TEN

Sorting

PENDAHULUAN

- Pengurutan data dalam struktur data sangat penting terutama untuk data yang beripe data numerik ataupun karakter.
- Pengurutan dapat dilakukan secara ascending (urut naik) dan descending (urut turun)
- Pengurutan (Sorting) adalah proses pengurutan data yang sebelumnya disusun secara acak sehingga tersusun secara teratur menurut aturan tertentu

Contoh:
Data Acak : 5 6 8 1 3 25 10

Ascending : 1 3 5 6 8 10 25

Descending : 25 10 8 6 5 3 1

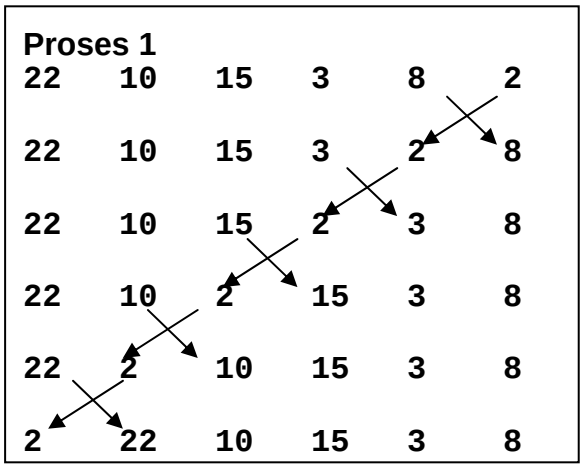
DEKLARASI ARRAY UNTUK SORTING

Deklarasikan secara global:
```
int data[100];
int n; //untuk jumlah data
```

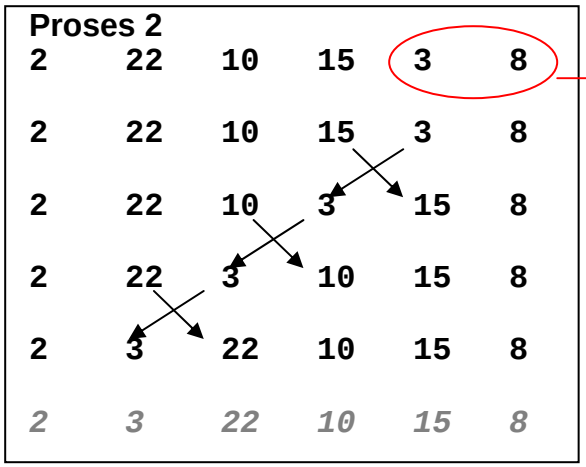
Prosedur Tukar 2 Buah Data:
```
void tukar(int a,int b){
     int tmp;
     tmp = data[a];
     data[a] = data[b];
     data[b] = tmp;
}
```

BUBBLE SORT

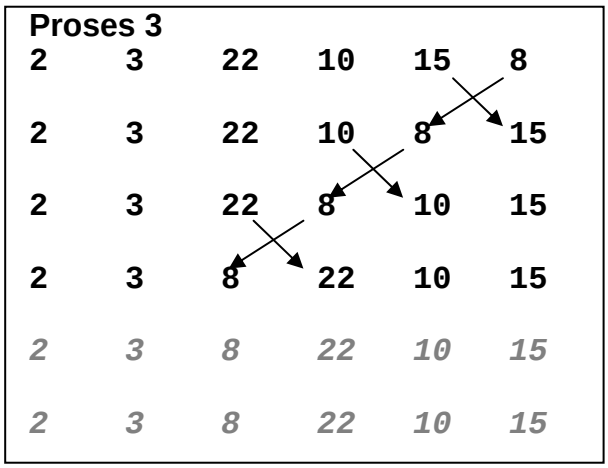
- Metode sorting termudah
- Diberi nama “Bubble” karena proses pengurutan secara berangsur-angsur bergerak/berpindah ke posisinya yang tepat, seperti gelembung yang keluar dari sebuah gelas bersoda.
- Bubble Sort mengurutkan data dengan cara membandingkan elemen sekarang dengan elemen berikutnya.
- Jika elemen sekarang **lebih besar** dari elemen berikutnya maka kedua elemen tersebut **ditukar,** jika pengurutan **ascending**.
- Jika elemen sekarang **lebih kecil** dari elemen berikutnya, maka kedua elemen tersebut **ditukar**, jika pengurutan **descending**
- Algoritma ini seolah-olah menggeser satu per satu elemen dari kanan ke kiri atau kiri ke kanan, tergantung jenis pengurutannya.
- Ketika satu proses telah selesai, maka bubble sort akan mengulangi proses, demikian seterusnya.
- Kapan berhentinya? Bubble sort berhenti jika seluruh array telah diperiksa dan tidak ada pertukaran lagi yang bisa dilakukan, serta tercapai perurutan yang telah diinginkan.

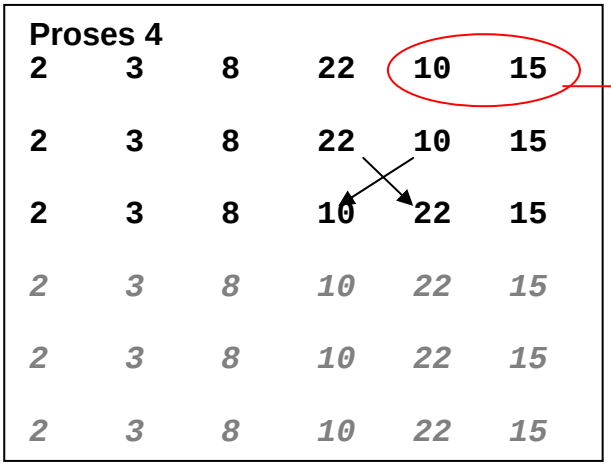


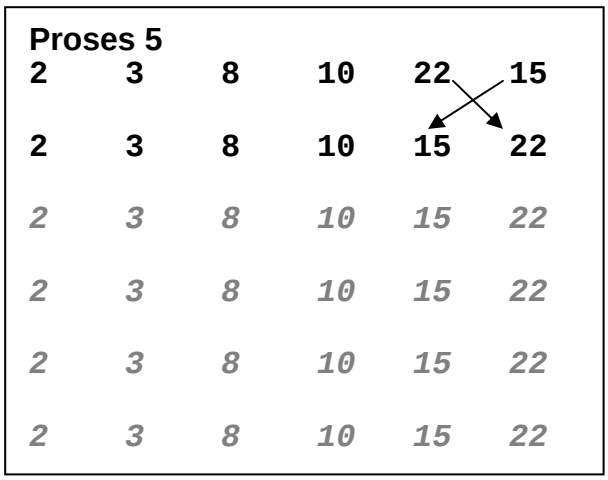
Pada gambar diatas, pegecekan dimulai dari data yang paling akhir, kemudian dibandingkan dengan data di depannya, jika data di depannya lebih besar maka akan ditukar.



Pada proses kedua, pengecekan dilakukan sampai dengan data ke-2 karena data pertama pasti sudah paling kecil.







Prosedur Bubble Sort

```
void bubble_sort(){
     for(int i=1;i<n;i++){
          for(int j=n-1;j>=i;j--){
               if(data[j]<data[j-1]) tukar(j,j-1); //ascending
          }
     }
}
```

Dengan prosedur diatas, **data terurut naik (ascending),** untuk **urut turun (descending)** silahkan ubah bagian:

```
if (data[j]<data[j-1]) tukar(j,j-1);
```

Menjadi:

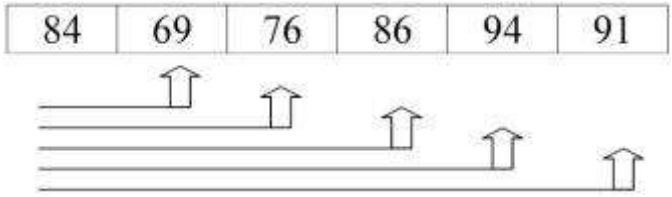
```
if (data[j]>data[j-1]) tukar(j,j-1);
```

"The bubble sort is an easy algorithm to program, but it is slower than many other sorts"

EXCHANGE SORT

- Sangat mirip dengan Bubble Sort
- Banyak yang mengatakan Bubble Sort sama dengan Exchange Sort
- Pebedaan : dalam hal bagaimana membandingkan antar elemen-elemennya.

- Exchange sort membandingkan suatu **elemen** dengan **elemen-elemen lainnya** dalam array tersebut, dan melakukan pertukaran elemen jika perlu. Jadi ada elemen yang selalu menjadi elemen **pusat (pivot).**
- Sedangkan Bubble sort akan membandingkan **elemen pertama/terakhir** dengan **elemen sebelumnya/sesudahnya**, kemudian elemen sebelum/sesudahnya itu akan menjadi **pusat (pivot)** untuk dibandingkan dengan elemen sebelumnya/sesudahnya lagi, begitu seterusnya.



Proses 1

Pivot (Pusat)

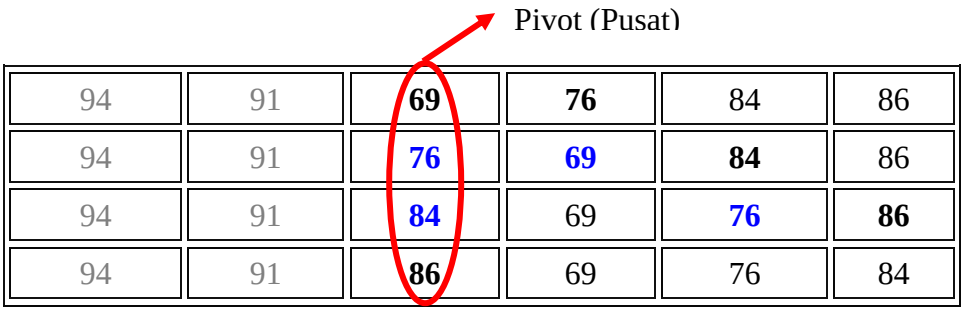
84	69	76	86	94	91
84	69	76	86	94	91
84	69	76	**86**	94	91
86	69	76	**84**	**94**	91
94	69	76	84	**86**	91
94	69	76	84	86	91

Proses 2

Pivot (Pusat)

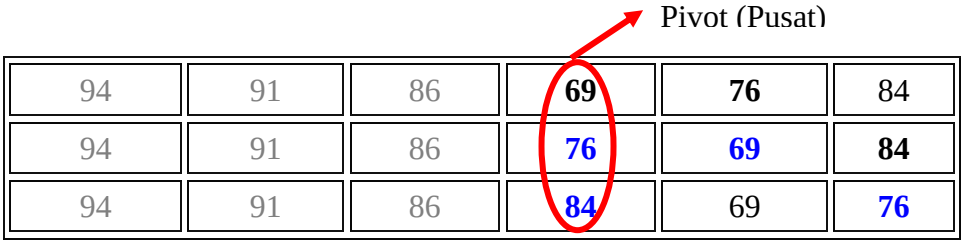
94	**69**	**76**	84	86	91
94	**76**	**69**	**84**	86	91
94	**84**	69	**76**	**86**	91
94	**86**	69	76	**84**	**91**
94	**91**	69	76	84	86

Proses 3



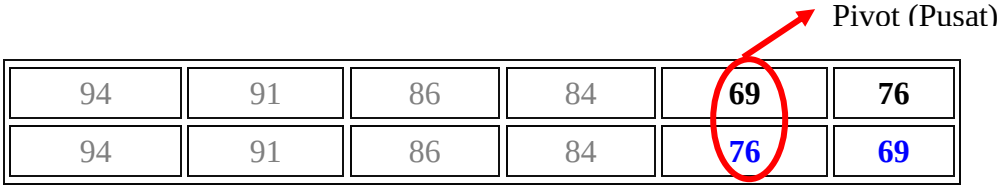
94	91	**69**	**76**	84	86
94	91	**76**	**69**	**84**	86
94	91	**84**	69	**76**	**86**
94	91	**86**	69	76	84

Proses 4



94	91	86	**69**	**76**	84
94	91	86	**76**	**69**	**84**
94	91	86	**84**	69	**76**

Proses 5



94	91	86	84	**69**	**76**
94	91	86	84	**76**	**69**

Prosedur Exchange Sort

```
void exchange_sort()
{
     for (int i=0; i<n-1; i++){
          for(int j = (i+1); j<n; j++){
               if (data [i] < data[j]) tukar(i,j); //descending
          }
     }
}
```

SELECTION SORT

- Merupakan kombinasi antara sorting dan searching
- Untuk setiap proses, akan dicari elemen-elemen yang belum diurutkan yang memiliki nilai terkecil atau terbesar akan dipertukarkan ke posisi yang tepat di dalam array.
- Misalnya untuk putaran pertama, akan dicari data dengan nilai terkecil dan data ini akan ditempatkan di indeks terkecil (data[0]), pada putaran kedua akan dicari data kedua terkecil, dan akan ditempatkan di indeks kedua (data[1]).

- Selama proses, pembandingan dan pengubahan **hanya dilakukan** pada **indeks** pembanding saja, pertukaran data secara fisik terjadi pada **akhir** proses.

Proses 1

0	1	2	3	4	5
32	75	69	58	21	40

Pembanding	**Posisi**
32 < 75	0
32 < 69	0
32 < 58	0
32 > 21 (**tukar idx**)	4
21 < 40	4

Tukar data ke-0 (**32**) dengan data ke-4 (**21**)

0	1	2	3	4	5
21	75	69	58	32	40

Proses 2

0	1	2	3	4	5
21	**75**	69	58	32	40

Pembanding	**Posisi**
75 > 69 (**tukar idx**)	2
69 > 58 (**tukar idx**)	3
58 > 32 (**tukar idx**)	4
32 < 40	4

Tukar data ke-1 (**75**) dengan data ke-4 (**32**)

0	1	2	3	4	5
21	32	69	58	75	40

Proses 3

0	1	2	3	4	5
21	**32**	**69**	58	75	40

Pembanding	**Posisi**
69 > 58 (**tukar idx**)	3
58 < 75	3
58 > 40	5

Tukar data ke-2 (**69**) dengan data ke-5 (**40**)

0	1	2	3	4	5
21	32	40	58	75	69

Proses 4

0	1	2	3	4	5
21	**32**	**40**	**58**	75	69

Pembanding	**Posisi**
58 < 75	3
58 < 69	3

Tukar data ke-3 (**58**) dengan data ke-3 (**58**)

0	1	2	3	4	5
21	32	40	58	75	69

Proses 5

0	1	2	3	4	5
21	**32**	**40**	**58**	**75**	69

Pembanding	**Posisi**
75 > 69	5

Tukar data ke-4 (**75**) dengan data ke-5 (**69**)

0	1	2	3	4	5
21	32	40	58	69	75

Prosedur Selection Sort

```
void selection_sort(){
     for(int i=0;i<n-1;i++){
          pos = i;
          for(int j=i+1;j<n;j++){
               if(data[j] < data[pos]) pos = j;   //ascending

          }
          if(pos != i) tukar(pos,i);
     }
}
```

INSERTION SORT

- Mirip dengan cara orang **mengurutkan** kartu, selembar demi selembar kartu diambil dan **disisipkan** (insert) ke tempat yang seharusnya.
- Pengurutan dimulai dari data ke-2 sampai dengan data terakhir, jika ditemukan data yang **lebih kecil**, maka akan ditempatkan (**diinsert**) diposisi yang seharusnya.
- Pada penyisipan elemen, maka elemen-elemen lain akan bergeser ke belakang.

Proses 1

0	1	2	3	4	5
22	**10**	15	3	8	2

Temp	Cek	Geser
10	Temp<22?	Data ke-0 ke posisi 1

Temp menempati posisi ke -0

0	1	2	3	4	5
10	22	15	3	8	2

Proses 2

0	1	2	3	4	5
10	22	**15**	3	8	2

Temp	Cek	Geser
15	Temp<22	Data ke-1 ke posisi 2
15	Temp>10	-

Temp menempati posisi ke-1

0	1	2	3	4	5
10	**15**	22	3	8	2

Proses 3

0	1	2	3	4	5
10	15	22	**3**	8	2

Temp	Cek	Geser
3	Temp<22	Data ke-2 ke posisi 3
3	Temp<15	Data ke-1 ke posisi 2
3	Temp<10	Data ke-0 ke posisi 1

Temp menempati posisi ke-0

0	1	2	3	4	5
3	10	15	22	8	2

Proses 4

0	1	2	3	4	5
3	10	15	22	**8**	2

Temp	Cek	Geser
8	Temp<22	Data ke-3 ke posisi 4
8	Temp<15	Data ke-2 ke posisi 3
8	Temp<10	Data ke-1 ke posisi 2
8	Temp>3	-

Temp menempati posisi ke-1

0	1	2	3	4	5
3	**8**	10	15	22	2

Proses 5

0	1	2	3	4	5
3	8	10	15	22	**2**

Temp	Cek	Geser
2	Temp<22	Data ke-4 ke posisi 5
2	Temp<15	Data ke-3 ke posisi 4
2	Temp<10	Data ke-2 ke posisi 3
2	Temp<8	Data ke-1 ke posisi 2
2	Temp<3	Data ke-0 ke posisi 1

Temp menempati posisi ke-0

0	1	2	3	4	5
2	3	8	10	15	22

Prosedur Insertion Sort

```
void insertion_sort(){
     int temp;
     for(int i=1;i<n;i++){
          temp = data[i];
          j = i -1;
          while(data[j]>temp && j>=0){
               data[j+1] = data[j];
               j--;
          }
          data[j+1] = temp;
     }
}
```

Program Lengkapnya:

```
#include <stdio.h>
#include <conio.h>

int data[10],data2[10];
int n;

void tukar(int a,int b){
     int t;
     t = data[b];
     data[b] = data[a];
     data[a] = t;
}

void bubble_sort(){
     for(int i=1;i<n;i++){
          for(int j=n-1;j>=i;j--){
               if(data[j]<data[j-1]) tukar(j,j-1);
          }
     }
     printf("bubble sort selesai!\n");
}

void exchange_sort(){
     for (int i=0; i<n-1; i++){
               for(int j = (i+1); j<n; j++){
                    if (data [i] > data[j]) tukar(i,j);
           }
     }
     printf("exchange sort selesai!\n");
}
```

```
void selection_sort(){
     int pos,i,j;
     for(i=0;i<n-1;i++){
          pos = i;
          for(j = i+1;j<n;j++){
               if(data[j] < data[pos]) pos = j;
          }
          if(pos != i) tukar(pos,i);
     }
     printf("selection sort selesai!\n");
}

void insertion_sort(){
     int temp,i,j;
     for(i=1;i<n;i++){
          temp = data[i];
          j = i -1;
          while(data[j]>temp && j>=0){
               data[j+1] = data[j];
               j--;
          }
          data[j+1] = temp;
     }
     printf("insertion sort selesai!\n");
}

void Input(){
     printf("Masukkan jumlah data = ");scanf("%d",&n);
     for(int i=0;i<n;i++){
          printf("Masukkan data ke-%d =
",(i+1));scanf("%d",&data[i]);
          data2[i] = data[i];
     }
}

void AcakLagi(){
     for(int i=0;i<n;i++){
          data[i] = data2[i];
     }
     printf("Data sudah teracak!\n");
}

void Tampil(){
          printf("Data : ");
          for(int i=0;i<n;i++){
               printf("%d ",data[i]);
          }
          printf("\n");
}
```

```
void main(){
     clrscr();
     int pil;
     do{
          clrscr();
          printf("1. Input Data\n");
          printf("2. Bubble Sort\n");
          printf("3. Exchange Sort\n");
          printf("4. Selection Sort\n");
          printf("5. Tampilkan Data\n");
          printf("6. Acak\n");
          printf("7. Exit\n");
          printf("Pilihan = ");scanf("%d",&pil);
          switch(pil){
               case 1:Input();break;
               case 2:bubble_sort();break;
               case 3:exchange_sort();break;
               case 4:selection_sort();break;
               case 5:Tampil();break;
               case 6:AcakLagi();break;
          }
          getch();
     }while(pil!=7);
}
```

Tabel Perbandingan Kecepatan Metode Pengurutan Data

Untuk data sejumlah 10.000 data pada komputer Pentium II 450 MHz

Metode	Waktu (detik)		
	Data Acak	Data Ascending	Data Descending
Bubble Sort	11,2	1,32	19,77
Insertion Sort	1,09	0,00	2,25
Selection Sort	1,32	1,32	19,77

SEARCHING

Binary Search

- Adalah teknik pencarian data dalam array dengan cara membagi array menjadi dua bagian setiap kali terjadi proses pengurutan.
- Prinsip pencarian biner adalah:
    - Data diambil dari posisi 1 sampai posisi akhir N
    - Kemudian cari posisi data tengah dengan rumus (posisi awal + posisi akhir) / 2
    - Kemudian data yang dicari dibandingkan dengan data yang di tengah, apakah sama atau lebih kecil, atau lebih besar?

- Jika lebih besar, maka proses pencarian dicari dengan posisi awal adalah posisi tengah + 1
- Jika lebih kecil, maka proses pencarian dicari dengan posisi akhir adalah posisi tengah – 1
- Jika data sama, berarti ketemu.

Contoh Data:
Misalnya data yang dicari **17**

0	1	2	3	4	5	6	7	8
3	**9**	**11**	**12**	**15**	**17**	**23**	**31**	**35**
A				**B**				**C**

Karena 17 > 15 (data tengah), maka: awal = tengah + 1

0	1	2	3	4	5	6	7	8
3	**9**	**11**	**12**	**15**	**17**	**23**	**31**	**35**
					A	**B**		**C**

Karena 17 < 23 (data tengah), maka: akhir = tengah – 1

0	1	2	3	4	5	6	7	8
3	**9**	**11**	**12**	**15**	**17**	**23**	**31**	**35**
					A=B=C			

Karena 17 = 17 (data tengah), maka KETEMU!

Programnya:

```
int binary_search(int cari){
     int l,r,m;
     l = 0;
     r = n-1;
     int ktm = 0;

     while(l<=r && ktm==0){
          m = (l+r)/2;
          if(data[m] == cari) ktm=1;
          else if (cari < data[m]) r=m-1;
          else l=m+1;
     }

     if(ktm==1) return 1; else return 0;
}
```

NEXT, THE LAST CLASS : FUNCTION by Reference dan by Value ☺