PRAKTIKUM
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*

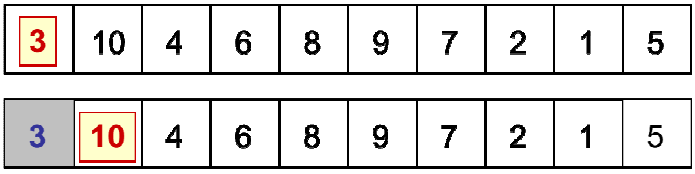
Salah satu algoritma sorting yang paling sederhana adalah *insertion sort*. Algoritma *insertion sort* pada dasarnya memilah data yang akan urutkan menjadi 2 bagian, yang belum diurutkan dan yang sudah diurutkan. Elemen pertama diambil dari bagian array yang belum diurutkan dan kemudian diletakkan sesuai 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

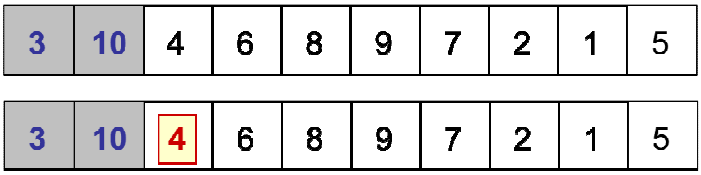
Gambar 1. Langkah-langkah pengurutan metode Insertion Sort (1)

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.



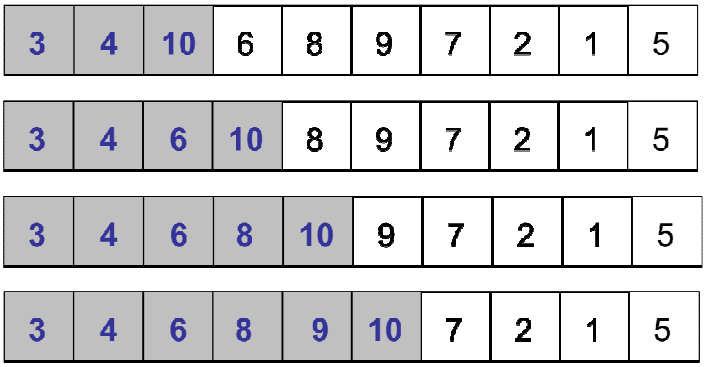
Gambar 2. Langkah-langkah pengurutan metode Insertion Sort (2)

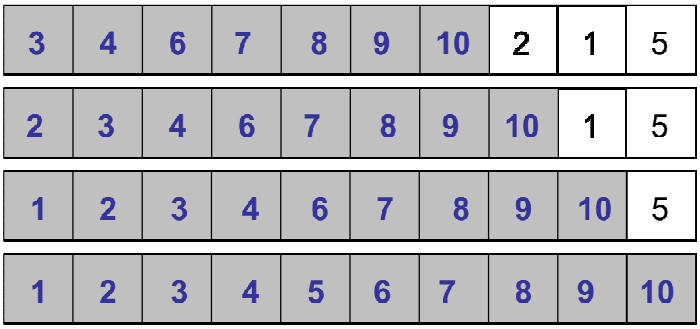
Pada kotak warna abu2 sudah dalam keadaan terurut. Kemudian membandingkan lagi pada bilangan selanjutnya yaitu bilangan ke 3 (4). Bandingkan dengan bilangan yang ada di sebelah kirinya. Pada kasus ini bilangan ke 2 bergeser dan digantikan bilangan ke 3.



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);
    }
}
```

PRAKTIKUM
ALGORITMA PENGURUTAN (SELECTION SORT)

A. TUJUAN PEMBELAJARAN

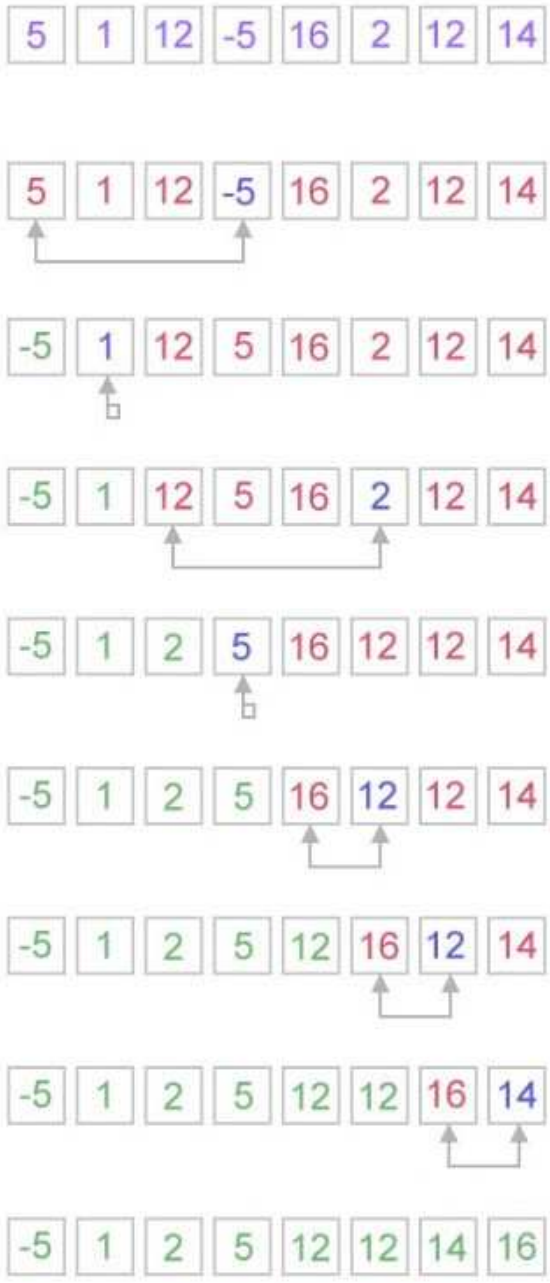
1. Memahami step by step algoritma pengurutan *selection sort.*
2. Mampu mengimplementasikan algoritma pengurutan *selection sort* dengan berbagai macam parameter berupa tipe data primitif atau tipe Generic.
3. Mampu mengimplementasikan algoritma pengurutan *selection sort* secara *ascending* dan *descending.*

B. DASAR TEORI

Algoritma *Selection Sort*

Ide utama adalah pada data indeks ke-0, dibandingkan dengan data sesudahnya untuk mencari elemen yang paling kecil, selanjutnya elemen terkecil tersebut ditukar dengan elemen pada indeks ke-0. Selanjutnya data indeks ke-1, dibandingkan dengan data sesudahnya untuk mencari elemen yang paling kecil, selanjutnya elemen terkecil tersebut ditukar dengan elemen pada indeks ke-1, dan seterusnya sampai data terurut secara ascending.

Contoh terdapat data 5, 1, 12, -5, 16, 2, 12, 14. Data acuan pada indek ke-0 yaitu 5 dibandingkan dengan data sesudahnya untuk mencari elemen terkecil. Elemen terkecil setelah 5 adalah -5, sehingga 5 ditukar dengan -5, sehingga data menjadi -5, 1, 12, 5, 16, 2, 12, 14. Data acuan berikutnya adalah indek ke-1 yaitu 1 dibandingkan dengan data sesudahnya untuk mencari elemen terkecil. Elemen terkecil setelah 1 ternyata adalah 1, sehingga 1 ditukar dengan 1, sehingga data menjadi -5, 1, 12, 5, 16, 2, 12, 14 dan seterusnya.


5 1 12 -5 16 2 12 14
5 1 12 -5 16 2 12 14
-5 1 12 5 16 2 12 14
-5 1 12 5 16 2 12 14
-5 1 2 5 16 12 12 14
-5 1 2 5 16 12 12 14
-5 1 2 5 12 16 12 14
-5 1 2 5 12 12 16 14
-5 1 2 5 12 12 14 16

C. TUGAS PENDAHULUAN

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

D. PERCOBAAN

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

```
public class SelectionDemo {
    public static void selectionSort(int[] arr) {
        // index of smallest element in the sublist
        int smallIndex;
        int pass, j, n = arr.length;
        int temp;

        for (pass = 0; pass < n - 1; pass++) {
            smallIndex = pass;
            for (j = pass + 1; j < n; j++)
            {
                if (arr[j]< arr[smallIndex]) {
                    smallIndex = j;
                }
            }
            // tukar nilai terkecil dengan arr[pass]
            temp = arr[pass];
            arr[pass] = arr[smallIndex];
            arr[smallIndex] = temp;
        }
    }

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

public class MainSelection {
    public static void main(String[] args) {
        int A[] = {10,6,8,3,1};
        SelectionDemo.tampil(A);
        SelectionDemo.selectionSort(A);
        SelectionDemo.tampil(A);
    }
}
```

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

```
public class SelectionDemo {
    public static void selectionSort(int[] arr) {
```

```
        // index of smallest element in the sublist
        int smallIndex;
        int pass, j, n = arr.length;
        int temp;

        for (pass = 0; pass < n - 1; pass++) {
            smallIndex = pass;
            for (j = pass + 1; j < n; j++)
            {
                if (arr[j]> arr[smallIndex]) {
                    smallIndex = j;
                }
            }
            // tukar nilai terkecil dengan arr[pass]
            temp = arr[pass];
            arr[pass] = arr[smallIndex];
            arr[smallIndex] = temp;
        }
    }

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

public class MainSelection {
    public static void main(String[] args) {
        int A[] = {10,6,8,3,1};
        SelectionDemo.tampil(A);
        SelectionDemo.selectionSort(A);
        SelectionDemo.tampil(A);
    }
}
```

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

```
public static void selectionSort(double[] arr) {
        // index of smallest element in the sublist
        int smallIndex;
        int pass, j, n = arr.length;
        double temp;

        for (pass = 0; pass < n - 1; pass++) {
            smallIndex = pass;
            for (j = pass + 1; j < n; j++)
            {
                if (arr[j]> arr[smallIndex]) {
                    smallIndex = j;
                }
            }
```

```
            // tukar nilai terkecil dengan arr[pass]
            temp = arr[pass];
            arr[pass] = arr[smallIndex];
            arr[smallIndex] = temp;
        }
    }

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

    public class MainSelection2 {
    public static void main(String[] args) {
        double A[] = {10.3,6.2,8.4,3.6,1.1};
        SelectionDemo.tampil(A);
        SelectionDemo.selectionSort(A);
        SelectionDemo.tampil(A);
    }
}
```

PRAKTIKUM
ALGORITMA PENGURUTAN (BUBBLE SORT)

A. TUJUAN PEMBELAJARAN

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

B. DASAR TEORI

Algoritma *Bubble Sort*

Metode gelembung (*bubble sort*) sering juga disebut dengan metode penukaran (*exchange sort*) adalah metode yang mengurutkan data dengan cara membandingkan masing-masing elemen, kemudian melakukan penukaran bila perlu. Metode ini mudah dipahami dan diprogram, tetapi bila dibandingkan dengan metode lain yang kita pelajari, metode ini merupakan metode yang paling tidak efisien.

Proses pengurutan metode gelembung ini menggunakan dua kalang. Kalang pertama melakukan pengulangan dari elemen ke 1 sampai dengan elemen ke N-1 (misalnya variable i), sedangkan kalang kedua melakukan pengulangan menurun dari elemen ke N-1 sampai elemen ke i (misalnya variable j). Pada setiap pengulangan, elemen ke j-1 dibandingkan dengan elemen ke j. Apabila data ke j-1 lebih besar daripada data ke j, dilakukan penukaran.

Algoritma gelembung dapat dituliskan sebagai berikut :

```
1    i ← 0
2    selama (i < N) kerjakan baris 3 sampai dengan 7
3    j ← N - 1
```

4 Selama (j >= i) kerjakan baris 5 sampai dengan 7

5 Jika (Data[j-1] > Data[j]) maka tukar Data[j-1] dengan Data[j]

6 $j \leftarrow j - 1$

7 $i \leftarrow i + 1$

Untuk lebih memperjelas langkah-langkah algoritma gelembung dapat dilihat pada tabel 6.3. Proses pengurutan pada tabel 6.3 dapat dijelaskan sebagai berikut:

- Pada saat i=1, nilai j diulang dari 9 sampai dengan 1. Pada pengulangan pertama Data[9] dibandingkan Data[8], karena 20 < 31 maka Data[9] dan Data[8] ditukar. Pada pengulangan kedua Data[8] dibandingkan Data[7], karena 20 > 15 maka proses dilanjutkan. Demikian seterusnya sampai j=1.
- Pada saat i=2, nilai j diulang dari 9 sampai dengan 2. Pada pengulangan pertama Data[9] dibandingkan Data[8], karena 31 > 20 maka proses dilanjutkan. Pada pengulangan kedua Data[8] dibandingkan Data[7], karena 20 < 23 Data[8] dan Data[7] ditukar. Demikian seterusnya sampai j=2.
- Pada saat i=3, nilai j diulang dari 9 sampai dengan 3. Pada pengulangan pertama Data[9] dibandingkan Data[8], karena 31 > 23 maka proses dilanjutkan. Pada pengulangan kedua Data[8] dibandingkan Data[7], karena 23 > 20 maka proses dilanjutkan. Demikian seterusnya sampai j=3.
- Dan seterusnya sampai dengan i=8.

Tabel 1 Proses Pengurutan dengan Metode Gelembung

Iterasi	Data[0]	Data[1]	Data[2]	Data[3]	Data[4]	Data[5]	Data[6]	Data[7]	Data[8]	Data[9]
Awal	12	35	9	11	3	17	23	15	31	20
i=1; j=9	12	35	9	11	3	17	23	15	**31**	**20**
j=8	12	35	9	11	3	17	23	**15**	**20**	31
j=7	12	35	9	11	3	17	**23**	**15**	20	31
j=6	12	35	9	11	3	**17**	**15**	23	20	31
j=5	12	35	9	11	**3**	**15**	17	23	20	31
j=4	12	35	9	**11**	**3**	15	17	23	20	31
j=3	12	35	**9**	**3**	11	15	17	23	20	31
j=2	12	**35**	**3**	9	11	15	17	23	20	31

j=1	**12**	**3**	35	9	11	15	17	23	20	31
i=2; j=9	3	12	35	9	11	15	17	23	**20**	**31**
j=8	3	12	35	9	11	15	17	**23**	**20**	31
j=7	3	12	35	9	11	15	**17**	**20**	23	31
j=6	3	12	35	9	11	**15**	**17**	20	23	31
j=5	3	12	35	9	**11**	**15**	17	20	23	31
j=4	3	12	35	**9**	**11**	15	17	20	23	31
j=3	3	12	**35**	**9**	11	15	17	20	23	31
j=2	3	**12**	**9**	35	11	15	17	20	23	31
i=3; j=9	3	9	12	35	11	15	17	20	**23**	**31**
j=8	3	9	12	35	11	15	17	**20**	**23**	31
j=7	3	9	12	35	11	15	**17**	**20**	23	31
j=6	3	9	12	35	11	**15**	**17**	20	23	31
j=5	3	9	12	35	**11**	**15**	17	20	23	31
j=4	3	9	12	**35**	**11**	15	17	20	23	31
j=3	3	9	**12**	**11**	35	15	17	20	23	31
i=4; j=9	3	9	11	12	35	15	17	20	**23**	**31**
j=8	3	9	11	12	35	15	17	**20**	**23**	31
j=7	3	9	11	12	35	15	**17**	**20**	23	31
j=6	3	9	11	12	35	**15**	**17**	20	23	31
j=5	3	9	11	12	**35**	**15**	17	20	23	31
j=4	3	9	11	**12**	**15**	35	17	20	23	31
i=5; j=9	3	9	11	12	15	35	17	20	**23**	**31**
j=8	3	9	11	12	15	35	17	**20**	**23**	31
j=7	3	9	11	12	15	35	**17**	**20**	23	31
j=6	3	9	11	12	15	**35**	**17**	20	23	31
j=5	3	9	11	12	**15**	**17**	35	20	23	31
i=6; j=9	3	9	11	12	15	17	35	20	**23**	**31**
j=8	3	9	11	12	15	17	35	**20**	**23**	31
j=7	3	9	11	12	15	17	**35**	**20**	23	31
j=6	3	9	11	12	15	**17**	**20**	35	23	31
i=7; j=9	3	9	11	12	15	17	20	35	**23**	**31**
j=8	3	9	11	12	15	17	20	**35**	**23**	31

j=7	3	9	11	12	15	17	**20**	**23**	35	31
i=8; j=9	3	9	11	12	15	17	20	23	**35**	**31**
j=8	3	9	11	12	15	17	20	**23**	**31**	35
i=9 j=9	3	9	11	12	15	17	20	23	**31**	**35**
Akhir	3	9	11	12	15	17	20	23	31	35

C. TUGAS PENDAHULUAN

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

D. PERCOBAAN

Percobaan 1 : *Bubble sort* secara *ascending* dengan data int.

```
public class BubbleDemo {
    public static void bubbleSort(int[] arr) {
        int i=1, j, n = arr.length;
        int temp;

        while (i<n){
            j = n-1 ;
            while(j>=i){
                if (arr[j-1]>arr[j]){
                    temp = arr[j];
                    arr[j] = arr[j-1];
                    arr[j-1] = temp;
                }
                j = j - 1;
            }
            i = i + 1;
        }
    }

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

public class MainBubble {
    public static void main(String[] args) {
        int A[] = {10,6,8,3,1};
        BubbleDemo.tampil(A);
        BubbleDemo.bubbleSort(A);
        BubbleDemo.tampil(A);
    }
```

```
}
```

Percobaan 2 : *Bubble sort* secara *descending* dengan data int.

```
public class BubbleDemo {
    public static void bubbleSort(int[] arr) {
        int i=1, j, n = arr.length;
        int temp;

        while (i<n){
            j = n-1 ;
            while(j>=i){
                if (arr[j-1] < arr[j]){
                    temp = arr[j];
                    arr[j] = arr[j-1];
                    arr[j-1] = temp;
                }
                j = j - 1;
            }
            i = i + 1;
        }
    }

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

public class MainBubble {
    public static void main(String[] args) {
        int A[] = {10,6,8,3,1};
        BubbleDemo.tampil(A);
        BubbleDemo.bubbleSort(A);
        BubbleDemo.tampil(A);
    }
}
```

Percobaan 3 : *Bubble sort* secara *ascending* dengan data double.

```
public class BubbleDemo {
    public static void bubbleSort(double[] arr) {
        int i=1, j, n = arr.length;
        double temp;

        while (i<n){
            j = n-1 ;
            while(j>=i){
                if (arr[j-1]>arr[j]){
                    temp = arr[j];
```

```
                    arr[j] = arr[j-1];
                    arr[j-1] = temp;
                }
                j = j - 1;
            }
            i = i + 1;
        }
    }

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

public class MainBubble2 {
    public static void main(String[] args) {
        double A[] = {10.3,6.2,8.4,3.6,1.1};
        BubbleDemo.tampil(A);
        BubbleDemo.bubbleSort(A);
        BubbleDemo.tampil(A);
    }
}
```