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

Trabalho 3 – *Hash Table* (v.2)

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

Aplicação de tabela hash na criação de um dicionário de sinônimos

Este trabalho consiste em implementar um dicionário de sinônimos para um idioma estrangeiro. Você deve criar um dicionário armazenado um conjunto de palavras e suas respectivas traduções. Porém, você pode se enganar na hora de inserir alguma palavra e, portanto, ser obrigado a atualizar o dicionário. Portanto, fique pronto para receber novas atualizações!

Este trabalho é uma adaptação do problema 10282 - *Babelfish. A versão original pode ser encontrada em:*
`http://uva.onlinejudge.org/index.php?option=com_onlinejudge&Itemid=8&category=14&page=show_problem&problem=1223`

Entrada

A entrada será composta de uma série de linhas, cada uma podendo conter:

1. Operador de inserção <dois pontos> palavra no idioma original <espaço em branco> palavra no idioma estrangeiro
2. Operador de remoção <dois pontos> palavra no idioma estrangeiro
3. Operador de busca <dois pontos> palavra no idioma estrangeiro

sendo:

- operador de inserção: caracter a ou A;
- operador de remoção: caracter r ou R;
- operador de busca: caracter f ou F;

Palavras válidas são iniciadas por letras de [a..Z] ou dígitos de [0..9]. Caracteres especiais são válidos desde que não iniciem uma palavra. Cada palavra não terá mais de **512** caracteres. Serão inseridos no máximo **10.000** pares de palavras.

Pode haver várias linhas em branco no arquivo de entrada, significando os dias do TUSCA que você faltou às aulas, ou mesmo antes dele, quando você já estava nos *esquentas*.

Saída

A saída será composta pelas traduções, que devem ser impressas assim que um operador de busca for encontrado. Se uma palavra não for encontrada, deve-se imprimir a expressão "`hein?`" (sem as aspas).

Exemplo de Entrada

```
a : dog ogday
a : cat atcay
A : pig igpay
a : froot ootfray

r : ootfray

f : atcay

f        : ittenkay

f : oopslay
a: loops oopslay
f : oopslay
```

OBS: você não está vendo torto por ressaca do TUSCA, tem quantidade indefinida de espaços em branco mesmo.

Dicas:

1. utilize o espaço de formatação do `scanf` para tratar a entrada (mais detalhes em: `http://www.lcad.icmc.usp.br/~paulovic/aulas/ED-I/entrada_saida.pdf`).
2. crie seus próprios arquivos de casos de entrada (`.in`) para testar seu algoritmo, facilitando o tratamento da condição de parada da entrada (o `scanf` retorna a constante `EOF` quando detecta que o arquivo de entrada terminou) e evitar ter que digitar manualmente a cada execução.

Exemplo de Saída

```
cat
hein?
hein?
loops
```

OBS: Após a impressão de cada tradução, deverá haver uma quebra de linha por caracter `\n`.

Instruções

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;
4. Boa endentação e uso de comentários no código.

☹**Restrições:**☹

- **Não** deverá ser utilizada qualquer variável global.
- Desenvolver utilizando: *Hash Estático Fechado (Endereçamento Aberto)* com arranjo circular e *overflow* progressivo. Atenção: a alocação da tabela hash deve ser estática, porém a alocação das palavras deve ser feita de forma a otimizar o uso de memória.
- As seguintes funções deverão **obrigatoriamente** ser implementadas e utilizadas:
  1. `int hash_code(Hash table[], char* key)` – deve receber como parâmetro a chave "key" e devolver o código *h*ash da *s*tring "key".
  2. `int find_ht(Hash table[], char* key)` – deve receber como parâmetro a tabela *h*ash "table" e a chave "key". Deve devolver a posição do item na tabela, caso o encontre, ou um endereço *h*ash inválido, caso contrário.
- 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.
- 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 `Trabalho3`. O sistema receberá trabalhos **apenas** entre os dias 26/11/2010, às 0h00, e 29/11/2010 às 23h59 (horário do servidor SQTPM, portanto enviem com antecedência). **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.