PRAKTIKUM 9
ALGORITMA PENGURUTAN (INSERTION SORT)

A. TUJUAN PEMBELAJARAN

1. Memahami step by step algoritma pengurutan *insertion sort.*
2. Mampu mengimplementasikan algoritma pengurutan *insertion sort* dengan berbagai macam parameter berupa tipe data primitif atau tipe Generic.
3. Mampu mengimplementasikan algoritma pengurutan *insertion sort* secara *ascending* dan *descending.*

B. DASAR TEORI

Algoritma *Insertion Sort*

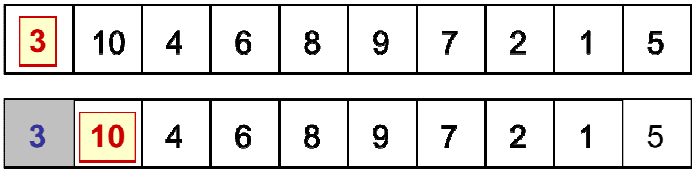
Salah satu algoritma sorting yang paling sederhana adalah *insertion sort*. Algoritma *insertion sort* pada dasarnya memilah data yang akan urutkan menjadi 2 bagian, yang belum diurutkan dan yang sudah diurutkan. Elemen pertama diambil dari bagian array yang belum diurutkan dan kemudian diletakkan sesuai posisinya pada bagian lain dari array yang telah diurutkan. Langkah ini dilakukan secara berulang hingga tak ada lagi elemen tersisa pada bagian array yang belum diurutka.

Metode insection sort merupakan metode yang mengurutkan bilangan-bilangan yang telah terbaca, dan berikutnya secara berulang akan menyisipkan bilangan-bilangan dalam array yang belum terbaca ke sisi kiri array yang telah terurut. Kita mengambil pada bilangan yang paling kiri. Bilangan tersebut dikatakan urut terhadap dirinya sendiri karena bilangan yang di bandingkan baru 1.

3	10	4	6	8	9	7	2	1	5

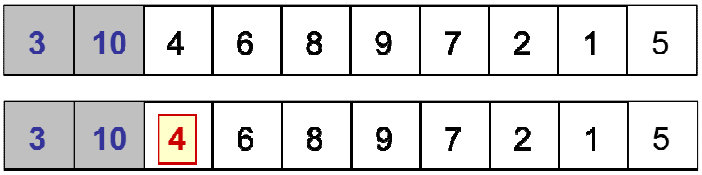
Gambar 1. Langkah-langkah pengurutan metode Insertion Sort (1)

Cek bilangan ke 2 (10) apakah lebih kecil dari bilangan yang ke 1(3).Apabila lebih kecil maka ditukar. Tapi kali ini bilangan ke 1 lebih kecil dari bilangan ke 2 maka tidak ditukar.



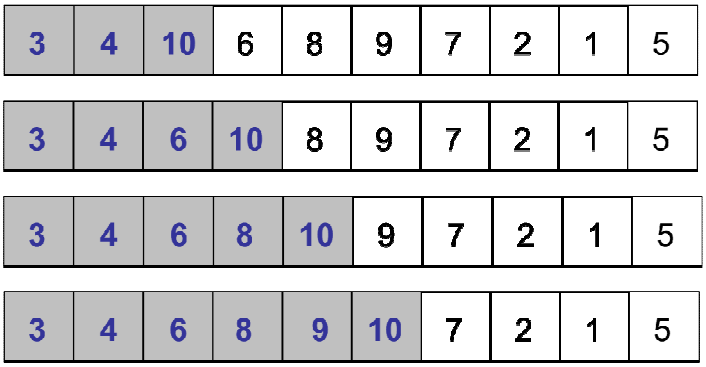
Gambar 2. Langkah-langkah pengurutan metode Insertion Sort (2)

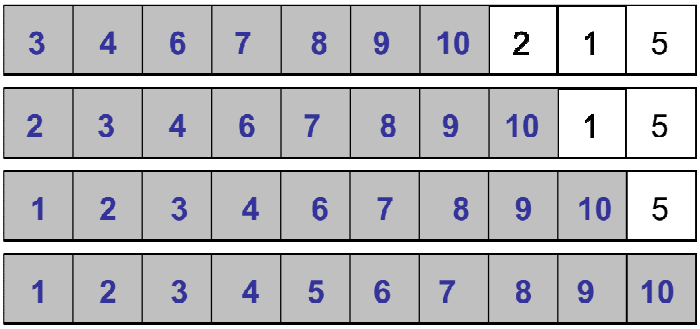
Pada kotak warna abu2 sudah dalam keadaan terurut. Kemudian membandingkan lagi pada bilangan selanjutnya yaitu bilangan ke 3 (4). Bandingkan dengan bilangan yang ada di sebelah kirinya. Pada kasus ini bilangan ke 2 bergeser dan digantikan bilangan ke 3.



Gambar 3. Langkah-langkah pengurutan metode Insertion Sort (3)

Lakukan langkah seperti di atas pada bilangan selanjutnya. Empat bilangan pertama sudah dalam keadaan terurut relatif. Ulangi proses tersebut sampai bilangan terakhir disisipkan.





Gambar 4. Langkah-langkah pengurutan metode Insertion Sort (4)

C. TUGAS PENDAHULUAN

Jelaskan algoritma pengurutan *insertion sort* secara *ascending* dengan data 5 6 3 1 2

D. PERCOBAAN

Percobaan 1 : *Insertion sort* secara *ascending* dengan data int

```
public class InsertionDemo {

    public static void insertionSort(int[] A) {
        for (int i = 1; i < A.length; i++) {
            int key = A[i];
            int j = i - 1;
            while ((j >= 0) && (A[j] > key)) {
                A[j + 1] = A[j];
                j--;
            }
            A[j + 1] = key;
        }
    }

    public static void tampil(int data[]) {
        for (int i = 0; i < data.length; i++) {
            System.out.print(data[i] + " ");
        }
        System.out.println();
    }
}

public class Main {
    public static void main(String[] args) {
        int A[] = {10,6,8,3,1};
        InsertionDemo.tampil(A);
        InsertionDemo.insertionSort(A);
        InsertionDemo.tampil(A);

    }
```

```
}
```

Percobaan 2 : *Insertion sort* secara *descending* dengan data int

```
public class InsertionDemo {

    public static void insertionSort(int[] A) {
        for (int i = 1; i < A.length; i++) {
            int key = A[i];
            int j = i - 1;
            while ((j >= 0) && (A[j] < key)) {
                A[j + 1] = A[j];
                j--;
            }
            A[j + 1] = key;
        }
    }

    public static void tampil(int data[]) {
        for (int i = 0; i < data.length; i++) {
            System.out.print(data[i] + " ");
        }
        System.out.println();
    }
}

public class Main {
    public static void main(String[] args) {
        int A[] = {10,6,8,3,1};
        InsertionDemo.tampil(A);
        InsertionDemo.insertionSort(A);
        InsertionDemo.tampil(A);

    }
}
```

Percobaan 3 : *Insertion sort* secara *ascending* dengan data double

```
public class InsertionDemo {
   public static void insertionSort(double[] A) {
        for (int i = 1; i < A.length; i++) {
            double key = A[i];
            int j = i - 1;
            while ((j >= 0) && (A[j] > key)) {
                A[j + 1] = A[j];
                j--;
            }
            A[j + 1] = key;
        }
    }

    public static void tampil(double data[]) {
```

```
        for (int i = 0; i < data.length; i++) {
            System.out.print(data[i] + " ");
        }
        System.out.println();
    }
}

public class Main2 {
    public static void main(String[] args) {
        double A[] = {10.3,6.2,8.4,3.6,1.1};
        InsertionDemo.tampil(A);
        InsertionDemo.insertionSort(A);
        InsertionDemo.tampil(A);
    }
}
```

E. LATIHAN

1. Buatlah program sorting Insertion dengan parameter array Integer (class Wrapper) !

```
public static void insertionSort(Integer[] A){…}
```

2. Buatlah program sorting Insertion dengan parameter array Double (class Wrapper) !

```
public static void insertionSort(Double[] A){…}
```

3. Buatlah fungsi `tampil()` untuk menampilkan data.

```
public static<T> void tampil(T data[]){ }
```

4. Lakukan pengujian fungsi insertionSort(), dengan membuat fungsi main() sebagai berikut :

```
public class Demo1 {
    public static void main(String[] args) {
        //Data Integer
        Integer arr3[] = {1,5,6,2,8,9};
        InsertionDemo.insertionSort(arr3);
        InsertionDemo.tampil(arr3);

        //data Double
        Double arr4[] = {1.3,5.2,6.6,2.7,8.8,9.1};
        InsertionDemo.insertionSort(arr4);
        InsertionDemo.tampil(arr4);
    }
}
```

5. Buatlah program sorting Insertion dengan parameter array Number !

```
public static<T extends Number> void insertionSort(T[] A){…}
```

6. Lakukan pengujian fungsi insertionSort(), dengan membuat fungsi main() sebagai berikut :

```
public class Demo2 {
    public static void main(String[] args) {
        Float arr5[] = {1.3f,5.2f,6.6f,2.7f,8.8f,9.1f};
        InsertionDemo.insertionSort(arr5);
        InsertionDemo.tampil(arr5);

        Byte arr6[] = {6,7,11,1,3,2};
        InsertionDemo.insertionSort(arr6);
        InsertionDemo.tampil(arr6);
    }
}
```

7. Buatlah program sorting Insertion dengan parameter array yang memenuhi T extends Comparable !

```
public static <T extends Comparable> void insertionSort2(T[]
arr) { }
```

8. Lakukan pengujian fungsi insertionSort(), dengan membuat fungsi main() sebagai berikut :

```
public class Demo3 {
    public static void main(String[] args) {
        //data String
        String arr7[]=
{"jeruk","anggur","belimbing","jambu","kelengkeng"};
        InsertionDemo.insertionSort2(arr7);
        InsertionDemo.tampil(arr7);
    }
}
```

9. Buatlah class Mahasiswa dengan variable nrp dan nama yang memiliki tipe String ! Class Mahasiswa mengimplementasikan interface Comparable, selanjutnya implementasikan fungsi abstract compareTo(), untuk membandingkan dua objek mahasiswa berdasarkan nrp.

```
public class Mahasiswa implements Comparable <Mahasiswa> {
    private String nrp ;
    private String nama ;
    @Override
    public int compareTo(Mahasiswa o) {…}

    @Override
    public String toString() {…}
}
```

10. Lakukan pengujian fungsi insertionSort() lagi, sebelumnya tambahkan pada fungsi main() seperti di bawah ini !

```
public class Demo4 {
    public static void main(String[] args) {
        Mahasiswa arr8[] = {new Mahasiswa("02", "Budi"), new
Mahasiswa("01", "Andi"), new Mahasiswa("04", "Udin"), new Mahasiswa("03",
"Candra")};
        InsertionDemo.insertionSort2(arr8);
        InsertionDemo.tampil(arr8);
    }
}
```

F. LAPORAN RESMI

Kerjakan hasil percobaan(D) dan latihan(E) di atas dan tambahkan analisa.