

Logika dan Algoritma

Untuk Merancang Aplikasi Komputer

Indra Ava Dianta, S.Kom., M.T



YAYASAN PRIMA AGUS TEKNIK

Logika dan Algoritma

Untuk Merancang Aplikasi Komputer

Indra Ava Dianta, S.Kom., M.T

ISBN 978-623-6141-42-7 (PDF)



YAYASAN PRIMA AGUS TEKNIK



Logika dan Algoritma Untuk Merancang Aplikasi Komputer

Penulis:

Indra Ava Dianta, S.Kom.,M.T

ISBN : 978-623-6141-42-7 (PDF)

Editor:

Danang, S.Kom., M.T

Penyunting :

Edwin Zusrony S.E., M.M., M.Kom

Desain Sampul dan Tata Letak :

Nuris Dwi Setiawan, S.Kom., M.T

Penerbit :

Yayasan Prima Agus Teknik

Redaksi:

Jln Majapahit No 605 Semarang

Tlpn. (024) 6723456

Fax . 024-6710144

Email: penerbit_ypat@stekom.ac.id

Distributor Tunggal:

UNIVERSITAS STEKOM

Jln Majapahit No 605 Semarang

Tlpn. (024) 6723456

Fax . 024-6710144

Email: info@stekom.ac.id

Hak Cipta dilindungi Undang undang Dilarang memperbanyak karya Tulis ini dalam bentuk dan dengan cara apapun tanpa ijin tertulis dari penerbit

KATA PENGANTAR

Puji syukur kami panjatkan ke hadirat Tuhan Yang Maha Esa karena dengan rahmat, karunia serta taufik dan hidayah-Nya kami dapat menyelesaikan penyusunan buku berjudul **Logika dan Algoritma Untuk Merancang Aplikasi Komputer** dengan harapan untuk dapat dipergunakan oleh kalangan para akademisi

Tujuan utama penyusunan buku ini adalah untuk memudahkan mahasiswa dalam memahami dan menguasai dasar ilmu mengenai logika berfikir dalam pemrograman. Buku teks ini memberikan gambaran tentang pemahaman lebih dalam mengenai logika algoritma pemrograman sehingga memudahkan para mahasiswa untuk memahami apa yang di sampaikan pada buku ini serta bagi pendidik dapat menunjang untuk proses kegiatan belajar mengajar.

Semoga buku ini dapat dipahami bagi siapapun yang membacanya. Sekiranya buku telah disusun ini dapat berguna bagi kami sendiri maupun orang yang membacanya. Sebelumnya kami mohon maaf apabila terdapat kesalahan kata-kata yang kurang berkenan dan kami memohon kritik dan saran yang membangun demi perbaikan di masa depan.

Semarang, April 2021

Indra Ava Dianta, S.Kom., M.T

Penulis

DAFTAR ISI

Kata Pengantar	iii
Daftar Isi	iv
BAB I Pengantar Logika Algoritma Pemrograman	1
A. Logika	1
B. Algoritma	2
C. Pemrograman	5
D. Bahasa Pemrograman	7
E. Penyajian Algoritma	8
F. Flowchart	9
G. Struktur Dasar Pemrograman	14
BAB II Pengenalan Bahasa Pemrograman Pascal	18
A. Sejarah Pascal	18
B. Struktur Bahasa Pascal	19
C. Ciri – ciri Pascal	19
D. Jenis Perluasan	19
E. Unit	19
F. Variabel	20
G. Tipe Data	20
H. Macam – macam Operator	25
BAB III Input dan Output Pemrograman Pascal	30
A. Input dan Output	30
B. Judul Program	30
C. Deklarasi	31

D. Pernyataan	31
BAB IV Kondisi atau Percabangan.....	41
A. Pengenalan Kondisi.....	41
B. Pilihan Percabangan	41
BAB V Kondisi IF Tersarang.....	51
A. Rumus Umum Kondisi IF	51
B. Menggabungkan Kondisi	57
BAB VI Perulangan.....	59
A. Pengertian Perulangan.....	59
B. Perulangan FOR.....	59
C. Perulangan While Do	62
D. Repeat Until	64
E. Perbandingan Perulangan	65
BAB VII Label Go To.....	67
A. Pengertian Label.....	67
B. Pengertian Go To	67
C. Contoh Program Label	68
BAB VIII Fungsi Character dan Numerik.....	76
A. Fungsi Concat.....	76
B. Fungsi Copy	79
C. Fungsi Delete	79
D. Fungsi Insert.....	79
E. Fungsi Length.....	80

BAB IX Fungsi Text Color	87
A. Pengenalan	87
B. Text Color (Windows)	87
C. Text Background.....	88
D. Windows	88
BAB X Fungsi Array.....	92
A. Pengertian Array	92
B. Jenis Array	93
C. Pendefinisian Array.....	93
D. Sifat Array.....	93
E. Array Satu Dimensi.....	94
BAB XI Array Lanjutan	102
A. Pengertian Array	102
B. Array Dua Dimensi	112
C. Array Multi Dimensi	116
BAB XII Operasi File.....	118
A. Pengertian File/Berkas	118
B. Manipulasi File.....	118
C. Tahapan Operasi File	119
D. Jenis Operasi File	119
E. File Teks.....	120
BAB XIII Tipe Record Data	129
A. Pengertian Deklarasi Record.....	129
B. Pemakaian Record.....	130

C. Procedure.....	131
D. Function.....	132
BAB XIV Function Procedure	134
A. Pemrograman Modular.....	134
B. Struktur Modular.....	134
C. Keuntungan Pemrograman Modular	135
D. Dua Bentuk Pemrograman Modular	135
E. Procedur	135
F. Function.....	139
G. Fungsi Standar.....	141
H. Perbedaan Fungsi dan Procedure	145
I. Rekursif.....	145
Daftar Pustaka	148

BAB I

PENGANTAR LOGIKA ALGORITMA PEMROGRAMAN

TUJUAN INSTRUSIONAL

1. Menjelaskan tentang Algoritma.
2. Menjelaskan sifat - sifat suatu algoritma.
3. Penyajian algoritma ke dalam bentuk bahasa, psseudocode dan flowchart
4. Menjelaskan tentang flowcahart program
5. Alur program sederhana melalui flowchart

A. Logika

Logika adalah hasil pertimbangan akal pikiran yang diutarakan lewat kata dan dinyatakan dalam bahasa. Kata Logika berasal dari Yunani kuno $\lambda\acute{o}\gamma\omicron\varsigma$ (logos) yang juga merupakan salah satu cabang ilmu filsafat. Sebagai sebuah ilmu, logika disebut dengan logike episteme (bahasa Latin: logica scientia) atau ilmu logika (ilmu pengetahuan) yang mempelajari kecakapan untuk berpikir secara lurus, tepat, dan teratur.

1. Pengertian Logika Menurut Para Ahli

Para ahli telah mendefinisikan beberapa pengertian logika diantaranya :

a. Jan Hendrik Rapar, 1996

Pengertian logika adalah ajaran tentang berpikir yang secara ilmiah membicarakan bentuk pikiran itu sendiri dan hukum-hukum yang menguasai pikiran.

b. Ahmad Taufik Nasution, 2006

Logika merupakan ilmu dan kecakapan menalar, berpikir dengan tepat.

c. Jan Hendrik Rapar, 1996

Logika adalah suatu pertimbangan akal atau pikiran yang diatur lewat kata dan dinyatakan dalam bahasa.

- d. Soekadijo, 2008
Pengertian Logika menurut Soekadijo adalah suatu metode atau teknik yang diciptakan untuk meneliti ketepatan nenalar.
- e. William Alston, 2008
Pengertian logika adalah studi tentang penyimpulan, secara lebih cermat usaha untuk menetapkan ukuran-ukuran guna memisahkan penyimpulan yang sah dan tidak sah.

Pengertian algoritma sangat lekat dengan kata logika, yaitu kemampuan seorang manusia untuk berfikir dengan akal tentang suatu masalah dan menghasilkan sebuah kebenaran, dapat dibuktikan dan masuk akal. Dalam menyelesaikan suatu masalah logika sangat diperlukan. Logika identik dengan masuk akal dan penalaran. Penalaran adalah salah satu bentuk pemikiran. Definisi logika sangat sederhana yaitu cara berfikir untuk tujuan tertentu namun menurut aturan yang berlaku.

B. Algoritma

Secara etimologi, logika berasal dari kata Yunani Logos yang artinya adalah Kata, Ucapan, Pikiran secara utuh atau juga berarti Ilmu Pengetahuan (Kusumah, 1986).

Definisi : urutan langkah-langkah untuk memecahkan masalah yang disusun secara sistematis dan logis. Menurut Kamus Besar Bahasa Indonesia: algoritma adalah urutan logis pengambilan putusan untuk pemecahan masalah. Algoritma dibutuhkan untuk memerintah komputer mengambil langkah-langkah tertentu dalam menyelesaikan masalah.

Alasan mengapa algoritma banyak digunakan dalam pemrograman:

1. Pembuatan atau penulisan algoritma tidak tergantung pada bahasa pemrograman manapun.
2. Notasi algoritma dapat diterjemahkan ke dalam berbagai bahasa pemrograman.
3. Apapun bahasa pemrogramannya, output yang akan dikeluarkan sama karena algoritmanya sama.

C. Pengertian Algoritma menurut para ahli :

1. Seymour Lipschutz, Ph.D dan Marc Lipson, Ph.D.
Algoritma adalah suatu daftar langkah-demi-langkah yang terhingga dari instruksi-instruksi yang terdefiniskan dengan jelas yang dipakai untuk permasalahan tertentu.
2. David Bolton
Algoritma adalah deskripsi dari suatu prosedur yang berakhir dengan sebuah hasil.
3. Andrey Andreyevich Markov
Algoritma adalah hal umum untuk dipahami sebagai suatu keputusan yang tepat untuk mendefinisikan proses komputasi yang mengarahkan dari data awal hingga hasil yang diinginkan.
4. Stone dan Knuth
Algoritma adalah suatu seperangkat aturan yang tepat mendefinisikan urutan operasi hingga sedemikian rupa sehingga setiap aturan yang efektif, jelas hingga sedemikian rupa sehingga urutan berakhir dalam waktu yang terbatas.
5. Amikom Yogyakarta
Pengertian algoritma adalah sebuah bentuk instruksi dalam bentuk cara atau metode yang akan membantu kamu dalam menyelesaikan program dengan cara yang lebih sistematis.
6. Minsky
Algoritma adalah seperangkat urutan yang memberitahukan kepada kita dari waktu ke waktu, tepatnya bagaimana untuk bertindak.

D. Sejarah Algoritma

Ahli Sejarah Matematika menemukan asal kata algoritma tersebut yang berasal dari nama penulis buku arab yang terkenal yaitu Abu Ja'far Muhammad Ibnu Musa Al-Khuwarizmi. Al-Khuwarizmi dibaca orang barat menjadi Algorism. Al-Khuwarizmi menulis buku yang berjudul Kitab Al Jabar Wal-Muqabala yang artinya "Buku pemugaran dan pengurangan" (The book of restoration and reduction).

Dari judul buku itulah diperoleh akar kata "*Aljabar*" (Algebra). Perubahan kata dari algorism menjadi *algorithm* muncul karena kata algorism sering dikelirukan dengan arithmetic, sehingga akhiran *usm*

berubah menjadi *uthm*. Karena perhitungan dengan angka Arab sudah menjadi hal yang biasa, maka lambat laun kata *algorithm* berangsur-angsur dipakai sebagai metode perhitungan (komputasi) secara umum, sehingga kehilangan makna kata aslinya.

Kata algoritma, bukan sesuatu yang asing bagi kita. Penemunya adalah seorang ahli matematika dari Uzbekistan yang bernama Abu Abdullah Muhammad Ibnu Musa Al-Khwarizmi (770-840). Di literatur barat dia lebih terkenal dengan sebutan Algorizm. Panggilan inilah yang kemudian dipakai untuk menyebut konsep *algorithm* yang ditemukannya. Dalam bahasa Indonesia kita kemudian menyebutkannya sebagai algoritma.

Algoritma adalah kunci dari bidang ilmu komputer, karena banyak bidang di bawah ilmu komputer yang lahir berdasarkan konsep algoritma ini. Pada hakekatnya algoritma juga adalah kunci dari kehidupan kita. Contoh Cara membuat masakan (resep masakan) adalah juga sebuah contoh nyata dari algoritma.

Ada referensi lain yang menyebutkan namanya adalah Abu Ja'far Muhammad Ibnu Musa Al-Khwarizmi. Al-Khwarizmi menulis buku yang berjudul *Al Jabar Wal-Muqabala* yang artinya “Buku Pemugaran dan pengurangan” (The book of restoration and reduction).

E. Syarat Algoritma

1. Tingkat kepercayaannya tinggi (*reability*) Hasil yang diperoleh dari proses harus berakurasi tinggi dan benar.
2. Pemrosesan yang efisien (*cost rendah*)
3. Proses harus diselesaikan secepat mungkin dan frekuensi kalkulasi yang sependek mungkin.
4. Sifatnya general
5. Bukan sesuatu yang hanya untuk menyelesaikan satu kasus saja, tapi juga untuk kasus lain yang lebih
6. Bisa dikembangkan (*expandable*)
7. Haruslah sesuatu yang dapat kita kembangkan lebih jauh berdasarkan perubahan requirement yang ada.
8. Mudah dimengerti

9. Siapapun yang melihat, dia akan bisa memahami algoritma Anda. Susah dimengertinya suatu program akan membuat susah di-maintenance (kelola).
10. Portabilitas yang tinggi (portability)
11. Bisa dengan mudah diimplementasikan di berbagai platform komputer.
12. Precise (tepat, betul, teliti)
13. Efektif
14. Tidak boleh ada instruksi yang tidak mungkin dikerjakan oleh pemroses yang akan menjalankannya.
15. Harus terminate
16. Jalannya algoritma harus ada kriteria berhenti.
17. Output yang dihasilkan tepat.

F. Pemrograman

1. Definisi

Program adalah kumpulan instruksi-instruksi tersendiri yang biasanya disebut *source code* yang dibuat oleh programmer (pembuat program).

Program : Realisasi dari Algoritma.

Program = Algoritma + Bahasa

Pengertian program menurut para ahli :

- a. Amikom Yogyakarta
Program adalah kumpulan instruksi komputer, sedangkan metode dan tahapan sistematis dalam program adalah algoritma.
- b. Ema Utami
Program adalah bahasa pemrograman.
- c. Sukrisno
Program adalah kata, ekspresi, atau pernyataan yang disusun dan dirangkai menjadi satu kesatuan prosedur, yang berupa urutan langkah, untuk menyelesaikan masalah yang diimplementasikan dengan menggunakan bahasa pemrograman sehingga dapat dieksekusi oleh komputer.

- d. Anwar Harjono
Program adalah urutan instruksi untuk menjalankan suatu komputasi.
- e. Saifuddin Anshari
Program adalah daftar terinci mengenai acara dan usaha yang akan dilaksanakan.
- f. Sunarto, S.Kom
Program adalah sekumpulan instruksi yang diwujudkan dalam bentuk bahasa, kode skema, ataupun bentuk lain, yang apabila digabungkan dengan media yang dapat dibaca dengan komputer akan mampu membuat komputer bekerja untuk melakukan fungsi-fungsi khusus, termasuk persiapan dalam merancang instruksi-instruksi tersebut.
- g. Wiryanto Dewobroto
Program adalah hasil penyusunan detail langkah-langkah solusi (algoritma) masalah tersebut.
- h. Binanto
Menurut Binanto, pemrograman dapat diartikan dalam beberapa hal, sebagai berikut:
- Mendeskripsikan instruksi-instruksi tersendiri yang biasanya disebut sebagai Source Code yang dibuat oleh programmer.
 - Mendeskripsikan suatu keseluruhan bagian dari software yang executable.
 - Program merupakan himpunan atau kumpulan instruksi tertulis yang dibuat oleh programmer atau suatu bagian executable dari suatu software.
 - Pemrograman berarti membuat program komputer.
 - Pemrograman merupakan suatu kumpulan urutan perintah ke komputer untuk mengerjakan sesuatu. Perintah-perintah ini membutuhkan suatu bahasa tersendiri yang dapat dimengerti oleh komputer.

2. Langkah Pembuatan Program

- a. Mendefinisikan masalah
- Kondisi awal, yaitu *input* yang tersedia.

- Kondisi akhir, yaitu output yang diinginkan.
 - Data lain yang tersedia.
 - Operator yang tersedia.
 - Syarat atau kendala yang harus dipenuhi.
- b. Buat Algoritma dan Struktur Cara Penyelesaian
Jika masalahnya kompleks, maka dibagi ke dalam modul-modul
 - c. Menulis program
Pilihlah bahasa yang mudah dipelajari, mudah digunakan, dan lebih baik lagi jika sudah dikuasai, memiliki tingkat kompatibilitas tinggi dengan perangkat keras dan platform lainnya.
 - d. Mencari Kesalahan
 - Kesalahan sintaks (penulisan program).
 - Kesalahan pelaksanaan: semantik, logika, dan ketelitian..
 - e. Uji dan Verifikasi Program
 - f. Dokumentasi Program
 - g. Pemeliharaan Program

G. Bahasa Pemrograman (Algoritma)

1. Struktur Penulisan Algoritma

Setiap Algoritma akan selalu terdiri dari tiga bagian yaitu :

- a. Judul (Header)
Judul adalah bagian teks algoritma yang digunakan sebagai tempat mendefinisikan nama dengan menentukan apakah teks tersebut adalah program, prosedur, fungsi.
- b. Kamus
Kamus adalah bagian teks algoritma sebagai tempat untuk mendefinisikan :
 - Nama type
 - Nama konstanta
 - Nama variabel
 - Nama fungsi
 - Nama prosedur

Kamus	
{Nama type, hanya untuk type yang bukan type dasar}	
type jam : <hh,mm,ss : integer >	{Type jam terdiri dari 3 masukan yaitu “hh” sebagai jam, “mm” sebagai menit dan “ss” sebagai detik}
{Nama konstanta, harus menyebutkan type dan nilai }	
constant phi : real = 3,14159	
constant nama : string = ‘Alex’	
constant benar : boolean = true	
{Nama Informasi, menyebutkan type}	
x,y : integer	{suatu nilai yang bertipe bilangan bulat}
NMax : real	{nilai maksimum yang bertipe bilangan real}
Nama : string	{suatu nilai yang merupakan kumpulan character}
P : point	{suatu nilai pada bidang kartesian}
Cari : Boolean	{suatu nilai logika}

Gambar 1.1 Kamus

c. Algoritma

Algoritma adalah bagian inti dari suatu algoritma yang berisi instruksi atau pemanggilan aksi yang telah didefinisikan

Algoritma	
input (c,d)	{menerima masukan 2 bilangan c dan d}
if c < d then	{operasi kondisional}
e ← a + b	{e di <i>assignment</i> oleh nilai a dan b}
else	
e ← a – b	
output (e)	{hasil keluaran berupa bilangan e}

Gambar 1.2 Algoritma

H. Penyajian Algoritma

Bentuk penyajian untuk algoritma dibagi menjadi 3 (tiga) bentuk penyajian, yaitu :

1. Algoritma dengan struktur Bahasa Indonesia

Sifat: Umum

- a. Tidak menggunakan simbol atau sintaks dari suatu bahasa pemrograman.
- b. Tidak tergantung pada suatu bahasa pemrograman.
- c. Notasi-notasinya dapat digunakan untuk seluruh bahasa manapun.

Contoh : Menghitung rata-rata tiga buah data

Algoritma dengan struktur bahasa Indonesia :

- a. Baca bilangan a, b, dan c
- b. Jumlahkan ketiga bilangan tersebut
- c. Bagi jumlah tersebut dengan 3
- d. Tulis hasilnya

2. Algoritma dengan Pseudocode

Penyajian algoritma dengan *pseudocode* berarti menggunakan kode yang mirip dengan kode pemrograman yang sebenarnya. Pseudocode lebih rinci dari English/Indonesia *Structure*.

Contoh (1) : 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. Algoritma dengan *Flowchart*

Flowchart adalah penggambaran secara grafik dari langkah-langkah dan urutan prosedur dari suatu program. Flowchart menolong analis dan programmer untuk memecahkan masalah kedalam segmen-segmen yang lebih kecil dan menolong dalam menganalisis alternatif-alternatif lain dalam pengoperasian.

Kegunaan:

- a. Untuk mendesain program
- b. Untuk merepresentasikan program

Maka, flowchart harus dapat merepresentasikan komponen-komponen dalam bahasa pemrograman.

4. Flowchart

a. Relationship

Flowchart dapat memberikan gambaran yang efektif, jelas, dan ringkas tentang prosedur *logic*. *Teknik penyajian yang* bersifat grafis jelas akan lebih baik daripada uraian-uraian yang bersifat teks khususnya dalam menyajikan logikalogika yang bersifat kompleks.

b. Analysis

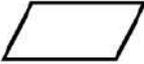
Dengan adanya pengungkapan yang jelas dalam model atau chart, maka para pembaca dapat dengan mudah melihat permasalahan atau memfokuskan perhatian pada area-area tertentu sistem informasi.

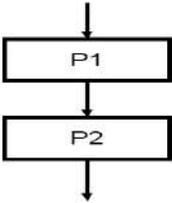
c. Communication

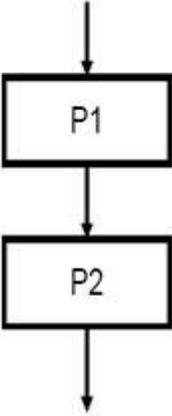
Karena simbol-simbol yang digunakan mengikuti suatu standar tertentu yang sudah diakui secara umum, maka flowchart dapat merupakan alat bantu yang sangat efektif dalam

mengkomunikasikan logika suatu masalah atau dalam mendokumentasikan logika tersebut.

Tabel 1.1 Lambang flowchart

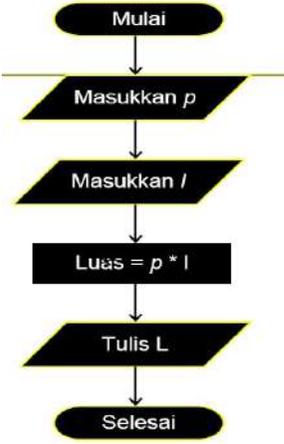
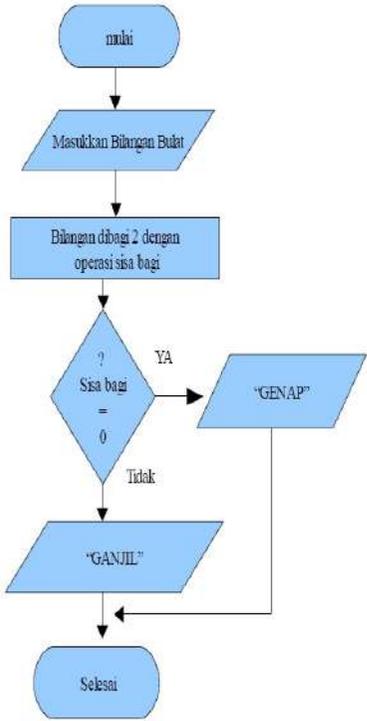
Keterangan	Lambang
Mulai/selesai <i>(terminator)</i>	
Aliran data	
<i>Input/Output</i>	
Proses	

Keterangan	Lambang
<i>Sequence Process</i>	

Keterangan	Lambang
<i>Sequence Process</i>	

Keterangan	Lambang
Percabangan (<i>Decision</i>)	
Pemberian nilai awal suatu variabel (<i>Preparation</i>)	
Memanggil prosedur/fungsi (<i>Call</i>)	
Keterangan	Lambang
Percabangan (<i>Decision</i>)	
Pemberian nilai awal suatu variabel (<i>Preparation</i>)	
Memanggil prosedur/fungsi (<i>Call</i>)	
Dokumen / Multi dokumen	
Harddisk	

Berikut contoh flowchart dalam penyelesaian masalah pemrograman.

<p>a. Problem: Menghitung Luas persegi panjang Algoritma: 1. Masukkan panjang (p) 2. Masukkan lebar (l) 3. Hitung luas (L), yaitu panjang kali lebar 4. Cetak luas (L)</p>	 <pre> graph TD Start([Mulai]) --> Input1[/Masukkan p/] Input1 --> Input2[/Masukkan l/] Input2 --> Process[Luas = p * l] Process --> Output[/Tulis L/] Output --> End([Selesai]) </pre>
<p>b. Problem: Menentukan Bilangan ganjil atau Genap</p>	 <pre> graph TD Start([mulai]) --> Input[/Masukkan Bilangan Bulat/] Input --> Process[Bilangan dibagi 2 dengan operasi sisa bagi] Process --> Decision{? Sisa bagi = 0} Decision -- YA --> Output1[/GENAP/] Decision -- Tidak --> Output2[/GANJIL/] Output1 --> End([Selesai]) Output2 --> End </pre>

I. Struktur Dasar Algoritma

1. Struktur Runtunan (Sequence Proses)

Sebuah runtunan terdiri dari satu atau lebih 'instruksi'. Tiap-tiap instruksi dilaksanakan secara berurutan sesuai dengan urutan penulisannya; sebuah instruksi baru bisa dilaksanakan setelah instruksi sebelumnya selesai dilaksanakan.

2. Struktur Pemilihan (Selection Proses)

Pada struktur ini, jika kondisi terpenuhi maka salah satu aksi akan dilaksanakan dan aksi yang ke dua diabaikan. Kondisi adalah persyaratan yang dapat dinilai benar atau salah sehingga akan memunculkan 'aksi' yang berbeda dengan 'kondisi' yang berbeda.

CONTOH :

Menentukan bilangan terbesar diantara 3 bilangan:

'if' $x > y$ 'then'

'if' $x > z$ 'then'

tulis x sebagai bilangan terbesar

'else'

tulis z sebagai bilangan terbesar

'else'

'if' $y > z$ 'then'

tulis y sebagai bilangan terbesar

'else'

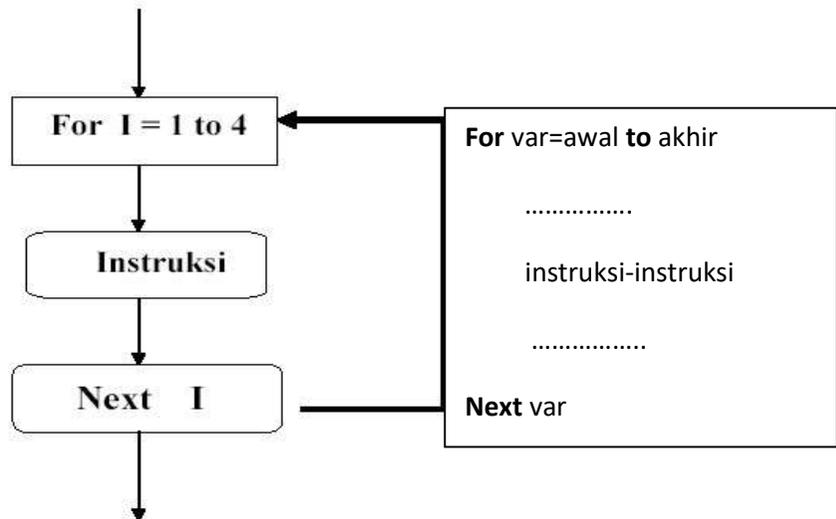
tulis z sebagai bilangan terbesar

3. Struktur Pengulangan (Iteration Proses)

Digunakan untuk program yang pernyataannya akan dieksekusi berulang-ulang. Instruksi dikerjakan selama memenuhi suatu kondisi

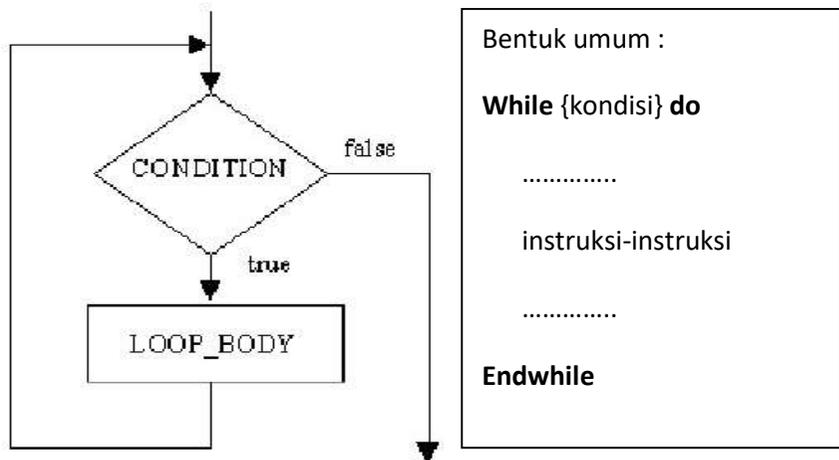
tertentu. Jika syarat (kondisi) masih terpenuhi maka pernyataan (aksi) akan terus dilakukan secara berulang.

a. For-next



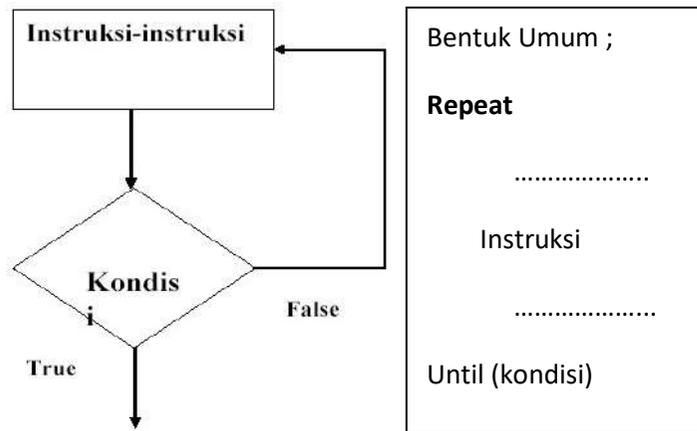
Gambar 1.1 Perulangan For - Next

b. While-do



Gambar 1.3 Perulangan While do

c. Repeat-until



Gambar 1.3 Perulangan Repeat Until

Contoh penerapan dalam program:

Algoritma Cetak_Angka

{mencetak 1, 2, ..., 8 ke piranti keluaran}

Deklarasi :

K: integer

Deskripsi :

K 1 {inisialisasi}

while k <= 8 do

 write (k)

 k k + 1

endwhile

Contoh 2:

Algoritma Cetak_Angka

{mencetak 1, 2, ..., 8 ke piranti keluaran}

Deklarasi :

K: integer

Deskripsi :

K ← 1 {inisialisasi}

repeat

write (k)

k ← k + 1

until k > 8

Latihan bab 1:

Buatlah struktur flowchart dalam kehidupan sehari hari, setiap mahasiswa cukup membuat 1 flowchart saja.

TUJUAN INSTRUSIONAL

1. Menjelaskan jenis-jenis data sederhana.
2. Menjelaskan pengertian integer, real, boolean dan char.
3. Memahami contoh deklarasi dari tipe data integer, real character dan boolean.
4. Menjelaskan pengertian Konstanta, variabel dan ekspresi serta dapat memberikan contoh untuk konstanta, variabel dan ekspresi.
5. Membedakan jenis type data yang akan digunakan dan dapat

A. Sejarah Pascal

Merupakan pengembangan dari bahasa ALGOL 60. Tahun 1960, beberapa ahli komputer bekerja untuk mengembangkan bahasa ALGOL, salah satunya adalah Dr. Niklaus Wirth dari *Swiss Federal Institute of Technology (ETH-Zurich)*, yang merupakan anggota grup yang membuat ALGOL.

Tahun 1971, Dr. Niklaus Wirth menerbitkan suatu spesifikasi untuk highly-structured language (bahasa tinggi yang terstruktur) yang menyerupai ALGOL yang dinamai dengan PASCAL (seorang filsuf dan ahli matematika dari Perancis). Diperkenalkan pada Komputer CDC 6000 (Control Data Corporation) Versi bahasa Pascal: UCSD (University of California at San Diego) Pascal, Ms (Microsoft) Pascal, Apple Pascal, Turbo Pascal, dsb.

Turbo Pascal yang dibuat oleh Borland Inc. adalah versi yang paling banyak digunakan karena menggunakan Compiler untuk menterjemahkannya dan juga mengikuti standard bahasa Pascal yang dibuat oleh Nicklaus Wirth dan K. Jensen. Pascal merupakan bahasa pemrograman tingkat tinggi (high level language) dan terstruktur (Structured Programming language).

1. Struktur Bahasa Pascal

Struktur program pada Pascal

- a. Bagian Judul Program
- b. Bagian Deklarasi
 - deklarasi label
 - definisi konstanta
 - definisi tipe
 - deklarasi variable
 - deklarasi prosedur
 - deklarasi fungsi
- c. Bagian Program utama/Pernyataan

2. Ciri-Ciri Pascal

- a. Berurutan
- b. Blok dengan batas-batas yang jelas
- c. Satu pintu masuk dan satu pintu keluar pada blok pemilihan dan pengulangan

3. Jenis Perluasan

Ada beberapa file yang digunakan dalam TURBO PASCAL standar yang perlu diketahui, yaitu :

- *.PAS : File source kode (naskah program) Pascal
- *.BAK : File cadangan (backup) dari source kode Pascal
- *.EXE : File file hasil compile source kode Pascal.
- *.TPU : File unit Pascal, yang berisi procedure dan function baik dari TP atau yang dibuat sendiri
- *.TPL : File Library, kumpulan file-file unit Pascal
- *.TP : File konfigurasi Pascal

B. Unit

Unit merupakan kumpulan dari konstanta, label, type, variabel, procedure dan function yang siap untuk dipakai untuk kegunaan proses tertentu dalam program Pascal. Beberapa unit standar yang disediakan adalah SYSTEM, DOS, CRT, PRINTER, OVERLAY, GRAPH. Kesemuanya ini sudah menyatu dalam sebuah file library bernama TURBO.TPL, kecuali untuk unit file TURBO3.TPU, GRAPH.TPU, dan GRAPH3 . TPU adalah berdiri sendiri.

C. Variabel

1. Variabel merupakan penyimpanan data yang bersifat sementara di memori komputer (RAM).
2. Aturan pemberian nama variabel
 - a. Kombinasi huruf, angka dan garis bawah “ _ ”
 - b. Diawali dengan huruf/abjad
 - c. Maximal 63 karakter
 - d. Tidak boleh ada spasi kosong
 - e. Tidak boleh memakai nama perintah pascal

D. Tipe Data

Macam – macam Tipe data

Ada 6 (enam) kelompok tipe data pada pascal, yaitu :

1. Tipe sederhana, dibagi jadi 2 tipe :
 - a. Tipe Ordinal, dibagi lagi;
 - Tipe bil. Bulat
 - Tipe boolean
 - Tipe karakter
 - Tipe terbilang
 - Tipe subjangkauan
 - b. Tipe real
2. Tipe string
3. Tipe terstruktur.
Dibagi menjadi lima (5) tipe, yaitu ;
 - a. Tipe larik
 - b. Tipe rekaman / record
 - c. Tipe objek
 - d. Tipe himpunan
 - e. Tipe berkas
4. Tipe pointer
 - a. Tipe prosedural
 - b. Tipe objek

E. Tipe Bilangan Bulat

Sesuai dengan namanya, tipe bilangan Bulat digunakan untuk menyimpan bilangan bulat.

Tabel 2.1 *macam-macam tipe bil. Bulat Pascal*

Tipe	Jangkauan	Ukuran
Shortint	-128...127	8 bit
Integer	-32768...32767	16 bit
Longint	-2147483648... 2147484647	32 bit
Byte	0...255	8 bit
Word	0...65535	16 bit

F. Tipe Bilangan Bulat

1. Tipe boolean

Tipe boolean adalah tipe yang hanya dapat bernilai true (benar) atau false (salah).

Tabel 2.2 Macam-macam tipe boolean

TIPE DATA	UKURAN
Boolean	1 byte
Bytebool	1 byte
Wordbool	2 byte (1 word)
Longbool	4 byte (2 word)

2. Tipe karakter

- a. Tipe karakter digunakan untuk menyimpan data alfanumeris / karakter, seperti : “A”...”Z”, “a”...”b”, “0”...”9”,”\$”,”@”,...
- b. Untuk memberi nilai pada variabel bertipe karakter, dapat menggunakan beberapa cara yaitu :
- c. menuliskan karakter di dalam tanda petik tunggal contoh :
- d. `ch := 'A';`

- e. menuliskan tanda # diikuti tanda dengan nomor ASCII dari karakter yang ingin dituliskan.

Contoh : `ch := #65` {sama artinya dgn `ch := 'A'`};

Mengkonversikan nomor ASCII ke karakter menggunakan fungsi `chr`.

contoh : `ch := chr(65)`; { sama artinya dgn `ch := 'A' }`

Kebalikan dari fungsi `chr` adalah `ord`.

contoh : `x := ord('A')`; { x akan bernilai 65 }

3. Tipe subjangkauan

Tipe subjangkauan memungkinkan untuk mendeklarasikan tipe yang berada pada jangkauan tertentu. Pendeklarasian tipe subjangkauan dilakukan dengan menuliskan batas bawah dan batas atas dari jangkauannya. Contoh :

Type

`bulan = 1..12;`

Mendeklarasikan tipe bulan yang memiliki jangkauan dari 1 sampai 12. Dengan demikian bila jika mempunyai variabel bertipe bulan, seperti contoh berikut :

Var

`januari : bulan ;`

Anda tidak bisa memberikan nilai kurang dari 1 atau lebih dari 12.

Contoh : `januari := 1;`

4. Tipe terbilang

Tipe terbilang memungkinkan anda memberi nama pada beberapa nilai tertentu. Sebagai contoh :

type

`tipehari := (Minggu, Senin, Selasa, Rabu, Kamis, Jumat, Sabtu);`

Memberi nama Minggu pada 0, senin pada 1, selasa pada 2, dan seterusnya. Dengan pendeklarasian tipehari seperti contoh diatas, anda tidak perlu menggunakan angka 0, 1, sampai dengan 6 untuk mempresentasikan hari.

Contoh : *var*

hari : tipehari;

Anda dapat menuliskan contoh pernyataan berikut :

hari := Minggu ;

hari := Senin ;

5. Tipe real

Tipe real digunakan untuk menyimpan bilangan real / pecahan.

Tabel 2.3 *Macam – macam tipe real*

TIPE	JANGKAUAN	DIGIT	UKURAN
Real	$2.9 * 10^{-39} .. 1.7 * 10^{38}$	11 – 12	6 byte
Single	$1.5 * 10^{-45} .. 3.4 * 10^{38}$	7 – 8	4 byte
Double	$5.0 * 10^{-324} .. 1.7 * 10^{308}$	15 – 16	8 byte
Extended	$3.4 * 10^{-4932} .. 1.1 * 10^{4932}$	19 – 20	10 byte
Comp	$-2^{63} + 1 .. 2^{63} - 1$	19 – 20	8 byte

6. Tipe string

Tipe string digunakan untuk menyimpan data yang berupa untaian karakter, seperti ‘pascal’, ‘algoritma’, de es be. Untuk mendeklarasikan string, digunakan kata kunci string. Contoh :

Var

kalimat : string ;

Pemberian nilai pada string dilakukan dengan meletakkan untaian karakter diantara tanda petik tunggal. Contoh :

kalimat := ‘pemrograman algoritma’;

7. Tipe Larik

Tipe larik memungkinkan untuk mendeklarasikan kumpulan variabel yang bertipe sama. Bentuk umum pendeklarasian larik :

Var

nama_larik : array [batas_bawah..batas_atas] of tipe larik ;

Contoh Anda ingin membuat delapan variabel bertipe longint. Tanpa menggunakan larik, mungkin mendeklarasikan variabel tersebut dengan cara berikut :

Var

a1, a2, a3, a4, a5, a6, a7, a8 : longint;

Dengan larik kita bisa menyederhanakan deklarasi kedelapan variabel diatas menjadi :

Var

a : array [1..8] of longint;

8. Tipe rekaman

Memungkinkan Anda menggabungkan beberapa variabel yang tipenya tidak harus sama. Untuk mendeklarasikan rekaman, digunakan kata kunci *record*. Contoh :

type

tkaryawan = record

nama : string ;

alamat : string ;

gaji : string ;

end;

Pemberian nilai pada variabel bertipe rekaman dilakukan dengan menyebutkan nama variabel rekaman diikuti tanda titik dan variabel didalam rekaman. Sebagai contoh, jika terdapat variabel karyawan yang betipe tkaryawan :

var

karyawan : tkaryawan ;

Dapat menuliskan pernyataan berikut ;

karyawan.nama := 'ronaldo';

karyawan.alamat := 'Jl. Selindung lama';

karyawan.gaji := 1000000;

9. Tipe himpunan

Digunakan untuk menyimpan kumpulan nilai (disebut juga anggota himpunan) yang bertipe sama. Sebagai contoh :

type

himpunankarakter = set of char;

Mendeklarasikan tipe himpunanankarakter sebagai himpunan dari karakter.

Contoh

Deklarasi variabel bertipe himpunan :

var

vokal : himpunankarakter;

huruf : himpunankarakter;

Pemberian nilai pada tipe himpunan dilakukan dengan menuliskan anggota himpunan

dalam kurung siku ([dan]). Contoh :

vokal := ['A', 'I', 'U', 'E', 'O'];

huruf := ['A'..'Z'];

10. Tipe pointer

Pointer adalah variabel yang menunjuk lokasi memori tertentu. Pendeklarasian pointer dilakukan dengan cara menambahkan tanda ^ didepan tipe pointer. Contoh :

Var

p1 : ^integer ;

p2 : ^double;

G. Macam – macam operator

Ada tujuh operator pada pascal, yaitu ;

1. Operator pemberian nilai

:=

Contoh:

A := 12;

B := 'Halo';

C := 3.14;

2. Operator aritmetik

3. Operator Aritmetik Tunggal

a. *+* (Contoh : *x := +y;*)

b. *-* (Contoh : *x := -y;*)

4. Operator Aritmetik Biner

a. *+* (Penjumlahan)

X := y + z;

b. *-* (Pengurangan)

X := a - b - c - d;

- c. * (Perkalian)
X := 5 * 9 * 3.14;
 - d. / (Pembagian)
X := a / b;
 - e. Div (Pembagian bilangan bulat)
Z := 10 div 2;
 - f. Mod (Sisa Pembagian / modulus)
Z := y mod k;
5. Operator manipulasi bit
Not, And, Or, Xor

Tabel 2.4 Operator Manipulasi Bit

A	B	Not A	Not B	A and B	A or B	A xor B
0	0	1	1	0	0	0
0	1	1	0	0	1	1
1	0	0	1	0	1	1
1	1	0	0	1	1	0

6. Operator boolean
Not, And, Or, Xor

Tabel 2.5 Operator Boolean

A	B	Not A	Not B	A and B	A or B	A xor B
false	False	true	true	false	false	false
false	True	true	false	false	true	true
true	False	false	true	false	true	true
true	True	false	false	true	true	false

7. Operator pembandingan

Tabel 2.6 Operator Pembandingan

=	Sama dengan
<>	Tidak sama dengan
<	Kurang dari
>	Lebih dari
<=	Kurang dari atau sama dengan
>=	Lebih dari atau sama dengan

Hasil dari operasi ini adalah tipe boolean (True/False)

8. Operator himpunan

Tabel 2.7 Operator Himpunan

Operator	Operasi
+	Union
-	Selisih
*	Interseksi
in	Anggota dari

Sebagai contoh : $a := b + c$;

Menggabungkan semua anggota himpunan B dan C ke dalam himpunan A. Jika A, B dan C bertipe Set Of Char dan isi himpunan B adalah ['P','Q'] dan isi himpunan C adalah ['R'] maka isi himpunan A adalah ['P','Q','R'].

9. Operator string

Pascal hanya mengenal satu macam *operator string*, yaitu penggabungan. Operator ini digunakan untuk menggabungkan dua atau lebih string. Operator ini menggunakan tanda '+'.
Contoh

Contoh

```
S := 'STEKOM' + 'Semarang';
```

```
Writeln(S);
```

10. Derajat Operator

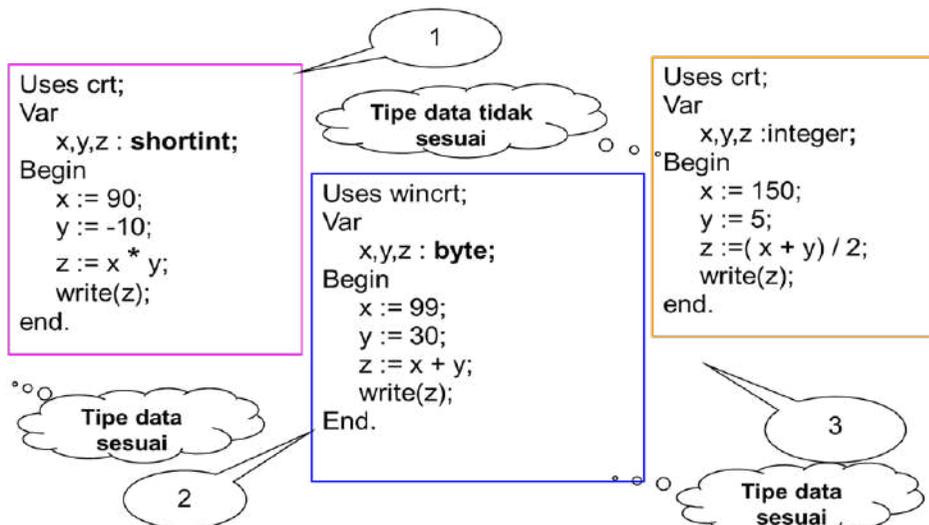
Berapa hasil dari $x = 2 + 3 * 4$?

Contoh Derajat Operator

- $X := 4 + 10 * 2;$
- $X := 6 - 2 * 4 + 10;$
- $X := 8 * (5 + 2);$
- $X := 4 / 2 + 10 * 4;$
- $X := 3 * 5 - 6 + 4;$

Latihan 1

Tentukan Listing program mana yang akan menghasilkan nilai yang benar :



Lanjutan 2

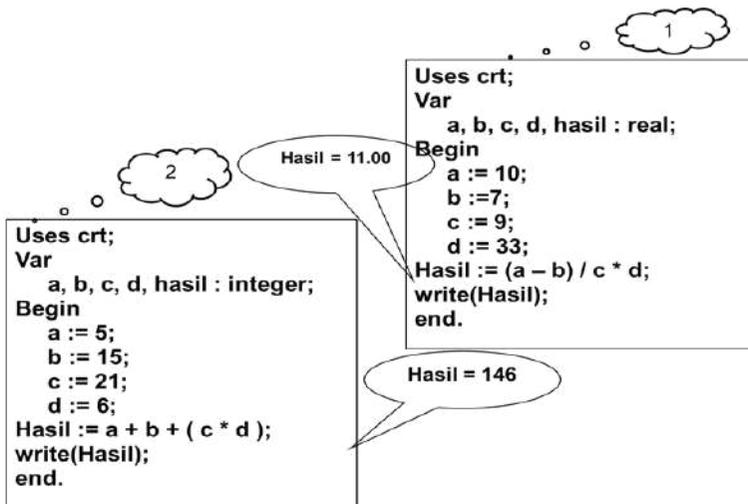
Jika diketahui x adalah variabel bertipe *integer*, dari nilai-nilai berikut, manakah yang dapat diberikan pada x?

- a. 10.1
- b. 200
- c. 32799
- d. -35600
- e. 0
- f. 80.55
- g. 21474846

A,C,D,F,G salah nilai
tidak sesuai dengan tipe

Latihan 3

Tentukan nilai dari variabel hasil :



BAB III

Input dan Output Pemrograman Pascal

TUJUAN INSTRUSIONAL

1. Membuat program input dan output.
2. Menjelaskan mengenai statemen read dan readln dan dapat menjelaskan perbedaan dari kedua statemen itu.
3. Menjelaskan mengenai statemen write dan writeln dan dapat menjelaskan perbedaan dari kedua statemen itu.
4. Membuat program input dan output dengan menggunakan perintah write dan read

A. Input dan Output Pascal

Secara ringkas, struktur suatu program Pascal dapat terdiri dari 3 bagian:

1. Bagian Judul Program
2. Bagian Deklarasi
 - a. Deklarasi tipe data (TYPE)
 - b. Deklarasi variabel (VAR)
 - c. Deklarasi konstanta (CONST)
 - d. Deklarasi label (LABEL)
 - e. Deklarasi sub-program (PROCEDURE dan FUNCTION)
3. Bagian Program Utama Statement

B. Judul Program

Judul program ini digunakan untuk memberi nama program dan sifatnya optional. Jika ditulis harus terletak pada awal dari program dan diakhiri dengan titik koma (;).

Contoh penulisan judul program :

1. PROGRAM latihan;
2. PROGRAM latihan(input,output);
3. PROGRAM lat_1;
4. PROGRAM lat_satu(output);

C. Deklarasi

Bagian ini menjelaskan / memperkenalkan secara rinci semua data yang akan digunakan pada suatu program. Dalam penulisannya tidak boleh sama dengan kata-kata cadangan (reserved words) dan selalu diakhiri dengan titik koma (;).

1. Deklarasi label

Deklarasi label digunakan Jika pada penulisan program akan menggunakan statemen GOTO (untuk meloncat ke suatu statement tertentu).

2. Deklarasi konstanta

Deklarasi ini digunakan untuk mengidentifikasi data yang nilainya sudah ditentukan dan pasti, tidak dapat dirubah dalam program.

3. Deklarasi tipe

Deklarasi ini digunakan untuk menyebutkan tipe setiap data yang akan digunakan pada program Pascal. Tipe data menentukan jangkauan nilai yang mungkin dari data yang digunakan

4. Deklarasi variabel/perubah

Deklarasi ini berisi data-data yang bisa berubah-ubah nilainya di dalam program. Deklarasi variabel harus di letakkan setelah deklarasi tipe (jika ada).

5. Deklarasi prosedur dan Fungsi

Program dapat dibagi menjadi beberapa agian/subprogram, yang terdiri dari satu program utama dan satu / lebih program bagian (bisa berupa prosedur / fungsi). Deklarasi prosedur/ fungsi terletak pada subprogram yang menggunakannya

D. Pernyataan

Bagian ini adalah bagian yang akan terproses dan terdapat dalam suatu blok yang diawali dengan BEGIN dan diakhiri dengan END (penulisan END diikuti dengan tanda titik).

Bagian ini berisi pernyataan / statemen yang merupakan instruksi program.

Setiap statemen diakhiri dgn tanda titik koma (;).

Bentuk umumnya adalah sbb :

```
begin
  ...
  statemen;
  statemen;
  ...
end.
```

E. Aturan Program

Setiap akhir pernyataan diakhiri titik koma (;), kecuali untuk nama label.

Akhir program diberi titik (.).

F. Komentar

Komentar adalah keterangan yang diberikan untuk keperluan dokumentasi. Tidak menghasilkan tindakan (tidak mempengaruhi jalannya program). Boleh menggunakan tanda : { ini komentar } atau (* ini komentar *)

G. Identifier

Merupakan identifier yang didefinisikan sendiri oleh pemrogram. Pemrogram mempunyai kebebasan untuk menentukan nama identifiernya, dengan syarat nama tersebut tidak sama dengan identifier standar dan reserved word yang akan dibahas lebih lanjut. Hal ini untuk mencegah kesalahan yang bisa timbul akibat tumpang tindih identifier dalam program

Syarat Identifier/Variabel :

1. Variabel merupakan penyimpan data yang bersifat sementara di memori komputer (RAM).
2. Aturan pemberian nama variabel
 - a. Diawali huruf
 - b. Tidak boleh ada spasi/blank
 - c. Tidak boleh menggunakan reserved word
 - d. Tidak boleh menggunakan simbol khusus, kecuali underscore(tanda bawah)
 - e. Panjang maximal 63 character

H. Statement-statementen pada Pascal

Reserved Word adalah kata-kata baku yang digunakan dalam program dan mempunyai bentuk serta kegunaan tertentu yang telah

didefinisikan oleh Pascal. Reserved Word tidak boleh didefinisikan kembali oleh pemakai, sehingga tidak dapat digunakan sebagai pengenal (Identifier). Dalam bahasa pemrograman Pascal, beberapa Reserved Word tersebut adalah :

1. AND
2. DOWNTO
3. IN
4. OF
5. STRING
6. ASM ELSE
7. INHERITED
8. OR
9. THEN
10. ARRAY
11. END
12. BEGIN
13. PROCEDURE
14. TYPE

I. Perintah Dasar

1. Write ('Text/tulisan',Variabel);
Perintah untuk menampilkan atau cetak dilayar monitor tanpa pindah baris
2. WriteLn('Text/Tulisan',Variabel);
Perintah untuk menampilkan/cetak dilayar monitor lalu pindah baris kebawah
3. Read(Variabel);
Perintah untuk menginput/mengisi data tanpa pindah baris
4. ReadLn(Variabel);
Perintah untuk menginput/mengisi data lalu pindah baris
5. Read/Readln
Digunakan untuk memasukkan (input) data lewat keyboard ke dalam suatu variabel.

Sintaks:

READ/READLN(V);

Keterangan :

V = variabel.

READ = pada statemen ini posisi kursor tidak pindah ke baris selanjutnya.

READLN = pada statemen ini posisi kursor akan pindah ke baris selanjutnya setelah di input.

6. Readkey

Untuk pembacaan sebuah karakter dari keyboard. Tipe data yang dihasilkan adalah char.

Sintaks: READKEY;

7. Write/Writeln

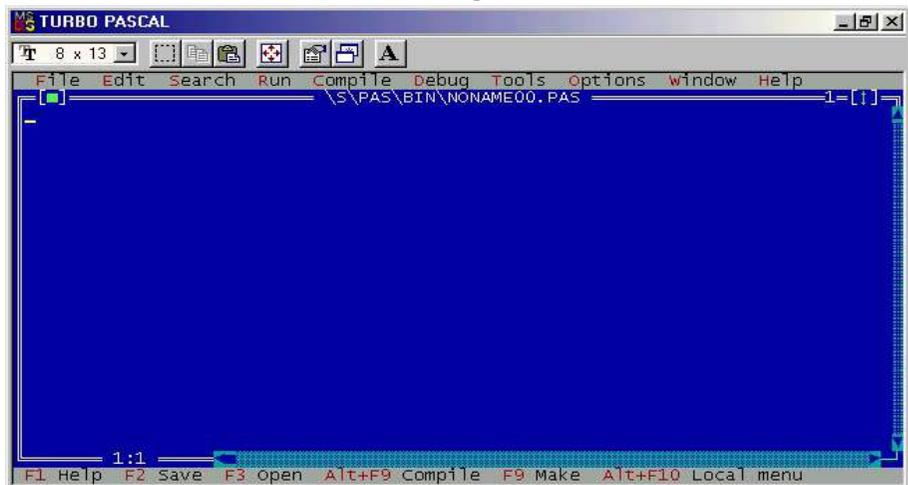
Digunakan untuk menampilkan isi dari suatu nilai variabel di layar.

Sintaks: WRITE/WRITELN(V);

Keterangan :

V = variabel.

WRITE/WRITELN = sama dengan READ/READLN.



Gambar 3.1 Tampilan Turbo Pascal

J. Tombol Hot Key

Tombol HOT KEY (kombinasi tombol/tombol yang dipergunakan untuk melaksanakan perintah tertentu) saat anda berada dalam text editor TURBO PASCAL adalah:

F1 : Help, minta bantuan keterangan dari T.P. (bila ada filenya)
 F2 : simpan naskah program ke diskette
 F3 : muat / panggil suatu file dari diskette
 F4 : Run bentuk 'Goto cursor'
 F5 : Zoom window text editor
 F6 : Pindah ke window yang aktif (Text Editor atau Watch)
 F7 : Run bentuk 'Trace intro'
 F8 : Run bentuk 'Step over'
 F9 : Compile bentuk 'Make'
 F10 : Kembali ke menu utama
 Alt + F5 : Melihat hasil eksekusi program terakhir kali.
 Alt + F9 : Lakukan proses Compile naskah program
 Alt + R : aktifkan menu 'Run'
 Alt + X : Keluar dari T.P.
 Ctrl+ F9 : Eksekusi/RUN naskah program

Ctrl + KB : Awal Blok
 Ctrl + KK : Akhir Blok
 Shift + Anak Panah : Pengeblokan
 Ctrl + KH : Batal Blok
 Ctrl + Insert : Copy Blok
 Shift + Insert : Paste
 Ctrl + KC : Copy+Paste Blok
 Ctrl + KV : Memindah Blok
 Shift + Del : Cut
 Ctrl + Del : Hapus Blok
 Ctrl + Y : Hapus satu baris

Contoh Program-1

Kode Program

```

Program Menampilkan_pesan;
use
begin
    write('Namaku Indra');
    write('Rumahku Demak');
    readln;
end.
  
```

Gambar 3.2 Koding Program 1

Hasil Program



Gambar 3.3 Hasil Program 1

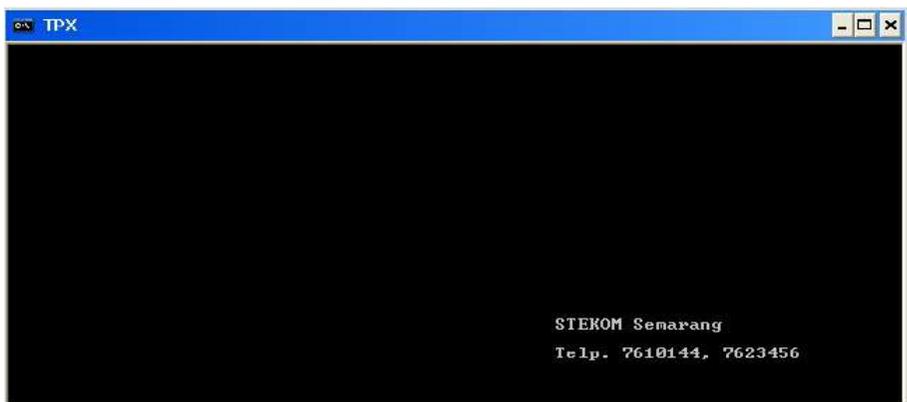
Contoh Program-2

Kode Program

```
File Edit Search Run Compile Debug Tools Op
[ ] SATU-C.PAS
Program Menampilkan_Pesan_atau_teks_dengan_gotoxy;
uses crt;
var
  X,Y : integer;
Begin
  clrscr;
  GotoXY(50,20);
  Write('STEKOM Semarang');
  GotoXY(50,22);
  Write('Telp. 7610144, 7623456');
  readln;
end.
```

Gambar 3.4 Koding Program 2

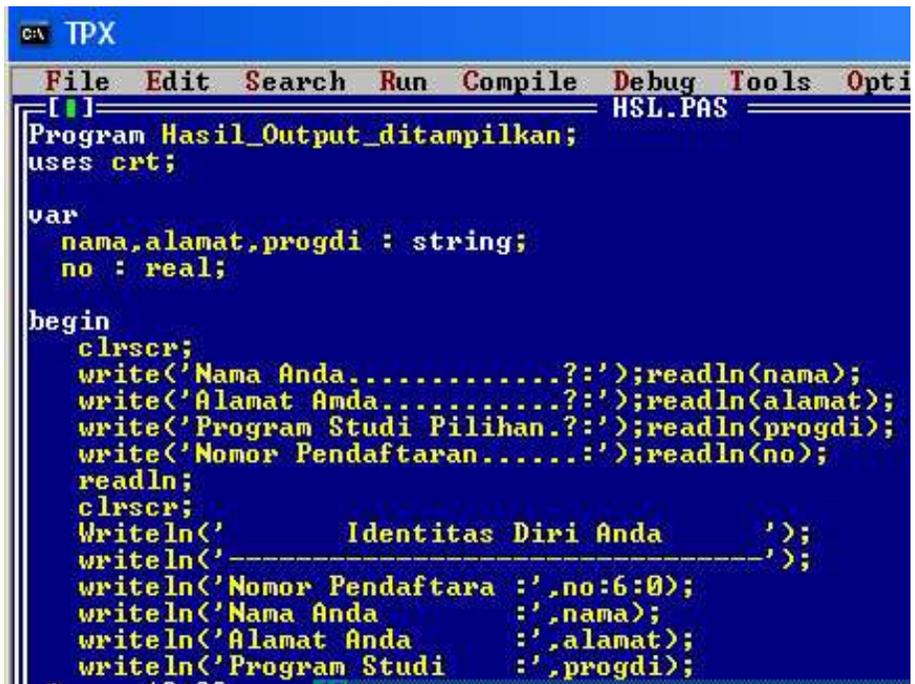
Hasil Program



Gambar 3.5 Hasil Program 2

Contoh Program 4

Kode Program



```
TPX
File Edit Search Run Compile Debug Tools Opti
[ ] HSL.PAS
Program Hasil_Output_ditampilkan;
uses crt;

var
  nama,alamat,progdi : string;
  no : real;

begin
  clrscr;
  write('Nama Anda.....?:');readln(nama);
  write('Alamat Anda.....?:');readln(alamat);
  write('Program Studi Pilihan.?:');readln(progdi);
  write('Nomor Pendaftaran.....:');readln(no);
  readln;
  clrscr;
  writeln('          Identitas Diri Anda          ');
  writeln('-----');
  writeln('Nomor Pendaftara :',no:6:0);
  writeln('Nama Anda       :',nama);
  writeln('Alamat Anda     :',alamat);
  writeln('Program Studi   :',progdi);
```

Gambar 3.6 Koding Program 3

Hasil Program :



```
TPX
          Identitas Diri Anda
-----
Nomor Pendaftara : 1234
Nama Anda       : Budi
Alamat Anda     : Semarang
Program Studi   : Manajemen
```

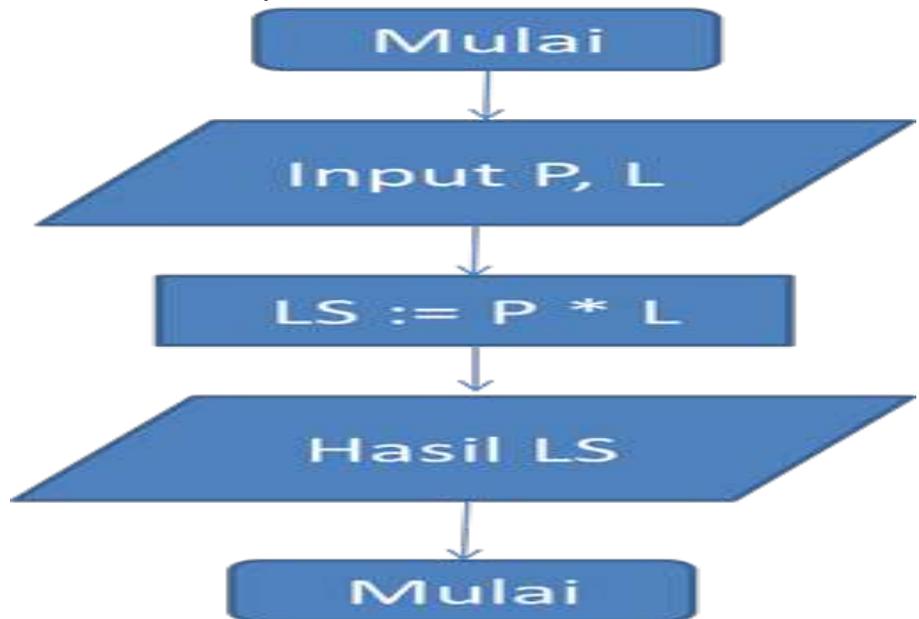
Gambar 3.7 Hasil Program 3

Contoh Program 5 :

Mencari Luas 4 Persegi panjang, bila di ketahui panjang dan lebarnya, buatlah dalam algoritma bahasa indonesia, algoritma flowchart, pseudocode

Algoritma Dalam Bahasa Indonesia

- a. Baca Panjang segi 4
- b. Baca Lebar segi 4
- c. Kalikan nilai panjang dan lebarnya
- d. Tuliskan hasilnya



Gambar 3.8 Flowhart Persegi Panjang

Algoritma Dalam Flowchart

- e. Algoritma Dalam Pseudocode
Input (P, L)
 $Ls := P * L$
Output LS

```
[[ ]] EMPAT.PAS
Program Hitung_luas_4_Persegi_Panjang;

uses crt;
var
  P   : integer;
  L   : integer;
  LS  : integer;

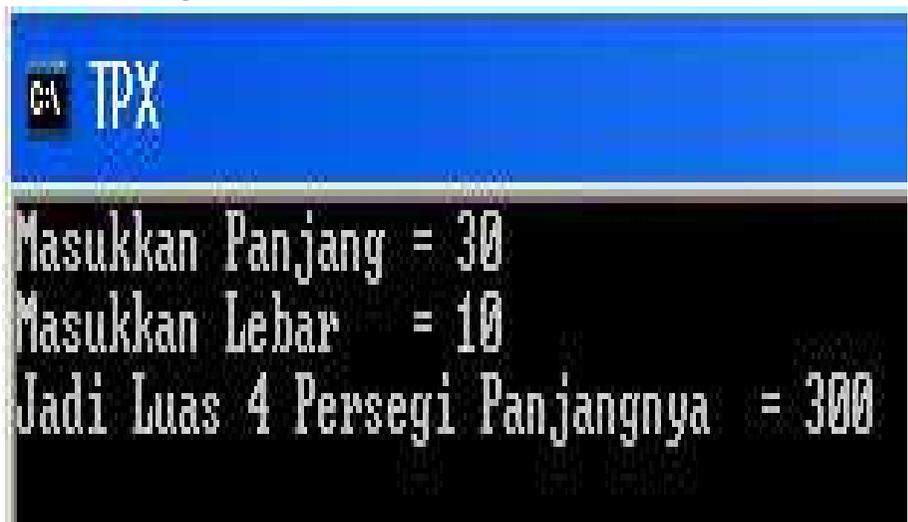
Begin
  clrscr;
  Write('Masukkan Panjang = ');readln(P);
  Write('Masukkan Lebar   = ');readln(L);

  LS := P*L;

  Writeln('Jadi Luas 4 Persegi Panjangnya = ',Ls);
  readln;
End.
```

Gambar 3.9 Koding Program 5

Hasil Program :



```
TPX
Masukkan Panjang = 30
Masukkan Lebar   = 10
Jadi Luas 4 Persegi Panjangnya = 300
```

Gambar 3.10 Hasil Program 5

Latihan :

1. Buat program untuk mengkonversikan derajat Celcius ke Reamur dan Fahrenheit.
Input : Celcius
Output : Reamur dan Fahrenheit
2. Buatlah program untuk menghitung Luas segi tiga
Input : Alas , Tinggi
Output : Luas segi tiga
3. Buatlah program untuk menghitung Luas dan keliling Lingkaran
Input : Jari-jari(r) / atau Diameter (D)
Output : Luas dan keliling
4. Buat Program Pascal dengan aturan Input dua buah bilangan,
Tampilkan hasil :
Penjumlahan, Pengurangan, Perkalian dan
Pembagian dari kedua bilangan tersebut

TUJUAN INSTRUSIONAL

1. Menyebutkan macam-macam statemen penyeleksian kondisi.
2. Menjelaskan bentuk umum dari statemen kondisi IF dan CASE.
3. Membuat contoh program sederhana dengan menggunakan statemen kondisi IF dan CASE

A. Pengenalan Kondisi/Percabangan

Tidak semua baris dalam program akan di eksekusi. Suatu aksi akan dilakukan bila memenuhi persyaratan atau kondisi tertentu. Kondisi umumnya berupa **Boolean** Kondisi *Boolean* adalah suatu ekspresi relasional yang bernilai *true* atau *false* bergantung pada nilai masing-masing operasi yang terlibat didalamnya. Penentuan kondisi **boolean** dan aksi yang dilakukan bergantung pada jumlah pilihan atau kasus yang terdapat pada masalah tersebut apakah terdapat satu pilihan, dua pilihan, ataukah terdiri atas banyak pilihan:

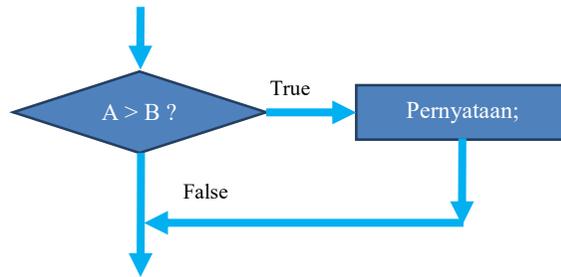
B. Pilihan Tunggal

Pada percabangan jenis ini, hanya ada satu kondisi yang menjadi syarat untuk melakukan satu buah atau satu blok instruksi. Format umum dari algoritma percabangan dengan satu kondisi adalah sebagai berikut:

```
IF kondisi THEN
Instruksi
ENDIF
```

Arti dari format di atas, jika “kondisi” bernilai benar atau tercapai, maka aksi dikerjakan. Sedangkan jika bernilai salah, maka instruksi tidak dikerjakan dan proses langsung keluar dari percabangan dan kembali lagi ke kondisi awal.

Bentuk Flowchart dari pilihan tunggal :



Gambar 4.1 Percabangan Tunggal

Contoh Program

```
IF_1A.PAS:2
Program Cari_Bilangan_Genap_dan_Ganjil;
uses crt;

Var
  Bil : Integer;
  Ket : String;

Begin
  clrscr;
  Write('Masukkan Sebuah Blangan:');Readln(Bil);

  If (bil mod 2) = 0 then
    Ket:='Genap';

    Writeln('Keterangan    = ',ket);
    Readln;

End.
```

Gambar 4.2 Program Ganjil Genap 1

Contoh lain dari penggunaan algoritma percabangan untuk satu kondisi adalah sebagai berikut:

```
if A > B then
  write (A)
end if
```

Instruksi di atas artinya instruksi akan menampilkan nilai A hanya jika kondisi “A lebih besar daripada B” bernilai benar. Jika bernilai

salah, maka tidak ada aksi yang akan dilakukan atau proses langsung keluar dari percabangan (end if).

Berikut ini kami berikan contoh beberapa contoh program algoritma percabangan untuk satu kondisi menggunakan macam-macam bahasa pemrograman. Berikut ini adalah contoh untuk program menggunakan bahasa Pascal adalah sebagai berikut:

```
uses crt;
var
jeniskelamin:char;
begin
clrscr;
writeln('Jenis Kelamin : ');
writeln('L untk laki-laki, P untuk perempuan');
writeln('Jenis kelamin anda: ');readln(jeniskelamin);
if(jeniskelamin = 'l') then writeln('Laki-laki');
if(jeniskelamin = 'p') then writeln('Perempuan');
readkey;
end
```

Contoh lainnya dari program percabangan untuk satu kondisi pada suatu program menggunakan bahasa C++ adalah sebagai berikut:

```
#include <iostream.h>
int main (){
int nilai;
char a;
cout<<"Masukkan Nilai Anda:";
cin>>nilai;
if (nilai>60){
cout<<"Selamat Anda Lulus!!";
}
cin>>a;
return 0;
}
```

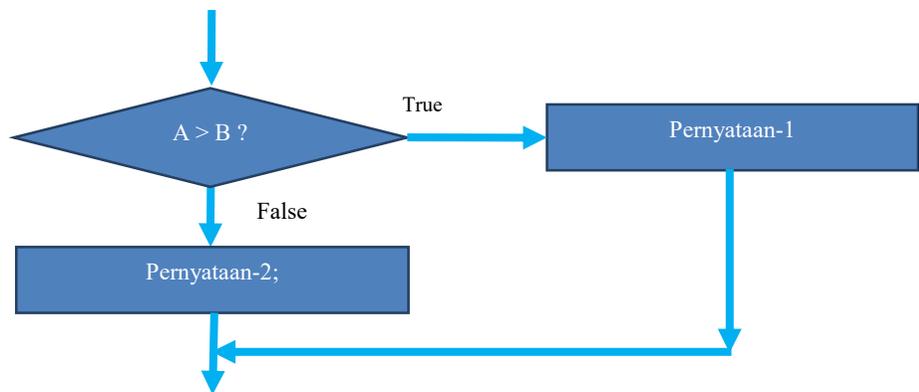
C. Pilihan Dua Kasus

Pada percabangan jenis ini, ada dua kondisi yang menjadi syarat untuk dikerjakannya salah satu dari dua instruksi. Kondisi ini bisa

bernilai benar atau salah. Bentuk umum dari percabangan dengan dua kondisi adalah sebagai berikut:

```
IF kondisi THEN
instruksi 1
ELSE
instruksi 2
ENDIF
```

Arti dari format di atas, jika “kondisi” bernilai benar maka instruksi 1 yang akan dikerjakan. Sedangkan jika bernilai salah), maka instruksi 2 yang akan dikerjakan. Perbedaannya dengan percabangan untuk satu kondisi terletak pada adanya dua instruksi untuk dua kondisi, yaitu kondisi bernilai benar dan kondisi bernilai salah.



Bentuk Flowchart If Ganda :

Gambar 4.3 Percabangan ganda

```

Program Cari_Bilangan_Genap_dan_Gasal;
uses crt;

Var
    Bil : Integer;
    Ket : String;

Begin
    clrscr;
    Write('Masukkan Sebuah Blangan:');Readln(Bil);

    If (bil mod 2) = 0 then
        Ket:='Genap'
    Else
        Ket:='Gasal';

        Writeln('Keterangan      = ',ket);
        Readln;

End.

```

Gambar 4.4 Program Genap Ganjil

```
Program Nilai_Huruf;
uses crt;
Var Nilai : Real;
    Ket   : String;
Begin
    clrscr;
    Write('Masukkan nilai :');Readln(Nilai);

    If nilai>=60 then
        Ket:='Lulus'
    Else
        Ket:='Tdk lulus';

        Writeln('Keterangan    = ',ket);
        Readln;

End.
```

Gambar 4.5 Program Grade Lulus

D. Pilihan Majemuk/Bertingkat

Algoritma percabangan untuk tiga kondisi atau lebih adalah bentuk pengembangan dari dua macam algoritma percabangan yang telah dibahas sebelumnya. Bentuk Umum Dimana kondisi -1 di uji jika hasil True jalankan Pernyataan-1, Jika False.Uji kondisi ke-2, jika hasil True jalankan Pernyataan-2, jika hasilnya False.Uji kondisi ke-3, jika hasil True Jalankan Pernyataan-3, jika false jalankan Pernyataan-4, dst

Karena itu, percabangan jenis ini akan memiliki banyak variasi. Secara umum, format percabangannya dapat dituliskan sebagai berikut :

```
If Kondisi-1 Then
    Pernyataan-1
Else
    If Kondisi-2 Then
```

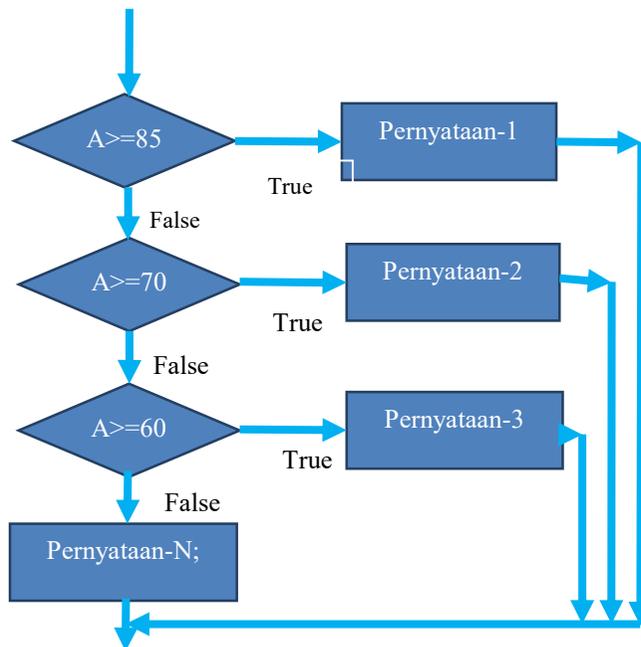
```

    Pernyataan-2
Else
    If Kondisi-3 Then
        Pernyataan-3
    Else
        Pernyataan-N;

```

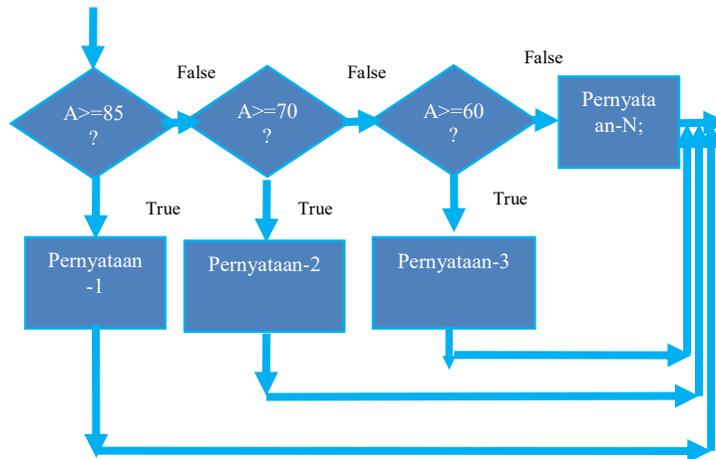
Maksud dari algoritma di atas, instruksi 1 akan dikerjakan jika “kondisi 1” bernilai benar. Jika bernilai salah, pemeriksaan dilanjutkan ke “kondisi 2”. Jika “kondisi 2” bernilai benar, maka instruksi 2 dikerjakan. Jika tidak, pemeriksaan dilanjutkan pada kondisi-kondisi lainnya. Pemeriksaan ini akan terus dilakukan terhadap semua kondisi yang ada. Jika tidak ada satu pun kondisi yang bernilai benar maka pernyataan yang dikerjakan adalah instruksi 3 atau instruksi (n+1) pada percabangan lebih dari 3 kondisi.

Flowchart If Bertingkat :



Gambar 4.6 Percabangan Mejemuk 1

Bentuk Lain Flowchart If Bertingkat :



Gambar 4.7 Percabangan Mejemuk 2

Bila dalam pernyataan lebih dari satu maka penulisan harus diantara Begin – End. Sebagai berikut :

If Kondisi Then

Begin

Pernyataan-1a;

Pernyataan-1b;

End

Else

Begin

Pernyataan-2a;

Pernyataan-2b;

End

End If

```

Program Nilai_Huruf;
uses crt;
Var Nilai : Real;
    Ket    : String;
Begin
  clrscr;
  Write<'Masukkan nilai : '>;Readln(Nilai);
  If nilai>=86 then
    Ket:='Sangat Baik'
  Else
    If nilai>=70 then
      Ket:='Baik'
    Else
      If nilai>=55 then
        ket:='Cukup'
      Else
        Ket:='Remidi';

  Writeln<'Keterangan = ',ket>;
  Readln;
End.

```

Gambar 4.8 Program Nilai Remidi

E. Pilihan Percabangan “Case of....2”

Selain menggunakan format yang dijelaskan pada poin 3, percabangan 3 kondisi atau lebih bisa juga menggunakan format “Case Of”. Format ini memiliki kegunaan yang sama, tetapi format ini digunakan untuk memeriksa data yang bertipe karakter atau integer. Struktur Case Of mempunyai suatu ungkapan logika yang disebut dengan selector dan sejumlah statement yang diawali dengan suatu label permasalahan (case label) yang mempunyai tipe yang sama dengan selector. Statement yang mempunyai case label yang bernilai sama dengan case label yang bernilai sama dengan nilai selector akan diproses sedang statemen yang lainnya tidak. Secara umum format penulisannya adalah sebagai berikut:

```

CASE <Variabel> OF
  Alternatif1 :
    Begin
      <Pernyataan-1a>;
      <Pernyataan-1b>;
    End;
  Alternatif2 : <Pernyataan-2>;
  Alternatif3 : <Pernyataan-3>;
  Alternatif4 : <Pernyataan-4>;
Else

```

End;
Alternatif Jika Bukan Pilihan diatas

```
NONAME00.PAS
Program Pemilihan_Dengan_Case_Of;
uses Crt;
var
  nilai : integer;
  ket   : string;
begin
  clrscr;
  Write<'Masukkan Nilai Ujian Anda : '>;readln<nilai>;

  Case nilai Of
    86..100 : Ket:='Sangat Baik';
    70..85  : Ket:='Baik';
    55..69  : Ket:='Cukup';
    0..54   : Ket:='Jelek';
  Else
    Writeln<'Input Salah..isi dengan 0-100'>;
  End;

  Write<'Hasil nilai Anda....: ',Ket>;
  readln;
End.
```

Gambar 4.9 Program Nilai Baik dan Jelek

Latihan :

Buatlah program untuk menampilkan hari dari senin sampai minggu menggunakan fungsi case of..

TUJUAN INSTRUSIONAL

1. Menjelaskan bentuk umum dari statemen kondisi IF majemuk
2. Menjelaskan bentuk statemen kondisi IF di dalam IF atau IF tersarang
3. Menjelaskan bentuk statemen kondisi CASE di dalam IF atau dalam bentuk IF tersarang
4. Mmembuat contoh program sederhana dengan menggunakan statemen kondisi IF dan CASE. Serta IF didalam IF atau Case di dalam IF

A. Rumus Umum Kondisi IF

Struktur if dibentuk dari pernyataan if dan sering digunakan untuk menyeleksi suatu kondisi tunggal. Bila proses yang diseleksi terpenuhi atau bernilai benar, maka pernyataan yang ada di dalam blok if akan diproses dan dikerjakan. Bentuk umum struktur kondisi if untuk satu kasus:

If (kondisi)
pernyataan

Pernyataan dilaksanakan jika dan hanya jika kondisi yang diinginkan terpenuhi, jika tidak program tidak memberikan hasil apa-apa. Struktur if dibentuk dari pernyataan if dan sering digunakan untuk menyeleksi suatu kondisi tunggal. Bila proses yang diseleksi terpenuhi atau bernilai benar, maka pernyataan yang ada di dalam blok if akan diproses dan dikerjakan. Bentuk umum struktur kondisi if untuk satu kasus : Pernyataan dilaksanakan jika dan hanya jika kondisi yang diinginkan terpenuhi, jika tidak program tidak memberikan hasil apa-apa.

Kondisi digunakan untuk menentukan pengambilan keputusan. Pernyataan dapat berupa pernyataan tunggal atau majemuk. Kondisi

akan dijalankan hanya jika kondisi bernilai benar ($\neq 0$). Perhatikan Diagram flowchart IF:



Catatan :

C++ selalu memperlakukan nilai tidak sama dengan nol sebagai benar, dan nilai nol sama dengan salah.

Gambar 5.1 Diagram Flowchart

Contoh Program: IF dipakai untuk menyeleksi seseorang boleh atau tidak boleh menonton film di bioskop. Kondisi yang digunakan , jika usia sudah 17 tahun.



Gambar 5.2 Program Ambil keputusan dengan IF

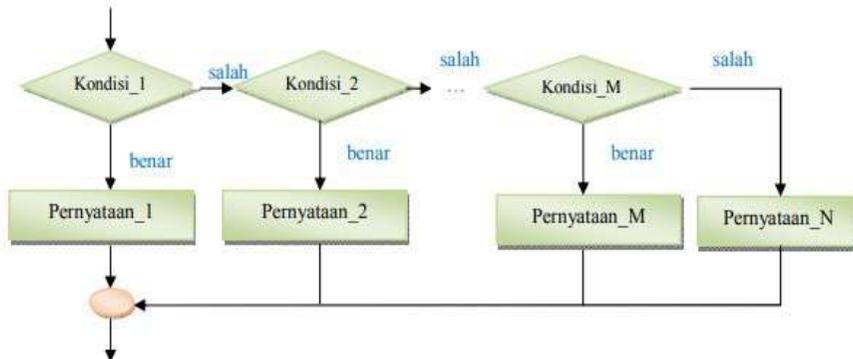
Dalam struktur IF kondisi, dikenalkan juga apa yang disebut dengan bentuk IF bersarang atau sering disebut *nested if*. Bentuk ini digunakan untuk pemilihan pernyataan dari sejumlah kemungkinan tindakan. Penyeleksian dilakukan secara bertingkat. Struktur percabangan *if bersarang / nested if* merupakan struktur *if* yang paling kompleks, karena merupakan perluasan dan kombinasi dari berbagai struktur *if* lainnya. Konsep dari percabangan ini adalah terdapat Struktur *If* yang berada didalam Struktur *If* lainnya. Artinya dalam pernyataan *If* bersarang jika kondisi *If* yang paling luar (paling atas) bernilai benar, maka kondisi *If* yang berada didalamnya baru akan dilihat (di cek). Misalnya untuk menentukan kode hari, atau kode bulan dan sebagainya, bisa memanfaatkan pernyataan bentuk *If* di dalam *If*. Bentuk umumnya sebagai berikut :

```

if (kondisi_1)
    pernyataan_1;
else if (kondisi_2)
    pernyataan_2;
else if (kondisi_3)
    pernyataan_3;
....
else if (kondisi_M)
    pernyataan_M;
else
    pernyataan_N;

```

Bentuk pernyataan di atas dapat digambarkan dengan flowchart berikut ini:



Gambar 5.3 Diagram Fungsi If di dalam If

Terdapat banyak variasi dari **nested IF**, tergantung kode program yang ingin kita rancang. Salah satunya adalah sebagai berikut:

```

IF (kondisi 1) THEN
  begin
    (kode program 1)
    IF (kondisi 1.1) THEN
      begin
        (kode program 1.1)
      end;
    end
  ELSE
    begin
      (kode program 2)
    end;
  
```

Contoh kondisi IF di dalam IF (nested IF):

```

IF (kondisi 1) THEN
  begin
    (kode program 1)
    IF (kondisi 1.1) THEN
      begin
        (kode program 1.1)
        IF (kondisi 1.1.1) THEN
          begin
            (kode program 1.1.1)
          end;
        end;
      end;
    end;
  end;
  
```

```

        end;
    end;
end
ELSE
begin
    (kode program 2)
    IF (kondisi 2.1) THEN
    begin
        (kode program 2.1)
    end;
end;
end;

```

```

P_IF_IF2.PAS
Program Seleksi_Di_Dalam_Seleksi;
uses Crt;
var
    Gaber,Gapok,Tunj    : real;
    Status              : char;
    Anak                : Byte;
    Gol                 : Byte;
    Nama                : String;
begin
    clrscr;
    Writeln('          PT. Ogah Rugi ');
    Writeln('Jl. Untung Terus No. 99 Semarang ');
    Writeln('Daftar Gaji Dan Tunjangan Anak ');
    Writeln('-----');
    Writeln(' ');
    Write('Nama Anda                               :');readln(nama);
    Write('Golongan Anda(1/2/3/4)                   :');readln(gol);
    Write('Perkawinan <B>-Bujang <K>-Kawin :');readln(status);
    Write('Jumlah Anak                                   :');readln(anak);
    Writeln(' ');
    Case Gol Of
        1 : Gapok := 1000000;
        2 : Gapok := 1500000;
        3 : Gapok := 2000000;
        4 : Gapok := 2500000;
    Else
        Writeln('Input Data Salah...!');
    End;

    If (Status='B') Then
        Tunj := 0
    Else
        If Anak=0 Then
            Tunj := 0
        Else
            If Anak=1 Then
                Tunj := 150000
            Else
                If Anak=2 Then

```

```

        Tunj :=200000
    Else
        Tunj :=300000;

    Gaber := Gapok + Tunj;

    Writeln('Nama Anda           :',Nama);
    Writeln('Golongan Anda       :',Gol);
    Writeln('Status perkawinan :',Status);
    Writeln('Jumlah Anak           :',Anak);
    Writeln('Gaji Pokok           :',Gapok:7:2);
    Writeln('Tunjangan Anak      :',Tunj:6:2);
    Writeln('Gaji Bersih         :',Gaber:7:2);
    readln;
End.

```

Gambar 5.4 Program Gaji Pokok

```

Program Seleksi_Di_Dalam_Seleksi;
uses Crt;
var
  biaya : real;
  tujuan : char;
  kelas : char;
begin
  clrscr;
  writeln('Kereta Api Ekspres "Super Cepat" ');
  writeln('Dari Stasiun "Tawang Semarang" ');
  writeln('Tujuan Jakarta, Bandung, Surabaya');
  writeln('-----');
  writeln('');
  write('Nota Tujuan (A)-Jakarta (B)-Bandung (C)-Surabaya :');
  readln(tujuan);
  write('Kelas (U)-VIP/Eksklusif (B)-Bisnis (E) Ekonomi :');
  readln(kelas);

  If (Tujuan='C') Then
    If (kelas='U') Then
      Biaya := 200000
    Else
      If (kelas='B') Then
        Biaya := 100000
      Else
        Biaya := 50000;

  writeln('Bayar Tiket      ',Biaya:7:2);
  readln;
End.

```

```

If (tujuan='A') Then
  If (kelas='U') Then
    Biaya := 300000
  Else
    If kelas='B' Then
      Biaya := 250000
    Else
      Biaya := 150000;

If (Tujuan='B') Then
  If (kelas='U') Then
    Biaya := 200000
  Else
    If (kelas='B') Then
      Biaya := 150000
    Else
      Biaya :=100000;

```

Gambar 5.5 Program Pembayaran Kereta Api

B. Menggabungkan Kondisi :

Terkadang kita perlu menggabungkan dua buah kondisi. Misalkan dari dua kondisi yang kita tentukan, dua-duanya harus bernilai benar barulah blok pernyataan dieksekusi. Untuk keperluan seperti itu, kita gunakan operator logika. Fungsi IF memungkinkan untuk membuat perbandingan logis antara nilai dan apa yang diharapkan dengan menguji kondisi dan mengembalikan hasil jika True atau False. Oleh karena itu, pernyataan IF dapat memiliki dua hasil. Hasil pertama jika perbandingan Anda Benar dan hasil kedua jika perbandingan Salah. Pernyataan IF sangat kuat, dan membentuk dasar dari banyak model lembar bentang, namun pernyataan ini juga merupakan penyebab utama dari banyak masalah lembar bentang. Secara ideal, pernyataan IF harus berlaku terhadap kondisi minimal, seperti Pria/Wanita, Ya/Tidak/Mungkin, misalnya, namun terkadang Anda mungkin perlu mengevaluasi skenario lebih kompleks yang membutuhkan penumpukan* lebih dari 3 fungsi IF secara bersamaan.

Latihan :

Buatlah program untuk menampilkan ;

Jika UAS ≥ 86 maka nilai A, Bonus Tas

Jika UAS ≥ 76 maka nilai AB, Bonus Buku

Jika UAS ≥ 66 maka nilai B, Tidak ada bonus

TUJUAN INSTRUSIONAL

1. Menjelaskan mengenai konsep penggunaan struktur perulangan While-Do, Repeat-Until dan For.
2. Membedakan pemakaian struktur perulangan while-do dan repeat-until
3. Memberikan contoh-contoh program dengan menggunakan while-do, repeat-unti l dan for

A. Pengertian Perulangan

Pengulangan (*loop*) merupakan bentuk yang sering ditemui di dalam suatu program aplikasi. Di dalam bahasa Pascal, dikenal tiga macam perulangan, yaitu dengan menggunakan pernyataan For, While-Do, dan Repeat Until.

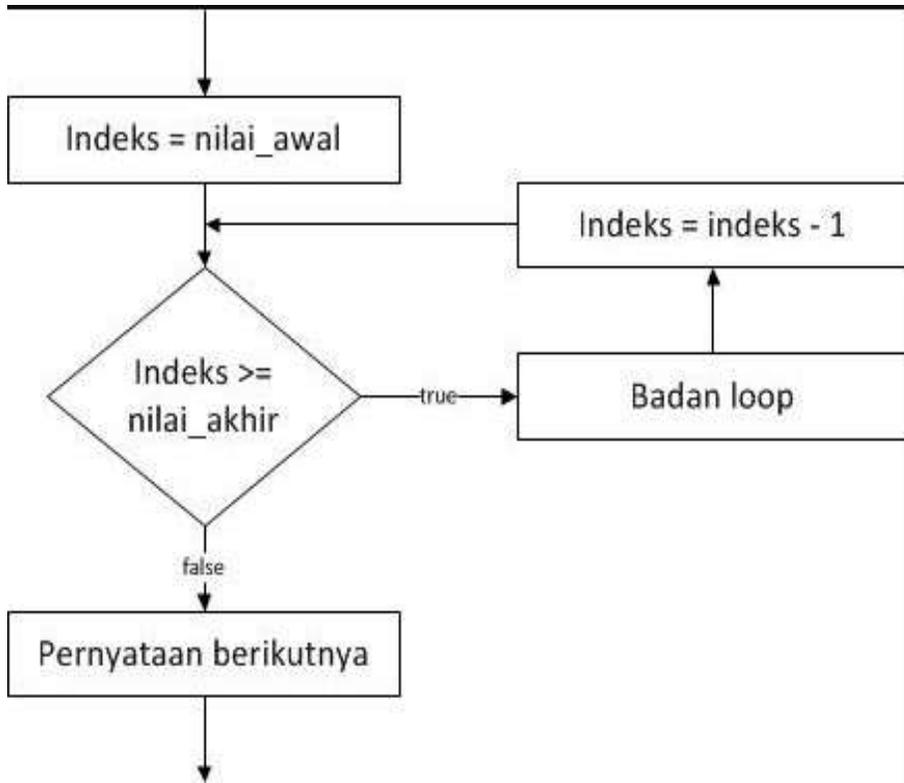
Perulangan proses, akan melakukan proses berulang-ulang selama kondisi masih bernilai True (atau dimana nilai batas yang ditentukan belum tercapai, dan kondisi akan berhenti jika keadaan berubah menjadi False (atau nilai batas telah tercapai)

B. Perulangan FOR

Perulangan dengan pernyataan For digunakan untuk mengulang pernyataan atau satu blok pernyataan berulang kali sejumlah yang ditentukan. Perulangan dengan pernyataan For dapat berbentuk perulangan positif, perulangan negatif dan perulangan tersarang.

Struktur perulangan for biasa digunakan untuk mengulang suatu proses yang telah diketahui jumlah perulangannya. Dari segi penulisannya, struktur perulangan for tampaknya lebih efisien karena susunannya lebih simpel dan sederhana. Pernyataan for digunakan untuk melakukan looping. Pada umumnya looping yang dilakukan oleh for telah diketahui batas awal, syarat looping dan perubahannya. Selama kondisi terpenuhi, maka pernyataan akan terus dieksekusi.

Flowchart Perulangan For:



Gambar 6.1 Flowchart Perulangan FOR

Bentuk umum perulangan for adalah sebagai berikut :

```

For ( inialisasiNilai; SyaratPerulangan;
PerubahanNilai )
{
    Statement yang diulang;
}
  
```

1. Ungkapan1 merupakan statement awal (inisialisasi)
2. Ungkapan2 merupakan kondisi/syarat perulangan dilakukan
3. Ungkapan3 merupakan statement control untuk perulangan
4. Statement merupakan pernyataan/perintah yang dijalankan jika syarat terpenuhi.

C. Perulangan Positif

Perulangan positif adalah perulangan dengan penghitung dari kecil ke besar atau pertambahan positif. Perulangan positif dapat dibentuk dengan menggunakan pernyataan **For-To-Do**, dengan bentuk umum:

For variabel-kontrol:=nilai awal **To** nilai akhir **Do** pernyataan
 Variabel Control, nilai awal, nilai akhir harus betipe integer

Contoh Program:



```
<---LATIH3.PAS--->
PROGRAM menampilkan_pascal_sebanyak_10x;
USES CRT;
VAR
  X : INTEGER;
BEGIN
  CLRSCR;
  FOR X :=1 TO 10 DO
    WRITELN('Pascal',X);
    READLN;
END.
```

Pascal1
Pascal2
Pascal3
Pascal4
Pascal5
Pascal6
Pascal7
Pascal8
Pascal9
Pascal10

Gambar 6.2 Program Fungsi FOR

Penjelasan:

1. Pernyataan **Writeln('Pascal')** akan diulang sebanyak 10 kali, yaitu dengan penghitung dari nilai awal 1 sampai dengan nilai akhir 10.
2. Apabila pernyataan diulang lebih dari satu pernyataan maka setelah **DO** harus memakai **Begin** kemudian beberapa pernyataan yang akan diulang dan diakhiri dengan **End;**

D. Perulangan Negatif

Perulangan negatif adalah perulangan dengan penghitung dari besar ke kecil atau penambahan negatif. Perulangan negatif dapat dibentuk dengan menggunakan pernyataan *For-DownTo-Do*, dengan bentuk umum:

For variabel-kontrol:=nilai awal **DownTo** nilai akhir **Do** pernyataan
Nilai Awal harus lebih kecil dari pada nilai Akhir

Contoh Program:



```
<---LATIH5--->
PROGRAM menampilkan_angka_dari_10_ke_1;
USES CRT;
VAR
  I : integer;
BEGIN
  clrscr;
  FOR I := 10 DOWNT0 1 DO
    writeln('Pascal ',I);
    readln;
end.
```

Pascal 10
Pascal 9
Pascal 8
Pascal 7
Pascal 6
Pascal 5
Pascal 4
Pascal 3
Pascal 2
Pascal 1

Gambar 6.3 Program Fungsi For DownTo

E. Perulangan Tersarang

Perulangan tersarang adalah perulangan yang berbeda di dalam perulangan yang lainnya. Perulangan yang lebih dalam akan diproses terlebih dahulu sampai habis, kemudian perulangan yang lebih luar baru akan akan bertambah, mengerjakan perulangan yang lebih dalam lagi mulai dari nilai awalnya dan seterusnya.

Contoh Program:



```

<---LATIH4.PAS--->
PROGRAM menampilkan_indeks_matriks_3_kali_3;
USES CRT;
VAR
  X,Y : INTEGER;
BEGIN
  CLRSCR;
  FOR X :=1 TO 3 DO
  BEGIN
    FOR Y :=1 TO 3 DO
      WRITE(X,Y,' ');
    Writeln(' ');
  END;
  Readln;
END.
```

11	12	13
21	22	23
31	32	33

Gambar 6.4 Program For di dalam For

F. Perulangan While-Do

Ini menggunakan pernyataan *While—Do*. Pernyataan *While—Do* digunakan untuk melakukan proses perulangan suatu pernyataan atau blok pernyataan terus-menerus selama kondisi ungkapan logika pada *While* masih bernilai logika benar. Perulangan WHILE banyak digunakan pada program yang terstruktur. Perulangan ini banyak digunakan bila jumlah perulangannya belum diketahui. Proses perulangan akan terus berlanjut selama kondisinya bernilai benar ($\neq 0$) dan akan berhenti bila kondisinya bernilai salah ($=0$).

```

while (syarat pengulangan)
{
    statement yang dijalankan;
    statement control:
}

```

Dua perintah di bawah ini adalah identik.

```

for (a = 1; a <= 5; a++)
{
    cout << "Hello world \n";
}

```

```

a = 1;
while (a <= 5){
    cout << "Hello world \n";
    a++;
}

```



Jika Anda menggunakan WHILE, pastikan bahwa suatu saat bagian kondisi sampai bernilai FALSE. Apabila tidak, proses perulangan akan terus berjalan selamanya.

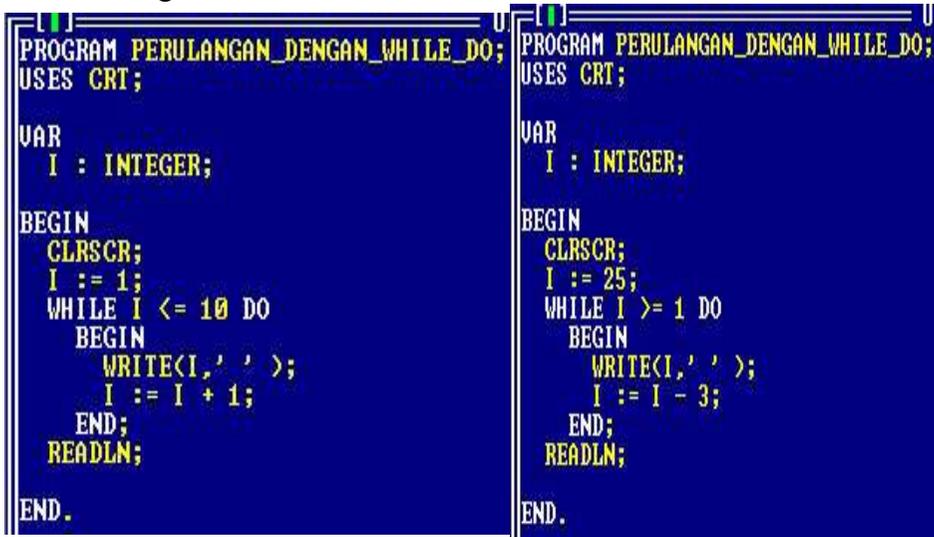
Contoh program di bawah ini digunakan untuk menjumlahkan sejumlah data angka. Angka yang akan dijumlahkan diinputkan satu-persatu. Proses pemasukan data angka akan berhenti ketika dimasukkan angka -1. Setelah itu tampil hasil penjumlahannya.

```

#include <iostream.h>
#include <conio.h>
void main()
{
    int data, jumlah, cacah;
    jumlah = 0;
    data = 0;
    cacah = 0;
    while (data != -1)
    {
        cout << "Masukkan data angka : ";
        cin >> data; jumlah += data; cacah++;
    }
    cout << "Jumlah data adalah : " << jumlah << endl;
    cout << "Rata-rata : " << jumlah/cacah;
}

```

Contoh Program:



```
PROGRAM PERULANGAN_DENGAN_WHILE_DO;
USES CRT;

VAR
  I : INTEGER;

BEGIN
  CLRSCR;
  I := 1;
  WHILE I <= 10 DO
  BEGIN
    WRITE(I, ' ');
    I := I + 1;
  END;
  READLN;
END.
```

```
PROGRAM PERULANGAN_DENGAN_WHILE_DO;
USES CRT;

VAR
  I : INTEGER;

BEGIN
  CLRSCR;
  I := 25;
  WHILE I >= 1 DO
  BEGIN
    WRITE(I, ' ');
    I := I - 3;
  END;
  READLN;
END.
```

Gambar 6.5 Program Fungsi While Do

Penjelasan:

Perulangan dari while akan terus menerus dikerjakan bila kondisinya masih benar. Dalam hal ini kondisinya adalah I dan bila nilai I kurang dari 5, berarti kondisi di dalam While masih terpenuhi dan perulangan akan selesai setelah nilai I lebih besar atau sama dengan 5.

G. Repeat-Until

Struktur Repeat...Until digunakan untuk mengulang pernyataan-pernyataan atau blok pernyataan sampai kondisi yang diseleksi di Until tidak terpenuhi. Bentuk umumnya adalah Repeat pernyataan Until. Repeat berarti ulangi dan until berarti sampai. Jadi, repeat-until adalah struktur pengulangan dimana aksi dilakukan hingga kondisi (persyaratan) berhenti terpenuhi.

Example:

Misalkan kita ingin mengetahui nilai dari mahasiswa jika data masukannya adalah NIM. Program Pencarian {Program mencari nilai mahasiswa didalam tabel dengan

NIM= a. Tabel sudah berisi data NIM, nama dan nilai}

Algoritmanya: tinjau entry pertama tabel repeatif NIM pada entry tabel sama dengan NIM yang dicari then ambil nama dari NIM tersebut setinjau entry berikutnya until nilai yang dicari ditemukan atau akhir tabel sudah terlampaui Pada repeat-until jumlah pengulangan tidak

dapat diketahui di awal. Karena pengulangan aksi akan terus dilakukan sampai ditemukan entry dari apa yang ditanyakan atau akhir tabel sudah terlampaui, berbeda dengan for-do yang jumlah pengulangannya dilaksanakan.

Contoh Program:

```

PROGRAM PERULANGAN_DENGAN_REPEAT_UNTIL;
USES CRT;

VAR
  I : INTEGER;

BEGIN
  CLRSCR;
  I := 1;
  REPEAT
    WRITE(I, ' ');
    I := I + 2;
  UNTIL I >= 20;

  READLN;

END.

```

```

PROGRAM PERULANGAN_DENGAN_REPEAT_UNTIL;
USES CRT;

VAR
  I : INTEGER;

BEGIN
  CLRSCR;
  I := 100;
  REPEAT
    WRITE(I, ' ');
    I := I - 10;
  UNTIL I <= 0;

  READLN;

END.

```

Gambar 6.6 Program Repeat Until

H. Perbedaan antara struktur Repeat..Until dengan struktur While Do

Perbedaan antara struktur Repeat..Until dengan struktur While Do adalah sebagai berikut:

1. Tak pada pengujian kondisi. Paling sedikit statemen-statemen dalam repeat until diproses sekali, karena seleksi kondisi ada pada statemen until yang terletak dibawah. Pada while do paling sedikit dikerjakan nol kali, karena seleksi kondisi ada pada statemen while yang terletak diatas, sehingga apabila kondisi tidak terpenuhi maka tidak akan masuk ke dalam lingkungan perulangannya.
2. Repeat until mengulang pernyataan selama kondisi belum terpenuhi sedang while do mengulang pernyataan selama kondisi masih terpenuhi.

3. Pada repeat until dapat tidak menggunakan blok statemen (BEGIN dan END) untuk menunjukan batas perulangannya, karena batas perulangannya sudah ditunjukkan oleh repeat sampai dengan until. Sedangkan WHILE DO harus menggunakan blok statemen (BEGIN dan END).

Tabel 6.1 Perbandingan Perulangan

<pre> Program For To ; Uses crt ; Var i : byte ; begin clrscr ; for i : = 1 to 5 do writeln (' Pascal ') ; readln ; end . </pre>	<pre> Program For DownTo; Uses crt ; Var i : byte ; begin clrscr ; for i : = 5 downto 1 do writeln (' Pascal ') ; readln ; end . </pre>	<pre> Program While ; Uses crt ; Var i : byte ; begin clrscr ; I : = 1 ; While I <= 5 Do begin writeln(I); I : = I + 1 ; end ; readln ; end . </pre>	<pre> Program Repeat; Uses crt ; Var i : byte ; begin clrscr ; I : = 1 ; Repeat writeln (I) ; I : = I + 1; Until I >= 5 readln ; end . </pre>
--	---	---	--

Latihan :

1. Tampilkan bilangan 100 sampai 1 menggunakan fungsi FOR
2. Tampilkan bilangan 1,3,5,7,9,11,13,15,17 menggunakan fungsi While – DO
3. Tampilkan bilangan 55,50,45,40,35,30,25,20,15 menggunakan fungsi Repeat - Until

TUJUAN INSTRUSIONAL

1. Menjelaskan konsep pemakaian statemen Label
2. Menjelaskan konsep pemakaian statemen GOTO serta mahasiswa diharapkan dapat membuat contoh program sederhananya.
3. Menerapkan Statemen GOTO yang di rujuk dengan Statemen LABEL.

A. Pengertian dari Label

Label adalah suatu nama tertentu (dapat terdiri angka atau huruf atau kombinasinya), yang dalam program nanti akan dituju oleh statemen *GOTO*. Namun dalam PASCAL sesungguhnya statement *GOTO* ini hanya diijinkan tetapi sama sekali TIDAK DIANJURKAN. Karena statemen Goto menyebabkan struktur program menjadi tidak jelas.

Bentuk Umum Deklarasi label

Label : Nama label

Label : Ulang;
Label : 100, Atas,

B. Pengertian dari GoTo

Pernyataan Goto digunakan untuk melakukan transfer eksekusi program ke pernyataan yang dideklarasikan oleh Label.

Bentuk Penulisan

Goto label;

1. Label harus berada didalam blok yang sama dengan pernyataan goto. Hal ini tidak memungkinkan melompat keluar dari dalam procedure atau function.
2. Suatu Label dideklarasikan pada bagian deklarasi Label.
3. Bentuk Penulisan :

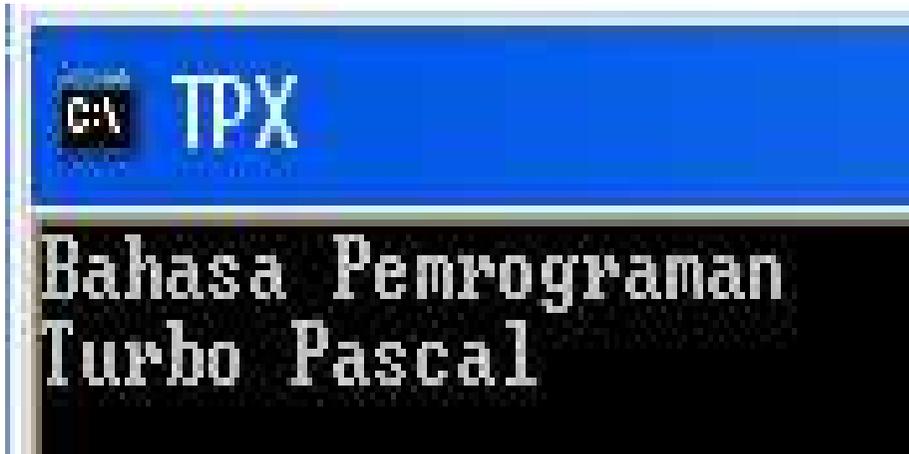
Label Identifier1, .. Identifier
4. Untuk penulisan identifier yang akan digunakan pada label, penggunaan digit secara urut diantara 0 dan 9999 dapat labe

C. Contoh Program Untuk Membuat Label

```
Program Contoh_Label;  
uses crt;  
label  
  100, selesai;  
  
Begin  
  clrscr;  
  writeln('Bahasa Pemrograman');  
  GOTO 100;  
  writeln('Visual Basic');  
  writeln('Visual Foxpro');  
  
  100 :  
    writeln('Turbo Pascal');  
  
  GOTO Selesai;  
  writeln ('Delphi');  
  
  Selesai :  
  readln;  
end.
```

Gambar 7.1 Program GoTo

Kemudian akan muncul hasil sebagai berikut :



Gambar 7.2 Hasil Program GoTo

Berikut ini merupakan program goto dalam label

```
PROGRAM Goto_dalam_label;
Uses Crt;
Label
  Atas;
Var
  Nil1, Nil2 : Integer;
  NilRata   : Real;
  Lagi      : Char;
Begin
  Atas:
  ClrScr;
  WriteLn('Program Hitung Nilai');
  WriteLn('-----');
  WriteLn;
  Write('Masukan Nilai 1 : '); ReadLn(Nil1);
  Write('Masukan Nilai 2 : '); ReadLn(Nil2);
  NilRata := (Nil1 + Nil2)/2;

  WriteLn('Nilai Rata - Rata : ', NilRata:5:2);

  WriteLn;
  Write('Ingin Hitung Lagi[Y/T] : ');
  ReadLn(Lagi);

  If (Lagi='Y') Then
  GoTo Atas;

End.
```

Gambar 7.3 GoTo dalam Label 1

Kemudian akan muncul hasil sebagai berikut :



```
C:\ TPX
Program Hitung Nilai
-----
Masukan Nilai 1 : 90
Masukan Nilai 2 : 80
Nilai Rata - Rata : 85.00
Ingin Hitung Lagi[Y/T] :
```

Gambar 7.4 Hasil Program GoTo Label 1



```
[[ ]] LABEL2.PAS
Program Goto_dalam_label;
Uses Crt;
Label
  Atas;
Var
  Nil1, Nil2 : Integer;
  NilRata    : Real;
  Lagi       : Char;
Begin
  Atas:
  ClrScr;
  WriteLn('Program Hitung Nilai');
  WriteLn('-----');
  WriteLn;
  Write('Masukan Nilai 1 : '); ReadLn(Nil1);
  Write('Masukan Nilai 2 : '); ReadLn(Nil2);
  NilRata := (Nil1 + Nil2)/2;
  Goto Atas;
End;
```

Gambar 7.5 Koding Program GoTo Label 2

Hasil programnya sebagai berikut :



```
TPX
Program Hitung Nilai
-----
Masukan Nilai 1 : 80
Masukan Nilai 2 : 90
Nilai Rata - Rata : 85.00
Tekan Sembarang Tombol Untuk Cetak
```

Gambar 7.6 Hasil Program GoTo Label 2



```
WriteLn('Nilai Rata - Rata : ',NilRata:5:2);
clrscr;
WriteLn('Tekan Sembarang Tombol Untuk Cetak');readln;
WriteLn('Program Hitung Nilai');
WriteLn(' +-----+');
WriteLn(' |Nilai 1      :',Nil1,' |');
WriteLn(' |Nilai 2      :',Nil2,' |');
WriteLn(' |Rata - Rata  :',NilRata:5:2,' |');
WriteLn(' +-----+');
WriteLn;
Write('Ingin Hitung Lagi(Y/T) : ');
ReadLn(Lagi);
If (Lagi='Y') Then
GoTo Atas;
End.
```

Gambar 7.7 Koding Program GoTo Label Rata – Rata

Hasil Programnya adalah sebagai berikut :



```
TPX
Program Hitung Nilai
+-----+
!Nilai 1      :80  !
!Nilai 2      :90  !
!Rata - Rata  :85.00!
+-----+

Ingin Hitung Lagi[Y/T] :
```

Gambar 7.8 Hasil Program GoTo Label Rata – Rata

Tidak hanya itu saja penggunaan label dan GOTO pada pascal juga dapat untuk menghitung bidang seperti segitiga, segiempat, persegi dss berikut adalah contohnya :



```
PROGRAM Hitung_luas_Segitiga;
uses crt;
label
  ulang;
var
  alas, tinggi : integer;
  luas         : real;
  lagi         : char;
Begin
  Ulang :
  clrscr;
  Write('Masukkan panjang alas = ');readln(alas);
  Write('Masukkan tinggi segitiga = ');readln(tinggi);
  Luas:=alas*tinggi/2;
  Writeln('Luas segitiga = ', luas:7:2);
  Write('Ingin Input lagi[Y/T] :');
  readln(lagi);

  If <lagi='Y'> or <lagi='y'> Then
    Goto Ulang
End.
```

Gambar 7.9 Koding Luas Segitiga

Di sini kita juga bisa belajar membuat desain program seperti perancangan aplikasi dengan sub program contohnya menu utama , menu merupakan daftar pilihan yang akan di eksekusi sesudah dengan pilihan yang ada contohnya saja seperti :

Menu Pilihan Cari Luas dan Keliling

1. Segi Tiga
2. Segi Empat
3. Lingkaran
4. Keluar

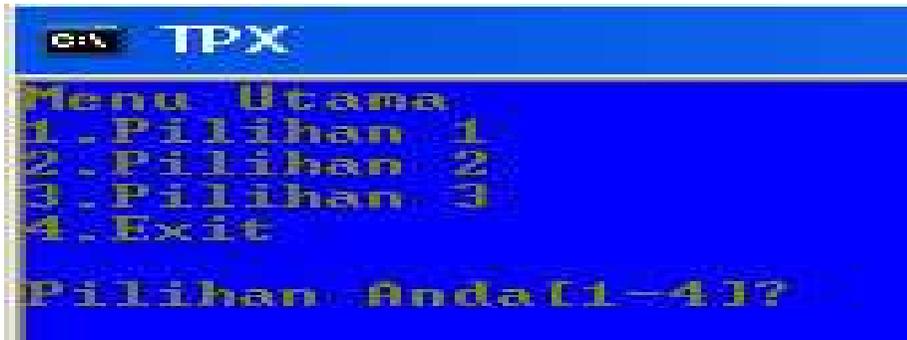
Cara membuat menu tidak berbeda jauh seperti contoh di atas

```
PROGRAM Pilihan_Menu<Output>;
Uses CRT;
VAR
    Pilihan      : Integer;
Begin
textcolor(6+blink);
textbackground(1);
    CLRSCR;
    Writeln('Menu Utama');
    Writeln('1.Pilihan 1');
    Writeln('2.Pilihan 2');
    Writeln('3.Pilihan 3');
    Writeln('4.Exit      ');
    Writeln;
    Write('Pilihan Anda[1-4]?'); Readln(Pilihan);

    Case Pilihan OF
    1 : Writeln('Anda Memilih Nomor Satu');
    2 : Writeln('Anda Memilih Nomor Dua');
    3 : Writeln('Anda Memilih Nomor Tiga');
    4 : Writeln('Anda Keluar Dari Menu ');
    Else
        Writeln('Anda Salah Pilih');
    End;
    Readln;
End.
```

Gambar 7.10 Koding Menu Utama

Hasilnya adalah :

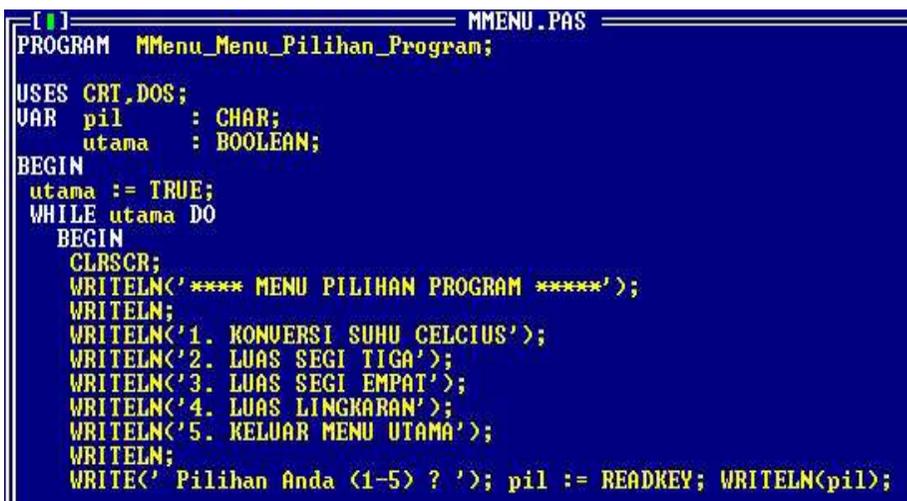


The screenshot shows a DOS command prompt window with a blue background. The title bar reads 'C:\> TPX'. The output of the program is as follows:

```
Menu Utama
1. Pilihan 1
2. Pilihan 2
3. Pilihan 3
4. Exit

Pilihan Anda [1-4]?
```

Gambar 7.11 Hasil Menu Utama



The screenshot shows a Turbo Pascal source code editor window titled 'MMENU.PAS'. The code is as follows:

```
PROGRAM MMenu_Menu_Pilihan_Program;
USES CRT,DOS;
VAR pil : CHAR;
    utama : BOOLEAN;
BEGIN
    utama := TRUE;
    WHILE utama DO
    BEGIN
        CLSCR;
        Writeln('**** MENU PILIHAN PROGRAM ****');
        Writeln;
        Writeln('1. KONVERSI SUHU CELCIUS');
        Writeln('2. LUAS SEGI TIGA');
        Writeln('3. LUAS SEGI EMPAT');
        Writeln('4. LUAS LINGKARAN');
        Writeln('5. KELUAR MENU UTAMA');
        Writeln;
        Write('Pilihan Anda (1-5) ? '); pil := READKEY; Writeln(pil);
```

```

        SWAPVECTORS;
        CASE pil OF
          '1':EXEC<'CELCIUS.EXE',''>;
          '2':EXEC<'SEGI3.EXE',''>;
          '3':EXEC<'SEGI4.EXE',''>;
          '4':EXEC<'LING.EXE',''>;
          '5': utama:=FALSE;
        END; < CASE pil OF >
        SWAPVECTORS;
      END; < WHILE UTAMA DO >
END.

CASE pilihan OF
  1 : Begin
      SwapVectors;
      EXEC<'CTIF2b.EXE',''>;
      SwapVectors;
    End;
  2 : WRITELN (' Anda memilih nomer dua ');
  3 : WRITELN (' Anda memilih nomer tiga ');
  4 : WRITELN (' Anda keluar dari menu ');
ELSE
      WRITELN (' Anda salah pilih ');
END;

```

Gambar 7.12 Koding Menu Utama 2

Dapat dilihat hasilnya adalah :

```

***** MENU PILIHAN PROGRAM *****
1. KONVERSI SUHU CELCIUS
2. LUAS SEGI TIGA
3. LUAS SEGI EMPAT
4. LUAS LINGKARAN
5. KELUAR MENU UTAMA

Pilihan Anda <1-5> ?

```

Gambar 7.13 Hasil Menu Utama 2

**FUNGSI KARAKTER
DAN NUMERIK**

TUJUAN INSTRUSIONAL

1. Menjelaskan mengenai konsep string atau manipulasi dengan string
2. Menggunakan pemakaian fungsi numeric
3. Membuat contoh program dengan menggunakan manipulasi string

A. CONCAT(fungsi)

1. Untuk menggabungkan 2 atau beberapa variabel string.
2. Sintaks: CONCAT(s1 [,s2,...,sn]: String) : STRING;
contoh: CONCAT('ABC','DEF') { ABCDEF }

Tabel 8.1 Argumen Deskripsi

Argumen	Deskripsi
teks1 (diperlukan)	Item teks yang akan digabungkan. String, atau larik string, seperti rentang sel.
[teks2, ...] (opsional)	Item teks tambahan yang akan digabungkan. Item teks dapat berisi maksimal 253 argumen teks. Masing-masing dapat berupa string atau larik string, seperti rentang sel.

CONCAT(text1, [text2],...)

Misalnya, = CONCAT ("Matahari," ", "akan"," ", "muncul"," ", "besok.") akan kembali ke **Matahari akan muncul besok.**

Keterangan

Jika string yang dihasilkan melebihi 32767 karakter (batas sel), CONCAT akan mengembalikan kesalahan #VALUE!.

Contoh : Salin contoh data di dalam setiap tabel berikut ini lalu tempel ke dalam sel A1 lembar kerja Excel yang baru. Agar rumus menampilkan hasil, pilih datanya, tekan F2, lalu tekan Enter. Jika perlu, Anda bisa menyesuaikan lebar kolom untuk melihat semua data.

Contoh 1

=CONCAT(B:B, C:C)	A's	B's
	a1	b1
	a2	b2
	a4	b4
	a5	b5
	a6	b6
	a7	b7

Karena fungsi ini memungkinkan referensi kolom dan baris secara penuh, dan akan kembali pada hasil berikut: **A'sa1a2a4a5a6a7B'sb1b2b4b5b6b7**

Contoh 2

=CONCAT(B2:C8)	A's	B's
	a1	b1
	a2	b2
	a4	b4
	a5	b5
	a6	b6
	a7	b7

Hasil: a1b1a2b2a4b4a5b5a6b6a7b7

Data	Nama Depan	Nama belakang
brook trout	Andreas	Hauser
spesies	Fourth	Pine
32		
Rumus	Deskripsi	Hasil
=CONCAT("Populasi arus untuk ", A2," ", A3, " adalah ", A4, "/mil.")	Membuat sebuah kalimat dengan menggabungkan data di kolom A dengan teks lainnya.	Populasi arus untuk spesies brook trout adalah 32/mil
=CONCAT(B2," ", C2)	Menggabungkan tiga hal: string dalam sel B2, karakter spasi, dan nilai dalam sel C2.	Andreas Hauser
=CONCAT(C2, ", ", B2)	Menggabungkan tiga hal: string dalam sel C2, string dengan karakter koma dan spasi, dan nilai dalam sel B2.	Hauser, Andreas
=CONCAT(B3," & ", C3)	Menggabungkan tiga hal: string dalam sel B3, string yang	Fourth & Pine

	terdiri dari spasi dengan simbol dan spasi lainnya, dan nilai dalam sel C3.
=B3 & " & " & C3	Menggabungkan item yang sama seperti contoh sebelumnya, tetapi dengan menggunakan simbol operator perhitungan (&) bukan fungsi CONCAT. Fourth & Pine

B. COPY(fungsi)

Mengambil satu(1) atau beberapa karakter dari sebuah string.

Sintaks: COPY(S,Index,Count) : String;

Keterangan :

S = sebuah string (string).

Index = posisi awal kita akan mengambil beberapa karakter (integer)

Count = banyaknya karakter yang akan diambil (integer).

C. DELETE(prosedur)

Menghapus sebagian karakter dari sebuah string.

Sintaks: DELETE(S,Index,Count);

Keterangan : sama dengan statemen COPY.

D. INSERT(prosedur)

Menyisipkan satu(1) atau beberapa karakter ke dalam sebuah string.

Sintaks: INSERT(Source,var S,Index);

Keterangan :

Source = sumber string untuk disisipi (string)

var S = string tujuan yang akan disisipi oleh string Source (string)

Index = posisi mulai (integer).

E. LENGTH (fungsi)

Memberikan nilai panjang dari suatu string (jumlah karakter dalam string).

Sintaks: LENGTH(S);

Keterangan :

S = string

LENGTH(S) menghasilkan nilai integer.

F. POS (fungsi)

Mencari posisi sebuah bagian string (substring) didalam sebuah string.

Sintaks: POS(Substr,S); {menghasilkan nilai Byte}

Keterangan :

Substr = substring yang akan dicari posisinya di dalam sebuah string S. Bila bernilai 0 berarti nilai string yang dicari tidak ada.

G. STR(prosedur)

Merubah nilai numerik ke dalam nilai string.

Sintaks: STR(N,S);

Keterangan :

N = data tipe integer,

S = data tipe string.

H. VAL(prosedur)

Merubah nilai string ke dalam nilai numerik.

Sintaks: VAL(S,N,P);

Keterangan :

S = nilai string,

N = nilai real,

P = posisi salah.

Nilai string harus berisi angka, plus atau minus, bila tidak berarti kesalahan dan letakkesalahannya ditunjukkan oleh variabel posisi salah. Jika benar, maka nilai variabel tsb = 0 (nol).

I. UPCASE(fungsi)

Memberikan huruf kapital dari argumen.

Sintaks: UPCASE(S);

Keterangan :

S = variabel bertipe karakter.

J. COS(fungsi)

Memberikan nilai dari fungsi Cosinus.

Sintaks: COS(x);

K. EXP(fungsi)

Menghitung nilai pangkat dari bilangan e (bilangan alam), yaitu sebesar x. Sintaks: EXP(x);

x dapat bertipe real atau integer dan akan menghasilkan nilai bertipe real.

L. INT(fungsi)

Memberikan nilai integer (bilangan bulat) dari suatu variabel dengan membuang bilangan di belakang koma. Sintaks: INT(X);

M. LN(fungsi)

Digunakan untuk menghitung nilai logaritma alam (natural logarithm) dari nilai x. Sintaks: LN(x);

N. SIN(fungsi)

Memberikan nilai dari fungsi Sinus. Sintaks: SIN(x);

O. SQR(fungsi)

Digunakan untuk menghitung nilai pangkat kuadrat dari suatu bilangan. Sintaks: SQR(x); Tipe dari x bisa berupa real maupun integer. Dan hasilnya akan sama dengan tipe dari x.

P. SQRT(fungsi)

Digunakan untuk menghitung nilai akar dari suatu bilangan. Sintaks: SQRT(x);

Q. CHR(fungsi)

Merubah nilai dari byte ke bentuk karakter yang sesuai dengan kode ASCII. Sintaks: CHR(x); Keterangan : x bertipe byte

R. ROUND(fungsi)

Membulatkan data tipe real ke data tipe longint. Sintaks: ROUND(X); Keterangan : Jika nilai pecahan $< 0,5$ maka dibulatkan kebawah. Jika nilai pecahan $> 0,5$ maka dibulatkan keatas. contoh : WRITELN('10/3 dibulatkan = ',ROUND(10/3)); hasilnya : 10/3 dibulatkan = 3

S. TRUNC(fungsi)

Membulatkan kebawah data tipe real ke data tipe longint. Sintaks: TRUNC(X); contoh : WRITELN('20/3 dibulatkan kebawah = ',TRUNC(20/3));

hasilnya : 20/3 dibulatkan kebawah = 6

T. TEXTCOLOR(prosedur)

Untuk mengatur warna dari karakter-karakter di layar. Sintaks: TEXTCOLOR(color : Byte); Catatan : untuk pilihan warna lihat pada buku Turbo Pascal.f8

U. TEXTBACKGROUND(prosedur)

Untuk mengatur warna latar belakang dari karakter-karakter dilayar. Sintaks: TEXTBACKGROUND(Color : Byte);

V. WINDOW(prosedur)

Untuk membuat suatu jendela (window) yang terletak pada layar. Sintaks: WINDOW(x1,x2,y1,y2 : Byte);

x1,x2 = kordinat kiri atas dengan nilai maksimal sesuai dengan mode layar

y1,y2 = kordinat kanan bawah dgn nilai maksimal sesuai dengan mode layar.

Berikut adalah contoh perogramnya :

Contoh Program 1:



```
CS Turbo Pascal 7.0
File Edit Search Run C
[ ]
Program Manipulasi_String;
uses crt;
var
  i : integer;
  s,c : string;
const
  x = 'STEKOM';
  y = 'Semarang';
  z = 'PAT';
begin
  clrscr;
  s := concat(x,y,z);
  writeln(s);
  insert('&',s,15);
  writeln(s);
  delete(s,7,8);
  writeln(s);
  c := copy(s,1,6);
  writeln(c);
  i := length(s);
  writeln('Panjang String S adalah ',i);
  writeln('Posisi "PAT" pada string S adalah ', Pos('PAT',s));
  readln;
end.
```

Gambar 8.1 Koding Manipulasi String

```

C:\ Turbo Pascal 7.0
STEKOMSemarangPAT
STEKOMSemarang & PAT
STEKOM & PAT
STEKOM
Panjang String S adalah :12
Posisi "PAT" pada string S adalah :10

```

Gambar 8.2 Hasil Manipulas String

```

C:\ TPX
File Edit Search Run Compile Debug Tools Optio
MAN_NUM.PAS
PROGRAM Aritmatik;
USES CRT;
VAR
  x : REAL;
BEGIN
  CLRSCR;
  WRITE<'masukkan nilai dari x = '>;READLN(x);

  IF x<0 THEN x:=ABS(x);
  WRITELN<'Nilai x = ',x:5:2>;
  WRITELN<'Nilai eksponensialnya = ',EXP(x):7:3>;
  WRITELN<'Nilai logaritma alamnya = ',LN(x):7:3>;
  WRITELN<'Nilai integernya = ',INT(x):5:2>;
  WRITELN<'Nilai fraksionalnya = ',FRAC(x):5:2>;
  WRITELN<'Nilai x dipangkatkan = ',SQR(x):9:3>;
  WRITELN<'Nilai x diakarkan = ',SQRT(x):9:3>;
  WRITE<'Nilai x jika dimasukkan dalam '>;
  WRITELN<'fungsi SIN,COS,TANGEN : '>;
  WRITELN<' - Sinus = ',SIN(x):9:3>;
  WRITELN<' - Cosinus = ',COS(x):9:3>;
  WRITELN<' - Tangen = ',ARCTAN(x):9:3>;
  READLN;
END.

```

Gambar 8.3 Koding Program Aritmatika

```

C:\ TPX
masukkan nilai dari x = 10
Nilai x = 10.00
Nilai eksponensialnya = 22026.466
Nilai logaritma alamnya = 2.303
Nilai integernya = 10.00
Nilai fraksionalnya = 0.00
Nilai x dipangkatkan = 100.000
Nilai x diakarkan = 3.162
Nilai x jika dimasukkan dalam fungsi SIN,COS,TANGEN :
- Sinus = -0.544
- Cosinus = -0.839
- Tangen = 1.471

```

Gambar 8.4 Hasil Program Aritmatika

```
TPX
File Edit Search Run Compile Debug Tools Options Window
MAN_DATA.PAS
USES CRT;
TYPE
    hari = <hr0,hr1,hr2,hr3,hr4,hr5,hr6,hr7>;
VAR
    urutanhr : hari;
CONST
    namahr : ARRAY[hr1..hr7] OF STRING[6]=
    ('Senin','Selasa','Rabu','Kamis','Jumat','Sabtu','Minggu');
BEGIN
    CLRSCR;
    WRITELN('DAFTAR NAMA HARI');
    urutanhr := hr0;
    WHILE Urutanhr < hr7 DO
    BEGIN
        urutanhr := SUCC(urutanhr);
        WRITE('hari ke ',ORD(Urutanhr):2,' adalah ');
        WRITELN(namahr[urutanhr]);
        READ;
    END;
    READLN;
END.
```

Gambar 8.5 Koding Program Hari

```
TPX
DAFTAR NAMA HARI
hari ke 1 adalah Senin
hari ke 2 adalah Selasa
hari ke 3 adalah Rabu
hari ke 4 adalah Kamis
hari ke 5 adalah Jumat
hari ke 6 adalah Sabtu
hari ke 7 adalah Minggu
```

Gambar 8.6 Hasil Program Hari

LATIHAN SOAL

1. Buatlah program dibawah ini dengan tampilan menggunakan perintah Window,
2. Textcolor, Textbackground, Gotoxy, dan Sound untuk memperindah tampilan.
 - a. Mengubah derajat temperatur, dari derajat Celcius ke derajat Fahrenheit dan Reamur (derajat Celcius diinput
 - b. Menghitung Luas dan Keliling lingkaran, dengan jari-jari diketahui (diinput).
 - c. Menghitung Luas dan Keliling segitiga sembarang yang diketahui ke tiga sisinya.
 - d. Mencari nilai Sinus, Cosinus, dan Tangen dengan sudut diinput.
 - e. Mencari akar dan kuadrat dari suatu nilai (nilai diinput).
 - f. Mencari nilai bulat dan pecahan dari suatu nilai yang dimasuk kan melalui keyboard (diinput). Nilai pecahan tersebut dibulatkan sampai 3 angka dibelakang koma (.).
 - g. Tampilkan nama dan NPM anda di dalam window, dan terletak pada tengah- tengah layar.
 - h. Tampilkan tulisan 'STEKOM Semarang' di dalam window pada pojok kanan atas dengan ukuran window sama dengan tulisan tersebut.

LATIHAN SOAL 2

1. Buatlah program pada soal jenis I (no. 1-6) dengan tampilan menggunakan 2 window.
2. Window yang pertama digunakan untuk nilai yang diinput. Window yang kedua untuk hasil dari program (output).
3. Buatlah program untuk menggabungkan 2 buah kata yang diinput. Setiap kata yang diinput harus berada didalam window yang dan hasilnya berada pada window yang berbeda pula.
4. Buatlah program untuk menampilkan window secara acak dengan warna yang berbeda.

**BAB
IX**

**FUNGSI
TEXT COLOR**

TUJUAN INSTRUSIONAL

1. Menjelaskan konsep dari text color dan Background color.
2. Menjelaskan perintah sound dan readkey
3. Membuat program dari perintah textcolor dan textbackground color, sound dan readkey .

A. Pengenalan

Dalam Turbo Pascal telah disediakan prosedur standar yang dapat digunakan untuk mengatur warna tampilan teks di layar, yaitu Textcolor, Textbackground, LowVideo, NormVideo dan HighVideo.

B. TEXTCOLOR (prosedur)

1. Untuk mengatur warna dari karakter-karakter di layar.
2. Sintaks: TEXTCOLOR(color : Byte);
3. Catatan : untuk pilihan warna lihat pada buku Turbo Pascal.f8

Penggunaan prosedur ini harus melibatkan unit standar Crt dan di dalam unit standar ini telah didefinisikan beberapa konstanta yaitu :
Penggunaan prosedur ini harus melibatkan unit standar Crt dan di dalam Unit standar ini telah didefinisikan beberapa konstanta, yaitu :

Tabel 9.1 Nomor Angka

0	BLACK	HITAM
1	BLUE	BIRU
2	GREEN	HIJAU
3	CYAN	BIRU MUDA
4	RED	MERAH
5	MAGENTA	MERAH MUDA
6	BROWN	COKELAT
7	LIGHTGRY	PUTIH NORMAL

8	DARKGGRY	ABU-ABU
9	LIGHTBLUE	BIRU TERANG
10	LIGHTGREEN	HIJAU TERANG
11	LIGHTCYAN	BIRU MUDA TERANG
12	LIGHTRED	MERAH TUA TERANG
13	LIGHTMAGENTA	MERAH MUDA TERANG
14	YELLOW	KUNING TERANG
15	WHITE	PUTIH TERANG

C. TEXTBACKGROUND (prosedur)

1. Untuk mengatur warna latar belakang dari karakter-karakter dilayar.
2. Sintaks: TEXTBACKGROUND(Color : Byte);

D. WINDOW(prosedur)

Dengan menggunakan prosedur standar Window, maka dimungkinkan untuk membuat suatu jendela (window) yang letaknya dimanapun dalam layar. Jika anda membuat suatu window, maka jendela tersebut akan berperan seperti layar yang utuh dengan mengabaikan bagian layar lainnya diluar jendela.

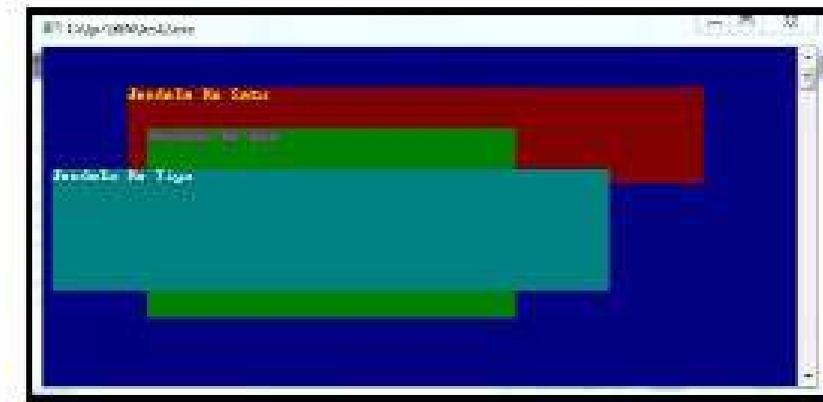
Untuk membuat suatu jendela (window) yang terletak pada layar.

Sintaks: WINDOW(x1,x2,y1,y2 : Byte);

1. Uses Crt;
2. Begin
3. Window(1,1,80,25);
4. Textcolor(1);Textbackground(1);
5. Clrscr;
6. Window(10,4,70,10);
7. Textcolor(14);Textbackground(4);
8. Clrscr;
9. Writeln('Jendela Ke Satu ');
10. Window(12,7,50,20);
11. Textcolor(13);Textbackground(2);
12. clrscr;
13. Writeln('Jendela Ke Dua');
14. Window(2,10,60,18);

15. Textcolor(15);Textbackground(3);
16. clrscr;
17. Writeln('Jendela Ke Tiga');
18. Readln;
19. End.

Contoh Output yang dihasilkan :



Gambar 9.1 Output Hasil Textcolor

x1,x2 = kordinat kiri atas dengan nilai maksimal sesuai dengan mode layar

y1,y2 = kordinat kanan bawah dgn nilai maksimal sesuai dengan mode layar.

E. Prosedur

Untuk mengatur lebar layar, 80 kolom atau 40 kolom. Sintaks: TEXTMODE(Mode: Byte); Default = C80

F. SOUND(prosedur)

Untuk mengaktifkan suara(beep) pada internal speaker. Sintaks: SOUND(Hz : word); Untuk menonaktifkannya, gunakan statemen NOSOUND. Tuliskan kalimat “saya belajar pascal” dengan warna text kuning cerah kemudian dibawahnya tuliskan “pascal memang asik “ dengan warna text hijau cerah dan terakhir tuliskan “Saya suka program pask” dengan warna text biru muda ! Lakukan langkah berikut :

1. Tentukan kode warna yang digunakan.
Kuning cerah =14
Hijau cerah =10

Biru muda =11

2. Tulislah perogam dengan rumus umum .

Maka penulisanya sebagai berikut :

```
TEXTCOLOR(14);
```

```
WEITELN(' Sayabelajar pascal');
```

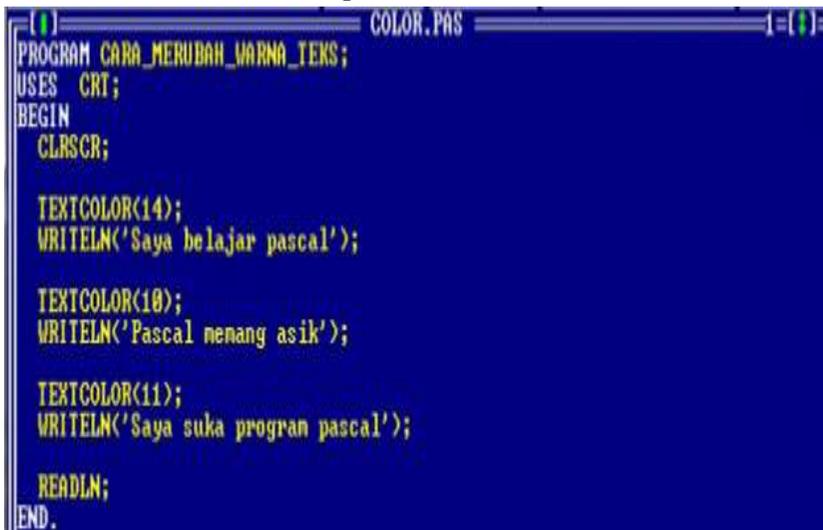
```
TEXTCOLOR(10);
```

```
WEITELN(' Pascal memang asik');
```

```
TEXTCOLOR(11);
```

```
WEITELN(' Saya suka program);
```

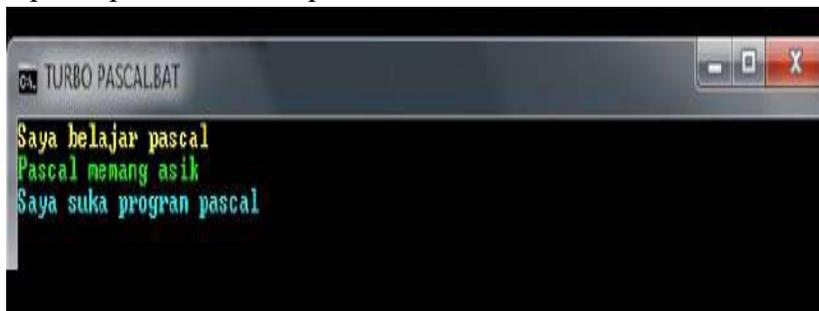
Atau bisa di tulis seperti ini :



```
PROGRAM CARA_MERUBAH_WARNA_TKRS;  
USES CRT;  
BEGIN  
  CLRSCR;  
  
  TEXTCOLOR(14);  
  WRITELN('Saya belajar pascal');  
  
  TEXTCOLOR(10);  
  WRITELN('Pascal memang asik');  
  
  TEXTCOLOR(11);  
  WRITELN('Saya suka program pascal');  
  
  READLN;  
END.
```

Gambar 9.2 Koding Merubah warna teks

Dapat di peroleh hasil seperti di bawah ini :



```
TURBO PASCAL.BAT  
Saya belajar pascal  
Pascal memang asik  
Saya suka program pascal
```

Gambar 9.3 Hasil Merubah warna teks

Berikut ada beberapa latihan soal silahkan kerjakan sesuai perintah :

1. Diketahui sebuah kotak mempunyai ukuran
Panjang Alas = 10 cm
Lebar Alas = 7 cm
Tinggi = 15 cm
2. Buatlah program untuk menghitung isi kotak dan luas permukaan kotak. Tampilan yang diinginkan adalah seperti berikut:
Ukuran Kotak = 10 cm * 7 cm * 15 cm
Isi kotak = 1050 cm³
3. Perusahaan Telepon di kota 'X' mempunyai tarif Rp. 150 per-pulsa. Setiap langganan dikenakan biaya langganan Rp. 25000,- per-bulan. Buatlah program untuk menghitung jumlah tagihan kepada salah seorang pelanggan
Masukan : Nama Langganan
Jumlah Pulsa
Keluaran: Nama Langganan Jumlah Tagihan
4. Soal sama dengan nomor 2, ubahlah bentuk layar masukan dan keluaran seperti berikut:
Nomor Langganan :
Nama :
Alamat :
Perincian Tagihan
Biaya Langganan : Rp.....
Pulsa : Rp. * JumlahPulsa
Total : Rp.....

BAB X

FUNGSI ARRAY

TUJUAN INSTRUSIONAL

1. Menjelaskan pengertian array
2. membuat deklarasi tipe data array, baik yang satu dimensi maupun yang dua dimensi.
3. Membuat contoh program sederhana dengan menggunakan tipe data array , baik yang satu dimensi maupun dua dimensi.

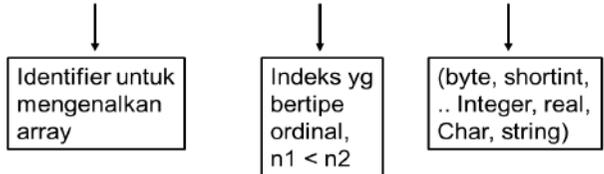
A. Pengertian Array

Array adalah sebuah struktur data yang terdiri atas banyak variabel dengan tipe data sama, dimana masing-masing elemen variabel mempunyai nilai indeks.

Setiap elemen array mampu untuk menyimpan satu jenis data (yaitu: variabel). Array selalu memuat tiga (3) komponen penting, yaitu :

1. Nnama array,
2. Indeks (yang bertipe ordinal), dan
3. Tipe data (sederhana) yang digunakan oleh array

Syntax : `Var Nama_array : array[n1..n2] of tipe_data;`



Contoh : `Var A : array[1..7] of real;`
`Var Nilai : array[1..40] of integer;`

4. Andaikan suatu data X mempunyai 10 anggota, yaitu :

X1 = 8	X6 = 9
X2 = 7	X7 = 8
X3 = 9	X8 = 9
X4 = 7	X9 = 9
X5 = 8	X10 = 7

Dalam Pascal, X yang mempunyai 10 data tersebut dapat dideklarasikan dengan array sebagai berikut :

```
Var X : array[1..10] of byte;
```

Kemudian untuk *mengisikan* atau *memanggil* tiap anggota array X tersebut ditulis perintah : X[i]

Misal : `X[4]` berarti anggota array X yang keempat.

`X[7] := 8;` berarti anggota array X yang ke tujuh di beri nilai 8

B. Jenis Array

Array dapat dibedakan menjadi 3 jenis, yaitu :

1. Array 1 (satu) dimensi, array yang terdiri dari satu baris dan banyak kolom atau satu kolom dan banyak baris.
2. Array 2 (dua) dimensi, array yang terdiri dari banyak baris dan banyak kolom
3. Array n dimensi, array yang tidak hanya terdiri dari baris dan kolom

C. Pendefinisian Array

1. Bentuk Umum Array

Nama_Array : array[n1..n2] OF < tipe data >;

2. Contoh Array

A : array[1..6] OF Integer;

1	2	3	4	5	6
---	---	---	---	---	---

Secara logika pendefinisian array di atas merupakan sekumpulan kotak , dimana tiap kotak mempunyai nilai indeks integer 1, 2, 3, ...,6 tiap elemen array ditandai dengan: A[1], A[2], A[3], A[4], A[5], A[6] untuk mengisi elemen array misal A[1]:=4;.

D. Sifat Array

1. Kumpulan data bertipe sama yang menggunakan nama yang sama
2. Sejumlah variabel dapat memakai nama yang sama
3. Antara satu variabel dengan variabel lain didalam array dibedakan berdasarkan subscript
4. Subscript adalah berupa bilangan di dalam kurung siku []
5. Melalui subscript elemen array dapat diakses

6. Elemen array tidak lain adalah masing-masing variabel di dalam array

Array merupakan struktur data yang statis, yaitu jumlah elemen yang ada harus ditentukan terlebih dahulu dan tak bisa di ubah saat program berjalan. Untuk menyatakan array dalam PASCAL kita harus terlebih dahulu: Mendefinisikan jumlah elemen array

Contoh. const N=10;

Type A= array [1..N] of integer

E. Array Satu Dimensi

Pendefinisian array secara umum adalah sebagai berikut: jika kita ingin membuat beberapa array dengan tipe/jenis yang sama, kita lebih baik jika mendeklarasikan dengan type selanjutnya dengan deklarasi var. Setiap elemen array dapat diakses melalui indeks. Indeks array (subscript) secara default dimulai dari 0.

1. Bentuk Array 1 Dimensi

```
• Var
  variabel_array : ARRAY[bawah..atas] of tipe_data;
```

Contoh

Indeks

```
Bil : Array[1..5] of Integer;
Gapok, Tunj : Array[1..10] of Real;
Nama : Array[1..20] of String;
Nama, Alamat : Array[1..20] of String;
```

2. Array dengan Type

```
Type
  nama_array = ARRAY[bawah..atas] of type data;

Var
  variabel_array : nama_array;
```

Contoh

```
Type
  a_string = ARRAY[1..20] OF STRING;
  a_real   = ARRAY[1..10] OF REAL;
  A_Bil    = ARRAY[1..10] Of Integer;
Var
  nama, alamat : a_string;
  gpok, tunj, gb : a_real;
  Bil : A_bil;
```

3. Contoh Program
Tanpa Array

```

PROGRAM menampilkan_angka_dengan_array;
USES CRT;
VAR
  BIL : INTEGER;
  I : INTEGER;
BEGIN
  CLRSCR;
  FOR I := 1 TO 5 DO
  BEGIN
    WRITE<'MASUKAN BILANGAN '>;
    READLN<BIL>;
  END;

  FOR I := 1 TO 5 DO
    WRITE<BIL,' '>;
    READLN;
  END.

```

C:\A TPX

MASUKAN	BILANGAN	=45
MASUKAN	BILANGAN	=90
MASUKAN	BILANGAN	=20
MASUKAN	BILANGAN	=10
MASUKAN	BILANGAN	=55
55	55	55
55	55	55

Dengan Array

```

PROGRAM menampilkan_angka_dengan_array;
USES CRT;
VAR
  BIL : ARRAY[1..5] OF INTEGER;
  I,N : INTEGER;
BEGIN
  CLRSCR;
  FOR I := 1 TO 5 DO
  BEGIN
    WRITE<'MASUKAN BILANGAN '>;
    READLN<BIL[I]>;
  END;

  FOR I := 1 TO 5 DO
    WRITE<BIL[I],' '>;
    READLN;
  END.

```

C:\A TPX

MASUKAN	BILANGAN	=55
MASUKAN	BILANGAN	=45
MASUKAN	BILANGAN	=70
MASUKAN	BILANGAN	=20
MASUKAN	BILANGAN	=90
55	45	70
20	90	

Gambar 10.1 Program Tanpa dan Dengan Array


```

c:\ TPX
PENGOLAHAN DATA PENJUALAN BARANG
BERAPA JUMLAH DATA BARANG [max : 25] ?3
DATA KE = 1
NAMA BARANG      ?DJARUM SUPER
HARGA SATUAN     ?9000
JUMLAH TERJUAL?5
TOTAL BAYAR =    45000.00

DATA KE = 2
NAMA BARANG      ?EXTRA JOSS
HARGA SATUAN     ?4000
JUMLAH TERJUAL?3
TOTAL BAYAR =    12000.00

DATA KE = 3
NAMA BARANG      ?KUKU BIMA
HARGA SATUAN     ?5000
JUMLAH TERJUAL??

c:\ TPX
DAFTAR PENJUALAN BARANG
+-----+-----+-----+-----+-----+
|INO | NAMA BARANG | HARGAE | QTY | TOTAL BAYAR|
+-----+-----+-----+-----+-----+
| I 1| KUKU BIMAI | 5000I  | 7I  | 35000.00I  |
| I 2| KUKU BIMAI | 5000I  | 7I  | 35000.00I  |
| I 3| KUKU BIMAI | 5000I  | 7I  | 35000.00I  |
+-----+-----+-----+-----+-----+
| I  | JUMLAH TOTAL BAYAR |          |      | 105000.00I |
+-----+-----+-----+-----+-----+

```

Gambar 10.3 Hasil Program Tanpa Array

Kesimpulan Hasil Program

Dari input nama barang yang dimasukkan diatas sebanyak 3 buah yaitu djarum super, extra joss, kuku bima dan jumlah masing masing 5,3 dan 7 serta harga masing masing 9000, 12000 dan 5000 tetapi ketika data di cetak yang tampil semua (ketiganya) adalah nama barang extra joss dan harga 5000 serta jumlah 5 semua. Ini menunjukkan kalau tidak memakai array maka data akan dicetak sebanyak permintaan tetapi data yang tercetak yang terakhir saja dan jumlah sesuai permintaan.


```

BERAPA JUMLAH DATA BARANG [max : 25] ?3
DATA KE = 1
NAMA BARANG      ?DJARUM SUPER
HARGA SATUAN     ?12000
JUMLAH TERJUAL?5
TOTAL BAYAR =    60000.00

DATA KE = 2
NAMA BARANG      ?EXTRA JOSS
HARGA SATUAN     ?5000
JUMLAH TERJUAL?4
TOTAL BAYAR =    20000.00

DATA KE = 3
NAMA BARANG      ?KUKU BIMA
HARGA SATUAN     ?7000
JUMLAH TERJUAL?7
TOTAL BAYAR =    49000.00

TEKAN ENTER UNTUK LIHAT DAFTAR...

```

TPX						
DAFTAR PENJUALAN BARANG						
INO	I	NAMA BARANG	I HARGA	I QTY	I TOTAL BAYARI	
I	1I	DJARUM SUPERI	12000I	5I	60000.00I	
I	2I	EXTRA JOSSI	5000I	4I	20000.00I	
I	3I	KUKU BIMAI	7000I	7I	49000.00I	
I	JUMLAH TOTAL BAYAR				:	129000.00I

Gambar 10.5 Hasil Program Dengan Array

Program kedua hasil modifikasi dari program yang pertama, dimana pada pedefinisaian variabel di gunakan array.

Dengan pendefinisian variabel array maka hasil programnya akan berbeda, dari hasil input sebanyak 3 data maka outpunya juga sebanyak tiga jadi sesuai dengan yang kita inginkan.

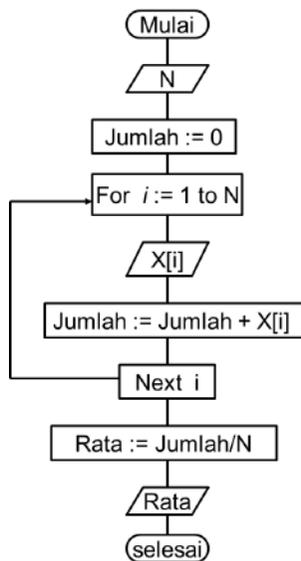
Jadi dengan array ini data yang banyak dapat di definikan / deklarasikan dengan menjadi lebih mudah/pendek, jadi tidak perlu mendefiniskan satu persatu(record/record).

F. Array Dimensi Tunggal

Array yang hanya mempunyai satu buah indeks. Persoalan-persoalan dalam statistic deskriptif biasanya dapat ditampung dalam pemakaian array dimensi tunggal.

Contoh : Buat program untuk menghitung rata-rata dari N buah data !

1. Analisis masalah :
 - a. Pertama, perlu mengetahui banyaknya data (N) terlebih dahulu.
 - b. Kedua, perlu mengetahui data ke 1 sampai data ke N
 - c. Ketiga, menjumlahkan nilai data ke 1 sampai data ke N
 - d. Akhirnya, rata-rata adalah jumlah dari N data tersebut dibagi dengan N.
2. *Flowchart* mencari rata-rata
 Algoritma :
 - a. Input N
 - b. Untuk I = 1 sampai N, input X[i]
 - c. Jumlah – X[1] + X[2]+ .. + X[N]
 - d. Rata-rata = Jumlah / N
3. Dari N buah data.



Contoh program :

```

Program Rata_rata;
uses Crt;
var i, N : byte;
    Jumlah, Rata : real;
    X : array[1..255] of real;
Begin
  Clrscr;
  Write('banyaknya data (maksimum 255) ?');
  readln(N);
  Jumlah := 0;
  For i := 1 to N do
  Begin
    Write('Data ke ', i, ' = '); readln(X[i]);
    Jumlah := Jumlah + X[i];
  End;
  Rata := Jumlah / N;
  writeln('banyaknya data = ', N);
  writeln('Data-data dari X[i]');
  for i := 1 to N do
  begin
    write(x[i] ', ');
  end;
  writeln('Jumlah nilai data = ', Jumlah :7:2);
  writeln('Rata-rata data = ', Rata:6:2);
end.
  
```

G. Data Sorting

Dalam persoalan statistika, banyak diperlukan *data sorting* (pengurutan data) terlebih dahulu untuk menjawab beberapa permasalahan, seperti pencarian median atau kuartil.

Sorting juga diperlukan misalnya untuk mencari ranking. Salah satu algoritma untuk melakukan sorting adalah : *linearsort*, yaitu :

```

(1) input N
(2) for i := 1 to N, input X[i]
(3) for i := 1 to N-1
    for j := 1 to N-1
        if X[j] > X[j+1]
            swap (X[j], X[j+1])
(4) for i := 1 to N, output X[i]

```

→ Untuk sorting dari kecil ke besar

Jika data ingin diurutkan dari besar ke kecil, langkah (3) diganti :
If X[j] < X[j + 1]

Problem :

Buat program untuk mencari median dari N buah data !

Algoritma dari problem tersebut adalah :

1. Input N
2. untuk i = 1 sampai N, input X[i]
3. Sorting X[i]
4. Jika N ganjil, median = $X[(N+1)/2]$. Selesai.
5. Jika N genap, median = $(X[N/2] + X[(N/2)+1])/2$. selesai.

Langkah 3. sorting X[i] mengikuti algoritma sorting yang telah disebutkan

sebelumnya.

<pre> Program Median; uses Crt; var i, J, N : byte; sementara, Median : real; X : array[1..255] of real; Begin Clrscr; Write('banyaknya data (maksimum 255) ?'); readln(N); {proses input data} For i := 1 to N do Begin Write('Data ke ', i, ' = ');readln(X[i]); End; {proses sorting } for i := 1 to N-1 do for J := 1 to N-1 do if X[J] > X[J+1] then begin sementara := X[J]; X[J] := X[J+1]; X[J+1] := sementara; end; </pre>	<pre> {menampilkan data yang diurutkan} clrscr; writeln; writeln('banyaknya data = ', N); writeln('Data-data dari X[i]'); for i := 1 to N do begin write(x[i] ', '); end; {menghitung / menampilkan median} if N mod 2 = 0 then median := (X[] + X[]) / 2; if N mod 2 <> 0 then median := X[]; writeln('Median dari data-data tersebut = ', median:4:2); Readln; end. </pre>
---	---

TUJUAN INSTRUSIONAL

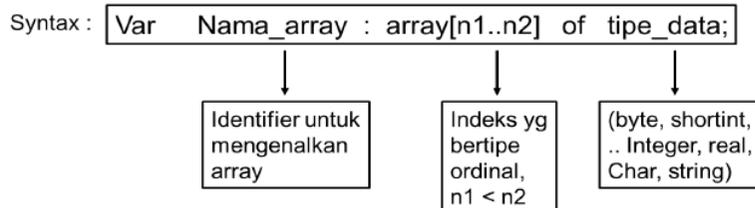
1. Menjelaskan pengertian array
2. Membuat deklarasi tipe data array, baik yang satu dimensi maupun yang dua dimensi.
3. Membuat contoh program sederhana dengan menggunakan tipe data array , baik yang satu dimensi maupun dua dimensi.

A. Pengertian Array

Array adalah sebuah struktur data yang terdiri atas banyak variabel dengan tipe data sama, dimana masing-masing elemen variabel mempunyai nilai indeks.

Setiap elemen array mampu untuk menyimpan satu jenis data (yaitu: variabel). Array selalu memuat tiga (3) komponen penting, yaitu :

1. Nama array,
2. Indeks (yang bertipe ordinal), dan
 - a. Tipe data (sederhana) yang digunakan oleh array



Contoh : `Var A : array[1..7] of real;`
`Var Nilai : array[1..40] of integer;`

Andaikan suatu data X mempunyai 10 anggota, yaitu :

X1 = 8	X6 = 9
X2 = 7	X7 = 8
X3 = 9	X8 = 9
X4 = 7	X9 = 9
X5 = 8	X10 = 7

Dalam Pascal, X yang mempunyai 10 data tersebut dapat dideklarasikan dengan array sebagai berikut :

```
Var X : array[1..10] of byte;
```

Kemudian untuk *mengisikan* atau *memanggil* tiap anggota array X tersebut ditulis perintah : X[i]

Misal : `X[4]` berarti anggota array X yang keempat.

`X[7] := 8;` berarti anggota array X yang ke tujuh di beri nilai 8

B. Jenis Array

Array dapat dibedakan menjadi 3 jenis, yaitu :

1. Array 1 (satu) dimensi, array yang terdiri dari satu baris dan banyak kolom atau satu kolom dan banyak baris.
2. Array 2 (dua) dimensi, array yang terdiri dari banyak baris dan banyak kolom
3. Array n dimensi, array yang tidak hanya terdiri dari baris dan kolom

C. Pendefinisian Array

Bentuk Umum Array

Nama_Array : array[n1..n2] OF < tipe data >;

Contoh Array

A : array[1..6] OF Integer;

1	2	3	4	5	6
---	---	---	---	---	---

Secara logika pendefinisian array di atas merupakan sekumpulan kotak , dimana tiap kotak mempunyai nilai indeks integer 1, 2, 3, ...,6 tiap elemen array ditandai dengan: A[1], A[2], A[3], A[4], A[5], A[6] untuk mengisi elemen array misal A[1]:=4;.

D. Sifat Array

1. Kumpulan data bertipe sama yang menggunakan nama yang sama
2. Sejumlah variabel dapat memakai nama yang sama
3. Antara satu variabel dengan variabel lain didalam array dibedakan berdasarkan subscript
4. Subscript adalah berupa bilangan di dalam kurung siku []
5. Melalui subscript elemen array dapat diakses

6. Elemen array tidak lain adalah masing-masing variabel di dalam array

Array merupakan struktur data yang statis, yaitu jumlah elemen yang ada harus ditentukan terlebih dahulu dan tak bisa di ubah saat program berjalan. Untuk menyatakan array dalam PASCAL kita harus terlebih dahulu:

Mendefinisikan jumlah elemen array

Contoh. const N=10;

Type A= array [1..N] of integer

E. Array Satu Dimensi

Pendefinisian array secara umum adalah sebagai berikut: jika kita ingin membuat beberapa array dengan tipe/jenis yang sama, kita lebih baik jika mendeklarasikan dengan type selanjutnya dengan deklarasi var. Setiap elemen array dapat diakses melalui indeks. Indeks array (subscript) secara default dimulai dari 0.

1. Bentuk Array 1 Dimensi

```
• Var
  variabel_array : ARRAY[bawah..atas] of tipe_data;
```

Contoh Indeks

```
Bil : Array[1..5] of Integer;
Gapok, Tunj : Array[1..10] of Real;
Nama : Array[1..20] of String;
Nama, Alamat : Array[1..20] of String;
```

2. Array dengan Type

```
Type
  nama_array = ARRAY[bawah..atas] of type data;

Var
  variabel_array : nama_array;
```

Contoh

```
Type
  a_string = ARRAY[1..20] OF STRING;
  a_real = ARRAY[1..10] OF REAL;
  A_Bil = ARRAY[1..10] Of Integer;

Var
  nama, alamat : a_string;
  gpok, tunj, gb : a_real;
  Bil : A_bil;
```

3. Contoh Program

```
PROGRAM menampilkan_angka_dengan_array;
USES CRT;
VAR
  BIL : INTEGER;
  I : INTEGER;
BEGIN
  CLRSCR;
  FOR I := 1 TO 5 DO
  BEGIN
    WRITE('MASUKAN BILANGAN ');
    READLN(BIL);
  END;

  FOR I := 1 TO 5 DO
    WRITE(BIL, ' ');
  READLN;
END.
```

TPX

MASUKAN	BILANGAN	=45
MASUKAN	BILANGAN	=90
MASUKAN	BILANGAN	=20
MASUKAN	BILANGAN	=10
MASUKAN	BILANGAN	=55
55	55	55
55	55	55

Gambar 11.1 Koding Tanpa Array

```
PROGRAM menampilkan_angka_dengan_array;
USES CRT;
VAR
  BIL : ARRAY[1..5] OF INTEGER;
  I, N : INTEGER;
BEGIN
  CLRSCR;
  FOR I := 1 TO 5 DO
  BEGIN
    WRITE('MASUKAN BILANGAN ');
    READLN(BIL[I]);
  END;

  FOR I := 1 TO 5 DO
    WRITE(BIL[I], ' ');
  READLN;
END.
```

TPX

MASUKAN	BILANGAN	=55
MASUKAN	BILANGAN	=45
MASUKAN	BILANGAN	=70
MASUKAN	BILANGAN	=20
MASUKAN	BILANGAN	=90
55	45	70
20	90	

Gambar 11.2 Hasil Tanpa Array

```

PROGRAM CONTOH_BUKUAN_PROGRAM_ARRAY;
USES CRT;

CONST MAKSIMAL = 25;

VAR
    NAMA      : STRING;
    HRG,JML,TOT : REAL;
    N,X       : BYTE;
    GRAND     : REAL;

BEGIN
    CLRSCR;
    GRAND := 0;
    Writeln('PENGOLAHAN DATA PENJUALAN BARANG');
    Writeln;

    Write('BERAPA JUMLAH DATA BARANG [max : 25] ?'); Readln(N);
    IF N > MAKSIMAL THEN
        BEGIN
            Writeln('ERROR:MAKSIMAL JUMLAH DATA : ',MAKSIMAL);
            Readln;
            Halt;
            End;
        FOR X := 1 TO N DO
            BEGIN
                Writeln('DATA KE = ', X);
                Writeln;
                Write('NAMA BARANG  ?'); Readln(NAMA);
                Write('HARGA SATUAN  ?'); Readln(HRG);
                Write('JUMLAH TERJUAL?'); Readln(JML);

                TOT := HRG * JML;

                Writeln('TOTAL BAYAR = ',TOT:10:2);
                Writeln;
            END;

            Writeln('TEKAN ENTER UNTUK LIHAT DAFTAR...'); Readln;

            CLRSCR;
            Writeln('DAFTAR PENJUALAN BARANG');
            Writeln;
            (123456789012345678901234567890123456789012345678901234);
            Writeln('-----+');
            Writeln('INO I NAMA BARANG      I HARGA   I QTY I TOTAL BAYAR');
            Writeln('-----+');

            FOR X := 1 TO N DO
                BEGIN
                    Write('I',X:3);
                    Write('I',NAMA :20);
                    Write('I',HRG :10:0);
                    Write('I',JML :5:0);
                    Write('I',TOT :12:2);
                    Writeln('I');

                    GRAND := GRAND + TOT;
                END;
            Writeln('-----+');
            Writeln('I JUMLAH TOTAL BAYAR      :',GRAND :12:2, 'I');
            Writeln('-----+');
            Readln;
        END.

```

Gambar 11.3 Koding Program Tanpa Array

```

C:\ TPX
PENGOLAHAN DATA PENJUALAN BARANG
BERAPA JUMLAH DATA BARANG [max : 25] ?3
DATA KE = 1
NAMA BARANG ?DJARUM SUPER
HARGA SATUAN ?9000
JUMLAH TERJUAL?5
TOTAL BAYAR = 45000.00
DATA KE = 2
NAMA BARANG ?EXTRA JOSS
HARGA SATUAN ?4000
JUMLAH TERJUAL?3
TOTAL BAYAR = 12000.00
DATA KE = 3
NAMA BARANG ?KUKU BIMA
HARGA SATUAN ?5000
JUMLAH TERJUAL?7

C:\ TPX
DAFTAR PENJUALAN BARANG
+-----+-----+-----+-----+
| I NO | I NAMA BARANG | I HARGAE | I QTY | I TOTAL BAYARI |
+-----+-----+-----+-----+
| I 1I | KUKU BIMAI | 5000I | 7I | 35000.00I |
| I 2I | KUKU BIMAI | 5000I | 7I | 35000.00I |
| I 3I | KUKU BIMAI | 5000I | 7I | 35000.00I |
+-----+-----+-----+-----+
| I JUMLAH TOTAL BAYAR | : | 105000.00I |
+-----+-----+-----+-----+

```

Gambar 11.4 Hasil Program Tanpa Array

Kesimpulan Hasil Program

Dari input nama barang yang dimasukkan diatas sebanyak 3 buah yaitu djarum super, extra joss, kuku bima dan jumlah masing masing 5,3 dan 7 serta harga masing masing 9000, 12000 dan 5000 tetapi ketika data di cetak yang tampil semua (ketiganya) adalah nama barang extra joss dan harga 5000 serta jumlah 5 semua. Ini menunjukkan kalau tidak memakai array maka data akan dicetak sebanyak permintaan tetapi data yang tercetak yang terakhir saja dan jumlah sesuai permintaan.

```

PROGRAM CONTOH_BUKUAN_PROGRAM_ARRAY;
USES CRT;
CONST MAKSIMAL = 25;
TYPE
  A_REAL = ARRAY[1..25] OF REAL;
VAR
  NAMA : ARRAY[1..25] OF STRING;
  HRG, JML, TOT : A_REAL;
  N, X : BYTE;
  GRAND : REAL;
BEGIN
  CLRSCR;
  GRAND := 0;
  WRITELN('PENGOLAHAN DATA PENJUALAN BARANG');
  WRITELN;
  WRITELN('BERAPA JUMLAH DATA BARANG [max : 25] ??'); READLN(N);
  IF N > MAKSIMAL THEN
  BEGIN
    WRITELN('ERROR:MAKSIMAL JUMLAH DATA : ', MAKSIMAL);
    READLN;
    HALT;
  END;
  FOR X := 1 TO N DO
  BEGIN
    WRITELN('DATA KE = ', X);
    WRITELN;
    WRITE('NAMA BARANG '); READLN(NAMA[X]);
    WRITE('HARGA SATUAN '); READLN(HRG[X]);
    WRITE('JUMLAH TERJUAL? '); READLN(JML[X]);

    TOT[X] := HRG[X] * JML[X];
    WRITELN('TOTAL BAYAR = ', TOT[X]:10:2);
    WRITELN;
  END;
  WRITELN('TEKAN ENTER UNTUK LIHAT DAFTAR...'); READLN;
  (laporan record data yang di masukkan )
  CLRSCR;
  WRITELN('DAFTAR PENJUALAN BARANG');
  WRITELN;
  WRITELN('123456789012345678901234567890123456789012345678901234');
  WRITELN('-----+-----+-----+-----+');
  WRITELN('INO I NAMA BARANG I HARGA I QTY I TOTAL BAYAR');
  WRITELN('-----+-----+-----+-----+');
  FOR X := 1 TO N DO
  BEGIN
    WRITE('I', X:3);
    WRITE('I', NAMA[X] :20);
    WRITE('I', HRG[X] :10:0);
    WRITE('I', JML[X] :5:0);
    WRITE('I', TOT[X] :12:2);
    WRITELN('I');

    GRAND := GRAND + TOT[X];
  END;
  WRITELN('-----+-----+-----+-----+');
  WRITELN('I JUMLAH TOTAL BAYAR :', GRAND :12:2, 'I');
  WRITELN('-----+-----+-----+-----+');
  READLN;
  END.

```

Gambar 11.5 Program Dengan Array

```

BERAPA JUMLAH DATA BARANG [max : 25] ?3
DATA KE = 1
NAMA BARANG ?DJARUM SUPER
HARGA SATUAN ?12000
JUMLAH TERJUAL?5
TOTAL BAYAR = 60000.00

DATA KE = 2
NAMA BARANG ?EXTRA JOSS
HARGA SATUAN ?5000
JUMLAH TERJUAL?4
TOTAL BAYAR = 20000.00

DATA KE = 3
NAMA BARANG ?KUKU BIMA
HARGA SATUAN ?7000
JUMLAH TERJUAL?7
TOTAL BAYAR = 49000.00
TEKAN ENTER UNTUK LIHAT DAFTAR...

```

TPX

DAFTAR PENJUALAN BARANG

INO	I	NAMA BARANG	I HARGA	I QTY	I TOTAL BAYAR
I 1I		DJARUM SUPER	12000I	5I	60000.00I
I 2I		EXTRA JOSSI	5000I	4I	20000.00I
I 3I		KUKU BIMAI	7000I	7I	49000.00I
I JUMLAH TOTAL BAYAR					: 129000.00I

Gambar 11.6 Hasil Program Dengan Array

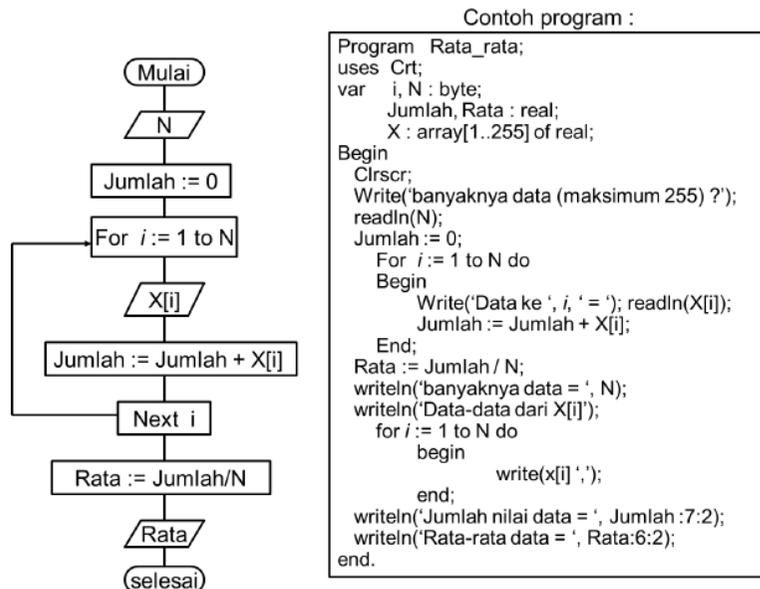
Program kedua hasil modifikasi dari program yang pertama, dimana pada pedefinisian variabel di gunakan array. Dengan pendefinisian variabel array maka hasil programnya akan berbeda, dari hasil input sebanyak 3 data maka outpunya juga sebanyak tiga jadi sesuai dengan yang kita inginkan. Jadi dengan array ini data yang banyak dapat di definikan / deklarasikan dengan menjadi lebih mudah/pendek, jadi tidak perlu mendefinisikan satu persatu(record/record).

F. Array Dimensi Tunggal

Array yang hanya mempunyai satu buah indeks. Persoalan-persoalan dalam statistic deskriptif biasanya dapat ditampung dalam pemakaian array dimensi tunggal.

Contoh : Buat program untuk menghitung rata-rata dari N buah data !

1. Analisis masalah :
 - Pertama, perlu mengetahui banyaknya data (N) terlebih dahulu.
 - Kedua, perlu mengetahui data ke 1 sampai data ke N
 - Ketiga, menjumlahkan nilai data ke 1 sampai data ke N
 - Akhirnya, rata-rata adalah jumlah dari N data tersebut dibagi dengan N.
2. *Flowchart* mencari rata-rata dari N buah data
 - a. Input N
 - b. Untuk I = 1 sampai N, input X[i]
 - c. Jumlah – X[1] + X[2]+ .. + X[N]
 - d. Rata-rata = Jumlah /



Gambar 11.7 Program Rata-rata dengan array

G. Data Sorting

Dalam persoalan statistika, banyak diperlukan *data sorting* (pengurutan data) terlebih dahulu untuk menjawab beberapa permasalahan, seperti pencarian median atau kuartil. Sorting juga diperlukan misalnya untuk mencari ranking. Salah satu algoritma untuk melakukan sorting adalah : *linearsort*, yaitu :

```

(1) input N
(2) for i := 1 to N, input X[i]
(3) for i := 1 to N-1
    for j := 1 to N-1
        if X[j] > X[j+1]
            swap (X[j], X[j+1])
(4) for i := 1 to N, output X[i]

```

→ Untuk sorting dari kecil ke besar

Jika data ingin diurutkan dari besar ke kecil, langkah (3) diganti :
 If $X[j] < X[j + 1]$

Problem :

Buat program untuk mencari median dari N buah data !

Algoritma dari problem tersebut adalah :

1. Input N
2. Untuk $i = 1$ sampai N, input $X[i]$
3. Sorting $X[i]$
4. Jika N ganjil, median = $X[(N+1)/2]$. Selesai.
5. Jika N genap, median = $(X[N/2] + X[(N/2)+1])/2$. selesai.

Langkah (3), sorting $X[i]$ mengikuti algoritma sorting yang telah disebutkan sebelumnya.

<pre> Program Median; uses Crt; var i, J, N : byte; sementara, Median : real; X : array[1..255] of real; Begin Clrscr; Write('banyaknya data (maksimum 255) ?'); readln(N); {proses input data} For i := 1 to N do Begin Write('Data ke ', i, ' = ');readln(X[i]); End; {proses sorting } for i := 1 to N-1 do for J := 1 to N-1 do if X[J] > X[J+1] then begin sementara := X[J]; X[J] := X[J+1]; X[J+1] := sementara; end; </pre>	<pre> {menampilkan data yang diurutkan} clrscr; writeln; writeln('banyaknya data = ', N); writeln('Data-data dari X[i]'); for i := 1 to N do begin write(x[i] ', '); end; {menghitung / menampilkan median} if N mod 2 = 0 then median := (X[] + X[]) / 2; if N mod 2 <> 0 then median := X[]; writeln('Median dari data-data tersebut = ', median:4:2); Readln; end. </pre>
---	--

Gambar 11.8 Program Media

H. Dua Dimensi

Array dua dimensi merupakan array yang terdiri dari m buah baris dan n buah kolom. Bentuknya dapat berupa matriks atau tabel. Digunakan untuk berbagai keperluan, misalnya dalam bentuk tabel, matrik dan sebagainya. Bentuk Umum :

Baris Kolom

• Var
var_array : ARRAY[indeks1, indeks2] of tipe_data;

Contoh

• Var
Bil : ARRAY[1..3, 1..5] of Integer;

```
PROGRAM Contoh_array_dua_dimensi;
USES CRT;
CONST x = 25 ;
      y = 10 ;
TYPE
  a_string1 = ARRAY[0..x] OF STRING[15];
  a_string2 = ARRAY[0..y] OF STRING[10];
  a_real    = ARRAY[0..x, 1..y] OF REAL;
VAR
  nm : a_string1;
  mk : a_string2;
  sc : a_real;
  nx,ny, a,b : byte;
  tot,rt : real;
  <total dan rata-rata >
BEGIN
  CLRSCR;
  WRITELN('PENGOLAHAN DATA NILAI MAHASISWA');
  WRITELN;
  WRITE('Berapa jml mahasiswa (max: ',x,' ) ? '); READLN( nx );
  WRITE('Berapa jml mata kuliah (max: ',y,' )? '); READLN( ny );
  IF (nx > x) OR (ny > y) THEN HALT;

  Writeln;
  Writeln( ' *** Input nama mahasiswa dulu');

  FOR a := 1 TO nx DO
  BEGIN
    Write('Nama mahasiswa ke : ', a , ' ? ');
    READLN( nm[a] );
  END;

  Writeln;
  Writeln( ' *** Input nama mata kuliah terlebih dahulu ');
```

```

FOR b := 1 TO ny DO
  BEGIN
    Write<'Nama matakuliah ke : ', b, ' ? ' >;
    READLN< mk[b] >;
  END;

Writeln;
Writeln<'*** Input data-data nilai semua mata kuliah'>;
FOR a:= 1 To nx DO
  BEGIN
    writeln<'Data² nilai u/ mahasiswa bernama : ', nm[a] >;

    FOR b := 1 TO ny DO
      BEGIN
        WRITE < mk[b] :15, ' => Nilai ? ' >;
        READLN< sc[a,b] >;
      END;

    Writeln;
  END;
END;

WRITE<'TEKAN ENTER U/ LIHAT DAFTAR...'>; READLN;

CLRSCLR;
Writeln<'Daftar Nilai Mahasiswa'>;
Writeln;

< 12 123456789012345 >
WRITE<'!No!Nama Mahasiswa !'>;
FOR b := 1 TO ny DO
  WRITE< mk[b] :10, '!' >;
  < 123456 123456 >
WRITELN<' Total! Rata !'>;

FOR a:= 1 TO nx DO
  BEGIN
    Write<'!' , a:2>;
    Write<'!' , nm[a]:15>;

    tot := 0; < setiap mahasiswa total diset = 0 dulu >
    FOR b:= 1 TO ny DO

      Begin
        write<'!' , sc[a,b] :10:2>;
        tot := tot + sc[a,b];
      End;
      rt := tot / b; < hitung rata-rata setelah total diketahui >
      write<'!' , tot :6:2 >;
      write<'!' , rt:6:2 >;
      writeln<'!'>;
    END;
  READLN;
END.

```

Gambar 11.9 Koding Array 2 Dimensi

```

PENGOLAHAN DATA NILAI MAHASISWA
Berapa jml mahasiswa <max:25 > ? 2
Berapa jml mata kuliah <max:10 >? 3

*** Input nama mahasiswa dulu
Nama mahasiswa ke : 1 ? Budi
Nama mahasiswa ke : 2 ? Tini

*** Input nama mata kuliah terlebih dahulu
Nama matakuliah ke : 1 ? Agama
Nama matakuliah ke : 2 ? Bhs Indonesia
Nama matakuliah ke : 3 ? Pacasila

*** Input data-data nilai semua mata kuliah
Data² nilai u/ mahasiswa bernama : Budi
    Agama => Nilai ? 90
    Bhs Indone => Nilai ? 90
    Pacasila => Nilai ? 85

Data² nilai u/ mahasiswa bernama : Tini
    Agama => Nilai ? 85
    Bhs Indone => Nilai ? 85
    Pacasila => Nilai ? 90

Daftar Nilai Mahasiswa

!No!Nama Mahasiswa !   Agama!Bhs Indone!   Pacasila! Total ! Rata !
! 1!      Budi!      90.00!   90.00!   85.00!265.00! 88.33!
! 2!      Tini!      85.00!   85.00!   90.00!260.00! 86.67!

```

Gambar 11.10 Hasil Program 2 Dimensi

Bentuk Array 2 dimensi

```

• Var
  Bil : ARRAY[1..3, 1..5] of Integer;

```

	1	2	3	4	5
1	1,1	1,2	1,3	1,4	1,5
2	2,1	2,2	2,3	2,4	2,5
3	3,1	3,2	3,3	3,4	3,5

Terdiri dari 3 baris dan 5 kolom

Jadi : indeks-1 menyatakan baris dan indeks-2 menyatakan kolom.

Array yang hanya mempunyai dua buah (atau lebih) indeks. Dalam matematika, suatu matriks misalnya mempunyai dua buah indeks, yaitu indeks baris dan indeks kolom. Untuk matriks $A = (a_{ij})$, dalam pascal deklarasi dengan array sebagai berikut :

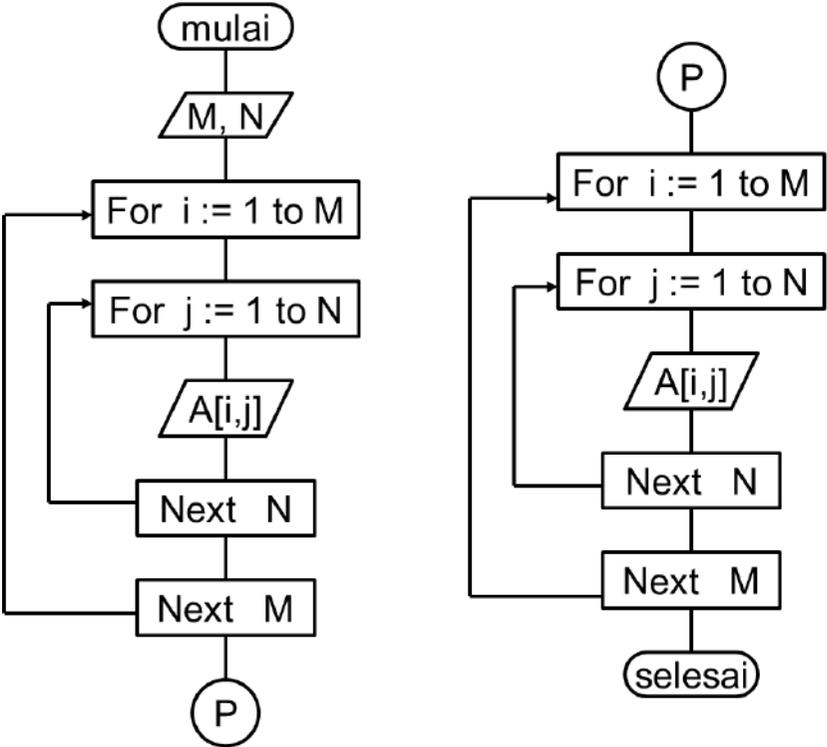
Var A : array[1..20, 1..20] of real;

Indeks
Pertama
i

Indeks
kedua
j

Pemanggilannya/penunjukkan : $A[i,j]$

Flowchart input/output Matriks A berdimensi mxn



Gambar 11.11 Flowchart input output matriks

Contoh program:

```
{Input dan menampilkan matriks A}
uses crt;
var i, j, m, n : byte;
    A : array[1..255, 1..255] of real;
Begin
  Clrscr;
  Write('MATRIKS');
  Write('Banyaknya baris matriks ? ');readln(m);
  Write('banyaknya kolom matriks ? ');readln(n);

  {proses memasukkan data matriks A}
  writeln('Data-data matriks A');
  for i := 1 to M do
  begin
    for j := 1 to n do
    begin
      write('A(' , i , ' , ' , J , ' ) ? ');
      readln(A[i,j]);
    end;
    writeln;
  end;
end;
```

```
{proses menampilkan matriks A}
writeln('matriks A = ');
writeln;
for i := 1 to M do
begin
  for j := 1 to n do
  begin
    write(A[i,j]);
  end;
  writeln;
end;
readln;
end.
```

I. Array Multi-Dimensi

Array multi-dimensi merupakan array yang mempunyai ukuran lebih dari dua. Bentuk pendeklarasian array sama saja dengan array dimensi satu maupun array dimensi dua.

1. Bentuk umum pendefinisian array berdimensi **tiga** :

Type nama_array[indeks1][indeks2][indeks3];

Contoh :

int huruf[2][8][8];

Merupakan pendefinisia data huruf sebagai array berdimensi tiga.

Gambaran

array berdimensi tiga sebagai berikut :

Int Huruf[2][8][8];

J. Kesimpulan

Array memiliki beberapa cirri-ciri khusus antara lain bahwa array merupakan kumpulan data bertipe sama yang menggunakan nama yang sama. Sejumlah variabel dapat memakai nama yang sama. Antara satu variabel dengan variabel lain didalam array dibedakan berdasarkan subscript. Subscript adalah berupa bilangan di dalam kurung siku []. Melalui subscript elemen array dapat diakses. Elemen array tidak lain adalah masing-masing variabel di dalam array. Array dapat dibedakan

menjadi beberapa ukuran atau dimensi, yaitu array berdimensi satu, dimensi dua dan multi dimensi (dimensi banyak). Pada array berdimensi satu digunakan pada kumpulan data bertipe sama yang menggunakan nama yang sama, misalnya array bernama bunga = { melati, mawar, dahlia..}. Array dimensi dua diimplementasikan pada pembuatan matrik. Array dimensi banyak dalam aplikasinya tidak terlalu sering dipakai. Namun demikian tidak berarti array dimensi banyak tidak dipelajari.

Latihan

- 1) Buatlah algoritma dan program untuk menyimpan data berikut ke dalam larik :
10 2 5 3 8 9 2 9 5
kemudian carilah bilangan yang terbesar !
- 2) Buatlah algoritma dan program untuk membaca data secara berulang dari keyboard dan meletakkannya ke dalam suatu larik. Jumlah maksimal yang dapat dimasukkan ke dalam larik adalah 10 buah. Setelah itu tampilkan seluruh data yang dimasukkan dari keyboard tadi.
- 3) Buatlah program untuk melakukan penjumlahan dan pengurangan duabuaah matrik A dan Matrik B.

BAB XII

OPERASI FILE

TUJUAN INSTRUSIONAL

1. Menjelaskan pembentukan operasi file
2. Menjelaskan mengenai pengiriman parameter dalam fungsi.
3. Menjelaskan cara pemanggilan pada procedure tersarang.
4. Menjelaskan cara fungsi memanggil dirinya sendiri
5. Membuat contoh program sederhana dengan menggunakan fungsi

A. Pengertian File/Berkas

File adalah kumpulan byte-byte yang disimpan dalam media penyimpanan. Merupakan komponen yang bertipe data sama, yang jumlahnya tidak tentu, yang dapat ditambah atau dikurangi jika dibutuhkan.

File pada Pascal dikenal dalam 3 jenis, yaitu :

1. File Text
2. File bertipe
3. File tidak bertipe

B. Manipulasi File/Berkas

Selain kita akan mempelajari tentang bagaimana membuat sebuah file atau menambahkan isi suatu file, kita dapat pula melakukan manipulasi File, yaitu :

1. Menggunakan parameter Mengenai Atribut File
2. Menghapus file
3. Mengubah nama file
4. Mengenai tanggal dan waktu file
5. Mencari file
6. Mengecek keberadaan file
7. Memberikan directory file

C. Tahapan Operasi File

1. Membuka/mengaktifkan file

Contoh Bentuk pengaktifan file :

```
if (pf = fopen("COBA.TXT", "w") == NULL)
{
printf("File tidak dapat diciptakan !\n");
exit(1); //keluar dari program
}
```

Keterangan :

> pf akan diisi dengan keluaran dari fungsi *fopen()*.

> Jika nilainya NULL, maka akan mencetak "File tidak dapat diciptakan", setelah itu program dihentikan.

2. Melaksanakan proses file

3. Menutup file

Apabila file sudah tidak diproses lagi, maka file tersebut ditutup, karena adanya keterbatasan jumlah file yang dapat dibuka secara serentak.

- a. Perintah yang digunakan :

`fclose();`

- b. Bentuk deklarasi :

`int fclose(FILE *pf);`

- c. Bentuk deklarasi yang lain :

`int fcloseall(void);`

`fcloseall();`

prototype yang digunakan : `stdio.h`

D. Jenis Operasi File

1. **r** menyatakan file hanya akan dibaca, jika file belum ada maka tidak akan berhasil.
2. **w** menyatakan bahwa file baru diciptakan. Jika file tersebut sudah ada dalam disk, isinya yang lama akan terhapus.
3. **a** untuk membuka file yang sudah ada untuk ditambah dengan data, jika file belum ada akan dibuat yang baru.
4. **r+** sama dengan "r" tetapi selain file dapat dibaca, file juga dapat ditulisi.
5. **w+** sama dengan "w" tetapi selain file dapat ditulisi, file juga dapat dibaca.

6. **a+** sama dengan “w” tetapi selain file dapat ditulisi, file juga dapat dibaca.

E. File Teks

Karakteristik File Teks

1. Berisi data karakter ASCII
2. Tiap record boleh memiliki panjang yang bervariasi
3. Setiap record diakhiri tanda EOL (end of Line)
4. Hanya dapat diakses secara sequensial (berurutan).
5. Isi filenya dapat dilihat oleh perintah dos type atau editor text

Bentuk Umum

```
Var
    TipeFile : Text;
Begin
    Assign(TipeFile, 'Data.txt');
```

F. Membuat File Teks

1. Mendeklarasikan variabel file

```
Var NmVar:TEXT;
```

NmVar :Nama variabel file text
2. Menghubungkan variabel file dengan nama file

```
Assign (NmVar, nama file);
```

NmVar :Nama variabel file text
 Nama file :Nama file dalam bentuk string, format 8:3
 persamaan dos,ditulis dalam bentuk string.

G. Membuat file text aktif

```
Rewrite(NmVar);
```

Dengan;
 NmVar :Nama variabel file text yang sudah di- assign

H. Menulis ke dalam file text

```
Write / writeln (NmVar, data item1, data item 2, ...)
```

Dengan:
 NmVar : Nama variabel file text
 Data item : text / string yang akan dituliskan atau bisa juga berupa
 isisuat variabel

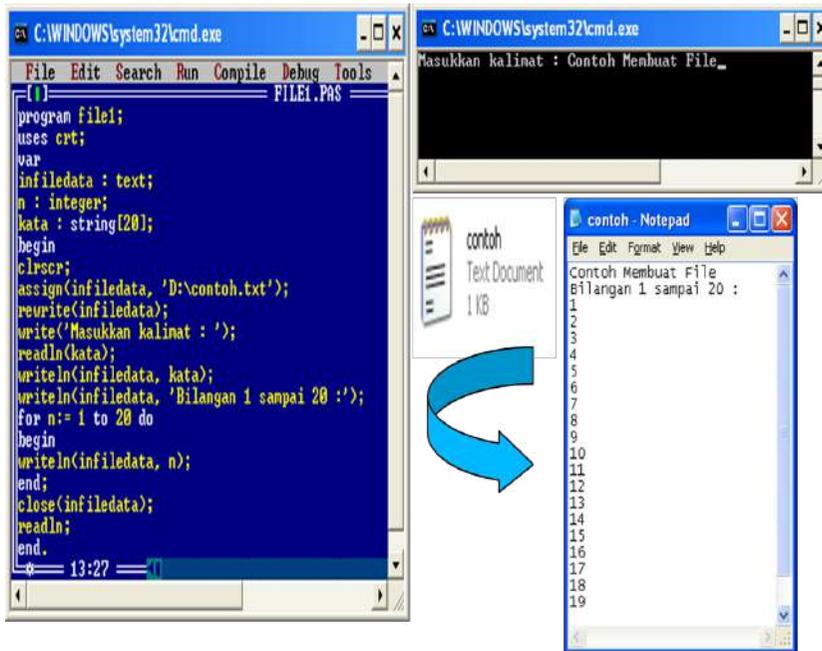
I. Menutup file

```
Close (NmVar);
```

Dengan:

NmVar : Nama variabel file text

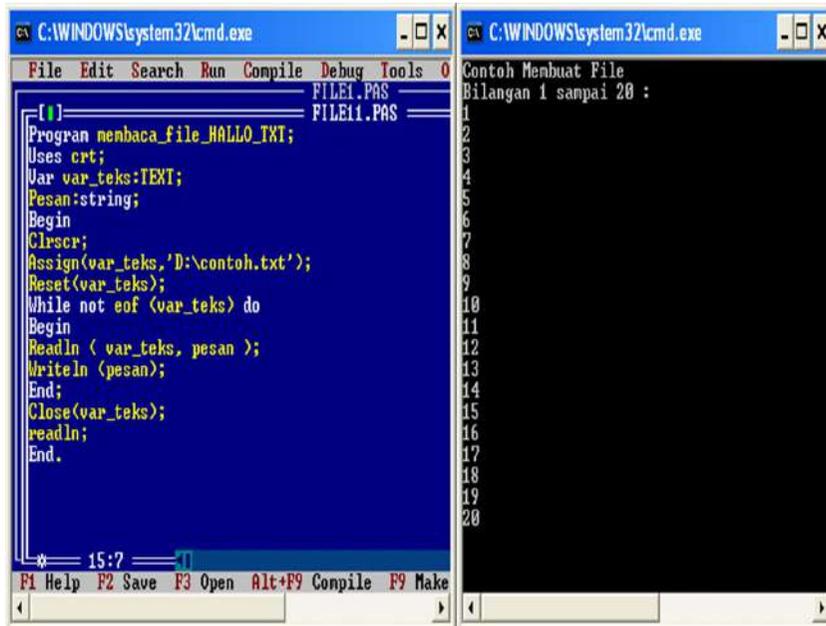
Contoh Program Membuat File Teks



Gambar 12.1 Program Penggunaan Variabel

1. Membaca File Teks
 - a. Mendeklarasikan variabel file
Var NmVar:TEXT;
 - b. Menghubungkan variabel file dengan nama file
Assign (NmVar, nama file)
 - c. Membaca isi file dan menampilkannya di layar
While not eof (NmVar) do
Begin
Read / readln (NmVar, data item 1, data item 2, ...);
Write / writeln (data item1, data item 2, ...);
End;
 - d. Menutup file
Close (NumVar);

Contoh Program Membaca File Teks



The image shows two windows from a Turbo Pascal IDE. The left window, titled 'C:\WINDOWS\system32\cmd.exe', contains the source code for a program named 'membaca_file_HALLO_TXT'. The code declares a text file variable, opens 'D:\contoh.txt', reads the file line by line, and prints each line to the screen. The right window, also titled 'C:\WINDOWS\system32\cmd.exe', shows the program's output: 'Contoh Membuat File' followed by 'Bilangan 1 sampai 20 :', and then a list of numbers from 1 to 20, one per line.

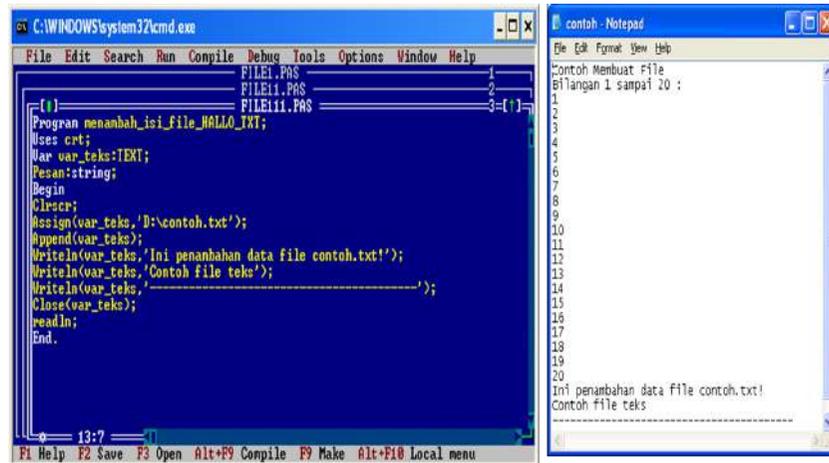
```
Program membaca_file_HALLO_TXT;
Uses crt;
Var var_teks:TEXT;
Pesan:string;
Begin
Clrscr;
Assign(var_teks,'D:\contoh.txt');
Reset(var_teks);
While not eof (var_teks) do
Begin
Readln ( var_teks, pesan );
Writeln (pesan);
End;
Close(var_teks);
readln;
End.
```

```
Contoh Membuat File
Bilangan 1 sampai 20 :
1
2
3
4
5
6
7
8
9
10
11
12
13
14
15
16
17
18
19
20
```

Gambar 12.2 Program Baca Teks File

2. Menambah Isi File Teks
 - a. Mendeklarasikan variabel file
Var NmVar:TEXT;
 - b. Menghubungkan variabel file dengan nama file
Assign (NmVar, nama file)
 - c. Menambah isi file
Append(NmVar)
 - d. Menampilkannya di layar
Write / writeln (NmVar, data item1, data item 2, ...)
 - e. Menutup file
Close (NumVar);

Contoh Program Menambah Isi File



Gambar 12.3 Program Isi File

J. File Bertipe

Karakteristik File Bertipe

1. Berisi data format biner, ukurannya lebih kecil dari file teks.
2. Tiap record memiliki tipe dan panjang yang sama. Bisa saja memiliki berbagai tipe asalkan dikelompokkan dalam RECORD.
3. Dapat diakses secara random, elemen-elemennya bisa dibaca secara acak yang seberapa saja.

Bentuk Umum

Var

FileData : file of byte;

Begin

Assign(FileData, 'Abc');

Rewrite(fileData);

K. Membuat File Bertipe

1. Mendeklarasikan variabel file

Var NmVar:FILE OF Type Variabel;

Dengan:

NmVar : Nama variabel file bertipe

Type variabel: Char, variabel tipe RECORD, variabel tipe array, real, variabel array tipe record. Untuk satu file satu tipe elemen.

Contoh:

Type DaftarBarang =Array [1..100] of integer;

DataKonsumen=RECORD

Nama string[15];

Alamat string[30];

Kode :1..3;

DataKinsumen=Array [1..100] of Datakonsumen

Var

FileBarang : File of DataBrang;

FileJumlah : File of integer;

FileData : File of DataKonsumen;

FileKode : File of Char;

2. Menghubungkan variabel file dengan nama file

Assign (NmVar, nama file);

Dengan:

NmVar : Nama variabel file bertipe

Nama file ; Nama file dlam bentuk string, format 8:3
persamaan dos,ditulis dalam bentuk string

3. Membuat /membuat file bertipe

Rewrite(NmVar);=> untuk membuat

Reset(NmVar); => untuk membuka

4. Menulis / membaca file bertipe

Write (NmVar, data item1, data item 2, ...) => untuk menulis

Read (NmVar, data item1, data item 2, ...) => untuk
membacadata item1, data Item 2 dan seterusnya, harus berupa
variabel, tidak bisa dituiskan secara langsung dalam bentuk
konstanta. Variabelnya harus sama dengan deklarasi tipe file-
nya.

Fungsi Writeln dan Readln tidak dapat digunakan pada file
bertipe.

5. Menutup file

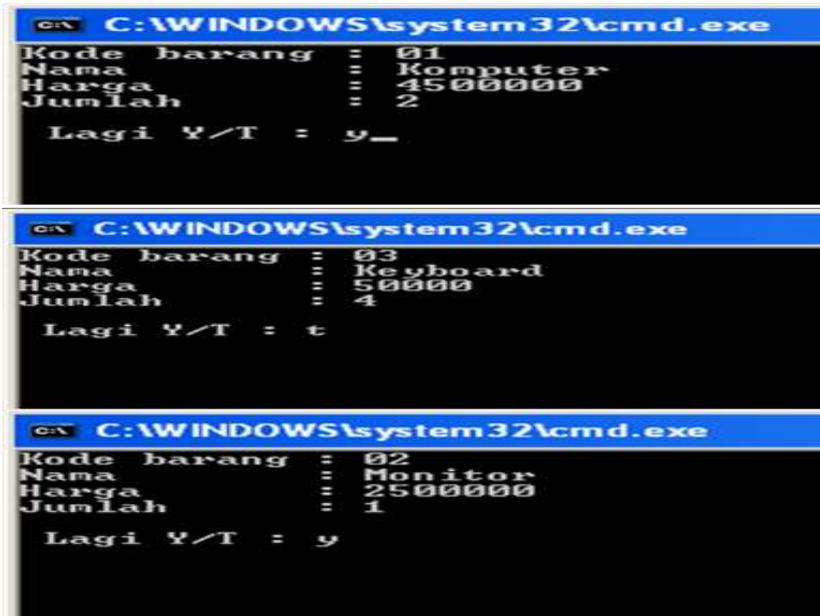
Close (NmVar);

Contoh Program File Bertipe

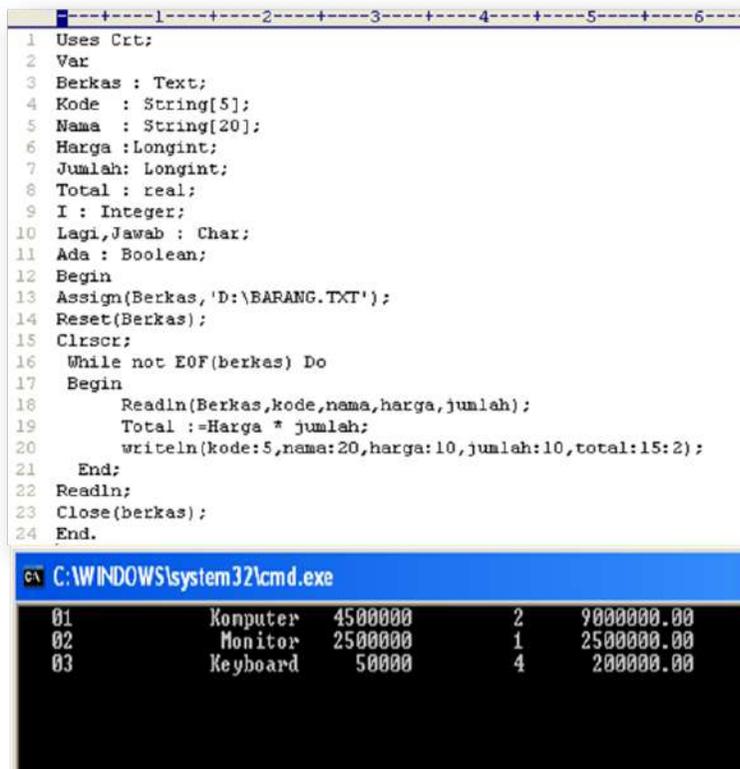
```
1 Uses Crt;
2 Var
3 Berkas : Text;
4 Kode : String[5];
5 Nama : String[20];
6 Harga : Longint;
7 Jumlah: Longint;
8 Lagi,Jawab : Char;
9 Ada : Boolean;
10 Begin
11 Assign(Berkas, 'D:\BARANG.TXT');
12 {$I-}
13 Reset(Berkas);
14 {$I+}
15 Append(Berkas);
16 Lagi := 'Y';
17 While upcase(lagi)='Y' do
18 begin
19 Clrscr;
20 Write('Kode barang : ');Readln(Kode);
21 Write('Nama : ');Readln(Nama);
22 Write('Harga : ');Readln(Harga);
23 Write('Jumlah : ');Readln(jumlah);
24 Writeln(Berkas,kode:5,nama:20,harga:10,jumlah:10);
25 Writeln;
26 Write(' Lagi Y/T : ');Readln(lagi);
27 End;
28 Close(berkas);
29 End.
```



Gambar 12.4 Program File Bertipe



Gambar 12.5 Hasil Program File Bertipe



Gambar 12.6 Program Tipe Berkas

L. File Tak Bertipe

Karakteristik File Tak Bertipe

1. File yang mengakses langsung ke media penyimpanan tanpa adanya pengenalan record dan sebagainya.
2. Digunakan untuk tugas-tugas yang berhubungan dengan file biner yang dapat diproses tanpa mengenal jenis recordnya.
3. File tak bertipe (*untyped file*) adalah channel I/O (*Input/output*) level rendah yang terutama digunakan untuk mengakses langsung suatu file di disk tidak peduli bagaimana tipe dan strukturnya.

Contoh : FileData : File;

1. Membuat File Tak Bertipe
 - a. Mendeklarasikan variabel file
Var NmVar:FILE;
 - b. Menghubungkan variabel file dengan nama file
Assign (NmVar, nama file);
NmVar :Nama variabel file BERTIPE
Nama file : Nama file dalam bentuk string, format 8:3 persamaan dos,ditulis dalam bentuk string.
 - c. Membuat file text aktif
Rewrite(NmVar[,brec]); => untuk membuat
Reset(NmVar[,brec]); => untuk membuka
Dengan;
Brec : Menunjukkan besar file dalam byte, opsional, boleh ditulis, boleh tidak, dan besarnya kita tentukan sendiri. Defaultnya 128 Byte.
 - d. Menulis / membaca file bertipe
Blockwrite (NmVar, Buffer, jumlah[jumtulis]) => untuk menulis
Read (NmVar, data item1, data item 2, ...) => untuk membaca

2. Contoh Menyalin File Tak Bertipe

```
1 Program menyalin_file
2 Uses crt;
3 Var
4 Fileasal, filetujuan :file;
5 Besar :real;
6 Buf :array [1..10240] of char;
7 Hbaca, Htulis :word;
8 nfile1,nfile2 :string;
9 Begin
10 besar:=0;clrscr;
11 write ('Input nama file asal :');readln(nfile1);
12 write ('Input nama file tujuan :');readln(nfile2);
13 {$I-}
14 Assign(fileasal,nfile1);
15 Assign(filetujuan,nfile2);
16 Reset(fileasal,1);
17 If IOResult <>0 then
18 Begin
19 Writeln('File asal tidak ada !');
20 Halt(0);
21 End;
22 Rewrite(filetujuan,1);
23 If IOResult <>0 then
24 Begin
25 Writeln('File tujuan tidak bisa dibuat !');
26 Halt(0);
27 End;{$I+}
28 Repeat
29 Blockread(fileasal,buf,10240,hbaca);
30 Besar:=besar + hbaca;
31 Blockwrite(filetujuan,buf,hbaca,htulis);
32 If hbaca <> htulis then
33 Begin
34 Writeln('Tujuan tidak bisa menampung besar file');
35 Halt(0);
36 End;
37 Until hbaca <>10240;
38 Close (fileasal);
39 Close (filetujuan);
40 Writeln ('Besar file yang dipindahkan :',besar:10:0);
41 End.
```



```
C:\WINDOWS\system32\cmd.exe
Input nama file asal :CONTOH.TXT
Input nama file tujuan :CONTOH2.TXT
Besar file yang dipindahkan : 215
```



TUJUAN INSTRUSIONAL

1. Membuat deklarasi tipe data record.
2. Membuat contoh program sederhana dengan menggunakan tipe data record

A. Pengertian

Tipe data record adalah tipe data khusus yang komponennya terdiri dari berbagai jenis tipe data lain. Sebuah record berisi beberapa *variabel lain* yang ‘dipaketkan’. Konsep struktur data seperti ini sedikit mirip dengan konsep object dalam bahasa pemrograman modern (walaupun di dalam pascal juga terdapat konsep tentang object).

Record adalah kumpulan elemen –elemen data yang digabungkan menjadi satu kesatuan. Masing2 elemen data disebut field. Field data tersebut dapat memiliki tipe data yg sama atau berbeda. Field – field tersebut digabungkan menjadi satu record dengan tujuan untuk memudahkan. Record juga mirip dengan array, dimana kita bisa membuat sebuah variabel yang berisi berbagai element. Perbedaannya, *record* bisa menampung berbagai jenis tipe data, tidak hanya 1 tipe data seperti *array*.

B. Deklarasi Record :

Bentuk Umum:

type

```
<NamaRecord> = record  
    <DataField-1>:<type1>;  
    <DataField-2>:<type2>;  
    ...  
    <DataField-N>:<typeN>;
```

var

```
<Namavariabel>:<NamaRecord>;
```

Deklarasi record pada umumnya diawali dengan kata baku type, namun anda juga dapat mendeklarasikan record langsung dengan menggunakan kata baku var seperti berikut :

```
var
    <NamaRecord>: record
        <DataField-1>:<type1>;
        <DataField-2>:<type2>;
        ...
        <DataField-N>:<typeN>;
    end;
```

Contoh :

```
type
    Mahasiswa = record
        Nim : string[10];
        Nama : string[20];
        Alamat : string[30];
        IPK : real;
var
    Mhs : Mahasiswa;
```

Pendeklarasian record selalu diawali oleh nama record, tanda sama dengan (=) dan kata baku record serta diakhiri oleh kata baku end. Perhatikan bahwa untuk membuat *record*, diawali dengan **nama_record**. Ini adalah variabel yang akan menampung seluruh isi record. Setelah itu, pembuatan 'isi' *record* berada di antara perintah **record** dan **end**; Disinilah seluruh variabel yang menjadi 'isi' record di defenisikan. Contoh sebuah *record* '**siswa**' yang terdiri dari **nama**, **umur**, **sekolah**, dan **kota**. Berikut cara penulisannya : Field – field dari record tersebut diletakkan diantara kata baku record dan end. Di dalam suatu record jika terdapat field-field yang bertipe sama dapat dideklarasikan bersamaan dengan dipisahkan oleh tanda koma (,) sehingga anda tidak perlu menuliskan tipe datanya berulang ulang.

C. Pemakaian Record

Untuk menggunakan Record tersebut, maka harus ditulis nama record beserta dengan fieldnya yang dipisahkan dengan tanda titik (.). Misal akan menulis NIM seorang mahasiswa ke layar maka penulisan yang benar adalah :

yang dibuat sudah terlalu panjang, maka akan sulit untuk membaca dan mengerti jalannya program tersebut. Sehingga ada baiknya memecahkannya menjadi beberapa bagian (modul) yang tentunya akan memudahkan dalam mencari kesalahan program dan memperbaikinya serta membuat dokumentasinya. Dalam Pascal disediakan dua pilihan :

1. Procedure
2. Function

D. Procedure

Procedure adalah Berguna untuk mengumpulkan statement statement yang dapat dijalankan menjadi satu dalam suatu blok dan untuk menjalankannya kembali hanya dengan menuliskan nama procedure yang menampungnya. Selain itu juga banyak dipakai untuk menampung baris-baris perintah yg sering dipakai dalam sebuah program. Untuk mendeklarasikan procedure dapat dilakukan dengan dua cara, yaitu :

1. Header procedure tanpa parameter
2. Header procedure dengan parameter

1. Header Procedure Tanpa Parameter

Bentuk Umum :

```
Procedure<NamaProcedure>;
```

Contoh :

```
Procedure BuatKotak;  
Procedure Input;  
Procedure Output;
```

Penulisan header procedure tanpa parameter diawali dengan kata baku procedure dan diikuti dengan nama procedure serta diakhiri dengan tanda titik koma (;)

Berikut digambarkan struktur blok program beserta procedure tanpa parameter

2. Header Procedure dengan Parameter

Bentuk Umum :

```
Procedure<NamaProcedure>(<Daftar Parameter>;
```

Contoh :

```
Procedure Hitung (a,b:byte; c:real);
```

Procedure Lingkaran(x,y, jari :integer);

Penulisan header procedure dengan parameter hampir sama dengan procedure tanpa parameter yaitu diawali dengan kata baku procedure, lalu nama procedure dan diikuti dengan parameter (yang berada dalam kurung) yang masing-masing dipisahkan dengan koma beserta dengan tipenya yang dipisahkan dengan titik dua (:) serta diakhiri dengan tanda titik koma (;).

Contoh:

```
uses wincrt;
var
p,q,x,y : byte;
procedure tambahkali(a,b :byte);
begin
p:= a+b;
q:= a*b;
writeln('x+y =' ,p);
writeln('x*y =' ,q);
end;
begin
write('x =');readln(x);
write('y =');readln(y);
tambahkali(x,y);
end.
```

E. Function

Function tidak hanya dapat dipakai untuk mengelompokkan baris-baris perintah seperti halnya procedure, tetapi function itu sendiri dapat menampung nilai yang disimpan pada nama function. Penulisan header function dapat dilakukan dengan dua cara, yaitu :

1. Header function tanpa parameter
2. Header function dengan parameter

1. Header Function Tanpa Parameter

Bentuk Umum

```
Function<NamaFunction>:TipeData;
```

Contoh:

```
Function Hitung : integer;
```

```
Function : string;
```

Function : boolean;

Header function selalu diawali dengan kata baku Function dan diikuti dengan nama function serta tipe datanya yang dipisahkan dengan tanda titik dua (:).

Contoh:

```
uses wincrt;
procedure hitung;
var
  x,y : byte;
  function tambah:byte;
  begin
    tambah:=x+y;
  end;
function kali:byte;
begin
  kali :=x*y;
end;
begin
write('x =');readln(x);
write('y =');readln(y);
writeln('x+y=',tambah);
writeln('x*y=',kali);
end.
```

2. Header Function dengan Parameter

Bentuk Umum:

Function<NamaFunction>(DaftarParameter>):TipeData:

Contoh

Function Hitung(a,b:byte) :integer;

Function CheckPosisi(x,y:integer):boolean;

Penulisannya tidak jauh berbeda dengan function tanpa parameter, hanya saja perlu ditambahkan parameter – parameter yang ditulis di dalam kurung ().

**BAB
XIV**

**FUGSI
FUNCTION PROCEDURE**

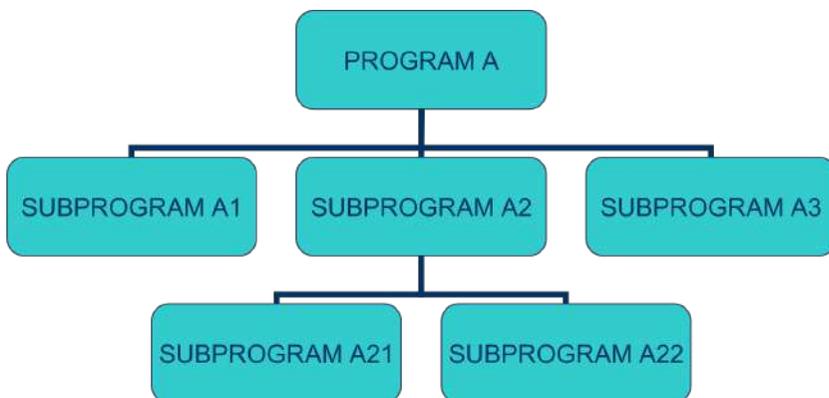
TUJUAN INSTRUSIONAL

1. Menjelaskan pengertian tentang suatu file.
2. Menyebutkan procedure dan fungsi standart untuk semua tipe file.
3. Membuat deklarasi untuk suatu file.
4. Mengerti cara membuat file, menambah data dan menampilkan data pada file

A. Pemrograman Modular

Pemrograman modular adalah suatu teknik pemrograman di mana program yang biasanya cukup besar dibagi-bagi menjadi beberapa bagian program yang lebih kecil. Dengan cara ini, program akan lebih mudah dimengerti, dan jika ada kesalahan di dalam proses pengolahan data akan lebih mudah untuk mencari kesalahannya. Dalam beberapa bahasa pemrograman disebut : sub-rutin(sub program), modul, prosedur, atau fungsi. Subprogram adalah bagian dari program yang dirancang untuk melaksanakan suatu tugas tertentu

B. Struktur



Gambar 14.1 Struktur Modular

C. Keuntungan Pemrograman Modular

1. Program lebih pendek
2. Mudah menulis (banyak local mmer)
3. Mudah dibaca dan dimengerti (bandingkan dg nonmodular dg banyak instruksi)
4. Mudah didokumentasi
5. Mengurangi kesalahan dan mudah mencari kesalahan(*debug*) program
6. Kesalahan yang terjadi bersifat “local”

D. Dua Bentuk Pemrograman Modular : Prosedur Dan Fungsi

1. Struktur setiap subprogram tersebut pada hakekatnya sama , yaitu :
 - a. Nama modul (subprogram)
 - b. Bagian deklarasi
 - c. Algoritma (intruksi yg akan dilaksanakan)
2. Perbedaan penggunaannya dalam bahasa pemrograman Pascal :
 - a. Prosedur merupakan modul(subprogram) yg melakukan aktifitas tertentu tanpa adanya pengembalian nilai
 - b. Fungsi terdapat pengembalian nilai

E. Prosedur

Prosedur dalam Pascal dapat berbentuk :

1. Prosedur yang didefinisikan dan dibuat sendiri oleh progammer
2. Prosedur yang telah disediakan oleh Pascal (Standard Procedure)

Prosedur yang dibuat sendiri oleh programmer harus dideklarasikan terlebih dahulu di deklarasi prosedur.

a. Deklarasi Prosedur

Prosedur merupakan bagian yang terpisah dari program dan dapat diaktifkan dimanapun di dalam program. Kata Kunci “Procedure” digunakan sebagai judul dari bagian deklarasi prosedur, diikuti oleh identifier yang merupakan nama dari prosedurnya dan secara optional dapat diikuti oleh kumpulan parameter yang diakhiri dengan titik koma.

b. Bentuk Umum Procedure

Procedure nama-prosedur [(deretan-parameter)] ;

[bagian deklarasi konstanta , tipe , variabel , dan prosedur / fungsi]

begin

[statemen–statemen dalam badan pros edur]

end ;

c. Penerapan Procedure

```
Program judul_program;
var
{bagian deklarasi prosedur, sifatnya global}

Procedure Nama_Prosedur; ←
var
{bagian deklarasi prosedur, sifatnya lokal}
begin
  statement-1;
  statement-2;
  -----↓
  Statement-n;
end;

{program utama}
begin
  Nama_Prosedur; -----
end.
```

Gambar 14.2 Penerapan Procedure

d. Contoh Program

```
PROC_1.PAS
Program Contoh_Procedure;
uses crt;
Procedure Coba;
begin
  Writeln('Saya Ada dalam Procedure Coba');
  readln;
end;

<Program Utama>
Begin
  clrscr;
  Writeln('Aku diluar procedure, di Program utama');
  coba;
end.
```

C:\ TPX

Aku diluar procedure, di Program utama
Saya Ada dalam Procedure Coba

Gambar 14.3 Program Dengan Procedure

```

[ ] PROC_3.
uses crt;
var
  a,t : integer;
  ls : real;

Procedure segitiga;
begin
  ls := a * t * 0.5;
  Write<'Jadi luas segitiga :',ls:5:2>;
end;

begin
  clrscr;
  write<'Nilai Alas :>;readln(a);
  write<'Nilai Tinggi :>;readln(t);
  segitiga;
  readln;
end.

```

```

TPX
Nilai Alas :80
Nilai Tinggi :90
Jadi luas segitiga :3600.00

```

Gambar 14.4 Program Procedure 2

e. Parameter dalam Prosedur

Nilai di dalam suatu modul Program Pascal sifatnya LOKAL (hanya dapat digunakan pada modul atau unit program yg bersangkutan saja, tidak dapat digunakan pada modul/unit program yg lain)

Untuk bersifat GLOBAL harus dideklarasikan di luar modul atau unit program.

f. Perbedaan Lokal dan Global

Contoh : Parameter Lokal	Contoh : Paramater Global
<pre> Program Coba_Procedure_Hitung; Procedure Hitung ; Var X, Y : real ; Begin Write (' Nilai X ? '); readln (X); Y := X * X ; Writeln (' Nilai Y ? ', Y :1: 0); End ; BEGIN Hitung ; END. </pre>	<pre> Program Coba_Procedure_Hitung Var X, Y : Byte ; Procedure Tambah ; Begin Write (' Input Nilai : '); readln (X); Y := A + A ; End ; BEGIN Tambah ; Writeln (' Nilai Y = ', Y); Readln; END. </pre>

g. Pengiriman Parameter

- By Value

- Prosedur dimulai dengan deklarasi prosedur dengan judul prosedur :

```
Procedure nama_prosedur (variabel_formal : tipe_data);
```

- Variabel Lokal yg hanya digunakan di prosedur tersebut dan tidak termasuk sebagai parameter formal harus didefinisikan sendiri didalam prosedur tersebut.

```
Var
```

```
variabel_lokal : tipe data;
```

- Hubungan antara variabel formal di prosedur dengan variabel nyata di program : satu arah yaitu nilai dari variabel nyata dikirim ke variabel formal

- By Reference

- Perubahan pada nilai variabel formal di prosedur akan mempengaruhi nilai variabel nyata.

```
Procedure nama_prosedur( VAR variabel_formal : tipe_data)
```

- Hubungan antara variabel formal di prosedur dengan variabel nyata di program dua arah/bolak-balik.

- Campur

- Pengiriman parameter dapat dicampur sebagian secara nilai dan sebagian secara acuan dalam suatu prosedur.

- Yg hanya dibutuhkan pada prosedur saja dapat dikirim by value, yg ingin dikirimkan balik dapat dilakukan by reference.

```
Procedure nama_prosedur(variabel_formal : tipe_data; VAR variabel_formal : tipe_data)
```

h. Tipe Data Variabel Formal

Tipe Sederhana seperti Integer, Byte, Char, Real, dan Boolean dapat digunakan sebagai tipe dari variabel formal. Untuk String dan Array harus dideklarasikan terlebih dahulu di luar prosedur.

- i. **Prosedur Memanggil Prosedur**
Di dalam suatu prosedur yang dibuat sendiri dapat memanggil prosedur lainnya.
 - j. **Prosedur Tersarang**
Disebut *Nested Procedure*, Prosedur yang berada di dalam prosedur yang lainnya.
 - k. **Prosedur Memanggil Dirinya Sendiri**
Suatu prosedur yang memanggil atau menggunakan prosedur itu juga (*recursion*)
3. **Prosedur Standar**
Prosedur yang disediakan oleh Turbo Pascal :
- a. **Prosedur standar EXIT**
Digunakan untuk keluar dari suatu blok.
 - b. **Prosedur standar HALT**
Digunakan untuk menghentikan proses program baik di program bagian maupun di program utama.
 - c. **Prosedur standar MOVE**
Bentuk umum : **MOVE (Var source,dest; count : word);**
Digunakan untuk menyalin suatu blok sebanyak count byte memori dari blok dimulai byte pertama source dan disalinkan ke byte pertama dest.
 - d. **Prosedur standar FILLCHAR**
Digunakan untuk mengisi sejumlah byte nilai ke dalam suatu variabel, sebagai berikut **FillChar(x;count :word;ch);** X adalah variabel yang dapat bertipe apapun yang akan diisi dengan nilai tipe ordinal Ch sebanyak count byte.

F. Function(Fungsi)

Fungsi secara garis besar sama dengan prosedur baik parameter maupun pemanggilan parameternya hanya yang membedakannya adalah nama fungsi harus dideklarasikan dengan tipe datanya , Sehingga dikatakan function dapat mengembalikan nilai.

Blok fungsi hampir sama dengan blok prosedur, hanya fungsi harus dideklarasikan dengan tipenya atau jenis hasilnya. Tipe deklarasi ini menunjukkan tipe hasil dari fungsi.

Pada bahasa Pascal dikenal beberapa fungsi, misalkan : abs, pred, sqrt, sqr, succ dan sebagainya.

1. Bentuk Umum

```
FUNCTION identifier(daftar parameter) : type;
```

2. Parameter Nilai dalam Fungsi

Parameter dalam function dapat dikirimkan secara nilai atau secara acuan.

Penulisan judul function yang menggunakan parameter secara Nilai adalah :

```
Function besar(a,b : real) : real;
```

3. Contoh Program

```
program penggunaan_parameter_nilai;
uses crt;
function besar(a,b :real) : real;
begin
    if a>b then
        besar:=a
    else
        besar:=b;
    end;
{modul utama}
var
    nil1,nil2 : real;
begin
    write('bilangan 1=');readln(nil1);
    write('bilangan 2=');readln(nil2);
    writeln('bilangan terbesar =',besar(nil1,nil2):6:2);
    readln;
end.
```

G. Fungsi Standar

1. Fungsi standar aritmatika

- a. Fungsi standar ABS Bentuk umum : $ABS(x)$;

Digunakan untuk memutlakkan suatu nilai yang ditunjukkan oleh argumen x .

Contoh :

Begin

$X := -2.3$;

Write('Nilai X = ', X, ' Nilai mutlaknya = ', Abs(X):3:1);

End.

- b. Fungsi standar EXP

Bentuk Umum : $EXP(x \odot : \text{real})$;

Digunakan untuk menghitung nilai pangkat dari bilangan e yaitu sebesar e^x . Hasilnya berupa nilai real.

- c. Fungsi standar LN

Bentuk umum : $LN(x) : \text{real}$;

Digunakan untuk menghitung nilai logaritma alam (natural logarithm) dari nilai x . Hasilnya berupa nilai real.

- d. Fungsi standar INT

Bentuk umum : $INT(x : \text{real}) : \text{real}$;

Digunakan untuk menghasilkan nilai integer dari x . Hasil dari fungsi adalah tipe real dengan nilai yang berupa pembulatan ke bawah (nilai pecahan dibuang) dari nilai x .

Contoh :

Begin

$X := 9.99$;

Write('Nilai yang akan dibulatkan = ', X);

Writeln('Nilai pembulatannya = ', Int(X):3:2);

End.

Hasil :

Nilai yang akan dibulatkan = 9.99

Nilai pembulatannya = 9.00

- e. Fungsi standar FRAC

Bentuk umum : $FRAC(x \odot : \text{real})$;

Digunakan untuk mendapatkan nilai pecahan dari argumen x. Argumen x dapat bernilai real maupun integer dan hasil dari fungsi adalah real.

Contoh :

Begin

X:=9.99;

Write('Nilai X = ',X,' Nilai pecahannya = ',Frac(X):4:3);

End.

Hasilnya : Nilai X = 9.99 Nilai pecahannya = 0.990

f. Fungsi standar SQR

Bentuk umum : SQR(x);

Digunakan untuk menghitung nilai pangkat kuadrat dari argumen x.

Contoh :

Begin

X:=2;

Write('Nilai X = ',X,' Nilai kuadratnya = ',sqr(x));

End.

Hasilnya : Nilai X = 2 Nilai kuadratnya = 4

g. Fungsi standar SQRT

Bentuk umum : SQRT(x) : real;

Digunakan untuk menghitung nilai akar dari argumen x, hasilnya berupa real.

h. Fungsi standar PI, SIN, COS, ARCTAN

2. Fungsi Standar Transfer Digunakan untuk merubah suatu nilai ke bentuk nilai lain.

a. Fungsi standar CHR

Bentuk umum : CHR(x:byte):char;

Digunakan untuk merubah nilai dari byte x ke bentuk karakter yang sesuai dengan kode ASCII.

Contoh :

X:= 66;

Write('Nilai X = ',x,' Nilai karakternya = ',CHR(X));

Hasilnya : Nilai X = 66 Nilai karakternya = B

b. Fungsi standar ORD

Bentuk umum : ORD(x):longint;

Digunakan untuk merubah nilai x ke bentuk nilai longint yang sesuai dengan kode ASCII, merupakan kebalikan dari fungsi CHR.

c. Fungsi standar ROUND

Bentuk umum : ROUND(x:real):longint;

Digunakan untuk membulatkan nilai dari real x ke nilai longint yang terdekat. Bila nilai pecahan sama dengan atau lebih besar dari 0.5 akan dibulatkan ke atas, sedang kalau lebih kecil dari 0.5 akan dibulatkan ke bawah.

Contoh :

```
Write('10 dibagi 3 hasil pembulatan terdekat  
'Round(10/3));
```

```
Writeln('20 dibagi 3 hasil pembulatan terdekat adalah  
'Round(20/3));
```

Hasilnya :

10 dibagi 3 hasil pembulatan terdekat adalah 3

20 dibagi 3 hasil pembulatan terdekat adalah 7

d. Fungsi standar TRUNC

Bentuk umum : TRUNC(x:real):longint;

Digunakan untuk membulatkan nilai dari real x ke nilai longint terkecil. Atau dengan kata lain membulatkan ke bawah.

Contoh :

```
Write('10 dibagi 3 hasil pembulatan terdekat  
'Trunc(10/3));
```

```
Writeln('20 dibagi 3 hasil pembulatan terdekat adalah  
'Trunc(20/3));
```

Hasilnya :

10 dibagi 3 hasil pembulatan terdekat adalah 3

20 dibagi 3 hasil pembulatan terdekat adalah 6

3. Fungsi Standar Lainnya Fungsi standar yang tidak termasuk dalam kelompok pembagian di atas :

a. Fungsi standar Hi, Lo, Swap

b. Fungsi standar Random

Bentuk umum : Random [(range :word)];

Digunakan untuk menghasilkan angka random berkisar dari nilai lebih besar atau sama dengan nol dan lebih kecil dari satu. Bila range tidak disebutkan, hasil dari fungsi ini adalah real, bila range disebutkan, hasilnya adalah word.

c. Fungsi standar SizeOf

Bentuk umum : SizeOf(x):word;

Digunakan untuk menunjukkan besarnya byte yang digunakan oleh suatu variabel x, hasilnya berupa nilai word.

d. Fungsi standar UPCASE

Bentuk umum : UpCase(Ch:char):char;

Digunakan untuk merubah argumen suatu karakter yang ditunjukkan oleh Ch menjadi bentuk karakter huruf besar (upper case).

4. Contoh

a. Akan dibuat suatu fungsi dengan nama MAX yang dapat menentukan integer terbesar di antara dua integer.

Function MAX (x,y : integer) : integer;

Begin

If x < y then MAX := y ;

Else MAX := x;

End;

Selanjutnya kita dapat menggunakan fungsi di atas dalam suatu program, misalnya dengan menyatakan sebagai berikut :

P := MAX(a,b);

*Z := MAX(a+b,a*b);*

Q := MAX(MAX(a,b),c);

.....

dsb.

b. Function LOG (x : real) : real;

Begin

LOG := ln (x) / ln (10.0);

End;

c. Function POWER (x,y : real) : real;

Begin

$POWER := \exp (y * \ln (X))$

End;

$a^b = POWER (a,b)$

$b^a = POWER (b,a)$

$(p + q)^{r/s} = POWER (p + q, r/s)$

.....

dll

H. Perbedaan fungsi dengan prosedur

Pada fungsi, nilai yang dikirimkan balik terdapat pada nama fungsinya (kalau pada prosedur pada parameter yang dikirimkan secara acuan). Pada contoh, nama fungsi tersebut adalah Hitung dan nilai yang dikirim balik berada pada nama fungsi tersebut. Sehingga nama fungsi ini harus digunakan untuk menampung hasil yang akan dikirimkan dari fungsi, sebagai berikut : Hitung := A + B;

Nama fungsi yang berisi nilai yang akan dikirimkan Karena nilai balik berada di nama fungsi tersebut, maka fungsi tersebut dapat langsung digunakan untuk dicetak hasilnya, sebagai berikut :

Writeln(X,' + ',Y,' = ',Hitung(X,Y));

_____ Nama fungsi yang langsung digunakan untuk ditampilkan hasilnya.

Atau nilai fungsi tersebut dapat juga langsung dipindahkan ke pengenal variabel yang lainnya, sebagai berikut :

Hasil := Hitung(X,Y);

Writeln(X,' + ',Y,' + ',Hasil);

Sedang pada prosedur, nama prosedur tersebut tidak dapat digunakan langsung, yang dapat langsung digunakan adalah parameternya yang mengandung nilai balik.

I. Rekursif

Suatu fungsi atau prosedur dalam bahasa Pascal dapat bersifat rekursif. Artinya, fungsi atau prosedur tersebut dapat memanggil dirinya sendiri. Berikut ini sebuah contoh fungsi dan prosedur yang rekursif.

1. function faktorial (nilai : integer) : integer;

begin

if nilai <= 0 *then* faktorial := 1;

else faktorial := nilai * faktorial (nilai-1)

end;

Var

N : integer;

Begin

Write('Berapa faktorial ? ');

Readln(*N*);

Writeln(*N*, ' faktorial = 'faktorial(*N*):9:0);

End.

$$\begin{aligned} \text{—————} \quad & \text{faktorial (4) = 4 * faktorial (3)} \\ & \quad \quad \quad 3 * \text{faktorial (2)} \\ & \quad \quad \quad \quad 2 * \text{faktorial (1)} \\ & \quad \quad \quad \quad \quad 1 * \text{faktorial (0)} \\ & \quad \quad \quad \quad \quad \quad 1 \\ & = 4 * 3 * 2 * 1 * 1 \\ & = 24 \end{aligned}$$

2. Bilangan Fibonanci:

F (0) = 0

F (1) = 1

F (n) = F (n-1) + F (n-2); untuk n >1

Function fibonacci (*n* : integer) : integer;

Begin

If n = 0 *then* fibonacci := 0

Else

If n := 1 *then* fibonacci := 1

Else fibonacci := fibonacci (n-1) + fibonacci (n-2);

End;

3. Procedure reverse (num : integer);

Begin

If num < 10 *then* write(num)

Else begin

Write(num mod 10);

Reverse(num div 10);

End;

End;

Latihan;

- a) Buatlah Procedure yang lebih dari satu procedure tiga atau empat procedure dalam program utama
- b) Buatlah Procedure untuk menghitung Luas dan keliling Segi Empat sendiri-sendiri
- c) Buatlah Program untuk menghitung Luas Segitiga yang input, proses dan laporan dibuatkan procedure (masuk, proses, laporan)

Daftar Pustaka

- Ahmad Taufik Nasution, 2006, "Filsafat Ilmu Hakikat Mencari Pengetahuan", Yogyakarta : CV. Budi Utama
- Gery B Shelly, 2012, "System Analysis and Design", Harry J. Rossenblatt, 9th Edition
- Jan Hendrik Rapar, 1996, "Pengantar Asas-asas Penalaran Sistematis", Yogyakarta : Kanisius Media
- Jogiyanto, 2016, "Analisis dan Disain Sistem Informasi", Yogyakarta : Andi Offset
- Kendall and Kendal, 2010, "System Analysis and Design", Prentice Hall, 8th Edition,
- Kengage Learning, 2010, "Systems Analysis and Design in Changing World", 5th Edition
- Soekadijo, 2008, "Majalah Ilmu Ilmu Sosial Indonesia", Jakarta : LIPI Press, Jakarta
- Sugiyono, 2012, "Penelitian Kualitatif, Kuantatif, Mix dan R&D", Bandung : Alfabeta
- Wiliam Aiston, 2008, "Majalah Ilmu Ilmu Sosial Indonesia", Jakarta : LIPI Press

LOGIKA ALGORITMA

Algoritma sangat lekat dengan kata logika, yaitu kemampuan seorang manusia untuk berfikir dengan akal tentang suatu masalah dan menghasilkan sebuah kebenaran, dapat dibuktikan dan masuk akal. Dalam menyelesaikan suatu masalah logika sangat diperlukan. Logika identik dengan masuk akal dan penalaran. Penalaran adalah salah satu bentuk pemikiran. Definisi logika sangat sederhana yaitu cara berfikir untuk tujuan tertentu namun menurut aturan yang berlaku.

Menurut Kamus Besar Bahasa Indonesia: algoritma adalah urutan logis pengambilan putusan untuk pemecahan masalah. Algoritma dibutuhkan untuk memerintah komputer mengambil langkah-langkah tertentu dalam menyelesaikan masalah

Distributor Tunggal:
UNIVERSITAS STEKOM
Jln Majapahit No 605 Semarang
Tlpn. (024) 6723456
Fax . 024-6710144
Email: info@stekom.ac.id

ISBN 978-623-6141-42-7 (PDF)

