

# PENGURUTAN (SORTING)

11



## Overview

---

---

Seringkali perancang program perlu mengurutkan sekumpulan data yang dimiliki untuk memudahkan pemrosesan selanjutnya terhadap data tersebut. Pengurutan adalah sebuah algoritma dasar yang sering diperlukan dalam pembuatan program. Berbagai algoritma pengurutan telah diciptakan dan dapat digunakan. Pemahaman tentang beberapa algoritma pengurutan dasar perlu diketahui, termasuk cara penggunaannya dalam program.



## Tujuan

---

---

- Memahami konsep pengurutan
- Mengenal beberapa algoritma pengurutan
- Menerapkan algoritma pengurutan dalam program

## 11.1 Pengertian Sort

*Sorting* atau pengurutan data adalah proses yang sering harus dilakukan dalam pengolahan data. **Sort** dalam hal ini diartikan mengurutkan data yang berada dalam suatu tempat penyimpanan, dengan urutan tertentu baik urut menaik (*ascending*) dari nilai terkecil sampai dengan nilai terbesar, atau urut menurun (*descending*) dari nilai terbesar sampai dengan nilai terkecil. **Sorting** adalah proses pengurutan.

Terdapat dua macam pengurutan:

- ↳ **Pengurutan internal (*internal sort*)**, yaitu pengurutan terhadap sekumpulan data yang disimpan dalam media internal komputer yang dapat diakses setiap elemennya secara langsung. Dapat dikatakan sebagai pengurutan tabel
- ↳ **Pengurutan eksternal (*external sort*)**, yaitu pengurutan data yang disimpan dalam memori sekunder, biasanya data bervolume besar sehingga tidak mampu untuk dimuat semuanya dalam memori.

Dalam *courseware* ini, hanya akan dibahas algoritma pengurutan internal, dengan data berada dalam *array* satu dimensi.

Algoritma pengurutan internal yang utama antara lain:

1. Bubble Sort
2. Selection Sort
3. Insertion Sort
4. Shell Sort
5. Merge Sort
6. Radix Sort
7. Quick Sort
8. Heap Sort

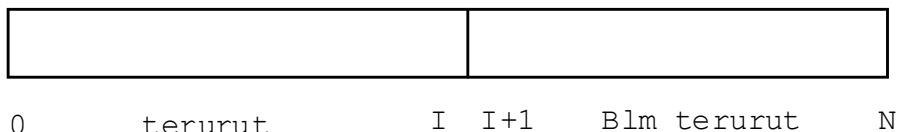
Dalam *courseware* ini hanya akan dibahas tiga metode *sort* yang pertama yang dianggap mudah, yaitu: **Bubble Sort , Selection Sort dan Insertion Sort**

### 11.1.1 Bubble Sort

*Bubble sort* adalah proses pengurutan sederhana yang bekerja dengan cara berulang kali membandingkan dua elemen data pada

suatu saat dan menukar elemen data yang urutannya salah. Ide dari *Bubble sort* adalah gelembung air yang akan "**mengapung**" untuk table yang terurut menaik (*ascending*). Elemen bernilai kecil akan "**diapungkan**" (ke indeks terkecil), artinya diangkat ke "**atas**" (indeks terkecil) melalui pertukaran. Karena algoritma ini melakukan pengurutan dengan cara membandingkan elemen-elemen data satu sama lain, maka *bubble sort* termasuk ke dalam jenis algoritma *comparison-based sorting*.

Proses dalam *Bubble sort* dilakukan sebanyak  $N-1$  langkah (pass) dengan  $N$  adalah ukuran *array*. Pada akhir setiap langkah ke –  $I$ , array  $L[0..N]$  akan terdiri atas dua bagian, yaitu bagian yang sudah terurut  $L[0..I]$  dan bagian yang belum terurut  $L[I+1..N-1]$ . Setelah langkah terakhir, diperoleh *array*  $L[0..N-1]$  yang terurut menaik.



Untuk mendapatkan urutan yang menaik, algoritmanya dapat ditulis secara global sebagai berikut :

Untuk setiap pass ke –  $I = 0, 1, \dots, N-2$ , lakukan :

Mulai dari elemen  $J = N-1, N-2, \dots, I + 1$ , lakukan :

- Bandingkan  $L[J-1]$  dengan  $L[J]$
- Pertukarkan  $L[J-1]$  dengan  $L[J]$  jika  $L[J-1] > L[J]$

Rincian setiap pass adalah sebagai berikut :

Pass 1:  $I = 0$ . Mulai dari elemen  $J = N-1, N-2, \dots, 1$ , bandingkan  $L[J-1]$  dengan  $L[J]$ . Jika  $L[J-1] > L[J]$ , pertukarkan  $L[J-1]$  dengan  $L[J]$ . Pada akhir langkah 1, elemen  $L[0]$  berisi harga minimum pertama.

Pass 2:  $I = 1$ . Mulai dari elemen  $J = N-1, N-2, \dots, 2$ , bandingkan  $L[J-1]$  dengan  $L[J]$ . Jika  $L[J-1] > L[J]$ , pertukarkan  $L[J-1]$  dengan  $L[J]$ . Pada akhir langkah 2, elemen  $L[1]$  berisi harga minimum kedua dan array  $L[0..1]$  terurut, sedangkan  $L[2..(N-1)]$  belum terurut.

Pass 3:  $I = 2$ . Mulai dari elemen  $J = N-1, N-2, \dots, 3$ , bandingkan  $L[J-1]$  dengan  $L[J]$ . Jika  $L[J-1] > L[J]$ , pertukarkan  $L[J-1]$  dengan  $L[J]$ . Pada akhir langkah 3, elemen  $L[2]$  berisi harga minimum ketiga dan array  $L[0..2]$  terurut, sedangkan  $L[3..(N-1)]$  belum terurut.

.....

Pass  $N-1$ : Mulai dari elemen  $J = N-1$ , bandingkan  $L[J-1]$  dengan  $L[J]$ . Jika  $L[J-1] > L[J]$ , pertukarkan  $L[J-1]$  dengan  $L[J]$ . Pada akhir langkah  $N-2$ , elemen  $L[N-2]$  berisi nilai minimum ke  $[N-2]$  dan array  $L[0..N-2]$  terurut menaik (elemen yang tersisa adalah  $L[N-1]$ , tidak perlu diurut karena hanya satu-satunya).

Misal array  $L$  dengan  $N = 5$  buah elemen yang belum terurut. Array akan diurutkan secara *ascending* (menaik).

	8	9	7	6	1
0 1 2 3 4					

### Pass 1 :

$I = 0 ; J = N-1 = 4$	8	9	7	1	6
$J = 3$	8	<b>9</b>	1	7	6
$J = 2$	<b>8</b>	1	9	7	6
$J = 1$	<b>1</b>	8	9	7	6

Hasil akhir langkah 1 :

	<b>1</b>	8	9	7	6
0 1 2 3 4					

### Pass 2 :

$I = 1 ; J = N-1 = 4$	<b>1</b>	8	<b>9</b>	<b>6</b>	7
$J = 3$	<b>1</b>	<b>8</b>	<b>6</b>	9	7
$J = 2$	<b>1</b>	<b>6</b>	8	9	7

Hasil akhir langkah 2 :

	<b>1</b>	<b>6</b>	8	9	7
0 1 2 3 4					

**Pass 3 :**

I = 2 ;J= N-1= 4      1      6      8      7      9  
 J = 3                    1      6      7      8      9

Hasil akhir langkah 3 :

<u>1</u>	<u>6</u>	<u>7</u>	8	9
0 1 2 3 4				

**Pass 4 :**

I = 3 ;J= N-1= 4      1      6      7      8      9

Hasil akhir langkah 4 :

<u>1</u>	<u>6</u>	<u>7</u>	<u>8</u>	<u>9</u>
0 1 2 3 4				

Selesai. Array L sudah terurut !!

Pseudocode prosedur algoritma *Bubble Sort* secara *Ascending*

- ```

1. //prosedur algoritma Bubble Sort secara Ascending
2. //I.S:array sudah berisi nilai integer yang belum terurut
3. //F.S:nilai-nilai dalam array terurut secara Ascending
4. procedure v_Bubble(input/output A:array[0..4]of integer,
input N:integer)
5. KAMUS:
6. i,j,temp:integer
7. ALGORITMA:
8. for(i=0;i<=(N-2);i++)
9. for(j=(N-1);j>=(i+1);j--)
10. if (A[j-1]>A[j])
11. temp←A[j-1]
12. A[j-1]←A[j]
13. A[j]←temp
14. endif
15. endfor
16. endfor
17.end procedure

```

## Program lengkap penerapan algoritma *Bubble Sort* dalam bahasa C

```
1. #include <stdio.h>
2. #include <conio.h>
3.
4. void v_Bubble(int A[],int N);
5. void main()
6. { int L[5];
7. int i,N;
8. //proses untuk memasukkan data array
9. printf("Banyak data : ");scanf("%i",&N);
10. for(i=0;i<N;i++)
11. { printf("Data ke-%i: ",i+1);
12. scanf("%i",&L[i]); } //end loop i
13. //memanggil procedure bubble sort
14. v_Bubble(L,N);
15.
16. //proses menampilkan kembali data array
17. printf("\nData Array Terurut\n");
18. for(i=0;i<N;i++)
19. { printf("%3i",L[i]); };
20. getch();
21. } //end main program
22.
23. void v_Bubble(int A[5],int N)
24. { int a,b,temp;
25. //proses sortir dengan bubble sort
26. for(a=0;a<=(N-2);a++)
27. { for(b=(N-1);b>=(a+1);b--)
28. { if (A[b-1] > A[b])
29. { temp = A[b-1];
30. A[b-1]= A[b];
31. A[b] = temp; } //endif
32. } //end loop j
33. } //end loop i
34. } //end procedure v_Bubble
```

Output yang dihasilkan:

```
D:\KULIAH\LATIHAN C\procBubble.exe
Banyak data : 5
Data ke-1: 8
Data ke-2: 9
Data ke-3: 7
Data ke-4: 6
Data ke-5: 1

Data Array Terurut
1 6 7 8 9
```

### 11.1.2 Selection Sort

Algoritma *Selection sort* memilih elemen maksimum/minimum *array*, lalu menempatkan elemen maksimum/minimum itu pada awal atau akhir *array* (tergantung pada urutannya *ascending/descending*). Selanjutnya elemen tersebut tidak disertakan pada proses selanjutnya. Karena setiap kali *selection sort* harus membandingkan elemen-elemen data, algoritma ini termasuk dalam *comparison-based sorting*.

Seperti pada algoritma *Bubble Sort*, proses memilih nilai maksimum /minimum dilakukan pada setiap pass. Jika *array* berukuran N, maka jumlah *pass* adalah N-1.

Terdapat dua pendekatan dalam metode pengurutan dengan *Selection Sort* :

- ↳ **Algoritma pengurutan maksimum** (*maximum selection sort*), yaitu memilih **elemen maksimum sebagai basis pengurutan**.
- ↳ **Algoritma pengurutan minimum** (*minimum selection sort*), yaitu memilih **elemen minimum sebagai basis pengurutan**.

#### 11.1.2.1 Maximum Selection Sort Ascending

Untuk mendapatkan *array* yang terurut menaik (*ascending*), algoritma *maximum selection sort* dapat ditulis sebagai berikut :

- ↳ Jumlah Pass = N-1 (*jumlah pass*)
- ↳ Untuk setiap pass ke – I = 0,1,....., jumlah pass lakukan :

- cari elemen maksimum (*maks*) mulai dari elemen ke – I sampai elemen ke – (N-1)
- pertukarkan *maks* dengan elemen ke – I
- kurangi N dengan satu

Rincian setiap *pass* adalah sebagai berikut :

**Langkah 1 :** Cari elemen maksimum di dalam  $L[0..(N-1)]$   
Pertukarkan elemen maksimum dengan elemen  $L[N-1]$

**Langkah 2 :** Cari elemen maksimum di dalam  $L[0..N-2]$   
Pertukarkan elemen maksimum dengan elemen  $L[N-2]$

**Langkah 3 :** Cari elemen maksimum di dalam  $L[0..N-3]$   
Pertukarkan elemen maksimum dengan elemen  $L[N-3]$

**Langkah N-1 :** Tentukan elemen maksimum di dalam  $L[0..1]$   
Pertukarkan elemen maksimum dengan elemen  $L[0]$

(elemen yang tersisa adalah  $L[0]$ , tidak perlu diurut karena hanya satu-satunya).

Jadi , pada setiap *pass* pengurutan terdapat proses mencari harga maksimum dan proses pertukaran dua buah elemen *array*.

Misal, terdapat *array* L dengan N = 5 buah elemen yang belum terurut. *Array* akan diurutkan secara **Ascending** (menaik), dengan algoritma ***maximum selection sort***.

|   |   |    |   |   |
|---|---|----|---|---|
| 9 | 7 | 12 | 6 | 1 |
| 0 | 1 | 2  | 3 | 4 |

**Pass 1 :**

- Cari elemen maksimum di dalam *array*  $L[0..4]$ . Maks= $L[2]=12$
- Tukar Maks dengan  $L[4]$ , diperoleh :

|   |   |   |   |    |
|---|---|---|---|----|
| 9 | 7 | 1 | 6 | 12 |
| 0 | 1 | 2 | 3 | 4  |

**Pass 2 :**

(berdasarkan susunan *array* pada Pass 1)

- Cari elemen maksimum di dalam array  $L[0..3]$ . Maks= $L[0]=9$
- Tukar Maks dengan  $L[3]$ , diperoleh :

|   |   |   |   |    |
|---|---|---|---|----|
| 6 | 7 | 1 | 9 | 12 |
| 0 | 1 | 2 | 3 | 4  |

### Pass 3:

(berdasarkan susunan array pada Pass 2)

- Cari elemen maksimum di dalam array  $L[0..2]$ . Maks= $L[1]=7$
- Tukar Maks dengan  $L[2]$ , diperoleh :

|   |   |   |   |    |
|---|---|---|---|----|
| 6 | 1 | 7 | 9 | 12 |
| 0 | 1 | 2 | 3 | 4  |

### Pass 4 :

(berdasarkan susunan array pada Pass 3)

- Cari elemen maksimum di dalam array  $L[0..1]$ . Maks= $L[0]=6$
- Tukar Maks dengan  $L[1]$ , diperoleh :

|   |   |   |   |    |
|---|---|---|---|----|
| 1 | 6 | 7 | 9 | 12 |
| 0 | 1 | 2 | 3 | 4  |

Selesai, array L sudah terurut secara Ascending.

Berikut ini akan diberikan *pseudocode procedure Maximum Selection Sort Ascending* dan *pseudocode procedure* untuk tukar tempat.

*Pseudocode Algoritma Maximum Selection Sort secara Ascending :*

1. //prosedur algoritma Maximum Selection Sort secara Ascending
2. //I.S:array sudah berisi nilai integer yang belum terurut
3. //F.S:nilai-nilai dalam array terurut secara Ascending
4. **procedure** v\_SelAsc(**input/output** A:array[0..4]of integer,  
**input** N:integer)
5. **KAMUS:**
6. maks,k,j,temp:integer
7. **ALGORITMA:**
8. **for**(k=(N-1);k>=0;k←k-1)

```

9. maks<-0;
10. // cari elemen maksimum
11. for(j=0;j<=k;j<-j+1)
12. if (A[j] > A[maks])
13. maks<-j;
14. endif
15. endfor
16. v_Tukar(A[k],A[maks]) //panggil procedure v_Tukar
17. endfor
18.end procedure

```

*Pseudocode Algoritma Tukar Tempat :*

```

1. //prosedur algoritma Tukar Tempat
2. //I.S:nilai-nilai yang dikirimkan sudah terdefinisi sebelumnya
3. //F.S:nilai yang dikirimkan tertukar nilainya
4. procedure v_Tukar(input/output P:integer,
input/output M:integer)
5. KAMUS:
6. temp:integer
7. ALGORITMA:
8. temp < P
9. P < M
10. M < temp
11.endprocedure

```

Program lengkap penerapan algoritma *Maximum Selection Sort Ascending* dalam bahasa C

```

#include <stdio.h>
#include <conio.h>
void v_SelAsc(int A[],int N);
void v_Tukar(int *P,int *M);

main()
{ int L[5];
int i,N;
//input data array
printf("Banyak Data: ");scanf("%i",&N);
for(i=0;i<N;i++)
{ printf("Data ke-%i: ",i+1);
scanf("%i",&L[i]); } //end loop i
//memanggil procedure v_SelAsc
v_SelAsc(L,N);

```

```

//menampilkan kembali data array
printf("\nData Terurut:\n");
for(i=0;i<N;i++)
{ printf("%3i",L[i]); } //end loop i
getche();
}

void v_SelAsc(int A[5],int N)
{ int maks,k,j,temp;
for(k=(N-1);k>=0;k--)
{ maks=0;
for(j=0;j<=k;j++)
{ if (A[j] > A[maks])
{ maks=j; } //endif
} //end loop j
v_Tukar(&A[k],&A[maks]);
} //end loop k
} //end procedure v_SelAsc

void v_Tukar(int *P,int *M)
{ int temp;
temp = *P;
*P = *M;
*M = temp;
} //end procedure v_Tukar

```

Output yang dihasilkan:

```

Banyak Data: 5
Data ke-1: 9
Data ke-2: 7
Data ke-3: 12
Data ke-4: 6
Data ke-5: 1

Data Terurut:
1 6 7 9 12_

```

### 11.1.2.2 Maximum Selection Sort Descending

Misal, terdapat *array L* dengan  $N = 5$  buah elemen yang belum terurut. *Array* akan diurutkan secara **Descending** (menurun), dengan algoritma ***maximum selection sort***.

|   |   |    |   |    |
|---|---|----|---|----|
| 9 | 8 | 11 | 7 | 12 |
| 0 | 1 | 2  | 3 | 4  |

### Pass 1 :

- Cari elemen maksimum di dalam array  $L[0..4]$ . Maks=L[4]=12
- Tukar Maks dengan L[0], diperoleh :

|    |   |    |   |   |
|----|---|----|---|---|
| 12 | 8 | 11 | 7 | 9 |
| 0  | 1 | 2  | 3 | 4 |

### Pass 2 :

(berdasarkan susunan array pada Pass 1)

- Cari elemen maksimum di dalam array  $L[1..4]$ . Maks=L[2]=11
- Tukar Maks dengan L[1], diperoleh :

|    |    |   |   |   |
|----|----|---|---|---|
| 12 | 11 | 8 | 7 | 9 |
| 0  | 1  | 2 | 3 | 4 |

### Pass 3 :

(berdasarkan susunan array pada Pass 2)

- Cari elemen maksimum di dalam array  $L[2..4]$ . Maks=L[4]=9
- Tukar Maks dengan L[2], diperoleh :

|    |    |   |   |   |
|----|----|---|---|---|
| 12 | 11 | 9 | 7 | 8 |
| 0  | 1  | 2 | 3 | 4 |

### Pass 4 :

(berdasarkan susunan array pada Pass 3)

- Cari elemen maksimum di dalam array  $L[3..4]$ . Maks=L[4]=8
- Tukar Maks dengan L[3], diperoleh :

|    |    |   |   |   |
|----|----|---|---|---|
| 12 | 11 | 9 | 8 | 7 |
| 0  | 1  | 2 | 3 | 4 |

Selesai array L sudah terurut secara Descending (menurun)

Pseudocode Algoritma *Maximum Selection Sort* secara *Descending* :

```

1. //prosedur algoritma Maximum Selection Sort secara Descending
2. //I.S:array sudah berisi nilai integer yang belum terurut
3. //F.S:nilai-nilai dalam array terurut secara Descending
4. procedure v_SelDesc(input/output A:array[0..4]of integer,
input N:integer)
5. KAMUS:
6. k,maks,j,temp:integer
7. ALGORITMA:
8. for(k=0;k<=(N-2);k←k+1)
9. //cari elemen maksimum
10. maks←k
11. for(j=(k+1);j<=(N-1);j←j+1)
12. if (A[j] > A[maks])
13. maks←j
14. endif
15. endfor
16. temp←A[k]
17. A[k]←A[maks]
18. A[maks]←temp
19. endfor

```

Program lengkap penerapan algoritma *Maximum Selection Sort Descending* dalam bahasa C

```

1. #include <stdio.h>
2. #include <conio.h>
3. void v_Tukar(int *P,int *M);
4. void v_SelDesc(int A[5],int N);
5. main()
6. { int L[5];
7. int i,k,j,maks,temp,N;
8. printf("Banyak Data: ");scanf("%i",&N);
9. //input data array
10. printf("Input Data Array\n");
11. for(i=0;i<N;i++)
12. { printf("Data ke-%i = ",i+1);
13. scanf("%i",&L[i]); } //endloop i
14. //panggil procedure v_SelDesc
15. v_SelDesc(L,N);
16. printf("\nOutput Data Array Terurut:\n");
17. for(i=0;i<N;i++)
18. { printf(" %5i",L[i]); } //endloop i
19.
20. printf("\nTekan Enter...\n");
21. getch();

```

```

22. } //end main program
23.
24. void v_SelDesc(int A[5],int N)
25. { int k,maks,j,temp;
26. //proses sorting max descending
27. for(k=0;k<=(N-2);k++)
28. { //cari elemen maksimum
29. maks=k;
30. for(j=(k+1);j<=(N-1);j++)
31. { if (A[j] > A[maks])
32. maks=j; } //endfor loop j
33. v_Tukar(&A[k],&A[maks]);
34. } //endfor loop k
35. } //end procedure v_SelDesc
36.
37. void v_Tukar(int *P,int *M)
38. { int temp;
39. temp = *P;
40. *P = *M;
41. *M = temp;
42. } //end procedure v_Tukar

```

Output yang dihasilkan:

```

D:\KULIAH\LATIHAN C\procMaxDesc.exe
Banyak Data: 5
Input Data Array
Data ke-1 = 9
Data ke-2 = 8
Data ke-3 = 11
Data ke-4 = 7
Data ke-5 = 12

Output Data Array Terurut:
12 11 9 8 7
Tekan Enter...

```

### 11.1.2.3 Minimum Selection Sort Ascending

Untuk mendapatkan *array* yang terurut menaik (*ascending*), algoritma minimum *selection sort* dapat ditulis sebagai berikut :

1. Jumlah Pass =  $N-1$  (*jumlah pass*)
2. Untuk setiap pass ke –  $I = 0,1,\dots,N-1$ , lakukan :
  - a. cari elemen minimum (*min*) mulai dari elemen ke –  $I$  sampai elemen ke –  $(N-1)$
  - b. pertukarkan *min* dengan elemen ke –  $I$

Rincian setiap *pass* adalah sebagai berikut :

**Langkah 1 :** Cari elemen minimum di dalam  $L[0..(N-1)]$   
Pertukarkan elemen terkecil dengan elemen  $L[0]$

**Langkah 2 :** Cari elemen minimum di dalam  $L[1..(N-1)]$   
Pertukarkan elemen terkecil dengan elemen  $L[1]$

**Langkah 3 :** Cari elemen minimum di dalam  $L[2..(N-1)]$   
Pertukarkan elemen terkecil dengan elemen  $L[2]$

.....  
**Langkah N-1:** Tentukan elemen minimum di dalam  $L[(N-2)..(N-1)]$   
Pertukarkan elemen terkecil dengan elemen  $L[N-2]$

(elemen yang tersisa adalah  $L[N-1]$ , tidak perlu diurut karena hanya satu-satunya).

Jadi, pada setiap *pass* pengurutan terdapat proses mencari harga minimum dan proses pertukaran dua buah elemen *array*.

Misal, terdapat *array* L dengan  $N = 5$  buah elemen yang belum terurut. *Array* akan diurutkan secara Ascending (menaik), dengan **algoritma minimum selection sort**.

|   |   |    |   |   |
|---|---|----|---|---|
| 9 | 7 | 12 | 6 | 1 |
|---|---|----|---|---|

0            1            2            3            4

**Pass 1 :**

- Cari elemen terkecil di dalam *array*  $L[0..4]$ . Min= $L[4]=1$
- Tukar Min dengan  $L[0]$ , diperoleh :

|   |   |    |   |   |
|---|---|----|---|---|
| 1 | 7 | 12 | 6 | 9 |
|---|---|----|---|---|

0            1            2            3            4

**Pass 2 :**

(berdasarkan susunan *array* pada Pass 1)

- Cari elemen terkecil di dalam *array*  $L[1..4]$ . Min= $L[3]=6$
- Tukar Min dengan  $L[1]$ , diperoleh :

|   |   |    |   |   |
|---|---|----|---|---|
| 1 | 6 | 12 | 7 | 9 |
|---|---|----|---|---|

0            1            2            3            4

### Pass 3:

(berdasarkan susunan array pada Pass 2)

- Cari elemen terkecil di dalam array L[2..4]. Min=L[3]=7
- Tukar Min dengan L[2], diperoleh :

|   |   |   |    |   |
|---|---|---|----|---|
| 1 | 6 | 7 | 12 | 9 |
|---|---|---|----|---|

0            1            2            3            4

### Pass 4 :

(berdasarkan susunan array pada Pass 3)

- Cari elemen terkecil di dalam array L[3..4]. Min=L[4]=9
- Tukar Min dengan L[3], diperoleh :

|   |   |   |   |    |
|---|---|---|---|----|
| 1 | 6 | 7 | 9 | 12 |
|---|---|---|---|----|

0            1            2            3            4

Selesai, array L sudah terurut secara Ascending.

Pseudocode Algoritma *Minimum Selection Sort* secara *Ascending* :

```

1. //prosedur algoritma Minimum Selection Sort secara Ascending
2. //I.S:array sudah berisi nilai integer yang belum terurut
3. //F.S:nilai-nilai dalam array terurut secara Ascending
4. procedure v_minAsc(input/output A:array[0..4]of integer,
input N:integer)
5. KAMUS:
6. k,min,j,temp:integer
7. ALGORITMA:
8. for(k=0;k<=(N-2);k←k+1)
9. //cari elemen terkecil
10. min ← k
11. for(j=(k+1);j<=(N-1);j←j+1)
12. if (A[j] < A[min])
13. min ← j
14. endif

```

```
15. endfor  
16. v_Tukar(A[k],A[min])  
17.endfor
```

Program lengkap penerapan algoritma *Minimum Selection Sort Ascending* dalam bahasa C

```
1. #include <stdio.h>  
2. #include <conio.h>  
3. void v_minAsc(int A[5],int N);  
4. void v_Tukar(int *P,int *M);  
5. main()  
6. { int L[5];  
7. int i,j,k,min,temp,N;  
8. //input data array  
9. printf("Input Data Array\n");  
10. printf("\nBanyak Data : "); scanf("%i",&N);  
11. for(i=0;i<N;i++)  
12. { printf(" Data ke-%i = ",i+1);  
13. scanf("%i",&L[i]); } //end loop i  
14. //panggil procedure v_minAsc  
15. v_minAsc(L,N);  
16. //output data array  
17. printf("\n Data Sortir:\n");  
18. for(i=0;i<N;i++)  
19. { printf(" %5i",L[i]); } //end loop i  
20. printf("\n Tekan Enter\n");  
21. getch();  
22. } //end main program  
23.  
24. void v_minAsc(int A[5],int N)  
25. { int k,min,j,temp;  
26. //proses minimum ascending selection sort  
27. for(k=0;k<=(N-2);k++)  
28. { min = k;  
29. for(j=(k+1);j<=(N-1);j++)  
30. { if (A[j] < A[min])  
31. min = j; } //endloop j  
32. v_Tukar(&A[k],&A[min]); } //end loop k  
33. } //end procedure  
34.  
35. void v_Tukar(int *P,int *M)  
36. { int temp;  
37. temp = *P;
```

```

38. *P = *M;
39. *M = temp;
40. } //end procedure v_Tukar

```

Output yang dihasilkan:

```

D:\KULIAH\LATIHAN C\procMinAsc.exe
Input Data Array

Banyak Data : 5
Data ke-1 = 9
Data ke-2 = 7
Data ke-3 = 12
Data ke-4 = 6
Data ke-5 = 1

Data Sortir:
1      6      7      9      12
Tekan Enter

```

#### 11.1.2.4 Minimum Selection Sort Descending

Misal, terdapat array L dengan  $N = 5$  buah elemen yang belum terurut. Array akan diurutkan secara **Descending** (menurun), dengan algoritma ***minimum selection sort***.

|   |   |    |   |    |
|---|---|----|---|----|
| 9 | 8 | 11 | 7 | 12 |
| 0 | 1 | 2  | 3 | 4  |

##### Pass 1 :

- Cari elemen terkecil di dalam array  $L[0..4]$ . Min= $L[3]=7$
- Tukar Min dengan  $L[4]$ , diperoleh :

|   |   |    |    |   |
|---|---|----|----|---|
| 9 | 8 | 11 | 12 | 7 |
| 0 | 1 | 2  | 3  | 4 |

##### Pass 2 :

(berdasarkan susunan array pada Pass 1)

- Cari elemen terkecil di dalam array  $L[0..3]$ . Min= $L[1]=8$
- Tukar Min dengan  $L[3]$ , diperoleh :

|   |    |    |   |   |
|---|----|----|---|---|
| 9 | 12 | 11 | 8 | 7 |
| 0 | 1  | 2  | 3 | 4 |

### Pass 3 :

(berdasarkan susunan array pada Pass 2)

- Cari elemen terkecil di dalam array L[0..2]. Min=L[0]=9
- Tukar Min dengan L[2], diperoleh :

|    |    |   |   |   |
|----|----|---|---|---|
| 11 | 12 | 9 | 8 | 7 |
| 0  | 1  | 2 | 3 | 4 |

### Pass 4 :

(berdasarkan susunan array pada Pass 3)

- Cari elemen terkecil di dalam array L[0..1]. Min=L[0]=11
- Tukar Min dengan L[1], diperoleh :

|    |    |   |   |   |
|----|----|---|---|---|
| 12 | 11 | 9 | 8 | 7 |
| 0  | 1  | 2 | 3 | 4 |

Selesai array L sudah terurut secara *Descending* (menurun)

Pseudocode Algoritma Minimum Selection Sort secara Descending :

```
1. //prosedur algoritma Minimum Selection Sort secara Descending
2. //I.S:array sudah berisi nilai integer yang belum terurut
3. //F.S:nilai-nilai dalam array terurut secara Descending
4. procedure v_minDesc(input/output A:array[0..4]of integer,
input N:integer)
5. KAMUS:
6. k,j,temp,min : integer
7. ALGORITMA:
8. //minimum selection sort descending
9. for(k=(N-1);k>=1;k←k-1)
10. min←0
11. //cari nilai terkecil
12. for(j=0;j<=k;j←j+1)
13. if (A[j] < A[min])
14. min←j
15. endif
16. endfor
17. v_Tukar(A[k],A[min])
20. endfor
```

Program lengkap penerapan algoritma *Minimum Selection Sort Descending* dalam bahasa C

```
1. #include <stdio.h>
2. #include <conio.h>
3. void v_minDesc(int A[5],int N);
4. void v_Tukar(int *P,int *M);
5. main()
6. { int L[5];
7. int i,N;
8. //input data array
9. printf("Input Data Array\n");
10. printf("\nBanyak Data : ");scanf("%i",&N);
11. for(i=0;i<N;i++)
12. { printf(" Data ke-%i = ",i+1);
13. scanf("%i",&L[i]); } //endloop i
14. //panggil procedure v_minDesc
15. v_minDesc(L,N);
16. //output data array
17. printf("\n Data Sortir:\n");
18. for(i=0;i<N;i++)
19. { printf(" %5i",L[i]); } //endloop i
20. printf("\n Tekan Enter...\n");
21. getche();
22. } //end main program
23.
24. void v_minDesc(int A[5],int N)
25. { int k,j,temp,min;
26. //minimum selection sort descending
27. for(k=(N-1);k>=1;k--)
28. { min = 0;
29. for(j=0;j<=k;j++)
30. { if (A[j] < A[min])
31. min=j; } //endloop j
32. v_Tukar(&A[k],&A[min]); } //endloop k
33. } //end procedure v_minDesc
34.
35. void v_Tukar(int *P,int *M)
36. { int temp;
37. temp = *P;
38. *P = *M;
39. *M = temp;
40. } //end procedure v_Tukar
```

Output yang dihasilkan:

```
D:\KULIAH\LATIHAN C\procMinDesc.exe
Input Data Array
Banyak Data : 5
Data ke-1 = 9
Data ke-2 = 8
Data ke-3 = 11
Data ke-4 = 7
Data ke-5 = 12

Data Sortir:
12    11    9    8    7
Tekan Enter...
```

### 11.1.3 Insertion Sort

*Insertion sort* adalah sebuah algoritma pengurutan yang membandingkan dua elemen data pertama, mengurutkannya, kemudian mengecek elemen data berikutnya satu persatu dan membandingkannya dengan elemen data yang telah diurutkan. Karena algoritma ini bekerja dengan membandingkan elemen-elemen data yang akan diurutkan, algoritma ini termasuk pula dalam *comparison-based sort*.

Ide dasar dari algoritma *Insertion Sort* ini adalah mencari tempat yang "tepat" untuk setiap elemen array, dengan cara *sequential search*. Proses ini kemudian menyisipkan sebuah elemen array yang diproses ke tempatnya yang seharusnya. Proses dilakukan sebanyak  $N-1$  tahapan (dalam *sorting* disebut sebagai "*pass*"), dengan indeks dimulai dari 0.

Proses pengurutan dengan menggunakan algoritma *Insertion Sort* dilakukan dengan cara membandingkan data ke- $i$  (dimana  $i$  dimulai dari data ke-2 sampai dengan data terakhir) dengan data berikutnya. Jika ditemukan data yang lebih kecil maka data tersebut disisipkan ke depan sesuai dengan posisi yang seharusnya.

Misal terdapat *array* satu dimensi  $L$ , yang terdiri dari 7 elemen *array* ( $n=7$ ). *Array*  $L$  sudah berisi data seperti dibawah ini dan akan diurutkan secara *ascending* dengan algoritma *Insertion Sort*.

|     |    |    |   |    |    |   |    |
|-----|----|----|---|----|----|---|----|
| L[] | 15 | 10 | 7 | 22 | 17 | 5 | 12 |
|     | 0  | 1  | 2 | 3  | 4  | 5 | 6  |

Tahapan *Insertion Sort*:

- ↳ Dimulai dari L[1]: Simpan nilai L[1] ke variabel X.  
**(Pass-1)** Geser masing-masing satu langkah ke kanan semua nilai yang ada disebelah kiri L[1] satu persatu apabila nilai tersebut lebih besar dari X.  
Setelah itu *insert*-kan (sisipkan) X di bekas tempat nilai yang terakhir digeser.
- ↳ Dilanjutkan ke L[2]: Simpan nilai L[2] ke variabel X  
**(Pass-2)** Geser masing-masing satu langkah ke kanan semua nilai yang ada disebelah kiri L[2] satu persatu apabila nilai tersebut lebih besar dari X.  
Setelah itu *insert*-kan (sisipkan) X di bekas tempat nilai yang terakhir di geser.
- ↳ Demikian seterusnya untuk L[3], L[4], L[5], dan terakhir L[6] bila n = 7. Sehingga untuk n = 7 ada 6 pass proses pengurutan.

Berikut ilustrasi dari 6 pass tersebut:

Data awal: 

|    |    |   |    |    |   |    |
|----|----|---|----|----|---|----|
| 15 | 10 | 7 | 22 | 17 | 5 | 12 |
| 0  | 1  | 2 | 3  | 4  | 5 | 6  |

**Pass-1:**

|    |    |   |    |    |   |    |    |
|----|----|---|----|----|---|----|----|
| 15 | 10 | 7 | 22 | 17 | 5 | 12 | 10 |
| 0  | 1  | 2 | 3  | 4  | 5 | 6  | X  |

Pass 1 dimulai dari kolom L[1], X=L[1]=10  
15 lebih besar dari 10, maka geser 15 ke kanan. Proses selesai karena sudah sampai kolom 1. Kemudian insert X mengantikan 15.

|    |    |   |    |    |   |    |
|----|----|---|----|----|---|----|
| 15 | 15 | 7 | 22 | 17 | 5 | 12 |
| 0  | 1  | 2 | 3  | 4  | 5 | 6  |

|    |    |   |    |    |   |    |
|----|----|---|----|----|---|----|
| 10 | 15 | 7 | 22 | 17 | 5 | 12 |
| 0  | 1  | 2 | 3  | 4  | 5 | 6  |

Hasil Pass 1:

|    |    |   |    |    |   |    |
|----|----|---|----|----|---|----|
| 10 | 15 | 7 | 22 | 17 | 5 | 12 |
| 0  | 1  | 2 | 3  | 4  | 5 | 6  |

Pass-2:

|    |    |   |    |    |   |    |   |
|----|----|---|----|----|---|----|---|
| 10 | 15 | 7 | 22 | 17 | 5 | 12 | 7 |
| 0  | 1  | 2 | 3  | 4  | 5 | 6  | X |

Pass 2 dimulai dari  $L[2]$ ,  $X=L[2]=7$ .

15 lebih besar dari 7, maka geser 15 ke kanan. 10 lebih besar dari 7, maka geser 10 ke kanan. Proses selesai karena sudah sampai kolom 1. Kemudian insert X menggantikan 10.

|   |    |    |    |    |   |    |
|---|----|----|----|----|---|----|
|   | 15 | 15 | 22 | 17 | 5 | 12 |
| 0 | 1  | 2  | 3  | 4  | 5 | 6  |

|    |    |    |    |    |   |    |
|----|----|----|----|----|---|----|
| 10 | 10 | 15 | 22 | 17 | 5 | 12 |
| 0  | 1  | 2  | 3  | 4  | 5 | 6  |

|   |    |    |    |    |   |    |
|---|----|----|----|----|---|----|
| 7 | 10 | 15 | 22 | 17 | 5 | 12 |
| 0 | 1  | 2  | 3  | 4  | 5 | 6  |

Hasil Pass 2:

|   |    |    |    |    |   |    |
|---|----|----|----|----|---|----|
| 7 | 10 | 15 | 22 | 17 | 5 | 12 |
| 0 | 1  | 2  | 3  | 4  | 5 | 6  |

Pass-3:

|   |    |    |    |    |   |    |    |
|---|----|----|----|----|---|----|----|
| 7 | 10 | 15 | 22 | 17 | 5 | 12 | 22 |
| 0 | 1  | 2  | 3  | 4  | 5 | 6  | X  |

Pass 3 dimulai dari  $L[3]$ ,  $X=L[3]=22$ .

15 tidak lebih besar dari 22, maka proses selesai. Kemudian insert X menggantikan 22.

Proses berlanjut sampai Pass 6. Hasil tiap pass dapat digambarkan sebagai berikut:

Data awal:

|    |    |   |    |    |   |    |
|----|----|---|----|----|---|----|
| 15 | 10 | 7 | 22 | 17 | 5 | 12 |
| 0  | 1  | 2 | 3  | 4  | 5 | 6  |

|         |                                                                                                                                                                                                    |    |    |    |    |    |   |    |   |   |   |   |   |   |   |
|---------|----------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------|----|----|----|----|----|---|----|---|---|---|---|---|---|---|
| Pass 1: | <table border="1"> <tr> <td>10</td><td>15</td><td>7</td><td>22</td><td>17</td><td>5</td><td>12</td></tr> <tr> <td>0</td><td>1</td><td>2</td><td>3</td><td>4</td><td>5</td><td>6</td></tr> </table> | 10 | 15 | 7  | 22 | 17 | 5 | 12 | 0 | 1 | 2 | 3 | 4 | 5 | 6 |
| 10      | 15                                                                                                                                                                                                 | 7  | 22 | 17 | 5  | 12 |   |    |   |   |   |   |   |   |   |
| 0       | 1                                                                                                                                                                                                  | 2  | 3  | 4  | 5  | 6  |   |    |   |   |   |   |   |   |   |

|         |                                                                                                                                                                                                    |    |    |    |    |    |   |    |   |   |   |   |   |   |   |
|---------|----------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------|----|----|----|----|----|---|----|---|---|---|---|---|---|---|
| Pass 2: | <table border="1"> <tr> <td>7</td><td>10</td><td>15</td><td>22</td><td>17</td><td>5</td><td>12</td></tr> <tr> <td>0</td><td>1</td><td>2</td><td>3</td><td>4</td><td>5</td><td>6</td></tr> </table> | 7  | 10 | 15 | 22 | 17 | 5 | 12 | 0 | 1 | 2 | 3 | 4 | 5 | 6 |
| 7       | 10                                                                                                                                                                                                 | 15 | 22 | 17 | 5  | 12 |   |    |   |   |   |   |   |   |   |
| 0       | 1                                                                                                                                                                                                  | 2  | 3  | 4  | 5  | 6  |   |    |   |   |   |   |   |   |   |

|         |                                                                                                                                                                                                    |    |    |    |    |    |   |    |   |   |   |   |   |   |   |
|---------|----------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------|----|----|----|----|----|---|----|---|---|---|---|---|---|---|
| Pass 3: | <table border="1"> <tr> <td>7</td><td>10</td><td>15</td><td>22</td><td>17</td><td>5</td><td>12</td></tr> <tr> <td>0</td><td>1</td><td>2</td><td>3</td><td>4</td><td>5</td><td>6</td></tr> </table> | 7  | 10 | 15 | 22 | 17 | 5 | 12 | 0 | 1 | 2 | 3 | 4 | 5 | 6 |
| 7       | 10                                                                                                                                                                                                 | 15 | 22 | 17 | 5  | 12 |   |    |   |   |   |   |   |   |   |
| 0       | 1                                                                                                                                                                                                  | 2  | 3  | 4  | 5  | 6  |   |    |   |   |   |   |   |   |   |

|         |                                                                                                                                                                                                    |    |    |    |    |    |   |    |   |   |   |   |   |   |   |
|---------|----------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------|----|----|----|----|----|---|----|---|---|---|---|---|---|---|
| Pass 4: | <table border="1"> <tr> <td>7</td><td>10</td><td>15</td><td>17</td><td>22</td><td>5</td><td>12</td></tr> <tr> <td>0</td><td>1</td><td>2</td><td>3</td><td>4</td><td>5</td><td>6</td></tr> </table> | 7  | 10 | 15 | 17 | 22 | 5 | 12 | 0 | 1 | 2 | 3 | 4 | 5 | 6 |
| 7       | 10                                                                                                                                                                                                 | 15 | 17 | 22 | 5  | 12 |   |    |   |   |   |   |   |   |   |
| 0       | 1                                                                                                                                                                                                  | 2  | 3  | 4  | 5  | 6  |   |    |   |   |   |   |   |   |   |

|         |                                                                                                                                                                                                    |    |    |    |    |    |    |    |   |   |   |   |   |   |   |
|---------|----------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------|----|----|----|----|----|----|----|---|---|---|---|---|---|---|
| Pass 5: | <table border="1"> <tr> <td>5</td><td>7</td><td>10</td><td>15</td><td>17</td><td>22</td><td>12</td></tr> <tr> <td>0</td><td>1</td><td>2</td><td>3</td><td>4</td><td>5</td><td>6</td></tr> </table> | 5  | 7  | 10 | 15 | 17 | 22 | 12 | 0 | 1 | 2 | 3 | 4 | 5 | 6 |
| 5       | 7                                                                                                                                                                                                  | 10 | 15 | 17 | 22 | 12 |    |    |   |   |   |   |   |   |   |
| 0       | 1                                                                                                                                                                                                  | 2  | 3  | 4  | 5  | 6  |    |    |   |   |   |   |   |   |   |

|         |                                                                                                                                                                                                    |    |    |    |    |    |    |    |   |   |   |   |   |   |   |
|---------|----------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------|----|----|----|----|----|----|----|---|---|---|---|---|---|---|
| Pass 6: | <table border="1"> <tr> <td>5</td><td>7</td><td>10</td><td>12</td><td>15</td><td>17</td><td>22</td></tr> <tr> <td>0</td><td>1</td><td>2</td><td>3</td><td>4</td><td>5</td><td>6</td></tr> </table> | 5  | 7  | 10 | 12 | 15 | 17 | 22 | 0 | 1 | 2 | 3 | 4 | 5 | 6 |
| 5       | 7                                                                                                                                                                                                  | 10 | 12 | 15 | 17 | 22 |    |    |   |   |   |   |   |   |   |
| 0       | 1                                                                                                                                                                                                  | 2  | 3  | 4  | 5  | 6  |    |    |   |   |   |   |   |   |   |

Selesai array L sudah terurut secara Ascending (menaik)

Pseudocode Algoritma *Insertion Sort* secara *Ascending* :

1. //prosedur algoritma Insertion Sort secara Ascending
2. //I.S:array sudah berisi nilai integer yang belum terurut
3. //F.S:nilai-nilai dalam array terurut secara Ascending
4. **procedure** v\_inAsc(**input/output** A:array[0..6]of integer,  
**input** N:integer)
5. **KAMUS:**
6. k,X,i:integer
7. **ALGORITMA:**
8. //insertion sort ascending
9. k←1
10. **while**(k<=N-1)
11. i←k
12. X←A[i]
13. **while**(i>=1 && A[i-1]>X)
14. A[i]←A[i-1]
15. i←i-1
16. **endwhile**
17. A[i]←X
18. k←k+1
19. **endwhile**

Program lengkap penerapan algoritma *Insertion Sort Ascending* dalam bahasa C

```
#include <stdio.h>
#include <conio.h>
main()
{ int L[7];
int i,N;
void v_insertAsc(int A[7],int N);

//input data array
printf("Input Data Array\n");
printf("\nBanyak Data: "); scanf("%i",&N);
for(i=0;i<N;i++)
{ printf("Nilai ke-%i = ",i+1);
scanf("%i",&L[i]); } //end loop i
//panggil procedure v_inAsc
v_insAsc(L,N);
//output data array
printf("Data terurut:\n");
for(i=0;i<N;i++)
{ printf("%5i",L[i]); } //end loop i
printf("\nTekan Enter...\n");
getche();
}

void v_insAsc(int A[7],int N)
{ int k,X,i;
//insertion sort ascending
k=1;
while(k<=N-1)
{ i=k;
X=A[i];
while(i>=1 && A[i-1]>X)
{ A[i]=A[i-1];
i--; } //endwhile
A[i]=X;
k++; } //endwhile
} //end procedure
```

Output yang dihasilkan:

```
C:\ D:\My Documents\in D\Maya\Data Maya 280502\LATIHAN C\procIns  
Input Data Array  
Banyak Data: 7  
Nilai ke-1 = 15  
Nilai ke-2 = 10  
Nilai ke-3 = 7  
Nilai ke-4 = 22  
Nilai ke-5 = 17  
Nilai ke-6 = 5  
Nilai ke-7 = 12  
Data terurut:  
      5    7   10   12   15   17   22  
Tekan Enter...
```



## Rangkuman

---

---

- Proses Sorting** merupakan proses mengurutkan data yang berada dalam suatu tempat penyimpanan, dengan urutan tertentu baik urut menaik (*ascending*) dari nilai terkecil sampai dengan nilai terbesar, atau urut menurun (*descending*) dari nilai terbesar sampai dengan nilai terkecil
- Terdapat dua macam proses pengurutan, yaitu pengurutan internal (*internal sort*) dan pengurutan eksternal (*external sort*).
- Bubble sort* adalah proses pengurutan sederhana yang bekerja dengan cara berulang kali membandingkan dua elemen data pada suatu saat dan menukar elemen data yang urutannya salah.
- Algoritma *Selection sort* memilih elemen maksimum/minimum *array*, lalu menempatkan elemen maksimum/minimum itu pada awal atau akhir *array* (tergantung pada urutannya *ascending/descending*).
- Algoritma *Insertion Sort*, mencari tempat yang "tepat" untuk setiap elemen *array*, dengan cara *sequential search*. Proses ini kemudian menyisipkan sebuah elemen *array* yang diproses ke tempatnya yang seharusnya.