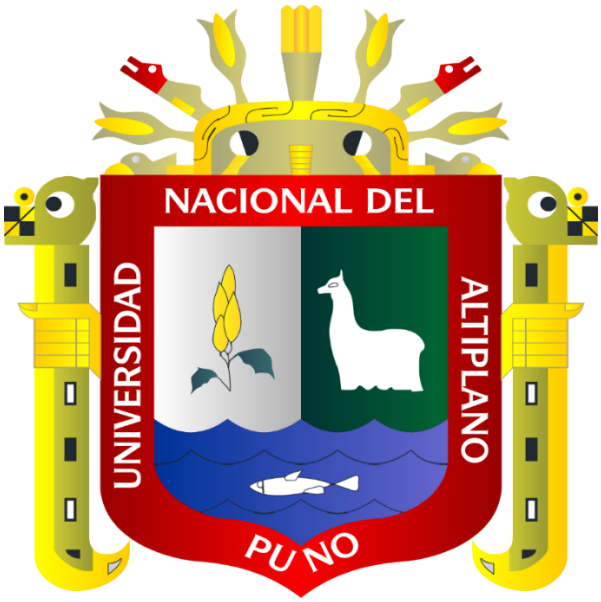
UNIVERSIDAD NACIONAL DEL ALTIPLANO



FACULTAD DE INGENIERIA MECANICA ELECTRICA, ELECTRONICA Y SISTEMAS

ESCUELA PROFESIONAL DE INGENIERIA DE SISTEMAS

CURSO:
ALGORITMOS Y ESTRUCTURAS DE DATOS
PRESENTADO POR:
DAYVID JUNIOR SUAQUITA CCANCCAPA
"Practica 1"

SEMESTRE: IV

PUNO – PERU
10 de abril de 2025

1. Diseñar un algoritmo que determine si una cadena es un palíndromo (sin usar funciones integradas).

La complejidad de este algoritmo es lineal puesto que aumenta en proporción al tamaño del arreglo.

Se usa un arreglo de caracteres porque nos permite acceder directamente a cualquier índice, lo cual es esencial para comparar caracteres desde ambos extremos eficientemente.

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

bool esPalindromo(int arreglo[], int tamaño) {
    int inicio = 0;
    int fin = tamaño - 1;

    while (inicio < fin) {
        if (arreglo[inicio] != arreglo[fin]) {
            return false;
        }
        inicio++;
        fin--;
    }
    return true;
}

int main() {
    int palindromo2[] = {5, 4, 4, 5};
    cout << "{5, 4, 4, 5} es palindromo? "
         << (esPalindromo(palindromo2, 4) ? "Sí" : "No") <<
endl;


    return 0;
}
```

Ejercio 1:

```
bool esPalindromo(int arreglo[], int tamano){
    int inicio=0;
    int fin = tamano-1;
    while(inicio < fin){
      if (arreglo[inicio] != arreglo[fin]) {
       return false;
      }
       inicio++
    } fin--
    return true;
}
```

Prueba array = {5, 4, 4, 5}

array	tamaño	inicio	fin	Palindromo?
5, 4, 4, 5	4	0	3	true
5, 4, 4, 5	4	1	2	tue
5, 4, 4, 5	4	2	1	true

si es un palindromo

2. Elaborar un algoritmo que recorra un arreglo de N elementos y determine el segundo valor más alto sin ordenarlo.

La complejidad temporal del algoritmo para encontrar el segundo valor más grande es O(n), donde n es el número de elementos en el array.

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

int segundoMasGrande(int arreglo[], int tamano) {
    if (tamaño < 2) {
        cout << "Error" << endl;
        return -1;
    }

    int max1 = 0;
    int max2 = 0;

    for (int i = 0; i < tamano; i++) {
        if (arreglo[i] > max1) {
            max2 = max1;
            max1 = arreglo[i];
        } else if (arreglo[i] > max2 && arreglo[i] != max1) {
            max2 = arreglo[i];
        }
    }
    return max2;
}

int main() {
    int arr1[] = {4,5,10,6,15,7};

    cout << "Segundo mayor = "<< segundoMasGrande(arr1, 6) << endl;

    return 0;
}
```

Ejercicio 2:

```
int segundo_Mas_grande (int arreglo[], int tamaño){
    int max1=0;
    int max2=0;
    for(int i=0; i< tamaño; i++){
        if (arreglo[i] > max1){
            max2= max1;
            max1= arreglo[i];
        }
        else if (arreglo[i] > max2 && arreglo[i] != max1)
            max2 = arreglo[i];
        }
    }
    return max2;
}
```

arreglo	tamano	max1	max2	i
4,5,10,6,7	6	0	0	0
	6	4	0	0
	6	5	4	1
	6	10	5	2
	6	10	6	3
	6	15	10	4
	6	50	10	5

retorna 10

Ejercicio 3:

Simular una calculadora de tarifas para transporte público basada en el tipo de usuario y distancia recorrida.

La complejidad del algoritmo es constante(O(1)) porque realiza operaciones fijas sin bucles, independientemente de la entrada.

```
#include <iostream>
#include <cmath>
using namespace std;

double calcularTarifa(int tipoUsuario, double distancia) {
    double tarifaBase;

    if (tipoUsuario == 1) {
        tarifaBase = 0.40;
    }
    else if (tipoUsuario == 2) {
        tarifaBase = 0.70;
    }
    else {
        tarifaBase = 1.00;
    }
    if (distancia <= 1.0) {
        return tarifaBase;
    }
    else {
        double kmExtra = distancia - 1.0;
        int segmentos = ceil(kmExtra / 0.5);
        return tarifaBase + (segmentos * 0.5);
    }
}

int main() {
    int usuario1 = 2;
    double distancia1 = 2.3;
    double tarifa1 = calcularTarifa(usuario1, distancia1);

    cout << "Caso 1:" << endl;
    cout << "Tipo: Universitario" << endl;
    cout << "Distancia: " << distancia1 << " km" << endl;
    cout << "Tarifa final: S/" << tarifa1 << endl << endl;

    int usuario2 = 3;
    double distancia2 = 0.8;
    double tarifa2 = calcularTarifa(usuario2, distancia2);

    cout << "Caso 2:" << endl;
    cout << "Tipo: Adulto" << endl;
    cout << "Distancia: " << distancia2 << " km" << endl;
    cout << "Tarifa final: S/" << tarifa2 << endl;

    return 0;
}
```

Ejercicio 3:

```
double calcularTarifa(int tipoUsuario, double distancia) {
  double tarifaBase;
  if (tipoUsuario==1) { tarifaBase=0.40;}
  else if (tipoUsuario==2) { tarifaBase=0.70;}
  else {tarifaBase=1.00;}
  if (distancia <=1.0) {return tarifaBase;}
  else {
  double kmExtra= distancia-1.0;
  int seg= ceil(kmExtra/0.5);
   return tarifaBase+(seg*0.5);
```

Caso Prueba : Adulto, 0.8 km

tipo Usuario	distancia	TarifaBase	km extra	seg
3	0.8	1.00		

se retorna S/.1.00

Caso 2: Universitario 2.3 km

Tipo Usuario	Distancia	Tarifa Base	Km extra	seg
2	2.3	0.7	1.3	3

Calcular tarifa = 0.7 + 1.5 = S/2.20

3*5 : aumenta 0.5 por cada 500 metros