PRÁCTICA 2

REPORTE

PARADIGMAS DE PROGRAMACIÓN

3CV1

ELIZALDE HERNÁNDEZ ALAN
SOLARES VELASCO ARTURO MISAEL
SOLIS LUGO MAYRA
REYES RUIZ YOSELYN ESTEFANI

DESARROLLO

Función generate_list_random(int n, int *lista)

- **Esta función toma dos argumentos: n, que representa la longitud de la lista, y lista, que es un puntero a un array de enteros.**
- **Se utiliza srand(time(NULL)) para inicializar la semilla del generador de números aleatorios.**
- **Luego, se genera n números aleatorios en el rango [-100, 100] y se almacenan en el array lista.**

```c
int generate_list_random(int n, int *lista) {
    srand(time(NULL));

    for (int i = 0; i < n; i++) {
        lista[i] = rand() % 200 - 100;
    }
    return 0;
}
```

```c
int minv2(int *lista, int n) {
    int mini = lista[0];
    for (int i = 1; i < n; i++) {
        if (lista[i] < mini) {
            mini = lista[i];
        }
    }
    return mini;
}

int maxv2(int *lista, int n) {
    int maxi = lista[0];
    for (int i = 1; i < n; i++) {
        if (lista[i] > maxi) {
            maxi = lista[i];
        }
    }
    return maxi;
}
```

Funciones minv2 y maxv2

- Estas funciones calculan el valor mínimo (**minv2**) y máximo (**maxv2**) en un array de enteros.
- Ambas funciones toman dos argumentos: **lista**, que es un puntero al array de enteros, y **n**, que representa la longitud del array.
- Se inicializan **mini** y **maxi** con el primer elemento del array.
- Luego, se recorre el array para encontrar el mínimo (**minv2**) o el máximo (**maxv2**), actualizando mini o maxi si se encuentra un valor más pequeño o más grande, respectivamente.

Función sumar_listas

- Esta función toma tres argumentos: **lista1** y **lista2**, que son punteros a arrays de enteros que se sumarán, **n**, que es la longitud de los arrays, y **suma_listas**, que es un puntero a un array de enteros donde se almacenarán los resultados de la suma.
- Suma los elementos correspondientes de **lista1** y **lista2** y los almacena en **suma_listas**.

```c
void sumar_listas(int *lista1, int *lista2, int n, int *suma_listas) {
    for (int i = 0; i < n; i++) {
        suma_listas[i] = lista1[i] + lista2[i];
    }
}
```

DESARROLLO

Main

Paso 1: Declaración de variables y generación de lista1

- Se declara **n** como una constante que define la longitud de las listas.
- Se declaran tres arrays de enteros: **lista1**, **lista2** y **suma_listas**.
- Se llama a **generate_list_random** para generar números aleatorios y llenar **lista1**.

Paso 2: Generación de lista2

- Se vuelve a inicializar la semilla del generador de números aleatorios usando **srand(time(NULL))**.
- Se llama a **generate_list_random** para generar números aleatorios y llenar **lista2**.
- Se utiliza un bucle **while** para asegurarse de que los números generados en **lista2** sean diferentes de los de **lista1**.

Paso 3: Impresión de las listas generadas

- Se imprime **lista1** y **lista2** en la consola.

Paso 4: Cálculo y impresión de los mínimos y máximos de las listas

- Se utiliza la función **minv2** para calcular el mínimo de **lista1** y **lista2**, y se imprime el resultado.
- Se utiliza la función **maxv2** para calcular el máximo de **lista1** y **lista2**, y se imprime el resultado.

Paso 5: Suma de las listas y cálculo del tiempo de ejecución

- Se mide el tiempo de inicio de la suma con **clock()**.
- Se llama a la función **sumar_listas** para sumar **lista1** y **lista2**, y se almacena el resultado en **suma_listas**.
- Se mide el tiempo de finalización de la suma con **clock()**.
- Se calcula el tiempo total de ejecución de la suma.

Paso 6: Impresión de la suma de las listas y el tiempo de ejecución

- Se imprime el resultado de la suma de **lista1** y **lista2**.
- Se calcula la suma total de los elementos de **suma_listas** y se imprime.
- Se imprime el tiempo total de ejecución del programa.

Paso 7: Fin del programa

- Se retorna 0 para indicar que el programa finalizó correctamente.

```
int main() {
    const int n = 10;
    int lista1[n], lista2[n], suma_listas[n];
    generate_list_random(n, lista1);

    // Generar lista2 con números aleatorios diferentes a lista1
    srand(time(NULL));
    generate_list_random(n, lista2);
    for (int i = 0; i < n; i++) {
        while (lista2[i] == lista1[i]) {
            lista2[i] = rand() % 200 - 100;
        }
    }
```

```c
    // Imprime las listas generadas
    printf("Lista 1: ");
    for (int i = 0; i < n; i++) {
        printf("%d ", lista1[i]);
    }
    printf("\n");
    printf("Lista 2: ");
    for (int i = 0; i < n; i++) {
        printf("%d ", lista2[i]);
    }
    printf("\n");

    // Calcula y muestra los mínimos y máximos de las listas
    printf("\nEl mínimo de lista 1 es %d\n", minv2(lista1, n));
    printf("El máximo de lista 1 es %d\n\n", maxv2(lista1, n));
    printf("El mínimo de lista 2 es %d\n", minv2(lista2, n));
    printf("El máximo de lista 2 es %d\n", maxv2(lista2, n));
```

```c
    // Suma las listas y muestra el resultado
    clock_t start_time = clock(); // Calcular el tiempo
    sumar_listas(lista1, lista2, n, suma_listas);
    clock_t end_time = clock();
    double execution_time = (double)(end_time - start_time) / CLOCKS_PER_SEC;

    printf("\nListas sumadas: ");
    for (int i = 0; i < n; i++) {
        printf("%d ", suma_listas[i]);
    }
    printf("\n");

    // Calcula la suma total de las listas sumadas
    int suma_total = 0;
    for (int i = 0; i < n; i++) {
        suma_total += suma_listas[i];
    }
    printf("Listas sumadas (total): %d\n", suma_total);
    printf("\nTiempo de ejecución: %f seconds\n", execution_time);
    return 0;
}
```