Tipos de Datos en Programación y su Importancia en la Ingeniería de Sistemas

Objetivo:

Comprender los diferentes tipos de datos utilizados en programación, su clasificación y su relevancia en el desarrollo de software y sistemas.

Introducción

En la ingeniería de sistemas y el desarrollo de software, los tipos de datos son fundamentales para definir cómo se almacena, procesa y manipula la información en un programa. Un tipo de dato determina las operaciones que se pueden realizar con una variable, así como el espacio que ocupa en memoria.

Una estructura de datos es una forma de organizar y almacenar datos en una computadora que puedan ser utilizados de manera eficiente.

El **diseño** es un proceso critico en la programación. Se deben de considerar factores como la eficiencia, la escalabilidad y la facilidad de mantenimiento para garantizar que la estructura sea adecuada para su propósito.

La **eficiencia** es un factor clave en el diseño de estructura de datos. Se deben de considerar factores como el tiempo de acceso, el espacio de almacenamiento y la complejidad del algoritmo para garantizar que la estructura sea eficiente en su uso.

Clasificación de los Tipos de Datos

Los tipos de datos se clasifican en dos categorías principales: *primitivos* y *compuestos*.

Tipos de Datos Primitivos

Son los tipos básicos que proporcionan los lenguajes de programación. Incluyen:

- **Enteros (int):**
  - Representan números enteros (positivos, negativos o cero).
  - Ejemplo: *int edad = 25;*
  - Tamaño en memoria: Depende del lenguaje (ej: 4 bytes en muchos casos).

- **Flotantes (float) y Dobles (double):**
  - Representan números con decimales.
  - *float* tiene menor precisión que *double*.
  - Ejemplo: *float precio = 19.99;*
  - Tamaño en memoria: *float* (4 bytes), *double* (8 bytes).
  - *float* puede mantener hasta 7 cifras decimales de manera precisa, mientras que *double* puede mantener hasta 15.

- **Caracter (char):**
  - Almacenan un único carácter.
  - Ejemplo: *char letra = 'A';*
  - Tamaño en memoria: 1 byte.

- **Booleanos (bool):**
  - Representan valores lógicos: `true` (verdadero) o `false` (falso).
  - Ejemplo: *bool esValido = true;*
  - Tamaño en memoria: 1 byte.

- **Punteros (pointer):**
  - Almacenan direcciones de memoria.
  - Ejemplo: *int* ptr = &edad;*
  - Tamaño en memoria: Depende de la arquitectura (ej: 4 bytes en sistemas de 32 bits).

Tipos de Datos Compuestos

Son estructuras que combinan múltiples valores o tipos de datos primitivos. Incluyen:

- Arreglos (arrays):
  - Colección de elementos del mismo tipo.
  - Estructura de datos lineales.
  - Ejemplo: *int numeros[5] = {1, 2, 3, 4, 5};*
  - Tamaño en memoria: Depende del número de elementos.

- Estructuras (struct):
  - Agrupan datos de diferentes tipos bajo un mismo nombre.
  - Ejemplo:

```
c

struct Persona {
  char nombre[50];
  int edad;
  float altura;
};
```

- Uniones (union):
  - Similar a una estructura, pero todos los miembros comparten la misma ubicación de memoria.
  - Ejemplo:

```
c

union Dato {
  int entero;
  float flotante;
  char caracter;
};
```

- Cadenas (strings):
    - Secuencia de caracteres.
    - Ejemplo: *char nombre[] = "Ingeniería de Sistemas";*

- Listas, Pilas y Colas:
    - Estructuras de datos lineal y dinámicas que permiten almacenar y manipular colecciones de elementos.
    - Ejemplo: FIFO

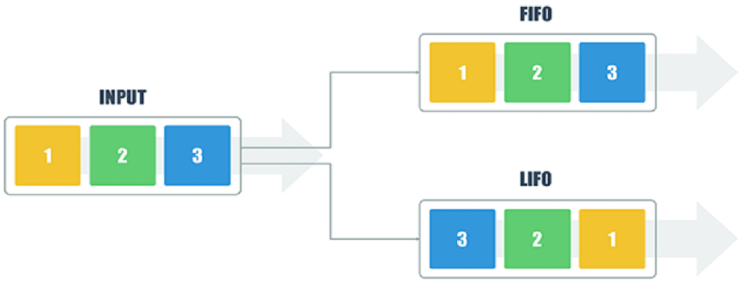
- Objetos (en programación orientada a objetos):
    - Instancias de clases que encapsulan datos y comportamientos.
    - Ejemplo:

```
java

class Persona {

  String nombre;

  int edad;

}
```

Tipos de Datos Abstractos (TDA)

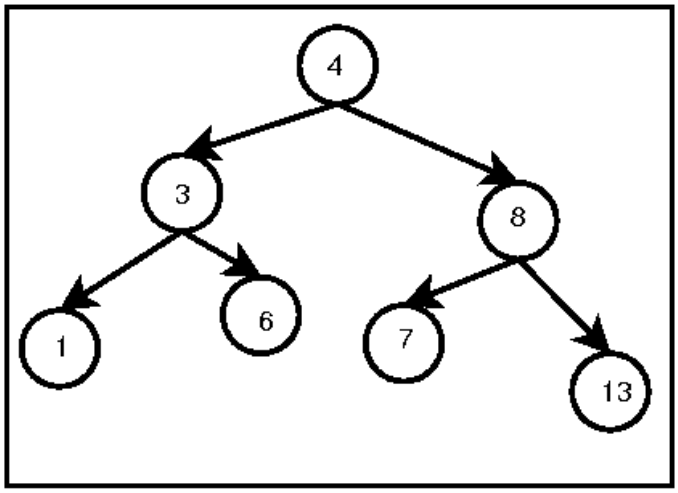
Los Tipos de Datos Abstractos son modelos matemáticos que definen un conjunto de valores y operaciones sobre esos valores, sin especificar cómo se implementan. Dentro de estos, existen las estructuras de datos lineales, en los que los datos se organizan de manera secuencial, como las listas, pilas y las colas :

- Pilas (LIFO):Último en entrar, primero en salir.
- Colas (FIFO):Primero en entrar, primero en salir.

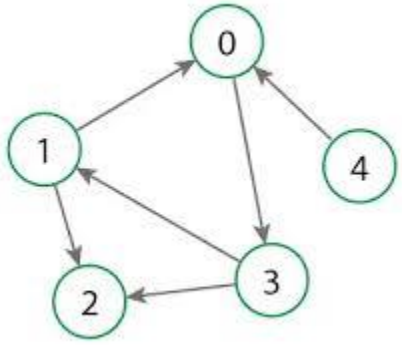


Además, los no lineales que son aquellos datos que no se organizan de forma secuencial, si no que pueden tener múltiples caminos o ramas, como lo Ejemplos comunes incluyen:

- Árboles: Estructuras de datos jerárquica, no lineales que constan de nodos conectados por ramas. Cada nodo tiene un valor y puede tener cero o más hijos.



- Grafos: Estructuras de datos no lineales que constan de nodos conectados por aristas. Cada nodo puede tener cero o mas aristas que lo conectan a otros nodos.



Importancia de los Tipos de Datos en la Ingeniería de Sistemas

- Eficiencia: Elegir el tipo de dato adecuado optimiza el uso de memoria y mejora el rendimiento del software.
- Precisión: Los tipos de datos garantizan que los valores se manipulen correctamente (ej: evitar errores en cálculos numéricos).
- Mantenibilidad: Un código bien estructurado con tipos de datos claros es más fácil de entender y modificar.
- Escalabilidad: Los tipos de datos compuestos y abstractos permiten manejar grandes volúmenes de información de manera eficiente.

Ejemplos Prácticos

Uso de tipos primitivos:

java

int edad = 30;

double salario = 1500.50;

char genero = 'M';

boolean esActivo = true;

Uso de tipos compuestos:

python

```
# Lista en Python
numeros = [1, 2, 3, 4, 5]

# Diccionario en Python
persona = {"nombre": "Juan", "edad": 25}
```

Implementación de un TDA (Pila):

c++

```
#include <stack>
using namespace std;

int main() {
    stack<int> pila; // type of container adaptors with LIFO(Last In First Out)
    pila.push(10);
    pila.push(20);
    pila.pop(); // Elimina el último elemento
    return 0;
}
```

PHP

```
<?php
// Crear un arreglo
$ciudades= array('Cali', 'Bogota', 'Medellin');

// Adicionar una ciudad al arreglo
```

```
$ciudades[]='Bucaramanga';
echo 'Adicionar ';
Print_r($ciudades);

// Eliminar una ciudad al arreglo
$ciudad_eliminar='Cali';
$llave= array_search($ciudad_eliminar, $ciudades);
if($llave!==false){
  unset($ciudades[$llave]);
}

// Identifiar los indices
  echo 'Eliminar ';
  Print_r($ciudades);

// Modificar una ciudad al arreglo
$ciudades[1]='Cali';
echo 'Modificar ';
Print_r($ciudades);

// Extraer dos ciudades del arreglo
$extraer= array_slice($ciudades,1,2);
echo 'Extraer ';
Print_r($extraer);
?>
```

Tipos de Cadenas (Strings) en Bases de Datos

*En las bases de datos, los **tipos de datos de cadena (strings)** son utilizados para almacenar texto, como nombres, direcciones, descripciones, entre otros. Cada sistema de gestión de bases de datos (SGBD) ofrece diferentes tipos de datos para manejar cadenas, cada uno con sus propias características y limitaciones. Este documento explica los tipos de cadenas más comunes, sus usos y ejemplos prácticos.*

Tipos de Datos de Cadena en Bases de Datos

*Los tipos de datos de cadena se clasifican principalmente en dos categorías: **cadenas de longitud fija** y **cadenas de longitud variable**. A continuación, se describen los tipos más comunes:*

Cadenas de Longitud Fija

Estos tipos de datos almacenan cadenas de texto con una longitud predefinida. Si la cadena es más corta que la longitud especificada, se rellena con espacios en blanco.

- CHAR(n):

- Descripción: Almacena cadenas de longitud fija, donde `n` es el número máximo de caracteres.

- Uso: Ideal para almacenar datos con una longitud constante, como códigos o identificadores.

- Ejemplo:

```sql
CREATE TABLE Productos (
    codigo CHAR(10),
    nombre VARCHAR(50)
```

);

```

En este caso, `codigo` siempre ocupará 10 caracteres, incluso si el valor almacenado es más corto.

### *Cadenas de Longitud Variable*

*Estos tipos de datos almacenan cadenas de texto con una longitud que puede variar hasta un máximo definido. No se rellenan con espacios en blanco.*

*- VARCHAR(n):*

*- Descripción: Almacena cadenas de longitud variable, donde `n` es el número máximo de caracteres.*

*- Uso: Ideal para almacenar datos con longitudes variables, como nombres o descripciones.*

*- Ejemplo:*

*```sql*

*CREATE TABLE Clientes (*

*nombre VARCHAR(50),*

*direccion VARCHAR(100)*

*);*

```

Aquí, `nombre` puede almacenar hasta 50 caracteres, pero solo ocupará el espacio necesario.

- TEXT:

- Descripción: Almacena cadenas de longitud variable con un límite mucho mayor que `VARCHAR`.

- Uso: Ideal para almacenar textos largos, como descripciones extensas o comentarios.

- **Ejemplo:**

```sql
CREATE TABLE Articulos (
    id INT PRIMARY KEY,
    contenido TEXT
);
```

En este caso, `contenido` puede almacenar grandes cantidades de texto.

Cadenas Unicode

Estos tipos de datos permiten almacenar caracteres especiales y de diferentes idiomas, utilizando codificación Unicode.

- NCHAR(n):

  - Descripción: Similar a `CHAR`, pero almacena cadenas de longitud fija en formato Unicode.

  - Uso: Ideal para almacenar texto en múltiples idiomas con longitud fija.

  - Ejemplo:

```sql
CREATE TABLE Idiomas (
    codigo NCHAR(5),
    nombre NVARCHAR(50)
);
```

- NVARCHAR(n):

- Descripción: Similar a `VARCHAR`, pero almacena cadenas de longitud variable en formato Unicode.

- Uso: Ideal para almacenar texto en múltiples idiomas con longitud variable.

- Ejemplo:

```sql
CREATE TABLE Traducciones (
    id INT PRIMARY KEY,
    texto NVARCHAR(200)
);
```

Otros Tipos de Cadenas

- BLOB (Binary Large Object):

- Descripción: Almacena datos binarios, como imágenes o archivos, pero también puede usarse para texto.

- Uso: Ideal para almacenar datos no estructurados.

- Ejemplo:

```sql
CREATE TABLE Archivos (
    id INT PRIMARY KEY,
    archivo BLOB
);
```

- ENUM:

- Descripción: Almacena un valor de una lista predefinida de cadenas.

- Uso: Ideal para campos con opciones limitadas, como estados o categorías.

- Ejemplo:

```sql
CREATE TABLE Pedidos (
    id INT PRIMARY KEY,
    estado ENUM('pendiente', 'en_proceso', 'completado')
);
```

Comparación entre Tipos de Cadenas

Tipo de Dato	Longitud	Codificación	Uso Típico
CHAR(n)	Fija	No Unicode	Códigos, identificadores
VARCHAR(n)	Variable	No Unicode	Nombres, direcciones
TEXT	Variable	No Unicode	Descripciones largas
NCHAR(n)	Fija	Unicode	Texto en múltiples idiomas
NVARCHAR(n)	Variable	Unicode	Texto en múltiples idiomas
BLOB	Variable	Binario	Archivos, datos binarios
ENUM	Fija	No Unicode	Opciones predefinidas

Ejemplo Práctico

Base de Datos de una Biblioteca

*1. **Tabla Libros:***

```sql
CREATE TABLE Libros (
    id INT PRIMARY KEY,
    titulo VARCHAR(100),
    autor VARCHAR(100),
    descripcion TEXT,
    idioma NCHAR(20)
);
```

*2. **Tabla Usuarios:***

```sql
CREATE TABLE Usuarios (
    id INT PRIMARY KEY,
    nombre VARCHAR(50),
    direccion VARCHAR(200),
    telefono CHAR(10)
);
```

*3. **Tabla Préstamos:***

```sql
CREATE TABLE Prestamos (
    id INT PRIMARY KEY,
    id_libro INT,
    id_usuario INT,
    estado ENUM('activo', 'devuelto'),
    FOREIGN KEY (id_libro) REFERENCES Libros(id),
    FOREIGN KEY (id_usuario) REFERENCES Usuarios(id)
);
```

Los tipos de datos de cadena son esenciales en el diseño de bases de datos, ya que permiten almacenar y manipular información textual de manera eficiente. La elección del tipo de cadena adecuado depende de factores como la longitud del texto, la necesidad de soporte para múltiples idiomas y el uso previsto de los datos. Comprender estas diferencias es fundamental para diseñar bases de datos optimizadas y eficaces.

Actividad Propuesta:

1. Crear una tabla en SQL que utilice los tipos de datos `CHAR`, `VARCHAR`, `TEXT` y `ENUM`.

2. Explicar en qué casos sería recomendable usar `NVARCHAR` en lugar de `VARCHAR`.

Referencias:

- MySQL Documentation: [String Data Types](https://dev.mysql.com/doc/refman/8.0/en/string-types.html)

- PostgreSQL Documentation: [Character Types](https://www.postgresql.org/docs/current/datatype-character.html)

- Microsoft SQL Server Documentation: [String Functions](https://docs.microsoft.com/en-us/sql/t-sql/functions/string-functions-transact-sql)

Herramientas:

- https://onecompiler.com/
- https://www.youtube.com/watch?v=SWCV9lb23WQ