**SCC0220 - Laboratório Introdução à Ciência da Computação II**

**Relatório - Aula 4**

**Alunos NUSP**

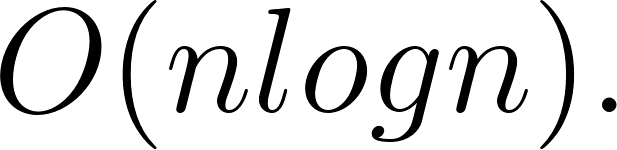
Pedro Henrique de Sousa Prestes 15507819

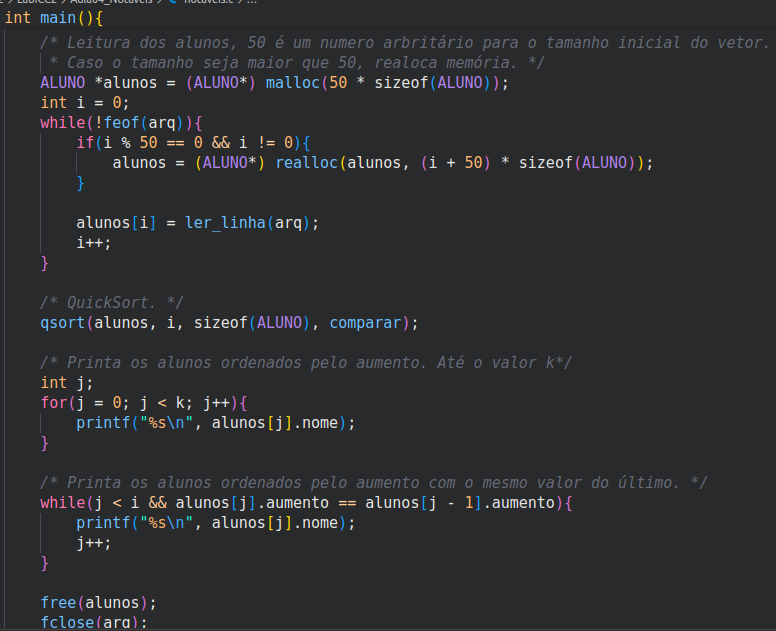
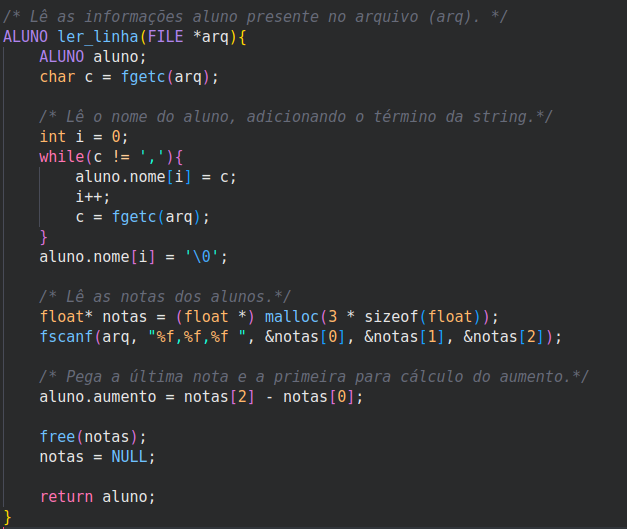
Pedro Lunkes Villela 15484287

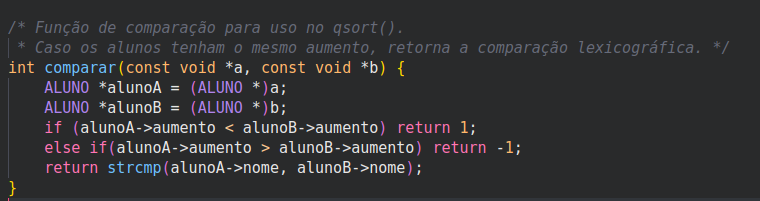
**Trabalho 4 - Notáveis**

**Ordenação / Arquivos**

**🡺 Comentário**

2. ❖O problema consiste em fornecer os K alunos com os maiores aumentos de nota baseado em uma lista de alunos em um arquivo .csv (planilha), considerando exceder esse limite para casos onde há alunos com o mesmo aumento do último aluno fornecido e considerando a lexicografia dos nomes dos alunos.
3. ❖ Para isso, é necessário armazenar os alunos dentro do programa e ordená-los. O algoritmo de ordenação escolhido para a resolução do exercício foi o QuickSort (qsort) implementado dentro da biblioteca <stdlib.h>, com complexidade média [](https://www.codecogs.com/eqnedit.php?latex=O(n%20log%20n).#0)

**🡺 Código**

****

**🡺 Saída**

Seguem alguns casos testados por nós comparando o desempenho de ambos algoritmos.

**1° Caso:**

Tempo de execução: 0.000091 segundos

Tempo de execução: 0.000103 segundos

Tempo de execução: 0.000086 segundos

**2° Caso:**

Tempo de execução: 0.117856 segundos

Tempo de execução: 0.079412 segundos

Tempo de execução: 0.080860 segundos

**3° Caso:**

Tempo de execução: 0.072552 segundos

Tempo de execução: 0.122333 segundos

Tempo de execução: 0.121983 segundos

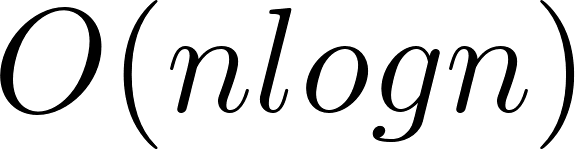
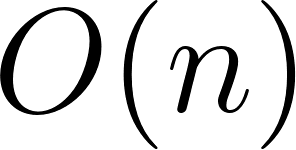
**4° Caso:**

Tempo de execução: 0.000213 segundos

Tempo de execução: 0.000218 segundos

Tempo de execução: 0.000226 segundos

**🡺 Conclusão**

A implementação feita é eficiente, pois sua complexidade média é [](https://www.codecogs.com/eqnedit.php?latex=O(n%20log%20n)#0) resultante do Quick Sort, já que as outras funções funcionam em [](https://www.codecogs.com/eqnedit.php?latex=O(n)#0). Além disso, a memória é utilizada de maneira adequada, lendo um aluno por vez e sempre liberando memória ao retornar os valores necessários.