**Argumentação de Corretude para a Função Avançar Elemento Corrente**

*Argumentação de seqüência:*

* AE - Assertivas de entrada

- Recebe uma lista já criada (mas pode ser vazia)

- Recebe um numero de elementos que pode ser positivo ou negativo ou zero (que é a quantia que deve-se avançar)

- Valem as assertivas estruturais da lista duplamente encadeada com cabeça

* AS - Assertivas de saída:

- Valem as assertivas estruturais da lista duplamente encadeada com cabeça

- O ponteiro corrente aponta para o elemento após numElem elementos, no caso que era o último, ele continua apontando para o mesmo elemento. Se numElem fosse negativo, elemento corrente aponta para numElem antes dessa quantidade de elementos, ou se a lista chegou ao inicio antes da quantidade, aponta para a origem da lista.

* AI1: Lista não está vazia. Elemento corrente aponta para índice válido.
* AI2: numElem é menor ou igual a 0
* AI3: O auxiliar pElem recebe o elemento corrente do pLista.
* AI4: ou o pElem é igual a NULL ou i = 0.
* AI5: auxiliar pElem recebe o elemento corrente do pLista.
* AI6: ou o pElem é igual a NULL ou i = 0.

*Arg. de Seleção:*

1. AE && (C==T) + B1 => AS

AE: AE principal

Pela assertiva de entrada, a lista pode estar vazia. Como (C==T), ela realmente está vazia. Nesse caso, B1 é executado e a função retorna uma condição de lista vazia. Valendo AS = AS principal.

1. AE && (C==F) +B2 => AS

Como (C==F), a lista não está vazia, e vale a AS que é AI1.

*Arg de Seleção 2:*

1. AE && (C==T) + B1 => AS

Pela assertiva de entrada, o numero de elementos pode ser positivo. Como (C==T), ele é. Nesse caso B2 é executado. AS: índice aponta para o numElem.

1. AE && (C==F) +B2 => AS

Como (C==F), o numero de elementos não é maior de zero, e valem a AS que é a AI2.

*Arg. Repetição:*

AE: AI3, AS: AI4

Ainv:

-existem dois conjuntos: a percorrer e já percorrido.

-indice aponta para elemento de a percorrer.

1. AE=> Ainv

Pela AE, índice aponta para o numElem e o conjunto já percorrido está vazio. Vale a Ainv.

1. AE && (C==F) => AS

(C==F) quando pElem==NULL. O primeiro ciclo não se completa (por causa do break) e vale a AS. Nesse caso, não teria outra situação para (C==F), pois a repetição se encontra dentro de um if que garante que o índice vai ser >0 (no primeiro ciclo).

1. AE && (C==T)+B => Ainv

Como (C==T), pElem != NULL. Logo, o índice é atualizado e o pElem agora é igual ao pElem->prox. Vale a Ainv.

1. Ainv && (C==T)+B => Ainv

Para que a Ainv continue valendo, B deve garantir que um elemento passe do conjunto a percorrer para o já percorrido e índice aponte para outro elemento de a percorrer.

1. Ainv && (C==F) => AS

No último ciclo, pElem==NULL ou i não é maior que 0. Nesse caso vale a AS.

1. Término

Como a cada ciclo, um elemento passa do conjunto a percorrer para já percorrido, e a quantidade de elementos desse conjunto e o tamanho da lista é finito, a repetição termina em um numero finito de passos.

*Arg.Seleção:*

1. AE && (C==T) + B => AS

Pela assertiva de entrada (AI4), o pElem pode ser diferente de NULL (a repetição pode ter acabado pois o índice não é mais >0). Como (C==T), isso é verdade. Logo, o bloco será executado, retornando uma condição de retorno ok. Vale a AS geral.

1. AE && (C==F) +B => AS

Como (C==F), pElem é NULL, e o elemento corrente da lista recebe o último da lista auxiliar. A função retorna uma condição de retorno fim lista. Vale a AS geral.

*Arg Seleção:*

1. AE && (C==T) + B => AS

Pela assertiva de entrada, o numero de elementos pode ser negativo. Como (C==T), ele é. Nesse caso B é executado. AS: índice aponta para o numElem.

1. AE && (C==F) +B => AS

O número de elementos é portanto 0, e valem as AS gerais.

*Arg. Repetição:*

AE: AI2, AS: AS

Ainv:

-existem dois conjuntos: a percorrer e já percorrido.

-indice aponta para elemento de a percorrer.

1. AE=> Ainv

Pela AE, índice aponta para o numElem e o conjunto já percorrido está vazio. Vale a Ainv.

1. AE && (C==F) => AS

(C==F) quando pElem==NULL. O primeiro ciclo não se completa (por causa do break) e vale a AS. Nesse caso, não teria outra situação para (C==F), pois a repetição se encontra dentro de um if que garante que o índice vai ser <0 (no primeiro ciclo).

1. AE && (C==T)+B => Ainv

Como (C==T), pElem != NULL. Logo, o índice é atualizado e o pElem agora é igual ao pElem->pAnt. Vale a Ainv.

1. Ainv && (C==T)+B => Ainv

Para que a Ainv continue valendo, B deve garantir que um elemento passe do conjunto a percorrer para o já percorrido e índice aponte para outro elemento de a percorrer.

1. Ainv && (C==F) => AS

No último ciclo, pElem==NULL ou i não é menor que 0. Nesse caso vale a AS.

1. Término

Como a cada ciclo, um elemento passa do conjunto a percorrer para já percorrido, e a quantidade de elementos desse conjunto e o tamanho da lista é finito, a repetição termina em um numero finito de passos.

*Arg.Seleção:*

1. AE && (C==T) + B => AS

Pela assertiva de entrada (AI5), o pElem pode ser diferente de NULL (a repetição pode ter acabado pois o índice não é mais <0). Como (C==T), isso é verdade. Logo, o bloco será executado, retornando uma condição de retorno ok. Vale a AS geral.

1. AE && (C==F) +B => AS

Como (C==F), pElem é NULL, e o elemento corrente da lista recebe o primeiro da lista auxiliar. A função retorna uma condição de retorno fim lista. Vale a AS geral.



