

Tugas Jobsheet 9



Dosen pengampu : Randi Proska Sandra, M.Sc

Kode Kelas : 202323430158

Disusun Oleh :

**Fajrul Huda Ash Shiddiq
23343063**

**PROGRAM STUDI INFORMATIKA (NK)
FAKULTAS TEKNIK
UNIVERSITAS NEGERI PADANG
2023**

Selection Sort

1. Source Code

```
//Created by_Fajrul Huda Ash Shiddiq_23343063
#include <stdio.h>

void selectionSort(int arr[], int n) {
    int i, j, min_idx;

    for (i = 0; i < n-1; i++) {
        // Mencari elemen terkecil dalam array yang belum diurutkan
        min_idx = i;
        for (j = i+1; j < n; j++) {
            if (arr[j] < arr[min_idx]) {
                min_idx = j;
            }
        }

        // Menukar elemen terkecil yang ditemukan dengan elemen
        pertama dalam array yang belum diurutkan
        int temp = arr[min_idx];
        arr[min_idx] = arr[i];
        arr[i] = temp;
    }
}

int main() {
    int arr[] = {14, 25, 12, 8, 33};
    int n = sizeof(arr)/sizeof(arr[0]);
    selectionSort(arr, n);
    printf("Array setelah diurutkan: \n");
    for (int i=0; i < n; i++)
```

```

        printf("%d ", arr[i]);

printf("\n");

return 0;

}

```

2. Screenshot

```

1 //Created by: Fajrul Huda Ash Shiddiq_23343063
2 #include <stdio.h>
3
4 void selectionSort(int arr[], int n) {
5     int i, j, min_idx;
6
7     for (i = 0; i < n-1; i++) {
8         // Mencari elemen terkecil dalam array yang belum diurutkan
9         min_idx = i;
10        for (j = i+1; j < n; j++) {
11            if (arr[j] < arr[min_idx]) {
12                min_idx = j;
13            }
14        }
15
16        // Menukar elemen terkecil yang ditemukan dengan elemen pertama dalam array yang belum diurutkan
17        int temp = arr[min_idx];
18        arr[min_idx] = arr[i];
19        arr[i] = temp;
20    }
21 }
22
23 int main() {
24     int arr[] = {14, 25, 12, 8, 33};
25     int n = sizeof(arr)/sizeof(arr[0]);
26     selectionSort(arr, n);
27     printf("Array setelah diurutkan: \n");
28     for (int i=0; i < n; i++) {
29         printf("%d ", arr[i]);
30     }
31     printf("\n");
32     return 0;
33 }

```

Compiler: TDM-GCC 9.2.0 64-bit Release

Compiler Output:

```

- Warnings: 0
- Output Filename: F:\Praktikum Struktur Data\Jobsheet 9\Tugas\Project\Merge Sort.exe
- Output Size: 324,5 KiB
- Compilation Time: 0,11s

```

Line: 22 Col: 1 Sel: 0 Lines: 32 Length: 867 Insert Done parsing in 0,016 seconds

```

F:\Praktikum Struktur Data\Jo x + v
Array setelah diurutkan:
8 12 14 25 33

-----
Process exited after 12.82 seconds with return value 0
Press any key to continue . . .

```

26°C Berawan 21:00 07/05/2024

3. Penjelasan

a. Fungsi 'void selectionSort':

- Ini adalah fungsi yang mengurutkan array menggunakan algoritma selection sort.
- Algoritma ini memilih elemen terkecil dari array yang belum diurutkan dan memindahkannya ke bagian depan array yang diurutkan.
- Pada setiap iterasi, elemen terkecil ditemukan dengan membandingkan setiap elemen dengan elemen terkecil yang ditemukan sebelumnya.
- Kemudian elemen terkecil ditukar dengan elemen pertama dari array yang belum diurutkan.
- Proses ini berlanjut hingga seluruh array diurutkan.

b. Fungsi 'main':

- Membuat array yang akan diurutkan.
- Memanggil fungsi selectionSort() untuk mengurutkan array.
- Mencetak array setelah diurutkan.

Merge Sort

1. Source Code

```
//Created by_Fajrul Huda Ash Shiddiq_23343063
#include <stdio.h>

void merge(int arr[], int l, int m, int r) {
    int i, j, k;
    int n1 = m - l + 1;
    int n2 = r - m;

    // Membuat array sementara
    int L[n1], R[n2];

    // Menyalin data ke array sementara L[] dan R[]
    for (i = 0; i < n1; i++)
        L[i] = arr[l + i];
    for (j = 0; j < n2; j++)
```

```
R[j] = arr[m + 1 + j];
```

```
// Menggabungkan array sementara kembali menjadi arr[l..r]
i = 0; // Indeks awal dari subarray pertama
j = 0; // Indeks awal dari subarray kedua
k = l; // Indeks awal dari subarray hasil penggabungan
while (i < n1 && j < n2) {
    if (L[i] <= R[j]) {
        arr[k] = L[i];
        i++;
    } else {
        arr[k] = R[j];
        j++;
    }
    k++;
}

// Menyalin elemen yang tersisa dari L[], jika ada
while (i < n1) {
    arr[k] = L[i];
    i++;
    k++;
}

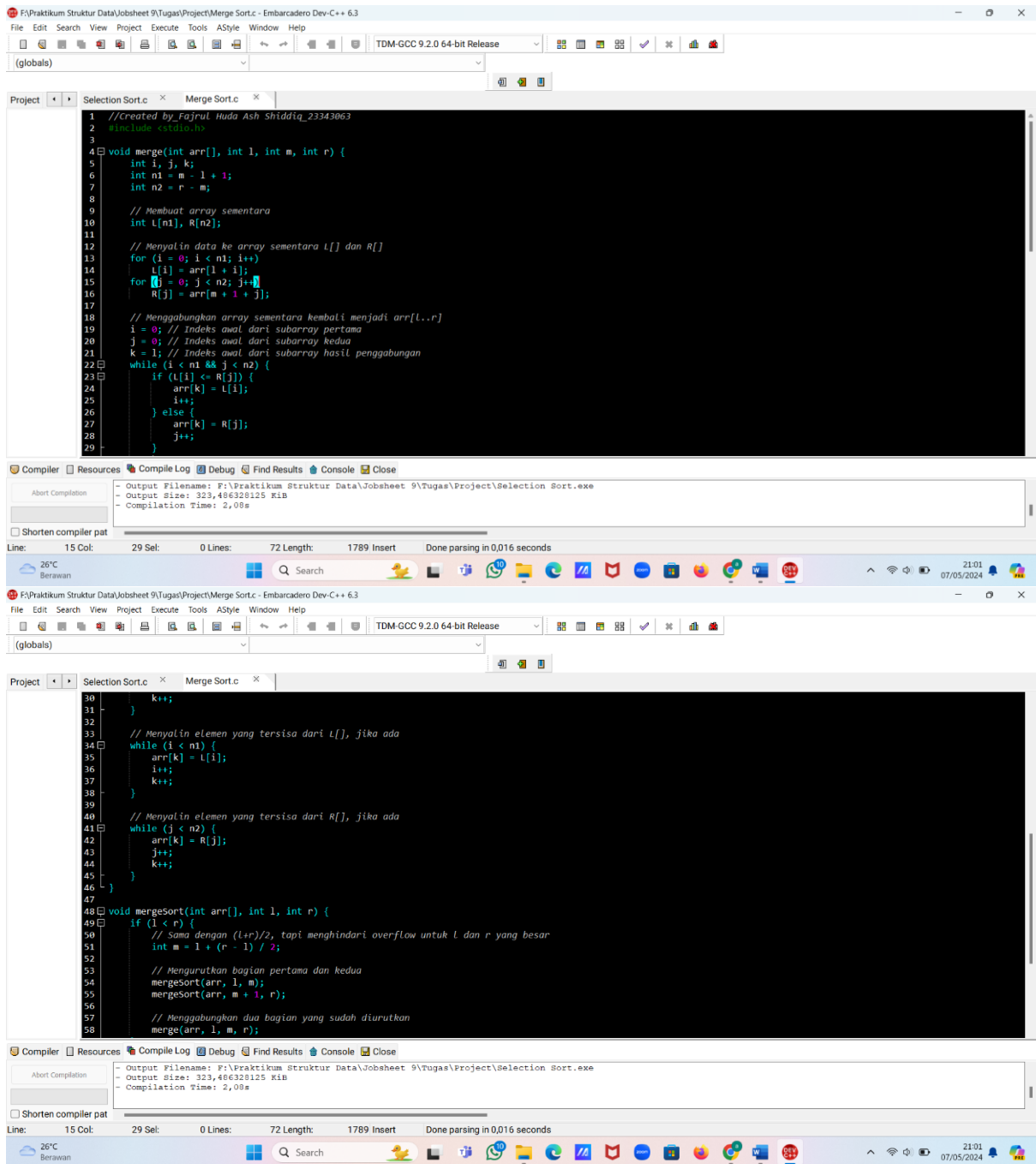
// Menyalin elemen yang tersisa dari R[], jika ada
while (j < n2) {
    arr[k] = R[j];
    j++;
    k++;
}
```

```
}
```

```
void mergeSort(int arr[], int l, int r) {  
    if (l < r) {  
        // Sama dengan (l+r)/2, tapi menghindari overflow untuk l  
        dan r yang besar  
        int m = l + (r - l) / 2;  
  
        // Mengurutkan bagian pertama dan kedua  
        mergeSort(arr, l, m);  
        mergeSort(arr, m + 1, r);  
  
        // Menggabungkan dua bagian yang sudah diurutkan  
        merge(arr, l, m, r);  
    }  
}
```

```
int main() {  
    int arr[] = {14, 25, 12, 8, 33};  
    int n = sizeof(arr) / sizeof(arr[0]);  
    mergeSort(arr, 0, n - 1);  
    printf("Array setelah diurutkan: \n");  
    for (int i = 0; i < n; i++)  
        printf("%d ", arr[i]);  
    printf("\n");  
    return 0;  
}
```

2. Screenshot



The screenshot displays a C++ IDE with a project named 'Merge Sort.c'. The code implements a recursive merge sort algorithm. The `mergesort` function takes an array `arr` and indices `l` and `r`. It recursively splits the array into two halves until a single element is reached, then merges them back in sorted order. The `main` function initializes an array `arr = {14, 25, 12, 8, 33}` and prints the sorted array.

```
44 k++;
45 }
46 }
47
48 void mergesort(int arr[], int l, int r) {
49     if (l < r) {
50         // Sama dengan (l+r)/2, tapi menghindari overflow untuk l dan r yang besar
51         int m = l + (r - l) / 2;
52
53         // Mengurutan bagian pertama dan kedua
54         mergesort(arr, l, m);
55         mergesort(arr, m + 1, r);
56
57         // Menggabungkan dua bagian yang sudah diurutkan
58         merge(arr, l, m, r);
59     }
60 }
61
62 int main() {
63     int arr[] = {14, 25, 12, 8, 33};
64     int n = sizeof(arr) / sizeof(arr[0]);
65     mergesort(arr, 0, n - 1);
66     printf("Array setelah diurutkan: \n");
67     for (int i = 0; i < n; i++)
68         printf("%d ", arr[i]);
69     printf("\n");
70     return 0;
71 }
72 }
```

The console output shows the sorted array: `8 12 14 25 33`. The process exited after 9.648 seconds with a return value of 0.

3. Penjelasan

- Fungsi ini bertanggung jawab untuk menggabungkan dua bagian terurut dari array menjadi satu.
- Fungsi ini menggunakan pendekatan rekursif untuk mengurutkan array.
- Argumen `arr` adalah array yang akan digabungkan, `l` adalah indeks awal bagian kiri, `m` adalah indeks tengah, dan `r` adalah indeks akhir bagian kanan.
- Pertama, dua bagian array (bagian kiri dan kanan) dipindahkan ke array sementara `L[]` dan `R[]`.
- Selama ada elemen di kedua bagian, elemen yang lebih kecil dari kedua bagian tersebut dipilih secara bergantian dan dimasukkan ke dalam array hasil.

- Setelah satu bagian kosong, sisa elemen dari bagian yang belum kosong langsung disalin ke array hasil.
- `mergeSort(int arr[], int l, int r):`
- Argumen `arr` adalah array yang akan diurutkan, `l` adalah indeks awal array, dan `r` adalah indeks akhir array.
- Algoritma ini membagi array menjadi dua bagian hingga hanya ada satu elemen dalam setiap bagian.
- Kemudian, bagian-bagian tersebut diurutkan secara terpisah dengan memanggil fungsi `mergeSort` secara rekursif.
- Akhirnya, fungsi `merge` dipanggil untuk menggabungkan dua bagian yang sudah diurutkan menjadi satu.