

---

---

Seri Buku Persiapan Seleksi Tim Olimpiade Komputer Indonesia

Buku Untuk Siswa  
**Referensi**  
**Pemrograman Bahasa Pascal**  
Menggunakan Free Pascal Versi 1.0.10

Disusun Oleh :  
Tim Pembina TOKI

Judul Buku	: Referensi Pemrograman Bahasa Pascal (Menggunakan Free Pascal versi 1.0.10)
Penyusun	: Tim Pembina TOKI
Kontributor	: Windra Swastika. Fauzan Joko
Naskah Sumber	: Online Help Free Pascal. <a href="http://www.freepascal.org/">http://www.freepascal.org/</a>
Penyunting	: TOKI
Disain Cover	: TOKI
Diterbitkan oleh	: Bagian Proyek Pengembangan Wawasan Keilmuan Direktorat Pendidikan Menengah Umum, Direktorat Jenderal Pendidikan Dasar dan Menengah, Departemen Pendidikan Nasional RI
Cetakan Pertama	: 2004

### **Seri Buku Periapan Seleksi Olimpiade Komputer Indonesia**

Buku 1 Untuk Siswa	: Referensi Pemrograman Bahasa Pascal ( <i>menggunakan free pascal ver. 1.0.10</i> )
Buku 2 Untuk Siswa dan Guru	: Konsep Dasar Pemrograman ( <i>dilengkapi dengan contoh soal, pembahasan dan solusi</i> )
Buku 3 Untuk Guru	: Aspek Pedagogi Pengajaran Pemrograman Pertama ( <i>Menggunakan Bahasa Pascal</i> )

#### *Copyleft :*

- ✍ *Seluruh isi dalam buku ini diijinkan untuk diperbanyak dan disebarluaskan sejauh untuk kepentingan pendidikan dan pengajaran, dengan tetap mencantumkan sumbernya.*
- ✍ *Buku ini juga dapat didownload dari situs web TOKI di [www2.toki.or.id](http://www2.toki.or.id) dan situs-situs pendukung lainnya*



## Kata Pengantar

Sejak keikutsertaan Indonesia dalam ajang International Olympiad in Informatics (IOI) pada tahun 1997, prestasi siswa-siswa dalam ajang tersebut cukup membanggakan. Hingga tahun 2004 ini (keikutsertaan yang ke 9, dari IOI yang ke 16) secara total Tim Olimpiade Komputer Indonesia telah mengumpulkan 1 Medali Emas, 6 Medali Perak dan 7 Medali Perunggu.

Sejauh ini Tim Olimpiade Komputer Indonesia yang dikirim ke ajang IOI masih didominasi oleh siswa-siswa dari wilayah-wilayah tertentu, situasi menunjukkan bahwa masih terdapat ketimpangan pengetahuan dan pembinaan di bidang Komputer/Informatika terhadap siswa-siswa di daerah lain. Hal ini dapat terjadi salah satunya adalah karena keterbatasan ketersediaan materi pelajaran komputer, khususnya yang mengarah kepada materi-materi yang dilombakan dalam ajang Olimpiade Komputer Indonesia.

Buku ini diterbitkan dalam tiga seri untuk siswa maupun untuk guru dimaksudkan agar dapat menjadi bahan pelajaran bagi siswa dan panduan pengajaran bagi guru yang memadai untuk mempersiapkan siswa menghadapi rangkaian seleksi Olimpiade Komputer Indonesia. Dengan harapan bahwa kesempatan dan peluang bagi siswa-siswa di seluruh Indonesia menjadi lebih terbuka.

Terima Kasih kepada semua pihak yang telah memberikan kontribusinya sehingga penerbitan buku ini dapat terwujud. Besar harapan kami buku ini dapat bermanfaat bagi semua pihak. Buku ini masih jauh dari sempurna, untuk itu kritik dan saran yang membangun sangat kami harapkan.

Jakarta, Nopember 2004

Pembina Tim Olimpiade Komputer Indonesia

Web site : [www2.toki.or.id](http://www2.toki.or.id) - Mailing list : [tokinet@yahoogroups.com](mailto:tokinet@yahoogroups.com)

## Daftar Isi

Kata Pengantar	iii
Daftar Isi	iv
Daftar Gambar	v
Daftar Tabel	vi
Daftar Diagram	vii
BAB I, MENGENAL BAHASA PEMROGRAMAN	1
BAB II, FREE PASCAL	7
BAB III, BAHASA PASCAL	24
BAB IV, OPERASI DI PASCAL	51
BAB V, STATEMENT	54
BAB VI, PROSEDUR DAN FUNGSI	60
BAB VII, STANDARD INPUT / OUTPUT	65
INDEX	68
Lampiran	71

## Daftar Gambar

Gambar 1.	John Backus, pencipta FORTRAN	2
Gambar 2.	Tampilan awal proses instalasi	9
Gambar 3.	Poses instalasi	9
Gambar 4.	Tampilan awal IDE Free Pascal	10
Gambar 5.	Kotak dialog untuk mengelola kata kunci	12
Gambar 6.	Kotak dialog untuk mengatur code templates	13
Gambar 7.	Kotak dialog untuk menambahkan parameter	14
Gambar 8.	Tampilan breakpoint	15
Gambar 9.	Kotak dialog watch.	15
Gambar 10.	Window watch	16
Gambar 11.	Window untuk call stack	16
Gambar 12.	Tabel ASCII di Free Pascal	18
Gambar 13.	Tampilan fasilitas calculator	19

## Daftar Tabel

Tabel 1.	Operasi pada fasilitas calculator	19
Tabel 2.	Shortcut untuk window IDE dan Help	20
Tabel 3.	Shortcut untuk me-RUN dan debugging	20
Tabel 4.	Shortcut untuk navigasi kursor	21
Tabel 5.	Shortcut untuk blok	21
Tabel 6.	Shortcut untuk pengeditan	22
Tabel 7.	Shortcut untuk merubah blok	22
Tabel 8.	Shortcut lain-lain	23
Tabel 9.	Perbandingan kata tercadang Free Pascal dan Turbo Pascal	28
Tabel 10.	Tipe bilangan bulat di Free Pascal	37
Tabel 11.	Daftar operator pada bilangan bulat	38
Tabel 12.	Tabel operator pada tipe real	39
Tabel 13.	Tabel operator untuk operasi aritmatika	51
Tabel 14.	Tabel operator logika	52
Tabel 15.	Tabel operator boolean	52
Tabel 16.	Tabel operator himpunan	53
Tabel 17.	Tabel operator relasi	53

## Daftar Diagram

Diagram 1.	Tipe data di Turbo Pascal	31
Diagram 2.	Tipe data di Free Pascal	32



---

---

# Bab 1

## MENGENAL BAHASA PEMROGRAMAN

---

---

Pada dasarnya, sebuah komputer tidak dapat mengerjakan apapun tanpa adanya perintah dari manusia. Perintah-perintah yang terstruktur dan sistematis untuk membuat komputer bekerja sesuai dengan apa yang diinginkan disebut **program**. Apa yang dapat dilakukan oleh program komputer? Komputer dapat diprogram untuk berbagai hal, misalnya diprogram untuk melakukan perhitungan suatu ekspresi matematika dan menampilkan hasilnya di layar monitor, diprogram untuk memainkan sebuah lagu, diprogram untuk mengurutkan data (misalnya mengurutkan data nama siswa, data nilai siswa), diprogram untuk permainan, diprogram untuk menggambar dan sebagainya. Program-program semacam itu dibuat oleh manusia, syarat utama dalam membuat program adalah perintah-perintah yang diberikan dalam program tersebut harus dimengerti oleh komputer.

Sayangnya, komputer hanya dapat mengerti sebuah bahasa yang disebut bahasa mesin, bahasa yang sangat berbeda dari bahasa manusia dan terlebih lagi akan amat menyulitkan untuk membuat sebuah program dalam bahasa mesin ini. Manusia menginginkan sebuah bahasa komputer yang sederhana yang dapat dimengerti dan mudah dipelajari oleh manusia sekaligus dapat dimengerti oleh komputer.

Bahasa komputer tersebut disebut bahasa pemrograman (*programming language*). Yang perlu diingat, konsep bahasa pemrograman adalah merubah/menerjemahkan perintah-perintah (program) yang diberikan oleh manusia ke dalam bahasa mesin yang dapat dimengerti oleh komputer. Jadi bahasa pemrograman adalah sarana interaksi antara manusia dan komputer. Seperti tujuan semula, bahasa pemrograman dibuat mudah dipelajari dan dimengerti agar manusia dapat mudah membuat program komputer dengan bahasa pemrograman ini (tak perlu menggunakan bahasa mesin untuk membuat program komputer).

Penerjemah bahasa pemrograman dibedakan menjadi tiga macam, yaitu:

1. Assembler adalah program yang digunakan untuk menerjemahkan kode sumber dalam bahasa rakitan (assembly) ke dalam bahasa mesin

2. Kompiler adalah program penerjemah yang mengonversi semua kode sumber selain dalam bahasa rakitan menjadi kode objek. Hasil berupa kode objek inilah yang bisa dijalankan oleh komputer. Perlu diketahui, proses untuk melakukan penerjemahan ini biasa disebut kompilasi. Bahasa pemrograman yang menggunakan proses kompilasi adalah: Bahasa COBOL, Pascal, Bahasa C
3. Interpreter adalah program yang menerjemahkan satu per satu instruksi dalam kode sumber dan kemudian segera menjalankan instruksi yang telah diterjemahkan tersebut. Bahasa seperti BASIC pada awalnya menggunakan konsep interpreter ini.

Pada intinya, bahasa pemrograman digunakan untuk mempermudah manusia dalam berinteraksi dengan komputer. Syarat utama untuk membuat program komputer adalah dengan menggunakannya sesuai dengan kaidah-kaidah yang berlaku dalam bahasa pemrograman tersebut. Masing-masing bahasa pemrograman mempunyai ciri khas/kaidah tersendiri. Karena itu, sebelum membuat sebuah program dengan menggunakan bahasa pemrograman, sangat wajib untuk mengerti tentang aturan penulisan bahasa pemrograman tersebut.

Saat ini ada banyak bahasa pemrograman yang beredar di pasaran. Masing-masing memberikan kemudahan dan fasilitas untuk membuat sebuah program komputer yang sesuai dengan keinginan.

### **FORTRAN**

FORTRAN kepanjangan dari Formula Translation. Pertama kali dikembangkan pada tahun 1956 oleh John Backus di IBM. Ditujukan untuk mempermudah pembuatan aplikasi matematika, ilmu pengetahuan dan teknik. Merupakan bahasa pemrograman tingkat tinggi pertama kali.



**Gambar 1.** John Backus, pencipta FORTRAN

Keunggulan FORTRAN terletak pada dukungan untuk menangani perhitungan termasuk bilangan kompleks. Kelemahan bahasa ini terletak pada operasi masukan/keluaran yang sangat kaku. Selain itu kode sumbernya lebih sulit dipahami dibandingkan dengan bahasa pemrograman tingkat tinggi lainnya.

Contoh program dalam bahasa FORTRAN:

```
// JOB
// FOR
* ONE WORD INTEGERS
* IOCS(DISK, TYPEWRITER, KEYBOARD, PAPERTAPE)
  DIMENSION IEMG(10, 15), IEMG1(13)
  DEFINE FILE 12(80, 150, U, K)
  WRITE(1, 10)
  10 FORMAT(' PAPERTAPE' //' GIVE NUMBER EXPERIMENT (1-5 IN INT)')
  READ(6, 30) M
  30 FORMAT(I1)
  PAUSE 1
  DO 25 N=1, 16
  DO 15 I=1, 15
  READ(4, 20) IEMG1
  20 FORMAT(13I4)
  DO 15 J=4, 13
  J3=J-3
  15 IEMG(J3, I)=IEMG1(J)
  NE=N+(M-1)*16
  25 WRITE(12' NE) IEMG
  CALL EXIT
  END
// DUP
*DELETE SJA1
*STORECI WS UA SJA1
*FILES(12, EMG)
```

## COBOL

COBOL (Common Business Oriented Language) dikembangkan tahun 1959 dan tergolong sebagai bahasa tingkat tinggi. Sesuai dengan kepanjangannya, bahasa ini ditujukan untuk mempermudah pembuatan aplikasi di bidang bisnis. Sejauh ini bahasa ini masih banyak digunakan terutama di lingkungan komputer minikomputer dan mainframe.

Keunggulan COBOL adalah:

- ? Sintaksnya yang menggunakan kata-kata bahasa Inggris sehingga mempermudah programmer.
- ? Kemudahan terhadap penanganan file.
- ? Kemudahan terhadap masukan / keluaran program

Contoh program dalam bahasa COBOL:

```
000100 IDENTIFICATION DIVISION.  
000200 PROGRAM-ID.      HELLOWORLD.  
000300  
000400*  
000500 ENVIRONMENT DIVISION.  
000600 CONFIGURATION SECTION.  
000700 SOURCE-COMPUTER.  RM-COBOL.  
000800 OBJECT-COMPUTER.  RM-COBOL.  
000900  
001000 DATA DIVISION.  
001100 FILE SECTION.  
001200  
100000 PROCEDURE DIVISION.  
100100  
100200 MAIN-LOGIC SECTION.  
100300 BEGIN.  
100400     DISPLAY " " LINE 1 POSITION 1 ERASE EOS.  
100500     DISPLAY "Hello world!" LINE 15 POSITION 10.  
100600     STOP RUN.  
100700 MAIN-LOGIC-EXIT.  
100800     EXIT.
```

## BASIC

BASIC adalah kepanjangan dari *Beginner All-purpose Symbolic Instruction Code* Dikembangkan tahun 1965 di Darmouth College. Penciptanya adalah John Kemeny dan Thomas Kurtz. Awalnya BASIC digunakan sebagai pengajaran dasar untuk bahasa pemrograman sederhana.

Keunggulan BASIC terletak pada kemudahannya untuk dipakai dan penggunaan bahasa Inggris yang mirip dengan kehidupan sehari-hari sebagai sintaksnya. BASIC merupakan bahasa pemrograman yang sangat populer sebelum Pascal dibuat.

Contoh program dalam bahasa BASIC

```
REM Program mencari rata-rata 3 buah bilangan  
  
INPUT "Masukkan tiga buah bilangan : ", a, b, c  
rata=(a+b+c)/3  
PRINT "Rata-rata ketiga bilangan adalah : "; rata
```

## PASCAL

Sejarah perkembangan Pascal dimulai pada tahun 1960, yaitu ketika bahasa pemrograman ALGOL 60 digunakan sebagai *algorithmic language* yang digunakan untuk memecahkan masalah sehari-hari dengan menggunakan komputer. Nama Pascal sendiri diambil dari nama seorang ahli matematika dan ilmu pengetahuan bangsa Perancis, yaitu Blaise Pascal (1623-1662). Niklaus Wirth dari Sekolah Teknik Tinggi Zurich - Swiss, menjadi terkenal sebagai perancang bahasa Pascal, *compiler* pertama yang dilaksanakan pada tahun 1970, kemudian direvisi pada tahun 1973 bersama K. Jensen. Kemudian pada tahun 1983 bahasa Pascal dapat dibakukan secara resmi dengan adanya Pascal Standard dari ISO.

Keunggulan bahasa Pascal adalah keteraturan dalam pembuatan program dan kelengkapan struktur data. Contoh program dalam bahasa Pascal:

```
PROGRAM CariMin;  
{Mencari Bilangan terkecil dari dua buah bilangan}  
  
VAR  
    x, y, min: integer;  
  
BEGIN  
    WRITE(' Bilangan pertama : '); READLN(x);  
    WRITE(' Bilangan kedua : '); READLN(y);  
    IF x>y THEN  
        min:=y  
    ELSE  
        min:=x;  
    WRITE(' Bilangan terkecil : ', min);  
END.
```

## BAHASA C

Bahasa C diciptakan oleh Brian W. Kernighan dan Dennis M. Ritchie pada tahun 1972 di laboratorium Bell AT&T. Bahasa ini menggabungkan kemampuan pengendalian mesin dalam aras rendah dan struktur data serta struktur kontrol aras tinggi. Jadi dapat disebut bahasa C adalah bahasa pemrograman yang menggabungkan kemudahan pengontrolan hardware dalam bahasa pemrograman tingkat rendah serta struktur kontrol dalam bahasa tingkat tinggi. Bahasa C ini digunakan untuk menyusun sistem operasi UNIX dan Linux.

Keunggulan bahasa C adalah:

- ? Sifat portabilitas, yaitu kode sumber pada sebuah platform dapat ditransfer ke platform lain tanpa ada perubahan
- ? Kemudahan akses terhadap hardware

? Cepat dan efisien

Pada tahun 1983, Bjarne Stroustrup mengembangkan bahasa C yang pada mulanya disebut sebagai “a better C”. Namun kemudian bahasa ini dikenal dengan nama C++ (C Plus plus) yang mengunggulkan kelebihanannya sebagai bahasa pemrograman berorientasi objek.

Contoh program dalam Bahasa C:

```
/*Mencari Bilangan terkecil dari dua buah bilangan*/  
#include <stdio.h>  
  
main ()  
{  
    int x,y, min;  
    printf ("Bilangan pertama : ");  
    scanf ("%1f", &x);  
  
    printf ("Bilangan kedua : ");  
    scanf ("%1f", &y);  
  
    if x>y  
        min=x  
    else  
        min=y;  
    printf("Bilangan terkecil : %1f\n", min);  
}
```

## BAHASA JAVA

Bahasa Java dikembangkan oleh Sun Microsystem pada tahun 1995. Merupakan bahasa yang berorientasi objek. Kode Java dikompilasi dalam format yang disebut bytecode. Bytecode ini dapat dijalankan di semua komputer yang telah dilengkapi dengan program Java Interpreter dan Java Virtual Machine.

Java sangat populer karena pada masa awal Internet menjadi populer, Java telah menyediakan sarana untuk membuat program (yang disebut sebagai applet) yang dapat berjalan pada *web browser* seperti Internet Explorer, Netscape Navigator.

Contoh program dalam bahasa Java:

```
Public class SayHello {  
    Public static void main(String[] args {  
        System.out.println("Hello world!");  
    }  
}
```

---

---

# Bab 2

## FREE PASCAL

---

---

**Mengapa Pascal ?** dalam TOKI ? Dan mengapa compilernya adalah Free Pascal? Alasan di bawah ini setidaknya akan menjawab pertanyaan di atas,

- ? *Very Clean Language.* Bahasa Pascal adalah bahasa yang sangat mudah dibaca dan dikelola dibandingkan dengan bahasa C.
- ? *No Makefiles.* Tidak seperti kebanyakan bahasa pemrograman lain (Clipper, Java, C), Pascal tidak memerlukan makefiles. Ini dapat menghemat banyak waktu, compiler hanya melakukan proses kompilasi terhadap file-file memang perlu dikompilasi.
- ? *Pascal Compiler are FAST.* Begitu proses kompilasi dijalankan, maka hampir pada saat yang bersamaan program tersebut telah selesai dikompilasi, bahkan untuk program yang besar
- ? *Each unit has it's own identifiers.* Pada Free Pascal, nama identifier (variabel, konstanta, tipe data, fungsi, prosedur) tidak harus berbeda untuk tiap unit. Walaupun dalam sebuah unit terdapat prosedur clrscr misalnya, maka dalam program yang menggunakan unit tersebut, nama clrscr masih dapat digunakan sebagai identifier.
- ? *Integrated Develoment Environment.* Free Pascal mempunyai IDE (editor) yang dapat bekerja dalam beberapa platform, di mana kita dapat langsung mengetikkan program, mengkompilasi serta melakukan debugging.
- ? *Great integration with Assembler.* Assembler adalah bahasa mesin yang dapat secara langsung mengakses hardware. Compiler Free Pascal mampu menggabungkan kemampuan dalam bahasa Assembler ini dengan bahasa Pascal.
- ? *Object Oriented Programming (OOP).* Bagi anda yang serius dengan pemrograman, tentu sangat tertarik dengan OOP ini (karena arah pemrograman di Windows adalah OOP). Free Pascal mensupport secara penuh OOP ini dalam bahasa Pascal.
- ? *Support Database.* Database yang disupport oleh Free Pascal adalah PostgreSQL, MySQL, Interbase, ODBC.

- ? *Smartlinking*. Free Pascal smart linker akan meninggalkan semua variabel-variabel yang tidak diperlukan sehingga program akan berukuran sangat kecil.
- ? *Compatible*. Kompatibilitas Free Pascal terhadap compiler Pascal lain sangat tinggi. Hampir semua program di Turbo Pascal atau di Delphi dapat dikompilasi di Free Pascal.

### **Tentang Free Pascal**

Free Pascal adalah compiler untuk bahasa Pascal. Free Pascal ini didistribusikan secara gratis di bawah lisensi GNU Public. Versi terakhir Free Pascal dapat di download di: <http://www.freepascal.org/>. Hingga bulan Juni 2004, versi Free Pascal yang paling stabil adalah versi 1.0.10. Sedangkan versi 1.9x masih dalam tahap pengembangan.

Free Pascal tersedia untuk berbagai macam prosesor, yaitu Intel x86, Motorola 680x0 (untuk versi 1.0 saja) dan PowerPC (mulai versi 1.9.2). Selain itu juga mendukung berbagai sistem operasi, yaitu: Linux, FreeBSD, NetBSD, MacOSX, DOS, Win32, OS/2, BeOS, SunOS (Solaris), QNX dan Classic Amiga.

### **Proses Instalasi Free Pascal**

Sebelum melakukan instalasi Free Pascal, hardware yang dibutuhkan adalah:

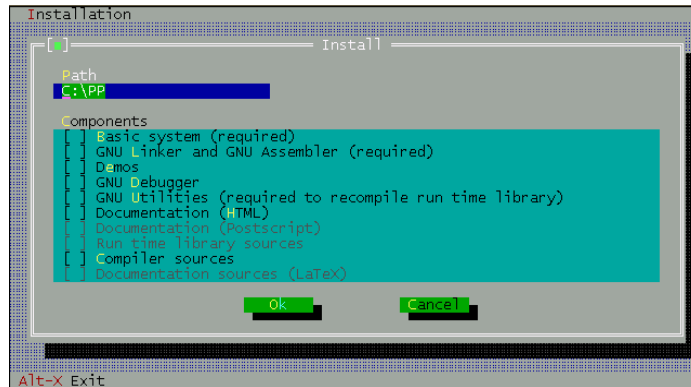
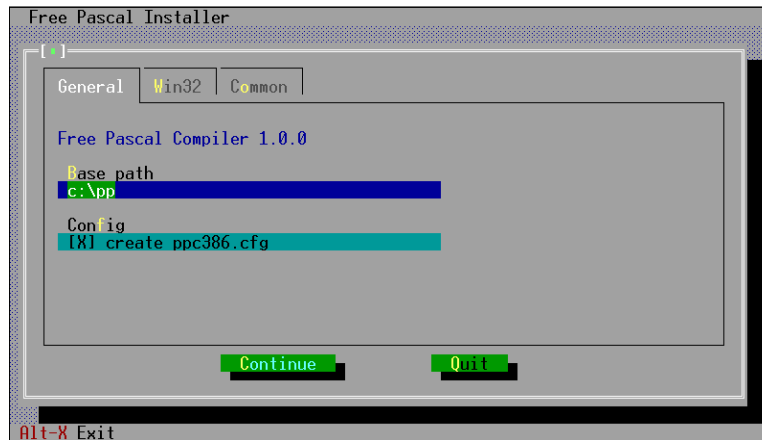
- ? Prosesor Intel 80386 (atau yang lebih tinggi). Coprosesor tidak mutlak dibutuhkan, walaupun dapat memperlambat jalannya program jika menggunakan perhitungan floating point (perhitungan dengan angka di belakang desimal).
- ? Memori sedikitnya 4 Mb.
- ? Free Space di hddisk sedikitnya 3 Mb

Untuk melakukan instalasi program Free Pascal, harus punya terlebih dulu versi terbaru dari FreePascal sesuai dengan platform untuk menjalankannya. Free Pascal didistribusikan dalam bentuk file Zip yang dapat dibuka/diekstrak dengan menggunakan WinZip atau program lain sejenis.

Berikut ini adalah langkah melakukan instalasi Free Pascal untuk DOS atau Windows.

1. Cari dan jalankan file INSTALL.EXE di folder Free Pascal yang telah diekstrak tadi, lihat gambar 2 dan 3



**Gambar 2.** Tampilan awal proses instalasi**Gambar 3.** Proses instalasi

2. Program instalasi akan memberikan pilihan komponen-komponen yang hendak diinstal.
3. Agar Free Pascal dapat dijalankan dari sembarang direktori dalam sistem, maka variabel path harus berisi: SET PATH=%PATH%;C:\PP\BIN\GO32V2 (pada DOS / ditambahkan di file AUTOEXEC.BAT). Atau SET PATH=%PATH%;C:\PP\BIN\WIN32 (untuk Windows) dan SET PATH=%PATH%;C:\PP\BIN\OS2 (untuk OS/2).


**IDE (*Integrated Developing Environment*)**

IDE menyediakan user interface yang nyaman untuk penulisan program, proses kompilasi serta debugging. IDE Free Pascal adalah aplikasi mode teks yang sama untuk setiap sistem operasi. Dimodelkan mirip dengan IDE di Turbo Pascal versi 7.0. Saat ini IDE Free Pascal ada untuk sistem operasi DOS, Windows, dan Linux. Untuk menjalankan IDE Free Pascal, jalankan file FP.EXE pada direktori C:\PP\BIN\GO32V2 (DOS) atau C:\PP\BIN\WIN32 (Windows).



**Gambar 4.** Tampilan awal IDE Free Pascal

Pada bagian paling atas disebut dengan *menu bar* dan bagian bawah disebut *status bar*. Area di antaranya disebut *desktop*.

Status bar menunjukkan shortcut keyboard yang akan sering digunakan dan memungkinkan dijalankan dengan mengkliknya. Untuk keluar dari IDE ini, dapat memilih menu File  Exit atau dengan menggunakan shortcut kombinasi tombol Alt-X.

Catatan: Jika file FP.ANS ditemukan pada current directory (directory / folder tempat IDE dijalankan), maka tampilan desktop akan diambil dari file tersebut. File ini berisi perintah penggambaran untuk layar dalam standard ANSI.

**Pengeditan pada IDE**

Pengeditan teks pada IDE sama seperti pada editor lain.

### ? **Insert Mode**

Secara standard, mode pengeditan pada IDE Free Pascal adalah insert mode. Artinya semua teks yang diketikkan akan diletakkan di samping teks yang ada setelah posisi kursor. Pada Overwrite mode, teks yang diketikkan akan menimpa teks yang ada. Untuk mengubah dari insert mode ke overwrite mode atau sebaliknya, tekan Ins atau Ctrl-V.

### ? **Blok**

Teks dapat diblok dengan 3 cara.

1. Menggunakan mouse dengan melakukan drag pada teks yang hendak diblok
2. Menggunakan keyboard dengan menekan Ctrl-K B sebagai awal teks yang hendak diblok dan Ctrl-K K sebagai akhir dari blok.
3. Menggunakan keyboard dengan menahan tombol Shift, lalu mengarahkan kursor (dengan tombol panah) ke akhir dari blok.
4. Ctrl – K L  $\approx$  untuk memilih satu baris
5. Ctrl – K T  $\approx$  untuk memilih satu kata

Perintah-perintah yang berlaku saat ada pengeblokan adalah:

1. Menggerakkan blok ke lokasi kursor (Ctrl – K V)
2. Kopi blok ke lokasi kursor (Ctrl – K C)
3. Hapus blok (Ctrl – K Y)
4. Tulis blok ke file (Ctrl – K W)
5. Membaca isi file (Ctrl – K R)
6. Meng-indentasi (maju) blok (Ctrl – K I)
7. Meng-unindentasi (mundur) blok (Ctrl – K U)
8. Mencetak blok (Ctrl – K P)

### ? **Bookmark (menandai)**

Untuk menandai suatu posisi, Free Pascal dapat memberikan tanda hingga 9 bookmarks dalam sebuah file. Untuk menandai suatu posisi, dengan menggunakan Ctrl – K | [1-9], sedangkan untuk menuju ke tanda tersebut menggunakan Ctrl – Q | [1-9]

## **Syntax Highlight**

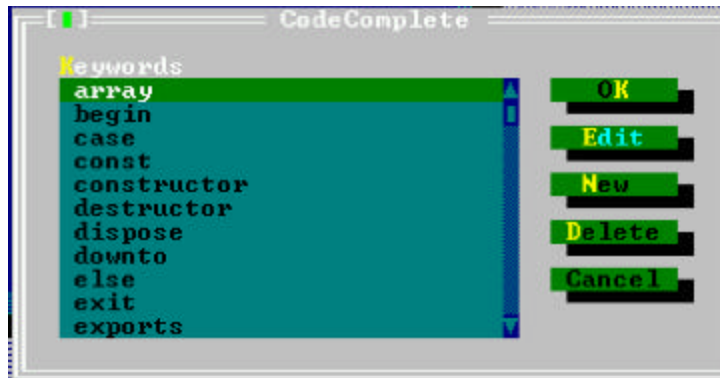
IDE Free Pascal mampu memberikan Syntax highlight (warna yang berbeda untuk tiap-tiap sintaks). Syntax highlight ini dapat diatur di menu Options  $\approx$  Environment  $\approx$  Colors. Pada kotak dialog color, grup syntax harus dipilih terlebih dulu. Yang dapat diubah syntax highlight adalah:

- ? **Whitespace**, warna teks yang kosong / spasi antar kata.
- ? **Comments**, komentar dalam Free Pascal
- ? **Reserved Word**, kata-kata yang dikenal oleh Free Pascal
- ? **Strings**, ekspresi untuk string
- ? **Numbers**, angka dalam notasi desimal
- ? **Hex Number**, angka dalam notasi heksa desimal
- ? **Assembler**, blok untuk bahasa Assembler
- ? **Symbol**, simbol-simbol yang dikenali
- ? **Directive**, compiler directive
- ? **Tabs**, karakter tab

### Code Completion

Code completion artinya editor akan memberikan kemungkinan perintah yang akan diketik untuk dilengkapi. Ini dilakukan dengan melakukan pengecekan teks yang diketikkan, lalu editor akan memberikan daftar kata kunci yang mungkin dimaksud oleh user. Tekan Enter untuk menerima daftar kata kunci tersebut.

Code completion dapat diatur di kotak dialog Code Completion lewat Options ➤ Preferences ➤ Codecompletion. Daftar kata kunci dapat dikelola di sini.



**Gambar 5.** Kotak dialog untuk mengelola kata kunci

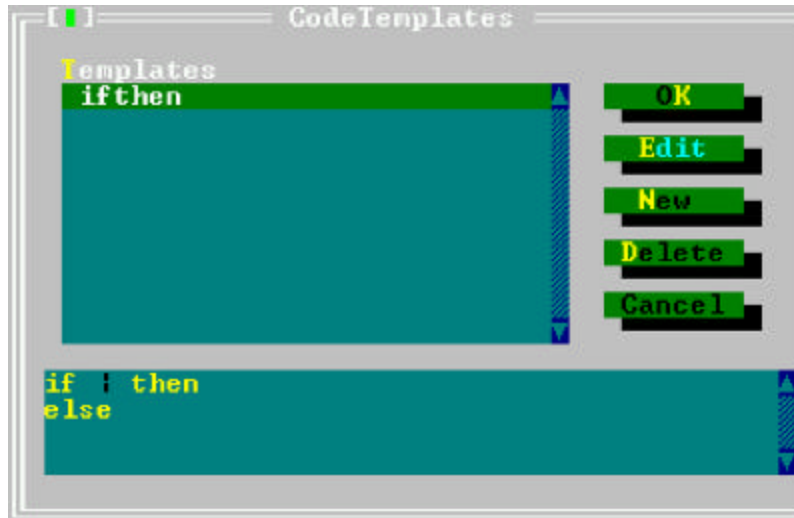
### Code Templates

Code Templates adalah sebuah cara untuk memasukkan kode-kode perintah secara sekaligus. Setiap code templates diidentifikasi dengan nama yang unik. Nama untuk code templates dapat diasosiasikan dengan kode-kode perintah.

Misalnya nama ifthen dapat diasosiasikan dengan kode-kode perintah berikut:

```
IF THEN  
Begin  
End
```

Code templates dapat dimasukkan dengan mengetikkan namanya, lalu menekan Ctrl-J ketika kursor berada di posisi nama template. Code templates ini dapat diatur di menu options  $\searrow$  preferences  $\searrow$  Codetemplates.



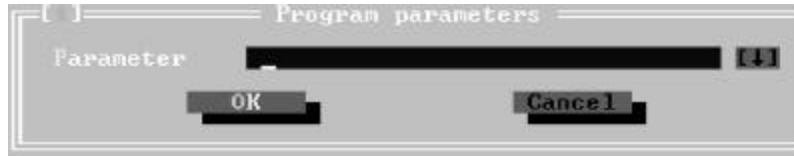
**Gambar 6.** Kotak dialog untuk mengatur code templates

### Menjalankan Program

Program yang telah dikompilasi dapat dijalankan langsung lewat IDE, yaitu dengan :

- ? Pilih menu Run  $\searrow$  Run atau
- ? Tekan Ctrl – F9

Bila ada parameter yang perlu diberikan untuk program, dapat ditambahkan pada Run  Parameter seperti pada gambar 7.



**Gambar 7.** Kotak dialog untuk menambahkan parameter

Saat program di-run, akan terus berjalan hingga:

1. Program berhenti dengan normal
2. Terjadi error
3. Karena ada breakpoint dalam program atau
4. Program di reset oleh user.


Alternatif terakhir hanya mungkin terjadi apabila program dikompilasi dengan debug information


Untuk menjalankan program hingga baris tertentu, dapat dilakukan dengan:

- ? Pilih Run  Goto Cursor atau Tekan F4 pada posisi yang diinginkan

Ini juga dapat dilakukan hanya apabila program dikompilasi dengan debug information.

Program dapat dijalankan dengan mengeksekusi baris demi baris. Tekan F8 untuk menjalankan sebuah perintah. Penekanan F8 secara berturut-turut akan menjalankan program baris demi baris. Jika diinginkan agar eksekusi masuk dalam subrutin, maka dapat dilakukan dengan menekan F7.

Saat masuk dalam sebuah subrutin, pilih Run  Until return untuk menjalankan program hingga akhir subrutin tersebut. Jika program memang harus dihentikan sebelum berhenti dengan normal, maka dapat melakukan:

- ? Pilih Run  Program reset atau Tekan Ctrl – F2

Program akan segera berhenti. Namun perlu diingat bahwa file-file yang terbuka (oleh perintah assign, reset) tidak secara otomatis tertutup.


## Debug Program

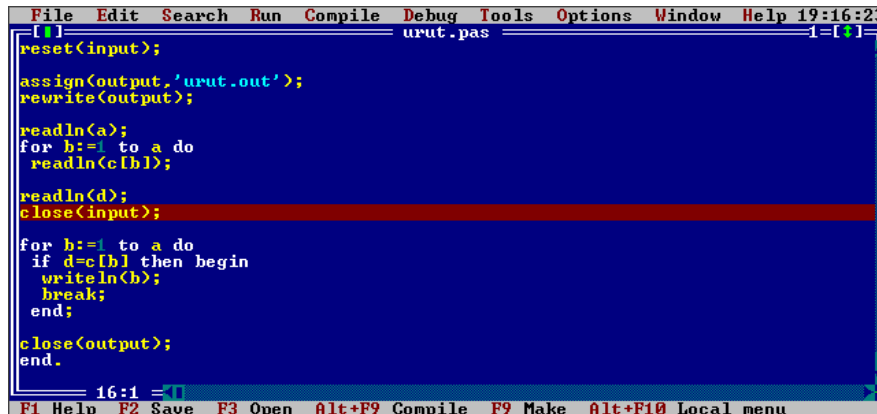
Dengan melakukan debugging program dapat dimungkinkan:

- ? Menjalankan program baris demi baris
- ? Menjalankan program hingga baris tertentu (breakpoint)


? Melihat isi dari variabel memori sementara program berjalan

Breakpoint akan mengakibatkan program berhenti pada baris di mana breakpoint diset/ditentukan. Pada saat ini, proses dikembalikan pada IDE dan dimungkinkan untuk dijalankan kembali.

Untuk melakukan set menentukan breakpoint, pada baris yang diinginkan pilih menu Debug  Breakpoint atau dengan menekan Ctrl – F8. Untuk menghapus breakpoint, dapat dengan menekan kembali Ctrl – F8.



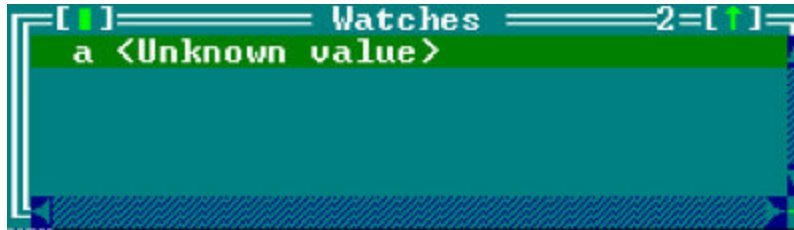
**Gambar 8.** Tampilan breakpoint

Fasilitas Watch adalah fasilitas untuk melihat isi variabel memori pada saat program dihentikan (oleh breakpoint). Watch ditampilkan di window baru dengan memilih menu Debug  Add Watch atau dengan menekan Ctrl – F7. Gambar 9 menunjukkan kotak dialog untuk watch.





**Gambar 9.** Kotak dialog watch.

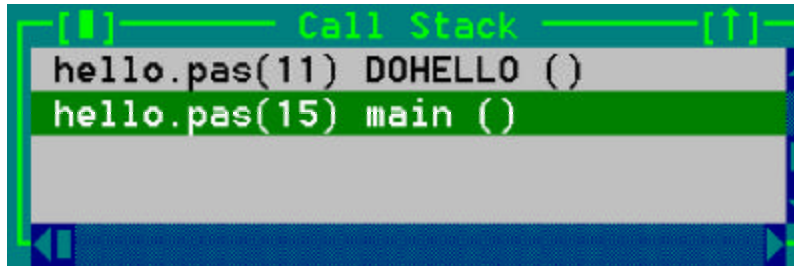
Setelah memasukkan nilai pada kotak dialog watch, tekan Enter. Akan muncul window baru, yaitu window watch (gambar 10).



**Gambar 10.** Window watch

Untuk berpindah antar window (dari watch ke window program atau sebaliknya, dapat menggunakan tombol F6 atau Shift-F6). Sedangkan untuk mengatur window, gunakan menu Window  Cascade.

Call stack adalah sebuah fasilitas di IDE untuk melihat eksekusi subrutin (termasuk parameternya) yang saat ini dijalankan program. Untuk menampilkan call stack lewat menu Debug  Call Stack atau dengan menekan Ctrl – F3.



**Gambar 11.** Window untuk call stack

Dengan menekan spasi pada window call stack, baris program akan diarahkan pada baris pemanggilan subrutin.

### **Free Pascal vs Turbo Pascal**

Berikut adalah hal-hal yang diizinkan di Turbo Pascal namun tidak berlaku di Free Pascal, yaitu:

1. Duplikasi label untuk case tidak diizinkan. Ini merupakan bug pada Turbo Pascal, dan Free Pascal masih tetap mempertahankannya.



2. Parameter list yang dideklarasikan pada function atau procedure sebelumnya, harus benar-benar persis untuk function atau procedure yang sesungguhnya.
3. Variabel Mem, MemW, MemL dan Port tidak berlaku lagi di Free Pascal.
4. PROTECTED, PUBLIC, PUBLISHED, TRY, FINALLY, EXCEPT, RAISE merupakan kata-kata tercadang (*reserved word*)
5. Kata tercadang FAR dan NEAR tidak lagi berlaku lagi, karena Free Pascal adalah compiler 32 bit
6. INTERRUPT hanya berlaku untuk mode DOS
7. File yang dibuka dengan perintah rewrite hanya dapat ditulisi saja, tidak dapat dibaca. Untuk dapat membaca, maka harus diberikan perintah reset.

Berikut adalah tambahan Free Pascal dibandingkan Turbo Pascal:

1. Ada lebih banyak reserved word. Lihat session IV – nomer 5
2. Function dapat mengembalikan tipe kompleks seperti record dan array
3. Function dapat dihandle dalam function itu sendiri, contoh:

```
function a : longint;  
begin  
  a:=12;  
  while a>4 do  
  begin  
    {...}  
  end;  
end;
```

Contoh di atas dapat dijalankan dalam Turbo Pascal, namun compiler mengasumsikan bahwa perintah while a>4 do adalah pemanggilan fungsi secara rekursif (pada Turbo Pascal). Untuk Free Pascal, pemanggilan rekursif pada kasus di atas adalah dengan menambahkan () (kurung buka tutup) pada a.

```
function a : longint;  
begin  
  a:=12;  
  { pemanggilan secara rekursif }  
  if a()>4 then  
  begin  
    {...}  
  end;  
end;
```

4. Pemanggilan EXIT dapat dengan memberikan nilai kembali

```
function a : longint;
begin
  a:=12;
  if a>4 then
  begin
    exit(a*67); {function menghasilkan a*67 }
  end;
end;
```


5. Free Pascal mendukung adanya fungsi yang *overloading*. Artinya kita bisa mendefinisikan nama fungsi yang sama dengan parameter yang berbeda, contoh:

```
procedure DoSomething (a : longint);
begin
  {...}
end;

procedure DoSomething (a : real);
begin
  {...}
end;
```


6. Mendukung nama file panjang. Mulai Windows 95, nama file dapat sepanjang 255 karakter. Free Pascal dapat mendukung penyimpanan dengan nama file panjang.

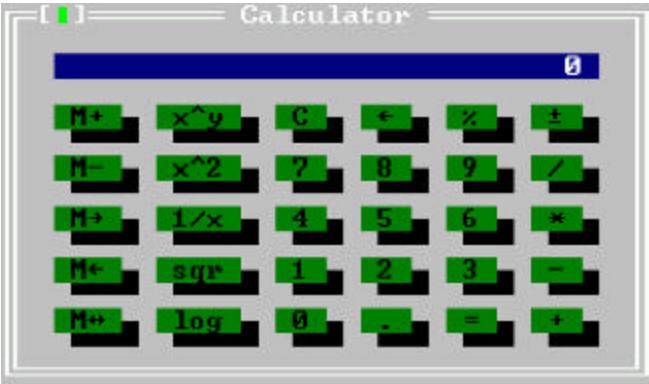
### Fasilitas Tambahan

Tabel ASCII dapat ditampilkan langsung lewat IDE, yaitu pada menu Tools  Ascii Table. Untuk memasukkan karakter ASCII pada editor, dapat langsung melakukan double click atau tekan Enter.



**Gambar 12.** Tabel ASCII di Free Pascal


Calculator juga tersedia lewat menu Tools  Calculator. Fasilitas calculator pada free pascal tidak mensupport tanda kurung. Hasil perhitungan dari kalkulator ini dapat langsung diletakkan pada editor dengan menekan Ctrl - Enter.



**Gambar 13.** Tampilan fasilitas calculator

Calculator ini mendukung operasi standard dalam matematika, seperti pada tabel 1

**Tabel 1.** Operasi pada fasilitas calculator

Operasi	Tombol di calculator	Penekanan keyboard
Penambahan dua bilangan	+	+
Pengurangan dua bilangan	-	-
Perkalian dua bilangan	*	*
Pembagian dua bilangan	/	/
Menghapus bilangan terakhir		Backspace
Menghapus bilangan	C	C
Merubah tanda	?	
Kalkulasi persen	%	%
Hasil	=	Enter
Pemangkatan	X^y	
Inverse	1/x	
Akar pangkat dua	Square root	
Logaritma natural	Log	
Pangkat dua	X^2	

### Daftar Shortcut pada IDE

Shortcut akan berguna untuk mempercepat pengeditan pada IDE. Berikut adalah daftar shortcut yang dikenal IDE Free Pascal.

**Tabel 2.** Shortcut untuk window IDE dan Help

Perintah	Shortcut	Alternatif
Help	F1	
Topik help terakhir	Alt – F1	
Mencari kata pada help	Ctrl – F1	
Indeks help	Shift – F1	
Menutup window aktif	Alt – F3	
Zoom / Unzoom window	F5	
Menggerakkan window	Ctrl – F5	
Berpindah ke window berikutnya	F6	
Berpindah ke window sebelumnya	Shift – F6	
Menu	F10	
Menu lokal	Alt – F10	
Daftar Window	Alt – 0	
Mengaktifkan window	Alt – [no. Urut window]	
Keluar dari IDE	Alt – X	

**Tabel 3** Shortcut untuk me-RUN dan debugging

Perintah	Shortcut	Alternatif
Mereset program	Ctrl – F2	
Menampilkan Call stack	Ctrl – F3	
Jalankan sampai posisi kursor	Ctrl – F4	
Melihat tampilan layar	Alt – F5	
Trace / penelusuran program	F7	
Menambah Watch	Ctrl – F7	
Step over	F8	
Make	F9	
Menjalankan program	Ctrl – F9	
Mengkompilasi program aktif	Alt – F9	
Message	F11	
Pesan Compiler	F12	

**Tabel 4.** Shortcut untuk navigasi kursor

Perintah	Shortcut	Alternatif
Ke kiri	Panah kiri	Ctrl – S
Ke kanan	Panah kanan	Ctrl – D
Ke atas	Panah atas	Ctrl – E
Ke bawah	Panah bawah	Ctrl – X
Satu huruf ke kiri	Ctrl – panah kiri	Ctrl – A
Satu huruf ke kanan	Ctrl – panah kanan	Ctrl – F
Scroll satu baris ke atas	Ctrl – W	
Scroll satu baris ke bawah	Ctrl – Z	
Satu halaman ke atas	Page Up	Ctrl – R
Satu halaman ke bawah	Page Down	
Awal baris	Home	Ctrl – Q S
Akhir baris	End	Ctrl – Q D
Awal baris program	Ctrl – Home	Ctrl – Q E
Akhir baris program	Ctrl – End	Ctrl – Q X
Posisi kursor terakhir	Ctrl – Q P	

**Tabel 5.** Shortcut untuk blok

Perintah	Shortcut	Alternatif
Ke awal blok	Ctrl – Q B	
Ke akhir blok	Ctrl – Q K	
Blok baris	Ctrl – K L	
Cetak blok	Ctrl – K P	
Pilih kata	Ctrl – K T	
Hapus teks yang diblok	Ctrl – Del	Ctrl – K Y
Kopi blok	Ctrl – K C	
Pindah blok	Ctrl – K V	
Kopi blok ke clipboard	Ctrl – Ins	
Pindah blok ke clipboard	Shift – Del	
Indentasi blok satu kolom	Ctrl – K I	
Un-indentasi blok satu kolom	Ctrl – K U	
Menyisipkan teks dari clipboard	Shift – Ins	
Tulis blok ke file	Ctrl – K W	
Menjadikan blok huruf besar	Ctrl – K N	

**Tabel 6.** Shortcut untuk pengeditan

Perintah	Shortcut	Alternatif
Hapus karakter	Del	Ctrl – G
Hapus di sebelah kiri kursor	Backspace	Ctrl – H
Hapus baris	Ctrl – Y	
Hapus sampai baris terakhir	Ctrl – Q Y	
Hapus kata	Ctrl – T	
Menyisipkan baris	Ctrl – N	
Mode Insert / overwrite	Ins	Ctrl – V

**Tabel 7.** Shortcut untuk merubah blok

Perintah	Shortcut	Alternatif
Mengawali blok	Ctrl – K B	
Mengakhiri blok	Ctrl – K K	
Menghilangkan blok	Ctrl – K Y	
Menambah blok satu karakter ke kiri	Shift – Panah kiri	
Menambah blok satu karakter ke kanan	Shift – Panah kanan	
Menambah blok ke akhir baris	Shift – End	
Menambah blok ke baris di atasnya	Shift – Panah atas	
Menambah blok ke baris di bawahnya	Shift – Panah bawah	
Menambah blok satu kata di kiri	Ctrl – Shift – Panah kiri	
Menambah blok satu kata di kanan	Ctrl – Shift – Panah kanan	
Menambah blok satu halaman ke atas	Shift – Page Up	
Menambah blok satu halaman ke bawah	Shift – Page Down	
Menambah blok hingga ke awal program	Ctrl – Shift – Home	
Menambah blok hingga ke akhir program	Ctrl – Shift – End	

**Tabel 8** Shortcut lain-lain

Perintah	Shortcut	Alternatif
Simpan file	F2	Ctrl – K S
Buka file	F3	
Cari	Ctrl – Q F	
Cari lagi	Ctrl – L	
Cari dan ganti	Ctrl – Q A	
Menandai	Ctrl – K [0-9]	
Menuju ke tanda	Ctrl – Q [0-9]	
Undo	Alt – Backspace	

---

---

# Bab 3

## BAHASA PASCAL

---

---

Sebelum membahas dengan detail tentang bahasa Pascal (dalam hal ini adalah compiler Free Pascal), perlu diketahui terlebih dulu struktur program dalam bahasa Pascal. Bentuk ini harus dipenuhi agar program dapat dikompilasi oleh compiler.

### Struktur Bahasa Pascal

Bentuk umum bahasa Pascal adalah sebagai berikut:

```
PROGRAM          nama(file1, file2, file3);
CONST            deklarasi konstanta;
VAR              deklarasi variabel;
TYPE             deklarasi type;
LABEL           deklarasi label;
FUNCTION         deklarasi fungsi;
PROCEDURE        deklarasi prosedur;

BEGIN
    statement1;
    statement2;
    statement3;
    ...
END.
```

Program Pascal terdiri dari 3 bagian pokok, yaitu:

#### 1. Nama Program

Nama program adalah hanya sekedar menuliskan judul dari program, tidak mempunyai arti apa-apa dalam proses kompilasi. Judul program dapat diikuti oleh file-file data yang berhubungan dengan program tersebut.

Pada Turbo Pascal, dapat ditambahkan klausa uses yang menunjukkan bahwa program menggunakan unit.

#### 2. Deklarasi

Bagian ini berisi deklarasi pengenalan maupun data yang dipergunakan di dalam program. Walaupun tampaknya membuang-buang waktu dan tidak berguna, namun sesungguhnya merupakan bagian



terpenting dari rangka penyusunan sebuah program yang terstruktur. Struktur program sangat penting dalam pembuatan program yang panjang, karena bagian ini akan mengingatkan programmer tentang variabel, tipe data, konstanta, fungsi, prosedur yang digunakan dalam program. Selain itu, orang lain yang membaca program akan lebih dapat mengerti jalannya program dengan deklarasi ini.

### 3. Program utama

Program utama berisi statement. Tentang statement akan dibahas pada bab selanjutnya.

### Deklarasi program

Bagian deklarasi ada 6 macam, yaitu:

#### 1. Deklarasi CONST

Deklarasi ini gunanya untuk mendeklarasikan nama konstanta tertentu. Nama konstanta adalah merupakan suatu pengenalan (*identifier*) yang nilainya tidak dapat berubah dalam program. Contoh:

```
CONST
  Pi = 3.14
  Ti ti koma = ' ; '
```

#### 2. Deklarasi VAR

Deklarasi ini gunanya adalah untuk menyatakan variabel yang digunakan dalam program. Variabel adalah suatu pengenalan (*identifier*) yang nilainya dapat berubah. Contoh:

```
VAR
  Data: array[1..100] of byte;
  Umur: 0..100;
```

#### 3. Deklarasi TYPE

Deklarasi type dipergunakan untuk menyusun suatu bentuk tipe data yang baru sebagai hasil penggabungan dari tipe-tipe yang sudah ada. Contoh:

```
TYPE
  Data = array[1..100] of byte;
  Hari = (Senin, Selasa, Rabu, Kamis, Jumat, Sabtu);
VAR
  Nilai: Data;
  Hari Kerja: Senin..Jumat;
```

#### 4. Deklarasi LABEL

Deklarasi label menjelaskan adanya label atau tujuan yang bisa melompati jalannya program dengan statement goto.

## 5. Deklarasi FUNCTION

Function adalah bagian dari program yang melakukan tugas tertentu dan menghasilkan suatu nilai.

Sintaks penulisan:

```
function namafunction: tipehasil;
                                Atau
function namafunction(daftarparameter): tipehasil;
```

Contoh:

```
function UpCaseStr(S: string): string;
var
  I: Integer;
begin
  for I := 1 to Length(S) do
    if (S[I] >= 'a') and (S[I] <= 'z') then
      Dec(S[I], 32);
  UpCaseStr := S;
end;
begin
  writeln(UpCaseStr('this is text'));
end.
```

## 6. Deklarasi PROCEDURE

Procedure adalah bagian dari program yang melakukan aksi tertentu, seringkali aksi tersebut dilakukan berdasarkan parameter.

Sintaks penulisan:

```
procedure namafunction;
                                Atau
procedure namafunction(daftarparameter);
```

Contoh:

```
procedure WrStr(X, Y: integer; S: string);
var
  SaveX, SaveY: Integer;
begin
  SaveX := WhereX;
  SaveY := WhereY;
  GotoXY(X, Y);
  Write(S);
  GotoXY(SaveX, SaveY);
end;

begin
  WrStr(10, 20, 'This is text');
end.
```

**Aturan penulisan program Pascal**

Suatu bahasa pemrograman selalu mempunyai aturan penulisan program. Hal ini menunjukkan konsistensi kompiler dalam melakukan proses kompilasi. Aturan pada program Pascal adalah sebagai berikut:

Program pascal dapat ditulis pada kolom berapa saja dan diakhiri pada kolom berapa saja.  
Antar statement / perintah dipisahkan dengan tanda ; (titik koma)

Akhir dari sebuah program Pascal ditandai dengan tanda . (titik) setelah perintah END. Semua statement / perintah setelah END. tidak akan dianggap sebagai perintah.

Spasi antar pengenalan (identifikasi) diabaikan.

Baris komentar diletakkan di antara tanda (\* dan \*) atau { dan }. Baris komentar tidak akan dieksekusi oleh komputer. Baris komentar biasanya dipergunakan untuk memberikan penjelasan-penjelasan guna memperjelas pengertian variabel atau tipe atau perintah dalam sebuah program.

**Simbol (Symbols)**

Pascal mengenal simbol-simbol yang dapat digunakan dalam program, yaitu:

huruf	: A..Z , a..z
digit	: 0..9
digit heksadesimal	: 0..9, A..F, a..f
Karakter khusus	: + - * / = < > [ ] . , ( ) : ^ @ { } \$ #
Pasangan karakter	: <= >= := (* *)

Selain dari karakter-karakter di atas, adalah karakter yang tidak dikenali oleh bahasa Pascal.

**Kata tercadang (Reserved Words)**

Kata tercadang adalah bagian dari bahasa Pascal dan tidak dapat dipakai untuk kegunaan lain dalam program (tidak dapat didefinisikan ulang). Kata tercadang ini tidaklah case sensitive, artinya for akan sama dengan FOR.

Berikut adalah perbandingan kata tercadang dalam Turbo Pascal dan free pascal:

**Tabel 9.** Perbandingan kata tercadang Free Pascal dan Turbo Pascal

	<b>Free Pascal</b>	<b>Turbo Pascal</b>
A	absolute, and, array, asm	absolute, and, array, asm
B	begin, break	begin, break
C	case, const, constructor, continue	case, const, constructor, continue
D	destructor, div, do, downto, <b>dispose</b>	destructor, div, do, downto
E	else, end, <b>exit</b>	else, end
F	file, for, function, <b>false</b>	file, for, function
G	goto	Goto
I	if, implementation, in, inherited, inline, interface	if, implementation, in, inherited, inline, interface
L	label	label
M	mod	mod
N	nil, not, <b>new</b>	nil, not
O	object, of, on, operator, or	object, of, on, operator, or
P	packed, procedure, program	packed, procedure, program
R	record, repeat	record, repeat
S	self, set, shl, shr, string	self, set, shl, shr, string
T	then, to, type, <b>true</b>	then, to, type
U	unit, until, uses	unit, until, uses
V	var, while, with	var, while, with
X	xor	xor

**Pengenal (Identifier)**

Identifier adalah nama yang diberikan untuk elemen-elemen dalam Pascal. Misalnya nama prosedur, nama tipe, nama fungsi, nama variabel dan nama label. Identifier ini harus dideklarasikan terlebih dulu agar dapat dikenali.

Syarat-syarat penamaan sebuah identifier adalah:

- ? Dapat sepanjang apapun, namun Turbo Pascal akan mengambil 63 karakter pertama dari nama identifier
- ? HARUS diawali dengan huruf atau underscore (\_)
- ? Karakter ke dua dan selanjutnya dapat berupa huruf, angka, atau underscore
- ? Tidak boleh ada 2 identifier yang sama dalam satu program

- ? Tidak boleh berupa *reserved word*. *Reserved Word* adalah kata yang telah dikenal oleh Pascal yang telah mempunyai kegunaan tertentu.

Contoh penulisan identifier yang benar:

```
coba1
jari_jari
programcoba_coba
integer (mengapa identifier ini diperbolehkan?)
```

Contoh penulisan identifier yang salah:

```
coba 1 (mengandung spasi)
jari-jari (mengandung karakter -)
2b (diawali dengan angka)
to (mengapa tidak diperbolehkan?)
```

Contoh penggunaan identifier:

```
program coba1;
var
  a, b: byte;
type
  c=word;
begin
end.
```

Pada program di atas, terdapat 4 buah identifier, yaitu coba1, a, b dan c. Coba1 digunakan sebagai identifier nama program, a dan b digunakan sebagai identifier dari deklarasi var dan c sebagai identifier nama type.

### Angka (Numbers)

Ada 2 macam penulisan angka dalam bilangan bulat yang dikenal dalam Pascal.

- ? Normal, dalam format desimal (basis 10).
- ? Heksadesimal (basis 16), penulisan angka dalam format heksadesimal harus didahului oleh tanda \$. Misalnya \$FF berarti 255 desimal.

Free Pascal menambahkan 2 format penulisan lagi, yaitu:

- ? Oktal (basis 8), penulisan angka dalam format oktal harus didahului oleh tanda &. Misalnya 15 desimal mempunyai nilai yang sama dengan &17.
- ? Biner (basis 2), penulisan angka dalam format biner harus didahului oleh tanda %. Misalnya 15 desimal mempunyai nilai yang sama dengan %1111.

**Tipe (Type)**

Tipe data yang dikenal oleh Pascal dapat diklasifikasikan menjadi: tipe sederhana, tipe string, tipe terstruktur dan tipe data pointer. Ada tipe yang sudah didefinisikan (*built-in*) sehingga compiler langsung mengenalinya tanpa perlu dideklarasikan. Tapi ada tipe yang harus dideklarasikan terlebih dulu (*user-defined types*).

✍ Berikut ini adalah taksonomi dari tipe data yang ada di **Turbo Pascal**

✍ Sederhana

✍ Ordinal

✍ Integer / bilangan bulat (dibagi lagi menjadi : Byte, Shortint, Word, Integer, Longint)

✍ Character

✍ Boolean

✍ Enumerated

✍ Subrange

✍ Real

✍ String

✍ Terstruktur

✍ Set

✍ Array

✍ Record

✍ File

✍ Pointer

✍ **Free Pascal** melakukan beberapa perubahan untuk taksonomi tipe data ini, yaitu:

✍ Tipe dasar / tipe sederhana (*Based Types*)

✍ Tipe ordinal (*ordinal types*)

✍ Tipe real (*real types*)

✍ Tipe karakter (*Character Types*)

✍ Karakter (*Char*)

✍ Untai (*Strings*)

✍ Untai pendek (*Short strings*)

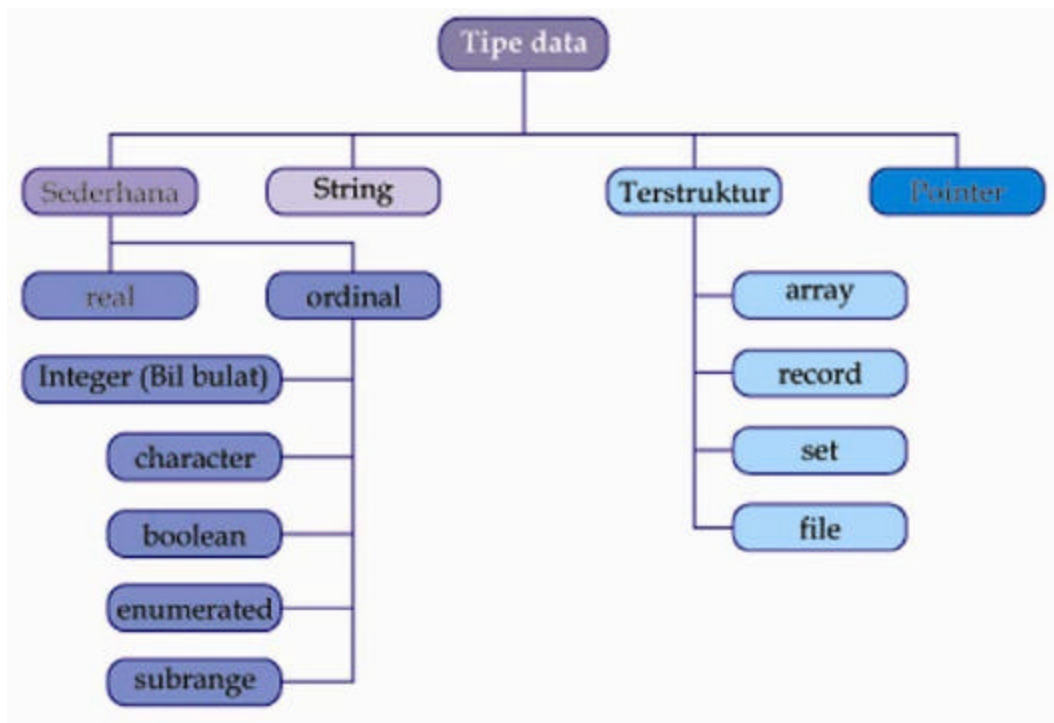
✍ Ansistrings

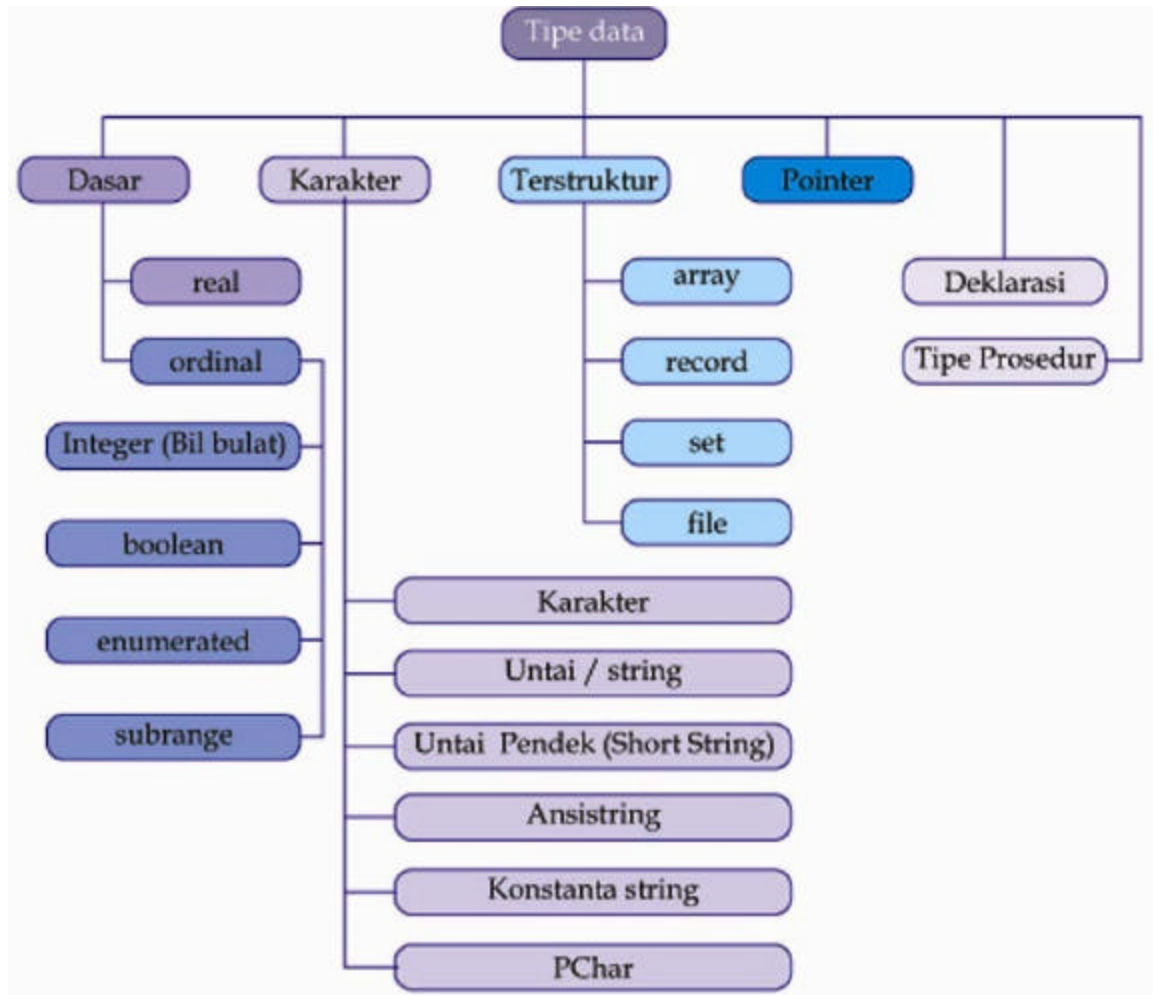
✍ Konstanta string (*Constant string*)

✍ Pchar – Null terminated string

- ✍ Tipe terstruktur (*Structured types*)
  - ✍ Array
  - ✍ Record
  - ✍ Set
  - ✍ File
- ✍ Pointer
- ✍ Deklarasi Forward
- ✍ Tipe Prosedur

**Diagram 1.** Tipe data di Turbo Pascal



**Diagram 2.** Tipe data di Free Pascal



## Penjelasan Tipe data pada Free Pascal

Tipe Dasar / Sederhana (Based Types or Simple Types)



Tipe dasar mendefinisikan suatu nilai yang berurutan. Ada 2 kelas dari tipe sederhana ini yaitu:

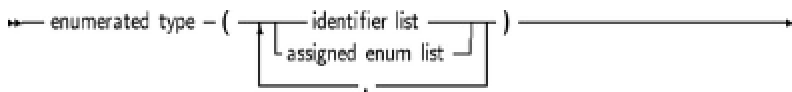
### Tipe Ordinal (*Ordinal type*)

Pada tipe ordinal, Free Pascal telah mendefinisikan 14 predefined type (apa yang dimaksud predefined type?), yaitu : bilangan bulat, boolean, char. (dimana 11 buah tipe data yang lain?). Artinya pembuat program tidak perlu mendefinisikan tipe untuk tipe-tipe data di atas. Sedangkan untuk user defined type pada tipe ordinal ada 2 macam, yaitu : tipe enumerasi (enumerated types) dan tipe subjangkauan (subrange types).

Tipe Ordinal dapat dioperasikan dengan fungsi/prosedur berikut:

- ? ORD (fungsi)
- ? PRED (fungsi)
- ? SUCC (fungsi)
- ? HIGH (fungsi)
- ? LOW (fungsi)
- ? INC (prosedur)
- ? DEC (prosedur)

### Tipe Enumerasi (*Enumerated*)



Identifier list mempunyai format penulisan:



Tipe Enumerasi mendefinisikan nilai yang berurutan ke suatu elemen dalam suatu daftar identifier. Elemen pertama mempunyai nilai 0, yang kedua mempunyai nilai 1 dan seterusnya.

Format penulisan:

```
type
  identifier=(identifier_1, identifier_2, ..., identifier_n)
```

*(Berdasarkan definisi tipe enumerasi, mana yang disebut elemen pada format penulisan di atas? Mana yang disebut nilai yang berurutan? Mana yang disebut daftar identifier? Dan di mana letak user defined type dari tipe enumerasi ini)*

Contoh tipe enumerasi yang benar:

```
type
  kartu=(club, heart, di amond, spade);
  musik=(j azz, blues, country, pop);
```

Untuk mengetahui urutan suatu nilai dari tipe enumerasi ini dengan menggunakan fungsi ord, untuk mengetahui nilai sebelumnya menggunakan fungsi pred dan untuk mengetahui nilai sesudahnya digunakan fungsi succ.

Contoh penggunaan tipe enumerasi dalam program:

```
var
  model : (small, medium, large);
begin
  model := medium;
  writeln(ord(model)); {berapa urutan dari model?}
  model := pred(model); {apa nilai dari model?}
  model := succ(model); {apa nilai dari model?}
end.
```

Pada Free Pascal tipe enumerasi dapat dibuat dengan model bahasa C, yaitu:

```
Type
  EnumType = (satu, dua, tiga, empatpuluh : = 40, empatsatu);
```

Hasilnya, ordinal dari forty adalah 40, bukan 3 seperti pada tipe enumerasi biasa dan ordinal dari fortyone adalah 41. Namun harap diperhatikan bahwa fungsi Pred dan Succ tidak dapat digunakan untuk tipe enumerasi seperti di atas.

Compiler akan menganggap salah untuk tipe enumerasi berikut:

```
Type
  EnumType = (satu, dua, tiga, empatpuluh : = 40, ti gapul uh=30);
```

Contoh penggunaan tipe enumerasi yang benar:

```
- var MyCard: (Club, Diamond, Heart, Spade);
- var Card1, Card2: (Club, Diamond, Heart, Spade);
- type Suit = (Club, Diamond, Heart, Spade);
- var
    Card1: Suit;
    Card2: Suit;
```

Contoh penggunaan tipe enumerasi yang salah:

```
var Card1: (Club, Diamond, Heart, Spade);
var Card2: (Club, Diamond, Heart, Spade);
```

(dimana letak kesalahannya?)

### Tipe Subjangkauan (*Subrange*)

subrange type = constant .. constant

Tipe subjangkauan mendefinisikan suatu elemen dari nilai terkecil sampai nilai terbesar.

Format penulisan

```
type
    identifier = nilai 1 .. nilai 2;
```

nilai1 dan nilai2 haruslah merupakan merupakan nilai dari tipe ordinal yang sama. Pada suatu tipe subjangkauan, juga dapat digunakan fungsi-fungsi ord, pred dan succ. Fungsi-fungsi tersebut masih tetap mengacu pada tipe enumerasi asalnya.

Contoh dalam program:

```
var
    nilai: 'a'..'e';
begin
    nilai := 'a';
    writeln(ord(nilai)); {berapa yang dihasilkan, mengapa?}
    writeln(pred(nilai)); {mengapa diperbolehkan?}
end.
```

**Tipe Boolean (*Boolean*)**

Tipe boolean adalah sebuah tipe yang hanya dapat bernilai false atau true. Dalam Free Pascal mendukung tipe ByteBool, WordBool dan LongBool.

Karena masih merupakan tipe ordinal, maka nilainya dapat diketahui urutannya dengan menggunakan fungsi ord.

```
ORD(FALSE) = 0  
ORD(TRUE)  = 1
```

Operasi-operasi yang dapat dilakukan pada tipe boolean adalah (diurutkan berdasarkan hirarki teratas sampai terendah): NOT, AND, OR

Contoh dalam program:

```
Var  
  X: boolean;  
  A: byte;  
Begin  
  A:=0;  
  If a>100 then x:=true else x:=true;  
End.  
  
var  
  Ok: boolean;  
  a, b: byte;  
Begin  
  a:=10;  
  b:=20;  
  Ok:=a=b;  
End.
```

Pada Free Pascal, ekspresi boolean akan dievaluasi sedemikian sehingga pada saat hasilnya telah diketahui maka ekspresi selanjutnya tidak akan dievaluasi. Contoh berikut akan menyebabkan fungsi FUNC (fungsi yang menghasilkan nilai boolean) tidak dijalankan.

```
...  
b:=false;  
a:=b and FUNC;  
...
```

Karena b bernilai false, maka a pasti akan bernilai false, sehingga FUNC tidak pernah dievaluasi.

**Tipe Bilangan Bulat (*Integer*)**

Ada 10 tipe bilangan bulat, yaitu:

**Tabel 10.** Tipe bilangan bulat di Free Pascal

Tipe	Range	Ukuran (byte)
Byte	0 .. 255	1
Shortint	-128 .. 127	1
Smallint	-32768 .. 32767	2
Word	0 .. 65535	2
Integer	Masuk dalam smallint, longint dan Int64	2, 4, 8
Cardinal	Masuk dalam word, longword, dan qword	2, 4, 8
Longint	-2147483648 .. 2147483647	4
Longword	0 .. 4294967295	4
Int64	-9223372036854775808 .. 9223372036854775807	8
Qword	0 .. 18446744073709551615	8

Tipe Integer secara standard akan diarahkan pada tipe smallint, sedangkan tipe cardinal selalu diarahkan pada tipe longword.

Tipe bilangan bulat ini dapat disebut sebagai predefined type, atau dapat juga dimasukkan dalam tipe subjangkauan.

Operasi-operasi yang dapat dilakukan pada tipe bilangan bulat adalah (diurutkan berdasarkan hirarki teratas sampai terendah):

- ? @, NOT
- ? \*, /, div, mod, and, shl, shr
- ? +, -, or, xor
- ? =, <>, <, >, <=, >=, in

**Tabel 11.** Daftar operator pada bilangan bulat

Operator	Proses	Tipe operand	Tipe hasil
+	Penjumlahan	Bilangan bulat	Bilangan bulat
-	Pengurangan	Bilangan bulat	Bilangan bulat
*	Perkalian	Bilangan bulat	Bilangan bulat
/	Pembagian	Bilangan bulat	Real
DIV	Pembagian integer	Bilangan bulat	Bilangan bulat
MOD	Sisa Bagi	Bilangan bulat	Bilangan bulat

Evaluasilah ekspresi di bawah ini, tentukan apakah ekspresi tersebut benar atau salah, jika benar berapa nilai akhirnya, jika salah tentukan alasannya!

`3 + 5 * 2`

`6 or 2 and 5 div 2`

`4 mod not 6`

`5 >= 6`

`10 and 3 = 0`

`(10 and 5 = 0) or (4 * 1 = 4)`

```
var
  x: boolean;
  a, b: byte;

begin
  a:=10;
  x:=true;
  if (a>=0) and x then write('a');
end.
```

```
begin
  a:=4;
  b:=a and 4;
  x:=true;
  if b and x then write('a');
end.
begin
  if b=4 and a=1 then write('a');
end.
```

### Tipe Real

Sebuah tipe real adalah anggota bilangan real, yang dinyatakan dalam notasi floating point. Ada 6 model tipe real, masing-masing mempunyai jangkauan, ketepatan angka dan ukuran.

Type	Jangkauan	Digit	Ukuran
Real	2.9.10-39 .. 1.7.1038	11-12	6 byte
Single	1.5.10-45 .. 3.4.1038	7-8	4 byte
Double	5.0.10-324 .. 1.7.10308	15-16	8 byte
Extended	3.4.10-4932 .. 1.1.104932	19-20	10 byte
Comp	-2.1063+1 .. 2.1063-1	19-20	8 byte

Ada 4 macam operator pada tipe real, yaitu:

**Tabel 12** Tabel operator pada tipe real

Operator	Proses	Tipe operand	Tipe hasil
+	Penjumlahan	Real	Real
-	Pengurangan	Real	Real
*	Perkalian	Real	Real
/	Pembagian	Real	Real

### Tipe Karakter (Character)

Merupakan suatu tipe data untuk menyimpan karakter ASCII. Ditulis dengan menggunakan tanda petik atau diawali dengan #. Karena masih masuk dalam tipe ordinal maka dapat dioperasikan dengan menggunakan fungsi ord (untuk mengetahui urutannya), fungsi succ, fungsi pred, prosedur inc, prosedur dec.

Tipe ini juga dapat menerima fungsi CHR (mengubah suatu bilangan bulat menjadi karakter yang sesuai dengan ASCII)

Contoh:

```

Var
  A: char;
Begin
  A:='A';
  A:=CHR(65)
  A:=#1;
End.
```

### Tipe Untai (*String*)



Tipe string adalah tipe yang diisi rangkaian karakter.

Format penulisan:

```
type
  identifier = string[panjang];
```

Bila panjang tidak didefinisikan, maka Pascal menganggap panjangnya adalah 255 karakter (panjang maksimum).

Contoh penggunaan dalam program

```
type
  str80 = string[80]; {bisa dideklarasikan dalam tipe terlebih dulu}
var
  a: str80;
  b: string[10]; {atau langsung dalam variabel}

const
  Heading: string[7] = 'Section';
  NewLine: string[2] = #13#10;
  TrueStr: string[5] = 'Yes';
  FalseStr: string[5] = 'No';
```

Fungsi standard dari Length adalah untuk mencari panjang dari string. Perbandingan antar string didefinisikan dengan melihat urutan karakter di dalamnya. Misalnya “AB” lebih besar dari “A”; sehingga ‘AB’ > ‘A’ menghasilkan TRUE. String kosong (panjangnya 0) adalah string yang mempunyai urutan terendah.

String dapat dicari urutannya seperti halnya Array. Jika S adalah sebuah variabel string dan I adalah sebuah ekspresi numerik yang menghasilkan bilangan bulat, maka S[I] akan menghasilkan karakter dari S yang ke i. Statement MyString[2] := ‘A’ akan mengisi karakter ke-2 dari MyString dengan karakter A.

Contoh berikut adalah penggunaan fungsi standard UpCase untuk mengubah sebuah string menjadi huruf besar.



```
var I: Integer;
begin
  I := Length(MyString);
  while I > 0 do
    begin
      MyString[I] := UpCase(MyString[I]);
      I := I - 1;
    end;
  end;
```

### **Tipe Untai Pendek (*Short String*)**

Deklarasi String akan mendeklarasikan sebagai shortstring apabila:

- ? Opsi dimatikan {SH-}, maka deklarasi string akan selalu shortstring
- ? Opsi dinyalakan {SH+} dan diberikan panjang tertentu, maka deklarasi tersebut adalah shortstring.

Predefined type untuk shortstring adalah:

```
ShortString = String[255];
```

### **Tipe Ansistring**

Format penulisan:

```
type
  identifier=ansi string;
```

Tipe ansistring adalah tipe string dengan panjang tanpa batas. Dalam proses kompilasi, ansistring diperlakukan sebagai pointer. Jika string adalah kosong (''), maka pointer akan menunjuk pada NIL. Jika ada isinya, pointer akan menunjuk suatu alamat pada heap memory.

Ketika tipe ansistring dideklarasikan, Free Pascal hanya mengalokasikan alamat untuk pointernya saja. Pointer ini pasti berisi NIL, sehingga isi mula-mula dari variabel ansistring pasti kosong (tidak mempunyai nilai apa-apa)

### **Tipe PChar**

Format penulisan:

```
type
  identifier=PChar;
```

Pchar adalah variabel berjenis pointer yang mengarah ke sebuah array dengan tipe Char, yang diakhiri dengan karakter 0 (#0). Dengan tipe Pchar, dimungkinkan adanya operasi tambahan pada tipe ini.

Perhatikan cara pemakaian PChar pada program di bawah ini:

```
program one;
var p : PChar;
begin
  P := 'This is a null-terminated string. ';
  WriteLn (P);
end.
```

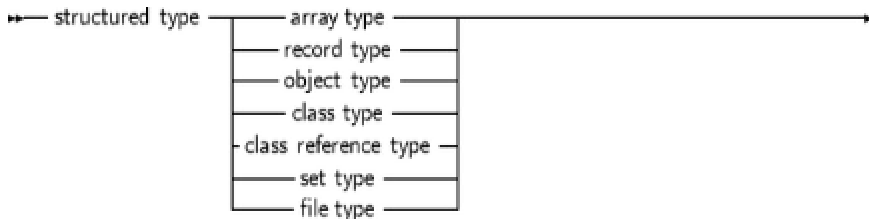
Akan sama dengan:

```
program two;
const P : PChar = 'This is a null-terminated string. '
begin
  WriteLn (P);
end.
```

Mungkin juga nilai sebuah string diberikan pada Pchar, caranya:

```
Program three;
Var S : String[30];
    P : PChar;
begin
  S := 'This is a null-terminated string. ' #0;
  P := @S[1];
  WriteLn (P);
end.
```

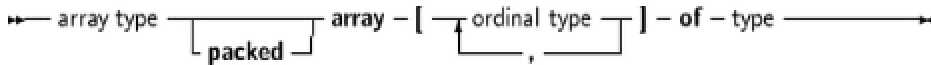
### Tipe Terstruktur (*Structured Type*)



Tipe terstruktur dalam pascal adalah tipe yang dapat menampung lebih dari 1 nilai.

Free Pascal mendefinisikan beberapa tipe terstruktur ini, yaitu:

## Tipe Array



Tipe array adalah suatu tipe yang dapat menampung beberapa nilai dengan tipe yang sama dalam bentuk satu dimensi atau multidimensi. Tiap nilai dalam array dapat diacu dengan nama array dan indeks yang diletakkan dalam kurung.

Format penulisan:

```

type
  identifier=array[tipe_indeks] of tipe_data

```

Tipe\_indeks adalah suatu tipe data ordinal (apa itu tipe data ordinal? Terdiri dari apa saja tipe data ordinal?)  
 Tipe\_data mendefinisikan bahwa array tersebut untuk menampung nilai dengan tipe data apa saja.

Contoh dalam program:

```

type
  a=array[boolean] of byte;
  c=5..10;
  b=array[1..10,c] of real; {termasuk tipe data apa c?}
var
  nilai_a: a;
  nilai_b: b;
begin
  nilai_a[true] := 10;
  nilai_b[1, 5] := 10.4;
end.

```

Dalam contoh di atas, tipe\_indeks apa yang digunakan? Berapa byte-kah yang digunakan program di atas untuk mendeklarasikan variabel dalam tipe tersebut?

## Tipe Himpunan (Set)



Himpunan adalah kumpulan dari beberapa nilai yang bertipe ordinal. Dalam sebuah himpunan, nilai-nilai di dalamnya akan otomatis terurut dari elemen yang paling kecil hingga yang paling besar. Dan tidak ada 2 nilai yang sama dalam sebuah himpunan.

Format penulisan:

```
type
  identifier = set of ordinal_type;
```

Contoh dalam program:

```
type TIntSet = set of 1..250;
var Set1, Set2: TIntSet;
begin
  Set1 := [1, 3, 5, 7, 9];
  Set2 := [2, 4, 6, 8, 10]
End.
```

Operasi-operasi yang dapat diberlakukan pada tipe data set:

Operation	Operand	Result type	Example
union	set	set	Set1 + Set2
difference	set	set	S - T
intersection	set	set	S * T
subset	set	Boolean	Q <= MySet
superset	set	Boolean	S1 >= S2
equality	set	Boolean	S2 = MySet
inequality	set	Boolean	MySet <> S1
membership	ordinal, set	Boolean	A in Set1

Evaluasilah ekspresi di bawah ini. Tentukan nilai akhir untuk ekspresi tersebut.

```
Var
  A, B: set of char
Begin
  A := [ 'A' .. 'E' ];
  B := [ 'D' .. 'H' ];
End.
```

- ? A + B
- ? A - B
- ? B - A
- ? A \* B
- ? 'A' in B
- ? A in B
- ? 'A' <= B

## Type Record



Untuk Field list, diagram penulisannya:



Untuk fixed fields diagram penulisannya:



Sebuah record dapat terdiri dari sejumlah komponen (atau field), yang masing-masing field dapat mempunyai tipe yang berbeda-beda.

Format penulisan:

```
Type identifier = record
  fieldList1: type1;
  ...
  fieldListn: typen;
end;
```

Contoh penulisan sebuah record:

```
Type
  Point = Record
    X, Y, Z : Real;
  end;
  RPoint = Record
    Case Boolean of
      False : (X, Y, Z : Real);
      True : (R, theta, phi : Real);
    end;
  BetterRPoint = Record
    Case UsePolar : Boolean of
      False : (X, Y, Z : Real);
      True : (R, theta, phi : Real);
    end;
  TDateRec = record
    Year: Integer;
    Month: (Jan, Feb, Mar, Apr, May, Jun,
            Jul, Aug, Sep, Oct, Nov, Dec);
    Day: 1..31;
  end;
```

```
var Record1, Record2: TDateRec;
```

Deklarasi variabel di atas (Record1 dan Record2) mempunyai tipe data yang sama. Untuk mengakses nilai untuk tiap field dalam sebuah record, menggunakan format:

```
Namarecord. namafield
```

Contoh:

```
Record1. Year := 1904;
Record1. Mnth := Jun;
Record1. Day := 16;
```

Atau dapat juga dengan menggunakan statement With.

```
with Record1 do
begin
  Year := 1904;
  Mnth := Jun;
  Day := 16;
end;
```

Untuk tipe data record yang sama, dapat berlaku statement

```
Record2 := Record1;
```

## Tipe File



File adalah sekumpulan elemen (data).

Sintaks penulisan untuk tipe data file:

```
type fileType = file of type
```

di mana fileType = sembarang nama identifier dan tipe adalah fixed-size type. Contoh:

```
type
  PhoneEntry = record
    FirstName, LastName: string[20];
    PhoneNumber: string[15];
    Listed: Boolean;
  end;
  PhoneList = file of PhoneEntry;
Var
  Filephone: PhoneList;
```

Deklarasi di atas, mendeklarasikan file bertipe record. Deklarasi di atas, dapat juga secara langsung dituliskan:

```
var List1: file of PhoneEntry;
```

Tipe data file dapat juga berupa file tidak bertipe. Format penulisan:

```
Var  
MyFile : File;
```

## Type Pointer

⇨ pointer type = ^ – type identifier ⇨

Pointer adalah sebuah variabel yang berisi alamat memori dan bukan berisi data numerik atau string seperti halnya variabel biasa. Format penulisan:

```
type pointerTypeName = ^type
```

Contoh dalam program:

```
var  
  X, Y: Integer;    // X and Y are Integer variables  
  P: ^Integer;      // P points to an Integer  
begin  
  X := 17;          // assign a value to X  
  P := @X;          // assign the address of X to P  
  Y := P^;          // dereference P; assign the result to Y  
end;
```

## Type prosedural (Procedural Types)

⇨ procedural type — function header ———— ⇨  
                          └── procedure header ─┘ └── of – object ─┘ └── ; – call modifiers ─┘

Tipe prosedural pada Free Pascal berbeda dengan Turbo Pascal walaupun secara konsep sama. Berikut adalah contoh pemakaian dalam program:

```
Type TOneArg = Procedure (Var X : integer);  
      TNoArg = Function : Real;  
var proc : TOneArg;  
    func : TNoArg;
```

**Pengubah (Variable)**

Variabel adalah penampung nilai yang nilai di dalamnya dapat berubah pada saat runtime. Variabel harus mempunyai tipe data tertentu, sehingga compiler dapat memperlakukan data yang ditampung dengan semestinya. Deklarasi untuk variabel terdiri dari 2 bagian, yaitu nama variabel dan tipe datanya, yang dipisahkan dengan semicolon (:).

Cara mendeklarasikan variabel adalah:

```
Var Identifierlist : type;
```

Identifierlist merupakan kumpulan identifier yang dipisahkan dengan tanda koma dan type adalah sembarang tipe data.

Contoh pendeklarasian variabel yang benar:

```
Var  
  a: byte;
```

```
Var a, b: byte;
```

```
Var  
  X, Y, Z: Double;  
  I, J, K: Integer;  
  Digit: 0..9;  
  Okay: Boolean;
```

Contoh pendeklarasian variabel yang salah:

```
Var  
  a, b;
```

```
var  
  a: 8;
```

**Konstanta (Constant)**

Konstanta berisi suatu nilai yang tidak dapat dirubah. Pemberian nilai untuk konstanta ini pada saat konstanta dideklarasikan.

Contoh deklarasi konstanta



```
const
  Pi = 3.14159;
  Answer = 342;
  ProductName = 'Delphi';
  MaxData = 1024 * 64 - 16;
  NumChars = Ord('Z') - Ord('A') + 1;
  Message = 'Hello world...';
```

Konstanta seperti di atas disebut sebagai konstanta tak bertipe. Sedangkan konstanta yang bertipe mempunyai sifat seperti variabel. Untuk konstanta bertipe deklarasinya:

```
const identifier: type = value
```

### Mendeklarasikan konstanta bertipe untuk ARRAY:

```
const Digits: array[0..9] of Char = ('0', '1', '2', '3', '4', '5', '6', '7', '8', '9');
Factorial: array[1..7] of Integer = (1, 2, 6, 24, 120, 720, 5040);
```

### Mendeklarasikan konstanta bertipe untuk RECORD

```
type
  Point = record
    X, Y: Real;
  end;
  Vector = array[0..1] of Point;
  Mnth = (Jan, Feb, Mar, Apr, May, Jun, Jly, Aug, Sep, Oct, Nov, Dec);
  Date = record
    D: 1..31;
    M: Mnth;
    Y: 1900..1999;
  end;
const
  Origin: Point = (X: 0.0; Y: 0.0);
  Line: Vector = ((X: -3.1; Y: 1.5), (X: 5.8; Y: 3.0));
  SomeDay: Date = (D: 2; M: Dec; Y: 1960);
```

### Mendeklarasikan konstanta bertipe untuk SET

```
type
  Digits = set of 0..9;
  Letters = set of 'A'..'Z';
const
  EvenDigits: Digits = [0, 2, 4, 6, 8];
  Vowels: Letters = ['A', 'E', 'I', 'O', 'U', 'Y'];
  HexDigits: set of '0'..'z' = ['0'..'9', 'A'..'F', 'a'..'f'];
```

**Mendeklarasikan konstanta bertipe untuk tipe data sederhana**

```
const
  Maximum: Integer = 9999;
  Factor: Real = -0.1;
  Breakchar: Char = #3;
```

**Mendeklarasikan konstanta bertipe untuk tipe data string**

```
const
  Heading: string[7] = 'Section';
  NewLine: string[2] = #13#10;
  TrueStr: string[5] = 'Yes';
  FalseStr: string[5] = 'No';
```

---

# Bab 4

## OPERASI DI PASCAL

---

### Operasi Aritmatika

Operasi Aritmatika adalah operasi untuk melaksanakan suatu proses perhitungan. Hasil dari operasi ini diberikan pada variabel yang dapat menampung nilai numerik.

Seperti yang telah ditulis pada pembahasan tipe bilangan bulat dan real, operator pada operasi aritmatika adalah:

**Tabel 13** Tabel operator untuk operasi aritmatika

OPERATOR	OPERASI	TIPE OPERAN	TIPE HASIL
+	Penjumlahan	Tipe bilangan bulat, tipe real	Tipe bilangan bulat, tipe real
-	Pengurangan	Tipe bilangan bulat, tipe real	Tipe bilangan bulat, tipe real
*	Perkalian	Tipe bilangan bulat, tipe real	Tipe bilangan bulat, tipe real
/	Pembagian	Tipe bilangan bulat, tipe real	Tipe real
Div	Pembagian bilangan bulat	Tipe bilangan bulat	Tipe Bilangan bulat
Mod	Sisa bagi	Tipe bilangan bulat	Tipe Bilangan bulat

Operator +, - dan \* juga dipergunakan untuk operasi himpunan (set)

Yang perlu diperhatikan dalam operator aritmatika adalah:

? Tipe bilangan bulat dapat diberikan kepada tipe real, tapi tidak berlaku sebaliknya. Contoh:

```
Var
  A: real;
  L: Longint;
begin
  A := L;           {statement yang benar}
  L := A;           {statement yang salah}
End.
```

? Tidak boleh ada dua atau lebih operator yang berurutan dalam sebuah ekspresi aritmatika. Contoh:

```
Var
  A: real;
  L: Longint;
Begin
  A := A + 1;           {statement yang benar}
  A := L++;             {statement yang salah}
End.
```

**Operasi Logika**

Operasi logika menggunakan sistem bilangan biner untuk perhitungannya. Ada 6 buah operator logika yang dikenal Pascal, yaitu:

**Tabel 14.** Tabel operator logika

OPERATOR	OPERASI	TIPE OPERAN	TIPE HASIL
NOT	Ingkaran bit	Tipe bilangan bulat	Tipe bilangan bulat
AND	Logika AND bit	Tipe bilangan bulat	Tipe bilangan bulat
OR	Logika OR bit	Tipe bilangan bulat	Tipe bilangan bulat
XOR	Logika XOR bit	Tipe bilangan bulat	Tipe bilangan bulat
SHL	Geser bit ke kiri	Tipe bilangan bulat	Tipe Bilangan bul at
SHR	Geser bit ke kanan	Tipe bilangan bulat	Tipe Bilangan bulat

**Operasi Boolean**

Operasi boolean menggunakan operand yang bertipe boolean serta hasil boolean untuk operasinya. Ada 4 operator boolean yang dikenal Pascal, yaitu:

**Tabel 15.** Tabel operator Boolean

OPERATOR	OPERASI
NOT	Ingkaran bit
AND	Logika AND
OR	Logika OR
XOR	Logika XOR

**Operasi String**

Operasi string pada Pascal hanya mengenal sebuah operator yaitu operator + atau penggabungan antar string.

### Operasi himpunan

Pada operasi himpunan selalu menggunakan tipe data himpunan (set) untuk operator maupun operannya. Ada 3 macam operator pada operasi himpunan, yaitu:

**Tabel 16** Tabel operator himpunan

OPERATOR	OPERASI
+	Union
-	Selisih
*	Interseksi

Hasil dari operasi-operasi himpunan berdasarkan aturan sebagai berikut:

- ? Nilai ordinal C ada di A+B hanya jika C ada di A atau B
- ? Nilai ordinal C ada di A-B hanya jika C terdapat di A dan bukan di B
- ? Nilai ordinal C ada di A\*B hanya jika C terdapat baik di A maupun B

### Operasi Relasi

Operasi ini digunakan untuk melakukan perbandingan dan menghasilkan boolean (true atau false) yang menunjukkan perbandingan tersebut bernilai benar atau salah.

**Tabel 17.** Tabel operator relasi

OPERATOR	OPERASI	TIPE OPERAN	TIPE HASIL
=	Sama dengan	Tipe sederhana, pointer, himpunan dan string	Boolean
<>	Tidak sama dengan	Tipe sederhana, pointer, himpunan dan string	Boolean
<	Lebih kecil	Tipe sederhana, pointer, himpunan dan string	Boolean
>	Lebih besar	Tipe sederhana, pointer, himpunan dan string	Boolean
<=	Lebih kecil sama dengan	Tipe sederhana, pointer, himpunan dan string	Boolean
>=	Lebih besar sama dengan	Tipe sederhana, pointer, himpunan dan string	Boolean
<=	Subset dari	Himpunan	Boolean
>=	Superset dari	Himpunan	Boolean
In	Anggota dari	Operan kiri: sembarang tipe ordinal Operan kanan: himpunan	Boolean

---

---

# Bab 5

## STATEMENT

---

---

Statement adalah perintah yang dikenal oleh Pascal. Dalam bahasa Pascal terdapat 11 statement, artinya SEMUA program dalam Pascal hanya menggunakan kombinasi dari ke-11 statement ini saja. Statement-statement yang dikenal Pascal adalah :

1. Assignment (pemberian nilai)
2. Compound (penggabungan)
3. IF – THEN – ELSE
4. CASE – OF
5. FOR – TO – DO
6. REPEAT – UNTIL
7. WHILE – DO
8. WITH
9. Procedure Call
10. Goto
11. Inline

Statement GOTO adalah statement yang jarang digunakan karena dianggap sebagai statement yang tidak sesuai dengan konsep pemrograman terstruktur. Sedangkan statement inline adalah statement untuk menjalankan instruksi bahasa mesin dan tidak digunakan dalam TOKI ataupun IOI.

### 1. Assignment (Pemberian nilai)

Statement Assignment digunakan untuk memberikan nilai pada sebuah variabel.

#### Sintaks penulisan:

```
Variabel := nilai
```

Contoh 1:

```
var a, b: byte;  
begin  
  a:=10;  
  a:=a*4;  
  if a>5 then b:=40;  
end.
```

Contoh 2:

```
var a, b: byte;  
begin  
  a:=0;  
  a=a+4;  
end.
```

Ada berapa assignment statement dari contoh di atas?

Catatan untuk FreePascal

Assignment	Hasil
A += b	Menambah <i>b</i> ke <i>a</i> , hasil disimpan di <i>a</i> .
A -= b	Mengurangi <i>b</i> dari <i>a</i> , hasil disimpan di <i>a</i> .
A *= b	Mengalikan <i>a</i> dengan <i>b</i> , hasil disimpan di <i>a</i> .
A /= b	Membagi <i>a</i> terhadap <i>b</i> , hasil disimpan di <i>a</i> .

## 2. Compound Statement

Digunakan untuk menggabungkan beberapa statement. Dalam teks lain disebutkan:

*Compound statements are a group of statements, separated by semicolons, that are surrounded by the keywords Begin and End. The Last statement doesn't need to be followed by a semicolon, although it is allowed.*

**Sintaks Penulisan:**

```
begin  
  statement;  
  statement;  
  ...  
  statement  
end
```

Contoh:

```
if a>5 then  
begin  
  a:=a*4;  
  b:=a-4  
end;
```

Ada berapa statement assignment pada potongan program di atas? Ada berapa statement pada potongan program di atas? Mana yang disebut compound statement?

## 3. IF – THEN – ELSE Statement

Merupakan perintah percabangan yang akan menjalankan statement sesuai dengan kondisi yang ada.

Ada 2 macam sintaks penulisan IF – THEN – ELSE:

**IF kondisi THEN statement**

Dan

**IF kondisi THEN statement ELSE statement**

Contoh:

```
If a>5 then a:=a+1;
If a-4=2 then begin a:=2; b:=a end;
if (a=3) and (b=2) then
begin
  a:=a-1;
  b:=a
end else b:=3;

if a>5 then a:=4
else begin
  a:=4 mod b;
  b:=b-1
end;
```

Tentukan kondisi-kondisi pada IF Statement yang ada! Apakah compound statement digunakan pada contoh di atas? Apakah assignment statement digunakan pada contoh di atas? Jika ya, tentukan di mana letak compound dan assignment statement!

#### 4. FOR – TO / DOWNTWO – DO Statement

Digunakan untuk mengulang statement

**Sintaks penulisan**

**FOR variabel := awal TO akhir DO statement**

Atau

**FOR variabel := akhir DOWNTWO awal DO statement**

**Catatan:**

Statement akan diulang sebanyak akhir-awal+1. Selama perulangan, nilai variabel akan bernilai dari awal sampai akhir

Contoh:

```
{Program 1}      {Program 2}
var              var
  a: byte;        a, b: byte;
begin            begin
  for a:=1 to 5 do  for a:=5 downto 2 do
    write(a);      for b:=1 to a do write(a+b)
  end.              end.
```



write(a) pada contoh 1, merupakan statement procedure call (pemanggilan prosedur)

Pada contoh 2, setelah for a:=5 downto 2 do, statement berikutnya adalah statement for – to – do. Berapa kali statement write(a+b) dieksekusi? Tuliskan output dari contoh 2.

## 5. CASE OF Statement

Statement case digunakan untuk perintah bercabang dengan banyak kondisi. Terdiri dari ekspresi (atau biasa disebut dengan selector) dan serangkaian statement.

### Sintaks penulisan:

```
case expression of
  case: statement;
  ...
  case: statement;
end
```

Atau:

```
case expression of
  case: statement;
  ...
  case: statement;
else
  statement
end
```

### Catatan:

*Jika pada sebuah kondisi dapat terdiri dari banyak range, maka dapat kondisi-kondisi yang ada dapat dipisahkan dengan koma.*

Contoh:

```
case Ch of
  'A'..'Z': WriteLn('Letter');
  '0'..'9': WriteLn('Digit');
  '+', '-', '*', '/': WriteLn('Operator');
else
  WriteLn('Special character');
end;
```

## 6. Repeat – Until Statement

Perintah yang ada di dalam statement repeat – until akan diulang sehingga kondisi boolean pada until bernilai true.

**Sintaks penulisan:**

```
repeat
  statement;
  statement;
  ...
  statement
until expression
```

Contoh:

```
repeat Ch := GetChar until Ch <> ' ';

repeat
  Write(' Enter value: ');
  ReadLn(I);
until (I >= 0) and (I <= '9');
```

Catatan: *GetChar pada contoh pertama merupakan statement procedure call.*

**7. While – Do Statement**

Statement while terdiri dari sebuah ekspresi boolean yang mengontrol eksekusi dari sebuah statement.

**Sintaks Penulisan:**

```
While expression do statement;
```

Atau

```
While expression do
Begin
Statement;
Statement;
...
End.
```

Contoh:

```
while Ch = ' ' do Ch := GetChar;

while not Eof(InFile) do
begin
  ReadLn(InFile, Line);
  WriteLn(OutFile, Line);
  Inc(LineCount);
end;
```

Catatan:

*Contoh kedua adalah statement while yang menggunakan compound statement di dalamnya.*

## 8. Statement With

Statement with digunakan untuk mereferensi field pada sebuah record.

### Sintaks Penulisan:

```
with var, var, ... var do statement
```

Contoh:

```
with Date[I] do  
begin  
    month := 1;  
    year  := year + 1;  
end;
```

Ekivalen dengan perintah:

```
Date[I].month := 1;  
Date[I].year  := Date[I].year + 1;
```

---

# Bab 6

## PROSEDUR DAN FUNGSI

---

Konsep subprogram atau modul merupakan konsep yang sangat bermanfaat dalam penyusunan sebuah program yang besar. Subprogram atau modul berkembang sejalan dengan teknik pemrograman terstruktur, yaitu membagi masalah yang besar menjadi masalah-masalah kecil yang diselesaikan dalam sebuah subprogram. Jika subprogram telah benar untuk memecahkan masalah-masalah kecil, maka diharapkan dalam penyusunan program yang besar, program dapat lebih disederhanakan sehingga memudahkan memecahkan masalah utama.

Dalam Pascal ada 2 macam subprogram, yaitu prosedur (*procedure*) dan fungsi (*function*). Bentuk umum penulisan prosedur atau fungsi dalam Pascal adalah:

```
PROGRAM      nama(file1, file2, file3);
CONST       dekl arasi konstanta;
VAR         dekl arasi vari abel ;
TYPE        dekl arasi type;
LABEL       dekl arasi label ;
FUNCTION    dekl arasi fungsi ;
PROCEDURE   dekl arasi prosedur;
```

```
PROCEDURE lain1;
BEGIN
    statement1;
    statement2;
    ...
END;
```

```
FUNCTION lain2: tipe data;
BEGIN
    statement1;
    statement2;
    ...
    lain2: =nilai ;
END;
```

```
BEGIN
    statement1;
    statement2;
    statement3;
    ...
END.
```

## Procedure

Prosedur mempunyai struktur yang sama dengan struktur program, yaitu terdiri dari nama prosedur, deklarasi-deklarasi dan bagian utama dari prosedur itu sendiri. Di dalam prosedur tersebut, dimungkinkan ada prosedur lagi yang strukturnya sama. Bentuk prosedur dalam prosedur tersebut disebut dengan prosedur tersarang (*nested procedure*). Struktur penulisan prosedur adalah sebagai berikut:

```
PROCEDURE namaprocedure(daftarparameterformal);  
{bagian deklarasi}  
BEGIN  
    Statement1;  
    Statement2;  
    Statement3;  
    ...  
END;
```

Daftar parameter formal dan tanda kurung (setelah namaprocedure) bersifat opsional. Artinya bisa digunakan, bisa juga tidak. Parameter formal terdiri dari 2 macam, yaitu:

- ? Parameter nilai (*value parameter*)
- ? Parameter perubah (*variable parameter*)

Sebelum membahas kedua parameter tersebut, maka sebelumnya harus diketahui bagaimana mekanisme suatu pemanggilan prosedur. Misalnya diketahui sebuah program sebagai berikut:

```
USES crt;  
  
PROCEDURE tulis(x, y: integer; s: string);  
BEGIN  
    GOTOXY(x, y);  
    WRITE(s);  
END;  
  
BEGIN  
    Tulis(10, 2, 'Hello World');  
END.
```

Dari program di atas, yang dimaksud dengan **parameter aktual** adalah 10, 2, dan 'Hello World' (pada pemanggilan prosedur tulis). Sedangkan pada deklarasi prosedur tulis, x, y, dan s disebut sebagai **parameter formal**.

Mekanisme pemanggilan prosedur tulis tersebut adalah:

Program bekerja dari program utama, yaitu memanggil procedure tulis. Pada waktu terjadi pemanggilan procedure tulis, maka isi parameter aktual diberikan diberikan pada parameter formalnya. Nilai ini mungkin diproses dalam procedure tersebut. Nilai terakhir dari parameter formal tersebut **TIDAK** dikembalikan pada parameter aktualnya. Parameter seperti inilah yang disebut dengan **parameter nilai**. Yang dapat digunakan sebagai parameter nilai adalah: konstanta, variabel dan ekspresi.

```

VAR x, y: integer;
PROCEDURE tukar(var a, b: integer);
  Var c: integer;
BEGIN
  c: =a;
  a: =b;
  b: =c;
END;

BEGIN
  x: =5;
  y: =1;
  tukar(x, y);
END.

```

Parameter aktual dari program di atas adalah x dan y. Sedangkan parameter formalnya adalah a dan b. Dalam parameter formal, terdapat kata var yang menunjukkan bahwa parameter tersebut adalah ***parameter perubah***.

Pada waktu procedure yang mempunyai *parameter perubah* dipanggil, parameter aktual (x dan y) pada pemanggilnya akan digantikan parameter formal. Jadi yang diolah pada procedure di atas adalah parameter aktualnya (x dan y). Konsekuensi pemanggilan procedure dengan menggunakan *parameter perubah* adalah setiap perubahan nilai yang terjadi dari parameter aktual (di dalam procedure) akan mengakibatkan turut berubahnya nilai parameter aktual (di dalam pemanggil atau program utama).

Yang dapat dijadikan *parameter perubah* adalah hanya variabel, sedangkan konstanta dan ekspresi TIDAK dapat dijadikan sebagai *parameter perubah*.

## Function

Sebenarnya, secara prinsip fungsi sama seperti prosedur (sama-sama merupakan subprogram). Perbedaan fungsi dan prosedur adalah fungsi dapat langsung memberikan nilai kembali sedangkan pada procedure harus menggunakan parameter perubah agar dapat mengembalikan suatu nilai.

Bentuk umum suatu fungsi dalam Pascal adalah sebagai berikut:

```

FUNCTION namafungsi (daftarparameterformal): tipe data;
{bagian deklarasi}
BEGIN
  Statement1;
  Statement2;
  Statement3;
  ...
  namafungsi := nilai;
END;

```

Dalam fungsi, semua parameter formal sebaiknya merupakan parameter nilai walaupun diperbolehkan menggunakan parameter perubah.

## Rekursi

Proses rekursi adalah suatu proses di mana sebuah subprogram dapat memanggil dirinya sendiri. Penggunaan rekursi ini akan sangat membantu pemecahan-pemecahan soal yang tidak dapat diselesaikan dengan menggunakan teknik iterasi biasa.

Karena rekursi merupakan pemanggilan dirinya sendiri (procedure atau fungsi), maka dalam rekursi tersebut haruslah ada yang namanya proses terminasi/point sentinel (kapan berhenti memanggil dirinya sendiri), ini menghindari proses rekursi yang tak terdefinisikan (terus menerus memanggil dirinya sendiri). Proses terminasi ini biasanya berupa kondisi. Perhatikan program di bawah ini:

```
FUNCTION jumlah(n: integer): integer;  
BEGIN  
  IF n=1 THEN  
    jumlah:=1;  
  ELSE  
    jumlah:=jumlah(n-1)+n;  
END;  
  
BEGIN  
  writeln(jumlah(3));  
END.
```

Cara kerja rekursi adalah secara terbalik/mundur sehingga terbentur dengan proses terminasi atau point sentinel. Setelah terbentur dengan point sentinel, maka proses akan berjalan maju dengan membawa proses perhitungan pada saat proses terbentur dengan point sentinel tadi.

Pada proses di atas, point sentinel adalah pada saat  $n=1$ . Selama  $n$  tidak sama dengan 1, maka function jumlah akan terus menerus dipanggil sampai terbentur dengan point sentinel (yaitu  $n=1$ ).

Jika function jumlah dipanggil dengan:

```
jumlah(3),
```

Maka proses yang terjadi adalah:

Jumlah(3) akan memanggil: jumlah(2) + 3

Jumlah(2) akan memanggil: Jumlah(1) + 2

Jumlah(1) akan menghasilkan 1 (karena terbentur dengan point sentinel, sehingga tidak memanggil dirinya sendiri lagi).

Setelah terbentur dengan point sentinel (dengan membawa hasil 1), maka proses akan berjalan maju untuk menyelesaikan tugasnya, yaitu kembali pada:

Jumlah(1) + 2, karena jumlah(1) membawa atau bernilai 1, maka pada proses ini akan menghasilkan nilai 3. Nilai 3 ini akan dibawa ke pemanggil berikutnya, yaitu jumlah(2).

Pada jumlah(2) + 3, akan terjadi proses  $3 + 3$  (karena jumlah(2) membawa hasil 3), sehingga pemanggilan jumlah(3) akan menghasilkan 6.



---

# Bab 7

## Standard Input / Output

---

Sebuah program komputer, selalu bekerja dengan 3 tahap, yaitu:

- Tahap 1 : MASUKAN
- Tahap II : PROSES
- Tahap III : KELUARAN

Contoh:

Program untuk menghitung nilai dua bilangan bulat A dan B, maka:

- masukan : A dan B
- proses :  $C = A + B$
- keluaran : C

Dalam bahasa Pascal:

```
var a, b, c: word;
begin
  write('Nilai A : '); readln(a);      {masukan}
  write('Nilai B : '); readln(b);      {masukan}
  c:=a+b;                               {proses}
  write('Hasil penjumlahan A+B : ', c) {keluaran}
end.
```

### Standard Masukan

Standard masukan pada Pascal adalah keyboard. Artinya semua masukan dimasukkan lewat keyboard. Standard output pada Pascal adalah monitor. Artinya semua output ditampilkan pada layar monitor.

Misalkan diminta untuk membuat program yang mencari nilai terbesar dari 1.000 buah data. Untuk program tersebut, entry data satu persatu melalui keyboard menjadi cara yang tidak efisien, karena setiap kali program di-RUN, akan meminta 1.000 masukan. Untuk itu, standard input dapat dialihkan ke sebuah file teks. Artinya, input tidak lagi dimasukkan dari keyboard, melainkan lewat sebuah file teks. Nantinya file teks inilah yang dibaca setiap kali program di-RUN (sehingga user tidak perlu menginput lagi).

Pada Pascal, untuk merubah standard input ke file teks adalah dengan perintah:

```
Assign(input, namafile);  
Reset(input);
```

Namafile pada perintah assign berupa string yang menunjukkan nama file untuk file yang hendak dibaca. Setelah kedua perintah tersebut, semua perintah READLN, langsung akan membaca dari namafile.

### Standard Keluaran

Standard keluaran pada Pascal adalah layar monitor. Namun layar monitor hanya dapat menampilkan maksimal 25 baris dan 80 karakter tiap barisnya. Jika diminta untuk membuat program yang dapat menampilkan output lebih dari 25 baris, maka layar monitor tidak lagi dapat menjadi standard output yang baik.

Untuk merubah standard output pada Pascal:

```
Assign(output, namafile);  
Rewrite(output);
```

Namafile berupa string yang menunjukkan nama file yang hendak dibuat / ditulis. Setelah kedua perintah tersebut, hasil output (dari perintah write), akan ditulis pada namafile.

Cara lain untuk merubah standard input atau output adalah dengan mendeklarasikan sebuah variabel bertipe text.

Contoh:

```
Var  
  finput, foutput: text;  
Begin  
  assign(finput, 'Input.in');  
  reset(finput);  
  readln(finput, a, b, c);  
  close(input)  
  
  {... proses ...}  
  
  assign(foutput, 'output.out');  
  rewrite(foutput);  
  write(foutput, 'something');  
  close(foutput);  
End.
```

---

Perlu diperhatikan agar seperlunya saja membuka file. Jika file tersebut sudah tidak diperlukan lagi, maka sebaiknya diberikan perintah `close(f)`. Begitu juga untuk membuat file, jika output belum siap ditulis karena masih dalam proses, sebaiknya file tersebut tidak di-rewrite terlebih dulu.

# INDEX

---

## A

*AND* · 36, 52  
*Ansistrings* · 30, 41  
*Aritmatika* · 51  
*Array* · 30, 31, 40, 43  
*ARRAY* · 49  
*ASCII* · 18, 39  
*Assembler* · 1, 7, 12  
*Assignment* · 54, 55

---

## B

*Bahasa C* · 2, 5  
*BAHASA C* · 5  
*Based Types* · 30, 33  
*BASIC* · 2, 4  
*Biner* · 29  
*Blok* · 11, 21  
*Bookmark* · 11  
*Boolean* · 30, 36, 44, 45, 46, 48, 52, 53  
*Byte* · 30, 37

---

## C

*C++* · 6  
*Calculator* · 19  
*Cardinal* · 37  
*CASE – OF* · 54

*CASE OF* · 57  
*Char* · 30, 42, 49, 50  
*Character* · 30, 39  
*CHR* · 39  
*COBOL* · 2, 3, 4  
*Code* · 4, 12, 13  
*Comments* · 12  
*Comp* · 39  
*compiler 32 bit* · 17  
*Completion* · 12  
*Compound* · 54, 55  
*CONST* · 24, 25, 60  
*Constant* · 30, 48  
*Constant string* · 30  
*Ctrl* · 11, 13, 14, 15, 16, 19, 20, 21, 22, 23

---

## D

*Debug* · 14, 15, 16  
*DEC* · 33  
*deklarasi* · 24, 25, 29, 41, 48, 61, 62  
*Deklarasi* · 24, 25, 26, 31, 41, 46, 47, 48  
*difference* · 44  
*Directive* · 12  
*Double* · 39, 48

---

## E

*Enumerated* · 30, 33  
*equality* · 44  
*EXCEPT* · 17

*EXIT* · 3, 4, 18  
*Extended* · 39

---

## F

*FAR* · 17  
*Field* · 45  
*File* · 10, 17, 30, 31, 46, 47  
*FINALLY* · 17  
*floating point* · 8, 39  
*FOR – TO – DO* · 54  
*FOR – TO / DOWNTO – DO* · 56  
*FORTRAN* · 2, 3  
*Free Pascal* · 7, 8, 10, 16, 17, 24  
*FUNC* · 36  
*Function* · 17, 26, 47, 62  
*FUNCTION* · 24, 26, 60, 62, 63  
*fungsi* · 7, 17, 18, 25, 28, 33, 34, 35, 36, 39, 40, 60, 62, 63

---

## G

*Goto* · 14, 28, 54

---

## H

*Hex Number* · 12  
*HIGH* · 33  
*himpunan* · 43, 51, 53

## I

IDE · 7, 10, 11, 13, 15, 16, 18, 20  
 Identifier · 28, 33  
 IF – THEN – ELSE · 54, 55, 56  
 INC · 33  
 indentifier · 29  
 inequality · 44  
 Inline · 54  
 Input · 65, 66  
 Insert · 11, 22  
 Int64 · 37  
 Integer · 26, 30, 37, 41, 45, 47, 48, 49, 50  
 Integrated Developing Environment · 10  
 Interpreter · 2, 6  
 INTERRUPT · 17  
 intersection · 44

## K

Keluaran · 66  
 Kompiler · 2

## L

LABEL · 24, 25, 60  
 Length · 26, 40, 41  
 Logika · 52  
 Longint · 30, 37, 51, 52  
 Longword · 37  
 LOW · 33

## M

Masukan · 65

membership · 44  
 modul · 60

## N

NEAR · 17  
 NOT · 36, 37, 52  
 Null terminated string · 30  
 Numbers · 12, 29

## O

Oktal · 29  
 Operator · 38, 39, 51, 57  
 OR · 36, 52  
 ORD · 33, 36  
 Ordinal · 30, 33  
 ordinal types · 30  
 Output · 65

## P

parameter formal · 61, 62, 63  
 Parameter list · 17  
 parameter nilai · 61, 63  
 Pascal · 2, 4, 5, 7, 8, 9, 10, 11, 12, 16, 17, 18, 20, 24, 27, 28, 29, 30, 31, 32, 33, 34, 36, 37, 40, 41, 42, 47, 52, 54, 60, 62, 65, 66  
 PASCAL · 5, 7, 24, 51  
 Pchar · 30, 42  
 Pointer · 30, 31, 41, 47  
 pred · 34, 35, 39  
 PRED · 33  
 Preferences · 12  
 Procedural · 47  
 Procedure · 26, 47, 54, 61

PROCEDURE · 4, 24, 26, 60, 61, 62  
 Procedure Call · 54  
 program · 1, 2, 3, 4, 5, 6, 7, 8, 10, 14, 15, 16, 20, 21, 22, 24, 25, 26, 27, 28, 29, 33, 34, 35, 36, 40, 42, 43, 44, 47, 54, 55, 60, 61, 62, 63, 65, 66  
 prosedur · 7, 25, 28, 33, 39, 57, 60, 61, 62  
 prosedur dec · 39  
 prosedur inc · 39  
 PROTECTED · 17  
 PUBLIC · 17  
 PUBLISHED · 17

## Q

Qword · 37

## R

RAISE · 17  
 Real · 30, 38, 39, 45, 47, 49, 50  
 real types · 30  
 record · 17, 28, 45, 46, 47, 49, 59  
 Record · 30, 31, 45  
 RECORD · 49  
 Rekursi · 63  
 Repeat – Until · 57  
 REPEAT – UNTIL · 54  
 reserved word · 17, 29  
 Reserved Word · 12  
 Reserved Words · 27  
 Run · 13, 14

---

## S

*set* · 15, 28, 44, 49, 51, 53  
*Set* · 30, 31, 43  
*SET* · 9, 49  
*Short strings* · 30  
*Shortcut* · 20, 21, 22, 23  
*Shortint* · 30, 37  
*shortstring* · 41  
*Single* · 39  
*Sintaks* · 26, 46, 54, 55, 56, 57, 58, 59  
*smallint* · 37  
*Smallint* · 37  
*statement* · 25, 27, 46, 51, 52, 54, 55, 56, 57, 58, 59  
*string* · 12, 26, 28, 30, 40, 41, 42, 46, 47, 50, 52, 53, 61, 66  
*String* · 6, 30, 40, 41, 42, 52  
*Strings* · 12, 30  
*Structured types* · 31  
*Subprogram* · 60

*Subrange* · 30, 35  
*subset* · 44  
*succ* · 34, 35, 39  
*SUCC* · 33  
*superset* · 44  
*Symbol*, · 12  
*Symbols* · 27  
*Syntax* · 11

---

## T

*Tabs* · 12  
*Tipe Prosedur* · 31  
*TRY* · 17  
*Turbo Pascal* · 8, 10, 16, 17, 28, 47  
*Type* · 30, 34, 39, 42, 45, 47  
*TYPE* · 24, 25, 60

---

## U

*union* · 44  
*UpCase* · 40, 41

---

## V

*VAR* · 5, 24, 25, 60, 62

---

## W

*Watch* · 15, 20  
*While – Do* · 58  
*WHILE – DO* · 54  
*Whitespace* · 12  
*With* · 46, 59  
*WITH* · 54  
*Word* · 12, 29, 30, 37

---

# Lampiran

---

Pada lampiran ini akan diberikan contoh soal dasar teori Pascal dan pembahasannya.

**Soal 1.**

Deklarasi konstanta manakah yang salah?

- a. `const rata-rata=10;`
- b. `const tinggi badan=150;`
- c. `const pi=22/4;`
- d. `const suhu=20000;`
- e. `const duaaxtiga=8;`

**Jawab:**

- a. `const rata-rata=10`

**Pembahasan:**

Salah satu syarat dari penamaan identifier adalah: Karakter ke dua dan selanjutnya dapat berupa huruf, angka, atau underscore

**Soal 2.**

Manakah yang benar dari deklarasi di bawah ini?

- a. `Program abc;`  
`Var x, y: real;`
- b. `Program pqrstu2343;`  
`Var x, y: real;`
- c. `Program 4343;`  
`Var x: =boolean; b: =integer;`
- d. `Program ku;`  
`var z, y: char=' a' ;`
- e. `Program gampang;`  
`Var a: byte, c: real;`

**Jawab:**

- a. `Program abc;`  
`Var x, y: real;`

**Pembahasan:**

- Opsi b, c dan d salah karena memberikan tidak memberikan tipe variabel dengan tanda: .
- Opsi e salah karena memisahkan antar variabel dengan tanda , (seharusnya tanda ; )

**Soal 3.**

Manakah yang mendeklarasikan tipe enumerasi dengan tepat?

- a. Type a=integer;
- b. Type a=1..300;
- c. Type a=(baik, jelek, buruk);
- d. Type a=[baik, jelek, buruk];
- e. Type a=baik, jelek, buruk;

**Jawab:**

- c. Type a=(baik, jelek, buruk);

**Soal 4.**

Bagaimana mendeklarasikan konstanta bertipe array 3 x 3?

- a. Const a: array[3, 3]=((1, 2, 3), (2, 3, 4), (3, 4, 5));
- b. Const a[3, 3]=(1, 2, 3, 2, 3, 4, 3, 4, 5);
- c. Const a: array[1..3, 1..3] of byte = ((1, 2, 3), (2, 3, 4), (3, 4, 5));
- d. Const a: array[1..3, 1..3] of byte = (1, 2, 3, 2, 3, 4, 3, 4, 5);
- e. Const a: array[1..3] of array[1..3] of byte = ((1, 2, 3), (2, 3, 4), (3, 4, 5));

**Jawab:**

- c. Const a: array[1..3, 1..3] of byte = ((1, 2, 3), (2, 3, 4), (3, 4, 5));

atau

- e. Const a: array[1..3] of array[1..3] of byte = ((1, 2, 3), (2, 3, 4), (3, 4, 5));

**Pembahasan:**

Baik c maupun e dapat mendeklarasikan array 3 x 3.

**Soal 5.**

Apakah output program berikut ini?

```
var U: Integer;
begin
  U:=3;
  Writeln('Umur saya', U:3, ' tahun'),
End.
```



- a. Umur saya 3 tahun
- b. Umur saya 1 tahun
- c. Umur saya 3 tahun
- d. Umur saya 003 tahun
- e. Umur saya 3.00 tahun

**Jawab:**

- c. Umur saya 3 tahun

### Soal 6.

Bagaimana cara anda mengetikkan input untuk program berikut ini?

```
var Umur, Tinggi, Berat: Real;  
begin  
  Write('Masukkan umur, tinggi ' dan berat badan Anda: ');  
  Readln(Umur, Tinggi, Berat);  
end.
```

- a. 16,170, 61.5
- b. 16; 170; 61.5
- c. 16 170 61.5
- d. 16 170 61.5
- e. 16;170;161,5

**Jawab:**

- a. 16,170,61.5

### Soal 7.

```
var r : real;  
begin  
  r:=147.0;  
  writein(r: 0: 5),  
end.
```

Apa tampilan program di atas ?

- a. 00147
- b. 0147.0
- c. 147.000
- d. 147.00000
- e. 147.000000

**Jawab:**

- e. 147.000000

**Pembahasan:**

Penulisan bilangan real pada statement writeln adalah:

```
writeln(OutputVariable : NumChars [: Decimals ])
```

Penulisan bilangan real pada statement writeln adalah:

NumChars adalah banyaknya space yang disiapkan untuk penulisan, dan Decimals adalah banyaknya angka desimal (angka di belakang koma).

**Soal 8.**

```
var i, j: byte;  
begin  
  i:=100; j:=200;  
  writeln(i*j)  
end.
```

Apa tampilan program di atas?

- a. 12                      b. 22                      c. 32
- d. 20000                e. 400000

**Jawab:**

- d. 20000

**Soal 9.**

Perhatikan source code di bawah ini

```
var  
  Mail: word;  
begin  
  Mail:= ' TOKI';  
  writein (' TOKI');  
  writeln (Mail);  
end.
```

Output yang tercetak di layar setelah eksekusi adalah

- a. TOKI
- b. TOKITOKI
- c. TOKITOKITOKI
- d. terjadi compiler error
- e. tidak ada jawaban yang benar

**Jawab:**

- d. terjadi compiler error

**Pembahasan:**

Tipe data untuk variabel Mail adalah berjenis numerik (Word) sehingga tidak dapat diberikan nilai yang berjenis teks atau string  
 $0101 \text{ shi } 2 = 10100 = 20$

**Soal 10.**

Bagaimana keluaran program di bawah ini?

```
Var  
  I: integer;  
Begin  
  I:=2;  
  Case I of  
    1, 3, 5, 7, 9: writeln(' Ganjil ');  
    2: writeln(' Prima genap ');  
    0..10: writeln(' Normal ');  
  else writeln(' Tidak normal ');  
  end;  
end;
```

- a. Prima genap
- b. Normal
- c. Prima genap  
Normal
- d. Normal  
Prima genap
- e. Prima genap  
Tidak normal

**Jawab:**

- a. Prima genap

**Pembahasan:**

Struktur kendali case akan segera keluar untuk menjalankan statement berikutnya setelah menemukan nilai yang tepat.

Perhatikan program di bawah ini:

```
var I, j, k: integer;  
    L: byte;  
begin  
  i:=3;  
  j:=4;  
  k:=32;  
  L:=0;  
  {If - 1 }  
  if i + j and k =0 then  
    writeln(' Betul ')  
  else  
    writeln(' Salah );  
  {If - 2 }  
  if ( i = 2) and (j < i) or (k > i) then  
    writeln(' Betul ')
```

```
else  
  writeln(' Salah');  
{If - 3}  
if not L in [1..120] then  
  writeln(' Betul')  
else  
  writeln(' Salah');  
end.
```

Program ini berisi tiga perintah if then else yang saling tidak berkaitan, masing-masing IF diberi nama IF - 1, IF - 2, IF - 3.

**Soal 11.**

Perintah if manakah yang tidak dibenarkan:

- a. If - 1
- b. If - 2
- c. If - 3
- d. If - 1 dan if - 2
- e. Tidak ada if yang salah

**Jawab:**

- e. Tidak ada if yang salah

**Pembahasan:**

Pada If - 1, ekspresi  $i+j$  and  $k$  adalah ekspresi matematika dengan urutan pengerjaan  $j$  and  $k$  kemudian ditambahkan dengan  $i$ . Ini merupakan ekspresi yang valid dalam bahasa Pascal

Pada If - 3, ekspresi Not  $L$  akan dioperasikan terlebih dulu. Ini juga merupakan ekspresi yang valid dalam bahasa Pascal.

**Soal 12.**

Pada program di atas, if mana yang menghasilkan output "Betul"?

- a. If - 1
- b. If - 2
- c. If - 3
- d. If - 1 dan if - 2
- e. Tidak ada if yang menghasilkan "Betul"

**Jawab:**

- b. If - 2

**Pembahasan:**

Urutan pengerjaan operator AND dan OR adalah AND akan dievaluasi terlebih dulu. Pada kondisi pertama,  $(i = 2)$  and  $(j < i)$  akan menghasilkan nilai FALSE, namun pada saat dievaluasi dengan menggunakan kondisi OR, yaitu  $(k > i)$ , akan menghasilkan TRUE, sehingga yang dicetak adalah "Betul"

**Soal 13.**

Pada program di atas, manakah pernyataan yang benar?

- a. Ekspresi  $\text{not } L \text{ in } [1..120]$  ekuivalen dengan  $L \text{ in } [0,121..255]$
- b. Ekspresi  $I + J \text{ and } K = 0$  dalam if – 1 ekuivalen dengan  $(I + J = 0)$  and  $(K = 0)$
- c. Kondisi pada if – 1 akan bernilai lain jika kondisi diubah menjadi  $(I + J) \text{ AND } K = 0$
- d. Kondisi pada if – 2 akan bernilai lain jika kondisi diubah menjadi  $((I=2) \text{ and } (J<1)) \text{ or } (k>1)$
- e. Kondisi pada if – 3 akan bernilai lain jika kondisi diubah menjadi  $\{\text{NOT } L \text{ in } [1..120]\} \text{ OR } (L \text{ in } [121,122])$

**Jawab:**

- c. Kondisi pada if – 1 akan bernilai lain jika kondisi diubah menjadi  $(i + j) \text{ AND } K = 0$

**Pembahasan:**

Hal ini disebabkan bahwa operator + dan AND yang dievaluasi terlebih dulu adalah operator AND. Jika tanda kurung diberikan untuk ekspresi  $(i+j)$ , maka penjumlahan  $i$  dan  $j$  akan dijalankan terlebih dulu baru kemudian dioperasikan dengan AND.

**Soal 14.**

Berapa keluaran program bila pemakai memberi nilai  $M=9$ ?

```
Var C, M, J: integer;
Begin
  J:=1;
  write('M = '); readln(M);
  For c:=5 to M do
    J := J * (M 4);
  Writeln('J = ', J);
End.
```

- a.  $J = 120$
- b.  $J = -120$
- c.  $J = 1$
- d.  $J = 24$
- e. Salah semua

**Jawab:**

- e. Salah semua

**Pembahasan:**

Dari perulangan di atas, instruksi  $j := j * (m - 4)$  akan diulang sebanyak 5 kali, sehingga dapat dirumuskan  $j$  adalah  $5 \times 5 \times 5 \times 5 = 3125$ .

**Soal 15.**

```
Var
  i, j: Shortint;
Begin
  For i:=1 to 200 do
    Inc(j);
  end.
```

Program diatas akan menghasilkan:

- Nilai  $j = 200$ ;
- Nilai  $j = 127$ ;
- Tidak dapat dipastikan
- Terjadi error (code 76).
- Program tidak dapat berhenti

**Jawab:**

- Terjadi error (code 76).

**Pembahasan:**

Shortint adalah sebuah tipe data yang mempunyai jangkauan  $-128..127$ . Jika dipaksa diberikan nilai hingga 200, maka yang muncul adalah Error 76: Constant out of range.

**Soal 16.**

```
var
  h, i, j: integer;
begin
  h:=0;
  for j:=1 to 10 do
    inc(h); i:=1;
  j:=0;
  repeat
    inc(i);
    inc(j);
  until i=10;
  if h=j then writeln(' GOLD')
  else writeln(' SILVER');
end.
```

Apa keluaran program di atas

- a. GOLD
- b. SILVER
- c. GOLD  
SILVER
- d. SILVER  
GOLD
- e. tidak ada keluaran

**Jawab:**

b. SILVER

**Pembahasan:**

```
h:=0;  
for j:=1 to 10 do  
  inc(h); i:=1;  
j:=0;
```

Saat selesai mengeksekusi statement di atas, maka h akan bernilai 10, i bernilai 1 dan j bernilai 0. Selanjutnya program akan mengeksekusi statement:

```
repeat  
  inc(i);  
  inc(j);  
until i=10;
```

Karena nilai i sebelumnya adalah 1, maka statement inc(i) dan inc(j) akan dijalankan sebanyak 9 kali, sehingga setelah keluar dari perulangan repeat until, nilai i akan bernilai 10 dan j bernilai 9.

```
if h=j then writeln(' GOLD' )  
else writeln(' SILVER' );
```

Statement if di atas akan menjalankan statement sesudah else (karena nilai h tidak sama dengan j).

**Soal 17.**

```
var i,j : integer;  
begin  
  j:= 1;  
  for i:= 1 to 5 do  
    begin  
      writeln(i,' ',j);  
      j:= i-1;  
    end;  
end.
```

Output dari program di atas adalah

a. 1 1	b. 1 1	c. 1 0	d. 1 5	e. 1 1
2 1	2 2	2 1	2 4	2 0
1 3	3 3	3 2	3 3	3 1
4 1	4 4	4 3	4 2	4 2
1 5	5 5	5 4	5 1	5 3

**Jawab:**

d. 1 5  
 2 4  
 3 3  
 4 2  
 5 1

Program untuk 2 soal berikutnya

```
1. var s : string;
2. i: integer;
3. begin
4.   for i := 1 to length(s) do
5.     begin
6.       s[i] := s[length(s)-i+1];
7.     end;
8. end.
```

**Soal 18.**

Output dari program di atas jika s='SC!P!O' adalah

- a. s='SC!!CS'
- b. s='SSSSSS'
- c. s='O!PP!O'
- d. s='CCCCCC'
- e. s='!O!O!O'

**Jawab:**

c. s='O!PP!O'

**Soal 19.**

Output dari program di atas jika s='HaShMaT' dan baris ke-4 diubah menjadi "for i:=length(s) downto 1" adalah



- a. s='TaMhMaT'
- b. s='HaShSaH'
- c. s='HHHHHHH'
- d. s='aaaaaaa'
- e. s='aHaHaHa'

**Jawab:**

- b. s='HaShSaH'

**Soal 20.**

```
function ABC(k : integer);  
var i,j : integer;  
begin  
  j:=1;  
  for i:=1 to k do  
    j:=j*2;  
  ABC:=j;  
end;  
  
begin  
  writeln(ABC(4));  
end.
```

Apa keluaran program di atas ?

- a. 16
- b. 8
- c. 4
- d. 2
- e. program tidak dapat dijalankan

**Jawab:**

- e. program tidak dapat dijalankan

**Pembahasan:**

Soal ini merupakan soal jebakan. Perhatikan bahwa deklarasi dari sebuah function adalah:

Function namafunction(parameter): tipe;

Sebuah function harus mempunyai nilai kembali. Namun pada soal tidak terdapat nilai kembali.

**Soal 21.**

```
Function Sum(const A, B: Integer): Integer;  
Begin  
  Sumr =A+B;  
end;  
  
Begin  
  Writeln(Sum(5, 10));  
end.
```

Apa yang dihasilkan oleh program diatas.

- a. 5
- b. 10
- c. 15
- d. 20
- e. Tidak bisa di compile.

**Jawab:**

c. 15

**Pembahasan:**

Parameter const a,b:integer merupakan parameter konstanta (constant parameter).

**Soal 22.**

```
Begin
  Assign(Output, 'ABC.TXT');
  Rewrite(Output);
  Write('GO GET GOLD');
  Close(Output);
end.
```

Program diatas akan menghasilkan:...

- a. File 'ABC' dengan isi 'GO GET GOD'
- b. Run-time error.
- c. Tidak dapat dicompile karena variable 'Ouiput' tidak ada
- d. Tidak menghasilkan apa-apa
- e. Tidak ada yang benar

**Jawab:**

a. File 'ABC' dengan isi 'GO GET GOD'

**Pembahasan:**

Perintah Assign(Output,'ABC.TXT') adalah perintah yang mengarahkan standard output (keluaran) kepada file ABC.TXT. Jika tidak dirubah, maka standard output adalah pada monitor.

**Soal 23.**

Perintah mana yang tidak boleh digunakan untuk file bertipe text?

- a. Assign
- b. Reset
- c. EOF

- d. FilePos
- e. Semua boleh digunakan untuk Text

**Jawab:**

d. FilePos

**Pembahasan:**

Perintah FilePos adalah perintah untuk mengetahui posisi file pointer (penunjuk file), dan hanya dapat dioperasikan untuk file bertipe bukan text.

Gunakan program berikut ini untuk menjawab dua soal di bawah ini:

```
program Uji;  
var T: Text;  
    i, j, k: integer;  
begin  
    Assign(T, 'INPUT.TXT');  
    Reset(T);  
    Readln(T, i, j, k);  
    Writeln(i, j, k);  
    Readln(T, i);  
    Readln(T, j);  
    Writeln(i, j);  
    Close(T);  
End.
```

**Soal 24.**

Misalkan file INPUT.TXT berisi baris-baris sebagai berikut:

```
3 1 4 9  
5 2 6  
8 7  
0
```

Bagaimanakah output dari program tersebut?

- a. 3 1 4 9  
5 2 6  
8 7
- b. 3 1 4  
9 5
- c. 3 1 4  
5 2

d. 3 1 4  
5 8

e. Terjadi runtime error karena isi file INPUT.TXT tidak sesuai untuk program ini.

**Jawab:**

d. 3 1 4  
5 8

**Pembahasan:**

Perintah Readln akan melakukan pembacaan di baris berikutnya. Perintah Readln pertama akan melakukan pembacaan pada file baris pertama, perintah Readln berikutnya melakukan pembacaan pada baris ke dua dan perintah Readln terakhir melakukan pembacaan pada baris ketiga.

**Soal 25.**

Bila isi file INPUT.TXT adalah sebagai berikut:

1  
3  
2  
4  
5  
6

bagaimana output program ini?

a. 2 3 1  
4 5

b. 1 0 0  
3 2

c. 1 3 2  
4 5

d. 1 3 2  
5 6

e. Terjadi runtime error karena isi file INPUT.TXT tidak sesuai untuk program ini.

**Jawab:**

e. Terjadi runtime error karena isi file INPUT.TXT tidak sesuai untuk program ini