Assertivas de entrada e saída do módulo Lista:

- **Criar Lista genérica duplamente encadeada (LIS_CriarLista)**

Assertivas de entrada:

- Argumento da função referencia a função de destruição do valor contido nos elementos a serem excluídos.
- Argumento da função é diferente de NULL.

Assertivas de saída:

- Ponteiro corrente referencia lista alocada dinamicamente, com tamanho da struct de Lista.
- Conteúdo da lista com ponteiro corrente igualado a NULL e valor de pLista->ExcluirValor é igualado ao ponteiro passado como parâmetro que referencia a função de destruição do valor.

- **Destruir Lista (LIS_DestruirLista)**

Assertivas de entrada:

- Ponteiro corrente referencia a lista, na qual não pode ser NULL, a ser excluída.

Assertivas de saída:

- Lista referenciada inicialmente foi excluída, os elementos contidos nela inicialmente foram liberados e a cabeça da lista foi limpa.
- O parâmetro ponteiro para lista não foi modificado.

- **Esvaziar lista (LIS_EsvaziarLista)**

Assertivas de entrada:

- Ponteiro corrente referencia a lista, na qual não pode ser NULL, a ser esvaziada.

Assertivas de saída:

- Lista referenciada inicialmente teve seus elementos liberados e a cabeça da lista foi limpa.

- **Inserir elemento antes (LIS_InserirElementoAntes)**

Assertivas de entrada:

- Ponteiro corrente referencia endereço da lista, após ao qual será inserido o elemento.
- Ponteiro para o valor, que pode ser NULL, do novo elemento a ser criado para ser inserido na lista.
- Caso a lista esteja vazia, o elemento a ser inserido será o primeiro da lista.

Assertivas de saída:

- Lista referenciada inicialmente teve o elemento criado, com o valor passado como parâmetro, inserido no pAnt (ponteiro para o anterior do corrente).
- Se o elemento corrente era NULL, o elemento foi inserido como ponteiro para origem e como fim da lista (primeiro elemento da lista).

- Se o elemento anterior ao corrente era NULL, o elemento foi inserido como ponteiro para origem da lista.

- **Inserir elemento após (LIS_InserirElementoApos)**

Assertivas de entrada:

- Ponteiro corrente referencia endereço da lista, anterior ao qual será inserido o elemento.
- Ponteiro para o valor, que pode ser NULL, do novo elemento a ser criado para ser inserido na lista.
- Caso a lista esteja vazia, o elemento a ser inserido será o primeiro da lista.

Assertivas de saída:

- Lista referenciada inicialmente teve o elemento criado, com o valor passado como parâmetro, inserido no pProx (ponteiro para o próximo do corrente).
- Se o elemento corrente era NULL, o elemento foi inserido como ponteiro para origem e como fim da lista (primeiro elemento da lista).
- Se o elemento próximo ao corrente era NULL, o elemento foi inserido como ponteiro para fim da lista.

- **Excluir elemento (LIS_ExcluirElemento)**

Assertivas de entrada:

- Ponteiro corrente referencia lista ao qual elemento corrente será excluído.

Assertivas de saída:

- Elemento anterior ao corrente, passado inicialmente, (pAnt) é o novo elemento corrente, caso esse elemento anterior existia no ponteiro para lista passado inicialmente.
- Elemento após ao corrente, passado inicialmente, (pProx) é o novo elemento corrente, caso esse elemento após existia e não existia um elemento anterior ao corrente, no ponteiro para lista passado inicialmente.
- A lista era vazia, caso nem o elemento anterior ao corrente, nem o elemento após ao corrente, passado inicialmente, existia no ponteiro para lista.

- **Obter referência para o valor contido no elemento (LIS_ObterValor)**

Assertivas de entrada:

- Ponteiro corrente referencia endereço da lista na qual será obtido o valor contido no elemento corrente da lista.

Assertivas de saída:

- Ponteiro para o valor referenciado pelo elemento corrente da lista passada inicialmente, foi retornado, caso a lista não era NULL, nem o elemento corrente.
- Foi retornado NULL, caso o elemento corrente era NULL ou a lista.

- **Ir para o elemento inicial (IrInicioLista)**

Assertivas de entrada:

- Ponteiro corrente referencia endereço da lista na qual será manipulada.

Assertivas de saída:

- Estado inicial foi preservado (nada foi feito), caso a lista passada inicialmente era NULL.
- Caso contrário, o ponteiro para o elemento corrente tornou-se o ponteiro para a origem da lista (o primeiro elemento da lista inicial).

- **Ir para o elemento final (IrFinalLista)**

Assertivas de entrada:

- Ponteiro corrente referencia endereço da lista na qual será manipulada.

Assertivas de saída:

- Estado inicial foi preservado (nada foi feito), caso a lista passada inicialmente era NULL.
- Caso contrário, o ponteiro para o elemento corrente tornou-se o ponteiro para o fim da lista (o último elemento da lista inicial).

- **Avançar elemento (LIS_AvancarElementoCorrente)**

Assertivas de entrada:

- Ponteiro corrente referencia endereço da lista na qual será manipulada.
- Número inteiro correspondente ao número de elementos a andar na lista, a partir do elemento corrente. O número inteiro passado pode ser positivo, negativo ou zero

Assertivas de saída:

- A lista passada inicialmente é vazia, caso o elemento corrente inicial era NULL.
- O ponteiro para o elemento corrente referencia o número de elementos retrocedidos, caso o número inteiro passado como parâmetro era negativo e a lista não chegou ao fim.
- O ponteiro para o elemento corrente referencia o número de elementos, caso o número inteiro passado como parâmetro era positivo e a lista não chegou ao fim.
- O ponteiro para o elemento corrente referencia o fim da lista, caso o número inteiro passado como parâmetro era positivo e a lista chegou ao fim.
- O ponteiro para o elemento corrente referencia a origem da lista, caso o número inteiro passado como parâmetro era negativo e a lista chegou ao início.
- A lista foi verificada como vazia ou não, caso o número inteiro passado como parâmetro era igual a zero.

- **Procurar elemento contendo valor (LIS_ProcurarValor)**

Assertivas de entrada:

- Ponteiro corrente referencia endereço da lista na qual será procurado o valor dado.
- Ponteiro para o valor que será procurado, podendo ser NULL.

Assertivas de saída:

- A lista passada inicialmente é vazia, caso o elemento corrente era NULL.

- Ponteiro para elemento corrente faz referência ao valor encontrado, sendo o primeiro elemento corrente.
- Ponteiro para elemento corrente continua o mesmo referenciado inicialmente, indicando que não foi encontrado o valor procurado.

Assertivas de entrada e saída do módulo Vértice:

- **Criar vértice (VER_CriarVertice)**

Assertivas de entrada:

- Ponteiro para endereço do vértice a ser criado, podendo ser inexistente.
- Ponteiro para o valor que será inserido no vértice a ser criado.
- Ponteiro para chave que será inserida no vértice a ser criado.
- Argumento da função referencia a função de destruição do valor referenciado pelo vértice a serem excluídos.

Assertivas de saída:

- Nada foi feito e condição de OK retornada, caso o endereço passado inicialmente era inexistente.
- Vértice alocado dinamicamente, com tamanho de uma struct do tamanho de vertice, referenciado pelo ponteiro do endereço passado inicialmente, os respectivos valores, passados como parâmetros, atribuídos ao vértice e condição de retorno OK.
- Vértice destruído, anteriormente referenciado pelo ponteiro de endereço passado inicialmente, caso já existia um vértice nesse endereço e novo vértice criado.

- **Inserir aresta (VER_InserirAresta)**

Assertivas de entrada:

- Ponteiro para o vértice, podendo ser inexistente, no qual será inserido a aresta.
- Ponteiro para a aresta, podendo ser inexistente, a qual será inserida.

Assertivas de saída:

- Condições de retorno respectivas foram retornadas, caso o endereço da aresta ou o endereço do vértice passados como parâmetro, não existiam.
- Aresta reflexiva inserida, no ponteiro para reflexiva do vértice inicial, caso a origem e destino da aresta eram iguais.
- Nada foi feito e condição de retorno OK, caso a aresta já existia no vértice.
- Aresta inserida, com ponteiro para antecessor do vértice referenciando o endereço da aresta passado inicialmente, caso a origem da aresta era diferente da chave do vértice.
- Aresta inserida, com ponteiro para sucessor do vértice referenciando o endereço da aresta passado inicialmente, caso o destino da aresta era diferente da chave do vértice.

- **Remover aresta (VER_RemoverAresta)**

Assertivas de entrada:

- Ponteiro para o vértice, podendo ser inexistente, no qual a aresta a ser removida está contida.
- Ponteiro para a aresta, podendo ser inexistente, a qual será removida.

Assertivas de saída:

- Condições de retorno respectivas foram retornadas, caso o endereço da aresta ou o endereço do vértice passados como parâmetro, não existiam.
- Nada foi modificado e condição de retorno Erro de Inserção retornada, caso a origem e destino da aresta eram diferentes da chave contida no vértice.
- Aresta reflexiva destruída, com ponteiro para reflexiva do vértice inicial igualado a NULL, caso a origem e destino da aresta eram iguais.
- Aresta removida e destruída, com ponteiro para antecessor do vértice tendo seu elemento excluído, caso a origem da aresta era diferente da chave do vértice.
- Aresta removida e destruída, com ponteiro para sucessor do vértice tendo seu elemento excluído, caso o destino da aresta era diferente da chave do vértice.

- **Modificar tipo do vértice (VER_ModificarTipo)**

Assertivas de entrada:

- Ponteiro para vértice corrente a ter seu tipo modificado, podendo ser inexistente.
- Valor dado a ser modificado do tipo VER_TipoVer (podendo ser inicial = 0, intermediário = 1 e final = 2).

Assertivas de saída:

- Nada foi modificado e condição de retorno Vertice Não existe retornada, caso vértice inexistente.
- Nada foi modificado e condição de retorno Erro Inserção retornada, caso tipo inical passado como parâmetro diferente de 0, 1 ou 2.
- Ponteiro para tipo do vértice passado inicialmente modificado com o novo valor de tipo.

- **Destruir Vértice (VER_DestruirVertice)**

Assertivas de entrada:

- Ponteiro para vértice corrente a ser destruído, podendo ser inexistente.

Assertivas de saída:

- Nada foi modificado, caso vértice inexistente.
- Ponteiro para aresta reflexiva liberado, ponteiro para excluir valor referenciado pelo valor do vértice, respectivos antecessores e sucessores existentes do vértice destruídos e vértice liberado.
- Depois de destruído o vértice nao foi anulado isto deve ser feito pelo caller da função.

- **Marcar visitado no vértice (VER_MarcarVisitado)**

Assertivas de entrada:

- Ponteiro para vértice corrente a ser marcado, podendo ser inexistente.

Assertivas de saída:

- Nada foi modificado e condição de retorno Vertice Não existe retornada, caso vértice inexistente.
- Ponteiro para visitado do vértice corrente foi marcado como visitado, caso inicialmente estivesse marcado como não visitado ou foi marcado como não visitado, caso inicialmente estivesse marcado como visitado.

- **Get Valor do vértice (VER_getValor)**

Assertivas de entrada:

- Ponteiro para vértice corrente, podendo ser inexistente no qual o valor será obtido.
- Ponteiro para endereço aonde o valor do vértice será colocado, podendo ser inválido.

Assertivas de saída:

- Nada foi obtido e condição de retorno OK retornada, caso o endereço passado como parâmetro era inválido.
- Nada foi obtido e condição de retorno Vertice Não existe retornada, caso vértice inexistente.
- Ponteiro do endereço passado como parâmetro teve o valor atribuído com o ponteiro para o valor do vértice passado inicialmente.