Relatório - Parte I da Aventura

Emanuel Lima (9009493) e João Seckler (4603521)

26/09/2018

Para usar o programa teste.c, execute:

```
$ gcc -Wall -o teste elemento.c lista.c tabela.c teste.c
$ ./teste.c
```

O programa de teste deve imprimir uma mensagem na saída padrão se houver algum erro.

Optamos por definir dois tipos para elemento: "elemento" e "Elemento". O segundo é um ponteiro para o primeiro. A desvantagem é que precisamos criar duas funções "elemento_cria" e "elemento_destroi", mas em compensação nos parece que o restante do código, nos módulos de lista e tabelas, fica mais limpo e legível. Fazemos isso inspirados na implementação de um módulo de pilhas feita em MAC0121 - Algoritmos e Estruturas de Dados.

Com essa diferença, algumas funções têm sua declaração alterada:

`Lista insere(Lista l, Elemento *val)`

torna-se Lista

`insere(Lista l, Elemento val);`

`Elemento *busca(Lista l, char *n)`

torna-se

`Elemento busca(Lista l, char *n); e`

`Elemento *retira(Lista l, Elemento *val)`

torna-se

`Elemento retira(Lista l, Elemento val)`

Além disso, decidimos não seguir a sugestão do enunciado para a implementação da lista ligada à risca. Ele sugeria que o tipo lista fosse definido como:

```
  typedef struct {
  Elo * cabec ;
} Lista ;
```

No entanto, optamos por definir o tipo Lista simplesmente como um ponteiro para o tipo Elo. Assim, a cabeça de toda lista é criada pela função cria_lista, e é ela mesma um "Elo" cujo "val" não aponta para nenhum lugar relevante. Se é verdade que essa implementação gasta o espaço desse "val", que não é usado, também é verdade que evitamos novas definições, que pareciam tornar o código menos claro. O benefício apresentado pelo enunciado, qual seja, "o endereço da lista não muda com inserções e deleções", continua valendo, pela definição de "Lista". Essa estratégia é inspirada nas página sobre lista encadeada do Prof. Paulo Feofiloff (https://www.ime.usp.br/ pf/algoritmos/aulas/lista.html).

Na implementação da função

```
tabela_insere(Tabela T, char *n, Elemento val)
```

optamos por devolver um código de erro se o valor val associado à chave n já estiver armazenado no mesmo índice de T. Esse evento pode ter dois significados: (i) que o usuário tentou associar uma mesma chave a um mesmo valor mais de uma vez ou (ii) que o usuário associou duas chaves a um mesmo valor, e essas chaves colidem na função hash.

Implementamos a função hash da tabela de símboloso inspirados no seguinte material: http://www.cse.yorku.ca/ oz/hash.html