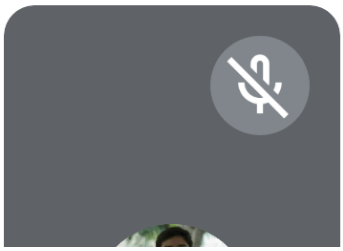
Nombre: Bryan Steve Montepeque Santos
Carné: 201700375

Tarea 2

Ordenamiento Rápido

```
void ordenaRapido(int a[], int primero, int ultimo)
{
    int central = (primero+ultimo)/2;
    int pivote = a[central];
    int aux;
    int i = primero, j = ultimo;
    do{
        while (a[i]<pivote) i++;
        while (a[j]>pivote) j--;
        if (i<=j)
        {
            aux = a[i];
            a[i] = a[j];
            a[j] = aux;
            i++;
            j--;
            itera++;
            mostrarArreglo(a,5);
        }
    }
    while(i<=j);
    if (primero<j)
        ordenaRapido(a, primero, j);
    if (i<ultimo)
        ordenaRapido(a, i, ultimo);
}
```



- **Paso 1:** Se crean las variables necesarias para poder realizar los cálculos, estos son central que indica cual es el dato central para poder ser utilizado como pivote, es decir, el valor en el cuál vamos a partir nuestro arreglo. Y auxiliar que nos va a servir para poder almacenar los datos mientras estemos organizando los datos. Así mismo, se reciben el primer y el ultimo dato del arreglo y se almacenan en i y j.

```
void ordenaRapido(int a[], int primero, int ultimo)
{
    int central = (primero+ultimo)/2;
    int pivote = a[central];
    int aux;
    int i = primero, j = ultimo;
```

Nombre: Bryan Steve Montepeque Santos
Carné: 201700375

- **Paso 2:** Se aumenta la posición de i en el arreglo hasta llegar a un número igual o mayor al pivote y se reduce el valor de j hasta llegar a un número igual o menor al pivote.

```
while (a[i]<pivote) i++;
while (a[j]>pivote) j--;
```

- **Paso 3:** Si el dato que es mayor o igual al pivote se encuentra a la izquierda de el valor igual o menor al pivote entonces auxiliar = dato en posición i, posición i del arreglo ahora es igual a el dato de la posición j del arreglo y ahora la posición j del arreglo tiene el valor del auxiliar, es decir el valor que estaba en la posición j del arreglo ahora se movió a la izquierda y el valor que estaba en la posición i del arreglo se movió a la derecha, ordenando así nuestro arreglo.

```
if (i<=j)
{
    aux = a[i];
    a[i] = a[j];
    a[j] = aux;
    i++;
    j--;
    itera++;
    mostrarArreglo(a,5);
}
```

Nombre: Bryan Steve Montepeque Santos
Carné: 201700375

- **Paso 4:** Todo este proceso se repite siempre y cuando i no sea mayor a j, es decir siempre que i esté a la izquierda de j.

```
do{
    while (a[i]<pivote) i++;
    while (a[j]>pivote) j--;
    if (i<=j)
    {
        aux = a[i];
        a[i] = a[j];
        a[j] = aux;
        i++;
        j--;
        itera++;
        mostrarArreglo(a,5);
    }
}
while(i<=j);
```

- **Paso 5:** Se repite el proceso de manera recursiva, pero si el primer dato sigue siendo menor a j entonces esta vez vuelve a ordenar, pero solo desde el primero hasta j, no desde el primero hasta el último. Igualmente, si i es menor al ultimo lo vuelve a ordenar, pero esta vez ya no ordena desde el primero hasta el ultimo sino solo desde i hasta el último. Entonces de esta forma las dos mitades que formamos en los pasos anteriores se ordenan de la misma forma hasta llegar a arreglos de 1 valor donde ya no es posible ordenar más.

```
if (primero<j)
    ordenaRapido(a, primero, j);
if (i<ultimo)
    ordenaRapido(a, i, ultimo);
```