Bab 2

Konsep Dasar Program

2.1 Objek

Suatu kelas *(class)* adalah suatu tipe data encapsulates dan operasi pada data di dalam unit tunggal. Sebelum pemrograman berorientasi objek *(object-oriented programming),* data dan operasi (fungsi) telah diperlakukan sebagai unsur-unsur terpisah. Suatu obyek adalah suatu kejadian dari suatu kelas. yaitu merupakan suatu nilai jenis kelas. Istilah obyek sering digunakan dengan bebas di mana perbedaan antara suatu kelas dan kejadian dari kelas tidaklah penting, istilah "objek" bisa juga mengacu pada suatu kelas.

Anda dapat mulai memahami objek jika anda memahami rekord Pascal atau struktur di dalam C. Rekord dibuat dari field yang berisi data, di mana masing-masing field mempunyai tipe sendiri. Record membuatnya mudah untuk mengacu pada suatu kumpulan dari variasi elemen data.

Objek adalah juga kumpulan elemen data. Tetapi objek – tidak seperti record--berisi prosedur dan fungsi yang beroperasi pada datanya. Fungsi dan Prosedur ini disebut *methods*.

Suatu objek elemen data diakses melalui propertis. Propertis dari kebanyakan objek Delphi mempunyai nilai-nilai yang dapat anda rubah pada waktu disain tanpa menulis kode. Jika anda ingin suatu nilai properti untuk merubah pada runtime, anda hanya menulis sedikit kode.

2.2 Komentar Program

Komentar dipakai untuk memberikan penjelasan atau keterangan di dalam baris program. Teks yang ditulis sebagai komentar tidak akan dikomilasi oleh kompiler pada saat program aplikasi dijalankan.

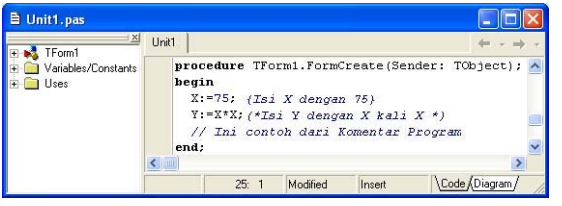
Untuk menuliskan sebuah komentar, Anda dapat menggunakan salah satu tanda dari tiga bentuk tanda yang disediakan, yaitu:

- Kurung kurawal : { Komentar program]
- Kurang bintang : (* Komentar program*)
- Slash ganda : // Komentar program

Komentar dengan tanda kurang kurawal dan kurung bintang menggunakan tanda pembuka dan tanda penutup, sehingga teks yang dianggap sebagai komentar adalah teks yang terletak di antara tanda pembuka dan tanda penutup.

Sedangkan tanda komentar slash ganda tidak menggunakan tanda penutup, sehingga semua teks yang terletak di belakang tanda slash ganda dianggap sebagai komentar.

Contoh:



Gambar 2.1 *Komentar pada program*

2.3 Subrutin

Subrutin terdiri dari dua, yaitu **procedure** dan **fungsi**. Kedua subrutin ini berisi beberapa pernyataan yang berfungsi untuk melakukan tugas tertentu. Perbedaan dari kedua subrutin tersebut adalah bahwa function selalu mengembalikan suatu nilai setelah dipanggil, sedangkan procedure tidak.

2.3.1 Procedure

Prosedur adalah bagian program yang melaksanakan program tertentu pada saat dipanggil dan kemudian kembali ke bagian pemanggilannya. Berikut ini adalah bentuk penulisan sebuah prosedur secara umum :

```
Procedure nama_proc (Parameter1,Parameter2,…);
Begin
    <pernyataan>
end;
```

Nama_proc merupakan sebuah nama yang diberikan untuk prosedur. Aturan pemberian prosedur sama dengan aturan penamaan variabel. Parameter1, Parameter2,……. Merupakan informasi yang diberikan ke prosedur. Dalam memberikan parameter, anda juga harus menentukan tipe parameternya.

```
Procedure Latih (A : String);
Begin
    <pernyataan>
end;
```

Procedure Latih di atas menggunakan parameter bertipe string, yaitu A

```
Procedure Latih (A : String;  B : Integer);
Begin
    <pernyataan>
end;
```

Procedure Latihan di atas memiliki beberapa parameter, dan penulisan dari kedua parameter tersebut dipisahkan dengan tanda titik koma. Pemanggilan sebuah prosedur dilakukan dengan

menyebutkan namanya. Bila terdapat parameter, letakkan parameter di dalam tanda kurang.

```
Latih ('Madiun')
```

Perintah di atas memanggil procedure Latih dengan parameter 'Madiun'. Sedangkan perintah di bawah ini memanggil beberapa parameter yang masing-masing penulisannya harus dipisahkan dengan tanda koma.

```
Latih ('Madiun',2000)
```

Contoh penulisan program di bawah ini melakukan suatu tugas yang berulang kali.

```
Begin
  Edit1.Text :=DateToStr(Waktu.Date);
  Edit2.Text :=DateToStr(Waktu.EndDate);

  Edit1.Text :=DateToStr(Waktu.Date);
  Edit2.Text :=DateToStr(Waktu.EndDate);
End.
```

Program di atas mengulangi tugas yang sama sebanyak dua kali, dan apabila pengulangan tersebut harus atau sering dilakukan maka hal ini dapat membuat program kita tampak lebih panjang dan lebar sehingga tampak kurang efektif dan efisien. Anda dapat meringkas penulisan program di atas dengan membuat sebuah procedure dan memanggilnya, seperti yang tampak pada contoh program berikut:

```
Procedure Tanggal;
Begin
  Edit1.Text :=DateToStr(Waktu.Date);
  Edit2.Text :=DateToStr(Waktu.EndDate);

  Edit1.Text :=DateToStr(Waktu.Date);
  Edit2.Text :=DateToStr(Waktu.EndDate);
End;

Begin
   Tanggal;
   Tanggal;
End;
```

2.3.2 Function

Function adalah bagian program yang melaksanakan program tertentu pada saat dipanggil dan kembali ke bagian pemanggilan dengan menghasilkan sebuah nilai.

Berikut ini adalah bentuk penulisan sebuah function secara umum :

```
Function nama_func (Parameter1,Parameter2,….):Tipe;
Begin
    <Pernyataan>
end;
```

Nama_func merupakan sebuah nama yang diberikan untuk function. Aturan pemberian nama function sama dengan aturan penamaan variabel.

Parameter1, Paramete2, …. Merupakan informasi yang dibeirkan ke function. Dalam memberikan parameter, Anda juga harus menentukan tipe parameternya.

Tipe adalah tipe dari nilai yang dikembalikan oleh fungsi. Anda dapat mengisi tipe ini dengna integer, string, real dan lain-lain.

```
Function Latih (A : Integer) : Real ;
Begin
   <Pernyataan>
end;
```

Function Latih di atas menggunakan parameter bertipe Integer, yaitu A.

```
Function Latih (A : String; B : Integer);
Begin
   <Pernyataan>
end;
```

Function Latihan di atas memiliki beberapa parameter, dan penulisan dari kedua parameter tersebut dipisahkan dengan tanda titik koma.

Untuk mengambil nilai dari sebuah function, anda harus menyediakan variabel. Contoh.

```
X  :=Latih (‘Madiun’)
```

Perintah di atas memanggil function Latih dengan nilai 'Madiun' yang diterima variabel X. Sedangkan perintah di bawah ini memanggil beberapa parameter yang masing-masing penulisannya harus dipisahkan dengan tanda koma.

```
X  :=Latih ('Madiun'2000)
```

Untuk memudahkan pemahaman tentang penggunaan perintah function, di bawah ini terdapat penulisan program untuk membuat function potongan yang didapat dari perhitungan 15% dari hasil perkalian antara Jumlah dan Harga.

```
Var
  Harga, jumlah : Integer
  Diskon : real ;

Function potongan (A,B; Integer ) : real;
Var Bayar : Real
Begin
  Bayar := (A*B)* 15 / 100 :
  Potongan := Bayar ;
End
Begin
  Jumlah := Str Tolnt (Edit1 : Text );
  Harga  := Str Tolnt ( edit2 : text )
  Diskon := Potongan (Jumlah ; Harga );
End
```

Perhatikan baris perintah function berikut :

```
Function Nil_Tot (A,B : Integer) : Real
```

Function mempunyai dua parameter nilai, yaitu A dan B yang bertipe integer. Hasil dari function Potongan bertipe Real. Dalam penulisan sebuah function harus ada pernyataan yang digunakan untuk menyimpan nilai function. Adapun pernyatan pada contoh tersebut di atas adalah :

```
Potongan : = Bayar
```

Sedangkan perintah yang dapat di pakai untuk memanggil function dalam program tersebut adalah

```
Diskon := Potongan (Jumlah, Harga) ;
```

Karena hasil dari Function Potongan bertipe Real maka variabel Diskon juga harus memiliki tipe yang sama, yaitu real.

Pada contoh di atas, Anda harus dapat membedakan variabel mana yang merupakan variabel global dan variabel lokal. Variabel global merupakan variabel yang berlaku untuk seluruh isi program, baik untuk proram utama maupun untuk procedure, seperti variabel Harga, Jumlah dan Diskon. Sedangkan variabel lokal hanya dipakai di dalam procedure sehingga hanya procedure tersebut yang akan mengenalnya, seperti variabel Bayar.

Berikut ini adalah tabel yang mengandung beberapa prosedur ataupun fungsi yang telah disediakan Delphi.

Tabel 2.1 *Prosedur dan Fungsi*

Nama	Kegunaan
AnsiUpperCase	Mengubah karakter menjadi huruf besar (kapital)
AppendStr	Menampilkan string ke string yang telah ada sebelumnya
Clrscr	Membersihkan layer
Copy	Menyalin sebuah string
DateToStr	Mengubah suatu nilai dari format date ke format string
Free	Merusak sebuah objek dengan cepat
IntToHex	Mengubah sebuah data integer ke data heksadesimal
MkDir	Membuat subdirektori
Reset	Membuka file yang telah ada
StrMove	Menyalin karakter-karater dari sebuah string ke string yan lain

2.4 UNIT

Sebuah program dibangun berdasarkan modul kode-kode program yang disebut dengan unit. Setiap kita membuat sebuah

form, unit akan dibuat dengan sendirinya. Unit tersebut berfungsi untuk mengatur serta mengendalikan segala sesuatu yang berhubungan dengan form. `

Unit memungkinkan Anda membagi program yang besar menjadi modul-modul yang dapat disunting secara terpisah. Unit jenis ini dapat berisi kumpulan function atau procedure yang telah dikompilasi, yang juga dapat dipakai program aplikasi lain. Jadi apabila Anda telah membangun sebuah procedure yang dimasukkan dalam suatu unit, maka anda bisa memanggil dan memasukkan ke dalam program lain yang memerlukan tanpa harus membuatnya lagi. Adapun bentuk penulisan umum dari sebuah unit :

```
Unit Nama_Unit;

Interface
  Uses
  ----
  Const
  ----
     Type
     ----
     Var
     ----
     Procedure
     ----
     Function
     ----

Implementation
     Uses
     ----
     Label
     ----
     Conts
       ----
     Type
     ----
     Var
     ----
     Procedure
     ----
```

```
     Function
     ----

Initialization
     Begin
     ----
     End;
Finalization
     Begin
     ----
     End;
     ----
End.
```

Ada tiga jenis unit yang dapat Anda, bentuk:

1. Unit yang terikat dengan sebuah form, yaitu unit yang mempunyai bentuk paling umum digunakan.
2. Unit yang hanya digunakan untuk menyimpan function dan procedure.
3. Unit yang digunakan untuk membangun komponen.

Unit yang terikat dengan sebuah form mempunyai struktur penulisan program sebagai berikut:

```
Unit Unit1;

interface

     uses
       Windows, Messages, SysUtils, Classes,
       Graphics, Controls, Forms, Dialogs;
     Type

     Fform1 = class (TForm)
       Procedure FormCreate(sender: TObject);
     private
       {Private declarations}
     public
       {Private declarations}
     end;
var
     Form1 : TForm1;
```

```
Implementation
{$R *.DFM}
procedure TForm1.FormCreate(Sender: TObject);
begin

end;
end.
```

Berikut ini adalah penjelasan dari masing-masing bagian yang dapat ditemukan atau dibentuk dalam sebuah unit:

- ***Header unit***, dinyatakan dengan kata Unit yang diikuti dengan nama unit yang juga merupakan file unit yang disimpan dengan ekstensi.pas.
- ***Interface*** merupakan bagian yang dapat berisi deklarasi tipe data (termasuk kelas), konstanta, variabel, procedure atau function. Segala sesuatu yang dideklarasikan pada bagian ini dapat diakses oleh unit lain. Bagian ini harus diletakkan sebelum kata kunci unit dan sebelum bagian implementation.
- ***Implementation*** merupakan bagian yang berisi implementasi metode kelas, procedure dan function yang telah dideklarasikan pada bagian interface. Bagian ini juga dapat berisi deklarasi tipe data, variabel, konstanta, procedure atau function yang bersifat internal terhadap unit. Bagian implementation harus diletakkan setelah kata kunci interface dan sebelum initialization atau finalization (jika ada).
- ***Initialization*** merupakan tempat untuk melakukan inisialisasi data. bagian ini harus diletakkan setelah kata kunci implementation dan sebelum end atau finalization (jika ada).
- ***Finalization*** merupakan tempat untuk melakukan pembersihan, seperti mendealokasikan memori. Bagian ini harus diletakkan setelah kata kunci initialization atau implementation dan sebelum end.
- ***Uses*** merupakan klausa yang menyatakan library yang akan dikomplikasi menjadi file eksekusi. Delphi secara otomatis akan menambahkan beberapa unit yang dituliskan pada

bagian ini. Anda juga dapat menambahkan unit buatan anda sendiri pada bagian ini.

- ***Type*** merupakan bagian yang digunakan untuk mendeklarasikan variabel
- ***Private.*** Modul dalam suatu private tidak dapat dipanggil dari modul lain. Properti dalam suatu private tidak dapat dibaca atau dituliskan pada modul lain.
- ***Public***. Modul dalam suatu public dapat dipanggil dari modul lain. Properti dalam suatu public dapat dibaca atau dituliskan pada modul lain.
- ***Var*** merupakan bagian yang dapat digunakan untuk mendeklarasikan variabel termasuk variabel objek.
- ***{$R *.DFM}***. Pada sebuah unit, Delphi menambahkan {$R *.DFM}. ini sangat penting karena untuk mengikat form ke file.dfm. jangan membuang bagian ini.

2.4.1 Menggunakan Unit

Untuk menggunakan unit, Anda harus menambahkan nama unit tersebut pada klausa uses. Klausa uses adalah tempat Anda menuliskan unit-unit yang akan dipakai disebuah program atau unit. Contoh berikut menunjukkan penggunaan sebuah unit yang bernmama Unitku:

```
Uses Unitku;
```

Sebuah unit dapat mempunyai dua buah klausa uses, yaitu pada bagian interface dan bagian implementation.

```
Unit Unitku;
Interface
Uses Unitku1;
  {bagian deklarasi public}
Implementation
Uses Unitku2;
  {bagian deklarasi private}
Initialization
  {bagian inisialsasi unit}
Finalization
  {bagian membersihkan unit}
End.
```

2.5 Jenis File Dan Komponen Delphi

2.5.1 File–File Pendukung Project

Sepintas sebuah program aplikasi yang dibuat dengan menggunakan Delphi hanya dari file project dan sebuah unit. Hal ini memang sering kita temukan saat kita melakukan proses penyimpanan. Namun kenyataannya terdapat beberapa file yang dibentuk pada saat anda membanun sebuah program aplikasi. Sedangkan file lain seperti bitmap. Ikon dan lain–lain merupakan file–file yang diambil dari sumbe lain.

File Project (Dpr) dan Fiole Unit (Pas)

File Project dipakai untuk menyimpan informasi mengenai form dan unit file tersebut berbentuk pada saat Anda memuat desain progam project yang berisikan, inisialisasi form utama dan form–form lain yang dibuat.

```
Program Project

Uses
      Forms
     Unit 1 in' Unit 1 Pas' (Form 1)
($R*, Res )
begin
     Aplication. Initialize ;
     Aplication. Create Form (Tform 1, Form 1)
     Aplication Run
End
```

Program tersebut meletakankan Unit 1 pada bagian Uses. File Project menyatakan semua unit dari form yang dipakai dalam project.

```
($R*. Res)
```

Pernyataan di atas menyatakan File resource Project, di mana file Resource berisi ikon program dan informasi versi. Pernyataan tersebut merupakan kompiler untuk menghubungkan project

dengan file resource yang namanya sama dengan nama file project dengan ekstensi RES

Sedangkan file unit dipakai untuk menyimpan program untuk menyiompan program. Terdapat tiga jenis unit, yaitu:

- ***Unit Form.*** Jenis unit ini dibuat secara otomatis oleh Delphi. Ada satu untuk setiap form dan Anda tidak dapat mempunyai dua form yang didefenisikan di dalam satu Unit.
- ***Unit Component.*** Jenis unit yang akan berbentuk pada saat Anda mulai membuat komponen baru.
- ***Unit Umum.*** Jenis unit yang akan dibuat untuk data, variabel, procedure dan class yang dapat digunakan dan class yang dapat digunakan dan diaplikasikan

File Form (.Dfm)

File form adalah file biner yang dibuat Delphi untuk menyimpan informasi yang berkaitan dengan form, dan setiap form mempuyai sebuah file unit (.Pas). Jika Anda melihat file unit dari sebuah form, anda akan melihat pernyataan berikut:

```
($R*. DFM)
```

Pernyataan di atas merupakan suatu kompiler untuk menghubungkan form dengan file resource yang namanya sama dengan nama file unit dengan ekstensi. DFM. Anda dapat memanggil file form ke dalam lembar kerja editor dan mengubah teks dari file tersebut. Cara melakukannya adalah:

1. Pilih perintah **Open** dari menu **File**
2. Pilih nama File yang akan dibuka
3. Dari form designer, klik kanan untuk membuka pop–Up menu dan pilih perintah **View as Text**

File Resource (Res)

File Resource merupakan File biner yang berisi sebuah ikon yang digunakan oleh project. File ini terus menerus di Update atau diubah oleh Delphi sehingga file ini tidak bisa diubah oleh pemakai.

Anda dapat menambahkan file resource pada aplikasi dan menghubungkan dengan file project, di mana Anda dapat menggunakan sebuah editor resource, misalnya Image Editor untuk membuat file resource

File Project Options (Dof) dan File Dekstop Setting (Dsk)

File Project options meupakan file yang berisi opsion–opsion dari suatu project yang dinyatakan melalui perintah options dari menu Project. File ini tersimpan pada saat file project disimpan. Sedangkan file dekstop setting berisi opsion–opsion yang dinyatakan melalui perintah environment Options dari menu Tool. Perbedaan diantara kedua jenis file tersebut adalah *file project options* dimiliki oleh setiap project sedangkan *file dekstop setting* dipakai untuk lingkungan Delphi

Kerusakan yang terjadi pada kedua jenis file tersebut dapat mengganggu proses kompilasi. Prosedur yang dapat menghapus kedua file tersebut, yaitu Dof dan .Dsk, karena kedua File tersebut akan terbentuk secara otomatis pada saat Anda menyimpan Project

File Backup (-dp,.-df, .-pa)

File–file dengan ekstensi di atas merupakan file backup dari suatu project, form dan unit, dan ketiga jenis file tersebut akan berbentuk pada saat proses penyimpanan untuk yang kedua kalinya. Karena ketiga file tersebut berjenis backup (cadangan) yang ketiga jenis file tersebut berisi salinan terakhir dari file–file utama sebelum disimpan lebih lanjut.

Apabila Anda menghendaki untuk tidak membentuk file backup, maka Anda dapat menghilangkan fasilitas pembentukan file backup, dengan perintah berikut:

1. Klik kanan di dalam Code Editor untuk menampilkan menu Pop–Up.
2. Pilih perintah sehingga akan tampil kotak dialog Edit Properties.

3. Aktifkan tab Display dan hilangkan tanda centang pada pilihan kota cek Create Backup File

File Jenis Lain

File–file dengan Ekstensi lain yang dapat ditemukan dalam folder tempat penyimpanan program aplikasi selain ekstensi yang telah disebutkan pada umumnya adalah file–file yang dibentuk oleh kompiler dan beberapa file window yang digunakan Delphi. File–file lain tersebut adalah:

- ***File Executabel (. Exc),*** file ini dibentuk oleh kompiler dan merupkan file eksekusi (Execuatable) dari program aplikasi Anda. File ini berdiri sendiri dan hanya memerlukan file library di DLL, VBX dan lain–lain.
- ***File unti Object (Dcu),*** file ini merupakan file unit (pas) yang telah dikompilasikan oleh kompiler yang akan dihubungkan dengan file eksekusi.
- ***File dinamic link Library (Dll),*** file ini dibentuk oleh kompiler ketika Anda merancang .DLL sendiri.
- ***File Help (Hlp),*** file ini merupakan file windows dan merupakan file help standar yang dapat digunakan diprogram aplikasi Delphi
- ***File Image (*. Wmf. BMP. Ico),*** file–file ini merupakan file windows dari program aplikasi yang dapat digunakan untuk mendukung program aplikasi yang Anda rancang agar tampak lebih mendukung program aplikasi yang Anda rancang agar tampak lebih menarik.

2.5.2 Komponen Delphi

Component palette terdiri dari beberapa komponen yang dapat dipilih yang digunakan untuk menangani beberapa komponen yang dapat dipilih yang digunakan untuk menangani beberapa tugas pemograman. Anda dapat menambah, menghapus dan mengelola komponen–komponen yang terdapat dalam componentt palette, dan Anda dapat membuat komponen yang berstatus sebagai *template* dan *frame* yang digunakan untuk mengelompokkan beberapa komponen

lain. Komponen–komponen yang terletak pada bagian Component palette sudah ditata dalam beberapa tab yang masing–masing menunjukkan maksud dan fungsi masing–masing tab ditampilkan dalam konfigurasi default yang itu semua juga tergantung pada versi program Delphi yang Anda gunakan

Tabel berikut menunjukkan daftar tab default dan beberapa komponen yang terdapat di dalamnya.

Tabel 2.1 *Daftar tab default*

Nama Tab	**Isi**
Standart	Kontrol–kontrol standar program windows dan menu.
Aditional	Kontrol–kontrol tambahan.
Win32	Kontrol–kontrol umum windows 9x/NT 4.0
System	Komponen dan kontrol dari sistem Komputer termasuk timer, multimedia dan DDE.
Data Access	Komponen–komponen non visual yang digunakan untuk mengakses tabel–tabel database, guery, dan report (laporan).
Data Control	Komponen–komponen Visual dan kontrol – kontrol data.
dbExpress	Komponen-komponen non visual yang digunakan aplikasi untuk berhubungan dengan database dengan menggunakan dbEpress.
DataSnap	Komponen dan kontrol-kontrol non-visual yang digunakan untuk membuat aplikasi database bertingkat *(multi-tiered).*
BDE	Komponen dan kontrol-kontrol non-visual yang digunakan untuk menghubungkan informasi database dengan menggunakan Borland Database Engine (BDE).
ADO	Komponen dan kontrol-kontrol non-visual yang digunakan untuk menghubungkan

	informasi database dengan menggunakan ActiveX Data Objects (ADO)
InterBase	Komponen dna kontrol-kontrol non-visual yang digunakan untuk menghubungkan secara langsung database Interbase (suatu hubungan database pada server) tanpa menggunakan BDE maupun ADO.
InternetExpress	Komponen yang digunakan untuk membangun aplikasi InternetExpress yang simultan dengan Web Server dan klien dari suatu aplikasi database bertingkat.

ooo^ooo

Bab 3 Operator Dan Tipe Data

3.1 Operator

Dalam melaksanakan proses pengelolahan data, Delphi menyediakan berbagai operator dengan urutan atau derajat proses pelaksanaan yang berbeda untuk beberapa operator yang dilibatkan pada suatu proses. Berikut ini adalah tabel derajat proses pengolahan data yang dimiliki Delphi.

Tabel 3.1 *Derajat proses pengolahan data*

Urutan	Operator
1	@, not
2	*, /, div, mod, and, shl, shr, as
3	+, -, or, xor
4	=, <,>, <=, >=, <>, in, is

Apabila terdapat ekspresi A + B / C, maka operasi yang akan dikerjakan terlebih dahulu B * C, baru setelah itu hasilnya ditambahkan dengan A. urutan proses tersebut terjadi karena proses perkalian mempunyai urutan derajat pengoperasian yang lebih tinggi dari penjumlahan.

Namun jika bentuk ekspresi diubah menjadi (A + B) / C, maka operasi yang akan didahulukan (A+B), kemudian hasilnya dikalikan dengan C.

Hal ini menunjukkan bahwa tanda kurung dapat digunakan untuk mendahulukan proses. Apabila ada proses yang terdiri dari beberapa operator yang mempunyai urutan derajat yang sama, maka proses pengoperasian akan mendahulukan proses yang ditulis paling kiri.

3.1.1 Operator Assignment

Assignment atau operator fungsi dituliskan dengan bentuk ":=" (titik dua sama dengan) dan berfungsi untuk memasukkan suatu nilai data ke dalam sebuah variabel, dengan bentuk penulisan:

Nama Variabel := Ekspresi ;

Contoh :

```
Harga : = 100;
Jumlah      : =   20;
Total : = Harga * Jumlah ;
```

3.1.2 Operator Aritmatika

Berikut ini adalah tabel operator aritmatika yang dapat digunakan untuk mendukung operasi aritmatika:

Tabel 3.2 *Operator aritmatika*

Operator	Fungsi	Tipe yang Diperoses	Tipe Hasil Proses
*	Perkalian	Integer, Real	Integer, Real
/	Pembagian real	Integer, Real	Integer, Real
+	Penjumlahan	Integer, Real	Integer, Real
-	Pengurangan	Integer, Real	Integer, Real
Div	Pembagian integer	Integer	Integer
Mod	Sisa hasil bagi	Integer	Integer

Contoh:

```
Angka : = 15 * 2, {Hasil 30}
Angka : = 18 / 2, {Hasil 9}
Angka : = 5 + 2,  {Hasil 7}
Angka : = 5 - 2,  {Hasil 3}
Angka : = 10 Div 2;     {Hasil 5}
Angka : = 10 mod 2;     {Hasil 0}
```

Hasil pengoperasian 10 div 2 menghasilkan angka 5 dengan pembulatan ke bawah, dan 10 mod 2 menghasilkan angka 0 dengan pembulatan ke atas. Sedangkan untuk operasi perpangkatan, Anda dapat menggunakan rumus :

$$A^b = \exp \{ b * \ln (a))$$

Contoh :

```
Angka1 : = 5;
Angka2 : = 2;
Hasil1   : = exp (Angkat1*In ( Angka 2)); {Hasil 25}
Hasil2   : = expe (Angka 2*In (Angka 1)); {Hasil 32}
```

3.1.3 Operator Relasi

Operator relasi berfungsi untuk membandingkan suatu nilai (ekspresi) dengan nilai (ekspresi) lain yang akan menghasilkan suatu nilai logika (boolean) yaitu **True** atau **False**. Kedua data nilai yang dibandingkan tersebut harus memiliki tipe data yang sama. Berikut ini adalah tabel operator relasi yang disediakan Delphi :

3.1.4 Operator Logika

Operator logika dibagi menjadi dua kelompok: ***Operator bit*** dan ***operator boolean***. Operator bit berhubungan dengan pergeseran atau pembandingan pada level bit. Operator boolean digunakan untuk menyatakan satu atau lebih data atau ekspresi logika yang akan menghasilkan nilai logika (boolean) yang baru True atau False.

Tabel 3.3 *Operator logika*

Operator	Keterangan	Tipe Data	Tipe Hasil
And	Dan	Integer	Boolean

Or	Atau	Integer	Boolean
Not	Tidak	Integer	Boolean
Xor	Exclusive Or	Integer	Boolean
Shl	Geser ke kiri	Integer	Boolean
Shr	Geser ke kanan	Integer	Boolean

Operator **boolean** selalu memberikan hasil true atau false, sedangkan operasi bit melakukan operasi bit per bit pada nilai tipe integer.

Operator	Keterangan	Tipe Data	Tipe Hasil
And	Dan	Boolean	Boolean
Or	Atau	Boolean	Boolean
Not	Tidak	Boolean	Boolean
Xor	Exclusive Or	Boolean	Boolean

Operator logika **And** hanya akan menghasilkan nilai True jika lebih dari satu ekspresi yang menggunakan operator Anda bernilai True. Apabila dari satu ekspresi atau lebih bernilai False maka operator logika Anda akan menghasilkan nilai False.

Contoh:

```
X : = (21 > 9) And (19 < 71); {Hasil x = True}
X : = (21 < 9) And (19 < 71); {Hasil x = False}
X : = (21 < 9) And (19 < 71); {Hasil x = False}
```

Operator logika **Or** akan menghasilkan nilai True jika salah satu atau seluruh ekspresi yang menggunakan operator Or bernilai True. Operator Or hanya akan bernilai False jika semua ekspresi yang menggunakan operator Or bernilai False.

Contoh :

```
X : = (21 > 9) Or (19 < 71);  {Hasil x = True}
X : = (21 < 9) Or (19 < 71);  {Hasil x = True}
X : = (21 < 9) Or (19 > 71);  {Hasil x = False}
```

Operator logika **Not** merupakan operator yang menyatakan kondisi kebalikan dari suatu ekspresi.

Contoh :

```
X : = Not (21 > 9)      {Hasil x = False}
```

```
X : = Not (21 < 9)      {Hasil x = True}
X : = Not (21 = 9)      {Hasil x = True}
```

Operator logika **Xor** akan menghasilkan nilai True jika ekspresi yang terletak dikiri operator Xor berbeda dengan ekspresi sebelah kanannya.

Operator Xor hampir memiliki fungsi yang sama dengan operator tidak sama dengan.

Contoh :

```
X : = (9 < 21) Xor (19 < 71); {Hasil x = False}
X : = (9 > 21) Xor (19 > 71); {Hasil x = False}
X : = (9 < 21) Xor (19 > 71); {Hasil x = True}
X : = (9 > 21) Xor (19 < 71); {Hasil x = True}
```

3.2 Tipe Data

Pada bahasan ini akan diulas beberapa tipe data yang dimiliki oleh Borland Delphi. Pemilihan tipe data sangat penting untuk dilakukan supaya kita dapat menggunakan data dengan benar.

Pemilihan tipe data yang tepat akan sangat berguna dalam penghematan memori, kecepatan proses, ketelitian penghitungan dan lain-lain. Ada beberapa hal yang perlu untuk diperhatikan dalam menentukan tipe data:

1. Penggunaan memori.

   Masing-masing tipe data memiliki perbedaan dalam hal penggunaan memori. Dalam hal ini usahakan untuk menggunakan tipe dan yang memiliki memori yang kecil.

2. Ketelitian penghitungan

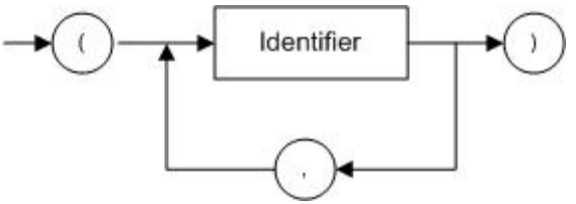
   Walaupun unsur penghematan memori sangat penting untuk diperhatikan, namun kebenaran dalam hal ketelitian penghitungan jauh lebih penting. Oleh sebab itu gunakanlah tipe data yang mempunyai tingkat ketelitian (presisi) paling tinggi.

Disamping tipe data standar yang telah disediakan oleh Delphi atau disebut dengan ***standard data type***. Dengan Delphi Anda juga masih dapat membuat dan mendeklarasikan tipe data sesuai dengan kebutuhan Anda, tipe data ini disebut dengan ***user defined data type***. Tipe data yang dideklarasikan ini dapat berupa tipe data scalar dan tipe data subrange.

3.2.1 Tipe Data Skalar

Tpe data skalar ***(scalar type)*** atau disebut dengan tipe data terbilang ***(enumerated type)*** atau disebut juga dengan tipe data skalar terdeklrasi ***(declared scalar type)*** menunjukkan kumpulan dari nilai yang urutannya sudah pasti. Nilai dari tipe yang dideklarasikan ini akan diawali dengan pengenal-pengenal *(identifiers)* yang akan menjadi suatu nilai konstanta.

Berikut ini adalah sintaks dari tipe data skalar:



Gambar 3.1 *Diagram sintaks tipe data skalar*

Contoh:

```
(Januari, Februari, Maret, April , Mei, Juni)
```

Januari, Februari, Maret, April, Mei, dan Juni adalah tipe data skalar berupa identifier yang akan menjadi suatu nilai konstanta. Tipe data scalar juga termasuk tipe data ordinal, karena mempunyai urutan nilai yang sudah pasti. Pada contoh nilai dari data Januari adalah urutan ke 0, nilai Februari adalah urutan ke 1, dan seterusnya.

Deklarasi Tipe Data Skalar

Tipe data skalar ini harus dideklarasikan terlebih dahulu dibagian deklarasi tipe yang diawali dengan kata cadangan Type.

Contoh:

```
Type
        Bahasa=(Pascal, Delphi, C++, Basic, Cobol);
        Jurusan=(IK, SI, MI, KA);
```

Penggunaan Tipe Data Skalar

Setelah tipe data skalar dideklarasikan deibagian deklarasi tipe, maka suatu variable dapat dideklarasikan dengan tipe data skalar ini sebagai berikut:

```
Type
        NamaBulan=(Januari, Februari, Maret, April, Mei,
        Juni);
Var
        Bulan:NamaBulan;
```

Variabel bulan telah dideklarasikan sebagai tipe data yang didefinisikan sendiri, yaitu bertipe NamaBulan. NamaBulan adalah tipe data skalar. Setelah variabel Bulan dideklarasikan dengan tipe data skalar ini, selanjutnya dapat digunakan di dalam program.

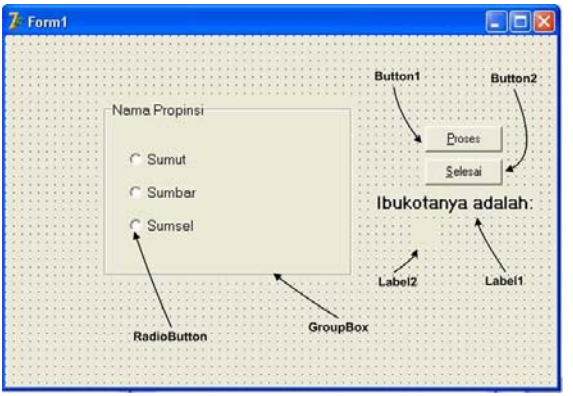
Di bawah ini adalah tipe data yang sama dengan barusan kita deklarasikan.

```
Type Propinsi=(Sumut, Sumbar, Sumsel);
Var
   IbuKotanya: Propinsi;
```

Propinsi adalah tipe data skalar yang didefinisikan sendiri, dimana Sumut menempati urutan ke 0, Sumbar urutan ke 1 dan seterusnya.

Contoh:

1. Buat Form seperti berikut:



Gambar 3.2 *Desain Form*

2. Ganti masing-masing properti untuk semua kontrol dengan nilai seperti di bawah ini:

Kontrol	Properti	Nilai
Label1	Caption	Ibu Kotanya adalah
Label2	Caption	Kosongkan
GroupBox1	Caption	Nama Propinsi
RadioButton1	Caption	Sumut
RadioButton2	Caption	Sumbar
RadioButton3	Caption	Sumsel
Button1	Caption	&Proses
Button2	Caption	&Selesai

3. Masukkan kode berikut untuk RadioButton1 (Sumut)

```
Procedure TForm1.RadioButton1Click(Sender:
TObject);
Begin
     Ibukotanya:=Sumut;
End;
```

4. Masukkan kode berikut untuk RadioButton2 (Sumbar)

```
Procedure TForm1.RadioButton1Click(Sender:
TObject);
Begin
     Ibukotanya:=Sumbar;
End;
```

5. Masukkan kode berikut untuk RadioButton3 (Sumsel)

```
Procedure TForm1.RadioButton1Click(Sender:
TObject);
Begin
     Ibukotanya:=Sumsel;
End;
```

6. Masukkan kode program berikut untuk tombol Proses

```
Type Propinsi=(Sumut, Sumbar, Sumsel);
Var
     Ibukotanya:Propinsi;
Procedure TForm1.Button1Click(Sender:TObject);
Begin
   Case Ibukotanya Of
     Sumut: Label1.Caption:='Medan';
     Sumbar: Label1.Caption:='Padang';
     Sumsel: Label1.Caption:='Palembang';
   End;
End;
```

7. Masukkan kode program berikut untuk tombol Selesai

```
ProcedureTForm1.Button2Click(Sender.TObject)
Begin
     Close;
End;
```

Program selengkapnya adalah sebagai berikut:

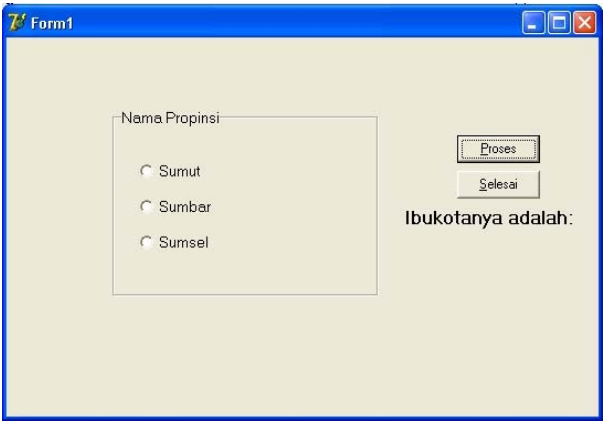
```
unit Unit1;

interface

uses
  Windows, Messages, SysUtils, Variants, Classes,
Graphics, Controls, Forms,
  Dialogs, StdCtrls;

type
  TForm1 = class(TForm)
    Label1: TLabel;
    Label2: TLabel;
    GroupBox1: TGroupBox;
    RadioButton1: TRadioButton;
    RadioButton2: TRadioButton;
```

```
    RadioButton3: TRadioButton;
    Button1: TButton;
    Button2: TButton;
    procedure RadioButton1Click(Sender: TObject);
    procedure Button1Click(Sender: TObject);
    procedure RadioButton2Click(Sender: TObject);
    procedure RadioButton3Click(Sender: TObject);
    procedure Button2Click(Sender: TObject);
  private
    { Private declarations }
  public
    { Public declarations }
  end;

var
  Form1: TForm1;

implementation

{$R *.dfm}


Type Propinsi=(Sumut, Sumbar, Sumsel);
Var
     Ibukotanya:Propinsi;
Procedure TForm1.Button1Click(Sender:TObject);
Begin
   Case Ibukotanya Of
     Sumut: Label1.Caption:='Medan';
     Sumbar: Label1.Caption:='Padang';
     Sumsel: Label1.Caption:='Palembang';
   End;
End;

procedure TForm1.RadioButton1Click(Sender: TObject);
begin
     Ibukotanya:=Sumut;
end;

procedure TForm1.RadioButton2Click(Sender: TObject);
begin
 Ibukotanya:=Sumbar;
end;
```

```
procedure TForm1.RadioButton3Click(Sender: TObject);
begin
 Ibukotanya:=Sumsel;
end;

procedure TForm1.Button2Click(Sender: TObject);
begin
  close;
end;

end.
```

8. Untuk menjalankan tekan tombol F9

Maka akan ditampilkan keluaran sebagai berikut:



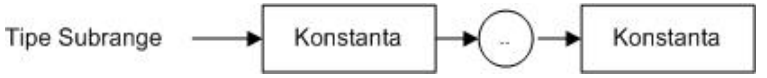
Gambar 3.3 *Hasil Run Program*

9. Tandai salah satu propinsi dengan klik sekali menggunakan mouse, kemudian klik Proses, maka akan tampil keluaran seperti berikut:

Gambar 3.4 *Hasil keluaran program ketika memilih Sumut*

3.2.2 Tipe data Subrange

Suatu tipe subrange adalah suatu range yang yang menunjukkan nilai terkecil dan nilai terbesar yang dapat dipergunakan. Tipe data subrange bila digambarkkan dalam bentuk diagram tampak sebagai berikut:



Gambar 3.5 *Diagram sintaks tipe data subrange*

Kedua konstanta di dalam subrange harus berupa tipe data ordinal yang sama dengan nilai konstanta pertama lebih kecil atau sama dengan nilai konstanta kedua.

Contoh dari tipe data subrange:

0..99
1..10
-10..5
'A'..'Z'

Deklarasi Tipe Data Subrange

Dalam Delphi yang telah Anda kenal ada tipe data Byte yang mempunyai range nilai diantaranya dari 0 sampai 255. Tipe data byte

ini sebenarnya adalh suatu subrange bertipe integer dan dapat dideklarasikan sebagai berikut:

```
Type
   Byte = 0..255;
```

Selanjutnya Anda dapat mendeklrasikan sendiri tipe data subrange lainnya dengan menyebutkan nilai terkecilnya dan nilai terbesarnya dalam suatu urutan yang menggunakan simbol ".." . tipe data real dapat digunakan sebagai nilai di subrange, karena bukan termasuk tipe data ordinal.

Contoh:

```
Type
  JamKerja  =1..10;
  Hari            =1..31;
  Huruf           ='A'..'Z';
```

Disamping itu nilai dari subrange dapat juga bertipe data skalar yang telah dideklarasikan.

Contoh:

```
Type
  Bulan     =(Jan, Feb, Mar, Apr, Mei, Jun, Jul,
Ags, Sept, Okt, Nop, Des);
  Semester1       =Jan..Juni;
  Semester2       =Jul..Des)
```

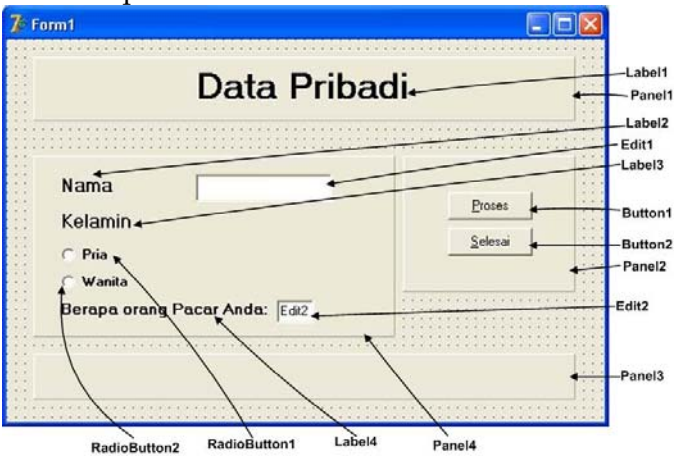
Penggunaan Tipe Data Subrange

Tipe data subrange biasanya digunakan untuk maksud program lebih mudah untuk dibaca dan dipahami, karena nilai dari suatu variabel akan terlihat dengan jelas batasannya. Keuntungan lain dari penggunaan tipe data subrange adalah untuk menghemat memori komputer. Bila suatu variabel hanya menggunakan nilai Integer positif kurang dari 256 yaitu dari 0..255 dan menggunakan tipe data byte, maka memori yang digunakan adalah sebesar 2 byte, kalau variabel ini dideklarasikan dengan menggunakan tipe data subrange dengan jangkauan nilai integer dari 0..255, maka hanya dibuthkan 1 byte memori saja.

Contoh:

Buatlah program **"Data Pribadi"**, ikuti langkah berikut ini:

1. Buat form seperti berikut ini:



Gambar 3.6 *Desain Form*

2. Ganti masing-masing properti yang ada dalam kontrol sesuai dengan form yang akan dibuat.

Kontrol	Properti	Nilai
Label1	Caption	Data Pribadi
Label2	Caption	Nama
Label3	Caption	Kelamin
Label4	Caption	Berapa orang Pacar Anda:
Label5	Caption	Kosongkan
Edit1	Name	Kosongkan
Edit2	Name	Kosongkan
RadioButton1	Caption	Pria
RadioButton2	Caption	Wanita
Panel1	Caption	Kosongkan
Panel2	Caption	Kosongkan
Panel3	Caption	Kosongkan
Panel4	Caption	Kosongkan
Button1	Caption	&Proses
Button2	Caption	&Selesai

3. Masukkan program berikut untuk Radio Button1 dan Radio Button2

```
procedure TForm1.RadioButton1Click(Sender:
TObject);
begin
  JenisKelamin:=Pria;
end;

procedure TForm1.RadioButton2Click(Sender:
TObject);
begin
  JenisKelamin:=Wanita;
end;
```

4. Masukkan program berikut untuk tombol Proses

```
procedure TForm1.Button1Click(Sender: TObject);
begin
  Nama:=Edit1.Text;
  Jumlah:=StrToInt(Edit2.Text);
  Case JenisKelamin Of
  Pria:
    Begin
      if Jumlah > 1 Then
        Label5.Caption:='Halo '+Nama+ ' Ternyata
Anda seorang Pria ' +'punya banyak pacar'
        Else
        Label5.Caption:='Halo '+Nama+ ' Ternyata
Anda seorang Pria '+'yang baik'
        End;
   Wanita:
   if Jumlah > 1 Then
        Label5.Caption:='Halo '+Nama+ ' Ternyata
Anda seorang Wanita '+'yang bahaya'
        Else
        Label5.Caption:='Halo '+Nama+ ' Ternyata
Anda seorang Wanita '+'yang baik dan jujur'
        End;

end;
```

5. Masukkan program berikut untuk tombol Selesai

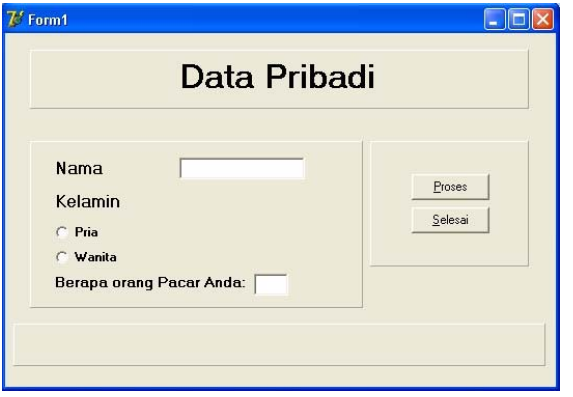
Program selengkapnya

```
unit DataPribadi;

interface

uses
  Windows, Messages, SysUtils, Variants, Classes,
Graphics, Controls, Forms,
  Dialogs, StdCtrls, ExtCtrls;

type
  TForm1 = class(TForm)
    Panel1: TPanel;
    Label1: TLabel;
    Panel2: TPanel;
    Label2: TLabel;
    Label3: TLabel;
    RadioButton1: TRadioButton;
    RadioButton2: TRadioButton;
    Label4: TLabel;
    Panel3: TPanel;
    Panel4: TPanel;
    Button1: TButton;
    Button2: TButton;
    Label5: TLabel;
    Edit2: TEdit;
    Edit1: TEdit;
    procedure RadioButton1Click(Sender: TObject);
    procedure RadioButton2Click(Sender: TObject);
    procedure Button1Click(Sender: TObject);
    procedure Button2Click(Sender: TObject);
  private
    { Private declarations }
  public
    { Public declarations }
  end;

var
  Form1: TForm1;

implementation

{$R *.dfm}
```

```
Type
  Kelamin=(Pria, Wanita);
Var
  JenisKelamin:Kelamin;
  Nama:String;
  Jumlah:Integer;

procedure TForm1.Button1Click(Sender: TObject);
begin
  Nama:=Edit1.Text;
  Jumlah:=StrToInt(Edit2.Text);
  Case JenisKelamin Of
  Pria:
    Begin
      if Jumlah > 1 Then
        Label5.Caption:='Halo '+Nama+ ' Ternyata
Anda seorang Pria ' +'punya banyak pacar'
        Else
        Label5.Caption:='Halo '+Nama+ ' Ternyata
Anda seorang Pria '+'yang baik'
        End;
   Wanita:
   if Jumlah > 1 Then
        Label5.Caption:='Halo '+Nama+ ' Ternyata
Anda seorang Wanita '+'yang bahaya'
        Else
        Label5.Caption:='Halo '+Nama+ ' Ternyata
Anda seorang Wanita '+'yang baik dan jujur'
        End;

end;

procedure TForm1.RadioButton1Click(Sender: TObject);
begin
  JenisKelamin:=Pria;
end;

procedure TForm1.RadioButton2Click(Sender: TObject);
begin
  JenisKelamin:=Wanita;
end;

procedure TForm1.Button2Click(Sender: TObject);
```

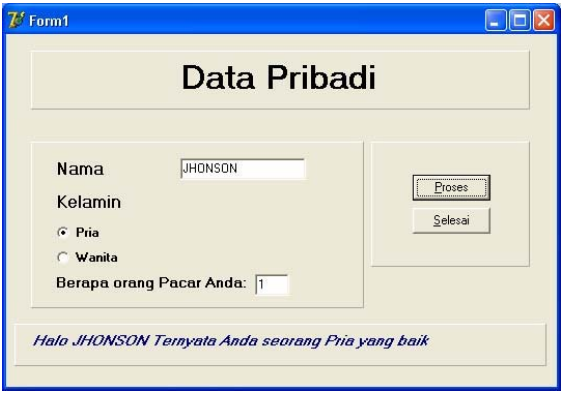
```
begin
  Close;
end;

end.
```

6. Pilih Run atau tekan F9 untuk menjalankan program, jika berhasil akan ditampilkan keluaran seperti berikut:

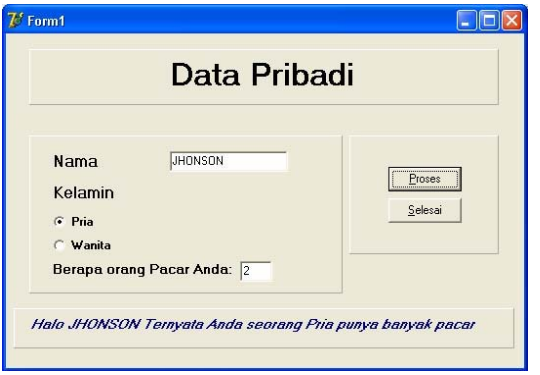


Gambar 3.7 *Keluaran program*

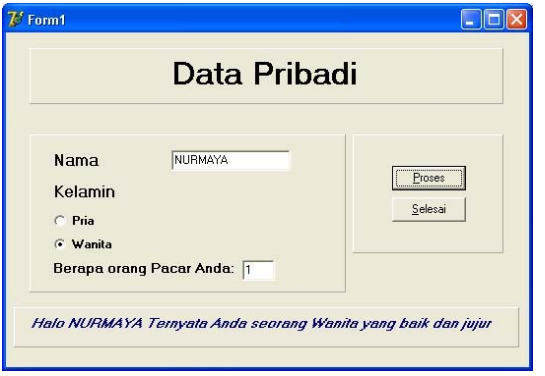
Masukkan pada kotak dialog yang ada, seperti pada contoh di bawah ini:



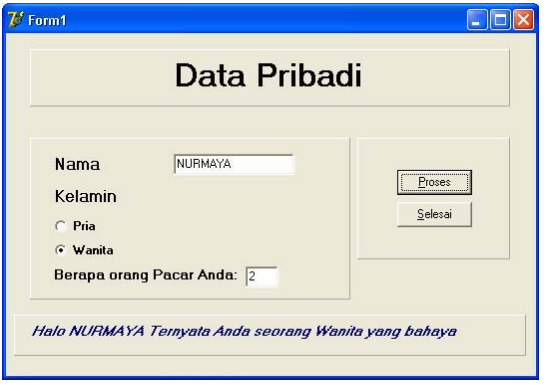
Gambar 3.8 *Keluaran program kondisi-1 pilihan Pria terpenuhi*



Gambar 3.9 *Keluaran program kondisi-2 pilihan Pria terpenuhi*



Gambar 3.10 *Keluaran program kondisi-1 pilihan Wanita terpenuhi*



Gambar 3.11 *Keluaran program kondisi-2 pilihan Wanita terpenuhi*

3.2.3 Tipe Integer

Tipe data integer digunakan untuk bilangan bulat atau bilangan yang tidak memiliki angka desimal. Tipe data integer memiliki beberapa tipe yang tergantung pada rentang nilai dan ukuran penggunaan memori.

Tabel 3.4 *Tipe data integer*

Tipe	Rentang Nilai	Byte
Byte	0 – 255	1
Word	0 – 65535	2
Shortint	-128 – 127	1
Smalllnt	-32768 – 32767	2
Integer	-2147483648 – 2147483648	4
Cardinal	0 – 2147483647	4
LongInt	-2147483648 – 2147483648	4

Semakin besar rentang nilai bilangan, semakin besar pula memori yang diperlukan.

3.2.4 Tipe Real

Tipe data real digunakan untuk bilangan yang memiliki desimal. Tipe data real memiliki beberapa tipe yang tergantung pada rentang nilai dan ukuran penggunaan memori.

Keuntungan dari penggunaan tipe data Currency adalah:

- Tipe data Currency mempunyai ketelitian yang lebih tinggi dalam menangani bilangan yang cukup besar.
- Tipe data Currency dipakai dalam Currency Field dan Komponen lain dan kompatibel dengan tipe database yang menyatakan uang.

3.2.5 Tipe Boolean

Tipe data boolean digunakan untuk data logika yang hanya berisi True (Benar) dan False (Salah). Tipe data boolean yang dapat digunakan antara lain:

Tabel 3.5 *Tipe data boolean*

Tipe	Byte
Boolean	1
ByteBool	1
WordBool	2
LongBool	4

Dari beberapa tipe yang ada disarankan untuk menggunakan tipe boolean, sedangkan untuk tipe lain hanya digunakan untuk menjaga kompatibilitas dengan program lain yang menggunakan tipe yang sama. Variabel tipe data boolean dapat menerima penggunaan operator logika AND, OR, dan NOT.

3.2.6 Tipe Character

Tipe data character digunakan untuk menyatakan karakter satu huruf dalam bentuk tiga tipe, yaitu:

Tabel 3.6 *Tipe data character*

Tipe	**Byte**	**Jumlah Maksimum**
Char	1	1 karakter ANSI
AnsiChar	1	1 karakter ANSI
WideChar	2	1 karakter Unicode

3.2.7 Tipe String

Tipe data string digunakan untuk menyatakan sederetan karakter, misalnya nama, alamat, kota dan lain-lain. Adapun tipe-tipe dari data string, yaitu:

Tabel 3.7 *Tipe data character*

Tipe	Byte	Jumlah Maksimum
ShortString	2 – 256	256 karakter
AnsiString	4 – 2 – GB	2^{31} karakter
String	255 – 3 GB	2^{31} karakter
WiderString	4 – 2 GB	2^{30} karakter

Tipe ShortString berfungsi untuk menyesuaikan kompatibilitas dengan versi sebelumnya, sedangkan AnsiString dan WideString dapat digunakan untuk menyimpan karakter Unicode. Variabel dengan tipe data string mampu menangani data string yang hampir tidak terbatas (3 GB).

3.2.8 Tipe Array

Array adalah suatu variabel tunggal yang digunakan untuk menyimpan sekumpulan data yang sejenis. Anda dapat memanipulasi, menyalin array hanya dengan sebuah nama. Dalam tipe data array menggunakan nomor elemen dikanan nama array.

Contoh:

```
Var
     Hari : array [1..7] of String;
     Mi05_01:array[1..30]0f string;
Begin
     Hari [1] : = ‘Senin’;
……
……
Hari [7] : = ‘Minggu’;
End
```

3.2.9 Tipe Record

Tipe data record digunakan untuk menyimpan sekumpulan data yang mungkin mempunyai tipe yang berbeda tetapi saling berhubungan. Elemen-elemen dalam array mempunyai tipe yang sama, tetapi elemen-elemen record dapat mempunyai tipe yang berbeda. Misalnya tipe data record yang digunakan untuk menampung data barang yang berisi kode, nama, dan harga barang.

Procedure Increment dan Decrement

Procedure Increment dan Decrement menghasilkan program yang sudah dioptimalkan untuk proses penambahan dan pengurangan pada variabel integer. Setelah dikompilasi, procedure Inc ( ) dan Dec ( ) akan menghasilkan satu perintah dalam bahasa mesin.

Anda dapat memanggil Inc ( ) atau Dec ( ) dengan satu atau dua parameter. Misalnya, perintah-perintah berikut akan menambah dan mengurangi variabel dengan nilai 1.

```
Inc (Variabel);
Dec (Variabel) ;
```

Procedure Inc ( ) dan Dec ( ) menghasilkan bahasa mesin yang sama dengan pernyataan variabel : = variabel + 1.

Contoh perintah berikut akan menambah dan mengurangi variael dengan nilai 5.

```
Inc (Variabel,5);
Dec (Variabel,5);
```

3.2.10 Tipe Varian

Tipe varian adalah data yang tipenya tidak dapat ditentukan pada saat kompilasi, karena tipe varian dapat berubah-ubah saat aplikasi dijalankan.

Contoh :

```
Var
  V : Variant;
Begin
   V := 'Komputer';    {berisi variant data string}
   V := 7;             {berisi variant data integer}
   V := True;          {berisi variant data boolean}
```

Varian dapat diisi dengan semua tipe data yang sederhana, seperti integer, floating point, string, boolean dan lain-lain. Varian dapat dipakai dengan menggunakan operator +,=,*,/, div, mod, shl, shr, and, or, xor, not,=,<>,<,>dan>=.

Contoh:

```
Var
  V1,v2,v3 : variant;
Begin
   V1 := '1000';
   V2 := '2000';
```

```
   V3 :=  3000;
   V1 := v1+v2+v3;
End;
```

Urutan pembacaan operasi contoh di atas adalah dari kiri ke kanan. Operasi pertama adalah VI + V2, karena kedua data bertipe string maka hasilnya adalah penggabungan dua string, yaitu 100002000. Kemudian hasil tersebut diubah menjadi integer dan ditambahkan dengan V3, sehingga hasilnya adalah bilangan bulat 10005000.

3.2.11 Tipe Himpunan

Tipe himpunan atau set digunakan untuk menyimpan kumpulan nilai atau dapat dikatakan sebagai anggota himpunan yang bertipe sama.

Contoh:

```
Type
HimpAngka = set of integer
```

Contoh di atas mendeklarasikan tipe HimpAngka sebagai himpunan dari karakter. Dengan pendeklarasian ini Anda bisa mendeklarasikan variabel bertipe HimpAngka, seperti yang tampak pada contoh di bawah ini :

```
VarAngka : HimpAngka;
```

Pemberian nilai pada tipe himpunan dilakukan dengan menuliskan anggota himpunan dalam kurang siku.

Contoh:

```
Angka ;=[1,2,3,4,5];
```

Pada contoh di atas, variabel angka memiliki lima anggota, yaitu bilangan 1,2,3,4 dan 5.

Berikut ini adalah tabel tipe data dalam Delphi.

Tabel 3.8 *Tipe data dalam Delphi*

Kelompok	Tipe Data	Keterangan	Memori
Integer	Integer	Seluruh bilangan antara	2 Byte

		± 32768	
	Shortint	Seluruh bilangan antara ± 128	1 Byte
	Longint	Seluruh bilangan antara ± 2747483638	4 Byte
	Byte	Seluruh bilangan antara 0 - 255	1 Byte
	Word	Seluruh bilangan antara 0 -65535	2 Byte
Real	Single	Bilangan desimal antara $1.59 X 10^{-45}$ $-3.4x10^{38}$	7-8 Byte
	Double	Bilangan desimal antara $5.0 X 10^{-325}$ $-1.7x10^{308}$	8 Byte
	Extended	Bilangan desimal antara $1.9 X 10^{-4951}$ $-1.1x10^{4932}$	10 Byte
	Comp	Seluruh bilangan desimal antara -263+1 sampai 263-1	8 Byte
	Real	Bilangan desimal antara $2.9 X 10^{-23}$ sampai $1.7x10^{38}$	6 Byte
Boolean	Boolean	Nilai Boolean berupa true dan false	1 Byte
Char	Char	Sebuah karakter ASCII	1 Byte
String	String	Urutan naik samapi 255 karakter ASCII	1 Byte
Pointer	Pointer	Sebuah pointer tak bertipe	

ooo^ooo