

**Jobsheet 10**  
**Praktikum Struktur Data**



**Dosen pengampu : Randi Proska Sandra, S.Pd, M.Sc**

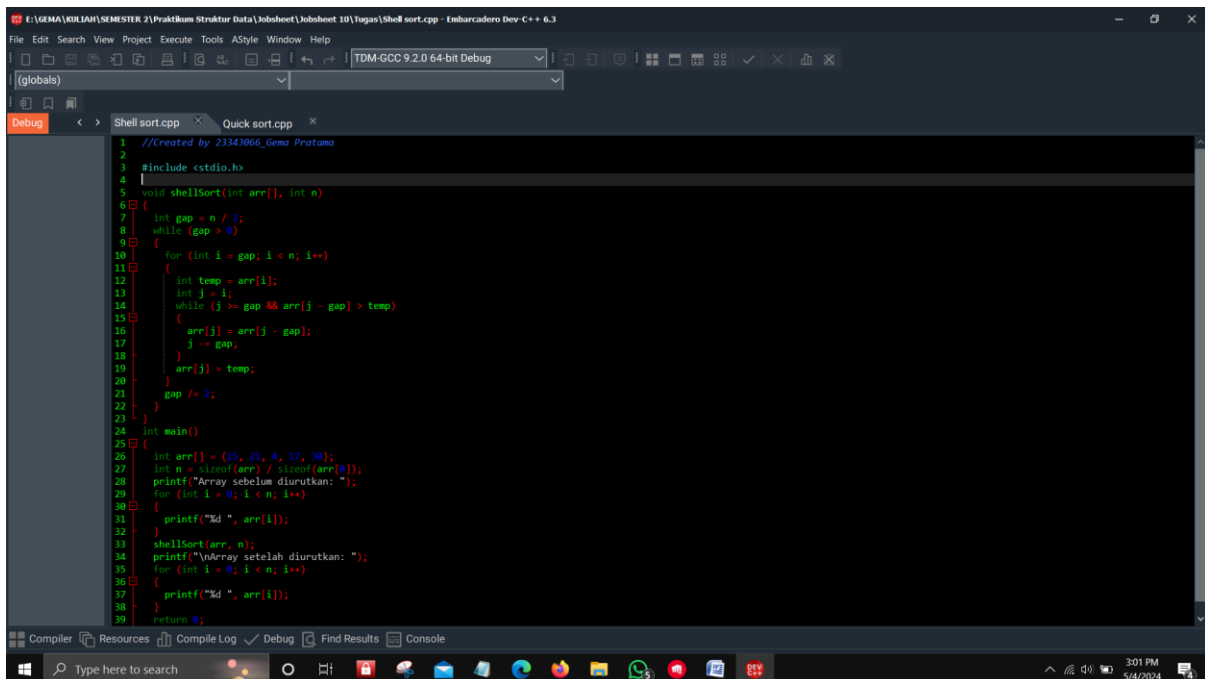
**Kode Kelas : 202323430158**

**Disusun Oleh :**

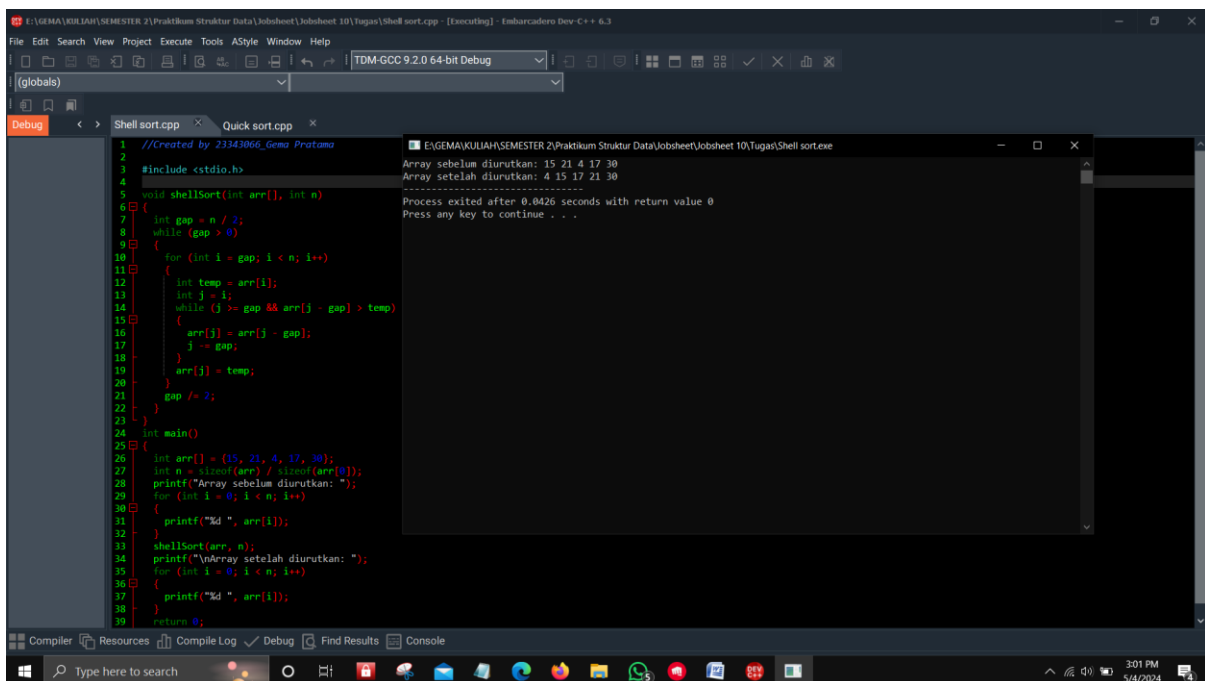
**Gema Pratama Mahadi Putera**  
**23343066**

**PROGRAM STUDI INFORMATIKA (NK)**  
**FAKULTAS TEKNIK**  
**UNIVERSITAS NEGERI PADANG**  
**2024**

## 1. Shell Sort :



```
1 //Created by 23343066_Gema Pratama
2
3 #include <stdio.h>
4
5 void shellSort(int arr[], int n)
6 {
7     int gap = n / 2;
8     while (gap > 0)
9     {
10         for (int i = gap; i < n; i++)
11         {
12             int temp = arr[i];
13             int j = i;
14             while (j >= gap && arr[j - gap] > temp)
15             {
16                 arr[j] = arr[j - gap];
17                 j -= gap;
18             }
19             arr[j] = temp;
20         }
21         gap /= 2;
22     }
23 }
24
25 int main()
26 {
27     int arr[] = {15, 21, 4, 17, 30};
28     int n = sizeof(arr) / sizeof(arr[0]);
29     printf("Array sebelum diurutkan: ");
30     for (int i = 0; i < n; i++)
31     {
32         printf("%d ", arr[i]);
33     }
34     shellSort(arr, n);
35     printf("\nArray setelah diurutkan: ");
36     for (int i = 0; i < n; i++)
37     {
38         printf("%d ", arr[i]);
39     }
40     return 0;
41 }
```



```
1 //Created by 23343066_Gema Pratama
2
3 #include <stdio.h>
4
5 void shellSort(int arr[], int n)
6 {
7     int gap = n / 2;
8     while (gap > 0)
9     {
10         for (int i = gap; i < n; i++)
11         {
12             int temp = arr[i];
13             int j = i;
14             while (j >= gap && arr[j - gap] > temp)
15             {
16                 arr[j] = arr[j - gap];
17                 j -= gap;
18             }
19             arr[j] = temp;
20         }
21         gap /= 2;
22     }
23 }
24
25 int main()
26 {
27     int arr[] = {15, 21, 4, 17, 30};
28     int n = sizeof(arr) / sizeof(arr[0]);
29     printf("Array sebelum diurutkan: ");
30     for (int i = 0; i < n; i++)
31     {
32         printf("%d ", arr[i]);
33     }
34     shellSort(arr, n);
35     printf("\nArray setelah diurutkan: ");
36     for (int i = 0; i < n; i++)
37     {
38         printf("%d ", arr[i]);
39     }
40     return 0;
41 }
```

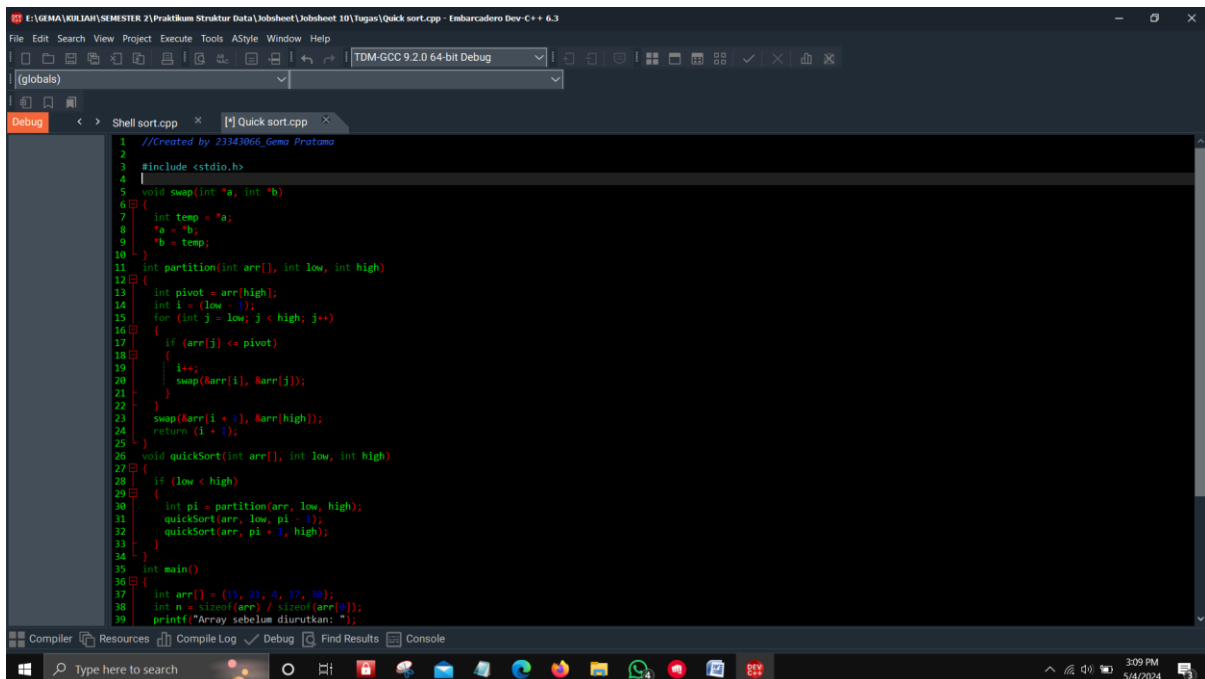
```
Array sebelum diurutkan: 15 21 4 17 30
Array setelah diurutkan: 4 15 17 21 30
-----
Process exited after 0.0426 seconds with return value 0
Press any key to continue . . .
```

## Penjelasan:

- Metode shell sort merupakan variasi dari metode insertion sort.
- Ia menggunakan konsep "gap" untuk membagi array menjadi sub-grup yang lebih kecil.
- Setelah array dibagi, setiap sub-grup diurutkan secara terpisah menggunakan insertion sort.
- Kemudian, gap dikurangi secara bertahap hingga menjadi 1 dan array diurutkan sepenuhnya.

- Shell sort adalah pilihan yang baik untuk mengurutkan data yang berukuran sedang. Ini lebih efisien daripada insertion sort standar dan dapat bersaing dengan merge sort dan quicksort dalam kasus tertentu. Namun, pemilihan algoritma sorting yang tepat tergantung pada karakteristik data dan kebutuhan aplikasi.

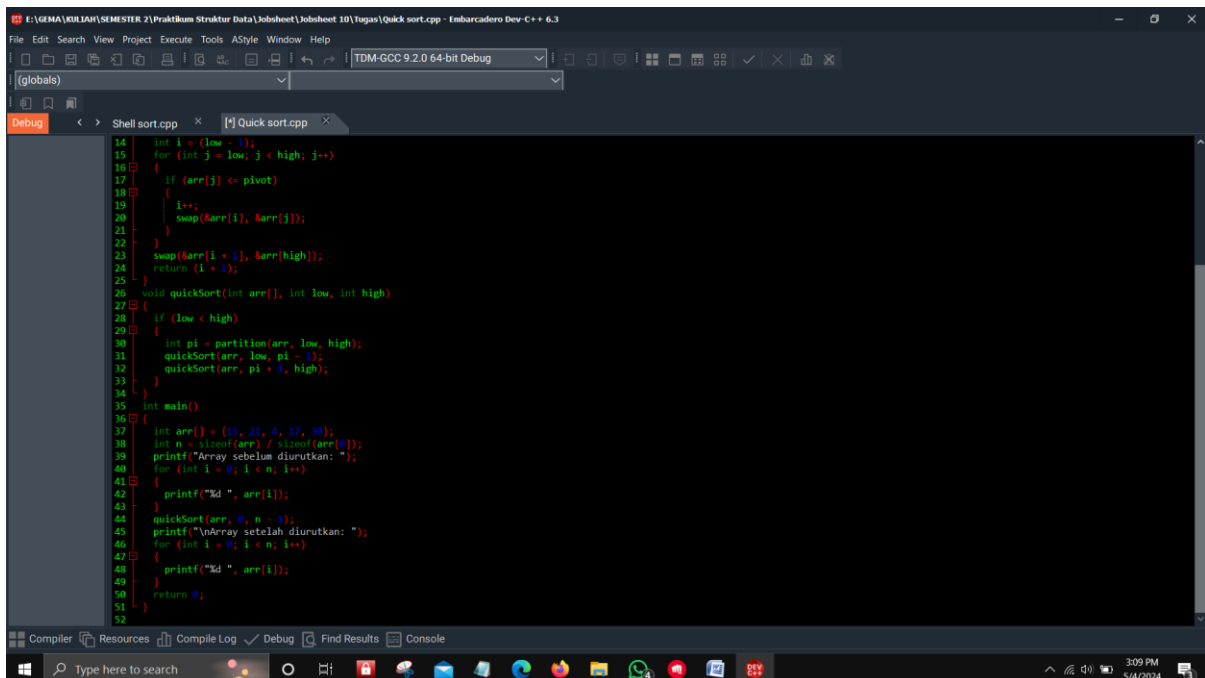
## 2. Quick Sort :



```

1 //Created by 23343066_Gema Pratama
2
3 #include <stdio.h>
4
5 void swap(int *a, int *b)
6 {
7     int temp = *a;
8     *a = *b;
9     *b = temp;
10 }
11
12 int partition(int arr[], int low, int high)
13 {
14     int pivot = arr[high];
15     int i = (low - 1);
16     for (int j = low; j < high; j++)
17     {
18         if (arr[j] <= pivot)
19         {
20             i++;
21             swap(&arr[i], &arr[j]);
22         }
23     }
24     swap(&arr[i + 1], &arr[high]);
25     return (i + 1);
26 }
27
28 void quickSort(int arr[], int low, int high)
29 {
30     if (low < high)
31     {
32         int pi = partition(arr, low, high);
33         quickSort(arr, low, pi - 1);
34         quickSort(arr, pi + 1, high);
35     }
36 }
37
38 int main()
39 {
40     int arr[] = {15, 21, 4, 17, 30};
41     int n = sizeof(arr) / sizeof(arr[0]);
42     printf("Array sebelum diurutkan: ");
43 }

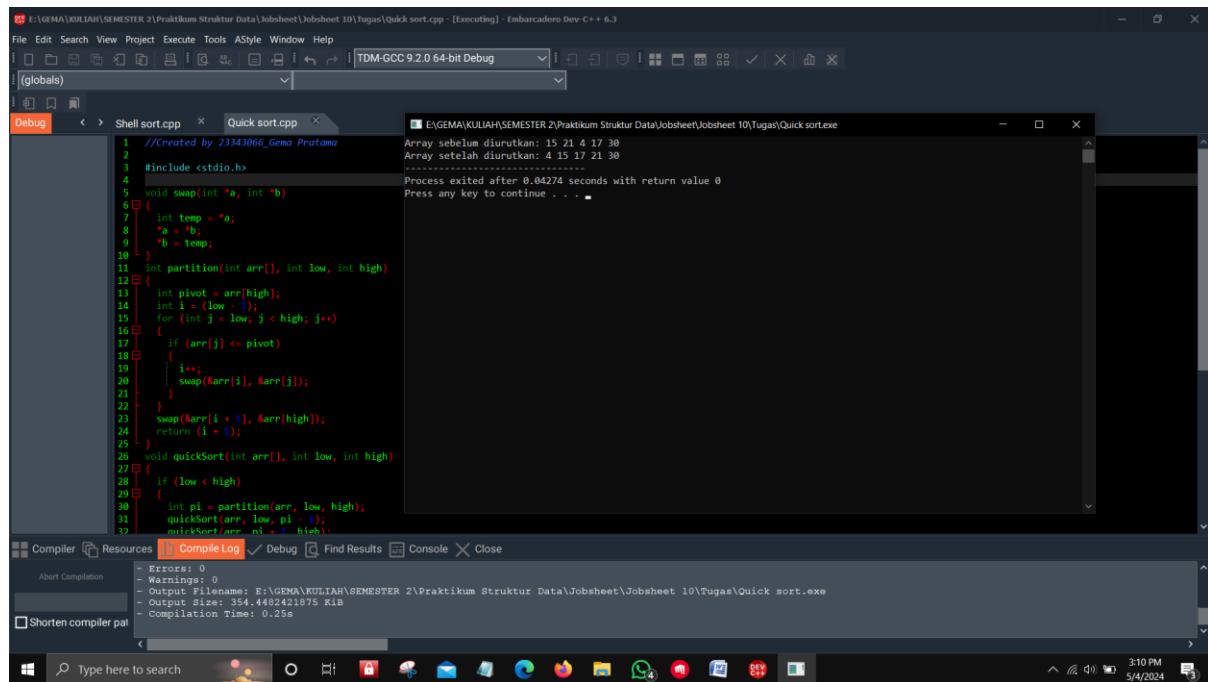
```



```

14     int i = (low - 1);
15     for (int j = low; j < high; j++)
16     {
17         if (arr[j] <= pivot)
18         {
19             i++;
20             swap(&arr[i], &arr[j]);
21         }
22     }
23     swap(&arr[i + 1], &arr[high]);
24     return (i + 1);
25 }
26
27 void quickSort(int arr[], int low, int high)
28 {
29     if (low < high)
30     {
31         int pi = partition(arr, low, high);
32         quickSort(arr, low, pi - 1);
33         quickSort(arr, pi + 1, high);
34     }
35 }
36
37 int main()
38 {
39     int arr[] = {15, 21, 4, 17, 30};
40     int n = sizeof(arr) / sizeof(arr[0]);
41     printf("Array sebelum diurutkan: ");
42     for (int i = 0; i < n; i++)
43     {
44         printf("%d ", arr[i]);
45     }
46     quickSort(arr, 0, n - 1);
47     printf("\nArray setelah diurutkan: ");
48     for (int i = 0; i < n; i++)
49     {
50         printf("%d ", arr[i]);
51     }
52     return 0;
53 }

```



## Penjelasan:

- Metode quick sort menggunakan pendekatan rekursif untuk membagi array menjadi dua sub-array berdasarkan suatu elemen pivot.
- Setiap elemen dalam array dibandingkan dengan pivot, dan elemen-elemen yang lebih kecil dari pivot ditempatkan di sebelah kiri pivot, sedangkan elemen-elemen yang lebih besar ditempatkan di sebelah kanan.
- Proses ini dilakukan secara rekursif untuk kedua sub-array hingga seluruh array diurutkan.
- Quicksort adalah pilihan yang baik untuk mengurutkan data berukuran besar karena efisiensinya. Namun, untuk data berukuran kecil atau data yang sudah terurut, algoritma lain seperti merge sort atau insertion sort mungkin lebih cocok. Penting untuk mempertimbangkan karakteristik data dan kebutuhan aplikasi saat memilih algoritma pengurutan.