**UNIVERSIDAD**

**NACIONAL DEL ALTIPLANO – PUNO**

**ESCUELA PROFESIONAL DE INGENIERIA DE SISTEMAS**

**CURSO: ESTRUCTURAS Y ALGORITMO DE DATOS**

Imagen que contiene Flecha

Descripción generada automáticamente

**CURSO:**

ESTRUCTURAS Y ALGORITMOS DE DATOS

**TEMA:**

PRÁCTICA 1 - AED - INTRODUCCION

**NOMBRES:**

YEFFERSON MIRANDA JOSEC

**PUNO – PERU**

2025

**Actividad En El Aula**

**¿Qué es un algoritmo? ¿Dónde los usamos?**

Un **algoritmo** es un conjunto de pasos ordenados que seguimos para resolver un problema o realizar una tarea. Es como una receta que nos indica qué hacer primero, qué hacer después y cómo llegar al resultado que queremos.

Usamos algoritmos todo el tiempo, aunque no siempre nos damos cuenta. Por ejemplo, cuando seguimos una receta de cocina, estamos siguiendo un algoritmo. También lo hacemos al resolver un problema de matemáticas o al seguir las instrucciones para armar un mueble.

En el mundo de la tecnología, los algoritmos están por todas partes. Cuando buscamos algo en Google, hay un algoritmo que decide qué resultados mostrar primero. O cuando usamos el GPS del celular, un algoritmo calcula la mejor ruta para llegar a nuestro destino.

En resumen, los algoritmos nos ayudan a hacer las cosas de forma ordenada, paso a paso, para llegar a una solución o completar una tarea con éxito

**Pasos para preparar un café, registrarse en un**

**Sistema, en Pseudocódigo.**

***1. Preparar un café***

**Inicio**

Hervir\_agua()

Colocar\_café\_en\_el\_filtro()

Verter\_agua\_caliente\_sobre\_el\_café()

Esperar\_que\_el\_café\_se\_filtre()

Servir\_el\_café\_en\_una\_taza()

**Si** desea\_azúcar

Agregar\_azúcar()

**FinSi**

**Si** desea\_leche

Agregar\_leche()

FinSi

**Fin**

***2. Registrarse en un sistema (en pseudocódigo)***

**Inicio**

Mostrar\_formulario\_de\_registro()

**Leer** nombre\_usuario

**Leer** correo\_electrónico

**Leer** contraseña

**Leer** confirmar\_contraseña

Si contraseña **==** confirmar\_contraseña Entonces

Guardar\_datos\_en\_la\_base\_de\_datos()

Mostrar "Registro exitoso"

Sino

Mostrar "Las contraseñas no coinciden"

FinSi

Fin

**Practica Programada**

1. ***Diseñar un algoritmo que determine si una cadena es un palíndromo (sin usar***

***funciones integradas).***

include <iostream>

**using** **namespace** std;

**bool** **esPalindromo**(string cadena) {

**int** l = **0**;

**while** (cadena[longitud] != '\0') {

longitud++;

}

**int** i = **0**, j = l - **1**;

**while** (i < j) {

**if** (cadena[i] != cadena[j]) {

**return** false;

}

i++;

j--;

}

**return** true;

}

1. ***Elaborar un algoritmo que recorra un arreglo de N elementos y determine el***

***Segundo valor más alto sin ordenarlo.***

#include <iostream>

**void** **encontrarSegundoMayor**(**int** arr[], **int** N) {

**if** (N < **2**) {

std::cout << "No hay suficientes elementos para encontrar el segundo mayor." << std::endl;

**return**;

}

**int** max1 = arr[**0**], max2 = arr[**0**];

**for** (**int** i = **1**; i < N; i++) {

**if** (arr[i] > max1) {

max2 = max1;

max1 = arr[i];

} **else** **if** (arr[i] > max2 && arr[i] != max1) {

max2 = arr[i];

}

}

**if** (max1 == max2) {

std::cout << "No hay un segundo mayor único en el arreglo." << std::endl;

} **else** {

std::cout << "El segundo valor más alto es: " << max2 << std::endl;

}

}

***3. Simular una calculadora de tarifas para transporte público basada en el tipo de***

***Usuario y distancia recorrida.***

***o Para cada algoritmo:***

***▪ Escribir en pseudocódigo.***

***▪ Determinar su complejidad temporal (O(n), O(1), etc.).***

***▪ Explicar por qué usarías un arreglo, lista o estructura específica para***

***Implementarlo.***

INICIO

DEFINIR TARIFAS\_BASE COMO DICCIONARIO {

"Adulto": 1.50,

"Estudiante": 1.00,

"Senior": 0.75

}

DEFINIR TARIFA\_POR\_KM COMO 0.25

FUNCION calcular\_tarifa(tipo\_usuario, distancia)

SI tipo\_usuario NO ESTÁ EN TARIFAS\_BASE

MOSTRAR "Tipo de usuario inválido"

RETORNAR -1

FIN SI

tarifa\_base ← TARIFAS\_BASE[tipo\_usuario]

tarifa\_total ← tarifa\_base + (distancia \* TARIFA\_POR\_KM)

RETORNAR tarifa\_total

FIN FUNCION

// Prueba de la función

LEER tipo\_usuario

LEER distancia

resultado ← calcular\_tarifa(tipo\_usuario, distancia)

SI resultado >= 0

MOSTRAR "Tarifa total: ", resultado

FIN SI

FIN

La complejidad de calcular\_tarifa es O(1) (tiempo constante), porque:

Acceder a un elemento en un diccionario por clave (TARIFAS\_BASE[tipo\_usuario]) es O(1). Realizar una suma y una multiplicación también es O(1).

Diccionario (TARIFAS\_BASE) ¿Por qué? Permite acceso rápido a la tarifa base según el tipo de usuario en O(1).