**LISTAS ENLAZADAS**

**Introducción**

La lista enlazada es un TDA que nos permite almacenar datos de una forma organizada, al igual que los vectores pero, a diferencia de estos, esta estructura es dinámica, por lo que no tenemos que saber "a priori" los elementos que puede contener.

En una lista enlazada, cada elemento apunta al siguiente excepto el último que no tiene sucesor y el valor del enlace es null. Por ello los elementos son registros que contienen el dato a almacenar y un enlace al siguiente elemento. Los elementos de una lista, suelen recibir también el nombre de nodos de la lista.

struct lista {

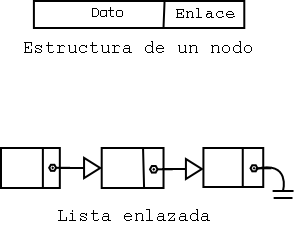
gint dato;

lista \*siguiente;

};

|  |  |
| --- | --- |
| 1 | Representa el dato a almacenar. Puede ser de cualquier tipo; en este ejemplo se trata de una lista de enteros. |
| 2 | Es un puntero al siguiente elemento de la lista; con este puntero enlazamos con el sucesor, de forma que podamos construir la lista. |

**Esquema de un nodo y una lista enlazada.**



Para que esta estructura sea un TDA lista enlazada, debe tener unos operadores asociados que permitan la manipulación de los datos que contiene. Los operadores básicos de una lista enlazada son:

* Insertar: inserta un nodo con dato x en la lista, pudiendo realizarse esta inserción al principio o final de la lista o bien en orden.
* Eliminar: elimina un nodo de la lista, puede ser según la posición o por el dato.
* Buscar: busca un elemento en la lista.
* Localizar: obtiene la posición del nodo en la lista.
* Vaciar: borra todos los elementos de la lista

Después de esta breve introducción, que sólo pretende servir como recordatorio, pasaremos a ver cómo es la estructura GSList que, junto con el conjunto de funciones que la acompañan, forman el TDA lista enlazada en GLib™.

**GSList**

La definición de la estructura GSList o, lo que es lo mismo, un nodo de la lista, está definido de la siguiente manera:

struct GSList {

gpointer data;

GSList \*next;

};

|  |  |
| --- | --- |
| 1 | Representa el dato a almacenar. Se utiliza un puntero genérico por lo que puede almacenar un puntero a cualquier tipo de dato o bien almacenar un entero utilizando las macros de conversión de tipos. |
| 2 | Se trata de un puntero al siguiente elemento de la lista. |

Las macros de conversión disponibles son las siguientes:

* GINT\_TO\_POINTER ()
* GPOINTER\_TO\_INT ()
* GUINT\_TO\_POINTER ()
* GPOINTER\_TO\_UINT ()

Más adelante, en esta misma sección, se verán ejemplos del uso de estas macros.

Las funciones que acompañan a la estructura GSList y que implementan los operadores básicos de las listas enlazadas, son las siguientes:

**Tabla 6. Operadores de inserción en listas enlazadas.**

| **Operador** | **Funciones asociadas a GSList.** |
| --- | --- |
| Insertar al principio. | GSList\* g\_slist\_prepend (GSList \*list, gpointer data) |
| Insertar al final. | GSList\* g\_slist\_append (GSList \*list, gpointer data) |
| Insertar en la posición indicada. | GSList\* g\_slist\_insert (GSList \*list, gpointer data, gint position) |
| Insertar en orden. | GSList\* g\_slist\_insert\_sorted (GSList \*list, gpointer data, GCompareFunc func) |

Las funciones de inserción al principio de la lista, g\_slist\_prepend, y al final, g\_slist\_append, son sencillas de usar. Sólo hay que pasarles como parámetros la lista donde queremos añadir el dato así como el dato a insertar y la función devuelve una lista con el nuevo dato insertado.

La función g\_slist\_insert inserta el dato en la posición indicada. Su uso también es sencillo como puede verse en el siguiente ejemplo.

**Ejemplo 14. Insertar un nuevo dato en una posición determinada.**

/\* obtiene el numero de nodos de la lista \*/

length = g\_slist\_length (list);

g\_print ("\nEscribe el nº de indice donde se insertara el dato (el indice maximo es %d): ", length);

scanf ("%d", &index);

/\* inserta el valor en la posicion indicada \*/

if (index < length) {

list = g\_slist\_insert (list, GINT\_TO\_POINTER (value), index);

print\_list (list);

}

En este ejemplo se utiliza la función g\_slist\_length para obtener el número de nodos que contiene la lista. A esta función hay que pasarle como parámetro la lista de la que se desea obtener el número de nodos y devuelve como resultado el número de nodos de ésta.

guint \* **g\_slist\_length**(*list*);  
GSList \* list ;

La función g\_slist\_insert\_sorted inserta los elementos a la lista de forma ordenada. Esta función utiliza el parámetro GCompareFunc para insertar el dato en la posición correcta.

GCompareFunc es una función que se utiliza para comparar dos valores y saber así cual de ellos hay que insertar primero. En los dos ejemplos que hay a continuación, se puede observar una función de tipo GCompareFunc y su uso para insertar datos en una lista en orden creciente.

**Ejemplo 15. Parámetro GCompareFunc para insertar en orden creciente.**

gint compare\_value1 (gconstpointer a, gconstpointer b) {

gint \*value1 = (gint \*) a;

gint \*value2 = (gint \*) b;

return value1 > value2;

}

**Ejemplo 16. Insertar elementos en orden creciente.**

gint values[] = {8, 14, 5, 12, 1, 27, 3, 13};

gint i;

/\* insertando valores en orden creciente \*/

for (i = 0; i < 8; i++) {

list = g\_slist\_insert\_sorted (list, GINT\_TO\_POINTER (values[i]),

compare\_value1);

}

**Tabla 7. Operadores de eliminación en listas enlazadas.**

| **Operador** | **Funciones asociadas a GSList.** |
| --- | --- |
| Eliminar un nodo. | GSList\* g\_slist\_remove (GSList \*list, gconstpointer data) |
| Eliminar nodos según un patrón. | GSList\* g\_slist\_remove\_all (GSList \*list, gconstpointer data) |

Las dos funciones expuestas para la eliminación de nodos, si bien tienen una definición prácticamente idéntica, el resultado obtenido es distinto. En el caso de g\_slist\_remove, se eliminará el nodo que contenga el valor *data*. Si hay varios nodos con el mismo valor, sólo se eliminará el primero. Si ningún nodo contiene ese valor, no se realiza ningún cambio en el GSList. En el caso de g\_slist\_remove\_all, se eliminan todos los nodos de la lista que contengan el valor *data* y nos devuelve la nueva lista resultante de la eliminación de los nodos.

**Ejemplo 17. Elimina un elemento de la lista.**

if (list2 != NULL) {

g\_print ("\nEl dato %d sera eliminado de la lista.\n", list2->data);

/\* eliminando un elemento de la lista \*/

g\_slist\_remove (list, list2->data);

}

**Tabla 8. Operadores de búsqueda en listas enlazadas.**

| **Operador** | **Funciones asociadas a GSList.** |
| --- | --- |
| Buscar un nodo según un valor. | GSList\* g\_slist\_find (GSList \*list, gconstpointer data) |
| Buscar un nodo según un criterio. | GSList\* g\_slist\_find\_custom (GSList \*list, gconstpointer data, GCompareFunc func) |
| Localizar el índice de un nodo. | GSList\* g\_slist\_index (GSList \*list, gconstpointer data) |
| Localizar la posición de un nodo. | GSList\* g\_slist\_position (GSList \*list, GSList \*llink) |
| Obtener el último nodo. | GSList\* g\_slist\_last (GSList \*list) |
| Obtener el siguiente nodo. | g\_slist\_next (slist) |
| Obtener un nodo por su posición. | GSList\* g\_slist\_nth (GSList \*list, guint n) |
| Obtener el dato de un nodo según su posición. | gpointer g\_slist\_nth\_data (GSList \*list, guint n) |

Todas estas funciones, a excepción de g\_slist\_nth\_data, devuelven un nodo de la lista o NULL si el elemento no existe. La función g\_slist\_nth\_data devuelve el valor del elemento según la posición que se le pasa como argumento en el parámetro *n* o NULL si la posición que se le pasa está más allá del final de la lista.

La función g\_slist\_next, es una macro que nos devuelve el siguiente nodo. Esta macro la podemos utilizar para recorrer la lista.

**Ejemplo 18. Función que imprime una lista.**

void print\_list (GSList \*list) {

gint i = 0;

while (list != NULL) {

g\_print ("Node %d content: %d.\n", i, list->data);

/\* apunta al siguiente nodo de la lista \*/

list = g\_slist\_next (list);

i++;

}

}

**Tabla 9. Operador para vaciar la lista.**

| **Operador** | **Funciones asociadas a GSList.** |
| --- | --- |
| Vacía la lista y libera la memoria usada. | void g\_slist\_free (GSList \*list) |

La función g\_slist\_free libera la memoria de la lista que se le pasa como parámetro.

Con estas funciones, quedan definidos los operadores básicos del TDA lista enlazada. GSList trae otras funciones además de los operadores básicos. Para más información sobre estas, está disponible el [manual de referencia de GLib™](http://developer.gnome.org/doc/API/2.0/glib/index.html).