

Jobsheet 9
Praktikum Struktur Data



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

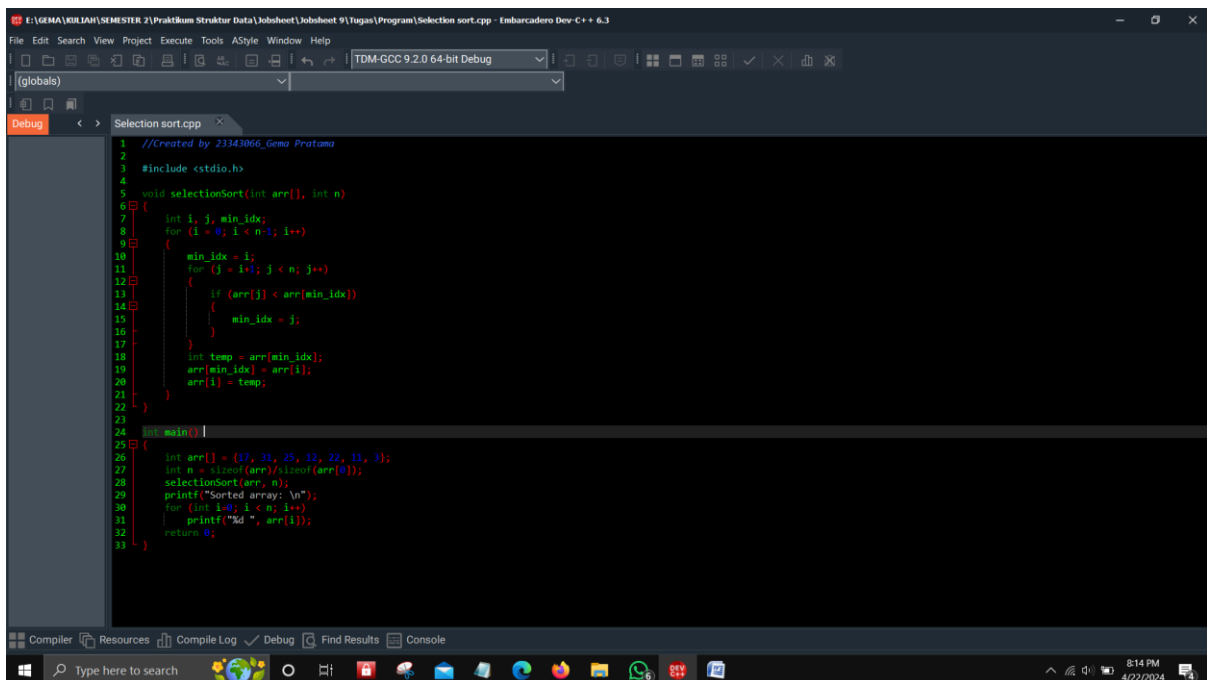
Kode Kelas : 202323430158

Disusun Oleh :

Gema Pratama Mahadi Putera
23343066

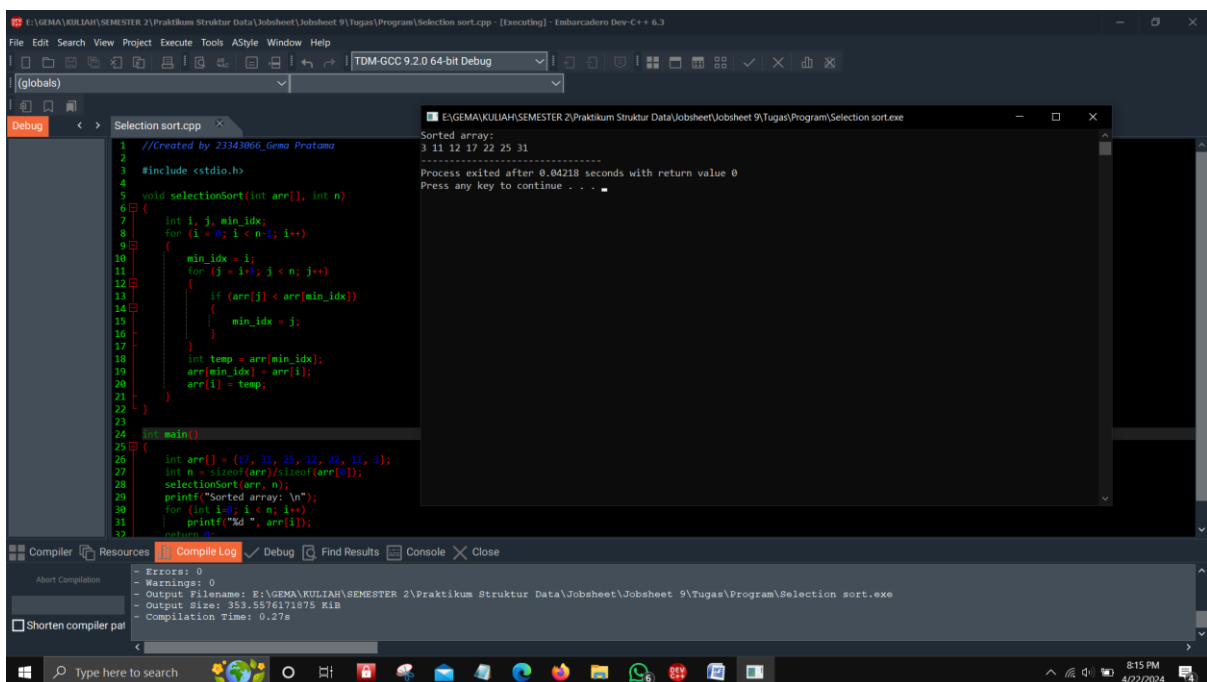
PROGRAM STUDI INFORMATIKA (NK)
FAKULTAS TEKNIK
UNIVERSITAS NEGERI PADANG
2024

1. Selection Sort :



The screenshot shows a C++ IDE with the following code in `Selection sort.cpp`:

```
1 //Created by 23343066_Gema Pratama
2
3 #include <stdio.h>
4
5 void selectionSort(int arr[], int n)
6 {
7     int i, j, min_idx;
8     for (i = 0; i < n-1; i++)
9     {
10         min_idx = i;
11         for (j = i+1; j < n; j++)
12         {
13             if (arr[j] < arr[min_idx])
14             {
15                 min_idx = j;
16             }
17         }
18         int temp = arr[min_idx];
19         arr[min_idx] = arr[i];
20         arr[i] = temp;
21     }
22 }
23
24 int main()
25 {
26     int arr[] = {17, 31, 25, 12, 22, 11, 3};
27     int n = sizeof(arr)/sizeof(arr[0]);
28     selectionSort(arr, n);
29     printf("Sorted array: \n");
30     for (int i=0; i < n; i++)
31     {
32         printf("%d ", arr[i]);
33     }
34     return 0;
35 }
```



The screenshot shows the same C++ IDE with the execution output and compile log visible.

Execution Output:

```
Sorted arrays:
3 11 12 17 22 25 31
-----
Process exited after 0.04218 seconds with return value 0
Press any key to continue . . .
```

Compile Log:

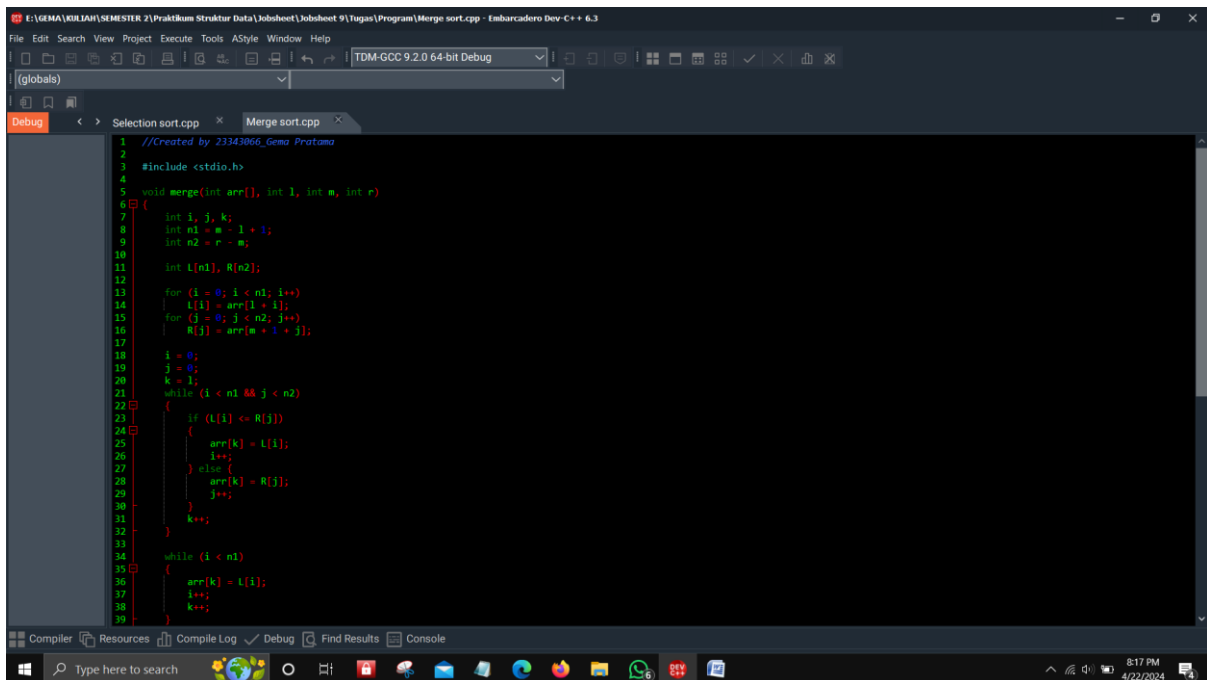
```
Errors: 0
Warnings: 0
Output Filename: E:\GEMA\KULIAH\SEMESTER 2\Praktikum Struktur Data\Jobsheet\Jobsheet 9\Tugas\Program\Selection sort.exe
Output Size: 353.5576171875 KiB
Compilation Time: 0.27s
```

Penjelasan:

- Algoritma selection sort bekerja dengan mencari elemen terkecil dalam array pada setiap iterasi.
- Elemen terkecil kemudian ditukar dengan elemen pada posisi awal array yang belum diurutkan.
- Proses ini diulang sampai semua elemen diurutkan.

- Prinsip utama selection sort adalah memilih elemen terkecil satu per satu dan menempatkannya di posisinya yang benar.
- Selection sort adalah pilihan yang baik untuk data berukuran kecil atau untuk skenario di mana kesederhanaan dan kemudahan implementasi menjadi prioritas. Namun, untuk data berukuran besar, disarankan menggunakan algoritma sorting yang lebih efisien seperti merge sort atau quicksort.

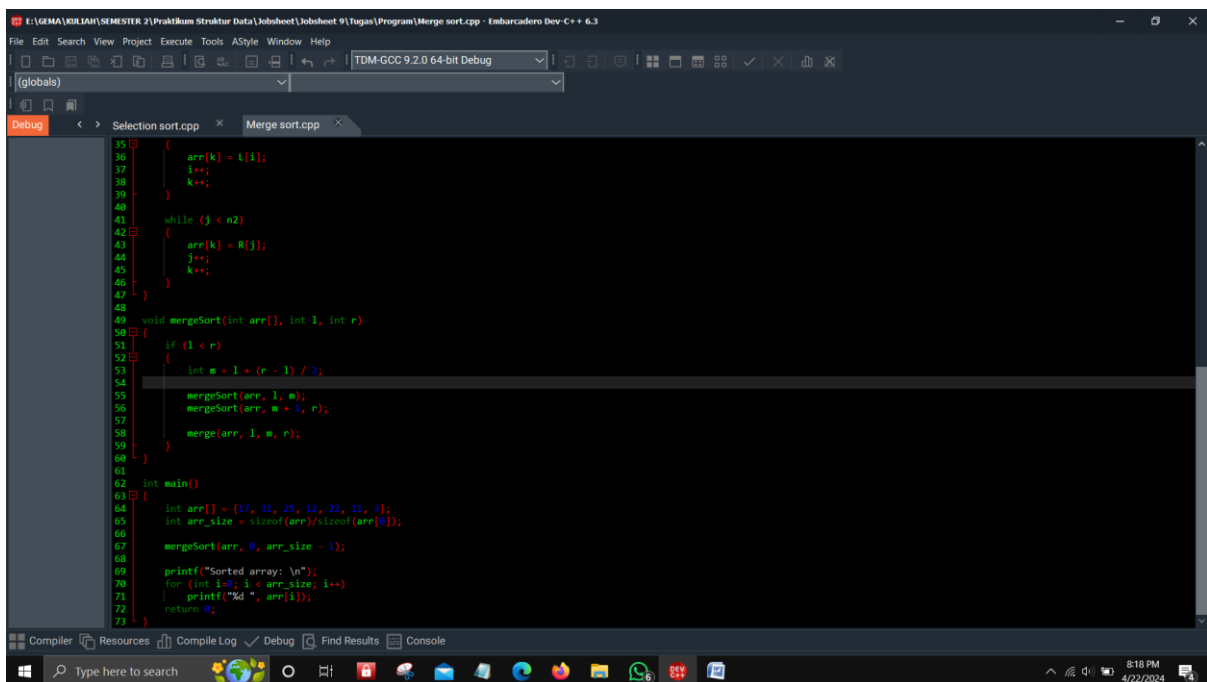
2. Merge Sort :



```

1 //Created by 23343066_Gema Pratama
2
3 #include <stdio.h>
4
5 void merge(int arr[], int l, int m, int r)
6 {
7     int i, j, k;
8     int n1 = m - l + 1;
9     int n2 = r - m;
10
11     int L[n1], R[n2];
12
13     for (i = 0; i < n1; i++)
14         L[i] = arr[l + i];
15     for (j = 0; j < n2; j++)
16         R[j] = arr[m + 1 + j];
17
18     i = 0;
19     j = 0;
20     k = l;
21     while (i < n1 && j < n2)
22     {
23         if (L[i] <= R[j])
24         {
25             arr[k] = L[i];
26             i++;
27         }
28         else
29         {
30             arr[k] = R[j];
31             j++;
32         }
33         k++;
34     }
35     while (i < n1)
36     {
37         arr[k] = L[i];
38         i++;
39         k++;
40     }
41     while (j < n2)
42     {
43         arr[k] = R[j];
44         j++;
45         k++;
46     }
47 }
48
49 void mergeSort(int arr[], int l, int r)
50 {
51     if (l < r)
52     {
53         int m = l + (r - l) / 2;
54
55         mergeSort(arr, l, m);
56         mergeSort(arr, m + 1, r);
57
58         merge(arr, l, m, r);
59     }
60 }
61
62 int main()
63 {
64     int arr[] = {37, 31, 25, 12, 22, 11, 3};
65     int arr_size = sizeof(arr)/sizeof(arr[0]);
66
67     mergeSort(arr, 0, arr_size - 1);
68
69     printf("Sorted array: \n");
70     for (int i=0; i < arr_size; i++)
71         printf("%d ", arr[i]);
72     return 0;
73 }

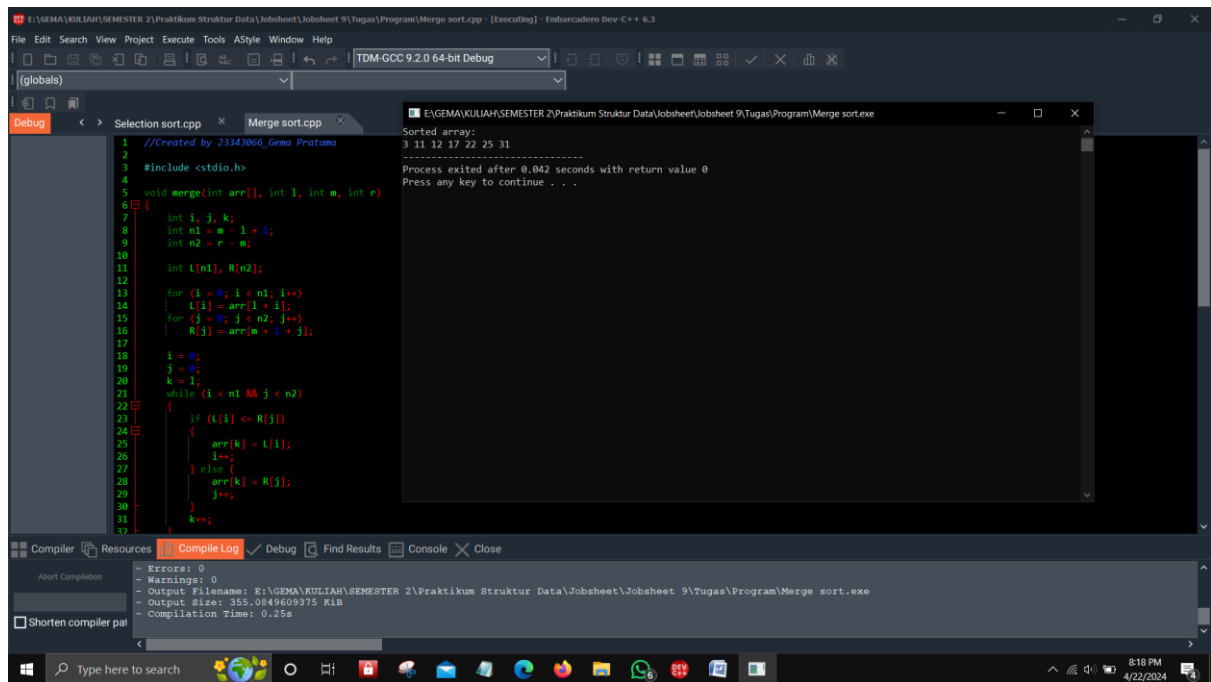
```



```

35 {
36     arr[k] = L[i];
37     i++;
38     k++;
39 }
40
41 while (j < n2)
42 {
43     arr[k] = R[j];
44     j++;
45     k++;
46 }
47 }
48
49 void mergeSort(int arr[], int l, int r)
50 {
51     if (l < r)
52     {
53         int m = l + (r - l) / 2;
54
55         mergeSort(arr, l, m);
56         mergeSort(arr, m + 1, r);
57
58         merge(arr, l, m, r);
59     }
60 }
61
62 int main()
63 {
64     int arr[] = {37, 31, 25, 12, 22, 11, 3};
65     int arr_size = sizeof(arr)/sizeof(arr[0]);
66
67     mergeSort(arr, 0, arr_size - 1);
68
69     printf("Sorted array: \n");
70     for (int i=0; i < arr_size; i++)
71         printf("%d ", arr[i]);
72     return 0;
73 }

```



Penjelasan:

- Metode merge sort menggunakan pendekatan rekursif untuk membagi array menjadi dua bagian hingga menjadi sub-array yang berisi satu elemen.
- Kemudian, sub-array tersebut digabungkan secara berurutan (merge) dengan membandingkan dan menggabungkan elemen-elemen dalam sub-array.
- Proses ini berlanjut hingga semua sub-array digabungkan kembali menjadi satu array yang terurut.
- Merge sort adalah pilihan yang baik untuk mengurutkan data berukuran besar karena efisiensinya. Namun, untuk data berukuran kecil, bubble sort atau selection sort mungkin lebih sederhana untuk diimplementasikan.