BAB II
TINJAUAN PUSTAKA

2.1 Konsep tipe data dan variabel

Dalam program terjadi pengolahan data menjadi informasi. Pengguna memberi masukan untuk kemudian diolah dan dimunculkan kembali kepada pengguna. Contohnya, program menghitung luas persegi panjang. Pengguna memasukkan nilai panjang dan lebar. Selanjutnya di-*Enter* keluarlah luas yang menjadi hasil perkalian panjang dan lebar. (Munir, 2007)

Program menggunakan memori komputer sebagai media penyimpanan data dan informasi. Data yang dimasukkan melalui *keyboard* akan dibaca oleh program lalu disimpan di memori (RAM). Hasil pengolahan juga dapat disimpan terlebih dulu di memori untuk kemudian ditampilkan di layar monitor. (Munir, 2007)

Dalam pengelolaan memori (sebagai media penyimpanan) digunakanlah konsep tipe data dan variabel. Memori dianalogikan sebagai rak lemari dengan slot-slot kecil penyusunnya yang berurutan dari bawah ke atas. Ada slot yang besar, ada pula yang kecil. Konsekuensinya, besar kecil isi yang bisa dimasukkan ke dalam slot tersebut tergantung pada besar kecilnya ukuran. Setiap slot diberi label yang berbeda antara satu sama lain. Tujuannya memudahkan pengaksesan dan tidak terjadi pengisian oleh data lain. (Munir, 2007)

Apabila rak adalah memori maka slot adalah bagian-bagian memori. Jenis tipe data mempengaruhi besar kecilnya ukuran slot. Contoh tipe data integer 2 byte, tipe data float 4 byte dan tipe data char memiliki besar 1 byte. Sedangkan label pada slot menandakan nama variabel dalam pemrograman. Sehingga dalam program diawali dengan nama-nama variabel beserta jenis tipe datanya. Biasanya dikenal dengan deklarasi variabel, yang diterjemahkan oleh *compiler* untuk menyiapkan memori sebelum digunakan oleh program. (Munir, 2007)

Variabel adalah suatu pengenal yang *programmer* definisikan untuk menyimpan nilai tertentu dalam program pada saat program sedang berjalan *(runtime)*. Nilai tersebut juga dapat diubah pada saat program sedang berjalan sesuai dengan kebutuhan.

Deklarasi variabel membutuhkan tipe data, fungsinya untuk memberitahu *compiler* bahwa variabel tersebut digunakan untuk menyimpan nilai dengan tipe data bersangkutan. (Munir, 2007)

Konsep variabel dalam pemrograman ibarat konsep variabel dalam matematika. Misalnya, 1 = x < 5 | x ? R artinya x adalah variabel bertipe real (domain x adalah bilangan real), dengan range (jangkauan) 1 sampai 5.

Contoh:

```
Program namaprogram;

var
nama_variabel : tipevariabel;
        nama_variabel2 : tipevariabel2;
begin
(*bagian program utama*)
end.
```

(Nugroho, 2004)

2.2 Tipe data dalam pascal

a. Tipe bilangan bulat

Tabel 2.1 Pembagian tipe bilangan bulat

Tipe data	Ukuran (dalam byte)	Rentang nilai

ShortInt	1	-128 sampai 127
Integer	2	-32.768 sampai 32.767
Longint	4	-2.147.483.648 sampai 2.147.483.647
Byte	1	0 sampai 255
Word	2	0 sampai 65.535

(Raharjo, 2008)

Apabila nilai yang dimasukkan melebihi rentang maksimum, maka nilainya akan dikembalikan ke keadaan minimum, kemudian ditambah dengan sisa yang ada. Sebagai contoh, variabel x yang bertipe shortint. Melalui suatu perhitungan di dalam program, variabel x ternyata harus menyimpan nilai 200, sedangkan rentang maksimum tipe shortint 127. Hal ini akan menyebabkan nilai dari variabel x menjadi -56, yang berasal dari perhitungan 200-256. Angka 256 merupakan besarnya rentang antara -128 sampai dengan 127. (Raharjo, 2008)

Integer memiliki representasi nilai dari serangkaian bilangan biner. Komputer memproses per blok bit yang umumnya terdiri dari 8 bit (1 byte). Byte yaitu integer 8 bit yang bisa menampung tipe bilangan asli, dan shortint yaitu tipe integer 8 bit yang menyimpan bilangan bulat.

Ukuran 8 bit artinya untuk menyimpan tipe tersebut diperlukan memori sebesar 8 bit, nilai yang bisa disimpan adalah 0000 0000 sampai dengan 1111 1111 atau 0 sampai dengan 255. Untuk bisa menyimpan nilai negatif dalam biner biasanya digunakan kebalikan dari representasi desimal ditambah dengan 1. Misalnya, menyimpan -5 desimal dengan merepresentasikan 5 yaitu 0000 0101, dibalik (*not*-kan) menjadi 1111 1010, dan ditambah 1 menjadi 11111011, sehingga -5 dalam biner adalah 11111011. Dengan cara ini maka nilai yang bisa ditampung dalam 8 bit adalah -128 sampai dengan 127. (Nugroho, 2004)

Operasi dasar untuk integer yaitu kali (disimbolkan dengan *), tambah (+), dan kurang (-). Misal, 5 + 5 = 10, 2 * 3 = 6, dan 3-2 = 1. Operasi pembagian (div) memberikan hasil pembagian yang dibulatkan, sehingga 4 div 2 = 2, dan 5 div 2 = 2. Operasi mod memberikan

sisa dari hasil bagi sehingga 5 mod 2 = 1, dan 2 mod 5 = 2 (2/5 jika dibulatkan adalah nol, sisanya adalah 2 – 0* 5 = 2).

(Nugroho, 2004)

b. Tipe data real

Tabel 2.2 Pembagian tipe data real

Tipe data	Ukuran (byte)	Digit penting	Rentang nilai
Real	6	11 sampai 12	$2.9x10^{-39}$ sampai $1.7x10^{38}$
Single	4	7 sampai 8	$1.5x10^{-45}$ sampai $3.4x10^{38}$
Double	8	15 sampai 16	$5.0x10^{-324}$ sampai $1.7x10^{308}$
Extended	10	19 sampai 20	$3.4x10^{-4932}$ sampai $1.1x10^{4932}$
Comp	8	19 sampai 20	$-9.2x10^{18}$ sampai $9.2x10^{18s}$

(Raharjo, 2008)

Tipe ini digunakan untuk merepresentasikan bilangan *floating point* atau bilangan yang mengandung angka dibelakang koma. Operasi terhadap real yaitu: tambah (+), kali (*), minus (-) (sama seperti integer), dan pembagian (memakai simbol “/” yang menghasilkan bilangan real).

(Nugroho, 2004)

c. Tipe string

Tipe string merupakan kumpulan dari karakter yang terangkai menjadi satu kata ataupun kalimat, misalnya 'Pemrograman Pascal', 'Informatika', 'Bandung'.

Contoh operasi penambahan (konketenasi) string:

```
program concat_string;
var s1, s2, s3 : string
begin
        s1:= 'hello';
```

```
    s2:='world';
    s3:= s1 + ' '+ s2;
writeln (s3);
end.
```

Nilai 'hello world' adalah gabungan string s1, spasi, dan string s2. Sebuah karakter juga bisa digabung dengan sebuah karakter:

```
var
    s:string;
    c:char
begin
    writeln ('Masukkan sebuah huruf:'); readln(c);
    s:= 'Huruf yang Anda masukkan adalah '+c;
    writeln(s);
end.
```

(Nugroho, 2004)

d. *Tipe data boolean*

Boolean adalah tipe data yang memiliki nilai *true* (benar) dan *false* (salah). Tipe ini diperlukan dalam kondisi perulangan dan kondisional (menggunakan if). Misalnya, 6 > 5 maka nilai ekspresi tersebut adalah *true* karena 6 lebih besar dari 5. Sedangkan, 6 < 2 maka nilai ekspresi *false.*

Dalam tipe boolean nilai 1 untuk *true* dan nilai 0 untuk *false*. Tipe boolean tidak bisa dioperasikan sebagai integer, ini berlaku hanya dalam bahasa Pascal. Namun, bilangan integer bisa dioperasikan dengan boolen dioperasikan melalui representasi bit pada bilangan integer, dan melakukan operasi boolean yang bersesuain terhadap bit (bit 1 untuk *true* , dan 0 untuk *false).* Dalam bahasa lain misalnya bahasa C, dua tipe ini bisa dipertukarkan. (Nugroho, 2004)

Contoh : 1 or 2 = 3

representasi biner untuk 1 desimal = 00000001

representasi biner untuk 2 desimal = 00000010

jika or-kan masing-masing bit hasilnya 00000011yang bernilai 3 desimal

Operator dalam boolean yaitu and, or, xor .

and menghasilkan *true,* jika keduanya bernilai *true*

or menghasilkan *true,* jika salah satunya bernilai *true*

xor menghasilkan nilai *true,* jika kedua nilai boolean berbeda

(Nugroho, 2004)

e. Tipe data karakter

Sebuah karakter direpresentasikan dengan bilangan integer 8 bit yang bernilai dari 0 sampai 255. Setiap nilai diberi simbol, misalnya nilai 65 dengan simbol A, 66 dengan simbol B, dan seterusnya.

Untuk yang bernilai 0-127 biasanya disebut dengan karakter ASCII (*American Standard Code For Information Interchange*). Dalam bahasa Pascal, terdapat fungsi ord untuk memetakan sebuah karakter menjadi nilainya, dan fungsi chr untuk melakukan proses kebalikannya.

Dalam tipe karakter tidak bisa dilakukan proses penjumlahan atau pengurangan. Tipe ini hanya bisa dioperasikan jika dikonversi menjadi integer dengan fungsi ord. (Nugroho, 2004)

2.3 Kompatibilitas tipe

Tipe data string tidak bisa dioperasikan dengan tipe integer. Pun halnya, dengan tipe data real tidak bisa langsung diisi dengan tipe data integer, karena mungkin dalam tipe data real mengandung pecahan.

Contoh penggunaan tipe data dengan banyak variabel:

```
a : integer ;
b : integer ;
d : integer ;
c : real ;
begin
        a :=1 ;
        b :=2 ;
        c :=a/b ;
        d :=a/b ;
```

end.

Operasi a/b menghasilan tipe data real, sehingga perintah terakhir d:=a/b tidak bisa di-*compile.* (Nugroho, 2004)

2.4 Konversi antar tipe data

Konversi antar beberapa tipe bisa dilakukan, contohnya konversi dari integer ke byte, real ke integer. Sebagian konversi melalui proses yang disebut *casting*. Contoh *casting* dengan shortint menjadi byte, atau byte menjadi integer, Namun, untuk tipe data real tidak bisa di-*cast* menjadi integer.
Konversi integer ke real bisa dilakukan. Sedangkan, dalam konversi real ke integer bisa menghasilkan *error*, karena ada beberapa yang dilakukan, apakah hasilnya akan dibulatkan ke atas atau ke bawah, atau bilangan di belakang koma semua dihilangkan.

Konversi antara yang *range*-nya lebih besar ke *range*-nya yang lebih kecil akan menghasilkan nilai yang tidak bisa diprediksi. Misalnya i adalah sebuah integer, dan b adalah sebuah byte, konversi dari i ke b (integer ke byte) akan menghasilkan hasil yang benar jika i memiliki nilai antara 0 sampai 255, jika i memiliki nilai di luar itu maka nilai b tidak bisa diprediksi.
(Nugroho, 2004)

Pemeranan tipe data (*typecasting*) adalah menganggap suatu variabel yang memiliki tipe data tertentu ke tipe data lainnya yang kompatibel. Dengan kata lain, *typecasting* dapat dianggap sebagia proses konversi tipe dat. Secara umum proses ini dibagi menjadi dua, yaitu:

a. Konversi implisit. Konversi implisit berarti proses konversi secara otomatis dilakukan kompiler. Sebagai contoh, pada saat menjumlahkan bilangan bulat dan bilangan rill, maka kompiler secara otomatis akan melakukan konversi terhadap bilangan bulat tersebut menjadi bilangan riil. Perhatikan contoh kode berikut ini.

```
var
        a: integer;
        b, c : real;
begin
        a := 2;
```

```
        b := 3.56;
        c :=  a + b ; { a secara otomatis akan dikonversi ke tipe real}

        …..
end.
```

Pada kode diatas, bilangan 2 akan dikonversi menjadi 2.00. (Nugroho, 2004)

b. Konversi eksplisit. Berbeda dengan konversi implisit, disini konversi dilakukan sendiri secara eksplisit. Sebagai contoh, variabel a yang bertipe char dan ingin dianggap variabel tersebut sebagai tipe integer, maka dapat dilakukan *typecasting* dengan cara berikut:

integer (a);

Untuk lebih jelasnya, perhatikan contoh kode di bawah ini:

```
var
        a : char;
        i : integer;
begin
        a :=  'A';          {Mengisikan variabel a dengan karakter 'A'}
        i  :=  integer  (a);  {  Menganggap  a  sebagai  variabel  integer  dan  menyimpan
                                -nya ke dalam varaibel I}

        ….
end.
```

Dari kode tersebut, varaiabel i akan bernilai 65, yang merupakan kode ASCI dari karakter 'A'. Sebaliknya, apabila ingin melakukan konversi variabel I ke tipe char, maka akan menuliskannya seperti berikut:

char (i);

Sebagai catatan, tidak semua tipe data dapat dilakukan konversi ke tipe data lain. Sebagai contoh, tidak dapat dilakukan konversi data dari real ke tipe char, begitu juga sebaliknya. Bahasa Pascal memiliki dua buah prosedur yang sering sekali digunakan untuk melakukan konversi tipe data antara numerik dengan string, yaitu prosedur Val dan Str.

(Raharjo, 2008)

Prosedur Val

Prosedur ini digunakan untuk melakukan konversi dari tipe numerik ke tipe string. Berikut ini prototipe dari prosedur ini:

procedure val (S; var V ; var Code ; integer);

di mana S adalah string yang akan dikonversi ke tipe numerik. Hasil konversi akan disimpan ke dalam variabel V. maka variabel Code akan bernilai 0, jika tidak maka varaiabel Code bernilai selain 0. Perhatikan contohnya di bawah ini:

```
function StrToNumber ( const s : string) : real;
var
        hasil ; real;
        kode: integer;
begin
        val(s, hasil, kode);
        if (kode = 0) then begin
                StrToNumber := hasil;
        end;
end.
```

Fungsi StrToNumber dapat digunakan untuk melakukan konversi dari tipe string ke numerik. Berikut ini contoh pemanggilan fungsi tersebut.

```
var
        n : real;
begin
        n := StrToNumber ( '3.14');
        ….
end.
```

(Raharjo, 2008)

Sampai di sini, variabel n akan bernilai 3.14 (bukan string '3.14') sehingga varaiabel n tersebut dapat digunakan digunakan dalam operasi numerik.

Prosedur Str

Prosedur ini digunakan untuk melakukan konversi dari tipe numerik ke tipe string. Adapun prototipe dari prosedur ini sebagai berikut:

procedure Str (X; var S);

dimana X merupakan tipe numerik yang akan dikonversi. Hasil konversi akan disimpan ke dalam varaiabel S. Berikut contohnya:

{fungsi untuk melakukan konversi dari bilangan bulat ke string}

```
function IntToStr (x: longinr) : string;
var
        hasil : string;
begin
        str (x, hasil);
        IntToStr := hasil;
```

```
end;

{fungsi untuk melakukan konversi dari bilangan riil ke string}
function FloatToStr ( x: real) : string;
var
        hasil : string;
begin
        str (x, hasil);
        IntToStr := hasil;
end;
```
(Raharjo, 2008)

Adapun contoh penggunaan dua buah fungsi tersebut sebagai berikut:

```
var
        sInt, sFloat : string;
begin
        sInt := IntToStr (100); { variabel sInt akan berisi string '100'}
        sFloat := FloatToStr (2.12); { varaibel sFloat akan berisi string '2.12'}

        ….
end.
```
(Raharjo, 2008)

2.5 *Runtime* Program

Dalam Pascal ada perintah *unit system* dan *unit* DOS, yang berada dalam file TURBO.TPL. *Unit system* merupakan sebuah *runtime* Turbo Pascal yang mendukung semua proses yang dibutuhkan pada waktu *runtime* (eksekusi program). Unit ini secara otomatis ditambahkan dalam bahasa Pascal, walaupun tidak tercantum saat meng-*compile*. (Raharjo, 2008)

2.6 Turbo Profiler

Sofware yang mendukung dalam kemampuan analisis program dan menyediakan laporan statistika yang mendetail adalah Borland Turbo Profiler. Salah satu kemampuan turbo profiler adalah analisis waktu eksekusi program (*runtime*). (Borland, 1990)

Borland Turbo Profiler adalah perangkat lunak dalam siklus pengembangan. Setelah melakukan *coding* program yang diinginkan, Turbo profiler akan membantu program agar lebih cepat dan lebih efisien . Profiler adalah perangkat lunak untuk mengukur kinerja program dengan mencari *bottleneck*–nya.

Borland Turbo Profiler memungkinkan untuk menyempurnakan program, membantu dalam mempercepat program. Profiler dijalankan di DOS. (Borland, 1990)

Profiller adalah salah satu *software* yang paling berguna dan penting setidaknya dipahami untuk pengembangan perangkat lunak yang baik. *Survey* menunjukkan bahwa hanya sebagian kecil dari *programmer* professional yang benar-benar menggunakan profiler untuk meningkatkan performa programnya. Studi lain menunjukkan bahwa sebagian besar waktu *programmer* dipakai untuk menebak permasalahan dalam programnya. (Borland, 1990)

Apa keuntungan untuk menggunakan *software* ini? Pertama, profiler dapat meningkatkan kinerja program secara keseluruhan. Kedua, profiler memiliki kemampuan menghasilkan kode program yang efisien. Intinya adalah profiler seperti *debugging*, dalam siklus pengembangan program. Profiler juga digunakan untuk mengetahui waktu eksekusi (*runtime*) program yang berkaitan dengan alokasi memori. Ketika pemakaian memori cukup besar ini berakibat pada *runtime* program yang lama. (Borland, 1990)

Sebelum memulai profiler, tentukan bagian program yang ingin diprofiller. Suatu bagian dalam program di mana ingin dilakukan pengujian dan pengunpulan data statistik, sehingga akan diperoleh informasi untuk analisis dari program tersebut. Untuk menganalisis sejumlah *runtime* dalam program harus diketahui berapa banyak waktu yang dibutuhkan setiap program ketika *running*
(Borland, 1990)

2.7 Analisis Ragam *One-way* Anova

Analisis Ragam atau analisis varians adalah sebuah metode yang sering dipakai untuk analisis data. Dalam pengertian uji hipotesis, analisis ragam digunakan untuk menilai kesamaan nilai tengah beberapa populasi yaitu memeriksa apakah $\mu_1 = \mu_2 \ldots = \mu_p$ dengan tandingannya berupa pernyataan bahwa paling sedikit ada sepasang populasi yang berbeda. Untuk pembandingan hasilnya setara dengan hasil pengujian uji-t. (Walpole, 1995)

Sebaran F dapat digunakan untuk memeriksa hipotesis tentang kesamaan ragam populasi atau lebih popular disebut dengan F_{hitung} yang dapat dibandingka dengan tabel F (F_{tabel}). Untuk

taraf $\alpha = 0.05$ yang cukup kecil berarti kecil peluang kedua populasi tersebut memiliki ragam yang sama.

Nilai F_{hitung} yang lebih besar dari F_{tabel} pada $\alpha = 0.05$ mengandung arti bahwa palings sedikit ada sepasang perlakuan yang berbeda dengan resiko jenis I sebesar 0.05.

Hakekatnya meskipun pendekatannya masalah ragam, tetapi yang diperiksa adalah nilai tengahnya, sedangkan tentang keragaman telah disederhanakan dengan asumsi kehomogenan ragam. Prosedur ini jelas kurang layak bila ragam tiap perlakuam tak sama. (Walpole, 1995)

Uji-t dapat dipakai untuk pembandingan 2 perlakuan, namun untuk pembandingan lebih dari 2 perlakuan sekaligus, uji F lebih hemat dan resiko α-nya mencakup pembandingan seluruh perlakuan.

Jika $F_{hitung} < F_{0.05}$ maka data tidak mendukung untuk menolak H_N bahwa semua perlakuan memiliki pengaruh yang sama yang bisa berarti tidak hanya nilai tengahnya tetapi juga ragamnya *homogen*. Akan tetapi bila ternyata diperoleh $F_{hitung} > F_{0.05}$, masalahnya adalah perbedaan ini mungkin tidak hanya menyangkut nilai tengahnya saja, tetapi juga ragamnya mungkin saling berbeda (dalam uji F, kehomogenan ragam hanya didasarkan pada asumsi). (Walpole, 1995)

Penolakan $H_N : \mu = \mu = \ldots = \mu_t$ akibat uji F memberikan dorongan untuk memeriksa lebih lanjut, perlakuan mana saja yang sebenarnya berbeda, oleh karena itu dapat dimengerti bila kemudian banyak orang mengembangkan metode pembandingan ganda dengan berbagai kelemahan atau kelebihannya.

Uji Bartlett sering digunakan untuk memeriksa kehomogenen ragam dalam bentuk pengujian hipotesis dengan H_N bahwa ragam antara perlakuan bersifat *homogen* . Sayangnya uji ini peka terhadap ketidak-normalan data, penolakan H_N mungkin hanya disebabkan oleh data yang tidak normal. Di samping itu tidak menolak H_N juga tidak membuktikan bahwa ragam antara perlakuan *homogen*, karena hal ini bisa disebabkan oleh kurangnya data atau datanya tidak normal.

Penggunaan uji F dalam analisis ragam cukup resisten terhadap ragam yang heterogenan atau ketak-normalan data terutama bila memiliki ulangan sama. Oleh karena itu uji Bartlett tidak dianjurkan sebelum melakukan proses analisis ragam.
(Aunuddin, 2005)

Analisis ragam adalah suatu metode untuk menguraikan keragaman total data menjadi komponen – komponen yang mengukur berbagai sumber keberagaman.

Klasifikasi pengamatan berdasarkan satu kriteria disebut klasifikasi satu- arah. Misalkan ada k populasi. Dari masing- masing populasi itu diambil contoh berukuran n. Misalkan pula bahwa k populasi itu bebas dan menyebar normal dengan nilai tengah μ_1, μ_2, …., μ_k dan ragam sama.
Pengujian hipotesis:
$H_0 : \mu_1 = \mu_2 = \ldots = \mu_k$,
H_1 : sekurang-kurangnya dua nilai tengah tidak sama.
(Walpole, 1995)

One-way ANOVA digunakan untuk membandingkan apakah terdapat perbedaan atau kesamaan rata-rata antara tiga atau lebih kelompok data untuk suatu kategori tertentu. Asumsi yang digunakan adalah variabel data berdistribusi normal dan homogenitas varians antara kelompok data.

Dalam ANOVA terdapat ketentuan:

a. Jika signifikansi < 0.005 maka varian kelompok data tidak sama, jika signifikan si> 0.005 maka varian kelompok data sama.
b. Langkah-langkah uji ANOVA sebagai berikut:
    1. Merumuskan hipotesis
    2. Menentukan F hitung dan signifikansi
    3. Menetukan F Tabel

       F Tabel dicari pada signifikansi 0.05, df1 (jumlah kelompok data-1) dan df2 (n-3).

    4. Kriteria pengujian :

a. Jika F hitung ≤ F tabel, maka H_0 tidak ditolak.

b. Jika F hitung > F tabel, maka H_0 ditolak.

5. Berdasarkan signifikansi:

   a. Jika signifikansi > 0.05, maka H_0 tidak ditolak.

   b. Jika signifikansi < 0.05, maka H_0 ditolak.

6. Membuat kesimpulan. (Yamin, 2009)