PRAKTIKUM STRUKTUR DATA
Tugas Jobsheet 8

Dosen Pengampu :
Randi Proska Sandra, M.Sc

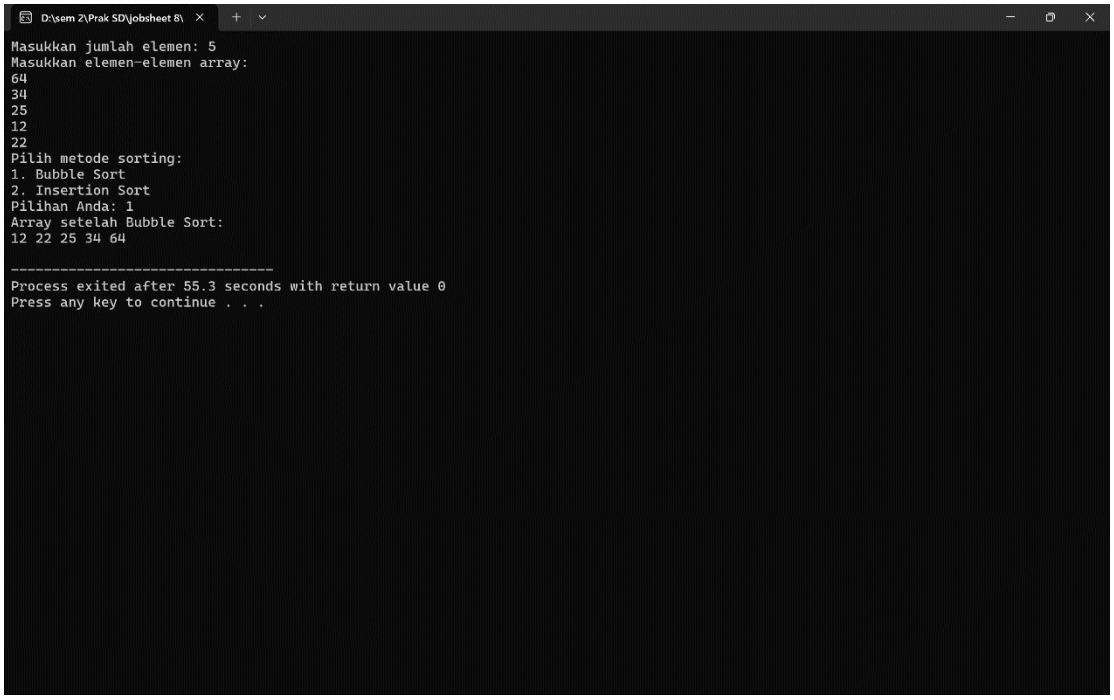
Disusun Oleh :

Joel Wiseda Simanungkalit
23343071

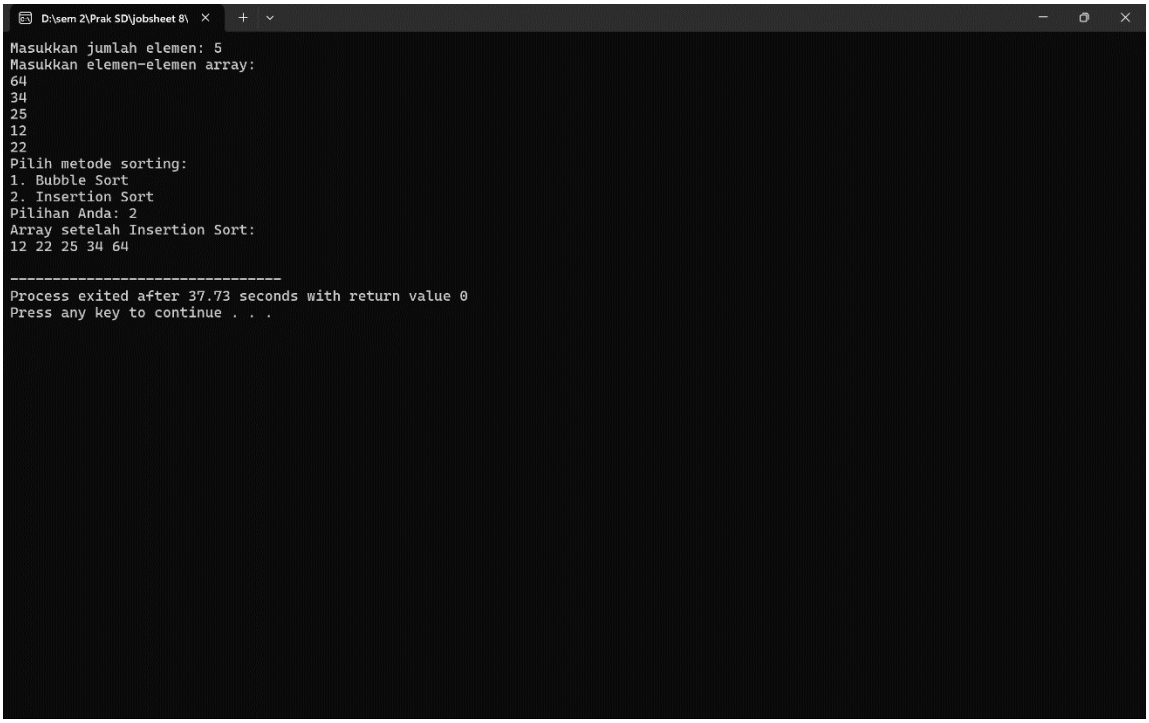
PROGRAM STUDI INFORMATIKA (NK)
FAKULTAS TEKNIK
UNIVERSITAS NEGERI PADANG
2024

A. Output

1. Bubble Sort



2. Insertion Sort



B. Penjelasan Prinsip Kerja Bubble Sort dan Insertion Sort

Bubble Sort

Bubble Sort adalah salah satu algoritma sorting paling sederhana. Prinsip kerja Bubble Sort dapat dijelaskan sebagai berikut:

1. **Perulangan Berulang Kali**: Algoritma ini mengulangi proses sorting dengan melewati daftar elemen yang akan diurutkan berkali-kali. Setiap kali melewati daftar, elemen-elemen bersebelahan dibandingkan dan ditukar jika urutannya salah. Proses ini disebut satu "pass".
2. **Perbandingan dan Penukaran**: Pada setiap pass, elemen-elemen bersebelahan dibandingkan satu per satu. Jika elemen di sebelah kiri lebih besar dari elemen di sebelah kanan, maka kedua elemen tersebut ditukar. Proses ini membuat elemen terbesar dalam daftar "menggelembung" ke posisi akhirnya pada akhir pass.
3. **Pengulangan hingga Terurut**: Proses perbandingan dan penukaran diulang hingga tidak ada lagi elemen yang perlu ditukar, yang berarti daftar sudah terurut. Dalam implementasi, ini biasanya diindikasikan oleh pass di mana tidak ada penukaran yang terjadi.

Pada program di atas, fungsi `bubbleSort` mengimplementasikan prinsip ini dengan menggunakan dua loop `for`. Loop pertama (`for (int i = 0; i < n-1; i++)`) mengontrol jumlah pass, sedangkan loop kedua (`for (int j = 0; j < n-i-1; j++)`) melakukan perbandingan dan penukaran elemen-elemen bersebelahan. Variabel `temp` digunakan untuk menyimpan sementara nilai elemen yang akan ditukar.

Kode Bubble Sort:

```
void bubbleSort(int arr[], int n) {
    int temp;
    for (int i = 0; i < n-1; i++) {
        for (int j = 0; j < n-i-1; j++) {
            if (arr[j] > arr[j+1]) {
                temp = arr[j];
                arr[j] = arr[j+1];
                arr[j+1] = temp;
            }
        }
    }
}
```

Insertion Sort

Insertion Sort adalah algoritma sorting yang lebih efisien dibandingkan Bubble Sort untuk jumlah elemen yang kecil. Prinsip kerja Insertion Sort dapat dijelaskan sebagai berikut:

1. **Pembagian Array**: Algoritma ini membagi array menjadi dua bagian: bagian yang sudah terurut dan bagian yang belum terurut. Pada awalnya, bagian yang terurut hanya berisi elemen pertama, dan bagian yang belum terurut berisi sisa elemen-elemen lainnya.
2. **Pengambilan dan Penempatan Elemen**: Elemen pertama dari bagian yang belum terurut diambil dan ditempatkan pada posisi yang tepat di bagian yang terurut. Ini dilakukan dengan cara membandingkan elemen yang diambil dengan elemen-elemen di bagian yang terurut dari belakang ke depan, dan menggeser elemen-elemen yang lebih besar ke kanan hingga posisi yang tepat ditemukan.
3. **Pengulangan hingga Terurut**: Proses ini diulangi untuk setiap elemen di bagian yang belum terurut hingga seluruh elemen dalam array sudah terurut.

Pada program di atas, fungsi `insertionSort` mengimplementasikan prinsip ini dengan menggunakan satu loop `for` yang berjalan dari elemen kedua hingga elemen terakhir (`for (int i = 1; i < n; i++)`). Dalam setiap iterasi, elemen saat ini (disebut `key`) dibandingkan dengan elemen-elemen di bagian yang terurut dan ditempatkan pada posisi yang tepat dengan menggunakan loop `while`.

Kode Insertion Sort:

```
void insertionSort(int arr[], int n) {
    int key, j;
    for (int i = 1; i < n; i++) {
        key = arr[i];
        j = i - 1;

        while (j >= 0 && arr[j] > key) {
            arr[j + 1] = arr[j];
            j = j - 1;
        }
        arr[j + 1] = key;
    }
}
```

Ilustrasi Proses Bubble Sort

Misalkan kita memiliki array `[64, 34, 25, 12, 22]`.

Pass 1:

- Bandingkan 64 dan 34. Tukar → `[34, 64, 25, 12, 22]`
- Bandingkan 64 dan 25. Tukar → `[34, 25, 64, 12, 22]`
- Bandingkan 64 dan 12. Tukar → `[34, 25, 12, 64, 22]`
- Bandingkan 64 dan 22. Tukar → `[34, 25, 12, 22, 64]`

Pass 2:

- Bandingkan 34 dan 25. Tukar → `[25, 34, 12, 22, 64]`
- Bandingkan 34 dan 12. Tukar → `[25, 12, 34, 22, 64]`

- Bandingkan 34 dan 22. Tukar → `[25, 12, 22, 34, 64]`
- Tidak perlu membandingkan 34 dan 64 karena sudah di tempatnya.

Pass 3:

- Bandingkan 25 dan 12. Tukar → `[12, 25, 22, 34, 64]`
- Bandingkan 25 dan 22. Tukar → `[12, 22, 25, 34, 64]`
- Tidak perlu membandingkan 25 dan 34 karena sudah di tempatnya.

Pass 4:

- Bandingkan 12 dan 22. Tidak perlu tukar.
- Tidak perlu membandingkan 22 dan 25 karena sudah di tempatnya.

Array sudah terurut: `[12, 22, 25, 34, 64]`.

Ilustrasi Proses Insertion Sort

Misalkan kita memiliki array `[64, 34, 25, 12, 22]`.

Iteration 1:

- Elemen `34` dibandingkan dengan `64`. Tukar → `[34, 64, 25, 12, 22]`

Iteration 2:

- Elemen `25` dibandingkan dengan `64`. Geser `64` ke kanan → `[34, 64, 64, 12, 22]`
- Elemen `25` dibandingkan dengan `34`. Geser `34` ke kanan → `[34, 34, 64, 12, 22]`
- Tempatkan `25` di posisi yang tepat → `[25, 34, 64, 12, 22]`

Iteration 3:

- Elemen `12` dibandingkan dengan `64`. Geser `64` ke kanan → `[25, 34, 64, 64, 22]`
- Elemen `12` dibandingkan dengan `34`. Geser `34` ke kanan → `[25, 34, 34, 64, 22]`
- Elemen `12` dibandingkan dengan `25`. Geser `25` ke kanan → `[25, 25, 34, 64, 22]`
- Tempatkan `12` di posisi yang tepat → `[12, 25, 34, 64, 22]`

Iteration 4:

- Elemen `22` dibandingkan dengan `64`. Geser `64` ke kanan → `[12, 25, 34, 64, 64]`
- Elemen `22` dibandingkan dengan `34`. Geser `34` ke kanan → `[12, 25, 34, 34, 64]`
- Elemen `22` dibandingkan dengan `25`. Geser `25` ke kanan → `[12, 25, 25, 34, 64]`
- Tempatkan `22` di posisi yang tepat → `[12, 22, 25, 34, 64]`

Array sudah terurut: `[12, 22, 25, 34, 64]`.

Kesimpulan

Bubble Sort dan Insertion Sort adalah algoritma sorting yang sederhana dan mudah diimplementasikan. Bubble Sort bekerja dengan cara membandingkan dan menukar elemen-elemen bersebelahan secara berulang hingga seluruh elemen terurut. Insertion Sort bekerja dengan cara mengurutkan elemen-elemen satu per satu dengan menempatkannya pada posisi yang tepat dalam bagian yang sudah terurut dari array. Meskipun keduanya tidak efisien untuk jumlah elemen yang besar, mereka cukup efektif dan mudah dipahami untuk jumlah elemen yang kecil.