

## Implemente, usando arquivos para simular as fitas de entrada e saída, o método de ordenação externa Intercalação Balanceada de Vários Caminhos:

Este código simula um método de ordenação externa usando intercalação balanceada de vários caminhos para ordenar blocos de caracteres de entrada e gravar os resultados em arquivos de saída separados. Cada bloco é dividido em três sub-blocos, e o Bubble Sort é usado para ordenar cada sub-bloco.

```
//MÉTODO DE ORDENAÇÃO BUBBLE SORT
void bubbleSort(vector<char> &bloco) {
    int n = bloco.size();
    for (int i = 0; i < n - 1; i++) {
        for (int j = 0; j < n - i - 1; j++) {
            if (bloco[j] > bloco[j + 1]) {
                //TROCA OS ELEMENTOS SE ESTIVEREM FORA DE ORDEM
                char temp = bloco[j];
                bloco[j] = bloco[j + 1];
                bloco[j + 1] = temp;
            }
        }
    }
}
```

Esta função implementa o algoritmo **Bubble Sort** para ordenar um vetor de caracteres. Compara elementos adjacentes e os troca se estiverem fora de ordem. Ele repete esse processo até que todo o vetor esteja ordenado.

```
//DIVIDI E ORDENA OS BLOCOS DE CARACTERES
void dividirOrdenarBlocos(const string &arquivoEntrada, const string &arquivoSaida, const string &arquivoSaida01, const string &arquivoSaida02) {
    //ABERTURA DOS ARQUIVOS DE ENTRADA E SAIDA
    ifstream arquivo(arquivoEntrada);
    ofstream saida(arquivoSaida);
    ofstream saida01(arquivoSaida01);
    ofstream saida02(arquivoSaida02);

    string palavra;

    //LOOP QUE LER CADA LINHA DO ARQUIVO DE ENTRADA (PALAVRA POR PALAVRA)
    while (getline(arquivo, palavra)) {
        vector<char> Fita, Fita01, Fita02; //ARMAZENA OS BLOCOS DE CARACTERES

        //PREENCHE OS VETORES FITAS COM OS PRIMEIROS 3 CARACTERES, 3 SEQUITES E RESPECTIVAMENTE
        for (int i = 0; i < 3 && i < palavra.length(); i++) {
            Fita.push_back(palavra[i]);
        }

        for (int i = 3; i < 6 && i < palavra.length(); i++) {
            Fita01.push_back(palavra[i]);
        }

        for (int i = 6; i < 9 && i < palavra.length(); i++) {
            Fita02.push_back(palavra[i]);
        }
    }
}
```

```

//ORDENA OS BLOCOS DE CARACTERES
bubbleSort(Fita);
bubbleSort(Fita01);
bubbleSort(Fita02);

//ESCREVE OS RESULTADOS NOS ARQUIVOS DE SAIDA
for (char c : Fita) {
    saida << c; //ESCREVE OS CARACTERES DA FITA01 NO ARQUIVO DE SAIDA01
}
saida01 << endl; //PULA PRA PRÓXIMA LINHA NO ARQUIVO DE SAIDA01

for (char c : Fita01) {
    saida01 << c; //ESCREVE OS CARACTERES DA FITA01 NO ARQUIVO DE SAIDA02
}
saida02 << endl; //PULA PRA PRÓXIMA LINHA NO ARQUIVO DE SAIDA02

for (char c : Fita02) {
    saida02 << c; //ESCREVE OS CARACTERES DA FITA01 NO ARQUIVO DE SAIDA02 (NOVAMENTE)
}
saida02 << endl; //PULA PRA PRÓXIMA LINHA NO ARQUIVO DE SAIDA02
}

//FECHAMENTO DOS ARQUIVOS
arquivo.close();
saida.close();
saida01.close();
saida02.close();

```

Essa função realiza a ordenação externa dos blocos de caracteres de acordo com o método da intercalação balanceada de vários caminhos. Ela lê linhas do arquivo de entrada, divide as palavras em três blocos de caracteres, ordena cada bloco usando o Bubble Sort e, em seguida, escreve os resultados nos arquivos de saída.

```

int main() {
    const string arquivoEntrada = "Entrada.txt";
    const string arquivoSaida = "Fita.txt";
    const string arquivoSaida01 = "Fita01.txt";
    const string arquivoSaida02 = "Fita02.txt";

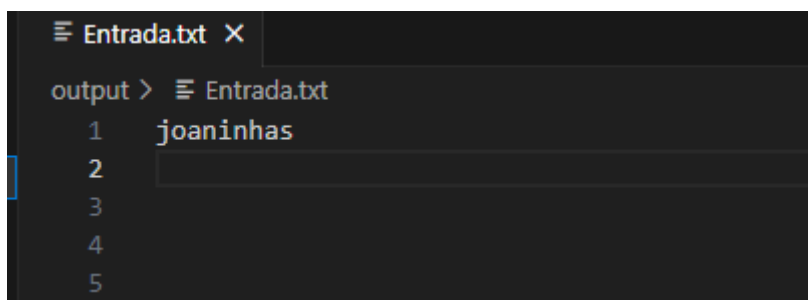
    //CHAMA A FUNÇÃO PARA DIVIDIR E ORDENAR OS BLOCOS
    dividirOrdenarBlocos(arquivoEntrada, arquivoSaida, arquivoSaida01, arquivoSaida02);

    return 0;
}

```

Aqui, a função main é definida. Ela configura os nomes dos arquivos de entrada e saída, em seguida, chama a função dividirOrdenarBlocos para realizar o processo de intercalação balanceada de vários caminhos nos blocos de caracteres. O programa finaliza retornando 0, indicando sucesso.

### Entrada.txt



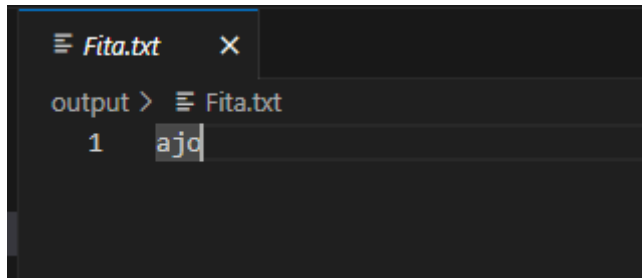
```

Entrada.txt X
output > Entrada.txt
1 joaninhas
2
3
4
5

```

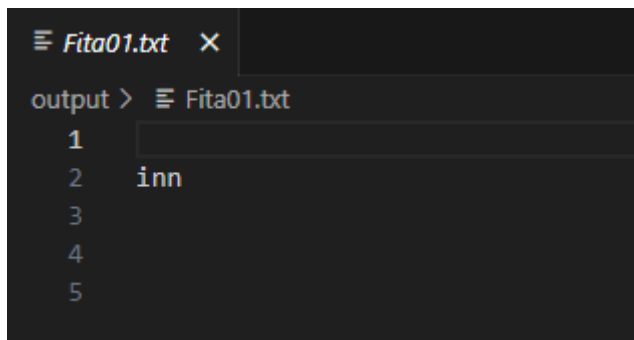
## RESULTADOS:

### Fita.txt



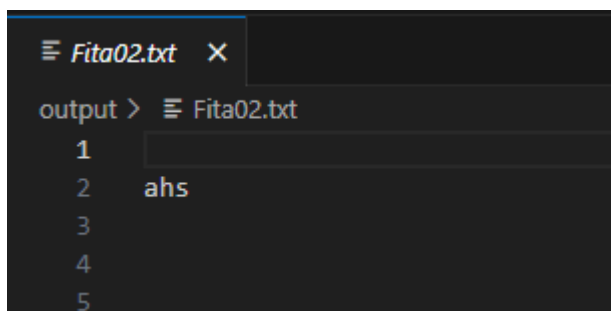
```
≡ Fita.txt X
output > ≡ Fita.txt
1 ajd
```

### Fita01.txt



```
≡ Fita01.txt X
output > ≡ Fita01.txt
1
2 inn
3
4
5
```

### Fita02.txt



```
≡ Fita02.txt X
output > ≡ Fita02.txt
1
2 ahs
3
4
5
```

**COMENTÁRIO:** como visto acima infelizmente não consegui cumprir totalmente com o que o professor propôs, tive muita dificuldade em executar essa atividade, achei ela bem complexa. Tive que fazer muitas pesquisas, pedir ajuda ao professor e aos colegas, mas o máximo em que cheguei foi na primeira parte do código onde divide a palavra em 3 blocos, não conseguir passar disso, mas continuarei a procurar uma solução de meu entendimento para conseguir executar todo o código com êxito.