Projeto 1 de MP

Gabriel Martins de Miranda
130111350
Universidade de Brasília
Email:gabrielmirandat@hotmail.com

***Abstract*—**

1) Nome da função, parâmetros e significado dos parâmetros. Especificação da função.
2 Como cada função foi testada. Para cada um dos testes em cada função.
2.1) Nome de cada teste;
2.2) O que vai ser testado;
2.3) Qual deve a ser a entrada;
2.4) Qual deve ser a saída;
2.5) Qual é o critério para passar no teste ;
2.6) Se a sua função efetivamente passou no teste ou não.
Para os testes, foi usado um $Lista * lista\_teste$; global.
A ordem das funções segue a ordem dos TESTES!!

I. Funções e Testes

$Lista* \ cria(char * pal)$;
Especificação:
- inicializa um nó com uma palavra do texto ou digitada pelo usuário.
Parâmetros:
-pal = string associada que representa pLista− >palavra de cada nó.
Retorno:
-p = pLista recém-criado,que irá representar um nó na lista. Como ainda não foi associado á lista corrente, possui os ponteiros de anterior e proximo nulos e n_de_ocorrencias iguais a zero.

/insere $aaaa$ em lista_teste./
/verifica igualdade entre a string inserida e o campo string do nó./
/verifica se nó foi criado com ant==NULL e prox==NULL./
/verifica se ocorrencias==0, será incrementado quando o nó for inserido na lista./
/ $passou\_no\_teste$./

$lista\_teste = NULL$;
$lista\_teste = cria("aaaa")$;
$CU\_ASSERT\_STRING\_EQUAL("aaaa", lista\_teste- > palavra)$;
$CU\_ASSERT\_PTR\_NULL(lista\_teste- > ant)$;
$CU\_ASSERT\_PTR\_NULL(lista\_teste- > prox)$;
$CU\_ASSERT(0 == lista\_teste- > ocorrencias)$;

$int \ vazia(Lista * l)$;
Especificação:
- verifica se pLista é NULO, ou seja, se não há elementos na lista.
Parâmetros:
- l = ponteiro para inicio da lista.
Retorno:
- booleano = 1(verdade== lista vazia) 0(falso== lista possui elementos).

/ inicia um ponteiro tipo lista nulo./
/ testa o retorno de $vazia$ para uma lista NULA e outra com elementos./
/ $passou\_no\_teste$./

$Lista * aux$;
$aux = NULL$; / inicia um ponteiro tipo lista nulo./
$CU\_ASSERT(1 == vazia(aux))$;
$CU\_ASSERT(0 == vazia(lista\_teste))$;

$Lista* \ insere\_ordenado(Lista * l, char * pal)$;
Especificação:
-insere elemento na lista corrente na ordem lexicografica crescente do ASCII, sendo que letras maiusculas são menores lexicograficamente do que as minusculas.
Parâmetros:
-l = ponteiro que representa o início da lista.
-pal = string associada que representa pLista− >palavra de cada nó. Gerada a partir do arquivo texto ou inserção do usuário.
-tam_atual = variavel global que representa o tamanho atual de elementos na lista.
Retorno:
-l = ponteiro que representa o inicio da lista atualizado, sendo que a atualização so se faz necessária se o elemento for inserido na lista vazia ou for o novo primeiro elemento da lista..

/É inserido *bbbb* e *cccc* em lista_teste./
/verifica igualdade entre a string inserida e o campo string do nó./
/verifica se ponteiros de prox e ant estão funcionando adequadamente em cada nó./
/verifica se o anterior do primeiro elemento e o proximo do ultimo elemento são NULOS./
/ $passou\_no\_teste$./

$lista\_teste = insere\_ordenado(lista\_teste, "bbbb")$;

$lista\_teste = insere\_ordenado(lista\_teste, "cccc");$
$CU\_ASSERT\_STRING\_EQUAL("aaaa", lista\_teste->palavra);$
$CU\_ASSERT\_STRING\_EQUAL("bbbb", lista\_teste->prox->palavra);$
$CU\_ASSERT\_STRING\_EQUAL("cccc", lista\_teste->prox->prox->palavra);$
$CU\_ASSERT\_STRING\_EQUAL("aaaa", lista\_teste->prox->ant->palavra);$
$CU\_ASSERT\_STRING\_EQUAL("bbbb", lista\_teste->prox->prox->ant->palavra);$
$CU\_ASSERT\_STRING\_EQUAL("cccc", lista\_teste->prox->prox->ant->prox->palavra);$
$CU\_ASSERT\_PTR\_NULL(lista\_teste->ant);$
$CU\_ASSERT\_PTR\_NULL(lista\_teste->prox->prox->prox);$

$Lista* construcao\_lista(Lista * l, FILE * fp);$
Especificação:
-percorre o arquivo texto de onde as palavras estão sendo tiradas, armazena-as numa string e manda para a criação de nós ordenados.
Parâmetros:
-l= referência para primeiro nó da lista.
-fp= ponteiro para o arquivo texto.
Retorno:
-l= lista criada com as palavras extraidas do texto.

/É criado um arquivo texto de teste aux.txt./
/As palavras tiradas de aux.txt foram colocadas na lista./
/verifica se ponteiros de prox e ant estão funcionando adequadamente em cada nó./
/verifica se os elementos foram colocados na ordem correta./
/ $passou\_no\_teste$./

$FILE * arq;$
$arq = fopen("aux.txt","w");$
$fprintf(arq," * dddd eeee((");$
$fclose(arq);$
$arq = fopen("aux.txt","r");$
$lista\_teste = construcao\_lista(lista\_teste, arq);$
$CU\_ASSERT\_STRING\_EQUAL("dddd", lista\_teste->prox->prox->prox->palavra);$
$CU\_ASSERT\_STRING\_EQUAL("eeee", lista\_teste->prox->prox->prox->prox->palavra);$

$Lista* retirar\_elemento(Lista * l, char * pal);$
Especificação:
-Retira a palavra inserida pelo usuário. Ela retira também todas as ocorrencias do elemento.
Parâmetros:
-l= referência para primeiro nó da lista.
-pal= palavra que o usuário quer retirar da lista.
-mudanca_lista = variavel global que indica se a lista foi alterada ou não. Usada para auxiliar o vetor de pesquisa binária.
-tamanho_atual = numero de elementos atualmente na lista.
Retorno:
-l = inicio de arquivo atualizado.

/Alguns elementos foram retirados./
/Foi testado se o ponteiro de primeiro elemento foi atualizado e se manteu-se a corretude da ordem e dos ponteiros da lista./
/ $passou\_no\_teste$./

$lista\_teste = retirar\_elemento(lista\_teste, "aaaa");$
$lista\_teste = retirar\_elemento(lista\_teste, "bbbb");$
$lista\_teste = retirar\_elemento(lista\_teste, "eeee");$
$CU\_ASSERT\_STRING\_EQUAL("cccc", lista\_teste->palavra);$
$CU\_ASSERT\_STRING\_EQUAL("dddd", lista\_teste->prox->palavra);$

$int\ buscar\_elemento(Lista * l, char * valor);$
Especificação:
-A partir da palavra inserida pelo usuário é feita uma busca binária para ver se o elemento pesquisado existe. É retornado o numero de ocorrências.
Parâmetros:
-l= referência para primeiro nó da lista.
-mudanca_lista = variavel global que indica se a lista foi alterada ou não. Usada para auxiliar o vetor de pesquisa binária.
-tamanho_atual = indica o tamanho atual da lista.
-valor= palavra que o usuário quer buscar na lista.
Retorno:
-n_ocorre = numero de vezes que a palavra pesquisada ocorre. É ZERO caso a palavra pesquisada não existir.

/Foi testado se o retorno da função estava condizente com a lista./
/Por algum motivo, a string $dddd$ não foi encontrada na lista, e o retorno da função foi ZERO ocorrencias./
/ $nao\_passou\_no\_teste$./

$CU\_ASSERT(1 == buscar\_elemento(lista\_teste, "cccc"));$
$CU\_ASSERT(1 == buscar\_elemento(lista\_teste, "dddd"));$
$CU\_ASSERT(0 == buscar\_elemento(lista\_teste, "eeee"));$

$Lista* inserir\_elemento(Lista * l, char * elemento);$
Especificação:
-Insere uma palavra escolhida pelo usuário, além das tiradas do arquivo texto.
Parâmetros:
-l= referência para primeiro nó da lista.
-mudanca_lista = variavel global que indica se a lista foi alterada ou não. Usada para auxiliar o vetor de pesquisa binária.
-elemento= palavra que o usuário quer inserir na lista.
Retorno:
-l = inicio de arquivo atualizado.

/Os elementos antes retirados foram colocados novamente./
/Foi testado se a função inseriu os elementos na posição correta./
/ $passou\_no\_teste$./

$lista\_teste = inserir\_elemento(lista\_teste, "aaaa");$
$lista\_teste = inserir\_elemento(lista\_teste, "eeee")$
$CU\_ASSERT\_STRING\_EQUAL("aaaa", lista\_teste->palavra);$
$CU\_ASSERT\_STRING\_EQUAL("eeee", lista\_teste->prox->prox->prox->palavra);$

$void\ libera(Lista * l);$
Especificação:
-libera a memória alocada pela lista duplamente encadeada.
Parâmetros:
-l= referencia para primeiro nó da lista.
Retorno:
-void.

/Foi testado se o ponteiro deixou de existir./
/ $passou\_no\_teste$./

$libera(lista\_teste);$
$CU\_ASSERT\_PTR\_NOT\_NULL(lista\_teste);$

$void\ imprime(Lista * l);$
Especificação:
-mostra todos os elementos presentes na lista.
Parâmetros:
-l= referência para primeiro nó da lista.
-tam_atual = var global que representa o tamanho atual da lista.
Retorno:
-void.

$Nao\ foi\ encontrada\ utilidade\ nem\ motivo\ para\ testar\ esta\ funcao.$

$void\ tela(Lista * l);$
Especificação:
-Um menu é apresentado ao usuário em que ele pode escolher o que fazer com a lista.
Parâmetros:
-l= referência para primeiro nó da lista.
Retorno:
-void.

$Nao\ foi\ encontrada\ utilidade\ nem\ motivo\ para\ testar\ esta\ funcao.$