

## **Laporan Jobsheet 10**



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

**Kode Kelas : 202323430158**

**Disusun Oleh :**

**Fhandy Nofalino Akhsan  
23343065**

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

## **Pengantar**

Mata kuliah Praktek Struktur Data merupakan salah satu mata kuliah yang wajib diambil oleh mahasiswa teknik informatika. Dalam mata kuliah ini, kita akan belajar mengenai konsep-konsep dasar dari struktur data, seperti pointer, struct, dan array. Selain itu, kita juga akan mempelajari tentang beberapa jenis linked list, yaitu link list, double link list, dancircular link list. Berikut adalah ringkasan mengenai materi-materi tersebut.

Pointer, Struct, dan Array: Pada dasarnya, pointer, struct, dan array adalah konsep-konsep dasar dari bahasa pemrograman C. Pada praktik struktur data, kita akan mempelajari cara menggunakannya dalam membuat struktur data yang lebih kompleks.

## 1. Shell Sort

Shell Sort adalah varian dari insertion sort yang mempercepat proses dengan membandingkan elemen yang berjauhan. Elemen-elemen yang berjauhan diurutkan, dan jarak antara elemen yang dibandingkan secara bertahap berkurang hingga menjadi 1, pada saat mana algoritma ini menjadi insertion sort.

### Source Code

```
//created by Fhandy Nofalino Akhsan 23343065
```

```
#include <stdio.h>
```

```
// Fungsi untuk mengimplementasikan Shell Sort
```

```
void shellSort(int arr[], int n) {
```

```
    // Mulai dengan gap besar, lalu kurangi gap secara bertahap
```

```
    for (int gap = n/2; gap > 0; gap /= 2) {
```

```
        // Lakukan insertion sort untuk elemen-elemen dengan jarak gap
```

```
        for (int i = gap; i < n; i += 1) {
```

```
            int temp = arr[i];
```

```
            int j;
```

```
            // Geser elemen arr[0..i-gap] yang lebih besar dari temp
```

```
            // ke satu posisi ke depan dari posisi sekarang
```

```
            for (j = i; j >= gap && arr[j - gap] > temp; j -= gap)
```

```
                arr[j] = arr[j - gap];
```

```
            // Tempatkan temp (elemen awal arr[i]) di lokasi yang benar
```

```
            arr[j] = temp;
```

```
        }
```

```
    }
```

```
}
```

```
// Fungsi untuk mencetak array
```

```
void printArray(int arr[], int size) {
```

```

        for (int i = 0; i < size; i++)
            printf("%d ", arr[i]);
        printf("\n");
    }

// Fungsi utama
int main() {
    int arr[] = {12, 34, 54, 2, 3};
    int n = sizeof(arr)/sizeof(arr[0]);
    printf("Array sebelum disorting: \n");
    printArray(arr, n);

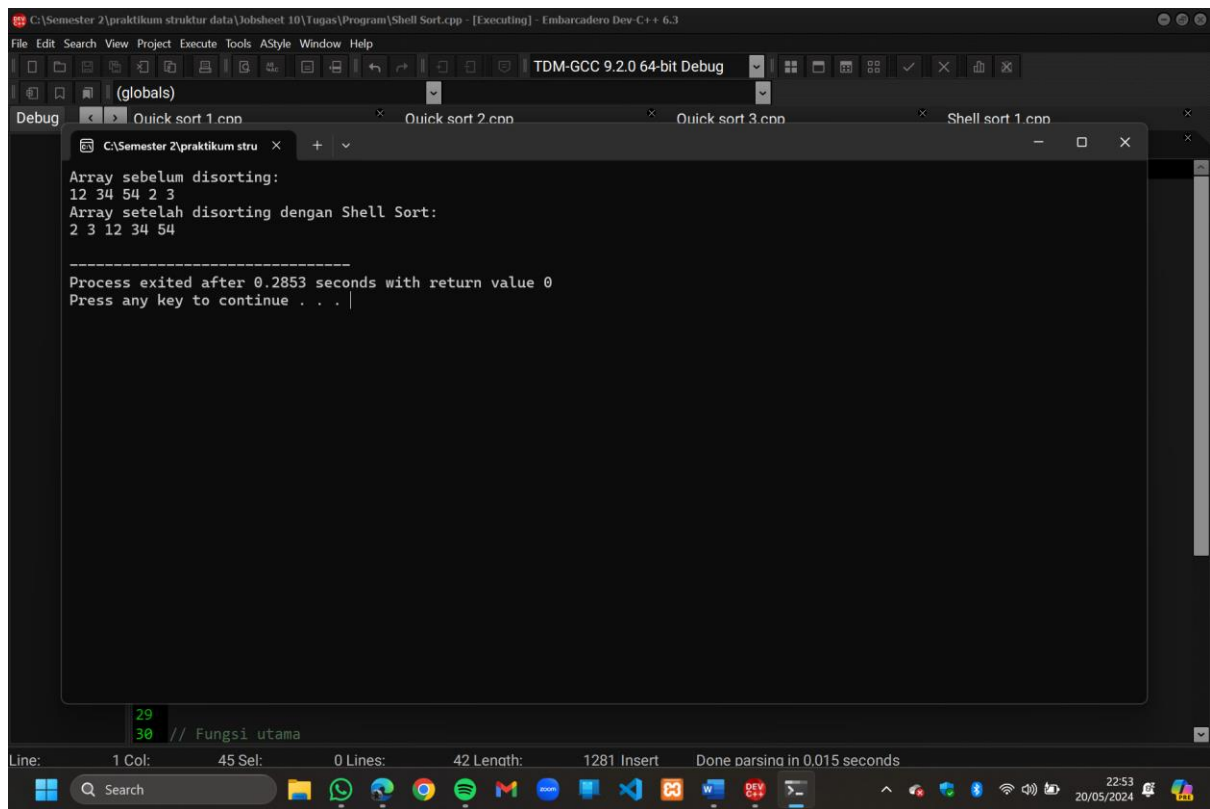
    shellSort(arr, n);
    printf("Array setelah disorting dengan Shell Sort: \n");
    printArray(arr, n);
    return 0;
}

```

### **Penjelasan**

1. Memilih gap awal yang besar, lalu mengurangi gap tersebut secara bertahap.
2. Melakukan insertion sort untuk elemen-elemen yang dipisahkan oleh gap tersebut.
3. Mengurangi gap dan mengulangi proses sampai gap menjadi 1.

## Output



```
C:\Semester 2\praktikum struktur data\Jobsheet 10\Tugas\Program\Shell Sort.cpp - [Executing] - Embarcadero Dev-C++ 6.3
File Edit Search View Project Execute Tools AStyle Window Help
TDM-GCC 9.2.0 64-bit Debug
(globals)
Debug Quick sort 1.cpp Quick sort 2.cpp Quick sort 3.cpp Shell sort 1.cpp
C:\Semester 2\praktikum stru
Array sebelum disorting:
12 34 54 2 3
Array setelah disorting dengan Shell Sort:
2 3 12 34 54

-----
Process exited after 0.2853 seconds with return value 0
Press any key to continue . . .
29
30 // Fungsi utama
```

Line: 1 Col: 45 Sel: 0 Lines: 42 Length: 1281 Insert Done parsing in 0.015 seconds

22:53 20/05/2024

## 2. Quick Sort

Quick Sort adalah algoritma yang menggunakan prinsip divide and conquer. Algoritma ini memilih sebuah pivot dan mengatur elemen-elemen dalam array sedemikian rupa sehingga elemen yang lebih kecil dari pivot berada di sebelah kiri dan elemen yang lebih besar berada di sebelah kanan. Proses ini kemudian diulang secara rekursif untuk sub-array di kiri dan kanan pivot.

### Source Code

```
//created by Fhandy Nofalino Akhsan 23343065
```

```
#include <stdio.h>
```

```
// Fungsi untuk menukar dua angka
```

```
void swap(int* a, int* b) {
```

```
    int t = *a;
```

```
    *a = *b;
```

```
    *b = t;
```

```
}
```

```
// Fungsi untuk mencari pivot yang tepat dan membagi array
```

```
int partition(int arr[], int low, int high) {
```

```
    int pivot = arr[high]; // pivot
```

```
    int i = (low - 1); // Indeks elemen yang lebih kecil
```

```
    for (int j = low; j <= high - 1; j++) {
```

```
        // Jika elemen saat ini lebih kecil dari atau sama dengan  
pivot
```

```
        if (arr[j] <= pivot) {
```

```
            i++; // Peningkatan indeks elemen yang lebih kecil
```

```
            swap(&arr[i], &arr[j]);
```

```
        }
```

```
    }
```

```

        swap(&arr[i + 1], &arr[high]);
        return (i + 1);
    }

// Fungsi utama untuk mengimplementasikan Quick Sort
void quickSort(int arr[], int low, int high) {
    if (low < high) {
        // pi adalah indeks pivot, arr[p] sudah pada tempat yang benar
        int pi = partition(arr, low, high);

        // Sorting rekursif elemen sebelum dan setelah partition
        quickSort(arr, low, pi - 1);
        quickSort(arr, pi + 1, high);
    }
}

// Fungsi untuk mencetak array
void printArray(int arr[], int size) {
    for (int i = 0; i < size; i++)
        printf("%d ", arr[i]);
    printf("\n");
}

// Fungsi utama
int main() {
    int arr[] = {10, 7, 8, 9, 1, 5};
    int n = sizeof(arr)/sizeof(arr[0]);
    printf("Array sebelum disorting: \n");
    printArray(arr, n);
}

```

```

    quickSort(arr, 0, n - 1);

    printf("Array setelah disorting dengan Quick Sort: \n");
    printArray(arr, n);

    return 0;
}

```

## Output

The screenshot shows a C++ IDE with the following output in the Debug window:

```

Array sebelum disorting:
10 7 8 9 1 5
Array setelah disorting dengan Quick Sort:
1 5 7 8 9 10
=====
Process exited after 0.561 seconds with return value 0
Press any key to continue . . .

```

The status bar at the bottom indicates the current line of code being executed is line 27, column 1.

## Penejlsan

- 1 Memilih pivot (di sini dipilih elemen terakhir dari array).
2. Membagi array menjadi dua sub-array, satu dengan elemen yang lebih kecil dari pivot dan satu lagi dengan elemen yang lebih besar.
3. Menukar elemen sesuai dengan pembagian tersebut.
4. Menerapkan quick sort secara rekursif pada sub-array yang terbentuk sampai seluruh array terurut.



## Kesimpulan

Shell Sort cenderung lebih efisien daripada insertion sort untuk array yang lebih besar karena elemen-elemen yang berjauhan lebih cepat terurut. Quick Sort sangat efisien untuk array besar dan digunakan secara luas karena kompleksitas rata-ratanya adalah  $O(n \log n)$ , meskipun dalam kasus terburuk bisa  $O(n^2)$  jika pivot yang dipilih tidak optimal.

## Referensi

1. [Algoritma Quick Sort: Pengertian, Kelebihan dan Contoh - DosenIT.comMerge Sort \(With Code in Python/C++/Java/C\) \(programiz.com\)](#)
2. [\(DOC\) Makalah Implementasi Shell Sort dan Quick Sort | Fedli Bagus Kurniawan - Academia.edu](#)
3. [Bab 1 \(pens.ac.id\)](#)
4. [Sebuah Kajian Pustaka: \(core.ac.uk\)](#)
5. [Quick Sort in C - GeeksforGeeks](#)
6. [Jurnal Ilmiah Komputer dan Informatika \(KOMPUTA\) \(kemdikbud.go.id\)](#)
7. [\[2023\] Job Sheet 10 - Shell and Quick Sort.pdf](#)