Lista enlazada

Se incluye en el sitio de descargas un ejemplo de uso de iteradores externos.

Estas son las primitivas de listas que tienen que implementar.

En esta entrega les agregamos el requerimiento de escribir la documentación completa de las primitivas, con sus correspondientes pre y post condiciones, para esto pueden usar de muestra los archivos .h que ya utilizaron para la implementación de pilas y colas.

Primitivas de la lista

```
typedef struct lista lista_t;

lista_t *lista_crear(void);
bool lista_esta_vacia(const lista_t *lista);
bool lista_insertar_primero(lista_t *lista, void *dato);
bool lista_insertar_ultimo(lista_t *lista, void *dato);
void *lista_borrar_primero(lista_t *lista);
void *lista_ver_primero(const lista_t *lista);
void *lista_ver_ultimo(const lista_t* lista);
size_t lista_largo(const lista_t *lista);
void lista_destruir(lista_t *lista, void (*destruir_dato)(void *));
```

Primitiva del iterador interno

```
void lista_iterar(lista_t *lista, bool visitar(void *dato, void *extra), void *extra);
```

Primitivas del iterador externo

```
typedef struct lista_iter lista_iter_t;

lista_iter_t *lista_iter_crear(lista_t *lista);
bool lista_iter_avanzar(lista_iter_t *iter);
void *lista_iter_ver_actual(const lista_iter_t *iter);
bool lista_iter_al_final(const lista_iter_t *iter);
void lista_iter_destruir(lista_iter_t *iter);
bool lista_iter_insertar(lista_iter_t *iter, void *dato);
void *lista_iter_borrar(lista_iter_t *iter);
```

Función de pruebas

```
void pruebas_lista_estudiante(void);
```

Aclaración: mantener el nombre de la estructura como `struct lista` para la lista, y `struct lista_iter` para el iterador, puesto que el corrector automático tiene configurados dichos nombres en el `lista.h` de la cátedra.

Considerar que todas las primitivas (exceptuando `lista_destruir` y `lista_iterar`) deben funcionar en tiempo constante.

Las pruebas deben incluir los casos básicos de TDA similares a los contemplados para la pila y la cola, y adicionalmente debe verificar los siguientes casos del iterador externo:

1. Al insertar un elemento en la posición en la que se crea el iterador, efectivamente se inserta al principio.
2. Insertar un elemento cuando el iterador está al final efectivamente es equivalente a insertar al final.
3. Insertar un elemento en el medio se hace en la posición correcta.
4. Al remover el elemento cuando se crea el iterador, cambia el primer elemento de la lista.
5. Remover el último elemento con el iterador cambia el último de la lista.
6. Verificar que al remover un elemento del medio, este no está.
7. Otros casos borde que pueden encontrarse al utilizar el iterador externo. Y los casos con / sin corte del iterador interno.

Al igual que en los casos anteriores, deberán entregar el código en papel, con el nombre y padrón y el nombre del ayudante correspondiente, imprimiendo los archivos `lista.h`, `lista.c` y `pruebas_lista.c`.

Además, deben subir el código a la página de entregas de la materia, con el código completo.

No olviden revisar las preguntas frecuentes de la lista enlazada

---

Bibliografia recomendada

- Weiss, Mark Allen, “Data Structures and Algorithm Analysis”: *3.2. The List ADT*.
- Cormen, Thomas H. “Introduction to Algorithms”: *10.2. Linked lists*.