UNIVERSIDAD TÉCNICA PARTICULAR DE LOJA

La Universidad Católica de Loja

MODALIDAD ABIERTA Y A DISTANCIA

FACULTAD DE INGENIERÍAS Y ARQUITECTURA

CARRERA TECNOLOGÍAS DE LA INFORMACIÓN

Practicante: TIPAN CHUQUITARCO ALEX VINICIO

Tutor académico: Ing. IRENE ROBALINO PEDRO DANIEL

ABRIL/2025 – AGOSTO/2025

1.Introducción

La Programación Orientada a Objetos (POO) es un paradigma de programación que modela sistemas complejos en base a objetos del mundo real. Uno de los elementos clave dentro de este paradigma son las estructuras de datos, ya que permiten organizar, almacenar y manipular eficientemente información. Dentro de estas estructuras, los arreglos son fundamentales por su simplicidad y versatilidad. El uso de arreglos en POO no solo permite manejar datos de manera secuencial, sino también facilita la implementación de otros conceptos como herencia, encapsulamiento y polimorfismo al interactuar con objetos.

¿Qué son los arreglos de tipos de datos simples o primitivos?

Un arreglo (o array) es una estructura de datos que almacena múltiples elementos del mismo tipo bajo un solo nombre de variable, organizados en posiciones numeradas llamadas índices. Cuando hablamos de tipos de datos simples o primitivos, nos referimos a datos básicos del lenguaje de programación, como:

- int (enteros)
- float o double (decimales)
- char (caracteres)
- boolean (verdadero o falso)

¿Para qué se usan los arreglos?

Los arreglos se utilizan en programación para gestionar y manipular conjuntos de datos de manera eficiente. Son una de las estructuras de datos más fundamentales y ampliamente utilizadas debido a su simplicidad, velocidad de acceso y organización.

1. Almacenar grandes cantidades de datos similares de forma estructurada

En lugar de declarar muchas variables individuales para representar datos similares (por ejemplo, las edades de todos los estudiantes en una clase), los arreglos permiten agruparlos en una sola estructura, facilitando su manejo, visualización y mantenimiento.

Ejemplo práctico: Si una lista de 100 temperaturas diarias fuera declarada sin arreglos, implicaría declarar 100 variables. Con un solo arreglo, se puede manejar toda la lista usando un solo identificador.

2. Ahorrar memoria y simplificar el código

Usar arreglos permite escribir menos código, reducir errores y reutilizar estructuras en funciones o ciclos. También se aprovecha mejor la memoria, ya que los datos están agrupados contiguamente y pueden ser manipulados con ciclos de forma ordenada.

Ejemplo práctico: En vez de copiar y pegar líneas de código para procesar datos repetitivos, puedes usar un for para recorrer un arreglo y aplicar operaciones automáticamente.

3. Recorrer elementos fácilmente con estructuras de control (for, while, foreach)

Los arreglos son especialmente útiles cuando se necesita procesar cada elemento de una colección, ya sea para calcular promedios, buscar valores, ordenar, filtrar, etc. Los ciclos permiten acceder a cada índice del arreglo con facilidad.

Ejemplo práctico: Contar cuántos números en un arreglo son mayores a 10:

```
int contador = 0;
for (int i = 0; i < numeros.length; i++) {
    if (numeros[i] > 10) {
        contador++;
    }
}
```

4. Acceder o modificar elementos directamente por su posición (índice)

Cada elemento de un arreglo puede ser accedido o modificado directamente usando su índice. Esto permite trabajar con datos específicos de manera rápida y precisa.

5. Facilitar operaciones masivas y procesamiento de datos

Los arreglos permiten aplicar algoritmos de forma eficiente sobre grandes volúmenes de información: búsqueda binaria, ordenamiento (burbuja, quicksort), suma total, generación de estadísticas, etc.

Ejemplo 1: Arreglo de int (enteros) – Calificaciones de un estudiante

```
int[] calificaciones = {85, 90, 78, 92, 88};
```

Se está creando un arreglo que almacena cinco calificaciones. Cada calificación es un número entero (int). En vez de tener cinco variables (nota1, nota2, etc.), usamos un solo arreglo llamado calificaciones.

Ejemplo 2: Arreglo de char – Letras de una palabra

```
char[] letras = {'H', 'O', 'L', 'A'};
```

Este arreglo guarda caracteres individuales, que juntos forman la palabra "HOLA".

Ejemplo 3: Arreglo de float – Precios en una tienda

```
float[] precios = {1.50f, 2.75f, 3.00f, 0.99f};
```

Cada valor decimal representa el precio de un producto. El sufijo f indica que son tipo float.

Ejemplo 4: Arreglo de boolean – Estado de interruptores

```
for (int i = 0; i < interruptores.length; i++) {
    if (interruptores[i]) {
        System.out.println("Interruptor " + i + " está encendido");
    } else {
        System.out.println("Interruptor " + i + " está apagado");
    }
}
```

Conclusión

Los arreglos de tipos de datos simples o primitivos son fundamentales en programación porque permiten trabajar de forma ordenada y eficiente con múltiples valores relacionados. Son ideales para representar conjuntos homogéneos de datos como listas de calificaciones, temperaturas, estados de sensores, caracteres de una palabra, entre otros.

Referencias

Alberca, A. S. (n.d.). *Tipos de Datos Primitivos Simples*. Aprende con Alf. Retrieved June 18, 2025, from https://aprendeconalf.es/docencia/python/manual/tipos-datos-simples/

Programación, E., & la mínima información que se tiene en un sistema., U. E. de D. es U. F. de O. un C. de D. E. C. el O. de F. su M. U. D. E. es. (n.d.). *DDEFINICIONES BÁSICAS DDE LAS ESTRUCTURAS DDE DDATOS*. Edu.Pe. Retrieved June 18, 2025, from http://biblioteca.uns.edu.pe/saladocentes/archivoz/curzoz/estructuradedatosarraysunidimensionales.pdf

Tipos de datos y estructuras en JavaScript. (n.d.). MDN Web Docs. Retrieved June 18, 2025, from https://developer.mozilla.org/es/docs/Web/JavaScript/Guide/Data_structures

- He aprendido que los arreglos son fundamentales para organizar y manipular datos de forma ordenada y eficiente. Antes podía resolver problemas con muchas variables

individuales, pero ahora entiendo que con un solo arreglo puedo manejar grandes cantidades de información, lo cual mejora la legibilidad y el mantenimiento del código.

- Al trabajar con arreglos en distintos ejemplos (calificaciones, precios, sensores), reforcé mi capacidad para usar estructuras repetitivas como ciclos for y condicionales. Esto me permitió desarrollar soluciones más dinámicas y prácticas, aplicables en proyectos reales, como sistemas escolares o automatización básica.
- Este tema me permitió ver a los arreglos no solo como una herramienta básica, sino como un pilar para construir estructuras más avanzadas como matrices, listas, pilas o colas. Aprender bien cómo funcionan los arreglos es un paso esencial para seguir desarrollándome en programación orientada a objetos y en algoritmos más elaborados.