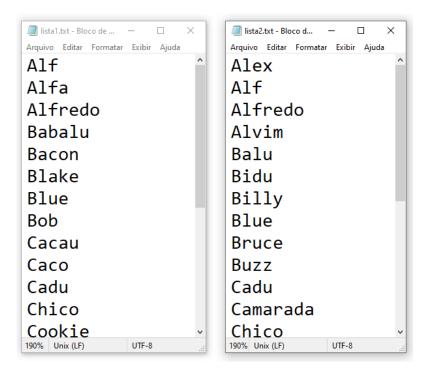
## <u>Atividade Prática</u>: Merge e K-way Merge

Organização e Recuperação de Dados Profa. Valéria

UEM - CTC - DIN

## Exercício 1

- O Exercício 1 um consiste na implementação do merge (intercalação) de duas listas ordenadas de nomes
  - Vamos supor que as listas estão armazenadas nos arquivos de texto *lista1.txt* e *lista2.txt* (disponíveis no *Classroom*)
  - Nos dois arquivos, cada nome ocupa exatamente uma linha



Estrutura geral do programa

# *Merge* de duas listas ordenadas

Lembre-se que as funções inicialize e leia nome retornarão tuplas de valores, logo você deve usar uma tupla de valores para receber o retorno dessas funções

```
import io
# CONSTANTES
VALOR BAIXO = ''
VALOR ALTO = '~'
def inicialize() -> tuple[str, str, io.TextIOWrapper, io.TextIOWrapper,
                                                      io.TextIOWrapper, bool]:
    return ant1, ant2, lista1, lista2, saida, existem mais nomes
def leia nome (lista: io. TextIOWrapper, nome ant: str, nome outra lista: str,
                         existem mais nomes: bool) -> tuple[str, str, bool]:
    return nome, nome ant, existem mais nomes
def merge() -> None:
if name == ' main ':
   merge()
```

#### Pseudocódigo inicialize

## Função de inicialização

### Função de leitura para o merge de duas listas

Para gerar um erro, use o comando *raise* 

#### Pseudocódigo leia nome

#### Pseudocódigo merge

### Função merge para duas listas

```
def merge() -> None:
    chame inicialize() para inicializar anterior1, anterior2,
                                 listal, lista2, saida, existem mais nomes
    chame leia nome para ler nomel da listal
    chame leia nome para ler nome2 da lista2
    enquanto existem mais nomes faça
        se nome1 < nome2 então # chave da LISTA1 é menor
             escreva nomel em saida
             chame leia nome() para ler nome1 da lista1
        senão se nomel > nome2 então # chave da LISTA2 é menor
             escreva nome2 em saida
            chame leia nome() para ler nome2 da lista2
        senão # as chaves são iquais
        escreva nomel em saida
             chame leia nome() para ler nome1 da lista1
             chame leia nome() para ler nome2 da lista2
    feche todos os arquivos
```

## Exercício 2

- O Exercício 2 consiste na implementação do *k-way merge* (*merge* de k listas), supondo que:
  - As listas a serem intercaladas estão ordenadas
  - Não há repetição de nomes nas listas
- Descompacte o arquivo listas.zip (disponível no Classroom) e use os arquivos (lista0.txt, lista1.txt, ..., lista99.txt) para testar a sua implementação
- Faça com que o valor de k (i.e., o número de listas a serem intercaladas) seja passado como um argumento pela linha da comando
  - Exemplo:
    - \$ python3 kwaymerge.py 100

```
import sys
import io
# CONSTANTES
VALOR BAIXO = ''
VALOR ALTO = '~'
# VAR GLOBAL
numEOF = 0
def inicialize(numListas: int) -> tuple[list[str], list[str], list[io.TextIOWrapper],
                                                                  io.TextIOWrapper, bool]:
    return anteriores, nomes, listas, saida, existem mais nomes
def finalize(listas: list[io.TextIOWrapper], saida: io.TextIOWrapper, numListas: int) -> None:
    . . .
def leia nome(lista: io.TextIOWrapper, nome ant: str, existem mais nomes: bool, numListas: int) ->
                                                                             tuple[str, str, bool]:
    return nome, nome ant, existem mais nomes
def kwaymerge(numListas: int) -> None:
    . . .
def main() -> None:
if name == ' main ':
    main()
```

#### Pseudocódigo inicialize

## Função de inicialização para o kwaymerge

## Função de finalização

#### Pseudocódigo finalize

#### Pseudocódigo leia nome

## Função de leitura para o kwaymerge

#### Pseudocódigo kwaymerge

### Função kwaymerge

```
def kwaymerge(numListas: int) -> None:
    # inicialize as variáveis
    chame inicialize() para inicializar anteriores, nomes, listas, saída,
                                                                  existem mais nomes
    inicialize nomes com uma lista de numListas elementos VALOR BAIXO
    # leia um nome de cada lista
    para i até numlistas faça
        chame leia nome para para ler nomes[i] da listas[i]
    enquanto existem mais nomes faça
        # encontre o índice do menor nome
        inicialize menor com zero
        para i até numListas faça
             se nomes[i] for menor que nomes[menor] então
             faça menor receber i
        # escreva o menor na saída
        escreva nomes[menor] no arquivo saída
        # leia o próximo nome da listas[menor]
        chame leia nome para ler nomes[menor] da listas[menor]
    chame finalize() para fechar os arquivos
```

#### Programa principal

#### Função main

```
def main() -> None:
    if len(sys.argv) < 2:
        raise TypeError('Numero incorreto de argumentos!')
        kwaymerge(int(sys.argv[1]))</pre>
```

sys.argv é uma lista que contém as strings passadas na linha de comando

