PRUEBA Nro. 1 TERCER PARCIAL

Enunciado

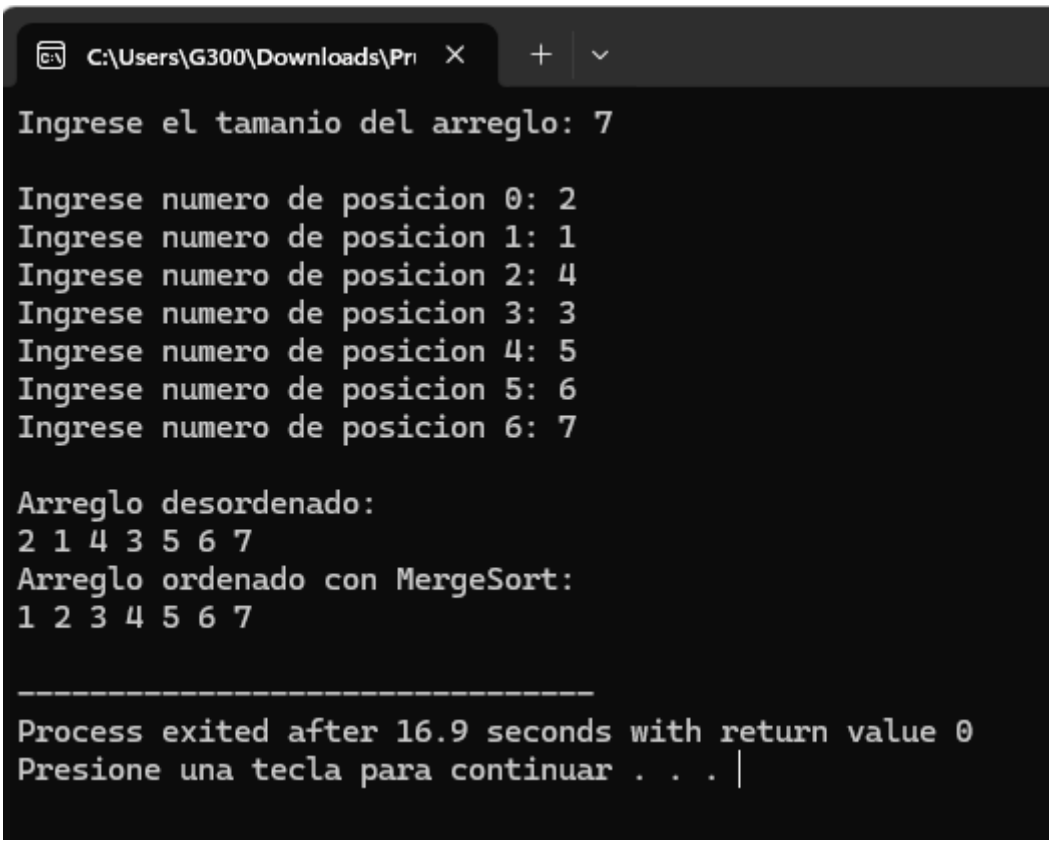
Implemente un programa en C++ que reciba un arreglo de números enteros de tamaño variable y realice su ordenamiento utilizando un proceso que permita dividir el problema en partes más pequeñas, resolver cada parte de manera independiente y combinar los resultados. El programa debe mostrar el arreglo original y el arreglo ordenado.

Código

```
/**
Implemente un programa en C++ que reciba un arreglo de números enteros de tamaño variable
y realice su ordenamiento utilizando un proceso que permita dividir el problema en partes
más pequeñas, resolver cada parte de manera independiente y combinar los resultados.
El programa debe mostrar el arreglo original y el arreglo ordenado.
**/
#include <iostream>
using namespace std;
class Ordenador {
public:
    void merge(int arr[], int inicio, int medio, int fin) {
        int n1 = medio - inicio + 1;
        int n2 = fin - medio;
        int* izquierda = new int[n1];
        int* derecha = new int[n2];
        for (int i = 0; i < n1; i++)
            izquierda[i] = arr[inicio + i];
        for (int j = 0; j < n2; j++)
            derecha[j] = arr[medio + 1 + j];
        int i = 0, j = 0, k = inicio;
        while (i < n1 && j < n2) {
            if (izquierda[i] <= derecha[j]) {
                arr[k] = izquierda[i];
                i++;
            } else {
                arr[k] = derecha[j];
                j++;
            }
            k++;
        }
        while (i < n1) {
            arr[k] = izquierda[i];
            i++;
            k++;
        }
        while (j < n2) {
            arr[k] = derecha[j];
            j++;
            k++;
        }
        delete[] izquierda;
        delete[] derecha;
    }
    void mergeSort(int arr[], int inicio, int fin) {
        if (inicio < fin) {
            int medio = inicio + (fin - inicio) / 2;
            mergeSort(arr, inicio, medio);
            mergeSort(arr, medio + 1, fin);
            merge(arr, inicio, medio, fin);
        }
    }
};
int main() {
    int arrtam;
    cout << "Ingrese el tamanio del arreglo: ";
    cin >> arrtam;
    cout << endl;
    int* arr = new int[arrtam];
    for (int i = 0; i < arrtam; i++) {
        cout << "Ingrese numero de posicion " << i << ": ";
        cin >> arr[i];
    }
    cout <<endl;
```

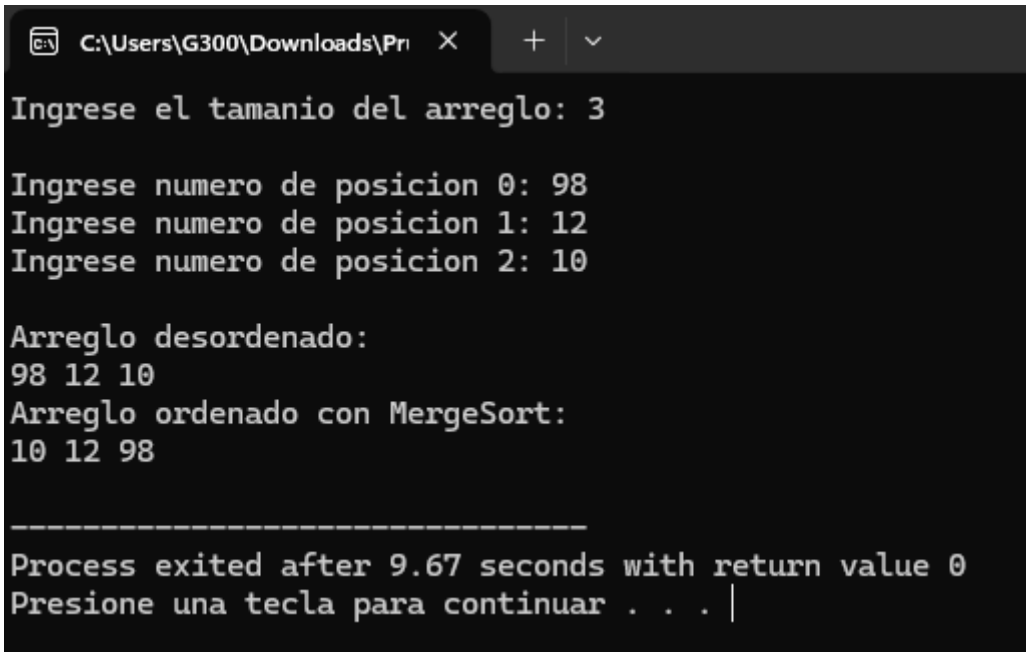
```
    cout << "Arreglo desordenado: " << endl;
    for (int j = 0; j < arrtam; j++) {
        cout << arr[j] << " ";
    }
    cout << endl;
    Ordenador ord;
    ord.mergeSort(arr, 0, arrtam - 1);

    cout << "Arreglo ordenado con MergeSort: " << endl;
    for (int k = 0; k < arrtam; k++) {
        cout << arr[k] << " ";
    }
    cout << endl;
    delete[] arr;
    return 0;
}
```

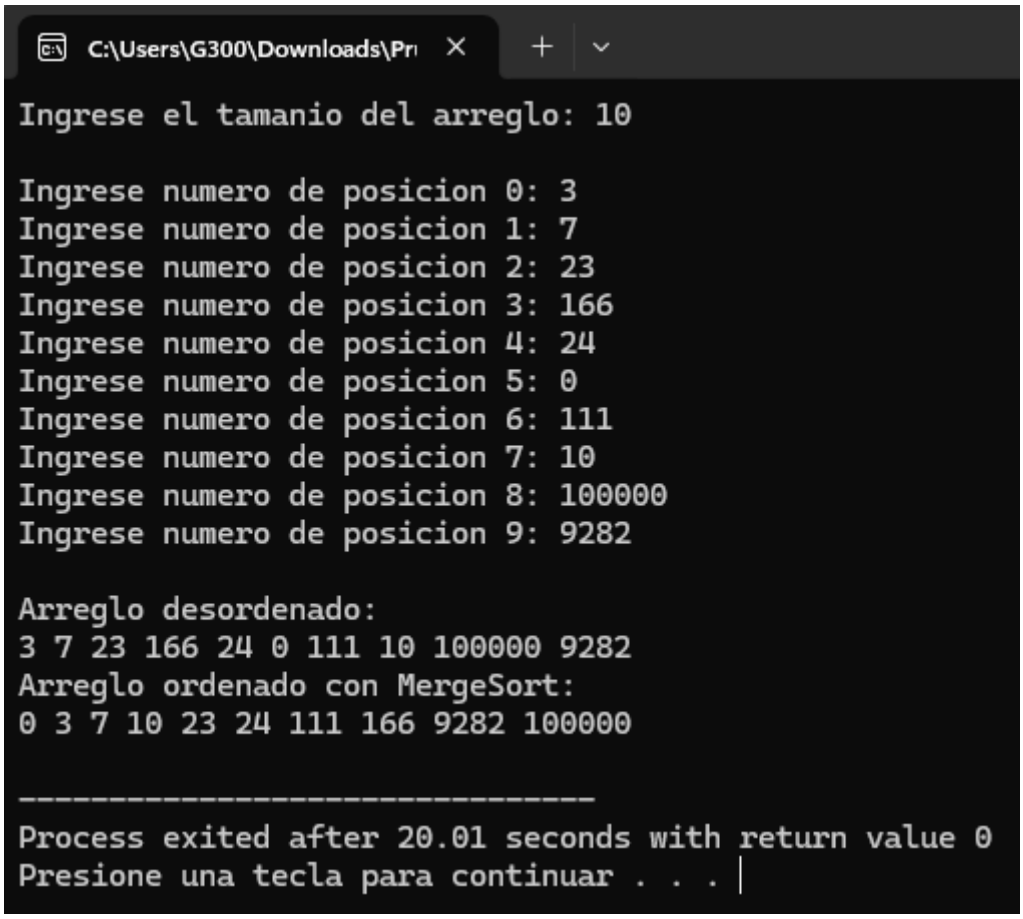
Compilación



Se puede evidenciar el uso de memoria dinámica al permitir al usuario determinar el tamaño del arreglo, además se evidencia el ordenamiento mostrando el arreglo antes y después de pasar por el algoritmo MergeSort.

A continuación, se presentan más evidencias con diferentes valores:


C:\Users\G300\Downloads\Pr
Ingrese el tamanio del arreglo: 3
Ingrese numero de posicion 0: 98
Ingrese numero de posicion 1: 12
Ingrese numero de posicion 2: 10
Arreglo desordenado:
98 12 10
Arreglo ordenado con MergeSort:
10 12 98
Process exited after 9.67 seconds with return value 0
Presione una tecla para continuar . . .


C:\Users\G300\Downloads\Pr
Ingrese el tamanio del arreglo: 10
Ingrese numero de posicion 0: 3
Ingrese numero de posicion 1: 7
Ingrese numero de posicion 2: 23
Ingrese numero de posicion 3: 166
Ingrese numero de posicion 4: 24
Ingrese numero de posicion 5: 0
Ingrese numero de posicion 6: 111
Ingrese numero de posicion 7: 10
Ingrese numero de posicion 8: 100000
Ingrese numero de posicion 9: 9282
Arreglo desordenado:
3 7 23 166 24 0 111 10 100000 9282
Arreglo ordenado con MergeSort:
0 3 7 10 23 24 111 166 9282 100000
Process exited after 20.01 seconds with return value 0
Presione una tecla para continuar . . .

EXPLICACIÓN

El enunciado cita lo siguiente:

*"realice su ordenamiento utilizando un proceso que permita **dividir** el problema en partes más pequeñas, resolver cada parte de manera independiente y combinar los resultados"*

Esto describe a la perfección el uso de el paradigma o metodología llamado **DIVINE Y VENCERÁS,** además solicita realizar su ordenamiento independientemente y al final combinarlos, esto es un claro uso del llamado **MergeSort**, este algoritmo se encuentra dentro de la metodología divide y vencerás, dividiendo el arreglo en mitades recursivamente, ordenarlos de forma independiente y combinándolos al final.

Además, se implementa el uso de memoria dinámica, puedo que el enunciado cita lo siguiente:

*"Implemente un programa en C++ que reciba un arreglo de números enteros de **tamaño variable**"*

Esto implica usar memoria dinámica, ya que el tamaño debe variar dependiendo lo que quiera el usuario.

```
    int arrtam;
    cout << "Ingrese el tamanio del arreglo: ";
    cin >> arrtam;
    cout << endl;
    int* arr = new int[arrtam];
```

Adicionalmente se implemento el uso de OOP, con clases, ya que el docente sugirió su uso antes de la realización de la prueba.

```
class Ordenador {
public:
        //Algoritmo MergeSort
};
```