TRABALHO 2 - CONCEITOS DE LINGUAGENS DE PROGRAMAÇÃO

Gustavo Augusto Reginatto, Lucas Schneider Ludwig, Sandro Frizon Junior

COMPARAÇÃO ENTRE AS LINGUAGENS C, RUST E GOLANG

ELIMINAÇÃO DE GAUSS

1. **Tipos de Dados**

Característica	C	Rust	Golang
Inteiros	Int, long	I32, i64, u32, usize	Int, int64, unit
Ponto Flutuante	Float, double	F32, f64	Float32, float64
Booleanos	Int (0 ou 1)	bool	bool
Strings	Char* (ponteiro)	String, &str	string
Arrays	Int arr[N]	[i32; N]	[N]Type ou slice
Structs	struct	struct	struct
Ponteiros	Int*	&T, Box<T>	Unsafe.Pointer

2. **Acesso às Variáveis**

C: Utiliza variáveis globais e locais, sendo possível modificar diretamente valores através de ponteiros.

Rust: Utiliza let para imutáveis e let mut para mutáveis. Segurança garantida pelo sistema de propriedade (ownership).

Golang: Usa var para declaração explícita e := para inferência. Suporte a concorrência via goroutines e channels.

3. **Organização de Memória**

Aspecto	C	Rust	Golang
Gerenciamento	Manual (malloc/free)	Automático (ownership)	Coletor de lixo (GC)
Stack/Heap	Controle manual	Determinado pelo compilador	GC decide
Segurança	Risco de vazamento	Sem vazamento (borrow checker)	Possível vazamento

4. **Chamadas de Função**

C: Uso direto de ponteiros e passagem de parâmetros por valor/referência.

Rust: Permite fn com referências (&T, &mut T) garantindo segurança.

Golang: Possui passagem por valor por padrão, com suporte a ponteiros para modificações.

5. **Controle de Fluxo**

Estrutura	C	Rust	Golang
Condicional	If, switch	If, match	If, switch
Laços	For, while	For, loop, while	for
Manipulação	Goto, break	Break, continue	Break, continue

Rust se destaca pelo uso de match, enquanto Go e C utilizam switch.

6. Quantidade de linhas e comandos, alocação dinâmica e modularização

Linguagem	C	Rust	Golang
Linhas	212	64	93
Comandos principais	6	5	5
Aloc. dinâmica	Não	Sim	Não
Modularização	Média	Alta	Média

A extensão em linhas do código em C se dá devido à falta de bibliotecas modernas para manipulação de matrizes e vetores. O código em Rust foi o mais compacto entre os três e o de Golang foi o meio termo, porém, ainda bem menor em relação ao C. Portanto, Rust e Go foram mais compactos devido ao uso eficiente de estruturas de dados.

7. Desempenho em tempo de execução

Tam. Matriz	C	Rust	Golang
50x50	0,000358	0,000892	1,417
100x100	0,001336	0,002751	2,492
500x500	0,218363	0,845729	3,736
1000x1000	2,21573	3,154892	5,367
2000x2000	10,986	12,874212	15,171

Os testes foram realizados em uma máquina com as seguintes características:
CPU: Intel Core i5-7200U
Memória: 8 GB DDR4 2666 MHz
Sistema Operacional: Windows 10 (WSL para C)

Seus tempos de execução na tabela estão sendo apresentados em segundos.

8. Conclusão

A análise comparativa das linguagens C, Rust e Golang na implementação do algoritmo de eliminação de Gauss mostrou diferenças significativas em desempenho. O C continua sendo a opção mais eficiente, apresentando tempos de execução significativamente menores devido à sua proximidade com o hardware e ausência de abstrações de alto nível. Rust demonstrou um desempenho competitivo, ficando entre C e Golang, o que evidencia sua capacidade de otimização e segurança de memória sem grandes perdas de eficiência. Já Golang,

apesar de sua simplicidade e produtividade, obteve tempos de execução mais elevados, tornando-se menos ideal para cálculos numéricos intensivos. Dessa forma, a escolha da linguagem deve levar em consideração o equilíbrio entre desempenho, segurança e facilidade de desenvolvimento, dependendo do contexto da aplicação.