ALPRO

GENESIS 64

Apa itu Algoritma dan Pemrograman?

- Algoritma: Urutan langkah-langkah untuk memecahkan masalah yang disusun secara sistematis dan logis.
- Program: Kumpulan instruksi-instruksi tersendiri yang biasanya disebut source code yang dibuat oleh programmer. Program merupakan realisasi dari Algoritma.

Syarat Algoritma yang Baik

1. Tingkat kepercayaannya tinggi (realibility)
2. Pemrosesan yang efisien (cost rendah)
3. Sifatnya general
4. Bisa dikembangkan (expandable)
5. Mudah dimengerti
6. Portabilitas yang tinggi (portability)
7. Tepat, betul, teliti (precise)
8. Efektif
9. Harus terminate (jalannya program harus ada kriteria berhenti)

Langkah Pembuatan Program

1. Mendefinisikan masalah
   a. Input apa saja yang tersedia
   b. Output yang diinginkan
   c. Kendala yang harus dihadapi
   d. Data apa saja yang dimiliki
   e. Operator yang digunakan
2. Buat algoritma dan struktur cara penyelesaian
3. Menulis program

   Memilh Bahasa yang dikuasai dan kompatibel terhadap banyak platform computer.
4. Mencari kesalahan
5. Uji dan Verifikasi Program
6. Dokumentasi Program
7. Pemeliharaan Program

Struktur Penulisan Algoritma

1. Judul (Header) adalah bagian teks algoritma yang digunakan sebagai tempat mendefinisikan nama dengan menentukan apakah teks tersebut adalah program, prosedur, fungsi. Contoh: Program_Luas_Segitiga."
2. Kamus (Deklarasi) adalah bagian teks algoritma sebagai tempat untuk mendefinisikan nama type, nama konstanta, nama variabel, nama fungsi, nama prosedur.
3. Algoritma (Deskripsi) adalah bagian inti dari suatu algoritma yang berisi instruksi atau pemanggilan aksi yang telah didefinisikan.

Penyajian Algoritma

1. Struktur Bahasa Indonesia

Penyajian algoritma dengan menggunakan bahasa Indonesia tidak memiliki aturan khusus seperti pada bahasa pemrograman dan dapat digunakan untuk seluruh bahasa manapun. Contoh: Menghitung rata-rata tiga buah data.

Algoritma bahasa Indonesia :

1. Baca bilangan a, b, dan c
2. Jumlahkan ketiga bilangan
3. Bagi jumlah dengan tiga
4. Tulis hasilnya

2. Pseudo-Code

Menggunakan kode yang mirip dengan kode pemrograman yang sebenarnya. Contoh: Menghitung rata-rata tiga buah data

Algoritma dengan struktur pseudocode :

1. Input (a, b, c)
2. Jml = a+b+c
3. Rerata = jml/3
4. Output rerata

3. Flowchart

Penggambaran secara grafik dari langkah-langkah dan urut-urutan prosedur dari suatu program. Flowchart membantu analis dan programmer untuk memecahkan masalah ke dalam segmen-segmen yang lebih kecil dan menolong dalam menganalisis alternatif-alternatif lain dalam pengoperasian.

Keterangan	Lambang
Mulai/selesai *(terminator)*	
Aliran data	
Input/Output	
Proses	

Keterangan	Lambang
Percabangan *(Decision)*	
Pemberian nilai awal suatu variabel *(Preparation)*	
Memanggil prosedur/fungsi *(Call)*	

Keterangan	Lambang
Perulangan	

Keterangan	Lambang
Connector (di halaman yg sama)	A
Off page Connector (halaman lain)	B

Dokumen / Multi dokumen	
Harddisk	

Contoh Flowchart

Problem: Menentukan bilangan ganjil a

Sturuktur Dasar Algoritma

1. Struktur pengulangan

Perintah program akan terus dilakukan selama intruksi memenuhi dan akan berhenti jika intruksi tidak memenuhi. Terdapat tiga jenis struktur perulangan,

1. For-next : instruksi akan terus dilakukan sampai kondisi yang diminta terpenuhi.

2. While-do : seleksi kondisi berada diatas, instruksi while-do akan terus mengulang selama kondisi masih memenuhi dan akan berhenti apabila instruksi bernilai false.

3. Repeat-until : seleksi kondisi berada di bawah, instruksi repeat until akan terus berulang selama kondisi belum terpenuhi dan akan berhenti hingga instruksi bernilai true.

Sturuktur Dasar Algoritma

2. Struktur pemilihan

Struktur yang membuat instruksi mengklasifikasikan apakah kondisi “A” bernilai benar atau salah sehingga dapat dilakukan aksi berbeda antara keduanya. Terdapat dua struktur pemilihan,

1. If-then : Jika kondisi bernilai true maka akan dilakukan instruksi selanjutnya, dan jika kondisi bernilai false maka instruksi akan selesai.
2. If-then-else : Jika kondisi bernilai true atau false maka akan dilakukan instruksi berbeda sesuai kondisi yang terpenuhi.

3. Struktur runtutan

Struktur dimana mengharuskan instruksi sebelumnya untuk lanjut ke instruksi selanjutnya (dilakukan secara berturutan).

Sturuktur Bahasa Pascal

Struktur program menggunakan Bahasa Pascal terdiri dari:

1. Judul Program
2. Blok Program atau Badan Program
   a. Bagian Deklarasi
   (deklarasi label, konstanta, tipe, variable, prosedur, fungsi)
   b. Bagian Pernyataan

Contoh Sturuktur Bahasa Pascal

```
Program   Tampil;
Var
        Nama:string[10];
Begin
        Nama:='Nina';
        Write('Nama  :  ');
        Write(Nama);
End.
```

Judul Program

Badan Program, Bagian Deklarasi

Badan Program, Bagian Pernyataan

Penulisan Program Pascal

JUDUL

- Judul program sifatnya adalah opsional dan tidak terlalu berarti dalam program. Judul program bila ditulis, harus terletak pada awal dari program dan diakhiri dengan titik koma.
- Judul program ditulis dalam satu kata, jika lebih dari duakata harus disambung dengan tanda hubung bawah (tidak boleh ada spasi kosong).
- Contoh:

program: rata_rata_hitung;

BADAN PROGRAM

- Badan program ditulis diatara kata Begin dan End. Akhir penulisan End harus diakhiri tanda titik.

BADAN PROGRAM: DEKLARASI

- Bagian deklarasi sering juga kamus program.
- Bagian deklarasi digunakan bila di dalam program menggunakan pengenal (identifier). Pengenal dapat berupa label, tipe, variabel, prosedur, atau fungsi.
- Contoh:

```
var
        nilai: integer;
```

BADAN PROGRAM: PERNYATAAN

- Penulisan pernyataan-pernyataan pada umumnya menjorok masuk beberapa kolom. Tidak berpengaruh padaproses, hanya dimaksudkan supaya mempermudah pembacaan program, sehingga akan lebih terlihat bagian-bagiannya.
- Setiap akhir pernyataan harus diakhiri dengan tanda titik koma.

Elemen-elemen Program Pascal

1. Simbol-simbol dasar
2. Reserved words
3. Identifier
4. Tipe Data
5. Tanda Operasi
6. Komentar Program

Simbol Dasar

Program Pascal dapat dibentuk dari simbol-simbol yang terdiri dari huruf-huruf, angka-angka dan simbol-simbol khusus.

1.Huruf

Huruf yang digunakan adalah huruf A sampai dengan Z, a sampai dengan z dan garis bawah (_). Huruf besar dan huruf kecil dianggap sama, tidak dibedakan.

2.Angka

Angka yang digunakan adalah angka 0 1 2 3 4 5 6 7 8 9

3.Simbol khusus

Simbol khusus yang dapat digunakan adalah:

+ -* / =^ () []{} . ,: ; '# $ <=>= := ..(* *) (..)

Reserved Word

kata-kata yang sudah didefiniskan oleh Pascal yang mempunyai maksud tertentu. Reserved words tidak dapat dipergunakan sebagai identifier.

And	Array	Begin	Case	Const
Div	Do	Downto	Else	End
File	For	Forward	Function	Goto
If	In	Label	Mod	Nil
Not	Of	Or	Procedure	Program
Record	Repeat	Set	Shl	Shr
String	Then	To	Type	Until
Uses	Var	While	With	Xor

Identifier

- Identifier adalah nama yang diberikan sendiri oleh pengguna untuk konstansa, variable, prosedur, fungsi.
- Identifier didefinsikan sendiri secara bebas, tetapi harus mengikuti ketentuan sebagai berikut.

1.Terdiri dari gabungan huruf dan angka dengan karakter pertama harus berupa huruf. Huruf besar dan huruf kecil dianggap sama.

2.Tidak boleh mengandung spasi.

3.Tidak boleh mengandung simbol-simbol khusus, kecuali garis bawah.

4.Panjangnya bebas, tetapi hanya 63 karakter pertama yang dianggap siginifikan.

Contoh Identifier

Pengenal	Keterangan
NamaSiswa	Benar
No_Induk	Benar
P3K	Benar
1X	Salah, karakter pertama harus huruf
A&B	Salah, tidak boleh mengandung simbol khusus
A B	Salah, tidak boleh ada spasi kosong

Tipe Data di Pascal

Tipe Data Integer

Tipe	Ukuran Memori	Jangkauan Nilai
Byte	1 byte	0..255
ShortInt	1 byte	-128..127
Integer	2 byte	-32768..32767
Word	2 byte	0..65535
LongInt	4 byte	-2147483648..2147483647

Contoh Tipe Data Integer

Contoh:

Misalnya kita akan menggunakan suatu variabel yang bernama Jumlah dan nilai akan dikandungnya tidak lebih dari 255, maka dapat digunakan tipe data byte.

```
Var
   Jumlah:byte;
Begin
   Jumlah:=200;
   Writeln('Nilai   Jumlah=',Jumlah);
End.
```

Tipe Data Real

REAL

- Nilai konstanta numeril real berkisar dari 1E-38 sampai dengan 1E+38 dengan mantisa yang signifikan sampai dengan 11 digit.
- E menunjukkan nilai 10 pangkat. Nilai konstanta numeril real menempati memori sebesar 6 byte.
- Contoh:

Misalnya kita akan membuat pengenal variabel yang bernama Luas dan akan bernilai desimal atau ada pecahannya.

Jenis-jenis Tipe Data Real

Type	Range	Significant digits	Ukuran (dalam byte)
Real	platform dependant* (2,9 E-39 .. 1.7 E38)	???* (11-12)	4 – 8 (6)
Single	1,5E-45 .. 3,4E38	7-8	4
Double	5,0E-324 .. 1,7E308	15-16	8
Extended	1,9E-4951 .. 1,1E4932	19-20	10
Comp	$-2^{63}+1$ .. $2^{63}-1$	19-20	8

Tipe Data Karakter

KARAKTER

Nilai data karakter berupa karakter yang ditulis diantara tanda petik tunggal, seperti ‘A’, ‘a’, ‘!’, ‘5’ dan sebagainya. Penggunaan variabel untuk menyimpan data tipe karakter ini harus dideklarasikan dengan tipe Char. Contoh:

```
    Karakter:char;
Begin
    Karakter:='A';
    Writeln('Karakter   tersebut=',karakter);
End.
```

Tipe Data String

- Nilai data string merupakan urut-urutandari karakteryang terletak di antara tanda petik tunggal.
- Bila karakter petik merupakan bagian dari string, maka dapat ditulis dengan menggunaan dua buah petik tunggal berurutan.
- Nilai datastring akan menemapati memori sebesar banyaknya karakter stringnya ditambah dengan 1 byte. Bila panjang dari suatu string di dalam deklarasi variabel tidak disebutkan, maka dianggap panjangnya adalah 255 karakter.

Contoh Tipe Data String

```
Var
    Tanggal:string[10];
    Hari      :string[6];
    Pasaran:string[6];
Begin
    Tanggal:='26-12-2005';
    Hari    :='Senin';
    Pasaran:='Pahing';
    Writeln(Tanggal,'  ',hari,'  ',Pasaran);
End.
```

Tipe Data Boolean

Tipe data boolean mempunyai dua buah nilai, yaitu True dan False. Kita dapat mendeklarasikan suatu variabel dengan tipe boolean dan mengisi variabel tersebut dengan nilai data boolean True atau False. Tipe data ini tidak dapat dioperasikan kedalam bentuk aritmatika.

```
Var
   Lagi:boolean;
Begin
   Lagi:=True;
   Writeln('Nilai   lagi=',lagi);
End.
```

Tipe Data Boolean

Tipe data boolean mempunyai dua buah nilai, yaitu True dan False. Kita dapat mendeklarasikan suatu variabel dengan tipe boolean dan mengisi variabel tersebut dengan nilai data boolean True atau False. Tipe data ini tidak dapat dioperasikan kedalam bentuk aritmatika.

```
Var
   Lagi:boolean;
Begin
   Lagi:=True;
   Writeln('Nilai   lagi=',lagi);
End.
```

Tipe Data Pointer

Suatu pointer adalah suatu variabel yang berisi alamat di memori. Dengan kata lain, pointer akan menunjukkan letak dari data di memori.

```
Type
   Tipestring=string[40];
   Pointerstring=^Tipestring;
     Var
        Letaknama:pinterstring;
     Begin
        Letaknama^:='Pascal';
        Writeln(Letaknama^);
     End.
```

Tipe Data Ordinal

Tipe data ordinal adalah tipe data yang merupakan subset dari tipe data sederhana. Yang termasuk tipe data ordinal adalah:

- Tipe data standar (Ineteger, ShortInt, LongInt, Byte, Word, Boolean, Char)
- Tipe data yang didefinisikan sendiri oleh pemakai.

Tipe Data Ordinal

Tipe data ordinal mempunyai 4 karakteristik sebagai berikut.

1. Nilai dari tipe data ordinal mempunyai nilai yang berurutan.
2. Fungsi standar Ord dapat digunakan untuk menghasilkan nilai urutannya.
3. Fungsi standar Pred dapat digunakan untuk mengasilkan nilai urutan sebelumnya.
4. Fungsi standar Succ dapat digunakan untuk menghasilkan nilai urutan sesudahnya.

Contoh Tipe Data Ordinal

```
Begin
   Writeln(Ord('A'));
   Writeln(Pred('B'));
   Writeln(Succ('C'));
End.
```

Bila program dijalankan, hasilnya:

```
65
A
D
```

Operator

Assignment operator adalah operator pengerjaan, operator ini menggunakan simbol titik dua diikuti oleh tanda sama dengan (:=).

```
Luas:=panjang*lebar;
Kota:='Kuningan';
```

Operator

Operator Aritmatik

Operator	Jenis Operasi	Tipe Operand	Tipe Hasil
+	Penjumlahan	Integer, real	Integer, real
-	Pengurangan	Integer, real	Integer, real
*	Perkalian	Integer, real	Integer, real
/	Pembagian untuk bilangan riil	Integer, real	real
div	Pembagian untuk bilangan bulat	Integer	Integer
mod	Sisa bagi	integer	integer

Operator

Operator Relasional

Misalkan

A = 10

B = 20

Operator	Description	Example
=	Checks if the values of two operands are equal or not, if yes, then condition becomes true.	(A = B) is not true.
<>	Checks if the values of two operands are equal or not, if values are not equal, then condition becomes true.	(A <> B) is true.
>	Checks if the value of left operand is greater than the value of right operand, if yes, then condition becomes true.	(A > B) is not true.
<	Checks if the value of left operand is less than the value of right operand, if yes, then condition becomes true.	(A < B) is true.
>=	Checks if the value of left operand is greater than or equal to the value of right operand, if yes, then condition becomes true.	(A >= B) is not true.
<=	Checks if the value of left operand is less than or equal to the value of right operand, if yes, then condition becomes true.	(A <= B) is true.

Operator

Operator Logika

Operator	Jenis Operasi	Tipe Operand	Tipe Hasil
not	Negasi	boolean	boolean
and	Conjunction	boolean	boolean
or	Disjunction	boolean	boolean
xor	Exlusive disjunction	boolean	boolean

Operator

Operator Bitwise

- Bitwise operator digunakan untuk operasi bit per bit pada nilai binary (biner).
- Operator ini dugunakan untuk operand integer dan menghasilkan ouput integer juga
- Misalkan A=60 dan B=13, dalam binary

A = 0011 1100

B = 0000 1101

Komentar Program

Komentar program dituliskan untuk memberikan catatan bagi programer tanpa merusak instruksi program. Contoh penulisan komentar program:

```
{this is a comment}
(* this is another comment *)
// this is a comment up to the end of the line
```

KONSEP PEMILIHAN

- Pemilihan Sederhana (If-then)

- Konsep If-then-else

Dalam pembuatan program sering ditemui berbagai masalah, salah satu masalah tersebut yaitu proses seleksi-seleksi atau pemilihan statemen atau biasa disebut dengan percabangan/pemilihan.

Percabangan adalah suatu pemilihan statemen yang akan dieksekusi dimana pilihan tersebut didasarkan atas kondisi tertentu untuk mengarahkan perjalanan suatu proses.

Artinya statemen yang terdapat pada suatu blok percabangan akan dieksekusi jika kondisi yang didefinisikan terpenuhi (bernilai benar) tetapi jika kondisi tersebut tidak terpenuhi (bernilai salah) maka statemen tersebut tidak akan dieksekusi atau diabaikan oleh compiler.

Bahasa pemrograman menyediakan pernyataan IF…THEN dan kata kunci yang lain seperti CASE untuk melakukan suatu percabangan.

Misalnya, dalam kondisi cuaca dingin:

"If it is cold then put your coat on!"

Aturan penulisan statemen If–Then adalah sebagai berikut:

```
If kondisi then
statemen1
else
statemen2
```

dimana kondisi adalah ekspresi boolean.

Contoh:

```
If Age < 18 then
writeln ('Sorry, this is underage');
```

PEMILIHAN SEDERHANA

Statemen if dengan dan tanpa bagian else diilustrasikan dengan bagan alir (flow chart) pada gambar berikut:

STRUKTUR PERMILIHAN UNTUK MASALAH BATASAN UMUR.

Sebuah aturan untuk menonton sebuah film tertentu adalah sebagai berikut, jika usia penonton lebih dari 17 tahun maka penonton diperbolehkan dan apabila kurang dari 17 tahun maka penonton tidak diperbolehkan nonton.

Mulai
Read usia
Usia>17
Ya
Tidak
Print "Silahkan Menonton"
Print "Anda Tidak Boleh Menonton"
Selesai

PEMILIHAN SATU KASUS (1)

Notasi algoritmik untuk analisis dengan satu kasus adalah dengan menggunakan struktur If – then

If kondisi then

aksi

endif

Aksi sesudah kata then (dapat berupa satu atau lebih aksi) hanya akan dilakukan apabila kondisi bernilai benar (true), sebaliknya apabila kondisi bernilai salah (false), tidak ada aksi apapun yang dikerjakan.

Kata endif sengaja ditambahkan untuk mempertegas awal dan akhir struktur if – then.

Contoh: Pascal Credit Card

```
PROGRAM SimpleDecision(INPUT, OUTPUT);
CONST
        Limit = 1000000;
VAR
        Jumlah: real;
BEGIN
       write('Masukkan jumlah:');
       Readln (Jumlah);
       If Jumlah <= Limit then
         Writeln('Tagihan Anda diterima.'); {Akhir pernyataan if}
       Writeln('Tekan ENTER untuk keluar ..');
        Readln
END.
```

Contoh: Pascal Credit Card dengan 2 kondisi

```
PROGRAM Twoconditions(INPUT, OUTPUT);
CONST
        Limit = 1000000;
VAR
        Jumlah: real;
BEGIN
    write('Masukkan jumlah:');
    Readln (Jumlah);
    if Jumlah <= Limit then
      Writeln('Tagihan Anda diterima.');
    if Jumlah> Limit then
      Writeln('Jumlah tersebut melebihi batas limit kredit Anda.');
   Writeln ('Terima kasih telah menggunakan kartu kredit Pascal.');
   Writeln ('Tekan ENTER untuk keluar ..');
    Readln
    END.
```

PEMILIHAN DUA KASUS

Notasi algoritmik untuk analisis dengan dua kasus adalah dengan menggunakan struktur If – then – else

```
If kondisi then
    aksi1
  else
    aksi2
endif
```

Aksi1 akan dilakukan jika kondisi bernilai benar (true), tetapi sebaliknya jika kondisi bernilai salah (false), maka aksi2 yang akan dikerjakan. Perhatikan bahwa “else” menyatakan lingkaran (negation) dari kondisi.

PEMILIHAN DUA KASUS

Dalam suatu perhitungan nilai P = X +Y.

Jika P positif, maka Q = X * Y, sedangkan jika negatif, maka nilai Q = X/Y.

Buatlah flowchart untuk mencari nilai P dan Q

Contoh: Pascal Credit Card dengan If-Then-Else

```
PROGRAM CreditCard(INPUT, OUTPUT);
CONST
        Limit = 1000000;
VAR
        Jumlah  :  real;
BEGIN
    write('Masukkan jumlah:');
    Readln (Jumlah);
    if Jumlah <= Limit then
        BEGIN
            Writeln('Tagihan Anda diterima.');
            Writeln('Your price + tax is $, 1.05*jumlah:0:2)  {; optional}
        END
    ELSE
        BEGIN
            Writeln('Jumlah tersebut melebihi batas limit kredit Anda.');
            writeln('batas maksimum dalam $', limit)
         END;
    Writeln ('Terima kasih telah menggunakan kartu kredit Pascal.');
    Writeln ('Tekan ENTER untuk melanjutkan ..');
    Readln
END.
```

PEMILIHAN TIGA KASUS ATAU LEBIH

Masalah yang mempunyai tiga buah kasus atau lebih tetap dapat dianalisis dengan struktur If – then – else sebagaimana halnya pada masalah dengan dua kasus (ELSE – IF Ladders)

```
If kondisi 1 then
    aksi1
  else
    if kondisi2 then
      aksi2
    else
        if kondisi3 then
          aksi3
        endif
    endif
endif
dan seterusnya
```

STATEMEN IF TERSARANG

Statemen if yang terkandung pada statemen if yang lain, yang lebih atas, disebut dengan statemen if tersarang (nested if). Hal ini mungkin terjadi jika statemen yang seharusnya dijalankan setelah pemeriksaan kondisi atau ekspresi boolean, tidak dapat langsung dijalankan karena harus memeriksa kondisi atau ekspresi boolean yang lain

Bentuk umum :

```
IF kondisi1 THEN
Begin
   IF kondisi2 THEN
         Statemen1
   ELSE
         Statemen2
    End ;
```

Bisa juga bentuknya :

```
IF kondisi1 THEN
   IF kondisi2 THEN
      Statemen1
   ELSE
      Statemen2
ELSE
   Statemen3;
```

STATEMEN IF TERSARANG

```
IF kondisiTHEN-1
   IF kondisiTHEN-2
     …
      IF kondisiTHEN-n
         Pernyataan-n1
      ELSE
         Pernyataan-n2
      …
   ELSE
    Pernyataan-2
ELSE
  pernyataan-1;
```

STATEMEN IF TERSARANG

Contoh: Skor dan Grade

Program ini menampilkan skor siswa dan gradenya sesuai dengan klasifikasi berikut:

1. Grade "A" dengan nilai dari 90% sampai 100%.
2. Grade "B" dengan nilai dari 80% menjadi 89%.
3. Grade "C" dengan nilai dari 70% menjadi 79%.
4. Grade "D" dengan nilai dari 60% menjadi 69%.
5. Grade “E" dengan nilai kurang dari 60%.

STATEMEN IF TERSARANG

```
PROGRAM ScoresAndGrades1 (INPUT, OUTPUT);
VAR
   Nilai : INTEGER;
BEGIN
   Write('Masukkan skor/nilai: ');
   Readln(Nilai);
   Writeln;
    If nilai > 59 THEN
     If nilai > 69 THEN
      If nilai > 79 THEN
       if nilai > 89 THEN
           Writeln ('Excellent. Grade Anda adalah' 'A' )
       ELSE
           Writeln ('Very good. Grade Anda' 'B' )
      ELSE
          Writeln(‘Good. Grade Anda' 'C' )
     ELSE
         Writeln (‘Passed. Grade Anda' 'D' )
    ELSE
     Writeln (‘Better luck next time. Grade Anda.' ‘E' );
Writeln ('Tekan ENTER untuk melanjutkan ..');
Readln
END.
```

PERBEDAAN IF SEKUENSIAL DAN IF TERSARANG

Konsep CASE

Untuk masalah dengan dua kasus atau lebih, struktur CASE dapat menyederhanakan penulisan IF–THEN–ELSE yang bertingkat– tingkat sebagaimana pada contoh–contoh sebelumnya:

Bentuk Umum:

1. CASE

```
CASE (nama)
   kondisi1 : aksi1
   kondisi2 : aksi2
   kondisi3 : aksi3
    ................
    kondisiN : aksiN
    [otherwise aksiX]
endcase
```

CASE OF

Bentuk Umum:

```
CASE ungkapan OF
    Daftar case label1 : statemen1 ;
    Daftar case label2 : statemen2 ;
    Daftar case label3 : statemen3 ;
    …
    …
    …
    Daftar case label n : statemen n ;
END ;
```

Contoh program dengan bentuk CASE OF:

```
Var
    Nilai : char ;
Begin
    Write ('Nilai huruf yang didapat ?'); readln (Nilai);
    Case Nilai of
        'A' : writeln ('Sangat baik') ;
        'B' : writeln ('baik') ;
        'C' : writeln ('Cukup') ;
        'D' : writeln ('Kurang') ;
        'E' : writeln ('Gagal') ;
    End ;
End.
```

CASE OF ELSE

Pada struktur Case Of bila tidak ada kondisi yang terpenuhi berarti tidak ada statemen didalam lingkungan case of yang diproses. Dengan struktur case of else, bila tidak ada kondisi yang terpenuhi, maka statemen yang akan diproses didalam lingkungan case of adalah statemen yang ada di else.

Struktur CASE di atas menggantikan analisis kasus yang ekivalen dengan struktur IF–THEN–ELSE berikut ini:

```
If kondisi1 THEN
   aksi1
ELSE
   IF kondisi2 THEN
      aksi2
   ELSE
      IF kondisi3 THEN
        aksi3
      ELSE
      ..........
      IF kondisiNTHEN
         aksiN
      ELSE {otherwise}
         aksiX
      endif
    endif
  endif
endif.
```

Dengan struktur CASE, algoritma untuk masalah tersebut dapat dibuat menjadi lebih singkat sbb:

```
Algoritma KATA_UNTUK_ANGKA
Deklarasi
    A: integer {angka yang dibaca}
Deskripsi:
    Read(A)
    CASE (A)
       A=1 : write('satu')
       A=2 : write('dua')
       A=3 : write('tiga')
       A=4 : write('empat')
     otherwise write('angka yang dimasukkan salah')
endcase
```

Struktur Pengulangan

a. Terdapat 2 bagian struktur pengulangan:

- Kondisi pengulangan
- Badan (body) pengulangan

b. Struktur pengulangan biasanya disertai bagian:

- Inisialisasi
- Terminasi

c. Di dalam algoritma, pengulangan (repetition atau loop) dapat dilakukan sejumlah n kali, atau sampai kondisi pengulangan berhenti tercapai.

d. Perulangan harus berhenti.

Struktur While-Do

- Bentuk umum Algoritma:

```
while <kondisi> do
      aksi
endwhile
```

- Translasi dalam Bahasa Pascal:

```
while kondisi do
      aksi;
```

Struktur While-Do

- Aksi akan dilaksanakan berulang-ulang sepanjang ⟨kondisi⟩ boolean bernilai true
- Jika ⟨kondisi⟩ boolean bernilai false, badan pengulangan tidak dilakukan lagi (selesai).

Contoh Struktur While-Do

- Mencetak HALO sebanyak sepuluh kali menggunakan while–do

```
program cetak_halo;
var
  n:integer;
begin
  n:=1;
  while n<=10 do
      begin
          writeln('HALO');
          n:=n+1;
      end;
  readln;
end.
```

Contoh Struktur While-Do

- Output cetak HALO tersebut adalah:

```
HALO
HALO
HALO
HALO
HALO
HALO
HALO
HALO
HALO
HALO
```

Struktur While-Do

- Bentuk umum Algoritma:

```
repeat
        aksi
until <kondisi>
```

- Translasi dalam Bahasa Pascal:

```
repeat
        aksi;
until kondisi;
```

Struktur Repeat-Until

- Aksi di dalam badan pengulangan terus diulang sampai kondisi boolean bernilai true
- Dengan kata lain, jika kondisi berhenti masih false, pengulangan masih terus dilakukan.
- Karena pengulangan harus berhenti, di dalam badan pengulangan harus ada aksi yang mengubah nilai kondisi

Contoh Struktur Repeat-until

- Mencetak HALO sebanyak sepuluh kali menggunakan while-do

```
program cetak_halo;
var
    n:integer;
begin
    n:=1;
    repeat
        begin
            writeln('HALO');
            n:=n+1;
        end;
    until n>10;
    readln;
end.
```

- Struktur REPEAT-UNTIL memiliki makna yang mirip dengan WHILE-DO

a. Namun pada struktur REPEAT-UNTIL, aksi (atau sekumpulan aksi) dilaksanakan minimal satu kali, karena kondisi pengulangan diperiksa pada akhir struktur.

b. Pada struktur WHILE-DO kondisi pengulangan diperiksa pada awal struktur sehingga memungkinkan badan pengulangan sama sekali tidak dilaksanakan.

Struktur FOR

- FOR Menaik
- FOR Menurun
- Nested Loop / Pengulangan bersarang

Struktur FOR

- Struktur FOR digunakan untuk menghasilkan pengulangan sejumlah kali tanpa penggunaan kondisi apapun.
- Struktur ini menyebabkan aksi diulangi sejumlah kali (tertentu)

FOR Menaik

Dalam Bahasa Pascal:

```
for variabel:=a to b do
 aksi;
```

- variabel harus bertipe sederhana kecuali tipe real
- nilai_awalharus lebih kecil atau sama dengan nilai_akhir
- Bila nilai_awal lebih besar dari nilai_akhir maka badan pengulangan tidak dimasuki
- Pada awalnya, variabel dinisialisasi dengan nilai_awal. Nilai variabel tersebut otomatis bertambah satu setiap kali aksi pengulangan dimasuki, sampai akhirnya nilai variabel sama dengan nilai_akhir.
- Jumlah pengulangan yang terjadi adalah nilai_akhir – nilai_awal + 1

Contoh 1

```
  program cetak_halo;
var
   n : integer;
begin
   for n:=1 to 10 do
   writeln('HALO');
end.
```

Contoh 2

```
program cetak_angka;
var
        angka:integer;
begin
       for angka:=1 to 20 do
          writeln(angka);
end.
```

FOR Menurun

Dalam Bahasa Pascal:

```
for variabel:=a downto b do
        aksi;
```

- variabel harus bertipe sederhana kecuali tipe real
- nilai_akhirharus lebih besar atau sama dengan nilai_awal
- Bila nilai_akhir lebih kecil dari nilai_awal maka badan pengulangan tidak dimasuki
- Pada awalnya, variabel dinisialisasi dengan nilai_akhir. Nilai variabel tersebut otomatis berkurang satu setiap kali aksi pengulangan dimasuki, sampai akhirnya nilai variabel sama dengan nilai_awal.
- Jumlah pengulangan yang terjadi adalah nilai_awal – nilai_akhir + 1

Contoh 4

```
program cetak_n_angka_menurun;
var
	n,angka:integer;
begin
	write('masukkan angka: ');
	readln(n);
	for angka:=n downto 1 do
	writeln(angka);
end.
```

Contoh 5

```
program peluncuran_roket;
var
    angka:integer;
begin
    for angka:=100 downto 0 do
    writeln(angka);
    writeln('Roket Meluncur');
    readln;
end.
```

Nested Loop (Loop Bersarang)

- Di dalam sebuah struktur pengulangan diperbolehkan untuk membuat pengulangan lainnya.
- Dengan kata lain, pengulangan di dalam pengulangan.
- Pada operasi FOR dalam FOR, FOR yang paling dalam akan diselesaikan lebih dahulu, baru dieksekusi FOR yang di luar
- sehingga instruksi-instruksi yang didapat pada FOR yang paling dalam akan paling banyak dieksekusi.

Syarat Nested Loop

- Syarat yang harus diperhatikan untuk operasi FOR pengulangan tersarang
- Setiap pengulangan tidak boleh menggunakan variabel counter yang sama
- Antara pengulangan-pengulangan tersebut tidak boleh saling berpotongan (overlapping)

Salah

Benar

Loop yang Terpisah (Benar, tapi bukan Nested)

Contoh Notasi Algoritma

```
while [kondisi1] do
        [pernyataan]
        while [kondisi2] do
                [pernyataan]
        endwhile
endwhile
```

```
repeat
        [pernyataan]
        repeat
                [pernyataan]
        until [kondisi2]
until [kondisi1]
```

Program Cetak

```
program cetak_angka;
var
      i,j:integer;
begin
     for i:=1 to 5 do
          for j:=0 to 9 do
             begin
                  write(j,' ');
                  if j=9 then writeln;
             end;
       readln;
end.
```