14 SORTING

Sorting atau pengurutan data adalah proses untuk menyusun kumpulan data yang seragam menjadi susunan tertentu. Kumpulan data dapat diurutkan secara *Ascending*(Urut Menaik), yaitu dari data yang nilainya paling kecil sampai data yang nilainya paling besar, atau diurutkan secara *Descending*(Urut Menurun), yaitu dari data yang nilainya paling besar sampai data yang nilainya paling kecil.

Metode-metode Sorting :

1. Bubble Sort

Pengurutan model ini mengambil ide dari gelembung air, yaitu mengapungkan elemen yang bernilai kecil dari bawah ke atas. Proses pengapungan dilakukan dengan pertukaran elemen-elemen tabel.

Apabila kita menginginkan array terurut menaik, maka elemen array yang berharga paling kecil ”diapungkan” artinya diangkat ke ”atas” (atau ke ujung kiri array) melalui proses pertukaran. Proses pengapungan ini dilakukan sebanyak n-1 langkah(satu langkah disebut satu kali *pass*) dengan n adalah ukuran array.

Ilustrasi Kasus :

Perhatikan array *TabInt* di bawah ini yang terdiri dari n = 6 elemen yang belum terurut. Array ini akan diurutkan **menaik** dengan metode **Bubble Sort.**

Elemen Array	**25**	**27**	**10**	**8**	**76**	**21**
Indeks	1	2	3	4	5	6

← Arah pembandingan

Pass 1 :

25	**27**	**10**	**8**	**21**	**76**
1	2	3	4	5	6

25	**27**	**10**	**8**	**21**	**76**

25	**27**	**8**	**10**	**21**	**76**

25	**8**	**27**	**10**	**21**	**76**

8	**25**	**27**	**10**	**21**	**76**

Hasil Akhir *Pass 1* :

8	**25**	**27**	**10**	**21**	**76**
1	2	3	4	5	6

Pass 2 : (Berdasarkan hasil akhir *Pass 1*)

8	**25**	**27**	**10**	**21**	**76**

8	**25**	**27**	**10**	**21**	**76**

8	**25**	**10**	**27**	**21**	**76**

8	**10**	**25**	**27**	**21**	**76**

Hasil akhir *Pass 2* :

8	**10**	**25**	**27**	**21**	**76**
1	2	3	4	5	6

Pass 3 : (Berdasarkan Hasil Akhir *Pass 2*)

8	**10**	**25**	**27**	**21**	**76**

8	**10**	**25**	**21**	**27**	**76**

8	**10**	**21**	**25**	**27**	**76**

Hasil Akhir *Pass 3* :

8	**10**	**21**	**25**	**27**	**76**
1	2	3	4	5	6

***Pass 4* :** (Berdasarkan Hasil Akhir *Pass 3*)

8	**10**	**21**	**25**	**27**	**76**

8	**10**	**21**	**25**	**27**	**76**

Hasil Akhir *Pass 4 :*

8	**10**	**21**	**25**	**27**	**76**
1	2	3	4	5	6

Pass 5 : (Berdasarkan Hasil Akhir *Pass 4*)

8	**10**	**21**	**25**	**27**	**76**

Hasil Akhir *Pass 5* :

8	**10**	**21**	**25**	**27**	**76**
1	2	3	4	5	6

Hasil akhir Pass 5 menyisakan satu elemen yang tidak perlu diurutkan lagi maka pengurutan selesai

Keterangan :

: Bagian data yang sudah diurutkan/diapungkan

: Bagian data yang dibandingkan dan mungkin ditukarkan posisinya

Procedure Bubble Sort

```
Procedure BubbleSort(input n:integer,input/output T:TabInteger)
{Mengurutkan Tabel Integer[1..N] dengan Bubble Sort}
Kamus
    pass : integer {pencacah untuk jumlah langkah}
    k    : integer {pencacah pengapungan untuk setiap langkah}
    temp : integer {variabel bantu untuk pertukaran}
Algoritma
    for pass ← 1 to (n-1) do
        for k ← n downto (pass+1) do
            if (T[k] < T[k-1]) then
                {pertukarkan T[k] dengan T[k-1]}
                temp ← T[k]
                T[k] ← T[k-1]
                T[k-1] ← temp
            endif
        endfor
    endfor
```

Implementasi Procedure Bubble Sort dalam Algoritma

```
procedure InputData(input n : integer; output T : TabInteger)
Kamus
        i : integer
Deskripsi
        for i ← 1 to n do
              read(T[i])
        endfor
Procedure BubbleSort(input n:integer,input/output T:TabInteger)
{Mengurutkan Tabel Integer[1..N] dengan Bubble Sort}
Kamus
      pass : integer {pencacah untuk jumlah langkah}
      k    : integer {pencacah pengapungan untuk setiap langkah}
      temp : integer {variabel bantu untuk pertukaran}
Algoritma
      for pass ← 1 to (n-1) do
            for k ← n downto (pass+1) do
                if (T[k] < T[k-1]) then
                      {pertukarkan T[k] dengan T[k-1]}
                      temp ← T[k]
                      T[k] ← T[k-1]
                      T[k-1] ← temp
                endif
            endfor
endfor
```

```
Algoritma Bubble_Sort
Kamus
    const  Nmax=100
    type TabInteger = array[1..NMax] of integer
    TabInt : TabInteger
    jml_data : integer

Deskripsi
    read (jml_data)        {memasukkan banyak data yang mau diinput}
    InputData(jml_data,TabInt)  {memanggil prosedur InputData}
    BubbleSort(jml_data,TabInt) {memanggil Prosedur BubbleSort}
```

Algoritma Bubble_Sort memanggil dua prosedur yaitu prosedur InputData(dengan parameter jml_data sebagai input untuk **parameter input n** dan TabInt sebagai output untuk **parameter output T**) dan prosedur BubbleSort(dengan jml_data sebagai input untuk **parameter input n** dan TabInt sebagai input sekaligus output untuk **parameter input/output T**)

2. Maximum Sort

Pengurutan model ini dilakukan dengan mencari **nilai terbesar/maksimum** dari suatu array. Nilai terbesar tersebut kemudian disimpan di awal array(jika diurutkan menurun) atau di akhir array(jika diurutkan menaik) dan diisolasi agar tidak disertakan lagi pada proses selanjutnya.

Ilustrasi Kasus :

Perhatikan array *TabInt* di bawah ini yang terdiri dari n = 6 elemen yang belum terurut. Array ini akan diurutkan **menaik** dengan metode **Bubble Sort.**

Elemen Array	**29**	**27**	**10**	**8**	**76**	**21**
Indeks	1	2	3	4	5	6

Pass 1 :

Cari elemen maksimum di dalam array *TabInt*[1..n] ➔ maks = TabInt[5] = 76
Pertukarkan TabInt[5] dengan TabInt[n]

Proses Pertukaran						**Hasil Akhir *Pass 1***					
29	**27**	**10**	**8**	**76**	**21**	**29**	**27**	**10**	**8**	**21**	**76**

Pass 2 : (Berdasarkan sususan Array hasil *Pass 1*)

Cari elemen maksimum di dalam array *TabInt*[1..5] ➔ maks = TabInt[1] = 29
Pertukarkan TabInt[1] dengan TabInt[5]

29	**27**	**10**	**8**	**21**	**76**	**21**	**27**	**10**	**8**	**29**	**76**

Pass 3 : (Berdasarkan sususan Array hasil *Pass 2*)

Cari elemen maksimum di dalam array *TabInt*[1..4] ➔ maks = TabInt[2] = 27
Pertukarkan TabInt[2] dengan TabInt[4]

21	**27**	**10**	**8**	**29**	**76**	**21**	**8**	**10**	**27**	**29**	**76**

Pass 4 : (Berdasarkan sususan Array hasil *Pass 3*)

Cari elemen maksimum di dalam array *TabInt*[1..3] ➔ maks = TabInt[1] = 21
Pertukarkan TabInt[1] dengan TabInt[3]

21	**8**	**10**	**27**	**29**	**76**	**10**	**8**	**21**	**27**	**29**	**76**

Pass 5 : (Berdasarkan sususan Array hasil *Pass 4*)

Cari elemen maksimum di dalam array *TabInt*[1..2] ➔ maks = TabInt[1] = 10
Pertukarkan TabInt[1] dengan TabInt[2]

10	**8**	**21**	**27**	**29**	**76**	**8**	**10**	**21**	**27**	**29**	**76**

Terisa satu elemen yaitu 8, maka pengurutan dihentikan. Array sudah terurut menaik

Keterangan :

: Bagian data yang sudah diurutkan

: Bagian data yang terbesar dan posisi yang akan ditukarkan dengan data terbesar

Algoritma 2Procedure Pengurutan dengan Metode Maximum Sort secara Menaik

```
Procedure MaxSort(input n:integer,input/output T:TabInteger)
{Mengurutkan Tabel Integer[1..N] dengan Maximum Sort}
Kamus
    i    : integer      {pencacah untuk jumlah langkah}
    k    : integer      {pencacah untuk mencari nilai maksimum}
    imaks: integer      {indeks yang berisi nilai maksimum sementara}
    temp : integer      {variabel bantu untuk pertukaran}
Algoritma
    for i ← n downto 2 do
        imaks ← 1
        for k ← 1 to (i-1) do
            if (T[k] > T[imaks]) then
                imaks ← k
            endif
        endfor
        {pertukarkan T[i] dengan T[imaks]}
        temp ← T[i]
        T[i] ← T[imaks]
        T[imaks] ← temp
    Endfor
```

3. Minimum Sort

Berbeda dengan algoritma pengurutan Maksimum, pada algoritma pengurutan minimum, basis pencarian adalah elemen minimum. Pengurutan model ini dilakukan dengan mencari **nilai terkecil/minimum** dari suatu array. Nilai terkecil tersebut kemudian disimpan di awal array(jika diurutkan menaik) atau di akhir array(jika diurutkan menurun) dan diisolasi agar tidak disertakan lagi pada proses selanjutnya.

Algoritma 3 Procedure Pengurutan dengan Metode Minimum Sort secara Menaik

```
Procedure MinSort(input n:integer,input/output T:TabInteger)
{Mengurutkan Tabel Integer[1..N] dengan Minimum Sort}
Kamus
    i    : integer      {pencacah untuk jumlah langkah}
    k    : integer      {pencacah untuk mencari nilai minimum}
    imin : integer      {indeks yang berisi nilai minimum sementara}
    temp : integer      {variabel bantu untuk pertukaran}
Algoritma
    for i ← 1 to (n-1) do
        imin ← i
        for k ← (i+1) to n do
            if (T[k] < T[imin]) then
                imin ← k
            endif
        endfor
        {pertukarkan T[i] dengan T[imin]}
        temp ← T[i]
        T[i] ← T[imin]
        T[imin] ← temp
    Endfor
```

LATIHAN SOAL

1. Untuk setiap metoda sort, coba analisis secara kualitatif apa yang dikerjakan jika :
   a. Elemen array sudah terurut
   b. Elemen array terurut terbailk
   c. Elemen array bernilai sama
2. Carilah performasi ke tiga algoritma pengurutan yang dibahas, dansebutkan kasus terbaik dan kasus terjeleknya.
3. Prosedur pengurutan maximum sort dan prosedur pengurutan minimum sort pada contoh yang dibahas, implementasikan kedalam bentuk algoritma seperti conoth prosedur bubble sort.
4. Perhatikanlah pada semua prosedur untuk sort yang sudah dibahas, dipakai kamus umum sebagai berikut.

```
Kamus Umum
    constant Nmax : integer = 100
    type TabInt : array [1..Nmax+1] of integer
    { jika diperlukan sebuah tabel, maka akan dibuat deklarasi sebagai berikut}
    T : TabInt {tabel integer}
    N : integer {indeks efektif, 1 ≤ N ≤ Nmax+1}
```

Jika prosedur diparametrisasi seperti pada spesifikasi yang diberikan, maka T dan N menjadi dua buah parameter. Padahal, nilai T dan N sebenarnya erat kaitannya satu sama lain. Deklarasi TabInt akan lebih baik jika dibungkus menjadi sebuah type komposisi sebagai berikut :

```
Kamus Umum
    constant Nmax : integer = 100
    type TabInt : < TI: array [1..Nmax+1] of integer
                    N : integer {indeks efektif, }
                    {maksimum tabel yang terdefinisi, 1 ≤ N ≤ Nmax+1}
    { jika diperlukan sebuah tabel, maka akan dibuat deklarasi sebagai berikut}
    T : TabInt {tabel integer}
```

Sebagai latihan, tuliskan ulang semua prosedur yang pernah didefinisikan dengan deklarasi type komposisi ini.

DAFTAR PUSTAKA

[1] Inggriani Liem. Diktat Kuliah Algoritma dan Pemrograman (Bagian Pemrograman Prosedural). 1999. Bandung : Diktat Kuliah Informatika ITB

[2] Kadir, Abdul. Dasar Pemrograman Delphi 5.0. 2000. Yogyakarta : ANDI

[3] Munir, Rinaldi. Algoritma dan Pemrograman 1. 2001. Bandung : Informatika

[4] Pardosi, Mico. Turbo Pascal 7.0. 1999. Surabaya : INDAH Surabaya

[5] Pranata, Antony. Algoritma dan Pemrograman. 2005. Yogyakarta : Graha Ilmu

[6] Wahid, Fathul. Dasar-dasar Algoritma dan Pemrograman. 2004. Yogyakarta : Andi Offset