n prev
Up: Estructuras de datos II **Previous:** Dependencias de un programa

Implementación del TAD *Secuencia*

Implementar en ADA el TAD *Secuencia*, que responda al interfaz abstracto y a la semántica mostrada más adelante. La estructura de datos usada en la implementación deberá utilizar memoria dinámica y punteros, de forma que no haya ningún límite preestablecido en el código a la longitud de las secuencias. Debe prestarse especial atención a la corrección del código, tanto en lo que respecta a las especificaciones (incluyendo la complejidad esperada de las operaciones) como al manejo de memoria: al eliminar elementos de una secuencia deberá liberarse la memoria empleada anteriormente para almacenarlos.

Todas las operaciones que modifiquen o creen una secuencia deberán devolver una estructura de datos que cumpla el predicado *EsEstructura*, lo que en alguna operación puede implicar la copia de elementos de la secuencia.

TAD *TipoSecuencia*
USA *TipoElemento*
OPERACIONES:
  FUNCIÓN *Vacía* : *TipoSecuencia*
  FUNCIÓN *Longitud* : *TipoSecuencia* $\rightarrow \mathbb{N}$
  FUNCIÓN *Insertar* : *TipoSecuencia* $\times \mathbb{N} \times$ *TipoElemento* $\rightarrow$ *TipoSecuencia*
  FUNCIÓN *Reemplazar* : *TipoSecuencia* $\times \mathbb{N} \times$ *TipoElemento* $\rightarrow$ *TipoSecuencia*
  FUNCIÓN *Borrar* : *TipoSecuencia* $\times \mathbb{N} \rightarrow$ *TipoSecuencia*
  FUNCIÓN *Enésimo* : *TipoSecuencia* $\times \mathbb{N} \rightarrow$ *TipoElemento*
  FUNCIÓN *Buscar* : *TipoSecuencia* $\times$ *TipoElemento* $\rightarrow \mathbb{N}$
  FUNCIÓN *Subsecuencia* : *TipoSecuencia* $\times \mathbb{N} \times \mathbb{N} \rightarrow$ *TipoSecuencia*

SEMÁNTICA:

DOMINIO:

TIPO: $TipoSecuencia = Secuencia(TipoElemento)$

PRE: *cierto*
$Vacía$
POST: $resultado = <>$
COMPLEJIDAD: $O(1)$

PRE: *cierto*
$Longitud(secuencia)$
POST: $resultado = longitud(secuencia)$
COMPLEJIDAD: $O(longitud(secuencia))$

PRE: $0 < posición \wedge posición \leq longitud(secuencia) + 1$
$Insertar(secuencia, posición, elemento)$
POST: $(longitud(resultado) = longitud(secuencia) + 1 \wedge resultado_{posición} = elemento$
$\wedge \forall \alpha \in \{1..posición - 1\}.(resultado_{\alpha} = secuencia_{\alpha})$
$\wedge \forall \alpha \in \{posición..longitud(secuencia)\}.(resultado_{\alpha+1} = secuencia_{\alpha}))$
COMPLEJIDAD: $O(longitud(secuencia))$

PRE: $0 < posición \wedge posición \leq longitud(secuencia)$
$Reemplazar(secuencia, posición, elemento)$
POST: $(longitud(resultado) = longitud(secuencia) \wedge resultado_{posición} = elemento$
$\wedge \forall \alpha \in \{1..longitud(secuencia)\} - \{posición\}.resultado_\alpha = secuencia_\alpha)$
COMPLEJIDAD: $O(longitud(secuencia))$

PRE: $0 < posición \wedge posición \leq longitud(secuencia)$
$Borrar(secuencia, posición)$
POST: $longitud(resultado) = longitud(secuencia) - 1$
$\wedge \forall \alpha \in \{1..posición - 1\}.resultado_\alpha = secuencia_\alpha$
$\wedge \forall \alpha \in \{posición..longitud(secuencia)\}.resultado_{\alpha-1} = secuencia_\alpha)$
COMPLEJIDAD: $O(longitud(secuencia))$

PRE: $0 < posición \wedge posición \leq longitud(secuencia)$
$Enésimo(secuencia, posición)$
POST: $resultado = secuencia_{posición}$
COMPLEJIDAD: $O(posición)$

PRE: $cierto$
$Buscar(secuencia, elemento)$
POST: $(secuencia_{resultado} = elemento \wedge \forall \alpha \in \{1..resultado - 1\}.secuencia_\alpha \neq elemento)$
$\vee (resultado = 0 \wedge \forall \alpha \in \{1..longitud(secuencia)\}.secuencia_\alpha \neq elemento)$
COMPLEJIDAD: $O(longitud(secuencia))$

PRE: $0 < i \wedge i \leq j \wedge j \leq longitud(secuencia)$
$Subsecuencia(secuencia, i, j)$
POST: $resultado = secuencia(i, j)$
COMPLEJIDAD: $O(j)$

Manuel Carro
2001-04-04