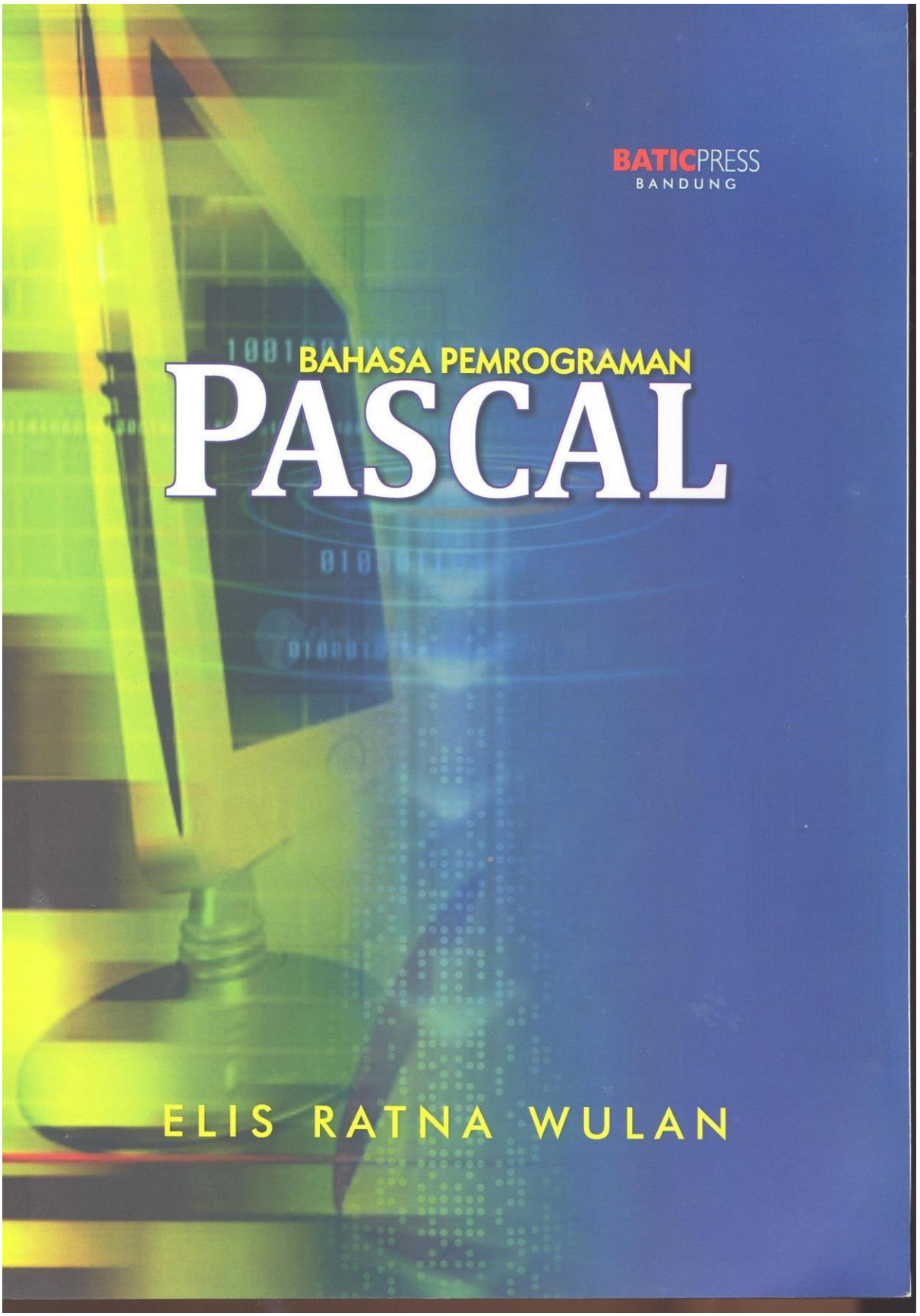

BATICPRESS
BANDUNG
BAHASA PEMROGRAMAN
PASCAL
ELIS RATNA WULAN

BAHASA PEMROGRAMAN PASCAL

Oleh :
Elis Ratna Wulan

Cetakan I, April 2010
Hak Cipta pada : Penulis

Diterbitkan oleh BATIC PRESS Bandung
Jl. Cikutra No. 276D, Bandung
Telp. (022) 7206964, Fax. (022) 7208592
Email : icmijabar@gmail.com

ISBN: 978-979-17799-6-8

E L I S

P A

BAB I
DASAR-DASAR ALGORITMA

1.1. PEMROGRAMAN KOMPUTER

Langkah-langkah yang kita lakukan dalam memberikan instruksi kepada komputer untuk memecahkan masalah dinamakan pemrograman komputer. Dalam pemrograman komputer kita harus menggunakan bahasa yang dimengerti oleh komputer untuk melakukan suatu instruksi.

Pada dasarnya, komputer adalah mesin digital, artinya komputer hanya mengenal kondisi ada arus listrik (dilambangkan dengan 1) dan tidak ada arus listrik (dilambangkan dengan 0). Oleh karena itu kita harus menggunakan sandi 0 dan 1 untuk melakukan pemrograman komputer. Bahasa pemrograman yang menggunakan sandi 0 dan 1 disebut bahasa mesin.

Karena bahasa mesin sangat susah, maka muncul ide untuk melambangkan untaian sandi 0 dan 1 dengan singkatan kata yang lebih mudah dipahami manusia. Singkatan kata ini disebut *mnemonic code*. Bahasa pemrograman yang menggunakan singkatan kata ini disebut bahasa *assembly*.

Sebagai contoh, dalam prosesor Intel, terdapat perintah 0011 1010 0000 1011. Perintah dalam bahasa mesin ini sama artinya dengan perintah *assembly* CMP AL, 0D, yang artinya bandingkan nilai register AL dan 0D. CMP adalah singkatan dari *CoMPare*. Perangkat lunak yang mengkonversikan perintah-perintah *assembly* ke dalam bahasa mesin disebut *assembler*.

Pemrograman dengan menggunakan bahasa *assembly* dirasakan masih terlalu sulit, akhirnya dikembangkan suatu bahasa pemrograman yang lebih mudah digunakan. Bahasa pemrograman ini menggunakan kata-kata yang mudah dikenali oleh manusia. Bahasa pemrograman ini disebut bahasa generasi ketiga atau disingkat 3GL (*third-generation language*). Atau disebut juga bahasa tingkat tinggi atau disingkat HLL (*high-level language*). Contohnya antara lain Basic, Pascal, C, C++ dan COBOL.

Bahasa generasi ketiga menggunakan kata-kata dalam bahasa Inggris karena bahasa tersebut merupakan bahasa internasional. Sebagai contoh, kita dapat menuliskan perintah berikut dalam bahasa Pascal:

```
writeln ('Algoritma');
```

atau perintah berikut dalam bahasa C:

```
printf (“Algoritma\n\r”);
```

atau perintah berikut dalam bahasa C++:

```
cout << “Algoritma” << end1;
```

Ketiga pernyataan di atas bertujuan sama, yaitu menuliskan teks ‘Algoritma’ ke keluaran standar (*standard output*), biasanya ke layar komputer. Ketiga perintah sederhana di atas sebenarnya terdiri dari ratusan pernyataan assembly. Perangkat lunak yang menerjemahkan program ke dalam bahasa *assembly* atau mesin ada dua macam yaitu :

a) Interpreter.

   Interpreter menerjemahkan program baris per baris, artinya jika suatu baris akan dieksekusi, maka baris tersebut baru diterjemahkan terlebih dulu ke bahasa mesin. Apabila baris berikutnya akan dieksekusi, maka baris tersebut baru diterjemahkan ke dalam bahasa mesin. Contoh bahasa pemrograman yang menggunakan interpreter adalah Basic.

b) Kompiler

   Kompiler menerjemahkan semua perintah ke dalam bahasa mesin kemudian menjalankan hasil penerjemahan. Hasil penerjemahan ini dapat disimpan di dalam file atau memori. Contoh bahasa pemrograman yang menggunakan kompiler adalah Pascal, C dan C++.

Perkembangan bahasa pemrograman sampai pada bahasa generasi keempat atau disingkat 4GL (*fourth-generation language*). Bahasa ini banyak digunakan untuk mengembangkan aplikasi basis data (*database*). Salah satu contohnya adalah SQL (*Structured Query Language*). Pada bahasa ini, perintah-perintah yang digunakan lebih manusiawi, misalnya “SELECT Nama, Alamat FROM Karyawan” untuk mengambil data Nama dan Alamat dari basis data Karyawan.

1.1.1. PERKEMBANGAN PASCAL

Pascal adalah bahasa tingkat tinggi (*high level language*) yang orientasinya pada segala tujuan, dirancang oleh Profesor Niklaus Wirth dari Technical University di Zurich, Switzerland. Nama Pascal diambil sebagai penghargaan terhadap Blaise Pascal, ahli matematika dan filosofi terkenal abad 17 dari Perancis.

Profesor Niklaus Wirth memperkenalkan kompiler bahasa Pascal pertama kali untuk komputer CDC 6000 (*Control Data Corporation*) yang dipublikasikan pada tahun 1971 dengan tujuan untuk membantu mengajar program komputer secara sistematis, khususnya untuk memperkenalkan pemrograman yang terstruktur (*structured programming*). Jadi Pascal adalah bahasa yang ditujukan untuk membuat program terstruktur.

Standar Pascal adalah bahasa Pascal yang didefinisikan oleh K. Jensen dan Niklaus Wirth. Standar Pascal di Eropa didefinisikan oleh ISO (*International Standards Organization*) dan di Amerika oleh kerjasama antara ANSI (*American National Standard Institute*) dengan IEEE (*Institute of Electrical and Electronic Engineers*).

Beberapa versi dari Pascal telah beredar di pasaran, diantaranya UCSD Pascal (University of California at San Diego Pascal), MS-Pascal (Microsoft Pascal), Apple Pascal, Turbo Pascal dan lain-lain. Turbo Pascal adalah copyright oleh Borland Inc. dan dapat digunakan pada sistem operasi PC-DOS, MS-DOS, CPM-86 dan CP/M-80.

1.2. LANGKAH-LANGKAH DALAM PEMROGRAMAN KOMPUTER

Dalam pemrograman komputer, kita memerlukan beberapa langkah :

1) Mendefinisikan masalah.

   Tentukan masalahnya seperti apa, kemudian apa saja yang harus dipecahkan dengan komputer, yang terakhir adalah apa masukannya dan bagaimana keluarannya.

2) Menentukan solusi.

   Setelah masalah didefinisikan dengan jelas, masukan apa yang diberikan sudah jelas, keluaran apa yang diinginkan sudah jelas, langkah selanjutnya adalah mencari jalan bagaimana masalah tersebut diselesaikan. Apabila permasalahan terlalu kompleks, biasanya kita harus membaginya ke dalam beberapa modul kecil agar lebih mudah diselesaikan.Hal semacam ini disebut subrutin.

3) Memilih algoritma.

   Langkah ini merupakan salah satu langkah penting dalam pemrograman komputer. Karena pemilihan algoritma yang salah akan menyebabkan program memiliki unjuk kerja yang kurang baik.

4) Menulis program.

Untuk menulis program kita dapat menggunakan salah satu bahasa generasi ketiga.

5) Menguji program.

   Setelah program selesai ditulis, kita harus mengujinya. Pengujian pertama adalah apakah program berhasil dikompilasi dengan baik? Pengujian berikutnya apakah program dapat menampilkan keluaran yang diinginkan?

   Kita juga harus menguji program dengan banyak kasus yang lain. Sering terjadi, suatu program berjalan baik untuk kasus A, B, C; tetapi menghasilkan sesuatu yang tidak diinginkan untuk kasus X,Y dan Z.

   Langkah keempat dan kelima ini dapat dilakukan berulang-ulang sampai program diyakini benar-benar berjalan sesuai yang diharapkan.

6) Menulis dokumentasi.

   Hal ini biasanya dilakukan bersamaan dengan menulis program, artinya pada setiap baris program atau setiap beberapa baris program, kita menambahkan komentar yang menjelaskan kegunaan dari suatu pernyataan.

   Dokumentasi ini fungsinya penting sekali apabila kita perlu melakukan perubahan atau perbaikan terhadap suatu program. Dokumentasi yang dijadikan satu dalam program, berupa komentar-komentar pendek, biasanya sudah cukup. Namun akan lebih baik jika kita juga menuliskannya dalam dokumen tersendiri kemudian mencetaknya di atas kertas.

7) Merawat program.

   Langkah ini dilakukan setelah program selesai dibuat dan sudah digunakan oleh pengguna kita. Hal yang paling sering terjadi di sini adalah munculnya bug yang sebelumnya tidak terdeteksi. Atau mungkin juga pengguna ingin tambahan suatu fasilitas baru. Apabila hal-hal seperti ini terjadi, kita harus melakukan revisi terhadap program kita.

1.3. DEFINISI ALGORITMA

Definisi algoritma menurut Microsoft Book-shelf adalah urutan langkah berhingga untuk memecahkan masalah logika atau matematika.

Dalam kehidupan sehari-hari, sebenarnya kita juga menggunakan algoritma untuk melakukan sesuatu. Sebagai contoh, kita ingin menulis surat, maka kita perlu melakukan beberapa langkah berikut :

1) Mempersiapkan kertas dan amplop.

2) Mempersiapkan alat tulis, seperti pena atau pensil.
3) Mulai menulis.
4) Memasukkan kertas ke dalam amplop.
5) Pergi ke kantor pos untuk mengeposkan surat tersebut.

Langkah-langkah dari nomor 1 sampai dengan nomor 5 di atas disebut dengan algoritma. Jadi sebenarnya kita sendiri juga sudah menggunakan algoritma baik sadar maupun tidak sadar.

Dalam banyak kasus, algoritma yang kita lakukan tidak selalu berurutan seperti di atas. Kadang-kadang kita harus memilih dua atau beberapa pilihan. Sebagai contoh, jika kita ingin makan, kita harus menentukan akan makan di rumah makan atau memasak sendiri. Jika kita memilih untuk makan di rumah makan, kita akan menjalankan algoritma yang berbeda dengan jika kita memilih memasak sendiri. Dalam dunia algoritma, hal semacam ini disebut percabangan.

Dalam kasus lain lagi, kita mungkin harus melakukan langkah-langkah tertentu beberapa kali. Sebagai contoh, saat kita menulis surat, sebelum memasukkan kertas ke dalam amplop, kita harus mengecek apakah surat tersebut sudah benar atau belum. Jika belum benar, berarti kita harus mempersiapkan kertas baru dan menulis lagi. Demikian seterusnya sampai surat kita sesuai dengan yang diharapkan. Dalam dunia algoritma, hal semacam ini disebut pengulangan.

1.4. CONTOH ALGORITMA

Berikut ini adalah contoh algoritma untuk memecahkan masalah matematika. Misalkan kita ingin menghitung luas lingkaran dari masukan berupa jari-jari lingkaran. Rumus luas lingkaran adalah :

$$L = \pi.R^2$$

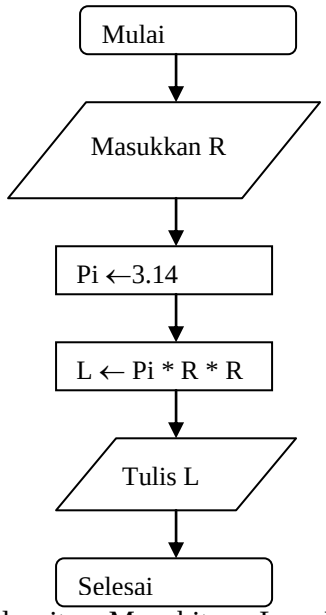
Berikut ini adalah contoh algoritma untuk menghitung luas lingkaran :

1. Masukkan R.
2. Pi ← 3.14.
3. L ← Pi * R * R.
4. Tulis L.

Perhatikan terdapat tanda ← pada baris kedua dan ketiga. Tanda ini berarti nilai di sebelah kanan diberikan pada operan di sebelah kiri. Sebagai contoh, untuk baris kedua, nilai 3.14 diberikan pada variabel Pi. Berikutnya, nilai Pi * R * R diberikan pada variabel L.

Baris pertama dari algoritma di atas meminta masukan dari pengguna berupa jari-jari lingkaran yang disimpan pada variabel R. Pada baris kedua nilai π disimpan pada variabel dengan nama Pi. Baris ketiga menghitung luas lingkaran dengan rumus πR^2 atau yang dituliskan Pi * R * R. Luas lingkaran ini disimpan pada variabel L. Baris terakhir menuliskan luas lingkaran tersebut.

Untuk memudahkan memahami algoritma, biasanya digambarkan suatu diagram alir (*flow chart*). Gambar 1.1 adalah contoh diagram alir untuk algoritma menghitung luas lingkaran.



Gambar 1.1. *Flowchart* untuk Algoritma Menghitung Luas Lingkaran

Algoritma sebenarnya digunakan untuk membantu kita dalam mengkonversikan suatu permasalahan ke dalam bahasa komputer. Oleh karena itu, banyak pemrogram yang sudah berpengalaman tidak pernah menuliskan algoritma di atas kertas lagi. Artinya dia menuliskan algoritma itu di dalam kepalanya dan dia langsung memprogram berdasarkan algoritma di kepalanya itu.

BAB II
STRUKTUR PROGRAM PASCAL

Struktur dari suatu program Pascal terdiri dari sebuah judul program (*program heading*) dan suatu blok program (program block) atau badan program (*body program*). Blok program dibagi lagi menjadi dua bagian, yaitu bagian deklarasi (*declaration part*) dan bagian pernyataan (*statement part*). Bagian deklarasi dapat terdiri dari deklarasi label (*labels declaration*), definisi konstanta (*constants definition*), definisi tipe (*type definition*), deklarasi variabel (*variables declaration*), deklarasi prosedur (*procedures declaration*) dan deklarasi fungsi (*function declaration*).

2.1. JUDUL PROGRAM

Di Turbo Pascal, judul program sifatnya adalah optional dan tidak signifikan di dalam program. Jika ditulis dapat digunakan untuk memberi nama program dan daftar dari parameter tentang komunikasi program dengan lingkungannya yang sifatnya sebagai dokumentasi saja. Judul program bila ditulis, harus terletak pada awal dari program dan diakhiri dengan titik koma.

Contoh:

```
program Contoh;
begin
        writeln ('Saya Pascal'); {diakhiri dengan titik koma}
        writeln ('----------');
end.
```

Jadi judul program sifatnya sebagai dokumentasi saja, tidak signifikan terhadap proses program. Misalnya program kita hanya menampilkan hasil saja, tetapi tidak meminta data input, pada judul program dapat didokumentasikan dengan ditulis sebagai berikut:

```
program contoh (output);
```

atau

```
program contoh (layar);
```

Parameter tentang komunikasi program dengan lingkungan ditulis di dalam tanda kurung buka dan tutup dan dapat ditulis apapun dengan dipisahkan oleh koma.

Contoh-contoh penulisan judul program yang benar:

```
program Gaji (input, output);
program Laporan (layar, cakram, printer, plotter);
```

2.2. BAGIAN DEKLARASI

Bagian deklarasi digunakan bila di dalam program kita menggunakan *identifier*. *Identifier* dapat berupa label, konstanta, tipe, variabel, prosedur dan fungsi. Jika suatu program menggunakan *identifier*, Pascal menuntut agar *identifier* tersebut dikenalkan terlebih dahulu sebelum digunakan, yaitu dideklarasikan terlebih dahulu pada bagian ini.

2.2.1. DEKLARASI LABEL

Jika program kita menggunakan pernyataan GOTO untuk meloncat ke suatu pernyataan yang tertentu, maka dibutuhkan suatu label pada pernyataan yang dituju dan label tersebut harus dideklarasikan terlebih dahulu pada bagian deklarasi. Mendeklarasikan label diawali dengan kata cadangan LABEL diikuti oleh kumpulan identifier label dengan dipisahkan oleh koma dan diakhiri dengan titik koma.

Contoh:

```
program Contoh_Label (layar);
label
        100, selesai; {identifier label}
begin
        writeln (‘Bahasa’);
        goto 100;
        writeln (‘BASIC’);
        writeln (‘COBOL’);
100: {label yang dituju}
        writeln (,Pascal’);
        goto selesai;
        writeln (‘FORTRAN’);
selesai: {label yang dituju}
end.
```

Bila program ini dijalankan, akan didapat hasil:

```
Bahasa
Pascal
```

Standar Pascal hanya mengizinkan label dalam bentuk angka maksimum panjangnya 4 digit, sedangkan Turbo Pascal dapat berupa angka maupun *identifier*.

2.2.2. DEFINISI KONSTANTA

Bila kita ingin menggunakan identifier yang berisi nilai-nilai konstanta, maka harus didefinisikan terlebih dahulu pada bagian ini. Definisi konstanta diawali dengan kata cadangan CONST diikuti oleh kumpulan *identifier* yang diberi suatu nilai konstanta.

Contoh:

```
program Contoh_Konstanta (layar);
const
        Potongan = 0.2;
        Gaji = 25000;
        NamaPerusahaan = 'P.T. Lali Jiwa';
begin
        writeln ('Potongan        = ', Potongan);
        writeln ('Gaji            = ', Gaji);
        writeln ('Nama   = ', NamaPerusahaan);
end.
```

Bila program ini dijalankan, akan didapat hasil:

```
Potongan        = 2.0000000000E-01
Gaji            = 25000
Nama            = P.T. Lali Jiwa
```

Turbo Pascal mempunyai beberapa konstanta yang tidak perlu didefinisikan, dapat dipergunakan langsung, yaitu:

PI dengan tipe real, bernilai 3.1415926536

FALSE dengan tipe Boolean, bernilai logika salah

TRUE dengan tipe Boolean, bernilai logika benar

MAXINT dengan tipe numeric integer, bernilai 32767

Contoh:

```
program Contoh_Konstanta_Terdefinisi (layar);
begin
        writeln ('Nilai pi                      = ', Pi);
        writeln ('Nilai logika benar                    = ', True);
        writeln ('Nilai logika salah                    = ', False);
        writeln ('Nilai integer maksimum = ', Maxint);
end.
```

Bila program ini dijalankan, akan didapatkan hasil:

```
Nilai pi                = 3.1415926536E+00
Nilai logika benar      = TRUE
Nilai logika salah      = FALSE
Nilai integer maksimum  = 32767
```

2.2.3. DEFINISI TIPE

Suatu data yang dipergunakan di program Pascal, identifier yang digunakan untuk data tersebut harus disebutkan tipenya, yaitu dengan cara mendefinisikannya di bagian definisi ini. Pascal menyediakan beberapa macam tipe data terdiri dari:

a. Data tipe sederhana (*simple type data*)

Data ini menghubungkan sebuah identifier dengan sebuah data. *Simple type data* dapat digolongkan menjadi tipe data standar (*standard data types*) dan tipe data yang didefinisikan oleh pemakai (*user defined data type*). Yang termasuk tipe data standar adalah data dengan tipe integer, real, char, string dan Boolean. Yang termasuk *user defined data type* adalah *enumerated* atau *scalar* (sejumlah data disebutkan satu per satu) dan sub range (range dari data).

b. Data tipe terstruktur (*structured type data*)

   Data ini terdiri dari beberapa data item yang dihubungkan satu dengan lainnya. Masing-masing grup dari data item dihubungkan dengan suatu identifier tertentu. Ada 4 macam tipe dari data terstruktur, yaitu array, record, file dan set.

c. Data tipe penunjuk (*pointer data type*)

   Data ini digunakan untuk membuat data terstruktur tipe dinamik.

Pembahasan mengenai tipe data ini disajikan pada bab IV.

Untuk mendefinisikan tipe dari data yang akan dipergunakan di dalam program, kata cadangan TYPE digunakan sebagai judul definisi tipe dan diikuti oleh satu atau lebih definisi tipe yang dipisahkan dengan titik koma.

Contoh mendefinisikan tipe data sederhana:

```
program Contoh_Definisi_tipe_Data Standar;
type
        Nomor = integer;
        Lagi = Boolean;
        NilaiUjian = real;
        NilaiHuruf = char;
        Nama = string [30];
        Alamat = string [35];
begin
end.
```

Dari definisi tipe tersebut, berarti identifier yang akan digunakan di dalam program adalah:

- Nomor dengan tipe integer
- Lagi dengan tipe Boolean (logika)
- NilaiUjian dengan tipe numeric real
- NilaiHuruf dengan tipe char panjang maksimum 1 karakter
- Nilai dengan tipe string panjang maksimum 30 karakter

- Alamat dengan tipe string panjang maksimum 35 karakter

2.2.4. DEKLARASI VARIABEL

Jika konstanta merupakan identifier berisi data konstanta yang nilainya sudah ditentukan dan pasti, tidak dapat diubah di dalam program, maka variabel adalah *identifier* yang berisi data yang dapat berubah-ubah nilainya di dalam program. Setiap variabel di dalam program Pascal harus dideklarasikan sebelum digunakan. Kata cadangan VAR digunakan sebagai judul di dalam bagian deklarasi variabel dan diikuti oleh satu atau lebih *identifier* yang dipisahkan koma, diikuti dengan titik dua dan tipe dari datanya serta diakhiri dengan titik koma.

Contoh:

```
var
        Total, Gaji, Tunjangan : real;
        Menikah : Boolean;
        JumlahAnak : integer;
        Keterangan : string [25];
begin
        Gaji := 50000;
        Menikah := True;
        JumlahAnak := 3;
        Tunjangan := 0.25 * Gaji + JumlahAnak * 30000;
        Total := Gaji + Tunjangan;
        Keterangan := 'Karyawan Teladan';
        writeln ('Gaji bulanan     : Rp ', Gaji);
        writeln ('Tunjangan              : Rp ' , Tunjangan);
        writeln ('Total Gaji             : Rp ', Total);
        writeln ('Sudah menikah : ', Menikah);
        writeln ('Jumlah Anak    : ', JumlahAnak);
        writeln ('Keterangan             :', Keterangan);
end.
```

Bila program ini dijalankan, akan didapat hasil:

```
Gaji bulanan          : Rp 5.0000000000E+04
Tunjangan             : Rp 1.0250000000E+05
Total Gaji            : Rp 1.5250000000E+05
Sudah menikah : TRUE
Jumlah Anak           : 3
Keterangan            : Karyawan Teladan
```

Hubungan antara bagian deklarasi variabel dengan bagian definisi tipe adalah *identifier* yang didefinisikan di bagian definisi tipe dapat digunakan sebagai tipe di deklarasi variabel.

Contoh:

```
type
        Pecahan = real;
        Logika = Boolean;
        Huruf = string [25];
        Bulat = integer;
var
        Total, Gaji, Tunjangan : Pecahan;
        Menikah : Logika;
        JumlahAnak : Bulat;
        Keterangan : Huruf;
begin
        Gaji := 50000;
        Menikah := True;
        JumlahAnak := 3;
        Tunjangan := 0.25 * Gaji + JumlahAnak * 30000;
        Total := Gaji + Tunjangan;
        Keterangan := 'Karyawan Teladan';
        writeln ('Gaji bulanan     : Rp ', Gaji);
        writeln ('Tunjangan              : Rp ' , Tunjangan);
        writeln ('Total Gaji             : Rp ', Total);
        writeln ('Sudah menikah: ', Menikah);
        writeln ('Jumlah Anak    : ', JumlahAnak);
        writeln ('Keterangan             :', Keterangan);
end.
```

2.2.5. DEKLARASI PROSEDUR

Pembahasan mengenai prosedur disajikan pada sub bab 7.1.

2.2.6. DEKLARASI FUNGSI

Pembahasan mengenai fungsi disajikan pada sub bab 7.2.

2.3. BAGIAN PERNYATAAN

Suatu program Pascal yang paling sederhana adalah program yang hanya terdiri dari sebuah pernyataan saja. Bagian pernyataan (*statement part*) merupakan bagian yang terakhir dari suatu blok. Bagian diawali dengan kata cadangan (*reserved word*) BEGIN dan diakhiri dengan kata cadangan END.

Jadi suatu program Pascal yang sederhana dapat berbentuk:

```
begin
        writeln ('Saya Pascal');
end.
```

Bila program ini dijalankan, maka akan ditampilkan tulisan di layar sebagai berikut:

```
Saya Pascal
```

Bagian pernyataan ini menunjukkan suatu tindakan yang akan dikerjakan oleh komputer. Tindakan yang dilakukan oleh program tergantung dari instruksi-instruksi yang diberikan. Pernyataan atau statemen (*statement*) yang merupakan instruksi program. Pernyataan-pernyattan yang akan diberikan untuk dikerjakan oleh komputer ditulis antara kata cadang BEGIN dan END. Akhir penulisan END diakhiri dengan tanda titik.

Bentuk umum dari bagian pernyataan ini adalah sebagai berikut:

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

Contoh:

```
begin
        writeln (‘Saya Pascal’); {diakhiri dengan titik koma}
        writeln (lst, ’----------‘);
end.
```

Bila program ini dijalankan, maka akan ditampilkan tulisan di layar sebagai berikut:

```
Saya Pascal
----------
```

Jika program Pascal kita ingin ditampilkan di printer, maka kita dapat menyebutkan nama printer tersebut, yaitu LST.

Contoh:

```
begin
        writeln (lst, ‘Saya Pascal’); {lst merupakan nama dari printer, menampilkan hasil di
printer}
        writeln (lst, ’----------‘);
end.
```

Bila program ini dijalankan, maka akan ditampilkan tulisan di printer sebagai berikut:

Saya Pascal

Jadi, jika nama alat tidak disebutkan, maka tampilan akan dilakukan di layar, sedangkan jika ingin ditampilkan di printer, maka nama alat printer harus disebutkan.

Program Pascal tidak mengenal aturan penulisan di kolom tertentu, jadi dapat ditulis mulai kolom berapapun. Penulisan statemen-statemen pada contoh program yang menjorok masuk beberapa kolom tidak mempunyai pengaruh di proses, hanya dimaksudkan supaya mempermudah pembacaan program, sehingga akan lebih terlihat bagian-bagiannya.

Contoh:

```
                begin
writeln (‘Saya Pascal’);
        writeln (‘----------‘);
end.
```

Penulisan seperti di ataspun boleh, bahkan dapat disambung dalam satu baris, sebagai berikut:

```
begin writeln (‘Saya Pascal’); writeln (‘----------‘); end.
```

Akan tetapi penulisan seperti itu tidaklah dianjurkan dan sedapat mungkin dihindari, karena tidak baik untuk dokumentasi program dan sulit untuk membacanya.

BAB III
ELEMEN DARI PROGRAM PASCAL

Untuk mempelajari suatu bahasa komputer agar dapat membuat program dengan benar dan dapat mengembangkannya, langkah pertama kita harus mengetahui terlebih dahulu struktur dari program yang akan dibuat dengan bahasa tersebut. Selanjutnya langkah kedua adalah kita harus mengetahui terlebih dahulu elemen-elemen yang membentuk program tersebut.

3.1. SIMBOL-SIMBOL DASAR (*BASIC SYMBOLS*)

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

Huruf-huruf: A sampai dengan Z, a sampai dengan z dan _ (garis bawah). Huruf besar (*uppercase*) dan huruf kecil (*lower case*) dianggap sama, tidak dibedakan.

Angka-angka: 0 1 2 3 4 5 6 7 8 9 (0 sampai dengan 9)

Simbol-simbol khusus: + - * / = ^ < > ( ) [ ] . , : ; ‘ $

3.2. KATA-KATA CADANGAN (*RESERVED WORDS*)

Kata-kata cadangan (*reserved words*) adalah kata-kata yang sudah didefinisikan oleh Pascal yang mempunyai maksud tertentu. Kata-kata cadangan tidak boleh didefinisikan ulang oleh pemakai, sehingga tidak dapat digunakan sebagai pengenal (*identifier*). Kata-kata cadangan yang dipergunakan dalam program Pascal disajikan dalam Tabel 3.1.

Tabel 3.1. Kata-kata Cadangan dalam Program Pascal

Absolute* And	External* File	Nil Not	Shl* Shr*

Array	Forward	Overlay*	String
Begin	For	Of	Then
Case	Function	Or	Type
Const	Goto	Packed	To
Div	Inline*	Procedure	Until
Do	If	Program	Var
Downto	In	Record	While
Else	Label	Repeat	With
End	Mod	Set	Xor*

* : kata-kata cadangan yang tidak terdapat di standar Pascal

3.3. PENGENAL STANDAR (*STANDARD IDENTIFIER*)

Turbo Pascal telah mendefinisikan sejumlah pengenal untuk pengenal tipe, pengenal konstanta, pengenal variabel, pengenal prosedur dan pengenal fungsi. Pengenal-pengenal standar ini dapat didefinisikan ulang oleh pemakai, tetapi akan berakibat hilangnya fasilitas yang diberikan, terganti dengan maksud baru yang didefinisikan. Pengenal-pengenal standar dalam program Pascal disajikan dalam tabel 3.2.

Tabel 3.2. Pengenal-pengenal Standar dalam Program Pascal

Addr	Delay	Length	Release
ArcTan	Delete	Ln	Rename
Assign	EOF	Lo	Reset
Aux	EOLN	LowVideo	Rewrite
AuxInPtr	Erase	Lst	Round
AuxOutPtr	Execute	LstOutPtr	Seek
BlockRead	Exit	Mark	Sin
BlockWrite	Exp	MaxInt	SizeOf
Boolean	False	Mem	SeekEof
Buflen	FilePos	MemAvail	SeekEoln
Byte	FileSize	Move	Sqr
Chain	FillChar	New	Sqet
Char	Flush	NormVideo	Str
Chr	Frac	Odd	Succ
Close	GetMem	Ord	Swap
ClrEOL	GotoXY	Output	Text
ClrScr	Halt	Pi	Trm
Con	Heap Ptr	Port	True
ConInPtr	Hi	Pos	Trunc
ConOutPtr	Ioresult	Pred	UpCase
ConCat	Input	Ptr	Usr
ConstPtr	Insline	Random	UsrInPtr
Copy	Insert	Randomize	UsrOutPtr
Cos	Int	Read	Val
CrtExit	Integer	Readln	Write
DelLine	KeyPressed		

3.4. PENGENAL DIDEFINISIKAN OLEH PEMAKAI (*USER DEFINED IDENTIFIER*)

Pemakai dapat mendefinisikan sendiri pengenal untuk pengenal label, pengenal tipe,

pengenal konstanta, pengenal variabel, pengenal prosedur dan pengenal fungsi. Pengenal yang didefinisikan sendiri ini bebas, tetapi dengan ketentuan-ketentuan sebagai berikut:

a. Terdiri dari gabungan huruf dan angka dengan karakter pertama harus berupa huruf.
b. Tidak boleh mengandung *blank.*
c. Tidak boleh mengandung simbol-simbol khusus, kecuali garis bawah.
d. Panjangnya maksimum 127 karakter.

Contoh:

Pengenal	Keterangan
NamaLangganan	Benar
No_Induk	Benar
P3K	Benar
5X	Salah, karakter pertama harus huruf
X&Y	Salah, tidak boleh mengandung simbol khusus
X Y	Salah, tidak boleh mengandung *blank*

3.5. KONSTANTA (*CONSTANT*)

Konstanta di dalam bahasa Pascal dapat berupa konstanta numerik integer, konstanta numerik real dan konstanta string. Konstanta akan disimpan di pengenal konstanta dan nilainya tidak berubah di dalam program.

3.5.1. KONSTANTA NUMERIK INTEGER

Konstanta numerik integer merupakan nilai bilangan bulat baik dalam bentuk desimal maupun hexadesimal. Memori yang digunakan untuk menyimpan konstanta ini adalah 2 byte. Nilai integer desimal mempunyai jangkauan antara –32768 sampai dengan 32767 dan nilai integer hexadesimal diawali dengan tanda dollar ($), mempunyai jangkauan antara $0000 sampai dengan $FFFF.

Contoh:

```
const
        Nilai1 = 2741;
        Nilai2 = $AB5;
begin
        writeln ('Nilai1 = ', Nilai1);
        writeln ('Nilai2 = ', Nilai2);
end.
```

Bila program ini dijalankan, akan didapat hasil:

```
Nilai1 = 2741
Nilai2 = 2741
```

3.5.2. KONSTANTA NUMERIK REAL

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

Contoh nilai konstanta numerik real:

```
123.45
12345.
12E5
12E+5
-12.34
1.23E-2
  .1234 {salah, titik desimal tidak boleh dimuka}
0.1234
```

Contoh:

```
const
        Nilai1 = 123.45;
        Nilai2 = 123.;
begin
        writeln ('Nilai1 = ', Nilai1);
        writeln ('Nilai2 = ', Nilai2);
end.
```

Bila program ini dijalankan, akan didapat hasil:

```
Nilai1 = 1.2345000000E+02
Nilai2 = 1.2300000000E+02
```

3.5.3. KONSTANTA STRING

Nilai konstanta string merupakan urut-urutan dari karakter yang terletak di antara tanda petik tunggal. Bila karakter petik merupakan bagian dari konstanta string, maka dapat dituliskan dengan menggunakan dua buah petik tunggal berurutan. Nilai konstanta string akan menempati memori sebesar banyaknya karakter stringnya.

Contoh:

```
const
        Tanggal         = '18-02-1987';
        Hari            = 'Jum' 'at';
        Pasaran         = 'Legi';
        jarak           = ' ';
        kosong          = ' ';
begin
```

```
        writeln (Tanggal, jarak, Hari, ' ', kosong, Pasaran);
end.
```

Bila program ini dijalankan, akan didapat hasil:

```
18-02-1987    Jum'at  Legi
```

3.6. KARAKTER KONTROL (*CONTROL CHARACTER*)

Turbo Pascal memungkinkan karakter-karakter kontrol untuk dilekatkan ke dalam suatu string. Dua buah notasi digunakan untuk karakter kontrol, yaitu:

1. Simbol # yang diikuti oleh suatu nilai integer di antara 0 sampai dengan 255, baik berupa nilai desimal maupun hexadesimal untuk menunjukkan suatu karakter yang dihubungkan dengan kode ASCII. Untuk lebih lengkapnya tabel ASCII dapat dilihat pada Lampiran A.

   Contoh:

   ```
   #7          (ASCII 7 adalah bel)
   #65 (ASCII 65 adalah karakter A)
   #13 (ASCII 13 adalah carriage return)
   ```

   Contoh:

   ```
   begin
       Writeln (#83#65#89#65#32#80#65#83#67#65#76);
   end.
   ```

   Bila program dijalankan, akan didapat hasil:

   ```
   SAYA PASCAL
   ```

2. Simbol ^ diikuti oleh suatu karakter, menunjukkan hubungannya dengan karakter kontrol.

   Contoh:

   ```
   ^ G (Control G berarti bel, sama dengan ASCII 7)
   ^ M (Control M berarti carriage return, sama dengan ASCII 13)
   ^ [   (Control [ berarti escape)
   ```

   Contoh:

   ```
   begin
       Writeln ('Apakah Anda mendengar bunyi bel  2 kali ?', ^g#7);
   ```

```
end.
```

Bila program dijalankan, akan didapat hasil tulisan dan bel dua kali sebagai berikut:

```
Apakah Anda mendengar bunyi bel 2 kali ?
```

3.7. TANDA OPERASI (*OPERATOR*) DAN UNGKAPAN (*EXPRESSION*)

Pembahasan mengenai tanda operasi (*operator*) di dalam bahasa Pascal disajikan pada sub bab 4.2. Gabungan dari operator, variabel, konstanta dan atau fungsi membentuk suatu ungkapan (*expression*).

3.8. KOMENTAR PROGRAM (*COMMENT*)

Untuk keperluan dokumentasi program, sehingga program mudah dibaca dan dipahami dan dapat ditambahkan komentar-komentar di dalam program yang tidak akan mempengaruhi proses dari program. Komentar program dapat diletakkan di manapun di dalam program. Suatu komentar ditulis dengan pembatas [dan] atau pembatas (* dan *).

Contoh:

```
{Ini adalah komentar program untuk memberi keterangan atau penjelasan program, sehingga baik untuk dokumentasi}
begin
end.
```

Penulisan komentar tersebut dapat juga digunakan pembatas (* dan *) sebagai berikut:

```
(*Ini adalah komentar program untuk memberi keterangan atau penjelasan program, sehingga baik untuk dokumentasi*)
begin
end.
```

3.9. PERINTAH KOMPILER (*COMPILER DIRECTIVE*)

Perintah kompiler (*compiler directive*) merupakan bentuk khusus dari Turbo Pascal yang digunakan untuk menghubungi kompiler Pascal secara langsung. Perintah

kompiler ditulis seperti komentar program, yaitu di antara pembatas { dan } atau pembatas (* dan *) dan diawali dengan tanda $.
Contoh perintah kompiler:

```
{$I-}
```
Pemberitahuan kepada kompiler tentang kesalahan I/O tidak diseleksi oleh kompiler.

3.10. STATEMEN (*STATEMENT*)

Statemen adalah perintah pengerjaan program. Kumpulan dari statemen membentuk suatu program. Bila suatu program Pascal tidak mengandung statemen atau disebut dengan statemen kosong (*empty statement*), maka tidak akan ada yang dikerjakan oleh program. Statemen terletak di bagian deklarasi statemen dengan diawali kata cadangan BEGIN dan ditutup dengan kata cadangan END. Masing-masing akhir dari statemen diakhiri dengan titik koma. Di dalam bahasa Pascal statemen dapat berupa statemen sederhana (*simple statement*) dan statemen terstruktur (*structured statement*).

3.10.1. STATEMEN SEDERHANA

Statemen sederhana adalah statemen yang tidak mengandung statemen yang lainnya. Yang termasuk statemen sederhana adalah:

1. Statemen pengerjaan (*assignment statement*).

   Statemen pengerjaan (*assignment statement*) merupakan statemen yang paling mendasar dari semua statement yang ada. Statemen ini digunakan untuk memasukkan suatu nilai tertentu ke dalam suatu variabel. Bentuk dari statemen ini terdiri dari pengenal variabel diikuti oleh operator := dan suatu ungkapan, sebagai berikut:

   ```
   pengenal_variabel := ungkapan;
   ```

   Contoh:

   ```
   Nilai := 5;
   Nilai := Nilai + 1;
   D :+ B*B – 4*A*C;
   ```

```
Lulus := (Nilai > 65.5);
Benar := True;
Lolos := (Kode = Sandi);
```

2. Statemen prosedur (*procedure statement*).

   Statemen prosedur digunakan untuk mengaktifkan suatu prosedur yang telah didefinisikan oleh pemakai (lihat sub bab 7.1) atau prosedur standar yang telah didefinisikan oleh Pascal.

   Contoh:

```
Cari (Nama, Alamat);
Tambah (2, 3, z);
Urutkan (Nilai);
Writeln (Fahrenheit);
```

3. Statemen GOTO.

   Statemen GOTO diawali dengan kata cadang GOTO yang diikuti oleh pengenal label. Pengenal label yang dipergunakan harus sudah dideklarasikan di bagian deklarasi label (lihat sub bab 2.2.1).

4. Statemen kosong (*empty statement*).

   Statemen kosong merupakan statemen yang tidak berisi symbol apapun (sebenarnya tidak mengandung statemen apapun) dan tidak mempunyai efek terhadap program.

   Contoh:

```
begin
end.
```

3.10.2. STATEMEN TERSTRUKTUR

Statemen terstruktur (*structured statement*) merupakan statemen yang dibentuk dari komposisi beberapa statemen. Statemen terstruktur dapat berupa:

1. Statemen jamak (*compound statement*).

   Statemen jamak digunakan jika lebih dari sebuah statemen harus dikerjakan, sedang bentuk umum dari Pascal hanya memungkinkan sebuah statemen saja yang disebutkan. Statemen jamak ditulis di dalam kata cadangan BEGIN dan END yang tersendiri.

   Contoh:

```
begin
```

```
    Keterangan := ‘Lulus’;
    Komentar := ‘Memuaskan’;
    end.
```

2. Statemen penyeleksian kondisi (*conditional statement*).

   Pembahasan mengenai statemen penyeleksian kondisi disajikan pada bab V.

3. Statemen perulangan (*repetitive statement*).

   Pembahasan mengenai statemen perulangan disajikan pada bab VI.

BAB IV
TIPE DATA

Apabila kita mendeklarasikan variabel pada Pascal, kita harus menentukan tipenya. Tipe ini menentukan nilai yang dapat disimpan variabel tersebut dan operator-operator apa saja yang dapat dikenakan padanya. Sebagai contoh, tipe real hanya dapat menyimpan bilangan real, dan operator yang dapat dikenakan padanya antara lain operator penjumlahan, perkalian dan sebagainya.

4.1. MACAM-MACAM TIPE DATA

Macam-macam tipe data pada Turbo Pascal 7 disajikan pada tabel 4.1.

Tabel 4.1. Tipe Data

No.	Kelompok tipe data1	Kelompok tipe data2	Kelompok tipe data3
1.	Tipe sederhana	a. Tipe ordinal	1) Tipe bilangan bulat 2) Tipe boolean 3) Tipe karakter 4) Tipe terbilang 5) Tipe subjangkauan
		b. Tipe real	
2.	Tipe string		
3.	Tipe terstruktur	a) Tipe larik b) Tipe rekaman c) Tipe objek d) Tipe himpunan e) Tipeberkas	
4.	Tipe pointer		
5.	Tipe prosedural		
6.	Tipe objek		

Tidak semua tipe dalam tabel 4.1 akan dibahas dalam diktat kuliah ini karena memang jarang digunakan dalam pembahasan algoritma.

4.1.1. TIPE BILANGAN BULAT

Tipe bilangan bulat digunakan untuk menyimpan bilangan bulat. Tabel 4.2 menunjukkan macam-macam tipe bilangan bulat yang dimiliki Pascal.

Tabel 4.2. Macam-macam Tipe Bilangan Bulat pada Pascal

Tipe	Jangkauan	Ukuran
Shortint	128..127	8 bit
Integer	-32768..32767	16 bit
Longint	-2147483648..2147483647	32bit
Byte	0..255	8 bit
Word	0..65535	16 bit

Sebagai contoh, dua pernyataan berikut ini masing-masing mendeklarasikan dua variabel bertipe integer yaitu x dan y, serta satu variabel bertipe word, yaitu z.

```
var
    x,y: integer;
    z: word;
```

Untuk memberikan nilai pada tipe bilangan bulat, kita dapat menggunakan basis desimal maupun heksadesimal. Pemberian nilai dalam basis heksadesimal dilakukan dengan menambahkan $ di depan bilangannya. Contoh:

```
x := 16; {dengan desimal}
x :=  $0A; {dengan heksadesimal}
```

4.1.2. TIPE BOOLEAN

Tipe Boolean adalah tipe yang hanya dapat bernilai benar atau salah. Turbo Pascal 7 menyediakan empat macam tipe Boolean seperti terlihat pada tabel 4.3.

Tabel 4.3. Macam-macam Tipe Boolean pada Turbo Pascal 7.0

Tipe Data	Ukuran
Boolean	1 byte
ByteBool	1 byte
WordBool	2 byte (1 word)
LongBool	4 byte (2 word)

Dari empat tipe di atas, yang paling sering digunakan adalah tipe Boolean. Sebagai contoh, pernyataan berikut ini mendeklarasikan sebuah variabel bertipe Boolean dengan nama b1:

```
var
    b1: Boolean;
```

Ada dua macam nilai yang dapat diberikan pada tipe Boolean, yaitu *true* dan *false*. Contoh:

```
b1 := true;
b1 := false;
```

4.1.3. TIPE KARAKTER

Tipe karakter digunakan untuk menyimpan data alfanumeris, seperti ‘A’, ‘Z’, ‘@’, ‘1’, ‘9’ dan sebagainya. Tipe karakter dideklarasikan dengan kata kunci char. Contoh :

```
var
    ch: char;
```

Untuk memberikan nilai pada variabel bertipe karakter, kita dapat menggunakan beberapa cara, yaitu :

a) Menuliskan karakter di dalam tanda petik tunggal. Contoh :
   ch := “A’;
b) Menuliskan tanda # diikuti dengan nomor ASCII dari karakter yang ingin dituliskan. Contoh :
   ch := #65; {sama artinya dengan ch := ‘A’;}
c) Mengkorvesikan nomor ASCII ke karakter menggunakan fungsi chr. Contoh :
   ch := chr (65); {sama artinya dengan ch := ‘A’;}
   Kebalikan dari fungsi chr adalah ord, yang digunakan untuk mengembalikan nomor ASCII dari suatu karakter. Contoh:
   x := ord (‘A’); { x akan bernilai 65}
   Untuk lebih lengkapnya tabel ASCII dapat dilihat pada Lampiran A.

4.1.4. TIPE SUBJANGKAUAN

Tipe subjangkauan memungkinkan kita mendeklarasikan tipe yang berada pada jangkauan tertentu. Pada dasarnya tipe subjangkauan hampir sama dengan tipe bilangan bulat, bedanya kita bebas menentukan jangkauan dari tipe ini, misalnya dari 1 sampai 100.

Pendeklarasian tipe subjangkauan dilakukan dengan menuliskan batas bawah dan batas atas dari jangkauannya. Sebagai contoh:

```
type
  Bulan = 1..12;
```

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

```
var
  Januari: Bulan;
```

kita tidak bisa memberikan nilai kurang dari 1 atau lebih dari 12. Contoh:

```
  Januari := 1;
```

4.1.5. TIPE TERBILANG

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

```
type
  TipeHari = (Minggu, Senin, Selasa, Rabu, Kamis, Jumat, Sabtu);
```

memberi nama Minggu pada 0, Senin pada 1, Selasa pada 2 dan seterusnya.

Dengan pendeklarasian TipeHari seperti contoh di atas, kita tidak perlu menggunakan angka 0,1 sampai dengan 6 untuk merepresentasikan hari. Sebagai contoh, jika kita mempunyai variabel Hari yang bertipe TipeHari:

```
var
    Hari: TipeHari;
```

kita dapat menuliskan contoh pernyataan berikut:

```
Hari := Minggu;
Hari := Senin;
```

4.1.6. TIPE REAL

Tipe real digunakan untuk menyimpan bilangan real. Macam-mcam tipe real pada Pascal dapat dilihat pada tabel 4.4.

Tabel 4.4. Macam-macam Tipe Real pada Pascal

Tipe Data	Jangkauan	Digit Penting	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	-263+1..263-1	19-20	8 byte

Sebagai contoh dua pernyataan berikut ini masing-masing mendeklarasikan dua variabel bertipe real, yaitu x dan y, serta satu variabel bertipe double, yaitu z.

```
var
  x, y: real;
  z: double;
```

Pemberian nilai pada tipe real dapat dilakukan dengan dua cara, yaitu:

1. Menuliskan nilai dengan tanda titik tanpa eksponen. Contoh:

```
x := 123.45;
```

2. Menuliskan nilai dengan eksponen. Contoh:

```
x := 1.2345E+2;
```

4.1.7. TIPE STRING

Tipe string digunakan untuk menyimpan data yang berupa untaian karakter, seperti ‘Pascal’, ‘Algoritma’, ‘Turbo Pascal 7’ dan sebagainya. Untuk mendeklarasikan string, digunakan kata kunci string. Contoh:

```
var
  kalimat : string;
```

Pemberian nilai pada string dilakukan dengan meletakkan untaian karaker di antara tanda petik tunggal. Contoh:

```
Kalimat := ‘Turbo Pascal 7.0’;
```

Untuk memberikan nilai yang mengandung karakter ‘ pada suatu string, tulislah karakter ‘ sebanyak dua kali. Contoh:

```
kalimat := ‘Don’t smoke’;
```

4.1.8. TIPE LARIK

Tipe larik memungkinkan kita mendeklarasikan kumpulan variabel yang bertipe sama. Pendeklarasian larik harus mengikuti bentuk umum berikut:

```
var
   nama_larik: array [batas_bawah..batas_atas] of tipe_larik;
```

Sebagai contoh kita ingin membuat delapan variabel bertipe longint. Tanpa menggunakan larik, kita mungkin mendeklarasikan variabel tersebut dengan cara berikut:

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

Dengan larik, kita dapat menyederhanakan deklarasi kedelapan variabel di atas menjadi:

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

Dengan pendeklarasian ini, seolah-olah kita mempunyai delapan variabel, yaitu a[1], a[2], a[3], a[4], a[5], a[6], a[7] dan a[8]. Contoh:

```
a[1] := 10;
a[2] := 5;
```

Indeks larik tidak harus dimulai dari 1. Kita dapat memulainya dari angka berapa pun. Sebagai contoh:

```
var
  A: array [5..10] of longint;
```

mendeklarasikan larik dengan indeks mulai dari 5 sampai dengan 10. Dengan demikian elemen-elemen larik kita adalah A[5], A[6], A[7], A[8], A[9] dan A[10].

Kita juga dapat mendeklarasikan larik multi dimensi, yaitu larik yang memiliki dimensi lebih dari satu. Sebagai contoh :

```
var
  A; array [1..5, 1..5] of longint;
```

mendeklarasikan larik dua dimensi berukuran 5 x 5. Dengan demikian kita mempunyai 25 elemen, yaitu A[1,1], A[1,2], A[1,3], …, A[3,1], A[3,2], A[3,3], …, A[5,4], dan A[5,5]. Untuk lebih jelasnya lihat Gambar 4.1.

1	2	3	4	5

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

Gambar 4.1. Larik Dua Dimensi

Contoh:

```
A[1,1] := 10;
A[1,2] := 5;
A[5,5] := 20;
```

4.1.9. TIPE REKAMAN

Bila dengan larik, kita menggabungkan beberapa variabel bertipe sama; dengan rekaman kita menggabungkan beberapa variabel yang tipenya tidak harus sama. Untuk mendeklarasikan rekaman, digunakan kata kunci record. Contoh:

```
type
  Tkaryawan = record
        Nama : string;
        Alamat : string;
        Gaji : longint;
  end;
```

Pemberian nilai pada variabel bertipe rekaman dilakukan dengan menyebutkan nama variabel rekaman diikuti tanda titik dan variabel di dalam rekaman. Sebagai contoh, jika terdapat variabel Karyawan yang bertipe Tkaryawan:

```
var
  Karyawan : Tkaryawan:
```

Kita dapat menuliskan pernyataan berikut:

```
Karyawan.Nama := 'Turbo Pascal7.0';
Karyawan.Alamat := 'Borland/Inprise';
Karyawan.Gaji := 10000;
```

4.1.10. TIPE HIMPUNAN

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

```
type
  HimpunanKarakter = set of char;
```

mendeklarasikan tipe HimpunanKarakter sebagai himpunan dari karakter. Berikut ini adalah contoh variabel yang dideklarasikan berdasarkan tipe tersebut:

```
var
  Vokal: HimpunanKarakter;
  Huruf: HimpunanKarakter;
```

Pemberian nilai pada tipe himpunan dilakukan dengan menuliskan anggota himpunan dalam kurung siku ([dan]). Contoh:

```
Vokal := ['A', 'I', 'U', 'E', 'O'];
Huruf := ['A'..'Z'];
```

4.1.11. TIPE POINTER

Pointer adalah variabel yang menunjukkan lokasi memori tertentu. Pendeklarasian pointer dilakukan dengan cara yang hampir sama dengan pendeklarasian variabel biasa, bedanya kita harus menambahkan tanda ^ di depan tipe pointer. Contoh :

```
var
  P1: ^integer;
  P2: ^double;
```

Bila kita mempunyai dua buah variabel bertipe integer dan double seperti contoh berikut:

```
var
  I: integer;
  D: double;
```

Kita dapat menugaskan P1 agar menunjuk I dan P2 agar menunjuk D dengan pernyataan berikut:

```
P1 := @I;
P2 := @D;
```

Karena P1 menunjuk pada alamat I, bila kita mengubah nilai pada alamat tersebut nilai I juga akan berubah. Sebagai contoh, pernyataan:

```
P1^ := 100;
```

sama artinya dengan I := 100. Demikian juga bila kita menuliskan pernyataan:

```
P2^ := 123.456;
```

nilai D akan berubah menjadi 123.456.

Selain dengan menugaskan pointer untuk menunjuk variabel lain, kita juga dapat menugaskan pointer untuk menunjuk lokasi memori tertentu dengan pasangan fungsi New dan Dispose.

Fungsi New akan mengalokasikan tempat di memori yang nantinya ditunjuk oleh sebuah pointer. Sebaliknya prosedur Dispose akan mendealokasikan memori agar nantinya dapat digunakan untuk keperluan lain. Sebagai contoh, jika kita mempunyai deklarasi pointer seperti ini:

```
P1: ^integer;
```

kita dapat mengalokasikan memori dengan pernyataan berikut:

```
new (P1);
```

Setelah pointer digunakan, kita harus mendelokasikannya, seperti pada pernyataan berikut:

```
dispose (P1);
```

Sebelum kita mengalokasikan memori untuk sebuah pointer, pointer akan menunjuk lokasi acak di memori. Bila kita ingin sebuah pointer tidak menunjuk ke mana-mana, berilah nilai nil. Contoh:

```
P := nil;
```

4.2. MACAM-MACAM OPERATOR

Pada dasarnya, ada tujuah macam operator pada Pascal, yaitu:

1) Operator pemberian nilai.
2) Operator aritmetik
3) Operator pemanipulasi bit.
4) Operator Boolean.
5) Operator pembanding.
6) Operator himpunan.
7) Operator string.

4.2.1. OPERATOR PEMBERIAN NILAI

Operator yang paling sering digunakan adalah operator pemberian nilai. Pemberian nilai pada Pascal dilakukan dengan menggunakan tanda :=. Contoh:

```
A := 12; {untuk tipe bilangan bulat}
B := 'Halo'; {untuk tipe string}
C := 3.14; {untuk tipe real}
D := [3, 4, 5]; {untuk tipe himpunan}
```

4.2.2. OPERATOR ARITMETIK

Operator aritmetik hanya dapat dikenakan pada operan bertipe bilangan bulat atau bilangan real. Pascal mengenal dua macam operator aritmetik, yaitu:

1) Operator aritmetik tunggal,operan yang digunakan hanya satu.

   Operator aritmetik tunggal pada Pascal ada dua macam seperti terlihat pada Tabel 4.5.

Tabel 4.5. Operator Aritmetik Tunggal

Operator	Operasi	Tipe Operan	Tipe Hasil
+	Identitas (tanda hasil operasi sama dengan tanda operan)	Bilangan bulat Bilangan real	Bilangan bulat Bilangan real
-	Invers (tanda hasil operasi adalah lawan dari tanda operan)	Bilangan bulat Bilangan real	Bilangan bulat Bilangan real

Berikut ini adalah contoh penggunaan operator aritmetik tunggal:

```
x := -y;
```

yang artinya nilai x sama dengan invers dari nilai y. Contoh lain:

```
x := +y;
```

yang sama artinya dengan

```
x := y;
```

Oleh karena itulah, operator identitas jarang digunakan. Lebih praktis untuk menuliskan y daripada +y.

2) Operator aritmetik biner, operan yang digunakan berjumlah dua atau lebih. Ada enam macam operator aritmetik biner pada Pascal seperti terlihat pada Tabel 4.6.

Tabel 4.6. Operator Aritmetik Biner

Operator	Operasi	Tipe Operan	Tipe Hasil
+	Penjumlahan	Bilangan bulat Bilangan real	Bilangan bulat Bilangan real
-	Pengurangan	Bilangan bulat Bilangan real	Bilangan bulat Bilangan real
*	Perkalian	Bilangan bulat Bilangan real	Bilangan bulat Bilangan real
/	Pembagian	Bilangan bulat Bilangan real	Bilangan bulat Bilangan real
Div	Pembagian bilangan bulat	Bilangan bulat	Bilangan bulat
Mod	Sisa pembagian (modulus)	Bilangan bulat	Bilangan bulat

Berikut ini adalah contoh penggunaan operator aritmetik biner:

```
x := y + z;
x := a – b – c – d;
x := 5 * 9 * 3.14;
x := a / b;
x := a + b – c * d;
x := 10 div 2;
x := i mod j;
```

Pascal mengenal dua macam pembagian, yaitu :

a) Pembagian dengan operator /, selalu menghasilkan bilangan real.
b) Pembagian dengan operator div, selalu menghasilkan bilangan bulat dan operatornya juga selalu bilangan bulat.

4.2.3. OPERATOR PEMANIPULASI BIT

Operasi ini berhubungan dengan pemanipulasian bit pada operan, misalnya menggeser bit ke kanan, memutar bit ke kiri, dan sebagainya. Operasi ini hanya dapat dikenakan pada operan bertipe bilangan bulat dan hasil operasinya juga selalu bilangan bulat. Ada enam macam operator pemanipulasi bit seperti terlihat pada Tabel 4.7.

Tabel 4.7. Operator Pemanipulasi Bit

Operator	Operasi
not	Invers
and	Logika and

or	Logika or
xor	Logika xor
shl	Penggeseran bit ke kiri
shr	Penggeseran bit ke kanan

Operator and, or dan xor mengevaluasi masing-masing bit pada operan-operannya. Pada operator and, jika bit dari kedua operan bernilai 1 maka bit hasil operasinya juga bernilai 1; jika tidak, bit hasil operasinya 0. Contoh:

```
x := 1 and 0; { x = 0}
```

Pada operator or, jika bit dari kedua operan mempunyai nilai 0 maka bit hasil operasinya juga bernilai 0; jika tidak, bit hasil operasinya bernilai 1. Contoh:

```
x := 1 or 0; { x = 1}
```

Operator xor akan mengembalikan nilai 1 jika salah satu bit bernilai 1 dan yang lain bernilai0. Jika tidak, hasil operasinya bernilai 0. Contoh:

```
x := 1 xor 0; { x = 1}
```

Operator nor membalik bit-bit operannya. Jika operannya bernilai 1, hasilnya adalah 0. Sebaliknya, jika operan bernilai 0, hasilnya adalah 1.

Tabel 4.8 adalah hasil operator not, and, or dan xor untuk berbagai kombinasi bit.

Tabel 4.8. Hasil Operator Not, And, Or dan Xor untuk Berbagai Kombinasi Bit.

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

Operator shl dan shr masing-masing digunakan untuk menggeser bit-bit operannya ke kiri atau ke kanan. Contoh:

```
x := y shl 2;
```

yang artinya bit-bit pada variabel y digeser ke kiri sejauh 2 bit. Contoh lain:

```
a := b shr 5;
```

yang artinya bit-bit pada variabel b digeser ke kanan sejauh 5 bit.

4.2.4. OPERATOR BOOLEAN

Operator Boolean dikenakan pada operan bertipe Boolean. Karena hanya dioperasikan pada operan Boolean, maka tipe operan dan tipe hasil operasinya adalah Boolean. Ada empat macam operator Boolean seperti terlihat pada Tabel 4.9.

Tabel 4.9. Operator Boolean

Operator	Operasi
not	Negasi
and	Logika and
or	Logika or
xor	Logika xor

Operator and akan mengembalikan true jika semua operan bernilai true; dan mengembalikan false jika salah satu operan bernilai false. Contoh:

```
a := true and false; {a = false}
```

Operator or akan mengembalikan true jika salah satu operan bernilai true; dan mengembalikan false jika semua operan bernilai false. Contoh:

```
a := true or false; {a = true}
```

Operator xor, jika dikenakan pada dua operan, akan mengembalikan true apabila satu operan bernilai true (dan operan lain bernilai false); dan mengembalikan false jika kedua operan bernilai true atau bernilai false. Contoh:

```
a := true xor false; {a = true}
```

Operator not hanya dapat dikenakan pada sebuah operan. Operator ini akan mengembalikan true jika operannya bernilai false; dan mengembalikan false jika operannya bernilai true. Contoh:

```
a := not b; {a = true jika b = false, atau a = flase jika b = true}
```

Tabel 4.10 adalah hasil operator not, and, or dan xor untuk berbagai kombinasi kondisi.

Tabel 4.10. Hasil Operator Not, And, Or dan Xor untuk Berbagai Kombinasi Kondisi

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

true	true	false	false	true	true	false

4.2.5. OPERATOR PEMBANDING

Operator pembanding digunakan untuk membandingkan dua buah operan. Hasil dari operator ini adalah data bertipe Boolean yang menunjukkan hasil perbandingan bernilai benar atau salah. Ada delapan macam operator pembanding seperti terlihat pada Tabel 4.11. Kolom ketiga, tipe operan adlah jenis operan yang dapat dikenai operator pembanding.

Tabel 4.11. Operator Pembanding

Operator	Operasi	Tipe Operan
=	Sama dengan	Tipe sederhana, pointer, himpunan, string
<>	Tidak sama dengan	Tipe sederhana, pointer, himpunan, string
<	Kurang dari	Tipe sederhana, string
>	Lebih dari	Tipe sederhana, string
<=	Kurang dari atau sama dengan	Tipe sederhana, string
>=	Lebih dari atau sama dengan	Tipe sederhana, string
<=	Subset dari	Tipe himpunan
>=	Superset dari	Tipe himpunan

Contoh :

```
a := 5 = 6; {a = false karena 5 tidak sama dengan 6}
a := 5 <> 6; {a = true}
a := 5 < 6; {a = true}
a := (3 < 4) and (5 > 6); {a = false karena 5<6}
a := (4 <= 4) or (5 > 6); {a = true}
a := (4 >= 4) and (5 >=5); {a = true}
```

4.2.6. OPERATOR HIMPUNAN

Operator himpunan hanya digunakan pada tipe himpunan. Ada empat macam operator himpunan seperti terlihat pada Tabel 4.12.

Tabel 4.12. Operator Himpunan

Operator	Operasi
+	Union
-	Selisih
*	Interseksi
in	Anggota dari

Sebagai contoh:

```
A := B + C;
```

menggabungkan semua anggota himpunan B dan C ke dalam A. Jika A, B dan bertipe set of char dan nilai A dan B masing-masing adalah [‘A’,’B’] dan [‘C’], maka variabel C akan bernilai [‘A’, ‘B’, ‘C’].
Contoh lain:

```
D := C – B; {D = [‘A’, ‘B’]}
E := C * B;  {D = [‘C’]}
```

4.2.7. OPERATOR STRING

Pascal hanya mengenal satu macam operator string, yaitu penggabungan. Operator ini digunakan untuk menggabungkan dua atau lebih operan string menjadi sebuah string yang lebih panjang. Simbol untuk operator ini sama dengan operator penjumlahan (+). Contoh:

```
S := ‘Turbo’ + ‘Pascal’; {sama artinya dengan S := ‘Turbo Pascal’;}
```

4.3. DERAJAT OPERATOR

Adanya operator-operator ini menyebabkan munculnya konsep derajat operator. Kegunaan derajat operator adalah menentukan operator mana yang harus dikerjakan lebih dulu dan operator mana yang terakhir dikerjakan.

Derajat operator disajikan pada Tabel 4.13.

Tabel 4.13. Derajat Operator

Operator	Operasi
@ not	Tertinggi
* / div mod as and shl shr	
+ - or xor	
= <> < > <= >= in	Terendah

Sebagai contoh, pernyataan:

```
x := 2 + 3 * 4;
```

akan menghasilkan 14 (=2 + 12). Hal ini dikarenakan operator perkalian memiliki derajat lebih tinggi daripada operator penjumlahan. Akibatnya operasi 3 * 4 harus dikerjakan terlebih dulu baru kemudian dijumlah dengan 2.

Bila kita ingin sebuah operasi dikerjakan terlebih dulu, letakkan operasi tersebut di dalam tand kurung. Sebagai contoh:

```
x := (2 + 3) * 4;
```

Nilai x adalah 20. Hal ini dikarenakan operasi (2 + 3) diletakkan di dalam tanda kurung, jadi

harus dikerjakan terlebih dahulu.

4.4. CONTOH SOAL

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

   a. $a
   b. 10.1
   c. 40000
   d. -$FF
   e. $AAAA
   f. 0010

   Jawab:

   a, d, f.

   Nilai b tidak dapat diberikan karena merupakan bilangan real.

   Nilai c tidak dapat diberikan karena lebih besar daripada 32767.

   Nilai e tidak dapat diberikan karena lebih besar daripada 32767.

2. Jika diketahui y adalah variabel bertipe real, dari nilai-nilai berikut, manakah yang dapat diberikan pada y?

   a. .3
   b. 1E2
   c. 3E+4.2
   d. $5.0
   e. -5E-5
   f. 3,14

   Jawab:

   b dan e.

   Nilai a tidak dapat diberikan karena tidak ada angka di depan tanda titik.

   Nilai c tidak dapat diberikan karena eksponennya bilangan real.

   Nilai d tidak dapat diberikan karena bilangan heksadesimal tersebut berupa bilangan real.

   Nilai f tidak dapat diberikan karena tanda koma seharusnya adalah titik.

3. Ubahlah beberapa ekspresi matematik berikut ke dalam bahasa Pascal.

a. $E = m.c^2$

b. $L = ¼\ \pi.d^2$

c. $c = \sqrt{a^2 + b}$

d. $L = ½\ a.b.\sin \theta$

e. $s = v.t + ½\ a.t^2$

f. $y = \sec^2 x - \tan^2 x$

g. $x = \frac{-b + \sqrt{b^2 - 4ac}}{2a}$

Jawab:

a. E := m * c * c; atau E := m * sqr (c);

b. L := 0.25 * Pi * sqr (d);

c. c := sqrt (sqr (a) + sqr (b));

d. L := 0.5 * a *b * sin (teta);

e. s := v * t + 0.5 * a * sqr (t);

f. y := sqr (1.0 / cos (x)) – sqr (sin (x) / cos (x));

g. x := (-b + sqrt (b * b – 4 * a * c)) / (2 * a);

BAB V
PERCABANGAN

Salah satu elemen algoritma yang paling sederhana,yaitu percabangan. Pernyataan percabangan memungkinkan suatu pernyataan dieksekusi hanya jika suatu kondisi terpenuhi atau tidak terpenuhi.

Contoh berikut ini adalah algoritma untuk menuliskan nilai absolut dari nilai yang dimasukkan pengguna. Definisi dari nilai absolut adalah sebagai berikut:

$|x| = x$, jika $x \leq 0$

$|x| = -x$, jika $x > 0$.

Berikut ini adalah algoritma untuk menuliskan nilai absolut dari masukan pengguna:

1. Masukkan x.
2. Jika (x < 0) maka kerjakan baris 3, jika tidak kerjakan baris 4.
3. x ← -x.
4. Tulis x.

Baris pertama meminta masukkan suatu bilangan dari pengguna yang disimpan pada variabel x.

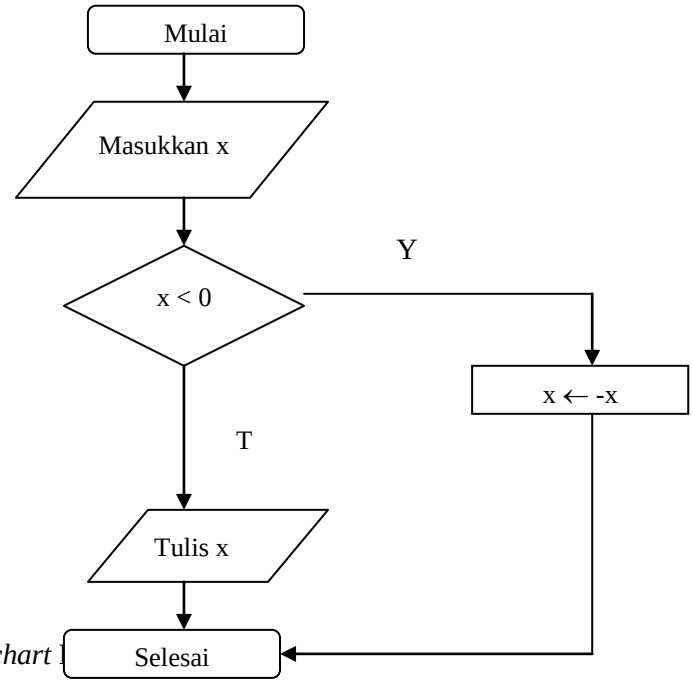
Baris kedua mengecek nilai variabel x lebih kecil dari 0 atau tidak. Jika ya, yang dikerjakan adalah baris ketiga, tetapi jika tidak yang dikerjakan adalah baris keempat.

Baris ketiga membuat nilai x menjadi positif karena baris ini hanya dikerjakan jika nilai x lebih

kecil dari 0.

Baris terakhir menuliskan nilai x yang selalu positif.

Gambar 5.1 adalah contoh diagram alir untuk algoritma di atas.



Gambar 5.1. *Flowchart* l

5.1. PERNYATAAN IF..ELSE

Pernyataan if..else digunkan untuk menguji sebuah kondisi. Bila kondisi yang diuji terpenuhi, program akan menjalankan pernyataan-pernyataan tertentu; dan bila kondisi yang diuji salah program akan menjalankan pernyataan-pernyataan lain. Bentuk umum pernyataan if..else adalah sebagai berikut:

```
if  kondisi then
begin
        {pernyataan-pernyataan yang dijalankan jika kondisi terpenuhi}
end
else
begin
        {pernyataan-pernyataan yang dijalankan jika kondisi tidak terpenuhi}
end
```

kondisi sendiri merupakan suatu ekspresi bertipe Boolean, artinya hanya dapat bernilai benar (true) atau

salah (false).

Contoh:

```
if (x > 0) then
        writeln (‘ x bilangan positif’)
else
        writeln (‘x bukan bilangan positif’);
```

Program ini akan menuliskan ‘x bilangan positif’ jika variabel x lebih besar daripada 0 dan menuliskan ‘x bukan bilangan positif’ jika x lebih kecil atau sama dengan 0.

Pernyataan if..else di atas tidak diletakkan di antara kata kunci bagin dan end. Kita tidak perlu menambahkan dua kata kunci ini jika pernyataan yang dituliskan hanya satu baris saja.

Contoh lain untuk pernyataan IF..ELSE:

```
var
        NilaiUjian : real;
begin
        write (‘Nilai yang didapat ?’);
        readln (NilaiUjian);
(*Seleksi nilai yang didapat, lulus atau tidak*)
        if NilaiUjian > 60 then
          writeln (‘Lulus’);
        else
          writeln (‘Tidak Lulus’);
end.
```

Bila program ini dijalankan, akan didapat hasil:

```
Nilai yang didapat ? 70
Lulus
```

Bila kita tidak ingin mengerjakan sesuatu jika kondisi yang ditentukan tidak terpenuhi, kita bisa menuliskan pernyataan if tanpa else. Contoh:

```
if (x < 0) then
        x := -x;
```

Pada program di atas, jika x adalah bilangan negatif, nilainya akan dibuat menjadi positif. Jika x adalah bilangan positif, nilainya tidak berubah.

5.2. PERNYATAAN IF..ELSE IF

Dalam kasus-kasus tertentu, mungkin saja kita harus meletakkan pernyataan if lain setelah else. Perhatikan contoh berikut ini:

```
if (x > 0) then
          writeln (‘ x bilangan positif’)
else if (x < 0) then
          writeln (‘x bilangan negatif’);
else
          writeln (‘x adalah nol’);
```

Pada contoh di atas, mula-mula program mengecek nilai x lebih besar daripada 0 atau tidak. Bila

kondisi ini terpenuhi, program akan menulis ‘x bilangan positif’. Sebaliknya bila tidak terpenuhi, program akan menjalankan pernyataan if kedua untuk mengecek nilai x lebih kecil daripada 0 atau tidak. Bila kondisi ini terpenuhi, program akan menuliskan ‘x bilangan negatif’ dan bila tidak terpenuhi, program menulis ‘x adalah nol’.

5.3. PERNYATAAN CASE

Struktur CASE mempunyai suatu ungkapan logika yang disebut dengan selector dan sejumlah statemen yang diawali dengan suatu label permasalahan (*case label*) yang mempunyai tipe yang sama dengan selector. Statemen yang mempunyai case label yang bernilai sama dengan selector akan diproses sedangkan statemen yang lainnya tidak. Perbedaan dengan struktur IF..ELSE adalah bila statemen IF..ELSE menyeleksi suatu kondisi dan terpenuhi, setelah memproses statemen dalam lingkungan yang terpenuhi tersebut, proses penyeleksian masih dilakukan terhadap statemen IF..ELSE berikutnya yang lain. Sedang pada struktur CASE bila salah satu kondisi terpenuhi (nilai case label sama dengan selector) dan statemen tersebut diproses, selanjutnya statemen-statemen yang lainnya dalam lingkungan CASE tersebut tidak akan diseleksi lagi.

Bentuk dari struktur CASE adalah sebagai berikut:

```
CASE ungkapan OF
        daftar case label 1 : statemen1;
        daftar case label 2 : statemen2;
        daftar case label 3 : statemen3;
                              .
                              .
                              .
        daftar case label n : statemenn;
```

Daftar case label dapat berupa sebuah konstanta, atau range dari konstanta yang bukan bertipe real.

Contoh:

```
1:        (nilai integer 1)
1,2,3,4: (nilai integer 1,2,3,4)
1..5:     (nilai integer 1,2,3,4,5)
‘A’:      (nilai karakter ‘A’)
‘A’,’B’:  (nilai karakter ‘A’ dan ‘B’)
‘A’..’D’: (nilai karakter ‘A’,’B’,’C’,’D’)
‘*’:      (nilai karakter ‘*’)
```

Contoh:

Nilai ujian yang diberikan dalam bentuk huruf A,B,C,D,E,F mempunyai arti sebagai berikut:

- Nilai ‘A’ berarti sangat baik
- Nilai ‘B’ berarti baik

- Nilai ‘C’ berarti cukup
- Nilai ‘D’ berarti kurang
- Nilai ‘E’ atau ‘F’ berarti gagal

```
var
        Nilai : char;
begin
        writeln (‘Nilai huruf yang didapat ? ‘); readln (Nilai);
        case Nilai of
                ‘A’ : writeln (‘Sangat Baik’);
                ‘B’ : writeln (‘Baik’);
                ‘C’ : writeln (‘Cukup’);
                ‘D’ : writeln (‘Kurang’);
                ‘E’, ‘F’ : writeln (‘Gagal’);
        end;
end.
```

Bila program ini dijalankan, akan didapat hasil:

```
Nilai huruf yang didapat ? C
Cukup
```

Pernyataan case digunakan untuk menyederhanakan konstruksi if..else if yang terlalu banyak. Sebagai contoh, program berikut:

```
if (x = 0) then
        writeln (‘ x bernilai 0’)
else if (x = 1) then
        writeln (‘x bernilai 1’)
else if (x = 2) then
        writeln (‘x bernilai 2’)
else if (x = 3) then
        writeln (‘x bernilai 3’)
else
        writeln (‘x tidak bernilai 0, 1, 2, ataupun 3’);
```

dapat diganti menjadi lebih ringkas dan mudah dibaca dengan program berikut:

```
case x of
        0: writeln (‘x bernilai 0’);
        1: writeln (‘x bernilai 1’);
        2: writeln (‘x bernilai 2’);
        3: writeln (‘x bernilai 3’);
else
        writeln (‘x tidak bernilai 0, 1, 2 ataupun 3’);
end;
```

Meskipun hasil kedua program tersebut sama tetapi penulisan dengan case lebih mudah dibaca. Struktur CASE OF mempunyai suatu ungkapan logika yang disebut dengan selector

5.4. CONTOH SOAL

1. Tulislah program yang menampilkan ‘Genap’ jika suatu bilangan adalah bilangan genap atau ‘Ganjil’ jika bilangan tersebut adalah bilangan ganjil. Masukan dari program ini adalah bilangan bulat, yaitu x, dimana $0 \leq x \leq 65535$.

Keluaran dari program ini adalah ‘Bilangan genap’ atau ‘Bilangan ganjil’.

Jawab :

Suatu bilangan dikatakan bilangan genap jika habis dibagi 2. Dengan demikian algoritma pengecekan bilangan genap atau ganjil dapat dituliskan sebagai berikut:

1. Masukkan x.
2. Jika (x habis dibagi 2) maka kerjakan baris 3; jika tidak kerjakan baris baris 4.
3. Tulis ‘Bilangan genap’. Selesai.
4. Tulis ‘Bilangan ganjil’.

Berikut ini adalah contoh program untuk mengecek suatu bilangan termasuk bilangan ganjil atau bilangan genap:

```
var
        x: word;
begin
        write (‘Masukkan suatu bilangan: ‘);
        readln (x);
        if (x mod 2 = 0) then
                writeln (‘Bilangan genap’);
        else
                writeln (‘Bilangan ganjil’);
end.
```

Keluaran dari program di atas adalah sebagai berikut:

```
Masukkan suatu bilangan: 3
Bilangan ganjil
Masukkan suatu bilangan: 10
Bilangan genap
```

2. Tulislah program yang meminta masukkan bilangan bulat dari pengguna. Jika pengguna memasukkan 0, program menampilkan ‘Minggu’; jika pengguna memasukkan 1, program menuliskan ‘Senin’, dan seterusnya sampai dengan ‘Sabtu’. Jika pengguna memasukkan nilai di luar jangkauan 0 sampai dengan 6, program menuliskan ‘Hari tidak valid’.

   Masukan dari program ini adalah bilangan bulat, yaitu x, di mana $0 \leq x \leq 255$.

   Keluaran dari program ini adalah ‘Minggu’, ‘Senin’, ‘Selasa’, .., ‘Sabtu’, atau ‘Hari tidak valid’.

   Jawab:

   Program semacam ini dapat dibuat dengan larik atau pernyataan case..of.

   Algoritma menampilkan hari dapat dituliskan sebagai berikut:

   1. Masukkan x.
   2. Jika (x = 0) Tulis ‘Minggu’. Selesai.
   3. Jika (x = 1) Tulis ‘Senin’. Selesai.
   4. Jika (x = 2) Tulis ‘Selasa’. Selesai.
   5. Jika (x = 3) Tulis ‘Rabu’. Selesai.
   6. Jika (x = 4) Tulis ‘Kamis’. Selesai.

7. Jika (x = 5) Tulis ‘Jumat’. Selesai.
8. Jika (x = 6) Tulis ‘Sabtu’. Selesai.
9. Tulis ‘Hari tidak valid.

Berikut ini adalah contoh program untuk menampilkan hari berdasarkan masukan bilangan bulat:

```
var
        x: byte;
begin
        write (‘Masukkan bilangan bulat (0 – 6):’);
        readln (x);
        case (x) of
                0: writeln (‘Minggu’);
                1: writeln (‘Senin’);
                2: writeln (‘Selasa’);
                3: writeln (‘Rabu’);
                4: writeln (‘Kamis’);
                5: writeln (‘Jumat’);
                6: writeln (‘Sabtu’);
        else
                writeln (‘Hari tidak valid’);
        end;
end.
```

Keluaran dari program di atas adalah sebagai berikut:

```
Masukkan bilangan bulat (0 – 6): 3
Rabu
Masukkan bilangan bulat (0 – 6): 10
Hari tidak valid
Masukkan bilangan bulat (0 – 6): 0
Minggu
```

3. Tulislah program untuk menghitung akar-akar persamaan kuadrat:

   $ax^2 + bx + c = 0$

   Masukan dari program ini adalah tiga bilangan real yaitu a, b dan c, yang masing-masing melambangkan koefisien-koefisien persamaan kuadrat.

   Keluaran dari program ini adalah akar-akar persamaan kuadrat.

   Jawab:

   Rumus pencarian akar-akar persamaan kuadrat $ax^2 + bx + c = 0$ dapat dituliskan sebagai berikut:

$$x_{1,2} = \frac{-b \pm \sqrt{b^2 - 4ac}}{2a}$$

   Nilai $b^2 – 4.a.c$ disebut diskriminan (=D).

   Ada tiga kasus dalam persamaan kuadrat dilihat dari nilai diskriminannya:

   1. Jika D > 0, berarti persamaan kuadrat memiliki akar-akar bilangan real. Sebagai contoh persamaan $x^2 - 5x + 6$ memiliki akar-akar 2 dan 3.
   2. Jika D = 0, berarti persamaan kuadrat memiliki akar kembar, yaitu –b / (2 * a). Sebagai contoh, persamaan $x^2 - 2x + 1$ memiliki akar kembar, yaitu 1.
   3. Jika D < 0, berarti persamaan kuadrat memiliki akar-akar bilangan kompleks. Sebagai contoh, persamaan $x^2 - x + 2$ memiliki akar-akar 0,5 + 1,323i dan 0,5 – 1,323i.

   Dengan demikian algoritma pencarian akar-akar persamaan kuadrat dapat dituliskan:

1. Masukkan a, b, c.
2. D = b * b – 4 * a * c.
3. Jika (D > 0.0) kerjakan baris 4 jika tidak kerjakan baris 7.
4. x1 = (-b + sqrt (D)) / (2 * a).
5. x2 = (-b - sqrt (D)) / (2 * a).
6. Tulis x1, x2. Selesai.
7. Jika (D < 0.0) kerjakan baris 8 jika tidak kerjakan baris 11.
8. x1 = -b / (2 * a).
9. D = sqrt (-D) / (2 * a).
10. Tulis ‘x1 + D.i’, ‘x1 – D.i. Selesai.
11. x1 = -b / (2 * a).
12. Tulis x1.

Program berikut ini adalah contoh mencari akar-akar persamaan kuadrat.

```
var
        A, B, C : real;
        D, x1, x2 : real;
begin
        write (‘A = ‘);
        readln (A);
        write (‘B = ‘);
        readln (B);
        write (‘C = ‘);
        readln (C);
        if (A = 0.0) and (B = 0.0) then
        begin
                writeln (‘Bukan persamaan kuadrat’);
            halt;
        end;

        D := B * B – 4 * A * C;
        {jika akar-akarnya bilangan real}
        if (D > 0.0) then
        begin
                x1 := (-B + sqrt (D)) / (2 * A);
                x2 := (-B - sqrt (D)) / (2 * A);
                writeln (‘x1 = ‘, x1:1:3);
                writeln (‘x2 = ‘, x1:1:3);
        end
        {jika akar-akarnya bilangan kompleks}
        else if (D < 0.0) then
        begin
                x1 := -B / (2 * A);
                D := sqrt (-D) / (2 * A);
                writeln (‘x1 = ‘, x1:1:3, ‘ + ‘, D:1:3, ‘i’);
                writeln (‘x2 = ‘, x1:1:3, ‘ + ‘, D:1:3, ‘i’);
        end
        {jika akarnya kembar}
        else
        begin
                x1 := -B / (2 * A);
                writeln (‘x1 = x2 = ‘, x1:1:3);
        end
end.
```

Keluaran dari program di atas adalah sebagai berikut:

A = 1
B = -5
C = 6
x1 = 3.000
x2 = 2.000

A = 1
B = -2
C = 1
x1 = x2 = 1.000

A = 1
B = -1
C = 2
x1 = 0.500 + 1.323i
x2 = 0.500 + 1.323i

BAB VI
PENGULANGAN

Pengulangan digunakan untuk menjalankan satu atau beberapa pernyataan sebanyak beberapa kali. Pengulangan memungkinkan kita menjalankan beberapa pernyataan hanya dengan menuliskan pernnyataan tersebut satu kali saja.

Berikut ini adalah algoritma untuk menghitung rata-rata dari sekumpulan data yang dimasukkan pengguna. Rumus mencari data-data dari data x_i yang berjumlah N adalah sebagai berikut:

$$\bar{x} = \frac{\sum_{i=1}^{N} x_i}{N}$$

Berikut ini adalah algoritma untuk menghitung rata-rata data yang dimasukkan pengguna:

1. Masukkan N.
2. i ← 1.
3. j ← 0.
4. Selama (i <= N) kerjakan baris 4 sampai dengan 7.
5. Masukkan dt.
6. i ← i + 1.
7. j ← j + dt.
8. Rata ← j / N.
9. Tulis Rata.

Baris pertama meminta pengguna memasukkan N, yaitu jumalh data.

Pada baris kedua, variabel i, yang berguna sebagai pencacah banyaknya data yang telah dimasukkan pengguna, diberi nilai 1.

Pada baris ketiga, variabel j, yang digunakan untuk menyimpan hasil penjumlahan data, diberi nilai

0.

Baris keempat memberikan perintah untuk mengulangi baris keempat sampai dengan baris ketujuh selama i kurang dari sama dengan N. Setelah i lebih besar dari N, baris kedelapan yang dijalankan.

Baris kelima meminta masukkan data yang ke-i.

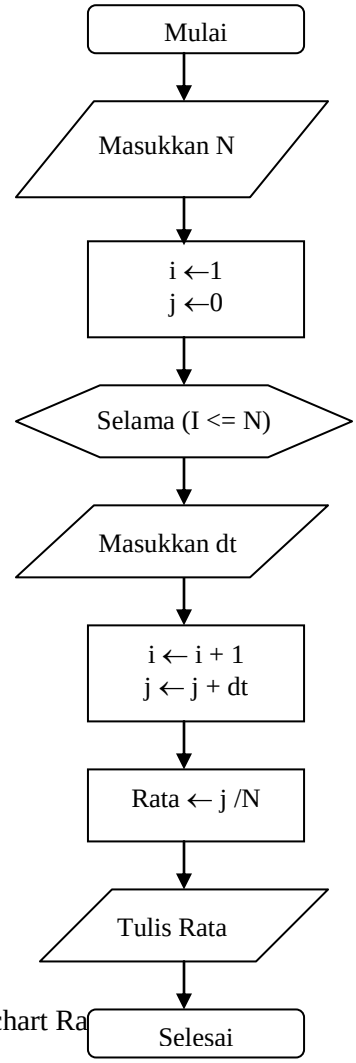
Baris keenam menambah variabel i dengan 1. Arti dari perintah i ← i + 1 adalah nilai i ditambah dengan 1 kemudian hasilnya disimpan pada variabel i kembali.

Baris ketujuh menambah variabel j dengan data yang dimasukkan pengguna. Variabel j digunakan untuk menyimpan hasil penjumlahan semua data, jadi untuk setiap masukan data, nilai variabel j harus ditambah dengan dt.

Baris kedelapan menghitung rata-rata dengan cara membagi hasil penjumlahan dengan banyaknya data.

Baris terakhir menuliskan rata-rata tersebut.

Gambar 6.1 adalah contoh diagram alir untuk algoritma di atas.



Gambar 6.1. Flowchart Ra

6.1. PERNYATAAN FOR

Pernyataan pengulangan yang paling sering digunakan adalah for. Pernyataan ini digunakan bila kita sudah tahu berapa kali kita akan mengulang satu atau beberapa pernyataan. Bentuk umum pernyataan for adalah sebagai berikut:

```
for pencacah := nilai_awal to nilai_akhir do
begin
        (pernyataan-pernyataan yang akan diulang}
end;
```

Blok berisi pernyataan-pernyataan yang harus diulang disebut kalang (loop).

Sebagai contoh, program berikut ini akan menuliskan string 'Turbo Pascal' sebanyak 10 kali:

```
for x := 1 to 10 do
        writeln ('Turbo Pascal');
```

Kita juga dapat menuliskan string 'Turbo Pascal' sebanyak 10 kali dengan menuliskan program berikut:

```
for x := 3 to 12 do
        writeln ('Turbo Pascal');
```

Contoh lain:

```
var
        I : integer;
begin
        for I := 1 to 5 do writeln ('Pascal');
end.
```

Bila program ini dijalankan, akan didapatkan hasil:

```
Pascal
Pascal
Pascal
Pascal
Pascal
```

Berarti statemen writeln ('Pascal') akan diulang sebanyak 5 kali, yaitu dengan penghitung I dari nilai awal 1 sampai dengan 5.

Kita dapat melakukan pengulangan dari bilangan yang lebih besar ke bilangan yang lebih kecil. Caranya adalah mengganti kata kunci to dengan downto. Contoh:

```
for x := 10 downto 1 do
    writeln ('Turbo Pascal');
```

6.2. PERNYATAAN FOR BERSARANG

Kita juga dapat menuliskan pernyataan for di dalam pernyataan for. Konstruksi semacam ini disebut

dengan pernyataan for bersarang. Contoh:

```
for x := 1 to 3 do
    for y := 1 to 2 do
        writeln (‘x, ‘ ‘ y);
```

Kalang for yang luar (dengan pencacah variabel x) akan menjalankan kalang yang dalam (dengan pencacah y) sebanyak 3 kali. Dan pada setiap pengulangan di layar akan dituliskan nilai x dan y. Berikut ini adalah keluaran dari program di atas:

```
1   1
1   2
2   1
2   2
3   1
3   2
```

Pada saat x bernilai 1, y diulang sebanyak 2 kali. Jadi pada layar akan tertulis 1 1 dan 1 2. Demikian juga pada saat x bernilai 2, y diulang sebanyak 2 kali. Jadi pada layar akan tertulis 2 1 dan 2 2. Hal yang sama terjadi pada saat x bernilai 3.

Dengan blok statemen, berarti statemen yang diulang dapat lebih dari sebuah statemen. Pada contoh berikut, yamg akan diproses berulang adalah sebanyak 2 buah statemen dalam satu blok statemen.

```
var
        I : integer;
begin
        for I := 1 to 5 do
        begin
                write (I);
                writeln (‘Pascal’);
        end;
end.
```

Bila program ini dijalankan, akan didapatkan hasil:

```
1 Pascal
2 Pascal
3 Pascal
4 Pascal
5 Pascal
```

6.3. PERNYATAAN WHILE

Pernyataan pengulangan ini biasanya digunakan bila kita belum tahu pasti berapa banya kita akan mengulang pernyataan-pernyataan. Berakhirnya pengulangan ini ditentukan oleh suatu kondisi. Bila kondisi sudah tidak terpenuhi maka pengulangan akan berakhir. Selama kondisi masih terpenuhi,

pengulangan akan terus dilakukan. Bentuk umum pernyataan pengulangan while adalah sebagai berikut:

```
while kondisi do
begin
    {pernyataan-pernyataan yang akan diulang}
end;
```

kondisi merupakan suatu ekspresi Boolean, artinya hanya dapat bernilai benar (true) atau salah (false). Contoh:

```
while (x > 0) do
    begin
        x := x – 1;
        y := y – 1;
    end;
```

Program ini akan mengulang dua pernyataan (x := x – 1) dan (y := y – 1) selama nilai x masih positif.

Contoh lain:

```
var
        I : integer;
begin
        I := 0;
        while I < 5 do
        begin
                writeln (I);
                I := I + 1;
        end;
end.
```

Bila program ini dijalankan, akan didapatkan hasil:

```
0
1
2
3
4
```

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

6.4.PERNYATAAN REPEAT..UNTIL

Pernyataan pengulangan ini hampir sama dengan pernyataan pengulangan while, dan biasanya digunakan bila jumlah pengulangan belum dapat ditentukan pada saat program ditulis. Untuk perbedaannya disajikan pada tabel 6.1.

Tabel 6.1. Perbedaan Pernyataan Repeat..Until dan While

Perbedaan	Repeat..until	While
Letak pengecekan kondisi	Kondisi dicek pada akhir kalang	Kondisi dicek pada awal kalang
Pengulangan kondisi	Mengulang pernyataan selama kondisi belum terpenuhi	Mengulang pernyataan selama kondisi masih terpenuhi

Bentuk umum pernyataan repeat..until adalah sebagai berikut:

```
repeat
        {pernyataan-pernyataan yang akan diulang}
until kondisi;
```

Contoh:

```
repeat
        X := X – 1;
        Y := Y – 1;
until (X <= 0);
```

Program ini akan mengulang dua pernyataan (X := X – 1) dan (Y := Y - 1) sampai nilai X lebih kecil atau sama dengan 0.

Contoh lain:

```
var
        I : integer;
begin
        I := 0;
        repeat
                I := I + 1;
                writeln (I);
        until I = 5;
end.
```

Bila program ini dijalankan, akan didapatkan hasil:

```
1
2
3
4
5
```

6.5. PERNYATAAN BREAK

Pada beberapa kasus, kita mungkin ingin menghentikan suatu pengulangan di tengah jalan. Untuk menghentikan pengulangan, digunakan pernyataan break. Kita dapat menggunakan pernyataan break pada pernyataan for, while maupun repeat..until. Contoh:

```
X :=0;
while (X >= 0) do
begin
    X := X + 1;
    if (X > 100) then
            break;
end;
```

Bila dilihat sekilas, kalang di atas sepertinya tidak pernah berakhir karena X selalu ditambah 1 padahal kondisi yang harus dipenuhi adalah (X >= 0). Namun bila dijalankan, setelah nilai X lebih besar daripada 100, pengulangan akan berhenti. Hal ini dikarenakan adanya pernyataan break.

Pernyataan break dapat juga digunakan pada pernyataan pengulangan bersarang (baik for, while maupun repeat..until). Pada pernyataan bersarang pernyataan break akan menghentikan kalang paling dalam.

6.6. PERNYATAAN CONTINUE

Pernyataan continue adalah bentuk pernyataan khusus lain yang digunakan untuk mengembalikan aliran program ke pengujian kondisi pengulangan. Pernyataan-pernyataan di bawah continue akan diabaikan. Contoh:

```
for i := 0 to 100 do
begin
    if ( (i mod 3 <> 0 ) then
            continue;
    writeln (i);
end;
```

Program ini akan menuliskan semua bilangan kelipatan tiga yang kurang dari 100, karena pada saat nilai pencacah pengulangan tidak habis dibagi 3, program memanggil pernyataan contnue, akibatnya pernyataan writeln (i) tidak dipanggil.

Pernyataan continue dapat juga digunakan pada pernyataan pengulangan bersarang (baik for, while maupun repeat..until). Pada pernyataan bersarang pernyataan continue akan melanjutkan kalang paling dalam.

6.7. CONTOH SOAL

1. Tulislah program untuk menampilkan semua bilangan ganjil kurang dari 100.

   Jawab:

   Untuk menampilkan semua bilangan ganjil yang kurang dari 100, kita harus melakukan pengulangan dari 1 sampai 99. Namun variabel pencacah harus ditambah 2 untuk setiap pengulangan, bukan 1 seperti biasanya. Jadi variabel pencacah akan bernilai 1, 3, 5, …, 99. Karena pernyataan for tidak memungkinkan kita menggunakan penambahan variabel pencacah sebanyak 2, berarti kita harus menggunakan pernyataan while..do atau repeat..until.

   Algoritma penulisan bilangan ganjil yang kurang dari 100 dapat dituliskan sebagai berikut:

   1. i ← 1.
   2. Selama (i < 100) kerjakan baris 3 sampai dengan 4.
   3. Tulis i.
   4. i ← i + 2.

   Berikut ini adalah contoh program untuk menuliskan bilangan ganjil yang kurang dari 100:

```
var
    i : byte;
begin
    i := 1;
            2.   while (i < 100 ) do
    begin
            writeln (i);
            i := i + 2;
    end;
end.
```

   Keluaran program di atas adalah sebagai berikut:

```
1
3
5
…
95
97
99
```

2. Tulislah program untuk menampilkan pola-pola bintang seperti berikut:

```
*           *           *
* *         * *         * *
* * *       * * *
* * * *
Untuk n = 4 Untuk n = 3 Untuk n = 2
```

   Masukkan dari program ini adalah nilai n, di mana 1 ≤ n ≤ 255.

   Keluaran dari program ini adalah pola bintang yang diharapkan.

   Jawab:

Untuk menuliskan pola-pola di atas, kita memerlukan dua variabel pencacah. Variabel pertama untuk mencacah baris. Jumlah baris dari pola di atas sesuai dengan nilai n. Jadi, variabel pertama ini digunakan untuk melakukan pengulangan dari 1 sampai dengan n.

Variabel pencacah kedua digunakan untuk menggambar bintang pada masing-masing baris. Untuk baris pertama digambar 1 bintang, untuk baris kedua digambar 2 bintang, untuk baris ketiga digambar 3 bintang, demikian seterusnya. Variabel ini harus melakukan pengulangan dari 1 sampai dengan variabel pencacah pertama.

Algoritma penulisan pola-pola bintang di atas dapat dituliskan sebagai berikut:

1. Masukkan N.
2. i ← 1.
3. Selama (i <= N) kerjakan baris 4 sampai dengan 9.
4. j ← 1.
5. Selama (j <= i) kerjakan baris 6 dan 7.
6. Tulis '* '.
7. j ← j + 1.
8. Ganti baris
9. i ← i + 1.

Berikut ini adalah contoh program untuk menampilkan pola-pola bintang di atas:

```
var
    N: byte;
    i, j: byte;
begin
    write ('Masukkan N: ');
    readln (N);
    for I := 1 to N do
    begin
            for j := 1 to I do
                    write ('* ');
            writeln;
    end;
end.
```

Keluaran program di atas adalah sebagai berikut:

```
Masukkan N: 2
*
* *
Masukkan N: 5
*
* *
* * *
* * * *
```

* * * * *

3. Pada program berikut ini, berapa kali string ‘Algoritma’ muncul di layar?

```
var
    x: byte;

begin
    for x := 1 to 10 do;
            writeln (‘Algoritma’);
end.
```

Jawab :

Satu kali, bukan sepuluh kali.

String ‘Algoritma’ hanya akan tertulis satu kali karena dibelakang pernyataan for..do terdapat tanda titik koma. Artinya tidak ada pernyataan yang perlu diulang. Dengan demikian string ‘Algoritma’ hanya akan tertulis sebanyak 1 kali saja.

BAB VII
SUBRUTIN

Ada dua macam subrutin pada Pascal, yaitu prosedur dan fungsi. Kedua jenis subrutin ini memiliki kegunaan yang sama, yaitu melakukan tugas tertentu. Perbedaannya fungsi selalu mengembalikan suatu nilai setelah dipanggil sedangkan prosedur tidak.

Subrutin diperlukan karena dalam program yang besar akan lebih baik jika tugas tertentu dilakukan oleh subrutin tertentu. Dengan demikian program menjadi lebih mudah dibaca. Contoh:

```
begin
        BacaData;
        ProsesData;
        TulisHasil;
end.
```

Di samping itu, pelacakan kesalahan program juga menjadi lebih mudah. Untuk contoh di atas, apabila terjadi kesalahan dalam pembacaan data, kita dapat memusatkan perhatian pada subrutin BacaData. Apabila berikutnya terjadi kesalahan dalam penulisan hasil, kita hanya perlu memusatkan perhatian pada subrutin TulisHasil.

7.1. PROSEDUR

Prosedur merupakan bagian terpisah dari program dan dapat diaktifkan dimanapun di dalam program. Prosedur dibuat bilamana program akan dibagi-bagi menjadi beberapa blok modul. Prosedur dibuat di dalam program dengan cara mendeklarasikannya di bagian deklarasi prosedur. Kata cadangan PROCEDURE digunakan sebagai judul dari bagian deklarasi prosedur, diikuti oleh identifier yang merupakan nama prosedurnya dan secara optional dapat diikuti oleh kumpulan parameter yang diakhiri dengan titik koma.

Berikut ini adalah sintaks penulisan prosedur secara umum:

```
procedure nama_prosedur (parameter1; parameter2; …);
begin
        {pernyataan-pernyataan}
end;
```

nama_prosedur merupakan nama yang kita berikan ke prosedur. Parameter1, parameter2, … merupakan informasi yang diberikan ke prosedur. Berikut ini adalah contoh sebuah prosedur dengan nama Prosedurku:

```
procedure Prosedurku;
begin
        {pernyataan-pernyataan}
end;
```

Prosedur di atas belum memiliki parameter. Berikut ini adalah contoh prosedur dengan parameter. Dalam memberikan parameter, kita harus menentukan tipe parameternya.

```
procedure Coba (S: string);
begin
        {pernyataan-pernyataan}
end;
```

Bila sebuah prosedur memerlukan beberapa parameter, masing-masing parameter dipisahkan dengan tanda titik koma. Contoh:

```
procedure CobaLagi (S: string; x: integer);
begin
        {pernyataan-pernyataan}
end;
```

Untuk beberapa parameter dan dengan tipe sama, kita cukup menuliskan tipe satu kali tetapi masing-masing parameter dipisahkan dengan tanda koma.
Contoh:

```
procedure Hitung (X,Y: integer);
begin
        {pernyataan-pernyataan}
end;
```

Pemanggilan sebuah prosedur dilakukan dengan menyebutkan namanya. Contoh:

```
Prosedurku;
```

Bila terdapat parameter, letakkan parameter di dalam tanda kurung. Sebagai contoh:

```
Coba (‘Pascal’);
```

memanggil prosedur Coba dengan parameter ‘Pascal’. Contoh lain:

```
CobaLagi (‘Pascal’, 100);
```

Berikut ini adalah contoh program dengan menggunakan prosedur:

```
program Contoh_Prosedur (layar);
procedure Tambah (x, y : integer; var hasil : integer);
begin
        hasil := x + y;
end;
{program utama}
```

```
var
        z : integer;
begin
        Tambah (2, 3, z);
        writeln ('2 + 3 = ', z);
end.
```

Bila program ini dijalankan, akan didapatkan hasil:

2 + 3 = 5

7.2. FUNGSI

Fungsi juga merupakan bagian program yang terpisah mirip dengan prosedur tetapi ada beberapa perbedaannya. Fungsi yang akan digunakan di dalam program harus dideklarasikan terlebih dahulu. Kata cadangan FUNCTION mengawali bagian deklarasi fungsi diikuti oleh identifier yang merupakan nama dari fungsinya dan secara optional dapat diikuti oleh kumpulan parameter, tipe dari fungsinya dan diakhiri dengan titik koma.

Function dapat digunakan secara berulang-ulang dalam program dan lebih cenderung untuk membuat perintah-perintah yang bersifat perhitungan. Hasil function haruslah berupa string, real, integer, char, Boolean atau pointer.

Berikut ini adalah sintaks penulisan fungsi secara umum:

```
function nama_fungsi (parameter1; parameter2; …): tipe_kembalian;
begin
        {pernyataan-pernyataan}
end;
```

nama_fungsi merupakan nama yang kita berikan ke fungsi. Tipe_kembalian adalah tipe data dari nilai yang dikembalikan oleh fungsi. Parameter1, parametr2, … merupakan informasi yang diberikan ke fungsi. Berikut ini adalah contoh sebuah fungsi dengan nama Fungsiku:

```
function Fungsiku;
begin
        {pernyataan-pernyataan}
end;
```

Fungsi di atas belum memiliki parameter. Dalam memberikan parameter, kita harus menentukan tipe parameternya. Contoh:

```
1. function sinus (X:real):real;
   begin
       sinus := sin (X/(180/PI));
       end;
2. function cosinus (X:real):real;
   begin
       cosinus := cos (X/(180/PI));
```

```
        end;
3.  function tangent (X:real):real;
    begin
        tangent := sinus (X) / cosinus (X);
    end;
4.  function log (X:real):real;
    begin
        log := ln (X) / ln (10);  {logarithma biasa dengan bilangan dasar 10};
    end;
```

Bila sebuah fungsi memerlukan beberapa parameter, masing-masing parameter dipisahkan dengan tanda titik koma.

Untuk beberapa parameter dan dengan tipe sama, kita cukup menuliskan tipe satu kali tetapi masing-masing parameter dipisahkan dengan tanda koma. Contoh:

```
1.  function Pangkat (X, Y: real): real;
    begin
        Pangkat := Exp (Ln (X)*Y);
    end;
2.  function LagiLagiCoba (i, j: longint): longint;
    begin
        {pernyataan-pernyataan}
    end;
```

Fungsi LagiLagiCoba di atas memerlukan dua parameter bertipe longint, yaitu i dan j. Fungsi LagiLagiCoba di atas mempunyai tipe kembalian longint. Untuk mengembalikan nilai pada sebuah fungsi, berilah nilai pada variabel yang bernama sama dengan nama fungsinya. Sebagai contoh, kita dapat menuliskan pernyataan berikut:

```
function LagiLagiCoba (i, j: longint): longint;
begin
        LagiLagiCoba := i + j;
end;
```

Pemanggilan sebuah fungsi dilakukan dengan menyebutkan namanya. Bila terdapat parameter, letakkan parameter di dalam tanda kurung.

Untuk menangkap nilai kembalian dari sebuah fungsi, kita harus menyediakan variabel. Contoh:

```
x := LagiLagiCoba (2, 5);
```

Berikut ini adalah contoh program dengan menggunakan fungsi:

```
program Contoh_Fungsi (layar);
procedure Tambah (x, y : integer) : integer;
begin
        Tambah := x + y;
end;
{program utama}
begin
```

```
        writeln ('2 + 3 = ',Tambah (2, 3));
end.
```

Bila program ini dijalankan, akan didapatkan hasil:

2 + 3 = 5

7.3. MELEWATKAN PARAMETER

Dalam melewatkan parameter ke sebuah prosedur atau fungsi, kita dapat melewatkannya dengan :

a. Nilai.

   Bila kita melewatkannya dengan nilai, subrutin yang dipanggil tidak dapat mengubah nilainya.

b. Referensi.

   Bila kita melewatkannya dengan referensi, subrutin yang dipanggil dapat mengubah nilainya.

Untuk melewatkan parameter dengan dengan referensi, tambahkan kata kunci var di depan nama parameter. Contoh:

```
procedure Coba (var i, j: longint);
begin
        i := 100;
        j := 200;
end;
```

Pada contoh di atas, nilai parameter i dan j masing-masing akan diubah menjadi 100 dan 200. Bila kita menghilangkan kata kunci var, nilai parameter i dan j tidak akan berubah setelah prosedur Coba dipanggil. Parameter i dan j hanya akan bernilai 100 dan 200 pada prosedur Coba saja.

7.4. DEKLARASI GLOBAL DAN LOKAL

Di dalam sebuah prosedur atau fungsi, kita dapat mendeklarasikan tipe, variabel, atau konstanta yang bersifat local. Kita juga dapat mendeklarasikan tipe, variabel, atau konstanta di luar prosedur, dengan demikian sifatnya global. Contoh:

```
var
        Global: integer;
procedure Coba;
var
        Lokal: integer;
begin
end;
```

7.5. CONTOH SOAL

1. Tulislah fungsi untuk memangkatkan suatu bilangan real dengan bilangan real lainnya.

   Fungsi ini menerima dua parameter bertipe bilangan real dan mengembalikan bilangan real hasil pemangkatan kedua parameter tersebut.

   Jawab:

   Berikut ini adalah contoh fungsi untuk melakukan pemangkatan:

```
function Pangkat (x, y: real): real;
var
    z: real;
begin
    if (x < 0.0) then
    begin
            if (Round (y) =y) then
            begin
                    if (round (y) mod 2 = 0) then
                            z := exp (y * ln (-x))
                    else
                            z := -exp (y * ln (-x));
            end
            else
            begin
                    Pangkat := 0.0;
                    Writeln ('Tidak dapat diselesaikan');
                    exit;
            end;
    end
    else if (x > 0.0) then
            z := exp (y * ln (x))
    else
            z := 0.0;
    Pangkat := z;
end;
```

   Berikut ini adalah contoh penggunaan fungsi pemangkatan di atas:

```
var
    x, y: real;
begin
    write ('Masukkan x: ');
    readln (x);
    write ('Masukkan y: ');
    readln (y);
    writeln ('x^y: ', Pangkat (x, y):1:3);
end.
```

   Keluaran program di atas adalah sebagai berikut:

```
Masukkan x: 3
Masukkan y: 4
x^y: 81.000

Masukkan x: -5
Masukkan y: -2
x^y: 0.040
```

2. Tulislah sebuah prosedur untuk menukarkan nilai dari dua variabel. Sebagai contoh jika terdapat variabel a dan b, setelah pemanggilan prosedur ini, nilai a sama dengan b dan nilai b sama dengan a. Prosedur ini menerima dua parameter bertipe bilangan real.

   Jawab:

   Untuk membuat prosedur penukaran dua variabel ini, kita harus menyediakan satu variabel sebagai penampung sementara. Hal ini dikarenakan kita tidak dapat menulis pernyataan berikut:

   a := b;

   b := a;

   Pada baris pertama nilai a menjadi sama dengan b, pada baris kedua nilai b dijadikan sama dengan a yang tidak lain sama dengan nilai b mula-mula. Jadi penukaran dua variabel dengan cara di atas tidak dapat dilakukan.

   Variabel penampung sementara digunakan untuk menyimpan salah satu variabel agar tidak 'hilang' seperti kasus di atas. Algoritma penukaran dua variabel dapat dituliskan sebagai berikut:

   1. Temp ← b.
   2. b ← a.
   3. a ← temp.

   Untuk memperjelas algoritma di atas, misalkan nilai a dan b mula-mula adalah 5 dan 7. Proses penukaran ini dapat dilihat pada tabel 7.1.

   Tabel 7.1. Proses Penukaran Dua Variabel

Pernyataan	a	b	temp
Mula-mula	5	7	-
Temp := b;	5	7	7
B := a;	5	5	7
A := temp;	7	5	7

   Berikut ini adalah contoh prosedur untuk menukar isi dua variabel:

```
procedure Tukar (var a, b: real);
var
    temp: real;
begin
    temp := b;
    b := a;
    a := temp;
end;
```

   Ada penambahan kata kunci var di depan parameter prosedur ini agar setelah pemanggilan prosedur, variabel a dan b dapat berubah.

   Berikut ini adalah contoh penggunaan prosedur penukaran di atas:

```
var
    a, b: real;
begin
    write ('Masukkan a = ');
```

```
    readln (a);
    write ('Masukkan b = ');
    readln (b);
    Tukar (a, b);
    writeln ('a = ', a:1:3);
    write ('b = ', b:1:3);
end.
```

Keluaran program di atas adalah sebagai berikut:

```
Masukkan a = 5
Masukkan b = 7
a = 7.000
b = 5.000

Masukkan a = -6
Masukkan b = 10
a = 10.000
b = -6.000
```

3. Tulislah prosedur untuk menyederhanakan bilangan rasional atau pecahan berbentuk a/b di mana a dan b adalah bilangan bulat dan a < b. Sebagai contoh masukan $5/10$ akan menghasilkan keluaran $1/2$, $60/90$ akan menghasilkan $2/3$,dan seterusnya.

   Prosedur ini menerima dua parameter bertipe bilangan bulat, yaitu pembilang dan penyebut dari bilangan rasional.

   Jawab:

   Untuk menyederhanakan suatu pecahan, kita harus mencari FPB (Faktor Persekutuan Terbesar) dari pembilang dan penyebut. Untuk itu, kita harus membuat fungsi FPB terlebih dahulu seperti contoh berikut:

```
function FPB (m, n: integer): integer;
var
    r: integer;
begin
    repeat
            r := m mod n;
            if (r <> 0) then
            begin
                    m := n;
                    n := r;
            end;
    until (r = 0);
    FPB := n;
end;
```

Setelah kita mendapatkan FPB dari pembilang dan penyebut, langkah selanjutnya adalah membagi pembilang dan penyebut dengan FPB. Berikut ini adalah contoh prosedur untuk menyederhanakan suatu pecahan:

```
procedure Sederhanakan (var x1, x2: integer);
var
    gcd: integer;
```

```
begin
    gcd := FPB (x1, x2);
    x1 := x1 div gcd;
    x2 := x2 div gcd;
end;
```

Berikut ini adalah contoh penggunaan prosedur Sederhanakan di atas:

```
var
    x1, x2: integer;
begin
    write (‘Masukkan pembilang: ‘);
    readln (x1);
    write (‘Masukkan penyebut: ‘);
    readln (x2);
    write (x1, ‘/’, x2, ‘ = ‘);
    Sederhanakan (x1, x2);
    writeln (x1, ‘/’, x2);
end.
```

Keluaran program di atas adalah sebagai berikut:

```
Masukkan pembilang: 5
Masukkan penyebut: 10
5/10 = ½

Masukkan pembilang: 66
Masukkan penyebut: 99
66/99 = 2/3

Masukkan pembilang: 104
Masukkan penyebut: 216
104/216 = 13/27
```

DAFTAR PUSTAKA

1. H. M., Jogiyanto, *Pengenalan Komputer*, Yogyakarta: Andi Offset,1995.
2. Pardosi, Mico, *TurboPascal 7.0*, Surabaya: Indah, 1999.
3. Pranata, Antony, *Algoritma dan Pemrograman*, Yogyakarta: J&J Learning, 2002.

LAMPIRAN A

TABEL ASCII

Des X_{10}	Heksa X_{16}	Karakter ASCII	Des X_{10}	Heksa X_{16}	Karakter ASCII
0	00	NULL	64	40	@
1	01	SOM	65	41	A
2	02	EOA	66	42	B
3	03	EOM	67	43	C
4	04	EOT	68	44	D
5	05	WRU	69	45	E
6	06	RU	70	46	F
7	07	BELL	71	47	G

8	08	BS	72	48	H
9	09	HT	73	49	I
10	0A	LF	74	4A	J
11	0B	VT	75	4B	K
12	0C	FF	76	4C	L
13	0D	CR	77	4D	M
14	0E	SO	78	4E	N
15	0F	SI	79	4F	O
16	10	DC0	80	50	P
17	11	DC1	81	51	Q
18	12	DC2	82	52	R
19	13	DC3	83	53	S
20	14	DC4	84	54	T
21	15	ERR	85	55	U
22	16	SYN	86	56	V
23	17	LEM	87	57	W
24	18	S0	88	58	X
25	19	S1	89	59	Y
26	1A	S2	90	5A	Z
27	1B	S3	91	5B	[
28	1C	S4	92	5C	\
29	1D	S5	93	5D	]
30	1E	S6	94	5E	^
31	1F	S7	95	5F	_
32	20	Spasi	96	60	`
33	21	!	97	61	a
34	22	"	98	62	b
35	23	#	99	63	c
36	24	$	100	64	d
37	25	%	101	65	e
38	26	&	102	66	f
39	27	`	103	67	g
40	28	(	104	68	h
41	29	)	105	69	i
42	2A	*	106	6A	j
43	2B	+	107	6B	k
44	2C	’	108	6C	l
45	2D	-	109	6D	m
46	2E	.	110	6E	n

47	2F	/	111	6F	o
48	30	0	112	70	p
49	31	1	113	71	q
50	32	2	114	72	r
51	33	3	115	73	s
52	34	4	116	74	t
53	35	5	117	75	u
54	36	6	118	76	v
55	37	7	119	77	w
56	38	8	120	78	x
57	39	9	121	79	y
58	3A	:	122	7A	z
59	3B	;	123	7B	{
60	3C	<	124	7C	|
61	3D	=	125	7D	}
62	3E	>	126	7E	~
63	3F	?	127	7F	DEL