

Semana 6: TDA lista Laboratorio

Semestre 2 - 2023

Análisis de Algoritmos y Estructura de Datos

Departamento de Ingeniería Informática

TDA Lista Enlazada

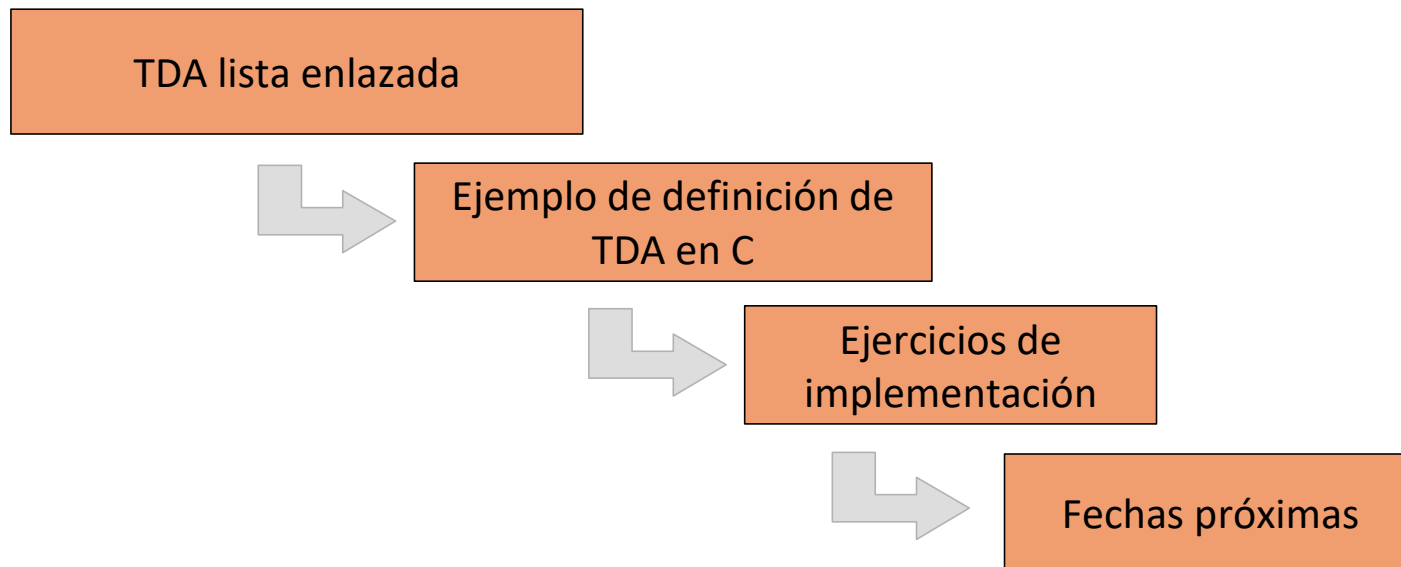
- **Contenidos:**

- Estructura de datos de TDA lista enlazada
- Operaciones de TDA lista enlazada

- **Objetivos:**

- Implementar TDA lista enlazada

Ruta de trabajo



Especificación e implementación de TDA lista enlazada

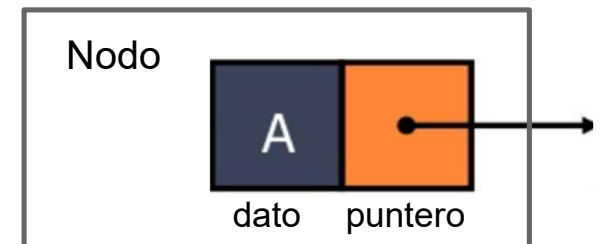




Especificación de TDA lista enlazada

- **Estructura de datos:**

- Una lista enlazada (LE) es una secuencia de nodos conectados
- A una lista con 0 nodos se le conoce como **lista vacía**
- Cada nodo contiene:
 - . Una parte de datos (cualquier tipo)
 - . Un puntero al siguiente nodo de la lista
- **Cabeza**: puntero al primer nodo
- El último nodo apunta a **nulo**

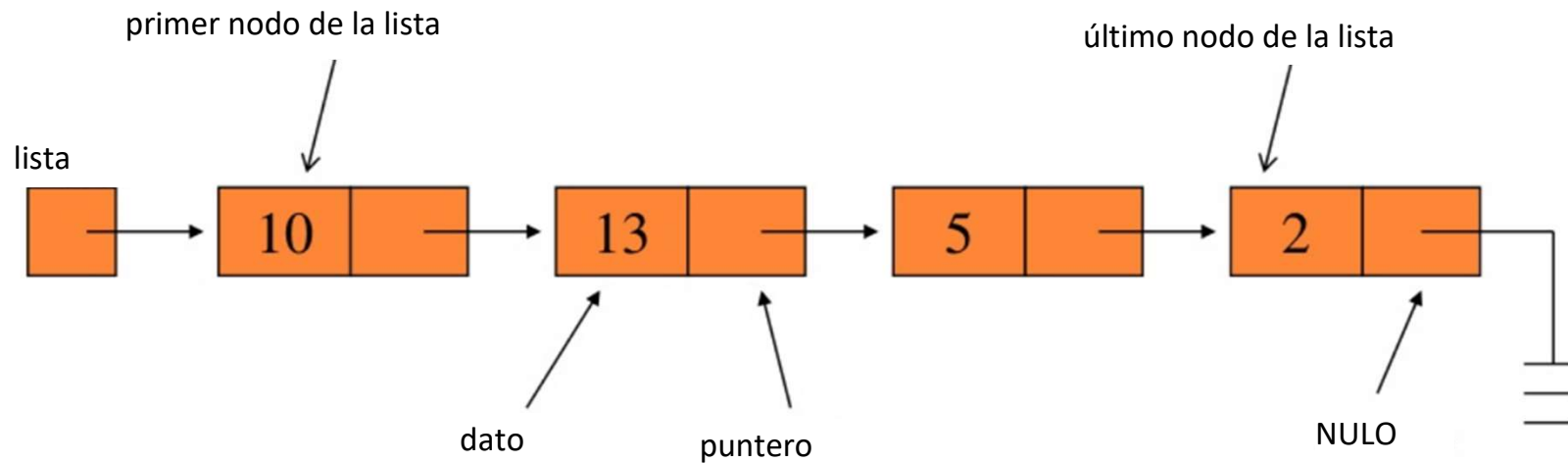


Especificación de TDA lista enlazada

- **Operaciones:**

- **esListaVacía(L)**: determina si lista L está vacía o no
- **insertarNodo(L,dato)**: inserta nodo con *dato* en lista L
- **eliminarNodo(L,dato)**: elimina nodo con *dato* de lista L
- **buscarNodo(L,dato)**: busca nodo con *dato* en lista L
- **recorrerLista(L)**: muestra contenido de cada nodo de lista L

Especificación de TDA lista enlazada

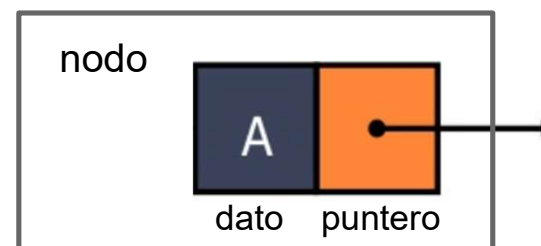


Lista enlazada de enteros

Implementación de estructura de datos de TDA lista enlazada

- La estructura de datos que representa un nodo de una lista enlazada simple es la siguiente:

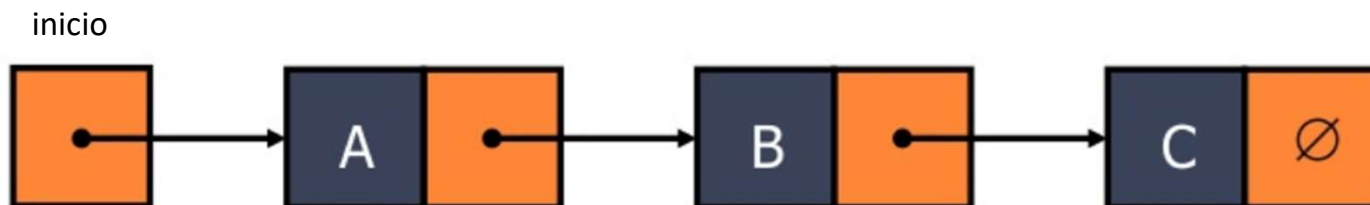
```
typedef struct nodo{  
    int dato;  
    struct nodo *siguiente;  
}nodo;
```



Implementación de estructura de datos de TDA lista enlazada

- La estructura de datos que representa una lista es la siguiente:

```
typedef struct{  
    nodo *inicio;  
}lista;
```



Implementación de operaciones de TDA lista enlazada

```
lista* nueva_lista(){  
    lista *l = (lista*)malloc(sizeof(lista));  
    l->inicio = NULL;  
    return l;  
}  
  
int es_lista_vacia(lista *l){  
    if (l->inicio == NULL)  
        return 1;  
    else  
        return 0;  
}
```

Implementación de operaciones de TDA lista enlazada

```
void inserta_inicio(lista *l, int d){
    nodo *nuevo_nodo = (nodo*)malloc(sizeof(nodo));
    nuevo_nodo->dato = d;
    nuevo_nodo->siguiente = l->inicio;
    l->inicio = nuevo_nodo;
}

void elimina_inicio(lista *l){
    nodo *aux;
    if(!es_lista_vacia(l)){
        aux = l->inicio;
        l->inicio = l->inicio->siguiente;
        free(aux);
    }
}
```

Implementación de operaciones de TDA lista enlazada

```
void imprime_lista(lista *l){
    printf("Lista: ");
    if (!es_lista_vacia(l)){
        nodo *aux = l->inicio;
        while (aux != NULL){
            printf("%d ", aux->dato);
            aux = aux->siguiente;
        }
        printf("\n");
    }
    else
        printf("%c\n", 157);
}

void libera_lista(lista *l){
    while (!es_lista_vacia(l)){
        elimina_inicio(l);
    }
    free(l);
}
```

Implementación de operaciones de TDA lista enlazada

- Usando la misma idea, se pueden implementar funciones básicas para trabajar con una lista enlazada simple:
 - `void inserta_final(lista *l, int d)`
 - `void inserta_despues(lista *l, int d, int d_anterior)`
 - `void elimina_final(lista *l)`
 - `void elimina_dato(lista *l, int dato)`
 - `int cuenta_nodos(lista *l)`
 - `int busca_dato(lista *l, int d)`
 - `nodo* retorna_nodo(lista* l, int pos)`

Actividades de implementación



Actividad 1

1. Compilar y ejecutar **lab06_lista_simple.c**
2. Experimentar con las funciones de **TDAlista.h**, haciendo llamadas desde la función *main()* en **lab06_lista_simple.c**:
 - Crear una lista
 - Insertar al inicio nodos en el siguiente orden: 5, 7, 4, 2
 - Mostrar la lista resultante
 - Eliminar nodo al inicio, 2 veces
 - Mostrar la lista resultante
 - Insertar al inicio nodo con valor 3
 - Mostrar la lista resultante

Actividad 2

1. Implementar una función que devuelva el número de elementos de *lista*. La función debe devolver 0 en caso de que *lista* esté vacía.

```
int cuenta_nodos(lista *l);
```

2. Evaluar la función creada, generando secuencia de llamadas desde la función *main()* en **lab06_lista_simple.c** para mostrar cuántos elementos tiene la siguiente lista



Actividad 2

```
int cuenta_nodos(lista *l);
```

```
cuenta_nodos(lista *l): num  
    num contador ← 0  
    si (no es_lista_vacia(l)) entonces  
        nodo *ptr ← l→inicio  
        mientras ptr <> NULL hacer  
            contador ← contador + 1  
            ptr ← ptr→puntero  
    return contador
```



Actividad 3

1. Implementar una función que busque un dato en una lista enlazada. La función debe devolver 1 en caso de encontrar el *dato* y 0 en caso contrario:

```
int busca_dato(lista *l, int d);
```

2. Evaluar la función creada, generando secuencia de llamadas desde la función main() en **lab06_lista_simple.c** para buscar los datos 6 y 14 en la siguiente lista:



Actividad 3

```
int busca_dato(lista *l, int d);
```

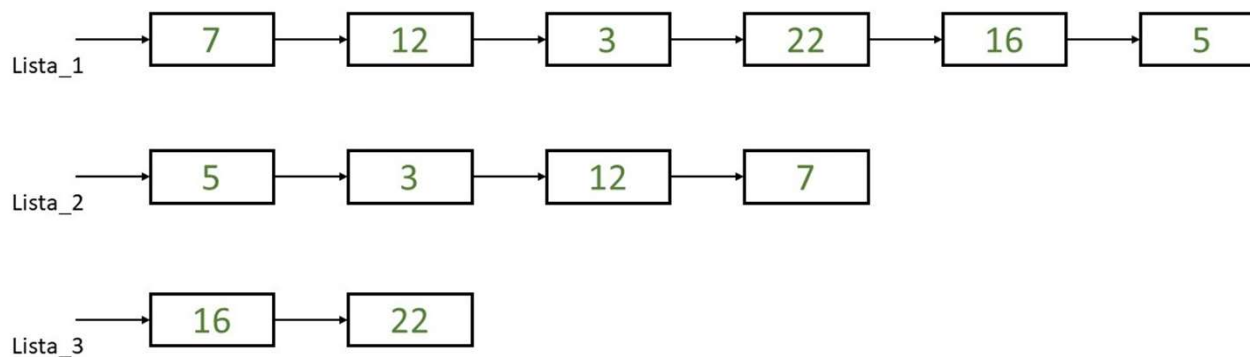
```
busca_dato(lista *l, num d): num  
    si (no es_lista_vacia(l)) entonces  
        nodo *ptr ← l→inicio  
        mientras ptr <> NULL hacer  
            si ptr→dato = d entonces  
                return 1  
            ptr ← ptr→puntero  
    return 0
```

Actividad 4

Implementar la siguiente función :

1. A partir de una lista enlazada de números enteros y de una posición x en la lista, generar dos listas, una con los números menores o iguales que el número en la posición indicada y otra con los números mayores.
2. Evaluar la función creada, generando secuencia de llamadas desde función *main()* en **lab06_lista_simple.c**.

Por ejemplo, para la lista_1 y la posición 2 (elemento con valor igual a 12):



Actividad 5 asíncrona

Usando **TDAlista.h**, **TDAlista_implementacion.c** y **lab06_lista_simple.c**, implementar las siguientes operaciones:

- `void inserta_final(lista *l, int d)`
- `void inserta_despues(lista *l, int d, int d_anterior)`
- `void elimina_final(lista *l)`
- `void elimina_dato(lista *l, int dato)`
- `nodo* retorna_nodo(lista* l, int pos)`

Evaluar todas las funciones creadas, generando secuencia de llamadas desde función *main()* en **lab06_lista_simple.c**

Entrega de actividad de laboratorio

- Entrega obligatoria
- Subir SOLO actividades 3 y 4 de esta sesión en buzón de Campus Virtual, en único archivo **s6_coordinacion_apellido_nombre.zip**
- Se espera TDAlista.h, TDAlista_implementacion.c y lab06_lista_simple.c (modificados para responder a actividades 3 y 4) comprimidos en archivo .zip
- Plazo: antes del **próximo** laboratorio de cada coordinación