MODUL V
Struktur Data

Judul	ALGORITMA PENGURUTAN	
Penyusun	**Distribusi**	**Perkuliahan**
Nixon Erzed	Teknik Informatika Universitas Esa Unggul	Pertemuan – V online

Tujuan :

Setelah mengikuti kuliah ini, mahasiswa dapat memahami berbagai jenis algoritma pengurutan dan mahasiswa dapat mengimplementasikannya algoritma pengurutan dalam kasus sederhana

Materi :

1. Pengertian Umum
2. Bubble sort
3. Exchange sort / Selection Sort
4. Insert sort

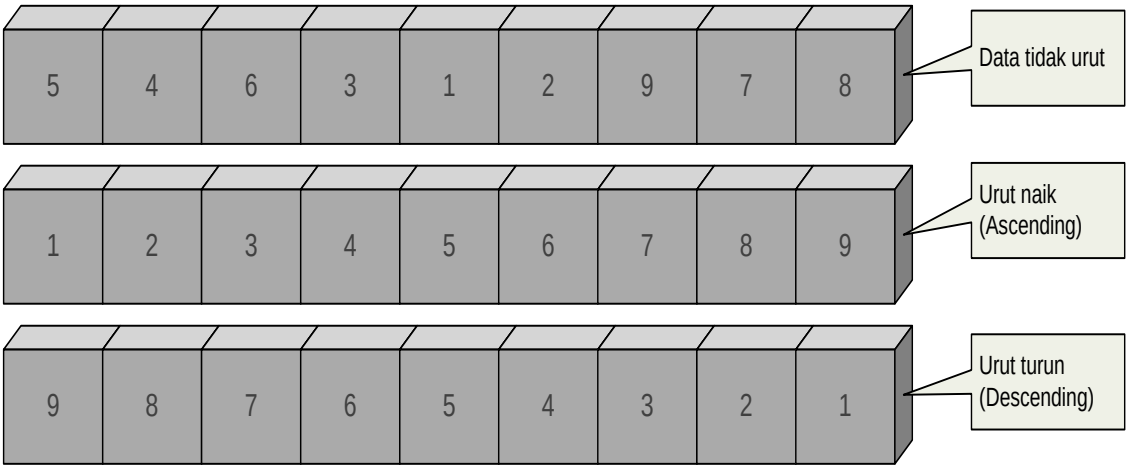
PENGURUTAN (SORTING)

Pengertian umum

Proses pengurutan banyak ditemukan dalam komputer. Hal itu karena data yang sudah urut akan lebih cepat untuk dicari. Untuk membentuk data yang tidak urut menjadi data yang urut, terdapat berbagi algoritma yang bisa digunakan. Beberapa algoritma akan dijelaskan pada bab ini.

Perlu diketahui bahwa pengurutan sendiri dapat dilakukan terhadap data yang secara keseluruhan diletakkan dalam memori ataupun terhadap data yang tersimpan pada pengingat eksternal. Pada bab ini pengurutan pada katagori pertama saja yang akan dibahas.

Di dalam pengurutan data terdapat istilah *ascending* dan *descending*. Pengurutan dengan dasar dari nilai yang kecil menuju ke nilai yang besar disebut *ascending* (urut naik), sedangkan yang disusun atas dasar dari nilai besar ke kecil disebut *descending* (urut turun).



Ada banyak cara pengurutan data, diantaranya yang popular:

- Metode Bubble Sort
- Metode Pengurutan Seleksi
- Exchange Sort
- Pengurutan dengan Penyisipan
- Pengurutan dengan Penyisipan Biner
- Metode Quick Sort

Metode Bubble Sort

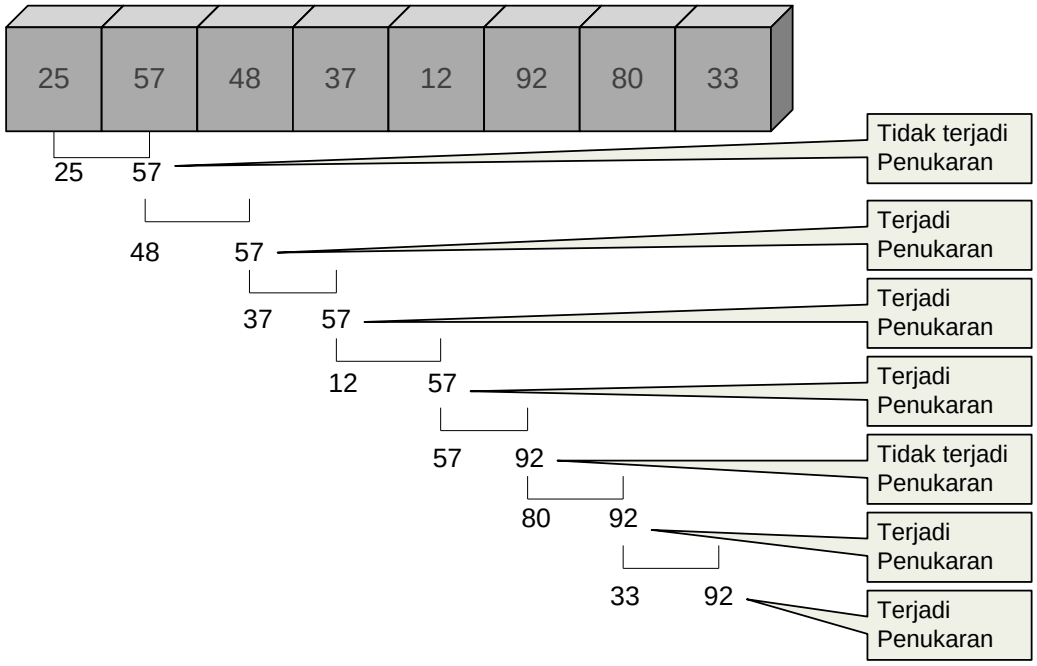
Metode *bubble sort*, merupakan metode tersederhana untuk melakukan pengurutan data, tetapi memiliki kinerja yang terburuk untuk data yang besar. Pengurutan dilakukan dengan membandingkan sebuah bilangan dengan seluruh bilangan yang terletak sesudah bilangan tersebut. Penukaran dilakukan kalau suatu kriteria dipenuhi.

Sebagai contoh, terdapat kumpulan seperti berikut.

25 57 48 37 12 92 80 33

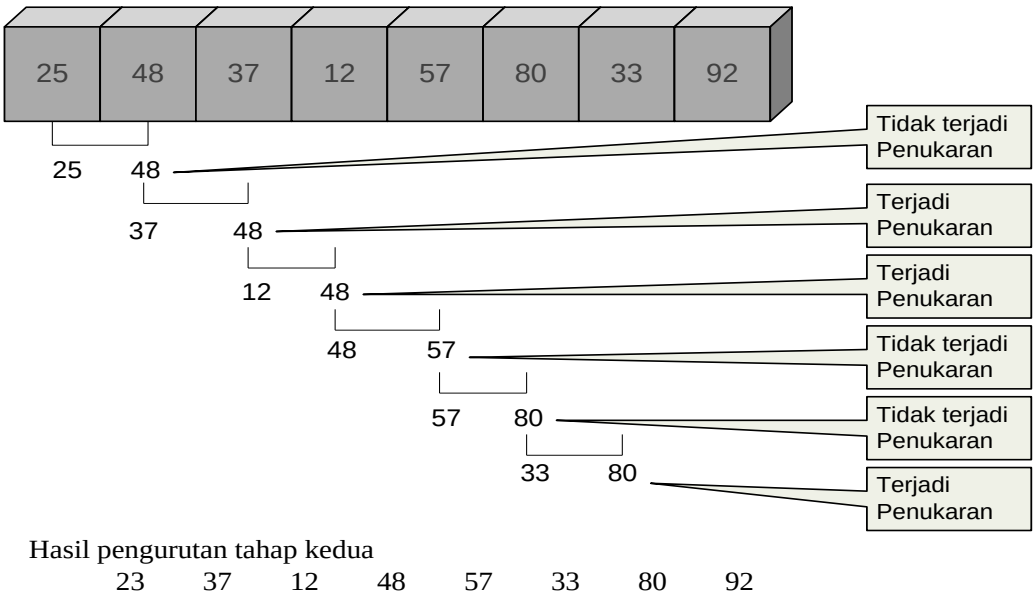
Contoh proses pengurutan dengan urut naik ditunjukkan pada gambar berikut :

Tahap 1 pengurutan :



Hasil pengurtan tahap 1 :

25 48 37 12 57 80 33 92



Jika jumlah data adalah *n*, maka terjadi *n-1* tahap pengurutan. Berarti pada contoh di di atas diperlukan 7 tahap pengurutan. Gambar berikut memperlihatkan setelah 7 tahap pengurutan dilakukan.

Awal	25	57	48	37	12	92	80	33
Tahap 1	25	48	37	12	57	80	33	92
Tahap 2	25	37	12	48	57	33	80	92
Tahap 3	25	12	37	48	33	57	80	92
Tahap 4	12	25	37	33	48	57	80	92
Tahap 5	12	25	33	37	48	57	80	92
Tahap 6	12	25	33	37	48	57	80	92
Tahap 7	12	25	33	37	48	57	80	92

Prinsip dasar Bubble Sort :

→ membanding data-data bertetangga, dan pertukarkan
- Min-max sort :
  jika data(i) > data(i+1) maka pertukarkan data
- Max-min sort :
  jika data(i) < data(i+1) pertukarkan data

→ dalam 1 putaran iterasi → data esktrim (terbesar atau terkecil) berada pada posisi terakhir Data[n]

Implementasi pengurutan dengan metode *bubble sort* baik dalam bentuk algoritma dan program.

Algoritma :

```
SUBRUTIN bubble_sort(L,n)
    Untuk tahap = 1 s/d n-1
        Untuk j ← 0 s/d n-tahap-1
            Jika L[j] > L[j+1] MAKA
                // Lakukan penukaran
                tmp ← L[j]
                L[j] ← L[j+1]
                L[j+1] ← tmp
                AKHIR – JIKA
            AKHIR – UNTUK
        AKHIR – UNTUK
    AKHIR – SUBRUTIN
```

Algoritma Bubble sort dalam Pseudocode Pascal Like

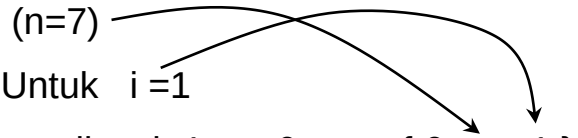
Algoritma Bubblesort;

```
Begin
  For   i = 1   to  n-1  do
      For j = 1  to  n-i  do
        Begin  IF  Data[ j ] > Data[j+1]
               Then  temp ⇐  Data[ j ]
                     Data[ j ] ⇐ Data[j+1]
                     Data[j+1] ⇐ temp
               Endif  .
       End-for .
  End-For
End-BubbleSort
```

Berikut ini adalah contoh penalaaran algoritma Bubble Sort diatas.

Misalkan diberikan data sbb:

28 30 51 8 19 5 22 → jumlah data n = 7



periksa j=1 to 6 { 6 = n -1 }

j=1 tdk ada pertukaran

j=2 tdk ada pertukaran

j=3	51 ←→ 8	28	30	8	51	19	5	22	
j=4	51 ←→ 19	28	30	8	19	51	5	22	
j=5	51 ←→ 5	28	30	8	19	5	51	22	
j=6	51 ←→ 22	28	30	8	19	5	22	**51**	

lajutkan untuk i=2

periksa j=1 to 5 {5 => 7- 2}

j=2	30 ←→ 8	28	8	30	19	5	22	**51**
j=3	30 ←→ 19	28	8	19	30	5	22	**51**
j=4	30 ←→ 5	28	8	19	5	30	22	**51**
j=5	30 ←→ 22	28	8	19	5	22	**30**	**51**

lanjutkan untuk i=3

periksa j=1 to 4

j=1	28 ←→ 8	8	28	19	5	22	**30**	**51**
j=2	28 ←→19	8	19	28	5	22	**30**	**51**
j=3	28 ←→5	8	19	5	28	22	**30**	**51**

lanjutkan hingga untuk i = 6

Pada contoh disajikan oleh gambar dihalaman 4, terlihat bahwa sebenarnya tidak diperlukan *n-1* tahap untuk mengurutkan data. Pada tahap kelima data sebenarnya sudah terurutkan. Oleh karena itu metode *bubble sort* dapat diperbaiki agar begitu data sudah urut dan walaupun n-1 tahap belum terpenuhi, pengurutan segera diakhiri. Hal ini dapat dilaksanakan dengan melakukan pemeriksaan apakah masih ada penukaran data atau tidak.
Kalau dalam suatu tahap ternyata tidak terjadi pertukaran data, maka iterasi segera dihentikan.

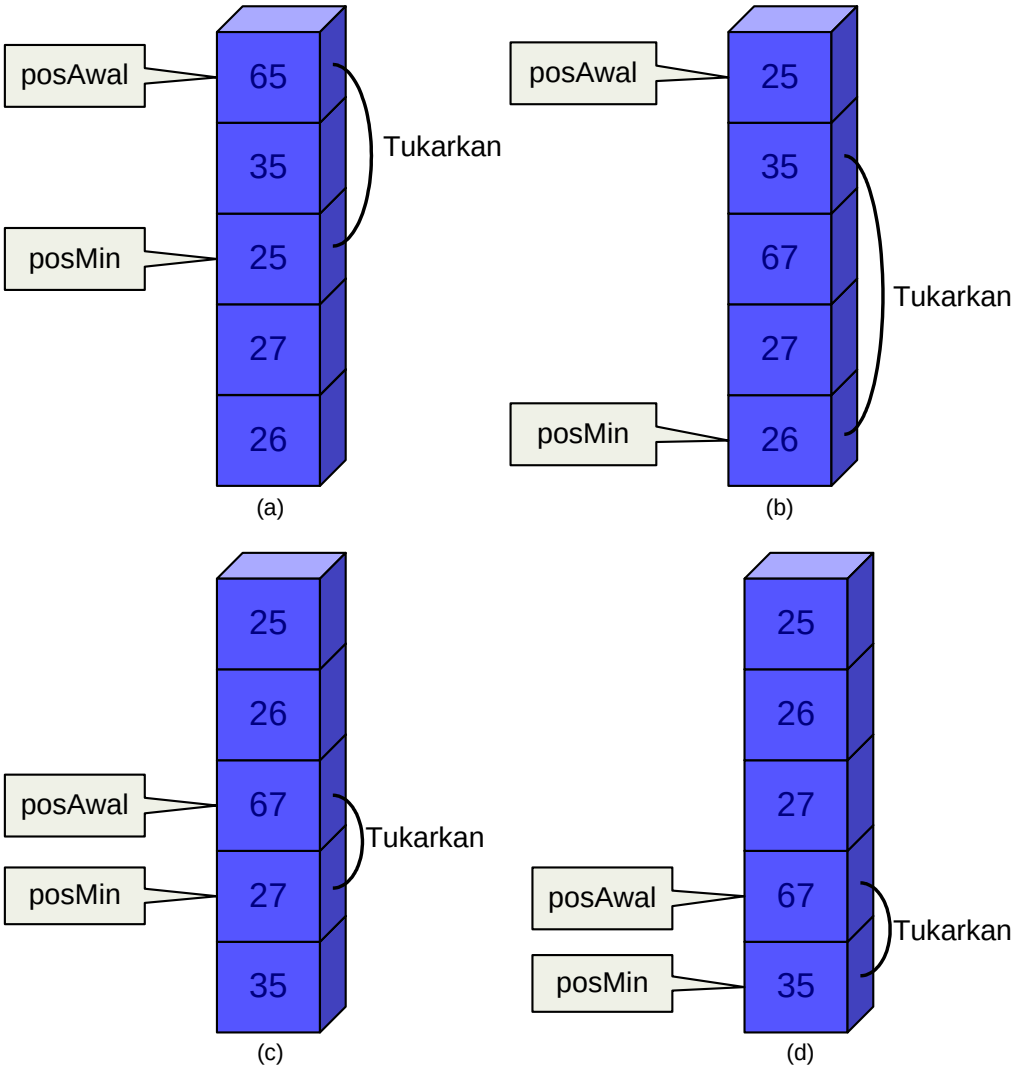
Cobalah untuk mengimplementasikannya.

Algoritma :

```
SUBRUTIN bubble_sort(L,n)
    Tahap ← 1
    Ada_penukaran ← BENAR
    ULANG SELAMA tahap ≤ n-1 DAN ada_penukaran
        Ada_penukaran ← SALAH
        UNTUK  j ← 0 S/D n-tahap-1
            JIKA L[j] > L[j+1] MAKA
                Ada penukaran ← BENAR
            // Lakukan penukaran
            tmp ← L[j]
            L[j] ← L[j+1]
            L[j+1] ← tmp
            AKHIR – JIKA
        AKHIR – UNTUK
        Tahap ← tahap + 1
    AKHIR – ULANG
AKHIR – SUBRUTIN
```

Metode Pengurutan Seleksi

Pengurutan seleksi (*selection sort*) mempunyai mekanisme seperti berikut :
Mula-mula suatu penunjuk (diberi nama posAwal), yang menunjuk ke lokasi awal pengurutan data, diatur agar berisi indeks pertama dalam larik. Selanjutnya dicari bilangan terkecil yang terletak antara posisi sesudah yang ditunjuk oleh petunjuk tersebut hingga elemen yang terakhir dalam larik. Lokasi bilangan ini ditunjuk oleh posMin. Lalu tukarkan nilai bilangan terkecil tersebut dengan nilai yang ditunjuk posAwal. Proses seperti itu diulang dari posAwal bernilai 0 hingga *n-2*, dengan *n* menyatakan jumlah elemen dalam larik.



Hasil Akhir : 25 26 27 67 35

Gambar Contoh Pengurutan Seleksi

Cara pengurutan seleksi tersebut secara lebih sederhana dilakukan dengan langkah-langkah sbb :

a. Mulai dari i = 1
b. Dari i s/d data terakhir : cari data ekstrim (terbesar atau terkecil) dan pertukarkan posisinya dengan data ke i = 1
c. Lanjutkan untuk i berikutnya untuk data ekstrim berikutnya

Untuk mengimplementasikan subrutin pengurutan seleksi baik dalam bentuk algoritma sebagai berikut.

Algoritma :

```
SUBRUTIN selection_sort (L,n)
    UNTUK posAwal = 0 S/D n-2
        PosMin ← posAwal
        UNTUK j ← posAwal + 1 S/D n-1
            JIKA L [posMin] > L[j] MAKA
                PosMin ← j
            AKHIR – JIKA
            AKHIR – UNTUK
        //Tukarkan
        tmp ← L[posAwal]
        L[posAwal] ← L[posMin]
    AKHIR – UNTUK
    AKHIR – SUBRUTIN
```

Dalam Pseudocode Pascal Like dan memanfaatkan repetitive For Next, algoritma Selection Sort adalah sebagai berikut:

```
ALGORITMA SELECTIONSORT
BEGIN
  For   i = 1   to  n-1  do
     Min ⇐  i
     For j = i+1  to  n  do
       Begin
          IF  Data[ min ] > Data[ j ]
           Then  min ⇐ j
        Endif  .
     End-for .
     If min <> i
       Then  temp ⇐  Data[ min ]
          Data[ min ] ⇐ Data[ i ]
          Data[ i ] ⇐ temp
      Endif  .
   End-For
END - Algoritma
```

Berikut ini adalah contoh penalaaran algoritma Selection Sort diatas.

Misalnya dimiliki data : 28 30 51 8 19 5 22

i=1 → min=1

periksa j=2 to 7

j=4 → min=4

j=6 → min =6

tukar data[i=1] dengan Data[min=6]

5 30 51 8 19 **28** 22

i=2

periksa dari j=3 to 7

min=4 2 ←→ 4

5 8 51 30 19 **28** 22

Dst untuk i=3

Contoh :
dimiliki data sbb : 32 15 71 18 25 9,
akan diurutkan dengan algoritma Selection Sort

	i = 1							i = 2					i = 3				
	J = 2 to 6							J = 3 to 6					J = 4 to 6				
	Awal	2	3	4	5	6		3	4	5	6		4	5	6		
1	32					32	9	9	9	9	9	9	9	9	9	9	
2	15	15	15	15	15		15	15	15	15	15	15	15	15	15	15	
3	71						71					71				18	
4	18						18					18	18	18	18	71	
5	25						25					25				25	
6	9					9	**32**					**32**				32	**dst**

Pembandingan Kinerja Algoritma Bubble Sort dan Selection Sort

1
2
 	
3
4
5
6
	For i = 1 to n-1 do
	For j = 1 to n-i do
	Begin
	IF Data[j] > Data[j+1]
	Then temp ⇐ Data[j]
	Data[j] ⇐ Data[j+1]
	Data[j+1] ⇐ temp
	Endif .
	End-for .
	End-For
		(n-1)
		½ .(n-1)n = ½ n^2 – ½ n
		
		½ n^2 – ½ n
		50% . (½ n^2 – ½ n)
		50% . (½ n^2 – ½ n)
		50% . (½ n^2 – ½ n)
		---------- +
		f(n) = 1¾ n^2 – ¾ n - 1
			Average Case
			f(n) = 1¾ n^2 – ¾ n - 1
			
			Best case
			f(n) = n^2 – 1
			
			worst case
			f(n) = 2½n^2 – 1½ n – 1
			
			contoh untuk average case
			
			2 data f(n) = 3,5
			1000data = 1,750jt-750-1
			≅ 1,750jt

1
2
3
 	
4
5
 	
6
7
8
9
	For i = 1 to n-1 do
	Min ⇐ i
	For j = i+1 to n do
	Begin
	IF Data[min] > Data[j]
	Then min ⇐ j
	Endif .
	End-for .
	If min <> i
	Then temp ⇐ Data[min]
	Data[min] ⇐ Data[i]
	Data[i] ⇐ temp
	Endif .
	End-For
		(n-1)
		(n-1)
		½ n^2 – ½ n
		
		½ n^2 – ½ n
		50%.(½ n^2 – ½ n)
		
		n-1
		50%(n-1)
		50%(n-1)
		50%(n-1)
		----------+
		f(n) = 1¼ n^2 + 3¼ n - 4½
			Average case :
			f(n) = 1¼ n^2 + 3¼ n - 4½
			
			Best Case
			f(n) = n^2 + 2n – 3
			
			Worst Case
			f(n) = 1½ n^2 + 4½ n – 6
			
			contoh unt Average case
			2 data f(n) = 7,5
			1000 f(n)=1,250jt+3250-4,5
			≅ 1,250jt

Pengurutan dengan Penyisipan

Pengurutan dengan penyisipan (*insertion sort*) adalah suatu metode yang melakukan pengurutan dengan cara menyisipkan data yang belum urut ke dalam bagian data yang telah diurutkan. Konsep ini biasa dilakukan pada permainan kartu. Ketika sebuah kartu baru didapatkan (hasil pembagian dari pengocokan kartu) kartu akan disisipkan oleh pemain pada posisi yang tepat sehingga penambahan kartu tersebut membuat semua kartu tetap terurutkan.

Gambar diatas adalah mengurutkan kartu dengan metode penyisipan kartu 7 disisipkan sehingga susunan kartu yang sebelumnya sudah urut tetap urut

Contoh

Bila *L* adalah larik dengan *n* elemen, mula-mula *L[0]* (elemen pertama) dianggap sebagai kumpulan data yang telah diurutkan, yang terdiri atas 1 buah data. Kemudian dilakukan penyisipan data dari *L[1]* sampai dengan *L[n-1]* ke dalam kumpulan data dari *L[0]* sampai dengan *L[k-1]* dengan $1 \leq k < n$. Dalam hal ini penyisipan dilakukan pada tempat yang tepat sehingga data pada *L[0]* sampai dengan *L[k]* menjadi urut.

Procedure Insertion Sort *{pada array yang sama)*

```
Begin
  For  i ⇐ 2 to n  do
    Begin
        j⇐ 1
       ketemu ⇐ false
       While j < i and not ketemu do
         begin
              if A[ i ] < A[ j ]
                then  ketemu ⇐ true
                else   j = j + 1
              end-if
        End-while
        IF ketemu
          then
             k = i
             temp ⇐  A[ i ]
             while k > j do
               begin
                     A[ k ] ⇐  A [k-1]
                     k ⇐ k – 1
              end-while
              A[ j ] ⇐ temp
         End-if
  End-for
```

Untuk penalaran algoritma, perhatian data contoh berikut :

	A [i]					
1	32	15	15	15	15	9
2	15	32	32	18	18	15
3	71	71	71	32	25	18
4	18	18	18	71	32	25
5	25	25	25	25	71	32
6	9	9	9	9	9	71

Ketemu → boolean

28 30 51 8 19 5 22

Versi-versi Insertion Sort

1. Versi1_Insertion Sort ***{dari Array A diinsert ke Array B}***

```
Algoritma
Begin
  B[1] ⇐ A[1]
  For  i ⇐ 2 to n  do

   Begin
      j⇐ 1
    ketemu ⇐ false
      while j < i and not ketemu do
       begin
          if A[ i ] < B[ j ]
            then  ketemu ⇐ true
            else   j = j + 1
          end-if
      end-while
       if not ketemu
        then  B[ j ] ⇐ A[ i ]
        else
            k = i
            while k > j do
              begin
                    B[ k ] ⇐  B [k-1]
                    k ⇐ k – 1
             end-while
             B[ j ] ⇐ A[ i ]
       End-if
   End-for
END – INSERTION
```

Versi2_Insertion Sort (x)

{dari Array A ke Array B yang sudah ada isinya dan Array A&B sudah terurut}

```
Algoritma
Begin
  For  i ⇐ 1 to n  do
    Begin
        j⇐ 1
     ketemu ⇐ false
        while j < x and not ketemu do
          begin
              if A[ i ] < B[ j ]
                then  ketemu ⇐ true
                else   j = j + 1
              end-if
        end-while
        if not ketemu
           then  B[ j ] ⇐ A[ i ]
           else
               k = x+1
               while k > j do
                 begin
                      B[ k ] ⇐  B [k-1]
                      k ⇐ k – 1
               end-while
               B[ j ] ⇐A[ i ]
         End-if
         x ⇐ x + 1
   End-for
END – INSERTION
```

Latihan Kasus Sort

1. Dimiliki sebuah tabel IPK Mahasiswa yang terdiri dari data : NIM, Nama, IPK. Data awal berupa data yang terurut berdasarkan NIM misalnya sbb:

NIM	Nama	IPK
V41055001	Ali	3,2
V41055002	Ani	2,4
V41055003	Ana	3,7
V41055004	Ale	2,2
Dst	Dst	
V410550xx	Ina	3,1

Jika tabel tsb direpresentasikan dengan sebuah array Mhs[i]

a. Deklarasikan struktur data yang sesuai
b. Definisikan sebuah prosedur singkat untuk mengisikan data
   ke array Mhs[i]
   sehingga data tersimpan apa adanya sesuai dengan tabel masukan
c. Definisikan sebuah prosedur untuk mengurutkan data berdasarkan IPK tertinggi ke terendah (max-min). Gunakan teknik Selection Sort
d. Ujilah prosedur yang didefinisikan dengan menggunakan data contoh diatas.

a. Struktur data :

```
Type    DataMhs  = record of
                NIM       : string [12]
                Nama      : string [20]
                IPK       : real

Var    DMahasiswa : aray [1..30] of DataMhs
```

Untuk pengolahan data tersebut, dibangun program utama, perhatikan algoritma berikut :

```
Program Utama

Begin
     n ⇐ 0
     IsiData (n)
     Selection_Sort (n)
End
```

b. Procedure IsiData (var jmlMhs : integer);
Var i : integer

```
Begin
    i ⇐ 1
    While masihadaData do
        Begin
            read( DMahasiswa[ i ].NIM)
            read( DMahasiswa [ i ].Nama)
            read( DMahasiswa [ i ].IPK)
            i ⇐ i + 1
        end –while

        jmlMhs ⇐ i - 1

end-procedure
```

Silahkan anda lanjutkan untuk soal c, dan d