Trabalho 1: Frequência de Caracteres (SSC0903-CAD - 2025/1)
Linguagem: C com OMP

Categoria: Estruturas e bibliotecas **Modelo de Programação: OpenMP**

Autores: Paulo Sérgio Lopes de Souza e Thiago de Jesus Oliveira Durães

Texto extraído do trabalho de Shahriar Manzoor, em URI Online Judge. Disponível em: https://www.urionlinejudge.com.br/judge/pt/problems/view/1251

Faça um programa em **C** utilizando **OpenMP** que, dado um texto de entrada, retorne as frequências de cada um dos caracteres presentes em cada linha do texto. Considere que as linhas fornecidas conterão apenas caracteres com códigos ASCII entre 32 e 126, inclusive. Não consideraremos os caracteres de fim de linha e DEL.

Considere como entrada um arquivo texto contendo os casos de teste. As linhas são separadas por uma quebra de linha simples. O texto deve ser lido por meio do redirecionamento de fluxo de entrada < (***stdin***), ou seja, não use ponteiros para arquivos no código.

Para executar no bash, por exemplo, utilize este padrão:

```
diga_freq < entrada.txt
```

Obs: na linha de comando acima, considera-se que o programa foi inserido em ***diga_freq.c*** e o executável ***diga_freq*** e está no diretório atual. A saída deve ser impressa utilizando a saída padrão (***stdout***).

Entrada do Algoritmo

A entrada contém vários casos de teste. Cada caso de teste é composto por uma única linha de texto com até 1000 caracteres. Segue um exemplo de entrada:

```
AAABBC11
122333A
```

Saída do Algoritmo

Imprima o valor ASCII de todos os caracteres presentes nas *strings* e as suas frequências de acordo com o formato abaixo. A ordem de impressão das strings (de cada linha) deve ser a mesma do arquivo de entrada. Uma linha em branco deverá separar 2 conjuntos de saída (i.e., o resultado de duas linhas/*strings*). Imprima os caracteres ASCII de uma *string* em ordem ascendente de frequência. Se dois caracteres de uma mesma *string* estiverem presentes com a mesma quantidade de frequência, imprima primeiro o caractere que tem valor ASCII menor. A saída é terminada por final de arquivo (EOF). Segue um exemplo de saída, considerando o exemplo de entrada de antes:

67 1
49 2
66 2
65 3

49 1
65 1
50 2
51 3

Soluções sequenciais:

Algumas soluções sequenciais estão disponíveis na internet, como as soluções encontradas em:
https://github.com/malbolgee/URI/blob/master/1251.c
http://muitomaiscodigoss.blogspot.com/2018/04/uri-problema-1251-diga-me-frequencia.html
https://www.life2coding.com/uri-online-judge-solution-1251-tell-me-the-frequencies-intermediate-problem/
https://urisolve.blogspot.com/2017/04/uri-solution-1251-tell-me-frequencies.html

O que deve ser entregue neste trabalho?

Um arquivo zip (não use outro padrão), contendo o código fonte (um único arquivo .c).

Não precisa entregar relatório do código desenvolvido.

Colocar no arquivo com o código fonte os **nomes** dos alunos, **em ordem alfabética**, que realmente participaram do trabalho.

Como devo submeter este trabalho?

No link disponível no e-disciplinas. Apenas uma submissão por grupo.

Como será a avaliação?

Serão considerados itens como: ter seguido à especificação dada, qualidade do código fonte desenvolvido, comentários, indentação, nomes significativos a identificadores e tempo de resposta do algoritmo. Lista não exaustiva.

Em termos de linguagem de programação, o que devo usar?

A linguagem C (não use C++ ou outra linguagem) e OpenMP.

Em relação ao OpenMP use da melhor forma possível todos os recursos que ele (o OpenMP) oferece. Alguns exemplos são: parallel, nested, reduction, for, task, simd, e sincronizações necessárias (lista não exaustiva).