

## Trabalho 1 – AEDS III

Pedro Henrique Santos

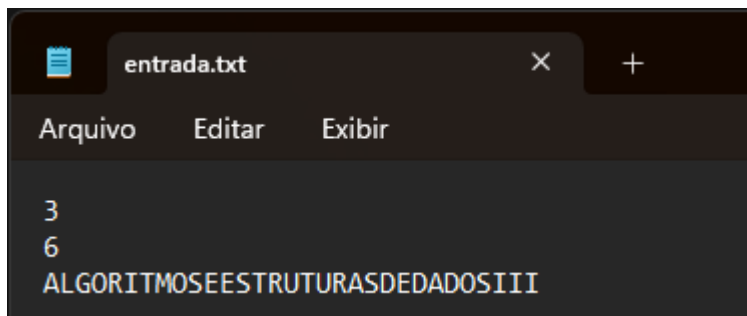
### 1. Proposta

Implemente, usando arquivos para simular as fitas de entrada e saída, o método de ordenação externa Intercalação Balanceada de Vários Caminhos. Considere como entrada um arquivo conforme ilustrado a seguir (podendo variar os valores) com um primeiro número indicando o tamanho da memória interna, um segundo número indicando a quantidade de fitas disponíveis e o conjunto de dados na sequência.

### 2. Entrada

O arquivo *entrada.txt* deve conter:

1. Quantidade de fitas
2. Tamanho da memória
3. Sequência de letras para ordenar



### 3. Código

A função *ordenacaoExterna* lê essas três informações do arquivo *entrada.txt*. Em seguida, chama funções para criar as fitas, montar blocos iniciais e intercalar os dados:

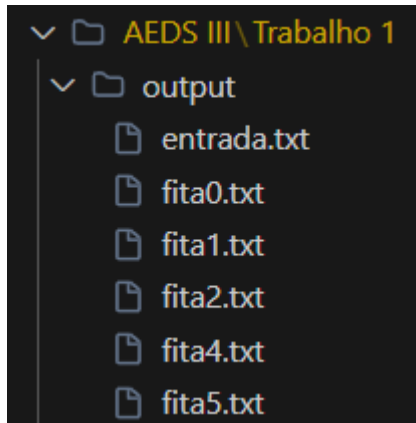
```
void ordenacaoExterna() {  
    ifstream entrada("entrada.txt");  
    if (!entrada) {  
        cerr << "Erro ao abrir o arquivo de entrada." << endl;  
        return;  
    }  
  
    // Ler quantidade de fitas, tamanho da memória e palavra a ser  
    ordenada  
    entrada >> tamanhoMemoria >> quantidadeFitas;  
    string dados;
```

```

    entrada.ignore(); // Ignorar o caractere de nova linha
    getline(entrada, dados);
    entrada.close();

    lendoMetadeCima = false;
    criarFitas();
    montarBlocosIniciais(dados);
    intercalacar();
}

```



A função *montarBlocosIniciais* divide a sequência de letras em blocos do tamanho especificado e os ordena. Os blocos ordenados são gravados alternadamente em fitas de saída para a etapa de intercalação.

```

void montarBlocosIniciais(const string& dados) {
    char memoria[tamanhoMemoria];
    int auxCont = 0;
    abrirFitas();

    for(size_t i = 0; i < dados.size(); i += tamanhoMemoria) {
        size_t tamanhoVetor = min(tamanhoMemoria,
static_cast<int>(dados.size() - i));
        strncpy(memoria, dados.c_str() + i, tamanhoVetor);

        qsort(memoria, tamanhoVetor, sizeof(char), compare);

        if(auxCont >= quantidadeFitas / 2) {
            auxCont = 0;
        }

        fitasGravacao[auxCont] << string(memoria, tamanhoVetor) <<
separador;
        auxCont++;
    }
    lendoMetadeCima = !lendoMetadeCima;
    fecharFitas();
}

```

A função *intercalar* foi a mais complicada, pois realiza a intercalação dos blocos das fitas até que todos os dados estejam em uma única fita. Os dados são lidos das fitas e intercalados de forma ordenada, salvando o resultado em outras fitas.

```
void intercalacar() {
    pair<char, int> memoria[tamanhoMemoria];

    bool fitaAberta[quantidadeFitas / 2];
    bool fitaVazia[quantidadeFitas / 2];
    bool intercalando = true;
    bool passosRestando = true;
    bool intercalacoesRestantes = true;
    bool foiGravada[quantidadeFitas / 2];

    int fitasVazias = 0;
    int contMemoria = 0;
    int contGravacao = 0;
    int contFitas = 0;
    int fitasDisponiveis = quantidadeFitas / 2;
    int fitasGravadas = quantidadeFitas / 2;

    int fitaFinal;

    char charAux = '\\0';

    while(intercalacoesRestantes) {

        abrirFitas();

        contGravacao = 0;
        fitasVazias = 0;
        intercalando = true;

        for(int i = 0; i < quantidadeFitas / 2; i++) {
            if(fitasLeitura[i].peek() == EOF) {
                fitaVazia[i] = true;
                fitasVazias++;
            } else {
                fitaVazia[i] = false;
            }
            foiGravada[i] = false;
        }

        while(intercalando) {
            for(int i = 0; i < quantidadeFitas / 2; i++) {
                fitaAberta[i] = true;
            }
        }
    }
}
```

```

        fitasDisponiveis = (quantidadeFitas / 2) - fitasVazias;
        contFitas = 0;
        contMemoria = 0;
        charAux = '\\0';
        passosRestando = true;

        while(passosRestando) {
            if(fitaAberta[contFitas] && !fitaVazia[contFitas]) {

                fitasLeitura[contFitas].get(charAux);

                if(charAux == separador) {
                    if(fitasLeitura[contFitas].peek() == EOF) {
                        fitaVazia[contFitas] =
true;

                        fitasVazias++;
                    }
                    fitaAberta[contFitas] = false;
                    fitasDisponiveis--;
                } else {
                    memoria[contMemoria].first =
charAux;

                    memoria[contMemoria].second = contFitas;
                    contMemoria++;
                }
            }

            contFitas++;
            if(contFitas == quantidadeFitas / 2) {
                contFitas = 0;
            }

            if(contMemoria == tamanhoMemoria || fitasDisponiveis ==
0) {

                if(contMemoria != 1) {
                    qsort(memoria, contMemoria, sizeof(pair<char,
int>), comparePair);
                }
                pair<char, int> aux = memoria[0];
                memoria[0] = memoria[contMemoria-1];
                memoria[contMemoria-1] = aux;

                fitasGravacao[contGravacao] << aux.first;
                contFitas = aux.second;
                contMemoria--;
            }

            if((fitasDisponiveis == 0 && contMemoria == 0)) {
                passosRestando = false;
            }
        }
    }
}

```

```

    }
}

fitasGravacao[contGravacao] << separador;
foiGravada[contGravacao] = true;
contGravacao++;

if(contGravacao == quantidadeFitas / 2) {
    contGravacao = 0;
}
if(fitasVazias == quantidadeFitas / 2) {
    intercalando = false;
}
}

fitasGravadas = 0;
for(int i = 0; i < quantidadeFitas / 2; i++) {
    if(foiGravada[i]) {
        fitasGravadas++;
        fitaFinal = i;
    }
}
fecharFitas();
if(fitasGravadas == 1) {
    intercalacoesRestantes = false;
    gerarSaida(fitaFinal);
}
lendoMetadeCima = !lendoMetadeCima;
}
}

```

Por fim, *gerarSaida* renomeia a fita final para *Saida.txt*, onde o usuário pode encontrar os dados ordenados.

```

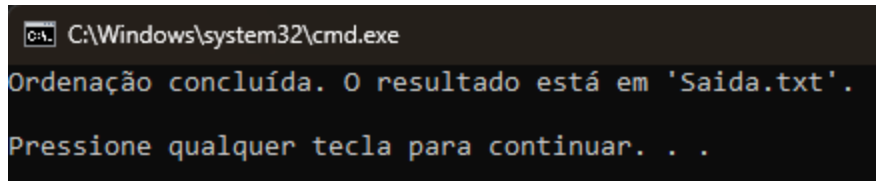
void gerarSaida(int fitaFinal) {
    if(lendoMetadeCima) {
        fitaFinal += quantidadeFitas / 2;
    }
    string aux = "fita" + to_string(fitaFinal) + ".txt";
    const char* saida = aux.c_str();

    remove("Saida.txt");
    rename(saida, "Saida.txt");
}

```

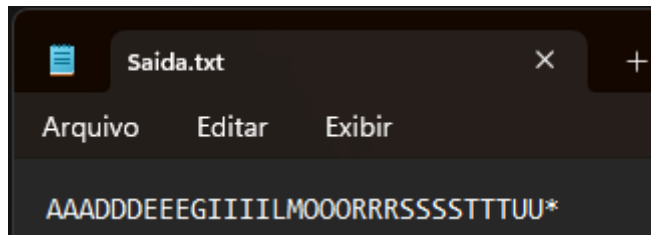
#### 4. Saída

Após a execução do programa será exibida a mensagem de conclusão.



```
C:\Windows\system32\cmd.exe
Ordenação concluída. O resultado está em 'Saida.txt'.
Pressione qualquer tecla para continuar. . .
```

O arquivo *Saida.txt* contém a sequência ordenada das letras tirada da uma fita utilizada.



```
Saida.txt
Arquivo  Editar  Exibir
AAADDDEEEGIIIIILM000RRRSSSTTTUU*
```