 ms	3.58 ms
6	**n = 4**	0.012 ms	718.4 ms	18.25 ms
7	**n = 5**	0.018 ms	1220 ms	36.79 ms

Especificações da máquina utilizada:

CPU = Ryzen 7 5700 G
Memória = 2x 8GB 3200MHz DDR4

Conclusões:

Com base nos testes realizados, é possível observar que a linguagem Go teve um melhor desempenho na implementação da Eliminação de Gauss. No entanto, é importante destacar que esse resultado não significa que a linguagem Go sempre superará as outras duas linguagens, mas sim que depende do contexto e da aplicação em questão. A linguagem Go foi projetada desde o início para ter recursos de concorrência nativos, tornando mais fácil escrever código paralelo eficiente, o que pode explicar o seu bom desempenho na implementação da Eliminação de Gauss. Em contraste, as linguagens C e Rust são mais conhecidas por sua eficiência em algoritmos numéricos sequenciais.