SCC0201/501 (Introdução à Ciência de Computação II) — Prof. Moacir P. Ponti Junior

Trabalho Substitutivo

Implemente sua atividade sozinho sem compartilhar, olhar código de seus colegas, ou buscar na Internet. Procure usar apenas os conceitos já vistos nas aulas.

Esse trabalho é **opcional** e substituirá a menor nota recebida dentre os três trabalhos oficiais da disciplina, **mesmo que o presente trabalho receba nota menor do que as já obtidas**.

Esse trabalho necessita de tempo para ser implementado. Comece a pensar e trabalhar nele o quanto antes.

Aplicação de tabela hash na criação de um contador de ocorrências de palavras

Seu programa irá utilizar uma tabela hash com encadeamento para contar o número de ocorrências de palavras em um arquivo de entrada. O algoritmo básico executa, para cada palavra:

1. Realizar busca na tabela hash
2. Se a palavra não existe na tabela, insira a palavra com o valor de 1 ocorrência. Nesse caso você deverá criar um novo nó e ligá-lo à uma posição da tabela hash.
3. Se a palavra já existe na tabela, incremente o valor de ocorrência e não crie um novo nó.

A escolha de m: a tabela deverá armazenar no máximo 5.000 palavras, cada uma com no máximo 64 caracteres. Para escolher o número de compartimentos m em tabelas hash com encadeamento é considerada uma boa prática (CORMEN et al, 2002; SEDGEWICK, 1998) dividir o número máximo de elementos por 5 ou 3 e tomar um número primo próximo do resultado. Um exemplo: como $5.000/5 \approx 1.000$, uma sugestão é utilizar m igual aos primos 1009 ou 997.

A função hash: programe uma função hash e o mapeamento de compressão conforme achar melhor. Em aula foram apresentadas várias funções diferentes que podem ser utilizadas nesse caso.

As m **listas encadeadas**: o programa deverá manter cada lista encadeada ordenada alfabeticamente pela palavra. Você poderá escolher como realizar essa ordenação. Isso será importante para obter a saída (ordenada) posteriormente.

A saída (resultado): a saída do programa será composta de todas as palavras no arquivo que possuam: **3 ou mais caracteres e 3 ou mais ocorrências** no texto, uma por linha, com a contagem logo a seguir da palavra. A saída deverá estar em ordem alfabética. Para isso você pode utilizar uma função baseada na intercalação (*merge*) das listas que, conforme gera a saída ordenada, libera a memória ocupada por cada nó de cada lista encadeada.

Entrada

A entrada será composta de uma série de palavras conforme descrito na seção anterior. Palavras válidas são compostas por letras de [a..z] e possuem no mínimo 2 caracteres. Não é preciso verificar a corretude da entrada, pois todos os casos de teste estarão corretos. Para facilitar o tratamento da entrada, todas as palavras estarão em letras minúsculas, não serão acentuadas, e o texto não irá conter pontuação, parênteses, hífens ou outros caracteres especiais. Exemplo de entrada:

```
tabela hash ou tambem conhecida por tabela de espalhamento ou de dispersao
uma estrutura que associa chaves compartimentos na tabela funcao hash
ou de espalhamento um codigo hash gerado partir da chave realizado
espalhamento pela tabela
```

Saída

A saída será composta pela impressão das palavras (em ordem alfabética) que atendam aos requisitos definidos na seção anterior seguido por um espaço em branco e pela contagem das suas ocorrências, uma por linha. Exemplo:

```
espalhamento 3
hash 3
tabela 4
```

OBS1: Após a impressão de cada palavra e seu número de ocorrências, deverá haver uma quebra de linha por caracter `\n`.

OBS2: Note que as palavras "de" e "ou", apesar de possuirem 3 ou mais ocorrências cada, não foram impressas na saída pois possuem menos do que 3 caracteres. Para verificar o número de caracteres em uma string utilize a função `strlen()`.

Instruções e observações

O projeto será avaliado principalmente levando em consideração:

1. Processamento correto das entradas e saídas do programa;
2. Realização das tarefas descritas;
3. Bom uso das técnicas de programação (estruturas de controle, memória, funções, etc.);
4. Bom uso de estruturas de dados e algoritmos (escolha dos algoritmos e seus parâmetros, implementação do TAD, etc.);
5. Boa endentação, clareza do código e uso de comentários relevantes.

☹**Restrições:**☹

- **Não** deverá ser utilizada qualquer variável global.
- Desenvolver utilizando *Hash* estático com encadeamento externo (arranjo de ponteiros).
- As seguintes funções deverão **obrigatoriamente** ser implementadas e utilizadas:
  1. `int hash_code(char* key)` – deve receber como parâmetro a chave "key" e devolver o código *h*ash da mesma.
  2. `void insert_ht(Hash table[], char* key)` – deve receber como parâmetro a tabela *h*ash "table" e a palavra "key". Deve verificar se a palavra já existe na tabela. Caso não exista, inserir a palavra de forma ordenada na tabela com 1 ocorrência. Caso já exista, encontrar a palavra e incrementar o seu número de ocorrências.
- Você poderá criar outras funções se quiser (é preciso!) – lembre-se de tornar a função criada útil para os propósitos de reuso e abstração, e de comentá-la corretamente.
- Não poderão ser utilizadas bibliotecas com funções prontas, exceto: `string.h`, `stdio.h` e `stdlib.h`.

ATENÇÃO: o projeto deverá ser entregue **apenas** pelo SQTPM no formato `C`, escolhendo a opção `TrabalhoSUB`. O sistema receberá trabalhos **apenas** entre os dias 03/12/2010, às 0h00, e 06/12/2010 às 23h59 (horário do servidor SQTPM). **Não** serão aceitos trabalhos com atraso.

Podem utilizar o editor de sua preferência, mas compilem e executem o código utilizando o `gcc`:

```
gcc numusp.c -o numusp -Wall
```

Dúvidas conceituais deverão ser colocadas nas monitorias. Dificuldades em implementação, por favor, envie e-mail para o estagiário PAE com o assunto `[trab3]duvida`, anexando o código e especificando o problema.

A detecção de cópia de parte ou de todo código-fonte, de qualquer origem, implicará em reprovação direta no trabalho. Partes do código cujas **idéias** foram desenvolvidas em colaboração com outro(s) aluno(s) devem ser devidamente documentadas em comentários no referido trecho. O que **NÃO** autoriza a cópia de trechos de código nem a codificação em conjunto. Portanto, compartilhem idéias, soluções, modos de resolver o problema, mas não o código. Qualquer dúvida entrem em contato com o professor.