Comparativo entre as implementações da Eliminação de Gauss nas linguagens C, Go e Rust

Os comparativos foram divididos em dois tópicos com a finalidade de facilitar a leitura e o entendimento de todos os itens. Por fim, este arquivo possui uma conclusão no qual o grupo buscou objetivamente analisar os resultados obtidos e concluir as vantagens e desvantagens de cada implementação.

Comparação entre as linguagens

Os dados utilizados na implementação em C foram ‘float’ e ‘int’, e arrays destes mesmos tipos citados. Em Rust utilizou-se ‘usize’ e ‘f32’, além de utilizar ‘Vec’. E, Go, utilizou dados ‘int’ e ‘float64’ e slices.

Tanto C quanto Go utilizaram tipos de dados muito semelhantes, além de “array” e “slice” que são, em termos de desempenho, também muito próximos. Porém a implementação em ‘Rust’ utilizou o tipo ‘Vec’, que, apesar de manter-se utilizando dados primitivos, assim como ‘C’ e ‘Go’, esse tipo é vetor dinâmico, que, em desempenho, se torna muito mais custoso que um array estático. Go, inclusive, não pode ter arrays que não sejam declarados com tamanhos estáticos, por isso foi utilizado o slice.

As linguagens C e Go são muito semelhantes nas implementações de controle de fluxo utilizando ‘for’, ‘if’ e ‘while’, e Rust apenas diferencia-se por utilizar, além das citadas (a sintaxe utilizada em rust é diferente da usada em C), o ‘switch’.

Em C, as variáveis globais são visíveis em todo o programa, a menos que sejam explicitamente declaradas como "static" dentro de um arquivo de origem, limitando sua visibilidade apenas ao arquivo. Em Rust, o sistema de módulos define a visibilidade de variáveis globais, exigindo o uso de palavras-chave como "pub" para torná-las acessíveis fora do módulo atual. Já em Go, as variáveis globais têm seu escopo limitado ao pacote em que são declaradas, sendo visíveis apenas dentro desse pacote.

Por fim, as linguagens Go e Rust fornecem maior segurança quanto ao gerenciamento de memória. Rust libera automaticamente recursos de memória que não são mais utilizados e evita vazamentos de memória, por exemplo. Enquanto que Go utiliza uma garbage collector para fazer esse gerenciamento, o que compromete o desempenho da linguagem em alguns aspectos. E a linguagem C que deixa o gerenciamento de memória sob responsabilidade do programador.

Comparação de desempenho

Comparação de desempenho (tempo):

	Tempo médio de execução em milisegundos (mesma matriz, e mesmo computador)
C	7,2998
Go	8,2899
Rust	8,6131

* Foi utilizada uma matriz de ordem 3.

Número de comandos:

	Número de comandos
C	42
Go	48
Rust	52

Número de linhas:

	Número de linhas
C	15
Go	16
Rust	17

Conclusão

Por ter sido feito apenas um teste, e com um input pequeno, é comprometedor fazer qualquer tipo de afirmação. Todavia percebe-se que C levou uma leve vantagem em todos os resultados, possui um menor número de linhas, com menos comandos e, por fim, é executado mais rapidamente. Ainda assim, não pode ser deixado de lado aspectos importantes de cada linguagem, como o suporte nativo para concorrência das linguagens Go e Rust, o qual não são usufruídos dado que o algoritmo usado é sequencial.