Pertemuan 3

Waktu : 135 menit

Tujuan Pembelajaran : Mahasiswa mampu menjelaskan teknik pemrograman menggunakan Modul.

Substansi Materi : Modul

Tabulasi Kegiatan Perkuliahan

No	Tahap Kegiatan	Kegiatan Pengajar	Kegiatan Mahasiswa	Media & Alat	Waktu
1	Pendahuluan	1. Membuka pertemuan 2. Mengulang materi pertemuan sebelumnya	Menyimak Bertanya	Papan Tulis	20 Menit
2	Penyajian Materi	1. Variabel dan Konstanta Lokal 2. Parameter formal dan aktual 3. Transfer Parameter 4. Nested Procedure 5. Rekursif 6. Contoh program 7. Contoh soal Modul	Menyimak Bertanya Menjawab Pertanyaan	Papan Tulis	80 Menit
3	Penutup	1. Menyimpulkan materi pertemuan 2. Memberikan tugas kecil 3. Menutup pertemuan	Menyimak	Papan tulis	35 Menit

MATERI KULIAH

VARIABEL DAN KONSTANTA LOKAL

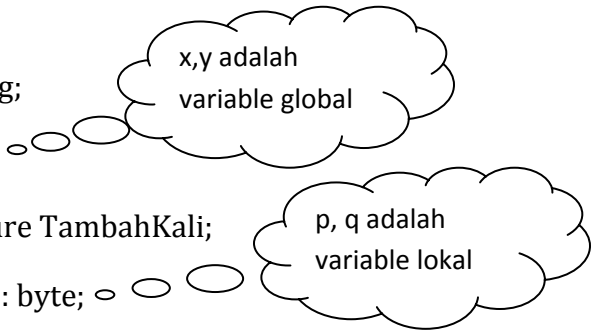
Pada kedua contoh pada materi sebelumnya, variable p, q, x dan y semuanya bersifat global sehingga dapat dikenal baik pada program utama dan pada procedure TambahKali sedangkan pada dua listing program terakhir, variable x dan y juga bersifat global namun variable p dan q tidak diperlukan dali karena digantikan oleh function itu sendiri yang dapat menampung nilai.

Ada baiknya membatasi suatu variable atau konstanta hanya pada batas local saja karena beberapa factor seperti kemudahan dalam mengecek jika terjadi kesalahan, memudahkan

pengontrolan, dapat memakai variable dengan nama yang sama tetapi dengan fungsi atau procedure yang berbeda.

Jika mendeklarasikan variable atau konstanta local di dalam suatu procedure atau function, maka variable atau konstanta tersebut hanya dapat digunakan pada procedure atau function yang bersangkutan dan tidak dapat digunakan pada procedure atau function lainnya ataupun pada program utama.

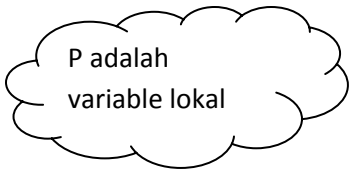
Contoh 1 :




```
Program Hitung;
Var x, y : byte;
    Procedure TambahKali;
    Var p, q : byte;
    Begin
    P := x + y;
    Q := x * y;
    Writeln(' X + Y = ', p);
    Writeln(' X * Y = ', q);
    End;
Begin
    Write(' X = '); ReadLn(x);
    Write(' Y = '); ReadLn(y);
    TambahKali; {Untuk menjalankan procedure TambahKali}
End.
```

Contoh 2 :

```
Program Hitung;
Var x, y : byte;
    Function Tambah:byte;
    Var p : byte;
    Begin
    P := x + y;
    Tambah := p;
    End;
    Function Kali:byte;
    Var q : byte;
    Begin
    q := x * y;
    Kali := q;
    End;
Begin
    Write(' X = '); ReadLn(x);
    Write(' Y = '); ReadLn(y);
    WriteLn( ' X + Y = ', Tambah);
    WriteLn(' X * Y = ', Kali);
End.
```





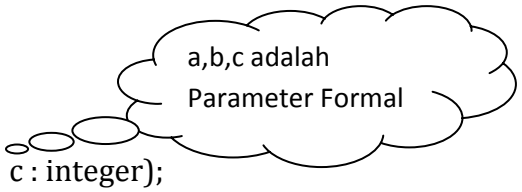
Pada kedua contoh diatas, variable x dan y bersifat variable global sehingga dapat dikenal baik pada program utama maupun pada procedur TambahKali atau function Tambah dan Kali. Sedangkan variable p dan q bersifat variable local, sehingga hanya dapat dikenal pada

procedure TambahKali saja dan function Tambah serta Kali. Apabila variable local tersebut dipaksakan berada pada program utama, maka jika decompile akan diberikan pesan error "unknown identifier" yang menandakan bahwa variable P dan Q tersebut tidak dikenal pada program utama.

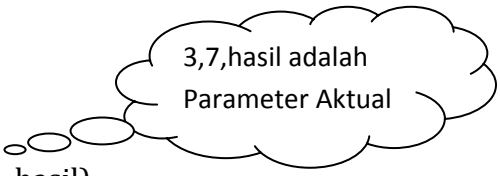
Parameter Formal dan Aktual

Jika sebuah modul, baik procedure ataupun function memiliki parameter yang berada di dalam kurung ( ), maka parameter tersebut disebut dengan parameter formal, sedangkan parameter yang terdapat pada baris perintah pemanggil modul tersebut disebut sebagai parameter actual.

Untuk lebih jelasnya lihat penjelasan dibawah ini :




```
Program HitungLuas;
    Procedure Hitung(a, b : byte; c : integer);
    Begin
    ……..
    End;
{Program Utama}
Var hasil : byte;
Begin
    Hitung (3, 7, hasil);
    ……..
End.
```



Pada contoh diatas procedure Hitung memiliki parameter a, b dan c yang disebut sebagai parameter formal. Procedure hitung tersebut dipanggil pada program utama dengan parameter 3, 7 dan hasil. Parameter 3, 7 dan hasil dikenal sebagai parameter actual. Kemudian parameter actual 3, 7 dan hasil tersebut masing-masing ditransfer kepada parameter formal a, b dan c sesuai dengan urutannya. Sehingga pada procedure hitung a bernilai 3, b bernilai 7, dan c berisi variable penampung hasil.

Transfer parameter akan dijelaskan lebih lanjut pada bagian berikutnya.

```
Program Gambar;
    Function CheckPosisi(x, y : byte ; Data : string) : Boolean;
    Begin
    ……..
    End;
{Program Utama}
Begin
    If CheckPosisi(3, 5,'*') then
    Write('Posisi Telah Benar');
    ……
End.
```

Parameter Formal

Parameter Aktual

Pengertian parameter formal dan actual pada function tidak berbeda dengan yang dijelaskan sebelumnya pada procedure. Function CheckPosisi mempunyai parameter X, Y dan Data. Ketiga parameter tersebut dikenal sebagai parameter formal. Function CheckPosisi dipanggil dalam program utama dengan parameter 3, 5, dan '*'. Parameter 3, 5, dan '*' tersebut dikenal sebagai parameter actual. Pada saat function CheckPosisi dijalankan, parameter X akan berisi nilai 3, Y berisi nilai 5, dan Data berisi karakter *. Setelah function selesai dijalankan, hasil akhir dari proses function CheckPosisi akan

disimpan pada function itu sendiri, yaitu pada "CheckPosisi" yang bertipe Boolean. Karena CheckPosisi bertipe Boolean, maka sudah dapat dipastikan bahwa nilai yang ditampungnya adalah "True" atau "False". Lalu nilai true atau false itu akan dikirimkan kembali pada baris program pemanggilnya, yaitu :

```
If CheckPosisi(3, 5,'*') then
Write('Posisi Anda Benar');
```

Jika CheckPosisi bernilai true maka pada layar akan dicetak "Posisi Anda Benar".

Transfer Parameter

Pada saat memanggil suatu modul (procedure /function) dengan parameter sebenarnya telah terjadi pengiriman parameter dari parameter actual (yang terdapat memangil procedure /function) ke parameter formal (yang terdapat pada procedure atau function tersebut) pengiriman parameter tersebut dapat dilakukan dengan dua cara , yaitu:

- Transfer parameter by value (secara nilai)
- Transfer parameter by location /reference (secara acuan)

Transfer Parameter by Value

Nilai yang tersimpan dalam parameter dikirim ke modul (procedure/ function) untuk diolah tetapi tidak minta hasil olahan tersebut untuk dikembalikan dalam mode ini terjadi aliran 1 arah yaitu dari pemangil procedure ke procedure itu.

Contoh 1:

```
Program Hitung;
Var x, y : byte;
    Procedure TambahKali(p, q : byte);
    Begin
    P := x + y;
```

```
        Q := x * y;
        Writeln(' X + Y = ', p);
        Writeln(' X * Y = ', q);
        End;
Begin
        Write(' X = '); ReadLn(x);
        Write(' Y = '); ReadLn(y);
        TambahKali(x,y); {Untuk menjalankan procedure TambahKali}
End.
```

Contoh 2 :

```
Program Hitung;
Var x, y : byte;
        Function Tambah(p, q : byte) : byte;
        Begin
        Tambah := x + y ;
        End;
        Function Kali(p, q :byte): byte;
        Begin
        Kali := x * y;
        End;
Begin
        Write(' X = '); ReadLn(x);
        Write(' Y = '); ReadLn(y);
```

```
        WriteLn( ' X + Y = ', Tambah(x, y));
        WriteLn(' X * Y = ', Kali(x, y));
End.
```

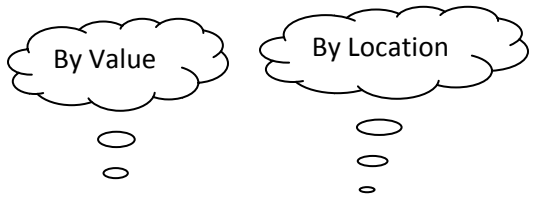
Transfer parameter by location

Transfer parameter by location sering dikenal juga dengan transfer parameter by reference (secara acuan) dalam kasus ini yang ditransfer hanya lokasinya saja (dapat berisi data atau kosong) untuk di olah , dan meminta hasil olahan tersebut untuk dikembalikan dan disimpan pada lokasi yang telah ditransfer tersebut dalam mode ini dapat terjadi aliran 2 arah dari pemanggil procedure ke procedure itu dan sebaliknya .

Contoh 1:

```
Program Hitung;
Var x, y : byte;
        Procedure TambahKali(var p, q : byte);
        Begin
        P := x + y;
        Q := x * y;
        End;
Begin
        Write(' X = '); ReadLn(x);
        Write(' Y = '); ReadLn(y);
        TambahKali(x,y);
        WriteLn('X + Y = ', x);
End.
```

Contoh 2 :

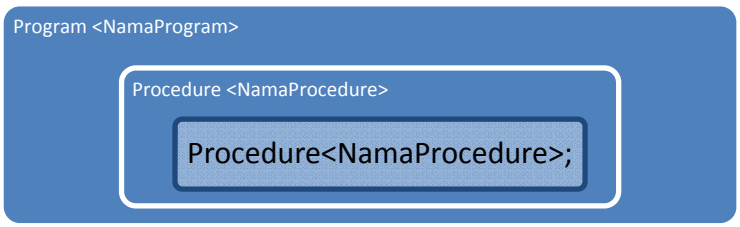



```
Program Hitung;
Var x, y, z1, z2 : byte;
    Procedure TambahKali(p, q : byte; var z1, z2 : byte) ;
    Begin
    Z1 := x + y;
    Z2 := x * y;
    End;
Begin
    Write(' X = '); ReadLn(x);
    Write(' Y = '); ReadLn(y);
    TambahKali(x, y, Z1, Z2);
    WriteLn( ' X + Y = ', Z1);
    WriteLn(' X * Y = ', Z2);
End.
```

Nested procedure (ProcedureTersarang)

Nested procedure adalah procedure yang terdapat di dalam procedure yang lainya .

Bentuk Umum :



Gambar 1. Nested Procedure

Procedure /function rekursif

Procedure/function rekursif adalah procedure /function yang dapat memanggil dirinya sendiri procedure/function rekursif ini dipakai karena memiliki kelebihan yaitu penulisan baris program dapat menjadi lebih singkat , tetapi juga memiliki kekurangan yaitu membutuhkan banyak memori karena setiap kali program bagian dipanggil oleh dirinya sendiri dibutuhkan sejumlah ruang memori tambahan.

Contoh :

```
Var   n:byte;
Procedure rekursif;
Begin
    Writeln(n);
    N:=n-1;
    If  n>0 then rekursif;
End;
{program utama}
Begin
  N:=10;
  Rekursif;
End.
```