STRUKTUR DATA

BAB I

PENGENALAN STRUKTUR DATA

TOPIK	MINGGU KE
STRUKTUR DATA DAN ALGORITMA	I
TIPE DATA, FUNGSI, PROSEDUR	II
ARRAY	III
MATRIKS (ARRAY MULTIDIMENSI)	IV
RECORD	V
STACK	VI
QUEUE	VII
MID TEST	VIII
POINTER	IX
LINKED LIST	X
DOUBLE LINKED LIST	XI
TREE	XII
SORTING	XIII
SEARCHING	XIV
GRAPH	XV
UAS	XVI

PENILAIAN

Penilaian yang diberikan meliputi :

Komponen	Bobot
Nilai Kehadiran	10 %
Nilai Tugas Individu (Praktek)	20 %
Nilai Tugas Kelompok (Praktek)	10 %
Nilai Ujian Tengah Semester (UTS)	25 %
Nilai Ujian Akhir Semester (UAS	35 %
Total Bobot semua komponen	100 %

Kriteria penilaian yang digunakan adalah:

Angka Mutu	Huruf Mutu	Rentang Ket.Huruf
> 95	A+	4.5
90 - 95	A	4.0
80 - 89	B+	3.5
70 - 79	B	3.0
60 - 69	C	2.5
50 - 59	D	2.0
< 49	E	1.0

TATA TERTIB PERKULIAHAN

1. **Kehadiran Kuliah:**
   Minimal kehadiran mengikuti aturan yang berlaku (75% minimum kehadiran tatap muka di kelas). Bila sakit atau berhalangan hadir kuliah, harus ada penjelasan dan pemberitahuan sebelum kuliah dimulai.
2. **Mahasiswa yang kehadirannya kurang dari 75% tidak akan**
   lulus, mendapat nilai E
3. **Kehadiran penuh akan dicatat, dan akan menjadi pertimbangan pemberian nilai bonus untuk nilai yang diperbatasan.**
4. **Saat perkuliahan berlangsung, mahasiswa diharapkan fokus kepada mata kuliah yang disampaikan dan terlibat aktif jika ada diskusi.**
5. **Bagi yang tidak menghormati waktu perkuliahan dalam kelas dapat meninggalkan ruangan**

Keterlambatan:

- **Keterlambatan hadir saat kuliah mengikuti aturan yang berlaku (max. 15 menit setelah kuliah dimulai),**

- **Seluruh kuis, latihan dan tugas wajib dikerjakan dikumpulkan pada waktunya.**
  Pengumpulan yang terlambat tidak dinilai.

STRULTUR DATA

STRUKTUR DATA

1. Model matematis atau bentuk logik dari suatu organisasi data.
2. Struktur data harus simpel dalam memproses data yang ada di dalamnya.
3. Struktur data menjadi dasar dalam langkah awal perancangan program

PENGERTIAN STRUKTUR DATA

Struktur data adalah cara menyimpan, menyusun dan mengatur data di dalam komputer sehingga data tersebut dapat digunakan secara efisien.

Mengapa data itu disimpan?

- Supaya bisa diakses/diproses di kemudian hari jika diperlukan

Mis. Cara penyusunan buku-buku pada rak buku di perpustakaan sehingga buku dibutuhkan dapat dengan mudah ditemukan

Mengapa dalam penyimpanan data diperlukan sebuah struktur?

- Supaya lebih mudah/efisien dalam pengaksesan/pemrosesan data tersebut

Bahasan Struktur Data

Struktur Data, meliputi :

a. Struktur data dasar/sederhana, yaitu array, record/struct dan himpunan
b. Struktur data lanjut/majemuk, yang terdiri dari :
   - Linier : Stack, Queue, serta List dan Multilist
   - Non Linier : Pohon Biner dan Graph


ALGORITMA

Contoh penerapan Algoritma

botol 1  botol 2 

Menukar isi botol 1 yang berisi air berwarna HIJAU dengan botol 2 yang berisi air berwarna MERAH. Sehingga nantinya botol 1 berisi air berwarna MERAH sedangkan botol 2 berisi air berwarna HIJAU.

Jika Algoritma yang dilakukan dengan cara :

- **tuangkan isi botol 1 ke botol 2, kemudian tuangkan isi botol 2 ke botol 1.**

Cara di atas adalah SALAH karena pada saat isi botol 1 dituangkan ke botol 2 maka air yang ada pada botol 2 akan tercampur dengan air yang ada, sehingga pada saat isi botol 2 dituangkan ke dalam botol 1 maka warnanya sudah tercampur juga.

botol 1 

botol 2 

botol 3 

Algoritma yang tepat adalah :

1. Siapkan sebuah botol lain dalam keadaan kosong botol 3
2. Kemudian isi botol 1 dituangkan kedalam botol 3 sehingga botol 1 dalam keadaan kosong
3. Langkah berikutnya isi botol 2 dituangkan kedalam botol 1 sehingga botol 2 sekarang dalam keadaan kosong.
4. Baru kemudian isi botol 3 dituangkan kedalam botol 2

❖ KRITERIA ALGORITMA YANG BAIK :

Algoritma yang baik jika memiliki kriteria sebagai berikut:

1. **Setiap langkah yang ada pada algritma harus definite (jelas dan pasti)**
2. **Algoritma atau program minimal memiliki sebuah output**
3. **Harus ada stopping criteria yaitu kondisi yang membuat program tersebut berhenti.**
4. **Hasil akhir tidak boleh tergantung kepada siapa yang menjalani algoritma tersebut**
5. **Suatu algoritma tidak boleh berakhir terbuka.**
6. **Algoritma harus cukup umum untuk menangani kep erluan apapun.**

Pengertian Algoritma :

adalah urutan langkah-langkah logis dalam penyelesaian masalah yang disusun secara sistematis.

- ✓ **Ditulis dengan notasi khusus**
- ✓ **notasi mudah dimengerti**
- ✓ **Notasi dapat diterjemahkan menjadi sintaks suatu bahasa pemrograman**

Dalam kamus besar bahasa Indonesia (Balai Pustaka 1988) secara formal mendefinisikan algoritma sebagai berikut:
" Algoritma " adalah urutan logis pengambilan putusan untuk pemecahan masalah."

Ciri Algoritma yang baik menurut " DONALD E. KNUTH "

1. Algoritma harus berhenti setelah melakukan sejumlah langkah terbatas
2. Setiap langkah algoritma harus didefinisikan dengan tepat dan tidak bermakna ganda (ambiguous)
3. Algoritma memiliki nol atau lebih masukan (input)
4. Algoritma memiliki satu atau beberapa keluaran (output)
5. Algoritma harus efektif

Contoh : **Algoritma Mencetak Absensi ..**

1. **Buka Data Absensi**
2. **Tentukan Mata Kuliah**
3. **Tentukan Kelas**
4. **Tentukan Format Absensi (4 / 14 kolom)**
5. **Tentukan banyak pencetakan**
6. **Ambil data mhs ke-1, lalu cetak**
7. **Ulangi langkah ke-6 sampai data habis**

Hubungan Struktur Data & Algoritma

- Program adalah kumpulan instruksi komputer, sedangkan Algoritma adalah metode dan tahapan sistematis dalam program.
  Program ini ditulis dengan menggunakan bahasa pemrograman
  Jadi bisa kita sebut bahwa program adalah suatu implementasi bahasa pemrograman
- Struktur data dan algoritma berhubungan sangat erat pada sebuah program.
- Pemakaian struktur data yang tepat di dalam proses pemrograman menjadikan algoritma lebih jelas dan tepat, sehingga secara keseluruhan program akan lebih efisien, sederhana dan efektif.
- Algoritma yang baik tanpa pemilihan struktur data yang tepat akan membuat program menjadi kurang baik, demikian juga sebaliknya.

PROGRAM = ALGORITMA + STRUKTUR DATA

MINGGU BERIKUTNYA :

- TIPE DATA
- FUNGSI
- PROSEDUR

BAB II

TIPE DATA, FUNGSI DAN PROSEDUR

- **TIPE DATA**
- **FUNGSI**
- **PROSEDUR**

- Tipe data digunakan untuk menentukan batasan nilai yang digunakan suatu peubah (variabel)

- Macam tipe data :
  - Tipe Sederhana (primitif)
  - Tipe Terstruktur
  - Tipe String
  - Tipe Reference/Pointer

1. Tipe Data Sederhana

- Disebut juga tipe data **skalar**, yaitu suatu tipe data yang memungkinkan sebuah peubah untuk menyimpan sebuah nilai

- **Macam tipe sederhana** :
  - Tipe ordinal/integer
    - ShortInt, Integer, LongInt, Byte, Word subrange, dan enumerated
  - Tipe floating point/real
    - Real, Single, Double, Extended
  - Tipe char
    - Char
  - Tipe boolean
    - Logika

Tipe data Integer

merupakan tipe data berupa bilangan bulat, terbagi atas beberapa kategori seperti terlihat dalam tabel di bawah ini :

Tipe	Ukuran memori (dalam byte)	Jangkauan nilai
BYTE	1	0..255
SHORTINT	1	-128..127
INTEGER	2	-32768..32767
WORD	2	0..65535
LONGINT	4	-2147483648..2147483647

Tipe Data Real

merupakan jenis pecahan, yang dapat dituliskan secara biasa atau model scientific.

Penggolongan tipe data bilangan real dapat dilihat pada tabel di bawah ini :

Tipe	Ukuran memori (dalam byte)	Jangkauan nilai	Digit signifikan
SINGLE	4	1.5x10E-45 .. 3.4x10E38	7-8
DOUBLE	8	5.0x10E-324 .. 1.7x10E308	15-16
EXTENDED	10	1.9x10E-4951 .. 1.1x10E4932	19-20
COMP	8	-2E+63+1 .. 2E+63-1	19-20

Tipe data Character

Tipe data ini menyimpan karakter yang diketikkan dari keyboard, memiliki 266 macam yang terdapat dalam tabel ASCII (American Standard Code for Information Interchange). Contoh: ‘a’ ‘B’ ‘+’, dsb.

Yang perlu diingat bahwa dalam menuliskannya harus dengan memakai tanda kutip tunggal. Jenis data ini memerlukan alokasi memori sebesar 1(satu) byte untuk masing-masing data

- Tipe Char terdiri atas 26 huruf Latin, 10 angka Arab, dan sejumlah karakter grafik, seperti **tanda seru**
- Karakter dapat berisi karakter kosong yang digunakan sebagai pemisah (spasi)
- Karakter kosong (*blank*) diberi notasi □.

Tipe Data Boolean

merupakan tipe data logika, yang berisi dua kemungkinan nilai: TRUE (benar) atau FALSE (salah). Turbo Pascal for Windows memiliki tiga macam jenis ini yaitu: Boolean, WordBool, dan LongBool. Tipe boolean memakai memori paling kecil, sedangkan WordBool dan LongBool dipakai untuk menulis program yang sesuai dengan lingkungan Windows.

Tipe Data	Ukuran Tempat
Boolean	1 byte
WordBool	2 byte
Longbool	3 byte

2. Tipe Data Terstruktur

- Adalah suatu tipe data yang membolehkan sebuah peubah untuk menyimpan lebih dari satu data

- Macam tipe terstruktur :
  - Tipe Larik (Array)
  - Tipe Rekaman (Record/Struct)
  - Tipe Objek (Objek/Class)
  - Tipe Himpunan (Set/Enum)
  - Tipe Berkas (File)

1. **Tipe Larik (Array)**
   Larik adalah struktur data statik yang menyimpan sekumpulan elemen yang bertipe sama.

2. **Rekaman (Record)**
   Record adalah struktur data yang tersusun atas elemen-elemen yang jumlahnya tertentu dan tipe data elemennya dapat berbeda-beda

3. **Himpunan (Set)**
   Himpunan merupakan tipe yang unik dari Pascal.
   Type ini memungkinkan kita untuk mengadakan operasi himpunan.

4. **Berkas (File)**
   File terdiri dari *record-record* yang menggambarkan satu kesatuan data yang sejenis. Misalnya *file* mata pelajaran berisi data tentang semua mata pelajaran yang ada

3. Tipe Data String :

merupakan suatu data yang menyimpan array (larik), sebagai contoh 'ABCDEF' merupakan sebuah konstanta string yang berisikan 6 byte karakter. Ukuran Tempat untuk tipe data ini adalah 2 s/d 256 byte, dengan jumlah elemen 1 s/d 255. String dideklarasikan dengan string [konstanta] atau string. Bila ukuran string tidak didefinisikan maka akan banyak memakan ruang, karena ukuran string menyesuaikan dengan defaultnya.

Misalkan

var kata: string [20]; atau var kata: string; karena string merupakan array dari karakter. Maka kata[1] merupakan karakter pertama dari string, kemudian kata[2], merupakan elemen kedua, dst.

4. Tipe Data Pointer :

Suatu variabel yang points(menunjuk) ke sesuatu sehingga disebut "pointer".

Pointer merupakan variabel khusus yang berisi suatu address (alamat) di lokasi lain didalam memory.

Ada dua macam pointer :

- **typed**(tertentu):
  merupakan pointer yang menunjuk pada tipe data tertentu pada variable.

- **generic**(umum):
  merupakan pointer yang tidak menunjuk pada tipe data tertentu pada variable

PROSEDUR DAN FUNGSI

Perbedaan penggunaannya dalam bahasa pemrograman Pascal :

- Prosedur merupakan modul(subprogram) yg melakukan aktifitas tertentu tanpa adanya pengembalian nilai

- Fungsi terdapat pengembalian nilai

PROSEDUR & FUNGSI

PROSEDUR atau FUNGSI

- **FUNGSI** digunakan apabila modul program mengembalikan sebuah nilai

- **PROSEDUR** digunakan apabila modul program menghasilkan efek *netto* dari satu atau sekumpulan aksi

PROSEDUR

PENGERTIAN " PROSEDUR " PADA BAHASA PEROGRAMAN PASCAL

Seperti pada bahasa pemrograman lain bahasa pemrograman pascal pun memiliki yang namanya Prosedur

Prosedur merupakan bagian yang terpisah dari program dan dapat diaktifkan di manapun di dalam program. Kata **PROSEDUR** digunakan sebagai judul di bagian deklarasi prosedur, diikuti oleh **identifier** yang merupakan nama dari prosedurnya secara optional dapat diikuti lagi oleh kumpulan parameter yang diakhiri dengan titik koma.

Parameter yang ada pada Prosedur

Pada prosedur terdapat 2 jenis paramter, yaitu :

1. Parameter Formal : merupakan nama-nama variable (list nama) yang dipakai dalam mendefinisikan prosedur dan membuat prosedur tersebut dapat dieksekusi dengan nama-nama yang berbeda ketika dipanggil. Ada 3 jenis parameter formal :
   - Paramter Input : yaitu parameter yang diperlukan prosedur sebagai masukan untuk melakukan aksi yang efektif.
   - Parameter Output : yaitu parameter yang nilainya akan dihasilkan oleh prosedur.
   - Parameter Input / Output : yaitu parameter yang nilainya diperlukan prosedur sebagai masukan untuk melakukan aksi, dan pada akhir prosedur akan dihasilkan nilai yang baru.

2. Paramter Aktual : adalah nama-nama informasi yang dipakai ketika prosedur itu dipakai.

STRUKTUR PROSEDUR

- JUDUL (*header)* → *nama prosedur* dan deklarasi parameter(*kalau ada*)
- DEKLARASI → mengumumkan nama-nama dan tipe data
- ALGORITMA → badan prosedur (instruksi)

Nama Prosedur

- Nama yang unik
- Sebaiknya diawali dengan kata kerja karena prosedur berisi suatu aktifitas
  - Misalnya: **HitungLuas, CariAlamat, IsiTabung**, dll.

Notasi algoritma untuk PROSEDUR

PROSEDUR NamaProsedur(deklarasi parameter, jika ada)

{spesifikasi prosedur, berisi penjelasan tentang apa yg dilakukan oleh prosedur ini.

K.awal : keadaan sebelum prosedur dilaksanakan.

K. akhir : keadaan setelah prosedur dilaksanakan}

DEKLARASI

{semua nama yg dipakai di dalam prosedur dan hanya berlaku lokal di dalam prosedur ini}

ALGORITMA

{badan prosedur, berisi urutan instruksi}

- **Prosedur sama dengan program biasa tapi dikelompokan menjadi satu kesatuan yang terpisah-pisah.**

- **Subprogram yang dapat dipanggil di dalam program (atau subprogram lain)**

- **Tidak menghasilkan nilai balik.**

- **<u>Deklarasi prosedur</u>:**

```
procedure nama_procedure();
  begin
    {proses}
  end;
```

Contoh :

Procedure HitungLuasSegitiga

{menghitung luas segitiga dengan rumus : L = (alas x tinggi)/2}

{K.awal : sembarang}

{K.akhir : luas segitiga tercetak}

DEKLARASI

Alas, tinggi, luas : real

ALGORITMA

Read(alas, tinggi)

Luas ← (alas * tingg) / 2

Write(luas)

Pemanggilan Prosedur

- Prosedur bukan program yg beridiri sendiri
- Prosedur tidak dapat dieksekusi secara langsung.
- Instruksi-instruksi di dalam prosedur dapat dilaksanakan bila prosedur itu diakses.
- Prosedur diakses dg cara memanggil namanya dari program pemanggil (misalnya dari program utama atau modul program lainnya)
- Jika prosedur tanpa parameter, maka pemanggilannya cukup dg nama prosedurnya saja,

  contoh : **HitungLuasSegitiga**

Notasi Algoritma :

```
PROGRAM Segitiga
{menghitung luas segitiga}
DEKLARASI
    Procedure HitungLuasSegitiga
    {menghitung luas segitiga dengan rumus : L = (alas x tinggi)/2}
    {K.awal : sembarang}
    {K.akhir : luas segitiga tercetak}
    DEKLARASI
      Alas, tinggi, luas : real
    ALGORITMA
      Read(alas, tinggi)
      Luas ← (alas * tinggi) / 2
      Write(luas)
ALGORITMA
  HitungLuasSegitiga
```

Program utama yg memanggil nama prosedur: harus mendeklarasikan nama prosedur dan memanggilnya dg parameter aktual yg sesuai

```
PROGRAM Segitiga
{menghitung luas N buah segitiga}
DEKLARASI
  I, N : integer
  alas, tinggi : real
   Procedure HitungLuasSegitiga(input alas, tinggi : real)
   {menghitung luas segitiga dengan rumus : L = (alas x tinggi)/2}
   {K.awal : alas dan tinggi sudah terdefinisi nilainya }
   {K.akhir : luas segitiga tercetak}
   DEKLARASI
    luas : real
   ALGORITMA
     Luas ← (alas * tinggi) / 2
     Write(luas)
ALGORITMA
    read(N)    { tentukan banyaknya segitiga }
     for I ← 1 to N do
       read(alas, tinggi
       HitungLuasSegitiga(alas,tinggi)
     end for
```

Perbedaan fungsi dengan prosedur adalah :

- Pada fungsi, nilai yang dikirimkan balik terdapat pada nama fungsinya (kalau pada prosedur pada parameter yang dikirimkan secara acuan).

- Karena nilai balik berada di nama fungsi tersebut, maka fungsi tersebut dapat langsung digunakan untuk dicetak hasilnya. Atau nilai fungsi tersebut dapat juga langsung dipindahkan ke pengenal variable yang lainnya.

- Pada prosedur, nama prosedur tidak dapat digunakan lagsung, yang dapat langsung digunakan adalah parameternya yang mengandung nilai balik.

FUNGSI

FUNGSI

- Sub Program yg mempunyai tujuan/tugas spesifik
- Fungsi dan prosedur adalah merupakan sub program yang ekivalen
- Dalam beberapa masalah penggunaan salah satunya adalah hal yg lebih tepat.

- Adalah sub program yg memberikan/mengembalikan (*return*) sebuah nilai dari tipe tertentu (tipe dasar atau tipe bentukan)
- Fungsi di dalam program bersesuain dengan definisi fungsi di dalam matematika

Contoh :

a. $f(x) = 2X^2 + 5X - 8$

b. $h(x,y) = 3x - y + xy$

Dimana: f dan h adalah nama fungsi, sedangkan x dan y adalah parameter fungsi yg bersangkutan. Nilai yg diberikan oleh fungsi bergantung pada masukan parameter

- *Misal : x=2, maka f(2) = 2*2² + 5*2 = 10*

  *x=1, y=2, maka h(1,2) = 3*1 – 2+1*2=3*

Nilai 10 dan 3 adalah nilai yang diberikan (*return*) oleh masing fungsi f dan fungsi h.

Struktur fungsi

Function NamaFungsi(input deklarasi parameter, jika ada) → tipe (*tipe nilai yg diberikan oleh fungsi)*
{spesifikasi fungsi, menjelaskan apa yang dilakukan dan yang dikembalikan oleh fungsi}

DEKLARASI
{semua yg dipakai di dalam fungsi dan hanya berlaku di dalam fungsi didefinisikan di sini}

ALGORITMA
{badan fungsi, berisi instruksi-instruksi untuk menghasilkan nilai yg akan dikembalikan oleh fungsi}
return *ekspresi* *{ pengembalian nilai yg dihasilkan fungsi }*

- *Catatan :* parameter formal selalu berjenis parameter masukan, sehingga parameter formal selalu diawal dengan *keyword* INPUT
- *Ekspresi :* dapat berupa konstanta, peubah(*variabel*) atau sebuah rumus

Pemanggilan FUNGSI

- Fungsi di akses dengan cara memanggil namanya diikuti dengan parameter aktual (kalau ada)

  - ***Nilai yang dikembalikan oleh fungsi ditampung di dalam sebuah peubah (variabel) yng bertipe sama dengan tipe fungsi***

    Peubah ← NamaFungsi(parameter aktual, jika ada)

    Contoh : y ← F(5) *{ y harus bertipe real }*

Mengubah fungsi menjadi prosedur:

Menyatakan nilai yg dikembalikan (return value) oleh fungsi tsb dikonfersikan sebagai parameter keluaran pada prosedur

- Fungsi

```
Function Maks(input
   a,b:integer)→integer
Deklarasi
Algoritma
 if a ≥ b then
    return a
 else
    return b
endif
```

- Prosedur

```
Function TentukanMaks(input
   a,b:integer, output maks :integer)
Deklarasi
Algoritma
 if a ≥ b then
    maks ← a
 else
    maks ← b
endif
```

Mengubah prosedur menjadi fungsi:

Prosedur yg mempunyai satu buah parameter keluaran dapat ditulis sebagai fungsi dengan menyatakan parameter keluaran sebagai nilai yg dikembalikan oleh fungsi

- Prosedur

```
Procedure hitungrata2(input
    ndata:integer, output u: real)
Deklarasi
  x: integer
  I : integer
  jumlah : integer
Algoritma:
  jumlah ← 0
  for I ← 1 to Ndata do
    read (x)
    jumlah ← jumlah + x
 endfor

U ← jumlah/Ndata
```

- Fungsi

```
Procedure Rata2(input ndata:integer)
    → real
Deklarasi
  x: integer
  I : integer
  jumlah : integer
Algoritma:
  jumlah ← 0
  for I ← 1 to Ndata do
    read (x)
    jumlah ← jumlah + x
 endfor

return  jumlah/Ndata
```

BAB III
ARRAY

Array (Larik)

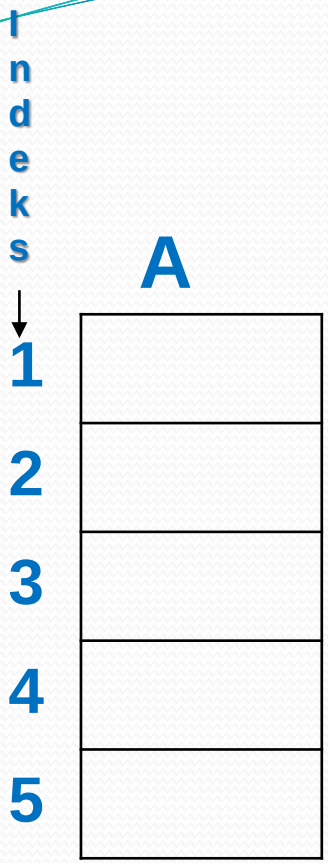
- **Suatu larik (array) adalah type terstruktur, yang terdiri dari sejumlah komponen-komponen yang mempunyai tipe yang sama.**
  Komponen-komponen ini disebut dengan tipe komponen (component type) atau type basis (base type).

- **Suatu larik mempunyai jumlah komponen yang banyaknya tetap.**

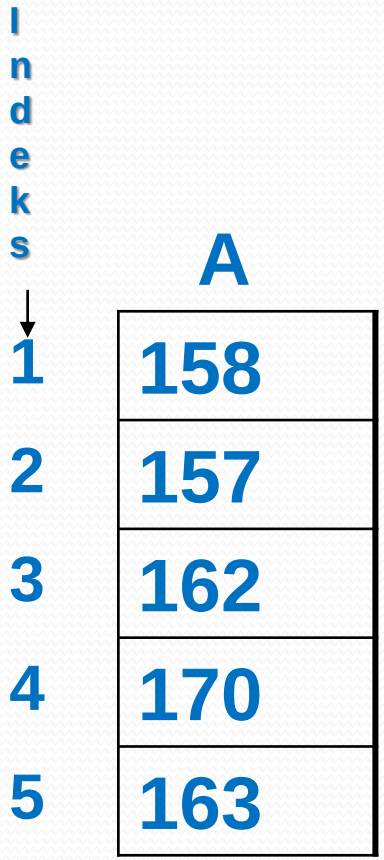
- **Banyaknya komponen dalam suatu larik ditunjukkan oleh suatu indeks yang disebut dengan tipe indek (index type).**
  Tipe indeks ini berbentuk ungkapan tipe ordinal.

- **Tiap-tiap komponen dilarik dapat diakses dengan menunjukkan nilai indeksnya (indeks value) atau disebut juga dengan istilah subscript,**

Elemen Array



Elemen Array kosong

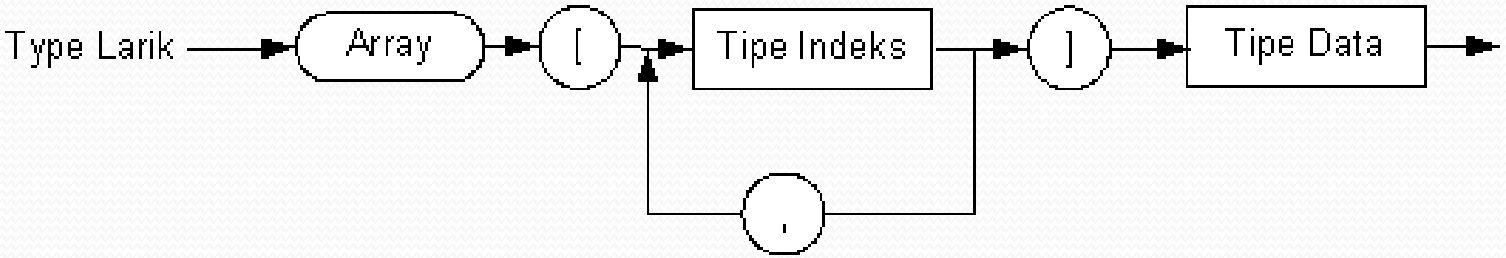


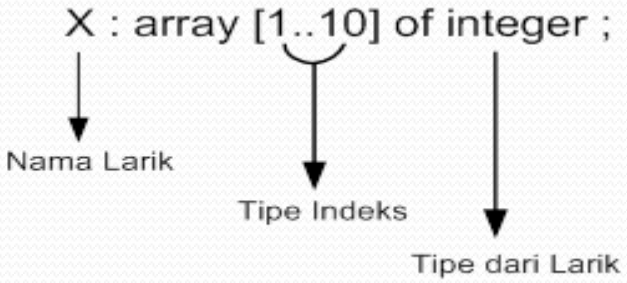
Elemen Array yg sudah diisi nilai

A[1], A[2], A[3], A[4], A[5]

Deklarasi Array

- **Array (Larik) adalah struktur(susunan) data yg statis, artinya elemen larik harus sudah diketahui sebelum program dieksekusi. Jumlah elemen larik tidak dapat diubah (ditambah/dikurangi) selama pelaksanaan program (program running).**





Keterangan

X telah dideklarasikan sebagai tipe larik yang bertipe integer dengan jumlah elemennya maksimum 10 elemen. Nilai-nilai elemen larik ini harus berisi nilai-nilai integer.

ilustrasi

Algoritma:

```
Kamus:
    nama_var_array:array[1..maks_array] of tipedata
```

Contoh:

```
Kamus:
    nama:array[1..5] of string
```

Algoritma:

```
Kamus:
    Const
        maks_array = ...

    nama_var_array:array[1..maks_array] of
tipedata
```

Contoh:

```
Kamus:
    Const
        maks_array = 5

    nama:array[1..maks_array] of string
```

Algoritma:

```
Kamus:
    Const
        maks_array = ...
    Type
        nama_type_array=array[1..maks_array] of tipedata

    nama_var_array:nama_type_array
```

Contoh:

```
Kamus:
    Const
        maks_array = 5
    Type
        data_nama=array[1..maks_array] of string

    nama:data_nama
```

- ***Array dari tipe data standar:***

  DEKLARASI :

  A : array[1.50] of integer

  NamaMhs : array[1..10] of string

  NilaiUjian : array[1..75} of real

- ***Dari tipe standar menjadi tipe baru :***

  DEKLARASI :

  Type

  LarikInt : array[1..100] of integer *{ nama tipe baru}*

  A:LarikInt *{A adalah sebuah variable (peubah) array dg 100 elemen dan bertipe integer}*

1. ***Sebagai sebuah konstanta :***

Mendefinisikan ukuran array (larik) sebagai sebuah **konstanta**

```
DEKLARASI:
    Const max= 10
    N: array[1..max] of real
    I: integer
ALGORITMA :
    For i:=1 to max do
    read(n[i])
    Endfor
```

2. ***Sebagai tipe bentukan terstruktur (RECORD) :***

- Tipe yang berbentuk rekaman (*record*), rekaman disusun oleh satu atau lebih *field*. Setiap *field* menyimpan data dari tipe dasar tertentu.

DEKLARASI :

Const NMaks = 100

Type Mahasiswa : record < NIM : integer
NAMA : String
IPK : real >

Type tabmhs : array[1..maks] of mahasiswa

Mhs : TabMhs

Penulisan elemen mhs :

Mhs[2] {elemen kedua dari array mhs}

Mhs[2].NIM {mengacu field NIM dari elemen kedua dari array Mhs}

Mhs[2].IPK {mengacu field IPK dari elemen kedua dari array Mhs}

Pengsian Nilai per field :

Mhs[i].NIM ← 102131002

Mhs{i].NAMA ← 'BUDI UTOMO'

Mhs[i].IPK ← 3.6

Menampilkan hasil per field :

Output([Mhs[i].NIM, Mhs[i].NAMA, Mhs[i].IPK)

Acuan Elemen Array(Larik)

- Elemen Array diacu melalui indeksnya.
- Nilai Indeks harus terdefinisi.
  - A[4] → mengacu elemen ke 4 dari larik A
  - NamaMhs[2] → mengacu elemen ke 2 dari larik namaMhs
  - A[i] → mencau elemen ke-I dari larik A, asalkan nilai I sudah terdefinisi.

contoh:

X [4] Mengacu elemen ke 4 dari larik X

Pemrosesan Array (Larik)

- Elemen array diproses secara beruntun melalu indeks yang terurut mulai dari elemen pertama sampai elemen terakhir dicapai

- skema umum algoritma memproses larik ialah **mengunjungi**

Operasi-operasi Array

1. **Penciptaan (create) array statis**

   Mempersiapkan array untuk diakses/diproses dengan asumsi elemen array diisi dengan angka 0 jika elemen arraynya diisi numerik/bilangan/angka atau diisi dengan karakter ” ”/””/’ ’untuk alphanumerik.

Algoritma:

```
Procedure create (Output nama_var_array:nama_type_array)
{I.S: elemen array diberi harga awal agar siap digunakan}
{F.S: menghasilkan array yang siap digunakan}

  indeks:integer

Algoritma:
  for indeks ← 1 to maks_array do
    nama_var_array(indeks)← 0 {elemen array numerik}
  endfor

EndProcedure
```

2. Traversal

Proses mengunjungi setiap elemen array satu persatu dari elemen pertama sampai elemen terakhir

Proses traversal:

1. Pengisian elemen array dengan data
2. Menampilkan elemen array
3. Penambahan data di array
4. Penyisipan data di indeks tertentu pada array
5. Penghapusan data di indeks tertentu pada array
6. Menentukan nilai maksimum dan minimum
7. Menghitung nilai rata-rata, dsb.

Algoritma

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

Algoritma:
  for indeks ← 1 to maks_array do
    proses
  endfor
  Terminasi {penutupan yang harus dilakukan setelah proses
        selesai}

EndProcedure
```

PROSES LARIK

Program Proses_Larik

KAMUS

Const : N = 8 {jumlah elemen larik}
Indeks : integer
A : array [1..N] of integer {deklarasi larik A dengan tipe data integer}

ALGORITMA

For Indeks ← 1 to N do
PROSES LARIK
Endfor

Catatan : **Tipe Data sejenis (homogen)**

Indeks data memiliki keterurutan

CONTOH PROSES

ALGORITMA

For Indeks ← 1 to N do

PROSES LARIK

Endfor

- ✓**Mengisi elemen larik dengan o (inisialisasi)**
- ✓**Mengisi elemen larik dari piranti masukan**
- ✓**Mencetak elemen larik ke piranti keluaran**

INISIALISASI

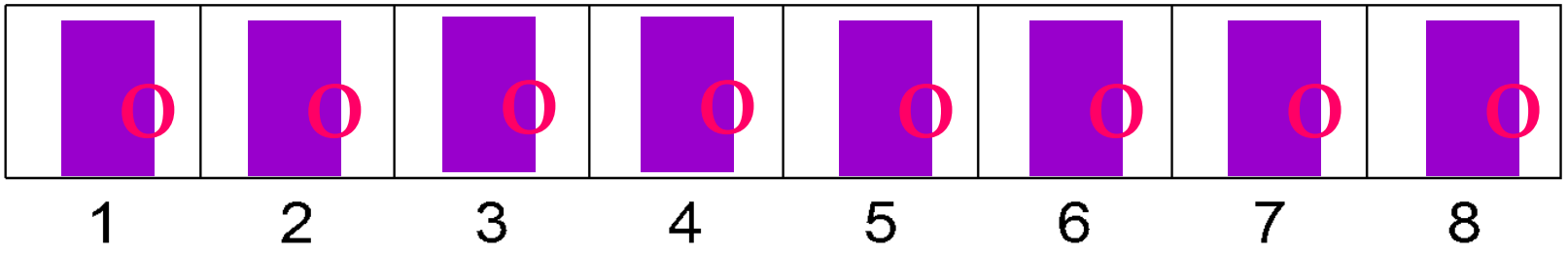
ALGORITMA

For Indeks ← 1 to 8 do

A[Indeks] = 0

Endfor

Array A satu dimensi :
8 indeks (1 s/d 8) dan data 1, 7, 18 dst.



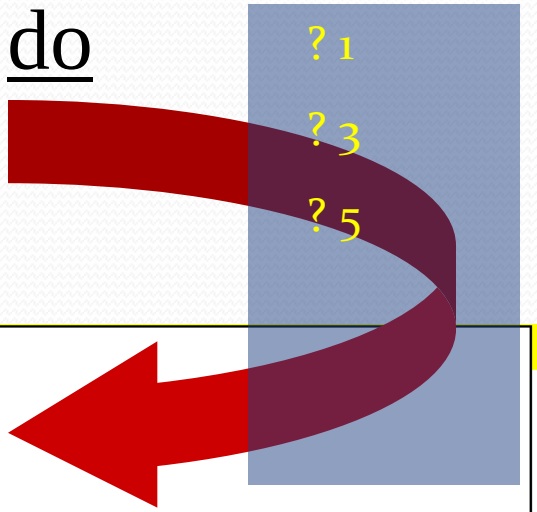
INPUT ELEMEN

ALGORITMA

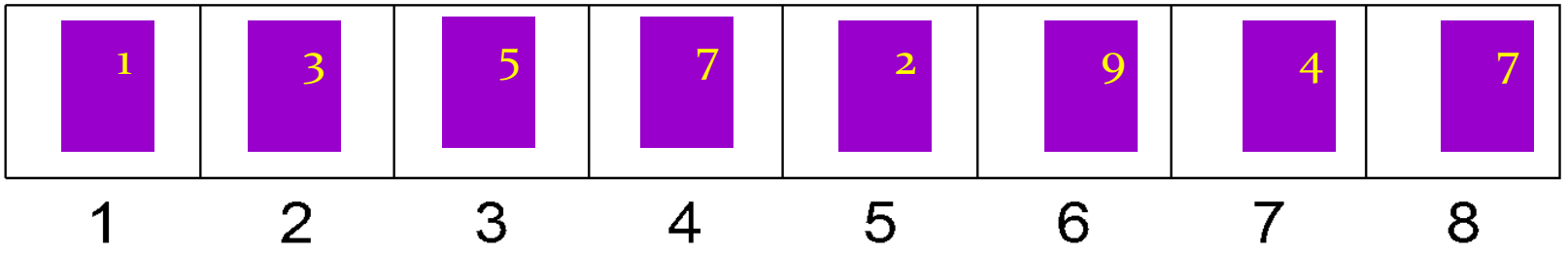
For Indeks ← 1 to 8 do

Input *A[Indeks]*

Endfor



Array A satu dimensi :
8 indeks (1 s/d 8) dan data 1, 7, 18 dst.



*CETAK ELEMEN

ALGORITMA

For Indeks ← 1 to 8 do

Print *A[Indeks]*

Endfor

Array A satu dimensi
8 indeks (1 s/d 8) dan data 1, 7, 18 dst.

1	2	3	4	5	6	7	8
1	3	5	7	2	9	4	7

OPERASI PADA ARRAY

1. **Operasi Memasukkan nilai**
   Bila array sudah dideklarasikan dan sudah diberi nama, maka dapat dimanfaatkan sesuai fungsinya sebagai objek data. Operasi **memasukkan** nilai adalah operasi untuk memasukkan nilai data ke dalam elemen-elemen array. Biasanya hal ini dilakukan dengan operasi penugasan (assignment) dengan objek array terletak sebagai operan di sebelah kiri tanda '**:=**'.

2. **Operasi mengambil nilai**
   adalah operasi untuk mendapatkan/membaca nilai dari suatu array. Dapat dilakukan ketika menggunakan array sebagai operan nada suatu operasi atau sebagai parameter sebuah fungsi/prosedur

3. Operasi Mengakses Array

Operasi mengakses suatu objek data lebih umum dikenal operasi memasukkan nilai ataupun membaca nilai. Jadi, operasi mengakses array dapat berupa memasukkan nilai atau membaca nilai array. Operasi ini dapat dilakukan pada array secara keseluruhan ataupun pada suatu elemen tertentu.

4. Mengakses Elemen Array Secara Acak/Random

Setiap elemen array dapat diperlakukan secara individual terlepas dari elemen-elemen lainnya. Misalnya dalam hal memasukkan data, nilai [8] dapat dimasukkan lebih dulu daripada elemen lainnya.meskipun akhirnya semua elemen array akan diakses, namun tidak ada aturan yang pasti tentang urutan mengaksesnya. Kita bisa saja mengakses nilai [6] tanpa mengakses komponen array lainnya.

5, Mengakses Elemen Array Secara Sequensial (Berurutan)

Struktur array yang elemen-elemennya tersusun secara berurutan, memungkinkan diakses sebagian atau seluruh elemen array secara berurutan.

Untuk proses mengunjungi elemen array (traversal of array) secara berurutan dapat dilakukan dengan proses looping (perulangan). Pada model-model di atas, elemen-elemen array diakses secara berurutan dengan selisih satu index.
Jadi elemen ke-8 akan diakses sebelum atau sesudah elemen ke-7 ataupun ke-9.
Pergeseran index dilakukan dengan menambah atau mengurangi index sebelumnya dengan 1, dapat juga membuat model lain dengan mengubah selisih indexnya menjadi 2, 3, atau sesuai keperluan, asalkan selalu berada pada range index.

6. Mengakses Array Secara Keseluruhan

Selain mengakses komponen array, dapat juga mengakses array secara keseluruhan yang akan mempengaruhi seluruh elemennya sekaligus. Sebagai contoh A dan B adalah dua buah variabel array yang tipenya sama, dan jumlah elemennya juga sama, maka dalam Pascal dapat dilakukan operasi penugasan A := B yang berarti memasukkan nilai dari setiap elemen array B ke semua elemen A pada Index-Index yang bersesuaia

Ukuran efektif Array (Larik)

- Jumlah elemen larik sudah ditentukan (statis), tapi belum tentu semua digunakan/dipakai
- Jumlah elemen yang dipakai itulah yang disebut dengan ukuran/jumlah efektif array.
- Ukuran efektif dapat dicatat di dalam peubah (variabel) tertentu, misalnya ***n*.**

Tiga cara untuk menentukan jumlah elemen efektif dari array (Larik) :

1. Jika Jumlah elemen efektif ditentukan di awal
2. Jika Jumkah elemen efektif diketahui di akhir pembacaan
3. Jika Jumlah elemen efektif baru diketahui di akhir pembacaan (variasi dari versi 2)

Jumlah elemen efektif ditentukan di awal

```
Procedure BacaLarik(Output A : LarikInt, input n : integer)
  {Mengisi elemen-elemen larik A[1..n] dengan pembacaan}
{K. Awal : n adalah jumlah elemen efektif larik, nilainya terdefinisi}
{ K. Akhir : setelah pembacaan, seluruh elemen larik A berisi nilai-nilai
    yang dibaca dari piranti masukan (keyboard)}
DEKLARASI
    i : integer  {pencatat indeks larik }
ALGORITMA
 for i ← 1 to n do
      read(A[i])
 endfor
```

Jumlah elemen efektif diketahui di akhir pembacaan

- Setiap kali selesai pembacaan satu buah elemen, akan dilakukan konfirmasi apakah masih ada lagi elemen larik yang akan dimasukkan, seperti statement di bawah ini :

  Write('Lagi? (y/t)')

  JIka jawabnya 'y' maka pembacaan dilanjutkan, jika 't' maka proses pembacaan dihentikan. Jumlah elemen yang dibaca di catat di dalam suatu variabel (peubah)

```
Procedure BacaLarik2(Output A: Larikint, Output n: integer)
{K. Awal : sembarang}
{K. Akhir : sebanyak n buah elemen larik A berisi nilai-nilai
yang dibaca; n berisi jumlah elemen larik yang diisi}
DEKLARASI
    Jawab : char
ALGORITMA
    N ← 0
    Repeat
      n ← n + 1
        Read(A[n])
        Write('Lagi? (y/t)')
        Read(jawab)
Until jawab = 't'
```

Jumlah elemen efektif baru diketahui di akhir pembacaan (variasi dari versi 2)

Proses pembacaan dianggap selesai jika nilai yang dibaca adalah suatu tanda, misalnya 9999.

```
Procedure BacaLarik3 (output A, LArikint, output n : integr)
{mengisi elemen-elemen larik A[1..n] dg cara pembacaan. Akhir pembacaan
    ditandai jika nilai yang dibaca adalah 9999}
{K.Awal : sembarang }
K. Akhit : sebanyak n buah elemen larik A berisi nilai-nilai yang dibaca; n
    berisi jumlah larik yang diisi.}
DEKLARASI
    x : integer  {menyimpan sementara nilai yang di baca}
ALGORITMA
 n ← 0
 read(x)
 while x ≠ 9999 do
     n ← n +1
     A[n] ← x
     read(x)
 endwhile
 {x = 9999}
```

Menghitung Nilai Rata-rata

- Data yang sudah disimpan di dalam Larik, selanjutnya data tersebut dapat dimanipulasi

```
Procedure hitungRataRata(input A:Larikint, input n:integer)
DEKLARASI
    I : integer
    Jumlah : real
ALGORITMA
    I ← 1   {dimulai dari elemen pertama}
    Jumlah ← 0  {jumlah total nilai mula mula}
    For i ← 1 to n do
       jumlah ← jumlah + A[i]
    Endfor
    U ← jumlah/n
```

Notasi Algoritma – hitung rata rata

```
PROGRAM Rerata

DEKLARASI
    const
            NMaks = 100
    type
            LarikInt : array[1..NMaks] of integer
    A : LArikInt
    n : integer
    u : integer      { nilai rata-rata }

    procedure BacaLarik1(output A : Larikint, input n :integer)
        { mengisi elemen larik A[1..n] dengan pembacaan }

    procedure HitungRataRata(input A : LArikint, input n : integer. Output u :
    real)
        {menghitung nilai rata-rata larik A}
```

ALGORITMA

read(n) {tentukan jumlah elemen larik yang akan digunakan }

BacaLarik1(A, n)

HitungRataRata(A, n, u)

write(u)

Kapan menggunakan Larik (array):

- Untuk menyimpan sejumlah data yang bertipe sama.
- Untuk menghindari penggunaan nama-nama peubah(variabel) yang banyak.
- Agar penulisan program lebih efisien dalam penulisan kode programnya.