Tema 4. Listas enlazadas

4.0. Contenido y documentación

https://s3-us-west-2.amazonaws.com/secure.notion-static.com/4565f03a-2852-43a7-b8fe-79f7688e7c04/U4_Listas.pdf

4.1. Listas

*Definición. Una **lista** es una colección de elementos organizados de tal manera que:*
- La inserción/extracción se puede hacer en cualquier posición.
- Todos los elementos salvo el último

4.1.1. Estructura de datos

Las listas enlazadas se construyen a partir de nodos, que tienen la siguiente estructura:

```
typedef struct _Node{
  void *info; /*contiene los datos del nodo*/
  struct _Node *next; /*puntero al siguiente nodo de la lista*/
} Node;
```

En el caso del último nodo de la lista, su campo `next` es `NULL`.

Por otra parte, la estructura de la lista es la siguiente:

```
typedef struct _List{
  Node *first; /*puntero al primer nodo de la lista*/
} List;
```

4.1.2. Primitivas de una lista

```
List list_new(); /*crea e inicializa una lista*/
void list_free(List l); /*libera la memoria asocidad a la lista*/
Boolean list_isEmpty(List l); /*devuelve TRUE si la lista está vacía, FALSE si no*/
Status list_pushFront(List l, Element e); /*inserta un elemento al principio de la lista*/
Status list_pushBack(List l, Element e); /*inserta un elemento al final de la lista*/
Element list_popFront(List l); /*extrae el primer elemento de la lista*/
Element list_popBack(List l); /*extrae el último elemento de la lista*/
```

Otras primitivas comunes son:

```
Int list_size(List l); /*devuelve el número de elementos de la lista*/
Status list_insert(List l, Position i, Element e); /*inserta una elemento en la posición i de la lista*/
Element list_extract(List l, Position i); /*extrae un elemento de la posición i de la lista*/
```

4.2. Implementaciones usando listas enlazadas

4.2.1. Pilas

Las listas enlazadas pueden usarse para implementar otros tipos de estructuras de datos, como puede ser una pila. Una pila implementada a partir de una lista enlazada tiene la siguiente estructura de datos:

```
typedef struct _Stack{
  List *pl; /*puntero a una lista enlazada*/
} Stack;
```

Una ventaja de usar esta implementación es que la pila nunca va a estar llena.

4.2.2. Colas

El caso de las colas es similar a las pilas, sustituimos la estructura de datos habitual de una cola por un puntero a una lista enlazada y nunca tendremos una cola llena.

```
typedef struct _Queue{
  List *pl; /*puntero a una lista enlazada*/
} Queue;
```

4.3. Tipos de listas

- Lista enlazada simple.

```
typedef struct _Node{
  void *info;
  struct _Node *next;
} Node;
```

```
typedef struct _List{
  Node *first;
} List;
```

- Lista enlazada simple con acceso posterior.

```
typedef struct _Node{
  void *info;
  struct _Node *next;
} Node;
```

```
typedef struct _List{
  Node *first;
  Node *last;
} List;
```

- Lista enlazada circular simple.

```
typedef struct _Node{
  void *info;
  struct _Node *next;
} Node;
```

```
typedef struct _List{
  Node *last;
} List;
```

- Lista doblemente enlazada

```
typedef struct _Node{
  void *info;
  struct _Node *next;
  struct _Node *prev;
} Node;
```

```
typedef struct _List{
  Node *first;
} List;
```

- Lista doblemente circular

```
typedef struct _Node{
  void *info;
  struct _Node *next;
} Node;
```

```
typedef struct _List{
  Node *first;
} List;
```

4.4. Listas circulares

En las listas **circulares** el campo `next` del último elemento de la lista apunta al primer elemento de la lista. Cuando la lista tiene solo un elemento, el campo de `next` de ese elemento se apunta a sí mismo.