Tarefa T08 – *Linked Lists*

Considere que queremos armazenar símbolos e seus significados em listas simplesmente encadeadas distribuídas em um vetor de ponteiros para essas listas. Por exemplo, considere os símbolos e definições da Fig.1 que são distribuídos no vetor *hashtab* de tamanho 10 da Fig.2[1].

```
PI  3.14
TRUE  1
FALSE  0
MAXSIZE  81
G  9.81
EPS  0.001
MSG1  ...yes ok
```

Fig. 1

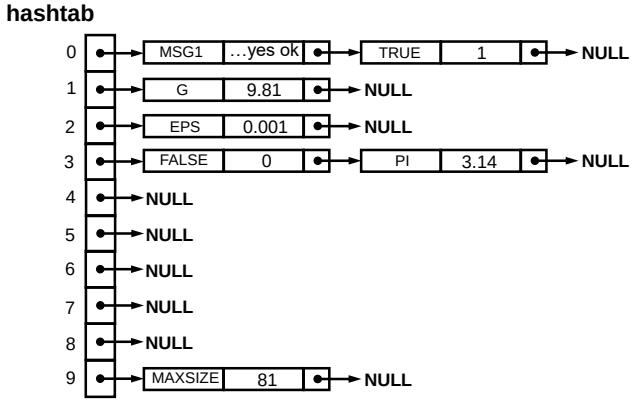

Fig. 2

O melhor cálculo para associar uma string a um índice (por exemplo, PI e FALSE estão associados ao índice 3 na Fig.2) não é muito fácil de encontrar. Isto é feito por uma função (chamada de *hashing function*) que dada uma string retorna um índice que garante a distribuição mais uniforme possível. Em futuros cursos, você vai aprender que uma *hashing function* simples e eficaz é a da Fig.3 (no exemplo acima HASHSIZE é 10).

```
unsigned int hash(char* s)
{
    unsigned int hashval;
    for (hashval = 0; *s; s++)
        hashval = *s + 31 * hashval;
    return hashval % HASHSIZE;
}
```

Fig. 3

```
struct nlist
{
    char * name;
    char * defn;
    struct nlist* next;
};
```

Fig. 4

Considere que o tipo estruturado para representar a lista encadeada é o da Fig. 4 (não use *typedef*). Considere também que o vetor *hashtab* é alocado estaticamente (com tamanho *HASHSIZE*).

Escreva as duas funções básicas de *hash table*: uma que, dado um nome e o vetor *hashtag*, retorna o endereço do elemento da lista que contém o nome (*i.e.*, retorna um `struct nlist*`)[2]; e a outra que, dados um nome, uma definição e o vetor *hashtag*, retorna o endereço do novo elemento de lista inserido (a inserção é sempre no início da lista existente). Nessa segunda função, use a primeira função para determinar se o nome sendo instalado já está presente (e se estiver, a nova definição deve substituir a antiga[3]). Essa segunda função deve retornar NULL se ocorrer algum problema de alocação dinâmica de memória (deixando para a main exibir mensagem de erro e interromper a execução).

Na *main*, use vetores de *strings* como dados de entrada, por exemplo:

```
char* symbols[] = { "PI", "TRUE", "FALSE" , "MAXSIZE", "G", "EPS", "MSG1"};
char* defs[] = { "3.14", "1", "0", "81", "9.81", "0.001", "...yes ok"};
```

O teste é usar a função de procura (a primeira função básica acima mencionada) para obter a definição de qualquer nome (e.g., se for dado EPS, tem-se impresso 0.001). Use exemplo completamente diferente do da Fig. 1.

---

[1] Compiladores C usam *hash table* para gerenciar as definições de constantes simbólicas feitas com *#define*.

[2] *Dica*: use a *hashing function* para calcular o índice (*n* = *hash*(...)) e obtenha o ponteiro para a cabeça da lista com *hashtab*[*n*].

[3] Não se esqueça de liberar a memória antes de criar uma nova.