Análisis de Algoritmos y Estructuras de Datos

Tema 7: Apéndice. Tipo Abstracto de Datos Lista Circular

Mª Teresa García Horcajadas José Fidel Argudo Argudo
Antonio García Domínguez Francisco Palomo Lozano

UCA
Universidad de Cádiz

Versión 1.0

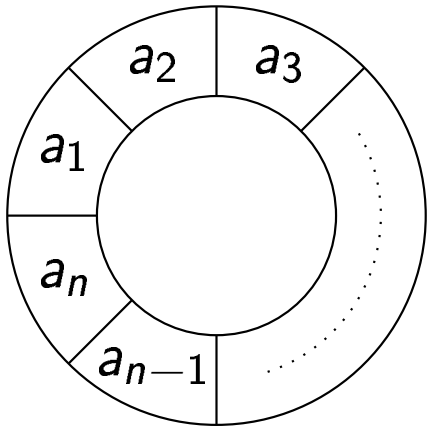
CC BY SA

Índice

Definición de Lista circular

Lista circular

- Secuencia de elementos sin extremos, es decir, en la que todos los elementos tienen un predecesor y un sucesor.
- Al igual que con las listas, es posible acceder, insertar y eliminar elementos en cualquier posición.



Definición de Lista circular

Lista circular (cont.)

- El comportamiento de una lista circular es muy parecido al de una lista, pero, al no tener extremos, no existen las posiciones inicial y final. En consecuencia, las operaciones del TAD serán las mismas que las del TAD Lista a excepción de *primera()* y *fin()*.
- Para recorrer la lista en un sentido o en el otro y tener acceso a cualquier elemento necesitaremos una operación que devuelva una posición desde la cual comenzar el recorrido. A esta operación que nos permite inicializar una variable de tipo posicion la llamaremos *inipos()*.

Especificación del TAD *Lista circular*

Definición:

Una lista circular es una secuencia de elementos de un mismo tipo en la que todos tienen un predecesor y un sucesor, es decir, es una secuencia sin extremos. Su longitud coincide con el número de elementos que la forman; si es 0, entonces la lista está vacía. Una lista circular de longitud n se puede representar de la forma

$$L = (a_1, a_2, \ldots, a_n, a_1)$$

donde repetimos a_1 después de a_n para indicar que el elemento que sigue a a_n es a_1 y el anterior a éste es a_n.

Definimos una posición como el lugar que ocupa un elemento en la lista. La constante POS_NULA denota una posición inexistente.

Especificación del TAD *Lista circular*

Operaciones:

ListaCir()

Postcondiciones: Crea y devuelve una lista circular vacía.

void insertar(const T& x, posicion p)

Precondiciones: $L = ()$ y p es irrelevante o bien,
$L = (a_1, a_2, \ldots, a_n, a_1)$ y $1 \leqslant p \leqslant n$

Postcondiciones: Si $L = ()$, entonces $L = (x, x)$ (lista circular con un único elemento); en caso contrario,

$$L = (a_1, \ldots, a_{p-1}, x, a_p, \ldots, a_n, a_1)$$

void eliminar(posicion p)

Precondiciones: $L = (a_1, a_2, \ldots, a_n, a_1)$
$1 \leqslant p \leqslant n$

Postcondiciones: $L = (a_1, \ldots, a_{p-1}, a_{p+1}, \ldots, a_n, a_1)$

Especificación del TAD *Lista circular*

const T& elemento(posicion p) const

T& elemento(posicion p)

Precondiciones: $L = (a_1, a_2, \ldots, a_n, a_1)$

$1 \leqslant p \leqslant n$

Postcondiciones: Devuelve a_p, el elemento que ocupa la posición p.

posicion buscar(const T& x) const

Postcondiciones: Devuelve la posición de una ocurrencia de x en la lista. Si x no pertenece a la lista, devuelve `POS_NULA`.

Especificación del TAD *Lista circular*

posicion inipos() const

Postcondiciones: Devuelve una posición indeterminada de la lista. Si la lista está vacía, devuelve `POS_NULA`. Esta operación se utilizará para inicializar una variable de tipo posicion.

posicion siguiente(posicion p) const

Precondiciones: $L = (a_1, a_2, \ldots, a_n, a_1)$
$1 \leqslant p \leqslant n$

Postcondiciones: Devuelve la posición siguiente a p.

posicion anterior(posicion p) const

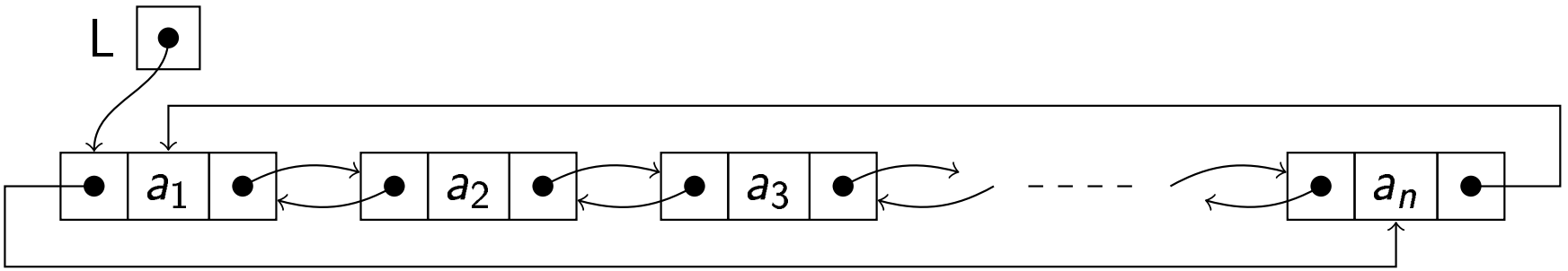
Precondiciones: $L = (a_1, a_2, \ldots, a_n, a_1)$
$1 \leqslant p \leqslant n$

Postcondiciones: Devuelve la posición anterior a p.

Implementación con una estructura doblemente enlazada

Estructura de datos

No se necesita un nodo cabecera, ya que cualquier nodo de la estructura está seguido (y por tanto precedido) de otro.



Representación de posiciones

Posición de un elemento Puntero al nodo anterior

inipos() Puntero a algún nodo de la estructura, por ejemplo, L.