O Gnome Sort é um algoritmo de ordenação que busca ordenar elementos que estejam desordenados, ele percorre um vetor seja de strings ou de números para comparar, se o elemento em análise for maior que o próximo elemento, troca a posição, caso contrário, permanece no lugar.

O programa feito serve para ordenar um arquivo que recebe o vetor de struct Alunos com o algoritmo de ordenação Gnome Sort, ele ordena alfabeticamente os nomes dos alunos matriculados. Essa função compara os elementos strings do vetor, se o elemento estiver em sua posição correta (ser menor que o elemento do próximo índice ou igual a 0) o Gnome Sort passa para o próximo índice do vetor, se não ele troca com o elemento do índice anterior, faz a analise nesse mesmo índice e repete esse processo até ordenar corretamente.

Essa função é um algoritmo fácil de implementar, ele só precisa de um laço de repetição While, é necessário para ser executado até que a posição atual chegue ao final do vetor, e algumas operações para fazer a troca de elementos, como o If, para comparar se o elemento na sua posição é maior ou igual ao elemento na posição anterior. O Gnome Sort pode ser útil se tiver poucos elementos no vetor para ordenar, facilita quando alguns elementos já estão em sua posição correta. Porém, se tiver muitos elementos para ordenar, o tempo de execução pode ser mais lento.

O Gnome Sort tem uma complexidade $O(n^2)$, n é o número de elementos que vão ser ordenados, por isso que o tempo de execução pode ser um pouco lento, dependendo da ordenação a ser feita. No pior caso cada elemento precisa ser comparado com todos os elementos anteriores para alcançar sua posição correta, que seria n*(n-1). Já no melhor caso ele tem uma complexidade O(n), que é quando o vetor já está ordenado e não precisa fazer troca de posições.

Além de aprimorar nossos conhecimentos, aprendemos sobre o funcionamento da organização de elementos no Gnome Sort e suas diversas formas de implementação, também aprendemos a manipular arquivos usando funções de bibliotecas diversas e atualizar um arquivo sem precisar apagar os dados dentro dele, conhecemos a função ‘strcmp’ que compara duas strings devolvendo um valor inteiro, e a função 'strcpy' que serve para copiar a informação de uma string para a outra, onde a string de destino e a string de cópia são parâmetros da função; e usar size_t corretamente, é um tipo para variável que armazena o tamanho do tipo do objeto em bytes, e não pode ser lida como inteiro (%i ou %d), ele é lido como unsigned long int (%lu), além disso aprendemos a fazer o tempo de execução, onde necessitou da inserção da biblioteca “time.h” que não conhecíamos, e precisa da função ‘clock_t’, que é uma função que retorna o tempo aproximado do processador consumido pelo programa.

Link do repositório do github:

https://github.com/classroom-ufersa/GnomeSort/tree/main