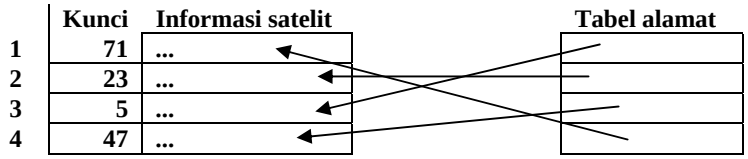
BAB V
SORTING (PENGURUTAN) INTERNAL

Sorting Internal : Proses pengurutan sekelompok data yang berada didalam memori utama komputer.

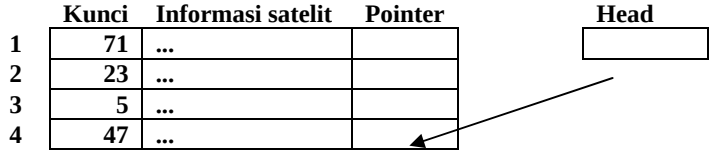
Sorting External : Proses pengurutan sekelompok data yang sebagian saja berada didalam memori yang ada (atau yang dialokasikan untuk proses) tidak dapat menampung semua data sekligus.

Sorting Tabel Alamat (Address Table Sorting) : Pengurutan tidak dilakukan secara fisik akan tetapi dengan cara membuat tabel alamat. Sebagai conoth hasil proses :



Untuk mempercepat prose, seringkali data kunci diletakkan juga pada tabel alamat, sedangkan informasi satelit tetap terpisah. Proses pengurutannya disebut Sorting Kunci (Key Sorting).

Sorting List (List Sorting) : Pengurutan tidak dilakuakn secra fisik, akan tetapi dengan mengubah isi pointer. Sebagai conoth hasil proses :



6.1 Bubble Sort (Pengurutan Gelembung)

Pada bubble sort setiap iterasi diperiksa dua data yang bersebelahan. Bila urutan tidak dipenuhi ke dua data tersebut saling bertukar tempat. Pada akhir tiap iterasi maka data terkecil yang ada pada sisa tabel telah bergeser ke bagian sebelah kiri /bagian atas dari tabel.

Untuk mendapatkan larik yang terurut *menaik*, algoritma pengurutan gelembung dapat ditulis sacara global sbb:

Untuk setiap langkah (*pass*) ke-I=1,2,…,N-1, lakukan :

Metode 1 (*untuk pengurutan ascending)* :

Mulai dari elemen J=N,N-1,…,I+1,lakukan :

1. Bandingkan L[J] dengan L[J-1].
2. Tukarkan L[J] dengan L[J-1] jika L[J] < L[J-1]

Rincian setip *pass* adalah sbb:

Pass 1 **:** Mulai dari elemen J=N,N-1,…,2, bandingkan L[J] dengan L[J-1]. Jika L[J] < L[J-1], tukarkan L[J] dengan L[J-1]. Pada akhir langkah 1, elemen L[1] berisi harga minimum pertama.

Pass 2 **:** Mulai dari elemen J=N,N-1,…,3, bandingkan L[J] dengan L[J-1]. Jika L[J] < L[J-1], tukarkan L[J] dengan L[J-1]. Pada akhir langkah 2, elemen L[2] berisi harga minimum kedua dan larik L[1..2] terurut, sedangkan L[3..N] belum terurut.

Pass 3 **:** Mulai dari elemen J=N,N-1,…,4, bandingkan L[J] dengan L[J-1]. Jika L[J] < L[J-1], tukarkan L[J] dengan L[J-1].]. Pada akhir langkah 3, elemen L[3] berisi harga minimum ketiga dan larik L[1..3] terurut, sedangkan L[4..N] belum terurut.

Pass N-1 **:** Mulai dari elemen K=N, bandingkan L[J] dengan L[J-1]. Jika L[J] < L[J-1], tukarkan L[J] dengan L[J-1].

Pada akhir langkah N-1, elemen L[N-1] berisi nilai minimum ke (N-1) danlarik L[1..N-1] terurut menaik(elemen yang tersisa adalah L[N] tidak perlu diurut karena hanya satu-satunya).

Contoh :

	44	**55**	**12**	**42**	**94**	**18**	**06**	**67**
Pass 1	**06**	**44**	**55**	**12**	**42**	**94**	**18**	**67**
	1	2	3	4	5	6	7	8
Pass 2	**06**	**12**	**44**	**55**	**18**	**42**	**94**	**67**
	1	2	3	4	5	6	7	8
Pass 3	**06**	**12**	**18**	**44**	**55**	**42**	**67**	**94**
	1	2	3	4	5	6	7	8
Pass 4	**06**	**12**	**18**	**42**	**44**	**55**	**67**	**94**
	1	2	3	4	5	6	7	8
Pass 5	**06**	**12**	**18**	**42**	**44**	**55**	**67**	**94**
	1	2	3	4	5	6	7	8
Pass 6	**06**	**12**	**18**	**42**	**44**	**55**	**67**	**94**
	1	2	3	4	5	6	7	8
Pass 7	**06**	**12**	**18**	**42**	**44**	**55**	**67**	**94**
	1	2	3	4	5	6	7	8

```
Procedure urutgelembung1_ascending(var L:larik;N:integer);
Var I      : integer; {pencacah untuk jumlah langkah}
    J      : integer; {pencacah untuk pengapungan pada setiap langkah}
   Temp    : integer; {peubah bantu pertukaran}
Begin
    For I := 1 to n-1 do
       For J := n downto I+1 do
           If L[J] < L[J-1] then
             {pertukaran L[J] dengan L[J-1];
              Temp   := L[J];
              L[J]   := L[J-1];
              L[J-1] := Temp;
End;
```

```
Procedure urutgelembung1_descending(var L:larik;N:integer);
Var I      : integer; {pencacah untuk jumlah langkah}
    J      : integer; {pencacah untuk pengapungan pada setiap langkah}
    Temp   : integer; {peubah bantu pertukaran}
Begin
    For I := 1 to n-1 do
        For J := n downto I+1 do
            If L[J] > L[J-1] then
              {pertukaran L[J] dengan L[J-1];
               Temp   := L[J];
               L[J]   := L[J-1];
               L[J-1] := Temp;
End;
```

Metode 2 (*untuk pengurutan ascending*) :

Mulai dari elemen J=1,1,…,N-1,lakukan :

1. Bandingkan L[J] dengan L[J+1].
2. Tukarkan L[J] dengan L[J+1] jika L[J] > L[J+1]

Contoh :

	44	**55**	**12**	**42**	**94**	**18**	**06**	**67**
Pass 1	**44**	**12**	**42**	**55**	**18**	**06**	**67**	**94**
	1	2	3	4	5	6	7	8
Pass 2	**12**	**42**	**44**	**18**	**06**	**55**	**67**	**94**
	1	2	3	4	5	6	7	8
Pass 3	**12**	**42**	**18**	**06**	**44**	**55**	**67**	**94**
	1	2	3	4	5	6	7	8
Pass 4	**12**	**18**	**06**	**42**	**44**	**55**	**67**	**94**
	1	2	3	4	5	6	7	8
Pass 5	**12**	**06**	**18**	**42**	**44**	**55**	**67**	**94**
	1	2	3	4	5	6	7	8
Pass 6	**06**	**12**	**18**	**42**	**44**	**55**	**67**	**94**
	1	2	3	4	5	6	7	8
Pass 7	**06**	**12**	**18**	**42**	**44**	**55**	**67**	**94**
	1	2	3	4	5	6	7	8

```
Procedure urutgelembung2_ascending(var L:larik;N:integer);
Var I      : integer; {pencacah untuk jumlah langkah}
    J      : integer; {pencacah untuk pengapungan pada setiap langkah}
    Temp   : integer; {peubah bantu pertukaran}
Begin
    For I := 1 to n-1 do
        For J := 1 to n-1 do
            If L[J] > L[J+1] then
              {pertukaran L[J] dengan L[J+1];
               Temp   := L[J];
               L[J]   := L[J+1];
               L[J-1] := Temp;
End;
```

```
Procedure urutgelembung2_desscending(var L:larik;N:integer);
Var I       : integer; {pencacah untuk jumlah langkah}
    J       : integer; {pencacah untuk pengapungan pada setiap langkah}
    Temp    : integer; {peubah bantu pertukaran}
Begin
    For I := 1 to n-1 do
        For J := 1 to n-1 do
            If L[J] < L[J+1] then
              {pertukaran L[J] dengan L[J+1];
               Temp   := L[J];
               L[J]   := L[J+1];
               L[J-1] := Temp;
End;
```

6.2 Selection Sort (Pengurutan Pilih)

Konsep dari metode ini adalah memilih elemen maksimum/iminimum dari larik, lalu menempatkan elemen maksimum/iminimum itu pad awal atau akhir larik (elemen terujung). Selanjutnya elemen terujung tersebut diisolasi dantidak disertakan pada proses selanjutnya. Proses yang sama di ulang untuk elemen larik yang tersisa yaitu elemen maksimum/iminimum berikutnya dan menukarkannya dengan elemen terujung larik sisa. Seperti pada pengurutan gelembung, proses memilih nilai maksimum/minimum dilakukan pada setiap pass. Jika larik berukuran N, maka jumlah pass adalah N-1.

Ada dua variasi algoritma pengurutan pilih ditinjau dari pemilihan elemen maksimum/iminimum, yaitu

1. Algoritma pengurutan maksimum, yaitu memilih elemen maksimum sebagai basis pengurutan.
2. Algoritma pengurutan iminimum, yaitu memilih elemen iminimum sebagai basis pengurutan.

6.2.1 Algoritma Pengurutan Iminimum

Untuk mendapatkan larik yang terurut **menaik**, algoritma pengurutan iminimum dapat ditulis secara global sebagai berikut :

Untuk setiap pass kei=1,2,…,N-1 lakukan :

1. cari elemen terkecil (imin) mulai dari elemen ke-I sampai elemen keN.
2. Tukarkan imin dengan elemen ke-I.

Rincian setiap pass adalah sebagai berikut :

Pass 1 **:** Cari elemen terkecil di dalam L[1..N].
Tukarkan elemen terkecil dengan elemen L[1].

Pass 2 **:** Cari elemen terkecil di dalam L[2..N].
Tukarkan elemen terkecil dengan elemen L[2].

Pass 3 **:** Cari elemen terkecil di dalam L[3..N].
Tukarkan elemen terkecil dengan elemen L[3].

Pass N-1 **:** Cari elemen terkecil di dalam L[N-1,N].
Tukarkan elemen terkecil dengan elemen L[N-1].
(elemen yang tersisa adalah L[N], tidak perlu di urut karena hanya satu-satunya)

Contoh :

44	55	12	42	94	18	06	67
1	2	3	4	5	6	7	8

Pass 1:

Cari elemen terkecil di dalam larik L[1..8] => Imin = 7, L[imin] =06. Tukar L[imin] dengan L[1], diperoleh :

06	55	12	42	94	18	44	67
1	2	3	4	5	6	7	8

Pass 2 :

(berdasarkan susunan larik hasil pass 1) Cari elemen terkecil di dalam larik [2..8] => Imin = 3, L[imin] = 12. Tukar L[Imin] dengan L[2], diperoleh :

06	12	55	42	94	18	44	67
1	2	3	4	5	6	7	8

Pass 3 :

(berdasarkan susunan larik hasil pass 2) Cari elemen terkecil di dalam larik [3..8] => Imin = 6, L[imin] = 18. Tukar L[Imin] dengan L[3], diperoleh :

06	12	18	42	94	55	44	67
1	2	3	4	5	6	7	8

Pass 4 :

(berdasarkan susunan larik hasil pass 3) Cari elemen terkecil di dalam larik [4..8] => Imin = 4, L[imin] = 42. Tukar L[Imin] dengan L[4] (sebenarnya tidak perlu dilakukan sebab 42 sudah berada pada posisi yang tepat), diperoleh :

06	12	18	42	94	55	44	67
1	2	3	4	5	6	7	8

Pass 5 :

(berdasarkan susunan larik hasil pass 4) Cari elemen terkecil di dalam larik [5..8] => Imin = 7, L[imin] = 44. Tukar L[Imin] dengan L[5], diperoleh :

06	12	18	42	44	55	94	67
1	2	3	4	5	6	7	8

Pass 6 :

(berdasarkan susunan larik hasil pass 5) Cari elemen terkecil di dalam larik [6..8] => Imin = 6, L[imin] = 55. Tukar L[Imin] dengan L[6] (sebenarnya tidak perlu dilakukan sebab 55 sudah berada pada posisi yang tepat), diperoleh :

06	12	18	42	44	55	94	67
1	2	3	4	5	6	7	8

Pass 7

(berdasarkan susunan larik hasil pass 6) Cari elemen terkecil di dalam larik [7..8] => Imin = 8, L[imin] = 67. Tukar L[Imin] dengan L[7], diperoleh :

06	12	18	42	44	55	67	94
1	2	3	4	5	6	7	8

Selesai. Larik L sudah terurut menaik !!!

```
Procedure urutpilihimin_ascending(var L:larik;N:integer);
Var i        : integer; {pencacah untuk jumlah langkah}
    J        : integer; {pencacah untuk pengapungan pada setiap langkah}
   Temp      : integer; {peubah bantu pertukaran}
    imin     : integer; {elemen iminimum}
Begin
    For i := 1 to n-1 do
      Begin
        imin:=i;
         For J := i+1 to N do
          If L[imin] > L[J] then
             imin     :=  J;
             Begin
               temp    := L[i];
               L[i]    := L[imin];
               L[imin] := temp;
             End;
     End;
End;
```

Untuk mendapatkan larik yang terurut **menurun**, algoritma pengurutan iminimum dapat ditulis sebagai berikut :

Rincian setiap pass adalah sebagai berikut :

Pass 1 **:** Cari elemen terkecil di dalam L[1..N].

Tukarkan elemen terkecil dengan elemen L[N].

Pass 2 **:** Cari elemen terkecil di dalam L[1..N-1].

Tukarkan elemen terkecil dengan elemen L[N-1].

Pass 3 **:** Cari elemen terkecil di dalam L[1..N-2].

Tukarkan elemen terkecil dengan elemen L[N-2].

Pass N-1 **:** Cari elemen terkecil di dalam L[1..2].

Tukarkan elemen terkecil dengan elemen L[2].

(elemen yang tersisa adalah L[1], tidak perlu di urut karena hanya satu-satunya)

Contoh :

44	**55**	**12**	**42**	**94**	**18**	**06**	**67**
1	2	3	4	5	6	7	8

Pass 1:

Cari elemen terkecil di dalam larik L[1..8] => Imin = 7, L[imin] =06. Tukar L[imin] dengan L[N] yaitu L[8], diperoleh :

44	**55**	**12**	**42**	**94**	**18**	**67**	**06**
1	2	3	4	5	6	7	8

Pass 2:

(berdasarkan susunan larik hasil pass 1) Cari elemen terkecil di dalam larik [1..7] => Imin = 3, L[imin] = 12. Tukar L[Imin] dengan L[7], diperoleh :

44	**55**	**67**	**42**	**94**	**18**	**12**	**06**
1	2	3	4	5	6	7	8

Pass 3:

(berdasarkan susunan larik hasil pass 2) Cari elemen terkecil di dalam larik [1..6] => Imin = 6, L[imin] = 18. Tukar L[Imin] dengan L[6] (sebenarnya tidak perlu dilakukan, karena 18 sudah berada pada posisi yang tepat), diperoleh :

44	**55**	**67**	**42**	**94**	**18**	**12**	**06**
1	2	3	4	5	6	7	8

Pass 4:
(berdasarkan susunan larik hasil pass 3) Cari elemen terkecil di dalam larik [1..5] => Imin = 4, L[imin] = 42. Tukar L[Imin] dengan L[5], diperoleh :

44	55	67	94	42	18	12	06
1	2	3	4	5	6	7	8

Pass 5:
(berdasarkan susunan larik hasil pass 4) Cari elemen terkecil di dalam larik [1..4] => Imin = 1, L[imin] = 44. Tukar L[Imin] dengan L[4], diperoleh :

94	55	67	44	42	18	12	06
1	2	3	4	5	6	7	8

Pass 6:
(berdasarkan susunan larik hasil pass 5) Cari elemen terkecil di dalam larik [1..3] => Imin = 2, L[imin] = 55. Tukar L[Imin] dengan L[3], diperoleh :

94	67	55	44	42	18	12	06
1	2	3	4	5	6	7	8

Pass 7:
(berdasarkan susunan larik hasil pass 6) Cari elemen terkecil di dalam larik [1..2] => Imin = 2, L[imin] = 67. Tukar L[Imin] dengan L[2] (sebenarnya tidak perlu dilakukan, karena 67 sudah berada pada posisi yang tepat), diperoleh :

94	67	55	44	42	18	12	06
1	2	3	4	5	6	7	8

Selesai. Larik L sudah terurut menaik !!!

```
Procedure urutpilihmin_descending(var L:larik;N:integer);
Var I       : integer; {pencacah untuk jumlah langkah}
    J       : integer; {pencacah untuk pengapungan pada setiap langkah}
    U       : integer;{indeks ujung kiri bagian larik yang telah terurut}
   Temp     : integer; {peubah bantu pertukaran}
    Imin    : integer; {elemen iminimum}
Begin
     U:=N;
     For I := 1 to n-1 do
     Begin
        Imin:=1;
         For J := I+1 to N do
          If L[imin] > L[J] then
             imin     :=  J;
             Begin
               Temp      := L[U];
              L[U]       := L[imin];
             L[imin]     := temp;
               U:=U-1;
             End;
     End;
End;
```

6.2.2 Algoritma Pengurutan Maksimum

Untuk mendapatkan larik yang terurut **menaik**, algoritma pengurutan maksimum dapat ditulis sebagai berikut :

Pass 1 **:** Cari elemen terbesar di dalam L[1..N].
Tukarkan elemen terbesar dengan elemen L[N].

Pass 2 **:** Cari elemen terbesar di dalam L[1..N-1].
Tukarkan elemen terbesar dengan elemen L[N-1].

Pass 3 **:** Cari elemen terbesar di dalam L[1..N-2].

Tukarkan elemen terbesar dengan elemen L[N-2].

Pass N-1 **:** Cari elemen terbesar di dalam L[1..2].

Tukarkan elemen terbesar dengan elemen L[2].

(elemen yang tersisa adalah L[1], tidak perlu di urut karena hanya satu-satunya)

Contoh :

44	**55**	**12**	**42**	**94**	**18**	**06**	**67**
1	2	3	4	5	6	7	8

Pass 1:

Cari elemen terbesar di dalam larik L[1..8] => Imaks = 5, L[imaks] =94. Tukar L[imaks] dengan L[N] yaitu L[8], diperoleh :

44	**55**	**12**	**42**	**67**	**18**	**06**	**94**
1	2	3	4	5	6	7	8

Pass 2:

(berdasarkan susunan larik hasil pass 1) Cari elemen terbesar di dalam larik [1..7] => Imaks = 5, L[imaks] = 67. Tukar L[Imaks] dengan L[7], diperoleh :

44	**55**	**12**	**42**	**06**	**18**	**67**	**94**
1	2	3	4	5	6	7	8

Pass 3:

(berdasarkan susunan larik hasil pass 2) Cari elemen terbesar di dalam larik [1..6] => Imaks = 2, L[imaks] = 55. Tukar L[Imaks] dengan L[6], diperoleh :

44	**18**	**12**	**42**	**06**	**55**	**67**	**94**
1	2	3	4	5	6	7	8

Pass 4:

(berdasarkan susunan larik hasil pass 3) Cari elemen terbesar di dalam larik [1..5] => Imaks = 1, L[imaks] = 44 . Tukar L[Imaks] dengan L[5], diperoleh :

06	**18**	**12**	**42**	**44**	**55**	**67**	**94**
1	2	3	4	5	6	7	8

Pass 5:

(berdasarkan susunan larik hasil pass 4) Cari elemen terbesar di dalam larik [1..4] => Imaks = 4, L[imaks] = 42. Tukar L[Imaks] dengan L[4] (sebenarnya tidak perlu dilakukan, karena 42 sudah berada pada posisi yang tepat), diperoleh :

06	**18**	**12**	**42**	**44**	**55**	**67**	**94**
1	2	3	4	5	6	7	8

Pass 6:

(berdasarkan susunan larik hasil pass 5) Cari elemen terbesar di dalam larik [1..3] => Imaks = 2, L[imaks] = 18 . Tukar L[Imaks] dengan L[3], diperoleh :

06	**12**	**18**	**42**	**44**	**55**	**67**	**94**
1	2	3	4	5	6	7	8

Pass 7:

(berdasarkan susunan larik hasil pass 6) Cari elemen terbesar di dalam larik [1..2] => Imaks = 2, L[imaks] = 12. Tukar L[Imaks] dengan L[2] (sebenarnya tidak perlu dilakukan, karena 12 sudah berada pada posisi yang tepat), diperoleh :

06	**12**	**18**	**42**	**44**	**55**	**67**	**94**
1	2	3	4	5	6	7	8

Selesai. Larik L sudah terurut menaik !!!

```
Procedure urutpilihmaks_ascending(var L:larik;N:integer);
Var I      : integer; {pencacah untuk jumlah langkah}
    J      : integer; {pencacah untuk pengapungan pada setiap langkah}
    U      : integer;{indeks ujung kiri bagian larik yang telah terurut}
  Temp     : integer; {peubah bantu pertukaran}
   imax    : integer; {elemen maksimum}
Begin
     U:=N;
     For I := 1 to n-1 do
     Begin
        iMax :=1;
         For J := i+1 to N do
          If L[J] > L[imax] then
             imax     :=  J;
             Begin
               Temp     := L[U];
              L[U]      := L[imax];
             L[imax]    := temp;
               U:=U-1;
             End;
     End;
End;
```

Untuk mendapatkan larik yang terurut **menurun**, algoritma pengurutan maksimum dapat ditulis sebagai berikut :

Pass 1 **:** Cari elemen terbesar di dalam L[1..N].

Tukarkan elemen terbesar dengan elemen L[1].

Pass 2 **:** Cari elemen terbesar di dalam L[2..N].

Tukarkan elemen terbesar dengan elemen L[2].

Pass 3 **:** Cari elemen terbesar di dalam L[3..N].

Tukarkan elemen terbesar dengan elemen L[3].

Pass N-1 **:** Cari elemen terbesar di dalam L[N-1..N]. Tukarkan elemen terbesar dengan elemen L[N-1]. (elemen yang tersisa adalah L[N], tidak perlu di urut karena hanya satu-satunya)

Contoh :

44	**55**	**12**	**42**	**94**	**18**	**06**	**67**
1	2	3	4	5	6	7	8

Pass 1:

Cari elemen terbesar di dalam larik L[1..8] => Imaks = 1, L[imaks] =94. Tukar L[imaks] dengan L[1] (sebenarnya tidak perlu dilakukan, karena 94 sudah berada pada posisi yang tepat), diperoleh :

94	**55**	**12**	**42**	**44**	**18**	**06**	**67**
1	2	3	4	5	6	7	8

Pass 2:

(berdasarkan susunan larik hasil pass 1) Cari elemen terbesar di dalam larik [2..8] => Imaks = 8, L[imaks] = 67. Tukar L[Imaks] dengan L[2], diperoleh :

94	**67**	**12**	**42**	**44**	**18**	**06**	**55**
1	2	3	4	5	6	7	8

Pass 3:

(berdasarkan susunan larik hasil pass 2) Cari elemen terbesar di dalam larik [3..8] => Imaks = 8, L[imaks] = 55. Tukar L[Imaks] dengan L[3], diperoleh :

94	**67**	**55**	**42**	**44**	**18**	**06**	**12**
1	2	3	4	5	6	7	8

Pass 4 :
(berdasarkan susunan larik hasil pass 3) Cari elemen terbesar di dalam larik [4..8] => Imaks = 5, L[imaks] = 44. Tukar L[Imaks] dengan L[4], diperoleh :

94	**67**	**55**	**44**	**42**	**18**	**06**	**12**
1	2	3	4	5	6	7	8

Pass 5:
(berdasarkan susunan larik hasil pass 4) Cari elemen terbesar di dalam larik L[5..8] =>Imaks= 5,L[imaks] = 42. Tukar L[imaks] dengan L[5] (sebenarnya tidak perlu dilakukan, karena 42 sudah berada pada posisi yang tepat), diperoleh :

94	**67**	**55**	**44**	**42**	**18**	**06**	**12**
1	2	3	4	5	6	7	8

Pass 6:
(berdasarkan susunan larik hasil pass 5) Cari elemen terbesar di dalam larik L[6..8] =>Imaks= 6,L[imaks] = 18. Tukar L[imaks] dengan L[6] (sebenarnya tidak perlu dilakukan, karena 18 sudah berada pada posisi yang tepat), diperoleh :

94	**67**	**55**	**44**	**42**	**18**	**06**	**12**
1	2	3	4	5	6	7	8

Pass 7:
(berdasarkan susunan larik hasil pass 6) Cari elemen terbesar di dalam larik [7..8] => Imaks = 8, L[imaks] = 12. Tukar L[Imaks] dengan L[7], diperoleh :

94	**67**	**55**	**44**	**42**	**18**	**12**	**06**
1	2	3	4	5	6	7	8

Selesai. Larik L sudah terurut menurun !!!

```
Procedure urutpilihmaks_descending(var L:larik;N:integer);
Var I       : integer; {pencacah untuk jumlah langkah}
    J       : integer; {pencacah untuk pengapungan pada setiap langkah}
    U       : integer;{indeks ujung kiri bagian larik yang telah terurut}
   Temp     : integer; {peubah bantu pertukaran}
    imax    : integer; {elemen maksimum}
Begin
      U:=N;
      For I := 1 to n-1 do
      Begin
         imax := i;
          For J := I+1 to N do
            If L[J] > L[imax] then
               imax     :=  J;
               Begin
                 Temp     := L[i];
                L[i]      := L[imax];
               L[imax]    := temp;
                 U:=U-1;
               End;
      End;
End;
```

6. 3 Insertion Sort (Pengurutan Sisip)

Untuk mendapatkan larik yang terurut **menaik** , algoritma pengurutan sisip dapat ditulis secara global sebagai berikut :

Untuk setiap pass ke- i = 2…N lakukan :

1. x ← L[i]
2. sisipkan x pada tempat yang sesuai antara L[1]…L[i]

Rincian setiap pass adalah sebagai berikut :

Andaian (pass 1) : L[1] dianggap sudah pada tempatnya

Pass 2 : x = L[2] harus dicari tempatnya yang tepat pada L[1..2] dengan cara menggeser elemen L[1..1] ke kanan (atau ke bawah, jika anda membayangkan larik terentang vertikal) bila L[1..1] lebih besar daripada L[2]. Misalkan posisi yang tepat adalah K. Sisipkan L[2] pada L[K].

Pass 3 : x = L[3] harus dicari tempatnya yang tepat pada L[1..3] dengan cara menggeser elemen L[1..2] ke kanan (atau ke bawah, jika anda membayangkan larik terentang vertikal) bila L[1..2] lebih besar daripada L[3]. Misalkan posisi yang tepat adalah K. Sisipkan L[3] pada L[K].

Pass N : x = L[N] harus dicari tempatnya yang tepat pada L[1..N] dengan cara menggeser elemen L[1..N-1] ke kanan (atau ke bawah, jika anda membayangkan larik terentang vertikal) bila L[1..N-1] lebih besar daripada L[N]. Misalkan posisi yang tepat adalah K. Sisipkan L[3] pada L[K].Hasil dari pass N : Larik L[1..N] sudah terurut, yaitu L[1] <= L[2} <=…<= L[N].

Contoh :

44	**55**	**12**	**42**	**94**	**18**	**06**	**67**
1	2	3	4	5	6	7	8

Andaian (**Pass 1**) :

Elemen x = L[1] dianggap sudah terurut.

44	**55**	**12**	**42**	**94**	**18**	**06**	**67**
1	2	3	4	5	6	7	8

Pass 2:

(berdasarkan susunan larik pada pass 1) Cari posisi yang tepat untuk x = L[2] pada L[1..2], diperoleh :

44	**55**	**12**	**42**	**94**	**18**	**06**	**67**
1	2	3	4	5	6	7	8

Pass 3:

(berdasarkan susunan larik pada pass 2) Cari posisi yang tepat untuk x = L[3] pada L[1..3], diperoleh :

12	**44**	**55**	**42**	**94**	**18**	**06**	**67**
1	2	3	4	5	6	7	8

Pass 4:

(berdasarkan susunan larik pada pass 3) Cari posisi yang tepat untuk x = L[4] pada L[1..4], diperoleh :

12	**42**	**44**	**55**	**94**	**18**	**06**	**67**
1	2	3	4	5	6	7	8

Pass 5:

(berdasarkan susunan larik pada pass 4) Cari posisi yang tepat untuk x = L[5] pada L[1..5], diperoleh :

12	**42**	**44**	**55**	**94**	**18**	**06**	**67**
1	2	3	4	5	6	7	8

Pass 6:

(berdasarkan susunan larik pada pass 5) Cari posisi yang tepat untuk x = L[6] pada L[1..6], diperoleh :

12	18	42	44	55	94	06	67
1	2	3	4	5	6	7	8

Pass 7:

(berdasarkan susunan larik pada pass 6) Cari posisi yang tepat untuk x = L[7] pada L[1..7], diperoleh :

12	18	06	42	44	55	94	67
1	2	3	4	5	6	7	8

Pass 8:

(berdasarkan susunan larik pada pass 8) Cari posisi yang tepat untuk x = L[8] pada L[1..8], diperoleh :

12	18	06	42	44	67	55	94
1	2	3	4	5	6	7	8

Selesai. Larik L sudah terurut menaik !!!

```
Procedure urutinsert_ascending(var L:larik;N:integer);
Var I          : integer; {pencacah untuk jumlah langkah}
    J          : integer; {pencacah untuk penelusuran larik}
    X          : integer; {peubah bantu agar L[K] tidak ditimpa selama
                 pergeseran}
    Boolean    : boolean; {peubah boolean untuk menyatakan posisi penyisipan
                 ditemukan}

Begin
     { elemen L[1] dianggap sudah terurut}
     For I := 2 to n do { mulai dari langkah 2 sampai langkah N}
     Begin
          x := L[i];
          {cari posisi yang tepat untuk x di dalam L[1..i-1] sambil menggeser}
          j := i -1;
          ketemu := false;
          while ( j >= 1) and (not ketemu) do
          Begin
               If  x < L[j] then
                 Begin
                      L[j+1] := L[j]; {geser}
                     J := j – 1;
                 End
                Else
                     Ketemu := true;
        End;
        L[j+1] := x;
      End;
End;
```