```c
#include<stdio.h>
#include<conio.h>

// Function to merge two sorted subarrays
void merge(int arr[], int left, int mid, int right)
{
    int n1 = mid - left + 1;
    int n2 = right - mid;

    // Create temporary arrays
    int leftArr[n1], rightArr[n2];

    // Copy data to temporary arrays
    for (int i = 0; i < n1; i++)
        leftArr[i] = arr[left + i];
    for (int j = 0; j < n2; j++)
        rightArr[j] = arr[mid + 1 + j];

    // Merge temporary arrays
    int i = 0, j = 0, k = left;
    while (i < n1 && j < n2) {
        if (leftArr[i] <= rightArr[j]) {
            arr[k] = leftArr[i];
            i++;
        } else {
            arr[k] = rightArr[j];
            j++;
        }
        k++;
    }

    // Copy remaining elements
    while (i < n1) {
        arr[k] = leftArr[i];
        i++;
        k++;
    }
    while (j < n2) {
        arr[k] = rightArr[j];
        j++;
        k++;
    }
}

// Function to implement Merge Sort
void mergeSort(int arr[], int left, int right) {
    if (left < right) {
        int mid = left + (right - left) / 2;

        // Recursively sort subarrays
```

```
        mergeSort(arr, left, mid);
        mergeSort(arr, mid + 1, right);

        // Merge sorted subarrays
        merge(arr, left, mid, right);
    }
}

// Function to print array
void printArray(int arr[], int size) {
    for (int i = 0; i < size; i++)
        printf("%d ", arr[i]);
    printf("\n");
}

// Driver program
int main() {
    int arr[] = {9, 3, 7, 5, 6, 4, 8, 2, 1};
    int n = sizeof(arr) / sizeof(arr[0]);

    printf("Original array: \n");
    printArray(arr, n);

    mergeSort(arr, 0, n - 1);

    printf("Sorted array: \n");
    printArray(arr, n);

    return 0;
}
```

Output:

```
Original array:
9 3 7 5 6 4 8 2 1
Sorted array:
1 2 3 4 5 6 7 8 9
```

Explanation:

1. The merge function merges two sorted subarrays into a single sorted subarray.
2. The mergeSort function recursively divides the array into smaller subarrays until each subarray contains only one element.
3. The merge function is then called to merge and sort these subarrays.
4. The printArray function displays the array elements.

```
97
98    Time Complexity: O(n log n)
99    Space Complexity: O(n)
```