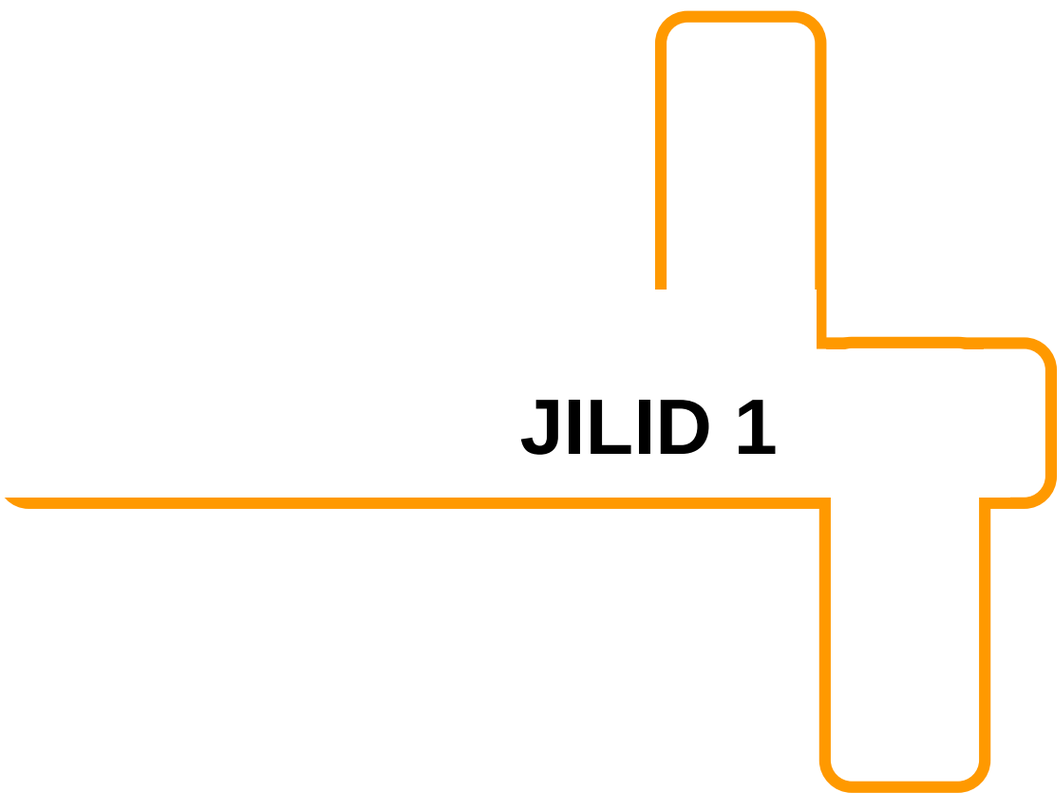
BERMAIN PEMOGRAMAN DENGAN PASCAL



JILID 1

Penulis:

Fauzi Marjalih

cool_sixiz@yahoo.com

KATA PENGANTAR

Alhamdulillah, puji syukur kepada Allah SWT yang masih memberikan umur yang sangat pendek ini. Dengan umur yang pendek ini, syukur sekali saya masih bisa memanfaatkannya dengan baik di atas jalan-Nya. Shalawat tak lupa selalu dihaturkan kepada Nabi Muhammad SAW, yang mana telah berjuang dengan gigih membawa dan mengajarkan ajaran-Nya kepada seluruh umat manusia.

Syukur sekali, pada kali ini saya bisa menyusun karya yang kecil ini, yang mungkin bisa bermanfaat walaupun sekecil zarrah untuk kelangsungan hidup dunia TI. Karya yang berjudul 'Bermain Pemograman Dengan Pascal' ini hanyalah coretan yang tak sengaja saya buat dalam waktu luang saya.

Tetapi ini hanyalah demi keberhasilan TOKI (Tim Olimpiade Komputer Indonesia) khususnya di daerah Bekasi, karena saya belum pernah memberikan hasil yang istimewa pada Olimpiade Komputer untuk daerah asal saya berada. Dalam karya ini ada beberapa materi yang tidak bisa sempat dijelaskan yaitu range (skalar), record, pointer dan file dikarenakan keterbatasan waktu. Tetapi dengan modal ini, cukup untuk anda mempelajarinya secara otodidak. Dengan otodidaklah TI-man menjadi sukses.

Akhir kata, saya ucapkan banyak terima kasih. Dan mungkin karya inilah yang dapat menghilangkan rasa 'putus asa' saya. Semoga dapat bermanfaat bagi para programmer pemula. Dan harapan saya satu-satunya yaitu semoga pada generasi pelajar Kab. Bekasi berikutnya ada yang bisa lolos ke OSN maupun IOI.

Bekasi, 5 Oktober 2005

Penulis

Fauzi Marjalih
cool_sixiz@yahoo.com

I SEKILAS TENTANG PROGRAMMING

Komputer adalah sebuah mesin yang bahasanya hanya terdiri dari karakter 1 dan 0 atau bilangan biner. Oleh karena itu agar komputer dapat dioperasikan oleh oleh manusia seperti biasanya, maka dibutuhkan penterjemah bahasa manusia ke bahasa mesin komputer. Dalam hal ini adalah *software* atau perangkat lunak yang menghubungkan manusia (*brainware*) dengan komputer dan komponennya (*hardware*).

Dalam proses pembuatan suatu sistem operasi atau aplikasi-aplikasi lainnya yang biasa disebut *programming* dibutuhkan sebuah *compiler* yang bertindak sebagai penghasil penterjemah bahasa manusia ke dalam bahasa mesin atau sebuah program atau aplikasi.

Dalam proses compile, dibutuhkan suatu code-code compiler atau code-code pemograman yang tergantung jenis dan bahasa compilernya dalam hal ini melanjut ke dalam programming. Menurut versi-nya ada dua jenis bahasa pemograman yang berkembang saat ini, pertama yang menggunakan sistem *console* dan kedua menggunakan sistem *visual*.

- **Sistem Console**

  Sistem console adalah pemograman yang mengandalkan dalam peng-code-an (*coding*), tanpa adanya kemudahan dalam *klick* & *drag* dan dengan interface yang kurang menarik. Akan tetapi dalam pemograman yang berbasis sistem operasi *Linux* atau *Unix,* sistem seperti ini masih digunakan karena akan menambah kemampuan seseorang dalam meng-coding. Contoh bahasa pemograman yang menggunakan sistem console :

  - Pascal
  - C
  - Q-Basic
  - Perl
  - Java
  - Java Script
  - C++
  - Dll

- **Sistem Visual / Object**

  Sistem Visual adalah perkembangan dari sistem console dengan yang berbasis *object frame oriented* dengan *interface* (tampilan grafis) yang lebih bagus dibanding console dan memfokuskan pada kemudahan dalam memprogram suatu aplikasi dengan metode klik & drag. Sistem visual ini mencakup juga pada pemograman animasi yang bersifat *design grafis*, akan tetapi tetap saja dibutuhkan code-code dalam memprogram. Contoh bahasa pemograman yang menggunakan sistem ini adalah :

  - Borland Delphi (Pascal)
  - Visual Phyton (Phyton)
  - Visual Basic (Basic)
  - Macromedia Flash MX
  - Visual C++ (C++)
  - Dll
  - Visual Foxpro (Foxpro)

Pada kali akan dipelajari bahasa pemograman Pascal yang bersifat console. Karena dalam Olimpiade Komputer atau Informathic Olympiad menggunakan bahasa pemograman pascal sebagai soal programming-nya, dan juga karena Pascal sangat

cocok untuk pemula dengan tingkat kesulitan menengah dibanding Basic yang sangat mudah atau C++, Java, dan C yang sangat sulit untuk dipelajari dan dipahami.

Compiler code Pascal yang digunakan kali ini adalah compiler *Turbo Pascal For Windows Versi 1.5*. Tetapi bisa juga menggunakan compiler *Turbo Pascal Versi 7.0* yang berbasis *MS-DOS (Microsoft Disk Operating System)*. Tetapi dalam olimpiade komputer, compiler yang digunakan adalah *Free Pascal* yang ukuran file-nya lebih besar daripada ukuran file dari Turbo Pascal. Akan tetapi karena Free Pascal lebih baik kinerjanya dibanding dengan Turbo Pascal dan bersifat Free (gratis), maka dari itu dipergunakan sebagai compiler pascal terbaik abad ini.

Setelah mahir menggunakan code-code Pascal (*coding* Pascal), langkah berikutnya yang disarankan adalah melanjutkannya ke versi visualnya yaitu *Borland Delphi* untuk dikembangkan atau mungkin berpindah mempelajari bahasa pemograman yang lebih sulit seperti C++, Java, C, My SQL, Java Script, dll. Mungkin saja bisa membuat suatu program yang bagus, unik dan menarik kemudian menjualnya ke perusahaan-perusahaan IT.

~~~ J J J ~~~
~~~

II KOMPONEN DASAR BAHASA PASCAL

A. SEKILAS TENTANG SYNTAX

Syntax adalah aturan-aturan peng-code-an struktur suatu bahasa pemograman, ibarat grammar dalam berbahasa Inggris. Setiap jenis bahasa pemograman mempunyai aturan syntax yang berbeda. Ada 7 (tujuh) macam syntax yang diperhatikan dalam bahasa Pascal yaitu :

- Ø Comment
- Ø White space
- Ø Symbol
- Ø String
- Ø Number
- Ø Identifier
- Ø Reserved Word

Apabila dalam penulisan code dengan syntax yang salah maka akan menimbulkan *error* atau kesalahan dalam kompilasi. Error dalam pemograman pascal ada dua macam :

- Ø Run-time Error

  Yaitu kesalahan yang terjadi pada saat pengoperasian program. Contohnya kesalahan memasukkan input, perbedaan pada tipe data, dan lain-lain.

- Ø Compile-time Error

  Yaitu kesalahan yang disebabkan aturan penulisan code yang salah yang memunculkan pesan error pertama kali saat dikompilasi. Contohnya kesalahan syntax, variable yang tidak diketahui, kesalahan struktur Begin – End, dan lain-lain.

Dalam Turbo Pascal, pesan kesalahan akan ditampilkan pada status bar dan secara bersamaan proses kompilasipun dihentikan. Lihat gambar 2.1!

Gambar 2.1 Pesan Error / Kesalahan

B. STRUKTUR BAHASA PASCAL

Pertama kali anda membuka program Turbo Pascal telah tersedia listing code standar seperti pada gambar 1.1. Struktur lengkap bahasa Pascal adalah sebagai berikut :

```
Program id_program;
Uses unit;
Label id_label;
Const id_konstanta : ekspresi;
Type id_tipe : tipe_bebas;
Var id_variabel : tipe_variabel;
Procedure id_procedure;
Begin
      Statement_procedure;
End;
Function id_function;
Begin
      Statement_function;
End;

Begin
      ...
      ...
      Statement_program_utama;
End.
```

Sebenarnya, struktur dasar bahasa pascal terdiri dari program, uses, begin dan end. Dalam setiap listing code selalu diawali dengan program dan uses. Pada bagian dari program, label, const, type, var, procedure, dan function selalu terdapat id atau *identifier*.

Statement adalah sebuah sekumpulan pendeklarasian code dari suatu bagian program atau inti program. Sedangkan *ekspresi* adalah sebuah statement pendek yang dideklarasikan setelah tanda *assignment operator* (**:=**). Di dalam pascal, setelah berakhirnya suatu statement atau ekspresi selalu diakhiri dengan tanda titik koma (*semicolon*), sedangkan akhir dari suatu program selalu diakhiri dengan " **End.** ". Deklarasi "End" ada dua macam, pertama *End* yang mengakhiri seluruh program dengan tanda titik (**End.**), kedua *End* yang mengakhiri suatu statement dengan tanda *semicolon* (**End;**).

Coba perhatikan contoh listing/code program sederhana berikut :

```
program welcome;
uses wincrt;

begin
     writeln('*******************************');
     writeln('Selamat belajar bahasa Pascal!!');
     writeln('*******************************');
end.
```

Dari code diatas dapat dilihat bahwa bagian terpenting dari struktur pascal adalah **program, uses, begin** dan **end**. Dan bila code di atas di compile maka outputnya adalah :

```
*******************************
Selamat belajar bahasa Pascal!!
*******************************
```

C. IDENTIFIER DALAM PASCAL

Seperi dijelaskan sebelumnya, bahwa di dalam pendeklarasian program, label, const, type, var, procedure, dan function harus mempunyai sebuah *identifier*. *Identifier* adalah sebuah pengenal atau nama dari bagian-bagian tersebut.

Identifier harus memenuhi syarat-syarat sebagai berikut :

- Panjang karakter tidak melebihi 255 buah karakter.
- 63 buah karakter pertama adalah karakter yang signifikan.
- Karaker pertama harus berupa huruf (alfabet).
- Karakter yang diperbolehkan hanya huruf, angka dan undercore / garis bawah (_).
- Tidak terdapat spasi di dalam identifier.

D. RESERVED WORD

Perintah-perintah atau ekspresi yang digunakan di dalam Pascal sebenarnya terdapat di dalam *unit*. Jadi *unit* adalah sebuah basis yang menampung *librari* atau pustaka bahasa Pascal yang biasa disebut ekspresi atau perintah. Jadi untuk menulis suatu ekspresi, terlebih dahulu unit yang menampung ekspresi tersebut harus dideklarasikan. Unit dideklarasikan pada bagian *uses*. Ada tujuh unit yang disediakan di dalam Turbo Pascal For Windows ini, antara lain **strings**, **system**, **wincrt**, **windos**, **winprocs**, **wintypes** dan **wobjects**.

Reserved word adalah perintah-perintah atau ekspresi cadangan yang tersedia llangsung di dalam pascal tanpa melalui *unit*. Berikut ini adalah reserved word yang biasa digunakan : and, array, begin, case, const, div, do, downto, else, end, for, function, goto, if,

in, label, mod, not, of, or procedure, program, record, repeat, set, shl, shr, string, then, to, type, unit, until, uses, var, while, with, xor, dll.

E. TIPE DATA

Coba perhatikan listing program sederhana berikut :

```
program tipe_data;
uses wincrt;
var   a:integer;
      b:real;
      c:string;
      d:boolean;
      e:array[1..100] of integer;
begin
      a:= 100;
      b:= 100.1;
      c:= 'PASCAL';
      d:= TRUE;
end.
```

Pada bagian deklarasi var terdapat identifier-identifier yaitu a, b, c, d dan e. Setelah penulisan identifier, dituliskan tipe-tipe identifier tersebut dengan dipisahkan tanda titik dua (*colon*). Tipe-tipe tersebut diatas antara lain integer, real, string, boolean dan array.

Tipe adalah sifat atau karakteristik yang berfungsi membatasi nilai atau jangkauan dari suatu identifier. Tipe dibedakan menjadi 9 (sembilan) jenis tipe, antara lain : **array**, **file**, **object**, **ordinal**, **pointer**, **real**, **record**, **set** dan **string**.

Dalam struktur pascal, bagian yang biasanya menggunakan tipe setelah identifier adalah bagian const, type, var, procedure dan function. Pada bagian ini akan dibahas lebih dulu penggunaan tipe pada deklarasi variabel. Dan tipe yang dibahas pada bagian ini adalah tipe ordinal, real dan string.

Variabel adalah suatu identifier yang mewakili suatu nilai dengan disertakan tipe-nya dan dideklarasikan pada bagian var. Jadi, pada contoh code di atas, identifier a, b, c, d dan e adalah variabel.

1. Tipe Ordinal

Tipe ordinal adalah tipe numerik yang bernilai suatu angka bulat negatif atau positif. Menurut jangkauan nilainya, tipe ordinal dibagi menjadi 7 (tujuh) tipe :

Tipe	Range / Jangkauan	Size / Ukuran
Shortint	-128 .. 127	8 bit
Byte	0 .. 255	8 bit
Integer	-32768 .. 32767	16 bit
Word	0 .. 65535	16 bit
Longint	-2147483648 .. 2147483647	32 bit
Boolean	TRUE .. FALSE	8 bit
Chr	ASCII	8 bit

Khusus untuk tipe boolean dan chr, tipe ini adalah tipe ordinal predefine atau tidak terdefinisi. Jadi chr dan boolean bisa dianggap bukan tipe ordinal karena tidak bisa digunakan pada operasi numerik. Sedangkan tipe ordinal selain boolean dan chr biasa disebut tipe integer.

Apabila nilai yang diberikan pada variabel tidak sesuai dengan jangkauan tipe varibel tersebut maka akan menimbulkan error. Berikut contoh listing code yang salah dalam pemakaian tipe variabel :

```
program tipe_variabel_salah;
uses wincrt;
var  x:integer;
     y:byte;
     z:longint;
begin
     x:= 32768;
     y:= -1;
     z:= -2200000;
end.
```

2. Tipe Real

Tipe real adalah tipe numerik yang mencakup bilangan real, baik itu bilangan bulat atau pecahan. Ciri yang membedakan dengan tipe ordinal yaitu bahwa tipe real selalu mempunyai nilai dibelakang koma, baik itu bulat atau pecahan. Maka dari itu penulisan output nilainya menggunakan notasi *floating-point* atau eksponensial. Perhatikan contoh penulisan nilai bertipe real berikut ini :

Bilangan Real	Penulisan Pada Tipe Real
25	2.5000000000E+01
10.5	1.0500000000E+01
0.0015	1.5000000000E-03
0.0705	7.0500000000E-02
2500.017009	2.5000170090E+03

Dalam penulisan pecahan terdapat tanda koma. Dalam Pascal tanda ini tidak memakai tanda koma biasa melainkan memakai **tanda titik**. Tipe real dibagi menjadi 5 (lima) menurut jangkauannya, yaitu :

Tipe	Range / Jangkauan	Size / Ukuran
Real	2.9e-39..1.7e38	6 bit
Single	1.5e-45..3.4e38	4 bit
Double	5.0e-324..1.7e308	8 bit
Extended	3.4e-4932..1.1e4932	10 bit
Comp	-9.2e18..9.2e18	8 bit

3. Tipe String

Tipe string adalah tipe variabel yang berupa serangkaian karakter yang bersifat teks. Panjang maksimal karakter dalam tipe string adalah 255 karakter. Ciri dari suatu nilai yang bertipe string adalah penggunaan **tanda kutip satu (aphostrope)** yang diberikan di awal dan di akhir string. Untuk lebih jelas, perhatikan listing code berikut :

```
program teks;
uses wincrt;
var  s:string;
     t:string[5];

begin
     s:= 'Bahasa Pascal';
     t:= 'SixiZ';
end.
```

Variabel **s** dengan variabel **t** berbeda dalam pendeklarasiaannya. String **t** memakai *square bracket* atau tanda kurung balok dengan isi 5. Angka di dalam *square bracket* adalah batasan panjang karakter string.

F. DEKLARASI VARIABEL

Pada contoh-contoh listing code sebelumnya, sudah banyak yang menjelaskan pendeklarasian variabel. Deklarasi suatu variabel beserta tipenya dilakukan di bagian var.

Ada dua cara dalam pendeklarasian variabel yaitu :

```
var id : type;
var id1,id2,..,idn : type;
```

Coba perhatikan listing code berikut :

```
program deklarasi_variabel_salah;
uses wincrt;
var   x,y:integer;
      a:real;
      s,t:string[50];
      tanya:boolean;
begin
      x:= 50;
      a:= x;
      y:= a;
      s:='Bahasa Pascal';
      tanya:= s;
      t:= TRUE;
end.
```

Variabel yang bertipe sama bisa dideklarasikan sekaligus dalam satu pendeklarasian. Pada contoh diatas bisa dilihat pada variabel **a** dan **b**, **s** dan **t**, **tanya** dan **error**.

Kemudian lihat pada bagian program utama, bagaimana cara variabel diberikan ekspresi atau statement. Misal pada variabel **x** yang bernilai 50, kemudian variabel **y** bernilai **x**. Karena nilai **x** itu 50 maka nilai **y** juga 50.

Namun apabila suatu variabel bernilai variabel yang lain yang tipenya berbeda, maka akan menimbulkan error. Terkecuali pada tipe real yang bisa mencakup nilai yang beripe ordinal. Perhatikan listing code berikut! Mana ekspresi atau statement yang salah?

```
program deklarasi_variabel;
uses wincrt;
var   x:integer; y:integer;
      a,b:real;
      s,t:string[50];
      tanya,error:boolean;
begin
      x:= 50;
      y:= x;
      b:= 12.50;
      a:= b;
      s:='Bahasa Pascal';
      tanya:= true;
end.
```

III OPERASI DASAR

Untuk memperlancar penganalisaan dan cara berpikir logika yang benar dalam pemograman bahasa Pascal, maka harus tahu terlebih dahulu fungsi dan prosedur dasar atau operasi dasar dari pascal. Untuk operasi dasar atau fungsi dan prosedur dasar akan diperkenalkan prosedur input-output, fungsi standar tipe data dan operasi logika.

Fungsi dan prosedur yang akan dibahas ini telah tersedia di dalam pascal atau yang biasa disebut *reserved word*. Jadi tanpa menggunakan suatu unit, operasi tersebut bisa dipergunakan.

A. OPERASI INPUT DAN OUTPUT

Input adalah masukan data ke dalam proses suatu aplikasi. Karena masukan berarti media yang digunakan adalah *keyboard*. Dari keyboard-lah data dimasukkan. Data yang dimasukkan tergantung program yang dijalankan.

Output adalah keluaran data yang dihasilkan dari proses suatu aplikasi. Berbeda dengan input, karena output adalah keluaran maka media yang digunakan adalah monitor atau printer. Data keluaran yang dihasilkan tergantung apa yang dihasilkan suatu aplikasi.

Prosedur untuk input standar ada tiga yaitu *read*, *readln* dan *readkey*. Prosedur input ini adalah prosedur untuk membaca suatu masukan berupa variabel. *Read*, *readln* dan *readkey* pada operasi biasa sama menghasilkan baris baru setelah masukannya. Akan tetapi pada operasi file akan berbeda fungsi. Syntax-nya sebagai berikut :

```
read(var1,var2,..,varn);
readln(var1,var2, ..,varn);
readkey(var1,var2, ..,varn);  è  Hanya untuk tipe char
```

Prosedur untuk output pada layar monitor ada dua yaitu *write* dan *writeln*. Kedua prosedur ini digunakan untuk menampilkan suatu output ke layar baik itu berupa variabel, teks/string ataupun angka. Write menampilkan output tanpa menambah baris baru setelah menampilkan outputnya. Sedangkan writeln menambah baris baru setelah menampilkan outputnya. Bentuknya sebagai berikut :

```
write(var1,var2,..,varn);
writeln(var1,var2, ..,varn);
```

Coba perhatikan listing code program sederhana berikut ini :

```
program input_output; uses wincrt;
var   x:integer;   s:string;
begin
      s:='Thanks much...!';
      writeln('Hallo newbie !!');
      write('Berapa usia anda ? ');
      read(x);
      Writeln('Ooo... Usia anda ',x,' tahun');
      Write(s);
end.
```

Bila listing code diatas di compile maka akan tampil output program berikut :

```
Hallo newbie !!
Berapa usia anda ? _
```

Anda akan menemukan kursor berkedip-kedip. Kursor tersebut adalah hasil dari operasi **readln(x)**. Dan yang anda lakukan sekarang adalah memasukan input untuk variabel **x**, misalnya adalah **17**. Kemudian tekan enter maka muncul keluaran berikutnya :

```
Hallo newbie !!
Berapa usia anda ? 17
Ooo... Usia anda 17 tahun
Thanks much...!
```

Dari hasil kompilasi program diatas dapat diketahui bahwa read dan write berbeda. Write mencetak atau menghasilkan keluaran, sedangkan read membutuhkan sebuah masukan menurut tipe variabel yang digunakan.

Penulisan keluaran pada prosedur write dan writeln ada dua. *Pertama*, penulisan teks biasa atau string. Penulisan string harus memakai tanda kutip satu (atrhostope). *Kedua*, penulisan angka dan variabel. Penulisan angka (numerik) dan variabel tidak perlu memakai tanda atrhostope. Sebagai pemisah digunakanlah tanda koma.

Untuk input anda harus memasukkannya sesuai dengan tipe variabelnya. Bila masukan input tidak sesuai dengan tipe-nya maka akan menimbulkan *run-time error*.

B. OPERASI BINER

Operasi biner yaitu operasi perhitungan yang menghubungkan dua nilai yang bertipe sama. Ada 6 (enam) operator biner dan , yaitu :

Operator	Operasi	Tipe Yang Dioperasikan	Hasil operasi
+	Penambahan (Addition)	Integer	Integer
		Real	Real
-	Pengurangan (Substraction)	Integer	Integer
		Real	Real
*****	Perkalian (Multiplicaion)	Integer	Integer
		Real	Real
/	Pembagian (Division)	**Integer**	**Real**
		Real	Real
div	Pembagian Integer / Bulat (Integer Division)	Integer	Integer
mod	Sisa bagi (Remainder)	Integer	Integer

Tipe real dapat beroperasi pada operator-operator tersebut diatas kecuali pada mod dan div. Integer dapat beroperasi pada semua operator diatas. Akan tetapi, pembagian integer dengan integer akan menghasilkan nilai yang bertipe real. Karena dalam pembagian, ada kemungkinan integer menjadi pecahan/real.

```
program operasi_biner; uses wincrt;
var  x,y,z:integer; a,b,c:real;
begin
     x:= 26;
     a:= 5.5;
     y:= x + 4;
     b:= a + 4;
     x:= x mod 7;
     c:= c * x;
     Writeln(y,' ',z,' ',x,' ',c);
end.
```

Bila listing program tersebut dieksekusi / compile maka outputnya adalah :

```
30 0 5 0.00000000E+0
```

Mengapa nilai **z** kosong/nol?? Variabel **z** sudah dideklarasikan di var. Akan tetapi belum diberi nilai pada program utama. Karena variabel **z** belum diberi nilai, maka nilainya adalah kosong. Begitu juga dengan **c** yang bertipe real. Hasil dari **c * x** adalah nol, karena **c** belum mempunyai nilai. Coba perhatikan program berikut ini! Apakah outputnya?

```
program hitung_sendiri; uses wincrt;
var  x,y:byte; a,b:real; i,j:integer;
begin
     x:= 200;
     y:= x * 5;
     writeln(y);
     a:= 5.25;
     b:= a * 5;
     writeln(b:2:1);
     i:= 20;
     j:= ((i mod 3)*(3 mod 1)) div 2;
     writeln(j);
end.
```

Anda akan menemukan kejanggalan setelah mencompile program. Fokuskan perhatian anda pada variabel bertipe byte, kemudian pada operasi **writeln** yang ke-2.

Tipe ordinal mempunyai ciri khusus, apabila nilainya dioperasikan dan hasil operasi tersebut melebihi range-nya maka nilainya akan dihitung kembali dari awal. Contoh diatas pada tipe byte yang range-nya 256, maka operasi 200 * 5 akan melebihi range-nya yaitu 1000. Oleh karena itu 1000 mod 256 maka hasilnya adalah 232. Bingung..!?? J

C. OPERASI ORDINAL DAN KARAKTER

Operasi ini digunakan untuk pengurutan tipe data. Operasi ordinal dan karakter terdiri dari dua fungsi dan dua prosedur.

Operasi	Fungsi	Jenis Operasi
Succ	Mengangkat nilai ke nilai beikutnya satu point	Function
Pred	Menurunkan nilai ke nilai sebelumnya satu point	Function
Inc	Menambah nilai (increase)	Procedure
Dec	Mengurangi nilai (decrease)	Procedure
Odd	Memastikan pada nilai yang ganjil	

Perhatikan listing code program berikut :

```
program ord_kar; uses wincrt;
var  x:byte; a:char;
begin
     x:=255;
     writeln(succ(x));
     a:=#117;
     writeln(pred(a));
     dec(a);
     writeln(a);
     inc(x,5);
     writeln(x);
end.
```

Bila dicompile maka outputnya seperti ini:

```
0
t
t
4
```

Tanda # (pagar) digunakan untuk menyatakan simbol ASCII menurut urutannya.

D. OPERASI LOGIKA / BITWISE

Pada operasi logika, anda harus tahu mengenai bilangan biner. Karena operasi ini adalah operasi bilangan biner yang hubungannya dengan tipe boolean yaitu 1 dan 0.

1. Operasi Not

Yaitu operasi pembalikan bitwise atau biner. Bilangan positif akan menjadi negatif begitu juga bilangan negatif akan menjadi bilangan positif. Operasi not adalah operasi yang mempunyai hierarki tertinggi diantara operasi bitwise yang lainnya.

Operand X	Not X
1	0
0	1

```
program operasi_not;
uses wincrt;

begin
     writeln(not 23);
end.
```

Bila dicompile maka outputnya adalah **-24**.

Operasi not 23, dapat digambarkan :

Biner 23 è 00010111

Not 23 è 11101000 è -**24**

Dapat disimpulkan bahwa operasi not bekerja sebagai berikut :

v Bilangan positif dioperasikan menjadi bilangan negatif kemudian diambil 1.

v Bilangan negatif dioperasikan menjadi bilangan positif kemudian diambil 1.

2. Operasi And

Yaitu operasi untuk membandingkan dua buah elemen atau nilai, hasilnya akan true bila keduanya true (true diberi nilai 1 false diberi nilai 0). Perhatikan contoh :

Operand		X and Y
X	Y	
1	1	1
1	0	0
0	1	0
0	0	0

```
program operasi_and;
uses wincrt;

begin
     writeln(12 and 23);
end.
```

Bila dieksekusi maka outputnya adalah 4.

Biner 12 è 00001100

Biner 23 è 00010111

12 and 23 è 00000100 = **4** desimal

3. Operasi Or

Yaitu operasi untuk membandingkan dua buah elemen atau nilai, hasilnya akan true bila salah satu atau keduanya true (true diberi nilai 1 false diberi nilai 0).

Operand		X or Y
X	Y	
1	1	1
1	0	1
0	1	1
0	0	0

```
program operasi_or;
uses wincrt;
begin
     writeln(12 or 23);
end.
```

Bila dieksekusi maka outputnya adalah **31**. Cara kerjanya adalah :

Biner 12 è 00001100

Biner 23 è 00010111

12 or 23 è 00011111 = **31** desimal

4. Operasi Xor

Yaitu digunakan untuk membandingkan dua buah elemen atau nilai, hasilnya akan true bila salah satunya saja yang true (true diberi nilai 1 false diberi nilai 0).

Operand		X xor Y
X	Y	
1	1	0
1	0	1
0	1	1
0	0	0

```
program operasi_xor;
uses wincrt;
begin
     writeln(12 xor 23);
end.
```

Bila dieksekusi, outputnya adalah 27. Cara kerjanya adalah :

Biner 12 è 00001100
Biner 23 è 00010111
12 xor 23 è 00011011 = **27** desimal

5. Operasi Shl (Shift Left)

Yaitu operasi yang digunakan menggeser (shift) bit/digit ke arah kiri tergantung seberapa besar pergeserannya. Hasil pergeserannya yang ditinggalkan akan bernilai biner 0. Perhatikan contoh berikut :

```
program operasi_shl;
uses wincrt;
begin
    writeln(5 shl 6);
end.
```

Output program disamping yaitu 320. Ingat, hasil pembuangan digit/bit akan bernilai 0.

Cara kerjanya sebagai berikut :

Biner 5 è 0000000000000101
Digeser ke kiri 5 bit è 0000000101000000 = **320** desimal

6. Operasi Shr (Shift Right)

Kebalikan dari shl, shr yaitu operasi yang digunakan menggeser (shift) bit/digit ke arah kanan tergantung seberapa besar pergeserannya. Perhatikan contoh berikut :

```
program operasi_shr;
uses wincrt;
begin
    writeln(160 shr 6);
end.
```

Output program disamping yaitu 2. Cara kerjanya sebagai berikut :

Biner 160 è 0000000010100000
Digeser ke kiri 5 bit è 0000000000000010 = **2** desimal

E. OPERASI RELASI

Relasi artinya berhubungan. Operasi relasi berarti operasi yang menghubungkan antara dua nilai tertentu. Operasi relasi hampir sama penggunaannya dengan operasi logika. Operator relasi antara lain **=**, **<>**, **>**, **<**, **>=** dan **<=**. Anda pasti sudah kenal dengan simbol-simbol tersebut.

Apabila suatu operasi tersebut benar maka akan menghasilkan output true, sebaliknya bila salah maka akan menghasilkan output false. Perhatikan contoh berikut :

```
program operasi_relasi;
uses wincrt;

begin
     writeln(16 > 15);
     writeln(0 >= 2);
     writeln(a < A);
     writeln('Hack'>'Crack');
end.
```

Output program disamping yaitu:

```
TRUE
FALSE
FALSE
TRUE
```

Untuk output pada baris ke-3 dan ke-4 anda pasti bingung. Pada bagian operasi ordinal sudah disinggung mengenai bilangan ASCII, jadi.....?#$@!??

F. OPERASI ARITMATIKA DAN REAL

Ada beberapa fungsi yang berhubungan dengan perhitungan atau aritmatika selain operasi biner. Lihat tabel dibawah ini :

Operasi	Fungsi
Abs	Menjadikan suatu nilai menjadi mutlak (absolut)
Sin	Mencari harga sin suatu derajat sudut
Cos	Mencari harga cos suatu derajat sudut
ArcTan	Mencari harga suatu sudut dengan tan
Sqr	Perpangkatan dua (kuadrat)
Sqrt	Akar pangkat dua
Exp	Mencari nilai dari suau eksponensial
Ln	Mencari bilangan alam dari suatu nilai
Frac	Mengambil nilai dibelakang koma
Int	Mengambil nilai didepan koma
Round	Membulatkan nilai sesuai dengan nilai dibelakang koma. Jika >0.5 maka ke atas, jika <0.5 maka ke bawah
Trunc	Membulatkan nilai ke bawah
Chr	Merubah numerik ordinal ke dalam suatu karakter (char) ASCII
Ord	Merubah suatu karakter (ASCII) ke dalam suatu numerik

Fungsi chr dan ord sebenarnya bukan operasi aritmatika tetapi fungsi *konversi* dalam ASCII yaitu antara simbol-nya dengan urutannya. ASCII mempunyai 255 karakter yang dapat dipergunakan dengan notasi # (pagar) atau dengan fungsi chr dan ord. Fungsi chr juga bisa dipergunakan pada pengoperasian type dan const (Bab VII). Untuk lebih jelasnya perhatikan listing code berikut :

```
program operasi_aritmatika;
uses wincrt;
var   x:integer;
      s:char;
      a,b:real;
begin
      s:=#97
      a:=65.59;
      x:=round(a);
      b:=trunc(int(a))-round(frac(a)+frac(a+5));
      writeln(s);
      writeln(ord(s));
      writeln(x);
      writeln(b:2:2);
      writeln(chr(x));
end.
```

Output yang di dapat adalah :

```
a
97
66
64.00
B
```

Cobalah anda buat listing code program yang bisa mengkonversi simbol-simbol ASCII ke dalam urutannya (ordinalnya) sesuai dengan input yang dimasukkan!!! Misal masukan inputnya adalah **A**, maka outputnya adalah **65**.

IV PERULANGAN (LOOPING)

A. PERULANGAN FOR-DO

Looping for-do digunakan untuk mengulang statement berulang kali sejumlah yang ditentukan. Bila terdapat lebih dari satu statement yang ada di dalam looping maka menggunakan **begin...end;**, jika hanya satu blok saja tidak perlu menggunakan begin...end;. Ada 3 bentuk looping for-do yaitu for-do positif, for-do negatif dan for-do tersarang.

1. Looping Positif/Negatif

Looping positif adalah looping dari perhitungan kecil ke perhitungan besar. Sedangkan looping negatif kebalikan dari looping positif. Bentuknya sebagai berikut :

```
For var_int := nilai_awal to nilai_akhir do statement; (+)
For var_int := nilai_awal downto nilai_akhir do statement; (-)
```

Var_int adalah variabel kontrol yang menentukan looping tersebut. Variabel kontrol harus bertipe sejenis integer. Example code :

```
program looping_without_begin; uses wincrt;
var     x:integer;
begin
        For x:= 1 to 5 do writeln('Sixiz gitu lho..!!');
end.
```

```
program looping_with_begin; uses wincrt;
var     x:integer;
begin
        For x:= 5 downto 1 do
        Begin
           write(x);
           writeln(' Sixiz gitu lho..!!');
        end;
end.
```

Output program looping_without_begin :

```
Sixiz gitu lho..!!
Sixiz gitu lho..!!
Sixiz gitu lho..!!
Sixiz gitu lho..!!
Sixiz gitu lho..!!
```

Output program looping_with_begin :

```
5 Sixiz gitu lho..!!
4 Sixiz gitu lho..!!
3 Sixiz gitu lho..!!
2 Sixiz gitu lho..!!
1 Sixiz gitu lho..!!
```

Coba perhatikan listing program berikut :

```
program yang_mana_bukan_looping; uses wincrt;
var  x:integer;
begin
     For x:= 5 downto 1 do
          write(x);
     writeln(' Sixiz gitu lho..!!');
end.
```

Outputnya adalah **54321 Sixiz gitu lho..!!** Statement **writeln(' Six...)** tidak termasuk ke dalam looping karena looping **x** berakhir pada satement **write(x);**.

Sekarang anda analisis output dari kedua lsiting code program berikut.

```
program mikir_donk; uses wincrt;
var  x,y:byte;
begin
     for x:= 10 downto 1 do y:=y+(x-1);
     writeln(x);
end.
```

```
program perpangkatan; uses wincrt;
var  x,x2,x3:integer;
begin
     writeln('--------------');
     writeln('  x   x2    x3');
     writeln('--------------');
     for x:= 1 to 10 do
     begin
          x2:=x*x;
          x3:=x2*x;
          writeln(x:3,x2:5,x3:6);
     end;
end.
```

2. Looping For-Do Tersarang (Nested For)

Looping tersarang yaitu looping yang ada di dalam looping lainnya. Looping yang lebih dalam akan diproses terlebih dahulu hingga habis kemudian looping yang lebih luar bertambah atau berkurang dan memproses kembali looping yang paling dalam. Untuk lebih jelasnya, perhatikan listing code program berikut :

```
program looping_tersarang;
uses wincrt;
var  i,j:integer;

begin
     for i:= 1 to 2 do
     begin
         for j:=1 to 3 do
         write(i,j,' ');
         writeln;
     end;
end.
```

Bila dicompile, maka outputnya :

```
11 12 13
21 22 23
```

Dengan digit/angka yang kiri adalah **i** dan yang kanan adalah **j**. Bagaimana bisa terjadi demikian?!??

Looping disamping bisa digambarkan seperti berikut :

Untuk i=1	è	Untuk j=1	è	write(i,j,' ');	=	**11**	
		Untuk j=2	è	write(i,j,' ');	=	**12**	
		Untuk j=3	è	write(i,j,' ');	=	**13**	writeln;
Untuk i=2	è	Untuk j=1	è	write(i,j,' ');	=	**21**	
		Untuk j=2	è	write(i,j,' ');	=	**22**	
		Untuk j=3	è	write(i,j,' ');	=	**23**	writeln;

Bingung khan..?!?? Cara kerja program diatas yaitu cabang looping harus didahulukan operasi loop-nya kemudian baru sub-nya. Looping tersebut berakhir sampai sub dan cabang bernilai akhir. Sekarang analisis kedua listing code berikut :

```
program triple_sum;
uses wincrt;
var  i,j,k:integer;
begin
     for i:= 5 downto 1 do
     begin
         for j:=1 to 3 do
             k:=k+(i+j);
         writeln(k);
     end;
end.
```

```
program kombinasi;
uses wincrt;
var  i,j,k:integer;
begin
     for i:= 1 to 3 do
        for j:=3 downto 1 do
            for k:=i to j do
            writeln(i,j,k);
end.
```

B. PERULANGAN WHILE-DO

Looping dengan while-do mempunyai bentuk seperi berikut :

while *ekspresi_logika/relasi* do *statement*;

Statement while-do digunakan untuk melakukan proses looping suatu statemen terus-menerus selama ekspresi_logika/relasi bernilai benar atau belum terpenuhi.

```
program while_do;
uses wincrt;
var  i:byte;

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

Outputnya adalah :

```
0
1
2
3
4
```

Ekspresi relasi-nya yaitu **i < 5**. Jadi jika **i** masih lebih kecil dari **5** maka statement akan terus diproses berulang, dan proses akan berhenti setelah **i** tidak lagi lebih besar dari 5 (**i > 5**).

Maka dari itu di dalam statement ada **i:=i+1;** yang akan menambahkan nilai **i** satu persatu supaya **i** > **5**. Lihat code berikut yang mebaca input dan mengoperasi-kannya. Kemudian ada penyeleksian input jika memenuhi akan diulangi jika tidak akan berhenti.

```
program cel_to_fah; uses wincrt;
var  cel,fah:real; lagi:char;
begin
     lagi:='y';
     while lagi='y' do
     begin
          clrscr;     {prosedur untuk menghapus layar}
          write('Nilai celcius ? ');
          readln(cel);
          fah:=1.8 * cel + 32;
          writeln('Fahrenheit = ',fah:5:2);
          write('Menghitung lagi (y/t) ? ');
          readln(lagi);
     end;
end.
```

Di dalam looping while-do juga terdapat while-do tersarang (nested while-do). Dan pada prinsipnya adalah sama dengan nested for-do.

C. PERULANGAN REPEAT..UNTIL

Repeat..Until adalah proses looping suatu statement secara terus menerus hingga ekspresi yang ada di dalam until bernilai false atau sudah terpenuhi. Dengan kata lain looping repeat..until prosesnya berkebalikan dengan looping while-do. Bentuknya:

Repeat *statement;* until *ekspresi_logika/relasi;*

Perhatikan contoh berikut :

```
program repeat_until;
uses wincrt;
var  i:byte;

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

Output yang dihasilkan adalah :

```
1
2
3
4
5
```

Proses looping akan terus dilakukan hingga ekspresi terpenuhi. Pada contoh, penambahan **i** terus dilakukan hingga **i=5**.

Sekarang pada program repeat_until, coba anda ganti ekspresi di dalam until menjadi **i>5**. Kemudian anda compile dan apa yang terjadi ??! Coba bandingkan contoh looping menggunakan repeat..until dengan looping while-do ini!!

```
program sample_repeat;
uses wincrt;
var  i:integer;
begin
     i:= 10;
     repeat
         writeln(i);
         i:=i + 1;
     until i>5;
end.
```

Output :

```
11
```

```
program sample_while;
uses wincrt;
var  i:integer;
begin
     i:= 10;
     while i<5 do
     begin
            writeln(i);
            i:=i+1;
     end;
end.
```

Output : Tidak ada hasil

Dari kedua contoh diatas dapat disimpulkan perbedaan kedua operasi tersebut :

- Paling sediki statement di dalam looping repeat diproses sekali, karena pemenuhan syarat until terletak setelah statement. Sedangkan looping while statement-nya paling sedikit dikerjakan nol kali, karena pemenuhan syarat while terletak sebelum statement.
- Looping *repeat* tidak memerlukan blok statement (begin..end;) pada statement yang lebih dari satu, karena batasnya sudah ditunjukkan oleh repeat..until. Sedangkan *while* memerlukan blok statement (begin..end;) jika statementnya lebih dari satu.

Didalam looping *repeat..until* juga terdapat looping tersarang (*nested repeat*). Sama seperti *nested for* dan *nested while* dalam hal aturan, yaitu di dalam looping terdapat looping yang lain. Perhatikan code program berikut yang menggunakan nested repeat. Program ini akan menghitung sisi-sisi segitiga siku-siku dengan menggunakan rumus phytagoras yaitu $C = A^2 + B^2$.

```
program phytagoras; uses wincrt;
var  a,b,c:real;
begin
     writeln('----------------------');
     writeln('Sisi A  Sisi B  Sisi C');
     writeln('----------------------');
     a:=1;
     repeat
          b:=0;
          repeat
               c:=sqrt(a*a + b*b);
               writeln(a:5:2,b:8:2,c:8:2);
               b:=b + 1;
          until b>4;
          a:=a + 1;
     until a>4;
end.
```

Setelah anda memahami program tersebut, coba anda analisis proses dari program berikut ini :

```
program bengong; uses wincrt;
var  a,b:byte;
begin
     a:=10;
     b:=0;
     repeat
          b:=b+1;
          a:=a-b;
     until a>b;
     writeln(a,' ',b);
end.
```

V PENYELEKSIAN KONDISI

Penyeleksian kondisi adalah penentuan kondisi yang memenuhi dalam penyeleksian. Penyeleksian kondisi ini hubungannya dengan operasi relasi yang membandingkan antara dua kondisi. Di dalam Pascal, penyeleksian kondisi menggunakan statement *If* dan statement *Case*.

A. STATEMENT IF

If atau jika yaitu penyeleksian dengan menggunakan logika pengandaian. Statement if mempunyai dua bentuk yaitu *If-Then* dan *If-Then..Else*. Seperti dalam looping, statement if juga mempunyai bentuk tersarang (*nested if*).

1. Struktur If-Then

Bentuk dari bentuk statement If-Then yaitu :

```
if kondisi then statement;
```

Bila kondisi memenuhi atau bernilai true maka statement akan diproses. Dan bila statement lebih dari satu maka diberlakukan blok statement (*begin..end;*). Contoh:

```
program if_then; uses wincrt;
var  nilai:real;    ket:string;
begin
     ket:='Tidak Lulus';       {nilai awal ket}
     write('Nilai yang didapat ? ');
     readln(nilai);
     if nilai >= 60 then ket:='Lulus';
     writeln(ket);
end.
```

Bila program diatas dijalankan :

```
Nilai yang didapat ? 60
Lulus
```

Perhatikan contoh yang kedua berikut :

```
program seleksi_nilai; uses wincrt;
var  nilai:real; ket:string;
begin
     ket:='Sangat Kurang Ajar!';
     write('Nilai yang didapat ? ');
     readln(nilai);
     if nilai > 30 then ket:='Kurang';
     if nilai >= 60 then ket:='Cukup Baik';
     if nilai >= 70 then ket:='Baik';
     if nilai >= 80 then ket:='Sangat Baik';
     if nilai = 100 then ket:='Luar Biasa!!';
     writeln(ket);
end.
```

Mungkin, sampai contoh yang kedua belum ada masalah dalam logika anda. Sekarang perhatikan ode berikut yang mencampurkan statement *for* dan statement *if* !

```
program segi_siku2; uses wincrt;
var i,j:integer;
begin
    for i:=1 to 5 do
       for j:=1 to i do
          if j=i then writeln(j)
          else then write(j);
end.
```

Output yang didapat adalah :

```
1
12
123
1234
12345
```

Cara pengerjaannya sama dengan program **looping_tersarang** pada Bab V. Hanya saja statement for untuk **j** memakai statement if. Kemudian perhatikan contoh penggunaan operasi logika/bitwise pada statement if berikut ini :

```
program my_language; uses wincrt;
var  s:string;
begin
     write('Pemograman apa ? ');
     readln(s);
     if (s='pascal') or (s='PASCAL') then
         writeln('Tepat sekali!!');
     if not((s='pascal') or (s='PASCAL')) then
         writeln('Terlalu mudah tuch..!');
end.
```

Bila program logic_3 dijalankan :

```
Pemograman apa ? pascal      Pemograman apa? visual basic
Tepat sekali!!               Terlalu mudah tuch..!
```

Or berarti 'atau' yaitu bila salah satu benar maka akan diproses. **Not** berarti 'bukan' yaitu bila bukan yang dimaksud maka akan diproses.

2. Struktur If-Then..Else

Tidak berbeda jauh dengan sruktur *if-then*, hanya ada penambahan *else* atau yang berarti 'lainnya'. Bentuknya adalah sebagai berikut :

```
if kondisi then statement_1 else statement_2;
```

Fungsi else disini adalah menampung statement yang akan dioperasikan bila kondisi tidak terpenuhi atau false. Jika memakai else, maka statement_1 tidak diperbolehkan adanya tanda akhir statement (;). Contoh :

```
program if_then; uses wincrt;
var  nilai:real;
begin
     write('Nilai didapat ? ');
     readln(nilai);
     if nilai >= 60 then
         writeln('Lulus')
     else
         writeln('Tidak lulus');
end.
```

Bila program dijalankan :

```
Nilai didapat ? 59
Tidak Lulus
```

Perhatikan contoh kedua, pengembangan dari program **logic_3** :

```
program gak_ngaruh; uses wincrt;
var  s:string;
begin
     write('Pemograman apa ? ');
     readln(s);
     if (s='pascal') or (s='PASCAL') then
         writeln('Tepat sekali!!')
     else
         writeln('Tidak terlalu buruk');
end.
```

3. Sruktur If Tersarang (Nested If)

Seperti halnya looping, if juga mempunyai nested if. Ada 4 (empat) bentuk nested if yang mungkin bisa dilakukan, yaitu ;

```
IF kondisi_1 THEN
   IF kondisi_2 THEN
        statement_1
   ELSE
        statement_2;
ELSE
   statement_3;
```

```
IF kondisi_1 THEN
BEGIN
   IF kondisi_2 THEN
        statement_1
   ELSE
        statement_2;
END;
```

```
IF kondisi_1 THEN
BEGIN
   IF kondisi_2 THEN
      IF kondisi_3 THEN
         statement_1
      ELSE
         statement_2;
   END;
END;
```

```
IF kondisi_1 THEN
   IF kondisi_2 THEN
        statement_1
   ELSE
        statement_2;
```

Program dibawah ini adalah contoh dari penggunaan nesed if. Program berikut akan memberitahukan kepada anda status bilangan yang anda masukkan.

```
program receh; uses wincrt;
var x:integer;
begin
    write('Masukkan bilangan bulat positif : ');
    readln(x);
    if x>1000 then writeln('Input lebih besar dari 1000')
    else
        if x>=100 then writeln('Input antara 100 hingga 1000')
        else
            if x>=10 then writeln('Input antara 10 hingga 100')
            else writeln('Input satuan');
end.
```

B. STATEMENT CASE

Perbedaan case dengan if adalah susunan case lebih. Bentuknya :

```
CASE ekspresi OF
     case_label1 : statement_1;
     case_label2 : statement_2;
     ..
END;
```

Struktur case-of mempunyai ekspresi logika yang kemudian dihubungkan pada case label sebagai nilai dari ekspresi tersebut. Case label dapat berupa sebuah nilai, konstanta atau range/jangkauan nilai yang bukan beripe real. Contoh case label :

```
1 :                  { nilai inteeger 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'..'C','a'..'c' :  { nilai karakter A,B,C,a,b,c }
'*' :                { nilai karaketr * }
```

Perhatikan contoh penggunaan case of pada program berikut :

```
program case_of; uses wincrt;
var  nilai:integer;
begin
     write('Nilai ulangan ? ');
     readln(nilai);
```

```
        case nilai of
             0..40 : writeln('Sangat kurang ajar!');
             41..59 : writeln('Kurang!');
             60..75 : writeln('Cukup');
             76..89 : writeln('Baik');
             90..99 : writeln('Hampir sempurna!');
             100 : writeln('Luar biasa!!!');
        end;
   end.
```

Bila dijalankan :

```
Nilai ulangan ? 85                  Nilai ulangan ? 10
Baik                                Sangat kurang ajar!
```

Bila input yang dimasukkan pada program **case_of** tidak sesuai atau tidak ada dalam case label maka tidak akan diproses. Supaya input yang tidak sesuai dengan case label diproses, maka diberlakukan struktur *Case-Of..Else*.Perhatikan program berikut yang akan menanyakan input pilihan (case label).

```
   program luas_2_dimensi; uses wincrt;
   var  pilih:byte; r,l,t,luas:real;
   begin
        writeln('    <<< Pilihan >>>');
        writeln('1. Luas lingkaran');
        writeln('2. Luas segitiga');
        write('Pilih nomor (1-2) ? ');
        readln(pilih);
        case pilih of
        1 :  begin
                  write('Jari-jari lingkarang ? ');
                  readln(r);
                  luas:=(22/7)*r*r;
                  writeln('Luas lingkaran = ',luas:9:2);
             end;
        2 :  begin
                  write('Panjang sisi alas ? ');
                  readln(l);
                  write('Tinggi segitiga ? ');
                  readln(t);
                  luas:=(l*t)/2;
                  writeln('Luas segitiga = ',luas:9:2);
             end;
        else begin
                  writeln('Pilihannya cuma 1 sampai 2!!');
                  writeln('Ngawur!!!');
             end;
        end;
   end.
```

Gimana..?? Huh..Buanyak bangget...! J Bila program dijalankan :

```
 <<<Pilihan>>>                      <<<Pilihan>>>
1. Luas lingkaran                  1. Luas lingkaran
2. Luas segitiga                   2. Luas segitiga
Pilih nomor (1-2) ? 2              Pilih nomor (1-2) ? 14
Panjang sisi alas ? 4.5            Pilihannya cuma 1 sampai 2!!
Tinggi segitiga ? 2.5              Ngawur!!!
Luas segitiga = 5.63
```

---<<\ J J J />>---

VI OPERASI STRING

A. MERANGKAI DAN MENYELEKSI STRING

Merangkai string berarti menyatukan beberapa buah string baik itu string langsung maupun variabel. Operator yang digunakan untuk adalah operator plus (**+**). Contoh :

```
program concatenated; uses wincrt;
var  s,t,u:string;
begin
     s:='Programmer';
     t:='Bahasa';
     u:='Pascal';
     writeln(s+' pemula harus bisa '+t+' '+u);
end.
```

Bila program running maka :

```
Programmer pemula harus bisa Bahasa Pascal
```

Menyeleksi string berarti membedakan antara kedua string, apakah sama atau tidak baik dalam ukuran, dan urutan. Urutan string disesuaikan dengan urutan simbol ASCII.

```
program seleksi_string; uses wincrt;
var  me,myfren:string;
begin
     write('Nama anda ? '); readln(me);
     write('Nama fren anda ? '); readln(myfren);
     if me = myfren then
          writeln('Anda sama dengan fren anda!');
     if me < myfren then
          writeln('Anda lebih kecil dari fren anda!');
     if me > myfren then
          writeln('Anda lebih besar dari fren anda!');
end.
```

Bila dijalankan :

```
Nama anda ? Joni
Nama fren anda ? Jono
Anda lebih kecil dari fren anda!
```

B. FUNGSI UNTUK OPERASI STRING

1. Larik String

Larik string yaitu urutan karakter didalam string. Larik string dinyatakan :

***var_string*[indeks]**

Perhatikan contoh berikut :

```
program get; uses wincrt;
var  s:string;
begin
     s:='sixiz.home';
     writeln(s[5]);
     s[6]:='@';
     writeln(s);
end.
```

Bila dijalankan, outputnya :

```
z
sixiz@home
```

z adalah indeks ke-5 dari variabel s. Indeks ke-6 variabel s diganti oleh karakter @ dan menjadi sixiz@home.

2. Fungsi Length

Fungsi length digunakan untuk menghasilkan panjang string dan menyimpannya dalam variabel bertipe integer. Bentuk umum fungsi length adalah :

Length(*var*:string):integer;

Perhatikan contoh berikut :

```
Program how_long_ur_name; uses wincrt;
var  s:string; x:byte;
begin
     write('Nama anda ? '); readln(s);
     x:=length(s);
     writeln('Panjang nama anda ',x,' karakter');
end.
```

Bila dirunning, maka :

```
Nama anda ? Bang Ozi
Panjang nama anda 8 karakter
```

Coba perhatikan kedua listing code berikut ini. Apakah outputnya ?

```
Program drop_right;
uses wincrt;
var s:string;
    i,j:byte;
begin
 s:='HACK';
 for i:=1 to length(s) do
      for j:=1 to i do
           write(s[j]);
end.
```

```
Program big_font; uses wincrt;
var s:string; i:byte;
begin
    s:='huruf besar';
    for i:=1 to length(s) do
        if  (s[i]>='a') and
            (s[i]<='z') then
            dec(s[i],32);
    write(s);
end.
```

3. Fungsi Concat

Fungsi concat gunanya sama dengan operator plus (+) yaitu untuk menyatukan atau merangkai beberapa string. Contoh :

```
program concat_string; uses wincrt;
var  s,t,u:string;
begin
     s:='Programmer';
     t:='Bahasa';
     u:='Pascal';
     writeln(concat(s,' pemula harus bisa ',t,' ',u));
end.
```

Outputnya sama dengan program **concatenated** pada operator +.

4. Fungsi Copy

Copy digunakan untuk menyalin sejumlah karakter ke variabel baru . Bentuk:

copy(*var*:string,*index*:integer,*count*:integer):string;

Var adalah variabel string yang akan dicopy. **Index** adalah letak awal karakter yang akan dicopy. **Count** adalah banyaknya karakter kedepan yang akan dicopy. Hasil salinan (copy-an) harus dimasukkan ke dalam variabel baru. Contoh :

```
program goank; uses wincrt;
var  sambel:string;
begin
    sambel:='cabebawangtomat';
    writeln(copy(bumbu,1,4));
    writeln(copy(bumbu,5,6));
    writeln(copy(bumbu,11,5));
end.
```

Bila dijalankan, output yang dihasilkan :

```
cabe
bawang
tomat
```

Coba compile listing code berikut ini. Apakah outputnya?!!

```
program balik; uses wincrt;
var  s,t:string; i:integer;
begin
     s:='retupmok';
```

```
        for i:=length(s) downto i do
        begin
            write(t);
            t:=copy(s,i,1);
        end;
    end.
```

5. Fungsi Pos

Fungsi ini digunakan untuk mencari posisi letak dari suatu string yang ada didalam string yang lain. Hasil dari fungsi ini bernilai integer. Bila nilainya 0 berarti letak string tidak ada. Contoh :

```
program where_r_u; uses wincrt;
var  alpha,a,b,c:string;
begin
     alpha:='ABCDEFGHIJKLMNOPQRSTUVWXYZ';
     a:='NOP'; b:='H'; c:='ABD';
     writeln(alpha);
     writeln(a,' diposisi ',pos(a,alpha));
     writeln(b,' diposisi ',pos(b,alpha));
     writeln(c,' diposisi ',pos(c,alpha));
end.
```

Bila dijalankan akan menghasilkan output :

```
ABCDEFGHIJKLMNOPQRSTUVWXYZ
NOP diposisi 14
H diposisi 8
ABD diposisi 0
```

C. PROSEDUR UNTUK OPERASI STRING

1. Prosedur Delete

Delete adalah prosedur untuk menghapus sejumlah karakter di dalam suatu string. Bentuknya adalah :

```
delete(var:string,index:integer,count:integer);
```

Var adalah variabel string yang akan dihapus karakternya. **Index** adalah letak awal karakter yang dihapus. **Count** adalah banyaknya karakter yang akan dihapus.

```
program little_down;
uses wincrt;
var   s:string; i:integer;
begin
      s:= 'sixiz.com';
      for i:=1 to length(s) do
      begin
          writeln(s);
          delete(s,10-i,1);
      end;
end.
```

Bila dirunning, outputnya :

```
sixiz.com
sixiz.co
sixiz.c
sixiz.
sixiz
sixi
six
si
s
```

2. Prosedur Insert

Insert digunakan untuk menyisipkan sejumlah karakter langsung atau variabel string ke variabel string lainnya sejumlah karakter di dalam suatu string. Syntaxnya :

```
insert(source:string,var:string,index:integer);
```

Source adalah sumber yang akan disisipkan, bisa berupa karakter langsung atau berupa variabel string. **Var** adalah variabel tujuan tempat disisipkannya karakter. **Index** adalah letak posisi awal tempat penyisipan. Contoh :

```
program penyusup; uses wincrt;
var  happy:string;
begin
     happy:='Tardjo anak pintar!';
     writeln(happy);
     insert('bukan ',happy,8);
     writeln(happy);
end.
```

Apakah anda bernama Tardjo??? Silahkan jalankan programnya.

3. Prosedur Str

Str digunakan untuk merubah nilai numerik menjadi nilai string. Syntaxnya :

```
str(x[:width[:decimal],var:string);
```

Variabel **x** adalah nilai yang akan dikonversi ke string. **Width** adalah panjang nilai didepan koma yang sudah terkonversi. **Decimal** adalah panjang dibelakang koma yang sudah terkonversi ke string. **Var** adalah variabel hasil yang bertipe string.

```
program sampel_str;
uses wincrt;
var  x,y:integer; a,b:string;
begin
     x:=1234;
     y:=567;
     writeln(x+y);
     str(x:4,a);
     str(y:4,b);
     writeln(a+b);
end.
```

Output program disamping yaitu :

```
1801
1234 567
```

4. Prosedur Val

Kebalikan dari *str*, *val* digunakan untuk merubah string ke numerik. Syantaxnya :

```
val(s:string,var,code:integer);
```

Variabel **s** adalah variabel string yang akan dikonversi. **Var** adalah variabel numerik hasil konversi, bisa real atau integer. Variabel **s** harus berisi karakter numerik, jika salah satu karakter ada yang bukan numerik maka akan muncul kesalahan. Letak karakter yang salah itu bisa terdeteksi oleh variabel **code**. Contoh :

```
program sampel_val;
uses wincrt;
var  s:string;
     x:real;
     code:integer;
begin
     s:=123.45';
     val(s,x,code);
     writeln('Nilai string : ',s);
     writeln('Nilai real : ',x);
     writeln('Posisi salah : ',code);
end.
```

Apa yang terjadi pada program diatas ??? Sekarang ubahlah nilai **s** menjadi **123.A5**. Selamat mencoba!

*****^^><^^*****

VII DEKLARASI CONST, TYPE DAN LABEL

Dari bab-bab sebelumnya bagian dari struktur yang belum dijelaskan adalah type, const, label, procedure dan function. Sekarang akan dijelaskan mengenai const, type dan label.

A. DEKLARASI CONST

Sama halnya dengan pendeklarasian variabel, pada bagian const juga terdapat identifier. Hanya saja tujuan dari pendeklarasian const hanyalah untuk memberikan nilai atau sebuah ekspresi yang bersifat **tetap**. Syntaxnya :

```
const identifier = ekspresi;
```

Contoh penggunaan const :

```
Const  data = 32767;
       maxdata = 1024 * 64;
       ordinal = ord('a') + ord('A');
       say = 'Hello fren';
       get : string[7] = 'hackers';
```

Jadi penggunaan konstanta (const) adalah memberikan nilai langsung kepada sebuah identifier tanpa harus mendaklarasikannya terlebih dahulu di bagian var dan kemudian memberikan nilainya di program utama. Contoh :

```
program sampel_const;
uses wincrt;
const  code:char='A';
begin
       inc(code);
       writeln(code);
       writeln(ord(code));
end.
```

Ouput yang dihasilkan :

```
B
66
```

```
program sampel_const;
uses wincrt;
const s:string='PASCAL';
var   i:integer;
begin
      for i:=1 to length(s) do
           dec(s[i]);
      writeln(s);
end.
```

Ouput yang dihasilkan :

```
PASCAL
```

B. DEKLARASI TYPE

Seperti deklarasi var, deklarasi type digunakan untuk memberikan tipe data pada suatu identifier. Akan tetapi identifier yang diberikan tipe data tersebut menjjaditipe data tersendiri. Tipe data terbagi 3, yaitu tipe data sederhana (*simple-type data*), tipe data terstruktur (*structure-type data*) dan tipe data penunjuk (*pointer-type data*).Tipe data sederhana terbagi dua lagi, yaitu tipe data standar (*standard data type*) yang terdiri dari tipe ordinal dan tipe string, dan tipe data didefinisikan pemakai (*user-define data type*).

```
type  pecahan = real;
      Nama = string[50];          } {standard data type}
      Logika = boolean;
      Hari = (Sun,Mon,Tue,Wed,Thu,Fri,Sat);
      Seminggu = 0..7;            } {user-defined}
      Masuk = Mon..Sat;
```

Tipe **hari** adalah tipe data skalar dan tipe **seminggu** dan **masuk** adalah tipe subrange. Isi dari ketiga tipe ini tidak dapat langsung ditampilkan dan diproses dibagian utama program, akan tetapi harus dideklarasikan terlebih dahulu dibagian **var**.

Contoh penggunaan Type :

```
program tipe_data; uses wincrt;
type  logika = boolean;
      hari = (Su,Mo,Tu,We,Th,Fr,Sa);
var   tanya:logika;
      uhari:hari;
begin
      tanya:=TRUE;
      writeln(succ(tanya));
      for uruthari:=Su to Sa do
         writeln('Hari ',ord(uhari));
end.
```

Output dari program disamping :

```
TRUE
Hari 0
Hari 1
Hari 2
Hari 3
Hari 4
Hari 5
Hari 6
```

Apabila pada **writeln** kedua anda tidak mencantumkan **ord**, maka akan menimbulkan error. Karena **uhari** isinya adalah tipe skalar jadi tidak boleh dtampilkan langsung oleh perintah **writeln**. Agar bisa ditampilkan langsung harus melalui larik.

Untuk lebih jelas baca manual help pada Turbo Pascal mengenai type.

C. DEKLARASI LABEL

Label adalah bagian statement tersendiri untuk mempermudah dalam menjalankan bagian-bagian tersebut. *Label* mempunyai identifier tetapi tidak berisikan tipe data. Dalam suatu program terstruktur, label sangat dibutuhkan agar mudah untuk 'meloncat' dari satu statement ke statement yang lain. Perintah untuk meloncat adalah '***goto label;***'. Contoh :

```
program stepby; uses wincrt;
label jump,stop;

begin
       writeln('Bahasa');
       goto jump;
       writeln('Basic');
       writeln('Foxpro');
jump:  writeln('Pascal');
       goto stop;
       writeln('Cobol');
stop:
end.
```

Output program disamping :

```
Bahasa
Pascal
```

Statement-statement setelah goto jump; akan dilewati dan akan langsung mengarah ke statement yang berlabel jump. Dan pada label stop, tidak ada statement apa-apa. Yang berari program dihentikan.

Perhatikan contoh berikut yang akan menyeleksi input. Jika input sesuai akan program mengulang, jika tidak program akan berhenti.

```
program cycle_out; uses wincrt;
label 1;
var   r,t,isi:real;
      jawab:char;

begin
1:    clrscr;
      write('Jari-jari lingkaran ? ');
      readln(r);
      write('Tinggi silinder ? ');
      readln(t);
      isi:=22/7*(r*r*t);
      writeln('Isi silinder = ',isi:9:2);
      writeln;
      write('Hitung lagi (y/t) ? ');
      readln(jawab);
      If (jawab='Y')or(jawab='y') then goto 1;
end.
```

Apa yang terjadi bila dijalankan ??!??#@$%&!%@

VIII PROSEDUR

A. DEKLARASI PROSEDUR

Procedure (prosedur) adalah suatu program yang terpisah dalam blok sendiri yang berfungsi sebagai program bagian. Bentuk umum dari procedure adalah :

```
Procedure identifier;
```

Contoh penggunaan procedure :

```
program sampel1; uses
wincrt;
procedure garis;
begin
     writeln('------');
end;
begin
     garis;
     writeln('Pascal');
     garis;
end.
```

Bila dijalankan :

```
------
Pascal
------
```

```
program sampel2; uses wincrt;
procedure kuadrat;
var   x,y:integer;
begin
     write('Nilai ? ');
     readln(x);
     writeln('Kuadrat = ',x*x);
end;
{ program utama }
begin
     kuadrat;
end.
```

Bila dijalankan :

```
Nilai ? 5
Kuadrat = 25
```

Pada program **sampel2**, didalam procedure terdapat variabel yang disebut variabel lokal. Apabila variabel lokal tersebut didekalrasikan di program utama maka akan terjadi error. Misal pada program **sampel2**, deklarasi **writeln(.....** dipindahkan ke program utama maka akan timbul **error 3 : unknown identifier**. Supaya variabel bisa digunakan di dua tempat tersebut, maka variabel harus dideklarasikan di program utama.

Akan tetapi bila program ingin mengakses variabel yang ada di procedure, maka procedure harus menggunakan parameter. Bentuknya :

```
Procedure identifier(x : type);          { By Value }
Procedure identifier(var x : type);      { By Reference }
```

Contoh by value :

```
program value1; uses wincrt;
procedure hitung(a,b:byte);
var c:byte;
begin
    c:=a+b;
    writeln('Nilai c = ',c);
end;

var x,y:byte;
begin
    write('Nilai x ? ');
    readln(x);
    write('Nilai y ? ');
    readln(y);
    hitung(x,y);
end.
```

Bila dijalankan :

```
Niali x ? 2
Nilai y ? 3
Nilai c = 5
```

```
program value2; uses wincrt;
procedure hitung(a,b,c:byte);
begin
    c:=a+b;
    writeln('a=',a,' b=',b,' c=',c);
end;

var x,y,z:byte;
begin
    x:=2;
    y:=3;
    z:=0;
    hitung(x,y,z);
    writeln('x=',x,' y=',y,' z=',z);
end.
```

Bila dijalankan :

```
a=2 b=3 c=5
x=2 y=3 z=0
```

Jadi, parameter berisi variabel yang digunakan oleh procedure. Paramater tersebut ada dua jenis, yaitu parameter yang berasal dari procedure (*parameter formal*) dan parameter yang berasal dari variabel-variabel program utama (*parameter nyata*).

Penggunaan parameter secara *by value*, berarti perubahan dalam parameter formal tidak akan merubah parameter nyata, seperti pada program **value1** dan **value2**. Sedangkan penggunaan parameter by reference, perubahan pada parameter formal akan ikut merubah parameter nyata. Contoh by reference :

```
program reference; uses wincrt;
procedure hitung(var a,b,c:byte);
begin
    c:=a+b;
end;

var x,y,z:byte;
begin
    x:=2;
    y:=3;
    hitung(x,y,z);
    writeln('x=',x,' y=',y,' z=',z);
end.
```

Output program disamping :

x=2 y=3 z=5

Meskipun variabel **z** belum diberi nilai, tetapi procedure telah memprosesnya menjadi bernilai **5**, karena variabel z menjadi parameter nyata untuk procedure hitung dan mewakili variabel **c**.

Sekarang, bagaimana jika by value dan by reference digabung. Perhatikan contoh :

```
program campur; uses wincrt;
procedure get(a,b:byte;var c:byte);
begin
    c:=a+b;
end;

var x,y,z:byte;
begin
    x:=2;
    y:=3;
    get(x,y,z);
    writeln('x=',x,' y=',y,' z=',z);
end.
```

Output program disamping :

x=2 y=3 z=5

Variabel **x** dan **y** hanyalah sebagai input untuk variabel **a** dan **b**. Sebagai hasil adalah variabel yang bisa berubah nilai yaitu variabel **z** dengan sifat *by reference*.

Telah dijelaskan bahwa type data variabel dideklarasikan di dalam parameter. Untuk type ordinal (integer, boolean dan char) dan type real dapat langsung dideklarasikan di bagian parameter. Akan tetapi untuk tipe data kompleks (string, array dan record) harus dideklarasikan terlebih dahulu dibagian *type*. Contoh :

```
program kompleks; uses wincrt;
type a = string;
procedure long(nama:a;var x:byte);
begin
     x:=length(nama);
end;

var  n:byte;
     myname:string;
begin
     write('Nama ente ? ');
     readln(myname);
     long(myname,n);
     writeln('Panjang nama ente ',n);
end.
```

B. PERTEMANAN PROSEDUR

Ada kemungkinan bila anda membuat suatu program ada 'pertemanan' diantara procedure didalam program anda. Perhatikan dan analisis contoh 'pertemanan' berikut :

```
program get_on; uses wincrt;
procedure get1(x1:byte);
begin
     writeln('x=',x1,' di prosedur get1');
end;
procedure get2(x2:byte);
begin
     writeln('x=',x2,' di prosedur get2');
     get1(x2);
end;
var  x:byte;
begin
     x:=5;
     get2(x);
end.
```

Bukan hanya antar prosedur saja 'pertemanan' itu terjadi. Prosedur juga bisa berteman (memanggil) dengan dirinya sendiri. Pemanggilan seperti ini disebut *rekursi*.

```
program looping; uses wincrt;
var  i:byte;
procedure rekursi;
begin
     writeln('Sixiz gitu..!');
     i:=i+1;
     if i<5 then rekursi;
end;
begin
     i:=1
     rekursi;
end.
```

Output program disamping :

```
Sixiz gitu..!
Sixiz gitu..!
Sixiz gitu..!
Sixiz gitu..!
```

Prosedur rekursi telah memanggil dirinya sebanyak 4 kali. Dengan ketentuan, rekursi akan terus dipanggil hingga i<5 tidak terpenuhi lagi.

Masih berhubungan dengan 'pertemanan' antar prosedur. Sekarang, bagaiman jika ada nested procedure (prosedur tersarang)??!?! Procedure dan function juga mempunyai 'versi' nested, dengan kata lain bertambahlah kebingungan anda... J Contoh :

```
program dizzy_nested; uses wincrt;
procedure first;
     procedure second;
     begin
          writeln('Second is in first');
     end;
     procedure third;
     begin
          writeln('Third is too');
     end;
begin
     writeln('It is first');
     second;
     third;
end;
begin
     writeln('It is main program');
     first;
end.
```

Output yang didapat :

```
It is main program
It is first
Second is in first
Third is too
```

--$<** J L J **>$--

IX FUNGSI

A. DEKLARASI FUNGSI

Fungsi (function) hampir sama dengan prosedur hanya ada dua perbedaan. *Pertama*, fungsi harus dideklarasikan dengan type-nya, karena fungsi akan menghasilkan suatu nilai. *Kedua*, fungsi harus memanggil dirinya sendiri untuk memberikan nilai pada dirinya tersebut. *Ketiga*, untuk diproses dibagian program utama, fungsi harus dideklarasikan seperti sebuah ekspresi. Sedangkan prosedur dideklarasikan sendiri.

Seperti halnya prosedur, fungsi bisa mempunyai parameter atau bisa juga tidak. Dan bila mempunyai parameter, ada dua kemungkinan yaitu berparameter *by value* atau berparameter *by reference*. Bentuk umum dari fungsi yaitu :

Function *identifier*(*parameter*):type;

Type adalah tipe data yang dihasilkan oleh fungsi tersebut. Dan **parameter** seperti halnya pada prosedur. Contoh pemakaian function :

```
program addition; uses wincrt;
function tung(var
a,b:byte):byte;
begin
    tung:=a+b;
end;
var x,y,z:byte;
begin
    write('Nilai x ? ');
    readln(x);
    write('Nilai y ? ');
    readln(y);
    z:=tung(x,y);
    writeln(x,'+',y,'=',z);
end.
```

Bila dijalankan :

```
Nilai x ? 4
Nilai y ? 5
4+5=9
```

Hampir mirip dengan prosedur. Tetapi mempunyai type tersendiri dan memberikan nilai sendiri. Kemudian pendeklarasian di bagian utama, berbeda dengan prosedur. Function harus 'diwakilkan' oleh suatu variabel, atau bisa langsung dideklarasikan oleh **writeln**.

Berikut contoh lain dari function :

```
program big_little; uses wincrt;
function big(x,y:real):real;
begin
    if x>y then big:=x
    else big:=y;
end;
var a,b,c:real;
begin
    write('Value 1 ? ');
    readln(a);
    write('Value 2 ? ');
    readln(b);
    c:=big(a,b);
    writeln('The Big = ',z:9:2);
end.
```

```
program calculate; uses wincrt;
function hi(var
a,b,c:word):word;
begin
    hi:=a+b;
    c:=a*b;
end;
var x,y,z,hit:word;
begin
    write('Value x ? ');
    readln(x);
    write('Value y ? ');
    readln(y);
    hit:=hi(x,y,z);
    writeln(x,'+',y,'=',hit);
    writeln(x,'*',y,'=',z);
end.
```

Apakah output dari kedua program diatas ???!? Tanya kenapa..!!??!?.

B. PERTEMANAN FUNGSI

Seperti halnya di dalam keluarga prosedur, yang mempunyai kekerabatan yang akrab. Fungsi juga mempunyai sifat seperti itu. Perhatikan contoh berikut :

```
program fren_lie;
uses wincrt;
function hex2(y:word):word;
begin
    hex2:=y*6;
end;

function hex1(x:word):word;
begin
    hex1:=hex2(x)*6;
end;

begin
    writeln(hex1(2));
end.
```

Output program disamping :

72

Bila digambarkan :

```
Hex1 := hex2(x) * 6;
Hex1 := (y * 6) * 6; { y = x }
Hex1 := (2 * 6) * 6;
Hex1 := 72;
```

Bila ditelusuri lebih jauh dan anda sudah terbiasa dengan pertemanan ini, maka kini anda beranjak ke *rekursi fungsi*. Yaitu fungsi berteman dengan dirinya sendiri.

Perhatikan contoh rekursi fungsi berikut :

```
program faktorial; uses wincrt;
function fak(x:word):real;
begin
     if x=0 then fak:=1
     else fak:=x*fak(x-1);
end;

var   n:word; z:real;
begin
     write('Faktorial ? ');
     readln(n);
     z:=fak(n);
     writeln(n,' Faktorial=',z:9:0);
end.
```

Bila dirunning : **Faktorial ? 5**
5 Faktorial=120

Bila digambarkan :

fak(5) => x<>0 => fak=5*fak(4) => fak=5*24 => **120**

fak(4) => x<>0 => fak=4*abc(3) => fak=4*6 => 24

fak(31) => x<>0 => fak=3*abc(2) => fak=3*2 => 6

fak(2) => x<>0 => fak=2*abc(1) => fak=2*1 => 2

fak(1) => x=0 => fak=1 => fak=1 => 1

Proses rekursi dihentikan karena b=0 dan abc(5,0)=1.

Bila disimpulkan : fak(5) = 5*(4*(3*(2*(1))))

= **120**

Eit....!!! Ini baru tingkat rekursi. Fungsi juga mempunyai nested function. Jadi kesimpulannya, bila anda menguasai nested procedure, insya Allah nested function juga bisa dikuasai.

L **>--** J **//--<**L

X LARIK / ARRAY

Larik atau *array* adalah tipe terstruktur yang terdiri dari sejumlah komponen yang mempunyai tipe yang sama. Banyaknya suatu komponen tersebut ditunjukkan oleh *indeks*. Array juga bisa diartikan sebagai sejumlah variabel yang terdapat dalam satu tipe atau variabel, maka dari itu ada indeks.

A. ARRAY SATU DIMENSI

Secara umum ada 3 bentuk array, yaitu sebagai berikut :

```
identifier : array[tipe_indeks] of type;              { var }
identifier : array[tipe_indeks] of type = (skalar);   { const }
identifier = array[tipe_indeks] of type;              { type }
```

Contoh sederhana pada bagian var :

```
var x:array[1..5] of integer;
```

Tipe indeks harus sesuai dengan type array. Seperti pada larik string, array diatas :

```
x[1] := 24;
x[2] := -2;
x[3] :=2000;
```

Seperti halnya prosedur dan fungsi, tipe data yang boleh langsung dideklarasikan hanyalah tipe ordinal dan real. Untuk string harus dideklarasikan pada bagian type dahulu.

Ex 1 :
```
type  huruf = string[30];
var   nama : array[1..10] of huruf;
```

Ex 2 :
```
type  batas = 17..90;
var   usia : array[1..1000] of batas;
```

Ex 3 :
```
var   usia : array[1..1000] of 17..90;
```

Ex 4 :
```
type  hari = (Sun,Mon,Tue,Wed,Thu,Fri,Sat);
var   Masuk : array[1..30] of hari;
```

Ex 5 :
```
type  hari = (Sun,Mon,Tue,Wed,Thu,Fri,Sat);
var   Masuk : array[mon..sat] of 1..6;
```

Ex 6 :
```
type  batas = 1..5;
      X = array[batas] of char;
var   nilai : x;
```

Ex 7 :
```
type  x = array[1..5] of char;
var   nilai : x;
```

Ex 8 :
```
const min = 1; Max = 5;
type  x = array[min..max] of char;
var   nilai : x;
```

Ex 9 :
```
var   x : array['#'..'}'] of byte;
```

Perhatikan contoh pemakaian array berikut :

```
program lipat; uses wincrt;
var   x:array[1..5] of byte;
      y:byte;
begin
      for y:=1 to 5 do
      begin
          x[y]:=y+x[y-1];
          writeln(x[y]);
      end;
end.
```

Ouput dari program disamping :

```
1     => x[1]
3     => x[2]
6     => x[3]
10    => x[4]
15    => x[5]
```

Indeks array **x** diisi oleh nilai variebel **y** dari statement **for**. Masalah proses, think by ur self...!

Perhatikan dan running contoh pemakaian skalar pada array berikut!

```
program larik1; uses wincrt;
const x:array[1..5] of byte = (7,32,9,2,56);
var  y:byte;
begin
     for  y:=1 to 5 do
     begin
          writeln('Konstanta larik ke ',y,' = ',x[y]);
     end;
end.
```

```
program larik2; uses wincrt;
type x=array[0..4] of string[10];
const bahasa:x=('Pascal','C++','Perl','Java','C');
var  i:byte;
begin
     for  i:=1 to 5 do
     begin
          writeln('Bahasa ke ',i,' = ',bahasa[i-1]);
     end;
end.
```

Setelah berbasa-basi dengan contoh, perhatikan soal Olimpiade Tk. Kabupaten ini :

```
program thn_2003; uses wincrt;
var angka:array[0..50]of longint;
    i:integer;
Begin
    angka[1]:=1;
    for i:=2 to 10 do
        angka[i]:=angka[i-1]*i;
    writeln(angka[5]);
End.
```

Bila dirunning :

```
120
```

Array disamping mempunyai indeks 51 buah. Dan indeks ke-1 bernilai 1. Selanjutnya dapat digambarkan:

```
angka[2]  :=angka[2-1]*2
          :=angka[1]*2
          :=1*2
          :=2
angka[3]  :=angka[3-1]*3
          :=angka[2]*3
          :=2*3
          :=6
```

```
angka[4]  :=angka[4-1]*4
          :=angka[3]*4
          :=6*4
          :=24
angka[5]  :=angka[5-1]*5
          :=angka[4]*5
          :=24*5
          :=120
```

OK, sekarang analisis soal olimpiade Tk. Kabupaten tahun 1997 ini :

```
program thn_1997; uses wincrt;
type ari=array[1..5]of byte ;
var  x:ari; i,j:byte;
begin
     for i:=1 to 5 do x[i]:=10 div i;
     j:=0;
     for i:=5 downto 1 do j:=j+x[i];
     writeln(j);
End.
```

B. ARRAY MULTIDIMENSI

Secara umum ada 3 bentuk array, yaitu sebagai berikut :

```
id : array[tipe_indeks1] of array[tipe_indeks2] of type;
id : array[tipe_indeks1,tipe_indeks2] of type;
```

Bila banyak elemen pada array satu dimensi tergantung dari banyaknya tipe indeks, maka pada array multi dimensi banyaknya elemen adalah perkalian antara tipe indeksnya.

```
Var matrik:array[1..10,1..10] of integer;    { 100 elemen }
Var matrik:array[1..3,1..3,1..3] of word;    { 27 elemen}
```

Contoh penggunaan array multi dimensi :

```
program multi; uses wincrt;
var  x:array[1..3,1..2] of byte;
     i,j:byte;
begin
     x[1,1]:=2;
     x[1,2]:=5;
     x[2,1]:=7;
     x[2,2]:=12;
     x[3,1]:=21;
     x[3,2]:=19;
     for i:=1 to 3 do
     begin
         for j:=1 to 2 do
             write(x[i,j],' ');
         writeln;
     end;
end.
```

Bila dirunning :

```
2 5
7 12
21 19
```

Array multidimensi ini banyak digunakan untuk operasi matriks. Karena, rata-rata operasi matriks mempunyai elemen banyak.

Perhatikan soal Olimpiade berikut ini :

```
program thn_2003_no44; uses wincrt;
var  a,b:array[1..3,1..3] of integer;
     x,y:integer;
begin
     a[1][1]:=0;  a[2][1]:=-1; a[3][1]:=8;
     a[1][2]:=17; a[2][2]:=-6; a[3][2]:=-17;
     a[1][3]:=4;  a[2][3]:=2;  a[3][3]:=-7;
     for x:=1 to 3 do
     begin
          for y:=1 to 3 do
          begin
               b[x][y]:=a[y][x];
               write(b[x][y],' ');
          end;
          writeln;
     end;
end.
```

Di dalam soal ini, terdapat looping yang statement-nya membalikkan nilai indeks array yaitu **b[x][y]:=a[y][x]**. Gambarannya seperti berikut :

```
Untuk x=1  =>  untuk y=1     => b[1][1]:=a[1][1]    =>   0
           =>  untuk y=2     => b[1][2]:=a[2][1]    =>  -1
           =>  untuk y=3     => b[1][3]:=a[3][1]    =>   8   (looping y habis)

Untuk x=2  =>  untuk y=1     => b[2][1]:=a[1][2]    =>  17
           =>  untuk y=2     => b[2][2]:=a[2][2]    =>  -6
           =>  untuk y=3     => b[2][3]:=a[3][2]    => -17   (looping y habis)

Untuk x=3  =>  untuk y=1     => b[3][1]:=a[1][3]    =>   4
           =>  untuk y=2     => b[3][2]:=a[2][3]    =>   2
           =>  untuk y=3     => b[3][3]:=a[3][3]    =>  -7   (looping y & x habis)
```

Bila dirunning, maka outputnya :

```
0 -1 8
17 -6 -17
4 2 -7
```

Untuk melatih kemampuan anda, sekarang buatlah sebuah program yang menginput masukan berupa matriks 2x3 dan menginversnya!!?!

Kemudian, selain untuk penyimpanan data yang banyak, array juga digunakan untuk pengurutan data. Pengurutan data biasa disebut sotrting. Ada dua jenis sorting, yaitu sort

ascending (kecil-besar) dan sort descending (besar-kecil). Perhatikan listing code sorting beikut :

```
program sort_ascending;
uses wincrt;
var  urut,temp:array[1..5] of integer;
     data,i,j:integer;
begin
     data:=5;
     for i:=1 to data do readln(urut[i]);
     {Proses sorting}
     for i:= 1 to data do
     begin
          for j:= i to data do
          begin
               if urut[j]<urut[i] then
               begin
                    temp:=urut[j];
                    urut[j]:=urut[i];   } {Perukaran variabel}
                    urut[i]:=temp;
               end;
          end;
          writeln(urut[i]); {Cetak data yang sudah terurut}
     end;
end.
```

Anggaplah banyaknya data 5 buah, dan ditempatkan pada array **urut**. Bila diproses maka sebagai berikut ini :

Input :	Output :
7	**2**
19	**7**
13	**13**
21	**19**
2	**21**

Contoh sorting diatas adalah ascending, lalu bagaimana untuk descending?? Gampangxs! Tugas anda hanyalah merubah tanda **<** (lebih kecil) menjadi tanda **>** (lebih besar) pada statement **if .. then**.

Lalu, bagaimana jika terdapat dua data yang berlainan variabel (array) yang akan disorting secara bersamaan..?? Caranya setelah looping ke-dua (**j**) anda masukkan array lain kedalamnya sama seperti variabel sebelumnya (**urut**). Masih bingung...?!???

\\ bye //*

SAMPEL PROGRAM 1

KONVERSI BILANGAN BINER KE DESIMAL

Listing program berikut akan memproses input yang berupa bilangan biner dan memprosesnya menjadi bilangan desimal. Bilangan biner yang digunakan adalah bilangan biner 16 bit positif atau setara dengan tipe word.

```
program biner_2_decimal;
uses wincrt;
type input=string[16];
procedure bin2dec(s:input;var k:word);
var  i,j,a:integer;
begin
     if s[length(s)]='1' then k:=1;
     for i:=length(s)-1 downto 1 do
         if s[i]='1' then
         begin
              a:=1;
              for j:=i to length(s)-1 do a:=a*2;
              k:=k+a;
         end;
end;
var  hasil:word;
     data:input;
begin
     readln(data);
     bin2dec(data,hasil);
     writeln(hasil);
end.
```

Bila anda memasukkan input 10111101011, maka output yang dihasilkan adalah 1515. Benarkah itu??! Coba hitung dengan kalkulator anda....!

Dalam listing program diatas tidak terdapat operasi **sqr** (perpangkatan), agar program terlihat lebih sederhana. Kemudian input dimasukkan kedalam variabel bertipe string, supaya ada pembatasan dalam karakter dan untuk mempermudah proses.

SAMPEL PROGRAM 2

KONVERSI BILANGAN DESIMAL KE BINER

Pada sampel program 1, program mengkonversi biner ke desimal. Lalu, bagaimana mengkonversi bilangan desimal ke biner??! Konversi bilangan desimal ke biner mudah sekali, tidak terlalu rumit seperti sampel program 1. Perhatikan :

```
program decimal_2_biner;
uses wincrt;
type teks=string[16];
procedure biner(y:longint;var t:teks);
var x:integer;
    s:teks;
begin
     x:=y mod 2;
     y:=y div 2;
     str(x,s);
     insert(s,t,1);
     if y>0 then biner(y,t);
end;
var  data:word;
     hasil:teks;
begin
     readln(data);
     biner(data,hasil);
     writeln(hasil);
end.
```

Bila anda memasukkan input 1515 maka output yang dihasilkan adalah 10111101011. Benarkah itu??! Coba hitung dengan kalkulator anda....! Lebih simple khan... J Hanya dengan rekursif prosedur dan sedikit prosedur string..!!

SAMPEL PROGRAM 3

KONVERSI BILANGAN DESIMAL KE HEKSA

Pada sampel program 1 dan 2, dijelaskan mengenai konversi bilangan biner. Nah.. lalu bagaimana dengan bilangan heksa..?! Program berikut akan mengkonversi bilangan desimal ke dalam heksa desimal, perhatikan :

```
program decimal_2_heksa;
uses wincrt;
type teks=string[16];
Procedure Hexa(i:longint;var t:teks);
Var  s:teks;
     j:longint;
Begin
     s:='0123456789ABCDEF';
     while i>0 do
     begin
           j:=i mod 16;
           i:=i div 16;
           insert(s[j+1],t,1);
     end;
end;
var  data:word;
     hasil:teks;
begin
     readln(data);
     hexa(data,hasil);
     writeln(hasil);
end.
```

Bila anda memasukkan input 42 maka output yang dihasilkan adalah 2A. Apa program tidak berbohong??! Coba hitung dengan kalkulator anda....! Lalu bagaimana dengan konversi heksa ke desimal??! Hal tersebut adalah sebagai latihan untuk anda..!!

Note :

Mohon maaf bila penulis tidak melampirkan soal olimpiade komputer Tk. Nasional maupun Internasional, dikarenakan tiga hal : Pertama, masih ada bahasan yang belum sempat dibahas terutama operasi file. Kedua, pemahaman analitiknya yang cukup sulit untuk pemula (beginner). Dan ketiga, banyak soal yang tidak bisa diselesaikan oleh compiler Turbo Pascal dan hanya bisa diselesaikan oleh compiler Free Pascal dikarenakan masalah **UPDATE**. Insya Allah pada Jilid Ke-2 akan penulis masukkan ke dalam lampirannya.

ABOUT AUTHOR

Fauzi Marjalih, S.Te, 2LeR, lahir di Cibarusah - Bekasi pada tanggal 28 September 1988. Pada waktu penulisan karyanya ini, ia masih berstatus pelajar SMA kelas 3 di SMAN 1 Cikarang Utara.

Seperlima dari kesehariannya ia habiskan di depan komputer (yang pasti bukan tidur lho...). Spiritnya yang tinggi mengajaknya untuk otodidak (manual) dari pengalaman yang ia hadapi tanpa guru dan dengan buku yang mengajarnya. Hasilnya, tidak begitu mengecewakan dan juga tidak membahagiakan. Ya... begitulah kalau berguru pada pengalaman, mungkin saja "modal buku satu biji menghasilkan buku sepuluh biji" bisa jadi kenyataan (note: tebal buku yang dihasilkan 1 rim).

Sekarang dia mengajar TI sekaligus menjadi teknisi di salah satu SMA swasta di Cikarang meski dia masih duduk di bangku SMA dan memulai untuk ngoprek open source dan pemograman jenis apa saja (yang pasti lebih menantang/sulit dari yang sebelumnya). Prinsipnya ialah "berguru pada pengalaman, mampu menelan kekalahan meski dengan rasa sungkan" dan cita-citanya adalah sukses dunia akhirat.

Bwt comment & suggest, send to : cool_sixiz@yahoo.com

DAFTAR CONTEKAN

Borland, *Turbo Pascal For Windows 1.5 User's Guide*, CA : Borland International, 1992.

Jogiyanto, Hartono, *Turbo Pascal Versi 5.0 Jilid 1*, Yogyakarta : Andi, 1989.

------, *Turbo Pascal Versi 5.0 Jilid 2*, Yogyakarta : Andi, 1989.

Lukito, Ediman, *Belajar Sendiri Pemograman Dengan Turbo Pascal 7.0,* Jakarta : Elex Media Komputindo, 1993.

www.geocities.yahoo.com/gb/

www.ilmukomputer.com

www2.toki.or.id