DEEC
DEPARTAMENTO DE ENGENHARIA ELETROTÉCNICA E DE COMPUTADORES
TÉCNICO LISBOA

Licenciatura em Engenharia Electrotécnica e de Computadores (LEEC)

Algoritmos e Estruturas de Dados
Aula de Laboratório #03 - Manipulação de tabelas e listas

Objectivos

Neste laboratório vão ser usadas as estruturas de dados Tabela e Lista, e verificada na prática a gestão de memória alocada dinamicamente.

É fornecido o código base, a ser completado, para realizar as duas partes do trabalho detalhado abaixo, bem como um conjunto de dados de teste.

Sugere-se comprovar as diferentes utilizações de memória nos pontos 1.2, 1.4 e 2.2 recolhendo dados produzidos pelo Valgrind, e.g. `total heap usage: [...] NNN,NNN bytes allocated`

1 Manipulação de tabelas

Considere um programa com a seguinte especificação:

> UniqPal é um problema computacional em que se pretende ler de um ficheiro de texto e escrever num ficheiro resultado uma lista das palavras distintas encontradas, bem como o número de ocorrências de cada palavra. Cada palavra deve aparecer uma única vez no ficheiro de saída. O ficheiro de entrada é especificado na linha de comando; o nome do ficheiro de saída deve ser igual ao de entrada mas acrescentando `<.palavras>` no final (por exemplo, se o ficheiro de entrada for `xpto.txt`, o ficheiro de saída será `xpto.txt.palavras`).

O ficheiro `palTab.c`, que se encontra na sua área de trabalho, contém uma solução para o problema descrito acima.

1.1. Examine o código e o fluxograma correspondente (figuras no fim do enunciado). Repare na forma como o código se encontra estruturado em várias funções e que as palavras distintas lidas do ficheiro de entrada são guardadas numa tabela.

1.2. Complete o programa fornecido com o código necessário para a alocação e libertação da memória dinâmica necessária para a sua execução. Corra o Valgrind para verificar que a gestão de memória dinâmica está bem efectuada.

Exemplo: `valgrind --leak-check=full ./palTab ../test/doc1.txt`

1.3. O programa proposto, no ficheiro `palTab.c`, para concretizar o problema UniqPal faz uma gestão muito ineficiente de memória. Identifique as razões para essa gestão ineficiente e proponha uma solução que reduza a memória utilizada. Modifique o fluxograma fornecido em conformidade.

Nota: Não se pretende aqui alterar por completo a solução apresentada. Quer-se apenas, mantendo a estrutura de dados usada, reduzir o desperdício desnecessário da memória alocada. Na segunda parte do laboratório temos uma solução que poderá gastar menos memória com uma aproximação diferente ao problema.

1.4. Altere o ficheiro `palTab.c` de acordo com a solução que propôs. Corra o Valgrind para verificar que a gestão de memória dinâmica está bem efectuada

2 Manipulação de listas

A versão desenvolvida para a primeira parte do laboratório, mesmo depois das alterações que possam ser introduzidas, pode ainda utilizar mais memória do que a realmente necessária. Uma alternativa consiste em guardar as palavras, não numa tabela, mas numa lista. Considere então o seguinte algoritmo:

```
Abrir Ficheiro_de_Entrada
Ler Ficheiro_de_Entrada
   Enquanto houver palavras:
      Ler palavra
      Testar se palavra é nova na lista
         Sim: guardar na lista e inicializar o número de ocorrências a 1
         Não: incrementar o número de ocorrências da palavra
   Fechar Ficheiro_de_Entrada
Abrir Ficheiro_de_Saída
Para cada palavra da lista
   Escrever no Ficheiro_de_Saída o número de ocorrências, a frequência e a palavra
Fechar Ficheiro_de_Saída
Escrever no écran o número de palavras lidas e o número de palavras distintas
```

Nos ficheiros `main.c`, `words.c` e `list.c` é fornecido código, incompleto, que implementa o algoritmo indicado. Note as especificações adicionais nos pontos 2 e 3.

2.1. Examine o código e faça um fluxograma do programa. Por uma questão de legibilidade sugere-se que, tal como foi feito no Problema 1, se considere a possibilidade de fazer fluxogramas separados para procedimentos específicos como sejam os associados com o teste da palavra lida ser nova.

2.2. Complete o programa fornecido nos ficheiros indicados em que as palavras lidas dum ficheiro são armazenadas numa lista, cuja interface se define em `list.h`.

No final do programa escreve-se para ficheiro a listagem das palavras, o número de ocorrências e a frequência de ocorrências correspondentes:

(<frequência>=<número de ocorrências da palavra> / <número total de palavras>)

A escrita é feita percorrendo a lista do início para o fim.

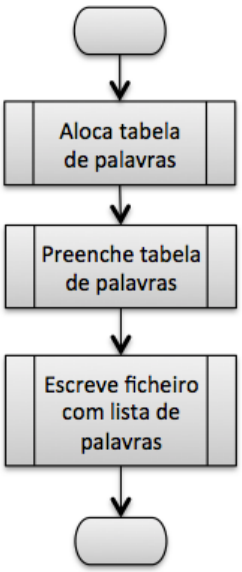
Assegure também a correcta libertação da memória alocada. Corra o Valgrind para verificar que a gestão de memória dinâmica está bem efectuada

2.3. Para finalizar, pretende-se que a listagem das palavras possa ser feita percorrendo a lista do início para o fim, ou do fim para o início, consoante um parâmetro adicional na linha de comandos tenha o valor `INICIO` ou `FIM`, respectivamente.

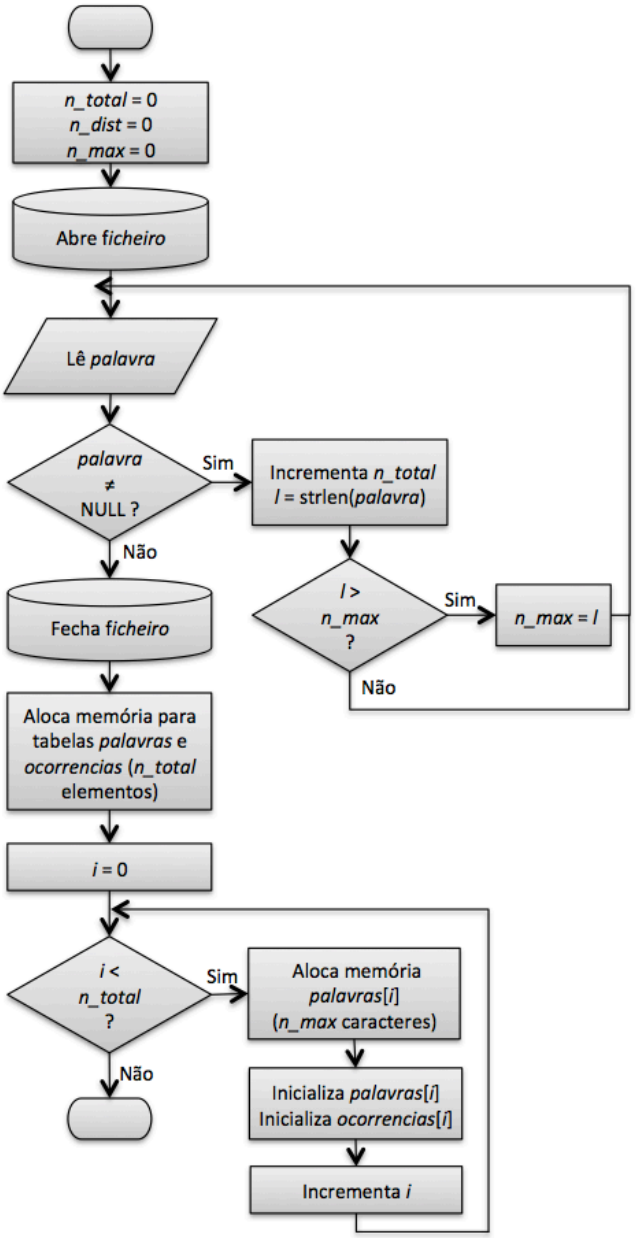
Modifique o código para cumprir esta especificação adicional. (Sugestão: Para a opção da listagem dos dados do fim para o início, pense numa implementação recursiva da função).

Nota: Pretende-se aqui que a informação dos elementos da mesma lista seja percorrida de formas distintas consoante o parâmetro adicional. Não se pretende que esses elementos sejam armazenados na lista de forma diferente em cada caso.

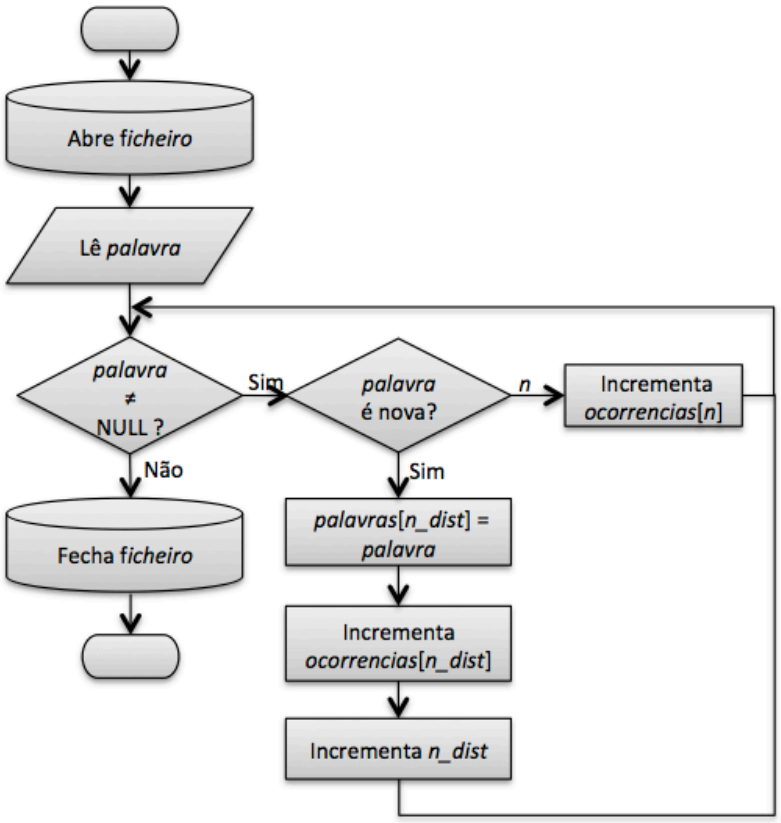
Fluxograma: palTab



Subrotina: Aloca tabela de palavras



Subrotina: Preenche tabela de palavras



Subrotina: Escreve ficheiro