DIKTAT PERKULIAHAN

PEMROGRAMAN II (TEORI & PRAKTIKUM)

BORLAND DELPHI

Oleh : Tim Programming IF UNIKOM

DAFTAR ISI

DAFTAR ISI I
BAB I PENGENALAN BORLAND DELPHI 1
BORLAND DELPHI 1
KELEBIHAN MENGGUNAKAN BORLAND DELPHI 1
APLIKASI YANG TELAH DIBANGUN DENGAN MENGGUNAKAN DELPHI 2
BAB II DASAR-DASAR OOP DENGAN DELPHI 3
OBJECT ORIENTED PROGRAMMING (OOP) DENGAN DELPHI 3
DEFINISI CLASS 3
DEFINISI OBJECT 4
ILUSTRASI DATA ABSTRACTION 4
ILUSTRASI ENCAPSULATION 5
ILUSTRASI INHERITANCE 5
ILUSTRASI POLIMORPHISM 6
MENGHIDUPKAN OBJEK 7
MEMATIKAN OBJEK 7
BAB III PENGENALAN IDE DELPHI 8
WINDOW UTAMA 9
COMPONENT PALETTE 9
OBJECT INSPECTOR 9
OBJECT TREEVIEW 10
FORM DESIGNER 11
CODE EDITOR 12
BAB IV FORM DAN KOMPONEN 13
MEMBUAT PROGRAM PERTAMA 13
MEMBUAT APLIKASI BARU 13
MENYIMPAN APLIKASI 14
MENGEKSEKUSI APLIKASI 15
MEMODIFIKASI FORM 16

MEMODIFIKASI FORM DENGAN OBJECT INSPECTOR 18
MEMODIFIKASI FORM DENGAN KODE PROGRAM 20
MENAMBAHKAN OBJEK LAIN KE DALAM FORM 22
BAB V BEKERJA DENGAN DATA 26
TIPE DATA 26
TIPE DATA UNTUK BILANGAN 26
TIPE DATA UNTUK TEKS 27
TIPE DATA BOOLEAN 28
VARIABEL 28
DEKLARASI VARIABEL 28
MENGISI NILAI KE VARIABEL 28
KONSTANTA 29
OPERATOR 29
OPERATOR ARITMETIKA 29
OPERATOR BOOLEAN 30
OPERATOR LOGIKA (BIT) 31
OPERATOR RELASIONAL 32
OPERATOR STRING 32
ATURAN PENGERJAAN OPERATOR 32
FUNGSI-FUNGSI UNTUK KONVERSI DATA 32
CONTOH PROGRAM MENGOLAH DATA 33
LATIHAN-LATIHAN 35

BAB I
PENGENALAN BORLAND DELPHI

BORLAND DELPHI

Borland Delphi adalah sebuah alat pengembangan aplikasi-aplikasi untuk sistem operasi Microsoft Windows. Delphi sangat berguna dan mudah digunakan untuk membuat suatu program berbasis GUI (Graphical user interface) atau console (mode teks).

Borland Delphi mempunyai "saudara" bernama Borland Kylix yaitu versi Delphi yang digunakan untuk membuat aplikasi pada sistem operasi Linux. Dengan dipasangkannya Borland Delphi dengan Borland Kylix maka pengembang software dapat membuat aplikasi berbasis Windows yang dapat dengan mudah dikompilasi ulang pada Linux.

Delphi merupakan bahasa pemrograman pertama yang memecahkan batasan antara bahasa tingkat tinggi, pengembangan aplikasi dengan cepat (Rapid Application Development/RAD).

Ketika membuat aplikasi GUI dengan Delphi, pengembang perangkat lunak akan mendapatkan bahasa pemrograman (dalam hal ini Object Pascal) yang dibungkus dalam lingkungan RAD. Semua user interface seperti form, tombol (button), dan objek list-list telah disertakan dalam Delphi dalam bentuk komponen atau control. Pengembang dapat dengan mudah menempatkan komponen-komponen tersebut ke dalam form. Pengembang dapat juga menempatkan control ActiveX pada form untuk membuat program-program khusus seperti Browser Web dalam waktu yang cepat. Delphi memungkinkan pengembang untuk merancang keseluruhan interface secara visual, dan dengan cepat dapat diimplementasikan sebuah kode perintah berbasis event (event driven) dengan mengklik mouse. Dengan Delphi, pengembang perangkat lunak dapat membuat program Windows dengan lebih cepat dan lebih mudah dari sebelumnya.

KELEBIHAN MENGGUNAKAN BORLAND DELPHI

Kelebihan-kelebihan yang dapat diambil ketika seorang pengembang perangkat lunak menggunakan Borland Delphi adalah :

- Delphi mendukung Pemrograman Berorientasi Objek (Object Oriented Programming/OOP)
- Pengembangan aplikasi secara cepat (Rapid Application Development/RAD)
- Menggunakan bahasa tingkat tinggi
- Hasil dari proses kompilasi berupa sebuah file yang dapat dieksekusi (executable file) sehingga mempermudah dalam pendistribusian program dan mengurangi banyaknya file pendukung DLL

- Delphi menyediakan banyak sekali komponen yang dapat digunakan. Selain itu banyak juga komponen yang bersumber dari pihak ketiga yang biasanya disertai dengan dokumentasi, source code dan lain-lain. Komponen dari pihak ketiga bisa yang komersil atau free.
- Mendukung banyak database server (MySQL, SQL Server, Interbase, Oracle dll) sehingga dapat mempermudah dalam membuat aplikasi database.

APLIKASI YANG TELAH DIBANGUN DENGAN MENGGUNAKAN DELPHI

Dengan kemudahan yang diberikan oleh Borland Delphi, telah banyak aplikasi-aplikasi terkenal yang dibangun dengan menggunakan Borland Delphi[1], diantaranya:

- Produk Borland : Borland Delphi, Borland C++ Builder, Borland JBuilder versi 1 dan 2
- Perangkat Lunak Akunting Panggilan : PhoneControl
- Game : Astral Masters, Astral Tournament, Smugglers series, Soldat, Quake conversion from the C source, Space Rangers, Space Rangers 2: Dominators
- Management Database : Tool MySQL (Administrator, Query Browser, Migration Toolkit)
- Internet Messaging : Skype (VoIP and IM), The Bat! (e-mail client), PopTray (e-mail check tool), FeedDemon (RSS/Atom feed viewer), XanaNews (newsgroup reader), Xnews (newsgroup reader)Customer relationship management: Sage SalesLogix
- Produksi Musik : FL Studio
- Pengembangan Software : : Dev-C++ (IDE), DUnit, Help & Manual (help system authoring), Inno Setup (installer engine), ConTEXT (Programmers editor)
- Pengembangan Web : Macromedia HomeSite (HTML editor), TopStyle Pro (CSS editor), Macromedia Captivate (screencast)
- Browser Web : Avant Browser, Netcaptor
- Utility : Spybot - Search & Destroy, Ad-Aware (anti-spyware), Total Commander (file manager), Copernic Desktop Search, PowerArchiver, ASuite

[1] http://en.wikipedia.org/wiki/Borland_Delphi

BAB II
DASAR-DASAR OOP DENGAN DELPHI

OBJECT ORIENTED PROGRAMMING (OOP) DENGAN DELPHI

Object Pascal merupakan bahasa dasar yang digunakan oleh Delphi. Object Pascal merupakan bahasa pemrograman yang berorientasi objek. Artinya bahwa bahasa ini membolehkan programmer untuk membuat dan memanipulasi objek. Lebih lanjut, ini artinya bahwa bahasa tersebut dapat mengimplementasikan 4 prinsip dasar dari pemrograman berorientasi objek[2] yaitu :

1. Abstraksi Data (Data Abstraction)
2. Enkapsulasi (Encapsulation)
3. Pewarisan (Inheritance)
4. Polimorpisme (Polymorphism)

Object Oriented Programming merupakan cara membuat program dengan memanipulasi objek. Delphi (begitu juga C++ dan Java) merupakan bahasa berorientasi objek. Prinsip pemrograman berbasis objek pada semua bahasa tersebut sebenarnya sama. Perbedaannya hanya pada bagian sintaknya saja. Sekali anda menguasai prinsip tersebut maka tanpa memandang bahasa yang dipakai pun anda akan dapat memahaminya dengan baik. Konsep seperti inheritance dan abstraksi data semuanya sama pada Delphi, C++, Java. Sekali lagi perbedaannya hanya pada sintak bahasanya saja.

DEFINISI CLASS

Ketika anda menggunakan bahasa pemrograman berorientasi objek, maka anda akan banyak mendengar istilah Class dan Object.

Sebuah Class mendefinisikan karakteristik abstrak dari sebuah benda (obyek), termasuk karakteristik benda (atribut atau ciri benda) dan perilaku benda tersebut (sesuatu yang bisa dilakukan oleh benda tersebut atau dikenal dengan method atau operasi). Banyak juga yang mendefinisikan class sebagai cetak biru yang menjelaskan tentang sesuatu. Contoh adalah obyek manusia. Manusia mempunyai atribut tinggi, berat badan, nama, warna rambut serta memiliki method atau operasi seperti manusia bisa melakukan makan, minum, berbicara dan lain-lain. Properti atau attribut dan method yang mendefinisikan suatu Class disebut dengan member (anggota).

Contoh pembuatan/pendeklarasian sebuah Class dengan Delphi adalah :

```
TManusia=class   {atau TManusia=class(Tobject) }
            Nama:String;
```

[2] http://www.webtechcorp.co.uk/web-developer-training-delphi-article-oop.htm, 11 Februari 2008

```
                Tinggi:Integer;
                Berat:Double;
                procedure UcapNama;
            end;

    {Implementasi method UcapNama}
    procedure TManusia.UcapNama;
    begin
         writeln('Saya adalah '+nama);
    end;
```

Pada contoh di atas, kita mempunyai sebuah kelas yang bernama TManusia yang mengabstraksikan benda manusia. Dalam class Tmanusia terdapat attribut berupa Nama, Tinggi dan Berat manusia tersebut. Setiap benda diabstraksikan dengan Nama, Tinggi, dan Berat. Sedangkan perilaku yang bisa dilakukan ole benda itu (method) adalah UcapNama. Dapat dilihat bahwa atribut dan perilaku(method) digabungkan dalam sebuah class. Itulah yang namanya enkapsulasi.

DEFINISI OBJECT

Object adalah sebuah instansiasi (instance) khusus dari sebuah class. Contoh : objek yang bernama Irma merupakan sebuah instansiasi dari sebuah class TManusia.

Untuk lebih jelas mengenai Object, perhatikan source code di bawah ini.

```
    var
       Orang:TManusia;
       Irma:TManusia;
```

Pernyataan di atas berarti kita mempunyai 2 buah object yang mempunyai kelas Tmanusia. Objek-objek tersebut adalah Orang dan Irma. Objek-objek tersebut masih belum bisa dipergunakan, karena objek harus dibuat / dihidupkan terlebih dahulu.

ILUSTRASI DATA ABSTRACTION

Abstraksi adalah penyederhanaan dari kenyataan yang kompleks dengan memodelkan class sesuai dengan masalah yang dihadapi. Jadi astraksi data bisa berarti bahwa memodelkan suatu benda dengan mewakilkan karakteristiknya dengan suatu data. Contoh : Benda seperti manusia bisa diabstraksikan dengan memiliki beberapa atribut seperti tinggi badan, berat badan, warna rambut dan lain-lain.

Contoh data abstraction dalam pendefinisian sebuah class adalah :

```
    TManusia=class   {atau TManusia=class(Tobject) karena semua objek adalah keturunan TObject}
                Nama:String;
                Tinggi:Integer;
                Berat:Double;
            end;

    TMahasiswa=class(TObject)
                   NIM:String;
                   Nama:String;
                   TempatLahir:String;
                   TanggalLahir:TDate;
            end;
```

ILUSTRASI ENCAPSULATION

Enkapsulasi berarti membungkus atribut dan method yang digunakan ke dalam class. Contoh : Manusia mempunyai method/operasi makan. Proses makan suatu objek mungkin berbeda dengan proses makan pada objek yang lain.

Contoh encapsulation dalam pendeklarasian sebuah Class dengan Delphi adalah :

```
    TOrangIndonesia=class   {atau TManusia=class(Tobject) }
                        Nama:String;
                        Tinggi:Integer;
                        Berat:Double;
                        procedure UcapNama; { ← Method disisipkan bersama atribut dalam sebuah
class}
             end;

    {Implementasi method UcapNama}
    procedure TOrangIndonesia.UcapNama;
    begin
         writeln('Saya adalah '+nama);
    end;
```

Potongan program di atas merupakan suatu potongan untuk mendefinisikan sebuah class. Dalam kelas tersebut diabstraksikan bahwa seorang ManusiaIndonesia mempunyai atribut berupa Nama,Tinggi dan Berat Badan. ManusiaIndonesia juga memiliki perilaku/method dengan nama UcapNama yang akan mengucapkan kalimat berupa "Saya adalah Andri" (jika nama orang tersebut adalah Andri). Dapat dilihat bahwa atribut dan method disatukan dalam sebuah class. Itulah yang namanya encapsulation (enkapsulasi/pengkapsulan).

ILUSTRASI INHERITANCE

Kemampuan suatu class untuk mewariskan atribut dan perilakunya kepada anak classnya (sub class). Class anak boleh direstruktur programnya sehingga mempunyai atribut dan perilaku tambahan sehingga tidak persis dengan class induknya.

Contoh encapsulation dalam pendeklarasian sebuah Class dengan Delphi adalah :

```
Type
    TOrangIndonesia=class(TObject)
                Nama:String;
                Tinggi:Integer;
                Berat:Double;
                procedure UcapNama;
             end;

    TOrangBetawi=class(TOrangIndonesia) {OrangBetawi adalah turunan dari OrangIndonesia}

             end;

    TOrangSunda=class(TOrangIndonesia)  {OrangSunda adalah turunan dari OrangIndonesia}
                    procedure UcapNama;
             end;

    TOrangJawa=class(TOrangIndonesia)   {OrangJawa adalah turunan dari OrangIndonesia}
                    procedure UcapNama;
             end;

Procedure TOrangIndonesia.UcapNama;
begin
```

```
     Writeln('Nama Saya '+Nama);
end;

procedure TOrangSunda.UcapNama;
begin
     Writeln('Nami Kuring '+Nama);
end;

procedure TOrangJawa.UcapNama;
begin
     Writeln('Nami Kulo  '+Nama);
end;
```

Pada potongan program di atas, selain kita mempunyai class TorangIndonesia, kita juga memiliki class lain yang merupakan turunan dari TOrangIndonesia yaitu TOrangBetawi, TOrangSunda dan TorangJawa. Class TOrangBetawi merupakan turunan murni dari class TOrangIndonesia tanpa mengubah atribut atau methodnya sama sekali. Ini dikarenakan diasumsikan bahwa pengucapan nama menggunakan bahasa yang dipakai oleh orang betawi cenderung identik dengan pengucapan nama dalam bahasa Indonesia. Sedangkan untuk class TOrangSunda dan TOrangJawa method UcapNama akan diganti dengan pengucapan dalam bahasa Sunda atau Jawa. Oleh karena itu procedure UcapNama dideklarasikan lagi dalam class turunannya, sehingga kalau kita menggunakan class TOrangSunda maka method UcapNama akan menampilkan pesan “Nami Kuring Andri” (Jika atribut Nama berisi Andri).

ILUSTRASI POLIMORPHISM

Polimorpisme memperbolehkan anda untuk memperlakukan anggota(member) suatu class yang merupakan class turunan sebagai anggota class induknya.

Contoh polimorphism dalam OOP dengan Delphi adalah :

```
var
   Orang1:TOrangIndonesia;
   Sunda1:TOrangSunda;
   Jawa1:TOrangJawa;
begin
     Orang1:=TOrangIndonesia.Create;
     Orang1.Nama:='Andri';
     Orang1.UcapNama;            { Menghasilkan : Nama Saya Andri}

     Sunda1:=TOrangSunda.Create;
     Sunda1.Nama:='Cecep';
     Sunda1.UcapNama;            { Menghasilkan : Nami Kuring Cecep}

     Jawa1:=TOrangJawa.Create;
     Jawa1.Nama:='Bejo';
     Jawa1.UcapNama;             { Menghasilkan : Nami Kulo  Bejo}

     Orang1:=Sunda1;             { Orang Sunda berpolimorp menjadi Orang Indonesia}
     Orang1.UcapNama;            { Menghasilkan : Nama Saya Cecep}

     Orang1:=Jawa1;              { Orang Jawa berpolimorp menjadi Orang Indonesia}
     Orang1.UcapNama;            { Menghasilkan : Nama Saya Bejo}
end.
```

MENGHIDUPKAN OBJEK

Ketika sebuah object dideklarasikan (Contoh Sunda1:TOrangSunda) bukan berarti objek tersebut bisa digunakan. Agar suatu objek dapat digunakan, maka objek tersebut harus dihidupkan terlebih dahulu. Jika sebuah objek digunakan tanpa dihidupkan terlebih dahulu maka akan menghasilkan error.

Contoh penggunaan objek tanpa dihidupkan terlebih dahulu :

```
var
   Orang1:TOrangIndonesia;
begin
     Orang1.UcapNama;
end.
```

Perintah di atas akan menghasilkan error seperti berikut :

Contoh pesan kesalahan ketika objek tidak dihidupkan terlebih dahulu :

```
Exception EAccessViolation in module Project2.exe at 0000825D.
Access violation at address 0040825D in module 'Project2.exe'. Read of address 00000004.
```

Menghidupkan sebuah objek adalah dengan memanggil method khusus yang disebut constructor. Constructor yang digunakan untuk menghidupkan suatu objek adalah constructor Create.

Contoh cara menghidupkan sebuah objek :

```
var
   Orang1:TOrangIndonesia;
begin
     Orang1:=TOrangIndonesia.Create;  { Hidupkan objek }
     { Menggunakan Objek setelah dihidupkan }
     Orang1.Nama:='Andri';
     Orang1.UcapNama;
end.
```

MEMATIKAN OBJEK

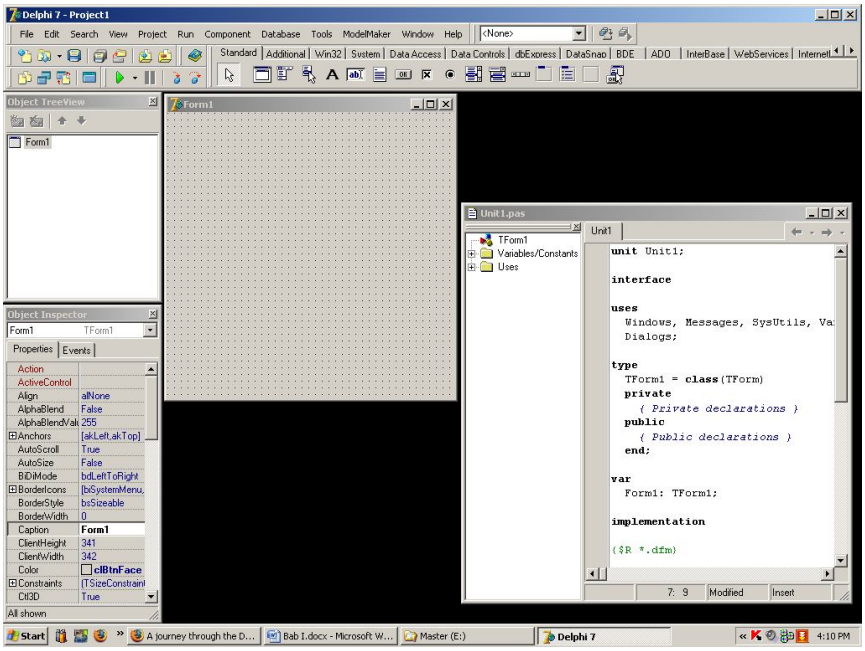
Sebuah objek membutuhkan alokasi di memori. Jika kita banyak membuat objek maka alokasi memori yang digunakan juga besar. Oleh karena itu ada baiknya jika sebuah objek yang sudah tidak digunakan lagi untuk dimatikan sehingga akan mengembalikan memori yang dipakainya. Untuk mematikan sebuah objek maka metode yang digunakan merupakan metode khusus yang disebut destructor. Destructor yang digunakan adalah destructor Destroy. Tetapi Borland sendiri menyarankan untuk mematikan objek adalah dengan menggunakan metode/procedure Free.

Contoh cara mematikan sebuah objek :

```
var
   Orang1:TOrangIndonesia;
begin
     Orang1:=TOrangIndonesia.Create;  { Hidupkan objek }
     ...
     ...
     Orang1.Free; { Matikan Objek / Hapus dari Memori }
end.
```

BAB III
PENGENALAN IDE DELPHI

Untuk memulai mempelajari Borland Delphi, langkah pertama yang biasa dilakukan adalah dengan mempelajari lingkungan pengembangan terintegrasi (Integrated Development Environment) dari Borland Delphi itu sendiri. Untuk memulai Borland Delphi, cara yang dilakukan adalah dengan mengklik tombol Start → Programs → Borland Delphi 7 → Delphi 7. Ketika Delphi dimulai, maka anda akan dihadapkan pada layar seperti di bawah ini.



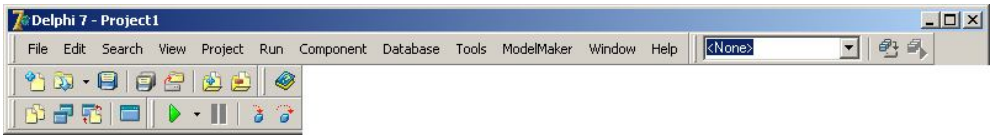
Gambar 3.1 IDE Delphi

IDE Delphi seperti yang terlihat di gambar 3.1 memiliki beberapa windows yang mempunyai fungsi tersendiri. Windows-windows itu adalah :

1. Window Utama
2. Component Palette
3. Object Inspector
4. Object TreeView
5. Form Designer
6. Code Editor

WINDOW UTAMA

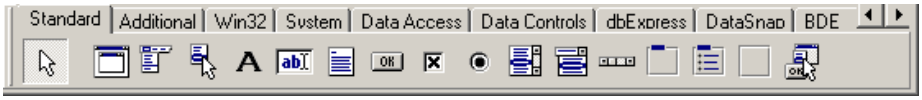
Window utama berada pada posisi atas dari layar. Window utama terdiri dari menu utama, toolbar dan component palette. Kotak judul diatas pada windows utama berisi nama dari project yang sedang dikerjakan. Kotak menu terdiri dari menu-menu drop-down. Pada bagian Toolbar terdapat sekumpulan shortcut/tombol untuk operasi-operasi yang sering digunakan (seperti menjalankan program, menambahkan form ke sebuah proyek, menyimpan unit dan lain-lain). Untuk melihat fungsi dari tiap-tiap tombol yang ada, dapat dilihat dengan meletakan cursor mouse di atas tombol tersebut. Tak lama kemudian akan muncul hint yang menampilkan fungsi dari tombol tersebut. Window utama dapat dilihat seperti pada gambar 3.2 di bawah ini.



Gambar 3.2 Menu Utama

COMPONENT PALETTE

Pada program-program yang berjalan pada sistem operasi Windows, pengguna disajikan dengan aplikasi yang terdiri dari layar dan objek-objek yang berbeda, seperti tombol, textbox, radiobutton, check box dan lain-lain. Dalam pemrograman Delphi, istilah objek-objek tersebut disebut dengan control atau komponen. Komponen adalah blok-blok bagian yang akan membentuk suatu aplikasi Delphi. Komponen-komponen tersebut dapat dilihat pada window Component Palette. Untuk menempatkan sebuah komponen ke sebuah windows, cukup dengan mengklik komponen dari component palette kemudian mengklik lokasi tempat penempatan komponen tersebut di dalam form. Setiap komponen mempunyai atribut tertentu yang memungkinkan bagi pengembang untuk mengatur aplikasi ketika waktu desain (design time) atau waktu dijalankan (run time). Component palette dapat dilihat pada gambar di bawah ini.



Gambar 3.3 Component Palette

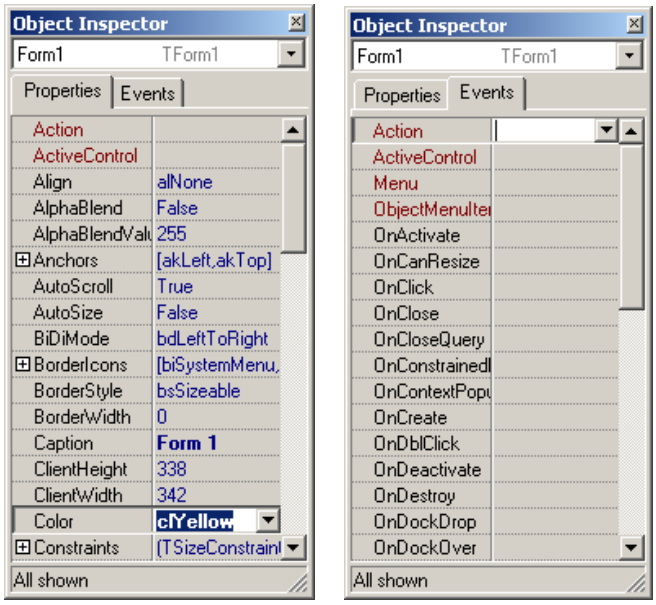
Komponen-komponen pada component palette dikelompokkan berdasarkan fungsi yang dapat dilakukannya. Setiap halaman tab pada component palette menampilkan sekumpulan icon yang mewakili komponen yang digunakan untuk mendesain interface aplikasi.

OBJECT INSPECTOR

Setiap komponen dan setiap form mempunyai sekumpulan properties (seperti warna, ukuran, posisi, judul (caption) yang dapat dimodifikasi pada IDE Delphi atau dalam kode program anda), dan sekumpulan event (seperti klik mouse, penekanan tombol) dimana anda dapat menentukan beberapa

perilaku tambahan. Objek Inspector menampilkan properties dan event untuk komponen yang sedang dipilih dan memperbolehkan anda untuk mengganti nilai properti atau memilih respon terhadap suatu event yang terjadi.

Object Inspector dapat dilihat pada gambar 3.4 di bawah ini.



Gambar 3.4 Object Inspector

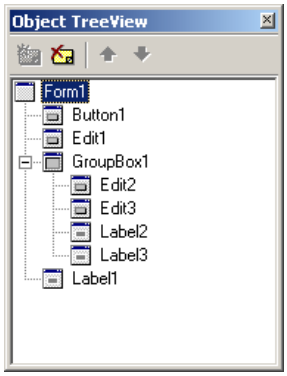
Sebagai contoh, setiap form mempunyai Caption (judul yang terlihat pada kotak judul). Untuk mengubahnya judul tersebut, caranya adalah dengan mengaktifkan form dengan mengklik form tersebut. Kemudian pada Object Inspector carilah properti Caption (pada kolom kiri), kemudian ganti isinya yang ada pada bagian kanan dengan mengisi Caption/judul yang diinginkan. Jangan lupa tekan tombol Enter jika telah yakin caption baru tersebut akan digunakan.

Mirip dengan properti, event juga dapat dipergunakan. Misalnya kita akan membuat menampilkan suatu pesan ketika form diklik, maka caranya adalah dengan aktifkan form dengan mengklik formnya kemudian di Object Inspector klik tab Event. Kemudian cari event OnClick dan kemudian double klik di bagian kanan atau isi nama metode yang akan digunakan ketika event klik terjadi.

OBJECT TREEVIEW

Di bagian atas Object Inspector anda akan melihat window Object TreeView. Awalnya window ini hanya terdiri dari nama Form. Tetapi ketika anda menambahkan sebuah komponen ke form tersebut, maka object baru tersebut akan terdaftar di dalam Object TreeView di bagian bawah form. Object TreeView akan menampilkan diagram pohon yang mencerminkan hubungan parent-child dari komponen-komponen.

Object TreeView dapat dilihat pada gambar 3.5 di bawah ini.



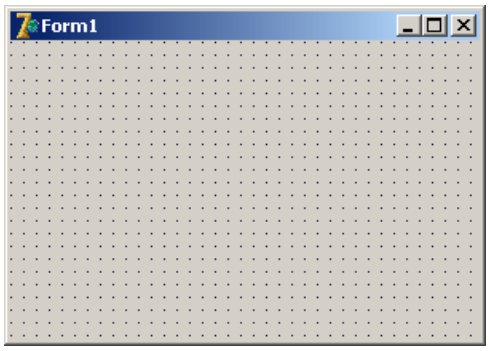
Gambar 3.5 Object TreeView

Object TreeView, Object Inspector dan Form Designer bekerja secara bersama. Jika sebuah objek dalam sebuah form diklik maka properti dan eventnya ditampilkan dalam Object Inspector dan komponen akan focus pada Object TreeView.

FORM DESIGNER

Setiap anda memulai Delphi, sebuah project dibuat yang terdiri dari sebuah window kosong. Umumnya aplikasi Delphi terdiri dari beberapa window. Window tersebut di Delphi disebut sebagai form. Awalnya form pertama akan diberi nama Form1. Kita bisa menganti namanya, melakukan perubahan ukuran atau mengatur posisinya. Window biasanya mempunyai caption/judul dan tiga tombol standar yaitu minimize, maximize dan tutup window.

Form designer merupakan suatu objek yang dapat dipakai sebagai tempat untuk merancang program aplikasi. Form berbentuk sebuah lembar kosong yang dapat diisi denagn komponen-komponen yang diambil dari Component Palette. Pada saat anda memulai Delphi, Delphi akan memberikan sebuah form kosong yang diberi nama Form1, seperti pada gambar 3.6 di bawah ini.



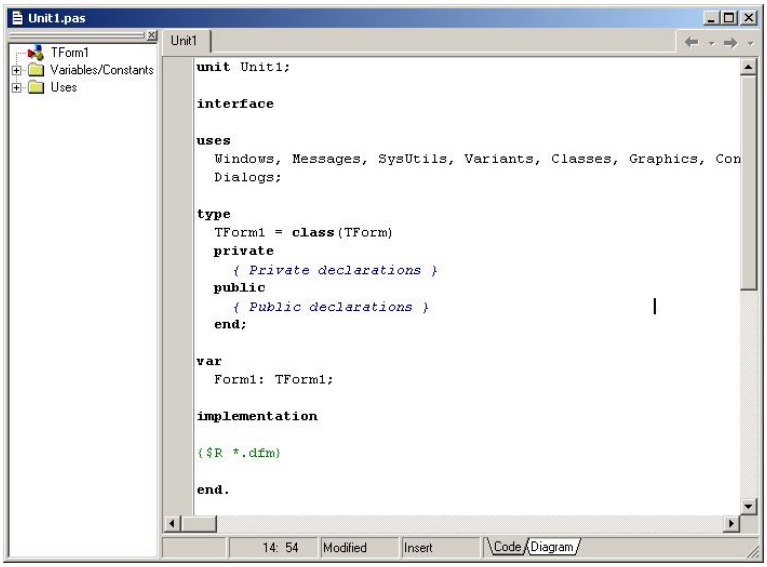
Gambar 3.6 Form kosong

Dalam sebuah form terdapat titik-titik yang disebut grid yang berguna untuk membantu pengaturan tata letak objek yang dipasang pada form.

CODE EDITOR

Code Editor merupakan tempat di mana anda dapat menuliskan kode program. Pada bagian ini anda dapat menuliskan pernyataan-pernyataan dalam Object Pascal. Keuntungan bagi pemakai Delphi adalah bahwa anda tidak perlu menuliskan kode-kode sumber, karena Delphi telah menyediakan kerangka penulisan sebuah program seperti tampak pada gambar 3.7.

Window ini akan menampilkan kode program yang sedang dibuat. Anda dapat membuka lebih dari sebuah file dalam Code Editor. Setiap file dibuka pada sebuah halaman baru dari Code Editor, dan setiap halaman diwakili dengan sebuah tab di atas window.



Gambar 3.7 Layar Code Editor

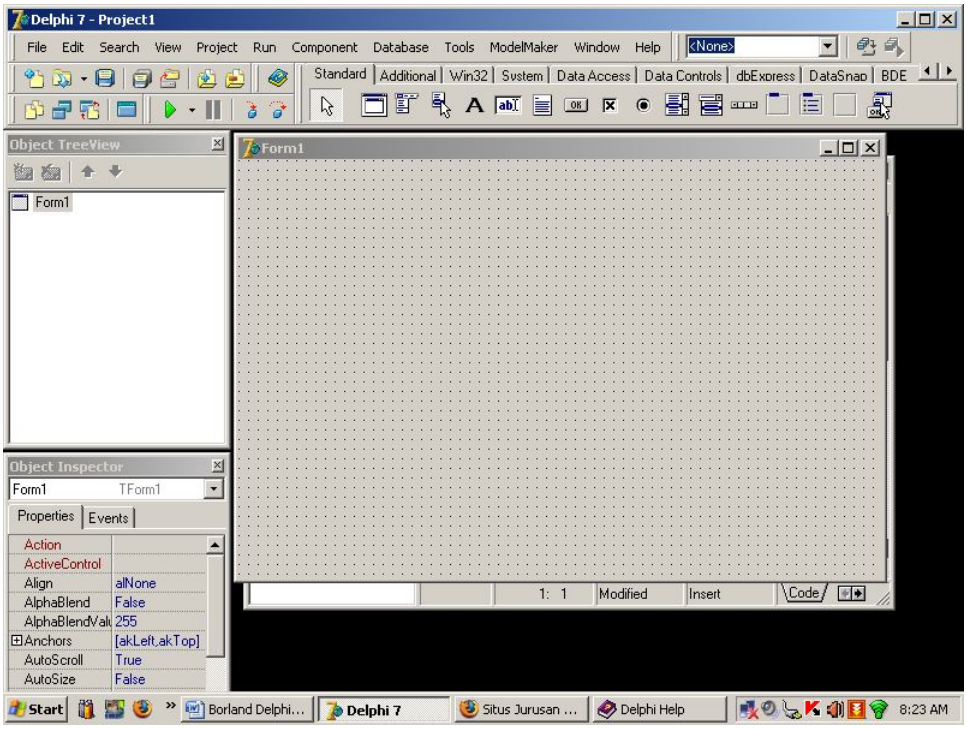
BAB IV
FORM DAN KOMPONEN

MEMBUAT PROGRAM PERTAMA

MEMBUAT APLIKASI BARU

Dalam Delphi, sebuah aplikasi/program dibangun berdasarkan project. File project berisi referensi kepada semua form dan unit yang digunakan dalam project.

Untuk membuat suatu aplikasi baru, Klik menu File → New → Application yang akan membuat suatu project baru dengan sebuah form. Gambar 4.1 memperlihatkan tampilan ketika sebuah project baru dibuat.



Gambar 4.1 Tampilan project baru

Pada gambar 4.1 dapat dilihat ketika membuat suatu aplikasi baru, maka akan muncul sebuah project baru dengan sebuah form dengan nama Form1.

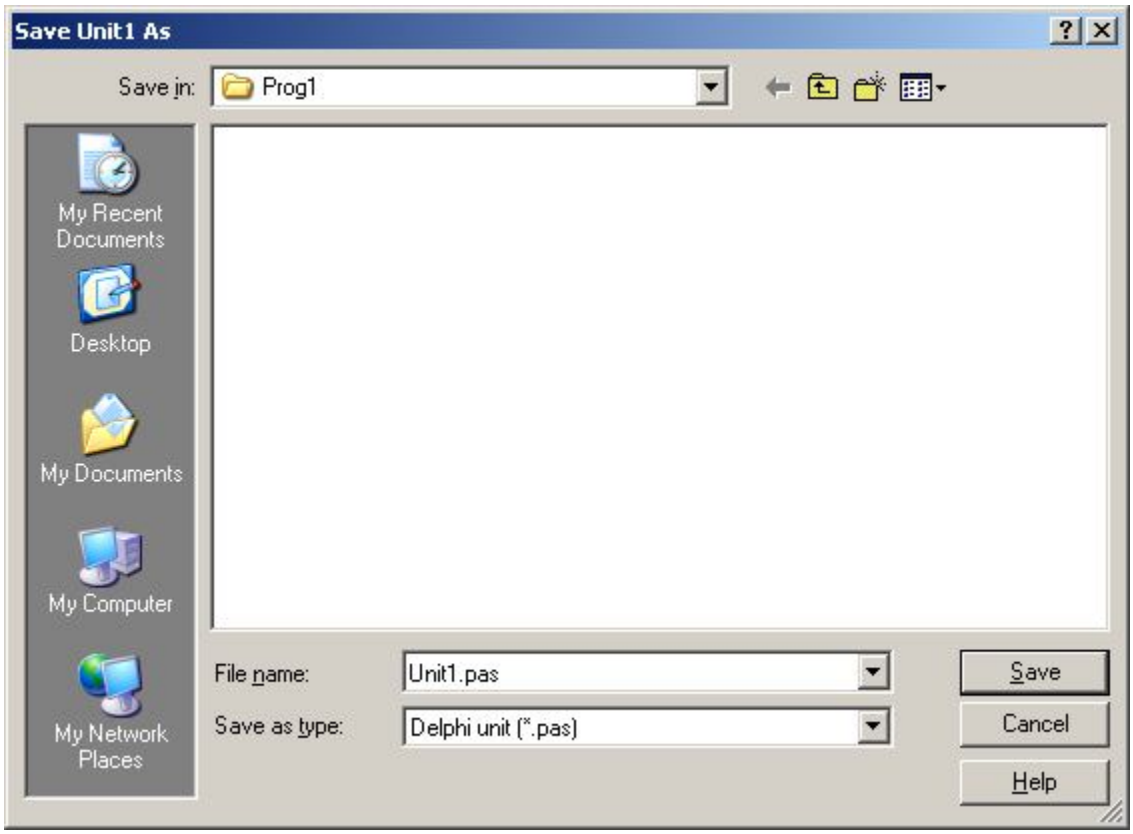
MENYIMPAN APLIKASI

Penyimpanan suatu aplikasi baru, memerlukan 2 langkah yaitu menyimpan file-file unit dan file project. Cara paling sederhana untuk menyimpan suatu aplikasi baru adalah dengan klik icon Save All ( ) atau File → Save All atau dengan menekan tombol Shift+Control+S. Save All berguna untuk menyimpan semua file yang sedang dibuka. Jadi Save All akan menyimpan file unit dan file projectnya. Perhatikan judul dialog penyimpanan file. Jika judulnya menyatakan : "Save Project1 As", itu artinya dialog penyimpanan project. Tetapi jika judulnya menyatakan : "Save Unit1 As" itu berarti dialog penyimpanan unit.

Menyimpan File Unit

Jika hanya ingin menyimpan file unitnya saja, langkah yang dilakukan adalah mengklik icon Save ( ) atau File → Save. Tapi harus diingat kalau file yang aktif di Code Editor merupakan file project, maka Save akan bekerja sebagai Save Project. File unit akan mempunyai ekstensi .pas.

Tampilan dialog penyimpanan Unit seperti pada gambar 4.2 di bawah ini.



Gambar 4.2 Tampilan dialog penyimpanan Unit

Jika project yang dibuat merupakan project baru, sangat disarankan agar sebelum anda menyimpan file, buat dulu folder baru. Klik tombol Save untuk menyimpan file unit tersebut.

Untuk mempermudah mengingat isi dari unit, sebaiknya setiap file unit diisi dengan nama sesuai dengan nama formnya dan umumnya dimulai dengan huruf U. Misalnya UTambahData.Pas untuk unit / form yang berguna untuk menambah data.

Contoh pemberian nama unit dapat dilihat pada tabel 4.1.

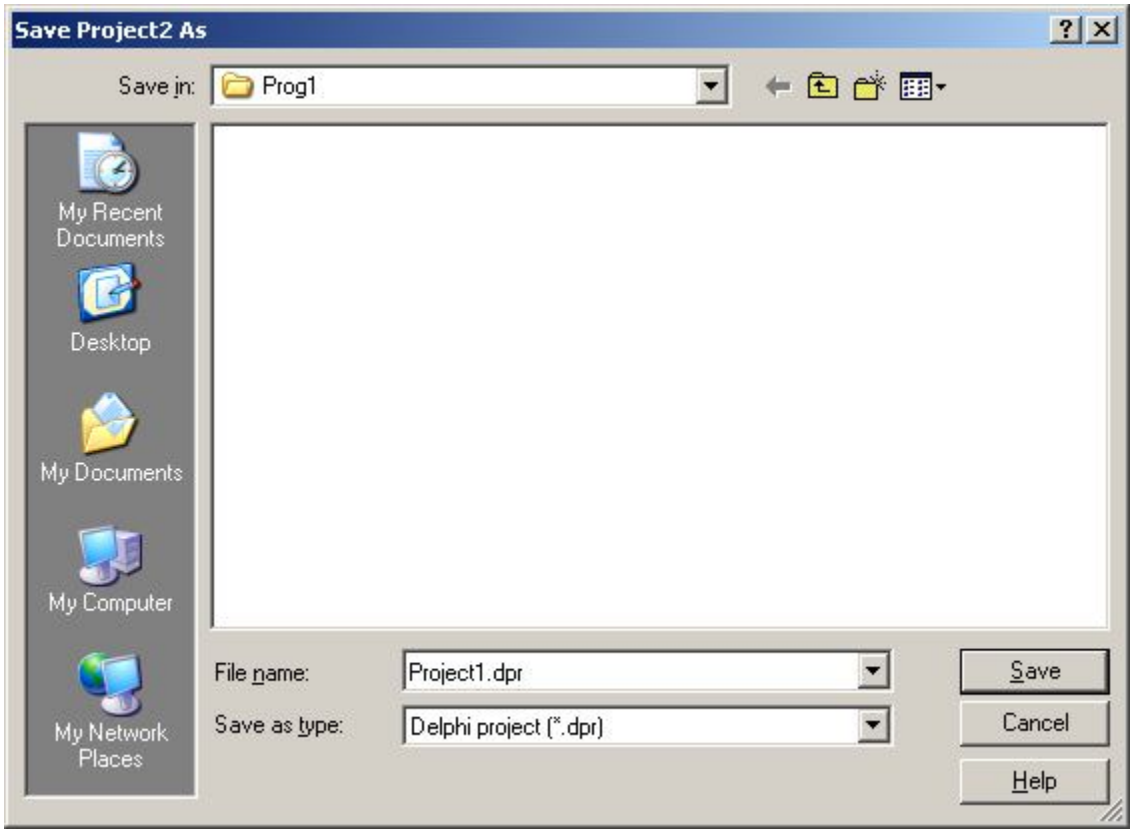
Tabel 4.1 Contoh pemberian nama unit dan form

Nama Unit	Nama Form	Kegunaan Unit
UCalculator	FCalculator	Form yang berisi kalkulator
ULapBulanan	FLapBulanan	Form yang berisi pembuatan laporan bulanan
UEditData	FEditData	Form yang berguna untuk mengedit data

Menyimpan File Project

Setelah unit disimpan, langkah selanjutnya adalah menyimpan file project. File project akan mempunyai ekstensi .dpr (Delphi project). File inilah yang akan dikompilasi menjadi suatu file executable (file exe).

Untuk menyimpan file project caranya adalah dengan mengklik menu File → Save Project As. Tampilan dialog penyimpanan file project dapat dilihat pada gambar 4.3.



Gambar 4.3 Tampilan diaog penyimpanan fle project

MENGEKSEKUSI APLIKASI

Untuk mengeksekusi aplikasi, cara yang dilakukan adalah dengan mengkompilasi project yang sedang dibuat dan kemudian menjalankannya.

Untuk menjalankan project langkah yang dilakukan adalah mengklik tombol Run → Run atau mengklik icon Run ( ) atau dengan menekan tombol F9. Jika tidak ada kesalahan dalam program, maka Delphi akan mengeksekusinya. Lihat contoh di di bawah ini pada gambar 4.4.

Gambar 4.4 Contoh eksekusi aplikasi

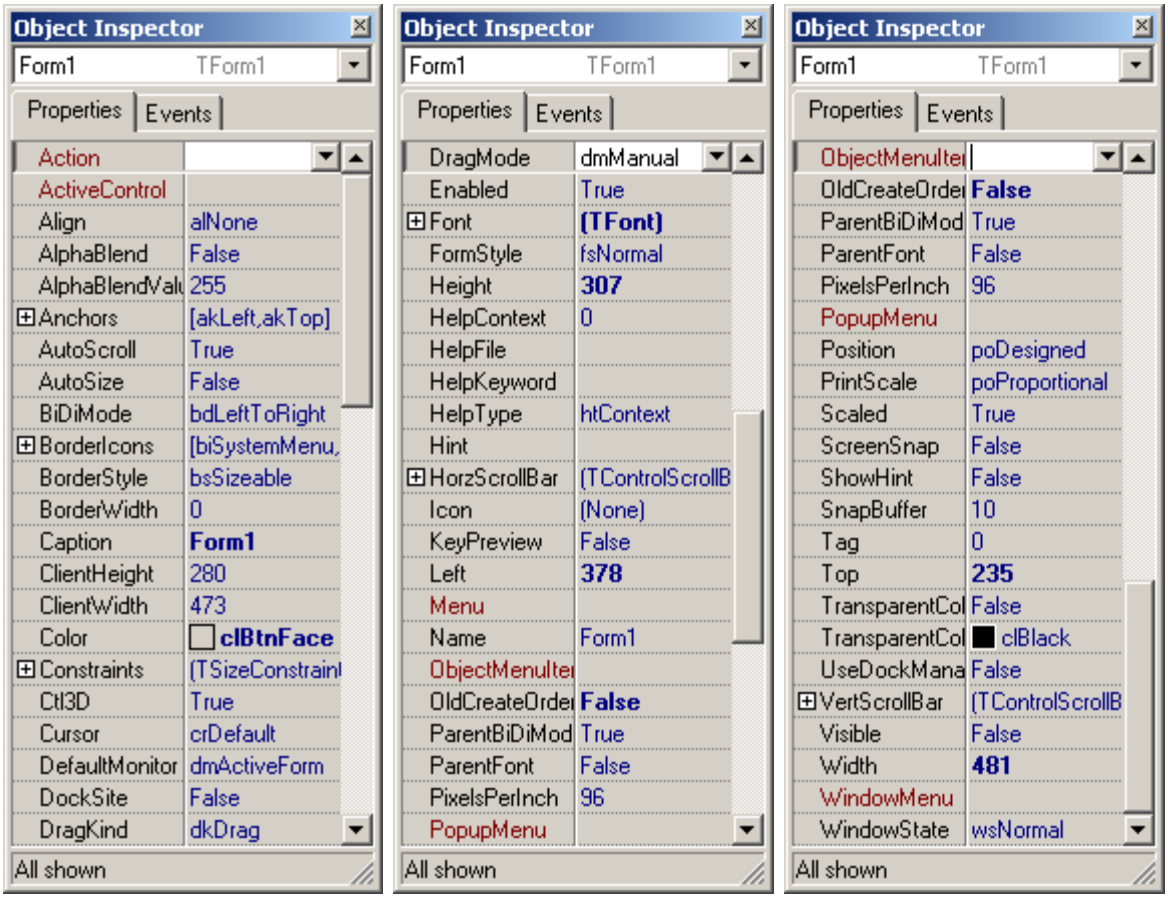
Dapat dilihat walaupun aplikasi yang dibuat masih kosong, tetapi sudah mempunyai fungsi window yang lengkap seperti dapat diresize, dipindahkan, diminimize, dimaximize dan tersedia pula tombol close. Pada bab-bab selanjutnya kita akan menempatkan komponen-komponen untuk melengkapi form tersebut.

Untuk kembali ke lingkungan IDE Delphi, klik tombol close ( ).

MEMODIFIKASI FORM

Aplikasi yang buat sebelumnya masih merupakan aplikasi yang paling sederhana yang bisa dibuat dengan Delphi. Langkah selanjutnya adalah coba memodifikasi form. Form memiliki banyak properti dan event. Properti mendefinisikan atribut dari suatu object, sedangkan event adalah kejadian apa yang dapat terjadi pada suatu object.

Untuk melihat properti dari suatu object, klik object tersebut, kemudian lihat di bagian Object Inspector. Jika anda melihat properti dari object Form, maka yang anda dapatkan adalah seperti pada gambar 4.5 di bawah ini.



Gambar 4.5 Properti dari object Form

Dari seluruh properti milik objek Form, ada beberapa properti yang sering dimodifikasi. Properti dari Form yang banyak dimodifikasi dapat dilihat pada tabel 4.2.

Tabel 4.2 Properti dari Form yang sering dimodifikasi

Properti	Kegunaan
BorderIcons	Menentukan tombol apa yang muncul pada kotak judul form (maximize, minimize, close, dan help).
BorderStyle	Menentukan tampilan dan perilaku border (sisi) form.
Caption	Menentukan judul dari form yang akan dilihat pada kotak judul bagian atas form.
Color	Menentukan warna latar dari form
Font	Menentukan font standar yang akan dipakai dalam form ini. Form ini akan dipakai oleh komponen-komponen yang dipasang dalam form.
Height	Menentukan tinggi dari form. Jika properti ini diubah, maka tinggi form juga akan berubah.
Hint	Menentukan pesan apa yang akan muncul ketika form berada di atas form.
Icon	Menentukan icon apa yang muncul di atas form.
Left	Menentukan lokasi koordinat kiri dari form.
Name	Menentukan nama dari object.
Position	Menentukan lokasi form.
Top	Menentukan lokasi koordinat atas dari form.

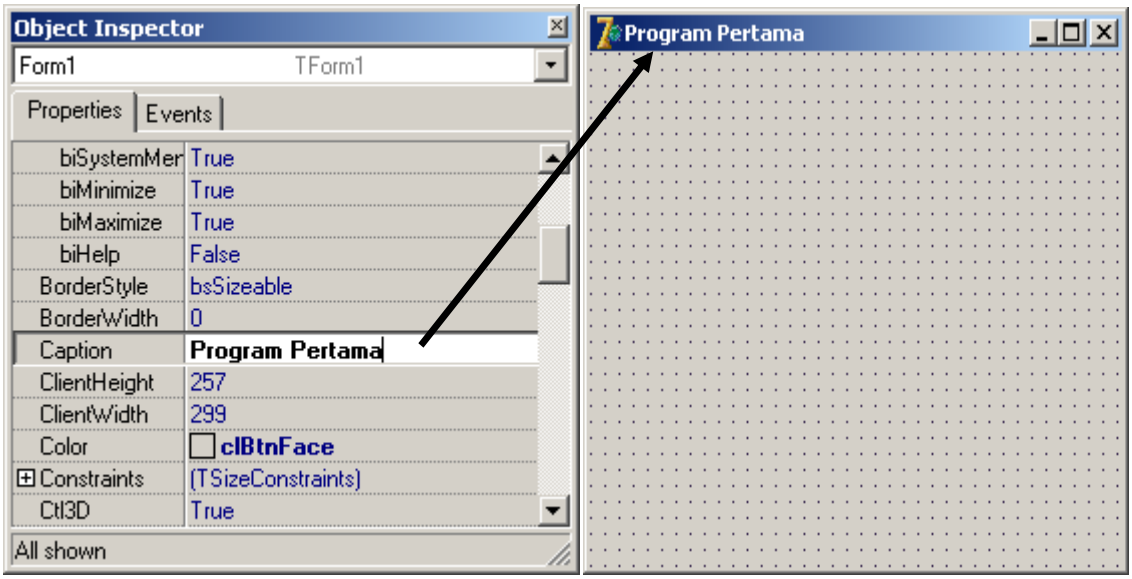
Width	Menentukan lebar dari form.
WindowState	Menentukan status windows dari form ketika dijalankan. (Normal, Minimize, Maximize).

MEMODIFIKASI FORM DENGAN OBJECT INSPECTOR

Cara memodifikasi form ketika mendesain form adalah dengan menggunakan object inspector.

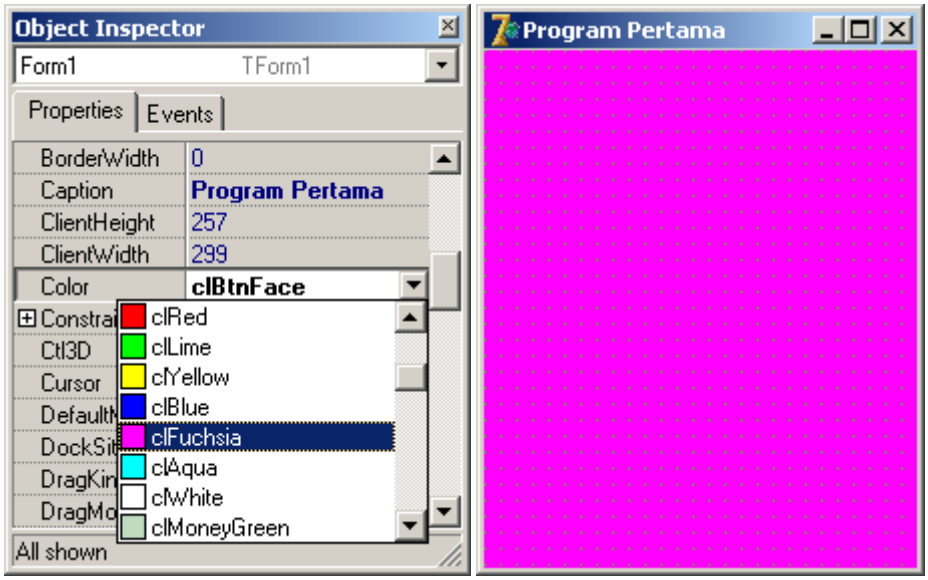
Contoh-contoh mengubah properti dari Form

1. Mengganti judul (Caption)



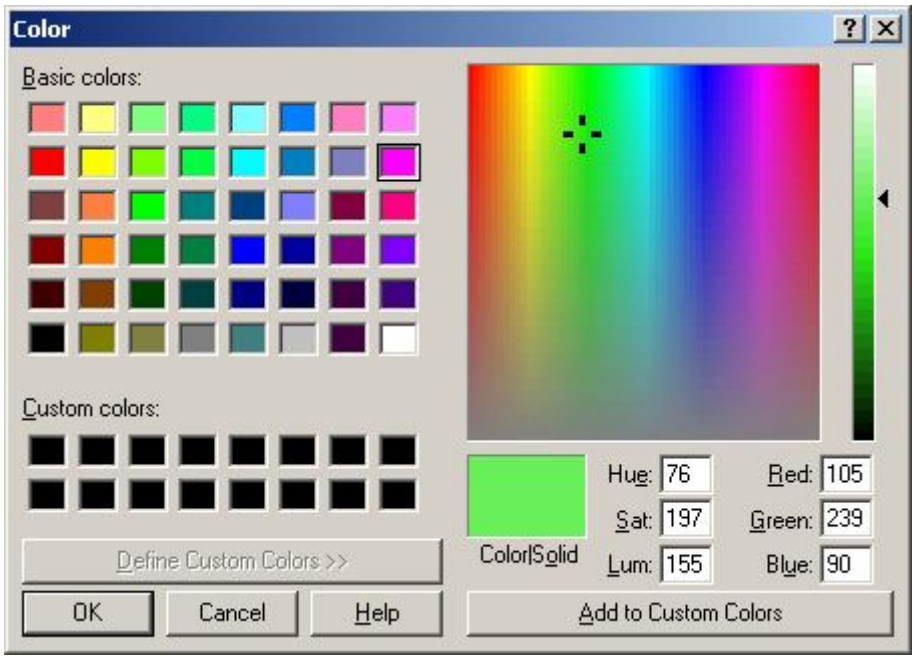
Gambar 4.6 Mengubah properti Caption

2. Mengganti Warna



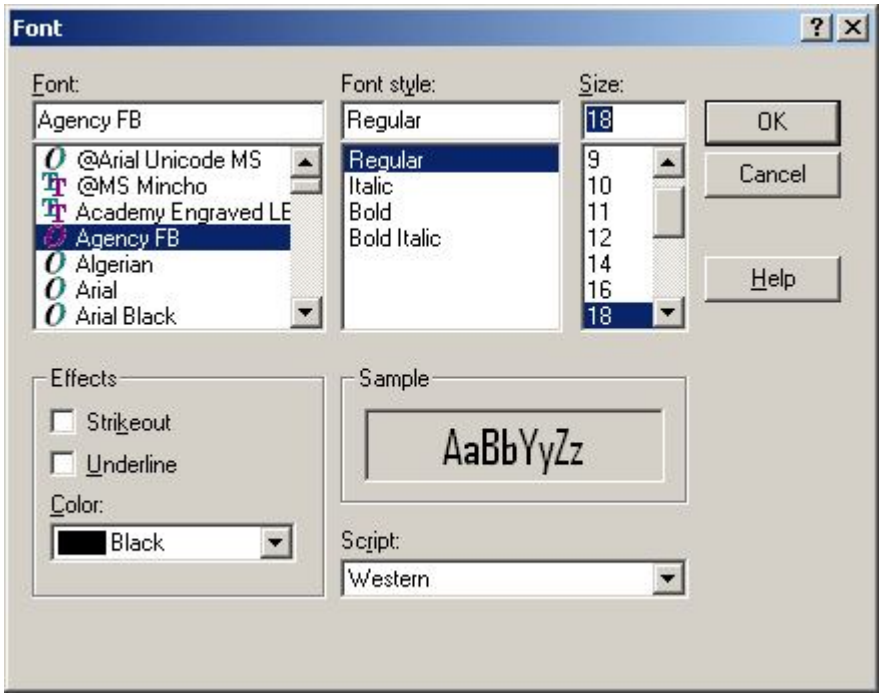
Gambar 4.7 Mengubah properti Color

Untuk memberikan warna khusus yang tidak tersedia pilihan warna adalah dengan mendoubleklik pada properti Color. Lihat gambar 4.8.



Gambar 4.8 Pengaturan warna khusus

3. Mengganti properti Font



Gambar 4.9 Pengaturan Font

Pengaturan font mungkin tidak terlihat perubahannya di form. Font ini adalah font standar yang akan digunakan oleh objek-objek yang ada dalam form tersebut.

MEMODIFIKASI FORM DENGAN KODE PROGRAM

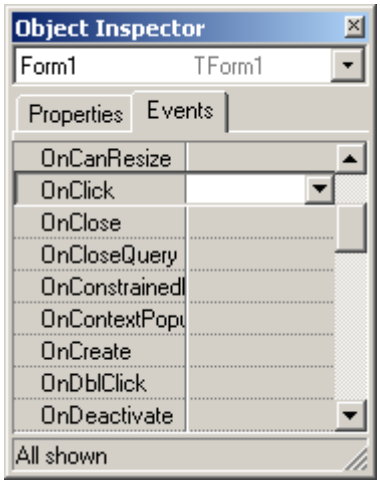
Memodifikasi properti suatu object tidak selalu menggunakan object inspector. Sangat dimungkinkan bagi kita untuk mengedit suatu properti dari suatu object untuk dimodifikasi ketika program sedang dijalankan. Oleh karena itu maka modifikasi yang dilakukan harus dengan menggunakan kode program delphi. Untuk melakukan hal tersebut, maka kita harus memanfaatkan event yang dimiliki oleh object tersebut.

Contoh misalnya : "kita ingin membuat form diubah warnanya menjadi merah ketika form diklik ". Dari pernyataan di atas dapat diambil beberapa hal yang harus diperhatikan :

- Kalau form diklik (Event OnClick digunakan)
- Warna form akan menjadi merah (Properti Color diisi merah)

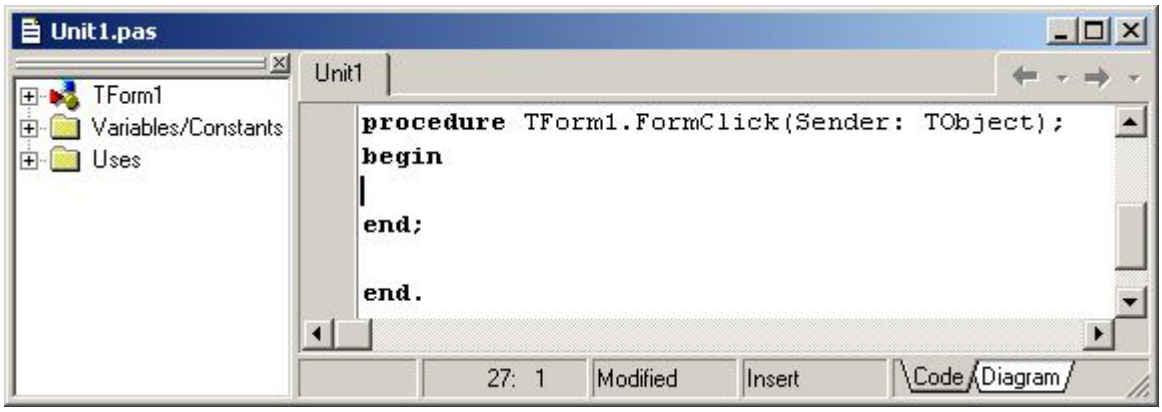
Langkah yang dilakukan adalah :

1. Klik di form untuk mengaktifkan form di object inspector
2. Klik tab Event



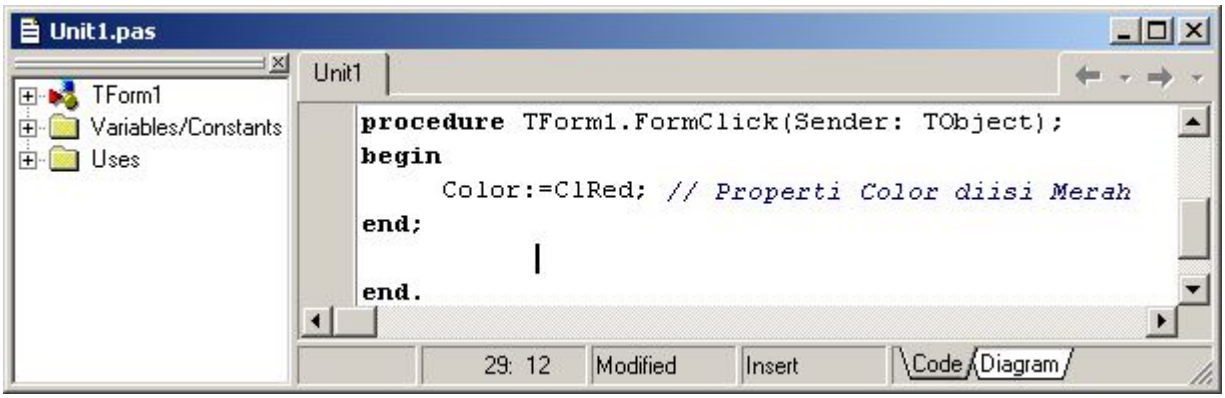
Gambar 4.10 Pemilihan Event di Object Inspector

3. Double klik di OnClick atau dengan mengisi nama metode di bagian OnClick kemudian tekan Enter. Dengan melakukan hal tersebut, maka akan muncul layar Code Editor.



Gambar 4.11 Mengisi kode program di code editor

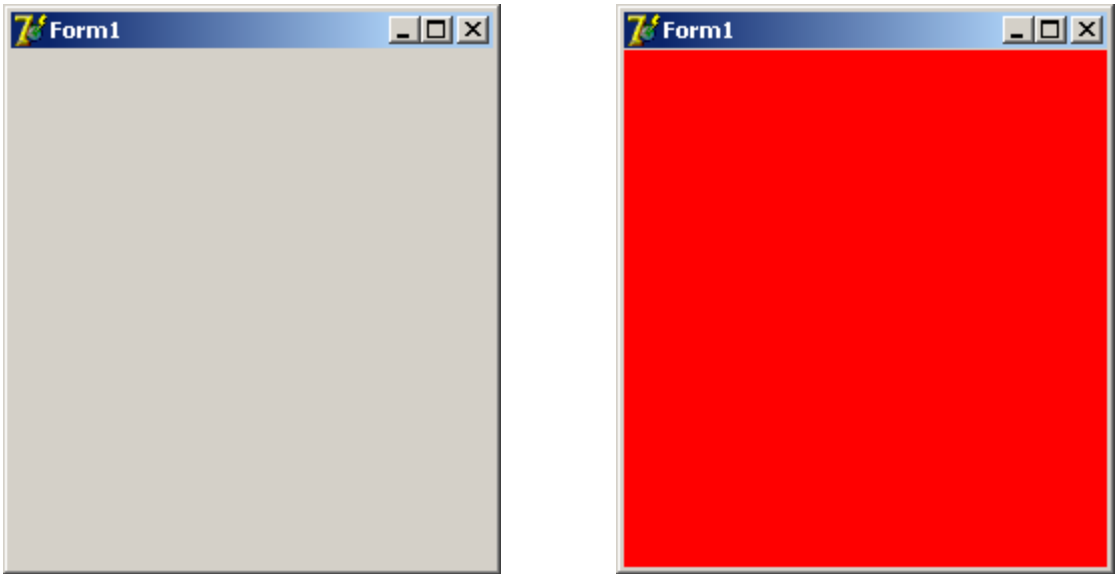
4. Isi kode program yang akan mengubah properti Color menjadi merah dengan perintah berikut.



Gambar 4.12 Mengisi kode program

Pada kode sumber program di atas berarti bahwa properti milik Form yaitu Color diisi (di Delphi penugasan/assignment menggunakan := bukan =) dengan warna ClRed (Konstanta warna Red).

5. Setelah kode program diisi, jalankan program dengan menggunakan tombol Run atau F9. Jika tidak ada kesalahan, maka akan menampilkan tampilan seperti gambar 4.13.



Sebelum diklik Setelah diklik

Gambar 4.13 Contoh Run program sebelum dan setelah diklik

Latihan :

Modifikasi program tersebut sehingga **ketika form didoubleklik, maka judul dari form akan menjadi "Program Pertamaku dengan Delphi"**. Event yang digunakan adalah OnDblClick.

Kode yang digunakan untuk hal tersebut adalah :

```
procedure TForm1.FormDblClick(Sender: TObject);
begin
     Caption:='Program Pertamaku dengan Delphi';
end;
```

MENAMBAHKAN OBJEK LAIN KE DALAM FORM

Aplikasi yang dibangun tentunya bukan hanya form saja. Tentu banyak komponen yang dapat digunakan dalam form tersebut. Komponen-komponen yang dapat digunakan ada dalam komponen palette. Salah satu tab yang ada dalam component palette adalah tab Standard. Dalam tab Standard berisi komponen-komponen yang sangat umum ada dalam suatu aplikasi. Komponen yang ada dalam tab standar ada dalam Tabel 4.3.

Tabel 4.3 Komponen di tab Standard

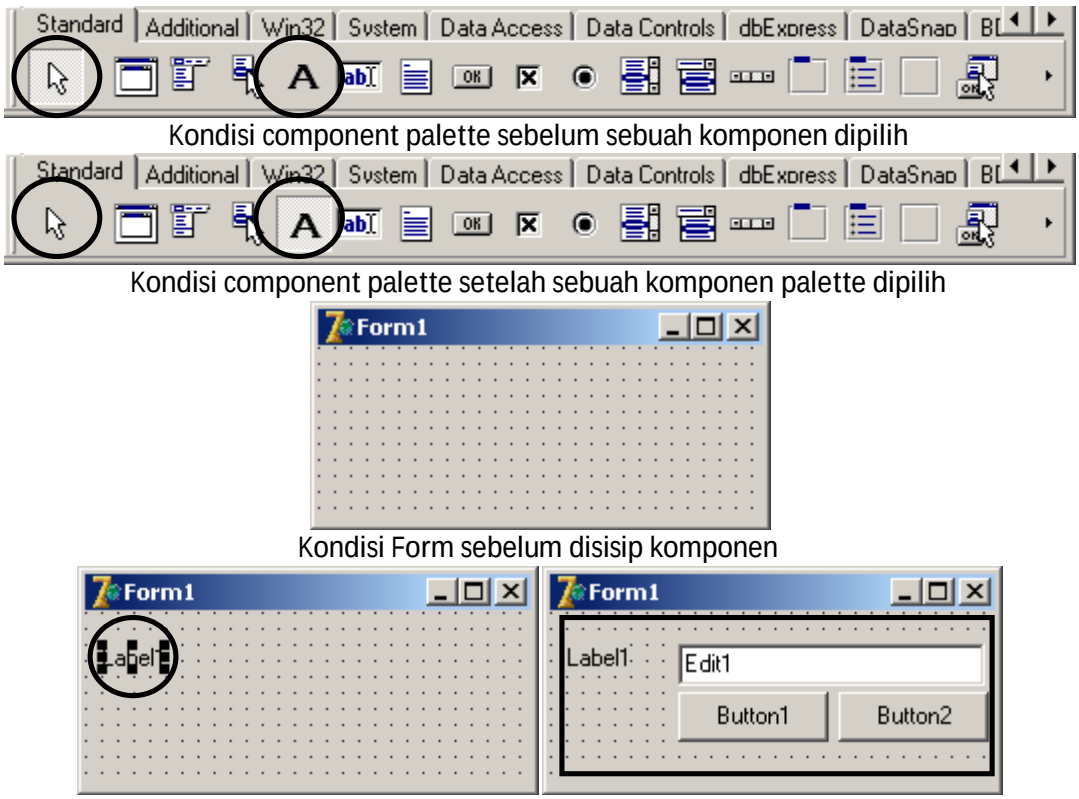
Nama Komponen	Icon	Kegunaan
Frame		Membuka sebuah kotak dialog yang akan menampilkan daftar frame yang ada dalam project yang sedang dibuka.
MainMenu		Membuat kotak menu pada suatu form
PopupMenu		Membuat menu popup yang akan muncul ketika pengguna mengklik kanan mouse
Label		Menampilkan text yang tidak bisa diedit oleh user. Jadi teksnya bersifat statis.
Edit		Menampilkan sebuah wilayah pengeditan dimanauser dapat memasukan sebaris text.
Memo		Displays an editing area where the user can enter or modify multiple lines of text.
Button		Membuat tombol yang dapat digunakan oleh pengguna untuk melakukan suatu aksi/proses.
CheckBox		Menampilkan sebuah pilihan dimana user dapat melakukan perubahan antara Ya/Tidak atau True/False. Checkbox digunakan untuk menampilkan sekumpulan pilihan dimana pengguna boleh memilih lebih dari satu pilihan.
RadioButton		Menampilkan sebuah pilihan dimana user dapat mengganti nilainya dengan True atau False. Radiobutton digunakan untuk menampilkan sekumpulan pilihan dimana user hanya boleh memilih satu pilihan saja.
ListBox		Menampilkan sebuah daftar pilihan yang dapat digulung.
ComboBox		Menampilkan daftar pilihan yang ditampilkan berbentuk kombinasi antara Listbox dan Edit. Pengguna dapat mengisi data dalam kotak Edit atau memilih sesuai dengan daftar yang ada.
ScrollBar		Menyediakan cara untuk mengubah bagian yang akan ditampilkan pada suatu list atau form.
GroupBox		Menyediakan sebuah kontainer (wilayah yang dapat diisi dengan komponen lain) untuk mengelompokkan pilihan-pilihan yang berelasi pada sebuah form.
RadioGroup		Membuat sebuah groupbox yang berisi sekumpulan radio button.
Panel		Membuat panel-panel yang dapat diisi dengan komponen lain.
ActionList		Membuat sekumpulan aksi-aksi yang memusatkan respon aplikasi anda terhadap aksi user.

Penjelasan lebih lanjut dari komponen-komponen tersebut akan dijelaskan pada bab-bab berikutnya, sejalan dengan perkuliahan.

Untuk menempatkan sebuah komponen ke dalam sebuah form caranya sangat sederhana yaitu dengan mengklik icon komponen yang ingin ditambahkan ke form diikuti dengan mengklik posisi komponen tersebut akan disimpan dalam form.

Contoh : "Kita diminta dibuat sebuah program yang akan menginputkan judul untuk form pada sebuah edit dan jika tombol **Ganti Judul** diklik maka judul tersebut akan terupdate di judul Form, dan jika tombol **Keluar** diklik maka program akan ditutup".

Perhatikan gambar 4.14 yang akan memperlihatkan proses penempatan komponen pada sebuah form.



Kondisi component palette sebelum sebuah komponen dipilih

Kondisi component palette setelah sebuah komponen palette dipilih

Kondisi Form sebelum disisip komponen

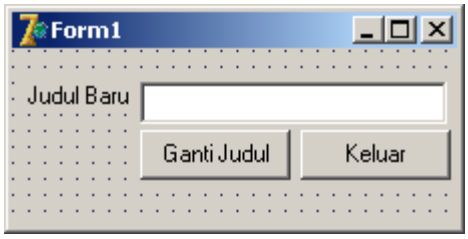
Kondisi form setelah disisip beberapa komponen

Gambar 4.14 Urutan penempatan komponen ke form

Setelah komponen ditempatkan, aturlah properti dari tiap komponen tadi, misalnya :

1. Objek **Label1**
   - Ganti properti **Caption** dengan **Judul Baru**
   - Silahkan atur **Font**, warna dan yang lainnya, dipersilahkan.
2. Object **Edit1**
   - Ganti properti **Text** dengan string kosong (dikosongkan)
   - Ganti properti **Name** dengan **EJudul** (Editbox Judul)
   - Silahkan atur **Font**, warna dan yang lainnya.
3. Object **Button1**
   - Ganti properti **Caption** dengan **Ganti Judul**
   - Ganti properti **Name** dengan **TblGantiJudul** (Tombol Ganti Judul)

4. Object **Button2**
   - Ganti properti **Caption** dengan **Keluar**
   - Ganti properti **Name** dengan **TblKeluar** (Tombol Keluar)
5. Langkah 1 sampai 4 harus menampilkan form seperti pada gambar 4.15



Gambar 4.15 Form contoh

6. Isi method pada event OnClick milik TblGantiJudul dengan kode berikut :

```
procedure TForm1.TblGantiJudulClick(Sender: TObject);
begin
            Caption:=EJudul.Text; // isikan isi Ejudul (text) ke judul (Caption) form
end;
```

7. Isi method pada event OnClick milik TblKeluar dengan kode berikut :

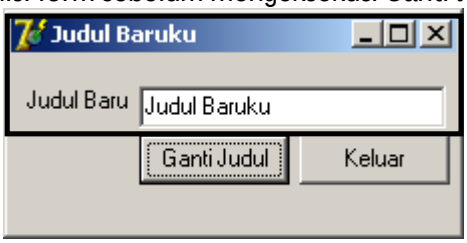
```
procedure TForm1.TblKeluarClick(Sender: TObject);
begin
      Close; // Tutup form
end;
```

Atau

```
procedure TForm1.TblKeluarClick(Sender: TObject);
begin
      Application.Terminate; // Tutup Aplikasi
end;
```

8. Run program dengan menekan tombol F9

Kondisi form sebelum mengeksekusi Ganti Judul



Kondisi form setelah mengisi judul baru dan menekan tombol Ganti Judul

Gambar 4. 16 Hasil run program

	Pada object Edit1, Button1, Button2 anda diminta untuk mengganti Name dari sebuah object. Ini **sangat diperlukan** karena ketika object yang diada dalam sebuah form hanya menggunakan penomoran, maka akan sulit untuk mengingat fungsi dari sebuah object. Oleh karena itu gantilah Name dari sebuah object sesuai dengan kegunaannya.

BAB V
BEKERJA DENGAN DATA

TIPE DATA

Tipe data pada dasarnya merupakan nama untuk sejenis data. Ketika kita mendeklarasikan sebuah variabel, maka variabel tersebut harus mempunyai sebuah tipe data, yang akan menentukan nilai-nilai yang dapat dimuat dan dioperasikan pada variabel tersebut. Ada beberapa tipe data yang telah disediakan oleh Delphi.

TIPE DATA UNTUK BILANGAN

Tipe data untuk bilangan bulat merepresentasikan sebagian dari seluruh bilangan yang ada. Tipe data bilangan terdiri dari 2 bagian yaitu tipe data bilangan bulat dan bilangan pecahan.

Bilangan Bulat

Tipe data bilangan bulat adalah tipe data yang dapat digunakan untuk variabel yang akan menyimpan data bilangan bulat. Tabel 5.1 berisi tipe data yang berguna untuk data berbentuk bilangan.

Tabel 5.1 Tipe Data Bilangan Bulat

Tipe	Memori (dalam byte)	Jangkauan Nilai
Byte	1	0 .. 255
Word	2	0 .. 65535
ShortInt	1	-128 .. 127
SmallInt	2	-32768 .. 32767
Integer	4	-2147483648 .. 2147483647
Cardinal	4	0 .. 4294967295
LongWord	4	0 .. 4294967295
LongInt	4	-2147483648 .. 2147483647
Int64	8	$-2^{63}..2^{63}-1$

Bilangan Pecahan (Real)

Sebuah tipe data real mendefinisikan sekumpulan bilangan yang dapat direpresentasikan dalam notasi pecahan (floating-point). Tabel 5.2 berisi tipe data yang dapat digunakan untuk varibel yang menampung bilangan nyata (real).

Tabel 5.2 Tipe Data Bilangan Pecahan

Tipe	Memori (dalam byte)	Jangkauan Nilai	Digit Signifikan
Real48	6	2.9 x 10^-39 .. 1.7 x 10^38	11 – 12
Single	4	1.5 x 10^-45 .. 3.4 x 10^38	7 – 8
Double	8	5.0 x 10^-324 .. 1.7 x 10^308	15 – 16
Extended	10	3.6 x 10^-4951 .. 1.1 x 10^4932	19 – 20
Real *	8	5.0 x 10^-324 .. 1.7 x 10^308	15 – 16
Currency	8	-922337203685477.5808 .. 922337203685477.5807	19 – 20

* Paling banyak digunakan

TIPE DATA UNTUK TEKS

Tipe data untuk teks berguna untuk menyimpan data karakter yang bisa alphabet, numerik, tanda baca, atau huruf lainnya. Ada dua jenis data yang digunakan untuk menampung data teks yaitu karakter (hanya menampung sebuah karakter) dan String (menampung banyak karakter).

Karakter

Tipe data ini hanya dapat menampung sebuah karakter saja. Tipe data karakter yang dapat digunakan dapat dilihat pada tabel 5.3.

Tabel 5.3 Tipe Data Karakter

Tipe	Memori (dalam byte)	Karakter yang dapat disimpan
ANSIChar	1	1 karakter ANSI
WideChar	2	1 karakter Unicode
Char *	1	1 Karakter ASCI

* Paling banyak digunakan

String

String adalah tipe data yang dapat digunakan untuk menyimpan sekumpulan karakter (1 atau lebih karakter). Tabel 5.4 memperlihatkan tipe data string yang dapat digunakan.

Tabel 5.3 Tipe Data String

Tipe	Panjang Maksimum (karakter)	Memori yang digunakan
ShortString	255	2 .. 256 bytes
AnsiString	2^{31}	4 byte .. 2GB
WideString	2^{30}	4 byte .. 2GB
String *	Bisa berperan sebagai ShortString atau AnsiString	

* Paling banyak digunakan. Standarnya mengacu AnsiString

TIPE DATA BOOLEAN

Tipe data boolean digunakan untuk menyimpan nilai logika (benar/salah, true/false). Tipe data boolean yang dapat digunakan, dapat dilihat pada tabel 5.4.

Tabel 5.4 Tipe Data Boolean

Tipe	Memori (byte)
Boolean *	1
ByteBool	1
Bool	2
WordBool	2
LongBool	4

* Paling banyak digunakan

VARIABEL

Variabel adalah sebuah pengenal (identifier) yang nilainya dapat berubah ketika program dijalankan. Sebuah variabel juga berarti sebuah nama untuk sebuah lokasi dalam memori. Anda dapat menggunakan nama tersebut untuk membaca dan menulis ke suatu lokasi memori. Variabel-variabel berperan sebagai penampung data dan karena setiap variabel mempunyai tipe data, maka kompiler (Delphi) akan mengerti bagaimana menginterpretasikan data yang ditampung variabel tersebut.

DEKLARASI VARIABEL

Sebelum suatu variabel dapat digunakan, variabel harus dapat dideklarasikan terlebih dahulu. Pengertian deklarasi di sini adalah menyebutkan nama variabel dan juga tipe datanya. Variabel dideklarasikan pada bagian yang diawali dengan kata **var**.

Contoh pendeklarasian variabel

```
var
        Pajak:Real;
        JenisKelamin:Char;
        Nama:String;
        JumlahAnak:Byte;
        Gaji:Currency;
        PunyaAnak:Boolean;
```

MENGISI NILAI KE VARIABEL

Setiap variabel dapat menampung data sesuai dengan jangkauan nilainya. Cara pengisian data untuk tiap jenis tipe data juga berbeda-beda. Ada beberapa aturan yang harus diperhatikan ketika pengisian data ke variabel, diantaranya :

- Selalu menggunakan operator penugasan (:=)
- Jika tipe data variabel berupa bilangan, maka bilangan tersebut ditulis secara langsung. Jika pecahan, gunakan tanda baca titik (.) sebagai pemisah pecahannya.

- Jika tipe data berupa teks, maka harus dimulai dengan tanda baca apostrop (petik satu/ ')
- Variabel bertipe data bilangan bulat tidak bisa menerima tipe data bilangan pecahan. Solusinya adalah dengan membulatkan bilangan pecahan tersebut. Tetapi tipe data bilangan pecahan bisa diisi dengan bilangan bulat.
- Variabel bertipe data bilangan tidak bisa menerima tipe data teks walaupun isi teksnya berupa bilangan. Solusinya adalah dengan mengkonversi data teks tersebut ke tipe data yang sesuai dengan variabelnya (lihat sub bab Fungsi-Fungsi Konversi Data).

Contoh pengisian variabel dapat dilihat dalam potongan program di bawah ini.

```
Pajak:=0.1; // 10%
JenisKelamin:='L';
Nama:='Susilawati';
JumlahAnak:=3;
Gaji:=1500000;
if JumlahAnak > 0) then  // Sama dengan perintah → PunyaAnak:=(JumlahAnak > 0);
    PunyaAnak:=True
else
    PunyaAnak:=False;
```

KONSTANTA

Konstanta adalah suatu nilai yang tetap yang terdapat dalam program. Konstanta tidak dapat diganti nilainya. Jika ada perintah yang mengubah nilai konstanta, maka program tersebut tidak akan dapat dikompilasi dan akan menampilkan pesan error. Cara menuliskan konstanta pada berbagai tipe data dapat dilihat di bawah ini.

Contoh pendeklarasian konstanta dapat dilihat dalam potongan program di bawah ini.

```
const
    Judul='Program dengan Delphi';
    Terkecil=0;
    Terbesar=100;
    Pi=3.17;
    Merah:Integer=clRed; // konstanta bertipe data
    Biru:TColor=clBlue;  // konstanta bertipe data
    Benar=true;
    Salah=false;
    Ya='Y';
    Enter=#13;
```

OPERATOR

Operator menyatakan operasi apa yang akan digunakan dalam suatu operasi. Ada beberapa jenis operator yang banyak dipakai dalam Delphi yaitu operator aritmatika, operator boolean, operator logika, operator relasional dan operator string.

OPERATOR ARITMETIKA

Operator digunakan untuk mengolah data-data bertipe bilangan (bilangan bulat dan pecahan). Ada beberapa operator yang dapat digunakan, dapat dilihat di tabel 5.5.

Tabel 5.5 Operator-Operator Aritmetika

Operator	Operasi	Tipe Operand	Tipe Hasil
Operator Aritmetika Binary (Membutuhkan 2 operand)			
+	Penambahan	integer, real	integer, real
-	Pengurangan	integer, real	integer, real
/	Pembagian	integer, real	real
*****	Perkalian	integer, real	integer, real
div	Pembagian Bulat	integer	integer
mod	Sisa Pembagian	integer	Integer
Operator Aritmetika Unary (Membutuhkan 1 operand)			
+	Penanda positif		
-	Penanda negatif		

Ada aturan-aturan yang berlaku dalam pengoperasian operator aritmatika, yaitu :

- Nilai dari operasi X / Y akan bertipe Extended, tanpa mempedulikan tipe data dari X dan Y. Jika salah satu operand bertipe real maka hasil akan bertipe Extended. Jika salah satu dari operand bertipe Int64 maka hasil akan bertipe Int64 jika tidak maka hasil akan bertipe Integer. Jika sebuah operand bertipe bagian dari tipe bilangan bulat, maka akan diperlakukan sebagai tipe integer.
- Hasil operasi X div Y adalah nilai pembagian X/Y dibulatkan menuju arah nol ke integer terdekat.
- Akan terjadi kesalahan runtime ketika Y bernilai 0 dalam ekspresi X/Y, X div Y atau X mod Y.
- Jangan melakukan perhitungan yang melebihi jangkauan nilai. Jika suatu perhitungan melebihi nilai maksimal jangkauan nilainya maka nilai akan berputar ke nilai terkecil dari variabel tersebut. Jika suatu perhitungan menghasilkan nilai lebih kecil dari jangkauan terkecilnya, maka nilai akan berputar ke nilai terbesar dari variabel tersebut.

OPERATOR BOOLEAN

Operator-operator boolean digunakan untuk mengoperasikan variabel bertipe Boolean, dan akan menghasilkan hasil bertipe Boolean pula. Operator boolean dapat dilihat di tabel 5.6, dan aturan operasi boolean dapat dilihat pada tabel 5.7.

Tabel 5.5 Operator-Operator Aritmatika

Operator	Operasi
NOT	Negasi
AND	Operator DAN
OR	Operator ATAU
XOR	Operator ATAU Eksklusif

Tabel 5.7 Aturan Operasi Boolean

A	B	NOT A	A AND B	A OR B	A XOR B
TRUE	TRUE	FALSE	TRUE	TRUE	FALSE
TRUE	FALSE	FALSE	FALSE	TRUE	TRUE
FALSE	TRUE	TRUE	FALSE	TRUE	TRUE
FALSE	FALSE	TRUE	FALSE	FALSE	FALSE

OPERATOR LOGIKA (BIT)

Operator logika adalah operator yang digunakan untuk melakukan operasi terhadap bilangan bulat pada level bit. Ada enam operator yang tergolong sebagai operator logika. Operator-operator ini dapat dilihat pada tabel 5.8.

Tabel 5.8 Operator-Operator Logika

Operator	Operasi
NOT	Negasi
AND	Operator DAN
OR	Operator ATAU
XOR	Operator ATAU Eksklusif
SHL	Operator penggeseran bit ke arah kiri
SHR	Operator penggeseran bit ke arah kanan

Tabel 5.9 Aturan Operasi Logika NOT, AND, OR, dan XOR

A	B	NOT A	A AND B	A OR B	A XOR B
1	1	0	1	1	0
1	0	0	0	1	1
0	1	1	0	1	1
0	0	1	0	0	0

Tabel 5.10 Contoh Penggunaan Operasi Logika SHL dan SHR

A	B	BINER A	A SHL B	A SHR B
10	2	1010	101000 = 40	10 = 2
8	3	1000	1000000 = 64	1 = 1
5	1	101	1010 = 10	10 = 2
25	1	11001	110010 = 50	1100 = 12

Operator-operator logika banyak digunakan dalam banyak hal seperti pengontrolan perangkat keras, kompresi data, security data (steganografi), pengolahan gambar.

OPERATOR RELASIONAL

Operator relasional adalah operator yang digunakan untuk membandingkan dua buah nilai dan menghasilkan nilai berupa True (benar) atau False (salah). Operator-operator yang dapat digunakan dapat dilihat pada tabel 5.11 di bawah ini.

Tabel 5.11 Operator-Operator Relasional

Operator	Operasi
>	Lebih dari
<	Kurang dari
>=	Lebih besar atau sama dengan
<=	Lebih kecil atau sama dengan
<>	Tidak sama dengan
=	Sama dengan

OPERATOR STRING

Operator string adalah operator-operator yang dapat digunakan untuk mengoperasikan variabel bertipe data string. Semua operator relasional (>, <, >=, <=, <>, =) dapat digunakan untuk mengoperasikan string. Adapula operator + yang berguna untuk operasi konkatenasi (penggabungan suatu string dengan string lain sehingga menghasilkan string gabungan).

ATURAN PENGERJAAN OPERATOR

Sejumlah operator telah diperkelankan. Dalam prakteknya, berbagai macam operator dapat digabungkan ke dalam sebuah ekspresi. Urutan pengeksekusian operator dapat dilihat pada tabel 5.12.

Tabel 5.12 Urutan Operasi Operator

Operator	Prioritas
Not	Pertama (tertinggi)
***,/,div, mod, shl, shr**	Kedua
+, -, or, xor	Ketiga
=, <>, <, >, <=, >=,	Keempat (terendah)

FUNGSI-FUNGSI UNTUK KONVERSI DATA

Ada waktunya kita membutuhkan untuk mengisikan sebuah string yang berisi angka. Tetapi jika sebuah angka masih disimpan dalam variabel string, maka variabel tersebut tidak dapat dioperasikan sebagai angka. Solusi untuk kasus tersebut adalah dengan menggunakan fungsi-fungsi yang mengkonveriksan data menjadi nilai yang mempunyai tipe data lain. Tabel 5.13 berisi fungsi-fungsi yang dapat digunakan dalam mengkonversikan data.

Tabel 5.13 Fungsu-Fungsi Konversi Data

Nama Fungsi	Kegunaan
StrToInt	Mengkonversikan data bertipe String yang berisi angka menjadi data bertipe Integer
IntToStr	Mengkonversikan data bertipe Integer menjadi data bertipe String
StrToInt64	Mirip StrToInt, tetapi menghasilkan data bertipe Int64
Int64ToStr	Mirip IntToStr, tetapi data yang dikonveri harus bertipe Int64
StrToFloat	Mengkonversikan data bertipe String yang berisi angka menjadi data bertipe Extended (pecahan).
FloatToStr	Mengkonversikan data pecahan menjadi data bertipe String.
StrToBool	Mengkonversikan data bertipe String yang berisi nilai TRUE, FALSE, atau angka menjadi data bertipe boolean. StrToBool akan bernilai TRUE jika data bersisi nilai bukan 0 atau string 'TRUE'
BoolToStr	Mengkonversikan data bertipe Boolean menjadi String.
StrToCurr	Mengkonversi data bertipe String menjadi Currency
CurrToStr	Mengkonversi data bertipe Currency menjadi String

CONTOH PROGRAM MENGOLAH DATA

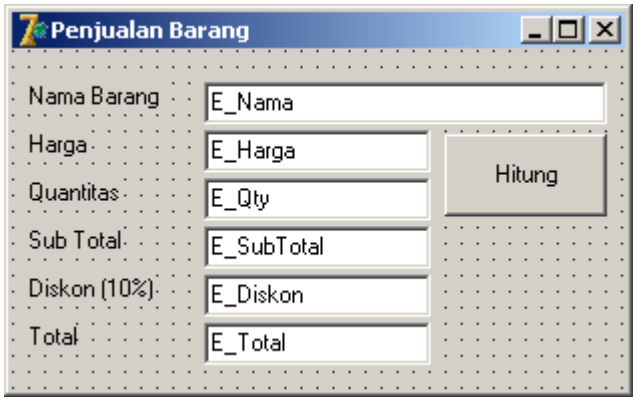
Untuk lebih memahami cara pengolahan data dengan menggunakan Delphi, perhatikan contoh kasus di bawah ini.

"Buat sebuah program yang akan melakukan perhitungan penjualan barang. Data yang diinputkan adalah **Nama Barang**, **Harga Barang**, dan **Quantitas** penjualan. Dari data tersebut akan didapatkan data **Sub Total**, **Diskon** dan **Total** dengan ketentuan **Sub Total** adalah **Harga Barang** * **Quantitas**, **Diskon** adalah **10 %** dari **Sub Total**, dan **Total** adalah **Sub Total** dikurangi **Diskon**. Perhitungan dilakukan ketika user menekan tombol **Hitung**".

Dari pernyataan di atas dapat disimpulkan beberapa hal, yaitu :

- Data yang diinputkan adalah Nama Barang, Harga Barang, dan Quantitas
- Data yang dikeluarkan adalah Sub Total, Diskon dan Total
- Perhitungan dilakukan ketika user menekan tombol Hitung
- Peraturan perhitungan adalah :
  + Sub Total = Harga Barang * Quantitas
  + Diskon = 10% * Sub Total
  + Total = Sub Total – Diskon

Dari kesimpulan di atas, maka didapatkan layout form seperti form di bawah ini.

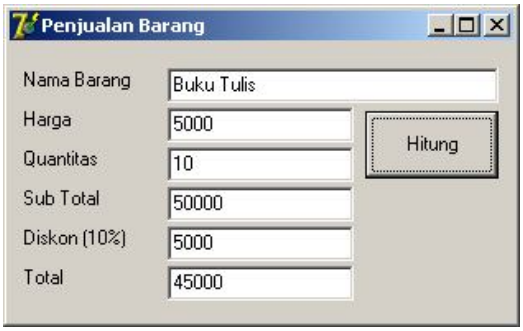


Gambar 5.1 Form Penjualan Barang

Adapun untuk method yang akan dieksekusi ketika tombol Hitung diklik adalah :

```
procedure TForm1.Tbl_HitungClick(Sender: TObject);
var
   Qty : Integer;
   Harga, SubTotal, Diskon, Total   : Currency;
begin
     Harga:=StrToCurr(E_Harga.Text);
     Qty:=StrToInt(E_Qty.Text);
     SubTotal:= Harga * Qty;
     E_SubTotal.Text:= CurrToStr(SubTotal);
     Diskon:= 0.1 * SubTotal;
     E_Diskon.Text:=CurrToStr(Diskon);
     Total:= SubTotal - Diskon;
     E_Total.Text:=CurrToStr(Total);
end;
```

Jika dijalankan, maka akan menghasilkan tampilan seperti di bawah ini.



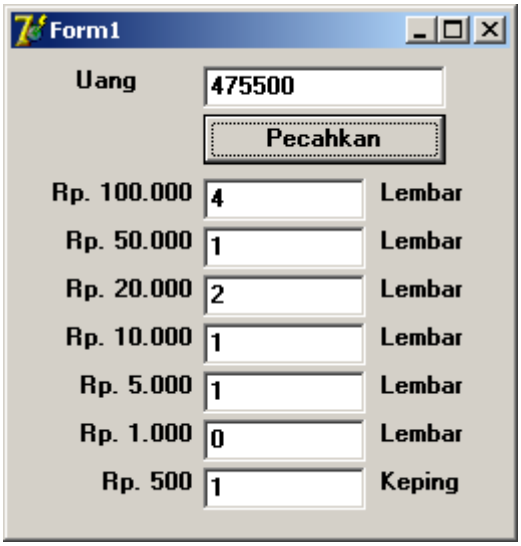
Gambar 5.2 Hasil eksekusi program penjualan barang

LATIHAN-LATIHAN

1. Buatlah program untuk menghitung Nilai Akhir suatu matakuliah yang diambil oleh seorang mahasiswa. Data yang diinputkan adalah Nilai Tugas, Persentase Absensi, Nilai UTS, dan Nilai UAS. Perhitungan dilakukan ketika user mengklik tombol Hitung NA. Aturan perhitungan untuk NilaiAkhir adalah 10% Absensi + 20% Tugas + 30% UTS + 40% UAS.
   Contoh Tampilan adalah :

2. Buatlah program yang dapat mendemokan operator aritmatika dan operator logika yang menginputkan dua buah data. Hasil yang ditampilkan harus sesuai dengan tombol yang diklik. Tombol yang disediakan adalah +, -, /, *, div, mod, shl, shr.
   Contoh tampilan adalah :

3. Buatlah program untuk memecahkan suatu bilangan ke dalam bentuk pecahan uang. Contoh dapat dilihat pada gambar di bawah ini.



4. Buatlah program yang berguna untuk menghitung gaji karyawan. Data yang diinputkan adalah Nama, Banyak Anak, dan Gaji Pokok. Jika tombol Hitung diklik, maka akan melakukan perhitungan yang menghasilkan data Tunjangan Istri, Tunjangan Anak, Total Tunjangan, Gaji Kotor, Pajak, dan Gaji Bersih. Adapun ketentuan perhitungannya adalah :
   - Tunjangan Istri = 20% Gaji Pokok
   - Tunjangan Anak = 5 % Gaji Pokok untuk setiap anak
   - Total Tunjangan = Tunjangan Anak + Tunjangan Istri
   - Gaji Kotor = Gaji Pokok + Total Tunjangan
   - Pajak = 10 % Gaji Kotor
   - Gaji Bersih adalah Gaji Kotor sesudah dikenai pajak

   Untuk design form, silahkan berimprovisasi.

5. Buatlah program untuk menghitung biaya pemakaian komputer di sebuah rental. Data yang diinputkan adalah data waktu masuk dan waktu keluar. Ketika tombol Hitung di klik, hitunglah berapa biaya pemakaian komputer jika biaya pemakaian per jam adalah Rp. 5000. Data yang dihasilkan adalah lama pakai dan biaya pakai.
   Tampilan form harap didesign sendiri.

BAB VI
PERCABANGAN

PENDAHULUAN

Percabangan digunakan untuk menentukan blok perintah mana yang akan dilakukan berdasarkan kondisi yang telah ditentukan. Jika kondisinya tercapai (bernilai true) maka pernyataan akan dikerjakan.

Ada dua jenis percabangan yang ada dalam Delphi, yaitu :

- Percabangan menggunakan pernyataan IF
- Percabangan menggunakan pernyataan CASE

PERNYATAAN IF

Pernyataan IF digunakan untuk memeriksa sebuah kondisi dan kemudian mengeksekusi bagian source code tertentu berdasarkan kondisi Benar/True atau Salah/False. Kondisi harus dibentuk dalam ekspresi Boolean.

Ada dua jenis pernyataan IF yaitu

1. Pernyataan IF ... THEN.
   Pernyataan ini hanya memeriksa apakah suatu blok kode program dapat dieksekusi atau tidak. Jika kondisi pernyataan ini bernilai True maka blok program yang ada di bawahnya akan dieksekusi. Tetapi jika kondisi pernyataan bernilai False maka alur program akan menganggap pernyataan IF telah selesai karena tidak mempunyai alternative lain.
   Bentuk dasar pernyataan IF ... THEN adalah sebagai berikut :

```
IF kondisi Boolean THEN
   pernyataan;
```

   Jika pernyataan yang akan dieksekusi ketika kondisi bernilai true lebih dari satu pernyataan, maka pernyataan-pernyataan tersebut harus diapit dengan begin dan end, sehingga bentuk dasar pernyataan IF .. THEN nya adalah sebagai berikut :

```
IF kondisi boolean THEN
Begin
     Pernyataan;
     Pernyataan;
     …
End;
```

2. Pernyataan IF ... THEN ... ELSE. Pernyataan ini hanya memeriksa apakah suatu blok kode program dapat dieksekusi atau tidak. Jika kondisi pernyataan ini bernilai True maka blok program yang ada di bawahnya akan dieksekusi. Tetapi jika kondisi pernyataan bernilai False maka alur program akan mengeksekusi pernyataan yang ada di bawah pernyataan ELSE.

Bentuk dasar pernyataan IF ... THEN ... ELSE adalah sebagai berikut :

```
IF kondisi boolean THEN
    pernyataan1
ELSE
    pernyataan2;
```

Jika pernyataan yang akan dieksekusi ketika kondisi bernilai true lebih dari satu pernyataan, maka pernyataan-pernyataan tersebut harus diapit dengan begin dan end, sehingga bentuk dasar pernyataan IF .. THEN nya adalah sebagai berikut :

```
IF kondisi boolean THEN
begin
    pernyataan1;
end
ELSE
begin
    pernyataan2;
end;
```

Contoh pernyataan IF ... THEN :

```
If Nilai>=50 then
    ShowMessage('Selamat Anda Lulus');
```

atau

```
If Nilai>=50 then
begin
     ShowMessage('Selamat Anda Lulus');
end;
```

Contoh pernyataan IF ... THEN ... ELSE :

```
If Nilai>=50 then
    ShowMessage('Selamat Anda Lulus')
Else
    ShowMessage('Anda Tidak Lulus');
```

atau

```
If Nilai>=50 then
begin
    ShowMessage('Selamat Anda Lulus');
end
Else
begin
    ShowMessage('Anda Tidak Lulus');
end;
```

Contoh pernyataan IF ... THEN ... ELSE ... dengan kondisi lebih dari 2 kondisi

```
If Nilai>=80 then
begin
    ShowMessage('Selamat Anda Lulus Dengan Nilai yang sangat baik');
end
Else
If Nilai>=50 then
begin
    ShowMessage('Selamat Anda Lulus Dengan Nilai yang cukup baik');
end
Else
begin
    ShowMessage('Anda Tidak Lulus');
end;
```

Ada beberapa hal yang harus diperhatikan dalam penggunaan pernyataan IF, diantaranya adalah :

1. Kondisi yang diperiksa harus menyatakan sebuah kondisi Boolean (True/False). Sehingga dalam bagian kondisi harus menggunakan operator relasional (=, >, <, >=, <=, <>) dan Operator Boolean (And, Or, Xor, dan NOT).
2. Pernyataan IF dianggap sebagai sebuah pernyataan saja. Oleh karena itu tidak boleh ada tanda semicolon (;) sebelum ELSE, karena ELSE adalah bagian dari pernyataan IF. Jika ada tanda semicolon sebelum ELSE, berarti pernyataan IF telah selesai dan bagian ELSE tersebut bukan lagi milik pernyataan IF yang di atasnya. Dan perintah ELSE tanpa didahului dengan pernyataan IF akan mengakibatkan kesalahan.
   Contoh :

```
If Nilai>=50 then
    ShowMessage('Selamat Anda Lulus'); // Pasti Error karena IF telah selesai di baris ini
Else // Pasti Error, karena ELSE baris ini tidak mempunyai pernyataan IF
    ShowMessage('Anda Tidak Lulus');
```

PERNYATAAN CASE

Pernyataan CASE, digunakan sebagai pengganti pernyataan IF. Pernyataan CASE akan memberikan alternative yang lebih gampang dibaca untuk menggantikan pernyataan IF. Tetapi tidak semua pernyataan IF bisa diganti dengan pernyataan CASE.

Bentuk umum dari CASE adalah sebagai berikut :

```
CASE EkspresiNilai Of
   DaftarKemungkinan1:Pernyataan1;
   DaftarKemungkinan2:Pernyataan2;
  ...
   DaftarKemungkinanN:PernyataanN;
   Else
       PernyataanLainnya;
End;
```

Jenis data yang boleh ada di bagian EkspresiNilai haruslah data bertipe ordinal (Bilangan Bulat, Karakter, Boolean). Tipe data string atau real tidak bisa digunakan dalam EkspresiNilai.

Contoh pernyataan CASE :

```
CASE nilai OF
   100       : ShowMessage('Selamat Anda Lulus dengan Nilai SEMPURNA');
   80 .. 99 : ShowMessage('Selamat Anda Lulus dengan sangat baik');
   50 .. 79  : ShowMessage('Selamat Anda Lulus dengan cukup baik');
   Else
       ShowMessage('Nilai Kurang, Anda Tidak Lulus');
End;
```

Contoh pernyataan CASE :

```
CASE huruf OF
   'A','I','U','E','O': ShowMessage('Huruf Vokal');
   Else
       ShowMessage('Bukan Huruf Vokal');
End;
```

Ada beberapa hal yang harus diperhatikan ketika menggunakan CASE yaitu :

1. Pada bagian EkspresiNilai harus berupa data ordinal (Bilangan bulat, Karakter, atau Boolean)
2. Bagian EkspresiNilai boleh berupa range nilai
3. Pada bagian DaftarKemungkinan, tidak boleh terjadi sebuah nilai berada pada 2 atau lebih DaftarKemungkinan. Tidak boleh sebuah nilai disebut lebih dari 1 kali.
4. Bagian ELSE digunakan jika nilai tidak terpenuhi pada salah satu dari DaftarKemungkinan di atasnya.

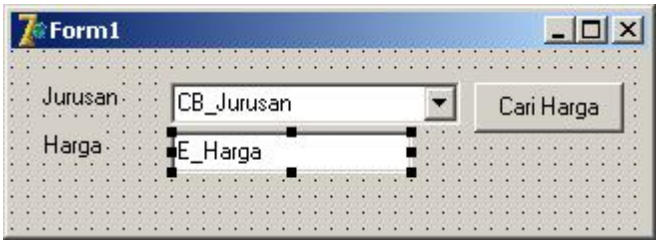
KOMPONEN YANG UMUM DIOLAH MENGGUNAKAN PERCABANGAN

Sebenarnya semua komponen yang ada dapat diolah menggunakan percabangan. Tetapi ada beberapa yang (bisa disebut pasti) menggunakan percabangan untuk mengolahnya., diantaranya adalah :

COMBOBOX

Combobox ( )digunakan untuk menampilkan daftar pilihan yang ditampilkan berbentuk kombinasi antara Listbox dan Edit. Pengguna dapat mengisi data dalam kotak Edit atau memilih sesuai dengan daftar yang ada. Daftar pilihan terdapat pada properti ITEMS. Dan pilihan yang dipilih oleh user dapat diakses melalui properti TEXT atau ITEMINDEX.

Contoh aplikasi menggunakan ComboBox



Kemudian tombol Cari Harga diisi dengan script berikut :

```
procedure TForm1.Tbl_Cari_HargaClick(Sender: TObject);
begin
     if CB_Jurusan.Text='Jakarta' then
         E_Harga.Text:='10000'
     else
     if CB_Jurusan.Text='Yogyakarta' then
         E_Harga.Text:='20000'
     else
     if CB_Jurusan.Text='Surabaya' then
         E_Harga.Text:='30000'
     else
         E_Harga.Text:='0';
end;
```

Jika dieksekusi akan menghasilkan tampilan seperti di bawah ini

RADIOBUTTON

RadioButton ( ) digunakan untuk menampilkan sebuah pilihan dimana user dapat mengganti nilainya dengan True atau False. Radiobutton digunakan untuk menampilkan sekumpulan pilihan dimana user hanya boleh memilih satu pilihan saja. Untuk mengakses apakah sebuah radio button dipilih oleh user, property yang digunakan adalah property CHECKED. Jika property CHECKED bernilai True berarti objek tersebut dipilih oleh user. Jika anda mempunyai 2 pilihan yang saling terpisah, maka groupkan tiap pilihan-pilihan tersebut dengan GroupBox.

Contoh aplikasi menggunakan RadioButton

Kemudian tombol Cari Harga diisi dengan script berikut :

```
procedure TForm1.Tbl_Cari_HargaClick(Sender: TObject);
begin
     if RB_Jakarta.Checked = True then
        E_Harga.Text:='10000'
```

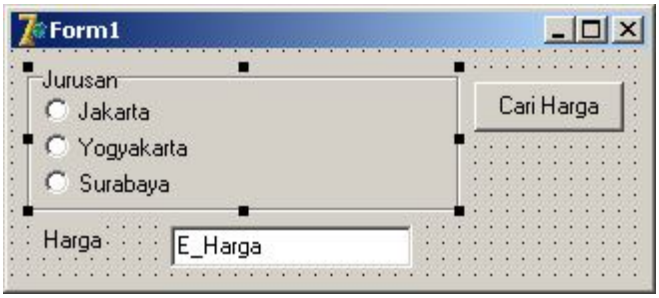
```
      else
      if RB_Yogyakarta.Checked then
         E_Harga.Text:='15000'
      else
      if RB_Surabaya.Checked then
         E_Harga.Text:='20000'
      else
         E_Harga.Text:='0';
end;
```

Jika dieksekusi akan menghasilkan tampilan seperti di bawah ini

RADIOGROUP

Radiogroup ( ) mempunyai fungsi yang sama dengan RadioButton. Perbedaannya adalah kalau suatu pilihan terdiri dari 4 pilihan maka kalau menggunakan RadioButton akan memerlukan 4 objek RadioButton tetapi jika menggunakan RadioGroup hanya akan membutuhkan sebuah RadioGroup saja. Untuk membuat isi pilihan pada RadioGroup, gunakan property ITEMS. Dan untuk mengakses pilihan mana yang dipilih oleh user, gunakan property ITEMINDEX. Jika itemindex bernilai 0 berarti pilihan pertama yang dipilih, jika itemindex bernilai 1 berarti pilihan kedua yang dipilih.

Contoh aplikasi menggunakan RadioGroup



Kemudian tombol Cari Harga diisi dengan script berikut :

```
procedure TForm1.Tbl_Cari_HargaClick(Sender: TObject);
begin
      if RG_Jurusan.ItemIndex = 0 then
         E_Harga.Text:='10000'
      else
      if RG_Jurusan.ItemIndex = 1 then
         E_Harga.Text:='15000'
      else
      if RG_Jurusan.ItemIndex = 2 then
         E_Harga.Text:='20000'
```

```
      else
         E_Harga.Text:='0';
end;
```

Atau jika anda menggunakan Case

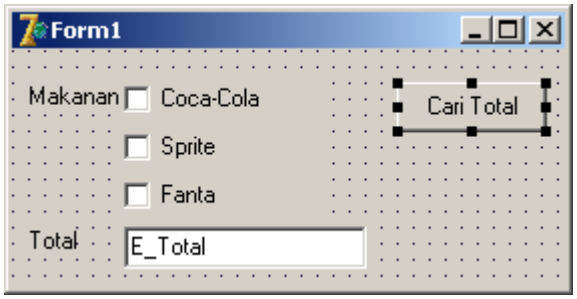
```
procedure TForm1.Tbl_Cari_HargaClick(Sender: TObject);
begin
      case RG_Jurusan.ItemIndex of
        0:E_Harga.Text:='10000';
        1:E_Harga.Text:='15000';
        2:E_Harga.Text:='20000';
        else
            E_Harga.Text:='0';
      end;
end;
```

Jika dieksekusi akan menghasilkan tampilan seperti di bawah ini

CHECKBOX

CheckBox ( ) digunakan untuk menampilkan sebuah pilihan dimana user dapat melakukan perubahan antara Ya/Tidak atau True/False. Checkbox digunakan untuk menampilkan sekumpulan pilihan dimana pengguna boleh memilih lebih dari satu pilihan. Untuk mengakses apakah sebuah CheckBox dipilih oleh user, property yang diakses adalah property CHECKED. Jika property Checked bernilai True, berarti Checkbox tersebut dipilih.

Contoh aplikasi menggunakan CheckBox



Kemudian tombol Cari Harga diisi dengan script berikut :

```
procedure TForm1.Tbl_Cari_TotalClick(Sender: TObject);
var
   Total:integer;
```

```
begin
     Total:=0;
     if CB_Cocacola.Checked=true then
         Total:=Total+2500;

     if CB_Sprite.Checked then
         Total:=Total+2000;

     if CB_Fanta.Checked then
         Total:=Total+2250;

     E_Total.Text:=Inttostr(Total);
end;
```

Jika dieksekusi akan menghasilkan tampilan seperti di bawah ini

PERCABANGAN DENGAN BANYAK KONDISI

Dalam kehidupan sehari-hari sangat dimungkinkan pengambilan keputusan berdasarkan kepada lebih dari 1 kondisi. Mungkin 2 kondisi, 3 kondisi atau lebih.

Untuk percabangan yang menggunakan banyak kondisi, haruslah menggunakan operator relasional yaitu or, xor, and.

Contoh Kasus :

Sebuah Perusahaan angkutan "Jalan Terus" memiliki ketentuan harga seperti berikut :

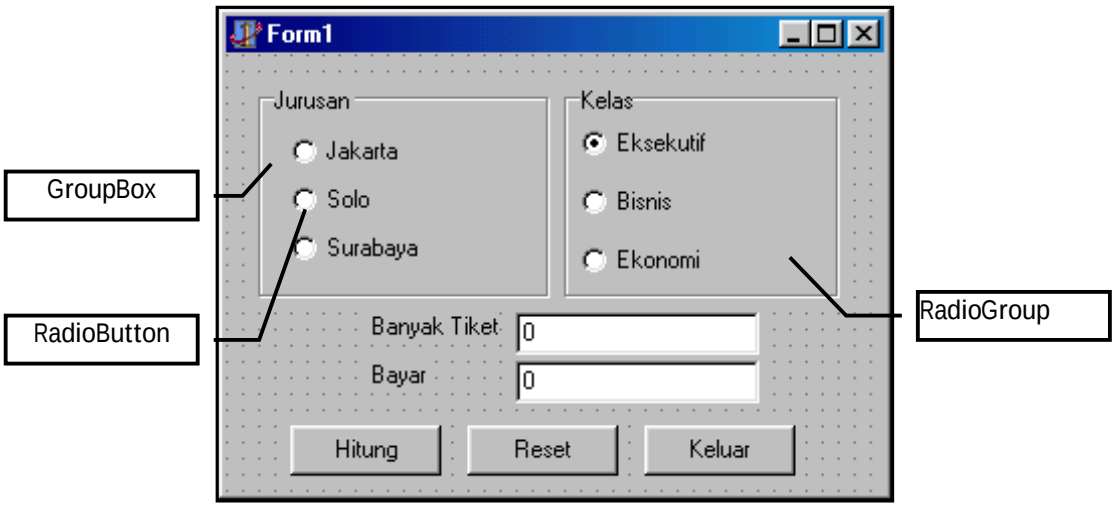
	Eksekutif	**Bisnis**	**Ekonomi**
Jakarta	70000	40000	10000
Solo	80000	50000	20000
Surabaya	90000	60000	30000

Karena ada masa promosi maka khusus untuk **Surabaya Ekonomi** atau **Solo Eksekutif** terdapat diskon 10%.

Dari kasus di atas, diketahui bahwa ada beberapa percabangan yang diperlukan, yaitu :

- Percabangan untuk menentukan harga tiket didasarkan kepada 2 kondisi yaitu Jurusan dan Kelas yang diambil.
- Percabangan untuk menentukan diskon yang hanya diberikan kepada jurusan dan kelas tertentu.

Penyelesaian untuk kasus di atas adalah :



Kemudian tombol Hitung diisi dengan script berikut :

```
procedure TForm1.THitungClick(Sender: TObject);
var
   jmltiket:integer;
   total,harga:real;
begin
     jmltiket:=StrToInt(ETiket.text);
     if (RBJakarta.Checked=true)and(RGKelas.ItemIndex=0) then
          harga:=70000
     else
     if (RBJakarta.Checked=true)and(RGKelas.ItemIndex=1) then
          harga:=40000
     else
     if (RBJakarta.Checked=true)and(RGKelas.ItemIndex=2) then
          harga:=10000
     else
     if (RBSolo.Checked=true) then
     begin
          if RGKelas.ItemIndex=0 then
               harga:=80000
          else
          if RGKelas.ItemIndex=1 then
               harga:=50000
          else
          if RGKelas.ItemIndex=2 then
               harga:=20000;
     end
     else
     if (RBSurabaya.Checked=true) then
     begin
          case RGKelas.ItemIndex of
               0:Harga:=90000;
               1:Harga:=60000;
               2:Harga:=30000;
          end;
```

```
      end;
      total:=harga*jmltiket;
      if ((RBSurabaya.checked=true) and (RGKelas.ItemIndex=2))
           or
         ((RBSolo.Checked=true) and (RGKelas.ItemIndex=0)) then
           total:=0.9*total;
      ETotal.Text:=CurrToStr(total);
end;
```

VALIDASI DATA DENGAN STATEMENT IF

Validasi data adalah suatu proses yang dijalankan agar semua data yang dimasukan sesuai dengan aturan yang berlaku. Validasi data sangat penting untuk menjaga jangan sampai komputer menampilkan keluaran/output yang salah dikarenakan proses input yang salah.

Validasi data pasti melibatkan pernyataan IF, karena akan melibatkan penentuan keputusan apakah data yang dimasukan merupakan data yang benar atau salah.

Banyak fungsi-fungsi yang digunakan dalam validasi data, diantaranya :

- TrimLeft, digunakan untuk menghapus spasi yang mengawali (sebelah kiri) suatu string.
- TrimRight, digunakan untuk menghapus spasi yang ada di akhir (sebelah kanan) suatu string.
- Trim, digunakan untuk menghapus spasi yang ada di awal dan diakhir suatu string.
- Copy, digunakan untuk mengambil suatu string dari string lain.
- Length, digunakan untuk memeriksa banyak karakter yang ada pada suatu string.
- Uppercase, digunakan untuk mengkonversi semua karakter yang ada dalam suatu string menjadi huruf kapital.
- Lowercase, digunakan untuk mengkonversi semua karakter yang ada dalam suatu string menjadi huruf kecil.
- TryStrToInt, digunakan untuk mengkonversi string menjadi Integer. Fungsi ini akan menghasilkan nilai true jika semua string dapat dikonversi. Jika ada string yang salah maka akan menghasilkan nilai false.
- TryStrToFloat, digunakan untuk mengkonversi string menjadi Float. Fungsi ini akan menghasilkan nilai true jika semua string dapat dikonversi. Jika ada string yang salah maka akan menghasilkan nilai false.
- TryStrToCurr, digunakan untuk mengkonversi string menjadi Currency. Fungsi ini akan menghasilkan nilai true jika semua string dapat dikonversi. Jika ada string yang salah maka akan menghasilkan nilai false.

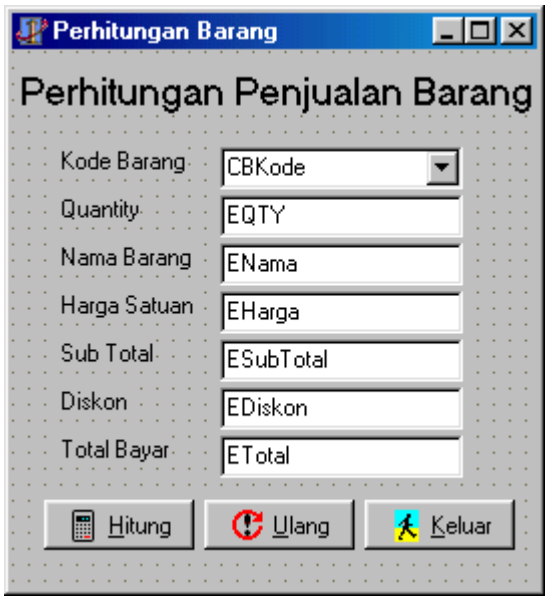
Contoh kasus :

Sebuah toko mempunyai tabel harga dan diskon seperti berikut :

Kode	Nama Barang	Harga Barang
A01	Speaker	50000
B02	Mouse	25000
C03	Harddisk	750000
D04	Mouse Pad	5000

SUB TOTAL	Diskon
>=100000	15%
>=50000	10%
>=25000	5%
<25000	0%

Tampilan yang diinginkan adalah :



Ketentuan :

Data yang diinputkan hanya kode barang dan quantity. Quantity pembelian tidak boleh lebih dari 10. Data yang dimasukan harus divalidasi.

Solusi untuk tombol Hitung adalah script berikut :

```
procedure TForm1.Tbl_HitungClick(Sender: TObject);
var
   kode, nama:string;
   harga, diskon, subtotal,total:currency;
   qty:integer;
begin
     kode:=cbkode.Text;
     kode:=trim(kode);
     // Di baris ini, kode sudah pasti tidak ada spasi di awal dan diakhir

     kode:=uppercase(kode);
     //               kode sudah pasti capital (tidak mempunyai huruf kecil)

     kode:=copy(kode,1,3);
     // Di baris ini, kode pasti 3 huruf atau kurang

     if length(kode)<>3 then
     begin
          showmessage('Kode harus 3 karakter');
          cbkode.SetFocus;
          exit;
     end;
     // Di baris ini, kode sudah pasti tidak ada spasi di awal dan diakhir
     //               kode sudah pasti capital (tidak mempunyai huruf kecil)
     //               kode pasti 3 huruf tidak mungkin kurang

     if kode='A01' then
     begin
          harga:=50000;
          nama:='Speaker';
     end
     else
     if kode='B02' then
     begin
          harga:=25000;
          nama:='Mouse';
     end
     else
     if kode='C03' then
     begin
          harga:=750000;
          nama:='Hardisk';
     end
     else
     if kode='D04' then
     begin
          harga:=5000;
          nama:='Mouse Pad';
     end
     else  // Jika kode tidak terdaftar
     begin
          showmessage('Kode tidak dikenal');
          exit;
     end;

     if TryStrToInt(EQty.Text,qty)=false then
     begin
          showmessage('Quantity harus angka.');
          exit;
     end;
     // Di baris ini, Qty pasti bilangan bulat tetapi masih mungkin diatas 10

     if (qty<1) or (qty>10) then
     begin
          showmessage('Hanya boleh beli 1 s/d 10 saja');
          exit;
     end;
     // Di baris ini, Qty pasti bilangan bulat dan pasti berisi 1 s/d 10
```

```
      ENama.text:=nama;
      EHarga.text:=CurrToSTr(harga);

      Subtotal:=  harga * qty;
      ESubTotal.Text:=CurrToStr(subtotal);

      if subtotal>=100000 then diskon:=0.15 * subtotal else
      if subtotal>=50000 then diskon:=0.1 * subtotal else
      if subtotal>=25000 then diskon:=0.05 * subtotal else
         diskon:=0;
      EDiskon.text:=CurrToStr(diskon);

      Total:=Subtotal-Diskon;
      ETotal.text:=CurrToStr(Total);
  end;
```

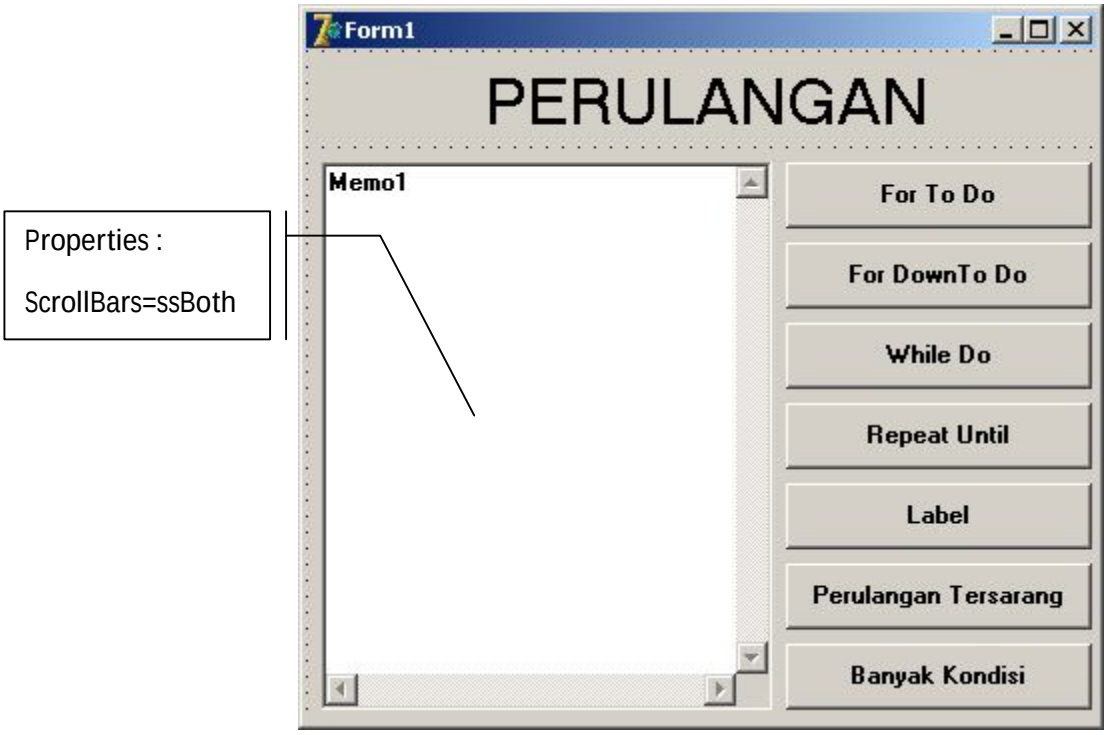
BAB VII
PERULANGAN

PENDAHULUAN

Salah satu kegunaan dari komputer adalah untuk melakukan proses yang berulang. Bagi manusia proses berulang bisa menjadi satu hal yang melelahkan, sehingga bisa membuat proses menghasilkan kesalahan karena ketidakkonsentrasian atau kelalaian. Oleh karena itu di semua bahasa pemrograman pasti memiliki mekanisme untuk melakukan perulangan. Demikian dengan juga dengan Delphi. Di Delphi perulangan dapat dilakukan dengan beberapa cara, diantaranya :

- Perulangan menggunakan Counter/Pencacah sebagai kendali perulangannya. Perulangan dengan cara ini dapat dilakukan dengan menggunakan pernyataan : **For To Do** atau **For DownTo Do.**
- Perulangan menggunakan Kondisi Boolean sebagai kendali perulangannya. Perulangan dengan cara ini dilakukan dengan menggunakan pernyataan : **While Do**, atau **Repeat Until**.
- Perulangan menggunakan Label. (cara ini tidak disarankan untuk digunakan. Label akan diterangkan sekedar untuk pembelajaran saja).

Untuk lebih memahami perulangan, buatlah form seperti di bawah ini yang akan digunakan untuk mendemonstrasikan berbagai jenis perulangan.



Komponen Memo ( ) umumnya digunakan untuk pemasukan data string yang bisa menampung data dalam bentuk beberapa paragraf (banyak baris). Komponen ini dapat kita samakan dengan Notepad. Isi dari komponen ini ada di properti **Lines**.

Method-method yang banyak dipakai yang dimiliki oleh memo adalah :

– **Clear**, gunanya untuk mengosongkan memo.
Contoh :

Memo1.Clear;

– **Lines.Add**, gunanya untuk memasukan/menambahkan baris di Memo.
Inputan untuk method add harus berupa string. Jika yang ingin ditampilkan berupa angka, maka harus dikonversikan dulu.

Contoh :

Memo1.Lines.Add('Text Yang Ditambahkan');

A:=100;

Memo1.Lines.Add('Nilai A adalah : '+inttostr(A));

– **Lines.Delete**, gunanya untuk menghapus suatu baris.
Contoh :

Memo1.Lines.Delete(0); // Menghapus Baris Ke-1

Properti yang biasa dimodifikasi diantaranya :
- Lines, properties yang menyimpan baris-baris yang ada dalam memo
- Alignment, digunakan untuk menentukan alignment baris (Left, Right, Center)
- ScrollBars, digunakan untuk menentukan scrollbar mana yang akan ditampilkan (None, Vertikal, Horizontal, Both/Keduanya)

PERULANGAN FOR TO DO

Perulangan For To Do merupakan perulangan yang menggunakan kendali perulangannya berupa counter/pencacah. Counter akan menjalankan perulangan dimulai dari nilai awal yang diberikan sampai counter mencapai nilai akhirnya. Syarat terjadinya perulangan adalah nilai awal counter harus lebih kecil atau sama dengan nilai akhir counter.

Bentuk dasar dari perulangan For To Do adalah sebagai berikut :

```
For counter := nilai awal To Nilai akhir Do
Begin
        Perintah;
        Perintah;
End;
```

Perulangan ini akan berulang selama nilai variabel counter masih lebih kecil atau sama dengan nilai akhir.
Contoh Program yang ditulis dalam event OnClick dari tombol **For To Do**

```
procedure TForm1.BitBtn1Click(Sender: TObject);
var
  i:integer;
  c:char;
begin
   Memo1.Clear;
   for i:=1 to 10 do
      Memo1.Lines.Add(IntToStr(i));
   for c:='A' to 'Z' do
      Memo1.Lines.Add(c);
end;
```

PERULANGAN FOR DOWNTO DO

Perulangan ini sama seperti perulangan dengan menggunakan For To Do tetapi arah isi counternya berkurang (mundur/turun).

Bentuk umum dari perulangan ini adalah :

```
For counter := nilaiawal DownTo Nilaiakhir Do
Begin
     Perintah;
     Perintah;
End;
```

Perulangan ini akan berulang selama nilai variabel counter masih lebih besar atau sama dengan nilai akhir. Nilai awal suatu counter harus lebih besar atau sama dengan daripada nilai akhirnya.
Contoh Program yang ditulis dalam event OnClick dari tombol **For To Do**

```
procedure TForm1.BitBtn2Click(Sender: TObject);
var
   i:integer;
   c:char;
begin
     Memo1.Clear;
     for i:=10 downto 1 do
          Memo1.Lines.Add(IntToStr(i));
     for c:='Z' downto 'A' do
          Memo1.Lines.Add(c);
end;
```

PERULANGAN REPEAT UNTIL

Perulangan ini bekerja sampai kondisi yang diinginkan tercapai. Perulangan ini bisa digunakan sebagai perulangan yang menggunakan banyak kondisi keluar (multi condition loop).

Bentuk umum :

```
Repeat
   Perintah;
   Perintah;
Until Kondisi;
```

Contoh Program yang ditulis dalam event OnClick dari tombol **Repeat Until**

```
procedure TForm1.BitBtn3Click(Sender: TObject);
var
  i:real;
begin
    Memo1.Clear;
    i:=0;
    repeat
        i:=i+0.5;
        Memo1.Lines.Add(FloatToStr(i));
    until  i=10;
end;
```

Perulangan ini berulang sampai kondisi yang ada setelah until mempunyai nilai true. Jadi repeat until berjalan selama kondisi salah sampai kondisi benar.

Pengecekan kondisi berada di bagian bawah yaitu setelah until sehingga semua program yang memakai perulangan ini akan melakukan minimal 1 kali proses perulangan.

PERULANGAN WHILE DO

Perulangan ini mirip dengan perulangan Repeat Until tetapi pengecekannya berada di awal sebelum melakukan proses yang berulang. Perulangan ini berjalan selama kondisi pengecekan bernilai true. Jika kondisi telah mempunyai nilai false maka perulangan tidak dilakukan lagi.

Bentuk umum dari While Do adalah seperti berikut :

```
While kondisi do
Begin
   Perintah;
   Perintah;
End;
```

Contoh program while do yang ditulis pada event onclick dari tombol While Do.

```
procedure TForm1.BitBtn4Click(Sender: TObject);
var
  i:real;
begin
    Memo1.Clear;
    i:=0;
```

```
    while i<10 do
    begin
        i:=i+0.5;
        Memo1.Lines.Add(FloatToStr(i));
    end;
end;
```

PERULANGAN LABEL

Label adalah suatu perintah untuk memandai suatu baris program. Untuk pindah ke posisi yang ditandai oleh label, maka digunakan perintah Goto.

Untuk lebih jelasnya lihatlah program berikut yang ditulis dalam event OnClick dari tombol Label.

```
procedure TForm1.BitBtn5Click(Sender: TObject);
label
    A;
var
    i:integer;
begin
    Memo1.Clear;
    i:=0;
    A:
        i:=i+1;
        Memo1.Lines.Add(IntToStr(i));
        if i<10 then
            Goto A;

End;
```

PERULANGAN TERSARANG

Perulangan tersarang (nested loop) adalah suatu perulangan dimana di dalam perulangan tersebut ada perulangan lain. Perulangan yang di ada di bagian terdalam, maka akan dikerjakan dulu sampai selesai baru mengerjakan perulangan yang ada diluarnya.

Nested loop bisa merupakan kombinasi antara For To Do, For DownTo Do, Repeat Until, While Do.

Contoh Program

```
procedure TForm1.BitBtn6Click(Sender: TObject);
var
    i,j:integer;
begin
      Memo1.Clear;
      for i:=1 to 5 do
      begin
            j:=0;
            while j<5 do
            begin
                  j:=j+1;
```

```
                Memo1.Lines.Add(IntToStr(i)+' X '+IntToStr(j)
                                +' = '+IntToStr(i*j));
            end;
            Memo1.Lines.Add('-------------------');
        end;
end;
```

PERULANGAN BANYAK KONDISI

Perulangan dengan banyak kondisi adalah suatu perulangan dimana berjalannya suatu perulangan tidak hanya berdasarkan suatu kondisi saja tetapi bisa lebih dari satu kondisi. Operator yang bisa digunakan biasanya seperti And, Or, Not dan lain-lain.

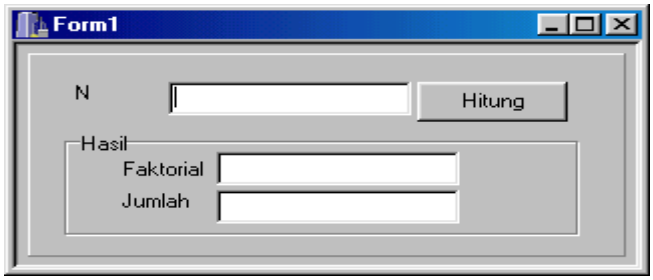
Contoh Program :

```
procedure TForm1.BitBtn7Click(Sender: TObject);
var
  i:integer;
  total:integer;
begin
    Memo1.Clear;
    total:=0;
    i:=0;
    repeat
        i:=i+1;
        Total:=Total+i;
        Memo1.Lines.Add('Isi I  = '+IntToStr(i));
        Memo1.Lines.Add('Isi Total = '+IntToStr(Total));
    until (total>150)or(i>25);

end;
```

LATIHAN - LATIHAN

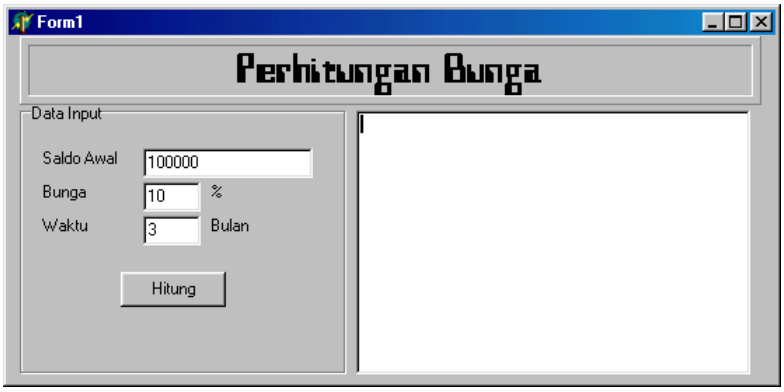
1. Buat suatu program untuk menentukan faktorial dan jumlah deret sampai ke N. Buat form sebagai berikut :



2. Buat suatu program untuk menghitung saldo akhir dari suatu tabungan dengan bunga dan jangka waktu tertentu :
   Contoh hasil :

   Saldo : 100000
   Bunga : 10
   Waktu : 3

   Ketika user menekan tombol Hitung, maka akan muncul hasil seperti berikut di memonya.
   Saldo Bulan 1 = Rp. 110000
   Saldo Bulan 2 = Rp. 121000
   Saldo Bulan 3 = Rp. 133100



3. Mirip dengan program ke-dua hanya input yang diberikan adalah : Saldo awal, bunga dan saldo akhir. Dari ketiga input tersebut maka akan didapatkan lamanya jangka waktu yang diinginkan agar saldo mencapai saldo akhir.
   Contoh :

   Saldo Awal : 100000
   Bunga : 10
   Saldo Akhir : 150000

   Maka hasil pada memonya adalah :

   Saldo Bulan 1 = Rp. 110,000
   Saldo Bulan 2 = Rp. 121,000
   Saldo Bulan 3 = Rp. 133,100
   Saldo Bulan 4 = Rp. 146,410
   Saldo Bulan 5 = Rp. 161,051

   Form yang diinginkan :