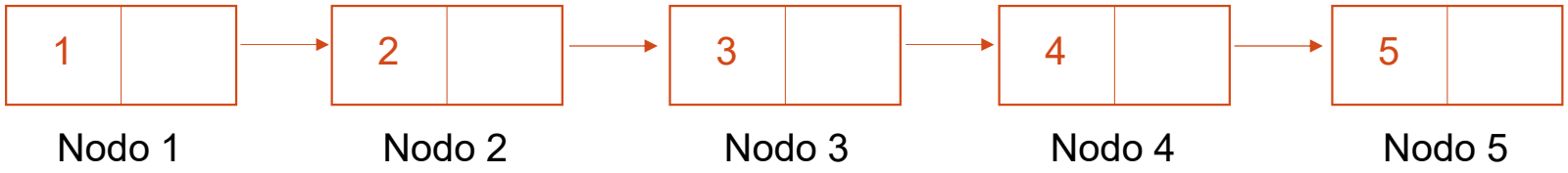
Listas Enlazadas Simples

- Características
- Diseño de la estructura de datos
- Operaciones básicas
  - Adicionar un elemento al inicio de la lista.
  - Recorrido completo (Traversal)
  - Buscar un dato en la lista.
  - Eliminar el primer elemento de la lista.
  - Eliminar un dato en la lista.

Características

- Estructura de Datos
- Unidimensional (Lista – Vector)
- Dinámica (Dimensión)
- Flexible
- Eficiente en uso de memoria
- Base para Árboles y Grafos

Diseño



Clase Nodo	Clase Lista
Almacena la información de la lista, elemento por elemento. Cada nodo conecta al siguiente elemento a través de una referencia.	Almacena la instancia del primer nodo. Implementa las operaciones requeridas para la gestión de los nodos (Adición, Búsqueda, Eliminación, etc.)

UNIVERSIDAD DE LA COSTA
CUC
1970

Adicionar al inicio

UNIVERSIDAD DE LA COSTA
CUC
1970

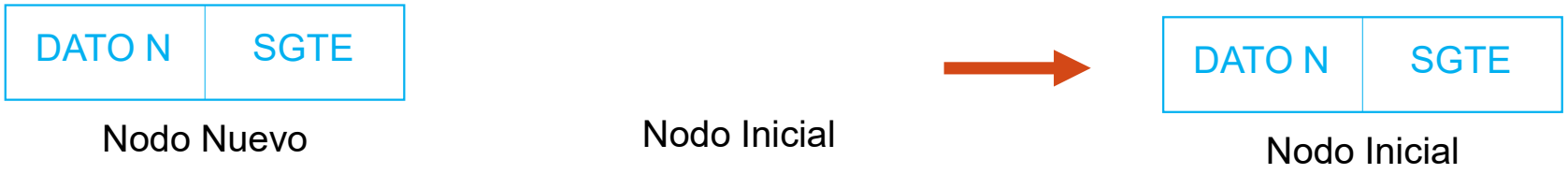
Adicionar al Inicio (1)

1. Crear el nuevo nodo con la información
2. Si la lista está vacía, el nuevo nodo se vuelve nodo inicial.

(1)

Nodo Nuevo

(2)



Nodo Nuevo

Nodo Inicial

Nodo Inicial

UNIVERSIDAD DE LA COSTA
1970

Adicionar al Inicio (2)

3. Si la lista tiene elementos:
   a) El nuevo nodo se conecta al nodo inicial
   b) El nuevo nodo se vuelve el nodo inicial.

(a)

(b)

UNIVERSIDAD DE LA COSTA
CUC
1970

Adicionar al Inicio

```
def adicionarAlInicio(self, dato_nuevo):
    nodoNuevo = NodoSimple(dato_nuevo)
    if self.estaVacia():
        self.nodoInicial = nodoNuevo
    else:
        nodoNuevo.siguiente = self.nodoInicial
        self.nodoInicial = nodoNuevo
```

UNIVERSIDAD DE LA COSTA
CUC
1970

Recorrido Completo (Traversal)

1. Si la lista está vacía, no hay recorrido.
2. Si la lista tiene elementos:
   a) Se ubica una variable nodoActual en el nodoInicial.
   b) Se almacena el dato del nodoActual en el recorrido.
   c) Se asigna el valor del nodoActual hacia el nodo de la derecha.
   d) Se repite el proceso hasta que el nodoActual sea nulo (fin de la lista).

UNIVERSIDAD DE LA COSTA 1970

Recorrido Completo (Traversal)

```
def __str__(self):
    recorrido = ""
    nodoActual = self.nodoInicial
    while nodoActual != None:
        recorrido += str(nodoActual.dato) + " "
        nodoActual = nodoActual.siguiente
    return recorrido
```

UNIVERSIDAD DE LA COSTA
CUC
1970

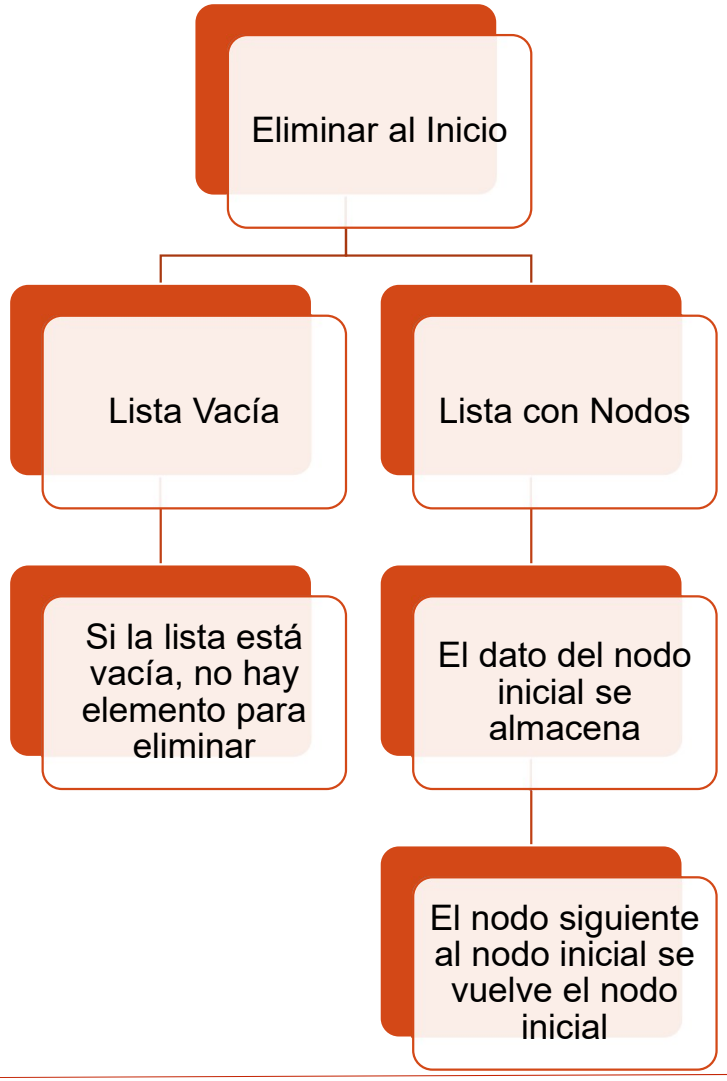
Buscar por dato

1. Si la lista está vacía, la búsqueda es negativa.
2. Si la lista tiene elementos:
   a) Se ubica una variable nodoActual en el nodoInicial.
   b) Se compara el dato del nodoActual con el dato buscado
      1) Si se encuentra, se retorna como positiva la búsqueda.
      2) Si no se encuentra, se mueve el nodoActual al siguiente y se repite el proceso
      3) Si se finaliza la lista, y no pudo encontrarse, la búsqueda es negativa.

Buscar por dato

```
def buscar(self, dato_buscar):
    if self.estaVacia():
        return False
    else:
        nodoActual = self.nodoInicial
        while nodoActual != None:
            if nodoActual.dato == dato_buscar:
                return True
            nodoActual = nodoActual.siguiente
        return False
```

UNIVERSIDAD DE LA COSTA
CUC
1970



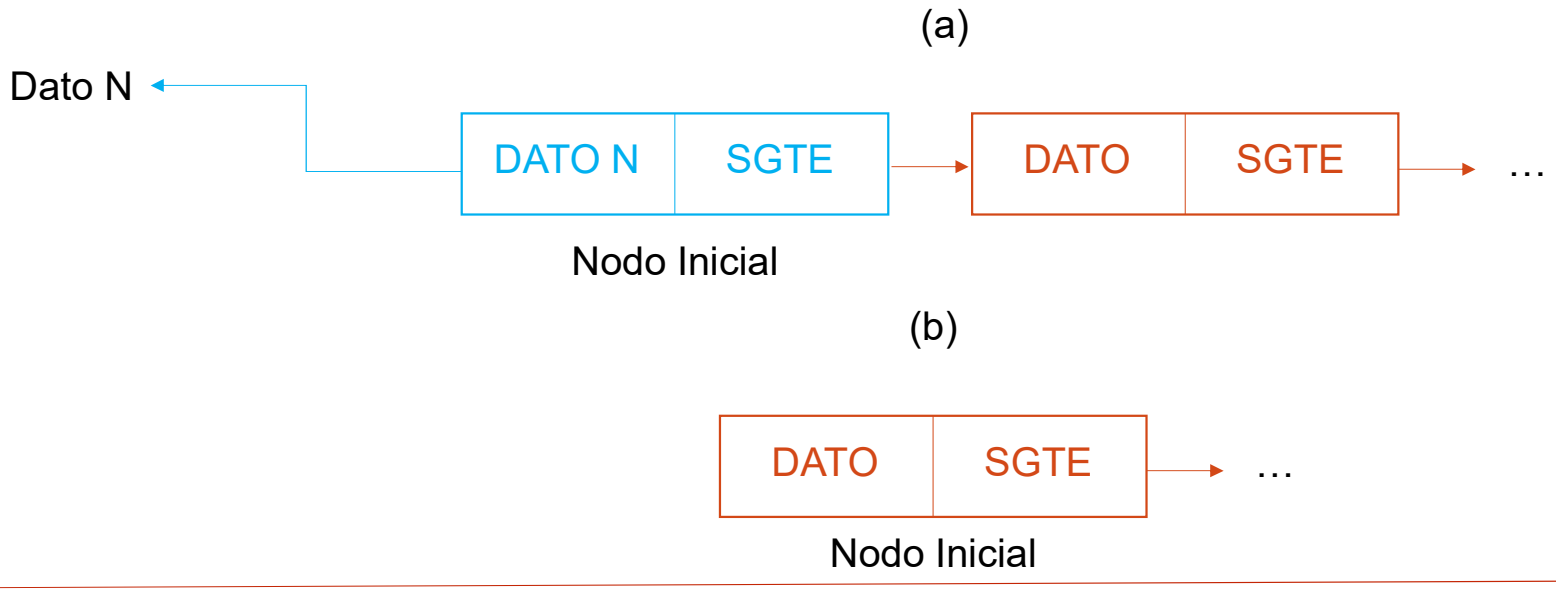
Eliminar al inicio

UNIVERSIDAD DE LA COSTA
CUC
1970

Eliminar al Inicio (1)

Si la lista tiene elementos:

a) El dato del nodo inicial se almacena.
b) El nodo siguiente al nodo inicial se vuelve el nodo inicial.



UNIVERSIDAD DE LA COSTA
CUC
1970

Eliminar al Inicio

```
def eliminarAlInicio(self):
    if self.estaVacia():
        return None
    else:
        dato = self.nodoInicial.dato
        self.nodoInicial = self.nodoInicial.siguiente
        return dato
```

UNIVERSIDAD DE LA COSTA
CUC
1970

Eliminar por información

1. Si la lista está vacía, no hay elemento para eliminar.
2. Si la lista tiene elementos:

2.1 Si el elemento a eliminar es el nodo inicial, se realiza eliminación al inicio.

2.2. Si el elemento a eliminar no es el nodo inicial, se procede a buscar el elemento en la lista, lo cual puede llevar a los siguientes casos:

2.2.1. El elemento no se encuentra en la lista, no hay elemento para eliminar.

2.2.2. El elemento se encuentra en la lista, y es el último elemento, se realiza eliminación por el final.

2.2.3. El elemento se encuentra en la lista, y es un elemento intermedio, se conecta el nodo anterior al dato, con el siguiente del dato.

UNIVERSIDAD DE LA COSTA
CUC
1970

Eliminar por información (1)

2.1 Si el elemento a eliminar es el nodo inicial, se realiza eliminación al inicio.

Ejemplo: La lista es: [A, X, Y, 20] y el elemento es A. La lista resultante es [X, Y, 20].

Eliminar por información (2)

2.2.1. El elemento no se encuentra en la lista, no hay elemento para eliminar.

Ejemplo: La lista es: [A, X, Y, 20] y el elemento es Z. La lista no se modifica.

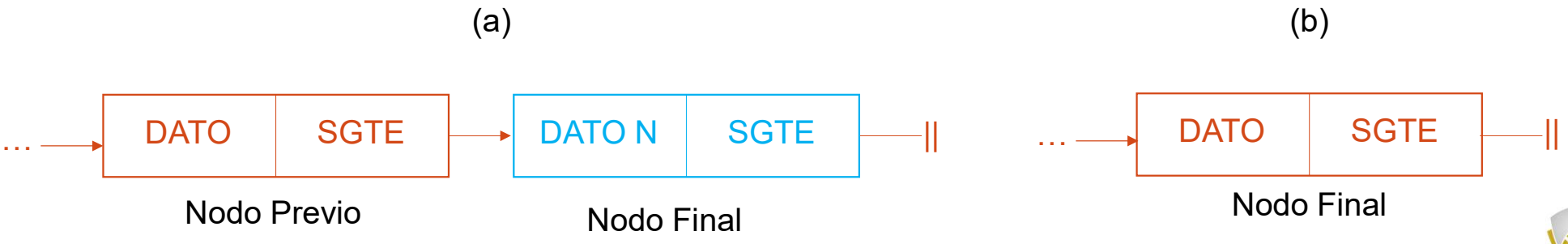
(a)

UNIVERSIDAD DE LA COSTA
CUC
1970

Eliminar por información (3)

2.2.2. El elemento se encuentra en la lista, y es el último elemento, se realiza eliminación por el final.

Ejemplo: La lista es: [A, X, Y, 20] y el elemento es 20. La lista resultante es [A, X, Y];



UNIVERSIDAD DE LA COSTA
CUC
1970

Eliminar por información (4)

2.2.3. El elemento se encuentra en la lista, y es un elemento intermedio, se conecta el nodo anterior al dato, con el siguiente del dato.

Ejemplo: La lista es: [A, X, Y, 20] y el elemento es X. La lista resultante es [A, Y, 20];

(a)

UNIVERSIDAD DE LA COSTA
CUC
1970

Eliminar por información (5)

2.2.3. El elemento se encuentra en la lista, y es un elemento intermedio, se conecta el nodo anterior al dato, con el siguiente del dato.

(a)

(b)

UNIVERSIDAD DE LA COSTA 1970

Eliminar por información

```
def eliminarInfo(self, dato_eliminar):
    if self.estaVacia():
        return False
    if self.nodoInicial.dato == dato_eliminar:
        self.eliminarAlInicio()
        return True
    nodoPrevio = None
    nodoActual = self.nodoInicial
    while nodoActual != None and nodoActual.dato != dato_eliminar:
        nodoPrevio = nodoActual
        nodoActual = nodoActual.siguiente
    if nodoActual == None:
        return False
    if nodoActual.siguiente == None:
        nodoPrevio.siguiente = None
    else:
        nodoPrevio.siguiente = nodoActual.siguiente
    return True
```

UNIVERSIDAD DE LA COSTA
CUC
1970

Preguntas

UNIVERSIDAD DE LA COSTA
CUC
1970

Taller de Diseño

Diseñar e implementar los siguientes programas en VS Code y Python, usando el diseño de objetos indicado en la cátedra:

Adicionar al diseño de la clase Lista Simple las siguientes funcionalidades:

- Adicionar un elemento al final de la lista.
- Eliminar el último de la lista.
- Indicar el número de apariciones de un dato en la lista.
- Indicar el elemento en una posición indicada de la lista, si existe.
- Indicar el último elemento de la lista.
- Eliminar una posición indicada de la lista, si existe.

UNIVERSIDAD DE LA COSTA
CUC
1970