MODUL III
Struktur Data

Judul	ARRAY dan OPERASI DASAR ARRAY	
Penyusun	**Distribusi**	**Perkuliahan**
Nixon Erzed	Teknik Informatika Universitas Esa Unggul	Pertemuan – III online

Tujuan :

Mahasiswa mengenal struktur dasar array dan memahami representasi data dengan array dan operasi array

Materi :

- Array dan representasi Kontigu
- Deklarasi Array & Dimensi Array
- Penciptaan dan Penghancuran Array
- Type Struct (Record) dan deklarasi Struct (Record)
- Deklarasi Type dan Function
- Operasi Dasar pada array
  - ✓ Struktur Perulangan
  - ✓ Penyimpanan dan pengambilan nilai
  - ✓ Penelusuran array
- Keunggulan dan Kelemahan Array

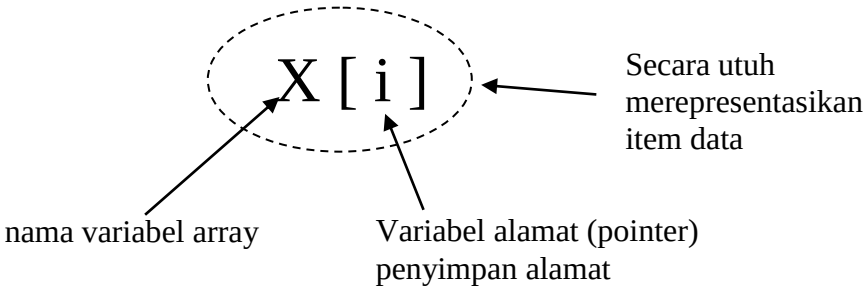
ARRAY (LARIK) DAN REPRESENTASI KONTIGUE

Definisi :

Array adalah suatu type data terstruktur yang terdapat dalam memori yang terdiri dari sejumlah elemen (tempat) yang mempunyai type data yang sama dan merupakan gabungan dari beberapa variabel sejenis serta memiliki jumlah komponen yang banyaknya tetap.

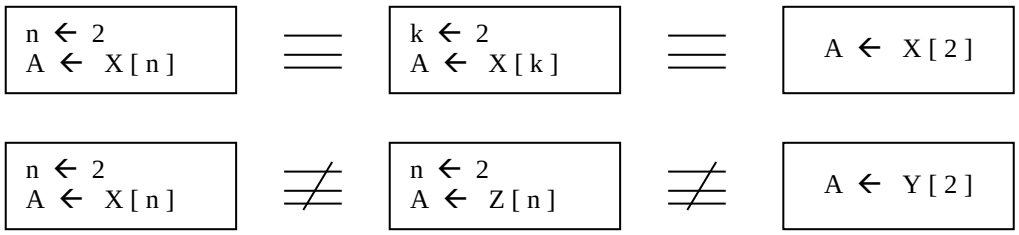
Notasi :

Suatu array yang dideklarasi dengan sebuah variable ***X*** dan penunjuk elemen ***i*** dituliskan sebagai :



Variabel alamat (indeks) merupakan representasi nilai alamat/lokasi ruang memori sehingga tidak bersifat tetap, artinya dapat diganti dengan variabel lain atau dengan sebuah nilai (integer), selama mengacu pada nilai alamat yang dimaksud. Sedangkan variabel array merupakan representasi bagi lokasi fisik memori tempat disimpannya data, sehingga bersifat tetap (tidak dapat digantikan oleh variabel lain atau suatu nilai).

Ilustrasi



Representasi Kontigu

Elemen-elemen dari array umumnya tersusun secara kontigu (berurutan dan sequensial) dalam memori komputer.

Penyimpanan array X dengan ***n*** elemen diilustrasikan sebagai berikut :

X	X [1]	X [2]	X [3]	. . .	X [n-1]	X [n]
	@1	@2	@3	. . .	@ n-1	@ n

Implementasi penyimpanan array di memori komputer umumnya menempati alamat-alamat yang berturutan.

Pemetaannya dapat dapat digambarkan sebagai berikut :

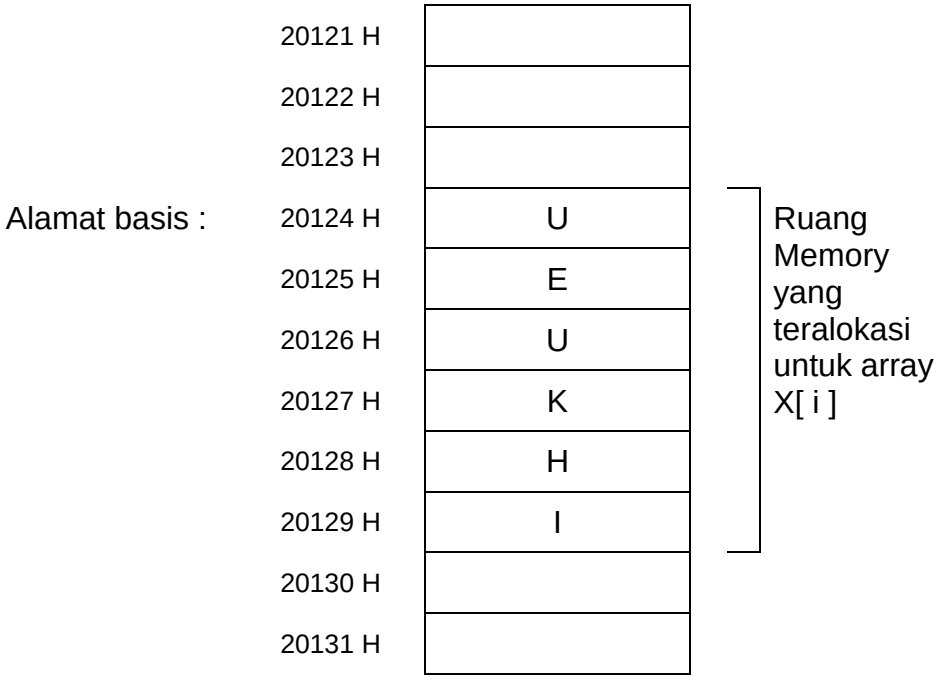
	Alamat	per unit ruang	
Alamat basis :	@ 1	X [1]	
	@ 2	X [2]	
	@ 3	X [3]	
	@ 4	X [4]	
		…	**n**
		…	
	@ j	X [j]	
		…	
		…	
	@ n-1	X [n-1]	
	@ n	X [n]	

Pemetaan yang diilustrasikan diatas adalah representasi logik struktur array didalam memory. Implementasi sesungguhnya diatur oleh sistem operasi, walaupun mungkin secara fisik elemen-elemen array tidak berada pada lokasi yang berturutan, tetapi secara logik berada dalam posisi yang berturutan.

Contoh lain :

Untuk array X[i] bertype char yang secara tepat akan menempati 1 alamat memory per elemen array X[i]. Misalkan X[i] mendapat alamat basis 20124 H, maka pemetaanya dapat digambarkan sebagai berikut.

X [i] :	U	E	U	K	H	I



DEKLARASI ARRAY

Deklarasi array meliputi tiga hal :

- Penamaan
- Pendefinisan type
- Pendefinisian range (jangkauan)

Cara pendeklerasian dalam program tergantung pada bahasa pemrograman yang digunakan.

Penomoran atau pengalamatan elemen array pada dasarnya selalu dimulai dengan 0..n, namun dalam bahasa pemrograman diimplementasikan dalam pengembangan yang berbeda-beda. Pada Pascal misalnya, penomoran dapat disesuaikan dengan logika struktur data, jadi dibolehkan mendeklarasikan alamat dari m..n.

Misalnya -5..0 atau 0..10 atau 1..100 atau 10..30

Ukuran range yang dideklarasikan disesuaikan dengan perkiraan maksimun jumlah data yang akan dikelola array. Artinya pendeklarasian array hanya dapat dilakukan jika jumlah maksimum data diketahui. Jika dalam pengoperasian program, ternyata jumlah data melebihi deklarasi range, maka program harus dibongkar dan deklarasi array diperbaiki langsung pada program sumber.

Contoh :

Pseudo code : *Deklarasi* *X adalah array bertype integer dengan jumlah elemen 20*
Pascal : *Var* *X : array [1..20] of integer*
C : *Int X [20]*

Untuk bahasa C penomoran/pengindeksan selalu dimulai dari 0. Untuk contoh deklarasi diatas, X [20], berarti dilakukan alokasi statis untuk 21 unit ruang memori.

PENCIPTAAN DAN PENGHANCURAN ARRAY

Secara umum, array dikenal sebagai variabel statis. Ketika program dieksekusi, untuk setiap variabel array yang dideklarasikan akan dialokasikan ruang memory secara tetap. Besarnya ruang memory yang dialokasikan sesuai dengan ukuran elemen array dan range yang dideklarasikan.

Proses pengalokasian tersebut disebut Operasi Penciptaan atau Inisialisasi.

Berikut ini adalah prosedur untuk memodelkan penciptaan array.

```
Procedure initTabInt (var T : tabInt; size : integer);
  Begin
        For  i := idxMin   to  idxMax  do  T.Telemen [ i ]  := 0;
        T.Size :=size;
        T.ErrorCode  :=  NoError;
  End;
```

Array yang diciptakan secara tetap akan menempati ruang memory, selama program yang mendeklarasikannya masih *run.* Dalam keadaan ruang memory yang terbatas, akan lebih baik jika variabel-variabel berukuran besar (seperti array) yang sudah tidak terpakai lagi di*dealokasi* dari ruang memory yang dikuasainya, dan menyerahkan ke *manajemen memory* agar dapat dipergunakan keperluan lain.

Proses *dealokasi* tersebut disebut Operasi Penghancuran

```
Procedure destroyTabel (var T : tabel);
  Begin
        T.Size := 0 ;
        T.ErrorCode  :=  NoError;
  End;
```

DIMENSI ARRAY

Elemen-elemen dari array tersusun secara sequensial (kontigu) dalam memory komputer. Array dapat berupa satu dimensi, dua dimensi, tiga dimensi, atau banyak dimensi.

Array Satu Dimensi

Merupakan kumpulan elemen-elemen identik yang tersusun dalam satu baris. Type data setiap elemen adalah sama, sedangkan isinya boleh berbeda.

21	32	42	22	31	8	90	20	15
@1	@2	@3	@ 4	@ 5	@ 6	@ 7	@ 8	@ 9

Pada gambar diatas terlihat bahwa data-data pada setiap elemen array bertype sama yaitu : *integer*

Array Dua Dimensi

Merupakan kumpulan elemen-elemen identik yang tersusun dalam baris dan kolom, berbentuk tabel. Setiap elemen array diidentifiaksi berdasarkan nomor baris dan nomor kolomnya. Jumlah baris dan kolom tidak mesti sama.

Misalkan array T [3,4]

	1	2	3	4
1	60	70	65	14
2	80	90	80	12
3	55	60	67	11

Misalkan array T [3,4] mendapat alokasi memory dengan alamat basis 20122 H.

	20121 H		
Alamat basis :	20122 H	60	T[1,1]
	20123 H	70	T[1,2]
	20124 H	65	T[1,3]
	20125 H	14	T[1,4]
	20126 H	80	T[2,1]
	20127 H	90	T[2,2]
	20128 H	80	T[2,3]
	20129 H	12	T[2,4]
	20130 H	55	T[3,1]
	20131 H	60	T[3,2]
	20132 H	67	T[3,3]
	20133 H	11	T[3,4]
	20134 H		

Ruang Memory yang teralokasi untuk array T[i , j]

Implementasi logika dari array dua dimensi, umumnya elemen pada baris yang sama merepresentasikan satu sub-kelompok data, sedangkan jumlah baris merepresentasikan kuantitas sub-kelompok data.

Misalnya :
Menyimpan data nilai Tugas, UTS, UAS, dan jumlah kehadiran mahasiswa pada sebuah array :

	1	2	3	4
@1	60	70	65	14
@2	80	90	80	12
@3	55	60	67	11
@ n	56	76	65	12

Secara logika perancangannya misalkan ditetapkan bahwa data pada baris @1 adalah nilai Tugas, UTS, UAS, dan Jumlah Kehadiran dari Mahasiswa –1, begitu seterusnya dengan untuk baris @ 2 s/d @ n

Array Tiga Dimensi dan Berdimensi Banyak

Mirip seperti array dua dimensi, pada array tiga dimensi setiap elemen array akan diidentifikasi dengan nomor Baris, Kolom dan Kedalaman

MANIPULASI ARRAY

Sebelum membahas manipulasi array, beberapa hal yang perlu diingat adalah :

1. Array adalah sekumpulan data bertype sama diberi nama yang sama, atau dengan definisi lain, array adalah identitas sekumpulan data dengan menggunakan nama yang sama
2. Untuk mengidentifikasi item data digunakan indeks
3. Indeks adalah angka berurutan yang merepresentasikan alamat ruang memory yang berurutan pula (kontigue)
4. Alamat yang berurutan tersebut memungkinkan melakukan pengolahan dengan suatu perulangan (looping) melalui manipulasi indeks

Struktur Perulangan (Looping)

Struktur perulangan diperlukan untuk mengunjungi seluruh elemen array secara sistematik . Pola perulangan (looping) yang dilakukan adalah dengan inisialisasi indeks awal dan kondisi “hingga terkunjungi elemen ke-n” atau ditemukan keadaan tertentu.

Struktur looping memiliki 3 elemen, yaitu

Inisialisasi : persiapan sebelum pemrosesan dimulai

Misalnya :
- pemberian nilai awal bagi variabel pointer,
- pemberian nilai awal bagi variabel “operasi”

Proses : operasi terhadap elemen yang dikunjungi

Misalnya :
Pengambilan nilai elemen array yang sedang dikunjungi. Proses yang sama akan dilakukan untuk setiap elemen yang dikunjungi, diimplementasikan dengan sebuah loop.

Pada looping untuk keperluan tertentu, proses boleh kosong.

Terminasi : mengakhiri permosesan seluruh elemen.

Terdapat 2 statement untuk melakukan terminasi, yaitu :

- kondisi untuk menentukan apakah loop dilanjutkan atau dihentikan
- instruksi/statemen yang mengubah nilai kebenaran kondisi.

Misalnya,

Penelusuran elemen array X [i] dari elemen ke-1, hingga elemen terakhir. Jumlah elemennya adalah n = 20

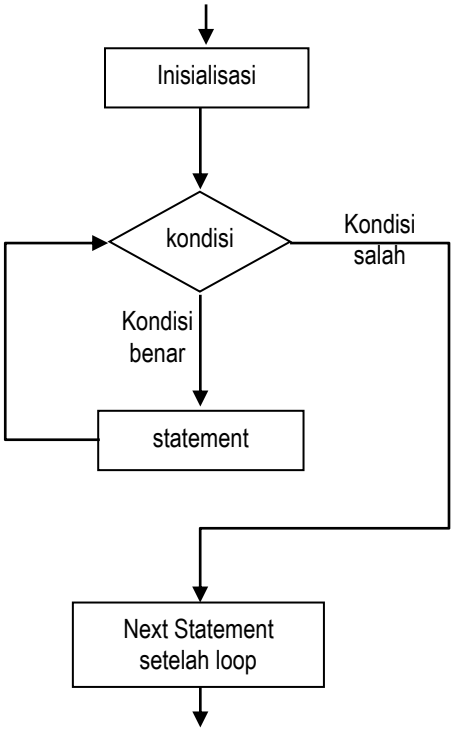
- pemeriksaan kondisi : i <= n, atau i <=20
- instruksi pengubah nilai i dengan cara menambahkan i+1 atau increment
- Increment i, akan mengubah nilai kebenaran pemeriksaan kondisi i <= 20

Jenis-jenis struktur looping

1. While – do
2. Do – While atau Do – Until atau Loop – Until
3. For – next i

Struktur perulangan While-Do

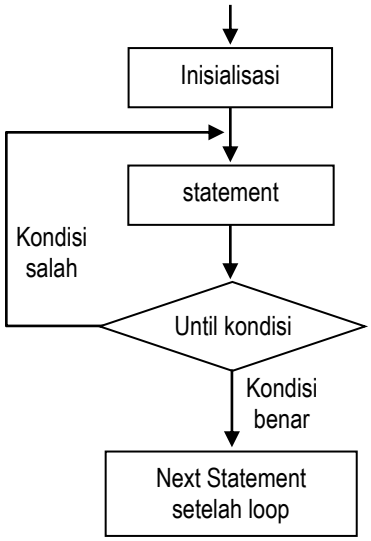
Perulangan menggunakan statemen While-Do yang mempunyai struktur dalam bentuk diagram sebagai berikut :



Statemen While-Do digunakan untuk melakukan proses perulangan suatu statemen atau blok statemen terus menerus selama kondisi ungkapan logika pada While terpenuhi atau bernilai logika benar.

Struktur perulangan Repeat – Until

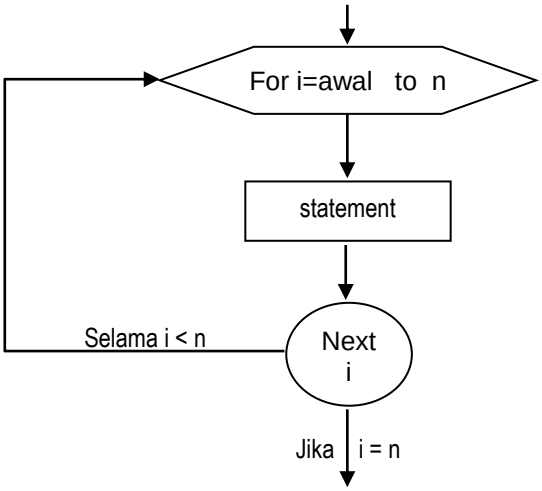
Perulangan menggunakan statemen Repeat - Until yang mempunyai struktur dalam bentuk diagram sebagai berikut :



Statemen Repeat – Until digunakan untuk melakukan proses perulangan suatu statemen atau blok statemen terus menerus selama kondisi ungkapan logika pada Until belum terpenuhi atau bernilai logika salah.

Struktur perulangan For - next i

Perulangan menggunakan statemen *For – next i* yang mempunyai struktur dalam bentuk diagram sebagai berikut :



Statemen *For – next I* digunakan untuk mengulang pernyataan atau satu blok pernyataan berulang kali sejumlah (n) yang ditentukan. Perulangan dengan pernyataan ***For*** dapat berbentuk perulangan positif, perulangan negatif dan perulangan tersarang.

Operasi Penyimpanan dan Pengambilan Nilai

Prinsip dasar yang harus diingat :

1. Elemen terurut, sebagai elemen pertama, ke dua dan seterusnya.

X	X [1]	X [2]	X [3]	. . .	X [n-1]	X [n]
	@1	@2	@3	. . .	@ n-1	@ n

2. Pengaksesan dilakukan berdasarkan indeks/penunjuk posisi

   Misalnya ingin diakses elemen yang ke-3 maka dapat ditulis sebagai X [n] dengan n = 3

Instruksi untuk penyimpanan dan pengambilan nilai elemen pada posisi tertentu di array tergantung bahasa pemrograman. Operasi terhadap elemen array dilakukan dengan pengaksesan langsung. Nilai dimasing masing posisi elemen dapat diambil dan nilai dapat disimpan tanpa melewati posisi-posisi lain.

Contoh

* X [12] <-- 20

  berarti penyimpanan nilai 20 ke posisi 12 dari array X

* C <-- X [12]

  berarti pengambilan nilai elemen posisi ke 12 dari array X dan disimpan di variabel C.

* read (X [12]); *{dalam sintaks Pascal}*

  membaca data dari media masukan (keyboard) dan menyimpannya ke posisi 12 dari array X

* write (X [12]); *{dalam sintaks Pascal}*

  Mengambil nilai elemen posisi ke 12 dari array X dan menuliskannya ke media keluaran (layar display)

Penelusuran array (Pemrosesan Traversal)

Operasi penelusuran atau traversal adalah operasi yang mengunjungi seluruh elemen array secara sistematik.

Berikut ini adalah beberapa penerapan operasi penelusuran, yaitu :

1. Prosedur untuk mengisi seluruh elemen array.
2. Prosedur untuk menuliskan seluruh nilai elemen array ke layar
3. Prosedur untuk mencari nilai ekstrim elemen array
4. Fungsi untuk menjumlahkan seluruh nilai elemen array
5. Fungsi untuk menghitung rata-rata seluruh elemen array.

Secara lebih detil, berikut ini akan diberikan algoritma-algoritma untuk operasi penelusuran tersebut.

1. Prosedur untuk mengisi seluruh elemen *array.*

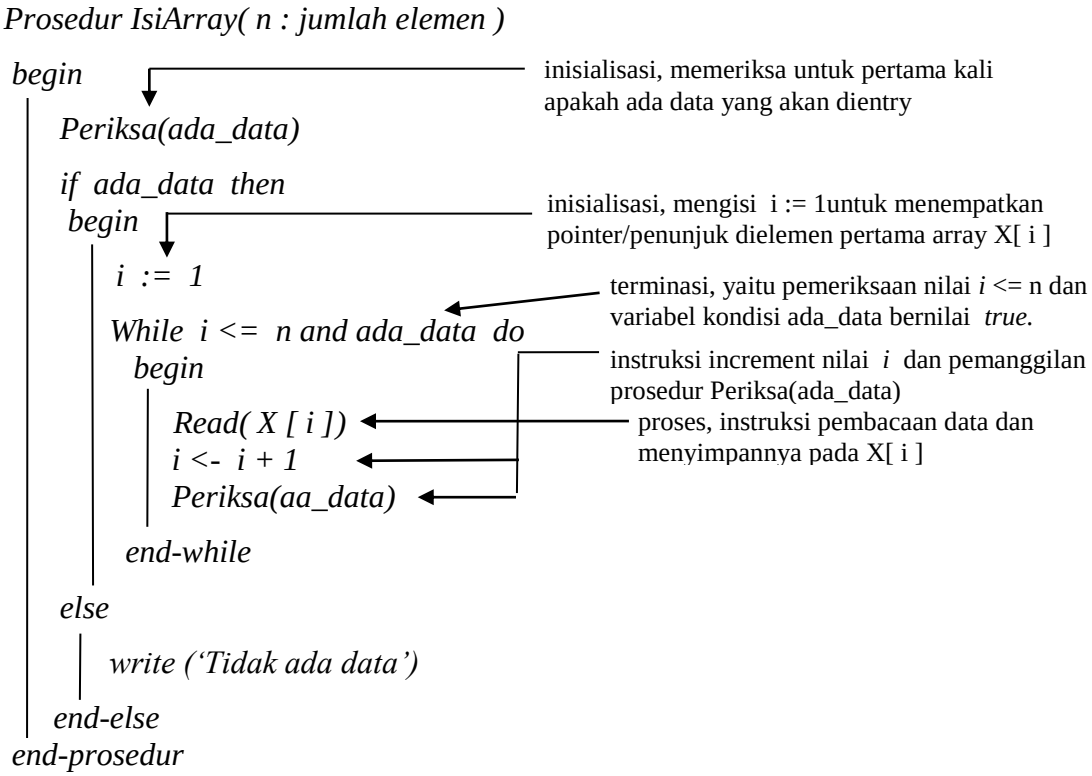
Jika dimiliki suatu array X bertype integer dengan panjang array n, maka

Asumsi 1 : jumlah data/elemen yang akan diisi diketahui

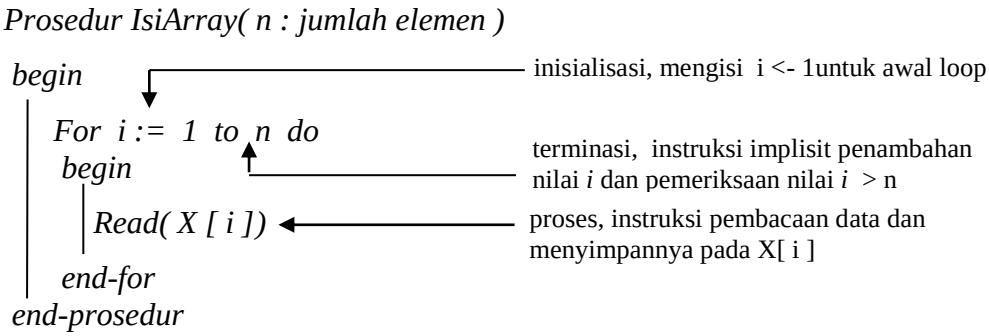
Diimplementasikan dalam struktur *while..do* akan dihasilkan algoritma sbb. :

```
Prosedur IsiArray( n : jumlah elemen )
begin
  i := 1                          <- inisialisasi, mengisi i := 1untuk menempatkan
While i <= n do                      poiter di elemen awal array X[ i ]
 begin                            <- terminasi, yaitu pemeriksaan nilai i <= n
   Read( X [ i ])                    dan instruksi increment nilai i
   i := i + 1                    <- proses, instruksi pembacaan data dan
                                    menyimpannya pada X[ i ]
  end-while
end-prosedur
```

Asumsi 2 : Jumlah data tidak diketahui



Pengisian array dapat juga dilakukan dengan menggunaka struktur For do next



2. Prosedur untuk menuliskan seluruh nilai elemen array ke layar

Sebagaimana sudah disebutikan sebelumnya, secara struktur porses traversal relatif sama, hnya perlu penyesuaian pada bagian proses.

```
Prosedur KeluarkanArray( n : jumlah elemen )
begin                                   <- inisialisasi, mengisi  i <- 1untuk awal loop
  For  i :=  1  to  n  do
   begin                                <- terminasi,  instruksi implisit penambahan
                                           nilai i dan pemeriksaan nilai i  > n
     write( X [ i ])                    <- proses, instruksi menuliskan data yang terdapat
                                           pada elemen ke –i dari array X ke layar
   end-for
end-prosedur
```

Pada prosedur keluarkan array tersebut, hanya instruksi pada body loop yang disesuaikan, yaitu *read (X [i])* menjadi *write (X[i])*, sementara instruksi yang lain sama.

Jika diimplementasikan dalam struktur *while..do* akan dihasilkan algoritma sbb. :

```
Prosedur KeluarkanArray( n : jumlah elemen )
begin                                   <- inisialisasi, mengisi  i := 1untuk menempatkan
                                           poiter di elemen awal array X[ i ]
  i  :=  1
  While  i <=  n  do                    <- terminasi, yaitu pemeriksaan nilai i <= n
   begin                                   dan  instruksi increment nilai  i
     Write( X [ i ])                    <- proses, instruksi menuliskan data yang terdapat
     i :=  i + 1                           pada elemen ke –i dari array X ke layar
   end-while
end-prosedur
```

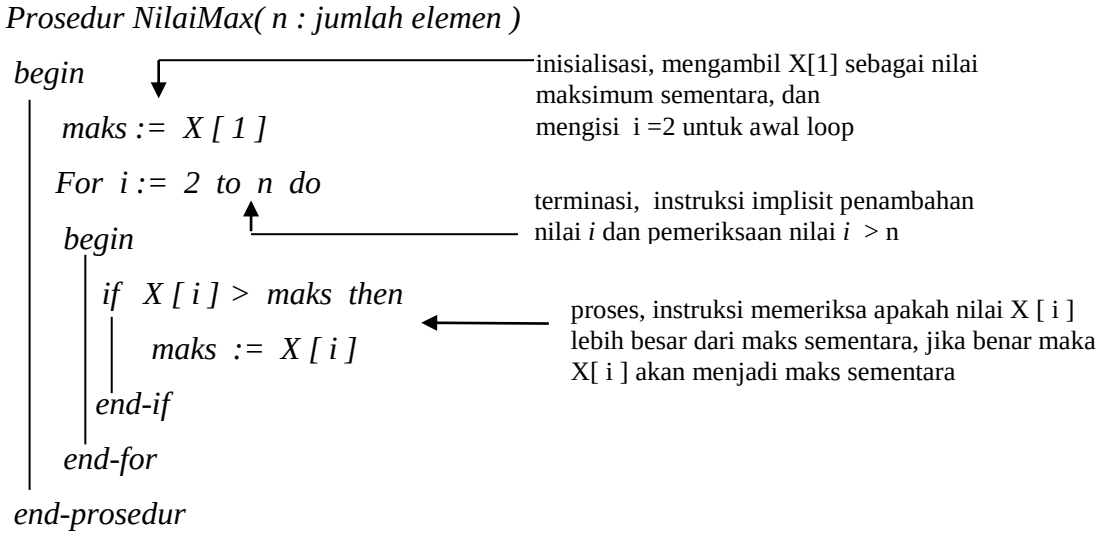
3. Jumlah seluruh nilai elemen

```
Prosedur Jumlah( n : jumlah elemen )
begin                                   <- inisialisasi, mengisi  jumlah = 0 untuk insialisasi
                                           awal pada operasi penjumlahan
  jumlah :=  0
  For  i :=  1  to  n  do
   begin
     Jumlah  :=  Jumlah  + X [ i ]      <- proses, instruksi menjumlahkan setiap elemen X[ i ]
                                           ke variabel jumlah.
   end-for
end-prosedur
```

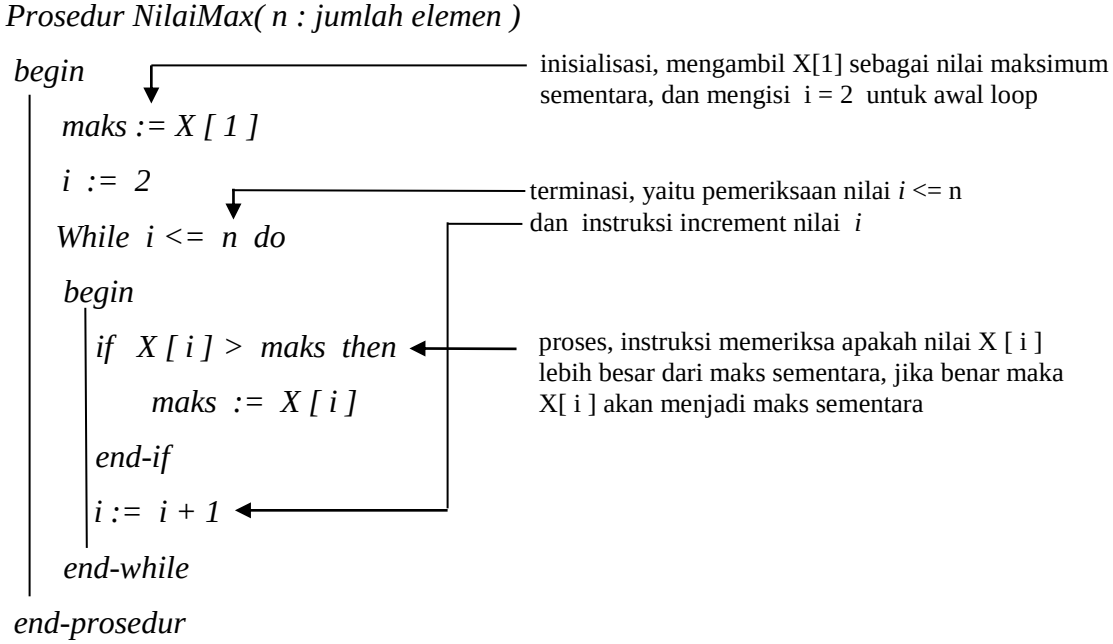
4. Pencarian Nilai Ekstrim

Pada pencarian nilai ekstrim, terdapat penyesuain pada bagian inisialisasi.

Misalnya prosedur untuk mencari nilai maksimum, yaitu sbb. :



Jika diimplementasikan dalam struktur *while..do* akan dihasilkan algoritma sbb. :



5. Menghitung Rata-rata

```
Prosedur Hitung_Rata_rata( n : jumlah elemen )
begin
    jumlah :=  0
    i := 1
    While i <= n  do
    begin
        Jumlah  :=  Jumlah  + X [ i ]
        i := i + 1
    end-while
    Rata_rata := jumlah / n
end-prosedur
```

inisialisasi, mengisi jumlah = 0 untuk insialisasi awal pada operasi penjumlahan

proses, instruksi menjumlahkan setiap elemen X[i] ke variabel jumlah.

Type Record atau Struktur

Dalam merepresentasikan sebuah objek data/masalah, kadangkala membutuhkan sekumpulan item data yang berbeda

Misalnya objek manusia
→ dalam persoalan data alamat
→ Minimal diperlukan 2 item data berbeda, yaitu Nama, Alamat

Dibutuhkan sebuah cara untuk merepresentasikannya dalam program, yaitu dengan struktur record

→ record atau structure

Sebagai contoh : data alamat

```
Type
 data_alamat = Record  of
    |  Nama   : string[30]
    |  Alamat : string[50]
    |  Kota    : string [20]
    |  Kd_pos : string[5]
 End

 data_telp_teman = Record  of
    |  Nama   : string[30]
    |  Notel   : string[12]
 end
```

Type record yang sudah didefinisikan akan menjadi atribut dalam suatu tabel.

Contoh :

Suatu daftar NIM, nama mahasiswa, data kontak/hp dan IPK

	NIM	Nama	No. HP	IPK
#1	20170801021	Parady	0812312122	3,6
#2	20170801031	Edri	0812442239	3,2
#3	20170801101	Zony	0817121322	3,7
#4	20170801104	Betty	0823378927	3,4
#5	20170801122	Rera	0878821121	3,6
#6	2017080123	Popy	0856312189	3,5
#7				

Untuk data pada tabel diatas, dapat dideklarasikan type record sebagai berikut :

```
Type
    D_mahasiswa  = record  of
      NIM      : string[12]
      Nama   : string[20]
      HP       : string [12]
      IPK      : real
    End
```

Untuk mengolah data tsb diperlukan array bertype rakitan

Array dengan Type Bentukan

Untuk penggunaan yang lebih kompleks array dapat juga didasarkan pada type bentukkan. Sehingga keidentikan type tidak dapat dilihat pada level item data, dengan kata lain item data dapat bertype yang berbeda. Lihat kembali ilustrasi yang diberikan pada halaman.

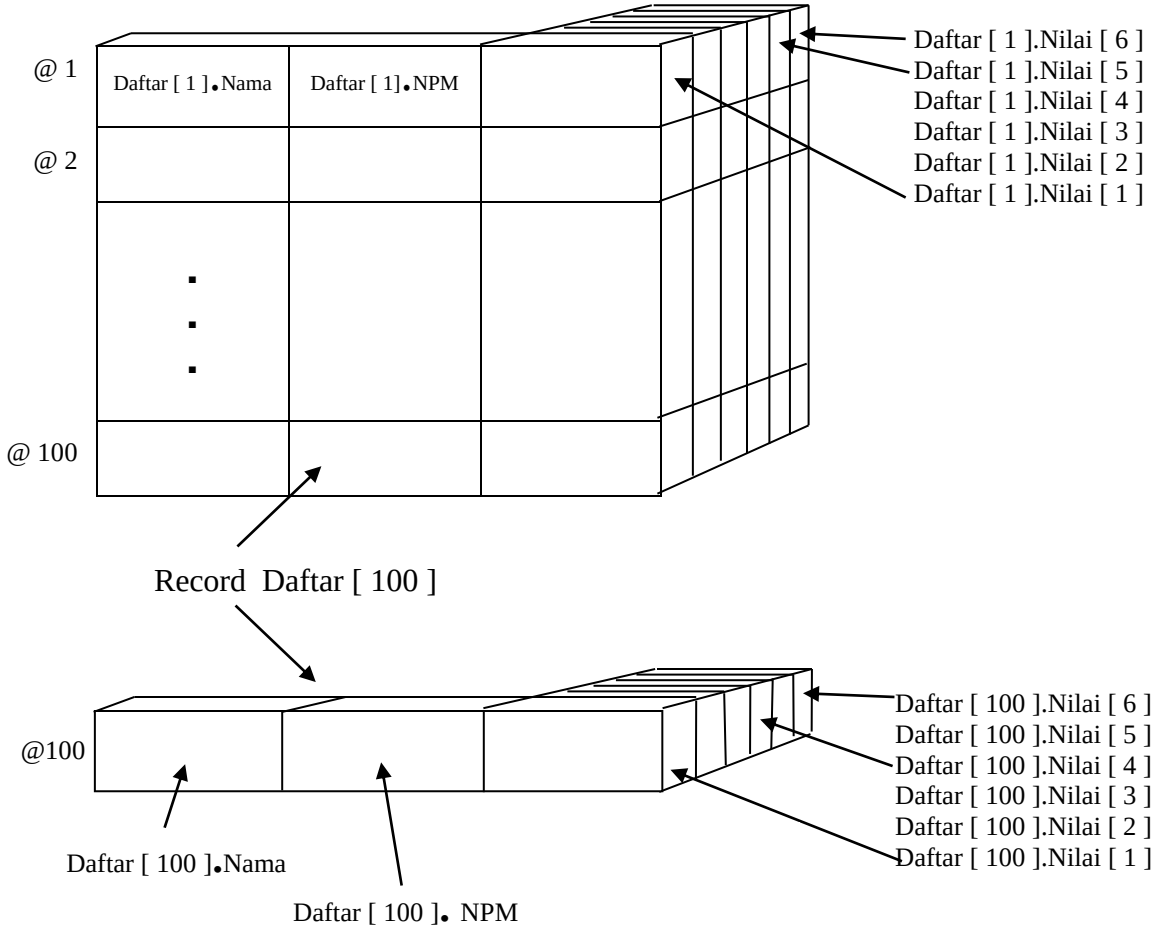
Pada beberapa bahasa pemrograman dapat dilakukan pendeklerasian array dengan type bentukan. Misalnya array bertype record, sehingga setiap elemen array adalah sebuah record data yang dibangun oleh beberapa field. Dalam hal ini keseragaman adalah pada record data, atau dengan kata lain elemen array adalah record data (bukan field). Dengan type bentukan, array yang dihasilkan dapat menjadi sangat kompleks.

Contoh :

```
Type
  RecMhs  =  record
                Nama  :  string [25];
               NPM  :  string [ 9 ];
               Nilai   :  array [ 1.. 6 ] of  real;
          End;

Var
  Daftar  :  array[1..100] of RecMhs
```

Deklarasi data tersebut dapat dilustrasikan sebagai berikut :



DEKLARASI TYPE DAN FUNCTION

Untuk type-type dasar sudah dideklarasikan oleh bahasa pemrograman, misalnya :

Type integer

Type Integer berlaku bagi objek data bilangan bulat → 0, ±1, ±2, dst

*Terhadap objek data ini hanya diizikan operasi +, -, *, DIV, dan MOD*

Operasi matematis lainnya akan ditolak. Terdapat pengecualian jika type tersebut hanya sebagai operand (mengambil nilai / get value / read only), misalnya :

A & B adalah integer,

Operasi bagi (/) diizinkan selama hasilnya disimpan ke variabel lain yang bertype real → C ⇐ A / B dimana C adalah Real

Contoh lain adalah type String.

Secara formal type integer dan type string disajikan dengan TDA sebagai berikut :

- TDA Type Integer

  Domain D = { *0, ±1, ±2, ±3, … }*

  Function F = {+, -, *, mod, div}

  Aksioma A = { }

- TDA Type String

  Domain D = { s | s [x] = string ASCII, 1<x< 256 & x∈ Z }

  Function F = { penggalan = f(s[x],n,m) }

  Aksioma A = { }

Begitu juga untuk beberapa type rakitan, sudah banyak dideklarasikan oleh bahasa pemrograman. Misalnya type *date*/tanggal dan type *time*/waktu saat ini sudah built-in dalam bahasa pemrograman.

Yang harus diingat ketika sebuah type data didefinisikan, maka operasi-operasi yang melekat pada type tersebut juga harus didefinisikan. Berikut ini akan diberikan beberapa contoh Deklarasi Type dan Function.

1. Type Jam

Data jam merupakan kombinasi hirarki 3 komponen yang basisnya adalah bilangan bulat dengan *range* tertentu, yaitu Jam, Menit, Detik. Domain untuk masing-masing komponen data memiliki range berbeda. Terhadap Jam, operasi yang umum dilakukan adalah : menghitung selisih 2 waktu (P/lama), menghitung nilai detik (TS) dari data jam (data skalar), mengambil komponen jam (Hr) dari data jam, mengambil komponen menit (Mn) dan mengambil komponen detik (Sc). Operasi selisih 2 waktu hanya untuk waktu pada hari/tanggal yang sama.

Agar lebih jelas, berikut ini didefinisikan dengan TDA Type Jam, yaitu:

Domain D = { j,m,d | $0 \le j \le 23$,
$0 \le m \le 59$,
$0 \le d \le 59$,
$j,m,d \in Z$ }

Function F = { Periode = f1(t_0, t_1),
TotalDetik = f2(t),
DJam = f3 (t),
DMenit = f4 (t),
DDtk = f5 (t) }

Aksioma A = { perhitungan lama dikoreksi jika tangga berbeda }

Deklarasi Type Jam

a. Type

```
jm   = [0…23]
ms   = [0…59]
wkt  = record of
       | j   : jm
       | m   : ms
       | d   : ms
  end-record
```

b. Function

Function TotalDetik (t : wkt) : Integer

```
Begin
   |   TotalDetik ⇐ (t.j * 3600) + (t.m * 60) + t.d
End TotalDetik
```

```
Function Periode(w0, w1 : wkt) : wkt
Var  dt0, dt1, dt : integer
Begin
    dt0 ⇐ w0.d + w0.m*60 + w0.j*3600
    dt1 ⇐ w1.d + w1.m*60 + w1.j*3600
    If  w1.j > w0.j
        Then  dt  ⇐  dt1 – dt0
        Else    dt  ⇐  dt0 – dt1
    End-if
    Periode.j ⇐  dt div 3600
    Periode.m ⇐ (dt mod 3600) div 60
    Periode.d ⇐ (dt mod 3600) mod 60
End-waktu
```

Untuk Function lainnya : DJam = f3 (t), DMenit = f4 (t), dan DDtk = f5 (t), silahkan deklarasikan fuction sebagai latihan.

2. Type Tanggal

Sama seperti data jam, data tanggal juga merupakan kombinasi hirarki 3 komponen yang basisnya adalah bilangan bulat dengan *range* tertentu, yaitu tanggal, bulan, dan tahun. Domain untuk masing-masing komponen data memiliki range berbeda. Terhadap Tanggal, operasi yang umum dilakukan adalah : menghitung selisih 2 tanggal, menghitung jumlah hari(Total_hari) dari data jam (data skalar), mengambil komponen jam (Hr) dari data jam, mengambil komponen menit (Mn) dan mengambil komponen detik (Sc). Operasi selisih 2 waktu hanya untuk waktu pada hari/tanggal yang sama.

Mendefinisikan Type Tanggal

Contoh data → 7 / 3 /2007

Identifikasi objek dan masalah-masalahnya

1. Objek

Tanggal → 1, 2, 3, 4.. 31

Bulan → 1, 2, .. 12

tahun → integer positif

2. Operasi yang legal :

menghitung hari → umur

nama bulan

3. aksiom

tahun kabisat, max hari perbulan berbeda-beda

TDA Tanggal

D : t = {d| $1 \leq d \leq 31$, $d \in Z$} ∪

{m| $1 \leq m \leq 12$, $m \in Z$} ∪

{y| $y \geq 0$, $y \in Z$}

F : fungsi = { hari = f(t1, t2), nabul = g(t.m) }

A : batas = **{** Max(t.m =1,3,5,7,8,10,12) = 31;

Max(t.m =4,6,9,11) = 30;

Max(t.m = 2) = 28;

t.y mod 4 = 0 ⇒ max(t.m = 2) = 29 **}**

Deklarasi Type

```
Type    date = [1…31]
        month = [1…12]
        tgl  = record of
                        d : date
                        m: month
                        y : integer
               end-record
```

Deklarasi Fungsi

Function Hari(t1, t2 : tgl) : integer

Var h1, h2, h3, x : integer

Begin

```
Case t1.m of
        1    :  h1 ⇐  365 – t1.d
        2    :  h1 ⇐  334 – t1.d
         :
        12:  h1 ⇐  31 – t1.d
end-case

IF   (t1.y mod 4 = 0) and ( t1.m ≤ 2)
  then
        h1 ⇐ h1+1
end-IF

x   = (t2.y – t1.y – 1)
h2 = x * 365 + x div 4

IF  [(x mod 4 = 1)  and  (t1.y+1 mod 4 = 0)]  or
    [(x mod 4 = 2) and ((t1.y+1 mod 4 = 0) or (t1.y+2 mod 4 = 0))]  or
    [(x mod 4 = 3) and ((t1.y+1 mod 4 = 0) or (t1.y+2 mod 4 =0) or (t1.y+3 mod 4 =0]
  then
        h2 ⇐ h2+1
end-IF

Case t2.m of
        1   : h3 ⇐  t2.d
        2   : h3 ⇐  t2.d + 31
        3   : h3 ⇐  t2.d + 59
         :
        12: h3 ⇐  t2.d +334
end-case

IF  (t2.y mod 4 = 0) and (t2.m ≥ 3)
   then
        h3 ⇐  h3+1
end-IF

Hari ⇐  h1 + h2 + h3
```

End-function

Function Hari-koreksi(t1, t2 : tgl) : integer
Var h1, h2, h3, x : integer

{memperhentikan kemungkinan menghitung hari pada tahun yang sama }

Begin

IF t2.y > t1.y
Then
Case t1.m of
1 : h1 ⇐ 365 – t1.d
2 : h1 ⇐ 334 – t1.d
:
12: h1 ⇐ 31 – t1.d
end-case

IF (t1.y mod 4 = 0) and (t1.m ≤ 2) then h1 ⇐ h1+1 end-IF

x = (t2.y – t1.y – 1)
h2 = x * 365 + x div 4

IF [(x mod 4 = 1) and (t1.y+1 mod 4 = 0)] or
[(x mod 4 = 2) and ((t1.y+1 mod 4 = 0) or (t1.y+2 mod 4 = 0))] or
[(x mod 4 = 3) and ((t1.y+1 mod 4 = 0) or (t1.y+2 mod 4 =0) or (t1.y+3 mod 4=0]
then h2 ⇐ h2+1
end-IF

Case t2.m of
1 : h3 ⇐ t2.d
2 : h3 ⇐ t2.d + 31
:
12: h3 ⇐ t2.d +334
end-case

IF (t2.y mod 4 = 0) and (t2.m ≥ 3) then h3 ⇐ h3+1 end-IF

Hari ⇐ h1 + h2 + h3

else
Case t1.m of
1 : h1 ⇐ 365 – t1.d
2 : h1 ⇐ 334 – t1.d
:
12: h1 ⇐ 31 – t1.d
end-case

IF (t1.y mod 4 = 0) and (t1.m ≤ 2) then h1 ⇐ h1+1 end-IF

Case t2.m of
1 : h3 ⇐ t2.d
2 : h3 ⇐ t2.d + 31
:
12: h3 ⇐ t2.d +334
end-case

IF (t2.y mod 4 = 0) and (t2.m ≥ 3) then h3 ⇐ h3+1 end-IF

IF t1.y mod 4 = 0
then Hari ⇐ h1 + h3 -366
else Hari ⇐ h1 + h3 -365
end-IF

end-IF
End-function

III. Deklarasi Struktur Data (REVIEW)

Kegunaan spesifikasi struktur data yang digambarkan dengan TDA adalah :

1. Untuk menjamin dan membuktikan kebenaran program yang menggunakan struktur data.

2. Untuk menjamin dan membuktikan kebenaran implementasi struktur data.

→ dengan mengetahui objek data, fungsi, aksioma

Pemrograman dapat diarahkan dalam batasan abstraksi data tersebut saja

Tujuan pembentukan struktur data adalah untuk mengkapsulkan informasi, yaitu :

1. Perubahan implementasi struktur data tidak mengubah program yang menggunakan struktur data bila interface pada struktur data tersebut dirancang secara luwes sehingga data independen terhadap program (prinsip independensi).
2. Dengan pembakuan interface, pemakaian dan pembuatan struktur data dapat dilakukan secara terpisah.
3. Struktur data adalah basis pemrograman modular yang dapat dijadikan dasar pembentukan team programmer atau pembagian kerja.

PEMBUATAN STRUKTUR DATA

Terdapat tiga tahap pembuatan struktur data :

1. Tahap pertama : spesifikasi
2. Tahap ke dua : implementasi
3. Tahap ke tiga : pemrograman

} **SPESIFIKASI, DEKLARASI STRUKTUR DATA DAN DEKLARASI FUNGSI**

Tahap Pertama : Spesifikasi

Pendeskripsian atau spesifikasi struktur data yang merupakan batasan-batasan struktur data.
Pendefinisian melibatkan logik sehingga sering digunakan ketetapan/ teorema matematika untuk menyatakan sifat-sifat struktur data yang dikehendaki.

Spesifikasi dapat disajikan dangan dua cara :

1. Spesifikasi secara formal
   Diwujudkan dalam bentuk Type Data Abstrak
2. Spesifikasi secara non formal

   Diwujudkan dengan membuat deskripsi dalam bahasa alami.

Tahap ke dua : Implementasi

Implementasi berisi deklarasi struktur penyimpanan item-item data (nama variabel dan typenya), serta algoritma-algoritma (algoritma fungsi) untuk implmentasi operasi-operasi yang menjadi karakteristik struktur data.

```
Contoh : objek jam pada kasus parkir kendaaraan
Objek jam :   hh:mm:ss
   Type       h = [0 ... 23]
              m = [0…59]
              s = [0…59]
              TJam = record
                       jam  :  h
                       mnt  : m
                       dtk   :  s
                     end-Tjam
   var
        Jmasuk, Jkeluar : Tjam
```

Function lamaparkir (jMasuk, jkeluar : tJam) : Tjam

```
{asumsi bukan parkir menginap}
Var  dtkmasuk, dtkkeluar, dtklama : integer
Begin
              Dtkmasuk <=  jmasuk.jam*3600 + jmasuk.mnt*60 + jmasuk.dtk
              Dtkkeluar <=  jkeluar.jam*3600 + jkeluar.mnt*60 + jkeluar.dtk
              Dtklama <= dtkkeluar – dtkmasuk

              Lamaparkir.jam <= dtklama div 3600
              Lamaparkir.mnt <= (dtklama mod 3600) div 60
              Lamaparkir.dtk <= (dtklama mod 3600) mod 60
End-function
```

Implementasi objek nopol kendaraan adalah :

```
Var   nopol : string[9]
```

Mengacu pada masalah yang harus diselesaikan → tidak memerlukan deklarasi fungsi

Tahap ke tiga : Pemrograman

Pemrograman struktur data adalah penterjemaahan menjadi pernyataan dalam bahasa pemrograman tertentu.

Terdiri dari proses :

1. Deklarasi yang mendefinisikan objek-objek data dan hubungannya
2. Pembuatan prosedur atau rutin untuk operasi-operasi dasar

Perancang harus memilih tipe-tipe data yang telah ada untuk merepresentasikan struktur data.

Struktur data dibangun menggunakan fasilitas pembetukan struktur data yang disediakan bahasa pemrograman.

KEUNGGULAN DAN KELEMAHAN ARRAY

Keunggulan :

1. Sangat bagus untuk random access (pengaksesan secara acak). Sembarang elemen di array dapat diacu secara langsung tanpa melalui elemen-elemen lain.
2. Pengaksesan elemen tetangga (predesesor atau suksesor) dapat dilakukan dengan teknik lompatan terbatas (intsruksi mesin) sehingga memiliki kinerja yang efisien.
3. Jika elemen-elemen array adalah nilai independen dan seluruhnya harus tersimpan maka penggunaan ruang memory menjadi sangat efisien

Kelemahan :

Fleksibelitas rendah, sehingga sulit diterapkan diberbagai aplikasi :

a. array mesti bertype homogen
b. penyisipan dan penghapusan elemen array melibatkan operasi yang kompleks
c. kebanyakan bahasa program mengimplementasikan array statik yang sulit diubah ukurannya di waktu eksekusi
d. Jika penambahan dan pengurangan terjadi dalam frekuensi yang tinggi, maka array statis : tidak efisien dalam penggunaan memory, menyiakan banyak waktu komputasi, dan tidak dapat diterapkan dalam suatu aplikasi.