Parte III: Tabela de símbolos

Uma **tabela de símbolos** (ST, *symbol table* ou *map* ou *associative array* é uma estrutura de dados que armazena itens que são pares key-val, sendo key uma *chave* e val o *valor* associado a chave. Uma ST não possui chaves duplicadas. Quando o valor é irrelevante a ST se torna essencialmente a representação de um conjunto.

Tabelas de símbolos oferecem pelo menos duas operação:

- *inserção*: dada uma chave key e um valor val, insere a chave key e o seu valor val na tabela. Se a chave key já se encontra na tabela deve-se atualizar o valor associado a key. (Há quem prefira, não nós, gerar um erro, devolver o valor que já está na tabela, etc.)
- *busca*: dada uma chave key, devolve o seu valor val ou NULL caso key não esteja na tabela.

Um exemplo de tabela de símbolos é a classe dict() de Python.

STable

Nesta parte do projeto você implementará uma ST. A implementação é formada pelos arquivo stable.h e stable.c O arquivo stable.h contém a interface da ST, ou seja, a sua API. A chaves da ST são alguns tipos de dados e o valores strings. Seu trabalho será, portanto, implementar a interface descrita em stable.h no arquivo stable.c A interface stable.h **não deve ser alterada** e pode ser copiada daqui

Tabelas de dispersão e colisões por listas encadeadas

Há diversas formas de implementar uma ST. Nessa parte do projeto você deverá implementar uma ST através de uma tabela de dispersão (*hash table*). Em particular você deverá tratar as colisões por listas de encadeamento (*separate chaining*). Sobre hashing e separate chaining você pode consultar as notas de aula, as páginas Hashing (Java) ou Hashing (C) de Paulo Feofiloff, o capítulo 3.4 de Sedgewick & Wayne sobre *hash tables* e suas implementações, bem como o artigo sobre separate chaining na Wikipedia.

Programa de teste para a tabela de símbolos

Para testar sua implementação da tabela de símbolos você deve escrever um programa chamado freq que lê um arquivo de texto e imprime a freqüência (i.e., número de ocorrências) de cada palavra que ocorre no arquivo. Por

palavra entenda uma seqüência maximal de caracteres *não brancos* (! isblank(c) em C).

Seu programa deve receber o nome do arquivo com o texto de entrada com o o primeiro argumento da linha de comando. Assim, a invocação:

```
$ freq foo.txt
```

deve ler o arquivo de texto foo.txt e contar o número de ocorrências de cada palavra.

A saída gerada pelo seu programa deve seguir o seguinte formato. Cada palavra deve vir numa linha separada e, logo em seguida, sua frequência. As palavras devem aparecer em ordem lexicográfica, de acordo com a função strcmp(). As frequências devem todas aparecer na mesma coluna da saída, assim você deve colocar um número variável de espaços entre a palavra e sua frequência. A maior palavra deve ser separada de sua frequência por exatamente um espaço. Por exemplo:

```
Abacate          5
Zarabatana       2
aalborg          1
inconstitucional 70
```

Desafio

Faça a sua implementação da ST com redimensionamento (*rehashing*) de maneira a manter a hash table com fator de carga não superior a 10. O **fator de carga** de carga de uma hash table com m entradas e n itens é $\alpha = n/m$.