Tugas Jobsheet 10

Dosen pengampu : Randi Proska Sandra, M.Sc

Kode Kelas : 202323430158

Disusun Oleh :

Fajrul Huda Ash Shiddiq
23343063

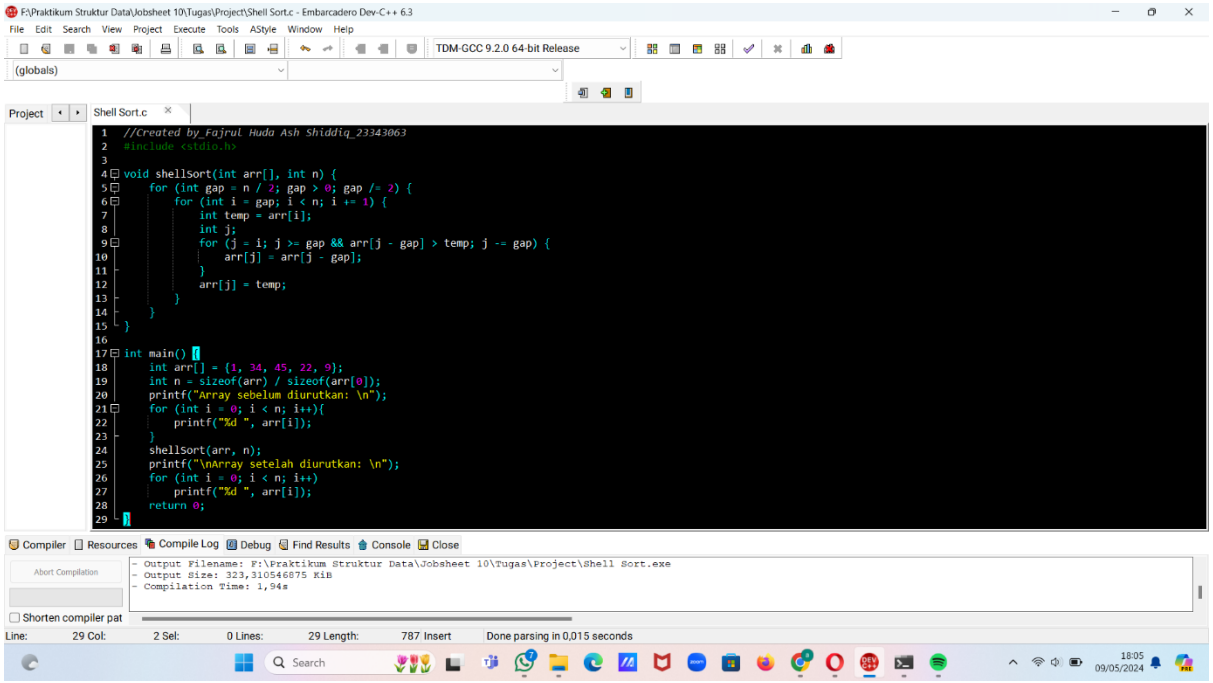
PROGRAM STUDI INFORMATIKA (NK)
FAKULTAS TEKNIK
UNIVERSITAS NEGERI PADANG
2023

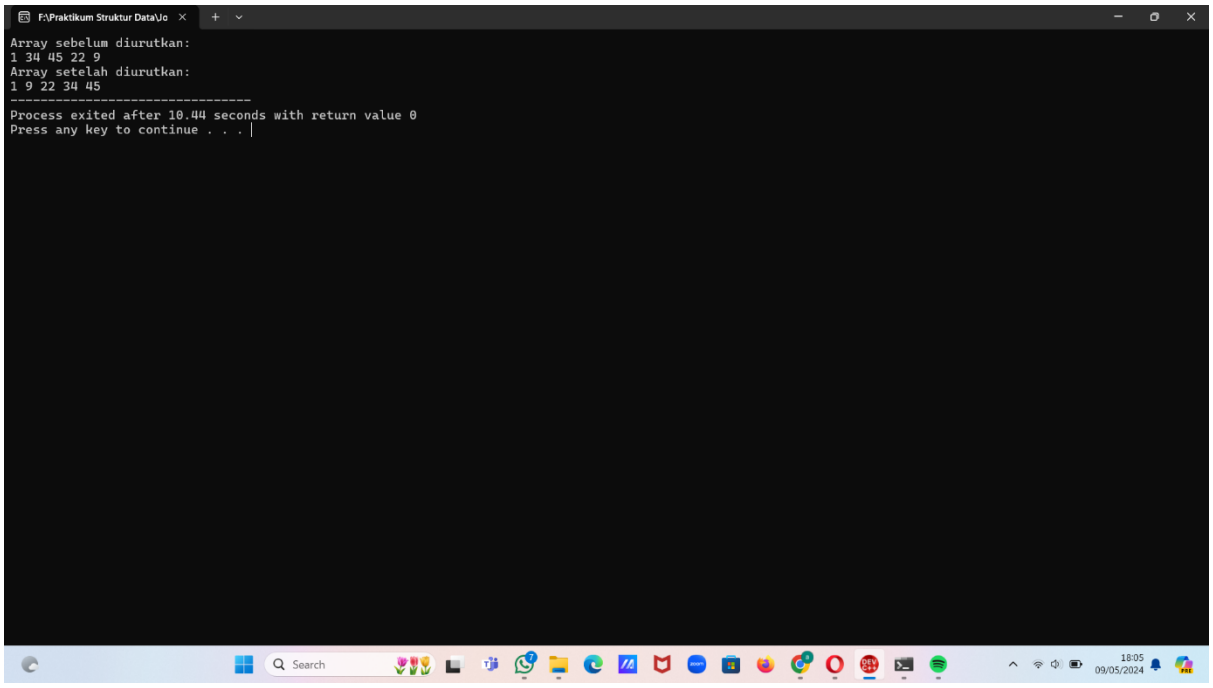
Shell Sort

1. Source Code

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

void shellSort(int arr[], int n) {
    for (int gap = n / 2; gap > 0; gap /= 2) {
        for (int i = gap; i < n; i += 1) {
            int temp = arr[i];
            int j;
            for (j = i; j >= gap && arr[j - gap] > temp; j -= gap) {
                arr[j] = arr[j - gap];
            }
            arr[j] = temp;
        }
    }
}

int main() {
    int arr[] = {1, 34, 45, 22, 9};
    int n = sizeof(arr) / sizeof(arr[0]);
    printf("Array sebelum diurutkan: \n");
    for (int i = 0; i < n; i++){
        printf("%d ", arr[i]);
    }
    shellSort(arr, n);
    printf("\nArray setelah diurutkan: \n");
    for (int i = 0; i < n; i++)
        printf("%d ", arr[i]);
    return 0;
}
```

2. Screenshot





3. Penjelasan

- Program dimulai dengan mendeklarasikan array yang akan diurutkan.
- Program mengatur nilai awal gap, biasanya setengah dari panjang array, dan menguranginya setiap iterasi hingga mencapai 1.
- Untuk setiap nilai gap, program melintasi array dan membandingkan elemen yang terpisah oleh jarak gap tersebut.
- Program menggunakan pendekatan insertion sort untuk setiap kelompok elemen yang memiliki jarak gap yang sama. Program membandingkan dan menukar elemen-

elemen ini sesuai kebutuhan untuk memastikan setiap kelompok elemen yang terpisah oleh gap tersebut terurut.

- Setelah satu iterasi selesai, gap dikurangi setengahnya, dan prosesnya diulangi sampai gap mencapai 1.
- Setelah gap mencapai 1, algoritma seperti insertion sort biasa, dan array dianggap telah terurut.

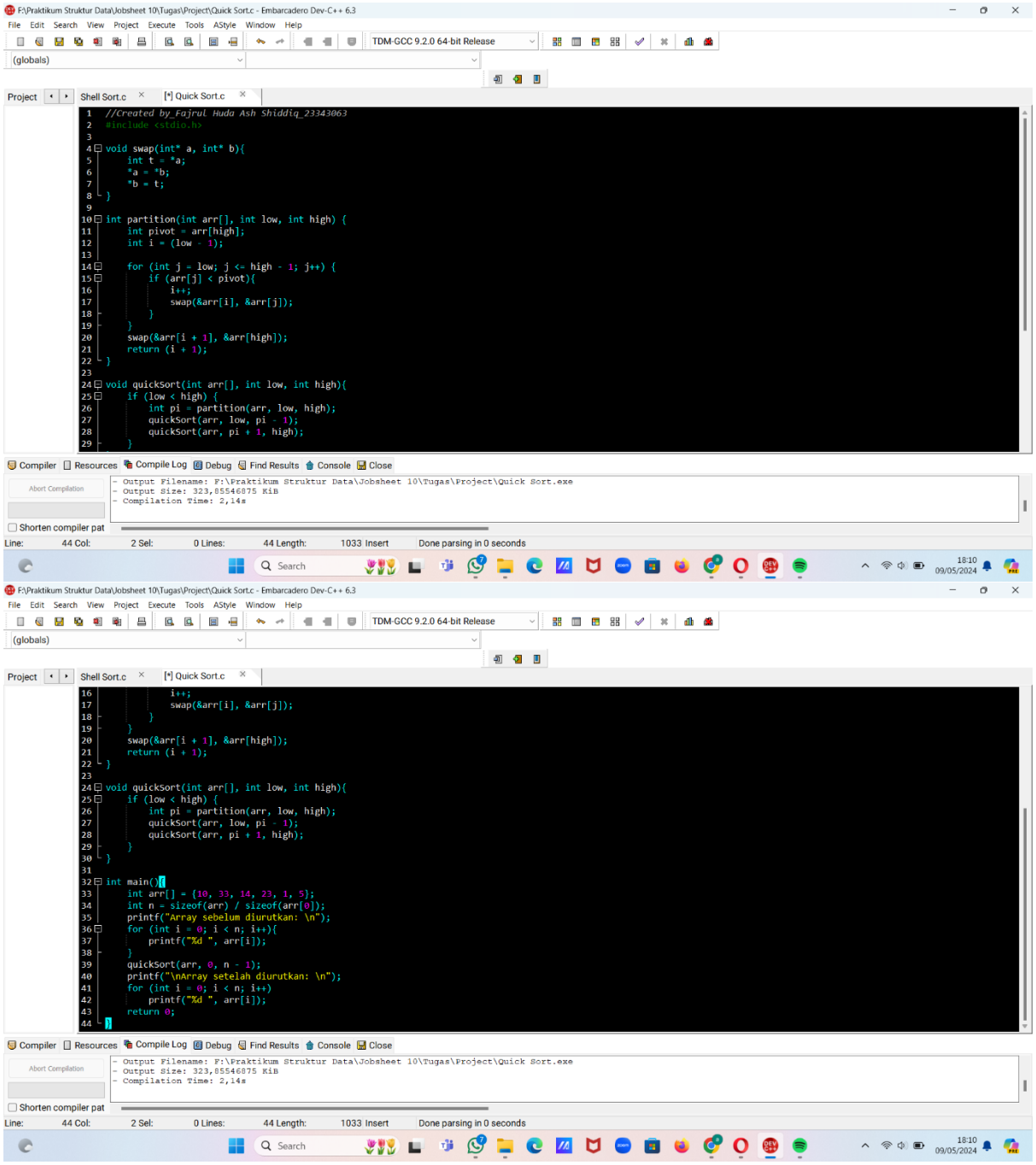
Quick Sort

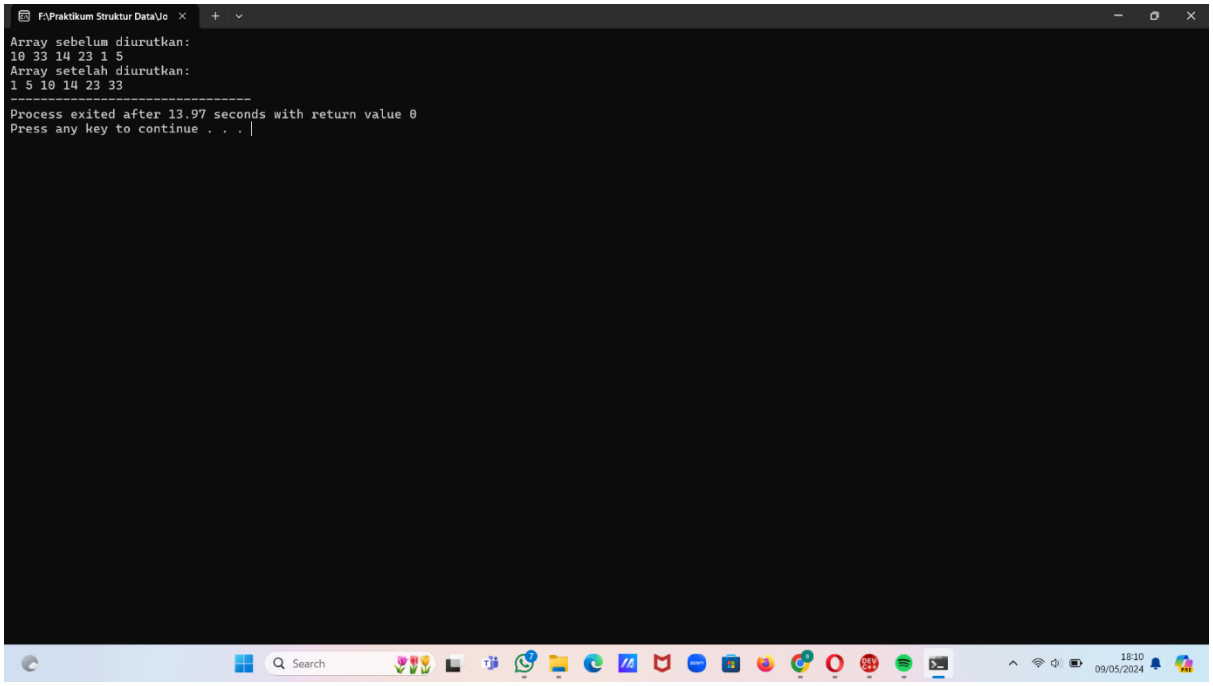
1. Source Code

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

void swap(int* a, int* b){
    int t = *a;
    *a = *b;
    *b = t;
}

int partition(int arr[], int low, int high) {
    int pivot = arr[high];
    int i = (low - 1);

    for (int j = low; j <= high - 1; j++) {
        if (arr[j] < pivot){
            i++;
            swap(&arr[i], &arr[j]);
        }
    }
    swap(&arr[i + 1], &arr[high]);
    return (i + 1);
}
```

```
void quickSort(int arr[], int low, int high){
    if (low < high) {
        int pi = partition(arr, low, high);
        quickSort(arr, low, pi - 1);
        quickSort(arr, pi + 1, high);
    }
}

int main(){
    int arr[] = {10, 33, 14, 23, 1, 5};
    int n = sizeof(arr) / sizeof(arr[0]);
    printf("Array sebelum diurutkan: \n");
    for (int i = 0; i < n; i++){
        printf("%d ", arr[i]);
    }
    quickSort(arr, 0, n - 1);
    printf("\nArray setelah diurutkan: \n");
    for (int i = 0; i < n; i++)
        printf("%d ", arr[i]);
    return 0;
}
```

2. Screenshot





3. Penjelasan

- Program dimulai dengan mendeklarasikan array yang akan diurutkan.
- Program memilih salah satu elemen array sebagai pivot. Pemilihan pivot bisa dilakukan dengan berbagai metode, salah satunya adalah memilih elemen terakhir.
- Setelah pivot dipilih, program membagi array menjadi dua bagian: satu bagian berisi elemen yang lebih kecil dari pivot, dan yang lainnya berisi elemen yang lebih besar. Ini dilakukan dengan memindahkan elemen-elemen tersebut ke sisi yang sesuai dengan pivot.
- Setelah pemisahan dilakukan, proses ini diulangi secara rekursif untuk setiap sub-array yang terbentuk dari pembagian sebelumnya. Hal ini terjadi pada kedua bagian array, yaitu bagian sebelum pivot dan bagian setelah pivot.
- Proses rekursif berhenti saat sub-array hanya memiliki satu elemen, karena satu elemen dianggap sudah terurut.
- Setelah seluruh proses rekursif selesai, seluruh array dianggap sudah terurut.