UNIVERSIDADE FEDERAL DE SANTA CATARINA
CENTRO DE CIÊNCIAS, TECNOLOGIAS E SAÚDE - CTS
DEPARTAMENTO DE COMPUTAÇÃO – DEC

DISCIPLINA: LINGUAGENS DE PROGRAMAÇÃO 2
PROFESSOR ANTONIO CARLOS SOBIERANSKI
a.sobieranski@ufsc.br

ENUNCIADO TRABALHO T1 – "FILE MANIPULATOR HACK"

A ser desenvolvido individualmente ou em duplas
Elementos utilizados: C++, entrada e saída em tela, vetores e strings, leitura e gravação de arquivos. Os arquivos possuem palavras (quantidade variada), sem espaços, separadas por quebra de linha, e podem ter a mesma palavra no mesmo arquivo e em arquivos distintos.

O nosso primeiro programa entregável corresponderá a um gerenciador de arquivos em disco que contém o seguinte menu abaixo:

```
--------------------------------------------------
FILE MANIPULATOR HACK
Select an option:

1. Open a File...
2. Search for Substrings...
3. Remove Words Containing a Substring
4. Remove all Repeated words
5. Show Statistics
6. Exit
--------------------------------------------------
Option:
```

1. A opção 1 abre um arquivo. Deve imprimir em uma nova linha o texto abaixo (cout), e em seguida abrir o arquivo, carregando-o em memória. Vários arquivos podem ser carregados em sequência na mesma execução do programa.
cout << "Enter a file to open:"

2. A opção 2 localiza a ocorrência de determinada substring em todos os arquivos que foram carregados na opção 1, conforme abaixo, e imprime em tela as ocorrências:
cout << "Enter a substring to search:"

O *output* em tela deve apresentar de onde veio determinada substring (o nome do arquivo) e a palavra completa. Exemplo de busca da substring "izz"

File1.txt has word: pizza
File1.txt has word: frizzy
File1.txt has word: pizzicato
File2.txt has word: blizzard
File3.txt has word: frizzy
File3.txt has word: frizz
File3.txt has word: frizzy

3. A opção 3 remove todas as substrings contendo determinada palavra em todos os arquivos que foram carregados, conforme *output* abaixo.
cout << “Enter a substring to remove all occurrences:”

Informar somente após isso se foram ou não removidas palavras (com *bool*), texto a sua escolha.

4. Eliminar a ocorrência de todas as palavras **REPETIDAS** que possam por ventura ocorrer no mesmo arquivo, e em outros arquivos. Somente a primeira que surgir em ordem de arquivo e *index* do vetor deve ser mantida, e todas as suas re’plicas eliminadas.

5. Apresenta estatísticas ATUAIS por arquivo em tela. Significa dizer que algum mecanismo de controle deve ser implementado internamente de modo a manter os dicionários separados em memória durante a execução do programa, e armazenar o seu respectivo nome original.
Exemplo:

Statistics:
File1.txt has 350000 words
File2.txt has 1500 words
File3.txt has 600 words

6. Sair do sistema e gravar um único arquivo contendo todas as palavras atuais (após as manipulaçoes do menu), chamado “**concatenation.txt**”, por ordem de arquivo inserido e depois pelo *index* do vetor.

Requisitos:

- Usar *std::vector<string>* para armazenar os arquivos, e *ifstream* e *ofstream* para acesso aos arquivos;
- compilar com GCC padrão (GNU-gcc, mingw). Vide *VirtualMachines* no Moodle, ou instalação própria. **Vou compilar com “g++ *.cpp -o exec”.**
- Não compilou e não executou, sem nota.
- Nota proporcional aos itens implementados e corretude.
- Entregar somente *source-code* em um arquivo compactado em formato "zip"

Superdica: *Opcionalmente*, o armazenamento de arquivos de palavras pode ser realizado com vetores multidimensionais, conforme slides no Moodle.

std::vector< std::vector < std::string > > FilesAndWords;
vector< vector < string > > FilesAndWords;