Lista Encadeada

O que fazer

Criar um tipo abstrato de dados (TAD) **Lista** por meio de encadeamento de nós. Um nó é uma posição de memória que guarda dados e um apontador (endereço) para o próximo nó; todo nó é alocado dinamicamente. Assim, o tamanho da lista não precisa ser definido antecipadamente (como ocorre, por exemplo, com a lista implementada por vetor). A Figura 1 ilustra esse tipo de lista.

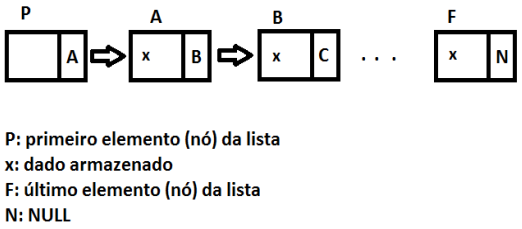

Figura 1. Lista com encadeamento de nós (www.vivaolinux.com.br).

O TAD Lista deve ser capaz de lidar com várias listas; não há limites à quantidade de variáveis do tipo Lista que um programador pode declarar em um programa. As funções que devem estar disponíveis para atuar sobre esse tipo de dado estão apresentadas nas figuras 2 e 3. No que concerne a essas funções, observar o que segue: todas as funções retornam um objeto que pode ser testado para verificar se a respectiva função foi realizada com sucesso; a única função que apresenta (imprime) algo na saída padrão (*stdout*) é a *showAll*; apresentação do

```
// count
// -----
// Return the quantity of items in l;
// or return -1, if l is not found.

int count (void * l);


// create
// ------
// Return a pointer to the created list, whose label is s;
// or return NULL, if the list was not created.

void * create (char *s);


// destroy
// -------
// Return NULL if l was destroyed; return l otherwise.
// Destroying l implies in deallocating all nodes of l.

void * destroy (void *l);


// find
// ----
// Return a pointer to the item whose id is m;
// or return NULL, if no item with id m was found.

Item * find (void *l, int m)
```

Figura 2. Protótipo das funções count, create, destroy e find.

resultado das demais funções, incluindo eventuais mensagens de erro, é obrigação da aplicação. As eventuais mensagens de erro devem apresentadas pela aplicação na saída padrão de erro (*stderr*). As mensagens de erro

```
// idl
// ---
// Return a pointer to list of items whose label is s;
// or return NULL, if does not exist a list with label s.

void * idl (char *s);


// insert
// ------
// Return l, if m is inserted into l; or return NULL, if
// either l was not found or space for m was not allocated.
// m is inserted into l in ascending order.

void * insert (void *l, Item m);


// label
// -----
// Return the label of l; or return NULL, if l is not found.

char * label (void *l);


// showAll
// -------
// Present all items of l.
// Return l; or return NULL, if l was not found.

void * showAll (void *l);
```

Figure 3. Protótipo das funções idl, insert, label e showAll.

Criar também uma aplicação que crie várias listas e aplique operações sobre as mesmas. As operações devem estar especificados em uma arquivo tipo texto; a aplicação dever ler esse arquivo, pela entrada padrão (*stdin*). Uma operação é uma quádrupla que possui o seguinte formato **<fn> <rl> <c1> <c2>**, tal que:

<fn> é uma letra que representa uma das funções das Figuras 2 e 3, conforme Tabela 1. Portanto, os valores possíveis para <fn> são c, i, w, u, e d. A coluna FUNÇÃO relaciona cada uma das letras às funções correspondentes da TAD Lista.

<rl> é uma cadeia de caracteres (*string*) que corresponde ao rótulo da lista.

<c1> é um número.

<c2> é uma cadeia de caracteres (*string*).

Entre cada campo da quádrupla há 1 (um) espaço em branco. Os campos <c1> e <c2>, respectivamente, em geral, têm valor **-1** e **NULL**. No entanto, quando <fn> = **i**, esses campos guardam os dados de um item, respectivamente, um número e um *string*.

Tabela 1. Sintaxe e semântica das operações da aplicação.

<FN>	FUNÇÃO	EXEMPLO DE USO	EXPLICAÇÃO
c	*create*	c aluno -1 NULL	cria a lista **aluno**
i	*insert*	i aluno 12 Fulano	insere na lista **aluno** os dados **12 Fulano**
w	*showAll*	w aluno -1 NULL	apresenta os dados contidos na lista **aluno**

u	*count*	u aluno -1 NULL	apresenta a quantidade de dados contidos na lista **aluno**
d	*destroy*	d aluno -1 NULL	destrói a lista **aluno**

Um exemplo de um arquivo txt (de operações) é apresentado na Figura 4. A primeira linha do arquivo, **c aa -1 NULL**, indica que a aplicação deve criar a Lista **aa**, via função *create*; a linha seguinte, **i aa 12 Fulano**, indica que a aplicação deve inserir, via função *insert*, o item 12 Fulano, na Lista **aa**; a linha seguinte é outra operação de inserção, na mesma lista; a linha **w aa -1 NULL** indica que a aplicação deve apresentar (imprimir) os itens da Lista **aa;** finalmente, a linha **u aa -1 NULL** indica que a aplicação deve retornar o número correspondente à quantidade de itens inseridos na Lista **aa**. Por fim, um item é uma *struct* cuja definição é apresentada na Figura 5. O tamanho do membro nome é 20.

```
c aa -1 NULL
i aa 12 Fulano
i aa 43 Beltrano
w aa -1 NULL
u aa -1 NULL
```

Figura 4. Exemplo de um arquivo de operações aplicada à Lista **aa**.

```
struct item
{
   int i;
   char *nome;
};
```

Figura 5. Estrutura do item a ser inserido em uma lista.

Como fazer

Este é um trabalho a ser realizado em grupo de **três** alunos; **todos os grupos** devem ter três alunos. A formação dos grupos é de livre escolha dos alunos, contudo, se um grupo tiver menos do que três alunos, o professor poderá atuar para na formação desse grupo.

O TAD e a aplicação devem ser construídos com a linguagem de programação C utilizando o padrão de codificação GNU (GNU Coding Standards).

É necessário apresentar os arquivos .h e arquivos .c envolvidos neste trabalho. Em outras palavras, o TAD deverá ser construído independente da aplicação que irá fazer uso do mesmo. Por sua vez, a aplicação, que fará uso do TAD, deve estar presente em um arquivo . c específico. A quantidade de arquivos a ser criada é uma questão de decisão de projeto do grupo. Contudo, obrigatoriamente, quatro arquivos precisam ser apresentados, conforme Tabela 2.

ARQUIVO	CONTEÚDO
lista.h	protótipos das funções implementadas no arquivo lista.c
lista.c	implementação das funções especificadas no arquivo lista.h
aplica.c	implementação que declara e variáveis do tipo Lista e aplica operações sobre as mesmas
readme	explicação sobre: o conteúdo de todos os arquivos contidos nesta tabela; outros arquivos, além daqueles mencionados nesta tabela, que eventualmente tenha sido criados; como se deve compilar todos os programas; como utilizar o programa aplica.c

A apresentação do resultado de uma operação deve ser claro, completo e identificar a lista correspondente ao resultado. Por exemplo, o resultado da operação w aa -1 NULL deve ser

uma listagem com todos os itens da Lista **aa**, além disso, **aa** dever ser o cabeçalho dessa listagem.

Uma mensagem de erro deve indicar qual foi o erro ocorrido de forma clara e sucinta, E qual foi a operação que ocasionou o erro. Por exemplo, considere que a operação **i qq 77 pendrive** foi capturada pela aplicação, mas a Lista **qq** não havia sido criada previamente. Ao ser submetida ao TAD Lista, essa operação deve retornar um erro. Especificamente, a função *insert* deve retornar um erro e a aplicação deve emitir uma mensagem adequada identificando a operação que provocou o erro.

Como entregar

Enviar uma mensagem com uma versão plenamente operacional do TAD e da aplicação para o email autran@ufu.br com os demais campos preenchidos tal como a abaixo:

- remetente – email do membro do grupo responsável pelo envio do trabalho
- assunto – [GSI006 ED1 2007/1] lista encadeada
- corpo da mensagem – número de matrícula e nome de cada membro do grupo (um por linha)
- anexo – arquivo comprimido do tipo **.tar.gz** contendo todos os arquivos .c e .h do trabalho. O nome desse arquivo comprimido deve ser
  gsi006-2017-01-linkedList-**mat**.tar.gz
  tal que mat corresponde ao número de matrícula do remetente da mensagem.