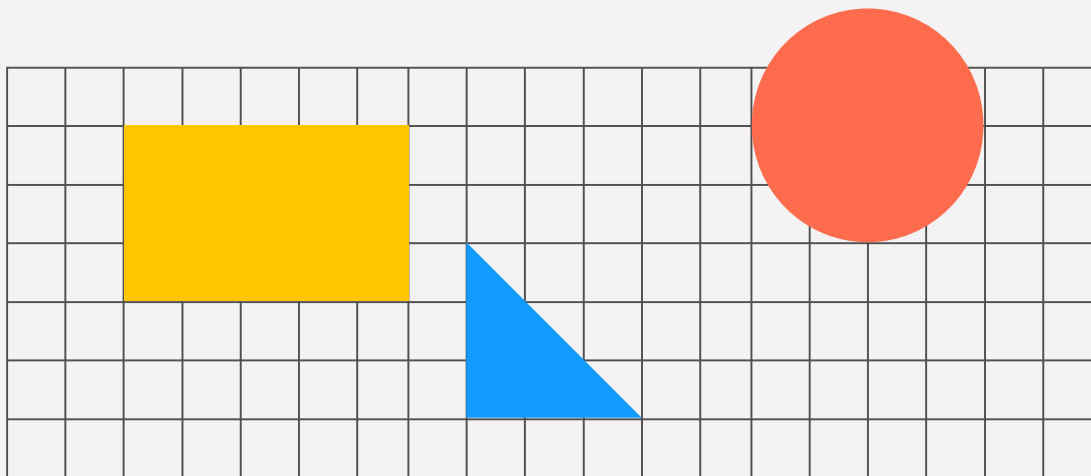



Unidad 4

Lista





Una **lista** es una estructura de datos dinámica que permite almacenar una colección de elementos de forma **lineal**. A diferencia de un array, **una lista puede crecer y reducir su tamaño en tiempo de ejecución**.

En C, como no hay listas integradas como en otros lenguajes, se implementan usando **estructuras (struct)** y **punteros (pointer)**.

Tipos de listas

- **Lista simplemente enlazada**
- **Lista doblemente enlazada**
- **Lista circular** (variante de las anteriores)

Lista Simplemente Enlazada



Cada elemento de la lista (llamado **nodo**) tiene dos partes:

- **Dato**
- **Puntero** al siguiente nodo

Lista que guarda enteros



Explicación del código

- `struct Nodo` define cómo luce un nodo (dato y puntero al siguiente).
- `insertarAlInicio` agrega un nuevo nodo al principio de la lista.
- `mostrarLista` recorre e imprime todos los nodos hasta que llega a `NULL`.

Definición del nodo

```
// Definición del nodo  
struct Nodo {  
    int dato;  
    struct Nodo* siguiente;  
};
```

Función para agregar un nodo al inicio

```
// Función para agregar un nodo al inicio
void insertarAlInicio(struct Nodo** cabeza, int valor) {
    struct Nodo* nuevoNodo = (struct Nodo*)malloc(sizeof(struct Nodo));
    nuevoNodo->dato = valor;
    nuevoNodo->siguiente = *cabeza;
    *cabeza = nuevoNodo;
}
```

Función para mostrar la lista

```
// Función para mostrar la lista
void mostrarLista(struct Nodo* nodo) {
    while (nodo != NULL) {
        printf("%d -> ", nodo->dato);
        nodo = nodo->siguiente;
    }
    printf("NULL\n");
}
```


Uso de los métodos par guardar un número entero

```
int main() {  
    struct Nodo* lista = NULL;  
  
    insertarAlInicio(&lista, 10);  
    insertarAlInicio(&lista, 20);  
    insertarAlInicio(&lista, 30);  
  
    printf("Lista enlazada: ");  
    mostrarLista(lista);  
  
    return 0;  
}
```

Operaciones comunes de Listas

- Insertar al inicio
- Insertar al final
- Eliminar un nodo
- Buscar un valor
- Mostrar todos los valores

Función para crear un nuevo nodo

```
// Función para crear un nuevo nodo  
Nodo* crearNodo(int dato) {  
    Nodo* nuevoNodo = (Nodo*)malloc(sizeof(Nodo));  
    nuevoNodo->dato = dato;  
    nuevoNodo->siguiente = NULL;  
    return nuevoNodo;  
}
```

Función para insertar un nodo al inicio de la lista

```
// Función para insertar un nodo al inicio de la lista  
void insertarInicio(Nodo** cabeza, int dato) {  
    Nodo* nuevoNodo = crearNodo(dato);  
    nuevoNodo->siguiente = *cabeza;  
    *cabeza = nuevoNodo;  
}
```

Función para insertar un nodo al final de la lista

```
// Función para insertar un nodo al final de la lista
void insertarFinal(Nodo** cabeza, int dato) {
    Nodo* nuevoNodo = crearNodo(dato);
    if (*cabeza == NULL) {
        *cabeza = nuevoNodo;
        return;
    }
    Nodo* temp = *cabeza;
    while (temp->siguiente != NULL) {
        temp = temp->siguiente;
    }
    temp->siguiente = nuevoNodo;
}
```

Función para eliminar un nodo con un valor específico

```
// Función para eliminar un nodo con un valor específico
void eliminarNodo(Nodo** cabeza, int valor) {
    Nodo* temp = *cabeza;
    Nodo* anterior = NULL;
    while (temp != NULL && temp->dato != valor) {
        anterior = temp;
        temp = temp->siguiente;
    }
    if (temp == NULL) {
        printf("Valor no encontrado en la lista.\n");
        return;
    }
    if (anterior == NULL) {
        *cabeza = temp->siguiente;
    } else {
        anterior->siguiente = temp->siguiente;
    }
    free(temp);
    printf("Nodo con valor %d eliminado.\n", valor);
}
```

Función para buscar un valor en la lista

```
// Función para buscar un valor en la lista
void buscarValor(Nodo* cabeza, int valor) {
    Nodo* temp = cabeza;
    while (temp != NULL) {
        if (temp->dato == valor) {
            printf("Valor %d encontrado en la lista.\n", valor);
            return;
        }
        temp = temp->siguiente;
    }
    printf("Valor %d no encontrado en la lista.\n", valor);
}
```

Función para mostrar todos los valores de la lista

```
// Función para mostrar todos los valores de la lista
void mostrarValores(Nodo* cabeza) {
    Nodo* temp = cabeza;
    if (temp == NULL) {
        printf("La lista está vacía.\n");
        return;
    }
    printf("Valores en la lista: ");
    while (temp != NULL) {
        printf("%d ", temp->dato);
        temp = temp->siguiente;
    }
    printf("\n");
}
```