ARRAY STATIS

Array (larik) merupakan tipe data terstruktur yang terdiri dari sejumlah elemen yang mempunyai **tipe data yang sama** dan diakses/diacu lewat indeksnya. Array memiliki jumlah komponen yang jumlahnya tetap.

Banyaknya komponen dalam array ditunjukkan oleh suatu indeks. Array dapat bertipe data sederhana/dasar ataupun bertipe data terstruktur lainnya seperti record.

Elemen-elemen dari array tersusun secara sekuensial (berurutan/kontigu) dalam memori komputer. Array dapat berupa satu dimensi, dua dimensi, tiga dimensi ataupun banyak dimensi.

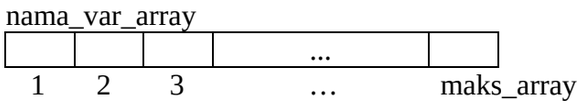
Deklarasi Array satu dimensi secara Umum:

```
Kamus :
    Const
       maks_array : integer = .....

    Type
       namatype_array = array [1..maks_array] of tipedata

    nama_var_array : namatype_array
    indeks : integer              {indeks array dari 1 sampai maksimum array}
```

Array 1 dimensi dapat digambarkan seperti di bawah ini :



Contoh:

```
Kamus :
    Const
       maks_array : integer = 100

    Type
       Larik = array [1..maks_array] of integer

    A : Larik
    i  : integer              {indeks array dr 1 sampai 100}
```

Contoh deklarasi array di atas menyatakan bahwa terdapat array dengan nama Larik telah dideklarasikan dengan tipe integer (bilangan bulat), dengan jumlah maksimum elemen array 100 elemen, nilai dari elemen array tersebut bertipe integer.

Proses-proses terhadap data array:

1. **Penciptaan (*Create*)Array**
   Mempersiapkan array untuk diakses/diproses dengan asumsi elemen array diisi dengan angka 0 jika elemen arraynya numerik/angka/bilangan, tapi jika alphanumerik, maka diisi dengan tanda ‘ ‘.
   Prosedur penciptaan secara umum sebagai berikut :

```
Procedure  create (Output nama_var_array : namatype_array)
{I.S : maksimum array sudah terdefinisi}
{F.S : menghasilkan array yang siap diproses}
Kamus :

Algoritma :

   for indeks ← 1  to  maks_array  do
         nama_var_array(indeks) ← 0          {Asumsi isi elemen array bilangan}
   endfor
EndProcedure
```

2. **Traversal**
   Proses mengunjungi setiap elemen array secara berurutan dari elemen pertama sampai elemen terakhir.
   Yang termasuk dalam proses traversal adalah :
   - Pengisian elemen array dengan data
   - Menampilkan elemen array
   - Penambahan data di array
   - Penyisipan data di indeks tertentu pada array
   - Penghapusan data di indeks tertentu pada array
   - Menentukan nilai maksimum/minimum
   - Menghitung rata-rata nilai
   - dll

   Prosedur traversal secara umum :

```
Procedure  traversal (I/O nama_var_array : namatype_array)
{I.S : maksimum array sudah terdefinisi}
{F.S : menghasilkan array yang siap diproses}
Kamus :

Algoritma :

   Inisialisasi      {pemberian nilai awal terhadap suatu variabel}
   for indeks ← 1  to  maks_array  do
         Proses
   endfor
   Terminasi      {penutupan yang harus dilakukan setelah pemrosesan selesai}
EndProcedure
```

3. Pencarian data (Searching)

Pencarian data berarti proses mencari suatu data yang terdapat dalam suatu array. Proses ini menghasilkan nilai benar atau salah, jika benar berarti data yang dicari ada dalam array, jika salah berarti data yang dicari tidak ditemukan.

Metode pencarian data adalah sebagai berikut:

1. Sequential/Linear Search

Pencarian yang dilakukan secara berurutan dari data awal sampai data ditemukan atau data terakhir.

Pencarian sekuensial dibagi menjadi 2 yaitu :

a. Pencarian sekuensial tanpa Boolean
   - Tanpa Sentinel
   - Dengan Sentinel

b. Pencarian sekuensial dengan Boolean

2. Binary Search

Pencarian yang dilakukan terhadap data yang sudah terurut baik secara ascending maupun descending. Pencarian teknik ini mempunyai beban komputasi yang lebih ringan dibandingkan dengan pencarian sekuensial. Mekanismenya adalah dengan cara membagi array menjadi 2 bagian yaitu bagian kiri dari indeks terkecil/ Ia sampai ke indeks tengah dan bagian kanan dari indeks tengah sampai keindeks terbesar/Ib. Indeks tengah (k) didapat dari :

k = (Ia + Ib) div 2

Jika data yang dicari lebih kecil dari data di indeks tengah, maka pencarian dilanjutkan kearray bagian kiri, tapi jika data yang dicari lebih besar dari data di indeks tengah, maka pencarian dilanjutkan kebagian kanan, dan seterusnya sampai data ditemukan atau sampai Ia>Ib.

Contoh 1 (kasus data yang dicari ada dalam array):

Data yang dicari adalah : 50

Banyak data : 10

Array :

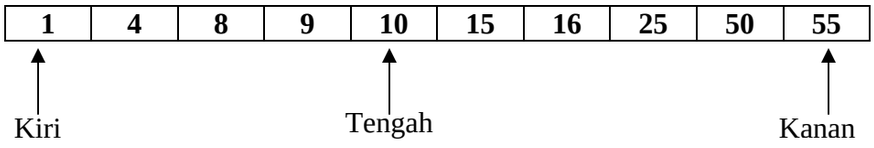
1	4	8	9	10	15	16	25	50	55

Urutan pencariannya adalah :

Kiri = 1

Kanan = 10

Tengah = (1+10)/2 = 5,5 = 5 [ambil bilangan bulatnya]



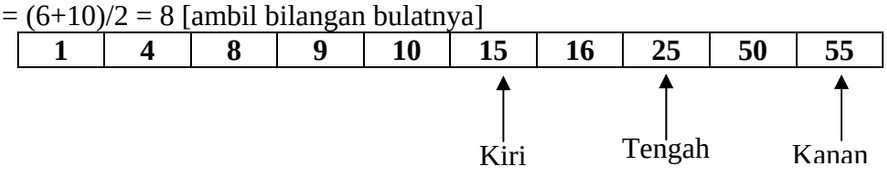
Kemudian bandingkan data yang dicari (50) dengan data posisi tengah (10). Ada 3 kemungkinan yang terjadi, yaitu :

- Jika data posisi tengah > data yang dicari, maka pencarian dilakukan di array bagian kiri, dengan cara memindahkan kanan ke posisi tengah-1 (kanan=tengah-1) dan posisi tengah dicari kembali dengan rumus (kiri+kanan)/2.
- Jika data posisi tengah < data yang dicari, maka pencarian dilakukan di array bagian kanan, dengan cara memindahkan posisi kiri dengan posisi tengah+1, dan posisi tengah dicari kembali dengan rumus (kiri+kanan)/2.

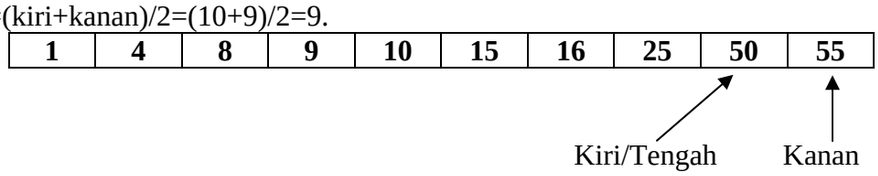
- Jika data posisi tengah = data yang dicari, berarti data telah ditemukan pada posisi tengah.

  Dengan kasus di atas (data[tengah]=10 dan dicari=50), maka pencarian yang akan dilakukan adalah pada array bagian kanan. Oleh karena itu posisi kiri berpindah dari 0 menjadi 6 (tengah+1).

  Kiri = 6
  Kanan = 10
  Tengah = (6+10)/2 = 8 [ambil bilangan bulatnya]

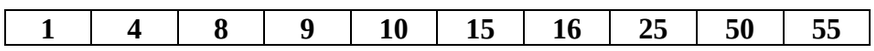


  Lakukan kembali proses pembandingan seperti tadi sehingga didapatkan bahwa nilai yang dicari lebih besar daripada nilai di posisi tengah, maka pencarian dilakukan lagi pada bagian kanan dengan kondisi :
  Kiri=tengah+1=9
  Kanan=10
  Tengah=(kiri+kanan)/2=(10+9)/2=9.

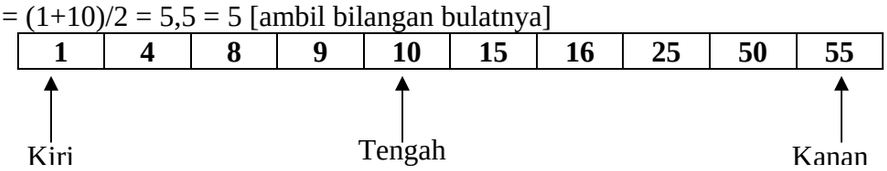


  Kemudian lakukan lagi perbandingan, sehingga didapatkan bahwa data yang dicari (50) sudah sama dengan data posisi tengah, maka itu berarti data telah ditemukan pada posisi ke 9.

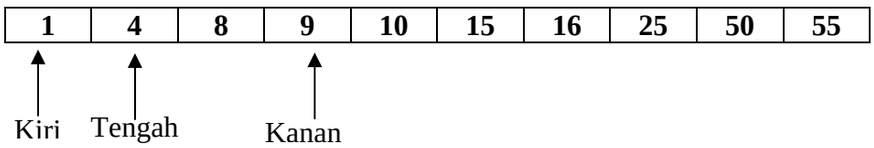
  Contoh 2 (kasus data yang dicari tidak ada dalam array) :
  Data yang dicari adalah : 5
  Banyak data : 5
  Array :



  Urutan pencariannya adalah :
  Kiri = 1
  Kanan = 10
  Tengah = (1+10)/2 = 5,5 = 5 [ambil bilangan bulatnya]



  Kemudian bandingkan data yang dicari (5) dengan data posisi tengah (10), sehingga didapatkan bahwa data yang dicari lebih kecil dari data posisi tengah, sehingga pencarian dilakukan pada array bagian kiri. Kondisi pencariannya :
  Kiri=1;
  Kanan=Tengah-1 = 5-1 = 4
  Tengah=(1+Kanan)/2 = (1+4)/2= 5/2 =2

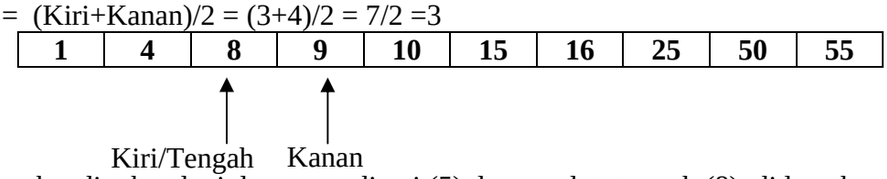


Kemudian bandingkan lagi data yang dicari (5) dengan data yang di posisi tengah (4). Didapatkan kondisi bahwa data yang dicari (5) lebih besar dari posisi tengah (4), maka pencarian kembali dilakukan di posisi kanan, sehingga kondisi pencarian adalah :

Kiri = Tengah+1 = 2+1 =3

Kanan = 4

Tengah = (Kiri+Kanan)/2 = (3+4)/2 = 7/2 =3

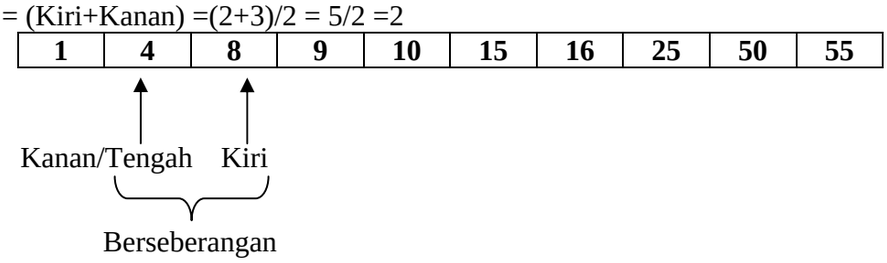


Kemudian bandingkan lagi data yang dicari (5) dengan data tengah (8), didapatkan kondisi bahwa data yang dicari lebih kecil dari data tengah sehingga pencarian harus dilakukan di posisi kiri sehingga kondisi pencarian adalah :

Kanan = (Tengah-1) = 3-1 =2

Kiri = 3

Tengah = (Kiri+Kanan) =(2+3)/2 = 5/2 =2



Karena posisi kiri telah melebihi posisi kanan, maka pencarian selesai dan itu menunjukan bahwa data tidak ditemukan

4. Pengurutan data (Sorting)

Proses menyusun data yang ada dalam array sehingga dapat terurut baik secara menaik (ascending) atau menurun (descending).

Adapun algoritma yang bisa digunakan sebagai berikut :

1. Buble Sort

   Algoritma pengurutan dengan menggunakan teknik bubble sort adalah dengan membandingkan sebuah elemen array ke-i dengan elemen array berikutnya (elemen ke-i+1). Jika isi elemen array ke-i lebih besar dari elemen array ke-i+1, maka tukarkan isinya. Algoritma ini adalah algoritma pengurutan yang paling sederhana, tetapi paling lambat untuk array yang memiliki elemen yang banyak.

   Contoh Kasus dengan banyak data adalah 10 :

Array:	5	3	7	9	2	3	6	4	3	1

   Catatan : untuk kasus di atas sengaja angka terkecil dipakai di akhir data sehingga ini akan menghasilkan kondisi terburuk dari pengurutan.

Awal:	5	3	7	9	2	3	6	4	3	1

L. 1	3	5	7	2	3	6	4	3	1	**9**
L. 2	3	5	2	3	6	4	3	1	**7**	**9**
L. 3	3	2	3	5	4	3	1	**6**	**7**	**9**
L. 4	2	3	3	4	3	1	**5**	**6**	**7**	**9**
L. 5	2	3	3	3	1	**4**	**5**	**6**	**7**	**9**
L. 6	2	3	3	1	**3**	**4**	**5**	**6**	**7**	**9**
L. 7	2	3	1	**3**	**3**	**4**	**5**	**6**	**7**	**9**
L. 8	2	1	**3**	**3**	**3**	**4**	**5**	**6**	**7**	**9**
L. 9	1	**2**	**3**	**3**	**3**	**4**	**5**	**6**	**7**	**9**

Catatan : data yang diarsir adalah data yang telah terurut.

Dari gambaran di atas, ketika banyak data adalah 10, maka proses pertukaran dilakukan sebanyak 9 langkah. Jadi dapat diketahui bahwa langkah yang dilakukan agar data terurut maksimal sebanyak N-1 langkah dimana N adalah banyaknya data.

Algoritma pengurutan data dengan algoritma bubble sort adalah :

a. Lakukan perulangan untuk ***i*** dimulai dari 1 sampai N-1, lakukan proses b.
b. Lakukan perulangan j dimulai dari 1 sampai N-1, lakukan perbandingan antara data ke-j dengan data ke-(j+1). Jika data ke-j lebih besar dari data ke-(j+1), maka tukarkan datanya.

2. Selection Sort

Algoritma selection sort bekerja dengan cara menyimpan data terkecil dari data ke posisi kiri. Data paling kiri yang telah terurut tidak perlu dibandingkan lagi. Untuk lebih jelas, perhatikan gambaran pengurutan di bawah ini :

Kondisi Awal :

Awal:	5	3	7	9	2	3	6	4	3	1

Langkah 1 :

Cari index bilangan terkecil dari data ke-1 sampai data terakhir terakhir untuk disimpan di posisi ke-1. Dari data di atas didapatkan bahwa data terkecil berada di index ke-10. Tukarkan isi data ke-1 dengan indek ke 10.

5	3	7	9	2	3	6	4	3	**1**

Menjadi

1	3	7	9	2	3	6	4	3	5

Langkah ke-2 :

Cari index bilangan terkecil dari data ke-2 sampai data terakhir untuk disimpan di posisi-2. Dari data sebelumnya didapatkan bahwa data terkecil berada di index ke-5. Tukarkan isi data ke-2 dengan data index ke-5.

1	3	7	9	2	3	6	4	3	5

Menjadi

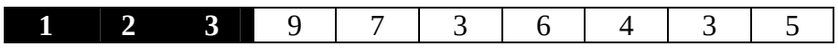
1	**2**	7	9	3	3	6	4	3	5

Langkah ke-3 :

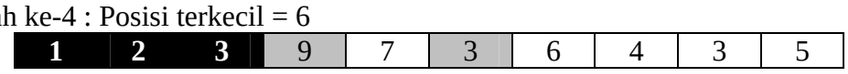
Cari index bilangan terkecil dari data ke-3 sampai data terakhir untuk disimpan di posisi-3. Dari data sebelumnya didapatkan bahwa data terkecil berada di index ke-5. Tukarkan isi data ke-3 dengan data index ke-5.

1	**2**	7	9	3	3	6	4	3	5

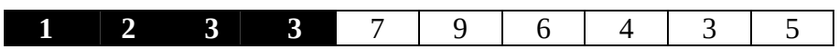
Menjadi



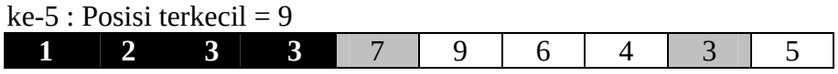
Langkah ke-4 : Posisi terkecil = 6



Menjadi



Langkah ke-5 : Posisi terkecil = 9



Menjadi

Langkah ke-6 : Posisi terkecil = 8

Menjadi

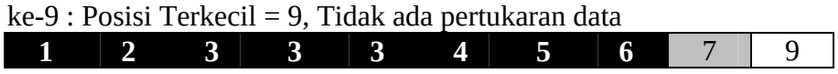
Langkah ke-7 : Posisi terkecil = 10

Menjadi

Langkah ke-8 : Posisi Terkecil = 10

Menjadi

Langkah ke-9 : Posisi Terkecil = 9, Tidak ada pertukaran data



Menjadi

Algoritma di atas adalah :

a. lakukan perulangan ***i*** untuk mewakili langkah dari 1 sampai N-1,dimana N adalah banyak data. Lakukan proses b.
b. Lakukan pencarian index terkecil dalam range data ke-i sampai data ke-N, dan simpan index posisinya dalam variabel idxterkecil. Lanjutkan ke proses c.

Jika data ke-i lebih besar dari data ke-indexterkecil, maka tukarkan data ke-i dengan data ke-indexterkecil.

3. Insertion Sort
4. Radix Sort
5. Merge Sort
6. Quick Sort

5. Penghancuran

Proses mengembalikan data array ke nilai awal.