**Atividade: Comparação de Algoritmos de Ordenação**

**Objetivo:** Implementar e comparar o desempenho dos algoritmos **Bubble Sort**, **Selection Sort** e **Insertion Sort** em C, utilizando um vetor de números aleatórios. O objetivo é entender a diferença de eficiência entre os algoritmos e analisar como o tempo de execução varia com o tamanho do vetor.

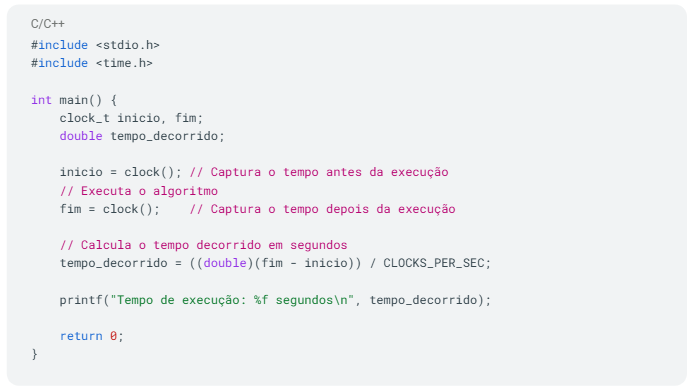
**Interpretação da atividade é parte da avaliação.**

**Descrição da Atividade:**

1. **Gerar um vetor de números aleatórios** com um tamanho definido pelo usuário use vetores de 10.000, 100.000 e 500.000. Realize testes com um vetor pequeno, imprimindo ele antes e depois de ordenar para garantir que os algoritmos estão corretos. Para números grandes não imprima o vetor, essa operação é demorada computacionalmente.

**Para que o teste seja justo, realize a ordenação do mesmo vetor usando os três algoritmos, então é necessário realizar cópias do vetor antes de ordenar.**

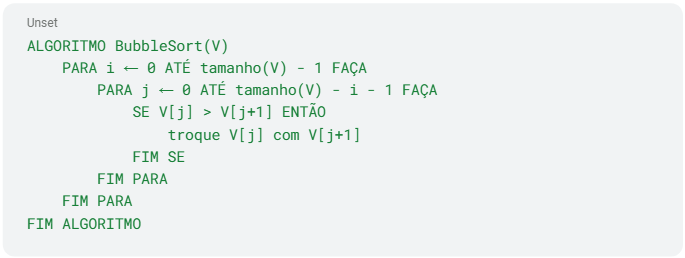
1. **Implementar o Bubble Sort** conforme o pseudocódigo fornecido.
2. **Implementar o Selection Sort** conforme o pseudocódigo fornecido.
3. **Implementar o Insertion Sort** conforme o pseudocódigo fornecido.
4. **Medir o tempo de execução** de cada algoritmo utilizando o clock da máquina, como mostra o exemplo abaixo:



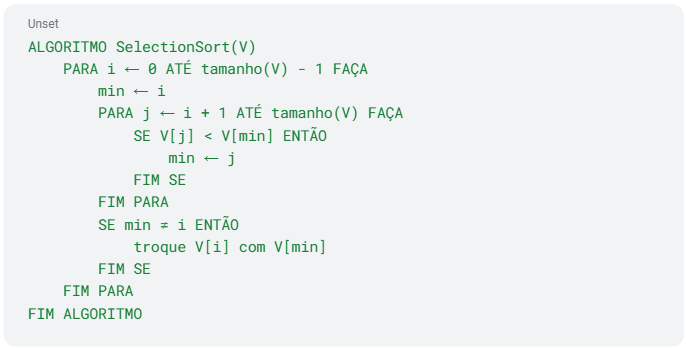
1. **Comparar os tempos obtidos** e responder às questões de análise.

**Pseudocódigos:**

**Bubble Sort:**

****

**Selection Sort:**

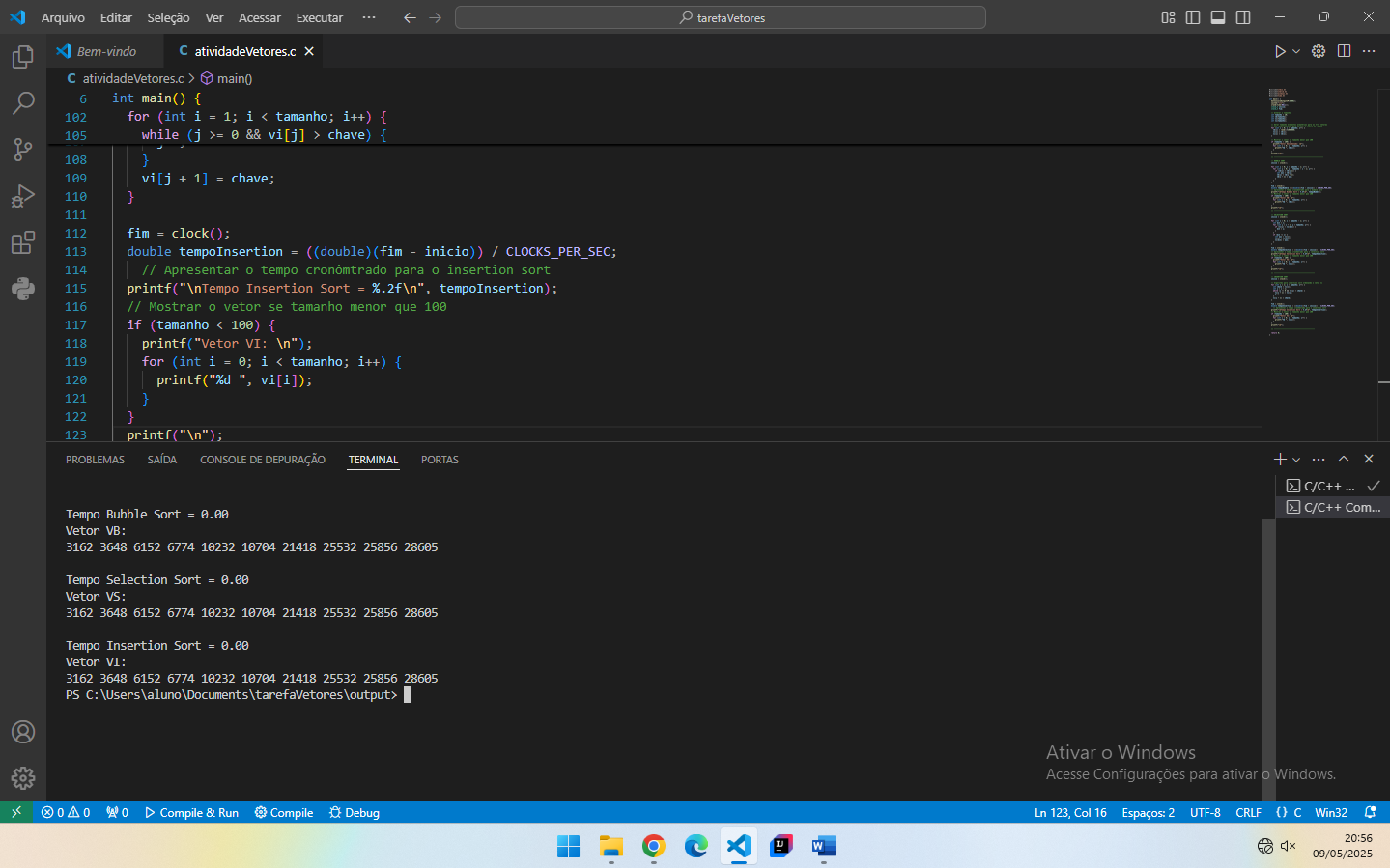
****

**Insertion Sort:**

****

**Critérios de Avaliação:**

* Implementação correta dos três algoritmos (apresentar prints de tela com vetores de 10 elementos ordenados).



* Uso adequado de vetores e manipulação de dados.
* Medição do tempo de execução e análise dos resultados.
* Respostas às questões de análise

**Questões para análise:**

1. Tempos de ordenação obtidos

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
| **Algoritmo** | **10.000 elementos** | **100.000 elementos** | **150.000 elementos** | **500.000 elementos** |
| Tempo de ordenação | Tempo de ordenação | Tempo de ordenação | Tempo de ordenação |
| Bubble Sort | 0.082000 | 15.125000 | 34.921000 | Não rodou |
| Selection Sort | 0.055000 | 3.554000 | 7.910000 | Não rodou |
| Insertion Sort | 0.032000 | 2.451000 | 5.377000 | Não rodou |

1. Qual dos três algoritmos apresentou melhor desempenho?

O Insertion Sort Apresento o melhor desempenho.

1. Realize uma análise dos códigos e aponte os possíveis motivos para o melhor desempenho de um algoritmo em comparação aos outros.

Bubble Sort percorre o vetor e compara o valor atual com o próximo, faz isso em todo o vetor, por isso é o mais lento em relação aos demais.

O Selection Sort armazena um valor mínimo, que começa com o primeiro valor da lista... Esse valor mínimo só troca se um valor menor for encontrado. Isso torna o algoritmo mais eficiente, pois esse mínimo serve como auxiliar que compara os elementos do vetor.

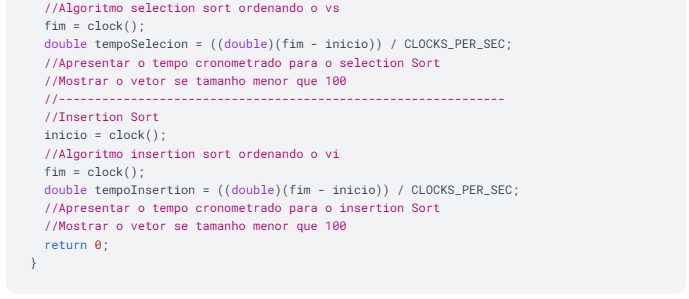
O Insertion Sort tem um auxiliar para fazer a verificação “chave”, que recebe a posição do valor, que é comparado com os demais de trás pra frente pelo while, o que talvez torne o código mais eficiente.

**Entrega (deve ser realizada no classroom):**

* Código-fonte C com a implementação dos algoritmos.
* Respostas às perguntas de análise.

**Estrutura básica do código que precisa ser implementado:**

****



Bom trabalho!