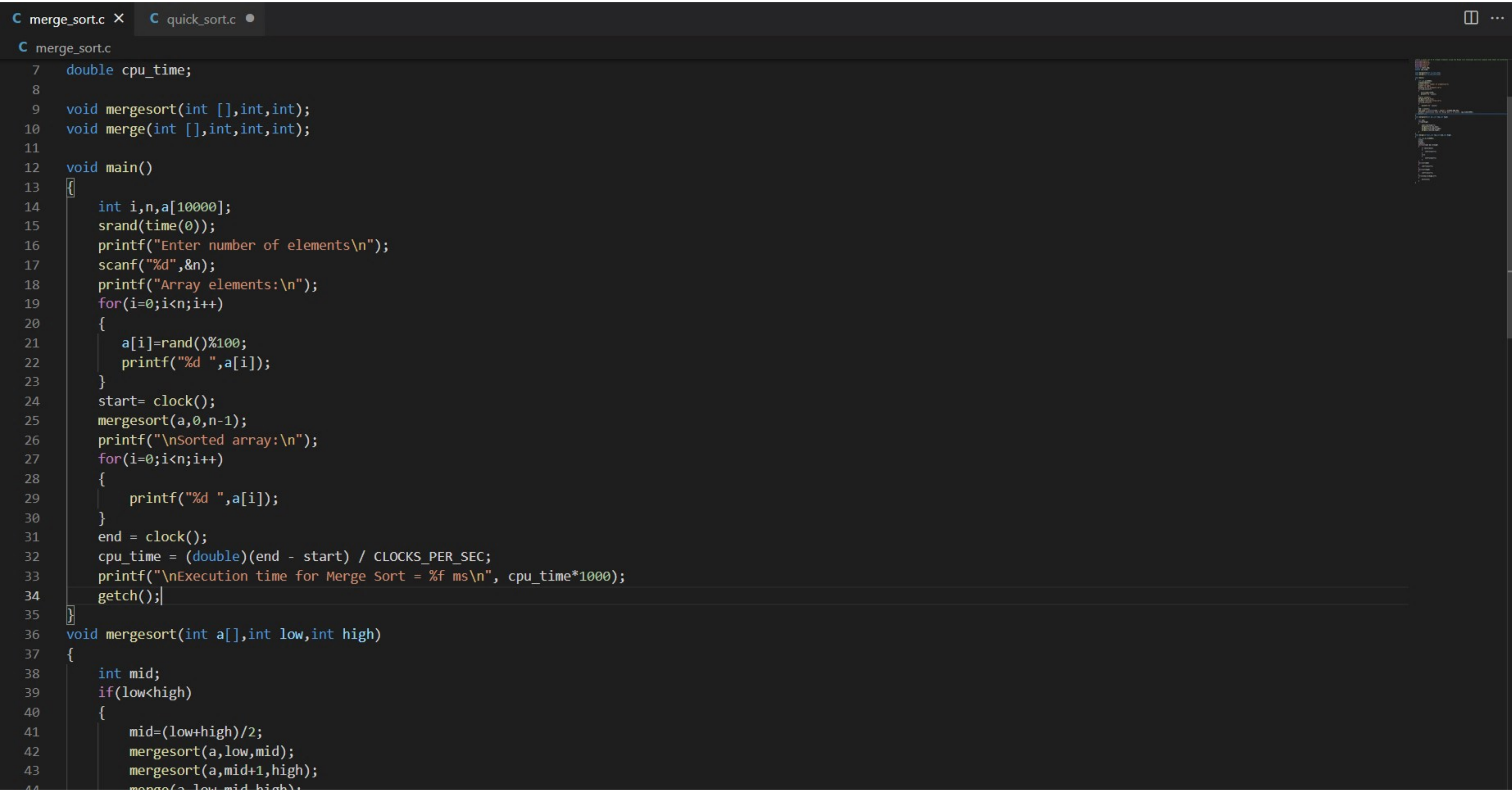
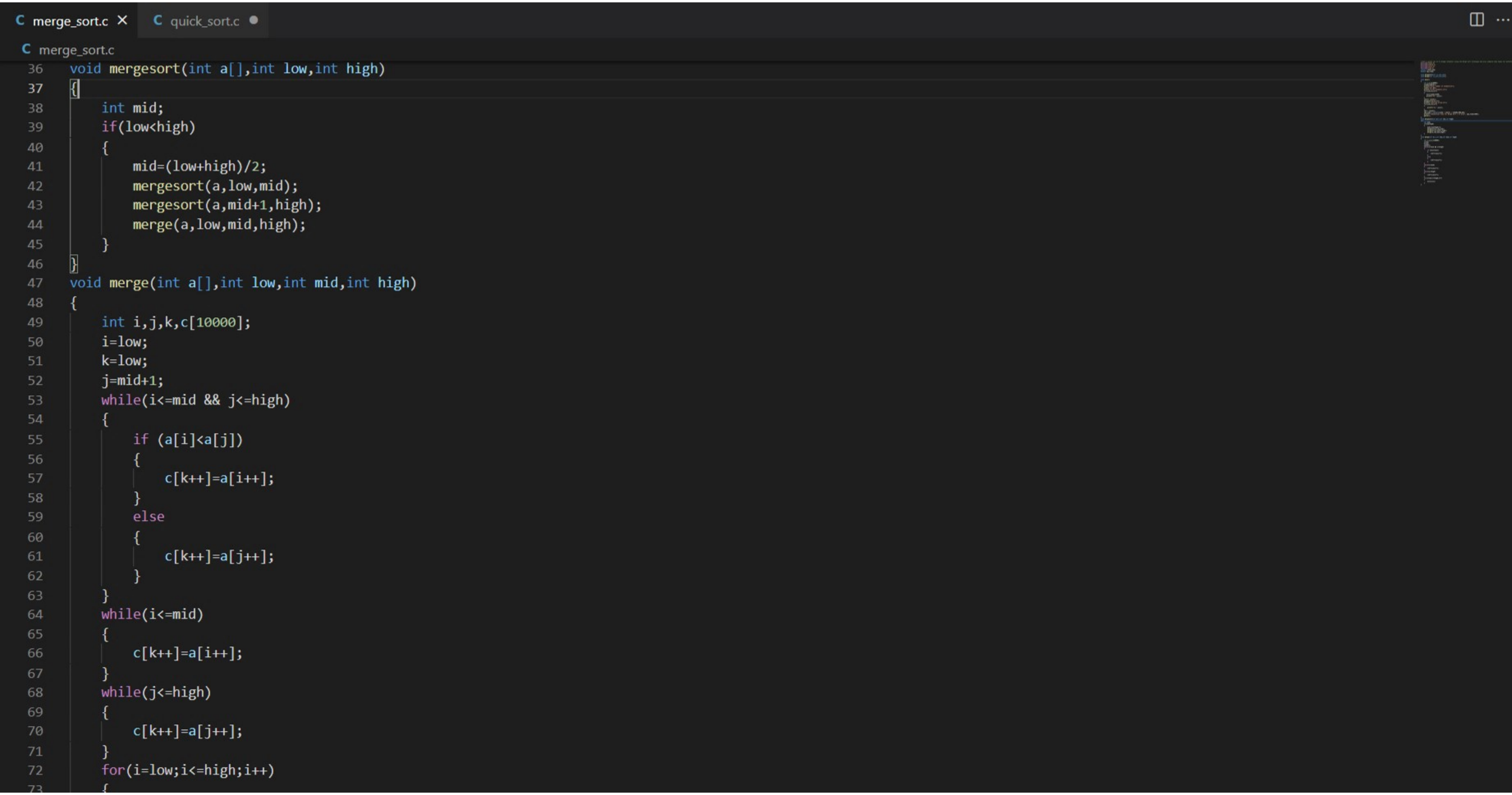


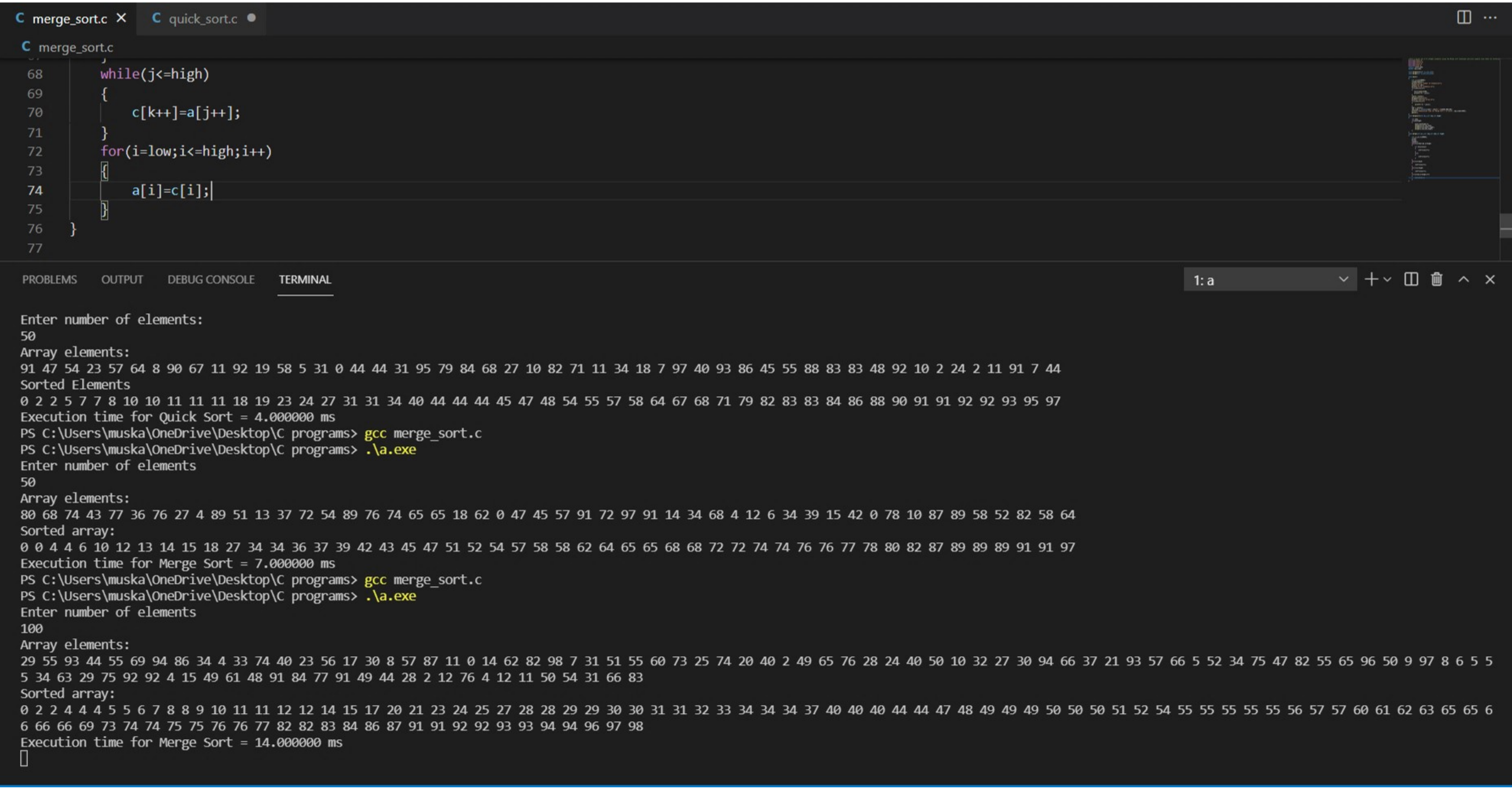
```c
double cpu_time;

void mergesort(int [],int,int);
void merge(int [],int,int,int);

void main()
{
    int i,n,a[10000];
    srand(time(0));
    printf("Enter number of elements\n");
    scanf("%d",&n);
    printf("Array elements:\n");
    for(i=0;i<n;i++)
    {
       a[i]=rand()%100;
       printf("%d ",a[i]);
    }
    start= clock();
    mergesort(a,0,n-1);
    printf("\nSorted array:\n");
    for(i=0;i<n;i++)
    {
        printf("%d ",a[i]);
    }
    end = clock();
    cpu_time = (double)(end - start) / CLOCKS_PER_SEC;
    printf("\nExecution time for Merge Sort = %f ms\n", cpu_time*1000);
    getch();
}
void mergesort(int a[],int low,int high)
{
    int mid;
    if(low<high)
    {
        mid=(low+high)/2;
        mergesort(a,low,mid);
        mergesort(a,mid+1,high);
```



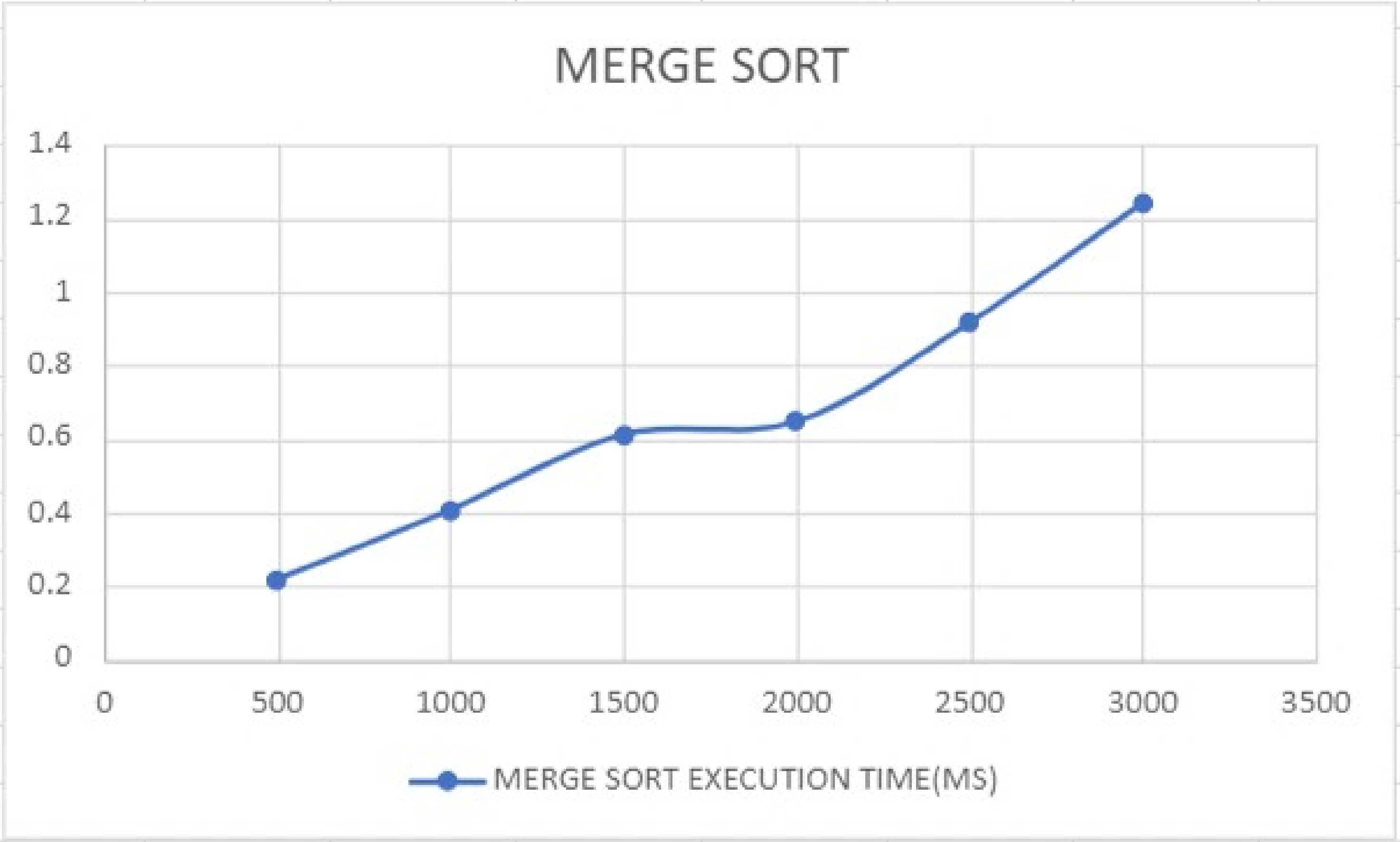
```c
void mergesort(int a[],int low,int high)
{
    int mid;
    if(low<high)
    {
        mid=(low+high)/2;
        mergesort(a,low,mid);
        mergesort(a,mid+1,high);
        merge(a,low,mid,high);
    }
}
void merge(int a[],int low,int mid,int high)
{
    int i,j,k,c[10000];
    i=low;
    k=low;
    j=mid+1;
    while(i<=mid && j<=high)
    {
        if (a[i]<a[j])
        {
            c[k++]=a[i++];
        }
        else
        {
            c[k++]=a[j++];
        }
    }
    while(i<=mid)
    {
        c[k++]=a[i++];
    }
    while(j<=high)
    {
        c[k++]=a[j++];
    }
    for(i=low;i<=high;i++)
    {
```



```c
    while(j<=high)
    {
        c[k++]=a[j++];
    }
    for(i=low;i<=high;i++)
    {
        a[i]=c[i];
    }
}
```

```
Enter number of elements:
50
Array elements:
91 47 54 23 57 64 8 90 67 11 92 19 58 5 31 0 44 44 31 95 79 84 68 27 10 82 71 11 34 18 7 97 40 93 86 45 55 88 83 83 48 92 10 2 24 2 11 91 7 44
Sorted Elements
0 2 2 5 7 7 8 10 10 11 11 11 18 19 23 24 27 31 31 34 40 44 44 44 45 47 48 54 55 57 58 64 67 68 71 79 82 83 83 84 86 88 90 91 91 92 92 93 95 97
Execution time for Quick Sort = 4.000000 ms
PS C:\Users\muska\OneDrive\Desktop\C programs> gcc merge_sort.c
PS C:\Users\muska\OneDrive\Desktop\C programs> .\a.exe
Enter number of elements
50
Array elements:
80 68 74 43 77 36 76 27 4 89 51 13 37 72 54 89 76 74 65 65 18 62 0 47 45 57 91 72 97 91 14 34 68 4 12 6 34 39 15 42 0 78 10 87 89 58 52 82 58 64
Sorted array:
0 0 4 4 6 10 12 13 14 15 18 27 34 34 36 37 39 42 43 45 47 51 52 54 57 58 58 62 64 65 65 68 68 72 72 74 74 76 76 77 78 80 82 87 89 89 89 91 91 97
Execution time for Merge Sort = 7.000000 ms
PS C:\Users\muska\OneDrive\Desktop\C programs> gcc merge_sort.c
PS C:\Users\muska\OneDrive\Desktop\C programs> .\a.exe
Enter number of elements
100
Array elements:
29 55 93 44 55 69 94 86 34 4 33 74 40 23 56 17 30 8 57 87 11 0 14 62 82 98 7 31 51 55 60 73 25 74 20 40 2 49 65 76 28 24 40 50 10 32 27 30 94 66 37 21 93 57 66 5 52 34 75 47 82 55 65 96 50 9 97 8 6 5 5
5 34 63 29 75 92 92 4 15 49 61 48 91 84 77 91 49 44 28 2 12 76 4 12 11 50 54 31 66 83
Sorted array:
0 2 2 4 4 4 5 5 6 7 8 8 9 10 11 11 12 12 14 15 17 20 21 23 24 25 27 28 28 29 29 30 30 31 31 32 33 34 34 34 37 40 40 40 44 44 47 48 49 49 49 50 50 50 51 52 54 55 55 55 55 55 56 57 57 60 61 62 63 65 65 6
6 66 66 69 73 74 74 75 75 76 76 77 82 82 83 84 86 87 91 91 92 92 93 93 94 94 96 97 98
Execution time for Merge Sort = 14.000000 ms
```

N	EXECUTION TIME(MS)
500	0.217
1000	0.407
1500	0.613
2000	0.647
2500	0.917
3000	1.243


MERGE SORT
MERGE SORT EXECUTION TIME(MS)