14/08/2022

Actividad 1.1, Implementación de la técnica de programación "Divide y vencerás"

Equipo:

Miguel Jiménez Padilla | A01423189

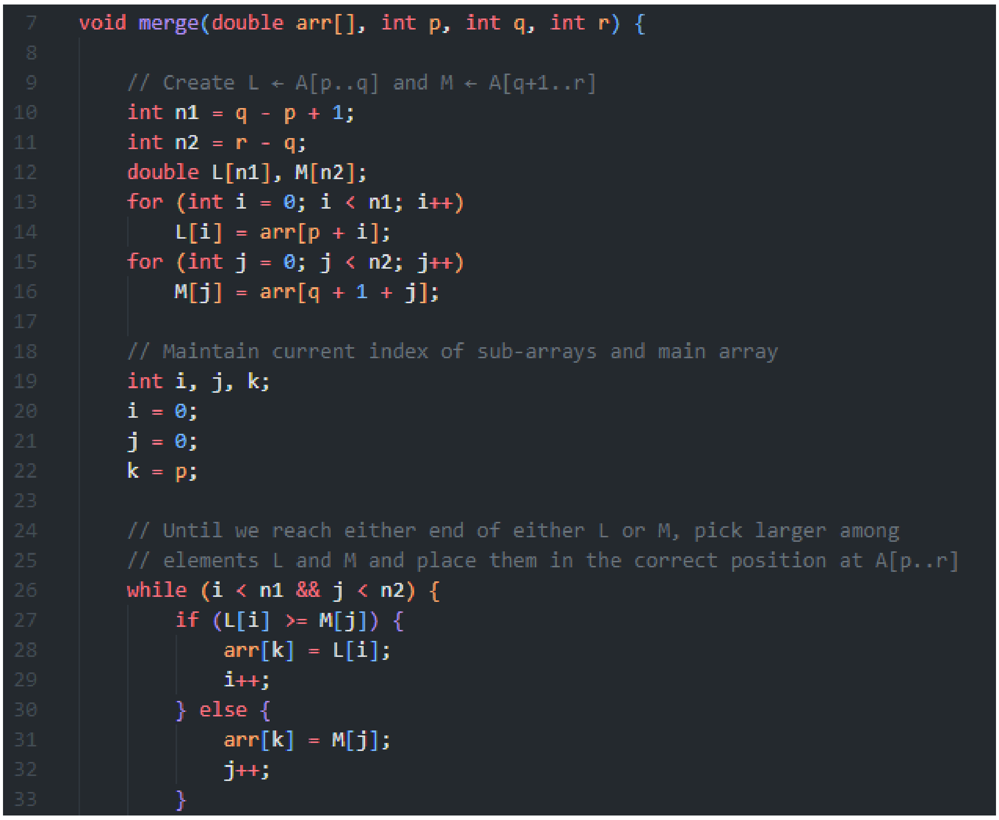
Marco Antonio Gardida Cortés | A01423221

Descripción:

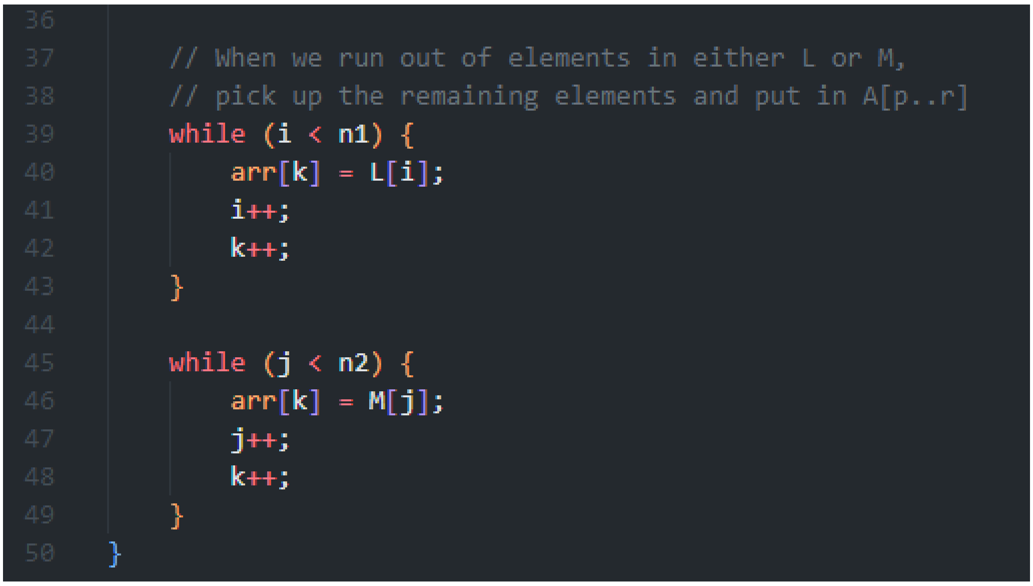
El propósito de la actividad es conocer el funcionamiento del algoritmo de ordenamiento "merge sort" que sirve gracias a la técnica "divide y vencerás". Este algoritmo funciona de manera recursiva y plantea dos funciones principales para el ordenamiento de un arreglo o vector en este caso de manera descendente:

Función "merge"

Complejidad -> O (n)



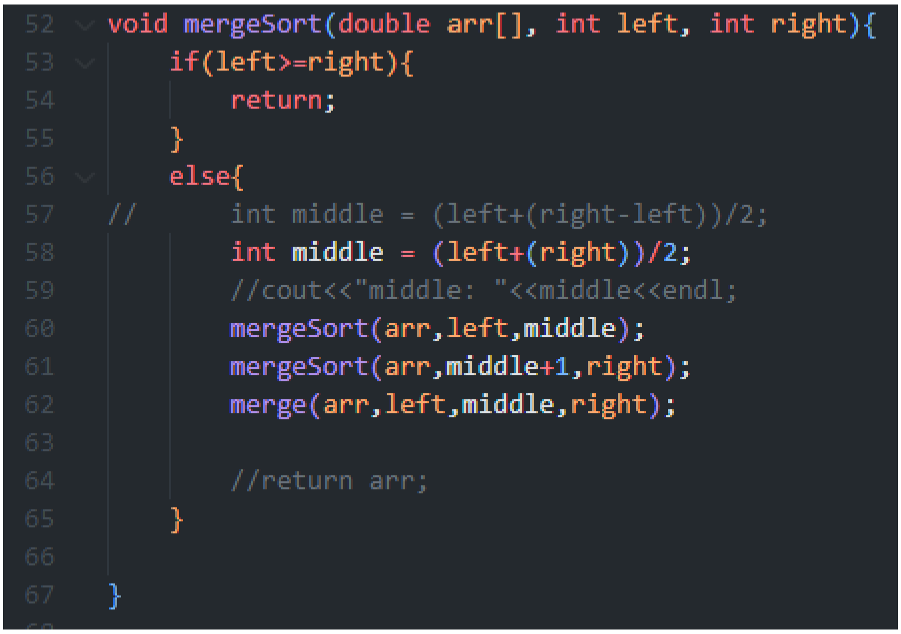
```
void merge(double arr[], int p, int q, int r) {

    // Create L ← A[p..q] and M ← A[q+1..r]
    int n1 = q - p + 1;
    int n2 = r - q;
    double L[n1], M[n2];
    for (int i = 0; i < n1; i++)
        L[i] = arr[p + i];
    for (int j = 0; j < n2; j++)
        M[j] = arr[q + 1 + j];

    // Maintain current index of sub-arrays and main array
    int i, j, k;
    i = 0;
    j = 0;
    k = p;

    // Until we reach either end of either L or M, pick larger among
    // elements L and M and place them in the correct position at A[p..r]
    while (i < n1 && j < n2) {
        if (L[i] >= M[j]) {
            arr[k] = L[i];
            i++;
        } else {
            arr[k] = M[j];
            j++;
        }
```



```

    // When we run out of elements in either L or M,
    // pick up the remaining elements and put in A[p..r]
    while (i < n1) {
        arr[k] = L[i];
        i++;
        k++;
    }

    while (j < n2) {
        arr[k] = M[j];
        j++;
        k++;
    }
}
```

Función “mergeSort”

Complejidad -> O (log n)

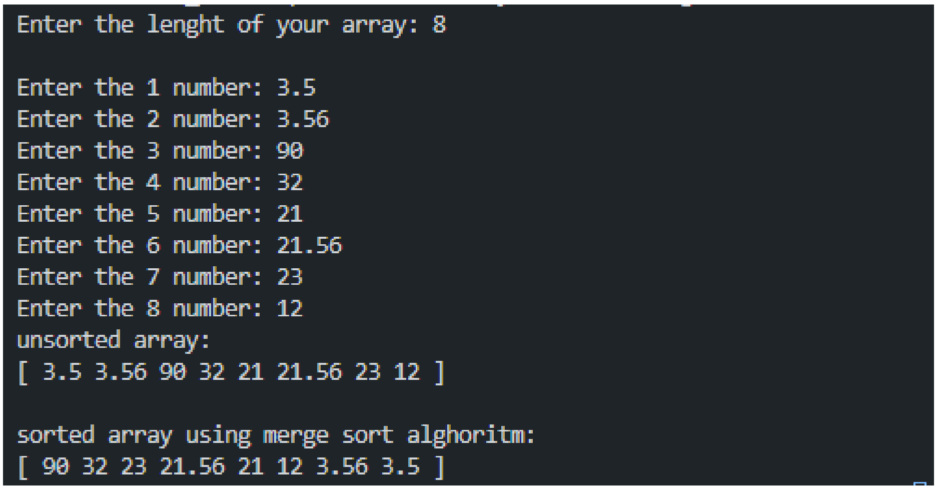


```
void mergeSort(double arr[], int left, int right){
    if(left>=right){
        return;
    }
    else{
//      int middle = (left+(right-left))/2;
        int middle = (left+(right))/2;
        //cout<<"middle: "<<middle<<endl;
        mergeSort(arr,left,middle);
        mergeSort(arr,middle+1,right);
        merge(arr,left,middle,right);

        //return arr;
    }

}
```

En la función mergeSort, se elige el índice medio y la función se llama a sí misma recursivamente para ir dividiendo el arreglo en partes más pequeñas, posteriormente, entra en juego la función “merge”, que reordena estos subgrupos del arreglo hasta ordenarlo completamente.

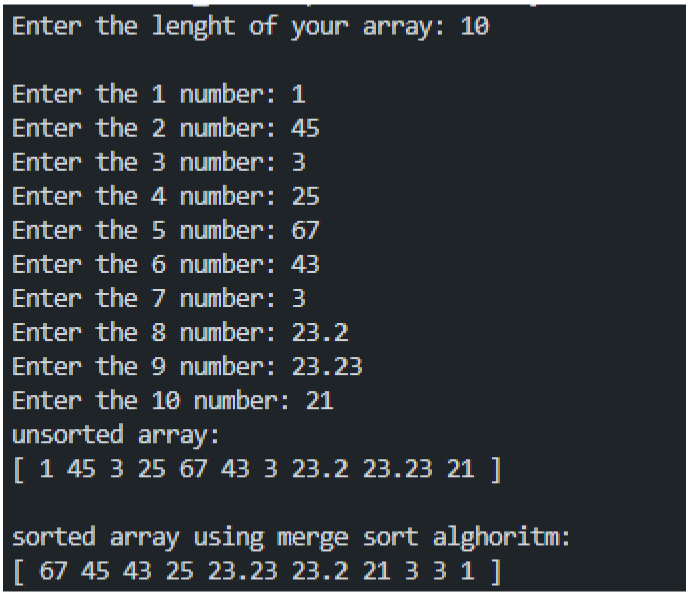
Casos de prueba:

Caso 1:



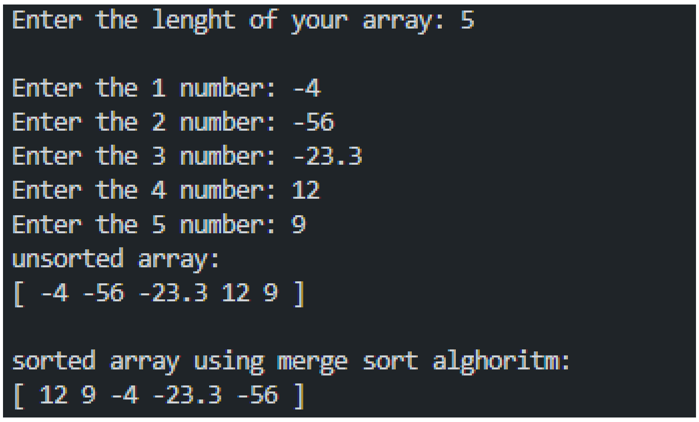
```
Enter the lenght of your array: 8

Enter the 1 number: 3.5
Enter the 2 number: 3.56
Enter the 3 number: 90
Enter the 4 number: 32
Enter the 5 number: 21
Enter the 6 number: 21.56
Enter the 7 number: 23
Enter the 8 number: 12
unsorted array:
[ 3.5 3.56 90 32 21 21.56 23 12 ]

sorted array using merge sort alghoritm:
[ 90 32 23 21.56 21 12 3.56 3.5 ]
```

Caso 2:



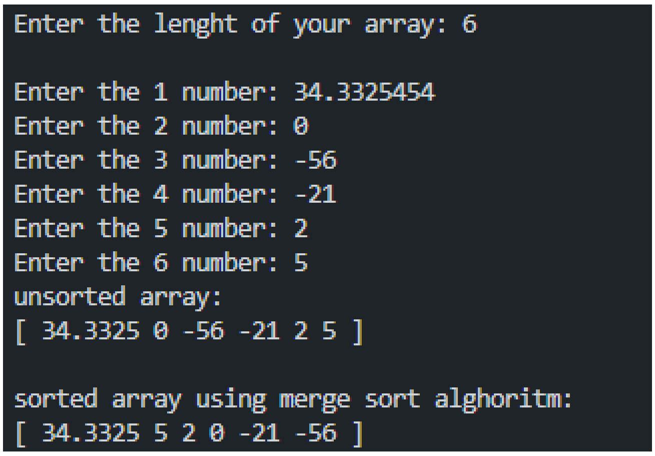
```
Enter the lenght of your array: 10

Enter the 1 number: 1
Enter the 2 number: 45
Enter the 3 number: 3
Enter the 4 number: 25
Enter the 5 number: 67
Enter the 6 number: 43
Enter the 7 number: 3
Enter the 8 number: 23.2
Enter the 9 number: 23.23
Enter the 10 number: 21
unsorted array:
[ 1 45 3 25 67 43 3 23.2 23.23 21 ]

sorted array using merge sort alghoritm:
[ 67 45 43 25 23.23 23.2 21 3 3 1 ]
```

Caso 3:



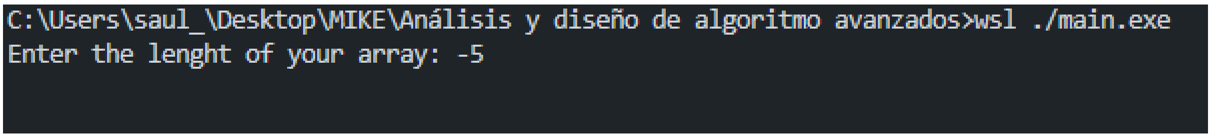
```
Enter the lenght of your array: 5

Enter the 1 number: -4
Enter the 2 number: -56
Enter the 3 number: -23.3
Enter the 4 number: 12
Enter the 5 number: 9
unsorted array:
[ -4 -56 -23.3 12 9 ]

sorted array using merge sort alghoritm:
[ 12 9 -4 -23.3 -56 ]
```

Caso 4:



```
Enter the lenght of your array: 6

Enter the 1 number: 34.3325454
Enter the 2 number: 0
Enter the 3 number: -56
Enter the 4 number: -21
Enter the 5 number: 2
Enter the 6 number: 5
unsorted array:
[ 34.3325 0 -56 -21 2 5 ]

sorted array using merge sort alghoritm:
[ 34.3325 5 2 0 -21 -56 ]
```

Caso en el que el algoritmo no funciona:



```
C:\Users\saul_\Desktop\MIKE\Análisis y diseño de algoritmo avanzados>wsl ./main.exe
Enter the lenght of your array: -5
```