DEPARTAMENTO DE
INGENIERÍA
INFORMÁTICA
UNIVERSIDAD DE SANTIAGO DE CHILE

Análisis de Algoritmos y Estructura de Datos

TDA lista enlazada

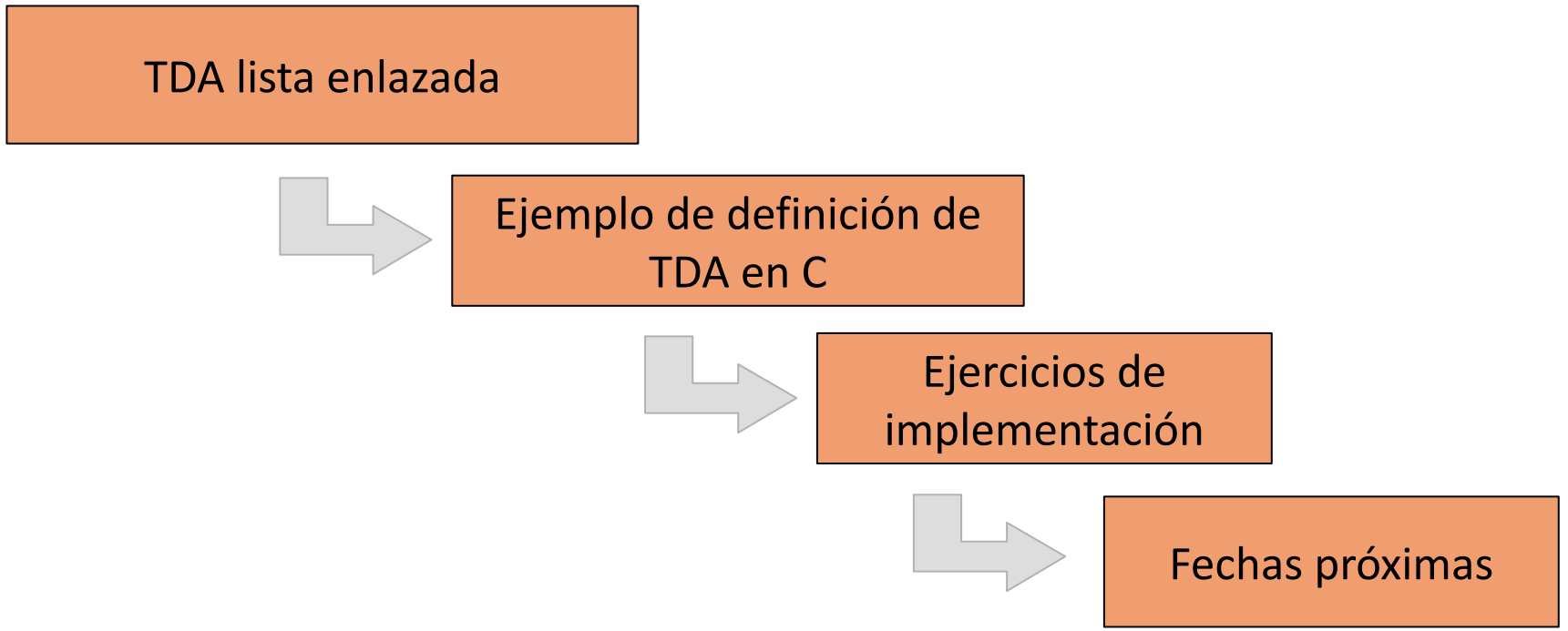
Prof. Violeta Chang C

Semestre 1 – 2022

TDA Lista Enlazada

- **Contenidos**:
  - Estructura de datos de TDA lista enlazada
  - Operaciones de TDA lista enlazada
- **Objetivos**:
  - Implementar TDA lista enlazada

Ruta de trabajo

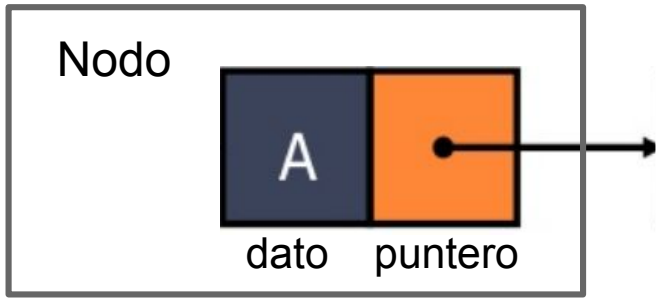


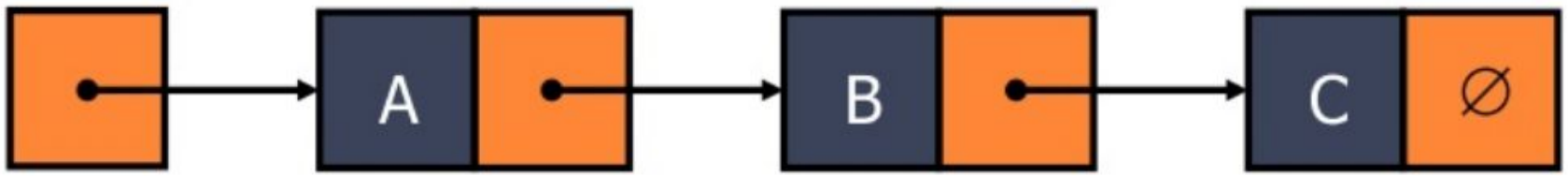
DEPARTAMENTO DE
INGENIERÍA
INFORMÁTICA
UNIVERSIDAD DE SANTIAGO DE CHILE

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



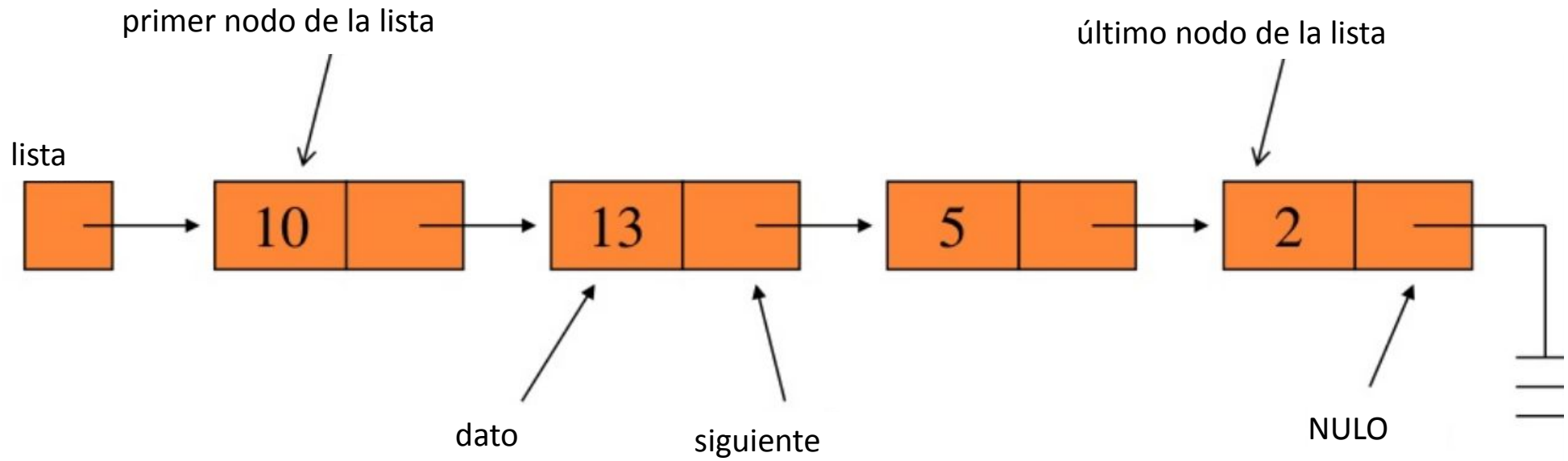


Khuram Shahzad, Data structures and algorithms

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



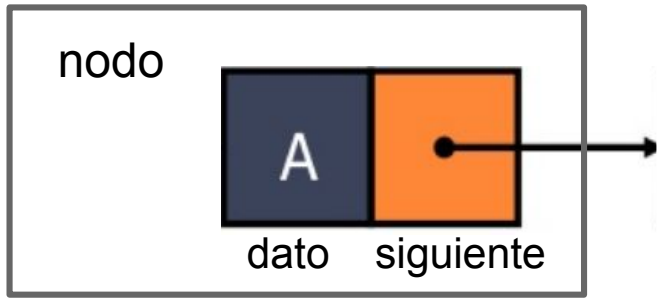
Lista enlazada de enteros

Khuram Shahzad, Data structures and algorithms

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:

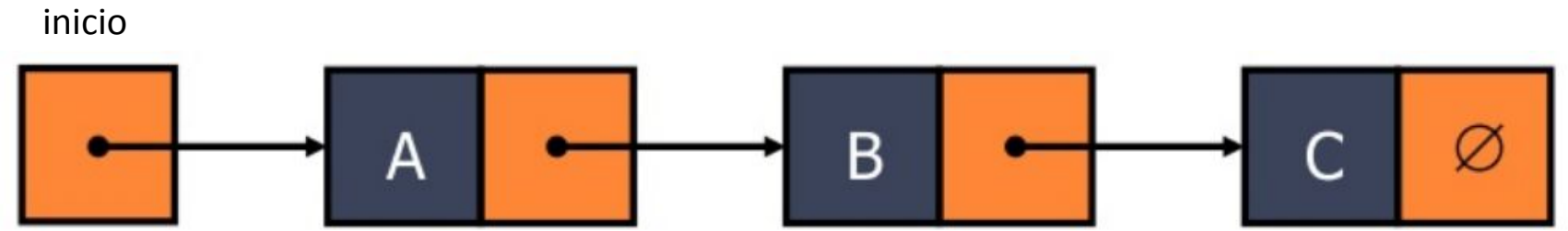
```
typedef struct nodoGenerico
{
  int dato;
  struct nodoGenerico* siguiente;
}nodo;
```



Implementación de estructura de datos de TDA lista enlazada

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

```
typedef struct listaGenerica
{
  nodo* inicio;
}TDAlista;
```



Implementación de operaciones de TDA lista enlazada

```
TDAlista* crearListaVacia()
{
  TDAlista* lista=(TDAlista*)malloc(sizeof(TDAlista));
  lista->inicio=NULL;
  return lista;
}

int esListaVacia(TDAlista* lista)
{
  if (lista->inicio == NULL)
    return 1;
  else
    return 0;
}
```

Implementación de operaciones de TDA lista enlazada

```
void insertarInicio(TDAlista* lista, int dato)
{
  nodo* nuevo=(nodo*)malloc(sizeof(nodo));
  nuevo->dato=dato;
  nuevo->siguiente = lista->inicio;
  lista->inicio=nuevo;
}

void eliminarInicio(TDAlista* lista)
{
  nodo* auxiliar;
  if(!esListaVacia(lista))
  {
    auxiliar=lista->inicio;
    lista->inicio=lista->inicio->siguiente;
    free(auxiliar);
  }
}
```

Implementación de operaciones de TDA lista enlazada

```
void recorrerLista(TDAlista* lista)
{
  if (!esListaVacia(lista))
  {
    nodo* auxiliar=lista->inicio;
    while (auxiliar!=NULL)
    {
      printf("%d ",auxiliar->dato);
      auxiliar=auxiliar->siguiente;
    }
    printf("\n");
  }
  else
    printf("La lista está vacía\n");
}
```

Implementación de operaciones de TDA lista enlazada

- Usando la misma idea, se pueden implementar funciones básicas para trabajar con una lista enlazada simple:
  - **void insertarNodoFinal(TDAlista* lista, int dato)**
  - **void insertarNodoDespues(TDAlista* lista, int dato, int datoAnterior)**
  - **void eliminarFinal(TDAlista* lista)**
  - **void eliminarNodoDato(TDAlista* lista, int dato)**
  - **int obtenerNumeroNodos(TDAlista* lista)**
  - **int buscarDato(TDAlista* lista, int dato)**
  - **nodo* obtenerNodo(TDAlista* lista, int posición)**
  - **void liberarLista(TDAlista* lista)**

Actividades de implementación

Actividad de implementación 1

1. Compilar y ejecutar **lab06-listaSimple.c**
2. Experimentar con las funciones implementadas en **TDAlista.h** haciendo llamadas desde función *main()* en **lab06-listaSimple.c**:
    - Insertar al inicio nodos en el siguiente orden: 5, 7, 4, 2
    - Recorrer la lista resultante
    - Eliminar nodo al inicio, 2 veces
    - Recorrer la lista resultante
    - Insertar al inicio nodo con valor 3
    - Recorrer la lista resultante

Actividad de implementación 2

INDICACIONES:
Primera función: implementación en conjunto, cada quien replica en su computador
Siguientes funciones: implementación individual

1. Implementar las siguientes funciones en **TDAlista.h**:

```
void liberarLista(TDAlista* lista);
int buscarDato(TDAlista* lista, int dato);
int obtenerNumeroNodos(TDAlista* lista);
```

2. Evaluar todas las funciones creadas, generando secuencia de llamadas desde función *main()* en lab06-listaSimple.c

Actividad de implementación 3

1. Usando **TDAlista.h** y **lab06-listaSimple.c,** implementar las siguientes funciones:

```
void insertarNodoFinal(TDAlista* lista, int dato);
void insertarNodoDespues(TDAlista* lista, int dato, int datoAnterior);
void eliminarFinal(TDAlista* lista);
void eliminarDato(TDAlista* lista, int dato);
nodo* obtenerNodo(TDAlista* lista, int posicion);
```

2. Evaluar todas las funciones creadas, generando secuencia de llamadas desde función *main()* en **lab06-listaSimple.c**

Entrega de actividad

1. Comprimir lab06-listaSimple.c y TDAlista.h para generar archivo **S6_<*coordinación*>_<*apellido*>_<*nombre*>.zip**

2. Subir el archivo ***.zip*** al buzón de entrega en uVirtual, pestaña s6-lab hasta hoy **viernes 22 de abril a las 23:59 hrs**.

Próximas fechas...

- Resumen de la semana:
  - *TDA lista enlazada*
  - *Listas enlazadas especiales*

- Próxima semana:
  - *TDA pila*
  - *TDA cola*
  - **Simulacro 2**: miércoles 27/abril

U2 - S6