INSTITUTO POLITÉCNICO NACIONAL

ESCUELA SUPERIOR DE CÓMPUTO

ACTIVIDAD: Ejercicio de laboratorio 1

ALUMNO: Parra Hernández Jorge Antonio

UNIDAD DE APRENDIZAJE: Paradigmas de Programación

PROFESOR: García Floriano Andrés

GRUPO: 3CV1

Ciudad de México, 05 de marzo del 2024.

EJERCICIO DE LABORATORIO 1

CÓDIGO DEL PROGRAMA

En C:

```
#include<stdio.h>
#include<stdlib.h>
#include<time.h>

#define size 1000

void numAleatorios(int arreglo[]);
void imprimirArreglo(int arreglo[]);
int busquedaSecuencial(int arreglo[], int valor, int tam);
void ordenarArreglo(int arreglo[], int tam);

int main(){

    int arreglo[size];
    int numeroBuscado;

    numAleatorios(arreglo);

    printf("\nEl arreglo generado es: \n");
    imprimirArreglo(arreglo);

    printf("\n\nIngrese el numero que desee buscar (menor a 1000): ");
    scanf("%d", &numeroBuscado);

    //Busqueda Secuencial

    int indice= busquedaSecuencial(arreglo, numeroBuscado, size);

    if(indice != -1){

        printf("\nEl valor %d se encuentra en el indice %d.\n", numeroBuscado,
indice);
    }

    else{
        printf("\nEl valor %d no se encuentra en el arreglo.\n", numeroBuscado);
    }

    ordenarArreglo(arreglo, size);

    printf("\nArreglo ordenado: ");
```

```
    imprimirArreglo(arreglo);

    printf("\nIngrese nuevamente el numero que desea buscar en el arreglo
ordenado: ");
    scanf("%d", &numeroBuscado);

    //Busqueda Secuencial en el arreglo ordenado

    indice= busquedaSecuencial(arreglo, numeroBuscado, size);

    if(indice != -1){

        printf("\n\nEl valor %d se encuentra en el indice %d del arreglo ordenado.\n",
numeroBuscado, indice);
    }

    else{

        printf("\n\nEl valor %d no se encuentra en el arreglo ordenado.\n",
numeroBuscado);
    }

return 0;
}

// Funcion que genera numeros aleatorios

void numAleatorios(int arreglo[]){
    clock_t inicio= clock(); // Inicia medicion de tiempo

    srand(time(NULL));
    for(int i=0; i<size; i++){

        arreglo[i]= rand() % 1000;
    }

    clock_t fin= clock(); // Finaliza medicion de tiempo
    double tiempo= (double)(fin - inicio) / CLOCKS_PER_SEC; // Calcula el tiempo
en segundos
    printf("\nTiempo de generacion del arreglo: %.6f seg\n", tiempo);
}

// Funcion que imprime el arreglo

void imprimirArreglo(int arreglo[]){
    clock_t inicio= clock();
```

```
    for(int i=0; i<size; i++){

        printf("%d, ", arreglo[i]);
    }

    printf("\n");

    clock_t fin= clock();
    double tiempo= (double)(fin - inicio) / CLOCKS_PER_SEC;
    printf("\nTiempo de impresion del arreglo: %.6f seg\n", tiempo);
}

//Funcion de busqueda secuencial

int busquedaSecuencial(int arreglo[], int valor, int tam){
    clock_t inicio= clock();

    for(int i=0; i<tam; i++){

        if(arreglo[i] == valor){

            return i;
        }

    }

    clock_t fin= clock();
    double tiempo= (double)(fin - inicio) / CLOCKS_PER_SEC;
    printf("\nTiempo de busqueda secuencial: %.6f seg\n", tiempo);

    return -1;
}

// Funcion que ordena el arreglo generado anteriormente

void ordenarArreglo(int arreglo[], int tam){
    clock_t inicio= clock();

    for(int i = 0; i < tam - 1; i++){

        int min_idx = i;

        for(int j = i + 1; j < tam; j++){

            if (arreglo[j] < arreglo[min_idx]) {
                min_idx = j;
            }
```

```
        }

        int temp = arreglo[min_idx];
        arreglo[min_idx] = arreglo[i];
        arreglo[i] = temp;
    }

    clock_t fin= clock();
    double tiempo= (double)(fin - inicio) / CLOCKS_PER_SEC;
    printf("\nTiempo de ordenacion del arreglo: %.6f seg\n", tiempo);

}
```

En Python:

```
import random
import time

size = 1000

# Función que genera números aleatorios
def num_aleatorios():
    inicio = time.time() # Inicia medicion de tiempo

    arreglo = [random.randint(0, 999) for _ in range(size)]

    fin = time.time() # Finaliza medicion de tiempo
    tiempo = fin - inicio # Calcula el tiempo en segundos
    print("\nTiempo de generacion del arreglo:", tiempo, "seg")

    return arreglo

# Función que imprime el arreglo
def imprimir_arreglo(arreglo):
    inicio = time.time()

    print(", ".join(map(str, arreglo)))

    fin = time.time()
    tiempo = fin - inicio
    print("\nTiempo de impresion del arreglo:", tiempo, "seg")

# Función de búsqueda secuencial
def busqueda_secuencial(arreglo, valor):
    inicio = time.time()

    for i, num in enumerate(arreglo):
```

```
        if num == valor:
            fin = time.time()
            tiempo = fin - inicio
            print("\nTiempo de busqueda secuencial:", tiempo, "seg")
            return i

    fin = time.time()
    tiempo = fin - inicio
    print("\nTiempo de busqueda secuencial:", tiempo, "seg")

    return -1

# Función que ordena el arreglo
def ordenar_arreglo(arreglo):
    inicio = time.time()

    for i in range(len(arreglo) - 1):
        min_idx = i
        for j in range(i + 1, len(arreglo)):
            if arreglo[j] < arreglo[min_idx]:
                min_idx = j
        arreglo[i], arreglo[min_idx] = arreglo[min_idx], arreglo[i]

    fin = time.time()
    tiempo = fin - inicio
    print("\nTiempo de ordenacion del arreglo:", tiempo, "seg")

def main():
    arreglo = num_aleatorios()

    print("\nEl arreglo generado es:")
    imprimir_arreglo(arreglo)

    numero_buscado = int(input("\nIngrese el número que desea buscar (menor a
1000): "))

    indice = busqueda_secuencial(arreglo, numero_buscado)
    if indice != -1:
        print(f"\nEl valor {numero_buscado} se encuentra en el índice {indice}.")
    else:
        print(f"\nEl valor {numero_buscado} no se encuentra en el arreglo.")

    ordenar_arreglo(arreglo)

    print("\nArreglo ordenado:")
    imprimir_arreglo(arreglo)
```

```
    numero_buscado = int(input("\nIngrese nuevamente el número que desea
buscar en el arreglo ordenado: "))

    indice = busqueda_secuencial(arreglo, numero_buscado)
    if indice != -1:
        print(f"\nEl valor {numero_buscado} se encuentra en el índice {indice} del
arreglo ordenado.")
    else:
        print(f"\nEl valor {numero_buscado} no se encuentra en el arreglo ordenado.")

if __name__ == "__main__":
    main()
```

DISEÑO Y DESARROLLO

Ante todo, considero que lo primero que se debe hacer para desarrollar un programa es leer por completo el problema que se nos plantea, analizar lo que se requiere y como podemos solucionarlo y hacer uso de nuestras habilidades y conocimientos para enfrentar los problemas planteados.

Por tanto, el diseño y desarrollo del programa consistió en las siguientes partes:

1. **Función que genere un arreglo aleatorio:** Se realizo una función que genere números aleatorios a partir de las funciones srand y random en C y Python respectivamente.

   Resultados:

   En C:

```
// Funcion que genera numeros aleatorios

void numAleatorios(int arreglo[]){
    clock_t inicio= clock(); // Inicia medicion de tiempo

    srand(time(NULL));
    for(int i=0; i<size; i++){

        arreglo[i]= rand() % 1000;
    }

    clock_t fin= clock(); // Finaliza medicion de tiempo
    double tiempo= (double)(fin - inicio) / CLOCKS_PER_SEC; // Calcula el
tiempo en segundos
    printf("\nTiempo de generacion del arreglo: %.6f seg\n", tiempo);
}
```

En Python:

```
# Función que genera números aleatorios
def num_aleatorios():
    inicio = time.time() # Inicia medicion de tiempo

    arreglo = [random.randint(0, 999) for _ in range(size)]

    fin = time.time() # Finaliza medicion de tiempo
    tiempo = fin - inicio # Calcula el tiempo en segundos
    print("\nTiempo de generacion del arreglo:", tiempo, "seg")

    return arreglo
```

2. **Función que imprima el contenido de los arreglos:** En C, para imprimir el arreglo, se hizo uso de un ciclo (en este caso for) para que pudiera imprimir todos los valores del arreglo generado. En Python se hizo uso de funciones para imprimir el arreglo.

   Resultados:

   En C:

```
// Funcion que imprime el arreglo

void imprimirArreglo(int arreglo[]){
    clock_t inicio= clock();

    for(int i=0; i<size; i++){

        printf("%d, ", arreglo[i]);
    }

    printf("\n");

    clock_t fin= clock();
    double tiempo= (double)(fin - inicio) / CLOCKS_PER_SEC;
    printf("\nTiempo de impresion del arreglo: %.6f seg\n", tiempo);
}
```

   En Python:

```
# Función que imprime el arreglo
def imprimir_arreglo(arreglo):
    inicio = time.time()

    print(", ".join(map(str, arreglo)))

    fin = time.time()
    tiempo = fin - inicio
```

```
    print("\nTiempo de impresion del arreglo:", tiempo, "seg")
```

3. **Función de búsqueda secuencial:** Se implemento el algoritmo de búsqueda secuencial visto en cursos anteriores, como Algoritmos y Estructuras de Datos.

   Resultados:

   En C:

```
//Funcion de busqueda secuencial

int busquedaSecuencial(int arreglo[], int valor, int tam){
    clock_t inicio= clock();

    for(int i=0; i<tam; i++){

        if(arreglo[i] == valor){

            return i;
        }

    }

    clock_t fin= clock();
    double tiempo= (double)(fin - inicio) / CLOCKS_PER_SEC;
    printf("\nTiempo de busqueda secuencial: %.6f seg\n", tiempo);

    return -1;
}
```

   En Python:

```
# Función de búsqueda secuencial
def busqueda_secuencial(arreglo, valor):
    inicio = time.time()

    for i, num in enumerate(arreglo):
        if num == valor:
            fin = time.time()
            tiempo = fin - inicio
            print("\nTiempo de busqueda secuencial:", tiempo, "seg")
            return i

    fin = time.time()
    tiempo = fin - inicio
    print("\nTiempo de busqueda secuencial:", tiempo, "seg")

    return -1
```

4. **Función que ordene el arreglo:** En ambos lenguajes, se utilizó un ciclo anidado for y un if que validara los valores del arreglo y, saliendo del ciclo, intercambiara valores hasta que todos los números del arreglo fueran ordenados.

   Resultados:

   En C:

```
// Funcion que ordena el arreglo generado anteriormente

void ordenarArreglo(int arreglo[], int tam){
    clock_t inicio= clock();

    for(int i = 0; i < tam - 1; i++){

        int min_idx = i;

        for(int j = i + 1; j < tam; j++){

            if (arreglo[j] < arreglo[min_idx]) {
                min_idx = j;
            }
        }

        int temp = arreglo[min_idx];
        arreglo[min_idx] = arreglo[i];
        arreglo[i] = temp;
    }

    clock_t fin= clock();
    double tiempo= (double)(fin - inicio) / CLOCKS_PER_SEC;
    printf("\nTiempo de ordenacion del arreglo: %.6f seg\n", tiempo);

}
```

   En Python:

```
# Función que ordena el arreglo
def ordenar_arreglo(arreglo):
    inicio = time.time()

    for i in range(len(arreglo) - 1):
        min_idx = i
        for j in range(i + 1, len(arreglo)):
            if arreglo[j] < arreglo[min_idx]:
                min_idx = j
        arreglo[i], arreglo[min_idx] = arreglo[min_idx], arreglo[i]
```

```
fin = time.time()
tiempo = fin - inicio
print("\nTiempo de ordenacion del arreglo:", tiempo, "seg")
```

5. **Tiempo de ejecución:** En cada lenguaje se hizo uso de funciones que nos ayudan a medir los tiempos de ejecución. En C se utilizó clock() y Python se utilizó time().

   Resultados:

   En C:

```
clock_t inicio= clock(); // Inicia medicion de tiempo
clock_t fin= clock(); // Finaliza medicion de tiempo
double tiempo= (double)(fin - inicio) / CLOCKS_PER_SEC; // Calcula el tiempo en segundos
```

   En Python:

```
inicio = time.time() # Inicia medicion de tiempo
fin = time.time() # Finaliza medicion de tiempo
tiempo = fin - inicio # Calcula el tiempo en segundos
```

CONCLUSIONES

Fue interesante desarrollar los programas porque reformo un poco mis conocimientos de C con funciones, algoritmos y sus respectivas implementaciones. Igualmente, me ayudo a conocer más de Python, ya que personalmente no he trabajado con este lenguaje y, como primeras impresiones, facilita muchas cosas en comparación con C y se reducen en buena parte las líneas de código implementadas.