ESPE

UNIVERSIDAD DE LAS FUERZAS ARMADAS

INNOVACIÓN PARA LA EXCELENCIA

Fundamentos de Programación

TIPOS DE DATOS ABSTRACTOS (T.D.A.)

Ing. Washington Loza H. Mgs.

Departamento de Ciencias de la Computación

ESPE

UNIVERSIDAD DE LAS FUERZAS ARMADAS

INNOVACIÓN PARA LA EXCELENCIA

ALGUNOS LINEAMIENTOS

- **RÚBRICAS DE LA EVALUACIÓN.**

Estructura de Datos		
Técnica de Evaluación	**Ponderación**	**Porcentual**
Tareas(Deberes) o guías	2	10
Talleres	2	10
Investigación Bibliográfica	1	5
Evaluaciones (Pruebas)	5	25
Exposición	3	15
Examen Parcial	7	35
	20	**100**

UNIDAD 1: INTRODUCCIÓN Y FUNDAMENTOS DE LAS ESTRUCTURAS DE DATOS

- Tipos de Datos Abstractos (TDA).
- Recursividad
- Listas
- Pilas
- Colas

ESPE

UNIVERSIDAD DE LAS FUERZAS ARMADAS

INNOVACIÓN PARA LA EXCELENCIA

Tipos de Datos Abstractos (TDA).

ESPE
UNIVERSIDAD DE LAS FUERZAS ARMADAS
INNOVACIÓN PARA LA EXCELENCIA

Tipos de Datos Abstractos (TDA)

Abstracción

"Los humanos hemos desarrollado una técnica excepcionalmente potente para tratar la **complejidad**: abstraernos de ella.
Incapaces de dominar en su totalidad los objetos complejos, se **ignoran** los detalles **no esenciales**, tratando en su lugar con el modelo ideal del objeto y centrándonos en el estudio de sus aspectos esenciales." — *William Wulf*

Tipos de Datos Abstractos (TDA)

Datos en la vida cotidiana

Cuando visita un portal de Internet del consumidor que permite navegar y ordenar en línea bienes.

Cuando visita un sitio Web interactivo de servicio al cliente

Si usa banca electrónica, puede recuperar registros de base de datos acerca de transacciones para sus cuentas

Cuando usa una tarjeta de crédito, espera la aprobación su compra antes de presentarle un recibo.

Registros médicos y datos de facturación se conservan en una base de datos.

Sus registros laborales y escolares se pueden mantener en una base de datos que almacena información básica

Tipos de Datos Abstractos (TDA)

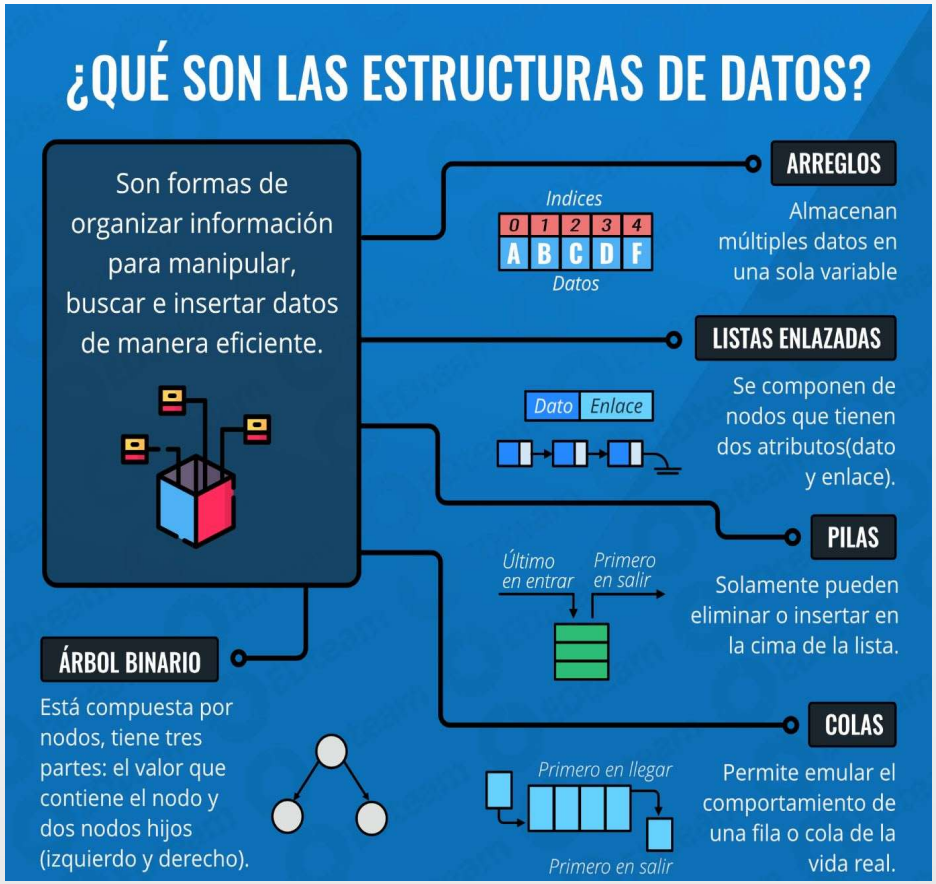
Introducción a las Estructuras de Datos y su relación con los Tipos Abstractos de Datos (TDA)

¿Qué es una Estructura de Datos (ED)?

Una **Estructura de Datos** es una **forma organizada de representar y administrar información dentro de un programa**.

No se trata solo de guardar datos, sino de **darles una forma lógica y eficiente** para que puedan ser **utilizados, buscados o modificados rápidamente**.

En otras palabras, una estructura de datos es una **manera sistemática de organizar los datos** para resolver un problema de la forma más óptima posible.



Tipos de Datos Abstractos (TDA)

¿Qué es una Estructura de Datos (ED)?

- **Definición formal:**

Una estructura de datos es la **representación organizada de un conjunto de información**, diseñada para permitir el acceso, modificación y procesamiento eficiente de los datos.

- **Definición conceptual:**

Es la **manera natural de manejar la información** dentro de un sistema, de acuerdo con el propósito que persigue el programa o el algoritmo.

Por ejemplo:

- Si necesitas guardar una lista ordenada de alumnos, puedes usar un **arreglo**.
- Si necesitas asociar cada alumno con su calificación, puedes usar un **diccionario o mapa**.
- Si necesitas organizar tareas pendientes, una **cola (queue)** puede ser más apropiada.

Cada estructura tiene ventajas, limitaciones y contextos de uso específicos.

Tipos de Datos Abstractos (TDA)

La relación entre Estructuras de Datos y Algoritmos

Las **estructuras de datos** y los **algoritmos** son dos pilares inseparables de la programación.

Podemos decir que **los datos sin algoritmos no se mueven**, y **los algoritmos sin datos no tienen sobre qué actuar**.

El algoritmo define el **cómo se hace**, mientras que la estructura de datos define **dónde y en qué forma se guarda la información**.

Concepto	Descripción
Algoritmo	Conjunto de pasos ordenados para resolver un problema.
Estructura de datos	Forma en que se organizan y almacenan los datos para que el algoritmo los use eficientemente.

Ejemplo: Supón que necesitas ordenar los nombres de los alumnos de un curso:

- La estructura de datos será un **arreglo o lista de nombres**.
- El algoritmo podría ser **burbuja, inserción, mergesort o quicksort**.

Si la estructura no está bien elegida, el mejor algoritmo del mundo se volverá lento e ineficiente.

Por eso, en todo sistema informático existe una **fuerte relación entre estructura de datos y algoritmo**: uno depende completamente del otro.

Tipos de Datos Abstractos (TDA)

Descripción formal de una Estructura de Datos

Las estructuras son registros de datos, cada elemento de la estructura se llama miembro y pueden ser de distintos tipos de datos, también pueden ser arreglos o incluso otras estructuras.

Para declarar una estructura:

```
struct nombre_estructura
{
        tipo_de_dato_miembro1 nombre_miembro1;
        tipo_de_dato_miembro2 nombre_miembro2;
};
```

Para definir variables que utilicen la estructura:

```
struct nombre_estructura
{
        tipo_de_dato_miembro1 nombre_miembro1;
        tipo_de_dato_miembro2 nombre_miembro2;
} variable1, variable2, variable3;
```

Para definir nuevas variables que utilicen una estructura previamente declarada:

```
struct nombre_estructura variable1, variable2, variable3;
```

Para inicializar una estructura:

```
struct nombre_estructura variable1 = { valor_miembro1, valor_miembro2……};
```

Para asignar un valor(**v**) en el miembro (**m**) de una variable de tipo estructura (**e**) se debe realizar:

e.m = v;

Tipos de Datos Abstractos (TDA)

Realizar un ejemplo de una Estructura de Datos

Realizar un programa que guarde el nombre, ciudad y edad de 3 personas usando un arreglo en donde cada elemento es una estructura y posteriormente muestre los datos de cada persona en una línea.

```
#include <iostream>

using namespace std;

int main()
{
    struct persona
    {
        char nombre [15];
        int edad;
        char ciudad[15];
    };
    struct persona p[3];
    for (int i=0;i<=2;i++)
    {
        cout<<"Ingresa el nombre "<<i+1<<endl;
        gets(p[i].nombre);
        cout<<"Ingresa la edad de "<<p[i].nombre<<endl;
        cin>>p[i].edad;
        fflush(stdin);
        cout<<"Ingresa la ciudad de "<<p[i].nombre<<endl;
        gets(p[i].ciudad);
    }
    for (int i=0;i<=2;i++)
    {
        cout<<p[i].nombre<<" tiene "<<p[i].edad<<" y vive en "<<p[i].ciudad<<endl;
    }
    return 0;
}
```

Tipos de Datos Abstractos (TDA)

Tipos concretos vs Tipos abstractos de datos

En los lenguajes de programación existen dos grandes familias de tipos de datos:

- **Los tipos concretos** (también llamados *primitivos* o *nativos*); y,
- **Los tipos abstractos** (o *definidos por el programador*).

Comprender la diferencia entre ambos es esencial para dominar la idea de **abstracción de datos**, base de todo diseño estructurado u orientado a objetos.

	Tipo de datos básico	Tipo de datos estructurado

Definición	`int`	`struct contact_information {`
		`  char firstname[100];`
		`  char lastname[200];`
		`  int homephone;`
		`  int mobilephone;`
		`};`

Declaración	`int a, b[100];`	`struct contact_information a, b[100];`

Tipos de Datos Abstractos (TDA)

Tipos concretos (O Nativos)

Los **tipos concretos** son aquellos que **ya vienen definidos por el lenguaje de programación**. Son “concretos” porque conocemos exactamente **cómo se representan en memoria** y **qué operaciones básicas permiten realizar**.

Estos tipos suelen ser la **materia prima** con la que se construyen estructuras más complejas.

Características principales:

Nativos del lenguaje: están definidos internamente por el compilador.

Implementación conocida: sabemos su tamaño, representación y comportamiento (por ejemplo, un `int` ocupa 4 bytes y almacena valores enteros).

Velocidad y simplicidad: son eficientes porque están optimizados a nivel del hardware.

Alcance limitado: representan información básica, pero no conceptos del mundo real por sí solos.

Ejemplos comunes:

- **Tipos básicos:** `int`, `float`, `double`, `char`, `bool`. Cada uno representa un tipo elemental: números, caracteres, valores lógicos.
- **Tipos estructurados o compuestos:** `array`, `struct`, `tuple`, `pointer`. Permiten agrupar o enlazar varios datos simples, pero sin imponer reglas de comportamiento.

Tipos de Datos Abstractos (TDA)

Tipos abstractos de datos (TAD o TDA)

Los **tipos abstractos** son definidos por el programador. Surgen del análisis de un problema y permiten **modelar objetos o conceptos del mundo real** mediante **valores, operaciones y reglas**.

Un TDA no especifica cómo están implementados los datos internamente, sino **qué se puede hacer con ellos**. En otras palabras, es una **abstracción del comportamiento**, no una descripción física de la memoria

Características principales:

Definidos por el usuario: se construyen mediante clases, estructuras o módulos.

Enfocados en el comportamiento: especifican operaciones, no representaciones internas.

Múltiples implementaciones posibles: un mismo TDA puede representarse con arrays, listas, archivos, etc.

Basados en un contrato: establecen reglas (precondiciones, postcondiciones e invariantes) que deben cumplirse siempre

Ejemplos comunes:

El TDA **Alumno**, que modela los datos y operaciones de un estudiante:

```
TDA Alumno

Valores: Cedula, ID, Nombre, Apellido, Edad, Carrera

Operaciones:

  crearAlumno(), mostrar(), actualizarEdad(), mismoID()
```

Tipos de Datos Abstractos (TDA)

Un **Tipo Abstracto de Datos (TDA)** es una **abstracción formal** que define un conjunto de datos y las operaciones que se pueden realizar sobre ellos, sin preocuparse por **cómo están implementados** en la memoria.

Un **TDA describe qué se puede hacer**, pero **no cómo se hace**. Su representación concreta está **oculta al usuario** y solo puede manipularse a través de las **operaciones definidas en su interfaz pública**.

El objetivo de un TDA es **proteger la integridad de los datos**, asegurando que todas las acciones sobre ellos cumplan las reglas definidas para mantener la coherencia del sistema.

```
class Alumno {
private:
    // Valores (estado interno)
    string nombre;
    string cedula;
    int edad;
    string carrera;
```

Tipos de Datos Abstractos (TDA)

Un **Tipo Abstracto de Datos (TDA)** es una **abstracción formal** que define un conjunto de datos y las operaciones que se pueden realizar sobre ellos, sin preocuparse por **cómo están implementados** en la memoria.

Un **TDA describe qué se puede hacer**, pero **no cómo se hace**. Su representación concreta está **oculta al usuario** y solo puede manipularse a través de las **operaciones definidas en su interfaz pública**.

El objetivo de un TDA es **proteger la integridad de los datos**, asegurando que todas las acciones sobre ellos cumplan las reglas definidas para mantener la coherencia del sistema.

```
class Alumno {
private:
    // Valores (estado interno)
    string nombre;
    string cedula;
    int edad;
    string carrera;
```

Tipos de Datos Abstractos (TDA)

Estructura General de un TDA

Elemento	Descripción	Ejemplo en el TDA `Alumno`
Valores o estado	Los datos que describen la entidad.	`nombre`, `cedula`, `edad`, `carrera`
Operaciones	Las acciones permitidas sobre los datos.	`crearAlumno()`, `mostrar()`, `actualizarEdad()`
Reglas o contrato	Condiciones que garantizan la coherencia (pre, post, invariantes).	`edad > 0`, `cedula != ""`

Tipos de Datos Abstractos (TDA)

De las Estructuras de Datos a los Tipos Abstractos de Datos (TDA)

Toda estructura de datos implementa, en realidad, un **Tipo Abstracto de Datos (TDA)**.

El **TDA** es el **modelo lógico o conceptual** que describe **qué operaciones** pueden realizarse sobre los datos y **qué reglas** deben cumplirse, **sin preocuparse por cómo** están implementadas internamente.

Por tanto, las estructuras de datos son **la materialización concreta** de esos tipos abstractos en un lenguaje de programación.

Dicho de forma simple:

- El **TDA** define el *comportamiento y las reglas* (la teoría).
- La **estructura de datos** define la *implementación y almacenamiento* (la práctica).

Tipos de Datos Abstractos (TDA)

Ejemplo del concepto al Código…!

Imagina que estás diseñando una aplicación para registrar alumnos.

TDA conceptual:

Entidad:
Alumno.
Valores:
nombre, cédula, edad, carrera.
Operaciones:
crearAlumno(), mostrar(), actualizarEdad(), mismoID().
Reglas:
edad > 0, cédula no vacía.

Estructura de datos concreta:

```
class Alumno {
private:
    std::string nombre;
    std::string cedula;
    int edad;
    std::string carrera;
public:
    void crearAlumno(std::string n, std::string c, int e, std::string car);
    void mostrar();
    void actualizarEdad(int nuevaEdad);
};
```

Tipos de Datos Abstractos (TDA)

Operaciones Básicas de un TDA

Toda estructura basada en un TDA debe implementar las siguientes operaciones esenciales. A continuación, se desarrollan **cada una de ellas con detalle**, aplicadas al **TDA Alumno**.

Inicialización

- La inicialización es el proceso de **crear el estado inicial del TDA**.
- Permite definir un objeto válido y preparado para su uso.

```
Alumno a1;
a1.crearAlumno("Maria Lopez", "0104567890", 20, "Ingenieria en Sistemas");
```

Inserción

- La operación de inserción consiste en **agregar un nuevo elemento** dentro de una colección o estructura de datos basada en el TDA.

```
ListaAlumnos.insertar(a1);
```

Tipos de Datos Abstractos (TDA)

Operaciones Básicas de un TDA

Eliminación

- Permite **borrar un elemento** específico de la estructura, identificándolo mediante una clave (por ejemplo, la cédula).

```
ListaAlumnos.eliminar("0104567890");
```

Búsqueda

- Es la operación que permite **localizar y recuperar** información de un elemento dentro del TDA.

```
Alumno encontrado = ListaAlumnos.buscar("0109876543");
encontrado.mostrar();
```

Vaciado

- El vaciado elimina **todos los elementos** del TDA, dejando la estructura vacía pero lista para volver a usarse.

```
ListaAlumnos.vaciar();
```

Tipos de Datos Abstractos (TDA)

Resumen de Operaciones de un TDA

Operación	Descripción	Ejemplo	Relación con el TDA
Inicialización	Crea el objeto con datos válidos.	`crearAlumno()`	Establece estado inicial.
Inserción	Agrega un objeto `Alumno` a una colección.	`insertar(a1)`	Añade nuevos elementos.
Eliminación	Borra un alumno identificado por su cédula.	`eliminar("0104567890")`	Mantiene integridad de datos.
Búsqueda	Localiza y devuelve un alumno existente.	`buscar("0109876543")`	Permite consultar sin modificar.
Vaciado	Limpia toda la estructura sin destruirla.	`vaciar()`	Restaura el estado vacío.

Tipos de Datos Abstractos (TDA)

```cpp
#include <iostream>
#include <string>
using namespace std;

/*-------------------------------------------------
   Definicion del TDA Alumno (Tipo Abstracto de Datos)
-------------------------------------------------*/
class Alumno {
private:
    // Valores (estado interno)
    string nombre;
    string cedula;
    int edad;
    string carrera;

public:
    // Constructor: Inicializa los atributos (operacion de insercion/inicializaci
    void crearAlumno(string n, string c, int e, string car) {
        if (e <= 0 || c.empty()) {
            cout << "Error: datos invalidos al crear el alumno." << endl;
            return;
        }
        nombre = n;
        cedula = c;
        edad = e;
        carrera = car;
    }

    // Operacion de consulta: muestra la informacion (operacion de busqueda)
    void mostrar() {
        cout << "---------------------------------" << endl;
        cout << "Nombre:  " << nombre << endl;
        cout << "Cedula:  " << cedula << endl;
        cout << "Edad:    " << edad << endl;
        cout << "Carrera: " << carrera << endl;
        cout << "---------------------------------" << endl;
    }

    // Operacion de modificacion: actualiza un valor existente (actualizacion con
    void actualizarEdad(int nuevaEdad) {
        if (nuevaEdad > 0)
            edad = nuevaEdad;
        else
            cout << "Error: edad invalida." << endl;
    }
};

/*-------------------------------------------------
   Programa principal para probar el TDA Alumno
-------------------------------------------------*/
int main() {
    Alumno a1;  // Declaramos un objeto del TDA Alumno

    // 1. Inicializacion (crear un alumno)
    a1.crearAlumno("Maria Lopez", "0104567890", 20, "Ingenieria en Sistemas");

    // 2. Consulta (mostrar los datos)
    cout << "Alumno registrado:" << endl;
    a1.mostrar();

    // 3. Modificacion (actualizar la edad)
    cout << "Actualizando edad..." << endl;
    a1.actualizarEdad(21);

    // 4. Consulta posterior (comprobar cambios)
    cout << "Datos despues de la actualizacion:" << endl;
    a1.mostrar();

    return 0;
}
```

Tipos de Datos Abstractos (TDA)

Actividad en clase 1.

1. ¿Qué es una estructura de datos?
2. Comente los diferentes tipos de estructuras de datos.
3. Comente la categorización de módulos.
4. Comente qué es un objeto abstracto.
5. Comente qué es un tipo de dato abstracto.