**Trabalho 1: Ordenação externa**

*Pesquisa e ordenação de dados*

**João Vitor Forgearini Beltrame | Talles Siqueira Ceolin**

jvbeltrame@inf.ufsm.br | tsceolin@inf.ufsm.br

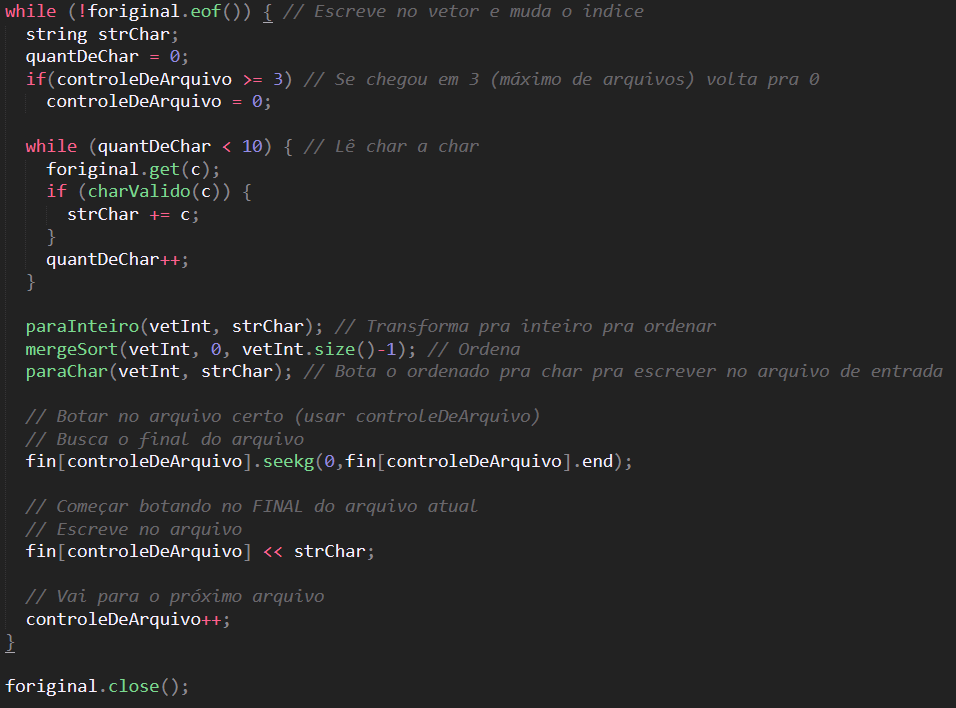
1. **Introdução**

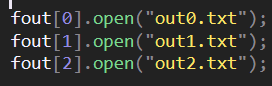
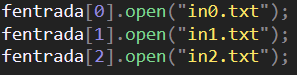
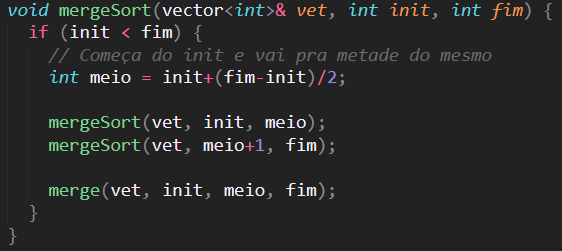
O trabalho consiste na implementação de um algoritmo envolvendo um método de ordenação externa baseado no Merge sort, onde a entrada será por um arquivo de texto e este possui a restrição de não caber na memória.

1. **Algoritmo**

A linguagem escolhida foi C++ pela sua maneira de trabalhar com arquivos e estruturas de dado como o *Vector* ser mais acessível.

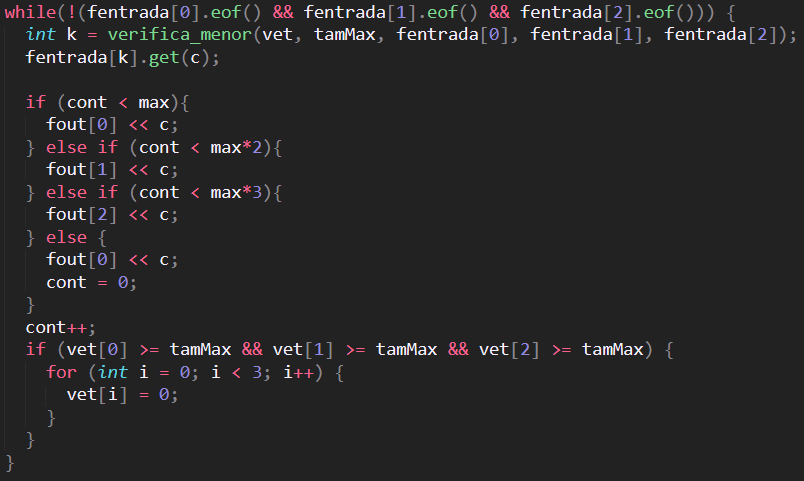
O algoritmo inicia pela abertura de quatro arquivos de texto, o texto original e os arquivos “in0.txt”, “in1.txt” e “in2.txt”, logo após é feito a leitura de caractere por caractere do arquivo original, considerando a restrição de no máximo 10 bytes.

A leitura de 10 caracteres é feita e em seguida estes são ordenados a partir do Merge Sort, logo após eles são escritos e um dos três arquivos “in.txt”, alternando entre eles pela variável “controleDeArquivo”.

O Merge sort foi adaptado para a linguagem C++, usando então uma estrutura *Vector* no lugar de um simples vetor.

Após, 3 arquivos de entrada e 3 de saída são abertos, os arquivos “in” possuem blocos de 10 caracteres já ordenados previamente, já os arquivos de saída (“out”) são os arquivos em que os caracteres serão inseridos enquanto são ordenados. O limite do bloco do arquivo de saída (variável “*max*”) é, inicialmente, definido por *(quantidade de caracteres \* quantidade de arquivos)* e, depois, dentro do laço de repetição, é multiplicado pelo tamanho inicial do primeiro bloco (*max\*=10*).

A ordenação é feita dentro de dois laços de repetição. Primeiramente os arquivos são abertos e os arquivos de saída são apagados e recriados, para ser possível manipular arquivos vazios, sem sobrescrever dados. A ordenação em si acontece dentro do segundo laço: Foi usado um vetor com 3 posições (relativas aos 3 arquivos de entrada) que controlam de qual bloco está vindo o caractere que será comparado, ou seja, a ordenação só passa para o próximo bloco quando todo o primeiro for ordenado no arquivo de saída. Quando todo o bloco for lido, o vetor é zerado e reutilizado no próximo bloco. Foi usado, também, um contador, que define em qual arquivo de saída os dados serão escritos, pois quando o contador ultrapassar o máximo que um bloco pode ter, os dados serão escritos no próximo arquivo ou no próximo bloco do próximo arquivo.

Há uma verificação de qual é o menor caractere válido e dentro do seu bloco entre os 3 arquivos de entrada e é colocado no arquivo de saída adequado (o contador citado anteriormente decidirá qual dos 3 arquivos de saída será usado).

Saindo do laço secundário, o programa retorna para o primário, que irá ajustar os ponteiros para seus respectivos arquivos, copiar os arquivos de saída para os de entrada, apagar os dados dos arquivos de saída e, se necessário, também repetirá o laço secundário. O laço principal repetirá até os dois últimos arquivos de saída estarem vazios, pois quando estiverem vazios, todos os dados estarão ordenados no primeiro arquivo de saída, que, como dito anteriormente, será copiado para o arquivo de entrada no final do laço.

Após os dois laços em que acontece a ordenação, a função “*resultado*” é chamada, copiando os dados do arquivo de entrada para um arquivo final (“*final.txt*”), que será criado pelo próprio programa na mesma pasta em que ele está sendo executado.

Todos os arquivos abertos serão fechados e a ordenação está devidamente concluída.

1. **Aplicações**

Contagem de caracteres específicos em um arquivo e ordenações byte a byte.

**Referências**

“Intercalação balanceada” by Marcos Kutova, <http://www.showme.com/sh/?h=0a0OGGW>;

“Merge Sort” by GeeksforGeeks <http://youtu.be/JSceec-wEyw> ;