**Ejercicio 1**

**Sintáctico y semántico**

Nuestro struct Pila está basado en dos componentes: un array de punteros a elementos y un top que nos indica el número de elementos que tiene la pila. Una lista al final puede ser utilizada como una pila si solo se utilizan ciertas primitivas, a la vez que también se puede usar como cola.

Tenemos que insertar un nuevo struct: el struct nodo, compuesto de un puntero a elemento y un puntero al nodo siguiente de nuestra lista. Nuestra lista solo necesitará un puntero al nodo inicial de la lista en la estructura, y, en nuestro caso, el tamaño.

El array estático de elementos se sustituye por un solo nodo inicial, enlazado al nodo siguiente, que estará enlazado al nodo siguiente, y así sucesivamente. Ahora, nuestro top será la primitiva list\_size, el equivalente a push un elemento será con list\_insertLast y el equivalente a pop un elemento de la pila será list\_extractLast. El resto de primitivas de la pila son triviles, como print o isFull, ya que su equivalente tiene incluso el mismo nombre en la lista.

**Uso de memoria**

Antes, usábamos memoria innecesaria al inicializar una pila, ya que reservábamos memoria para punteros a elementos de la pila que todavía no existían, y que además podían no existir nunca. Ahora, cada vez que queremos insertar un nuevo elemento, inicializamos un nodo que reserva su memoria en ese mismo instante.

Esto tiene dos ventajas: en primer lugar, no reservamos memoria que, en primera instancia no estamos utilizando. En segundo lugar, no tenemos que limitarnos al número de elementos establecidos en nuestro array. Nuestra nueva pila inicializada como lista ahora es capaz de almacenar nodos ilimitados.

**Ejercicio 2**

Con memoria dinámica:

Primero utilizamos la primitiva list\_size, que solo tiene que hacer un solo acceso a memoria gracias a que hemos implementado en la estructura un entero que representa el tamaño. Si llamamos al tamaño “n”, ahora nos toca hacer un bucle for que se ejecutará n/2 veces, y que extraerá todos los elementos con list\_extractFirst hasta llegar al elemento que queremos extraer. Una vez realizado este bucle, lo volveremos a realizar para volver a meter todos los elementos con list\_insertFirst, menos el que queríamos extraer. Accesos a memoria: n+1.

Con memoria estática:

Utilizamos la primitiva list\_size, como en la primera ocasión, solamente que ahora tenemos que acceder al array de direcciones de los elementos, liberar o extraer el elemento que queramos quitar de la lista y, mediante un bucle for, mover una posición en el array todas las direcciones que estén después de la dirección del elemento extraído. Accesos a memoria: n/2 +1.

Hacerlo con memoria estática nos ayuda más en este caso, hay que hacer prácticamente la mitad de accesos a memoria que con memoria dinámica, por lo que la memoria estática es mucho más eficiente, aunque reservemos memoria no necesaria.

**Ejercicio 3**

Ventajas del primer grafo:

Para toda primitiva que trate de ver si dos nodos están conectados, solo hay que comprobar con un if cierta posición de la matriz de adyacencia.

Desventajas del primer grafo:

Con memoria estática, se reserva memoria para 4096 nodos, lo que significa que la matriz de adyacencia contará con 16777216 enteros, de los cuales en el segundo grafo se prescinde de todos los ceros, que son la mayoría de los enteros.

En caso de insertar o extraer un nuevo nodo, tenemos que modificar una fila y una columna entera de la matriz de adyacencia mediante un bucle for.

Ventajas del segundo grafo:

Especialmente en este ejercicio se puede ahorrar una gran cantidad de memoria haciendo reservas de memoria dinámicas mediante listas, ya que la martriz de adyacencia aumenta de forma cuadrática su dimensión con respecto al número de nodos del grafo.

En caso de insertar o extraer un nodo, solo tenemos que realizar un list\_insertFirst en in\_connections y out\_connections según las conexiones que tenga el nodo. En este caso nos ahorramos todos los ceros de la matriz de adyacencia.

Desventajas del segundo grafo:

Para ver si un nodo está conectado a otro, hay que ver la lista de nodos conectados de uno de los dos nodos, y es necesario un bucle for para comprobar si el otro nodo está en la lista.